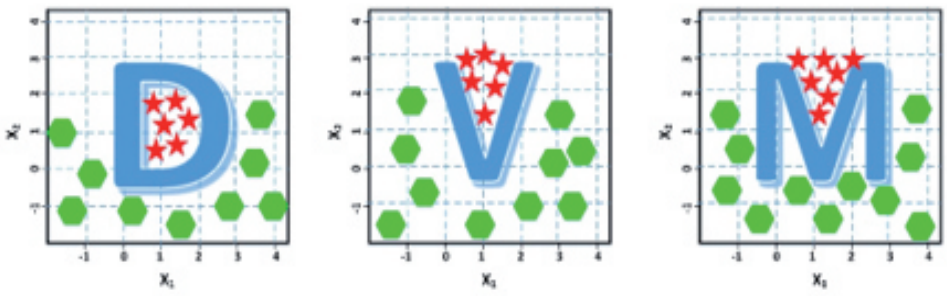
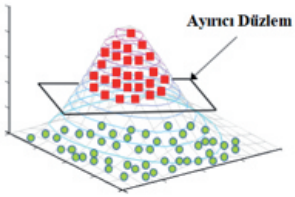


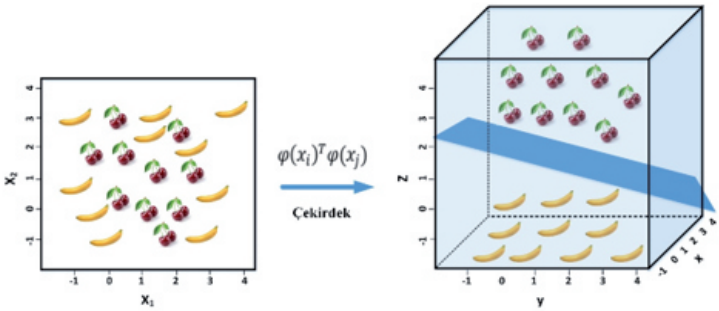
MAKİNE ÖĞRENMESİNDE, TEORİDEN ÖRNEK MATLAB UYGULAMALARINA KADAR

DESTEK VEKTÖR MAKİNELERİ

Dr. Öğr. Üyesi Sedat METLEK

Dr. Öğr. Üyesi Kıyas KAYAALP





İKSAD Publishing House

MAKİNE ÖĞRENMESİNDE,

TEORİDEN ÖRNEK MATLAB UYGULAMALARINA KADAR

DESTEK VEKTÖR MAKİNELERİ

Dr. Öğr. Üyesi Sedat METLEK **Dr. Öğr. Üyesi Kıyas KAYAALP**

İKSAD Publishing House

Copyright © 2020 by iksad publishing house
All rights reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by
any means, including photocopying, recording or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of
brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. Institution of Economic Development and Social
Researches Publications®
(The Licence Number of Publicator: 2014/31220)
TURKEY TR: +90 342 606 06 75
USA: +1 631 685 0 853
E mail: iksadyayinevi@gmail.com
www.iksadyayinevi.com

It is responsibility of the author to abide by the publishing ethics rules.
Iksad Publications – 2020©

ISBN: 978-625-7279-74-1
Cover Design: İbrahim KAYA
December / 2020
Ankara / Turkey
Size = 16 x 24 cm

ÖNSÖZ

“Makineler düşünebilir mi?” sorusunu ortaya atan Alan Mathisan Turing “Yapay zeka” kavramının ortaya çıkmasına neden olmuştur. Yapay zeka ile bilgisayarların gelişimi benzer tarihlerde olsa da 1970’li yıllarda yapay zeka üzerine yapılan çalışmalar bir ara sekteye uğramıştır. Sonraki yıllarda özellikle XOR probleminin bir yapay sinir ağı modeli ile çözülmesinden sonra gerekli finansman desteği almış ve hızlı bir gelişim sağlamıştır.

Sonraki yıllarda ise, yapay zeka’nın bir alt dalı olan Makine Öğrenme teoremi ile yapay zeka alanında yapılan çalışmalar, ikinci bir ivme daha kazanmıştır. Günümüzde makine öğrenmesinden sonra 21. yüzyılın başlarında Derin Öğrenme kavramı ile büyük miktardaki verilerden otomatik olarak öznitelik çıkarma işlemleri de gerçekleştirilmeye başlanmıştır. Böylece yapay zeka kavramı sağlıktan askeriyeye, sanal ticaretten tarıma kadar birçok alanda uygulama geliştirilmesinde temel oluşturmuştur.

Makine öğrenmesi ve Derin Öğrenme algoritmaları temel bazı algoritmalar üzerinden gelişimine devam etmektedir. Genellikle geliştirilen makine öğrenme modelleri denetimli ve denetimsiz öğrenme olmak üzere iki temel yapıda geliştirilmektedir. Bu öğrenme modelleri dışında da destekleyici öğrenme modeli gibi literatürde bazı öğrenme modelleri de bulunmaktadır.

Denetimli öğrenme mimarisinde kullanılan veriler, etiketli olmak zorundadır. Denetimli öğrenme modelinde kullanılan algoritmalar

eğitim verileri ile çıkış değerleri arasındaki ilişkiyi keşfeder. Denetimsiz öğrenme mimarisinde kullanılan veriler ise, etiketsiz verilerdir. Bu mimari, etiketsiz verilerin arasındaki ilişkinin keşfedilmesi mantığına dayanır. Her iki mimaride de keşfedilen ilişkiler, test verileri üzerinde uygulanarak sistemin başarısı ölçülmektedir.

Makine öğrenme alanında, sınıflandırma ve regresyondaki başarısından dolayı, denetimli bir mimariye sahip olan Destek Vektör Makineleri birçok uygulamada tercih edilmektedir.

Bu kitapta da son derece geniş kullanım alanına sahip olan Destek Vektör Makineleri hakkında ilk önce teorik bilgiler verilmiş, sonrasında da örnek uygulamalar üzerinde kullanımı anlatılmıştır. Hazırlanan kitap bu yönüyle başta Yazılım, Elektrik-Elektronik, Mekatronik, Bilgisayar Teknolojileri olmak üzere, İktisat, Ekonomi gibi diğer bilim dallarında da Destek Vektör Makinelerini kullanmak isteyen araştırmacılara, yeterli düzeyde bilgi sahibi olmalarını sağlayacak kaynak bir kitap niteliğindedir. Hazırlanan kitabın akademiye, bilime ve makine öğrenmesi konularında meraklı okuyuculara da yararlı olmasını temenni ediyoruz.

Saygılarımızla...

Dr. Öğr. Üyesi. Sedat METLEK **Dr. Öğr. Üyesi. Kıyas KAYAALP**

Aralık, 2020

İÇİNDEKİLER

ÖNSÖZ i
ŞEKİLLER DİZİNİ v
TABLOLAR DİZİNİ vii
BÖLÜM 1 - GİRİŞ 1
1. Yapay Zeka 1
2. Makine Öğrenmesi 2
3. Derin Öğrenme 3
4. Makine Öğrenimi Türleri 3
4. 1. Denetimli Öğrenme 4
4.1.1. Sınıflandırma 6
4.1.2. Regresyon 10
4.2. Denetimsiz Öğrenme 14
BÖLÜM 2 - DESTEK VEKTÖR MAKİNELERİ 21
1. Destek Vektör Makineleri 21
2. DVM'nin Kullanım Alanları 22
3. DVM Algoritmalarında Kullanılan Veri Yapılarının Önemi 26
3.1. Tam Ayrıştırılabilir Veri 26
3.2. Tam Ayrıştırılamayan Veri 41
4. Çekirdekler 47
4.1. Doğrusal Çekirdek 51
4.2. Polinom Çekirdek 51
4.3. Gauss Çekirdek 51
4.4. Laplace Çekirdek 52
4.5. Anova Çekirdek 52
4.6. Sigmoid Çekirdek 53

4.7. Bessel Çekirdek 53
5. Çoklu Sınıf 56
5.1. Tüm Çiftler 56
5.2. Biri Diğerlerine Karşı 57
BÖLÜM 3 - MATLAB UYGULAMALARI 59
3. Matlab Uygulamaları 59
3.1. fitcSVM fonksiyonu 59
3.1.1. fitcSVM fonksiyonunun parametreleri 61
3.2. Uygulama 1 65
3.2.1. Ionosphere Veri Seti 65
3.2.2. Matlab Uygulaması 66
3.3. Uygulama 2 68
3.3.1. Iris Veri Seti 68
3.3.2. Matlab Uygulaması 70
3.4. Uygulama 3 75
3.4.1. Iris Veri Seti 75
3.4.2. Matlab Uygulaması 75
3.5. Uygulama 4 83
3.5.1. Iris Veri Seti 83
3.5.2. Matlab Uygulaması 83
3.6. Uygulama 5 89
3.6.1. Veri Seti 89
3.6.2. Matlab Uygulaması 89
KAYNAKÇA 93
ÖZGEÇMİŞLER 97

ŞEKİLLER DİZİNİ

Şekil 1: Yapay Zekanın Gelişim Süreci..... 2
Şekil 2: Makine Öğrenme Modelleri..... 4
Şekil 3: Denetimli Öğrenme Modeli 5
Şekil 4: Denetimsiz Öğrenme 15
Şekil 5: DVM Uygulama Alanları.....22
Şekil 6: İki Farklı Sınıfın Düzlemler ile Ayrılması.....27
Şekil 7: $\beta^T X + \beta_0$ Doğrusu 28
Şekil 8: (a) $\beta^T X + \beta_0$ Doğrusunun İki Sınıfı Birbirinden Ayırması, (b) Yeni Eklenen Değer.....29
Şekil 9: $\beta TX + \beta 0 = 0$ Doğrusunun Konumu 30
Şekil 10: Tam Ayrıştırılabilir Veri 31
Şekil 11: Düzlemler Arasındaki Mesafe.....33
Şekil 12: $\beta Txi + \beta 0 - 1 = 0 \quad ve \quad \beta Txi + \beta 0 + 1 = 0$ Doğruları 35
Şekil 13: İki Doğru Arasındaki Fark 36
Şekil 14: İki Doğru Arasındaki Mesafenin β ile İlişkisi.....37
Şekil 15: Yeni Eklenen Değere Göre Destek Vektörünün Yeniden Belirlenmesi....40
Şekil 16: Tam Ayrıştırılamayan Veri ($\beta Tx + \beta 0 - 1 = 0$ Doğrusu).....42
Şekil 17: Tam Ayrıştırılamayan Veri ($\beta Tx + \beta 0 + 1 = 0$ Doğrusu).....43
Şekil 18: İki Doğru (Düzlem) Arasındaki Mesafe 44
Şekil 19: c Parametresine Göre Esneme Paylarının Gösterimi.....46
Şekil 20: Birbirinden Bir Doğru İle Ayrılamayan Veriler.....47
Şekil 21: Çekirdek Fonksiyonları.....54
Şekil 22: Boyut Arttırma.....54
Şekil 23: Çekirdek Fonksiyon ile Ayırıcı Düzlemi Belirleme.....55
Şekil 24: Tüm Çiftler.....56
Şekil 25: Biri Diğerine Karşı 57
Şekil 26. fisheriris Veri Setinin İçeriği 71
Şekil 27: İris Veri Setinde Tek Sınıf İçin Tespit Edilen Aykırı Değerler 74
Şekil 28: Iris Veri Seti.....75

Şekil 29: Versicolor ve Virginia Sınıflarına Ait Dağılım 77
Şekil 30: Classtur Değişkeninin İçeriği 78
Şekil 31: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları 79
Şekil 32: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (RBF) 81
Şekil 33: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları 81
Şekil 34: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (Polynomial-PolynomialOrder,2) 82
Şekil 35: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (Polynomial-PolynomialOrder,4) 82
Şekil 36: Iris Veri Setinin Çoklu Sınıflandırılması 88
Şekil 37: Simule Edilen Veriler 90
Şekil 38: Simule Edilmiş Verilerin Destek Vektörleri ile Gösterimi 92

TABLOLAR DİZİNİ

Tablo 1: Veri Tablosu 61
Tablo 2: Tüm Veri Tablosu 62
Tablo 3: Çekirdek Fonksiyonları 63
Tablo 4: fitcSVM Nesne Fonksiyonları 64
Tablo 5: Ionosphere Veri Seti 66
Tablo 6: Iris Veri Setindeki Çiçek Türleri.....69
Tablo 7: Iris Veri Seti.....70
Tablo 8: Destek Vektör Noktaları.....79
Tablo 9: Test Edilen Fonksiyon ve Parametreler.....80

BÖLÜM 1 - GİRİŞ

1. Yapay Zeka

Yapay zeka, veriler üzerinden tahmin ve karar verebilme işlemlerini gerçekleştiren esnek hesaplama yöntemlerine dayanan algoritmalara verilen genel isimdir. Yapay zeka kavramı ilk kez 1956 yılında ABD'de ortaya çıkmıştır. Teknolojinin ve esnek hesaplama sistemlerinin gelişmesine bağlı olarak, kendi içerisinde de alt sınıflara ayrılmıştır.

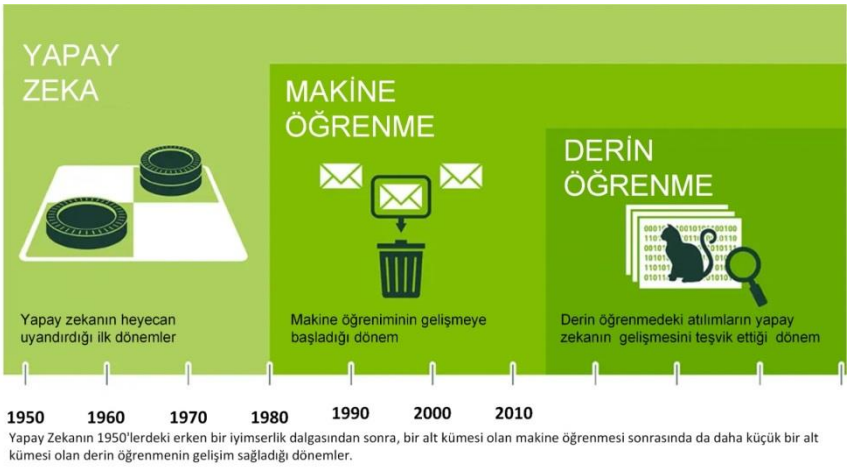
1980'li yılların başlarında, Yapay Zeka kavramına bir alt sınıf olarak Makine Öğrenmesi kavramı eklenmiştir. 2010'lu yıllarda ise Makine Öğrenmesi kavramına ek olarak Derin Öğrenme kavramı literatüre girmiştir.

Her teknolojinin geliştirilme evresinde savaşlar önemli rol oynamaktadır. Yapay Zeka kavramının geliştirilmesinde de ABD-Rusya arasındaki soğuk savaş yıllarının etkisi olmuştur. ABD hükümeti, Rus istihbaratının şifreli yazışmalarını çözmek için Yapay Zeka kavramının geliştirilmesi konusunda, finansal destek sağlamıştır. Sağlanan bu finansal destekler 1973 yılına kadar yeterli gelişme sağlanamadığı için kesilmiştir.

Sonraki yıllarda, bir Yapay Sinir Ağı (YSA) modeli ile XOR probleminin çözümü gerçekleştirildikten sonrada tüm gözler tekrar Yapay Zeka kavramına çevrilmiştir.

Yapay Zeka kavramının tekrar popüler hale gelmesinin sonucunda, özel şirketlerde Yapay Zeka sistemlerine bütçe ayırmaya başlamışlardır. Bu şirketlerden biriside IBM'dir. 1996 yılında, IBM'in geliştirdiği Yapay Zeka yazılımı (Deep Blue) ile Dünya Satranç Şampiyonu Kasparov arasında bir satranç maçı dahi oynanmıştır (Karaküçük & Eker, 2020).

Şekil 1'de, Yapay Zeka kavramının Derin Öğrenme kavramına kadar gelişim evreleri yıllara göre detaylı olarak gösterilmiştir.



Şekil 1: Yapay Zekanın Gelişim Süreci (Copeland, 2020)

2. Makine Öğrenmesi

Bilgisayarlı görme, örüntü tanıma, sınıflandırma, doğal dil işleme ve biyoinformatik gibi birçok farklı alanda kullanılan Makine Öğrenmesi, giriş verileri ile çıkış verileri arasında doğrusal bir formül ile ifade edilemeyen problemleri çözebilmek için geliştirilmiş güncel esnek hesaplama sistemlerine verilen genel addır.

Çok az sayıda veri yada yanlış veri ile geliştirilen Makine Öğrenme modellerinin doğruluğu, yeterli miktarda ve doğru veri ile geliştirilen modellere göre oldukça düşüktür. Bu nedenle Makine Öğrenmesi sonucunda elde edilen bilgiler ve başarım oranı tamamen giriş verilerine bağlıdır.

Makine Öğrenme modelleri denetimli, denetimsiz ve destekleyici olmak üzere üçe ayrılmasına rağmen, literatürde genel olarak denetimli ve denetimsiz olarak sunulmaktadır.

3. Derin Öğrenme

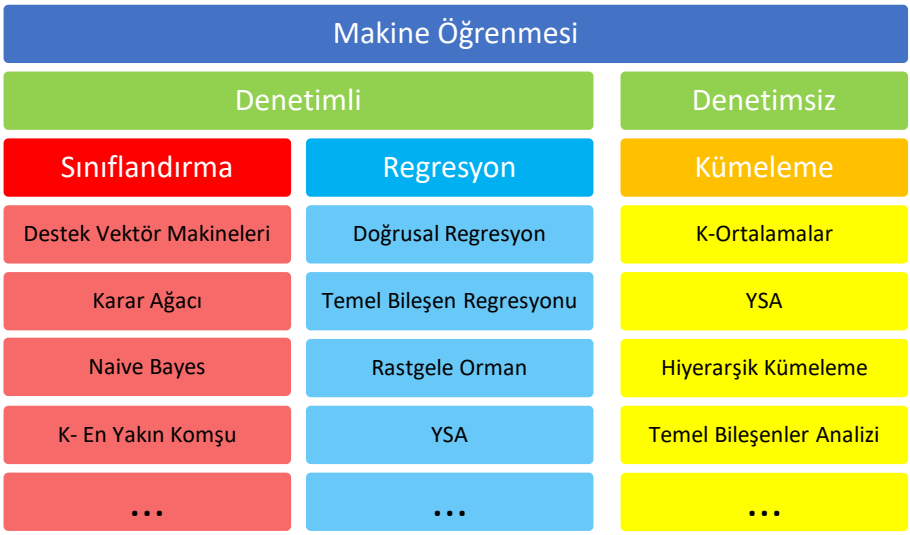
Derin öğrenme, günümüzde Yapay Zeka ve Makine Öğrenmesinin geldiği son noktadır. Derin öğrenme, Yapay Zeka ve Makine Öğrenme modellerinden farklı olarak, büyük miktarlardaki verilerden otomatik olarak öznitelik çıkarma işlemlerini kolayca gerçekleştirebilmektedir. Bu yönüyle diğer esnek hesaplama yöntemlerinden ayrılmaktadır.

Derin öğrenme modellerinden elde edilen öznitelikler, yine Yapay Zeka ve Makine Öğrenmesi modellerinde olduğu gibi, denetimli ve denetimsiz öğrenme modellerinde kullanılmaktadır.

4. Makine Öğrenimi Türleri

Makine öğrenme algoritmalarını temelde, denetimli öğrenme, denetimsiz öğrenme ve destekleyici öğrenme olmak üzere üç farklı kategoride sınıflandırılsada, literatürde birçok yerde Şekil 2'de gösterildiği üzere denetimli ve denetimsiz olmak üzere iki grupta sunulmaktadır.

Denetimli öğrenme modelleri genel olarak sınıflandırma ve regresyon uygulamalarında kullanılmaktadır. Bunun yanında, Denetimsiz Öğrenme Modelleri ise, daha çok kümeleme uygulamalarında kullanılmaktadır. Bu modellerde tercih edilen bazı algoritmalar Şekil 2'de detaylı olarak sunulmuştur.



