BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİKSEL SEMBOLLERİN TANINMASINA YÖNELİK YENİ BİR ALGORİTMA

CEYHUN ÇAKAR

YÜKSEK LİSANS TEZİ

2015

MATEMATİKSEL SEMBOLLERİN TANINMASINA YÖNELİK YENİ BİR ALGORİTMA

A NEW ALGORITHM FOR RECOGNITION OF MATHEMATICAL SYMBOLS

CEYHUN ÇAKAR

Başkent Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2015

“MATEMATİKSEL SEMBOLLERİN TANINMASINA YÖNELİK YENİ BİR ALGORİTMA” başlıklı bu çalışma, jürimiz tarafından, /01/2015 tarihinde, **ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan	Doç. Dr. Hasan OĞUL
Üye (Danışman)	Doç. Dr. Hamit ERDEM
Üye	Yrd. Doç. Dr. Derya YILMAZ

ONAY
/01/2015

Prof. Dr. Emin AKATA
Fen Bilimleri Enstitüsü Müdürü

TEŞEKKÜR

Yazar, bu çalışmanın gerçekleşmesinde katkılarından dolayı, Sayın Doç. Dr. Hamit Erdem’e (tez danışmanı), çalışmanın sonuca ulaştırılmasında, her zaman yardımcı ve yol gösterici olduğu için, Sayın Adnan Cihan Çakar'a ve Sayın Eren Çakar'a yardımcı olduğu için, teşekkür ederim.

ÖZ

MATEMATİKSEL SEMBOLLERİN TANINMASINA YÖNELİK YENİ BİR ALGORİTMA

Ceyhun Çakar
Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Matematiksel İfadelerin Tanıma (MİT), matematiksel ifadelerin bilimsel yazındaki yaygınlığı nedeniyle önemli bir gerekliliktir. Standart Yazı Tanımanın (SYT) aksine MİT'de simgeler yatay olarak sıralanmazlar ve yakın büyüklükte olmayabilirler. Sonuç olarak, matematiksel ifadelerin tanınması standart yazıya göre çok daha zor olabilir. Bu nedenle günümüzde MİT üzerine yapılan akademik çalışmalar etkin olarak devam etmektedir. Bu tez çalışması, Simge Ayrıştırma, Simge Tanıma ve Yapısal Çözümleme algoritmalarından oluşan İstatiksel Örüntü Tanıma temelli çevrimdışı bir MİT sistemi önermektedir. Ayrıca, Simge Tanıma aşamasının doğruluğunu ve hızını arttırmak için tezde Yetim-Piksel-Oranı/Yerel-Yetim-Piksel-Oranı (YPO/YYPO) olarak isimlendirilen yeni bir istatistiksel nitelik ailesi önermektedir. YPO/YYPO nitelikleri simgeyi oluşturan siyah piksellerin, beyaz pikselleri ne şekilde çevrelediklerine göre tanımlanırlar. Bu tezde YPO/YYPO nitelikleri kullanılarak oluşturulmuş bir nitelik vektörünün, sistemin yabancı simgeleri tanıma başarısını ve tanıma hızını önemli ölçüde artırabileceği diğer yaygın nitelik vektörleri ile karşılaştırılarak incelenecektir. Bu amaçla, YPO/YYPO nitelik vektörü üç farklı sınıflandırma yöntemi (Kstar, MLP, KNN) ile sınıflandırılıp elde edilen sonuçlar tanıma hızı ve doğruluğu açısından diğer iki yaygın nitelik vektörü (3 x 3 Bitmap, dalgacık) ile karşılaştırılacaktır. Önerilen çevrimdışı MİT sistemi Java tabanlı Weka yazılım paketi kullanılarak gerçekleştirilmiştir.

ANAHTAR SÖZCÜKLER: matematiksel ifade tanıma, istatistiksel örüntü tanıma, nitelik vektörü, sınıflandırma, çok katmanlı algılayıcı, yetim piksel oranı

Danışman: Doç. Dr. Hamit ERDEM, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü.

ABSTRACT

A NEW ALGORITHM FOR RECOGNITION OF MATHEMATİCAL SYMBOLS

Ceyhun Çakar
Başkent University Institute of Science and Engeenering
Electrical and Electronics Engeneering Department

Mathematical Expression Recognition (MER) is an important requirement in science because of the prevalence of the mathematical expressions in the science literature. The symbols are not lined up horizantally and their size may not be similar in MER in contrast with Standart Text Recognition (STR). Thus, recogniton of the mathematical expression can be very difficult in comparison with STR. So, academical studies on MER are goes on effectively today. In this thesis, a software system; which is composed of Symbol Segmentation, Symbol Recognition and Spatial Analysis steps; are proposed. Also, A new statistical feature family called Orphan-Pixel-Rate/Local-Orphan-Pixel-Rate (OPR/LOPR) are introduced. OPR/LOPR features are defined by how black pixels of the binary image encloses its white pixels. In this thesis, it will be shown that a feature vector mainly created by using OPR/LOPR can increase significantly the accuracy of recogniton. For that purpose, OPR/LOPR vector will be compared with two other common feature vectors.(3 x 3 Bitmap, wavelet) according to their results of recognition speed and accuracy after its classification with three different method (Kstar, MLP, KNN). This offline MER sistem have been implemented using Weka software packet based on Java.

KEYWORDS: mathematical expession recognition, statistical pattern recognition, feature vector, classification, mult-layer perceptron, orphan pixel rate

Advisor: Associated Prof. Hamit ERDEM, Başkent Üniversity, Electrical and Electronics Department.

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ SAYFASI

MİT Matematiksel İfade Tanıma

SYT Standart Yazı Tanıma

İÖT İstatistiksel Örüntü Tanıma

YPO Yetim Piksel Oranı

YYPO Yerel Yetim Piksel Oranı

MLP Çok Katmanlı Algılayıcı (Multi Layer Perceptron)

KNN K-Yakın Komşu (K-Nearest Neighbor)

YSA Yapay Sinir Ağları

HMM Gizli Markov Modelleri (Hidden Markov Models)

NV Nitelik Vektörü

1 GİRİŞ

Matematiksel ifadeler bilimsel yazım içinde yaygın olarak kullanılır. Bu ifadeleri oluşturan simgeler standart yazıların aksine yatay bir hizalama çizgisi üzerinde sıralanmazlar ve genellikle büyüklükleri birbirlerine yakın değildir. Ayrıca, simgelerin geometrik diziliş ve konumları, ifade bütününün anlamını etkilediği için çözümlenmelidir. Sonuç olarak, Matematiksel İfade Tanıma, Standart Yazı Tanıma ile aynı temel yöntemlere dayanmakla birlikte, kendine özel zorluklar içerir ve bu zorluklara özel ek yöntemler gerektirir.

SYT ve MİT üzerine 40 yıla yakın zamandır araştırma yapılıyor [1]. Bu çalışmalar 90'lı yıllarda hız kazandı. 1985 yılında ilk sürümü çıkan Tessaract [2] [3] 1994 yılına kadar Hewlett Packard tarafından lisanslı olarak geliştirildi. Bugün Google sponsorluğunda ve açık kaynaklı olarak ulaşılabilir. Bununla beraber, Tessaract MİT için özelleştirilmiş bir program değil. 1999 yılında, FFES ismiyle MİT için özelleştirilmiş bir çerçeve-kod Steve Smithier tarafından geliştirildi. Bu kod FFES/DRACULAE [4] adıyla geliştirilmeye devam ediyor ve kendi sitesinden ücretsiz indirilebiliyor. Suzuki tarafından 2000'li yıllarda geliştirilen Infty Reader [5] görme engelli veya görme bozukluğu olan kişileri hedefleyen ticari bir yazılım. FFES/DRACULE ile karşılaştırıldığında daha sofistike ama kodları açık olmadığından üzerinde akademik çalışma yapmak daha zor. 2007'de Fink ve Plötz tarafından başlatılan ESMERALDA [6][7] projesi, temelde karakter tanıma amaçlı olarak ortaya çıkmamasına rağmen, yapı bir ek modülle yazı tanımaya uyarlandı. ESMERALDA, yazı tanımayı istatistiksel sınıflandırmayla değil, Gizli Markov Modelleri (HMM) [6] kullanarak yapıyor.

HMM örüntüyü doğası gereği tek boyutlu bir girdiye ihtiyaç duyar. MIT (ya da daha genel anlamıyla karmaşık simgelerin tanınması) iki boyutlu girdi olduğu için ham imajı tek boyutlu bir örüntüye çeviren bir süzgeçten sonra HMM uygulanır. Oysa HMM'nin yaygın olarak kullanıldığı ses tanımadaki temel avantajı zaten tek boyutlu bir sayı dizisi olan sesin ek bir işleme tabi tutulmasına gerek bırakmamasıdır. Bu yüzden MIT için yaygın olarak kullanılmamaktadır. Yukarıda bahsedilen diğer teknikler ise İÖT kullanırlar. İÖT sisteminde yine iki boyutlu imaj direk olarak sınıflandırılamaz. O imajdan çıkarılan nitelikler sınıflandırılır. Bu nitelik çıkarma aşaması hızlı olmalı ve simgenin kategorisini belirleyen özellikleri

içermeli (kategorik özellikleri) ama diğer özellikleri (ki bu önemsiz özellikleri ayırmak zordur) olabildiğince az içermelidir (örneğin yazma alışkanlığından gelen özellikler). Simgenin kategorik özellikleri nitelik tarafından ne kadar iyi temsil edilirse simgenin tanıma doğruluğu o kadar artacaktır. Bununla beraber, temelde birbiriyle çelişen bu iki özelliği taşıyan bir nitelik bulmak oldukça zordur. Bu tezde oldukça hızlı çıkarılabilen ayrıca simgenin kategorik özelliklerini yoğun olarak taşıyan YPO/YYPO nitelikleri tanıtılarak bu nitelikler üzerine yapılandırılmış bir algoritma oluşturulacaktır. Böyle bir algoritma sistemin hem hızını hem doğruluğunu birlikte artırabilecektir. Bu tezde, YPO/YYPO'ya dayanan böyle bir çevrimdışı bir MİT sistemi önerilmiştir. Bu sistem olarak üç ayrı algoritmadan oluşmaktadır [8]:

- Simge Ayrıştırma
- Simge Tanıma
- Yapısal Çözümleme

Simge Ayrıştırma, işlenmek üzere bütün olarak alınan kod girdisinin onu oluşturan simgelere ayrıştırılmasıdır. Çevrim-dışı bir MİT sisteminde kod girdisi, genellikle matematiksel ifadeler içeren bir imajdır. İmajı oluşturan simgeler ise rakam, noktalama işareti ve benzeri matematiksel olarak anlamlı örüntüler olarak tanımlanabilir.

Simge Tanıma, ayrıştırma aşamasında elde edilen simgelerin her birinin, tanımlı kategorilerle eşleştirilmesidir. Bu aşamada, her bir simgeden istatiksel nitelikler (simgeden matematiksel bir fonksiyonun ürettiği skaler değerler) çıkarılır ve bu nitelikler bir sınıflandırma veya kümeleme yöntemiyle seçilen bir kategori kümesinin elemanlarıyla eşleştirilir.

Yapısal Çözümleme; tanınan simgelerin, birbirleri arasındaki konum ve büyüklük ilişkilerin çözümlenmesidir. Standart yazı tanımanın aksine; MİT'te Yapısal Çözümleme, Simge Tanıma kadar önemli ve zordur. Bu aşamada ayrışmış ve tanınmış simgeler, bilgisayarca anlamlı kodlar olarak yeniden birleştirilirler. MİT'in bu aşamadaki çıktısı TEX gibi herhangi bir matematiksel işaretleme dili olabilir. Bu tezde işlenen yapısal çözümleme algoritması Garain'in [9] makalesinde tanıtılan algoritmanın benzeridir. Her iki algoritmada imajın dik ve yatay olarak

yinelgen dilimlenmesine dayanıyor. Ancak, Garain algoritmasında dilimleme gerçek imaj üzerinde yapılırken, bu tezde dilimleme Simge Ayrıştırma'nın çıkış verilerden yaratılan bir ızgara üzerinde yapılıyor. Bu yaklaşım, ayrıca tartışılmıştır.

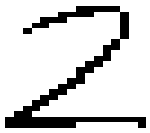
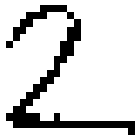
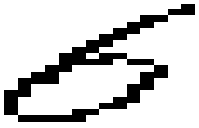
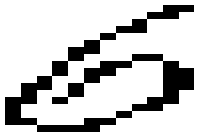
Tezde Simge Tanıma algoritması sırasında matematiksel ifadelerin tanınması için farklı bir nitelik vektörü kullanılmıştır. Örneğin, tanıma sisteminin bir simgeyi '2' veya '6' kategorilerinden biriyle eşleştirilmesi gerekiyor olsun. Bunu yapmanın oldukça iyi bir yolu simgenin yalıtılmış bir beyaz piksel alanı içerip içermediğine bakmaktır. Fakat, bu yöntemle Şekil 1d, '6' değil '2' olarak algılanır ki '6' simgesinin bu şekilde yazılması oldukça yaygındır. Yani, yalıtılmış bir beyaz piksel alanının var olup olmamasını kategoriler arası ayırt edici olarak kullanmak yazarın yazım biçimine oldukça hassastır. Oysa, simgenin yalnızca soldan, sağdan, üstten ve alttan siyah piksellerle çerçevelenip çerçevelenmediğine bakılırsa, '2' ve '6' kategorilerini ayırt etmek için yazım biçiminden bağımsız bir nitelik elde edilmiş olur. Sol, sağ, alt ve üst çerçevelenme durumlarının diğer kombinasyonları kullanılarak benzer şekilde daha fazla nitelik tanımlanabilir. Tezde bu nitelik ailesi Yetim-Piksel-Oranı (YPO) nitelikleri olarak isimlendirilmiştir. YPO nitelik ailesinin imajın tamamı yerine belli bir bölgesi için çıkarılması ile elde edilen aile ise Yerel-Yetim-Piksel-Oranı(YYPO) isimlendirilmiştir. Deneyler Bölümünde, bu ailenin tanıma doğruluğunu önemli şekilde artırdığı gözlemlenebilir.

MİT'in tek bir teze sığmayacak kadar geniş bir konu olmasından dolayı; tez boyunca girdi resminin aşağıda belirtilen sınırlamalara uyduğunu varsayacağım:

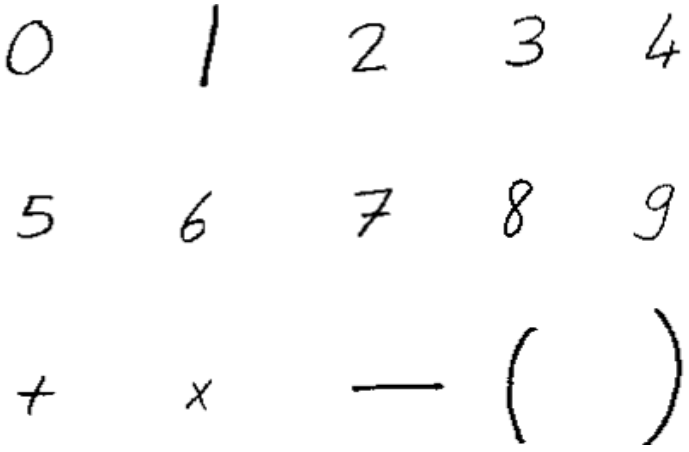
1. Girdi resmi gri-seviye imajı olarak kaydedilmiştir ve yeterince temizdir.
2. Girdi resmi; yalnızca 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -, x, (,) simgelerini içerir.
3. Farklı simgeler, birbirine temas etmezler.
4. Simgelerin hiç birinde kopma yoktur.
5. İmaj sadece matematiksel ifadeler içerir. (Matematiksel ve matematiksel olmayan ifadelerin karması değildir.)
6. İmaj yalnızca bir matematiksel ifade grubu (tek bir formül gibi) içerir.
7. Aynı simge için, genel kabul görmüş yalnızca bir versiyon kullanılmıştır. Tezde varsayılan versiyonlar Şekil 2'de gösterilmiştir.

Ayrıca, önerilen sistem Java (JDK 1.7) ile yazılıp derlenmiş; Windows 8.1 İşletim Sistemi üzerinde Eclipse Java EE IDE for Web Developers Luna Release 4.4.0 - Debug mod'da çalıştırılmıştır. Ayrıca bahsi geçen tüm sınıflandırma yöntemleri için Weka 3.7.11 kütüphanesi, nitelik vektörlerinden birinde kullanılan dalgacık dönüşümü için JWave 2.0 kullanılmıştır.

Deneylerde kullanılmak üzere üç farklı kişiden toplan 4535 örnek alınmıştır. Bu kişiler deney sonuçlarının verildiği çizelgelerde ceyhun (1508 örnek), eren (1507 örnek) ve cihan(1520 örnek) olarak geçmektedir. Bu örnekler, tez tarafından önerilen YPO/YYPO'da dahil olmak üzere üç nitelik vektörü (YPO/YYPO, 3x3 bitmap imaj, dalgacık dönüşümü çıktısı) ile ve bu deneyden bağımsız olarak üç sınıflandırma yöntemi ile denenmiştir (MLP, Kstar KNN).

