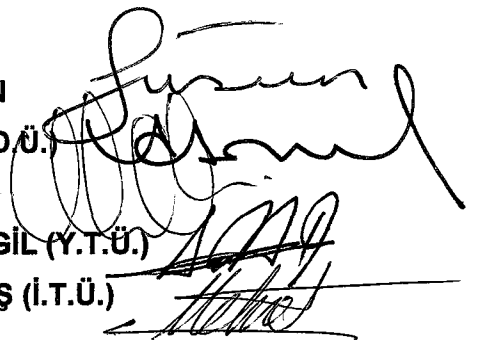
İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

LOJİSTİK SİSTEMLERİNİN YAPAY SİNİR AĞLARI İLE MODELLENMESİ, GERÇEKLENMESİ VE KONTROLÜ

DOKTORA TEZİ
Y. Müh. Murat ERMİŞ
(507982010)

Tezin Enstitüye Verildiği Tarih : 12 Mart 2004
Tezin Savunulduğu Tarih : 10 Mart 2005

Tez Danışmanı	Prof.Dr. Füsun ÜLENGİN
Diğer Jüri Üyeleri	Prof.Dr. Ataç SOYSAL (D.Ü.)
	Prof.Dr. İlhan OR (B.Ü.)
	Prof.Dr. Hüseyin BAŞLIGİL (Y.T.Ü.)
	Doç.Dr. Mehmet TANYAŞ (İ.T.Ü.)



MART 2005

İÇİNDEKİLER

KISALTMALAR

KTP	: Konum-Tahsis Problemleri
İTZ	: İşbirlikçi Tedarik Zinciri
KEP	: Kombinatorik Eniyileme Problemi
YA	: Yerel Arama
TB	: Tavlama Benzetimi
GA	: Genetik Algoritmalar
TGA	: Titreşimli Genetik Algoritma
GETTZY	: Gezgin Etmen Temelli Tedarik Zinciri Yönetimi
YSA	: Yapay Sinir Ağı
İKP	: İşletme Kaynakları Planlaması
TZY	: Tedarik Zinciri Yönetimi
BT	: Bilgi Teknolojileri
EDI	: Electronic Data Interchange
YA/EM	: Yöneylem Araştırması ve Endüstri Mühendisliği
YYSA	: Yasaklı Yapay Sinir Ağı
H-T	: Hopfield - Tank
SOM	: Kendini Örgütleyen Ağlar
FSCL	: Frekans-Duyarlı Rekabetçi Öğrenme Algoritması
EKKO	: En Küçük Kareler Ortalaması
HKO	: Hata Kareleri Ortalaması
İE	: İşlem Elemanı, Nöron
ÇKİB	: Çok-Katmanlı İleri Beslemeli
GY	: Geriye Yayılım
TSKİB	: Tek Saklı Katmanlı İleri Beslemeli
MOM	: Moment ile Öğrenme
RTF	: Radyal Temelli Fonksiyonlar
LBG	: Grup Olarak Öğrenme
PCA	: En Önemli Bileşen Analizi
VN	: Vektör Nicelendirme
ÖVN	: Öğrenen Vektör Nicelendirme
CAM	: Content - Addressable Memory
VLSI	: Very Large Scale Integration
GSP	: Gezgin Satıcı Problemi
SÖKT	: Sürekli Örten Konum-Tahsis
HAKA	: Hiyerarşik Artan Kümeleme Algoritmasını
ÇYA	: Çift Yönlü Alfa
SÇYA	: Sinüsoidal Çift Yönlü Alfa
SUS	: Evrensel Örnekleme
USAHLP	: Uncapacitated Single Allocation Hub Location Problem
USApHMP	: Uncapacitated Single Allocation p-Hub Median Problem
CMAHLP	: Capacitated Multiple Allocation Hub Location Problem
UMApHMP	: Uncapacitated Multiple Allocation Hub Location Problem
HN	: Hopfield Ağı
HTTN	: Hopfield Tepe Tırmanan Ağ
JVM	: Java Virtual Machine
RMI	: Uzaktan Metot Çağırma

İGE	: İşletme düzeyi Gezgin Etmen
TGE	: Tedarikçi düzeyi Gezgin Etmen
GUI	: Kullanıcı Grafik Arabirimi
FIFO	: İlk Gelen İlk Çıkar

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

SEMBOL LİSTESİ

J	: Performans ölçütü (hata kareleri ortalaması)
∇	: Gradyan
η, η_i, η_f	: Öğrenme oranı
ε	: Tahmin hatası
w	: Ağırlık değerleri
b	: Önyargı (yanlılık) değeri
x	: Aktivasyon değeri
y	: Çıktı değeri
d	: Hedef çıktı değeri
δ	: Yerel hata
η_0	: Başlangıç adım büyüklüğü
n_0	: Yineleme sayısı
α	: Moment sabiti
H	: Hesiyen matrisi
s	: Çarpıcılık değeri
F	: Uygunluk değeri
V	: Voronoi bölgesi
A	: Ağdaki işlem birimleri kümesi
w_c	: Referans vektörler kümesi
Λ	: Komşuluk fonksiyonu
I_i	: Harici girdi
λ	: Aktivasyon fonksiyonunun eğimi
u_i	: Aktivasyon düzeyi
g_i	: Aktivasyon fonksiyonu
E	: Hopfield enerji fonksiyonu
E_a	: Ayrık Hopfield enerji fonksiyonu
E_s	: Sürekli Hopfield enerji fonksiyonu
W	: Ağırlık matrisi
f	: Amaç fonksiyonu
γ, δ, α	: Düzeltilmiş Hopfield enerji fonksiyonu parametreleri
G	: Genlik
P	: Salınım periyodu
MA	: Ana genlik
AF_0, AF_k	: Genetik sürecin başlangıç ve içinde bulunulan adımındaki ortalama uygunluk değerleri
P_c	: Çaprazlama oranı
P_m	: Mutasyon oranı
$\otimes$	: Kronecker çarpımı

LOJİSTİK SİSTEMLERİNİN YAPAY SİNİR AĞLARI İLE MODELLENMESİ, GERÇEKLENMESİ VE KONTROLÜ

ÖZET

Günümüz Lojistik Sistemlerinin, müşteri odaklı olarak, malzeme akışına göre şekillendirilmesi işlemi görünürde çok açık ve anlaşılırdır. Ancak, dinamik ve rekabetçi bir ortamda sanal şirket, elektronik ticaret gibi teknolojik gelişmelere ve yeni buluşlara hızla ayak uydurabilecek, farklı uygulama süreçlerinde ekonomi, biyoloji, sosyoloji ve psikoloji gibi alanlardan da teorik anlamda faydalanabilecek bütünleşik bir sistemi yönetmek oldukça karmaşıktır. Parçalara ayırma, bütünleştirme ve geri besleme mekanizmaları yoluyla sistem karmaşıklığı ile başa çıkmada, çeşitli avantajları nedeniyle *hiyerarşik karar verme* mekanizması kullanılmaktadır. Özellikle gerçek zamanlı, hata toleransını minimize eden, değişen pazar koşullarına hızlı uyum sağlayabilen yeni ve farklı yaklaşımlar üzerinde araştırmalar yapılmaktadır.

Önceleri, tedarik zincirinde pazarlama, dağıtım, planlama, imalat ve satın alma işlemleri birbirinden bağımsız olarak gerçekleştirilirdi. Bu bölümler, genellikle birbirleriyle çatışan bireysel amaçlara sahiptirler. Pazarlamanın yüksek müşteri düzeyi ve maksimum satış hedefi, imalat ve dağıtım bölümlerinin amaçları ile çatışır. Bir çok imalat işlemi, düşük maliyetlerin envanter düzeyleri ve dağıtım yeteneklerine olan etkisini çok az göz önünde bulundurarak, üretilen işi enbüyükleyecek biçimde tasarlanır. Satın alma anlaşmaları, genellikle geçmiş satın alma örüntülerinin ötesinde çok az bilgi ile müzakere edilir. Bu yüzden, bu farklı işlevlerin bütünleştirilebileceği bir mekanizmaya gereksinim vardır.

Bütünleşik lojistik sistemlerinden beklenen özellikleri karşılayabilmek için klasik yaklaşımların yerine; işletmenin farklı işlevsel alanlarını, tedarik zincirinin koordinasyonu altında bütünleştirici bir yöntemle ele alan, alt bileşenlerin kendilerine has perspektiflere ve güdüleyicilere sahip olmalarından dolayı tek tip bir karar verme yapısına zorlanmadıkları bir *dağıtılmış karar verme mekanizması* kullanılabilir. Bu yaklaşım ile gevşek bağlı (*loosely-coupled*) bilgisayarların kullanımı sayesinde

lojistik sisteminin bilgi yönetimi köklü olarak basitleştirilebilir. Paralel işleme ve adaptif öğrenme ile daha kararlı ve güçlü yapıların, daha hızlı olarak gerçekleştirilmesi yeteneği kazanılabilir.

Bütünleşik lojistik sistemlerinden beklenenlere uyumlu bir çözüm sağlamak amacıyla; seçilen tez konusunun kapsamında, karmaşık bir yapıya sahip, dağıtılmış karar verme ve paralel işlem yapma süreçlerini gerektiren, maliyetlerde ve/veya taleplerde belirsizliklerin söz konusu olduğu, karar mekanizmaların ve analizlerin bir veri tabanı yönetim sistemi yardımıyla gerçekleştirildiği bir lojistik sistemi için, dağıtılmış bir sistem modelleme yaklaşımının uygulandığı, öğrenme ve çıkarım mekanizmalarında bir yapay sinir ağı altyapısının kullanıldığı sistem tasarlanıp gerçeklenmiştir. Bu yeni altyapı sistemi ve onu oluşturan ve koordineli olarak eniyilenmeye çalışılan modeller ile lojistik sisteminin operasyonel kararlarının kendiliğinden alınması, stratejik kararlarda üst düzey yöneticilere rapor verilmesi ve yol gösterilmesi, ayrıca performans analizleri yaparak kontrol edilmesi ve ileriye dönük projeksiyonların yapılması mümkün olacaktır.

Özellikle lojistik yönetimi gibi dinamik sistemlerde, çok hızlı bir biçimde çözüm elde ederek karar verici için alternatifler oluşturmak oldukça önemlidir. Bu tip sistemlerde göreceli olarak kısa zamanda elde edilen iyi bir çözüm, uzun sürede bulunabilen optimal çözümden daha faydalıdır. Bu yüzden daha hızlı ve optimale yakın çözüm veren sezgisel yaklaşımlar üzerinde çalışılmaktadır.

Bu tip sezgisel yöntemlerden bazıları Yerel Arama, Tavlama Benzetimi, Genetik Algoritmalar ve Yapay Sinir Ağları'dır. Doksanlı yıllardan itibaren Yapay Sinir Ağlarının Yöneylem Araştırmasının inceleme sahalarına giren öngörüleme, modelleme, araç rotalama, kümeleme, sınıflandırma gibi bir çok alandaki problemleri çözmede kullanılmaya başlandığı gözlemlenmektedir. Sinir ağları yapay zeka ve beynin modellenmesi alanlarından geliştirilse de, aslında regresyon ve benzer yöntemlerde olduğu gibi bağımlı ve bağımsız değişkenler arasındaki ilişkiyi öğrenen bir fonksiyon tahmin aracından başka bir şey değildir. Sinir ağları ile istatistiksel yöntemler arasındaki temel farklılık, sinir ağlarının istatistiksel dağılımla veya verilerin özellikleri ile ilgili herhangi bir varsayım yapmamasıdır (parametrik değil). Bir diğer önemli özelliği de doğrusal olmayan bir tahmin yöntemi olduğundan karmaşık verilerin modellenmesinde daha uygundur.

Bu tez çalışmasında, lojistik sistemlerinin modellenmesinde ve eniyilenmesinde yapay sinir ağlarının kullanım potansiyelini değerlendirmek, gerçek zamanlı bir

sistemde kullanılabilecek etmen temelli ve sinir ağını kullanan bir karar destek sistemi tasarlayıp gerçeklemek hedeflenmiştir. Bu amaçla, lojistik sistemlerin matematiksel modelleri kurularak eniyilenmesinin gerektiği alt bileşenlerinde (öngörüleme, tedarikçi seçimi, depolama, fiziksel dağıtım, vs.) kullanılabilecek yapay sinir ağı modelleri olan çok katmanlı yapay sinir ağları, kendini örgütleyen ağlar ve Hopfield ağları bu tez çalışmasında detaylı olarak incelenmiştir.

İlk olarak çok geniş bir alanda yaygın olarak kullanılan çok katmanlı sinir ağları incelenmiş, kullanımı sırasında sıklıkla karşılaşılan problemler ve çözüm yöntemleri üzerinde durulmuş, performans iyileştirici değişiklikler ele alınmış ve Genetik Algoritmalar ile melez olarak kullanımı da değerlendirilmiştir.

Kombinatorik eniyileme problemlerini çözmede kullanılan iki temel yaklaşım; Hopfield-Tank sinir ağları ve Kohonen'in kendini düzenleyen ağlarıdır. Ancak, her iki yaklaşımın da bazı kusurları vardır ve farklı problem tiplerinin çözümünü bulmada gösterdikleri başarı değişkendir. Kohonen ağları öklit olmayan problemlerde genelleştirilememe, Hopfield ağları ise genellikle olursuz çözüm üretme ve en iyi çözümü elde etmek için çok uzun parametre ayarlamaları gerektirme gibi dezavantajlara sahip olmalarına rağmen; tavlama, melez modeller, vb. yaklaşımlarla özellikle ikinci dereceden programlamada oldukça iyi sonuçlar elde edilmiştir.

Özellikle çok büyük miktardaki bilgileri sürekli olarak sınıflandırabilmek, sınıflandırmayı güncelleyebilmek lojistik sistemlerin performansı açısından önemli olduğundan, eğitme kümesine gereksinim duymadan veriler arasındaki ilişkileri ortaya çıkaran kendi kendini örgütleyen ağlar ele alınmış ve tedarikçilerin sınıflandırılması ile sürekli örten konum-tahsis problemlerine uygulanmıştır.

Hopfield sinir ağlarının kombinatorik eniyileme problemlerindeki başarısı dikkate alınarak özellikle ağın performansını iyileştirici düzenlemeler üzerinde durulmuştur. Bu amaçla Gee ve Smith'in çalışmaları temel alınarak çözüm kalitesini artırmaya dönük ağın içsel dinamikleri değiştirilmiş ve merkez üsler problemini çözmede kullanılmıştır. Yapay sinir ağı modelleri literatürde yaygın olarak kullanılan gerçek test verileri kullanılarak diğer tekniklerle çözüm kalitesi ve tutarlılığı açısından karşılaştırılmıştır.

Konum – Tahsis problemlerini (KTP), özel olarak sürekli örten, p-medyan ve merkez üs modellerini çözmede yapay sinir ağlarının nasıl kullanılabileceği irdelenmiş, kendini örgütleyen ağlar ile yinelemeli mimariye sahip bir yapay sinir ağı olan

Hopfield -Tank yapay sinir ağı kullanılarak çözülmüştür. Ayrıca, p-medyan ve sürekli örten KTP problemlerini çözmek için yeni bir yaklaşım olarak *Titreşimli Genetik Algoritma* yaklaşımı önerilmiştir.

Önerilen yöntemlerin etkinliğini test etmek için, literatürde sıklıkla kullanılan test veri setleri kullanılmış, Yapay Sinir Ağı temelli yöntemler için elde edilen sonuçlar ile literatürde yer alan en iyi çözümler ve Tavlama Benzetimi ve Genetik Algoritmalar gibi diğer sezgisel yaklaşımlardan elde edilen sonuçlarla karşılaştırılmıştır. Özellikle konum-tahsis problemleri için çarpıcı sonuçlar elde edilmiş ve yine önerilen Titreşimli Genetik Algoritma yaklaşımı da çok iyi sonuçlar vermiştir. Ayrıca, önerilen yapay sinir ağı modelleri üretim yapan uluslararası bir firmanın lojistik verileri kullanılarak test edilmiş, böylelikle gerçek yaşam sistemlerindeki uygulama başarısı da görülmeye çalışılmıştır.

Son olarak, bütünleşik tedarik zinciri yönetimi sistemleri için gezgin etmen mimarisini temel alan, ana modül ile alt bileşenlerinin koordineli olarak hareket etmesi nedeniyle operasyonel kararların bilgi sistemi tarafından otomatik olarak alındığı, sistemin çatısının değişik lojistik sistemi tiplerine kolaylıkla genişletilebilecek yeterlikte genel olduğu, talep tahmini ile tedarikçilerin sınıflandırılması süreçlerinde yapay sinir ağı modellerini kullanan bir karar destek sistemi Java platformunda tasarlanmış ve gerçeklenmiştir.

MODELING, IMPLEMENTATION AND CONTROL OF LOGISTICS SYSTEMS USING ARTIFICIAL NEURAL NETWORKS

SUMMARY

Although designing current logistics systems with a customer-oriented perspective and according to the flow of material seems quite clear and understandable, management of an integrated system which will make theoretical use of economics, biology, sociology, and psychology in different application processes is fairly complicated. Because of many advantages, *hierarchical decision making* mechanism is used to cope with system complexity by using the mechanisms of dividing into pieces, integration, and feed-back. There has been some researches especially on new and different approaches, which are real time, minimize error tolerance and can adapt to the changing market conditions.

Traditionally, marketing, distribution, planning, manufacturing, and the purchasing organizations along the supply chain operated independently. These organizations have their own objectives and these are often conflicting. Marketing's objective of high customer service and maximum sales conflict with manufacturing and distribution goals. Many manufacturing operations are designed to maximize throughput and minimize costs with little consideration for the impact on inventory level and distribution capabilities. Purchasing contracts are often negotiated with very little information beyond historical buying patterns. Clearly, there is a need for a mechanism through which these different functions can be integrated together.

In order to satisfy the issues that are expected from integrated logistics systems, instead of traditional approaches, a distributed decision making mechanism which considers the different functional fields of enterprise with an integration approach under the coordination of supply chain can be used. Because of the special perspectives and motivators of components, these mechanisms are not compelled to the uniform decision making structure. Using this approach, data management of logistics system might be simplified thoroughly by making use of loosely-coupled

computers. Also, the use of parallel processing and adaptive learning will facilitate building stable and stronger structures quicker.

In this thesis, the aim is to find an alternative solution to the issues that are expected from integrated logistics systems. A complex logistics system requires distributed decision making and parallel processing. The decision mechanism and analysis are supported by a data base management system and a neural network based framework is designed and implemented. By using this novel framework, it will give reports to the managers in strategical decisions in order to achieve and to control performance analysis and to make the projections about future.

In many real-world situations such as logistics system management, it is important that a solution be attained fairly quickly, so that it can be considered as a real option in the decision process. Under such circumstances, a good answer computed relatively quickly is more useful than an optimal solution which takes an unreasonable amount of time or which is even infeasible to calculate. For these reasons, the importance of heuristics which can provide very rapid, near optimal solutions is understood.

Some of such heuristics are the Local Search, Simulated Annealing, Genetic Algorithms and Artificial Neural Networks. Over the last decade, we have seen a rapid acceptance of Artificial Neural Networks for solving a wide range of Operations Research such as forecasting, modeling, clustering and classification. While neural networks have developed from the field of artificial intelligence and brain modeling, they are nothing more than function approximation tools which learn the relationship between independent variables and dependent variables, much like regression or other similar approaches. The principal difference between neural networks and statistical approaches is that neural networks make no assumptions about the statistical distribution or properties of the data (nonparametric). Neural networks are also an inherently nonlinear approach giving them much accuracy when modeling complex data patterns.

In this thesis, our objective was to exploit the features of artificial neural networks in modeling and optimization of logistics systems, to design and implement a mobile agent based decision support system that uses artificial neural networks in its subsystems (e.g., forecasting, vendor selection, warehousing, physical distribution, etc.) which can be used in a real time system. For this reason, in this thesis artificial neural networks such as multi layer feed forward, self-organizing and Hopfield

networks which can be used in the components of logistics systems that must be formulated and optimized as the mathematical models have been studied in details.

First, the multi layer feed forward neural networks that are widely used in many fields were investigated, frequently encountered problems and solving techniques were dwelled on, modifications to enhance the performance were considered and their integration with Genetic Algorithms were also evaluated.

Two basic approaches that are used to solve combinatorial optimization problems are the Hopfield neural network and self-organizing approaches. However, both approaches have some deficiencies and their success in solving different problem is dependent on the problem. Although, Kohonen maps can not be generalized in non-Euclidean problem, and Hopfield neural networks usually give infeasible solutions, and they necessitate tedious adjustments to get the optimal solution; the use of annealing, hybrid models, etc. provided satisfactory results especially in quadratic programming.

Classifying large amounts of data continuously and updating the classification are important for the performance of logistics systems. For this reason, self organizing maps that can extract the relations between data without a need for training data set are considered. Subsequently, this approach is applied to continuous covering location – allocation and vendor classification problems.

Considering the success of Hopfield neural networks in combinatorial optimization problems, especially the arrangements that improve the performance of network are analysed. For this reason, considering Gee and Smith's studies as a basis, the inner dynamics of networks are changed to improve the problem solving quality and used in solving the hub location problems. Using the test data, which are widely used in the literature, artificial neural network models are compared with another techniques by means of solution quality and consistency.

This thesis also investigates the use use of artificial neural networks to solve location-allocation problems, specifically continuous covering, p-median, and hub location models. The problem is solved by using self-organizing maps and Hopfield-Tank artificial neural networks which has a recurrent architecture. In addition, a Vibrational Genetic Algorithm is proposed to solve p-median and continuous covering location-allocation problems.

In order to test the effectiveness of the proposed methods, test data sets which are widely used in the literature are used. The results are compared with the results attained for artificial neural network-based methods, with the best results in the literature and with the results found by using heuristic approaches such as Simulated Annealing and Genetic Algorithms. Remarking results are attained for the location-allocation problems, and also proposed Vibrational Genetic Algorithm produced very good results. In addition, the proposed artificial neural network models were tested by using the logistics data of an international manufacturing company. In this way, the success of the system in real life systems is investigated.

Lastly, a mobile-agent based decision support system for integrated supply chain management systems were designed and implemented in Java environment. In this proposed framework, the structure of the system is sufficiently general to be easily applied to different logistics systems, and the artificial neural network models are used in the processes of demand forecasting and vendor classification.

1. GİRİŞ

Günümüz Lojistik Sistemlerinin müşteri odaklı olarak ve malzeme akışına göre şekillendirilmesi işlemi görünürde çok açık ve anlaşılır olmasına karşın; dinamik ve rekabetçi bir ortamda sanal şirket, elektronik ticaret gibi teknolojik gelişmelere ve yeni buluşlara hızla ayak uydurabilecek, farklı uygulama süreçlerinde ekonomi, biyoloji, sosyoloji ve psikoloji gibi alanlardan da teorik anlamda faydalanabilecek bütünleşik bir sistemi yönetmek oldukça karmaşıktır. Çeşitli avantajları nedeniyle parçalara ayırma, bütünleştirme ve geri besleme mekanizmaları yoluyla, sistem karmaşıklığı ile başa çıkmada *hiyerarşik karar verme* mekanizması kullanılmaktadır. Özellikle gerçek zamanlı, hata toleransını minimize eden, değişen pazar koşullarına hızlı uyum sağlayabilen yeni ve farklı yaklaşımlar üzerinde araştırmalar yapılmaktadır (Chandra ve diğ., 2001, Boyson ve diğ., 2003).

Önceleri tedarik zincirinde pazarlama, dağıtım, planlama, imalat ve satın alma işlemleri birbirinden bağımsız olarak gerçekleştirilmiştir. Bu bölümler, genellikle birbirleriyle çatışan bireysel amaçlara sahiptirler. Pazarlamanın yüksek müşteri düzeyi ve maksimum satış hedefi, imalat ve dağıtım bölümlerinin amaçları ile çatışır. Birçok imalat işlemi, düşük maliyetlerin envanter düzeyleri ve etkin dağıtıma olan etkisini çok az göz önünde bulundurarak, üretimi enbüyüklemeye yönelik olarak tasarlanır. Satın alma anlaşmaları, genellikle geçmiş satın alma örüntülerinin ötesinde çok az bilgi ile müzakere edilir. Bu yüzden, bu farklı işlevlerin bütünleştirilebileceği bir mekanizmaya gereksinim vardır (Kaihara, 2003).

Bütünleşik lojistik sistemlerinden beklenen özellikleri karşılayabilmek için klasik yaklaşımların yerine; işletmenin farklı işlevsel alanlarını, tedarik zincirinin koordinasyonu altında bütünleştirici bir yönetimle ele alan, alt bileşenlerin kendilerine özgü perspektiflere ve güdüleyicilere sahip olmalarından dolayı tek tip bir karar verme yapısına zorlanmadıkları bir *dağıtılmış karar verme mekanizması* kullanılabilir. Bu yaklaşım ile gevşek bağlı (*loosely-coupled*) bilgisayarların kullanımı sayesinde lojistik sisteminin bilgi yönetimi köklü olarak basitleştirilebilir. Paralel işleme ve adaptif öğrenme ile daha kararlı ve güçlü yapıların, daha hızlı olarak gerçekleştirilmesi yeteneği kazanılabilir.

Dağıtılmış problem çözme yaklaşımını temel alan bütünleşik tedarik zinciri sistemlerin güzel bir örneği İşbirlikçi Tedarik Zinciri (İTZ) yaklaşımıdır (Chandra ve diğ., 2001). İTZ en az bir grup ve birden fazla üyeden oluşur ve üyeleri arasındaki malzeme, süreç ve bilgi akışına göre şekillenir. Üyeleri arasındaki etkileşim bir tüketici ve tedarikçi ilişkisi biçiminde gerçekleşir. Sistemin bileşenleri otonom varlıklardır ve bir problemi çözme işi bu otonom varlıklar ile sistem arasında paylaştırılmıştır. Probleme ilişkin bilgiler ve çözümler üyeler arasında işbirlikçi olarak bölüşülmüş ve paylaşılmıştır. Grup varlığı, tedarik zincirinde koordinasyondan sorumludur. Üye varlığı ise uzman bilgisini ve teknolojiyi tedarik zincirine getirir. Grubun karar verme süreci (genel hedefler ve politikalar) merkezi olarak gerçekleşir. Ancak, üyelerin karar verme süreci dağıtılmıştır, her bir üye kendi amaçlarını, hedeflerini ve politikalarını bağımsız olarak takip eder. Sistemde kontrol, hedefler, politikalar ve amaçlar yoluyla hiyerarşik karar verme sürecinde eşzamanlı olarak sürdürülür. İşbirlikçi tedarik zincirinin eniyilenme prensipleri işletmenin işlemlerinde yöntemler, standartlar ve maliyetler arasındaki ilişkiler araştırılarak belirlenir. Bir sistemin bileşen varlıklarının davranışlarını göstermede dağıtılmış problem çözme yaklaşımının kullanılması, büyük ölçekli tedarik zinciri sistemlerinin modüler olarak modellenmesi fırsatını sunmaktadır.

Bütünleşik lojistik sistemlerinden beklenenlere alternatif bir çözüm sağlamak amacıyla, seçilen tez konusunun kapsamı içinde, karmaşık bir yapıya sahip, dağıtılmış karar verme ve paralel işlem yapma süreçlerini gerektiren, maliyetlerde ve/veya taleplerde belirsizliklerin söz konusu olduğu, karar mekanizmaların ve analizlerin bir veri tabanı yönetim sistemi yardımıyla gerçekleştirildiği bir lojistik sistemi için, dağıtılmış bir sistem modelleme yaklaşımının uygulandığı, öğrenme ve çıkarım mekanizmalarında bir yapay sinir ağı (YSA) altyapısının kullanıldığı sistem tasarlanacak ve gerçeklenecektir. Bu yeni altyapı sistemi ve onu oluşturup koordineli olarak eniyilenmeye çalışılacak modeller ile lojistik sisteminin operasyonel kararlarının kendiliğinden alınması, stratejik kararlarda üst düzey yöneticilere rapor verilmesi ve yol gösterilmesi, ayrıca performans analizleri yaparak kontrol edilmesi ve ileriye dönük projeksiyonların yapılması mümkün olacaktır.

1.1 Eniyileme Problemlerinin Çözümünde Geleneksel Yöntemler

Gerçek yaşamdaki durumlardan kaynaklanan bir çok problem Kombinatorik Eniyileme Problemi (KEP) olarak formüle edilebilir. Bu problemlerin en iyi çözümü, bütün kısıtları sağlayan ve verilen bir amaç fonksiyonunu eniyileyen çözümdür. Karar değişkenleri, örneğin gezgin satıcı probleminde olduğu gibi her bir şehre

gönderilecek ürün miktarı, gidilecek şehirlerin sırası gibi tamsayı değerler alan ayrık değişkenlerden oluşur. Kısıtlar tamsayı olma koşuluna ilave sınırlamalar getirir (talep edilen miktarın karşılanması gibi). Bu koşullara rağmen problemin uygun bir çözümünü veren bir çok kombinasyon olabilir. En iyi çözümün araştırılması amaç fonksiyonunu eniyileyen çözümün bulunması ile sonlandırılır. Eğer amaç fonksiyonu ve kısıtlar doğrusalsa genel optimum bulunabilir. Doğrusal olmayan bir amaç fonksiyonunda ise sıklıkla yerel olarak optimum olan noktaya ulaşılır.

Genel olarak kombinatorik eniyileme problemleri kesin veya sezgisel yaklaşımlarla çözülebilir. Kesin yaklaşımlar genel minimuma ulaşmayı garanti ederken, sezgisel yaklaşımlar genellikle en iyi olmayan ancak belirli ölçütlere (hesaplama süresi, çözüm kalitesi, vs.) göre iyice (suboptimal) çözümler verirler (Nemhauser ve Wolsey, 1999). Eğer amaç fonksiyonu ve kısıtlar doğrusal ve problem küçük boyutlu ise optimal çözüm kesin yöntemlerle bulunabilir. Oysa gerçek yaşamdaki problemler büyük ölçeklidir ve çoğunlukla kesin yöntemler çok fazla zamana ve bilgisayar hafızasına ihtiyaç duyacağından uygulanamaz.

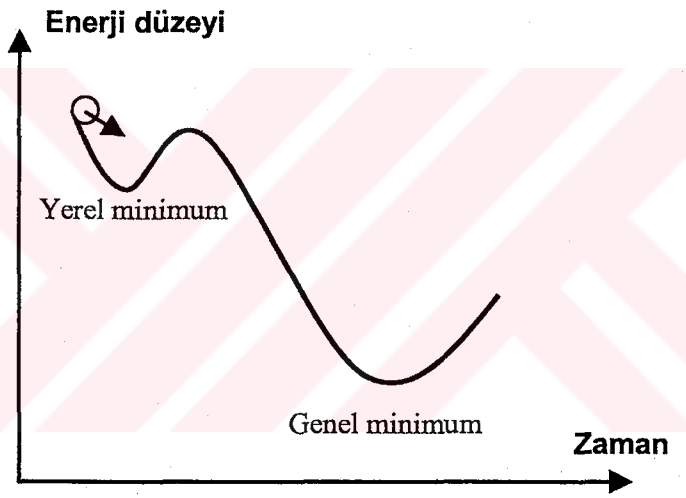
Özellikle lojistik sistemi yönetimi gibi sistemlerde çok hızlı bir şekilde bir çözüm elde ederek karar verici için bir alternatif oluşturmak oldukça önemlidir. Bu tip sistemlerde göreceli olarak kısa zamanda elde edilen iyi bir çözüm, çok uzun zamanda bulunabilen optimal çözümden daha faydalıdır. Bu yüzden daha hızlı ve optimale yakın çözüm veren sezgisel yaklaşımlar üzerinde çalışılmaktadır (Reeves, 1993).

Bu tip sezgisel yöntemlerden birisi Yerel Arama (YA) algoritmasıdır. YA belkide en eski eniyileme yöntemi olan deneme yanılma yaklaşımına dayanır. Aslında bir çok kombinatorik eniyileme probleminde yerel aramanın şaşırtıcı biçimde başarılı olduğu gösterilmiş olsa da bu yöntemde düşünce basit ve doğaldır. Bu teknikte olurlu bir başlangıç çözümü bulunduktan sonra, bazı kriterlere göre belirlenmiş komşular (örneğin, tamsayı programlamada tamsayı değere sahip komşular) daha iyi bir çözüm değeri elde etmek için araştırılır. Eğer daha iyi bir çözüm bulunursa yeni olurlu çözüm olarak bu çözüm alınır ve bu yeni noktanın bütün komşuları araştırılarak amaç işlevinde en fazla iyileşmeye neden olan çözüm bulunmaya çalışılır. Süreç araştırılan komşular içerisinde en iyi çözüm noktası (bir anlamda yerel minimum) bulununcaya kadar bu şekilde devam eder. Bu yaklaşım kullanılacağı zaman ilk olarak olurlu bir başlangıç çözümünün nasıl bulunacağına, "iyi" bir komşuluk tanımına ve bu komşuluğa göre uygulanacak arama yöntemine karar verilmesi gerekir. Bu algoritmanın en büyük dezavantajı, arama çizgisi belirli olduğundan, en sonunda ulaşılacak sonucun bütünüyle başlangıç çözümüne bağlı olmasıdır (Papadimitriou ve Steiglitz, 1998).

Bir diğer sezgisel yaklaşım ise Tavlama Benzetimi (TB) tekniğidir (Aarts ve Korst, 1989). Gerçek yaşamda metalürjide kullanılan tavlama tekniğinde; bir metal erime noktasının üzerinde yüksek bir enerji düzeyine çıkarılıp daha sonra yavaş yavaş soğutulur. Yüksek sıcaklıkta metal atomlarının enerji düzeyi de yüksektir. Sıcaklık azaldıkça atomların enerji seviyesi azalır ve metal minimum enerji konfigürasyonuna ulaşır. Eğer soğutma yeteri kadar yavaş yapılmışsa, sistemin enerji seviyesinin genel minimuma ulaştığı bir duruma erişilebilir. Verilen bir sıcaklıkta, sistem enerjisinin olasılık dağılımı Boltzmann olasılık faktörü ile hesaplanır:

$$P_T\{X = i\} = e^{-(E_i/kT)} \tag{1.1}$$

Yüksek sistem enerjisi olasılığı sıcaklık düştükçe hızlı bir şekilde azalır. Bu nedenle, düşük sıcaklıkta düşük enerji durumuna doğru güçlü bir eğilim vardır (Şekil 1.1).



Şekil 1.1 : Tavlama benzetimi

Tavlama yönteminin metalin soğutulup minimum enerjiye sahip bir kristal yapısına ulaşmasını sağlayan bu özelliği ile daha genel bir sistemde bir minimum noktanın araştırılması arasında analoji yapılarak rassal bir arama tekniği olan Tavlama Benzetimi (TB) geliştirilmiştir. TB bir eniyileme tekniği olarak kombinatorik eniyileme ve diğer eniyileme problemlerine başarıyla uygulanmaktadır.

Tavlama Benzetimi tekniği dört temel bileşene ihtiyaç duyar:

1. Olası çözümlerin bir gösterimi (solution representation)
2. Çözümlerde rasgele değişimleri yaratacak bir üreteç (a generator)
3. Çözüm kalitesini ölçmede kullanılacak amaç fonksiyonu

4. *Soğutma çizelgesi:* Bir başlangıç sıcaklığı ile arama süreci ilerledikçe bu sıcaklığın nasıl azaltılacağını belirleyen kurallar

TB'de bulunan yeni çözümleri kabul edip etmeme kararı genellikle Metropolis kriterine göre verilir. Bu kriter ile kabul olasılığı:

$$\mathrm{P}\{kabul\} = \begin{cases} 1 & eğer\ \Delta f < 0 \\ \exp(-\Delta f / T) & eğer\ \Delta f \geq 0 \end{cases} \tag{1.2}$$

şeklinde hesaplanır. Bu eşitlikte T kontrol parametresidir, Δf ise eniyileme problemi için yeni çözüm ile güncel çözüm arasındaki maliyet farkını gösterir.

TB algoritmasının ilk kontrol parametresi tavlama sıcaklığının (T_0) başlangıç değeridir ve aşağıdaki gibi hesaplanabilir (Aarts ve Korst, 1989):

$$\gamma = \frac{m_1 + m_2 \times e^{\frac{-\Delta f}{T_0}}}{m_1 + m_2} \tag{1.3}$$

Bu formülasyonda, γ kabul oranını, m_1 maliyette azalmaya yol açan hareketlerin sayısını, m_2 ise bir önceki adıma göre maliyette artışa neden olan hareketlerin sayısını ve Δf ise m_2 hareketleri için maliyetteki ortalama artışı gösterir. Başlangıç sıcaklığı Markov zinciri için geçici büyük bir sayıya eşitlenir. Bir tam Markov zinciri tamamlandıktan sonra, başlangıç sıcaklığı (T_0) Denklem 1.2 kullanılarak hesaplanır. Markov zincirinin uzunluğu (L_k) problemin büyüklüğüne bağlı olarak belirlenir.

TB, araştırmasına rasgele bir başlangıç noktası ile başlar ve bu noktadan kullanıcı tarafından önceden belli prensipler çerçevesinde belirlenmiş bir komşu noktaya ilerler. Yeni noktadaki amaç fonksiyon değeri başlangıçtaki amaç fonksiyonu değeri ile karşılaştırılır ve yeni değerin küçük olup olmadığı belirlenir. Enküçükleme durumunda, amaç fonksiyon değeri azalırsa otomatik olarak kabul edilir ve o nokta araştırmanın devam ettirileceği nokta olur. Ardından algoritma başka bir adım ile işleme devam eder. Amaç fonksiyonunun yüksek çıkan değerleri de Metropolis kriteri ile belirlenmiş bir olasılık dahilinde kabul edilebilir. Nadiren kabul edilen amaç fonksiyonunun yüksek değerleri sayesinde, TB algoritması yerel minimumlardan kaçabilir. Algoritma ilerleyip son çözüme doğru yaklaşıldıkça, araştırma yapılacak komşu noktaları belirlemede kullanılan komşuluk yarıçapı azalır. Metropolis kriteri kullanıcı tanımlı başlangıç parametrelerinde (sıcaklık ve sıcaklık azaltma faktörü), amaç fonksiyonunun yüksek bir değerini kabul etme olasılığını belirlemede kullanılır.

Gerçek tavalama sürecine paralel olarak sıcaklık azaldıkça yüksek değerleri kabul etme olasılığı da azalır. Sıcaklık, belirli sayıdaki yinelemeden sonra Denklem 1.3 kullanılarak azaltılır.

$$T_{n+1} = f(T_n), \quad \forall n = 0,1,...,n-1 \tag{1.4}$$

TB kullanılarak bulunan çözümler başlangıç çözümüne bağımlı değillerdir ve optimal çözüme yakın çözüm verirler. Bu yüzden YA tekniğinin dezavantajlarını içermezler. Fakat, TB yöntemi YA tekniğine göre daha yavaş çalışır ve parametrelerinin seçimi genellikle çözülmeye çalışılan probleme göre belirlenmek zorundadır. Yine de TB yaklaşımı çoğu eniyileme problemini çözmek için yaygın olarak kullanılmaktadır ve bu yöntemle elde edilen değerler genellikle maliyetler için üst sınırı oluşturur.

Eniyileme problemlerinin çözümünde kullanılan bir diğer sezgisel yöntem ise Genetik Algoritmalar'dır (GA). GA, Darwin'in biyolojideki bir popülasyonun evrilmesi ile ilgili olarak öne sürdüğü evrim teorisi fikrini taklit eden, bir noktalar kümesinden diğerine araştırma yapan genel arama prosedürüdür (Goldberg, 1989; Mitchell, 1998). GA sürekli olarak parametre uzayında örnekleme yaptığından, arama çok uzaktaki en iyi çözüm alanına doğru yönlendirilir. Bu algoritma, doğrusal olmayan zor fonksiyonlar için genel çözümlerin bulunmasında oldukça iyi performans göstermektedir.

GA bir eniyileme problemine uygulandığında, probleme önerilen değişik çözümler (popülasyondaki bireyler), amaç fonksiyonuna göre birer uygunluk değerleri alacaklardır. Amaç fonksiyonunun sürekli veya türevi alınabilir olmasına gerek yoktur. Daha sonra bu değerler popülasyondaki her bir noktaya olasılık atamada kullanılır. Uygunluk değeri yüksek olan bireylerin çevreye daha iyi uyum sağladığı kabul edilerek seçilirken, uygunluk değeri küçük olan bireyler elimine edilir ve iyi bireylerden çaprazlama ve mutasyon yoluyla yeni bir nesil (popülasyon) oluşturulur. Bu işlemler hedeflenen uygunluk değerlerine ulaşılıncaya kadar devam eder.

Bu yöntemde aynı anda pek çok yönde araştırma yapıldığından, genel optimumu bulma olasılığı oldukça fazladır. GA nesiller boyunca gelişeceğinden, amaç fonksiyonunu eniyilemek için en geçerli parametreler üretilecektir ve gelecek nesillerde zayıf performans gösteren parametrelerin yok olmasıyla iyileşecektir. Oysa gradyan esaslı yöntemler sadece bir noktada çözümü arar. GA'lar gradyan esaslı yöntemlerde olduğu gibi amaç fonksiyonunun türevinin hesaplanmasına da gereksinim duymaz. GA'lar popülasyona bağlı olarak işlem yaptığı için paralel hesaplamaya da uygundur. Basit bir düzenlemeyle popülasyondaki bireylerle ilgili

hesaplamalar paralel olarak gerçekleştirilebilir. En büyük dezavantajlarından birisi ise problemin tipine bağlı olarak hesaplama süresinin oldukça uzun olabilmesidir.

İlerleyen bölümlerde, yapay sinir ağlarının performansı ile tavlama benzetimi ve genetik algoritmalarınki karşılaştırılacaktır. Kombinatorik eniyileme problemleri için son yıllarda yasaklı arama (Sue, 1998), melez yaklaşımlar (Berger ve Barkaoui, 2004; Wang ve Wu, 2004), karınca kolonileri optimizasyonu (Solimanpur ve diğ., 2004; Rajendran ve Ziegler, 2004) gibi diğer yöntemler de kullanılmaya başlanmıştır. Ancak bu sezgisel yöntemler bu çalışmada referans olarak kullanılmayacaktır.

1.2 Lojistik Sistemlerin Eniyilenmesinde Sinir Ağı Yöntemleri

Doksanlı yıllardan itibaren sinir ağları gibi yeni teknolojilerin yöneylem araştırmasının inceleme sahalarına giren öngörüleme, modelleme, araç rotalama, gezgin satıcı problemi, kümeleme, sınıflandırma gibi bir çok alandaki problemleri çözmede kullanılmaya başlandığı gözlemlenmektedir. Sinir ağları yapay zeka ve beynin modellenmesi alanlarından geliştirilse de, aslında regresyon ve benzer yöntemlerde olduğu gibi bağımlı ve bağımsız değişkenler arasındaki ilişkiyi öğrenen bir fonksiyon tahmin aracından başka bir şey değildir. Sinir ağları ile istatistiksel yöntemler arasındaki temel farklılık, sinir ağlarının istatistiksel dağılımla veya verilerin özellikleri ile ilgili herhangi bir varsayım yapmamasıdır (parametrik değil). Bir diğer önemli özelliği de doğrusal olmayan bir tahmin yöntemi olduğundan karmaşık verilerin modellenmesinde daha kesin olmasıdır.

Bölüm 2, 3 ve 4'de detaylı olarak açıklanacağı gibi, lojistik sistemlerinin matematiksel modelleri kurularak eniyilenmesinin gerektiği alt bileşenlerinde (öngörüleme, tedarikçi seçimi, depolama, fiziksel dağıtım, vs.) kullanılabilecek YSA modelleri; çok katmanlı yapay sinir ağları, kendi kendini düzenleyen ağlar ve Hopfield ağları olmak üzere üç ana kategoride incelenebilir.

İstatistikçiler uzun yıllar verilerin örüntülerini modellemede eğer eğitici küme olarak adlandırılan veriler (girdiler ve bilinen çıktı değerleri) mevcutsa diskriminant analizi ve regresyonu, bu tip veriler yoksa kümeleme tekniklerini kullanmışlardır. Bu teknikler sinir ağlarıyla paralellik gösterir: eğitici veriler mevcutsa geriye yayılımı kullanan çok katmanlı ileri beslemeli YSA ve eğitici küme yoksa kümeleme tekniği olarak kendi kendini düzenleyen ağlar. Kümeleme her zaman verilerin doğal yapılarına bağlı olarak gruplandırmıştır. Uygun bir kümeleme algoritmasının amacı, farklı kümelere ait örüntülere ait benzerlikleri enküçüklerken, bir kümedeki örüntüler arasındaki benzerlik derecesini enbüyüklemektir. Çok boyutlu bir girdi uzayındaki

örüntüler çok karmaşık bir yapıya sahip olabileceklerinden, bu yapıyı daha saydam ve basit yapmak için bir, iki veya üç boyutlu özellik uzayında kümelendirilirler. Kohonen büyük veri kümelerindeki güçlü ilişkileri otomatik olarak bulmayı sağlayacak bir yol olarak kendi kendini düzenleyen özellik haritalarını geliştirmiştir (Kohonen, 1984, 1985, 1995). Kohonen ağları çok boyutlu girdi uzayını az boyutlu özellik uzayına dönüştürerek kümelerin görünür olmasını sağlar.

Kohonen ağları çok katmanlı ağlarla karşılaştırıldığında; bu ağlar da geriye yayılımda olduğu gibi öğrenmeyi gösteren ağırlıkların adaptasyonunu içerir, fakat istenilen çıktı değerleri mevcut olmadığından öğrenme gözetimsizdir. Bir diğer farklılık da ağın mimarisi ve öğrenme sürecinde nöronların konumlarının rolüdür. Kendi kendini düzenleyen ağlar ağırlıklı olarak kümeleme ve öz nitelik çıkarımında kullanılmıştır. Ayrıca, eniyileme problemlerini çözmede Hopfield ağlarının yerine kullanma konusunda da bir çok araştırma yapılmıştır. Gezgin satıcı problemi dışında araç sıralama, konum-tahsis ve dağıtım kanallarının atanması gibi problemlerde de başarıyla uygulanmıştır (Smith ve diğ., 1996a)

Yapay sinir ağlarının eniyileme problemlerini (özellikle NP-sıkı problemleri) çözmede kullanılması fikri 80'li yıllara kadar dayanmaktadır. İlk olarak Hopfield ve Tank Gezgin Satıcı Probleminin Hopfield YSA kullanılarak nasıl çözülebileceğini 1985 yılında göstermişlerdir (Hopfield ve Tank, 1985). Ancak birçok parametre ve terim içeren enerji fonksiyonunun enküçüklenmesi çoğunlukla olursuz çözüm üretmiştir (Wilson ve Pawley, 1988). Olurlu çözüm üretebilmek için araştırmacılar enerji fonksiyonunu değiştirmeye veya bir çok parametreyi optimal olarak ayarlamaya çalışmışlardır. Hopfield ağı yakınsarken son çözümünü olurlu kısıt düzlemine hapseden bir yöntem bulunmuştur (Gee, 1993). Hopfield ağının çözüm kalitesini artırmak için bir çok yöntem denenmiştir. Bu yöntemler kabaca deterministik ve olasılığa dayananlar olarak ikiye ayrılabilir. Olasılığa dayalı yöntemler tavlama benzetiminin yerel minimumdan kurtulma prensipini Hopfield ağına uyguladığından, muhtemelen çözüm kalitesini artırmada daha başarılıdırlar. Hopfield ağı, doğrusal ve doğrusal olmayan programlamanın yanı sıra çizgeleri parçalara ayırma, sırt çantası ve kısıt doyumu problemlerinde de uygulanmıştır (Peterson and Anderson, 1989; Van den Bout and Miller III, 1990).

Pratikteki bir çok zor problemde sezgisel yaklaşımlar hızlı çözüm elde etme ihtiyacından kullanılmaktadır. Genel optimum çözüme ulaşmak, optimale yakın bir çözümü hızlı bir şekilde elde etmek kadar zorunlu değildir. Sinir ağlarının ana avantajlarından birisi hızlı hesaplama gücü olduğundan, özellikle endüstriyel uygulamalarda bu özellik çok daha değer kazanmaktadır.

1.3 Tezin Kapsamı ve Katkıları

Bu tez çalışmasında, lojistik sistemlerinin modellenmesinde ve eniyilenmesinde yapay sinir ağlarının kullanım potansiyelini değerlendirmek, gerçek zamanlı bir sistemde kullanılabilecek gezgin etmen temelli ve sinir ağını kullanan bir karar destek sistemini tasarlayıp gerçeklemek hedeflenmiştir.

İlk olarak çok geniş bir alanda yaygın olarak kullanılan çok katmanlı sinir ağları incelenmiş, kullanımı sırasında sıklıkla karşılaşılan problemler ve çözüm yöntemleri üzerinde durulmuş, performans iyileştirici değişiklikler ele alınmış ve genetik algoritmalar ile melez olarak kullanımı da değerlendirilmiştir.

Özellikle çok büyük miktardaki bilgileri sürekli olarak sınıflandırabilmek, sınıflandırmayı güncelleyebilmek lojistik sistemlerin performansı açısından önemli olduğundan, eğitme kümesine gereksinim duymadan veriler arasındaki ilişkileri ortaya çıkaran kendi kendini düzenleyen ağlar ele alınmış ve tedarikçilerin sınıflandırılması ile sürekli örten Konum-Tahsis Problemlerine (KTP) uygulanmıştır.

Hopfield sinir ağlarının kombinatorik eniyileme problemlerindeki başarısı dikkate alınarak, özellikle ağın performansını iyileştirici düzenlemeler üzerinde durulmuştur. Bu amaçla Gee ve Smith'in çalışmaları temel alınarak çözüm kalitesini artırmaya dönük ağın içsel dinamikleri değiştirilmiş ve merkez üsler problemini çözmede kullanılmıştır (Gee, 1991; Smith ve diğ., 1996). Sinir ağı modelleri literatürde yaygın olarak kullanılan gerçek test verileri kullanılarak diğer tekniklerle çözüm kalitesi ve tutarlılığı açısından karşılaştırılmıştır.

Son olarak, bütünleşik tedarik zinciri yönetimi sistemleri için gezgin etmen mimarisini temel alan, ana modül ile alt bileşenlerinin koordineli olarak hareket etmesi nedeniyle operasyonel kararların, bilgi sistemi tarafından otomatik olarak alındığı, sistemin çatısının değişik lojistik sistemi tiplerine kolaylıkla genişletilebilecek yeterlikte genel olduğu, talep tahmini ile tedarikçi seçimi süreçlerinde YSA modellerini kullanan bir karar destek sistemi Java platformunda tasarlanmış ve gerçeklenmiştir.

Özet olarak, tezin temel katkıları:

- Çok katmanlı ileri beslemeli yapay sinir ağlarının talep tahmininde ve sınıflandırma problemlerinde kullanım sınırlarının ve özelliklerinin ortaya konması
- Tedarikçilerin sınıflandırılması ve sürekli örten uzay konum-tahsis problemlerinin çözümünde Kohonen ağlarının kullanılması

- Sürekli örten uzay konum-tahsis problemleri için yeni bir çözüm yöntemi olarak Titreşimli Genetik Algoritmanın önerilmesi
- Hopfield sinir ağının tutarlı ve yakınsayan çözümler üretmesini sağlayacak enerji fonksiyonu değişikliklerinin incelenmesi
- Hopfield ağının merkez üslerin konumlandırılması gibi zor eniyileme problemlerinde yerel minimumdan kaçınarak en iyi çözüm elde edilmesinin sağlanması
- Sinir ağları tekniklerinin çok sayıda test verileri ile analiz edilerek sonuçların diğer tekniklerle karşılaştırılması
- Yapay sinir ağları ile desteklenmiş etmen temelli ve platformdan bağımsız bir lojistik karar destek sisteminin tasarlanarak gerçeklenmesi
- Bu etmen temelli sistemin adaptif davranma ile yeni durumlara uyum sağlayabilme ve dağıtılmış, paralel bilgi işleme ve karar vermeyi desteklemesi
- Önerilen YSA modellerinin üretim yapan uluslararası bir firmanın lojistik verileri kullanılarak test edilmesi

1.4 Tez Planı

Tezin ikinci bölümünde, lojistik sistemlerinin yönetimindeki yeni eğilimler ele alınmış ve yapay sinir ağları gibi akıllı teknolojilerin bu sistemlerin hangi alanlarında kullanılabileceği, neden yapay sinir ağları ile bütünleşik bir tedarik zinciri yönetim sistemine gereksinim duyulduğu irdelenmiştir.

Üçüncü bölümde, doğrusal olmayan sistemler için evrensel bir kestireç olarak yaygın olarak kullanılan çok katmanlı ileri beslemeli yapay sinir ağları ele alınmış, mimarisi, öğrenme algoritmaları, kısıtlamaları ve genetik algoritmalarla beraber kullanımı incelenmiştir.

Dördüncü bölümde, rekabet düşüncesini açıkça kullanan ve özellikle fazla ve karmaşık verilerin analizinde veya özellik çıkarımında kullanılan Kohonen ağları (kendi kendini düzenleyen ağlar) detaylı olarak ele alınmıştır.

Beşinci bölümde, kombinatorik eniyileme problemlerini çözmede ve özellikle *ikinci dereceden* programlamada kullanılan Hopfield-Tank sinir ağları açıklanmıştır.

Altıncı bölümde, talep tahmininde kullanılmak üzere çok katmanlı YSA modeli tasarlanmış ve gerçeklenmiştir. Farklı öğrenme algoritmaları ve farklı ağ parametreleri için testler yapılarak en uygun ağ mimarisi belirlenmeye çalışılmıştır.

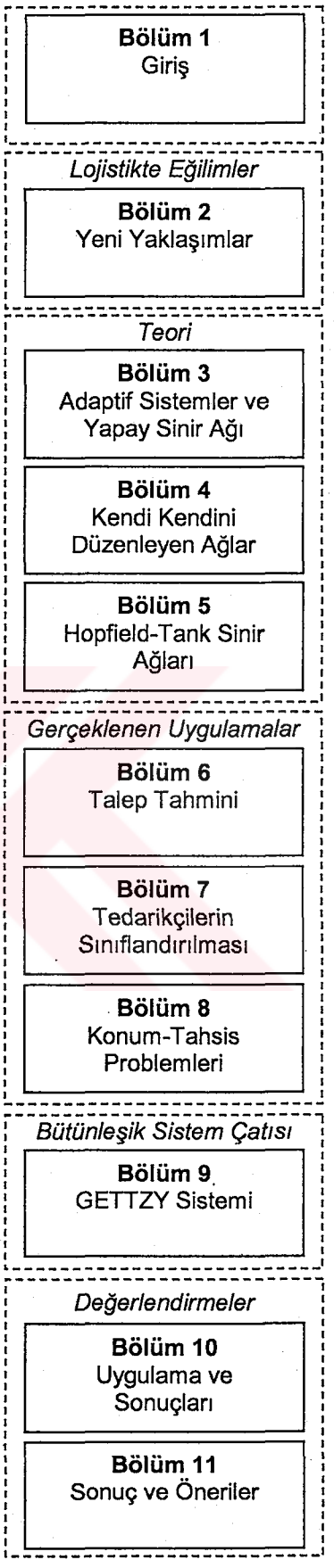
Yedinci bölümde, tedarikçilerin sınıflandırılmasında kullanılmak üzere Kohonen ağı mimarisine sahip bir YSA tasarlanıp gerçeklenerek test verileri ile analiz edilmiştir.

Sekizinci bölümde, KTP'leri, özel olarak sürekli örten, p-medyan ve merkez üslerin konumlandırılması problemlerini çözmede yapay sinir ağlarının nasıl kullanılabileceği irdelenmiş, kendi kendini düzenleyen ağlar (Kohonen) ile yinelemeli mimariye sahip bir YSA olan Hopfield - Tank YSA kullanılarak KTP çözülmüştür. Ayrıca, p-medyan ve sürekli örten KTP problemlerini çözmek için yeni bir yaklaşım olarak *Titreşimli Genetik Algoritma* (TGA) yaklaşımı önerilmiştir.

Dokuzuncu bölümde, bütünleşik tedarik zinciri yönetimi sistemleri için gezgin etmen mimarisini temel alan, talep tahmini ile tedarikçi seçimi süreçleri YSA modelleri kullanılarak Java platformunda gerçeklenen karar destek sistemi (GETTZY) açıklanmıştır.

Onuncu bölümde, önerilen GETTZY sistemi üzerinde bir firmadan temin edilen veriler için gerçekleştirilen testler ve bu testlerden elde edilen sonuçlar verilmiştir.

Onbirinci ve son bölüm olan Sonuç ve Öneriler bölümünde, tasarlanan ve geliştirilen yapay sinir ağları modelleri ile gezgin etmen temelli tedarik zinciri yönetimi sisteminin sağladığı katkılar anlatılmış ve öneri olarak, gelecekte sistemin daha da geliştirilmesine yönelik yapılabilecek çalışmaların neler olabileceği üzerinde durulmuştur.



Şekil 1.1 : Tezin bölümleri

2. LOJİSTİK SİSTEMLERİNDE YENİ YAKLAŞIMLAR

Yaklaşık 10-15 yıldır imalat sanayi, rekabetçiliğini artırmaya yönelik sürekli bir baskıya maruz kalmıştır ve genel anlamda kabul edilen değişimlerden birisi lojistik yoluyla gerçekleştirilenlerdir. Literatürde ve pratikte, imalatın mükemmelliğine etkin lojistiğin nasıl katkıda bulunabileceğine dönük pek çok çalışma yer almaktadır. Ancak herhangi bir özel durumda lojistik yaklaşımının ne olacağı sorusunu soran lojistik yöneticileri oldukça karışık, belirsiz cevaplar ve genel düşüncelerle karşılaşmışlardır. Faaliyetlerin malzeme akışına ve müşterilere göre yeniden yapılandırılması prensibi çok açık ve anlaşılır olmasına rağmen, yeni dinamik imalat ortamında lojistiği yönetme işi oldukça karmaşıklaşmıştır. Bu gerçeği destekleyen lojistik ortamındaki değişimlerden bazıları:

- Pek çok imalat şirketinde rekabetçi konum, şu alanlardaki baskıdan sorumludur: *fiyat*, *kalite*, *zaman* ve *hizmet*. Lojistik, rekabete dayanan avantajların kazanılması ve sürdürülmesindeki anahtarlardan birisidir.

- Teknolojik gelişmelerin ve buluşların hızı, imalatçı firmayı yönetmede yeni fırsatlar ve değişen durumlar yaratır. Teknoloji, rekabetçilik için bir anahtardır ve sonuçta, yeni teknolojinin etkin yayılımı lojistik yönetimi işinin önemli bir parçasıdır.

- Endüstriyel yapı, bir sanal yatırımcı gibi karakterize edilen küresel üretime doğru değişmektedir. Sanal şirkette tedarik zinciri altyapısı ortaklığa dayanır ve lojistik ile malzeme akışı bütünleştirici faktörlerdir.

- Araştırmalar, sadece teknolojik süreçlerde değil; yönetim, ekonomi, sosyoloji ve psikoloji gibi alanlarda da imalata dönük yeni teorik bakışlar üretmiştir.

- Lojistikteki zaman boyutu kısalmıştır; malzeme akışının dinamikleri, belirsizlikler ve sabit değişim, yeni ve daha kararlı yapıların yaratılması yeteneğine ihtiyaç duymaktadır.

2.1 Bütünleşik Lojistik ve Dağıtılmış Karar Verme

Lojistiğin kapsamı genişlemektedir; bütün yaşam çevriminde malzeme akışının, tedarikçiden alınıp, imalat sürecinden geçip son müşteriye ulaşan, geri dönüşümü veya elden çıkarılması da göz önüne alınan bir unsur olarak kabul edildiğinde, pek çok organizasyon ve sistemle entegrasyonuna ihtiyaç duyulur.

İçsel ve dışsal müşteri ve tedarikçilerle koordinasyonu araştıran bir imalat unsuru, farklı operasyonel kararlar, yerel kısıtlar, çatışan amaçlar ve yanlış güdülenmiş istekler gibi zorluklarla çabucak yüz yüze gelir. Eğer bir çeşit merkezi koordinasyon oluşturulursa, önemli miktardaki zaman ve kaynak öncelikle bu farklılıkların giderilmesine sarf edilebilir. Bununla birlikte, koordinasyonun düzeyi ve kapsamı arttıkça merkezi koordinasyon nosyonu, sistem karmaşıklığının sınır değerine ulaştığı noktada işlemez hale gelir ve yerel otonomiye sahip merkezi olmayan koordinasyon yapısı kaçınılmaz olur. Parçalara ayırma, bütünleştirme ve geri besleme mekanizmaları yoluyla sistem karmaşıklığı ile başa çıkmada hiyerarşik karar verme kullanılmaktadır. Yeni ve farklı bir yaklaşım ise, **dağıtılmış karar verme** ve **pazar dengesi** fikridir. Üretimdeki pek çok karar unsurunun, kendilerine has perspektiflere ve ekonomik güdüleyicilere sahip olduklarına inanılmaktadır. Bütün karar unsurlarını tek tip bir karar yapısına zorlamak yerine, kendi alanları ile ilgilenen etmenler olarak görmek yardımcı olabilir. Bu karar etmenleri, *rekabetçi pazar mekanizmaları* kullanımı yoluyla, daha basit davranış biçimleri kümesine bağlı olarak koordine edilebilir, uzun-dönem davranışları tahmin edilebilir ve farklı denge şartlarına göre modellenebilir (Wu ve Harker, 1999).

Yeni yaklaşımın en önemli avantajı, bilgi yönetimindeki köklü basitleşmedir. Geleneksel İşletme Kaynakları Planlaması (İKP) sistemindeki temel paradigma, yukardan aşağıya, hiyerarşik olarak sistem detaylarının *"toplam görünürlüğünün"* araştırılmasıdır. Bu ise, sofistike bilgiler ve veri tabanı yönetim sistemleri kullanılarak, organizasyonun bütün algılanabilir yönlerine ait korkunç detaylı bilgilerin devamlılığının sağlanması yoluyla başarılabilir. Bu bilgiler organizasyonda karar vermeye temel teşkil ettiğinden, güncel olarak tutulması zorunludur. Dağıtılmış bir sistemde, etmenler sahip oldukları kendilerine ait bilgilere ve yerel tercihlere/kısıtlara dayalı olarak yerel özerk kararlar verdiklerinden, merkezileştirilmiş bilgi yönetimi ayrıştırılabilir, izleme ve bilgilerin devamlılığı bölümlendirilerek çok daha etkin bir şekilde yürütülebilir (Walker, 1997).

2.2 Yeni Tedarikçi Perspektifleri

Yeni tedarikçi yaklaşımları belki de son yıllarda en önemli yeni lojistik kavramıdır. Tedarikçi yönetiminin kontrol edilebilirliğini iyileştirme prensipleri evrensel olmakla birlikte hedeflerin önceliği ve arzı kontrol etme niyetleri durumsal faktörlere bağlıdır.

2.2.1 Tedarik Zinciri Yönetimi

Tedarik Zinciri Yönetimi (TZY), tedarikçi yönetimi ile çok yakından ilgisi olan bir kavramdır ve şebekelerdeki tedarik zincirinin geliştirilmesi üzerine odaklanır. Bu genel düşüncede, malzeme akışı yönetimine bütünsel bir yaklaşım önerilir ve ortaklık ve güven, tedarikçi ilişkilerinde ve malzeme akışının yönetiminde birer unsur olarak ortaya çıkar. TZY'ye dair araştırmalar stratejik, organizasyonel ve lojistik konularının her türünde yapılmaktadır fakat çoğunlukla tedarikçiler üzerine odaklanılmıştır (Mertz, 1994).

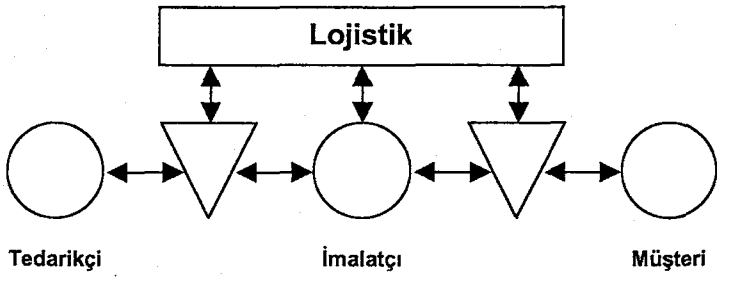
ABD ve Avrupa'da TZY'nin farklı anlamlar taşıdığına ilişkin bir eğilim vardır. ABD'de müşteri ilişkilerine, servise ve toplam zincir maliyetine odaklanılmıştır. Avrupa'da ise odak ortaklıktır. Her iki durumda da zincirin şeffaflığı, ilişkilerin güvenle oluşturulması ve işbirliği, engeller aynıdır. TZY, gelecekte ve şu an üzerinde çalışılan etmenler (zaman ve maliyete odaklanma, organizasyonlar arası odaklanma, motivasyon, bilgi sistemleri, vs.) için temel bir vizyondur. Dört temel yaklaşım belirlenmiştir: *tepeden-aşağıya*, *tabandan-yukarıya*, *ürün odaklı* ve *müşteri odaklı*. Karşıt tepkiler ise; *TZY modelleme*, *TZY performans ölçütleri*, *kantitatif analizler*, *örgütsel engeller*, *kaynak çatışmaları*, *eğitim ve bilgi teknolojisinin etkin kullanımı* içerisinde yer almaktadır.

2.3 Bütünleşik Lojistikte Yeni Çalışmalar

Yeni üretim ortamında lojistiğin rolü hatırı sayılır ölçüde artmaktadır: Rekabet, küreselleşme, yeni teknoloji vs. gibi en önemli değişim gösteren faktörler, lojistiğe üretimde yeni ve önemli bir stratejik rol (yani, lojistik stratejisinin ikinci boyutu) vererek en üst ilgiye sahip olmuşlardır. Bu yüzden lojistik araştırmalarının odak noktası, reaktif bir maliyet azaltma aracı olmaktan, şirket stratejisinde proaktif bir bileşen olmaya doğru kaymıştır.

Lojistik yönetimi; müşteri hizmeti, zaman ve kalite için yarışmada rekabetçi bir strateji gibiydi. Bu nedenle lojistiğin odağı, içsel üretkenlik ve işlemlerde etkinliğe doğru genişlemiştir. Aynı zamanda, lojistik yönetimi çok daha karmaşık bir iş halini

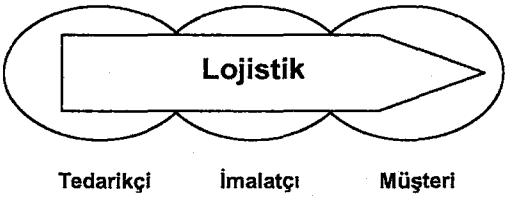
almıştır. Bu ise, rekabetçi ortam ve sonuçta yüksek performans gereksinimi ve belirsizlik nedeniyledir. Belirsizlik ve sürekli değişim lojistiğin karmaşıklığını artırmaktadır. Sürekli değişim, lojistik operasyonlarını mevcut durumla eşleştirmek için optimize etmeden daha fazla, lojistik yapısının tasarımına ilginin artmasına yol açmıştır. Yeni durum Şekil 2.1'deki gibi gösterilebilir.



Şekil 2.1 : Eski lojistik çatısı

Yeni lojistik çatısında odaklanılan nokta, her bir bileşenin performansından ziyade toplam performanstır ve toplam malzeme akışı tek bir unsurmuş gibi idare edilir. Entegrasyon, yeni çatıda önemli bir kavramdır ve malzeme akışı pek çok boyutun bütünleştiricisi olarak görülür.

Yeni lojistik çatısında örgütsel düzeylerin ve envanterlerin, eski çatıda malzeme akışının koordinasyonunda kullanılan elemanlardaki gevşeklik gibi, çoklu koordinasyonu için zaman yoktur (Christopher, 1992). Onun yerine, malzeme akışını kontrol etme, tedarik zinciri yapısı ile bütünleştirilmiştir (Şekil 2.2).



Şekil 2.2 : Bütünleşik lojistik çatısı

Yeni lojistik çatısında süreçlerin, bölümlerin, işlevlerin, organizasyonların, disiplinlerin, sistemlerin vs. entegrasyonu gereklidir ve bu yüzden yönetsel ilişkiler ve ortaklıklar önemlidir.

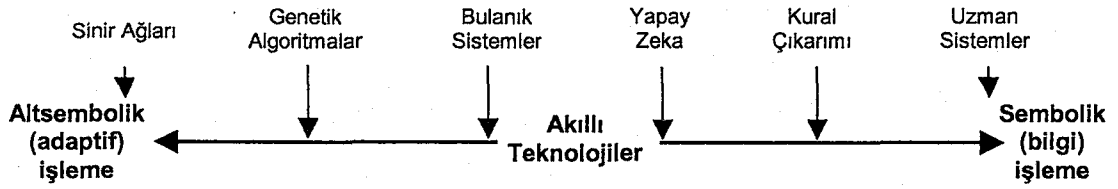
Yeni lojistik çatısı tek bir firma, bölümler veya işlevlerin kendi başlarına bağımsız üniteler gibi artık düşünülmediği, fakat bir bütünün parçaları gibi görüldüğü ve malzeme akışının, sanal şirketi veya yatırımları bütünleştiren paylaşılmış nesneler

olduğu yeni bir işyeri algılama şeklidir. Lojistik sadece malzemelerin hareketi ve depolanması değildir. Lojistik, malzeme akışı ile direkt veya endirekt ilgili bütün faaliyetlerin ve sistemlerin bütünleştiricisidir. Örneğin; kalite sistemleri, tasarım sistemleri, malzeme yönetim sistemleri, üretim planlama ve kontrol sistemleri, dağıtım sistemleri, vs. Faaliyetler ve sistemler, bilgi ve kontrol prensipleri gibi daha soyut formları da içermelidir.

Lojistik stratejileri, genel rekabetçiliği ve pazar isteklerine hızlı cevap verebilmeyi destekleyecek biçime dönüşmüştür. Bu nedenle, lojistik sistemlerinin zaman maliyeti ve kalite odaklı olmaya devam ederek, daha fazla zamana yönelimli olması kaçınılmazdır. Geleneksel eniyileme teknikleri müşteri isteklerinin ve zamanın ön plana çıktığı bu tip problemlere gürbüz çözümlerin türetilmesinde oldukça zayıf kalır. Örneğin, küçük, beklenmeyen hata veya arızalar eniyilenmiş bir planın hiçbir işe yaramamasına yol açabilir. Bunun yanında, yeni en iyi veya iyice çözümü bulmak için gereken süre çok uzun olabilir ve eniyileme süreci ile elde edilecek kazanımları geride bırakabilir. Özellikle rotalama, konum-tahsis gibi NP-zor yapıdaki problemlerde, kaliteli çözümler elde etmek için gereken süre, sistemin karmaşıklığı ve gerçekliği ile doğru orantılı olarak üssel bir biçimde artar. Gerçek zamanda gerçekleşen olaylar, işlemsel çevrenin değişkenliği ve belirsizliğinden dolayı ciddi etkilere yol açmaksızın, mümkün olan en üst düzeyde etkin bir akışı sürdürmek için gerçek zamanlı çözümlere gereksinim duyar. Akıllı teknolojiler bu noktada belirsizlikleri taklit etmede çok başarılıdırlar ve arzu edilen durumda herhangi bir değişiklik tespit edildiğinde gürbüz, zamanında çözüm üretirler. Ayrıca, sistemin bütününün çok fazla etkilenmesini önleyerek verimliliği devam ettirirler.

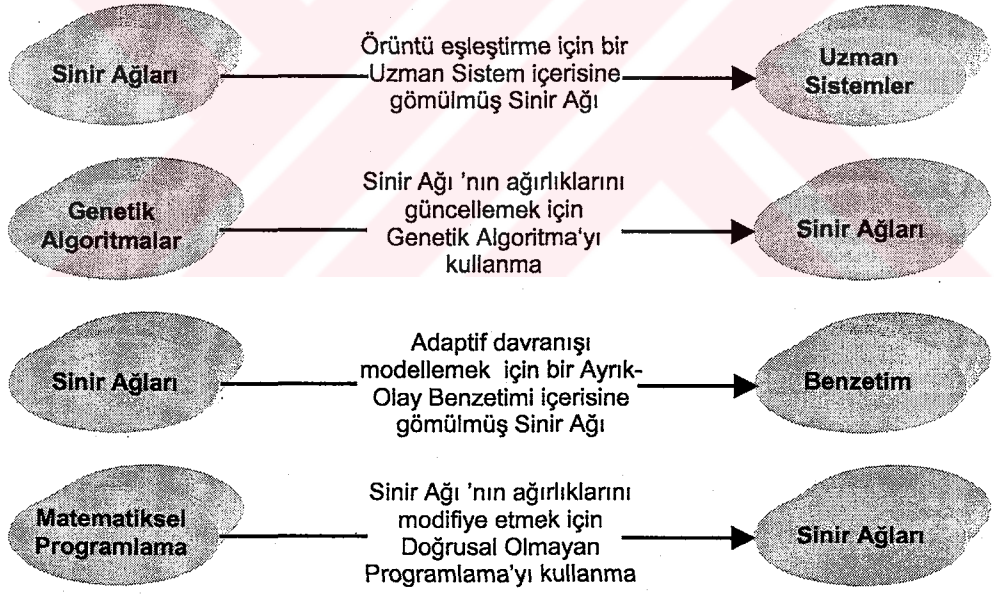
2.3.1 Akıllı Teknolojiler

Eniyileme problemlerinin çözümünde de kullanılan akıllı teknolojiler; yapay zekâ, uzman sistemler, sinir ağları, bulanık sistemler, kural çıkarımı ve genetik algoritmaların değişik tekniklerini içerir. Bu teknolojiler Şekil 2.3'de görüldüğü gibi, bir uçta sinir ağları gibi altsimgesel işleme yapısına sahip, diğer uçta ise uzman sistemler gibi simgesel işleme özelliği taşıyan teknolojiler arasında yer alırlar. Daha yeni akıllı teknolojiler ise *nanoteknoloji* adı altında gündemdedirler. Melez akıllı sistemler, en az bir tanesi akıllı teknoloji olan birkaç teknolojinin bütünleştirilmesi ile oluşturulmuş problem-çözüm sistemleridir. Eğer bir kaç sinir ağı, gevşek olarak bağlı ve hiyerarşik bir yapıda veya özel bir problemi çözmek için bir "ağların ağı" şeklinde biçimlendirilmişse, bu sistem genellikle melez sinir ağı veya melez akıllı sistem olarak adlandırılır.



Şekil 2.3 : Akıllı teknolojiler (Dağlı,1994)

Melez akıllı sistemlere örnek olarak, örüntü eşleştirme işlevini gerçekleştirmek için sinir ağını kullanan bir uzman sistem verilebilir (Livneh, 1999). Bu örnekte, her iki teknoloji de (*uzman sistem* ve *sinir ağı*) akıllı teknolojilerdir. Ayrıca geleneksel problem-çözüm tekniği ile bütünleştirilmiş akıllı teknolojiyi kullanan bir sistem de oluşturulabilir (Örneğin, sinir ağları ve ayrık-olay benzetimi). Bu tip bir melez akıllı sistem, lojistik sistemini veya üretim sistemini, öğrenme ve adaptif davranma özelliklerinden dolayı daha gerçekçi olarak modellemede kullanılabilir. Bu örneklerde, melez akıllı bir sistem oluşturmak için en azından bir akıllı teknoloji diğer yaklaşımlarla bütünleştirilmektedir. Bu tür sistemlerin nasıl bütünleştirilebileceğinin örnekleri Şekil 2.4'te gösterilmiştir.



Şekil 2.4 : Melez akıllı sistemlerin nasıl bütünleşik edileceğinin örnekleri (Dağlı, 1994)

Melez sistemler ve özellikle akıllı melez sistemler üzerine motive olunmasının nedeni; tek bir yaklaşımın gereksinim duyduğu varsayımların tam olarak karşılanamadığı problemleri etkin olarak çözmek için, değişik teknolojilerin gücünü birleştiren bir sistemi ortaya koyabilmektir. Bu teknolojiler, her birinin gücü diğerinin zayıf yönlerini telafi edecek şekilde karıştırılıp birleştirilebilir (Dağlı, 1994).

2.3.2 Lojistik Sistemlerinin Yönetiminde Yapay Sinir Ağı Uygulamaları

Son on yılda, bilişsel ve nöral bilimlerdeki gelişmelerin yanı sıra, yeni ağ topolojilerinin, yeni aktivasyon işlevlerinin ve yeni öğrenme algoritmalarının ortaya konması neticesinde, yapay sinir ağlarına ilişkin çalışmalarda heyecan verici bir diriliş olmuştur. Karmaşık problemlerin çözücüsü olarak yapay sinir ağları pek çok farklı uygulamalarda kullanılmıştır. Sinirsel bilgi işlemenin arzu edilen özellikleri, karmaşık problemlerin çözümünde YSA'yı cazip yapmıştır (Obaidat, 1998; Wong ve diğ., 2000; Smith ve Gupta, 2000). Sinir ağları uygulamalarının başlangıçtaki başarıları, sanayi ve işletmelerin tazelenmiş ilgilerine ilham vermiştir.

Son yıllarda önemli ilerlemeler kaydedilmiş olmasına rağmen, pek çok problem henüz tamamıyla çözülememiştir ve sinir ağlarının araştırma ve geliştirilmesinde hala daha fazla çalışmaya ihtiyaç duyulmaktadır. Sinir ağları Yöneylem Araştırması ve Endüstri Mühendisliği (YA/EM) ile pek çok yakın bağa sahiptir (Sharda ve Wang, 1996). YA/EM perspektifinden sinir ağları üç fırsat sunmaktadır:

1. Sinir ağları, eniyileme problemlerini çözmede geleneksel YA algoritmaları veya sezgisel yaklaşımlara alternatiftir.
2. Sinir ağları, biyolojiden esinlenilmiş istatistiksel metotlardır.
3. Sinir ağları, Yöneylem Araştırması algoritmalarının uygulanabileceği bir problem alanı da sunar.

YSA'nın başlıca uygulama alanları aşağıdaki gibi sınıflandırılabilir:

- Sınıflandırma, kümeleme, tanı ve işbirliği
- Eniyileme
- Regresyon ve/veya genelleme
- Örüntü tamamlama

2.3.2.1 Tesis Yeri Seçimi

Yapay sinir ağları ve yasaklı arama, ayrık eniyileme problemlerinin çözümü için son zamanlarda ortaya çıkan iki çok önemli tekniktir. Sinir ağları, güçlü paralel donanımlarda cazip uygulanabilirlik kalitesine sahipken, yasaklı arama güçlü bir genel arama yöntemi olarak ön plana çıkmaktadır. Yasaklı Yapay Sinir Ağları (YYSA), donanım uygulanılabilirliğine sahip güçlü bir paralel ve analog genel arama stratejisi oluşturmak için, yasaklı aramanın kısa dönemlik hafıza bileşeninin bir analog versiyonunun sinir ağları ile bütünleştirilmesidir. YYSA'da, yasak

statüsündeki karar elemanlarının devamlılığının sağlanmasının yanı sıra, yasaklı listesine girecek elemanın seçimi de sinirsel faaliyetlerle başarılır (Vaithyanathan ve diğ., 1996).

Eniyileme problemlerine sinir ağlarının uygulanabilirliği ilk olarak Hopfield ve Tank (1985) tarafından gösterilmiştir. Bununla birlikte, literatürde tekrar tekrar rastlanılabileceği ve araştırmacılar tarafından da kaydedildiği gibi, sinir ağlarının yazılım benzetimi nadiren iyi performans gösterir (Aarts ve Korst, 1989). Sinir ağlarının en üst potansiyelini fark edebilmek için en azından tümleşik eniyileme uygulamalarında donanım gerçeklemesi gereklidir. Tamamen ekonomiksel bakış açısıyla, donanım imalini gerçekleştirmek için, önemli bir yatırım yapmadan önce şu iki önemli konunun ortaya konması gerekir:

- Sinir ağının yazılım benzetimi, en azından orta büyüklükteki problemlerin çok değişik versiyonları için, oluşturulan çözümün kalitesi bağlamında, sürekli olarak iyi performans gösterebilmelidir.

- Sinir ağının çatısı, değişik problem tiplerine kolaylıkla genişletilebilecek yeterlikte genel olmalıdır.

Yasaklı arama, YYSA'da kritik bir rol oynar ve bu çatı içinde uygulanması, daha önceki sinir ağları yaklaşımlarından önemli ölçüde ayrılmasına yol açar. Tek bir tesisin kuruluş yerinin seçiminde bir yasaklı eleman, yüksek sabit ve tahsis maliyetlerine sahip bir yerin karşılığı olabilir. Bu yer, yasaklı listesine girdiğinde listeden çıkarılıncaya kadar müşteriler bu yere tahsis edilmeyecektir (Vaithyanathan ve diğ., 1996).

YYSA tesis yeri seçimi problemine uygulandığında, H-T ağlarından ortalama %85 daha iyi sonuç elde edilmiştir. Performanstaki bu iyileşme birincil olarak, sinir ağının arama uzayının yasaklı arama prensipleri kullanılarak daha verimli bir şekilde araştırılması neticesinde elde edilmiştir. Çözülen problemlerde, bu sinir ağı tutarlı olarak iyi performans göstermiştir. Unutulmamalıdır ki yasaklı sinir ağlarının parametreleri çözümün kalitesi açısından önemlidir. Örneğin, yineleme (arama) sayısı problemin büyüklüğü göz önüne alınmaksızın 100 olarak belirlenmiştir. Büyük problemler için bu sayı artırıldığında çözümün kalitesi iyileşebilecektir. YYSA'nın tasarım esnekliğinin, pek çok tümleşik eniyileme problemine uygulanabilir olması açısından çok fazla olduğu da unutulmamalıdır. Farklı problemlerde arama sürecine yardımcı olması için probleme özgü sezgisel yaklaşımlar da bu çatıya dahil edilebilir.

2.3.2.2 Konum-Tahsis Problemleri

Konum-Tahsis problemleri, birden fazla tesisin, talep merkezlerinden oluşan bir gruba hizmet vermesinin gerekli olduğu durumlarda söz konusudur. Konum-Tahsis problemi pek çok pratik uygulamada (acil servis sistemleri, dağıtım, telekomünikasyon ağları, kamu hizmetleri vs.) karşılaşılan bir problemdir. Birden fazla tesisin konumlandırılması ile ilgilenildiğinde, sıklıkla bununla ilgili bir tahsis problemi de söz konusu olur. Bu problemlerin eş zamanlı olarak ele alınmaları zorunlu olmakla birlikte, tesis konumu bir kez belirlendikten sonra tahsis problemi daha basit olarak çözülebilir. Tersine, bir tahsis çözümü verildiğinde ise, problem bağımsız tek tesis yeri seçimi problemlerine ayrıştırılabilir.

Talep merkezleri ayrık veya verilen bir alan için yine bilinen bir olasılığa göre dağılmış olabilir. Arz merkezleri ise genellikle birbirlerinden bağımsız olarak çalışıyor kabul edilir. Eniyileme kriteri sıklıkla problemin özel/kamu sektörü olmasına ve olağan/sıra dışı ürün veya hizmet sunulmasına bağlıdır.

Lozano'nun gerçekleştirdiği çalışmada; bağımsız, kapasite sınırlaması olmayan arz merkezleri ile servis verdiği sürekli bir uzayda yer alan talep merkezlerinin konumlandırılması sorunuyla ilgilenilmiştir (Lozano ve diğ., 1998). Yerleştirme maliyetlerinin, kat edilen Öklit mesafesinin karesi ile orantılı olduğu kabul edilmiştir. Bu problem, bazı Kümeleme Analizi ve Vektör Nicelendirme problemleri ile eşdeğer gibi görülmüştür. Böyle bir ilişkilendirme Lozano'yu, Yapay Sinir Ağlarının yapısal özelliklerini ortaya koyma kapasitesine sahip, girdi verileri için yerel uyarlama kurallarına dayalı bir kendi kendini yapılandırma sürecinin söz konusu olduğu, Kohonen haritalarının kullanılmasına yöneltmiştir.

Konum-Tahsis problemleri ile Kümeleme Analizi arasındaki ilişkinin farkına çok önceleri varılmıştır (Hartigan ve Wong, 1979). Diğer yandan, kendi kendini düzenleyen ağların (SOM) kümelemede kullanılması açık ve kolaylıkla anlaşılabilirdir. Konum-Tahsis problemlerini çözmede kullanılan SOM'da girdi deseni, talep merkezlerinin K-boyutlu konum vektörüdür ve D_i olasılık dağılım fonksiyonuna göre rasgele dağılır. Çıktı katmanı ise konumlandırılacak M-tesis ile ilgilidir. Her birisi için ağırlık vektörleri, atandıkları arz merkezinin konumu ile ilişkili olarak belirlenir. Öğrenme kuralının etkisi, arz merkezlerinin tamamını talep merkezlerine doğru çekmektir. Ancak, sadece en yakın arz merkezi ve onun komşuları önemli ölçüde etkilenir. Başlangıçta, sıralama evresini hızlandırmak amacıyla komşuluk alanı, çıktıların çoğunluğunun her bir yinelemede ağırlıklarını

uyarlamalarını sağlayacak ölçüde büyük olmalıdır. Daha sonra, öğrenme hızı yavaş yavaş azaltılarak, konumda daha ince ayarlamaların yapılması sağlanır.

Enküçüklenmeye çalışılan orijinal amaç fonksiyonu, mesafelerin karelerinin ağırlıklı ortalaması olmasına rağmen bu yaklaşım, Öklit mesafe fonksiyonu için de kullanılabilir.

2.3.2.3 Araç Atama Problemi

Araçlar taşıma talebine atanırken, genellikle bilinen bir mesafeye eldeki araçlarla gönderilmek üzere, bilinen miktardaki yüklerin atanması işlemini planlamacılar yapısal olarak bulanık kurallara göre gerçekleştirirler. Bulanık sistemler, karmaşık süreçlerin kontrolü için eğitilebilecek öğrenme yetenekleri ile donanmışlardır. Genellikle planlamacıdan alınan çok az ham veri ile başlanılır veya planlamacının davranışları gözlemlenmek suretiyle kurallar çıkarılabilir. Vukadinovic'in çalışmasında, daha iyi performans elde edebilmek için bir bulanık sistemi arıtmada ve uyarlamada yapay sinir ağı kullanılmıştır (Vukadinovic ve diğ., 1998). Bulanık sistemler sayısal sistemler olmasına rağmen, gözlemlenen planlamacı davranış ve bilgilerine dayanan dilsel kontrol stratejilerini bir otomatik kontrol stratejisine dönüştürür. Fakat bulanık çatı, bilgiyi elde etmenin ağır yükünden analisti kurtarmaz. Bir bulanık sistemin oluşturulmasındaki ana problemler, belirgin sezgisel kontrol kurallarının çıkarılması ve girdi/çıktı değişkenlerinin üyelik fonksiyonu parametrelerinin belirlenmesidir. Böylece, geliştirilen bulanık sistemin performansı mevcut planlamacı sayısına ve yapılan mülakat ile onların atama davranış biçimlerini ortaya koyabilmeye bağlı olacaktır.

Vukadinovic'in yaptığı çalışmanın neticesinde gerçek veriler üzerinde yapılan uygulamada, ilk bulanık sistemleri ayarlamada, denetçili öğrenme yetenekleri ile birlikte ileriye doğru beslemeli adaptif sinir ağlarının kullanılabileceği gösterilmiştir.

Nygard (1990), modüler bir sinir ağı sistemi olarak adlandırdıkları taşıt rotalama problemleri için çözüm tekniği seçiminde kullanılan bir sistem geliştirdiler. Bu sistem geriye-yayılımlı 7 sinir ağından oluşmaktadır. Ağlardan birinin saklı katmanı özellik çıkarımında ve ikinci ağ ise geriye kalan beş ağın hangisinin çözüm tekniği önerisini yapmada kullanılacağını belirlemede kullanılmaktadır

2.3.2.4 Kümeleme (Gruplandırma) Analizleri

Kümeleme analizi (bir veri kümesindeki doğal alt-grupların belirlenmesi), pek çok bağlamda kullanılan önemli bir istatistiksel metodolojidir. Bugün kullanılan hiyerarşik kümeleme metotlarındaki ana problem, yoğun olarak izole edilmiş kümelerin ideal şartlarından ampirik veriler uzaklaştığında ortaya çıkan sınıflandırma hatalarına olan eğilimdir. Pek çok ampirik veri kümeleri, grupların oluşturulmasını allak bullak eden yapısal bozuklukları içerir.

SOM ağları, pazar bölümleme, kredi analizi, kalite problemleri ve hareket problemleri gibi karmaşık verilerin gruplandırılması analizlerini gerektiren kararların kalitesini iyileştirebilir. Karmaşık ampirik verilerde grup üyeliklerinin belirlenmesi zor bir problemdir ve dağılma, ilgisiz bilgi, düzenli olmayan küme yoğunluğu gibi veri bozuklukları ile daha da karmaşık hale gelir. Deneysel sonuçlar, bugün yaygın olarak kullanılan 7 hiyerarşik kümeleme analizi ile kıyaslandığında SOM metodunun gayet açık olarak çok iyi sonuç verdiğini göstermektedir (Pauler ve diğ., 1999). Dahası, SOM ağının performansının verilerdeki yapısal bozukluklara rağmen sapasağlam olduğu gösterilmiştir. Verilerdeki yayılma düzeyi arttıkça, SOM ağının performans avantajı, diğer hiyerarşik gruplandırma metotlarına göre daha da üst seviyeye çıkar.

Mangiameli'nin yaptığı çalışmada, SOM ağı test edilen 252 veri kümesinin 191'i için en doğru metottur (%75.8). SOM ağı, veri kümelerinin %87.3'ü için kesinlikte birinci veya ikinci sıradadır. Yayılımı çok yüksek veriler için SOM ağı, %90.2-%91.5 oranı ile birinci veya ikinci sıradadır. Diğer metotlar, ortalama %30-70 arasında doğruluğa sahipken SOM ağı %85 kesinliğe sahiptir. İncelenen 252 veri kümesinden sadece altısında SOM ağı ile kötü sonuç elde edilmiştir (Mangiameli ve diğ., 1996).

SOM ağı sadece etkin değil, aynı zamanda kullanımı da kolay bir tekniktir. Gerçekte SOM ağı, sinir ağları konusunda kapsamlı tecrübeye sahip analiste ihtiyaç duymaz. Ağın yapısında, nöron sayısında veya ilişkilendirme deseninde karmaşıklık yoktur. Gruplandırma sonuçları ağın başlangıç öğrenme katsayısına duyarlı da değildir. Kısaca, karmaşık veri olması durumunda SOM ağı, gözlem verilerinin gruplandırılmasında çok daha kesin sonuç verir. Dahası, olası veri eksikliklerinde SOM ağının sağlamlılığı diğer sezgisel yöntemlerden daha fazladır (Kuo ve diğ., 2002).

2.3.2.5 Pazar Paylaşımı

Sinir ağlarının ses ve görüntü işleme gibi mühendislik alanlarının değişik uygulamalarındaki başarısından hareketle, diğer alanlarda da benzer problemlere uygulamayı düşünmek doğaldır. Yönetim biliminde ortaya çıkan ilgili bir problem de gruplandırma tekniklerinin kullanıldığı pazar bölümlemedir. Balakrishnan, özel bir sinir ağının (Frekans-Duyarlı Rekabetçi Öğrenme Algoritması-FSCL) stratejik pazar kararlarının oluşturulması için verilerin gruplandırılmasındaki yeteneğini araştırmıştır (Balakrishnan ve diğ., 1996). Bu çalışmada, FSCL ile k-ortalamalar yöntemi kümeleme tekniğinin performans karşılaştırmalarını yapmışlardır. Yapılan uygulamada, her iki teknik oldukça farklı gruplandırmalar oluşturmuşlardır. Uygun metodoloji hakkında uyuşmazlığın nedenini ortaya koymak amacıyla, veri koleksiyonu ve ölçüm hatalarını yansıtacak, değişen veri kalitesini taklit edecek oldukça büyük bir deneysel tasarımla, bilinen gruplandırma sonuçları ile birlikte benzetimi yapılmış veriler kullanılarak, karşılaştırmalı bir performans araştırması yapılmıştır. Bu çalışmaların neticesine göre; bu iki metodolojinin kombinasyonu (FSCL ağının sonuçları k-ortalamalar yöntemine girdidir) daha yönetsel olarak bölümleme tasarısını kavramayı sağlar gibi gözükmektedir. Bu çalışmayla ilgili bazı sınırlamalar da söz konusudur. Birincisi, veri noktalarının sayısı sinir ağları için kullanılanlarla karşılaştırıldığında göreceli olarak az kabul edilebilir. İkincisi, hata içermeyen veriler üzerinde analizler yapmak ve verilerin grup sayılarını doğru olarak bilmek, k-ortalamalar yaklaşımında en ciddi problem olan başlangıç grup kaynağının seçiminde olduğu gibi lehte olarak ön yargılı hareket edilmesine yol açabilir. Doğru grup sayısı ve hata içermeyen verilerin kullanılması, kaynak seçimi problemini önemli ölçüde azaltır. Sınırlamalara rağmen k-ortalamalar algoritması iyi çalışmıştır.

