İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TAŞITIN YANAL VE DOĞRUSAL KONTROLÜ İÇİN SÜRÜCÜNÜN MODELLENMESİ

YÜKSEK LİSANS TEZİ

Mak. Müh. İ. İlker DELİCE
(503021600)

Tezin Enstitüye Verildiği Tarih : 9 Mayıs 2005
Tezin Savunulduğu Tarih : 31 Mayıs 2005

Tez Danışmanı : Y.Doç.Dr. Şeniz ERTUĞRUL
Diğer Jüri Üyeleri Prof.Dr. Ahmet KUZUCU (İ.T.Ü.)
Prof.Dr. Serhat ŞEKER (İ.T.Ü.)

MAYIS 2005

ÖNSÖZ

Lisans eğitimim sırasında bana Otomatik Kontrol'ü sevdiren ve çalışmamda danışmanlık yapan Sayın Hocam Y. Doç. Dr. Şeniz Ertuğrul'a teşekkür ederim. Değerli bilgilerini ve sürücü verilerini bizimle paylaşan Dr. Orhan Atabay'a da en içten şükran duygularımı sunarım.

Ayrıca TÜBİTAK-Münir Birsel Vakfı'na ve son olarak da manevi desteklerini esirgemeyen Sistem Dinamiği ve Kontrol yüksek lisans programından dönem arkadaşlarım İlker Çardaklı, Erdal Genç, Kılıç Özen ve Atilla Kılıçarslan'a teşekkürlerimi sunmak isterim.

Mayıs 2005 İsmail İlker DELİCE

İÇİNDEKİLER

KISALTMALAR

PRBS : Pseudo Random Binary Sequence
FIR : Sonlu Darbe Cevabı (Finite Impulse Response)
ARX : Auto Regressive with eXogenous input
ARMAX : Auto Regressive Moving Average with eXogenous input
YU : Yüzde Uyum
YSA : Yapay Sinir Ağı
LM : Levenberg-Marquardt
SKNS : Saklı Katmandaki Nöron Sayısı
Logsig : Logaritmik Sigmoid Etkinlik Fonksiyonu
Tansig : Tanjant Sigmoid Etkinlik Fonksiyonu

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

SEMBOL LİSTESİ

n_A	: Çıkışın model mertebesi
n_B	: Girişin model mertebesi
n_k	: Girişin gecikme mertebesi
r_u	: Giriş için korelasyon fonksiyonu
$Q(s)$	: Lipschitz sayısı ($s = n_A + n_B$)
σ	: Standart sapma
φ	: Regresyon vektörü
θ	: Sütün parametre vektörünü,
ε	: Model hataları (kalıntılar)
I	: Yapay nöronun girişi
Φ	: Etkinlik veya aktivasyon fonksiyonu
α	: Aktivasyon fonksiyonlarının eğim parametresi
ε^2	: Karesel maliyet fonksiyonu
$\mathbf{J}$	: Hata vektörünün Jakobiani
$\mathbf{g}$	: Maliyet Fonksiyonunun Gradyeni
$\mathbf{H}$	: Hessian Matrisi
δ	: Hata işaret terimi
$w_{hp.j}$	: Saklı katman ağırlık terimi
$w_{pq.k}$	: Çıkış katmanı ağırlık terimi
η_{hp}	: Saklı katmandaki ağırlıkların öğrenme oranı
η_{pq}	: Çıkış katmandaki ağırlıkların öğrenme oranı
μ	: Momentum terimi

TAŞITIN YANAL VE DOĞRUSAL KONTROLÜ İÇİN SÜRÜCÜNÜN MODELLENMESİ

ÖZET

Klasik ve akıllı kontrol yöntemlerindeki bütün gelişmelere rağmen, birçok uygulamanın karmaşık ve belirsiz olması nedeni ile insan operatörlerin yerine otomatik kontrol sistemleri hala kullanılamamaktadır. Araba kullanma, uçağın kumandası veya karmaşık bir prosesin elle kontrolü bu uygulamalara örnek gösterilebilir. Genellikle insan operatör görsel geri besleme bilgisini kullanarak makina ile etkileşim halindedir. Bu görsel bilgiye dayanarak operatör yapacağı eylemin tipine ve miktarına karar verir ve böylece kapalı çevrimi oluşturur.

Motor ya da taşıtın gelişimiyle uğraşan mühendisler bunları açık çevrim olarak simüle etmişlerdir. Ancak, bir kontrolör olan sürücü olmadan simülasyonlardan en iyi performansın elde edilmesi mümkün değildir. Taşıt sürücüsünün gerçeğe en yakın şekilde modellenmesi ile kapalı çevrim çalışan başarılı simülasyonlar yapılabilir.

Bu tez çalışmasının amacı, doğrusal ve yanal kontrolü bir arada gerçekleştiren bir sürücü modeli çıkarmaktır. Sürücünün taşıtla etkileşiminin karmaşıklığından, insan operatörlerin iç rasgeleliğinden ve yanlılığından ötürü modelleme zorlaşmakta ve bu durum, çalışmayı, verilerin içindeki bilginin kullanılmasına dayalı sistem tanılama yöntemlerine götürmektedir. Çalışmalara ilk olarak doğrusal parametrik model olan ARX (Auto Regressive with eXogenous input) ile başlanmış ve daha iyi performans vereceği düşünülerek doğrusal olmayan modeller (YSA Yapay Sinir Ağları) de denenmiştir. Uygun giriş-çıkış bileşenleri ve uygun ağ mimarisi seçilerek yapılan denemelerde YSA modelde gaz pedalı için %82.5, fren için %83.5, vites için %81.7 ve direksiyon açısı için %93.3 başarım elde edilmiştir.

Çalışma, taşıt sürücüsü modeline olan ihtiyacın açıklanması ile başlamaktadır ve konuyla ilgili literatürde yer alan, bu çalışmaya esin kaynağı olan yayınların anlatılması ile devam etmektedir. Bölüm 2'de ise detaylı bir şekilde sistem tanılamanın tüm aşamaları anlatılmaktadır. Bölüm 3 ve 4'te ise klasik ve akıllı yöntemler ile modelleme hakkında teorik bilgiler verilmektedir. Bölüm 5 ise daha önceki bölümlerde anlatılan bilgiler kullanılarak taşıt sürücüsün modellenmesi için yapılan çalışmaları içermektedir. Son bölümde de çıkarılan sonuçlar tartışılmıştır.

HUMAN DRIVER MODELING FOR LATERAL AND LONGITUDINAL CONTROL OF A VEHICLE

SUMMARY

Despite many classical and intelligent control methods, it is still very difficult to replace human operators in many applications due to complexity, uncertainty and vagueness. Driving a car, piloting an airplane or manually controlling a complicated process can be given as examples for such applications. Generally a human operator interacts with a machine by using the visual data as feedback. Based on the visual information, the operator decides on the type and the amount of action to be taken, hence closing the loop between the inputs and the outputs of the plant.

Engine and vehicle have been simulated as open loop models by engineers that have been working on engine and vehicle development. But without having the human driver as a controller it is not possible to constitute realistic simulations. Unless an accurate model of the driver exists, perfect closed loop simulations can not be accomplished.

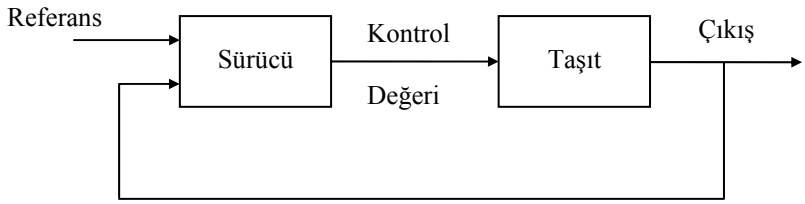
Main objective of this study is obtaining a human driver model containing both lateral and longitudinal control of a vehicle. Human driver-vehicle interactions, human operators' randomness and bias make the modeling difficult and system identification approach seems to be the only choice for modeling. Firstly, human driver was modeled using a linear parametric model structure, namely ARX (Auto Regressive with eXogenous input), then nonlinear model structures based on Neural Networks are also tried. Neural Network model outputs fit the real data much more satisfactorily. For the best results, fitting percentages for accelerator pedal, brake, gear, steering wheel angle are 82.5%, 83.5%, 81.7%, 93.3% respectively.

Study begins with explaining the need for a human driver model. Next, the literature survey is summarized. In Section 2, system identification procedures that are applied to obtain the human driver model are mentioned. Section 3 contains the information on classical and intelligent modeling. Studies on modeling the human driver based on this information are given in Section 5. Finally, modeling results are discussed in Section 6.

1. GİRİŞ

1.1 Çalışmanın Amacı

Bu proje ile amaçlanan, bir kontrolör olarak taşıt sürücüsünün modellenmesi ve taşıtla birlikte çalışan kapalı bir çevrim (kontrollü bir çevrim) oluşturulmasıdır (Şekil 1.1). Böylece bilgisayar ortamında değiştirilen çeşitli motor parametrelerinin sürüş yapılmadan gerçeğe en yakın kapalı çevrim simülasyonunu yapmak mümkün olabilecektir.



Şekil 1.1 : Sürüş Kapalı Çevrimi

Sürücünün taşıtla etkileşimi çok karmaşık bir görevdir. Bu yüzden motor ya da taşıtın gelişimiyle uğraşan mühendisler bunları açık çevrim olarak simüle etmişlerdir. Ancak, kontrolör yani sürücü olmadan simülasyonlardan en iyi performansın elde edilmesi mümkün değildir.

Mitschke (1996)'da taşıtı kontrol eden sürücünün iki önemli görevinden bahsetmektedir. Bunlar;

- Doğrusal kontrol (Yol doğrultusu boyunca)
- Yanal kontrol

Doğrusal kontrolde gaz pedalı ve fren pedalının konumuna karar verilir, uygun vitesler seçilir. Karar verme işi, öndeki araca olan mesafe ve yaklaşma hızı ile ilişkilidir. Yanal kontrolde ise direksiyon açısına karar verilir. Yolun şekli ve planlanan yol çizgisi taşıtın yanal kontrolüne etki eden bileşenlerdir **(Atabay, 2004)**.

Doğrusal ve yanal kontrolü bir arada gerçekleştiren bir sürücü modeli çıkarmak bu tez çalışmasının asıl amacıdır.

1.2 Literatür Özeti

Sürücü modellemede asıl amaç kapalı çevrimde tüm sürücü dinamiklerini elde edebilmektir. Dinamiklerin bulunmasının zorluğu sürücünün yani insanın iç rasgeleliğinden ve yanlılığından ileri gelmektedir. Aynı durum için hiçbir zaman aynı cevabı vermez. Bu nedenlerden dolayı modelleme zorlaşmakta ve bu iş için harcanan zaman artmaktadır. Çeşitli amaçlar için kurulmuş Avrupa'daki Prometheus (PROgraMme for a European Traffic with Highest Efficiency and Unprecedented Safety) Amerika'daki PATH Research (Partners for Advanced Transit and Highways), ALVIN (Autonomous Land Vehicle In a Neural Network), Japonya'daki ARTS, ASV sürücü modelleri üzerinde çalışmaktadır. Örneğin Prometheus'un kuruluş amaçları,

- Trafik güvenliği (Çarpışmaların engellenmesi)
- Çevre kirliliğinin azaltılması (Gürültü ve yakıt kontrolü)
- Sürücü yükünün azaltılması (Akıllı uyarı sistemleri)
- Trafik akışının düzenlenmesi (Yol kapasitesinin optimizasyonu)

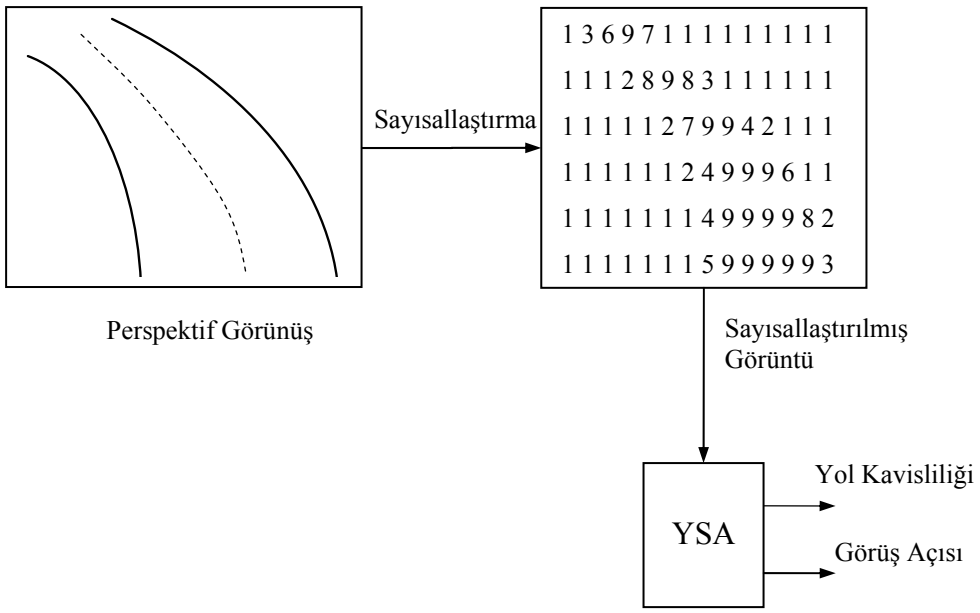
diye sıralanabilir **(An ve diğ., 1994 , An ve Harris, 1996)**.

Sürücü modeli üzerinde bu kadar çok çalışılmasının en önemli amacı kazaların azaltılmasıdır. **Fenton (1994)**'de yılda, 70 milyar doların kazalardan doğan zararlar için harcandığından bahsetmektedir. Günümüzde trafiğin iki kat arttığı düşünülürse zararın büyüklüğü anlaşılabilir. İstatistiklerle bu kazaların sorumlusunun çoğunlukla insan olduğu gösterilmiştir. Oysa taşıtı sürücü ile birlikte kullanan sanal bir sürücünün varlığı ile sürüş esnasında oluşabilecek aşırılıklar önceden fark edilerek sürücü uyarılabilecek hatta direk olarak engellenebilecektir. Bunun daha da ileri aşaması taşıtın sanal sürücü tarafından kullanılmasıdır. Zaten çalışmaların bu hızda devam ettirilmesi halinde yakın gelecekte insansız taşıtlar yapılabilinecektir.

Shim ve diğ. (1995)'de taşıtın önündeki yol için Yapay Sinir Ağları (YSA) ile görsel bir veri işleme metodu geliştirilmiştir. Buna göre YSA görsel bilgiden yolun yarıçapını çıkarmakta ve sürücü modeline giriş olarak verilmesini sağlamaktadır.

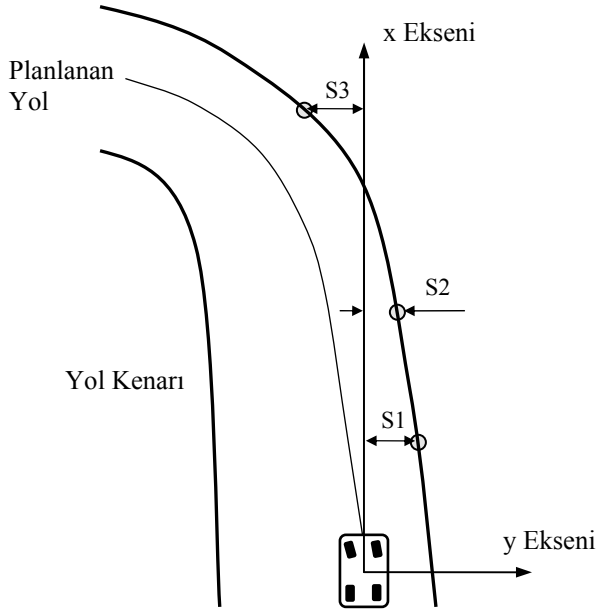
Sürücü modeli çıkış bileşeni olan direksiyon hareketi için, eğrilik yarıçapı ana bileşendir. Çıkarılan sürücü modelinin başka bir yolda da başarılı olabilmesi için yol bilgisinin doğru elde edilmesi gerekmektedir. Kuş bakışından elde edilen yol bilgisinin eksikliğine değinen bu çalışmada, yolun perspektif şeklinden bu bilginin çıkarılmasına çalışılmıştır. Bunun için orta şeride yakın yerlere 9 sayısı, yolun kenarlarına yakın yerlere 1 sayısı verilerek yol sayısallaştırılmış ve eğitilmiş YSA aracılığı ile yol kavisliliği ve görüş açısı elde edilmiştir (Şekil 1.2). YSA ile yol kavisliliği ve görüş açısı; bilinen yol kavisliliği ve görüş açısı ile kıyaslanarak çıkarılır. Eğer yolun bütün özellikleri simülatördeki gibi belli ise bütün yol bilgisi mevcut olduğu için bir daha türetilmesine gerek yoktur.

Sürücü, kuş bakışı görünüşte olduğu gibi olayın dışında değil, aslında bizzat içindedir. Taşıtın içinde olduğundan üzerine etki eden fiziksel büyüklükleri ve bunların türevlerini hisseder. Böylece bu büyüklükler taşıtın kumandası üzerinde dolaylı olarak etkili olur. Kuş bakışı görünüşte, sürücünün gerçekte görebildiğinden daha uzun mesafeler göz önüne alınır. Bu yüzden de modellemede önemli olan, sürücünün doğru bakış açısını ve doğru görüş mesafesini giriş olarak yansıtabilmektir. Örnek olarak sürücü, taşıt hızlandığı zaman daha uzağa, yavaşladığı zaman da yakına bakar; sürekli olarak belirli sabit bir mesafe ileriye bakmaz.



