T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Suna YILDIRIM

ARIZA TEŞHİSİNDE DESTEK VEKTÖR MAKİNELERİNİN KULLANIMI

Tez Yöneticisi: Doç.Dr. Erhan AKIN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2006

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARIZA TEŞHİSİNDE DESTEK VEKTÖR MAKİNELERİNİN KULLANIMI

Suna YILDIRIM

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez, ... tarihinde aşağıda belirtilen jüri tarafından oybirliği /oyçokluğu ile başarılı / başarısız olarak değerlendirilmiştir.

Danışman : Doç.Dr. Erhan AKIN

Üye : Prof.Dr. Yakup DEMİR

Üye : Yrd. Doç.Dr. Burhan ERGEN

Üye:

Üye:

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

Bu tezde, bana yardımcı olan ve sürekli desteğini esirgemeden, sabırla beni yönlendiren danışman hocam Sayın Doç. Dr. Erhan AKIN'a ve Sayın Yrd. Doç. Dr. Mehmet KARAKÖSE'ye teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

SİMGELER

x_i:	Giriş Verileri
y_i:	Giriş Verilerine Uyan Çıkış Etiketleri
f(x):	Hiperdüzlem Denklemi
w:	Ağırlık Terimi
b:	Bias terimi
i_F:	Gösterge Fonksiyonu
L(w,b):	Langrangian Fonksiyonu
α_i:	Langrange Çarpanları
D:	Herhangi Bir Nokta ile Hiperdüzlem Arasındaki Mesafe
ξ:	Serbestlik Değişkenleri
$\phi(x)$:	Kernel Fonksiyonu
k:	Sınıf Sayısı
n:	Eğitim Örnekleri Sayısı

KISALTMALAR

DVM: Destek Vektör Makineleri

DV: Destek Vektörleri

VC: Vapnik Chervonenkis Boyutu

ÖZET

Yüksek Lisans Tezi

ARIZA TEŞHİSİNDE DESTEK VEKTÖR MAKİNELERİNİN KULLANIMI

Suna YILDIRIM

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
2006, Sayfa:77

Asenkron motorlar endüstride çok sık kullanılan motor türleridir. Motor içerisindeki bazı parçaların arızalanması bütün sistemi kötü etkileyebilir. Bu nedenle, herhangi bir parçada meydana gelen arızanın en kısa sürede tespit edilip, bütün sistemin etkilemeden onarılması çok önemlidir.

Destek Vektör Makineleri, metin karakterizasyonu, imaj tanıma, biyoinformatik gibi birçok uygulamada iyi sonuçlar vermesine rağmen arıza teşhisinde yeni kullanılmaya başlanmıştır. Bu tezde ise asenkron motorların arıza teşhisinde Destek Vektör Makineleri yöntemi kullanılmıştır ve çok iyi sonuç verdiği gözlemlenmiştir.

Bu tezde, Destek Vektör Makineleri için asenkron motorlardan alınan veriler giriş olarak kullanılmıştır. Sağlıklı motor verileri ile kırık rotor çubuğu arızası olan motordan alınan veriler sınıflandırılmıştır. İkili sınıflandırma ve çoklu sınıflandırma işlemleri gerçekleştirilmiştir. Çoklu sınıflandırma yöntemlerinden birine-karşı-biri metodu kullanılmıştır.

Anahtar Kelimeler: Destek Vektör Makineleri, Arıza Teşhisi, Sınıflandırma, Kernel Fonksiyonları

ABSTRACT

Master Thesis

THE USING OF SUPPORT VECTOR MACHINES IN FAULT DIAGNOSIS

Suna YILDIRIM

Firat University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering
2006, Page: 77

Induction motors have been commonly used in the industries.The fault diagnosis in some parts affects all of the system adversely. So, it is very important to determine the fault occured in any part in a short time and to repaire the all of the system without affecting the all of the system.

Although the Support Vector Machines give good results from most problems like text categorization, image recognition, bioinformatics, it has not been used in fault diagnosis. In this thesis, Support Vector Machines used in fault diagnosis and it has beeen observed that it gave good results.

In this thesis, the data received from induction motors used for the input of Support Vector Machines. The classification between healthy motor data and broken rotor bar faulty data has been classified. Binary classification and multi-class classification processes are realized. One-against-one method is used for multi-class classification.

Key Words: Support Vector Machines, Fault Diagnosis, Classification, Kernel Functions.

1. GİRİŞ

Asenkron motorlar, sağlam ve güvenilir yapılarından dolayı endüstride en fazla kullanılan motorlardır. Bazı elektromekanik parçalardaki arızaların teşhisi ve bunların onarımının çok zaman alması bütün sistemi kötü yönde etkilemektedir. Bu nedenle, herhangi bir parçada meydana gelen arıza en kısa sürede tespit edilip, bütün sistemi etkilemeden onarılmalıdır.

Asenkron motorlar günlük hayatımızda kullandığımız buzdolabı ve çamaşır makinesi gibi birçok eşyada bulunmaktadır. Bu tür asenkron motorlar düşük güçlü motorlardır. Daha güçlü asenkron motorlar ise fabrikalarda kullanılmaktadır. Asenkron motorların bu kadar çok kullanılmalarının sebeplerini şöyle sıralayabiliriz [1]:

1. Ucuz olmaları,
2. Diğer motorlara göre daha az bakıma ihtiyaç duymaları,
3. Sağlam ve güvenilir bir yapıda olmalarıdır.

Motorların arıza tespiti, teşhisi ve izlenmesi önemli konulardan biridir. Uygun bir izleme tekniğinin kullanılması ve uygun arıza teşhis algoritmalarıyla erken arıza tespiti sistem genelinde büyük arızaların oluşumunu engelleyebilmektedir. Son zamanlardaki çalışmalar motor arızalarının %90'ının motorun dahili bileşenlerde meydana geldiğini göstermektedir[1]. Fabrikaların çoğunda reaktif koruma ve arıza durumunda korumayı kullanmaktadırlar. Reaktif koruma, pahalı olması bakımından çok tercih edilen bir yöntem değildir. Koruyucu bakım yöntemi ise arızaların oluşmadan teşhisi için donanımın sürekli izlenmesini gerektiren bir yöntemdir[2].

Destek Vektör Makineleri (DVM), V.N. Vapnik tarafından geliştirilen istatistiksel öğrenme teorisi üzerine kurulu olan modern hesaba dayalı bir öğrenme metodudur[3]. DVM'lerde giriş uzayı yüksek boyutlu bir özellik uzayına düşürülür ve özellik uzayında sınıflandırıcının genelleme yeteneğini arttırmak için optimal bir hiperdüzlem seçilir. Bu hiperdüzlem aracılığıyla özelliklerin hangi sınıfa ait olduğu tespit edilir [4].

DVM geleneksel sınıflandırıcılarla karşılaştırıldığında, ek özelliklerinin olduğu görülür. En önemli karakteristikleri çok geniş özellik uzaylarını tutabilmeleridir. DVM'lerde kullanılacak örnek sayısı önemli değildir. Genelleme yetenekleri ve hesaba dayalı verimlilikleri, giriş uzayının boyutundan bağımsızdır. Bu özelliğiyle de yüksek boyutlu sınıflandırma problemlerinde sıkça kullanılmaktadırlar. DVM yönteminde arıza özelliklerinin sayısında bir sınırlandırma yapılmamıştır, bu da çok kullanışlı bir özelliktir. DVM'lerin diğer yöntemlere

göre daha iyi sonuçlar veren özelliklerinden bir tanesi de çalışılan probleme global optimal çözümü vermesidir. Son yıllarda yüksek boyutlu problemlerde bile çok iyi sonuçlar verdiği için büyük bir ilgi toplamıştır[14].

DVM'ler özellikle iki-sınıflı sınıflandırma problemlerinde çok fazla kullanılmaya başlanmıştır. Girişte alınan veriler destek vektör sayısı ile tanımlanabilen bir hiperdüzlem tarafından ikiye ayrılır. Hiperdüzlemler her iki sınıfın karar hiperdüzlemine en yakın veri noktalarından geçer ve bu iki hiperdüzlem birbirine paraleldir. Bu hiperdüzlemler arasındaki mesafe karar hiperdüzleminin kalitesini belirler[6].

DVM'ler iki-sınıflı sınıflandırma yöntemi olmasına rağmen çoklu-sınıflı sınıflandırma problemlerinde de kullanılmaktadır. Çünkü günlük hayatımızda karşılaştığımız problemler karmaşık yapıdadırlar[7].

DVM'ler el yazısı tanıma, metin karakterizasyonu, imaj tanıma, tıpta meme kanseri tanıma, biyoinformatik, protein bağı tanımada sıkça kullanılmaktadır[8], [9], [10], [11], [12]. Buna rağmen DVM'lerin arıza teşhisi için kullanıldığı çalışmalar diğerleri kadar çok değildir.

1.1 Asenkron Motorlarda Arıza Teşhisi

Endüstriyel tesislerde çok sayıda asenkron motor kullanılmaktadır. Bu motorlarda zamanla elektriksel, mekaniksel ve çevre şartlarından dolayı çeşitli arızalar meydana gelmektedir. Bu arızalar zamanında tespit edilemezse, bütün sisteme zarar vererek büyük maliyette kayıplara sebep olmaktadırlar. Uygun arıza teşhis yöntemleri ve uygun izleme teknikleri kullanılarak motorlardaki arızalar tespit edilip, onarımı yapıldığında sistem büyük bir arızadan kurtarılmış olacaktır.

Asenkron motorlarda meydana gelen arızalar, dahili arıza ve harici arıza olarak sınıflandırılabilirler. Dahili motor arızalarına, motor bağlama tellerinin kısa devre olması, sarımdan sarıma kısa devreler, topraklama arızaları ve kırık rotor çubukları örnek verilebilir. Harici motor arızalarına ise faz arızası, ana destek asimetrisi, mekaniksel aşırı yüklenmeler ve tıkalı rotorlar örnek verilebilir.

Asenkon motordaki arızalar motorun üç bileşeninin herhangi birinde meydana çıkabilir. Bunlar; stator, rotor ve rulmanlardır. [1]'de motorlardaki arızalar üzerine geniş bir araştırma yapılmıştır. Araştırma 5000 motor içermektedir, bunların %97'si üç-fazlı sincap kafesi asenkron motorlarıdır. En genel arıza aşınmış motor sapmaları ile ilgilidir. Yapılan bu çalışmada arıza yüzdeleri şu şekilde verilmiştir:

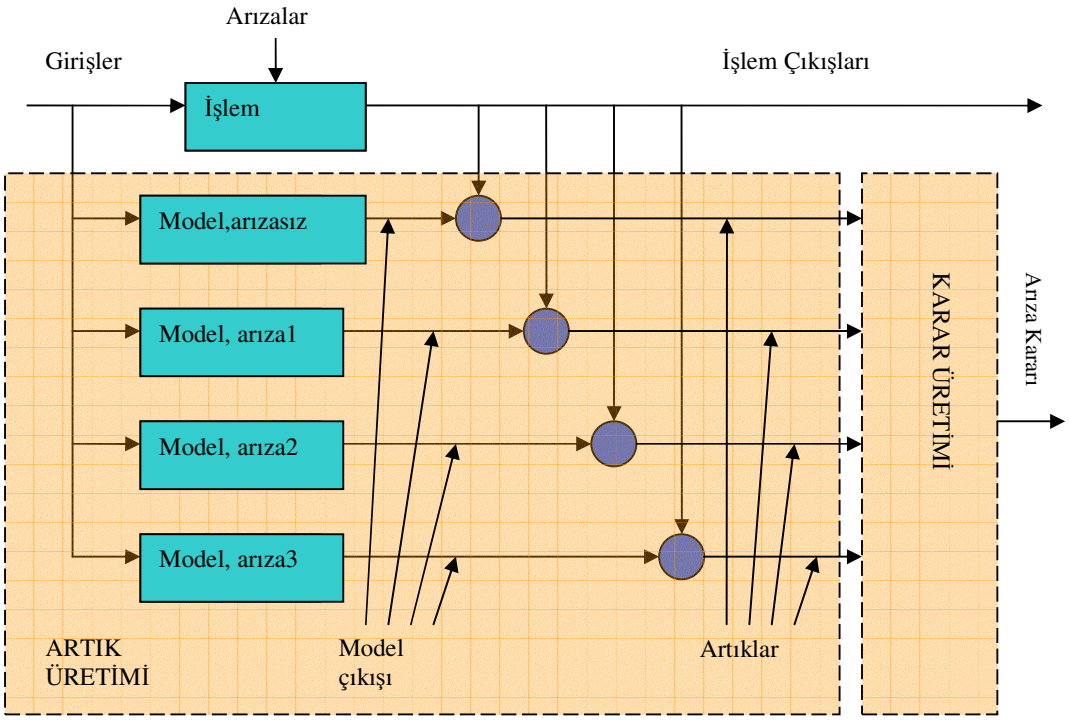
- %41 rulmanla ilgili arızalar
- %27 stator yalıtımıyla ilgili arızalar
- %10 statorla ilgili diğer arızalar
- %10 rotorla ilgili arızalar
- %12 diğer arızalar

1.2 Asenkron Motorlar İçin Arıza Teşhis Metotları

Motorlardaki arıza teşhisleri ve durum izlemesi çoğu endüstriyel işlemlerin önemli parçalarıdır. İşlemlerdeki kontrol edilemez arızalar önemli ekonomik kayıplara, işlemin performansının kalitesini düşürmeye ya da insan hayatı veya sağlığı ve çevre için ciddi hasarlara sebep olabilir. Arıza teşhisleri genellikle arıza sezme ve arıza tanımlama olarak iki kısma ayrılır. Arızaların erken teşhisi, performans ve kalite azalması ile mekanizmaya ve insan hayatına yönelik hasarı engeller. Arızaların tanımlanması sonraki onarımlar ve alarm olayları için doğru kararlar vermeye imkan sağlar. Bir çok arıza teşhisi metotları vardır ve bu teknikleri kategorize etmek için de bir çok yol mevcuttur. Arıza teşhisi tekniklerini sinyal–tabanlı ve model–tabanlı metotlara bölmek mümkündür. Sinyal-tabanlı metotlar durum görüntüleme sistemleri için temeller inşa eder ve genellikle performansları çeşitli model-tabanlı tekniklerle arttırılır [14].

Sinyal-tabanlı metotların uygulanmasının en basit örneklerden biri bir tanktaki sıvı seviyesi ve seviye belirli bir limitin üstüne çıktığında ya da altına indiğinde çalan alarmdır. Bu tür arıza teşhisi sistemleri, insanların her gün kullandığı çoğu teknik aletlerde uygulanmıştır. Geniş endüstriyel işlemlerde, ölçülmüş sinyaller daha fazla karmaşık olabilir ve işlem operasyonunun karakteristiklerini belirgin bir şekilde ortaya çıkarmak için kullanılabilirler [14].

Modern arıza sezme ve tanımlama genellikle sinyal izlemeden başlar ve çeşitli model-tabanlı metotlar tamamen bağımsız ve akıllı durum izleme sistemleri oluşturmak için uygulanır. Modeller üç sınıfa ayrılabilir. Bunlar analitik[13], veri-tabanlı[14] ve bilgi-tabanlı[15] modellerdir. Analitik modeller, fabrikalardaki arızalı bilinen fiziksel etkileyiciler üzerine kuruludur, halbuki veri-tabanlı modeller çalışılan işlemden alınan veri üzerine inşa edilmiştir. Bilgi-tabanlı modeller işlemin insan bilgilerine ve arızalarına bağlıdır. Model-tabanlı metodu Şekil 1.1'de görüldüğü gibi çizebiliriz [14]:



Şekil 1.1 Model-tabanlı arıza teşhisi şeması

1.2.1 Analitik Modeller

Analitik modeller, işlemde bilinen fiziksel etkileşimlerine bağlı olarak inşa edilirler. Bu modeller; gözlemciler, parametre tahmini ya da eşlik eşitlikleri kullanılarak uygulanır. Analitik modelde temel fikir, gözlemciler kullanılarak sistemin mevcut giriş ve çıkışlarından sistem modeli ile sistem çıkışlarını tahmin etmektir. Tahmini ve gerçek çıkışlar arasındaki fark hesaplanır ve arıza teşhisi bu ölçüm üzerine kurulur. Eşlik eşitlikleri istatistiksel ya da dinamik olabilir ve arıza teşhisi eşitliklerin çıkış örnekleri üzerine kuruludur[14].

1.2.2 Bilgi-tabanlı Modeller

Bilgi-tabanlı modeller, uzman bilgisine dayalı bir metottur. Motorun çalışma prensibi ve arıza durumunda oluşabilecek arızalar hakkında yeterli bilgisi olan uzmanlar aracılığıyla bu metot uygulanır. Bu metodun model-tabanlı metoda göre daha avantajlı olmasına rağmen burada da bazı zorluklar bulunmaktadır. Örneğin uzman kişilerin eğitilmesi ve doğru sonuçlar alınması bazen oldukça zorlaşmaktadır. Bazı yerlerde de uzman bilgisinin yanında bulanık mantık yöntemleri de ek olarak kullanılabilmektedir[15].

1.2.3 Veri-tabanlı modeller

Bir asenkron motordaki arızayı teşhis etmek için eğer analitik model kullanılacaksa, öncelikle sistemin analitik modelinin çıkarılması gerekir. Bu da bazen mümkün olmayabilir. Bilgi-tabanlı modellerde ise bazen uzman bilgisinin yeterli olmadığı zamanlar olabilir. Bu gibi durumlarda en kullanışlı yöntem bir veri-tabanlı model kullanmaktır.

Veri-tabanlı modeller, analitik formda işlemin modeli bilinmediğinde ve arıza altındaki işlemin performansına ait uzman bilgisi yetersizse uygulanır. Veri-tabanlı modeller çeşitli yollarla oluşturulabilir. En geleneksel yaklaşım zaman serileri analizidir. Zaman serileri analizi lineer modellerde sonuçlanır. Zaman serileri aynı zamanda Hammerstein modelleri, nöral ağlar ya da bulanık sistemler kullanarak nonlineer duruma genişletilebilir[14].

Arıza teşhislerinde, analitik model kullanmanın bazı zorluklarının üstesinden gelinebilir ve arıza teşhisi algoritmaları nöral ağları kullanarak gerçek sistemlere daha fazla uygulanabilir. Nöral ağlar kalanları üretmek ve bir arıza kararı vermek için kullanılabilir. Nöral ağların ana özelliklerinden biri örneklerden öğrenebilme yeteneğidir. Böylece, nöral ağlar sistemin çok sayıda ölçüm verilerini alma durumlarında sıkça kullanılır. Sistemden alınan sayısal verinin çoğu nöral ağı eğitmek için gereklidir. Eğer işlemin bütün durumlarından alınan veri yeterli değilse, güvenilir bir ağ oluşturmada zorluklar meydana gelebilir.Nöral ağların diğer bir dezavantajı, ağırlık faktörlü bir ağ mimarisinin insan tarafından oluşturulmasındaki zorluktur. Bu, sistemi ayarlamada ya da sistem operatörüne teşhis sonuçlarını açıklamada problem olabilir.

Arıza teşhisi sisteminin karar verme kısmındaki nöral ağ uygulaması nöral ağ tabanlı arıza sınıflandırması ya da örüntü tanıma olarak da adlandırılabilir. Sınıflandırma ve örüntü tanıma, özellik gibi, çoklu sayısal ölçüme bağlı şeyleri kategorize eden ya da sınıflandıran veri-tabanlı algoritmalar için genel adlardır. Sınıflandırma metotları belirgin bir artık üretme ve karar verme kısımları olmadan arıza teşhislerinde uygulanabilir. Sınıflandırıcılar sistemin ölçüm verileri arasındaki direkt ilişkiyi ve kesin arıza şartlarını göstermek için eğitilebilir. Nöral ağ tabanlı sınıflandırma modellerine ek olarak geleneksel Bayesian sınıflandırıcı, basit uzaklık tabanlı sınıflandırıcılar ya da DVM gibi sınıflandırma algoritmaları da mevcuttur [16].

DVM'ler geleneksel örüntü tanıma ve sınıflandırma sistemlerinde bize yeni fikirler vermektedir. Diğer sınıflandırıcılarla karşılaştırıldığında birçok avantajının olduğu görülmektedir. En önemli avantajı ise istatistiksel sınıflandırıcıların, çoğu zaman başarısız olduğu yüksek boyutlu sınıflandırma problemlerindeki verimidir. Bir probleme DVM yöntemini uygularken, sınıflandırıcının genelleme yeteneği giriş uzayının boyutu ne olursa olsun iki sınıfın sınırında yer alan örnek sayısı ile ölçülebilir. Aynı zamanda hesaplamalar da giriş uzayının

boyutundan bağımsızdır, çünkü giriş datası Gram matrisiyle tutulur. DVM'lerin diğer bir avantajı ise nonlineer problemlere uygulanabilirliğidir.

Bununla birlikte, DVM tabanlı sınıflandırmanın asenkron motorların arıza teşhisine uygulandığı birkaç çalışma mevcuttur[5], [17], [18], [19], [20], [21], [22], [23]. DVM farklı türdeki sınıflandırma problemlerinde iyi performans vermesine rağmen genel olarak arıza teşhisinde geniş olarak çalışılmamıştır.

1.3. Arıza Teşhisinde Destek Vektör Makinelerinin Kullanılması

DVM'ler 1960'ların sonunda V. Vapnik tarafından bulunan istatistiksel bir metottur[3]. DVM'ler yapısal risk minimizasyonu prensibi etrafında formülleştirilmiştir. Beklenen riskin üst sınırı için en küçük değeri bulmaya çalışır.

DVM'ler günümüzde birçok uygulamada kullanılmaktadır. Bu uygulamaların sonucunda da çok iyi verim elde edilmiştir.

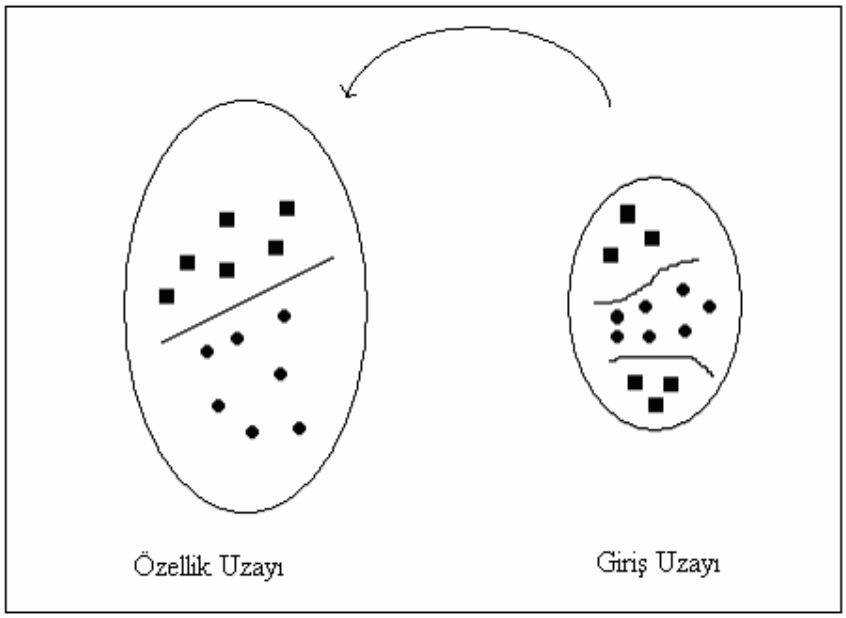
DVM'ler lineer ve non-lineer durumlarda uygulanabilir. Lineer olarak ayrılabilen durumlar için, N boyutlu bir $S = \{(x_i, y_i)\}_{i=1}^{N}$ eğitim kümesi verilmiş olsun. Burada, i=1,2,…,N için $x_i \in R^n$ ve $y_i \in \{-1, +1\}$'dir. Diğer bir deyişle S'teki örneklerin iki sınıftan birine ait olduğu varsayılmıştır.

$$f(x)=(w.x)+b \qquad (1.1)$$

(1.1) eşitliğinde iki sınıfı birbirinden ayıran hiperdüzlemin denklemi verilmiştir. Burada w ağırlık katsayısını, x giriş verisini ve b'de sabit bir sayıyı göstermektedir. Bir x_i giriş verisi için $f(x)\geq 0$ ise bu veriler birinci sınıfa, $f(x)<0$ ise veriler ikinci sınıfa aittir.

DVM'de, hiperdüzleme en yakın noktalarda bulunan giriş verilerine Destek Vektörleri (DV) adı verilmektedir. Destek vektörleri ile hiperdüzlem arasındaki mesafe ne kadar büyükse, sınıflandırma o kadar iyidir. Destek vektörleri ile hiperdüzlem arasındaki mesafeye sınır adı verilmektedir. İyi bir sınıflandırma yapabilmek için sınır değerinin büyük olması gerekmektedir.

Günlük hayatımızda karşılaştığımız problemler genellikle karmaşıktır. Bu gibi problemlere lineer sınıflandırma uygulamak doğru bir çözüm değildir [25]. Burada da, nonlineer DVM yöntemi uygulanmaktadır. Nonlineer problemlerdeki giriş verileri, Kernel fonksiyonları adı verilen fonksiyonlardan geçirilerek özellik uzayına düşürülür.



Şekil 1.2 Giriş uzayından özellik uzayına geçiş

Giriş uzayındaki verilerin sayısının çokluğu DVM'ler için önemli değildir. Bütün verileri uygun bir Kernel fonksiyonu seçildikten sonra, bu fonksiyondan çıkan veriler sınıflandırılıp, nonlineer problemler için de çözüm üretilmiş olur.

$$f(x)=(w.\phi(x))+b \quad (1.2)$$

Burada $\phi(x)$ terimi kernel fonksiyonundan geçen giriş verisini göstermektedir. Burada da aynı şekilde, eğer fonksiyonun sonucu 0'dan büyükse veri birinci sınıfa, değilse veri ikinci sınıfa aittir.

DVM'lerin arıza teşhisinde kullanılması çok yeni bir çalışmadır. Asenkron motorların arıza teşhisinde, motordan alınan veriler çeşitli özellik çıkarma algoritmalardan geçirildikten sonra, sınıflandırma için DVM yöntemi uygulanır. DVM'ler esasında 2-sınıflı bir sınıflandırma yöntemidir. Eğer motor için 2-sınıflı bir sınıflandırma uygulanacaksa bu işlem çok basittir. Fakat motor denilince birçok arıza karşımıza çıkabilir. Bu durumda da, DVM'lerin içinde bulunan çeşitli birleştirme algoritmaları kullanılarak sınıflandırma işlemi gerçekleştirilir.

DVM'lerin arıza teşhisinde kullanılması ilk olarak S. Pöyhönen tarafından gerçekleştirilmiştir[5]. Bu çalışmada çeşitli motorlardan alınan veriler çeşitli özellik çıkarma algoritmalarından geçirildikten sonra DVM'ler ve diğer yöntemler uygulanmış ve birbirleriyle karşılaştırılmıştır. Bu karşılaştırmalar sonucunda DVM'lerin hepsinden daha iyi sonuç verdiği görülmüştür. Yapılan bu çalışmada, ilk olarak 15 kw'lık bir asenkron motor kullanılmış ve motor akım imza analizi ile veriler işlenmiştir. Daha sonra da bu verilere DVM yöntemi

uygulanıp sınıflandırma yapılmıştır. Diğer bir çalışmada, verilere gürültü verisi de eklenip o şekilde sınıflandırma yapılmıştır. Ve ortamda gürültü olsa bile DVM yönteminin çok iyi sonuçlar verdiği görülmüştür. Başka bir çalışmada ise 35 kW'lık bir motor kullanılmıştır. İkiden fazla durum söz konusu olduğu için bir karışım matrisi birleştirme planı kullanılmıştır. Yapılan bütün bu çalışmalarda DVM'lerin çok iyi sonuçlar verdiği görülmüştür.

Bu tez çalışmasında asenkron motorlarda oluşan arızalar incelenip, arıza teşhisi için DVM yöntemi uygulanmıştır. Diğer model-tabanlı ve bilgi-tabanlı teşhis yöntemlerinde çeşitli dezavantajlar bulunmaktadır. Mesela bir sistemin matematiksel modelinin çıkarılması bazen çok zor olabilir. Ve bazı durumlarda da uzman bilgisi arıza teşhisi için yeterli olmayabilir. DVM'lerde sadece giriş ve çıkış verisi kullanılarak çok büyük bir başarı oranıyla arıza teşhisi yapılabilmektedir.