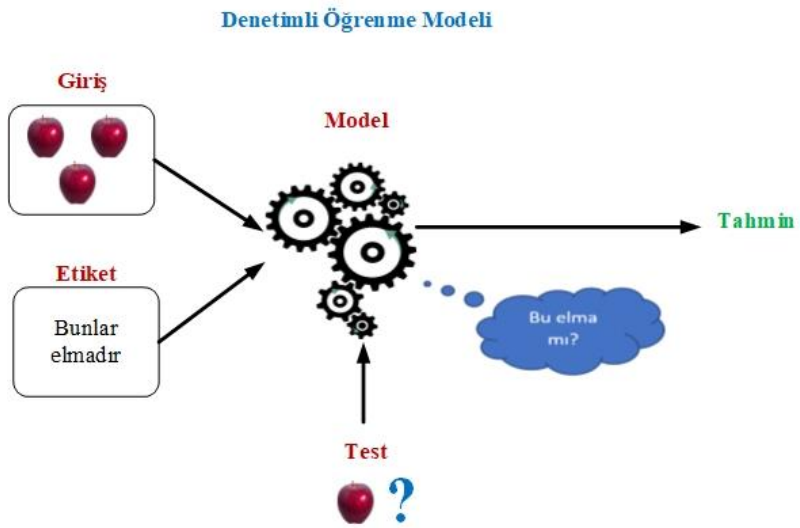
Şekil 2: Makine Öğrenme Modelleri

4. 1. Denetimli Öğrenme

Makine öğrenme uygulamalarında çok sık kullanılan yöntemlerden biriside Şekil 3'de gösterilen denetimli öğrenme modelidir. Giriş-çıkış verileri arasında kesin bir bağıntının olduğu veriler için yaygın olarak kullanılan yöntemlerdendir. Bu yöntemde kullanılan verilerin tümünün etiketli veri olması gerekmektedir. Eğitim aşamasında, bu etiketli verilerin büyük bir kısmı sistemi eğitmek, diğerleri de test etmek için

kullanılır. Eğitim için kullanılan verilerin, etiketini taşımış olduğu özelliklerin tümünü taşıması beklenir. Çünkü hazırlanacak denetimli öğrenme modeli bu etiketli veriler üzerinden öğrenme işlemini gerçekleştirmektedir. Öğrenen modelde test verileri ile tahminler veya sınıflandırmalar yapılmaktadır. Eğitim için kullanılan veriler ile eğitilen modelin, eğitme işlemi sonunda üretmiş olduğu değerlerin, gerçekten üretmesi gereken değer olup olmadığı denetlenir. Bu yüzden bu modele literatürde "Denetimli Öğrenme" modeli denilmektedir.

Genellikle denetimli öğrenme modellerinde sistemi eğitmek için kullanılan giriş verileri ile çıkış verileri arasında doğrusal bir bağlantı fonksiyonu bulunmamaktadır. Bu nedenle eğitim için uygulanan yöntemlerin geliştirilmesi ile modelin daha iyi bir sonuç üretmesi hedeflenmektedir.



Şekil 3: Denetimli Öğrenme Modeli

Sistemin öğrenmesi için sınıfı tam anlamıyla temsil eden örnek sayısı ne kadar fazla ise sistemin başarısı da o oranda yüksek olmaktadır.

Denetimli Öğrenmenin Temel Özellikleri;

- Uygulamada kullanılan eğitim setinde, giriş ve çıkış verileri bulunmak zorundadır.
- Veri setindeki eğitim verileri kullanılarak, tüm veriler hakkında bilgi verebilecek bir model oluşturulur.
- Eğitilen modele, sistemin daha önce görmediği giriş değerleri verilerek bunlara karşılık gelen çıkış değerleri tahmin edilir.

Denetimli öğrenme modelleri, temel olarak sınıflandırma ve regresyon uygulamalarında kullanılmaktadır. Sınıflandırma işleminin yapıldığı denetimli öğrenme uygulamalarında genel olarak Destek Vektör Makineleri (DVM), Karar Ağacı, Naive Bayes ve K-En Yakın Komşuluk metotları gibi esnek hesaplama metotları kullanılmaktadır. Regresyon işleminin yapıldığı denetimli öğrenme uygulamalarında ise, Doğrusal Regresyon, Temel Bileşen Regresyonu, Rastgele Orman ve Yapay Sinir Ağları modelleri gibi esnek hesaplama metotları kullanılmaktadır.

4.1.1. Sınıflandırma

Genel olarak veri setindeki verileri, önceden belirlenen etiketlere uygun olarak ayrılması işlemine sınıflandırma denilmektedir. Sınıflandırma işleminde eğitim setindeki verilerin sınıfları önceden bilindiği için, veri setini denetimli olarak öğrenmiş sayılmaktadır. Literatürde en çok

kullanılan denetimli sınıflandırma yöntemlerinden bazıları aşağıda detaylı olarak açıklanmıştır.

4.1.1.1. Destek Vektör Makineleri (DVM)

Tasarlanan DVM'lerin ilk versiyonları, ikili veri kümesini doğrusal vektörler yardımıyla sınıflara ayırmak için kullanılan güçlü sınıflandırıcılardı. Daha sonraki yıllarda geliştirilen DVM modelleri ise çekirdek fonksiyonları kullanarak iç içe geçmiş birden fazla sınıfı da kolayca ayırabilecek kabiliyettedir. Bu kabiliyetinden dolayı da birçok makine öğrenme uygulamasında tercih nedeni olmuştur. Bu kadar fazla tercih edilen DVM'ler, kitabın ilerleyen bölümlerinde detaylı olarak anlatılmış ve uygulamalar üzerinde gösterilmiştir.

4.1.1.2. Karar Ağacı (KA)

Veri setindeki, istatistiksel olarak anlamlı veri gruplarını tespit eden KA, tespit ettiği bu anlamlı ilişkileri ağaç diyagramları şeklinde göstermektedir. Bu gösterimin kolay ve anlaşılabilir olmasından dolayı, birçok makine öğrenme uygulamasında kullanılmaktadır. Girdi ve çıktı veri setlerinden oluşan KA denetimli bir öğrenme yöntemi ile çalışmaktadır. Bu denetimli yapıda KA, girdi ve çıktı arasındaki bağlantıyı tespit etmektedir (Metlek & Kayaalp, 2020).

Oluşturulan KA yapısında, kök, dal, yaprak ve bunlar arasındaki düğümler bulunmaktadır. Aynı sınıfa ait olan veriler aynı dal üzerindeki yapraklarda bulunmaktadır. Bir dal üzerinde bulunan yaprakta, olası bütün çıktı sınıflarına karşılık gelen sonuç değerleri bulunmaktadır. KA yapısında veri setinde bulunan verilerin miktarının fazla olması sonucu

etkilemez. Bu nedenle KA yapısı ile oluşturulan tekniklerin hesaplama yöntemi ucuz ve hızlıdır (Tan, Steinbach, & Kumar, 2016). Bu yapısı sayesinde birçok gömülü sistemde tercih nedeni olmuştur.

Bu yapının tek bir dezavantaj oluşturduğu durum vardır. Eğer kullanılan veri seti çok fazla sınıftan oluşuyorsa, oluşturulan yapı enine genişlemekte ve bunun sonucunda da düğüm sayısı artmaktadır. Düğüm sayısının artması sonucunda, sınıf bilgisi azalacağı için sistemin güvenirliliği düşmektedir.

4.1.1.3. Naive Bayes

Naive Bayes algoritmasının genel çalışma mantığı, aynı kriterlerin sonuca olan etkilerinin olasılıksal olarak hesaplanması temeline dayanmaktadır. Birçok yazılım uygulamasında kullanılmasının ana nedeni de budur.

Naive Bayes algoritmasında sıralı verilerin kullanılması tercih edilmez. Bu nedenle sıralı değerler içeren, bağımlı ya da bağımsız değişkenler kategorik verilere dönüştürülmelidir. Örneğin veri setinde bulunan bağımsız değişkenlerden birisi yaş ise, kategorik verilere aşağıdaki şekilde dönüştürülebilir;

- yaş ≤ 20
- $21 \leq$ yaş ≤ 30
- $31 \leq$ yaş ≤ 40

Eğitim aşamasında Naive Bayes algoritmasında, her çıktının eğitim kümesinde kaç kere meydana geldiği hesaplanır. Elde edilen değer öncelikli olasılık olarak adlandırılır. Tüm olasılıkların toplam değeri ise

1’dir. Bu olasılıklar içerisinde elde edilen en büyük değer o çıktının sınıfını verir (Akbulut, 2006).

4.1.1.4. K-En Yakın Komşu (K-Nearest Neighbors, KNN)

K-NN bilinen en eski ve basit sınıflandırma algoritmalarından birisidir. Algoritmanın genel mantığı, yeni gelen örnek bir veri ile K adet komşu arasındaki mesafenin ölçülmesine dayanmaktadır. K-NN algoritması denetimli bir algoritma olması nedeniyle kullanım esnasında veriler ilk olarak eğitim ve test olarak ikiye ayrılmaktadır. Sonrasında;

- K parametresi belirlenir. Bu parametre verilen bir noktaya en yakın komşuların sayısını ifade etmektedir. K değeri, komşu sınıflar arasındaki uzaklık ve sınıflandırma performansını doğrudan etkilemektedir.
- Örnek veri setine katılacak olan yeni verinin, mevcut verilere göre uzaklığı tek tek hesaplanır.
- Uzaklık hesaplanması için farklı fonksiyonlar kullanılabilir. Literatürde ise yaygın olarak Minkowski, Öklid, Manhattan ve Chebyschev fonksiyonları kullanılmaktadır.
- Hesaplanan uzaklıklardan en yakın K. komşu ele alınır. Öznitelik değerlerine göre K. komşunun sınıfına atanır.
- Atanılan sınıf, tahmin edilmesi beklenen sınıf değeri olarak kabul edilir. Yani yeni verinin sınıfı bulunmuş olur (Metlek & Kayaalp, 2020).

4.1.2. Regresyon

Makine öğrenmesi uygulamalarında, sınıflandırma ve kümeleme yöntemlerinden farklı olarak kullanılan yöntemlerden biriside regresyondur. Regresyon, bir ya da daha fazla bağımsız değişken ile hedef değişken arasındaki ilişkiyi matematiksel olarak modelleyen bir yöntemdir. Literatürde kullanılan regresyon modellerinden bazıları aşağıda sunulmuştur.

4.1.2.1. Doğrusal Regresyon

Denetimli regresyon modellerinden biri olan doğrusal regresyon modeli, doğrusal ve sürekli değişkenler için kullanılan bir yöntemdir. Doğrusal regresyon, ilk olarak iki veya daha çok değişken arasındaki ilişkiyi, doğrusal bir fonksiyon ile göstermek için kullanılmaktadır. Bu fonksiyondaki değişkenlerin bir tanesi bağımlı, en az bir tanesi de bağımsız değişken olarak kullanılmaktadır. Doğrusal regresyonda değişkenlerden en az bir tanesi bilindiğinden dolayı, diğer değişkenler hakkında tahminler yapılabilir. Literatürde, tek bağımsız değişkeni olan regresyon modeline, basit doğrusal regresyon modeli denilmektedir. Eğer birden çok bağımsız değişkeni bulunuyorsa buna da çoklu doğrusal regresyon modeli denilmektedir.

Basit doğrusal regresyon modelinde, tek bir bağımsız değişken ile bağımlı değişken arasındaki doğrusal (lineer) ilişki Denklem 1 ile ifade edilmektedir. Bu denklemde bağımsız değişken yardımıyla, bağımlı değişken tahmin edilmektedir.

$$Y = \beta_0 + \beta_1 X + \in \quad (1)$$

Denklem 1'de X değeri bağımsız değişkeni, Y değeri X bağımsız değişkenine bağlı değişkeni, β_0 değeri doğrunun kesim noktasını, β_1 değeri doğrunun eğimini, $\in$ değeri de hatayı ifade etmektedir.

Birden fazla bağımsız değişkenin kullanıldığı çoklu doğrusal regresyon modelinde, bağımsız değişkenler ile bağımlı değişken arasındaki ilişki Denklem 2 ile ifade edilmektedir.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \in_i \quad (2)$$

Denklem 2'de kullanılan $\in_i$ *i.* tahmin için hata terimini ifade etmektedir. *i.* hata terimi de Denklem 3 ile hesaplanmaktadır.

$$\in_i = Y - \widehat{Y_i} \quad (3)$$

Denklem 3'deki Y gerçek değeri ve $\widehat{Y_i}$ ise *i.* tahmin değerini ifade etmektedir.

4.1.2.2. Temel Bileşen Regresyonu

Temel bileşen regresyonu, temel bileşen analizine dayalı regresyon modellerinden bir tanesidir. Temel bileşenler regresyonunun kullanımı, bağımsız değişkenlerin sayısının gözlem sayısından daha çok olduğu ve bağımsız değişkenlerin arasında da yüksek ilişki olan durumlarda tercih edilen bir yöntemdir (Derman Dildar, 2019). Gözlem sayısının çok olduğu ve bağımsız değişkenlerin arasında yüksek ilişkinin bulunduğu durumlarda, temel bileşenler regresyonu açıklayıcı değişkenlerin sayısını azaltabilir ve regresyon katsayıları tahminini yüksek

doğrulukta sağlayabilir (Rencher, 2003). Çoklu bağlantı sorununun yaşandığı durumlarda, temel bileşenler regresyonunu oluşturan her doğrusal modelin bir dik açıklayıcı değişkenler kümesi ile gösterilmesini sağlar (Özkan, 2009).

Temel bileşenler regresyonunun amacı, bir arada bulunan çok sayıdaki değişkenin, boyutlarını azaltarak daha az sayıda değişken ile gösterilmesini sağlamaktır. Bu azaltma işlemi ile aralarında herhangi bir korelasyon durumu olmayan bağımsız değişkenler elde edilir (Alma & Vupa, 2008). Temel bileşenler regresyon analizinde boyut azaltma yöntemi ile elde edilebilecek en fazla değişken sayısı, orjinal değişken sayısı kadar olmak zorundadır. Temel bileşenler regresyonunun genel amacı, değişkenler arasındaki en yüksek varyansı açıklayabilecek yeni değişkenler üretmektir (Karakaya, 2011).

4.1.2.3. Rastgele Orman

Rastgele orman algoritması hem sınıflandırma hem de regresyon için kullanılan denetimli bir öğrenme algoritmasıdır. Literatürde bulunan torbalama yönteminin geliştirilmiş bir versiyonudur (Breiman, 1996). Rastgele orman algoritmasının doğruluğu, makine öğrenimi algoritmalarındaki en yüksek algoritmalardan biri olarak kabul edilir (Breiman & Cutler, n.d.). Algoritmanın ana mantığı, bir ormanın karar ağaçlarından oluşturulması ve bu ormanda ağaç sayısı ne kadar fazla olursa, o kadar sağlam bir orman oluşturulmasına dayanmaktadır. Bu algoritmada, veri örnekleri üzerinde karar ağaçları oluşturulur ve daha sonra her birinden tahmin değerleri alınır. Sonuç olarak da elde edilen en iyi tahmin değeri, çözüm olarak seçilir. Böylece, rastgele orman

algoritması ile tek bir karar ağacına göre çok daha iyi sonuçlar elde edilir.

Rastgele orman algoritmasında kullanıcıya her düğümde kullanılacak ağaç sayısı ve değişken sayısı sorulur. Her ağaç, eğitim verilerinden rastgele seçilen değişkenler tarafından oluşturulur ve algoritmanın "rastgele" olarak adlandırılması da buradan gelmektedir.

Rastgele orman algoritması ile N adet ağacın en altında bulunan yapraklardan alınan değerlerden sonuçlar elde edilir ve verilerin değişkenlerinin ağırlıkları alınarak sonuç hesaplanır. Rastgele orman regresyon makine öğrenme türünde her ağaçtan ortalama sonuçlar alınarak sonuç değeri hesaplanır. Rastgele orman modeli daha sonra verilerin geri kalanı için de sonuçları tekrar hesaplamak için kullanılır.

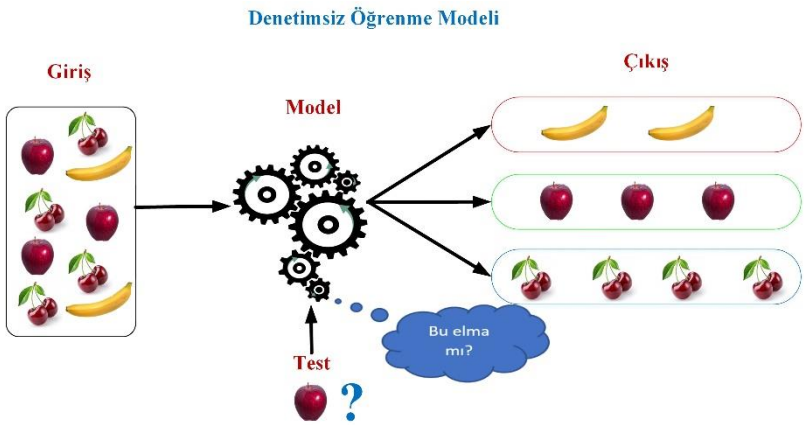
4.1.2.4. Yapay Sinir Ağları (YSA)

Yapay Sinir Ağları, makine öğrenmesinin popüler alt konularının birisidir. Multidisipliner olması nedeniyle birçok uygulamada tercih edilmektedir (Baş, 2006; Elmas, 2003; Haton & Haton, 1989). Bu kadar çok tercih edilmesinin sonucunda da yeni YSA modelleri geliştirilmektedir. YSA'lar genel mantık olarak insanların algılama ve karar verme gibi düşünsel süreçlerini taklit eden bir yapıya sahiptir. Biyolojik sinir hücrelerinden esinlenerek geliştirilen YSA modelleri birbirine bağlı hücrelerden oluşmaktadır. Bu hücreler kendilerine gelen bilgiyi çözümleyebilen ve kendinden sonrasına aktarabilen bir yapıya sahiptir. YSA'ya herhangi bir konu ile ilgili yeterli sayıda giriş ve çıkış bilgileri verildiğinde, YSA kendi iç yapısında giriş ile çıkış verileri arasında bir bağıntı kurabilir. Ağa bu şekilde giriş ve çıkış bilgilerinin

verilmesi işlemine ağın eğitilmesi denilmektedir. Ağın eğitilmesinden sonra, eğitildiği konu ile ilgili daha önce görmediği bir örnek verilirse, ağ bazı çıkarımlarda bulunabilir. YSA'lar bu çıkarım yeteneği sayesinde geniş kullanım alanı bulmuştur (Haton & Haton, 1989; Metlek & Özkan, 2019; Pirim, 2006). YSA yapısında bulunan giriş katmanında, algılayıcılardan ya da veri tabanından gelen bilgiler doğrudan ağa aktarılmaktadır. Ara katmanlarda giriş katmanında gelen bilgiler ağırlık katsayıları ile çarpılıp aktivasyon fonksiyonlarından geçirilerek çıkış katmanına gönderilmektedir. Çıkışta elde edilmesi hedeflenen değer ile YSA'nın üretmiş olduğu değer arasındaki fark alınarak hesaplanan hata değeri, uygun fonksiyonlar ile ağda kullanılan ağırlık katsayılarına yansıtılmaktadır. Bu ileri-geri hesaplama işlemleri istenilen hata değerine yada belirlenen iterasyon sayısına ulaşıncaya kadar tekrar edilmektedir (Mengi & Metlek, 2020).

4.2. Denetimsiz Öğrenme

Denetimsiz öğrenme algoritmalarının genel mantığı etiketsiz veriler için kullanılmasıdır. Bu model mevcut olan verilerden, veriler arasındaki ilişkiyi keşfederek, verileri kendi içerisinde gruplamaktadır. Denetimsiz öğrenme algoritmaları, Şekil 4'de gösterildiği üzere giriş verilerini benzer bölümlerine, yapılarına, yoğunluklarına ve diğer benzer özelliklere göre gruplamaktadır.