Şekil 1.2 : Perspektif Yol Görünüşünün YSA'ya Giriş Olarak Verilmesi

Macadam ve Johnson (1996)'da yaptıkları çalışmada bir sensörden gelen üç farklı mesafe bilgisi ile direksiyon açısını elde etmişlerdir. Bu çalışmayı diğer çalışmalardan ayıran özellik, sensör bilgilerinin zaman gecikmelerinin de giriş olarak kullanılmasıdır. Şekil 1.3'de görüldüğü gibi S2 ile S1'in farkının iki ölçüm arasındaki doğrusal mesafeye bölünmesiyle yol ile taşıt arasındaki oryantasyon açısı ortaya çıkar (relative yaw angle). İki oryantasyon açısının farkıyla da ilerideki yolun kavisliliğini belirlemişlerdir. Zamana göre türev bilgisinin elde edilmesi için de sensör ölçümlerinin zaman gecikmeleri YSA'ya verilip, eğitimi gerçekleştirmişlerdir. Burada, önemli olan sürücünün algısına yakın yol bilgisinin modele verilmesidir. Sürücü modelinin karmaşıklığından ve doğrusal olmayışından ötürü modelleme için Yapay Sinir Ağları (YSA) kullanmışlardır. Fakat model tek giriş tek çıkışlı olarak çıkartılmıştır (sensör bilgisi giriş – direksiyon açısı çıkış). YSA'yı eğitmek için gereken direksiyon açısı verileri hem simülatörden hem de gerçek yol şartlarından elde etmişlerdir. Aynı çalışmada, **Fujioka ve Takubo (1991) , Kornhausser, (1991), Lubin ve diğ. (1992)** sürücü modeli çıkarmak için simülatör verileri kullanıp YSA modelleri elde ettiklerinden bahsedilmektedir. Bunun yanında aynı mantıkla optik sensör bilgisini giriş olarak kullanan **Neusser ve diğ. (1993)** ise gerçek sürücü verilerini kullanarak YSA modeli çıkarmışlardır.



Şekil 1.3 : Yolun Üstten Görünüşü ve Sensör Ölçümleri

Çoğu çalışmada yapıldığı gibi adı geçen çalışmada da çıkarılan modeller basitleştirilmiş ve bazı ortamlarda simüle edilmiştir. Senaryo olarak sabit hızla çift şerit değişimi ve her iki tarafı eşit yarıçaplı S-eğrisi kullanılmıştır. Sürücü modelinin içinde hızın etkisi olmadığından simülasyonların başarılı olması için taşıt hızını sabit tutmuşlardır. Ancak bu şekilde model gerçeklikten oldukça uzaklaşmış ve fazlasıyla kısıtlanmış olmaktadır. Daha gerçekçi bir model için doğrusal ve yanal bileşenleri birlikte içeren ve her iki harekete de beraber karar verebilen modeller çıkarılması gerekmektedir.

Sürücü modellemesinde en önemli girişlerden biri yol bilgisidir. Fakat bu bilginin nasıl verileceği önemli bir sorudur. Sadece aracın önü ile şerit arasındaki mesafenin (look-down) verilmesi, ya da sürücüye öngörü kazandıracak ilerisindeki bazı noktalardaki yol bilgisinin (look-ahead) verilmesi gibi iki değişik çözüm görünmektedir. Buradaki en önemli ölçüt taşıtın hızına göre sürücüye belli bir avans kazandırmaktır. **Pilutti ve Ulsoy (1999)**'da yaptıkları çalışmada gerçek sürücünün 40 dB/dec'lik bir avansa sahip olduğunu göstermişlerdir. Eğer taşıtımız yavaş seyir hızlarında kullanılacaksa yere bakma (look-down) ile de yeterli avans sürücüye kazandırılmaktadır. İleri bakma mesafesi ile modellemede en önemli kriter, aynı insan davranışında olduğu gibi taşıt hızlandığı zaman ileri bakma mesafesinin artması, yavaşladığı zaman da azalmasıdır (**Guldner ve diğ., 1996**). İleri bakma mesafesi kavramından belirli sabit mesafeler anlaşılırsa yukarıda bahsedilen uyum sağlanamaz. İleri bakmada mesafe yerine, zaman bilgisi modele girilirse gerçeğe daha yakın bir sürücü modeli elde edilmiş olur.

Sürücü modelinde uygulanabilirlik açısından bazı özelliklerin bulunması gerekir. Birinci olarak model sürüş koşullarına uyum sağlamalı, kullanıldıkça yeni bilgiler öğrenmelidir. Model parametreleri değiştirilerek her duruma ayarlanabilmelidir. İkinci olarak, model, yol ve manevralardan bağımsız olmalıdır. Böylece herhangi başka bir yolda sürüş rahatça gerçekleştirilebilir. Modelleme için gereken veriler, test sürücülerinin yani profesyonel sürücülerin taşıtı kullanmasından elde edilmelidir. Sürücü hataları, model içine karıştırılmamalıdır. Ayrıca sürücü modeli için kullanılan giriş-çıkış bileşenleri ve bazı işlevler Tablo 1.1'de verilmiştir **(Riedel ve Wurster, 1998)**.

Tablo 1.1 : Sürücü Modeli için Giriş-Çıkış Bileşenleri ve bazı İşlevler

Girişler	İşlevler	Çıkışlar
Yolun Şekli Direksiyon Açısı Devir Direksiyon Momenti Tekerlek izi	Rota Hız Seçimi Sürmek Gaz / Fren Debriyaj Öğrenme Hız Sınırı	Direksiyon Açısı veya Momenti Gaz Pedalı Debriyaj Vites

Giriş çıkış bileşenlerinin bulunması modellemenin en önemli kısmını oluşturur. Eksik bilgi model başarımını düşürdüğünden en iyi giriş çıkış çiftinin aranması gerekir. Oysa **An ve diğ. (1994) , An ve Harris (1996)**'da doğrusal sürücü modeli çıkarırken giriş olarak taşıt hızı, öndeki taşıtla olan mesafe, öndeki taşıta yaklaşma hızı, yanlılık ve bir önceki kelebek açıklığını alırken; park etme işi için kartezyen koordinatlar, yerel orijine olan uzaklık ve bir önceki hız doğrultusu giriş olarak alınmıştır. Aynı şekilde birinci model için çıkış kelebek açıklığı iken, ikinci modelde çıkış istenen hız ve eğriliktir. Yani, değişen sürüş durumları için farklı giriş-çıkış bileşenlerinin seçilmesi daha uygundur.

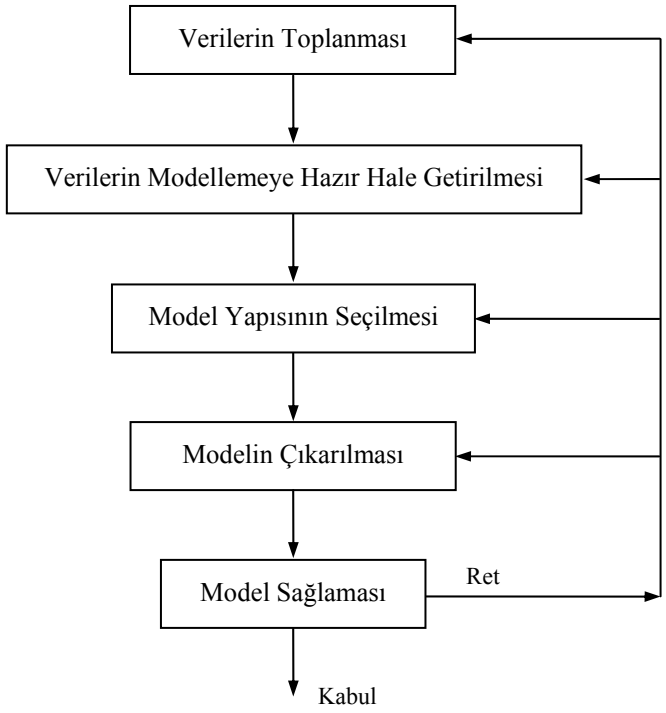
Buradan ve diğer çalışmalardan da anlaşılacağı gibi bir tek optimum (en iyi) giriş-çıkış bileşenleri yoktur. Probleme ve basitleştirmelere bağlı olarak değişen görevle ilgili giriş çıkış bileşenleri bulunabilir. Zaten **Apel (1997)**'de sürücü modeli için kullanabilecek olan fiziksel büyüklükler verilmiştir **(Atabay, 2004)**. Örneğin verilerin toplandığı parkurun düz olmasından dolayı giriş olarak modele eğimin sokulmasına gerek yoktur. Sürücü modeli için seçilen giriş-çıkış bileşenleri Bölüm 5'te verilmiştir. Giriş-çıkış bileşenleri tekrar tekrar denenerek çıkarılmış ve projenin en zahmetli kısımlarından birini oluşturmuştur.

Birçok sürücü modeli gerçek sürücü performansına yaklaşmaktadır. Ancak, hiçbirinden tam anlamıyla gerçek sürücüyü yansıtması beklenmemektedir **(Chen ve**

Ulsoy, 1999). Sürücünün tüm özelliklerini kapsayacak bir model; eğer-o zaman (if-then) ifadeleri ile oluşturulmaya çalışılırsa 10000 ila 50000 arasında kural tanımlanması gerekmektedir **(Atabay, 2004)**. Bu yüzden de her görevi yerine getiren bir model çıkarmak yerine, amacımıza uygun model çıkarmak daha mantıklı olacaktır.

2. SİSTEM TANILAMA

Sistem tanılama, fizik kurallarından yararlanarak matematiksel denklemlerle proseslerin ifadesinden çok, veriler yardımıyla veya doğrusal olmayan modellerin çıkarılması olarak tanımlanabilir. Belli başlı birkaç aşamadan oluşur (Şekil 2.1) **(Babuska, 1998 , Norgaard ve diğ., 2000)**.



Şekil 2.1 : Sistem Tanılamanın Aşamaları

Model çıkarılıp, sağlaması yapıldıktan sonra gerekirse aşamaların herhangi bir yerine, hatta verilerin tekrar toplanması aşamasına kadar dönülebilir.

2.1 Verilerin Toplanması

Verilerin toplanması, ilk ve en önemli aşamadır. Eğer yanlış veri (sistem bilgisini yansıtmayan) toplanırsa bundan sonraki aşamaların hiçbir değeri yoktur. Verinin

sistem bilgisini içermesi için yani zengin veri olması için giriş sinyalinin kesintisiz ve tüm modları uyaran olması gerekir.

2.1.1 Kesintisiz ve tüm modları uyaran giriş sinyali

Bir giriş sinyalinde iki önemli özellik aranır.

a) Kesintisiz Uyarma

b) Tüm Modları Uyarma

Tüm modları uyarma sinyalin frekans bandıyla ilişkilidir ve $[-\infty,\infty]$ arası tüm frekanslarda bileşenleri varsa bu türden bir işarettir (beyaz gürültü). Gerçekte ise $[-\infty,\infty]$ yerine pembe gürültü denilen istenilen bantta bileşen içeren sinyaller giriş için yeterlidir. Sürücü modelinde de giriş sinyalinin 2 Hertz'e kadar bileşen içermesi tüm modları uyarması için yeterlidir. Çünkü modellemede kullanılacak değerler (hız, vites vb.) daha büyük frekanslarda bileşen içermemektedir (Şekil 5.4).

Giriş sinyalimizin sistemimizi kesintisiz bir şekilde uyarmasını istiyorsak, beyaz gürültü, PRBS (Pseudo Random Binary Sequences) gibi giriş sinyallerinin kullanılması gerekir. Elde edilmiş giriş sinyali için yapılacak şey ise, kaçıncı dereceye kadar kesintisiz uyaran olduğun araştırılmasıdır.

n. dereceden kesintisiz uyaran sinyal şu şartları sağlar.

i. $$r_u(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} [u(t+\tau) - m_u][u^T(t) - m_u{}^T] \tag{2.1}$$

limiti var olmalı, ve $m_u = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} u(t)$ 'dir.

ii. $$R_u(n) = \begin{bmatrix} r_u(0) & r_u(1) & \cdots & r_u(n-1) \\ r_u(-1) & r_u(0) & & \\ \vdots & & & \\ r_u(1-n) & & \cdots & r_u(0) \end{bmatrix} \tag{2.2}$$

matrisi pozitif belirli olmalıdır. N veri sayısını, n ise model mertebesini göstermektedir.

Eğer sinyalimiz ergodik ise limit operasyonu beklenen değer operasyonuna dönüşebilir. Ergodiklik durağan olmanın alt şartıdır. Yani her ergodik sinyal durağandır. Böylece R_u (n) bilinen kovaryans matrisine dönüşür. Eğer sinyal periyodik ise $r_u(-m) = r_u(m)$ ve R_u (n) hesaplanması kolaylaşır.

Bu özellikleri bilmeden de kesintisiz uyaranlık şu şekilde bulunabilir. Regresyon vektörü $\varphi(t) = [\, u^T(t-1) \ \ldots \ u^T(t-n) \,]^T$ ise

$$R_u(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} \varphi(t)\varphi^T(t) \tag{2.3}$$

pozitif belirli olmadır **(Söderström ve Stoica, 1989)**.

2.2 Verilerin Modellemeye Hazır Hale Getirilmesi

Bu aşamada yapılacak iki işlemden söz edilebilinir. Biri filtreleme diğeri ise ölçeklemedir. Belli bir frekansın üzerinde gürültü olduğunu düşündüğümüz frekans bileşenleri varsa ikiz-yanılsamaya (anti-aliasing) neden olmadan, bu frekansların sinyalden uzaklaştırılması gerekir. Aslında verinin uygun örnekleme periyodu ve gerekli filtreleme ile toplanması en uygundur.

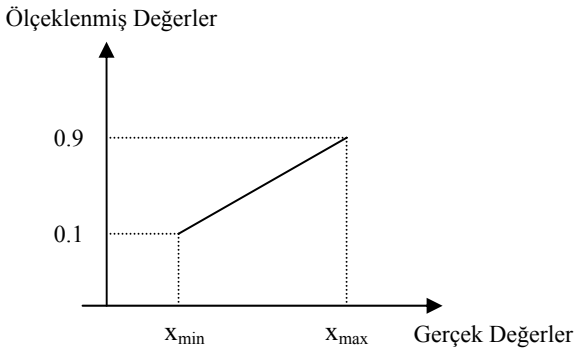
2.2.1 Ölçekleme

Özellikle, Yapay Sinir Ağları (YSA) verilerin mutlak büyüklüklerine duyarlıdır. Mesela birinci çıkış 100'le 1000 arasında ikinci çıkış 0'la 1 arasında ise birinci çıkıştaki dalgalanmalar ikinci çıkışı önemsiz kılacak ve YSA kendini daha çok birinci çıkışa göre eğitecektir. Bu yüzden bütün veriler belli bir aralıkta olmalı, YSA çıkış fonksiyonları da ona göre seçilmelidir (**Arslan, 1999 , Tsoukalas ve Uhrig , 1997**).

Yapay Sinir Ağları ve Bulanık Mantık gibi yapılar doğrusal olmayan sistemleri öğrenebilirler. Ölçekleme de eğitim zamanını azaltır ve içyapıyı basitleştirerek, yüksek performans sağlar.

Üç çeşit ölçeklemeden bahsedebiliriz.

1) Normalizasyon: Bütün verilerin rasgele bir veriye (genellikle en büyük veriye) bölünmesi işlemidir. Eğer birkaç veri diğerlerinden çok daha büyükse bu tür ölçekleme bilgi kaybına neden olabilir.
2) Lineer Ölçekleme: Verilerin istenilen bir aralığa lineer olarak dağıtılmasıdır. Şekil 2.2'de görüldüğü gibi $x_{min} - x_{max}$ arasındaki veriler 0.1-0.9 yada başka bir aralığa Denklem (2.4) ile dağıtılabilir.



Şekil 2.2 : Lineer Ölçekleme

$$y = mx + b$$
$$y = [0.8/(x_{max} - x_{min})]x + [0.9 - 0.8x_{max}/(x_{max} - x_{min})] \quad (2.4)$$

3) Z-score: z-score ise her veriden ortalamasının çıkarılması ve standart sapmaya bölünmesiyle yapılır (Denklem (2.5)). Böylece sıfır ortalama ve birim standart sapmaya sahip bir sinyal elde edilir. Bu yöntemde bilgi kaybı yoktur, aksine veriler ölçü biriminden bağımsız hale gelmiştir.

$$z = \frac{x - m_u}{\sigma} \quad (2.5)$$

burada m_u verilerin ortalamasını, σ ise standart sapmasını göstermektedir.

2.3 Yapının Seçilmesi

Bu aşamanın iki önemli amacı vardır. Birincisi ilgili giriş-çıkış bileşenlerinin belirlenmesi, ikincisi dinamiği yansıtabilecek model mertebelerinin bulunmasıdır. Bu işlemler yapıldıktan sonra problem basit bir statik eşlemeye dönüşür.

2.3.1 Giriş-çıkış bileşenleri

Giriş-çıkış değerlerinin seçilmesi genellikle proses bilgisine sahip modelleyici tarafından belirlenir. İstatistiki korelasyon testlerinin de kullanılabileceği gibi, giriş-çıkış bileşenleri genellikle deneme-yanılma yoluyla bulunabilir. İlgili bileşen modele eklenip performans artışı incelenir. Eğer giriş sistemi doğrudan etkiyen bir bileşen ise belirgin bir oranda iyileşme gözlenir. Aksine, sistemi etkilemiyorsa performans artmaz ve giriş-çıkış bileşenlerinden çıkarılır. Böylece en az giriş-çıkış bileşenine sahip, etkin bir model yapısı belirlenir. Sürücünün modellenmesinde de bu aşamalar izlenmiş, uygun görülen giriş-çıkış bileşenleri Bölüm 5'de anlatılacaktır.

