İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

PROTEİN İŞLEV KESTİRİMİNDE YAPISAL BİLGİNİN KATKISI VE DİZİ GEÇİŞ OLASILIKLARI İLE PEPTİT SINIFLANDIRMA

YÜKSEK LİSANS TEZİ

Eser AYGÜN

Anabilim Dalı : **BİLGİSAYAR MÜHENDİSLİĞİ**

Programı : **BİLGİSAYAR MÜHENDİSLİĞİ**

OCAK 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

PROTEİN İŞLEV KESTİRİMİNDE YAPISAL BİLGİNİN KATKISI VE DİZİ GEÇİŞ OLASILIKLARI İLE PEPTİT SINIFLANDIRMA

YÜKSEK LİSANS TEZİ
Eser AYGÜN
(504061512)

Tezin Enstitüye Verildiği Tarih : 29 Aralık 2008
Tezin Savunulduğu Tarih : 22 Ocak 2009

Tez Danışmanı : Doç. Dr. Zehra ÇATALTEPE

Diğer Jüri Üyeleri : Yrd. Doç. Dr. Şule GÜNDÜZ ÖĞÜDÜCÜ (İ.T.Ü.)

Doç. Dr. Uğur SEZERMAN (S.Ü.)

OCAK 2009

ÖNSÖZ

Bu çalışma, Doç. Dr. Zehra Çataltepe ile birlikte çalıştığımız TÜBİTAK destekli bir biyoenformatik projesinin ürünüdür. Her şeyden önce, bana bu projede çalışma fırsatı veren, yoluma ışık tutan ve hiçbir konuda yardımını esirgemeyen anlayışlı danışmanım Zehra Çataltepe'ye teşekkür ederim.

Yayınladığı dergi ve kitaplarla ufkumu genişleten, üniversiteye girmemden yüksek lisansımı tamamlamama kadar pek çok konuda beni maddî olarak destekleyen TÜBİTAK'a da bir teşekkür borçluyum.

Ayrıca, çalışmam boyunca yanımda oldukları için çalışma arkadaşlarım Aslı Filiz, Caner Kömürlü, Kenan Kule ve Yusuf Yaslan'a; çalışmalarını bizimle paylaşan ve gerektiğinde sorularımıza cevap veren Dr. B. John Oommen'a; tezin hazırlanışı ve yazımı sırasında akademik konuları danışabildiğim nadir insanlardan, arkadaşım ve meslektaşım Amaç Herdağdelen'e teşekkür ederim.

Son olarak, sevgileriyle, özverileriyle ve hayata bakışlarıyla bana sürekli huzur, güven ve merak aşılayan aileme, Gönül Aygün, Hasan Aygün ve Fatih Aygün'e –bir kere daha– teşekkür ederim.

Ocak 2009

Eser AYGÜN

Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

PROTEİN İŞLEV KESTİRİMİNDE YAPISAL BİLGİNİN KATKISI VE DİZİ GEÇİŞ OLASILIKLARI İLE PEPTİT SINIFLANDIRMA

ÖZET

Biyolojik dizi analizi, nükleotid ve amino asit dizilerinin evrimsel, yapısal ve işlevsel özelliklerini ortaya çıkarmayı amaçlar. Biyolojik dizilerin analizinde kullanılan pek çok araç vardır; ikili hizalama algoritmaları, çoklu hizalama algoritmaları, gizli Markov modelleri, motifler, vb. Bu çalışma genel olarak ikili hizalama algoritmalarına yoğunlaşmıştır, ve özel olarak şunları hedefler:

- Standart ikili hizalama algoritmalarının bir derlemesini sunmak,
- Oommen ve Kashyap'ın tanımladığı dizi geçiş olasılığını bir biyolojik dizi benzerlik ölçütü olarak tanıtmak,
- Yapısal bilginin protein işlev kestiriminin başarısını nasıl arttırdığını göstermek,
- Oommen ve Kashyap'ın dizi geçiş olasılığını, iki peptit sınıflandırma problemi üzerine standart dizi benzerlik ölçütleriyle kıyaslamak,
- Gereken dizi analiz araçlarını bir bilgisayar yazılımı olarak gerçeklemek.

Çalışmanın deneysel kısmında umut veren sonuçlar elde edilmiştir. Birinci deney sonuçları, ikincil yapı dizilerini amino asit dizisi hizalamalarıyla birlikte kullanmanın moleküler işlev kestirim başarısını arttırdığını açıkça ortaya koymuştur. Buna karşılık kestirilmiş ikincil yapıların kestirime herhangi bir katkısının olmadığı gözlenmiştir. Bu sonuç, yapısal bilginin katlanma tanıma başarısını arttırdığını ortaya koyan önceki çalışmaların bulgularıyla uyumludur ve onların bir uzantısıdır.

İkinci olarak, dizi geçiş olasılığı ölçümünün biyoloji alanındaki ilk uygulaması iki peptit sınıflandırma problemi üzerinde gerçekleştirilmiştir. Dizi geçiş olasılıkları, sınıflandırıcıya sunulan nitelikler olarak, standart genel hizalama puanları ile kıyaslanmıştır. Sınıflandırma başarısı ölçümleri, dizi geçiş olasılıklarının genel hizalama puanlarından çok daha iyi nitelikler sağladığını şüpheye yer bırakmayacak şekilde ortaya koymuştur. Önerilen yöntem ayrıca aynı veri kümeleri üzerinde uygulanmış önceki yöntemlerin neredeyse hepsinden daha başarılı olarak genel kabul görmüş peptit benzerlik ölçütü olmaya aday olduğunu kanıtlamıştır.

IMPROVEMENT OF PROTEIN FUNCTION PREDICTION USING STRUCTURAL INFORMATION AND PEPTIDE CLASSIFICATION USING SYNTACTIC TRANSITION PROBABILITIES

SUMMARY

Biological sequence analysis deals with nucleotide and amino acid sequences, aiming to expose their evolutionary, structural and functional properties. There are many tools that help the analysis of biological sequences, such as pairwise alignment algorithms, multiple alignment algorithms, hidden Markov models, motifs, etc. This study focuses generally on the pairwise alignment algorithms, and it intends specifically

- to provide a review of well known pairwise alignment methods,
- to introduce the syntactic transition probability of Oommen and Kashyap as a biological sequence similarity metric,
- to demonstrate how the structural information improves protein function prediction,
- to compare syntactic transition probability of Oommen and Kashyap with standard sequence similarity metrics on two peptide classification problems,
- and to implement necessary sequence analysis tools as a computer software.

The outcomes of the experimental parts of this study are promising. First of all, the results clearly indicate that the use of secondary structure sequences along with amino acid sequence alignments improves molecular function prediction performance, while the use of predicted secondary structures does not. This conclusion extends the findings of the previous works, which state that the structural information indeed improves the fold recognition performance.

Secondly, the first biological application of –otherwise known– syntactic transition probability measurement is carried out on two peptide classification problems. Syntactic transition probabilities are compared with standard global alignment scores as being features fed into a machine learning classifier. The classification performance measurements undoubtedly proved that syntactic transition probabilities are much better features than global alignment scores for peptides. The proposed method also outperformed almost all of the previously reported methods that were applied on the same data sets, making the syntactical transition probabilities a candidate for the state-of-the-art similarity metric of peptides.

1. GİRİŞ

Biyoenformatik çalışmaları, bilim insanlarının cevap arayışının son ürünlerindendir. Bilgisayarlı biyoloji olarak da anılabilecek biyoenformatik, biyoloji sorunlarına bilgisayar kullanarak çözüm üretilen her çalışma alanını kapsar. Bu alanların en önemlilerinden bir tanesi biyolojik dizi analizidir. Biyolojik dizi analizi, biyolojide önemi çok büyük olan iki çeşit molekülü inceler: Nükleotit dizilerinden oluşan DNA (*deoksiribonükleik asit*) molekülleri ve amino asit dizilerinden oluşan protein molekülleri. Her iki çeşit molekül de, tıpkı bilgisayarda saklanan katarlar (*string*) gibi, sonlu bir kümeden gelen yapıtaşlarının belli bir sırayla yan yana dizilmesiyle elde edilir. DNA molekülleri, canlıların fizyokimyasal davranışlarını, yani genetik kodunu saklayan bir bilgi bankasıdır. Protein molekülleri ise, bahsedilen fizyokimyasal davranışları gerçekleyen araçlardır. Biyolojik dizi analizi, bu moleküllerin evrimsel, yapısal ve işlevsel özelliklerini, onları yalnızca basit birer dizi olarak kabul ederek ortaya çıkarmayı amaçlar.

Biyolojik dizi analizi, az miktarda nesneyi ele alıyor olsa da geniş bir konudur. İkili dizi hizalama (*pairwise sequence alignment*), çoklu dizi hizalama (*multiple sequence alignment*), veritabanında hizalanma arama, gizli Markov modelleri (*hidden Markov model*) ile dizi analizi, dizi profilleri ve motifler bu konunun altında sayılabilecek başlıklardan birkaçıdır [1]. Bu çalışma, kabaca ikili dizi hizalama[1] üzerinedir. Daha kesin olarak söylemek gerekirse, çalışmanın ana hedefleri şöyle sıralanabilir:

1. Literatürdeki ikili dizi hizalama yöntemlerinin derlenmesi,

2. Şu âna kadar biyoloji alanına uygulanmamış bir dizi analiz yönteminin tanıtılması,

[1] İkili hizalama dışındaki hizalama tekniklerine bu yazıda değinilmeyecektir. Bu yüzden yazının geri kalanında "hizalama" sözcüğü ikili dizi hizalamaya karşılık olarak kullanılacaktır.

3. Proteinlerin temel yapısal özelliklerinin bilinmesinin protein işlev kestirimine katkısının ölçülmesi,

4. Önerilen yeni dizi analiz yönteminin peptit sınıflandırma problemi üzerinde denenmesi,

5. Ele alınan tüm dizi analiz araçlarının bilgisayarda gerçeklenmesi ve bir yazılım olarak sunulması.

Çalışmada bahsedilecek biyolojik dizi analizi araçları, iki farklı problem üzerinde işletilecektir. Bu problemlerin ilki protein işlev kestirimidir. Proteinler, vücutta binlerce değişik iş gören karmaşık moleküllerdir. Bir proteinin belli bir işi görüp görmediğine laboratuvarda karar vermek, oldukça fazla iş gücü ve zaman gerektirir. Oysa bilinen proteinlerin sayısı, genom araştırmalarının hızlanmasının bir sonucu olarak üstel bir hızla artmaktadır. Bilgisayar üzerinde çalışan işlev kestirim araçları, oluşan iş gücü boşluğunu doldurmaya, gereken hızı sağlamaya adaydır.

Bu çalışmanın deneysel kısmının birinci aşamasında amaç, amino asit dizileri üzerinden yapılan bir protein işlev kestiriminin, ikincil yapıların veya kestirilmiş ikincil yapıların da hesaba katılmasından nasıl etkileneceğini ölçmektir. Bunun için Wallqvist ve arkadaşlarının [2] önerdiği şekilde; ikincil yapı bilgisi, protein dizisi hizalama algoritmalarıyla bütünleştirilecektir. Ardından, ikincil yapı bilgisini farklı ağırlıklarla işin içine katarak başarı ölçümleri yapılacaktır. Sonuçta, kestirilmiş ikincil yapının değil fakat, gerçek ikincil yapının 0.25 oranında hizalamaya dâhil edilmesinin protein işlev kestirimi başarısını istatistikî olarak anlamlı derecede iyileştirdiği gösterilecektir.

Çalışmanın deneysel kısmının ikinci aşaması, Oommen-Kashyap geçiş olasılıklarının [3] biyolojik veriler üzerindeki başarısının ölçülmesi üzerinedir. Bunun için seçilen problemler yine birer işlev kestirim problemidir. Ancak bu sefer işlevleri kestirilecek olan amino asit zincirleri protein denemeyecek kadar kısadır. Biyolojide –farklı tanımları olmasına rağmen– 50 amino asitten daha az yapıtaşı içeren amino asit dizilerine *peptit* denir. Peptitler, proteinlerin içindeki bölgeler olabileceği gibi, kendi başına işlev gören moleküller de olabilir ve tıpkı proteinler gibi kendilerine has işlevleri bulunabilir: Peptitler belli enzimlerden

etkilenebilirler veya belli elementleri kendilerine bağlayabilirler. Peptitlerin bu etkinlikleri onları biyoloji, tıp, farmakoloji, patoloji ve nanoteknoloji alanlarında çalışan araştırmacılar için önemli kılar [4, 5, 6, 7, 8].

Peptit sınıflandırmanın önemi ve Oommen-Kashyap geçiş olasılıklarının peptitler üzerinde çalışmaya uygun olabileceğinin bugüne kadar fark edilmemiş olması, ikinci aşamanın motivasyonudur. Bu aşamada; daha önce peptit sınıflandırma üzerine yapılmış çalışmaların bir taraması yapılacak, Li Liao ve William S. Noble tarafından farklı bir problem için ortaya atılmış bir sınıflandırma yöntemi [9] geçiş olasılıklarını kullanacak şekilde peptit sınıflandırma problemine uyarlanacak, uyarlanan yöntem iki farklı veri kümesi üzerinde sınanacak ve elde edilen sonuçlar standart bir hizalama algoritmasının ürettiği sonuçlarla karşılaştırılacaktır. Neticede, önerilen yeni yöntemin hem standart hizalama algoritmalarından hem de şimdiye kadar aynı veri kümesi üzerinde denenmiş önceki yöntemlerden başarılı olduğu ortaya konacaktır.

2. BİYOLOJİK DİZİ ANALİZİ

Biyolojik diziler, nükleotit dizileri ve amino asit dizilerini kapsar. Nükleotit dizileri DNA moleküllerini, amino asit dizileri ise protein moleküllerini meydana getirir. Canlıların neredeyse tüm biyolojik etkinliklerini belirleyen proteinler, DNA moleküllerinin yorumlanması ile oluşturulurlar. Farklı DNA molekülleri farklı proteinlere, farklı proteinler ise farklı biyolojik etkinliklere yol açar. Canlıların dış görünüşleri, hareket tarzları, beslenme alışkanlıkları ve çoğalma biçimlerindeki çeşitlilik bu farklılıkların sonucudur: Bir canlıyı tanımlayan ve onu diğer canlılardan farklı yapan bilginin neredeyse tamamı doğrudan hücrelerindeki DNA moleküllerinde, dolaylı olarak da sentezledikleri proteinlerde saklıdır. *Biyolojik dizi analizi*, nükleotit dizileri (yani DNA'lar) ya da karşılık geldikleri amino asit dizileri (yani proteinler) arasındaki evrimsel, yapısal ve işlevsel ilişkileri ortaya çıkarmayı amaçlar [10].

2.1 Proteinlerin Evrimi

Bilindiği gibi, DNA moleküllerinin en önemli özelliklerinden birisi kendi kendini kopyalama yeteneğidir. Çeşitli enzimlerin yardımıyla ve yeterli hammadde ile bir DNA molekülü, kendisiyle aynı nükleotit dizisini içeren yeni bir DNA molekülü üretebilir. Bu sayede ait olduğu canlıya ait etkinlikleri tanımlayan bilgiyi bir hücreden bir yenisine taşıyabilir. Bu, canlıların çoğalmasını mümkün kılar. Gelgelelim, yine bilindiği gibi, DNA moleküllerinin kopyalanması her zaman kusursuz olmaz. Kopyalama sırasındaki kimi hatalar, oluşan yeni DNA molekülünün orijinal molekülden az da olsa farklı bir nükleotit dizisi içermesine yol açabilir. Benzer şekilde, mor ötesi ışınlar gibi dış etkiler mevcut bir DNA molekülünün yapısını etkileyip nükleotit dizilimini değiştirebilir. Nükleotit dizilimlerinin bu tür yollarla değişime uğramasına *mutasyon* adı verilir.

Açık ki; mutasyonlar yalnızca DNA moleküllerinin nükleotit dizilerinin değişmesiyle kalmaz. Değişen nükleotit dizisi kimi zaman farklı proteinlere, farklı proteinler de kimi zaman farklı biyolojik etkinliklere yol açar. Etkinliği değişen canlı bazen bu değişiklikler yüzünden temel işlevlerini yitirip ölür; bazen çoğalma işlevini yitirdiği için soyunu sürdüremez; bazen değişiklikten etkilenmeden eskisi gibi hayatını sürdürür. Nadiren ise, soyunu daha verimli bir şekilde sürdürmek adına yeni bir işlev kazanır. Soyun sürdürülmesini engelleyen değişiklikler ait oldukları canlı ile birlikte yok olurken, soyun sürdürülmesini engellemeyen ya da kolaylaştıran değişiklikler canlının çocukları üzerinden varlığını sürdürür. Nükleotit dizileri üzerinde saklanan bilginin bu şekilde kopyalanması, değişmesi ve elenmesi ile *biyolojik evrim* meydana gelir.

Nükleotit dizilerinin nesiller boyunca süren evrimi, mevcut proteinlerden yeni proteinler türemesiyle kendini gösterir. Aynı proteinden yola çıkıp farklı yollar izleyerek türemiş proteinlere, türeyiş sırasında yapılarının ya da işlevlerinin değişip değişmemiş olmasına bakmaksızın, *eş kökenli (homologous)* proteinler denir. Biyolojik dizi analizinin bir uygulama alanı, proteinlerin eş kökenliliğinin tespit edilmesidir.

2.2 Proteinlerin Yapıları

Biyolojik dizi analizinin diğer bir uygulama alanı, proteinlerin yapılarının benzerliğinin tespit edilmesidir. Proteinler sentezlendikleri andan itibaren tıpkı bir yay gibi kendi üzerlerine katlanırlar. Bu katlanışın ne şekilde olacağı proteinin içerdiği amino asitlerin karmaşık etkileşimlerine bağlıdır. Etkileşimlerin yerelliğine göre proteinlerin yapısı dört katmanda incelenir:

- Birincil yapı: Amino asitlerin hangi sırayla dizildiklerinin doğrudan sonucudur.
- İkincil yapı: Uzayda birbirine yakın amino asitlerin etkileşimiyle belirlenen yerel biçimlerin sonucudur.
- Üçüncül yapı: Zincir boyunca meydana gelen uzun ya da kısa mesafeli tüm etkileşimlerle belirlenen üç boyutlu biçimin sonucudur.

- Dördüncül yapı: Birden fazla zincire sahip proteinlerde zincirler arası etkileşimlerle belirlenen biçimin sonucudur.

Bu sıralamada üst katmanlara çıktıkça amino asit dizisinin belirleyiciliği azalırken proteinin özellikleri hakkında kazanılan bilgi miktarı artar. Proteinlerin amino asit dizilerinin tespiti ve temsili görece kolaydır. Canlılar toplam yirmi farklı amino asit üretirler; dolayısıyla birincil yapı yirmi harfli bir alfabenin sözcükleriyle temsil edilebilir (Çizelge 2.1).

İkincil yapının da sürekli tekrar eden örüntüleri vardır. Bunlar çoğunlukla DSSP (*Dictionary of Protein Secondary Structure*) [11] ile belirlenmiş sekiz harfle simgelenirler (Çizelge 2.2). Bu sayede ikincil yapı da sekiz harfli bir alfabenin sözcükleriyle temsil edilebilir.

Üçüncül yapı, ayrık alfabeli diziler ile ifade edilemeyecek kadar karmaşıktır. Bu yüzden atomların üç boyutlu konumları ile temsil edilir. Yine de üçüncül yapının daha kolay incelenebilmesi için, protein katlanışlarını temel bazı özelliklerine göre sınıflandıran hiyerarşiler oluşturulmuştur. Böylece biyologlar proteinin kesin üçüncül yapısını bilmek yerine hiyerarşideki yerini bilmekle yetinebilirler. En çok başvurulan hiyerarşiler SCOP[12], CATH[13] ve FSSP[14] hiyerarşileridir. Bunlardan FSSP tamamen üç boyutlu hizalama algoritmalarıyla, otomatik olarak oluşturulmuştur. CATH kısmen elle, kısmen otomatik; SCOP ise tamamen elle oluşturulmuştur.

Çizelge 2.1: Proteinleri oluşturan yirmi amino asit.

