İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SEYREK İŞARET İŞLEMEDE SINIFLANDIRMA UYGULAMALARI VE ÇEKİRDEK TABANLI YAKLAŞIMLAR

YÜKSEK LİSANS TEZİ

Abdurrahman YEŞİLOĞLU

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Programı

ARALIK 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SEYREK İŞARET İŞLEMEDE SINIFLANDIRMA UYGULAMALARI VE ÇEKİRDEK TABANLI YAKLAŞIMLAR

YÜKSEK LİSANS TEZİ

Abdurrahman YEŞİLOĞLU
(504121303)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Ender Mete EKŞİOĞLU

ARALIK 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 504121303 numaralı Yüksek Lisans Öğrencisi Abdurrahman YEŞİLOĞLU, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "SEYREK İŞARET İŞLEMEDE SINIFLANDIRMA UYGULAMALARI VE ÇEKİRDEK TABANLI YAKLAŞIMLAR" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Doç. Dr. Ender Mete EKŞİOĞLU** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Selçuk PAKER** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Ahmet Korhan TANÇ Kırklareli Üniversitesi	.....

Teslim Tarihi : 26 Kasım 2015
Savunma Tarihi : 23 Aralık 2015

Eşime ve aileme,

ÖNSÖZ

Yüksek lisans tez çalışmamın tüm adımlarında öneri ve görüşlerini esirgemeyen değerli danışman hocam Sayın Doç. Dr. Ender Mete Ekşioğlu'na ve bu süreçte desteklerini hiçbir zaman eksik etmeyen değerli eşim Kübra Özmal Yeşiloğlu ve aileme teşekkürlerimi sunarım.

Kasım 2015 Abdurrahman Yeşiloğlu
(Elektrik-Elektronik Mühendisi - Bilgisayar Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

SVM	**:** Support Vector Machines
NN	**:** Nearest Neighbor
SRC	**:** Sparse Representation Classification
K-SRC	**:** Kernel Sparse Representation Classification
KSR	**:** Kernel Sparse Representation
MLP	**:** Multi-Layer Perceptron
MP	**:** Matching Pursuit
OMP	**:** Orthogonal Matching Pursuit
MOD	**:** Method of Optimal Directions
K-SVD	**:** K- times Singular Value Decomposition
CS	**:** Compressive Sensing
PCA	**:** Principal Component Analysis
ICA	**:** Independent Component Analysis
NP Hard	**:** Non-Deterministic Polynomial-Time Hard

SEMBOLLER

Φ	**:** Doğrusal Olmayan Dönüşüm Fonksiyonu
$\boldsymbol{D}$	**:** Sözlük
$\boldsymbol{a}$	**:** Seyreklik Katsayısı
$\boldsymbol{x}$	**:** Test Objesi
$\boldsymbol{T_0}$	**:** Seyreklik Seviyesi
$\boldsymbol{k}$	**:** NN Algoritması İçin Komşuluk Sayısı
$\boldsymbol{g}$	**:** Gürültü Terimi
$\boldsymbol{\epsilon}$	**:** Hata Eşik Değeri
$\boldsymbol{S}$	**:** Doğrusal Olmayan Uzayda Sözlüğün İç Çarpımı
$\boldsymbol{r}$	**:** Artık
$\boldsymbol{\delta_i}$	: i. sınıfa Ait Elemanları Seçen Fonksiyon

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SEYREK İŞARET İŞLEMEDE SINIFLANDIRMA UYGULAMALARI VE ÇEKİRDEK TABANLI YAKLAŞIMLAR

ÖZET

Seyreklik tabanlı işaret işleme araştırmacılar tarafından oldukça fazla ilgi çeken ve örüntü tanıma, görüntü işleme ve bilgisayar görmesi gibi alanlarda sıklıkla kullanılan bir teoridir. Seyrek işaret işleme, işaret ve görüntü işlemenin birçok disiplininde teorik ve pratik olarak gittikçe artan bir popülariteye sahip olmuştur. Günümüzde de bu alandaki çalışmalar aktif olarak devam etmektedir.

Seyrek işaret işlemenin temeli son yıllarda ortaya çıkmış sıkıştırılmış algılama teorisine dayanmaktadır. Shannon örnekleme teoremi ancak bir sinyalin en büyük frekansının iki katı ile örneklenmesi durumunda bu örnekler ile tekrar oluşturulabileceğini konu alır. Fakat bu örnekler birbiri arasında büyük bir ilintiye ve yedekliliğe sahiptir. İşte bu noktada sıkıştırılmış algılama bir sinyalin mümkün olduğu kadar seyrek ve sıkıştırılabilir temsil edilebilmesi durumunda sadece bir kaç değer ile tekrar oluşturulabileceğini konu alır. Sıkıştırılmış algılama bu özelliği nedeniyle literatürde yeni bir örnekleme teoremi olarak gösterilmektedir.

Sıkıştırılmış algılama 3 temel adımdan oluşmaktadır. Bunlar seyrek temsil, kodlama ve yeniden oluşturma adımlarıdır. Bu adımlardan ilki olan seyrek temsil, büyük boyuttaki veriyi örneklerken geleneksel örnekleme ve sıkıştırma adımlarını birleştirerek Shannon örnekleme teoreminden daha etkin örneklemeye imkân sağlamaktadır.

Seyrek işaret işleme, bu özellikleri ile sınıflandırma uygulamalarında da oldukça fazla dikkat çekmiştir. Bu tezde de seyrek işaret işlemede sınıflandırma uygulamaları ile doğrusal olmayan sınıflandırma için kullanılan çekirdek tabanlı yaklaşımlar ele alınmıştır.

Seyrek işaret işlemede gözetimli sınıflandırma, sınıfları bilinen eğitim seti ile test nesnesinin seyrek gösterilmesi ile gerçeklenir. Bu problem eğitim setindeki eleman sayısının test nesnesi boyutundan çok büyük olması nedeniyle eksik belirtilmiş doğrusal bir optimizasyon problemidir. Seyrek vektör l_0 optimizasyon probleminin çözümü ile bulunur. Bu problem eşleştirmeli takip ve dikey eşleştirmeli takip gibi açgözlü algoritmalarla çözülebileceği gibi problem l_1 veya l_2 normlu optimizasyon problemlerine dönüştürülerek de yaklaşık çözüm elde edilebilir.

Optimizasyon probleminin çözümünden sonra test nesnesi, elde edilen seyrek vektör ile her bir sınıfa ait eğitim seti kullanılarak temsili olarak tekrar oluşturulur. Her bir sınıfa ait temsili nesne ile orjinal test nesnesi arasındaki fark hesaplanır. Hangi sınıf ile yapılan temsil daha az fark oluşturuyor ise test nesnesinin o sınıfa ait olduğuna karar verilir. Bu yaklaşım tezde doğrusal sınıflandırma yapan seyreklik tabanlı sınıflandırma olarak ele alınmıştır.

Doğrusal olmayan sınıflandırma için giriş uzayında doğrusal olarak ayrılmayan iki sınıf, bir doğrusal olmayan fonksiyon ile yüksek boyutlu öznitelik uzayına çevrilir. Fakat bu doğrusal olmayan dönüşüm oldukça büyük bir işlemsel yük ve zaman gerektireceğinden dönüşüm SVM'de kullanılan çekirdek fonksiyonları ile yapılır. Seyreklik tabanlı sınıflandırmada doğrusal olmayan ayrım, bu doğrusal olmayan öznitelik boyutuna haritalanmış test nesnesini yine doğrusal olmayan öznitelik uzayına haritalanmış eğitim seti ile doğrusal olarak temsil edebilmektir. Bu dönüşüm uygun bir çekirdek yaklaşımı ile yapılabilmektedir. Çekirdek fonksiyonları doğrusal, Gauss ve polinom gibi fonksiyonlar olabilmektedir. Her uygulama için en iyi başarımı veren çekirdek fonksiyonu ve parametreleri farklılık göstermektedir. Bu yaklaşım tezde seyreklik tabanlı sınıflandırmada çekirdek yaklaşımı olarak ele alınmıştır.

Teze konu yaklaşımda doğrusal sınıflandırma yapan seyreklik tabanlı sınıflandırma ve doğrusal olmayan sınıflandırma yapan seyreklik tabanlı sınıflandırmada çekirdek yaklaşımı yukarıdaki bilgiler ışığında irdelenmiştir. Her iki sınıflandırıcı için yüz tanıma ve rakam tanıma uygulamalarındaki benzetimleri MATLAB ortamında gerçekleştirilmiştir. Benzetimlerde elde edilen sonuçlar sınıflandırma sistemlerinde oldukça bilinen doğrusal destek vektör makineleri ve en yakın komşuluk sınıflandırıcıları ile karşılaştırılmıştır.

CLASSIFICATION APPLICATIONS OF SPARSE SIGNAL PROCESSING AND KERNEL BASED METHODS

SUMMARY

Sparse signal processing has attracted much consideration from scientists in field of pattern recognition, image processing and computer vision. Sparse signal processing has most important theoretical and practical applications in many disciplines of signal and image processing. In this field of research still continues especially in the field of signal processing and applied mathematics.

The basis of sparse signal processing is related to compressed sensing (CS) theory that has emerged in recent years. As Shannon's sampling theorem, if the sampling frequency is greater than twice the frequency of the signal, then the signal can be reconstructed perfectly. However, these samples have a great relevance and redundancy between each other. At this point, if a signal can be represented as sparse or compressible on appropriate basis elements, the original signal can be reconstructed by exploiting a few measured values. Generally, these measured values are much less than Shannon's sampling theorem. Thanks to this feature, compressed sensing is shown as a new sampling theorem in the literature.

Compressed sensing includes the three basic steps, sparse representation, encoding measuring, and reconstructing. The efficient sampling from mass data is a big problem for sampling. Sparse representation step can overcome this difficulty of signal sampling. Also, sparse signal processing integrates the steps of conventional sampling and compression in one step.

Sparse signal processing has attracted in much consideration in classification problem. For this thesis, sparsity based linear classification and kernel approaches for the non-linear classification have been conducted to have the knowledge of the sparse classification application.

It is very critical that the classification applications need to have high performance and reliable solutions. Classification can be divided 3 different categories, type of learning, assumption on data distribution and number of outputs. In this thesis, supervised, hard and non-parametric classifiers are discussed. It is important that test object should be represented on training set with enough sparsity level for the classification with sparse signal processing. Due to this, training set can be selected pre-determined known mathematical transformation such as curvelet, wavelet. Instead of using pre-determined training set, the data set of the samples can be used directly or can be learned a training set from the data.

For the sparsity based classification, firstly sparse vector must be obtained using training samples. According to sparse representation, the test object must be linearly represented by all training samples. Therefore, for obtaining the sparse vector, training samples are required to be quite large than the test object features. Finally, the sparse vector can be obtained from the results of the under-determined linear optimization problem.

This linear equation is a non-deterministic polynomial-time hard and no known algorithm for finding the exact sparsest solution. Instead of finding exact solution, approximate solution can be obtained with a few approximation algorithms.

Sparse vector can be found the result of l_0 optimization problem. It denotes l_0 norm which counts the number of non-zero entries in a sparse vector. This problem can be solved with greedy approximation algorithms such as Matching Pursuit or Orthogonal Matching Pursuit. In addition, this problem can be solved by transforming to l_1 or l_2 optimization problem. For all methods, sparsity level plays an important role in sparse representation.

After solution of optimization problem, test object is reconstructed again representatively using the sparse vector obtained from training set of each classes. The difference between the representative object of each class and orginal test object called residual. The test object is assigned to the class, which has the minimum residual value. This approach is called as sparsity based linear classification method.

Classification with linear decision boundries is not feasible for many applications. Thus, we need non-linear classification in applications. Support Vector Machines (SVM) is one of the non-linear classification methods. SVM proposes mapping of input space into high dimensional feature space using kernel approaches and using linear classification in this space. This approach can be applied to sparsity based classification methods.

For non-linear classification, two classes which may not separated linearly in the input space, can be mapped into high dimensional feature space using non-linear function. Since non-linear mapping requires considerably high computational load and time, kernel functions of SVM can be used to reduce this computational load.

In sparsity based classification, non-linear separation means linear representation of the test object mapped into feature space with training set mapped into feature space. This mapping can be done with appropriate kernel approach. Kernel approaches can be linear, Gaussian and polynomial functions. The best performance can be gain through different kernel functions and parameters in diffirent applications.

It this thesis, linear sparsity based classification and kernel approach in non-linear sparsity based classification is studied. Face recognition and digit recognition applications are implemented in MATLAB environment for this two classifiers. The results are compared with SVM and Nearest Neighbour (NN) algorithms.

Extended Yale B database is used for face recoginition application. Some of the problems faced in face recognition are light, high dimensional input space, dictionary size and noise. In this work, tests are performed with face images, which has different light, input dimension, dictionary size and noise. To calculate average performance, one-half of images of a person are selected for training and one-half of it for test. The experimental results show that the sparsity level should be 5 for the best performance. Also Gaussian and polynomial kernel functions are compared and it is shown that the polynomial functions that has 0.6 degree gives the best performance. These classifiers are compared with SVM and 3-NN classifiers and it is shown that both in noise free images and images with %10-%70 noise, the sparsity based classifiers gives the best performance. Also both in low and high feature dimension, sparsity based classifiers outperformed other classifers. Especially kernel based approach gives better results than linear sparsity classifier in noise free images and different dictionary size.

Handwritten USPS database is used for digit recognition. 350 training images and 350 test images are used for every digit. Tests are performed for noiseless and noisy images. Level of sparsity is determined as 5 and 4.degree polynomial kernel function is used in experiments. These classifiers are compared with linear SVM and 1-NN classifiers. In noiseless and low noisy levels sparsity based classifers give better results. It is shown that in noiseless case, recognition rates of sparse classifiers are not affected by the dimension of the feature and dictionary size. Kernel based approach outperformed other linear sparsity based classifiers in noise free and low noisy levels.

When we compare the simulation results obtained in face and digit recognition applications, sparsity based classifiers have achieved quite high and consistent performance for different feture and noise levels for face recognition. However for the digit recognition application, sparsity based classifiers show quite consistent and good success in noiseless case, but they do not give the expected performance in noisy situations. Consequently, SRC and K-SRC algorithms may be used as a good method in face recogintion applications and also for digit recognition applications in noiseless case.

1. GİRİŞ

Teknolojinin hızla ilerlediği son günlerde bilimsel çalışmalara ve sektörün ihtiyaçlarına uygun olarak sınıflandırma sistemlerine büyük bir ilgi duyulmaktadır. Özellikle kamu güvenliği anlamında büyük talep gören parmak izi ve yüz tanıma sistemleri, terör saldırıları olmak üzere birçok güvenlik durumları için kullanılmaktadır [1].

Yukarıdaki durumlar dışında da birçok uygulama sahası bulunan sınıflandırma sistemleri otomatik ve başarılı sonuç veren yaklaşımlara ihtiyaç duymaktadır. Bu yaklaşımlar genel olarak oldukça karmaşık yapılara sahiptir. Bu nedenle sınıflandırmadaki genel amaç düşük karmaşıklığa sahip tutarlı ve uygulanabilir sistemler geliştirmektir.

Bu uygulamalarda kullanmak amacıyla literatürde birçok sınıflandırma algoritması geliştirilmiştir. Bu algoritmaların en bilinenleri karar ağaçları [2], yapay sinir ağları [3], en yakın komşuluk [4] ve destek vektör makineleridir [5]. Birçok sınıflandırma uygulaması bu yöntemleri kullanırken matematik alanındaki yeni gelişmeler yeni sınıflandırma metotlarını ortaya çıkarmıştır. Bunlardan biride seyrek işaret işleme temelli sınıflandırma yaklaşımıdır. Seyrek işaret işleme, sıkıştırılmış algılama [6] ile ortaya çıkmış ve bir sinyalin belirli bir baz kümesi ile seyrek olarak gösterilebileceğini temel alan bir teoridir. Bu teorinin geleneksel sıkıştırma ve kodlama işlemlerine göre daha iyi performans göstermesi özellikle bilgisayar görmesi, görüntü işleme, örüntü tanıma ve makine öğrenmesi alanlarında birçok başarılı uygulamalara sahip olmasını sağlamıştır. Daha ayrıntılı olarak bakmak gerekirse seyreklik tabanlı işaret işleme, gürültü giderme, çözünürlük yükseltme, görüntü restorasyonu, kalite değerlendirmesi, sınıflandırma, bölüntüleme, işaret işleme, biyometri ve diğer birçok yapay zekâ sistemlerinde oldukça başarılı uygulamalara sahip olmuştur.

Seyreklik tabanlı işaret işleme üzerine yapılan ilk çalışmalar sıkıştırılmış algılama teorisine dayanan, bir sinyali örnekleme veya sıkıştırma çalışmalarıdır. Fakat yapılan

son çalışmalar seyrekliğin sadece işaret işleme için değil sınıflandırma içinde oldukça geçerli olanaklar sunduğunu göstermiştir. Seyrekliğin doğasındaki bu sınıflandırma yeteneği birçok araştırmacının ilgisini çekmiş ve değişik alanlarda birçok uygulama geliştirilmiştir.

Seyreklik tabanlı sınıflandırma alanında yapılan ilk çalışma Wright ve diğerleri tarafından önerilmiştir [7]. Bu çalışma temelde seyreklik tabanlı yüz tanıma sistemini konu almıştır. Çalışmanın ana hedefi önceden tanımlı yüz sınıfları kullanılarak yeni gelen bir yüz resmini bu sınıflardan birine atamaktır. Önerilen sınıflandırıcının ilk varsayımı, bilinen eğitim örneklerinin doğrusal birleşimi ile test resminin yeterli miktarda seyrek olarak gösterilebileceğidir. Bu varsayımla önerilen çalışma, sözlüğün doğrusal birleşimi ile test nesnesini seyrek olarak temsil etmiş ve test resmi ile temsil resmi arasındaki hatayı minimum yapan sınıfa karar verecek şekilde tasarlanmıştır. Bu çalışmanın deneysel sonuçları oldukça rekabetçi başarımlar verirken, özellikle öznitelik seçiminin seyreklik tabanlı sınıflandırma algoritması için çok kritik bir rol üstlenmediğini göstermiştir.

Seyreklik tabanlı sınıflandırma temelde doğrusal karar sınırları ile sınıflandırma yapan bir sınıflandırıcıdır. Fakat birçok problemin doğası gereği doğrusal sınıflandırma mümkün değildir. Bu nedenle doğrusal olmayan sınıflandırıcılara ihtiyaç duyulmuştur. Destek vektör makineleri (Support Vector Machine-SVM) temelinde ilk doğrusal olmayan sınıflandırıcı yaklaşımı Boser ve diğerleri tarafından önerilmiştir [8]. Bu çalışmada doğrusal olarak sınıflandırılamayan giriş uzayı, çekirdek fonksiyonları ile doğrusal ayrım yapılabilecek yüksek boyutlu bir öznitelik uzayına dönüştürülmüştür. Seyreklik temelli ilk çekirdek yaklaşımları ise l_1 normlu SVM [9] ve çekirdek tabanlı seyrek temsil (Kernel Sparse Representation-KSR) [10] çalışmalarıdır. Özellikle KSR, teze konu çekirdek yaklaşımı ile oldukça ilintilidir.

Bu bağlamda teze konu yaklaşım sıkıştırılmış algılama ile birlikte son dönemlerde birçok çalışmaya konu olmuş seyreklik tabanlı sınıflandırma yaklaşımıdır. Tezin yazılmasındaki genel amaç seyreklik tabanlı işaret işlemenin sınıflandırmadaki başarısını irdelemek ve çekirdek yaklaşımının sınıflandırmaya olan başarım etkisini gözlemlemektir. Bu amaçla yüz tanıma ve rakam tanıma uygulamaları için bilgisayar benzetimleri gerçekleştirilmiş, deneysel sonuçlar destek vektör makineleri ve en yakın komşuluk algoritmaları ile karşılaştırılmıştır.

Yukarıda ele aldığımız amaç çerçevesinde tezin ana kurgusu aşağıdaki gibi olacaktır.

- 2. Bölüm, sınıflandırma probleminin tanımı, adımları ve bilinen yaklaşım algoritmaları anlatılmıştır.
- 3. Bölüm, seyreklik tabanlı işaret işleme ve takip algoritmaları anlatılmıştır.
- 4. Bölüm, seyreklik tabanlı sınıflandırma algoritması ve seyreklik tabanlı sınıflandırmada çekirdek yaklaşımı anlatılmıştır.
- 5. Bölüm, Yale B yüz ve USPS rakam veri setleri üzerinde seyreklik tabanlı sınıflandırıcıların diğer algoritmalarla karşılaştırmalı benzetim sonuçları anlatılmıştır.
- 6. Bölüm, elde edilen sonuçlar ile teze konu yaklaşımın başarım değerlendirmesi yapılmış ve gelecek çalışmalar anlatılmıştır.

2. SINIFLANDIRMA PROBLEMİ

2.1 Makine Öğrenmesi

Bir problem bilgisayar ortamında çözülmek istendiğinde problemi çözecek bir algoritma geliştirilmelidir. Bu duruma örnek olarak sayıları sıralayan bir algoritma verilebilir. Bu problemin tanımlanması ve gerekli algoritmanın geliştirilmesi oldukça kolaydır. Fakat bazı problemlerde gerekli algoritmayı tanımlamak oldukça zordur. Bu duruma örnek olarak istenmeyen e-postaları engelleme uygulamasını verebiliriz. Bu problem için giriş, e-postaları oluşturan karakterler, çıkış ise bu e-postanın istenmeyen e-posta olup olmadığıdır. Fakat her bir birey için hangi e-postanın istenen ya da istenmeyen olduğunu bilmek mümkün değildir. Fakat istenen veya istenmeyen e-posta olup olmadığı bilinen yüzlerce örnek mevcutsa, istenmeyen e-postalar bu örneklerle karşılaştırma yapılarak bulunabilir. Diğer bir deyişle bilgisayarın (makinenin) bu işi yapacak algoritmayı otomatik olarak oluşturması (öğrenmesi) sağlanabilir.

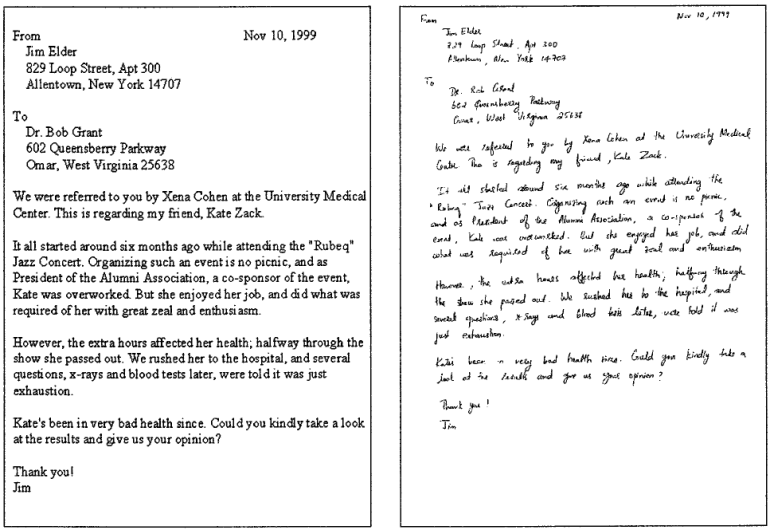
Örnek olarak, bir süpermarket şubesinin satış eğilimi, bir kitaba olabilecek talep, bir web sitesinin ziyaretçi sayısı ya da bir kişinin ziyaret etmek istediği bir şehri kesin olarak bilmek mümkün olmasa da gerekli veriler toplandığı durumda bu soruların cevapları bilgisayar ve makine öğrenmesi ile büyük doğruluk oranı ile ortaya çıkartılabilir.

Bu bağlamda makine öğrenmesi, veri madenciliği ve yapay zekânın bütüncül bir yaklaşımı olarak ortaya çıkmaktadır. Günümüzde makine öğrenmesi üzerine yapılan birçok çalışma bulmak mümkündür. Özellikle örüntü tanıma, ses tanıma, yüz tanıma ve robotik bunlardan bazılarıdır [11].

2.2 Örüntü Tanıma

Makine öğrenmesinin önemli uygulama alanlarından biri örüntü tanımadır. Örüntü tanıma aralarında belirli bir ilişki kurulabilen sinyalleri veya nesneleri önceden belirlenmiş bir karar mekanizması ile tanımlayan veya sınıflandıran alanın genel

adıdır. Karakter tanıma, parmak izi tanıma, yüz tanıma, ses tanıma ve tıbbi görüntüleme örüntü tanımanın yoğun olarak kullanıldığı uygulama alanlarıdır (Şekil 2.1).



From Nov 10, 1999
Jim Elder
829 Loop Street, Apt 300
Allentown, New York 14707

To
Dr. Bob Grant
602 Queensberry Parkway
Omar, West Virginia 25638

We were referred to you by Xena Cohen at the University Medical Center. This is regarding my friend, Kate Zack.

It all started around six months ago while attending the "Rubeq" Jazz Concert. Organizing such an event is no picnic, and as President of the Alumni Association, a co-sponsor of the event, Kate was overworked. But she enjoyed her job, and did what was required of her with great zeal and enthusiasm.

However, the extra hours affected her health; halfway through the show she passed out. We rushed her to the hospital, and several questions, x-rays and blood tests later, were told it was just exhaustion.

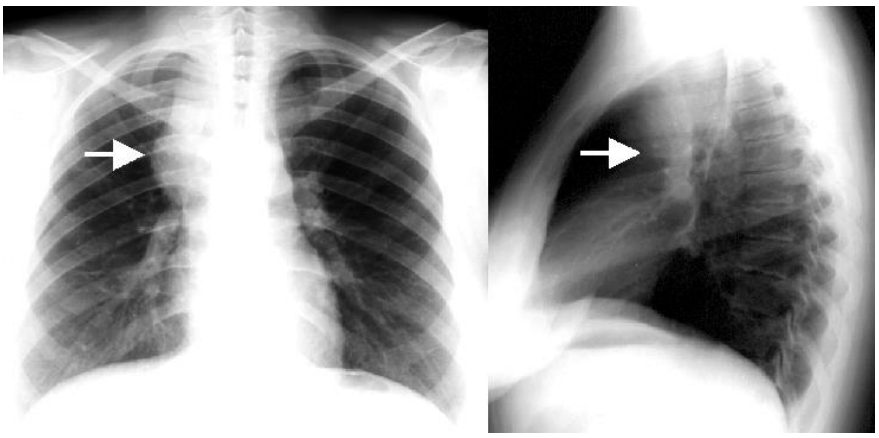
Kate's been in very bad health since. Could you kindly take a look at the results and give us your opinion?

Thank you!
Jim

Şekil 2.1 : Sağdaki karakter tanıması yapılacak el yazması metin, soldaki karakter tanıma sisteminin çıkışı [12].