2.3.2 Model mertebesinin tayini

Model mertebesinin bulunması (n_A, n_B), modelleme prosedürünün en kritik aşamasıdır. Bu parametreler iki değişik yaklaşım ile bulunabilir. Doğrusal modelin farklı mertebeler için çıkarılması, kıyaslanması, ona göre n_A, n_B'nin bulunması zor ve uğraştırıcı bir yoldur. Çünkü her n_A, n_B için bütün parametreler bulunur, simülasyonu yapılır. Simülasyonun yapılması da, veri sayısına göre değişmekle birlikte, işlem zamanını artırmaktadır. Onun yerine doğrudan model mertebelerinin bulunabileceği Lipschitz sayısı **(Bomberger ve Seborg , 1998)**, en yakın yanlış komşu (FNN: false nearest neighbour **(Abonyi ve diğ., 2004)**) gibi algoritmalar kullanılabilir. Bu projede, sürücünün modellemesinde mertebe seçimi için Lipschitz sayısı yöntemini kullanılmıştır (Denklem (2.6),(2.7)).

$$q(i,j)^{(s)} = \frac{|y(i) - y(j)|}{\|\varphi(i) - \varphi(j)\|_2} \tag{2.6}$$

$i \neq j$ ve $i, j = 1, 2, ..., N$. N veri sayısını gösterirken, $s = n_A + n_B$ 'dir.

$$Q(s) \equiv \left[\sqrt{s} \prod_{k=1}^{r} q^{(s)}(k) \right]^{\frac{1}{r}} \tag{2.7}$$

burada r pozitif bir sayı olmakla beraber veri sayısının yüzde biri olarak alınabilir. $q^{(s)}(k)$ ise k tane en büyük Lipschitz çarpanıdır. Eğer, optimum model mertebesinden düşük bir mertebe için Lipschitz sayısını $Q(s)$ hesaplarsak bu değer optimal değere karşılık gelen $Q(s)$'den çok büyüktür. Model mertebesinin artırılması ile Lipschitz sayısı küçülmez, nerdeyse sabit kalır. İşte bu noktada model mertebesine karşılık gelen Lipschitz sayıları grafiğinde dirsek noktası en iyi mertebe olarak seçilir. Gerekirse bütün mertebeler için oluşturulan tablodan herhangi bir n_A veya n_B için ince ayar yapılabilir.

2.4 Modelin Çıkarılması

Bu kısımda çok kullanılan doğrusal ve doğrusal olmayan model yapıları anlatılacak, taşıt sürücüsü modelinde kullanılacak yapılar ile ilgili teorik bilgiler Bölüm 3 ve 4'te verilecektir. Doğrusal modelin genel yapısı Denklem (2.8)'de verilmiştir **(Norgaard ve diğ., 2000)**.

$$A(q^{-1})y(t) = q^{-nk}\frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t) \tag{2.8}$$

burada

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na} \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \ldots + b_{nb} q^{-nb} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc} \\ D(q^{-1}) &= 1 + d_1 q^{-1} + \ldots + d_{nd} q^{-nd} \\ F(q^{-1}) &= 1 + f_1 q^{-1} + \ldots + f_{nf} q^{-nf} \end{aligned} \tag{2.9}$$

Denklem (2.8) ile verilen genel yapı birkaç özel hale indirgenebilir. Bunlar sonlu darbe cevaplı model, ARX ve ARMAX modeldir. Doğrusal olmayan modeller ise doğrusal dinamikler temel alınarak YSA mimarisinde çıkarılabilir.

2.4.1 Sonlu darbe cevaplı model (FIR)

Bu tür model yapısı Denklem (2.10)'da gösterilmektedir.

$$\hat{y}(t \mid \theta) = q^{-nk} B(q^{-1}) u(t) \tag{2.10}$$

Denklem (2.10)'u regresyon yapısında yazarsak

$$\hat{y}(t \mid \theta) = \varphi^T(t)\theta \tag{2.11}$$

burada φ regresyon vektörü

$$\varphi(t) = [u(t-n_k) \ ... \ u(t-n_k-n_b)] \quad \text{'dir.} \tag{2.12}$$

n_b girişlerin model mertebesini, n_k girişlerin gecikmesini göstermektedir. Çıkış yani y ise regresand olarak adlandırılır. Parametre vektörü θ ise

$$\theta = [b_0 \ ... \ b_{nb}]^T \ \text{'dir} \tag{2.13}$$

Oldukça basit bir model olan FIR modelin katsayıları darbe cevabının ilk $n_b + 1$ terimine karşılık gelir.

2.4.2 ARX model (AutoRegressive eXogenous input)

Bu tür model yapısı Denklem (2.14) ile gösterilmektedir.

$$\hat{y}(t \mid \theta) = q^{-nk} B(q^{-1}) u(t) + [1 - A(q^{-1})] y(t) \tag{2.14}$$

Denklem (2.14)'i regresyon yapısında yazarsak

$$\hat{y}(t \mid \theta) = \varphi^T(t)\theta \tag{2.15}$$

burada φ regresyon vektörü

$$\varphi(t) = [-y(t-1) \ldots -y(t-n_a)\ \ u(t-n_k) \ldots u(t-n_k-n_b)] \quad \text{'dir.} \tag{2.16}$$

n_a çıkışların model mertebesini, n_b girişlerin model mertebesini ve n_k girişlerin gecikmesini göstermektedir. Çıkış yani y ise regresand olarak adlandırılır. Parametre vektörü θ ise

$$\theta = [a_1 \ldots a_{na}\ \ b_0 \ldots b_{nb}]^T \ \text{'dir} \tag{2.17}$$

ARX tipli doğrusal model en çok kullanılan, çıkarılması basit olduğu kadar sistem dinamiğini de iyi temsil edebilen bir modeldir.

2.4.3 ARMAX model (AutoRegressive Moving Average eXogenous input)

Bu tür model yapısı Denklem (2.18) ile gösterilmektedir.

$$\begin{aligned} \hat{y}(t \mid \theta) &= q^{-nk}\frac{B(q^{-1})}{C(q^{-1})}u(t) + [1 - \frac{A(q^{-1})}{C(q^{-1})}]y(t) \\ &= q^{-nk}B(q^{-1})u(t) + [1 - A(q^{-1})]y(t) + [C(q^{-1}) - 1]\varepsilon(t,\theta) \end{aligned} \tag{2.18}$$

$\varepsilon(t,\theta) = y - \hat{y}(t \mid \theta)$ öngörü hatalarını temsil eder. Denklem (2.18)'i regresyon yapısında yazarsak

$$\hat{y}(t \mid \theta) = \varphi^T(t)\theta \tag{2.19}$$

burada φ regresyon vektörü

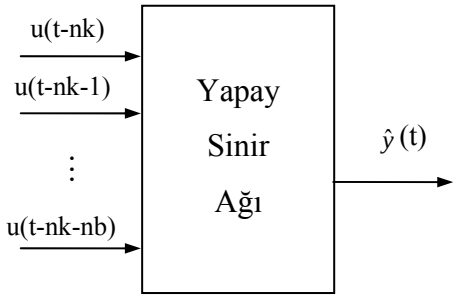
$$\varphi(t) = [-y(t-1) \ldots -y(t-n_a)\ \ u(t-n_k) \ldots u(t-n_k-n_b)\ \ \varepsilon(t-1,\theta) \ldots \varepsilon(t-n_c,\theta)] \tag{2.20}$$

'dir. n_a çıkışların model mertebesini, n_b girişlerin model mertebesini, n_k girişlerin gecikmesini ve n_c öngörü hatalarının model mertebesini göstermektedir. Çıkış yani y ise regresand olarak adlandırılır. Parametre vektörü θ ise

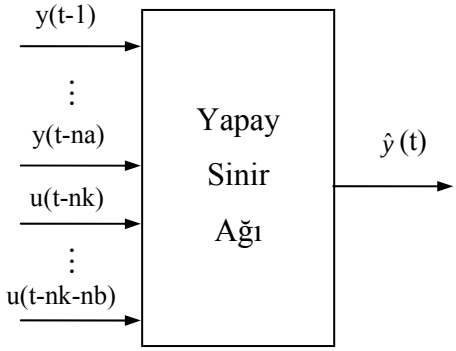
$$\theta = [a_1 \ldots a_{na} \; b_0 \ldots b_{nb} \; c_1 \ldots c_{nc}]^T \quad \text{'dir.} \tag{2.21}$$

2.4.4 Yapay sinir ağı tabanlı doğrusal olmayan model yapıları

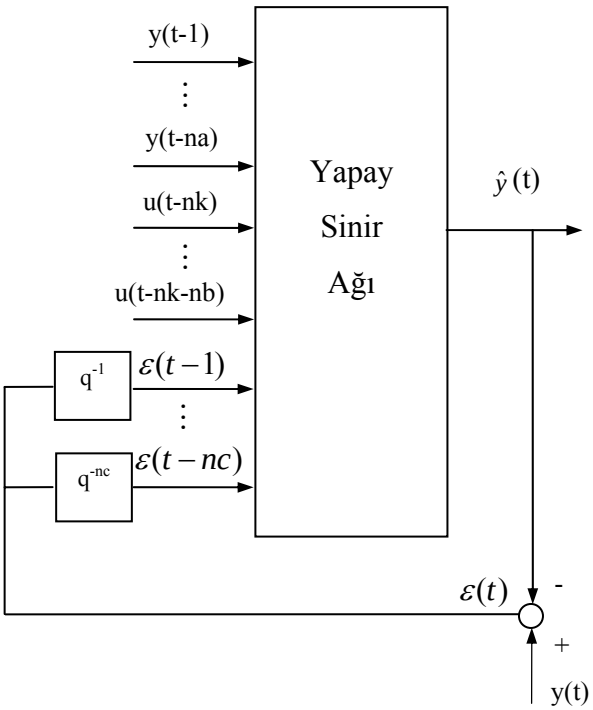
Doğrusal FIR, ARX ve ARMAX yapılarının Yapay Sinir Ağı (YSA) tabanlı doğrusal olmayan yapıları ise Şekil 2.3, Şekil 2.4 ve Şekil 2.5'de gösterilmiştir. Görüldüğü gibi sadece giriş dinamiği değişmekte ancak YSA'nın yapısı benzer kalmaktadır.



Şekil 2.3 : YSA-FIR Model



Şekil 2.4 : YSA-ARX Model



Şekil 2.5 : YSA-ARMAX Model

2.5 Model Sağlaması

Doğrusal ve doğrusal olmayan yöntemlerle çıkarılmış modelin geçerliliğinin test edilmesi gerekir. Bunun anlamı modelin daha önce hiç görmediği veriler üzerinde başarımının araştırılmasıdır. İlk olarak modelin sağlanması görsel olarak çıkışların incelenmesi ile yapılır. İkinci olarak da sınama verilerine bakılır.

Genellikle sistemden toplanan verilerin yarısı eğitim için diğer yarısı ise sınama için kullanılır. Eğitim verileri ile model parametreleri çıkarılırken sınama verileri için yalnızca simüle edilir, parametrelerin ayarlanması üzerinde bir etkinliği yoktur. Sürücü modellemesinde ise bir sürücü verisi eğitim verisi, bir başka sürücü verisi ise sınama verisi olarak alınmıştır. Model sağlamasındaki asıl amaç modelleme verisinden bağımsız modelin elde edilip edilmediğinin kontrolüdür **(Sandberg ve diğ., 2001)**.

3. DOĞRUSAL MODELLEME - ARX MODEL

Bu bölümde en küçük kareler metodu ile doğrusal parametrelerin nasıl bulunacağı anlatılacaktır. Daha sonra ise regresyon vektörünün özel bir hali olan ARX model yapısı için parametrelerin en küçük kareler metodu ile çıkarılmasına değinilecektir.

3.1 En Küçük Kareler Metodu

Bu metot basitliğinin yanında doğrusal parametrelerin kestirimi için en etkili yöntemlerden biridir. Bu nedenlerden dolayı, kontrol mühendisleri için vazgeçilmez bir optimizasyon algoritmasıdır. Çıkarılacak modelin Denklem (3.1) gibi olduğunu düşünelim **(Ikonen ve Najim, 2002 , Abonyi, 2003)**.

$$y(t) = \theta^T \varphi(t) + \varepsilon(t) \tag{3.1}$$

burada θ sütun parametre vektörünü, $\varphi(t)$ regresyon vektörünü, $\varepsilon(t)$ model hatalarını ve $y(k)$ ise hedeflenen çıkışları göstermektedir. Regresyon vektörünün iki önemli özelliği vardır. Birincisi parametre vektöründen bağımsız olması, ikincisi geçmiş zamana ait proses bilgisini içermesidir. Regresyon vektörü statik veya dinamik bir şekilde oluşturulabilinir. Bunun yanında sadece giriş u ya da çıkış y'in yerine bunların çeşitli fonksiyonları trigonometrik, logaritmik vb. şeklinde de oluşturulabilinir. Karesel bir maliyet fonksiyonu tanımlayarak en uygun parametre vektörü bulunabilinir. Minimize edilecek maliyet fonksiyonu;

$$E(\theta) = \sum_{t=1}^{N} (y(t) - \theta^T \varphi(t))^2 \tag{3.2}$$

burada N veri sayısını belirtmektedir. Karesel bir maliyet fonksiyonunun avantajı analitik olarak çözülebilmesi ve tek bir minimum ya da maksimum noktası olmasıdır. Bu noktada Denklem (3.2)'in türevini alınıp sıfıra eşitlemesi ile bulunabilinir.

$$\frac{\partial E}{\partial \theta} = 0 \tag{3.3}$$

$$\frac{\partial E}{\partial \theta} = 2\left[[\sum_{t=1}^{N} \varphi(t)\varphi^T(t)]\ \theta - [\sum_{t=1}^{N} \varphi(t)y(t)] \right] = 0 \tag{3.4}$$

$$[\sum_{t=1}^{N} \varphi(t)\varphi^T(t)]\ \theta = [\sum_{t=1}^{N} \varphi(t)y(t)] \tag{3.5}$$

Denklem (3.5) ‘den θ çekilirse

$$\hat{\theta} = [\sum_{t=1}^{N} \varphi(t)\varphi^T(t)]^{-1}\ [\sum_{t=1}^{N} \varphi(t)y(t)] \tag{3.6}$$

şeklinde bulunur. Denklem (3.6) ile hesaplanan $\hat{\theta}$ vektörü maliyet fonksiyonunu minimum yapması için Denklem (3.2)’nin ikinci türevinin (Denklem (3.7)) pozitif belirli olması gerekir (EK A).

$$\frac{\partial^2 E}{\partial \theta^2} = \sum_{t=1}^{N} \varphi(t)\varphi^T(t) \tag{3.7}$$

En küçük kareler metodunu daha kompakt formda yani matris formunda yazmak istersek,

$$\Phi = \begin{bmatrix} \varphi^T(1) \\ \varphi^T(2) \\ \vdots \\ \varphi^T(N) \end{bmatrix} \tag{3.8}$$

$$\mathbf{y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \tag{3.9}$$

Denklem (3.8) ve (3.9) kullanılarak Denklem (3.1)'i matris formunda yazarsak

$$\mathbf{y} = \Phi\theta + \mathcal{E} \tag{3.10}$$

elde edilir. Maliyet fonksiyonu ise,

$$E(\theta) = (\mathbf{y} - \Phi\theta)^T (\mathbf{y} - \Phi\theta) \quad \text{'dir.} \tag{3.11}$$

Türevinin alınıp, sıfıra eşitlenmesi ile

$$\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T \mathbf{y} \tag{3.12}$$

bulunur. İkinci türev ise

$$\frac{\partial^2 E}{\partial \theta^2} = \Phi^T \Phi \quad \text{'dir.} \tag{3.13}$$

Denklem (3.12)'in hesaplanabilmesi için Denklem (3.13)'ün pozitif belirli olması gerekir (EK A). Aslında $\Phi^T\Phi$ matrisinin tersinin alınabilmesi giriş sinyalinin yeterince uyaran olması ile ilgilidir. Bir sonraki bölümde giriş ve çıkışın dinamikleri ile birlikte içeren ARX doğrusal model en küçük kareler metodu ile çıkarılmaya çalışılacaktır.

3.2 Doğrusal Fark Denklemleri – ARX Model

Sürekli zamanda incelenen dinamik davranış denklemleri sistem verilerinin sürekli olarak toplanabildiği durumlarda geçerlidir. Günümüzde yaygın olarak kullanılan

sayısal ölçme aletleri örnekleme yaparak ölçüm yaptıklarından dolayı bu denklemlerin ayrık zaman domeninde yazılması gereklidir. Herhangi bir doğrusal sistemin dinamik davranış denklemini ayrık zamanlı olarak bir fark denklemi şeklinde ifade etmek mümkündür. Bu durumda genel olarak fark denklemi şu şekilde olacaktır.

$$y(t) = -a_1 y(t-1) - \ldots - a_{n_A} y(t-n_A) + b_0 u(t-n_k) + \ \ldots \ + b_{n_B} u(t-n_k-n_B) \quad (3.14)$$

Denklem (3.14), t anındaki bir sistem çıkışının daha önceki sistem girişleri ve çıkışları cinsinden ifade edilmesi şeklinde açıklanabilir (u: giriş, y: çıkış, n_k: giriş ile çıkış arasındaki gecikme, n_B , n_A sırasıyla giriş ve çıkış mertebeleri). Denklem (3.14) matris formunda yazılırsa;

$$\theta = \left[\ a_1 \ \ldots \ a_{n_A} \ b_0 \ \ldots \ b_{n_B} \ \right]^T \quad (3.15)$$

$$\varphi(t) = \left[\ -y(t-1) \ \ldots \ -y(t-n_A) \ \ u(t-n_k) \ \ldots \ u(t-n_k-n_B) \ \right]^T \quad (3.16)$$

olmak üzere;

$$\hat{y}(t|\theta) = \varphi^T(t) \ \theta \quad (3.17)$$

Denklem (3.17)'de kullanılan $\hat{y}(t|\theta)$, θ vektörüne bağlı hesaplanan $y(t)$ değerini göstermektedir. Denklem (3.16) ile verilen regresyon yapısı ile Denklem (3.8) oluşturulur. Denklem (3.12) kullanılarak da a ve b parametreleri hesaplanabilir.