Adı	Harf Kodu
Alanin	A
Arginin	R
Asparagin	N
Aspartik asit	D
Sistein	C
Glutamik asit	E
Glutamin	Q
Glisin	G
Histidin	H
İzolösin	I

Adı	Harf Kodu
Lösin	L
Lizin	K
Metiyonin	M
Fenilalanin	F
Prolin	P
Serin	S
Treonin	T
Triptofan	W
Tirozin	Y
Valin	V

Araştırmacıların tercihlerine göre elle oluşturulduğundan, bilgisayarlı kestirimlerde çoğunlukla SCOP hiyerarşisi temel alınır. Bir proteinin üçüncül yapısının SCOP gibi bir hiyerarşideki yerini belirleme işine *katlanış tanıma (fold recognition)* adı verilir. Katlanış tanıma, biyolojik dizi analizinin sık sık başvurulduğu başka bir araştırma konusudur.

2.3 Proteinlerin İşlevleri

Proteinler işlevleri yüzünden vardır ve araştırmacılar onları çoğu zaman yalnızca işlevleri yüzünden araştırırlar. Zararlı bir proteinin baskılanması, yararlı bir proteinin tetiklenmesi ya da tamamen özel amaçlar için yeni bir proteinin tasarlanabilmesi için proteinlerin işlevlerini verimli bir şekilde belirleyebiliyor olmak gerekir. Bu verim bilgisayarlar sayesinde sağlanabilirse zahmetli ve maliyetli laboratuvar çalışmalarına olan ihtiyaç azalmış olur.

Proteinler çok çeşitli işler görürler. Tıpkı üçüncül yapıda olduğu gibi, olası tüm işlevleri tek tek saymak da mümkün değildir ve bu yüzden araştırmacılar işlev haritaları çıkarmışlardır. GO[15] (Gene Ontology) genel amaçlı ve çok sık başvurulan bir işlev haritasıdır. GO ontolojisinin içerisinde proteinler üç farklı şekilde etiketlenir: (i) katıldıkları biyolojik etkinliklere göre (*biological process*), (ii) hücrede yer aldıkları bileşenlere göre (*cellular component*) ve (iii) moleküler seviyedeki işlevlerine göre (*molecular function*). Her üç alanda da, bir diğerini kapsayan ya da bir diğeri tarafından kapsanan pek çok terim bulunur. Bir terim, kendinden üst seviyedeki terimlere çocuğu olma ("*is-a*") veya parçası olma ("*has-a*") ilişkisiyle bağlanır. İki tür ilişki de çoktan-çokadır; bir terim, iki veya

Çizelge 2.2: DSSP ile belirlenmiş sekiz ikincil yapı türü.

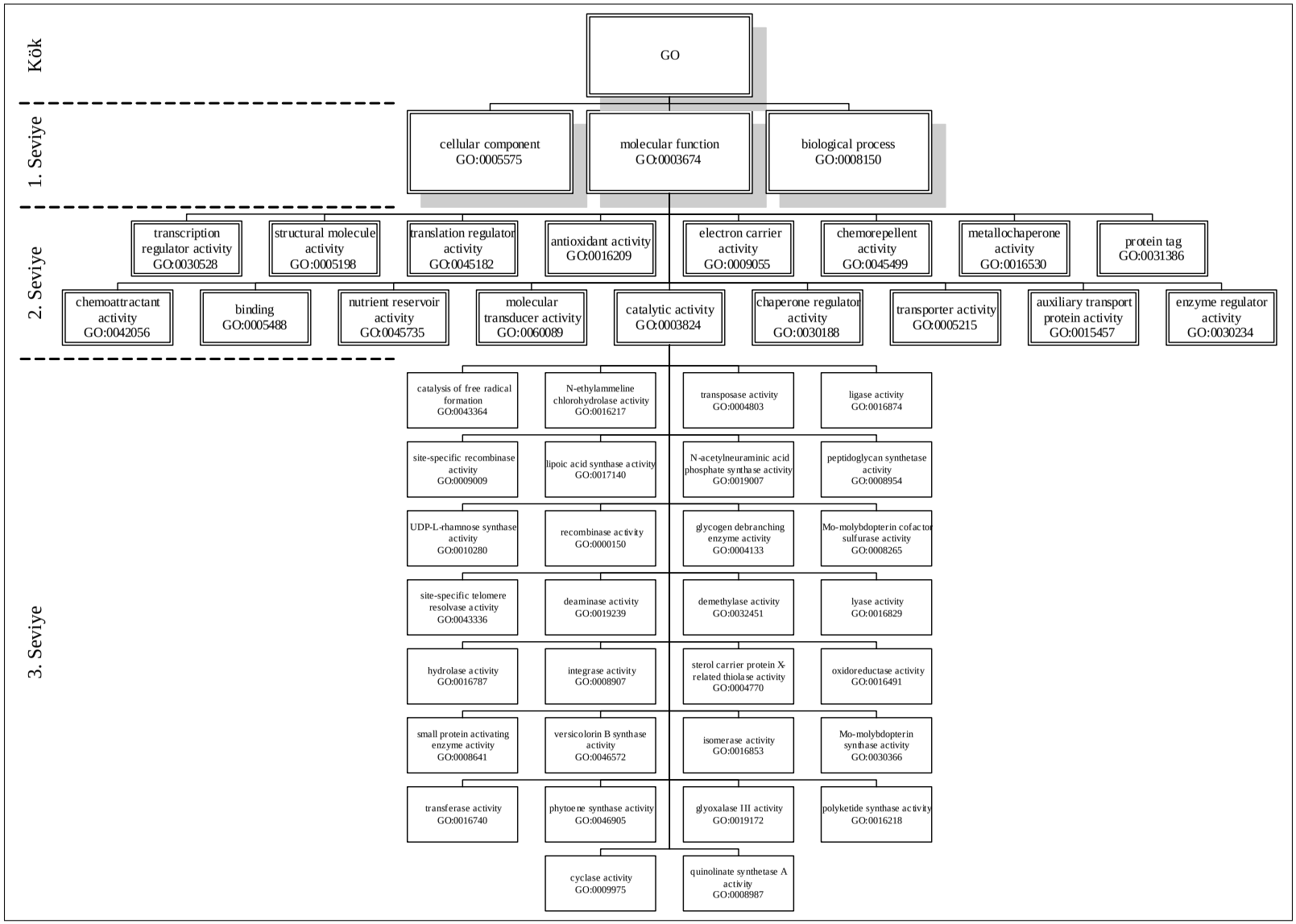
Biçimi	Harf kodu
3_{10} sarmalı	G
α sarmalı	H
π sarmalı	I
Hidrojen bağlı dönüş	T
Geniş sarım	E
İzole β köprüsü	B
Dirsek	S
Döngü ya da sıra dışı biçim	(boşluk)

daha fazla terimin çocuğu veya parçası olabilir. GO ontolojisinden bir parça Şekil 2.1'de görülebilir.

Örneğin HIV-1 proteaz enzimi, biyolojik etkinlik olarak "viral enfeksiyon başlatma" ve "provirüs entegrasyonu" etiketleriyle; hücrede yer aldığı bileşene göre ise "sitozol" etiketiyle etiketlenmiştir. Bu etiketlerden "viral enfeksiyon başlatma", "viral çoğalma etkinliği" etiketi tarafından, o da "çoğalma etkinliği" etiketi tarafından kapsanır. HIV-1 proteazın diğer etiketleri de daha genel etiketlerin altında bulunur.

Bilgisayarlı biyolojide *işlev kestirimi (function prediction)*, çoğu zaman proteinin hangi GO terimleriyle etiketlendiğine karar vermek demektir. Eğer bir işlevin ortaya çıkmasına yol açan tüm şartlar doğrudan bilinebilseydi, bir proteinin işlevine, dolayısıyla GO etiketlerine, yalnızca yapısına bakarak karar verilebilirdi. Ancak bu çoğu zaman mümkün olmaz. Onun yerine işlevi belirlenmiş proteinlerle işlevi merak edilen proteinler arasındaki yapısal benzerliklere bakılarak tahminde bulunulur. Çünkü yapıları benzer olan iki proteinin işlevleri de çoğu zaman benzerdir [16].

Proteinler hakkındaki bu gözlem çok önemlidir: Evrimsel, yapısal ve işlevsel benzerliklerin her biri diğerlerini destekler. Buna ek olarak; birincil yapı, ikincil yapı ve üçüncül yapı hakkında bilgi verir. Dolayısıyla sadece birincil yapıların, yani amino asit dizilerinin benzerliklerinden yola çıkarak hem evrimsel, hem yapısal, hem de işlevsel ilişkileri belirlemek mümkündür. Biyolojik dizi analizini önemli kılan budur. Bir sonraki bölümde biyolojik dizi analizinin dizi benzerliklerini hesaplama konusunda ne tür yaklaşımlar önerdiği incelenecektir.



Şekil 2.1: GO ontolojisinden bir bölüm. Yalnızca *molecular function* ve *catalytic activity* terimlerinin çocukları görünüyor.

3. DİZİ BENZERLİĞİ VE DİZİ HİZALAMA

Bu bölümde iki dizi arasındaki benzerliğin diziler arasındaki farklı ilişkileri ortaya çıkarmak için ne şekillerde ölçülebileceği incelenecek. Bunun için öncelikle dizi[1] kavramının ve iki dizi üzerinde çalışan fonksiyonlar olan dizi benzerliklerinin formel bir tanımı yapılacak. Bölümün geri kalanında bu çalışmada ele alınmış olan benzerlik ölçme yöntemleri tarif edilecek.

3.1 Dizi, Dizi Benzerliği ve Dizi Hizalama

Diziler doğal yazı dilindeki sözcüklere benzer. Boş olmayan sonlu bir Σ kümesinin elemanlarından seçilmiş sonlu sayıdaki karakter belli bir sırayla yan yana geldiğinde bir dizi tanımlar. Örneğin $\Sigma = \{\mathtt{A},\mathtt{B}\}$ ise $\mathtt{ABBA}$, Σ alfabesi üzerinde geçerli bir dizidir. Diziyi oluşturan karakter sayısına dizinin uzunluğu denir. n uzunluklu genel bir U dizisi $u_1u_2\ldots u_n$ şeklinde, uzunluğu 0 olan dizi ise ε şeklinde gösterilir. Σ^* gösterimi, Σ alfabesi ile oluşturulabilecek tüm dizilerin kümesini temsil eder. Örneğin $\Sigma = \{\mathtt{A},\mathtt{B}\}$ ise, Σ^*, $\{\varepsilon,\mathtt{A},\mathtt{B},\mathtt{AA},\mathtt{AB},\mathtt{BA},\mathtt{BB},\mathtt{AAA},\ldots\}$ şeklinde sayılabilen sonsuz elemanlı bir kümedir.

Dizi benzerliği, basitçe, iki diziyi bir adet gerçek sayıya bağlayan bir fonksiyon olarak tanımlanabilir. Formel olarak, $f : \Sigma^* \times \Sigma^* \mapsto \mathbb{R}$ biçimdeki bir fonksiyona Σ alfabesi üzerindeki diziler için bir *dizi benzerliği* denir. Dizi benzerlikleri için metrik uzaydaki üçgen eşitsizliği gibi kısıtlar tanımlanması beklenebilir, ancak buna gerek yoktur. Bir dizi benzerliğinin, ortaya çıkarmak istediği ortak özelliğin kuvvetiyle artmasını beklemek yeterlidir.

Bir çift dizideki her bir karakteri dizi içindeki sırasını değiştirmeden karşıdaki dizinin bir karakterine ya da boşluklara denk getirecek şekilde eşleştirmeye *hizalama (alignment)* denir. Bir karakterin diğer dizideki bir karakterle eşleşmesi, *değiştirme* işlemine, bir karakterin boşluk ile eşleşmesi ise *silme* veya *ekleme*

[1] Bu yazıda dizi (*sequence*) olarak anılan kavram, matematikteki katar (*string*) kavramına denk düşmektedir. Biyoloji metinlerine uyumluluk açısından bu adlandırma tercih edilmiştir.

işlemlerine karşılık düşer. Bir Σ alfabesi üzerinde tanımlı $U = u_1u_2\ldots u_m$ ve $V = v_1v_2\ldots v_n$ dizileri hizalandığında, $\Sigma' = \Sigma \cup \{-\}$ alfabesinde tanımlı $U' = u'_1u'_2\ldots u'_z$ ve $V' = v'_1v'_2\ldots v'_z$ dizileri elde edilir ($z \geq m$ ve $z \geq n$). U' ve V' dizilerinin U ve V dizilerinden farkı, uzunlukları z'ye eşit olana kadar başlarına, sonlarına veya aralarına boşluk karakteri ($-$) eklenmiş olmasıdır. Öyle ki; diğer karakterlerin sırasını korurken boşlukları ortadan kaldıran bir $D : \Sigma'^* \mapsto \Sigma^*$ fonksiyonu tanımlanırsa $U = D(U')$ ve $V = D(V')$ yazılabilir. Örneğin $\Sigma = \{\texttt{A}, \texttt{B}\}$ alfabesinde tanımlı $U = \texttt{ABBA}$ ve $V = \texttt{BABA}$ dizilerinin bir hizalanışı şöyledir:

$$\begin{aligned} U' &= \texttt{ABBA-} \\ V' &= \texttt{BA-BA}. \end{aligned}$$

Hizalama algoritmaları, verilen iki diziyi (*genel hizalama*) ya da onların alt dizilerini (*yerel hizalama*), olabildiğince az maliyetli değiştirmelerle ve olabildiğince az boşluk kullanarak hizalamayı amaçlar. Dizilerin bu yöntemlerle hizalanması, değişim sırasında dizilerin başına ne gelmiş olabileceğini ve değişime rağmen paylaşmaya devam ettikleri ortak bilgiyi tespit etme olanağı verir. Biyolojik evrim söz konusu olduğunda; farklı nükleotit ya da amino asit dizileri içinde yer alan ortak bilginin –ne kadar süredir korunduğuyla orantılı olarak– canlının verimli bir şekilde üremesine faydalı bir bilgi olduğu söylenebilir. Bu biyolojik evrimin doğal bir sonucudur: Canlının üreme verimini etkilemeyen bölgeler mutasyonlar ile bozulacak ve bu bozukluklar hiç hissedilmeden yeni nesillere aktarılacaktır. Oysa üremeyi az ya da çok kolaylaştıran bölgeler bozulduğunda bozuklukların yeni nesillere aktarılması zorlaşır. Bu da üremeyi kolaylaştıran bilgilinin mutasyonlara karşı daha "dirençli" olmasını sağlar. Hizalama algoritmaları, hem biyologlara eş kökenli dizilerin hizalanmalarına bakıp bu dirençli bölgeleri göz ile görme olanağı verirler, hem de hizalanmalar için ürettikleri puanlarla eş kökenlilik tespiti ve işlev kestirimi gibi problemlerde kullanılabilecek birer benzerlik ölçütü tanımlarlar. Önemli ve yeni bazı hizalama yöntemleri ilerleyen bölümlerde incelenecektir. Yine ilerleyen bölümlerde hizalama algoritmalarına değil, yazımsal örüntü tanımaya dayalı ve biyoloji uygulamalarında ilk defa kullanılan oldukça yeni bir ölçüt de tarif edilecektir.

3.2 Levenshtein Uzaklığı

Dizi benzerliği ile ilgili ilk önemli çalışma 1966'da Vladimir Levenshtein tarafından yapılmıştır [17]. Levenshtein, iki dizinin yazım uzaklığını ölçmek üzere, verilen bir diziden verilen başka bir diziye karakter silerek, karakter ekleyerek ya da bir karakteri başkasıyla değiştirerek en az kaç adımda ulaşılabileceğini hesaplamıştır. Bu hesabın sonucuna *Levenshtein uzaklığı* denir[2]. Örneğin `KİTAP` kelimesinin `MEKTUP` kelimesine olan Levenshtein uzaklığı 4'tür:

1	`KİTAP`	$\rightarrow$	`KİTUP`	(değiştirme)
2	`KİTUP`	$\rightarrow$	`KTUP`	(silme)
3	`KTUP`	$\rightarrow$	`EKTUP`	(ekleme)
4	`EKTUP`	$\rightarrow$	`MEKTUP`	(ekleme)

Herhangi iki $U = u_1u_2\ldots u_m$ ve $V = v_1v_2\ldots v_n$ dizisi arasındaki Levenshtein uzaklığı, $L(U,V)$, özyinelemeli olarak aşağıdaki gibi tanımlanabilir:

$$L(\varepsilon,\varepsilon) = 0 \tag{3.1}$$

$$L(u_1u_2\ldots u_i,\varepsilon) = L(u_1u_2\ldots u_{i-1},\varepsilon)+1 \tag{3.2}$$

$$L\left(\varepsilon,v_1v_2\ldots v_j\right) = L\left(\varepsilon,v_1v_2\ldots v_{j-1}\right)+1 \tag{3.3}$$

$$\begin{aligned} L\left(u_1u_2\ldots u_i,v_1v_2\ldots v_j\right) = \min\big\{&L\left(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_j\right)+1,\\ &L\left(u_1u_2\ldots u_i,v_1v_2\ldots v_{j-1}\right)+1,\\ &L\left(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_{j-1}\right)+\left[u_i\neq v_j\right]\big\} \end{aligned} \tag{3.4}$$

Burada $[a \neq b]$,

$$[a\neq b] = \begin{cases} 0 & a = b \\ 1 & a \neq b \end{cases}$$

şeklinde tanımlanmış bir fonksiyon, $1 \leq i \leq m$ ve $1 \leq j \leq n$'dir. Bu özyinelemeli tanımdan yola çıkarak Levenshtein uzaklığını hesaplayacak dinamik programlama tabanlı bir algoritma tasarlanabilir. Böyle bir algoritmada $(m+1)\times(n+1)$ boyutunda bir tamsayı matrisi yaratılır ve bu matris sol üst köşesinden başlanarak yukarıdaki tanıma göre doldurulur. Tanım gereği, işlem tamamlandığında matrisin sağ alt köşesindeki hücre ele alınan diziler arasındaki Levenshtein uzaklığına eşit olacaktır.

[2]Levenshtein'ın ölçtüğü şey bir benzerlik değil, bir uzaklıktır. Uzaklık arttıkça dizilerin daha az benzer olduğu, azaldıkça dizilerin daha çok benzer olduğu anlaşılır. Bu yazıda anlatım kolaylığı için uzaklıklar ters yönde benzerlikler olarak ele alınacaktır.

		M	E	K	T	U	P
	0	1	2	3	4	5	6
K	1	1	2	2	3	4	5
İ	2	2	2	3	3	4	5
T	3	3	3	3	3	4	5
A	4	4	4	4	4	4	5
P	5	5	5	5	5	5	**4**

Şekil 3.1: Dinamik programlama ile KİTAP ve MEKTUP dizileri arasındaki Levenshtein uzaklığının hesaplanışı.

Örnek 3.1. Alfabe $\Sigma = \{\texttt{A},\texttt{E},\texttt{İ},\texttt{K},\texttt{M},\texttt{P},\texttt{T},\texttt{U}\}$ olsun. Bu alfabede KİTAP ve MEKTUP dizilerini ele alalım. 6×7 boyutunda bir matris oluşturulur ve tanımlandığı gibi doldurulursa Şekil 3.1'deki matrise ulaşılır. Öyleyse $L(\texttt{KİTAP},\texttt{MEKTUP})$, 4'e eşittir.

Levenshtein uzaklığı, genellikle yazım denetimi uygulamalarında yanlış yazılmış kelimeleri düzeltmekte kullanılır. Zayıf tarafı, tüm silme, ekleme ve değiştirme işlemlerinin eşit maliyet ile tanımlanmış olmasıdır. Örneğin yazımda bir harfin diğeriyle karışma olasılığı büyük oranda klavyenin düzenine bağlıdır, fakat bu bağlılık Levenshtein uzaklığına yansıtılamaz. Yazımda olduğu gibi, nükleotit ve amino asit dizilerinin mutasyonunda da eşit maliyet varsayımı çoğu zaman kötü sonuç verir. Bu yüzden Levenshtein uzaklığından yola çıkılarak gelişmiş dizi hizalama algoritmaları üretilmiştir.

3.3 Needleman-Wunsch Algoritması ile Genel Hizalama

1970 yılında, Saul B. Needleman ve Christian D. Wunsch, karakterlerin silinme, eklenme ve başka karakterlerle değiştirilme puanları[3] verildiğinde iki dizinin olası hizalanmaları arasından en yüksek puanlı olanını seçen bir algoritma önerdiler. Önerilen algoritma genel olarak Levenshtein uzaklığı hesaplayan algoritmaya çok benzer. Ancak puanlamanın nasıl yapılacağı parametreler ile belirlenebildiğinden Levenshtein uzaklığındaki eşit maliyet varsayımı kısıdı Needleman-Wunsch algoritmasında bulunmaz. Hizalamaların puanlanışı belirleyen iki parametre vardır:

[3]Burada puan tam olarak negatif maliyet anlamına gelmektedir ve işlemin kolaylığı olarak düşünülebilir.