Şekil 4: Denetimsiz Öğrenme

Denetimsiz öğrenme algoritmaları Hebbian Öğrenme kuralına dayanmaktadır. Bu kurala göre, birbirine bağlı iki nörondan her ikisi de aktif ise aralarındaki bağ artmakta, aktif değil ise azalmaktadır. Denetimsiz öğrenmenin temel kuralları;

- Kullanılan veri setindeki örnekler bilinir fakat bu örneklere karşılık gelen küme değerleri bilinmez. Bu nedenle büyük veri uygulamalarında sıklıkla tercih edilir.
- Denetimsiz öğrenme modellerindeki ana gaye veri setinde bulunan verilerden bir model veya örüntü (pattern) çıkarmaktır.
- Denetimsiz öğrenme algoritmaları ile veri setinin yapısını belirlemek üzere, verilerin sınıflara ayırılmasına kümeleme (clustering), ortaya çıkan sınıflara da küme denir.

Denetimsiz öğrenmede yaygın olarak kullanılan yöntemler genelde kümeleme yöntemleridir.

4.2.1. Kümeleme

Veri setinde bulunan benzer verileri bir araya getirme işlemine kümeleme denilmektedir. Kümeleme analizinin temel amacı, verilerin sahip oldukları ortak özellikleri temel alarak gruplamaktır. Bazı uygulamalarda, verilerin aynı anda birden çok özelliği kullanılarak kümeleme işlemi de gerçekleştirilmektedir. Literatürde kullanılan, bazı kümeleme yöntemleri aşağıda açıklanmıştır.

4.2.1.1. K-Ortalamalar

K-ortalamalar algoritması bir denetimsiz öğrenme ve kümeleme algoritmasıdır. K-ortalamalar algoritmasındaki K değeri küme sayısını ifade eder ve bu K değerini parametre olarak kullanıcının belirlemesi gerekmektedir. Bu durum bir dezavantaj gibi görülebilir. K değerini otomatik olarak hesaplayan X-Means adında başka bir kümeleme algoritması da literatürde bulunmaktadır (Hamerly & Elkan, 2004).

Kümeleme işlemi, K-ortalamalar algoritmasında genel olarak dört adımda gerçekleştirilmektedir. Bunar;

1. Her kümenin merkez noktasını veya ortalamasını temsil etmek üzere K adet veri rastgele seçilir.
2. Merkez dışındaki verilerin mesafelerine göre kümelendirilmesi yapılır.
3. Yapılan kümelendirmeye göre yeni merkezlerin belirlenmesi (veya eski merkezlerin yeni merkeze kaydırılması) sağlanır.
4. Sistem kararlı hale (stable state) gelinceye kadar 2. ve 3. adımlar tekrarlanır.

4.2.1.2. Yapay Sinir Ağları

Denetimsiz öğrenme modellerinden birisi olan YSA Modeli genel olarak kitabın 4.1.2.4 numaralı başlığı altında açıklanmıştır.

4.2.1.3. Hiyerarşik Kümeleme

Denetimsiz öğrenme modellerinden biriside Hiyerarşik Kümeleme yöntemidir. Bu yöntem diğer kümeleme yöntemlerine göre daha bilgilendirici bir sonuç üretir. K-ortalamalar algoritmasında küme sayısının önceden algoritmaya girilmesi gerekirken, Hiyerarşik kümeleme algoritmasında küme sayısının önceden belirlenmesine gerek yoktur. Hiyerarşik kümeleme algoritmasının parçadan bütüne (Agglomerative) ve bütünden parçaya (Divisive) olarak iki türü bulunmaktadır (Manning, Raghavan, & Schutze, n.d.).

a) Parçadan Bütüne (Agglomerative)

Parçadan Bütüne hiyerarşik kümelemede başlangıçta her bir değer bağımsız bir küme olarak değerlendirilir ve bu değerler çeşitli algoritmalarla birleştirilip, her aşamada bir üst küme oluşturulur. Aşağıda bu algoritmanın adımları verilmiştir.

1. İlk olarak her bir veri, bir küme haline getirilir yani N tane eleman varsa N adet küme oluşur.
2. Daha sonra birbirine mesafe olarak yakın olan kümeler birleşerek yeni bir küme oluşturur.
3. Sistem kararlı oluncaya kadar 2. Adımı tekrar edilir.

b) Bütünden parçaya (Divisive)

Bütünden parçaya algoritması, parçadan bütüne algoritmasının tam tersidir. İlk başta tüm verilerden tek bir küme oluşturulur. Daha sonra bu küme çeşitli ayrıştırma algoritmalarıyla parçalanarak yeni alt kümeler elde edilir (Johnson, 1967). İşlem sonunda dendrogram denen tüm verilerin ilişkilerini gösteren bir ağaç yapısı oluşturulur. Aşağıda bu algoritmanın adımları verilmiştir.

1. Her veri için bir değer oluşturulur ve her veri bir küme olarak atanır (Bu durumda N tane veri için N adet küme oluşur). Böylece N x N boyutunda bir matris oluşturulur.
2. En yakın olan iki veri gruplanır. Belirlenen uzaklık fonksiyonu ve matrisi kullanılarak veriler arasındaki uzaklıklar hesaplanır.
3. N=N-1 boyutunda yeni bir matris oluşturulur. N değeri 1 oluncaya kadar 2. adıma geri dönülür.

4.2.1.3. Temel Bileşenler Analizi

Günümüzde büyük veri kümeleri giderek yaygınlaşmakta ve bunlar üzerinden işlemler yapılmaktadır. Veri kümeleri büyüdükçe de anlamlı veri çıkarmak zorlaşmaktadır. Temel Bileşenler Analizi (Principle Components Analysis, PCA), bu tür veri kümelerinin boyutluluğunu azaltmak, yorumlanabilirliği artırmak için kullanılan denetimsiz bir kümeleme yöntemidir. Veri boyutunun azaltıldığı durumlarda bazı veri kayıpları da yaşanmaktadır (Alkan, 2008).

Temel Bileşenler Analizi'nde p sayıda başlangıç değişkenine karşılık elde edilen p sayıda temel bileşenin her biri, orijinal değişkenlerin

doğrusal bir bileşimidir. Dolayısıyla, her bir temel bileşen bünyesinde tüm değişkenlerden belirli oranda bilgiyi barındırır. Bu özelliği sayesinde Temel Bileşenler Analizi, p boyutlu veri kümesi yerine, ilk m önemli temel bileşenin kullanılması yoluyla boyut indirgemesi sağlayabilmektedir. İlk m temel bileşen, toplam varyansın büyük kısmını açıklıyorsa, geriye kalan p-m temel bileşen ihmal edilebilir. Klasik değişken seçimi teknikleri ile karşılaştırıldığında, temel bileşen analizinde önemli bilgi kaybı oldukça azdır. Temel Bileşenler Analizi kullanılırken dikkat edilmesi gereken bazı hususlar vardır. Bunlar;

- Farklı ölçeklerdeki veriler yanıltıcı bileşenlere sebep olabileceği için uygulamaya başlamadan önce verilere standardizasyon işlemi uygulanmalıdır.
- Analiz aykırı gözlemlerden (outlier) fazlaca etkilenir. Bu nedenle analizden önce veriler aykırı gözlemlerden ayrılmalıdır. Bu işlem Randomized PCA, Sparse PCA gibi alternatif yöntemler ile de gerçekleştirilebilir.

Temel Bileşenler Analizi yapılırken sırasıyla aşağıdaki adımlar takip edilmelidir (Muratlar, 2019).

1. Her boyut için ortalama vektör hesaplanır.
2. Kovaryans matrisleri hesaplanır.
3. Her boyut için özvektörler ve karşılık gelen özdeğerleri hesaplanır.
4. Her bir özdeğerin, özdeğerler toplamına bölünmesi ile temel bileşenlerin toplam varyansı açıklama yüzdeleri elde edilir.

BÖLÜM 2 - DESTEK VEKTÖR MAKİNELERİ

Kitabın bu bölümünde Destek Vektör Makinelerinin teorik altyapısı açıklayıcı şekillerle ve ilgili formüller ile kapsamlı olarak anlatılmıştır. Uygulamalara geçilmeden önce, konuya hakimiyet sağlanması açısından bu bölümün detaylı olarak incelenmesi faydalı olacaktır.

1. Destek Vektör Makineleri

Destek vektör makineleri literatürde, hem sınıflandırma hem de regresyon analizinde kullanılmasına rağmen genel olarak sınıflandırma işlemlerinde daha çok tercih edilen bir makine öğrenme yöntemidir. Bu yöntem denetimli bir öğrenme modeline dayanmaktadır. Algoritmanın çalışması esnasında verilerin türüne bağlı olarak çekirdek fonksiyonlarda kullanılabilmektedir. Bu sayede hem doğrusal hem de doğrusal olmayan sınıflandırma işlemlerini gerçekleştirilebilmektedir. Eğer sınıflandırma işleminde, tam ayrıştırılabilir veriler kullanılırsa genellikle tüm veriler bir hiper düzlem ile sınıflandırılabilmektedir. Fakat, eğer tam ayrıştırılamayan veriler kullanılırsa, çoğunlukla aynı boyutta tek bir düzlem ile sınıflandırılamamaktadır. Bu nedenle de farklı çekirdek fonksiyonları kullanılmaktadır.

2. DVM'nin Kullanım Alanları

DVM'ler literatürde birçok örüntü uygulamasında kullanılmaktadır. Şekil 5'de gösterilen yüz algılama, metin ve köprü metni sınıflandırma, görüntü sınıflandırma, biyoinformatik, protein çaprazlama, uzaktan homoloji tespiti, el yazısı tanıma, jeoleji ve çevre bilimleri, genelleştirilmiş tahmine dayalı kontrol uygulamaları bunlardan sadece birkaçıdır (Gour, 2019).

Şekil 5: DVM Uygulama Alanları

2.1. Yüz Algılama

Görüntünün bölümleri yüz ve yüz olmayan bölümler olarak sınıflandırılır. NxN boyutundaki görüntüdeki her bir piksel değeri yüz ve yüz olmayan bölümler olarak iki farklı etiketle etiketlenir. Bu veriler eğitim verilerini oluşturur. Sonrasında piksel parlaklığına göre yüzlerin etrafında bir karesel sınır oluşturulur ve her bir görüntü için aynı işlem tekrar edilerek, DVM ile sınıflandırma işlemleri gerçekleştirilir.

2.2. Metin ve Köprü Metni Sınıflandırma

Metin ve köprü metinlerin sınıflandırılması DVM ile gerçekleştirilebilir. Bu işlem için ilk olarak metinleri; haber makaleleri, e-postalar ve web sayfaları gibi farklı kategorilerde sınıflandırmak için eğitim verileri kullanılır.

Örneğin:

- Haber makalelerinin "iş" ve "filmler" olarak sınıflandırılması,
- Web sayfalarının kişisel ana sayfalar ve diğerleri olarak sınıflandırılması.

Algoritmada sonraki adımda her belge için bir puan hesaplanır. Hesaplanan değer önceden tanımlanmış bir eşik değeriyle karşılaştırılır. Bir metnin puanı belirlenen eşik değerini aştığında, metin belirli bir kategoride sınıflandırılır. Eşik değerini geçmezse, genel bir metin olarak değerlendirir. Her metin için puan hesaplanarak ve öğrenilen eşikle karşılaştırılarak yeni örneklerin sınıflandırılması gerçekleştirilir.

2.3. Görüntülerin Sınıflandırılması

DVM'ler, görüntüleri daha yüksek arama doğruluğu ile sınıflandırabilir. Doğruluğu, geleneksel sorgu tabanlı ayrıntılandırma şemalarından daha yüksektir.

2.4. Biyoinformatik

Hesaplamalı biyoloji alanında, protein homoloji tespiti yaygın bir problemdir. Bu problemi çözmek için kullanılan en etkili yöntemlerden birisi de DVM yöntemidir. Bu yöntem, biyolojik diziler arasında tanımlama yapmak için yaygın olarak kullanılmaktadır. Örneğin; hastaların genlerine göre ve diğer birçok biyolojik problemlerine göre sınıflandırılması için kullanılır.

2.5. Protein Çaprazlama ve Uzaktan Homoloji Tespiti

Protein uzaktan homoloji tespiti, protein yapıları ve fonksiyonları çalışmalarında hayati bir rol oynar. Neredeyse esnek hesaplama algoritmalarının tamamı, protein dizilerini temsil etmek için sabit uzunluk özelliklerine ihtiyaç duyar. Bununla birlikte, sınırlı protein bilgisi ile ayırt edici özellikleri tespit etmek kolay bir iş değildir. Buna karşın literatürde bu işlemlerde DVM'ler oldukça başarılı sonuçlar vermektedir.

2.6. El Yazısı Tanıma

Literatürde belgeler üzerindeki imzaları ve elle yazılmış karakterleri tanımak için DVM yöntemleri kullanılmaktadır.

2.7. Jeoleji ve Çevre Bilimleri

DVM'ler coğrafi (mekansal) ve mekansal-zamansal çevresel veri analizi ve modelleme işlemleri içinde kullanılmaktadır. Özellikle uydu görüntüleri üzerinden haritalama uygulamalarında da kullanılmaktadır. Bu tarz uygulamalarda farklı çekirdek fonksiyonlarının başarısı üzerine de yapılmış akademik çalışmalar bulunmaktadır (Kavzoğlu & Çölkesen, 2010).

2.8. Genelleştirilmiş Tahmine Dayalı Kontrol (GTK)

Kaotik dinamikleri kullanışlı parametrelerle kontrol etmek için DVM tabanlı GTK kullanılır. Bu yöntem sistem kontrolünde çok iyi derecede performans sağlar. Sistem, hedefin yerel istikrarına göre kaotik dinamikleri takip eder.

Kaotik sistemleri kontrol etmek için DVM'lerin kullanılmasının avantajları:

- Kaotik bir sistemi hedefe yönlendirmek için nispeten küçük parametre algoritmalarının kullanılmasına izin verir.
- Kaotik sistemler için bekleme süresini azaltır.
- Sistemlerin performansını korur.

3. DVM Algoritmalarında Kullanılan Veri Yapılarının Önemi

Yukarıda konunun başında da belirtildiği gibi, DVM'lerde kullanılan verilerin birbiri içerisinden kolaylıkla ayrılıp ayrılmaması, verileri sınıflandırmak için kullanılan düzlemlerin nasıl ve kaç adet olacağını doğrudan etkilemektedir. Bu nedenle DVM'lerde tercih edilen yöntemler,

1. Tam ayrıştırılabilir veriler,
2. Tam ayrıştırılamayan veriler

için olmak üzere ikiye ayrılmaktadır.

3.1. Tam Ayrıştırılabilir Veri

Genellikle destek vektör makinelerinde, tam ayrıştırılabilir veriler için Denklem 4'de ki gibi etiketli veri tipleri kullanılır;

$$(x_i, y_i),\ i=1,2,\ldots,n \qquad \text{ve} \qquad x_i \in R^+, \quad y_i \in \{-1,+1\} \tag{4}$$

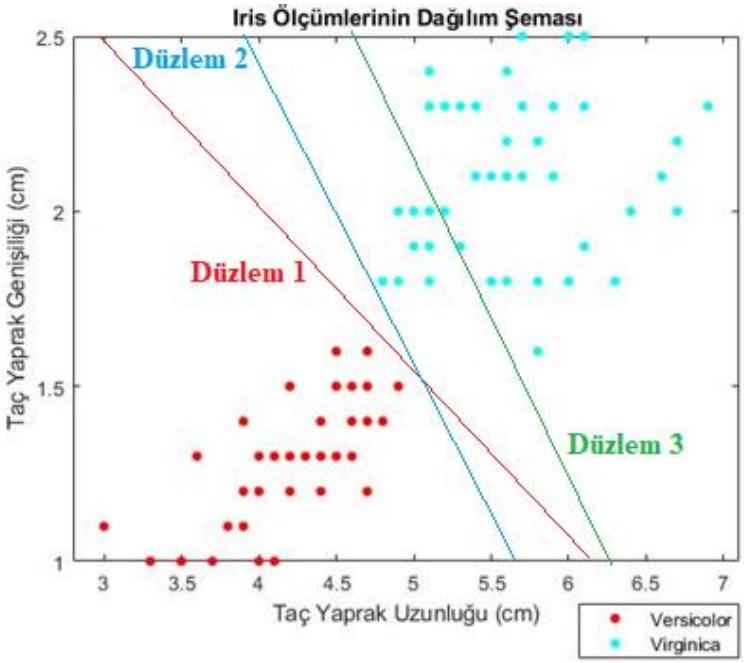
Denklem 4'de kullanılan y_i ifadesi verinin etiketini ifade etmektedir. DVM'ler, Denklem 4'de gösterildiği gibi ikili ayrıştırma yöntemi olarak da düşünülebilir. Bu ikili ayrıştırma işlemi etiket çeşidinin ikiden fazla olma durumunda da benzer işlemler yapılarak uygulanabilmektedir. Konunun daha temel düzeyde anlaşılması için ilk etapta ikili ayrıştırma üzerinde durulmaktadır. İkili sınıfları ayrıştırmanın temel prensibi; **bir düzlem yardımı ile etiketli veri noktalarını iki sınıfa ayırmaktır.**

Örneğin elimizde kırmızılar ve siyahlar şeklinde iki farklı etiketli veri varsa, bu etiketli verileri birbirinden ayırmak için bir düzlem kullanılır.

Bu düzlemin bir tarafında kırmızılar diğer tarafında siyah bulunacak şekilde sınıflandırma işlemi yapılır. Etiketli verilere örnek verecek olursak;

- Sağlıklı bireyler ile hasta bireyler,
- Zenginler ile fakirler,
- Bitkiler ile hayvanlar, vb. etiketli verilerdir.

Örneğin Şekil 6'da Iris veri tabanında bulunan Versicolor ve Virginica yaprak çeşitlerinin nasıl sınıflandırıldığı gösterilmektedir. Şekil 6'da, x ekseni yaprakların uzunluğunu, y ekseni yaprakların genişliğini temsil etmektedir.



Şekil 6: İki Farklı Sınıfın Düzlemler ile Ayrılması

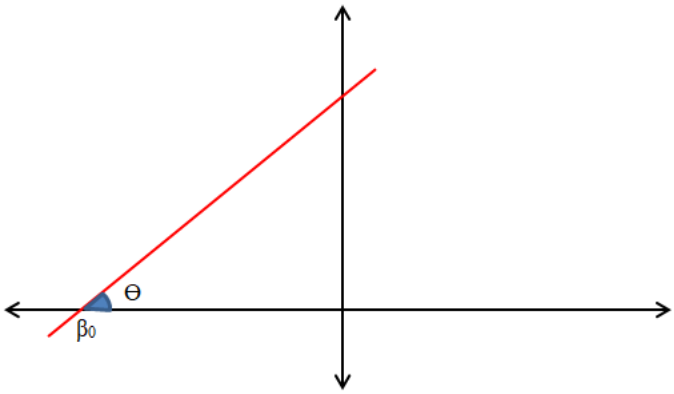
Bu iki eksenin kesişimi ise yaprağın sınıf etiketini oluşturmaktadır. Şekil 6'da bulunan düzlemler ile bu iki özniteliğe göre, iki farklı yaprak türünün sınıflandırılması gösterilmektedir. Sınıflandırma işleminde ikiden fazla öznitelik kullanılması durumunda, sınıfların ayrışımını iki boyutlu bir grafikle göstermek oldukça zordur.

Şekil 6'da gösterilen iki farklı sınıfı ayırmak için üç adet düzlem kullanılmıştır. Şekilden de anlaşılacağı üzere Düzlem 1 ve Düzlem 2 iki farklı sınıfı birbirinden ayırabilirken, Düzlem 3 iki farklı sınıfı birbirinden ayıramamaktadır.