Doğrusal modellemede zaman gecikmesi n_k için iki önemli nokta vardır.

- $n_k \geq 1$ koşulu sistemin t = 0 anında darbe cevabının sıfir olmasını sağlar (casual). Proses girişi aynı anda proses çıkışını etkileyemez.
- $n_k \leq 0$ olması durumu sinyalin filtreleneceği zamanlarda kullanılır ($n_k = 0$ çevrim içi, $n_k < 0$ çevrim dışı filtreleme).

Performans ölçütü olarak kullanılan yüzde uyum formülü YU şu şekilde tanımlanır,

$$YU = [1 - \frac{\|y - y_m\|_2}{\|y\|_2}] * 100 \tag{3.18}$$

burada y sistem çıkışı y_m modelden elde edilen çıkıştır.

Şu ana kadar tanımlanan denklemler tek-giriş tek-çıkışlı bir sistem için verilmiştir. Denklem (3.19)-(3.23) arasında ise çok-giriş çok-çıkışlı bir sistem için gerekli denklemler bulunabilir. Artık a, b, n_A, n_B katsayıları skaler değil, matristir ($\mathbf{A, B, n_A, n_B}$).

$$\mathbf{A}(q^{-1})\mathbf{y}(t) = \mathbf{B}(q^{-1})\mathbf{u}(t) + \boldsymbol{\varepsilon}(t) \tag{3.19}$$

$\mathbf{A}$ ve $\mathbf{B}$ matrisleri ise,

$$\mathbf{A}(q^{-1}) = I + A_1 q^{-1} + \ldots + A_{n_A} q^{-n_A} \tag{3.20}$$

$$\mathbf{B}(q^{-1}) = B_1 q^{-1} + \ldots + B_{n_B} q^{-n_B} \tag{3.21}$$

Sistem çıkışı gerçek değer için yazılırsa,

$$\mathbf{y}(t) = \varphi^T(t)\theta + \boldsymbol{\varepsilon}(t) \tag{3.22}$$

$$\varphi(t) = [-\mathbf{y}(t-1) \ldots -\mathbf{y}(t-n_A) \;\; \mathbf{u}(t-d) \ldots \mathbf{u}(t-d-n_B)]^T \tag{3.23}$$

olur **(Söderström ve Stoica, 1989)**.

4. DOĞRUSAL OLMAYAN MODELLEME – YAPAY SİNİR AĞLARI

4.1 Yapay Sinir Ağlarının Gelişmesi

Uzun yıllardan beri, insan beyninin yapısına ve yeteneklerine olan hayranlık ve merak, pek çok araştırıcının ilgisi sayesinde yeni çalışma alanlarının ortaya çıkmasına neden olmuştur. Mevcut bilgisayarlarla ve algoritmalarla çözülemeyen ya da iyi sonuçlar alınamayan fakat insan beyninin kolayca yapabileceği karmaşık problemlere çözümler üretmek için son yıllarda yapılan araştırmalar sonucu, yeni bir bilgi işleme yöntemi olan Yapay Sinir Ağları (YSA) doğmuştur **(Yıldızdoğan, 1995)**.

Yapay Sinir Ağları birbirlerine paralel olarak bağlanmış basit işlem elemanlarından oluşmuş ve biyolojik sinir sisteminin yaptığı işlevlere benzer şekilde hiyerarşik olarak hazırlanmış ağlardır.

YSA ile ilgili çalışmalar 1943 yılında McCulloch ve Pitts tarafından geliştirilen ilk hücre modeli ile başlamıştır. Hebb, 1949 yılında hücreler arasındaki bağlantıları düzenlemek için ilk öğrenme kuralını geliştirmiştir. 1958 yılında Rosenblatt'ın geliştirdiği algılayıcı (perceptron) modeli ve öğrenme kuralı ile bugün kullanılan kuralların temelleri ortaya çıkmıştır.

Minsky ve Papert'ın 1969 yılında yaptıkları algılayıcının analizi ile perceptronun karmaşık lojik fonksiyonlar için kullanılamayacağını ispatlamaları ile YSA üzerine yapılan araştırmalar neredeyse durma noktasına gelmiştir. 1976'da Grossberg'in Adaptif Rezonans Teorisi (ART), 1982'de Hopfield'in doğrusal olmayan dinamik Hopfield ağını (recurrent network) ve 1984'te Kohenen'in eğiticisiz öğrenen bir YSA geliştirmesi ile YSA üzerine yapılan çalışmalar yeniden başlamıştır.

1986 yılında Rumelhart, 1974 yılında Werbos tarafından geliştirilen çok katmanlı algılayıcı tipi ağlar için geriye yayılma (back propagation) öğrenme algoritmasını geliştirmiştir. Böylece günümüzde en sık kullanılan öğrenme algoritması ortaya çıkmıştır. Günümüzde YSA birçok alanda da başarı ile kullanılmakta ve devamlı geliştirilmektedir **(Hagan ve diğ., 1995)**.

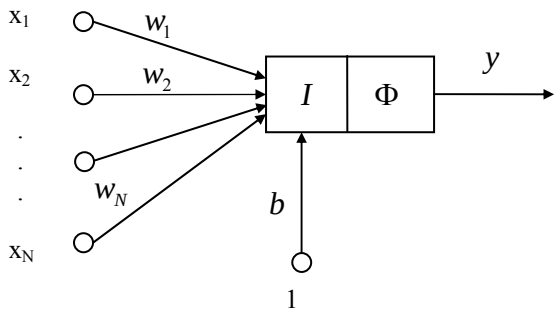
4.1.1 Yapay sinir ağlarının özellikleri

YSA'nın alışılagelmiş bilgi işleme yöntemlerine göre üstünlükleri maddeler halinde şu şekilde sıralanabilir **(Yıldızdoğan, 1995 , Efe ve Kaynak, 2000)**;

- Paralellik: Alışılagelmiş bilgi işleme yöntemlerinin çoğu seri işlemlerden oluşmaktadır. Bu da özellikle hız problemlerine sebep olmaktadır. Mesela bilgisayarlar, beyine göre çok daha hızlı çalışmasına rağmen beynin toplam hızı bilgisayara göre kıyaslanamayacak kadar yüksektir. YSA'da ise aynı katmandaki hücreler arasında zaman bağımlılığı yoktur. Bu yüzden tamamıyla eşzamanlı çalışabilirler. Bu da hızı çok arttırmaktadır.
- Hata Toleransı: Geleneksel bir bilgisayarda, herhangi bir işlem elemanını yerinden almak onu etkisiz bir makineye dönüştürür. Ancak YSA'da bir elemanda meydana gelen hasar performansta ciddi bir düşüşe yol açmaz.
- Gerçekleme Kolaylığı: YSA, karışık fonksiyonlar yerine basit işlemleri içerdiği için gerçekleme kolaydır. Bu sayede doğrusal olmayan sistemlerin tanımlanmasında önemli bir role sahiptir.
- Yerel Bilgi İşleme: YSA'da her bir işlem birimi, çözülecek problemin tümü ile ilgilenmek yerine, sadece problemin bir parçası ile ilgilenmektedir. Hücrelerin çok basit işlemler yapmalarına rağmen, sağlanan görev paylaşımı sayesinde çok karmaşık ve zor problemler çözülebilmektedir.
- Öğrenebilirlik: Alışılagelmiş veri işleme yöntemlerinin çoğu programlama yolu ile hesaplamaya dayanmaktadır. Bu yöntemler ile tam tanımlı olmayan bir problem çözülemez. Ayrıca, herhangi bir problemin çözümü için probleme yönelik bir algoritma geliştirmek gerekir. YSA ise problemleri, verilen örnekler ile çözer. Çözülecek problemler için algoritma değil, parametreler değiştirilir.

4.2 Basit Nöron Yapısı ve Etkinlik Fonksiyonları

Yapay sinir ağları, insanın biyolojik sinirsel yapısının incelenmesi ve modellenmesi ile ortaya çıkmıştır. Yapay bir nöron, gerçek bir sinir hücresinin sahip olduğu bağlantılarla aynı işlevi yapan bağlantılara sahiptir. Yapay bir nöronun matematik modeli Şekil 4.1'de görülebilir.



Şekil 4.1 : Yapay Nöronun Matematik Modeli

Burada, $\mathbf{x} = [x_1 \; x_2 \; \ldots \; x_N]^T$ vektörü nörona uygulanan girişlerden oluşur. Bütün girişler belirli ağırlık oranına göre nörona etki eder. Yani her bir giriş kendisiyle ilgili ağırlık terimiyle çarpılır. $\mathbf{w} = [w_1 \; w_2 \; \ldots \; w_N]$ ağırlık vektörünü gösterirken, b ise sabit girişin ağırlığı olup yanlılık terimi (bias) olarak adlandırılır. Bu tanımlar altında yapay nöronun girişi (I) ve çıkışı (y) şöyle yazılabilir:

$$I = w_1 x_1 + w_2 x_2 + \ldots + w_N x_N + b = \sum_{i=1}^{N} \mathbf{w}\mathbf{x} + b \tag{4.1}$$

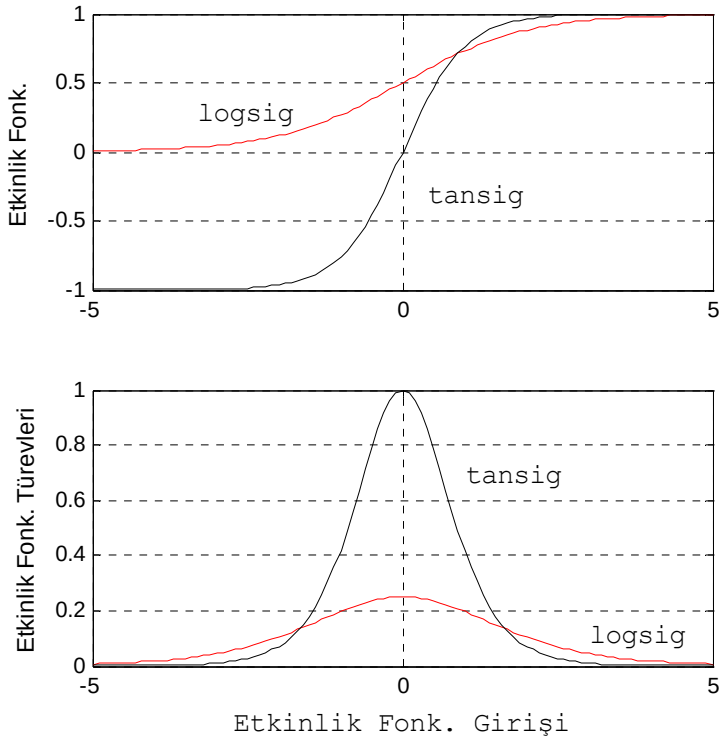
$$y = \Phi(I) \tag{4.2}$$

Buradaki Φ etkinlik veya aktivasyon fonksiyonu olarak adlandırılır. YSA'larda kullanılan bazı etkinlik fonksiyonları ve türevleri Tablo 4.1'de görülebilir.

Tablo 4.1 : Etkinlik Fonksiyonları ve Türevleri

Doğrusal	$\Phi(I) = I$	$\dot{\Phi}(I) = 1$
Logaritmik Sigmoid	$\Phi(I) = \dfrac{1}{1+e^{-\alpha I}}$	$\dot{\Phi}(I) = \alpha \; \Phi(I)(1-\Phi(I))$
Tanjant Sigmoid	$\Phi(I) = \dfrac{e^{\alpha I} - e^{-\alpha I}}{e^{\alpha I} + e^{-\alpha I}}$	$\dot{\Phi}(I) = \alpha \; (1-\Phi(I)^2)$

Tablo 4.1'deki α aktivasyon fonksiyonlarının eğim parametresidir. α artıkça etkinlik fonksiyonları dikleşir, azaldıkça eğikleşir. $\alpha = 1$ için logaritmik sigmoid (logsig) ve tanjant sigmoid (tansig) etkinlik fonksiyonlarının ve türevlerin şekli Şekil 4.2'de görülebilir.



Şekil 4.2 : Logsig ve Tansig Etkinlik Fonksiyonları ve Türevlerinin Şekli

4.3 Doğrusal Olmayan En Küçük Kareler Metodu

Bu bölümde en küçük kareler metodunun, doğrusal olmayan parametreleri bulmak için kullanılacak algoritmalar anlatılacaktır. Daha sonraki bölümlerde ise çıkarılan optimizasyon algoritmalarının Yapay Sinir Ağlarına uygulanışı ve bu şekilde ağırlıkların güncellenmesi irdelenecektir. Minimize edilecek karesel maliyet fonksiyonu,

$$\begin{aligned} \varepsilon^2(\boldsymbol{\theta}) &= \sum_{k=1}^{N} (T_k - \hat{y}_k)^2 \\ &= \mathbf{e}^T(\boldsymbol{\theta})\mathbf{e}(\boldsymbol{\theta}) \end{aligned} \tag{4.3}$$

dir. Burada T_k hedef değeri, $\hat{y}_k$ hesaplanan model çıkışını, $\mathbf{e}$ hata vektörünü göstermektedir. Model çıkışını, φ giriş vektörü için $\boldsymbol{\theta}$ parametre vektörü ile Denklem (4.4)'den hesaplanabilir.

$$y = f(\varphi, \boldsymbol{\theta}) \tag{4.4}$$

f herhangi bir fonsiyonu temsil etmek üzere Denklem (4.3)'ün Gradyen ve Hessian'inin bulunması ile Gauss-Newton ve Levenberg-Marquardt gibi algoritmalar türetilebilir. Karesel hatanın gradyeni

$$\begin{aligned} \mathbf{g}(\boldsymbol{\theta}) &= \frac{\partial \varepsilon^2(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\ &= 2\sum_{k=1}^{N} e_k(\boldsymbol{\theta}) \frac{\partial e_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\ &= 2\mathbf{J}^T\mathbf{e} \end{aligned} \tag{4.5}$$

Burada $\mathbf{J}$ hata vektörünün Jakobian'idir. İkinci türevleri içeren Hessian matrisi ise

$$\begin{aligned} \mathbf{H}(\boldsymbol{\theta}) &= \frac{\partial \varepsilon^2(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \\ &= 2\sum_{k=1}^{N} \left[\frac{\partial e_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{\partial e_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^T} + e_k(\boldsymbol{\theta}) \frac{\partial^2 e_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \right] \\ &= 2(\mathbf{J}^T\mathbf{J} + \mathbf{S}) \end{aligned} \tag{4.6}$$

$\mathbf{S}$ matrisi

$$\mathbf{S}(\boldsymbol{\theta}) = \sum_{k=1}^{N} e_k(\boldsymbol{\theta}) \frac{\partial^2 e_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \quad \text{'dir.} \tag{4.7}$$

Denklem (4.4)'ün birinci derece Taylor serisi ne açılımı ve karesel hatanın parametre vektörüne göre türevinin sıfıra eşitlenmesinden ardışık iki parametre vektörü

arasındaki ilişki çıkarılır **(Jang ve diğ., 1997)**. Gauss-Newton formülü ile bilinen bu formül Denklem (4.8)'de verilmiştir.

$$\begin{aligned}\boldsymbol{\theta}(i+1) &= \boldsymbol{\theta}(i) - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e} \\ &= \boldsymbol{\theta}(i) - \frac{1}{2}(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{g}\end{aligned} \tag{4.8}$$

$\mathbf{J}^T\mathbf{J}$ matrisinin basitçe birim matris gibi düşünülmesi halinde algoritma eğim düşüm yöntemine dönüşür. Aslında Gauss-Newton algoritması, Newton algoritmasından da türetilebilir. Newton algoritması,

$$\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) - \frac{1}{2}\mathbf{H}^{-1}\mathbf{g} \quad \text{'dir.} \tag{4.9}$$

Denklem (4.9) ikinci derece Taylor serisi açılımından elde edilmiştir. Denklem (4.6) 'dan Hessian ve Denklem (4.5)'den Gradyen Denklem (4.9)'da yerine konursa

$$\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) - (\mathbf{J}^T\mathbf{J} + \mathbf{S})^{-1}\mathbf{J}^T\mathbf{e} \tag{4.10}$$

elde edilir. Görüldüğü gibi Gauss-Newton algoritması, Newton algoritmasındaki $\mathbf{S}$'in $\mathbf{J}^T\mathbf{J}$'in yanında ihmal edilmesine dayanır.