$g \in \mathbb{R}$ Boşluk cezası (*gap penalty*). Bir diziden diğerine geçiş sırasında silinen ya da eklenen karakterler için puana eklenecek değer. Algoritmanın ürettiği puan negatif maliyete denk düştüğünden boşlukların puana katkısı, g, her zaman 0'dan küçük ya da ona eşit olmalıdır.

$s : \Sigma \times \Sigma \mapsto \mathbb{R}$ Puanlama fonksiyonu[4]. Denk gelen ya da değiştirilen karakterler için puana eklenecek değeri belirleyen fonksiyon. Buna göre bir $a \in \Sigma$ karakterini bir $b \in \Sigma$ karakteri ile değiştirmenin toplam puana katkısı $s(a,b)$'dir. Puanlama fonksiyonu s, değişikliği yapmak maliyet getirdiğinde negatif, kazanç sağladığında pozitif değer almalıdır.

Maliyet yerine puan hesaplandığı akılda tutularak bu parametreler işin içine katıldığında iki $U = u_1u_2\ldots u_m$ ve $V = v_1v_2\ldots v_n$ dizisi arasındaki Needleman-Wunsch puanı, $N_{g,s}(U,V)$, Levenshtein uzaklığına benzer şekilde özyinelemeli olarak aşağıdaki gibi tanımlanabilir:

$$N_{g,s}(\varepsilon,\varepsilon) = 0 \tag{3.5}$$

$$N_{g,s}(u_1u_2\ldots u_i,\varepsilon) = N_{g,s}(u_1u_2\ldots u_{i-1},\varepsilon)+g \tag{3.6}$$

$$N_{g,s}(\varepsilon,v_1v_2\ldots v_j) = N_{g,s}(\varepsilon,v_1v_2\ldots v_{j-1})+g \tag{3.7}$$

$$\begin{aligned} N_{g,s}(u_1u_2\ldots u_i,v_1v_2\ldots v_j) = \max\{&N_{g,s}(u_1u_2\ldots u_i,v_1v_2\ldots v_{j-1})+g,\\ &N_{g,s}(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_j)+g,\\ &N_{g,s}(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_{j-1})+s(u_i,v_j)\} \end{aligned} \tag{3.8}$$

Burada $1 \le i \le m$ ve $1 \le j \le n$'dir. Yine bu özyinelemeli tanımdan yola çıkılarak dinamik programlama ile Needleman-Wunsch puanı hesaplanabilir. Bunun için $(m+1)\times(n+1)$ boyutunda bir matris oluşturulur ve bu matris sol üst köşesinden başlanarak yukarıdaki tanıma göre doldurulur. Tüm matris doldurulduğunda sağ alt köşedeki hücrenin değeri, tanım gereği, dizilerin Needleman-Wunsch puanına eşittir.

Örnek 3.2. Alfabe $\Sigma = \{\texttt{A},\texttt{E},\texttt{İ},\texttt{K},\texttt{M},\texttt{P},\texttt{T},\texttt{U}\}$ olsun. Boşluk cezasını $g = -2$ olarak alalım. Puanlama fonksiyonu $s : \Sigma \times \Sigma \mapsto \mathbb{R}$'yi ise standart Türkçe Q klavyedeki tuş uzaklıklarından faydalanarak Şekil 3.2'deki gibi tanımlayalım. Bu durumda KİTAP

[4]Literatürde puanlama fonksiyonları yerine genellikle puanlama matrisleri (*score matrix*, *scoring matrix* veya *substitution matrix*) kullanılır. Ancak bu yazıda anlatım kolaylığı için bir karakteri diğeriyle değiştirmenin puana katkısı puanlama fonksiyonları ile hesaplanacaktır.

s(a,b)	A	E	İ	K	M	P	T	U
A	0	-2	-10	-7	-7	-9	-4	-6
E	-2	0	-9	-6	-6	-7	-2	-4
İ	-10	-9	0	-3	-4	-2	-6	-5
K	-7	-6	-3	0	-1	-2	-4	-2
M	-7	-6	-4	-1	0	-3	-4	-2
P	-9	-7	-2	-2	-3	0	-5	-3
T	-4	-2	-6	-4	-4	-5	0	-2
U	-6	-4	-5	-2	-2	-3	-2	0

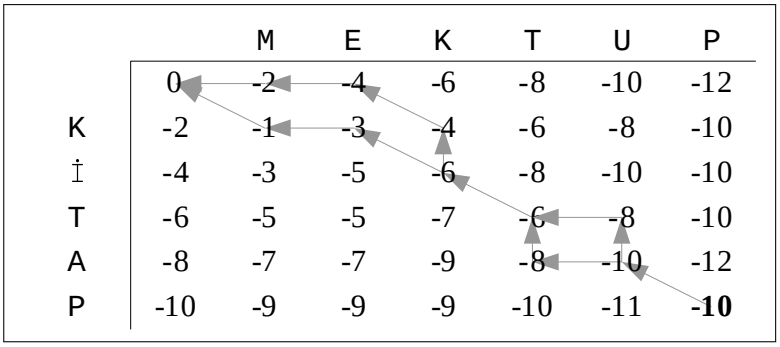
Şekil 3.2: Örnek 3.2 için tanımlanmış puanlama fonksiyonu. Değerler standart Türkçe Q klavyedeki tuş uzaklıklarına göre seçilmiştir.

		M	E	K	T	U	P
	0	-2	-4	-6	-8	-10	-12
K	-2	-1	-3	-4	-6	-8	-10
İ	-4	-3	-5	-6	-8	-10	-10
T	-6	-5	-5	-7	-6	-8	-10
A	-8	-7	-7	-9	-8	-10	-12
P	-10	-9	-9	-9	-10	-11	**-10**

Şekil 3.3: Dinamik programlama ile `KİTAP` ve `MEKTUP` dizilerinin Needleman-Wunsch puanının hesaplanışı.

ve `MEKTUP` dizilerinin Needleman-Wunsch puanı Şekil 3.3'teki gibi hesaplanır ve $N_{g,s}(\texttt{KİTAP},\texttt{MEKTUP}) = -10$ bulunur. Elde edilen değer klavye düzeninin değiştirmeler, silmeler ve eklemeler üzerindeki etkisini yansıttığından, bu değerin bir yazım denetim uygulaması için Örnek 3.1'de hesaplanan Levenshtein uzaklığından daha faydalı olmasını bekleyebiliriz.

Bir kere dinamik programlama matrisi oluşturulduktan sonra, en yüksek puanlı hizalanma (veya hizalanmalar) da üretilebilir. Dinamik programlama matrisine bakarak en yüksek puanlı hizalanmanın üretilmesi işlemine *geri izleme (traceback)* denir. Geri izlemede amaç, sağ alt köşeden başlayarak, karşılaşılan her bir hücreden, maksimum seçilerek o hücreye değerini veren hücreye atlaya atlaya sol üst köşedeki hücreye varmaktır. Bu gezi tamamlandığında en yüksek puanlı hizalanmalardan biri ortaya çıkar: Eğer bir hücre değerini sol üst çaprazındaki hücreden aldıysa, o hücreye denk gelen karakterler *değiştirilmiş* demektir. Eğer hücre değerini üstündeki hücreden aldıysa, birinci dizideki karakter silinmiş; solundaki hücreden aldıysa, ikinci dizideki karakter eklenmiştir.



Şekil 3.4: Örnek 3.2 için geri izleme. Geri izleme eşit puanlara sahip farklı hizalanmalar üretebilir.

Örnek 3.3. Örnek 3.2'te ele alınan problem için yapılan bir geri izleme Şekil 3.4'teki yollardan birini izleyebilir. Buna göre, `KİTAP` ve `MEKTUP` dizilerini en yüksek puanla hizalamanın dört farklı yolu vardır:

```
--KİT-AP    --KİTA-P    K-İT-AP    K-İTA-P
MEK-TU-P    MEK-T-UP    MEKTU-P    MEKT-UP
```

3.3.1 Needleman-Wunsch algoritmasının hesap karmaşıklığı

Needleman-Wunsch algoritmasının dinamik programlama aşaması $(m+1) \times (n+1)$ boyutunda bir matrisi doldurmaktan ibarettir. Her bir hücrenin doldurulması sabit sayıda toplama ve bir adet maksimum hesaplama işlemi gerektirir ve asimptotik olarak $O(1)$ birim zaman alır. Dolayısıyla dinamik programlama aşamasının zaman karmaşıklığı $O(m \cdot n)$ olur. Alan karmaşıklığı ise matrisin tamamı geri izleme için bellekte tutulduğundan yine $O(m \cdot n)$'dir.

Eğer en yüksek puanı veren hizalanmalardan yalnızca bir tanesi üretilecekse, geri izleme aşaması matrisin bir köşesinden diğerine gitmek için $O(m+n)$ birim zaman ve sonucu tutmak için $O(m+n)$ birim alan gerekir. Bu da Needleman-Wunsch algoritmasının toplam zaman karmaşıklığını da alan karmaşıklığını da $O(m \cdot n)$ yapar. Needleman-Wunsch algoritması yeterli alan verildiğinde oldukça verimlidir. Yine de aynı işi daha verimli olarak yapan bir algoritma [18]'de bulunabilir.

3.4 Smith-Waterman Algoritması ile Yerel Hizalama

1981 yılında Tempe Smith ve Michael Waterman, verilen iki dizinin en yüksek hizalanma puanını üreten alt dizilerini bulacak bir algoritma önerdiler.

s(a,b)	A	E	İ	K	M	P	T	U
A	4	-2	-10	-7	-7	-9	-4	-6
E	-2	4	-9	-6	-6	-7	-2	-4
İ	-10	-9	4	-3	-4	-2	-6	-5
K	-7	-6	-3	4	-1	-2	-4	-2
M	-7	-6	-4	-1	4	-3	-4	-2
P	-9	-7	-2	-2	-3	4	-5	-3
T	-4	-2	-6	-4	-4	-5	4	-2
U	-6	-4	-5	-2	-2	-3	-2	4

Şekil 3.5: Örnek 3.4 için tanımlanmış puanlama fonksiyonu. Smith-Waterman algoritması için köşegendeki değerler vurgulanmıştır.

Önerdikleri algoritma Needleman-Wunsch algoritmasında yalnızca küçük bir değişiklik gerektirir. Needleman-Wunsch algoritması, her yeni adımda iki karakteri eşlemek, birinci karakteri silmek ve ikinci karakteri eklemek seçeneklerinden birisini seçiyordu. Smith-Waterman algoritmasında seçeneklere bir yenisi eklenir: Önceki hizalamayı unutup tam o adımda hizalamaya başlamak. Bunun için dinamik programlama matrisinin özyinelemeli tanımında yer alan max fonksiyonu içine yeni bir terim, 0, eklemek yeterlidir:

$$W_{g,s}(\varepsilon,\varepsilon) = 0 \tag{3.9}$$

$$W_{g,s}(u_1u_2\ldots u_i,\varepsilon) = 0 \tag{3.10}$$

$$W_{g,s}(\varepsilon,v_1v_2\ldots v_j) = 0 \tag{3.11}$$

$$\begin{aligned} W_{g,s}(u_1u_2\ldots u_i,v_1v_2\ldots v_j) = \max\{0, & \\ & W_{g,s}(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_j)+g, \\ & W_{g,s}(u_1u_2\ldots u_i,v_1v_2\ldots v_{j-1})+g, \\ & W_{g,s}(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_{j-1})+s(u_i,v_j)\} \end{aligned} \tag{3.12}$$

Smith-Waterman algoritmasında, yine Needleman-Wunsch'tan farklı olarak, geri izleme işlemine matrisin sağ alt köşesinden başlanmaz. Geri izleme, matrisin en yüksek değerli hücresinden başlar (bu hücrenin değeri aynı zamanda üretilecek hizalanmanın puanıdır) ve değeri 0 olan bir hücre görülene kadar devam eder.

Örnek 3.4. Yine alfabe $\Sigma = \{\mathtt{A},\mathtt{E},\mathtt{İ},\mathtt{K},\mathtt{M},\mathtt{P},\mathtt{T},\mathtt{U}\}$ ve boşluk cezası $g = -2$ olsun. Smith-Waterman algoritması dinamik programlama matrisinde pozitif değerler bulunmasın gerektirir. Bunun için puanlama fonksiyonunu köşegendeki

		E	K	M	E	K	K	İ	T	A	P
	0	0	0	0	0	0	0	0	0	0	0
K	0	0	4	2	0	4	4	2	0	0	0
İ	0	0	2	0	0	2	2	8	6	4	2
T	0	0	0	0	0	0	0	6	12	10	8
A	0	0	0	0	0	0	0	4	10	16	14
P	0	0	0	0	0	0	0	2	8	14	**20**
M	0	0	0	4	2	0	0	0	6	12	18
E	0	4	2	2	8	6	4	2	4	10	16
K	0	2	8	6	6	12	10	8	6	8	14
T	0	0	6	4	4	10	8	6	12	10	12
U	0	0	4	4	2	8	8	6	10	8	10
P	0	0	2	2	0	6	6	6	8	6	12

Şekil 3.6: Örnek 3.4 için dinamik programlama matrisi ve geri izleme.

değerleri pozitif yapacak şekilde Şekil 3.5'teki gibi değiştirelim. Bu şartlar altında `KİTAPMEKTUP` ve `EKMEKKİTAP` dizilerini Smith-Waterman algoritması ile hizalarsak Şekil 3.6'daki dinamik programlama matrisine ulaşırız. Buna göre bu iki dizinin alt dizileri arasındaki en iyi hizalanma aşağıdaki gibidir:

```
KİTAP
KİTAP
```

Aynı iki diziyi aynı parametreleri kullanarak Needleman-Wunsch algoritması ile hizalasaydık şu hizalanmayı elde edecektik:

```
-KİTAPME-K-T-UP
EK----MEKKİTA-P
```

Görüldüğü gibi bu örnekte Smith-Waterman algoritması dizilerin gerçekte başına ne geldiği konusunda daha akla yatkın bir cevap veriyor. Bunu sebebi Smith-Waterman algoritmasının, bilginin dizi içindeki konumunu önemsemeden birbirine uzak bölgeleri de hizalayabiliyor olmasıdır. Böylece bu örnekteki gibi *genel* olarak birbirine benzemeyen diziler arasındaki *yerel* benzerlikleri yakalayabilir. Bu, biyoloji bağlamında, birbirine uzaktan akraba olabilecek dizileri incelerken ihtiyaç duyulan bir özelliktir. Needleman-Wunsch algoritması kısa vadede dizilerin başına ne gelmiş olabileceğini ortaya koyarken, Smith-Waterman algoritması uzun vadede korunan bilgiyi ortaya çıkarır.

Smith-Waterman algoritmasının zaman ve alan karmaşıklığı, Needleman-Wunsch algoritmasında olduğu gibi, $O(m \cdot n)$'dir.

3.5 Biyolojik Puanlama Matrisleri

Hizalama algoritmalarının ürettikleri sonuçlar kullanılan puanlama fonksiyonu ile doğrudan bağlantılıdır. Yukarda verilen örneklerde puanlama için tuşların standart Türkçe Q klavye düzenindeki uzaklıklarından elde edilmiş matrisler kullanılmıştı. Bu seçim, hizalama algoritmalarının `A` karakterini `İ` karakteriyle eşleştirmekten çekinmelerine yol açar. Fakat eğer verilen diziler standart Türkçe F klavye düzeninde yazılmış yazılardan alınmış olsaydı, seçilen puanlama fonksiyonu yanlış sonuçlara götürecekti: Standart Türkçe F klavye düzeninde `A` karakteri ile `İ` karakterini karıştırmak, Q klavye düzenindekine göre çok daha kolaydır.

Amino asit dizilerinde de buna benzer bir durum söz konusudur. Amino asitlerin kimyasal özellikleri (kutupluluk, yük, su severlik, vb.), bazı amino asitlerin birbiri arasında değişmesini daha mümkün kılarken, bazı amino asitlerin birbirinin yerine geçmesini zorlaştırır. Öyleyse amino asit dizilerinin hizalanmasında, bu gerçeği destekleyecek şekilde oluşturulmuş bir puanlama fonksiyonuna ihtiyaç vardır.

3.5.1 PAM matrisleri

Margaret B. Dayhoff ve arkadaşları [19] 1978'de bir amino asidin diğerinin yerin geçme olasılığını ampirik olarak ölçerek amino asitler için bir puanlama fonksiyonunu sundular. Çalışmalarında, 71 aileye ayrılmış 1572 yakın akraba proteini ele alarak bu proteinlerde meydana gelmiş nokta mutasyonları belirlediler ve bu mutasyonların birbirlerinden bağımsız olduğunu varsayarak bir amino asidin diğerinin yerine geçme olasılığını hesapladılar. *Kabul edilmiş nokta mutasyon (point accepted mutation – PAM)*, bu olasılıklar kullanılarak oluşturulmuş 20×20 boyutundaki matrislerinin adıdır. Farklı uzunlukta mutasyon serileri için farklı PAM matrisleri kullanılır. Mutasyon serisi uzadıkça bir amino asidin başka bir amino aside dönüşme olasılığı artar. $\texttt{PAM}_1$, amino asitlerin %1'i değiştikten sonra ölçülen olasılıkları içerir. Daha uzun seriler için PAM matrisleri, $\texttt{PAM}_1$ matrisinin kendi kendisi ile çarpılmasıyla elde edilebilir. Örneğin $\texttt{PAM}_2$ matrisi

$\texttt{PAM}_{249} \times \texttt{PAM}_1$'e eşittir. Hangi PAM matrisinin kullanılacağı, eldeki proteinlerin evrimsel uzaklıklarına bakılarak karar verilmesi gereken bir parametredir.

Her bir yapıtaşının diğer yapıtaşlarından bağımsız olarak mutasyona uğradığı varsayımıyla, bir hizalanmanın olasılığı alt alta gelen karakterlerin PAM olasılıklarının çarpılmasıyla bulunabilir. Hizalanma algoritmalarıysa toplayarak işlem yaparlar. Bu yüzden PAM matrisleri hizalama algoritmalarında kullanılacakları zaman, *logaritmik ihtimal oranı* (*log-odds*) içeren LOGPAM[5] matrislerine çevrilirler. $a \in \Sigma$ ve $b \in \Sigma$ gibi iki karakter için logaritmik ihtimal oranı aşağıdaki gibi hesaplanır:

$$l(a,b) = \log\left(\frac{p_{ab}}{p_a p_b}\right) \tag{3.13}$$

Burada p_{ab}, a ve b karakterlerinin alt alta gözlenme olasılığı, p_a ve p_b ise, sırasıyla, a ve b karakterlerinin kendi başlarına gözlenme olasılıklarıdır. Bu şekilde hesaplanan logaritmik ihtimal oranları üzerinde toplama işlemi yapılabilir. Bu yüzden LOGPAM matrisleri hizalama algoritmalarında kullanılmaya uygundur. Farklı PAM matrislerinden yola çıkılarak, farklı uzunluktaki mutasyon serileri için farklı LOGPAM matrisleri hesaplanabilir (ör. $\texttt{LOGPAM}_1$, $\texttt{LOGPAM}_{100}$, vb.).

3.5.2 BLOSUM matrisleri

Kısa mutasyon serileri için PAM matrisleri yararlı olsa da, mutasyon serileri uzadıkça mutasyonların birbirinden bağımsız olduğu varsayımı geçersiz olmaya başlar. Uzak akrabalıkların tespitinde Steven Henikoff ve Jorja G. Henikoff [20] tarafından hazırlanmış olan BLOSUM matrisleri kullanılır. PAM matrislerinin aksine, BLOSUM matrisleri kısa mutasyon serilerinden yola çıkarak uzun serilere varmaya çalışmaz. BLOSUM matrisleri doğrudan uzun seriler üzerinden ölçüm yapılarak oluşturulmuştur. Bunun için, birbirine belli oranda benzeyen proteinler alınıp yerel hizalamaya tabi tutulur ve elde edilen hizalanmalarda bir amino asidin bir başkasıyla eşleşme olasılığını ölçülür. Üzerinde ölçüm yapılan proteinlerin birbirine ne oranda benzediği, elde edilen matrisin ne kadar uzak akrabalıkları tanıyacağını belirler. En uzak akrabalıkların tanınması için %45 oranında

[5]Hizalamada sürekli logaritmik ihtimal oranına çevrilmiş matrisler kullanıldığından bu yazıda LOGPAM olarak anılan matrisler literatürde çoğunlukla yalnızca PAM olarak anılır. Ancak ilerleyen bölümlerde olasılık içeren PAM matrislerine de başvurulacağından bu yazıda olasılık içeren matrisler PAM, logaritmik ihtimal oranı içeren matrisler LOGPAM adıyla anılacaktır.

benzeyen proteinlerden üretilmiş $\texttt{BLOSUM}_{45}$ kullanılırken, yakın akrabalıkların tanınmasında %80 oranında benzeyen proteinlerden üretilmiş $\texttt{BLOSUM}_{80}$ kullanılır. Henikoff ve Henikoff bu şekilde $\texttt{BLOSUM}_{45}$, $\texttt{BLOSUM}_{60}$, $\texttt{BLOSUM}_{62}$, $\texttt{BLOSUM}_{70}$ ve $\texttt{BLOSUM}_{80}$ matrislerini üretmişlerdir. Bu matrisler ile yapılan yerel hizalamaların uzak akrabalıkların tespitinde PAM matrisleri ile yapılanlardan daha başarılı olduğu çeşitli kereler ortaya konmuştur [20, 21].