1.4. Asenkron Motorların Arıza Teşhisinde DVM'lerin Kullanılması

Asenkron motorlar endüstride çok geniş yer kaplamaktadır. Bu nedenle, asenkron motorların arıza teşhis ve tespit işlemleri önemli yer tutmaktadır. Birçok araştırmacı bu konuda büyük çalışmalar yapmışlardır. Gerek analitik modeller gerekse uzman bilgisine dayalı modeller günümüzde çok sık kullanılmaktadır. Fakat her iki modelin de kendine göre dezavantajları bulunmaktadır.

DVM'ler, birçok sınıflandırma probleminde çok iyi sonuçlar vermesine rağmen arıza teşhisi için pek kullanılmamıştır. Asenkron motorların arıza teşhisi için DVM'lerin kullanıldığı sadece birkaç çalışma bulunmaktadır. Bu çalışmalarda da diğer yöntemlere göre DVM'lerin çok daha iyi sonuçlar verdiği görülmüştür[5], [17], [18], [19], [20], [21], [22], [23].

Motorlardaki arıza teşhisi için çeşitli sinyaller kullanılabilmektedir. Motordan, hız, moment, akım, titreşim ve gerilim sinyalleri alınabilir. Yalnız alınan bu sinyallerin direkt olarak kullanılması uygun değildir. Alınan sinyaller çeşitli algoritmalardan geçirilip, uygun veriler seçilerek arıza teşhisi yapılabilir.

Arıza teşhisi konusunda yapılan birçok çalışma vardır. Asenkron motorlarda arıza teşhisi için DVM kullanılan çalışmalar S. Pöyhönen tarafından yapılmıştır [2]. Pöyhönen, çeşitli motorlardan alınan verileri belirli özellik çıkarma algoritmaları kullanarak DVM'lerle sınıflandırıp arıza teşhisi yapmıştır.

Pöyhönen ve diğ. çalışmalarında, motor imza analizi yaklaşımı takip edilmiştir [17], [18]. Her iki makalede de, 15 kW'lık bir asenkron motorun stator hat akımının güç spektrum yoğunluğunun tahminleri bir orta arıza bulma olarak kullanılmıştır. [17]'de, yeni DVM tabanlı sınıflandırıcılar sağlıklı verileri arızalı verilerden ve arızalı verileri de birbirinden ayırt etmek

için eğitilmiştir. [18]'de, DVM çiftlerinin çıkışları basit bir çoğunluk oylaması yaklaşımı ile kaynaştırılmıştır. DVM uygulaması bütün güç spektrum yoğunluğu tahminlerini sınıflandırmada ölçümlerin küçük bir miktarı yerine özellik vektörü olarak kullanımını mümkün yapmıştır. Altı farklı arıza, motorun sağlıklı işlemesine ek olarak çalışılmıştır. Sayısal manyetik alan analizi, motordan sanal ölçüm verisi sağlamak için kullanılmıştır. Motorun stator akımının yoğunluğu tahminleri Welch metodu ile hesaplanmıştır. Arızaların çoğu birbirinden doğru olarak ayrılmıştır. Destek vektör sayıları yüzdelerinden, sınıflandırmanın başarılı olduğu görülmektedir. [18]'de aynı zamanda gürültünün etkisi de çalışılmıştır. DVM, bu çalışmalardan önce asenkron motorda arıza teşhisi için uygulanmamıştır. Bu çalışmalarda gürültüsüz sınıflandırma yapısı iyi işlemektedir, fakat gürültü toplam sınıflandırma oranını alçaltmaktadır. Gürültü filtreleme ile arıza bulma oranı sadece biraz artmaktadır. Bu, stator hat akımının bir arıza teşhis ortamı için en iyi seçim olup olmadığı sorusunu büyütür. Bu geniş olarak kullanılmıştır, çünkü akım ölçümleri motora ulaşmayı gerektirmez, ama yine de akım verimli bir arıza teşhis ortamı olarak uygulanacak motor arızaları hakkında yeterince bilgi içermeyebilir.

Gürültüsüz durumda 2-sınıflı sınıflandırıcının çıkışındaki arızalar, sonuç çok-sınıflı sınıflandırıcının tekrar yapılandırılmasında toplanır. 2-sınıflı sınıflandırıcıların dört farklı birleştirme tekniği motor durumunun global kararını almak için çalışılmıştır [19]. Gereği gibi ayarlanan nöral ağ birleştirme, en doğru çok-sınıflı sınıflandırma sonucuyla sonuçlanır, ama karışım matrisi gibi daha basit bir tekrar yapılandırma yaklaşımı uygulanabilir. Bu uygulamada, lineer birleştirme tekrar yapılandırma planı için pratik bir seçimdir, çünkü bir nöral ağı eğitmek ve ayarlamak yorucu bir iştir. 2-sınıflı sınıflandırıcıların farklı birleştirme tekniklerinin karşılaştırılması sınıflandırma metoduna karar vermede önemli bir iştir.

Pöyhönen ve diğ. yaptıkları çalışmada, DVM tabanlı sınıflandırmayı bir 35 kW kafes asenkron motorun ve slip-ring üretecinin arıza teşhisine uygulamıştır [20]. 2-sınıflı DVMleri birleştirme için bir karışım matrisi planı kullanılmıştır. Stator hat akımı, paralel kollar arasında dolaşan akımlar ve makinenin rotoru üzerinde etki eden kuvvetler arıza göstergesi olarak karşılaştırılmıştır. Rotor üzerindeki kuvvetler ve paralel kollar arasında dolaşan akımların, stator akımına göre arızaları daha iyi gösterdiği görülmüştür. Kuvvetlerin ölçümü zordur, fakat bunlar da asenkron motorun durum izlemesinde geniş olarak kullanılan ölçülebilir motor titreşimleriyle doğrudan alakalıdır.

Pöyhönen ve diğ. bu çalışmada da motor imza analizi yerine titreşim izleme çalışmışlardır [21]. Titreşim sinyalinin birkaç özelliği 35 kW asenkron motorun kırık rotor milinin göstergesi olarak karşılaştırılmıştır. Arıza bulma işi ve özellik karşılaştırması DVM tabanlı sınıflandırma ile gerçekleştirilmiştir. Özellik çıkarma için en iyi metot otoragresif

modelinin katsayıları uygulaması olarak görünmektedir. Sonuçlar çeşitli motor durumlarından ve yük durumlarından alınan gerçek ölçümlerden alınmıştır.

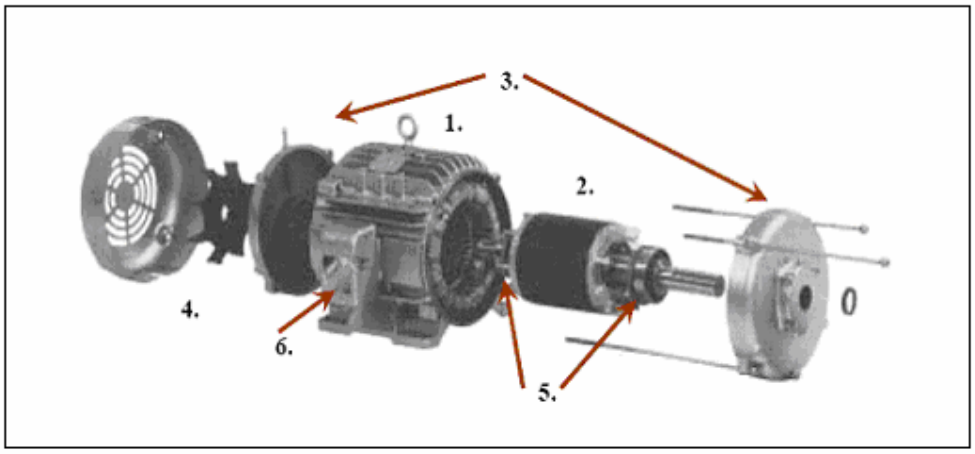
Aynı zamanda titreşim izlemeyle ilgili bir çalışma daha mevcuttur [22]. Motorun titreşimleri çoklu sensörlerle ölçülmüştür. Bu, bağımsız bileşen analiziyle (Independent Component Analysis-ICA) gerçekleştirilmiştir. Sadece güç spektrum yoğunluğu tahmini özellik çıkarma aracı olarak kullanılmıştır.

Pöyhönen ve diğ., yaptıkları çalışmada asenkron motor arıza teşhisi için farklı bir görüş olarak yalıtım sistemlerinin durum izlemesini çalışmışlardır [23]. Kısmi deşarj ölçümleri farklı yüksek ve orta gerilim aparatlarında yalıtım durumunun teşhisi için geniş olarak kullanılmıştır. İlk olarak, çeşitli parametreler kısmi deşarj dağıtımlarından çıkarılmıştır ve istatistiksel analiz yalıtım arızasının yerini bulmak için iyi parametreler bulmak için çalışılmıştır. k-en yakın komşu sınıflandırıcıyı (k-NN), olasılıksal nöral ağı ve DVMyi karşılaştırdığımız zaman en iyi sonuçlar DVM ile elde edilmiştir.

2. ASENKRON MOTORLAR VE ARIZA TÜRLERİ

Asenkron motorlar basitliklerine, sağlam yapılarına, ucuzluklarına ve kolay monte edilebilirliklerine bağlı olarak endüstride çok kullanılırlar. Asenkron motorların devir sayıları yükle çok az değişir, bu motorlar sabit devirli motorlar sınıfına girerler. Doğru akım şönt motorlarında devir sayısı büyük sınırlar içinde değiştirilebilir. Halbuki asenkron motorun devir sayısı sınırlı olarak bir veya iki kademeli olarak değiştirilebilir. Asenkron motorlara indüksiyon motorları da denilmektedir. Asenkron motorların çok sık tercih edilmelerinin sebepleri şu şekilde sıralanabilir:

- Asenkron motorlar bakıma çok az ihtiyaç gösterirler.
- Asenkron motorlar ucuzdur.
- Asenkron motorların çalışmaları sırasında elektrik arkı meydana gelmez.
- Asenkron motorlar yüklü çalışırken devir sayıları çok fazla değişmez.



Şekil 2.1 Asenkron motorun yapısı (1) stator, (2) rotor, (3) sonlandırma halkaları, (4) fan, (5) motor mili, (6) terminal kutusu

Şekil 2.1'de bir asenkron motor ve parçaları gösterilmiştir. Asenkron motorun en dışında stator bulunmaktadır. Stator asenkron motorun duran kısmıdır. 0.4, 0.5 veya 0.8 mm kalınlığında silisyumlu demir saclar, özel kalıplarla preste basılır. Sargılar stator üzerine, üçgen

veya yıldız bağlı olarak iki şekilde yerleştirilebilir. Stator sargılarından geçen akım alternatif akım olduğundan manyetik devrede periyodik olarak değişen bir alternatif alan oluşturur. Alternatif alanın her bir harmoniği iki döner alana ayrılabilir. Bu döner alanlardan biri saat ibresi yönünde dönüyorsa diğeri saat ibresinin tersi yönünde döner ve her ikisinin de açısal dönme hızı aynıdır.

Asenkron motorların diğer bir parçası da rotordur. Rotor, mil yatağı üzerine monte edilmiştir ve üzerinde bakır çubuklar bulunmaktadır. Asenkron motorlarda rotor, motorun tipini belirler. Buna göre asenkron motorlar sincap kafes veya sargılı biçimde olmak üzere ikiye ayrılır. Sincap kafes asenkron motor aynı zamanda kısa devreli, sargılı motor ise bilezikli olarak adlandırılabilir.

Bir asenkron motorun çalışma şekli manyetik alanla ilgilidir. Kalıcı bir mıknatısa sahiptir. Mıknatısın içine yerleştirilmiş sincap kafes rotor bir mil yatağı ile sabitlenmiştir. Rotor ile mıknatıs arasında herhangi bir sürtünme söz konusu değildir. Rotorun dönmesi tamamen arada oluşan manyetik alan ile alakalıdır. Sincap kafes rotor, sonlandırıcı halka veya plakalar ve bakır çubuklardan yapılır. Rotor çubukları kapalı bir devre oluşturmak amacıyla sonlandırıcı halkalara kaynaklanmıştır. Mıknatısın dönme yönü ile rotorun dönme yönü aynıdır. Mıknatısın dönme hızı yavaşlatıldığında veya hızlandırıldığında rotor da bu harekete doğru orantılı olarak davranır [24].

Mıknatıs dönmeye başladığında mıknatısın iki kutbu arasındaki akı rotor çubukları tarafından kesilir. Rotor çubuklarında bir gerilim indüklenir ve bu gerilim Faraday kanunları ile verilir. Gerilim, kapalı devrede bir akım oluşturur. Akım taşıyan her bir rotor çubuğu etrafındaki manyetik alan kalıcı mıknatısın manyetik alanını etkiler. Bir elektromanyetik dönme momenti üretilir ve bu moment rotoru mıknatısın dönme yönünde dönmeye zorlar [24].

Üç-fazlı bir asenkron motorun stator sarımı üç-fazlı bir güç kaynağına bağlı olduğunda sabit bir büyüklüğü olan bir manyetik alan üretir ve rotorun etrafında senkron hızda döner. Eğer *f* stator sarımındaki akımın frekansı ve *P* çubuk sayısı ise, dönen alanın senkron hızı:

$$\omega_s = (4\pi f / P) \tag{2.1}$$

olarak hesaplanır.

Dönen alan rotor sarımı içinde elektromotor güce (EMF'ye) neden olur. Rotor sarımı kapalı bir döngü oluşturana kadar, her bobinde oluşan Elektromotor güç, o bobin içerisinde oluşan bir elektromotor güce artış verir. Akım-taşıyan bir bobin manyetik alan içinde olduğu zaman kendisini döndürmeye çalışan bir güçle karşılaşır. Rotor gücünü, rotor hızı ve dönen alan arasındaki bağlı devinim varsa, indüksiyon ile alır. Rotor dönen alanın senkron hızından daha düşük bir hızla döndüğü için bir indüksiyon motoru asenkron motor olarak da bilinir.

Stator ve rotor arasındaki bağıl hız kayma hızı olarak da adlandırılır. Eğer rotor hızı ω_m ise kayma hızı $\omega_r = \omega_s - \omega_m$'dir. Kaymanın terimleri içinde kayma hızını göstermek için, s, kullanılır ki bu da kayma hızının senkron hıza oranıdır.

$$s = \omega_r / \omega_s \quad (2.2)$$

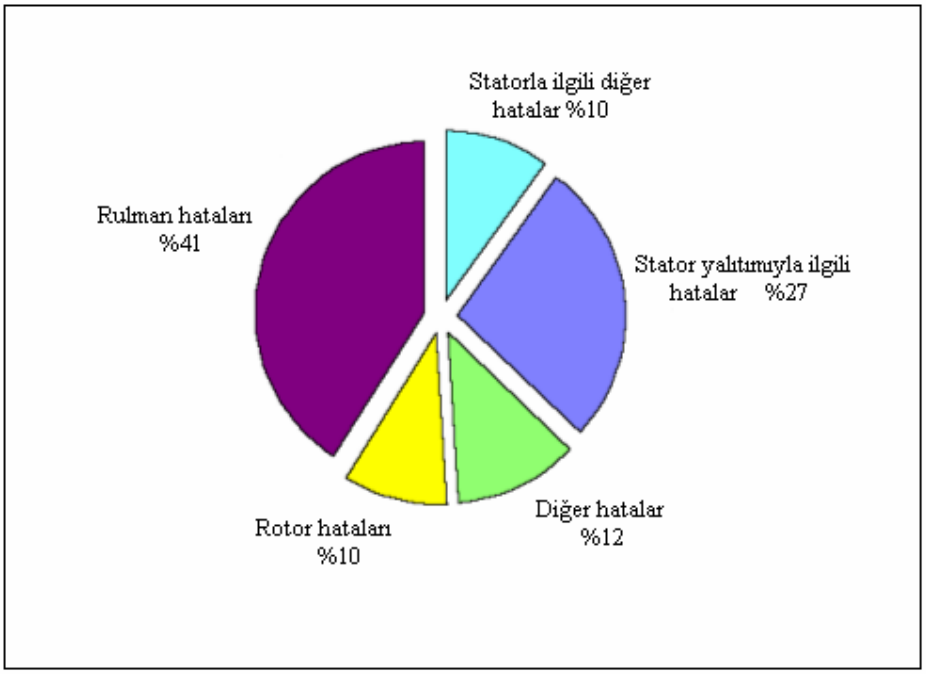
Asenkron motorlardan daha çok kullanılan tip sincap kafesli asenkron motorlardır. Sincap kafesli asenkron motorlarda her bir rotor çubuğu bir sonlandırıcı halka ile sonlandırılıp kapalı devre bir sistem oluşturulmuştur. Rotor çubukları genellikle slotlarda yerleştirilen magnezyum, alüminyum veya bakırdan oluşur.

2.1. Asenkron Motorlarda Meydana Gelen Arızalar

Asenkron motorda herhangi bir arızanın habercisi performans düşüklüğüdür. Arıza meydana gelmeden motorun performansının izlenmesi gerekmektedir. Performansın sürekli izlenmesi durum izleme olarak adlandırılmaktadır. Durum izlenmesinde, arıza oluşmadan önce motordaki sorunların ortadan kaldırılması, analiz ve arızaların belirlenmesi söz konusudur.

Asenkron motorlarda arıza meydana gelmeden önce sistemin izlenmesi ve bazı küçük problemlerin giderilmesi maliyet yönünden de avantaj getirmektedir. Motordaki donanımsal yapının izlenip, arızaların giderilmeye çalışılması hem daha zor hem de daha pahalıdır. Bu nedenle bazı arıza teşhis algoritmalarıyla sistemin kontrolü daha verimli olmaktadır. Genellikle motorun durum izlemesinde, motor akımı, hız, gerilim ve moment gibi gerçek zamanda ölçülebilen ve daha düşük maliyetli olan verilerin kullanılması daha faydalıdır.

Motorlardaki arızalar üzerine yapılan bir çalışmada motorlar ve arızaları hakkındaki bilgiler şu şekildedir [1]: Araştırmada 5000 motor kullanılmıştır ve bunların %97'si üç-fazlı sincap kafesli asenkron motorlardır. Şekil 2.3'te araştırmaya bağlı olan ve birbirinden ayrı arızaların meydana gelişi gösterilmiştir. En genel arıza aşınmış motor sapmaları ile ilgilidir ve bu ekstra titreşimler, gürültü ve rotor milinin olası uyuşmazlığına sebep olur. Statorla ilgili çoğu arızalar, bir kısa devre, fazdan-faza ya da fazdan-toprağa kısa devrelerine yol açan stator sarımlarındaki alçaltılmış yalıtımlara bağlıdır. Bunlar tam makine arızasıyla sonuçlanabilecek ciddi arızalardır. Rotor arızaları, rotor ekzantrikliği ve rotorun fiziksel hasarıyla ilgili olarak ikiye ayrılabilir ve bunlar genellikle yavaş gelişir, sonuçta kırık rotor barları stator sarımlarına zarar verebilir.



Şekil 2.2. Motorlarda meydana gelen arızalar ve yüzdeleri [1].

Asenkron motorlarda en çok oluşan arıza türü rulman (mil yatağı) arızalarıdır. Mil yatağının eğilmesi, milin sürtünmesi, mil yatağındaki bilyelerde oluşan arızalar, en çok karşılaşılan mil yatağı arıza türleridir. Mil yatağının eskimesi, yağ sızıntısı ve yağın eskimesi ile mil yatağında sürtünme arızaları meydana gelir [14]. Oluşan bu sürtünmeler erken bir aşamada giderilmezse mil yatağının kilitlenmesine ve motorun yanmasına sebep olur. Mil yatağında bilyeler iç ve dış halka olmak üzere iki bölümden oluşur. Bu halkalar içindeki bilyeler motorun dönmeye başlamasıyla dönerler [24].

Asenkron motorlarda oluşan bir diğer arıza da rotor arızalarıdır. Rotorlarda meydana gelen arızalar ise kırık rotor ve kırık sonlandırıcı halkalardır. Bu arızalar genellikle elektromanyetik, termal, çevresel ve mekaniksel sebeplerden oluşur. Rotorlardaki fiziksel arızalar ise rotoda meydana gelen statik ve dinamik ekzantrikliktir. Motorun bulunduğu ortamdaki nem miktarı, havalandırma eksikliği kırık rotor arızalarına neden olmaktadır. Mil yatağında oluşan sürtünme ve milin eğilmesi gibi mekanik arızalar da rotorda arızalanmaya neden olur [24].

Asenkron motorlarda stator ile rotor arasında belirli bir mesafe bulunmaktadır. Bu belirli mesafenin bozulmasıyla aradaki denge bozulur ve hava boşluğu ekzantrikliği arızaları meydana gelir. Hava boşluğu ekzantrikliği statik ve dinamik olmak üzere ikiye ayrılır. Minimum hava boşluğu ekzantrikliğinin oluşma aşamasında dengesiz bir manyetik çekim oluşur ve bu çekim rotorun kötüleşmesine neden olur [24].

Bir asenkron motorda bazı parametreler için belirlenen değerlerin dışına çıkıldığında bazı aksaklıklar meydana gelir. Asenkron motorlardaki statorla ilgili arızalar önemli bir yer tutmaktadır. Asenkron motorlarda stator sargılarında meydana gelebilecek bir kısa devreyi önlemek için yalıtım yapılmıştır. Bazen bu yalıtımlar da yetersiz kalmaktadır. Böyle durumlarda, stator sargılarının belirli kısımlarında incelme, yalıtım kayıpları ve çatlamalar meydana gelir.

Literatürde asenkron motorlardaki arıza çeşitlerini belirlemek için çeşitli yöntemler kullanılmıştır. Motordan alınan akım, titreşim veya gerilim gibi veriler çeşitli işaret işleme teknikleri kullanıldıktan sonra teşhis için uygun algoritmalar kullanılıp araştırılmıştır[5], [17], [18], [19], [20], [21], [22], [23].

3. DESTEK VEKTÖR MAKİNELERİ VE ÖĞRENME METODOLOJİSİ

3.1. Lineer Öğrenme Makineleri

Makinelerin öğrenebilme yetenekleri uzun yıllardan beridir insanların merakını alan bir konudur. Sistemlerin öğrenebilme yetenekleri, herhangi bir matematiksel modeli olmayan problemlerin çözümünde önemli bir yer tutmaktadır. Mesela el yazısı tanıma, bir DNA serisindeki genleri bulma gibi problemlerde öğrenebilen makineler kullanılmaktadır [8], [12].

Bilgisayarların bir problemi çözmesi için yapılması gereken şey, problemin nasıl çözüleceğini bilgisayara komutlarla öğretmektir. Yani bilgisayarın, adım adım takip edebileceği bir komut serisi olmalıdır.

Bununla birlikte, günümüzde bilgisayarların çözmesi gereken çok daha karmaşık işler bulunmaktadır. Karmaşık işlerden kasıt, bilgisayarın hangi durumlara hangi sonuçları vereceğini bilmemesidir. Bu tip durumlara örnekler; kompleks bir kimyasal reaksiyonu modelleme, bir DNA serisine bağlı protein tiplerinin sınıflandırılması olarak verilebilir.

Bu tip işleri, geleneksel programlama yaklaşımıyla çözmek mümkün değildir. Çünkü sistem tasarımcısı giriş verisini kullanarak doğru çıkışı nasıl bulacağını bilmeyebilir. Bu tip problemleri çözmek için alternatif bir strateji, örneklerden giriş-çıkış fonksiyonelliğini öğrenmeye yönelim olabilir. Programların sentezi için örneklerin kullanılması yaklaşımı öğrenme metodolojisi ve belirli durumlarda örnekler giriş-çıkış çiftleri ise buna da yönlendirilmiş öğrenme adı verilir. Giriş-çıkış örneklerinin işlevselliği eğitim verisi olarak bilinir.

Yönlendirilen öğrenmede, öğrenme makinelerine bir eğitim seti verilir. Genellikle örnekler nitelik vektörlerinin formu olurlar ve dolayısıyla giriş uzayı, R^n'nin bir alt kümesidir. Eğer nitelik vektörleri elde mevcutsa, problem için hipotezler kümesinden birkaç tane seçilebilir. Bununla beraber en iyi anlaşılan ve klasik nöral ağ literatürü iki sınıf örneği arasındaki farkı göstermek için ve interpolasyon için lineer fonksiyonlar geliştirmiştir.

3.1.1. Lineer Sınıflandırma

İkili sınıflandırma sıklıkla bir $f:x \subseteq R^n \rightarrow R$ gerçek-değerli fonksiyonu şu şekilde uygulanarak yapılır: $x=(x_1, x_2, \ldots, x_n)$ girişi eğer $f(x) \geq 0$ pozitif sınıfa, değilse negatif sınıfa atanmıştır. $f(x)$, $x \in X$'in bir lineer fonksiyonu ise,

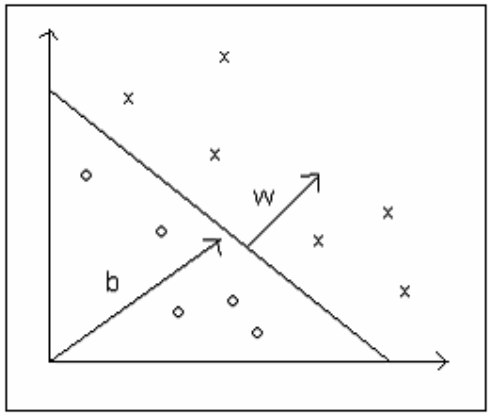
$$f(x) = \langle w.x \rangle + b$$

$$f(x) = \sum_{i=1}^{n} w_i x_i + b$$

(3.1)

olarak yazılabilir.

Burada (w,b) ∈ R^n * R parametreleri, fonksiyonu ve sgn(f(x)) ile verilen karar kuralını kontrol eder (sgn(0)=1). Öğrenme metodolojisi bu parametrelerin, veriden öğrenilmesi gerektiğini belirtir.



Şekil 3.1. İki boyutlu eğitim seti için bir (w,b) ayırıcı hiperdüzlemi

Bu tür bir varsayımın geometrik bir açıklaması X giriş uzayının <w.x> + b = 0 eşitliği ile tanımlanan hiperdüzlem ile ikiye bölündüğüdür. Bir hiperdüzlem iki ayrı sınıfı barındıran uzayı ikiye böler. Şekil 3.1'deki hiperdüzlemin üst kısmı pozitif bölgeyi , alt kısmı ise negatif bölgeyi gösterir. x vektörü hiperdüzleme dikey bir yön tanımlar. b'nin değişen değeri hiperdüzlemi kendisine paralel taşır.

Hem istatistikçiler hem de nöral ağ araştırmacıları bu basit sınıflandırıcıyı lineer diskriminantlar ve perseptronlar olarak adlandırarak kullanırlar. Lineer diskriminant teorisi Fischer tarafından 1936'da geliştirilmiştir, nöral ağ araştırmacıları ise Rosenblatt'ın araştırması nedeniyle 1960'larda perseptronları çalışmışlardır. w ve b nicelikleri nöral ağ literatüründeki gibi ağırlık vektörü ve bias (eğilim,verev) olarak kullanılacaktır

Tanım: X giriş uzayını ve Y çıkış alanını gösterecektir. Genellikle $x \subseteq R^n$, ikili sınıflandırma için Y={-1,+1} , m-sınıflı sınıflandırma için Y={1 , 2 , ... , m} ve regresyon için $Y \subseteq$ R'dir. Bir eğitim kümesi eğitim örneklerinden oluşmuştur ve bunlar eğitim verisi olarak adlandırılır. Genellikle

$S=((x_1 , y_1) , ... , (x_n , y_n)) \subseteq (X*Y)^n$ olarak belirtilir.

n: örnek sayısı

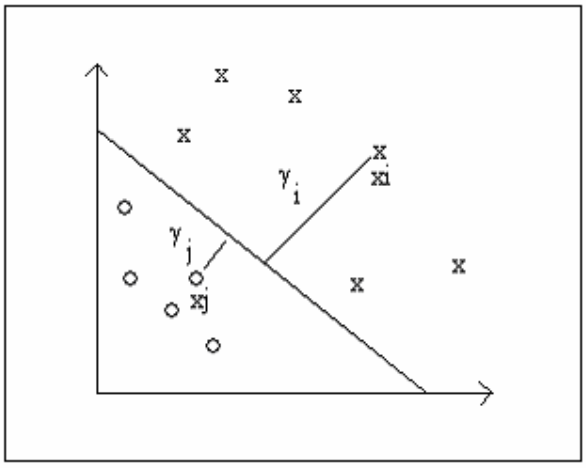
x_i 'ler örnek, y_i'ler ise etiketleri olarak gösterilir. S eğitim kümesi, eğer tüm örneklerin etiketleri eşit ise önemsizdir. Eğer x bir vektör uzayı ise giriş vektörleri ağırlık vektörleri olarak sütun vektörleridir.