	
(a) Simge İki – Örnek 1	(b) Simge İki – Örnek 2
	
(c) Simge Altı – Örnek 1	(d) Simge Altı – Örnek 2

Şekil 1 İki ve Altı Simgeleri Örnek Simgeleri



Şekil 2 Tezde Varsayılan Kategori Kümesi

2 İSTATİKSEL ÖRÜNTÜ TANIMA

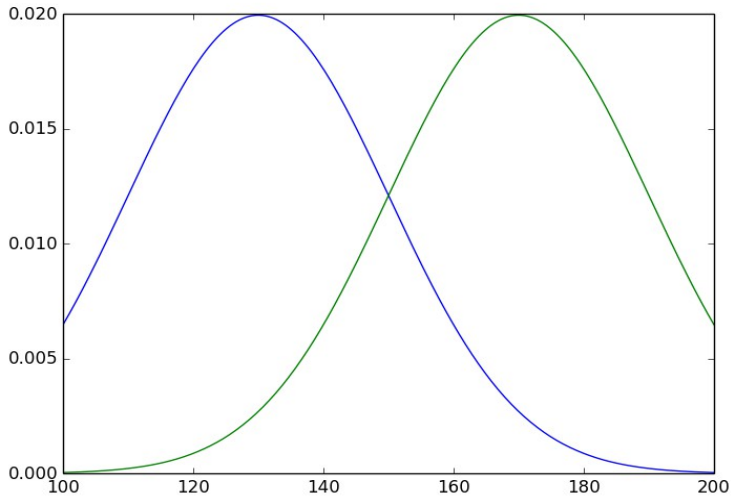
Bir nesnenin ürettiği ya da içerdiği veri tamamen rastgele değil istatiksel bir kurala uyuyorsa, bu nesnenin bir örüntüye sahip olduğu söylenebilir. Söz konusu nesne bir imaj, ses ya da insan DNA'sı olabilir. Sonuç olarak bir nesnenin, örüntüsüne göre bir kategori kümesinin elemanlarıyla eşleştirilmesini örüntü tanıma olarak tanımlayabiliriz. [10].

Yukarıda tanımlandığı şekliyle örüntü istatistiksel bir kavramdır. Doğal olarak, örüntü tanıma'ya tamamen istatiksel olarak yaklaşılabilir. Bugün oldukça yaygın olan bu yaklaşım İstatistiksel Örüntü Tanıma (İÖT) olarak adlandırılmaktadır.

İÖT'de tanınacak nesnenin belirli sayıda istatiksel niteliği alınır. Bu nitelikler birleştirilerek bir nitelik vektörü oluşturulur [11]. Son olarak bu nitelik vektörü seçilen bir sınıflandırma yöntemiyle sınıflandırılır.

2.1 Bayesian Sınıflandırıcı

İÖT'de kullanılan en basit ve en eski sınıflandırma yöntemi sınıflandırıcısıdır.



Şekil 3 Bayesian sınıflandırıcı için örnek normalize edilmiş çocuk ve yetişkin dağılımı

Örnek olarak, sınıflandırıcının görevi boyu bilinen kişileri çocuk veya yetişkin olarak gruplamak olsun. Çocuk ve yetişkinler için boy dağılım grafiğini (1) ve (2)'deki gibi tanımlanmış olsun.

$$PDF_{çocuk} = Gaussian[\mu = 130, \sigma = 20] \quad (1)$$

$$PDF_{yetişkin} = Gaussian[\mu = 170, \sigma = 20] \quad (2)$$

İki dağılım grafiği Şekil 3'deki gibi 150 cm'de kesişir. Bayesian sınıflandırıcı bu kesişim noktasını kategori sınırı olarak kabul eder. 150 cm'den kısa tüm bireyler sınıflandırıcı tarafından çocuk, daha uzun olanlarsa yetişkin olarak atanırlar. Tabi ki bireye ait tek bir nitelik çoğu zaman yeterli değildir. Birden fazla nitelik içinde de bir Bayesian sınıflandırıcı tanımlanabilir. Bu durumda, ayırt edici olarak skaler bir değer değil nitelik sayısına göre bir doğru, yüzey ya da hiper-yüzey kullanılır. En genel haliyle ayırt edici olarak niteliklerden kategori kümesinin elemanlarına götüren her hangi bir fonksiyon kullanılabilir. Bu fonksiyona kernel fonksiyonu denir.

Bayesian sınıflandırıcısının temel problemi pratik uygulamalarda nadiren eldeki kategoriler için bir dağılım fonksiyonu bulunabilir. Bu yüzden, burada anlatıldığı şekliyle Bayesian sınıflandırıcı nadiren kullanılabilir. Yine de bu yöntem İÖT'de pratikte kullanılan yöntemler için sağlam bir temel oluşturmaktadır.

2.2 Makine Öğrenmesi

Bayesian Sınıflandırıcı'nın temel varsayımı nesnenin niteliğiyle (veya nitelikleriyle) ait olduğu kategori arasındaki korrelasyonun tam olarak bilindiğidir. Bu, pratikte nadiren geçerlidir. Bu nedenle sınıflandırıcının pratikte bu korrelasyonu ya da tahminde bulunmasına yardım edecek sisteme ait her hangi bir bilgiyi öğrenmesi gerekir. Bu amaçla İÖT sistemi gözetimli, gözetimsiz, veya zorlamalı öğrenme yöntemlerinden birini seçebilir.

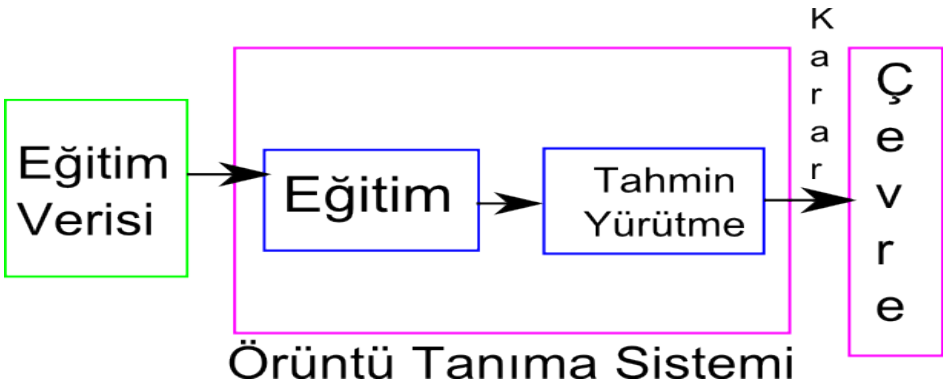
2.3 Gözetimli Öğrenme

Doğru olduğu bilinen bir veri kümesinin sınıflandırıcıyı eğitmek için kullanılmasıdır. Bu yapıda eğitim verileri, test verilerinden ayrıdır ve sistemin eğitimi, sistem kullanılmaya başlamadan önce yapılır (Şekil 4). Bayesian sınıflandırmada elimizde dağılım fonksiyonlarının olduğu varsayılmıştı. Oysa elde Çizelge 1'deki gibi kategorileri (çocuk veya yetişkin) ve boylarını bildiğimiz bir kişi listesi olması çok daha olasıdır. Bu teknikte sınıflandırıcı eldeki bu listeyle eğitilir.

Eğitim sonucunda listede olmayan ve yalnız boyu bilinen birinin çocuk mu büyük mü olduğuna dair bir tahmin yapılabilir.

Çizelge 1 Örnek Eğitim Verisi

Nitelik	Kategori		Nitelik	Kategori
114 cm	Çocuk		201 cm	Yetişkin
151 cm	Çocuk		155 cm	Yetişkin
129 cm	Çocuk		162 cm	Yetişkin
130 cm	Çocuk		128 cm	Çocuk
176 cm	Yetişkin		106 cm	Çocuk
153 cm	Çocuk		131 cm	Çocuk
175 cm	Yetişkin		109 cm	Çocuk
167 cm	Yetişkin		180 cm	Yetişkin
130 cm	Çocuk		193 cm	Yetişkin
186 cm	Yetişkin		147 cm	Yetişkin
172 cm	Yetişkin		158 cm	Çocuk
145 cm	Çocuk		186 cm	Yetişkin

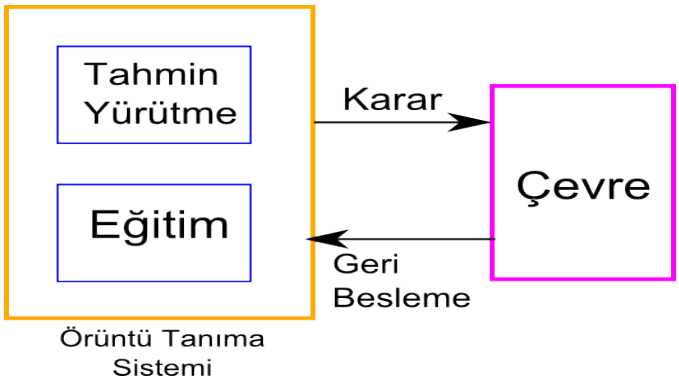


Şekil 4 Gözetimli Eğitim

2.4 Pekiştirmeli Eğitim

Gözetimli Eğitim'de eğitim işlemi, tahmin yürütme işleminden ayrı olarak ve daha önce yapılır (Şekil 4). Sistem ilk tahmin yürütmesini yaptığında tamamen eğitilmiştir. Bu ancak elimizde doğru olduğu bilinen bir eğitim verisi varsa mümkündür. Eğer böyle bir veri yoksa ama tahminlerinin doğruluğuna dair bir geri bildirim mevcutsa, sistem kendi tahminlerini eğitim verisi olarak kullanabilir (Şekil

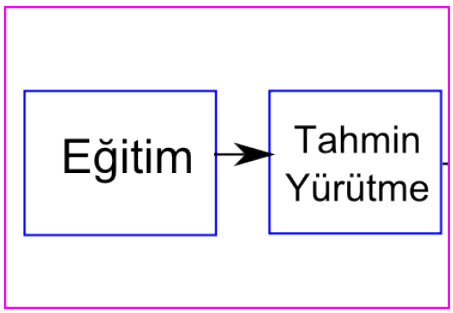
5). Başlangıçta tahminleri çoğu zaman başarısız olmakla birlikte zamanla tahmin başarısı artacaktır.



Şekil 5 Pekiştirmeli Eğitim

2.5 Gözetimsiz Eğitim

Makine eğitimi için en kötü durum ne bir eğitim verisine ne de çevreden gelen tahminlerin başarısına dair bir geri beslemeye sahip olmaktır (Şekil 6). Yine de eğer tahmin yürütülecek olan nesne niteliklerinin benzerliklerini temsil eden bir ölçek tanımlanmışsa sistem eğitilebilir. Sisteme eğitim verisi veya geri besleme yoluyla sağlandığı için, gözetimsiz eğitim sistemleri önce den tanımlanmış bir kategori kümesine sahip değildir. Bunun yerine gözetimsiz eğitim sistemi eğitim ve tahmin yürütme sırasında kendi oluşturduğu kümelere nesneleri atar.



Şekil 6 Gözetimsiz Eğitim

2.6 Naive-Bayes Sınıflandırıcı [12]

Bayes Sınıflandırıcı gözetimli eğitim tekniğiyle yeniden tasarlanabilir. Yani elimizde olasılık dağılım fonksiyonları olmasa da eğitim verisi olarak kullanabileceğimiz bir eğitim verisi olabilir. Bayes Sınıflandırıcının bu şekilde değiştirilmesi ile Naive-Bayes Sınıflandırıcı elde edilir.

Bu sınıflandırıcı geçmiş eğitim verileri ışığında en olası (V_{nb}) kategori tahminini (3)'e göre yapar [12].

$$V_{nb}=argmax V_{JEV} P_{(v_j)} \prod P(a_j/V_j) \quad (3)$$

Örnek olarak, sınıflandırıcı Çizelge 2'deki verilere göre kırmızı, yerli bir SUV arabayı çalıntı veya değil olarak sınıflandırması isteniyor olsun.

Çizelge 2 Naive-Bayes için Örnek Girdi Meisner'den [12, s. 1] alınmıştır.

Örnek No	Renk	Tip	Köken	Kategori (Çalıntı mı?)
1	Kırmızı	Spor	Yerli	Evet
2	Kırmızı	Spor	Yerli	Hayır
3	Kırmızı	Spor	Yerli	Evet
4	Sarı	Spor	Yerli	Hayır
5	Sarı	Spor	İthal	Evet
6	Sarı	SUV	İthal	Hayır
7	Sarı	SUV	İthal	Evet
8	Sarı	SUV	Yerli	Hayır
9	Kırmızı	SUV	İthal	Hayır
10	Kırmızı	Spor	Yerli	Evet

$$P(a_i \mid V_i)=\frac{n_c+mp}{n+m} \quad \text{[12, s.1]} \quad (4)$$

$$n: V=V_j \; olan \; eğitim \; verisi \; sayısı$$

$$n_c: V=V_j \; ve \; a=a_j \; olan \; eğitim \; verisi \; sayısı$$

$$p: eski \; olasılık \; tahmini \; (0{,}5 \; ile \; ilklendirilir)$$

$$m: sabit \; (keyfi \; olarak \; belirlenir)$$

Bunun için, eğitim verileri kullanılarak P(Kırmızı | Çalıntı), P(SUV | Çalıntı), P(Yerli | Çalıntı), P(Kırmızı | Değil), P(SUV | Değil) ve P(Yerli | Değil) olasılıkları (4) kullanılarak hesaplanır. Sonuç olarak (5), (6), (7) ve (8)'de görülen sonuçlar elde edilir.

$$P(Kırmızı \mid Çalıntı)=\frac{1+3\times 0{,}5}{5+3}=0{,}31 \quad (5)$$

$$P(SUV|Çalıntı)=\frac{2+3\times0,5}{5+3}=0,43 \quad (6)$$

$$P(Yerli|Çalıntı)=\frac{2+3\times0,5}{5+3}=0,43 \quad (7)$$

$$P(Kırmızı|Değil)=\frac{3+3\times0,5}{5+3}=0,56 \quad (8)$$

$$P(SUV|Değil)=\frac{3+3\times0,5}{5+3}=0,56 \quad (9)$$

$$P(Yerli|Değil)=\frac{3+3\times0,5}{5+3}=0,56 \quad (10)$$

Eğer keyfi seçilen bir arabanın çalıntı olma veya olmama olasılığı eşitse; yani $P(Çalıntı)=P(Değil)=0,5$; ise arabanın Çalıntı ve Çalıntı Değil durumu için (4)'ü kullanılırsa v=Çalıntı için 0.037 (11), v=Değil için 0.069 (12) değerleri bulunur. Bu durumda 0.069 > 0.037 olduğu için Naive-Bayes sınıflandırıcı bu arabanın çalıntı DEĞİL olarak sınıflandıracaktır.

$$P(Çal.)\times P(Kırm.|Çal.)\times P(SUV|Çal.)\times P(Yerli|Çal.)=0,037 \quad (11)$$

$$P(Değil)\times P(Kırm.|Değil)\times P(SUV|Değil)\times P(Yerli|Değil)=0,069 \quad (12)$$

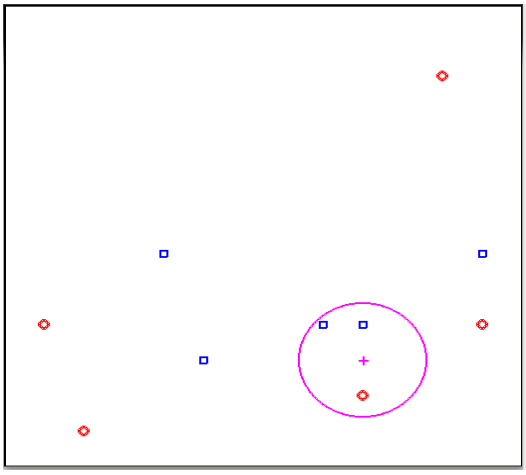
2.7 K-Yakın-Komşu (KNN) Algoritması [13]

Sınıflandırma algoritmaları içinde, ikinci önemli grup öğe-temelli algoritmalardır. Kernel algoritmalarının aksine, bu algoritmalar ayırt edici bir fonksiyon yerine eldeki verileri ya da verilerin bir kısmını direk olarak kullanır.

Bu algoritma grubu içindeki en basit algoritma K-Yakın-Komşu algoritmasıdır. X, Y ekseninde tanımlanmış iki niteliği bulunan ve iki kategoriye göre sınıflandırılması istenen Şekil 7'deki sistemi düşünelim. Daire ile gösterilen noktalar 1. kategoriye ait eğitim verileri; kare ile gösterilenler ise, 2. kategoriye ait eğitim verileri olsun. Kategorisi bilinmeyen + verisini k=1 için tahmin etmek istenirse; +'e en yakın veri bulunur. Buradaki yakınlık tanımı için iki veri arasındaki benzerliği temsil eden herhangi bir norm kullanılabilir. Bu örnekte, basit olarak Öklit mesafesi kullanırsak, en yakın komşunun 1. kategoridendir (daire). Sonuç olarak KNN + noktasının kategorisini 1. kategori (daire) olarak tahmin edecektir.