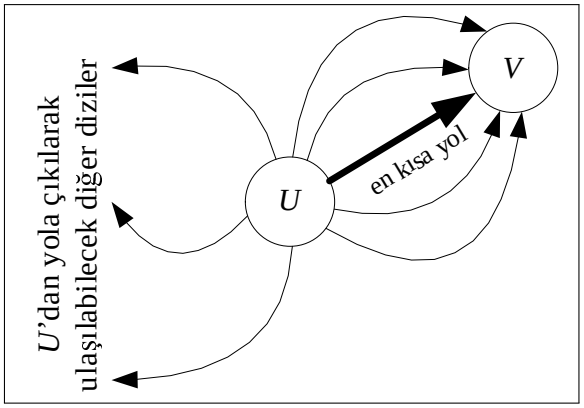
Bu çalışmanın protein işlev kestirimi yapılan birinci kısmında, uzak akraba olabilecek proteinlerin işlevlerini ortaya çıkaran görece kısa bölgelerin tespit edilebilmesi için BLOSUM matrisleri ile yerel hizalama yapılmıştır. İkinci kısımda ise, eşit uzunluktaki kısa zincirlerden oluşan peptitler üzerinde çalışıldığından, geçiş olasılıkları ile karşılaştırmak üzere PAM matrisleri kullanılarak genel hizalama yapılması uygun görülmüştür.

3.6 Birleşik Hizalama ve Wallqvist Matrisi

Aders Wallqvist, Yoshimufi Fukunishi, Lynne Reed Murphy, Addi Fadel ve Roland M. Levy [2], 2000 yılında yaptıkları çalışmada, amino asit dizilerini (birincil yapıları) ikincil yapılarla birlikte hizalamanın, katlanış tanımada yalnızca amino asit dizilerini hizalamaktan daha iyi sonuç verdiğini gösterdiler. Bir proteinin ikincil yapısını Çizelge 2.2'de gösterilen DSSP kodları ile ifade ettiğinizde, proteinin amino asit dizisine eşit uzunlukta yeni bir dizi elde edersiniz. Wallqvist ve arkadaşları bu ikincil yapı dizisini, amino asit dizisiyle birlikte hizalamanın bir yolunu sundular. Bunun için iki diziyi, tek bir birleşik dizi ile ifade ettiler. Formel olarak, yaptıkları, amino asit dizilerinin alfabesi Σ_{AA} ile ikincil yapı dizilerinin alfabesi Σ_{SS}'nin kartezyen çarpımını yeni bir alfabe, $\Sigma_C = \Sigma_{AA} \times \Sigma_{SS}$ olarak ele alıp puanlama fonksiyonunu ve Smith-Waterman hizalama algoritmasını bu yeni alfabe üzerinde tanımlamaya denktir. Wallqvist ve arkadaşlarının çalışmasında birleşik puanlama fonksiyonu, $s_c : \Sigma_C \times \Sigma_C \mapsto \mathbb{R}$ şöyle tanımlanmıştır:

$$s_c\left[(a,x),(b,y)\right] = (1-\alpha)\, s_{AA}(a,b) + \alpha s_{SS}(x,y) \qquad \textbf{(3.14)}$$

Burada $(a,x) \in \Sigma_{AA} \times \Sigma_{SS}$ ve $(b,y) \in \Sigma_{AA} \times \Sigma_{SS}$, sırasıyla, birinci ve ikinci diziden gelen birincil yapı-ikincil yapı çiftleridir. α parametresi, birincil yapı ile ikincil



Şekil 3.7: Hizalama algoritmaları bir diziden diğerine ulaşmanın en kısa yolunu ölçerler.

yapının sonuç üzerindeki dengesini belirler. s_{AA} amino asitlerin puanlamasını yapan fonksiyondur ve tanımında BLOSUM gibi standart matrisler kullanılabilir. s_{SS} ise ikincil yapıların puanlamasını yapar.

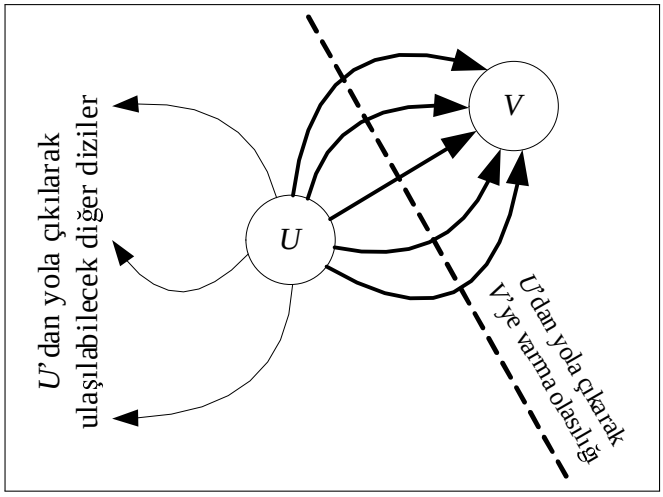
İkincil yapıların puanlamasını yapan bir matris yine Wallqvist ve arkadaşları tarafından aynı çalışmada ortaya konmuştur. Yazarlar, BLOSUM matrislerinin oluşturulmasında birincil yapılar için izlenen yolu, üç boyutlu olarak hizalanmış ve 86 yapısal sınıfa ayrılmış 455 protein içeren 3D_ali veritabanında [22] ikincil yapılar için izleyerek bir ikincil yapı puanlama matrisi elde etmişlerdir. Elde ettikleri matris bu yazıda `WALLQVIST` şeklinde gösterilecektir.

Bu çalışmanın deneysel kısmının ilk aşaması, Wallqvist ve arkadaşlarının tekniğini kullanarak, protein işlev kestiriminde ikincil yapının katkısını ölçmektir.

3.7 Oommen-Kashyap Geçiş Olasılığı

Buraya kadar bahsedilen tüm dizi karşılaştırma yöntemleri birer hizalama algoritmasına dayanır. Hizalama algoritmaları söz konusu olduğunda, dizi benzerliği, en uygun hizalanma puanının bir fonksiyonu olarak tanımlanır. Dizileri iki boyutlu harita üzerinde birer şehir gibi düşünürsek, bu yaklaşım bir şehirden diğerine gitmenin ne kadar kolay olduğuna iki şehir arasındaki en kısa yolun uzunluğuna bakarak karar vermeye benzer (Şekil 3.7).

Bu ölçüm tarzı, yolun özellikle kısa olacak şekilde önceden seçildiği varsayımıyla mantıklıdır. Ancak biyolojik evrimde süreç bu şekilde işlemez. Biyolojik dizilerin mutasyonu, dizi uzayında bilinçli bir hareketten çok, bir rastgele yürüyüşe



Şekil 3.8: Mutasyonları modellemenin uygun bir yolu bulunursa, bir diziden diğerine ulaşmanın olasılığı ölçülebilir.

(*random walk*) benzer. Bu yüzden, ölçülmesi gereken şey daha çok Şekil 3.8'deki gibidir. İyi tanımlanmış bir mutasyon modelinde, iki diziyi birbirine bağlayan tüm yolları hesabına katan bir ölçüt, tek bir yolu temsil eden bir hizalanma puanından daha çok bilgi içerecektir.

B. John Oommen ve Rangasami L. Kashyap, 1998 yılında, diziler üzerinde çalışılan pek çok alanda uygulanabilecek iyi tanımlanmış bir mutasyon modeli ortaya attılar, bu model üzerinde bir diziden başka bir diziye geçme olasılığını veren eşitliği ortaya koydular, bu eşitliğin eksizliğini ve tutarlılığını gösterdiler ve geçiş olasılığını verimli bir şekilde hesaplayan bir algoritma sundular. İlerleyen alt bölümlerde Oommen ve Kashyap'ın çalışması parça parça ele alınacak ve bu çalışmada biyolojik dizi analizine nasıl uyarlandığı anlatılacaktır.

3.7.1 Oommen-Kashyap modeli

Oommen ve Kashyap, mutasyonları bir diziden yeni bir dizi üreten rastlantısal bir süreç olarak modellemişlerdir. Süreç, M^*, tıpkı hizalamada olduğu gibi, bir karakteri başkasıyla *değiştirme*, karakter *ekleme* ve karakter *silme* temel işlemlerine dayanır ve şu şekilde işler: Öncelikle, sürece giren diziye eklenecek karakter sayısı, z, G olasılık dağılımına göre belirlenir. İkinci adımda z adet boşluk dizinin rastgele yerlerine eklenir. Boşlukların eklenmesinin ardından, tüm

Algoritma 1 Oommen-Kashyap modelinde bir diziyi diğerine çeviren süreç, M^*.

Girdi: Dizi U, eklenecek karakter sayısı dağılımı G, değiştirme dağılımı S.
Çıktı: Dizi V.

1: G dağılımını kullanarak diziye eklenecek karakter sayısı z'yi belirle.
2: U dizisinin rastgele z noktasına boşluk karakteri ekleyerek U' dizisini elde et.
3: U' dizisindeki her bir karakterleri S dağılımına göre yeni bir karakterle değiştirerek V' dizisini elde et.
4: V' dizisindeki boşluk karakterlerini silerek V dizisini elde et.
5: Sonuç olarak V dizisini dön.

karakterler S dağılımına göre yeni karakterlerle değiştirilir. Kalan boşlukların ortadan kaldırılmasıyla elde edilen yeni dizi, sürecin çıktısıdır[6] (Algoritma 1).

Eklenecek karakter sayısı dağılımı, G, aşağıdaki şartı sağlayan bir olasılık dağılımıdır[7]:

$$\sum_{z=0}^{\infty} G(z) = 1 \tag{3.15}$$

Bunun dışında G'nin alabileceği şekilleri kısıtlayan hiçbir şart yoktur. Uygulamaya göre geometrik dağılım, iki terimli (*binomial*) dağılım, Poisson dağılımı veya istenilen herhangi başka bir dağılım kullanılabilir. Modelin bu esnekliği, onu üstün kılan yanlarından bir tanesidir.

Değiştirme dağılımı S, giren dizideki bir karakterin, çıkan dizideki belli bir karaktere dönüşme olasılığını belirler. Girdide $a \in \Sigma \cup \{-\}$ karakteri görülen yere çıktıda $b \in \Sigma \cup \{-\}$ karakterinin yerleşmesi ihtimali $S(b|a)$ olarak ifade edilir ve buna göre her $a \in \Sigma \cup \{-\}$ için S aşağıdaki iki şartı sağlamalıdır[8]:

$$\sum_{b \in \Sigma \cup \{-\}} S(b|a) = 1 \tag{3.16}$$

$$S(-|-) = 0 \tag{3.17}$$

Görüldüğü gibi giren bir karakter yerine boşluk karakteri $(-)$ üretilebilir. Bu, girdideki karakterin silindiği anlamına gelir. Ayrıca, giren diziye eklenen boşluk

[6] Oommen ve Kashyap, giren dizinin alfabesi ile çıkan dizinin alfabesini ayrı tutmuşlardır. Bu ayrım modele fazladan bir esneklik sağlar. Ancak bu çalışmada giren diziler ile çıkan diziler her zaman aynı alfabe üzerinde tanımlı olacağından bu esnekliğe ihtiyaç yoktur. O yüzden tanımlar girdi alfabesi ile çıktı alfabesinin aynı olduğu varsayımına göre yapılacaktır.

[7] Oommen ve Kashyap, G dağılımının istenirse giren dizi U'ya bağlı olarak tanımlanabileceğini belirtmişlerdir. Bu çalışmada G'nin her zaman U'dan bağımsız olduğu varsayılacaktır.

[8] Oommen ve Kashyap, boşluk karakterlerinin yeni karakterlerle değiştirilme olasılıklarını veren *ekleme dağılımını* S'den ayrı olarak ele almıştır. $S(-|-) = 0$ kısıdının korunduğuna dikkat edildiği sürece buna gerek olmadığından, bu çalışmada ayrıca bir ekleme dağılımı tanımlanmayacaktır.

karakterleri boşluk dışındaki bir karaktere dönüşmek zorundadır. Bu da yeni karakterlerin eklenmesi olayıdır.

Sürecin tüm parametreleri G ve S'dir. Bu parametrelerin uygun şekilde verildiği varsayımıyla örnek olarak KİTAP dizisinin süreç içinde izleyebileceği yollardan birini inceleyelim. Öncelikle eklenecek karakter sayısına karar verilmeli: Eklenecek karakter sayısının 3 olarak belirlendiğini varsayalım. İkinci adım, eklenecek karakterlerin yerini tutacak 3 adet boşluk karakterini dizi boyunca rastgele dağıtmak. Bu adımda her olasılığın eşit olduğu kabul edildiğine dikkat edin. Bu işlemin sonucunda -KİTA-P elde edilmiş olsun. Şimdi yapılması gereken, her bir karakteri S dağılımına göre başka bir karakterle değiştirmek. Bunun sonucunda da MEK-T-UP elde edilsin. Artık tek yapılması gereken değiştirme işleminden arta kalan boşlukları aradan çıkarmak. Bunun sonucu da MEKTUP dizisidir. Görüldüğü gibi gerekli parametrelerin verildiği varsayımıyla bu modelde KİTAP dizisinden MEKTUP dizisine *tarif edilen şekilde* geçme olasılığı kolayca hesaplanabilir. Tabii ki; bir diziden ötekine geçmenin bu tarifin dışında pek çok yolu bulunur. Asıl soru tüm yolların olasılığının toplamının nasıl hesaplanacağı sorusudur.

3.7.2 Oommen-Kashyap modelinde geçiş olasılığı

Oommen ve Kashyap, çalışmalarında, tanımladıkları modele göre bir diziden diğer bir diziye geçmenin tüm yollarının olasılıklarının toplamının nasıl hesaplanacağını ortaya koymuşlardır. Bu hesap yolların sayılmasıyla elde edilir. Eğer Σ alfabesi üzerinde tanımlı bir $U = u_1 u_2 \dots u_m$ dizisinden bir $V = v_1 v_2 \dots v_n$ dizisine M^* süreci ile geçme olasılığına $\Pr(V|U)$ denirse, $\Pr(V|U)$ için aşağıdaki eşitlik yazılabilir:

$$\Pr(V|U) = \sum_{z=\max\{0,n-m\}}^{n} \frac{G(z)\, m!\, z!}{(m+z)!} \sum_{U'} \sum_{V'} \prod_{i=1}^{m+z} S\left(v'_i|u'_i\right) \qquad \textbf{(3.18)}$$

Burada U' alt indisi, U dizisinden çeşitli yerlere boşluk karakterleri eklenerek elde edilebilecek $n+z$ uzunluğundaki tüm diziler üzerinden toplam yapıldığı anlamına gelir. Buna göre bahsi geçen toplam ifadesinin içinde U', $\Sigma \cup \{-\}$ alfabesi üzerinde tanımlanmış $U' = u'_1 u'_2 \dots u_{m+z}$ biçiminde bir dizidir. V' alt indisinin anlamı da V dizisiyle ilgili olmak üzere aynıdır: Toplamın içinde V', $\Sigma \cup \{-\}$ alfabesi üzerinde

tanımlanmış $V' = v'_1 v'_2 \dots v'_{m+z}$ biçiminde bir dizidir. Bölüm 3.1'de tanımı yapılan D fonksiyonu hatırlanırsa, $U = D(U')$ ve $V = D(V')$ ifadeleri yazılabilir.

Eşitlik 3.18'in ispatı, farklı yolların sayımının nasıl yapıldığı anlaşılarak yapılabilir. İçten dışarı gidelim. $n+z$ uzunluğundaki belli bir U' dizisi ele alındığında, olası V' dizilerinin her birinin elde edilme olasılığı, alt alta gelen karakterlerin S ile tanımlanmış değiştirilme olasılıkları çarpılarak kolayca elde edilebilir. Bu olasılıklar tüm olası V' dizileri üzerinden toplanırsa, belli bir U' dizisinden çıktı dizisi V'ye geçme olasılığı elde edilir. Tüm olası U' dizilerinin olasılığı tanım gereği eşittir; dolayısıyla bir önceki cümlede bahsedilen olasılıklar $n+z$ uzunluğundaki tüm U' dizileri üzerinden toplanıp, olası U' dizilerinin sayısına bölünürse, eklenecek karakter sayısı z'nin belli bir seçimi için U'dan V'ye geçme olasılığı elde edilir. Belli bir z için olası U' dizilerinin sayısı, tekrarlı permütasyon gereği $(n+z)!/(n!z!)$'e eşittir. Son olarak, belli bir z değerinin seçilme olasılığı tanım gereği $G(z)$'ye eşittir ve her bir z için elde edilen olasılıklar $G(z)$ ile çarpılmalıdır. Nihayet, z'nin alabileceği değerler üzerinden bu çarpımlar toplanırsa U dizisinden V dizisine, modelin izin verdiği *herhangi bir yoldan* ulaşma olasılığı bulunmuş olur.

3.7.3 Geçiş olasılığının hesaplanması: Oommen-Kashyap algoritması

Oommen-Kashyap modeline göre geçiş olasılığının Eşitlik 3.18'de ifade edildiği şekilde hesaplanması oldukça zordur. Fakat eğer hizalama algoritmalarındaki gibi özyinelemeli bir ifade üretilebilirse dinamik programlamaya başvurularak verimli bir algoritmaya ulaşılabilir. Oommen ve Kashyap'ın yaptığı tam olarak budur.

Eklenecek karakter sayısının sabiti olduğu durumu ele alalım ($Z = z$). Bu durumda $U = u_1 u_2 \dots u_m$ ve $V = v_1 v_2 \dots v_n$ dizileri için aşağıdaki ifade yazılabilir:

$$\Pr(V|U;Z=z) = \frac{m!\,z!}{(m+z)!} \sum_{U'} \sum_{V'} \prod_{i=1}^{m+z} S\left(v'_i|u'_i\right) \qquad \textbf{(3.19)}$$

Bu ifadeyi özyinelemeli olarak hesaplayabilmek için, kombinatorik çarpanın dışında kalan toplamı veren bir Q_S fonksiyonu tanımlayalım. Öyle ki;

$$
\begin{aligned}
Q_S(\varepsilon,\varepsilon;0) &= 1 && \textbf{(3.20)}\\
Q_S(u_1u_2\ldots u_i,\varepsilon;0) &= Q_S(u_1u_2\ldots u_{i-1},\varepsilon;0)\cdot S(-|u_i) && \textbf{(3.21)}\\
Q_S(\varepsilon,v_1v_2\ldots v_j;0) &= 0 && \textbf{(3.22)}\\
Q_S(u_1u_2\ldots u_i,v_1v_2\ldots v_j;0) &= Q_S(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_{j-1};0)\cdot S(v_j|u_i) && \textbf{(3.23)}\\
&\quad +Q_S(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_j;0)\cdot S(-|u_i)\\
Q_S(\varepsilon,\varepsilon;z) &= 0 && \textbf{(3.24)}\\
Q_S(u_1u_2\ldots u_i,\varepsilon;z) &= Q_S(u_1u_2\ldots u_{i-1},\varepsilon;z)\cdot S(-|u_i) && \textbf{(3.25)}\\
Q_S(\varepsilon,v_1v_2\ldots v_j;z) &= Q_S(\varepsilon,v_1v_2\ldots v_{j-1};z-1)\cdot S(v_j|-) && \textbf{(3.26)}\\
Q_S(u_1u_2\ldots u_i,v_1v_2\ldots v_j;z) &= Q_S(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_{j-1};z)\cdot S(v_j|u_i) && \textbf{(3.27)}\\
&\quad +Q_S(u_1u_2\ldots u_{i-1},v_1v_2\ldots v_j;z)\cdot S(-|u_i)\\
&\quad +Q_S(u_1u_2\ldots u_i,v_1v_2\ldots v_{j-1};z-1)\cdot S(v_j|-)
\end{aligned}
$$

olsun. Burada $1 \le i \le m$, $1 \le j \le n$ ve $z \ge 1$'dir. Özünde Q_S, hizalama algoritmalarında tanımlanan özyinelemeli fonksiyonlara benzese de, eklenecek karakter sayısı z'nin kaydını tutabilmek için ekleme işlemlerini silme işlemlerinden ayrı ele alır. Hesaba eklenen bu karmaşıklık dinamik programlama matrisine yeni bir boyut olarak yansır: Oommen-Kashyap algoritmasında dinamik programlama matrisi $(m+1)\times(n+1)\times m$ boyutunda üç boyutlu bir matristir. Sol-üst-ön köşesinden başlanarak sağ-alt-arka köşesine kadar yukarıdaki formüle göre doldurulur. Bir kere Q_S'nin tüm değerleri matriste erişilebilir olduktan sonra

$$\Pr(V|U;Z=z) = \frac{m!\,z!}{(m+z)!}Q_S(U,V;z) \qquad \textbf{(3.28)}$$

yazılabilir. Ve böylece;

$$\Pr(V|U) = \sum_{z=\max\{0,n-m\}}^{n} \frac{G(z)\,m!\,z!}{(m+z)!}Q_S(U,V;z) \qquad \textbf{(3.29)}$$

olarak hesaplanır.