Şekil 6'da bahsedilen düzlemlerin matematiksel ifadesi bir doğru denklemi şeklinde, Denklem 5 ile gösterilmektedir.

$$\{X \in R^{+}: \beta^{T}X + \beta_0 = 0, \qquad \beta \in R^{+}, \beta_0 \in R\} \tag{5}$$

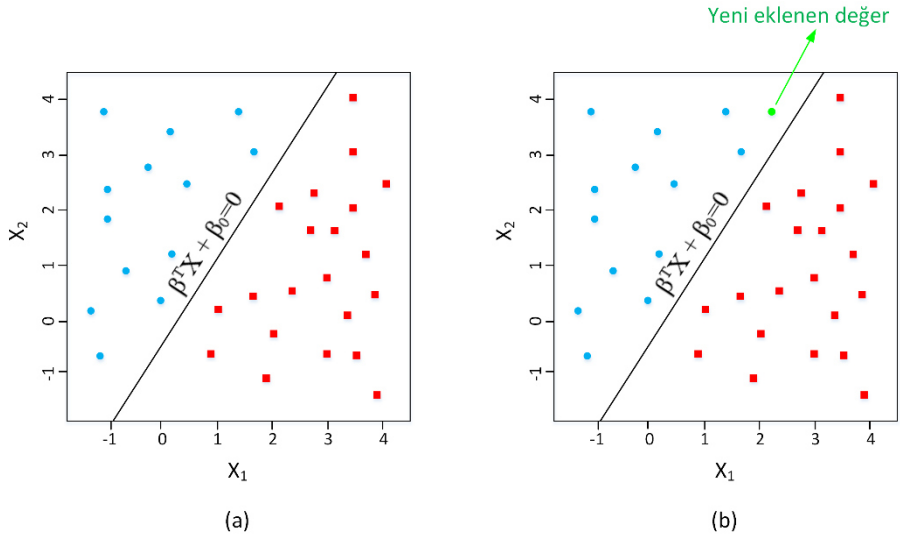
Şekil 7'de, Denklem 5'deki doğrunun, düzlemi kestiği nokta β_0 ile ifade edilmektedir. Bu doğrunun eğimi Denklem 6'da gösterildiği gibi β^{T} ile ifade edilmektedir.



Şekil 7: $\beta^{T}X + \beta_0$ Doğrusu

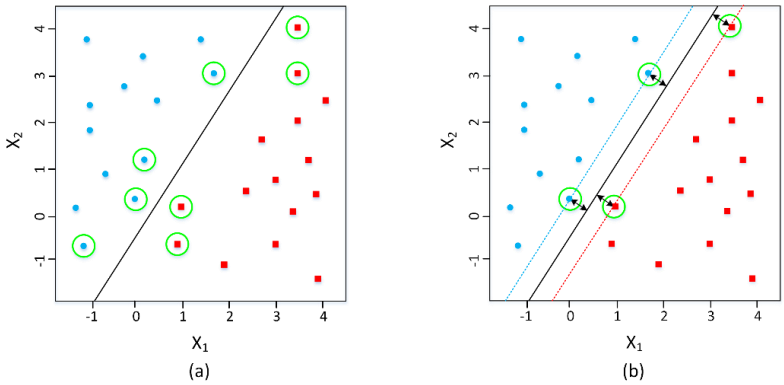
$$Eğim = \frac{\sin\theta}{\cos\theta} = \frac{\sin\theta}{\beta_0} = \beta^T \tag{6}$$

Şekil 7'de düzlem olarak ifade edilen doğrunun iki farklı sınıfı birbirinden ayırması Şekil 8 (a)'da gösterilmiştir. Eğer sisteme Şekil 8 (b)'de gösterildiği gibi yeni bir değer eklenirse, doğrunun (düzlemin) ne tarafında kaldığına bakılarak sınıflandırma işlemi yapılır.



Şekil 8: (a) $\beta^T X + \beta_0$ Doğrusunun İki Sınıfı Birbirinden Ayırması, (b) Yeni Eklenen Değer

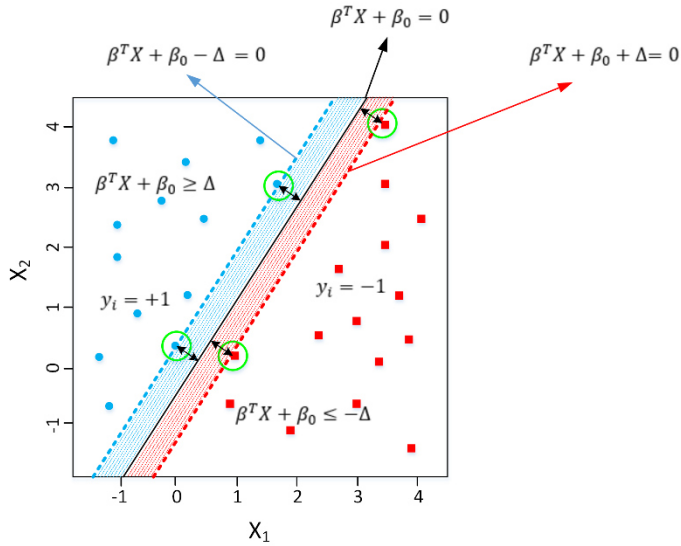
Bu işlem için x_1 ve x_2 değerleri doğru (düzlem) denkleminde (Denklem 5) yerine konulur. Çıkan sonuç 0'dan büyük olması durumunda yeni eklenen değer düzlemin sol tarafında, 0'dan küçük olması durumunda ise düzlemin sağ tarafındaki sınıfa dahil edilir.



Şekil 9: $\beta^T X + \beta_0 = 0$ Doğrusunun Konumu

Şekil 9 (a)'da gösterilen $\beta^T X + \beta_0 = 0$ doğrusunun konumunu kendisine en yakın olan farklı sınıflardaki noktalar belirlemektedir. Örneğin, Şekil 9 (a)'da gösterildiği gibi bu denkleme farklı sınıflardan yakın birçok nokta bulunmaktadır. Fakat doğrunun konumunu Şekil 9 (b)'de gösterilen en yakın noktalar belirlemektedir. Farklı sınıflarda bulunan, $\beta^T X + \beta_0 = 0$ doğrusuna en yakın noktaların üzerinden geçen ve aynı zamanda $\beta^T X + \beta_0 = 0$ doğrusuna paralel olan başka doğrularda bulunmaktadır. Bu doğrular Şekil 9 (b)'de kırmızı ve mavi noktalar ile gösterilen doğrulardır ve bu doğrulara da destek vektörü denilmektedir.

$\beta^T X + \beta_0 = 0$ doğrusuna en yakın olan farklı sınıflardan geçen başka doğrular da bulunmaktadır.



Şekil 10: Tam Ayrıştırılabilir Veri

Şekil 10'da da belirtildiği üzere sistemde kullanılan $y_i \epsilon \{-1,+1\}$ şeklinde iki farklı etiket bilgisi bulunmaktadır. $\beta^T X + \beta_0 = 0$ doğrusuna en yakın noktalardan geçen destek vektörleri Δ kadar yakından geçmektedir. Bu nedenle $\beta^T X + \beta_0 = 0$ doğrusuna bu değer eklenerek ve çıkarılarak destek vektör doğruları bulunabilmektedir. Bu durum Denklem 7, 8 ve 9'da gösterilmektedir.

$$\beta^T X + \beta_0 + \Delta = 0 \qquad \text{Üst destek vektör doğru denklemi} \qquad (7)$$

$$\beta^T X + \beta_0 = 0 \qquad \text{Düzlem doğru denklemi} \qquad (8)$$

$$\beta^T X + \beta_0 - \Delta = 0 \qquad \text{Alt destek vektör doğru denklemi} \qquad (9)$$

Bu işlemlerden sonra ana amacımız bu destek vektörlerini kullanarak sınıfları ayırmak için kullanılacak bir düzlem bulmaktır. Bu düzlem ile

destek vektörleri arasındaki mesafe en fazla olacak şekilde ayarlanmalıdır.

Amaç: β ve Δ parametrelerini,

$$\beta^T X + \beta_0 - \Delta = 0 \quad \text{ve} \quad \beta^T X + \beta_0 + \Delta = 0 \tag{10}$$

ile gösterilen iki düzlem arasındaki mesafe en fazla olacak şekilde seçmek.

Bu istenilen düzlemi bulabilmek için β^T ve β_0 değişkenlerini bulmak gerekmektedir. Bir Δ değeri için bu denklemi yazmak gerekirse (Bu Δ değerini de destek vektörleri belirleyecektir. Bu Δ değerine tüm denklemi böleceğiz).

Çalışmada hesaplama kolaylığı olması açısından $\Delta = 1$ olarak alınmıştır. Bu durumda aşağıdaki denklem elde edilir.

Eşitliklerin her iki tarafı Δ değerine bölünüp, $\Delta = 1$ olarak alınabilir.

$$i = 1,2,\ldots,n \;\; için \;\; (x_i, y_i) \begin{cases} y_i = +1 \rightarrow \beta^T x_i + \beta_0 - 1 \geq 0 \\ y_i = -1 \rightarrow \beta^T x_i + \beta_0 + 1 \leq 0 \end{cases} \tag{11}$$

Denklem 11'de x_i ile ifade edilen değerler Şekil 10'da gösterilen x_1 ve x_2 özniteliklerdir. i değeri ise elimizdeki veri vektörünün eleman sayısını ifade etmektedir. y_i ile ifade edilen değer ise {-1,+1} şeklindeki sınıf bilgileridir. Yukarıdaki denklemde, β^T ve β_0 değerleri yerine

konulduğundan elde edilen değer sıfırdan büyükse +1 sınıfına, değilse -1 sınıfına yerleştirilmektedir.

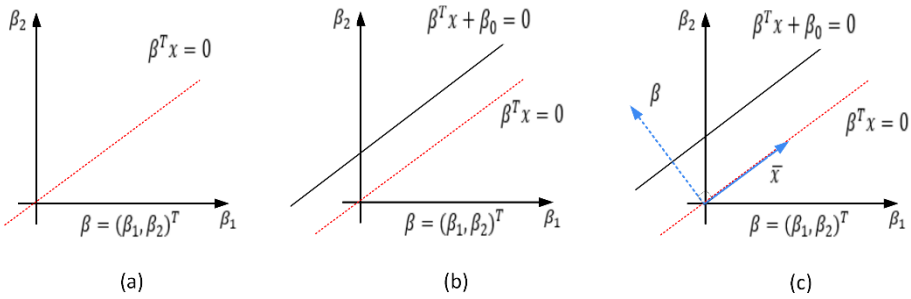
Denklem 11 tekrar düzenlenecek olursa, yeni eşitlik Denklem 12'deki gibi olur.

$$y_i(\beta^T x_i + \beta_0) \geq 1 \tag{12}$$

Denklem 12'ye dikkat edilecek olursa y_i'nin durumuna göre çıkış da değişecektir. Böylece Denklem 12, Denklem 10 ve 11'in yerine kullanılabilecektir.

Düzlemler Arasındaki Mesafe

Destek vektör makinelerinde genel amaç, iki düzlem arasındaki mesafeyi en uzak tutmaktır. Bunun için ilk olarak düzlemler arasındaki mesafenin hesaplanması gerekmektedir. Bunun için $\beta^T x = 0$ doğrusu Şekil 11(a)'da gösterilmiştir.



Şekil 11: Düzlemler Arasındaki Mesafe

Bu doğruya 0 noktasından geçtiği için "Alt Uzay" denir. Bu denkleme β_0 değeri eklenirse Şekil 11(b) elde edilir. Şekil 11(b)'den de anlaşılacağı üzere β_0 noktası doğrunun düzlemi kestiği noktayı

göstermektedir. $\beta^T x = 0$ doğrusu üzerinde Şekil 11(c)'de gösterildiği üzere bir nokta alınırsa; bu noktanın bulunduğu $\beta^T x = 0$ doğrusuna (düzlemine) dik olan bir vektör çizildiğinde, Şekil 11(c)'de de görüleceği üzere bu vektör $\beta^T x = 0$ doğrusundaki β'dır. Bu β'ya normal doğrusu da denilmektedir. Elde edilen bu β vektörü (normal doğru) $\beta^T x = 0$ doğrusuna dik olduğu gibi $\beta^T x_i + \beta_0 = 1$ ve $\beta^T x_i + \beta_0 = -1$ doğrularına(düzlemine) diktir. Bu durum aşağıda özetlenmiştir.

β: alt uzaya dik doğrultu

$$\beta^T x = 0 \rightarrow \beta^T x_i = 0$$

Ayrıca β vektörü

$$\beta^T x + \beta_0 = 1 \rightarrow \beta^T x_i + \beta_0 = 1$$

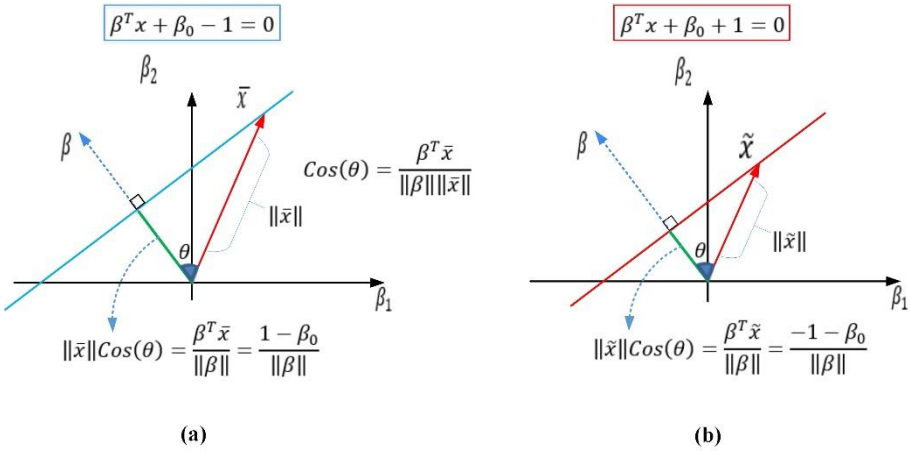
ve

$$\beta^T x + \beta_0 = -1 \rightarrow \beta^T x_i + \beta_0 = -1$$

düzlemlerine de dik.

Eğer $\beta^T x_i + \beta_0 - 1 = 0$ noktasını yukarıya kaydırırsak ve bunun üzerinde Şekil 12(a)'da gösterildiği gibi bir $\bar{x}$ noktası alırsak, daha önce elde ettiğimiz ve bahsettiğimiz β vektörü (normal doğru) bu doğruya da diktir. Sıfır noktasından, $\beta^T x_i + \beta_0 - 1 = 0$ doğrusu üzerinde

bulunan $\bar{x}$ noktasına Şekil 12 (a)'da ki gibi bir doğru çizilir. Bu doğru ile β vektörü (normal doğru) arasındaki açıya θ dersek bu açının Cosinus'ü Denklem 13'de gösterildiği gibi hesaplanır.



Şekil 12: $\beta^T x_i + \beta_0 - 1 = 0$ *ve* $\beta^T x_i + \beta_0 + 1 = 0$ Doğruları

$$Cos(\theta) = \frac{\beta^T \bar{x}}{\|\beta\|\|\bar{x}\|} \tag{13}$$

Denklem 13'de $Cos(\theta)$ hesaplanırken β^T ve $\bar{x}$ nin vektörel çarpımları yapılarak, bu vektörlerin büyüklüklerinin çarpımına bölünür. Bu işlemden sonra Şekil 12 (a)'da koyu yeşil renkle gösterilen mesafe Denklem 14 ile hesaplanır.

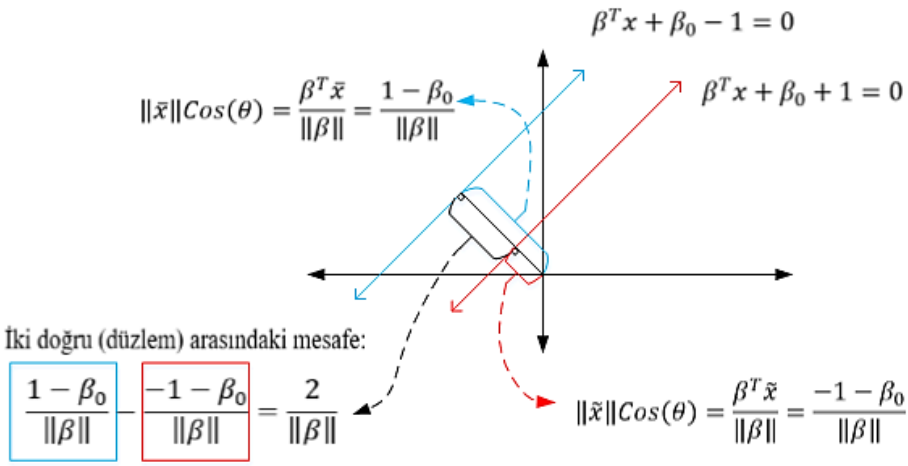
$$\|\bar{x}\|Cos(\theta) = \frac{\beta^T \bar{x}}{\|\beta\|} = \frac{1-\beta_0}{\|\beta\|} \tag{14}$$

Denklem 14'deki $\beta^T\bar{x}$ ifadesinin yerine, Şekil 11'de gösterilen x vektörü yerine bu vektörde bulunan x_i vektörel değeri kullanılarak $\beta^T x_i + \beta_0 = 1$ denkleminden elde edilen $1 - \beta_0$ ifadesi kullanılmıştır.

Aynı işlemler Şekil 12 (b)'de gösterilen $\beta^T x + \beta_0 + 1 = 0$ doğrusu üzerindeki $\tilde{x}$ noktası içinde yapılır. Bu işlemden sonra Şekil 12 (b)'de koyu yeşil renkle gösterilen mesafe Denklem 15 ile hesaplanır.

$$\|\tilde{x}\| Cos(\theta) = \frac{\beta^T \tilde{x}}{\|\beta\|} = \frac{-1 - \beta_0}{\|\beta\|} \tag{15}$$

Her iki doğru (düzlem) aynı anda Şekil 13'de gösterilmektedir. Bu şekilden de anlaşılacağı üzere her iki doğru arasındaki mesafe Denklem 16 ile hesaplanabilmektedir.

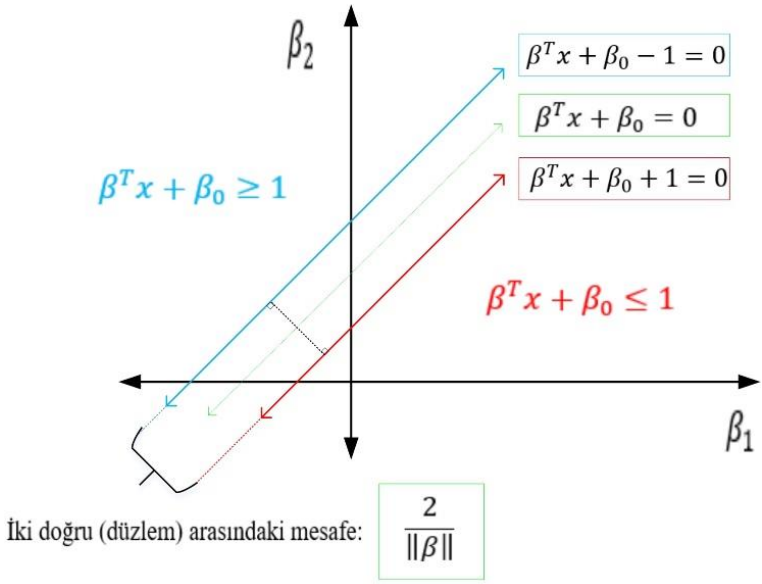


Şekil 13: İki Doğru Arasındaki Fark

İki doğru (düzlem) arasındaki mesafe:

$$\frac{1-\beta_0}{\|\beta\|} - \frac{-1-\beta_0}{\|\beta\|} = \frac{2}{\|\beta\|} \tag{16}$$

Sonuç itibari ile Şekil 13'de gösterilen iki düzlem arasındaki mesafe β katsayısı ile ters orantılı olarak elde edilmiştir. Bu durum Şekil 14'de gösterilmektedir.



Şekil 14: İki Doğru Arasındaki Mesafenin β ile İlişkisi

Şekil 14 elde edildikten sonra ki amaç iki doğru (düzlem) arasındaki mesafeyi maksimum yapmaktır. Bunun için optimizasyon aşamasına geçmek gerekmektedir. Bu işlemi yapmak için aşağıda tekrar gösterilen Denklem 10'u hatırlamakta fayda vardır. Burada gösterilen şartında sağlanması gerekmektedir.