Tanım: Bir (x_i, y_i) örneğinin sınırı (fonksiyonel), (w,b) düzlemi için

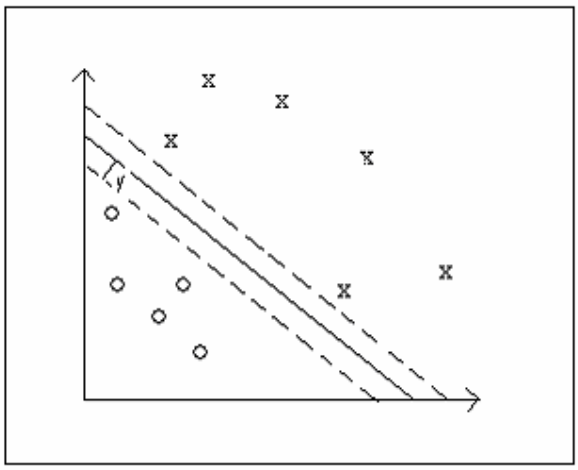
$$\gamma_i = y_i(\langle w.x_i \rangle + b) \tag{3.2}$$

şeklinde tanımlanır.

$\gamma_i > 0$, (x_i, y_i)'nin doğru sınıflandırmasını ifade eder. S eğitim kümesinin sınırı, bütün hiperdüzlemlerin üzerindeki maksimum geometrik sınırdır. Bu en büyük değeri gerçekleştiren hiperdüzlem, en büyük sınır hiperdüzlemi olarak bilinir. Kendisinin sınırının boyutu, lineer ayrılabilen eğitim kümesi için pozitif olacaktır. Şekil 3.2 iki boyutlu bir hiperdüzlem yönünden iki noktadaki geometrik sınırı gösterir. Geometrik sınır , eğer ağırlık vektörü birim vektör ise fonksiyonel sınıra eşit olacaktır. Şekil 3.3 bir eğitim kümesinin sınırını gösteriyor.



Şekil 3.2. İki noktanın geometrik sınırı



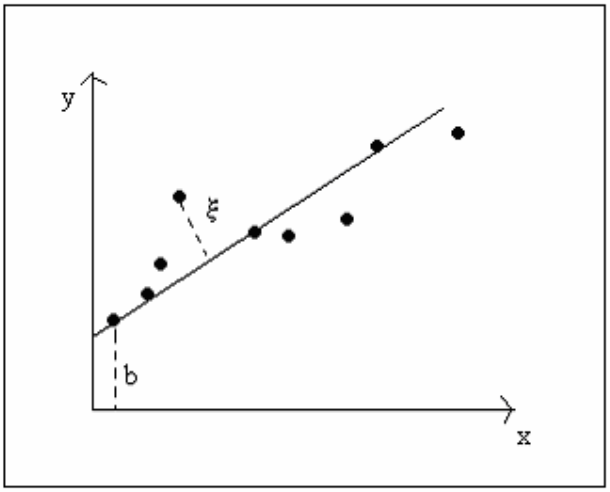
Şekil 3.3. Eğitim kümesinin sınırı

3.1.2. Lineer Regresyon

Lineer regresyon problemi Y⊆R'den etiketlenen verilmiş eğitim noktalarının S kümesini en iyi interpole eden bir

$$f(x) = \langle w.x \rangle + b \tag{3.3}$$

lineer fonksiyonunu bulmada oluşur. Geometrik olarak verilen bu noktalara uygun bir hiperdüzleme uyar. Şekil 3.4 bir-boyutlu bir lineer regresyon fonksiyonu gösterir. Şekilde ξ olarak gösterilen uzaklık belirli eğitim örneği için olan arızadır.



Şekil 3.4. Bir-boyutlu bir lineer regresyon fonksiyonu

3.1.3. Kernel Tabanlı Özellik Uzayı

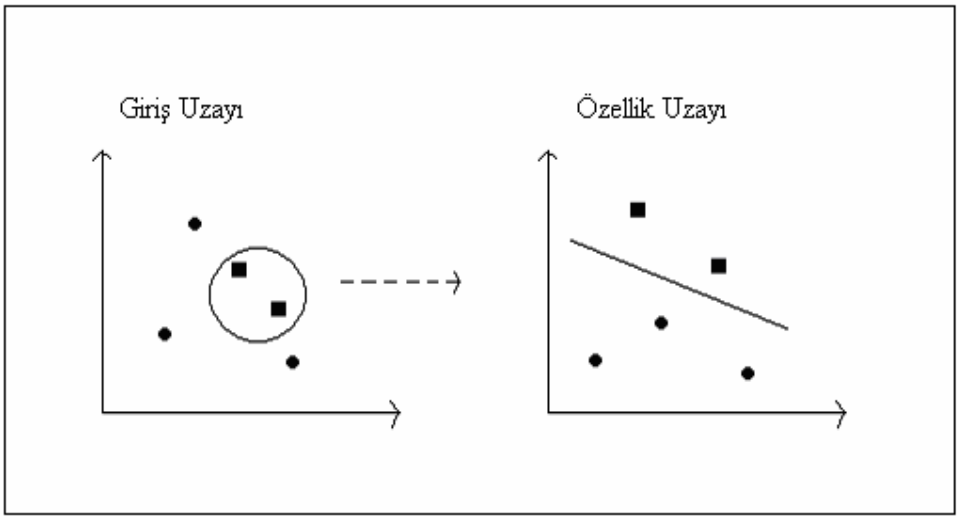
Kompleks gerçek-dünya uygulamaları lineer fonksiyonlardan daha anlamlı varsayım uzayları gerektirir. Bu problemi görüntülemenin diğer bir yolu; hedef konsept verilen niteliklerin basit lineer kombinasyonu olarak ifade edilemez. Çok katmanlı eşikli lineer fonksiyonlar bu probleme bir çözüm olarak düşünüldü ve bu yaklaşım çok-katmanlı nöral ağların ve geri-yayılımlı böyle sistemlerin eğitimi için öğrenme algoritmalarının gelişimine ön ayak oldu.

Kernel ifadeleri, lineer öğrenme makinelerinin hesapsal gücünü artırmak için veriyi yüksek boyutlu bir özellik uzayına düşürerek alternatif bir çözüm sunar. Lineer makinelerin ikili ifadedeki kullanımı bu adımı kesinlikle gerçekleştirmeyi mümkün kılar.

Kernel teknikleri DVM'lerin ana taşlarından biridir.

3.1.4. Özellik Uzayları ve Kerneller:

Şekil 3.5, giriş uzayından özellik uzayına dönüşümü temel alan DVM'lerin temel fikrini göstermektedir.



Şekil 3.5. Giriş uzayından özellik uzayına dönüşüm

$$\Phi : R^N \rightarrow F$$
$$K(x, y) = (\Phi(x).\Phi(y)) \qquad (3.4)$$

Eğer F yüksek-boyutluysa yukarıdaki eşitliğin sağ tarafının hesabı çok uğraştırıcı olacaktır. Bununla birlikte hesabı verimli kılan kerneller mevcuttur. Örneğin polinomal kernel;

$$K(x, y) = (x.y)^d \quad \text{(polinomal kernel)} \tag{3.5}$$

d=2 ve x,y$\in R^2$ için, örneğin

$(x, y)^2 = (\Phi(x).\Phi(y))$ oluyorsa;

$\Phi(x) = (x_1^2, \sqrt{2x_1x_2}, x_2^2)$ olacaktır. (3.6)

Polinomal kernelin yanı sıra en çok kullanılan kerneller sigmoid kernel, RBF (Radial Basis Function) kernelleridir.

$$K(x, y) = \tanh(\kappa(x.y) + \Phi) \quad \text{(Sigmoid Kernel)} \tag{3.7}$$

$$K(x, y) = \exp(-\|x - y\|^2 / (2\sigma^2)) \quad \text{(RBF Kernel)} \tag{3.8}$$

3.1.5. Özellik Uzayında Öğrenme

Öğrenilecek hedef fonksiyonun karmaşıklığı gerçekleştirme yoluna dayanır ve bundan dolayı öğrenme işinin zorluğu değişir. İdeal olarak spesifik öğrenme problemine uyan ifade seçilebilir. Dolayısıyla makine öğrenmesinde yaygın bir önişleme stratejisi verinin ifadesinde değişim içerir.

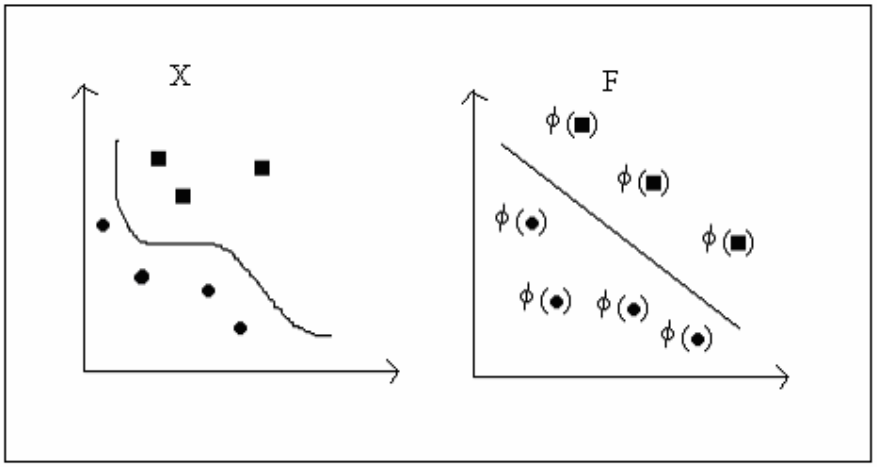
$$x = (x_1, ..., x_n) \rightarrow \Phi(x) = (\Phi_1(x), ..., \Phi_N(x)) \tag{3.9}$$

(3.8) ifadesi X giriş uzayını yeni bir uzaya haritalamaya eşittir.

$$F = \{\Phi(x) | x \in X\} \tag{3.10}$$

Veriyi başka bir uzaya haritalamak, makine öğrenmesindeki işi çok basitleştirir ve verinin en iyi ifadesinin seçimi için teknik sayısını artırır. Veriyi anlatmak için tanıtılan nicelikler genellikle özellik olarak adlandırılır, en uygun ifadeyi seçme işi ise özellik seçimi olarak bilinir. X uzayı giriş uzayı olarak alınır ve $F = \{\Phi(x) | x \in X\}$özellik uzayı olarak adlandırılır.

Şekil 3.6 iki boyutlu bir giriş uzayından , iki boyutlu bir özellik uzayına haritalamayı gösteren bir örnektir. Veri giriş uzayında lineer bir fonksiyonla ayrılamazken , özellik uzayında ayrılabilir.



Şekil 3.6. Sınıflandırma işini basitleştiren bir özellik haritası

3.1.6. Genelleme Teorisi

Vapnik ve Chervonenkis (VC) teorisi DVM'leri açıklamak için en uygun yoldur.

3.1.7. Vapnik ve Chervonenkis (VC) Teorisi ve VC Boyutu

Lineer ve nonlineer DVM ifadelerinde, sınır (margin) ifadesinin büyük bir rol oynadığı görülür.

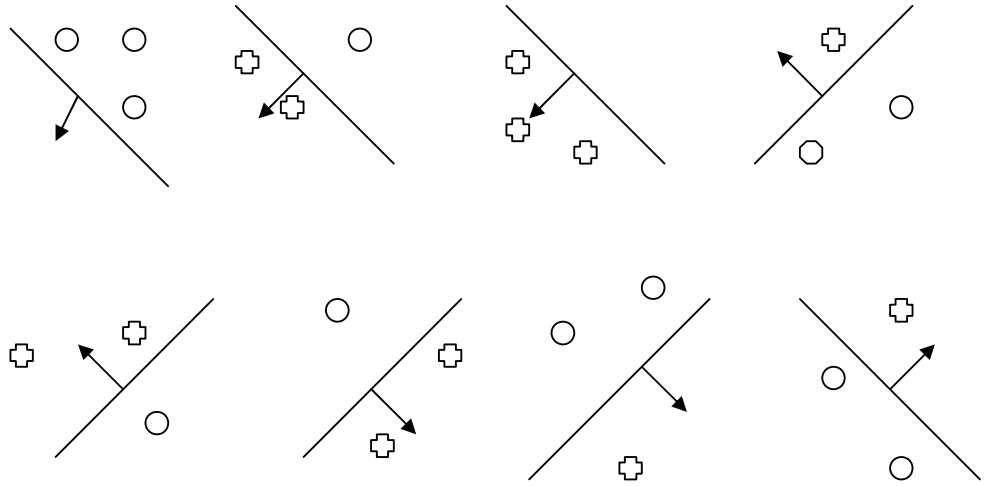
Klasik nöral ağlarda spesifik onaylama kümeleri seçilirken , VC teorisinin ana hedefi spesifik data kümeleri üzerindeki arıza yerine genelleme arızasını karakterize etmektir[37].

VC boyutu genellikle h ile belirtilir. VC boyutu sınıflandırma fonksiyonlarının kapasitesini ölçer. Fonksiyonlar kümesi için bir VC boyutu h ile parçalanan eğitim örneklerinin maksimum sayısı olarak tanımlanır.

$$i_F(x,w) \in \{0,1\} \text{ ya da } i_F(x,w) \in \{-1,1\} \tag{3.11}$$

i_F gösterge fonksiyonlarını belirten bir ifadedir. İki sınıflı sınıflandırma işlemlerinde $i_F(x,w)$ gösterge fonksiyonlarının bir kümesinin VC boyutu, bütün olağan durumlarda ayrılabilecek noktaların en büyük h sayısını gösterir. İki sınıflı örüntü tanıma problemlerinde, *n* tane nokta kümesi 2^n mümkün yolla etiketlenebilir. Eğer kümenin üyeleri, bütün etiketleri doğru olarak işaret ediyorsa, bu fonksiyonlar kümesinin VC boyutu $h = n$ 'dir.

İki boyutlu sınıflandırma problemlerinde, $R^2(x_1, x_2)$ özellik uzayında, hiperdüzlemin bir tarafındaki bütün noktalar +1, diğer taraftaki noktalar da -1 değerini alır.



Şekil 3.7. Üç noktanın değişik şekillerde hiperdüzlemle ayrılması

Şekil 3.7'de üç nokta için mümkün olan bütün durumlar gösterilmiştir. Üç noktanın bir hiperdüzlem tarafından nasıl ayrıldığı $2^3 = 8$ şeklinde hesaplanır.

Bir n-boyutlu giriş uzayında, yönlü hiperdüzlem gösterge fonksiyonunun VC boyutu

$$h = n + 1 \tag{3.12}$$

olarak hesaplanır.

VC boyutu istatistiksel öğrenme teorisindeki bütün sonuçlarda kullanılır. Kolay bir iş değildir. Bu nicelik, spesifik tahmin fonksiyonuna ve çözülecek öğrenme probleminin tipine bağlıdır (sınıflandırma veya regresyon problemlerinden biri).

3.1.8. Optimizasyon Teorisi

Optimizasyon teorisi matematiğin, problemlerin sınıflarının çözümlerini karakterize etme üzerine yoğunlaşan ve onlar için verimli algoritmalar geliştiren dalıdır. Dolayısıyla makine öğrenmesi problemi optimizasyon teorisinin çerçevesi içinde analiz edilecek hale çevrilmiştir.

Optimizasyon teorisi bize sadece algoritmik teknikler sağlamakla kalmaz, çözüm olarak verilen bir fonksiyon için gerekli ve yeterli şartları da açıklar.

3.1.9. Langrangian Teorisi:

Langrangian teorisinin amacı başlangıçta hiçbir eşitsizlik sabiti olmayan bir optimizasyon probleminin çözümünü karakterize etmektir. Bu teorinin ana konsepti Langrange çarpanları ve Langrangian fonksiyonudur. Bu metot 1797'de mekaniksel problemler için Langrange tarafından geliştirilmiştir.

$f(w)$ amaç fonksiyonlu ve $h_i(w) = 0, i = 1,...,m$ eşitlik sabitli verilen bir optimizasyon problemi için

$$L(w, \beta) = f(w) + \sum_{i=1}^{m} \beta_i h_i(w) \tag{3.13}$$

Langrangian fonksiyonudur ve β_i katsayıları Langrange çarpanları olarak adlandırılır.

3.2. Destek Vektör Makineleri

DVM'ler 1960'ların sonlarında V.Vapnik tarafından bulunan istatistiksel bir metottur. DVM'ler yapısal risk minimizasyonu prensibi etrafında formülleştirilmiştir. Beklenen riskin üst sınırını küçüklemeye çalışır.

DVM uygulamaları diğer geleneksel metotlardan daha iyi sonuçlar vermektedir. DVM'ler el yazısı tanıma , ses tanıma ve uzaysal veri analizi gibi alanlarda ve çoğu sınıflandırma probleminde kullanılmıştır.

Destek Vektörü öğrenme iki bakımdan çok kullanışlıdır. Birincisi Destek Vektörü (DV) öğrenme basit fikirler üzerine kurulmuştur. İkincisi yüksek performanslı pratik uygulamalarda kullanılabilir.

DV algoritması öğrenme teorisinin ve pratiğinin kesiştiği bir uygulamadır; bazı belirli algoritma tipleri için istatistiksel öğrenme teorisinde hesaplamaların başarılı olması için faktörler tam olarak istenir. Gerçek-dünya uygulamaları, sık sık teorik olarak çözülmesi zor ve kompleks olan uygulamalardır. DV algoritması iki zorluğu da basitçe kaldırabilir. Yeterince kompleks olan modellere çözüm getirebilir.

Nöral ağların, radyal tabanlı fonksiyon ağlarının ve polinomal sınıflandırıcıların geniş bir kısmını içerir. Aynı zamanda matematiksel olarak analizi basittir, çünkü nonlineer giriş uzayındaki verileri dönüşümler yaparak özellik uzayına düşürür ve çözüm kolaylaşır.

DVM istatistiksel öğrenme teorisinde iyi şekilde kurulmuş bir teoriye sahiptir ve sınıflandırma ile regresyon problemlerine çözüm için uygundur. Vapnik'in teorisi eğitim kümesindeki arıza ile VC-boyutuna göre ifade edilen hipotez uzayının karmaşıklığının her ikisini de küçükleyen çözümün bulunduğunu göstermektir. Bu bakımdan, DVM ile bulunan

fonksiyon, veriye yakınlık ve çözümün karmaşıklığı arasındaki geçiştir. Özellikle iki sınıflı sınıflandırma probleminde DVM iki sınıf arasındaki sınırı büyükleyen optimal ayırt etme yüzeyini belirlemekte, yani eğitim kümesi ile ayırt etme yüzeyine en yakın noktaların arasındaki mesafeyi en büyüklemektedir. Ayrıca, optimal ayırt etme yüzeyi, destek vektörleri olarak adlandırılan veri noktalarının küçük bir kümesinde merkezlenmiş Kernel fonksiyonların doğrusal bir kombinasyonu olarak ifade edilmektedir. Genelde destek vektörleri çok az miktarda olduğu için optimal ayırt etme yüzeyinin gösterimi seyrektir, bir anlamda veri noktalarının sadece bir bölümü sınıflandırma görevi ile ilgilidir. Buna ilaveten, doğrusal kombinasyon katsayıları doğrusal eşitsizlik ve eşitlik kısıtları dahilinde bir konveks ikinci dereceli (quadratic) programlama probleminin çözümü ile belirlenmektedir. Çok katmanlı perseptron ve radyal tabanlı ağlar, uygun çekirdek fonksiyonları ile elde edilen DVM ağların özel durumları olarak gösterilebilir.

DVM eğitimindeki dizayn adımlarını şu şekilde verebiliriz [4]:

Adım 1:Sınıflandırma probleminde karar fonksiyonunun, regresyon probleminde regresyon fonksiyonunun biçimini belirleyen Kernel fonksiyonunun belirlenmesi

Adım 2: Seçilen Kernel fonksiyonunun parametre seçimi

Adım 3: C yaptırım faktörünün seçimi

Adım 4: İkinci dereceden problemin çözümü

3.2.1. En Büyük Sınırlı Sınıflandırma ve Lineer Destek Vektör Makineleri

DVM'lerin en basit modeli en büyük sınırlı sınıflandırma olarak bilinmektedir. DVM'lerin bu kısmı sadece lineer olarak ayrılabilen özellik uzayı için geçerlidir. Bu nedenle, çoğu gerçek-dünya uygulamalarında kullanılması oldukça zor bir yöntemdir. Ama anlaşılması en kolay algoritmadır ve daha karmaşık DVM için temel kısmı inşa etmektedir. DVM için geçerli olan çoğu anahtar özellikleri içinde bulundurur.

Giriş verilerinin Kernel fonksiyonlarından geçirilip özellik uzayına düşürülmesine gerek yoktur. Sadece verileri birbirinden en iyi şekilde ayıran ve en büyük sınırlı sınıflandırmayı yapan hiperdüzlemi bulmak yeterli olacaktır.

Hiperdüzlem tarafından ayrılan veriler x^+ ve x^- ise,

$$\left\langle w.x^+ \right\rangle + b = +1 \tag{4.1}$$

$$\left\langle w.x^- \right\rangle + b = -1 \text{'dir.} \tag{4.2}$$

Sınıflandırıcının sınırı γ ile gösterilecek olursa;

$$\gamma = \frac{1}{2}\left(\left\langle \frac{w}{\|w\|_2} \cdot x^{+} \right\rangle - \left\langle \frac{w}{\|w\|_2} \cdot x^{-} \right\rangle \right)$$

(4.3)

$$= \frac{1}{2\|w\|_2}\left(\langle w.x^{+} \rangle - \langle w.x^{-} \rangle \right)$$

$$= \frac{1}{\|w\|_2} \qquad \text{ifadesi geometrik sınırı verecektir.} \tag{4.4}$$

$S = ((x_1, y_1),...,(x_n, y_n))$ ile verilmiş bir lineer olarak ayrılabilen eğitim kümesinde optimizasyon problemini çözen (w,b) hiperdüzlemi en büyük değerli sınırlı hiperdüzlemin geometrik sınırı $\gamma = 1/\|w\|_2$ olarak hesaplanır.

Yukarıdaki denklemlerle verilen ifadelerin çözümü zor olacağından bu problemi uygun bir dual forma çevirmek iyi olacaktır.

$$L(w,b,\alpha) = \frac{1}{2}\langle w.w \rangle - \sum_{i=1}^{n} \alpha_i \left[y_i (\langle w.x_i \rangle + b) - 1 \right] \tag{4.5}$$

burada $\alpha_i \geq 0$ değerleri Langrange çarpanlarıdır.

(4.5) ifadesinde w ve b'ye göre türev alınacak olursa,

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{n} y_i \alpha_i x_i = 0, \tag{4.6}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0\,, \tag{4.7}$$

bu iki ifadeyi birleştirirsek,

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i, \tag{4.8}$$

$$0 = \sum_{i=1}^{n} y_i \alpha_i, \quad \text{ifadeleri bulunur.} \tag{4.9}$$

Hesaplanan bütün bu değerler birleştirilirse,

$$L(w,b,\alpha) = \frac{1}{2}\langle w.w \rangle - \sum_{i=1}^{n} \alpha_i \left[y_i (\langle w.x_i \rangle + b) - 1 \right] \tag{4.10}$$

$$= \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i . x_j \rangle - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i x_j \rangle + \sum_{i=1}^{n} \alpha_i$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i x_j \rangle.$$

(4.11)

ifadesi elde edilir.

$S = ((x_1, y_1),...,(x_n, y_n))$ ile verilen lineer olarak ayrılabilen bir eğitim kümesinde α^* parametrelerinin ikinci dereceden bir optimizasyon problemini çözdüğünü düşünürsek,

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i . x_j \rangle, \tag{4.12}$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0, \qquad \alpha_i \geq 0, i = 1,...,n \tag{4.13}$$

Ağırlık vektörü $w^* = \sum_{i=1}^{n} y_i \alpha_i^* x_i$, en büyük sınırlı hiperdüzlemin geometrik sınırı

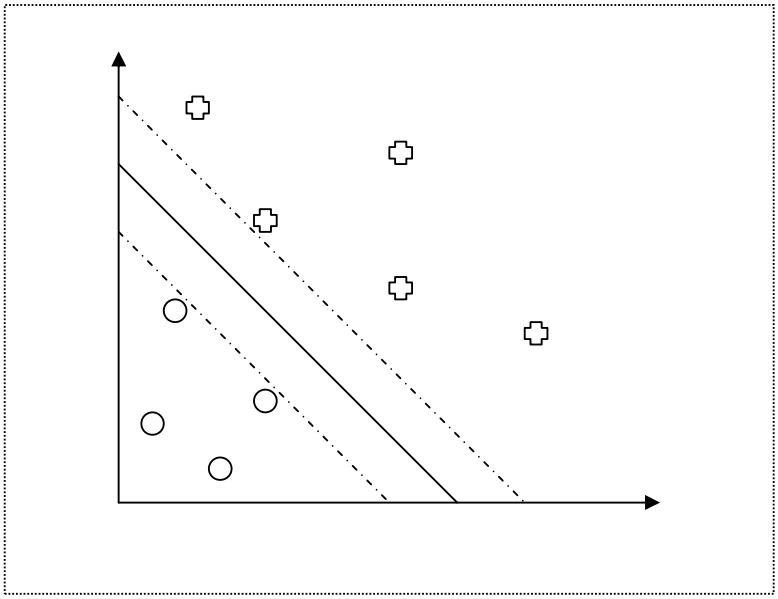
$\gamma = 1 / \|w^*\|_2$ olarak hesaplanır. (4.14)

b değeri dual problemde bulunmamaktadır. Bu nedenle birincil sabitler kullanılarak b^* değeri bulunmalıdır.

$$b^* = -\frac{\max_{y_i=-1}(\langle w^* . x_i \rangle) + \min_{y_i=1}(\langle w^* . x_i \rangle)}{2} \tag{4.15}$$

Karush-Kuhn-Tucker şartları kullanılarak,

$\alpha_i^* \left[y_i (\langle w^* . x_i \rangle + b^*) - 1 \right] = 0, \ i = 1,...,n$ ifadesi bulunur. (4.16)



Şekil 3.8. 3 tane destek vektörlü maksimum sınırlı hiperdüzlem

Hiperdüzleme en yakın noktalar yani Destek Vektörleri için α_i^* değerleri 0 değildir. Diğer noktalar için bütün α_i^* değerleri 0'a eşittir. Bu nedenle ağırlık vektörü w için sadece Destek Vektörlerinin (DV) α_i^* değerleri geçerlidir.