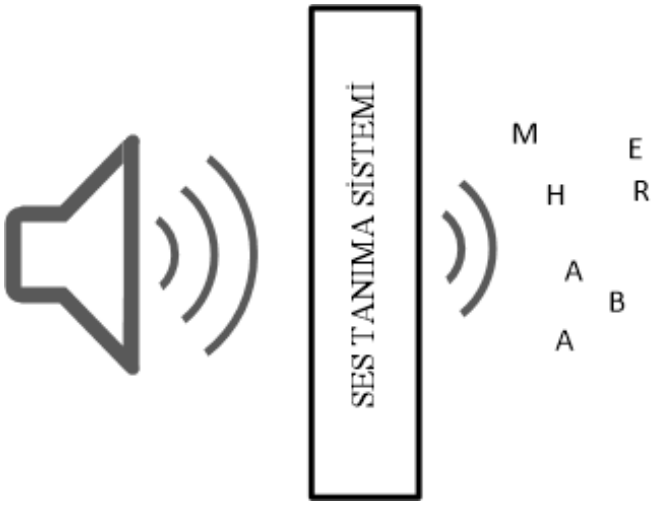
Örüntü tanımanın en önemli uygulama alanlarından biri yüz tanımadır. Yüz tanıma, kimliği belirsiz kişinin, kimliği belirli kişiler kümesi içerisinden bir bireye atama işlemidir. Yüz tanıma özellikle renkli görüntü ve değişik yüz varyasyonlarının olduğu durumlarda oldukça zorlu bir uygulamadır.

Sağlık alanında kullanılan tanılama sistemleri de örüntü tanıma için önemli bir çalışma sahasıdır. Bu alanda yapılan çalışmaların genel amacı tıbbi görüntüleme ile hastalık teşhisi koyabilmektir. Şekil 2.2'den görülebileceği gibi röntgen görüntüsü üzerinden örüntü tanıma ile hastalıklı oluşum kolaylıkla tespit edilebilmektedir.



Şekil 2.2 : İnsan vücudundaki hastalığın teşhisi için röntgen kullanımı [13].

Günümüzde oldukça fazla ilgi çeken diğer bir alan ise ses tanıma sistemleridir. Ses tanıma, bir ses sinyalini bir algoritma vasıtasıyla kelimeler dizisine veya bilinen bir ses sinyaline eşleştirme sürecidir. Ses tanıma üzerine yapılan çalışmaların bir kısmı insanların mekanik modeller ile insan sesini simüle etmek istemesinden ileri gelmektedir. Ayrıca ses tanıma, bilgisayar tabanlı sistemlerin insan sesi ile verilen komutları takip etmesini ve anlamasını mümkün kılabilmektedir. Örneğin insan makine arayüzü olarak, telefon üzerinden otomatik çağrı başlatma, komut tabanlı bilgi seyahat sistemleriyle iletişim, hava durumunu öğrenme, banka işlemleri, havacılık, konuşmayı metne çevirme ve özellikle Telekom operatörleri tarafından otomatik işlemler yapılması ses tanıma sistemlerinin başlıca uygulama alanlarıdır (Şekil 2.3).



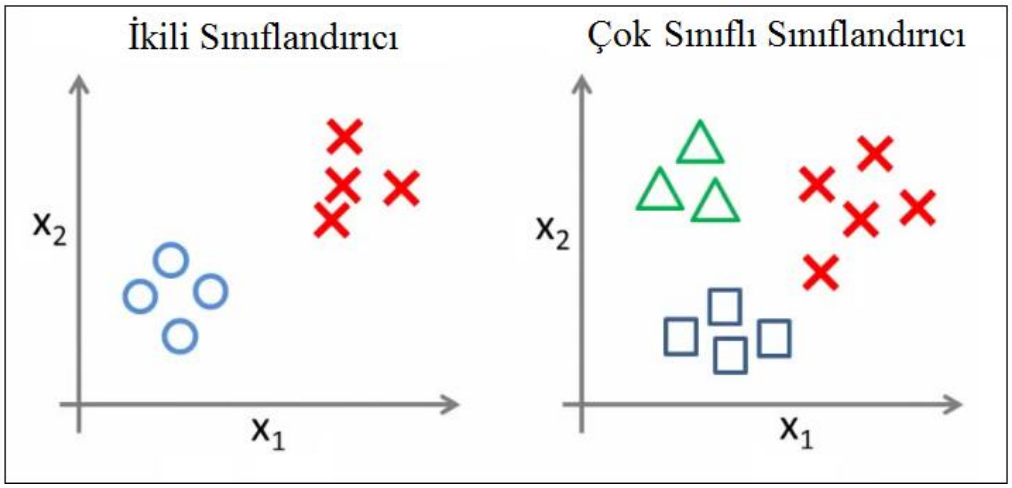
Şekil 2.3 : Ses tanıma sistemi için temsili çalışma prensibi.

Ayrıca örüntü tanıma biyometrik ve askeri savunma sistemleri gibi daha birçok sınıflandırma uygulamaları için çalışma sahası bulmaktadır.

2.3 Sınıflandırma

Sınıflandırma, örüntü tanıma sistemlerinde nesneleri tanımlamak veya sınıflandırmak amacıyla kullanılan yöntemin genel adıdır. Temel olarak elde edilen verinin belirli ayırt edici özelliklerinden faydalanarak birbirlerinden ayrıştırılması için kullanılmaktadır. Sınıflandırma, bu ayrıştırmayı yaparken verinin doğasında bulunan ayırt edici özellikleri kullanabileceği gibi önceden belirlenmiş bir karar kuralına uygun olarak gerçekleştirebilmektedir.

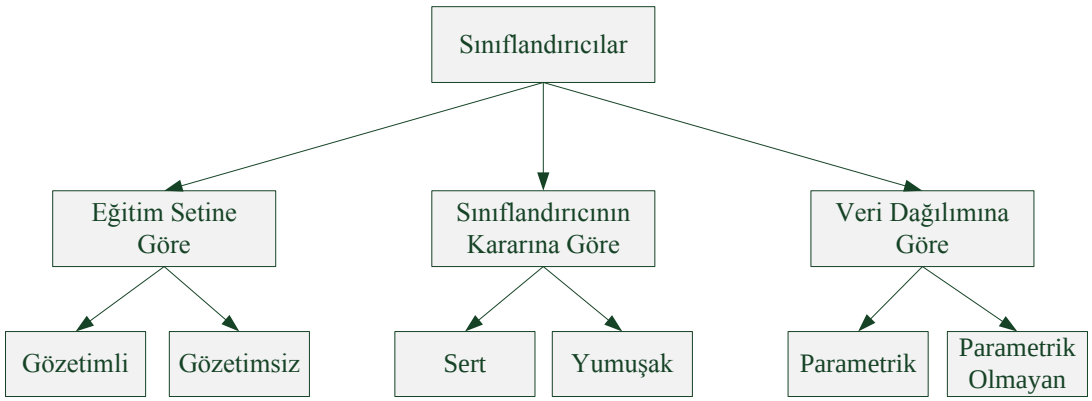
Sınıflandırıcılar ayırt edebildikleri sınıf sayısına göre 2 farklı şekilde isimlendirilirler. Birincisi, sadece iki sınıfı tanımlayabilen ikili sınıflandırıcılardır. Buna en güzel örnek algıladığı bilginin bir sınıfa ait olup olmadığını tespit eden detektörlerdir. İkincisi ikiden çok sınıfı tanımlayabilen çok sınıflı sınıflandırıcılardır. Bu tipe verilebilecek örnek alfabedeki karakterleri ya da rakamları tanıyan sistemlerdir (Şekil 2.4). Çok sınıflı sınıflandırıcılar temelde birçok ikili sınıflandırıcının birleştirilmesi ile oluşturulan sınıflandırıcılardır.



Şekil 2.4 : İkili ve çok sınıflı sınıflandırıcı [14].

2.3.1 Sınıflandırma ağacı

Sınıflandırma üzerine yapılan çalışmalar incelendiğinde değişik bakış açıları ile hazırlanmış birçok genel sınıflandırma ağacı bulunabilmektedir. Bu tezde bu yaklaşımları kapsamlı olarak ele alan Şekil 2.5'teki sınıflandırma ağacı ele alınmıştır [15].



Şekil 2.5 : Sınıflandırma ağacı.

2.3.1.1 Eğitim setine göre

Gözetimli sınıflandırma

Bilinmeyen bir kimliğe ait nesneyi sınıflandırmak için sınıfı bilinen eğitim örneklerinin kullanılmasıdır.

Gözetimsiz sınıflandırma

Bu sınıflandırma tipinde kimliği belirlenmiş herhangi bir nesne bulunmasına gerek yoktur. Bu sınıflandırmada kimliği belirsiz nesne kümesi, doğal karakteristiğine uygun olarak sınıflandırılır.

2.3.1.2 Sınıflandırıcının kararına göre

Sert sınıflandırma

Bu tip sınıflandırıcılarda her bir nesne tek bir sınıfa üyelik göstermek zorundadır.

Yumuşak sınıflandırma

Bu tip sınıflandırıcılarda her bir nesne çok sayıda sınıfa veya bir sınıfa üyelik gösterebilir. Bu üyeliklerin olasılıkları toplamı bir olmalıdır. Bu sınıflandırıcıya örnek olarak hava durumu tahminlerini verebiliriz (%70 olasılıkla yağmurlu, %30 olasılıkla yağmursuz).

2.3.1.3 Veri dağılımına göre

Parametrik sınıflandırıcılar

Bu sınıflandırıcılarda verinin istatistiksel dağılımı dikkate alınır. Sıklıkla eğitim örnekleri üzerinden gerekli istatistiksel bilgi üretilir.

Parametrik olmayan sınıflandırıcılar

Bu tipte veri hakkında herhangi bir ön varsayım yapılmaz. Parametrik olmayan sınıflandırıcılar sınıfların ayrımlarını hesaplamak için istatistiksel bir parametreden faydalanmazlar.

2.3.2 Sınıflandırma prosedürü

Sınıflandırmanın kullanılacağı alana özgü olarak literatürde birçok sınıflandırma prosedürü bulmak mümkündür. Sınıflandırma 5 temel adımdan oluşmaktadır. Bunlar sınıflandırılacak verinin belirlenmesi, öznitelik seçimi, karar kuralının belirlenmesi,

sınıflandırma ve başarımı hesaplama adımlarıdır. Sınıflandırma prosedüründeki her adımda belirli tanımlamalar ve işlemler gerçekleştirilir. Genel olarak sınıflandırma prosedürü Şekil 2.6'daki gibi ifade edilmektedir.

Şekil 2.6 : Genel sınıflandırma prosedürü [15].

Sınıflandırma prosedüründeki her bir adımın görevi kısaca aşağıdaki gibidir.

Sınıflandırılacak verinin belirlenmesi: Sınıflandırılacak nesnenin karakteristiğine göre veri setinin net bir şekilde belirlendiği adımdır. Örneğin alfabedeki karakterlerin sınıflandırması amaçlanıyorsa tüm karakterlere ait resimlerin bulunduğu veri setinin hazırlanmasıdır.

Öznitelik seçimi: Belirlenen veri setindeki her bir nesneye ait en ayırt edici özelliklerin seçildiği adımdır. Örneğin alfabedeki karakterlerin sınıflandırması için her bir karakteri birbirinden ayırt eden kenarların veya çizgilerin belirlenmesidir.

Karar Kuralının Belirlenmesi: Öznitelikleri çıkarılmış veri seti üzerinde başarılı bir sınıflandırma yapmak için en uygun karar kuralının belirlendiği adımdır. Bir bakıma en yüksek başarım verecek sınıflandırıcıyı belirlemektir.

Sınıflandırma: Belirlenen karar kuralına bağlı olarak veri setindeki nesnelerin sınıflandırmasının yapıldığı adımdır.

Başarımı Hesaplama: Yapılan sınıflandırmanın başarımını hesaplamak için gerekli doğrulamanın yapıldığı adımdır.

2.3.3 Sınıflandırma Algoritmaları

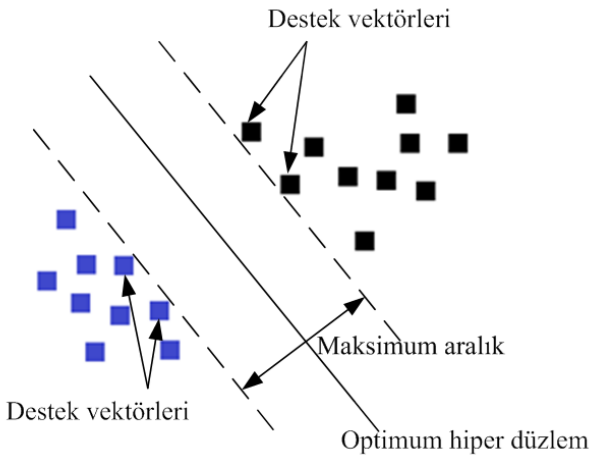
Bu bölümde, tezin deneysel benzetiminde kullanılmış ve literatürde oldukça fazla uygulaması bulunan destek vektör makineleri ve en yakın komşuluk algoritmaları ele alınmıştır.

2.3.4 Destek vektör makineleri

Destek Vektör Makineleri (Support Vector Machines-SVM), V.Vapnik ve diğerleri tarafından önerilmiş bir sınıflandırma algoritmasıdır [5]. SVM temelde iki sınıflı verilerin sınıflandırılması için geliştirilmiş fakat daha sonra çok sınıflı ve doğrusal olmayan problemler için genişletilmiştir. SVM' in temel çalışma prensibi sınıfları birbirinden ayıran en uygun karar sınırlarını (hiper düzlemi) belirleyerek sınıflandırma yapmaktır.

Literatüre baktığımızda yüz tespiti ve tanıma için G.Guo ve diğerlerinin yaptığı çalışmada 2 büyük yüz veri seti kullanılarak SVM ile yüz tanıma çalışması yapılmıştır [16]. Osuna ve diğerleri yüz tespiti için SVM'i kullanmış ve gerçek bir yüz ile olmayan yüzleri birbirinden ayırt edecek uygulama üzerinde çalışmıştır [17]. Bir diğer örnek çalışma ise Pontil ve Verri tarafından SVM ile "Columbia object image library" veri setindeki 3D objeleri tanıma çalışmasıdır [18]. Bu çalışmalarla birlikte SVM'in geleneksel yöntemlerden daha üstün başarı göstermiş olması, birçok uygulama alanında kullanılmasına neden olmuştur [19-21].

SVM sistemini diğer sınıflandırıcılardan ayıran temel özellik, iki sınıfı birbirinden ayıran karar sınırını, o sınır ile ona en yakın farklı sınıf verileri arasındaki mesafeyi maksimum yapacak şekilde belirleyebilmesidir. Buna örnek olarak Şekil 2.7 verilebilir.

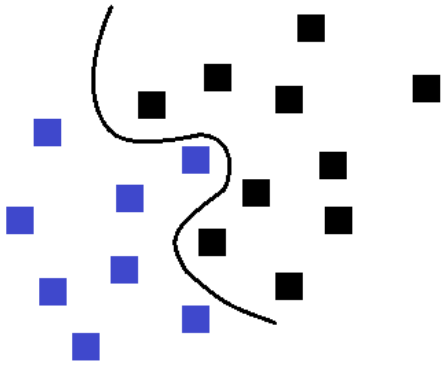


Şekil 2.7 : Doğrusal SVM karar yapısı.

Şekil 2.7'deki karar sınırı veya optimum hiper düzlem maviler ile siyahların tamamını birbirinden ayıran optimum sınırı meydana getirmiştir. SVM ile elde edilen optimum hiper düzlem, her iki sınıf arasındaki aralığı maksimum yapan bir düzlem

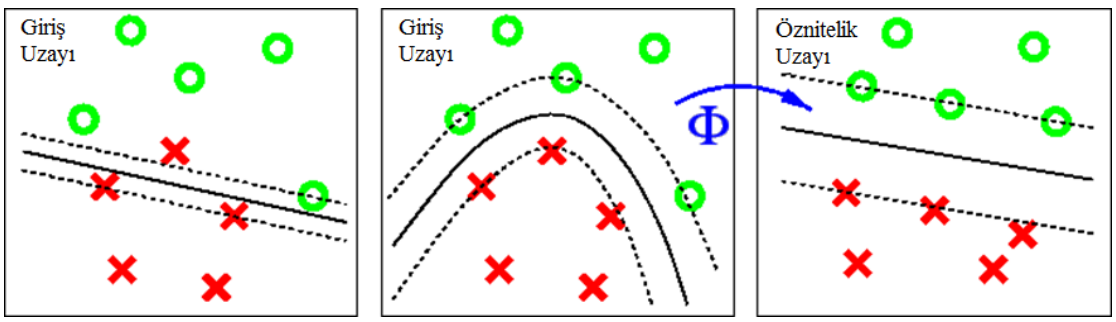
olmak zorundadır. Şekilde görülen destek vektörleri ise maksimum aralığın garanti edilmesinde kullanılan sınıf elemanlarıdır.

Yukarıdaki örnek, doğrusal bir karar sınırı kullanması nedeniyle doğrusal sınıflandırıcı olarak adlandırılır. Fakat birçok sınıflandırma problemi bu kadar basit değildir ve genellikle optimum sınıflandırma için oldukça fazla karmaşık yapıya ihtiyaç duyulmaktadır. Bu durumu Şekil 2.8'de görebiliriz. Bir önceki örnekle karşılaştırdığımızda buradaki örnekte mavi ve siyahları ayırmak için doğrusal olmayan bir eğri kullanılması gereklidir. SVM algoritması bu tip uygulamalar için de oldukça uygun bir sınıflandırıcıdır.



Şekil 2.8 : Doğrusal olarak sınıflandırılamayan bir problem.

Doğrusal olmayan karar sınırları ilk olarak Boser ve diğerleri [8] tarafından önerilmiştir. Bu çalışmada nesnenin, doğrusal olmayan bir fonksiyon ile daha yüksek boyutlu bir öznitelik uzayına dönüştürülebileceği ifade edilmiştir. Diğer bir deyişle giriş uzayında doğrusal olarak sınıflandırılamayan veri kümesi, daha yüksek boyutlu bir uzayda temsil edilerek, bu uzayda doğrusal sınıflandırılmıştır. Şekil 2.9, doğrusal olmayan SVM'in dayandığı temel fikri göstermektedir.



Şekil 2.9 : Soldaki doğrusal ayrım yapıldığı durum, ortadaki doğrusal olmayan sınıflandırma yapıldığı durum, sağdaki Φ dönüşümü ile öznitelik uzayına geçiş [22].

Yapılan dönüşümü matematiksel olarak ifade etmek gerekirse, giriş uzayındaki veri $\boldsymbol{x}$ olmak üzere, önceden belirlenmiş doğrusal olmayan bir Φ dönüşümü ile X orjinal giriş uzayı, yüksek boyutlu F öznitelik uzayına denklem 2.1'deki gibi haritalanabilir.

$$\Phi : X \rightarrow F \tag{2.1}$$

$$\boldsymbol{x} \rightarrow \Phi(\boldsymbol{x}) \tag{2.2}$$

Fakat doğrusal olmayan sınıflandırma için yüksek boyutlu öznitelik uzayına geçiş ve bu uzayda yapılan sınıflandırma oldukça fazla işlem yükü getirmektedir. Bu nedenle yüksek boyutlu uzayın işlemsel karmaşıklığından kurtulmak amacıyla $\boldsymbol{x}$ verisinin $\Phi(\boldsymbol{x})$ öznitelik uzayındaki temsilini, giriş uzayında hesap edebilecek bir yaklaşım uygulanabilmektedir [5, 23]. Bu dönüşüm Mercer çekirdek teoreminden faydalanılarak yapılmaktadır [24]. Bir Mercer çekirdeği olan $k(\boldsymbol{x}, \boldsymbol{y})$ fonksiyonunun Mercer şartlarını sağladığı düşünüldüğünde [25, 26], $k(\boldsymbol{x}, \boldsymbol{y})$ fonksiyonu Hilbert öznitelik uzayındaki Φ dönüşümünü ifade eden bir fonksiyon olarak tanımlanabilmektedir ve $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$ iç çarpımlar cinsinden ifade edilebilmektedir.

Öznitelik uzayındaki işlemlerin giriş uzayında yapılmasını sağlayan bu çekirdek fonksiyonları, yüksek boyutlu öznitelik uzayındaki işlemsel yükü ortadan kaldırmaktadır. Diğer bir deyişle bu fonksiyonlar öznitelik uzayındaki doğrusal sınıflandırmanın giriş uzayında da yapılabilmesini sağlamaktadır. Fakat çekirdek fonksiyonları her uygulama veya veri setine göre değişiklik göstermektedir. Bu nedenle bir çekirdek fonksiyonunun belirli bir kullanım alanı olmadığı gibi ancak deneysel sonuçlar ışığında uygulamaya özgü çekirdek fonksiyonları bulunabilmektedir. Örneğin Şekil 2.10'da bulunan veri dikkate alındığında giriş uzayında bir çember ile ayrıştırılabilecek iki ayrı veri, değerlerinin karesini alan polinom bir çekirdek foksiyonu ile öznitelik uzayına haritalanarak doğrusal olarak sınıflandırılabilmektedir.

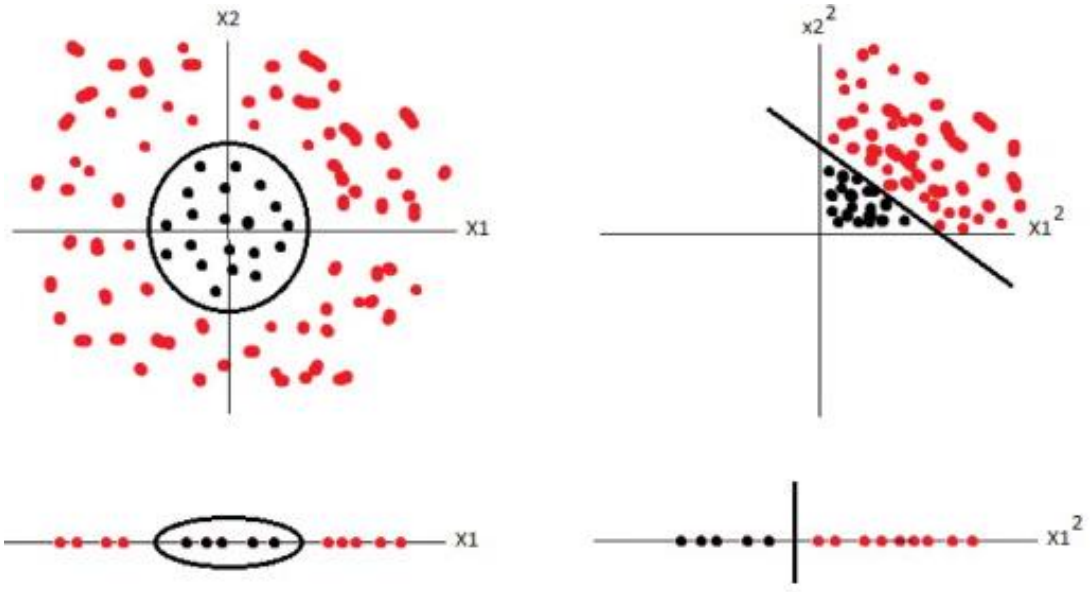
$\boldsymbol{x}$ ve $\boldsymbol{y}$ vektör veya matris olmak üzere en çok bilinen çekirdek fonksiyonları denklem 2.3 ve denklem 2.4'teki gibidir.

- Polinom çekirdek fonksiyonu (d: polinom derecesi)

$$k(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}^T \boldsymbol{y})^d \tag{2.3}$$

- Gauss çekirdek fonksiyonu (t: kullanıcı tanımlı parametre)

$$k(\boldsymbol{x}, \boldsymbol{y}) = e^{(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/t)} \tag{2.4}$$



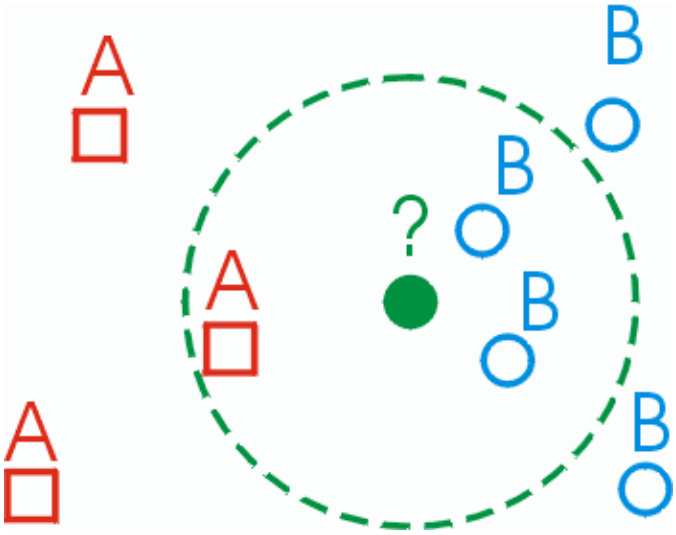
Şekil 2.10 : Polinom çekirdek fonksiyonu ile öznitelik uzayında doğrusal ayrımın yapılması [27].

2.3.5 En yakın komşuluk algoritması

En yakın komşuluk algoritması (Nearest Neighbor-NN), oldukça basit ve etkin bir nesne tanıma tekniğidir. İlk olarak 1967 yılında T. M. Cover ve P. E. Hart tarafından önerilen NN algoritması, sınıfı belli olmayan test nesnesini, o nesne ile daha önceden sınıfları belirlenmiş nesneler arasındaki Öklid uzaklığını dikkate alarak sınıflandırma yapmayı sağlamaktadır [28]. NN sınıflandırıcısı literatürde çok geniş bir kullanım alanına sahiptir. En önemli uygulama alanlarından bazıları örüntü tanıma [29, 30] ve nesne tanımadır [31].

Bu algoritma dikkate alacağı k adet en yakın komşu sayısına göre k-NN olarak isimlendirilmiştir. k-NN sınıflandırıcısı test nesnesine en yakın mesafede olan k adet nesne içerisinden ağırlıklı olan sınıfa karar vererek çalışmaktadır. Basit olması tekniğin en büyük avantajı olmasına rağmen yüksek hafıza gereksinimi ve gerektirdiği işlemsel yük birer dezavantaj olarak gözükmüştür. Fakat NN sınıflandırıcısının performansını artıracak birçok çalışma literatürde yer almaktadır [32- 34].

Örneğin, Şekil 2.11'de A ve B'nin bilinen iki sınıf olduğunu düşünürsek 3-NN sınıflandırıcısı, yeşil test verisini en yakın 3 komşusunu dikkate alarak sınıflandırabilir. Bu durumda en yakın 3 komşunun 2 tanesi B ve 1 tanesi A sınıfına ait olduğu için 3-NN sınıflandırıcısı yeşil test noktasının B sınıfına ait olduğuna karar verecektir.



Şekil 2.11 : 3-NN ile sınıflandırma yapılması [35].

Sınıfları bilinen örnekler $\boldsymbol{d_i} \in \mathbb{R}^s, i = 1,2,..,n$ ve sınıfını bulmak istediğimiz test nesnesi $\boldsymbol{x} \in \mathbb{R}^s$ olmak üzere NN sınıflandırıcısına ait temsili algoritma Çizelge 2.1'deki gibidir.

Çizelge 2.1 : NN algoritması.

NN Algoritması

1. Giriş: k komşuluk sayısı, $\boldsymbol{d_i} \in \mathbb{R}^s, i = 1,2,..,n$ sınıfı bilinen örnekler ve $\boldsymbol{x}$ test nesnesi belirlenir.
2. $\boldsymbol{x}$ test nesnesi ile tüm $\boldsymbol{d_i}$ elemanları arasındaki Öklid uzaklığı hesap edilir.
3. Hesaplanan tüm mesafeler küçükten büyüğe sıralanır ve en küçük k adet mesafe seçilir.
4. Bu k adet komşunun ait olduğu sınıfların hangisi çoğunlukta ise o sınıfa karar verilir.