Amaç: β ve Δ parametrelerini,

$$\beta^T X + \beta_0 - \Delta = 0 \quad \text{ve} \quad \beta^T X + \beta_0 + \Delta = 0 \tag{10}$$

ile gösterilen iki düzlem arasındaki mesafe en fazla olacak şekilde seçmek.

β katsayısının en yüksek olması ve düzlem üzerindeki tüm noktaların Denklem 17'de gösterilen şartları sağlaması gerekmektedir.

$$\begin{gathered} en\ büyükleme: \frac{2}{\|\beta\|} \\ ve \\ öyleki:\ y_i(\beta^T x_i + \beta_0) \geq 1,\ \ i = 1, \ldots, n \end{gathered} \tag{17}$$

Denklem 17'deki β ve β_0 vektör değerlerini bulmamız gerekmektedir. Burada β değeri öyle bir değer seçilmelidir ki, $\frac{2}{\|\beta\|}$ değeri en büyük olmalı yani iki düzlem arasındaki mesafe en uzun olmalıdır. Aynı zamanda seçilen β ve β_0 değerleri, Denklem 17'de yerine konulduğunda tüm x_i ve y_i noktaları da belirtilen şartı sağlamalıdır.

Optimizasyon işlemlerinde araştırmacılar genellikle en küçükleme işlemleri üzerine çalışmaktadırlar. Bu nedenle Denklem 17'deki en büyükleme işlemindeki, $\frac{2}{\|\beta\|}$ değerini en küçükleme optimizasyonu yapmak için ve paydaya gelen vektör uzunluğunu ($\|\beta\|$) paya geçirmek için Denklem 18 kullanılmaktadır.

$$en\ küçükleme: \frac{1}{2}\|\beta\|$$

$$ve \tag{18}$$

$$öyleki:\ y_i(\beta^T x_i + \beta_0) \geq 1,\ \ i = 1, \ldots, n$$

Genellikle Denklem 18'deki gibi bir vektörün büyüklüğü (normu) verildiğinde bazı problemlerle karşılaşılmaktadır. Bu problemlerden ilki $\frac{1}{2}\|\beta\|$ ifadesi türevlenebilen bir ifade olmamasıdır. Bu türevlenebilme şartını sağlayabilmek için tercih edilen işlemlerden ilki $\frac{1}{2}\|\beta\|$ ifadesinin karesini alarak $\frac{1}{2}\|\beta\|^2$ formuna dönüştürmektir. Böylece kareli ifadenin türevi alındığında, $\frac{1}{2}\|\beta\|$ ile aynı değer olan $\frac{1}{2}\beta^T\beta$ değeri elde edilebilmektedir. Bu durum Denklem 19'da gösterilmektedir.

$$en\ küçükleme: \frac{1}{2}\beta^T\beta$$

$$ve \tag{19}$$

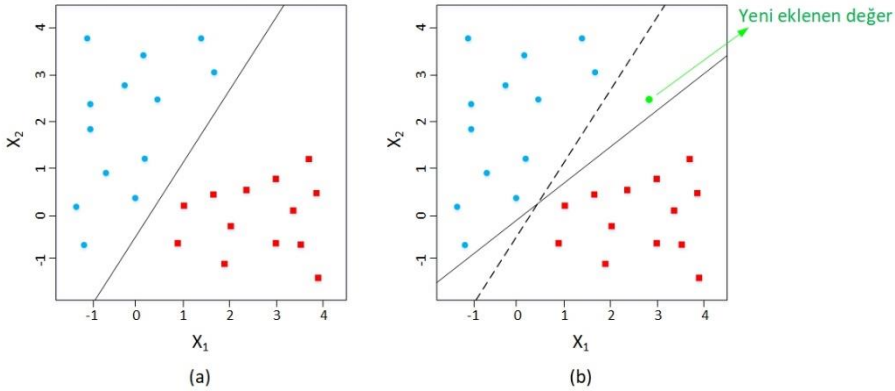
$$öyleki:\ y_i(\beta^T x_i + \beta_0) \geq 1,\ \ i = 1, \ldots, n$$

Burada optimizasyon işlemini gerçekleştirebilmek için $\frac{1}{2}\beta^T\beta$ amaç fonksiyonu dış bükey bir fonksiyondur.

$y_i(\beta^T x_i + \beta_0) \geq 1$ ifadesindeki β ve β_0 bilinmeyenleri ise doğrusal kısıtlılıklarımızı oluşturur. Amaç fonksiyonumuz karesel bir ifade olduğu için bu optimizasyon işlemi doğrusal olarak değil karesel programlama (quadratic) ile çözülebilecektir.

Bu matematiksel model, karesel ve dışbükey amaç fonksiyonu ile doğrusal kısıtlardan oluşuyor. Bu yapıdaki modellere karesel programlama (quadratic programming) modelleri denir.

Şu aşamaya kadar yapılan işlemler tam ayrıştırılabilir veri kümeleri için yapıldı ve veriler Şekil 15 (a)'da gösterildiği gibi bir doğru yardımı ile birbirinden ayrılabildi. Eğer bu veriler içerisine Şekil 15 (b)'de gösterildiği gibi yeni bir veri eklenirse ve eklenen veri de destek vektörüne en yakın değer ise, ilk durumda sınıfları birbirinden ayıran kesikli çizgi ile gösterilen doğrunun yeri değiştirilerek ve düz çizgi ile gösterilen konuma getirilmelidir. Buradan da anlaşılacağı üzere sınıf düzleminin konumu doğrusal vektörlere sıkı sıkıya bağlıdır ve stabil bir çözüm olmadığı görülmektedir. Bulunan ayırıcı doğrunun herhangi bir esneme payı yoktur. Bu durumda sistemin varyansını yükseltmektedir.



Şekil 15: Yeni Eklenen Değere Göre Destek Vektörünün Yeniden Belirlenmesi

Burada varyans kavramı, herhangi bir modelin tahmin ettiği değerin, gerçek verilerin etrafında nasıl (ne kadar) dağıldığı hakkında bilgi verir.

Eğer tasarlanan sistemin varyans değeri çok büyük ise, seçilen modelin fazla genel olduğunu gösterir. Bu duruma da aşırı öğrenme (ezberleme-overfitting) denir.

Herhangi bir modelin ne kadar yanlış olduğu da yanlılık kavramı ile ifade edilir. Yüksek yanlılık değeri elde edilmesinin en önemli nedeni, yanlış varsayımlardır. Örneğin kuadratik bir yapının, doğrusal bir yapı olduğunu varsaymak gibi yanlış varsayımlar, yüksek yanlılık değeri elde edilmesine neden olur.

Eğer tasarlanan sistemde sınıfları ayıran doğruya biraz esneme payı eklenecek olursa, örneğin sınıfları oluşturan verilerden birkaçı bu doğrunun diğer tarafına çok az geçmiş bir şekilde bulunsa biraz esneme payı vermiş oluruz. Bu esneme payı özellikle tam ayrıştırılamayan sınıfları ayırmak için kullanılabilecek bir yöntemdir (Birbil, 2018).

3.2. Tam Ayrıştırılamayan Veri

A) $\boldsymbol{\beta^T x + \beta_0 - 1 = 0}$ doğrusu (düzlemi)

Şekil 16'da görüleceği üzere, $\beta^T x + \beta_0 - 1 = 0$ doğrusunun üst tarafında ($\beta^T x + \beta_0 \geq 1$, bölgesi) bulunan veriler 1 sınıf etiketine ($y_i = 1$) sahiptir. Sisteme eklenen A verisi bulunması gereken sınıftadır. Bundan dolayı esneme katsayısı (Soft margin) olan ε değeri, sistemi etkilemeyecek şekilde Denklem 20'de gösterildiği üzere sıfırdır.

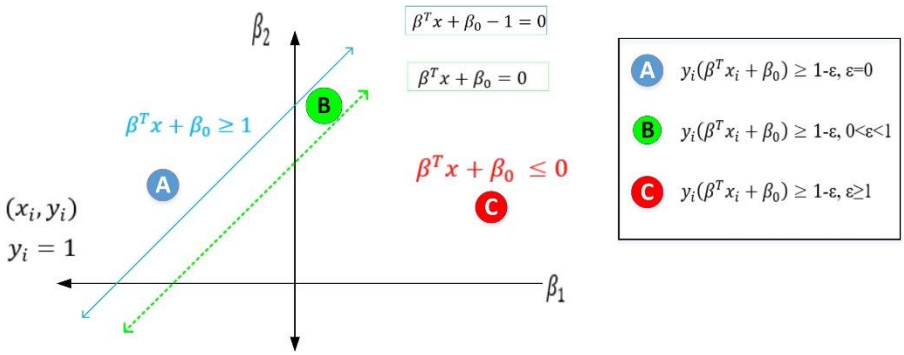
Sisteme eklenen değer B verisi ise $\beta^T x + \beta_0 - 1 = 0$ doğrusu ile $\beta^T x + \beta_0 = 0$ doğrusu arasındadır. Yani ε değeri Denklem 21'de gösterildiği gibi 0 ile 1 arasında bir değerdir. Sisteme son olarak

eklenen C verisi ise; Şekil 16'dan da anlaşılacağı üzere esneme değerinin (ε) 1'den büyük olduğu durumlarda elde edilen değer, diğer sınıfa ($\beta^T x + \beta_0 \leq 0$) geçmektedir. Bu durum Denklem 22 ile ifade edilmiştir.

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad \varepsilon=0 \tag{20}$$

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad 0<\varepsilon<1 \tag{21}$$

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad \varepsilon\geq 1 \tag{22}$$

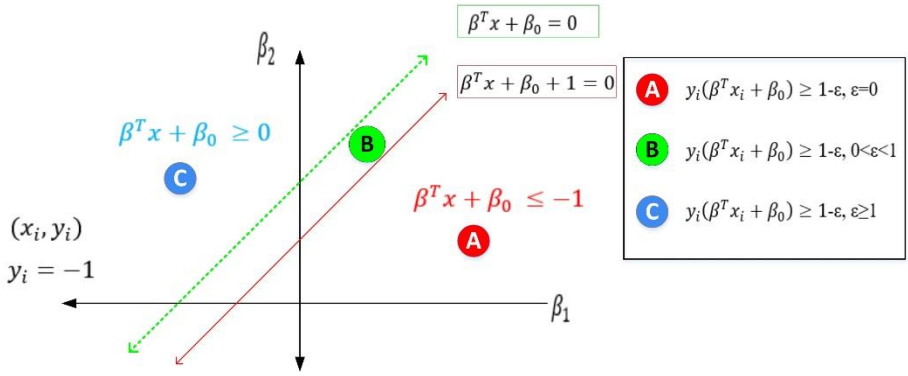


Şekil 16: Tam Ayrıştırılamayan Veri ($\beta^T x + \beta_0 - 1 = 0$ Doğrusu)

B) $\boldsymbol{\beta^T x + \beta_0 + 1 = 0}$ doğrusu (düzlemi)

Şekil 17'de görüleceği üzere, $\beta^T x + \beta_0 + 1 = 0$ doğrusunun alt tarafında ($\beta^T x + \beta_0 \leq -1$, bölgesi) bulunan veriler -1 sınıf etiketine ($y_i = -1$) sahiptir. Sisteme eklenen A verisi bulunması gereken sınıftadır. Bundan dolayı esneme katsayısı (Soft margin) olan ε değeri,

sistemi etkilemeyecek şekilde Denklem 23'de gösterildiği üzere sıfırdır.



Şekil 17: Tam Ayrıştırılamayan Veri ($\beta^T x + \beta_0 + 1 = 0$ Doğrusu)

Sisteme eklenen değer B verisi ise $\beta^T x + \beta_0 + 1 = 0$ doğrusu ile $\beta^T x + \beta_0 = 0$ doğrusu arasındadır. Yani ε değeri Denklem 24'de gösterildiği gibi 0 ile 1 arasında bir değerdir.

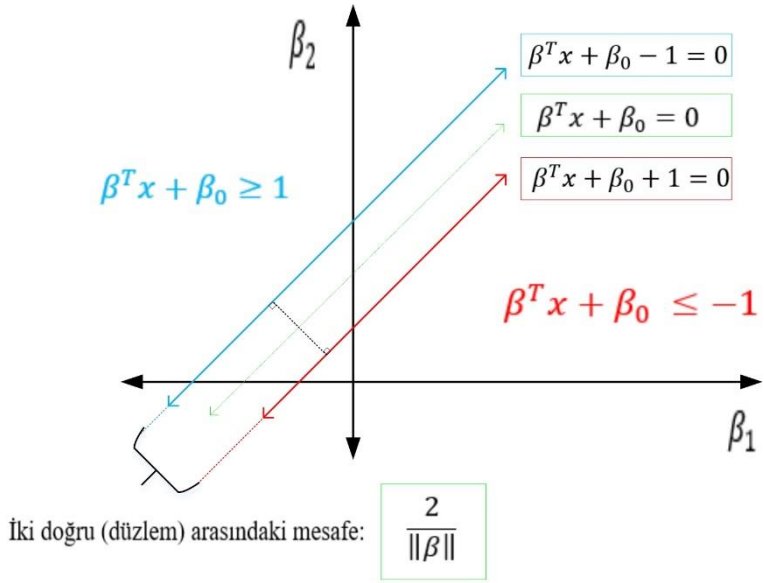
Sisteme son olarak eklenen C verisi ise; Şekil 17'den de anlaşılacağı üzere esneme değerinin (ε) 1'den büyük olduğu durumlarda elde edilen değer diğer sınıfa ($\beta^T x + \beta_0 \geq 0$) geçmektedir. Bu durum Denklem 25 ile ifade edilmiştir.

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad \varepsilon=0 \tag{23}$$

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad 0<\varepsilon<1 \tag{24}$$

$$y_i(\beta^T x_i + \beta_0) \geq 1\text{-}\varepsilon, \quad \varepsilon\geq 1 \tag{25}$$

Tasarlanan sistemin ilk haline geri gidildiğinde iki düzlem arasındaki mesafe Şekil 18'de gösterildiği gibidir. Sisteme eklenen esneme payları (soft margin) her iki sınıf için aynı olmakla beraber Denklem 20-25 arasında gösterilmiştir. Bu aşamadan sonra optimizasyon denklemlerinde de esneklik katsayısı (ε) eklenmiştir.



Şekil 18: İki Doğru (Düzlem) Arasındaki Mesafe

Yeni en küçükleme optimizasyon eşitsizlikleri Denklem 26'da verilmiştir.

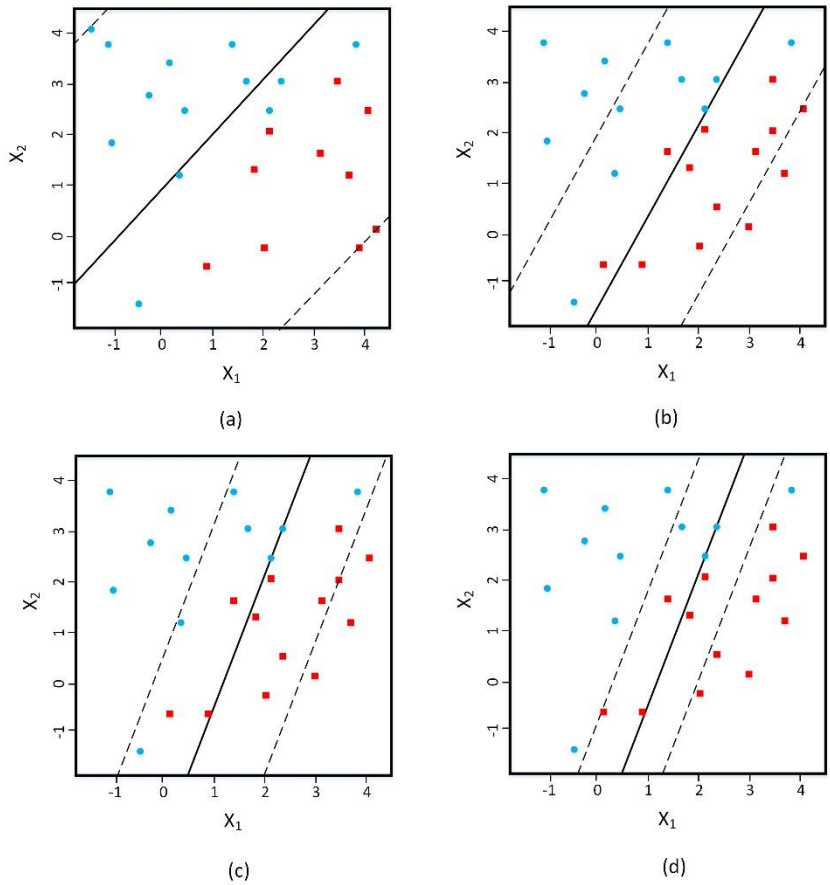
$$\begin{aligned} &en\ küçükleme && ½\, \beta^T\beta + c\sum_{i=1}^{n} \varepsilon_i^d \\ & && ve \\ &öyleki && y_i\,(\beta^T x_i + \beta_0) \geq 1 - \varepsilon_i,\ \ i = 1,\ldots,n \\ & && \varepsilon_i \geq 0,\ \ i = 1,\ldots,n \end{aligned} \tag{26}$$

Denklem 26'da ki *c* parametresi bize ikili bir durum oluşturmaktadır. Eğer *c* parametresi çok büyük olursa bize çok fazla esneklik sağlamış olacak ve bu durumda herşeyi kabul edecektir. Burada ki amaç, iki düzlem arasındaki mesafeyi en büyük tutmak ve bu esnada da esneme paylarını kabul edilebilir büyüklükte tutmaktır. Esneme katsayılarının toplamlarının hesaplanmasında kullanılan *d* parametresi 1 alınırsa esneme payı aynen alınır, 2 alınır ise esneme payının karesi alınır. Böylece negatif seçilen esneme katsayıları pozitife çevrilerek toplanması sağlanır (Birbil, 2018).

Burada *c* parametresi çapraz geçerlilik sınaması ile belirlenebilir (uygulamalarda *d*=1 ve *d*=2 değerleri sıkça kullanılır)

$$\text{en küçükleme} \quad \tfrac{1}{2}\beta^T\beta + c\sum_{i=1}^{n}\varepsilon_i^d \quad \text{ve} \quad \text{öyleki} \quad y_i\,(\beta^T x_i + \beta_0) \geq 1 - \varepsilon_i,\ \ i = 1,\ldots,n$$

$$\varepsilon_i \geq 0,\ \ i = 1,\ldots,n$$

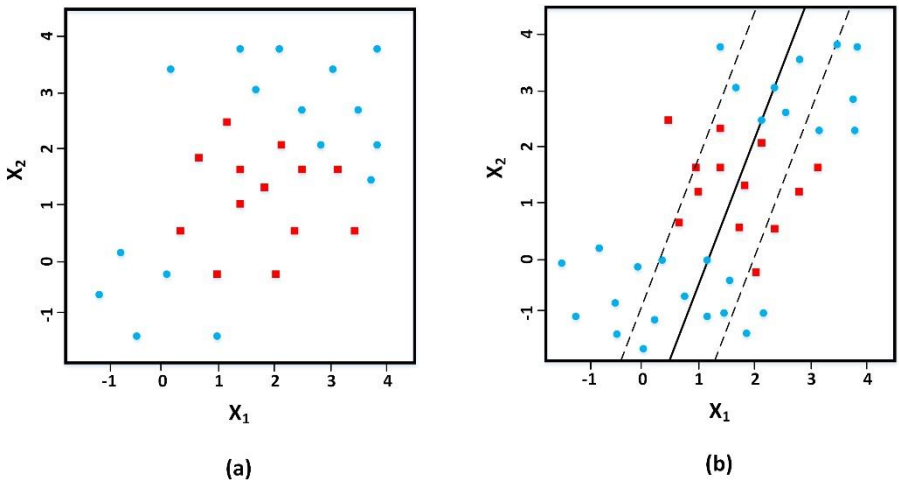


Şekil 19: c Parametresine Göre Esneme Paylarının Gösterimi

c parametresinin çok küçük seçilmesi durumunda Şekil 19 (a)'da gösterildiği gibi esneme çok büyük olacaktır. *c* parametresinin çok büyük seçilmesi durumunda Şekil 19 (d)'de gösterildiği gibi esneme çok küçük olacaktır.