Bu açıklamalardan sonra optimal hiperdüzlem şu şekilde verilir.

$$f(x,\alpha^*,b^*) = \sum_{i=1}^{n} y_i \alpha_i^* \langle x_i . x \rangle + b^* \tag{4.17}$$

$$= \sum_{i \in DV} y_i \alpha_i^* \langle x_i . x \rangle + b^* \tag{4.18}$$

$j \in DV$ için,

$$y_j f(x_j,\alpha^*,b^*) = y_j (\sum_{i \in DV} y_i \alpha_i^* \langle x_i . x_j \rangle + b^*) = 1 \tag{4.19}$$

$$\langle w^* . w^* \rangle = \sum_{i,j=1}^{n} y_i y_j \alpha_i^* \alpha_j^* \langle x_i . x_j \rangle \tag{4.20}$$

$$= \sum_{j \in DV} \alpha_j^* y_j \sum_{i \in DV} y_i \alpha_i^* \langle x_i . x_j \rangle$$

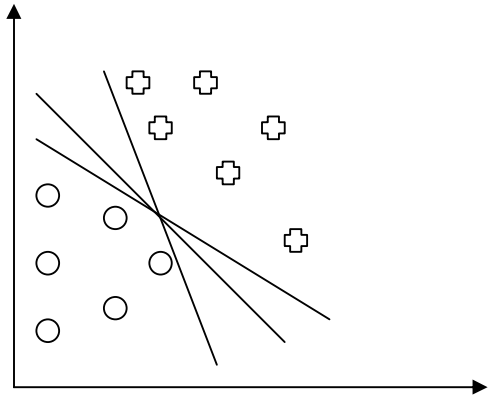
$$= \sum_{j \in DV} \alpha_j^* (1 - y_j b^*)$$

$$= \sum_{i \in DV} \alpha_i^* \tag{4.21}$$

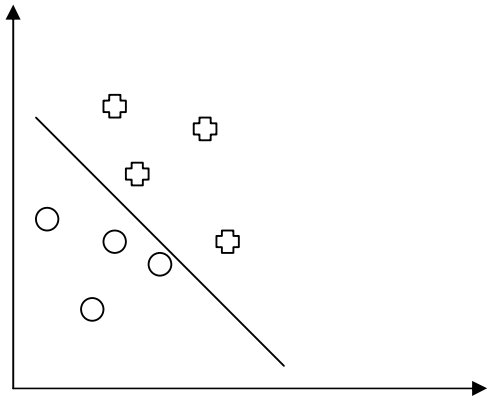
3.2.2. Sınır (Margin)

Regresyon için nöral ağların içeriğindeki iyi genellemeyi elde etmek için ağırlık yenilemesi ne kadar önemliyse, sınıflandırma problemlerinde sınır aynı rolü oynar. Sınır konsepti DVM'lerin formülasyonunu anlamak için en önemli adımdır.

Şekil 3.9 ve Şekil 3.10 iki boyutlu bir giriş uzayında verilmiş ayrılabilir probleme bir örnektir. Şekil 3.9'da iki sınıf verisini ayıran birkaç tane hiperdüzlem görülür.



Şekil 3.9. Hiperdüzlemin tek olmadığı durum



Şekil 3.10. Hiperdüzlemin tek olduğu durum

3.2.3. Lineer Ayırt Edilebilir Sınıflar

N boyutlu bir $S = \{(x_i + y_i)\}_{i=1}^{N}$ eğitim kümesi verilmiş olsun, burada i=1,2, ... , N için $x_i \in R^n$ ve $y_i \in \{-1,+1\}$. Diğer deyişle, S'teki örneklerin iki sınıftan birine ait olduğu varsayılmıştır. Bununla beraber sınıfların hiperdüzlemin aynı tarafındaki bir sınıfın bütün elemanlarının yer aldığı bir hiperdüzlem ile ayırt edilebildiği kabul edilmiştir. Amaç iki sınıfı ayıran hiperdüzlemin eşitliğini bulmaktır. Sınıflandırma için DVM yapısında çözümü zorlayan kısıt, optimal ayırt etme yüzeyinin eğitim kümesine en yakın noktalar ile birlikte mesafeyi büyüklemek zorunda olmasıdır. Bu kavramları daha anlaşılır hale getirmek amacıyla bir takım tanımlara ihtiyaç vardır.

Tanım: Eğer $w \in R^n$ ve $w \in R$ olursa, aşağıdaki durumları sağlayacak şekilde S kümesi doğrusal olarak ayrılabilir:

$$i = 1,2,...,N \text{ için } y_i(w.x_i + b) \geq 1.$$

(4.22)

(w,b) çiftinin w.x+b = 0 eşitliğine ait ayırt etme hiper uzayını tanımlamaktadır. d_i , ayırt etme hiper uzayından x_i noktasına kadar olan mesafeyi göstersin :

$$d_i = (w.x_i + b) \Big/ \|w\| \tag{4.23}$$

burada ||w|| sembolü, w'nin normunu göstermektedir. Böylece bütün $x_i \in$ S'ler için aşağıdaki eşitsizlik sağlanır:

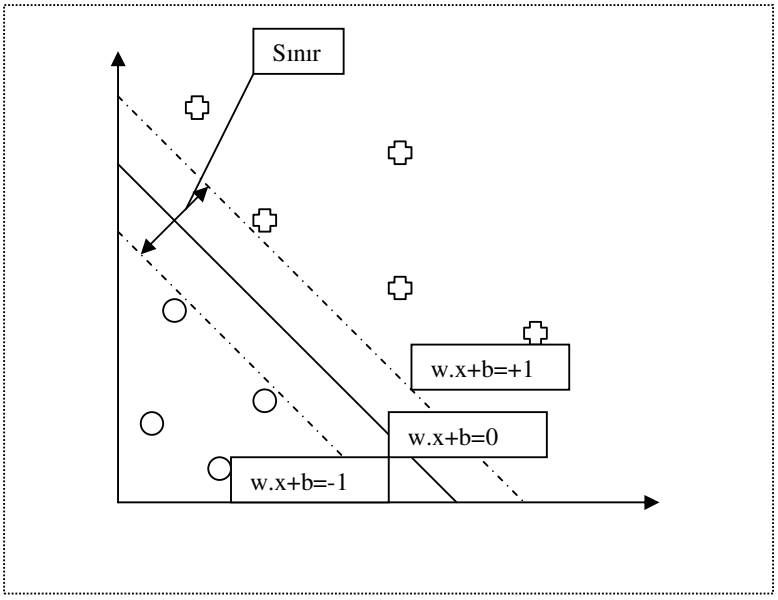
$$1 \Big/ \|w\| \leq y_i d_i \tag{4.24}$$

y_i d_i her zaman pozitif bir niceliktir. Buna ilaveten, 1/||w||, x_i noktaları ve ayırt etme hiperdüzlemi (w,b) arasındaki mesafenin alt sınırıdır; yani S noktaları ve en az 1/||w||'ye eşit olan ayırt etme hiperdüzlemi arasındaki mesafedir.

Tanım: Doğrusal bir şekilde ayrılabilen S kümesi için optimal ayırt etme hiperdüzlemi, S'e ait en yakın noktanın mesafesini en büyükleyen hiperdüzlemdir.

Optimal ayırt etme hiperdüzlemi ve S'e ait en yakın nokta arasındaki mesafe 1/||w|| olduğu için optimal ayırt etme hiperdüzlemi, 1/||w|| niceliğini en büyükleyen S için ayırt etme hiperdüzlemi olarak yorumlanabilir.

1/||w||'nin en büyüklenmesi, ||w||'nin en küçüklenmesine eşit olduğu için bu problemin çözümü S'in optimal ayırt etme hiperdüzlemine eşittir. 2/||w|| niceliği sınıflar arasındaki sınır olarak adlandırılır ve optimal ayırt etme hiper yüzeyi sınırı en büyükleyen hiperdüzlem olarak yorumlanabilir. Sınır, S kümesinin ayırt edilebilirlik ölçütüdür veya diğer bir deyişle sınıflandırma probleminin karmaşıklığının bir ölçütüdür.



Şekil 3.11. Sınır değerleri

Lineer olarak ayrılabilen durumlarda, sınıfları birbirinden ayıracak birkaç hiperdüzlem bulunabilir. Önemli olan bu hiperdüzlemler içinde en iyisini seçebilmektir. Lineer olarak ayrılabilen verinin sınıflandırılması durumunda, bütün hiperdüzlemler içinde eğitim arızasını minimize eden, en büyük sınır değerine sahip olan hiperdüzlem seçilir. Büyük değerli sınıra sahip olmayan hiperdüzlemin beklenen riski yüksek olacaktır.

Lineer olarak ayrılabilen durumlarda herhangi bir nokta ile hiperdüzlem arasındaki mesafeyle ilgili şöyle bir örnek verilebilir.

$w_1x_1 + w_2x_2 + ... + w_nx_n \pm b = 0$ ile tanımlanan bir

$d(x, w, b) = 0$ hiperdüzlemi ve

$P(x_{1p}, x_{2p}, ..., x_{np})$ noktası olsun.

P noktasından d hiperdüzlemine olan D uzaklığı;

$$D = \frac{|(w.x_p) \pm b|}{\|w\|} = \frac{|w_1x_{1p} + w_2x_{2p} + ... + w_nx_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + ... + w_n^2}}$$ olarak hesaplanır. (4.25)

(4.18) eşitliğine göre (1,1,1,1) noktası ve $x_1 + 2x_2 + 3x_3 + 4x_4 - 2 = 0$ hiperdüzlemi arasındaki mesafe;

$b = -2,$

$w_1 = 1, w_2 = 2, w_3 = 3, w_4 = 4$ olarak verilmiştir. Buna göre,

$D = \frac{|[1,2,3,,4]]1,1,1,1] - 2|}{\sqrt{30}} = \frac{8}{\sqrt{30}}$ olarak bulunur.

3.2.4. Lineer Olarak Ayırt Edilemeyen Sınıflar

Gerçek hayatta karşılaşılan çoğu problemler ya lineer ya da nonlineer durumlarda ayırt edilemezler. Genellikle mümkün olduğu kadar sınıfları ayırabilen giriş kümeleri bulmaya çalışılır, fakat çoğunlukla ilgili girişler kayıptır, veri tam değildir, güvenilir değildir veya gürültülüdür.

Lineer DVM'lerin ayırt edilemez durumuna genişletilmesi Cortes ve Vapnik tarafından yapılmıştır. Temel olarak, problem formülasyonlarında ek serbestlik değişkenleri alarak yapılmıştır.

Doğrusal olarak ayırt edilebilen sınıflar hipotezi pratik uygulamalar için oldukça sınırlıdır. Bu nedenle önceki sonuçları düzenlemeye gerek vardır. Bu durum, $y_i(w.x_i + b) + \xi_i < 1$ ifadesini sağlayacak şekilde bir takım (x_i , y_i) ∈ S noktalarının var olduğu anlama gelir. Negatif olmayan bir ξ_i serbestlik değişkeninin girilmesi $y_i(w.x_i + b) \geq 1$ ile aynı formda doğru şekilde sınıflandırılmamış bir nokta için kısıt belirtmeye ve doğrusal olarak ayırt edilemeyen eğitim kümesi durumunda önceki analizleri genişletmeye izin verir. Bu amaçla, aşağıdaki ifadeyi sağlayacak şekilde N adet negatif olmayan serbestlik değişkeni (S'teki her nokta için) $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$ verilirse;

$$y_i(w.x_i + b) + \xi_i \geq 1, \quad i=1,2,\ldots,N \tag{4.26}$$

x_i noktası doğru bir şekilde sınıflandırılmış ise ξ_i =0 eşitliğine göre $y_i(w.x_i + b) + \xi_i \geq 1$'dir ve $y_i(w.x_i + b) + \xi_i \geq 1$ kısıtı $y_i(w.x_i + b) \geq 1$'e dönüşür. x_i noktası doğru bir şekilde sınıflandırılmamış ise $y_i(w.x_i + b) + \xi_i \geq 1$'in sağlanması için serbestlik değişkenine uygun şekilde bir $\xi_i > 0$ değeri varsayılacaktır.

Lineer metottan, nonlineer tekniğe geçişi yapabilmek için Vapnik giriş verisini çok boyutlu özellik uzayına haritalamayı gerçekleştirmiştir. Giriş değerleri özellik uzayına düşürülerek lineer ayırıcı hiperdüzlem yapılmıştır.

$K(x,z) = \Phi(x).\Phi(z)$ uygulaması sıklıkla Kernel oyunu diye adlandırılır. Bu bize kesin hesaplamalar yapmadan çok boyutlu özellik uzayında çalışma kolaylığı sunar. Bu Kernel oyununu uyguladıktan sonra hesaplamalar başka uzayda gerçekleştirilir. DVM'lerde $\Phi(.)$ dönüşümleri yapılarak çok boyutlu özellik uzayında formülasyonlar yapılır.

3.2.5. Nonlineer Destek Vektör Makineleri Sınıflandırıcıları

Lineer Destek Vektör Makineleri Sınıflandırıcılarından nonlineer DVM sınıflandırıcılara genişletme basittir. x ile Φ(x) biçimsel olarak yer değiştirebilir ve Kernel oyunu uygulanabilir.

Nonlineer DVM'leri dizayn etmek için temel fikir, giriş uzayındaki verileri yüksek boyutlu özellik uzayına düşürmektir.

$$x \in R^n \rightarrow z(x) = [a_1\phi_1(x), a_2\phi_2(x), ..., a_n\phi_n(x)]^T \in R^f$$

Yukarıdaki gösterimde $R^n \rightarrow R^f$ 'e bir dönüşüm gösterilmiştir. Bir $\phi(x)$ eşleştirmesi önceden seçilen, sabit bir fonksiyondur. Bir x-giriş uzayı, x giriş vektörünün x_i bileşenlerinden oluşur ve bir f-özellik uzayı (z-uzayı) z vektörünün $\phi_i(x)$ bileşenlerinden oluşur. Bu dönüşüm yapıldıktan sonra z-uzayında lineer algoritma uygulanarak çözüme ulaşılması beklenilir.

Bir x-giriş uzayından, yüksek dereceli z-uzayına düşürme işleminde iki problemle karşılaşabiliriz: ϕi(x) eşleştirmesinin seçimi ve eğer f özelliklerinin sayısı çok fazla ise hesaplanması gözü korkutan bir $z^T(x).z(x)$ skaler çarpımının hesaplama işlemidir. Bu işlemlerden de kurtulabilmenin yolu Kernel fonksiyonlarının kullanımıdır.

Bir özellik uzayındaki $\phi^T(x_i).\phi(x_j)$ skaler çarpımlarını, herhangi bir Kernel fonksiyonu seçildikten sonra $K(x_i, x_j)$ hesaplamasını yaparak bulabiliriz. Bu çok uğraştırıcı bir işlem değildir.

En çok kullanılan kernel fonksiyonları şunlardır:

$$K(x, y) = (x.y)^d \text{ (polinomal kernel)} \tag{4.27}$$

$$K(x, y) = \tanh(\kappa(x.y) + \Phi) \text{ (sigmoidal kernel)} \tag{4.28}$$

$$K(x, y) = \exp(-\|x - y\|^2 / (2\sigma^2)) \text{ (RBF Kernel)} \tag{4.29}$$

Nonlineer durumlarda da Langrange çarpanlarını kullanıp gerekli işlemler yapılır ve özellik uzayına düşürülen veriler sınıflandırılır.

$$L_d(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j z_i^T z_j \tag{4.30}$$

$$L_d(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{4.31}$$

Burada $\alpha_i \geq 0, i = 1,...,n$ ve $\sum_{i=1}^{n} \alpha_i y_i = 0$'dır.

DVM ve sınıflandırma problemi için anlaması en kolay problem XOR (Exclusive OR) problemi verilebilir. Bu problemin çözümü ve işlem sırası kısaca şu şekilde verilebilir:

Bir x noktası ile (w,b) düzlemi arasındaki mesafe;

$$\frac{|wx+b|}{\|w\|}\text{'dir.} \tag{4.32}$$

Farklı sınıflardaki iki destek vektörü arasındaki mesafe sınır olarak adlandırılır ve

$$m = \frac{2}{\|w\|} \text{ olarak hesaplanmıştır.} \tag{4.33}$$

Sınırın büyük olması sınıflandırmanın iyi olduğunu göstermektedir. Bu yüzden sınırı maximum etmek

$J(w)=\frac{1}{2}\|w\|^2$'yi minimize etmeye eşittir. $(y_i\ (w^T\ x_i + b) \geq 1)$

J(w) ifadesini minimize etmek için

$L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i (w^T x_i + b) - 1]$ ifadesini çözmek gerekir.

(w,b) parametrelerini elimine etmek için $L_p(w,b,\alpha)$ ifadesinin w'ye ve b'ye göre türevi alınıp 0'a eşitlenir.

$$\frac{\partial L_p(w,b,\alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{4.34}$$

$$\frac{\partial L_p(w,b,\alpha)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.35}$$

Bu değerlere bağlı olarak ve $\partial J / \partial w = 0$ optimallik şartını kullanarak

$$L_p(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b\sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \quad (L_p\text{'yi açarsak}) \tag{4.36}$$

$$w^T w = w^T . \sum_{i=1}^{N} \alpha_i y_i x_i = \sum_{i=1}^{N} \alpha_i y_i w^T x_i = \sum_{i=1}^{N} \alpha_i y_i [\sum_{i=1}^{N} \alpha_j y_j x_j]^T x_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{4.37}$$

L_p'nin ikinci terimi de aynı şekilde ifade edilebilir.

L_p'nin üçüncü terimi $\partial J / \partial b$'ye bağlı olarak 0 olur.

Bunları birleştirirsek;

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

(4.38)

$\alpha_i \geq 0$ ve $\sum_{i=1}^{N} \alpha_i y_i = 0$ değerlerine bağlı olarak.

Dikkat edilirse denklemde artık w ve b terimleri yoktur. KKT şartlarına göre eğitim setindeki her örnek;

$\alpha_i [y_i (w^T x_i + b) - 1] = 0$ denklemini sağlamak zorundadır.

Yani $\alpha_i = 0$ veya $[y_i (w^T x_i + b) - 1] = 0$ olmalıdır. (Destek vektörleri için $\alpha_i > 0$ 'dır.)

Lineer olarak ayrılamayan durumlarda

$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_i)$ denklemi uygulanır.

$\Phi(x_i)^T \Phi(x_i) = k_{ij}$ olarak gösterilir.

XOR probleminin DVM yöntemiyle çözümü:

Sınıf 1: x_1=(-1,-1), x_4=(1,1)

Sınıf 2: x_2=(-1,1), x_3=(1,-1)

Kernel fonksiyonu olarak ikinci dereceden polinomal kernel seçildi.

$K(x, x') = (x^T x' + 1)^2$

Bu fonksiyona uygun eşleştirme ise;

$\Phi(x_i) = (1, x_{i1}^2, \sqrt{2} x_{i1} x_{i2}, x_{i2}^2, \sqrt{2} x_{i1}, \sqrt{2} x_{i2})^T$

Bütün x ve y elemanları Kernel fonksiyonunda yerine konur ve Kernel matrisi elde edilir.

K= $\begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$ elde edilir.

$$L_D(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j k_{ij}$$

$$L_D(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2} [9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 +$$
$$9\alpha_2^2 + 2\alpha_2\alpha_3 - \alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2]$$

α_i 'lere göre türev alınırsa;

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$
$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$
$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$
$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$

Buradan $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$ olur. $\alpha_i > 0$ olduğu için bütün noktalar destek vektörleridir.

$$w = \sum_{i=1}^{N} \alpha_i y_i \Phi(x_i)$$

$$= \alpha_1 y_1 \Phi(x_1) + \alpha_2 y_2 \Phi(x_2) + \alpha_3 y_3 \Phi(x_3) + \alpha_4 y_4 \Phi(x_4)$$
$$= -0.125\Phi(x_1) + 0.125\Phi(x_2) + 0.125\Phi(x_3) - 0.125\Phi(x_4)$$
$$= \frac{1}{2}[-\Phi(x_1) + \Phi(x_2) + \Phi(x_3) - \Phi(x_4)]$$

$$w = -\frac{1}{8}\begin{bmatrix}1\\1\\\sqrt{2}\\1\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} + \frac{1}{8}\begin{bmatrix}1\\1\\-\sqrt{2}\\1\\-\sqrt{2}\\\sqrt{2}\end{bmatrix} + \frac{1}{8}\begin{bmatrix}1\\1\\-\sqrt{2}\\1\\\sqrt{2}\\-\sqrt{2}\end{bmatrix} - \frac{1}{8}\begin{bmatrix}1\\1\\\sqrt{2}\\1\\\sqrt{2}\\\sqrt{2}\end{bmatrix}$$

$$= \begin{bmatrix}0\\0\\-1/\sqrt{2}\\0\\0\\0\end{bmatrix}$$

w'nin ilk elemanı b'yi verir. (b=0)

Optimal hiperdüzlem

$w^T \Phi(x) + b = 0 \Rightarrow w^T \Phi(x) = 0$ olarak verilmişti.

O halde

$$f(x) = (0\ 0\ -1/\sqrt{2}\ 0\ 0\ 0) \begin{bmatrix} 1 \\ x_1^{\,2} \\ \sqrt{2}x_1x_2 \\ x_2^{\,2} \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = -x_1x_2$$

Bu problem için ayırıcı hiperdüzlemin denklemi

$f(x) = -x_1x_2$ olarak bulunur.

3.2.6. Çok-sınıflı sınıflandırma

DVM'ler aslında ikili sınıflandırıcılardır. Sadece iki sınıfı birbirinden ayırmak için dizayn edilmişlerdir. Bununla birlikte, çoğu gerçek uygulamalarda, çok-sınıflı sınıflandırma gerekmektedir. Örneğin, bir asenkron motorun arıza sınıflandırmasında, sağlıklı operasyona ek olarak birkaç arıza sınıfı vardır.

Bir çözüm çok-sınıflı bir problemi birkaç 2-sınıflı probleme ayrıştırmak, sınıflandırıcıları bu problemleri çözmek için eğitmek ve çıkış için bunları tekrar yapılandırmaktır. En kolay çok-sınıflı sınıflandırma yapılarından biri birine-karşı-diğerleri yaklaşımıdır. Bu metotta, K sınıflandırıcı, her sınıflandırıcı diğerlerinden bir sınıf ayıracak şekilde oluşturulmuştur. Bununla birlikte, çoğu uygulamalarda, bu yaklaşım diğerlerine göre alt seviyede kalmaktadır.

İkişerli sınıflandırıcıların çözümlerinin çıkışlarından son sınıflandırma çözümünü yapılandırmak için çeşitli planlar vardır. En basit metotlar çoğunluk oylamasına bakar. İkişerli sınıflandırıcılar sınıflar için oylar verir ve en çok oyu alan sınıf dikkate alınan örnek için son karar sınıfı olarak seçilir. Eğer tekrar oluşturma kullanılmışsa, sınıflandırıcılar sadece ikili oylar (-1 ya da 1) kullanabilir, fakat yumuşak oluşturmada sınıflandırıcıların kesin çıkışları oy olarak göz önüne alınmıştır. DVM tabanlı bir sınıflandırıcının kesin çıkışı ne kadar yüksekse, sınıflandırılacak örnek pozitif sınıfa aittir ve çıkış ne kadar küçükse örnek negatif sınıfa aittir. Eğer çıkış sıfıra yakınsa sınıflandırma kararı güvenilmezdir.

Çoğunluk oylaması uygularken önemli bir problem meydana gelir. Verilen bir x örneği için, oylama planı bütün çift sınıflandırıcıların çıkışlarını değerleri dikkate alınmadan eşit olarak ağırlıklandırır. Tabi ki, sınıflandırıcının başarısıyla ilgilenen konu ile ilgili sınıflandırıcılar gelişmede bilinmemektedir. Bununla birlikte, bazı çift sınıflandırıcıların fazlalığı bir karışım

matrisiyle birlikte düşünülebilir. Bu yaklaşımla birlikte, sınıflandırıcıların çıkışları karışım matrisiyle lineer olarak birleştirilir.

Literatürde önerilen birleştirme planlarından bir diğeri de, örneğin, ikili ağaçlar [7] ve bulanık mantık tabanlı metotlardır [26]. İkili ağaçları uygularken, sınıflandırıcıların tam bir hiyerarşisi sınıflandırıcıların eğitiminden önce bilinmelidir. Bu sınıflandırma probleminin çözümünün ve karmaşık kümelemenin uygulaması ve vektör ölçüm algoritmalarının bilinmesini gerektirir. Bulanık mantık yaklaşımı kullanıldığı zaman, üyelik fonksiyonlarının seçimi ve ayarlaması uygulamaya bağlı bir iştir ve bazı zamanlar oldukça zaman yok edicidir.

3.2.7. Bire-karşı-biri Metodu

Bu metotta, K tane sınıf varsa eğer $k.(k-1)/2$ tane sınıflandırıcı yapılandırılır ve aşağıdaki ikili sınıflandırma problemi çözülür.

$$\min_{w^{ij},b^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_n^{ij}(w^{ij})T \tag{4.39}$$

$$(w^{ij})^T \phi(x_n) + b^{ij} \geq 1 - \xi_n^{ij}, \text{ eğer} \quad y_n = i \text{ ise} \tag{4.40}$$

$$(w^{ij})^T \phi(x_n) + b^{ij} \leq -1 - \xi_n^{ij}, \text{ eğer } y_n = j \text{ ise} \tag{4.41}$$

Bütün M tane sınıflandırıcının yapılandırılmasından sonra bir sonraki testi yapmak için farklı metotlar vardır.

3.2.8. Bire-karşı-diğerleri Metodu

Bu metotta da, K-sınıflı bir problem içinK tane ikili sınıflandırıcı oluşturulur. i. Sınıftaki veriyi pozitif etiketli, diğerlerini negatif etiketli olarak alır. i. DVM aşağıdaki problemi çözer.

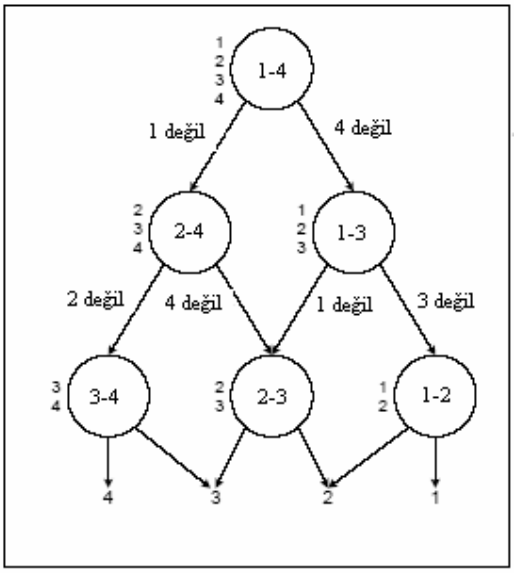
$$\min_{w^i,b^i,\xi^i} \frac{1}{2}(w^i)^T w^i + C\sum_{j=1}^{n} \xi_j^i \tag{4.42}$$

$$(w^i)^T \phi(x_j) + b_i \geq 1 - \xi_j^i, \text{ eğer } y_j = i \text{ ise} \tag{4.43}$$

$$(w^i)^T \phi(x_j) + b_i \leq -1 + \xi_j^i, \text{ eğer } y_j \neq i \text{ ise} \tag{4.44}$$

3.2.9. Yönlendirilmiş Çevrimsiz Çizge Metodu

Bu metotta, bir çizge mantığı kullanılır. Çizge $k.(k-1)/2$ tane iç düğüme ve k tane de yaprağa sahiptir.



Şekil 3.12. Yönlendirilmiş çevrimsiz çizge

Şekil 3.12'de yönlendirilmiş çevrimsiz çizge verilmiştir. İlk düğümde, 1. ve 4. sınıf karşılaştırılmış ve 1. sınıfa ait olmayan veriler sol kola, 4. sınıfa ait olmayan veriler ise sağ kola yönlenmiştir. 2-4 düğümünde ise 2. sınıf ve 4. sınıf karşılaştırılması yapılmıştır. Ve işlemler her düğümde böyle devam ederek sınıflandırma işlemi yapılmıştır.

4. UYGULAMA SONUÇLARI

Bu tezde, uygulamalar için kullanılmak üzere tek fazlı asenkron motor kullanılmıştır. Öncelikle sağlam motordan veriler alınmış ve daha sonra rotor çubukları delinerek kırık rotor çubuğu arızası elde edilmiştir. Elde edilen veriler, uygun özellik çıkarma algoritmalarından geçirildikten sonra arıza teşhisi için kullanılabilir hale getirilmiştir.