2.4 Yüz Tanıma Problemi

2.4.1 Literatür

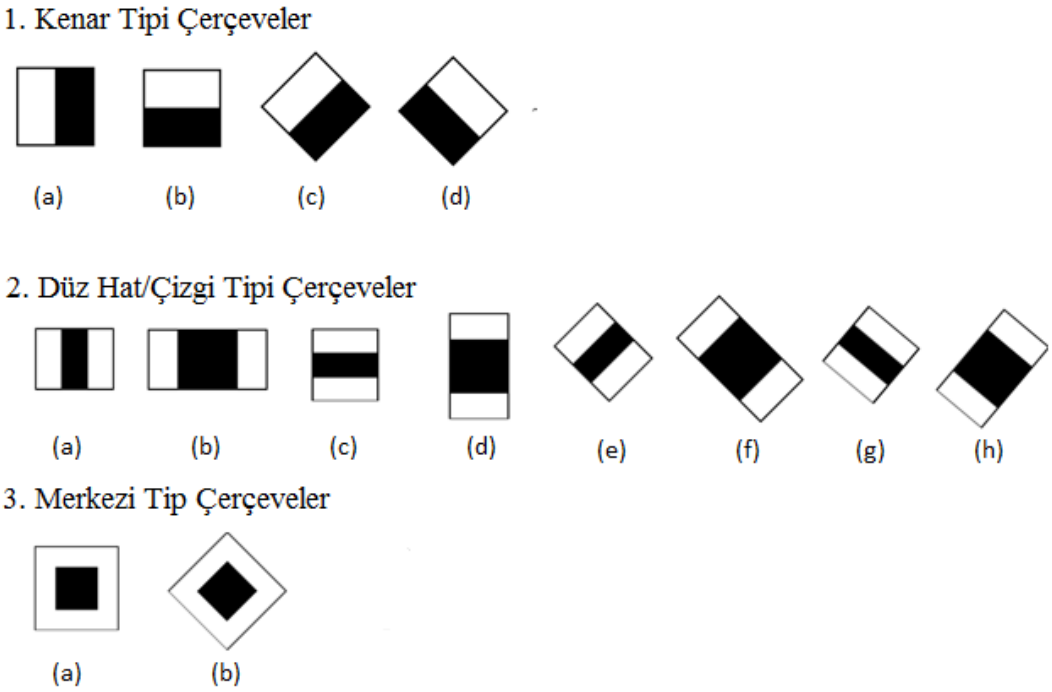
Bilinen en eski yüz tanıma çalışmaları Darwin [36] ve Galton [37] tarafından yürütülmüştür. Darwin bireyin değişken duygusal durumlarını dikkate alarak farklı yüz ifadelerini analiz ederken, Galton yüz profillerini analiz ederek çalışmalar yürütmüştür. Ancak yüz tanımada ilk gerçekçi yaklaşımlar 1960'lı yılların sonu 1970'li yılların başında yarı otomatik yüz tanıma sisteminin geliştirilmesi ile başlamıştır. Bu yaklaşımın temel prensibi sınıflandırma için göz, kulak, burun ve ağız çevresi gibi yüzün en ayırt edici özniteliklerinin kullanılmasıdır. Örneğin A. J. Goldstein ve L. D. Harmon'un 1971 yılında yayınlanan makalesinde saç rengi ve dudak kalınlığı gibi 21 farklı sübjektif gösterge kullanılmıştır [38]. Geliştirilen diğer tutarlı yüz tanıma yaklaşımları ise M. A. Fischler ve R. A. Elschlager [39] ile A. Yuille, P. Hallinan, ve D. Cohen [40] tarafından gerçekleştirilmiştir.

Fakat pratik uygulamalara temel oluşturmuş ilk yüz tanıma sistemi 2001 yılında Paula Viola ve Michael Jones tarafından geliştirilen Viola-Jones nesne tespit sistemidir [41]. Bu sistem gerçek zamanlı uygulamalarda oldukça rekabetçi doğruluk oranı yakalamış ilk nesne tanıma sistemidir. İlk zamanlarda tespit sistemi için birçok nesne kullanılmışsa da bu sistem temelde yüz tanıma sistemi için tasarlanmıştır. Bu nedenle Viola-Jones yapılan bu çalışmada yüz resimlerini önden ve dik gören şekilde seçerek çalışmışlardır. Şöyle ki yüzü tespit etmek için yüzün tamamı kameraya bakıyor olmalı ve yüz kesinlikle bir yöne eğimli olmamalıdır. Bu durum sistem için bir kısıt olsa da yüz tanımanın bu şekilde rekabetçi sonuçlar vermesi çalışmanın başarılı olarak değerlendirilmesine neden olmuştur.

Bu sistemde sınıflandırıcı, yüz resmini Şekil 2.12'deki gibi belirli büyüklerde ayarlanmış çerçevelerle tarayarak çerçeve içerisinde bulunan toplam siyah piksel değeri ile toplam beyaz piksel değerlerinin oranlarına göre belirli sınıflandırma sınırları belirlemiştir.

Şekil 2.13'te yüz resmine uygulanmış değişik çerçeveler gösterilmiştir. Örnekte görüldüğü gibi göz bölgesindeki siyah bölge yanakların üst tarafından daha fazladır. Benzer şekilde sınıflandırıcı tasarımında, burun, saç, ağız gibi bölgelerinde siyah-beyaz oranları da oluşturulmuştur. Nihai olarak bu sınıflandırıcı her bir çerçeve için

minimum ve maksimum siyah/beyaz değerlerini dikkate alarak sınıflandırma yapmıştır.



Şekil 2.12 : Viola-Jones yönteminde kullanılan çerçeveler [42].

Şekil 2.13 : Çerçevelerin birey üzerinde uygulanmış hali [42].

Özellikle Viola-Jones algoritması yüksek başarısı, basitliği ve reel olarak uygulanabilirliği sayesinde yüz tanıma uygulamalarında önemli bir yere sahiptir.

2.4.2 Yüz tanıma adımları

Yüz tanımanın adımları temelde sınıflandırma adımları ile aynıdır. Bu bölümde Şekil 2.6'daki sınıflandırma adımlarının yüz tanıma sistemlerinde karşılık geldiği anlamlar ele alınacaktır.

2.4.2.1 Sınıflandırılacak verilerin belirlenmesi

Yüz tanıma uygulamalarında sınıflandırılacak veri iki aşama ile belirlenir. İlk olarak elde edilen görüntüdeki yüz algılanır, daha sonra görüntü işleme metotlarıyla iyileştirilerek öznitelik çıkarma adımına geçilir.

Yüz algılama

Yüzün algılanması, yüz bölgesinin tespit edilmesi ve resimden çıkarılması ile yapılır. Bunun için ilk geliştirilen algoritmalar bireyin başına ait sınırların tespitine dayanmaktadır [43]. Günümüzdeki gelişmiş yüz algılama sistemleri yüze ait diğer belirgin özniteliklerin algılanmasından da faydalanmaktadır. Bunlardan bazıları kulak, kaş, burun ve ağızdır. Bununla birlikte gerçek zamanlı yüz takip sistemleri gibi devamlı yüz algılama gerektiren uygulamalarda mevcuttur. Buradaki hedef gerçek zamanlı güvenlik kamerasından alınan görüntü üzerinden yüzü algılayarak yüzün konum değişikliklerini anlık olarak takip etmektir.

Yüz algılamanın temel problemi resmin içinde yüzün dışında birçok nesnenin bulunuyor olmasıdır [44, 45]. Bu bağlamda yüz algılamayı zorlaştıran etmenleri aşağıdaki gibi özetleyebiliriz.

Yüzde bulunan ifade: Resmin çekildiği anda kişinin yüzünde bulunan ifadedir (gülme, ağlama vb.).

Duruş: Resmin çekildiği anda bireyin hareketi veya kameranın açısına bağlı olarak resmin içindeki yüzün farklı bir açıda olması durumudur.

Kısmi kapalı yüz: Kişinin gözlük veya şapka takması, sakallı olması ya da resimdeki yüzün belirli bir kısmının başka bir cisimle kapatılmış olması durumudur.

Çevresel koşullar: Resmin çekildiği yerdeki çevresel koşulların (karlı bir bölge, ışık altı vb.) resmin içindeki yüzün netliğini ve görünürlüğünü etkilemesidir.

Fakat günümüzdeki bazı yüz tanıma uygulamaları yüzü tüm çehresiyle algılamayı gerektirmez. Bunu gerektirmemesinin nedeni uygulama öncesi yüz resimlerinin bir yerde depolanmış ve normalize edilmiş olmasıdır. Bu yüzler uygulamalar için standart bir formattadır. Bu durum için verebileceğimiz en iyi örnekler güvenlik güçlerinin çektiği mahkûm fotoğrafları ve pasaport için çekilen biyometrik fotoğraflardır.

Önişleme

Bu aşamada, görüntü işleme teknikleri kullanılarak, yüz resimleri sistemin tanıma performansını arttıracak şekilde geliştirilirler. Yüz tanıma sistemlerinde aşağıdaki önişleme adımlarından birkaçı veya hepsi kullanılabilir.

Aydınlık dengeleme: Farklı aydınlık seviyelerinde çekilen yüz resimlerinin tanıma performansını düşürmemesi için aydınlık dengelemesi yapılır.

Görüntü boyutunu normalize etme: Bu yöntem genellikle resim boyutunu sistem tarafından kullanılacak resim boyutuna çevirmek için kullanılır (örneğin 128×128). Bu önişleme, daha çok yüz resmini bir bütün olarak değerlendirilen sistemlerde gerekli olmaktadır.

Histogram dengeleme: Bu yöntem genellikle çok koyu ve çok parlak resimlerde resim kalitesini arttırarak tanıma performansını iyileştirmek için kullanılır. Bu yöntem resmin dinamik (kontrast) aralığını değiştirerek önemli yüz özniteliklerini daha belirgin hale getirir.

Ortanca süzgeci: Resimlerde bulunan gürültüyü mümkün olduğunca bilgi kaybı olmaksızın temizlemek için kullanılır.

Yüksek geçiren filtre: Yüzün ana hatlarına bağlı öznitelikleri daha belirgin hale getirmek için kullanılır.

Arka plan temizleme: Sadece yüz resmi ile sınıflandırma yapabilmek için arka plan resimden çıkarılabilir. Bu yöntem özellikle resmi bütüncül olarak kullanan sınıflandırıcılar için gereklidir.

Kaymayı ve dönmeyi normalize etme: Bazı durumlarda baş kısmının kaydırılmış veya döndürülmüş olduğu yüz resimleri ile çalışmak gerekebilir. Baş, yüz özniteliklerinin belirlenmesinde en önemli kısımdır. Özellikle yüzün ön görünüşüne dayalı yüz tanıma sistemlerinde kafa pozisyonundaki kayma ve dönmelerin algılanması ve normalize edilmesi oldukça önemlidir.

2.4.2.2 Öznitelik belirlenmesi

Veri setinin belirlenmesinden sonra, sınıflandırma için kullanılacak yüze ait anahtar kısımlar öznitelik çıkartma modülü ile elde edilir. Diğer bir deyişle, bu modül yüz resmini yeterli düzeyde temsil eden öznitelik vektörünü bulmakla görevlidir.

Öznitelik çıkarma, örüntü tanıma sistemlerinin başarımı için oldukça kritiktir. Fakat genellikle etkin bir öznitelik çıkarımı örüntü tanıma ve veri madenciliği için oldukça zorlu bir görevdir. Bu nedenle konu hakkında oldukça fazla çalışma bulmak mümkündür. Bunlardan bazıları temel bileşen analizi (Principal Component Analysis-PCA) [46] ve bağımsız bileşen analizidir (Independent Component Analysis-ICA) [47].

Yüz tanıma sistemlerinde sıklıkla kullanılan bütüncül öznitelik çıkarma metotlarından bazıları ise Eigenface [46], Laplacianface [48], Downsampled ve Randomface'dir [7] (Şekil 2.14).

Şekil 2.14 : Değişik öznitelik metotlarına ait öznitelik resimleri.

2.4.2.3 Karar kuralının belirlenmesi ve sınıflandırma

Yüz tanıma için karar kuralının belirlenmesi ve sınıflandırma, diğer bir deyişle yüz tanıma adımı test resminin hangi sınıfa ait olacağına karar verecek algoritmadır. Sınıflandırıcı seçimi, yüz tanıma sisteminin tasarlanış amacına göre değişiklik gösterebilir. Fakat sınıflandırma algoritmaları bölümünde anlatılan Doğrusal SVM ve NN algoritmaları yüz tanıma sistemleri içinde sıklıkla kullanılan algoritmalardır.

2.5 Rakam Tanıma Problemi

Elle yazılmış rakamları tanıma problemi makine öğrenmesinin oldukça bilinen bir uygulamasıdır. Ana hedef ayrı ayrı yazılmış rakamların bulunduğu resimdeki 0-9 arası rakamları tanımaktır. Pratikte değişik yazı stilleri, kötü el yazısı veya aynı rakamlar için anlaşılmaz yazılar bu tanımayı oldukça zorlaştırır. Çünkü her birey birbirinden farklı yazı stiline sahiptir. Hatta bir birey aynı karakteri iki kere aynı şekilde yazamamaktadır. Bu nedenle tüm elle yazılmış rakamları yüksek doğrulukla tanıyacak bir sistemi geliştirmek oldukça zordur.

Elle yazılmış rakamları tanıma, karakter tanıma sistemleriyle benzer bir altyapıya sahiptir. Karakter tanıma için kullanılan en yaygın metotlardan biri Multi-Layer

Perceptron (MLP) yöntemidir [49, 50]. MLP birçok sınıf içerisinde karakter tanımada oldukça başarılıdır. MLP'nin başarılı olmasındaki nedenlerden biri doğrusal olmayan sınıflandırma yapabilmesidir.

Bununla birlikte SVM'in karakter tanımadaki başarısı da bu alanda geniş bir kullanım sahası bulmasını sağlamıştır [51]. Bu çalışmalardan biri Neves ve diğerlerinin geliştirdiği SVM tabanlı rakam tanıma sistemidir [52]. Bu sistemde her bir rakam ikilisi için bir kere SVM algoritması koşturulmuştur. Bu nedenle 10 farklı rakamın birbirinden farklı ikili çiftleri için 45 adet SVM koşturulmuş ve sonuçlar içerisinden hangi rakamın daha fazla etiketlendiğine göre sınıflandırma yapılmıştır.

Karakter tanıma sistemleri çevrimiçi ve çevrimdışı olmak üzere ikiye ayrılmaktadır. Çevrimiçi karakter tanıma sistemleri karakterin oluşturulduğu anda tanımayı yaparken, çevrimdışı sistemler karakter oluşturulduktan, tarandıktan ve bilgisayar ortamına aktarıldıktan sonra tanıma yapmaktadır. Bu bağlamda teze konu rakam tanıma benzetimleri için çevrimdışı karakter tanıma sistemi dikkate alınmıştır.

2.6 Başarım Kriterleri

Sınıflandırma algoritmalarının başarımını değerlendirmek düşünüldüğü kadar kolay bir çalışma değildir. Çünkü her bir sınıflandırıcı geliştiriliş amacına uygun olarak farklı başarım kriterlerine sahiptir. Örneğin çevrimiçi sınıflandırıcı için en önemli başarım kriterlerinden biri hesaplama hızı iken, çevrimdışı sınıflandırıcılar için bu önemli bir kriter değildir. Bu bağlamda literatür çalışmalarında baz alınan temel başarım kriterleri aşağıdaki gibidir.

- Doğruluk oranı / Hata oranı
- Hesaplama hızı

Bu parametrelerle birlikte özellikte pratikte kullanılacak bir sınıflandırma sisteminin başarımını ölçerken aşağıdaki kriterlerinde dikkate alınması daha tutarlı sonuçlar verecektir.

- Bellek kullanımı
- Ölçeklenebilirlik
- Uyarlanabilirlik
- Taşınabilirlik

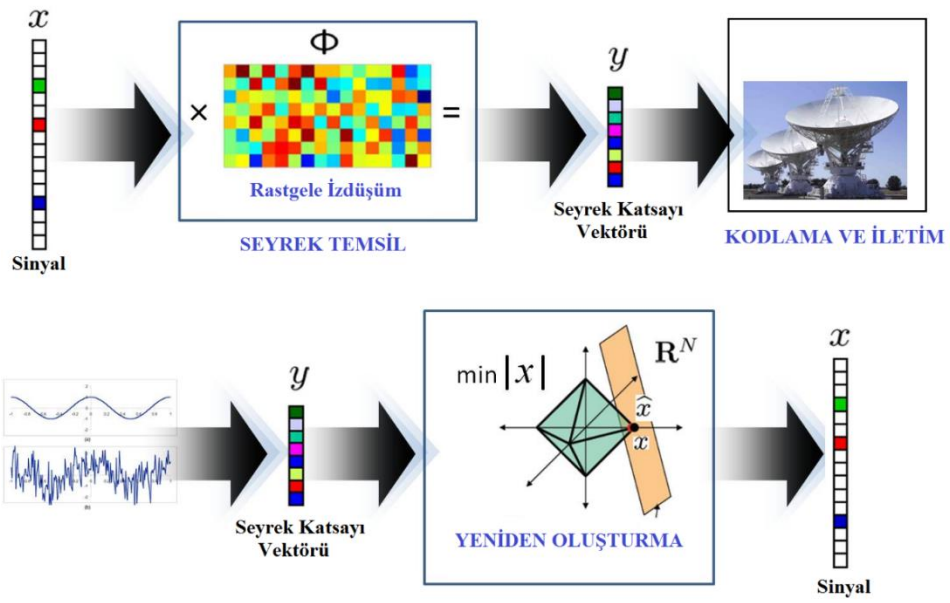
3. SEYREKLİK TABANLI İŞARET İŞLEME

Seyreklik, son yıllarda ortaya çıkmış ve birçok bilimsel alanda başarılı çözüme temel oluşturmuş önemli bir konsepttir. Seyreklik, özellikle işaret işleme, makine öğrenmesi ve görüntü işleme gibi alanlarda oldukça güçlü bir uygulama sahasına sahiptir. Görüntü işleme problemlerinden, gürültü giderme, görüntü restorasyonu, çözünürlük değiştirme, görsel izleme, görüntü sınıflandırma vb. uygulamalar için literatürde oldukça fazla çalışma bulmak mümkündür [53, 54]. Seyreklik, temelde sıkıştırılmış algılama teoremi ile oldukça ilintili bir konsepttir.

3.1 Sıkıştırılmış Algılama

Uzun yıllardır sinyallerin sıkıştırılması veya bant sınırlı sinyallerin örneklenmesi problemlerinin çözümünde geleneksel Shannon örneklem teoremi kullanılmaktadır. Shannon bu teoreminde bant sınırlı bir sinyalin maksimum frekansının 2 katı ile örneklenmesi ile aynı sinyalin tekrar elde edilebileceğini konu alır. Fakat son dönemlerde yapılan araştırmalar bilinen sinyal sayısının oldukça fazla olması durumunda, bir sinyali bu bilinen sinyallerin birkaçının birleşimiyle (seyrek) ifade ederek, bu sinyali Shannon örneklem teoreminden daha efektif bir şekilde temsil edebileceğimizi göstermiştir. İşte bu yaklaşım literatürde sıkıştırılmış algılama (Compressive Sensing-CS) teorisi olarak geçmektedir. CS üzerine 1960'lı yıllara dayanan çalışmalar bulunabilse de günümüzde kullanılan CS yapısı ilk olarak Donoho tarafından önerilmiştir [6]. Daha sonra bu CS teoremi Candes ve diğerleri tarafından matematiksel olarak ele alınmış ve rasyonel bir yaklaşım olduğu ispat edilmiştir [55]. CS teorisinin bu rasyonel yapısı ve uygulamalardaki başarısı yeni bir örnekleme teoremi olarak gösterilmesini sağlamıştır [56- 58].

CS teorisi, seyrek temsil, kodlama ve yeniden oluşturma olmak üzere 3 temel aşamadan oluşmaktadır (Şekil 3.1). CS'in günümüzdeki uygulama alanlarından bazıları bilgi kuramı, işaret işleme, görüntü işleme, medikal, jeofizik ve astronomik görüntülemedir.



Şekil 3.1 : Sıkıştırılmış algılamanın adımları [59].

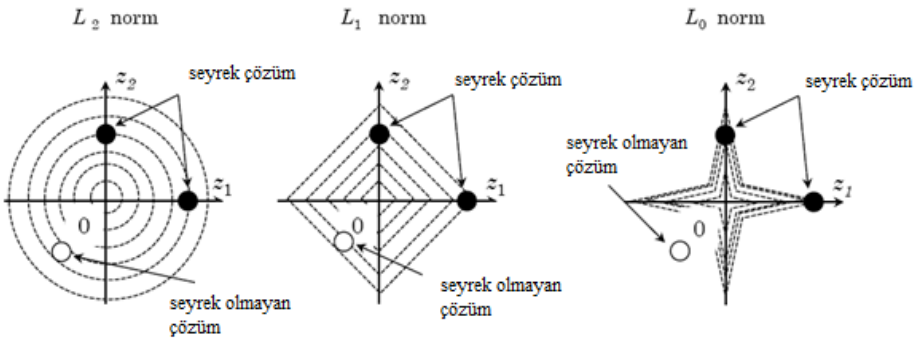
3.2 Seyrek İşaret İşlemede Kullanılan Norm İfadeleri

Bu kısımda seyreklik tabanlı işaret işleme denklemlerinde sıklıkla kullanılan normlar kısaca izah edilecektir.

Normlar vektör ya da matris uzayında vektörlerin toplam uzunluğunu veya büyüklüğünü matematiksel olarak ifade etmeye yararlar. $\boldsymbol{x}$ bir vektör veya matris olmak üzere l_p normu formal olarak denklem 3.1'deki gibi tanımlanır.

$$\|\boldsymbol{x}\|_p = \sqrt[p]{\sum_i |x_i|^p} \tag{3.1}$$

Değişik p değerlerine göre l_p normları birbirine çok benziyor gözüksede matematiksel olarak ciddi farklılıklara sahiptir (Şekil 3.2).



Şekil 3.2 : Normların geometrik gösterimi [60].

3.2.1 l_0 normu

l_0 normu bir vektördeki 0'dan farklı eleman sayısını bulan uzunluk normudur. Normu incelediğimizde sıfırdan büyük her değer için sonucun 1 olacağını görebiliriz. $supp(\boldsymbol{x})$, $\boldsymbol{x}$ vektöründeki 0'dan farklı elemanların pozisyonlarını içeren küme olmak üzere l_0 normu denklem 3.2'deki gibi ifade edilir.

$$\|\boldsymbol{x}\|_0 = |supp(\boldsymbol{x})| \tag{3.2}$$

CS'in gelişmesi ile seyrek çözümlerde sıklıkla kullanılan bir normdur. l_0 normunun içbükey (konveks) olmaması nedeniyle, bu norm için en iyi çözümü bulmak oldukça zorlu hatta imkânsızdır. Bu nedenle birçok durumda içbükey olduğu bilinen bir norm ile çözüm yapılmaya çalışılır.

3.2.2 l_1 normu

Birçok sahada kullanılan bu norm Manhattan norm olarak da bilinir. l_1 normu denklem 3.3'teki gibi tanımlanır. İçbükey bir norm olan l_1 normu genellikle l_0 normunun yerine kullanılarak seyrek çözüm elde edilir.

$$\|\boldsymbol{x}\|_1 = \sum_i |x_i| \tag{3.3}$$

3.2.3 l_2 normu

Tüm normlar içinde en popüler olanı l_2 normudur. Öklid normu olarak da bilinir ve neredeyse tüm mühendislik uygulamalarında kullanılır. l_2 normu denklem 3.4'teki gibi tanımlanır.

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_i {x_i}^2} \tag{3.4}$$

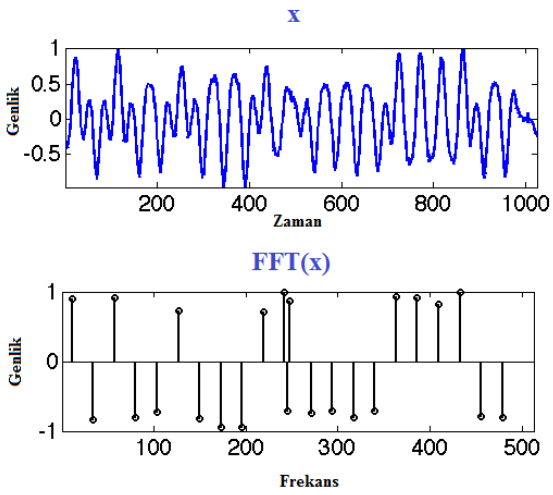
İşaret işlemedeki bilinen en büyük kullanımı ortalama karesel hatanın hesaplanmasıdır. Fakat verdiği sonuçların tüm sınıflar üzerindeki dağılımı nedeniyle genellikle seyrek yaklaşımlarda kullanılmaz.

Bununla birlikte son dönemlerde yapılan çalışmalar l_2 normunun bazı seyreklik problemlerinin çözümünde başarılı sonuçlar verdiğini göstermiştir.

3.3 Seyreklik

Eğer $\boldsymbol{x} \in \mathbb{R}^s$ vektör sinyalinin elemanlarının büyük bir kısmı sıfır ve sadece birkaç elemanı sıfırdan farklı ise bu sinyal seyrek sinyal olarak isimlendirilir. Literatürde seyrek sinyallerin içerdiği sıfırdan farklı eleman sayısına göre isimlendirildiği görülebilir. Örneğin 5 elemanı sıfırdan farklı olan bir sinyal seyreklik seviyesi 5 sinyal olarak isimlendirilir.

Seyrek sinyale en güzel örneklerden biri birkaç sinüs sinyalinin birleşimi ile oluşan $\boldsymbol{x}$ sinyalinin Fourier dönüşümü ile elde edilen frekans bileşenleridir. Şekil 3.3'te görülebileceği gibi orijinal $\boldsymbol{x}$ sinyali bir zaman bölgesi sinyalidir. Fakat $\boldsymbol{x}$ sinyalinin Fourier dönüşümü alındığında oluşan frekans-genlik bileşenleri Şekil 3.3'te görülebileceği gibi seyrektir.



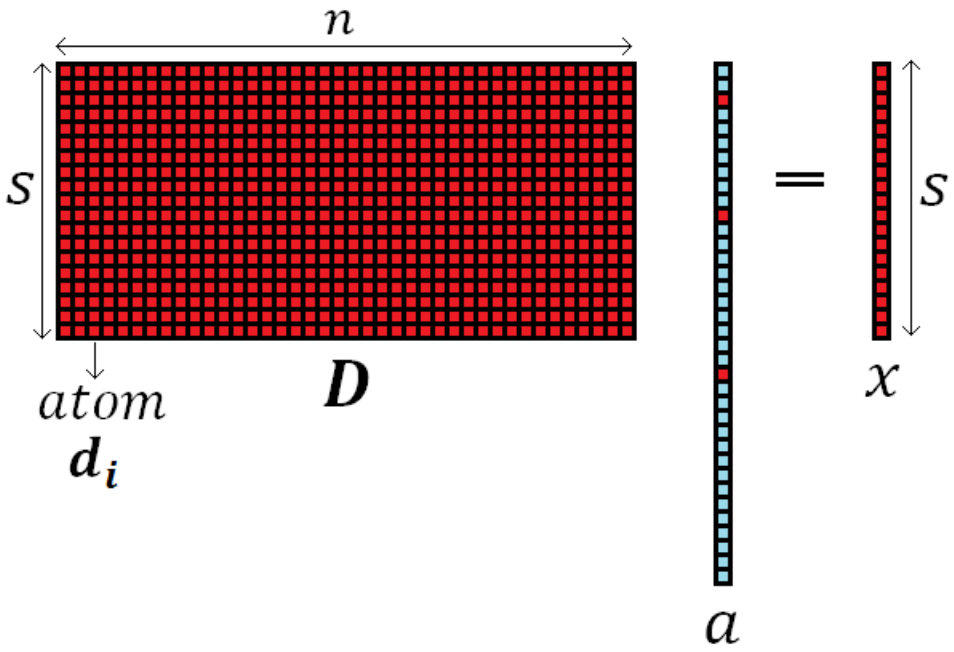
Şekil 3.3 : Zaman bölgesi sinyalin (üst), Fourier dönüşümü (alt) ile seyrek gösterimi [61].