Seçilen FSCL ağı iyi performans göstermese de bu alanda ilerde yapılacak çalışmalara temel sağlaması açısından önemlidir. FSCL ağında kullanılan gözetimsiz öğrenme yaklaşımı, kümeleme analizinde sinir ağlarının kullanılmasını doğal bir başlangıç noktası yapar. Daha önemlisi, yanlış sınıflandırmayı enküçüklemek için ihtiyaç duyulan bilgiler, her bir kümeyi gösteren birçok ağırlık vektörünü içerir. Bu vektörler, bir kümenin merkezinden çok sınırlarını tanımlayan vektörlerdir.

Sinir ağları için bir diğer problem parametrelerin seçimidir. Gözetimsiz öğrenme çatısında, bir veri kümesi için uygun ağ parametrelerinin nasıl seçileceğine dair açık bir yanıt bulmak zordur.

FSCL ağında oluşturulan kümelerin kalitesi, daha ilgili niteliklerin eklenmesi ile geliştirilebilir. Daha fazla faydalı bilgiler FSCL ağı için mevcut olduğunda, daha iyi iyileşme olacağı söylenebilir. Benzetim veri sonuçlarına bakarak, FSCL metodunun kümeleme amaçlı olarak oldukça dikkatli kullanılması gerektiği söylenebilir. Pazar verileri için sinir ağları metodolojisi ile k-yöntemi yaklaşımının kombinasyonu kullanıldığında daha iyi sonuçlar elde edilmiştir.

Gruplandırmaya dayalı bir diğer çalışma da Hruschka tarafından gerçekleştirilmiştir (Hruschka ve Natter, 1990). Bu çalışmada, özel olarak tasarlanmış ileri beslemeli yapay sinir ağı ile bir saklı tabaka içeren k-ortalamalar kümeleme tekniğinin performansları karşılaştırılmıştır. Analizi yapılan veri kümesi, farklı kullanım durumlarında ürün kategorilerinin kullanımını içermektedir. İleri beslemeli sinir ağı modeli uygun testlerle onaylanan iki bölümlü bir sonuç vermiştir. Diğer yandan, k-ortalamalar algoritması daha güçlü küme yapılarının bulunmasında başarısız olmuştur.

Pazar paylaşımında k-ortalamalar algoritması ile SOM ağını bütünleştiren başka bir çalışmada, üç kümeleme yaklaşımı karşılaştırılmıştır (Kuo ve diğ., 2002). Bu yaklaşımlar; klasik iki-aşama yöntemi, kendi kendini düzenleyen ağlar ve Kuo ve diğ.'nin önerdiği iki-aşama yöntemi. Önerilen iki-aşama yöntemi kendi kendini düzenleyen ağlar ile k-ortalamalar yönteminin bir birleşimidir ve bu yöntem klasik iki-aşama yöntemine göre biraz daha iyi sonuç vermiştir.

2.4 Dağıtılmış Akıllı Sistemlerin Lojistik Sistemlerinde Kullanılması İhtiyacı

Literatürde dağıtılmış problem çözme yaklaşımını temel alan İşbirlikçi Tedarik Zinciri yaklaşımı, Etmen Temelli TZY sistemleri gibi bütünleşik tedarik zinciri sistemleri veya lojistiğin alt alanlarını birbirinden bağımsız olarak sezgisel yöntemlerle ayrı ayrı eniyilemeye çalışan yöntemler geliştirilse de, bu alt alanları bir bütün olarak ele alan ve sinir ağları ile denetleyen, eniyileyen gezgin etmen temelli bir sistem tasarlanarak gerçeklenmemiştir. Ortaklar, tedarikçiler ve müşteriler arasında arabuluculuk yapan ve etmen koordinasyon sistemlerinin farklı düzeylerinde öğrenme yeteneklerine sahip dağıtılmış akıllı sistemleri (örneğin, yapay sinir ağlarını) kullanan bir altyapıya ihtiyaç duyulduğu görülerek, bu çalışmada böyle bir sistem tasarlamak ve gerçeklemek hedeflenmiştir.

3. ADAPTİF SİSTEMLER VE YAPAY SİNİR AĞLARI

Bir sistemin tasarımında, ilk olarak problem ele alınarak incelenir ve modellenir, spesifikasyonlar ortaya konur ve akabinde belirlenen kriterleri sağlayacak şekilde bir sistem oluşturulur. Önemli olan nokta, dış koşullar değişse dahi sistemin mevcut spesifikasyonlara uygun olması ve her zaman tasarlanmış parametre kümesini kullanmasıdır.

Bu bölümde ortaya konacak sistem tasarımı yaklaşımı ise biyolojiden esinlenilen ve adaptasyona dayanan farklı bir yaklaşımdır. Başlangıçta sistem parametreleri çok fazla hataya yol açacak şekilde istenilenin dışında olabilir. Ancak, hatadan alınan geri besleme ile sistem olabildiğince hatayı azaltacak şekilde parametrelerini değiştirir. Bir adaptif sistem, sadece kendisinden beklenen işleri gerçekleştirmek değil, aynı zamanda parametrelerini adapte etmesini sağlayacak bir alt-sistemi de içermesi gerektiğinden, çok karmaşıktır. Fakat gelecekteki veriler değişse dahi bu tasarım metodolojisi, sistem parametrelerini mümkün olan en iyi performansı elde edecek şekilde değiştirecektir. Ayrıca, aynı sistem birden fazla problem için kullanılabilir.

3.1 Sinirsel ve Adaptif Sistemler

Yapay Sinir Ağları kavramı, beynin çalışma ilkelerinin sayısal bilgisayarlar üzerinde taklit edilmesi fikri ile ortaya çıkmış ve ilk çalışmalar beyni oluşturan *biyolojik hücrelerin* (nöronlar) matematiksel olarak modellenmesi üzerinde yoğunlaşmıştır. Bu çalışmaların ortaya koyduğu bulgular, her bir sinir hücresinin komşu sinir hücrelerinden bazı bilgiler aldığı ve bu bilgilerin biyolojik sinir sistemi dinamiğinin öngördüğü biçimde bir çıktıya dönüştürüldüğü şeklindedir. Bugün YSA olarak isimlendirilen alan, birçok sinir hücresinin belirli biçimde bir araya getirilip bir işlevin gerçeklenmesi üzerindeki, yapısal olduğu kadar matematiksel ve felsefi sorunlara da yanıt arayan bir bilim dalı olmuştur. İnsan beyninin çalışma mekanizmasını taklit etmeye çalışan bu sistemler, her ne kadar günümüz teknolojisinin ürettiği, birim işlem zamanı nanosaniyeler mertebesinde olan silikon lojik kapılar ile gerçeklenebilseler de, insan beyninin birim işlem zamanı milisaniyeler

mertebesindeki biyolojik sinir hücrelerinin toplu biçimde ele alındıklarındaki işlevselliklerinden çok uzakta kalırlar. YSA'nın karar hızı açısından insan beyni ile yarışabilecek aşamayı henüz kat edememiş olmasına karşın, karmaşık eşleştirmelerin hassas bir biçimde gerçeklenebilmesi ve *yapısal gürbüzlüğe* (structural robustness) sahip olması nedeniyle, uygulama alanları gün geçtikçe genişlemektedir (Lippmann, 1987).

YSA'nın çözebileceği problem uzayı, aslında insan beyninin çözebildiği problem uzayının oldukça kısıtlanmış bir alt kümesidir ve insan beyninin bilgiyi işleme, kavramları ilişkilendirme ve çıkarım mekanizmalarındaki üstünlüğü tartışmasızdır. Bununla birlikte, YSA'yı çekici kılan bir takım temel özellikler vardır:

- Pek çok YSA, veriye dayalı olarak bağlantı ağırlıklarının ayarlandığı bir çeşit **eğitme** kuralına sahiptir. Diğer bir deyişle, YSA örneklerden öğrenir (çocukların köpekleri, daha önce kendisine gösterilen köpeklerden hareketle tanıması gibi) ve **genelleme** yeteneği gösterir, başka bir deyişle ağ yapısının eğitim esnasında kullanılan sayısal bilgilerden eşleştirmeyi tanımlayan kaba özellikleri çıkarır ve böylelikle eğitim sırasında kullanılmayan girdiler için de anlamlı yanıtlar üretebilir.

- YSA, normalde bileşenlerinin gerçekleştirdiği hesaplamalar diğerlerinkinden büyük ölçüde bağımsız olduğundan muazzam bir **paralellik** potansiyeline sahiptir ve toplamsal işlev yapısal olarak dağılmıştır (Haykin, 1994). Diğer bir deyişle birçok nöron eşzamanlı olarak çalışır ve karmaşık bir işlev çok sayıda küçük nöron aktivitesinin bir araya gelmesinden oluşur. Bu da, zaman içerisinde herhangi bir nöronun işlev dışı kalması durumunda ağ başarımının dikkate değer ölçüde etkilenmeyeceği anlamına gelir.

- **Hataya toleranslıdır.** Başka bir deyişle, gürültülü ve eksik verilerden anlamlı düzeyde doğru çıktılar üretebilir, oysa geleneksel bilgisayarlar genellikle doğru verilere gereksinim duyarlar.

- Tasarlanması ve gerçeklenmesi pahalı değildir, ancak eğitilmesi yoğun hesaplamayı gerektirir.

- Özellikle çözümü karmaşık ve belirtilmesi zor olan ve çok fazla miktarda veri içeren problemler için uygundur.

YSA, ilke olarak herhangi bir fonksiyonu hesaplayabilir; başka bir deyişle normal bir sayısal bilgisayarın yapabileceği her şeyi (Valiant, 1988; Siegelmann ve Sontag,

1999; Orponen, 2000; Sima ve Orponen, 2001) veya bazı varsayımlar yapılarak daha da fazlasını (Siegelmann, 1998; Hadley, 1999) yapabilir. Pratikte elde çok verinin olduğu, fakat zor ve çabuk kuralların kolaylıkla uygulanamadığı özellikle sınıflandırma ve fonksiyon tahmini/eşleştirme problemlerinde kullanılabilir.

3.1.1 Yapay Sinir Ağı Türleri

YSA'da kullanılan öğrenme algoritmalarına bakıldığında, literatürde birçok öğrenme algoritmasından bahsedildiği ve bunların öğrenme denen olguyu, matematiksel kurallarla ölçülebilir büyüklüklere dönüştürerek bir başarım/maliyet ölçütünün oluşturulmasına ve bu ölçütün zaman içerisinde artırılmasını/azaltılmasını sağlayacak parametre değişikliklerinin hesaplanmasına dayandıkları görülebilir. Pek çok yapay sinir ağı çeşidi olduğundan ve sürekli olarak yenileri bulunduğundan tam olarak kaç çeşit olduğunu söylemek pek mümkün değildir. Ancak en çok bilinen yöntemler için sınıflandırma yapmak mümkün olabilir. Tasarım esnekliğinin ana teması, öğrenme algoritmalarındaki parametre güncelleme işlemi için türetilen bilginin hangi yöntemlerle oluşturulduğudur. Öğrenme mekanizması açısından YSA *gözetimli* ve *gözetimsiz* olmak üzere iki ana gruba ayrılabilir:

- **Gözetimli (*supervised*) öğrenme:** Doğru sonuçlar (hedef değerler, istenen çıktılar) biliniyor ve YSA'nın çıktı değerlerini hedef değerlerle aynı olmasını sağlayacak şekilde ağırlıklarını güncelleştirmesi için eğitme esnasında ağa girdi olarak veriliyorsa bu tip öğrenme gözetimli öğrenmedir. Eğitmeden sonra, sadece girdi verileri uygulanarak YSA test edilir ve doğru hedef değerlere ne kadar yakın değerler ürettiği görülür.

- **Gözetimsiz (*unsupervised*) öğrenme:** Eğitme esnasında istenen çıktı değerleri mevcut değildir. Gözetimsiz YSA genellikle veri boyutunun azaltılması veya kümeleme analizleri gibi daha çok sistemin geçmişte karşı karşıya kaldığı veri kümesinin içerdiği istatistiksel bilgilerin çıkarsanmasında kullanılır.

Gözetimli ve gözetimsiz öğrenme yöntemleri arasındaki ayrım her zaman çok açık değildir. Gözetimsiz yöntem, bir olasılık dağılımını kabaca öğrenir ve bu bilgiyi ileriye dönük tahmin yapmada kullanabilir. Ayrıca, gözetimli yöntemler iki alt çeşide ayrılabilir: *kendi kendine ilişkilendirme* (auto-associative) ile öğrenme ve *karşıt-ilişkilendirme* (hetero-associative) ile öğrenme. Birincisinde, girdiler ile hedef değerler aynı iken, ikincisinde hedef değerler genellikle girdilerden farklıdır. Pek çok

gözetimsiz yöntem kendi kendine ilişkilendirmeli gözetimli yönteme karşılık gelir (Deco ve Obradovic, 1996).

Tasarımda ikinci seçenek mimari üzerinedir ve *ileri beslemeli* ve *geri beslemeli* olmak üzere iki alt başlıkta değerlendirilebilir:

- **İleri beslemeli (*feedforward*) YSA:** Eğer verinin akışı bir çevrim oluşturmayacak şekilde ileriye doğru ise bu yapıya sahip ağ modelleri ileri beslemelidir. Bu tip ağlar, yapay sinir ağları için geliştirilmiş algoritmalara ek olarak, birçok sayıda geleneksel sayısal yöntemler kullanılarak da eğitilebilirler.

- **Geri beslemeli (*feedback*) veya yinelemeli (*recurrent*) YSA:** Ağ üzerinde geri besleme bağlantıları vardır. Bu tip YSA'da, ağa uygulanan her bir girdi değeri için YSA bir sonuç üretmeden önce, potansiyel olarak uzun bir süre yineleme yapmak zorundadır. Bu tip ağları eğitmek ileri beslemelileri eğitmekten daha zordur.

Bazı yapay sinir ağları (*kazanan-hepsini-alır* gibi) hem ileri beslemeli hem de geri beslemeli olarak gerçeklenebilir.

YSA kabul ettikleri veri türüne göre de *kategorik* ve *nicel* olmak üzere iki ana gruba ayrılabilir:

- **Kategorik değişkenler:** Sadece sonlu sayıda olası değerler alırlar ve genellikle her bir kategoride birkaç veya daha fazla değer yer alır. Kategorik değişkenler sembolik değerler (örneğin, "kırmızı", "yeşil" ve "mavi") olabilir, ancak ağa bu değerler uygulanmadan önce sayısal olarak kodlanmalıdır. Hedef değerleri kategorik değişkenler olan gözetimli ve gözetimsiz öğrenmenin her ikisi de sınıflandırma olarak adlandırılır.

- **Nicel değişkenler:** Bazı niteliklerin (örneğin, uzunluk gibi) sayısal ölçüm değerleridir ve ölçümler, en azından ölçümler arasındaki bazı aritmetik ilişkiler ölçülen nesnelerin nitelikleri arasındaki ilişkiyi yansıtacak şekilde yapılmalıdır. Hedef değerleri nicel olan gözetimli öğrenme regresyon olarak adlandırılır.

Aşağıda yaygın olarak kullanılan YSA'nın genel bir sınıflandırılması yapılmıştır (Tablo 3.1).

Tablo 3.1 : Yapay sinir ağlarının sınıflandırılması

Öğrenme Mekanizması	Mimarisi	Yapısı	Tipi	Kaynaklar
Gözetimli	İleri Beslemeli	Doğrusal	Hebbian	▪Hebb (1949) ▪Fausett (1994)
			Perceptron	▪Rosenblatt (1958) ▪Minsky ve Papert (1969/1988) ▪Fausett (1994)
			Adaline	▪Widrow ve Hoff (1960) ▪Fausett (1994)
			Yüksek Mertebe	▪Bishop (1995)
			İşlevsel Bağlantı	▪Pao (1989)
		Çok Katmanlı Algılayıcı (MLP): ▪Bishop (1995) ▪Reed ve Marks (1999) ▪Fausett (1994)	Geriye Yayılım	▪Bishop (1995) ▪Reed ve Marks (1999) ▪Fausett (1994)
			Kademeli Dizi Korelasyonu	▪Fahlman ve Lebiere (1990) ▪Fausett (1994)
			Quickprop	▪Fahlman (1989)
			RPROP	▪Riedmiller ve Braun (1993)
		Radyal Temelli Fonksiyonlar: ▪Bishop (1995) ▪Orr (1996)	OLS: Dikey En Küçük Kareler	▪Chen ve diğ. (1991)
		CMAC: Serebellar Modeli Eklem Kontrol Edicisi	▪Albus (1975) ▪Brown ve Harris (1994)	
		Sadece Sınıflandırma	LVQ: Vektör Nicelendirmeyi Öğrenme	▪Kohonen (1988) ▪Fausett (1994)
			PNN: Olasılıksal Yapay Sinir Ağı	▪Specht (1990) ▪Masters (1993) ▪Hand (1982) ▪Fausett (1994)
		Sadece Regresyon	GNN: Genel Yapay Sinir Ağı	▪Specht (1991) ▪Nadaraya (1964) ▪Watson (1964)
	Geri Beslemeli	Yinelemeli Zaman Serileri	Zaman Serisinde Geriye Yayılım	▪Werbos (1990)
			Elman	▪Elman (1990)
			FIR: Sonlu İtki Tepkisi	▪Wan (1990)
			Jordan	▪Jordan (1986)
			Gerçek Zamanlı Yinelemeli	▪Williams ve Zipser (1989)
			Yinelemeli Geriye Yayılım	▪Pineda (1989) ▪Fausett (1994)
			TDNN: Zaman Geciktirmeli SA	▪Lang, Waibel ve Hinton (1990)

Tablo 3.1 : (devamı) Yapay sinir ağlarının sınıflandırılması

Öğrenme Mekanizması	Mimarisi	Yapısı	Tipi	Kaynaklar
Gözetimli	Geri Beslemeli	BAM: İki-yönlü İlişkilendirici Hafıza		▪Kosko (1992) ▪Fausett (1994)
		Boltzman Makinesi		▪Ackley ve diğ. (1985) ▪Fausett (1994)
	Rekabetçi	ARTMAP		▪Carpenter ve diğ. (1991)
		Bulanık ARTMAP		▪Carpenter ve diğ. (1992) ▪Kasuba (1993)
		Gaussian ARTMAP		▪Williamson (1995)
		Karşıtyayılım		▪Hecht-Nielsen (1987; 88; 90) ▪Fausett (1994)
		Neocognitron		▪Fukushima, Miyake ve Ito (1983) ▪Fukushima (1988), ▪Fausett (1994)
Gözetimsiz	Rekabetçi	VQ: Vektör Nicelendirme	Grossberg	▪Grossberg (1996)
			Kohonen	▪Kohonen (1984)
			Conscience	▪Desieno (1988)
		Kendi Kendini Düzenleyen Haritalar	Kohonen	▪Kohonen (1995) ▪Fausett (1994)
			GTM	▪Bishop ve diğ. (1997)
			Yerel Doğrusal	▪Mulier ve Cherkassky (995)
		Adaptif Rezonans Teorisi	ART 1	▪Carpenter ve Grossberg (987a) ▪Moore (1988) ▪Fausett (1994)
			ART 2	▪Carpenter ve Grossberg (987b) ▪Fausett (1994)
			ART 2A	▪Carpenter ve diğ (1991a)
			ART 3	▪Carpenter ve Grossberg (1990)
			Bulanık ART	▪Carpenter ve diğ. (1991b)
		DCL: Diferansiyel Rekabetçi Öğrenme		▪Kosko (1992)
	Boyut İndirgeme (Diamantaras ve Kung (1996))	Hebbian		▪Hebb (1949) ▪Fausett (1994)
		Oja		▪Oja (1989)
		Sanger		▪Sanger (1989)
		Diferansiyel Hebbian		▪Kosko (1992)
	Kendi Kendine İlişkilendirme	Doğrusal oto-ilişkilendiriciler		▪Anderson ve diğ. (1977) ▪Fausett (1994)
		Optimal Beyin Hasarı		
		Hopfield		▪Hopfield (1982), Fausett (1994)
Öğrenmeyen	Hopfield	▪Hertz, Krogh ve Palmer (1991)		
	Eniyileme için farklı ağlar	▪Cichocki ve Unbehauen (1993)		

Parametre güncelleme işleminin zamanlaması da bir diğer seçenektir. Gözetimli öğrenme yaklaşımında parametre güncelleme işlemi, normal çalışma esnasında, anlık gözlemlerden elde edilen bilgi ile yapılıyorsa, buna *eşzamanlı* (on-line) öğrenme denir. Eğer yapay sinir ağı daha önceden belirlenen girdi/çıktı eşleştirmesini gerçeklemeye çalışıyorsa, buna da *zamandan bağımsız* (off-line) öğrenme denir (Piché, 1994).

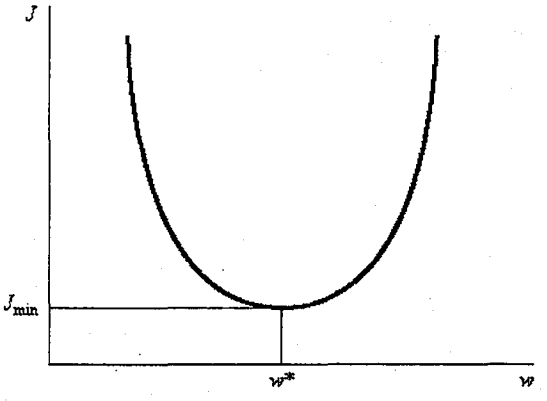
Son olarak parametre güncelleme işlemi için iki seçeneğin varlığından bahsedilebilir. Eğer ağ parametreleri, eğitim çiftlerinin tamamının ağ üzerinden geçirilip her bir geçişte hesaplanan değişim miktarlarının toplamı ile güncelleniyorsa *grup* (batch, epoch), her bir eğitim çifti için hesaplanan değişim miktarı o anda uygulanıyorsa *bireysel* (incremental, instantaneous, pattern) güncellemeden bahsedilebilir.

3.1.2 Performans Yüzeyi

Performans yüzeyi, ağırlıkların (karar değişkenlerinin) adaptasyonunun performans kriterini nasıl etkilediğini göz önüne getirmeye yardımcı olan önemli bir araçtır. Örneğin, genel eniyileme problemlerine basit bir örnek olması açısından doğrusal regresyon modeli ($y = wx + b$) için hata kareleri ortalaması (J) ele alınırsa (basitleştirmek için önyargı terimi b = 0 olarak alınmıştır);

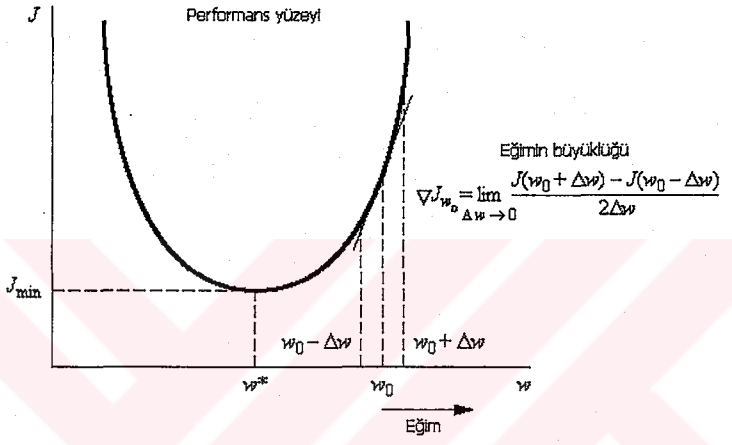
$$J = \frac{1}{2N}\sum_i (d_i - wx_i)^2 = \frac{1}{2N}\sum_i \left(x_i^2 w^2 - 2d_i x_i w + d_i^2\right) \qquad (3.1)$$

J sadece tek bir değişkenin (w) fonksiyonu olacaktır (Şekil 3.1).



Şekil 3.1 : Regresyon problemi için performans yüzeyi (Principe ve diğ., 2000, s.19)

Performans yüzeyinin gradyanı, boyutu w olan ve her zaman J'nin maksimum değişim yönünü gösteren, büyüklüğü performans yüzeyine teğet olan doğrunun eğimine eşit bir vektördür (Şekil 3.2). Eğer performans yüzeyi bir yamaç gibi düşünülürse, yamaçtaki her bir nokta, bu noktadaki en hızlı çıkış yönünü (steepest ascent) gösteren bir gradyan çizgisine sahip olacaktır. Yamaçtan aşağıya bırakılan bir top her zaman gradyan çizgisinin tersi yönünde yuvarlanacaktır (steepest descent). Dipteki eğim sıfır olduğundan gradyanı da sıfırdır, başka bir deyişle hata kareleri ortalaması enküçüklenmiştir (J_{min}). Genelleştirme yapılarak genel eniyileme (enküçükleme) problemleri için de performans yüzeyleri oluşturulabilir.



Şekil 3.2 : Performans yüzeyi ve gradyan (Principe ve diğ., 2000, s.20)

3.1.3 En Hızlı İniş ile Performans Yüzeyinin Araştırılması

Bir fonksiyonun minimumunun araştırılması, gradyan bilgisine dayanan genel yöntemler kullanılarak verimli bir şekilde yapılabilir. Gradyan, arama işlemi açısından iki ana üstünlüğe sahiptir:

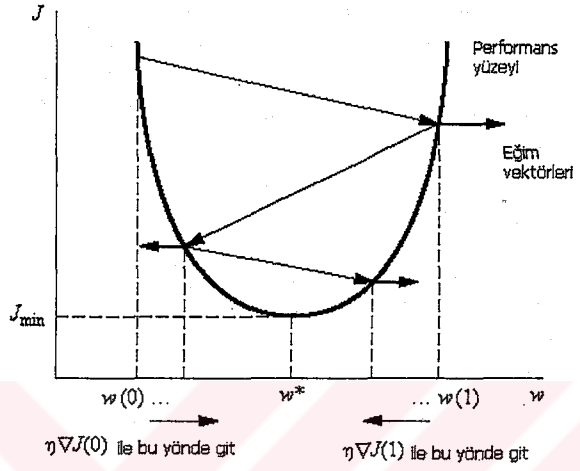
- Gradyan yerel olarak hesaplanabilir.
- Gradyan her zaman maksimum değişim yönünü gösterir.

Eğer hedef minimuma ulaşmak ise, arama işlemi gradyanın tersi yönünde gerçekleştirilmelidir. Bu yüzden arama yöntemi aşağıdaki gibi gerçekleştirilebilir:

Keyfi bir başlangıç ağırlığı $w(0)$ ile arama işlemine başlanılır. Performans yüzeyinin $w(0)$'daki gradyanı hesaplanır ve başlangıç ağırlığı bu gradyanın negatifi ile orantılı olacak şekilde değiştirilir. Bu işlem sonucunda $w(1)$ noktası elde edilmiş olur. Ardından, yeni noktada gradyan hesaplanır ve aynı prosedür tekrarlanır. Yani;

$$w(n+1) = w(n) - \eta \nabla J(n) \tag{3.2}$$

η küçük bir sabit ve $\nabla J(n)$ performans yüzeyinin n'nci yinelemedeki gradyanıdır. Sabit η değeri, arama işlemindeki kararlılığı sürdürebilmek, yani işlem noktasının performans yüzeyinde çok uzağa gitmesini önlemek için kullanılır. Bu arama yöntemi *en hızlı iniş* (steepest descent) yöntemi olarak bilinmektedir (Şekil 3.3).



Şekil 3.3 : Gradyan bilgisini kullanarak araştırma (Principe ve diğ., 2000, s.24)

3.1.4 Gradyanın Tahmin Edilmesi

Adaptif sistem parametrelerini eniyilemek için gradyan kullanılabilir. Ancak, gradyan açık olarak genellikle bilinmez ve bu nedenle tahmin edilmesi gerekir. Geleneksel olarak kullanılan *türevi bulmak için farktan yararlanma* yöntemi çok basit olmakla birlikte çok pratik değildir.

Widrow, 1960'ların sonlarına doğru gradyanı tahmin etmek için, gradyan iniş yöntemi uygulamaları açısından devrim niteliği taşıyan oldukça basit bir yöntem önermiştir. Bu yönteme göre gerçek değeri tahmin etmek için anlık değer kullanılır. Şaşırtıcı olan nokta, gradyanın ağırlık (karar değişkeni) başına sadece bir çarpım işlemi gerçekleştirilerek tahmin edilebilmesidir. Bu gradyan tahmini bizi *en küçük kareler ortalaması* (EKKO) yöntemine götürür. Tahmin gürültülü olmakla beraber adaptasyon süresince gradyandaki gürültü bileşeninin ortalaması alındığından filtrelenir. Widrow'un gradyan tahmini (3.2)'de yerine konursa en hızlı iniş denklemi, EKKO *yöntemi* olarak bilinen aşağıdaki eşitliğe dönüşür (Widrow, 1960).

$$w(n+1) = w(n) + \eta\varepsilon(n)x(n) \tag{3.3}$$

η küçük bir sabittir ve *adım büyüklüğü* veya *öğrenme oranı* olarak adlandırılır. $x(n)$ n'nci yinelemedeki bağımsız değişkeni ve $\varepsilon(n)$ ise regresyon hesaplamadaki tahmin hatasını gösterir.

EKKO yöntemi ile tanımlanan gradyan tahminini (3.3) tek boyuttan çok boyuta genişletmek oldukça basittir:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\varepsilon(n)\mathbf{x}(n) \tag{3.4}$$

İlginç bir özellik olarak, EKKO adaptasyon kuralı çok boyutta da yerel hesaplamaları kullanır. Yani, i'nci ağırlık elemanı için:

$$w_i(n+1) = w_i(n) + \eta\varepsilon(n)x_i(n) \tag{3.5}$$

Yani, i'nci nöron için girdi değeri ile bir önceki adımdaki hata değerinin ve öğrenme oranının çarpımı eski ağırlık değeriyle toplanarak yeni ağırlık değeri hesaplanır.

3.1.5 Grup / Çevrim-İçi Öğrenme

EKKO algoritması, her bir örnek girdisi için ağırlık düzeltmelerinin hesaplandığı ve her bir örnekten sonra ağırlıkların değiştirildiği bir yaklaşım sunar. Bu prosedür *bireysel* öğrenme veya *çevrim-içi* öğrenme olarak adlandırılır. Gradyanın tahmin değeri, gradyan yönü dolaylarında zigzaglı olacağından gürültülüdür.

Alternatif bir çözüm ise, her bir örnek girdisi için ağırlık güncelleştirmelerini hesaplamak ve eğitici kümenin tamamı işlemden geçirildikten sonra (*epoch*-döngü) bu değerleri (ağırlıkları değiştirmeksizin) saklamaktır. Döngünün sonunda, bütün ağırlık güncellemeleri toplanır ve ancak ondan sonra ağırlıklar bileşik değere göre değiştirilir. Bu yöntem ağırlıkları kümülâtif bir ağırlık değişimine göre adapte eder, bu yüzden gradyanı daha yakından izler. Bu yöntem *grup öğrenme modu* olarak adlandırılır ve *en hızlı iniş* prosedürünün de bir uygulamasıdır. Gerçekte grup olarak öğrenme, gradyanı EKKO'ya göre daha düzgün tahmin eder.

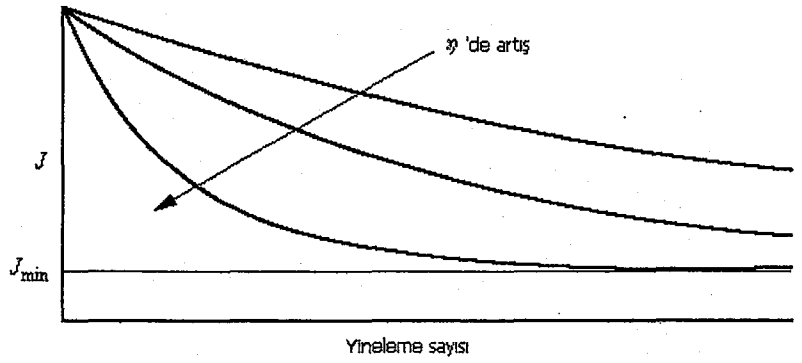
Aynı sayıda veri için iki yöntemin ağırlıkları güncelleme sayıları oldukça farklıdır. Çevrim-içi yöntemi (EKKO) her bir örnekte güncelleme yaparken, grup yöntemi her döngüde bir güncelleme yapar.

3.1.6 Gürbüzlük ve Sistemi Test Etme

EKKO çözümünün ilginç yönlerinden birisi gürbüzlüğüdür. Ağırlıkların başlangıç değerleri ne olursa olsun, çözüm her zaman aynı değere yakınsar. İstenilen çıktı değerlerine bir miktar gürültü eklense de parametreler aslında değişmez. Bu gürbüzlük, gürültünün her yerde olabildiği gerçek problemler için oldukça önemlidir. Sistemi eğitmede kullanılan örnek girdiler ile istenilen çıktılardan oluşan grup *eğitici küme* olarak adlandırılır. Bir kez en iyi parametreler bulunduktan sonra, sistem parametreleri sabitlenebilir. Sisteme daha önce hiç karşılaşılmamış yeni girdiler verildiğinde, her bir girdi için, daha önce eğitim aşamasında elde edilen parametrelere bağlı olarak bir yanıt (tepki) üretilecektir. Eğer yeni veri aynı deneyden geliyorsa, tepki bu belirli girdi değeri için istenilen değere benzer. Sistem genişletildiğinde genel olarak eğitici kümeden elde edilen performansın yeni verilere de uygulanması arzu edildiğinden, ekstraplasyon yapabilme yeteneği çok önemlidir. Fakat parametre değerlerini türetmede kullanılan metodolojiden dolayı sistemin yeni verilere ne kadar iyi tepki vereceğinden hiçbir zaman tam emin olamayız. Bu nedenle, gerçek yaşamdaki uygulamalara geçmeden önce sistem performansının bir test veri kümesi kullanılarak değerlendirilmesi iyi bir yaklaşımdır.

3.1.7 Öğrenme Eğrisi

Ağırlıklar en iyi değere yaklaştığında, $J(n)$'nin değerleri de minimum değeri olan J_{min}'e yaklaşarak azalacaktır. Adaptasyon sürecinin yakınsamasını gözlemenin en iyi yollarından birisi, her yinelemedeki hatayı grafikte göstermektir. Yinelemelerdeki *Hata Kareleri Ortalamasının* (HKO) grafiksel gösterimi, öğrenme eğrileri olarak adlandırılır (Şekil 3.4).



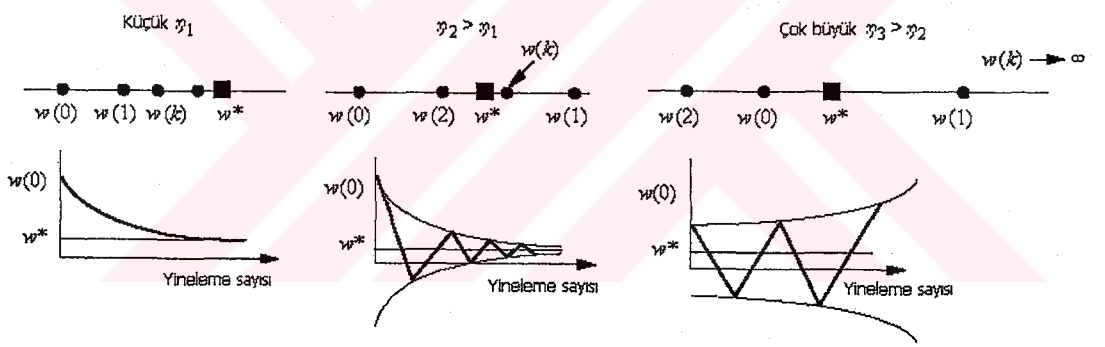
Şekil 3.4 : Öğrenme eğrileri (Principe ve diğ., 2000, s.32)

Öğrenme eğrileri; harici, skalar ve sistemin ne kadar iyi öğrendiğini kolaylıkla hesaplayabilen bir ölçüttür. Ancak, sistem öğrenemediği zaman bunun nedenini bize söylemez.

Hatadaki azalma oranı, adım değerinin büyüklüğüne (η) bağlıdır. Adım değerinin büyük olması daha az yineleme ile minimum değere yakın bir değer elde edilmesini, yani adaptasyonun yakınsamasını sağlar. Ancak, adımın değeri çok büyük olduğu zaman ıraksak bir yinelemeli sürece neden olur ve en iyi çözüm elde edilemez. Bu nedenle, yakınsamayı garanti edecek mümkün olan en büyük adım değerini bulmak için bir yol araştırılmalıdır.

3.1.8 Ağırlık İzleri

Bir adaptif sistem, ağırlıklarını en iyi çözümü bulma gayretiyle değiştirir. Zamana göre ağırlığın gösterildiği grafik ağırlık izi olarak adlandırılır (Şekil 3.5). Ağırlık izleri önemlidir ve doğrudan adaptasyon sürecinin bir ölçümüdür.



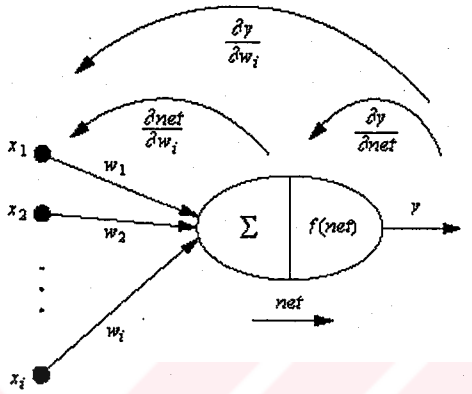
Şekil 3.5 : Ağırlık izlerinin yineleme sayısına bağlı olarak üç farklı adım değerine (η) göre grafiği (Principe ve diğ., 2000, s.33)

Gradyan inme adaptasyonunda, ağırlıklardaki değişiklikler iki nicelikle gerçekleştirilir: *adım büyüklüğü* (η) ve o noktadaki *gradyanın değeri*. İkinci dereceden performans yüzeyinin eğimi, tabanına yakın yerde azaldığından, adaptasyon w^*'a yaklaştıkça ağırlıklardaki değişiklikler sabit bir adım değeri için bile giderek küçülecektir. Bu nedenle ağırlıklar son değerlerine asimptotik olarak yaklaşırlar.

3.1.9 Delta Kuralı

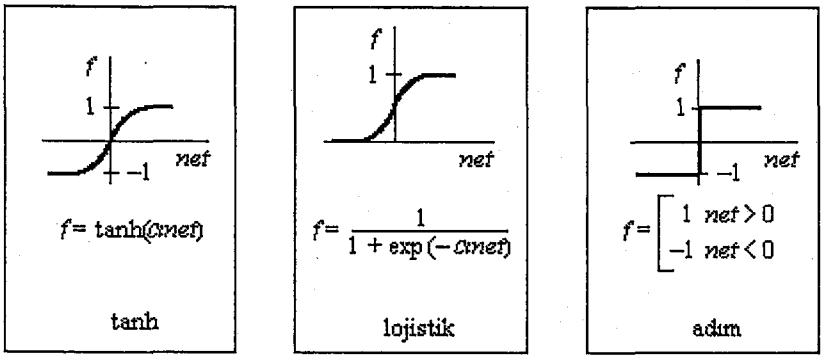
Bölüm 3.1.3'de, bir doğrusal sistemin (*Adaline*-adaptive linear element) performans yüzeyinin minimum değerine erişmek için ağırlıklarını uyarlayacak, gradyan inme

(EKKO) yöntemine dayanan ve adım-adım uygulanan bir sistematik prosedür anlatılmıştı. Bu bölümde ise, sinir ağlarında öğrenmenin temelini oluşturan farklı bir kavram kullanılarak EKKO algoritması yeniden ele alınacaktır. Bu kavram, matematikte eskiden beri kullanılan *zincir* kuralıdır. EKKO kuralının, maliyet fonksiyonunun (J) bilinmeyen parametrelere olan duyarlılığını hesaplamada zincir kuralına denk olduğu gösterilebilir.



Şekil 3.6 : Doğrusal olmayan bir İşlem Elemanının duyarlılığının hesaplanması

Şekil 3.6'da gösterilen doğrusal olmayan bir sistem ele alındığında (f (.) doğrusal olmayan bir fonksiyon, *net* ise tartılandırılmış girdilerin toplamıdır ($net = \sum_{k=1}^{i} w_k x_k$)).



Şekil 3.7 : Yaygın olarak kullanılan doğrusal olmayan Sigmoidal fonksiyonlar

Sigmoidal fonksiyonlar ($f(net)$) olarak genellikle lojistik ve hiperbolik tanjant fonksiyonu ($tanh$) kullanılır (Şekil 3.7).

Hiperbolik tanjant: $f(net) = \tanh(\alpha.net) = \dfrac{e^{\alpha \cdot net} - e^{-\alpha \cdot net}}{e^{\alpha \cdot net} + e^{-\alpha \cdot net}}$ (3.6)

Lojistik: $f(net) = \dfrac{1}{1 + e^{-\alpha \cdot net}}$ (3.7)

α, bir eğim parametresidir ve genellikle 1'e eşitlenir.

Daha önce anlatılan doğrusal sistemlerde olduğu gibi İşlem Elemanlarının (İE) ağırlıklarına göre kısmi türevini zincir kuralı ile hesaplayabiliriz:

$$\frac{\partial y}{\partial w_i} = \frac{\partial y}{\partial net}\frac{\partial net}{\partial w_i} = f'(net)x_i \tag{3.8}$$

$f'(net)$, $f(net)$'in kısmi türevidir. İE'nin (doğrusal olmayan işlem elemanı) türevi alınabiliyorsa, ağırlıktaki (δw_i) bir değişimin çıktı değerini (y) ne kadar değiştirdiğini hesaplayabiliriz. Doğrusal olmayan İE'de sadece bir ara nokta (*net*) vardır, fakat normalde kaç tane ara nokta olduğu önemli değildir.

Pratikte, çıktı değerinde bir hata (*gerçek çıktı değeri ile istenilen çıktı değeri arasındaki fark*) vardır ve bütün İE'lerin ağırlıkları ayarlanarak bu hatanın istatistiksel anlamda minimize edilmesine çalışılır. Yapılacak ayarlamada, çıktı değerinin her bir ağırlığa olan duyarlılığına göre dağıtılması gerekir. Çünkü çıktı hatası minimum yapılmak isteniliyorsa, çıktı değerini en çok etkileyen ağırlıkta daha büyük değişiklik yapmak gerekir. Bu yaklaşım ise gradyan inmenin (EKKO kuralının) gerçekleştirdiği şeydir.

Ağırlıkları ayarlamak için, çıktı hatasının topolojideki ara noktalara yayılması ve (3.4)'de tanımlanan yöntemle ölçeklenmesi gerekir. Ara noktalardaki (örneğin, *net*) hataların belirgin olarak bilinmesi gerekmediğinden bu yöntem çok güçlüdür. Zincir kuralı otomatikman hatanın dağıtımını gerçekleştirir. Bu sonuç özellikle daha karmaşık topolojilerin adaptasyonunda çok önemlidir ve sonuçta *geriye yayılım algoritması* elde edilecektir.

Eğer i indisin ağırlığını, p indisin örüntüsünü ve n de yineleme sayısını gösterirse, hata kareleri ortalamaları;

$$J = \frac{1}{2N}\sum_{p=1}^{N}\left(d_p - y_p\right)^2 \tag{3.9a}$$

$$y_p = f\left(\sum_i w_i x_{ip}\right) \tag{3.9b}$$

Zincir kuralı uygulandığında;

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y_p}\frac{\partial y_p}{\partial net_p}\frac{\partial net_p}{\partial w_i} = -(d_p - y_p) f'(net_p) x_{ip} \tag{3.10a}$$

Örüntü p için tahmin hata miktarı (d_p - y_p) yerine ε_p koyulursa;

$$\frac{\partial J}{\partial w_i} = -\varepsilon_p f'(net_p) x_{ip} \tag{3.10b}$$

elde edilir. Gradyan iniş uygulandığında;

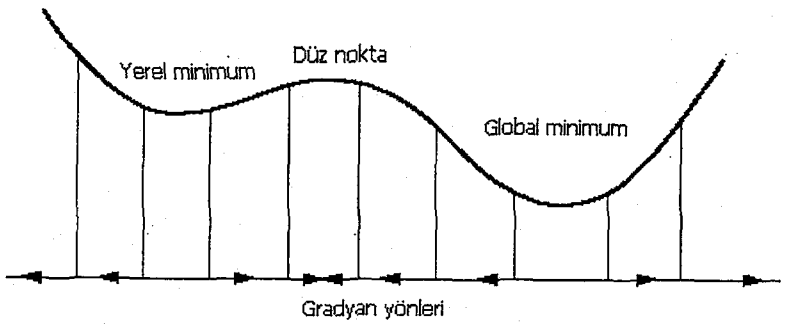
$$w_i(n+1) = w_i(n) + \eta \varepsilon_p(n) x_{ip}(n) f'(net_p(n)) \tag{3.11}$$

Bu kural *delta kuralı* olarak adlandırılır ve EKKO kuralının doğrusal olmayan türevi alınabilir sistemler için geliştirilmiş halidir. Doğrusal olmayan fonksiyonun türevi, ilgili girdi örüntüsü için işlem noktasında ($net_p(n)$) hesaplanır. Ayrıca, delta kuralı örüntü ve ağırlığa yerel olarak uygulanır; yani sadece belirli bir örüntü (İE'nin hatası) ve onun girdi değerine gereksinim duyar.

3.1.10 İşlem Elemanlarının Doğrusal Olmamasının Anlamları

Doğrusal olmama işin içine girdiğinde performans yüzeyi ile ilgili pek çok önemli kavram değişir. Performans, çıktı hataları üreten ağ topolojisine bağlıdır. Bu yüzden doğrusal olmayan işlem elemanları verilen bir problemi çözmede kullanıldığı zaman performans-ağırlıklar ilişkisi doğrusal olmaz ve tek bir minimum noktası artık garanti değildir. Performans yüzeyi birden fazla minimum noktası içerebilir ve dış bükey olmayan performans yüzeyi olarak adlandırılır (Şekil 3.8). En küçük hatayı veren nokta *genel minimum* (*global*) diğerleri ise *yerel minimum* (*local*) olarak adlandırılır. Performans yüzeyinin dış bükey olmaması, gradyan inme yöntemi performans yüzeyini araştırmada yerel bilgi kullandığından, arama planını değiştirir.

Gradyanın sıfıra eşit olduğu diğer durumlar *eyer noktası* olarak adlandırılır. Ağırlık güncellemesi, sonuçta gradyan tarafından gerçekleştirildiğinden gradyan çok küçük olacaktır, ağırlıklar fazla değişmeyecektir ve adaptasyon durabilecektir (*stall*). En iyi sonuca ulaşıldığı düşünülebileceğinden bu nedenle iyice (*suboptimal*) performans söz konusu olabilecektir.



Şekil 3.8 : Dışbükey olmayan performans yüzeyleri (Principe ve diğ., 2000, s.121)

Gürültü içeren gradyan, yerel minimum noktası veya eyer noktasından kurtulabilme şansını artırır. Dış bükey olmayan performans yüzeylerinde adaptasyon algoritmasının kontrolü çok daha hassas ve dikkat isteyen bir yapıdadır. Gradyan arama daha az sağlam olmasına rağmen, doğrusal olmayan işlem elemanı kullanmamızın nedeni, daha iyi performans elde edebilmeden ve hesaplama gücünden kaynaklanmaktadır.

3.2 Algılayıcı

Algılayıcı (perceptron) ilk olarak örüntü tanıma makinesi olarak 1958 yılında F.Rosenblatt tarafından keşfedilmiştir (Rosenblatt, 1958). Birden fazla girdinin bir çıktı katmanına doğrudan bağlandığı, her bir x_j girdi değerinin ayarlanabilir bir sabit w_{ij} (ağırlık) ile çarpılıp bir eğilim değeri (b_i) ile toplanarak, adım fonksiyonundan geçirilip çıktı değerinin (y_i) elde edildiği ağlardır.

$$y_i = f(net_i) = f\left(\sum_j w_{ij} x_j + b_i\right) \tag{3.12}$$

Rosenblatt, algılayıcının Denklem 3.3 ile sonlu adımda eğitilerek doğrusal olarak ayrıştırılabilir örüntüleri tanıyabildiğini göstermiştir (Rosenblatt, 1958). Bu başarı

sadece adaptif aygıtın faydalı örüntü sınıflandırıcı üretmediğini, aynı zamanda sonlu sayıdaki işlem adımında ağırlıkları değiştirerek yakınsayan sistematik algoritmaların olduğunu göstermiştir. Bu algoritmalar *öğrenme kuralları* olarak adlandırılır. Algılayıcının ikinci özelliği ise genelleştirme yeteneğidir. Öğrenme teorisi alanının algılayıcı ile başladığı söylenebilir.

3.2.1 Delta Kuralının Algılayıcıya Uygulanması

Tek bir İE'den algılayıcıya geçildiğinde Delta Kuralında fazla değişiklik olmaz. Sadece Denklem 3.7a'nın maliyeti eğitim kümesinin toplamı olarak değil, ilave olarak her bir çıktı İE'nin de toplamı olarak hesaplanır.

$$J = \frac{1}{2N}\sum_{p=1}^{N}\sum_{i=1}^{M}\varepsilon_{ip}^2 \tag{3.13}$$

Denklem 3.8a ve 3.8b'yi, i'nci İE'nin j'nci ağırlığı için yeniden düzenleyerek;

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_{ip}}\frac{\partial y_{ip}}{\partial net_{ip}}\frac{\partial net_{ip}}{\partial w_{ij}} = -(d_{ip} - y_{ip})f'(net_{ip})x_{jp} = -\varepsilon_{ip}f'(net_{ip})x_{jp} \tag{3.14}$$

elde edilir. Eğer i'nci İE için *yerel hata* (δ_{ip});

$$\delta_{ip} = \frac{\partial J}{\partial y_{ip}}f'(net_{ip}) \tag{3.15}$$

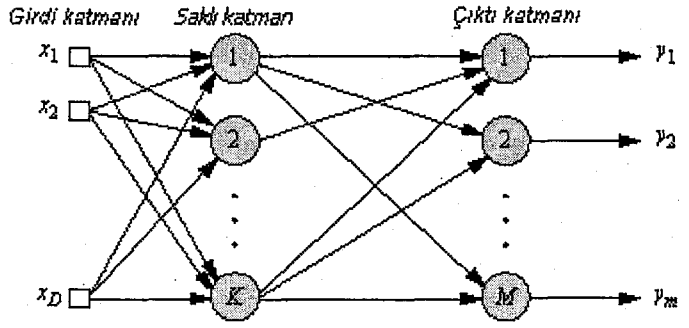
olarak tanımlanırsa, delta kuralı kullanılarak yapılan ağırlık güncellemesi;

$$w_i(n+1) = w_i(n) - \eta\frac{\partial J}{\partial w_{ij}} = w_i(n) + \eta\delta_{ip}x_{jp} \tag{3.16}$$

olacaktır. Bu algoritma ile ilgili olarak şaşırtıcı bir özellik, ağırlık güncellemesinin yerel olmasıdır. Belirli bir ağırlığı güncellemek için sadece yerel hata (δ_i) ve yerel aktivasyona (x_j) gereksinim duyulur.

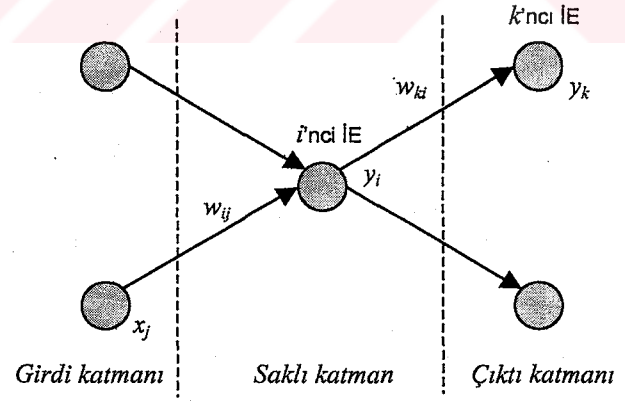
3.3 Çok-Katmanlı İleri Beslemeli Yapay Sinir Ağları

Çok-Katmanlı İleri Beslemeli Yapay Sinir Ağları (ÇKİB YSA), bir girdi kümesi ile bilinen çıktılar arasındaki ilişkiyi öğrenmeyi gerektiren problemleri çözmek için kullanılır. ÇKİB sinir ağları bu ilişkiyi öğrenebilmek için eğitme veri kümesine gereksinim duyduğundan, bir gözetimli öğrenme tekniğidir.



Şekil 3.9 : Çok-katmanlı ileri beslemeli yapay sinir ağı

Şekil 3.9'da mimarisi verilen ÇKİB sinir ağları, birbirlerine tartılandırılarak bağlanmış iki veya daha fazla sinir (İşlem Elemanı) katmanından oluşur. Bilgi akışı soldan sağa doğru, girdilerin (x) saklı katman sinirleri üzerinden çıktı katmanına ulaşması yoluyla sağlanır.



Şekil 3.10 : Saklı katmanın ayrıntısı

Şekil 3.10'da j'nci girdi elemanının saklı katmanın i'nci sinirine olan bağlantısının ağırlığı w_{ij}, saklı katmanın i'nci sinirinin çıktı katmanındaki k'ncı sinire olan bağlantısının ağırlığı ise w_{ki} ile gösterilmiştir.

Her bir sinir, verilen girdi vektöründen (x) aldığı uyarımın miktarına göre çıktı değerini hesaplar. Daha açıkça ifade etmek gerekirse, bir sinirin net girdi değeri, ona olan girdilerin tartılandırılmış toplamına eşittir ve çıktısı ise bu net girdi değerinin büyüklüğünü gösteren bir aktivasyon fonksiyonuna göre hesaplanır. Yani saklı katmanın j'nci siniri için (l = 1 girdi katmanı; n yineleme sayısı);

$$net_i^l(n) = \sum_{j=1}^{p} w_{ij} x_j(n) \quad \text{ve} \quad y_i(n) = f(net_i^l(n)), \tag{3.17}$$

ve k'nci çıktı siniri için ($l = L$ çıktı katmanı);

$$net_k^L(n) = \sum_{i=1}^{p} w_{ki} y_i(n) \quad \text{ve} \quad y_k(n) = f(net_k^L(n)). \tag{3.18}$$

Belirli bir girdi örüntüsü için, ağ bir çıktı (veya çıktılar kümesi) y_k üretir ve bu tepki her bir sinirin istenilen mevcut çıktıları d_k ile karşılaştırılır. Akabinde, sinir ağının ağırlıkları hatayı düzeltmek veya azaltmak için değiştirilir ve yeni örüntü oluşturulur. Ağırlıklar bu şekilde, toplam hata daha önceden belirlenmiş tolerans düzeyinin altında kalıncaya veya sinir ağının test kümesindeki performansının kötüleşmesi ile ölçülen aşırı eğitilmeye başlayıncaya kadar sürekli olarak değiştirilir. Ağırlıkları *Geriye Yayılım* (GY) ile güncelleştirme;

$$w_{ij}(n+1) = w_{ij}(n) + \eta f'(net_i(n)) \left(\sum_k \varepsilon_k(n) f'(net_k(n)) w_{ki}(n) \right) y_j(n) \tag{3.19}$$

eşitliği kullanılarak gerçekleştirilir (GY'nin türetimi için EK-A'ya bakınız). GY ile güncelleştirme denklemi (Denklem 3.19), yerel hata tanımı kullanılarak (Denklem 3.15) yeniden yorumlanırsa; denklemdeki toplam değeri, çıktı işlem elemanını i'nci işlem elemanına bağlayan ağırlık değeri ile ölçeklendirilmiş ağın her bir çıktı sinirindeki yerel hataların (δ_k) toplamıdır. Bu nedenle Denklem 3.19'daki parantez içerisindeki terim, çıktı katmanından i'nci İE'ye ulaşmadaki toplam hatayı etkin olarak hesaplar. Yerel hata;

$$\delta_i(n) = f'(net_i(n)) \sum_k \delta_k w_{ki}(n) \tag{3.20}$$

şeklinde yazılabilir. Gradyan inerek öğrenmede ağırlıklar, ilk olarak EKKO kuralında görülen, anlık gradyanın Widrow tahminine göre *yerel hata* ($\delta_i(n)$) ile *yerel aktivasyonun* ($y_j(n)$) çarpılması ile güncellenir.

$$\Delta w_{ij}(n) = \eta \delta_i(n) y_j(n) \tag{3.21}$$

Yerel hatanın hesaplanmasındaki farklılık, İE'nin doğrusal olup olmamasına ve bir çıktı işlem elemanı veya saklı-katman işlem elemanına ağırlığın iliştirilmiş olup olmamasına bağlıdır.

Durum 1: Eğer işlem elemanı *doğrusal* ve *çıktı katmanında* ise; $f(.)$ sabit 1 değerine sahip olduğundan,

$$\delta_i(n) = -\varepsilon_i(n) \tag{3.22}$$

Bu durum EKKO kuralında bulunan durumdur.

Durum 2: Eğer işlem elemanı *doğrusal değil* ve *çıktı katmanında* ise; siniri eğitmede kullanılacak delta kuralı, δ_i için aşağıdaki eşitlik kullanılacağından tam olarak Denklem 3.21'e eşittir.

$$\delta_i(n) = -\varepsilon_i(n) f'(net_i(n)) \tag{3.23}$$

ε_i, i'nci çıktıya ait hatadır. ÇKİB'daki çıktı ağırlıkları delta kuralına göre güncellenir.

Durum 3: Eğer işlem elemanı *doğrusal değil* ve *saklı katmanda* ise yerel hata, ağırlıklara göre ölçeklendirilmiş çıktı katmanındaki yerel hatalara olan bütün katkıların toplanması ile hesaplanır.

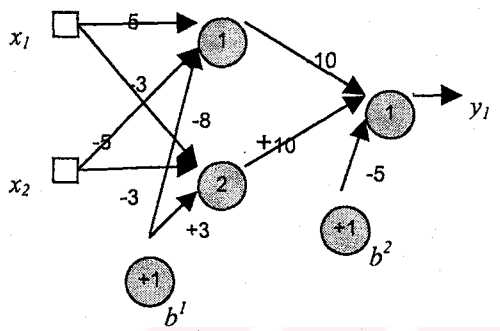
$$\delta_i(n) = f'(net_i(n)) \sum_k \delta_k w_{ki}(n) \tag{3.24}$$

Öğrenme hala gradyan azaltma ile gerçekleştiğinden, yapısal olarak ağırlık güncelleme denklemleri değişmez.

Örnek

Aşağıda görüldüğü gibi tek saklı katmanlı ÇKİB YSA verilmiş olsun. Her bir bağlantının ağırlıkları üzerlerinde gösterilmiştir. Saklı ve çıktı katmanın aktivasyon işlevleri

$f(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ *olarak verilmiştir.*



Bu YSA'nın $\mathbf{x} = (-1,-1)$ *girdisi ve d = -1 için çıktı değeri bulunup, η=1 için geriye yayılım yöntemi ile bir yineleme yapıldığı takdirde aşağıdaki sonuçlar elde edilir:*

$$net_1^1 = (-5)*(-1)+(-5)*(-1)-8 = 2; \quad f_1^1 = e^{2} - e^{-2} / e^{2} + e^{-2} = 0.964$$

$$net_2^1 = (-3)*(-1)+(-3)*(-1)+3 = 9; \quad f_2^1 = e^{9} - e^{-9} / e^{9} + e^{-9} = 1$$

$$net_1^2 = (-10)*(0.964)+(10)*(1)-3 = -2.64;$$

$$y = e^{2.64} - e^{-2.64} / e^{2.64} + e^{-2.64} = 0.9899$$

Denklem 3.23 ve 3.24 kullanılarak:

$$\delta_1^2 = -\varepsilon f'(net_1^2) = -(1-y^2)(d-y) = -(1-0.9899^2)1.7297 = -0.0348$$

$$\delta_1^1 = f'(net_1^1)\delta_1^2 w_1^2 = (1-0.964^2)(-0.348)(-10) = 0.246$$

$$\delta_2^1 = f'(net_2^1)\delta_1^2 w_2^2 = (1-1^2)(-0.348)(10) = 0$$

Yeni ağırlıklar Denklem 3.19 ve 3.21'i kullanılarak bulunur:

$$w_{1,1}^2(1) = w_{1,1}^2(0) + \eta\delta_1^2 f_1^1 = -10 + (1)(-0.0348)(0.964) = -10.3354$$

$$w_{1,2}^2(1) = w_{1,2}^2(0) + \eta\delta_1^2 f_2^1 = 10 + (1)(-0.0348)(1) = 9.9652$$

$$b^2(1) = b^2(0) + \eta\delta_1^2 = -10 + (1)(-0.0348) = -10.0348$$

$$w_{1,1}^1(1) = w_{1,1}^1(0) + \eta\delta_1^1 x_1 = -5 + (1)(-0.246)(-1) = -4.754$$

$$w_{1,2}^1(1) = w_{1,2}^1(0) + \eta\delta_1^1 x_2 = -5 + (1)(-0.246)(-1) = -4.754$$

$$w_{2,1}^1(1) = w_{2,1}^1(0) + \eta\delta_2^1 x_1 = -3 + (1)(0)(-1) = -3$$

$$w_{2,2}^1(1) = w_{2,2}^1(0) + \eta\delta_2^1 x_2 = -3 + (1)(0)(-1) = -3$$

$$b_1^1(1) = b_1^1(0) + \eta\delta_1^1 = -8 + (1)(-0.246) = -8.246$$

$$b_2^1(1) = b_2^1(0) + \eta\delta_2^1 = 3 + (1)(0) = 3$$

Tek saklı katmanlı ileri beslemeli (TSKİB) topolojiye sahip ve EKKO kullanılarak eğitilmiş sinir ağları; örnekleme, hedef gürültü değerleri, saklı İE sayısı, ağırlıkların büyüklükleri ve saklı birimin aktivasyon işlevine ilişkin pratikte uygulanabilir varsayımlara göre, doğrusal olmayan herhangi bir regresyon fonksiyonunun istatistiksel olarak tutarlı kestireçleridir (White, 1990). Ayrıca, bu tip YSA regresyon fonksiyonlarının türevlerinin istatistiksel anlamda tutarlı kestireçleri olarak da eğitilebilirler (White ve Gallant, 1992). Adım veya sigmoid aktivasyon işlevi kullanılan TSKİB ağları, ikili sınıflandırma için benzer varsayımlar altında tutarlı kestireçlerdir (Faragó ve Lugosi, 1993; Lugosi ve Zeger, 1995; Devroye ve diğ., 1996).

Ancak, yukarıdaki tutarlılık sonuçları pratikte uygulanamayacak bir varsayıma dayanmaktadır: *ağlar, genel minimuma keyfi olarak yaklaşmayı sağlayacak bir hata enküçükleme tekniği ile eğitilirler.* Böyle bir enküçükleme yaklaşımında, basit veya küçük problemler hariç, hesaplamalar kolaylıkla kontrol edilemez (Blum ve Rivest, 1989; Judd, 1990).

Bir yapay sinir ağının, bütün eğitici veri kümesini hatırlamaksızın öğrenemeyeceği pek çok önemli problemler de vardır. Bunlardan bazıları:

- Rastsal veya rastsalımsı sayıların tahmini
- Büyük tam sayıların çarpanlara ayrılması
- Büyük bir tamsayının asal olup olmadığının belirlenmesi
- İyi bir algoritma ile kodlanmış herhangi bir şeyin kodunun çözülmesi

Ayrıca unutulmamalıdır ki, YSA'yı eğitmek için eğitici veri kümesinde yer almayan bilgileri büyülü bir şekilde yaratan bir yöntem yoktur.

3.3.1 Saklı İşlem Elemanı Sayısının Etkileri

YSA uygulamalarında ana konulardan birisi, saklı işlem elemanı sayısının ne olmasının uygun olacağına karar verilmesidir. İki uç durum vardır: sinir ağı yapması gereken iş için gereğinden ya çok daha fazla veya çok daha az saklı işlem elemanı içerir. Her iki durumu da anlamak çok önemlidir. Çünkü İE sayısını doğru olarak belirlemek hala çok zor bir iştir. En uygun saklı işlem elemanı sayısı, aşağıdaki etmenlerin karmaşık olarak bileşimine bağlıdır:

- Girdi ve çıktı İE'lerinin sayısı
- Eğitici set sayısı
- Hedef değerlerin içerdiği gürültü miktarı
- Öğrenilecek fonksiyon veya sınıflandırmanın karmaşıklığı
- Ağ mimarisi
- Saklı birim aktivasyon fonksiyonunun tipi
- Eğitme algoritması
- Düzenleme

Genel olarak öğrenme algoritması, ilk olarak örneklerin çoğunluğunu doğru olarak sınıflandıran diskriminant fonksiyonunun doğru biçim ve konumunu bulur ve ardından daha az örnekli bölgelere sınıflandıracak şekilde yavaşça hareket eder. Eğer öğrenme mekanizması yeterli serbestlik derecesine sahip değilse, sistem problemi çözemeyeceğinden, hata yüksek bir değerde dengelenecektir. Öğrenme oranları büyük olduğunda bazen salınımlar oluşabilir. Bu salınımlar, iki iyice çözüm arasında ağırlıkların ani değişiminden kaynaklanır.

3.3.2 İki Saklı Katmanlı İleri Beslemeli Yapay Sinir Ağı ve Geriye Yayılım Prosedürü ile Eğitme

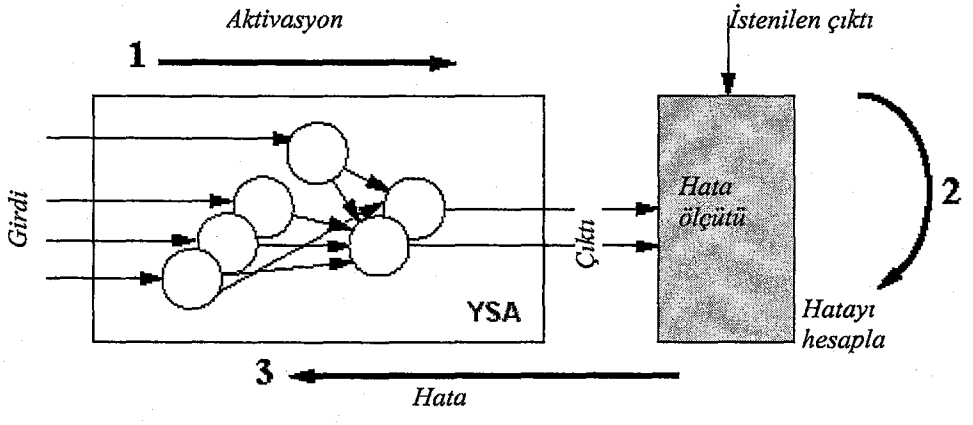
İki saklı katman içeren bir ÇKİB de tek saklı katman içeren ÇKİB gibi girdi-çıktı eşleştirmesini yapabilir. Ancak, "Problemi çözmek için kaç işlem elemanına ve katmana ihtiyaç vardır?" sorusuna, mevcut teoremler cevap verememektedirler. Bu soru, üzerinde hala denemeler yapılması gereken açık uçlu bir sorudur. Yapısal öğrenme teorisi genelleştirmeye teorik bir yanıt verir. Ancak, bu teorinin ÇKİB YSA'ya uygulanması zordur. Bununla birlikte, öğrenme mekanizmalarının büyüklüğüne dair iki temel yaklaşım Bölüm 3.4.5'de anlatılacaktır.

Eğitmede geriye yayılım algoritmasını kullanmanın güzelliği, ağ topolojisi ve girdinin boyutlarından bağımsız olarak uygulanabilen, sistematik ve adım adım işleyen bir prosedür olmasıdır. Çok daha karmaşık ağların daha fazla eğitme zamanına gereksinim duyma olasılığına karşın geriye yayılım, bu ağları da bütünüyle eğitme yeteneğine sahiptir.

İki-saklı katmanlı ÇKİB, normalde tek-saklı katmanlı ÇKİB'lerden daha yavaş eğitildiğinden, kural olarak denemelere genellikle daha basit topolojilerle başlanır.

3.3.2.1 Statik Ağların Geriye Yayılım Prosedürü ile Eğitilmesi

Geriye yayılım algoritması, herhangi bir değişiklik yapılmaksızın iki saklı katmanlı ÇKİB'ye veya herhangi bir ileri beslemeli topolojiye uygulanabilir. Çünkü Denklem 3.24 yerel olarak her saklı İE'ye topolojinin neresinde konumlanmışsa konumlansın uygulanır.



Şekil 3.11 : Geriye yayılım algoritmasında işlemler zinciri

Yerel hataların hesaplanması, ağın çıktısından başlayıp girdiye doğru yapılması ile gerçekleştirilir (Şekil 3.11). İlk olarak bir örnek veri, çıktı değerini elde etmek ve hatayı hesaplamak için ağ boyunca işlemden geçirilir. Ardından, çıktı katmanındaki hata hesaplanarak, çıktı işlem elamanlarının girdilerine bu hata değeri yansıtılır (Denklem 3.21'deki δ_k). Daha sonra, bir önceki katmandaki hatalar Denklem 3.24 kullanılarak hesaplanır ve bu işlem girdi katmanına ulaşılıncaya kadar sürdürülür. Hataların geriye yayılımında eski ağırlıklar kullanılır. Bütün yerel hatalar bulunduktan sonra Denklem 3.10 kullanılarak çıktı ağırlıkları, Denklem 3.19 kullanılarak saklı-katman ağırlıkları güncellenir.

ÇKİB sinir ağlarına başarı ile uygulanan ve öğrenilen ilişkileri yeni verilere genelleştiren pek çok eğitme yaklaşımları vardır. Bunu kesinleştirmek amacıyla veriler, bir eğitme kümesi ile performansı YSA sonuçlarının genelleştirilmesini göstermede kullanılan bir test kümesine bölünür. Diğer önemli noktalar ise; saklı sinirlerin sayısını da içeren pek çok eğitme parametresinin en iyi seçimi, hata fonksiyonunun yerel minimuma yakınsamasıdır. Son zamanlarda ÇKİB için en iyi ağırlıkları belirlemede, GY öğrenme kuralının yerine toplam hata kareleri ortalamasını minimum yapmak için genetik algoritmalar gibi sezgisel yaklaşımlar kullanılmaya başlanmıştır.

3.4 Çok Katmanlı Algılayıcıların Tasarımı ve Eğitilmesi

İyi tasarlanmış ancak kötü eğitilmiş bir sinir ağı yetersiz sonuçlar üretir. Bu nedenle, öğrenme sürecinin kontrolü ve hızlandırılması, sinir ağı uygulamalarının oldukça önemli bir kısmını oluşturur. Bu yüzden, maksimum performans için öğrenme oranı, daha verimli arama kriterleri ve öğrenmenin nasıl/ne zaman durdurulması gibi konuların üzerinde durulması gerekir.

3.4.1 Öğrenmenin Kontrol Edilmesi

Öğrenme (veya adaptasyon), sinir ağları teknolojisinde önemli bir adımdır ve girdi verilerinden gereksinim duyulan bilgilerin elde edilmesi prosedürüdür. Eğer öğrenme tam olmazsa, ağırlık değerleri en iyi değerlere yakın olmayacaktır. İşin kötüsü arama sezgisel olarak kontrol edilebilir. Sinir ağlarını eğitmede aşağıdaki konuların her birisini etkileyen etmenlerin iyice anlaşılması gerekir.

- Kullanıcı doğrudan aramayı etkiler

- Ağırlıkların başlangıç değerlerinin seçimi
- Öğrenme oranı
- Arama algoritmaları
- Durdurma kriterleri

Ayrıca, performansın iyi olması sistemi eğitmede kullanılan veri kümesinin miktarı ve kalitesine de bağlıdır.

3.4.1.1 Öğrenme Esnasında Adım Büyüklüğünün Kontrol Edilmesi

Doğrusal durumda öğrenme oranı, adaptasyonun hızı ile son ağırlık değerlerinin kesinliği arasında ödünleşme yapılarak belirlenebilir. İkinci dereceden performans yüzeylerinde ise, her bir yinelemede adım büyüklüğünü çizgisel arama yaparak, en iyi biçimde belirlemenin yolları vardır. Ancak, her bir yinelemedeki en iyi adım büyüklüğünün çizgisel arama ile belirlenmesinin zor olmasından dolayı normalde deneme-yanılma yaklaşımı kullanılır. ÇKİB'ler için çizgisel arama algoritması analitik olarak hesaplanamadığından nadiren kullanılır.

Doğrusal olmayan ağlarda (örneğin, ÇKİB) adım büyüklüğü çok daha önemlidir. Yerel minimum veya eyer noktalarının olması öğrenmenin durmasına neden olabilir.

Adım büyüklüğünün belirlenmesinde kullanılan yaygın tekniklerden birisi, eğitmenin başlangıcında, öğrenmenin arama aşamasında harcanan zamanı azaltmak için büyük bir öğrenme oranının kullanılması ve ardından ince ayar aşamasında öğrenme oranını azaltarak son ağırlık değerleri için yeteri kadar hassas sonuçların elde edilmesidir. Bu yaklaşım bazen *öğrenme oranını çizelgeleme* veya *soğutma* (*annealing*) olarak adlandırılır. Adım büyüklüğünün çizelgelenmesinde, adım büyüklüğünü aşağıdaki denklem ile kontrol eden yaklaşım kullanılabilir.

$$\eta(n) = \frac{\eta_0}{1 + \frac{n}{n_0}} \tag{3.25}$$

η_0 başlangıç adım büyüklüğü, n_0 yineleme sayısını göstermektedir. η_0 ve n_0 değerleri deneysel olarak bulunabilir. Eğer η_0'ın başlangıç değeri çok yüksek alınırsa öğrenme ıraksayacaktır. n_0 değerinin seçilmesi performans yüzeyine çok bağlı olduğundan deneyim gerektirir. Eğer n_0 çok küçükse, arama evresi çok kısa

olabilir ve öğrenme kilitlenebilir. Eğer n_0 çok büyükse, daha düşük öğrenme oranları ile çözümün ince ayarı yapılmadan önce arama evresinde çok zaman harcanır.

Dışbükey olmayan yüzeylerde soğutma çizelgesi, arama esnasında yerel minimum noktaları ile erken karşılaşılırsa, o noktalardan kurtulabilmek için avantaj sağlar. Büyük bir öğrenme oranı ile yerel minimum noktası aşılır ve öğrenme oranı küçüldüğünde genel minimum noktasına daha kesin olarak ulaşılabilir. Bu yaklaşımdaki en büyük problem önceden en iyi çizelge oranının bilinmemesidir, bu yüzden (3.25)'deki öğrenme katsayılarının seçimi probleme bağlıdır.

3.4.1.2 Ağ İşlem Elemanları için Öğrenme Oranlarının Belirlenmesi

Sinir ağları literatüründe (Haykin, 1994), kararlı ve hızlı yakınsama için bütün adaptif ağ parametrelerinin aynı oranda öğrenmesi önerilmektedir. Bu durum doğrusal ağlarda mümkündür, ancak ÇKİB'ler için bu o kadar da kolay değildir. Bu yüzden, öğrenme oranını uygun bir şekilde belirleyebilmek için hatanın ağ içerisindeki akışını iyi anlamak önemlidir. Çıktı katmanından girdi katmanına kadar öğrenme oranını [2,5] arasında bir faktörle artırmak akılcı bir yoldur.

3.4.1.3 Ağırlıkların Başlangıç Durumuna Getirilmesi

Hatanın bir katman boyunca azaltılması, öğrenme sürecinde bazı kısıtlara da yol açar. Bunlardan ikisi; ağırlıkların başlangıç durumuna getirilmesi ve saklı katman sayısının belirlenmesidir.

Girdiye yakın katmanlar çok yavaş eğitileceğinden eğitmenin verimliliği açısından çok katman içeren topolojiler önerilmemektedir (Principe ve diğ., 2000). Bir problemi çözmeye, çabucak eğitilebildiğinden, genellikle tek bir algılayıcı ile başlanılır. Eğer algılayıcı makul bir hata vermezse tek-saklı katmanlı ÇKİB denenir ve son olarak da iki-saklı katmanlı ÇKİB test edilir. İki-saklı katmanlı ÇKİB genel bir dönüştürücü olduğundan, ikiden fazla doğrusal olmayan saklı katmanlar nadiren kullanılır.

Eğitme çalışmasına başlamadan önce değeri belirlenmesi gereken bir diğer değişken ise ağırlıkların başlangıç değeridir. Başlangıç ağırlıkları sinir ağının öğrenme performansını etkiler. Çünkü son ağırlık değerlerinden çok farklı başlangıç değerleri eğitme süresini artıracaktır, ayrıca ağdaki bütün İE'lerin aynı hızda öğrenmeleri arzu edildiğinden öğrenme performansı etkilenecektir.

3.4.2 Diğer Arama Yaklaşımları

Gradyan inme yönteminin rağbet görmesinin nedeni, arama gücünden ziyade daha çok basitliğidir. Geriye yayılım yönteminden çok daha güçlü olan arama yöntemleri de vardır. Ağırlıkları adapte etmek için eğrilik bilgisini kullanan Newton yöntemi bunlardan birisidir ve *ikinci-dereceden* bir yöntemdir. Ancak, Newton yöntemi gerçeklenmesi, hesaplama süresi açısından çok maliyetli olan ve İE'de mevcut olmayan bilgiye gereksinim duyan bir yöntem olduğundan sinir ağlarında çok az kullanılır. Çok güçlü olmasına rağmen bu yöntem de bir yerel arama metodudur ve yerel minimuma takılabilir veya ıraksayabilir. *Tavlama Benzetimi* (Simulated Annealing) ve *Genetik Algoritmalar* gibi diğer teknikler genel arama prosedürleridir ve yerel minimumdan kurtulabilirler. Ancak, sinir ağları gibi dağıtılmış sistemlerde gerçeklenmeleri çok masraflıdır (Horst ve diğ., 1995).

Arama sadece yerel gradyan bilgisine göre gerçekleştirildiğinden, gradyan temelli yöntemler yerel minimumda takılı kalabilirler. Yerel eğrilik açısından yerel minimum ile genel minimum hemen hemen aynıdır, bu yüzden gradyan inme metodu performans yüzeyinin herhangi bir yerel minimumunda tuzağa düşebilir. Arama performans yüzeyindeki bir düzlükten geçerken, ağırlıklar gradyan ile orantılı olarak değiştirildiğinden, gradyan inme metodu çok yavaş hareket edecektir. Eğer gradyan küçükse, ağırlık değişimleri de küçük olacaktır (sabit adım değeri için) ve bu nedenle düz bir noktayı geçmek için pek çok yineleme gerekecektir. Bu durum, adaptasyonun sona erdiği performans yüzeyinde genel minimuma erişilmesi durumu ile kolaylıkla karıştırılabilir. Bu yüzden, adım büyüklüğü ve öğrenme oranının uygun bir şekilde kontrol edilmesine ve iyileştirilmiş arama yöntemlerine gereksinim vardır. Gradyan temelli yöntemlere göre daha hızlı olan ve pratikte kullanılan algoritmalar iki kategoriye ayrılabilir:

- Birinci kategoride yer alan algoritmalar, standart en hızlı iniş algoritmasının performans analizinden hareketle geliştirilmiş sezgisel tekniklerdir. Bu teknikler; *devinirlik ile öğrenme* (momentum), *değişken öğrenme oranı* (adaptif adım büyüklüğü) ve *esnek geriye yayılım* (resilient) olarak sıralanabilir.

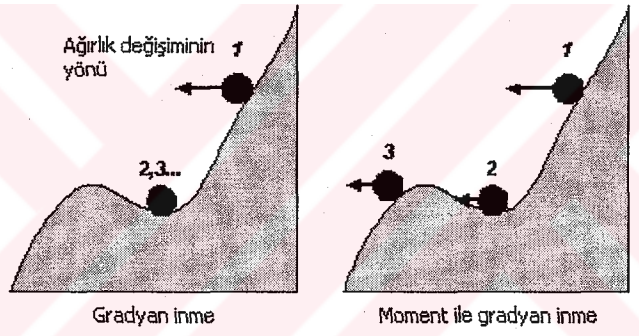
- İkinci kategoride yer alan algoritmalar, standart sayısal eniyileme tekniklerini kullanırlar (Hagan ve diğ., 1996). Bu tekniklerden başlıca üç tipi; eşlenik gradyan, yarı-Newton ve Levenberg-Marquardt olarak sayılabilir.

3.4.2.1 Devinirlik (Moment) ile Öğrenme

Devinirlik ile öğrenme (MOM), düz gradyan inme metoduna, *yakınsamayı hızlandırmak* ve *dengelemek* için bir hafıza teriminin eklenmesi ile elde edilen bir arama tekniğidir. Bu teknikte ağırlıkların güncellendiği eşitlik;

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_i(n)x_j(n) + \alpha\left(w_{ij}(n) - w_{ij}(n-1)\right) \qquad (3.26)$$

α moment sabitidir ve normalde α, 0.5 ile 0.9 arasında bir değer alır. Ağırlıklar en son yinelemede güncellendikleri miktarla orantılı olarak değiştirilir. Bu yüzden, arama işlemi tepeden aşağı inerek bir düzlüğe erişirse, gradyandan dolayı değil de ağırlıklardaki değişim oranından dolayı ağırlıklar hala değişir. Benzer şekilde, gradyanın tepeler arasında ileri geri sıçrama eğiliminde olduğu dar bir vadide MOM, ağırlıkların daha düz bir yol izlemesini sağlayacağından aramayı dengeli hale getirir.



Şekil 3.12 : Devinirlikle öğrenme yönteminin üstün yönü

Yukarıdaki şekil incelenirse: bir topun (ağırlık vektörünün pozisyonu) bir tepeden (performans yüzeyi) aşağıya yuvarlandığı düşünülsün. Eğer top tepenin küçük bir düz bölümüne erişirse, devinirliğinden dolayı bu yerel minimumu geçecektir. Ancak devinirliği olmayan bir top bu vadide takılıp kalacaktır. MOM yöntemi öğrenmeyi hızlandıran güçlü bir tekniktir.

3.4.2.2 Adaptif Adım Büyüklüğü

Eğitme evresinin bütününde her ağırlık için aynı adım büyüklüğünün kullanılmasındaki temel neden basitleştirmedir. Ancak, belirli bir yön için adım büyüklüğü öz değere (eigenvalue) göre belirlenmelidir. Fakat normalde öz değerler bilinmezler. Bununla birlikte, hatanın, ağırlık izlerinin veya her ikisinin birlikte

gözlemlenmesi ile eğitmenin daha iyi kontrol edilebileceği düşünülebilir. Öğrenme eğrisi düz olduğunda, öğrenmeyi hızlandırmak için adım büyüklüğü artırılabilir. Diğer yandan, öğrenme eğrisi yukarı aşağı salınım gösteriyorsa adım büyüklüğü azaltılmalıdır. En uç durum olarak hata sürekli artıyorsa, öğrenme kararsızdır ve bu durumda ağ öğrenme için yeniden başlatılmalıdır. Bu yaklaşım, eğitme evresi boyunca her bir ağırlık için adım büyüklüklerinin bağımsız olarak uyarlayan bir algoritma ile otomatikleştirilebilir.

Aslında düşünce basittir: öğrenme eğrisine bakmak yerine ağırlık izleri kullanılır. Birbirini izleyen ağırlık güncellemeleri aynı hata işaretini üretirse (yani, ardışık iki adımın her ikisinde de pozitif veya negatif hata değeri elde edilirse) öğrenme oranı çok yavaştır. Aksine, hatanın işareti yinelemeden yinelemeye değişiyorsa adım büyüklüğü çok büyüktür. Bu basit prensipler adaptif adım büyüklüğü olarak adlandırılan bir kurala konabilir. Yöntemin doğru olarak çalışabilmesi için her bir ağırlık için bağımsız bir adım büyüklüğünün kullanılması zorunludur. Aksi takdirde performansta çok önemli bir iyileşme olmayacaktır. Eğer w_{ij} ağırlığı için öğrenme oranı η_{ij} ile gösterilirse, her bir adım büyüklüğünü güncellemek için;

$$\Delta\eta_{ij}(n+1)=\begin{cases} n & \text{eğer } S_{ij}(n-1)\nabla_{ij}(n)>0 \\ -b\eta_{ij}(n) & \text{eğer } S_{ij}(n-1)\nabla_{ij}(n)<0 \\ 0 & \text{diğer durumlarda} \end{cases} \tag{3.27}$$

eşitliğinden faydalanılır. S_{ij} önceki gradyanların ortalamasını ve ∇_{ij} ise o anki gradyan değerini gösterir. S_{ij} ve ∇_{ij}'nin işareti aynı ise çarpımları sıfırdan büyük olacaktır. İlk durum yavaş yakınsamayı göstermektedir ve bu yüzden adım büyüklüğü her yinelemede sayısal olarak artırılır. İkinci durum, ağırlığın salınım gösterdiği durumdur ve bu nedenle algoritma adım büyüklüğünü o anki değeri ile orantılı olarak geometrik bir biçimde azaltır. Adım büyüklüğünü yavaş artırma ve hızlı azaltma ıraksamadan kaçınmak için tasarlanır. Eğer algoritma ıraksarsa, öğrendiği bütün bilgileri çabucak yitirir ve yeniden başlatılmalıdır. Bu algoritma *Delta-Bar-Delta* olarak adlandırılır. Adım büyüklüğünü ayarlayan Fahlman'ın *Quickprop* ve Almedia'nın *Adaptif Adım* gibi diğer alternatif algoritmalar da vardır.

3.4.2.3 Esnek Geriye Yayılım

ÇKİB YSA'da saklı katmanlarda genellikle sigmoid transfer fonksiyonu kullanılır. Bu fonksiyonlar, sonsuz girdi aralığını sonlu bir çıktı aralığına sıkıştırdığından sıklıkla

bastırıcı fonksiyonlar olarak adlandırılır. Bu fonksiyonların eğimi, girdi değerleri büyüdükçe sıfıra doğru yaklaşır. Bu ise, sigmoid fonksiyonlu ÇKİB ağları eğitmede en hızlı iniş kullanıldığında, gradyan çok küçük değere sahip olacağından, probleme yol açar. Bunun sonucunda, en iyi değerlerden çok uzakta olunsa da ağırlık ve yanlılıkta küçük değişimlere yol açar.

Esnek geriye yayılımın amacı, kısmi türevlerin büyüklüklerindeki bu zararlı etkileri bertaraf etmektir. Ağırlık güncellemenin yönünü belirlemede sadece türevin işareti kullanılır, türevin büyüklüğü ağırlık güncellemede herhangi bir etkiye sahip değildir. Ağırlık değişiminin miktarı ayrı bir güncelleme değeri ile belirlenir. Her bir ağırlık ve yanlılığın güncelleme değeri, performans işlevinin ağırlığa göre türevi birbirini takip eden iki yinelemede aynı işaretli ise *Δinc* faktörü kadar artırılır. Türevin ağırlığa göre değişimin işareti bir önceki yinelemeye göre değişmişse, güncelleme değeri *Δinc* faktörü kadar azaltılır. Eğer türev sıfır ise güncelleme değeri aynı kalır. Ağırlıklar dalgalanıyorsa, ağırlık değişimi azaltılacaktır. Eğer ağırlık pek çok yineleme boyunca aynı yönde değişim gösterirse, ağırlık değişiminin büyüklüğü artırılacaktır (Riedmiller ve Braun, 1993). Esnek geriye yayılım standart en hızlı iniş algoritmasından genellikle daha hızlıdır.

3.4.2.4 Eşlenik Gradyan Algoritmaları

Temel geriye yayılım algoritması ağırlıkları en hızlı iniş yönünde günceller. Bu yön, performans işlevinin en hızlı biçimde azaldığı yöndür. Performans işlevi gradyanın tersi yönünde çok hızlı biçimde azalsa da bunun en hızlı biçimde yakınsamayı sağlayacağı söylenemez. Eşlenik gradyan algoritmalarında, en hızlı inişten genellikle daha hızlı yakınsayan eşlenik yönlerde arama yapılır.

Eşlenik gradyan algoritmalarının pek çoğunda adım büyüklüğü her bir yinelemede güncellenir. Performans işlevini minimum yapan adım büyüklüğünü belirlemek için, eşlenik gradyan yönü boyunca bir arama yapılır. Öğrenme fonksiyonları için farklı arama fonksiyonları kullanılabilir.

Fletcher-Reeves Güncelleştirmesi

Eşlenik gradyan algoritmalarının tamamı ilk yinelemede en hızlı iniş yönünde araştırma ile başlar:

$$\mathbf{p}_0 = -\nabla J(0) \tag{3.28}$$

Ardından, mevcut arama yönü boyunca hareket edilecek en iyi mesafeyi belirlemek için bir çizgisel arama gerçekleştirilir:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n) p_{ij}(n) \tag{3.29}$$

Ardından, önceki arama yönlerinin eşleniği olacak şekilde yeni arama yönü belirlenir. Yeni arama yönünü belirlemede genel prosedür, yeni en hızlı iniş yönü ile bir önceki arama yönünü birleştirmektir:

$$\mathbf{p}_n = -\nabla_n + \beta_n \mathbf{p}_{n-1} \tag{3.30}$$

β_n sabitinin hesaplanış biçimine göre eşlenik gradyan yönteminin birçok farklı versiyonları vardır. Fletcher-Reeves güncellemesinde prosedür:

$$\beta_n = \frac{\nabla_n^T \nabla_n}{\nabla_{n-1}^T \nabla_{n-1}} \tag{3.31}$$

Bu oran mevcut gradyanın karesinin bir önceki gradyanın karesine bölünmesi ile elde edilir (Hagan ve diğ., 1996).

Eşlenik gradyan algoritmaları genellikle adaptif adım büyüklüğü algoritmalarına göre daha hızlıdırlar. Daha basit algoritmalara göre hafıza gereksinimi birazcık daha fazladır. Bu nedenle, çok fazla İE olan ve dolayısıyla ağırlık sayısı fazla olan ağlar için çoğunlukla iyi bir seçimdir.

Polak-Ribiére Güncelleştirmesi

Eşlenik gradyan algoritmasının bir başka çeşidi Polak ve Ribiére tarafından önerilmiştir. Fletcher-Reeves algoritmasında olduğu gibi, her bir yinelemedeki arama yönü (3.30)'daki gibi hesaplanır. Ancak, β_n sabiti aşağıdaki gibi hesaplanır:

$$\beta_n = \frac{\Delta \nabla_{n-1}^T \nabla_n}{\nabla_{n-1}^T \nabla_{n-1}} \tag{3.32}$$

Yani, β_n sabiti bir önceki yinelemedeki gradyandaki değişim ile mevcut gradyanın çarpımının bir önceki gradyanın karesine bölünmesi ile elde edilir (Hagan, 1996).

Powal-Beale Yeniden Başlatma Tekniği

Bütün eşlenik gradyan algoritmalarında, periyodik olarak çizgisel arama gradyanın tersine eşitlenir. Standart sıfırlama noktası, yineleme sayısının ağ parametre sayısına eşit olduğu andır, fakat eğitmenin verimliliğini artırabilecek başka sıfırlama yöntemleri de vardır. Bu yöntemlerden birisi, daha önce Beale (1972) tarafından önerilen ve Powel (1977) tarafından geliştirilmiş olanıdır. Bu teknikte, mevcut gradyan ile bir önceki gradyan arasında çok az bir diklik kalmışsa sıfırlama gerçekleşir. Bu koşul ise aşağıdaki eşitsizlik ile test edilir:

$$\left|\nabla_{n-1}^{T}\nabla_{n}\right| \geq 0.2\left\|\nabla_{n}\right\|^{2} \tag{3.33}$$

Eğer bu şart sağlanıyorsa çizgisel arama gradyanın tersine eşitlenir.

Ölçeklendirilmiş Eşlenik Gradyan Tekniği

Daha önce anlatılan eşlenik gradyan algoritmalarının tamamında, her yinelemede bir çizgisel arama işlemi gerçekleştirilir. Her aramada bütün eğitici veri girdileri için ağdan elde edilecek sonuçların pek çok kez hesaplanmasına gereksinim duyulduğundan, bu çizgisel arama işlemi sayısal olarak yoğundur. Ölçeklendirilmiş eşlenik gradyan tekniği Moller (1993) tarafından, zaman kaybettirici çizgisel arama yönteminden kurtulmak için tasarlanmıştır. Bu algoritma çok karmaşık olmakla birlikte temel düşünce, eşlenik gradyan yaklaşımı ile Levenberg-Marquardt algoritmasında da kullanılan modele güvenen bölge yaklaşımının birleştirilmesidir.

3.4.2.5 Çizgisel Arama Yöntemleri

Eşlenik gradyan ve yarı-Newton algoritmalarının çoğunda bir çizgisel arama işlemine gereksinim duyulur. Pek çok çizgisel arama yöntemi olmakla birlikte YSA'da bunlardan hangisinin daha iyi sonuç verdiğini tahmin etmek güçtür. Çizgisel arama yöntemlerinden bazıları aşağıda verilmiştir:

- Altın Kısım Arama
- Brent'in Arama Yaklaşımı
- Melez İkiye Bölme-Kübik Arama
- Charalambous'un Arama Yaklaşımı
- Geri İzleme

3.4.2.6 Yarı-Newton Algoritmaları

Newton'un önerdiği yöntem, eşlenik gradyan algoritmasına hızlı eniyilemede bir alternatif oluşturur. Newton'un yönteminde ağırlıklar;

$$w_{ij}(n+1) = w_{ij}(n) - A^{-1}(n)\nabla_{ij}(n) \tag{3.34}$$

şeklinde hesaplanır. $A(n)$ performans indeksinin mevcut ağırlık ve yanlılık değerleri için oluşturulan Hesiyen (ikinci türevler) matrisidir. Newton tekniği genellikle eşlenik gradyan yöntemine göre daha hızlı yakınsar. Ancak, ileri beslemeli yapay sinir ağlarında Hesiyen matrisini hesaplamak hem karmaşık hem de zaman alıcıdır. Newton metoduna dayanan fakat ikinci türevlerin hesaplanmasına gereksinim duymayan ve yarı-Newton yöntemleri olarak adlandırılan algoritmalar da vardır. Bu algoritmalarda her yinelemede Hesiyen matrisi yaklaşık değerlere göre güncellenir. Güncelleme, gradyanın bir fonksiyonu olarak gerçekleştirilir.