4. Çekirdekler

Bu bölüme kadar anlatılan bölümde, veriler bir doğru ile birbirinden ayrılabilmektedir. Fakat kullanılan veriler Şekil 20 (a)'da ki gibi ise, bunları düz bir doğru (düzlem) ile Şekil 20 (b)'de gösterildiği üzere birbirinden ayırmak mümkün değildir. Bu tip veri setlerinin başka bir yöntemle ayrılması gerekmektedir. Çekirdek kavramları da tam bu noktada, bu sorunların çözümü olmaktadır. DVM'lerin literatürde bu kadar başarılı olmalarının sebebi çekirdek yöntemlerinin kullanılmasıdır.



Şekil 20: Birbirinden Bir Doğru ile Ayrılamayan Veriler

DVM'lerin bu kadar başarılı olmasının en önemli etkenlerden biri çekirdeklerin işin içerisine girmesidir. İlk defa destek vektör makineleri öneren Wapnik ve arkadaşlarının bu problemi çözmek için dışbükey bir amaç fonksiyonu ve doğrusal kısıtları kullanmışlardır (Vapnik, 1998).

Doğrusal olmayan programlama da bir problemin optimallik şartlarını yazarken yapılan ilk işlemlerden bir tanesi amaç fonksiyonunun yanına kısıtları da yazmaktır. Buna Lagrange fonksiyonu denilmektedir. Lagrange fonksiyonundaki α ifadesi Lagrange çarpanını, A ifadesi de aktif kısıtlar kümesini göstermektedir (Birbil, 2018).

$Lagrange\ Fonksiyonu$:

$$\mathcal{L}(\beta, \beta_0; \alpha) = \frac{1}{2}\beta^T\beta - \sum_{i\epsilon A} \alpha_i(y_i(\beta^T x_i + \beta_0) - 1) \tag{27}$$

Bu fonksiyondaki amaç fonksiyonumuz ½ $\beta^T\beta$, yanına da kısıtlarımızı ekliyoruz. Buradaki kısıtlarımız Denklem 28'de verilmiştir;

$$\sum_{i\epsilon A} \alpha_i(y_i(\beta^T x_i + \beta_0) - 1) \tag{28}$$

Amaç fonksiyonuna dikkat edilecek olursa, yanına a Lagrange çarpanları eklenmektedir. Burada yapılan iş aslında kısıtlı olan bir problemi kısıtsızmış gibi varsaymaktır. Ayrıca optimallik şartlarını da buna göre yazmak gerekmektedir. Optimallik şartlarına bakarken tüm kısıtlara değil sadece o anki ($i\epsilon A$) anlık şartlara bakılmaktadır.

Bu kısaca size herhangi bir β çifti verilirse o anki şartı sağlayanlara ($i = 1, \ldots, n$) bakılır.

Buradaki optimallik şartlarında, Lagrange fonksiyonunda β göre türevi alınır. Çok boyutlu olduğu için Gradyan şeklinde $\nabla_\beta \mathcal{L}(\beta, \beta_0; \alpha)$ gösterilir. Çünkü elimize ki ifade bir vektördür. Ayrıca ½ $\beta^T\beta$ ifadesi

2. dereceden olduğu için doğrudan β nın kendisi gelir. Sonuç olarak β ve β_0'a göre türevinin sıfır olması gerekir.

Burada bilinmeyenler β ve β_0 dır ve bunlar bizim için optimallik şartlarımızdır. Bu durum Denklem 29 ve 30'da gösterilmiştir.

$$\nabla_\beta \mathcal{L}(\beta, \beta_0; \alpha) = \beta - \sum_{i \epsilon A} \alpha_i y_i x_i = 0 \quad \rightarrow \quad \beta = \sum_{i \epsilon A} \alpha_i y_i x_i \tag{29}$$

$$\frac{\partial \mathcal{L}(\beta, \beta_0; \alpha)}{\partial \beta_0} = -\sum_{i \epsilon A} \alpha_i y_i = 0 \quad \rightarrow \sum_{i \epsilon A} \alpha_i y_i = 0 \tag{30}$$

Eğer $\sum_{i \epsilon A} \alpha_i y_i = 0$ ve $\sum_{i \epsilon A} \alpha_i y_i x_i$ işlemleri yapılır ve yerine konulursa Denklem 31 elde edilir.

$$\begin{aligned} \mathcal{L}(\beta, \beta_0; \alpha) &= \frac{1}{2}\left(\sum_{i \epsilon A} \alpha_i y_i x_i\right)^T \left(\sum_{j \epsilon A} \alpha_j y_j x_j\right) \\ &- \left(\sum_{i \epsilon A} \alpha_i y_i \left(\sum_{j \epsilon A} \alpha_j y_j x_j\right)^T x_i\right) \\ &- \sum_{i \epsilon A} \alpha_i y_i \beta_0 + \sum_{i \epsilon A} \alpha_i \\ \mathcal{L}(\beta, \beta_0; \alpha) &= \sum_{i \epsilon A} \alpha_i - \frac{1}{2} \sum_{i \epsilon A} \sum_{j \epsilon A} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned} \tag{31}$$

Burada girdilerin iç çarpanlarına $(x_i^T x_j)$ bağlı bir vektör ortaya çıkmaktadır. Bu nedenle yapılması gereken bu iç çarpımları

hesaplamaktır. Burada x_i ve x_j vektörlerinin çarpımı, aralarındaki benzerlik ile yani aralarındaki açı ile ilgilidir.

Hesaplamadaki esas fikir;

Bu iç çarpım hesabını daha yüksek boyutlu bir uzaya taşıyarak (yani 2 boyutluyu 3 boyuta, 3 boyutluyu 4 boyuta gibi) iki vektör arasında benzerliği tespit etmektir. Buradaki benzerliği ölçen, **çekirdek**tir. Böylece bir düzlem ile ayrılmayan veri setlerindeki gruplar birbirinden ayırabilir.

Çekirdek kullanımının önemi;

Kullanılan veriler Şekil 20(a)'da ki gibi bir dağılım gösterirse, bu verileri doğrusal düzlemler ile birbirinden ayırmak mümkün değildir. Bu nedenle Şekil 20 (a)'da veri setinde bulunan sınıflardan birisi, bir miktar yukarıya kaldırılırsa, altta kalan sınıf ile üste çıkan sınıf arasına yerleştirilecek bir düzlem ile iki sınıfı birbirinden ayırmak mümkün olacaktır. Burada sınıflardan birisini bir miktar yukarıya kaldırma işlemi çekirdekler vasıtası ile yapılmaktadır. Aynı mantıkla iç içe girmiş birçok sınıf birbirinden çekirdekler vasıtası ile kolayca ayrılabilir.

Bu çözüm sayesinde DVM'ler birçok farklı uygulama alanında sınıflandırma işlemleri için tercih edilmektedir. Bu sayede bir doğru ile ayrılayamayan sınıfların boyutunu artırarak bir düzlem vasıtası ile ayırmak mümkün olabilmektedir. Bu amaç için literatürde kullanılan farklı tipte çekirdek fonksiyonları bulunmaktadır. Bunlardan bazıları;

- Doğrusal,
- Polinom,
- Gauss,
- Laplace,
- Anova,
- Sigmoid,
- Bessel çekirdek fonksiyonlarıdır.

4.1. Doğrusal Çekirdek

Doğrusal çekirdek, en basit çekirdek fonksiyonudur. İç çarpımı ile ifade edilen doğrusal çekirdek fonksiyonu Denklem 32'de gösterilmektedir.

$$K(x_i, x_j) = x_i^T x_j \tag{32}$$

4.2. Polinom Çekirdek

Polinom çekirdek fonksiyonu durağan olmayan bir çekirdektir. Polinom çekirdeği, genellikle tüm eğitim verilerinin normalleştirildiği problemler için uygundur ve Denklem 33 ile gösterilmektedir.

$$K(x_i, x_j) = \left(1 + x_i^T x_j\right)^d \tag{33}$$

4.3. Gauss Çekirdek

Gauss çekirdeği radyal taban fonksiyonu çekirdeğine bir örnektir. İlgili çekirdek fonksiyonu Denklem 34 ile ifade edilir.

Gauss çekirdeğin performansında, ayarlanabilir parametreler önemli bir rol oynamaktadır. Uygulamaya göre dikkatle seçilmelidir. Aşırı tahmin

söz konusu olduğunda, üstel ifade yaklaşık olarak doğrusallaşır ve bir doğru gibi hareket etmeye başlar. Doğrusal olmayan yapısını kaybetmeye başlar.

$$K(x_i, x_j) = (e\ )^{-\gamma\|x_i - x_j\|^2}, \gamma > 0 \tag{34}$$

4.4. Laplace Çekirdek

Laplace çekirdeği bütünüyle üstel çekirdek fonksiyonuna eşdeğerdir, fakat sigma değerindeki değişikliklere karşı daha az duyarlıdır. Ayrıca Laplace çekirdek fonksiyonu, radyal tabanlı fonksiyon çekirdeği ile eşdeğerdir. Laplace çekirdek fonksiyonu Denklem 35 ile ifade edilmektedir.

$$K(x_i, x_j) = (e\ )^{\left(-\frac{\|x_i - x_j\|}{\sigma}\right)} \tag{35}$$

4.5. Anova Çekirdek

Anova radyal tabanlı fonksiyon çekirdek fonksiyonu, Gauss ve Laplace çekirdeği gibi radyal tabanlı bir fonksiyon çekirdeğidir. Çok boyutlu regresyon problemlerinde iyi performans gösterdiği bilinmektedir. Anova çekirdeğinin fonksiyonu Denklem 36'da gösterilmektedir.

$$K(x_i, x_j) = \sum_{i=1}^{n} exp\left(-\sigma\left(x_i^i - x_j^i\right)^2\right)^d \tag{36}$$

4.6. Sigmoid Çekirdek

Sigmoid çekirdek fonksiyonu birçok uygulamada iyi performans göstermektedir. Sigmoid çekirdeğinde α eğim ve c kesme sabiti olmak üzere iki ayarlanabilir parametresi vardır. α için ortak bir değer (1 / N) kullanılabilir ve N değeri veri boyutunu ifade etmektedir. Sigmoid çekirdek fonksiyonu Denklem 37 ile ifade edilmektedir.

$$K(x_i, x_j) = tanh(\alpha x_i^T x_j + c) \tag{37}$$

4.7. Bessel Çekirdek

Bessel çekirdeği düzgün fraksiyonel uzaylar teorisinde iyi bilinen bir çekirdektir. Bu çekirdek fonksiyon Denklem 38 ile gösterilmektedir (Souza, 2010). Denklem 38'deki J ifadesi birinci türden bessel değeridir.

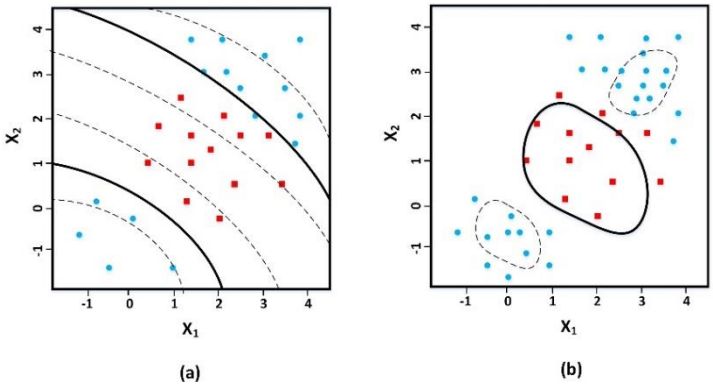
$$K(x_i, x_j) = \frac{J_{v+1}(\sigma\|x_i - x_j\|)}{\|x_i - x_j\|^{-n(v+1)}} \tag{38}$$

Yukarıda anlatılan literatürde bulunan çekirdek fonksiyonlar vasıtasıyla, veriler iki boyutlu uzaydan, çok boyutlu uzaylara kolayca aktarabilmektedir. Bunun sonucunda da iki boyutlu uzayda ayrıştırılamayan veriler ayrıştırılabilir hale gelmektedir.

Örneğin, Şekil 21(a)'da gösterilen veriler kullanıldığında, bu verileri düzlemler ile ayırmak mümkün değildir.

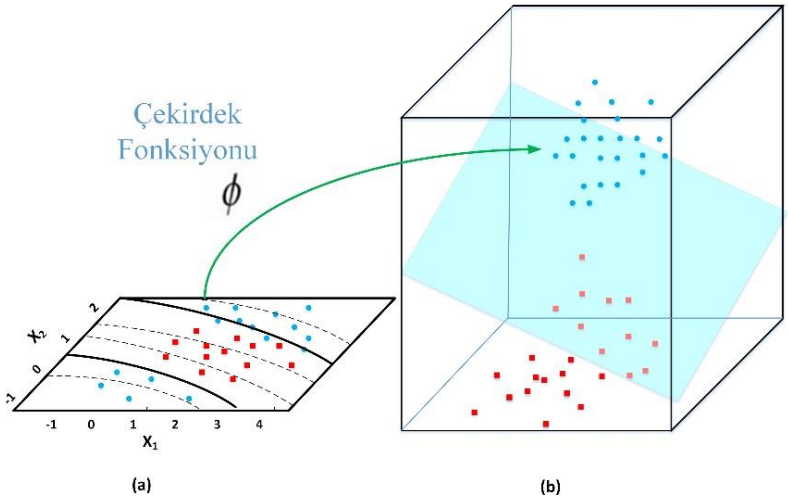
Bu nedenle, yine Şekil 21(a)'da gösterilen eğriler kullanılmaktadır. Fakat Şekil 21(b)'de gösterilen veriler kullanıldığında bunları yine

eğriler ile ayırabilmek mümkün değildir. Bu nedenle Şekil 21(b)'de gösterilen noktalar bulunduğu düzlemden bir miktar yukarıya kaldırılır.



Şekil 21: Çekirdek Fonksiyonları

Böylece yukarıya doğru 3. bir boyut daha üretilmiş olmaktadır. Bunun sonucunda da kırmızı ve mavi veri noktaları arasından Şekil 22'de gösterilen bir düzlem daha geçirilebilecektir. Bu düzlem ile de sınıflar birbirinden ayrılabilmektedir.



Şekil 22: Boyut Arttırma

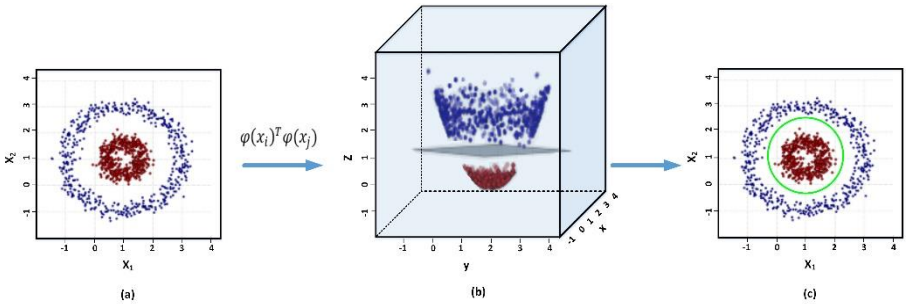
Konu ile ilgili detaylı başka bir örnekte çekirdek fonksiyonu φ ise ;

$$\varphi: R^P \rightarrow R^q$$

Kullanılan çekirdek fonksiyonun içeriği;

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

Bu çekirdek fonksiyonu kullanılarak sınıflandırma işlemi yapılırsa yani aynı düzlemdeki iki sınıftan birisi bir miktar yukarıya kaldırılırsa Şekil 23 (b)'de ki durum ortaya çıkmaktadır.



Şekil 23: Çekirdek Fonksiyon ile Ayırıcı Düzlemi Belirleme

Burada her uygulamada farklı veri setleri için farklı çekirdekler test edilerek sınıflandırma işlemi yapılabilmektedir. Farklı durumlarda probleme özel çekirdeklerde kullanılabilmektedir. Örnek olarak görüntü işleme uygulamalarında elde edilen istatiksel veriler doğrultusunda radyal çekirdeklerin daha iyi sonuç verdiği tespit edilmiştir.

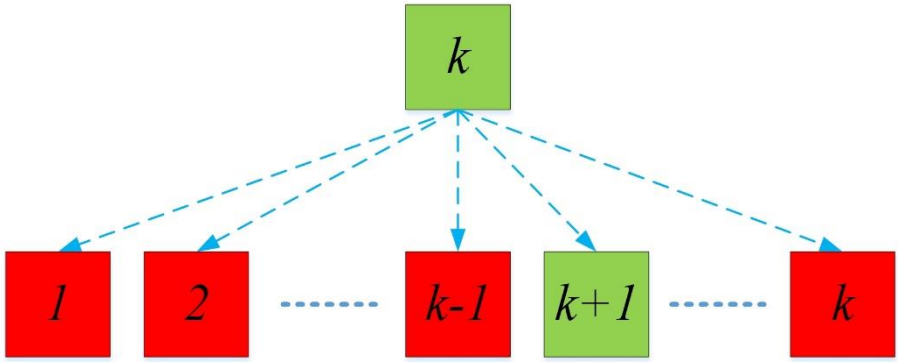
5. Çoklu Sınıf

Şimdiye kadar anlatılan işlemler +1 ve -1 şeklinde iki sınıf için anlatılmıştır. Eğer elimizdeki veriler ikiden fazla sınıf içerirse; tüm çiftler ve biri diğerlerine karşı yöntemleri kullanılmaktadır.

5.1. Tüm Çiftler

Burada ilk olarak, bir veri alınır ve bu veri Şekil 24'de gösterildiği gibi k. sınıfta mı yoksa k+1. sınıfta mı diye tüm ikili çiftler teker teker kontrol edilir. Bu işlem matematiksel olarak;

$\binom{K}{2}$ DVM, şeklinde gösterilir.



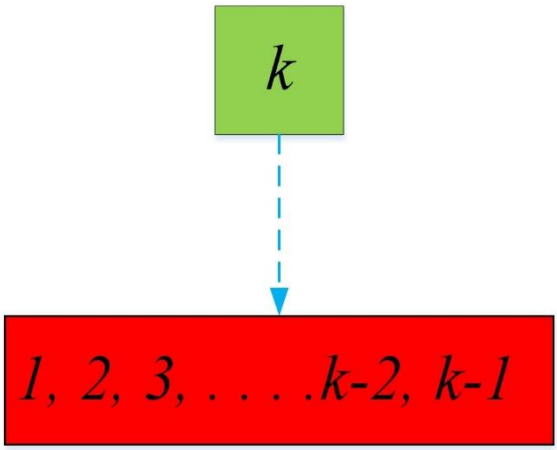
Şekil 24: Tüm Çiftler

Daha sonrasında k+1. sınıf alınır ve k+2. sınıfta mı diye geri kalan tüm sınıflar kontrol edilir. Bu işlem tüm sınıflara ikili çiftler halinde uygulanınca sona erer. Başka bir yeni veri gelirse, daha önce en çok hangi sınıfa atanmışsa o sınıfa dahil edilir.

5.2. Biri Diğerlerine Karşı

Bu sınıflandırma yönteminde, bir veri alınır bir sınıfa dahil edilir. Geri kalan tüm sınıflarda Şekil 25'de gösterildiği gibi tek bir sınıf gibi kabul edilir ve diğerlerine karşı kıyaslama işlemi yapılır. İsmini de bu durumdan almış ve matematiksel olarak da Denklem 39'daki gibi gösterilir;

$$\beta_0^{(k)} + \beta^{(k)T} x_0 \tag{39}$$



Şekil 25: Biri Diğerine Karşı

Sistemde var olan her bir veri, diğerleri ile kıyaslandığı için k adet DVM elde edilir. Bunların her biri için β_0 ve β vektörleri bulunur. Bu işlemde k tane sınıf olduğu için, k tane karşılaştırma yapılır ve en yüksek x_0 değeri kaçıncı DVM'de elde edilirse yeni eklenen veri o sınıfa dahil edilir.