Yukarıdaki örnekten de görülebileceği gibi herhangi bir sinyal bilinen bir dönüşüm metodu ile seyrek hale getirilebilir. Bu dönüşüm metodu yukarıda Fourier seçilse de her uygulamada sinyali seyrekleştirecek değişik dönüşüm metotları seçilebilir.

Seyrekliğin genel bir ifadesini yapabilmek için $\boldsymbol{x} \in \mathbb{R}^s$ boyutlu bir vektör sinyal olduğunu düşünelim. $\boldsymbol{x}$ sinyali ile ilintili baz vektörler $\boldsymbol{d}_i \in \mathbb{R}^s,\ i = 1, \ldots, n$ olmak üzere bu vektörlerin birleşiminden oluşan baz matris $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n] \in \mathbb{R}^{s \times n}$ olsun. Bu durumda seyreklikte $\boldsymbol{D} \in \mathbb{R}^{s \times n}$ matrisi sözlük, bu matrisin kolonlarını oluşturan $\boldsymbol{d}_i \in \mathbb{R}^s$ vektörleri ise atom olarak isimlendirilir. Bu durumda $\boldsymbol{x}$ sinyali

bilinen bu sözlük atomlarının doğrusal birleşimi olarak denklem 3.5'teki gibi ifade edilebilir (Şekil 3.4).

$$\boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{d}_i a_i = \boldsymbol{D}\boldsymbol{a} \tag{3.5}$$



Şekil 3.4 : Seyrek gösterim.

Burada $\boldsymbol{a} \in \mathbb{R}^n$, $n \times 1$ boyutlu seyrek katsayı vektörüdür. Dikkat edilirse seyrek bir vektör elde edebilmek için sözlükteki atom sayısının nesne boyutundan oldukça büyük olması gereklidir ($n \gg s$). Buradaki örnekte $\boldsymbol{a}$ seyrek katsayı vektörünün T_0 adet sıfırdan farklı elemanı olduğunu varsayarak T_0 seyreklik seviyeli vektör denklem 3.6'daki gibi ifade edilir.

$$\boldsymbol{a} = [0,0,a_3,0,\dots,0,a_{11},0,\dots,0] \in \mathbb{R}^n \tag{3.6}$$

$\boldsymbol{a}$ seyrek katsayı vektöründeki T_0 adet sıfırdan farklı elemanın sayısını, l_0 normu ile denklem 3.7'deki gibi ifade edebiliriz.

$$T_0 = \|\boldsymbol{a}\|_0 \tag{3.7}$$

Buraya kadar yaptığımız tanımlamalar $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ eksik belirtilmiş (under-determined) doğrusal denklem sisteminin çözümü ile anlamlı hale gelmektedir. Fakat bu eksik belirtilmiş doğrusal denklemin birden fazla çözümü olabileceğini rahatlıkla

görebiliriz. Eğer denklemin çözümü için ilave bir kısıt verilmez ise bu denklemin tekil bir çözümünü bulmamız mümkün olmayacaktır. Bu durumda denklemin çözümünde seyrekliğin tanımında belirtildiği gibi sıfırdan farklı eleman sayısının çok çok az olmasının istenmesi önemli bir kısıt oluşturmaktadır. Bunu ifade eden optimizasyon problemini denklem 3.8'deki gibi yazabiliriz.

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 , \qquad \boldsymbol{x} = \boldsymbol{D}\boldsymbol{a} \text{ koşulu altında} \tag{3.8}$$

Buradan hareketle $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ denkleminin çözümü için T_0 sıfırdan farklı eleman sayısını sınırlayarak bir çözüm elde edilmesi mümkün olmaktadır. $T_0 \ll n$ olduğu varsayılarak $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ denkleminin çözümü denklem 3.9'daki optimizasyon problemine eşit olmaktadır.

$$\|\boldsymbol{a}\|_0 \leq T_0 , \qquad \boldsymbol{x} = \boldsymbol{D}\boldsymbol{a} \text{ koşulu altında} \tag{3.9}$$

Buraya kadar sinyalin herhangi bir gürültü içermediği varsayılmıştır. Fakat gerçek zamanda gürültüsüz bir sinyal üzerinde çalışmak mümkün değildir. Eğer $\boldsymbol{x}$ sinyalinin küçük bir gürültüye sahip olduğunu varsayarsak yukarıdaki optimizasyon probleminin çözümünde sözlüğün doğrusal birleşimleri ile $\boldsymbol{x}$ sinyalini eksiksiz ve tam olarak ifade edebilecek bir çözüm bulmak mümkün olmayacaktır. Bu nedenle bozucu gürültüyü de dikkate alarak temel problemi denklem 3.10'daki gibi yazabiliriz.

$$\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a} + \boldsymbol{g} \tag{3.10}$$

Burada $\boldsymbol{g} \in \mathbb{R}^s$ olmak üzere gürültü terimini ifade etmektedir. Bu gürültü teriminin enerjisinin $\|\boldsymbol{g}\|_2 \leq \varepsilon$ hata eşik değeri ile sınırlandırıldığını varsayarsak optimizasyon problemini gürültülü durum için denklem 3.11'deki gibi tanımlayabiliriz.

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 , \qquad \|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}\|_2 \leq \varepsilon \text{ koşulu altında} \tag{3.11}$$

Bu eksik belirtilmiş l_0 optimizasyon problemi, belirleyici (deterministik) olmayan polinom zaman karmaşıklıklı (NP Hard) bir problemdir ve seyrek bir çözümünü bulmak oldukça zordur [61]. Günümüzde l_0 optimizasyon problemi için etkin ve kesin bir çözüm yöntemi bulunmasa da, çözüme yakınsayan değişik takip algoritmaları dizayn edilmiştir. Bu algoritmalardan başlıcaları aşağıdaki gibidir.

- **Açgözlü (Greedy) Algoritmalar** [62]
  - Eşleştirmeli Takip (Matching Pursuit-MP)
  - Dikey Eşleştirmeli Takip (Orthogonal Matching Pursuit-OMP)
  - En Küçük Kareler-Dikey Eşleştirmeli Takip (Least-Squares OMP)
  - Zayıf Eşleştirmeli-Dikey Eşleştirmeli Takip (Weak Matching OMP)
  - Blok Eşleştirmeli Takip (Block MP)
- **Gevşeme (Relaxation) Tabanlı Algoritmalar** [62]
  - Temel Takip (Basis Pursuit)
  - Dantzig Seçicisi (Dantzig Selector)
- **Hibrit (Hybrid) Algoritmalar** [62]
  - Kademeli-Dikey Eşleştirmeli Takip (Stage-wise OMP)
  - Sıkıştırılmış Örneklemeli-Dikey Eşleştirmeli Takip (Compressive Sampling OMP)
  - Altuzay Takip (Subspace Pursuit)
  - Tekrarlı Sert Eşikleme (Iterative Hard-Thresholding)

Bir sonraki bölümde l_0 optimizasyon probleminin çözümü için literatüde sıkça kullanılan ve tezin ilerleyen bölümlerinde de faydalanacağımız açgözlü algoritmalardan MP ve OMP algoritmalarına değinilecektir.

3.4 l_0 Optimizasyon Problemi İçin Takip Algoritmaları

3.4.1 Açgözlü algoritmalar

Açgözlü algoritmalar bir önceki bölümde de belirtildiği gibi l_0 optimizasyon probleminin çözümü için kullanılan bir yaklaşımdır. Açgözlü algoritmalarla ilgili ilk çalışmalar 1960'lı yıllara kadar dayanmaktadır. Temelde bu algoritmalar iteratif olarak her bir adımda sadece bir adet seyrek vektör elemanını oluşturarak yerel çözümlerden genel çözüme yakınsamayı amaçlar [62, 63]. En çok bilinenleri eşleştirmeli takip ve dikey eşleştirmeli takiptir (Çizelge 3.1 ve Çizelge 3.2).

3.4.1.1 Eşleştirmeli takip

Eşleştirmeli takip (Matching Pursuit-MP), en eski ve en temel takip algoritmasıdır [64]. İlk adımda $\boldsymbol{x}$ sinyalini en iyi temsil eden bir atom bulunur. Bulunan bu atom ile $\boldsymbol{x}$ sinyali arasındaki fark (artık) hesap edilir ve bir sonraki adımda kalan artığı en iyi temsil edecek yeni bir atom bulunur. Bu şekilde iteratif olarak devam eden algoritma artığın belirlenen hata sınırının altına inmesi ile sonlanır. l_0 optimizasyon problemi için kullanılan MP algoritmasının sözde kodu Çizelge 3.1'deki gibidir [62].

Çizelge 3.1 : MP algoritması.

MP Algoritması

- **Temel Problem:** $\boldsymbol{a} = \text{argmin}_{\boldsymbol{a}}\|\boldsymbol{a}\|_0$, $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ koşulu altında
- **Verilen Parametreler:** $\boldsymbol{D}$ sözlüğü, $\boldsymbol{x}$ vektör sinyali ve ε_0 hata eşik değeri
- **Başlangıç Atamaları**
  - İndis, $p = 0$
  - Seyrek vektör, $\boldsymbol{a}_0 = 0$
  - Artık, $\boldsymbol{r}_0 = \boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}_0$
  - $S_0 = Seçilen\ Eleman\{\boldsymbol{a}_0\} = \emptyset$
- **Ana İterasyon:** p'yi bir artır ve aşağıdaki adımları gerçekleştir
  1. **Artığı Bul:** $\boldsymbol{z}_j^* = \boldsymbol{d}_j^T \boldsymbol{r}_{p-1} / \|\boldsymbol{d}_j\|_2^2$ Optimum seçimini kullanarak tüm j'ler için $\varepsilon(j) = \min_{z_j}\|\boldsymbol{d}_j \boldsymbol{z}_j - \boldsymbol{r}_{p-1}\|_2^2$ artık değerini bul.
  2. **Seçilen Elemanı Güncelle:** $\varepsilon(j): \forall\ 1 \le j \le n$ olmak üzere $\varepsilon(j_0) \le \varepsilon(j)$ olan ve minimum artık oluşturan atoma ait j_0 elemanını $S_p = S_{p-1} \cup \{j_0\}$ Seçilen_Eleman kümesine ata.
  3. **Yerel Çözümü Güncelle:** $\boldsymbol{a}_p = \boldsymbol{a}_{p-1}$ ve $\boldsymbol{a}_p(j_0) = \boldsymbol{a}_p(j_0) + \boldsymbol{z}_j^*$ işlemlerini yaparak seyrek vektöre yeni bulunan yerel çözümü ilave et.
  4. **Kalan Artığı Bul:** $\boldsymbol{r}_p = \boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}_p = \boldsymbol{r}_{p-1} - \boldsymbol{z}_{j_0}^* \boldsymbol{d}_{j_0}$
  5. **Karar:** Eğer $\|\boldsymbol{r}_p\|_2 < \varepsilon_0$ ise iterasyonu bitir, değilse ana iterasyona devam et.
- **Çözüm:** p iterasyon sonucunda $\boldsymbol{a}_p$ seyrek vektörü bulunur.

3.4.1.2 Dikey eşleştirmeli takip

l_0 optimizasyon problemi için kullanılan OMP algoritmasının sözde kodu Çizelge 3.2'deki gibidir [62].

Çizelge 3.2 : OMP algoritması.

OMP Algoritması

- **Temel Problem:** $\boldsymbol{a} = \text{argmin}_{\boldsymbol{a}}\|\boldsymbol{a}\|_0$, $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ koşulu altında
- **Verilen Parametreler:** $\boldsymbol{D}$ sözlüğü, $\boldsymbol{x}$ vektör sinyali ve ε_0 hata eşik değeri
- **Başlangıç Atamaları**
  - İndis, $p = 0$
  - Seyrek vektör, $\boldsymbol{a}_0 = 0$
  - Artık, $\boldsymbol{r}_0 = \boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}_0\ atom$
  - $S_0 = Seçilen\ Eleman\{a_0\} = \emptyset$
- **Ana İterasyon:** p'yi bir artır ve aşağıdaki adımları gerçekleştir
  1. **Artığı Bul:** $\boldsymbol{z}_j^* = \boldsymbol{d}_j^T \boldsymbol{r}_{p-1} / \|\boldsymbol{d}_j\|_2^2$ Optimum seçimini kullanarak tüm j'ler için $\varepsilon(j) = \min_{z_j}\|\boldsymbol{d}_j \boldsymbol{z}_j - \boldsymbol{r}_{p-1}\|_2^2$ artık değerini bul.
  2. **Seçilen Elemanı Güncelle:** $\varepsilon(j): \forall\ 1 \le j \le n$ olmak üzere $\varepsilon(j_0) \le \varepsilon(j)$ olan ve minimum artık oluşturan atoma ait j_0 elemanını $S_p = S_{p-1} \cup \{j_0\}$ Seçilen_Eleman kümesine ata.
  3. **Yerel Çözümü Güncelle:** Sadece S_p Seçilen_Eleman kümesini dikkate alarak ve $\boldsymbol{a} = argmin\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}\|_2^2$ denklemini çözerek $\boldsymbol{a}_p$ seyrek vektörüne ait tüm yerel çözümleri güncelle.
  4. **Kalan Artığı Bul:** $\boldsymbol{r}_p = \boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}_p$
  5. **Karar:** Eğer $\|\boldsymbol{r}_p\|_2 < \varepsilon_0$ ise iterasyonu bitir, değilse ana iterasyona devam et.
- **Çözüm:** p iterasyon sonucunda $\boldsymbol{a}_p$ seyrek vektörü bulunur.

MP algoritması ile dikey eşleştirmeli takip (Orthogonal Matching Pursuit-OMP) algoritması oldukça benzer iki algoritmadır [65, 66]. İkisi arasındaki fark MP algoritmasını basitleştirirken başarımını bir miktar düşürür. MP algoritmasının yerel çözümü güncelleme aşamasında, j_0 olarak seçilen yeni eleman S_{p-1} seçilmiş

elemanlar kümesindeki seyrek değerlerde herhangi bir değişiklik yapılmadan seyrek vektöre ilave edilir. Fakat OMP algoritmasında j_0 olarak seçilen yeni eleman S_{p-1} kümesindeki tüm elemanlarla birlikte $\boldsymbol{a} = argmin\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}\|_2^2$ olarak bir en küçük kareler (Least Square) çözüme zorlanır ve $\boldsymbol{a}_p$ seyrek vektöründeki tüm elemanlar güncellenir. Kalan artığın güncellenmesi aşamasında da yeni seyrek vektör dikkate alınarak kalan artık hesap edilir.

3.4.2 Seyrek gösterim ve l_1 normu

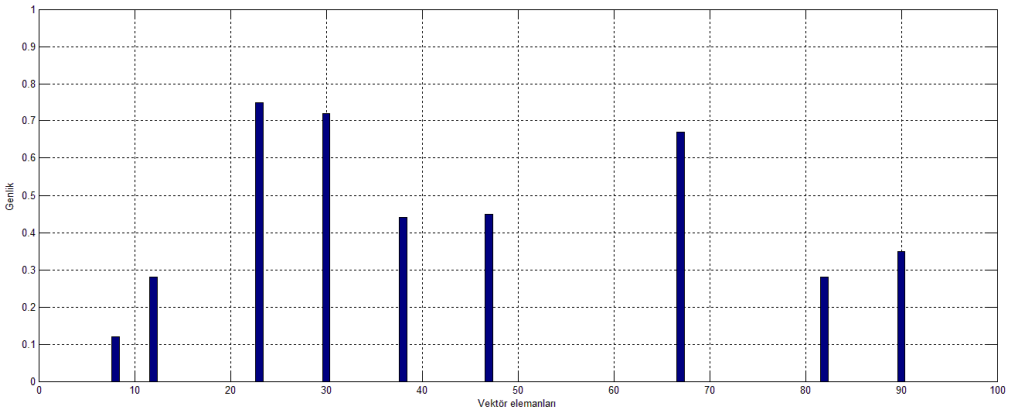
Seyrek temsil ve CS üzerine yapılan son araştırmalarda, eğer $\boldsymbol{a}$ seyreklik vektörü yeteri kadar seyrek bulunabiliyorsa, l_0 optimizasyon probleminin çözümü denklem 3.12'deki l_1 optimizasyon probleminin çözümüne eşit olmaktadır [7].

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\text{argmin}}\|\boldsymbol{a}\|_1, \qquad \boldsymbol{x} = \boldsymbol{D}\boldsymbol{a} \ \text{ koşulu altında} \tag{3.12}$$

Elde edilen bu optimizasyon probleminin çözümü l_0 normuna göre oldukça kolaydır. Örnek olarak bu problem polinom zamanda bilinen bir doğrusal programlama metodu ile kolaylıkla çözülebilir.

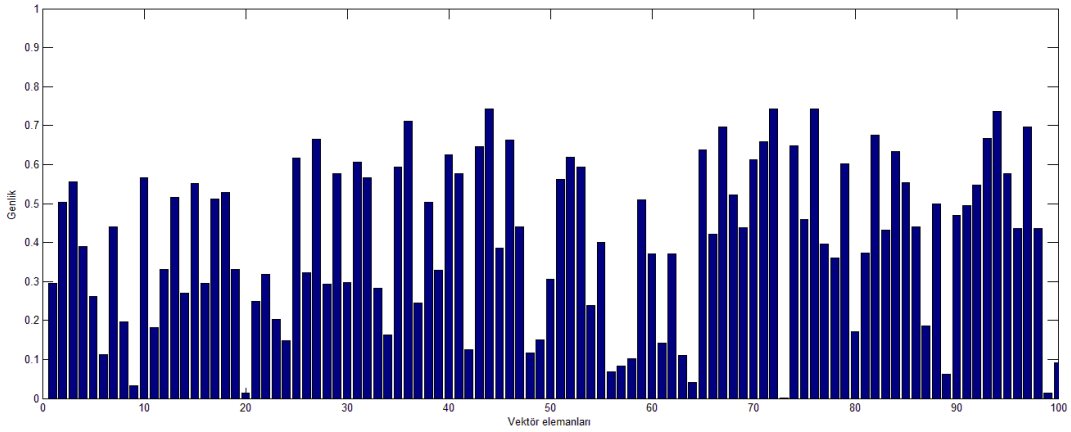
3.5 Sözlük Seçimi

Bölüm 3.3'te bahsedildiği gibi test nesnesi ancak önceden belirlenmiş bir sözlüğün bileşenleri olarak ifade edilebildiğinde seyrek olabilmektedir. Bu nedenle sözlüğün seçimi sinyalin seyrek gösterim başarımında çok önemli bir yere sahiptir. Örneğin sinyali iyi temsil edebilen bir sözlük seçtiğimizi düşünürsek Şekil 3.5'teki gibi seyreklik seviyesi 9 olan bir çözüm elde edebiliriz.



Şekil 3.5 : Seyreklik seviyesi 9 olan vektör.

Fakat aynı sinyali doğru seçilmemiş bir sözlükle temsil etmeye çalıştığımızda Şekil 3.6'daki gibi tüm sözlüğe yayılmış seyrek olmayan bir çözüm elde ederiz.



Şekil 3.6 : Seyrek olmayan vektör.

Bu nedenle seyreklik tabanlı işaret işlemede ortak gereksinim tamamen sinyalle ilintili sözlük atomlarının seçilmesidir. Örneğin ses sinyalinin seyrek bir temsilini elde etmek için, sözlükteki her bir atom bireyin sesiyle ilintili özellikleri barındırmalıdır. Buradan hareketle genel anlamda etkin sözlüğün oluşturulabilmesi iki farklı yolla yapılabilir.

1. Verinin matematiksel modeli temelinde seyrekleştirici sözlük kullanmak
2. Bilinen veri setini kullanmak veya o seti kullanarak en iyi sözlüğü öğrenmek

Birinci yaklaşımda, sinyali en iyi şekilde temsil edecek sözlük Curvelet ve Wavelet gibi matematiksel modellerden oluşturulur. İkinci yaklaşımda ise her bir sınıfa ait sinyaller herhangi bir değişiklik yapılmadan direkt sözlük olarak kullanılabilir veya bu örnekler kullanılarak en iyi sözlük optimum yön metodu (Method of Optimal Directions-MOD) veya tekil değer ayrıştırma (K- times Singular Value Decomposition-K-SVD) gibi algoritmalar ile öğrenilebilir [67].

4. SEYREKLİK TABANLI SINIFLANDIRMA

4.1 Giriş

Seyreklik tabanlı işaret işlemenin kullanıldığı ilk sınıflandırma uygulaması Wright ve diğerleri tarafından geliştirilmiştir [7]. Önerilen bu sınıflandırıcı parametrik olmayan ve sözlük öğrenme gerektirmeyen bir yapıda tasarlanmış ve direkt olarak gelen test resmini sözlük verisi üzerinde seyrek temsilini yaparak sınıflandırma yapmıştır. Bu çalışma ile birlikte seyrek sınıflandırma üzerine yapılan çalışmalar büyük bir artış göstermiştir [68]. Örneğin Xu ve diğerleri [69], l_2 normunu kullanarak iyi bir başarım elde eden seyrek sınıflandırıcı sistemini önermiştir. Yine Deng ve diğerleri [70], seyreklik tabanlı sınıflandırıcının değişik yüz varyasyonları (kamuflaj, yüz ifadeleri, ışık seviyeleri gibi.) için daha iyi performans gösteren genişletilmiş bir versiyonunu önermiştir.

Bu bağlamda tezin bu bölümünde seyreklik tabanlı sınıflandırma algoritması ve doğrusal olmayan sınıflandırmalar için çekirdek tabanlı yaklaşım modeli ele alınmıştır.

4.2 Temel Problem

Gözetimli sınıflandırmadaki temel problem sınıfı bilinmeyen test nesnesini k farklı bilinen sınıf seti içindeki doğru sınıfa atamaktır. Bu sınıflandırmayı seyreklik tabanlı olarak yapmak istediğimizde, öncelikle nesne ve sözlük arasındaki ilişkiyi ifade eden seyreklik vektörünü elde etmemiz gereklidir. Çünkü bir test nesnesine ait seyreklik vektörü, onun elde edilebilmesi için kullanılan sözlük atomlarının belirli bir kısmında yoğunlaşır. Bu durum, test nesnesinin sözlükte yoğunlaştığı atomların doğrusal birleşimi ile elde edildiğini ve benzerliklerinin bir ölçüsü olarak kabul edebileceğimizi gösterir. Buradan hareketle k farklı sınıfın her biri için belirli sayıda atom içeren bir sözlük kümesinin verildiğini düşündüğümüzde, test resmini k farklı bilinen sınıfa ait bu sözlük atomları cinsinden seyrek olarak ifade edebiliriz. Seyreklik tabanlı sınıflandırma işte bu seyreklik vektörünün yoğunlaştığı sözlük

atomları ile test nesnesi arasındaki ilişkinin büyüklüğünü ölçecek bir metot ile sınıflandırma yapan sert, parametrik olmayan ve gözetimli bir sınıflandırma algoritmasıdır.

4.2.1 Sözlüğün oluşturulması

Sınıflandırmada kullanılacak tüm nesnelerin $\mathbb{R}^s$ boyutunda bir vektör olduğu varsayılırsa, her bir sınıfa ait sözlükteki atom sayısı n_i olmak üzere, k farklı sınıfa ait toplam $n = n_1 + n_2 + \cdots + n_k$ adet atom sözlüğün kolonları olacak şekilde $\boldsymbol{D}$ sözlük matrisi oluşturulur. $\boldsymbol{d}_{i,1}$, i. sınıfa ait birinci sözlük atomunu ifade etmek üzere, i. sınıfa ait sözlük denklem 4.1'deki gibi ifade edilir.

$$\boldsymbol{D}_i = \left[\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, .. \boldsymbol{d}_{i,n_i}\right] \in \mathbb{R}^{s \times n_i} \tag{4.1}$$

Bu durumda k adet sınıfa ait sözlüklerin birleşimi ile oluşan $\boldsymbol{D}$ sözlüğü denklem 4.2'deki gibi ifade edilir.

$$\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \dots, \boldsymbol{D}_k] \tag{4.2}$$

4.2.2 Test nesnesinin seyrek olarak gösterilmesi

Seyreklik tabanlı işaret işleme bölümünde ifade ettiğimiz gibi $\boldsymbol{x} \in \mathbb{R}^s$ test nesnesini bilinen $\boldsymbol{D}$ sözlüğünün doğrusal bileşenleri olarak denklem 4.3'teki gibi ifade edebiliriz.

$$\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a} \in \mathbb{R}^s \tag{4.3}$$

Burada $\boldsymbol{a} \in \mathbb{R}^n$ olmak üzere seyrek katsayı vektörüdür. Yukarıdaki denklemin çözümü ile elde edilen bu $\boldsymbol{a}$ seyrek vektörünün i. nesneye ait $\boldsymbol{a}_i = [a_{i,1}\, a_{i,1}, a_{i,2}, \dots, a_{i,n_i}]$ elemanlarının sıfırdan farklı olduğunu varsayarak, $\boldsymbol{a}$ seyrek vektörünü denklem 4.4'teki gibi gösterebiliriz.

$$\boldsymbol{a} = [0, \dots, 0, a_{i,1}, a_{i,2}, \dots, a_{i,n_i}, 0, \dots, 0] \tag{4.4}$$

Sınıflandırma uygulamalarında seyrek katsayı vektörünü elde edebilmek için sözlükteki atom sayısı nesne boyutundan büyük olarak seçilir ($n \gg s$). Bu durumda $\boldsymbol{x} = \boldsymbol{D}\boldsymbol{a}$ eksik belirtilmiş doğrusal denkleminin tekil bir çözümünü bulmak mümkün değildir. Bu denklemin çözümü Bölüm 3'te anlatıldığı gibi ilave bir kısıt altında

mümkün olmaktadır. Kısıt olarak gerçek zamanlı sistemlerdeki gürültü etkisini dikkate alırsak, $\|\boldsymbol{g}\|_2 \leq \varepsilon$ enerjisine sahip $\boldsymbol{g} \in \mathbb{R}^s$ gürültü terimini sisteme ilave edebiliriz. Son durumda optimizasyon problemini denklem 4.5'teki gibi özetleyebiliriz.

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \ , \qquad \|\boldsymbol{D}\boldsymbol{a} - \boldsymbol{x}\|_2 \leq \varepsilon \ \text{ koşulu altında} \tag{4.5}$$

Bu optimizasyon problemi Bölüm 3'te ele alınan MP veya OMP algoritmaları ile kolaylıkla çözülebilmektedir.

4.3 Seyreklik Tabanlı Sınıflandırma Algoritması

Seyreklik tabanlı sınıflandırma (Sparse Representation Classification-SRC) algoritmasına ait sözde kod Çizelge 4.1'deki gibidir.

Çizelge 4.1 : SRC algoritması.