3.4.2.7 Levenberg-Marquardt Algoritması

Yarı-Newton yöntemlerinden olduğu gibi Levenberg-Marquardt algoritması da Hesiyen matrisini hesaplamaksızın ikinci-mertebeden eğitme hızını yakalamak üzere tasarlanmıştır. Performans işlevi karelerin toplamı (ileri beslemeli ağlarda olduğu gibi) biçiminde olduğunda, Hesiyen matrisi ve gradyan yaklaşık olarak aşağıdaki gibi hesaplanabilir:

$$\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J} \tag{3.35a}$$

$$\nabla = \mathbf{J}^{\mathrm{T}}\mathbf{e} \tag{3.35b}$$

Yukarıdaki eşitliklerde $\mathbf{J}$, ağ hatalarının ağırlık ve yanlılığa göre türevlerinin oluşturduğu Jacobian matrisi, $\mathbf{e}$ ise ağ hatalarının bir vektörüdür. Jacobian matrisi, Hesiyen matrisine göre çok daha az karmaşık olan standart geriye yayılım tekniği ile hesaplanabilir.

Levenberg-Marquardt algoritması, aşağıdaki Newton'a benzeyen güncellemede Hesiyen matrisi için bu yaklaşımı kullanır:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \left[\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\,\mathbf{I}\right]^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{e} \tag{3.36}$$

Eğer μ sıfıra eşitse, Hesiyen matrisi için yaklaşık değerler kullanan Newton yöntemine eşittir. μ büyük bir değere sahipse, (3.36) adım büyüklüğü küçük olan gradyan inmeye karşılık gelir. Newton yöntemi hatayı minimum yapan bir noktanın yakınında daha hızlı ve daha hassas olduğundan, mümkün olduğunca kısa sürede Newton yöntemine doğru kaymak hedeflenir. Bu nedenle, performans fonksiyonunda azalmanın olduğu her adımdan sonra μ değeri azaltılır ve sadece bir deneme adımında performans fonksiyonu arttığı zaman artırılır. Böylece, performans fonksiyonu algoritmanın her yinelemesinde daima azalacaktır. Levenberg-Marquardt algoritmasının yapay sinir ağlarını eğitmede kullanılması Hagan tarafından açıklanmıştır (Hagan ve Menhaj, 1994; 1996).Bu algoritma, orta büyüklükteki ileri beslemeli yapay sinir ağlarını eğitmede en hızlı olan tekniktir.

İyileştirilmiş arama yöntemlerinin zayıf ve üstün yönleri özet olarak Tablo 3.2'de verilmiştir.

Tablo 3.2 : Hızlı arama algoritmalarının üstün ve zayıf yönleri

Kategori	Yöntem	Alt-Yöntem	Üstün Yönleri	Zayıf Yönleri
Sezgisel Teknikler	Devinirlik ile Öğrenme		▪Yerel minimumlardan kurtulur	▪Pratik problemlere uygulanamayacak kadar yavaş
	Adaptif Adım Büyüklüğ		▪Salınıma engel olur	▪En yavaş teknik
	Esnek Geriye Yayılım		▪Az hafıza gereksinimi ▪Büyük problemler için uygun	▪Problem tipine bağlı performans
Sayısal Eniyileme Teknikleri	Eşlenik Gradyan Yöntemleri	Fletcher-Reeves Güncelleştirmesi	▪Az hafıza gereksinimi ▪Büyük problemler için uygun	
		Polak-Ribiére Güncelleştirmesi		▪Hafıza gereksinimi Fletcher-Reeves 'den fazladır
		Powal-Beale Yeniden Başlatma Tekniği	▪Diğer eşlenik gradyan yöntemlerine göre hızlıdır	▪Hafıza gereksinimine en fazla ihtiyaç duyar
		Ölçeklendirilmiş Eşlenik Gradyan Tekniği	▪Az hafıza gereksinimi ▪Büyük problemler için uygun	
	Çizgisel Arama Yöntemleri (Altın Kısım Arama, Brent'in Arama Yaklaşımı, Melez İkiye Bölme-Kübik Arama, vs.)		Çizgisel aramaya Yarı-Newton ve Eşlenik Gradyan Yöntemleri ihtiyaç duyar. Yakınsamayı artırıcı faydaları vardır.	
	Yarı-Newton Yöntemleri		▪Orta büyüklükteki problemler için uygun	▪Hafıza gereksinimi fazladır
	Levenberg-Marquardt Algoritması		▪Çok hızlı yakınsar	▪Bir kaç yüz ağırlık içeren problemler için uygun

3.4.3 Durdurma Kriterleri

Öğrenme işi gerçekleştikten sonra eğitmeye son vermek gerekir. Ancak, sinir ağının eğitimini nasıl ve ne zaman durdurmak gerektiğine ilişkin doğrudan bir gösterge yoktur.

3.4.3.1 Eğitici Kümedeki Hataya Dayanan Durdurma

Basit yaklaşımlardan birisi, eğitme evresini sonlandırmak için önceden bir yineleme sayısının belirlenmesidir. Ancak bu yaklaşım, öğrenme öncesinde veya öğrenme esnasında sistemden herhangi bir geri besleme almaz. Dolayısıyla, önceden belirlenmiş olan maksimum yineleme sayısına gelindiğinde, öğrenme makinesinin bulduğu katsayıların en iyi değerlere yakınlığı konusunda herhangi bir garanti yoktur.

Eğitmeyi sona erdirmek için çıktı HKO'nun analizi yapılabilir. Problem için kabul edilebilir bir hata düzeyi seçilir ve HKO'ya bir eşik değer uygulanır. Ancak, HKO için bir eşik değerinin seçilmesi yanıltıcı olabilir, çünkü HKO sınıflandırmada sadece endirekt değişkendir. Ayrıca, öğrenme sisteminin belirlenen eşik değere ulaşabileceği de garanti değildir, bu durumda eğitme asla sona ermez.

Bir diğer alternatif yaklaşım, bir yinelemeden diğerine HKO'daki değişimin verilen bir değerden düşük oluncaya kadar eğitmeye devam etmektir. Bu yaklaşımın dezavantajı ise, eğitme olgunlaşmadan performans yüzeyinin düz bölgelerinde sona erdirilmesidir.

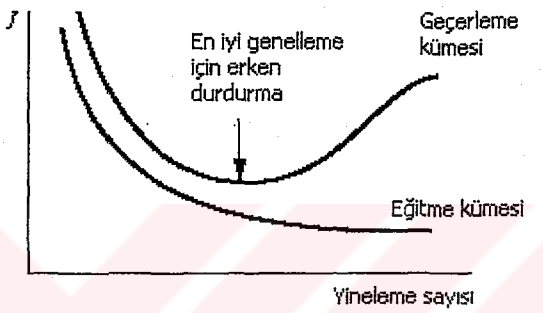
3.4.3.2 Genellemeye Dayanan Durdurma Kriterleri

Veri modellemeye dair geleneksel bilgiler ve öğrenme teorisindeki yeni gelişmeler, kritik bir noktadan sonra geriye yayılım ile eğitilen bir ÇKİB'nin eğitme kümesinde daha iyi sonuç vermeye devam etmesine karşın, test kümesindeki performansın bozulmaya başladığını açıkça göstermektedir (Vapnik, 1995). Bu fenomen **aşırı eğitme** (*overfitting*) olarak adlandırılmaktadır.

Bu problemi çözmenin bir yolu, maksimum genellemenin olduğu noktada eğitmeyi durdurmaktır. Bu yöntem erken durdurma veya çapraz geçerleme ile durdurma olarak adlandırılır. Yeteri kadar büyük bir sinir ağında, yineleme sayısı arttıkça eğitme hatasının her zaman azaldığı deneysel olarak gösterilmiştir. Eğer ağın eğitilmediği veri kümesi (*geçerleme kümesi*) için hatanın grafiği çizilirse, hatanın

başlangıçta azaldığı ancak daha sonra tekrar artmaya başladığı (Şekil 3.13) görülebilir. Bu nedenle eğitme, geçerleme kümesinde en az hatanın oluştuğu noktada kesilmelidir.

Bu yöntemi uygulamak için eğitme kümesi; eğitme ve çapraz geçerleme kümeleri olarak ikiye bölünmelidir. Çapraz geçerleme kümesi normalde toplam eğitme kümesinin %10'u olarak belirlenmelidir. Mevcut ağırlıklarla öğrenme makinesinin performansı çapraz geçerleme kümesi kullanılarak test edilmelidir. Çapraz geçerleme setindeki hata artmaya başladığı anda eğitmeye son verilmelidir. Durulan nokta maksimum genellemenin yapıldığı noktadır.



Şekil 3.13 : Çapraz geçerleme veya erken durdurma

Bu yöntemdeki problem, çapraz geçerlemenin eğitme kümesinin büyüklüğünü azaltmasıdır. Sinir ağlarında başlangıç verilerinin az olması probleme yol açtığından, çapraz geçerleme bu durumu daha da kötüleştirecektir. Ancak, bu metodun faydaları (durma noktasının kesinliği) maliyetlerine (az veri) ağır basmaktadır.

3.4.4 Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları Ne Kadar İyi Öğrenirler?

3.4.4.1 Eğitme Kümesinin Büyüklüğü

Eğitme kümesinin büyüklüğü parametrik olmadan (sinir ağları gibi) eğitilmiş herhangi bir sınıflandırıcının performansını doğrudan etkiler. Test örneklerini δ hata ile sınıflandırmak için gereksinimi duyulan eğitme örüntüsünün sayısı (N) aşağıdaki gibi verilir (Haykin, 1995):

$$N > \frac{W}{\delta} \tag{3.37}$$

W ağdaki ağırlık sayısını gösterir. Bu denklem, ihtiyaç duyulan eğitme örüntüsü sayısının ÇKİB'nin serbest parametrelerinin sayısı ile doğrusal orantılı olarak arttığını gösterir. Sezgisel bir yaklaşım olarak $N \approx 10W$, yani eğitme kümesinin büyüklüğü ağ ağırlıkları (karar değişkenleri) sayısının 10 katı olması durumunda %90 hassasiyetle test verilerinin doğru sınıflandırılacağı öngörülmüştür. Eğitme kümesi verileri oluşturulurken ana odak noktası, modellenmek istenen problemin bilinen bütün çalışma koşullarını kapsayacak şekilde verileri toplamak olmalıdır. Eğer eğitici küme örüntü uzayının bazı kısımlarından veri içermezse, bu alanlara ilişkin sınıflandırma ekstrapolasyona dayanacaktır. Bu ise doğru sınıflandırma sınırını içerebilir veya içermeyebilir.

3.4.4.2 Ölçeklendirilebilme

Çok önemli noktalardan birisi de problemin büyüklüğü arttığında sinir ağının özelliklerinin ne kadar iyi ölçeklendirilebileceğidir. Literatürde küçük problemler için çok iyi performans gösterip büyük problemlerde aynı performansı yakalayamayan pek çok örnek sistem yer almaktadır. Barron tarafından pek çok farklı büyüklükteki problemde HKO'nın analizi neticesinde geliştirilen bir ispata göre, büyük eğitme kümeleri için ÇKİB'nin (bir-saklı katmanlı) hatası girdi uzayının büyüklüğünden bağımsızdır ve saklı işlem elemanı sayısının tersi ile ölçülebilir ($O(1/N)$) (Barron, 1993). Bu sonuç, D boyutlu uzayda geometrik olarak azalan klasik kestireçlerinkinden ($O(1/\sqrt[D]{N^2})$) daha iyidir.

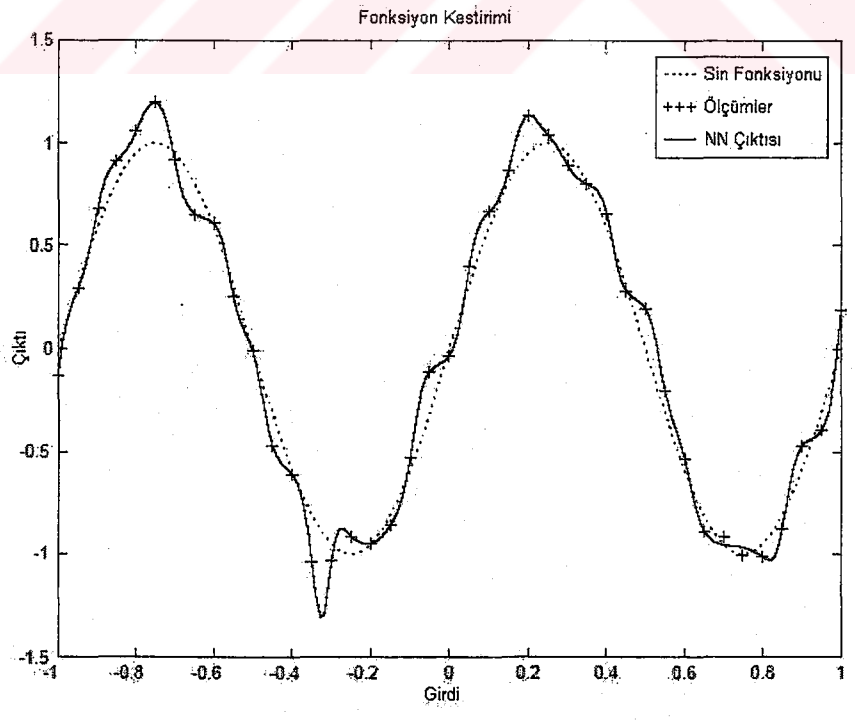
3.4.4.3 Eğitilebilme

Geriye yayılım kullanılan ÇKİB'de eğitme zamanının problem büyüdükçe üssel olarak arttığı deneysel olarak belirlenmiştir. Gereksinim duyulan örüntü sayısı ağırlık sayısına bağlı olarak sadece doğrusal olarak artmasına rağmen, daha büyük ağların eğitme zamanının büyüklükleri ile orantılı olarak üssel bir biçimde arttığı görülmektedir. Bu durum, geriye yayılım ile eğitilmiş ÇKİB'lerin pratikte çözemeyeceği problemlerin olduğunu gösterir. Ancak, bu davranışa karşılık olarak modüler ağ mimarisi veya seyrek bağlantılar ve ileri eğitme kuralları kullanılabilir. Pratikte küçük ağlarla modellemeye başlanır ve eğer sonuç memnun edici değilse daha büyük ağlara geçilir.

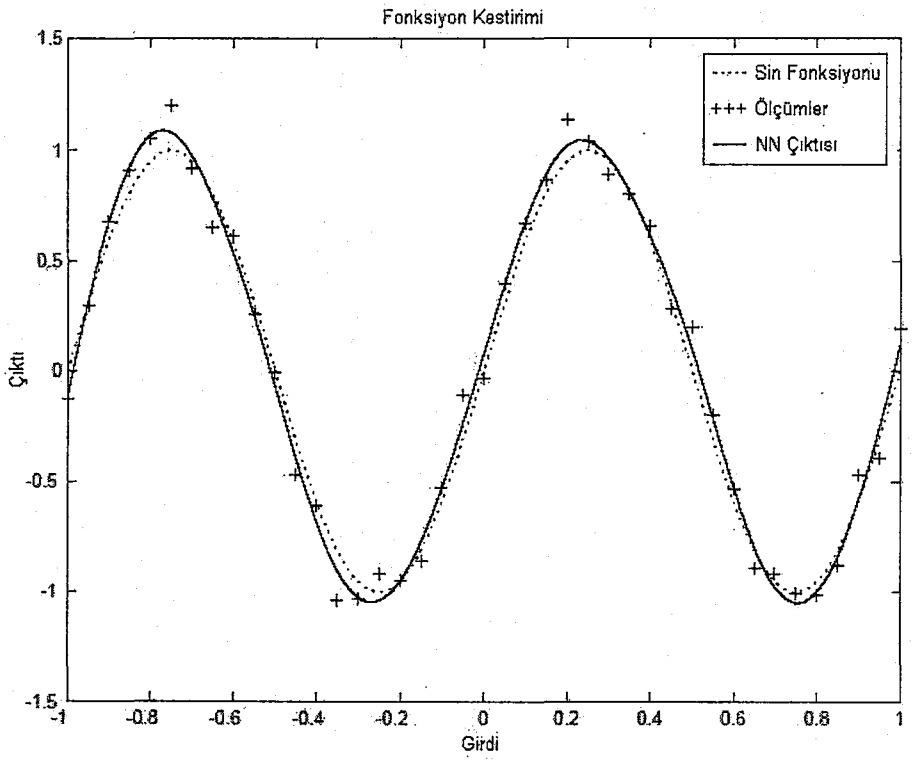
3.4.5 Genelleştirme ve Ağ Büyüklüğü

Herhangi bir pratik uygulamada temel soru, sinir ağının test veri kümesindeki performansının nasıl olacağıdır ve bu ise genelleştirme problemidir. Geriye yayılım ile eğitilmiş ÇKİB'ler genelleştirme yeteneklerini kontrol edemezler. Eğitmeyi durdurmak için kullanılan çapraz geçerleme, verilen bir ağ büyüklüğü için genelleştirmeyi maksimum yapmaya izin verir. Ancak, bu yöntem genelleştirme için en iyi ağ topolojisini kurmayı sağlayacak bir mekanizma sunmaz. Modelin büyüklüğü (bazen modelin karmaşıklığı olarak adlandırılır) performansla ilgilidir: serbestlik derecesinin (ağırlıklar) çok az olması, ağın hedef fonksiyona iyi uydurulmasını etkiler. Ancak, eğer ağ çok büyükse iyi genelleştirme yapamayacaktır, zira uygunluk eğitme veri kümesine özgü olacaktır (*ezberleme*). Bu yüzden, ÇKİB tasarım metodolojisinde iyi bir performans için ağın karmaşıklığını kontrol eden yöntemler zorunludur.

Şekil 3.14 (a)'da '+' ile gösterilen ölçüm değerleri için beklenen sinüs fonksiyonu kesikli çizgi ile 1-20-1 yapısındaki ÇKİB YSA'dan elde edilen sonuçlar ise düz çizgi ile gösterilmiştir. Görüldüğü gibi ölçüm değerlerini bire bir gerçekleyen bir fonksiyon elde edilmiş ancak gerçeklemedeki genellik sağlanamamıştır. Oysa Şekil 3.14 (b)'de uygun büyüklükteki YSA'nın kullanılmasıyla, sürekli çizgi ile gerçeklenen bir fonksiyonla ifade edilmiş ve genelliğe ulaşılmıştır.



(a)



(b)

Şekil 3.14 : Ezberleme ile genelleme arasındaki fark (Demuth ve Beale, s. 5-51-55)

İyi genelleme için gerekli ama yeterli olmayan üç koşul vardır. Bunlar:

1. Ağın girdisi olan veriler, hedef değerlere ilişkin yeterli düzeyde bilgiyi içermelidirler. Öyle ki; doğru çıktılar ile girdileri istenilen hassasiyette ilişkilendiren bir matematiksel fonksiyon var olmalıdır. Bir ağ için iyi verilerin bulunması ve yeterli düzeyde eğitme verisinin toplanması, genellikle o ağı eğitmek için harcanan zaman ve çabadan çok daha fazlasını gerektirir.

2. Ağın öğrenmeye çalıştığı fonksiyon bir anlamda düzgün olmalıdır. Başka bir deyişle, girdilerdeki küçük değişiklikler çoğu zaman çıktılarda da küçük değişikliklere yol açmalıdır. Sürekli girdi ve hedef değerler için fonksiyonun düzgünlüğü, sürekliliği ve girdi uzayının büyük bir kısmında ilk türeve dair sınırlamaları vurgular. Girdi uzayının doğrusal olmayan bir dönüşümü sıklıkla fonksiyonun düzgünlüğünü ve genelleştirmeyi artırabilir.

3. Eğitme vakaları yeteri kadar büyük ve genellemeye çalışılan bütün vakaların (*popülasyon*) oluşturduğu kümenin bir temsili altkümesi (*örnek*) olmalıdır. Bu koşulun önemi iki farklı tipte genelleştirmenin olmasıyla yakından ilişkilidir: interpolasyon ve ekstrapolasyon. Özellikle eğitici veri aralığının dışında kalan vakalar ekstrapolasyona gereksinim duyar. Interpolasyon çoğunlukla güvenli bir

şekilde yapılabilir, fakat ekstrapolasyon güvenilmezdir. Bu yüzden ekstrapolasyona gereksinim duyulmayacak kadar yeterli düzeyde eğitme verisinin bulunması önemlidir.

Gerçek verilerdeki gürültü eğitici küme ne kadar büyük olursa olsun genelleştirmenin duyarlılığını sınırlandıracağından istenmeyen bir durumdur. Bununla birlikte, eğitme periyodunda girdilere yapay gürültünün eklenmesi, eğitme kümesinin küçük olduğu düzgün fonksiyonlar için genelleştirmeyi artırıcı birkaç yöntemden birisidir. Genellikle gürültü dağılımının ortalamasının sıfır ve varyansının sonlu olduğu varsayılır. Hedef değerlerdeki gürültü ezberleme riskini artırır (Moody, 1992).

Ağ büyüklüğü problemi basitleştirilmiş olarak şu şekilde ifade edilebilir: "*Herhangi bir öğrenme mekanizması problemi çözmek için yeteri kadar büyük olmalıdır, fakat daha büyük değil*" (Hagan ve diğ., 1996). Öğrenme mekanizmalarının büyüklüğüne dair iki temel yaklaşım vardır (Hertz ve diğ., 1991):

1. *Büyüme Metodu :* Küçük bir ağ ile başlanır ve giderek büyütülür.
2. *Budama Metodu :* Büyük bir ağ ile başlanır ve önemsiz bileşenler atılarak ağ küçültülür.

Budamada, ağın toplam performansını önemli ölçüde etkilemeksizin hangi parametrelerin atılacağını belirlemeye dönük kriterler bulunmaya çalışılır. Bu kriterlerden basit olan iki tanesi: değerlerine bakarak ağırlıkların elenmesi ve eşleştirmede ağırlıkların önemlerinin hesaplanmasıdır.

3.4.5.1 Büyüyen Ağlar

Yapıcı öğrenme, eğitim sürecinde sinir ağına işlem elemanları ve bağlantılar ekler. Yapıcı öğrenme hiç saklı katmanı olmayan bir ağ ile başlar ve bir süre eğitir. Daha sonra mevcut ağırlıkları değiştirmeden bir veya daha fazla yeni saklı elemanı ağa ekler ve eğitim tekrar baştan yapılır. Bu süreç tekrarlanarak devam ettirilir. Farklı bağlantı örüntüleri ile ağırlıkları dondurma veya eritme için farklı şemaların kullanılması gibi değişik varyasyonlar mümkündür. En çok bilinen yapıcı algoritma Kademeli Dizi Korelasyonu'dur (Fahlman ve Lebiere, 1990).

Yapıcı algoritmalar, amaç fonksiyonunun yerel minimumlarından kaçınmada oldukça etkilidirler ve genellikle tavlama benzetimi ve genetik algoritmalara göre daha hızlıdırlar. Standart geriye yayılım ile öğrenilmesi çok zor olan fakat göreceli olarak Kademeli Dizi Korelasyonu ile daha kolay öğrenilen iki-spiral problemi buna çok iyi

bir örnektir. Yapıcı öğrenme, özellikle aktivasyon işlevleri adım fonksiyonu olan ÇKİB ağları eğitmede kullanılır. Bu tip ağlarda amaç fonksiyonu süreksiz olduğundan gradyan azaltma yöntemi kullanılamaz. Bu yüzden bu tip sinir ağlarını yapıcı öğrenmenin dışında farklı bir yöntemle eğitmek çok zordur. Yapıcı öğrenmenin çoğu, ağ yeteri kadar büyüdüğünde eğitme hatası sıfıra gidecek şekilde tasarlanır. Bu da genelleştirme hatasının uygulanmamasını garanti altına alır.

3.4.5.2 Ağırlık Eleme

Ağırlık elemedeki düşünce, adaptasyon esnasında bütün ağırlıkları azaltarak sıfır yapmaya çalışacak bir itici güç oluşturmaktır. Eğer girdi-çıktı eşleştirmesi bazı büyük ağırlıklara gereksinim duyarsa, öğrenme ile önemli ağırlıklar artmaya devam ederken önemsiz olanlar sıfıra doğru itilecektir. Bu yaklaşım *ağırlık azaltma* olarak adlandırılır ve ağırlık adaptasyon eşitliğine basit bir ekstra terimin eklenmesi ile kolaylıkla gerçeklenebilir:

$$w_{ij}(n+1) = w_{ij}(n)(1-\lambda) + \eta\delta_i x_j \tag{3.38}$$

Bu denklemde, δ yerel hatayı, x yerel aktivasyonu, η öğrenme oranını ve λ ise ağırlık azaltma sabitini göstermektedir. Ağın toplam serbestlik derecesi azaltılarak belirli bir değerden küçük olan ağırlıklar elenebilir.

3.4.5.3 Optimal Beyin Hasarı

Ağırlık eleme yöntemi, girdi-çıktı eşleştirmesinde ağırlıkların önem derecesinin kabaca ölçümü olarak yalnızca ağırlığın büyüklüğüne dayanır. Daha iyi bir ölçüt, bir ağırlığı sıfır yapmanın maliyetini bulma olan ağırlığın *çarpıcılığıdır* (LeCun ve diğ., 1997). Aşağıda verilen elemanlardan oluşan Hesyan'ın (H) istenilen bilgileri içerdiği gösterilebilir.

$$H_{ij} = \frac{\delta J}{\partial w_i \partial w_j} \tag{3.39}$$

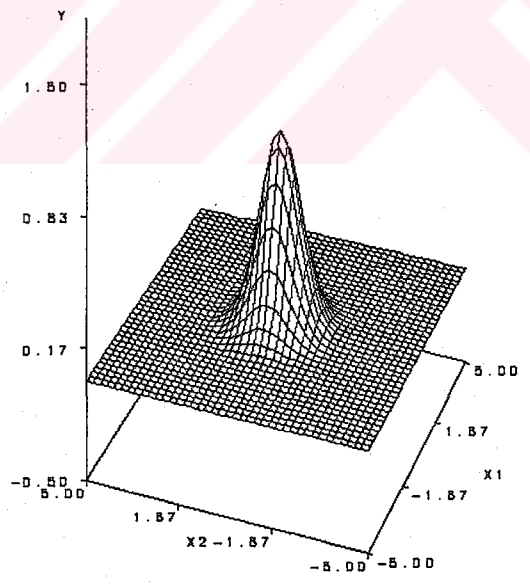
Ancak bu hesaplamadaki problem yerel olmamasıdır. Hesiyen'in yerel bir tahmininin (bazen kötü olabilir), sadece köşegendeki terimlerin dikkate alınarak hesaplanması ile aşağıda verilen her bir ağırlık (w_i) için çarpıcılığın (s_i) bulunması mümkün olur:

$$s_i = H_{ij} w_i^2 / 2 \tag{3.40}$$

Bu yöntemi uygulamak için, büyük bir sinir ağı normal yoldan eğitilir ve her bir ağırlığın çarpıcılığı hesaplanır. Ardından ağırlıklar çarpıcılıklarına göre sıralanır ve yüzde olarak düşük çarpıcılıklar atılır. Başlangıç koşulu olarak ağ ağırlıkların önceki değerleri kullanılarak yeniden eğitilir ve süreç tekrarlanır.

3.4.5.4 Ağ Büyüklüğünde Saklı Katman Sayısının Etkileri

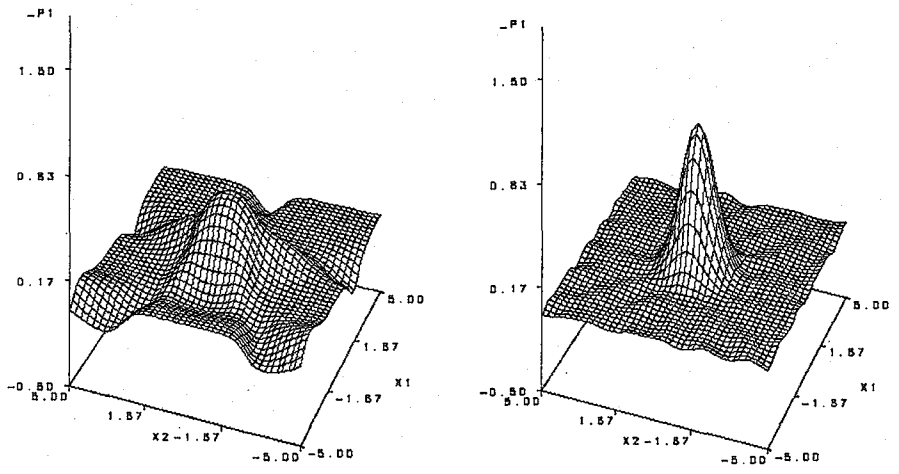
Herhangi bir sürekli doğrusal olmayan saklı katman aktivasyon fonksiyonuna sahip ÇKİB YSA'nın "genel kestireç" özelliğini taşıyabilmesi için fazla miktarda saklı birim içeren tek bir saklı katman yeterlidir (Hornik ve diğ., 1989; Hornik, 1993; Bishop, 1995, p130; Ripley, 1996, pp 173-180). Ancak, verilen bir fonksiyonu kestirebilmek için kaç adet saklı elemanın gerektiğini söyleyebilecek bir teori henüz bulunamamıştır. Adım/Eşik aktivasyon fonksiyonu içeren YSA'da, tam genelleştirme için ise iki saklı katmana gereksinim duyulur (Sontag, 1992).Eğer sadece bir girdi varsa, tek saklı katmandan daha fazlasını kullanmak avantajlı değildir. Fakat iki veya daha fazla girdi olursa işler daha çok karmaşıklaşacaktır.



Şekil 3.15 : Hedef fonksiyonu için tepe verisi (Sarle, 1997)

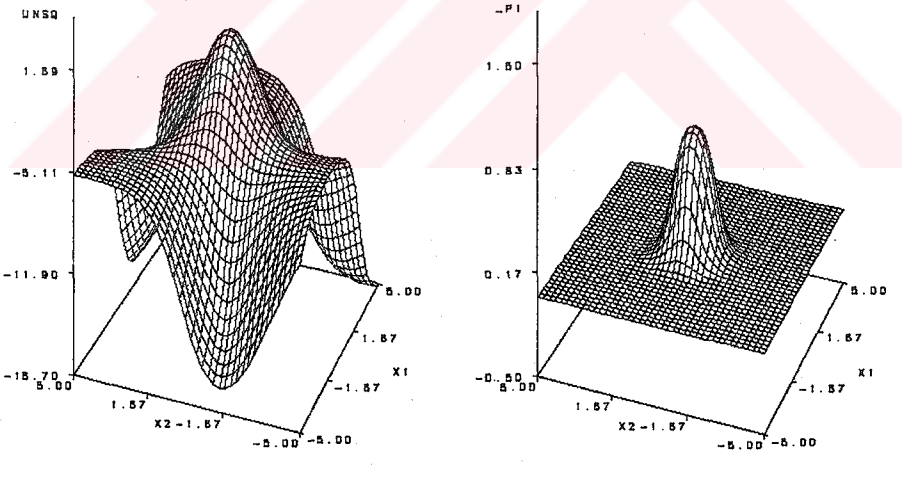
İki girdisi olan ve bir düzlemin ortasında normal dağılıma uyan bir tepe yer alan bir hedef fonksiyonu ele alınırsa (Şekil 3.15); çıktı aktivasyon fonksiyonu olarak özdeşliği kullanan bir ÇKİB YSA, az sayıda sigmoid (lojistik, tanh, arctan, vs.) saklı

birim ile tepeyi kolaylıkla uydurabilir, fakat düz olması gereken bölgede sahte sırt ve vadiler olacaktır (Şekil 3.16 (a)). Bu düzlemi hassas olarak düzgünleştirmek için birkaç düzine saklı birim eklenmesi gerekir (Şekil 3.16 (b)).



(a) (b)

Şekil 3.16 : (a) 6 saklı İE içeren ÇKİB YSA mimarisi; (b) 30 saklı İE içeren ÇKİB YSA (Sarle, 1997)



(a) (b)

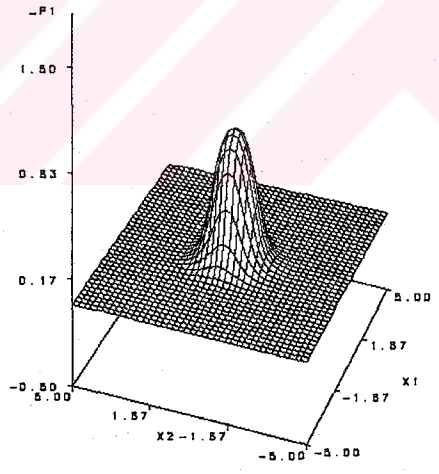
Şekil 3.17 : (a) 3 saklı İE içeren lojistik aktivasyon fonksiyonuna sahip bastırılmamış çıktılı ÇKİB YSA model; (b) Aynı modelin çıktısı bastırılmış şekli (Sarle, 1997)

Eğer çıktı katmanı aktivasyon fonksiyonu olarak lojistik kullanılırsa, lojistik fonksiyonunun girdi değeri negatif sonsuza giderken çıktısı sıfıra yaklaşır. Normal dağılıma sahip hedef fonksiyonunda düzlem de sıfır değerine sahiptir. Eğer çıktı

katmanında ağırlıklar ve yanlılık, tepenin tabanının dışında kalacak büyüklükte negatif değerler üretirlerse lojistik fonksiyonu daha önce elde edilen sahte sırt ve tepeleri düzgünleştirir. Bu yüzden, hedef fonksiyonun düz kısmını uydurmak kolaylaşır (Şekil 3.17 (a) ve (b)). Fakat lojistik fonksiyonu tepenin yüksekliğinin de azalmasına yol açacaktır.

Eğer yuvarlak bir tepe yerine yassı ve yanları dik ve yassı kısımda 1 değerine sahip bir tepe olursa, lojistik çıktı aktivasyon fonksiyonu aşağıda yer alan düzlemi düzgünleştirdiği gibi tepenin üst kısmını da yassılaştırabilir. Bu gibi hedef fonksiyonlar, pek çok sınıflandırma probleminde karşılaşılan gürültüsüz ve 0 veya 1 değerine sahip büyük düz alanlar içeren fonksiyonlardır. Tek saklı katman bu tip problemlerde iyi çalışır.

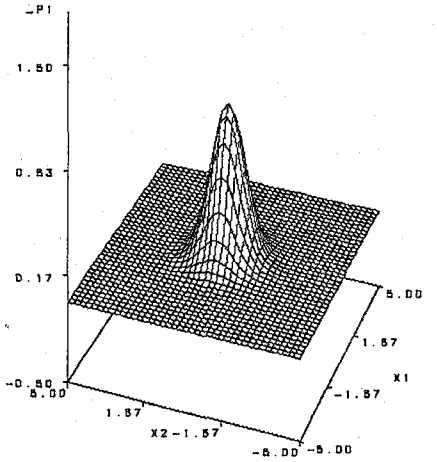
Çıktı katmanı aktivasyon fonksiyonu olarak lojistik kullanıldığında, hedef değerlerinin 0.1 ile 0.9 arasında ölçeklendirilmesi yaygın olarak kullanılan bir yaklaşımdır. Gürültü içermeyen sınıflandırma problemlerinde bu tip bir ölçeklendirme, lojistik fonksiyonunun düz alanları düzgünleştirmesini engelleyeceğinden iyi değildir (Şekil 3.18).



Şekil 3.18 : [0.1, 0.9] arasında ölçeklendirilmiş lojistik fonksiyonlu 3 saklı İE'li ÇKİB YSA (Sarle, 1997)

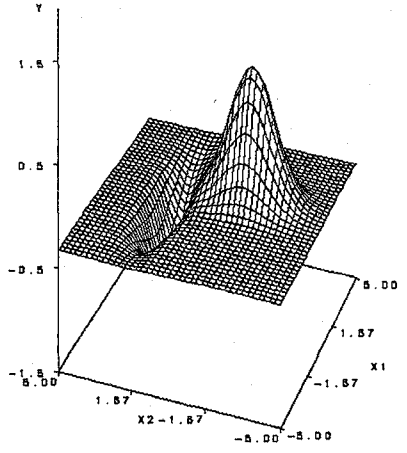
Hedef fonksiyonun normal dağılıma uyduğu durumlarda, [0.1, 0.9] arasında ölçeklendirme tepenin üst kısmının uydurulmasını kolaylaştırır, ancak, düzlemde yine dalgalanmalara yol açar. Bu durumda hedef fonksiyonun [0, 0.9] arasında ölçeklendirilmesi daha iyi sonuç verir.

Sigmoid aktivasyon fonksiyonlu ikinci bir saklı katman ekleyip çıktı katmanında özdeşliği kullanarak en iyi ölçeklendirme görülebilir (Şekil 3.19). Aslında, çıktı katmanının ağırlık ve yanlılık değerleri, hedef değerlerden ziyade çıktıyı ölçeklendirir ve hedef değerler için hangi aralık kullanılırsa kullanılsın uygundur.



Şekil 3.19 : Üç ve bir saklı İE içeren iki saklı katmanlı YSA (Sarle, 1997)

Özellikle birçok vadi ve tepeden oluşan daha karmaşık hedef fonksiyonlar için, ikinci saklı katmanda pek çok saklı İE'nin yer alması faydalı olur (Şekil 3.20). İkinci saklı katmandaki her bir İE, ağın ayrı bir tepe veya vadiyi tahmin etmesini sağlar. Bu yüzden, iki saklı katmanlı YSA sıklıkla tek saklı katmanlı YSA'ya göre daha az ağırlık ile daha hassas kestirimler verebilir.



Şekil 3.20 : Hedef fonksiyonu için tepe ve vadi verisi (Sarle, 1997)

İkiden daha fazla saklı katman, Kademeli Dizi Korelasyonu (Fahlman ve Lebiere, 1990) gibi bazı ağ mimarilerinde ve iki spiral problemi (Lang ve diğ., 1990) gibi uygulamalarda faydalı olabilir.

3.4.6 Verilerin Standartlaştırılması

Eğer YSA'ya ait girdi ve hedef çıktı değerleri bazı ön işlemlerden geçirilirlerse, ağın eğitilmesi daha verimli olarak yapılabilir. Bu ön işlemler, verilerin standardize edilmesi veya yeniden ölçeklendirilmesi olarak sayılabilir.

Normalize etme, bir vektörü normuna bölmekle gerçekleştirilebilir. Bu yöntemlerden biri, bir vektörün minimum ve aralık değerini bütün elemanları 0 ile 1 arasında değişecek şekilde yeniden ölçeklendirilmesidir. Ancak, ÇKİB ağlarda girdilerin [0,1] aralığında olması gerekir gibi yaygın bir yanlış düşünce varsa da aslında böyle bir koşul yoktur. Fakat girdi değerlerinin merkez sıfır olacak şekilde değişmesi daha iyi bir seçimdir.

Çıktı katmanında [0,1] aralığında değer alan bir aktivasyon fonksiyonu kullanılıyorsa, hedef değerlerin bu aralık içerisinde kalmasına dikkat edilmesi gerekir. Fakat genellikle daha iyi bir çözüm olarak hedef değerlerin dağılımına uygun bir aktivasyon fonksiyonunun kullanılmasıdır.

3.4.6.1 Girdi Değerlerinin Standartlaştırılması

Girdi değerlerinin standartlaştırılması kararı, ağın öncelikle bir sonraki katman (saklı veya çıktı katmanı) için net girdi değerini nasıl hesapladığına bağlıdır. Eğer girdiler, RBF ağında olduğu gibi bir mesafe fonksiyonu yardımıyla birleştiriliyorsa, girdi değerlerinin standartlaştırılması çok önemlidir. Eğer bir girdi 0 ile 1 aralığında, diğeri ise 0 ile 1.000.000 arasında değer alabiliyorsa, birinci verinin mesafeye olan katkısı ikinci veri tarafından zor duruma sokulacaktır. Bu nedenle verilerin, değişkenlikleri önem derecelerini yansıtacak en azından önemleriyle ters doğrultuda olmayacak şekilde yeniden ölçeklendirilmesi zorunludur.

Eğer girdiler, ÇKİB ağlarda olduğu gibi doğrusal olarak bir araya getiriliyorlarsa, verilerin standartlaştırılmasına nadiren gereksinim duyulur. Bununla birlikte, verilerin standartlaştırılmasının ağı daha hızlı eğitme ve yerel minimuma takılma olasılığını azaltma gibi pek çok pratik nedenleri vardır. Ayrıca, *ağırlık eksiltme* ve *Bayesgil tahmini* standart veriler ile daha uygun biçimde yapılabilir.

Girdi değişkenlerinin standartlaştırılması, ağırlıkların başlangıç durumuna getirilmesinde çok daha önemli etkiye sahiptir. Tek saklı katmanı olan bir ÇKİB ağın bir sınıflandırma problemini çözmede kullanıldığı varsayılsın: her bir hiperdüzlem, saklı birime net girdinin sıfır olduğu ve böylelikle sınıflandırma sınırının bu saklı birim tarafından somutlaştırıldığı noktaların oluşturduğu yüzeydir. Girdiler ile bir saklı birim arasındaki ağırlıklar hiperdüzlemin yönelimini belirler. Yanlılık ise hiperdüzlemin orijine olan mesafesini tayin eder. Bu yüzden, eğer veriler orijinde ortalanmamışsa, hiperdüzlem veri kümesinin üzerinden geçmeyebilir. Eğer bütün verilerdeki değişim katsayıları küçükse, başlangıçtaki hiperdüzlemlerin tamamının veri kümesinin dışında kalması bütünüyle mümkündür. Böylesine kötü bir başlangıç durumundan dolayı yerel minimuma takılma büyük olasılıkla söz konusu olacaktır. Bundan dolayı iyi bir rastsal başlangıç durumu için verilerin merkezileştirilmesi çok önemlidir.

ÇKİB ağlarda girdi verilerinin standartlaştırılması, farklı eğitme algoritmalarında farklı etkilere yol açar:

- *En hızlı iniş* ölçeklendirmeye çok hassastır. Hesiyen matrisinde gürültünün çok olması, yakınsamanın yavaşlamasına yol açar. Bu yüzden, gradyan azaltma yöntemlerinde ölçeklendirme çok önemlidir.
- *Yarı-Newton* ve *eşlenik gradyan* yöntemleri en hızlı iniş adımı ile başladıklarından onlar da ölçeklendirmeye hassastırlar. Ancak, eğitme süreci devam ederken ikinci-mertebeden bilgileri topladıklarından gradyan azaltma yöntemine göre daha az hassastırlar.
- *Newton-Raphson* ve *Gauss-Newton* yöntemi, eğer doğru gerçeklenmişlerse, ölçek değişimlerinden etkilenmezler.
- *Levenberg-Marquardt* yöntemi genellikle etkilenmezler.

3.4.6.2 Hedef Değerlerin Standartlaştırılması

Hedef değerlerin standartlaştırılması, iyi başlangıç ağırlıkları elde etmek için gerekli olmaktan daha çok uygundur. Ancak, eğer iki veya daha fazla hedef değişken var ve hata fonksiyonu ölçeklendirmeye duyarlı ise (EKKO gibi), her bir hedef değişkenin diğerlerine göre olan değişkenliği, ağın hedefi ne kadar iyi öğrendiğini etkileyebilir. Eğer, bir hedef değişken 0 ile 1 arasında değer alırken diğeri 0 ile 1.000.000 arasında değişiyorsa, ağ ikinci hedefi öğrenebilmek için eforunun büyük bir kısmını ona ayıracaktır. Bu yüzden, en azından değişkenlikleri önem derecelerini yansıtacak şekilde hedeflerin yeniden ölçeklendirilmesi önemlidir.

3.4.6.3 Verilerin Doğrusal Olmayacak Biçimde Dönüştürülmeleri

Hedef değerlerin doğrusal olmayan dönüşümleri, özellikle gürültülü veriler olduğunda çok önemlidir. Kullanılan hata fonksiyonlarından yaygın olan pek çoğu yalnızca bir farkın işlevidir (yani, abs(*hedef-çıktı*) gibi). Doğrusal olmayan dönüşüm bu farkların göreceli büyüklüklerini değiştirir. Pek çok hata fonksiyonunda ağ, gayretinin büyük bir kısmını bu farkın büyük olduğu hedef değerlerini öğrenmeye çalışmakla harcar.

Örneğin, bir hisse senedinin fiyatı YSA ile tahmin edilmeye çalışılsın. Eğer senedin fiyatı 10 ve ağın çıktısı 5 veya 15 ise, bu çok büyük bir hatadır. Eğer senedin fiyatı 1000 ve ağın çıktısı 995 veya 1005 ise hata oldukça önemsizdir. Ağın bu iki farkı aynı önemde görmesi istenmez. Logaritmaları alınarak hatalar, fark yerine oranlar cinsinden daha etkin olarak ölçülebilir, çünkü iki logaritma arasındaki fark orijinal değerlerin oranına karşılık gelir.

Ayrıca, düz fonksiyonların öğrenmesi düzgün olmayanlara göre genellikle daha kolaydır. Genelleştirme de yine düz fonksiyonlarda daha iyidir. Bu yüzden, girdi-çıktı fonksiyonunu düzgünleştiren dönüşümler genellikle faydalıdır.

3.5 Genetik Algoritma Teorisi

Genetik Algoritmalar (GA), çok büyük arama uzayının söz konusu olduğu problemler için uygun olan eniyileme yöntemleridir. GA, doğadaki Darwin'in evrimsel sürecini ve popülasyonun genetik özelliklerini temel alır (Goldberg, 1989). Bu yaklaşımda, sistemin bir çözümü kromozom olarak adlandırılan ve her birisi bir bireyi gösteren bir değerler dizgisi gibi betimlenir. Her birey, *uygunluk* olarak adlandırılan ve bireyin gücünü (yani, hayatta kalma ve bir sonraki nesilde üreme olasılığı) veya başka bir deyişle, çözümün iyiliğini gösteren bir özelliğe sahiptir. Genellikle yüksek uygunluk değeri daha iyi bir çözümü gösterir. Bu enbüyükleme problemleri için geçerlidir, fakat problem enküçükleme ise eniyileme işlevi enbüyükleme işlevine dönüştürülmelidir. Çözümlerden oluşan küme *popülasyon* olarak adlandırılır. Popülasyondaki bireyler çiftleşme (çaprazlama işlemi) yoluyla çoğalırlar ve genetik karakteristikleri doğal olmayan sonuçlardan (mutasyon işlemi) etkilenirler. Üreme için eşlerin seçimi genellikle daha güçlü bireylerin (daha iyi uygunluk değerlerine sahip çözümler) tercih edilmesiyle gerçekleştirilir. Yeni nesillerin üretilmesi, yinelemeli olarak bir önceki nesile GA operatörleri uygulanarak sistemin belirli bir yakınsama düzeyine (belirli bir

eşik değerinin üzerinde uygunluğa sahip çözüm veya çözümler üretilinceye kadar) veya verilen bir yineleme sayısına erişinceye kadar sürdürülür.

3.5.1 Operatörler

Genetik Algoritmalarda başlıca üç operatör vardır: Çaprazlama, Mutasyon ve Seçim. Operatörlerin tanımları açık olmakla birlikte, bir probleme uygulanmaları oldukça farklı olabilir.

3.5.1.1 Çaprazlama

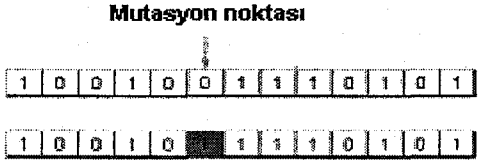
Evrilme süreci kromozom temelli bir süreçtir, yani üreme süreci bireyler üzerinde değil de kromozomlarda gerçekleşir. Üreme gerçekleştiğinde her bir ebeveynin genetik bilgilerinin bir kısmı evlatlarına aktarılır. Kromozomların örüntüleri değiş tokuş yapılarak, ebeveynlerinin bazı karakteristiklerini taşıyan evlatlar üretecek şekilde birleştirilirler. Yani, her bir ebeveynin karakteristiklerinden bir kısmı çocuğa miras olarak geçer. Evrimsel süreçteki bu işlem *çaprazlama* olarak adlandırılır.

Şekil 3.21 : Çaprazlama operatörü

3.5.1.2 Mutasyon

Mutasyon, belirli bir bireyin kromozom düzeninde rastsal bir değişikliğin yapılmasıdır. Ancak, mutasyon oranı çok yüksekse yaratılan bireyler rasgele oluşturulduğundan bir önceki nesilden çok farklı olacaktır. Bu ise kararsızlığa veya sistemin yakınsamamasına neden olabilir. Diğer yandan, uygun bir mutasyon düzeyi sistemin prematüre bir yakınsamasına (yerel minimum) engel olur. Prematüre

yakınsama, birbirine çok benzeyen ve yüksek, ancak olası en yüksek olmayan uygunluk değerine sahip bireylerin popülasyonu oluşturduğu bir durumdur. Popülasyondaki bireyler çok benzer olduklarından, sistemin daha yüksek uygunluğa sahip bireyleri değerlendirme şansı yoktur. Bu yüzden, doğada olduğu gibi sisteme bir miktar mutasyon eklemek faydalı olur.



Şekil 3.22 : Mutasyon operatörü

3.5.1.3 Seçim

Çaprazlama ve mutasyon işlemlerinden sonra bazıları daha yüksek uygunluk değerlerine sahip farklı bireylerden oluşan yeni nesil oluşur. Ardından seçim işlemi başlar ve popülasyondaki daha güçlü bireyler hayatta kalarak bir sonraki neslin bireylerine geçebildikleri takdirde üreme şansı kazanırlar. *Seçici basınç* terimi seçimin daha yüksek uygunluk değerine sahip bireylere eğilim gösterildiğini ifade eder. Ancak, bazı problemlerde, yüksek seçici basınç olması durumunda sonuç yerel olarak en iyi olmakla birlikte bir kaç nesil sonra genel en iyiyi kaçırabilir veya yakınsamayabilir.

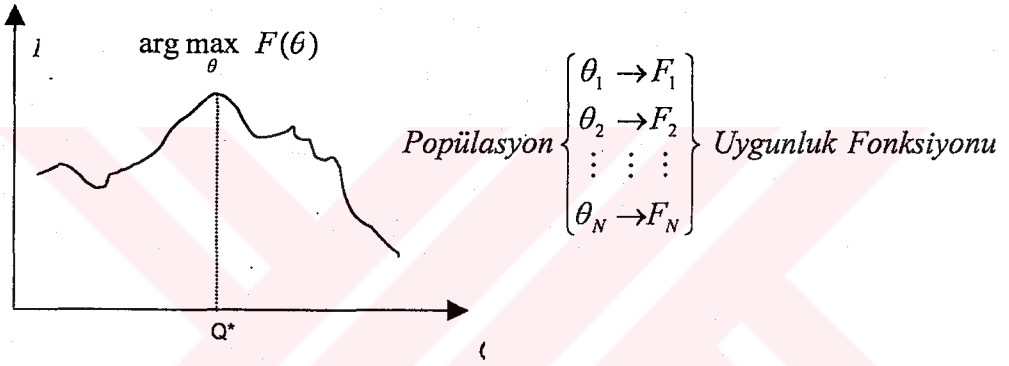
3.5.2 Başlangıç Popülasyonu

Her nesil bir öncekinden elde edildiğinden, bireylerin uygunluk değerlerinin popülasyondaki dağılımı bir önceki popülasyona oldukça bağımlıdır. Bu yüzden başlangıç durumu dikkatle ele alınmalıdır. Rastsallığa dayanan yaklaşımlar oldukça uygun görülebilir. Aslında uygun olmakla beraber rastsallığın doğası gereği çok büyük miktardaki birey oldukça düşük uygunluk değerine sahip olabilir. Bu ise sürece bazı sezgisel yaklaşımlar ekleyerek ve başlangıç popülasyonundaki bireyleri sadece rastsal olarak değil aynı zamanda çözüm uzayı ile ilgili bazı bilgileri temel alarak bertaraf edilebilir. Fakat sezgiselliğin düzeyi çok dikkatlice ele alınmalıdır. Çünkü çözümün kalitesi bakımından diğerlerine benzeyen bireyleri oluşturmak sistemin prematüre yakınsamasına veya yerel en iyiye gitmesine neden olabilir.

3.5.3 Genetik Algoritmalar Kullanarak Çok Katmanlı İleri Beslemeli Yapay Sinir Ağında Parametre Tahmini

Genetik eniyileme, bir sinir ağının verimliliğini ve etkinliğini geliştirmek için güçlü bir araçtır. Sinir ağının performansı büyük ölçüde iyileşecek şekilde ağ parametrelerinin hassas ayarı genetik eniyileme ile yapılabilir.

Genetik algoritmalarda arama, yeniden üretim mekanizmasıyla yüksek uygunluğa sahip bölgelere doğru yönlendirilir. Bu bölgelerdeki yeni çözüm noktalarına ulaşmada ise genetik operatörler kullanılır. Genetik operatörler, yığının genetik bilgilerini kullanarak yeni çözümleri elde ederler. Genetik algoritma ile arama prosedürü bir başlangıç popülasyonu ile başlar ve başarılı popülasyonları genetik operatörlerle uygunluk fonksiyonuna sokarak türetir (Şekil 3.23).



Şekil 3.23 : Genetik algoritmalarda uygunluk fonksiyonu

Popülasyondan seçilen $2N$ eleman uygunluk değerlerine göre kodlanarak N eleman içeren iki dizi (kromozom) oluşturulur. Daha sonra bu iki dizi tek veya çok noktadan çaprazlanarak yeni N eleman içeren iki dizi elde edilir (Goldberg, 1989). Çaprazlamadaki temel amaç, farklı uygun çözümler arasında bilgi değişimini sağlayarak arama uzayının farklı bölgelerine ulaşmaktır. Bir popülasyona çaprazlama P_i olasılıkla uygulanır.

$$P_i = \frac{F_i}{\sum_{ka} F_k} \; ; \; F_i = F(\theta_i) + \alpha(Çeşitlilik) \tag{3.41}$$

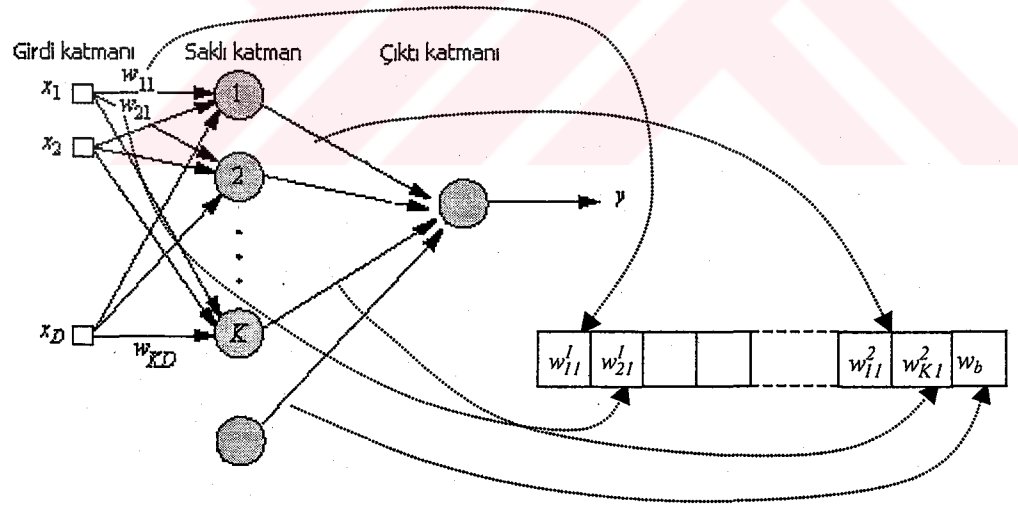
Mutasyon kullanılarak bir kromozom içerisindeki bir veya birkaç değer küçük bir olasılıkla rastsal olarak değiştirilerek yığında yeni dizilerin (arama uzayında yeni çözüm noktalarının) elde edilmesi sağlanır (Goldberg, 1989). Her yığına mutasyon

operatörü P_m olasılığı ile uygulanır. Ayrıca, *çeşitlilik* (diversity) ile hem $F(\theta)$ hem de θ değerine bakılarak farklılıklar bir sonraki nesile taşınmaya çalışılır.

$$Çeşitlilik = \left| \theta_i - \frac{\sum_k \theta_k}{N} \right| \tag{3.42}$$

GA'da *üretim-değerlendirme-seçim* çevrimi, önceden belirlenen çevrim sayısına ulaşıncaya ya da yığının ortalama uygunluk değeri yığındaki en iyi dizinin uygunluk değerine önceden belirlenen oranda yaklaşıncaya kadar devam eder.

GA'nın en yaygın gösterimi ikili (*binary*) düzendedir (Goldberg, 1989). Kromozomlar, genellikle {0,1} değerlerinden oluşan genlerden meydana gelirler. Bu yüzden bir kromozom l adet gen (c_i) içeren bir x vektörüdür: $\{x = (c_1, c_2, ..., c_l),\ c_1 \in \{0,1\}\}$. Ancak, değişkenleri sürekli tanım kümesinde yer alan eniyileme problemlerinde, genleri doğrudan gerçek sayılar olarak göstermek daha doğaldır. Yani, genotip (*kodlama*) ile fenotip (*arama uzayı*) arasında bir fark yoktur.



Şekil 3.24 : ÇKİB'nin ağırlıklarının genetik algoritma ile hesaplamak için kodlanması

Yapay sinir ağında, gerçek sayı olan ağırlıklar kodlanarak ağırlık dizgesi (kromozom) oluşturulur (Şekil 3.24). GA operatörleri ve arama prosedürü kullanılarak en iyi ağırlık değerleri bulunmaya çalışılır. Yerel bir arama tekniği olan GY yerine genel bir arama prosedürü olan GA kullanılarak en iyi ağırlık değerleri bulunabilir.

4. REKABETÇİ ÖĞRENME VE KENDİNİ DÜZENLEYEN AĞLAR

Kombinatorik eniyileme problemlerini çözmede kullanılan sinir ağlarından birisi Kohonen'in kendini düzenleyen ağlarıdır (Smith ve diğ., 1996a). Kohonen ağları Öklit olmayan problemlerde genelleştirilememe gibi dezavantaja sahip olmalarına rağmen; tavlama, melez modeller, vb. yaklaşımlarla birlikte özellikle kümeleme ve ikinci dereceden programlamada oldukça iyi sonuçlar vermiştir (Smith ve diğ., 2000). Bu bölümde, rekabetçi öğrenme ve kendini düzenleyen ağlar açıklanacaktır.

4.1 Rekabetçi Öğrenme

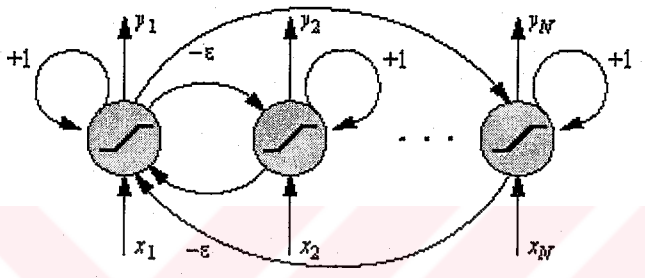
Bir dağıtık sisteme ait elemanların işlevlerini eniyileme ve çeşitlendirmenin yollarından birisinin kaynaklar için yarışma olduğu söylenebilir. Bu rekabetçi durumu, toplumun örgütlenme biçiminden biyolojiye kadar her alanda gözlemleyebiliriz. Rekabetin önemli bir yönü, sistem kaynaklarını paylaşmada genel kontrole gereksinim duymaksızın yerel düzeyde eniyilemeyi sağlamasıdır. Genel kontrol, dağıtık sistemlerde muazzam miktarlarda büyük maliyete katlanmadan gerçeklenemeyeceğinden, rekabet bu sistemler açısından büyük öneme sahiptir.

Rekabetçi ağda işlem elemanları girdiden eşdeğer bilgileri alırlar, ancak, topolojideki yanal ketleyiciler veya öğrenme kuralının formülasyonu vasıtasıyla kaynaklar için yarışırlar (Şekil 4.1). İE girdi uzayının farklı bir alanında uzmanlaşır ve çıktıları, girdi uzayı yapısının farklı bir şekilde gösterilmesinde kullanılabilir.

Çok katmanlı ileri beslemeli ağlar, Radyal Temelli Fonksiyonlar (RTF) gibi mimariler ve daha önceki bölümde değinilen öğrenme kuralları rekabet düşüncesini açık olarak kullanmazlar. Doğrusal olmayan bir ağda hataların geriye yayılımı İE'nin özelleşmesine yol açar, fakat çok maliyetlidir (yavaş yakınsama) ve sadece dolaylı olarak gerçekleşir. Rekabet, özünde doğrusal olmayan bir işlemdir ve matematiksel işlemleri, adaptif sistemlerin diğer alanlarının gerisinde kalmıştır (Principe ve diğ., 2000). Ancak, ağ kaynaklarını kendi kendine düzenleyen hızlı bir yöntem olarak cazibesi ulaşılmazdır ve pratikte çok etkin olarak kullanılmaktadır (Sarle, 1994).

Rekabet, **güçlü** ve **hafif** olarak iki temel kategoriye ayrılabilir. Güçlü rekabette, sadece tek bir işlem elemanı kaynakları elde eder (*kazanan İE*). Hafif rekabette ise ortada açık olarak bir kazanan vardır fakat kazananın komşuları da sistem kaynaklarından küçük yüzdelerde pay alırlar. Nörobiyoloji'de hafif rekabet çok yaygındır ve bu alanda geliştirilen modellere esin kaynağı olmuştur (Kohonen, 1982; 1990; 1995; 1997).

$$y_k = \begin{cases} 1 & x_k \text{ enbüyük} \\ 0 & \text{diğer durumlarda} \end{cases}$$



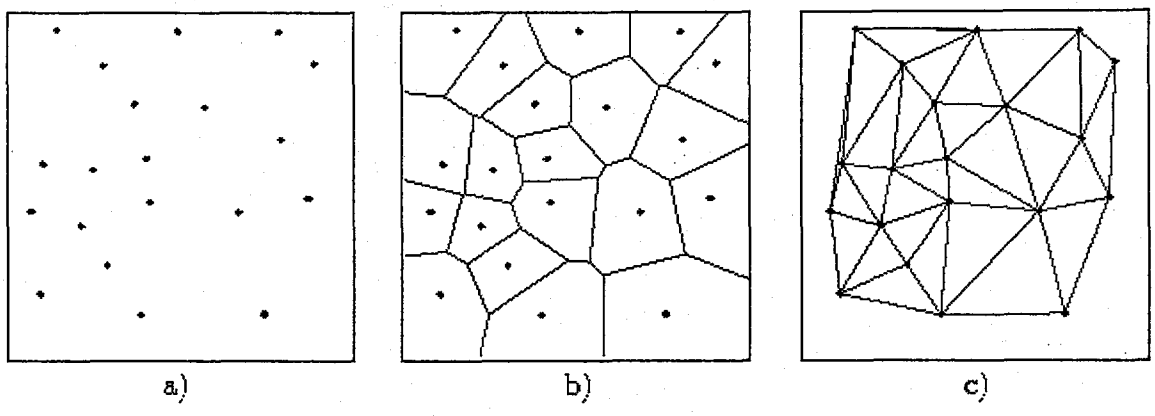
Şekil 4.1 : Rekabetçi öğrenme

Rekabetçi öğrenmeyi anlamada önemli olan, sayısal geometri açısından birbiriyle yakından ilişkili ve temel iki kavram *Voronoi Çokgenleri* ve *Delaunay Üçgenleri*'dir:

$\Re^n$'de $\mathbf{w}_1$, $\mathbf{w}_2$, ... , $\mathbf{w}_N$'den oluşan bir vektör kümesi verilsin (Bkz. Şekil 4.2 (a)). Herhangi bir vektörün ($\mathbf{w}_i$) *Voronoi* bölgesi, $\Re^n$'de kendilerine en yakın vektörün $\mathbf{w}_i$ olduğu bütün noktaların oluşturduğu küme olarak tanımlanabilir:

$$V_i = \left\{ \mathbf{x} \in \Re^n \,\middle|\, i = \arg\min_{j \in \{1,\ldots,N\}} \left\| \mathbf{x} - \mathbf{w}_j \right\| \right\} \qquad (4.1)$$

Her bir veri noktasının sadece bir Voronoi bölgesi ile ilişkilendirilmesini sağlamak için birden fazla alternatif olması durumunda en yakındaki referans vektörlerinden birisine rastsal olarak eşleştirilir. Her Voronoi bölgesi (V_i) dışbükey bir alandır.



Şekil 4.2 : (a) $\Re^2$'deki noktalar kümesi, (b) İlgili Voronoi çokgenleri, (c) İlgili Delaunay üçgenleri (Fritzke, 1994a)

Voronoi çokgenleri ile oluşturulmuş $\Re^n$'deki bal peteği şeklindeki bölümler **Voronoi çokgenleri** olarak adlandırılır (Şekil 4.2 (b)). Bu bölünmeleri hesaplamada sadece iki-boyutlu veri kümeleri için etkin algoritmalar geliştirilmiştir (Preparata ve Shamos, 1990). Ancak bu kavram herhangi bir boyuttaki uzayda uygulanabilmektedir.

Eğer bir kenarı ortak olan Voronoi bölgelerine ait noktalar birleştirilirse **Delaunay üçgenleri** elde edilir (Şekil 4.2 (c)). Delaunay üçgenleri, farklı hususlara göre oluşturulabilecek diğer üçgenlere göre özeldir. Bu üçgenlerin fonksiyon enterpolasyonunda en iyi olduğu gösterilebilir (Fritzke, 1994a). İlerde anlatılacak olan *rekabetçi Hebbian öğrenme* yöntemi, Delaunay üçgenlerinin sadece girdi uzayının verilerin bulunduğu kısımları ile sınırlı bir alt grafiğini oluşturur.

4.2 Rekabetçi Öğrenmede Amaçlar

Rekabetçi öğrenme sistemleri için farklı ve çoğunlukla çok yönlü amaçlar belirlenebilir. Bunlardan bazılarına bu bölümde değinilecektir.

4.2.1 Hatanın Minimize Edilmesi

Sıklıkla karşılaşılan hedeflerden birisi beklenen sayısal ölçüm hatasının minimize edilmesidir. Girdi verilerinin sürekli bir dağılıma ($p(\mathbf{x})$) göre olması durumunda, verilen ağdaki işlem birimleri kümesi ($\mathcal{A}$: $\mathcal{A}=\{ c_1, c_2, \ldots, c_N \}$) için, referans vektörleri w_c, $c \in \mathcal{A}$'nın değerlerini hatayı minimum yapacak şekilde bulmaktır.

$$E(p(\mathbf{x}), \mathcal{A}) = \sum_{c \in \mathcal{A}} \int_{V_c} \|\mathbf{x} - \mathbf{w}_c\|^2 p(\mathbf{x}) d\mathbf{x} \qquad (4.2)$$

Girdi verilerinin ayrık (sonlu) olması durumunda (D) ise hata:

$$\mathrm{E}(D, \mathcal{A})=\frac{1}{|D|}\sum_{c\in A}\sum_{\mathbf{x}\in R_c}\|\mathbf{x}-\mathbf{w}_c\|^2 \tag{4.3}$$

c biriminin Voronoi kümesi olan R_c'ye göre minimum yapılmaya çalışılır. Hatanın minimize edilmesinin önemli olduğu uygulamalara tipik bir örnek olarak *vektör nicelendirme* (Linde ve diğ., 1980; Gray, 1984) verilebilir.

4.2.2 Entropinin Maksimize Edilmesi

Referans vektörleri, rasgele oluşturulmuş girdi verilerine göre kazanan olmak için her birisi eşit şansa sahip olacak şekilde dağılmış olabilir:

$$P(s(\mathbf{x})=c)=\frac{1}{|A|} \qquad (\forall c\in A) \tag{4.4}$$

Eğer bir girdi verisinin oluşturulması ve ondan sonra $\mathcal{A}$'daki en yakın birime eşleştirme, bir $\mathbf{x}\in\mathcal{A}$'nın rastsal bir değişken olan X'e atamanın yapıldığı rastsal bir deney gibi yorumlanırsa, o zaman (4.4) eşitliği aşağıdaki entropinin maksimize edilmesine eşit olacaktır (Fritzke, 1994a).

$$H(X)=-\sum_{x\in A}P(x)\log(P(x))=E\left(\log\left(\frac{1}{P(x)}\right)\right) \tag{4.5}$$

Eğer girdi verileri sürekli bir olasılık dağılımından ($p(\mathbf{x})$) yaratılmışsa, (4.4) eşitliği;

$$\int_{V_c}p(\mathbf{x})d\mathbf{x}=\frac{1}{|A|} \qquad (\forall c\in A) \tag{4.6}$$

Girdi verilerinin ayrık (sonlu) olması durumunda (D) ise:

$$\frac{|R_c|}{|D|}\approx\frac{1}{|A|} \qquad (\forall c\in A) \tag{4.7}$$

Referans vektörlerini entropiyi maksimum yapacak şekilde seçmenin bir avantajı, sonuçta elde edilen sistemin içsel olarak gürbüz olmasıdır. Herhangi bir referans vektörünün çıkarılması (veya yok olması) verileri çok sınırlı oranda etkiler.

Entropinin enbüyüklenmesi ile hatanın enküçüklenmesi genellikle aynı anda gerçekleştirilemez. Özellikle verilerin dağılımı büyük ölçüde düzgün değilse (*non-uniform*), her iki hedef birbirinden oldukça farklılaşacaktır.

4.2.3 Özellik Eşleştirme

Bazı ağ mimarilerinde, büyük boyuttaki girdi verilerini, aralarında mevcut olan benzerlik ilişkilerini koruyacak şekilde daha küçük boyuttaki bir yapıya dönüştürmek mümkündür. Bu işlem *öznitelik çıkarımı* veya *özellik eşleştirme* olarak adlandırılır ve özellikle verilerin görselleştirilmesinde çok faydalı olabilmektedir (Ripley, 1996). Bu yöntemin ön koşulu kullanılan ağın sabit büyüklükte olmasıdır (örneğin, kendi kendini düzenleyen ağlar).

Girdi verileri uzayından ayrık ağ yapısına dönüştürürken topolojinin (başka bir deyişle benzerliklerin) nasıl korunacağı hemen akla gelebilir. Bu konuda önerilen pek çok sayısal ölçümler önerilmiştir (Bauer ve Pawelzik, 1992; Bauer ve Villmann, 1995).

4.3 Güçlü Rekabet

Güçlü rekabetçi öğrenme (yani, kazanan hepsini alır), her bir girdi verisinin sadece bir birimin (yani kazananın) uyarlamasını belirlediği yöntemleri içerir. Farklı yöntemler, *grup* veya *eşzamanlı* güncelleme kullanılarak elde edilebilir. Grup yöntemlerinde (örneğin, LBG), herhangi bir uyarlanmadan önce ilk olarak bütün olası girdi verileri değerlendirilir. Bu işlem birçok kez tekrarlanır. Eşzamanlı yöntemlerde ise (örneğin, k-ortalamaları) her girdi verisinden sonra doğrudan güncelleme yapılır.

Güçlü rekabetçi öğrenmede karşılaşılan genel problemlerden birisi "*ölü nöronların*" olmasıdır. Bu nöronlar, bir girdi verisi için hiçbir zaman kazanan olamaz ve pozisyonları süresiz olarak aynı kalır. Bu nöronlar, amaç ne olursa olsun ağa herhangi bir katkıda bulunmazlar ve ağın kaynaklarını boşu boşuna kullandıkları için zararlı kabul edilmelidirler. Ölü nöronlardan kaçınmanın bir yolu, referans vektörlerini başlangıç durumuna getirirken $p(\mathbf{x})$'e göre farklı örnekleme vektörlerini kullanmaktır.

Ayrıca bir diğer yaklaşım da referans vektörlerinin dağılımını belirli bir hedefe göre uyarlarken nöronların araya eklenmesi ve çıkarılmasında yerel istatistiklerin kullanılmasıdır.

Bir diğer problem, farklı rastsal başlangıç durumlarının farklı sonuçlara yol açmasıdır. Bütünüyle yerel olan uyarlamalar, sistemin başlangıç durumuna bağlı olmaksızın yerel minimumlardan kurtulmasını sağlayamayabilir. Bu problemi çözmenin yollarından birisi, güçlü rekabetçi öğrenmenin *"kazanan hepsini alır"* yaklaşımı yerine hafif rekabetçi öğrenmenin *"kazanan çoğunu alır"* yaklaşımını kullanmaktır. Bu durumda sadece kazanan değil diğer bazı nöronlar da uyarlanırlar. Bu ise başlangıç durumuna olan bağımlılığı azaltır.

4.3.1 Grup Olarak Güncelleme

Grup Olarak Güncelleme (LBG) algoritması (Linde ve diğ., 1980; Lloyd, 1957), bütün referans vektörlerini, tekrarlı olarak kendilerine ait Voronoi kümelerinin aritmetik ortalamalarına doğru hareket ettirmek suretiyle gerçeklenir. Bunun teorik gerekçesi ise, referans vektörlerinin oluşturduğu bir küme $\{\mathbf{w}_c | c \in A\}$ için hatayı minimum (Denklem 4.3) yapmada gerekli bir koşul, her bir referans vektörünün ($\mathbf{w}_c$) *kitle merkezi şartını* karşılamasıdır (Gray, 1992). Girdi verilerinin sonlu olması ve Öklit mesafe fonksiyonunun kullanılması durumunda kitle merkezi şartı aşağıdaki biçime indirgenir:

$$\mathbf{w}_c = \frac{1}{|R_c|} \sum_{\mathbf{x} \in R_c} \mathbf{x} \tag{4.8}$$

LBG algoritmasının bütün adımları aşağıda verilmiştir:

1. A kümesini, D sonlu veri kümesinden rasgele seçilmiş $\mathbf{w}_{c_i} \in \Re^n$ referans vektörlerine sahip, N adet c_i'yi içerecek şekilde başlangıç durumuna getir:

$$A = \{ c_1, c_2, \ldots, c_N \} \tag{4.9}$$

2. Her işlem birimi ($c \in A$) için o birime ait Voronoi kümesini (R_c) hesapla.
3. Her işlem biriminin referans vektörünü ona ait Voronoi kümesinin ortalamasına götür (Denklem 4.8).

4. Eğer Adım 3'de herhangi bir w_c değişirse, Adım 2'ye git.

5. Halihazırdaki referans vektörleri kümesini belirle.

Adım 2 ve 3, hata işlevinin azalmasını veya en azından değişmeden kalmasını sağlayan *Lloyd yinelemesi* olarak adlandırılan formu oluşturur. LBG sonlu sayıda Lloyd yinelemesinden sonra hata işlevinin bir yerel minimumuna yakınsamayı garanti altına alır.

LBG-U olarak adlandırılan ve LBG'nin daha geliştirilmiş hali olan yöntem LBG tarafından bulunan yerel minimumdan genellikle daha iyi sonuç verebilmektedir (Fritzke, 1997).

4.3.2 Eşzamanlı Güncelleme

Bazı durumlarda girdi veri kümesi (D), grup olarak güncelleme yöntemlerinin uygulanmasını elverişsiz kılacak kadar büyük olabilir. Veya girdi verileri, grup yöntemlerinin uygulanmasını bütünüyle olanaksız kılacak şekilde sonsuz uzunluktaki sürekli bir diziden gelebilir. Bu durumda *eşzamanlı güncelleme* uygulanır:

1. $\mathcal{A}$ kümesini, $p(x)$'e göre rasgele seçilmiş $\mathbf{w}_{c_i} \in \Re^n$ referans vektörlerine sahip, N adet c_i'yi içerecek şekilde başlangıç durumuna getir (Denklem 4.9).

2. $p(x)$'ye göre rasgele bir girdi verisi (x) oluştur.

3. Kazanan nöronu belirle ($s = s(x)$):

$$s(\mathbf{x}) = \arg\min_{c \in A} \|\mathbf{x} - \mathbf{w}_c\| \tag{4.10}$$

4. Kazanan nöronun referans vektörünü x'e doğru uyarla:

$$\Delta(\mathbf{w}_s) = \eta \|\mathbf{x} - \mathbf{w}_s\| \tag{4.11}$$

5. Maksimum işlem sayısına ulaşılmamışsa Adım 2'ye git.

Öğrenme oranı (η), kazananın girdi verisine doğru uyarılma oranını belirler. Öğrenme oranının sabit veya azalan değerler almasına göre farklı yöntemler kullanılabilir.

4.3.2.1 Sabit Öğrenme Oranı

Eğer öğrenme oranı sabitse ($\eta = \eta_0$, $0 < \eta_0 \leq 1$), kazanan nöron c için her bir referans vektörünün ($\mathbf{w}_c$) değeri girdi verilerinin üssel olarak azalan ortalamasına eşit olacaktır. Yani, c'nin kazanan nöron olduğu durum için, $\mathbf{x}_1^c$, $\mathbf{x}_2^c$,..., $\mathbf{x}_n^c$ girdi verileri dizisi olarak alınırsa, $\mathbf{w}_c$'nin eşit olduğu ardışık değerler aşağıdaki gibi yazılabilir:

$$\mathbf{w}_c(0) = p(\mathbf{x})\text{'ye göre rastsal girdi verisi}$$

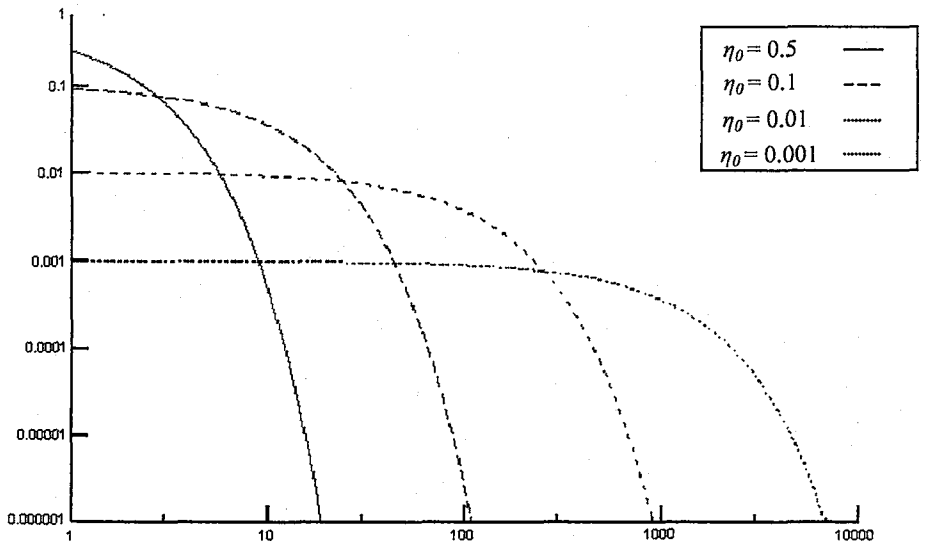
$$\begin{aligned}\mathbf{w}_c(1) &= \mathbf{w}_c(0) + \eta_0 (\mathbf{x}_1^c - \mathbf{w}_c(0)) \\ &= (1 - \eta_0)\, \mathbf{w}_c(0) + \eta_0 \mathbf{x}_1^c \end{aligned} \tag{4.12}$$

$$\begin{aligned}\mathbf{w}_c(2) &= (1 - \eta_0)\mathbf{w}_c(1) + \eta_0 \mathbf{x}_2^c \\ &= (1 - \eta_0)^2 \mathbf{w}_c(0) + (1 - \eta_0)\eta_0\, \mathbf{x}_1^c + \eta_0 \mathbf{x}_2^c \end{aligned} \tag{4.13}$$

$$\vdots$$

$$\begin{aligned}\mathbf{w}_c(n) &= (1 - \eta_0)\mathbf{w}_c(n - 1) + \eta_0\, \mathbf{x}_n^c \\ &= (1 - \eta_0)^n \mathbf{w}_c(0) + \eta_0 \sum_{i=1}^{n} (1-\eta_0)^{n-i}\, \mathbf{x}_i^c \end{aligned} \tag{4.14}$$

Denklem 4.14'den de görülebileceği gibi, geçmiş girdi verilerinin sonraki girdi verilerine olan etkisi üssel olarak hızlı bir şekilde azalacaktır (Şekil 4.3). Ancak, en son girdi verisi her zaman $\mathbf{w}_c$'nin o anki değerinin bir oranını (η) belirler. Buradan iki sonuca varılabilir: Birincisi, böyle bir sistem adaptif davranış gösterir ve bu nedenle prensipte girdi verilerinin kararlı olmayan dağılımını ($p(\mathbf{x})$) takip edebilir. İkincisi, yakınsama gerçekleşmez. Çok büyük miktarda girdi verileri kullanıldıktan sonra dahi güncel girdi verisi kazananın referans vektöründe önemli ölçüde değişime yol açabilir. Eğer verilerin dağılımı kararlı olursa, referans vektörleri yarı-kararlı bir noktaya doğru hareket edecekler ve bu nokta etrafında salınım hareketi yapacaklardır. Daha iyi bir yarı-kararlı noktaya erişebilmek için daha küçük uyarılma oranı kullanılmalıdır. Ancak bu durumda da daha fazla uyarılma işlem adımına gereksinim duyulur.



Şekil 4.3 : Girdi verilerinin (x) farklı öğrenme oranları için, ardışık girdi verisi sayısının bir fonksiyonu olarak kazanan nöron (s) vektörü üzerindeki etkileri (Fritzke, 1994a)

4.3.2.2 Harmonik Olarak Azalan Öğrenme Oranı

Sabit bir öğrenme oranı yerine, zamana bağlı olarak azalan bir öğrenme oranı da kullanılabilir. Her bir $c \in \mathcal{A}$ için ayrı bir öğrenme oranı kullanılabilir ve harmonik bir seriye göre değer almaları sağlanabilir:

$$\eta(t) = \frac{1}{t} \tag{4.15}$$

Bu seride zaman parametresi (t), o ana kadar kazanan birim için olan girdi verilerinin sayısını gösterir. Bu algoritma *k-ortalamaları* (MacQueen, 1967) olarak bilinir. $\mathbf{w}_c$'nin ardışık değerler dizisi:

$\mathbf{w}_c(0) = p(\mathbf{x})$'ye göre rassal girdi verisi

$$\begin{aligned} \mathbf{w}_c(1) &= \mathbf{w}_c(0) + \eta(1)(\mathbf{x}_1^c - \mathbf{w}_c(0)) \\ &= \mathbf{x}_1^c \end{aligned} \tag{4.16}$$

$$\begin{aligned} \mathbf{w}_c(2) &= \mathbf{w}_c(1) + \eta(2)(\mathbf{x}_2^c - \mathbf{w}_c(1)) \\ &= (\mathbf{x}_1^c + \mathbf{x}_2^c)/2 \end{aligned} \tag{4.17}$$

$$\mathbf{w}_c(t) = \mathbf{w}_c(t-1) + \eta(t)(\mathbf{x}_n^c - \mathbf{w}_c(t-1))$$
$$= (\mathbf{x}_1^c + \mathbf{x}_2^c + \ldots + \mathbf{x}_n^c)/t \quad (4.18)$$

Belli bir birimin (c) kazanan olduğu durumda $\mathbf{x}_1^c, \mathbf{x}_2^c, \ldots, \mathbf{x}_n^c$ girdi verileri kümesi c'nin o anki Voronoi bölgesinin dışında kalan elemanları da içerebileceği unutulmamalıdır. Çünkü $\mathbf{w}_c$'nin her uyarılmasında Voronoi bölgesi (V_c) değişir.

k-ortalamaları algoritması ile ilgili bir diğer önemli nokta da LBG'de olduğu gibi kesin bir yakınsamanın olmamasıdır. Çünkü harmonik serilerin toplamı ıraksar (4.19). Bu ıraksamadan dolayı, birçok girdi verisinden ve düşük öğrenme oranından sonra dahi her bir girdi vektöründe büyük değişiklikler söz konusu olabilir.

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{i} = \infty \quad (4.19)$$

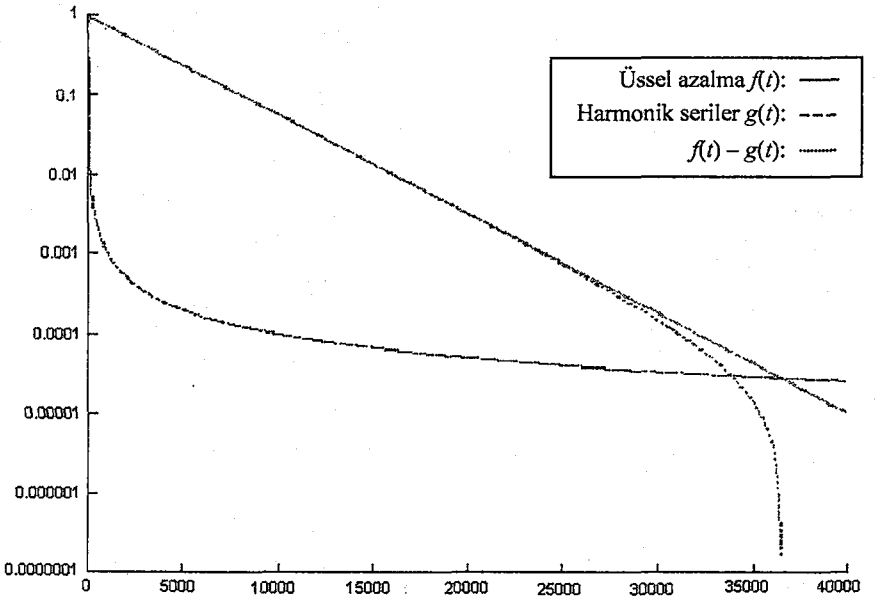
4.3.2.3 Üssel Olarak Azalan Öğrenme Oranı

Kendi kendini düzenleyen ağlar bağlamında, uyarlama oranını azaltmanın bir diğer yolu da Ritter ve diğ. (1991) tarafından önerilen üssel olarak azaltmadır:

$$\eta(t) = \eta_i (\eta_f / \eta_i)^{t/t_{max}} \quad (4.20)$$

Denklem 4.20'de η_f ve η_i öğrenme oranının başlangıç ve son değerleri, t_{max} toplam uyarılma adımı sayısıdır.

Şekil 4.4'de üssel olarak azalan öğrenme oranı ile harmonik serilerin karşılaştırılması grafiksel olarak gösterilmiştir. Başlangıçta harmonik serilere nazaran üssel olarak azalan öğrenme oranı oldukça büyüktür. Bu durum, sisteme gürültü ekleyip daha sonra yavaş yavaş azaltmaya benzediğinden tavlama benzetimi tekniği ile ilişkilendirilmiştir (Kirkpatrick ve diğ., 1983). Tavlama benzetimi tekniği, bir sistemin yerel minimumlarından kaçınabilmesine olanak tanır.



Şekil 4.4 : Üssel olarak azalan öğrenme oranı ile harmonik serilerin belirli parametre değerleri için karşılaştırılması ($\eta_i = 1$, $\eta_f = 0.00001$, $t_{max} = 40000$) (Fritzke, 1994a)

4.4 Hafif Rekabet

Hafif rekabet, güçlü rekabetin aksine sadece kazananın değil, farklı yollarla tanımlanabilen komşularının da aktif olmasını sağlar. Hafif rekabet, çıktı uzayında aktivitenin, en yakın İE'nin en çok aktif (en fazla çıktı) ve komşularının daha az aktif olduğu bir *baloncuğunu* yaratır. Hafif rekabet sinir sisteminde çok yaygındır ve genel kontrol olmaksızın kaynakların özelleştirilmesi, gösterim biçimlerinin genellenmesi ve bilgiye ulaşmayı basitleştirme gibi önemli işlevleri gerçekler. Hafif rekabet ağı yanal geri besleme kullanılarak oluşturulabilir. Ancak, bu durumda, yanal ağırlıklar işlem elemanına olan uzaklığa göre değişir. Yakın olan işlem elemanları birbirlerini güçlendirirken, uzak işlem elemanlarını bastırırlar. Her girdi için birden fazla İE aktif olduğundan, güncelleme işlem elemanlarının kazanana olan uzaklığına göre hafifletilmesine rağmen, her girdi için birden fazla İE'nin ağırlıkları güncellenir.

Yüzeysel olarak hafif rekabet ile güçlü rekabet birbirine benzemesine rağmen bu benzerlik yanıltıcıdır. Hafif rekabet, güçlü rekabetten farklı olarak işlem elemanları arasında komşuluk ilişkileri oluşturur. Başka bir deyişle işlem elemanları bir benzerlik ölçütüne göre birbirlerine bağlanırlar. Girdi uzayından işlem elemanları uzayına topolojik eşleştirme mümkündür.

Topolojik eşleştirmenin en önemli uygulamalarından birisi veri indirgemeyi gerçekleştirme ve yerel komşuluk bilgilerini korumadır. Girdi verilerinin daha küçük bir uzaya projeksiyonu daha farklı yöntemlerle de gerçekleştirilebilir (örneğin, *En Önemli Bileşen Analizi* - PCA). Fakat hafif rekabet ile gerçekleştirilen projeksiyonlar veri kümelerinin hassas detaylarını muhafaza eder. Bu yüzden, eğer iki girdinin çıktı uzayında dönüştürüldüğü noktalar birbirine yakınsa, bu iki veri girdi uzayında da birbirine yakındır.

Hafif rekabetçi öğrenmeyi ağ büyüklüğünün sabit olup olmamasına göre iki kategoriye ayrılabilir. Bu iki kategoride kullanılan *sinirsel gaz*, *rekabetçi Hebbian öğrenme*, *büyüyen sinirsel gaz*, *büyüyen hücre yapıları*, *büyüyen ızgara yapıları* ve *kendi kendini düzenleyen özellik haritaları* gibi farklı tekniklerden başlıcalarına bu bölümde değinilecektir.

4.4.1 Ağ Büyüklüğünün Sabit Olmadığı Hafif Rekabetçi Öğrenme

Ağ büyüklüğünün sabit olmadığı hafif rekabetçi öğrenme yöntemlerinin ortak özellikleri, uygulandıkları ağın sabit büyüklükte bir yapısının olmamasıdır. Bu yöntemlerden birisinde, bütünüyle hiç bir geometrik yapı yoktur (*sinirsel gaz*). Diğerlerinde ise, ağın büyüklüğü verinin yerel boyutuna bağlıdır ve girdi uzayı içinde değişebilir. Başlıcaları:

- Sinirsel Gaz (Martinetz ve Schulten, 1991)
- Rekabetçi Hebbian Öğrenme (Hertz ve diğ., 1991; Martinetz ve Schulten, 1991; Martinetz, 1993)
- Rekabetçi Hebbian Öğrenme ile Sinirsel Gaz (Martinetz ve Schulten, 1991; Martinetz, 1993)
- Büyüyen Sinirsel Gaz (Fritzke, 1994b, 1995a)

4.4.2 Ağ Büyüklüğünün Sabit Olduğu Hafif Rekabetçi Öğrenme

Sabit ağ büyüklüğünün avantajlarından birisi; böyle bir ağın n-boyutlu girdi uzayından (n istenildiği kadar büyük olabilir) k-boyutlu yapıya bir eşleştirme (dönüşüm) yapmasıdır. Bu özellik, verilerin görsel amaçlı olarak kullanılmasına olanak sağlayacak şekilde, daha düşük boyutta gösterimini olanaklı kılar. Uygulandıkları ağın büyüklüğünün sabit olduğu (k) bazı hafif rekabetçi öğrenme yöntemleri:

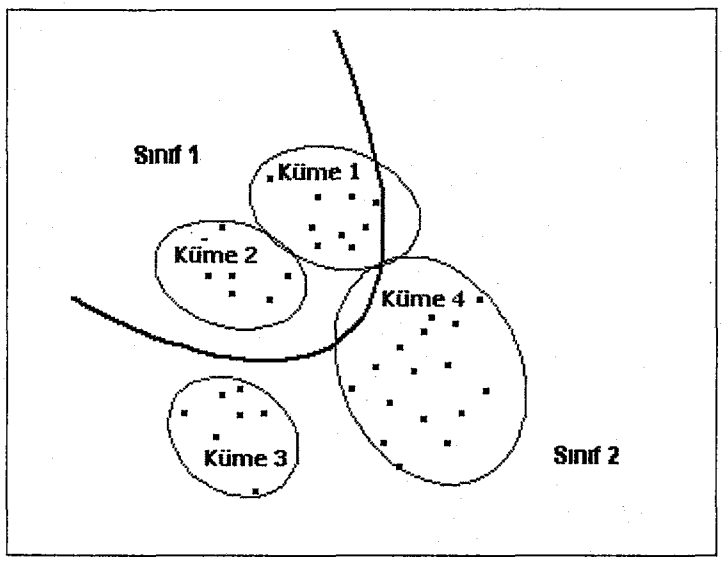
- Büyüyen Hücre Yapıları (Fritzke, 1994a)
- Büyüyen Izgara
- Kendi Kendini Düzenleyen Özellik Haritaları (Kohonen, 1995)

4.5 Kendi Kendini Düzenleyen Ağlar

Kohonen ağları (her ne kadar VN ve ÖVN ağları da bu gruba girmesine rağmen) olarak da adlandırılan *kendi kendini düzenleyen ağlar* (SOM), aslında "Ağ büyüklüğünün sabit olduğu hafif rekabetçi öğrenme" sınıfında yer alsa da, lojistik sistemlerin YSA ile modellenmesinde yoğunlukla kullanılacağından ayrı bir başlık altında ve daha detaylı olarak incelenecektir.