BÖLÜM 3 - MATLAB UYGULAMALARI

3. Matlab Uygulamaları

Bu bölümde, makine öğrenmesinin denetimli öğrenme modellerinden birisi olan ve genellikle sınıflandırma uygulamalarında kullanılan DVM'ler ile ilgili örnek uygulamalar sunulmuştur. DVM'ler her ne kadar sınıflandırma işlemleri için kullanılsa da regresyon için kullanılan DVM modelleri de literatürde bulunmaktadır. Bu nedenle DVM'lerin kullanım alanları çok geniştir. Bunlara örnek vermek gerekirse;

- Yüz algılama,
- Metin ve hiper metin sınıflandırma,
- Görüntülerin sınıflandırılması,
- Biyoinformatik,
- El yazısı tanıma.

DVM için Matlab, Pyhton, C# gibi ortamlarda geliştirilen özel fonksiyonlar bulunmaktadır. Kitabın bu bölümünde Matlab ortamında fitcSVM fonksiyonu ile gerçekleştirilen DVM uygulamaları anlatılacaktır. Bu nedenle ilk olarak fitcSVM fonksiyonunun genel yapısı ve giriş parametreleri aşağıda ayrıntılı olarak anlatılmıştır.

3.1. fitcSVM fonksiyonu

fitcSVM fonksiyonu, sınıflandırma işlemleri için Matlab ortamında kullanılan bir DVM fonksiyonudur. Genellikle düşük veya orta

boyutlardaki veri kümelerini sınıflandırmak için kullanılmaktadır. Kullanıcı isteğine bağlı olarak, fitcSVM fonksiyonu çekirdek fonksiyonları ile de kullanılabilmektedir. fitcSVM fonksiyonu ikinci dereceden programlama yoluyla, Sıralı Minimum Optimizasyonu (Sequential Minimal Optimization), Yinelemeli Tek Veri Algoritmasını (Iterative Single Data Algorithm) veya L1 yumuşak marj (L1 Soft-Margin) ile sınıflandırma hatalarını en aza indirmeyi hedeflemektedir.

Yüksek boyutlu bir veri kümesinde, yani birçok özniteliğin bulunduğu bir veri kümesinde, ikili sınıflandırmayı eğitmek için Matlab'da bulunan **fitclinear** fonksiyonu da kullanılabilmektedir.

Çok sınıflı uygulamalarda (İkiden fazla etiket bilgisi bulunan veriler), tüm çiftler yada biri diğerlerine karşı yöntemleri kullanıldığında, hata tespiti (Error-Correcting Performance Codes - ECOC) ve sınıflandırma için, Matlab'ın **fitcecoc** fonksiyonu kullanılabilir.

Matlab'da DVM modelleri ile sınıflandırma işlemlerinden farklı olarak, regresyon işlemleri gerçekleştirilmek istenilirse **fitrSVM** fonksiyonu kullanılabilir. Genel olarak düşük ve orta boyutlu veri kümeleri ile regresyon tahmini için **fitrSVM** fonksiyonu kullanılmaktadır. Ayrıca, çok boyutlu veriler ile regresyon işlemleri gerçekleştirilmek istenirse **fitrlinear** fonksiyonu da kullanılabilir.

Bu bölümde örnek uygulamalar üzerinde sınıflandırma işlemleri gerçekleştirileceği için **fitcSVM** fonksiyonu kullanılacaktır. Bu nedenle **fitcSVM** fonksiyonu detaylı olarak anlatılmıştır.

3.1.1. fitcSVM fonksiyonunun parametreleri

Matlab ortamında DVM modelleri ile sınıflandırma işlemlerinde aşağıda genel kullanım şekli verilen **fitcSVM** fonksiyonu kullanılmaktadır (Andrew, 2000; Fan, Chen, & Lin, 2005; Hastie, Tibshirani, & Friedman, 2009; Kecman, Huang, & Vogt, 2005). Aşağıda verilen yapıda fitcSVM fonksiyonunda X, Y, KernelFunction, Standardize, BoxConstraint, ClassNames parametreleri kullanılmıştır.

```
DVMModel = fitcSVM(X,Y,…
                    'KernelFunction','rbf',…
                    'Standardize',true,…
                    'BoxConstraint',Inf,…
                    'ClassNames',{'negClass','posClass'});
```

Fonksiyonda kullanılan giriş parametrelerinin özellikleri;

X: Tablo 1'de gösterilen veri matrisi X ile ifade edilmektedir. Bu matrisin her satırında bir gözlem ve her sütununda da sınıfa ait bir öznitelik değeri bulunmaktadır.

Tablo 1: Veri Tablosu

Gözlem No	**Öznitelikler**				
	1	2	3	4	5
1	5,1	3,5	1,4	0,2	5,1
2	4,9	3	1,4	0,2	4,9
3	4,7	3,2	1,3	0,2	4,7
4	4,6	3,1	1,5	0,2	4,6
5	5	3,6	1,4	0,2	5

Y: Tablo 2'de her gözleme ait koyu renkte gösterilen sınıf bilgisi, Y ile ifade edilmektedir. Sınıf bilgisi olarak Matlab'da kullanılan veri türleri; karakter, string, mantıksal değer veya vektör olabilir.

Tablo 2: Tüm Veri Tablosu

Gözlem No	**Öznitelikler**					**Sınıf Bilgisi**
	1	2	3	4	5	
1	5,1	3,5	1,4	0,2	5,1	A
2	4,9	3	1,4	0,2	4,9	B
3	4,7	3,2	1,3	0,2	4,7	A
4	4,6	3,1	1,5	0,2	4,6	B
5	5	3,6	1,4	0,2	5	B

KernelFunction: DVM sınıflandırıcısını başarılı bir şekilde eğitmenin en önemli adımı, uygun bir çekirdek fonksiyonunun (**KernelFunction**) seçilmesidir. KernelFunction'ın varsayılan değeri doğrusaldır (linear). Doğrusal ifadesi iki sınıf veri etiketine sahip veri kümesindeki verileri, bir hiper düzlem ile ayıran doğrudur (düzlem). Veriler iki boyutlu uzayda bulunduğunda, bu verileri bir doğru vasıtası ile birbirinden ayırmak mümkündür. Eğer veriler iki boyutlu uzaydan daha büyük boyutlu bir uzayda bulunuyorsa, verileri bir doğru yerine düzlem (hiper düzlem) ile ayırmak mümkündür.

fitcSVM fonksiyonunda varsayılan çekirdek fonksiyonu Gaussian veya radyal tabanlı bir fonksiyon olan 'rbf' fonksiyonudur. Tablo 3'de **fitcSVM** fonksiyonu ile kullanılan temel çekirdek fonksiyonları ayrıntılı bir şekilde açıklanmıştır. Tablo 3'de bulunan $G(x_j, x_k)$ ifadesi, çekirdek fonksiyondan elde edilen değeri ifade etmektedir.

Tablo 3: Çekirdek Fonksiyonları

Çekirdek Fonksiyon	**Tanımı**	**Formülü** $G(x_j, x_k)$
'gaussian' veya 'rbf'	Tek sınıflı öğrenmede varsayılan olarak Gaussian veya Radial Basis Function (RBF) çekirdeği kullanılır.	$\exp\left(-\left\|x_j, x_k\right\|^2\right)$
'linear'	İki sınıflı öğrenme için varsayılan olarak Linear çekirdek kullanılır.	$x_j' x_k$
'polynomial'	q. dereceden bir polinom fonksiyonu için Polynomial çekirdeği kullanılır.	$\left(1 + x_j' x_k\right)^q$

Standardize: Sınıflandırıcıyı eğitmeden önce kullanılan veriler üzerinde standartlaştırma işleminin gerekli olup olmadığını belirten işarettir.

BoxConstraint: Kutu kısıtlama parametresi olarak kullanılır. Tam bir sınıflandırma yapmak için kutu kısıtlama parametresinin değeri "Inf" olarak ayarlanır, böylece yanlış sınıflandırılmış eğitim noktasının olma ihtimali ortadan kaldırılır.

ClassNames: Verilerin hangi sınıflara dahil edileceğini belirten parametredir (Sınıf etiketleri negatif ve pozitif sınıf olarak ayrılır). ClassNames, **Y** parametresi ile aynı veri tipinde olmalıdır. Özellikle farklı sınıflandırıcıların performansını karşılaştırıyorsanız, sınıf isimlerini belirtmek iyi bir yöntem olabilir. fitcSVM fonksiyonu ile birlikte kullanılan diğer fonksiyonlar Tablo 4'de açıklamaları ile birlikte sunulmuştur.

Tablo 4: fitcSVM Nesne Fonksiyonları

Fonksiyon İsmi	**Açıklaması**
compact	DVM sınıflandırıcısının boyutunu azaltır.
compareHoldout	Yeni verileri kullanarak iki sınıflandırma modelinin doğruluğunu karşılaştırır.
crossval	DVM çapraz doğrulamasını yapar. Varsayılan değer olarak 10 değerini alır.
discardSupportVectors	Doğrusal DVM sınıflandırıcısı için destek vektörlerini atar.
edge	Sınıflandırma kenarlarını bulur.
fitPosterior	Uygun olasılıkları hesaplar.
incrementalLearner	İkili sınıflandırmada artımlı öğrenme için DVM modeli oluşturma.
loss	Sınıflandırma hatasını bulur.
margin	Marjinlerin sınıflarını bulur.
partialDependence	Kısmi bağımlılığı hesaplar.
plotPartialDependence	Kısmi bağımlılık grafiği ve bireysel koşullu beklenti grafikleri oluşturur.
predict	Sınıfları tahmin eder.
resubEdge	Sınıflandırma kenarını tekrar oluşturur.
resubLoss	Sınıflandırma hatasını tekrar hesaplar.
resubMargin	Marjin sınıflarını tekrar bulur.
resubPredict	Sınıfları tekrar tahmin eder.
resume	Eğitimini tekrar başlatır.

3.2. Uygulama 1

3.2.1. Ionosphere Veri Seti

Uygulamada kullanılan veriler radar verileridir. Bu veriler, Johns Hopkins Üniversitesi, Uygulamalı Fizik Laboratuvarı, Uzay Fiziği Grubu tarafından toplanmıştır. Farklı amaçlarda kullanımı için 1989 yılında da kamuoyu ile paylaşılmıştır. Veriler Labrador, Goose Bay'de bulunan toplam 16 adet yüksek frekanslı antenden oluşan radar sisteminden elde edilmiştir. Radarları, iyonosferin 100-500 km'leri arasında bulunan E ve F katmanlarının fiziğini incelemek için kullanmışlardır.

Çalışmanın hedefi, iyonosferde serbest halde dolaşan 10^{-30} m^2 kesit civarındaki küçük elektronların tespit edilmesidir. Bunun için gönderdikleri radar sinyalinin geri dönüş değerlerindeki gürültüleri **iyi** yada **kötü** olarak sınıflandırmışlardır. Sınıflandırdıkları verileri de başta California Üniversite (UCI) olmak üzere, kamuoyuna açık birçok veri tabanında paylaşmışlardır (Baker, Greenwald, Villain, & Wing, 1987; Dua, Dheeru and Graff, 2017; Greenwald, Baker, Hutchins, & Hanuise, 1985; Sigillito, Wing, Hutton, & Baker, 1989; Walker, Greenwald, & Baker, 1987).

Veri setinde 351 adet gözlem ve her bir gözleme ait 34 adet ölçüm ve son olarak da her bir gözlemden elde edilen 351 adet sonuç değeri bulunmaktadır. Veri setinde bulunan değerlerden bazıları Tablo 5'de verilmiştir.

Tablo 5: Ionosphere Veri Seti

Gözlem No	Ölçüm No					Sonuç
	1	2	3	…….	34	Y
1	-0,05889	0,85243	0,02306	…….	-0,453	g
2	-0,18829	0,93035	-0,36156	…….	-0,02447	b
3	-0,03365	1	0,00485	…….	-0,38238	g
⋮	⋮	⋮	⋮	⋮	⋮	⋮
349	-0,00034	0,93207	-0,03227	…….	-0,00577	g
350	-0,01657	0,98122	-0,01989	…….	-0,16243	g
351	0,13533	0,73638	-0,06151	…….	-0,06151	g

3.2.2. Matlab Uygulaması

İlk olarak Matlab'da hazır olarak bulunan iyonosfer veri seti çalışma alanına (Workspace) aşağıdaki şekilde yüklenmektedir (Matlab, 2020).

```
clc;

clear all;

close all;

load ionosphere

rng(1);
```

- Uygulamada Radial (RBF) temelli çekirdek kullanarak bir DVM sınıflandırıcı oluşturma hedeflenmektedir. Bu nedenle çekirdek fonksiyonuna aşağıdaki şekilde parametre girilmiştir.

  'KernelFunction','RBF'

- fitcSVM fonksiyonunda kullanılacak çekirdeğin otomatik bir ölçek değeri bulmasına izin verilmesi gerekmektedir. Bunun içinde aşağıdaki parametre fitcSVM fonksiyonuna eklenmelidir.
  'KernelScale','auto'

- Kullanılan verilerin standart hale getirilmesi gerekmektedir. Bu yüzden Standardize giriş parametresine true değeri girilmelidir.
  'Standardize',true

fitcSVM fonksiyonuna yukarda belirtilen parametreler eklendiğinde, DVMModel nesnesi aşağıdaki forma dönüşmektedir.

```
DVMModel =      fitcSVM(X,Y,…
                'Standardize',true, …
                'KernelFunction','RBF',…
                'KernelScale','auto');
```

- Oluşturulan DVMModel nesnesi denetimli bir sınıflandırıcıdır. Bu nedenle çapraz doğrulama yapmak gerekebilir. Bunun için ise aşağıdaki belirtilen crossval komut kullanılır. Aksi belirtilmedikçe varsayılan çapraz doğrulama (k-fold) değeri 10 olarak kullanılır.

```
CVDVMModel = crossval(DVMModel);
```

CVDVMModel, nesnesi artık çapraz doğrulanmış bir sınıflandırıcı nesnesidir. Yanlış sınıflandırma oranı hesaplanmak istenirse aşağıda kullanılan kfoldLoss fonksiyonu ile hesaplanabilir.

```
kayip = kfoldLoss(CVDVMModel)
```

```
kayip = 0.0484
```

Genel olarak uygulamanın hata oranı, yukarıdaki parametreler ile yaklaşık % 5 (0.0484) olarak hesaplanmıştır.

3.3. Uygulama 2

3.3.1. Iris Veri Seti

Bu uygulamada kullanılan veri seti, İngiliz biyolog ve istatistikçi Ronald Fisher tarafından 1936 tarihli makalesinde, çoklu ölçümlerin doğrusal ayırıcı analizine bir örnek olarak sunduğu çok değişkenli bir veri kümesidir. Bu veri setine, iris veri setinin yanında bazen Fisher's bazen de Anderson’s veri seti de denilmektedir (Anderson, 1936; Fisher, 1936b).

Veri setinde Iris çiçeğinin, Tablo 6’da gösterilen setosa, versicolor ve virginica türlerine ait özellikler bulunmaktadır. Bu üç türün her birinden 50 adet olmak üzere toplam 150 adet numuneden ölçüm değerleri alınmıştır. Her numuneden dört özellik ölçülmüştür. Bu ölçülen özellikler çanak yaprakları ve taç yapraklarının santimetre cinsinden uzunluğu ve genişliğidir.

Tablo 6: Iris Veri Setindeki Çiçek Türleri (Kolhe & Ranjana, 2010)

Setosa	Versicolor	Virginica

Temeli Fisher'in 1936 yılındaki makalesinde doğrusal diskriminant modeline dayanan bu veri seti, başta DVM gibi makine öğreniminde kullanılan birçok istatistiksel sınıflandırma tekniğinde test amaçlı kullanılmıştır (Fisher, 1936a). Bu, belki de sınıflandırma ile ilgili literatürde bulunabilecek en iyi bilinen veri tabanıdır. Bu nedenle gerçekleştirilen uygulamada da bu veri seti tercih edilmiştir.

Iris veri setindeki bir sınıf diğer iki sınıftan doğrusal olarak ayrılabilirken, diğer iki sınıf birbirinden doğrusal olarak ayrılamaz. Bu nedenle, Iris veri setindeki üç türü birbirinden ayırabilmek için doğrusal olmayan yöntemler kullanılır. Iris veri setinin bu yapısı sayesinde, makine öğrenmesinde denetimli ve denetimsiz yöntemler arasındaki farkları açıklamak için iyi bir örnek haline gelmiştir. Veri setinde bulunan değerlerden bazıları Tablo 7’de sunulmuştur.

Tablo 7: Iris Veri Seti

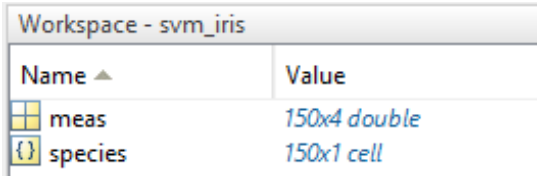
Gözlem No	Ölçüm No				Sonuç
	1	2	3	4	Y
1	5,1	3,5	1,4	0,2	setosa
2	4,9	3	1,4	0,2	setosa
⋮	⋮	⋮	⋮	⋮	⋮
51	7	3,2	4,7	1,4	versicolor
52	6,4	3,2	4,5	1,5	versicolor
⋮	⋮	⋮	⋮	⋮	⋮
101	6,3	3,3	6	2,5	virginica
102	5,8	2,7	5,1	1,9	virginica
⋮	⋮	⋮	⋮	⋮	⋮
150	5,9	3	5,1	1,8	virginica

3.3.2. Matlab Uygulaması

Bu uygulamada genel amaç, farklı sınıflara ait olduğu bilinen verileri sanki tek sınıfmış gibi değerlendirerek, aykırı olan verileri göstermektir. Bunun için ilk olarak Matlab'da yüklü olan fisheriris veri seti çalışma alanına (Workspace) aşağıdaki şekilde yüklenmektedir. Sonrasında da tüm sınıfların, sanki tek sınıfmış gibi değerlendirmek için ölçüm sonuçları aşağıdaki şekilde değiştirilmektedir (Matlab, 2020).

```
clc;
clear all;
close all;
load fisheriris
rng(1);
X = meas(:,1:2);
y = ones(size(X,1),1);
```

Çalışma alanını fisheriris veri seti yüklendiğinde Şekil 26'da görüldüğü üzere species ve meas değişkenleri de otomatik olarak yüklenmiştir.



Şekil 26. fisheriris Veri Setinin İçeriği

meas değişkeni double tipinde bir matristir. Bu matrisin içerisinde Tablo 7'de gösterilen ölçüm değerleri bulunmaktadır. species değişkenin içerisinde de çiçek sınıf bilgileri bulunmaktadır.

Yukarıdaki kod parçacığındaki;

- X = meas(:,1:2);

ifadesi ile Iris veri setindeki tüm verilerden sadece 1. ve 2. ölçüm değerleri (Taç yaprak genişlik ve uzunluğu) alınarak X değişkenine aktarılmıştır.

- y = ones(size(X,1),1);

ifadesi ile tüm sınıfları sanki tek bir sınıf gibi değerlendirebilmek için y değişkenine tüm sınıf bilgileri 1 olarak atanmıştır.

- Uygulamada ölçüm sonuçlarının belli bir kısmının uç değer olabileceği düşünülmektedir. Bunun için OutlierFraction fonksiyonu ile bunların ne kadarının uç değer olabileceği bilgisi DVM sınıflandırıcısına aşağıdaki parametre ile aktarılmıştır.
  ' OutlierFraction ', 0.05

fitcSVM fonksiyonuna yukarda belirtilen parametreler eklendiğinde aşağıdaki forma dönüşmektedir.