Seyreklik Tabanlı Sınıflandırma Algoritması (SRC)

1. Giriş: $\boldsymbol{D}$ sözlüğü oluşturulur ve $\boldsymbol{x}$ test nesnesi belirlenir.
2. $\boldsymbol{D}$ sözlüğüne ait tüm atomlar birim l_2 normuna göre normalize edilir.
3. l_0 minimizasyon problemi çözülür:

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \, , \qquad \|\boldsymbol{D}\boldsymbol{a} - \boldsymbol{x}\|_2 \leq \varepsilon \ \text{ koşulu altında}$$

4. Her bir sınıf için artık hesabı yapılır:

$$\min_i r_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{D}\delta_i(\boldsymbol{a})\|_2$$

5. Karar verilir:

$$\text{Sınıflandır}(\boldsymbol{x}) = \underset{i}{\operatorname{argmin}}\, r_i(\boldsymbol{x})$$

Denklem 4.5' teki optimizasyon probleminin çözümüyle elde edilen $\boldsymbol{a}$ seyrek vektörünün 0'dan farklı elemanları $\boldsymbol{D}$ sözlüğündeki belirli bir sınıf üzerinde yoğunlaşacaktır. Bunun nedeni $\boldsymbol{x}$ test nesnesinin, büyük oranda kendisi ile benzer olan sınıfa ait atomların doğrusal birleşimi ile ifade edilebilmesidir. Bir başka deyişle seyrekliğin bu özelliği sayesinde $\boldsymbol{x}$ test nesnesi, $\boldsymbol{a}$ seyrek vektörünün 0'dan farklı elemanlarının yoğunlaştığı sınıfa ait olduğuna karar verilebilmektedir.

Fakat bazı durumlarda bu temel yaklaşım yanlış sınıflandırmalara neden olabilmektedir. Bunun yerine kullanılabilecek en iyi yöntemlerden biri, bulunan $\boldsymbol{a}$ seyrek vektöründeki her bir sınıfa ait katsayıları kullanarak $\boldsymbol{x}$ test nesnesini tekrar oluşturmak ve orijinal nesneden çıkartarak, minimum artık oluşturan sınıfa atama yapmaktır. Buradan hareketle $\boldsymbol{a}$ seyrek vektöründeki her bir sınıfa ait katsayıları seçen fonksiyonu $\delta_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ olarak tanımlarsak, $r_i(\boldsymbol{x})$ $i.$ sınıfa ait artığı göstermek üzere sınıflandırma yapılacak sınıf bu artığı minimum yapacak sınıftır. Bu durumda sınıflandırma algoritması denklem 4.6' daki karar kuralı ile ilgili sınıfa karar verecektir.

$$\min_i r_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{D}\delta_i(\boldsymbol{a})\|_2 \tag{4.6}$$

4.4 Seyreklik Tabanlı Sınıflandırmada Çekirdek Yaklaşımı

Seyreklik tabanlı sınıflandırmada ilk çekirdek yaklaşımları l_1 normlu SVM [9] ve çekirdek tabanlı seyrek temsil (Kernel Sparse Representation-KSR) [10] çalışmalarıdır. Her iki yaklaşımda temelde doğrusal olmayan sınıflandırma yapabilmek amacıyla nesneleri yüksek boyutlu öznitelik uzayına dönüştürmüş ve çekirdek yaklaşımını kullanarak sınıflandırma yapmıştır. Özellikle bu çalışmalar içerisinden KSR çalışması teze konu yaklaşımla oldukça ilintilidir. Bu bağlamda bu bölümde çekirdek yaklaşımının seyreklik tabanlı sınıflandırma sistemlerindeki kullanımı ele alınacaktır.

4.4.1 Temel problem

Doğrusal olmayan SVM algoritmasında olduğu gibi $\Phi : \mathbb{R}^s \rightarrow F \subset \mathbb{R}^{\hat{s}}$ dönüşümü nesnenin reel elemanlarını, öznitelik boyutundaki elemanlara çeviren doğrusal olmayan bir haritalama yöntemi olsun. Bu durumda SRC yönteminden farklı olarak çekirdek tabanlı yaklaşımdaki hedefimiz, bu Φ doğrusal olmayan öznitelik boyutuna haritalanmış test nesnesini yine Φ öznitelik uzayına haritalanmış sözlük elemanları ile doğrusal olarak temsil edebilmektir.

Çekirdek yaklaşımının temel problemi SRC algoritmasında ifade edilen parametre ve tanımlamalar ile yapılacaktır. Bu nedenle Çizelge 4.2'de SRC algoritmasında ifade edilen tanımlar özetlenmiştir.

Çizelge 4.2 : Problemin tanımında kullanılacak parametre listesi.

Parametre Adı	Parametre
Her bir nesnenin boyutu	$\mathbb{R}^s$
i. sınıfa ait sözlükteki atom sayısı	n_i
Toplam sınıf sayısı	k
Sözlükteki atom sayısı	$n = n_1 + n_2 + \cdots + n_k$
i. sınıfa ait sözlük	$\boldsymbol{D}_i = \left[\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, .., \boldsymbol{d}_{i,n_i}\right] \in \mathbb{R}^{s \times n_i}$
Sözlük	$\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \dots, \boldsymbol{D}_k] \in \mathbb{R}^{s \times n}$
Test nesnesi	$\boldsymbol{x} \in \mathbb{R}^s$

Φ dönüşümü ile öznitelik boyutuna haritalanmış sözlüğün, öznitelik uzayındaki temsilini $\Phi(\boldsymbol{D})$ olarak ifade edelim. Bu durumda sözlüğümüz denklem 4.7'deki gibi her bir sınıfa ait sözlüğün öznitelik uzayına dönüştürülmüş hallerinin birleşimi olacaktır.

$$\Phi(\boldsymbol{D}) = [\Phi(\boldsymbol{D}_1), \Phi(\boldsymbol{D}_2), \dots, \Phi(\boldsymbol{D}_k)] \in \mathbb{R}^{\hat{s} \times n} \tag{4.7}$$

Benzer şekilde Φ dönüşümü ile test nesnesinin öznitelik uzayındaki temsilini $\Phi(\boldsymbol{x})$ olarak ifade edebiliriz. Nihai olarak öznitelik uzayındaki $\Phi(\boldsymbol{x})$ test nesnesi $\Phi(\boldsymbol{D})$ sözlüğünün doğrusal birleşimi olarak denklem 4.8'deki gibi ifade edilebiliriz.

$$\Phi(\boldsymbol{x}) = \Phi(\boldsymbol{D})\boldsymbol{a} \in \mathbb{R}^{\hat{s}} \tag{4.8}$$

Bu denklemin çözümü için $\boldsymbol{g} \in \mathbb{R}^{\hat{s}}$ olmak üzere $\|\boldsymbol{g}\|_2 \leq \varepsilon$ enerjisine sahip gürültü terimini dikkate alırsak doğrusal olmayan sınıflandırma için optimizasyon problemini denklem 4.9'daki gibi ifade edebiliriz.

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \ , \qquad \|\Phi(\boldsymbol{D})\boldsymbol{a} - \Phi(\boldsymbol{x})\|_2 \leq \varepsilon \ \text{ koşulu altında} \tag{4.9}$$

Φ dönüşümünün, giriş uzayında doğrusal olarak sınıflandırılamayan nesneleri çok yüksek boyutlu uzaya çevirmesinden dolayı $s \ll \hat{s}$ olmak zorundadır. Bu nedenle Φ uzayında çalışmak büyük bir işlemsel yük ve zaman gerektireceğinden, SVM'de bahsedildiği gibi çekirdek yönteminin kullanılması uygun olacaktır. Bu durumda nesnelerin iç çarpımları yoluyla öznitelik uzayındaki eşdeğerini veren çekirdek fonksiyonu $k(\boldsymbol{i}, \boldsymbol{j}) = \langle \Phi(\boldsymbol{i}), \Phi(\boldsymbol{j}) \rangle = \Phi(\boldsymbol{i})^T \Phi(\boldsymbol{j})$ şeklinde tanımlanabilir. Bu durumda denklem 4.8'in her iki tarafı $\Phi(\boldsymbol{D})^T$ ile genişletilerek denklem 4.10 ve 4.11'deki çekirdek fonksiyonlarına geçiş sağlanır.

$$\Phi(\boldsymbol{D})^T\Phi(\boldsymbol{x}) = \Phi(\boldsymbol{D})^T\Phi(D)\boldsymbol{a} \tag{4.10}$$

$$k(\boldsymbol{D},\boldsymbol{x}) = \boldsymbol{S}\boldsymbol{a} \tag{4.11}$$

Yukarıdaki denklemde $\Phi(\boldsymbol{D})^T\Phi(\boldsymbol{x}) = \langle\Phi(\boldsymbol{D}),\Phi(\boldsymbol{x})\rangle = k(\boldsymbol{D},\boldsymbol{x}) \in \mathbb{R}^{1\times n}$ boyutlu bir vektör ve $\boldsymbol{S} = \Phi(\boldsymbol{D})^T\Phi(\boldsymbol{D}) = \langle\Phi(\boldsymbol{D}),\Phi(\boldsymbol{D})\rangle = k(\boldsymbol{D},\boldsymbol{D}) \in \mathbb{R}^{n\times n}$ simetrik ve pozitif tanımlı gram matrisdir. Bu durumda çekirdek denklemlerinin açık ifadeleri denklem 4.12 ve denklem 4.13'teki gibi olacaktır.

$$\boldsymbol{S}_{i,j} = \langle\Phi(\boldsymbol{d}_i),\Phi(\boldsymbol{d}_j)\rangle = \Phi(\boldsymbol{d}_i)^T\Phi(\boldsymbol{d}_j) = k(\boldsymbol{d}_i,\boldsymbol{d}_j) \tag{4.12}$$

$$k(\boldsymbol{D},\boldsymbol{x}) = [k(\boldsymbol{d}_1,\boldsymbol{x}),k(\boldsymbol{d}_2,\boldsymbol{x}),..,k(\boldsymbol{d}_n,\boldsymbol{x})] \tag{4.13}$$

Çekirdek fonksiyonları ile elde edilmiş bu denklemin çözümü için $\boldsymbol{g} \in \mathbb{R}^n$ olmak üzere $\|\boldsymbol{g}\|_2 \leq \varepsilon$ enerjisine sahip gürültü terimini dikkate alırsak optimizasyon problemi denklem 4.14'teki gibi ifade edilebilir.

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}}\|\boldsymbol{a}\|_0 \ , \qquad \|\boldsymbol{S}\boldsymbol{a} - k(\boldsymbol{D},\boldsymbol{x})\|_2 \leq \varepsilon \ \text{ koşulu altında} \tag{4.14}$$

Doğrusal olmayan seyreklik tabanlı sınıflandırma için yukarıdaki optimizasyon probleminin çözümü [10] ve [71] makalelerinde olduğu gibi 2 farklı yolla yapılabilir. Bunlardan biri direkt olarak l_1 normu ile seyrek çözümü elde etmek, diğeri ise çekirdek OMP (Kernel-OMP) kullanarak seyrek çözümü elde etmektir. Bölüm 4.4.2'de teze konu benzetimlerde kullanılan çekirdek OMP algoritması yer almaktadır.

4.4.2 Çekirdek OMP

$\boldsymbol{D}$ sözlüğünün daha önceden belirlendiğini varsayarak, $\boldsymbol{a}$ seyrek vektörünü bulabilmek için OMP algoritmasını kullanabiliriz. Fakat burada ele aldığımız çekirdek yaklaşımı nedeniyle bilinen OMP algoritmasında bazı değişiklikler yapılması gereklidir.

OMP algoritmasını incelediğimizde temelde birçok iç çarpım ifadesi içerdiğini görebiliriz. Buradan hareketle OMP algoritmasındaki ifadelerin iç çarpımlar cinsinden ifade edilmesi ile çekirdek fonksiyonlarına geçiş sağlanır ve çekirdek OMP algoritması elde edilir.

Özetle denklem 4.14'ün l_0 normu ile seyrek çözümünü elde eden çekirdek OMP algoritmasına ait temsili kod Çizelge 4.3'teki gibidir [71].

Çizelge 4.3 : Çekirdek OMP algoritması.

Çekirdek OMP Algoritması

- **Temel Problem:** $\boldsymbol{a} = \text{argmin}_{\boldsymbol{a}}\|\boldsymbol{a}\|_0$, $\Phi(\boldsymbol{x}) = \Phi(\boldsymbol{D})\boldsymbol{a}$ koşulu altında
- **Verilen Parametreler:** $\boldsymbol{S}$ matrisi, $k(\boldsymbol{D}, \boldsymbol{x})$ vektör sinyali, ε_0 hata eşik değeri
- **Başlangıç Atamaları**
  - İndis, $p = 0$
  - Seyrek vektör $\boldsymbol{a}_0 = 0$
  - İki vektörün iç çarpımı, $\Omega_p = \langle \Phi(\boldsymbol{d}_p), \Phi(\boldsymbol{x}) \rangle$
  - $\delta_0 = Seçilen\ Eleman\{\boldsymbol{a}_0\} = \emptyset$
- **Ana İterasyon:** p'yi bir artır ve aşağıdaki adımları gerçekleştir.
  1. **Artığı Bul:** tüm j'ler için $\Delta_p = \text{argmax}(\Omega_j - \boldsymbol{a}_p^T \boldsymbol{S}[j, \delta_p])$ iç çarpım değerini maksimum yapan atomu bul. $j = 1,2,..,n$
  2. **Seçilen Elemanı Güncelle:** Minimum artık oluşturan atoma ait Δ_p elemanını $\delta_p = [\delta_p \ \ \Delta_p]$ Seçilen_Eleman kümesine ata.
  3. δ_p seçilen elemanlara ait $\boldsymbol{S}_p = \boldsymbol{S}[\delta_p, \delta_p]$ ve $\Omega_p = \Omega[\delta_p]$ altkümelerini oluştur.
  4. **Yerel Çözümü Güncelle:** Sadece δ_p Seçilen_Eleman kümesindeki elemanları kullanarak $\boldsymbol{a}_p = \boldsymbol{S}_p^{-1}\Omega_p$ denklemini çöz ve $\boldsymbol{a}_p$ seyrek vektörüne ait tüm yerel çözümleri güncelle.
  5. **Kalan Artığı Bul:** $\|\boldsymbol{r}_p\|_2^2 = k(\boldsymbol{x}, \boldsymbol{x}) - \Omega_p^T \boldsymbol{a}_p$
  6. **Karar:** Eğer $\|\boldsymbol{r}_p\|_2^2 < \varepsilon_0^2$ ise iterasyonu bitir, değilse ana iterasyona devam et.
- **Çözüm:** p iterasyon sonucunda $\boldsymbol{a}_p$ seyrek vektörü bulunur.

4.4.3 Doğrusal olmayan seyreklik tabanlı sınıflandırma algoritması

Doğrusal olmayan seyreklik tabanlı sınıflandırma algoritması (Kernel Sparse Representation Classification-K-SRC), SRC ile oldukça benzerdir. İki algoritma arasındaki tek fark SRC algoritmasının 3. adımındaki optimizasyon probleminin K-

SRC algoritmasında çekirdek fonksiyonlarına sahip olmasıdır. Bu bağlamda K-SRC algoritmasına ait sözde kod Çizelge 4.4'teki gibidir.

Çizelge 4.4 : K-SRC algoritması.

Doğrusal Olmayan Seyreklik Tabanlı Sınıflandırma Algoritması (K-SRC)

1. Giriş: $\boldsymbol{D}$ sözlüğü oluşturulur, $k(\boldsymbol{i},\boldsymbol{j})$ çekirdek fonksiyonu seçimi yapılır ve $\boldsymbol{x}$ test nesnesi belirlenir.
2. $\Phi(\boldsymbol{D})$ ve $\Phi(\boldsymbol{x})$'e ait tüm atomlar birim l_2 normuna göre normalize edilir.
3. l_0 minimizasyon problemi çözülür:

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \ , \qquad \|\Phi(\boldsymbol{D})\boldsymbol{a} - \Phi(\boldsymbol{x})\|_2 \leq \varepsilon \ \text{ koşulu altında}$$

veya

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \ , \qquad \|\boldsymbol{S}\boldsymbol{a} - k(\boldsymbol{D},\boldsymbol{x})\|_2 \leq \varepsilon \ \text{ koşulu altında}$$

4. Her bir sınıf için artık hesabı yapılır:

$$\min_i r_i(\boldsymbol{x}) = \|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{D})\delta_i(\boldsymbol{a})\|_2$$

veya

$$\min_i r_i(\boldsymbol{x}) = \|k(\boldsymbol{D},\boldsymbol{x}) - \boldsymbol{S}\delta_i(\boldsymbol{a})\|_2$$

5. Karar verilir:

$$\text{Sınıflandır}(\boldsymbol{x}) = \underset{i}{\operatorname{argmin}}\, r_i(\boldsymbol{x})$$

5. DENEYSEL BENZETİM SONUÇLARI

Bu bölümün amacı, tezde ele alınan seyrek sınıflandırma yöntemlerinin deneysel sonuçlar üzerinden değerlendirmesini yapmak ve diğer klasik yöntemlerle karşılaştırmaktır. Bu kapsamda bu bölümde gerçek hayatta geniş bir kullanım alanına sahip yüz tanıma ve rakam tanıma sistemleri için seyrek sınıflandırma algoritmalarının karşılaştırmalı performans testleri ele alınacaktır.

Bu kapsamda SRC ve K-SRC algoritmaları sınıflandırma problemlerinde sıklıkla kullanılan NN ve Doğrusal SVM algoritmaları ile karşılaştırılmıştır. Her iki sınıflandırma problemi için SRC, K-SRC, NN ve Doğrusal SVM benzetimleri MATLAB ortamında hazırlanmıştır. NN algoritması MATLAB'e ait hazır "knnclassify" metodu ile gerçekleştirilmiştir. Doğrusal SVM benzetimindeki optimizasyon problemleri MATLAB ortamında ve LIBSVM [72] yazılım paketi ile çözülmüştür. SRC ve K-SRC için optimizasyon problemleri sırasıyla T_0 seyreklik seviyesi kısıtı ile çözüm bulan OMP ve çekirdek OMP algoritmaları ile çözülmüştür [73].

Tüm deneysel sonuçlar 4 çekirdekli 2.4Ghz işlemci hızına sahip 8gb bellek kapasiteli Windows 8.1 64 bit işletim sistemine sahip ve MATLAB 8.0.0.783 (R2012b) yüklü bir bilgisayar ile elde edilmiştir.

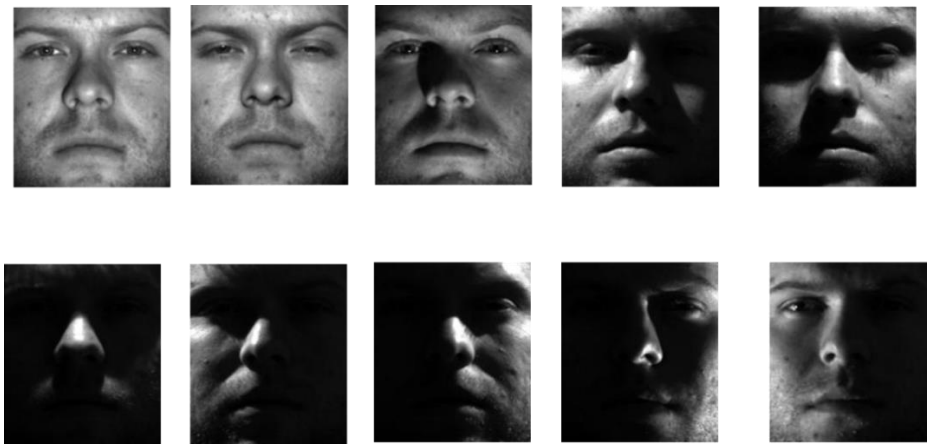
5.1 Yüz Tanıma

Bu bölümde yüz tanıma sistemine ait gerçekleştirilen deneysel benzetim sonuçları ele alınacaktır. Veri seti olarak Genişletilmiş (Extended) Yale B yüz veri seti kullanılmıştır. Bu veri seti her bir resimde tek bir birey içeren, gerekli ön işlemleri yapılmış ve sadece yüz bölgesini içeren bir veri setidir [74].

5.1.1 Genişletilmiş Yale B veri seti

Bu veri seti 38 bireye ait toplam 2432 yüz resmi içermektedir. Her bir yüz için değişik laboratuvar ışıklandırma seviyelerinde ve değişik yüz duruşlarında toplam 64

resim bulunmaktadır (Şekil 5.1). Her resim 192×168 piksel boyutunda ve gri seviyelidir [74].



Şekil 5.1 : Genişletilmiş Yale B veri setine ait örnek resimler.

5.1.2 SRC ve K-SRC algoritmalarının yüz tanımaya uygulanması

SRC ve K-SRC algoritmalarının başarımı ilk olarak yüz tanıma sistemleri için ele alınmıştır. Bu problem için oldukça bilinen ve bir önceki bölümde ele alınan Genişletilmiş Yale B veri seti kullanılmıştır. Bu veri seti yüz tanıma uygulamaları için laboratuvar ortamında hazırlanmış gürültüden temizlenmiş, gerekli ön işlemleri yapılmış ve sadece yüz bölgesini içeren sınıflandırma için oldukça uygun bir veri setidir. Bu nedenle gerçekleştireceğimiz deneysel sonuçlarda bu veri setindeki resimler sadece farklı öznitelik boyutlarına Downsampled edilerek kullanılmıştır.

Deneysel sonuçlar için veri seti sözlük ve test olarak ikiye ayrılmıştır. Bunun için veri seti içindeki her bir bireye ait 64 resmin yarısı rastgele test için (32 resim), diğer yarısı (32 resim) ise sözlük için ayrılmıştır. Nihai olarak $N_{sözlük}$=1216 ve N_{test}=1216 adet resimden oluşmuştur.

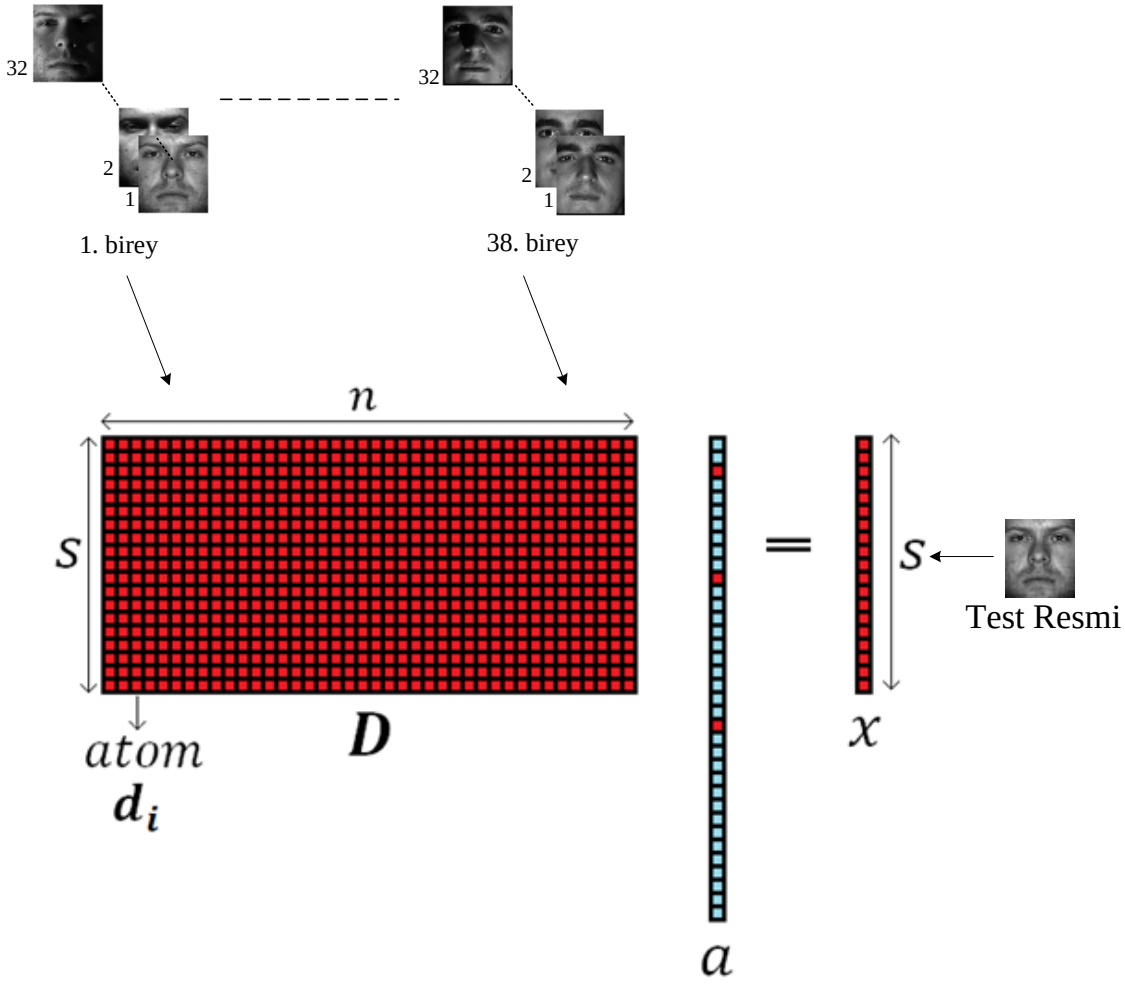
Bu durumda SRC ve K-SRC bölümünde ifade edildiği gibi test resmi vektör haline getirilmiş $\boldsymbol{x} \in \mathbb{R}^s$ olarak tanımlanır. Benzer şekilde i. bireye ait sözlük $n_i = 32$ olmak üzere $\boldsymbol{D}_i = [\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, .., \boldsymbol{d}_{i,32}] \in \mathbb{R}^{s \times 32}$ şeklinde tanımlanır. Sisteme ait tüm sözlük ise 38 birey için $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \dots, \boldsymbol{D}_{38}]$ şeklinde ifade edilir.