Eğitici veri kümesi (girdiler ve bilinen çıktıları) elde bulunduğu zaman desenleri modellemek için geriye yayılım ile eğitilen ÇKİB sinir ağı, elde bulunmadığında ise kendi kendini düzenleyen ağlar kümeleme tekniği olarak kullanılır. Kümeleme, genellikle verilerin doğal yapısına dayanarak verileri gruplandırma yaklaşımıdır. Uygun bir kümeleme algoritmasının amacı, bir kümedeki desenler ile başka bir kümedeki desenler arasındaki benzerliği minimum yapmaya çalışırken, aynı kümedeki desenler arasındaki benzerliği ise maksimum yapmaktır. Girdi uzayı bakış açısıyla ele alındığında, kümeleme; tanım uzayını her birisi bir çıktı işlem elemanı ile ilişkili olan yerel bölgelere ayırmaktır. Girdi uzayındaki noktaları gösteren işlem elemanlarının ağırlıkları prototip vektörler olarak adlandırılır. Eğer prototip vektörler bir çizgi ile birleştirilirse, dikey bisektörü, bal peteğine benzeyen bir bölüm oluşturacak şekilde diğer bisektörlerle birleşir (Voronoi çokgenleri). Bölgelerin içinde kalan veri örnekleri ilgili prototip vektörlere atanır. Bu nedenle kümeleme bir sürekli-ayrık uzay dönüşümüdür.

Kümeleme gözetimsiz bir gruplandırma süreciyken sınıflandırma gözetimlidir. Şekil 4.5'de kümeleme ve sınıflandırmaya bir örnek gösterilmiştir. Dört farklı veri kümesi ve iki farklı veri sınıfı görülmektedir. Kümeleme sadece girdi verilerinin doğal olarak gruplanmaları ile belirlendiğinden, kombinatorik olarak pek çok kümeleme alternatifi söz konusudur. Örneğin, Küme 1 ve 2 aynı kümede kabul edilebilir.



Şekil 4.5 : Kümeleme/Sınıflandırma

Kümeleme gözetimsiz olduğundan, doğrudan sınıflandırma için kullanılamaz. Ancak, pek çok pratik uygulamada, her bir sınıftaki veriler yoğunlaşır ve sınıflar arasında doğal bir ayrım oluşur. Böyle durumlarda kümeleme sınıflandırma için bir ön süreç olabilir.

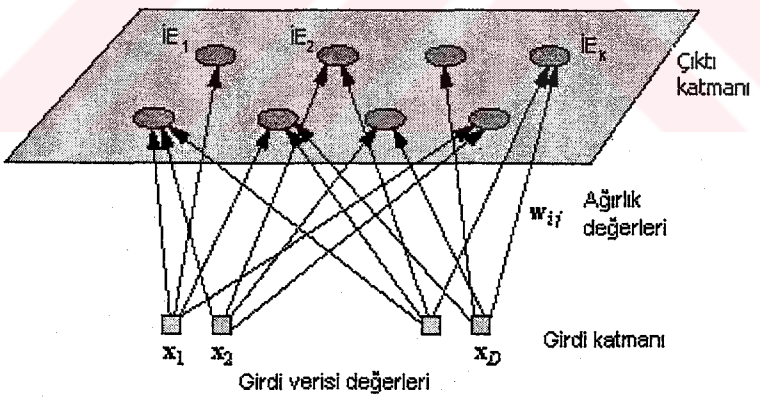
Çok boyutlu girdi uzayındaki desenler sıklıkla çok karmaşık bir yapıya sahiptirler, fakat bu yapı bir veya bir kaç boyutlu öznitelik uzayında kümelenilirlerse daha fazla saydam ve basit olur. Kohonen, büyük veri kümelerinde güçlü öznitelikleri otomatik olarak algılamak için bir yöntem olarak kendi kendini düzenleyen öznitelik haritalarını geliştirmiştir (Kohonen, 1995). SOM'lar, çok boyutlu girdi uzayını daha az boyuta sahip öznitelik uzayına dönüştüren ve böylece kümelerin indirgenmiş boyutta görülebildiği bir eşleştirme bulur.

Som'un kökeninde *rekabetçi öğrenme* vardır ve doğrusal işlem elemanları içeren tek katmanlı ağlardır. Bu tip ağlarda, her girdi deseni için sadece bir tane kazanan işlem elemanı vardır. Rekabetçi ağlarda, sadece kazanan düğüm noktasının ağırlığı güncellenir. Kohonen küçük bir değişiklik yaparak çok büyük sonuçlar elde etmiştir. Sadece kazanan işlem elemanını güncellemek yerine SOM ağında yer alan komşu İE'nin de ağırlıkları daha küçük bir adım büyüklüğü kullanılarak güncellenir (*hafif rekabet*). Bunun anlamı, öğrenme sürecinde girdi verilerinin özniteliklerine bağlı olarak uzaysal konumlarına göre komşuluk ilişkileri yaratılır. Gerçekte, girdi uzayındaki benzer veri noktalarının, Kohonen'in SOM katmanında küçük

komşuluklara dönüştürüldüğü gösterilebilir. Beynimiz pek çok bilinen topografik haritalara (görsel ve işitsel korteks) sahiptir.

Kohonen'in kendi kendini düzenleyen eşleştirme ağı, sürekli girdi uzayından ayrık bir çıktı uzayına, girdinin topolojik özelliklerini koruyarak dönüşümü gerçekleştirir. Bunun anlamı, girdi uzayında birbirine yakın olan noktalar çıktı uzayında aynı veya komşu işlem elemanları ile eşleştirilirler (Şekil 4.6).

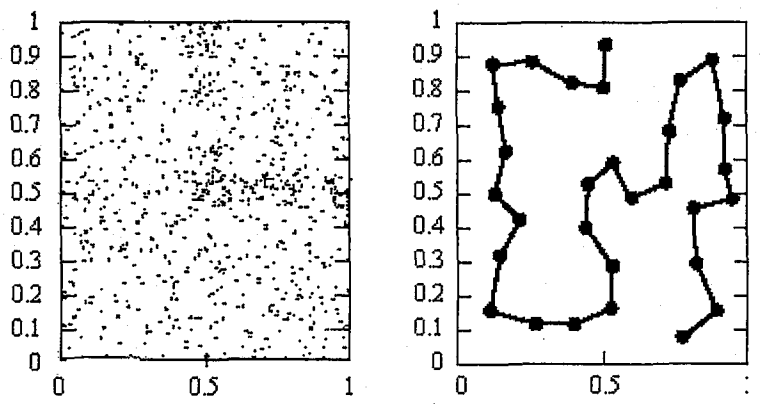
Kohonen'in orijinal SOM algoritması bir enerji fonksiyonunu eniyilemez (Erwin ve diğ., 1992; Kohonen, 1995, s. 126 & 237). SOM algoritması nicelendirmenin kesinliği ile topolojik eşleştirmenin düzgünlüğü arasında bir ödünleşmeyi gerçekler. Ancak, bu iki özelliği açık olarak birbiriyle ilişkilendiren bir enerji fonksiyonu yoktur. Bu yüzden SOM diğer gözetimsiz öğrenme ağları gibi basit bir bilgi-sıkıştırma yöntemi değildir. Kohonen'in SOM'u kolaylıkla bir yoğunluk kestireç yöntemi gibi de yorumlanamaz. Kohonen SOM'un ne öğrendiğinin en iyi açıklaması, SOM ile ana eğri ve yüzeyler arasında bağlantı kurmak suretiyle Mulier ve Cherkassky (1995) ile Ritter ve diğ. (1992) tarafından yapılmıştır. Grup SOM algoritması Hastie ve Stuetzle (1989) tarafından önerilen ana eğri ve yüzey algoritmasına çok benzer. Ana eğri aslında bir ana bileşenin doğrusal olmayan genelleştirilmiş halidir.



Şekil 4.6 : 2-Boyutlu çıktılı bir SOM'un mimarisi (Principe ve diğ., 2000, s.349)

Tek-boyutlu komşuluk İE elemanlardan oluşan bir çizgi boyunca sıraladığından, her bir elemanın kendisinden önce ve sonra gelen olmak üzere iki komşusu vardır. Tek-boyutlu SOM, her biri iki komşusuna da yakın olmak durumunda olan işlem elemanlarının oluşturduğu bir dizi gibi düşünülebilir. SOM daha büyük boyuttaki bir girdiyi uyarladığında, girdi uzayını kaplayacak şekilde kendini esnetir ve büker.

Örneğin Şekil 4.7, 2-boyutlu girdi uzayının 1-boyuta SOM ile dönüşümünü göstermektedir.



Şekil 4.7 : 2-Boyutlu girdi verileri ve 1-Boyutlu SOM dönüşümü (Principe ve diğ., 2000, s.349)

Girdi, iki boyutlu rasgele seçilmiş noktaların oluşturduğu bir kümedir. İşlem elemanları ikinci grafikte koyu noktalar olarak gösterilmiştir ve komşu İE de çizgilerle birleştirilmiştir. Çıktı İE, hala gereksinim duyulan komşuluk ilişkilerini sürdürerek girdi uzayını kaplayacak biçimde esneyip bükülen bir dizgide düzenlenirler. Diğer yandan, bir 2-boyutlu komşuluk, daha fazla esnek eşleştirme yapmayı etkileyecek fazladan komşular yaratır. İki-boyutlu uzayda 2-boyutlu komşuluğu seçersek, işlem elemanlarının veri örneklerine doğru daha hızlı bir şekilde yayıldığını görürüz.

4.5.1 Öğrenme Algoritması

Hafif rekabet düşüncesi, Kohonen Som'un işlevlerini anlamada çok önemlidir. Girdileri çıktılarla birleştiren ağırlıklar girdilerle ağırlıklar arasında işbirliğini sağlarlar. Ağırlık vektörü mevcut girdiye en yakın olan İE rekabeti kazanır.

$$\mathbf{w}_{ij}(t+1) = \mathbf{w}_{ij}(t) + \eta \mathbf{y}_i(t)\left(\mathbf{x}_j(t) - \mathbf{w}_{ij}(t)\right) \tag{4.21}$$

Kohonen SOM'unda sadece rekabeti kazanan değil, ayrıca komşuları da ağırlıklarını rekabetçi kurala göre güncellerler (Denklem 4.21). Algoritmanın hesaplamasını basitleştirmek için, yanal ketleyici ağın merkezi kazanan işlem elemanında olan bir Gauss dağılımı ürettiği varsayılır. Bu yüzden, her bir işlem elemanının aktivitesini yinelemeli olarak hesaplamak yerine kazanan işlem elemanını buluruz ve diğer İE, her birinin kazanana olan uzaklıklarına göre değerlendirilen bir Gauss fonksiyonu ile

orantılı bir aktiviteye sahip olduğu varsayılır. Her bir işlem elemanının çıktı aktivitesi ile rekabetçi kuralı dengeleyen bir öğrenme kuralı uygulandığından, Kohonen SOM rekabetçi kuralı aşağıdaki şekle dönüşür:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Lambda_{i,i^*}(t)\eta(t)\big(x(t) - w_i(t)\big) \tag{4.22}$$

Λ fonksiyonu, kazanan işlem elemanında merkezlenmiş bir komşuluk fonksiyonudur. Komşuluk ve adım büyüklüğünün her ikisi de yineleme sayısı ile değişir (bu yüzden adım adım azaltılabilir). Komşuluk fonksiyonu Λ normalde Gauss dağılımıdır:

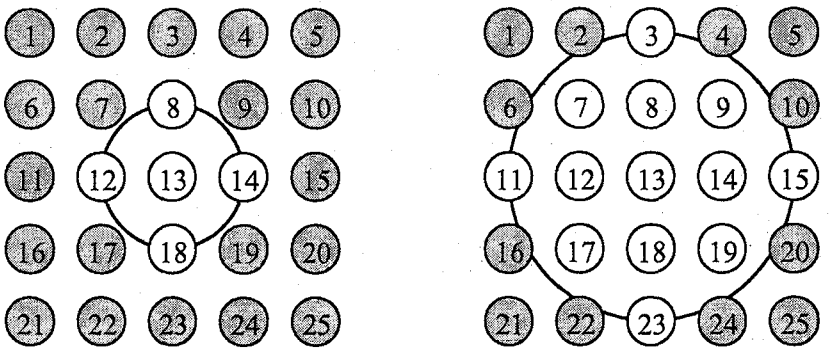
$$\Lambda_{i,i^*}(t) = \exp\left(\frac{-d_{i,i^*}^2}{2\sigma^2(t)}\right) \tag{4.23}$$

ve varyans yinelemeye bağlı olarak azalır. Hemen hemen bütün haritayı kaplayacak şekilde başlar ve ardından ilerleyerek sıfır komşuluğa kadar azalır, yani sadece kazanan işlem elemanı güncellenir. Kazanan işlem elemanı için uyarlama kuralı:

$$w_{i^*}(n+1) = w_{i^*}(n) + \eta\big(x_j(n) - w_{i^*}(n)\big) \tag{4.24}$$

Fakat kazanan işlem elemanının komşuları için, bu işlem elemanına olan uzaklığa göre üssel olarak azaltılır. Λ'nın etkin alanının dışında kalan diğer bütün ağırlıklar değiştirilmez. Komşuluğun derecesi azaldıkça ağ, "çok hafif" rekabetten (aşağı yukarı bütün işlem elemanları bir şekilde aktif) güçlü rekabete (sadece kazanan işlem elemanı aktif) geçer.

Uzaysal komşuluk, Kohonen haritalarını diğer rekabetçi sinir ağlarından çok farklı yapar. Uzaysal komşuluk, girdi uzayından SOM çıktısına olan yapıyı gösterir. $N_i(d)$ komşuluğu, i nöronuna en fazla d yarıçaplık bir mesafede olan bütün nöronları içerir (Şekil 4.8). Som'un girdi uzayındaki komşulukların korunduğu topolojik ilişkilerin yer aldığı bir ayrık çıktı uzayı oluşturduğunun kanıtları vardır. Bunun anlamı, girdi uzayındaki veri örneklerinin dağılımının yaklaşık olarak korunmasıdır. Bu ise SOM'u olasılık yoğunluk fonksiyonlarını tahmin etmede faydalı yapar. Üstelik girdi uzayında birbirine yakın olan noktaların eşleştirildikleri işlem elemanları da komşudurlar. Bu özellik, bir girdi uzayı metriğinin korunmasına gereksinim duyan bazı uygulamalar için çok ilginçtir. Bütün bu özellikler sadece girdideki bilgiler kullanılarak elde edilir ve SOM gözetimsiz olarak oluşturulur.



Şekil 4.8 : SOM'da uzaysal komşuluk

Uzaysal komşuluğun bu topolojik dönüşümü nasıl yarattığını anlamak için, kullanılan rekabetçi kuralın ağırlık vektörlerini mevcut girdiye doğru götürdüğünü hatırlamak gerekir (Şekil 4.1). Kazanan işlem elemanı ve komşuları her adımda güncellendiğinden, komşuların kazanan işlem elemanına olan uzaklıkları arttıkça daha yavaş hareket etmeleri söz konusu olmasına rağmen, kazanan ve onun bütün komşuları aynı pozisyona doğru harekete ederler. Zaman içerisinde bu süreç işlem elemanlarını, başlangıç konumlarına bağlı olmaksızın komşu işlem elemanlarını (SOM'un çıktı uzayındaki) girdi uzayında aynı alanın gösterimini paylaşacak şekilde örgütlerler.

Parametrelerin seçimi, girdi uzayından ayrık çıktılara topolojiyi koruyan bir dönüşümü gerçekleştirmede çok önemlidir. Geçmiş deneyimlerden SOM öğrenmede iki evre olduğu bulunmuştur. İlk evre, ağırlıkların topolojik sıralanması yani komşulukların tanımlanması ile ilgilenir. Bu evrenin N_0 yinelemede gerçekleştiğini varsayalım. Bu evrede, komşuluk fonksiyonu, İE'nin benzer girdileri bir araya getirmesine izin vermesini sağlamak amacıyla bütün çıktı uzayını kapsayacak şekilde başlar. Fakat komşuluk tek bir işlem elemanına veya bir işlem elemanı ve onun komşularına kadar azalmaya gereksinim duyar. Normalde aşağıdaki gibi belirtilen komşuluk yarıçapındaki doğrusal azalma;

$$\sigma(t) = \sigma_0(1 - t/N_0) \tag{4.25}$$

kullanılır. Bu periyot süresince öğrenme oranı ağın kendi kendini organize etmesini sağlayacak büyüklükte (0.1'in üzerinde) olmalıdır. η'nün çizelgesi de normalde doğrusaldır:

$$\Delta\eta(t) = \eta_0\left(1 - t/(N+K)\right) \tag{4.26}$$

η_0 öğrenme oranının başlangıç değeridir ve K son öğrenme oranını belirlemeye yardım eder.

Öğrenmenin ikinci evresi yakınsama fazı olarak adlandırılır ve genellikle daha uzundur. Bu evrede İE'nin girdi dağılımının detaylarına göre ince ayarı yapılır ve en küçük komşuluk (sadece işlem elemanı veya onun en yakın komşuları) kullanılırken öğrenme oranı küçük (0.01) tutulmalıdır. İşlem elemanlarının kaç adet olacağı deneysel olarak belirlenir. Çıktı işlem elemanının sayısı dönüşümün doğruluğunu ve eğitme zamanını etkiler. İE sayısını artırmak haritanın çözünürlüğünü artırır fakat eğitme zamanını da çarpıcı bir biçimde artırır.

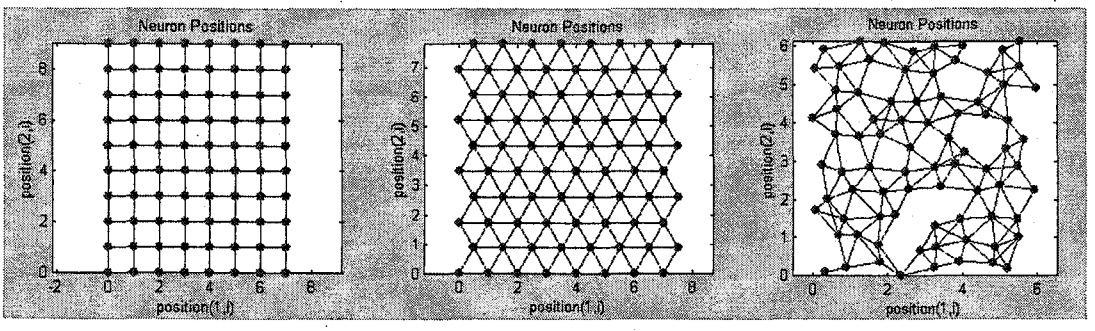
4.5.2 Kohonen Haritalarının Özellikleri

4.5.2.1 Girdi Uzayının Tahmin Edilmesi

SOM girdi uzayının yapısını göreceli olarak iyi sürdürecek özelliğe sahiptir. Girdi uzayının metriğinin kaybedilmesi ve sadece dağılım kümelerinin konumunun modellenmesi rekabetçi ağların eksikliklerinden birisidir. Veri azaltmada bu uygulama çok önemlidir ve vektör nicelendirmenin hedefidir.

4.5.2.2 Topolojik Sıralama

SOM'daki işlem elemanları, girdi uzayında benzer bölgelere karşılık gelen komşu işlem elemanları olacak şekilde topolojik olarak sıralanırlar (Şekil 4.9).



(a) (b) (c)

Şekil 4.9 : (a) Grid, (b) hexonal ve (c) rastsal topolojiler

4.5.2.3 Yoğunluk Eşleştirme

SOM girdi uzayındaki veri yoğunluğunu orta iyilikte korur, yani, daha fazla veri içeren bölgeler çıktı uzayında daha büyük alanlara dönüştürülürler.

4.6 Vektör Nicelendirme Algoritması Olarak Kohonen Ağları

Gözetimsiz yoğunluk kestireçleri veya kendi kendine-ilişkilendiriceler gibi görülebilen (Kohonen, 1995-1997; Hecht-Nielsen 1990), k-ortalama kümeleme analizleri ile yakından ilişkili rekabetçi ağlardır (MacQueen, 1967; Anderberg, 1973). Her bir rekabetçi birim, bir kümenin referans vektörü olarak isimlendirilen merkezine karşılık gelir. Kohonen'in öğrenme kuralı, her bir eğitim durumuna en yakın referans vektörünü (*codebook*) bulan ve kazanan referans vektörünü eğitim durumuna yaklaştıran bir eşzamanlı algoritmadır. Referans vektörü kendisi ile eğitim durumu arasındaki mesafenin belirli bir oranında hareket ettirilir. Bu oran öğrenme oranı ile belirlenir ve aşağıdaki gibi hesaplanır:

$$\text{yeni\_codebook} = \text{eski\_codebook} * (1\text{-öğrenme\_oranı}) + \text{veri} * \text{öğrenme\_oranı} \quad (4.27)$$

Literatürde benzer pek çok algoritma geliştirilmiştir ve bunların kısa bir özeti Hecht-Nielsen (1990)'de görülebilir. Aslında MacQueen'in eşzamanlı k-ortalama algoritması, öğrenme oranının kazanan kümeye atanan durumların sayısına bölünmesi hariç, Kohonen'in öğrenme kuralı ile aynıdır. Verilen eğitim durumlarının işlemden geçirildiğini varsayarsak, N durumun kazanan referans vektörüne daha önceden atandığını düşünelim. Referans vektörü aşağıdaki gibi güncellenir:

$$\text{yeni\_codebook} = \text{eski\_codebook} * N/(N+1) + \text{veri} * 1/(N+1) \quad (4.28)$$

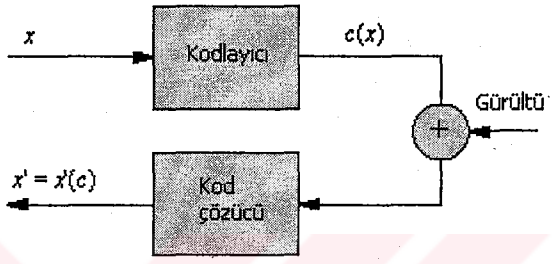
Öğrenme oranındaki bu indirgeme, her bir referans vektörünü kendi kümesine atanan bütün durumların ortalaması yapar ve hata fonksiyonunun bir en iyi minimum değerine yakınsamasını eğitim durumu sayısı sonsuza yaklaştıkça garanti altına alır.

Vektör nicelendirmeyi gerçekleştiren LBG algoritması gibi iyi tasarlanmış algoritmalar vardır (Bkz., Bölüm 4.2.1). LBG algoritmasının, girdi örneği $\mathbf{x}$ ile iletilmiş

x' vektörü (sınıf merkezi) arasındaki ortalama çarpıklık ölçümü açısından en iyi olduğu gösterilmiştir.

$$D = \frac{1}{2}\int_{-\infty}^{\infty} f(x)|x - x'|^2 dx \tag{4.29}$$

Yukarıdaki eşitlikte $f(x)$ girdinin olasılık dağılım fonksiyonudur. Teoride, ilk olarak girdinin kodlanarak $c(\mathbf{x})$'in oluşturulduğu ve gürültü eklenip kod çözülerek sonuçta x' in elde edildiği bir dahili kodlama-çözme adımı varsayılır (Şekil 4.10)



Şekil 4.10 : Vektör nicelendirme modeli (Principe ve diğ., 2000, s.353)

İlginç olan nokta, en iyiye ulaşmak için iki koşula gereksinim duyulmasıdır:

- Girdi örneği x için kod c(x), x ile yeniden üretme vektörü x' arasındaki çarpıklığı minimum yapacak şekilde seçilmelidir.
- Kod c verildiğinde yeniden üretme vektörü x', c(x) olarak kodlanan girdi vektörünün merkezi olmalıdır.

LBG algoritması, Kohonen algoritmasının komşuluk derecesi sıfıra eşit olan bir zamandan bağımsız versiyonudur. Bu yüzden Kohonen SOM en iyi vektör niceleme algoritması olarak yorumlanabilir.

Kohonen VN, genellikle eğitim verilerinin saklandığı ve Kohonen öğrenme kuralının her bir duruma sırayla uygulandığı zamandan bağımsız öğrenme için veri kümesi üzerinden pek çok kez geçilerek (*marjinal öğrenme*) kullanılır. Eğer öğrenme oranı uygun bir şekilde azaltılırsa, eğitim süresi sonsuza giderken yerel bir minimuma yakınsama elde edilebilir. Ancak, sonlu sayıda yinelemeden sonra yakınsayan hem grup hem de marjinal zamandan bağımsız k-ortalama algoritmaları vardır (Anderberg, 1973; Hartigan, 1975; Hartigan and Wong, 1979). Marjinal yöntemler ya her bir durumun kümeye üyeliğinin veya iki en yakın küme arasındaki hesaplamaların saklanmasına gereksinim duyar. Bu nedenle, grup algoritmaları

büyük veri kümeleri için avantajlıdır (Forgy, 1965; Anderberg, 1973). Forgy'nin yaklaşımı birbirini takip eden basit en küçük kareler algoritmasıdır ve işlem adımları aşağıdaki gibidir:

- Referans vektörlerini başlangıç durumuna getir.
- Yakınsama gerçekleşinceye kadar aşağıdaki iki adımı tekrarla:
  1. Verileri oku, her bir durumu en yakındaki (Öklit mesafe fonksiyonuna göre) referans vektöre ata.
  2. Her bir referans vektörünü, ona atanan durumların ortalaması ile değiştir.

En hızlı eğitme genellikle ilk geçişte MacQueen'in eşzamanlı algoritması kullanılır ve sonraki geçişlerde zamandan bağımsız k-ortalama algoritmaları uygulanırsa elde edilmektedir (Bottou and Bengio, 1995). Ancak, bu eğitme yöntemleri hata işlevinin bir genel en iyiye mutlaka yakınsamayabilir. Bir genel en iyiyi bulma şansı, rasyonel başlangıç durumları, çoklu rastsal başlangıç durumları veya genel eniyileme için geliştirilmiş ancak vakit alan eğitme yöntemleri kullanılarak artırılabilir (Ismail and Kamel, 1989; Zeger ve diğ., 1992).

Kohonen'in çalışmaları VN'yi yoğunluk kestireci ve bu nedenle eşit derecede olası kümelerin çekiciliği olarak ön plana çıkarmıştır (Kohonen 1984; Hecht-Nielsen 1990). Ancak, Kohonen'in öğrenme kuralı eşit derecede olası kümeler üretmez (yani, her bir kümeye atanan eğitici durumların oranı genellikle eşit değildir). Eğer N girdi varsa ve kümelerin sayısı fazlaysa, referans vektörlerinin yoğunluğu eğitme kümesinin yoğunluğunun $N/(N+2)$'nci kuvvetine yakınsar (Kohonen, 1995, s. 35; Ripley, 1996, s. 202; Zador, 1982). Bu nedenle kümeler, sadece eğer verilerin yoğunluğu birörnek veya girdi sayısı büyükse yaklaşık olarak eşit derecede olasıdır.

4.7 Öğrenen Vektör Nicelendirme

Öğrenen Vektör Nicelendirme (ÖVN) (*Learning Vector Quantization - LVQ*), gözetimli sınıflandırma için kullanılan bir rekabetçi sinir ağı çeşididir (Kohonen, 1988, 1995; Ripley, 1996). Her bir referans vektörü hedef sınıflardan birisine atanır. Her sınıf bir veya daha fazla referans vektörü içerebilir. Bir veri, en yakın referans vektörü bulunup ait olduğu sınıfa atanmak suretiyle sınıflandırılır. Bu yüzden, ÖVN bir çeşit en-yakın komşuluk kuralını kullanır.

Kohonen'in VN veya k-ortalamalar gibi sıradan VN yöntemleri gözetimli sınıflandırma için kolaylıkla kullanılabilir. Her bir sınıftan her bir kümeye atanan eğitme durumu sayısı bulunup kümedeki toplam durun sayısına bölünerek son olasılıklar elde edilir. Verilen bir durum için, en büyük son olasılığa sahip sınıf (yani, en yakın kümedeki çoğunluğu oluşturan sınıf) çıktı olur. Bu tip yöntemler, referans vektörleri gözetimsiz algoritmalarla elde edilse dahi tutarlı genel sınıflandırıcıdırlar (Devroye ve diğ., 1996). ÖVN, bu yaklaşımı, referans vektörünü gözetimli olarak uyarlamak yoluyla iyileştirmeye çalışır. ÖVN'nin sezgiselliğe dayanan birkaç farklı türü vardır: ÖVN1, ÖVN2 ve ÖVN3. Ancak, ÖVN normalde ileri beslemeli bir sinir ağı kullanılarak ve herhangi bir hata işlevi eniyilenerek eğitilebilir

Kohonen ağları, özellikle Gezgin Satıcı Problemi ile Öklit düzlemine gömülen eniyileme problemlerini çözmede kullanılabilir. Ancak, daha genel ve pratik eniyileme problemlerini çözebilmek için elastik bant yöntemine dayanmayan, permütasyon matrisinin satırlarının en küçük maliyete göre sıralandığı kendi kendini düzenleyen sinir ağı mimarisi kullanılabilir (Smith ve diğ., 1996). Bu ağların başarısı ise sadece eldeki problemin uygun biçimde formüle edilmesine bağlıdır.

5. HOPFIELD-TANK SİNİR AĞLARI

Kombinatorik eniyileme problemlerini çözmede kullanılan bir diğer yaklaşım; Hopfield-Tank sinir ağlarıdır (Smith ve diğ., 1996a). Ancak, bu yaklaşımın da bazı kusurları vardır ve farklı problem tiplerinin çözümünü bulmada gösterdikleri başarı değişkendir. Hopfield ağları genellikle olursuz çözüm üretme ve en iyi çözümü elde etmek için sıkıcı parametre ayarlamaları gerektirme gibi dezavantajlara sahip olmalarına rağmen, olurluluğu sağlayıcı ve çözüm kalitesini artırıcı düzenlemelerle özellikle ikinci dereceden programlamada oldukça iyi sonuçlar elde edilmiştir (Smith ve diğ., 2000). Bu bölümde, *yinelenen* (recurrent), geri beslemeli yapay sinir ağları ve bu alanda en çok bilinen Hopfield-Tank mimarisi açıklanacaktır.

5.1 Yinelenen Yapay Sinir Ağları

Yinelenen yapay sinir ağları, eşik İE'ler kullanan ve saklı işlem elemanı olmayan, çıktılarından girdilerine geri besleme bağlantısı bulunan, simetrik ara bağlantı matrisine[1] sahip modellerdir (Hagan ve diğ., 1996; Principe ve diğ., 2000). Yinelenen ağlar, uzaysal örüntüler kadar geçici örüntüleri de tanıma ve hatırlama yeteneklerinden dolayı, ileri beslemeli ağlardan potansiyel olarak daha güçlüdürler. Ancak, bu mimariye sahip ağların davranış biçimleri ileri beslemeli olanlara göre çok daha karmaşıktır.

İleri beslemeli ağlarda, değişmeyen bir girdi için çıktı sabittir ve sadece girdi değerlerinin bir işlevidir. Yinelenen ağlarda ise, ağın çıktısı zamanın bir fonksiyonudur. Belirli bir girdi değeri ve ağın başlangıç çıktı değerleri için, ağdan elde edilen çıktılar kararlı bir sonuca yakınsayabilir. Bununla beraber çıktı değerleri salınım gösterebilir, sonsuza gidebilir veya kaotik bir örüntü izleyebilir. Kararlı durumda bu sistemlerin nasıl performans göstereceği önemlidir ve genellikle sinir ağının sabit bir çıktı değerine, yani kararlı bir denge noktasına, yakınsadığı

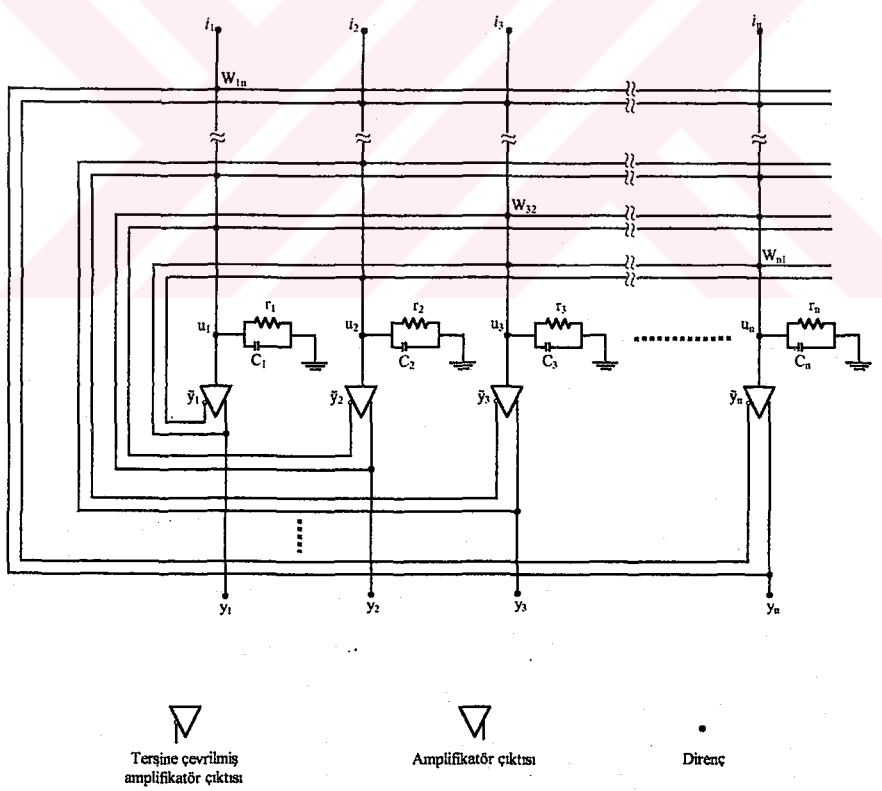
---

[1] Vidyasagar (1993), bazı asimetrik koşullar altında yakınsamanın hala geçerli olduğunu göstermiştir.

durumlarla ilgilenilir. Doğrusal olmayan bir sistemde pek çok kararlı nokta olabilir. Bazı yapay sinir ağlarında bu kararlı noktalar daha önceden saklanmış ilk örnek örüntüleri gösterir. Eğer mümkünse, kararlı noktaların nerede olduğu ve hangi başlangıç koşullarının verilen bir kararlı noktaya yakınsadığı bilinmek istenir.

5.2 Hopfield Sinir Ağları

Hopfield 1982 ve 1984 yıllarında çok etkileyici iki makale yazdı (Hopfield, 1982; Hopfield, 1984). Bu makalelerdeki pek çok düşünce, daha önceki araştırmacıların çalışmalarına dayanmaktaydı. Ancak, Hopfield bu makalelerinde, çok önemli pek çok kavramı bir araya getirmekte ve açık, anlaşılır matematiksel analizlerle sunmaktaydı. John Hopfield nöronlardan oluşan bir sistemi sayısal işlemleri gerçekleştirecek biçimde modellemede yeni bir yol önermişti (Şekil 5.1). Hopfield, belirli koşullar altında sisteme bir girdi verildiğinde sinir ağının ne yaptığını açıklayabilmekteydi. Hopfield sinir ağı başlangıçta bir *içerik-adreslenebilir hafıza* (content-addressable memory - CAM) olarak ortaya çıktı.

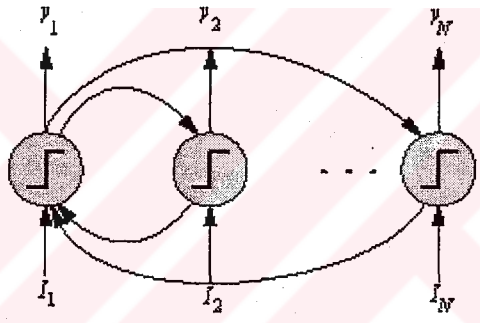


Şekil 5.1 : Hopfield sinir ağının elektronik devre olarak gösterimi (Hopfield, 1984)

Hopfield ağında girdi verisi, sistemi durum uzayında bir noktaya yerleştirir ve yinelemeli bağlantılarla oluşturulan ağ dinamikleri, sistemin en yakın denge noktasına erişmesini sağlar. Eğer denge noktaları önceden seçilmişse, sistem bir

ilişkilendirici hafıza gibi çalışabilir. Son durum, durum uzayında verilen girdi verisine en yakın durum olacaktır. Daha sonra girdi verisi sınıflandırılabilir veya içerik adresleme özelliği kullanılarak yeniden çağırılabilir. Aslında böyle bir sistem gürültüye karşı oldukça gürbüzdür ve örüntü tamamlama özelliği de gösterir. Hopfield ağında hafıza ağırlıklar kümesinde yer alır. Bu yüzden, Hopfield ağı insan beyninin bazı özelliklerine sahip olduğu bilinen bir doğrusal olmayan ilişkilendirici hafızayı uygular. Hopfield modelinin, sadece yeteri büyüklükteki herhangi bir parçasından tüm hafızayı doğru olarak ortaya koyabilme kapasitesi değil, aynı zamanda genelleştirme, sınıflandırma, hata düzeltme, benzerliği teşhis etme yetenekleri de vardır.

Ayrık Hopfield ağı, eşik İE'ler (0 veya 1 değeri alan) kullanan ve saklı işlem elemanı olmayan, n İE'nin birbirine tamamen bağlı olduğu özel bir yinelemeli sistemdir. Orijinal Hopfield mimarisinde kendi üzerine dönme bağlantısı yoktur (Şekil 5.2).



Şekil 5.2 : Hopfield sinir ağı

Sistemin zamana bağlı olarak iyileşmesi için kullanılan algoritma senkronize olmayan paralel işlemeye dayanır. Her bir i nöronu için sabit bir eşik değeri (U_i) vardır. Her bir nöron rasgele ve asenkronize olarak güncellenir.

Hopfield ağını ayrık zamanda aşağıdaki gibi tanımlayabiliriz:

$$y_i(n+1) = \text{sgn}\left(\sum_{j=1}^{N} w_{ij} y_j(n+1) - b_i + I_i(n)\right) \quad i = 1,...,N \qquad (5.1)$$

sgn doğrusal olmayan eşiği (-1,1), b ise eğilimi gösterir. Güncelleştirmelerin İE numarasına göre sırayla yapıldığı varsayılır. Girdi, $\mathbf{I} = [I_1, I_2, ..., I_N]^T$, ikili örüntüdedir ve bir başlangıç koşulu gibi çalışır; ağa verilir ve ardından ağın serbest kalmasını

sağlamak amacıyla çıkarılır. İşaret fonksiyonundan dolayı serbest bırakma esnasında ziyaret edilen noktalar N boyutlu bir hiperküpün köşe noktalarıdır. Basit olması açısından eğilim değeri (b) sıfır olarak alınabilir.

1984 yılında Hopfield tarafından modele daha fazla gerçeklilik katıldı. Hopfield ağı, orijinal modelin bütün önemli özelliklerini koruyacak şekilde sürekli değişkenleri ve yanıtlarının da yer aldığı biçime dönüştürüldü (Hopfield, 1984). Her bir nöron için harici bir girdi (I_i) yine bu modelde de yer almaktaydı. Zaman gecikmeleri modele dahil edilmişti ve sistem durumunun zamana göre evrilmesi bir diferansiyel denkleme göre gerçekleştirilmekteydi. Bu model, Hopfield'in orijinal makalesinde elektronik devre elemanları olarak tanımlanmış olsa da, aktivasyon potansiyelleri ile artırıcı ve bastırıcı sinapsların da benzer yolla hesaplanabileceği gösterilebilir (Hopfield, 1984). Genel anlamda sürekli Hopfield mimarisi, sürekli zamanda işleyen bir sinir ağı örneği olup, geri besleme yinelemesini kullanan ve ara bağlantı ağırlıklarındaki değişimlerin İE'nin aktivasyon değerlerindeki değişime kıyasla ihmal edilebilir düzeyde olduğundan sabit kabul edildiği ağlardır (Tagliarini ve diğ., 1991; Wacholder, 1989). Sürekli Hopfield ağında bir nöronun içsel durumu ile çıktı düzeyi arasındaki ilişki, 0 ile 1 arasında sınırlandırılmış bir aktivasyon fonksiyonu (g_i) kullanılarak belirlenir. Genellikle bu aktivasyon aşağıdaki gibi tanımlanır:

$$y_i = g_i(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{\lambda}\right)\right) \tag{5.2}$$

Bu ifadede λ parametresi aktivasyon fonksiyonunun artışını (eğimini) kontrol etmede kullanılır. Sürekli Hopfield sinir ağı modelinde bir İE'nin davranışı, aşağıdaki diferansiyel denklem ile ifade edilen ve sürekli bir değişken olan aktivasyon düzeyi (u_i) ile karakterize edilir:

$$\frac{du_i}{dt} = -\frac{u_i}{\eta_i} + \sum_{j=1}^{n} W_{ij} g_j(u_j) + I_i \tag{5.3}$$

Öyle ki;

η_i : i'nci nöronun öğrenme oranı

W_{ij} : i'nci nöron ile j'nci nöron arasındaki bağlantının ağırlığı

I_i : i'nci nörona harici girdi değeri

$g_j(u_j)$: j'nci nöronun aktivasyon fonksiyonu

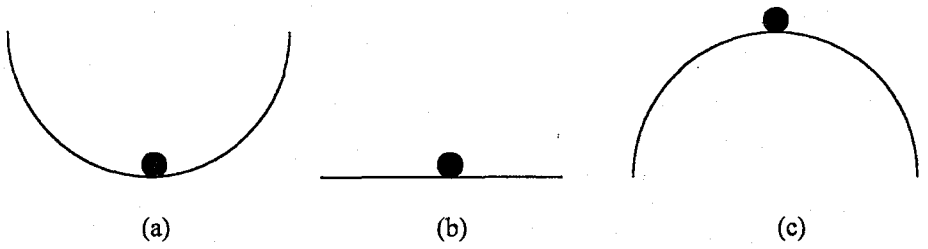
Öğrenme oranı (η_i), i'nci nörona dışardan ve diğer nöronlardan tartılandırılmış girdiler olmadığı takdirde sıfıra doğru azalma oranını verir. i'nci nöronun çıktısı, [0,1] arasında bir değer alan y_i durum değişkeni ile açıklanır ve aktivasyon düzeyi ile $y_i=g_i(u_i)$ şeklinde ilişkilendirilir. Eğer harici girdiler değişmezse, ağ başlangıç durumundan bağımsız olarak er ya da geç dengelenir (kararlı durum).

Hopfield ağının önemi, Hopfield tarafından yapılan işlevlerine dönük esin kaynağı yorumlardan kaynaklanmaktadır. Birbirine olan bağlantıların kapsamlı olmasından dolayı, girdi uygulandığında ağın ne yaptığını anlamaya çalışmak umutsuz bir çaba gibi gözükebilir. Her bir İE için dinamik eşitlikler, Denklem 5.3 gibi yazılabilir, fakat bu farklı denklemler pek çok çift oluşturduklarından ve doğrusal olmadıklarından, çözümleri ulaşılmaz gözükebilir. Ancak, pratikte hiç de öyle olmadığı gerçekleştirilen uygulamalarda görülmüştür.

5.3 Kararlılık Kavramları

Temel kararlılık kavramlarını basit, sezgisel bir örnekle açıklamak için, *sürtünmenin* ve *yerçekiminin* olduğu bir ortamda bir bilyenin hareketini ele alalım. Eğer bilye Şekil 5.3 (a)'daki bir ortamda farklı konumlardan bırakılırsa, ileri geri salınım hareketi yapacaktır, ancak, sürtünmeden dolayı sonuçta en alt noktada duracaktır. Bilyenin duracağı bu nokta *asimptotik olarak kararlı* nokta olarak adlandırılır.

Eğer bilye düz bir yüzey üzerinde merkezi olarak konumlandırılırsa (Şekil 5.3 (b)), hangi konuma konursa konsun hareket etmeyecektir. Bilye merkezden öteye doğru hareket ettirilirse tekrar merkeze gelmeyeceğinden, yüzeyin merkezindeki konum *asimptotik olarak kararlı* nokta değildir. Ancak, bilye merkez konumundan çok fazla uzağa gidemeyeceğinden bu nokta bir anlamda kararlıdır. Bu tip konumlar *Liapunov anlamında kararlı* olarak adlandırılır (Hagan ve diğ., 1996).



Şekil 5.3 : Bir bilyenin kararlılık durumları

Eğer bilye, Şekil 5.3 (c)'de olduğu gibi, bir tepenin en üst noktasında konumlandırılırsa, bu nokta denge noktasıdır. Çünkü eğer bilye dikkatlice tepe noktasına konursa o noktada sabit kalacaktır. Ancak, bilyeye küçük bir hareket verilirse tepeden aşağıya yuvarlanacaktır. Burası *kararlı olmayan* denge noktasıdır.

Tanım 5.1 Kararlılık (Liapunov): *Eğer verilen herhangi bir* $\varepsilon > 0$ *değeri için,* $||\mathbf{y}(0)|| < \delta$ *olacak şekilde bir* $\delta(\varepsilon) > 0$ *sayısı var ve sonuçta oluşan hareket* $\mathbf{y}(t)$, $t > 0$ *için* $||\mathbf{y}(t)|| < \varepsilon$*'i sağlıyorsa merkez kararlı bir denge noktasıdır.*

Bu tanım, eğer ilk konum kararlı noktaya yakınsa, sistemin çıktısının kararlı noktadan çok uzağa gitmeyeceğini ifade eder. Şekil 5.3 (b)'deki topun bulunduğu konum, yüzey sürtünmeli olduğu sürece Liapunov anlamında kararlıdır. Eğer yüzey sürtünmesiz ise, herhangi bir başlangıç hızı verildiğinde sonsuza giden yatay bir hareket, $\mathbf{y}(t)$, oluşur.

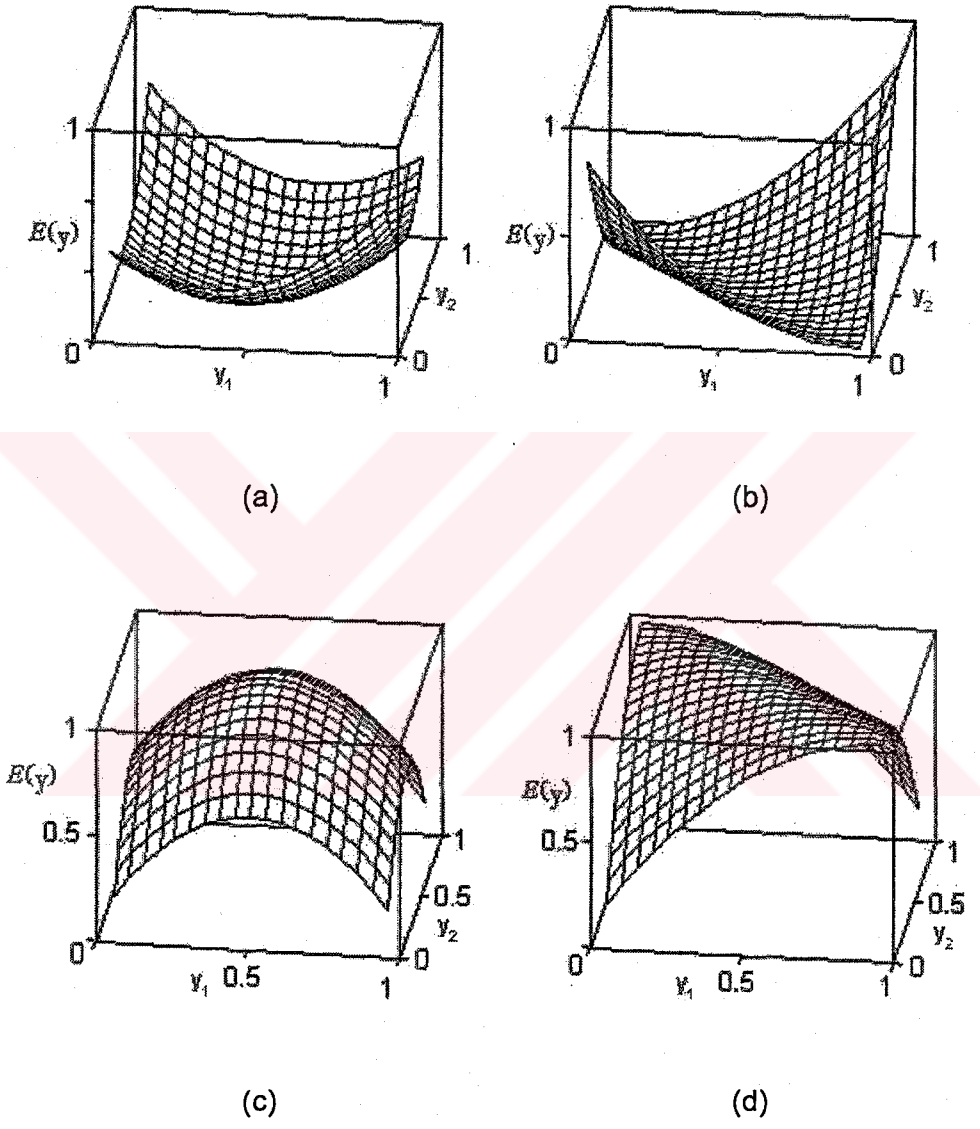
Tanım 5.2 Asimptotik Olarak Kararlılık: *Eğer* $\delta > 0$ *olacak şekilde bir* δ *sayısı varsa,* $||\mathbf{y}(0)|| < \delta$ *olduğunda sonuçta oluşan hareket,* $t \to \infty$ *için* $||\mathbf{y}(t)|| \to 0$ *oluyorsa merkez asimptotik olarak kararlı bir denge noktasıdır.*

Bu tanım kararlılığın daha güçlü bir ifadesidir. Sistemin çıktısı, başlangıçta kararlı noktanın δ kadar uzağındaysa, çıktı neticede kararlı noktaya yakınsayacaktır. Şekil 5.3 (a)'daki topun bulunduğu konum (sıfır hızına sahip), yüzey sürtünmeli olduğu sürece asimptotik olarak kararlıdır. Eğer sürtünme yoksa konum Liapunov anlamında kararlıdır. Hopfield sinir ağlarının tasarımında, her birisi ilk örnek örüntüleri gösteren birçok asimptotik olarak kararlı noktalar açık olarak belirtilir.

Kararlılık tanımlarının yanı sıra, kararlılığın analizinde kullanılan başka bir kavram ise kesin (*definitive*) fonksiyon kavramıdır.

Tanım 5.3 Pozitif Tanımlı: *Bir skalar fonksiyon* ($E(\mathbf{y})$), *eğer* $E(\mathbf{y}) = 0$ ve $\mathbf{y} \neq \mathbf{0}$ *için* $E(\mathbf{y})>0$ *ise pozitif tanımlıdır (Şekil 5.4 (a)).*

Tanım 5.4 Pozitif Kısmi-Tanımlı: *Bir skalar fonksiyon* ($E(\mathbf{y})$), *eğer bütün* $\mathbf{y}$*'ler için* $E(\mathbf{y})\geq 0$ *ise pozitif kısmi-tanımlıdır (Şekil 5.4 (b)).*



Şekil 5.4 : Enerji fonksiyonları: (a) Pozitif tanımlı, (b) Pozitif kısmi-tanımlı, (c) Negatif tanımlı, (d) Negatif kısmi-tanımlı (Hagan ve diğ., 1996)

Teorem 5.5 Aşağıdaki şekilde ifade edilen özerk (kuvvet uygulanmayan, harici olarak zamana bağımlı olmayan) bir sistem verilsin:

$$\frac{d\mathbf{y}}{dt} = g(\mathbf{y}) \tag{5.4}$$

Liapunov kararlılık teoremi şu şekilde tanımlanabilir:

Teorem 5.5 *Eğer pozitif tanımlı bir fonksiyon $E(\mathbf{y})$, $dE(\mathbf{y})/dt$ negatif kısmi-tanımlı olacak şekilde bulunabiliyorsa, Denklem 5.1'de verilen sistem için merkez ($\mathbf{y} = 0$) kararlıdır. Eğer, $dE(\mathbf{y})/dt$ negatif tanımlı olacak şekilde pozitif tanımlı bir fonksiyon $E(\mathbf{y})$ bulunabiliyorsa, o zaman merkez ($\mathbf{y} = 0$) asimptotik olarak kararlıdır. Her iki durumda da E sistemin bir Liapunov fonksiyonu olarak adlandırılır.*

$E(\mathbf{y})$ genelleştirilmiş bir enerji fonksiyonu olarak düşünülebilir. Eğer bir sistemin enerjisi sürekli azalıyorsa ($dE(\mathbf{y})/dt$ negatif tanımlı), eninde sonunda bir minimum enerji seviyesinde sabitlenecektir. Liapunov'un yaklaşımı enerji kavramını genelleştirmektir. Böylelikle bu teorem, enerjiyi ifade etmenin zor olduğu veya herhangi bir anlam taşımadığı sistemlere de uygulanabilir.

Tanım 5.6 Liapunov Fonksiyonu: *$\Re^n$'den $\Re$'ye sürekli türevi alınabilen bir fonksiyon E olsun. Eğer G, $\Re^n$'nin herhangi bir alt kümesi ise, $d\mathbf{y}/dt = g(\mathbf{y})$ sistemi için E, eğer;*

$$\frac{dE(\mathbf{y})}{dt} = (\nabla E(\mathbf{y}))^T g(\mathbf{y}) \tag{5.5}$$

G'de işaret değiştirmez ise, G'de bir Liapunov fonksiyonudur.

Bu tanım, bir önceki Liapunov fonksiyonu tanımının daha genelleştirilmiş halidir. Bu tanımda fonksiyonun pozitif tanımlı olmasına gerek yoktur. Koşul olarak sadece E'nin türevinin G'de işaret değiştirmemesi yer almaktadır. Türev, eğer *negatif* veya *pozitif kısmi-tanımlı ise* işaret değiştirmez.

Tanım 5.7 Z Kümesi:

$$Z = \{\mathbf{y} : dE(\mathbf{y})/dt = 0, \mathbf{y}\ G'de\ yer\ almakta\ ise\} \tag{5.6}$$

Tanım 5.8 Değişmez Küme: *Eğer bu kümeden başlayan $d\mathbf{y}/dt=g(\mathbf{y})$'nın her çözümü yine bu kümede yer alıyorsa, $\Re^n$'de yer alan noktalar kümesi $d\mathbf{y}/dt=g(\mathbf{y})$'ye göre değişmezdir.*

Tanım 5.9 L Kümesi: *Z'deki en büyük değişmez kümedir.*

Bu küme çözümün yakınsayabileceği bütün olası noktaları içerir. Liapunov fonksiyonu L'de değişmez (çünkü türevi sıfırdır) ve eğrisi L'de kapana kısılmış olarak kalacaktır (çünkü değişmez küme). Bu nedenle, eğer bu küme tek bir kararlı noktaya sahipse, bu nokta asimptotik olarak kararlıdır.

Teorem 5.10 LaSalle'nin Değişmezlik Teoremi: *Eğer* $d\mathbf{y}/dt = g(\mathbf{y})$ *için* E *bir Liapunov fonksiyonu ise, bütün* $t > 0$ *için* G*'de kalan her bir çözüm,* $\mathbf{y}(t)$, $t \to \infty$ *için* $L^{\circ} = L \cup \{\infty\}$*'ye yaklaşır. Eğer bütün eğriler sınırlandırılmış ise,* $t \to \infty$ *için* $\mathbf{y}(t) \to L$ *olur.*

Eğer bir eğri G'de kalırsa, ya L'ye yakınsayacaktır ya da sonsuza gidecektir. Eğer bütün yörüngeler sınırlandırılmış ise, o zaman hepsi L'ye yakınsayacaktır.

LaSalle Değişmezlik Teoremi, Liapunov Kararlılık Teoreminin kapsamını genişletir ve Hopfield ağlarının tasarımında kullanılır.

Sonuç 5.11 LaSalle'nin Sonucu: G, $\Omega_{\eta} = \{\mathbf{y} : E(\mathbf{y}) < \eta\}$*'nin bir bileşeni olsun.* G*'nin sınırlı olduğunu,* $dE(\mathbf{y})/dt \leq 0$ *varsayalım ve* $L^{\circ} = kapayan\ kısım(L \cap G)$ G*'nin bir alt kümesi olsun. Bu durumda* L° *çekim merkezidir ve* G*'de çekim bölgesinde yer alır.*

LaSalle teoremi ve sonucu çok güçlüdür ve sadece hangi noktanın kararlı (L°) olduğunu söylemez, aynı zamanda kısmi çekim bölgesini de (G) belirler.

Örnek

Aşağıda verilen doğrusal olmayan sistem için;

$$dy(t)/dt = -(y(t)-1)(y(t)-2) \ ; \ E(\mathbf{y}) = (y-2)^2$$

sistemin denge noktaları ve çekim bölgesine ilişkin bilgiler aşağıdaki gibi bulunabilir:

Denge noktalarını bulabilmek için, $dy(t)/dt = 0$'a eşitlenir.

$$0 = -(y-1)(y-2) \Rightarrow y = 1, y = 2 \text{ denge noktalarıdır.}$$

LaSalle'nin sonucunu kullanabilmek için dE/dt'nin bulunması gerekir.

$$\frac{dE}{dt} = \frac{\partial E}{\partial y}\left(\frac{dy}{dt}\right) = 2(y-2)[-(y-1)(y-2)] = -2(y-1)(y-2)^2$$

$$G = \Omega_\eta = \{y: E(y) < \eta\}$$

Örneğin, η = 0.5 için;

$$G = \Omega_{0.5} = \{y: (y\text{-}2)^2 < 0.5\}$$

$(y\text{-}2)^2 < 0.5$'in çözümü; $\pm (y\text{-}2) < (0.5)^{1/2}$ veya $1.3 < y < 2.7$ olur.

Bu durumda dE/dt G'de negatif tanımlıdır.

G'de yer alan ve dE/dt'nin sıfıra eşit olduğu noktalardan oluşan Z kümesinin bulunması: dE/dt'nin sıfıra eşit olduğu iki nokta vardır ($y = 1$ ve $y = 2$) ve bunlardan sadece bir tanesi G'de yer alır. Bu yüzden;

$$Z = \{y: y = 2\}$$

Z'deki en büyük değişmez kümenin (L) bulunması:Z'de sadece tek bir nokta olduğundan bu nokta denge noktasıdır.

$$L^\circ = L = Z$$

Bunun anlamı, G, 2 için çekim bölgesinde yer alır.

Aynı yaklaşımı, η'nün 1.0'a kadar olan değerleri için de kullanılabilir. Bu yüzden, y=2 için çekim bölgesi en azından $\{y: 1 < y < 3\}$'i içermelidir.

Eğer $\eta \geq 1$ olduğu bölgeler ele alınırsa, Z hem 1'i hem de 2'yi içerir ve dE/dt G'de işaret değiştirecektir. Bu durumda, $y = 1$ için çekim bölgesi hakkında Liapunov fonksiyonu ve LaSalle'nin Değişmezlik Teoreminin sonucunu kullanarak herhangi bir şey söylenemez.

5.3.1 Hopfield Ağlarında Kararlılık

Bir sinir ağı modeli ile ilişkilendirilen ve ağın durumu değiştikçe değeri sürekli azalan bir fonksiyonun var olduğu varsayılırsa, er veya geç bu fonksiyon bir minimum değere erişir ve ağın kararlı olduğunu garanti edecek şekilde o değerde kalır. Bu özellikteki fonksiyonlar Liapunov fonksiyonlarıdır. Böyle bir modelin kararlılığını ispat etmek için bir önceki bölümde değinilen LaSalle Değişmezlik Teoremi veya Cohen-Grossberg kararlılık teoremi kullanılır (Cohen ve Grossberg, 1983).

Teorem 5.12 Cohen-Grossberg Teoremi: *Aşağıdaki biçimde ifade edilen;*

$$\frac{dx_i}{dt} = a_i(x_i)\left[b_i(x_i) - \sum_{j=1}^{n} W_{ij} g_j(x_j) \right] \tag{5.7}$$

ve şu koşulları sağlayan;

- $\mathbf{W}=[W_{ij}]$ *matrisi simetriktir.*
- a_i $(a_i \geq 0)$ *ve* b_i *fonksiyonları süreklidir.*
- a_i *monoton olarak azalmayan ve türevi alınabilen bir fonksiyondur.*

herhangi bir doğrusal olmayan dinamik sistem, Liapunov enerji fonksiyonuna dönüştürülebilir:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} g_i(x_i) g_j(x_j) - \sum_{i=1}^{n} b_i(x_i) g_i(x_i) \tag{5.8}$$

Bu teorem, davranışı (5.7) ile yönetilen bir dinamik sistemle ilgili olarak, (5.8) formunda ifade edilebilen ve sistem evrildikçe asla artmayan bir fonksiyonun var olduğunu ispatlar. Bu nedenle bu teorem, yukarıdaki koşullar sağlandığında, başlangıç durumu ne olursa olsun sistemin kararlı olacağını garanti eder. Ancak, sistemin başlangıç durumu yerel minimumunun konumunu etkiler.

Teorem 5.13 *Eğer* $\mathbf{W}$ *pozitif tanımlı veya pozitif kısmi-tanımlı ise birim hiperküpün içinde veya yanal yüzlerinde* E*'nin minimum noktası yoktur.*

Eğer $\mathbf{W}$ pozitif tanımlı ise E ikinci mertebeden ve konkav bir fonksiyon olduğundan tek bir genel maksimum noktasına sahiptir. Eğer $\mathbf{W}$ pozitif kısmi-tanımlı ise E, $\mathbf{W}$'nin en az bir özdeğeri (eigenvalue) sıfırdır ve E'nin yerel minimumu olan bir sırtın oluşmasına yol açar. Kısıtlandırılmış (0 veya 1 olarak) bir genel maksimum bu sırtın hiperküpün sınırları ile kesiştiği noktada oluşabilir. Ancak enerjinin enküçüklenmesi bu noktanın sabit bir nokta olmasını önler.

Teorem 5.14 *Eğer* $\mathbf{W}$ *negatif tanımlı ise, birim hiperküpün içinde sabit bir nokta* E*'nin tek kısıtlandırılmamış genel bir minimum noktasıdır. Hiperküpün yanal yüzlerindeki sabit noktalar* E*'nin kısıtlandırılmış yerel minimumudur.*

Eğer $\mathbf{W}$ negatif tanımlı ise E konveks bir fonksiyon olduğundan, tek bir kısıtlandırılmamış genel minimum noktasına sahiptir. Hiperküpün sınırları boyunca olan konvekslik yanal yüzlerde kısıtlandırılmış yerel minimuma yol açar.

Teorem 5.15 *Eğer* $\mathbf{W}$ *negatif kısmi-tanımlı ise birim hiperküpün içinde birçok nokta* E*'nin kısıtlandırılmamış genel minimum noktalarıdır. Hiperküpün yanal yüzlerdeki sabit noktalar ise kısıtlandırılmış yerel minimumlarıdır.*

Eğer $\mathbf{W}$ negatif kısmi-tanımlı ise, E hiperküpün iç kısmında bir vadi boyunca ve bu vadinin hiperküpü kestiği yanal yüzeylerde birçok genel minimuma yol açacak biçimde enküçüklenir.

Bu sonuçlar, ağırlık matrisine bağlı olarak, yakınsamanın hiperküpün içindeki bir sabit noktadan sabit bir köşe noktasına doğru yönlenmesini sağlamak için tavlama tekniğine gerek olup olmadığını belirleyebilmeyi sağlar. Aslında, $\mathbf{W}$ pozitif tanımlı veya pozitif kısmi-tanımlı ise hiperküpün iç kısmında E'nin bir yerel minimumu olmayacaktır (Şekil 5.4).

Sonuç olarak, Hopfield sinir ağlarında sabit noktaların kararlılığı ve konumu ile ilgili olarak aşağıdaki açıklamalar yapılabilir (Smith ve diğ, 1996a; Principe ve diğ, 2000):

Açıklama 5.16 Sistemin başlangıç durumunun birim hiperküpün içinde olduğu kabul edildiğinde, araştırma eğrisi bu hiperküpün dışına asla çıkmayacaktır. Ayrıca, ağ parametrelerine bağlı olarak, birim hiperküpün bir köşe noktasına, içindeki veya yanal yüzündeki bir noktaya yakınsayacaktır.

Açıklama 5.17 Eğer $W_{ii} \geq 0, \forall i$ için ise $\mathbf{W}$ ya pozitif tanımlı, pozitif kısmi-tanımlı veya tanımsızdır (köşegendeki terimlere bağlı olarak). Teorem 5.12'den görüldüğü gibi, birim hiperküpün içinde E'nin minimum noktası yoktur. E'nin yerel minimumları birim hiperküpün köşe noktalarıdır. Ayrıca, bütün minimum köşe noktaları asimptotik olarak kararlıdır. Eğer $\left|\frac{W_{ii}}{2}\right| > \left|\Delta_i E(\mathrm{y})\right|$ değilse maksimum köşe noktaları kararlı değildir.

Açıklama 5.18 Eğer $W_{ii} < 0$ ise, $\mathbf{W}$ ya negatif tanımlı, negatif kısmi-tanımlı veya tanımsızdır. Eğer $\mathbf{W}$ negatif tanımlı veya negatif kısmi-tanımlı ise birim hiperküpün içindeki bir sabit nokta E'nin yerel minimumudur. Eğer $\mathbf{W}$ negatif kısmi-tanımlı veya tanımsız ise birim hiperküpün yanal yüzlerindeki sabit noktalar da E'nin yerel

minimumu olabilir. E'nin bütün maksimum değer veren köşe noktaları kararsızdır. Eğer $\left|\frac{W_{ii}}{2}\right| > \left|\Delta_i E(y)\right|$ değilse bütün minimum değer veren köşe noktaları asimptotik olarak kararlıdır.

5.3.2 Hopfield Ağlarında Enerji Fonksiyonu ve Yakınsama

Sürekli bir Hopfield sinir ağı için, (5.8) denklemindeki gibi karakterize edilen Liapunov enerji fonksiyonu aşağıdaki biçimde ifade edilebilir:

$$E_s = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} g_i(u_i) W_{ij} g_j(u_j) - \sum_{i=1}^{n} I_i g_i(u_i) + \sum_{i=1}^{n}\int_0^{y_i} g_i^{-1}(v)dv \qquad (5.9)$$

Enerji fonksiyonu ifadesi, bazı fiziksel sistemlerle benzeştiği için kullanılır. Çünkü İE'den oluşan sinir ağı, ağırlık matrisi (W) simetrik olduğunda, fiziksel sistemin minimum enerji düzeyinde bulunması gibi bir minimuma doğru dengelenir (EK-B). Bu yüzden, sinir ağının kararlı durumu, enerji fonksiyonunun genel minimum olması gerekmeyen bazı minimumlarına karşılık gelir.

Sürekli Hopfield ağı, aktivasyon fonksiyonunun eğiminin yüksek olması (high-gain) durumunda (adım fonksiyonu), $g(u_i)$, eşik değeri sıfır olan ayrık Hopfield modelinin davranışını sergiler ve E_s'deki entegralli terim yok olur. Böylece, eğimi dik olan aktivasyon fonksiyonunda;

$$E_s \rightarrow E_a = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq 1}^{n} W_{ij} y_i y_j - \sum_{i=1}^{n} I_i y_i \qquad (5.10)$$

ayrık Hopfield modeli için bir Liapunov fonksiyonuna dönüşür. E_a'nin yerel minimumu, ayrık Hopfield modelinin durum uzayı olan bir birim hiperküpün köşe noktalarından birisidir. Bu nedenle, aktivasyon fonksiyonunun dik eğimli olması durumunda, ağırlık matrisi simetrik ve $g'(u_i)$'nin ters fonksiyonu mevcutsa, sürekli Hopfield ağı da E_a'yı minimum yapan bir 0-1 köşe noktasına yakınsar.

Sürekli modelin kararlı durumları için ağırlık matrisinin köşegeninin sıfır (W_{ii}=0) olması gerektiğine dair yanlış bir inanış vardır. Bu düşünce hiç şüphesiz Hopfield'in basitleştirici varsayımına (W_{ii}=0) dayanmaktadır. Oysa gerçekte sürekli modelin

E_s'nin bir minimumuna yakınsaması için, $\mathbf{W}$'nin köşegenlerine dair herhangi bir kısıtlama söz konusu değildir (Smith, 1996a). Ancak, eğer $W_{ii}<0$ ise enerji fonksiyonu dışbükey olduğundan minimum hiperküpün iç kısmında olabilir. Bu durumda, çözümün köşe noktalarına yönelmesi için genellikle tavlama benzetimi teknikleri kullanılır.

Örnek

Hopfield'in (1984) orijinal makalesinde yer alan amplifikatör örneği ele alınırsa; bu örnekte, karakteristik fonksiyonu aşağıdaki gibi verilen bir sistem incelenmiştir.

$$y = g(u) = \frac{2}{\pi}\tan^{-1}\left(\frac{\gamma\pi n}{2}\right) \quad \text{veya} \quad u = \frac{2}{\gamma\pi}\tan\left(\frac{\pi}{2}y\right)$$

İki amplifikatörün birinin çıktısı diğerinin girdisi olacak ve arada 1 birimlik direnç olacak şekilde bağlandığı varsayılsın. Bu durumda ağırlık matrisi;

$$\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Eğer amplifikatörün girdi kapasitansı da 1 kabul edilirse, η = 1 (=*direnç*kapasitans*) olacaktır. Ayrıca γ=1.4 ve $I_1 = I_2 = 0$ olsun. Bu durumda, $\mathbf{I}^T$ = [0, 0] olur. Liapunov fonksiyonunun matris biçiminde gösterimi;

$$E(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} - \mathbf{I}^T\mathbf{y} + \sum_{i=1}^{n}\int_0^{y_i} g_i^{-1}(v)dv$$

Bu örnek için Liapunov fonksiyonunun ilk terimi;

$$-\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} = -\frac{1}{2}[y_1, y_2]\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -y_1 y_2$$

İkinci terim sıfır olacaktır, çünkü sapmalar sıfırdır. Üçüncü terimin i'nci parçası;

$$\int_0^{y_i} g_i^{-1}(v)dv = \frac{2}{\gamma\pi}\int_0^{y_i}\tan(\frac{\pi}{2}v)dv = \frac{2}{\gamma\pi}\left[-\log\left[\cos(\frac{\pi}{2}v)\right]\frac{2}{\pi}\right]_0^{y_i}$$

Bu ifade sadeleştirilirse;

$$\int_0^{y_i} g_i^{-1}(v)dv = -\frac{4}{\gamma\pi^2}\log\left[\cos(\frac{\pi}{2}y_i)\right]$$

Sonuçta, bu terimler enerji fonksiyonunda yerine koyulursa;

$$E(\mathbf{y}) = -y_1y_2 - \frac{4}{1.4\pi^2}\left[\log\{\cos(\frac{\pi}{2}y_1)\} + \log\{\cos(\frac{\pi}{2}y_2)\}\right]$$

Ağ denklemleri (5.3) yazılmak istenirse, η=1 ve $\mathbf{I}$=$\mathbf{0}$ için;

$$\frac{d\mathbf{u}}{dt} = -\mathbf{u} + \mathbf{W}g(\mathbf{u}) = -\mathbf{u} + \mathbf{W}\mathbf{y}$$

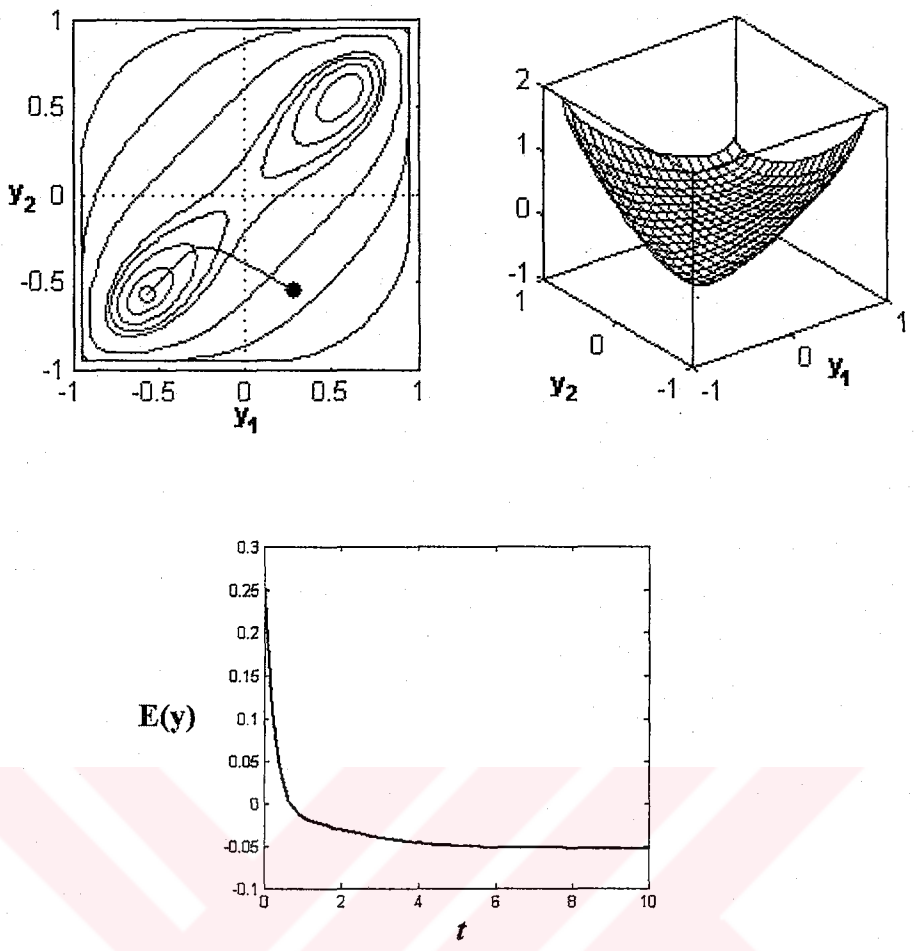
Eğer, ağırlık matrisi ($\mathbf{W}$) bu denklemde yerine koyulursa, aşağıdaki iki eşitlik elde edilir;

$$\frac{du_1}{dt} = y_2 - u_1; \qquad \frac{du_2}{dt} = y_1 - u_2$$

Nöronların çıktıları;

$$y_1 = \frac{2}{\pi}\tan^{-1}\left(\frac{1.4\pi}{2}n_1\right); \qquad y_2 = \frac{2}{\pi}\tan^{-1}\left(\frac{1.4\pi}{2}n_2\right)$$

Bu örneğin Liapunov fonksiyonu konturları, enerji yüzeyi ve enerji fonksiyonunun zamana bağlı olarak değişimi Şekil 5.5'de görüldüğü gibi bulunur. Şekildeki kontur çizgileri, Liapunov fonksiyonunun sabit değerlerini gösterir. Sistem, biri şeklin sol altında diğeri ise sağ üstte yer alan iki çekim noktasına sahiptir. Sistem, sağ alttan başlandığında sol alttaki kararlı noktaya yakınsar.



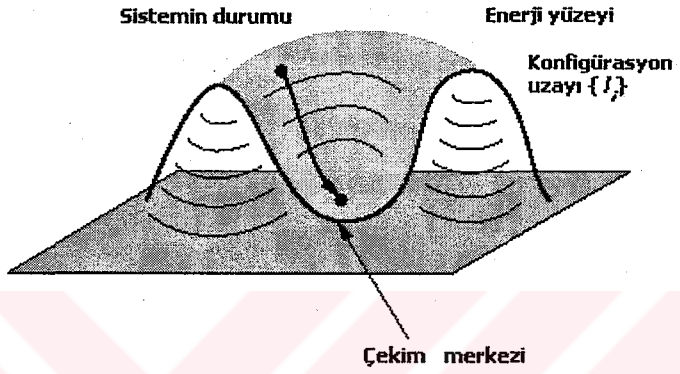
Şekil 5.5 : Hopfield örneğinde Liapunov fonksiyonu ve yakınsama

Enerji fonksiyonu, durumlara ait { y_i } konfigürasyonunun bir işlevidir, yani ağ herhangi bir girdiye cevap verdiğinde artmaz. Çünkü her zaman bir işlem elemanı durumu değiştirir, E_a azalır; İE değişmediği zaman E_a aynı kalır. Bunun anlamı, genel ağ dinamikleri saklanmış örüntülerden birisine göre sistemin durumunu E_a'nın gradyanı boyunca bir minimuma çeker. Girdi uzayındaki minimumun konumu, ağ için seçilen ağırlıklarla belirlenir. Sistem minimuma eriştiğinde orada kalacaktır, bu nedenle minimum sabit bir noktadır.

Hopfield ağı bir girdi aldığında, sistemin durumu ağırlık uzayında herhangi bir yerde konuşlanır. Ardından, sistem dinamikleri girdi örüntüsüne en yakın olan hafızayı serbest bırakır. Bu yüzden, her bir sabit nokta civarında dinamikleri minimuma götüren bir çekim havuzu vardır. Bu durum, Hopfield ağının neden girdi örüntüsündeki belirsizliklere (gürültü eklenmesi veya kısmi girdi) karşı çok tutarlı olduğunu açıklar. Saklanmış bir örüntü için sistemin durumu bir kez çekim havuzuna girerse, sistem çarpıtılmamış örüntüleri serbest bırakır. Bununla birlikte pratikte

karşılaşılan bazı problemler de vardır: örneğin, sisteme örüntüler yüklendiğinde sistemi bilinmeyen konumlara çekebilecek sahte hafızalar gibi.

Minimum noktası ile çekim havuzunun yer aldığı bir enerji yüzeyi, yerçekiminin olduğu bir ortamda bir taneciğe benzeyen sayısal enerji yüzeyinin düşünsel bir resmini ortaya koyar (Şekil 5.6). Çok karmaşık bir ağın genel özellikleri çok basit terimlerle ifade edildiğinden bu benzetim çok güçlüdür. Aynı zamanda, her bir bağlantıyı yerel olarak belirlemenin yolları da vardır.



Şekil 5.6 : Konfigürasyon uzayında sayısal enerji (Principe ve diğ., 2000)

Hopfield bilgi işleme ile sistem dinamikleri arasındaki güçlü bağlardan birisini bulmuştur. Sayısal enerji fonksiyonunun varlığı, Hopfield ağının durumunun, aynen yerçekiminin olduğu bir yerde tepeden yuvarlanmaya bırakılan bir top gibi aynı dinamik problemin saklanmış örüntüye yakınsamasını sağlar (Bkz. Bölüm 5.3). Noktasal çekim merkezlerinin yer aldığı bir dinamik sistem donanımı, ister *Çok Büyük Çapta Tümleşim* (Very Large Scale Integration - VLSI) veya biyolojik nöron olsun veya olmasın ilişkilendirici hafızayı gerçekler.

Hopfield ağı dinamik bir sistem olmasına karşın yakınsamadan sonra statikleşir. Bu nedenle beynimizde sürekli gerçekleşen geçici işlemleri anlamak için diğer paradigmalar dikkatle incelenmelidir. Dinamik sistemler, noktasal çekim merkezi ve kaotik çekim noktaları gibi daha fazla tuhaflıklar içerebilir.

5.4 Hopfield-Tank Sinir Ağlarının Eniyilemede Kullanılması

Enerji yüzeyi, bir problemin kısıtları ile ilişkilendirilebildiğinden, Hopfield ağları eniyilemede de kullanılır. Probleme ait çözümünün ağa dönüştürülmesi zor olan

kısımdır ve durum durum ele alınarak gerçekleştirilmesi gerekir (Hopfield ve Tank, 1985).

Hopfiled ve Tank tarafından önerilen yöntem kullanılarak, ağın enerji fonksiyonu minimum yapılmaya çalışılan kombinatorik eniyileme probleminin amaç fonksiyonuna denk hale getirilir ve problemin kısıtları enerji fonksiyonuna ceza terimleri olarak dahil edilir. Bu şekilde, bilinen başlangıç parametreleri için, ağ sistem enerjisi minimize edilmek suretiyle bir en iyi veya iyice çözümde dengelenecektir. $\mathbf{A}_i$ (katsayı matrisi $\mathbf{A}$'nın i'nci satırı) ve $\mathbf{x}$'in N boyutlu bir vektör olduğu bir eniyileme problemi verilsin:

Amaç fonksiyonu: $\min \quad f(\mathbf{x})$

Kısıtlar:
$$\begin{aligned} [\mathbf{A}]_1 \mathbf{x} &= b_1 \\ [\mathbf{A}]_2 \mathbf{x} &= b_2 \\ &\vdots \\ [\mathbf{A}]_m \mathbf{x} &= b_m \end{aligned}$$

$[\mathbf{A}]_k$ kısıt matrisi olan $\mathbf{A}$'nın k'ncı satırını, $\mathbf{x}$ ise $\underline{n}$ boyutlu karar değişkeni vektörünü ve k da kısıt sayısını gösterir. Kısıtsız bir model elde etmek için, kısıtlar olursuz çözümleri cezalandıran terimlere dönüştürülmelidir (Lagrangian relaxation). Kısıtlar, karar değişkenleri ($x_1,x_2,...,x_n$) sadece olası çözümleri gösterdiklerinde, k = 1,2,...,m için $C_k(x_1,x_2,...,x_n)$=0 (yani, $C_k(x_1,x_2,...,x_n) = [\mathbf{A}]_k \mathbf{x} - b_k$) şeklinde negatif değer almayan ceza fonksiyonları olarak gösterilebilirler. Ceza fonksiyonları amaç fonksiyonu ile toplanarak, orijinal kısıtlı eniyileme problemi kısıtsız bir eniyileme problemi olarak yeniden formüle edilebilir ve amaç;

$$G = \lambda_0 f(x_1,x_2,...,x_n) + \sum_{k=1}^{m} \lambda_k C_k(x_1,x_2,...,x_n) \qquad (5.11)$$

veya $$G = \lambda_0 f(\mathbf{x}) + \lambda_1([\mathbf{A}]_1 \mathbf{x} - b_1) + \lambda_2([\mathbf{A}]_2 \mathbf{x} - b_2) + \cdots + \lambda_m([\mathbf{A}]_m \mathbf{x} - b_m)$$

değerini minimize etmektir, öyle ki; λ_0 amaç fonksiyonunun ölçekleme faktörü, λ_k ise k'ncı kısıtın ölçekleme faktörüdür (Lagranj çarpanı).

Bir sonraki aşamada, Liapunov enerji fonksiyonu (5.10) ile maliyet fonksiyonu (5.11) arasında cebirsel tanımlama yapılır. Amaç fonksiyonunun (f) aşağıdaki gibi bir enerji fonksiyonu:

$$E_{f(\mathbf{x})} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}^{obj} x_i x_j - \sum_{i=1}^{n} I_i^{obj} x_i \tag{5.12}$$

ve problem kısıtlarının C_k (k = 1, 2, ... , m) ise:

$$E_{C_k(\mathbf{x})} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}^{k} x_i x_j - \sum_{i=1}^{n} I_i^{k} x_i \tag{5.13}$$

biçiminde ifade edilebileceğini varsayalım. Daha sonra, (5.12) ve (5.13) kısıtsız eniyileme probleminde (5.11) yerine koyularak, ilgili enerji fonksiyonu aşağıdaki gibi formüle edilebilir (Bkz. Tagliarini ve diğ., 1991):

$$E_G = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\lambda_0 W_{ij}^{obj} + \lambda_1 W_{ij}^{1} + \lambda_2 W_{ij}^{2} + \ldots + \lambda_m W_{ij}^{m}\right] x_i x_j$$

$$-\sum_{i=1}^{n}\left[\lambda_0 I_i^{obj} + \lambda_1 I_i^{1} + \lambda_2 I_i^{2} + \ldots + + \lambda_m I_i^{m}\right] x_i \tag{5.14}$$

Liapunov enerji fonksiyonu yukarıdaki denklemde gösterilen sürekli Hopfield-Tank modelinin diferansiyel denklemi aşağıdaki gibi yazılabilir:

$$\frac{du_i}{dt} = -\frac{u_i}{\eta_i} + \sum_{j=1}^{n}\left[\lambda_0 W_{ij}^{obj} + \lambda_1 W_{ij}^{1} + \lambda_2 W_{ij}^{2} + \ldots + \lambda_m W_{ij}^{m}\right] x_j$$

$$-\left[\lambda_0 I_i^{obj} + \lambda_1 I_i^{1} + \lambda_2 I_i^{2} + \ldots + \lambda_m I_i^{m}\right] \tag{5.15}$$

Bu denklem, ara bağlantıların mukavemeti veya ağırlıkları ile bir minimumda dengelenen bir kısıtsız eniyileme problemini gösteren yapay sinir ağı için ihtiyaç duyulan harici verileri belirler. Toplam bağlantı mukavemeti, bağımsız olarak amaç fonksiyonu ve kısıtlardan elde edilen ara bağlantı mukavemetlerinin toplamına eşittir. Benzer şekilde, verilen bir İE'ye olan harici girdi, amaç fonksiyonu ve

kısıtlardan birbirinden bağımsız olarak ortaya çıkan girdilerin toplanmasıyla elde edilir.

Hopfield ağındaki ağırlıklar, kararlı bir ağı garanti eden Hebbian öğrenme kullanılarak hesaplanabilir. Ağırlıkları hesaplamada yinelemeli geriye yayılım da kullanılabilir, fakat bu durumda ağırlıkların simetrik olması garanti değildir.

Uygun bir enerji fonksiyonu ve ağ parametreleri (ağırlıklar ve girdi verileri) seçildikten sonra, her bir İE'nin etkinlik düzeyi 0.5 civarında rasgele dağılacak şekilde başlangıç durumuna getirilir. Böylelikle, sistemin başlangıç durumu yaklaşık olarak N-boyutlu hiperküpün merkezi olur ve başlangıç durumunun yansız olması sağlanır. Senkronize olmayan güncelleme ile ağın bir minimum enerji düzeyine gitmesi sağlanır. Ancak, bu kararlı durumların KEP'in olurlu veya iyi çözümlerine karşılık gelmesi söz konusu olmayabilir ve bu durum H-T modelinin en büyük tuzaklarından birisidir. Enerji fonksiyonu birçok terim içerdiğinden, pek çok yerel minimum vardır ve bu terimlerden hangisinin enküçüklenmesi gerektiğine dair ödünleşme yapmak gerekir. KEP'in olursuz bir çözümü, en az bir kısıtın ceza teriminin sıfırdan farklı olması durumunda ortaya çıkar. Eğer böyle bir şey söz konusu olursa, aynı minimum enerji düzeyi için amaç fonksiyonu değeri oldukça küçüktür. Dolayısıyla çözüm iyi ancak olursuzdur. Alternatif olarak, bütün kısıtlar sağlanmış olabilir, fakat amaç fonksiyonunu genel olarak enküçüklemeyen yerel bir minimum elde edilebilir. Bu durumda, çözüm olurludur ancak iyi değildir. Amaç fonksiyonuna ait çarpan (λ_0) artırılarak ilgili terim minimum olmaya zorlanabilir. Ancak bu durum genellikle diğer terimlerin artmasına neden olur. Bu ödünleşme problemi, enerji fonksiyonunun terimlerini dengeleyen ve her bir terimin eşit öncelikle enküçüklenmesini temin eden en iyi ceza parametre değerleri (Lagranj çarpanları) bulunarak çözülür. Ancak o zaman kısıt terimleri sıfır olacaktır (*olurlu çözüm*) ve amaç fonksiyonu da en küçüklenecektir (*iyice çözüm*).

Hopfield ve Tank, yöntemlerini pek çok eniyileme problemine başarıyla uygulamışlardır (Tank ve Hopfield, 1986). Bununla birlikte, en çok dikkati *gezgin satıcı problemi* (GSP) için elde ettikleri sonuçlar çekmiştir. Ancak, Hopfield ve Tank diferansiyel denklemin benzetiminde kullandıkları yöntem ve durdurma kriteri gibi bazı pratik detayları makalelerinde belirtmemişlerdi. İki İngiliz araştırmacı Wilson ve Pawley tarafından 1988 yılında yayınlanan bir makale, KEP'i çözmede H-T yönteminin güvenirliliği ve geçerliliğine dair soru işaretlerine neden olmuştur (Wilson ve Pawley, 1988). Aynı problem seti için buldukları sonuçlar Hopfiled ve Tank'ın

bulduklarından çarpıcı bir biçimde farklı ve yetersizdi. Wilson ve Pawley orijinal H-T formülasyonu için birçok değişiklik önerdiler. Ancak, bu değişikliklerin hiç birisi de çözümün kalitesini ve geçerliliğini önemli oranda artırmadı. Sonuçta, H-T yönteminin güvenilir olmadığı ve iyileştirilmesi için de fazla bir fırsat vaat etmediğini belirttiler.

Şüphesiz olursuz çözüm elde edilmesi, enerji fonksiyonundaki pek çok terimden ve bu terimleri dengeleyecek şekilde ceza parametrelerini seçme zorluğundan kaynaklanır. Ancak, göz ardı edilen gerçek; Hopfield ve Tank'in GSP için kullandığı matematiksel modelin, diğerlerinin kullandığı 0-1 Tamsayılı Programlama modeline denk ancak farklı bir model (ikinci dereceden) olması ve standart Hopfield ağları için çok daha uygunluğudur (Smith, 1996).

Hopfield-Tank enerji fonksiyonundaki ceza terimlerini eşit olarak dengeleyebilecek bir yöntem bulma çalışmaları hala devam eden bir konudur. Ağ olurlu bir çözüme yakınsamayı başarsa da diğer tekniklerle kıyaslandığında çözüm kalitesi büyük olasılıkla kötüdür. Çünkü Hopfield ağı bir iniş tekniği olduğundan karşılaştığı ilk yerel minimuma yakınsar.

5.5 Olurluluğun Sağlanması

İlk olarak Aiyer (1991) tarafından önerilen genel enerji fonksiyonu:

$$E(\mathbf{x}) = f(\mathbf{x}) + \frac{\gamma}{2}\mathbf{C}^g(\mathbf{x}) \tag{5.16}$$

şeklinde yazılabilir. Bu formülasyonda $f(\mathbf{x})$, standart ikinci dereceden amaç fonksiyonudur ($f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \mathbf{x}^T\mathbf{Q}\mathbf{x}$). $\mathbf{C}^g$ tek kısıt terimidir ve $\mathbf{x}$ vektörünün kısıt düzleminden ($\mathbf{Ax} = \mathbf{b}$) sapma miktarını gösterir. Bu enerji fonksiyonunun avantajı sadece bir parametreye (γ) gereksinim duymasıdır. Eğer γ yeteri kadar büyükse kısıt terimi sıfır olmaya zorlanarak çözümün geçerliliği garanti altına alınır. Böylece çözüm kısıt düzleminde yer alacaktır. Gerekli düzenlemeler EK-C'deki biçimde yapıldığında, olurlu çözüm veren Hopfield ağının enerji fonksiyonu aşağıdaki gibi elde edilir:

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\left[-2\mathbf{Q} + 2\delta\mathbf{I} + \gamma(\mathbf{\wp} - \mathbf{I})\right]\mathbf{x} - \mathbf{x}^T\left[\gamma\mathbf{s} - \mathbf{c} - \delta\mathbf{1}\right] + \frac{\gamma}{2}\mathbf{s}^T\mathbf{s}$$

Olurlu Hopfield ağının ağırlık matrisi ve harici girdi vektörü:

$$\mathbf{W} = -2\mathbf{Q} + 2\delta\,\mathbf{I} + \gamma(\wp - \mathbf{I}) \tag{5.17}$$

$$\mathbf{I} = \gamma\,\mathbf{s} - \mathbf{c} - \delta\,\mathbf{1} \tag{5.18}$$

Ağın yakınsadığı çözümün en iyi çözüme yakın, hem 0-1 tamsayı hem de olurlu olması için parametrelerin (γ ve δ) en uygun biçimde nasıl seçilmesi gerektiği hala bir problemdir. Ancak bu sorun, γ için yakınsamanın kısıt düzleminde kalmasını sağlayacak biçimde yeteri kadar büyük bir değer atanırsa bertaraf edilebilir. Diğer parametre (δ) ise, çözümün bir tepe noktasına gitmesine sağlayacak ve γ'nın küçük bir oranına eşit olacak biçimde seçilebilir. Böylelikle enerji fonksiyonunun Liapunov anlamında azalması, olurlu çözümün maliyetini enküçükleyecek şekilde köşelere doğru hareket etmesini garanti altına alır.

Ancak, Aiyer tarafından önerilen bu değiştirilmiş enerji fonksiyonuna sahip Hopfield ağının elektronik devre olarak gösterimi yaklaşımı çok uygun ve ümit verici olmasına rağmen, bilgisayar ortamında benzetiminin oldukça zor olduğu Gee tarafından gösterilmiştir (Gee, 1983). Bununla beraber, eğer γ'nın büyük değerleri için enerji fonksiyonunun (5.16) enküçüklenmesi ile amaç fonksiyonunun ($f(\mathbf{x})$) en dik inişi arasında bir ilişkilendirme yapılırsa, ağın benzetimi etkin bir biçimde gerçekleştirilebilir (Gee, 1993; Chu, 1992).