Eğer k=3 olsaydı, + noktasının kategorisini tahmin etmek için bu veriye en yakın 3 eğitim verisi alınır. Figür'de görüldüğü gibi bu 3 veriden ikisi 1. kategori biri 2. kategorisindedir. Baskın olan 1. kategori olduğu için KNN + nesnesini yine daire olarak sınıflandırır. Bu işlem her tek k sayısı için uygulanabilir.

KNN oldukça basit bir yöntem olmasına karşın, hem kendisi hem de türevleri pratik uygulamalarda yaygın olarak kullanılırlar.



Şekil 7 K-Yakın-Komşu (KNN)

2.8 Kolgomorov Mesafesi ve Entropi Temelli Uzaklık Ölçümü [14]

Şu ana kadar tartışılan tüm algoritmalar, nesneleri birbiriyle karşılaştırabilmek için bir uzaklık ölçütüne ihtiyaç duyuyordu. Bu amaçla şimdiye kadar nesnelerin nitelik vektörleri arasındaki öklit mesafesi kullanıldı. Buna göre

$$f_0=\{a_{0,}a_{1,\ldots\ldots},a_n\} \tag{13}$$

$$f_1=\{b_{0,}b_{1,\ldots\ldots},b_n\} \tag{14}$$

olmak üzere iki nesneye ait nitelik vektörü ise, bu iki nesne arasındaki öklit mesafesi;

$$d(f_{0,}f_1)=\sqrt{(b_0-a_0)^2+(b_1-a_1)^2+\ldots\ldots+(b_n-a_n)^2} \tag{15}$$

olarak tanımlanır. Bu uzaklık (veya nesneler arasındaki benzemezlik) ölçütü, tek seçenek değildir. Nitelik vektörleri üzerinde tanımlanmış bir fonksiyon aşağıda verilen norm tanımına uyuyorsa, bir uzaklık ölçütü olarak kullanılabilir.

1. $p(a\vec{v}) = ap(\vec{v})$ (16)

2. $p(\vec{u}+\vec{v}) \leq p(\vec{u}) + p(\vec{v})$ (17)

3. $p(\vec{v}) \Leftrightarrow \vec{v} = \vec{0}$ (18)

Farklı uzaklık ölçütlerinin birbirlerine göre avantaj ve dezavantajları olabilir. Bu yüzden, uygulamaya özel seçilebilirler. Bir nitelik vektörünü diğerine çevirmek için gerekli adım sayısını bir norm olarak tanımlayabiliriz. Bu tanım Kolgomorov'un karmaşıklık tanımının; uzaklık ölçütüne uyarlanmış biçimidir. Kolgomorov'a göre bir nesnenin karmaşıklığı, o nesneyi üreten en kısa programın uzunluğu ile ölçülür. Aynı şekilde iki nesnenin birbirinden farkı (uzaklığı) birinci nesneyi ikinci nesneye çeviren en kısa programın yönerge sayısı ile ölçülebilir. Tabi bu uzaklık programın kullanımına izin verdiği yönergelere de bağlıdır. Yani amaca göre tanımlanmış her yönerge grubu için yeni bir uzaklık tanımı elde edilir. Örneğin satrançta iki kare arasındaki kale mesafesi veya at mesafesini kullanmak (1. kareden ikincisine at veya kale kaç hamlede gider) bize iki ayrı uzaklık ölçütü verir.

2.9 K* (Kstar) Sınıflandırma Algoritması [14]

K* önceki bölümde tanıtılan Kolgomorov mesafesini bir yönüyle değiştirir. İlk niteliği, ikinciye çevirebilen en kısa program yerine; olası tüm programların ortaya çıkma olasılıklarının toplamı uzaklık ölçütü olarak kullanılır. M uzunluğunda bir programın rastgele ortaya çıkma olasılığı 2^m dir. Bu tür sonsuz sayıda program bile olsa, bu programların olasılık toplamları 1'den küçük bir olasılık değeri olacaktır. Aslında hesapladığımız, Kolgomorov programı için tanımlı yönergelere uymak dışında tamamen kör hareket eden bir sistemin 1. niteliği, 2.'ye çevirme olasılığıdır. Bu tür bir uzaklığı ölçmek zor olmakla birlikte örneğin DNA analizinde, standart öklit uzaklığından daha iyi sonuç verdiği saptanmıştır. K* temelde bu uzaklık ölçütünün KNN için kullanılmasıdır.

2.10 Nitelik Çıkarma

İstatistiksel Örüntü Tanıma (İÖT) iki temel aşamadan oluşur: Nesnenin nitelik vektörünü oluşturma ve bu nitelik vektörünü sınıflandırma. Önceki bölümlerde

sınıflandırma aşamasını incelerken nitelik vektörünün ya hazır olduğunu ya da fiziksel bir nesneden deney yoluyla elde edildiğini yani ayrıca üretilmesine gerek olmadığı varsayıldı. Birçok uygulamada durum bunun tersidir. Sınıflandırma yapmadan önce niteliklerin ne olacağına karar verilmesi (nitelik seçme) ve nesneden üretilmesi (nitelik çıkarma) gerekmektedir. Bu aşamaların ayrıca sınıflandırma aşamasından temel bir farkı vardır: Nitelik seçme ve çıkarma uygulamaya bağımlı işlemlerdir. Bu yüzden bu bölümde her türlü nitelik çıkarma ve seçmeye değil; yalnızca tez konusu olan MİT'ye ya da onun üst-disiplini olduğu için karakter tanımayla ilgili nitelik çıkarma ve seçme yöntemlerine değinilecektir.

2.10.1 m x n bit imaj nitelik vektörü – bölgeleme [15]

Çok hızlı ve kolayca elde edilebildiği için çok yaygın kullanılan bir tekniktir. Tanınacak karakterin imajı m x n boyutlarında gri-seviye olarak yeniden boyutlandırılır ve eğer renkliyse indirgenir. Elde edilen her gri seviye değeri (toplam m x n sayıda) nitelik vektörünün elemanlarını oluşturur [15]. Tezde bu yöntem eşdağılımlı olarak kullanılmıştır. Yani, elde edilen küçültülmüş imajın her pikseli, girdi imajının aynı boyuttaki bir bölgesinden elde edilmiştir. Bununla birlikte, aynı yöntem [16]'te olduğu gibi eşdağılımlı olmayan bir şekilde de kullanılabilir. Sonuç olarak, imajın bir bölümü; örneğin merkezi; diğer bölümlerinden daha önemliyse, bu önem farkı bu şekilde sisteme yansıtılabilir.

2.10.2 İmaj inceliği veya genişliği

Tanınmak istenen imajın en temel özellikleri nitelik vektörün elemanı olarak kullanılabilir. Yeter ki imajın eşleneceği kategorilerle bir korrelasyon taşısın. Örneğin Yükseklik/Genişlik oranı çok büyük olan bir imajın '1' olma olasılığı '-' olma olasılığından çok daha yüksektir. Bu oran ayrıca tez konusu sistemde nitelik vektörünün bir elemanı olarak kullanılmıştır.

2.11 Yapay Sinir Ağları (YSA) – Çok Katmanlı Algılayıcı (MLP)

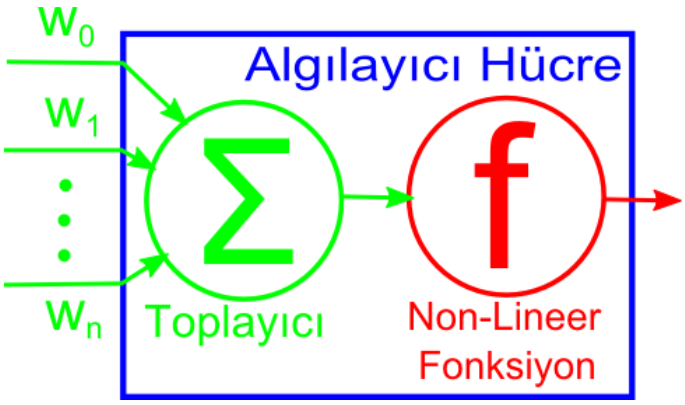
Yapay Sinir Ağları (YSA), canlıların sinir sistemini taklit eden bilgisayar sistemleridir. YSA sistemi standart bilgisayar sistemlerinden farklı olarak gelişmiş tekil bir merkezi işlemci içermez. Bunun yerine insan beyninde olduğu gibi

birbiriyle bağlı ve paralel işleyen ilkel işlemcilerin oluşturduğu bir ağdır [11]. Her bir işlemci kendi içinde basit olmasına rağmen, işlemcilerin birlikte oluşturdukları ağ, karmaşık bir bilgisayar sistemi oluşturur. Bununla beraber, YSA sistemi standart bir bilgisayarda çalışan bir program tarafından da taklit edilebilir.

$$y = \vec{w} \cdot \vec{x} + w_{bias} \tag{19}$$

$$z = f(y) \tag{20}$$

$$z = f(y) = \frac{1}{1 + y^{-1}} \tag{21}$$



Şekil 8 Yapay Sinir Ağları – Algılayıcı Yapısı

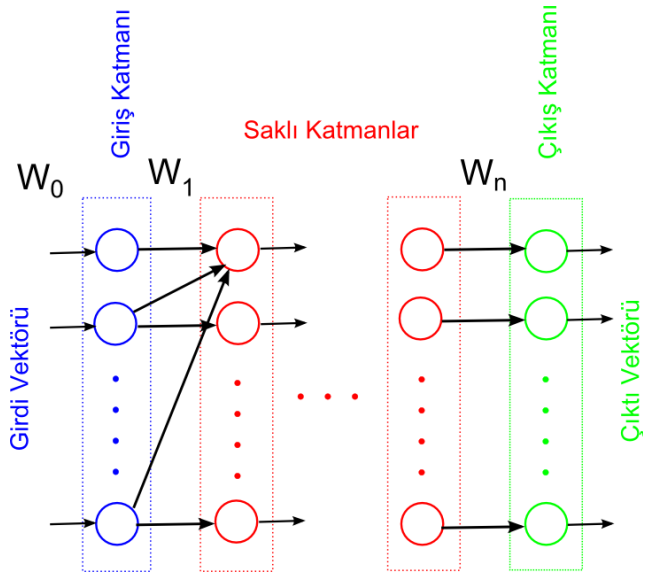
Çok Katmanlı Algılayıcı (MLP) yapısında her bir algılayıcı $\mathbb{m}$ boyutlu bir $\vec{x}$ vektörünü girdi olarak alır. Bu vektörden, bir lineer fonksiyonla (19) bir skaler üretilir. Bu skalerden, (20) doğrusal olmayan f fonksiyonu ile z skaleri üretilir (Şekil 8). Doğrusal olmayan bu fonksiyon sistemin tüm algılayıcıları için aynıdır. En yaygın kullanılan fonksiyon (21) lojistik fonksiyondur. Her biri bu şekilde tanımlanan algıyacılar Şekil 9'da görüldüğü gibi katmanlar halinde gruplanır ve sıralanırlar. Sıralamadaki ilk katmana bir girdi vektörü girilir. Her bir katman bir çıktı vektörü üretir ve bu vektör kendinden sonraki katmana bağlanır. Çıkış katmanı olarak isimlendirilen son katman sistemin çıktısını üretir. Ara katmanlar saklı-katman olarak isimlendirilir. Hiç saklı katman olmayabileceği gibi sistem istendiği kadar saklı-katman içerebilir. Böylece sistem $\vec{I}$ girdi vektöründen her katmanın ağırlık vektörlerine $(W_0, W_1, W_2 ... W_n)$ bağlı bir $\vec{O}$ çıktı vektörü üretir [17].

$$\vec{O} = MLP(\vec{I}, W_o, W_1, ..., W_n) \quad (22)$$

Böyle bir sistem tanıma amaçlı kullanılabilir. Eğer katmanların ağırlık matriksleri doğru ayarlanmışsa, MLP sistemine girdi olarak tanınacak simgeden üretilmiş nitelik vektörü girdiğimizde sistem o nitelik vektörünü bir çıktı vektörünün temsil ettiği kategori ile eşleyebilir. Bu noktada temel sorun nitelik vektörlerini, o vektörlerin kategorilerine eşleyen uygun bir ağırlık matriksi grubunun nasıl elde edileceğidir. Bu, nitelik vektörleri ve kategorileri belli olan bir veri grubu kullanarak ağırlık vektörlerinin ayarlanması yoluyla yapılabilir. Yani MLP sistemi doğru ağırlık vektörü için eğitilebilir. Bu amaçla kullanılan bir çok algoritma olmakla birlikte bunların içinde en yaygın olanı aşağıdaki geri-besleme algoritmasıdır [17]:

1. Eğitim verisi $\vec{I}$ girdisi ve bu girdinin beklenen sonucu $O_{\vec{beklenen}}$ olarak tanımlansın. Ayrıca η değeri 0 ile 1 arasında bir sabit olsun.
2. Ağırlık matriksleri keyfi $W = [W_0, W_1, ...W_n]$ olarak atanır.
3. Bu ağırlık matriksleri ile (22) kullanılarak $O_{\vec{sonuç}}$ sonuç çıktı vektörü elde edilir.
4. Çıkış katmanının her bir algılayıcısı için $\delta = O_{\vec{sonuç}}(1 - O_{\vec{sonuç}})(O_{\vec{beklenen}} - O_{\vec{sonuç}})$ hesaplanır.
5. Her bir algılayıcı için o algılayıcıya ait ağırlık vektörü $\vec{w_{yeni}} = \vec{w_{eski}} + \eta\,\delta$ formülüyle eski vektöre göre değiştirilir.
6. Katmanın ağırlık matriksi, yeni ağırlık vektörlerinden yeniden oluşturulur.
7. 3, 4 ve 5. giriş katmanına ulaşana kadar her katman için tekrarlanır.
8. Tüm katmanların ağırlık matriksi yenilendikten sonra işlem diğer eğitim veri elemanları için de tekrarlanır.

Eğer, sistem bu tezde olduğu gibi bir gözetimli eğitimse, yukarıdaki eğitim tamamlandıktan sonra ağırlık matriksleri sabitlenir ve kategorisi bilinmeyen her nitelik vektörü için bu ağırlık matriksleri kullanılarak tahmin yapılır.



Şekil 9 Yapay Sinir Ağları – MLP Yapısı

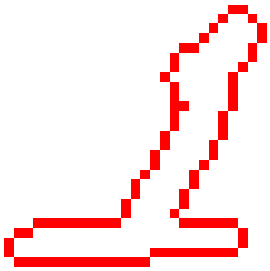
2.12 Zincir Kodları

Bu tezde hiç kullanılmayacak olmalarına karşın; karakter tanıma konusunda oldukça yaygın kullanılmaları nedeniyle zincir-kodları'ndan bahsetmekte yarar var. Bir gri-seviye veya ikili imajın zincir kodunu çıkarmak için önce sınır çizgileri elde edilir. Sınır çizgileri yönleri temsil eden bir alfabeyle kodlanır. Yön alfabesini örneğin şu şekilde tanımlayabiliriz:

- 0 : Doğu (D)
- 1 : Kuzey-Doğu (KD)
- 2 : Kuzey (K)
- 3 : Kuzey-Batı (KB)
- 4 : Batı (B)
- 5 : Güney-Batı (GB)
- 6 : Güney (G)
- 7 : Güney-Doğu (GD)

Eğer imajı ikili imaj olarak tanımlı yeni bir boyuta yeniden-boyutlandırırsak: Eldeki yeni imajı yukarıdaki alfabeyle kodlayabiliriz [18].

-1,-1	0,-1	1,-1
-1,0		1,0
-1,1	0,1	1,1



D:GD:GD:G:GB:G:GB:GB:G:G:G:GB:G:G:GB:G:GB:GB:G:GB:
G:GB:GD:D:D:D:D:D:GD:G:GB:B:B:B:B:B:B:B:B:GB:B:B:B:B:B:
B:B:B:B:B:B:B:B:
KB:K:KD:D:KD:D:D:D:D:D:D:D:D:KD:K:KD:K:KD:KD:K:KD:K:KD
:K:K:K:K:KB:KD:K:KD:D:KD:KD:KD:KD

Şekil 10 Zincir Kodları'nın Çıkarılması

İmajın kodlanmış şeklinden ayırt-edici nitelikler çıkarabiliriz. Örneğin çok sayıda 4 ve 0 grubu varsa, imaj yatay çizgiler içermelidir ve ' – ' veya 'V' olma olasılığı düşüktür.

2.13 Şablon Eşleştirme [19]

Şablon eşleştirme, İÖT ile birlikte en yaygın örüntü tanıma yöntemlerinden biridir. İÖT'den farklı olarak her hangi bir sınıflandırma yapılmaz. Tanınacak nesne, kayıtlı nesne örnekleriyle karşılaştırılır. Bu kayıtlı nesne örneklerine şablonlar olarak isimlendirilir. Yeni nesne hangi kategorinin şablon (veya şablonlarına) daha çok benziyorsa o kategoriye ait olarak tanınır.

2.13.1 Düzeltme mesafesi

Şablon Eşleştirme de İOT gibi nesnelerin benzerlik ya da benzemezliğini ölçmek için bir uzaklık ölçütüne ihtiyaç duyar. Düzeltme Mesafesi, Kolgomorov Mesafesi olarak daha önce işlenen ölçütün özel bir durumudur. Düzeltme Mesafesi iki kod-katarı (örneğin Şekil 10'daki zincir-kod) arasında aşağıdaki yönergeler üzerinden tanımlanmış. Kolgomorov Mesafesidir.