5.1.2.1 Dağıtık ve işbirlikçi sınıflandırma yaklaşımı

Seyrek sınıflandırma algoritmaları için Van Nguyen ve diğerleri iki farklı sınıflandırma yaklaşımı önermiştir [66]. Birinci yaklaşım test nesnesini tüm sözlüğün

doğrusal birleşimi ile seyrek ifade etmektir. Bu yaklaşım tüm sözlük kullanıldığı için işbirlikçi sınıflandırma olarak isimlendirilmiştir. SRC ve K-SRC algoritmaları mevcut yapısı ile işbirlikçi sınıflandırma yaklaşımı ile sınıflandırma yapmaktadır. Bu nedenle işbirlikçi sınıflandırma için denklem 5.1'teki optimizasyon probleminin her bir test resmi için bir kez çözülmesi yeterlidir (Şekil 5.2).

$$\boldsymbol{a} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \|\boldsymbol{a}\|_0 \ , \qquad \|\boldsymbol{D}\boldsymbol{a} - \boldsymbol{x}\|_2 \leq \varepsilon \quad \text{koşulu altında} \tag{5.1}$$

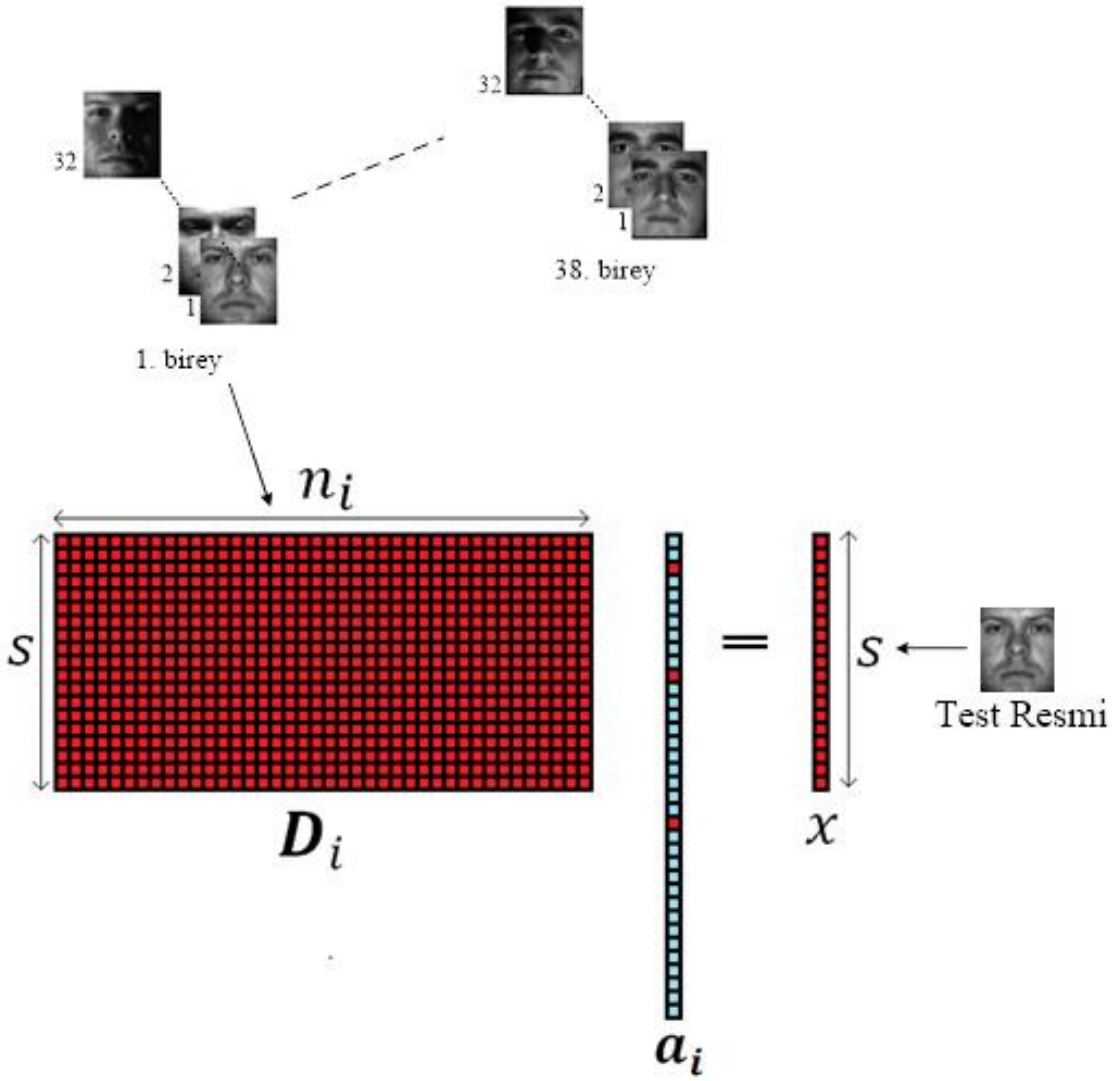


Şekil 5.2 : İşbirlikçi sınıflandırma için sözlük yapısı ve temel problem.

Fakat ikinci yaklaşım ise test nesnesini her bir sınıfın sözlüğü ile ayrı ayrı seyrek ifade etmektir. Bu yaklaşım sınıf bazlı olduğu için dağıtık sınıflandırma olarak isimlendirilmiştir. Dağıtık yaklaşımın işbirlikçi yaklaşımdan tek farkı sadece seyrek vektörün her bir sınıf için ayrı ayrı elde edilmesidir. Diğer bir deyişle dağıtık sınıflandırmanın tek farkı SRC ve K-SRC algoritmalarının 3. adımındaki

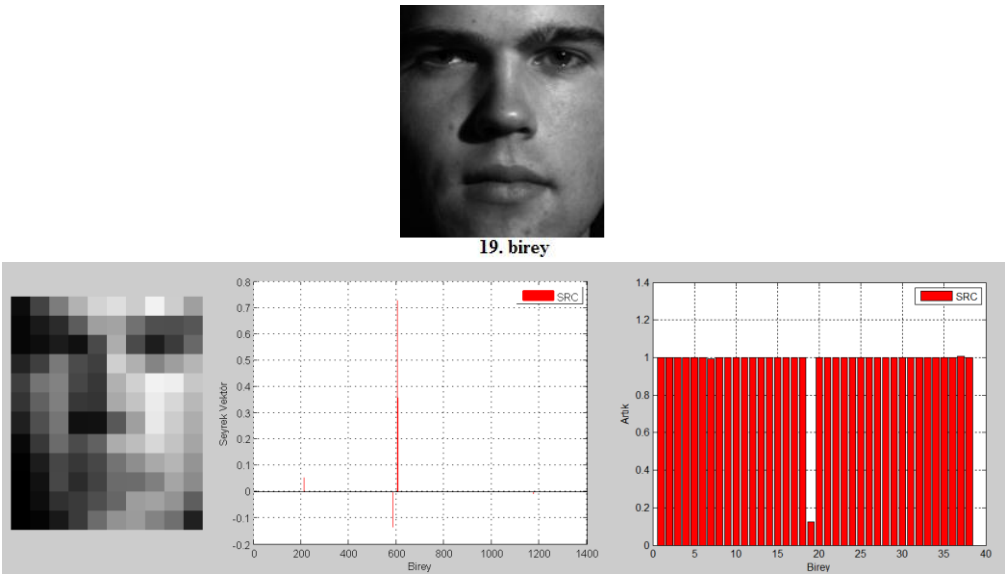
optimizasyon probleminin toplam sınıf sayısı kadar çözülmesidir. Bu durumda, i sınıf numarasını göstermek üzere k sınıf sayısı kadar çözülecek optimizasyon problemi denklem 5.2'deki gibidir. Bu denklemde $\boldsymbol{a_i} \in \mathbb{R}^n$ olmak üzere i. sınıfa ait seyrek katsayı vektörünü göstermektedir (Şekil 5.3).

$$\boldsymbol{a_i} = \underset{\boldsymbol{a_i}}{\operatorname{argmin}} \|\boldsymbol{a_i}\|_0 \ , \qquad \|\boldsymbol{D_i a_i} - \boldsymbol{x}\|_2 \leq \varepsilon \ \text{ koşulu altında} \tag{5.2}$$



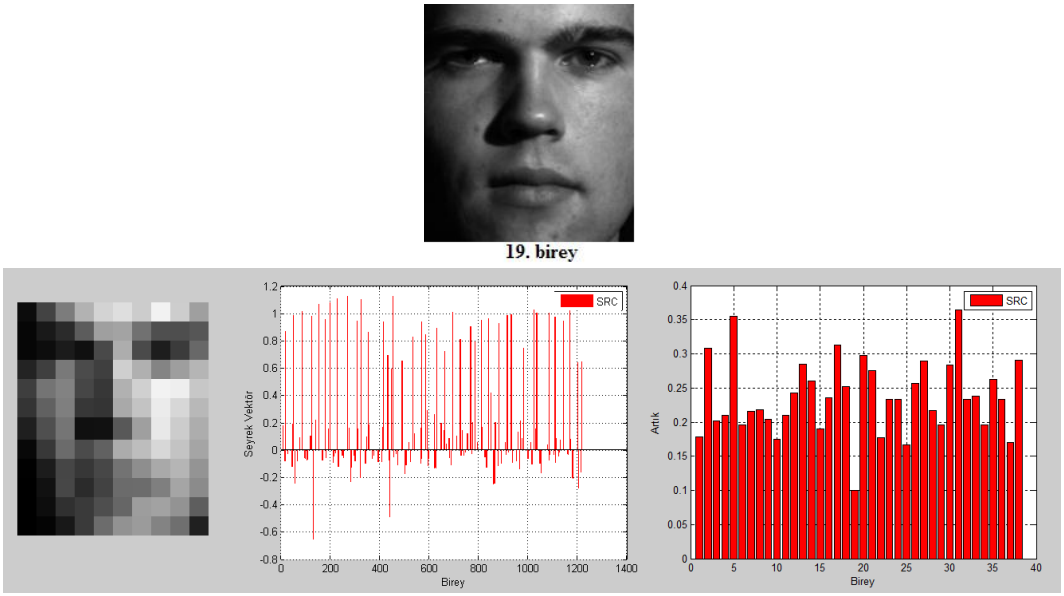
Şekil 5.3 : Dağıtık sınıflandırma için sözlük yapısı ve temel problem.

İşbirlikçi ve dağıtık sınıflandırmanın farkını görebilmek için SRC algoritması, Yale B veri setinin 19. bireyi için koşturulmuş ve Şekil 5.4 ve Şekil 5.5'teki seyrek vektör ve artık elde edilmiştir. Her iki yaklaşım için artık grafiği incelendiğinde minimum artığın 19. bireye ait olduğu ve doğru sınıflandırma yapıldığı görülebilmektedir.



Şekil 5.4 : Veri setinin 19. bireyi için işbirlikçi sınıflandırma yapıldığında elde edilen seyrek vektör ve artık grafiği (Öznitelik boyutu: 120, T_0 = 5).

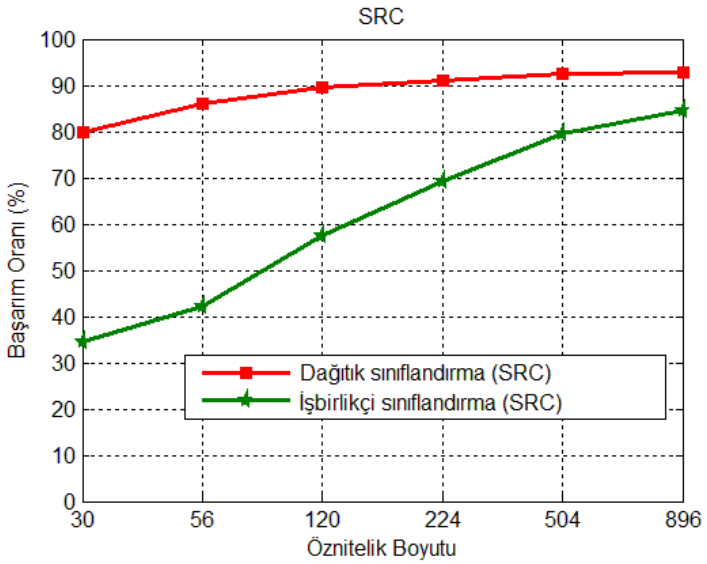
Bununla birlikte işbirlikçi sınıflandırma için seyrek vektör grafiğine bakıldığında seyrek vektörün doğru birey üzerinde yoğunlaştığı rahatlıkla görülebilmektedir. Fakat dağıtık sınıflandırma için seyrek vektör grafiğine bakıldığında seyrek vektör tüm bireyler üzerine yayılmıştır. Bunun nedeni dağıtık sınıflandırmada her bir sınıf için seyrek vektörün ayrı ayrı elde edilmiş olmasıdır.



Şekil 5.5 : Veri setinin 19. bireyi için dağıtık sınıflandırma yapıldığında elde edilen seyrek vektör ve artık grafiği (Öznitelik boyutu: 120, T_0 = 5).

Her iki yaklaşımın yüz tanıma için başarım grafiği Şekil 5.6'da verilmiştir. Grafikten de görülebileceği gibi yüz tanıma için dağıtık sınıflandırma tüm öznitelik

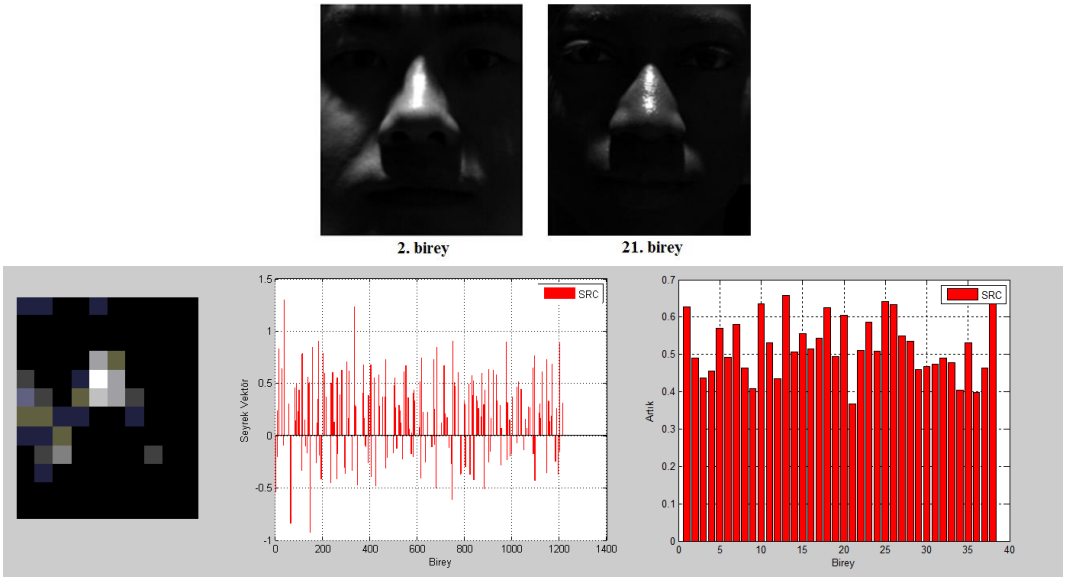
boyutlarında daha yüksek başarım göstermiştir. Bu nedenle bu noktadan sonraki benzetimlerde SRC ve K-SRC algoritmalarında dağıtık yaklaşım kullanılmıştır.



Şekil 5.6 : Dağıtık ve işbirlikçi sınıflandırma için başarımlar (Öznitelik boyutu: 120, T_0 = 5).

5.1.2.2 Hatalı sınıflandırma örneği

Şekil 5.7'de SRC algoritmasının yaptığı hatalı bir sınıflandırma örneği verilmiştir. Grafikten de görülebileceği yüzün oldukça karanlık olmasından dolayı 2. bireye ait olan test resmi sınıflandırıcı tarafından 21. bireye ait olarak bulunmuştur.



Şekil 5.7 : Dağıtık SRC sınıflandırma için hatalı yüz sınıflandırması (Öznitelik boyutu: 120, T_0 = 5).

5.1.3 Benzetim sonuçları

SRC ve K-SRC algoritmalarına ait deneysel sonuçlar MATLAB ortamında geliştirilmiş NN ve Doğrusal SVM algoritmaları ile karşılaştırılmıştır. Tüm algoritmalar tamamen aynı test ve sözlük resimleri üzerinde koşturulmuştur. SRC algoritması için seyreklik seviyesi $T_0 = 5$ ve NN algoritması için k değeri 3 olarak alınmıştır [75].

Ayrıca tüm algoritmalar 10 farklı sözlük ve test seti üzerinde 10 sefer koşturulmuş ve başarım ortalamaları dikkate alınmıştır. Bu sayede seçilen sözlük ve test resimlerinin başarım üzerinde yaratabileceği etki minimize edilmiştir.

i. iterasyondaki doğru sınıfa atama yapılan test resmi sayısı $N^i_{doğru}$ olmak üzere benzetimlerdeki sınıflandırma başarımı denklem 5.3'teki gibi hesaplanmıştır.

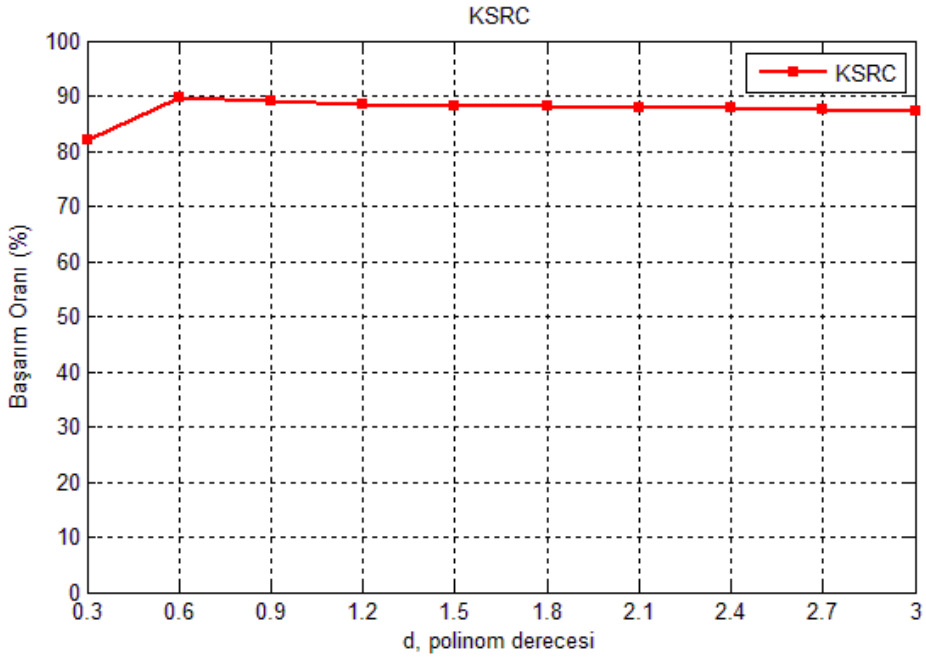
$$Sınıflandırma\ başarımı = 100 \times \frac{(N^1_{doğru} + N^2_{doğru} + \cdots + N^{10}_{doğru})}{10 \times N_{test}} \qquad (5.3)$$

5.1.3.1 Değişik çekirdek fonksiyonları için K-SRC sınıflandırıcısının başarımı

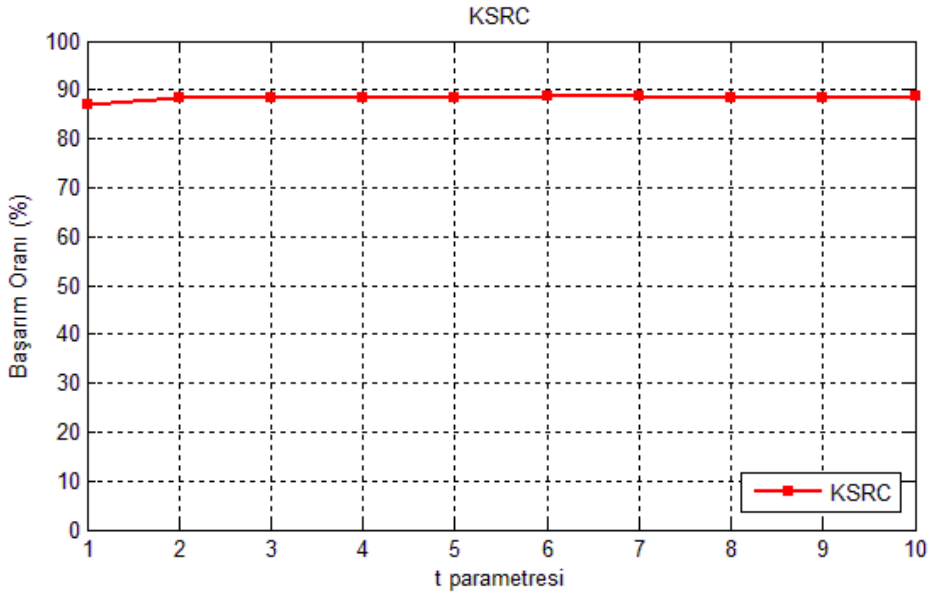
Bu benzetimde K-SRC algoritmasında kullanılan çekirdek fonksiyonlarının Yale B yüz veri seti üzerindeki başarım etkisi araştırılmıştır. Bu deney için $T_0 = 5$ ve öznitelik boyutu 120 alınmıştır. K-SRC bölümünde anlatıldığı gibi her çekirdek fonksiyonu farklı uygulamalar için oldukça farklı sonuçlar verebilmektedir. Bu nedenle bu benzetimlerde en yüksek başarımı veren çekirdek fonksiyonu araştırılmıştır. Benzetimlerde Gauss ve polinom çekirdek fonksiyonları kullanılmıştır.

Polinom çekirdek fonksiyonu ile gerçekleştirilen başarım testlerinde K-SRC, Şekil 5.8'de görülebileceği gibi maksimum başarımı 0,6 polinom derecesinde yakalamıştır. Şekil 5.9'da Gauss bir çekirdek fonksiyonu için başarım grafiği verilmiştir. Grafikten de görülebileceği gibi t parametresi 7 seçildiğinde maksimum başarım elde edilmiştir.

Her iki çekirdek fonksiyonunu birbiri ile karşılaştırdığımızda en yüksek başarımın derecesi 0.6 seçilen polinom fonksiyon olduğunu görebiliriz. Bu nedenle bu aşamadan sonraki tüm benzetimlerde K-SRC algoritması 0.6 derecesine sahip $k(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}^T \boldsymbol{y})^{0.6}$ polinom çekirdek fonksiyonu ile koşturulmuştur.



Şekil 5.8 : Polinom çekirdek fonksiyonu için değişik polinom derecelerindeki K-SRC başarım grafiği.

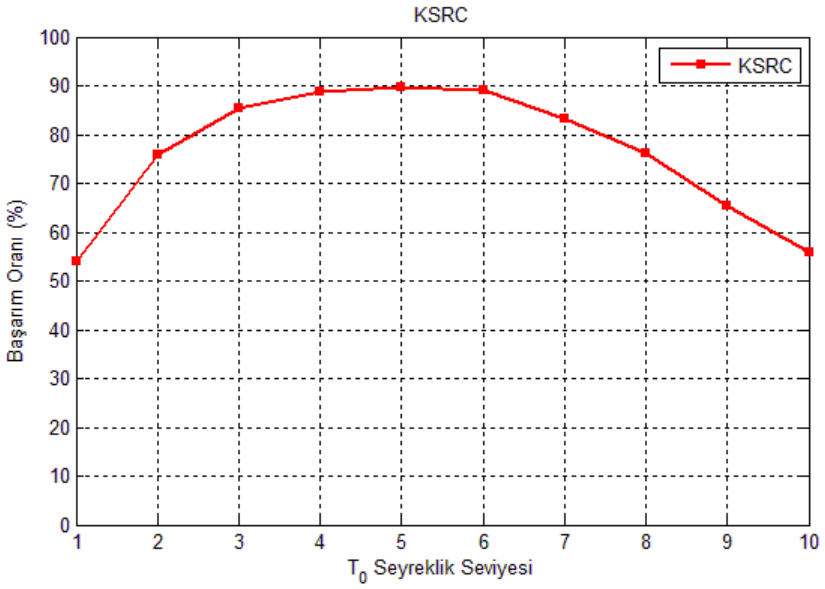


Şekil 5.9 : Gauss çekirdek fonksiyonu için değişik t parametrelerinde K-SRC başarım grafiği.

5.1.3.2 Değişik seyreklik seviyelerinde K-SRC sınıflandırıcısının başarımı

K-SRC sınıflandırıcısının başarımı büyük oranda test nesnesini en iyi temsil edecek seyrek vektör değerlerine bağlıdır. Eğer T_0 seyreklik seviyesi büyük seçilirse seyrek vektör birçok sınıfa yakınsayacak, çok küçük seçilirse de test nesnesini iyi bir şekilde temsil edecek yeterli seyrek elemana ulaşamayacaktır. Bu nedenle T_0 seyreklik

seviyesi belirli bir karar değerinde olmalıdır. Bu benzetimde değişik T_0 seyreklik seviyeleri için K-SRC algoritmasının başarımı ölçülmüştür. K-SRC algoritması için elde edilen bu veriler bir önceki bölümde bahsedildiği gibi polinom çekirdek fonksiyonu (0.6 dereceli) ile elde edilmiştir (Şekil 5.10).



Şekil 5.10 : Değişik T_0 seyreklik seviyeleri için K-SRC başarım grafiği.

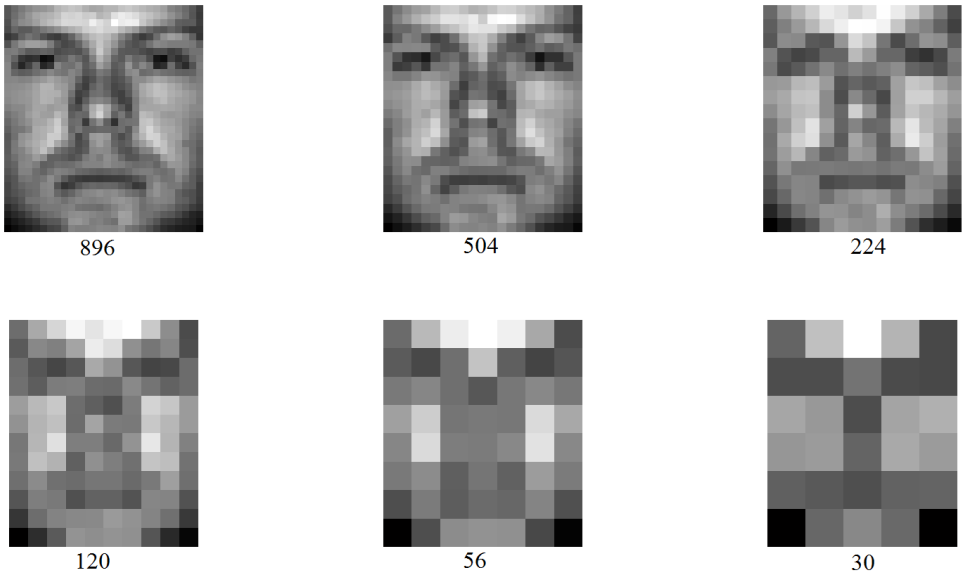
Şekil 5.10'dan görülebileceği gibi K-SRC en yüksek başarıma $T_0 = 5$ olduğunda erişebilmiştir. Bu nedenle bundan sonra gerçekleştirilen K-SRC benzetimlerinde $T_0 = 5$ alınmıştır.

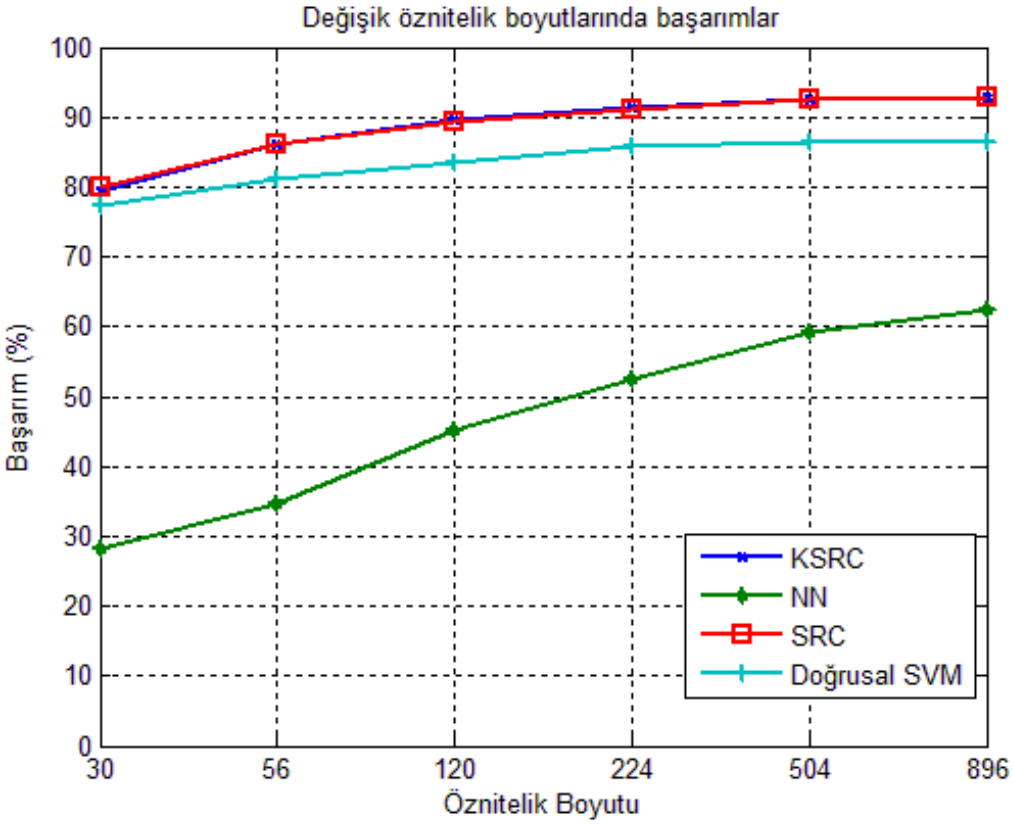
5.1.3.3 Gürültüsüz durum

Bu benzetimde gürültüsüz test resimleri için sınıflandırıcıların performansı incelenmiştir. Bu amaçla tüm yüz veri seti gürültüsüz haliyle sözlük ve test olarak kullanılmıştır. Deneysel sonuçlar 30, 56, 120, 224, 504 ve 896 öznitelik boyutları için ayrı ayrı elde edilmiştir.