Gee'nin önerdiği benzetim yönteminde Hopfield ağının içsel dinamiklerinde, (5.3) denklemindeki ($-\frac{u_i}{\eta_i}$) terimi atılarak küçük bir değişiklik yapılmıştır. Bunun gerekçesi ise, bu terimin ağın yakınsamasına engel olduğu gösterilmiştir (Takefuji, 1992). Ayrıca, tanjant hiperbolik aktivasyon işlevi yerine aşağıda verilen parçalı doğrusal fonksiyon kullanılmıştır:

$$x_i = g(u_i) = \begin{cases} 0 & u_i \le 0 \\ u_i & 0 < u_i < 1 \\ 1 & u_i \ge 1 \end{cases} \tag{5.19}$$

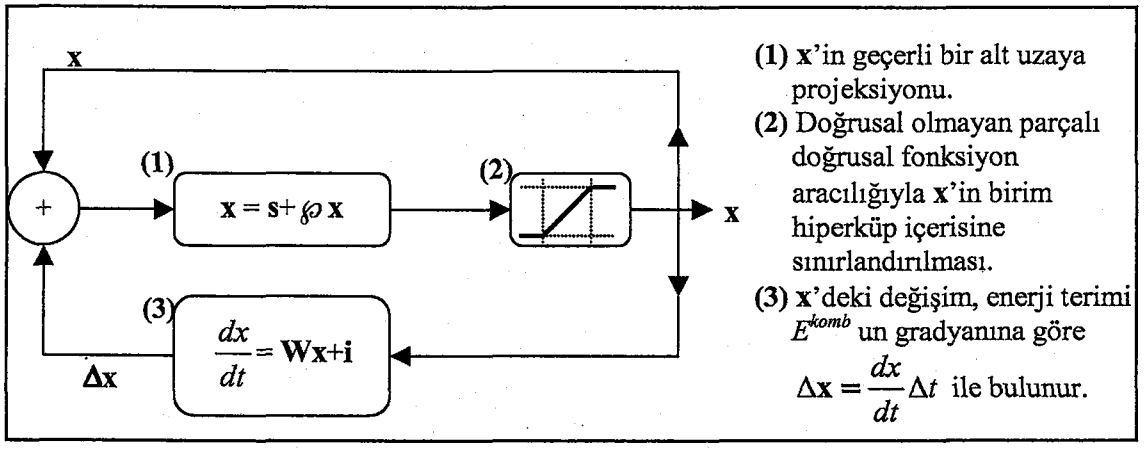
Böylelikle,

$$\frac{du_i}{dt} = \sum_{j=1}^{n} W_{ij} x_j + I_i = -\frac{\partial E(x)}{\partial x_i} = \frac{dx_i}{dt} \text{ (hiperküpte)} \qquad (5.20)$$

ve $$\frac{\partial E(x)}{\partial x_i} = \frac{df(x)}{dx_i} \text{ (kısıt düzleminde, büyük } \gamma \text{ için)}$$

O nedenle benzetim yaklaşımı, **x**'in birim hiperküpün içerisinde kalmasını sağlayarak γ'ya büyük değer verilmek suretiyle kısıt düzlemine iliştirilmesi yoluyla amaç fonksiyonunun en dik inişini gerçekleştirdiği görülebilir. Bu prosedür şu şekilde işler:

Herhangi bir rasgele seçilmiş başlangıç çözümü $\mathbf{x}_0$'dan (olurlu olması gerekmez) hareketle, yeni bir çözüm vektörü ($\mathbf{x}_1$) verecek şekilde kısıt düzlemine projeksiyonu gerçekleştirilir (Şekil 5.7 (1)). $\mathbf{x}_1$ birim hiperküpün içerisinde yer almayabileceğinden, doğrusal olmayan parçalı doğrusal fonksiyon aracılığıyla birim hiperküpün içerisinde yer alan yeni bir vektör ($\mathbf{x}_2$) elde edilir (Şekil 5.7 (2)). Ancak, $\mathbf{x}_2$ vektörü kısıt düzleminde olmayabilir. Bu yüzden, bu süreç hem kısıt düzleminde hem de birim hiperküpün içerisinde yer alan bir $\mathbf{x}_n$ noktası elde edilinceye kadar sürekli tekrarlanır. Böyle bir noktaya erişildiğinde, eniyilenmeye çalışılan enerji teriminin (E) gradyanına göre Denklem 5.20 kullanılarak değiştirilir (Şekil 5.7 (3)). Bu çevrim maliyet fonksiyonunda iyileşme olmayıncaya kadar tekrarlanır.



Şekil 5.7 : Değiştirilmiş Hopfield ağının gerçeklenmesinin şematik gösterimi

5.6 Çözüm Kalitesinin Artırılması

Hopfield ağının çözüm kalitesini artırmak için birçok yöntem kullanılabilir. Ancak, bu yöntemler kabaca deterministik ve stokastik olarak sınıflandırılabilir. Deterministik yaklaşımlara örnek olarak, probleme özel iyileştiriciler (Foo ve diğ., 1989), deterministik tepe-tırmanma (Lo, 1992), Hopfield ağında *kazanan hepsini alır* gibi alternatif nöron modellerinin kullanılması (Amartur ve diğ., 1992) verilebilir. Stokastik yaklaşımlar yerel minimumdan kurtulmayı hedefleyen yöntemleri içerir. Hopfield ağlarına olasılığı sokmanın aslında dört ana yöntemi vardır:

1. Sigmoidal aktivasyon işlevi yerine stokastik olarak karar vermeyi sağlayan aktivasyon işlevinin kullanılması.
2. Ağın ağırlıklarına gürültü eklenmesi.
3. Ağın harici girdilerine (eğilim) gürültü eklenmesi.
4. Yukarıdaki yöntemlerin herhangi bir kombinasyonu.

Boltzman makinesi ayrık bir Hopfield modelini temel alarak birinci yöntemi kullanır (Ackley ve diğ., 1985). Ayrık aktivasyon işlevi olasılıksal olacak şekilde değiştirilir. Tavlama benzetiminde olduğu gibi, her nöronun ikili aktivasyon düzeyinin değiştirilmesinin sonuçları Boltzmann olasılık etmenine göre değerlendirilir. Bu model yerel minimumlardan kurtulmayı sağlar ancak hesaplama zamanı oldukça fazladır. Boltzmann makinesinin verimliliğini ve hızını artırmak için Akiyama sürekli Hopfield ağları ile Boltzmann makinesini bir araya getiren *Gaussgil makinesini* önermiştir (Akiyama ve diğ., 1989). Gausgil makinesi, Hopfield ağında olduğu gibi deterministik aktivasyon işlevi ile sürekli çıktılara sahiptir, ancak her nöronun harici girdisine rastsal bir gürültü eklenmiştir. Bu gürültü, ortalaması sıfır ve standart sapması bir sıcaklık parametresine bağlı olarak değişecek şekilde normal dağılmıştır. Fakat Cauchy makinesi çözüm kalitesini daha da iyileştirmiştir ve genel minimuma yakınsamada Gausgil dağılımından daha fazla şansa sahiptir. Cauchy gürültüsü hem yerel rastsal yürümeyi hem de çözüm uzayında daha büyük rastsal sıçramayı gerçekleştirirken, Gausgil gürültüsü sadece rastsal yürümeyi mümkün kılar. Stokastik aktivasyon işlevinin gradyanının eğiminin büyük olması durumunda, Cauchy makinesi davranış olarak ayrık Hopfield ağına yaklaşır. Bir diğer stokastik yöntem *ortalama-alan tavlamasıdır* (mean-field annealing) ve stokastik ikili Boltzmann makinesinin ortalama aktivasyon düzeyini hesaplar. Stokastik sinir ağları sıklıkla çözümün yerel minimumdan kurtulmasını sağlayacak biçimde tasarlanmıştır. Geçici olarak enerji düzeyinde artışa izin vererek yerel minimumdan kurtulmaya

yardımcı olacak biçimde bir tepe-tırmanan Hopfield ağı da önerilmiştir (Smith, 1996). Bu yaklaşımda, enerji fonksiyonunun azalmasına her zaman izin verilirken, nöronların hareket yönleri değiştirilmek suretiyle yerel minimumdan kurtulmak sağlanır. Bu yöntem tavlama benzetimindeki düşünceye şüphesiz benzer. Bu yöntemin diğer sürekli stokastik yaklaşımlara göre avantajı istikrarlı ve olurlu çözümü garanti altına almasıdır (Smith, 1996).

Nöronlardaki değişim oranı bir $\alpha(t)$ çarpanı ile kontrol edilir ve $\mathbf{x}$'in birim hiperküpün içerisinde;

$$\frac{du_i}{dt} = \alpha(t)\left(\sum_{j=1}^{n} W_{ij}x_j + I_i\right) \tag{5.21a}$$

ve kısıt düzleminde kalmasını sağlar:

$$\frac{\partial x_i}{\partial t} = -\alpha(t)\frac{df(x)}{dx_i} \tag{5.21b}$$

Böylelikle, en dik iniş veya çıkış $\alpha(t) = \pm 1$ değeri için elde edilir. $\alpha(t)$ çarpanı için soğutma çizelgesi;

$$\alpha(t) = random(k(t),1) \qquad k(t) = 1 - 2e^{-t/T_k} \tag{5.22}$$

Her bir zaman noktası için arama uzayında izin verilen rastsal yürüme sayısı problemin büyüklüğüne bağlı olarak sabit bir değer alınır. Başlangıçta $k(0)$=-1 olduğundan enerji değeri artacaktır. Ancak $t \to \infty$ giderken $k(0) \to 1$ ve dolayısıyla $\alpha(t)$ en dik iniş için gerekli olan bir değerine yaklaşacaktır. Böylelikle, başlangıçta enerjinin rasgele artışına izin verilirken, sonuçta ağ bir en dik iniş algoritmasına doğru eğilim gösterecektir.

Bu yöntem, deterministik girdili, aktivasyon fonksiyonuna tavlama benzetimi tipinde bir karar kriterinin iliştirildiği ve sürekli nöron çıktılarına sahip bir stokastik yaklaşımdır ve diğer yöntemlerin çoğu faydalı ve etkili özelliklerini içermesine ilaveten, istikrar ve olurlu çözümü garanti altına alma avantajlarına da sahiptir.

6. TALEP TAHMİNİNİN YAPAY SİNİR AĞLARI İLE GERÇEKLENMESİ

Bu bölümde, çevrim-içi ve çevrim-dışı talep tahmini için farklı sinir ağı modelleri tasarlanarak, en iyi model parametrelerinin ne olacağı, hangi tip öğrenme tekniğinin hangi modelde kullanılması gerektiği, saklı katman sayısı, katmanlarda bulunması gereken işlem elemanlarının sayısını, vb. belirlemek amacıyla test verileri kullanılarak testler yapılmış ve karşılaştırılmıştır.

6.1 Giriş

Ticari bir işletmenin, sürekli dalgalanan bir piyasada rekabetçi pozisyonunu koruyabilmesi ve artırabilmesi için, yöneticilerin eldeki verileri kullanarak zamanında doğru kararlar vermesi gerekir. Kuramsal olarak, eğer pazarlama bölümü bir sonraki dönemin satış miktarlarını tahmin edebiliyorsa, malzeme/satın alma bölümü tam zamanında teslimat yapabilmek için stok seviyesini etkin olarak kontrol edebilir. Ayrıca, üretim bölümü de detaylı planlama ve tesis kullanım oranlarını düzenleyebilir. Böylece üretim maliyetleri düşürülebilir. Bu yüzden, doğru ve zamanında satış tahmininde bulunmak özellikle tedarik zinciri yönetiminde çok önemlidir.

Talep tahmininde geleneksel yöntemler olarak istatistiksel teknikler (tek/çok değişkenli, doğrusal/doğrusal olmayan regresyon, zaman serileri analizi, vs.) yaygın olarak kullanılmaktadır ve bu teknikler çoğunlukla parametriktir. Özellikle eksik veya fazla hatalı verilerin olması veya veriler arasında karmaşık ilişkilerin olması gibi durumlarda istatistiksel yöntemleri kullanmak oldukça zordur. Aslında gerçek yaşamdaki veriler genellikle basit istatistiksel parametrelerle açıklanamaz ve bu verilere ait istatistikler genellikle zamana bağlı olarak değişkenlik gösterirler. Böylesi durumlarda, adaptif sinir ağı teknikleri hem daha etkilidir (yarı-parametrik veya parametrik olmayan), hem de geleneksel yöntemlere göre daha ekonomiktirler. Sinir ağları ile istatistiksel yöntemler arasındaki temel farklılık, sinir ağlarının istatistiksel dağılımla veya verilerin özellikleri ile ilgili herhangi bir varsayım yapmamasıdır. Bir diğer önemli özelliği de doğrusal olmayan bir tahmin yöntemi olduğundan karmaşık verilerin modellenmesinde daha güvenilir olmasıdır. Veriler arasındaki ilişkilerin

dinamik olarak değiştiği sistemlerde analiz yaparken adaptif bir yöntem kullanmak gerekir ki sinir ağları bu özelliği doğal olarak içerisinde barındırır.

Bu ve benzeri nedenlerden dolayı, içerisinde açıklanamayan veriler de içerebilen, genellikle kararsız bir yapı gösteren veya deseni istatistiksel tekniklerle kolaylıkla açıklanamayan talep tahminleri için sinir ağları kullanılabilir. Verilerin tamamının önceden elde bulunması (çevrim dışı - off-line) veya tek tek elde ediliyor olmasına (çevrim içi - on-line) göre tasarlanacak sinir ağı modeli ve çözüm yaklaşımı farklılık gösterecektir.

6.2 Yapay Sinir Ağı Modeli

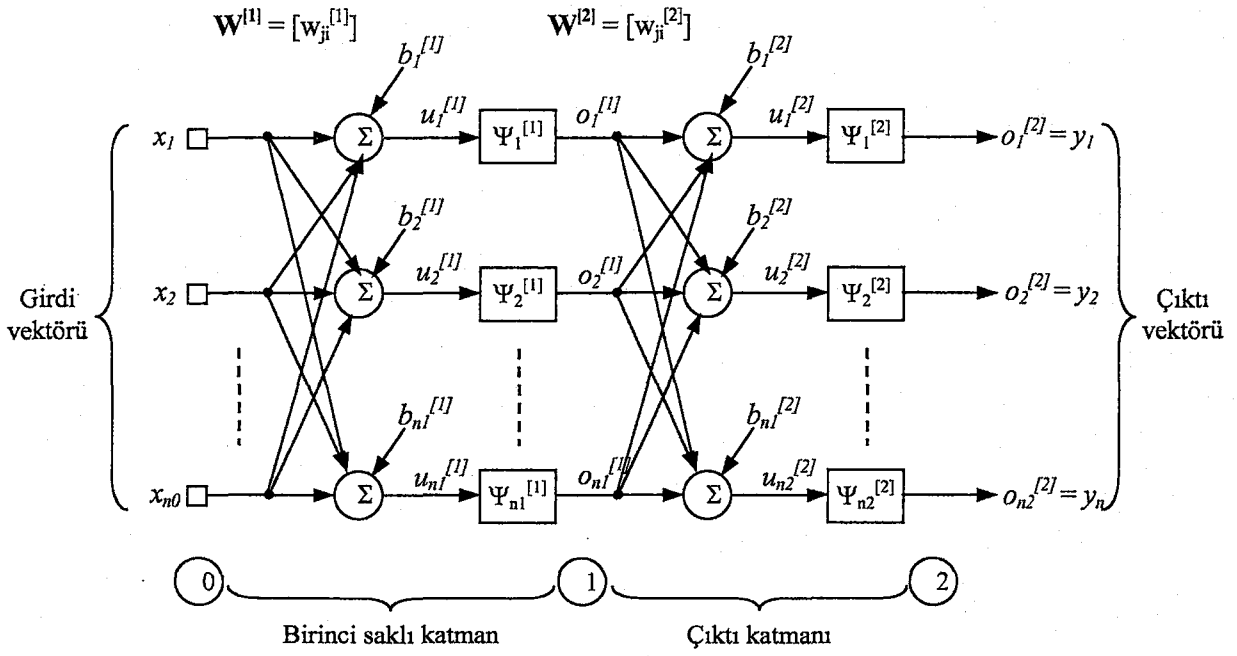
Genel olarak statik örüntü sınıflandırmada, iki saklı katmanlı ÇKİB YSA üniversal bir sınıflayıcıdır. Başka bir deyişle diskriminant fonksiyonları, girdi veri kümelerine bağlı olarak herhangi bir biçimde olabilir. Ayrıca, ağırlıklar uygun bir biçimde standartlaştırıldığında ve çıktı sınıfları (0,1)'e normalleştirildiğinde, sınıflandırma bakış açısına göre en iyi performans gösterir. Ayrıca, eşleştirme yetenekleri açısından ÇKİB YSA'nın herhangi bir fonksiyonu yaklaşık olarak temsil edebileceğine inanılmaktadır (Lapedes ve Farber, 1995).

ÇKİB YSA normalde geriye yayılım algoritması ile eğitilir (Rumelhart, 1986). Widrow tarafından önerilen *en küçük kareler ortalaması* (EKKO) öğrenme algoritması saklı katmanlardaki algılayıcılara uygulanamaz, çünkü saklı katmanlar için istenilen sinyal değerleri bilinmemektedir (Widrow, 1960). Oysa geriye yayılım algoritması, hataların sondan başa doğru yayılmasını sağlayarak saklı katmanlardaki algılayıcıların kendilerini adapte etmelerine izin verir.

Çok katmanlı algılayıcıların iki önemli karakteristiği:

- Doğrusal olmayan işlem elemanlarının eğriliğinin düzgün olması (lojistik fonksiyonu ve hiperbolik tanjant en yaygın olarak kullanılır),
- Çok fazla miktarda birbirine bağlı olma (yani, verilen herhangi katmandaki elemanın bir sonraki katmanda yer alan bütün elemanları beslemesi)

olarak açıklanabilir. Bu çalışmada, statik talep tahmini modeli için kullanılan ÇKİB YSA sinir ağı genel olarak Şekil 6.1'de verilen yapıda oluşturulmuştur.



Şekil 6.1 : Çok katmanlı algılayıcı (ÇKİB YSA) tipindeki yapay sinir ağının yapısı

ÇKİB YSA'nın girdi katmanındaki bazı nöronlar geçmişteki (örneğin, t - p döneminden t - 1 dönemine kadar) satış verilerini gösterir. Saklı katman, çıktı katmanında t dönemindeki satışı gösteren tek bir işlem elemanına bağlı olan birden fazla nörondan oluşur. Girdi işlem elemanı $y = f(x) = x + b$ fonksiyonunu kullanırken, ağdaki doğrusal olmayan elemanlar için hiperbolik tanjant tipinde fonksiyonlar kullanılmıştır (6.1).

$$\psi(net) = \tanh(\alpha.net) = \frac{e^{\alpha \cdot net} - e^{-\alpha \cdot net}}{e^{\alpha \cdot net} + e^{-\alpha \cdot net}} \tag{6.1}$$

Amaç aşağıdaki gibi tanımlanan hata kareleri ortalamasını (J) minimum yapmaktır:

$$J = \frac{1}{N}\sum_{p=1}^{N}\left(d_p - y_p\right)^2 \tag{6.2}$$

Bu formülasyonda d_p, arzu edilen çıktı değeri, y_p ise sinir ağından elde edilen çıktı değeridir. Yukarıda anlatılan yapıdaki ÇKİB YSA'da geçmiş veriler kullanılarak t dönemindeki satışlar ile $(t - p)$ döneminden $(t - 1)$ dönemine kadar olan satışlar arasında bir ilişki kurulabilir.

Modeli test etmede kullanılan girdi verileri 200 ile 600 arasında değiştiğinden, bu değerler yapay sinir ağında kullanılan doğrusal olmayan fonksiyonlar tarafından hissedilmek için büyüktür. Zira tanjant hiperbolik fonksiyonu 2.65'den daha büyük değerler için (0.99 - 1.00) arasında değerler alır. Değişimlerin hissedilebilmesi için [-0.9 , +0.9] aralığında değişen parametreler kullanılarak standartlaştırılması gerekir. Sinir ağına bu dönüştürülmüş değişken öğretildikten sonra ağın çıktılarına da ters dönüşüm (denormalization) işlemi uygulanırsa, sonuçta ağın istenilen değişken değerlerini tahmin etmesi sağlanabilir.

6.2.1 Çevrimdışı Eğitim Metodu

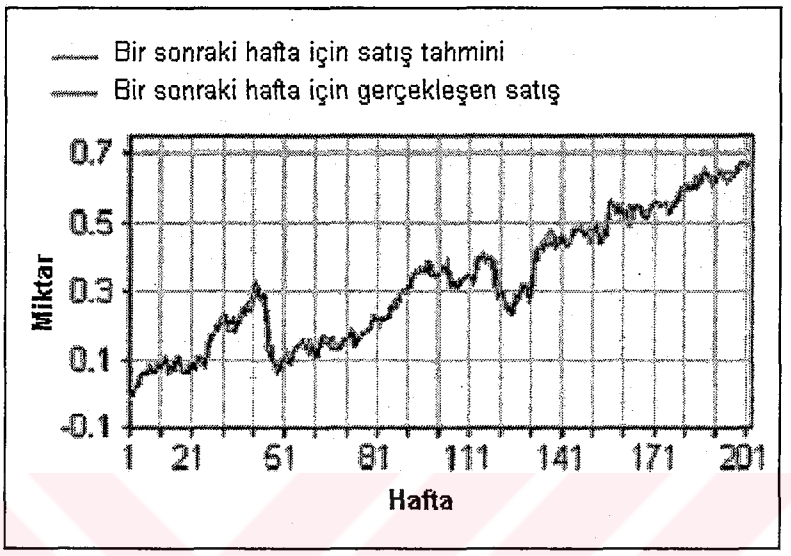
ÇKİB YSA hata düzeltici öğrenme ile eğitilir. Bunun için de sistemin istenen çıktı değerlerinin bilinmesi gerekir. Örüntü tanımada girdi verileri, hangi veri hangi denemeye ait olduğunu gösterecek şekilde etiketlendiğinden, normalde bu değerler bilinir. Çevrimdışı eğitmede sinir ağı daha önce elde edilmiş belirli bir süreç için satış tahminlerini içeren veri kümesi ile eğitilir. Eğitme tamamlandıktan sonra ÇKİB YSA, verileri öğretilen sürece yakın bir andaki değeri genelleme yeteneğini kullanarak tahmin eder.

Daha önce de anlatıldığı gibi geriye yayılım algoritması, ağdaki her bir ağırlığa göre bir maliyet fonksiyonunun duyarlılığını hesaplar ve her bir ağırlığı bu duyarlılıkla orantılı olarak günceller. Bu yaklaşımın güzel yanı, yerel bilgi ile uygulanabilir olması ve yapılan hesaplamanın ekonomikliğidir. GY bir gradyan inme prosedürü olduğundan sadece yerel bilgileri kullanır ve bu yüzden yerel minimuma erişilebilir. Ayrıca, gradyan için kaba bir tahmin kullanıldığından prosedür özünde gürültülüdür ve bu da yavaş yakınsamaya yol açar. Bunu önleyebilmek için geriye yayılım algoritması daha da geliştirilmiş ve yakınsamayı hızlandırıcı pek çok yöntem önerilmiştir. Yapılan bu çalışmada, önerilen bu yöntemler tasarlanan model üzerinde uygulanmış ve performansları karşılaştırılmıştır.

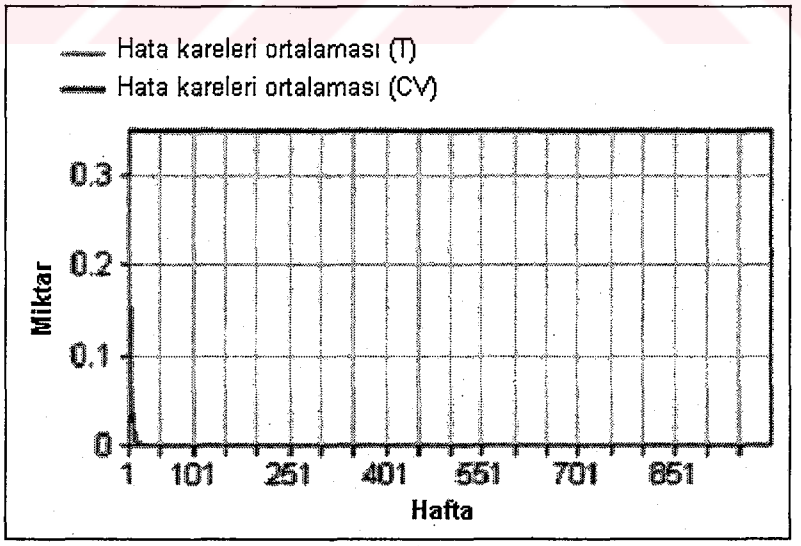
Çevrim dışı eğitmede kullanılan ÇKİB YSA için bu çalışmada belirlenen başlangıç parametreleri şunlardır:

- *Yapay sinir ağının yapısı* : 4-4-1 nöron (bir saklı katmanlı)
- *Öğrenme tekniği* : Geriye Yayılım + MOMENTUM (oran = 0.7)
- *Eğitme oranı* : 0.1
- *Döngü sayısı* : 1000
- *Örnek sayısı* : 253

Aşırı eğitilmeyi önlemek için eğitme kümesi verileri eğitme (T) ve çapraz geçerleme (CV) verileri olarak (%80 - %20) oranında ikiye bölünmüştür. Tek saklı katmanı olan ve toplam 9 sinir ağı elemanı içeren ÇKİB YSA 1000 döngü çalıştırılarak eğitilmiştir. Eğitme işlemi sonucunda modelin öğrenmedeki başarısı Şekil 6.2'de görülmektedir.

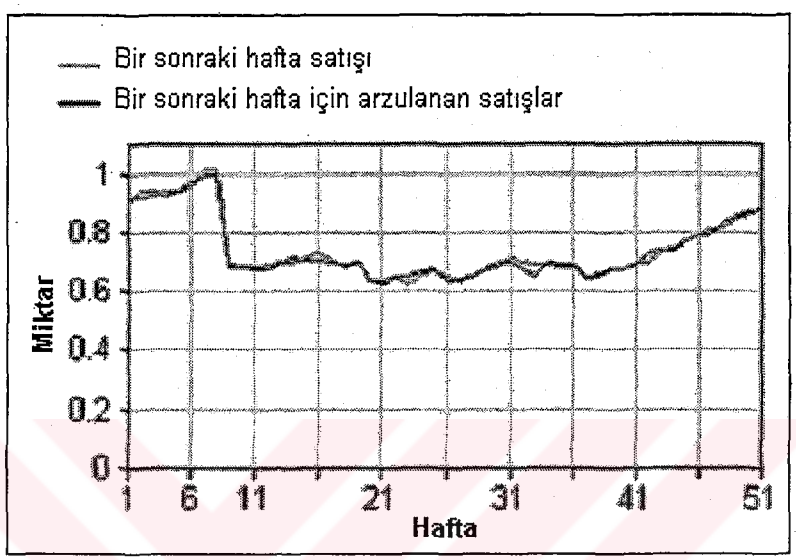


Şekil 6.2 : Haftalık satış miktarının çevrimdışı eğitme metodu kullanılarak öğrenilmesi



Şekil 6.3 : Eğitme ve çapraz geçerleme periyotları için etkin hata değerlerinin karşılaştırılması

Sinir ağının öğrenme eğrisinde (Şekil 6.3), eğitme periyodu ile çapraz geçerleme periyodu karşılaştırılmış ve ağın genelleme başarısı öngörülmeye çalışılmıştır. Çapraz geçerleme periyodunda ağın öğrenme seviyesi Şekil 6.4'de görülmektedir. Yukarda belirtilen parametrelere göre tasarlanan sinir ağı için eğitme periyodu sonunda elde edilen en iyi hata değeri 0.000162, çapraz kontrol periyodu içinse 0.000385 olarak bulunmuştur.

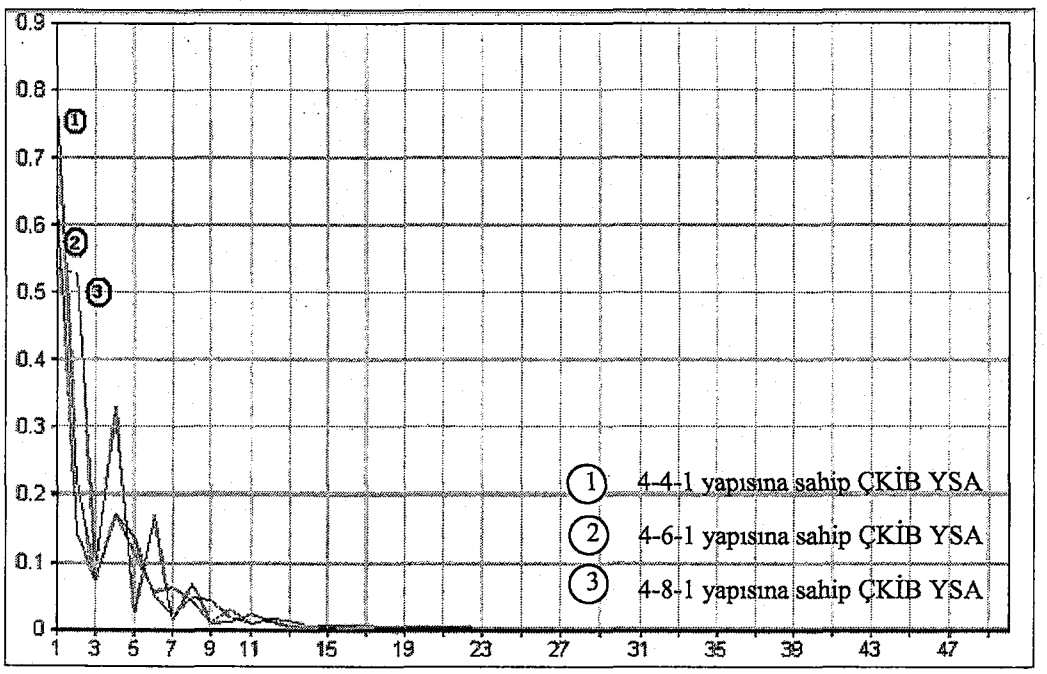


Şekil 6.4 : Çapraz geçerleme sonuçları ile gerçek değerlerin karşılaştırılması

6.2.1.1 Saklı Katmandaki Sinir Hücre Sayısının Öğrenme Hatasına Etkisi

Yapay sinir ağında yer alan sinir hücresi sayısı öğrenme hatasını dolayısıyla da YSA'nın performansını doğrudan etkiler. Sanılanın aksine, daha önce de belirtildiği gibi algılayıcı sayısının çok olması veya saklı katman sayısının fazlalığı (>2) sinir ağının çok iyi öğrenmesine yol açmaz. Aynı şekilde, az olması da her zaman iyi sonuç vermeyebilir (Bkz. Bölüm 3.4.5.4). Üzerinde çalışılan problem için en iyi sonuç veren katman ve sinir hücresi sayısını bulabilmek için farklı parametrelerle test yapmak (duyarlılık analizi) gerekir.

Yukarıdaki model için yapılan sınırlı sayıdaki test sonuçlarına göre 4-4-1 yapısına sahip sinir ağı daha kararlı olarak öğrenmekle birlikte, aslında denenen üç farklı sinir ağı da arzu edilir seviyede ve zamanda eğitilmiştir (Şekil 6.5). En uygun parametreleri bulabilmek için daha farklı yapıdaki sinir ağları ile daha fazla testler yapılabilir ve Bölüm 3.4.5'de anlatılan yöntemlerden (budama, ağırlık eleme) veya MacKay'ın Bayesgil çatısına dayanan yönteminden de faydalanılabilir.



Şekil 6.5 : Eğitme döngüsü sayısının ve sinir hücresi sayısının öğrenme hatasına etkisi

6.2.1.2 Modelin Farklı Öğrenme Algoritmaları ile Test Edilmesi

Verilen bir problem için hangi öğrenme algoritmasının daha hızlı olduğunu (yakınsadığını) bilmek oldukça güçtür. Çünkü öğrenme hızı, problemin karmaşıklığı, eğitme kümesindeki veri sayısının miktarı, sinir ağında yer alan sinir hücresi ve ağırlıkların sayısı, hedeflenen hata, sinir ağının hangi tip problem için kullanıldığı gibi pek çok faktöre bağlıdır. Bu bölümde, farklı öğrenme algoritmalarını kıyaslamak için karşılaştırmalar yapılmıştır. Testlerde, saklı katmandaki sinir hücrelerinde *tanh* transfer fonksiyonu ve çıktı katmanında doğrusal transfer fonksiyonu kullanan, 4-4-1 mimarisine sahip bir ÇKİB YSA kullanıldı. Aşağıdaki tabloda, beş farklı öğrenme algoritması için elde edilen sonuçlar verilmiştir. Tabloda yer alan her bir yöntem için, rassal başlangıç ağırlıkları kullanılarak, diğer tüm parametreler aynı kalmak şartıyla 20 farklı deneme (replikasyon) yapılmıştır. Her birisinde, ağ hata kareleri ortalaması 0.002'den küçük oluncaya kadar eğitilmiştir (Eşik değer = 0.002). Testler için kullanılan bilgisayar Microsoft XP işletim sistemine sahip Pentium III-800'dür.

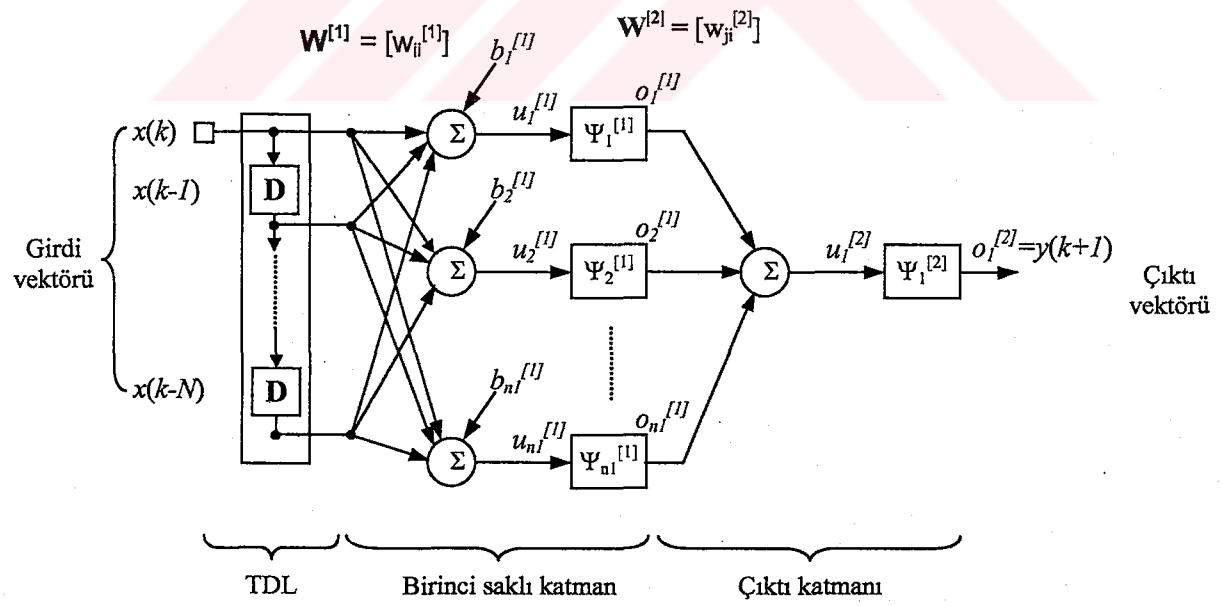
Tablo 6.1 : Öğrenme algoritmalarının çevrimdışı ÇKİB YSA için karşılaştırılması

Algoritma	Ortalama Süre (sn)	Oran	Minimum Süre (sn)	Maksimum Süre (sn)	Std. Sapma
Delta-Bar-Delta	16.20	1.00	15	17	0.62
Geriye yayılım+Momentum	16.55	1.02	15	19	0.94
Quickprop	16.80	1.04	16	19	0.89
Conjugate Gradyan	16.85	1.04	16	19	0.99
Geriye Yayılım	19.30	1.19	15	41	5.91

Tablo 6.1'de görüldüğü gibi, öğrenme algoritmalarının bu modeldeki performansı birbirlerine çok yakın olmakla birlikte geriye yayılım en kötü sonucu vermiştir.

6.2.2 Çevrimiçi Eğitim Metodu

Bu yöntemde yapay sinir ağının kullanılma amacı, yapay sinir ağı bazlı kontrol algoritmalarında kestireç olarak kullanılan ağ ile aynıdır. Sistemin yakın geçmişteki birkaç örnekleme değeri ağa öğretilerek bir ya da birkaç periyot sonraki örnekleme noktasında alacağı değeri kestirmesi istenir (Şekil 6.6).



Şekil 6.6 : Çevrimiçi öğrenme için kullanılan TDL içeren ÇKİB YSA

Bu yöntemde, testi gerçekleştirebilmek için, parametresi (satış miktarı) tahmin edilmek istenen sürecin başlangıçtaki değerlerinin bilinmesi gerekir. Tam olarak kaç noktadaki değerin gerektiği (N) kullanılmaya karar verilen veri kümesine göre değişebilir. Haftalık satış miktarının belli bir süreçteki değişimi tahmin edilmek isteniliyorsa, işlemin bu süreç içerisindeki her örnekleme noktası için tekrar edilmesi gerekir. Her örnekleme noktasında sinir ağını önceki birkaç noktanın değerleri ile eğitmek gerektiğinden ve verilerin tek tek geldiği kabul edildiğinden, bu yönteme çevrimiçi eğitme metodu denmiştir. TDL-ÇKİB YSA modelini eğitmek için kullanılan veri kümesi aşağıdaki gibidir:

$$\left\{\left[\begin{pmatrix} x(k-1) \\ x(k-2) \\ \vdots \\ x(k-N) \end{pmatrix}, y(k)\right], \left[\begin{pmatrix} x(k-2) \\ x(k-3) \\ \vdots \\ x(k-N-1) \end{pmatrix}, y(k-1)\right], \ldots, \left[\begin{pmatrix} x(k-n) \\ x(k-n-1) \\ \vdots \\ x(k-n-N+1) \end{pmatrix}, y(k-n+1)\right]\right\} \quad (6.3)$$

Burada $x(k)$ haftalık satış miktarını, N girdi vektörünün boyutunu (tapped delay line), n ise veri kümesindeki girdi-çıktı çiftlerinin sayısını göstermektedir. Bu küme satış miktarının gelecekte alacağı değerin geçmişteki değerleriyle olan ilişkisini öğretir. (6.3)'deki veri kümesi TDL-ÇKİB YSA modeline öğretildikten sonra, içinde bulunulan an k ile gösterilirse, bir sonraki örnekleme anındaki (k + 1) satış miktarının alacağı değeri tahmin etmesi için şu vektör girilir:

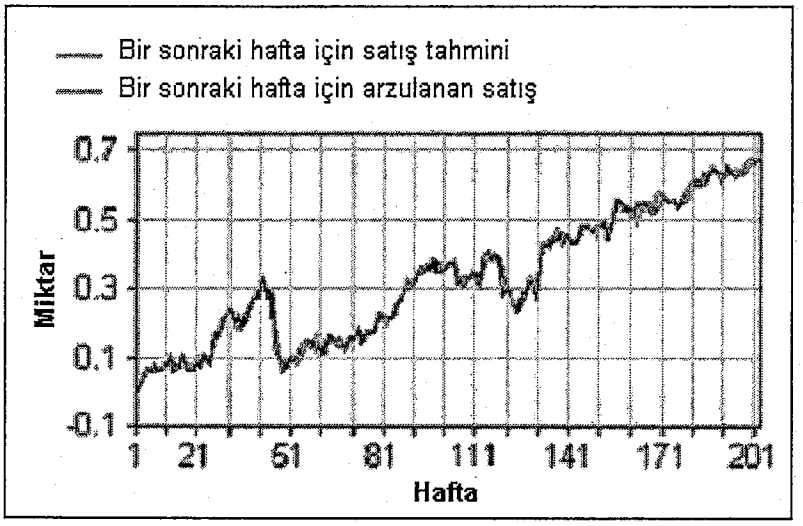
$$\left(x(k), x(k-1), \ldots, (x-N+1)\right)^T \quad (6.4)$$

TDL-ÇKİB YSA modeli, (6.3)'deki veri kümesinden satış miktarlarının ardışık n değeri girildiğinde, parametrenin bir sonraki örnekleme noktasında alacağı değeri öğrenmiştir. Öğrendiği bilgilerden genelleme yaparak (6.4)'de girilen vektöre karşılık gelen (k + 1) anındaki değeri tahmin eder.

Çevrim içi eğitme kullanılan TDL-ÇKİB YSA için belirlenen başlangıç parametreleri şunlardır:

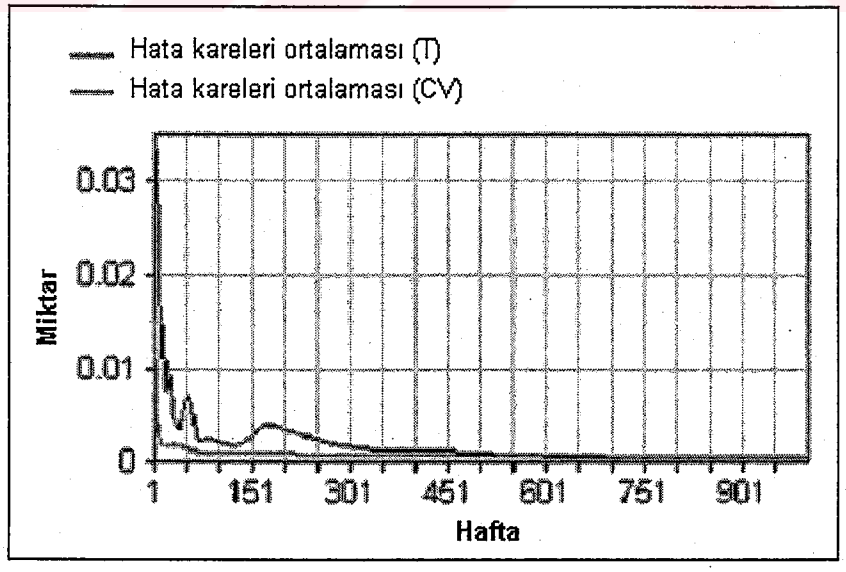
- *Yapay sinir ağının yapısı* : 4-4-1 nöron (bir saklı katmanlı)
- *Geçmiş veri sayısı* : 3
- *Öğrenme tekniği* : Geriye Yayılım + MOMENTUM (oran = 0.7)
- *Eğitme oranı* : 0.1
- *Döngü sayısı* : 1000

Tek saklı katmanı olan ve katmanlarında 4-4-1 adet işlem elemanı yer alan TDL-ÇKİB YSA sinir ağı 1000 döngü çalıştırılarak eğitildi. Eğitme işlemi sonucunda modelin öğrenmedeki başarısı Şekil 6.7'de görülmektedir.

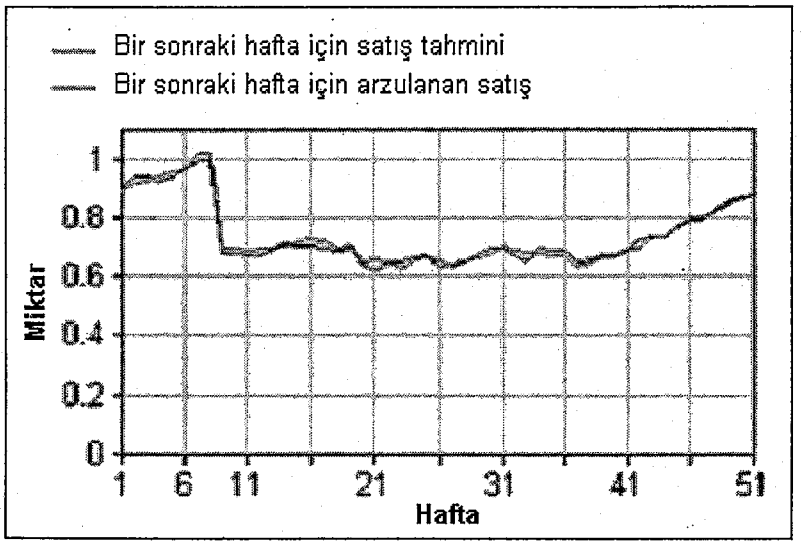


Şekil 6.7 : Haftalık satış miktarının çevrimiçi eğitme metodu kullanılarak öğrenilmesi

Şekil 6.8'de sinir ağının öğrenme eğrisine çizilerek, eğitme periyodu ile çapraz geçerleme periyodu karşılaştırılmış ve ağın genelleme başarısı öngörülmeye çalışılmıştır. Çapraz geçerleme periyodunda ağın öğrenme seviyesi Şekil 6.9'da görülmektedir.

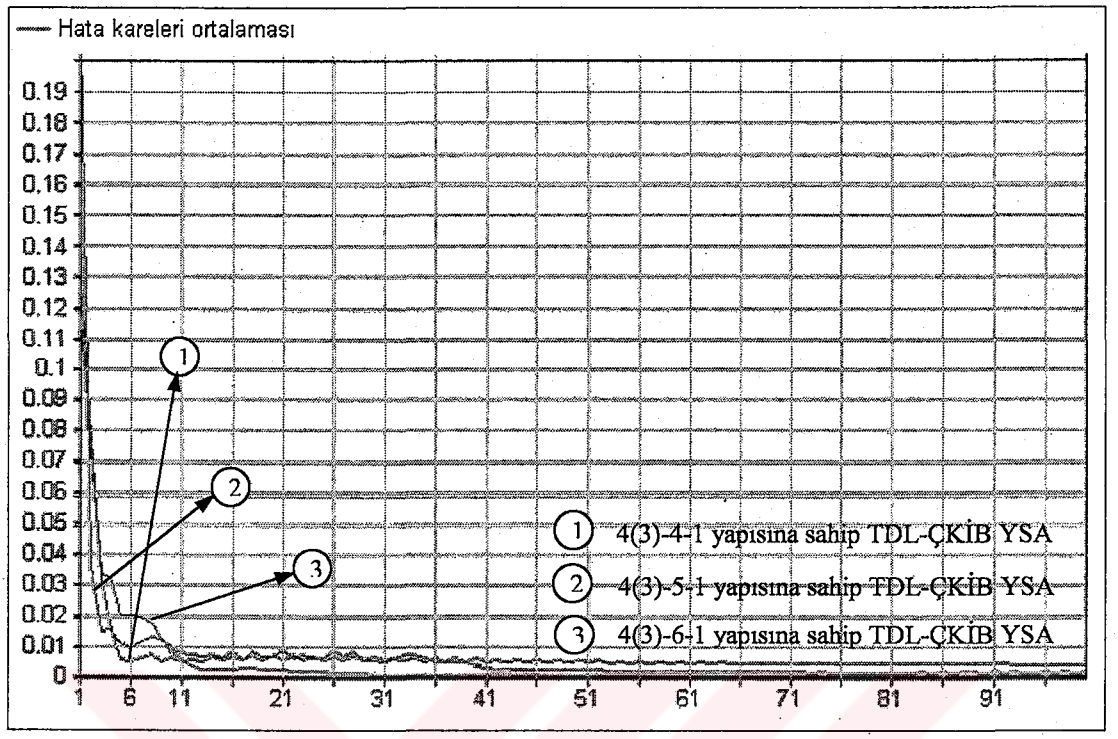


Şekil 6.8 : Eğitme ve çapraz geçerleme periyotları için etkin hata değerlerinin karşılaştırılması



Şekil 6.9 : Çapraz geçerleme sonuçları ile gerçek değerlerin karşılaştırılması

Yukarıda belirtilen parametrelere göre tasarlanan sinir ağı için ortalama 5.17 dakikada elde edilen eğitme periyodu sonunda elde edilen en iyi hata değeri 0.000646, çapraz kontrol periyodu içinse 0.001293 olarak bulunmuştur. Hata değerleri çevrimdışı eğitme metodu için elde edilen değerlerle karşılaştırıldığında daha yüksektir. Aslında bu da beklenen bir durumdur, zira çevrimdışı eğitilen sinir ağlarının daha iyi yakınsadıkları literatürde yer almaktadır (Bishop, 1996; Principe, 2000). TDN-ÇKİB YSA'da sinir hücresi sayısı ve döngü sayısının öğrenme hatası üzerine etkisi incelendiğinde, Şekil 6.10'daki sonuç elde edilmiştir. İlk altı döngü sonucunda *hata kareleri ortalamasında* hızlı bir düşüş söz konusuyken yaklaşık kırkıncı döngüden sonra hemen hemen sabitlenmiştir. Şekil 6.10'da da görüldüğü gibi, ağ yapısı büyüdükçe beklenenin aksine hata kareleri ortalaması yükselmektedir. Bu da daha önce öngörüldüğü gibi, ağ yapısının büyüklüğü ile elde edilen sonuçların kalitesi arasında doğrusal bir ilişki bulunmadığı savını doğrulamaktadır. Ancak, belirli bir döngü sayısından sonra farklı yapıdaki ağlar yaklaşık olarak aynı değere yakınsamıştır.



Şekil 6.10 : Eğitme döngüsü sayısı ile sinir hücresi sayısının öğrenme hatasına etkisi

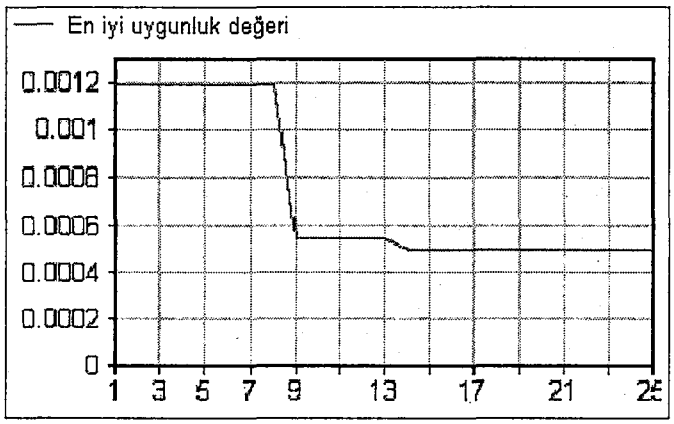
6.3 Genetik Eniyileme

Genetik eniyileme kullanılarak en uygun ağ parametre değerleri (öğrenme oranı, adım büyüklüğü, sinir hücresi sayısı, vs.) belirlenebilir (Bkz. Bölüm 3.5). Bu yaklaşımda, farklı parametre kombinasyonları denenip uygunluk değerine göre test edilerek iyi parametreler bir popülasyondan diğerine geçer. Farklı parametrelerle ağ tekrar tekrar eğitildiğinden ve her birisi için en iyi HKO hesaplandığından, sinir ağını genetik eniyileme ile eğitmek için çok fazla zamana gereksinim duyulur.

Farklı düzeylerde genetik eniyilemeden faydalanılabilir. Yani sadece öğrenme oranlarını veya hem öğrenme oranlarını hem de her bir katmandaki sinir hücresi sayısını belirlemede genetik eniyileme kullanılabilir.

Bu bölümde, TDL-ÇKİB YSA modelinde ağ parametrelerini eniyilemek için genetik algoritmalar kullanarak elde edilen etkin hata değerinin daha önceki sonuçlarla karşılaştırması yapılacaktır. Genetik eniyileme iki farklı ağ parametre grubu için tekrarlanmıştır ve başlangıç parametreleri olarak:

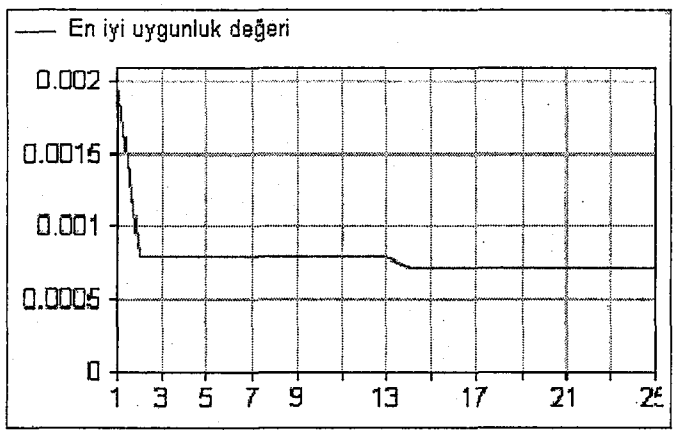
- *Yapay sinir ağının yapısı* : 4(3)-4-1 nöron (tek saklı katmanlı)
- *Popülasyon sayısı* : 25
- *Çaprazlama tipi* : tek-noktadan
- *Çaprazlama olasılığı* : 0.9
- *Mutasyon olasılığı* : 0.01



Şekil 6.11 : Öğrenme oranları için elde edilen en iyi uygunluk değerleri

Sadece öğrenme oranları için (adım büyüklüğü ve momentum oranı) genetik eniyileme kullanıldığında elde edilen en iyi uygunluk değerleri Şekil 6.11'de görülmektedir. Yine bu modelde çapraz geçerleme için elde edilen HKO değeri 0.000937 olup daha önceki değerden (0.001293) düşüktür.

Yine aynı model üzerinde hem öğrenme oranları hem de sinir hücresi sayısı için genetik eniyileme kullanıldığında elde edilen en iyi uygunluk değerleri Şekil 6.12'de verilmiştir.



Şekil 6.12 : Öğrenme oranları ve sinir hücresi sayıları için elde edilen en iyi uygunluk değerleri

6.4 Değerlendirmeler

Yapılan testler sonucunda ÇKİB YSA'nın, içerisinde açıklanamayan veriler de içerebilen, genellikle kararsız bir yapı gösteren veya deseni istatistiksel tekniklerle kolaylıkla açıklanamayan talep tahminleri için başarıyla kullanılabileceği görülmüştür. Gerçek zamanlı talep tahmini sistemlerinde de TDL-ÇKİB YSA modelinin çevrim içi eğitilenler kadar olmasa da arzu edilen düzeyde iyi sonuçlar üretebildiği bulunmuştur.

Ayrıca, incelenen sinir ağı modeli için, eğitme zamanını azaltmaya yarayacak ve daha iyi performans göstermesini sağlayacak bulgular aşağıda verilmiştir:

1. Eğitme verilerinin standartlaştırılması
2. Lojistik fonksiyonu yerine tanh fonksiyonunun kullanılması
3. Adım büyüklüğünün girdiye doğru artırılması: Yani, bir saklı katman içeren ÇKİB YSA için, girdi ile saklı katman arasındaki ağırlıklar için adım büyüklüğünün 0.05 ve çıktı katmanı ile saklı katman arasındakinin ise 0.01 olarak belirlenmesi.
4. Daha üstün öğrenme yöntemlerin kullanılması (Reslient BP, Levenberg-Marquardt, Quickprop gibi).
5. Ağırlık sayısından daha fazla eğitme örüntüsüne sahip olunması: Daha önceki bölümlerde de açıklandığı gibi test kümesinde ÇKİB YSA'nın performansı N >W/e ile sınırlanmıştır (N, eğitme döngü sayısı, W, ağırlıkların toplam sayısı, e, performans hatası). Bu yüzden, hata kareleri ortalaması e/2'den düşük oluncaya kadar eğitmeye devam etmek gerekir.

7. TEDARİKÇİLERİN SINIFLANDIRILMASININ YAPAY SİNİR AĞLARI İLE GERÇEKLENMESİ

Bu bölümde, lojistik sistemlerinde tedarikçilerin sınıflandırılması işlemi için, genel olarak Çok Katmanlı İleri Beslemeli Yapay Sinir Ağları, özellikle kümeleme analizi için Kohonen ağları tasarlanmış, ÇKİB YSA'ya göre çok daha hızlı yakınsayan, gözetimli ve gözetimsiz öğrenmeyi birleştiren karşıt yayılım ağları da modellenerek test verileri için elde edilen sonuçlar karşılaştırılmıştır.

7.1 Giriş

Tedarikçiler üretim zincirinde ve dolayısıyla bir şirketin uzun dönemde ayakta kalmasında can alıcı bir rol oynarlar. Tedarik fonksiyonunun sorumluluğu, çoğu zaman yeterli kalite ve miktarda, uygun fiyata, uygun bir teslimatla hammaddenin, teçhizatın ve malzemenin tedariki olarak tanımlanır (Tam ve Tummala, 2001).

Geleneksel olarak tedarikçinin seçimi ve değerlendirilmesi, geç teslim zamanı, üretimdeki aksamalar, teslim edilen ürünlerin kalitesinin düşüklüğü gibi diğer önemli endirekt tedarikçi maliyetleri ihmal edilerek en düşük maliyet faturalı tedarikçiye dayandırılarak yapılırdı.

Son yıllarda müşteri taleplerindeki artış, teknolojik olanaklar ve yenilikler ile düzensiz piyasa trendlerine bağlı olarak endüstride bir çok yenilikler, gelişmeler ve değişimler olmuştur. Teknolojik değişimlere bağlı olarak müşterilerin ihtiyaçları gelişmiş ve daha düşük fiyat ve daha yüksek kaliteyi aynı zamanda talep etmeye başlamışlardır.

Tedarikçilerle sadece fiyat üzerinde pazarlık etmek yerine neden onlarla bütünleşmek gerekir? Çünkü fiyatı belirleyen yegâne bileşen maliyettir ve maliyetleri azaltmak da bütün bileşenlerinin kontrol altına alınmasından geçer. Dağıtımdaki gecikmeler dışardan temin edilen parçaların uygunsuzluğu, sıklıkla karşılaşılan ve üretimde kalitesizliğe yol açan problemlerdir. Çok düşük maliyetli bir parça, eğer üretim hattının durmasına neden oluyorsa ağır bir maliyete yol açabilir. Yenilikçi ve kaliteli ürünler üretmek, rekabetçilik ve zamanında teslim ölçütlerine bağlıdır ve genellikle kalite ve rekabetçilik tedarikçilerin kalitesi ve rekabetçiliği ile doğrudan

ilişkilidir. Dolayısıyla tedarikçilerin seçimi ve onlarla bütünleşme büyük önem arz eder. Tedarikçiler sadece satın alma bölümüne karşı sorumlu değildir. Tedarikçilerle bütünleşme, bir fiyat için pazarlık yapıp siparişin gerçekleştirilmesinden çok daha fazla anlam ifade eder. Bütünleşme bütün alanları etkiler: Ar-Ge, Endüstri Mühendisliği, kalite, üretim, finans, vs. Toplam kaliteyi ve mümkün olan en düşük maliyeti gerçekleştirebilmek için böyle bir bütünleşme kaçınılmazdır.

7.2 Tedarikçi Seçim Modellerinin Sınıflandırılması

Bir şirketin uygun bir tedarikçi olup olamayacağını değerlendirmede tek bir ölçüt yeterli değildir. Bir firma kalite düzeyi açısından mükemmel olabilir ve kusursuz parçaların dağıtımını gerçekleştirebilir. Ancak, bu mükemmellik gelişme yeteneği olmayan eskimiş süreçlerle bağlantılı olabilir. Benzer şekilde, tatmin edici düzeyde karlı gözükenler gerçekte ana sermaye veya varlıklar açısından gerçekteki zayıflıklarını gizleyebilir, bu da ilerde problemlere yol açar. Bu yüzden tedarikçi seçiminde maliyet, kalite, performans, teknoloji vb. birçok ölçüt göz önünde bulundurulması gerekir. Sadece malzeme maliyeti değil aynı zamanda işletme maliyetleri, bakım, geliştirme ve destekleme maliyetleri de bu seçimde göz önünde bulundurulması gereken unsurlardır. Bundan dolayı ekonomiklik ve performans ile ilgili ölçütler arasında sistematik bir satıcı seçim sürecini elde etmede kullanılmak üzere ölçütlerin değerlendirilip öncelik sırasına konulmasına gereksinim duyulur. Bu süreç aynı zamanda hem seçim sürecini kısaltacak hem de karar vermede başarıyı artıracaktır.

Diğer faktörleri de dikkate alacak birçok performans göstergesini tek bir puan olarak hesaplayan ve derecelendirme modelleri olarak adlandırılan birçok alternatif yöntem önerilmiştir. Bunlardan en basiti farklı tedarikçi karakteristiklerini "iyi", "yeterli", "nötr" ve "yetersiz" gibi sıralayan *Kategorik Yöntemidir* (Timmerman, 1986). En yaygın olarak kullanılan yaklaşım ise, bir çok ölçüte farklı ağırlıklar vererek en yüksek tartılandırılmış toplam skora sahip tedarikçinin seçildiği *Tartılandırılmış Nokta Planı*'dır (Gregory, 1986). Bir diğer teknik ise tedarikçilerin verilen bir ölçüte göre göreceli pozisyonlarının ikili karşılaştırmalarla belirlendiği *Analitik Hiyerarşi Süreci* yöntemidir (Barbarosoğlu ve Yazgaç, 1997).

Tedarikçi seçiminde kullanılan ölçütlerden başlıcaları Tablo 7.1'de verilmiştir (Taylor, 1997). Bu ölçütler kurulan yapay sinir ağının girdi vektörünü oluşturur. Başka bir

deyişle kurulan YSA modelinin girdi katmanı, her birisi Tablo 7.1'deki bir ölçütle ilgili olan 10 sinir hücresinden (işlem elemanı) oluşmaktadır.

Tablo 7.1 : Tedarikçi seçimi için belirlenen ölçütler

S/N	Nitelik	Tanımı
1	Fiyat	Satın alınacak parça veya projenin ayrıntılı maliyeti
2	Teslimat	Teslimat için temin süresi ve koşulları
3	Kalite	Tedarikçiden beklenen kalite düzeyi (genellikle hata oranı cinsinden ifade edilir)
4	İnnovasyon	Tedarikçi ne kadar yenilikçi? Olağan koşullarda yeni ve orijinal fikirler üretebilmekteler mi?
5	Teknoloji düzeyi	Tedarikçi/alıcı teknolojik olarak ne kadar yeterli? Yeterli düzeyde entegrasyonu sağlayabilecek sistemlere sahipler mi? Pazarda lider, izleyici, vs. mi? Kilit teknolojilere sahipler mi (mesela, kompozit teknolojilerinde yetkin olma)
6	Kültür	Neyi severler? Güvenilir miler? Bu insanlarla daha yakından çalışılabilir mi?
7	Ticari tecrübe	Anlaşmalarda ne kadar iyidirler? Kontrat ile ilgili düzenlemelerde katımıdırlar veya daha basitleştirilmiş düzenlemelere mi eğilimlidirler?
8	Üretim esnekliği	Tedarikçi/alıcıların mevcut ve öngörülen kapasite yükleri nedir? Teslimat programındaki değişimleri veya artan sipariş miktarları ile başa çıkabilecekler mi?
9	İletişim kolaylığı	Bilgilerin serbestçe akışına izin verecek şekilde, yeterli düzeyde iletişim sistemlerinin entegrasyonu gerek işletme içerisinde gerekse işletmeler arasında sağlanmış mı?
10	Mevcut itibarı	Pazardaki mevcut itibarları nedir? Finansal açıdan istikrarlılar mı?

Bu ölçütlerin büyük bir kısmı (ilk üç ölçüt sayısal) sembolik/kategorik verilerdir. Bu ölçütlere ait veriler sinir ağında girdi olarak kullanılırken bir ön işlemden geçirilerek sayısallaştırılması gerekir.

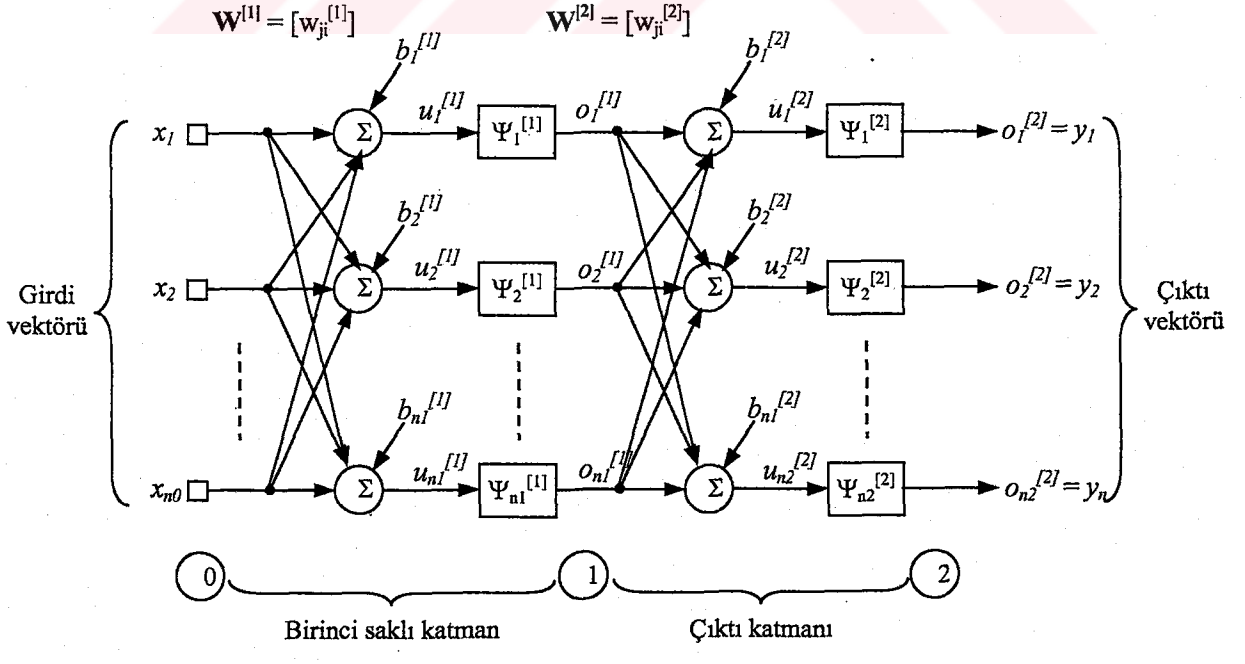
Tedarikçilerin, yukarıdaki tabloda belirtilen veya işletmelerin kendi yapılarına özgü belirledikleri ölçütlere göre yapay sinir ağları kullanılarak nasıl sınıflandırılabileceği ya da aynı ölçütlere göre farklı gruplara (A, B, C gibi kümeler) nasıl ayrıştırılabileceği ilerleyen bölümlerde incelenecektir. Sınıflandırma veya kümeleme analizi sayesinde işletmelerin tedarikçileri önceliklendirmesi ve hangi mal / ham maddenin hangi tedarikçiden temin edileceğini daha sağlıklı olarak yapabilmesi mümkün olacaktır. Zira YSA çok boyutlu girdi vektörünü oluşturan değerlendirme ölçütlerini iki boyuta indirgeyip görselleştirerek karar verme sürecini daha kesin hale getirebilir.

Sınıflandırma problemleri için genel olarak ÇKİB modeller kullanılsa da özellikle kümeleme analizinde Kohonen ağları çok iyi sonuçlar vermiştir (Kohonen, 1995). Yapay sinir ağının girdi katmanı, her birisi Tablo 7.1'deki bir kriterle ilgili olan 10 sinir hücresinden (işlem elemanı) oluşturulabilir. Çıktı katmanı ise verinin hangi sınıfa ait olduğunu belirleyecektir.

7.3 Sınıflandırma Modeli Olarak Çok Katmanlı İleri Beslemeli Sinir Ağı

Çoklu-grup sınıflandırma probleminde çıktı katmanında *softmax* aktivasyon işlevini kullanan çok katmanlı algılayıcılardan oluşan bir ağ kullanılacaktır. Softmax kullanmanın avantajı, ağ HKO ölçütüne göre eğitildiği zaman çıktıların olasılıklar olarak yorumlanabilmesidir. Softmax, işlem elemanını besleyen aktivitelerin toplamını bir olmaya zorlayan etkili bir rekabetçi yapıdır. Bu aktivitelerin normalleştirilmesinin rekabet olarak yorumlanabileceğinin farkına varılması önemlidir. Çünkü eğer bir işlem elemanı (sinir hücresi) daha büyük bir çıktı üretirse toplam değer sınırlandırıldığından, diğerleri daha düşük olmaya zorlanacaktır.

Başka bir bakış açısına göre, sınırlı kaynakların da rekabeti empoze edeceği söylenebilir. Rekabet prensibi, pek çok sistemde ve istatistikte dolaylı olarak kullanılır. Ancak softmax ile bu prensip daha açık hale getirilmiştir ve rekabete dayalı olarak öğrenen ağlar geliştirilebilir.



Şekil 7.1 : Çıktı katmanında softmax fonksiyonu kullanan ÇKİB YSA

ÇKİB YSA yaklaşımı ile modelleme yapıldığında, yapay sinir ağı (Şekil 7.1), 10 tane işlem elemanından oluşan bir girdi katmanı ile bir saklı katman ve iki sinir hücresi içeren çıktı katmanından oluşacak biçimde tasarlanmıştır. Sınıflandırıcıların performansı genellikle sınıflandırma hatası açısından ölçülür. Sınıflandırıcının doğruluğu sınıflandırma hatasının 1'den çıkarılması ile bulunur. Bu nedenle tam olarak yanlış sınıflandırma sayılarını gösteren düzensizlik matrisini (confusion matrix) kurmak daha iyi bir yaklaşımdır. Düzensizlik matrisi sınıflandırıcının çıktılarının doğru sınıflandırma ile karşılaştırıldığı bir tablodur. Performans ölçütü olarak HKO ölçütü ve yanlış sınıflandırmayı ölçmek için düzensizlik matrisi kullanılmıştır. Saklı katmanda kullanılan aktivasyon fonksiyonu ise lojistik sigmoid fonksiyonudur.

Sinir ağının çıktısı sıklıkla bir benzerlik ölçütü üretir. Bu benzerlik ölçütünü bir olasılığa dönüştürmek için ağın çıktı katmanında softmax aktivasyon fonksiyonu (Φ) kullanan iki adet sinir hücresi yer almaktadır.

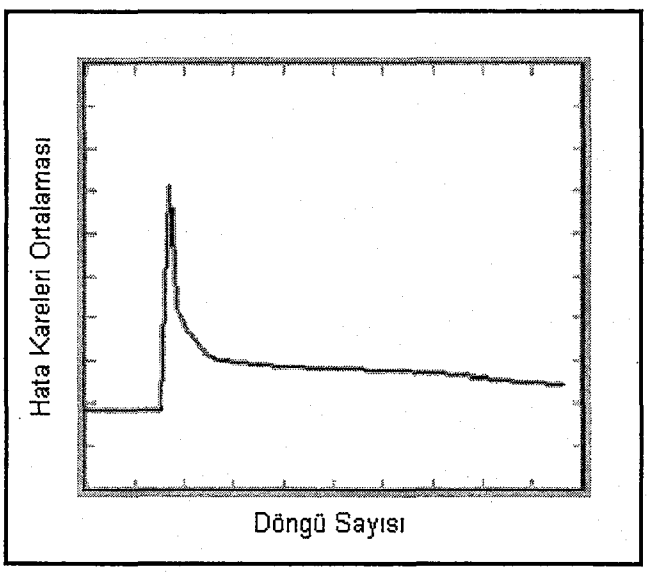
$$f(x_i, w_i) = \frac{\exp\left[x_i^{lin}\right]}{\sum_i \exp\left[x_i^{lin}\right]} \tag{7.1}$$

$$x_i^{lin} = \beta x_i$$

Kullanılan ÇKİB YSA için belirlenen başlangıç parametreleri şunlardır:

- Yapay sinir ağının yapısı: 10-5-2 sinir hücresi (tek saklı katmanlı)
- Öğrenme tekniği: Geriye Yayılım + MOMENTUM (oran = 0.7)
- Eğitme oranı: 0.2
- Öğrenme tekniği: Çevrimiçi öğrenme
- Döngü sayısı: 50
- Örnek Sayısı: 50

Gerçekleştirilen test neticesinde elde edilen eğitici küme performansı ile hata kareleri ortalaması Şekil 7.2'de görülmektedir.



Şekil 7.2 : Yapay sinir ağı modelinin eğitici küme performansı

Tablo 7.2'de görülen düzensizlik matrisi sonuçlarına göre, yanlış sınıflandırılan veri sayısı 3 (% 6) olarak görülmektedir. Test neticesinde eğitim periyodunun sonunda elde edilen hata kareleri ortalaması değeri 0.012518 olarak bulunmuştur.

Tablo 7.2 : Yapılan sınıflandırmanın düzensizlik matrisi

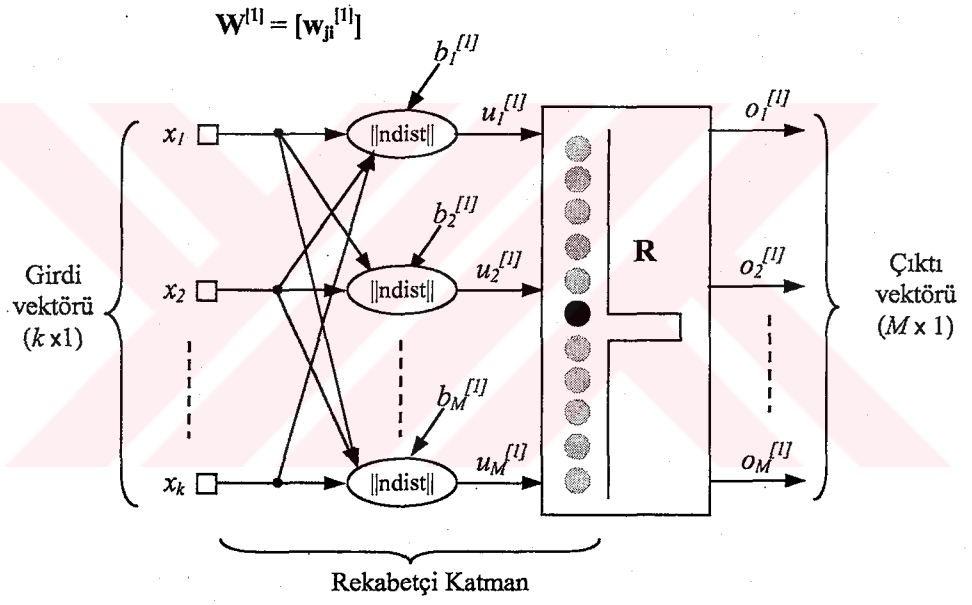
	1 nci Sınıf	**2 nci Sınıf**
1 nci Sınıf	22	1
2 nci Sınıf	2	25

Öğrenme eğrisi öğrenmenin gelişimini gözlemlemek için önemli bir göstergedir. Ancak, eğitim veya test kümelerindeki HKO, sınıflandırma performansının sadece endirekt bir ölçümüdür. HKO, standartlaştırılmasına ve girdi verileri ile istenilen verilerin karakteristiklerine bağlıdır. Bir sınıflayıcının performansı sınıflandırma hatası ile ölçülür. Sınıflayıcının doğruluğu (1 - *sınıflandırma hatasına*) eşittir. Bu nedenle yanlış sınıflandırmanın sayısını ölçen düzensizlik matrisini kullanmak daha iyi bir yaklaşımdır. Gerçekleştirilen testte döngü sayısı artırıldığında HKO azalmakla birlikte düzensizlik matrisinde yanlış sınıflandırılan veri sayısı artmaktadır. Bu sonuç, öngörülen HKO'nun sınıflandırma için iyi bir ölçüt olmadığı düşüncesini doğrulamaktadır.

7.4 Kohonen Ağları ile Tedarikçilerin Kümelenme Analizi

Kendi kendini düzenleyen ağlar, önceden eğitilmeye gereksinim duymaksızın girdi verilerindeki düzenlilik ve ilişkileri keşfederek, girdilere göre sonraki tepkilerini adapte eder. Bu tip sinir ağlarında yer alan işlem elemanları benzer girdi vektörlerinin oluşturduğu grupları tanımayı öğrenir. Bu nedenle, özellikle girdi verilerinin kaç gruptan elde edildiğinin bilinmemesi veya çok fazla miktarda verinin elde bulunması ve bu veriler arasında gruplandırmaya gereksinim duyulmasında (gözetimsiz öğrenme) bu tip sinir ağı kullanılır.

Tedarikçilerin kaç grupta sınıflandırılabileceği eğer önceden bilinmiyorsa ve eldeki verilerden hareketle gruplandırma bu mimariye sahip sinir ağı kullanılarak gerçekleştirilebilir (Şekil 7.3).



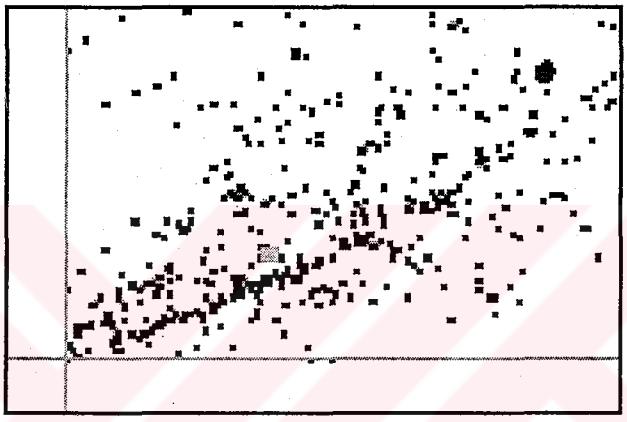
Şekil 7.3 : Rekabetçi öğrenme mimarisi

Rekabetçi yapıdaki bu sinir ağında Rx1 boyutunda girdi veri vektörü kullanılır. ÇKİB YSA'da olduğu gibi net girdi değerini hesaplamak için ağırlıklarla girdi değerlerini çarpıp önyargı değeri ile toplamak yerine, rekabetçi sinir ağlarında ağırlıklar (**W**) ile girdi vektörü (**x**) değerleri arasındaki mesafeye bakılır (||dist||) ve n_1 eleman içeren bir vektör elde edilir. Bu vektördeki elemanların değeri, ağırlıklar ile girdi değerleri arasındaki mesafenin negatifine eşittir (7.2).

$$u_i^1 = -\|W_i - x\| \tag{7.2}$$

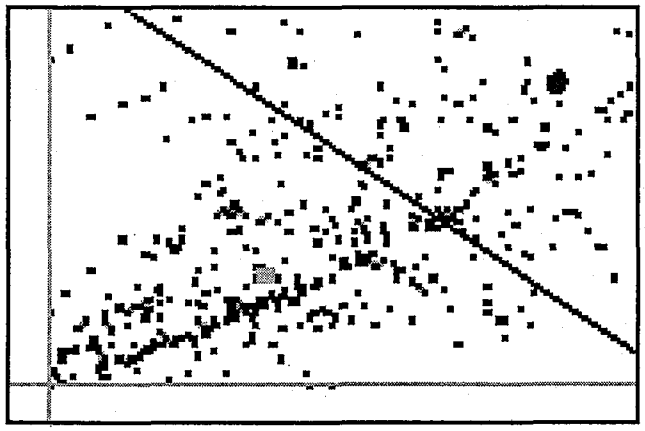
Rekabetçi transfer fonksiyonu (R) net girdi vektörünü alarak, kazanan (net girdi vektöründe en büyük pozitif değere sahip olan sinir hücresi) sinir hücresi dışında kalan İE'lerin çıktıları 0 olur, kazananın ise 1'dir. Modelde girdi vektörü olarak Tablo 7.1'de yer alan ölçütlerden sadece sayısal olan ilk ikisi kullanılan birinci sinir ağı test modelindeki başlangıç parametreleri:

- Adım büyüklüğü: 0.01
- Minimum adım büyüklüğü: 0.002
- Döngü sayısı: 100
- Örnek sayısı: 100
- Ortalama test süresi: 23 sn.



Şekil 7.4 : Girdi verileri ile grup merkezlerinin konumu

Gerçekleştirilen test denemeleri sonucunda, girdi verileri ile iki çıktı sinir hücresinin (iki grup) konumu Şekil 7.4'de görülmektedir. İki sinir hücresinin merkezinden eşit mesafedeki bisektörün konumu (Voronoi çokgenleri) Şekil 7.5'de verilmiştir.



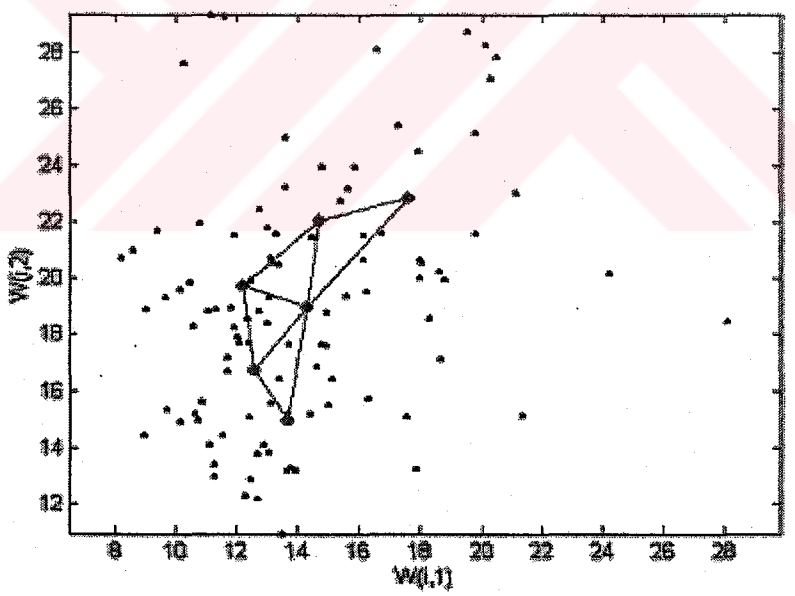
Şekil 7.5 : İE'lerin konumları ve Voronoi diyagramı

Bu iki kümenin test sonucunda elde edilen merkez konumları Tablo 7.3'de yer almaktadır.

Tablo 7.3 : Küme merkezlerinin koordinatı

	Küme 1	**Küme 2**
Küme 1	0.524335	0.493120
Küme 2	0.222556	0.181197

Eğer grup sayısı artırılırsa İE merkezlerinin konumu ve Delaunay üçgenleri değişecektir (Şekil 7.6). En uygun küme sayısının ne olacağı SOM'da deneme-yanılma yöntemi ile test replikasyonları yapılarak belirlenebilir. ARTMap mimarisinde en uygun küme sayısı belirlenebilmektedir. Ancak bu sinir ağında da "ihtiyat" parametresinin ne olacağına karar verebilmek için de deneme-yanılma yöntemi kullanılır. Ayrıca ARTMap başlangıçta sürekli adaptif sistemler için önerilmiştir ve arka planında ağır matematik hesaplamaları vardır. Ancak daha sonra yapılan çalışmalarda ayrık adaptif sistemler için de yeni versiyonları geliştirilmiştir.

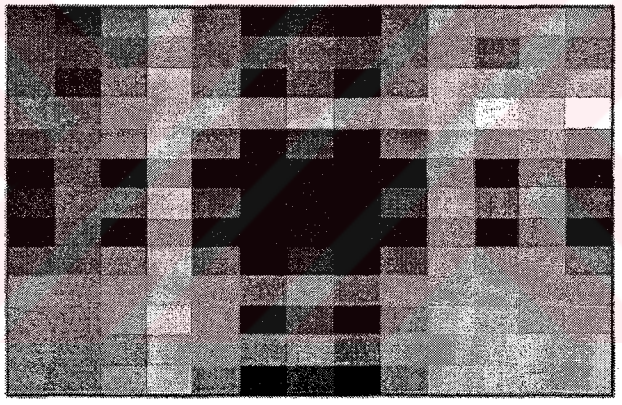


Şekil 7.6 : Altı adet küme merkezinin konumları ve Delaunay üçgenleri

Tablo 7.1'de yer alan bütün ölçütlerin yer aldığı test verileri kullanılarak yeni bir SOM ağı oluşturulmuş ve bu ağın testi yapılarak tedarikçiler gruplandırılmaya çalışılmıştır. Geliştirilen bu SOM ağındaki başlangıç parametreleri:

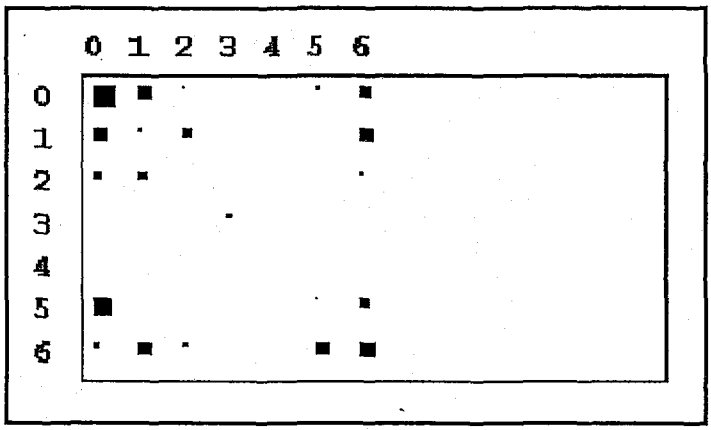
- Girdi vektörünün büyüklüğü: 10x1
- Çıktı katmanında (rekabetçi katman) bulunan işlem elemanı sayısı: 10x10
- Komşuluk derecesi: 4
- Adım büyüklüğü: 0.01
- Minimum adım büyüklüğü: 0.002
- Döngü sayısı: 1000
- Örnek sayısı: 100
- Ortalama test süresi: 11 sn.

Kohonen ağında, kümelemeyi değerlendirirken yapılan dönüşümü (eşleştirmeyi) anlamak önemlidir. İşlem elemanı sayısı genelde beklenen grup sayısından çok daha fazladır. Böylelikle mantıksal olarak bir grup birden fazla işlem elemanı tarafından zapt edilir. SOM dönüşümünden beklenen, verilerin tek bir kümesini gösteren işlem elemanları grubunun oluşmasıdır. SOM'da kümelerin nerede olduğunun nasıl belirlenebileceği önemlidir.



Şekil 7.7 : Gözetimsiz ketleyicilerdeki birleştirilmiş uzaklıklar

Kohonen ağında kümelenmeyi belirlemenin bir yolu, küme merkezleri arasındaki uzaklığa bakmaktır. Girdilerden her bir işlem elemanına olan ağırlık SOM'un sinir hücresi küme merkezini verir. Girdilerden oluşan bir kümede, SOM sinir hücreleri birbirine yakındır. Komşu sinir hücreleri arasında büyük olan mesafeleri belirleyerek girdilerin SOM ağında nerede kümelendiğini bulabiliriz. U-matrisi her sinir hücresinin komşularına olan uzaklığını gösterir. Fazla olan uzaklıklar (Şekil 7.7'deki koyu renkli alanlar, Şekil 7.8'deki büyük siyah kareler) girdi kümelerinin sınırlarını gösterir.



Şekil 7.8 : Gözetimsiz ketleyicilerdeki frekanslar

SOM'u, İE'lerin 2-boyutlu ızgarasının girdi uzayının menzili içerisinde esnemesi şeklinde düşünebiliriz (elastik bant kuramı). Bir sinir hücresi ile komşuları arasında bir lastik bant düşünelim. Bu sinir hücreleri girdi uzayını dönüştürmek için esneyecektir, ancak bu esnemenin de bir sınırı olacaktır. Bir küme içinde sinir hücreleri, bu bölgedeki bütün girdiler benzer olduğundan birbirlerine iyice yaklaşacaktır. Ancak, sinir hücreleri arasında, girdi uzayının boş bir alanında olanlar bulunabileceğinden (bu sinir hücreleri herhangi bir rekabette başarısız olmuştur), bunlar "ölü sinir hücreleri" olarak isimlendirilirler. Bu ölü hücreleri bularak, SOM içerisinde yer alan kümelerin sınırlarını konumlandırabiliriz.

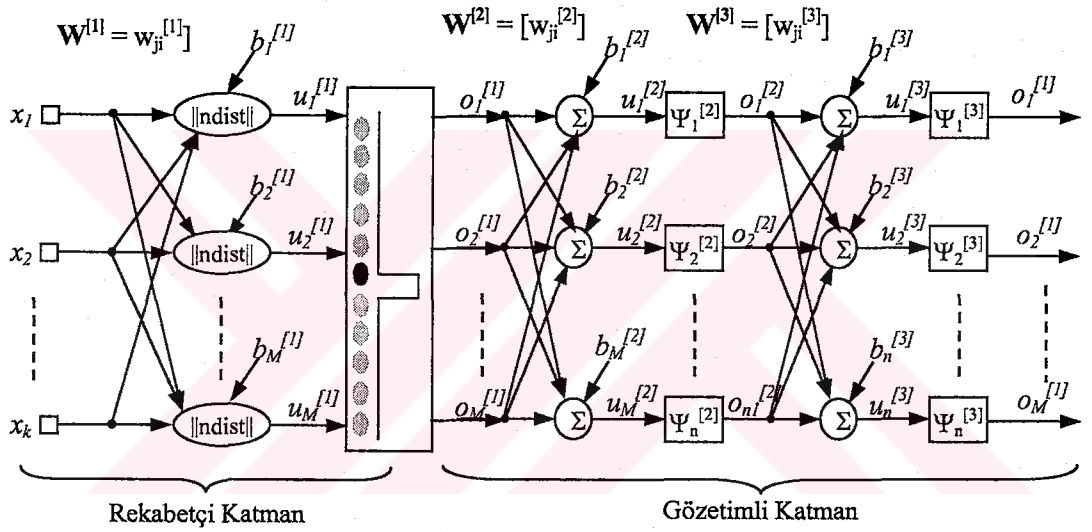
Ortalama nicelendirme hatası kümeleme algoritmasının ne kadar iyi uygunlaştırıldığını ölçer. Bu değer her bir girdi ile kazanan sinir hücresi arasındaki ortalama mesafedir. Eğer bu değer büyükse, kazanan sinir hücresi girdi değerlerinin iyi bir gösterimi değildir. Tedarikçi seçimi için elde edilen ortalama nicelendirme hatası 0.474652 civarındadır. Sinir hücresi sayısı artırıldığında bu değerin azaldığı görülmüştür.

7.5 Karşıtyayılım Ağları ile Tedarikçilerin Sınıflandırılması

SOM ağlarının temelde gözetimsiz öğrenmenin söz konusu olduğu kümeleme, verilerin görselleştirilmesi veya öznitelik çıkarımı gibi alanlarda iyi sonuçlar verdiği ve iyi bir sınıflandırıcı olmadığı daha önce belirtilmişti. Sınıflandırma problemleri için Bölüm 3.1'de de anlatıldığı gibi ÇKİB YSA kullanılabileceği gibi, Kohonen tarafından geliştirilen öğrenen vektör nicelendirme sinir ağı modeli de kullanılabilir. Ancak bu sinir ağını kullanabilmek için *sınıf etiketleri* olarak adlandırılan sınıf merkezlerinin başlangıç konumlarının bilinmesi gerekir (gözetimli öğrenme yöntemi). Eğer bu

değerler bilinmiyorsa o zaman gözetimsiz kümeleme yöntemlerine (örneğin, SOM) başvurularak "k-sınıf etiketi" vektörleri elde edilir. Bu vektörler ÖVN algoritması için başlangıç değerleri olarak kullanılabilir.

Karşıtyayılım ağlarının temel avantajı, kendini örgütleme süreci ile girdi uzayını azaltan SOM ağı kullanılarak kümeleme elde edilmesidir. Böylelikle girdi uzayının yapısı korunurken, boyutu küçültülmüş olur. Karşıtyayılım ağları bütünüyle bir vektör nicelendiriciyi gerçekleyebilir. Bu ağ mimarisinde gözetimsiz kısım kümelemeyi gerçekleştirir, gözetimli kısım ise alıcıya gönderilecek kodu uygulayabilir (Şekil 7.9). Bu ağlar ÇKİB YSA'ya kıyasla çok daha hızlı eğitilir.



Şekil 7.9 : Modellenen karşıtyayılım ağının yapısı

Karşıtyayılım sinir ağı test modelinde, girdi vektörü olarak Tablo 7.1'de yer alan ölçütlerden sadece sayısal olan ilk ikisi kullanılmıştır. Ağın başlangıç parametreleri:

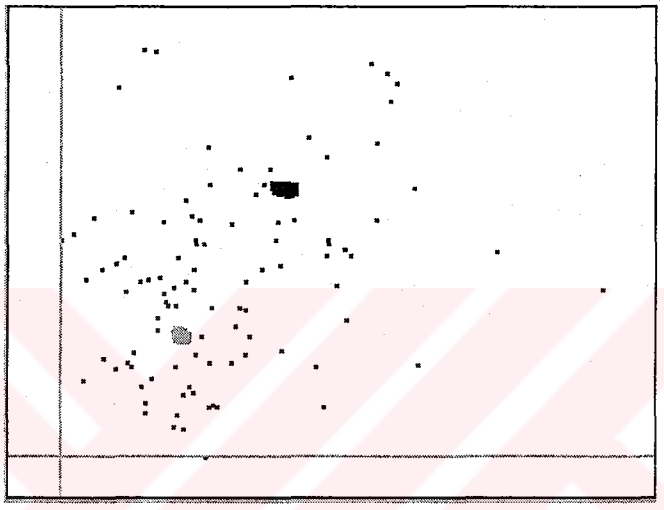
Gözetimsiz Katman:

- *Adım büyüklüğü* : 0.01
- *Başlangıç komşuluk yarıçapı* : 7
- *Soğutma fonksiyonu* : Doğrusal
- *Soğutma çizelgesi parametresi (β)* : 0.02-0.002
- *Döngü sayısı* : 200

- *Örnek sayısı* : 100

Gözetimli Katman:

- *Yapay sinir ağının yapısı* : 3 işlem elemanı
- *Öğrenme tekniği* : Geriye Yayılım + ADIM (oran = 0.7)
- *Performans Ölçütü* : HKO
- *Eğitme oranı* : 0.1
- *Döngü sayısı* : 200
- *Örnek sayısı* : 100



Şekil 7.10 : Girdi verileri ile küme merkezlerinin konumu

Gerçekleştirilen test denemeleri sonucunda, girdi verileri ile iki çıktı sinir hücresinin (iki grup) konumu Şekil 7.10'da görülmektedir. Bu iki kümenin test sonucunda elde edilen merkez konumları Tablo 7.4'de yer almaktadır.