1. Katar içindeki harflerden herhangi birini at.
2. Katar içinde herhangi bir yere harf ekle
3. Katar içinde herhangi bir harfi, başkasıyla değiştir.

Örneğin "istemsiz" sözcüğü (harf-katarı)'nı "istekli"ye çevirmek için;

1. 'z' harfini at
2. 's' harfini 'l' ile değiştir.
3. 'm' harfini 'k' ile değiştir.

olmak üzere üç işlem yapmalıyız. Diğer bir değişle, "istemsiz" ile "istekli" arasındaki düzeltme mesafesi üçtür. Benzer bir sistem tanınmak istenen nesnelere uygulanarak veri tabanında eldeki şablonlardan en yakın olanın kategorisi ile eşlenebilir. Bu örnek teknik Şablon Eşleştirmenin düzeltme mesafesi ile kullanılmasıdır.

3 YETİM PİKSEL ORANI NİTELİKLERİ

Yetim Piksel Oranı (YPO) Nitelikleri çıkarılmak istenilen ikili imaj $A_{m\times n}$ (23) matriksiyle temsil ediliyor olsun.

$$A_{m\times n}=\begin{pmatrix} a_{00} & a_{01} & .... & a_{0n} \\ a_{10} & a_{11} & .... & a_{1n} \\ .. & .. & .... & .. \\ a_{m0} & a_{m1} & .... & a_{mn} \end{pmatrix} \quad a_{ij}\in\{0,1\} \tag{23}$$

Buna göre a_{ij} pikseli aşağıdaki iki kurala uyuyorsa, bu pikseli α açısı yönünde görüşü açık olarak tanımlanır.

1. $a_{ij}=0$ (Piksel beyazdır)
2. $x\in\mathbb{R}\Rightarrow a_{(\lfloor x\cdot\cos(\alpha)\rfloor)(\lfloor x\cdot\cos(\alpha)\rfloor)}=0$ (Bu pikselden α açısı yönünde imajın dış çerçevesine çizilen ışın siyah piksel içermez)

Eğer $A_{m\times n}$ imajına ait a_{ij} pikseli aşağıdaki kurallara uyuyorsa, bu pikseli α açısına göre yetim olarak tanımlanır ve tezde y_α ile temsil edilir:

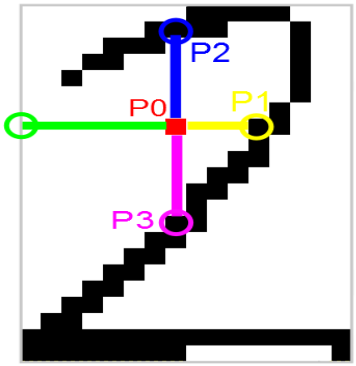
1. a_{ij} pikselinin α açısına göre görüşü açıktır.
2. a_{ij} pikselinin $(\alpha),(\alpha+\frac{\pi}{2}),(\alpha+\pi)$ ve $(\alpha+\frac{3\pi}{2})$ açılarına göre görüşleri kapalıdır.

Sonuç olarak, her bir α açısı için temel bir YPO niteliğini (24)'deki gibi tanımlanabilir.

$$YPO_\alpha = \frac{A_{mxn}\text{imajındaki } Y_\alpha \text{ piksellerin sayısı}}{A_{mxn}\text{imajındaki beyaz piksellerin sayısı}} \tag{24}$$

Bu tezde, özel önem taşıyan $YPO_{0,}YPO_{\frac{\pi}{2}},YPO_{\pi},YPO_{\frac{3\pi}{2}}$ nitelikleri sırasıyla $YPO_{sag},YPO_{ust},YPO_{sol},YPO_{alt}$ sembolleri ile temsil edilip yine sırasıyla sağ-yetim-oranı, üst-yetim-oranı, sol-yetim-oranı, alt-yetim-oranı olarak isimlendirilmiştir. Ek olarak, yine $A_{m\times n}$ imajına ait a_{ij} pikseli aşağıdaki kurala uyuyorsa, bu piksel yetim değil diye tanımlanır ve tezde $y_{değil}$ şeklinde temsil edilir:

1. a_{ij} pikselinin $(\alpha),(\alpha+\frac{\pi}{2}),(\alpha+\pi)$ ve $(\alpha+\frac{3\pi}{2})$ açılarına göre görüşleri kapalıdır.



Şekil 11 Bir Pikselin YPO Nitelikleri

Yukardaki YPO niteliklerine ek olarak, yeni bir YPO niteliğini (tezde olarak simgelenip yetim-değil-oranı olarak isimlendirilecektir) şu şekilde tanımlanır:

$$YPO_{değil} = \frac{A_{mxn}\, imajındaki \;\; Y_{değil}\, piksellerin \;\; sayısı}{A_{mxn}\, imajındaki \;\; beyaz\, piksellerin \;\; sayısı} \qquad (25)$$

Bu tanıma göre beyaz bir pikselin aij pikselinin α açısına göre $\alpha-yetim$, $(\alpha+\pi/2)-yetim$ $(\alpha+\pi/2)-yetim$, $(\alpha+3\pi/2)-yetim$ veya $yetim-degil$ özelliklerinden birini taşıyıp taşımadığı toplu olarak şu şekilde belirlenir.

1. İmajı derece saat yönüne çevir.
2. Yeni imaj üzerinde a_{ij} pikselini içeren satırda aij'nin solunda siyah piksel varsa sol_kapali'a 'true' aksi halde 'false' ata.
3. Eğer a_{ij} pikselini içeren satırda aij'nin sağında siyah piksel varsa sag_kapali'a 'true' aksi halde 'false' ata.
4. Eğer a_{ij} pikselini içeren sutünda aij'nin üstünde siyah piksel varsa ust_kapali'a 'true' aksi halde 'false' ata.
5. Eğer a_{ij} pikselini içeren sutünda aij'nin altında siyah piksel varsa alt_kapali'a 'true' aksi halde 'false' ata.
6. Eğer sag_kapali 'false', sol_kapali, ust_kapali ve alt_kapali 'true' ise a_{ij} pikseli alpha-yetimdir.

7. Aksi takdirde; eğer ust_kapali 'false', sol_kapali, sag_kapali ve alt_kapali 'true' ise a_{ij} pikseli $(\alpha+\pi/2)-yetim$ olarak tanımlanır.
8. Aksi takdirde; eğer sol_kapali 'false', sag_kapali, ust_kapali ve alt_kapali 'true' ise a_{ij} pikseli $(\alpha+\pi)-yetim$ olarak tanımlanır.
9. Aksi takdirde; eğer alt_kapali 'false', sol_kapali, sag_kapali ve ust kapali 'true' ise a_{ij} pikseli $(\alpha+3\pi/2)-yetim$ olarak tanımlanır.

Her beyaz pikselin α açısına göre özelliği yukarıdaki algoritma ile belirlenip aşağıdaki şekilde $YPO_{\alpha}, YPO_{\alpha+\pi/2}, YPO_{\alpha+\pi}, YPO_{\alpha+3\pi/2}, YPO_{degil}$ oranları hesaplanabilir:

1. yposag, ypoust, yposol, ypoalt, ypodegil, beyazpiksel değişkenlerine 0 ata
2. İmajın her bir pikseli için 3, 4, 5, 6 aşamalrını işlet
3. Eğer a_{ij} siyahsa bir sonraki piksele atla
4. a_{ij} pikselini yukarıdaki algoritmaya sok
5. Eğer a_{ij} ; $(\alpha)-yetim$ ise yposag, $(\alpha+\pi/2)-yetim$ ise ypoust, $(\alpha+\pi)-yetim$ ise yposol, $(\alpha+3\pi/2)-yetim$ ise ypoalt, yetim-degil ise ypodegil değerini bir artır.
6. Beyazpiksel değerini bir artır.
7. $YPO_{\alpha}, YPO_{\alpha+\frac{\pi}{2}}, YPO_{\alpha+\pi}, YPO_{\alpha+\frac{3\pi}{2}}, YPO_{degil}$ değerlerini sırasıyla $\frac{yposag}{beyazpiksel}, \frac{ypoust}{beyazpiksel}, \frac{yposol}{beyazpiksel}, \frac{ypoust}{beyazpiksel}$ değerlerini ata.

3.1 YPO Nitelik Çıkarma – Örnek

YPO niteliklerinin çıkarılmasının ve sonuç olarak tanıma açısından yararlılıklarının görüldüğü basit bir ikili tanıma örnek durum çalışması tasarlanabilir. Bu örnek durum çalışmasında Şekil 1'de görülen dört simge iki ve altı olarak ayrıştırılacaktır. Ayrıca, bu örnek kapsamında aşağıdaki üç YPO niteliğinin çıkarılması incelenebilir

1. Yetim-Sol Oranı (YPO_{sol}).

2. Yetim-Sağ Oranı ($YPO_{sağ}$).

3. Yetim-Değil Oranı ($YPO_{değil}$).

Şekil 1'deki dört simge aynı sırayla aşağıda (26), (27), (28), (29) matriksleriyle temsil edilebilir.

$$\mathbf{S_0} = \begin{pmatrix} 0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1&1&1&0&0&1&0&0\\ 0&0&0&0&1&1&1&0&0&0&0&0&0&1&0&0\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{26}$$

$$\mathbf{S_1} = \begin{pmatrix} 0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{27}$$

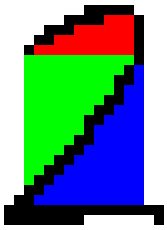
$$\mathbf{S_2} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&1&1&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&1&1&0&0&0 \\ 0&0&0&1&1&0&0&0&0&0&0&1&0&0 \\ 0&0&1&1&0&0&0&0&0&0&1&1&0&0 \\ 0&1&1&1&0&0&0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&1&1&0&0&0 \\ 1&1&0&0&0&0&0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 1&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 1&0&0&0&0&1&1&0&0&0&0&0&0&0 \\ 0&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix} \tag{28}$$

$$\mathbf{S_3} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&0&0&1&1&0&0&1&0 \\ 0&0&0&1&0&1&1&0&0&0&1&0 \\ 0&0&1&0&0&1&0&0&0&0&1&0 \\ 0&1&1&0&1&0&0&0&0&0&1&0 \\ 0&1&0&0&1&0&0&0&0&0&1&0 \\ 1&1&0&1&0&0&0&0&0&1&1&0 \\ 1&0&0&0&0&0&0&0&1&1&0&0 \\ 1&0&0&0&0&0&0&1&0&0&0&0 \\ 1&1&0&0&0&1&1&0&0&0&0&0 \\ 0&0&1&1&1&1&0&0&0&0&0&0 \end{pmatrix} \tag{29}$$

Şekil 1a imajı S_0 için bu yetim-sol, yetim-sağ, yetim-değil pikselleri sırasıyla yeşil, mavi ve kırmızı renklerle Şekil 11'de gösterilmiştir. Her gruptaki piksel sayıları buna göre şöyledir;

Yetim-Sol Piksel Sayısı = 81

Yetim-Sağ Piksel Sayısı = 74



Şekil 12 Şekil 1a yetim-sol (yeşil), yetim sağ (mavi) ve yetim-değil (kırmızı) pikseller

Yetim-Değil Piksel Sayısı = 27
Toplam Beyaz Piksel Sayısı = 282

Piksel Sayılarından $YPO_{sol}, YPO_{sağ}$ ve $YPO_{değil}$ hesaplanırsa elde edilecek sonuçlar aşağıdaki gibi olur:

$$YPO_{sol}(S_0)=\frac{81}{282}=0.287 \quad (30)$$

$$YPO_{sağ}(S_0)=\frac{74}{282}=0.261 \quad (31)$$

$$YPO_{degil}(S_0)=\frac{27}{282}=0.287 \quad (32)$$

Şekil 1b S_1 , Şekil 1c S_2 ve Şekil 1d S_3 simgeleri için de $YPO_{sol}, YPO_{sağ}$ ve $YPO_{değil}$ nitelikleri çıkarılırsa,

$$YPO_{sol}(S_1)=\frac{43}{279}=0,154 \quad (33)$$

$$YPO_{değil}(S_1)=\frac{31}{279}=0,111 \quad (34)$$

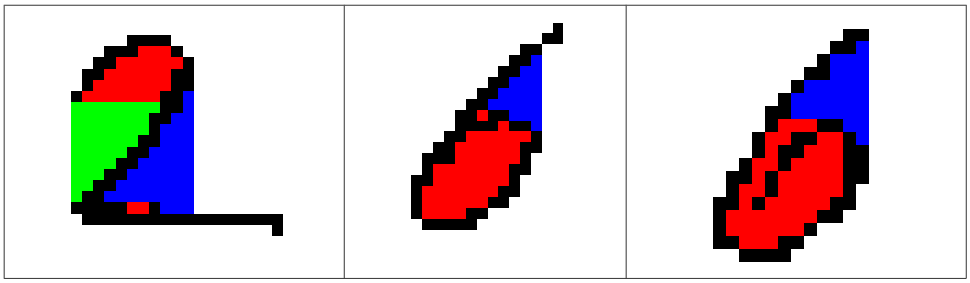
$$YPO_{sol}(S_2)=\frac{0}{224}=0 \quad (35)$$

$$YPO_{sağ}(S_2)=\frac{19}{224}=0,085 \quad (36)$$

$$YPO_{değil}(S_2)=\frac{51}{224}=0,228 \quad (37)$$

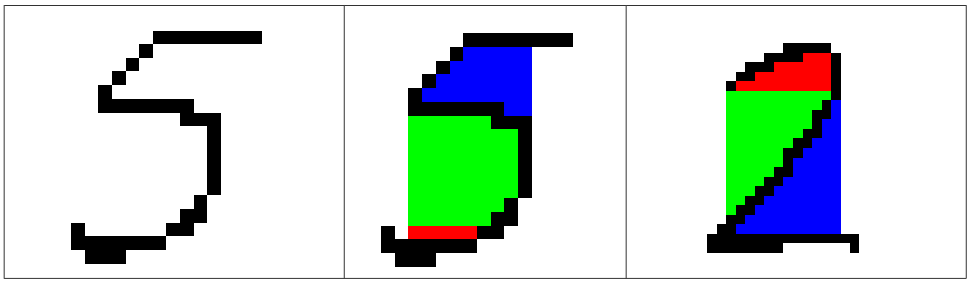
$$YPO_{sol}(S_3)=\frac{0}{163}=0 \quad (38)$$

$$YPO_{değil}(S_3)=\frac{52}{163}=0{,}319 \quad (39)$$



Şekil 13 Şekil 1b, 1c, 1d yetim-sol (yeşil), yetim-sağ (mavi) ve yetim-değil (kırmızı) pikseller

(33), (34), (35), (36), (37), (38) ve (39) değerleri elde edilir. Fark edildiği gibi bu üç nitelik birlikte simgeler arasında güçlü bir ayırt edicidir. Hatta, '2' ve '6'yı birbirinden ayırt etmek konusunda diğer ikisinden daha güçlü olan YPO_{sol} ikili sınıflandırma için tek başına kullanılabilir. '6' olarak kategorize edilecek iki simgede de YPO_{sol} 0'dır. Oysa '2' simgesi önemli sayıda yetim-sol piksel içerir. Bunun sonucunda '2' olarak kategorize edilmesi gereken bir simgenin YPO_{sol} niteliğinin 0 olması istatistiksel olarak çok küçük bir olasılıktır. Yani, bu iki kategori arasındaki bir sınıflandırmayı YPO_{sol} niteliğini kullanırsak; YPO_{sol} değeri 0.1'den büyük değerleri '2' diğerlerini '6' olarak sınıflandırarak yapabiliriz. Tabi ki YPO_{sol} tek başına daha fazla sayıda kategori olduğu zaman işe yaramaz. Ama, Şekil 13'deki renklendirmeden fark edileceği gibi, aynı kategoriye sahip simgelerin istatistiksel olarak benzer temel YPO niteliklerine sahip olma eğilimleri oldukça güçlüdür. Bu nedenle, temel YPO nitelikleri karakter tanıma konusunda oldukça güçlü (ayırt edici) niteliklerdir.