3.7.4 Oommen-Kashyap algoritmasının hesap karmaşıklığı

Dinamik programlamanın üç boyutlu matris üzerinde yapılıyor olması zaman ve alan karmaşıklığını kötü yönde etkiler. Yukarda tanımlandığı şekliyle

Oommen-Kashyap algoritmasının zaman ve alan karmaşıklığı $O\left(m^2 \cdot n\right)$'dir. Karmaşıklıklar hâlâ çok terimli (*polynomial*) olsa da, bu algoritmayla büyük dizilerin geçiş olasılığının hesaplanması zordur. Her ikisi de 1000 karakter içeren iki dizi için, 1 milyar adet kayan noktalı sayıyı *(floating point number)* tutacak kadar alan (≥ 4 GB) ve bu sayıları tek tek üretecek kadar zaman gerekir.

Neyse ki; zaman karmaşıklığı azaltılamasa da, algoritmanın alan karmaşıklığı, üç boyutlu matrisin yalnızca belli bir z'ye ve onun öncülü $z-1$'e denk gelen iki boyutlu dilimleri bellekte tutularak azaltılabilir. Bu yaklaşım çalışma zamanına sabit bir yük getirse de asimptotik zaman karmaşıklığını değiştirmez. Alan karmaşıklığı ise $O(m \cdot n)$'ye düşer. Bu çalışmanın bir ürünü olan BALIGN programında, alan verimli Oommen-Kashyap algoritması kullanılmıştır.

3.7.5 Oommen-Kashyap algoritmasında alttan taşma sorunu

Alttan taşma (underflow), mutlak değer olarak çok küçük kayan noktalı sayıları birbiriyle çarparken karşılaşılabilen bir sorundur. Kayan noktalı sayılar bir adet mantis ve bir adet üstel çarpan ile temsil edilirler. Mantis sayının anlamlı basamaklarını belirtirken, üstel çarpan sayının üstel büyüklüğünü belirler. Böylece çok küçük ve çok büyük sayılar üzerinde işlem yapılabilir. Yine de tutulabilecek sayıların küçüklüğü ve büyüklüğü üzerinde bir kısıtlama bulunur: IEEE 754 standardına [23] göre çift kesinlikli bir kayan noktalı sayının üstel çarpanı 2^{-1022} ile 2^{1023} arasında değer alabilir. Bu aralığın dışındaki sayılar kayan noktalı sayılarla temsil edilemezler.

Oommen-Kashyap geçiş olasılıkları, özellikle uzun diziler için, mutlak değerce oldukça küçük değerler alabilir. Her ikisi de 1000 karakter uzunluğundaki iki dizi örneğine tekrar bakarsak; bu örnekte dinamik programlama matrisinin son hücresi (sağ-alt-arka hücresi) $[0,1]$ aralığındaki 1000 adet sayının çarpılması ve toplanması ile elde edilir. Çarpılan değeri 0.1, ve her adımda gerçekleşen üç sayının toplanması işlemini 3 ile çarpmaya denk kabul eden kaba bir hesap, son hücrenin değerinin 0.3^{-1000} mertebesinde olacağını gösterir. Bu sayı 2^{-1736}'dan daha küçüktür. Dolayısıyla sayılar standart kayan noktalı sayılarla temsil edilirse alttan taşma yüzünden anlamlı değerler yerine 0'a ulaşılır. Üstelik uygulamada, çarpılan değerler (değiştirme olasılıkları) 0.1'den çok daha küçük

olabilir. Oommen ve Kashyap uzun dizilerle çalışmadıklarından çalışmalarında bu soruna değinmemişlerdir. Biyolojide ise bir protein 1000'den fazla amino asit içerebilir. Böyle uzun diziler arasındaki geçiş olasılıklarını hesaplayabilmek için alttan taşma sorununa bir çözüm bulmak gerekir.

Bu çalışmada önerilen çözüm, tüm hesabın logaritma uzayında yapılmasıdır. Logaritma uzayında çalışmak, çok küçük ve çok büyük sayıları rahatça saklayabilme olanağı verir. Ancak bu uzayda çalışabilmek için, hesapta kullanılan işlemlerin logaritma uzayında karşılıklarını tanımlamak gerekir. Bir sayıyı logaritma uzayına taşımak kolaydır:

$$\bar{x} = \log x \tag{3.30}$$

Doğrusal uzaydaki bir sayının logaritma uzayındaki karşılığı, sayının logaritmasıdır. Logaritma uzayındaki bir sayıyı yeniden doğrusal uzaya döndürmek de kolaydır:

$$x = \exp \bar{x} \tag{3.31}$$

Oommen-Kashyap algoritmasında yalnızca iki işlem kullanılır: Toplama ve çarpma. Çarpma işleminin logaritma uzayındaki karşılığı toplama işlemidir:

$$\log(x \cdot y) = \bar{x} + \bar{y} \tag{3.32}$$

Bu yüzden çarpma işlemleri hiç bilgi kaybetmeden ve hatta doğrusal uzayda olduğunda daha hızlı bir şekilde gerçekleştirilebilir. Toplama işlemini logaritma uzayına taşımak ise aynı derecede kolay değildir. Gerçekte, sayıları yeniden doğrusal uzaya döndürmeden $\log(x+y) = F(\bar{x}, \bar{y})$ eşitliğini sağlayacak ilkel bir F fonksiyonu yoktur. Sayıları doğrusal uzaya döndürmek ise yeniden alttan taşma sorunları yaratacağından tüm girişimi anlamsız kılar.

Geçici olarak $x \geq y$ olduğunu varsayalım. Toplama sorununa çözüm şu gerçeğin bilinmesiyle gelir:

$$x \gg y \Rightarrow \log(x+y) \approx \log x \tag{3.33}$$

Eğer $\log(x)$ üzerine x ile y birbirine yakın olduğunda anlamlı olacak bir düzeltici terim eklenirse, eklenen terimin kesinliği ölçüsünde başarılı bir yaklaştırma elde edilir. Gereken düzeltme terimi logaritma fonksiyonunun kendisinden çıkar:

$$\log(x+y) = \log x + \log[1 + \exp(\log y - \log x)] \tag{3.34}$$

Dikkat edilirse, bu sefer de düzeltme teriminin hesabının doğrusal uzaya dönmeyi gerektirdiği görülür. Fakat bu sefer x veya y sayılarının değil, y/x sayısının kayan noktalı sayılarla temsili olması yeterlidir. x, y'den 2^{1022} kat daha büyük olmadığı sürece sayıların kendisi ne kadar küçük veya büyük olursa olsun y/x sayısı kayan noktalı olarak temsil edilebilir. Yine de, eğer x, y'den çok çok daha büyükse ve y/x sayısı kayan noktalı olarak temsil edilemiyorsa düzeltme terimi alttan taşma yüzünden 0 olarak hesaplanır. Ancak bu sefer bu taşma herhangi bir problem yaratmaz. Çünkü $y/x < 2^{-1022}$ ise düzeltme teriminden beklenen değer 2^{-1022}'den çok çok daha küçük olacaktır. Hesaplanan değer ile beklenen değer arasındaki fark, kayan noktalı bir sayı ile temsil edilemeyecek kadar küçüktür! $y > x$ olduğu durumda aynı kesinlikte cevaplar x ve y değişkenlerini takas ederek elde edilebilir. Toparlarsak; toplama işleminin logaritma uzayındaki karşılığı aşağıdaki gibi tanımlanır:

$$\log(x+y) = \begin{cases} \bar{x} + \log[1+\exp(\bar{y}-\bar{x})] & x \geq y \\ \bar{y} + \log[1+\exp(\bar{x}-\bar{y})] & y > x \end{cases} \tag{3.35}$$

Artık elimizde tüm işlemleri logaritma uzayında yapmak için gereken her şey var. Öyleyse bir önceki bölümde tanımlanan algoritma çok çok küçük olasılıkları bile yüksek kesinlikle hesaplayabilecek şekilde güncellenebilir: Bunun için tüm girdiler algoritmanın en başında, Eşitlik 3.30'de tanımlanan şekilde logaritma uzayına taşınır. Algoritmadaki tüm çarpma işlemleri Eşitlik 3.32'te belirtilen şekilde toplamalarla değiştirilir. Benzer şekilde, tüm toplama işlemleri de Eşitlik 3.35'te ifade edilen işlemlerle değiştirilir. Böylece Oommen-Kashyap algoritmasının uzun diziler için de çalışabilen yeni bir sürümü elde edilmiş olur. Bu sürümde algoritmanın çıktısı, beklenen çıktının logaritma uzayındaki karşılığı olacaktır. Eğer gerçek değer isteniyorsa (ve gerçek değer taşma yaratmayacaksa) çıktı doğrusal uzaya taşınabilir. Bu çalışmada geçiş olasılıkları zaten logaritması alındıktan sonra kullanıldığından (Bölüm 4) çıktıyı doğrusal uzaya taşımaya gerek duyulmamıştır.

4. PROTEİN İŞLEV KESTİRİMİNDE İKİNCİL YAPININ KATKISI

Bu bölümde, amino asit dizileriyle ikincil yapıları Bölüm 3.6'da anlatıldığı şekilde beraber hizalamanın protein işlev kestiriminini iyileştirip iyileştirmediği sınanacaktır. Sınamanın yapıldığı veri kümesinin nasıl oluşturulduğu, kullanılan nitelikler, sınıflandırıcı ve değerlendirme yöntemleri ilerleyen bölümlerde anlatılacak ardından deney sonuçları sunulacaktır.

4.1 Veri Kümesi

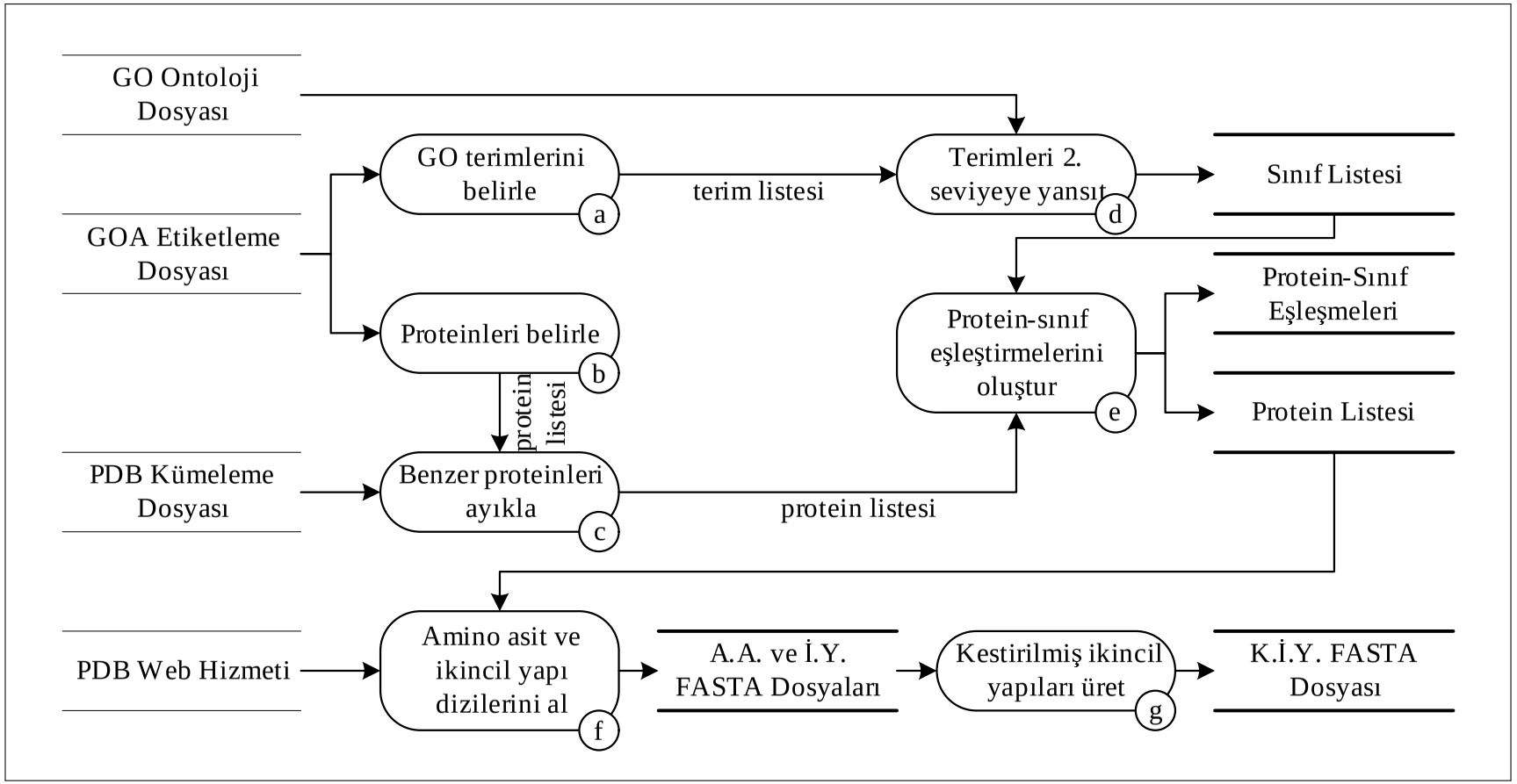
Yapılan literatür taramasında işlev kestirimi konusunda çokça kullanılıp standartlaşmış bir veri kümesine rastlanmadı. Bu yüzden bu çalışmada amaca uygun yeni bir veri kümesi hazırlandı. Veri kümesinin hazırlanması sırasında GO [15], GOA [24], PDB [25] projelerine ait web sitelerine ve PSIPRED [26] yazılımına başvuruldu. Veri hazırlama işine ilişkin veri akış diyagramı Şekil 4.1'te görülebilir. Her adımın ayrıntısı ilerleyen alt bölümlerde anlatılacaktır.

4.1.1 GOA etiketlemelerinin okunması

Gene Ontology Annotation (GOA) projesi, UniProt Knowledgebase (UniProtKB) [27] ve International Protein Index (IPI) [28] gibi geniş protein veritabanlarında listelenmiş proteinlerin Gene Ontology (GO) etiketletlerini tek bir noktada toplamayı amaçlayan bir projedir. VERİ5 veri kümesi üretildiği sırada proje kapsamında yedi adet cinsin proteomlarına ait etiketler bulunmaktaydı: İnsan, fare, sıçan, arabidopsis[1], zebra balığı, tavuk ve inek. GOA, protein etiketlerini İnternet üzerinden erişilebilecek sıkıştırılmış metin dosyaları hâlinde yayınlamaktadır[2] (Şekil 4.2). Tüm etiketler tek bir dosya olarak indirilebileceği

[1] Arabidopsis, turpgiller ailesine ait bir cinstir. Bu cinsin altında bulunan bir tür olan *arabidopsis thaliana*, biyoloji araştırmalarında sık sık model olarak kullanılır.

[2] ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/



Şekil 4.1: VERİ5 veri kümesinin nasıl hazırlandığını özetleyen veri akış diyagramı.

```
PDB 104L_A 104L_A GO:0016998 ···
PDB 104L_B 104L_B GO:0003796 ···
PDB 104L_B 104L_B GO:0009253 ···
```

Şekil 4.2: GOA tarafından sunulan bir etiketleme dosyasından alınmış bir parça.

gibi, cinslere göre veya veritabanlarına göre kategorilere ayrılmışdosyalar da indirilebilir.

VERİ5 veri kümesi hazırlanırken, belirlenen proteinlerin amino asit dizileri ve ikincil yapıları, başka bir protein veritabanı olan RCSB Protein Data Bank (PDB) sunucularından çekilecekti. Bu yüzden yalnızca PDB veritabanında numarası olan proteinlerin seçilmesi gerekiyordu. GOA'nın sunduğu ayrı dosyalardan birisi de bu iş için düşünülmüştür: GOA sunucularında yalnızca PDB numarası olan proteinleri içeren ayrı bir etiketleme dosyası bulunur[3]. Bu dosya 187339 satır içermektedir ve her satırında bir PDB numarası-GO terimi numarası eşleşmesi bulunur. Veri kümesi hazırlama işinin ilk ve ikinci (a ve b) adımlarında, bir PYTHON programı kullanılarak bu dosyadaki GO terimlerinin ve PDB numaralarının bir listesi oluşturuldu.

4.1.2 PDB kümelemeleri ile benzer proteinlerin ayıklanması

İkinci adımda (b) seçilen proteinlerin bir çoğu birbirine gerektiğinden fazla benzer. Yapılan çalışmalar, genel hizalaması yapıldığında %40'tan fazla oranda eşleşme içeren dizi çiftlerinin, diğer özelliklerinden bağımsız olarak eş kökenli olduğunu göstermiştir [29]. Bu yüzden, işlev kestirimi problemine gerçekçi bir çözümün, birbirine bu derecede benzeyen proteinler üzerinde sınanmaması gerekir.

Proteinler benzerliklerine göre ayıklanırken, PDB tarafından sunulan protein kümelemeleri (*clustering*) kullanılmıştır. Protein kümelemeleri, birbirine belli orandan fazla benzeyen proteinlerin gruplanmasıyla üretilir. Her grup, o grubu en iyi temsil eden proteinden en az temsil eden proteine doğru ilerleyen bir protein listesinden oluşur. Böylece, her grubun yalnızca en iyi temsilcisini alıp geri kalan üyelerini ayıklayarak birbirine benzeyen proteinlerden kurtulmak mümkün olur.

[3] ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/PDB/

```
2Q7Z_A 1HAQ_A 2GSX_A 1MVM_A ···
1YO8_A 2F83_A 2RHP_A 2GY5_A ···
2NYY_A 2NZ9_A 2NZ9_B 3BTA_A ···
```

Şekil 4.3: PDB tarafından sunulan BLASTCLUST ile hazırlanmış bir protein kümelemesi dosyasından bir parça.