Tablo 7.4 : Karşıtyayılım ile bulunan küme merkezlerinin koordinatı

	Küme 1	**Küme 2**
Küme 1	0.415193	0.650407
Küme 2	0.217799	0.298863

Karşıtyayılım ağı testinde tedarikçi seçimi için elde edilen ortalama nicelendirme hatası 0.062313 civarındadır. Sinir hücresi sayısı artırıldığında bu değerin azaldığı görülmüştür.

8. YAPAY SİNİR AĞLARI İLE KONUM-TAHSİS PROBLEMLERİNİN ÇÖZÜLMESİ

Bu bölümde, Konum – Tahsis problemlerini, özel olarak sürekli örten, p-medyan ve merkez üs modellerini çözmede yapay sinir ağlarının nasıl kullanılabileceği irdelenmiş, kendi kendini düzenleyen ağlar (Kohonen) ile yinelemeli mimariye sahip bir yapay sinir ağı olan Hopfield -Tank yapay sinir ağı kullanılarak çözülmüştür. Ayrıca, p-medyan ve Sürekli Örten Konum-Tahsis problemlerini (SÖKT - Continuous Covering Location Problem) çözmek için yeni bir yaklaşım olarak *Titreşimli Genetik Algoritma* yaklaşımı önerilmiştir (Ermiş ve diğ., 2002a,b).

Konum - tahsis problemleri, talep merkezlerine hizmet verecek birden fazla tesise gereksinim duyulduğunda söz konusu olmaktadır. Birden fazla tesisin konumunun ne olacağı ile ilgilenildiğinde, sıklıkla bununla ilişkili olarak bir tahsis problemi de söz konusu olur. Konum - tahsis konusunun eşzamanlı olarak ele alınması gerekse de, tesislerin konumları belirlendikten sonra talep merkezlerinin paylaşımı göreceli olarak basittir. Eğer tersi söz konusu olursa, yani talep merkezlerinin paylaşımı biliniyor tesislerin konumu bilinmiyorsa, problem birbirinden bağımsız tek tesis yeri seçimi problemine dönüşür.

KTP ile gündelik hayatın pek çok alanında karşılaşmak mümkündür : dağıtım, resmi hizmetler, iletişim, aktarma merkezleri, depolama, yangın söndürme istasyonları, vs. KTP'nin genel bir tanımı; "Verilen talep merkezleri konumları ile talepler için, tedarik merkezlerinin sayısını, konumunu ve karşılayacağı talepleri bazı eniyileme ölçütlerini sağlayacak şekilde bulmak" olarak yapılabilir (Drezner ve Hamacher, 2002).

Talep merkezleri noktasal olabilir veya belirli bir bölgede bir olasılık dağılım fonksiyonuna göre dağılmış olabilir. Arz merkezleri birbirlerinden bağımsız olarak işlevlerini yerine getirirler. Bilinen sabit bir kapasiteye sahip olabilirler veya kendilerine tahsis edilen talep merkezlerinin istem miktarı toplamına göre belirlenebilir. Ağ, tek boyutlu, küresel ve genel $\Re^K$ durumları söz konusu olsa da KTP, yaygın olarak bir düzlemde gerçekleşir. Düzlemsel problemlerde, Minkowsky'nin ℓ_p normu genellikle zigzaglı veya Öklit olarak alınır. Potansiyel

yerleşim noktaları p-medyan problemlerinde olduğu gibi ayrık olabilir veya bazı durumlarda potansiyel yerleşim noktalarının yakınındaki alanların da değerlendirmesi gerektiğinden sürekli örten yaklaşımı kullanılabilir. Eniyileme kriteri problemin ürün ve hizmetin sıradan/sıra dışı olmasına bağlı olarak değişir. Ancak, her iki durumda da amaç fonksiyonunun her bir arz merkezi ile müşterileri arasındaki mesafelere bağlı olduğu söylenebilir. Ulaştırma maliyetleri, arz merkezlerinin müşterilere bağımsız veya koordineli olarak hizmet vermesine bağlı olarak değişir. İkinci durumda konum - rotalama problemi söz konusu olabilir (Current, 2001).

Son yıllarda konum-tahsis ağlarını daha gerçekçi yapabilmek için akışa bağlı ölçek ekonomisi (O'Kelly ve diğ., 1998), sadece yeterli büyüklükteki akışa izin vermek amacıyla iskontonun yer aldığı akış eşikleri modeli (Podnar ve diğ., 1999) ve merkez üs arkları (Campbell ve diğ., 2003) gibi birçok yeni yaklaşım önerilmiştir. Ark - konum problemi, düğüm noktasına odaklı değil de ark odaklı bir yaklaşımdır ve iskontolanmış arklarla birbirine bağlı olan belirli sayıdaki merkez üssü konumlandırmak yerine, belirli sayıdaki iskontolanmış arkı araştırarak konumlandırmaya çalışır. Bu nedenle, merkez üs ark konumu modeli yoğun taşımalar için daha büyük araçları yansıtan iskontolanmış arkları kullanmada büyük esnekliğe sahiptir.

8.1 Sürekli Örten Konum-Tahsis Problemleri

SÖKT problemleri, herhangi bir yerin serbestçe seçilebildiği yeni bir tesisin konumlandırılması gerektiğinde söz konusudur. Konum uzayı, genellikle koordinatları gösteren sürekli değişkenlerle ifade edilir. Bu yöntem, sadece ulaştırma maliyetinin göz önünde bulundurulduğu problemlerde ideal konumu belirlemede kullanılabilir. İdealden kasıt, böyle bir yerde bir tesisin kurulmasının fiziksel olarak mümkün olup olmamasını vurgulamaktır. Ayrıca SÖKT problemlerinde, ayrık konum-tahsis problemlerinde olduğu gibi, mevcut uygun olabilecek yerlerin geniş bir listesi de verilmez. Bununla beraber aday olabilmek için yerlerin uygulanabilir bölge veya bölgelerde yer alması gerekir. Genellikle bu bölgeler çok karmaşık olmayan yapıdaki kısıtlarla tanımlanır. Konum uzayı coğrafi bir yapıda ise uygulanabilir bölge genellikle bir poligonal bölge dizgesinin iç kısmı olarak tanımlanır. Teknik, ekonomik veya politik düzenlemeler, seçilen konumun bazı hassas yerlere çok yakın olmamasını gerektirdiği durumlarda daha karmaşık bölgeler ortaya çıkacaktır. Bazen de modelde uygulanabilir bölgeyle ilgili herhangi bir kısıtlama olmaz. Bu tip modeller daha çok teorik olarak ilgi çekmekle birlikte çözüm sürecinin ilk basamağı olarak

kullanılabilir ve göz ardı edilmemesi gerekir. SÖKT problemlerinde, talep noktaları ile x-y koordinatları ve her bir talep noktasından bir tesise olan akışlar bilinir. Maliyet, talep ile her bir talep noktası ve tesis arasındaki mesafe ile doğru orantılı olarak hesaplanır.

8.1.1 Problemin Formülasyonu

SÖKT problemini modellemek için aşağıdaki varsayımlar yapılmıştır:

- Talep noktalarına bağımsız olarak hizmet verilir.
- Tedarik merkezleri $\Re^K$ uzayında herhangi bir yerde konumlandırılabilir.
- Her talep kendisine en yakın tedarik merkezine atanır.
- Kuruluş maliyetleri göz ardı edilmiştir ve tedarik merkezi sayısı belirlidir.
- Ulaştırma maliyetlerinin Öklit / düz uzaklığın karesi mesafesi ile orantılı olduğu varsayılmıştır.

Bu problemi formüle etmek için, aşağıdaki girdi ve parametreler tanımlanmıştır:

I : i (N) olarak endekslenmiş talep noktalarının oluşturduğu küme

J : j (M) olarak endekslenmiş aday tesislerin oluşturduğu küme

x_i : i'nci talep noktasının konumları

w_j : aday tesislerin konumları

h_i : i'nci talep noktasının göreceli talepleri

ve karar değişkeni;

$$y_{ij} = \begin{cases} 1 & \text{Eğer i'nci talep noktası j'nci konuma yerleştirilmişse} \\ 0 & \text{Aksi takdirde} \end{cases}$$

Bu notasyon kullanılarak, SÖKT problemi aşağıdaki gibi yazılabilir:

$$\min \quad \sum_{i \in I} \sum_{j \in J} h_i d_{ij} y_{ij} \tag{8.1}$$

$$\text{s.t.} \quad \sum_{j \in J} y_{ij} = 1, \ \forall i \in I \tag{8.2}$$

$$y_{ij} \in \{0,1\} \tag{8.3}$$

Öyle ki:

$$d_{ij} = \left\| w_j - x_i \right\|^2 \tag{8.4}$$

Veya mesafe fonksiyonu olarak Öklit kullanıldığında;

$$d_{ij} = \sqrt{\left\| w_j - x_i \right\|^2} \tag{8.5}$$

Talebin bir olasılık dağılımına göre değişmesi durumunda:

$$\min \quad \int_{x \in X} D(x) \left\| w_{Y(x)} - x \right\|^2 \tag{8.6}$$

Öyle ki:

X : talep bölgesi

D : talebin çok değişkenli dağılım fonksiyonu ($x \in X \rightarrow [0,1]$)

Y : paylaşım fonksiyonu ($x \in X \rightarrow \{1,2,..., J\}$)

w_j : aday tesis yerlerinin konumları

Arz merkezlerinin konumlarının verilmesi durumunda, talebin hangi tesisten karşılanacağına ilişkin soruya cevap en yakın tesistir. "En yakın tesis hangisidir?" sorusunun cevabı ise Voronoi çokgenidir (Okabe ve diğ., 1992)

Tartılandırılmış mesafeler kullanıldığında, hangi ağırlıkların kullanılması gerektiği sorusu hemen akla gelir. Bu nedenle, ağırlıklar değiştirildiğinde en iyi yerleşim merkezlerinin kararlı olmayan davranışı gözlemlenir (Erkut ve Öncü, 1991). Buradan hareketle diğer mesafe fonksiyonları da ele alınmıştır (özellikle, dikdörtgensel mesafe (Carrizosa ve diğ., 1995; Drezner ve diğ., 1985; Appa ve diğ., 1994). Fakat bu modellerin uygulanabilme potansiyelleri, dikdörtgensel mesafe ile bir itme etkisinin nasıl yayılacağı açık olmadığından, oldukça az gibi gözükmektedir.

Ayrık talep durumu ve öklit mesafe fonksiyonu için önceki çalışmalarda yerel aramaya dayalı sezgisel yaklaşım önerilmiştir (Drezner ve Hamacher, 2002). Hesaplama süresi açısından daha ümit verici bir yaklaşım, tavlama benzetimi gibi kombinatorik eniyileme (Liu ve diğ., 1994), yasaklı arama veya genetik algoritma yöntemlerini kullanmaktır (Abdinnour-Helm, 1998). GA farklı eniyileme problemleri için test edilmiş ve genellikle iyi sonuç vermiştir (Drezner ve Hamacher, 2002). Ancak, KTP'yi çözmede GA kullanmak için çok az çalışma yapılmıştır.

8.1.2 Sürekli Örten Konum-Tahsis Probleminin Kohonen Ağları ile Çözülmesi

SÖKT problemleri bazı kümeleme analizlerine ve vektör nicelendirme problemlerine benzerlik gösterir. Moreno ve diğ. (1991), ayrık veya sürekli uzay, zigzaglı veya öklit mesafe fonksiyonlu KT problemlerini çözmek için Hiyerarşik Artan Kümeleme Algoritmasını (HAKA) temel alan sezgisel bir yöntem önermiştir.

Diğer yandan, SOM'un kümeleme için uygulanabilirliliği açıktır. Mangiameli SOM'un hiyerarşik kümeleme yöntemleri için çok uygun olduğunu bulmuştur (Mangiameli ve diğ., 1986). Wang, bir Doğrusal Diskriminant Analizi yönteminin yanlış sınıflandırmalarının kümeleme analizinde kullanmıştır (Wang, 1996).

Bu bölümde, kümeleme analizlerinde oldukça başarılı sonuçlar veren ve Bölüm 4'te anlatılan SOM yapay sinir ağı modeli SÖKT problemlerini çözmek için kullanılmıştır. Girdi örüntüsü, talep merkezlerinin k-boyutlu konum vektörleridir ve $D(x)$ olarak verilen olasılık dağılımına göre sinir ağına rastsal olarak verilirler. Sinir ağı, m-tesis (arz merkezi) ile ilgili olarak çıktı katmanına sahiptir. Her birisi için ağırlık vektörü tedarik merkezine atanacak konuma karşılık gelir. Öğrenme kuralının etkisi tedarik merkezlerini girdi olan talep merkezlerine doğru çekmektir. Ancak, sadece en yakındaki tedarik merkezi ve σ genişliğindeki komşuları önemli ölçüde etkilenir. Başlangıçta, düzenleme evresini hızlandırmak için, σ komşuluğu çoğu çıktı biriminin her yinelemede ağırlıklarını adapte etmesine izin verecek yeterlilikte büyük olmalıdır. Daha sonra, hem komşuluk hem de öğrenme oranı harita sabitleninceye kadar yavaşça azaltılır.

Çıktı katmanındaki birimlerin düzenleniş biçimi ve kullanılan metrik, girdi verisinin topolojisine bağlı olarak belirlenebilir. Eğer mümkünse, farklı düzenlemeler denenerek elde edilen farklı haritalarla ilgili çözümler arasından en iyisi seçilir. Farklı düzenlemelerden elde edilen çözümler birbirinden çok az farklı olsa da en iyi düzenleme problemden probleme değişebilir. Çünkü en iyi çözüm sadece talebin dağılımına değil, aynı zamanda konumlandırılacak tedarik merkezi sayısına da bağlıdır.

Orijinal amaç fonksiyonu, düz uzaklığın karesi toplamını enküçükleme olmasına rağmen, önerilen yaklaşım öklit mesafe fonksiyonunun söz konusu olduğu durumlar için de yaklaşık bir çözüm bulmada kullanılabilir. Bir diğer olasılık da SOM ile elde edilen tahsis matrisini başlangıç konumları gibi kullanarak m-bağımsız tek tesis konumlandırma problemi olarak çözmektir. Düz uzaklığın karesi durumunda, bu

problemler her bir tedarik merkezine ait ağırlık merkezi hesaplanarak çözülür. Öklit mesafe fonksiyonu söz konusu olduğunda ise Weiszfeld algoritması kullanılabilir (Drezner ve Hamacher, 2002).

Ağırlıkların başlangıç durumuna getirilmesi, öğrenme oranı ve komşuluk genişliği hakkında bazı noktaları aydınlatmak gerekir. Girdi örüntülerinin sunumunun stokastik bir yapıda olması, bir bükülme etkisi oluşturarak, daha iyi çözümlerin araştırılmasına izin vermek yoluyla başlangıçta bir yerel minimumda takılıp kalmasını önler. Yeteri kadar yavaş bir "soğutma" çizelgesi kullanılarak, başlangıç ağırlıklarından bağımsız iyi bir çözüm elde edilebilir. Böyle bir çizelge, eğer çıktı katmanı çok fazla sayıda İE içeriyorsa hesaplama bakış açısıyla pratik olmayabilir. Ancak, KTP'de tedarik merkezlerinin sayısı genellikle kontrol edilebilir düzeyde olduğundan, her bir öğrenme çevriminde gereksinim duyulan hesaplama çok fazla değildir ve anlamlı düzeyde yineleme gerçekleştirilebilir.

8.1.3 Sürekli Örten Konum-Tahsis Probleminin Genetik Algoritma ile Çözülmesi

GA genel bir çözüm tekniğidir ve çizelgeleme, bilgisayar bilimleri, yönetim, vs. gibi alanlarda herhangi bir probleme, problemin çözümleri için kodlama şeması geliştirdikten ve parametrelerin ne olacağına karar verdikten sonra uygulanabilir. Ancak, herhangi bir eniyileme problemi için GA tasarlamak mümkün olsa da, GA bütün problemlerde eşit ölçüde etkin değildir. Gerçekte, GA kısıt sayılarının az ve amaç fonksiyonunun karmaşık olduğu problemlerde çok iyi performans gösterir (Reeves, 1993). Bu özelliğinden dolayı GA, sürekli KT problemini çözmek için uygun bir yöntemdir.

GA'da en yaygın gösterim şekli ikilik (binary) kodlama sistemidir (Goldberg 1989). Bu kodlamada kromozomlar genlerin oluşturduğu bir kümeyi içerir ve bu nedenle bir kromozom l genden (g_l) oluşan bir x vektörüdür:

$$x = (g_1, g_2, \ldots, g_l),\ g_l \in \{0, 1\} \tag{8.7}$$

Ancak, parametreleri sürekli uzayda değer alan eniyileme problemlerinde, genleri doğrudan reel sayılar olarak göstermek genotip (*kodlama*) ile fenotip (*arama uzayı*) arasında herhangi bir farklılığın olmamasını sağlayacağından daha doğal olacaktır. Reel genetik kodlamanın sürekli uzayda sayısal eniyilemede kullanılmasına ilişkin pek çok uygulama vardır (Michalewicz,1992). Kromozomun uzunluğu problemin

çözümünün oluşturduğu vektörün uzunluğudur (l adet tedarik merkezi); bu yüzden, her bir gen problemin bir değişkenini gösterir (bir tedarik merkezinin koordinatları). Gen değerleri, gösterdikleri değişkenlerin oluşturduğu aralıkta kalmaya zorlandığından, genetik operatörler bu ihtiyacı karşılamalıdır.

Obayashi'nin belirttiği gibi, reel kodlu GA'lar için kullanılan mutasyon oranının, ikilik sistemde kodlanmış GA'larda kullanılana göre daha büyük değerler alması gerekir (Obayashi ve diğ., 1992). Bunun nedeni, ikilik sistemde kodlanmış bir sayının bir hanesinde yapılacak bir değişikliğin, sayı değerini büyük oranda değiştirebilecek olmasıdır. Oysa reel kodlu bir sayı için benzeri bir işlemin önemli bir değişikliğe yol açma şansı daha azdır. Dolayısıyla, ikilik sistemdeki bir GA ile aynı mutasyon oranını kullanan reel kodlu bir GA dizayn uzayını arama açısından daha zayıf kalacaktır. Reel kodlu GA kullanılırken mutasyon oranının yüksek tutulması, tasarım uzayının çok farklı noktalarına algoritma tarafından erişilerek araştırılabilmesini sağlayacaktır. Bu düşünce, önerilen ve ilerleyen bölümlerde açıklanacak olan Titreşimli Mutasyon (Vibrational Mutation) tekniği ile gerçekleştirilebilir. Önerilen Titreşimli Genetik Algoritma yaklaşımı, arama/kıymetlendirme arasındaki etkin denge ve çeşitliliği temin eder. Bu yüzden, TGA yakınsama oranı açısından çok iyi performans gösterir. TGA'daki titreşim, genetik süreçte kullanılan bazı parametrelerde bir çeşit dalga formunda üretilen salınım olarak tanımlanabilir.

8.1.3.1 Titreşimli Genetik Algoritmanın Genel Hatları

Önerilen genetik algoritmanın işleyişi aşağıdaki işlem adımlarıyla açıklanabilir:

1. *Başlangıç Popülasyonu:* başlangıç popülasyonunda yer alan her bir kromozom rastsal olarak oluşturulur ve deney sonuçlarından da görülebileceği gibi popülasyonun büyük olması gerekmez.
2. Mevcut popülasyonda yer alan her birey için uygunluk değerini hesapla.
   a. Eğer mevcut popülasyondaki bireylerden birisi problemin durdurma kriterini (hata oranı veya maksimum yineleme sayısı gibi) karşılıyorsa, DUR.
   b. Diğer durumlarda, bir sonraki adıma geç.
3. *Seçim:* bir seçim operatörü kullanarak mevcut popülasyondan bireyleri çıkararak bir ara popülasyon oluşturulur.
4. *Yeniden Düzenleme:* bu ara popülasyona çaprazlama ve mutasyon operatörlerini uygulayarak yeni bir popülasyon oluştur.
   a. Eğer nesil sayısı IP'ye eşitse titreşimli mutasyonu uygula.
   b. Diğer durumlarda, Adım 2'ye git.

```
main()
{
  read the data /*demand and demand center locations
    matrices of the n interacting nodes*/
  initialize min_cost /*the best cost solution*/
  initialize the random number generator
  genetic()
  report min_cost
}

genetic()
{
  gc=0 /*initialize the generation counter
    genetic search*/
  seeds = 0
  initialize min_cost /*the minimum cost
    solution for genetic
  generate P(gc) /*the population of solutions for
    generation gc*/
  evaluate P(gc) /*calculate the cost of a chromosome
    using the distance based rule*/
  while (gc ≤ max_gc)
    gc = gc+1
    seeds = seeds+1
    select P(gc) from P(gc-1)
    recombinate P(gc)
    if (seeds = IP)
      vibrational_mutation()
      seeds = 0
    evaluate P(gc)
    update min_cost
}
```

Şekil 8.1 : TGA'nın formülasyonunun C pseudo-kodu ile gösterimi

8.1.3.2 Titreşimli Çaprazlama

BLX-α çaprazlama tekniği, reel kodlu genetik algoritmalarda, eski iki bireyden yeni iki birey elde etmek için sıkça kullanılan bir tekniktir (Eshelman, 1993). Bu teknikte yeni bireyler eski bireylerin parçalarından oluşturulur. Eski bireylerin parçalarının nasıl dağılacağı, kullanıcı tarafından tayin edilen α parametresi ile belirlenir. $x_i^{(1,t)} < x_i^{(2,t)}$ olmak üzere iki eski birey $x_i^{(1,t)}$ ve $x_i^{(2,t)}$ için BLX-α $[x_i^{(1,t)} - \alpha(x_i^{(2,t)} - x_i^{(1,t)}), x_i^{(2,t)} + \alpha(x_i^{(2,t)} - x_i^{(1,t)})]$ aralığında bir nokta seçecektir. Burada u, [0,1] aralığında bir rastsal sayıdır. Buna göre yeni bireyler aşağıdaki gibi üretilecektir:

$$x_i^{(1,t+1)} = (1-\gamma)x_i^{(1,t)} + \gamma\, x_i^{(2,t)} \tag{8.8}$$

$$x_i^{(2,t+1)} = (1-\gamma)x_i^{(2,t)} + \gamma\, x_i^{(1,t)} \tag{8.9}$$

$$\gamma = (1+2\alpha)u - \alpha \tag{8.10}$$

Buradaki γ değeri [-α, 1+α] aralığında düzgün bir şekilde dağılacaktır.

Titreşim yaklaşımının çaprazlama işlemlerinde uygulamasını yapmak için, BLX-α yöntemi değiştirilerek geliştirilen Çift Yönlü Alfa (ÇYA) metodu kullanılmıştır (Hacıoğlu ve diğ., 2001). Bu yöntemde α değerine salınım yaptırılmaktadır. Bu salınım, genetik süreç içerisinde periyodik olarak doğrusal, sinüzoidal yada öngörülen başka bir şekilde olabilir. Eğer α bir m ana değeri etrafında, bir G genliğinde sinüzoidal formda salınım yapacaksa, tek yineleme sayıları için [m-G/2, m] aralığında, çift yineleme sayıları için [m+G/2, m] aralığında çift yönlü olarak salınım yapacaktır. Bu salınım genetik süreç boyunca periyodik olarak devam edecektir. Sinüzoidal yaklaşımın kullanıldığı (Sinüzoidal Çift Yönlü Alfa, SÇYA) durum için formüller aşağıdaki gibi yazılabilir:

Tek yineleme sayıları için:

$$\alpha = m + G \cdot Sin\left[\frac{(NS - \bmod P)}{P}\pi\right] \tag{8.11}$$

Çift yineleme sayıları için:

$$\alpha = m - G \cdot Sin\left[\frac{(NS - \bmod P)}{P}\pi\right] \tag{8.12}$$

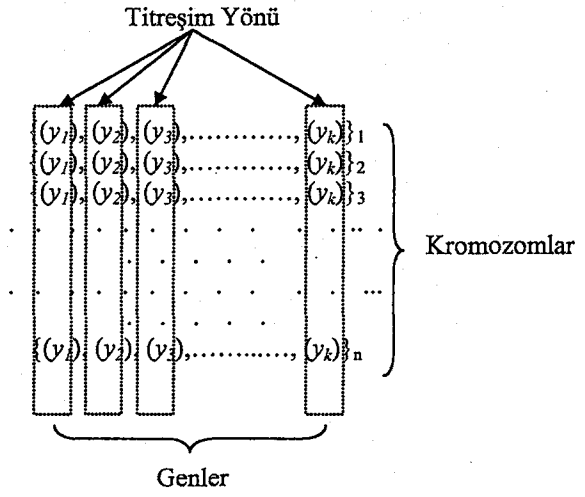
Yukarıda belirtildiği gibi, burada m ana değer, G dalga genliği ve P salınım periyodudur.

8.1.3.3 Titreşimli Mutasyon

Mutasyon esaslı titreşim yaklaşımı yenilemeden hemen sonra uygulanır ve genetik sürecin başlangıcından itibaren periyodik olarak gerçekleştirilir. İlk adımı takiben, rastsal bir dalga popülasyona etki ettirilir. Titreşimli mutasyon sırasında, popülasyondaki bütün kromozomların (bireyler) tüm genleri aşağıdaki rastsal dalgaya bağlı olarak mutasyon geçirirler.

$$\begin{aligned} &y_{new,i}^{m} = y_i^m\left[1 + MA \cdot u_1(0.5 - u_2)\right] \\ &m = 1,\ldots,n;\ i = 1,\ldots,kn;\ u_1, u_2 \in [0,1] \end{aligned} \tag{8.13}$$

Burada y_i^m gen (kontrol noktası), kn kromozomdaki toplam gen sayısı, n popülasyondaki toplam birey (kromozom) sayısı, MA ana genlik ve u_1 ve u_2 [0,1] aralığında rastsal bir reel sayıdır.



Şekil 8.2 : Titreşimli mutasyonda titreşim yönü

Titreşim uygulaması, ilk kromozomun belirli bir sırasındaki genden başlar ve Şekil 8.2'de gösterildiği gibi, diğer kromozomlardaki aynı sıradaki genler boyunca devam eder. Genetik sürecin ilk adımında, uygunluk değerlerinin hesaplanması, uygun bireylerin seçimi ve yenileme işlemlerini takiben ortaya çıkan yeni bireylere titreşim uygulaması yapılır. İlk olarak, bütün kromozomların ilk sıradaki genleri (j=1) baştan sona (i=1'den n'e kadar) titreşime tabi tutulur. Bunu takiben bütün kromozomların ikinci sırasındaki (j=2) genler aynı şekilde (i=1'den n'e kadar) titreşimden geçer. Tüm kromozomların son sırasındaki genler (j=kn) de titreşimden geçtikten sonra başlangıç adımı için titreşim uygulaması sana erer. Genetik süreç, bunu takip eden IP-1 adımında (mutasyon oranı P_m =1/IP), titreşim uygulaması olmaksızın düzenli bir şekilde (uygunluk değerlerinin hesaplanması, seçim, yenileme) devam eder. IP'nci adımda, aynı ilk adımda olduğu gibi titreşim uygulaması yapılır ve bu uygulama her IP adımda bir tekrarlanır.

Titreşim uygulaması, yeni popülasyondaki bireylerin çözüm bölgesinde rastsal bir şekilde yayılmasını sağlar. Bu yeni popülasyondan itibaren genetik süreç belli bir süre titreşim uygulaması uygulanmaksızın devam eder. Çünkü titreşim sonucu ortaya çıkan popülasyondan en uygun bireylerin elde edilebilmesi biraz zaman alacaktır. Sonra tekrar titreşim uygulaması yapılarak en son adımda bulunmuş olan popülasyonun çözüm bölgesine yayılması sağlanır. Titreşimle ortaya çıkan rastsal

bir şekilde çözüm bölgesine yayılmış popülasyon, titreşim öncesi bulunan en iyi çözümün komşularının araştırılmasına olanak sağlayarak daha iyi çözümlerin bulunabilmesini, başka bir deyişle çözüm uzayının daha iyi araştırılabilmesini temin eder. Öte yandan, genetik süreç devam ederken popülasyonun ortalama uygunluk değerine dikkat edilmesi gereklidir. Ortalama uygunluk değeri azalırken, titreşim amacıyla kullanılan dalganın ana genliği azaltılmalıdır. Ortalama uygunluk değerinin azalmasıyla genel optimuma yaklaşılacağı için, titreşim uygulaması sırasında popülasyonun başlangıçtaki gibi çok geniş bir bölgeye yayılması gereksiz olacaktır ve bu aynı zamanda performansı olumsuz etkileyecektir. Bununla beraber, genel optimuma yaklaşılırken popülasyonu dar bir aralıkta titreşime maruz bırakmak genel optimumu yakalamayı hızlandıracaktır. Bu nedenle ana genlik değeri MA genetik süreç boyunca aşağıdaki gibi belirlenir:

$$MA = \left[\frac{\log(1 + AF_0)}{\log(1 + AF_k)} \right]^r \tag{8.14}$$

AF_0 ve AF_k sırasıyla genetik sürecin başlangıç adımındaki ve içinde bulunulan adımındaki ortalama uygunluk değerleri olup r reel bir sayıdır. Genetik sürecin ilk adımında MA 1 olacaktır. Eğer ilk adım için birden farklı bir sayı arzu edilirse, MA bir parametre ile çarpılarak istenen ayarlama yapılabilir. MA, rastsal dalga için olabilecek en büyük dalga genliğini belirler. Denklem 8.14'deki r, MA değerinin azalma hızını belirler. Hızlı bir azalma için r büyük bir değer almalı, yavaş bir azalma için ise r küçültülmelidir.

8.1.4 Deneyler ve Elde Edilen Sonuçlar

İki farklı test veri grubu kullanılarak, öklit ve düz uzaklığın karesi mesafe fonksiyonları için klasik GA, TGA ve SOM yaklaşımları ile ayrı ayrı deneyler yapılmış ve sonuçlar karşılaştırılmıştır.

8.1.4.1 Öklit Mesafe Fonksiyonu için Yapılan Deneyler

Bu bölümde, ABD'nin büyük şehirlerinin yer aldığı 147 talep merkezini içeren SÖKT problemi ele alınmıştır (Daskin, 1995). Deneyler başlangıçta farklı popülasyon büyüklükleri için gerçekleştirilmiştir (sadece genel GA'da). Çaprazlama oranına (P_c) 1, mutasyon oranına (P_m) ise titreşimli mutasyon tekniğinin periyoduna bağlı olarak

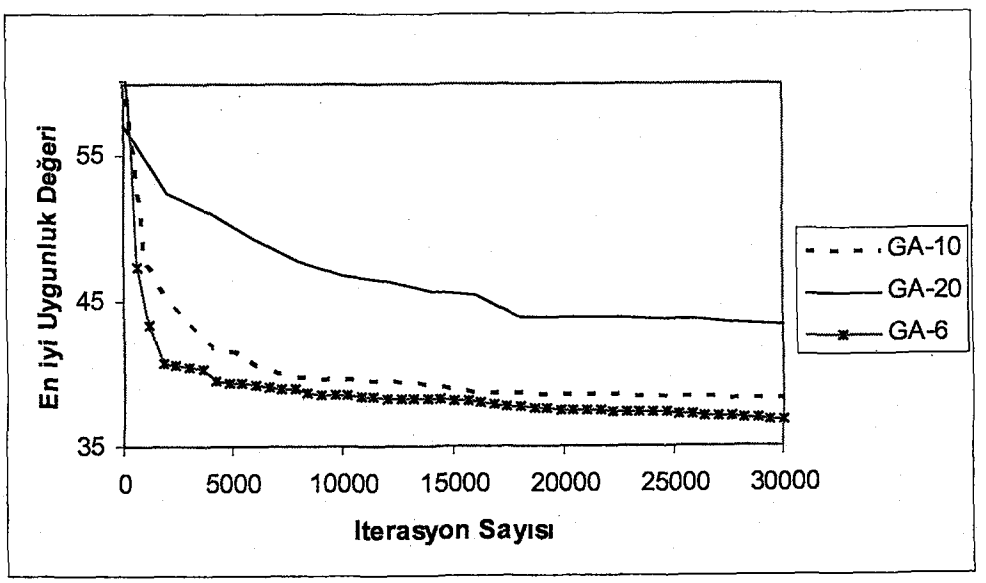
bir değer verilmiştir. Ebeveynler, Evrensel Örnekleme'ye (SUS) göre seçilmiştir (Baker, 1987). Sunulan sonuçlar 10 farklı denemenin ortalamasıdır. Tasarlanan ve gerçekleştirilen deneyler:

1. *Deney 1*: Farklı popülasyon büyüklüğüne sahip (n=6,10 ve 20), geleneksel reel kodlu GA, α = 0.5, mutasyon olasılığı P_m =0.02 ve mutasyon metodu düzgün olmayan mutasyon (Wright 1990).

2. *Deney 2*: Çaprazlama tekniği SÇYA; Sinüzoidal Çift Yönlü Alfa (m=0.7, G=0.2, P=50 ve m ortalama değer, G genlik ve P periyot), mutasyon olasılığı P_m =0.02 ve mutasyon metodu düzgün olmayan mutasyon; popülasyon büyüklüğü 6.

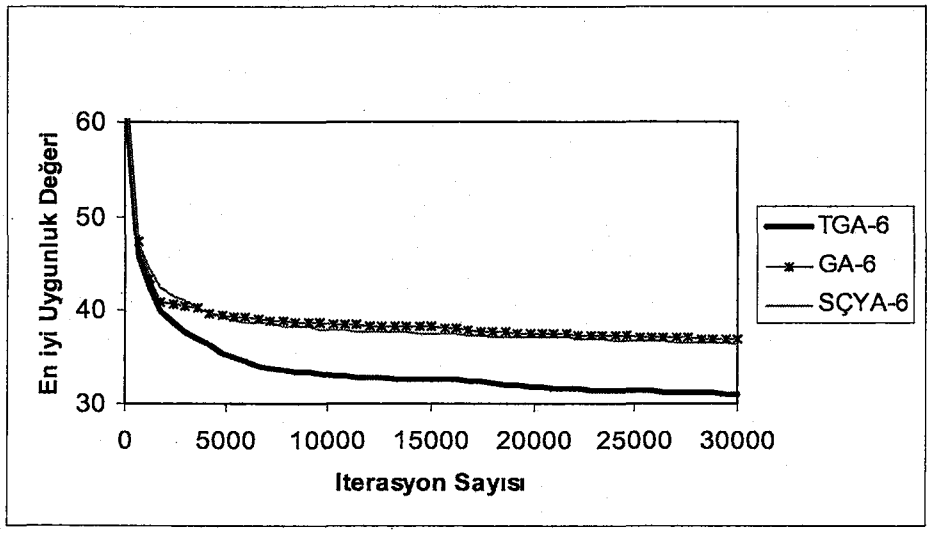
3. *Deney 3*: TGA yöntemi; IP = 25, r = 3, α = 0.5, popülasyon büyüklüğü 6.

α = 0.5 iken, bireyin ebeveynlerinin dışında kalması olasılığı ebeveynlerinin arasında kalma olasılığına eşit olacağından yakınsama (kıymetlendirme) ve ıraksama (araştırma) arasında dengeli bir ilişkiye erişilir. α büyüdükçe, serbest kalan kıymetlendirme bölgeleri araştırma bölgelerine yayılacağından araştırma düzeyi popülasyondaki çeşitliliği artıracak şekilde daha fazlalaşır. Şekil 8.3, farklı popülasyon büyüklükleri için Deney 1'de elde edilen en iyi uygunluk fonksiyonu değerlerini göstermektedir.



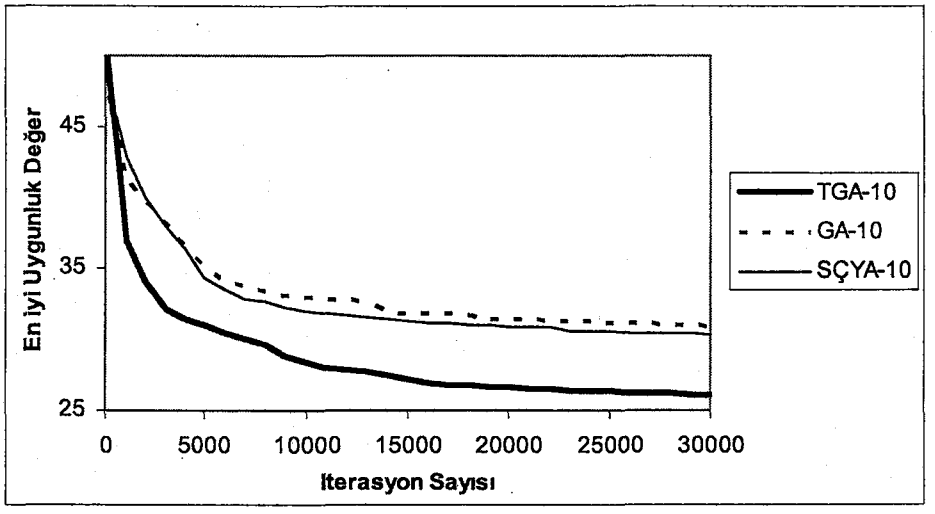
Şekil 8.3 : Farklı popülasyon büyüklükleri (n = 6, 10 ve 20) en iyi uygunluk değerlerinin karşılaştırılması

Sonuçlar, popülasyon sayısı arttıkça iyileşmemektedir. Bu nedenle diğer deneylerde, en iyi uygunluk değerini veren popülasyon büyüklüğü olan n = 6 kullanılmıştır.



Şekil 8.4 : Tedarik merkezi sayısı 10 olduğunda farklı çözüm yöntemleri (TGA, SÇYA ve GA) için elde edilen ortalama uygunluk değerleri

Şekil 8.4'de, 10 tedarik merkezi için klasik GA, SÇYA ve TGA sonuçları görülmektedir. Şekil 8.4 dikkatlice incelendiğinde, SÇYA için elde edilen sonuçların GA'dan elde edilenlerinden biraz daha iyi olduğu fakat TGA'nın çok daha iyi sonuç verdiği görülebilir. Şekil 8.5'de, 15 tedarik merkezi için elde edilen sonuç gösterilmiştir ve bir öncekine benzer sonuçlar elde edilmiştir.



Şekil 8.5 : Tedarik merkezi sayısı 15 olduğunda farklı çözüm yöntemleri (TGA, SÇYA ve GA) için elde edilen ortalama uygunluk değerleri

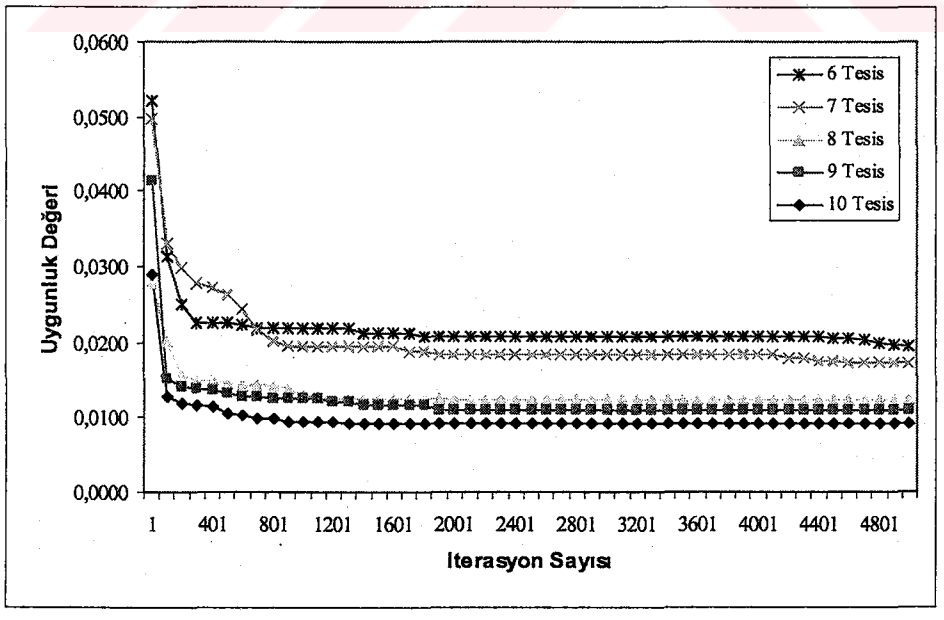
8.1.4.2 Düz Uzaklığın Karesi Mesafe Fonksiyonu için Yapılan Deneyler

Bu bölümde, Lazano tarafından verilen ve 49 talep merkezine ilişkin verilerin yer aldığı veri seti kullanılmış (EK-D) ve önerilen TGA ile SOM'un sonuçları karşılaştırılmıştır (Lazano ve diğ., 1998).

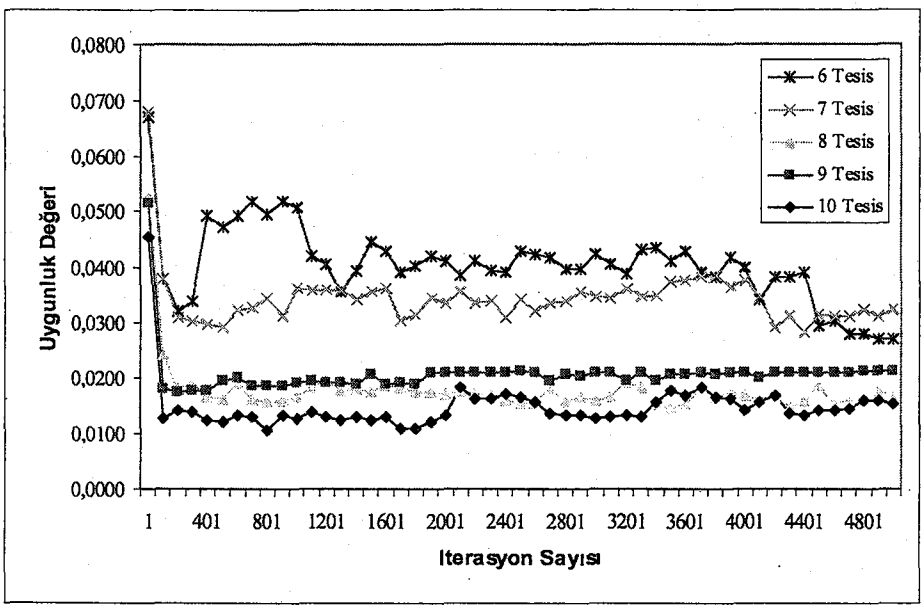
Farklı sayıda tesis (2'den 10'a kadar) için TGA ve SOM ile çözüm yapılmıştır. TGA küçük popülasyonlarda bile prematüre yakınsak çözümlere takılmadığından, deneyler aynı popülasyon büyüklüğü (n=6) ile gerçekleştirilmiştir. Sonuçta, TGA'nın sonuçları klasik GA ile karşılaştırıldığında hesaplama süresinin çok düşük olduğu ve yaklaşık 5000 nesilden sonra minimum uygunluk değerine yakınsadığı görülmüştür.

Çaprazlama oranı 1 ve mutasyon oranı titreşimli mutasyon tekniğinin periyoduna bağlı değer almıştır. Ebeveynler Evrensel Örnekleme'ye göre seçildi. Titreşimli mutasyon için Denklem 8.14'deki r değeri 5 olarak alınmıştır. Sunulan sonuçlar 10 farklı denemenin ortalamasıdır.

Şekil 8.6, farklı sayıda tesis (arz merkezi) için elde edilen sonuçların zamana bağlı olarak yakınsaması gösterilmiştir. TGA uygunluk değerini şekilde görüldüğü gibi başlangıçta çok hızlı biçimde azaltarak çok kısa bir periyot sonra kararlı duruma erişmektedir. Şekil 8.7'de, kromozomların her bir üremedeki ortalama uygunluk değerleri görülmektedir.

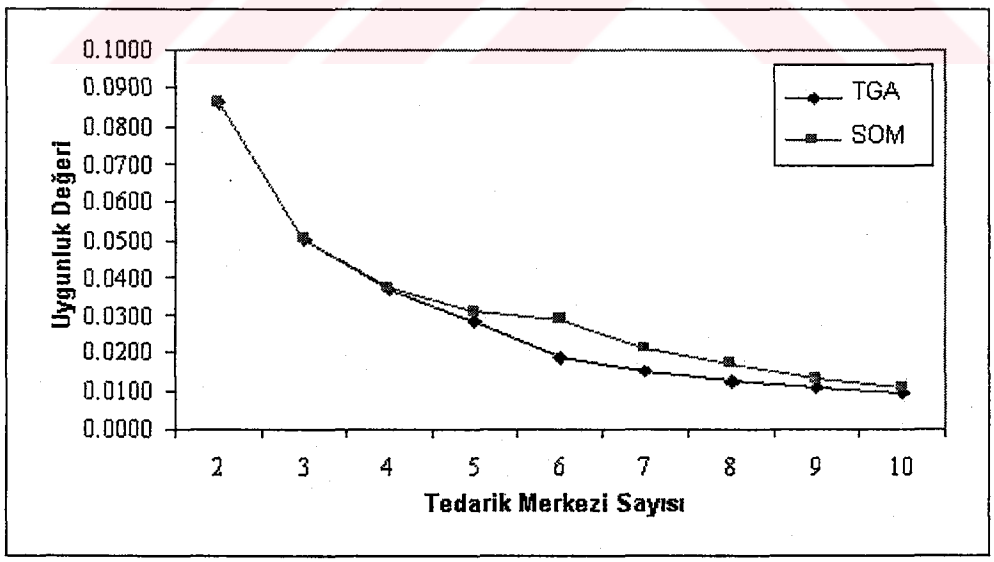


Şekil 8.6 : Farklı sayıda tesis için elde edilen sonuçların yakınsaması

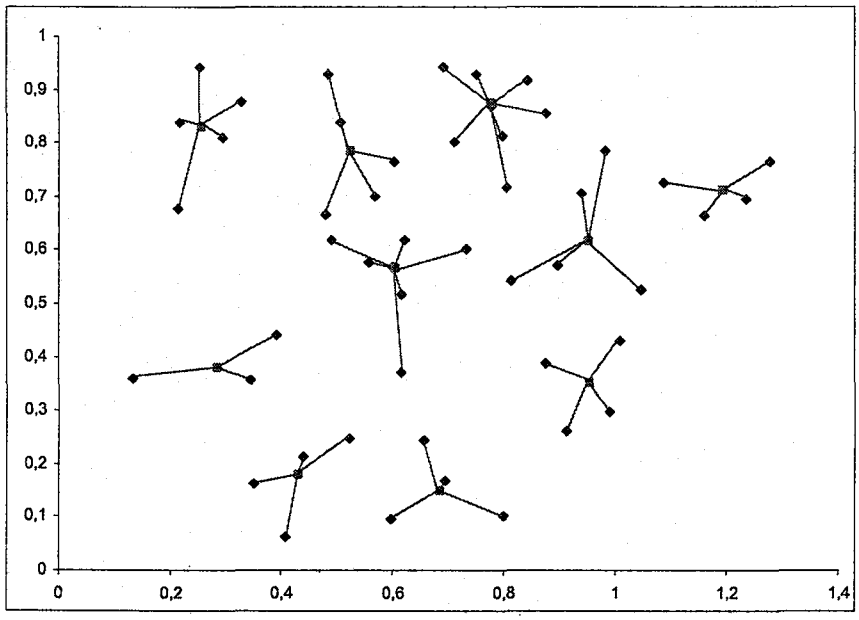


Şekil 8.7 : Farklı sayıdaki tesisin ortalama uygunluk değerleri (n=6)

Şekil 8.8'de, farklı tesis sayıları için SOM ve TGA kullanılarak elde edilen en iyi sonuçlar gösterilmiştir.

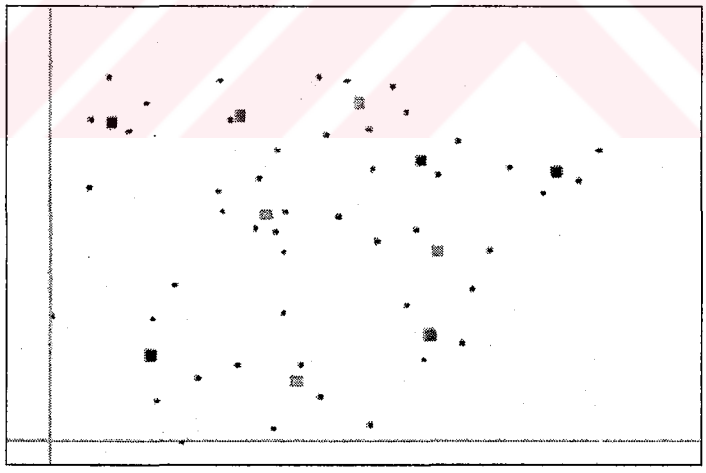


Şekil 8.8 : Farklı sayıdaki tesis için TGA ve SOM kullanılarak elde edilen en iyi sonuçlar



Şekil 8.9 : TGA kullanılarak bulunan 10 tesisin konumları ve talep merkezlerinin paylaşımı

Şekil 8.9'da TGA kullanılarak bulunan 10 tesisin konumu ve talep merkezlerinin paylaşımı görülmektedir. Aynı sayıda tesis için SOM kullanılarak bulunan konumlar ise Şekil 8.10'da görülmektedir.



Şekil 8.10 : SOM kullanılarak elde edilen 10 tesisin konumları

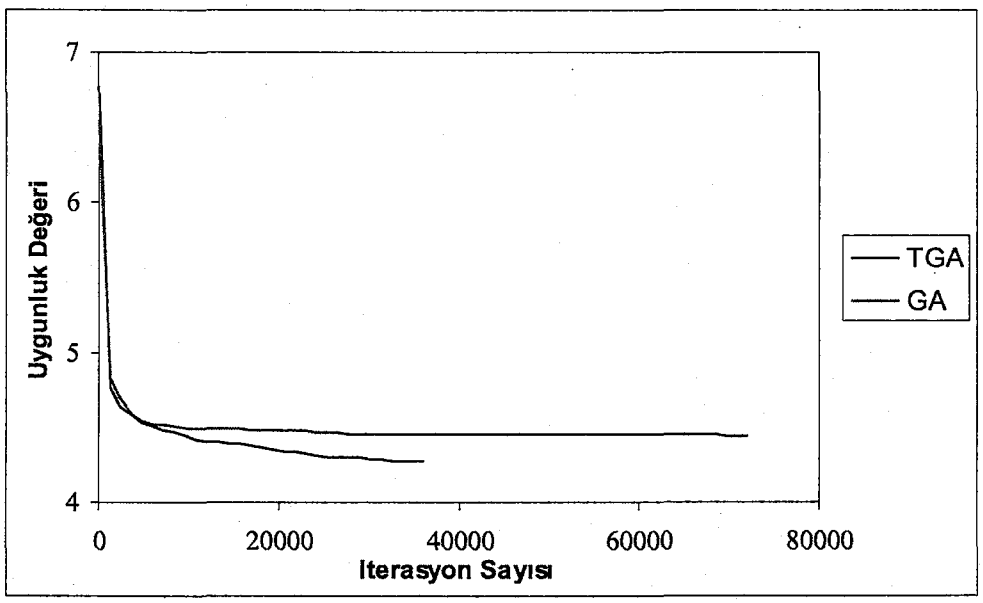
TGA'nın daha büyük problemlerdeki performansını görebilmek ve SOM ile karşılaştırmak amacıyla, Avustralya Posta İdaresine ait olan gerçek veriler (AP veri seti) kullanılmıştır (Ernst ve Krishnamoorthy, 1996). Bu veri seti posta merkezlerini gösteren 200 düğüm noktası (talep merkezi) içermektedir. n = 50, 100, 150, 200'lik problemler de veri setinden elde edilebilmektedir. Tablo 8.1'de, farklı sayıda talep

merkezleri (düğüm noktaları) ve tesisler için geleneksel GA, TGA ve SOM kullanılarak elde edilen sonuçları gösterilmiştir. Tablodaki sonuçlar, 36.000 işlemden (yani, 3.000 nesilden) sonra elde edilen en iyi uygunluk değerleridir.

Tablo 8.1 : Sonuçların karşılaştırılması (n = 12)

Talep merkezi sayısı	Tedarik merkezi sayısı	GA	TGA	SOM
50	6	5.68	5.38	5.63
	8	4.11	3.95.	4.14
	10	3.32	3.06	3.25
100	6	4.58	4.48	4.61
	8	3.94	3.88	3.97
	10	3.44	3.16	3.34
150	6	5.42	5.31	5.40
	8	4.56	4.48	4.53
	10	4.00	3.99	4.00
150	6	5.68	5.60	5.60
	8	4.87	4.70	4.73
	10	4.27	4.06	4.11

TGA ile karşılaştırıldığında geleneksel GA'nın çok daha fazla hesaplama zamanına gereksinim duyduğu görülmüştür. 200 talep merkezinin olduğu durumda TGA 9.612 hesaplamadan sonra yakınsamış, ancak GA, TGA ile bulunandan daha kötü olan kendisine ait en iyi uygunluk değerine 72.012 yinelemeden sonra ulaşabilmiştir (Şekil 8.11).



Şekil 8.11 : TGA ve GA ile hesaplanan en iyi uygunluk değeri için yineleme sayısı

8.2 Ayrık Konum-Tahsis Problemleri

Ayrık konum problemleri için kullanılan modellerden temel olan sekiz tanesi: örten küme, maksimal örten, p-merkez, p-yayılma, p-medyan, sabit maliyet, merkez üs ve maksisum'dır. İlk dördü maksimum, son dördü ise toplam (veya ortalama) mesafe kriterine göre ele alınır.

8.2.1 p-Medyan Problemi

8.2.1.1 Problemin Formülasyonu

Bu modelde, talep noktaları ile eşleştirildikleri tesisler arasındaki talebe göre tartılandırılmış toplam mesafeyi minimum yapan p tesisin konumu bulunmaya çalışılır.

i : i endeksi ile gösterilen talep noktaları kümesi

J : j endeksi ile gösterilen aday tesis yerleri kümesi

d_{ij} : i'nci talep noktası ile j'nci aday yer arasındaki mesafe

h_i : i'nci düğüm noktasındaki talep

p : yerleştirme yapılacak tesis sayısı

Karar değişkenleri:

$$x_j = \begin{cases} 1 & \text{Eğer j'nci konuma yerleştirilmişse} \\ 0 & \text{Aksi takdirde} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{Eğer i'nci talep noktası j'nci konuma yerleştirilmişse} \\ 0 & \text{Aksi takdirde} \end{cases}$$

0-1 Tamsayı Programlama Modeli:

$$\min \quad \sum_{i \in I} \sum_{j \in J} h_i d_{ij} y_{ij} \tag{8.15}$$

$$\text{s.t.} \quad \sum_{j \in J} x_j = p \tag{8.16}$$

$$\sum_{j \in J} y_{ij} = 1, \qquad \forall i \in I \tag{8.17}$$

$$y_{ij} - x_j \leq 0, \qquad \forall i \in I, j \in J \tag{8.18}$$

$$x_j \in \{0,1\}, \qquad \forall j \in J \tag{8.19}$$

$$y_{ij} \in \{0,1\}, \qquad \forall i \in I, j \in J \tag{8.20}$$

Amaç Fonksiyonu (8.15), talebe göre tartılandırılmış toplam ulaştırma mesafesini minimum yapmaya çalışır. Birinci kısıt (8.16), p tane tesisin konumlandırılmasını sağlar. Kısıt kümesi (8.17), her bir talep düğüm noktasının tek bir tesise atanmasını sağlar. Kısıt kümesi (8.18), talep noktasının sadece açık olan tesislere atanmasını sağlar. Kısıt (8.19), konum değişkeninin 0 veya 1 değeri almasını, Kısıt (8.20) bir düğüm noktasındaki talebin sadece bir tesis tarafından karşılanmasını sağlar. Ancak, Kısıt (8.20) yerine sadece $y_{ij} \geq 0$ yazılması da Kısıt (8.18)'den dolayı yeterli olur.

p-Medyan problemini çözmek çok kolay değildir. Sayısal örnekle gösterilmek istenirse, p-Medyan probleminde N müşteri (talep noktası) sayısı ve p de konumlandırılacak tesis adedi ise, olası çözümlerin sayısı:

$$\binom{N}{p} = \frac{N!}{p!(N-p)!} \tag{8.21}$$

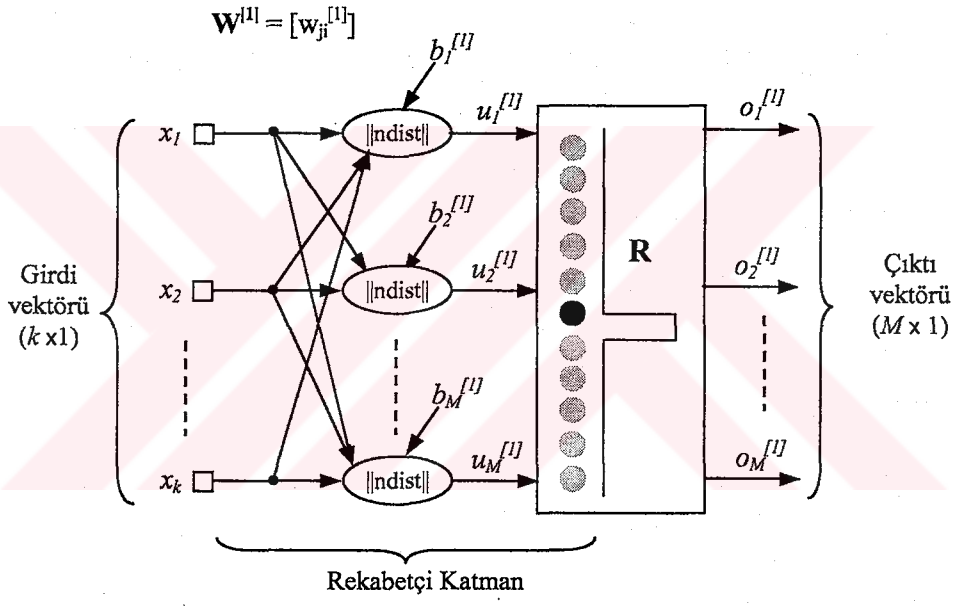
ile hesaplanabilir. Örneğin, N = 20 ve p = 5 için, hesaplanan olası alternatiflerin sayısı 15.504 olur. Çoğu standarda göre büyük problem kabul edilmeyen N = 50 ve p = 10 için ise hesaplanan olası alternatiflerin sayısı 10^{10}'dur. Sonuç olarak p-Medyan problemi, p'nin sabit bir değer almaması durumunda NP-sıkıdır (Garey ve Johnson, 1979). Bu nedenle, p-medyan problemini çözmek için pek çok sezgisel yaklaşım geliştirilmiştir. Bunlardan başlıcaları:

- Miyop / Açgözlü sezgisel yaklaşım
- İyileştirme
  - ⇒ Komşuluk (Maranzan, 1964)
  - ⇒ Değiş-tokuş (Tietz and Bart, 1968)
  - ⇒ Genetik Algoritma
- Lagranjın Gevşetme Yaklaşımı

olarak verilebilir.

8.2.1.2 p-Medyan Konum-Tahsis Problemlerinin Kohonen Ağları ile Çözülmesi

p-Medyan problemini Kohonen ağları ile çözmek için oluşturulan yapay sinir ağında girdi örüntüleri talep merkezlerinin k-boyutlu konum vektörüdür. Çıktı katmanı konumları belirlenecek M adet tesise karşılık gelir (Şekil 8.12). Öğrenme kuralının etkisi, tedarik merkezlerini talep merkezlerine doğru çekmektir. Ancak, sadece çok yakın tedarik merkezleri ve onların komşuları önemli ölçüde etkilenir. Başlangıçta, düzenleme evresini hızlandırmak için, çıktı birimlerinin çoğunun her bir yinelemede ağırlıklarını adapte etmesine izin verecek şekilde komşuluk genişliği d yeteri kadar büyük tutulmalıdır. Daha sonra hem d hem de öğrenme oranı η yavaşça azaltılır, daha az sayıdaki tedarik merkezlerinin konumu daha küçük oranlarda ayarlanır.



Şekil 8.12 : KT problemi için kullanılan Kohonen yapay sinir ağı

Çıktı katmanında İE'lerin düzenlenişi ve kullanılacak ölçüt, modelleyicinin girdi verilerinin topolojisini tahmin etmesine göre kararlaştırılır. Eğer mümkünse, farklı düzenlemeler denenerek en iyi sonucu veren seçilebilir. En iyi düzenleme probleme bağlı olarak değişebilir. Çünkü en iyi düzenleme sadece taleplerin uzaysal dağılımına değil, aynı zamanda tedarik merkezinin sayısına da bağlıdır.

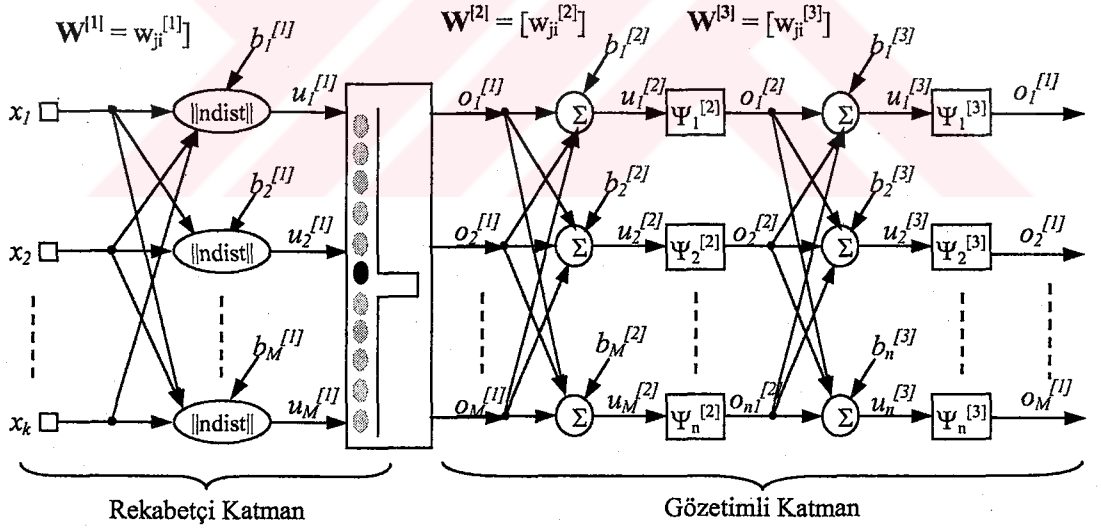
Orijinal modelde amaç fonksiyonu tartılandırılmış öklit mesafe fonksiyonunu minimum yapmak olsa da, tartılandırılmış düz uzaklığın karesini minimum yapmak için de bu yaklaşım kullanılabilir. Diğer bir olasılık, öğrenme kuralını girdi

örüntüsünün, ona ait tedarik merkezini itmesini sağlayacak şekilde uyarlamaktır. Böylelikle *maksisum* amaç fonksiyonu araştırılabilir.

8.2.1.3 Karşıtyayılım Ağları

SOM ağları temelde gözetimsiz öğrenmenin söz konusu kümeleme, verilerin görselleştirilmesi veya öznitelik çıkarımı gibi alanlarda iyi sonuçlar verir. SOM ağları, ÇKİB YSA veya ÖVN gibi gözetimli öğrenme ağlarına girdi olarak kullanılabilir. İlk olarak rekabetçi katman eğitilir ve ardından çıktı ağırlıkları, sınıf etiketleri kullanılarak En Küçük Kareler Ortalaması (EKKO) kuralına göre eğitilir. Bu ağların temel avantajı, kendini örgütleme süreci ile girdi uzayını azaltan SOM ağı kullanılarak kümeleme elde edilmesidir. Böylelikle girdi uzayının yapısı korunurken, boyutu küçültülmüş olur. Karşıtyayılım ağının ilginç bir özelliği, saklı katmanın sayısal aktivasyona (on/off) sahip olmasından dolayı, EKKO ağırlık güncellemesinin aşağıdaki gibi yazılabilmesidir.

$$\Delta w_{ij} = \eta(d_i - y_i)x_j = \eta(d_i - w_{ij})x_j \tag{8.22}$$



Şekil 8.13 : Modellenen karşıtyayılım ağının yapısı

Yani, çıktı katmanındaki ağırlıklar belirli bir çokgenler bölgesi için sınıf etiketinin ortalama değeridir. Karşıtyayılım ağları bütünüyle bir vektör nicelendiriciyi gerçekleyebilir. Bu ağ mimarisinde gözetimsiz kısım kümelemeyi gerçekleştirir,

gözetimli kısım ise alıcıya gönderilecek kodu uygulayabilir (Şekil 8.13). Bu ağlar ÇKİB YSA'ya kıyasla çok daha hızlı eğitilir.

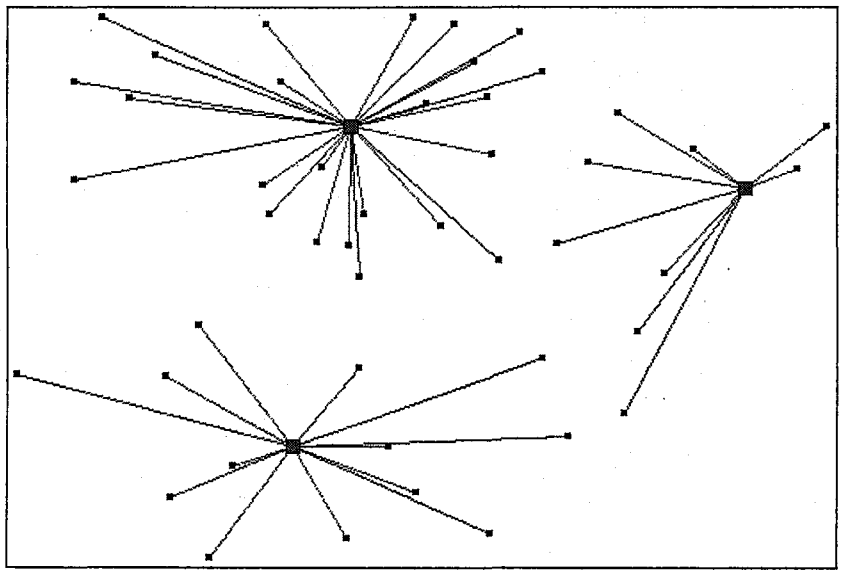
Yapay sinir ağının gözetimsiz (rekabetçi) bölümünde eğitmeyi sonlandırmak için ya maksimum döngü sayısı belirlenir veya ağırlıklardaki değişim oranına bakılır. Başlangıçta maksimum döngü sayısı, eğitimin erken sona ermeyeceği büyük bir değere eşitlenir. Ardından, ağırlık değişim oranına göre sonlandırma aktive edilir. Bu durumda, bütün ağırlıklardaki değişim bir döngüden diğerine önceden belirlenmiş bir değerden küçükse eğitme işlemi sonlandırılır.

Diğer gözetimsiz öğrenme parametreleri öğrenme oranını belirler. Gözetimsiz öğrenme genellikle öğrenme oranı yüksek bir değere eşitlenip eğitme süresince adım adım azaltıldığında çok iyi performans gösterir. Böylelikle sinir ağı yaklaşık bir çözümü çabucak bulur ve ardından problemin detaylarına odaklanılır. Gözetimli katmanda performans kriteri olarak hata kareleri ortalaması kullanılır. Öğrenme yaklaşımı olarak da çevrim içi veya çevrim dışı yaklaşımlar kullanılabilir.

8.2.1.4 Deneyler ve Elde Edilen Sonuçlar

Bu bölümde daha önce anlatılan yaklaşımlar kullanılarak çözülen p-medyan problemi için elde edilen sonuçlar açıklanacaktır. Çözülen test probleminde 49 talep merkezi yer almaktadır ve bu merkezlerin konumları ile talepleri EK D'de verilmiştir.

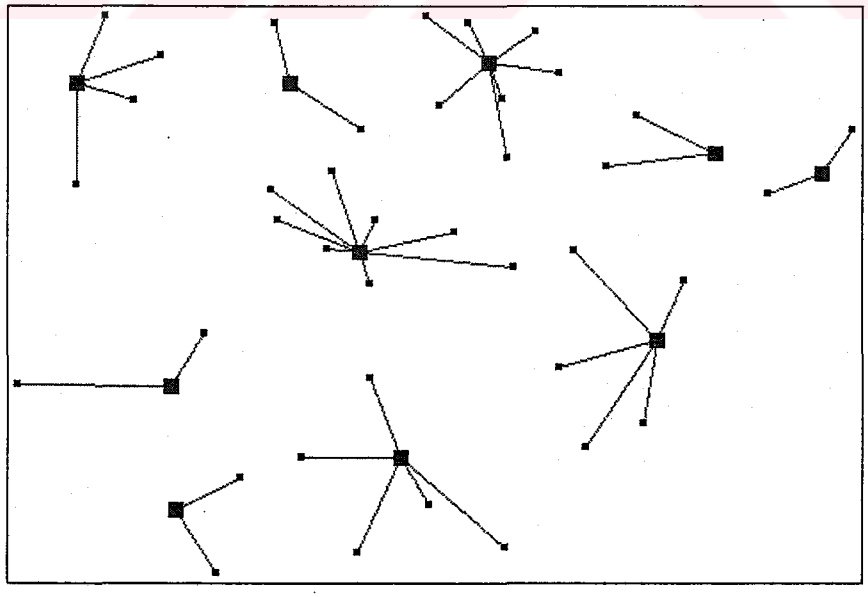
Problem öncelikle klasik çözüm teknikleri ve Genetik Algoritma (GA) ile çözülmüştür. Daha sonra, bir önceki bölümde anlatılan yaklaşımlar kullanılarak SOM ve Karşıtyayılım yapay sinir ağı modelleri gerçeklenmiş ve elde edilen sonuçlar klasik yöntemler ve GA için elde edilenlerle karşılaştırılmıştır.Şekil 8.14'de Lagranjın gevşetme yaklaşımı kullanılarak elde edilen, tartılandırılmış ortalama mesafe ile tesis sayısı arasındaki ilişkiyi göstermektedir. Şekil 8.15'de ise, GA yaklaşımı ile 10 tesis yeri için çözülmüş olan p-medyan problemi için elde edilen tesis konumları grafik olarak gösterilmektedir.



Şekil 8.14 : Lagranjın gevşetme yaklaşımı ile üç tesis yeri için elde edilen sonuç

Genetik Algoritma ile çözümde kullanılan parametreler:

- *Popülasyon büyüklüğü* : 50
- *İyileşme gerçekleşmeyen nesil sayısı* : 50
- *Minimum uygunluk* : 0,1
- *Mutasyon olasılığı* : 0,25

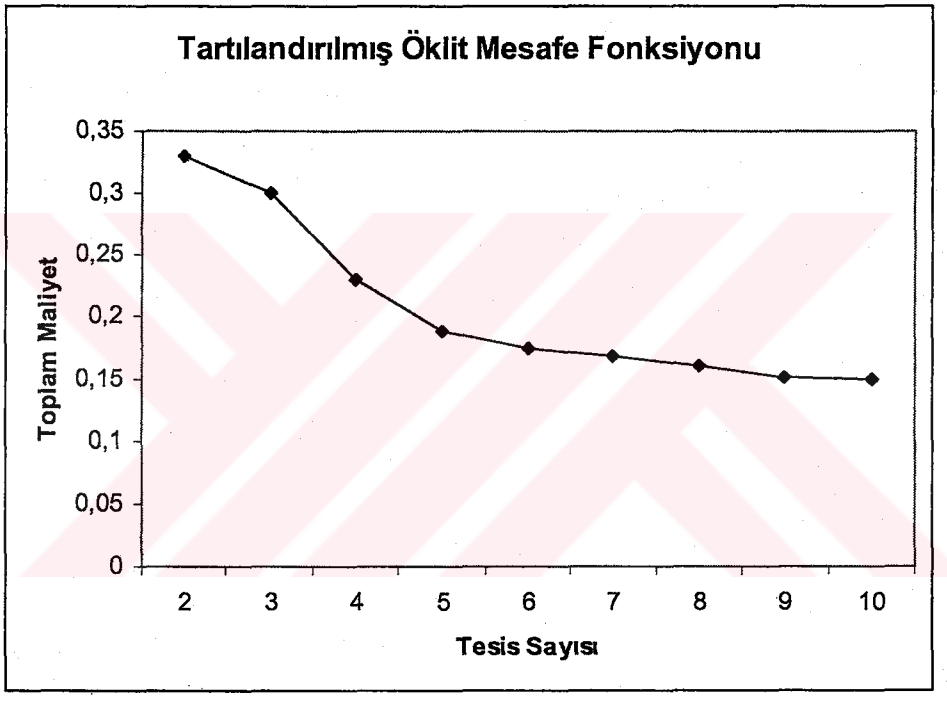


Şekil 8.15 : GA yaklaşımı ile 10 tesis yeri için elde edilen sonuç

SOM sinir ağı benzetim modelindeki başlangıç parametreleri:

- *Adım büyüklüğü* : 0.01
- *Başlangıç komşuluk yarıçapı* : 7
- *Soğutma fonksiyonu* : Üssel
- *Soğutma çizelgesi parametresi (β)* : 0.98
- *Döngü sayısı* : 400
- *Örnek sayısı* : 49

KT probleminin SOM benzetimi yapıldığında, farklı tedarik merkezleri için elde edilen toplam ulaştırma maliyetleri Şekil 8.16'da görüldüğü gibi bulunmuştur.



Şekil 8.16 : *p*-Medyan probleminde 2 - 10 tedarik merkezi için SOM YSA'dan elde edilen Toplam Maliyet

Karşıtyayılım sinir ağı benzetim modelinde kullanılan başlangıç parametrelerine göre toplam ulaştırma maliyetleri Şekil 8.17'de görüldüğü gibi bulunmuştur.:

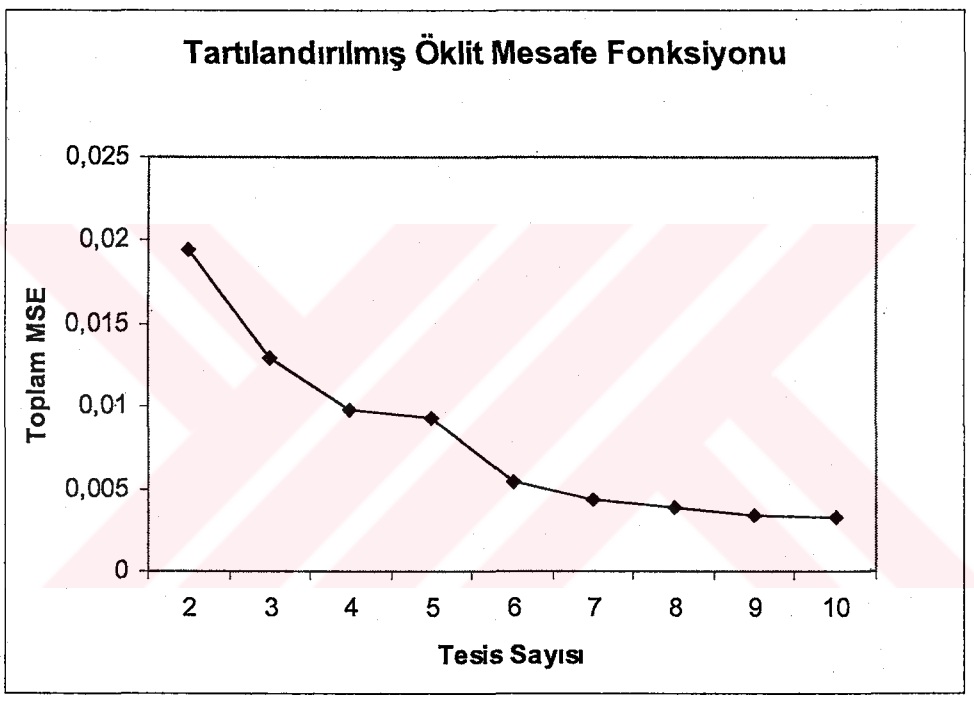
Gözetimsiz Katman:

- *Adım büyüklüğü* : 0.01
- *Başlangıç komşuluk yarıçapı* : 7
- *Soğutma fonksiyonu* : Doğrusal
- *Soğutma çizelgesi parametresi (β)* : 0.02-0.002

- *Döngü sayısı* : 400
- *Örnek sayısı* : 49

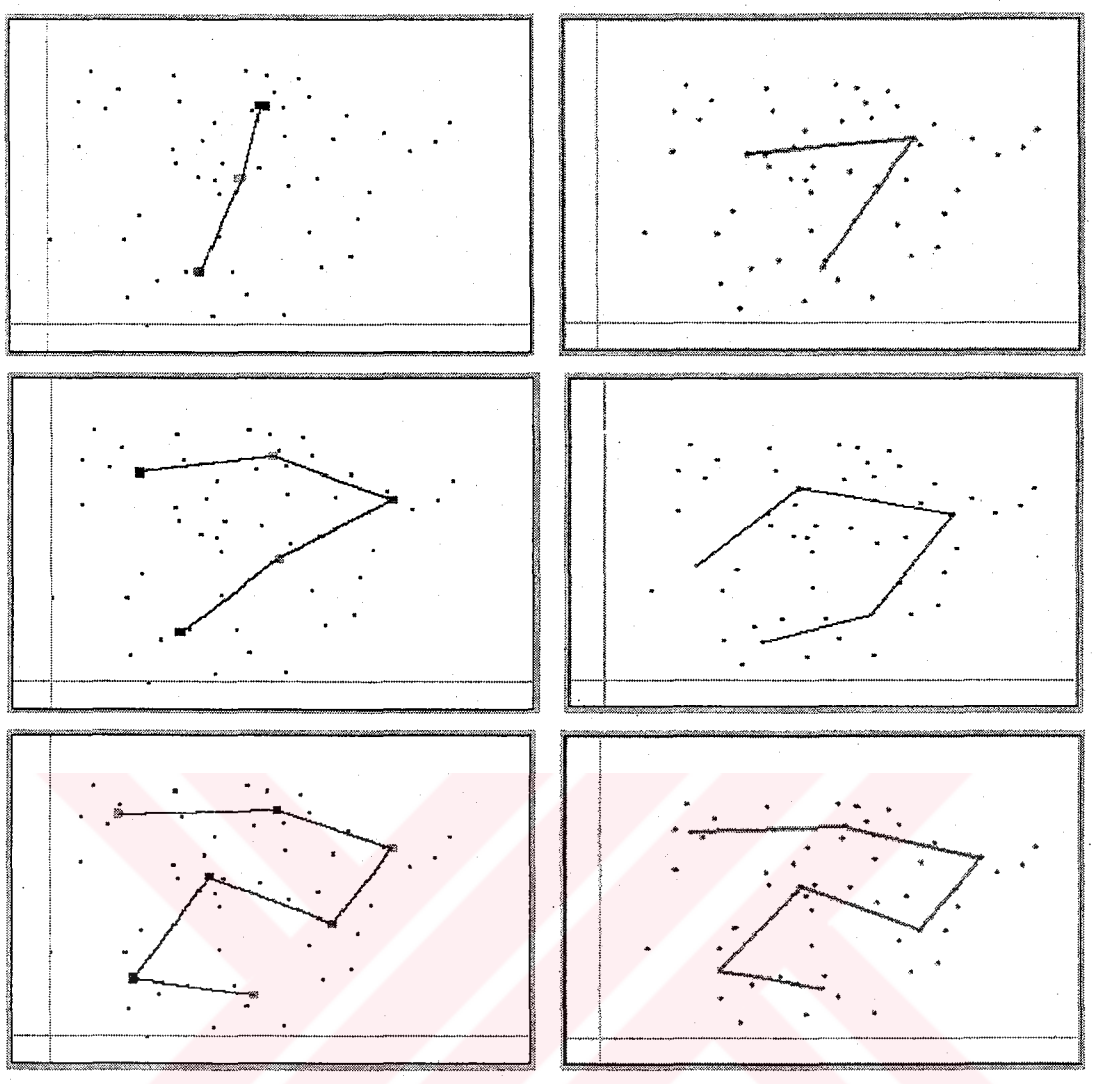
Gözetimli Katman:

- *Yapay sinir ağının yapısı* : M -1 işlem elemanı
- *Öğrenme tekniği* : Geriye Yayılım + ADIM (oran = 0.7)
- *Performans Kriteri* : HKO
- *Eğitme oranı* : 0.1
- *Döngü sayısı* : 400
- *Örnek sayısı* : 49



Şekil 8.17 : p-Medyan probleminde 2 - 10 tedarik merkezi için Karşıt-Yayılım YSA'dan elde edilen HKO

Kohonen ve Karşıtyayılım sinir ağları kullanılarak 3,5 ve 7 tedarik merkezi için elde edilen çözümler Şekil 8.18'de görülmektedir. İki model arasında elde edilen tedarik merkezi konumları açısından farklılıklar görülmektedir. Şekil 8.18'de 1-boyutlu SOM dönüşümü kullanıldığından tedarik merkezleri çizgisel olarak birleştirilmiştir. Tedarik merkezlerinin konumları bulunmakla birlikte şekil üzerinde tahsisler gösterilmemektedir. Basit olarak her bir talep merkezi en yakınındaki tedarik merkezine atanabilir.



(a) (b)

Şekil 8.18 : p-Medyan probleminde 3,5 ve 7 tedarik merkezi için doğrusal SOFM (a) ve Karşıt Yayılım (b) YSA'dan elde edilen çözümler

8.2.2 Merkez Üslerin Konumlandırılması Problemi

KT problemlerinin özel bir çeşidi olan merkez üs (*hub*) konumu araştırmaları son yıllarda konum teorisinin önemli bir araştırma alanı olmuştur. Modern ulaştırma ve telekomünikasyon sistemlerinin büyük bir kısmında merkez üsler ağlarının kullanılması bu alandaki araştırmaların öneminin artmasında etken olmuştur. Bu sistemler, maliyette ölçek ekonomisinin söz konusu olduğu birçok hareket ve varış noktaları arasında seyahat veya iletişimin gerçekleşmesini sağlar. Özellikle ulaştırmada daha büyük araçlar birim başına daha düşük taşıma maliyetine sahiptirler. Örneğin, büyük bir uçakta (mesela, Airbus 340) mil başına bir koltuğun maliyeti, küçük bir uçağın (mesela, RJ 70) mil başına koltuk maliyetinden daha düşüktür.

Merkez üsler konum problemleri birçok açıdan klasik tesis konum problemlerinden farklılık gösterir. Klasik bir ayrık tesis konum probleminde, talep ayrık noktalarda gerçekleşir, tesisler ayrık noktalarda konumlandırılmıştır ve amaç genellikle talep noktaları ile tesisler arasındaki mesafe ve maliyetle ilgilidir. Merkez üsler konum problemlerinde ise, birbirleriyle iletişim halinde olan düğümlerden oluşmuş bir ağ söz konusudur. İletişimden kastedilen, birçok hareket ve varış noktaları arasındaki akıştır ve merkez üsler hareket-varış noktası akışları için aktarma ve birleştirme noktaları gibi hizmet verirler. Bir merkez üs, birçok ayrı küçük akışı daha büyük akışlara yönlendirir veya birleştirirler. Ayrıca, farklı varış noktaları için büyük bir akışı daha küçük akışlara da bölebilirler. Bu nedenle, bağlantı noktaları hareket-varış noktası akışlarının takip ettiği yoldaki ara noktalardır (Drezner ve Hamacher, 2002). Merkez üs konum problemlerinin ulaştırma (hava yolu ile seyahat, hava yolu ile nakliyat, bir gecelik dağıtım sistemleri, posta dağıtımı, vs.) ve telekomünikasyon (bilgisayar iletişimi, telefon ağları, dağıtık bilgisayar işleme vs.) alanında birçok uygulamaları vardır.

Merkez üs konum problemlerinin pek çok farklı türü vardır (O'Kelly, 1987). Bir merkez üssün toplayabileceği akışın miktarına ilişkin bir kapasite kısıtı olabilir; herhangi bir düğüm noktasını merkez üs olarak kurulmasında sabit bir maliyet olabilir veya düğüm noktaları bir veya birden fazla merkez üsse paylaştırılabilir. Ancak, merkez üs problemlerinin bütün çeşitlerinde amaç fonksiyonu, ağın toplam maliyeti enküçüklenecek şekilde bağlantı noktalarının konumlarını ve düğüm noktalarının (*spoke*) paylaşımını bulmaktır. Eğer bütün akışlar merkez üs aracılığıyla yapılmakta ve merkez üs olmayan her bir düğüm noktası sadece bir merkez üsse tahsis edilmişse, bu problem kapasite sınırlaması olmayan tek tahsisli merkez üs konumlandırma problemi (*uncapacitated single allocation hub location problem* - USAHLP) adını alır. Bu problemde, merkez üs sayısı karar değişkenidir ve matematiksel formülasyona sabit bir maliyet de dahil edilmiştir. Eğer merkez üs sayısı sabit ise (p merkez üs), bu problem kapasite sınırlaması olmayan tek tahsisli p-merkez üs problemi (*uncapacitated single allocation p-hub median problem*- USApHMP) olarak isimlendirilmiştir. Bağlantı noktalarındaki akış miktarına ilişkin bir sınırlandırma da olabilir (*capacitated* - CMAHLP) veya her bir düğüm noktası birden fazla merkez üsse tahsis edilmiş de olabilir (*multiple allocation* - UMApHMP). Diğer bir değişik biçimi ise, bugüne kadar kapsamlı olarak çalışılmamış olan, bazı bağlantılarda veya tamamında minimum akışa gereksinim duyulan (akış eşikleri) modellerdir. Örneğin, çoklu tahsisin söz konusu olduğu havayolları uygulamasında, şirket tarafından işletilen en küçük uçak büyüklüğüne ilişkin akış eşiği, ağda yer alıp

da ekonomik olmayan bağlantıları önlemeyi sağlayabilir. KT problemi p-medyan ve p-merkez problemleri gibi zor problemler olarak bilinirler. Konumlar bilindiğinde diğer problemler sıradanlaşırken, merkez üs problemi bu durumda dahi zor problem olarak kalır (Love ve diğ., 1988).

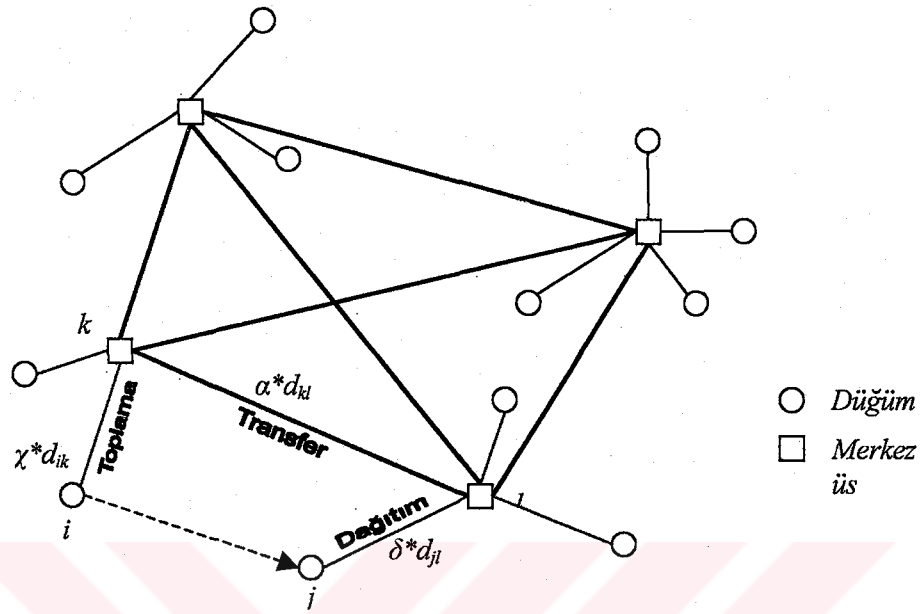
Bu güne kadar araştırmacıların yoğun ilgisini çeken merkez üs problemleri p-merkez üs medyan problemi ve sonlu-sonsuz kapasiteli merkez üs problemleridir. Merkez üs problemi ikinci dereceden tamsayı programlama olarak ilk kez O'Kelly tarafından modellenmiştir (O'Kelly, 1987). Tek tahsisli durumu Aykin (1994), Campbell (1994, 1996), Klincewicz (1991), O'Kelly ve diğ.(1995) tarafından çalışılmıştır. Her bir durum farklı yollardan formüle edilebilmekle beraber, en etkin genel bir yaklaşım Ernst ve Krishnamoorthy tarafından ortaya konmuştur (1996, 1998). Bu yaklaşımda her bir orijin için arklardaki akışlara dayanan bir yöntem önerilmiştir.

O'Kelly, USAHLP'yi çözmek için, bir havayolu şirketinin uçuş rotalarını modelleyen ve her bir hava limanına belirli sayıda merkez üsse atamaya çalışan iki sezgisel yöntemi ele almıştır (O'Kelly, 1995). Klincewicz Konum-Tahsis problemlerini ele alarak p-merkez üslerin bütün olası kombinasyonları deneyen ve her bir durum için en küçük mesafeye sahip düğüm noktasını atayan bir sezgisel yöntem kullanmıştır (Klincewicz, 1991). Yasaklı aramayı konum-tahsis problemlerinin çözümünde ilk olarak Skorin-Kapov ve Skorin-Kapov kullanmıştır (Skorin-Kapov ve Skorin-Kapov, 1994). Kullanılan bir diğer yaklaşım ise tavlama benzetimi tekniğidir (Ernst ve Krishnamoorthy, 1996). Çözüm kalitesinin tahmini bir değerini verebilecek alt sınırlar oluşturmada referans çözümü kullanan ilginç bir sezgisel çözüm tekniği de O'Kelly tarafından önerilmiştir (O'Kelly ve diğ., 1995). Abdinnour-Helm ise USAHLP için iyi çözümler elde etmede genetik algoritmalar ile yasaklı aramayı bir arada kullanan melez bir sezgisel yaklaşım kullanmıştır (Abdinnour-Helm, 1998). Smith ise sonsuz kapasiteli, tek paylaşımlı p-merkez üs medyan problemini çözmek için yapay sinir ağları yaklaşımını önermiştir (Smith ve diğ., 1996). Topçuoğlu ise USAHLP için bu güne kadar literatürde yer alan en iyi çözümleri veren genetik algoritmalara dayanan bir sezgisel yaklaşım önermiştir (Topçuoğlu ve diğ., 2005).

8.2.2.1 Problemin Formülasyonu

Merkez üs konumuyla ilgili mevcut temel bilgiler, her bir düğüm noktası çifti (i, j) arasında bilinen bir W_{ij} miktarındaki akışın (yolcu, veri, posta, bilgi, vs.)

gerçekleşmesi için ihtiyaç duyulan n düğüm noktasıdır. Bunu gerçekleştirmenin en basit yolu her bir düğüm noktası çiftini doğrudan birbirleriyle birleştirmektir, ancak bu çok verimsiz bir yöntemdir. Verimliliği artırmak için bütün akışların Şekil 8.19'da görüldüğü gibi bağlantı noktaları aracılığıyla gerçekleştirilmesi gerekir.



Şekil 8.19 : Merkez üs ağında toplama ve dağıtım süreci

Bağlantı noktalarının konumları orijinal düğüm noktaları kümesinden seçilmelidir. Genellikle bağlantı noktalarını tamamen birbirlerine bağlıdır ve transfer maliyetleri üçgen eşitsizliğini sağlar. Benzer şekilde, merkez üs olmayan herhangi bir düğüm noktası doğrudan bir merkez üsse bağlanmalıdır. Bu varsayımlar ve sınırlamalarla bütün transferler bir veya en fazla iki merkez üs üzerinden rotalanmalıdır. Genellikle bu akış, i'den j'ye k ve l bağlantı noktaları üzerinden olan transfer eğer k ve l eşitse, tek bir merkez üs üzerinden gerçekleşen akış olarak yazılır. Benzer şekilde eğer i bir merkez üs ise $i = k$ ve eğer j bir merkez üs $l = j$ olarak yazılır. Bu bağlantıların ulaştırma maliyeti, i'den k'ya olan toplama, k'dan l'ye transfer ve l'den j'ye dağıtım maliyetlerinin toplamına eşittir. Eğer düğüm noktalarından ikisi birbirinin aynısı ise bu maliyetlerden bazıları sıfıra eşit olabilir.

USA*p*HMP için ilk tamsayı programlama (*IP*) formülasyonu O'Kelly tarafından önerilmiştir ve bu modelde amaç fonksiyonu ikinci derecedendir (O'Kelly, 1987). Böyle bir ikinci dereceden formülasyonun avantajı, sadece N^2 ikili değişkene ve $(1+N+N^2)$ kısıta gereksinim duymasıdır. Dezavantajı ise, ikinci dereceden

formülasyonun doğasından gelen genel minimuma ulaşmanın garanti olmamasıdır. Ayrıca bu problem için karışık tamsayı programlama (MILP) formülasyonu da geliştirilmiştir (Campbell, 1994; Ernst, 1999; Ebery, 2001). İkinci dereceden tamsayı programlama modeli Yapay Sinir Ağı çatısına dönüştürmeye uygun olduğundan, bu çalışmada bazı notasyon değişiklikleri yaparak O'Kelly'nin formülasyonu kullanılmıştır.