Şekil 14 '2' ve '5' Simgelerinin YPO Nitelikleri Renklendirilmesi

3.2 Yerel Yetim Piksel Oranı Nitelikleri

Şekil 14'de '2' ve '5' kategorilerine ait iki simge görünüyor. Daha önce kullanılmamış olan '5' simgesi S_5

$$\mathbf{S_3} = \begin{pmatrix} 0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&1&1&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 1&1&1&1&1&1&1&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix} \tag{40}$$

matrisi ile temsil ediliyor olsun. Her iki simgede de yetim-sol ve yetim-sağ pikselleri baskın ve daha az sayıda yetim-değil piksel var. '2'ye ait yetim-değil pikseller daha fazla olmasına rağmen bu iki simge için yetim-değil pikselleri de iyi bir ayırt edici değil, çünkü yazarı tarafından alt ucu kısa tutulmuş '5' simgesini başka bir yazar alt ucunu uzun tutabilirdi ve '2' simgesine yakın yetim-değil pikseli içerecek bu simgenin hala '5' olarak sınıflandırılması gerekirdi. Yani bu iki simge arasında $YPO_{değil}$ niteliği simgenin kategorisine değil yazarın yazım biçimine ait bilgi taşımaktadır. Kısaca bu üç nitelik '2' ve '5' arasında yapılacak bir sınıflandırma için yetersizdir. Ama, dikkat edilirse yetim-sol pikselleri '2' simgesinde yukarıda olma eğilimindeyken '5' simgesinde yetim-sağ piksellerinin altında olma eğilimindedir. Bu eğilim, yalnızca bu iki simgeye bağlı değildir. '2' ve '5' olarak tanımlanacak tüm simgelerde vardır ve yazardan bağımsızdır. Ne yazık ki bu eğilim temel YPO nitelikleriyle ölçülemez çünkü temel YPO nitelikleri piksel konumlarından tamamen bağımsızdır. Bu amaçla, temel YPO nitelikleri pikselin konum bilgisinden de etkilenecek şekilde değiştirilmiştir ve bu değiştirilmiş niteliklere yerel YPO nitelikleri adı verilmiştir. A_{mxn} (40) matriksi için tanımlayabilir. Eğer A_{mxn} imajına ait a_{ij} pikseli aşağıdaki kurallara uyuyorsa, bu pikseli α ve (41) ile tanımlanan dikdörtgene göre yetim olarak tanımlanır ve

tezde ile $y_\alpha\ [r_{0,}r_{1,}r_{2,}r_3]$ temsil edilir:

1. a_{ij} pikselinin α açısına göre yetimdir.

2. $\frac{r_0\times m}{100} \le i \le \frac{r_1\times m}{100} \wedge \frac{r_2\times n}{100} \le j \le \frac{r_3\times n}{100}$ koşulu sağlanmalıdır.

$$R : [r_0, r_1, r_2, r_3]\ 0 \le k_n \le 100 \wedge n \in \{0,1,2,3\} \tag{41}$$

Benzer şekilde, $y_{değil}\ [r_{0,}r_{1,}r_{2,}r_3]$ şu şekilde tanımlanabilir:

1. a_{ij} pikseli yetim değildir.

2. $\frac{r_0\times m}{100} \le i \le \frac{r_1\times m}{100} \wedge \frac{r_2\times n}{100} \le j \le \frac{r_3\times n}{100}$ koşulu sağlanmalıdır.

Sonuç olarak yerel YPO nitelikleri;

$$YPO_\alpha\ [r_{0,}r_{1,}r_{2,}r_3] = \frac{A_{m\times n}\ imajındaki\ y_\alpha\ [r_{0,}r_{1,}r_{2,}r_3]\ piksellerin\ \ sayısı}{A_{m\times n}\ imajındaki\ \ beyaz\ \ piksellerin\ \ sayısı} \tag{42}$$

$$YPO_{değil}\ [r_{0,}r_{1,}r_{2,}r_3] = \frac{A_{m\times n}\ imajındaki\ y_{değil}\ [r_{0,}r_{1,}r_{2,}r_3]\ piksellerin\ \ sayısı}{A_{m\times n}\ imajındaki\ \ beyaz\ \ piksellerin\ \ sayısı} \tag{43}$$

olarak tanımlanır. YPO'daki gibi yerel YPO için de , $(\alpha = 0)$, $(\alpha=\frac{\pi}{2})$, $(\alpha = \pi)$, $(\alpha = 3\frac{\pi}{2})$, (α) ;değerleri bu tezde özel olarak, $YPO_{sağ}[r_{0,}r_{1,}r_{2,}r_3]$, $YPO_{üst}[r_{0,}r_{1,}r_{2,}r_3]$, $YPO_{sol}[r_{0,}r_{1,}r_{2,}r_3]$ ve $YPO_{alt}[r_{0,}r_{1,}r_{2,}r_3]$ olarak isimlendirilmiştir.

Bu tezde, üç yerel YPO niteliği kullanıldı. Bunlar sırasıyla; '2' ve '5' simgelerini ayırmak için $YPO_{sağ}[0{,}100{,}50{,}100]$, '3' ve '7' simgelerini ayırmak için $YPO_{sol}[0{,}100{,}75{,}100]$, '4' ve '6' simgelerini ayırmak için $YPO_{değil}[0{,}100{,}50{,}100]$ nitelikleridir. Bu yöntemin yararlılığını görmek için a (S_0) ve b (S_5) imajlarını '2' ve '5' kategorilerine ikili olarak ayırmak için $YPO_{sağ}$ ve $YPO_{sağ}[r_{0,}r_{1,}r_{2,}r_3]$ nitelikleri;

$$YPO_{sağ}(S_0)=\frac{74}{282}=0.261 \quad (44)$$

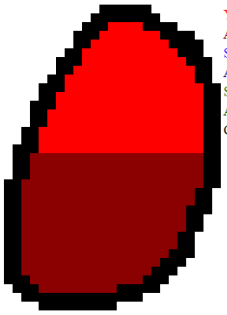
$$YPO_{sağ}(S_5)=\frac{28}{195}=0.144 \quad (45)$$

$$YPO_{sağ}(S_0)[0,100,50,100]=\frac{68}{148}=0.460 \quad (46)$$

$$YPO_{sağ}(S_5)[0,100,50,100]=\frac{0}{106}=0 \quad (47)$$

çıkarılır. Görüldüğü gibi, $YPO_{sağ}$ görece yakın değerler verirken $YPO_{sağ}(S_0)$ $[0,100,50,100]$ niteliğinde ise iki kategori için alınan değerler arasındaki makas açılmaktadır. Bu eğilim, yazardan bağımsız olarak istatistiksel açıdan oldukça makul bir sonuçtur. '5' simgesinin alt yarısında yetim-sağ pikseli bulmak oldukça düşük bir olasılıktır. Oysa '2' simgesinde bu olasılık oldukça yüksektir.

YPO/YYPO 15 kategoriden daha fazla kategorinin olduğu tanıma sistemleri için de kullanılabilir. İdeal durumda, her YPO/YYPO niteliğinin grubu 2'ye böldüğünü düşünülürse n karakteri tanımak için $[\log_2(n)]$ YPO/YYPO karakterine ihtiyaç duyulması gerekir. Tabi, pratikte iki nedenden ötürü bu sayı daha yüksek olacaktır. İlki, YPO/YYPO nitelik vektörünün tanımladığı kategoriler, amaçlanan kategori grubuyla birebir örtüşmeyecek bunun için vektöre daha fazla nitelik eklemek gerekecektir. İkincisi, YPO/YYPO nitelikleri birbirini yüzde yüz karşılıklı dışlamak zorunda değildir. Bu durumda gerekli nitelik sayısını arttırabilir. Tezde verilen 15 kategori için 6 YPO/YYPO niteliği gerekti. Buna göre kategori sayısı ile gerekli nitelik sayısı arasındaki ilişki $1.5\ [\log_2(n)]$ olarak yazılabilir.

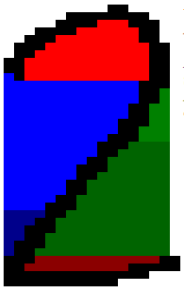


Yetim-Değil Oranı	0,723188
Alt Yarı için Yetim-Değil Oranı	0,411594
Sol-Yetim Oranı	0,000000
Alt Çeyrek için Sol-Yetim Oranı	0,000000
Sağ-Yetim Oranı	0,000000
Alt Yarı için Sağ-Yetim Oranı	0,000000
Genişlik Yükseklik Oranı	1,400000

a

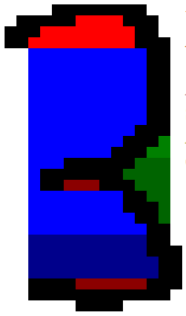
Yetim-Değil Oranı	0,000000
Alt Yarı için Yetim-Değil Oranı	0,000000
Sol-Yetim Oranı	0,000000
Alt Çeyrek için Sol-Yetim Oranı	0,000000
Sağ-Yetim Oranı	0,000000
Alt Yarı için Sağ-Yetim Oranı	0,000000
Genişlik Yükseklik Oranı	17,000000

b



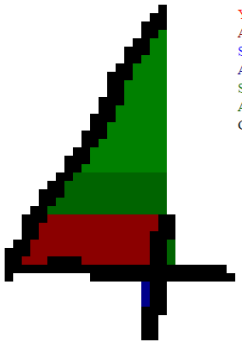
Yetim-Değil Oranı	0,195918
Alt Yarı için Yetim-Değil Oranı	0,044898
Sol-Yetim Oranı	0,346939
Alt Çeyrek için Sol-Yetim Oranı	0,026531
Sağ-Yetim Oranı	0,295918
Alt Yarı için Sağ-Yetim Oranı	0,267347
Genişlik Yükseklik Oranı	2,176471

c



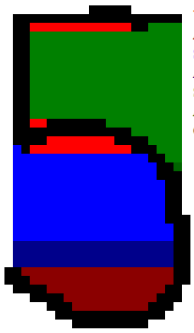
Yetim-Değil Oranı	0,096154
Alt Yarı için Yetim-Değil Oranı	0,028846
Sol-Yetim Oranı	0,564103
Alt Çeyrek için Sol-Yetim Oranı	0,134615
Sağ-Yetim Oranı	0,054487
Alt Yarı için Sağ-Yetim Oranı	0,044872
Genişlik Yükseklik Oranı	1,866667

d



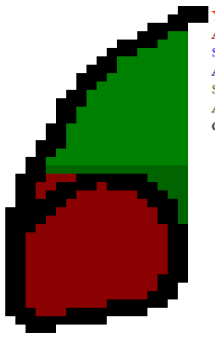
Yetim-Değil Oranı	0,084673
Alt Yarı için Yetim-Değil Oranı	0,084673
Sol-Yetim Oranı	0,003215
Alt Çeyrek için Sol-Yetim Oranı	0,003215
Sağ-Yetim Oranı	0,169346
Alt Yarı için Sağ-Yetim Oranı	0,069668
Genişlik Yükseklik Oranı	1,481481

e



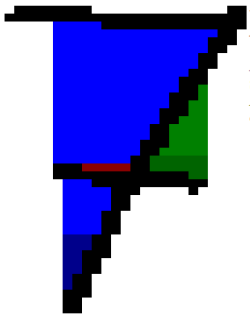
Yetim-Değil Oranı	0,157975
Alt Yarı için Yetim-Değil Oranı	0,105828
Sol-Yetim Oranı	0,361963
Alt Çeyrek için Sol-Yetim Oranı	0,087423
Sağ-Yetim Oranı	0,320552
Alt Yarı için Sağ-Yetim Oranı	0,001534
Genişlik Yükseklik Oranı	1,681818

f



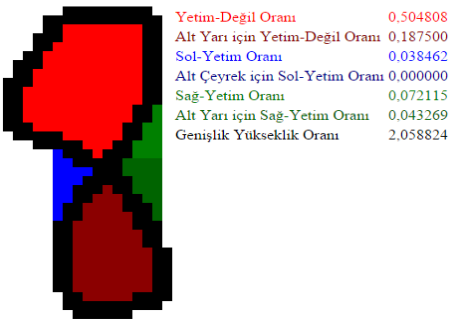
Yetim-Değil Oranı	0,286656
Alt Yarı için Yetim-Değil Oranı	0,286656
Sol-Yetim Oranı	0,000000
Alt Çeyrek için Sol-Yetim Oranı	0,000000
Sağ-Yetim Oranı	0,283361
Alt Yarı için Sağ-Yetim Oranı	0,042834
Genişlik Yükseklik Oranı	1,950000

g



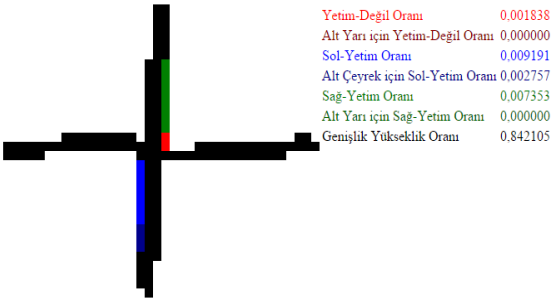
Yetim-Değil Oranı	0,006105
Alt Yarı için Yetim-Değil Oranı	0,006105
Sol-Yetim Oranı	0,317460
Alt Çeyrek için Sol-Yetim Oranı	0,017094
Sağ-Yetim Oranı	0,048840
Alt Yarı için Sağ-Yetim Oranı	0,017094
Genişlik Yükseklik Oranı	1,560000

h



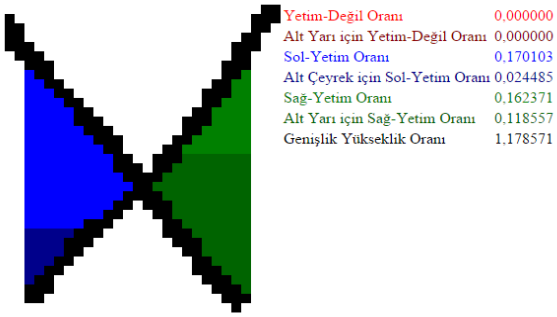
ı

j



k

l



m

n

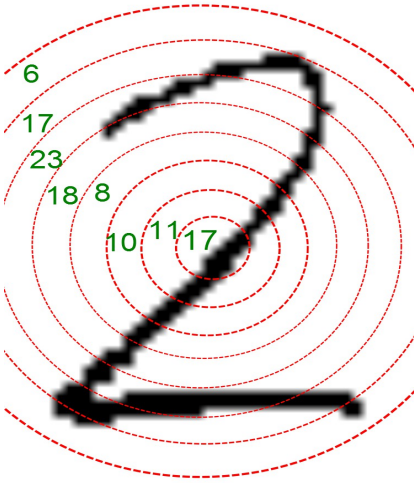
o

Şekil 15 YPO/YYPO Nitelik Vektörü Örnekleri

4 1B-DALGACIK DÖNÜŞÜMÜ İLE NİTELİK ÇIKARTMA

İÖT'de nitelik çıkarma için kullanılan en yaygın yöntemlerden biri Dalgacık dönüşümüdür. Karakter Tanıma işleminde tanınması gereken imaj iki boyutlu olduğu için ya performans açısından çıkarması daha zor olan 2B-dalgacık dönüşümü kullanılır ya da iki boyutlu imaj bir izdüşüm yöntemiyle bir boyutlu bir fonksiyona çevrilip ondan sonra 1B-dalgacık dönüşümü bu fonksiyona uygulanır. Bu tezin önerdiği sistem dalgacık dönüşümüne dayanan bir nitelik çıkarma kullanmamasına rağmen tez sonuçlarını karşılaştırıldığı iki yöntemden biri [20] makalesi temel alınarak yazılan 1B-Dalgacık Dönüşümü ile Nitelik Çıkartma yöntemidir. Bu yöntemde, ikili imaj halka-izdüşüm histogram tekniğiyle ikili imajdan keyfi olarak seçilen bir n için 2^n boyutunda bir sayı dizisi üretilir. Elde edilen diziye 1B-dalgacık dönüşümü uygulanıp aynı uzunlukta yeni bir sayı dizisi elde edildikten sonra bu dizinin başından itibaren istenen sayıda elemanı nitelik vektörü olarak kullanılır.

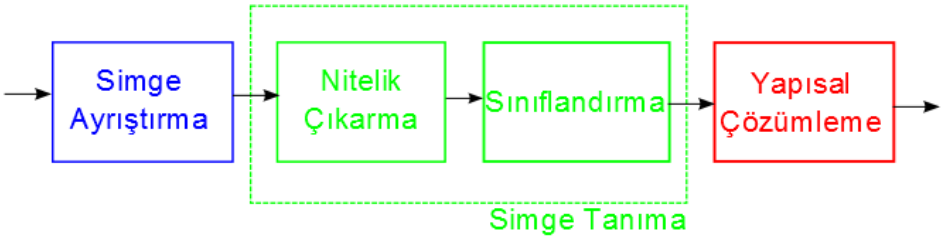
Bu tezde, halka-izdüşüm histogram tekniğiyle üretilen dizinin uzunluğu 32 olarak seçilmiş ve dönüşüm olarak elde edilen 32 değerden ilk 10'u nitelik vektörü olarak kullanılmıştır. Bu yöntemin tüm ayrıntısı [20] makalesinde bulunabilir.