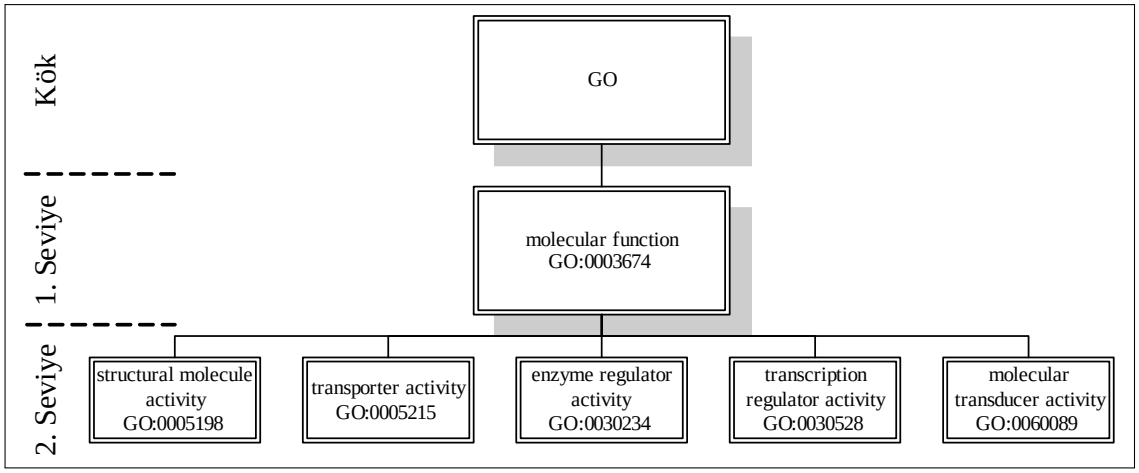
PDB, iki farklı algoritmayla oluşturulmuş 10 adet kümeleme sunmaktadır[4]. Bu kümelemelerin dört tanesi CD-HIT algoritmasıyla [30] %50, %70, %90 ve %95 oranında benzerlikler için oluşturulmuştur. CD-HIT algoritması görece daha anlamlı sonuçlar üretse de, ancak %50 ve daha yüksek orandan fazla benzeyen proteinleri kümeleyebilmektedir. Kalan altı kümeleme, BLASTCLUST algoritmasıyla [31] %30, %40, %50, %70, %90 ve %95 oranında benzerlikler için oluşturulmuştur[5] (Şekil 4.3). Proteinler ayıklanırken, BLASTCLUST ile hazırlanmış, %40 oranında benzeyen proteinleri gruplayan kümeleme kullanıldı. Bir PYTHON programı kullanılarak kümeleme dosyası okundu ve ikinci adımda (b) elde edilen protein listesindeki proteinlerden yalnızca grubunu en iyi temsil edenler seçilerek beşinci adıma (e) iletildi.

4.1.3 Sınıflara karar verilmesi

Bölüm 2.3'te bahsedildiği gibi, GO terimleri büyük bir çizgenin düğümleridir. Bu çizgede kök düğüm hariç her düğümün bir veya birkaç atası ("*is-a*" ilişkileri) ve bir veya birkaç kapsayanı ("*has-a*" ilişkileri) vardır. Hiyerarşide aşağılara inildikçe daha özelleşmiş terimlere varılır. İşe yarar bir işlev kestirim aracında terimlerin ne çok az örnek içerecek kadar özel ne de örneklerin büyük bölümünü içerecek kadar genel olması iyidir. Bu yüzden hiyerarşide belli bir seviye seçip, terimleri bu seviyedeki atalarıyla temsil etmek uygun olur.

Öncelikle, bu bir işlev kestirimi veri kümesi olacağından GO'nun üç alanından moleküler işlev (*molecular function*) altındaki terimler korundu, diğer iki alandaki (biyolojik etkinlik ve hücre bileşeni) terimler yok sayıldı. Elde kalan terimler, hiyerarşinin ikinci seviyesindeki atalarıyla temsil edildi. (Bu hiyerarşi bir ağaç oluşturmadığından, bir terimin birinci seviyede birden fazla atası olabilmektedir.

[4] ftp://ftp.rcsb.org/pub/pdb/derived_data/NR/
[5] ftp://ftp.rcsb.org/pub/pdb/derived_data/NR/blastclust/



Şekil 4.4: VERİ5 için seçilen beş GO teriminin GO ontolojisindeki yeri.

Bu tür durumlarda ataların hepsi dikkate alındı.) İkinci seviyedeki bu ata terimlerden beş tanesi sınıf olarak kullanılmak üzere seçildi (Şekil 4.4 ve Çizelge 4.1). Son olarak, üçüncü adımda (c) elde edilmiş olan ayıklanmış protein listesinden bu beş sınıftan en az birine ait olanlar seçilerek nihai 785 adet proteine ulaşıldı.

Çizelge 4.1: VERİ5 içindeki beş adet moleküler işlev sınıfı ve her bir sınıfın kapsadığı protein adedi.

GO numarası	Terim adı	Kısaltma	Protein adedi
GO:0005198	*structural molecule activity*	SMA	171
GO:0005215	*transporter activity*	TA	214
GO:0030234	*enzyme regulator activity*	ERA	127
GO:0030528	*transcription regulator activity*	TRA	208
GO:0060089	*molecular transducer activity*	MTA	119

4.1.4 Amino asit dizileri, ikincil yapılar ve kestirilmiş ikincil yapılar

Veri kümesinin kapsadığı proteinler kesinleştikten sonra, PDB sitesinin sunduğu web hizmetine (*web service*)[6] bir PYTHON programı ile bağlanılarak her bir proteinin amino asit dizisi ve ikincil yapısı okundu. Okunan diziler FASTA formatında iki ayrı dosyaya kaydedildi (Şekil 4.5). Kestirilmiş ikincil yapılar için PSIPRED yazılımı kullanıldı. PSIPRED'in varsayılan parametrelerle ürettiği kestirilmiş ikincil yapılar üçüncü bir FASTA dosyasına kaydedildi. Proteinlere ait amino asit, ikincil yapı ve kestirilmiş ikincil yapı dizilerini içeren FASTA dosyaları, nitelik üretimi aşamasında BALIGN yazılımına girdi olacaktır.

[6]http://www.rcsb.org/robohelp/webservices/summary.htm

```
>1A02:F
MKRRIRRERNKMAAAKSRNRRRELTDTLQAETDQLEDEKSALQTEIANLLKEKEKL
>1A02:N
MRGSHHHHHHTDPHASSVPLEWPLSSQSGSYELRIEVQPKPHHRAHYETEGSRGAV
KAPTGGHPVVQLHGYMENKPLGLQIFIGTADERILKPHAFYQVHRITGKTVTTTSY
EKIVGNTKVLEIPLEPKNNMRATIDCAGILKLRNADIELRKGETDIGRKNTRVRLV
FRVHIPESSGRIVSLQTASNPIECSQRSAHELPMVERQDTDSCLVYGGQQMILTGQ
NFTSESKVVFTEKTTDGQQIWEMEATVDKDKSQPNMLFVEIPEYRNKHIRTPVKVN
FYVINGKRKRSQPQHFTYHPV
```

Şekil 4.5: PDB web hizmetinden alınan dizilerle oluşturulmuş FASTA dosyasından bir parça.

4.2 Nitelik Üretimi

Mekanik öğrenmede nitelikler, ele alınacak nesneleri birbirinden ayırt etmeyi sağlayan ölçümlerdir. Sınıflandırıcılar nesneler üzerinde yapılan bu ölçümleri kullanarak hangi nesnenin hangi sınıfa ait olduğuna karar vermeye çalışırlar. Örneğin kadınları erkeklerden ayırmaya çalışan bir sınıflandırıcı saç uzunluğu, kilo, ayakkabı numarası gibi ölçümleri kullanabilir.

Proteinleri birbirinden ayırmak için de proteinler üzerinde bazı ölçümler yapmak gereklidir. Bunun için bir seçenek, proteinlerin birbirilerine olan benzerliklerini ölçmektir. Böylece amino asit dizisinin biyolojik, fiziksel veya kimyasal aytıntılarına yoğunlaşmadan nitelik matrisleri elde edilebilir. Bu yaklaşım Liao ve Noble tarafından katlanış tanıma problemine uygulanmıştır ve oldukça başarılı sonuçlar alınmıştır [9]. Liao ve Noble'ın çalışması hâlâ pek çok yeni biyoenformatik çalışmasına referans olmaktadır [32, 33]. Proteinler arası benzerlik ölçümüne yoğunlaşmış bu çalışmada da –beklendiği gibi– proteinleri birbirilerine olan benzerlikleriyle niteleme yaklaşımı benimsendi.

Çalışmanın bu aşamasında, benzerlik ölçümü için Bölüm 3.6'da tanımalanan birleşik Smith-Waterman algoritması kullanıldı. Hizalamalar sırasında amino asit dizileri $\texttt{BLOSUM}_{62}$ matrisiyle, ikincil yapılar `WALLQVIST` matrisiyle puanlandırıldı. Her protein çifti toplamda on farklı şekilde hizalandı ve elde edilen hizalanma puanlarıyla on farklı nitelik matrisi üretildi. On hizalamanın beşinde gerçek ikincil yapılar, beşinde PSIPRED yazılımıyla kestirilmiş ikincil yapılar kullanıldı.

Her durumda, ikincil yapının hizalamaya katılma oranı α, 0, 0.25, 0.5, 0.75 ve 1 değerlerini alacak şekilde beş hizalama yapıldı.

Nitelik matrisleriyle ilgili son bir noktaya değinmek gerekir. Gerek Needleman-Wunsch, gerekse Smith-Waterman algoritmaları, geri kalan her şey sabit tutulduğunda, uzun diziler için görece yüksek, kısa diziler için görece düşük puanlar üretme eğilimindedir. Bu, elde edilen nitelik matrisinde bir dengesizlik yaratır. Bu dengesizliği gidermek için, nitelik matrisleri sınıflandırıcıya verilmeden hemen önce bir normalleştirmeye tabi tutuldu. Öyle ki; bir $F = (f_{i,j})$ nitelik matrisi, normalleştirmeden sonra şu $C = (c_{i,j})$ matrisine dönüşür:

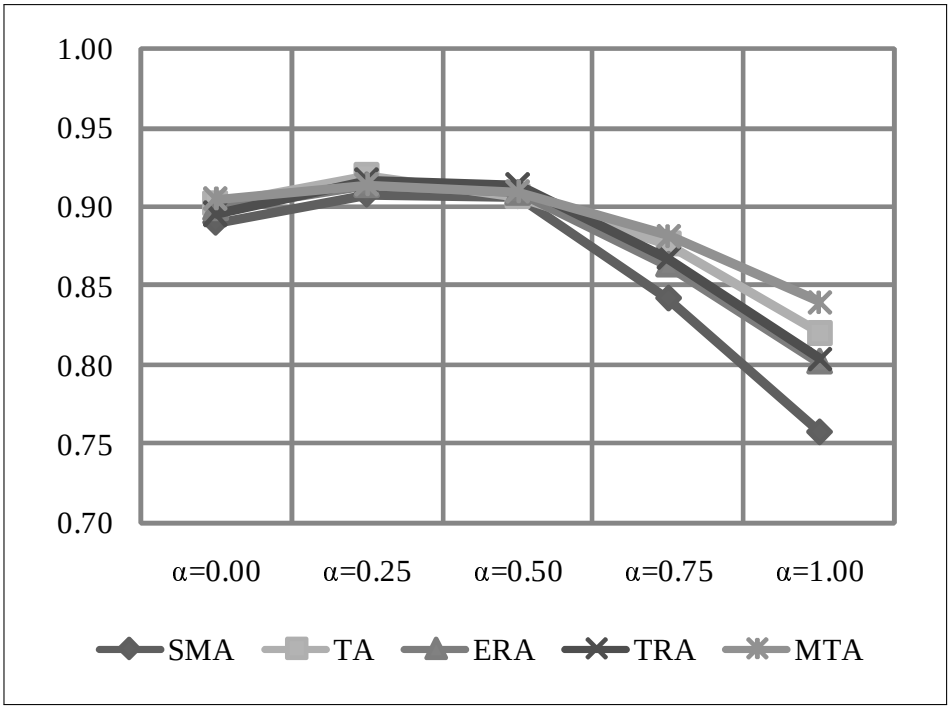
$$c_{i,j} = \frac{m_{i,j}}{\max\{m_{i,i}, m_{j,j}\}} \tag{4.1}$$

Bu hesabın gereği olarak, muhafaza puanları $[0,1]$ aralığında değer alırlar ve köşegen üzerinde her zaman 1'dir. Yine hesaba göre, uzun dizilerin kendilerine benzerlikleri yüksek olacağından nitelik vektörleri büyük bir sayıya, kısa dizilerin kendilerine benzerlikleri düşük olacağından nitelik vektörleri küçük bir sayıya bölünür. Bu şekilde dizi uzunluğuna bağlı dengesizlik giderilmiş olur.

4.3 Sınıflandırma ve Değerlendirme

On nitelik matrisinin bir karşılaştırmasını yeterince başarılı ve hızlı şekilde yapabilmek için, sınıflandırma algoritması olarak *en yakın komşu (nearest neighbor – NN)* algoritması kullanıldı. Bu algoritma, sınama örneklerini nitelik uzayındaki konumlarını kullanarak, en yakınlarındaki eğitim örnekleriyle ilişkilendirir. Her sınama örneği, ilişkili olduğu eğitim örneğinin sınıfına atanır. Her sınama örneğinin en yakınındaki eğitim örneğini bulmaktan ibaret bu algoritma, en kötü durumda $O(n_{\text{sınama}} \cdot n_{\text{eğitim}})$ zamanda tamamlanır.

Üzerinde çalışılan veri kümesinin doğası gereği bir protein birden fazla sınıfta bulunabildiğinden, sınıflandırmada *bire karşı hepsi* (*one against all*) yordamı izlendi. Buna göre her bir sınıf için ayrı bir sınıflandırıcı oluşturuldu. Her bir sınıflandırıcı, yalnızca ele alınan sınıftan ibaret bir pozitif sınıf ve geri kalan sınıfların hepsini kapsayan bir negatif sınıf varmış gibi eğitildi. Bu sayede bir proteinin, gerçekte olduğu gibi, birden fazla sınıfa yerleştirilebilmesi mümkün kılındı.



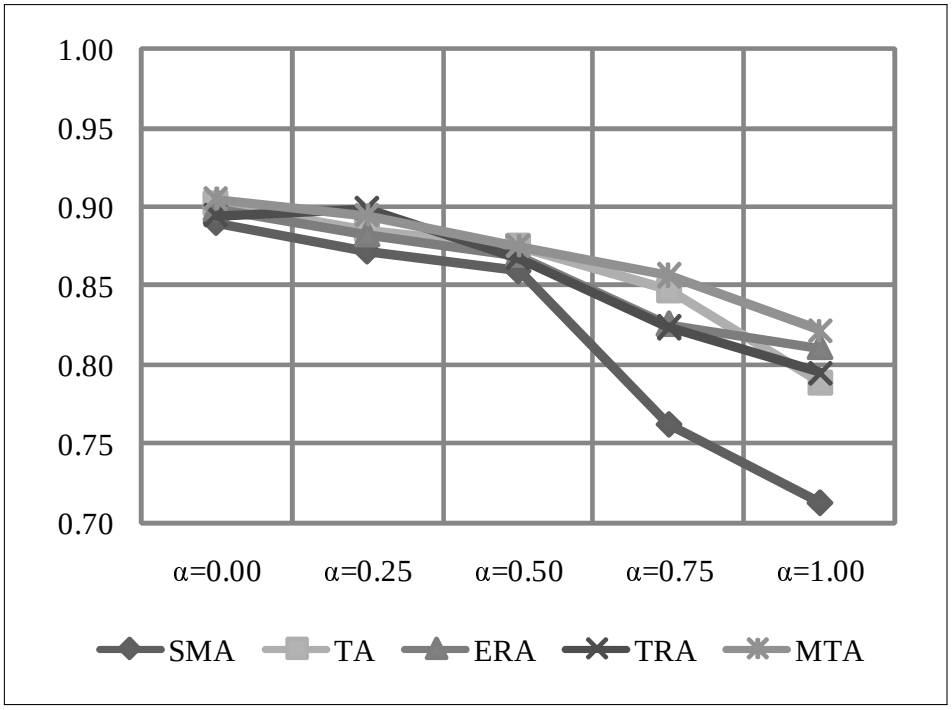
Şekil 4.6: Gerçek ikincil yapının farklı oranlarda kullanımının kestirim başarısına etkisi.

Performansının ölçümünde *çapraz doğrulama (cross-validation)* tekniğine başvuruldu. Bunun için, veri kümesi pozitif ve negatif sınıfların dengesi korunacak şekilde 10 parçaya ayrıldı (bu iş her sınıf/sınıflandırıcı için ayrı ayrı yapıldı). Yapılan 10 farklı testin her birinde parçalardan yalnız bir tanesi sınama kümesi olarak, kalan dokuz tanesi eğitim kümesi olarak kullanıldı. Bu şekilde her parçanın bir defa sınama kümesinde, dokuz defa eğitim kümesinde bulunması sağlandı. Raporlanan çıktılar, bu 10 sınama üzerinden hesaplanan ortalamalardır.

Performans kriteri olarak *ROC (receiver operating characteristic) eğrisinin altında kalan alan (area under ROC curve – AUC)* kullanıldı [34]. ROC eğrisi, basitçe, sınıflandırıcı tarafından üretilmiş sınıf olasılıklarının farklı eşik

Çizelge 4.2: NN algortimasının *gerçek* ikincil yapılar ve 5 sınıf için ürettiği ortalama AUC değerleri. Her satırdaki en yüksek değer koyu harflerle vurgulanmıştır.

Sınıf	$\alpha = 0.00$	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 0.75$	$\alpha = 1.00$
SMA	0.890	**0.908**	0.907	0.842	0.758
TA	0.902	**0.920**	0.907	0.877	0.821
ERA	0.899	**0.914**	0.909	0.863	0.802
TRA	0.895	**0.916**	0.914	0.868	0.804
MTA	0.905	**0.914**	0.910	0.882	0.840



Şekil 4.7: Gerçek ikincil yapının farklı oranlarda kullanımının kestirim başarısına etkisi.

değerlerinde nasıl sonuç vereceğini gösteren bir grafiktir. AUC, ROC eğrisinin altında kalan alan olduğu gibi, aynı zamanda sınıflandırıcının ayırt etme yeteneğini ortaya koyan bir sayıdır. Öyle ki; AUC değerinin 0.5 ölçülmesi, sınıflandırıcının hiçbir bilgi üretmediği anlamına gelir. Aksine AUC değeri 1 ise, sınıflandırıcı eşik değerinden bağımsız olarak kusursuz sınıflandırma yapmış demektir. Başarılı bir sınıflandırıcının 0.5'ten büyük, 1'e yakın AUC değerleri üretmesi beklenir.

Tüm sınıflandırma ve değerlendirme işleri MATHWORKS MATLAB yazılımı [35] üzerinde PRTOOLS örüntü tanıma paketi [36] kullanılarak yapıldı.

Çizelge 4.3: NN algortimasının *kestirilmiş* ikincil yapılar ve 5 sınıf için ürettiği ortalama AUC değerleri. Her satırdaki en yüksek değer koyu harflerle vurgulanmıştır.

Sınıf	$\alpha = 0.00$	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 0.75$	$\alpha = 1.00$
SMA	**0.890**	0.872	0.860	0.763	0.713
TA	**0.902**	0.886	0.875	0.847	0.789
ERA	**0.899**	0.883	0.870	0.826	0.811
TRA	0.895	**0.900**	0.868	0.825	0.796
MTA	**0.905**	0.895	0.875	0.857	0.822

Çizelge 4.4: Gerçek ikincil yapılar ile $\alpha = 0.25$ için alınan sonuçların, $\alpha = 0$ için alınanlardan yüksek olup olmadığını sınayan t-testleri. Hassasiyet %95 olarak seçilmiştir.

Sınıf	$AUC_{\alpha=0.25} > AUC_{\alpha=0.0}$	p-değeri
SMA	**doğru**	0.03
TA	**doğru**	0.01
ERA	yanlış	0.11
TRA	**doğru**	0.00
MTA	yanlış	0.07

4.4 Deney Sonuçları

NN algoritması gerçek ikincil yapılar kullanılarak oluşturulmuş 5 nitelik matrisi ve 5 sınıf üzerinde çapraz doğrulama ile çalıştırılmıştır. Buradan elde edilen ortalama AUC değerleri ve bu değerlerin değişim grafiği Çizelge 4.2 ve Şekil 4.6'da görülebilir. Açıkça görülüyor ki; ikincil yapının katkısı, α, 0.25 değerini aldığında tüm sınıflardaki kestirim başarısı artıyor. Ancak, başarı α'nın daha büyük değerleri için düşüyor ve sonunda ikincil yapının hiç kullanılmadığı $\alpha = 0$ durumundaki başarının altına iniyor.

PSIPRED yazılımıyla kestirilmiş ikincil yapıların kullanıldığı deneylerin sonuçları Çizelge 4.3 ve Şekil 4.7'de görülebilir. Görüldüğü üzere, kestirilmiş ikincil yapı TRA sınıfı dışında tüm sınıfların kestirim başarısını dosdoğru aşağı çekiyor. TRA sınıfında kestirilmiş yapının hizalamaya 0.25 oranında katılması, hiç katılmamasından daha sonuç veriyor; fakat bu sonuç da gerçek ikincil yapıyla elde edilenden iyi değil. Ayrıca ERA sınıfı için sadece ikincil yapıların kullanıldığı $\alpha = 1$ durumunda, kestirilmiş yapının gerçek ikincil yapıdan daha iyi sonuç verdiği kayda değer.