4.3.1 Levenberg-Marquardt optimizasyon algoritması

Uygulamada Gauss-Newton algoritmasının (Denklem(4.8)) çok önemli bir sorunu vardır. Hata yüzeyinin bazı noktalarında $\mathbf{J}^T\mathbf{J}$'in tersi alınamamakta ya da matrisin pozitif-belirli (EK A) olmamasından dolayı minimum noktayı göstermemektedir. Bu gibi problemler $\mathbf{J}^T\mathbf{J}$'in köşegenine $\lambda\mathbf{I}$'ın eklenmesiyle giderilebilir. Dikkat edilirse, λ'ın çok büyük değerleri için algoritma eğim düşüm yöntemine dönüşür. Levenberg-Marquardt algoritması,

$$\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) - \frac{1}{2}(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{g} \quad \text{'dir.} \tag{4.11}$$

λ 'nın başlangıç değerleri probleme bağlıdır ve deneme yanılmayla bulunabilir. Nasıl güncelleneceği ile ilgili bilgiler **Fletcher (1971)** , **More (1977)** , **Nash (1977)**'de bulunabilir **(Jang ve diğ., 1997)**.

4.4 Geriye Yayılım Algoritması

Çok katmanlı Yapay Sinir Ağı (YSA) eğitimi iki aşamadan oluşur. İleri geçişte, giriş sinyali çıkışa doğru yayılır. Geriye gelişte ise, hata sinyali ağa yayılarak ağırlıkların ayarlanması sağlanır. Çıkış katmanında hedef belli olduğundan hata kolayca bulunur ve Delta kuralına göre ağırlıklar güncellenebilir. Hedefi belli olmayan saklı katmandaki nöronların ağırlıkları ise çıkış katmandaki hatanın zincir kuralı ile katman katman geriye yayılmasıyla güncellenebilir. Prosedür şu şekilde gerçekleşir **(Tsoukalas ve Uhrig , 1997)**:

1. Rasgele küçük değerler, genellikle -1 ile 1 arasında, ağırlıklara atanır. Ağırlıkların rasgele atanması yerine, daha etkili bir yol **Nguyen ve diğ. (1990)'**dan incelenebilinir. Bu ilk atama yolu ile aktivasyon fonksiyonlarının aktif bölgeleri kullanılarak ağırlıklar belirlenir ve daha iyi bir yakınsama sağlanır.

2. Bir eğitim çifti (giriş-çıkış) seçilir.

3. Giriş ağa uygulanır.

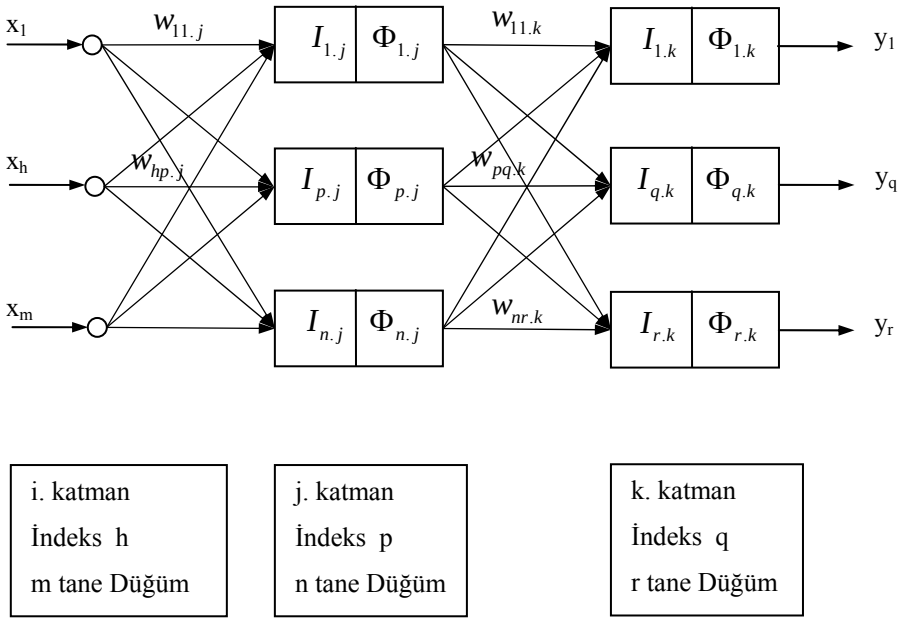
4. Ağın çıkışı hesaplanır.

5. Hedef çıkışla, ağ çıkışı arasındaki hata hesaplanır.

6. Hata (ya da kayıp) fonksiyonunu azaltacak şekilde ağırlıklar ayarlanır.

7. Eğitim kümesindeki hatalar kabul edilebilir düzeye inene kadar 2.-6. adımlar tekrarlanır.

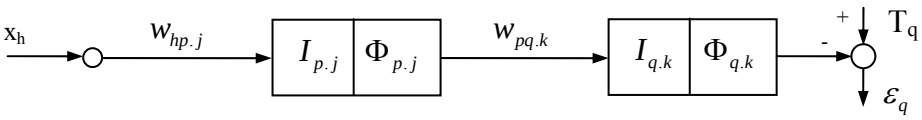
Altıncı adım ile ağırlıların güncellenmesi işlemine çok katmanlı bir ağda Geriye Yayılım Algoritması denir. Geriye yayılım algoritması basitlik açısından üç katmanlı ileri beslemeli bir ağ (Şekil 4.3) için bütün ağdaki aktivasyon fonksiyonları logaritmik sigmoid olacak şekilde çıkarılacaktır.



Şekil 4.3 : Üç Katmanlı İleri Beslemeli Yapay Sinir Ağı

4.4.1 Çıkış katmanındaki ağırlıkların hesaplanması

Şekil 4.4'de herhangi iki p ve q nöronunun aralarındaki güncellenecek çıkış ağırlığı $w_{pq.k}$ görülmektedir. Sembollerdeki k indisi çıkış katmanını işaret etmektedir. T_q hedef değeri olmak üzere, minimize edilecek karesel hata,



Şekil 4.4 : Çıkış Katmanındaki Ağırlıkların Hesaplanması

$$\varepsilon^2 = \varepsilon_q^2 = (T_q - \Phi_{q.k})^2 \tag{4.12}$$

olarak tanımlanır. Delta kuralına göre ağırlık değişimi

$$\Delta w_{pq.k} = -\eta_{pq} \frac{\partial \varepsilon_q^2}{\partial w_{pq.k}} \tag{4.13}$$

burada η_{pq} çıkış katmanının ağırlıklarının öğrenme oranı olarak adlandırılır. Zincir kuralı ile karesel hatanın türevi

$$\frac{\partial \varepsilon_q^2}{\partial w_{pq.k}} = \frac{\partial \varepsilon_q^2}{\partial \Phi_{q.k}} \frac{\partial \Phi_{q.k}}{\partial I_{q.k}} \frac{\partial I_{q.k}}{\partial w_{pq.k}} \tag{4.14}$$

şeklinde yazılabilir. Denklemin sağ tarafının karşıtları ise,

$$\frac{\partial \varepsilon_q^2}{\partial \Phi_{q.k}} = -2(T_q - \Phi_{q.k}) \tag{4.15}$$

$$\frac{\partial \Phi_{q.k}}{\partial I_{q.k}} = \alpha \ \Phi_{q.k}(1 - \Phi_{q.k}) \tag{4.16}$$

$$\frac{\partial I_{q.k}}{\partial w_{pq.k}} = \Phi_{p.j} \tag{4.17}$$

Denklem (4.15), Denklem (4.12)'den; Denklem (4.16) logaritmik sigmoid aktivasyon fonksiyonun türevinden (Tablo 4.1); Denklem (4.17) ise Denklem (4.18) 'den elde edilmiştir. Denklem (4.18)'in yazılışı için Şekil 4.3'den yararlanılmıştır.

$$I_{q.k} = \sum_{p=1}^{n} w_{pq.k} \Phi_{p.j} \tag{4.18}$$

Böylece Denklem (4.14)

$$\frac{\partial \varepsilon_q^2}{\partial w_{pq.k}} = -2\alpha(T_q - \Phi_{q.k})\Phi_{q.k}(1 - \Phi_{q.k})\Phi_{p.j} \tag{4.19}$$

halini alır. Hata işaret terimi

$$\delta_{pq.k} = 2\alpha(T_q - \Phi_{q.k})\Phi_{q.k}(1-\Phi_{q.k}) \quad \text{'dir.} \tag{4.20}$$

Son olarak çıkış katmanındaki ağırlık değişimi

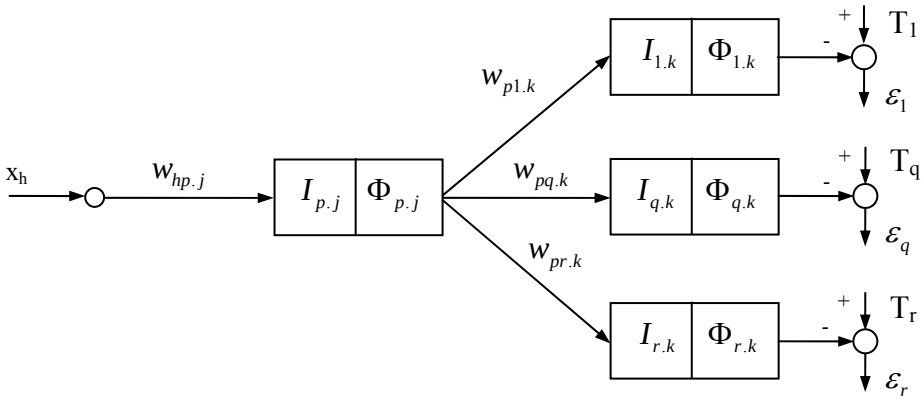
$$\Delta w_{pq.k} = -\eta_{pq}\frac{\partial \varepsilon_q^2}{\partial w_{pq.k}} = \eta_{pq}\delta_{pq.k}\Phi_{p.j} \tag{4.21}$$

$$w_{pq.k}(N+1) = w_{pq.k}(N) + \eta_{pq}\delta_{pq.k}\Phi_{p.j} \tag{4.22}$$

olarak bulunur. N iterasyon sayısını gösterir.

4.4.2 Saklı katmandaki ağırlıkların hesaplanması

Saklı katmandaki ağırlıkların hedef değerleri olmadığından çıkış katmanındaki hatanın bu katmana yayılması gerekir. Buda saklı katmandaki her nöron için hata işaret terimi olan $\delta_{hp.j}$ 'nin hesaplanması gerektirir. Bütün yapılması gereken işlemler çıkış katmanındaki gibidir. Ancak, parçalı türevler bir toplam işareti altında bütün çıkış nöronları için hesaplanır (Şekil 4.5). Çünkü çıkış katmanındaki bir ağırlık tek bir çıkıştaki hatadan etkilenirken, saklı katmandaki bir ağırlık tüm çıkışlardaki hatadan etkilenmektedir.



Şekil 4.5 : Saklı Katmandaki Ağırlıkların Hesaplanması

Saklı katman içinde delta kuralını uygularsak

$$\Delta w_{hp.j} = -\eta_{hp} \frac{\partial \varepsilon^2}{\partial w_{hp.j}} = -\eta_{hp} \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial w_{hp.j}} \tag{4.23}$$

burada η_{hp} saklı katmandaki ağırlıkların öğrenme oranı olarak adlandırılır. Zincir kuralı ile karesel hatanın türevi

$$\frac{\partial \varepsilon^2}{\partial w_{hp.j}} = \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial \Phi_{q.k}} \frac{\partial \Phi_{q.k}}{\partial I_{q.k}} \frac{\partial I_{q.k}}{\partial \Phi_{p.j}} \frac{\partial \Phi_{p.j}}{\partial I_{p.j}} \frac{\partial I_{p.j}}{\partial w_{hp.j}} \tag{4.24}$$

şeklinde yazılabilir. Denklemin sağ tarafındaki ilk iki terim Denklem (4.15) ve (4.16) ile aynıdır. Diğer üç tarafın eşitlikleri ise

$$\frac{\partial I_{q.k}}{\partial \Phi_{p.j}} = w_{pq.k} \tag{4.25}$$

$$\frac{\partial \Phi_{p.j}}{\partial I_{p.j}} = \alpha \ \Phi_{p.j}(1 - \Phi_{p.j}) \tag{4.26}$$

$$\frac{\partial I_{p.j}}{\partial w_{hp.j}} = x_h \quad \text{‘dir.} \tag{4.27}$$

Denklem (4.26), saklı katmandaki lojistik aktivasyon fonksiyonun türevidir (Tablo 4.1). Denklem (4.25) ve (4.27), sırasıyla aşağıdaki Denklem (4.28) ve (4.29)’den elde edilmiştir. Denklem (4.28) ve (4.29)’in yazılışı için Şekil 4.3 incelenebilir.

$$I_{q.k} = \sum_{p=1}^{n} w_{pq.k} \Phi_{p.j} \tag{4.28}$$

$$I_{p.j} = \sum_{h=1}^{m} w_{hp.j} x_h \tag{4.29}$$

Böylece Denklem (4.24)

$$\frac{\partial \varepsilon^2}{\partial w_{hp.j}} = \sum_{q=1}^{r} -2\alpha(T_q - \Phi_{q.k})[\Phi_{q.k}(1-\Phi_{q.k})]w_{pq.k}\alpha\ [\Phi_{p.j}(1-\Phi_{p.j})]x_h \qquad (4.30)$$

halini alır. Saklı katman için

$$\delta_{\text{hp.j}} = \delta_{\text{pq.k}}\ w_{pq.k}\alpha\ [\Phi_{p.j}(1-\Phi_{p.j})] \quad \text{'dir.} \qquad (4.31)$$

Saklı katmandaki ağırlık değişimi ise

$$\Delta w_{hp.j} = -\eta_{hp}\frac{\partial \varepsilon^2}{\partial w_{hp.j}} = \eta_{hp}\sum_{q=1}^{r}\delta_{hp.j}x_h \qquad (4.32)$$

$$w_{hp.j}(N+1) = w_{hp.j}(N) + \eta_{hp}x_h\sum_{q=1}^{r}\delta_{hp.j} \qquad (4.33)$$

olarak bulunur. N iterasyon sayısını gösterir.

Sadece ağırlık terimleri ile çok katmanlı bir ağ, sıfır girişlere sıfır olmayan çıkışlar üretemez. Bunun için her bir nöronun yanlılık (bias) terimlerinin olması gerekir. Eğitim aynen ağırlıkların eğitimi gibi olup yalnızca girişi +1 kabul edilir (-1'de olabilir). Aynı nöron sayısı için parametre sayısı arttığından ağın performansı artar.

$$b_{pq.k}(N+1) = b_{pq.k}(N) + \eta_{pq}\delta_{pq.k}\ (+1) \qquad (4.34)$$

$$b_{hp.j}(N+1) = b_{hp.j}(N) + \eta_{hp}\ (+1)\sum_{q=1}^{r}\delta_{hp.j} \qquad (4.35)$$

Eğim düşüm yöntemi ağırlıkların ve yanlılık terimlerinin güncellenmesi kararlı fakat yavaş bir algoritmadır. Eğitim zamanını azaltmak için ya Uyarlamalı Eğitim kullanılabilinir ya da ağırlık değişimine (ağırlık değişimi ile hem ağırlık hem de yanlılık terimleri kastedilmektedir) Momentum terimi eklenebilir. Uyarlamalı eğitimde hata gittikçe azalıyorsa öğrenme oranı belli oranda arttırılır. Eğer birkaç

iterasyon hata artıyorsa öğrenme oranı azaltılır **(Jang ve diğ., 1997 , Hagan ve diğ., 1995)**. Öğrenme oranının uyarlanması ile minimum noktasına yaklaşıldığında türev genliklerinin küçülmesinden kaynaklanan yavaşlamada engellenmiş olacaktır. Verilerde bulunabilecek gürültülerden, türev genliklerini anlık büyümelerinden kaynaklanan sorunlar hata teriminde dalgalanmalara sebep olacaktır. Bu sorun Denklem (4.36) ile sanki birinci derece alçak geçiren bir filtre'den geçirilmiş gibi çözülebilinir **(Efe ve Kaynak, 2000)**.

$$\Delta w_{pq.k}(N+1) = \eta_{pq}\delta_{pq.k}\Phi_{p.j} + \mu\,\Delta w_{pq.k}(N) \tag{4.36}$$

Burada μ momentum terimidir. Momentum metodunun bir önemli avantajı da yerel minimumlara takılmayı engelleyip, algoritmayı global minimuma götürmesidir.. Ancak bir önceki ağırlık değişimini de hafızada tutmak gerektiğinden hesap yükü eğim düşüm yöntemine göre daha fazladır.

4.5 YSA için Levenberg-Marquardt Algoritması

Geriye yayılım algoritması YSA'ların eğitimi için en çok tercih edilen algoritmalardan biridir. Algoritmanın basitliği, hesap yükünün azlığı gibi avantajlarının yanında, uzun eğitim zamanı gerektirmesi, yakınsama problemleri gibi dezavantajları vardır. Bu bölümde standart eğim düşüm yönteminin çok hızlı yakınsayan Levenberg-Marquardt (LM) algoritmasına dönüştürülmesi anlatılacaktır. LM, grup uyarlamalı eğitim (batch training) için çıkarılacak yani ağırlıklar tüm eğitim verileri için her iterasyonda bir defa güncellenecektir **(Hagan ve Menhaj, 1994)**.