4.1. Uygulama Çalışması 1

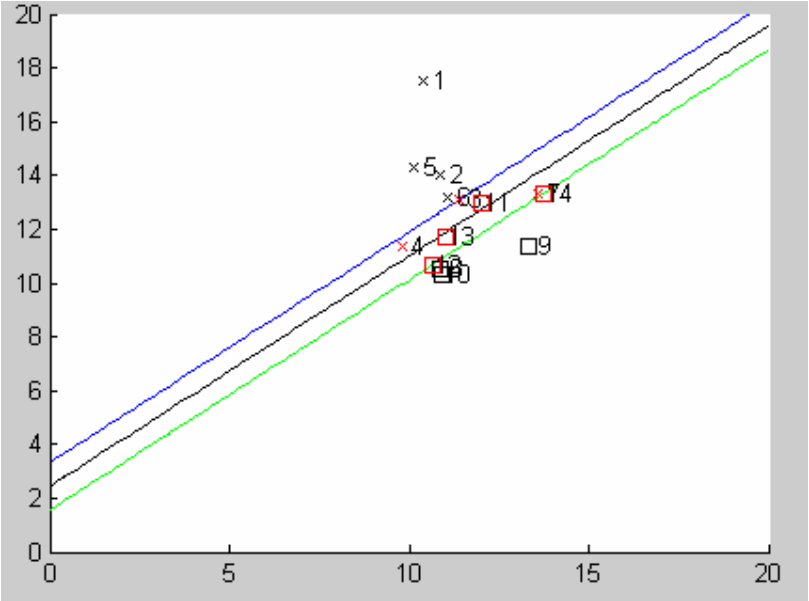
Bu uygulama çalışmasında, tek fazlı sincap kafesli asenkron motor kullanılmıştır. Bu çalışmada, sağlam motor verisi ile bir kırık rotorlu motordan alınan veriler sınıflandırılmıştır. Kullanılan motorun özellikleri Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Tek fazlı sincap kafes asenkron motor özellikleri

Özellik	Değer
Güç (Hp)	1/4
Gerilim (V)	110
Rotor Sayısı	36
Frekans (Hz)	60
X_{ls} (ohms)	2.79
X_M (ohms)	66.8
X_{lr} (ohms)	2.12
r_r (ohms)	4.12
J (kg m^2)	0.015

4.1.1. Simülasyon Sonuçları

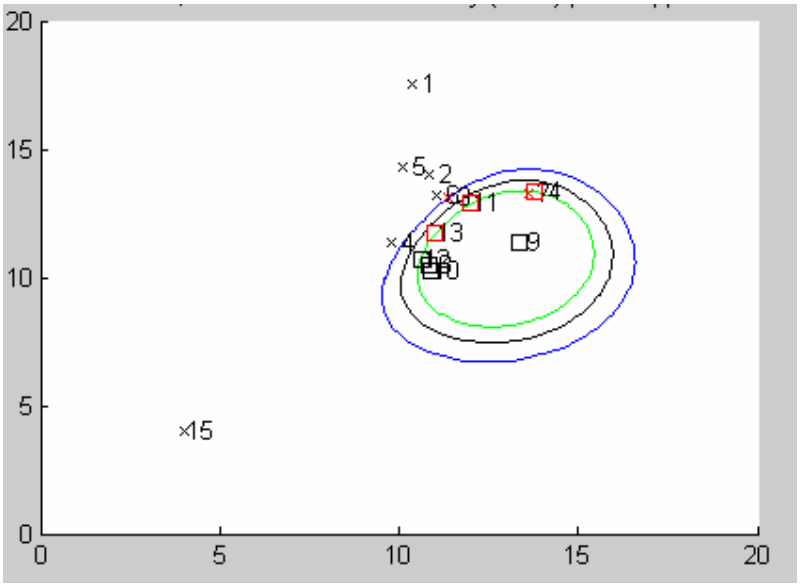
Bu uygulamada iki-sınıflı bir sınıflandırma işlemi uygulanmıştır. DVM için alınan kernel fonksiyonu lineer kerneldir. Lineer kernel çoğu karmaşık durumlarda iyi sonuç vermemektedir. DVM için önemli bir parametre de C yaptırım parametresidir. Bu uygulamada C değeri 10 olarak alınmıştır. İki sınıfa ait veriler alınıp, sınıflandırma işlemi uygulandıktan sonra elde edilen çıktı şu şekildedir:



Şekil 4.1. Sağlıklı ve arızalı durum sınıflandırması

Şekil 4.1'de her sınıftan 7'şer örnek alınıp sınıflandırma işlemi yapılmıştır. 7 tane DV bulunmaktadır. DV sayısının çokluğu hesapsal kolaylığı da ortadan kaldırır. Oysa ki DVM yönteminin önemli bir özelliği hesapsal kolaylık sağlamasıdır. DV'ler dışındaki giriş örneklerinin Lagrange çarpanları 0'a eşittir. DV'lerin Lagrange çarpanları 0'a eşit olmadığı için ne kadar çok DV olursa işleme alınacak örnek sayısı da artacak ve işlemleri zorlaştıracaktır. DV'lerin sayısının çok olması iyi bir sınıflandırma yapılmadığını göstermektedir. Bu durum seçilen Kernel fonksiyonuna ve C yaptırım parametresine bağlıdır.

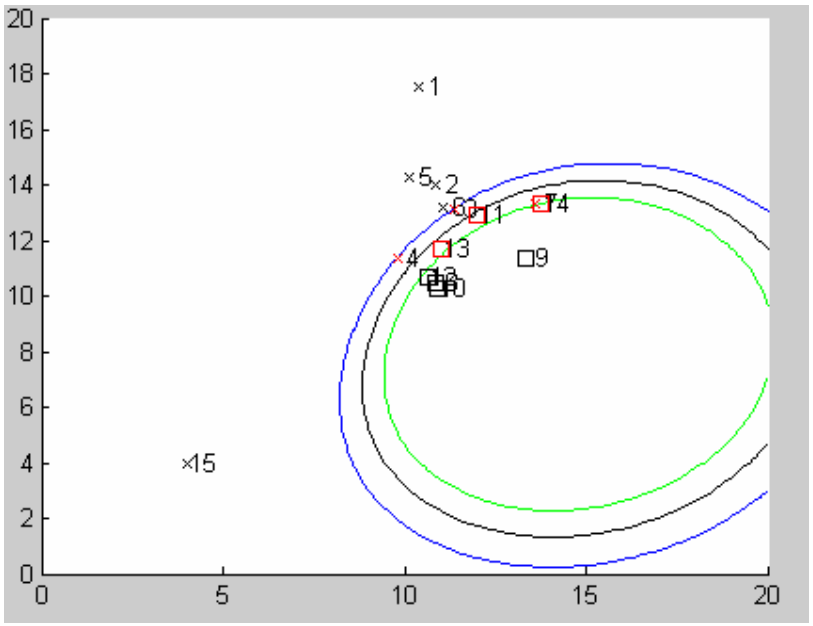
İkinci çalışma olarak Kernel fonksiyonu değiştirilmiş ve C yaptırım parametresinin değerlerine göre sistem incelenmiştir.



Şekil 4.2. Genişliği 8 olan RBF kernel uygulaması

Şekil 4.2'de genişliği 8 olan RBF kernel fonksiyonu kullanılmıştır. DVM'lerde sıkça kullanılan bir kernel fonksiyonu ise RBF kernelidir. Bu fonksiyon içinde de belirlenmesi gereken bir parametre vardır. Bu parametre seçimi iyi yapılırsa daha iyi sonuçlar elde edilecektir.

Bu uygulamada, DV'lerin sayısının bir önceki lineer kernelli uygulamaya göre daha az olduğu gözlemlenmiştir. Bu da sınıflandırmanın daha iyi olduğunu göstermektedir.



Şekil 4.3. Genişliği 36 olan RBF kernelli uygulama

Şekil 4.3'te de RBF fonksiyonunun genişliği 36 olarak ayarlanmıştır. Bu örnekte de DV sayısının bir önceki çalışmadan daha fazla olduğu gözlemlenmiştir. Sağlam motor verileri ile kırık rotorlu motordan alınan verilerinin birbirine çok yakın olduğu görülmüştür. DV'ler arasındaki mesafe de azdır.

4.2. Uygulama Çalışması 2

Bu uygulama çalışmasında, yapılan birçok sınıflandırma işlemi incelenmiştir. Bu çalışmada da yine iki-sınıflı sınıflandırma işlemi yapılmıştır. Sağlıklı motor verileri ile bir kırık rotorlu motor verileri sınıflandırılmıştır. Kernel fonksiyonu olarak RBF seçilmiş ve bu fonksiyon için gerekli olan parametre seçimi yapılmıştır. Toplam 30 veri sınıflandırılmıştır. Bu veriler ait oldukları sınıflara atanmış ve C yaptırım parametresinin seçimine göre de DV sayıları karşılaştırılmıştır.

Çizelge 4.2. RBF kernel fonksiyonu ve parametrelerine göre DV sayıları

Kernel Parametresi	C Yaptırım değeri	DV Sayıları
1	100	14
10	100	8
16	100	11
70	100	19
70	500	12
70	Sonsuz	6
100	Sonsuz	3

Çizelge 4.2'de görüldüğü gibi en iyi sonuç genişliği 100 ve C parametresi sonsuz seçilince elde edilmiştir. Bu parametre seçimleri için herhangi bir zorunluluk yoktur. Bazı değerler seçilir ve sınıflandırma sonuçları karşılaştırılıp çalışılan problem için en uygun değerler belirlenir.

Bu çalışmada, DV sayılarına bakılırsa bazı seçimlerin çok iyi olmadığı görülür. Bu nedenle uygun parametre seçimi dikkatle yapılmalıdır.

4.3. Uygulama Çalışması 3

Bu çalışmada da, farklı bir kernel fonksiyonu olan polinomal kernel seçilip, gerekli parametre ayarları yapılmıştır. Giriş verileri alındıktan sonra sınıflandırma için farklı yapılar denenmiştir. RBF kernelinde C yaptırım parametresi sonsuz seçilince iyi sonuç vermiştir fakat burada ise bu parametre için daha küçük değerlerin iyi sonuç verdiği gözlemlenmiştir. Sonuçlar Çizelge 4.3'te verilmiştir.

Çizelge 4.3. Polinomal kernel fonksiyonu ve parametrelerine göre DV Sayısı

Kernel Parametresi	C Yaptırım Değeri	DV Sayısı
2	100	7
2	5	14
4	5	7

Kernel fonksiyonu parametresi ile C parametresinin farklı değerleri için sağlıklı durum ve arızalı durum sınıflandırılması yapıldığında bir uygulamada giriş verilerinin neredeyse yarısının DV olduğu görülmüştür. Çizelge 6.3'te de görüldüğü gibi en iyi sonucu derecesi 4 olan ve C değeri 5 olan polinomal kernel vermiştir.

4.4. Uygulama Çalışması 4

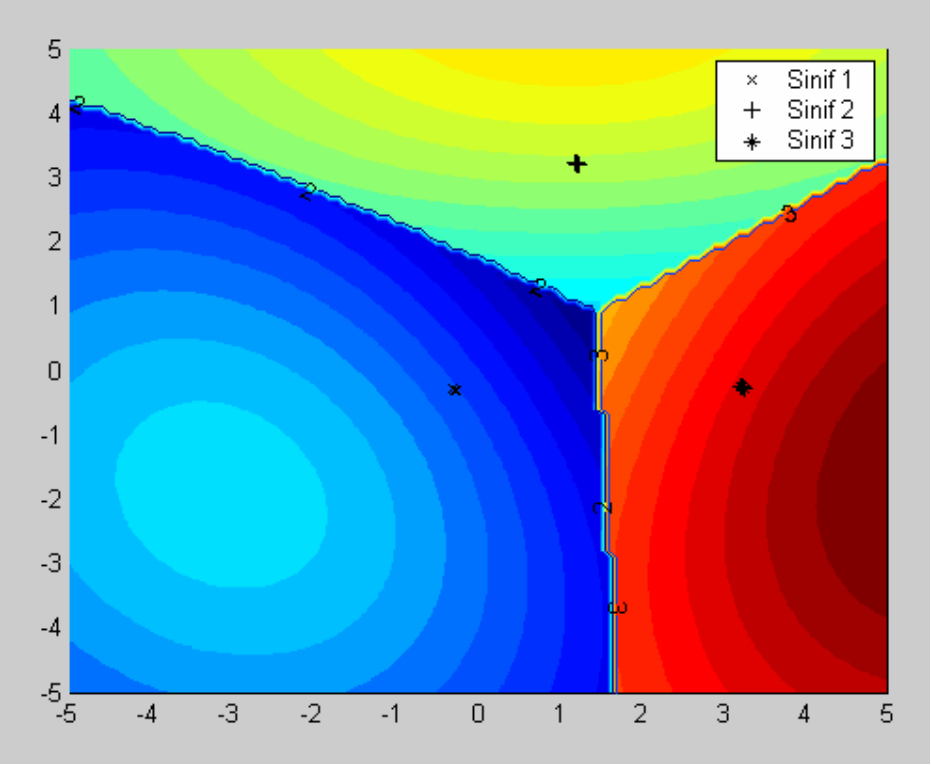
Bu uygulamada da asenkron motordan alınan veriler kullanılmıştır. Önce motorun sağlıklı durumundan veriler alınmıştır. Daha sonra ikinci motorun bir rotoru delinip bir kırık rotor arızası oluşturulmuştur. Bu veriler de alındıktan sonra bir rotor daha delinip ikinci kırık rotor elde edilmiştir. Motordan alınan sağlıklı motor verileri ve arızalı motor verileri park vektör işaret işleme tekniğinden geçirilmiştir. Bu uygulamada, sadece iki tane sınıf değil, üç sınıf birbirinden ayrılmıştır. Bir sağlıklı duruma ek olarak, bir ve iki kırık rotor çubuğu durumları sınıflandırılmıştır.

4.4.1. Simülasyon Sonuçları

Motordan alınan veriler MATLAB programında sınıflandırılmıştır. Bu çalışmada çoklu sınıflandırma yapılmıştır. Çoklu sınıflandırma metotlarından birine-karşı-biri yöntemi kullanılmıştır. Birine-karşı-biri metodunda önce iki sınıf alınır, bu iki sınıf birbirinden ayrılır,

daha sonra diğer ikili gruplar sınıflandırılır. Son olarak da birbirinden ayrılan bütün sınıflar birleştirme yöntemi kullanılarak çoklu sınıflandırma işlemi gerçekleştirilmiş olur.

Bu uygulamada kernel fonksiyonu olarak RBF kerneli kullanılmıştır ve diğer kernel fonksiyonlarına göre bu fonksiyonun daha iyi sonuç verdiği görülmüştür. RBF kernelinin genişliği 5 olarak seçilmiş ve C yaptırım parametresi de 10.000 olarak ayarlanmıştır.



Şekil 4.4. Üç-sınıflı sınıflandırma

Şekil 4.4'te sınıflandırma sonucu gösterilmiştir. Giriş verilerinin değerleri birbirine çok yakın olduğundan bu şekilin genişletilmiş hali Şekil 6.5'te verilmiştir. Burada, Sınıf 1 (x ile gösterilmiştir) sağlıklı durumu, Sınıf 2 (+ ile gösterilmiştir) bir kırık rotor çubuğu durumunu, Sınıf 3 (* ile gösterilmiştir) ise iki kırık rotor çubuğu durumunu göstermektedir. Sağlıklı durumdan elde edilen 100 veri, bir kırık rotor çubuklu durumdan elde edilen 100 veri ve iki kırık rotor çubuğu durumundan elde edilen 100 veri olmak üzere toplam 300 veri kullanılmıştır. Yapılan sınıflandırma,iyi bir sınıflandırma olarak nitelendirilebilir.

5. SONUÇLAR

Asenkron motorlar endüstride önemli bir rol oynar. Arıza teşhisleri ve durum izlemeleri geniş olarak çalışılan bir konudur. Geleneksel analitik model tabanlı arıza teşhisi yöntemlerine ek olarak veri tabanlı modellerin çeşitli türleri elektriksel makinelerin arıza teşhisi için çok popüler hale gelmiştir. Örneğin, yapay nöral ağ tabanlı uygulamaların çok çeşitli uygulamaları literatürde kolayca bulunabilir. Çok çeşitli sınıflandırma ve örüntü tanıma problemlerinde çok sıkça kullanılmış olan DVM'lerin arıza teşhisinde kullanıldığı çalışma sayısı çok azdır.

Gelişmiş metotlar asenkron motorun çevrimiçi durum izlemesinde uygulanabilmektedir. Test fazında, daha geniş ölçüm enstrümantasyonu uygulanabilir. Veri tabanlı sınıflandırma metotları özellikle geniş serilerde üretilen test motorları için uygundur, çünkü bu durumlarda, çeşitli motorlardan alınan büyük miktarda ölçüm verileri toplanabilir. Küçük serilerde üretilen motorlar dikkate alındığında, sanal ölçüm dataları üretmek yerine tam simülasyon modelleri kullanılabilir.

Elektronik bilgi hacminin artmasıyla, insanlara daha iyisini bulma ve bu kaynakları daha iyi yönetme için geliştirme araçlarındaki ilgi büyümüştür. Makine öğrenmesi metodları, DVM'leri içeren, elektronik kaynakları daha verimli organize etmeye yardım eden çok iyi bir potansiyele sahiptir.

DVM'ler istatistiksel öğrenme teorisinde iyi şekilde kurulmuş bir teoriye sahiptir ve sınıflandırma ile regresyon problemlerine yaklaşım için uygundur. Vapnik'in teorisi eğitim kümesindeki arıza ile Vapnik-Chervonenkis(VC)-boyutuna göre ifade edilen hipotez uzayının karmaşıklığının her ikisini de küçükleyen çözümün bulunduğunu göstermektir. Bu bakımdan, DVM ile bulunan fonksiyon, veriye yakınlık ve çözümün karmaşıklığı arasındaki geçiştir. Özellikle iki sınıflı sınıflandırma probleminde DVM iki sınıf arasındaki sınırı büyükleyen optimal ayırt etme yüzeyini belirlemekte, yani eğitim kümesi ile ayırt etme yüzeyine en yakın noktaların arasındaki mesafeyi en büyüklemektedir. Ayrıca, optimal ayırt etme yüzeyi, destek vektörleri olarak adlandırılan veri noktalarının küçük bir kümesinde merkezlenmiş Kernel fonksiyonların doğrusal bir kombinasyonu olarak ifade edilmektedir. Genelde destek vektörleri çok az miktarda olduğu için optimal ayırt etme yüzeyinin gösterimi seyrektir, bir anlamda veri noktalarının sadece bir bölümü sınıflandırma görevi ile ilgilidir. Buna ilaveten, doğrusal kombinasyon katsayıları doğrusal eşitsizlik ve eşitlik kısıtları dahilinde bir konveks ikinci dereceli (quadratic) programlama probleminin çözümü ile belirlenmektedir. Çok katmanlı perseptron ve radyal tabanlı ağlar, uygun çekirdek fonksiyonları ile elde edilen DVM ağların özel durumları olarak gösterilebilir.

DVM son yıllarda oldukça fazla kullanılmaya başlanmıştır. Klasik yöntemlerle çözüm imkanı zor olan alanlarda DVM'ler kullanılmış ve verimli sonuçlar alınmıştır. DVM'lerin kullanıldığı alanlar veri madenciliği, metin karakterizasyonu, el yazısı tanıma, yüz tanıma, nesne tanıma ve ses tanıma sistemleridir.

DVM'ler çeşitli sınıflandırma işlerinde mükemmel sonuçlar vermesine rağmen, dezavantajlarından biri özellikle 2-sınıflı sınıflandırıcı olmalarıdır. Çoğu gerçek uygulamalarda, bir asenkron motorun arıza sınıflandırması gibi, bir çok-sınıflı sınıflandırma problemi çözülmelidir.

Bu tezde, sadece iki-sınıflı sınıflandırma yapılmamış aynı zamanda ikiden çok durum da göz önüne alınmıştır. Çoklu-sınıflı metotlardan en kullanışlısı olan birine-karşı-biri metodu kullanılmıştır. Ve uygulamaların çoğunda iyi sonuçlar elde edilmiştir.

Bu tezde, veri-tabanlı model uygulanmıştır, çünkü asenkron motorların basit analitik modelleri genellikle çeşitli motor arızalarının tanımı için yeterli değildir. Motorun geliştirilmiş sayısal modelleri, manyetik alanın sonlu eleman analizi gibi, motor performansını gösterebilir. Fakat hesapsal donanımın verimliliğinin artışına rağmen, detaylı sayısal modelleri için gereken hesaplama zamanı çok uzun olabilir. Bilgi-tabanlı modellerin dezavantajı ise bazen yeterli uzman bilgisinin mevcut olmayışıdır.

Bu tezde, DVM veri-tabanlı arıza sınıflandırma problemi için seçilmiştir. Bunun birden fazla nedeni bulunmaktadır. DVM'ler diğer geleneksel sınıflandırıcılara göre daha iyi genelleme yeteneğine sahiptir. Giriş verisi ne kadar çok olursa olsun, bu DVM yöntemi için bir dezavantaj değildir. Kısaca DVM'lerin genelleme yeteneği ve hesapsal verimliliği problemin boyutuna bağlı değildir. Söz konusu problem arıza sınıflandırma olunca bu özellikler ön plana çıkmaktadır. Çünkü arıza sayısı önemli değildir. DVM'lerin uğraştırıcı kısmı kernel fonksiyonlarının ve bu fonksiyonların parametrelerinin seçimi olabilir.

Bu tezde, kullanılan asenkron motorlardan hem sağlıklı durumlarında hem de arızalı durumlarında veriler alınmıştır. Sağlıklı duruma ek olarak, rotor çubukları delinip kırık rotor çubuğu arızaları elde edilmiştir.

Birinci uygulamada, önce iki-sınıflı bir sınıflandırma gerçekleştirilmiştir. Sağlıklı verilerle bir kırık rotor çubuğu arızası olan motor verileri sınıflandırılmıştır. Bu sınıflandırma için lineer kernel fonksiyonu seçilip işlem yapılmıştır. Daha sonra da RBF kernelinin çeşitli uygulamaları gerçekleştirilmiştir. RBF kerneli için de fonksiyonun parametresine çeşitli değerler verilip en uygunu seçilmiştir.

İkinci uygulamada, iki-sınıflı sınıflandırma için RBF kerneli seçilmiş ve bu kernel fonksiyonu için gerekli parametre ayarlamaları ve C yaptırım parametresi seçilip en iyi sonucu veren seçim araştırılmıştır.

DVM uygulamaları için kernel fonksiyonu seçimi çok önemli ve zor bir iştir. Zor olmasının nedeni herhangi bir uygulama için belirli bir kernel fonksiyonu ve parametresinin belli olmamasıdır. Üçüncü uygulamada, iki-sınıflı sınıflandırma için polinomal kernel seçilmiştir ve parametre seçimleri yapılıp sonuçlar karşılaştırılmıştır.

Dördüncü uygulamada ise, çoklu-sınıflandırma işlemi yapılmıştır. Sağlıklı motor verileri ile bir kırık rotor çubuğu ve iki kırık rotor çubuğu arızaları sınıflandırılmıştır. Toplam 300 veri sınıflandırılmıştır. Çoklu sınıflandırma işlemi gerçekleştirilirken kullanılabilecek birkaç yöntem vardır. Bu yöntemlerden en çok bilinenleri; bire-karşı-biri, bire-karşı-diğerleri ve yönlendirilmiş çevrimsiz çizge metotlarıdır. Bu çalışmada bire-karşı-biri metodu kullanılmıştır. Kernel fonksiyonu olarak RBF kerneli seçilip parametresi 5 olarak ayarlanmıştır. C yaptırım parametresi ise 10.000 alınmıştır.

KAYNAKLAR

[1] EPRI Publication EL-2678, Improved Motors for Utility Applications, Vol.1, October 1982.

[2] Chow, M. Y., Mangum, P. M., Yee, S. O., 1991, A Neural Network Approach to Real- Time Condition Monitoring of Induction Motors, IEEE Transaction on Industrial Electronics, Vol.38, No. 6, 448-453.

[3] Vapnik, V.N., The Nature of Statistical Learning Theory, Springer-Verlag, New York, 2000.

[4] Kecman, V., Learning and Soft Computing; Support Vector Machines, Neural Networks and Fuzzy Logic Models, The MIT Press, 2001.

[5] Pöyhönen, S., Support Vector Machine Based Classification in Condition Monitoring of Induction Motors, Helsinki University of Technology Control Engineering Laboratory, 2004.

[6] Burges, C., A Tutorial on Support Vector Machines for Pattern Recognition, Journal of Data Mining and Knowledge Discovery , 1998

[7] Schwenker, F., Hierarchical Support Vector Machines for Multi-Class Pattern Recognition, Proc. of 4th Int. Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Vol. 2, pp. 561 .565, 2000.

[8] Boser, B.E., Guyon, I.M., Vapnik, V.N., A Training Algorithm for Optimal Margin Classifiers., Proc. Of the 5th Annual ACM Workshop on Computational Learning Theory, pp.142-152, ACM Press, 1992.

[9] Joachims, T., Text Categorization with Support Vector Machines: Learning with Many Relevant Features., University of Dortmund, Technical Report, 1997.

[10] Pontil, M., Verri, A., Object Recognition with Support Vector Machines., IEEE Trans. On PAMI, Vol.20, pp. 637-646, 1998

[11] Chin, K.K., Support Vector Machines applied to Speech Pattern Classification**,** MPhil. Thesis, Department of Engineering, University of Cambridge, 1998.

[12] Ding, C. H.Q, Dubchak, I.,Multi-Class Protein Fold Recognition Using Support Vector Machines and Neural Networks., Bioinformatics, Vol.17, No.4, pp.349-358, April 2001.

[13] Isermann, R., Fault Diagnosis of Machines via Parameter Estimation and Knowledge Processing, Automatica, Vol.29, No.4,pp. 815-835,1993.

[14] Patton, R.J., Lopez-Toribio, C.J., Uppal, F.J., Artificial Intelligence Approaches to Fault Diagnosis, Condition Monitoring: IEE Colloquium on Machinery, External Structures and Health, pp.5/1-5/181999.

[15] Isermann, R., On Fuzzy Logic Applications for Automatic Control, Supervision, and Fault Diagnosis, IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, Vol.28, No.2, pp. 221-235, March 1998.

[16] Duda, R.O., Hart, P.E., Stork, D.G., Pattern Classification., John Wiley & Sons, Inc., 2001.

[17] Pöyhönen, S., Negrea, M., Arkkio, A., Hyötyniemi, H., Koivo, H., Support Vector Classification for Fault Diagnostics of an Electrical Machine., Proc. of the 6th Int. Conf. on Signal Processing, ICSP.02, Vol.2, pp. 1719-1722, Beijing-China, August, 2002.

[18] Pöyhönen, S., Negrea, M., Arkkio, A., Hyötyniemi, H., Koivo, H.,Fault Diagnostics of an Electrical Machine with Multiple Support Vector Classifiers., Proc. of the 17th IEEE Int. Symp. on Intelligent Control, ISIC'02, Vol. 1, pp. 373-378, Vancouver, British Columbia, Canada, October, 2002.

[19] Pöyhönen, S., Arkkio, A., Hyötyniemi, H.,Coupling Pairwise Support Vector Machines for Fault Classification., Proc. of the 5th IFAC Symp. on Fault Detection, Supervision and Safety of Technical Processes, Safeprocess2003, pp. 705-710, Washington, D.C., USA, June, 2003.

[20] Pöyhönen, S., Negrea, M., Jover, P., Arkkio, A., Hyötyniemi, H.,Numerical Magnetic Field Analysis and Signal Processing for Fault Diagnostics of Electrical Machines., The Int. Journal for Computation and Mathematics in Electrical and Electronic Engineering, COMPEL, Vol. 2, No.4, pp. 969-981, 2003.

[21] Pöyhönen, S., Jover, P., Hyötyniemi, H.,Signal Processing of Vibrations for Condition Monitoring of an Induction Motor", Proc. of the 1st IEEE-EURASIP Int. Symp. on Control, Communications, and Signal Processing, ISCCSP 2004, pp. 499-502, Hammamet, Tunisia, March 2004.

[22] Pöyhönen, S., Jover, P., Hyötyniemi, H.,Independent Component Analysis of Vibrations for Fault Diagnosis of an Induction Motor., Proc. of the IASTED Int. Conf. on Circuits, Signals, and Systems, CSS 2003, Vol. 1, pp. 203-208, May 19-21, Cancun, Mexico, 2003.

[23] Pöyhönen, S., Conti. M., Cavallini, A., Montanari, G.C., Filippetti, F.,Insulation Defect Localization through Partial Discharge Measurements and Numerical Classification., Proc. of the IEEE Int. Symp. on Industrial Electronics, ISIE 2004, pp. 417 . 422, Ajaccio, France, May 2004.