Eğer i'nci düğüm noktası k'ncı düğüm noktasına konumlandırılmış merkez üsse tahsis edilmişse, tamsayı karar değişkeni X_{ik} 1'e, aksi takdirde 0'a eşit olacaktır. Her bir merkez üs kendi kendine atanmıştır; yani, k'ncı düğüm noktası eğer bir merkez üs ise X_{kk} 1'e eşittir. d_{ik} i'nci ve k'ncı düğüm noktaları arasındaki mesafe, W_{ij} i'nci düğüm noktasından j'nci düğüm noktasına olan akış miktarı ve χ, α ve δ toplama, transfer ve dağıtım birim maliyetleri olsun. Bir merkez üssü k'ncı düğüm noktasına kurmanın maliyeti de f_k olsun. Literatürde, bağlantı noktaları arasındaki indirgenmiş birim maliyeti elde edebilmek için bir iskonto faktörü olarak $\alpha < 1.0$ ve $\chi = \delta = 1.0$ değerleri kabul edilmiştir.

Bu tanımlamalar kullanılarak USAHLP, 0-1 ikinci dereceden programlama olarak aşağıdaki gibi modellenebilir:

$$\min f(x) = \sum_i \sum_j \sum_k \chi W_{ij} d_{ik} X_{ik} + \sum_i \sum_j \sum_l \chi W_{ji} d_{jl} X_{jl} \qquad (8.23)$$

$$\sum_i \sum_k \sum_l \sum_j \alpha W_{ij} d_{kl} X_{ik} X_{jl} + \sum_k f_k X_{kk}$$

Kısıtlar;

$$\sum_k X_{ik} = 1, \qquad \forall i \in N \qquad (8.24)$$

$$X_{ik} - X_{kk} \le 0, \quad \forall i,k \in N \qquad (8.25)$$

$$X_{ik} \in \{0,1\}, \qquad \forall i,k \in N \qquad (8.26)$$

Orijin i'den dışarıya olan toplam akış miktarı O_i ise:

$$O_i = \sum_j W_{ij}, \qquad \forall i \in N \qquad (8.27)$$

ve D_i varış noktası i'de sonlanan toplam akış miktarı ise:

$$D_i = \sum_j W_{ji}, \quad \forall i \in N \tag{8.28}$$

Eğer maliyet matrisi simetrik ise (yani, $d_{ik} = d_{ki}$), amaç fonksiyonu (8.23) aşağıdaki gibi yazılabilir:

$$f(x) = \sum_i \sum_j X_{ij} d_{ij} (\chi O_i + \delta D_i) + \sum_i \sum_k \sum_l \sum_j \alpha W_{ij} d_{kl} X_{ik} X_{jl} + \sum_k f_k X_{kk} \tag{8.29}$$

Amaç fonksiyonu (8.23), toplama, transfer ve dağıtım maliyetlerinin toplamına eşittir. Kısıt (8.24), her bir düğüm noktasının sadece tek bir merkez üsse tahsis edilmesini garanti altına alır. Kısıt (8.25), her toplama ve dağıtım hareketi için bağlantı noktalarının tespit edilmesini sağlar. Dördüncü ve son kısıt (8.26), bütün değişkenlerin 0-1 tamsayı değeri almaya zorlar.

8.2.2.2 Hopfield Sinir Ağı Modeli

USAHLP, amaç fonksiyonu ve kısıtlar X çözüm matrisi değil de x çözüm vektörü gibi yazılarak standart ikinci dereceden programlama biçimine dönüştürülebilir. $vec(x)$, nxn'lik matris $\mathbf{X}$'i mn - elemanlı vektör $\mathbf{x}$'e dönüştüren fonksiyon olsun. Bu fonksiyon şu şekilde tanımlanabilir:

$$\mathbf{x} = vec(\mathbf{X}) = [X_{11}, X_{21}, ..., X_{n1}, X_{12}, X_{22}, ..., X_{n2}, ..., X_{1n}, X_{2n}, ..., X_{nn}]^T \tag{8.30}$$

ve c aşağıdaki gibi tanımlanır:

$$\mathbf{c} = [d_{11}(O_1 + D_1) + f_1, d_{21}(O_1 + D_1), ..., d_{n1}(O_1 + D_1), ...$$
$$...., d_{1n}(O_n + D_n), d_{2n}(O_n + D_n), ..., d_{nn}(O_n + D_n) + f_n]^T \tag{8.31}$$

(8.28)'deki ikinci dereceden terim $\mathbf{x}^T \alpha \mathbf{d}^T \mathbf{x} \mathbf{W}$ gibi gösterilebilir. Bu ifadede, $\mathbf{d}$ maliyetlerin (d_{ij}) oluşturduğu matris, $\mathbf{W}$ akışların (W_{ij}) oluşturduğu matristir. Bu ifade $\mathbf{x}^T(\alpha \mathbf{d}^T \otimes \mathbf{W})\mathbf{x}$'e eşittir ve $\otimes$ Kronecker çarpımıdır (Gee ve diğ., 1991).

$$\alpha \mathbf{d}^{\mathrm{T}} \otimes \mathbf{W} = \alpha \left[\begin{array}{ccc|ccc|c|ccc} d_{11}W_{11} & \cdots & d_{11}W_{1n} & d_{21}W_{11} & \cdots & d_{21}W_{1n} & & d_{n1}W_{11} & \cdots & d_{n1}W_{1n} \\ \vdots & & \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\ d_{11}W_{n1} & \cdots & d_{11}W_{nn} & d_{21}W_{n1} & \cdots & d_{21}W_{nn} & & d_{n1}W_{n1} & \cdots & d_{n1}W_{nn} \\ \hline d_{12}W_{11} & \cdots & d_{12}W_{1n} & d_{22}W_{11} & \cdots & d_{22}W_{1n} & & d_{n2}W_{11} & \cdots & d_{n2}W_{1n} \\ \vdots & & \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\ d_{12}W_{n1} & \cdots & d_{12}W_{nn} & d_{22}W_{n1} & \cdots & d_{22}W_{nn} & & d_{n2}W_{n1} & \cdots & d_{n2}W_{nn} \\ \hline \vdots & & \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\ \hline d_{1n}W_{11} & \cdots & d_{1n}W_{1n} & d_{2n}W_{11} & \cdots & d_{2n}W_{1n} & & d_{nn}W_{11} & \cdots & d_{nn}W_{1n} \\ \vdots & & \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\ d_{1n}W_{n\cdots 1} & \cdots & d_{1n}W_{nn} & d_{2n}W_{n1} & \cdots & d_{2n}W_{nn} & & d_{nn}W_{n1} & \cdots & d_{nn}W_{nn} \end{array}\right]$$

USAHLP çözüm vektörü ($\mathbf{x}$) cinsinden yeniden şu şekilde ifade edilebilir:

$$\min f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} \tag{8.32}$$

s.t.

$$\mathbf{A}\mathbf{x} \le \mathbf{b} \tag{8.33}$$

$$x_i \in \{0,1\}, \qquad \forall i \tag{8.34}$$

Öyle ki; $\mathbf{Q}$, $\mathbf{d}$ ve $\mathbf{W}$'nin Kronecker çarpımı ($=\alpha\ \mathbf{d}^{\mathrm{T}}\otimes\mathbf{W}$), $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ ise bir önceki formülasyondan elde edilen değerlerdir. Bu genel problem için uygun enerji fonksiyonu Aiyer ve Gee tarafından gerçekleştirilen çalışmalar temel alınarak şu şekilde verilebilir:

$$E = f(\mathbf{x}) + \frac{1}{2}c_0 E^g \tag{8.35}$$

Bu eşitlikte E^g terimi $\mathbf{x}$ vektörünün kısıt düzlemi $\mathbf{Ax} = \mathbf{b}$'den sapma miktarına eşittir. Bu değişikliğin avantajı sadece tek bir ceza parametresinin (c_0) seçilmesine gereksinim duymasıdır. Eğer c_0 yeteri kadar büyükse, enerjinin enküçüklenmesi esnasında kısıt terimini ortadan kalkmaya zorlayacağından, çözümün geçerliliği sağlanır. Bu nedenle, çözüm ister istemez kısıt düzleminde yer alacaktır.

8.2.2.3 Değiştirilmiş Hopfield Tank Enerji Fonksiyonu

$\mathbf{x}$ vektörünün kısıt düzleminin çözüm uzayına projeksiyonu yapılırsa enerji fonksiyonu (8.35) aşağıdaki gibi yeniden ifade edilebilir:

$$E = -\frac{1}{2}\mathbf{x}^T(-2\mathbf{Q} + c_0(\mathbf{P} - \mathbf{I}))\mathbf{x} - \mathbf{x}^T(c_0\mathbf{s} - \mathbf{c}) + \frac{1}{2}c_0\mathbf{s}^T\mathbf{s} \quad (8.36)$$

E standart Hopfield enerji fonksiyonu ile karşılaştırıldığında, ağın ağırlıkları ($\mathbf{W}$) ile harici girdilerinin ($\mathbf{I}$):

$$\mathbf{W} = -2\mathbf{Q} + c_0(\mathbf{P} - \mathbf{I}) \quad (8.37)$$

$$\mathbf{I} = c_0\mathbf{s} - \mathbf{c}$$

şeklinde olduğu görülebilir.

Bu enerji fonksiyonu ve Bölüm 5.5'de anlatılan yöntem kullanılarak değiştirilmiş Hopfield ağı Merkez Üslerin Konumlandırılması problemini çözmek için kullanılabilir. Bölüm 5.5'de anlatılan yöntem çözümün olurluluğunu garanti etmekle beraber çözüm kalitesi açısından yetersiz kalabilir. Çözüm kalitesini artırmak için geçici olarak enerji düzeyinin artışına izin veren ve böylelikle enerji fonksiyonunun yerel minimumlarından kurtulmasını sağlayan HN* ağı da kullanılabilir (Bölüm 5.6).

Algoritma:

Adım 1: Ağ parametrelerini ($\mathbf{W}$, $\mathbf{I}$, $\mathbf{x}$, Δt ve β) başlangıç durumuna getir.

Adım 2: HN* için; $k(t)$ = 1- 2exp(-t/β) olarak güncelle ve $\alpha(t)$'yi [$k(t)$, 1] aralığından rasgele seç.

Adım 3: Nöronları;

$$x_{i,new} = x_i - \alpha(t)\frac{\partial f}{\partial x_i}\Delta t$$

olarak güncelle ($\mathbf{x}$, büyük olasılıkla kısıt düzleminin dışına çıkacaktır)

Adım 4: $\mathbf{x}$'i aşağıdaki yinelemeli prosedüre göre kısıt düzlemine projeksiyonunu yap:

```
While (‖x − (℘x + s)‖² > limit)
    x ← ℘x + s
    For i = 1 to N
        x_i = g(u_i)
Endwhile
```

Adım 5: t'yi artır. Eğer $k(t)$ = 1 ve $\frac{dx_i}{dt} = 0, \forall i$ ise DUR. Yoksa Adım 2'ye git.

8.2.2.4 Deneyler ve Elde Edilen Sonuçlar

Literatürde Sivil Havacılık Örgütü (Civil Aeronautics Board - CAB) veri seti merkez üs konumlandırma problemi algoritmalarının etkinliklerinin değerlendirilmesinde sıklıkla kullanılmıştır (O'Kelly, 1987). Bu veri setinde şehirler arasındaki akışlar simetriktir ve 25 şehirlik veriden n = 10'luk bir problem seti oluşturularak bu çalışmada kullanılmıştır. Problem setinde farklı transfer maliyetleri (yani, iskonto oranları) α = {0.2,0.4,0.6,0.8,1.0} kullanıldı. Toplama ve dağıtım birim maliyetleri ise sabit olarak $\chi = \delta$ =1 kabul edilmiştir. Her bir iskonto oranı için, bir düğüm noktasında merkez üs kurmanın sabit maliyeti ise $f_k \in${100, 150, 200} olarak belirlendi. Referans maliyet olarak kesin sonuçlar ise Abdinnour-Helm'in çalışmasından türetilmiştir (Abdinnour-Helm, 1998).

HN ve HN* sonuçlarını başka bir yöntemle de karşılaştırmak için, orijinali Ernst tarafından uygulanan tavlama benzetimi yaklaşımını p-merkez üs problemine uyarlandı (Ernst ve Krishnamoorthy, 1996). TB algoritmasının ilk kontrol parametresi tavlama sıcaklığının (T^0) başlangıç değeridir ve aşağıdaki gibi hesaplanabilir (Aarts ve Korst, 1989):

$$\gamma = \frac{m_1 + m_2 \times e^{\frac{-\Delta f}{T^0}}}{m_1 + m_2} \tag{8.38}$$

Burada, γ (kabul oranı) 0.95 olarak belirlenmiştir. Bu formülasyonda, m_1 maliyette azalmaya yol açan hareketlerin sayısını, m_2 ise bir önceki adıma göre maliyette artışa neden olan hareketlerin sayısını ve Δf de m_2 hareketleri için maliyetteki ortalama artışı gösterir. Başlangıç sıcaklığı Markov zinciri için geçici büyük bir sayıya (deneyimizde 100.000) eşitlenir. Bir tam Markov zinciri tamamlandıktan sonra, başlangıç sıcaklığı (T^0) Denklem 8.44 kullanılarak hesaplanır. Markov zincirinin uzunluğu $2np$ (n problemin büyüklüğü, p merkez üs sayısı) olarak belirlenir. Tavlama benzetimini n = 50 ve n = 5 için çalıştırdığımızda T^0 için 224.000 değeri elde edilmiştir.

Aynı merkez üsse tahsis edilen düğüm noktaları grubu *küme* olarak tanımlanmıştır. TB sezgisel yaklaşımında komşuluk çözümlerini elde etmek için iki ana işlem vardır: 1) rasgele seçilmiş bir düğüm noktasının farklı bir kümeye tahsis edilmesi ve 2) rasgele seçilmiş bir kümeden rasgele seçilen bir düğüm noktasının merkez üs olarak

belirlenmesi (Ernst ve Krishnamoorthy, 1996). Bu işlemlerden bir tanesi daha önce belirlenmiş bir olasılıkla uygulanır. TB sezgisel yaklaşımının uygulanması esnasında yerel minimumdan kurtulmak için bir yeniden ısıtma mekanizması da kullanılır. Deneylerimizde soğutma oranını 0.97 olarak kullandık.

Deneyler neticesinde elde edilen HN, HN* ve TB sonuçları Tablo 8.2'de gösterilmiştir. Hopfield ağı matris işlemlerinden dolayı çok fazla hesaplama zamanı ve hafızaya gereksinim duyduğundan tavlama benzetimi ile hesaplama süresi karşılaştırması yapılmamıştır.

Tablo 8.2 : Sonuçların karşılaştırılması (n = 10)

α	f_k	Bağlantı noktaları	Eniyi çözüm	HN	HN*	TB
0.2	100	4,6,7	791.93	**18.8%**	0.0%	0.0%
	150	7,9	915.99	0.0%	0.0%	0.0%
	200	7,9	1015.99	0.0%	0.0%	0.0%
0.4	100	4,6,7	867.91	**15.9%**	0.0%	0.0%
	150	7,9	974.30	0.0%	0.0%	0.0%
	200	7,9	1074.30	0.0%	0.0%	0.0%
0.6	100	7,9	932.62	0.0%	0.0%	0.0%
	150	7,9	1032.62	0.0%	0.0%	0.0%
	200	4	1131.05	0.0%	0.0%	0.0%
0.8	100	7,9	990.94	0.0%	0.0%	0.0%
	150	4	1081.05	0.0%	0.0%	0.0%
	200	4	1131.05	0.0%	0.0%	0.0%
1.0	100	4	1031.05	0.0%	0.0%	0.0%
	150	4	1081.05	0.0%	0.0%	0.0%
	200	4	1131.05	0.0%	0.0%	0.0%

HN ilk yerel minimuma takıldığından bazı durumlar için kötü sonuçlar vermiştir. Ancak HN* çözüm kalitesini oldukça artırmış ve veri setindeki bütün durumlar için en iyi çözümü bulmuştur. Yine TB sezgisel yaklaşımı da yeniden ısıtma yöntemi sayesinde yerel minimumlara takılmadan en iyi çözümlere ulaşmıştır.

9. YAPAY SİNİR AĞLARI İLE KONTROL EDİLEN GEZGİN ETMEN TEMELLİ TEDARİK ZİNCİRİ YÖNETİM SİSTEMİNİN TASARIMI

Bu bölümde, entegre tedarik zinciri yönetimi sistemleri için **gezgin etmen** (mobile agents) mimarisini temel alan, talep tahmini ile tedarikçi seçimi süreçleri daha önce Bölüm 2 ve 3'de anlatılan yapay sinir ağı modelleri kullanılarak Java platformunda gerçeklenen karar destek sistemi açıklanacaktır.

9.1 Giriş

Son yıllarda, ekonominin küreselleşmesi ile birlikte, tedarik zinciri faaliyetlerinin yerel olmaktan çıkıp yeryüzü boyutunda yönetilmeye başlandığı görülmektedir. Ürünler artık aynı coğrafyada üretilip satılmamaktadır. Bir ürünün farklı parçaları bile sıklıkla dünyanın çok farklı yerlerinden gelmektedir. Bu durum ise daha karmaşık tedarik zincirlerinin oluşmasına ve dolayısıyla tedarik zinciri yönetimindeki gereksinimlerin farklılaşmasına yol açmıştır. Bilişim ve yazılım mühendisliğindeki hızlı ilerleme, tedarik zincirinin entegrasyonu ve koordinasyonunun sağlanmasında yeni fırsatları da ortaya çıkarmıştır.

Yirmi birinci yüzyılın rekabetçi piyasası, organizasyonlar arasında ürün, süreç ve bilgi akışını birleştiren, gürbüz ve geniş işletme yönetimi çözüm/girişimlerine gereksinim duymaktadır. Bu da içsel ve dışsal işletme süreçlerini tanımlama, iletme ve sürekli iyileştirmede elektronik veri değişimi/Internet'in etkin olarak kullanılması ile mümkündür. Bilgilerin elektronik ortamda değişimi hataların azalmasına yol açarak süreçlerin verimliliğini artırır. Tedarik zincirinde bir şirket diğer şirketlerin bilgilerini kullanabildiklerinde, belirsizliğin negatif etkileri (örneğin, hatalı talep tahminleri, yüksek envanter düzeyleri, vs.) azaltılabilir.

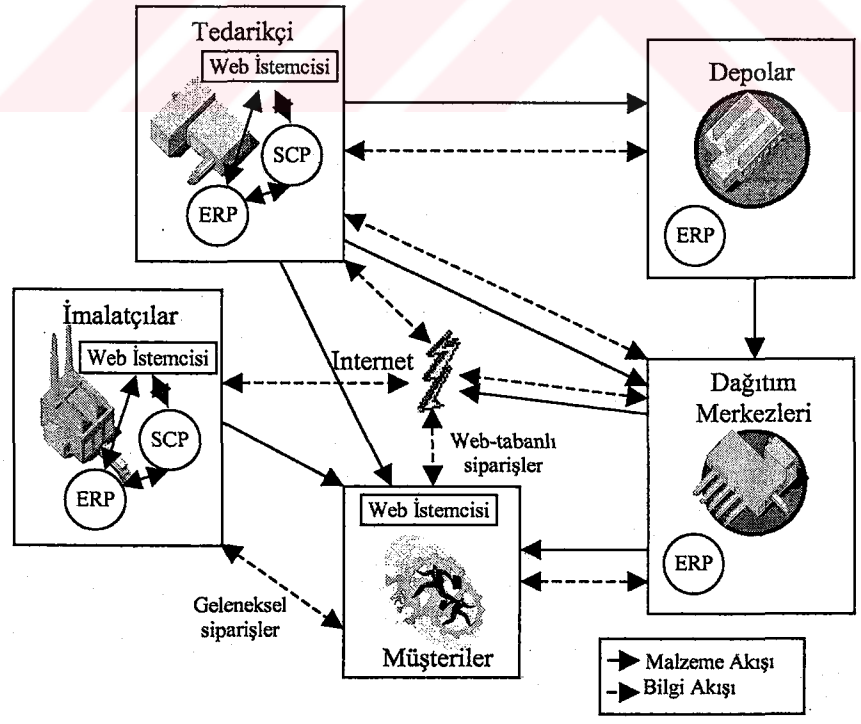
Tedarik zinciri işlemlerini iyileştirmede sadece etkin bilgi transferi değil bilginin zamanında hazır bulunması da anahtar role sahiptir. Aslında, parça talebinin görünürlüğü, konumu ve lojistik ağındaki bütün parçaların durumundan emin olmak için bilişim sistemlerinin kullanımı imalatın önemli bir niteliği olarak 90'ların başında tanımlanmıştır (Kehoe ve Boughton, 2001a; Kehoe ve Boughton, 2001b). Internet,

bir ana kaynakta yer alan bilgilerin herkesin kullanımına sunulmasına olanak sağlayan ilk kanaldır. Internet ve ilgili teknolojilerin mevcudiyeti, işlevsel engellerin ortadan kaldırılması ve bilgi akışının iyileştirilmesinde daha önemli gelişmelerin gerçekleştirilmesine fırsat sağlamaktadır (Boyson ve diğ., 2003).

Modern bilgisayar ağları bilgilerin ilgili bütün varlıklara hızlı bir şekilde dağıtılmasını sağlayacak kabiliyettedir. Bütün dünyaya yayılmış farklı işletmelerin bilgisayarları, parçaların mevcudiyeti, siparişlerin verilmesi ve kesinleştirilmesi, dağıtım çizelgelerinin müzakere edilmesi gibi konuları belirlemek için birbirleriyle iletişim kurabilmektedirler (Sarkis ve Sundarraj, 2002).

9.2 Tedarik Zincirlerinde Koordinasyon Mekanizmaları

Elektronik ortamda TZY sistemlerinde bütünleşme düzeylerinin işlevselliği ve karmaşıklığı ileri düzeyde bütünleşikten kısmen bütünleşik olana kadar değişebilir (Pant ve diğ., 2003). Yüksek düzeyde bütünleşik TZY sistemleri genellikle karmaşık içsel ve dışsal işlemleri içerirler. TZY'de çok farklı miktar ve çeşitte parça ve bileşenlerin satın alındığı bir çok tedarikçi, küresel olarak yayılmış olabilir. Rekabetçi konumda kalabilmek için etkin ve verimli olarak karmaşık işlemleri birbirine bağlayan TZY sistemi kurulabilir.



Şekil 9.1 : Entegre tedarik zinciri yönetimi sistemlerinde siparişlerin gerçekleştirilmesi

Bu sistemlerde, esasen üç ana oyuncu vardır: imalatçı, tedarikçiler ve dağıtıcılar (Şekil 9.1). Ayrıca lojistik ve depolama kolaylıkları da yer alır. Sistemin, zincirdeki envanteri azaltıp, gecikmeleri önleyip, daha iyi müşteri hizmeti sağlayarak rekabetçi konumunu sürdürebilmesi için iki ana görevi gerçekleştirmesi gerekir: 1) tedarik zincirinin değişik parçaları arasında ortaklaşa planlama ve 2) müşteri sorgulamalarına cevap verme (Simchi-Levi ve diğ., 2000).

Yazılım etmenleri, Internet üzerinden tedarik ve ürünlerin satışlarının da yer aldığı bir çok işin otomatikleştirilmesine yardımcı olur. Gereksinim duyulan ve tedarik edilen ürünlerle ilgili etmen faaliyetleri, bir etmenin farklı ürünlerin alımının karşılaştırılması problemini azaltacak biçimde tanımlanırlar. Bu tip etmenler, bir kullanıcının aşırı bilgi ile yüklenmesini veya araştırma maliyetlerini azaltacak biçimde tasarlanırlar. Bu tip sistemlerle klasik sistemler arasındaki fark, nüfuz alanına özel, sürekli çalışan ve otonom olmalarıdır (Wu, 2001).

Çok-etmenli sistemlerin üretim ve tedarik zinciri yönetimindeki uygulamaları yeni değildir. Akıllı üretimlerde etmenler, imal edilecek ürünlerin yapılanış tasarımı (Darr ve Birmingham, 1996), çevik üretimde çok-etmenli sistemlerin koordinasyonu (Barbuceanu ve Fox, 1996), üretim işlemlerinin çizelgelenmesi ve kontrolü (Sadeh, 1994; Choi ve Park, 1997), araç rotalama ve işletmelerin modellenmesi (Kwok ve Norrie, 1993), ve üretim sıralamasının belirlenmesi (Wooldridge ve diğ., 1996) gibi farklı işlevsel alanlarda kullanılmıştır. Sikora ve Show bilişim sistemlerinin koordinasyon ve entegrasyonu için bir çok-etmenli çatı oluşturmuşlardır (Sikora ve Shaw, 1998). Gilman sanal işletme kuralları, akıllı etmenler, STEP ve iş akışında tasarım ve imalatı bütünleştirmiştir (Gilman ve diğ., 1997). Shen ve Norrie akıllı imalat için etmen-temelli sistemlerin bir taramasını yapmıştır.

Yung ve Yang, TZY problemini çözmek için çok-etmenli teknoloji ile kısıt ağını bütünleştirmeyi önermiştir (Yung ve Yang, 1999). Yan ağ akış modelinde maliyet paylaşımı temelli dağıtılmış elektrik güç iletimi için bir çok-etmen temelli görüşme destek sistemi geliştirmiştir (Yan ve diğ., 2000). Ancak, gezgin etmenler kullanan koordinasyon mekanizması ne etmen temelli imalatta ne de etmen temelli tedarik zincirlerinde ele alınmamıştır.

9.3 Gezgin Etmenler

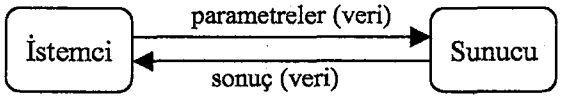
Etmen; bir fiziksel çevre için yerleşmiş olan, kendi tasarım amaçlarına uygun olarak otonom fiiller yapabilme yeteneği olan bilgisayar sistemleridir / yazılımlarıdır. Bir

etmenin repertuarı; o etmenin amacına ulaşmak için, çevreden algıladığı ve kendi bilgi tabanındaki verilere göre yerine getirebileceği fiiller kümesini ifade etmektedir. Bu repertuar etmenin çevreyi etkileyebilme kapasitesini göstermektedir. Burada karşılaşılan temel problem ise etmenin daha önce karşılaşmadığı ve/veya etmen geliştirici tarafından tanımlanmamış olan bir problem karşısında, tasarım amacına uygun olarak bu fiillerden hangisini seçeceğine karar vermesidir. Bu karar verme süreci çeşitli çevresel değişkenler tarafından etkilenmektedir.

Gezgin etmen ise; bir ağ üzerinde (Internet, yerel alan ağı, geniş alan ağı, vs), kullanıcı adına belirli işlevleri yerine getirirken, otonom olarak kendisini bir makineden başka bir makineye taşıyarak (göç ettirerek) görevine/araştırmalarına yeni yerleşkesinde devam edebilen etmendir. Gezgin etmenin hangi görevleri yerine getireceği, hangi makineler üzerinde dolaşacağı, nasıl bir karar verme stratejisi uygulayacağı, ilk üretim zamanında gezgin etmene kullanıcının talepleri doğrultusunda etmen sistemi tarafından yüklenir. Bu veriler, etmen farklı uçlar arasında dolaşırken edindiği bilgilere göre yeniden düzenlenebilir. Bu aşamadan sonra etmenin hareketi yüklenen amacına göre otonom olarak devam eder.

Taşınabilirliğin evrimsel gelişme sürecinin ilk adımında dosyaların bir makineden diğer bir makineye değişik protokollerle (örneğin, FTP) transferi amaçlanmıştır. Bu protokollerde karşı makineye bağlanarak, gerekli güvenlik kontrollerinden sonra istenilen dosyanın yerel makineye indirilmesi sağlanmaktadır.

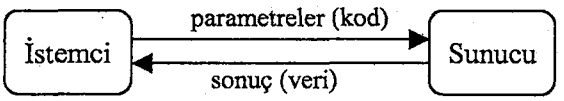
İkinci adım Uzaktan Yordam Çağırmadır (Remote Procedure Call-RPC). Uzaktan yordam çağırma işlemiyle, basit dosya transferinden öte, çalışan bir programın çalışma akışında kontrolün sunucu makineye geçmesi sağlanmıştır. Bu modelde istemci, sunucu üzerindeki bir metodu çağırırken, o metodun ihtiyaç duyduğu parametreleri de karşı tarafa göndermektedir (Şekil 9.2). Sunucu bu parametreleri aldığında artık çalışan programın kontrolü kendi elindedir. Diğer taraf sunucudan gelecek olan sonuç değerini beklemektedir.



Şekil 9.2 : Uzaktan Yordam Çağırma

Her ne kadar uzaktan yordam çağırma modelinde uzak bir makinedeki metot çağrısının yerel makinedeki metot çağrısı gibi yapmak amacı gerçeklenmesi basit bir

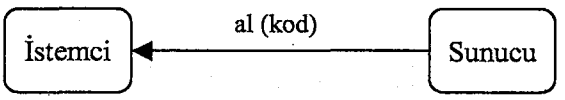
amaç olsa da, bu model, bilgisayar biliminde yeni yaklaşımlar için güçlü bir etkiye sahip olmuştur. Bunun üzerine bir sonraki adım olarak, basit bir veri (parametre) transferi yerine çalışacak programın karşı tarafa gönderilmesi fikri ortaya çıkmıştır. Şekil 9.3'de görüldüğü gibi, sunucu makinede çalıştırılması istenen program parçası, parametre olarak gönderilerek bu program parçasının çalıştırılması ve sonucunun istemciye gönderilmesi beklenir. Bu model Uzaktan Değerlendirme (Remote Evaluation - REV) olarak adlandırılmaktadır (Stamos ve Gifford, 1990).



Şekil 9.3 : Uzaktan Değerlendirme

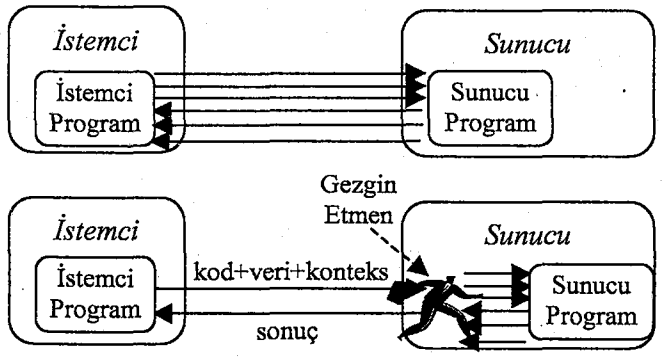
Uzaktan değerlendirme paradigmasında programı çalıştıran bileşen, istemci, görevin yapabilme bilgisine (know-how) sahiptir, fakat bunun yerine getirilmesi için ihtiyaç duyulan kaynak uzaktaki bir makinededir (sunucu). Bu sebeple istemci, kendisindeki yapabilme bilgisini karşı tarafa sunucuya göndererek, bu program parçasının karşı tarafta çalıştırılıp sonucunun kendisine gönderilmesini bekler. Sunucu gönderilen kodu çalıştırarak, kendi yerel verilerine daha hızlı erişim ile istenen sonuca ulaşıp bunu istemciye gönderir.

Talepte Kod Alma (Code on Demand - COD) paradigmasında, istemci kaynaklarına yerel erişim hakkına sahiptir, fakat görevi yerine getirmesi için gereken yapabilme bilgisine sahip değildir. Bu sebeple istemci bu bilgiye sahip olan sunucudan, ihtiyaç duyduğu yapabilme bilgisine erişir (Şekil 9.4'de görüldüğü gibi) ve kendi görevini sonlandırır.



Şekil 9.4 : Talepte Kod Alma

Taşınabilir kodun bir örneği olarak gezgin etmen kavramı kullanılmıştır. Bir etmen, görevi ile ilgili karar alınmasında ve görevin idamesinde kullanıcı ile her zaman doğrudan bir etkileşim göstermeden farklı otonom derecelerinde çalışabilen aktif bir yazılım varlığıdır. Bir gezgin etmen ise ağ üzerinde farklı makinelerde dolaşabilen ve konak makinelerinin özkaynaklarını ve hizmetlerini Şekil 9.5'de gösterildiği gibi kullanabilen etmenlerdir.



Şekil 9.5 : Gezgin Etmen

Gezgin etmenler taşınabilir, esnek, otonom, dinamik ve verimli bir yapıya sahiptirler. Bir görevin içine yerleştirildikleri zaman, uzaktaki bir makineye kendi ana makinesinden gönderilirler. Bu makinede görevini tamamladıktan sonra ulaştığı sonucu asıl makineye bir mesajla gönderirler veya doğrudan kendisi gelerek sonucu iletirler. Gezgin etmen kavramı aynı zamanda dağıtılmış ortamlarda ve İnternet'te paralel çalışmaya da olanak sağlamaktadır. Görevler, ufak ve çok sayıda gezgin etmen içine gömülerek, daha sonra çalışmaları için ağ üzerindeki farklı makinelere gönderilirler. Her etmen kendine yüklenen görevi yerine getirmek için bağımsız olarak çalışır ve ulaştıkları sonuç, bir ana sunucu üzerinde veya yönetici görev üzerinde toplanır. Bu sayede paralelleştirilebilen görevlerde hızlı sonuç elde edilebilmesine olanak sağlanmaktadır.

Taşınabilir sistemler aslında homojen yapıda sistemler için ortaya çıkmışlardır. Bu sistemlerde verilerin ikili (binary) gösterimleri ve sistem özellikleri aynı olmasına rağmen gene de bazı problemlerle karşılaşılmaktadır. Günümüzdeki heterojen ortamlara doğru olan genişlemeyi göz önüne aldığımızda, taşınabilirlik daha da zor bir problem olarak karşımıza çıkmaktadır. Buna cevap olarak bulunan bir yol ise Java programlama dilidir. Java nesneye yönelik programlama stilini, bir ara kod dönüşümü ile bayt kodlara (bytecode) çevirmekte ve JVM'e sahip her platformda aynı şekilde yorumlanmaktadır.

9.3.1 Taşınabilirlik Türleri

Gezgin etmenler, dağıtılmış sistemlerin tasarlanmasını ve idamesini kolaylaştırılmasına olanak sağlayan bir teknoloji olarak ortaya çıkmıştır. Gezgin etmenlerin bir makineden diğer bir makineye taşınmasının üç temel türü vardır (Cugola ve diğ., 1997; Hohlfeld ve Yee, 2003).

- **Zayıf Taşıma:** Farklı bir makineden gelen etmenin kaynak koduna dinamik bir bağlantı kurulması ile yapılan taşımadır. Kaynak kodun tamamı yerine sadece bulunabileceği adresi alıcı tarafa gönderilir. Karşı taraftaki makine bu etmeni aktif hale getirmek istediğinde, kendisine gelen adres bilgisinden bu etmenin kaynak koduna ulaşır ve kendi makinesinin belleğine yükler. Aglet (Baumann ve diğ., 1998) ve Mole (Lange ve Oshima, 1998) gibi Java tabanlı etmen sistemlerini hemen hemen hepsi zayıf taşımalı sistemlerdir.

- **Güçlü Taşıma:** Bir etmenin kaynak kodunun ve çalışma durum değişkenlerinin ve veri uzayının karşı makineye taşınması ve bu makineden etmenin çalışmaya kaldığı noktadan devam edebilmesine olanak sağlayan taşıma şeklidir. "Telescript" gibi sistemler, bu tip taşıma uygun olarak hazırlanmış olan dil yorumlayıcısı sayesinde bir görevciğin çalışma durum değişkenlerini yakalayabilmekte ve karşı tarafa gönderebilmektedir (White, 1994)

- **Tam Taşıma:** Çoğu yaklaşımda bu taşıma modeli olmamasına rağmen bazı araştırmacılar bunu da bir taşıma modeli olarak gördükleri için incelemekte fayda vardır. Tam Taşıma, çalışan bir programın tüm görevciklerinin, adlandırma alanlarını yığınlarının, program sayaçlarının ve diğer kaynaklarının başka bir makineye taşınmasıdır. Bu yapı göç ettirme mekanizmasının daha da saydamlaştırıldığı güçlü taşımanın genel bir hali olarak görülebilir. Tam taşıma, eğer göç ettirilen etmen yük dengeleme gibi görevler için tamamıyla saydam bir şekilde yapılması gerekiyorsa kullanılması gereklidir. LOCUS bu modeli kullanan bir dağıtılmış işletim sistemidir (Walker ve diğ., 1983).

9.3.2 Gezgin Etmen Kullanmanın Faydaları

Gezgin etmen kullanımı için en az yedi nedenin olduğu söylenebilir.

1. *Ağ yükünü azaltırlar.* Dağıtılmış sistemler genellikle verilen bir görevi tamamlamak için çoklu etkileşim içeren haberleşme protokollerinden faydalanırlar. Bunun sonucu olarak da ağ üzerinde ciddi bir veri trafiğine sebep olmaktadırlar. Gezgin etmenler, kullanıcıların bu karşılıklı konuşmalarını (protokollerini) birer paket haline getirerek etkileşimin olacağı yerel makineye gönderilmesine olanak sağlamaktadır. Gezgin etmenler aynı zamanda ağ üzerinden ham veri akışını da ciddi ölçüde azaltabilmektedir. Büyük ölçeklerdeki bir veri bloğunun uzaktaki bir makinede saklanması durumunda, bu verinin işlenmesi için veri bloğunun kendi makinemize

alınması yerine, gönderilecek bir gezgin etmen ile gerekli işlemin uzak makinede, yerel olarak halledilebilmesi de mümkündür.

2. *Ağ gecikmelerini engeller.* Üretim süreçlerindeki robotlar gibi, kritik gerçek zamanlı sistemler çalıştıkları ortamdaki değişikliklerden hızlı şekilde haberdar olmak isterler. Bu gibi kontrol işlemlerinin merkezi bir şekilde yapılması ciddi gecikmelere ve merkezde yoğunluğa yol açmaktadır. Gezgin etmen kullanımı sayesinde merkezi kontrolden kurtularak, bu etmenler alıcıların olduğu yere yerleştirilerek, yönetici görevciğin direktiflerine göre hareket etmekte ve bu sayede gecikmeleri azaltmaktadır.

3. *Protokolleri içerir.* Bir veri, dağıtılmış sistem üzerindeki uçlar arasında el değiştirirken, her uç bu giden veriyi kodlamak ve gelen veriyi yorumlamak için gerekli protokollere (yorumlayacak program parçacıklarına) sahiptirler. Bununla birlikte protokoller, etkin kulanım ve/veya güvenlik gibi yeni gereksinimleri karşılamak için gelişirken, uçlardaki bu program parçacıklarının da güncellenerek yeni ortama uyum sağlaması ve hantal yapıdan kurtulunması gerekmektedir ve bunu sağlamak zor ve masraflıdır. Bunun sonucu olarak protokoller sık sık bir kalıtımsal problem olarak karşımıza çıkmaktadır. Gezgin etmenlerin kullanımı sayesinde bu protokoller etmenin içine gömülerek sisteme dinamik ve güncel bir yapı kazandırılabilmektedir.

4. *Asenkron ve otonom olarak çalışabilmektedirler.* Günümüzde cep telefonu ve PDA (Personal Digital Assistant) gibi gezgin aygıtlar pahalı ve kırılgan ağ bağlantıları üzerinde yoğunlukla çalışmaktadırlar. Sabit ağ ile gezgin aygıt arasındaki hattın daima açık olmasını gerektiren görevlerin bu gibi sistemlerde kullanılması, teknik olarak uygun olmadığı gibi ekonomik de değildir. Bu problemi çözmek için, gerekli görev bir gezgin etmen içine gömülerek ağ üzerinden ilgili aygıta gönderilebilir. Bu dağıtım işleminden sonra etmen kendini üreten etmenlerden bağımsız, asenkron ve otonom olarak çalışabilir ve böylece bağlantının/hattın daima açık kalmasına gerek kalmaz. Gezgin aygıt daha sonra uygun olduğu bir zaman sisteme bağlanarak işleminin sonucuna ulaşabilir/gönderebilir.

5. *Ortama dinamik olarak uyum sağlayabilir.* Gezgin etmenler, çalıştıkları ortamı algılayarak, çevrede olan değişikliklere otonom olarak cevap verebilmektedirler. Çok sayıda ve farklı tiplerdeki gezgin etmelerin ağ üzerinde uygun şekilde yayılarak belirli bir problemin çözümü sağlanabilir.

6. *Doğal olarak heterojendir.* Ağ işlemleri/hesaplamaları gerek yazılımsal, gerekse donanımsal olarak bakıldığında genel olarak heterojendir. Gezgin etmenlerde genelde bilgisayar ve ulaştırma katmanından bağımsız olup (sadece çalışma ortamına bağımlı) bağlantısız sistem bütünleştirmesi için en iyi koşulları sağlamaktadır.

7. *Gürbüz ve hata hoşgörülüdür.* Gezgin etmenlerin beklenmeyen durumlara karşı dinamik olarak karşılık verme yeteneklerinden ötürü gürbüz ve hata hoşgörülü dağıtılmış sistem tasarımına olanak sağlamaktadır. Bir uçtaki makinenin kapatılması gerekince bu uçta çalışan etmenler kendilerini bu kapatma sürecinde ağ üzerindeki başka bir makineye naklederek çalışmalarına orada devam edebilirler.

9.3.3 Gezgin Etmen Modelleri

Gezgin etmen modelleri beş ana sınıf altında toplanabilir: Seyahat Programlı Etmen, Mekik Etmen, Seri Etmen, Sanal Paralel Etmen ve Seri Paralel Etmen (Wang ve Tan, 2002).

- **Seyahat Programlı Etmen:** İlk geliştirilen gezgin etmen modeli olup, üretici etmen tarafından oluşturularak, dolaşacağı makinelerin adresleri bu sırada etmene yüklenir[1]. Etmen hedef makineler arasında birer birer göç ederek kendine verilen görevi yerine getirir. Etmen, bir makineden diğerine göç ederken, daha önce eriştiği makinelerden topladığı verileri de yanında götürmektedir. Hedefindeki bütün makineleri dolaşan etmen, topladığı bütün veriyle beraber üretici makineye geri gelir.

- **Mekik Etmen:** Bir üretici etmen ve bir işçi etmen vardır. İşçi etmen üretici tarafından yüklenen hedef listesindeki ilk makineye gönderilir. Bu makinede etmenin işi bitince erişmiş olduğu bilgiyi tekrar üretici makineye gönderir ve kendisini bir sonraki hedef makineye göç ettirir. Bu işlem bütün hedef makineler dolaşılıncaya kadar devam eder.

- **Seri Etmen:** Mekik etmen modelinde olduğu gibi bu modelde de, bir üretici etmen vardır ve işçi etmen bunun tarafından ilk hedef makineye gönderilir. İşçi etmen aradığı bilgiye ulaştığı zaman, bunu bir mesaj ile üretici etmene ileterek,

---

[1] İlk geliştirilen modellerinde bu şekilde olmasına rağmen, daha sonraki modellerde bu makinelerin adresleri gezgin etmen tarafından yeni bilgilere ulaşıldıkça güncelleştirilmiştir.

kendisini yok eder. Üretici etmen bu mesajı aldıktan sonra bunu değerlendirerek, bir sonraki makineye yeni oluşturmuş olduğu işçi etmeni gönderir. Bu işlem, bütün hedef makineler dolaşılıncaya veya başka ilgi duyulan makine kalmayana kadar devam eder.

- **Sanal Paralel Etmen:** Üretici etmen aynı zamanda istemci makinesinde de oluşturularak, orada bir görev ipliği olarak çalıştırılır. Bu etmen, amacına ulaşınca sonlanır ama seri modelde olduğu gibi kendisini yok etmez. Bu modelde herhangi bir işçi etmen kullanılmaz.

- **Seri Paralel Etmen:** Bu modelde ise paralel veri erişiminden faydalanılmaktadır. Bu modelde üretici etmen çok sayıda işçi etmen oluşturarak bunları bütün hedef makinelere ayrı ayrı gönderir. İşçi etmenler uzak makinede yerel verilere hızlı erişim sağlayarak kendi görevlerini sonlandırır ve üretici etmene sonucu birer mesaj olarak gönderirler.

9.3.4 Java'nın Gezginlik Desteği

Java programlama dili, dinamik sınıf yükleme özelliği ile (örneğin apletler) bir program kodunun makineler arasında taşınmasına olanak sağlaması sayesinde ağ tabanlı programlamaya güçlü bir destek vermektedir. Buna ek olarak zayıf taşınabilirliğin bir formu olarak nesneleri serileştirebilmekte ve başka JVM'lere soketler ve/veya Uzaktan Metot Çağırma (RMI) ile gönderilebilmektedir. Serileştirme mekanizması, durum değişkenlerinin değerlerinin saklanmasına ve gönderilmesine olanak sağlasa da, bir program parçasının çalışmasının başka bir makinede kaldığı yerden devam etmesine olanak sağlamamaktadır. Serileştirilen nesne karşı tarafa ulaşınca tekrar nesne haline getirilip, tanımlanan bir metodu çağrılarak aktifleştirilir. Metodun adı ve parametre gönderilip gönderilmemesi farklı gezgin etmen sistemlerinde değişik şekillerde ifade edilmişlerdir. Bazıları *run* metodunu bazıları ise *go* metodunu aktifleştirici metot olarak kullanmaktadır.

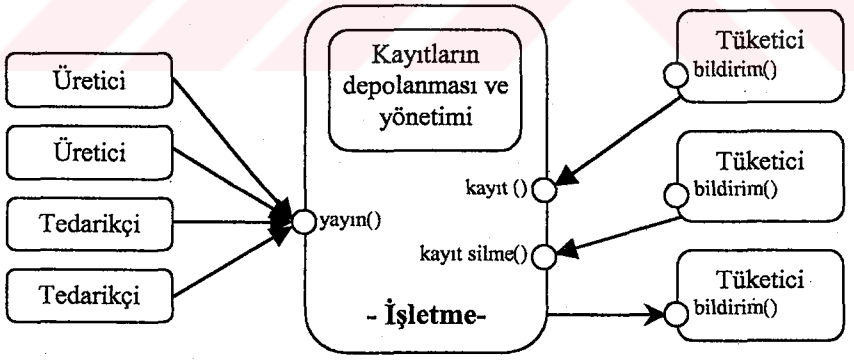
Java tabanlı gezgin etmen sitemleri bir etmeni bir veya daha fazla görev ipliği olarak algılamaktadırlar. SUN'ın resmi JVM'i güçlü taşınabilirliği desteklememektedir. Bunu destekleyebilmesi için JVM kodunun, taşınacak kodun bellekte tutulan program sayacı ve yığındaki verilerinin de transfer edilebilmesine olanak sağlaması gerekmektedir. Bunun için de bu veriler özellikle saklanmalı ve bir etmen formunda karşı tarafa serileştirilerek gönderilmelidir. Bu alanda yapılan çalışmalarda

görülmüştür ki, program sayacına ulaşmak nispeten kolay bir işlem olmasına rağmen[2] Java yığınlarının kontrolünde ciddi sıkıntılar ortaya çıkmaktadır.

Java tabanlı gezgin etmenler, kalıtımsal olarak Java programlarının temel özelliklerinden olan bilgisayardan bağımsız çalışabilme özelliklerini alabilmekte ve bu sayede heterojen veri tabanları ve veri kaynaklarına platformdan bağımsız erişime olanak sağlamaktadır.

9.4 Önerilen Gezgin Etmen Sisteminin Tasarlanması

Pek çok müşteri ve tedarikçilerin olduğu büyük ölçekli ve dinamik bir ortamda tedarik zinciri süreçlerini gerçeklemek için gezgin etmen temelli ve yapay sinir ağlarını kullanan bir çerçeve model tasarlanmış ve gerçeklenmiştir (Ermiş ve diğ., 2004). Bu modelde tedarikçiler, sistemin dinamik yapısına uygun olarak herhangi bir zamanda sisteme bağlanabilir, kayıt olabilir veya kayıt sildirebilirler. Bu yüzden, önerilen yapı bir çok gevşek bağlı varlıkların etkileşimini destekleyecek yayın/kayıt paradigmasına dayanır (Şekil 9.6). Bu modelde, müşteriler tedarikçilerin sayısını veya adreslerini bilmezler. Bir müşteri isteğini İşletme'ye gezgin etmenler vasıtasıyla iletir ve İşletme seçilmiş tedarikçilere gezgin etmenleri dağıtır. Bu çatı sadece müşteri ve tedarikçi faaliyetlerini desteklemez, aynı zamanda, eşzamanlı olarak tedarikçilerde gezgin etmenleri çalıştırarak paralel hesaplamayı da kolaylaştırır.



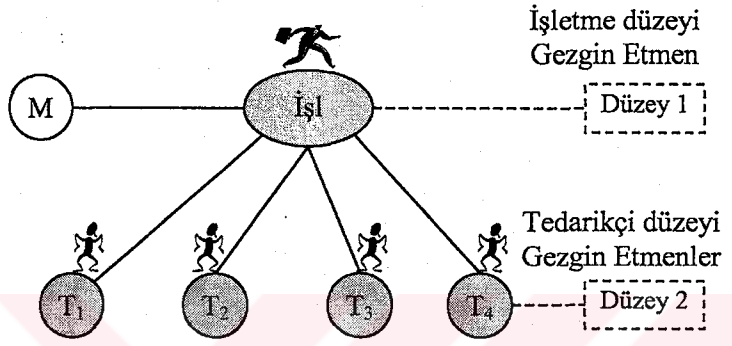
Şekil 9.6 : Yayın/kayıt modeli

Sistem gezgin etmenleri, müşterilerle üreticiler ve tedarikçiler arasında aracı olarak kullanır. Gezgin etmenler, Şekil 9.7'de görüldüğü gibi, uygulama ve sorumluluklar

---

[2]JVM içinden erişilebilmekte, transfer edilebilmekte ve yeniden oluşturulabilmektedir.

açısından iki farklı düzeye sahiptirler: İşletme düzeyi Gezgin Etmen (İGE) ve Tedarikçi düzeyi Gezgin Etmen (TGE). Bir İGE Müşteri Etmeni tarafından yaratılır ve İşletmeye gönderilir. Bu İGE, TGE'leri yaratarak tedarikçilere ait veritabanlarını araştırmak, seçim yapmak ve gerekiyorsa müzakere yapmak için onlara gönderilir. Sistem, tek tek her tedarikçiyi dolaşan sadece bir gezgin etmen içermez. Onun yerine, eşzamanlı olarak her tedarikçiye gezgin etmenin bir replikasyonu gönderilerek paralel işleme gerçekleştirilir. Bu paralel işleme modeli özellikle tedarikçi sayısının çok olduğu ve İşletmenin karar vermesi için zamanının az olduğu durumlarda önemlidir.



Şekil 9.7 : İki-Düzeyli Gezgin Etmen modeli

Bir çok tedarik zinciri ve elektronik ticaret sisteminde, bir üretici başlangıçta belirlenen sabit sayıda tedarikçiye sahiptir. Yeni bir tedarikçi ekleneceği zaman, ona ait bilgiler ve gerekli parametreler manüel olarak kayıt edilmek zorundadır. Büyük ölçekli ve dinamik bir ortamda, herhangi bir anda bir çok tedarikçi ve müşteri söz konusu olabilir. Bu nedenle sistem kendisini bu dinamik ortama adapte edebilmelidir. Bu gereksinimi karşılamak için, önerilen çatıda, kayıt ve işlemlerin dağıtılmasında verimliliği ve satın alma sürecinin etkinliğini maliyet, kalite, performans ve zamanlama açısından artırmak için yayın/kayıt paradigması kullanılmıştır.

Ürün satın almak isteyen kullanıcı bir etmen oluşturur, ona bir takım stratejik direktifler verir ve ilgili işletmenin merkezi etmen sistemine gönderir. Sistemin gezgin etmenleri proaktif olarak tedarikçileri gezer ve ait olduğu kişinin (işletmenin) adına müzakere eder. Her bir etmenin hedefi, istenen fiyat, kabul edilebilir en düşük fiyat ve teslim tarihi gibi kullanıcıya özel kısıtlara göre kabul edilebilir bir anlaşmayı gerçekleştirmektir.

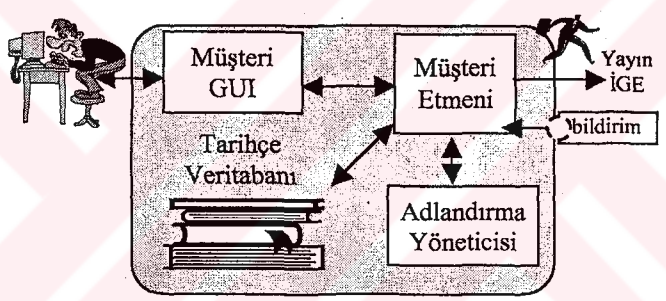
Geliştirilen modelde tedarik zinciri, statik tedarik zinciri etmenleri (müşteri etmeni, imalatçı etmeni ve tedarikçi etmeni gibi) ve gezgin tedarik zinciri etmenlerinin

(işletme düzeyi gezgin etmen ve tedarikçi düzeyi gezgin etmen) örneklerini gösteren yazılım bileşenlerinden oluşur. Etmen tanımlamaları, farklı tedarik zinciri varlıklarının statik ve dinamik karakteristiklerini belirtme yeteneğini sağlar. Her bir etmen tedarik zincirindeki tasarlanan rolüne göre özelleştirilir.

Sistemdeki tedarik süreci üç ana oyuncudan oluşur: müşteriler, işletmeden ürün veya hizmet satın almak isterler. Tedarikçiler, mamul, yarı mamul veya hammadde temin ederler ve Kontrol/Eniyileme Sistemi, tedarikçilerin seçimi, satın alınacak miktarın belirlenmesi, imalat sonrası müşterilere teslimi gibi hizmetleri kontrol ederler.

9.4.1 Müşteri Alt-sistemi

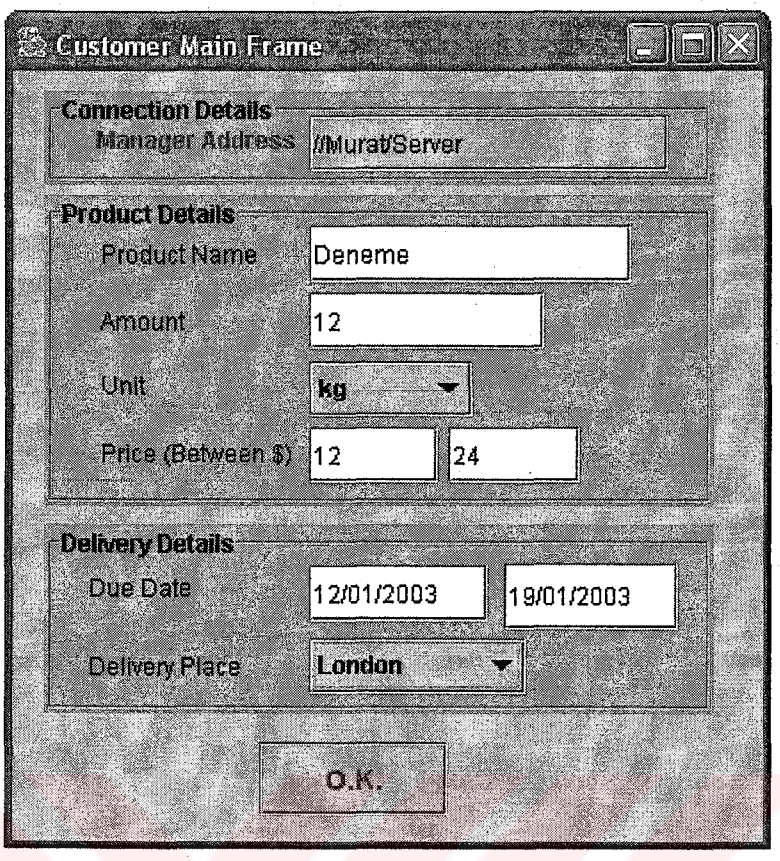
Müşteri, sistemden bir satın alma emrini gerçekleştirmek için kendi makinesinde bir Müşteri Alt-sistem'ini başlatmalıdır. Müşteriler bağlanacakları denetleyici etmenin adresini (URL) bilmek zorundadırlar.



Şekil 9.8 : Müşteri Alt-sistemi'nin yapısı

Müşteri Alt-sistemi Şekil 9.8'de görüldüğü gibi dört ana bileşenden oluşur.

- Tarihçe Veritabanı: geçmiş tedarik raporlarını içerir ve bir kullanıcı GUI aracılığıyla bu bilgileri sorgulayabilir ve herhangi bir anda tedarikçilerle ilgili bazı bilgileri değiştirebilir.
- Kullanıcı Grafik Arabirimi (GUI): kullanıcılarla etkileşim için (yani, bir müşteriden yeni bir satın alma sürecinin alınması veya Tarihçe Veritabanının araştırılması gibi) kullanılır (Şekil 9.9).
- Müşteri Etmeni: kullanıcının adına hareket eder.
- Adlandırma Yöneticisi: gezgin etmenlerin sistemde tek bir tanıtıcı gibi adlandırılmasında kullanılır.



Şekil 9.9 : Tasarlanan müşteri grafiksel kullanıcı ara birimi

Bir çok görev senaryosu mümkün olduğundan, kullanıcının farklı davranış karakteristiklerine sahip İşletme düzeyi Gezgin Etmenler yaratmasına izin verilerek sistemin esnekliği artırılır. Bir kullanıcı Müşteri Etmeni ile Müşteri GUI modülü aracılığıyla etkileşimde bulunur. Bir işlemin başlangıcında, kullanıcı gerekli bilgileri sağlar. Siparişler, ürünün tipi, ihtiyaç duyulan miktar, gönderileceği yer ve teslim tarihi gibi bilgileri içerir. Müşteri GUI'si kullanıcının işlemleri kontrol etmesine ve takibine, Tarihçe Veritabanı'ndan geçmiş işlemleri sorgulamasına imkan tanır.

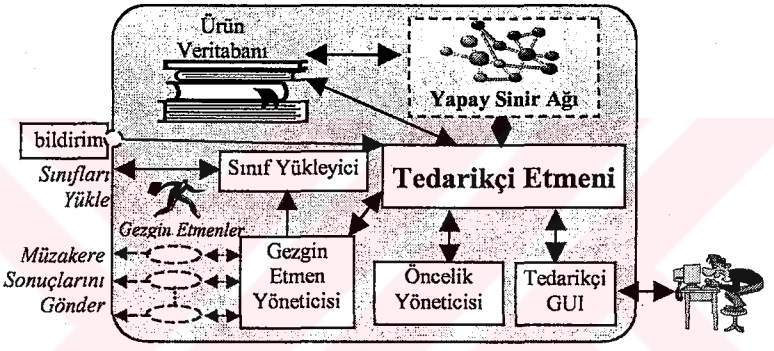
Müşteri Etmeni, Müşteri Alt-sistemi'nin ana işlemidir. Müşteri Etmeni, alt-sistemin başlangıç adımında yaratılan sabit bir etmendir. Son kullanıcılara, sorgulama işlemlerinin girilmesi ve işlemlerini gerçekleştirmek amacıyla oluşturduğu İşletme düzeyi Gezgin Etmenler ile iletişim kurulması için bir ara yüz sunmaktan sorumludur. Müşteri Etmeni, rutin ve basit görevleri gerçekleştirecek temel yeteneklere sahip olarak yaratılır. Fazla karmaşık görevler bir insanın yönlendirmesini gerektirebilir, fakat bir kez yönlendirildikten sonra etmenler gelecekte onları kullanabilirler. Bu ise Müşteri Etmeni'nde öğrenme metodolojilerinin kullanılması ile gerçekleştirilebilir. Müşteri Etmeni bir kullanıcıdan satın alma isteği aldığında, ürün bilgilerini

araştıracak ve sistemde imalatını/teminini sağlayacak bir Gezgin Etmen yaratır, ürünün temini için gerekli kriterleri belirler ve denetleyiciye İGE'yi dağıtır.

9.4.2 Tedarikçi Alt-sistemi

Bir tedarikçi, sisteme katılmak için kendi makinesinde Tedarikçi Alt-sistem'ini başlatmalıdır. Tedarikçi sistemi ilk kez yaratıldığında, adres ve ürün bilgileri ile sisteme kayıt olur. Eğer tedarikçi bir ürünün teslimine başlar veya bitirirse, yine sırasıyla kayıt olma ve kayıt silme işlemlerini gerçekleştirir.

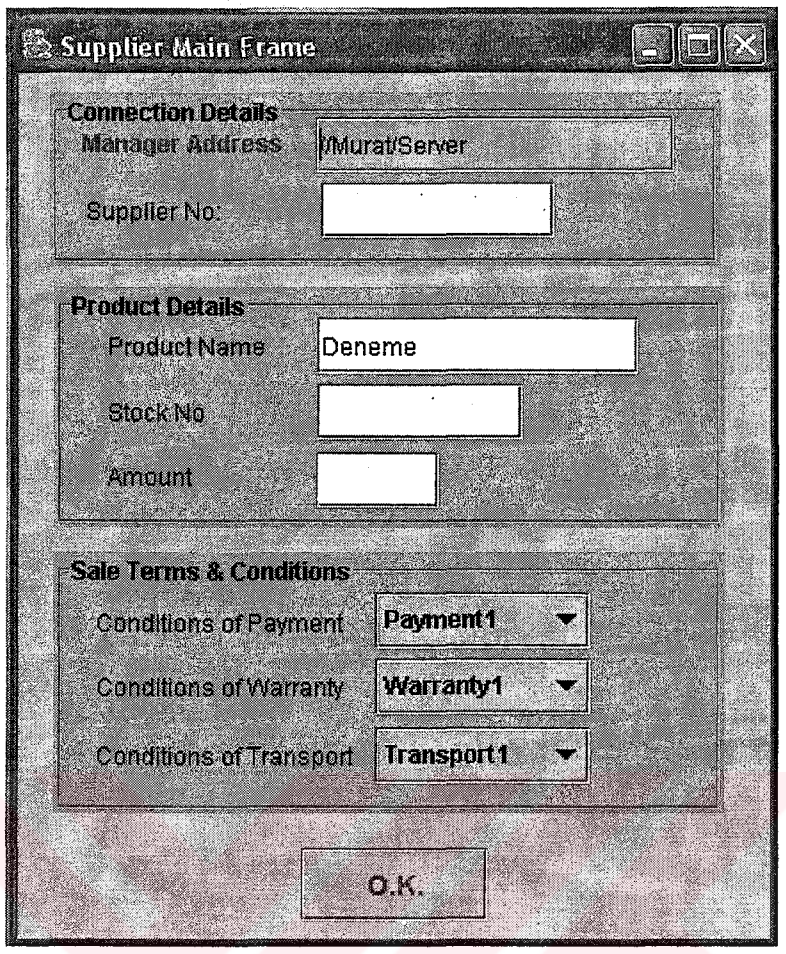
Her tedarikçi etmeni sisteme bağlanabilmek için denetleyicinin adresini bilmek zorundadır. Tedarikçi etmeni denetleyiciye ürün bilgilerini göndererek kayıt olur ve işletmenin kendisinden alımı gerçekleştirmesini bekler.



Şekil 9.10 : Tedarikçi Alt-sistemi'nin yapısı

Tedarikçi alt-sistemi Şekil 9.10'da görüldüğü gibi yedi ana bileşenden meydana gelir:

- Ürün Veritabanı: tedarikçinin ürün bilgilerini içerir.
- Kullanıcı Grafik Arabirimi (GUI): kullanıcılarla etkileşim için (yani, bir etmenden geçmişteki veya mevcut işlemler hakkında sorgulama yapılması veya satış stratejilerinin belirlenmesi gibi) kullanılır (Şekil 9.11).
- Gezgin Etmen Yöneticisi: gelen TGE'leri kontrol ve koordine eder.
- Öncelik Yöneticisi: bekleyen istekleri (TGE) kontrol eder ve en yüksek önceliğe sahip olanı seçer.
- Sınıf Yükleyici: Denetleyici'den gerekli sınıfları yükler.
- Yapay Sinir Ağı: tedarikçinin gelecek dönemlere ait ürün taleplerini tahmin ederek zamanında ürün dağıtımını gerçekleştirmesini sağlar.
- Tedarikçi Etmeni: kullanıcının adına hareket eder.



Şekil 9.11 : Tasarlanan tedarikçi grafiksel kullanıcı ara birimi

Bir çok yazılım sistemi, olayları gerçeklemede ilk gelen ilk çıkar (FIFO) gibi basit kontrol mekanizmaları kullanırlar. Ancak, tedarik zincirindeki etkileşimler daha karmaşık kontrol mekanizmalarını içerirler. Örneğin, önemli bir sipariş geldiğinde, diğerleri ertelenerek ilk olarak o gerçekleştirilebilir.

Etmenin diğer yükümlülükleri kadar (maliyet, teslim zamanı gibi) performans ölçütleri de onun önceliğini etkiler. Bu öncelikler gelen isteklerin hangi sırada işleneceğine ilişkin sırayı belirler. Örneğin, tedarikçi etmeni müşterileri A-B-C sınıflandırmasına göre önceliklendirebilirler. Eğer bütün siparişler aynı önceliğe sahiplerse, o zaman FIFO kullanılır.

Tedarikçi etmeni, son kullanıcılara bilgileri girmeleri için bir ara yüz sunan, işlemleri kontrol eden ve sorgulama görevlerini giren sabit bir etmendir. Tedarikçi Etmeni, işletmeden gelen satın alma emirlerini gerçekleştirir ve kullanıcı tarafından belirlenen satış stratejilerine göre işlerin nasıl yürütüleceğine karar verir. Tedarikçiler

farklı ürünler satabileceklerinden, bir tedarikçi etmeni sisteme çevrimiçi olarak konmadan kişiselleştirilmelidir.

Yapay sinir ağı ile İşletme tarafından talep edilen ürünlerin zamanında teslimine olanak sağlayacak bir biçimde tahmin edilmesi sağlanır. Ürün veri tabanı da güncel talep bilgileri ile güncellenir. Java'da gerçeklenen YSA'nın detayları Bölüm 9.4.4'de anlatılacaktır.

9.4.3 Kontrol/Eniyileme Sistemi

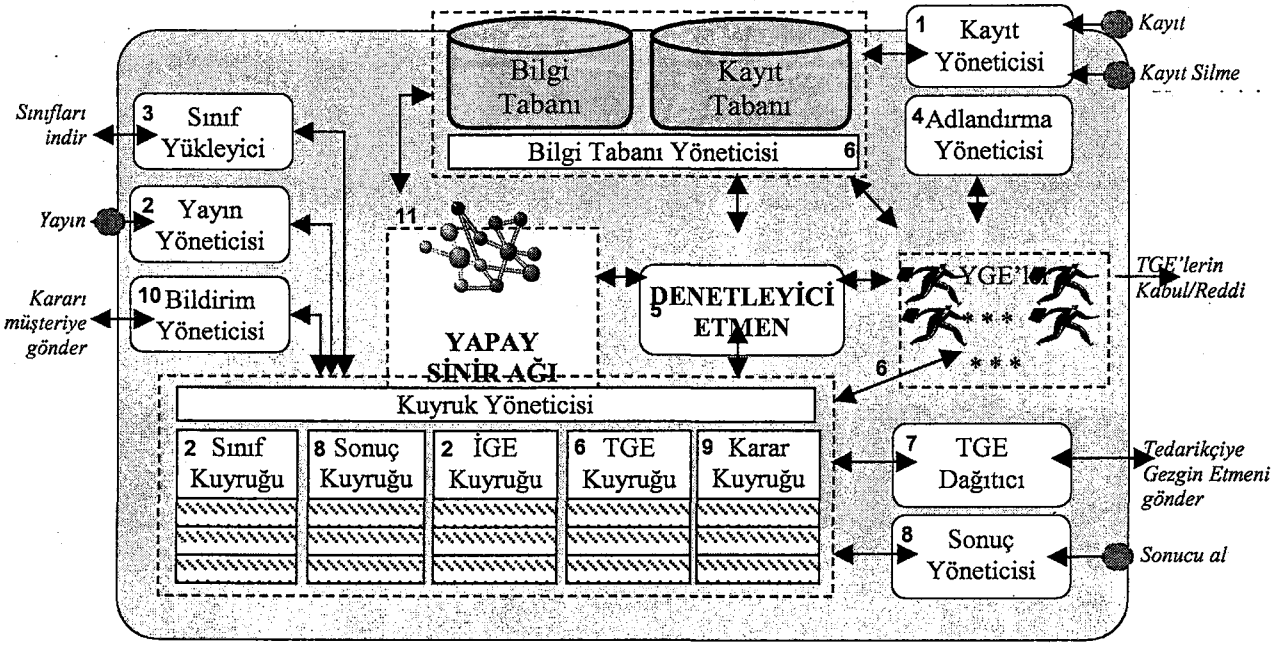
Kontrol/Eniyileme Sistemi geliştirilen modelde en önemli rolü üstlenir. Müşterilerle tedarikçiler arasındaki tüm süreçleri kontrol eden, mantıksal olarak (fiziksel olarak da) merkezde (işletmede/imalatçıda) yer alan bir modüldür. Kontrol/Eniyileme Sistemi'nin ana bileşeni "Denetleyici"dir. Denetleyici, özellikle tedarik zincirinde pek çok müşteri ve tedarikçi bulunduğu, araştırma maliyetleri göreceli olarak yüksek olduğu veya güvenilir hizmete gereksinim duyulduğu zaman faydalıdır. Denetleyici'nin iç yapısı Şekil 9.12'de verilmiştir.

İstemci-sunucu paradigması, sistemin yaşam çevriminde karşılaşılan problemleri tatmin edici düzeyde çözmede sıklıkla yeterli değildir. Genellikle bir tedarikçi veya müşteri, tedarik zinciri sistemine tasarım periyodundan sonra eklenmek istenir. Bu sistemde Denetleyici, satın alma işlemlerini yayınlayıp, sistemin tedarikçi bileşenlerince bildirimler aracılığıyla kullanılan kayıt/yayın paradigmasını gerçekler.

Sistemde iki ana yönetici modülü vardır: Bilgi Tabanı Yöneticisi ve Kuyruk Yöneticisi modülü.

Bilgi Tabanı Yöneticisi iki önemli veritabanını işler:

- Bilgi Tabanı: sistemdeki tedarikçiler, müşteriler ve ürünlerle ilgili, müşteri ve tedarikçi etmenleri arasındaki mesaj hareketine göre belirlenen istatistiksel bilgileri saklar.
- Kayıt Tabanı: tedarikçilerce temin edilen ürünlerin isimleri gibi basit bilgileri saklar. Ayrıca, tedarikçilerin yetkilendirilmesi için kullanıcı adı ve şifre bilgilerini de içerir.



Şekil 9.12 : Denetleyicinin iç yapısı

Kuyruk Yöneticisi, Denetleyici'de beş ana kuyruğa erişir. Bu kuyruklar eşzamanlı olarak Denetleyici modülünün yürütülmesinde kullanılır.

- Sınıf Kuyruğu: İGE'ler tarafından indirilecek sınıfların isim ve adreslerini içerir.
- İGE Kuyruğu: müşterilerden gelen İGE'lerin serileştirilmiş formunu içerir.
- TGE Kuyruğu: tedarikçilere giden TGE'lerin serileştirilmiş formunu içerir.
- Sonuç Kuyruğu: TGE'lerden gelen sonuçları içerir.
- Karar Kuyruğu: Müşteri Etmenine gönderilecek, İGE'ye ait karar verilerini içerir.

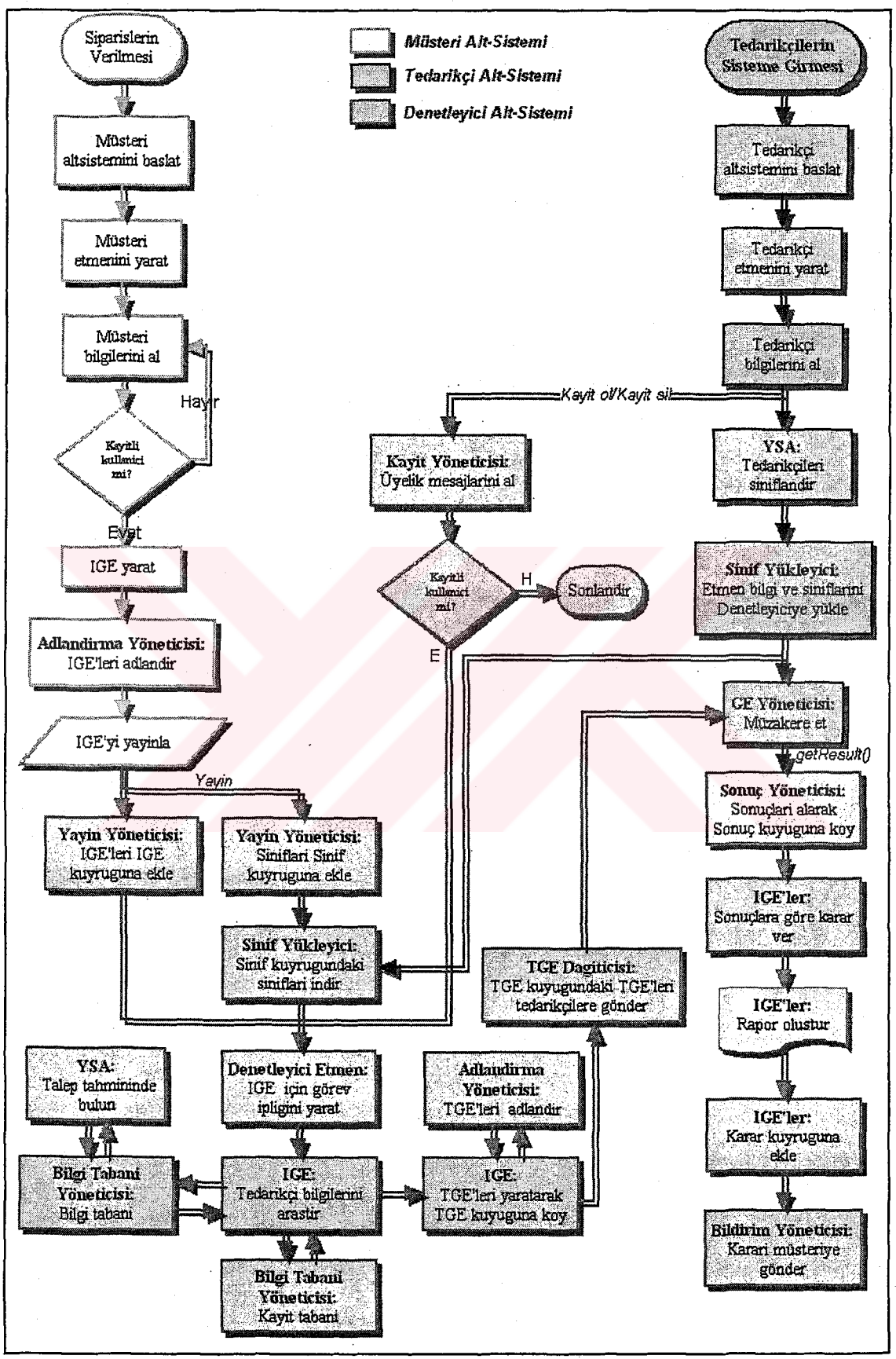
Bu kuyruklarda, her bir etmen sistemdeki gelen mesajları FIFO kuralına göre seçer. Bütün kuyruklar için gelen mesajların aktivasyon zamanına göre sıralandıkları genel bir listesi vardır ve en erken mesajı olan etmen ilk olarak işleme alınır.

Denetleyici, sistemde paralel olarak çalışan dokuz tane yönetici modülünden oluşur. Bunlar; Sınıf Yükleyici, Yayın Yöneticisi, Bildirim Yöneticisi, Kayıt Yöneticisi, TGE Dağıtıcısı, Sonuç Yöneticisi, Adlandırma Yöneticisi, Yapay Sinir Ağı ve Denetleyici Etmen.

Bu yöneticilerin görevleri ve Denetleyici'deki satın alma sürecinin yürütme akışı şu şekildedir:

1. Kayıt Yöneticisi, tedarikçilerden kayıt ol ve kayıt sil metotları vasıtasıyla gelen üyelik mesajlarını alır, tedarikçi kullanıcı adı ve şifresini Kayıt Tabanı ile karşılaştırır ve uygun hareket tarzına göre devam eder.
2. Yayın Yöneticisi, yayın metodu aracılığıyla Müşteri Etmeni'nden gelen İGE'lerin serileştirilmiş formunu alır ve İGE Kuyruğuna ekler. Aynı yöntemle gerekli sınıfların isim ve adreslerini de alır ve Sınıf Kuyruğuna ekler.
3. Sınıf Yükleyici, Sınıf Kuyruğu'ndaki sınıfları Müşteri alt-sistemi'nden Müşteri Etmeni'nin `downloadClass()` metodunu çağırarak indirir.
4. Adlandırma Yöneticisi TGE'lerinin adlandırma işlemlerinde kullanılır.
5. Denetleyici Etmen, İGE Kuyruğundaki her bir İGE için bir görev ipliğini (thread) yaratır ve etkinleştirir.
6. Her bir İGE, hem Bilgi Tabanı'na hem de Kayıt Tabanı'na okuma ve yazma yetkisine sahiptir. Bir İGE bu tablolarda tedarikçilerin bilgilerini araştırır, her bir tedarikçi için bir TGE yaratır ve daha sonra her bir TGE'nin serileştirilmiş biçimini TGE Kuyruğu'na koyar.
7. TGE Dağıtıcısı TGE Kuyruğu'ndaki her bir TGE'ni ilgili tedarikçiye gönderir.
8. Her bir TGE arama ve müzakere sonuçlarını `getResult()` metodu aracılığıyla Denetleyici'ye gönderir. Bu sonuçlar Sonuç Yöneticisi tarafından alınır ve Sonuç Kuyruğuna eklenir.
9. İGE'ler Sonuç Kuyruğu'ndaki sonuçları dikkatlice gözden geçirir ve karar verir. Bir karara vardıktan sonra, bir İGE bir karar raporu oluşturur ve raporu Karar Kuyruğu'na koyar.
10. Bildirim Yöneticisi Karar Kuyruğu'ndaki her bir kararı hedef müşteriye gönderir.
11. Yapay Sinir Ağı: farklı kriterlere göre tedarikçileri sınıflandırır ve bir sonraki dönem için müşteri taleplerini tahmin eder.

Denetleyici Etmen, Kontrol/Eniyileme Sistemi'nin ana bileşenidir. Sistemdeki bütün işlemleri kontrol eder, gelen mesajları değerlendirir ve Bilgi Tabanı ile Kayıt Tabanı'ndaki bilgileri, YSA modülünde kullanarak hedef tedarikçilerin listesini oluşturur. Denetleyici Etmen, gelen gezgin etmenler için TGE'lerinin yaratılıp çalıştırılması ve uygun tedarikçilere dağıtılmasını sağlayacak bir platform sağlar. Ayrıca, yayın/kayıt paradigmasına uygun olarak kayıt ve dağıtım işlemlerini de destekler. GETTZY sisteminde gerçekleştirilen işlem ve süreçlerin akış şeması bir bütün olarak Şekil 9.13'de verilmiştir.



Şekil 9.13 : GETTZY sistemine ait süreçlerin akış şeması

9.5 Yapay Sinir Ağı Modülleri

Kontrol/Eniyileme sistemi ile Tedarikçi alt-sisteminde talep tahmini ve tedarikçilerin sınıflandırılması işlemlerinde ÇKİB YSA ve Kohonen ağları kullanılmıştır. Bu alt-bölümde gezgin etmen sisteminde kullanılan YSA mimarilerinin Java'da gerçeklemelerinin detayları açıklanacaktır.

9.5.1 Çok Katmanlı İleri Beslemeli Yapay Sinir Ağı ve Geriye Yayılım Prosedürünün Java ile Gerçeklenmesi

Geriye yayılım algoritmasının gerçeklenmesinde dört farklı veri ve parametre seti kullanılmıştır. İlk grup verilerin yönetiminde kullanılan parametreleri içermektedir. İlk parametre DataSet nesnesine referans olarak *dataset* nesnesidir. İkinci parametre eğitme verilerinin saklandığı *data* vektörüdür. Güncel kayıt numarasını göstermek için *recInx*, veri kümesindeki toplam kayıt sayısı için *numRecs* ve her bir kayıt için kaç adet alanın olduğunu göstermek için de *fieldsPerRec* parametreleri kullanılmıştır.

```
// veri parametreleri
private DataSet dataset;
private Vector data;
private int recInx = 0;
private int numRecs = 0;
private int fieldsPerRec = 0;
```

İkinci veri grubu ağın mimarisi ile ilgili parametreleri içerir. Girdi, çıktı ve saklı katmanda kaç adet nöronun bulunacağı *nInputs*, *nOutputs* ve *nHid1* ile gösterilir. Ağda birden fazla saklı katmanın yer alması isteniyorsa ağın yapısı kolaylıkla genişletilebilir. Toplam nöron sayısı ve toplam ağırlık sayısı da *nUnits* ve *nWeights* ile gösterilmiştir.

```
// ağ mimarisi parametreleri
private int nInputs;
private int nHid1;
private int nOutputs;
private int nUnits;
private int nWeights;
```

Ağ kontrol parametreleri olarak da öğrenme oranı için *learnRate*, momentum için *momentum* ve hatanın toleransını kontrol etmek için de *tolerance* kullanılmıştır.

```
// ağ kontrol parametreleri
private double learnRate;
private double momentum;
private double tolerance;
```

Kullanılan dördüncü ve son parametre grubu ise ağ verileridir. Aktivasyon değerlerini, ağırlıkları ve eşik değerleri saklamak için *activations*, *weights* ve *thresholds* dizileri kullanılmıştır. Aşağıda verilen diğer diziler de ağın eğitilmesi esnasında kullanılan parametreler içindir.

```
// ağ verileri
private double activations[];
private double weights[];
private double thresholds[];
private double wDerivs[];
private double tDerivs[];
private double tDeltas[];
private double teach[];
private double error[];
private double deltas[];
private double wDeltas[];
```

Verilerin bir tekst dosyasından hafızaya yüklenmesi ve ekrandan da izlenebilmesi için *DataSet* sınıfı tasarlanmış ve gerçeklenmiştir. ÇKİB YSA'nı yaratmadan önce ağı eğitmede kullanılacak veri seti bu sınıfın bir örneği kullanılarak programa yüklenir. Bilgilerin dosyadan hafızaya yüklenmesi için `loadDataFile()` metodu kullanılır.

Yapay Sinir Ağları genellikle girdi verilerinin belirli bir aralıkta olmasına ihtiyaç duyduğundan verilerin normalize edilmesi gerekir. Eğer girdiler sürekli değerler alıyorsa 0 ile 1 arasında doğrusal bir ölçeklendirme yapılır. Girdi değerleri kesikli ise, değişkenin indeks değeri alınarak N boyutlu vektöre dönüştürülür. Örneğin, eğer kesikli bir değişken olarak tanımlanan {*evet*, *hayır*, *belki*} şeklindeki üç alternatif dizgi varsa, her bir sembol uzunluğu kesikli değişken sayısına eşit ikilik düzende bir koda dönüştürülür. Böylece, {*evet*} (1 0 0)'a, {*hayır*} (0 1 0)'a ve {*belki*} de (0 0 1)'e eşlenir.

Ölçeklendirme işlemi için kullanılan `normalizeData()` metodunda öncelikle normalize edilecek verilerin saklanacağı bir veri vektörü yaratılır ve toplam kayıt sayısı da ham verideki kayıt sayısına eşitlenir (Şekil 9.14). Ham veri kümesindeki ilk elemandan başlayarak son elemana kadar tek tek o değişkene ait `normalize()` metodu çağrılarak kayıt normalize edilir. Daha sonra normalize edilmiş bu kayıt normalize veri kümesine eklenir. Bütün işlem bittiğinde orijinal veri setine karşılık

gelen ve yapay sinir ağında kullanılmaya hazır bir normalize veri vektörü elde edilmiştir.

```
void normalizeData() {
  - normalize veri vektörünü yarat;
  - kayıttaki alan sayısını normalize edilecek kayıt sayısına eşitle;
  - ham veri kümesini veri elemanlarına eşitle;
  while (ham veri kümesi daha elemana sahip) {
    inx ← 0;                                  // indeksi sıfırla
    - normalize kayıt sayısını yarat;
    for (sayaç kayıttaki alan sayısına eşit oluncaya kadar) {
      Variable ← sonraki değişken;// sonraki değişkeni bul
      inx ← Variable.normalize;    //normalize edilmiş değerin indeksi
    }end-for
    - normalize edilmiş kayıdı normalize veri kümesine ekle
  }end-while
}end-method
```

Şekil 9.14 : `normalizeData()` metodu içinde yürütülen işlem adımları

Sürekli değişkenler için kullanılan normalizasyon işleminde en büyük değer ile en küçük değer arasındaki fark bulunarak her bir ham değer bulunan bu değere bölünür (Şekil 9.15). Girdi verilerinin kesikli olması durumunda ise bir sembol N koddan birisine dönüştürülür. Yapay sinir ağının eğitilmesi tamamlandıktan sonra sonuçlar gösterilirken `denormalize()` metodu eski aralığına dönüştürülür.

```
int normalize(String inValue, double[] outArray, int idx){
  - inValue'yu double formatına dönüştür.
  if1 (inValue minimum veri değerine eşit veya küçükse) {
    outValue ← min;                           // normalize değeri min'e eşitle
  } end-if1
  if2 (inValue maksimum veri değerine eşit veya büyükse) {
    outValue ← 1;                             // normalize değeri 1'e eşitle
  } end-if2
  else {
    factor ← max - min;                       // normalizasyon aralığını bul
    outValue ← inValue / factor;              // normalize et
  }end-else
  outValue'nun indeksini bul;
  return indeks;
} end-method
```

Şekil 9.15 : `normalize()` metodu içinde yürütülen işlem adımları

Geriye yayılım YSA'nı oluşturmak için `createNetwork()` metodu kullanılır (Şekil 9.16). Bu metot girdi olarak girdi nöron sayısı, saklı nöron sayısı ve çıktı nöron sayısını alır ve kullanılacak bazı ağ parameterelerini hesaplayarak dizilere atar.

```
void createNetwork(int nIn, int nHidden, int nOut) {
   - ağ parametrelerini set et
   - toplam nöron sayısını ve ağırlık sayısını hesapla
   learnRate ← 0.2;              // öğrenme oranını 0.2'ye eşitle
   momentum ← 0.7;               // momentumu 0.7'ye eşitle
   tolerance ← 0.1;              // toleransı 0.1'e eşitle
   aveMSE ← 0;                   // ortalama MSE'yi sıfırla
   - ağ ağırlıklarını ve hata dizilerini yarat
   - ağ ağırlıklarını ve hata dizilerini başlangıç durumuna getir
} end-method
```

Şekil 9.16 : `createNetwork()` metodu içinde yürütülen işlem adımları

YSA'nın ileri besleme işlemi aşağıdaki metotlar kullanılarak gerçekleştirilir:

```
public void readInputs(){...}
public void computeOutputs(){...}
public double logistic(double sum){...}
```

`readInputs()` metodu veri setindeki bir kayda ait girdi değerlerini alarak girdi katmanındaki nöronların aktivasyonlarına kopyalar. Ayrıca hedef değerleri de ilgili diziye kopyalar. `computeOutputs()` metodu ise ilk katmandan başlayarak eşik değerlerin toplamını ilgili ağırlıklarla çarpıp aktivasyon değerini hesaplamak için `logistic()` metodunu çağırır.

Geriye yayılım prosedürü ise son katmandan ilk katmana doğru hesaplamaların yapılması ile gerçekleştirilir. Bu amaçla öncelikle çıktı katmanının hata değerleri hedef değerler ile hesaplanan değerler arasındaki fark olarak hesaplanır. Bu amaçla `computeErrors()` metodu çağırısı yürütülür ve elde edilen aktivasyon değerleri ile hedef değerler arasındaki farklar bulunur (Şekil 9.17). Yine bu metot yardımı ile hata kareleri ortalaması da hesaplanır. Daha sonra aktivasyon işlevinin türevi kullanılarak delta değerleri de bulunur.

```
void computeError() {
  for1 (girdi sayısı toplam nöron sayısına eşit olana kadar) {
    error[] ← 0;                                  // hataları sıfırla
  }end-for1

//çıktı katmanının hata ve delta değerlerinin hesaplanması

  for2 (çıktı katmanındaki nöronlar) {
    - hedef değer ile aktivasyon arasındaki farkı hata olarak belirle
    sumSqrError ← error[]*error[];
    if1 (hatanın mutlak değeri toleranstan küçükse) {
      error[]← 0.0;                               // yeteri kadar yakın
    } end-if1
    - deltaları hata ile aktivasyon işlevinin türevininçarpımı olarak hesapla
  } end-for2

  // saklı katmandaki döronların hatalarının hesaplanması

  for3 (çıktı katmanındaki nöronlar) {
    for4 (saklı katmandaki nöronlar) {
      - ağırlıklandırılmış türevleri hesapla
      - saklı katman hatalarını hesapla
    } end-for4
  } end-for3
  - saklı katmandaki nöronların delta değerlerini hesapla

  // girdi katmanındaki döronların hatalarının hesaplanması

  for5 (saklı katmandaki nöronlar) {
    for6 (girdi katmanındaki nöronlar) {
      - ağırlıklandırılmış türevleri hesapla
      - girdi katmanı nöronlarının hatalarını hesapla
    } end-for6
  } end-for5
} end-method
```

Şekil 9.17 : `computeErrors()` metodu içinde yürütülen işlem adımları

Ağırlıkları güncellemek için `adjustWeights()` metodu yürütülür (Şekil 9.18). Bu metot ilk olarak güncel ağırlık deltalarını hesaplar ve ardından bu deltaları ağırlıklara ekler. Ağırlıklardaki değişim toplamını göstermek için bir dizi kullanılmıştır. Örüntünün güncellenmesinde bu dizi sadece güncel örüntüden kaynaklanan değişimleri içerir. Ancak, eğer grup güncellemesi kullanılıyorsa bu dizi veri kümesindeki bütün örüntülerden kaynaklanan değişimleri saklar. Ağırlıklar güncellendikten sonra bu dizi sıfırlanır.

```
void adjustWeights() {
  for1 (sayaç ağırlıkların sayısına eşitleninceye kadar) {
    - deltaları türevle öğrenme oranının çarpımına momentum ile delta
      değerinin çarpımını ekleyerek güncelle
    - ağırlıkları güncelle
    wDerivs[] ← 0.0;                          // türevleri sıfırla
  } end-for1
  for2 {
    - eşikler için aynı işlemleri tekrarla
  } end-for2
  - eğer bir döngünün sonuna gelinmişse HKO'nı hesapla
} end-method
```

Şekil 9.18 : `adjustWeights()` metodu içinde yürütülen işlem adımları

9.5.2 Kohonen Ağlarının Java ile Gerçeklenmesi

Kohonen ağlarının Java'da gerçeklenmesi ÇKİB YSA'ndan çok farklı olmamakla birlikte temel değişiklik gözetimli öğrenme ile gözetimsiz öğrenmeden kaynaklanmaktadır. Kohonen ağları gözetimsiz öğrenmeyi uyguladığından, hedef çıktı değerleri yoktur sadece girdi değerleri vardır.

Kohonen ağlarının gerçeklenmesinde dört farklı veri ve parametre seti kullanılmıştır. İlk grup verilerin yönetiminde kullanılan parametreleri içermektedir Bütün girdi verileri `InputMatrix sınıfında tanımlanmış` *imtrx* değişkeninde saklanır. *inputX*, *inputY* ve *inputZ* dizileri girdilerin farklı boyutlardaki değerlerinin saklanmasında kullanılır. *inputs* ve *weights* dizileri girdilerin görsel olarak gösteriminde kullanılır.

```
// veri parametreleri
InputMatrix imtrx;
int inputX[];
int inputY[];
int inputZ[];
Point3D inputs[];
Point3D weights[];
```

İkinci veri grubu ağın mimarisi ile ilgili parametreleri içerir. Girdi ile çıktı katmanında kaç adet nöronun bulunacağı *inputSize*, *mapSizeX* ve *mapSizeY* ile gösterilir. Girdilerin *x*, *y* ve *z* bileşenleri sırasıyla *xSize*, *ySize* ve *zSize* değişkenlerinde saklanır. Girdilerin kaç boyutta gösterileceği *inputDimension* parametresi ile kontrol edilir.

```
// ağ mimarisi parametreleri
int inputSize;
int xSize;
int ySize;
int zSize;
int mapSizeX;
int mapSizeY;
int inputDimension;
```

Ağ kontrol parametreleri olarak öğrenme oranı için *initRate*, komşuluk işlevinin hesaplanmasında kullanılmak üzere *initArea* ve *stopArea* , maksimum çevrim sayısı için *maxCycle* ve öğrenmenin gerçekleşip gerçekleşmediğini kontrol için de *itLearns* parametreleri kullanılmıştır.

```
// ağ kontrol parametreleri
boolean itLearns;
int initRate;
int initArea;
double stopArea;
int maxCycle;
```

Kullanılan dördüncü ve son parametre ise Kohonen ağı sınıfıdır (`KohonenFM kfm`). Bu sınıf uygulama tarafından başlatılır ve soyut sınıf `NeuralNet`'i genişletir. Ağın yaratılması, yürütülmesi ve sonlandırılmasında kullanılan diğer tüm sınıfları içerir.