Gerçek ikincil yapının başarıya görünür katkısını istatistikî olarak sınayan %95 hassasiyetli t-testi sonuçları Çizelge 4.4'te görülebilir. t-testi, gerçek ikincil yapı hizalamaya 0.25 oranında katıldığında alınan sonuçları, ikincil yapı hizalamaya hiç katılmadığında alınan sonuçlardan yüksek olup olmadığını sınamak için kullanıldı. Buna göre, %95 hassasiyet için, gerçek ikincil yapı kullanımının SMA, TA ve TRA sınıflarında istatistikî olarak anlamlı bir başarı artışına yol açtığı söylenebilir.

Elde edilen sonuçların tartışması Bölüm 6'da yapılacaktır.

5. GEÇİŞ OLASILIKLARI İLE PEPTİT SINIFLANDIRMA

Çalışmanın ikinci deneysel aşamasında, Bölüm 3'te bahsedilen Oommen-Kashyap geçiş olasılıklarının biyolojide de kullanılabileceğini göstermek üzere iki adet peptit sınıflandırma problemi ele alındı. Peptit sınıflandırma problemi üzerinde on yıllardır çalışma yapılan bir problemdir. 1998 yılında, Cai ve Chou [37] bu problemin çözümü yolunda önemli bir adım attılar. Her bir yapıtaşına karşılık 20 adet girdi düğümü içeren, toplam 160 girdili bir yapay sinir ağı (*artificial neural network*) tasarladır. Bu sinir ağı üzerinde peptitleri 20 bitlik ortonormal gösterimle temsil ederek ($A = 100\ldots00, B = 010\ldots00, \ldots, Y = 000\ldots01$) bir sınıflandırıcı eğittiler. Zhao ve arkadaşları [38], benzer şekilde, her bir amino asit üzerinde 10 ölçüm yaparak nitelik vektörleri oluşturdular ve bu vektörler üzerinde destek vektörü makineleri (*support vector machine*) ile sınıflandırma yaptılar. Yapılan ölçümler, amino asitlerin su severlik, β yapısı tercihi gibi biyokimyasal özelliklerine dayanıyordu. Zhao ve arkadaşları, 20 bitlik ortonormal gösterim yerine 10 biyokimyasal özelliğe dayanan bu gösterimi kullanarak, nitelik uzayını yarı yarıya küçültmeyi başardılar. Nitelik vektörlerinin bilgi içeriğini daha da arttırabilmek adına Thomson ve arkadaşları [39] biyolojik benzerlik tabanlı (*bio-basis*) yapay sinir ağlarını ortaya attılar. Bu sinir ağları, radyal uzaklık tabanlı (*radial-basis*) sinir ağlarını örnek alıyor, fakat radyal uzaklık yerine biyolojik benzerlikleri koyuyordu. Trudgian ve Yang [40], bu yaklaşımı, benzerlik hesabında kullanılan puanlama fonksiyonlarını iyileştirmek suretiyle ileriye götürdüler. Bunlara ek olarak, Kim ve arkadaşları [41] kural tabanlı sınıflandırma yöntemleri kullanarak yorumlanabilir sonuçlara ulaşmaya çalıştılar. Tüm bu çalışmalara öncülük etmiş, nicel matris (*quantitative matrix*) özelliklerine [42], bağlanma motiflerine (*binding motif*) [43] ve gizli Markov modellerine (*hidden Markov model*) [44] dayalı çalışmalar da peptit sınıflandırma literatürünün bir parçası olarak sayılabilir.

Amino asit dizilerini nitelendirmenin bambaşka bir yolu, diğer bir biyolojik dizi analizi problemi olan katlanma tanıma için Liao ve Noble [9] tarafından ortaya atılmıştır. Liao ve Noble, SVM-Pairwise adını verdikleri sınıflandırıclarında, dizileri, standart hizalama algoritmalarıyla ölçülen karşılıklı benzerlikleriyle temsil ettiler. Neticede, bu nitelik vektörlerini kullanan SVM-Pairwise sınıflandırıcısının katlanma tanıma sorununa başarılı bir çözüm getirdiğini gösterdiler.

Burada yapılacak olan, SVM-Pairwise benzeri bir sınıflandırıcıyı peptit sınıflandırma için kullanmak olacak. Fakat nitelikler, hizalanma puanları yerine Oommen-Kashyap geçiş olasılıklarından üretilecek. Böylece, Oommen-Kashyap geçiş olasılıklarının üstün özelliklerinin peptit sınıflandırma sorununa daha iyi bir çözüm getirip getirmediği gözlenecek. Bunun için, iki veri kümesi üzerinde, geçiş olasılıklarını kullanan sınıflandırıcılar ile Needleman-Wunsch hizalanma puanlarını kullanan sınıflandırıcılar karşılaştırılacak. Bu veri kümelerinin bir tanesi HIV-1 proteaz yarılma bölgesi (*cleavage site*) belirleme, diğeri ise T-hücresi epitopu belirleme problemleri için hazırlanmış olacak. Neticede, Oommen-Kashyap geçiş olasılıklarının, ele alınan problemler için çok başarılı sonuçlar verdiği ortaya koyulacak.

5.1 Veri Kümeleri

Protein işlev kestiriminin aksine, peptit sınıflandırma için sık kullanılmış birkaç veri kümesi bulunuyor. Deneylerde iki veri kümesi kullanıldı. Bunlardan ilki HIV-1 proteaz yarılma bölgesi belirleme problemi için Kim ve arkadaşları [41] tarafından sunulan HIV veri kümesidir. HIV-1 proteaz, AIDS hastalığına yol açan HIV virüsünün ürettiği bir enzimdir. Bu enzim, insan hücrelerindeki proteinlere belli yerlerden bağlanarak onları yarıp parçalar. Oluşan parçalar yeni HIV virüslerinin üretiminde kullanılır. HIV-1 proteaz enziminin proteinlere bağlanmayı seçtikleri yerlere *yarılma bölgesi* denir. Bu bölgelerin başarıyla tespit edilmesi, HIV virüsüne karşı bağışıklık kazandıracak aşıların tasarlanmasına yardımcı olur. HIV veri kümesinde 754 adet, 8 yapıtaşı içeren peptit bulunur. Bu peptitlerin 396 tanesi yarılma bölgesi olarak işaretlenmiştir. Kalan 358 peptit negatif örnekleri oluşturur.

İkinci veri kümesi, T-hücresi epitopu belirleme problemi için Zhao ve arkadaşları [38] tarafından sunulmuş TCL veri kümesidir. T-hücreleri, dolaşım ve lenf sisteminde bulunan ve antijenlere bağlanarak bağışıklığı tetikleyen hücrelerdir. T-hücreleri her maddeye değil, yalnızca belli amino asit dizilerini içeren *T-hücresi epitoplarına* bağlanır. Verilen bir peptidin T-hücresi epitopu olup olmadığına karar vermek, yine aşı üretimi gibi bağışıklık sistemini ilgilendiren meseleler için önemlidir. Bu meselelere bir örnek için çölyak hastalığı incelenebilir [45]. TCL veri kümesinde T-hücresi epitopu olduğu bilinen 36, T-hücresi epitopu olmadığı bilinen 167 peptit bulunur. Bu 203 peptidin hepsi 10 yapıtaşı içermektedir.

5.2 Nitelik Üretimi

Oommen-Kashyap geçiş olasılıklarını hesaplamak için önce modelin parametrelerine karar vermek gerekir. Modelin iki parametresi vardır: Eklenecek karakter sayısı dağılımı G ve değiştirme dağılımı S. Ele alınan modelin benzerliğinden dolayı, değiştirme dağılımının seçilmesinde [19] tarafından oluşturulmuş PAM matrisleri temel alındı. Önceden tanımlandığı gibi, $\texttt{PAM}_1$ matrisi 20×20 boyutunda bir matristir ve her bir hücresi, $\texttt{pam}_{1,i,j}$, amino asitlerin %1'i mutasyona uğradıktan sonra, i. amino asidin j. amino asit ile değiştirilmiş olması olasılığını verir. $\texttt{PAM}_1$ matrisini kendisiyle çarparak daha uzun mutasyon serileri için $\texttt{PAM}_{100}$, $\texttt{PAM}_{250}$ gibi matrisler elde etmek mümkündür (bkz. Bölüm 3).

Bir eksiklik olarak, PAM matrisleri, amino asitlerin silinmesi veya eklenmesi olasılıklarını içermez. Oysa S dağılımını elde etmek için silinme ve eklenme anlamına gelen boşluk karakteri için de bir satır ve bir sütuna ihtiyaç vardır. Bu yüzden, $\texttt{PAM}_1$ matrisine $-$ karakterine karşılık gelen yeni bir sütun ve satır eklendi. Her proteinin silinme ($-$ ile değiştirilme) olasılığı eşit, d, kabul edilerek, her $a \in \Sigma$ için $-$ sütununa d değeri yazıldı ve matris aşağıdaki şartı koruyacak şekilde normalleştirildi:

$$\sum_{b \in \Sigma \cup \{-\}} S(b|a) = 1 \qquad \textbf{(5.1)}$$

Literatürde amino asitlerin silinme olasılığı d'ye karar vermenin standart bir yoluna rastlanmadı. Bu yüzden sık kullanılan boşluk cezaları ile yapılan bir karşılaştırmayla $d = 0.0001$ olarak seçilmesine karar verildi.

Eklenme olasılıkları için eklenen satırın doldurulması için amino asitlerin gözlenme sıklıklarına başvuruldu. Her bir $b \in \Sigma$ amino asidinin eklenme olasılığı $S(b|-)$, b amino asidinin gözlenme sıklığı olan $f(b)$'ye eşitlendi. Gözlenme sıklıkları $\texttt{PAM}_n$ matrislerinin n sonsuza giderkenki limiti kullanılarak hesaplandı. Çünkü limit $\texttt{PAM}_{n\to\infty}$ matrisinde her satır f fonksiyonuna yakınsar.

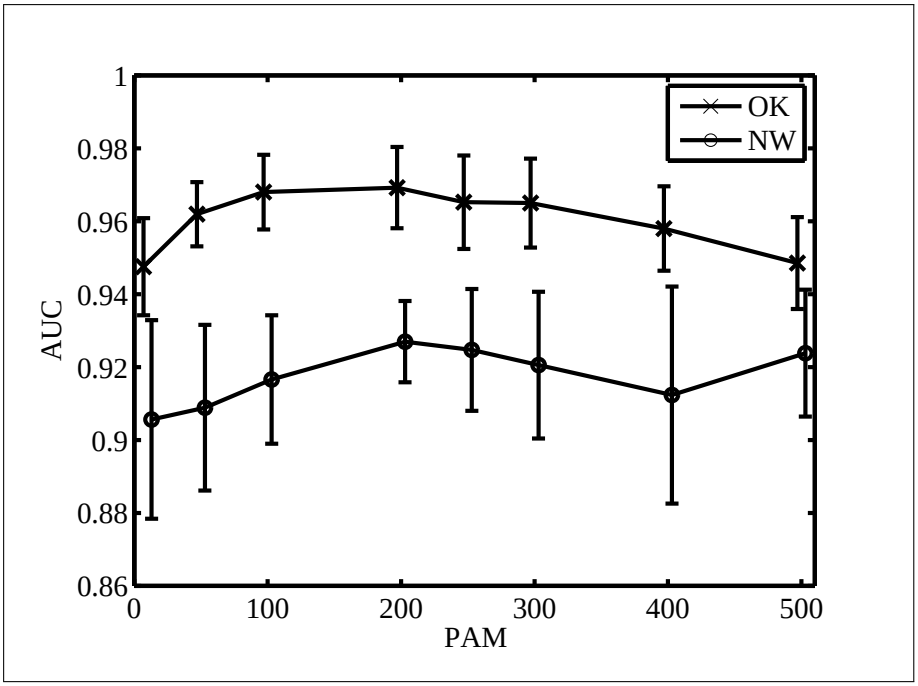
Böylece yeni matriste geriye yalnızca $S(-|-)$ olasılığına denk düşen hücre kaldı. Bu hücreye de tanım gereği 0 değeri atandı ve 21×21 boyutunda yeni bir matris elde edildi. Yeni matris, Oommen adına atfen, $\texttt{OPAM}_1$ olarak adlandırıldı. Dikkat edilirse, tıpkı $\texttt{PAM}_1$ matrisi gibi, $\texttt{OPAM}_1$ matrisini de kendisiyle çarparak uzun mutasyon serileri için yeni matrisler elde etmek mümkündür. Örneğin $\texttt{OPAM}_{250}$, $\texttt{OPAM}_{249} \times \texttt{OPAM}_1$'e eşittir.

Oommen-Kashyap modelinin ikinci parametresi, eklenecek karakter sayısını belirleyen G dağılımıdır. G'nin belirlenmesi sırasında, bir PAM'a karşılık gelen mutasyon serisi boyunca bir amino asit eklenme olasılığı, silinme olasılığı d'ye eşit kabul edildi. Daha uzun seriler için bunun anlamı, eklenecek karakter sayısı dağılımının aşağıdaki gibi bir Poisson dağılımına yakınsayacak olmasıdır:

$$G_{n,d}(z) = \texttt{Poisson}(z; n \cdot d) = \frac{(n \cdot d)^z e^{-n \cdot d}}{z!} \qquad \textbf{(5.2)}$$

Burada n, PAM serisinin uzunluğunu belirler. Diğer bir deyişle, değiştirme dağılımı S, $\texttt{OPAM}_n$ kullanılarak oluşturulduğunda, eklenecek karakter sayısı dağılımı $G(z)$, $\texttt{Poisson}(z; n \cdot d)$ olarak seçildi.

Parametreler yukarıda anlatıldığı gibi seçildikten sonra, BALIGN yazılımı ile her iki veri kümesindeki peptitler için Oommen-Kashyap geçiş olasılıkları ve Needleman-Wunsch hizalanma puanları hesaplandı. Mutasyon serisi hakkındaki varsayımın sonuçları nasıl etkilediğini anlamak amacıyla hesaplar sekiz PAM/OPAM matrisiyle tekrarlandı. 10, 50, 100, 200, 250, 300, 400 ve 500 uzunluğunda mutasyon serileri için PAM ve OPAM matrisleri kullanılarak, geçiş olasılıklarıyla 8, genel hizalanmayla 8 olmak üzere, her bir veri kümesinde toplam 16 nitelik matrisi elde edildi.



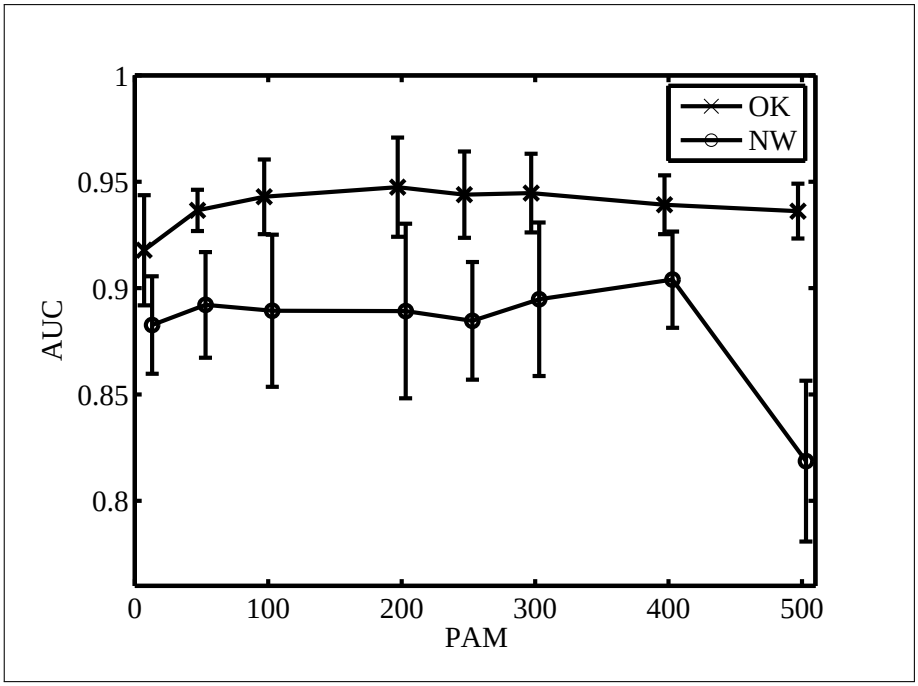
Şekil 5.1: HIV veri kümesinde, mutasyon uzunluğu varsayımının OK ve NW başarısı üzerine etkisi. Hata çubukları %95 güven aralıklarını göstermektedir.

5.3 Sınıflandırma ve Değerlendirme

Üretilen 32 adet nitelik matrisi, sınıflandırma için doğrusal çekirdek fonksiyonlu destek vektörü makinelerine (*support vector machine – SVM*) verildi. Bir önceki deneyde olduğu gibi, bu deneyde de çapraz-doğrulama yapıldı. Bunun için

Çizelge 5.1: HIV veri kümesi üzerinde Oommen-Kashyap geçiş olasılıkları (OK) ve Needleman-Wunsch hizalanma puanları (NW) için başarı değerleri. Her sütunda en büyük değer koyu harfler ile vurgulanmıştır. Son satır, ölçümlerin %95 güven aralığı genişliklerinin (w) ortalamasını göstermektedir.

(O)PAM	OK				NW			
	AUC	Acc	Sens	PPV	AUC	Acc	Sens	PPV
10	0.948	0.887	0.863	0.884	0.906	0.839	0.821	0.837
50	0.962	0.902	0.891	0.904	0.909	0.849	0.841	0.843
100	0.968	**0.917**	**0.897**	0.927	0.917	0.846	**0.846**	0.833
200	**0.969**	0.911	0.877	0.932	**0.927**	**0.857**	0.833	**0.862**
250	0.965	0.913	0.874	0.938	0.925	0.853	0.830	0.857
300	0.965	0.911	0.863	**0.948**	0.921	0.849	0.829	0.852
400	0.958	0.901	0.849	0.937	0.912	0.849	0.838	0.848
500	0.949	0.893	0.830	0.938	0.924	0.846	0.813	0.859
Ort. w	0.011	0.018	0.037	0.021	0.019	0.025	0.040	0.029



Şekil 5.2: TCL veri kümesinde, mutasyon uzunluğu varsayımının OK ve NW başarısı üzerine etkisi. Hata çubukları %95 güven aralıklarını göstermektedir.

HIV veri kümesi 10 parçaya, görece küçük olan TCL kümesi ise 5 parçaya ayrıldı. Her bir sınamada sınıflandırıcının çıktıları üzerinde ROC eğrisinin altında kalan alan (AUC), doğruluk (*accuracy* – *Acc*), duyarlılık (*sensitivity* – *Sens*) ve pozitif kestirim değeri (*positive predictive value* – *PPV*) ölçüldü. AUC değerinden bir önceki bölümde bahsedilmişti. Doğruluk, sınıflandırcının doğru sınıflandırdığı örneklerin sayısının, toplam örnek sayısına oranıdır; örneklerin

Çizelge 5.2: TCL veri kümesi üzerinde Oommen-Kashyap geçiş olasılıkları (OK) ve Needleman-Wunsch hizalanma puanları (NW) için başarı değerleri. Her sütunda en büyük değer koyu harfler ile vurgulanmıştır. Son satır, ölçümlerin %95 güven aralığı genişliklerinin (w) ortalamasını göstermektedir.

(O)PAM	STP				NW			
	AUC	Acc	Sens	PPV	AUC	Acc	Sens	PPV
10	0.918	0.852	0.922	0.901	0.883	0.837	**0.928**	0.882
50	0.937	0.872	0.934	0.912	0.892	0.842	0.922	0.891
100	0.943	0.882	0.929	0.928	0.889	0.847	0.922	0.895
200	**0.947**	0.897	0.940	0.935	0.889	0.853	0.905	0.917
250	0.944	**0.902**	**0.946**	**0.936**	0.885	0.853	0.893	0.927
300	0.945	0.887	0.940	0.924	0.895	0.852	0.916	0.905
400	0.939	0.887	**0.946**	0.919	**0.904**	**0.867**	0.911	**0.928**
500	0.936	0.882	0.929	0.928	0.819	0.793	0.881	0.871
Ort. w	0.016	0.023	0.022	0.020	0.028	0.030	0.041	0.021

Çizelge 5.3: %95 hassasiyetle yapılmış t-testi sonuçları. Testler OK için elde edilen AUC değerlerinin NW için elde edilenlerden büyük olup olmadığını sınamaktadır.