$\boldsymbol{\theta}$ ağırlık ve yanlılık terimlerinden oluşan parametre vektörüdür ve minimize edilecek maliyet fonksiyonu

$$\varepsilon^2 = \sum_{k=1}^{N}(T_k - \hat{y}_k)^2 = \sum_{k=1}^{N}\mathbf{e}^T{}_k\mathbf{e}_k \quad \text{'dır.} \tag{4.37}$$

Burada T_k hedeflenen değeri, $\hat{y}_k$ ağın çıkışını ve $\mathbf{e}_k$ ise k. girişe karşılık gelen ağın hatasını ve N veri sayısını göstermektedir. Levenberg-Marquardt formülü ağırlıkların değişimi için yazılması gerekirse,

$$\Delta\boldsymbol{\theta} = -(\mathbf{J}^T(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta}) + \lambda I)^{-1}\mathbf{J}^T(\boldsymbol{\theta})\mathbf{e}(\boldsymbol{\theta}) \tag{4.38}$$

denklemi elde edilir. Jakobian matrisi,

$$\mathbf{J}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial e_1(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial e_1(\boldsymbol{\theta})}{\partial \theta_2} & \cdots & \frac{\partial e_1(\boldsymbol{\theta})}{\partial \theta_n} \\ \frac{\partial e_2(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial e_2(\boldsymbol{\theta})}{\partial \theta_2} & \cdots & \frac{\partial e_2(\boldsymbol{\theta})}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial e_N(\boldsymbol{\theta})}{\partial \theta_2} & \cdots & \frac{\partial e_N(\boldsymbol{\theta})}{\partial \theta_n} \end{bmatrix} \tag{4.39}$$

Çok çıkışlı bir YSA için birinci ve ikinci satır arasında diğer çıkışlara ait hataların parametlere göre türevlerinin de yazılması gerektiği unutulmamalıdır. YSA için maliyet fonksiyonu

$$\varepsilon^2(\boldsymbol{\theta}) = \sum_{k=1}^{N*r} e_k{}^2(\boldsymbol{\theta}) \quad \text{'dir.} \tag{4.40}$$

N veri sayısını, r ise YSA'nın çıkışın nöron sayısını gösterir.

Herhangi bir iterasyon adımında maliyet fonksiyonu artarsa, λ parametresi belli bir katsayı ile çarpılarak büyütülür. Maliyet fonksiyonun azalması halinde ise aynı katsayı ile bölünerek diğer iterasyona geçilir. Büyük λ sayıları için LM algoritması $1/\lambda$ öğrenme oranına sahip eğim düşüm yöntemine dönüşürken λ'nün çok küçük değerleri için Gauss-Newton algoritmasına dönüşür **(Norgaard ve diğ., 2000)**.

LM algoritmasının kullanılabilmesi için parametre vektörünün Jakobian'inin (Denklem (4.39)) hesaplanması gerekmektedir. Jakobian matrisinin elemanları

standart geriye yayılım algoritması ile hesaplanan $\delta_{pq.k}$ teriminde ufak bir değişiklikle hesaplanabilir. Son katmandaki $\delta_{pq.k}$ terimi,

$$\delta_{pq.k} = \alpha \ \Phi_{q.k}(1-\Phi_{q.k}) \tag{4.41}$$

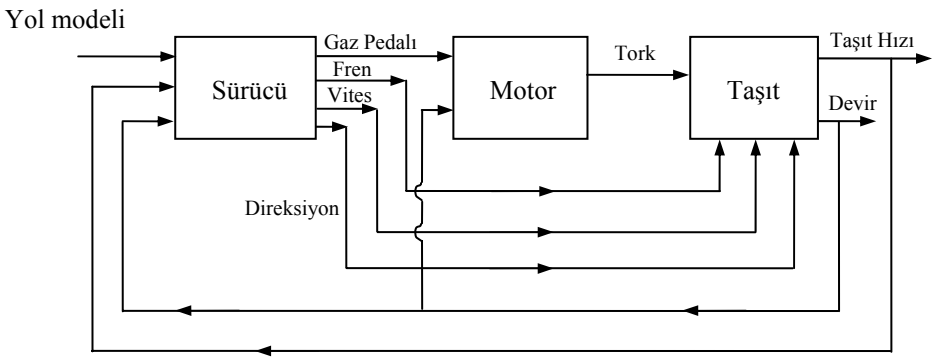
olarak alırsak Geriye Yayılım Algoritması LM algoritmasına dönüştürülmüş olur. Yukarıdaki denklemin Denklem (4.20)'den farkı hata terimini içermemesidir, çünkü hatanın karesi yerine hatanın türevi alınmıştır.

LM algoritması şu şekilde çalışır;

1. Bütün girişler ağa verilerek bunlara karşılık gelen çıkışlar hesaplanır. Ağ çıkışları kullanılarak hatalar
2. Denklem (4.41), (4.31), ve Denklem (4.37)'nin ağırlıklara göre türevi kullanılarak Jakobian matrisi hesaplanır.
3. Denklem (4.38)'den $\Delta\boldsymbol{\theta}$ bulunur.
4. Maliyet fonksiyonu $\boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ için tekrar hesaplanır. Eğer maliyet fonksiyonu küçülürse λ belli bir oranda küçültülür ve 1. aşamaya geri dönülür. Maliyet fonksiyonun artması halinde ise λ büyültülerek 3. aşamaya geri dönülür.
5. Algoritma istenen maliyet fonksiyonuna ya da iterasyon sayısına ulaşıncaya kadar tekrarlanır.

5. TAŞITIN YANAL VE DOĞRUSAL KONTROLÜ İÇİN SÜRÜCÜNÜN MODELLENMESİ

Basitçe belirtilmiş sürüş kapalı çevrimini (Şekil 1.1) biraz daha açmak gerekirse, bu çevrimi oluşturan üç blok (Sürücü, Motor, Taşıt blokları) ve etkileşimleri Şekil 5.1'de görülebilir. Bilgisayar simülasyonları için bu üç bloğun modellenmesi gerekmektedir. Bu çalışmada sürücü bloğunu temsil edecek bir model elde edilmiştir. Matematik modeller yerine sistem tanılama ile modelleme çalışmaları yürütüldüğünden, giriş-çıkış ilişkilerini iyi betimleyen veri grubunun mutlaka elde edilmesi gerekmektedir. Bu nedenden dolayı giriş (Yol, Taşıt hızı, Motor devir sayısı) – çıkış (Gaz Pedalı, Fren, Vites, Direksiyon açısı) bileşenlerinden herhangi birinin eksikliği, başarımın (performansın) düşmesine neden olmaktadır.



Şekil 5.1 : Kapalı Çevrim Blok Diyagramı

Bir kontrolör olan sürücüye ait veriler sürücü-motor-taşıt kapalı çevriminden elde edilmektedir. Kapalı çevrimden elde edilen verilerin mümkün olduğu kadar sistemi uyaran, bir başka deyişle zengin olması gerekmektedir. Çünkü giriş zengin bile olsa kapalı çevrimde fakirleşir (**Shaw, 1993**). Bu yüzden sistem tanılama için normal bir sürüş sonucunda elde edilen verilerin yeterince uyaran olup olmadığı kontrol edilmeli, değil ise başka bir sürücü verisi denenmelidir (**Pilutti ve Ulsoy, 1999**).

Bu aşamada herhangi bir sensöre gerek duymadan bütün verilerin istenilen şekilde toplanması bir simülatör aracığıyla gerçekleştirilmiştir. Gerçek şartlarda yol bilgisinin zor toplanması, uğraştırıcılığı ve bazı sensörlerden toplanan bilginin 1/0 şeklinde olması simülatörün diğer önemli avantajlarını gösterir.

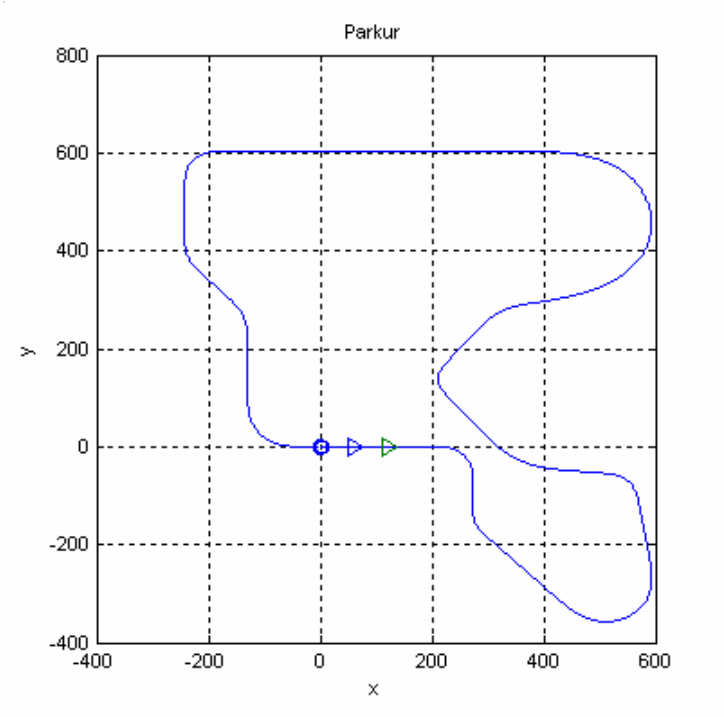
5.1 Simülatör

Simülatör Visual C++ yazılımı ile Dr. Orhan Atabay tarafından 6 yıllık bir çalışmanın sonucunda gerçekleştirilmiş olup, motor, taşıt dinamiği, yol ve çevre gibi her türlü unsur modellenmiştir. İlk olarak sürüş simülatörüne baz olacak şekilde dört tekerlekli bir karayolu taşıtının dinamiği belirli kabuller eşliğinde modellenmiş, denklemlerle ve haritalarla ifade edilmiştir. Taşıt modeli, pnömatik lastik tekerlek, direksiyon sistemi, tekerlek asılış sistemi, fren sistemi, motor ve aktarma organları sistemleri ve gövde alt modellerinden oluşmaktadır **(Atabay, 2004)**.

Taşıt modeli bir sürüş simülatörüne hizmet etmekte olduğundan, bir yandan "gerçek zaman" da çalışması sağlanmış, diğer taraftan da çok geniş seyir durumları uzayı için test edilerek akla yakın sonuçlar üretip üretmediği sorgulanmıştır. Taşıt modelinin sürüş simülatöründen bağımsız olarak yalnız çalışabilmesini temin etmek üzere, modelin veri tahrikli olarak çalışabilen bir "çevrim dışı" modu da oluşturulmuştur. Bu durumda herhangi bir sürücü giriş verisi ile sistem, gerçek bir sürücü olmadan da (gerçek zamandan daha hızlı bir şekilde) çalışabilmektedir.

Sürüş simülatöründe en temel öğelerden olan sanal çevre görüntüsünün ve gürültüsünün oluşturulabilmesi için Microsoft DirectX® SDK kütüphaneleri kullanılmıştır. Sürücünün ihtiyaç duyacağı şartlara uygun ve yüksek kaliteli bir görüntü elde edilerek, sanal ağaçlar, trafik işaretleri ve pürüzlü asfalt kaplama görüntüsü ile sürücünün gerçeğe yakın bir algı ortamı içinde sürme işini yapması sağlanmıştır.

Sürüş parkurunun hazırlanması için ayrı bir yazılım oluşturulmuştur. Bu yazılım ile karayolu yapımı ilkelerine ters düşülmeden uygun geometride bir sürüş parkuru hazırlanmıştır (Şekil 5.2). Ses ve gürültü simülasyonu alt modülü ile içten yanmalı motor sesinin sürücü tarafından duyulması sağlanmıştır. Tüm simülasyon donanımını taşıyacak şekilde, sabit bir simülatör platformu hazırlanmıştır (Şekil 5.3).

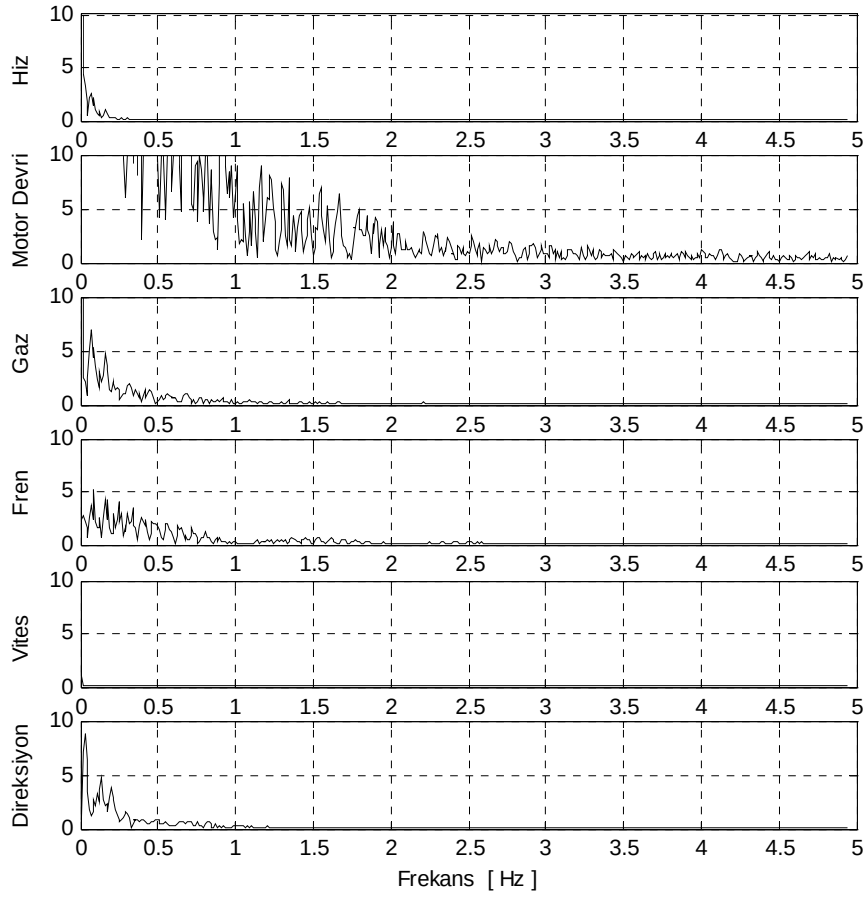


Şekil 5.2 : Sürüş Parkuru

Şekil 5.3 : Masaüstü PC Simülatör

Simülatörde çok sayıda sürücüden istenen her türlü veri toplanabilecek, bu veriler kullanılarak elde edilecek "Sanal Sürücü" ile sanal yol testi ve motor testleri yapılabilecektir. En önemlisi, gerçek sürücüyle yapılan yol testi ile sanal sürücüyle yapılan sanal yol testi sonuçları arasındaki yakıt tüketimi, egzoz emisyonu vb. farklar yüzde olarak elde edilebilecektir. Simülatör normal çalışma esnasında taşıta ve sürücüye ait 65 kanal veriyi 20 Hz örnekleme hızında kaydedebilmektedir. Kanal sayısı azaltılarak örnekleme hızı arttırılabilir.

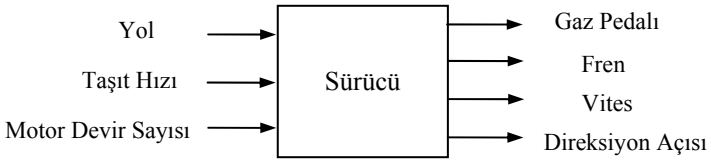
İnsan operatörlerinin band genişliği birkaç Hertz'i geçmemektedir. Şekil 5.4'den de görülebileceği gibi sürücü modelinde kullanılan giriş-çıkış bileşenleri bazı manevralar hariç 2 Hz'in üzerindeki frekanslarda bileşen içermemektedir. Bu bilgiler eşliğinde en yüksek frekanslı bileşeni kaçırmamak için örnekleme frekansının 10 Hz seçilmesi yeterlidir (**An ve Harris, 1996 , Pilutti ve Ulsoy, 1999 , Shaw, 1993 , Guldner ve diğ., 1996**).



Şekil 5.4 : Modelde Kullanılacak Bileşenlerin Frekans Bandı

5.2 Sürücü Modellemesi

Modellemede en önemli nokta, sistemi etkileyen giriş-çıkış büyüklüklerinin belirlenmesidir. Bu işlemin matematiksel bir temeli veya sistematik bir yaklaşımı yoktur. Bu büyüklükler ancak ve ancak modeli iyi tanıyan ya da bilen bir operatör tarafından verilebilir. Projenin en uğraştırıcı kısmı bu büyüklüklerin belirlenmesi olmuştur. Çünkü belirlenen her farklı model için işlemlerin hepsi tekrarlanmakta, ve sonuçlar karşılaştırılmaktadır. Yapılan denemelerde en iyi performansı veren model Şekil 5.5'de görülmektedir. Çıkışların kaç gecikme ile geri besleneceği Şekil 5.5'de gösterilmemiş, model mertebesi belirleme yöntemleri ve deneme-yanılma yolu ile bulunmuştur. Çıkarılacak sürücü modelimizi diğer çalışmalardan ayıran en önemli özellik, yol bilgisinin sürücüye dönme açısı olarak verilmesidir.



Şekil 5.5 : Sürücü Modeli Giriş-Çıkış Bileşenleri

Modeldeki en önemli parametre ileri bakma mesafesidir (look-ahead distance) ve şu şekilde hesaplanmıştır: Öndeki taşıtla aradaki mesafe, taşıt hızının yarısının metre cinsinden karşılığı olmalıdır. Başka bir deyişle bu 2 saniyeye karşılık gelir. Veriler 10 Hertz ile örneklendiğinden en fazla ileri bakma mesafesi 20 örnekleme ötesi olarak bulunur. Bunun dinamik olarak modele verildiği düşünülürse, sürücü 20 örnekleme ötesi ile taşıt arasındaki bölgeye bakmaktadır [yol(k+20) yol(k+19) ... yol(k+2) yol(k+1)]. Buradaki önemli bir çıkarım da, bu kadar çok girişin kullanılmasının gereksiz olduğunun görülmesidir. Yani yol verisi dizisindeki 20 değerin hepsinin kullanılmasına gerek yoktur. Ardışık olmayan birkaç verinin seçilmesi daha uygundur (k+20,k+13,k+6 gibi). Fazla girişin kullanılması, saklı katmanda daha fazla nöron kullanılmasına bir başka deyişle yapay sinir ağının daha karmaşık bir hal almasına neden olur. Büyük bir ağ uzun eğitim zamanı ve ezberleme sorunlarını da beraberinde getirmektedir. Aynı şekilde, Bulanık Mantıkla modellemede girişin fazla olması üstel olarak kural sayısını artırmaktadır.