Bir Kohonen ağını tanımlamak için ana uygulamada `KohonenFM` sınıfının bir örneği `new KohonenFeatureMap()` komutu ile `createNetwork()` metodu yürütülerek yaratılır (Şekil 9.19). Bu metot girdi olarak girdi nöron sayısı, çıktı gridindeki nöron sayılarını alır ve kullanılacak bazı ağ parametrelerini hesaplayarak dizilere atar. Çıktı haritası `createMapLayer()` metodu çağrılarak yaratılır. Girdi katmanındaki nöronlar ile çıktı haritasını oluşturan nöronlar arasındaki bağlantı `connectLayers()` metodu yürütülerek oluşturulur.

```
void createNetwork() {
   - KohonenFM sınıfının bir örneğini yarat
   - ağ parametrelerini set et
   - çıktı haritasını yarat
   - ağ ağırlıkları başlangıç durumuna getir
   - girdi nöronları ile çıktı tabakası arasındaki bağlantıları oluştur.
} end-method
```

Şekil 9.19 : `createNetwork()` metodu içinde yürütülen işlem adımları

Bir girdi örüntüsü aşağıdaki metotlar kullanılarak işlenir:

```
void learn(){...}
MapNeuron computeActCentre(){...}
void changeWeights(){...}
void decreaseActArea(){...}
public void setWeights(){...}
```

`learn()` metodu bir öğrenme çevrimini yürütür ve genellikle bir döngü içerisinde öğrenme tam olarak gerçekleşinceye kadar çağrılır (Şekil 9.20). Öğrenmede durdurma kriteri olarak komşuluk yarıçapı (aktivasyon alanı) veya maksimum öğrenme çevrim sayısı kullanılmıştır. Her bir öğrenme çevriminde yeni girdi değerleri alınır, girdi değerleri ile ağırlıklar arasındaki Öklit mesafesi hesaplanarak aktivasyon merkezi bulunur. Ağın ağırlıkları `changeWeights()` metodu yürütülerek güncellenir ve aktivasyon alanı `decreaseActArea()` metodu çağrılarak azaltılır. Eğer durdurma kriterlerinden birisi gerçekleşmişse öğrenme çevriminden çıkılır.

```
void learn(){
  if (aktivasyon alanı durdurma kriterinden büyükse ve öğrenme çevrimi
      maksimum öğrenme çevriminden küçükse veya maksimum öğrenme
      çevrimi -1'e eşitse){
      learningCycle ← learningCycle+1;
      - girdi katmanına girdi değerlerini ver;
      - aktivasyon merkezini hesapla;
      - ağın ağırlıklarını değiştir;
      - aktivasyon alanını daralt;
      return;
  } end-if
  else {
    stopLearning ← true;
    return;
  } end-else
} end-method
```

Şekil 9.20 : `learn()` metodu içinde yürütülen işlem adımları

Kohonen ağında girdi verileri ile eşleştirildikleri çıktı değerlerini grafiksel olarak gösteren nesne metodları da kullanılmıştır. Yapılan seçime göre iki veya üç boyutlu olarak girdi verileri ve çıktı haritası çizilir.

10. UYGULAMA SONUÇLARI VE DEĞERLENDİRME

Bu bölümde, tezde önerilen gezgin etmen temelli tedarik zinciri yönetimi sisteminin gerçek yaşamdaki başarımını görmek amacıyla, otomotiv endüstrisinde üretim yapan uluslar arası bir firmanın (Valeo A.Ş.) verileri ile gerçekleştirilen testler ve elde edilen sonuçlar incelenecektir.

Valeo'nun Türkiye'deki varlığı 1989 yılında kendi lisansı ile üretim yapan Transtürk debriyaj hisselerinin %51'ini satın alması ve mevcut üretim tesislerine yatırımlar yapması ile başlamıştır. Eylül 1993'de ise şirketin tüm hisseleri Valeo tarafından satın alınmıştır. 1998 yılında İtalya'nın Mondovi şehrindeki Valeo Frizioni fabrikasından komple traktör hattının Bursa'ya transferi başlatılmış, transferin tamamlanmasının ardından işletme dünya üzerindeki en büyük traktör debriyajı üreticileri arasına girmiştir.

Valeo, toplam 143 fabrika, 53 Ar-Ge merkezi, 10 dağıtım şirketi ve 70.000 çalışanıyla 25 ülkede faaliyet göstermekte olup ürünleri 97 ülkede pazarlanan dünyanın en büyük bağımsız otomotiv sistemleri üreticilerinden biridir. Bu fabrikalar arasında sadece 14 tanesi debriyaj üretimi yapmaktadır. Valeo'nun bugünkü başarısı kendi iç sisteminde geliştirdiği 5 aks sistemine bağlıdır. Bu sistemi oluşturan öğeler; personelin katılımı, üretim sistemi, imalatçılarla bütünleşme, sürekli yenilik ve toplam kalitedir.

İşletme 4875 m^2'lik kapalı alanı ile binek, ağır vasıta ve traktör olmak üzere tüm araçları kapsayan geniş bir ürün gamına sahiptir. OEM'deki yurtiçi pazarı %90 olup müşterileri Renault, Otosan, Karsan, Tofaş, Otokar, Otoyol, Chrysler, Uzel, Türk Traktör, Tümosan, Hema ve TZDK'dır. Yurtdışı OEM müşterileri ise İtalya'da Fiat, Same, Case New Holland, Landini, Goldoni ve AEBI, Fransa'da Renault Agri, Cezayir'de PMA, İran'da ITMCO Pars Khodro, İran Khodro, Çin'de Luoyang Tractors, Hindistan'da Mahindra&Mahindra ve Same Greaves'tir. Valeo, kategorisinde dünyada ilk 10 şirket içerisinde yer almaktadır. Valeo Otomotiv Sistemleri Endüstrisi A.Ş. Bursa Bölümü otomobil, ticari araç ve traktörler için debriyaj ve transmisyon parçaları üreten bir kuruluştur. Merkezi Paris'te bulunan Valeo S.A. grubunun bir üyesidir.

10.1 Test Ortamı

GETTZY ve YSA kullanan modüller için yapılan tüm başarım ölçümleri, özellikleri Tablo 9.1'de verilen yapı üzerinde gerçekleştirilmiştir. Yerel alan ağının iş yükü günün değişik saatlerinde farklılıklar gösterdiği için ölçümler de günün farklı saatlerinde 10'ar kez tekrarlanarak uygulanmıştır. İzleyen bölümlerde verilecek olan ölçüm değerleri 10'ar kez uygulanmış olan bu testlere ait değerlerin aritmetik ortalamaları alınarak elde edilmiş olan sonuçlardır.

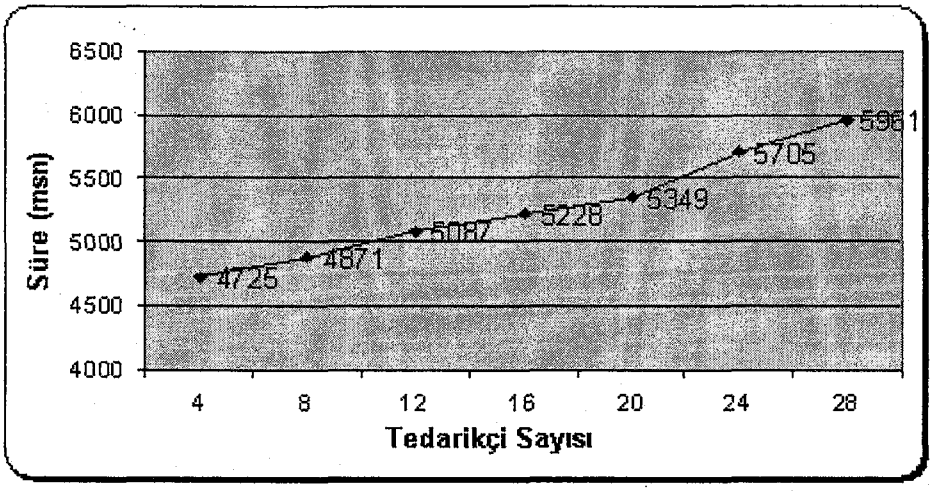
Her ne kadar testler yerel alan ağı üzerinde gerçekleştirilmiş olsa da, Internet üzerinde yapılacak bir uygulamada, gezgin etmen temelli sisteme ait temel işlemlerde bir farklılık olmayacaktır. Yalnızca, müşterek uygulamaların birbirleri arasındaki fiziksel uzaklıktan kaynaklanan iletişim gecikmesi değişebilecek, tüm diğer parametreler sabit kalacaktır.

Tablo 10.1 : Başarım ölçümlerinde kullanılan sistemin özellikleri

Özellik	Açıklama
Kullanılan ağ tipi	Yerel alan ağı (100 Mbit Ethernet)
Ağ işletim sistemi	Windows XP
Bilgisayar özellikleri	Pentium III 800 MHz işlemci, 256 MB RAM, Windows XP işletim sistemi
Kullanılan bilgisayar adedi	4-28

10.2 Gezgin Etmen Sistemi'nden Elde Edilen Test Sonuçları

Gezgin etmen sisteminin performansını değerlendirmede siparişler için karar verme sürelerini uygun bir kıyas kriteri olarak seçilmiştir. Sipariş karar süresini, siparişlerin alınması ile SMA Sonuç raporundan seçimin yapılması arasında geçen toplam süre olarak tanımlanmıştır. Bu analizde, siparişlerin yaratılması müşteri tarafından Kontrol/Eniyileme Sistemi'ne bir siparişin verilmesi ile gerçekleştirilir. Öncelikle, Kontrol/Eniyileme Sistemi'nin dağıtım etkinliğini, sisteme eşanlı olarak yüklenen siparişlere göre belirleyecek bir amaç belirlenmiştir. Bu analizdeki hedef, sistemin performans ve ölçeklendirilebilirliğini incelemektir.



Şekil 10.1 : Siparişlerin iletilmesi ve karar verilmesi süreleri

Tek bir denetçi, müşteri ve birden fazla tedarikçinin olduğu bir test tasarlandı. Testte kullanılan bilgisayarlar 512 MB SDRAM hafızaya sahip Compaq 1000 MHz Intel Pentium III'lerdir. Tedarikçilerin sayısı 4 ile 28 arasında değiştirilerek denemeler yapılmıştır. Tek sipariş kaynağı kullanılan deneylerde sipariş oluşturma oranı, sistemdeki diğer makinelerin verilen siparişi bir sonraki sipariş alınıncaya kadar gerçekleştirmesine yetecek kadar yavaştır. Böylelikle siparişlerin dağıtımında çakışma olmasının önüne geçilmiş, teorik olarak en iyi sipariş karşılama süreleri hesaplanabilmiştir. Şekil 10.1'de siparişlerin dağıtım ve karar verme süreleri gösterilmiştir. Doğal olarak tedarikçi sayısı arttıkça, tedarikçilerin sorgulanması ve bir kararın oluşturulması süreci de uzamaktadır.

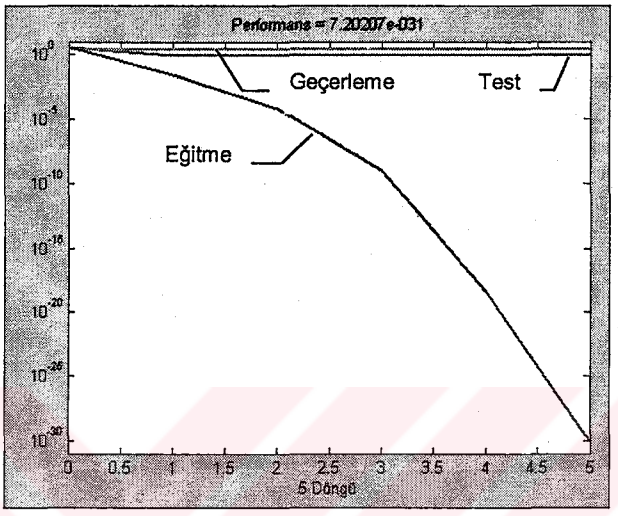
10.3 YSA'dan Elde Edilen Test Sonuçları

10.3.1 Talep Tahmini Modülü Test Sonuçları

Talep tahmini modülünün performansını değerlendirmek için, Valeo'nun 2001, 2002 ve 2003 yıllarında, yaklaşık 115 farklı firma için yaptığı üretimin aylık miktarları kullanılmıştır. Günlük veya haftalık satış bilgileri bulunmadığı için 34 aylık bir üretim periyodundaki üretim miktarları örnek verileri olarak alınmıştır. Bir yılda yaklaşık 230 farklı ürün tipinin üretildiği fabrikada, aylık toplam üretim miktarı 80.000 ile 380.000 arasında değişmektedir. Aylık toplam satış miktarının yanı sıra yoğun üretimi yapılan ürünlerden KIT-3 AMB.İ180, KIT-3 AMB.İ190 ve Rulman için de yapay sinir ağları kullanılarak talep öngörüsü yapılarak irdelenmiştir. Çevrim dışı eğitmede kullanılarak gerçeklenen ÇKİB YSA için kullanılan başlangıç parametreler şunlardır:

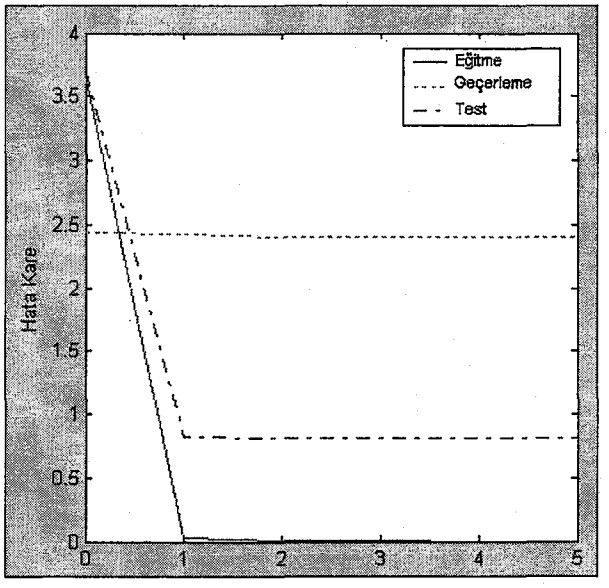
- *Yapay sinir ağının yapısı* : 1-30-1 nöron (bir saklı katmanlı)
- *Öğrenme tekniği* : Geriye Yayılım + Levenberg-Marquardt
- *Eğitme oranı* : 0.05
- *Örnek sayısı* : 34

Aşırı eğitilmeyi önlemek için eğitme kümesi verileri eğitme, geçerleme ve test verileri olarak (%50 - %25 - %25) oranında dörde bölünmüştür.



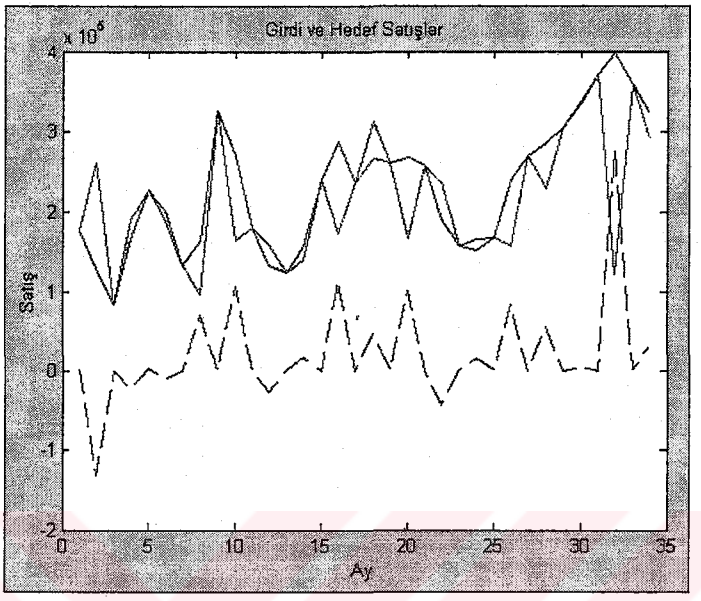
Şekil 10.2 : Eğitme, geçerleme ve test verileri sonuçları

Sinir ağının öğrenme eğrisinde (Şekil 10.2), eğitme periyodu ile çapraz geçerleme periyodu karşılaştırılmış ve ağın genelleme başarısı öngörülmeye çalışılmıştır.



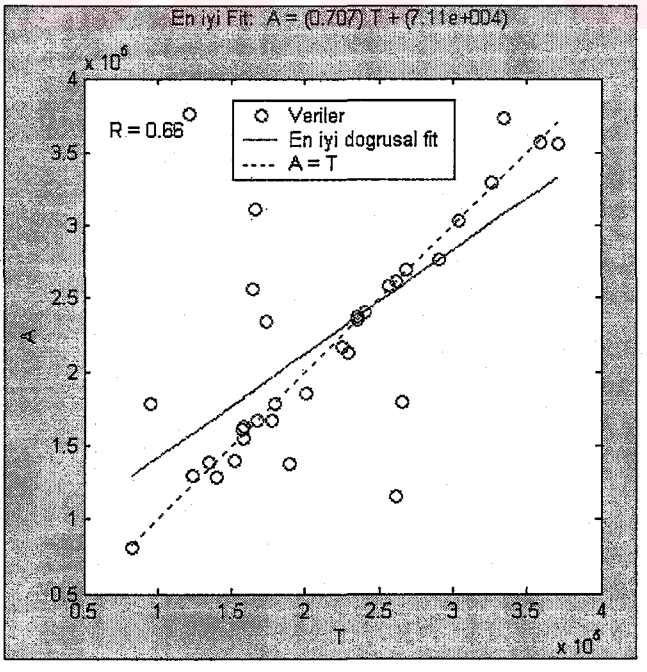
Şekil 10.3 : Eğitme ve çapraz geçerleme periyotları için etkin hata değerlerinin karşılaştırılması

Eğitme, çapraz geçerleme ve test periyotlarında hata kareleri ortalamalarının seviyesi Şekil 10.3'de görülmektedir. Geçerleme hatası artmaya başladığından eğitme işlemi 5 döngü sonunda kesilmiştir. Hesaplanan ve gerçekleşen satışlar ile tahmin hataları Şekil 10.4'de görülmektedir.



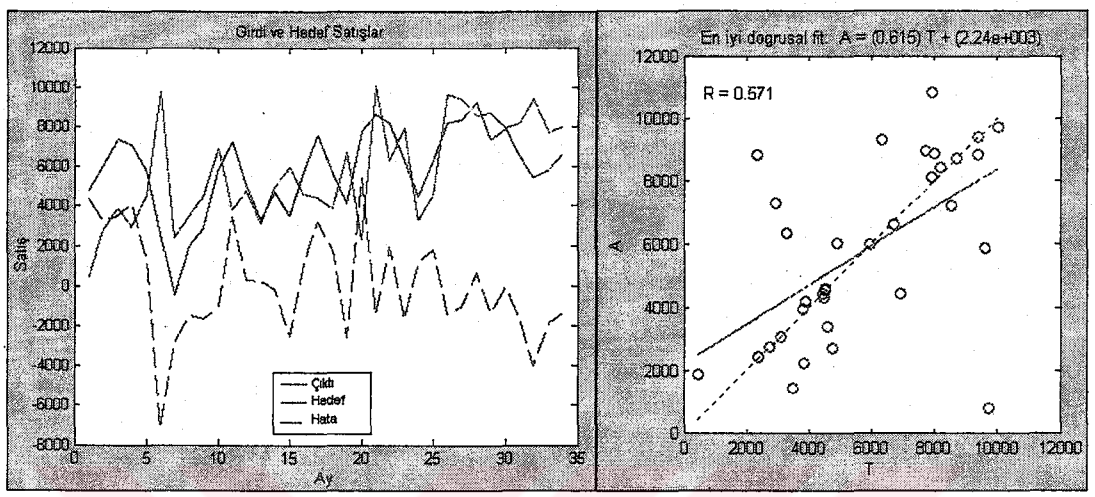
Şekil 10.4 : Hesaplanan ve gerçekleşen satışlar ile tahmin hataları

Bir sonraki adımda, yapay sinir ağları ile yapılan tahmin değeri ile gerçekleşen değer arasında doğrusal regresyon yapılmıştır (Şekil 10.5).



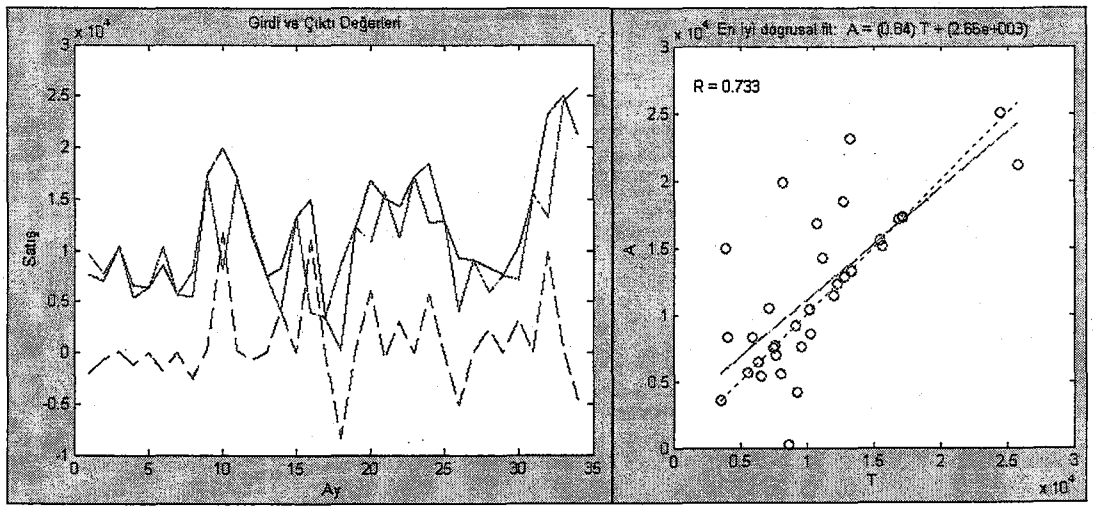
Şekil 10.5 : Tahmin ile gerçekleşen değerler arasındaki regresyon

Şekil 10.5'de R değerine bakıldığında (0.66), tahmin ve gerçekleşen satışlar arasındaki ilişkinin YSA ile güçlü bir şekilde modellenemediği söylenebilir. Daha farklı modellemeler üzerinde çalışıldığında da (birden fazla saklı katman, işlem elemanı sayısının artırılması gibi) benzer sonuçlar elde edilmiştir. Bu sonucun muhtemel nedeni elde bulunan veri sayısının çok kısıtlı olmasıdır.

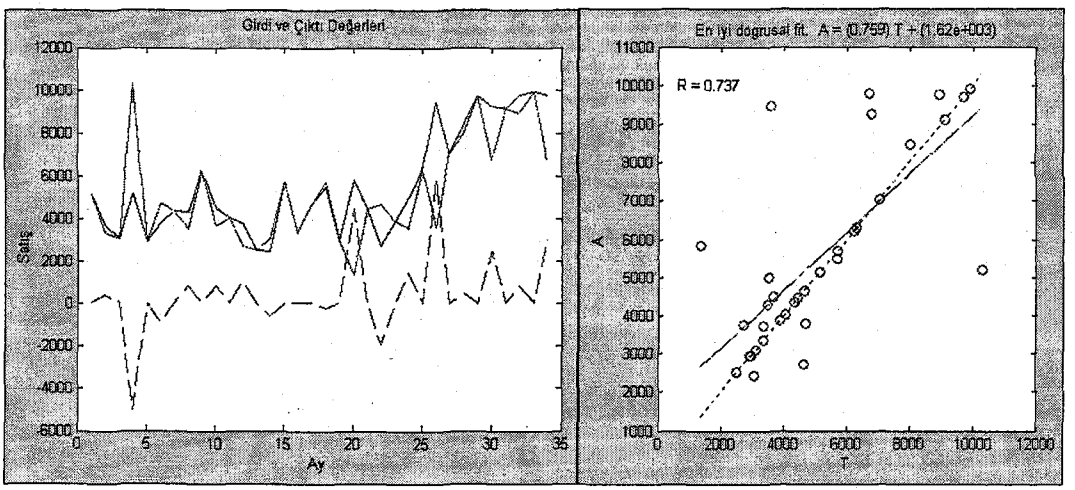


Şekil 10.6 : Kit-3 AMB.I180 için hesaplanan - gerçekleşen satışlar ile tahmin hataları ve regresyon sonuçları

KIT-3 AMB.İ180, KIT-3 AMB.İ190 ve Rulman için yapay sinir ağları kullanılarak yapılan talep öngörülerinin değerleri, öngörü hataları ve doğrusal regresyon sonuçları sırasıyla Şekil 10.6, Şekil 10.7 ve Şekil 10.8'de gösterilmiştir. Grafiklerden de görüldüğü gibi, bu ürünler için elde edilen sonuçlar incelendiğinde toplam aylık satış miktarları ile benzer sonuçlar bulunmuştur.

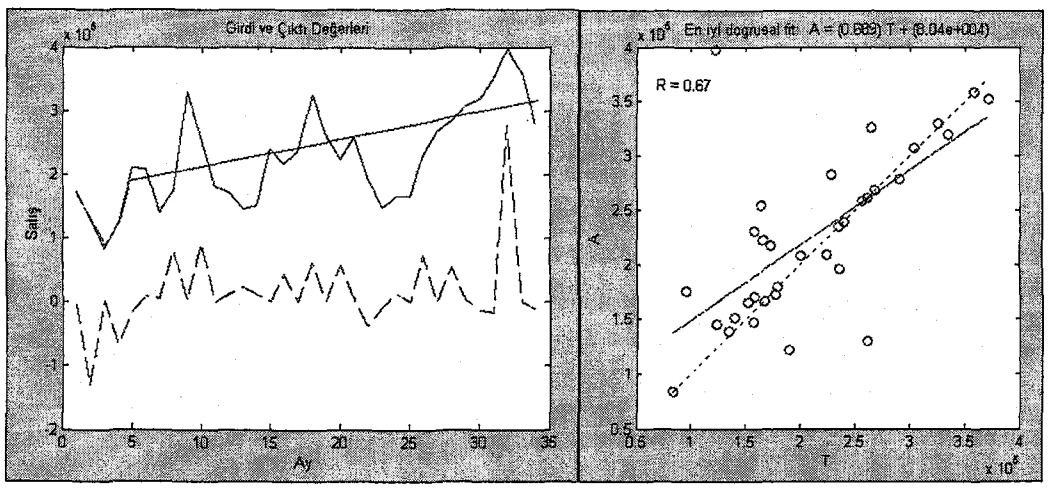


Şekil 10.7 : Kit-3 AMB.I190 için hesaplanan - gerçekleşen satışlar ile tahmin hataları ve regresyon sonuçları



Şekil 10.8 : Rulman için hesaplanan - gerçekleşen satışlar ile tahmin hataları ve regresyon sonuçları

Sistemin yakın geçmişteki birkaç örnekleme değeri ağa öğretilip bir ya da birkaç periyot sonraki örnekleme noktasında alacağı değeri kestirmek için çevrim içi eğitme kullanılarak toplam üretim miktarları için kestirim yapmak mümkündür. Özellikle gerçek zamanlı kontrol sistemlerinde veya zaman serileri analizinde kullanılan bu yaklaşımla oluşturulan YSA için elde edilen sonuçlar Şekil 10.9'da gösterilmiştir. YSA'nın yapısı daha önce kullanılan ağ ile aynı olmakla birlikte, hafızada tutulan geçmiş veri sayısı 3 olarak belirlenmiştir. Sonuçlar incelendiğinde, beklenenden daha iyi olarak çevrim dışı eğitme için elde edilen sonuçlara çok yakın olduğu görülebilir. Elde edilen en iyi hata değeri 0.078935 olarak bulunmuştur.



Şekil 10.9 : Çevrim içi eğitme ile hesaplanan - gerçekleşen satışlar ile tahmin hataları ve regresyon sonuçları

10.3.2 Elde edilen Test Sonuçlarının Değerlendirmesi

Geliştirilen GETTZY sisteminin performans ve ölçeklendirilebilirliğini ölçmek için farklı sayıda tedarikçi ile yerel ağ üzerinde yapılan testlerden elde edilen sonuçlar, sistemin ortalama 1 sn civarında bulunan işlem süreleri açısından yeteri kadar hızlı olduğunu göstermiştir. Ayrıca tedarikçi sayısının artmasına bağlı olarak işlem süresi beklendiği gibi uzasa da, sürenin üssel olarak artmadığı (doğrusal artış) gözlemlenmiştir. Bu da sistemin ölçeklendirilebilir olduğunun göstergesidir.

GETTZY sisteminde yer alan YSA modüllerinin gerçek bir problem üzerindeki etkinliğini ölçmek için Valeo A.Ş.'den temin edilen aylık satış bilgileri kullanılarak yapılan deneylerde ise, daha önce test verileri üzerinde elde edilen başarılı sonuçlara göre biraz daha kötü performans gösterdiği görülmüştür. Bunun olası nedeni ise temin edilen veri sayısının az olmasıdır. Günlük veya haftalık verinin bulunmayışı nedeniyle aylık veriler kullanılmıştır. Elde bulunan 34 aylık verinin %25'i geçerleme, %25'i test amaçlı olarak kullanılmış, ağı eğitmeye ise sadece 17 aylık veri kalmıştır.

Sonuç olarak, veri sayısının az olması YSA'nın performansını kötü yönde etkilemiştir. YSA'nın iyi bir modelleme ve genelleştirme yapabilmesi için daha fazla veri ile eğitilmesi gerektiği görülmüştür.

11. SONUÇ VE ÖNERİLER

11.1 Tezden Elde Edilen Sonuçlar

Bu tez çalışmasında, lojistik sistemlerinin modellenmesinde ve eniyilenmesinde yapay sinir ağlarının kullanım potansiyeli incelenmiş, çözüm kalitesini iyileştirici yöntemler geliştirilmiş, gerçek zamanlı bir sistemde kullanılabilecek gezgin etmen temelli ve sinir ağlarını kullanan bir karar destek sistemi tasarlanarak gerçeklenmiştir. Önerilen gezgin etmen temelli sistem, pek çok müşteri ve tedarikçilerin olduğu büyük ölçekli ve dinamik bir ortamda tedarik zinciri süreçleri için bir koordinasyon mekanizması oluşturan; talep tahmini, tedarikçilerin sınıflandırılması gibi işlemlerin yapay sinir ağı modelleri ile gerçekleyen bir çerçeve modeldir.

Bölüm 1'de, KEP'i çözmede geleneksel sezgisel yöntemlerle sinir ağlarının performansının karşılaştırmalı olarak değerlendirilmesi ihtiyacı ortaya konmuştur. Zira bu ihtiyaç, geliştirilecek sinir ağları diğer tekniklerle karşılaştırıldığında, YSA'nın avantajları doğru olarak kullanılıp pratik uygulamalarda uygulanacaksa kaçınılmazdır. Yapay sinir ağlarının donanım gerçeklemeleri, aynı uygulamanın ortam değişikliğine bağlı olarak tekrar tekrar çözülmesinin gerektiği endüstriyel uygulamalar için ideal olmasına karşın, problemin sadece bir kez çözülmesinin yeterli olduğu ve benzetime dayalı yöntemlerin kullanıldığı yöneylem araştırması problemleri için maliyeti yüksek bir çözümdür.

Yapay sinir ağlarının donanım gerçeklemelerine başlamadan önce, benzetimle elde edilen çözümün kalitesinin böyle bir kaynak yatırımına değecek nitelikte olup olmadığının belirlenmesi önemlidir. Bu nedenle, TZY sistemlerinin içerdiği süreçlerden talep tahmini, tedarikçi seçimi ve konum - tahsis problemleri YSA modelleri ile MatLab ve Java'da gerçeklenmiş ve tavlama benzetimi gibi sezgisel yöntemlerle çözüm kalitesi açısından karşılaştırılmıştır. Literatürde yer alan ve yaygın olarak karşılaştırma kıstası olarak kullanılan test verileri ile yapılan deneylerde, YSA diğer tekniklerle etkili bir biçimde yarışabilmiş, bir çok durumda da onlardan daha iyi sonuç vermiştir. Yapılan deneylerde hesaplama sürelerine ilişkin

istatistik ÇKİB YSA dışında tutulmamıştır. Zira bilgisayar benzetiminde sezgisel yöntemler YSA'dan daha iyi sonuç verirler. Oysa YSA'nın donanım uygulamaları gerçekleştirildiğinde bu dezavantajı karşılayacağından, çözüm kalitesi üzerinde durulmuştur.

Çok geniş uygulama alanı olan ve temelde fonksiyon yaklaştırma işlevine sahip olan çok katmanlı ileri beslemeli yapay sinir ağları Bölüm 2'de ele alınmıştır. Tek katmanlı sinir ağları sadece doğrusal ayırıcı iken, iki saklı katmanlı sinir ağları genel bir yaklaştırıcıdır. Başka bir deyişle, iki saklı katmanlı YSA karmaşık girdi-çıktı eşleştirmesini yapabilir.

Yapılan benzetimler sonucunda ÇKİB YSA'nın, içerisinde açıklanamayan veriler de içerebilen, genellikle kararsız bir yapı gösteren veya deseni istatistiksel tekniklerle kolaylıkla açıklanamayan talep tahminleri için başarıyla kullanılabileceği görülmüştür. Gerçek zamanlı talep tahmini sistemlerinde de TDL-ÇKİB YSA modelinin çevrim içi eğitilenler kadar olmasa da arzu edilen düzeyde iyi sonuçlar üretebildiği bulunmuştur. Ayrıca, eğitme zamanını azaltmaya yarayacak ve daha iyi performans göstermesini sağlayacak eğitme verilerinin standartlaştırılması, adım büyüklüğünün girdiye doğru artırılması, ağırlık sayısından daha fazla eğitme örüntüsüne sahip olunması gibi değişikliklerin kullanılması daha iyi sonuçların elde edilmesini sağlamıştır.

KEP'i çözmede kullanılan iki temel yaklaşım olan Hopfield-Tank sinir ağları ile Kohonen'in kendini düzenleyen ağlarının bazı kusurları vardır ve farklı problem tiplerinin çözümünü bulmada gösterdikleri başarı değişkendir. Bölüm 3 ve 4'de anlatıldığı gibi Kohonen ağları elastik bant teorisine dayandığından, Öklit olmayan problemlerde genelleştirilememe, Hopfield ağları ise genellikle olursuz çözüm üretme ve en iyi çözümü elde etmek için sıkıcı parametre ayarlamaları gerektirme gibi dezavantajlara sahiplerdir. Bu nedenle yapay sinir ağları ile elde edilen çözümlerin kalitesi, gerçek yaşamdaki süreçlere uygulanmasının önünde ciddi bir engel oluşturur. Son 10-15 yılda Hopfield ağlarının bilgisayardaki benzetimleri, bu tekniğin eniyileme tekniği olarak uygunluğu konusunda ciddi tartışma ve şüphe yaratmıştır. Bölüm 4'de değinildiği gibi, Hopfield ağının olurluluğunu sağlamaya yönelik bir çok çalışma yapılmıştır. Hopfield sinir ağının, Hopfield tarafından da anlatıldığı gibi, ağın ağırlıklarının simetrik olması durumunda enerji fonksiyonunun 0-1 yerel minimuma yakınsamasının garanti altına alındığı bu bölümde gösterilmiştir. Hopfield ağının olurluluğunun sağlanması için uygun enerji fonksiyonu gösterimi seçilerek, Gee tarafından sinir ağının benzetimi için önerilen yaklaşım uyarlanmak

suretiyle, eniyileme problemi için daha verimli bir benzetim algoritması kullanılmıştır (Gee, 1991). Ağın içsel dinamiklerinde değişiklik yaparak yerel minimumlardan kurtulmayı sağlayan ve daha iyi sonuç veren değişiklikler de önerilmiştir.

Modern lojistik, ulaştırma ve telekomünikasyon sistemlerinin büyük bir kısmında p-medyan ve merkez üs ağlarının kullanılması bu alandaki araştırmaların öneminin artmasında etken olmuştur. KT problemlerinden SÖKT problemini çözmek için yeni bir yaklaşım olarak önerdiğimiz *Titreşimli Genetik Algoritma* yöntemi klasik GA yaklaşımından ve Kohonen ağları ile yapılan çözümden daha iyi sonuç vermiş ve TGA 9.612 hesaplamadan sonra yakınsarken, GA TGA ile bulunandan daha kötü olan kendisine ait en iyi uygunluk değerine 72.012 yinelemeden sonra ulaşabilmiştir. SOM ise GA'dan daha kısa sürede daha iyi sonuç vermiştir. Merkez üslerin konumlandırılması problemini çözmek için Hopfield ağına dayanan bir çözüm yöntemi önerilmiş, test verilerinden elde edilen sonuçlar tavlama benzetimi çözümünden elde edilen sonuçlarla karşılaştırılmıştır. İyileştirilmiş (yerel minimumdan kurtulan) Hopfield ağı ile tavlama benzetimi bütün durumlar için en iyi çözümü bulurken normal Hopfield ağı iki durum için en iyi çözümü bulamamıştır. Bu sonuçlar iyileştirilmiş Hopfield ağının çözüm kalitesinin yüksek olduğunu göstermiştir.

Bölüm 9'da detaylı olarak açıklandığı gibi, yüksek düzeyde bütünleşik TZY sistemleri genellikle karmaşık içsel ve dışsal işlemleri içerirler. TZY'de çok farklı miktar ve çeşitte parça ve bileşenlerin satın alındığı bir çok tedarikçi, küresel olarak yayılmış olabilir. Rekabetçi konumda kalabilmek için etkin ve verimli olarak karmaşık işlemleri birbirine bağlayan bütünleşik TZY sistemi kurulabilir. Bu amacı gerçekleştirmede yazılım etmenleri, Internet üzerinden tedarik ve ürünlerin satışlarının da yer aldığı bir çok işin otomatikleştirilmesine yardımcı olarak, gereksinim duyulan ve tedarik edilen ürünlerle ilgili faaliyetleri, farklı ürünlerin alımının karşılaştırılması gibi problemleri azaltacak biçimde işlev görürler. Çok-etmenli sistemlerin üretim ve tedarik zinciri yönetimindeki uygulamaları yeni olmamakla birlikte, gezgin etmen kullanan koordinasyon mekanizması ne etmen temelli üretimde ne de tedarik zincirlerinde ele alınmamıştır. Bu nedenle, pek çok müşteri ve tedarikçinin olduğu büyük ölçekli ve dinamik bir ortamda, tedarik zinciri süreçlerini gerçeklemek için gezgin etmen temelli ve yapay sinir ağlarını kullanan bir çerçeve model tasarlanmış ve Java'da gerçeklenmiştir.

Son olarak gerçeklenen gezgin etmen temelli tedarik zinciri yönetim sistemi ve yapay sinir ağı modüllerinin başarımını görmek için, otomotiv endüstrisinde faaliyet

gösteren uluslar arası bir firmadan elde edilen veriler kullanılarak test edilmiştir. Doğal olarak sinir ağı modüllerinin benzetimi yapıldığından işlem süreleri uzun sürmüş, ancak GETTZY sisteminin iletişim süreleri YSA modüllerinin süreleri ile karşılaştırılamayacak kadar düşük olmuştur.

11.2 Geliştirme Önerileri

Bu tez çalışması, lojistik sistemlerin modellenmesinde ve eniyilenmesinde yapay sinir ağlarının çözüm kalitesi açısından geleneksel yöntemlerle (kesin veya sezgisel) etkili bir biçimde yarışabileceğini göstermiştir. Daha iyi sonuçlar alınabilmesi ise YSA'nın doğası gereği donanım uygulamalarının geliştirilmesine bağlıdır.

Merkez üslerin konumlandırılmasında başarıyla uygulanan Hopfield YSA modelinin, merkez üs arkları ve en büyük mesafenin en küçüklenmeye çalışıldığı merkez üs problemleri gibi güncel KTP'yi çözmede de kullanılabilir. Ancak, Hopfield ağı matris işlemlerinden dolayı çok fazla hesaplama zamanı ve hafızaya gereksinim duyduğundan, daha büyük modeller için test yapmak ancak donanım uygulaması ile mümkün olabilir. Gelecekte bu konuda yapılabilecek bir diğer çalışma da daha büyük modelleri çözebilecek paralel işlem yapabilen ağların tasarlanması ve donanım uygulamalarının gerçekleştirilmesidir.

Sürekli uzay problemleri için önerdiğimiz, klasik GA ve YSA'ya göre daha iyi sonuç veren TGA yaklaşımının ayrık uzay problemlerine de uygulanabilmesine dönük bir çalışma da yapılabilir.

KAYNAKLAR

Aaarts, E.H.L., and Korst, J., 1989. Simulated annealing and Boltzmann machines, John Wiley & Sons, Essex.

Abdinnour-Helm, S., 1998. A hybrid heuristic for the uncapacitated hub location problem, *European Journal of Operational Research*, **106**, 489-499.

Accenture, B.D., Accenture, J.M., and Accenture, B.Y., 1999. Technology in the Next Generation of Supply Chain Outsourcing - Leveraging Capabilities of Fouth Party Logistics, Ascet 1, http://www.ascet.com/documents.asp?d_ID=229.

Ackley, D.H., Hinton, G.F., and Sejnowski, T.J., 1985. A learning algorithm for Boltzman machines, *Cognitive Science*, **9**, 147-169.

Aiyer, S.V.B., Niranjan, M., and Fallside, F., 1990. A theoretical investigation into the performance of the Hopfield model, *IEEE Transactions on Neural Networks*, **1(2)**, 204-215.

Aiyer, S.V.B., 1991. Solving combinatorial optimization problems using neural networks, *Technical Report*, **CUED/F-INFENG/TR 89**, Cambridge University Engineering Department, 1991.

Akiyama, Y., Yamashita, A., Kajiura, M., and Aiso, H., 1989. Combinatorial Optimization with Gaussian Machines, *Proceedings IEEE International Joint Conference on Neural Networks*, **1**, 533-540.

Albus, J.S., 1975. New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, September, 220-27.

Amartur, S.C., Piraino, D., and Takefuji, Y., 1992. Optimization Neural Networks for the Segmentation of Magnetic Resonance Images, *IEEE Transactions on Medical Imaging*, **11(2)**, 215-220.

Anderberg, M.R., 1973. Cluster Analysis for Applications, Academic Press, Inc., New York.

Anderson, J.A., and Rosenfeld, E., 1988. Neurocomputing: Foundations of Research, The MIT Press, Cambridge, MA.

Anderson, J.A., Silverstein, J.W., Ritz, S.A., and Jones, R.S., 1977. Distinctive features, categorical perception and probability learning: Some applications of a neural model, *Psychological Review*, **84**, 413-451. Reprinted in Anderson and Rosenfeld (1988).

Anonymous, 1998. Cross docking: A common practice today, sure to grow tomorrow, *Modern Materials Handling*, **53(6)**, 19-21.

Appa, G., and Giannikos, I., 1994. Is Linear Programming Necessary for Single Facility Location with maximin of Rectlinear Distance?, *Journal of the Operational Research Society*, **45**, 97-107.

Astley, W.G., and van de Ven, A.H., 1983. Central Perspectives and Debates in Organizational Theory, *Administrative Science Quarterly*, **38**,245-273.

Aykin, T., 1994. Lagrangian Relaxation Based Approaches Hub-and-Spoke Network Design Problem, *European Journal of Operational Research*, **79**, 501-523.

Baker, J.E., 1987. Reducing Bias and Inefficiency in the Selection Algorithm, *Proceedings of the Second International Conference on Genetic Algorithms*, Morgan Kaufman Publishers, 14-21.

Balakrishnan, P.V., Cooper, M.C., Jacob, V.S., and Lewis, P.A., 1996. Comparative performance of the FSCL neural net and K-means algorithm for market segmentation, *European Journal of Operations Research*, **93**, 346-357.

Barbarosoğlu, G. ve Yazgaç, T., 1997. An application of the analytical hierarchy process to the supplier selection problem, *Production and Inventory Management Journal*, 1st quarter, 14-21.

Barbuceanu, M., and Fox, M., 1996. Capturing and modeling coordination knowledge for multi-agent systems, *International Journal of Cooperative Information Systems*, **5**, 2-3.

Barron, A., 1993. Universal approximation bounds for superpositions of sigmoid functions, *IEEE Transaction on Information Theory*, **39 (3)**, 930-945.

Bauer, H.U., and Pawelzik, K., 1992. Quantifying the neighborhood preservation of self-organizing feature maps, *IEEE Transactions on Neural Networks*, **3(4)**, -579.

Bauer, H.U., and Villmann, T., 1995. Growing a hypercubical output space in a self-organizing feature map, *Technical Report* **95-030**, International Computer Science Institute, Berkeley.

Bauknight, D.N., and Miller, J.R., 1999. Fourth Party Logistics: Evolution of Supply Chain Outsourcing, http://www.infochain.org/quarterly/Smr99/Fourth.html.

Baumann, J., Hohl, F., Rothermel, K., and Straber, M., 1998. Concepts of a Mobile Agent System, *The World Wide Web Journal*, **1(3)**, 123-137.

Berger, J., and Barkaoui, M., 2004. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows, *Computers and Operations Research*, **31**(12), 2037-2053.

Bishop, C., 1995. Neural Networks for Pattern Recognition, Oxford Press, Oxford.

Bishop, C.M., Svensén, M., and Williams, C.K.I, 1997. GTM: A principled alternative to the self-organizing map, in Mozer, M.C., Jordan, M.I. and Petsche, T., (eds.) *Advances in Neural Information Processing Systems* **9**, The MIT Press, Cambridge, MA, 354-360.

Blum, A., and Rivest, R.L. ,1989. Training a 3-node neural network is NP-complete, in Touretzky, D.S. (ed.), *Advances in Neural Information Processing Systems 1,* Morgan Kaufmann, San Mateo, CA, 494-501.

Bottou, L., and Bengio, Y., 1995. Convergence properties of the k-Means algorithms, in Tesauro, G., Touretzky, D., and Leen, T., (eds.) *Advances in Neural Information Processing Systems 7,* The MIT Press, Cambridge, MA, 585-592.

Boyson, S., Corsi, T., and Verbraeck, A., 2003. The e-supply chain portal: a core business model, *Transportation Research Part E*, **39**, 175-192.

Brown, M., and Harris, C., 1994. Neurofuzzy Adaptive Modelling and Control, Prentice Hall, New York.

Campbell, J.F., 1994. Integer Programming Formulations of Discrete Hub Location Problems, *European Journal of Operational Research*, **72**, 387-405.

Campbell, J.F., 1996. Hub Location and the p-Hub Median Problem, *Operations Research*, **44**, 923-935.

Campbell, J.F., Stiehr, G., Ernst, A.T., and Krishnamoorthy, M., 2003. Solving Hub Arc Location Problems on a Cluster of Workstations. Parallel Computing, **29**, 555-574.

Carpenter, G.A., and Grossberg, S., 1987a. A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics and Image Processing*, **37**, 54-115.

Carpenter, G.A., and Grossberg, S., 1987b. ART 2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics,* **26**, 4919-4930.

Carpenter, G.A., and Grossberg, S., 1990. ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures, *Neural Networks*, **3**, 129-152.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B., 1992. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on Neural Networks*, **3**, 698-713

Carpenter, G.A., Grossberg, S., and Reynolds, J.H., 1991. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, *Neural Networks*, **4**, 565-588.

Carpenter, G.A., Grossberg, S., and Rosen, D.B., 1991a. ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition, *Neural Networks*, **4**, 493-504.

Carpenter, G.A., Grossberg, S., and Rosen, D.B., 1991b. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Networks*, **4**, 759-771.

Carrizosa, E., Munoz, M., and Puerto, J., 1995. On the Set of Optimal Points to the Weighted maximin Problem, Studies in Locational Analysis, **7**, 21-33.

Cichocki, A., Unbehauen, R., Weinzierl, K. and Hölzel, R., 1996. A new neural network for solving linear programming problems, *European Journal of Operational Research* **93**, 244-256.

Chandra, C., Smirnov, A.V., and Sheremetov, L.B. 2001. Agent-Based Infrastructure for Management of consumer-Focused Smart Companies, *Proceedings of the IASTED-2001*, July 2001, Cancun, Mexico.

Chen, S., Cowan, C.F.N., and Grant, P.M., 1991. Orthogonal least squares learning for radial basis function networks, *IEEE Transactions on Neural Networks*, **2**, 302-309.

Choi, H.S., and Park, K.H., 1997. Shop-floor scheduling at ship building yards using the multiple intelligent agent system, *J. of Intelligent Manufacturing*, **8(6)**, 505-515.

Christopher, M., and Braithwaite, A., 1989. Managing Strategic Lead-times, *Logistics Information Management*, **2(4)**.

Christopher, M., 1992. Logistics and Supply Chain Management: Strategies for Reducing Cost and Improving Service, Pitman Publishing, Financial Times.

Chu, P., 1992. A neural network for solving optimization problems with linear equality constraints, *Proceedings IEEE International Joint Conference on Neural Networks*, **2**, 272-277.

Cichocki, A., and Unbehauen, R., 1993. Neural Networks for Optimization and Signal Processing, Wiley, New York.

Cohen, M.A., and Grossberg, S., 1983. Absolute stability of global pattern and parallel memory storage by competitive neural networks, *IEEE Transactions on Systems, Man and Cybernetics*, **13(5)**, 815-825.

Cugola, G., Ghezzi, C., Picco, G., and Vigna, G., 1997. Analyzing Mobile Code Languages, Mobile Object Systems, *Lecture Notes in Computer Science*, **1222**, Springer-Verlag, 94-109.

Dağlı, C., 1994. Artificial Neural Networks for Intelligent Manufacturing, Chapman & Hall, London.

Daskin, M.S., 1995. Network and Discrete Location: Models, Algorithms and Applications, John Wiley & Sons, Inc., New York.

Darr, T.P., and Birmingham, E.P.,1996. An attribute-space representation and algorithm for concurrent engineering, *AI EDAM*, **10(1)**, 21-35.

Deco, G., and Obradovic, D., 1996. An Information-Theoretic Approach to Neural Computing, NY: Springer-Verlag.

Demuth, H.B., and Beale, M.H., 1998. Neural Network Toolbox: User's Guide, The Math Works Inc., MA.

Desieno, D., 1988. Adding a conscience to competitive learning, *Proc. Int. Conf. on Neural Networks*, IEEE Press, **1**, 117-124.

Devroye, L., Györfi, L., and Lugosi, G., 1996. A Probabilistic Theory of Pattern Recognition, Springer-Verlag, New York.

Diamantaras, K.I., and Kung, S.Y., 1996. Principal Component Neural Networks: Theory and Applications, Wiley, New York.

Drezner, Z., and Wesolowsky, G., 1983. Minimax and maximin Facility Location Problems on a Sphere, *Naval Research Logistics Quarterly*, **30**, 305-312.

Drezner, Z., and Hamacher, H.W., 2002. Facility Location: Applications and Theory. Springer-Verlag, Berlin Heidelberg New York.

Ebery, J., 2001. Solving Large Single Allocation *p*-Hub Problems with Two or Three Hubs, *European Journal of Operational Research*, **128**, 447-458.

Elman, J.L., 1990. Finding structure in time, *Cognitive Science*, **14**, 179-211.

Erkut, E., and Öncü, T., 1991. A parametric 1-maximin Location Problem, *Journal of the Operational Research Society*, **42**, 49-55.

Ermiş M., Ülengin F., and Hacıoğlu A., 2002a. Vibrational Genetic Algorithm (VGA) For Solving Continuous Covering Location Problems, *Lecture Notes in Computer Science*, **2457**, 293-302.

Ermiş M., Hacıoğlu A., and Ülengin F., 2002b. A Real Coded Genetic Algorithm For A Class of Continuous Space Location-Allocation Problems,

Proceedings of International Conference on Fuzzy Systems & Soft Computational Intelligence, İstanbul, Türkiye, 29-31 May, 106-113.

Ermiş, M., Şahingöz, Ö., and Ülengin, F., 2004a. Mobile Agent Based Supply Chain Modeling with Neural Network Controlled Services, *International ICSC Symposium on Engineering of Intelligent Systems (EIS 2004)*, Madeira, Portugal.

Ermiş, M., Şahingöz, Ö., and Ülengin, F., 2004b. An Agent Based Supply Chain System with Neural Network Controlled Processes, *Lecture Notes in Computer Science*, **3314**, 837-846.

Ernst,A.T., and Krishnamoorthy, M., 1996. Efficient Algorithms for the Uncapacitated Single Allocation *p*-hub median problem, *Location Science*, **4**, 139-154.

Ernst, A.T., and Krishnamoorthy, M., 1998. Exact and Heuristic Algorithms for the Uncapacitated Multiple Allocation p-Hub Median Problem, *European Journal of Operational Research*, **104**, 100-112.

Ernst, A.T., and M. Krishnamoorthy, 1999. Solution algorithms for the capacitated single allocation hub location problem, *Annals of Operations Research*, **86**, 141-159.

Erwin, E., Obermayer, K., and Schulten, K., 1992. Self-organizing maps: Ordering, convergence properties and energy functions, *Biological Cybernetics*, **67**, 47-55.

Eshelman, L.J., and Schaffer, J.D., 1993. Real Coded Genetic Algorithms and Interval Schema, Foundations of Genetic Algorithms, Morgan Kaufman Publishers, 187-202.

Fahlman, S.E., 1989. Faster-Learning Variations on Back-Propagation: An Empirical Study, in **Touretzky, D., Hinton, G., and Sejnowski, T.,** eds., *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 38-51.

Fahlman, S.E., and Lebiere, C., 1990. The Cascade-Correlation Learning Architecture, in **Touretzky, D.S.** (ed.), *Advances in Neural Information Processing Systems 2,*, Morgan Kaufmann Publishers, Los Altos, CA, 524-532.

Fausett, L., 1994. Fundamentals of Neural Networks, Prentice Hall, Englewood Cliffs, NJ.

Faragó, A., and Lugosi, G., 1993. Strong Universal Consistency of Neural Network Classifiers, *IEEE Transactions on Information Theory*, **39**, 1146-1151.

Foo, Y.P.S., and Szu. H., 1989. Solving Large-Scale Optimization Problems by Divide-Conquer Neural Networks, *Proceedings IEEE International Joint Conference on Neural Networks*, **1**, 507-511.

Forgy, E.W., 1965. Cluster analysis of multivariate data: Efficiency versus interpretability, *Biometric Society Meetings*, Riverside, CA. Abstract in Biomatrics, **21**, 768.

Freeman, James A., and Skapura, David M., 1992. Neural Networks: Algorithms, Applications and Programming Techniques, Addison-Wesley, New York.

Fritzke, B., 1994a. Growing cell structures: A self-organizing network for unsupervised and supervised learning, *Neural Networks*, **7**, 1460.

Fritzke, B., 1994b. Fast learning with incremental RBF networks, *Neural Processing Letters*, **1**, 1-5.

Fritzke, B., 1995a. A growing neural gas network learns topologies, in G.Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA, 625-632.

Fritzke, B., 1995b. Incremental learning of local linear mappings, in F.Fogelman and P.Gallinari, editors, *ICANN'95: International Conference on Artificial Neural Networks*, Paris, France, 217-222.

Fritzke, B., 1997. The LBG-U method for vector quantization - an improvement over LBG inspired from neural networks, *Neural Processing Letters*, **5(1)**.

Fukushima, K., Miyake, S., and Ito, T., 1983. Neocognitron: A neural network model for a mechanism of visual pattern recognition, *IEEE Transactions on Systems, Man and Cybernetics*, **13**, 826-834.

Fukushima, K., 1988. Neocognitron: A hierarchical neural network capable of visual pattern recognition, *Neural Networks*, **1**, 119-130.

Garey ve Johnson, 1979. Computers and Intractibility: A Guide to the Theory of NP-Completeness, W.H.Freeman and Co., New York.

Gee, A.H., Aiyer, S.V., and Fallside, F., 1991. Neural Networks and Combinatorial Optimization Problems-The Key to a Successful Mapping, *Technical* Report, **CUED/F-INFENG/TR 77**, Cambridge University Engineering Department.

Gee, A.H., 1993. Problem Solving with Optimization Networks, *PhD Thesis*, Queen's College, Cambridge.

Gilman,C.R., Aparicio,M., Barry,J., Durniak,T., Lam,H., and Ramnath,R. 1997. Integration of design and manufacturing in a virtual enterprise using enterprise rules, intelligent agents, STEP and work flow, *SPIE Proceedings on Architectures, Networks and Intelligent Systems for Manufacturing Integration*, 160-171.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

Gray, R.M., 1984. Vector quantization, *IEEE ASSP Magazine*, **1**, -29.

Gray, R.M., 1992. Vector Quantization and Signal Compression, Kluwer Academic Press.

Gregory, R.E., 1986. Source selection: A matrix approach, *Journal of Purchasing and Materials Management*, summer, 24-29.

Grossberg, S., 1976. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors, *Biological Cybernetics*, **23**, 121-134.

Gue, K.R., 2001. Crossdocking Just-In-Time for Distribution, Graduate School of Business & Public Policy Naval Postgraduate School, Monterey, CA (http://web.nps.navy.mil/~krgue/Teaching/xdock-mba.pdf).

Hacıoğlu, A., and Özkol, İ., 2001. Modified BLX-α: Double Directional Alpha Method, *Proceedings of the Sixteenth International Symposium On Computer And Information Sciences* (*ISCIS XVI*), Antalya, Türkiye, 5-7 Kasım, 185-192.

Hadley, R.F., 1999. Cognition and the computational power of connectionist networks.

Hagan, M.T., and Menhaj, M., 1994. Training feedforward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks*, **5(6)**, 989-993.

Hagan, M.T., Demuth, H.B., and Beale, M.H., 1996. Neural Network Design, PWS Publishing, Boston.

Hartigan, J.A., 1975. Clustering Algorithms, Wiley, NY.

Hartigan, J.A., and Wong, M.A., 1979. Algorithm AS136: A k-means clustering algorithm, *Applied Statistics*, **28**, 100-108.

Hastie, T., and Stuetzle, W., 1989. Principal curves, *Journal of the American Statistical Association*, **84**, 502-516.

Haykin, S., 1995. Artificial Neural Networks: A Comprehensive Foundation, IEEE Press, New York.

Hecht-Nielsen, R., 1987. Counterpropagation networks, *Applied Optics*, **26**, 4979-4984.

Hecht-Nielsen, R., 1988. Applications of counterpropagation networks, *Neural Networks*, **1**, 131-139.

Hecht-Nielsen, R., 1990. Neurocomputing, Addison-Wesley, Reading, MA.

Hertz, J., Krogh, A., and Palmer, R., 1991. Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, CA.

Hohlfeld, M., and Yee, B., 2003. How to Migrate Agents, Available at http://www.cs.ucsd.edu/~bsy

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, **79**, 2554-2558.

Hopfield, J.J., 1984. Neurons with Graded Response have Collective Computational Properties Like Those of Two-State Neurons, *Proceedings National Academy Sciences*, **81**, 3088-3092.

Hopfield, J.J., and Tank, D.W., 1985. Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics*, **52**, 141-152.

Hornik, K., Stinchcombe, M., and White, H., 1989. MLPs are universal approximators, *Neural Networks*, **2**, 359-366.

Hornik, K., 1993. Some new results on neural network approximation, *Neural Networks*, **6**, 1069-1072.

Horst, R., Pardalos, P., and Thoai, N., 1995. Introduction to Global Optimization, *Kluwer*.

Hruschka, H., and Natter, M., 1990. Comparing performance of feedforward neural nets and K-means for cluster-based market segmentation, *European Journal of Operations Research*, **114**, 346-353.

Ismail, M.A., and Kamel, M.S., 1989. Multidimensional data clustering utilizing hybrid search strategies, *Pattern Recognition*, **22**, 75-89.

Jaiswal, N.K., 1997. Military Operations Research: Quantitative Decision Making, Kluwer Academic Publishers, Boston.

Jones, D.T., Hines, P., and Rich, N., 1997. Lean logistics, *International Journal of Physical Distribution & Logistics Management*, **27(3/4)**, 153-173.

Judd, J.S., 1990. Neural Network Design and the Complexity of Learning, The MIT Press, Cambridge, MA.

Kaihara, T., 2003. Multi-agent based supply chain modeling with dynamic environment, *Int. J. of Production Economics*, **85**, 263-269.

Kakeya, H., and Okabe, Y., 2000. Fast Combinatorial Optimization with Parallel Digital Computers, *IEEE Transactions on Neural Networks*, **11(6)**, 1323-1331.

Kasuba, T., 1993. Simplified Fuzzy ARTMAP, *AI Expert*, **8**, 18-25.

Kehoe ve Boughton, 2001a. Internet based supply chain management, *International Journal of Operations and Production Management*, **21(4)**.

Kehoe ve Boughton, 2001b. New paradigms in planning and control across manufacturing supply chains, the utilization of Internet technologies, *International Journal of Operations and Production Management*, **21(5/6)**.

Kirkpatrick, S., Jr., C.D.G., and Vecchi, M.P., 1983. Optimization by simulated annealing, *Science*, 220.

Klincewicz, J.G., 1991. Heuristics for the *p*-Hub Location Problem, *European Journal of Operational Research*, **79**, 25-37.

Kohonen, T., 1984. Self-Organization and Associative Memory, Springer, Berlin.

Kohonen, T., 1988. Learning Vector Quantization, *Neural Networks*, **1**, 303.

Kohonen, T., 1995/1997. Self-Organizing Maps, Springer, Berlin. First edition was 1995, second edition 1997.

Kosko, B., 1992. Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, NJ.

Kuo, R.J., Chen, C.H., and Hwang, Y.C., 2001. An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network, *Fuzzy Sets and Systems*, **118**, 21-45.

Kuo, R.J., Ho, L.M., and Hu, C.M., 2002. Cluster analysis in industrial market segmentation through artificial neural network, *Computers and Industrial Engineering*, **42(2-4)**, 391-399.

Kuo, R.J., Ho, L.M., and Hu, C.M., 2002. Integration of self-organizing feature map and K-means algorithm for market segmentation, *Computers and Operations Research*, **29(11)**, 1475-1493.

Kwok, A., and Norrie, D., 1993. Intelligent agent systems for manufacturing applications, *J. of Intelligent Manufacturing*, **4(4)**, 285-293.

Lang, K. J., Waibel, A. H., and Hinton, G., 1990. A time-delay neural network architecture for isolated word recognition, *Neural Networks*, **3**, 23-44.

Lange, D., and Oshima, M., 1998. Mobile Agents with Java: The Aglet API, In "Special issue on Distributed World Wide Web Processing: Applications and Techniques of Web Agents" Baltzer Science Publishers.

Lapedes, A.S., Steeg, E.W., and Farber, R.M., 1995. Use of adaptive networks to define highly predictable protein secondary-structure slasses, *Machine Learning*, **21(1-2)**, 103-124.

Lazano, F., Guerrero, F., Onieva, L., and Larraneta, J., 1998. Kohonen Maps for Solving A Class of Location-Allocation Problems, *European Journal of Operations Research*, **108**, 106-117.

Le Cun,Y., Denker, J., and Solla, S., 1993. Optimal brain damage, *Advances in Neural Information Processing Systems*, **2**, 598-605.

Le Gall, A., and Zissimopoulos, V., 1999. Extended Hopfield Models for Combinatorial Optimization, *IEEE Transactions on Neural Networks*, **10(1)**, 72-80.

Linde, Y., Buzo, A., and Gray, R.M., 1980. An algorithm for vector quantizer design, *IEEE Transactions on Communication*, **28**, -95.

Lippmann, R.P., 1987. An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, 4-22.

Liu, C.M., Kao, R.L., and Wang, A.H., 1994. Solving Location-Allocation Problems with Rectlinear Distances by Simulated Annealing, *Journal of the Operational Research Society*, **45**, 1304-1315.

Livneh, B., 1999. Neural Networks and Expert Systems for Process Control, KBE, (http://www.kbe.co.za/articles/nnets/nnets-benzi/nnets-in-control.htm).

Lloyd, S.P, 1957. Least squares quantization in pcm, *Technical Note*, Bell Laboratories, published in 1982 in *IEEE Transactions on Information Theory*.

Lo, J.T-H., 1992. A New Approach to Global Optimization and Its Applications to Neural Networks, *Proceedings IEEE International Joint Conference on Neural Networks*, **4**, 600-605.

Love, R.F., Morris, J.G., and Wesolowsky, G.O., 1988. Facility Location: Models and Methods, *Publications in OR*, **7**, North Holland, New York.

Lozano, S., Guerrero, F., Onieva, L., and Larraneta, 1998. Kohonen maps solving a class of location-allocation problems, *European Journal of Operations Research*, **108**, 106-117.

Lugosi, G., and Zeger, K., 1995. Nonparametric Estimation via Empirical Risk Minimization, *IEEE Transactions on Information Theory*, **41**, 677-678.

MacKay, D.J.C., 1992. Bayesian interpolation, *Neural Computation*, **4(3)**, 415-447.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations, **1**, *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, Berkeley, University of California Press, 281-297.

Magent, M.A., 1996. Combining Neural Networks and Tabu Search in a Fast Neural Network Simulation for Combinatorial Optimization, *PhD Thesis*, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, USA.

Mangiameli, P., Chen, S.K., and West, D., 1996. A comparison of SOM neural network and hierarchical clustring methods, *European Journal of Operations Research*, **93**, 402-417.

Maranzana, F.E., 1964. On the location of supply points to minimize transport costs, *Operational Research Quarterly*, **15**, 261-270.

Martinetz, T.M., 1993. Competitive Hebbian learning rule forms perfectly topology preserving maps, In *ICANN'93: International Conference on Artificial Neural Networks*, Amsterdam, Springer, 427-434.

Martinetz, T.M., Berkovich, T.M., and Schulten, K.J., 1993. Neural-gas network for vector quantization and its application to time-series prediction, *IEEE Transactions on Neural Networks*, **4(4)**, -569.

Martinetz, T.M., Ritter, H.J., and Schulten, K.J., 1989. 3D-neural-network for learning visuomotor-coordination of a robot arm, In *International Joint Conference on Neural Networks*, **4(4)**, 351-556.

Martinetz, T.M., and Schulten, K.J., 1991. A neural-gas network learns topologies, In T.Kohonen, K.Mäkisara, O.Simula and J.Kangas, editors, *Artificial Neural Networks*, 397-402.

Martinetz, T.M., and Schulten, K.J., 1994. Topology representing networks, *Neural Networks*, **7(3)**, -522.

Masters, T., 1993. Practical Neural Network Recipes in C++, Academic Press, San Diego.

Matsuda, S., 1998. "Optimal" Hopfield Network for Combinatorial Optimization with Linear Cost Function, *IEEE Transactions on Neural Networks*, **9(6)**, 1319-1330.

Medsker, L.R., and Jain, L.C., 2000. Recurrent Neural Networks: Design and Applications, CRC Press, Boca Raton, FL.

Mertz, P., 1994. Supply Chain Management: Report on the state-of-the-art, Center for Transportation Studies, MIT.

Michalewicz, Z, and Fogel , D.B., 2000. How to Solve It: Modern Heuristics, Springer-Verlag, Berlin.

Miller, J.G., and Vollmann, T.E., 1985. The hidden factory, *Harvard Business Review*, Sep-Oct.

Minsky, M.L., and Papert, S.A., (1969/1988), Perceptrons, MIT Press (first edition, 1969; expanded edition, 1988), Cambridge, MA.

Mitchell, M., (1998). An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA.

Moller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks*, **6**, 525-533.

Moller, C., 1995. Logistics Concept Development Towards a Theory for Designing Effective Systems, *PhD Thesis*, Department of Production, Aalborg University, Denmark.

Moody, J.E., and Darken, C.J., 1989. Fast learning in networks of locally-tuned processing units, *Neural Computation*, **1**, 281-294.

Moody, J.E., 1992. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems, *Advances in Neural Information Processing Systems*, **4**, 847-854.

Moore, B., 1988. ART 1 and Pattern Clustering, in **Touretzky, D., Hinton, G., and Sejnowski, T.,** eds., *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, CA, 174-185.

Moreno, J., Rodriguez, C., and Jimenez, N., 1991. Heuristic cluster algorithm for multiple facility location-allocation problem, *Recherche Operationnelle*, **25**, 97-107.

Mulier, F., and Cherkassky, V., 1995. Self-Organization as an Iterative Kernel Smoothing Process, *Neural Computation*, **7**, 1165-1177.

Nadaraya, E.A., 1964. On estimating regression, *Theory of Probability Application*, **10**, 186-90.

Nemhauser, G.L., and Wolsey, L.A., 1999. Integer and Combinatorial Optimization, John-Wiley & Sons, Inc.

O'Kelly, M., 1987. A Quadratic Integer Program for the Location of Interacting Hub Facilities, *European Journal of Operational Research*, **32**, 393-404.

O'Kelly, M., Skorin-Kapov, D., and Skorin-Kapov, J., 1995. Lower Bounds for the Hub Location Problem, *Management Science*, **41**, 713-721.

O'Kelly, M.E., and Bryan, D., 1998. Hub Location with Flow Economies of Scale, Transportation Research B, **32(8)**, 605-616.

Obaidat, M.S., 1998. Artificial neural networks to systems, man and cybernetics: characteristics, structures, and applications, *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, **28(4)**, 489-495.

Obayashi, S., Takanashi, S., and Takeguchi, Y., 1992. Niching and Elitist Model for MOGAs, Paralel Problem Solving from Nature-PPSN V, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 260-269.

Oja, E., 1989. Neural networks, principal components and subspaces, International *Journal of Neural Systems*, **1**, 61-68.

Okabe, A., Boots, B., Sugihara, K., 1992. Spatial Tesellations: Concepts and Applications of Voronoi Diagrams, Wiley, Chichester.

Oliver, N., 1990. JIT: Issues and Items for the Research Agenda, *International Journal of Physical Distribution & Materials Management*, **20(7)**.

Orponen, P., 2000. An overview of the computational power of recurrent neural networks, *Finnish AI Conference*, Helsinki.

Orr, M.J.L., 1996. Introduction to radial basis function networks, http://www.anc.ed.ac.uk/~mjo/papers/intro.ps

Pant, S., Sethi, R., and Bhandri, M., 2003. Making sense of the e-supply chain landscape: an implementation framework, *Int. J. of Information Management*, 23, 201-221.

Pao, Y. H., 1989. Adaptive Pattern Recognition and Neural Networks, Reading, Addison-Wesley, MA.

Papadimitriou, C.H., and Steiglitz, K., 1998. Combinatorial Optimization: Algorithms and Complexity, Dover Pub.Inc., N.Y.

Pauler, G., 1999. Development of a Neuro-Fuzzy approach for treating multimodal distributions and spurious clusters, *European Journal of Operations Research*, **112**, 207-219.

Persson, G., 1990. Organization Design Strategies for Business Logistics, *International Journal of Physical Distribution & Materials Management*, **8(6)**.

Peterson, C., and Anderson, J., 1989. Neural Networks and NP-Complete optimization problems: a performance study on the graph bisection problem, *Complex systems*, **2**, 59-89.

Piché, S.W., 1994. Steepest Descent Algorithms for Neural Network Controllers and Filters, *IEEE Transactions on Neural Networks*, **5(2)**, 198-212.

Pineda, F.J., 1989. Recurrent back-propagation and the dynamical approach to neural computation, *Neural Computation*, **1**, 161-172.

Podnar, H., Skorin-Kapov, J., and Skorin-Kapov, D., 1999. Network Cost Minimization Using Threshold Based Discounting, Working Paper.

Porter, M.E., 1990. The competitive advantage of nations, *Harvard Business Review*, March-April.

Preparata, F.P., and Shamos, M.I., 1990. Computational Geometry, Springer, New York, 1990.

Principe, J.C., Euliano, N.R., and Lefebvre, W.C., 2000. Neural and Adaptive Systems, *John Wiley & Sons*, New York.

Rajendran, C., and Ziegler, H., 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs, *European Journal of Operational Research*, **155(2)**, 426-438.

Reed, R.D., and Marks, R.J, 1999. Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, The MIT Press, Cambridge, MA.

Reeves, C.R., 1993. Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford.

Riedmiller, M., and Braun, H., 1993. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, IEEE, San Francisco.

Ripley, B.D., 1996. Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press.

Ritter, H., Martinentz, T., and Schulten, K., 1992. Neural computation and self-organizing maps, Addison-Wesley, Reading.

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, **65**, 386-408.

Rowat, C., 1998. Cross-docking: the move from supply to demand, *Distribution*, (http://www.dmg.co.uk/distribution/library/9804i.htm).

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., 1986. Learning internal representations by error propagation, in **Rumelhart, D.E., and McClelland, J. L.,** eds. (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* The MIT Press, Cambridge, MA, **1**, 318-362.

Sadeh, N., 1994. Micro-opportunistic scheduling: The MICRO-BOSS factory scheduler, *Technical Report*, **CMU-RI-TR-94-04**, Carnegie Mellon University, Pittsburg, PA.

Sanger, T.D., 1989. Optimal unsupervised learning in a single-layer linear feedforward neural network, *Neural Networks*, **2**, 459-473.

Sarkis, J., and Sundarraj, R.P., 2002. Evolution of brokering; paradigms in e-commerce enabled manufacturing, *Int. J. Production Economics*, **75**, 21-31.

Sarle, W.S., 1994. Neural Networks and Statistical Models, in SAS Institute Inc., *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 1538-1550. ftp://ftp.sas.com/pub/neural/neural1.ps.

Sarle, W.S., 1995. Stopped Training and Other Remedies for Overfitting, *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, 352-360.

Sarle, W.S., 1997. Neural Network FAQ, part 1 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets.

Sexton, R.S., Dorsey, R.E., and Johnson, J.D., 1999. Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing, *European Journal of Operations Research*, **114**, 589-601.

Sharda, R., and Wang, J., 1996. Neural Networks and Operations Research/ Management Science, *European Journal of Operations Research*, **93**, 227-229.

Siegelmann, H.T., 1998. Neural Networks and Analog Computation: Beyond the Turing Limit, Boston - Birkhauser.

Siegelmann, H.T., and Sontag, E.D., 1999. Turing Computability with Neural Networks, *Applied Mathematics Letters*, **4**, 77-80.

Sikora, R., and Shaw, M., 1998. A multi-agent framework for the coordination and integration of information systems, *Management Science*, **40(11)**, 65-78.

Sima, J., and Orponen, P., 2001. Computing with continuous-time Liapunov systems, *33rd ACM STOC*.

Simchi-Levi, D., Kaminsky, P., and Simchi-Levi, E., 2000. Designing and Managing the Supply Chain, McGraw-Hill, Singapore.

Skapura, David M., 1996. Building Neural Networks, Addison-Wesley, New York.

Skorin-Kapov, D., and Skorin-Kapov, J., 1994. On Tabu Search for the Location of Interacting Hub Facilities, *European Journal of Operational Research*, **73**, 502-509.

Smith, K.A., Palaniswami, M., and Krishnamoorthy, M., 1996a. A Hybrid Neural Approach to Combinatorial Optimization, *Computers & Operations Research*, **23**, 597-610.

Smith, K.A., Palaniswami, M., and Krishnamoorthy, M., 1996b. Traditional Heuristic versus Hopfield Neural Network Approaches to a Car Sequencing Problem, *European Journal of Operational Research*, **93**, 300-316.

Smith, K.A., Palaniswami, M., and Krishnamoorthy, M., 1998. Neural Techniques for Combinatorial Optimization with Applications, *IEEE Transactions on Neural Networks*, **9**, 1301-1318.

Smith, K.A., and Gupta, J.N.D., 2000. Neural networks in business: techniques and applications for the operations researcher, *Computers & Operations Research*, **27**, 1023-1044.

Solimanpur, M., Vrat, P., and Shankar, R., 2004. Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing, *European Journal of Operational Research*, **157(3)**, 592-606.

Sontag, E.D., 1992. Feedback stabilization using two-hidden-layer nets, *IEEE Transactions on Neural Networks*, **3**, 981-990.

Specht, D.F., 1990. Probabilistic neural networks, *Neural Networks*, **3**, 110-118.

Specht, D.F., 1991. A Generalized Regression Neural Network, *IEEE Transactions on Neural Networks*, **2**, 568-576.

Stamos, J.W., and Gifford, D.K., 1990. Remote Evaluation. *ACM Transaction on Programming Languages and Systems*, **14(4)**, 537-565.

Tagliarini, G.A., Fury, J.F., and Page, E.W., 1991. Optimization using Neural Networks, *IEEE Transactions on Computers*, **40(10)**, 1347-1358.

Takefuji, Y., 1992. Neural Network Parallel Computing, Kluwer Academic Publishers, Massachusetts.

Tam ve Tummala, 2001. An application of the AHP in vendor selection of a telecommunications system, *Omega*, **29**, 171-182.

Tank, D.W., and Hopfield, J.J., 1986. Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit and a Linear Programming Circuit, *IEEE Transactions on Circuit Systems*, **33(5)**, 533-541.

Taylor, D., 1997. Global Cases in Logistics and Supply Chain Management, ITP.

Teece, D.J., 1982. Towards an Economic Theory of the Multi product firm, *Journal of Economic Behavior and Organization*, **3(1)**, 39-44.

Teitz, M.B., and Bart, P., 1968. Heuristic methods for estimating generalized vertex median of a weighted graph, *Operations Research*, **16**, 955-961.

Timmerman, 1986. An approach to vendor performance evaluation, *Journal of Purchasing and Supply Management*, **1**, 27-32.

Topcuoglu, H., Corut, F., Ermis, M., and Yilmaz, G., 2005. Solving the uncapacitated hub location problem using genetic algorithms, *Computers and Operations Research*, **32 (4)**, 967-984.

Tsuchiya, K., Bharitkar, S., and Takefuji, Y., 1996. A neural network approach to facility layout problems, *European Journal of Operations Research*, **89**, 556-563.

Vaithyanathan, S., Burke, L.I., and Magent, M.A., 1996. Massively parallel analog tabu search using neural networks applied to simple plant location problems, *European Journal of Operations Research*, **93**, 317-330.

Valiant, L., 1988. Functionality in Neural Nets, *Learning and Knowledge Acquisition, Proc. AAAI*, 629-634.

Van den Bout, D.E., and Miller III, T.K., 1990. Graph partitioning using annealed neural networks, *IEEE Transactions on Neural Networks*, **1**, 129-139.

Vapnik, V., 1995. The Nature of Statistical Learning Theory, Springer Verlag, New York.

Vidyasagar, M., 1993. Location and Stability of the High-Gain Equilibria of Nonlinear Neural Networks, *IEEE Transactions on Neural Networks*, **4(4)**, 660-672.

Vukadinovic, K., Teodorovic, D., and Pavkovic, G., 1999. An application of neurofuzzy modeling: The vehicle assignment problem, *European Journal of Operations Research*, **114**, 474-488.

Wacholder, E., 1989. A Neural Network based optimization algorithm for the static weapon-target assignment problem, *ORSA Journal on Computing*, **4**, 232-245.

Walker, J.A., 1997. Neural nets and business, DS 495, (*http://weeks.ch.twsu.edu/Spring1997StudentPapers/JosephAWalker.htm*).

Walker, B., Popek, G., English, R., Kline, C., and Thiel, G., 1983. The LOCUS distributed operating system, *Proceedings Ninth Symposium on Operating Systems Principles*, Bretton Woods, NH, 49-70.

Wang, H.F., and Wu, K.Y., 2004. Hybrid genetic algorithm for optimization problems with permutation property, *Computers and Operations Research*, **31(14)**, 2453-2471.

Wang, S., 1996. A self-organizing approach to manegerial nonlinear discriminant analysis: A hybrid method of linear discriminant analysis and neural Networks, *INFORMS Journal on Computing*, **8**, 118-124.

Wang, Y., and Tan, K. L., 2002. A Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing, *World Wide Web: Internet and Web Information Systems*, **5**, 41–66.

Werbos, P.J., 1990. Backpropagtion through time: What it is and how to do it, *Proceedings of the IEEE*, **78**, 1550-1560.

White, H., 1990. Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings, *Neural Networks*, **3**, 535-550. Reprinted in White (1992b).

White, H., 1992b. Artificial Neural Networks: Approximation and Learning Theory, Blackwell.

White, H., and Gallant, A.R., 1992. On Learning the Derivatives of an Unknown Mapping with Multilayer Feedforward Networks, *Neural Networks*, **5**, 129-138. Reprinted in White (1992b).

White, J.E., 1994. Telescript Technology: The Foundation for the Electronic Marketplace, *White paper*, General Magic Inc., Mountain View, CA.

Widrow, B., and Hoff, M.E.Jr., 1960. Adaptive switching circuits, IRE WESCON Convention Record, part 4, 96-104. Reprinted in Anderson and Rosenfeld (1988).

Widrow, B., and Stearns, S., 1985. Adaptive Signal Processing, Prentice Hall.

Williams, R.J., and Zipser, D., 1989. A learning algorithm for continually running fully recurrent neurla networks, *Neural Computation*, **1**, 270-280.

Williamson, J.R., 1995. Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps, *Technical Report*, **CAS/CNS-95-003**, Boston University, Center of Adaptive Systems and Department of Cognitive and Neural Systems.

Wilppu, E., 1996. Controlling logistics with neural networks, *TUCS Technical Report* **79**.

Wilson, G.V., and Pawley, G.S., 1988. On the Stability of the TSP Algorithm of Hopfield and Tank, *Biological Cybernetics*, **58**, 63-70.

Womac, J.P., Jones, D.T., and Ross, D., 1990. The Machine that Changed the World, Rawson Associates.

Wong, B.K., Lai, V.S., and Lam, J., 2000. A bibliography of neural network business applications research: 1994-1998, *Computers and Operations Research*, **27**, 1045-1076.

Wooldridge, M., Bussmann,S., and Klosterberg, M., 1996. Production sequencing and negotiation, *PAAM-96*, London, UK.

Wu, D.S., and Harker, P.T., 1999. Toward a distributed paradigm for manufacturing logistics, (*http://www.lehigh.edu/~sdw1/nsf99.html*).

Wu, D.J., 2001. Software agents for knowledge management: coordination in multi-agent supply chains and auctions, *Expert Systems with Applications*, **20**, 51-64.

Yalçınöz, T., Cory, B.J., and Short, M.J., 2001. Hopfield neural network approaches to economic dispatch problems, *Electrical Power and Energy Systems*, **23**, 435-442.

Yan, Y., Yen, J., and Bui, T., 2000. A multi-agent based negotiation support system for distributed transmission cost allocation, *HICSS-33*.

Yung, S., and Yang, C., 1999. A new approach to solve supply chain management problem by integrating multi-agent technology and constraint network, *HICS-32*.

Zador, P.L., 1982. Asymptotic quantization error of continuous signals and the quantization dimension, *IEEE Transactions on Information Theory*, **28**, 139-149.

Zeger, K., Vaisey, J., and Gersho, A., 1992. Globally optimal vector quantizer design by stochastic relaxation, *IEEE Transactions on Signal Processing*, **40**, 310-322.

EK A

GERİYE YAYILIMIN TÜRETİMİ

İlk olarak k'ncı sinirin ağdaki tek çıktı siniri olduğu varsayılır ve i'nci İE'nin yerel hatası δ olarak tanımlanırsa;

$$\delta_i = \frac{\partial J}{\partial y_i} f'(net_i) \tag{A.1}$$

Zincir kuralı uygulandığında, maliyetin ağırlığa göre türevi;

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}} = -\delta_i y_j \tag{A. 2}$$

Görüldüğü gibi ağırlıklara göre maliyetin duyarlılığı, maliyetin y_i durumuna göre duyarlılığı ile durumun yerel ağırlıklara olan duyarlılığının çarpımına eşittir. Durumun duyarlılığı ($\partial J/\partial y$) zincir kuralı ile bulunabilir.

$$\frac{\partial J}{\partial y_i} = \frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial y_i} = \frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial net_k}\frac{\partial net_k}{\partial y_i} \tag{A.3}$$

Bulunan bu kısmi türevleri ağdaki niceliklerle ifade eder ve A.2 denkleminde yerine koyulursa;

$$\frac{\partial J}{\partial w_{ij}} = -\varepsilon_k f'(net_k) w_{ki} f'(net_i) y_j \tag{A.4}$$

Bu denklem maliyetin ağırlığa (w_{ij}) göre olan eğimini hesaplar. Birden fazla İE için de benzer yol izlenir. Tek fark, artık i'nci İE'yle ilişkilendirilmiş birden fazla çıktı İE'nin (k ile gösterilen) olmasıdır. Yani;

$$\frac{\partial J}{\partial y_i} = \left(\sum_k \frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial y_i} \right) = \sum_k \varepsilon_k f'(net_k) w_{ki} \qquad \text{(A.5)}$$

Denklem A.5'i A.2'de yerine koyarsak, sonuçta aşağıdaki eşitlik elde edilir.

$$\frac{\partial J}{\partial w_{ij}} = \left[\sum_k \varepsilon_k f'(net_k) w_{ki} \right] f'(net_i)(-y_j) \qquad \text{(A.6)}$$

Eğim inerek öğrenmenin kuralları gereği, ağırlıklar Denklem A.6'nın negatifi ile orantılı olarak ayarlanır. Böylelikle GY ile ağırlık güncelleme denklemi elde edilir:

$$w_{ij}(n+1) = w_{ij}(n) + \eta f'(net_i(n)) \left(\sum_k \varepsilon_k(n) f'(net_k(n)) w_{ki}(n) \right) y_j(n) \qquad \text{(A.7)}$$

EK B

HOPFIELD SİNİR AĞINDA ENERJİ FONKSİYONUNUN YAKINSAMASI

Sürekli Hopfield sinir ağı için aşağıdaki enerji fonksiyonunu ele alınsın:

$$E_s = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} g_i(u_i) W_{ij} g_j(u_j) - \sum_{i=1}^{n} x_i g_i(u_i) + \sum_{j=1}^{y_i}\int_0 g_i^{-1}(y)dy \qquad \text{(B.1)}$$

W matrisi simetrik olarak verildiğinde E_s'nin zamana göre türevi:

$$\frac{dE_s}{dt} = -\sum_{i=1}^{n}\frac{dy_i}{dt}\left(\sum_{j=1}^{n} W_{ij} y_j + x_i - u_i\right)$$

$$= -\sum_{i=1}^{n} C_i \left(\frac{dy_i}{dt}\right)\left(\frac{du_i}{dt}\right)$$

Tanım gereği $u_i = g_i^{-1}(y_i)$ olduğundan,

$$= -\sum_{i=1}^{n} C_i g_i^{-1'}(y_i)\left(\frac{dy_i}{dt}\right)^2$$

$g_i^{-1}(y_i)$ fonksiyonu monoton olarak artan ve C_i pozitif olduğundan,

$$\frac{dE_s}{dt} \le 0, \qquad ve \qquad \frac{dE_s}{dt} = 0 \qquad \Rightarrow \qquad \frac{dy_i}{dt} = 0 \qquad \forall i \qquad \text{(B.2)}$$

E_s sınırlı olduğundan (B.2) denklemi, 4.3 diferansiyel denkleminin kontrolü altında E_s'nin bir minimuma yakınsayacak şekilde azalarak orada kalacağını gösterir.

HOPFIELD SİNİR AĞINDA OLURLU ÇÖZÜM VEREN ENERJİ FONKSİYONU

Bir $\mathbf{x}$ vektörü kısıt düzleminin ($\mathbf{Ax} = \mathbf{b}$) çözüm uzayına projeksiyonu yapılırsa:

$$\mathbf{x} \leftarrow \wp.\mathbf{x}+\mathbf{s} \tag{C.1}$$

Öyleki; $$\wp = \mathbf{I}-\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \tag{C.2}$$

$$\mathbf{s} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \tag{C.3}$$

Bu yüzden, $\mathbf{x}$ vektörü kısıt düzleminden sapma miktarı aşağıdaki gibi bulunabilir:

$$\left\|\mathbf{x}-(\wp.\mathbf{x}+\mathbf{s})\right\|^2 \tag{C.4}$$

Enerji fonksiyonu; $$E(\mathbf{x}) = f(\mathbf{x})+\frac{\gamma}{2}C^g(\mathbf{x})$$

$$E(\mathbf{x}) = \mathbf{c}^T\mathbf{x}+\mathbf{x}^T\mathbf{Q}\mathbf{x}+\frac{\gamma}{2}\left\|(\wp.\mathbf{x}+\mathbf{s})-\mathbf{x}\right\|^2$$

$$= -\frac{1}{2}\mathbf{x}^T\left[-2\mathbf{Q}+\gamma(\wp-\mathbf{I})\right]\mathbf{x}-\mathbf{x}^T\left[\gamma\mathbf{s}-\mathbf{c}\right]+\frac{\gamma}{2}\mathbf{s}^T\mathbf{s} \tag{C.5}$$

$E(\mathbf{x})$ standart enerji fonksiyonu ile karşılaştırıldığında ($E_a = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq 1}^{n}W_{ij}y_iy_j - \sum_{i=1}^{n}I_iy_i$)

ağın ağırlıkları ($\mathbf{W}$) ile harici girdilerinin ($\mathbf{I}$) neye eşit olduğu anlaşılabilir. W_{ii}, ikinci dereceden formun ($\mathbf{Q}$), kısıt düzleminin yapısına ve γ değerine bağlı olarak pozitif, negatif veya sıfır olabilir. Negatif değerler dışbükey bir enerji fonksiyonuna yol açarak tamsayı olmayan bir olurlu çözüme yakınsayabileceğinden, çözümün bir tepe noktasına doğru gitmesini sağlamak amacıyla enerji fonksiyonuna başka bir terim ($\delta\mathbf{x}^T(\mathbf{1}-\mathbf{x})$) daha eklenmiştir (Smith, 1996a). Yeni enerji fonksiyonu:

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\left[-2\mathbf{Q}+2\delta\mathbf{I}+\gamma(\wp-\mathbf{I})\right]\mathbf{x}-\mathbf{x}^T\left[\gamma\mathbf{s}-\mathbf{c}-\delta\mathbf{1}\right]+\frac{\gamma}{2}\mathbf{s}^T\mathbf{s} \tag{C.6}$$

EK D

p-MEDYAN KONUM-TAHSİS PROBLEMLERİ İÇİN ÖRNEK VERİ

	x-koordinatı	y-koordinatı	Talep oranı
1	0,804348	0,717391	0,007951
2	0,507246	0,835749	0,010326
3	0,615942	0,369565	0,010998
4	0,797101	0,811594	0,011411
5	0,980676	0,785024	0,011514
6	0,567633	0,698068	0,012133
7	0,811594	0,543478	0,012185
8	0,480676	0,666667	0,012650
9	0,391304	0,439614	0,012701
10	0,603865	0,763285	0,012908
11	0,896135	0,570048	0,013114
12	0,657005	0,243961	0,013682
13	0,135266	0,359903	0,015489
14	0,485507	0,927536	0,016006
15	0,799517	0,101449	0,016212
16	0,217391	0,835749	0,016522
17	0,874396	0,386473	0,016574
18	0,777778	0,867150	0,017038
19	0,330918	0,876812	0,017038
20	0,294686	0,809179	0,012908
21	0,490338	0,618357	0,017245
22	0,557971	0,574879	0,018587
23	0,731884	0,601449	0,018587
24	0,910628	0,260870	0,018949
25	0,599034	0,094203	0,018949
26	0,345411	0,357488	0,018949
27	0,939614	0,705314	0,019000
28	0,990338	0,297101	0,019052
29	0,615942	0,516908	0,021737
30	0,352657	0,161836	0,022460
31	0,214976	0,676328	0,022511
32	0,710145	0,799517	0,022821
33	0,524155	0,246377	0,023596
34	0,695652	0,169082	0,023596
35	0,253623	0,939614	0,023751
36	0,623188	0,618357	0,024164
37	0,874396	0,852657	0,024267
38	1,045894	0,524155	0,024525
39	0,408213	0,062802	0,024732
40	0,693237	0,939614	0,026332
41	1,009662	0,429952	0,026900
42	1,159420	0,661836	0,027984
43	1,086957	0,724638	0,028242
44	0,751208	0,927536	0,028397
45	0,842995	0,915459	0,028397
46	1,275362	0,763285	0,030308
47	1,234300	0,693237	0,040634
48	0,439614	0,212560	0,040737
49	0,601449	0,567633	0,045229

ÖZGEÇMİŞ

1986 yılında Ankara Fen Lisesi'nden mezun olan Murat ERMİŞ, lisans eğitimini Boğaziçi Üniversitesi Endüstri Mühendisliği Bölümünde 1991 yılında tamamlamıştır. Mezuniyetinin akabinde Kayseri Hava İkmal ve Bakım Merkezi Komutanlığında İmalat Üretim Planlama ve Kontrol Subayı olarak göreve başlamıştır. Orta Doğu Teknik Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'ndaki yüksek lisans öğrenimini 1997 yılında tamamlamış ve aynı yıl Hava Harp Okulu'na atanmıştır. 1998 yılında İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda doktora öğrenimine başlayan Murat ERMİŞ, halen Hava Harp Okulu Endüstri Mühendisliği Bölümü'nde öğretim görevlisi olarak görev yapmaktadır.