[24] Aydın, İ., Arıza Teşhisinde Veri Madenciliği ve Yumuşak Hesaplama Tekniklerinin Kullanımı, Yüksek Lisans Tezi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü,2006.

[25] Cristianini, N., Shawe–Taylor, J., An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000

[26] Inoue, T., Abe, S., Fuzzy Support Vector Machines for Pattern Classification., Proc. Of Int. Joint Conference on Neural Networks, IJCNN '01, Vol. 2, pp. 1449 -1454, 2001.

[27] www.princeton.edu

ÖZGEÇMİŞ

SUNA YILDIRIM

Fırat Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü
23114, ELAZIĞ

E_Posta: sunayildirim@firat.edu.tr

1978	Elazığ'da doğdu.
1989-1996	Elazığ Anadolu Lisesi'nden mesun oldu.
1997-2001	Fırat Üniversitesi Bilgisayar Mühendisliği bölümünden mezun oldu.
2002-….	Fırat Üniversitesi Tunceli Meslek Yüksek Okulu'nda Öğretim Görevlisi olarak görev yapmaktadır.

EK-1 (DESTEK VEKTÖR MAKİNELERİ VE MATLAB)

Destek Vektör Makineleri ile ilgili birçok uygulama yapılmaktadır. Aşağıda verilen örnek MATLAB'ta yapılmış olup, Destek Vektör Makineleri ile verilen örnekleri sınıflandıran bir uygulamadır. Bu program Anton Schwaighofer tarafından hazırlanmıştır [27]. Bu uygulamada kullanılan fonksiyonlar da açıklanmıştır.

Function demsvm1():Temel Destek Vektör Makinesi ile sınıflandırmayı göstermektedir. Bu fonksiyon, yapay olarak üretilmiş bir data kümesinin farklı Kernel fonksiyonlarını kullanarak DVM ile sınıflandırılmasını göstermektedir. Burada; X ile gösterilen veriler sınıflandırmaya sokulacak örnekleri, Y ile gösterilen kısım da X'teki örneklerin etiketlerini yani sınıflarını göstermektedir. x1ran ve x2ran; data kümesinin ve karar fonksiyonunun çizdirileceği aralığı göstermektedir. Çizimde kullanılan 'kare' ve 'çarpı' işaretleri sırasıyla +1 sınıfını ve -1 sınıfını göstermektedir.

```
X = [2 7; 3 6; 2 2; 8 1; 6 4; 4 8; 9 5; 9 9; 9 4; 6 9; 7 4];

Y = [ +1;  +1;  +1;  +1;  +1;  -1;  -1;  -1;  -1;  -1;  -1];

x1ran = [0 10];

x2ran = [0 10];
disp(' ');
disp('Bu program DVM uygulamaları ile ilgilidir.');
disp(' ');
ind = [1:length(Y)]'
fprintf('X%2i = (%2i, %2i) with label Y%2i = %2i\n', [ind, X, ind, Y]');
disp(' ')
disp('Noktaları çizdirmek için bir tuşa basınız.');
pause
f1 = figure;
plotdata(X, Y, x1ran, x2ran);
```

'plotdata' fonksiyonu +1 ve -1 sınıflarındaki örnekleri ve indislerini bulup örneklerin çizimini yapmaktadır. Aynı zamanda örnekleri 'kare' ve 'çarpı' olarak çizdirip yanlarında kaçıncı örnek olduklarını da belirtmektedir.

```
fprintf('\n\n\n\n');
fprintf('The data is plotted in figure %i, where\n', f1);
```

```
  net = svm(size(X, 2), 'linear', [], 10);
net = svmtrain(net, X, Y);
```

Burada kullanılan ‘svm’ ve ‘svmtrain’ fonksiyonları daha sonra anlatılacaktır.

```
f2 = figure;
plotboundary(net, x1ran, x2ran);
plotdata(X, Y, x1ran, x2ran);
plotsv(net, X, Y);
```

‘plotboundary’ fonksiyonu x1ran ve x2ran aralığında DVM karar fonksiyonunu çizdirmektedir. ‘net’ yapısı içerisinde tip, giriş sayısı, kernel, kernel parametreleri, üst sınır parametresi, DVM’nin normu, ikinci dereceden denklemin çözümü için gerekli parametreleri içeren bir yapıdır.

‘plotsv’ fonksiyonu ise her iki sınıftaki Destek Vektörlerini bulup çizdiren fonksiyondur.

```
fprintf('\n\n\n\n');
disp(' ')
pause
net = svm(size(X, 2), 'rbf', [36], 100);
net = svmtrain(net, X, Y);
f6 = figure;
plotboundary(net, x1ran, x2ran);
plotdata(X, Y, x1ran, x2ran);
plotsv(net, X, Y);
pause
delete(f1);
delete(f2);
delete(f3);
delete(f4);
delete(f5);
delete(f6);
function plotdata(X, Y, x1ran, x2ran)
hold on;
```

```
  ind = find(Y>0)
plot(X(ind,1), X(ind,2), 'ks');
ind = find(Y<0);
plot(X(ind,1), X(ind,2), 'kx');
text(X(:,1)+.2,X(:,2), int2str([1:length(Y)]'));
axis([x1ran x2ran]);
axis xy;
function plotsv(net, X, Y)
hold on;
ind = find(Y(net.svind)>0);
plot(X(net.svind(ind),1),X(net.svind(ind),2),'rs');
ind = find(Y(net.svind)<0);
plot(X(net.svind(ind),1),X(net.svind(ind),2),'rx');
function [x11, x22, x1x2out] = plotboundary(net, x1ran, x2ran)
hold on;
nbpoints = 100;
x1 = x1ran(1):(x1ran(2)-x1ran(1))/nbpoints:x1ran(2);
x2 = x2ran(1):(x2ran(2)-x2ran(1))/nbpoints:x2ran(2);
[x11, x22] = meshgrid(x1, x2);
[dummy, x1x2out] = svmfwd(net, [x11(:),x22(:)]);
x1x2out = reshape(x1x2out, [length(x1) length(x2)]);
contour(x11, x22, x1x2out, [-0.99 -0.99], 'b-');
contour(x11, x22, x1x2out, [0 0], 'k-');
contour(x11, x22, x1x2out, [0.99 0.99], 'g-');
```

'svmfwd' fonksiyonu daha sonra anlatılacaktır.

```
function net = svm(nin, kernel, kernelpar, C, use2norm, qpsolver, qpsize)
NET = SVM(NIN, KERNEL, KERNELPAR, C, USE2NORM, QPSOLVER, QPSIZE)
```

Bu fonksiyon bir DVM sınıflandırıcı için temel ayarlamaları içeren bir NET yapısı oluşturur. DVM'nin NIN (Number of Inputs) boyutunda girişi olduğu farzedilir. KERNEL, çalışılan kernel fonksiyonunu gösterir. Eğer kernel fonksiyonu ekstra parametreler gerektirirse, KERNELPAR dizisinde bu parametreler verilmelidir. Geçerli kernel fonksiyonları SVMKERNEL fonksiyonunda açıklanacaktır.

NET yapısı aşağıdaki alanlara sahiptir:

Temel DVM parametreleri:

‘type’: ‘svm’

‘nin’: NIN, giriş sayısı

‘nout’: 1, çıkış sayısı

‘kernel’: KERNEL, kernel fonksiyonu

‘kernelpar’: KERNELPAR, kernel fonksiyonu için parametreler

‘c’: C, eğitim süresince NET.alpha katsayıları için üst sınır. NET.c’nin boyutuna bağlı olarak, bu değer şu şekildedir:

LENGTH(NET.c)=1: Bütün katsayılar için aynı üst sınır

LENGTH(NET.c)=1: Pozitif (+1) ve negatif (-1) örnekler için farklı üst sınırlar. NET.c(1) pozitif örnekler için ve NET.c(2) negatif örnekler için üst sınırdır.

LENGTH(NET.c)=N: i örneğiyle alakalı alpha katsayısının üst sınırıdır. Yani her örnek için farklı üst sınırlar tanımlanır. Default değeri 1’dir.

‘use2norm’: USE2NORM, standart DVM uygulamalarında 1norm kullanılır. Bunun da default değeri 0’dır.

Eğitim süresince DVM ile ayarlanacak alanlar ise şunlardır:

‘nbexamples’: Eğitim örneklerinin sayısı

‘alpha’: Eğitimden sonra, bu alan her eğitim örneği için katsayılı bir sütun vektörü içerir. NET.alpha daha sonra hiçbir DVM rutininde kullanılmaz. Dolayısıyla, eğitimden sonra kaldırılabilir.

‘svind’: Eğitimden sonra, bu alan destek vektörü olan eğitim örneklerinin indislerini içerir.

‘sv’: Destek vektörü olan bütün örnekleri içerir.

‘svcoeff’: Eğitimden sonra, bu alan destek vektörü olan her alan için uygun eğitim örneğinin etiketinin NET.alpha zamanlarının çarpımını içerir. NET.sv’de verilen örneklerle aynı sırada verilmiştir.

‘bias’: DVM karar fonksiyonunun lineer terimidir.

‘normalw’: Örnekleri ayıran hiperdüzlemin normal vektörüdür. Bu, sadece NET.kernel=’linear’, yani lineer kernel, kullanıldığında hesaplanır.

SVMTRAIN için belirli parametreler ise şunlardır: (bu parametreler nadiren değişir)

‘qpsolver’: İkinci dereceden denklemi çözen fonksiyonu gösterir.

‘qpsize’: QP çözücüye verilmiş noktaların maksimum sayısıdır.

‘alphatol’: NET.alpha katsayılarını kapsayan bütün karşılaştırmalar için verilmiş bir toleranstır.

‘kkttol’: KKT koşullarını kontrol için toleranstır.

```
function net = svm(nin, kernel, kernelpar, C, use2norm, qpsolver, qpsize)
if nargin < 7,
  qpsize = 50;
end
if nargin < 6,
  qpsolver = '';
end
if nargin < 5,
  use2norm = 0;
end
if nargin < 4,
  C = 1;
end
if nargin < 3,
  kernelpar = [];
end
net.type = 'svm';
net.nin = nin;
net.nout = 1;
net.kernel = kernel;
net.kernelpar = kernelpar;
net.c = C;
net.use2norm = use2norm;
net.nbexamples = 0;
net.alpha = [];
net.svcoeff = [];
net.sv = [];
net.svind = [];
net.bias = [];
net.normalw = [];
net.qpsolver = qpsolver;
net.qpsize = qpsize;
net.alphatol = 1e-2;
net.kkttol = 5e-2;
```

```
  net.chunksize = 500;
net.recompute = Inf;
Function net=svmtrain(net,X,Y,alpha0,dodisplay)
```

Bu fonksiyon Y hedef değerli X eğitim datasını kullanarak NET yapısıyla verilen DVM'yi eğitir. X, N eğitim örnekli (N,NET.nin) boyutlu bir matristir. Y, X'teki her örnek için hedef değerleri (sınıfları) içeren bir sütun matrisidir.
Bütün eğitim parametreleri NET yapısında verilmiştir. İlgili diğer parametreler NET.c, NET.qpsize, NET.alphatol ve NET.kkttol parametreleridir. Bu parametreler daha önce anlatılmıştı.

Fonksiyon yürütülürken iterasyonlar arasındaki tutarlılık da kontrol edilmektedir. Errstring=consist(net, 'svm', X, 'Y') satırında ağın doğru sayıda çıkışının olup olmadığına ve giriş ve çıkıştaki örüntülerin sayısının aynı olup olmadığına bakılır. Eğer hata yoksa boş bir string döner ve eğer hata varsa stringte ilgili hata mesajı görüntülenir.

```
errstring = consist(net, 'svm', X, Y);
 if ~isempty(errstring);
  error(errstring);
 end
[N, d] = size(X);
 if N==0,
  error('No training examples given');
 end
net.nbexamples = N;
 if nargin<5,
  dodisplay = 0;
end
if nargin<4,
  alpha0 = [];
elseif (~isempty(alpha0)) & (~all(size(alpha0)==[N 1])),
 error(['Initial values ALPHA0 must be a column vector with the same length' ...
          ' as X']);
end
class1 = logical(uint8(Y>=0));
class0 = logical(uint8(Y<0));
```

```
  if length(net.c(:))==1,
  C = repmat(net.c, [N 1]);
elseif length(net.c(:))==2,
  C = zeros([N 1]);
  C(class1) = net.c(1);
  C(class0) = net.c(2);
  else
  C = net.c;
  if ~all(size(C)==[N 1]),
    error(['Upper bound C must be a column vector with the same length' ...
                ' as X']);
  end
end
if min(C)<net.alphatol,
  error('NET.C must be positive and larger than NET.alphatol');
end
if ~isfield(net, 'use2norm'),
  net.use2norm = 0;
end
if ~isfield(net, 'qpsolver'),
  net.qpsolver = '';
end
qpsolver = net.qpsolver;
if isempty(qpsolver),
    checkseq = {'quadprog', 'loqo', 'qp'};
  i = 1;
  while (i <= length(checkseq)),
    e = exist(checkseq{i});
    if (e==2) | (e==3),
      qpsolver = checkseq{i};
      break;
    end
    i = i+1;
  end
  if isempty(qpsolver),
```

```
    error('No quadratic programming solver (QUADPROG,LOQO,QP) found.');
  end
end
if strcmp(qpsolver, 'quadprog') & (dodisplay==0),
  quadprogopt = optimset('Display', 'off', 'MaxIter', 500);
else
  quadprogopt = [];
end
QPsize = min(N, net.qpsize);
chsize = net.chunksize;
SVMout = zeros(N, 1);
Y(class1) = 1;
Y(class0) = -1;
if dodisplay>0,
  fprintf('Training set: %i examples (%i positive, %i negative)\n', ...
          length(Y), length(find(class1)), length(find(class0)));
end
if ~any(alpha0),
  net.alpha = zeros([N 1]);
  randomWS = 1;
else
  randomWS = 0;
   if ~net.use2norm,
   net.alpha = min(C, alpha0);
  end
end
alphaOld = net.alpha;
if length(find(Y>0))==N,
  net.bias = 1;
  net.svcoeff = [];
  net.sv = [];
  net.svind = [];
  net.alpha = zeros([N 1]);
  return;
elseif length(find(Y<0))==N,
```

```
    net.bias = 1;
  net.svcoeff = [];
  net.sv = [];
  net.svind = [];
  net.alpha = zeros([N 1]);
  return;
end
iteration = 0;
workset = logical(uint8(zeros(N, 1)));
sameWS = 0;
net.bias = 0;
while 1,
  if dodisplay>0,
    fprintf('\nIteration %i: ', iteration+1);
  end
  [net, SVthresh, SV, SVbound, SVnonbound] = findSV(net, C);
  if dodisplay>0,
     fprintf(['Working set of size %i: %i Support Vectors, %i of them at' ...
                ' bound C\n'], length(find(workset)), length(find(workset & SV)), ...
               length(find(workset & SVbound)));
    fprintf(['Whole training set: %i Support Vectors, %i of them at upper' ...
                ' bound C.\n'], length(net.svind), length(find(SVbound)));
    if dodisplay>1,
      fprintf('The Support Vectors (threshold %g) are the examples\n', ...
                 SVthresh);
      fprintf(' %i', net.svind);
      fprintf('\n');
    end
  end
  if (iteration==0) | (mod(iteration, net.recompute)==0),
    changedSV = net.svind;
    changedAlpha = net.alpha(changedSV);
    SVMout = zeros(N, 1);
    if strcmp(net.kernel, 'linear'),
      net.normalw = zeros([1 d]);
```

```
    end
  else
    changedSV = find(net.alpha~=alphaOld);
    changedAlpha = net.alpha(changedSV)-alphaOld(changedSV);
  end

  if strcmp(net.kernel, 'linear'),
    chunks = ceil(length(changedSV)/chsize);
    for ch = 1:chunks,
      ind = (1+(ch-1)*chsize):min(length(changedSV), ch*chsize);
      temp = changedAlpha(ind).*Y(changedSV(ind));
      net.normalw = net.normalw+temp'*X(changedSV(ind), :);
    end
    SVMout = zeros(N, 1);
    chunks = ceil(N/chsize);
    for ch = 1:chunks,
      ind = (1+(ch-1)*chsize):min(N, ch*chsize);
      SVMout(ind) = X(ind,:)*(net.normalw');
    end
  else
    chunks1 = ceil(N/chsize);
    chunks2 = ceil(length(changedSV)/chsize);
    for ch1 = 1:chunks1,
      ind1 = (1+(ch1-1)*chsize):min(N, ch1*chsize);
      for ch2 = 1:chunks2,
          ind2 = (1+(ch2-1)*chsize):min(length(changedSV), ch2*chsize);
          K12 = svmkernel(net, X(ind1, :), X(changedSV(ind2), :));
          coeff = changedAlpha(ind2).*Y(changedSV(ind2));
          SVMout(ind1) = SVMout(ind1)+K12*coeff;
      end
      if dodisplay>2,
          K1all = svmkernel(net, X(ind1,:), X(net.svind,:));
          coeff2 = net.alpha(net.svind).*Y(net.svind);
          fprintf('Maximum error due to matrix partitioning: %g\n', ...
                  max((SVMout(ind1)-K1all*coeff2)'));
```

```
    end
  end
end
if net.use2norm,
  workSV = find(SV & workset);
  if ~isempty(workSV),
    net.bias = mean((1-net.alpha(workSV)./C(workSV)).*Y(workSV)- ...
                SVMout(workSV));
  end
else
  workSV = find(SVnonbound & workset);
  if ~isempty(workSV),
    net.bias = mean(Y(workSV)-SVMout(workSV));
  end
end
if isempty(workSV) & (dodisplay>0),
  disp('No Support Vectors in the current working set.');
  disp('Leaving the bias unchanged.');
end
if net.use2norm,
  KKT = (SVMout+net.bias).*Y-1+net.alpha./C;
  KKTviolations = logical(uint8((SV & (abs(KKT)>net.kkttol)) | ...
                          (~SV & (KKT<-net.kkttol))));
else
  KKT = (SVMout+net.bias).*Y-1;
  KKTviolations = logical(uint8((SVnonbound & (abs(KKT)>net.kkttol)) | ...
                          (SVbound & (KKT>net.kkttol)) | ...
                          (~SV & (KKT<-net.kkttol))));
end
ind = find(KKTviolations & workset);
if ~isempty(ind),
    if dodisplay>0,
    fprintf('KKT conditions not met in working set (value %g)', ...
          max(abs(KKT(ind))));
  end
```

```
  end
if dodisplay>0,
  fprintf('%i violations of the KKT conditions.\n', ...
            length(find(KKTviolations)));
  fprintf(['(%i violations from positive examples, %i from negative' ...
            ' examples)\n'], length(find(KKTviolations & class1)), ...
            length(find(KKTviolations & class0)));
  if (dodisplay>1) & ~isempty(find(KKTviolations)),
    disp('The following examples violate the KKT conditions:');
    fprintf(' %i', find(KKTviolations));
    fprintf('\n');
  end
end
if length(find(KKTviolations)) == 0,
  break;
end
if net.use2norm,
  searchDir = SVMout+Y.*(net.alpha./C-1);
  set1 = logical(uint8(SV | class0));
  set2 = logical(uint8(SV | class1));
else
  searchDir = SVMout-Y;
  set1 = logical(uint8((SV | class0) & (~SVbound | class1)));
  set2 = logical(uint8((SV | class1) & (~SVbound | class0)));
end
if randomWS,
  searchDir = rand([N 1]);
  set1 = class1;
  set2 = class0;
  randomWS = 0;
end
worksetOld = workset;
workset = logical(uint8(zeros(N, 1)));
if length(find(set1 | set2)) <= QPsize,
  workset(set1 | set2) = 1;
```

```
    elseif length(find(set1)) <= floor(QPsize/2),
  workset(set1) = 1;
  set2 = find(set2 & ~workset);
  [dummy, ind] = sort(searchDir(set2));
  from2 = min(length(set2), QPsize-length(find(workset)));
  workset(set2(ind(1:from2))) = 1;
elseif length(find(set2)) <= floor(QPsize/2),
  workset(set2) = 1;
  set1 = find(set1 & ~workset);
  [dummy, ind] = sort(-searchDir(set1));
  from1 = min(length(set1), QPsize-length(find(workset)));
  workset(set1(ind(1:from1))) = 1;
else
  set1 = find(set1);
  [dummy, ind] = sort(-searchDir(set1));
  from1 = min(length(set1), floor(QPsize/2));
  workset(set1(ind(1:from1))) = 1;
  set2 = find(set2 & ~workset);
  dummy, ind] = sort(searchDir(set2));
  from2 = min(length(set2), QPsize-length(find(workset)));
  workset(set2(ind(1:from2))) = 1;
end
worksetind = find(workset);
if all(workset==worksetOld),
  sameWS = sameWS+1;
  if ((sameWS==3) | strcmp(qpsolver, 'loqo')),
    warnstr = 'Working set not changed - check accuracy. Exiting.';
    if dodisplay>0,
      disp(warnstr);
    end
    warning(warnstr);
    break;
  end
else
  sameWS = 0;
```

```
  end
worksize = length(find(workset));
nonworkset = ~workset;
if dodisplay>1,
  disp('Working set consists of examples ');
  fprintf(' %i', find(workset));
  fprintf('\n');
end
nonworkSV = find(nonworkset & SV);
qBN = 0;
if length(nonworkSV)>0,
  chunks = ceil(length(nonworkSV)/chsize);
  for ch = 1:chunks,
    ind = (1+(ch-1)*chsize):min(length(nonworkSV), ch*chsize);
    Ki = svmkernel(net, X(worksetind, :), X(nonworkSV(ind), :));
    qBN = qBN+Ki*(net.alpha(nonworkSV(ind)).*Y(nonworkSV(ind)));
  end
  qBN = qBN.*Y(workset);
end
f = qBN-ones(worksize, 1);
H = svmkernel(net, X(worksetind,:), X(worksetind,:));
if net.use2norm,
  H = H+diag(1./C(workset));
else
  H = H+diag(ones(worksize, 1)*eps^(2/3));
end
H = H.*(Y(workset)*Y(workset)');
A = Y(workset)';
if length(nonworkSV)>0,
  eqconstr = -net.alpha(nonworkSV)'*Y(nonworkSV);
else
  eqconstr = 0;
end
VLB = zeros(worksize, 1);
if net.use2norm,
```

```
    VUB = [];
  else
    VUB = C(workset);
  end
  tic;
  startVal = net.alpha(workset);
  switch qpsolver
    case 'quadprog'
      workalpha = quadprog(H, f, [], [], A, eqconstr, VLB, VUB, startVal, ...
                           quadprogopt);
    case 'qp'
      workalpha = qp(H, f, A, eqconstr, VLB, VUB, startVal, 1);
    case 'loqo'
      if isempty(VUB),
        VUB = repmat(1e7, size(VLB));
      end
      workalpha = loqo(H, f, A, eqconstr, VLB, VUB, startVal, 1);
  end
  t = toc;
  if dodisplay>1,
    fprintf('QP subproblem solved after %i minutes, %2.1f seconds.\n', ...
            floor(t/60), mod(t, 60));
  end
  if any(imag(workalpha)>0),
    warning(['The QP solver returned a complex solution. '...
           'Check condition number cond(H).']);
    workalpha = real(workalpha);
  end
  alphaOld = net.alpha;
  net.alpha(workset) = workalpha;
  iteration = iteration+1;
end
net.svcoeff = net.alpha(net.svind).*Y(net.svind);
net.sv = X(net.svind, :);
if dodisplay>0,
```

```
    fprintf('\n\n\nTraining finished.\n');
  svmstat(net,1);
end
function [net, SVthresh, SV, SVbound, SVnonbound] = findSV(net, C)
maxalpha = max(net.alpha);
if maxalpha > net.alphatol,
  SVthresh = net.alphatol;
else
  SVthresh = exp((log(max(eps,maxalpha))+log(eps))/2);
end
SV = logical(uint8(net.alpha>=SVthresh));
if net.use2norm,
  SVbound = logical(repmat(uint8(0), size(net.alpha)));
else
  SVbound = logical(uint8(net.alpha>(C-net.alphatol)));
end
SVnonbound = SV & (~SVbound);
net.svind = find(SV);
```

Svmkernel fonksiyonunda, SVM yapısı; NET, kernel fonksiyonunu ve parametrelerini seçen NET.kernel ve NET.kernelpar alanına sahip olmalıdır. Bu fonksiyondan seçilebilecek kerneller Lineer, Polinomal, RBF ve RBFFull kernelleridir.

```
errstring = consist(net, 'svm', X1);
if ~isempty(errstring);
  error(errstring);
end
errstring = consist(net, 'svm', X2);
if ~isempty(errstring);
  error(errstring);
end
[N1, d] = size(X1);
[N2, d] = size(X2);
switch net.kernel
  case 'linear'
    K = X1*X2';
```

```
    case 'poly'
    K = (1+X1*X2').^net.kernelpar(1);
  case 'rbf'
    dist2 = repmat(sum((X1.^2)', 1), [N2 1])' + ...
            repmat(sum((X2.^2)',1), [N1 1]) - ...
            2*X1*(X2');
    K = exp(-dist2/(net.nin*net.kernelpar(1)));
  case 'rbffull'
    bias = 0;
    if any(all(repmat(size(net.kernelpar), [4 1]) == ...
               [d 1; 1 d; d+1 1; 1 d+1], 2), 1),
      weights = diag(net.kernelpar(1:d));
      if length(net.kernelpar)>d,
        bias = net.kernelpar(end);
      end
    elseif all(size(net.kernelpar)==[d d]),
      weights = net.kernelpar;
    else
      error('Size of NET.kernelpar does not match the chosen kernel "rbffull"');
    end
    dist2 = (X1.^2)*weights*ones([d N2]) + ...
            ones([N1 d])*weights*(X2.^2)' - ...
            2*X1*weights*(X2');
    K = exp(bias-dist2/net.nin);
  otherwise
    error('Unknown kernel function');
end
K = double(K);
```

Svmfwd Fonksiyonu

Bir NET data yapısı için, X vektörlerinin matrisi NET ile tanımlanan DVM'ye giriştir ve Y çıkışlarının matrisi hesaplanmıştır. NET; boş olmayan NET.sv, NET.svcoeff ve NET.bias alanlarına sahip olmalıdır. Bu alanlar eğitim boyunca SVMtrain fonksiyonu ile ayarlanır.

```
errstring = consist(net, 'svm', X);
```

```
  if ~isempty(errstring);
  error(errstring);
end
[N d] = size(X);
if strcmp(net.kernel, 'linear'),
  if ~isfield(net, 'normalw') | ~all(size(net.normalw)==[1 d]),
    error('Structure NET does not contain a valid field "normalw"');
  end
else
  if ~isfield(net, 'sv') | ((size(net.sv, 2)~=d) & ~isempty(net.sv)),
    error('Structure NET does not contain a valid field "sv"');
  end
  nbSV = size(net.sv, 1);
  if nbSV~=size(net.svcoeff, 1),
    error('Structure NET does not contain a valid field "svcoeff"');
  end
  if ~isfield(net, 'bias') | ~all(size(net.bias)==[1 1]),
    error('Structure NET does not contain a valid field "bias"');
  end
end
if strcmp(net.kernel, 'linear'),
  Y1 = X*(net.normalw');
else
  chsize = net.chunksize;
  Y1 = zeros(N, 1);
  chunks1 = ceil(N/chsize);
  chunks2 = ceil(nbSV/chsize);
  for ch1 = 1:chunks1,
    ind1 = (1+(ch1-1)*chsize):min(N, ch1*chsize);
    for ch2 = 1:chunks2,
      ind2 = (1+(ch2-1)*chsize):min(nbSV, ch2*chsize);
      K12 = svmkernel(net, X(ind1, :), net.sv(ind2, :));
      Y1(ind1) = Y1(ind1)+K12*net.svcoeff(ind2);
    end
  end
```

```
  end
Y1 = Y1+net.bias;
Y = sign(Y1);
Y(Y==0) = 1;
```