(O)PAM	HIV		TCL	
	$AUC_{OK} > AUC_{NW}$	p-değeri	$AUC_{OK} > AUC_{NW}$	p-değeri
10	**doğru**	0.013	**doğru**	0.018
50	**doğru**	0.001	**doğru**	0.025
100	**doğru**	<0.001	**doğru**	0.047
200	**doğru**	<0.001	**doğru**	0.014
250	**doğru**	<0.001	**doğru**	<0.001
300	**doğru**	<0.001	**doğru**	0.015
400	**doğru**	0.012	**doğru**	0.001
500	**doğru**	0.014	**doğru**	0.001

kaçta kaçının doğru sınıflandırıldığını ölçer. Duyarlılık, yalnızca pozitif örneklerin kaçta kaçının doğru sınıflandırıldığını ölçer. Pozitif kestirim değeri ise pozitif olarak sınıflandırılmış örneklerin kaçta kaçının pozitif olduğunu söyler. Yapılan 32 sınamada ölçülen değerler bir sonraki bölümde verilmiştir.

5.4 Deney Sonuçları

Sınamaların sonucunda, Oommen-Kashyap geçiş olasılıklarının (OK) başarısı Needleman-Wunsch hizalanma puanlarının (NW) başarısıyla kıyaslandı. Her sınamada sınıflandırıcı çıktılarında ROC altında kalan alan (AUC), doğruluk (Acc), duyarlılık (Sens) ve pozitif kestirim değeri (PPV) ölçüldü. Çizelge 5.1 ve 5.2, çapraz doğrulama ile ölçülen değerlerin ortalamalarını ve ortalama %95 güven aralığı genişliklerini göstermektedir. Görülebileceği gibi güven aralığı genişlikleri, TCL veri kümesinde HIV kümesinde olduğundan fazladır. Bunun nedeni TCL veri kümesinde çapraz doğrulamanın 10 yerine 5 parça üzerinden yapılmış olmasıdır. Bunun dışında, çizelgeler OK başarısının NW başarısından ölçüm kriteri ve veri kümesi farkı gözetmeden yüksek olduğunu göstermektedir. OK için ölçülen AUC değerleri HIV veri kümesi iiçin 0.96'yı, TCL veri kümesi için 0.94'ü bulmaktadır.

PAM ve OPAM puanlama matrislerinin sınıflandırma başarısı üzerine etkisi Şekil 5.1 ve 5.2'de görülebilir. Bu iki grafik, mutasyon uzunlukları üzerinde yapılan varsayım 10 PAM'dan 500 PAM'a doğru artarken AUC değerlerinin

nasıl değiştiğini göstermektedir. Grafiklere bakılarak, HIV veri kümesinde hem OK'nin hem de NW'nun en yüksek değerlerine 100 ve 300 PAM arasında ulaştığı görülebilir. TCL veri kümesinde ise NW, $\texttt{PAM}_{400}$ matrisini yeğlemektedir.

Görüldüğü gibi ortalama AUC değerleri ele alındığında OK, en kötü PAM seçimleri için bile NW'dan daha başarılıdır. Bu iddianın istatistikî olarak desteklenmesi için bir takım t-testleri de yapılmıştır. Çizelge 5.3, her PAM değeri için OK başarısının, NW başarısından yüksek olup olmadığını sınayan t-testi sonuçlarını içermektedir.

Alınan sonuçların tartışması ve literatür karşılaştırması Bölüm 6'de yapılacaktır.

6. TARTIŞMA VE SONUÇ

Çalışmanın sonucunda iki ayrı problem üzerinde yapılan deneylerden önemli sonuçlar elde edildi. Bu sonuçlara tekrar dönmek gerekirse; öncelikle ikincil yapının protein işlev kestirimine katkısı araştırıldı. Bu aşamada ikincil yapının *katlanış tanımaya* katkısını inceleyen Wallqvist ve arkadaşlarının [2] çizdiği yol izlendi: Smith-Waterman hizalama algoritması, amino asit dizileriyle ikincil yapı dizilerini aynı anda hizalayacak şekilde genelleştirildi. Ayrıca, üzerinde çalışılacak, 785 proteinli ve her biri GO moleküler işlev terimlerine karşılık gelen 5 sınıflı bir işlev kestirimi veri kümesi, VERİ5, farklı kaynaklardan derlenen bilgiler dikkate alınarak oluşturuldu. Algoritma, veri kümesi üzerinde çalıştırılarak ayrı ayrı gerçek ikincil yapıların ve kestirilmiş ikincil yapıların 0, 0.25, 0.5, 0.75 ve 1 oranında hesaba katıldığı farklı nitelik matrisleri yaratıldı ve bu matrisler üzerinde en yakın komşu yöntemiyle sınıflandırmalar yapıldı. Alınan sonuçlar, kestirilmiş ikincil yapıların ele alınan durum için protein işlev kestirimine katkısı olmadığını ortaya koydu. Yine deney sonuçları, gerçek ikincil yapının 0.25 oranın hizalamaya katılmasının kestirim başarısını anlamlı oranda arttırdığını gösterdi. Bu gözlem t-testleri ile desteklendi.

Varılan noktada, gerçek ikincil yapıların protein işlev kestirimine açık bir katkısı olduğu söylenebilir. Bu katkı muhtemelen üst seviyedeki yapının işlev üzerinde alt seviyedeki yapılara göre daha fazla etkisi olmasından kaynaklanmaktadır. Öyleyse bu etkiyi daha iyi ortaya çıkarmanın yolları düşünülebilir. Örneğin, ikincil yapıları bu çalışmadaki gibi birer kara kutu olarak ele almak yerine, fiziksel ve kimyasal özelliklerini ortaya çıkaracak şekilde ele almak çok daha iyi sonuçlar verebilir. Sadece sarmalların genellikle uzama eğiliminde olduğunu ve şeritlerin dizi üzerinde eşleri olması gerektiğini bilmek bile birleşik hizalamayı oldukça geliştirebilecek değişikliklere kapı açacaktır.

Ele alınan ikinci problemden alınan sonuçlar ise çok daha etkileyicidir. İkinci olarak, Oommen ve Kashyap'ın [3] tanımladığı dizi geçiş olasılıkları, şimdiye kadar hiç uygulanmadığı bir alana, biyoloji alanına uygulandı. İki önemli peptit sınıflandırma verisi üzerinde Oommen-Kashyap geçiş olasılıkları, Needleman-Wunsch hizalanma puanları ile karşılaştırıldı. Farklı mutasyon serisi uzunlukları (10, 50, 100, 200, 250, 300, 400 ve 500 PAM) için destek vektörü makineleri ile yapılan sınıflandırmalar, geçiş olasılıklarının, hizalanma puanlarından çok daha iyi sonuç ürettiğini gösterdi. Literatürde aynı veri kümeleri için alınan sonuçlara bakıldığında ise şu manzarayla karşılaşılır: HIV veri kümesinde alınmış en iyi sonuçları Kim ve arkadaşları [41] raporlamışlardır. On farklı yöntem için raporladıkları sonuçların yalnızca bir tanesi doğruluk değeri olarak bu çalışmada elde edilen başarıya erişmiştir; geri kalan dokuz yöntem geçiş olasılıklarının başarısının gerisinde kalmaktadır. TCL veri kümesinde ise fark çok daha belirgindir. Zhao ve arkadaşlarının [38] bu veri kümesi için raporladıkları sonuçlar, bu çalışmada Needleman-Wunsch için elde edilen sonuçlardan bile kötüdür. Oommen-Kashyap geçiş olasılığı, hesabının basitliğine rağmen, peptit sınıflandırma için şu ana kadar önerilmiş farklı yaklaşımlar arasında en başarılısı olmaya adaydır.

Diğer taraftan, Oommen-Kashyap modelinin neden peptit sınıflandırmada bu kadar başarılı olduğunu da tartışmaya açmak gerekir. Bu model, gerçekte simetrik olan bir ilişki için asimetrik bir ölçüt sunmaktadır. Üstelik, eş işlevli peptitlerin her ele alınan problemde birbirinden evrildiği söylenemez. Tüm bunlar göze alındığında, Oommen-Kashyap modelinin peptit benzerliklerini biyolojik olarak anlamlı bir şekilde yorumladığını iddia etmek çok doğru olmaz. Fakat elimizde aynı şeyi daha iyi yorumlayan bir modelin henüz olmadığı da ortadadır. Bu durum bizi ister istemez faydacı yaklaşmaya ve mevcut yöntemlerin sunmak istediği bilgiyi fazlasıyla ve eksiksiz sunan Oommen-Kashyap modelini pratik uygulamalarda kullanmaya itecektir.

Yapılan deneyler ve alınan sonuçlar bir kenara bırakılırsa, bu çalışma aynı zamanda biyolojik dizi analizi ve özellikle de hizalama algoritmaları üzerine şimdiye kadar yapılmış araştırmaların kısmi bir derlemesini sunmaktadır. Proteinlerin evrimsel, yapısal ve işlevsel özelliklerinden

bahsedilmiş ve Needleman-Wunsch, Smith-Waterman, birleşik Smith-Waterman, Oommen-Kashyap algoritmaları formel bir dille ayrıntılı olarak incelenmiştir. Ayrıca, bu algoritmalarda kullanılan puanlama matrisleri PAM ve BLOSUM'un nasıl oluşturulduğuna değinilmiş ve PAM matrisinden yola çıkılarak Oommen-Kashyap modeli için bir değiştirme dağılımına nasıl ulaşılabileceği örneklenmiştir.

Biyolojik dizi analizi, elbette bu çalışmada ele alındığından daha geniş ve daha önemli bir konudur. Zamanla bu öneminin artmasını beklemek gerekir. Çünkü sürekli gelişen bilgisayarlar ve artan hesap gücü, biyolojik dizi analizi gibi indirgemeci yaklaşımlara yönelişimizi günbegün hızlandırmaktadır. Bu çalışmada anlatılanlar ve alınan sonuçlar, bu yönelişi haklı çıkaran sebeplerin küçük bir bölümüdür.

KAYNAKLAR

[1] **Durbin, R.**, 1998. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press.

[2] **Wallqvist, A.**, **Fukunishi, Y.**, **Murphy, L.**, **Fadel, A. and Levy, R.**, 2000. Iterative sequence/secondary structure search for protein homologs: comparison with amino acid sequence alignments and application to fold recognition in genome databases., *Bioinformatics (Oxford, England)*, **16(11)**, 988.

[3] **Oommen, B. and Kashyap, R.**, 1998. A formal theory for optimal and information theoretic syntactic pattern recognition, *Pattern Recognition*, **31(8)**, 1159–1177.

[4] **Sloan-Lancaster, J. and Allen, P.**, 1996. Altered Peptide Ligand-Induced Partial T Cell Activation: Molecular Mechanisms and Role in T Cell Biology, *Annual Reviews in Immunology*, **14(1)**, 1–27.

[5] **Selivanova, G.**, **Iotsova, V.**, **Okan, I.**, **Fritsche, M.**, **Stroem, M.**, **Groner, B.**, **Grafstroem, R. and Wiman, K.**, 1997. Restoration of the growth suppression function of mutant p53 by a synthetic peptide derived from the p53 C-terminal domain, *Nature Medicine*, **3**, 632–638.

[6] **Gozes, I.**, **Perl, O.**, **Giladi, E.**, **Davidson, A.**, **Ashur-Fabian, O.**, **Rubinraut, S. and Fridkin, M.**, 1999. Mapping the active site in vasoactive intestinal peptide to a core of four amino acids: neuroprotective drug design., *Proc Natl Acad Sci US A*, **96(7)**, 4143–8.

[7] **Sigurdsson, E.**, **Scholtzova, H.**, **Mehta, P.**, **Frangione, B.**, **Wisniewski, T.**, **Survival, C.**, **Cortex-pathology, C.**, **Assay, E.**, **Mice, T. and Fragments-immunology, P.**, 2001. Immunization with a nontoxic/nonfibrillar amyloid-beta homologous peptide reduces Alzheimer's disease-associated pathology in transgenic mice., *Am J Pathol*, **159(2)**, 439–447.

[8] **Sarikaya, M.**, **Tamerler, C.**, **Jen, A.**, **Schulten, K. and Baneyx, F.**, 2003. Molecular biomimetics: nanotechnology through biology., *Nat Mater*, **2(9)**, 577–85.

[9] **Liao, L. and Noble, W.S.**, 2003. Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships, *Journal of Computational Biology*, **10(6)**, 857–868.

[10] **Mount, D.**, 2004. Bioinformatics: Sequence and Genome Analysis, CSHL Press.

[11] **Kabsch, W. and Sander, C.**, 1983. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features., *Biopolymers*, **22(12)**, 2577–637.

[12] **Murzin, A.**, **Brenner, S.**, **Hubbard, T. and Chothia, C.**, 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures, *Journal of Molecular Biology*, **247(4)**, 536–540.

[13] **Orengo, C.**, **Michie, A.**, **Jones, S.**, **Jones, D.**, **Swindells, M. and Thornton, J.**, 1997. CATH–a hierarchic classification of protein domain structures, *Structure*, **5(8)**, 1093–1108.

[14] **Holm, L. and Sander, C.**, 1994. The FSSP database of structurally aligned protein fold families, *NUCLEIC ACIDS RESEARCH*, **22**, 3600–3600.

[15] **Ashburner, M.**, **Ball, C.**, **Blake, J.**, **Botstein, D.**, **Butler, H.**, **Cherry, J.**, **Davis, A.**, **Dolinski, K.**, **Dwight, S.**, **Eppig, J.** *et al.*, 2000. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium., *Nat Genet*, **25(1)**, 25–9.

[16] **Saigo, H.**, **Vert, J.**, **Ueda, N. and Akutsu, T.**, 2004. Protein homology detection using string alignment kernels, *BIOINFORMATICS*, **20(11)**, 1682–1689.

[17] **Levenshtein, V.**, 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals, Soviet Physics Doklady, volume 10, p. 707.

[18] **Gotoh, O.**, 1982. An Improved Algorithm for Matching Biological Sequences, *J. Mol. Biol*, **162**, 705–708.

[19] **Dayhoff, M.**, **Schwartz, R. and Orcutt, B.**, 1978. A model of evolutionary change in proteins, *Atlas of Protein Sequence and Structure*, **5(Suppl 3)**, 345–352.

[20] **Henikoff, S. and Henikoff, J.**, 1992. Amino Acid Substitution Matrices from Protein Blocks, *Proceedings of the National Academy of Sciences*, **89(22)**, 10915–10919.

[21] **Pearson, W.R.**, 1995. Comparison of methods for searching protein sequence databases, *Protein Science*, **4(6)**, 1145.

[22] **Pascarella, S. and Argos, P.**, 1992. A data bank merging related protein structures and sequences, *Protein Engineering Design and Selection*, **5(2)**, 121–137.

[23] **Hough, D.**, 1981. Applications of the Proposed IEEE 754 Standard for Floating-Point Arithetic, *Computer*, **14(3)**, 70–74.

[24] **Barrell, D., Dimmer, E., Huntley, R., Binns, D., O'Donovan, C. and Apweiler, R.**, 2008. The GOA database in 2009–an integrated Gene Ontology Annotation resource, *Nucleic Acids Research*.

[25] **Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I. and Bourne, P.** The protein data bank, *logo*, **58(1 Part 6)**, 899–907.

[26] **Jones, D.**, 1999. Protein secondary structure prediction based on position-specific scoring matrices, *Journal of Molecular Biology*, **292(2)**, 195–202.

[27] **Apweiler, R., Bairoch, A., Wu, C., Barker, W., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M.** *et al.*, 2004. UniProt: the Universal Protein knowledgebase., *Nucleic Acids Research*, **32**, D115.

[28] **Kersey, P., Duarte, J., Williams, A., Karavidopoulou, Y., Birney, E. and Apweiler, R.**, 2004. Technical Brief The International Protein Index: An integrated database for proteomics experiments, *Proteomics*, **4**, 1985–1988.

[29] **Rost, B.**, 1999. Twilight zone of protein sequence alignments, *Protein Engineering*, **12(2)**, 85–94.

[30] **Li, W. and Godzik, A.**, 2006. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, *Bioinformatics*, **22(13)**, 1658–1659.

[31] **Dondoshansky, I.**, 2002. Blastclust (NCBI Software Development Toolkit), *NCBI, Bethesda, MD*.

[32] **Hou, Y., Hsu, W., Lee, M. and Bystroff, C.**, 2003. Efficient remote homology detection using local structure., *Bioinformatics*, **19(17)**, 2294–301.

[33] **Rangwala, H. and Karypis, G.**, 2005. Profile-based direct kernels for remote homology detection and fold recognition, *Bioinformatics*, **21(23)**, 4239–4247.

[34] **Hanley, J. and McNeil, B.**, 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve., *Radiology*, **143(1)**, 29–36.

[35] **Guide, M.R.**, 1998. The MathWorks, *Inc., Natick, MA*.

[36] **Duin, R.P.W., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D. and Tax, D.M.J.**, 2004. PRTools, a Matlab Toolbox for Pattern Recognition, *Delft University of Technology*.

[37] **Cai, Y.D. and Chou, K.C.**, 1998. Artificial neural network model for predicting HIV protease cleavage sites in protein, *Advances in Engineering Software*, **29(2)**, 119–128.

[38] **Zhao, Y.**, **Pinilla, C.**, **Valmori, D.**, **Martin, R. and Simon, R.**, 2003. Application of support vector machines for T-cell epitopes prediction., *Bioinformatics*, **19(15)**, 1978–84.

[39] **Thomson, R.**, **Hodgman, T.C.**, **Yang, Z.R. and Doyle, A.K.**, 2003. Characterizing proteolytic cleavage site activity using bio-basis function neural networks, *Bioinformatics*, **19(14)**, 1741–1747.

[40] **Trudgian, D.C. and Yang, Z.R.**, 2007. Substitution Matrix Optimisation for Peptide Classification, *Lecture Notes in Computer Science*, **4447**, 291.

[41] **Kim, H.**, **Zhang, Y.**, **Heo, Y.S.**, **Oh, H.B. and Chen, S.S.**, 2008. Specificity rule discovery in HIV-1 protease cleavage site analysis, *Computational Biology and Chemistry*, **32(1)**, 71–78.

[42] **Parker, K.C.**, **Bednarek, M.A. and Coligan, J.E.**, 1994. Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains., *J Immunol*, **152(1)**, 163–75.

[43] **Rammensee, H.G.**, **Friede, T. and Stevanoviic, S.**, 1995. MHC ligands and peptide motifs: first listing., *Immunogenetics*, **41(4)**, 178–228.

[44] **Mamitsuka, H.**, 1998. Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models., *Proteins*, **33(4)**, 460–74.

[45] **Anderson, R.**, **Degano, P.**, **Godkin, A.**, **Jewell, D. and Hill, A.**, 2000. In vivo antigen challenge in celiac disease identifies a single transglutaminase-modified peptide as the dominant A-gliadin T-cell epitope, *Nature Medicine*, **6**, 337–342.

ÖZGEÇMİŞ

Ad Soyad: Eser Aygün

Doğum Yeri ve Tarihi: İzmir, 11 Ağustos 1983

Lisans Üniversite: İstanbul Teknik Üniversitesi

Yayın Listesi:

Herdağdelen, A.; Aygün, E. & Bingol, H. Measuring preferential attachment, Europhysics Letters, EDP Sciences, 2007, 78, 60007

Aygün, E.; Komurlu, C.; Aydin, Z. & Cataltepe, Z. Protein Function Prediction with Amino Acid Sequence and Secondary Structure Alignment Scores, International Symposium on Health Informatics and Bioinformatics, 2008

Filiz, A.; Aygün, E.; Keskin, O. & Cataltepe, Z. Importance of Secondary Structure Elements for Prediction of GO Annotations, International Symposium on Health Informatics and Bioinformatics, 2008

Aygün E. & Cataltepe Z. Gene Ontology (GO) Molecular Function Prediction Based on Alignment Scores, International Symposium on Health Informatics and Bioinformatics, 2007

Cataltepe, Z. & Aygün, E. An Improvement of Centroid-Based Classification Algorithm for Text Classification, Data Engineering Workshop, 2007 IEEE 23rd International Conference on, 2007, 952-956

Herdagdelen, A.; Aygün, E. & Bingol, H. Measuring generalized preferential attachment in an online hyper-textual dictionary: Eksi Sozluk, ECCS European Conference on Complex Systems, Paris, France, November, 2005