Şekil 5.11'de görülebileceği gibi gürültüsüz test resimleri için değişik öznitelik boyutlarında K-SRC ve SRC, diğer sınıflandırıcılardan daha yüksek başarım elde etmiştir. K-SRC ise tüm öznitelik boyutlarında SRC algoritmasından yaklaşık %0.1-%0.2 daha yüksek başarım göstermiştir. Buna karşın SVM kabul edilebilir bir tanıma başarımı elde ederken, NN yüksek öznitelik boyutlarında dahi ancak %63 tanıma başarımına ulaşabilmiştir.

Bu benzetimler için algoritmaların çalışma süreleri Çizelge 5.1' de verilmiştir.





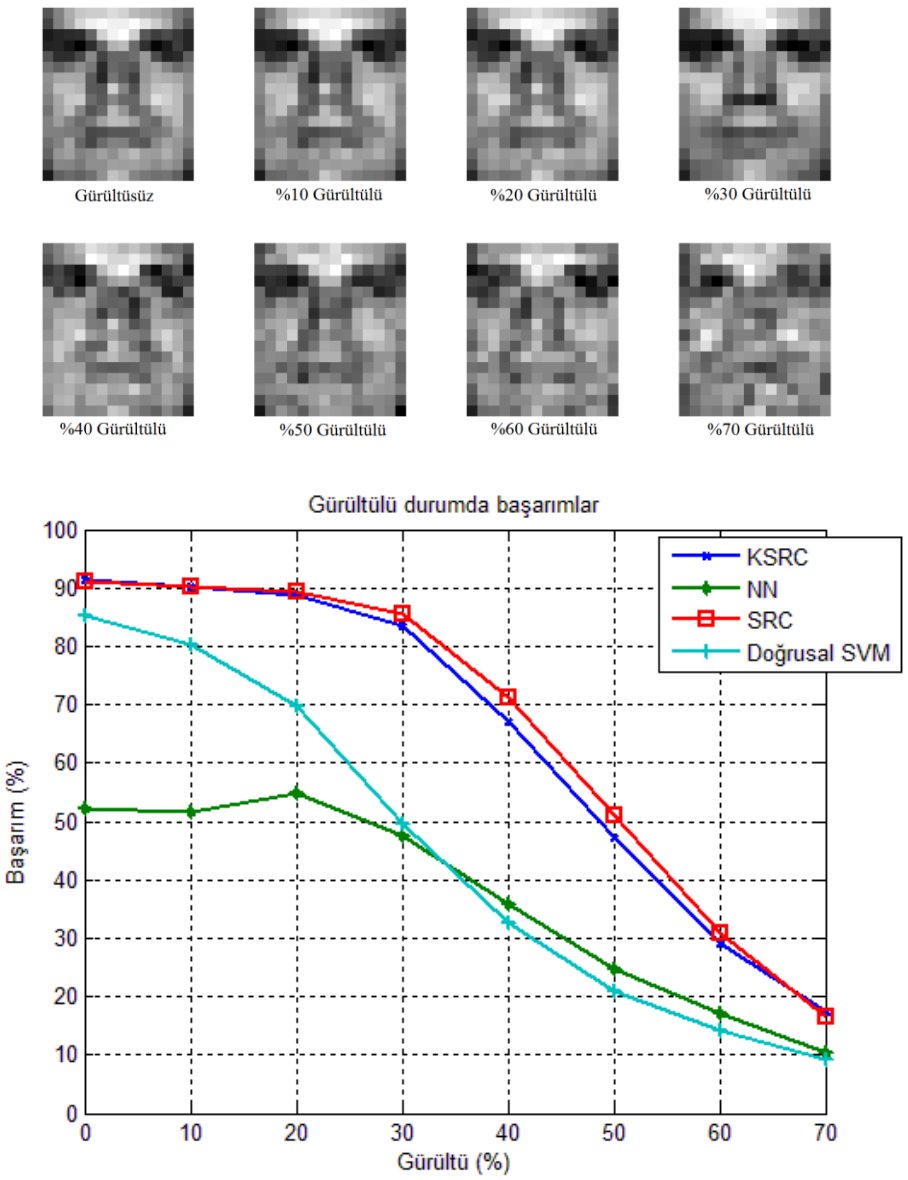
Şekil 5.11 : Gürültüsüz test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri.

Çizelge 5.1 : Gürültüsüz durumda algoritmaların çalışma süreleri (Yale B).

Algoritma	Süre (Sn)
KSRC	256,8
SRC	20,2
NN	37,1
Doğrusal SVM	76,1

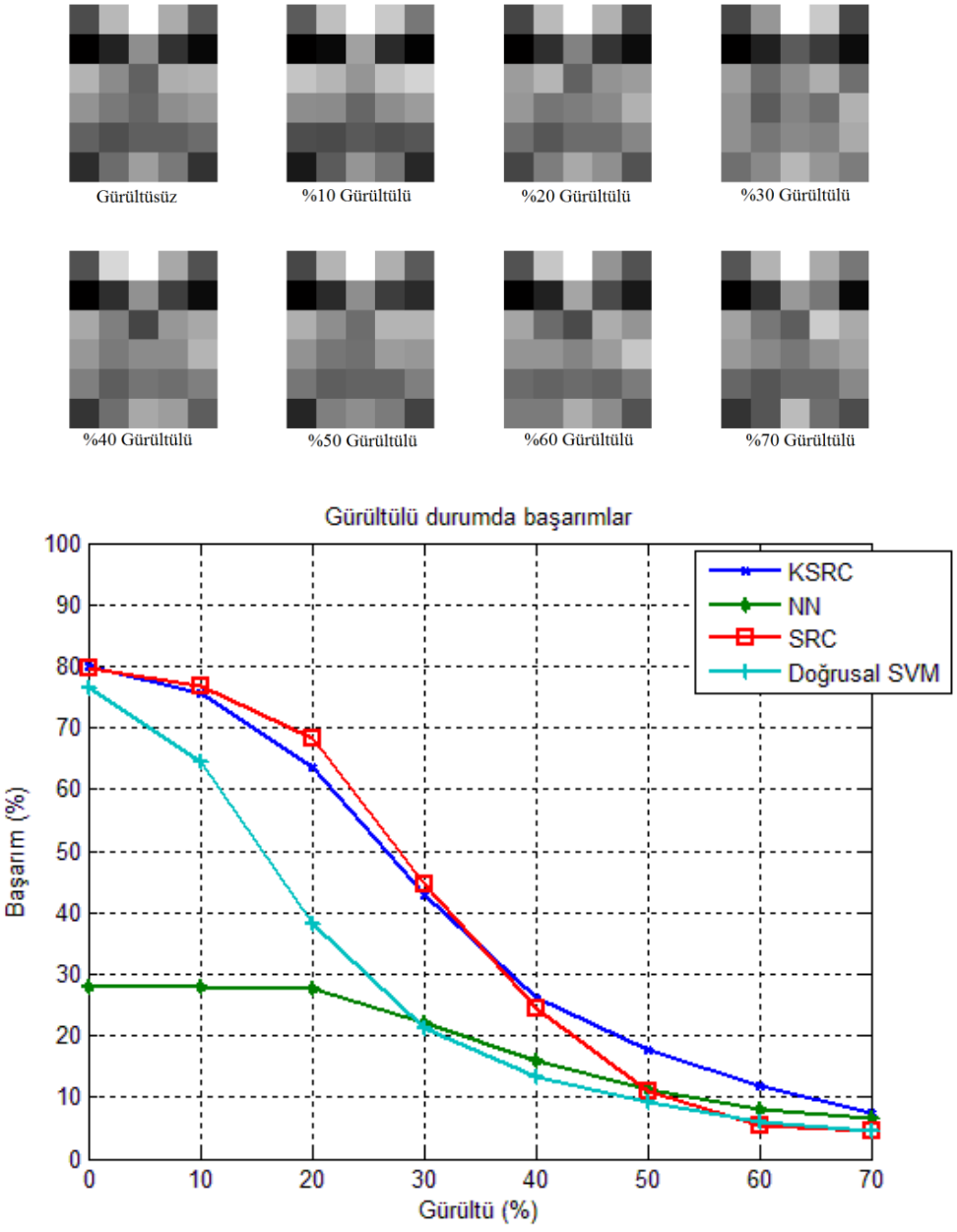
5.1.3.4 Gürültülü durum

Bu benzetimde gürültülü test resimleri için sınıflandırıcıların performansı incelenmiştir. Bu deney için tüm test resimlerinin her biri rastgele olarak seçilmiş ve toplam piksel sayısının %0'ı ile %70'i arasında %10'luk adımlarla bağımsız özdeşçe dağılmış (Independent and Identically Distributed-IID) gürültü ile bozulmuştur. Tüm resimler gri seviyeli olduğu için eklenen gürültü değeri [0, 255] arasında seçilmiştir. Sözlük ise gürültü içermeyen tamamen orjinal yüz resimlerinden oluşturulmuştur. Deneysel sonuçlar öznitelik boyutları 224 ve 30 için elde edilmiştir (Şekil 5.12 ve Şekil 5.13).



Şekil 5.12 : Gürültülü test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 224).

Şekil 5.12'den görülebileceği gibi öznitelik boyutu 224 olarak seçildiğinde, tüm seviyelerdeki gürültülü test resimleri için K-SRC ve SRC diğer sınıflandırıcılardan daha yüksek başarım elde etmiştir. K-SRC, gürültüsüz, %10 gürültülü ve gürültünün %65'ten fazla olduğu durumlarda SRC algoritmasından daha yüksek başarım gösterirken, %10-65 aralığındaki gürültü seviyelerinde SRC, K-SRC'den daha yüksek başarım göstermiştir. Sınıflandırıcılar içindeki en düşük performansı NN algoritması gösterirken Doğrusal SVM gürültüsüz durumdaki başarısını gürültülü durumlarda gösterememiştir.

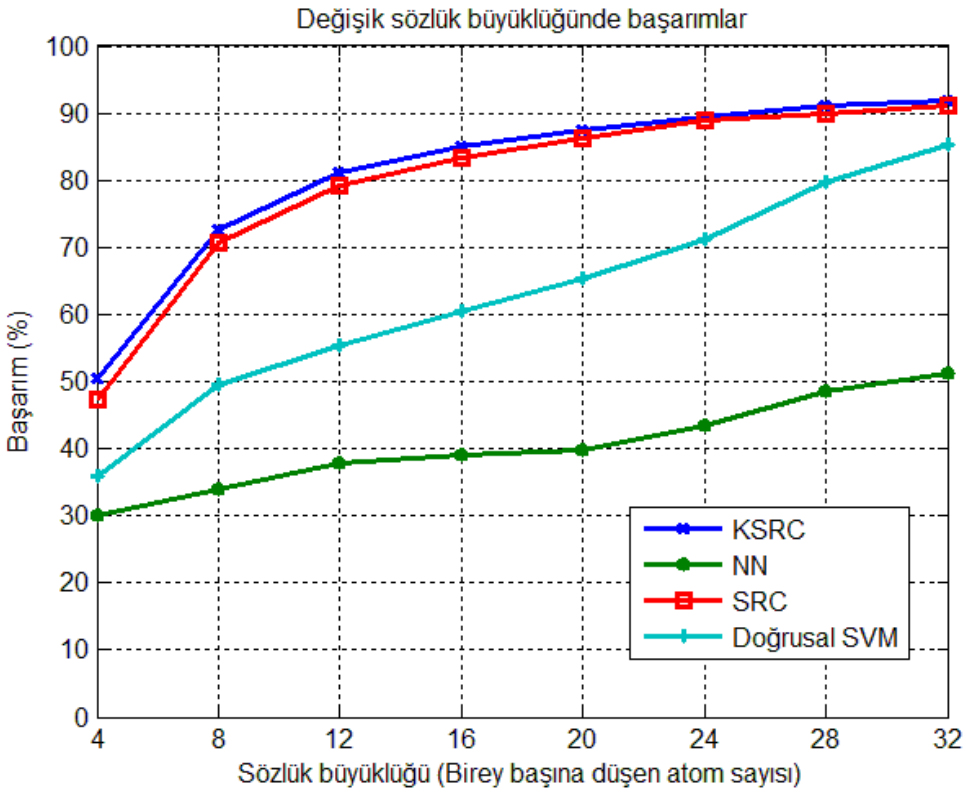


Şekil 5.13 : Gürültülü test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 30).

Şekil 5.13'ten görülebileceği gibi öznitelik boyutu 30 gibi oldukça düşük bir değer seçildiğinde, gürültülü test resimleri için K-SRC ve SRC diğer sınıflandırıcılardan daha yüksek başarım elde etmiştir. K-SRC, gürültüsüz ve gürültünün %35'ten fazla olduğu durumlarda SRC algoritmasından daha yüksek başarım gösterirken, %10-35 aralığındaki gürültü seviyelerinde SRC, K-SRC'den daha yüksek başarım göstermiştir. %20 gürültü seviyesini dikkate aldığımızda SVM ve NN %40'ın altında başarım gösterirken SRC ve K-SRC %65-70 aralığında bir başarım göstermiştir.

5.1.3.5 Değişik sözlük büyüklüğünde başarımlar

Bu benzetimde değişik sözlük büyüklüklerinde sınıflandırıcıların performansı incelenmiştir. Bu amaçla tüm yüz veri seti gürültüsüz olarak sözlük ve test olarak kullanılmıştır. Deneysel sonuçlar birey başına düşen sözlükteki atom sayısı 4, 8, 12, 16, 20, 24, 28 ve 32 için ayrı ayrı elde edilmiştir. Benzetimde öznitelik boyutu 224 alınmıştır (Şekil 5.14).



Şekil 5.14 : Değişik sözlük büyüklükleri için sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 224).

Şekil 5.14'ten görülebileceği gibi birey başına düşen sözlükteki atom sayısının değiştirilmesi durumunda K-SRC diğer sınıflandırıcılardan daha yüksek başarım

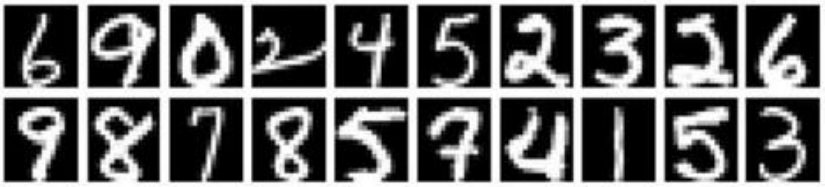
göstermiştir. SRC, diğer sınıflandırıcılardan daha iyi performans göstermiş olsa da K-SRC sınıflandırıcısından yaklaşık %1-%3 daha düşük başarım elde etmiştir. Sınıflandırıcılar içindeki en düşük performansı NN algoritması gösterirken Doğrusal SVM sözlüğün küçültülmesi ile büyük başarım kaybı yaşamıştır.

5.2 Rakam Tanıma

Bu bölümde çevrimdışı ve elle yazılmış rakam tanıma sistemine ait gerçekleştirilen deneysel benzetim sonuçları ele alınmıştır. Veri seti olarak USPS rakam veri seti kullanılmıştır. Bu veri seti her bir resimde tek bir karakter içeren, gerekli ön işlemleri yapılmış ve sadece rakam bölgesini içeren bir veri setidir [76].

5.2.1 USPS rakam veri seti

Bu veri seti Amerikan Posta Hizmetlerine ait zarflar üzerindeki el yazması rakamları içeren bir veri setidir [76]. Orjinalinde değişik boyutlarda ve açılarda olan bu el yazması rakamlar taranarak resim haline dönüştürülmüştür. Veri setindeki resimler, rakamlara odaklı olarak boyutları eşitlenmiş 16×16 gri seviyeli resimlerdir (Şekil 5.15).



Şekil 5.15 : USPS veri setine ait örnek rakamlar.

Bu veri setinde 7291 sözlük resmi ve 2007 test resmi bulunmaktadır. Tüm veri setinin rakamlar üzerindeki dağılımı Çizelge 5.2'deki gibidir.

Çizelge 5.2 : USPS veri setine ait sözlük ve test örnek sayıları.

Rakam	0	1	2	3	4	5	6	7	8	9	Toplam
Sözlük	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

5.2.2 SRC ve K- SRC'nin rakam tanımaya uygulanması

USPS veri seti üzerinde gerçekleştireceğimiz deneysel sonuçlar için veri seti ikiye ayrılmıştır. Bunun için veri seti içerisindeki her bir rakama ait rastgele 350 resim test

ve rastgele 350 resim sözlük için ayrılmıştır. Nihai olarak 10 farklı rakam için toplam test resmi sayısı $N_{test} = 3500$ ve sözlük resmi sayısı $N_{sözlük} = 3500$ olmuştur.

Bu durumda SRC ve K-SRC bölümünde ifade ettiğimiz gibi öznitelik boyutu s olmak üzere test resmi vektör haline getirilmiş $\boldsymbol{x} \in \mathbb{R}^s$ olarak tanımlanır. i. rakama ait sözlük $n_i = 350$ olmak üzere $\boldsymbol{D}_i = [\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, .., \boldsymbol{d}_{i,350}] \in \mathbb{R}^{s \times 350}$ şeklinde tanımlanır. Sisteme ait tüm sözlük ise 10 farklı rakam için $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \dots, \boldsymbol{D}_{10}]$ şeklinde ifade edilir.

5.2.3 Benzetim sonuçları

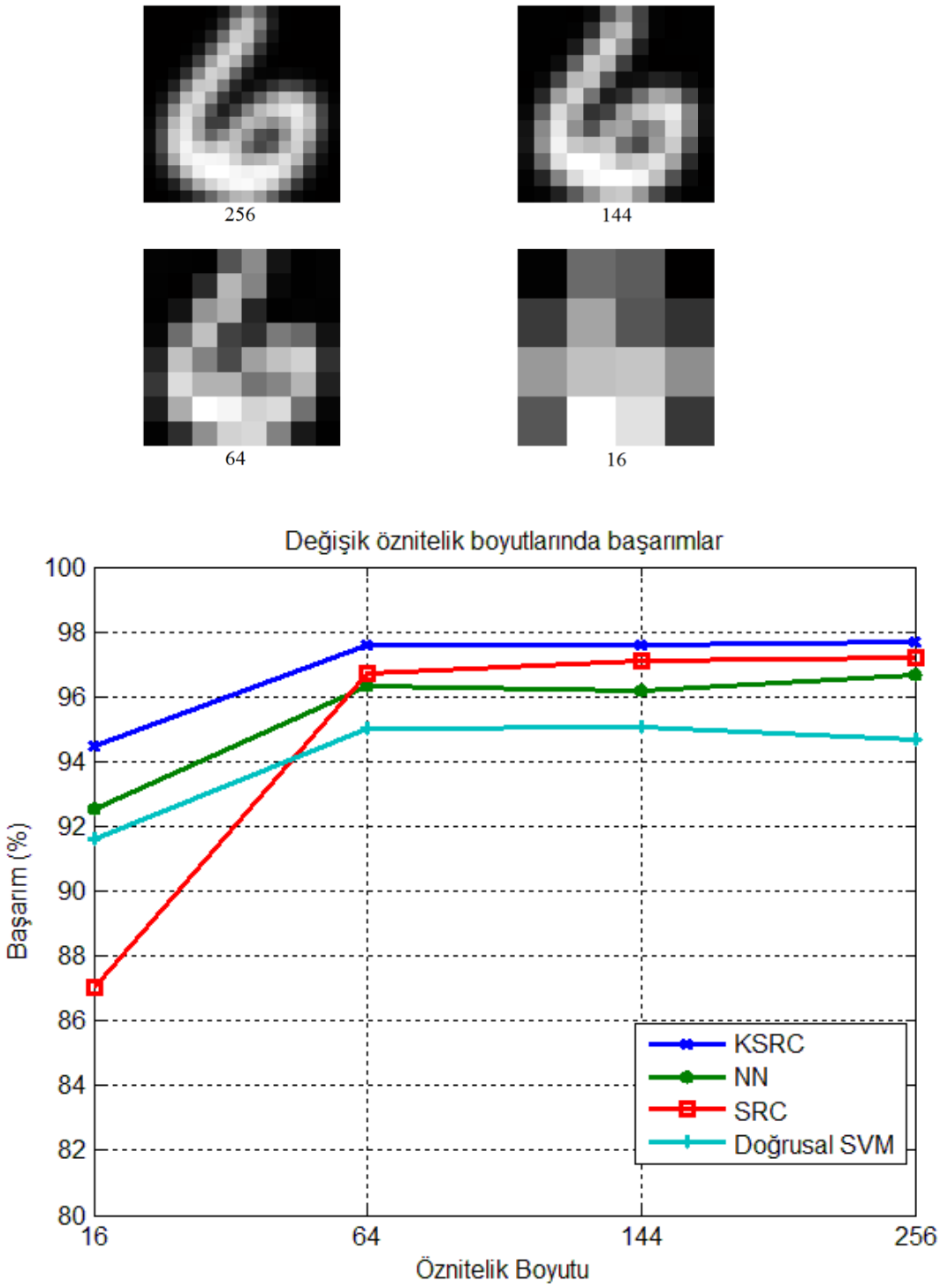
Deneysel benzetim sonuçları MATLAB ortamında geliştirilmiş NN ve Doğrusal SVM algoritmaları ile karşılaştırılmıştır. Tüm algoritmalar tamamen aynı test ve sözlük resimleri üzerinde koşturulmuştur. Van Nguyen ve diğerlerinin çalışmasında elde edilen sonuçlar dikkate alınarak SRC ve K-SRC algoritması için $T_0 = 10$ alınmış ve sınıflandırmalar dağıtık yaklaşımla yapılmıştır [66]. Yine bu çalışma dikkate alınarak USPS veri seti için K-SRC algoritmasında en yüksek başarımı veren çekirdek fonksiyonu, 4. derece polinom olarak seçilmiştir. NN algoritması için k değeri 1 olarak alınmıştır [77].

Ayrıca tüm algoritmalar 10 farklı sözlük ve test seti üzerinde 10 sefer koşturulmuş ve başarım ortalamaları dikkate alınmıştır. Bu sayede seçilen sözlük ve test resimlerinin başarım üzerinde yaratabileceği etki minimize edilmiştir. Sınıflandırma başarımları ise yüz tanımadaki ile benzer olarak denklem 5.3 ile hesap edilmiştir.

5.2.3.1 Gürültüsüz durum

Bu benzetimde tüm rakam veri seti gürültüsüz haliyle sözlük ve test olarak kullanılmıştır. Deneysel sonuçlar 16, 64, 144 ve 256 öznitelik boyutları için ayrı ayrı elde edilmiştir. Algoritmaların çalışma süreleri Çizelge 5.3' te verilmiştir.

Şekil 5.16'dan görülebileceği gibi gürültüsüz test resimleri için tüm öznitelik boyutlarında K-SRC sınıflandırıcısı diğerlerinden daha yüksek başarım elde etmiştir. SRC, düşük öznitelik boyutunda SVM ve NN'den daha düşük performans gösterirken diğer tüm öznitelik boyutları için K-SRC'den sonra en iyi başarım gösteren sınıflandırıcı olmuştur. NN ise tüm öznitelik boyutlarında Doğrusal SVM'den daha yüksek performans göstermiştir.



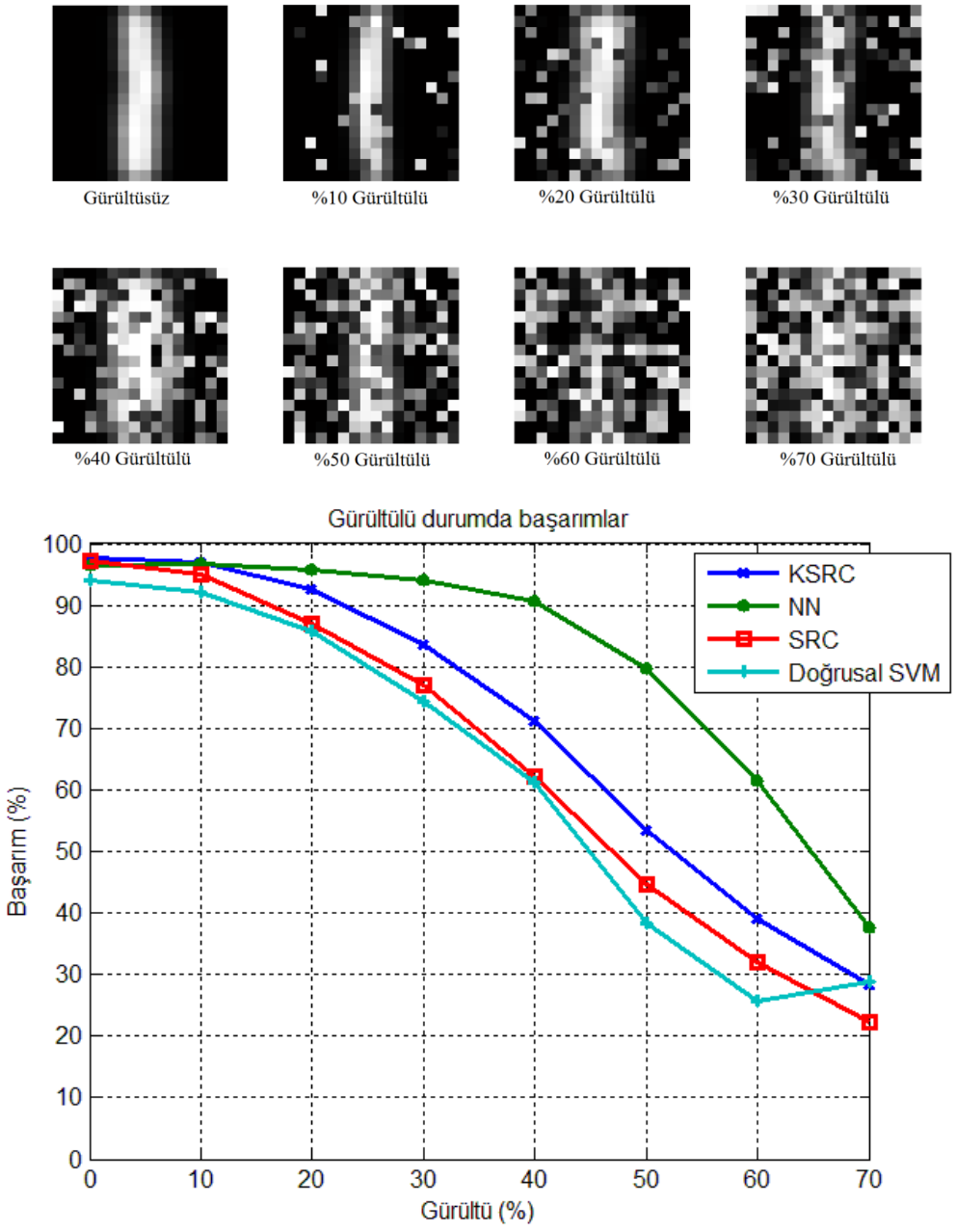
Şekil 5.16 : USPS gürültüsüz test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri.