Svmstat Fonksiyonu

Eğitim boyunca oluşan DVM istatistiklerini tutar. NET yapısında, eğitilmiş bir DVM için en önemli kısımlar hesaplanmıştır:

normw: Hiperdüzlemin normu

nbsv: Destek Vektörlerinin toplam sayısı

nbboundsv: Katsayıları üst sınırda olan DVlerin sayısı (Bunlar yanlış sınıflandırılmış eğitim örnekleridir)

possv: Pozitif örneklerdeki DVlerin indisleri

negsv: Negatif örneklerdeki DVlerin indisleri

posbound: Pozitif örneklerde sınırda olan DVlerin indisleri

negbound: Negatif örneklerde sınırda olan DVlerin indisleri

```
if nargin<2,
  doDisplay = 0;
end
nbSV = 0;
fracSV = 0;
nbPosSV = 0;
nbNegSV = 0;
normW = 0;
if (net.nbexamples<=0) | (size(net.svcoeff,1)==0),
  warning('The given SVM has not been trained yet');
  return;
end
if isfield(net, 'use2norm'),
  use2norm = net.use2norm;
else
  use2norm = 0;
end
posSV = find(net.svcoeff>0);
negSV = find(net.svcoeff<0);
```

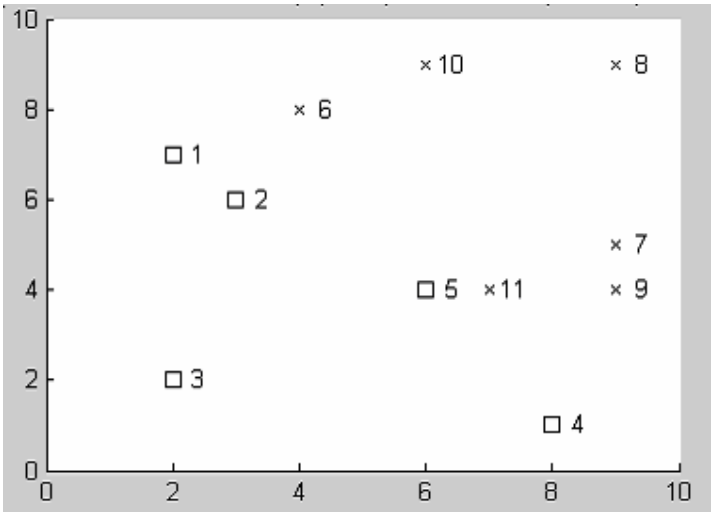
```
  nbPosSV = length(posSV);
nbNegSV = length(negSV);
nbSV = nbPosSV+nbNegSV;
fracSV = nbSV/net.nbexamples;
if (nargout<2) & ~doDisplay,
   return;
end
if length(net.c(:))==1,
  C = repmat(net.c, [length(net.svcoeff) 1]);
elseif length(net.c(:))==2,
  C = zeros([length(net.svcoeff) 1]);
  C(posSV) = net.c(1);
  C(negSV) = net.c(2);
else
  C = net.c(net.svind);
end

if isfield(net, 'alphatol'),
  tol = net.alphatol;
else
  tol = net.accur;
end
if use2norm,
  posBound = [];
  negBound = [];
else
  posBound = find(abs(net.svcoeff(posSV))>(C(posSV)-tol));
  negBound = find(abs(net.svcoeff(negSV))>(C(negSV)-tol));
end
nbPosBound = length(posBound);
nbNegBound = length(negBound);
nbBoundSV = nbPosBound+nbNegBound;
if strcmp(net.kernel, 'linear') & isfield(net, 'normalw'),
  normW = norm(net.normalw);
else
```

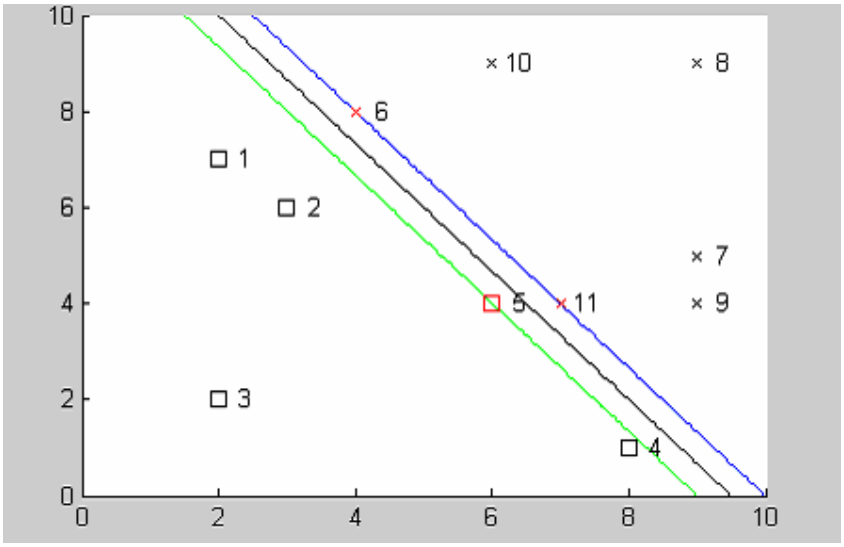
```
    if use2norm,
    alpha = abs(net.svcoeff);
    normW = sqrt(sum(alpha)+sum((alpha.^2)./C));
  else
    dummy, svOutput] = svmfwd(net, net.sv);
    svOutput = svOutput-net.bias;
    normW = sqrt(net.svcoeff'*svOutput);
    end
end
if doDisplay,
  fprintf('Number of Support Vectors (SV): %i\n', nbSV);
  fprintf('  (that is a fraction of %2.3f%% of the training examples)\n', ...
             100*fracSV);
  if ~use2norm,
    fprintf('  %i of the SV have a coefficient ALPHA at the upper bound\n', ...
          nbBoundSV);
  end
  fprintf('  %i Support Vectors from positive examples\n', nbPosSV);
  if ~use2norm,
    fprintf('     %i of them have a coefficient ALPHA at the upper bound\n', ...
          nbPosBound);
  end
  fprintf('  %i Support Vectors from negative examples\n', nbNegSV);
  if ~use2norm,
    fprintf('  %i of them have a coefficient ALPHA at the upper bound\n', ...
          nbNegBound);
  end
  fprintf('Norm of the separating hyperplane: %g\n', normW);
end
```

EK- 2 (PROGRAM ÇIKTILARI)

MATLAB'ta hazırlanan program çalıştırıldığında elde edilen sonuçlar bu bölümde verilecektir. Başlangıçta 11 tane giriş elemanı vardır ve bu elemanlardan +1 sınıfına ait olanlar kare, -1 sınıfına ait olanlar ise çarpı olarak gösterilmiştir. Şekil 1'de bu durum gösterilmiş

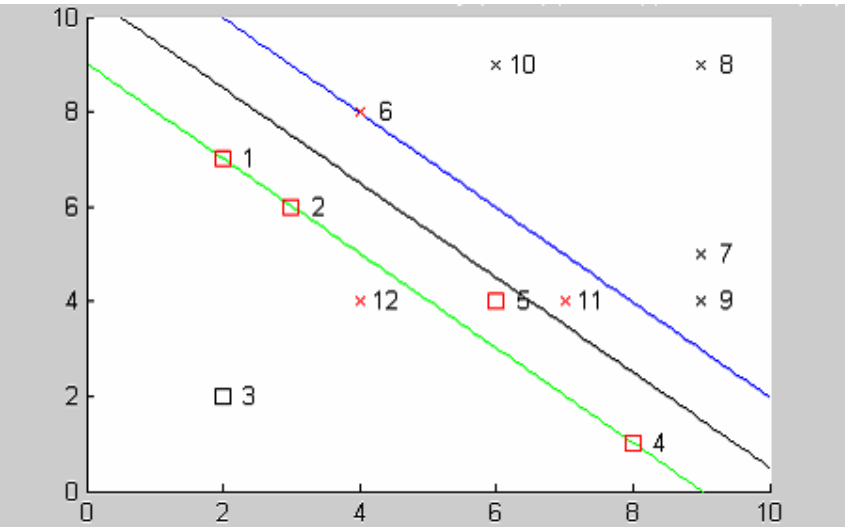


Şekil 1. X girişleri verilen veri kümesinin çizdirilmesi



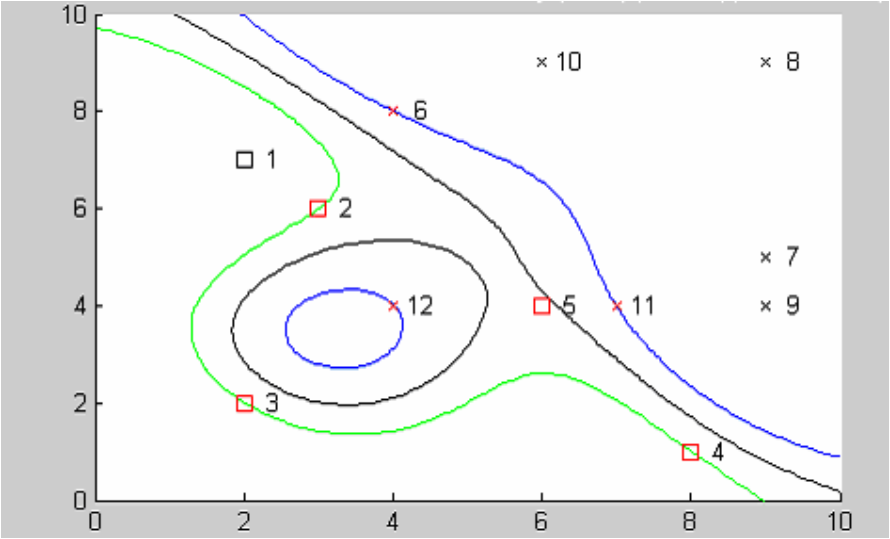
Şekil 2. İki sınıfı birbirinden ayıran hiperdüzlem

Şekil 2'de iki sınıfı birbirinden ayıran hiperdüzlem gösterilmiştir. Lineer kernel kullanılmıştır. Siyah renkle gösterilen hiperdüzlem iki sınıfı birbirinden ayıran optimal hiperdüzlemdir. Mavi ve yeşil çizgiler ise hiperdüzleme en yakın noktalarla hiperdüzlen arasındaki mesafeyi göstermektedir. Kırmızı renkte gösterilen noktalar hiperdüzleme en yakın noktalar, yani Destek Vektörleridir.



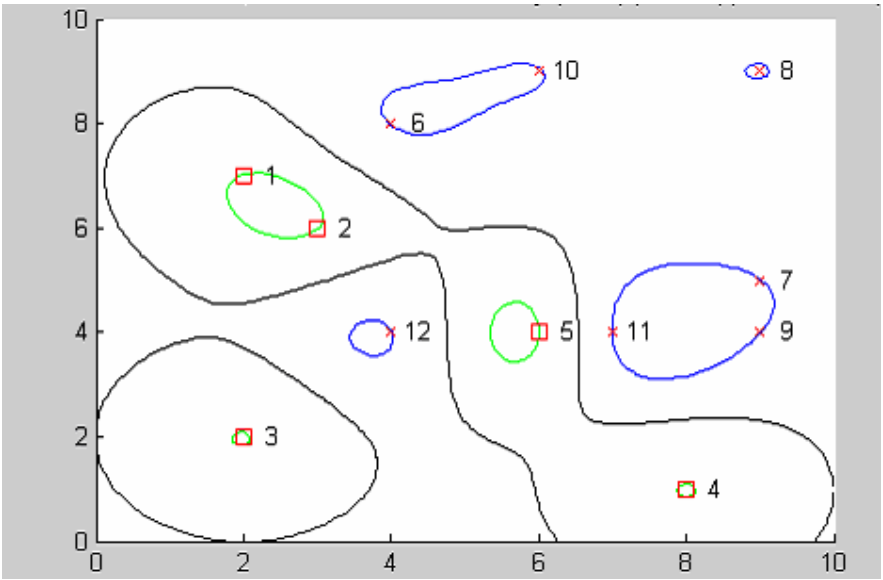
Şekil 3. 12. noktanın eklenmesinin ardından sınıflandırma

Şekil 3'te diğer verilere ek olarak sonradan bir X12 noktası eklenmiştir. Kullanılan kernel fonksiyonu yine lineer kerneldir. Bu noktanın eklenmesi sınıflandırmada Destek Vektörleri sayısını artırmış ve yanlış sınıflandırılmış bir veriye yol açmıştır. Bu şekilde, Destek Vektör Makineleri iyi bir sonuç vermemiştir. Hiperdüzleme ne kadar çok yakın veri varsa sınıflandırma o kadar kötüdür.



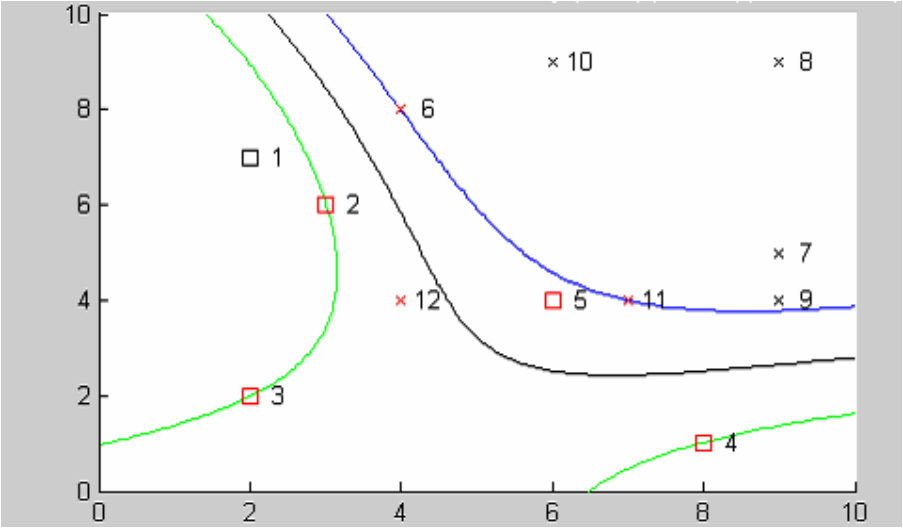
Şekil 4. RBF kerneli kullanılması ile elde edilen durum

Şekil 4'te farklı bir kernel fonksiyonu kullanılmıştır. Genişliği 8 olan, RBF kerneli kullanılmıştır. Burada da 12. noktanın etrafında bir adacık oluşmuştur.



Şekil 5. Genişliği 1 olan RBF kerneli ile sınıflandırma

Şekil 5'te RBF kernelinin parametresi değiştirilerek program çıktısı alınmıştır.



Şekil 6. Genişliği 36 olan RBF kerneli ile sınıflandırma

Şekil 6'da yine kernel fonksiyonu olarak RBF kerneli kullanılmış, fakat bu defa parametre 36 olarak seçilmiştir. Kernel fonksiyonlarında parametre seçimi tamamen kullanıcıya kalmış bir konudur. Şekil 2'ye benzer bir durum sözkonusudur.

EK-1 (DESTEK VEKTÖR MAKİNELERİ VE MATLAB)

Destek Vektör Makineleri ile ilgili birçok uygulama yapılmaktadır. Aşağıda verilen örnek MATLAB'ta yapılmış olup, Destek Vektör Makineleri ile verilen örnekleri sınıflandıran bir uygulamadır. Bu program Anton Schwaighofer tarafından hazırlanmıştır [27]. Bu uygulamada kullanılan fonksiyonlar da açıklanmıştır.

Function demsvm1():Temel Destek Vektör Makinesi ile sınıflandırmayı göstermektedir. Bu fonksiyon, yapay olarak üretilmiş bir data kümesinin farklı Kernel fonksiyonlarını kullanarak DVM ile sınıflandırılmasını göstermektedir. Burada; X ile gösterilen veriler sınıflandırmaya sokulacak örnekleri, Y ile gösterilen kısım da X'teki örneklerin etiketlerini yani sınıflarını göstermektedir. x1ran ve x2ran; data kümesinin ve karar fonksiyonunun çizdirileceği aralığı göstermektedir. Çizimde kullanılan 'kare' ve 'çarpı' işaretleri sırasıyla +1 sınıfını ve -1 sınıfını göstermektedir.

```
X = [2 7; 3 6; 2 2; 8 1; 6 4; 4 8; 9 5; 9 9; 9 4; 6 9; 7 4];
Y = [ +1;  +1;  +1;  +1;  +1;  -1;  -1;  -1;  -1;  -1;  -1];
x1ran = [0 10];
x2ran = [0 10];
disp(' ');
disp('Bu program DVM uygulamaları ile ilgilidir.');
disp(' ');
ind = [1:length(Y)]'
fprintf('X%2i = (%2i, %2i) with label Y%2i = %2i\n', [ind, X, ind, Y]');
disp(' ')
disp('Noktaları çizdirmek için bir tuşa basınız.');
pause
f1 = figure;
plotdata(X, Y, x1ran, x2ran);
```

'plotdata' fonksiyonu +1 ve -1 sınıflarındaki örnekleri ve indislerini bulup örneklerin çizimini yapmaktadır. Aynı zamanda örnekleri 'kare' ve 'çarpı' olarak çizdirip yanlarında kaçıncı örnek olduklarını da belirtmektedir.

```
fprintf('\n\n\n\n');
fprintf('The data is plotted in figure %i, where\n', f1);
```

```
net = svm(size(X, 2), 'linear', [], 10);
net = svmtrain(net, X, Y);
```

Burada kullanılan ‘svm’ ve ‘svmtrain’ fonksiyonları daha sonra anlatılacaktır.

```
f2 = figure;
plotboundary(net, x1ran, x2ran);
plotdata(X, Y, x1ran, x2ran);
plotsv(net, X, Y);
```

‘plotboundary’ fonksiyonu x1ran ve x2ran aralığında DVM karar fonksiyonunu çizdirmektedir. ‘net’ yapısı içerisinde tip, giriş sayısı, kernel, kernel parametreleri, üst sınır parametresi, DVM’nin normu, ikinci dereceden denklemin çözümü için gerekli parametreleri içeren bir yapıdır.

‘plotsv’ fonksiyonu ise her iki sınıftaki Destek Vektörlerini bulup çizdiren fonksiyondur.

```
fprintf('\n\n\n\n');
disp(' ')
pause
net = svm(size(X, 2), 'rbf', [36], 100);
net = svmtrain(net, X, Y);
f6 = figure;
plotboundary(net, x1ran, x2ran);
plotdata(X, Y, x1ran, x2ran);
plotsv(net, X, Y);
pause
delete(f1);
delete(f2);
delete(f3);
delete(f4);
delete(f5);
delete(f6);
function plotdata(X, Y, x1ran, x2ran)
hold on;
```

```
ind = find(Y>0)
plot(X(ind,1), X(ind,2), 'ks');
ind = find(Y<0);
plot(X(ind,1), X(ind,2), 'kx');
text(X(:,1)+.2,X(:,2), int2str([1:length(Y)]'));
axis([x1ran x2ran]);
axis xy;
function plotsv(net, X, Y)
hold on;
ind = find(Y(net.svind)>0);
plot(X(net.svind(ind),1),X(net.svind(ind),2),'rs');
ind = find(Y(net.svind)<0);
plot(X(net.svind(ind),1),X(net.svind(ind),2),'rx');
function [x11, x22, x1x2out] = plotboundary(net, x1ran, x2ran)
hold on;
nbpoints = 100;
x1 = x1ran(1):(x1ran(2)-x1ran(1))/nbpoints:x1ran(2);
x2 = x2ran(1):(x2ran(2)-x2ran(1))/nbpoints:x2ran(2);
[x11, x22] = meshgrid(x1, x2);
[dummy, x1x2out] = svmfwd(net, [x11(:),x22(:)]);
x1x2out = reshape(x1x2out, [length(x1) length(x2)]);
contour(x11, x22, x1x2out, [-0.99 -0.99], 'b-');
contour(x11, x22, x1x2out, [0 0], 'k-');
contour(x11, x22, x1x2out, [0.99 0.99], 'g-');
```

‘svmfwd’ fonksiyonu daha sonra anlatılacaktır.

```
function net = svm(nin, kernel, kernelpar, C, use2norm, qpsolver, qpsize)
NET = SVM(NIN, KERNEL, KERNELPAR, C, USE2NORM, QPSOLVER, QPSIZE)
```

Bu fonksiyon bir DVM sınıflandırıcı için temel ayarlamaları içeren bir NET yapısı oluşturur. DVM’nin NIN (Number of Inputs) boyutunda girişi olduğu farzedilir. KERNEL, çalışılan kernel fonksiyonunu gösterir. Eğer kernel fonksiyonu ekstra parametreler gerektirirse, KERNELPAR dizisinde bu parametreler verilmelidir. Geçerli kernel fonksiyonları SVMKERNEL fonksiyonunda açıklanacaktır.

NET yapısı aşağıdaki alanlara sahiptir:

Temel DVM parametreleri:

‘type’: ‘svm’

‘nin’: NIN, giriş sayısı

‘nout’: 1, çıkış sayısı

‘kernel’: KERNEL, kernel fonksiyonu

‘kernelpar’: KERNELPAR, kernel fonksiyonu için parametreler

‘c’: C, eğitim süresince NET.alpha katsayıları için üst sınır. NET.c’nin boyutuna bağlı olarak, bu değer şu şekildedir:

LENGTH(NET.c)=1: Bütün katsayılar için aynı üst sınır

LENGTH(NET.c)=1: Pozitif (+1) ve negatif (-1) örnekler için farklı üst sınırlar. NET.c(1) pozitif örnekler için ve NET.c(2) negatif örnekler için üst sınırdır.

LENGTH(NET.c)=N: i örneğiyle alakalı alpha katsayısının üst sınırıdır. Yani her örnek için farklı üst sınırlar tanımlanır. Default değeri 1’dir.

‘use2norm’: USE2NORM, standart DVM uygulamalarında 1norm kullanılır. Bunun da default değeri 0’dır.

Eğitim süresince DVM ile ayarlanacak alanlar ise şunlardır:

‘nbexamples’: Eğitim örneklerinin sayısı

‘alpha’: Eğitimden sonra, bu alan her eğitim örneği için katsayılı bir sütun vektörü içerir. NET.alpha daha sonra hiçbir DVM rutininde kullanılmaz. Dolayısıyla, eğitimden sonra kaldırılabilir.

‘svind’: Eğitimden sonra, bu alan destek vektörü olan eğitim örneklerinin indislerini içerir.

‘sv’: Destek vektörü olan bütün örnekleri içerir.

‘svcoeff’: Eğitimden sonra, bu alan destek vektörü olan her alan için uygun eğitim örneğinin etiketinin NET.alpha zamanlarının çarpımını içerir. NET.sv’de verilen örneklerle aynı sırada verilmiştir.

‘bias’: DVM karar fonksiyonunun lineer terimidir.

‘normalw’: Örnekleri ayıran hiperdüzlemin normal vektörüdür. Bu, sadece NET.kernel=’linear’, yani lineer kernel, kullanıldığında hesaplanır.

SVMTRAIN için belirli parametreler ise şunlardır: (bu parametreler nadiren değişir)

‘qpsolver’: İkinci dereceden denklemi çözen fonksiyonu gösterir.

‘qpsize’: QP çözücüye verilmiş noktaların maksimum sayısıdır.

‘alphatol’: NET.alpha katsayılarını kapsayan bütün karşılaştırmalar için verilmiş bir toleranstır.

‘kkttol’: KKT koşullarını kontrol için toleranstır.

```
function net = svm(nin, kernel, kernelpar, C, use2norm, qpsolver, qpsize)
if nargin < 7,
  qpsize = 50;
end
if nargin < 6,
  qpsolver = '';
end
if nargin < 5,
  use2norm = 0;
end
if nargin < 4,
  C = 1;
end
if nargin < 3,
  kernelpar = [];
end
net.type = 'svm';
net.nin = nin;
net.nout = 1;
net.kernel = kernel;
net.kernelpar = kernelpar;
net.c = C;
net.use2norm = use2norm;
net.nbexamples = 0;
net.alpha = [];
net.svcoeff = [];
net.sv = [];
net.svind = [];
net.bias = [];
net.normalw = [];
net.qpsolver = qpsolver;
net.qpsize = qpsize;
net.alphatol = 1e-2;
net.kkttol = 5e-2;
```

```
net.chunksize = 500;
net.recompute = Inf;
Function net=svmtrain(net,X,Y,alpha0,dodisplay)
```

Bu fonksiyon Y hedef değerli X eğitim datasını kullanarak NET yapısıyla verilen DVM'yi eğitir. X, N eğitim örnekli (N,NET.nin) boyutlu bir matristir. Y, X'teki her örnek için hedef değerleri (sınıfları) içeren bir sütun matrisidir.
Bütün eğitim parametreleri NET yapısında verilmiştir. İlgili diğer parametreler NET.c, NET.qpsize, NET.alphatol ve NET.kkttol parametreleridir. Bu parametreler daha önce anlatılmıştı.