Ülkemiz şartlarında kullanılacak insansız bir araç ya da kapalı çevrimde yapılacak motor testleri için, bütün yol bilgilerimizi içerecek zenginlikte olan bir yol bilgisinin veya parkurun varlığı, modelin gerçekçiliğini arttıracaktır. Bir verinin matematiksel olarak nasıl zengin olacağı Sistem Tanılama bölümünde anlatılmıştır. Kısaca değinmek gerekirse, zenginlikten kastedilen, yolun olabildiğince çeşitli eğrilik yarıçapları ve iniş-çıkışlar içermesidir.

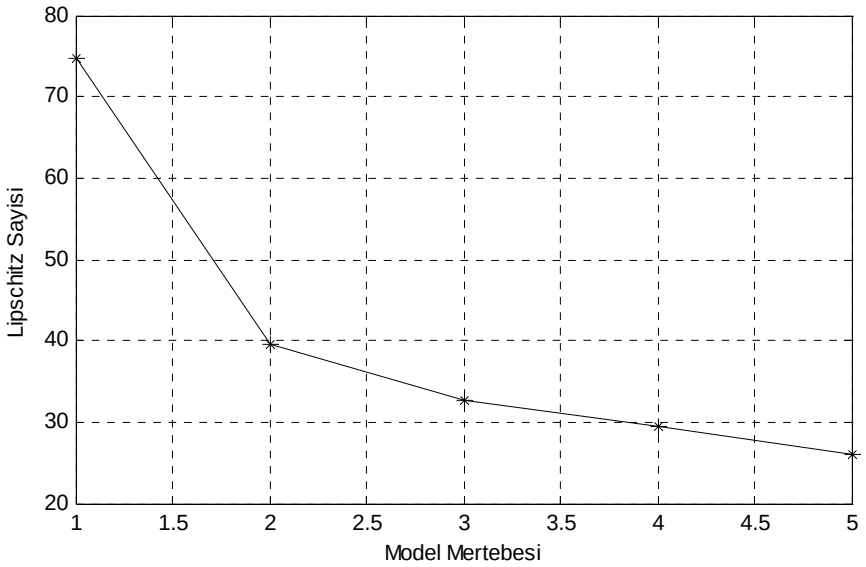
Aslında bir sürücü modelinin çıkarılması karmaşık ve zor bir problemdir. Çünkü sürücü her durum için bilgiliyken, çıkarılacak model sadece verilerin içerdiği bilgiye sahiptir. Yani model, trafiğin olmadığı eğimsiz bir yolda çok iyi bir sürüş yapıp parkuru tamamlayabilirken, mesela park etme işlemini gerçekleştiremeyebilir. Bunun için her işlemi gerçekleştirebilen alt parçaların (modül) hazırlanması, duruma göre o parçanın devreye girmesi gerekmektedir. Bugüne kadar yapılmış çalışmalar hep tek bir senaryo için tasarlanmıştır. Sürücü, şeridi korumak (lane-keeping), sabit hızda belli bir yarıçaptaki yolu gitmek, vb. gibi farklı görevler için modellenmiştir. Kontrollü bir deneyde bir değişken dışındaki tüm değişkenlerin sabit tutulması gibi, hızın sabit olması onu modelden düşürür, böylece model basitleşir. Model basitleştikçe de başarımın artması beklenebilir. Yapılan en önemli basitleştirme ise sürücünün tek çıkışlı olarak modellenmesidir. Şekil 5.5'den de görülebileceği gibi bu çalışmada dört çıkışlı, taşıtın doğrusal ve yanal kumandasını birlikte sağlayan bir sürücü modeli çıkarılmaya çalışılmıştır.

Denklem (2.3) kullanılarak sinyalimizin 8. dereceye kadar kesintisiz uyaran olduğu bulunmuştur. Sonuç olarak 10 Hertz'le örneklenmiş sinyal kesintisizlik özelliğini sağlamakta ve sistemimizin modlarını uyarmaktadır. Yani literatürdeki adı ile verilerin "Zengin Veri" olduğu söylenebilir. Bu da elde edilecek sürücü modelinin, modelleme verisinden bağımsız olarak sürücünün tüm dinamik karakteristiklerini yansıtmasını sağlayacaktır.

Veriler, ortalamalarının çıkarılması ve standart sapmalarına bölünmesiyle ölçeklenmiş, böylece ölçü biriminden bağımsız hale getirilmiştir.

Model mertebesine karşılık gelen Lipschitz sayıları grafiğinde (Şekil 5.6) dirsek noktası en iyi mertebe olarak seçilir. Şekilden de görüldüğü gibi giriş ve çıkışın en uygun model mertebesi 2'dir. Gerekirse Tablo 5.1'den de herhangi bir n_A veya n_B için ince ayar yapılabilir. **Söderström ve Stoica (1989)**'daki uyarma derecesi ≥

2*(model mertebesi) olması yeterli ilişkisi de hatırlanarak zaten 4. dereceye kadar mertebe bulabileceğimiz unutulmamalıdır.



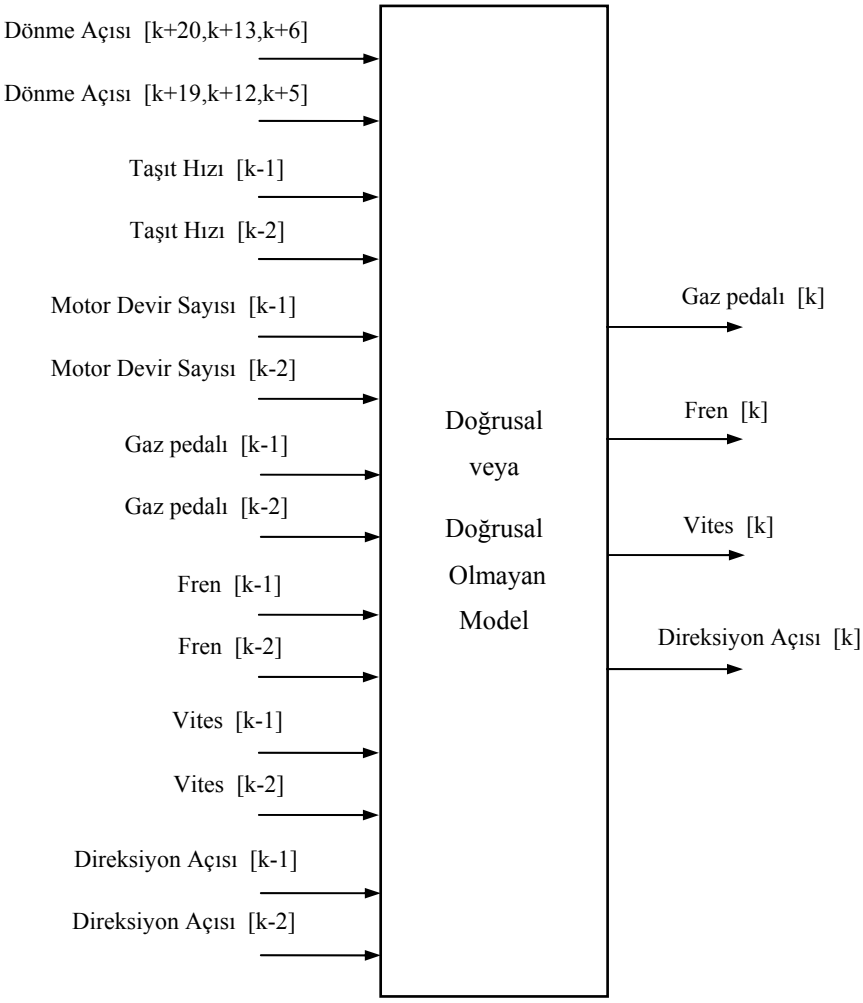
Şekil 5.6 : Model Mertebesine ($n_A = n_B$) Karşı Bulunan Lipschitz Sayıları

Tablo 5.1 : Model Mertebelerine Karşı Bulunan Lipschitz Sayıları

	nb	1	2	3	4	5
na	1	74.73	43.37	34.68	29.17	26.66
	2	64.65	39.58	32.53	29.23	26.82
	3	56.05	39.89	32.75	29.41	26.96
	4	54.15	39.17	32.51	29.56	27.09
	5	46.07	35.00	30.36	27.95	25.99

Giriş-çıkış bileşenleri ve model mertebesi bulunmuş, ayrıca ölçeklenmiş veriler için bundan sonra yapılması gereken regresyon vektörü ile regresand arasındaki ilişkinin (eşlemenin) bulunmasıdır. Bu ilişki öncelikle doğrusal model sonra da doğrusal olmayan model kullanılarak bulunmuştur. Şekil 5.7'de giriş-çıkışın mertebeleri ile modele verilmesi gösterilmiştir. En genel haldeki regresyon vektörü ARX model için Denklem (2.16) ile verilmiş YSA-ARX model için ise Şekil 2.4'de gösterilmiştir. Bu bilgiler ışığında regresyon vektörünün Dönme açısı[k+20, k+19, k+13, k+12, k+6, k+5], Taşıt hızı[k-1, k-2], Motor devir sayısı[k-1, k-2], Gaz pedalı[k-1, k-2], Fren[k-1, k-2], Vites[k-1, k-2], Direksiyon açısı [k-1, k-2] ; regresand'ın ise Gaz pedalı[k], Fren[k], Vites[k], Direksiyon açısı[k] olacağı açıktır. Regresyon ve regresand

vektörüne bakılarak 18 girişli 4 çıkışlı bir dinamik model çıkarılmış olduğu görülebilir.



Şekil 5.7 : Sürücünün Modellenmesi

En küçük kareler metodu kullanılarak (çok giriş çok çıkışlı matris formu) ARX modelin parametreleri bulunmuştur. Ancak çıkarılan ARX yapısındaki doğrusal model kararsızdır. Doğrusal olmayan sürüş görevi için ARX model yetersiz kalmıştır.

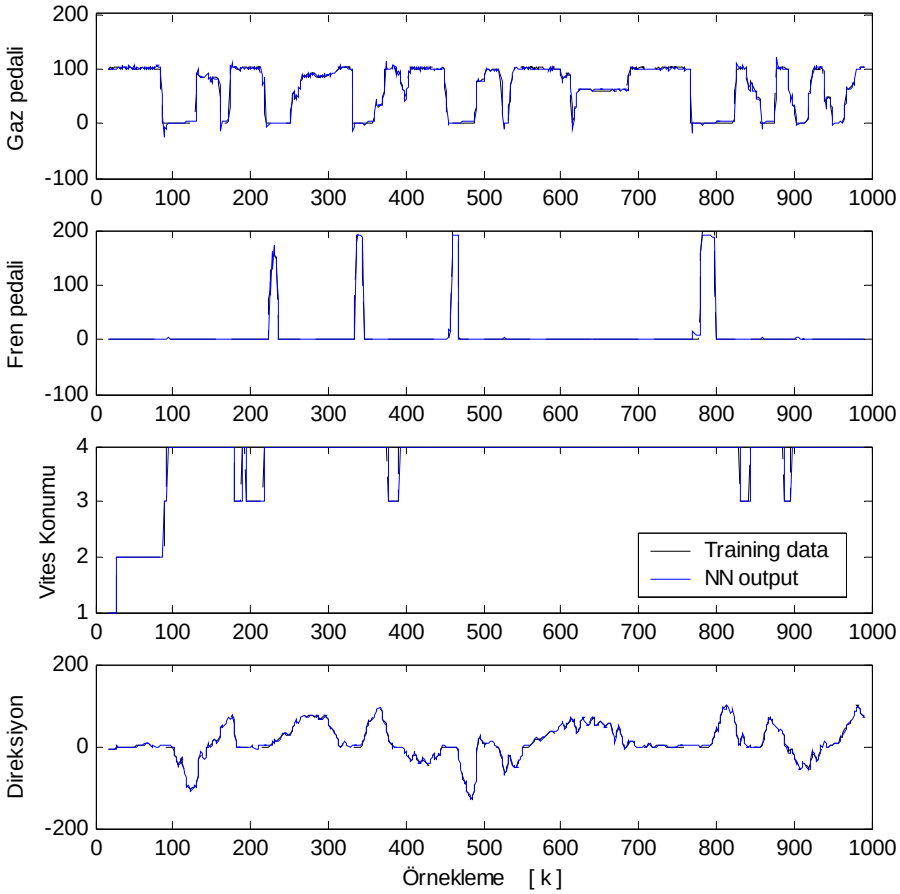
YSA tabanlı doğrusal olmayan model çıkarılmasının, doğrusal ARX model çıkarılmasından belli başlı farklılıklarının yanında karar verilmesi gereken tasarım parametreleri de vardır. YSA'da eğitim algoritması olarak, Bölüm 4.4'de anlatılmış geriye yayılım algoritmasının değiştirilmesi ile elde edilen Levenberg-Marquardt algoritması kullanılmıştır. Bütün eğitimler için çevrim sayısı 20 alınmıştır. Çevrim

sayısına, LM algoritması ile birkaç eğitimden sonra karar verilmiştir. Çünkü 20 iterasyondan sonra hataların karelerinin ortalamaları düşmemekte, sabit kalmaktadır. Ayrıca YSA artan iterasyon sayısı ile gürültülere duyarlı hale gelir ve gerçek sistem karakteristiğini kaybeder. Seçilen bu çevrim sayısı ile aslında genelleştirme, gürbüzlük ve kesinlik arasında optimum bir nokta belirlenmiş olur.

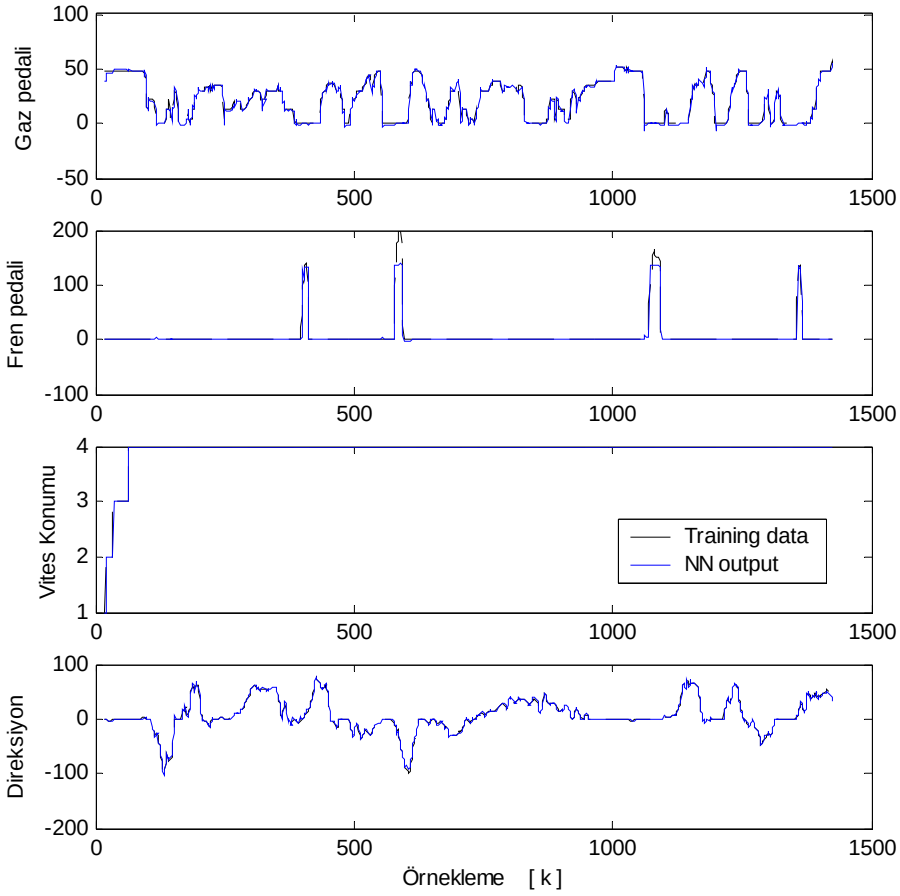
Saklı katmandaki en uygun nöron sayısını ve aktivasyon fonksiyonunu bulabilmek için yapılan denemeler Tablo 5.2'de görülebilir. Çıkış katmanında doğrusal aktivasyon fonksiyonları kullanılmıştır. Bunun yanında tek saklı katman istenen performansın elde edilmesinde yeterli olmuştur. Tablo 5.2'deki hataların karelerinin ortalamaları ve çıkışların yüzde uyumları, YSA'nın başlangıç ağırlıklarına bağımlı olmasından dolayı birkaç kez çalıştırılıp bulunmuştur. Saklı katmanda kullanılan iki etkinlik fonksiyonunun (logsig ve tansig) birbirine göre belirgin bir avantajı yoktur. Saklı katmanda tanjant aktivasyon fonksiyonları kullanılmıştır. Saklı katmandaki en uygun nöron sayısı ise 6 olarak bulunmuş, model çıkışları eğitim verileri için Şekil 5.8 ve sınama verileri için Şekil 5.9'da gösterilmiştir.