Çizelge 5.3 : Gürültüsüz durumda algoritmaların çalışma süreleri (USPS).

Algoritma	Süre (Sn)
KSRC	494,7
SRC	44,4
NN	122,1
Doğrusal SVM	31,1

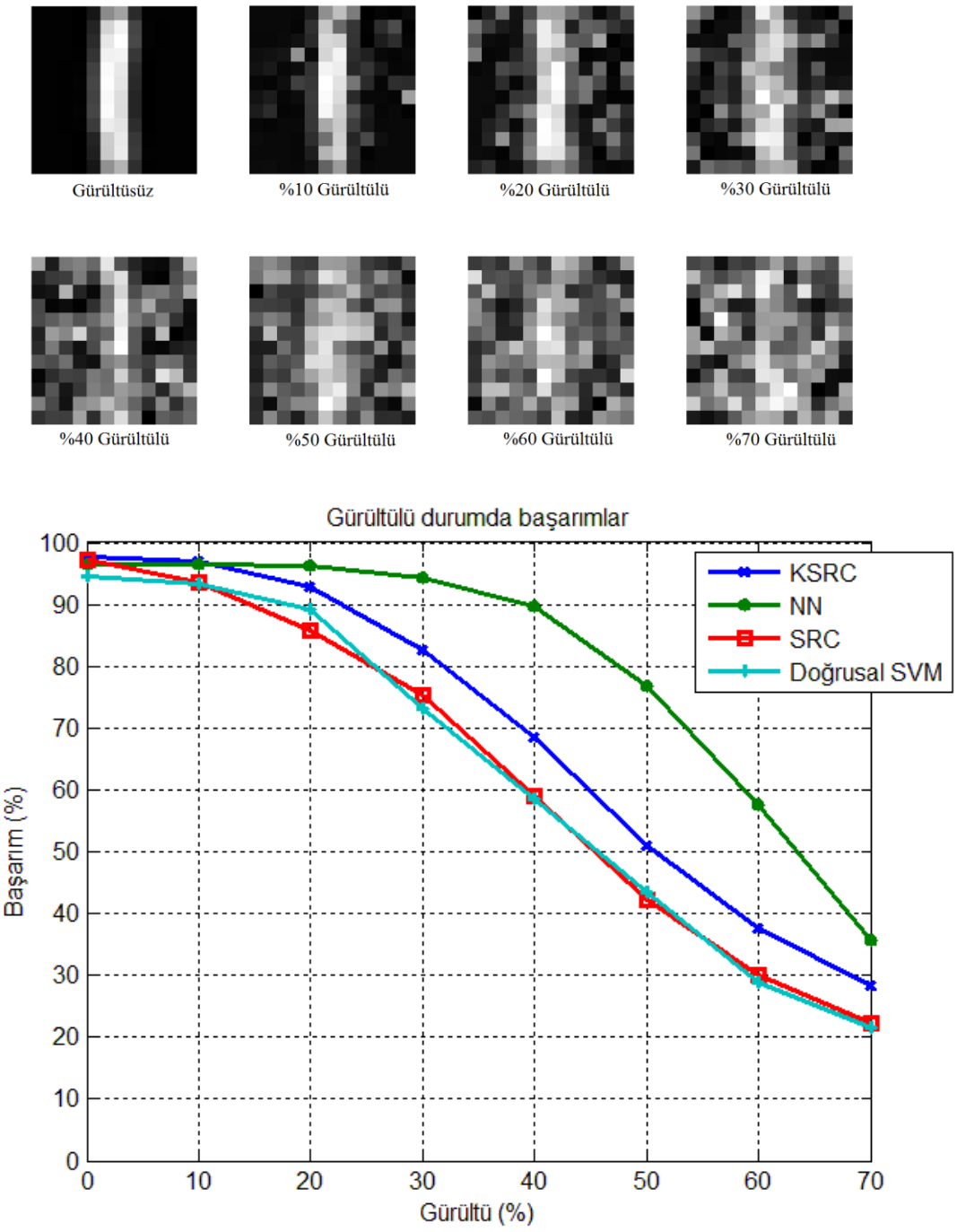
5.2.3.2 Gürültülü durum

Bu benzetimde gürültülü test resimleri için sınıflandırıcıların performansı incelenmiştir. Bu deney için tüm test resimlerinin her biri rastgele olarak seçilmiş ve toplam piksel sayısının %0'ı ile %70'i arasında %10'luk adımlarla IID gürültü ile bozulmuştur. Tüm resimler gri seviyeli olduğu için eklenen gürültü değeri [0, 255] arasında seçilmiştir. Sözlük ise gürültü içermeyen tamamen orjinal rakam resimlerinden oluşturulmuştur. Deneysel sonuçlar 256 ve 144 öznitelik boyutları için elde edilmiştir (Şekil 5.17 ve Şekil 5.18)



Şekil 5.17 : USPS gürültülü test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 256).

Şekil 5.17'de görülebileceği gibi öznitelik boyutu 256 seçildiğinde, gürültüsüz ve %10 gürültülü test resimleri için K-SRC diğer sınıflandırıcılardan daha yüksek başarım göstermiştir. Fakat %10-%70 gürültülü test resimleri için NN en yüksek başarımı göstermiştir. SRC, %60 gürültü seviyesine kadar SVM'den daha yüksek başarım gösterirken %70 gürültü seviyesinde SVM, SRC'den daha yüksek başarım göstermiştir.

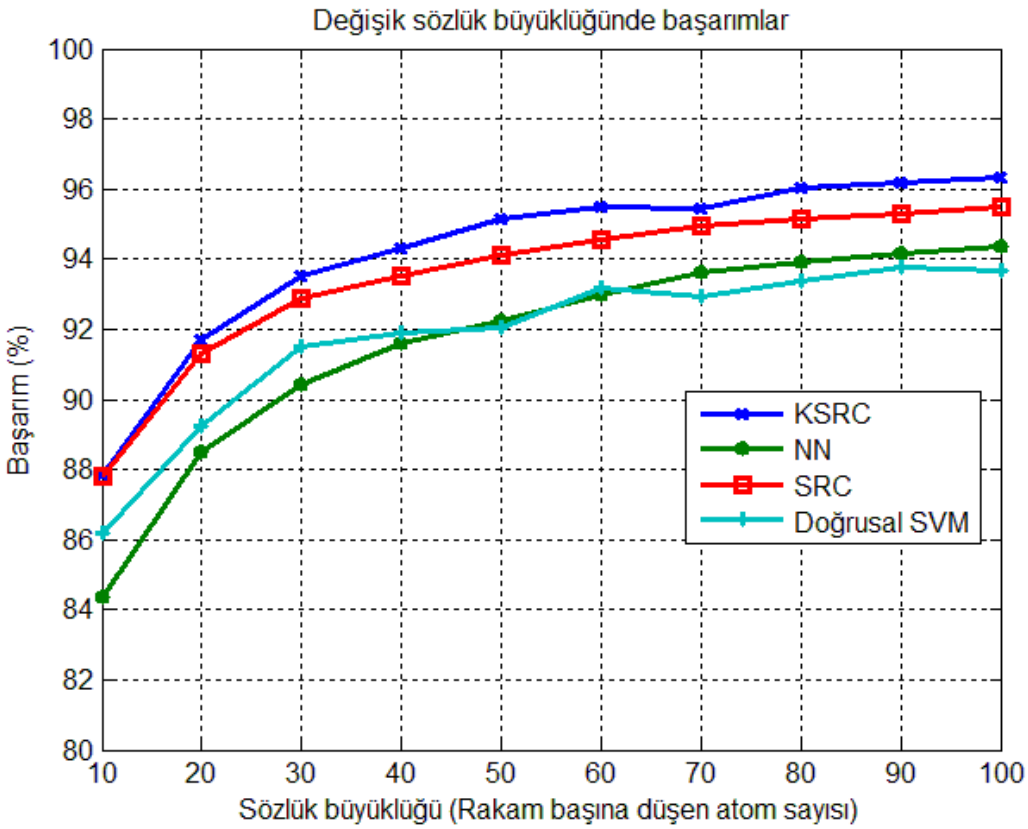


Şekil 5.18 : USPS gürültülü test resimleri için değişik sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 144).

Şekil 5.18'de görülebileceği gibi öznitelik boyutu düşürülerek 144 seçildiğinde, gürültüsüz ve %10 gürültülü durumlarda K-SRC en yüksek performansı göstermiştir. Tüm sınıflandırıcıların başarımı bir miktar düşerken genel itibariyle tüm sınıflandırıcılar 144 ve 256 öznitelik boyutları için oldukça benzer başarımlar göstermiştir.

5.2.3.3 Değişik sözlük büyüklüğünde başarımlar

Bu benzetimde değişik sözlük büyüklüklerinde sınıflandırıcıların performansı incelenmiştir. Bu amaçla tüm rakam veri seti gürültüsüz olarak sözlük ve test olarak kullanılmıştır. Deneysel sonuçlar rakam başına düşen sözlükteki atom sayısı 10, 20, 30, 40, 50, 60, 70, 80, 90 ve 100 için ayrı ayrı elde edilmiştir. Benzetimde öznitelik boyutu 144 alınmıştır (Şekil 5.19).



Şekil 5.19 : Değişik sözlük büyüklükleri için sınıflandırıcılara ait başarım yüzdeleri (Öznitelik boyutu: 144).

Şekil 5.19'dan görülebileceği gibi rakam başına düşen sözlükteki atom sayısının değiştirilmesi durumunda K-SRC diğer sınıflandırıcılardan daha yüksek başarım göstermiştir. SRC, diğer sınıflandırıcılardan daha iyi performans göstermiş olsa da

K-SRC sınıflandırıcısından yaklaşık %1 daha düşük başarım elde etmiştir. Düşük sözlük boyutlarında Doğrusal SVM, NN'den daha yüksek başarım gösterirken sözlük boyutunun 60 ve üzerine çıkması durumunda NN algoritması Doğrusal SVM'den daha yüksek başarım elde etmiştir.

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında seyreklik tabanlı sınıflandırma ve çekirdek yaklaşımı anlatılmıştır. Bu kapsamda SRC ve K-SRC algoritmaları ile birlikte karşılaştırma için kullanılan Doğrusal SVM ve NN algoritmaları MATLAB ortamında gerçekleştirilmiştir. Genel olarak seyreklik tabanlı sınıflandırıcıların Doğrusal SVM ve NN gibi bilinen sınıflandırıcılar ile karşılaştırıldığında oldukça rekabetçi performans gösterdiği gözlenmiştir.

Yüz tanıma uygulaması için gerçekleştirilen değişik öznitelik boyutundaki benzetim sonuçlarına baktığımızda K-SRC algoritması diğer tüm sınıflandırıcılardan daha yüksek başarım göstermiştir. Bununla birlikte SRC algoritması K-SRC algoritmasına oldukça yakın performans göstermiştir. Bunun nedeni olarak Yale B veri setindeki yüz resimlerinin doğrusal olarak oldukça yüksek bir başarımda ayrılabiliyor olmasını gösterebiliriz. Bu nedenle yüz veri seti için doğrusal olmayan K-SRC sınıflandırıcısı SRC sınıflandırıcısına göre büyük bir başarım farkı yaratmamıştır. Benzer şekilde gürültülü durumda da SRC ve K-SRC algoritmaları oldukça yakın performans göstermiştir. Öznitelik boyutu 30 için yapılan gürültülü benzetimde, %10-%35 arası gürültü durumları dışında K-SRC, SRC'den daha yüksek başarım göstermiştir. Ayrıca değişik sözlük büyüklüklerinde yapılan benzetim sonuçlarında K-SRC ve SRC diğer sınıflandırıcılardan daha yüksek başarım göstermiştir.

Rakam tanıma uygulaması için benzetim sonuçlarına baktığımızda değişik öznitelik boyutları için K-SRC algoritması, SRC ve diğer klasik yöntemlerden daha yüksek başarım göstermiştir. Bununla birlikte gürültülü benzetim sonuçlarına baktığımızda %10 gürültü durumunda K-SRC oldukça iyi bir başarım göstermiş ve doğrusal sınıflandırıcıların tamamından daha iyi bir sınıflandırma yapmıştır. Fakat %10 ile %70 arası gürültülü test resimlerinde 1-NN algoritması Doğrusal SVM ve seyrek sınıflandırıcılardan daha yüksek başarım göstermiştir. Bunun nedeni hem doğrusal hem de doğrusal olmayan seyrek sınıflandırıcıların rakam tanıma uygulamaları için gürültüye karşı oldukça hassas olmasıdır.

Yüz tanıma ve rakam tanıma uygulamalarında elde edilen benzetim sonuçlarını karşılaştırdığımızda, seyreklik tabanlı sınıflandırıcılar yüz tanıma için değişik sözlük, öznitelik ve gürültü seviyelerinde oldukça yüksek ve tutarlı başarımlar göstermiştir. Fakat rakam tanıma için gürültüsüz durumda seyreklik tabanlı sınıflandırıcılar oldukça tutarlı ve iyi bir başarım gösterirken, gürültülü durumlarda beklenen performansı verememiştir. Özetle SRC ve K-SRC algoritmaları yüz tanıma uygulamaları ile gürültüsüz durumda rakam tanıma uygulamalarında tercih edilebilir iyi bir yöntem olarak karşımıza çıkmıştır.

Bu sonuçlarla birlikte günümüzde seyreklik tabanlı sınıflandırıcılar üzerine yapılan çalışmalar sınıflandırmada l_2 normunun ağırlık olarak kullanılabileceğini göstermiştir [78]. Bu ağırlık mekanizması, sözlük atomları ile test resmi arasındaki Öklid uzaklığının sınıflandırmada kullanılması ile yapılmaktadır. Bu bağlamda tezde ele alınan SRC ve K-SRC algoritmalarına bu ağırlık mekanizmasının ilave edilmesi benzetimlerde elde edilen başarımları bir miktar artırabileceği düşünülmektedir.

KAYNAKLAR

[1] Selvarani, S., Jebapriya, S. ve Mary, R. S., (2014). Automatic Identification and Detection of Altered Fingerprints, *in Intelligent Computing Applications (ICICA) 2014 International Conference*, 239-243

[2] Breiman, L., Freidman, J., Olshen, R. A. ve Stone, J. (1984). *Classification and Regression Trees,* Chapman and Hall.

[3] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition,* Oxford U. Press, New York.

[4] Stephen, D. B. (1998). Combining nearest neighbor classifiers through multiple feature subsets, *In Proceedings of the 17th International Conference on Machine Learning*, 37-45.

[5] Vapnik, V. (1995). The nature of statistical learning theory. *Springer-Verlag,* New York.

[6] Donoho, D. L. (2006). Compressed sensing, *IEEE Trans. Inf. Theory, 52(4), 1289-1306.*

[7] Wright, J., Yang A. Y., Ganesh A., Sastry S. S. ve Ma, Y. (2009). Robust face recognition via sparse representation, *IEEE Transactions on Pattern analysis and Machine Intelligence,* 31(2), 210-227.

[8] Boser, B., E., Guyon, I. M. and Vapnik, V. (1992). A training algorithm for optimum margin classifiers, *In Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, ACM.

[9] Zhang, L. ve Zhou, W. (2009). On the sparseness of 1-norm support vector machines, *Neural Networks*, 23(3), 373-385.

[10] Gao, S., Tsang, I.W.-H. ve Chia, L.-T (2010). Kernel sparse representation for image classification and face recognition, *11th Eur. Conf. Comput. Vis.*, 6314, 1-14.

[11] Alpaydın, E. (2009). *Introduction to Machine Learning.* London: The MIT Press.

[12] Url-1 < *http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551/slides/cs551_intro.pdf* > , erişim tarihi 29.10.2015.

[13] Url-2 < *http://oftankonyv.reak.bme.hu/tiki-download_file.php?fileId=2015&display* > , erişim tarihi 29.10.2015.

[14] Url-3 < *http://yuenshome-wordpress.stor.sinaapp.com/uploads/2014/10/QQ%E6%88% AA%E5%9B%BE20141027160518.jpg*>, erişim tarihi 29.10.2015.

[15] Japan Association on Remote Sensing (1996). *Remote Sensing Note***.** Tokyo, Japan: JARS.

[16] Guo G., Li S.Z. ve Chan, L.C. (2000). Face recognition by support vector machines, *Automatic Face and Gesture Recognition, Proceedings Fourth IEEE International Conference,* 196-201.

[17] Osuna, E., Freund R., ve Gırosı, F. (1997). Training support vector machines: an application to face detection, *Proceedings of the IEEE CVPR,* 130-136.

[18] Pontil, M. ve Verri, A. (1998). Support vector machines for 3D object recognition. *Pattern Analysis and Machine Intelligence,* 20(6), 637-646.

[19] Chapelle, O., Haffner, P. ve Vapnik, V.N (1999). Support vector machines for histogram-based image classification, *Neural Networks* 10(5), 1048–1054.

[20] Drucker, H., Wu, D. H., ve Vapnik, V. N (1999). Support Vector Machines for Spam Categorization, *Neural Networks,* 10(5), 1055–1064.

[21] Zhan, Y. Q. ve Shen, D. G., (2005). Design Efficient Support Vector Machine for Fast Classification, *Pattern Recognition,* 38(1), 157–161.

[22] Url-4 < *http://fedc.wiwi.hu-berlin.de/xplore/ebooks/html/csa/node221.html*>, erişim tarihi 29.10.2015.

[23] Özkan, C. (2008). Veri madenciliği yöntemleri. *Papatya yayıncılık eğitim.* İstanbul.

[24] Mathur, A. ve Foody, G.M. (2008). Multiclass and Binary SVM Classification: Implications for Training and Classification Users, *Geoscience and Remote Sensing Letter IEEE,* 5(2), 241-245.

[25] Cortes, C. ve Vapnik V. (1995). Support-Vector Networks. *Machine Learning IEEE,* 20 (3), 273-297.

[26] Courant, R. ve Hilbert, D. (1953). Methods of Mathematical Physics. *Interscience.* New York.

[27] Url-5 < *http://jblomo.github.io/datamining290/slides/2013-03-01-SVM.html*>, erişim tarihi 29.10.2015.

[28] Cover, T. M. ve Hart, P. E. (1967). Nearest Neighbor Pattern Classification, *IEEE Trans. Inform. Theory,* 13(1), 21-27.

[29] Vaidehi, V. ve Vasuhi, S. (2008). Person Authentication using Face Recognition, *Proceedings of the world congress on Eng. and computer science.* San Francisco.

[30] Shizen, Y. Wu, (2008). An Algorithm for Remote Sensing Image Classification based on Artificial Immune b-cell Network, *International Archives of the Photogrammetry, Remote Sensing and Spatial Inf. Scien.,* 40(PartB6b), 107-111.

[31] Bajramovic, F., Mattern, F., Butko, N. ve Denzler, J. (2006). A Comparison of Nearest Neighbor Search Algorithms for Generic Object Recognition, *ACIVS 2006,* 1186-1197.

[32] Chidananda, K. ve Krishna, G. (1979). The condensed nearest neighbor rule using the concept of mutual nearest neighbor, *IEEE Trans. Information Theory*, 25(4), 488-490.

[33] Gates, G.W. (1972). Reduced Nearest Neighbor Rule, *Information Theory, IEEE Transactions on*, 18(3), 431-433.

[34] Guo, G., Wang, H. ve Bell, D. (2003). KNN Model-Based Approach in Classification, *Springer Berlin Heidelberg, 2888.*

[35] Url-6 < *http://pages.videotron.com/djihess/knn/ProjectionMethod.html*>, erişim tarihi 29.10.2015.

[36] Darwin, C. (1872). *The Expression of the Emotions in Man and Animals*, London: John Murray.

[37] Galton, F. (1888). *Personal identification and description, Nature, 173-188.*

[38] Goldstein, J. A. ve Harmon, L. D (1971). Identification of human faces, *Proceedings of the IEEE*, 34(9), 748-760.

[39] Fischler, M. A. ve Elschlager, R. A. (1973). The representation and matching of pictorial structures, *IEEE Transactions on Computers*, 22(1), 67-92.

[40] Yuille, A., Hallinan, P. ve Cohen, D. (1992). Feature extraction from faces using deformable templates, *International Journal of computer Vision*

[41] Viola, P. ve Jones, M. (2001). Robust real-time face detection, *ICCV 2001. Proceedings. Eighth IEEE International Conference*, 57(2), 137-154.

[42] Url-7 < *http://stackoverflow.com/questions/5808434/how-does-the-viola-jones-face-detection-method-work* >, erişim tarihi 29.10.2015.

[43] Lam, K. M. ve Yan, H. (1994). . Fast algorithm for locating head boundaries, *Journal of Electronic Imaging*, 3(04), 351-359.

[44] Yang, M. H., Kriegman, D. ve Ahuja, N. (2002). Detecting faces in images:A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1), 34-58.

[45] Zhao, W., Chellappa, A., Rosenfeld, A. ve Phillips, P. (2003). Face recognition: A literature survey, *ACM Computing Surveys*, 399-458.

[46] Turk, M. ve Pentland, A. (1991). Eigenfaces for Recognition, *IEEE Int'l Conf. Computer Vision and Pattern Recognition.*

[47] Kumar, G. ve Bhatia, P.K. (2014). A Detailed Review of Feature Extraction in Image Processing Systems, *in Advanced Computing & Communication Technologies (ACCT)*, 5-12.

[48] He, X., Yan, S., Hu, Y., Niyogi, P., ve Zhang, H. (2005). Face Recognition Using Laplacianfaces, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(3), 328-340.

[49] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press,Oxford-UK.

[50] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*, 2nd Edition, Prentice Hall.

[51] Haykin, S. (1998). *Statistical Learning Theory*, John Wiley and Sons, New York, USA.

[52] Neves, R. F. P., Filho, A. N. G. L., Mello, C. A. B., ve Zanchettin, C. (2011). A SVM based off-line handwritten digit recognizer, *IEEE Systems, Man, and Cybernetics (SMC)*, 3(04), 351-359.

[53] Lu, X. ve Li, X. (2014). Group sparse reconstruction for image segmentation, *Neurocomputing*, 136, 41-48.

[54] Elad, M., Figueiredo, M. A. T. ve Ma, Y. (2010). On the role of sparse and redundant representations in image processing, *Proc. IEEE*, 98(6), 972-982.

[55] Candès, E. J., Romberg, J. ve Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. Inf. Theory*, 52(2), 489-509.

[56] Elad, M., Figueiredo, M. A. T. ve Ma Y. (2010). On the role of sparse and redundant representations in image processing, *Proc. IEEE*, 98(6), 972-982.

[57] Mallat, S. (2008). A Wavelet Tour of Signal Processing: The Sparse Way, New York, NY, USA: Academic.

[58] Starck, J. L., Murtagh, F. ve Fadili, J. M. (2010). *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity*, Cambridge, U.K. Cambridge Univ. Press.

[59] Url-8 < *https://tianyizhou.files.wordpress.com/2010/08/cs11.jpg*>, erişim tarihi 29.10.2015.

[60] Url-9 < *http://www.amepc.org/qims/article/viewFile/2233/3083/8679*>, erişim tarihi 29.10.2015.

[61] Url-10 < *https://www.ceremade.dauphine.fr/~peyre/matlab/sparsity/content .html*>, erişim tarihi 22.11.2015.

[62] Elad, M. (2009). *Sparse and Redundant Representations.* London: Springer.

[63] Tropp, J. A, Gilbert, A. C., ve Strauss, **M. J.** (2006). Algorithms for simultaneous sparse approximation: part I: Greedy pursuit, *Signal Process*. 86(3), 572-588.

[64] Mallat, S. ve Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries, *IEEE Transactions on Signal Processing*, 41(12), 3397-3415.

[65] Pati, Y. C., Rezaiifar, R. ve Krishnaprasad, P. S., (1993). "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in Signals, Systems and Computers, *Conference Record of The Twenty-Seventh Asilomar Conference*, 1, 40-44.

[66] Tropp, J. A. ve Gilbert, A. C. (2007). Signal recovery from random measurements via orthogonal matching pursuit, *IEEE Trans. Inf. Theory*, 53(12), 4655-4666.

[67] Van Nguyen, H., Patel, V.M., Nasrabadi, N.M. ve Chellappa, R. (2013). Design of Non-Linear Kernel Dictionaries for Object Recognition, *IEEE Transactions on Image Processing*, 22(12), 5123-5135.

[68] Zhang, L., Yang, M., ve Feng, X. (2011). Sparse representation or collaborative representation: which helps face recognition, *IEEE Conference on Computer Vision*, 471-478

[69] Xu, Y., Zhang, D., Yang, J. ve Yang, J. Y. (2011). *A two-phase test sample sparse representation method for use with face recognition, IEEE Trans.Circuits Syst. Video Techno*, 21(9), 1255-1262.

[70] Deng, W., Hu, J., ve Guo, J. (2012). Extended SRC: Undersampled face recognition via intraclass variant dictionary, *IEEE Trans. Pattern Anal. Mach. Intell*, 34(9), 1864-1870.

[71] Li, H., Gao Y. ve Sun, J. (2011). Fast Kernel Sparse Representation, *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference*, 72-77.

[72] Chang, C. C. ve Lin, C. J. (2001). *LIBSVM: A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/Libsvm.*

[73] Elad, M., Rubinstein, R. ve Zibulevsky, M. (2008). Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit, Technical Report - CS, Technion.

[74] Georghiades, A. S., Belhumeur, P. N. ve Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose, *IEEE Trans. Pattern Anal. Mach. Intell*, 23(6) 643-660.

[75] Ebrahimpour H. ve Abbas K. (2007). Face Recognition Using Bagging KNN, *International Conference on Signal Processing and Communication Systems*, 17-19.

[76] Hull, J.J. (1994). A database for handwritten text recognition research, *IEEE Trans. Pattern Anal. Mach. Intell*, 16(5), 550-554.

[77] Babu, U. R., Venkateswarlu, Y. ve Chintha, A. K. (2014). Handwritten Digit Recognition Using K-Nearest Neighbour Classifier, *in Computing and Communication Technologies (WCCCT)*, 60-65.

[78] Fan, Z., Ni M., Zhu, Q. ve Liu, E. (2015). Weighted sparse representation for face recognition, *Neurocomputing*, 304-309.

ÖZGEÇMİŞ

Ad-Soyad **: Abdurrahman Yeşiloğlu**

Doğum Tarihi ve Yeri **: 25.11.1989 - Kadıköy**

E-posta **:** abdurrahmanyesiloglu@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:**2010, Sakarya Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü (Çift Anadal)
- **Lisans** **:**2010, Sakarya Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2010 yılında Elektrik-Elektronik Mühendisliği bölümünü derece ile tamamladı.
- 2007-2010 yılları arasında çift anadal programı ile Bilgisayar Mühendisliği bölümünü tamamladı.