Şekil 16 Halka-İzdüşüm Histogram Tekniği

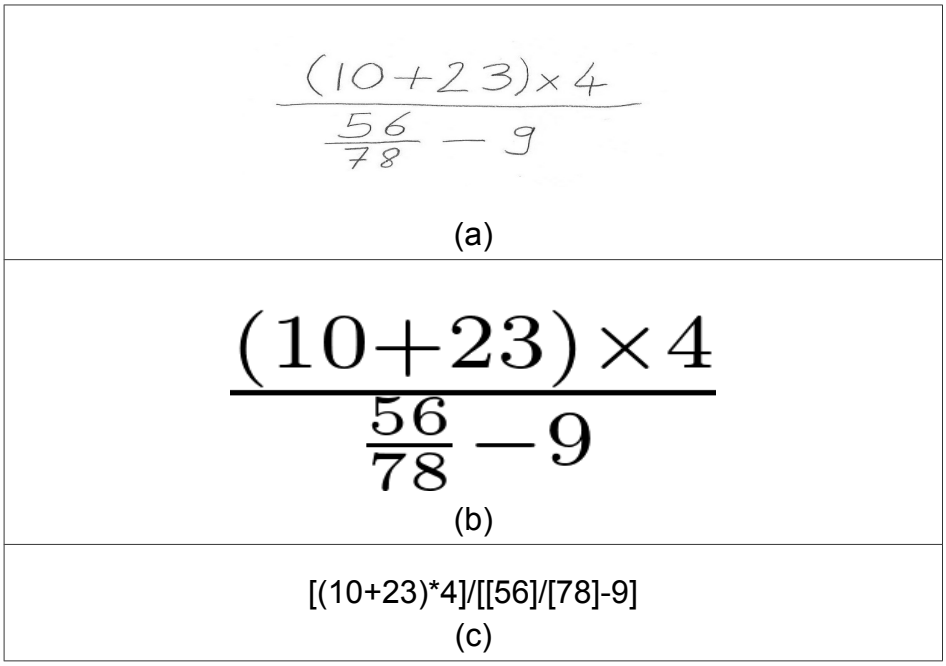
5 ÖNERİLEN SİSTEM MİMARİSİ

Tezde önerilen mimari, Simge Ayrıştırma, Simge Tanıma ve Yapısal Çözümleme ve TEX'e Çevirme algoritmalarından oluşuyor (Şekil 17). Her algoritma, kendinden öncekinin (veya öncekilerinin) çıktılarını girdi olarak kullanır. Şekil 18'de örnek bir sistem girdisi (png imajı) ve bu girdiden beklenen TEX çıktısı görülebilir.



Şekil 17 Önerilen Sistem Aşamaları

Sistemin girdi imajının Şekil 18'a kadar temiz ve okunaklı olmayabilir. Bu durumda, girdinin Simge Ayrıştırma'dan önce okunurluğunu artırmak için daha karmaşık bir ön-işleme aşaması gerekebilir.



Şekil 18 (a) Orginal İmaj (b) TEX Çıktısı (c) Anlamlı Matematiksel İfade

5.1 Simge Ayrıştırma

Mimarinin simgeleri birbirinden ayırabilmesi için simge kavramının ne olduğunu bilmesi gerekir. Bu nedenle, öncelikle simge kavramını tanımlamalıyız.

Ayrıştırma aşamasında; bir ikili imaj üzerindeki birbirine 8-bağlı (8-connected) siyah pikseller bir simgedir. Bu tanımın temel dezavantajı ikili imaj için yapılmış olmasıdır. Oysa ayrıştırma girdisi gri-seviye imajı olabilir. Bu nedenle, gri-seviye imajı ayrıştırılmadan önce eşik değeri 220 olan bir ikili süzgeçten geçirilmiştir.

5.1.1 Ayrıştırma

Yukarıda elde edilmiş olan ikili imajı A_{mxn} matriksiyle temsil edilebilir.

$$A_{m\times n} = \begin{pmatrix} a_{00} & a_{01} & .... & a_{0n} \\ a_{10} & a_{11} & .... & a_{1n} \\ .. & .. & a_{ij} & .. \\ a_{m0} & a_{m1} & .... & a_{mn} \end{pmatrix} \quad a_{ij}\in\{0,1\} \tag{48}$$

Eğer imaj K sayıda simge içeriyorsa, ayrıştırma algoritması,

$$\vec{S}=\begin{bmatrix} S_0 \\ S_1 \\ .... \\ S_k \\ .... \\ S_{K-1} \end{bmatrix} \tag{49}$$

vektörünü üretir. Burada, her bir S_k algoritma tarafından üretilen bir simge kümesini temsil eder. Buna göre, eğer $a_{ij}=1$ (siyah piksel) ve $a_{ij}k$ 'ıncı simgeye ait bir piksel ise (ij) vektörü S_k'ın bir elemanı kabul ederiz. Buna göre, her bir S_k iki boyutlu vektörlerin (piksellerin) kümesidir ve şöyle yazılabilir:

$$S_k = \{(x_0, y_0), (x_1, y_1), ...(x_t, y_t), ...(x_T, y_T)\} \tag{50}$$

$$0 \le x_t \le n - 1 \tag{51}$$

$$0 \le y_t \le m - 1 \tag{52}$$

Bu tanımlar ışığında, aşağıdaki yinelgen ayrıştırma algoritması A_{mxn} imajından $\vec{S}$ simge kümesini üretecektir:

simgeyeEkle(A, i, j, k, komsuluk)

1. a_{ij} = 0 (beyaz piksel) veya a_{ij} zaten bir simgeye ait ise fonksiyondan dön.
2. a_{ij} bir simgeye ait diğer bir pikselin komşusu değil ise (komsuluk=false) k'yı 1 artır.
3. $a_{ij} S_k$ 'ya ekle.
4. simgeyeEkle fonksiyonunu (i,j)'ye 8-bağlı tüm pikseller için komsuluk = true parametresiyle çağır.

Simgeye Ekle fonksiyonunu A_{mxn} imajının tüm pikselleri için çağırıldığında A_{mxn} imajı $\vec{S}$ simge kümelerine ayrışır.

5.2 Simge Tanıma

Simge Tanıma algoritması; ayrıştırma algoritmasının ürettiği simge kümesinin $(\vec{S})$ elemanlarını, bir kategori kümesi K'nın elemanları ile eşler. Girişte belirtildiği gibi sınırlandırmalara uygun olarak bu tezde kullanılan kategori kümesidir (53)'dir.

$$K = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, x, (,) \} \quad (53)$$

Ayrıca, eşleştirme işlemi için kullanılan yöntemler içinde en yaygını olan İstatiksel Örüntü Tanıma (İÖT) kullanılmıştır. Bu yöntem;

1. Nitelik vektörü (NV) çıkarma.
2. NV'yi sınıflandırma

olarak iki aşamadan oluşur. Her iki aşama için de temel bir seçim yapılmalıdır: Hangi nitelik vektörü seçilmeli ve hangi sınıflandırma algoritması kullanılmalı. Bu iki aşama, istenirse farklı nitelik vektörleri ve sınıflandırma algoritmalarıyla birden çok kez tekrarlanabilir.

Bir simgenin niteliği; o simgeden, determinist bir fonksiyonla üretilmiş herhangi bir skaler değerdir. [10] Bununla beraber, simgenin tanınması, nadiren tek bir nitelikle yapılabilir. Genellikle birden fazla nitelik kullanılarak işleme aşamasında kullanmak için nitelik vektörü oluşturulur [10]. Ne yazık ki nitelik vektöründe, kaç farklı nitelik olacağı ve hangi niteliklerin seçileceğine dair bir standart yoktur. Birlikte kullanılan niteliklerin bir kısmı ortak bilgi içerebilir (ortoganal olmayabilir). Bu, aynı bilginin iki veya daha çok işlenmesi, yani tanıma performansının düşmesine yol açar. Ya da, gerektiğinden az nitelik seçilmiş ise tanıma doğruluğu dramatik bir şekilde düşebilir. Yani, nitelik vektörünün belirlenmesi sistem tasarımcısı tarafından çözülmesi gereken karmaşık bir sorundur.[10]

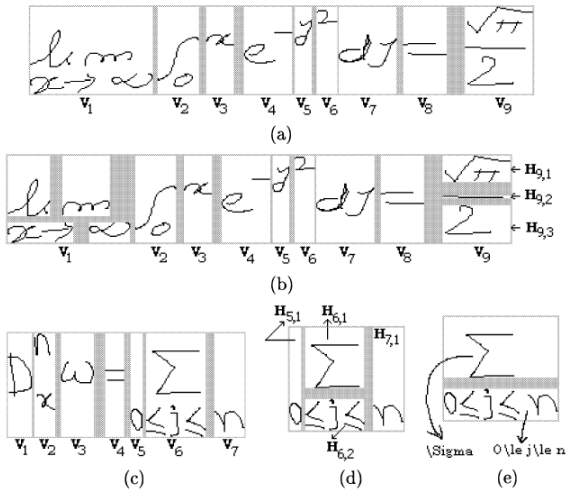
Nitelik vektörü saptandıktan sonra, vektör bir sınıflandırma algoritmasıyla sınıflandırılır. Testler ve Sonuçlar bölümünde, görüldüğü gibi sistem Weka'ya ait MLP, IBk(KNN) ve KStar olmak üzere üç farklı sınıflandırma algoritmasıyla test edilmiş, bunlardan KStar eğitim hızı ve doğruluk açısından optimum sonuçları nedeniyle tercih edilmiştir. Bununla beraber, kodun konfigürasyonu değiştirilerek sistem başka bir sınıflandırma algoritmasıyla birlikte kullanılabilir. Bu tezde önerilen tanıma aşamaları şöyledir:

1. Kategori kümesini $K \leftarrow \{0,1,2,3,4,5,6,7,8,9,(,),+,-,x\}$ olarak ata.
2. NV nitelik vektörünü $NV \leftarrow \{YPO_{değil}, YPO_{sol}, YPO_{sağ},\}$ olarak ata.
3. NV nitelik vektörüne $YPO_{sağ}, [0,100,50,100]$ (Yerel YPO-sağ imajın alt yarısı için) YYPO niteliğini ekle
4. NV nitelik vektörüne $YPO_{sol}, [0,100,75,100]$ (Yerel YPO-sol imajın alt çeyreği için) YYPO niteliğini ekle.
5. NV nitelik vektörünü $YPO_{değil}, [0,100,50,100]$ (Yerel YPO-değil imajın alt yarısı için) YYPO niteliğini ekle.
6. NV nitelik vektörüne genişlik-yükseklik-oranı niteliğini ekle.
7. Tüm simgeleri K kategori kümesi için 7 elemanlı NV vektörüyle sınıflandır.

5.3 Yapısal Çözümleme

Yapısal Çözümleme aşamasında mimari simgeler arası uzamsal ilişkileri belirler ve matematiksel ifadeyi buna göre yorumlar. Önerilen algoritma bir çok açıdan Garain'de [9] önerilenin benzeridir. Bu makale de Garain [9] ile benzer şekilde imajı sırayla dikey ve yatay olarak düz çizgilerle kesmeyi elde edilen imaj parçalarında da aynı işlemi yinelgen olarak uygulamayı öneriyorlar [9] [21]. Yatay ve dikey kesme işlemi artık ayırma yapılamayıncaya kadar devam eder. Kesme işlemi yapılamıyorsa;

1. İmaj parçası tek bir simgeden oluşmaktadır ya da,
2. Yatay veya dikey tek bir doğruyla ayrılmayan simgelerden oluşmaktadır.



Şekil 19 Çevrimdışı İşleme: Dikey ve Yatay Ayrıştırma. [9, Şekil 5]'ten alınmıştır.

İstenmeyen ikinci durumda bileşik simgeli imaj parçası; Hg'deki gibi; ayrıca analiz edilir. Bunla beraber, Bu tezde [9]'dan farklı olarak gerçek ikili imajı değil, ayrıştırma verilerini kullanarak yaratılmış bir ızgaranın yatay ve dikey parçalara ayrılması tercih ediliyor. Bu ızgara, yapısal çözümlemenin ilgili olduğu tüm ilişkiler açısında gerçek imaj yerine kullanılabilir. Böylece, ayrıştırma işlemi ile yapısal çözümlemeyi birbirinden ayrılmış olur.

$$\vec{S}=\begin{bmatrix} S_0 \\ S_1 \\ .... \\ S_k \\ .... \\ S_{K-1} \end{bmatrix} \quad (54)$$

$$\vec{C}=\begin{bmatrix} C_0 \\ C_1 \\ .... \\ C_k \\ .... \\ C_{K-1} \end{bmatrix} \quad (55)$$

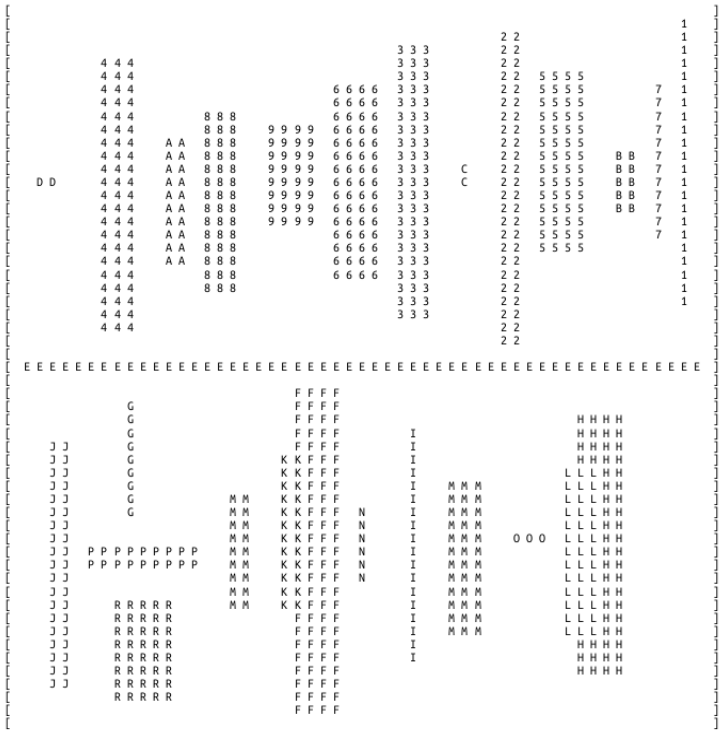
$\vec{S_k}$ imajdan ayrıştırılmış k. simge $\vec{C_k}$ $\vec{S_k}$ ve'ın ait olduğu kategori olarak düşündüğümüzde, yapısal çözümlemenin ilk aşaması ayrıştırma verilerinden ızgarayı aşağıdaki şekilde çıkartabiliriz:

1. $\vec{S_k}$ simgelerinin $x_m\text{in}$ ve $x_m\text{ax}$ (simge piksellerinin x koordinatlarının minimum ve maksimumu) özelliklerini birleştirerek m uzunluğunda $\vec{X_s}$ vektörünü oluştur.
2. $\vec{X_s}$ simgelerinin $\vec{X_s}$ ve $y_m\text{ax}$ (simge piksellerinin y koordinatlarının minimum ve maksimumu) özelliklerini birleştirerek n uzunluğunda $\vec{Y_s}$ vektörünü oluştur.
3. $\vec{X_s}$ ve $\vec{X_s}$ sırala
4. m x n uzunluğunda bir sıfır matriksi yarat (G_{mxn})
5. Her bir simge için $\vec{S_k}$ için $\vec{X_s}$ ve $\vec{Y_s}$ vektörlerindeki indekslerinden oluşan dikdörtgeni k değeri ile işaretle.

Bu algoritma kullanılarak Şekil 19'dan oluşturulmuş örnek bir ızgara matriks (0'lar gösterilmemiş ve iki basamaklı sayılar yerine harfler kullanılmıştır) Şekil 20'dedir. Izgara oluşturulduktan sonra bu ızgara aşağıdaki algoritma ile yinelgen olarak dilimlenir.

1. Çıktı dizisine sütun başlangıcı işareti konur.

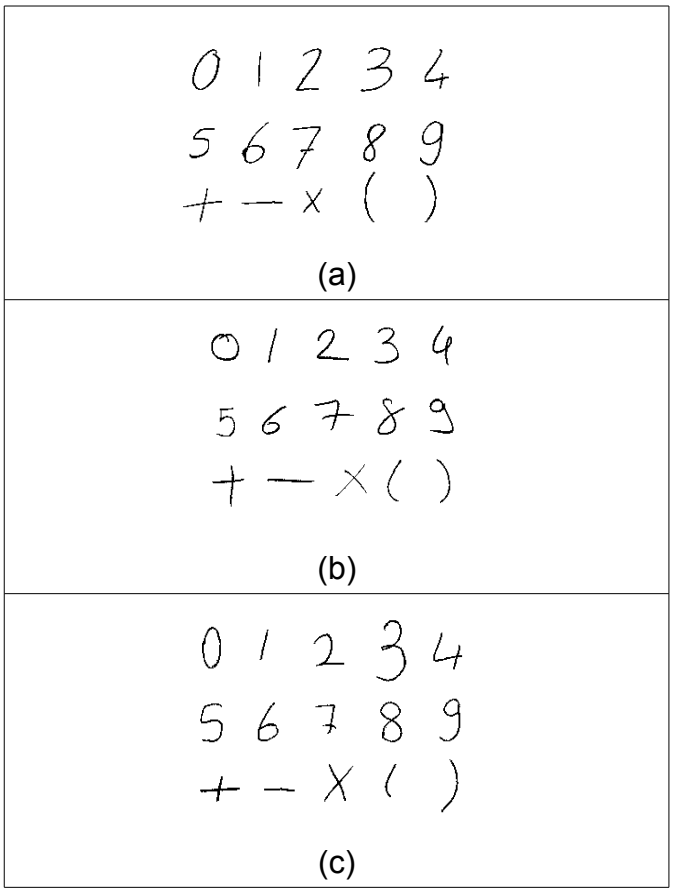
2. Izgara yukarıdan aşağıya tümü 0 olan sütunlar kullanarak dilimlenir. Bu kurala uyan komşu birden fazla sütun varsa biri kullanılır. Dikey dilimlenmiş ızgara parçaları oluşur.
3. Çıktı dizisine sütun sonu işareti konur.
4. Çıktı dizisine satır başlangıcı işareti konur.
5. Her ızgara parçası sırayla, soldan sağa tümü 0 olan satırlar kullanarak dilimlenir. Bu kurala uyan komşu birden fazla sütun varsa biri kullanılır.
6. Çıktı dizisine satır sonu işareti konur.
7. Eğer birden fazla ızgara parçası varsa 1'e dönülür.
8. Izgara parçası tek simgeden oluşuyorsa, bu simge çıktı dizisine eklenir.
9. Eğer ızgara parçası birden fazla yapışık simgeden oluşuyorsa temas sıralarına göre (soldan sağa ya da yukarıdan aşağıya) simgeler çıktı dizisine eklenir.