Fonksiyon yürütülürken iterasyonlar arasındaki tutarlılık da kontrol edilmektedir. Errstring=consist(net, 'svm', X, 'Y') satırında ağın doğru sayıda çıkışının olup olmadığına ve giriş ve çıkıştaki örüntülerin sayısının aynı olup olmadığına bakılır. Eğer hata yoksa boş bir string döner ve eğer hata varsa stringte ilgili hata mesajı görüntülenir.

```
errstring = consist(net, 'svm', X, Y);
 if ~isempty(errstring);
  error(errstring);
 end
[N, d] = size(X);
 if N==0,
  error('No training examples given');
 end
net.nbexamples = N;
 if nargin<5,
  dodisplay = 0;
end
if nargin<4,
  alpha0 = [];
elseif (~isempty(alpha0)) & (~all(size(alpha0)==[N 1])),
 error(['Initial values ALPHA0 must be a column vector with the same length' ...
          ' as X']);
end
class1 = logical(uint8(Y>=0));
class0 = logical(uint8(Y<0));
```

```
if length(net.c(:))==1,
  C = repmat(net.c, [N 1]);
elseif length(net.c(:))==2,
  C = zeros([N 1]);
  C(class1) = net.c(1);
  C(class0) = net.c(2);
  else
  C = net.c;
  if ~all(size(C)==[N 1]),
    error(['Upper bound C must be a column vector with the same length' ...
                ' as X']);
  end
end
if min(C)<net.alphatol,
  error('NET.C must be positive and larger than NET.alphatol');
end
if ~isfield(net, 'use2norm'),
  net.use2norm = 0;
end
if ~isfield(net, 'qpsolver'),
  net.qpsolver = '';
end
qpsolver = net.qpsolver;
if isempty(qpsolver),
    checkseq = {'quadprog', 'loqo', 'qp'};
  i = 1;
  while (i <= length(checkseq)),
    e = exist(checkseq{i});
    if (e==2) | (e==3),
      qpsolver = checkseq{i};
      break;
    end
    i = i+1;
  end
  if isempty(qpsolver),
```

```
    error('No quadratic programming solver (QUADPROG,LOQO,QP) found.');
  end
end
if strcmp(qpsolver, 'quadprog') & (dodisplay==0),
  quadprogopt = optimset('Display', 'off', 'MaxIter', 500);
else
  quadprogopt = [];
end
QPsize = min(N, net.qpsize);
chsize = net.chunksize;
SVMout = zeros(N, 1);
Y(class1) = 1;
Y(class0) = -1;
if dodisplay>0,
  fprintf('Training set: %i examples (%i positive, %i negative)\n', ...
            length(Y), length(find(class1)), length(find(class0)));
end
if ~any(alpha0),
  net.alpha = zeros([N 1]);
  randomWS = 1;
else
  randomWS = 0;
    if ~net.use2norm,
    net.alpha = min(C, alpha0);
  end
end
alphaOld = net.alpha;
if length(find(Y>0))==N,
  net.bias = 1;
  net.svcoeff = [];
  net.sv = [];
  net.svind = [];
  net.alpha = zeros([N 1]);
  return;
elseif length(find(Y<0))==N,
```

```
  net.bias = 1;
  net.svcoeff = [];
  net.sv = [];
  net.svind = [];
  net.alpha = zeros([N 1]);
  return;
end
iteration = 0;
workset = logical(uint8(zeros(N, 1)));
sameWS = 0;
net.bias = 0;
while 1,
  if dodisplay>0,
    fprintf('\nIteration %i: ', iteration+1);
  end
  [net, SVthresh, SV, SVbound, SVnonbound] = findSV(net, C);
  if dodisplay>0,
     fprintf(['Working set of size %i: %i Support Vectors, %i of them at' ...
                ' bound C\n'], length(find(workset)), length(find(workset & SV)), ...
               length(find(workset & SVbound)));
    fprintf(['Whole training set: %i Support Vectors, %i of them at upper' ...
                ' bound C.\n'], length(net.svind), length(find(SVbound)));
    if dodisplay>1,
      fprintf('The Support Vectors (threshold %g) are the examples\n', ...
                 SVthresh);
      fprintf(' %i', net.svind);
      fprintf('\n');
    end
  end
  if (iteration==0) | (mod(iteration, net.recompute)==0),
    changedSV = net.svind;
    changedAlpha = net.alpha(changedSV);
    SVMout = zeros(N, 1);
    if strcmp(net.kernel, 'linear'),
      net.normalw = zeros([1 d]);
```

```
  end
else
  changedSV = find(net.alpha~=alphaOld);
  changedAlpha = net.alpha(changedSV)-alphaOld(changedSV);
end

if strcmp(net.kernel, 'linear'),
  chunks = ceil(length(changedSV)/chsize);
  for ch = 1:chunks,
    ind = (1+(ch-1)*chsize):min(length(changedSV), ch*chsize);
    temp = changedAlpha(ind).*Y(changedSV(ind));
    net.normalw = net.normalw+temp'*X(changedSV(ind), :);
  end
  SVMout = zeros(N, 1);
  chunks = ceil(N/chsize);
  for ch = 1:chunks,
    ind = (1+(ch-1)*chsize):min(N, ch*chsize);
    SVMout(ind) = X(ind,:)*(net.normalw');
  end
else
  chunks1 = ceil(N/chsize);
  chunks2 = ceil(length(changedSV)/chsize);
  for ch1 = 1:chunks1,
    ind1 = (1+(ch1-1)*chsize):min(N, ch1*chsize);
    for ch2 = 1:chunks2,
        ind2 = (1+(ch2-1)*chsize):min(length(changedSV), ch2*chsize);
        K12 = svmkernel(net, X(ind1, :), X(changedSV(ind2), :));
        coeff = changedAlpha(ind2).*Y(changedSV(ind2));
        SVMout(ind1) = SVMout(ind1)+K12*coeff;
    end
    if dodisplay>2,
        K1all = svmkernel(net, X(ind1,:), X(net.svind,:));
        coeff2 = net.alpha(net.svind).*Y(net.svind);
        fprintf('Maximum error due to matrix partitioning: %g\n', ...
                max((SVMout(ind1)-K1all*coeff2)'));
```

```
    end
  end
end
if net.use2norm,
  workSV = find(SV & workset);
  if ~isempty(workSV),
    net.bias = mean((1-net.alpha(workSV)./C(workSV)).*Y(workSV)- ...
                SVMout(workSV));
  end
else
  workSV = find(SVnonbound & workset);
  if ~isempty(workSV),
    net.bias = mean(Y(workSV)-SVMout(workSV));
  end
end
if isempty(workSV) & (dodisplay>0),
  disp('No Support Vectors in the current working set.');
  disp('Leaving the bias unchanged.');
end
if net.use2norm,
  KKT = (SVMout+net.bias).*Y-1+net.alpha./C;
  KKTviolations = logical(uint8((SV & (abs(KKT)>net.kkttol)) | ...
                          (~SV & (KKT<-net.kkttol))));
else
  KKT = (SVMout+net.bias).*Y-1;
  KKTviolations = logical(uint8((SVnonbound & (abs(KKT)>net.kkttol)) | ...
                          (SVbound & (KKT>net.kkttol)) | ...
                          (~SV & (KKT<-net.kkttol))));
end
ind = find(KKTviolations & workset);
if ~isempty(ind),
    if dodisplay>0,
    fprintf('KKT conditions not met in working set (value %g)', ...
          max(abs(KKT(ind))));
  end
```

```
end
if dodisplay>0,
  fprintf('%i violations of the KKT conditions.\n', ...
              length(find(KKTviolations)));
  fprintf(['(%i violations from positive examples, %i from negative' ...
               ' examples)\n'], length(find(KKTviolations & class1)), ...
              length(find(KKTviolations & class0)));
  if (dodisplay>1) & ~isempty(find(KKTviolations)),
    disp('The following examples violate the KKT conditions:');
    fprintf(' %i', find(KKTviolations));
    fprintf('\n');
  end
end
if length(find(KKTviolations)) == 0,
  break;
end
if net.use2norm,
  searchDir = SVMout+Y.*(net.alpha./C-1);
  set1 = logical(uint8(SV | class0));
  set2 = logical(uint8(SV | class1));
else
  searchDir = SVMout-Y;
  set1 = logical(uint8((SV | class0) & (~SVbound | class1)));
  set2 = logical(uint8((SV | class1) & (~SVbound | class0)));
end
if randomWS,
  searchDir = rand([N 1]);
  set1 = class1;
  set2 = class0;
  randomWS = 0;
end
worksetOld = workset;
workset = logical(uint8(zeros(N, 1)));
if length(find(set1 | set2)) <= QPsize,
  workset(set1 | set2) = 1;
```

```
  elseif length(find(set1)) <= floor(QPsize/2),
  workset(set1) = 1;
  set2 = find(set2 & ~workset);
  [dummy, ind] = sort(searchDir(set2));
  from2 = min(length(set2), QPsize-length(find(workset)));
  workset(set2(ind(1:from2))) = 1;
elseif length(find(set2)) <= floor(QPsize/2),
  workset(set2) = 1;
  set1 = find(set1 & ~workset);
  [dummy, ind] = sort(-searchDir(set1));
  from1 = min(length(set1), QPsize-length(find(workset)));
  workset(set1(ind(1:from1))) = 1;
else
  set1 = find(set1);
  [dummy, ind] = sort(-searchDir(set1));
  from1 = min(length(set1), floor(QPsize/2));
  workset(set1(ind(1:from1))) = 1;
  set2 = find(set2 & ~workset);
  dummy, ind] = sort(searchDir(set2));
  from2 = min(length(set2), QPsize-length(find(workset)));
  workset(set2(ind(1:from2))) = 1;
end
worksetind = find(workset);
if all(workset==worksetOld),
  sameWS = sameWS+1;
  if ((sameWS==3) | strcmp(qpsolver, 'loqo')),
    warnstr = 'Working set not changed - check accuracy. Exiting.';
    if dodisplay>0,
      disp(warnstr);
    end
    warning(warnstr);
    break;
  end
else
  sameWS = 0;
```

```
end
worksize = length(find(workset));
nonworkset = ~workset;
if dodisplay>1,
  disp('Working set consists of examples ');
  fprintf(' %i', find(workset));
  fprintf('\n');
end
nonworkSV = find(nonworkset & SV);
qBN = 0;
if length(nonworkSV)>0,
  chunks = ceil(length(nonworkSV)/chsize);
  for ch = 1:chunks,
    ind = (1+(ch-1)*chsize):min(length(nonworkSV), ch*chsize);
    Ki = svmkernel(net, X(worksetind, :), X(nonworkSV(ind), :));
    qBN = qBN+Ki*(net.alpha(nonworkSV(ind)).*Y(nonworkSV(ind)));
  end
  qBN = qBN.*Y(workset);
end
f = qBN-ones(worksize, 1);
H = svmkernel(net, X(worksetind,:), X(worksetind,:));
if net.use2norm,
  H = H+diag(1./C(workset));
else
  H = H+diag(ones(worksize, 1)*eps^(2/3));
end
H = H.*(Y(workset)*Y(workset)');
A = Y(workset)';
if length(nonworkSV)>0,
  eqconstr = -net.alpha(nonworkSV)'*Y(nonworkSV);
else
  eqconstr = 0;
end
VLB = zeros(worksize, 1);
if net.use2norm,
```

```
    VUB = [];
  else
    VUB = C(workset);
  end
  tic;
  startVal = net.alpha(workset);
  switch qpsolver
    case 'quadprog'
      workalpha = quadprog(H, f, [], [], A, eqconstr, VLB, VUB, startVal, ...
                           quadprogopt);
    case 'qp'
      workalpha = qp(H, f, A, eqconstr, VLB, VUB, startVal, 1);
    case 'loqo'
      if isempty(VUB),
        VUB = repmat(1e7, size(VLB));
      end
      workalpha = loqo(H, f, A, eqconstr, VLB, VUB, startVal, 1);
  end
  t = toc;
  if dodisplay>1,
    fprintf('QP subproblem solved after %i minutes, %2.1f seconds.\n', ...
              floor(t/60), mod(t, 60));
  end
  if any(imag(workalpha)>0),
    warning(['The QP solver returned a complex solution. '...
             'Check condition number cond(H).']);
    workalpha = real(workalpha);
  end
  alphaOld = net.alpha;
  net.alpha(workset) = workalpha;
  iteration = iteration+1;
end
net.svcoeff = net.alpha(net.svind).*Y(net.svind);
net.sv = X(net.svind, :);
if dodisplay>0,
```

```
  fprintf('\n\n\nTraining finished.\n');
  svmstat(net,1);
end
function [net, SVthresh, SV, SVbound, SVnonbound] = findSV(net, C)
maxalpha = max(net.alpha);
if maxalpha > net.alphatol,
  SVthresh = net.alphatol;
else
  SVthresh = exp((log(max(eps,maxalpha))+log(eps))/2);
end
SV = logical(uint8(net.alpha>=SVthresh));
if net.use2norm,
  SVbound = logical(repmat(uint8(0), size(net.alpha)));
else
  SVbound = logical(uint8(net.alpha>(C-net.alphatol)));
end
SVnonbound = SV & (~SVbound);
net.svind = find(SV);
```

Svmkernel fonksiyonunda, SVM yapısı; NET, kernel fonksiyonunu ve parametrelerini seçen NET.kernel ve NET.kernelpar alanına sahip olmalıdır. Bu fonksiyondan seçilebilecek kerneller Lineer, Polinomal, RBF ve RBFFull kernelleridir.

```
errstring = consist(net, 'svm', X1);
if ~isempty(errstring);
  error(errstring);
end
errstring = consist(net, 'svm', X2);
if ~isempty(errstring);
  error(errstring);
end
[N1, d] = size(X1);
[N2, d] = size(X2);
switch net.kernel
  case 'linear'
    K = X1*X2';
```

```
  case 'poly'
    K = (1+X1*X2').^net.kernelpar(1);
  case 'rbf'
    dist2 = repmat(sum((X1.^2)', 1), [N2 1])' + ...
            repmat(sum((X2.^2)',1), [N1 1]) - ...
            2*X1*(X2');
    K = exp(-dist2/(net.nin*net.kernelpar(1)));
  case 'rbffull'
    bias = 0;
    if any(all(repmat(size(net.kernelpar), [4 1]) == ...
               [d 1; 1 d; d+1 1; 1 d+1], 2), 1),
      weights = diag(net.kernelpar(1:d));
      if length(net.kernelpar)>d,
        bias = net.kernelpar(end);
      end
    elseif all(size(net.kernelpar)==[d d]),
      weights = net.kernelpar;
    else
      error('Size of NET.kernelpar does not match the chosen kernel "rbffull"');
    end
    dist2 = (X1.^2)*weights*ones([d N2]) + ...
            ones([N1 d])*weights*(X2.^2)' - ...
            2*X1*weights*(X2');
    K = exp(bias-dist2/net.nin);
  otherwise
    error('Unknown kernel function');
end
K = double(K);
```

Svmfwd Fonksiyonu

Bir NET data yapısı için, X vektörlerinin matrisi NET ile tanımlanan DVM'ye giriştir ve Y çıkışlarının matrisi hesaplanmıştır. NET; boş olmayan NET.sv, NET.svcoeff ve NET.bias alanlarına sahip olmalıdır. Bu alanlar eğitim boyunca SVMtrain fonksiyonu ile ayarlanır.

```
errstring = consist(net, 'svm', X);
```

```
if ~isempty(errstring);
  error(errstring);
end
[N d] = size(X);
if strcmp(net.kernel, 'linear'),
  if ~isfield(net, 'normalw') | ~all(size(net.normalw)==[1 d]),
    error('Structure NET does not contain a valid field "normalw"');
  end
else
  if ~isfield(net, 'sv') | ((size(net.sv, 2)~=d) & ~isempty(net.sv)),
    error('Structure NET does not contain a valid field "sv"');
  end
  nbSV = size(net.sv, 1);
  if nbSV~=size(net.svcoeff, 1),
    error('Structure NET does not contain a valid field "svcoeff"');
  end
  if ~isfield(net, 'bias') | ~all(size(net.bias)==[1 1]),
    error('Structure NET does not contain a valid field "bias"');
  end
end
if strcmp(net.kernel, 'linear'),
  Y1 = X*(net.normalw');
else
  chsize = net.chunksize;
  Y1 = zeros(N, 1);
  chunks1 = ceil(N/chsize);
  chunks2 = ceil(nbSV/chsize);
  for ch1 = 1:chunks1,
    ind1 = (1+(ch1-1)*chsize):min(N, ch1*chsize);
    for ch2 = 1:chunks2,
      ind2 = (1+(ch2-1)*chsize):min(nbSV, ch2*chsize);
      K12 = svmkernel(net, X(ind1, :), net.sv(ind2, :));
      Y1(ind1) = Y1(ind1)+K12*net.svcoeff(ind2);
    end
  end
```

```
end
Y1 = Y1+net.bias;
Y = sign(Y1);
Y(Y==0) = 1;
```

Svmstat Fonksiyonu

Eğitim boyunca oluşan DVM istatistiklerini tutar. NET yapısında, eğitilmiş bir DVM için en önemli kısımlar hesaplanmıştır:

normw: Hiperdüzlemin normu

nbsv: Destek Vektörlerinin toplam sayısı

nbboundsv: Katsayıları üst sınırda olan DVlerin sayısı (Bunlar yanlış sınıflandırılmış eğitim örnekleridir)

possv: Pozitif örneklerdeki DVlerin indisleri

negsv: Negatif örneklerdeki DVlerin indisleri

posbound: Pozitif örneklerde sınırda olan DVlerin indisleri

negbound: Negatif örneklerde sınırda olan DVlerin indisleri

```
if nargin<2,
  doDisplay = 0;
end
nbSV = 0;
fracSV = 0;
nbPosSV = 0;
nbNegSV = 0;
normW = 0;
if (net.nbexamples<=0) | (size(net.svcoeff,1)==0),
  warning('The given SVM has not been trained yet');
  return;
end
if isfield(net, 'use2norm'),
  use2norm = net.use2norm;
else
  use2norm = 0;
end
posSV = find(net.svcoeff>0);
negSV = find(net.svcoeff<0);
```

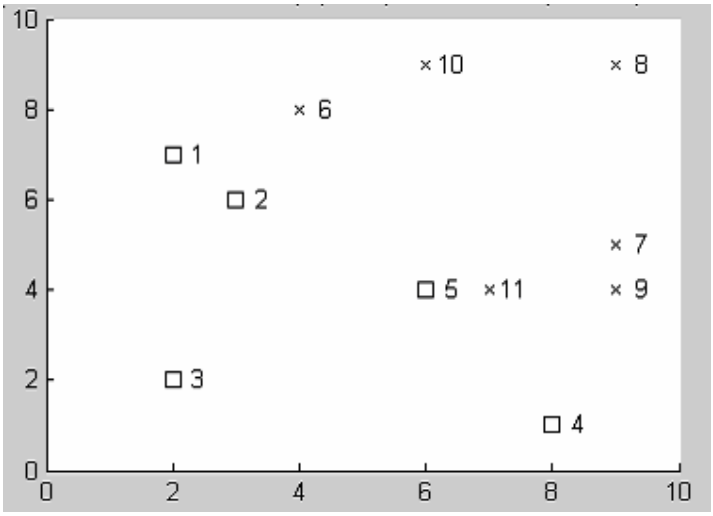
```
nbPosSV = length(posSV);
nbNegSV = length(negSV);
nbSV = nbPosSV+nbNegSV;
fracSV = nbSV/net.nbexamples;
if (nargout<2) & ~doDisplay,
   return;
end
if length(net.c(:))==1,
  C = repmat(net.c, [length(net.svcoeff) 1]);
elseif length(net.c(:))==2,
  C = zeros([length(net.svcoeff) 1]);
  C(posSV) = net.c(1);
  C(negSV) = net.c(2);
else
  C = net.c(net.svind);
end

if isfield(net, 'alphatol'),
  tol = net.alphatol;
else
  tol = net.accur;
end
if use2norm,
  posBound = [];
  negBound = [];
else
  posBound = find(abs(net.svcoeff(posSV))>(C(posSV)-tol));
  negBound = find(abs(net.svcoeff(negSV))>(C(negSV)-tol));
end
nbPosBound = length(posBound);
nbNegBound = length(negBound);
nbBoundSV = nbPosBound+nbNegBound;
if strcmp(net.kernel, 'linear') & isfield(net, 'normalw'),
  normW = norm(net.normalw);
else
```

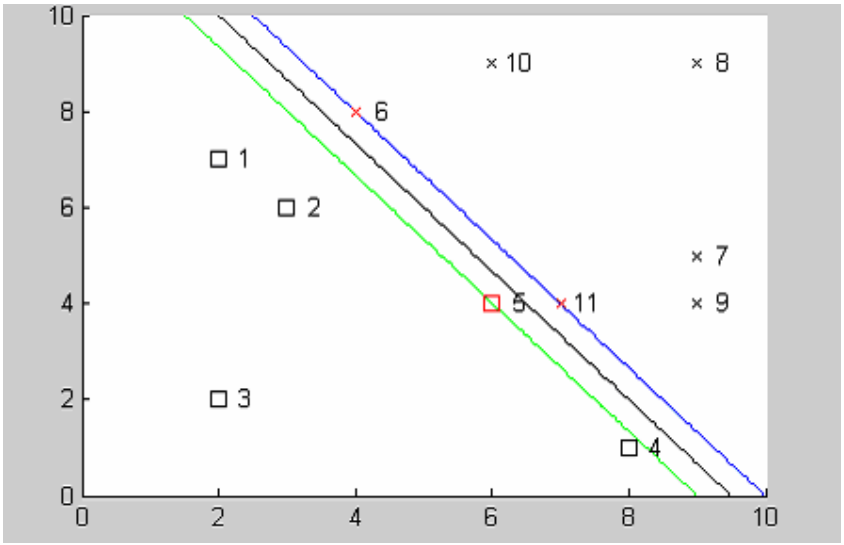
```
  if use2norm,
    alpha = abs(net.svcoeff);
    normW = sqrt(sum(alpha)+sum((alpha.^2)./C));
  else
    dummy, svOutput] = svmfwd(net, net.sv);
    svOutput = svOutput-net.bias;
    normW = sqrt(net.svcoeff'*svOutput);
    end
end
if doDisplay,
  fprintf('Number of Support Vectors (SV): %i\n', nbSV);
  fprintf('  (that is a fraction of %2.3f%% of the training examples)\n', ...
            100*fracSV);
  if ~use2norm,
    fprintf('  %i of the SV have a coefficient ALPHA at the upper bound\n', ...
          nbBoundSV);
  end
  fprintf('  %i Support Vectors from positive examples\n', nbPosSV);
  if ~use2norm,
    fprintf('     %i of them have a coefficient ALPHA at the upper bound\n', ...
          nbPosBound);
  end
  fprintf('  %i Support Vectors from negative examples\n', nbNegSV);
  if ~use2norm,
    fprintf('  %i of them have a coefficient ALPHA at the upper bound\n', ...
          nbNegBound);
  end
  fprintf('Norm of the separating hyperplane: %g\n', normW);
end
```

EK- 2 (PROGRAM ÇIKTILARI)

MATLAB'ta hazırlanan program çalıştırıldığında elde edilen sonuçlar bu bölümde verilecektir. Başlangıçta 11 tane giriş elemanı vardır ve bu elemanlardan +1 sınıfına ait olanlar kare, -1 sınıfına ait olanlar ise çarpı olarak gösterilmiştir. Şekil 1'de bu durum gösterilmiş

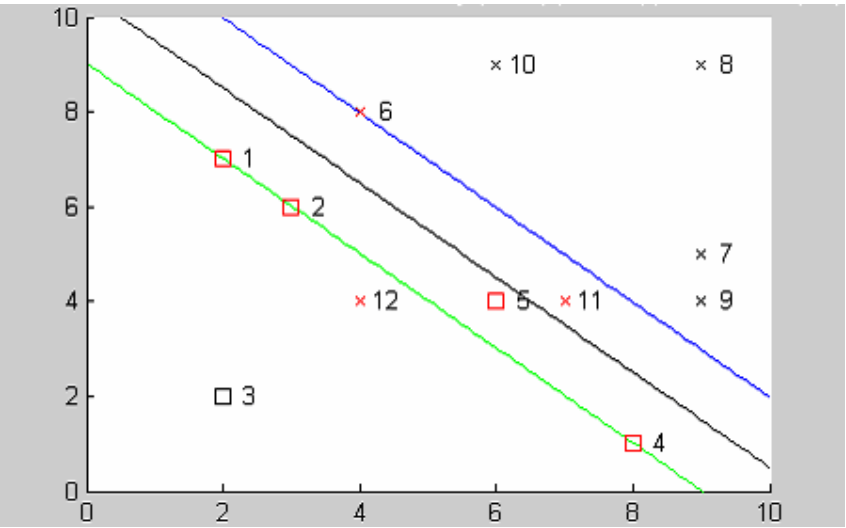


Şekil 1. X girişleri verilen veri kümesinin çizdirilmesi



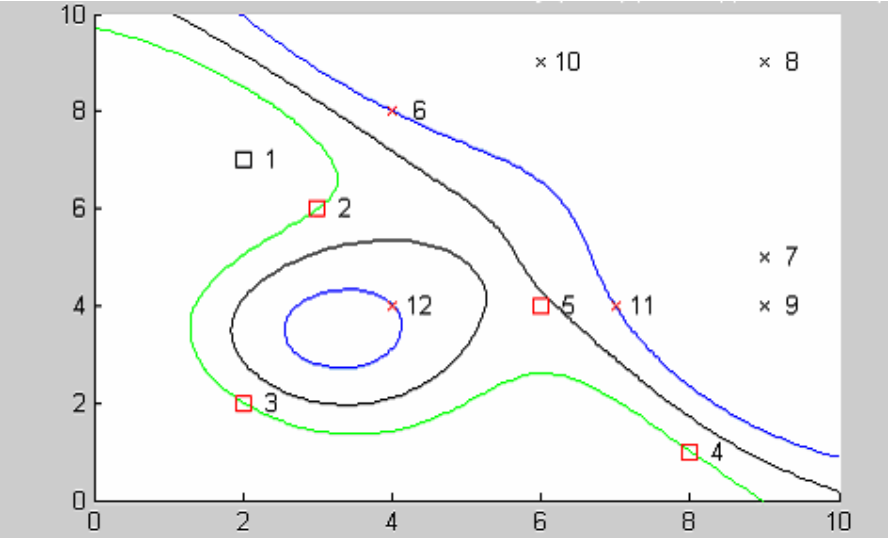
Şekil 2. İki sınıfı birbirinden ayıran hiperdüzlem

Şekil 2'de iki sınıfı birbirinden ayıran hiperdüzlem gösterilmiştir. Lineer kernel kullanılmıştır. Siyah renkle gösterilen hiperdüzlem iki sınıfı birbirinden ayıran optimal hiperdüzlemdir. Mavi ve yeşil çizgiler ise hiperdüzleme en yakın noktalarla hiperdüzlen arasındaki mesafeyi göstermektedir. Kırmızı renkte gösterilen noktalar hiperdüzleme en yakın noktalar, yani Destek Vektörleridir.



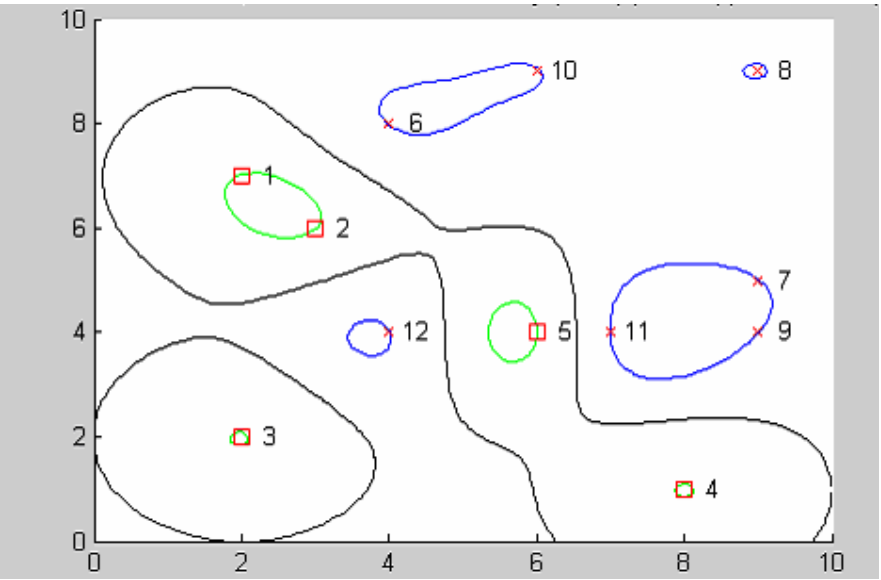
Şekil 3. 12. noktanın eklenmesinin ardından sınıflandırma

Şekil 3'te diğer verilere ek olarak sonradan bir X12 noktası eklenmiştir. Kullanılan kernel fonksiyonu yine lineer kerneldir. Bu noktanın eklenmesi sınıflandırmada Destek Vektörleri sayısını artırmış ve yanlış sınıflandırılmış bir veriye yol açmıştır. Bu şekilde, Destek Vektör Makineleri iyi bir sonuç vermemiştir. Hiperdüzleme ne kadar çok yakın veri varsa sınıflandırma o kadar kötüdür.



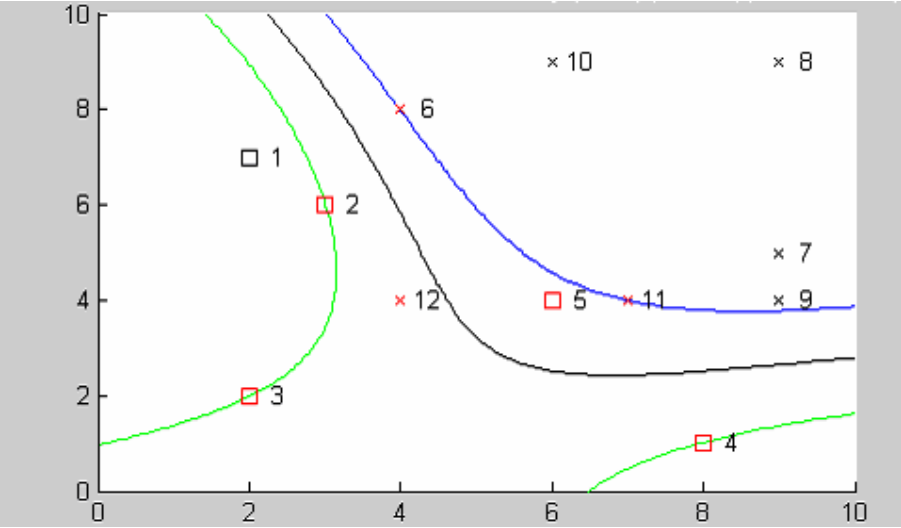
Şekil 4. RBF kerneli kullanılması ile elde edilen durum

Şekil 4'te farklı bir kernel fonksiyonu kullanılmıştır. Genişliği 8 olan, RBF kerneli kullanılmıştır. Burada da 12. noktanın etrafında bir adacık oluşmuştur.



Şekil 5. Genişliği 1 olan RBF kerneli ile sınıflandırma

Şekil 5'te RBF kernelinin parametresi değiştirilerek program çıktısı alınmıştır.



Şekil 6. Genişliği 36 olan RBF kerneli ile sınıflandırma

Şekil 6'da yine kernel fonksiyonu olarak RBF kerneli kullanılmış, fakat bu defa parametre 36 olarak seçilmiştir. Kernel fonksiyonlarında parametre seçimi tamamen kullanıcıya kalmış bir konudur. Şekil 2'ye benzer bir durum sözkonusudur.