Tablo 5.2 : Sürücü Modeli için Denenen bazı YSA Yapılandırmaları

		Hataların Karesel Ortalaması		Yüzde Uyum								Bağımsız Parametre Sayısı
				Gaz Pedalı		Fren		Vites		Direksiyon Açısı		
Saklı Katmandaki Nöron Sayısı	Saklı Katmandaki Etkinlik Fonksiyonları	Eğitim Kümesi	Sınama Kümesi	Eğitim Kümesi	Sınama Kümesi	Eğitim Kümesi	Sınama Kümesi	Eğitim Kümesi	Sınama Kümesi	Eğitim Kümesi	Sınama Kümesi	
1	logsig	0.5946	0.5619	45.3	43.6	30.4	27.90	29.8	-1.4	0	-0.1	27
3	logsig	0.1511	0.1392	51	55.3	74.6	74.90	42.6	60.70	84.80	86.00	73
5	logsig	0.0245	0.035	82.9	77.8	80.4	73.9	81.7	78.7	92.6	92.6	119
6	logsig	0.0202	0.0602	83	77.4	83.9	76.5	82.5	78.7	93	93.3	142
8	logsig	0.02	0.0611	83	81.4	82.6	78.9	83.3	78.7	93	90.3	188
10	logsig	0.0204	0.0786	83	79.9	83.0	72.1	81.7	65.6	92.6	91.7	234
1	tansig	0.5939	0.554	44.9	42.8	31.80	29	23.4	-6.7	0.3	1.1	27
3	tansig	0.152	0.1405	56.6	65.3	77.5	68.1	38.6	50.4	87.9	89.3	73
5	tansig	0.023	0.0375	82.5	76.5	82.4	74.8	80.9	78.7	92.9	92.4	119
6	tansig	0.0211	0.0454	82.5	80.8	83.5	77.3	81.7	78.7	93.3	93.8	142
8	tansig	0.0179	0.0669	84.1	80	83.9	73	82.5	83.5	92.9	93.1	188
10	tansig	0.0169	0.0811	84.2	79.9	85.1	73.7	83.3	78.7	93.3	92.9	234



Şekil 5.8 : Eğitim Verileri için Gerçek ve YSA Çıkışları (SKNS =6)



Şekil 5.9 : Sınama Verileri için Gerçek ve YSA Çıkışları (SKNS =6)

6. SONUÇLAR

Bu projede, taşıtın yanal ve doğrusal kumandasını bir arada gerçekleştiren sürücü modeli, klasik ve akıllı yöntemlerle çıkarılmıştır. Öncelikle, doğrusal parametrik bir model olan ARX model üzerinde çalışılmış daha sonra ARX model dinamiğine sahip YSA ile sürücü modeli çıkarılmıştır. Bir kontrolör olan sürücü modelinin çıkarılması ile de sürücü-taşıt kapalı çevrimi oluşturulabilecektir. Oluşturulan bu çevrim taşıtın ve motorun bilgisayar ortamında geliştirilmesini sağlayacaktır.

Sürücünün taşıtla etkileşimi insan operatörün karakteristiği itibariyle çok karmaşıktır. Modellemenin zorluğu insanın iç rasgeleliğinden ve yanlılığından ileri gelmektedir. Çünkü insan aynı durum için hiçbir zaman aynı cevabı vermez. Bu nedenlerden dolayı, sistemi, yani taşıt sürücüsünü deterministik bir model ile tanımlamak zorlaşmaktadır. Sistem modelini verecek denklemlerin doğrudan yazılması mümkün olmadığı için sistem tanılama yöntemleri kullanılarak doğrusal ve doğrusal olmayan parametrik modeller çıkarılması zorunludur.

Taşıt sürücüsünün modellenmesinde en önemli aşama giriş-çıkış bileşenlerinin belirlenmesidir. Sürücünün taşıt üzerinde etkileşimde olduğu fiziksel büyüklükler göz önünde bulundurularak (Şekil 5.1) giriş (yol bilgisi, taşıt hızı, motor devir sayısı) ve çıkışlar (gaz pedalı, fren, vites, direksiyon açısı) seçilmiştir. Giriş ve çıkışlardan modeli en fazla etkileyen bileşen yol bilgisidir. Sürücünün algısına yakın yol bilgisinin modele verilmesi yüksek başarım için şarttır. Taşıtın önündeki tek nokta yerine, ilerisindeki bazı noktalardaki yol bilgisinin sürücüye verilmesi modele öngörü kazandırmıştır. Bu noktalar arasındaki ilişkinin de mesafe yerine zaman olarak alınması, gerçek sürücüde olduğu gibi taşıt hızlandığı zaman daha uzaktaki, yavaşladığı zaman ise daha yakındaki yol verilerinin dikkate alınmasını sağlamıştır. Buradan, kısıtlamalar azaltıldığı ve gerçeklik artırıldığı ölçüde iyi bir model çıkarılabileceği sonucuna varılmıştır. Çıkarılan sürücü modelini diğer çalışmalardan ayıran önemli bir özellik, yol bilgisinin sürücüye dönme açısı olarak verilmesidir.

Modellenecek sistemin karmaşıklığının ve doğrusal olmayışının yanında sürücüye öngörü katabilmek için “casual olmayan” (transfer fonksiyonu paydasının derecesi

payın derecesinden küçük olan sistem) bir model yapısı oluşturulmuştur (Şekil 5.7). Yani t anındaki çıkış t+λ'daki girişten etkilenmektedir. Literatürdeki insan operatörleri de hep sistemin tersi gibi etki edecek şekilde tasarlanmıştır. Örneğin, ikinci mertebe bir sistemi süren operatörün transfer fonksiyonunda iki tane sıfırı vardır.

Bu çalışmada, öncelikle doğrusal parametrik model çıkarılmıştır. Doğrusal modellerle çalışmanın önemli bir avantajı vardır. Geçmiş giriş ve çıkışların o anki çıkış üzerindeki etkisi çok açık bir şekilde görülebilir. Yapay sinir ağlarında ise tam tersi bir durum söz konusudur. Kara-kutu modelleme tekniği olarak da bilinen YSA'nın, giriş ile çıkış arasında kurduğu ilişki açık değildir. YSA ile performansı yüksek modeller çıkarılabilmesine rağmen sistemin çalışması hakkında bilgi sahibi olunamaz. Doğrusal modelin basitlik ve açıklık gibi avantajlarının yanında karmaşık sistemleri modelleyememesi gibi bir dezavantajı vardır. Sürücü için çıkarılan ARX model de yetersiz kalmıştır (kararsız model). Ancak doğrusal modelin doğrusal olmayan modelden önce mutlaka denenip, yeterli performans sağlanıyorsa doğrusal olmayan modelle hiç uğraşılmaması gerekir.

En küçük kareler yöntemi ile tek seferde bulunan doğrusal model parametreleri maliyet fonksiyonunu en küçültmekte ama sistemi kararsız yapmaktadır. Aynı YSA'da olduğu gibi iteratif algoritmalarla parametre uzayında başka noktalar bulunup model kararlı yapılmıştır. Fakat çıkarılan kararlı doğrusal model Tablo 5.2'de verilen en kötü YSA modellerinden bile daha düşük performansa sahiptir. Dikkat edilirse, doğrusal model çıkarılırken yapılan veriler üzerindeki ön işlemler, YSA ile model elde ederken de aynı şekilde uygulanmıştır. Buna rağmen elde edilen doğrusal modelin başarımı YSA modellerin başarımının çok altındadır.

Yapay sinir ağları ile sürücü modellenmesinde yeterli bağımsız parametre sayısına ulaşıldığında istenen başarım elde edilmiştir. Buradan varılabilecek bir sonuç; gerçekte başarımı getirenin saklı katmandaki nöron sayısı değil, bağımsız parametre sayısı olduğudur. Regresyon vektöründe farklı mertebeler ele alınırsa, bağımsız parametre sayısını sabit tutmak için saklı katmandaki nöron sayısını da değiştirmek gerekir. Herhangi bir modelleme işlemi için optimum bağımsız parametre sayısının belirtilmesi, optimum nöron sayısının verilmesinden çok daha anlamlıdır.

İyi bir model, sistemi olabildiğince yansıtmakla birlikte mümkün olduğunca az parametre sayısına sahip olmalıdır. Parametre sayısı arttırıldıkça modelin performansı artar ancak genelleştirme yeteneği düşer. Amaç mümkün olan en az parametre ile en iyi genelleştirmeyi veren modeli bulmaktır. Saklı katmanda altı nöron kullanılmasıyla (142 bağımsız parametre) optimum model performansı elde edilmiştir. Aynı zamanda, bir insan operatör (taşıt sürücüsü) için çıkarılan modelin bir başka operatör için de oldukça iyi genelleştirme yapabilmesi dikkat çekici önemli bir unsurdur (Tablo 5.2). Bunun nedeni verilerin zengin veri olmasıdır. Zengin veri kullanıldığında model, veri kaynağından bağımsız olarak sürücünün tüm dinamik karakteristiklerini yansıtır. Diğer önemli bir unsur ise, simülasyon şartları altında çok az nöron sayısında bile YSA modelinin kararlı kalmasıdır.

Çalışmanın ileriki aşamalarında yapay sinir ağlarının sağladığı avantaj ile giriş-çıkış bileşenleri değiştirilmeden farklı dinamikler ağırlıklarda saklanabilir. Sürücü modellemesinde en önemli kısmı giriş-çıkış bileşenlerinin bulunması oluşturur. Yapılan çalışmalardan da görülebileceği gibi tek bir optimum giriş-çıkış bileşeni yoktur. Örneğin düz bir yolda sürüş ile park etme birbirinden çok farklı senaryolardır, farklı dinamikler içerirler ve farklı giriş-çıkış bileşenlerinin kullanılmasını gerektirirler. YSA'nın kullanılmasıyla farklı senaryolar için gerekli bilgi değiştirilen ağırlıklarda saklanabilir. Böylelikle daha önceki çalışmalarda yapıldığı gibi değişik giriş-çıkış bileşenlerinin aranmasına gerek kalmaz. Şu ana kadar çıkarılmış modeller çoğunlukla tek çıkışlı, kısıtlamaları olan, gerçeklikten oldukça uzak sürücü modelleridir. Yanal ve doğrusal kontrolü birleştiren bu çalışmadaki sürücü modeli, YSA'nın belirtilen özelliğini kullanarak farklı dinamikleri içeren senaryoları da kapsayabilecek şekilde genişletilebilir.

KAYNAKLAR

Abonyi, J. , 2003. Fuzzy Model Identification for Control, Boston; Birkhäuser.

Abonyi, J. , Feil, B. , Szeifert, F. , 2004. Model order selection of nonlinear input-output models-a clustering based approach, *Journal of Process Control*, **14**, 593-602.

An, P.E. , Brown, M. , Harris, C.J. , 1994. On real time driver modeling and vehicle guidance within Prometheus, *IFAC Transportation Systems Tianjin*, 91-96.

An, P.E. , Harris, C.J. , 1996. An Intelligent Driver Warning System for Vehicle Collision Avoidance, *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, **26**(2), 254-261.

Apel, A. , 1997. Modellierung des Fahrerverhaltens bei Laengs- und Querregelung von Pkw, *Dissertation*, TU Braunschweig.

Arslan, H. , 1999. Yapay Sinir Ağları ile Robotlarda Hareket Kontrolü, *Doktora Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

Atabay, O. , 2004. Sürücü-Taşıt-Çevre etkileşim etütlerine yönelik bir sürüş simülatörü geliştirilmesi, *Doktora Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

Babuska, R. , 1998. Fuzzy Modeling for Control, Boston; Kluwer Academic Publishers.

Bomberger, J.D. , Seborg D.E. , 1998. Determination of model order for NARX models directly from input-output data, *J. Proc. Cont,* **8**(5), 459-468.

Chen, L.K. , Ulsoy, A.G. , 1999. Driver Model Uncertainty, *American Control Conference*, **1,** 714 - 718.

Efe M.O. , Kaynak, O. , 2000. Yapay Sinir Ağları ve Uygulamaları, Bogaziçi Universitesi Basımı.

Fenton, R.E. , 1994. IVHS/AHS: Driving into the Future, *IEEE Control Systems Magazine*, **14**(6), 13-20.

Fletcher, R. , 1971. A modified Marquardt subroutine for nonlinear least squares, *Technical Report AERE-R6799, Harwell Report*.

Fujioka T. , Takubo N. , 1991. Driver model obtained by neural network system. *JSAE Review*, **12**(2), 82-85.

Guldner, J. , Tan H. S. , Patwardhan S. , 1996. Analysis of Automatic Steering Control for Highway Vehicles with Look-down Lateral reference Systems, *Vehicle System Dynamics,* **26,** 243-269.

Hagan, M.T. , 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. , Beale, M. , 1995. Neural Network Design, PWS Publishing, Boston.

Ikonen, E. , Najim, K. , 2002. Advanced Process Identification and Control, New York; Marcel Dekker.

Jang, J-S.R. , Sun, T. , Mizutani, E. , 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, New Jersey.

Kornhausser, A.L. , 1991. Neural network approaches for lateral control of autonomous highway vehicles, *Proceedings, Vehicle Navigation & Information Systems*.

Lubin, J.M. ve diğ. , 1992. Analysis of a neural network lateral controller for an autonomous road vehicle, *Proceedings, Future Transportation Technology Conference and Exposition*.

MacAdam, C. , Johnson, G. E. , 1996. Application of Elementary Neural Networks and preview Sensors for Representing Driver Steering Control Behaviour, *Vehicle System Dynamics,* **25**(1), 3-30.

Mitschke, M. , 1996. Verifikation des Fahrermodells für kritische Fahrsituationen (3. Fahrerverhalten bei normaler Kurvenfahrt). Bericht Nr. 744. Institut für Fahrzeugtechnik, Technische Universitaet Braunschweig.

More, J.J. , 1977. The Levenberg-Marquardt algorithm: implementation and theory. Numerical analysis, lecture notes in mathematics 630, Springer-Verlag.

Nash, J.C. , 1977. Minimizing a nonlinear sum of squares function on a small computer, *J. Inst. Math. Appl.,* **19**, 231-237.

Neusser, S. , Nijhuis, J. , Spaanenburg, L. Hoefflinger, B. , Franke, U. , Fritz, H. , 1993. Neurocontrol for Lateral Vehicle Guidance, *IEEE Micro*, **13**(1), 57-66.

Nguyen, D. , Widrow, B. , 1990. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, *Proc. of the Int. Joint Conference on Neural Networks,* **3,** 21–26.

Norgaard, M. , Ravn O. , Poulsen N.K. , Hansen L.K. , 2000. Neural networks for modeling and control of dynamic systems : a practitioner's handbook, London; Springer.

Ogata, K. , 1967. State space analysis of control systems, Englewood Cliffs, N.J., Prentice-Hall.

Pilutti, T. , Ulsoy, G. , 1999. Identification of Driver State for Lane-Keeping Tasks, *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, **29**(5), 486-502.

Riedel, A. , Wurster, U. , 1998. Fahrermodelle in der Praxis, *Automotive Engineering Partners*, **3**, 88-91.

Shaw, I.S. , 1993. Fuzzy model of a human control operator in a compensatory tracking loop, *Int. J. Man-Machine Studies*, **38**, 305-332.

Sandberg, I.W. , Lo J.T. , Fancourt C.L. , Principe J.C. , Katagiri S. , Haykin S. , 2001. Nonlinear dynamical systems : feedforward neural network perspectives, A Wiley-Interscience Publication.

Shim, J.S. , Yoon, Y.Y. , Heo, S.J. , Yoo, Y.M. , 1995. Closed-loop simulation of a vehicle system with an artificial driver, *Mech. Struct. & Mach.*, **23**(1), 87-113.

Söderström, T. , Stoica, P. , 1989. System Identification, New York; Prentice Hall.

Tsoukalas, L.H. , Uhrig, R.E. , 1997. Fuzzy and Neural Approaches in Engineering, New York; Wiley.

Yıldızdoğan, M. , 1995. Yapay Sinir Ağları ile Sistem Tanıma, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

EKLER

EK A

BELİRLİLİK VE YARI-BELİRLİLİK

Kuadratik formda yazılmış x'Ax'de A matrisi Pozitif Belirlidir, eğer **(Ogata, 1967)**

$x'Ax > 0$ $x \neq 0$ için
$x'Ax = 0$ $x = 0$ için

Pozitif Yarı- Belirli

$x'Ax \geq 0$ $x \neq 0$ için
$x'Ax = 0$ $x = 0$ için

Negatif Belirli

$x'Ax < 0$ $x \neq 0$ için
$x'Ax = 0$ $x = 0$ için

Negatif Yarı-Belirli

$x'Ax \leq 0$ $x \neq 0$ için
$x'Ax = 0$ $x = 0$ için

POZİTİF BELİRLİLİK

$$a_{11} > 0 \ , \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0 \ , \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} > 0 \ , \quad \dots$$

ve

$$\det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} > 0 \quad , \quad a_{ij} = a_{ji}$$

POZİTİF YARI- BELİRLİLİK

$$a_{11} \geq 0 \ , \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \geq 0 \ , \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \geq 0 \ , \quad \dots$$

ve

$$\det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = 0 \quad , \quad a_{ij} = a_{ji}$$

ÖZGEÇMİŞ

İsmail İlker DELİCE 1978 yılında İstanbul'da doğdu. 1993-1997 yılları arasında Kadir Has Lisesi'nde okuduktan sonra 1997 yılında İstanbul Teknik Üniversitesi, Makina Mühendisliği Bölümünü kazandı. Mezun olduğu 2002 yılında İTÜ Fen Bilimleri Enstitüsü Makina Teorisi, Sistem Dinamiği ve Kontrol Programı'nda yüksek lisansa başladı. DELİCE, şu anda araştırma görevlisi olarak çalışmaktadır.