Şekil 20 Şekil 18a'ya ait Yerleşim Çözümleme Izgarası

Yinelgen dilimleme işlemi bu tezin kapsadığı temel matematiksel işlemler için yeterlidir. Daha karmaşık yapısal çözümleme gerektiren işlemler için Xiang'ın makalesi [22] incelenebilir.

6 DENEYLER



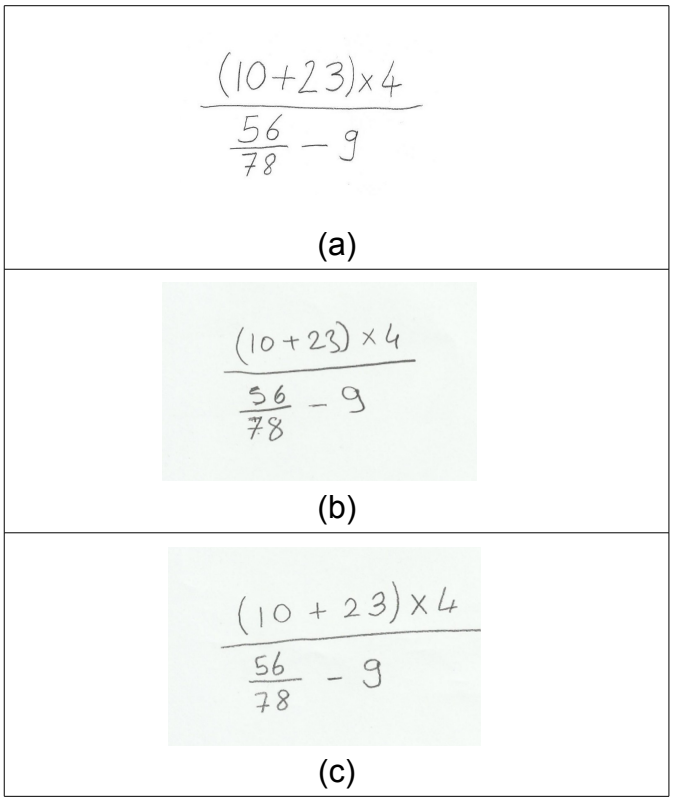
Şekil 21 (a) Ceyhun (b) Eren (c) Cihan El Yazısı Örnekleri

MİT'i sisteminin, doğruluk ve hız açısından başarımını ölçmek için üç grup deney yapılmıştır. İlk gruptaki 4 deney, sisteme en uygun sınıflandırma yöntemini seçmek için KStar, KNN ve MLP yöntemlerini karşılaştırmaktadır. İkinci grup deneyler sabit sınıflandırma yöntemi altında bu makalenin önerdiği YPO ve YYPO'dan oluşturulmuş nitelik vektörü ile birlikte üç nitelik vektörünün başarımını ölçmektedir. Üçüncü grup, sistemi bütün olarak test etmektedir. Bunun için sistem matematiksel ilişkiler içeren gerçek bir imajla test edilmekte ve aynı matematiksel ilişkileri içeren anlamlı bir metin üretilip üretilmediğine bakılmakta, böylece sistemin üç aşaması (Ayrıştırma, Simge Tanıma, Yapısal Çözümleme) birlikte test edilmektedir.

Bu deneylerde kullanılan tüm testler ve tez tarafından önerilen sistem Java (JDK 1.7) ile yazılıp derlenmiş; Windows 8.1 İşletim Sistemi üzerinde Eclipse Java EE IDE for Web Developers Luna Release 4.4.0 - Debug mod'da çalıştırılmıştır.

Ayrıca bahsi geçen tüm sınıflandırma yöntemleri için Weka 3.7.11 kütüphanesi, nitelik vektörlerinden birinde kullanılan dalgacık dönüşümü için JWave 2.0 kullanılmıştır.

Deneylerde kullanılmak üzere üç farklı kişiden toplan 4535 örnek alınmıştır. Bu kişiler deney sonuçlarının verildiği çizelgelerde ceyhun (1508 örnek), eren (1507 örnek) ve cihan(1520 örnek) olarak geçmektedir.



Şekil 22 (a) Ceyhun (b) Eren (c) Cihan El Yazısı Örnekleri Tam Formül

6.1 Grup 1: Sınıflandırma Yöntemleri Karşılaştırması

Bu grupta Tanıma aşaması tek başına çalıştırılmış ve her bir deneyde Sınıflandırma Yöntemi dışındaki diğer parametreler sabit bırakılmıştır. Çizelge 3'de üç yazara ait tüm örnekler, test grubu eğitim grubunun yarısı olacak şekilde rastgele iki gruba ayrılmıştır. Eğitim ve test grubu aynı yazarlara ait olduğu için doğruluk oranları çok yüksek ve birbirine yakındır. Çizelge 4, Çizelge 5 ve Çizelge 6'de eğitim ve test grupları ayrı yazara aittir. Her bir deney için eğitim verisi için iki yazar, test verisi için bir yazar seçilmiş ve test verisi yazarı sürekli değiştirilmiştir.

Tüm çizelgelerdeki doğruluk değerleri görece birbirine yakın çıkmakla birlikte eğitim ve tahmin süreleri açısından KStar oldukça kötü bir performans sergiliyor. Doğruluk açısından her deneyde MLP, KNN'den biraz daha iyi ama bunun karşılığında MLP'in eğitim süreleri oldukça kötü. Bir kez eğitim yapılıp elde edilen sonuçların tüm tahminlerde kullanıldığı supervised tanıma sistemleri için -ki deneylerde kullanılan kod bu gruba girmektedir- eğitim süresi kritik olmadığı için en uygun seçim MLP'dir. Bu yüzden bu tez için MLP seçilmiştir. Bununla beraber, eğer kod supervised olmayan bir sisteme uyarlanacaksa KNN kullanmak daha doğru olacaktır.

Çizelge 3 Deney 1: Sınıflandırma Yöntemleri Çapraz Karşılaştırma Deneyi

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/ Tahmin Süresi	Doğruluk
1	YPO/YYPO	KStar	ceyhun+eren+cihan (çapraz)	2434ms/ 47332ms	98.40%
2	YPO/YYPO	MLP	ceyhun+eren+cihan (çapraz)	13500ms/ 1000ms	98.40%
3	YPO/YYPO	KNN	ceyhun+eren+cihan (çapraz)	2108ms/ 1262ms	98.46%

Çizelge 4 Deney 2: Sınıflandırma Yöntemleri Eğitim ceyhun+eren Test:cihan

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	KStar	ceyhun+eren/cihan	2066ms/48538ms	92.37%
2	YPO/YYPO	MLP	ceyhun+eren/cihan	14096ms/1128ms	89.74%
3	YPO/YYPO	KNN	ceyhun+eren/cihan	1952ms/1331ms	87.17%

Çizelge 5 Deney 3: Sınıflandırma Yöntemleri Eğitim: ceyhun+cihan Test: eren

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	KStar	ceyhun+cihan/eren	2136ms/46248ms	82.08%
2	YPO/YYPO	MLP	ceyhun+cihan/eren	13345ms/997ms	88.59%
3	YPO/YYPO	KNN	ceyhun+cihan/eren	2164ms/1239ms	86.13%

Çizelge 6 Deney 4: Sınıflandırma Yöntemleri Eğitim: cihan+eren Test:ceyhun

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	KStar	cihan+eren/ceyhun	2213ms/47721ms	81.50%
2	YPO/YYPO	MLP	cihan+eren/ceyhun	13299ms/1017ms	91.64%
3	YPO/YYPO	KNN	cihan+eren/ceyhun	2219ms/1237ms	88.40%

6.2 Nitelik Vektörleri Karşılaştırması

İkinci grup deneyler, tez tarafından önerilen YPO ve YYPO nitelik vektörünün başarısını ölçmeyi amaçlamaktadır. Bu yüzden yalnızca tanıma aşaması teste dahil edilmiş ve bir önceki deney grubunda seçildiği için her deneyde sınıflandırma yöntemi olarak MLP kullanılmış bunun dışında tüm parametrelerde sabit bırakılmıştır.

Çizelge 7'de üç yazardan elde edilen tüm örnekler test verisi, eğitim verisinin yarısı olacak şekilde iki gruba ayrılmıştır. Eğitim ve test verisinin aynı yazara ait olduğu bu durumda YPO/YYPO, 3x3 bitmap nitelik vektöründen biraz daha iyi bir doğruluk değeri veriyor. Bununla birlikte Çizelge 8, Çizelge 9 ve Çizelge 10'da görülebileceği gibi test verisi yabancı bir yazara ait olduğu durumda YPO/YYPO başarısı önemli bir miktarda artmaktadır. Diğer yandan, her bir deneyde YPO/YYPO nitelik vektörünün tahmin süresini yüzde 40 civarı arttırdığı görülebilir. Bununla birlikte, nitelik vektörünü oluşturan her niteliğin aynı gruptan olmasına gerek yoktur. Başarıyı artırmak için gerekli görülen tüm nitelikler nitelik vektörüne eklenebilir. Nitelik vektörünün nitelik sayısı artırılarak doğruluk artırabilir. Ancak her bir nitelik ek bir yük getirerek performansı düşürecektir. Çizelge 11'de YPO/YYPO nitelikleri aynı vektörde birleştirilmiş bu da doğruluğu önemli oranda artırmıştır. Ancak, eğitim ve tahmin sürelerindeki gözle görülür artma yine bu çizelgede görülebilir.

Çizelge 7 Deney 5: N. Vektörleri Karşılaştırma Eğitim: cihan+eren+ceyhun (çapraz)

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	MLP	ceyhun+eren+cihan (çapraz)	13500ms /1000ms	98.40%
2	3x3 Bitmap	MLP	cihan+eren+ceyhun (çapraz)	13479ms /695ms	94.32%
3	Wavelet	MLP	cihan+eren+ceyhun (çapraz)	12479ms /733ms	72.13%

Çizelge 8 Deney 6: N. Vektörleri Karşılaştırma Eğiim: ceyhun+eren Test: cihan

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	MLP	eren+ceyhun/cihan	23157ms /1378ms	89.67%
2	3x3 Bitmap	MLP	eren+ceyhun/cihan	13156ms /761ms	79.34%
3	Wavelet	MLP	eren+ceyhun/cihan	12121ms /831ms	45.99%

Çizelge 9 Deney 7: N. Vektörleri Karşılaştırma Eğitim: ceyhun+cihan Test: eren

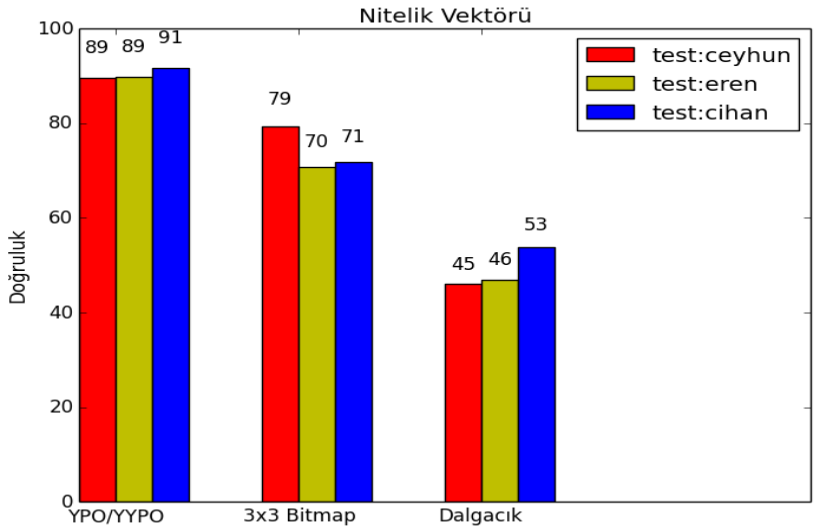
No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	MLP	ceyhun+cihan/eren	14096ms /1128ms	89.74%
2	3x3 Bitmap	MLP	ceyhun+cihan/eren	13156ms /761ms	79.34%
3	Wavelet	MLP	ceyhun+cihan/eren	12121ms /831ms	45.99%

Çizelge 10 Deney 8: N. Vektörleri Karşılaştırma Eğitim: eren+cihanTest: ceyhun

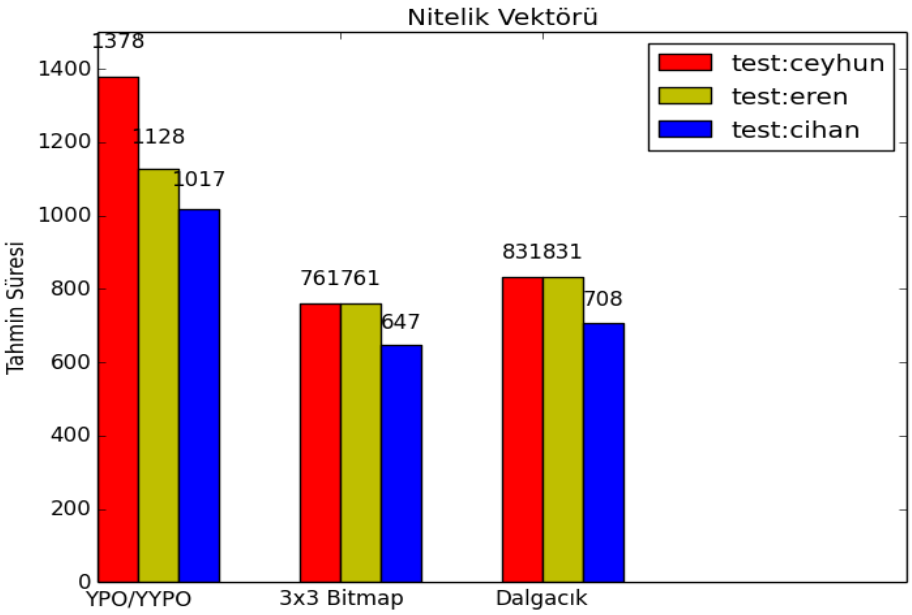
No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO/YYPO	MLP	cihan+eren/ceyhun	13299ms /1017ms	91.64%
2	3x3 Bitmap	MLP	cihan+eren/ceyhun	13266ms /647ms	71.75%
3	Wavelet	MLP	cihan+eren/ceyhun	12290ms /708ms	53.85%

Çizelge 11 Deney 9: N. Vektörleri YPO/YYPO ve 3x3 Bitmap Karma

No	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	Eğitim/Tahmin Süresi	Doğruluk
1	YPO+3x3 Bitmap	MLP	ceyhun+eren+cihan (çapraz)	59980ms /3781ms	99.47%
2	YPO+3x3 Bitmap	MLP	ceyhun+eren/eren	57745ms /4193ms	95.46%
3	YPO+3x3 Bitmap	MLP	ceyhun+cihan/eren	58961ms /3726ms	90.91%
4	YPO+3x3 Bitmap	MLP	cihan+eren/ceyhun	58573ms /3513	92.64



Şekil 23 Nitelik Vektörleri Doğruluk Karşılaştırması



Şekil 24 Nitelik Vektörleri Tahmin Süresi Karşılaştırması

6.3 Tüm Sistem Testi

Bu üçüncü grup deneylerde tüm sistem (Ayrıştırma, Tanıma ve Yapısal Çözümleme) birlikte test edilmektedir. Input.png dosyasının yazarı ceyhun isimli yazardır. Bu nedenle diğer iki yazardan alınan 3027 örnek bu deneylerde eğitim verisi olarak kullanılmış, test verisi ise input.png dosyasından elde edilmiştir. Bu dosya, tanıma sisteminin içerdiği 15 kategoriyi içerecek şekilde üretilmiştir. Bu deney grubunda YPO/YYPO'nun başarısı özellikle öne çıkmaktadır. Her deneyde ayrıştırma ve yapısal çözümleme başarıyla uygulanmıştır. YPO/YYPO'nun farklı sınıflandırma yöntemleriyle kullanıldığı ilk deneyde (Çizelge 11) YPO/YYPO nitelik vektörü her yöntemle başarılı sonuç veriyor. Oysa Çizelge 12, Çizelge 13 ve Çizelge 14 'de farklı sınıflandırma yöntemleriyle yalnızca YPO/YYPO tam olarak başarılı sonuç vermiş diğerleriyse 21 simgenin bir kısmını yanlış tanımıştır.

Çizelge 12 Deney 10: Tüm Sistem Testi – Sınıflandırma Yöntemi Karşılaştırma (YPO/YYPO için)

No	Girdi	N. Vektörü	Sınıflan d.	Yazar(Eğitim/Test)	İşlem Süresi	Başarı
1	Şekil 15a	YPO/YYPO	KStar	cihan+eren/ceyhun	1024ms	100%
			[(10+23)*4]/[[56]/[78]-9]			
2	Şekil 15a	YPO/YYPO	MLP	cihan+eren/ceyhun	170ms	88%
			[(10+*3)*4]/[[56]/[79]-9]			
3	Şekil 15a	YPO/YYPO	KNN	cihan+eren/ceyhun	199ms	94%
			[(10+23)*4]/[[56]/[78]-7]			

Çizelge 13 Deney 11: Tüm Sistem Testi - N. Vektörleri Karşılaştırma (MLP için)

No	Girdi	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	İşlem Süresi	Doğruluk
1	Şekil 15a	YPO/YYPO	MLP	cihan+eren/ceyhun	6ms	88%
			[(10+*3)*4]/[[56]/[79]-9]			
2	Şekil 15a	3x3 Bitmap	MLP	cihan+eren/ceyhun	4ms	70%
			[(10+23)*4]92[[26]1[71]-2]			
3	Şekil 15a	Wavelet	MLP	cihan+eren/ceyhun	10ms	53%
			[)10123)+6](([[56]1[68]19]			
	Şekil 15b	YPO/YYPO	MLP	cihan+ceyhun/eren	3ms	59%
			[(10+2+)*4]/[[76]//9][78]			
	Şekil 15b	3x3 Bitmap	MLP	cihan+ceyhun/eren	7ms	35%
			[(10*2/7*4]1[[2/]1/3][78]			
	Şekil 15b	Wavelet	MLP	cihan+ceyhun/eren	7ms	53%
			[(/0*389*4])[[56]/*8][68]			
	Şekil 15c	YPO/YYPO	MLP	ceyhun+eren/cihan	<1ms	71%
			[(10+93)*4]/[[56]//3][78]			
	Şekil 15c	3x3 Bitmap	MLP	ceyhun+eren/cihan	1ms	59%
			[(/0+23)*4]7[[56]669][5/]			
	Şekil 15c	Wavelet	MLP	ceyhun+eren/cihan	<1ms	47%
			[)10*58(*4]/[[86]/*8][6*]			

Çizelge 14 Deney 12: Tüm Sistem Testi – N. Vektörleri Karşılaştırma (Kstar için)

No	Girdi	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	İşlem Süresi	Doğruluk
1	Şekil 15a	YPO/YYPO	KStar	cihan+eren/ceyhun	23ms	100%
[(10+23)*4]/[[56]/[78]-9]						
2	Şekil 15a	3x3 Bitmap	KStar	cihan+eren/ceyhun	24ms	71%
[(10+63)*4]36[[56]1[70]-5]						
3	Şekil 15a	Wavelet	KStar	cihan+eren/ceyhun	27ms	41%
[)/0128(*4]))[[56])[67](9]						
4	Şekil 15b	YPO/YYPO	KStar	cihan+ceyhun/eren	15ms	59%
[(*0+27)*4]/[[76]//9][78]						
5	Şekil 15b	3x3 Bitmap	KStar	cihan+ceyhun/eren	13ms	47%
[(10*27)*5]2[[56]1/5][08]						
6	Şekil 15b	Wavelet	KStar	cihan+ceyhun/eren	19ms	41%
[)/0*289*4]([[56]/+9][67]						
7	Şekil 15c	YPO/YYPO	KStar	ceyhun+eren/cihan	15ms	77%
[(10+53)*4]/[[56]//8][78]						
8	Şekil 15c	3x3 Bitmap	KStar	ceyhun+eren/cihan	16ms	47%
[(/0+53)*5]7[[56]115][90]						
9	Şekil 15c	Wavelet	KStar	ceyhun+eren/cihan	17ms	47%
[(/0+53)*5]7[[56]115][90]						

Çizelge 15 Deney 13: Tüm Sistem Testi – N. Vektörleri Karşılaştırma (KNN için)

No	Girdi	N. Vektörü	Sınıfland.	Yazar(Eğitim/Test)	İşlem Süresi	Doğruluk
1	Şekil 15a	YPOYYPO	KNN	cihan+eren/ceyhun	8ms	94%
			[(10+23)*4]/[[56]/[78]-7]			
2	Şekil 15a	3x3 Bitmap	KNN	cihan+eren/ceyhun	7ms	76%
			[(10+23)*4]96[[56]1[76]-5]			
3	Şekil 15a	Wavelet	KNN	cihan+eren/ceyhun	8ms	41%
			[)/0128(*4](([[56]([6*](9]			
4	Şekil 15b	YPO/YYPO	KNN	cihan+ceyhun/eren	3ms	58%
			[(10+253*4]/[[76]//9][78]			
5	Şekil 15b	3x3 Bitmap	KNN	cihan+ceyhun/eren	11ms	53%
			[(10*2/)*4]2[[56]1/5][*8]			
6	Şekil 15b	Wavelet	KNN	cihan+ceyhun/eren	13ms	47%
			[(/0*339*4]/[[5*]/+9][6*]			
7	Şekil 15c	YPO/YYPO	KNN	ceyhun+eren/cihan	1ms	71%
			[(10+23)*6]/[[56]//3][78]			
8	Şekil 15c	3x3 Bitmap	KNN	ceyhun+eren/cihan	<1ms	47%
			[(/0+53)*2]7[[56]1/9][)*]			
9	Şekil 15c	Wavelet	KNN	ceyhun+eren/cihan	1ms	47%
			[(10*58/*4]/[[86]/*3][5*]			

7 SONUÇ

Bu tezde, çevrimdışı bir matematiksel ifade tanıma sistemi tasarlayabilmek için İstatistiksel Örüntü Tanıma'nın nasıl kullanılabileceği aşama aşama incelendi. Ayrıca, yetim-piksel-oranı ve yerel yetim piksel oranı (YPO/YYPO) diye isimlendirilen bir yöntem ortaya atıldı. Deneyler bölümünde YPO/YYPO nitelikleri kullanıldığında karakter tanıma doğruluğunun, özellikle sistem hiç tanımadığı bir yazara ait karakterlerle karşılaştığında çok az miktarda düştüğü görüldü. Ek olarak, YPO/YYPO niteliklerinin en uygun oldukları kategoriler için optimum olarak kullanıldığı aşamalı bir sistemde bu doğruluk daha da artıyor ve %96 gibi bir sonuç elde edildi. Önerilen algoritmanın başarısı, farklı sınıflandırma yöntemleriyle ve farklı nitelik vektörleriyle test edildi. Yani YPO nitelik ailesini matematiksel ifade tanıma (veya diğer karakter tanıma) sistemlerinde kullanmanın bir diğer avantajı bu niteliklerin çok hızlı bir şekilde çıkarılabilmesi. Her pikselin üzerinden yalnız bir kez geçilerek (O(n) karmaşıklığı) gerekli tüm YPO nitelikleri bir kerede üretilebilir.

YPO/YYPO simgelerin kategorilerinin farklı olma olasılığının geometrik farklılıklarıyla ilişkili olduğunu varsaymaktadır. Günümüzde kullanılan çoğu simge için bu geçerlidir. Bu nedenle, YPO/YYPO nitelik vektörü kullanmak, çoğu simgenin tanınma doğruluğunu artırabilir. Ama, bu varsayımın geçerli olmadığı durumlarda nitelik vektörüne YPO/YYPO'ya yardımcı ek nitelikler eklemek gerekir. Örneğin 'C' ve 'Ç' simgeleri arasında geometrik olarak küçük bir fark vardır. Bu fark 't' simgesinin altı kıvrık olan ve olmayan versiyonlarından daha fazladır. Bu sorun nedeniyle, tezde '1' ve '-' kategorilerinin geometrik benzerliğinden ötürü YPO/YYPO niteliklerine genişlik-yükseklik oranı niteliği eklendi. Bununla beraber, nitelik vektörü için Temel Element Analizi kullanılması bu tür ayrıntısal farklarda daha başarılı olabilir (Bu teknik için [23]'e bakılablir). Tabi, bu önemli bir performans mailyeti getirecektir. Ayrıca YPO/YYPO'nun kullandığı topoloji açısından birbirleri ile özdeş veya benzer olan kategorileri ayırt etmekte YPO/YYPO tüm diğer nitelik vektörlerinin gerisine düşebilir. Örneğin 'D' ve 'O' simgeleri YPO açısında özdeş YYPO açısından ise oldukça benzerdir ($YPO_{sol}[0,25,0,100]$ sol çeyrek için sol-yetim-oranı kısmen bu iki simgeyi ayırt edebilir). Bu yüzden, YPO/YYPO'nun etkisiz kalacağı kategoriler kategori

listesinde ise YPO/YYPO niteliklerini de içeren karma bir nitelik vektörü daha sağlıklı sonuç verecektir.

Ayrıca, YPO/YYPO simgelerin kaba geometrik özelliklerini kullandığından bu tezde tanımlandığı şekliyle, Çin veya Hint alfabesi karakterleri gibi aşırı karmaşık karakterlerde başarısız olabilir (Hint alfabesinin tanınması için [24] incelenebilir. Bununla beraber Ayrıştırma aşaması değiştirildiğinde Arap Alfabesi için kullanılabilirler. Bununla beraber, YPO/YYPO latin alfabesi, yunan alfabesi ve özel matematiksel simgeleri içeren MİT'in kullanabileceği çok daha geniş bir simge kümesi için özellikle uygundur. Böyle bir yapıda Simge Ayrıştırma aynı şekilde kullanılabilir. Yapısal Çözümleme ise kök gibi bazı özel işlemler ne yatay ne dikey olarak ayırılabildiği için bu iki işlem işe yaramadığı durumlar için genişletilmelidir.

KAYNAKLAR LİSTESİ

[1] Zanibbi, R., Blostein, D., *Recognition and retrieval of mathematical expressions* International Journal of Document Analysis and Recognition, DOI 10.1007/s10032-011-0174-4, 2010

[2] Internet: Tessaract (software), 2013, In Wikipedia. http://en.wikipedia.org/wiki/tessaract_(software)

[3] Internet: Tessaract-ocr, 2013, In Google Code. https://code.google.com/p/tesseract-ocr/

[4] Internet: The Freehand Formula Entry System 2007, In Rochester Institute of Technology. http://www.cs.rit.edu/~rlaz/ffes/

[5] Internet: Infty Reader, 2013, In Infty Reader Inc. http://www.inftyreader.org/

[6] Fink, G. A., Plötz, T., Developing Pattern Recognition System Based on Markov Models: The ESMERALDA Framework, Pattern Recognition and Image Analysis. DOI 10.1134/S1054661808020041, vol. 18, no. 2, 2008

[7] Internet: Esmeralda Project, 2013. In Sourceforge. http://sourceforge.net/projects/esmeralda/

[8] Chan, K.F., Yeung, D.Y., Mathematical expression recognition: a survey, International Journal of Document Analysis and Recognition. IJDAR (2000) 3: 3–15, 2000

[9] Garain, U., Chaudhuri, B. B., *Recognition of Online Handwritten Mathematical Expressions*, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. PART B: CYBERNETICS, VOL. 34, NO. 6, DECEMBER, 2004

[10] Theodoridis, S., Koutroumbas, K., *Pattern Recognition*, Academic Press Elsevier, 4. Edition 2009

[11] Jain, A. K., Duin, R. P. W., Mao, J., *Statistical Pattern Recognition: A Review*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 22, NO. 1, 2000

[12] Mesnier, E., *Naive Bayes Classifier Example* Lecture Notes, 2013

[13] Bernard, M., Fromond, *E., Habbard, A., Sebban, M., Handwritten Digit Recognition using Edit Distance-Based KNN*, Teaching Machine Learning Workshop, Edinburgh, Scotland : United Kingdom, 2012

[14] Cleary, J. G., Trigg, L. E., *K*: An Instance-based Learner Using an Entropic Distance Measure*, Dept. of Computer Science, University of Waikato, New Zealand., 1995

[15] Trier, O. D., Jain, A. K., Taxt, T., Feature Extraction Methods For Character Recognition A Survey, Pattern Recognition Vol. 29, No. 4 Pp. 641-662 1996

[16] Bhattacharya, U., Shridhar, M., Parui, S. K., Sen, P. K., Chaudhuri, B. B., Offline recognition of handwritten Bangla characters: an efficient two-stage approach, Pattern Anal Applic (2012) 15:445–458, DOI 10.1007/s10044-012-0278-6, 2012

[17] Riedmiller, M., Multi Layer Perceptron, Machine Learning Lab Special Lecture, University of Freiburg, 2014

[18] Yılmaz, D., *Dizi İfadelerine Dayanan Yeni Bir Optiksel Karakter Tanıma Algoritması* Gazi Üniversitesi Fen Bilimleri Eğitimi, Elektronik-Bilgisayar Eğitim Yüksek Lisans Eğitimi, 1999

[19] Connell, S. S, Jain, A. K., *Template-based online character recognition*, Journal of Pattern Recognition Society, Pattern Recognition 34 (2001) 1 - 14, 2001

[20] Chen, Z., Wei, Z., *A Handwriting Numeral Character Recognition System* IEEE Multimedia Technology (ICMT), 2010 International Conference on 10.1109/ICMULT.2010.5631304 2010

[21] Aoshima, S., Suzuki,T., Mori, K., Suenaga, Y., *Real-Time Recognition of Handwritten Math Formulas* Electronics and Communications in Japan, Part 3, Vol. 87, No. 8, 2004

[22] Xiangwei, Q., Weimin, P., Wang Yang, Y., The study of structral analysis strategy in handwritten recognition of general mathematical expresion, International Forum on Information Technology and Applications, IEEE, DOI 10.1109/IFITA.2009.169, 2009

[23] Ahmadi T., Jameel, A., Ahmad, B., Pattern Recognition using Statistical and Neural Techniques, Computer Network and Information Technology (ICCNIT), 2011 International Conference, IEEE, p87-91, DOI 10.1109/ICCNIT.2011.6020913, 2011

[24] Bhattacharya, U., Shridhar, M., Parui, S. K., Sen, P. K., Chaudhuri, B. B., Offline recognition of handwritten Bangla characters: an efficient two-stage approach, Pattern Anal Applic (2012) 15:445-448, DOI 10.1007/s10044-012-0278-6, 2012

EK 1

YPO/YYPO Nitelik Vektörü Çıkarımı Java Kodu

```
public class FeatureYPO extends AnyFeature {

   private static final long serialVersionUID = -8609206855541248172L;

   @Override
 public double[] extract(ISymbol symbol) {
        String symbol_str = symbol.toString();
        int total = 0;
        int not_orphan_matched = 0;
        int orphan_right_matched = 0;
        int orphan_left_matched = 0;
        int orphan_right_for_bottom_half_matched = 0;
        int orphan_left_for_bottom_quarter = 0;
        int not_orphan_for_bottom_half = 0;

        String[] transposed = transpose(symbol_str).split("\n");
        String[] straight = symbol_str.split("\n");
        double width_over_height = ((double)straight.length)/transposed.length;

        for (int i=0; i<straight.length; ++i) {
             String row = straight[i];
            for (int j=0; j<transposed.length; ++j) {
                String col = transposed[j];

                if (row.charAt(j) == '0') {
                    String left = row.substring(0, j);
                    String right = row.substring(j, transposed.length);
                    String top = col.substring(0, i);
                   String bottom = col.substring(i, straight.length);

                   if (left.indexOf('1') != -1 && right.indexOf('1') != -1
                           && top.indexOf('1') != -1 && bottom.indexOf('1') != -1) {
```

```
                    ++not_orphan_matched;
                    if (i>=(straight.length/2)) {
                        ++not_orphan_for_bottom_half;

                    }
                } else if (left.indexOf('1') != -1 && right.indexOf('1') == -1
                        && top.indexOf('1') != -1 && bottom.indexOf('1') != -1) {
                    ++orphan_right_matched;
                    if (i>=(straight.length/2)) {
                        ++orphan_right_for_bottom_half_matched;
                    }
                } else if (left.indexOf('1') == -1 && right.indexOf('1') != -1
                        && top.indexOf('1') != -1 && bottom.indexOf('1') != -1) {
                    ++orphan_left_matched;
                    if (i>=(3*straight.length/4)) {
                        ++orphan_left_for_bottom_quarter;
                    }
                }

                ++total;
            }
        }
    }

    double[] features = new double[7];
    features[0] = total == 0 ? 0.0 : (double)not_orphan_matched/total;
    features[1] = total == 0 ? 0.0 : (double)orphan_right_matched/total;
    features[2] = total == 0 ? 0.0 : (double)orphan_left_matched/total;
    features[3] = total == 0 ? 0.0 : (double)orphan_right_for_bottom_half_matched/total;
    features[4] = total == 0 ? 0.0 : (double)orphan_left_for_bottom_quarter/total;
    features[5] = total == 0 ? 0.0 : (double)not_orphan_for_bottom_half/total;
    features[6] = width_over_height;
    return features;
```

```
    }

    @Override
    public String getName() {
        return "YPO";
    }

    @Override
    public int numberOfSubFeatures() {
        return 7;
    }
}
```