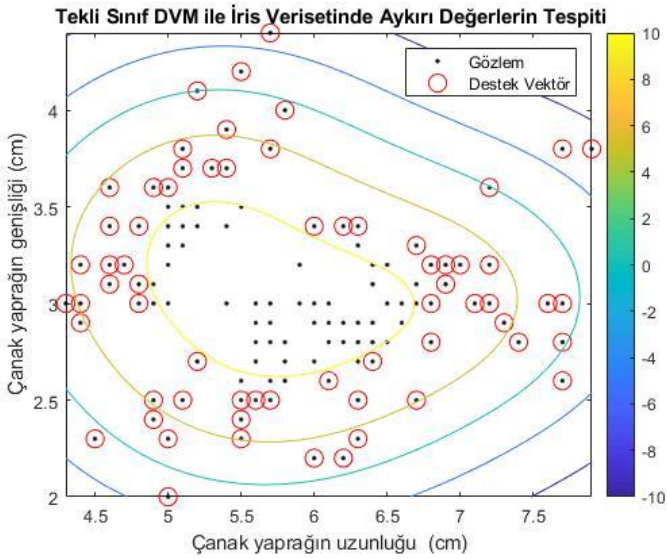
```
DVMModel =      fitcSVM(X,y,…
                'Standardize',true, …
                'OutlierFraction', 0.05,…
                'KernelScale','auto');
```

Şu aşamada verilerin hepsi sanki aynı sınıftaymış gibi bir değerlendirme yapılarak benzer özellik değerlerine sahip olan sınıf keşfedilmeye çalışılmaktadır. Yani aynı zamanda aykırı değerlerde gözlemlenmeye çalışılmaktadır.

DVMModel, denetimli bir ClassificationDVM sınıflandırıcısıdır. Varsayılan olarak Matlab tek sınıflı öğrenme için Tablo 3'de de gösterildiği gibi Gaussian çekirdeğini kullanır.

Gözlemleri ve karar sınırlarını çizmek ve ardından destek vektörleri ile potansiyel aykırı değerleri işaretlemek için aşağıdaki kod bloğu kullanılmaktadır.

```
svInd = DVMModel.IsSupportVector;
h = 0.02; % Izgara için adım boyutu
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)),...
                min(X(:,2)):h:max(X(:,2)));
[~,score] = predict(DVMModel,[X1(:),X2(:)]);
scoreGrid = reshape(score,size(X1,1),size(X2,2));
figure  % Elde edilen sonuçlarının grafiğinin çizilmesi
plot(X(:,1),X(:,2),'k.')
hold on
plot(X(svInd,1),X(svInd,2),'ro','MarkerSize',10)
contour(X1,X2,scoreGrid)
colorbar;
title('{\bf Tekli Sınıf DVM ile İris Verisetinde Aykırı Değerlerinin…
        Tespiti}')
xlabel('Çanak yaprağın uzunluğu  (cm)')
ylabel('Çanak yaprağın genişliği (cm)')
legend('Gözlem','Destek Vektör')
hold off
```



Şekil 27: İris Veri Setinde Tek Sınıf İçin Tespit Edilen Aykırı Değerler

Şekil 27'de görüldüğü gibi aykırı değerleri, verilerin geri kalanından ayıran sınır kontur değerinin sıfır olduğu yerde oluşmaktadır.

Çapraz doğrulanmış verilerde negatif puanlı gözlemlerin fraksiyonunun %5'e yakın olduğunu doğrulamak için aşağıdaki kod bloğu yazılmaktadır.

```
CVDVMModel = crossval(DVMModel);

[~,scorePred] = kfoldPredict(CVDVMModel);

aykirilik_Orani = mean(scorePred<0)
```

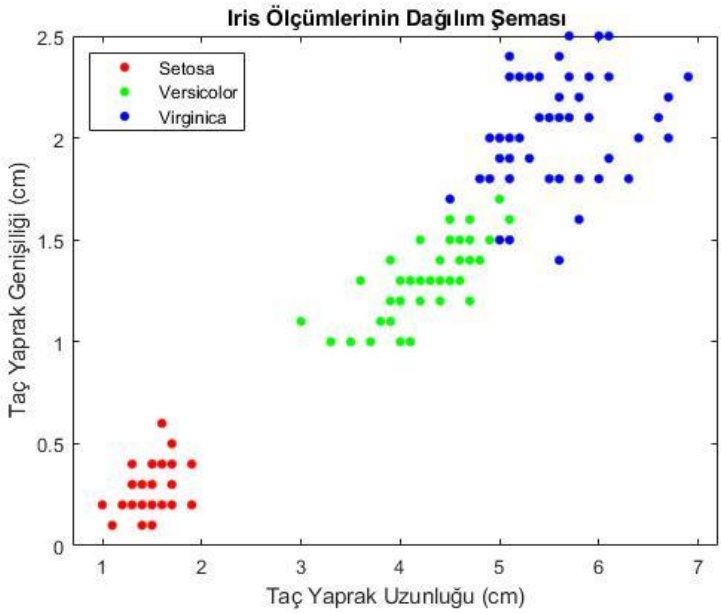
```
aykirilik_Orani =    0.0467
```

Genel olarak uygulamanın aykırılık oranı, yukarıdaki parametreler ile yaklaşık %5 (0.0467) olarak hesaplanmıştır.

3.4. Uygulama 3

3.4.1. Iris Veri Seti

Uygulamada 2'de kullanılan Iris veri seti bu uygulamada da aynı şekilde kullanılmıştır. Hatırlanacağı üzere Iris veri setindeki bir sınıf, diğer iki sınıftan doğrusal olarak ayrılabilirken; diğer iki sınıf Şekil 28'de görüleceği üzere birbirinden doğrusal olarak ayrılamıyordu. Bu nedenle bu uygulamada birbirinden doğrusal olarak ayrılamayan (Versicolor ve Virginia) sınıflar kullanılmıştır.



Şekil 28: Iris Veri Seti.

3.4.2. Matlab Uygulaması

Bu uygulamanın genel amacı, doğrusal olarak birbirinden ayrılamayan iki sınıfı birbirinden ayırmaktır. Bunun için ilk olarak Matlab'da yüklü

olan fisheriris veri seti çalışma alanına (Workspace) yüklenmektedir (Matlab, 2020).

```
load fisheriris
inds = ~strcmp(species,'setosa');
X = meas(inds,3:4);
Y = species(inds);
```

- inds = ~strcmp(species,'setosa');

komutu satırı ile fisheriris veri setinde bulunan sınıflardan, Setosa sınıfına ait olmayan sınıfların gözlem numarası alınarak, inds vektörüne atanmıştır. Böylece diğer iki sınıf olan Versicolor ve Virginia sınıflarına ait gözlem numaraları elde edilmiştir.

- X = meas(inds,3:4);

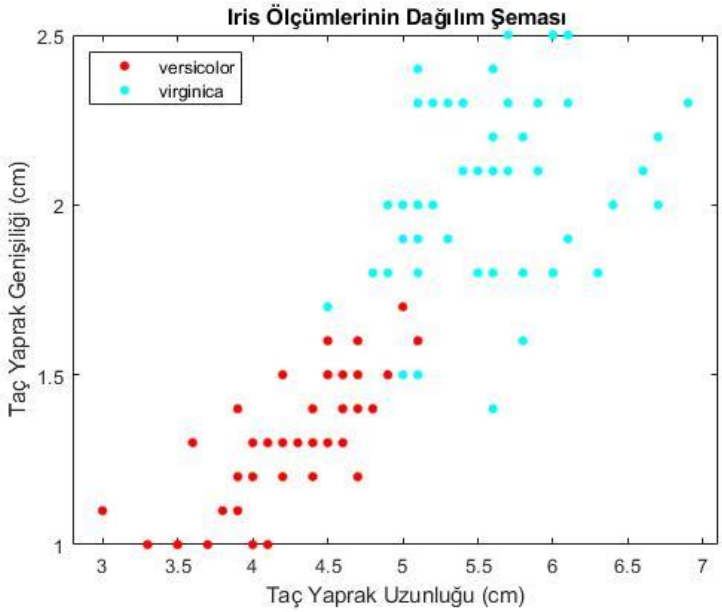
komutu ile de Versicolor ve Virginia sınıflarına ait gözlem verilerinin taç yapraklarına ait uzunluk ve genişlik değerleri X vektörüne atanmıştır.

- Y = species(inds);

komutu ile de, Y vektörüne Versicolor ve Virginia sınıf etiketleri atanmıştır.

Sonrasın da ekranda sınıfların dağılımını göstermek için, aşağıdaki kod bloğu yazılmıştır.

```
figure
gscatter(X(:,1),X(:,2),Y);
h = gca;
lims = [h.XLim h.YLim];
title('{\bf Iris Ölçümlerinin Dağılım Şeması}');
xlabel('Taç Yaprak Uzunluğu (cm)');
ylabel('Taç Yaprak Genişiliği (cm)');
legend('Location','Northwest');
```



Şekil 29: Versicolor ve Virginia Sınıflarına Ait Dağılım

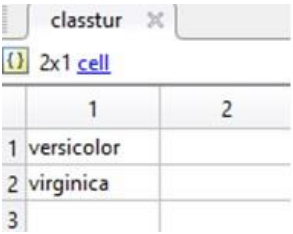
Elde edilen grafik Şekil 29'da verilmiştir. Şekil 29' da görüleceği üzere, Versicolor ve Virginia sınıfları taç yaprak boyutlarına göre sınıflandırılacak olursa, sınıfların iç içe geçmiş olduğu görülmektedir.

fitcSVM fonksiyonuna bu uygulamada herhangi bir parametre girilmeden aşağıdaki şekilde varsayılan ayarları ile kullanılmıştır.

```
DVMModel = fitcSVM(X,Y);
```

Sonrasında, sınıf tür isimleri classtur değişkenine DVMModel.ClassNames objesinden aşağıdaki şekilde aktarılmıştır. Şekil 30'da da classtur değişkeninin içeriği gösterilmektedir.

```
classtur = DVMModel.ClassNames;
```

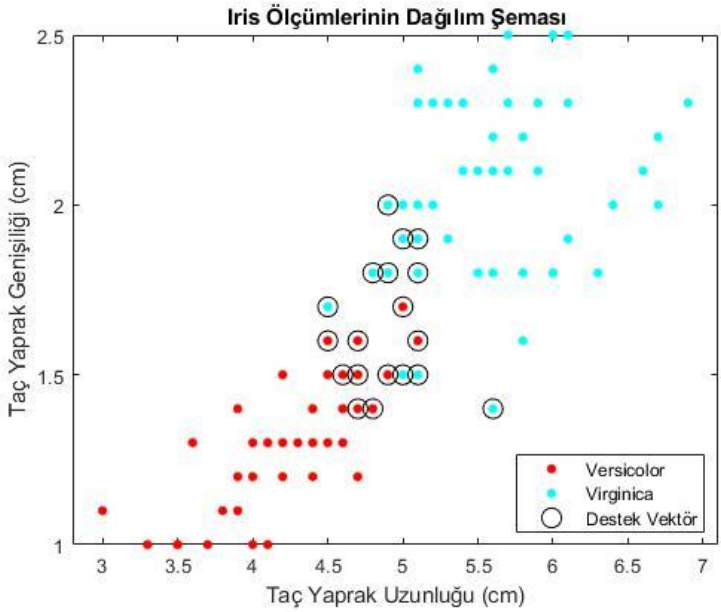


Şekil 30: Classtur Değişkeninin İçeriği

Destek vektör noktalarını elde etmek için, aşağıdaki kod bloğu çalıştırılmaktadır.

```
sv = DVMModel.SupportVectors;
```

Yukarıdaki kod bloğu çalıştırıldığında toplam 24 adet destek vektör noktası elde edilmektedir. Şekil 31'den de anlaşılacağı üzere destek vektör noktaları çember içine alınmıştır.



Şekil 31: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları

Ancak burada, sadece 19 adet destek vektör noktası görülmektedir. Tablo 8'de de görüldüğü üzere, 3 adet destek vektör noktasında iki ve 1 adet destek vektör noktasında da üç adet nokta üst üste çakışmaktadır. Bu nedenle grafikte 19 adet destek vektör noktası gösterilmektedir.

Tablo 8: Destek Vektör Noktaları

Destek Vektör No	Taç Yaprak Uzunluk	Taç Yaprak Genişlik
1	4,5	1,6
2	4,5	1,7
3	4,6	1,5
4	4,7	1,4
5	4,7	1,6
6	4,7	1,4
7	4,7	1,5
8	4,8	1,8
9	4,8	1,4
10	4,8	1,8
11	4,8	1,8
12	4,9	1,5
13	4,9	1,5
14	4,9	2
15	4,9	1,8
16	4,9	1,8
17	5	1,7
18	5	1,5
19	5	1,9
20	5,1	1,6
21	5,1	1,9
22	5,1	1,5
23	5,1	1,8
24	5,6	1,4

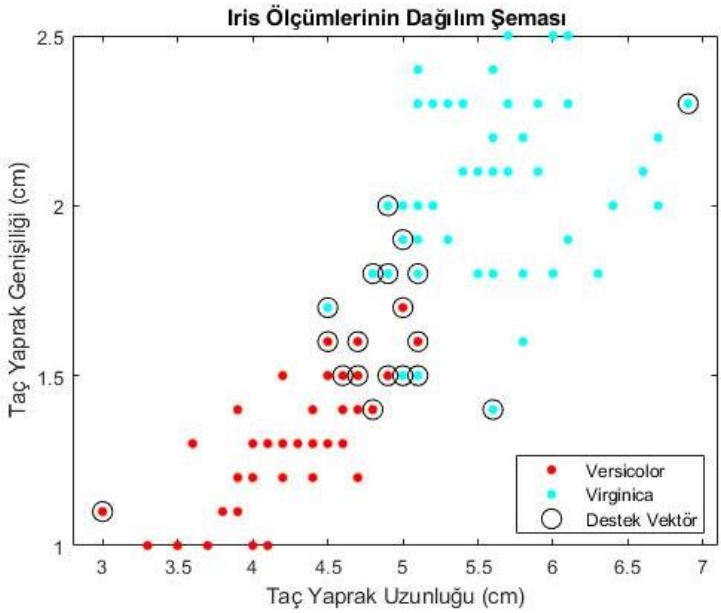
Destek vektör noktalarını grafik üzerinde çizdirmek için aşağıdaki kod parçacığı yazılmaktadır.

```
figure
gscatter(X(:,1),X(:,2),Y)
hold on
plot(sv(:,1),sv(:,2),'ko','MarkerSize',10)
legend('Versicolor','Virginica','Destek Vektör')
title('{\bf Iris Ölçümlerinin Dağılım Şeması}');
xlabel('Taç Yaprak Uzunluğu (cm)');
ylabel('Taç Yaprak Genişiliği (cm)');
hold off
```

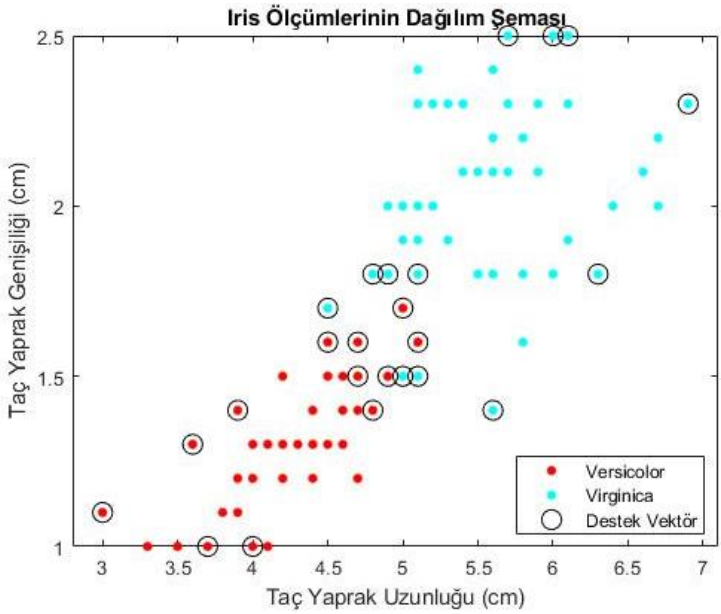
Şekil 31'de kullanılan DVM modelinde varsayılan olarak Gaussian fonksiyonunu kullanılmaktadır. Uygulamada ayrıca farklı parametreler ile RBF ve polynomial fonksiyonları da test edilmiş ve elde edilen destek vektör sayıları da Tablo 9'da verilmiştir. Test edilen fonksiyonların sonuçları Şekil 32, 33, 34 ve 35'de sunulmuştur.

Tablo 9: Test Edilen Fonksiyon ve Parametreler

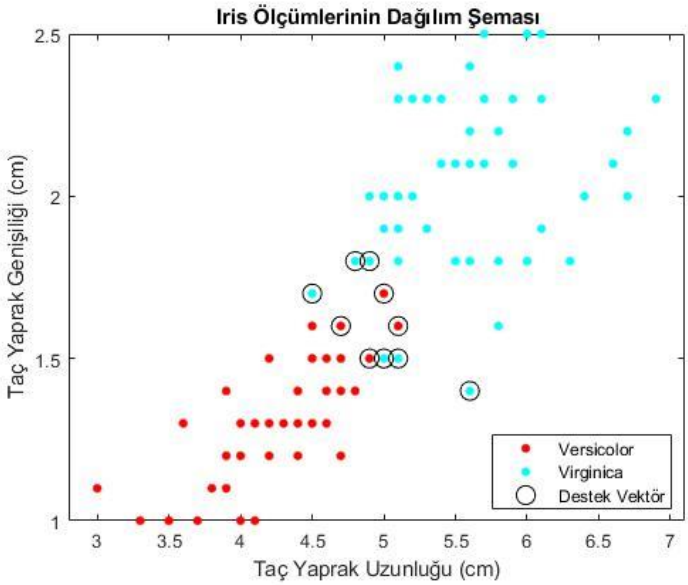
Fonksiyon	**Parametre**	**Destek Vektör Nokta Sayısı**
Gaussian	-	24
RBF	-	23
RBF	`'KernelScale','auto'`	28
Polynomial	`'PolynomialOrder',2`	13
Polynomial	`'PolynomialOrder',4`	12



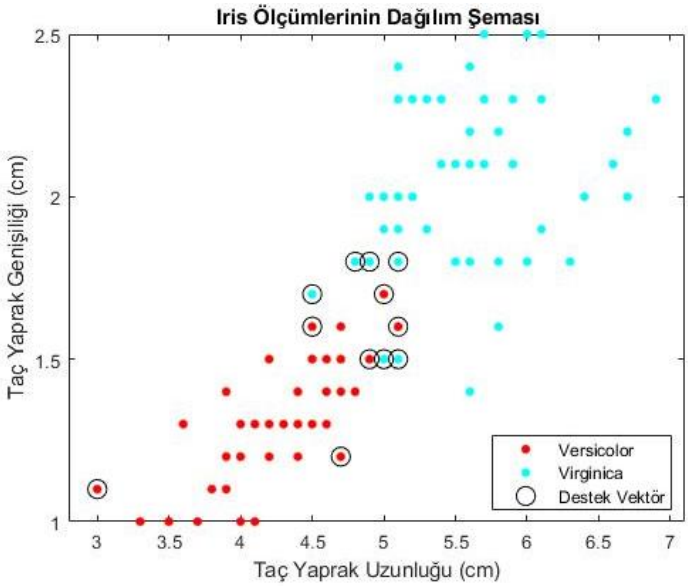
Şekil 32: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (RBF)



Şekil 33: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (RBF-KernelScale, auto)



Şekil 34: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (Polynomial-PolynomialOrder,2)



Şekil 35: Versicolor ve Virginia Sınıflarının Destek Vektör Noktaları (Polynomial-PolynomialOrder,4)

3.5. Uygulama 4

Bu uygulamanın ana gayesi, ikili DVM yapısını kullanarak birçok sınıftan oluşan verilerin destek vektör sınırlarını bulmaktır. Bu amaç doğrultusunda, üç sınıftan oluşan Iris veri seti kullanılmıştır.

3.5.1. Iris Veri Seti

Uygulama bütünlüğünü sağlamak adına, Uygulama 2 ve 3'de kullanılan Iris veri seti bu uygulamada da tercih edilmiştir. Şekil 28'de de gösterildiği üzere Iris veri setindeki veriler üç sınıftan oluşmaktadır.

Bu sınıfları birbirinden ayırabilmek için:

1. Bir gözlemin, bir sınıfın üyesi olup olmadığını gösteren mantıksal bir vektör (indx) oluşturulur.

2. Tahmin verilerini ve indx'i kullanarak bir DVM sınıflandırıcı eğitilir.

3. Sınıflandırıcı, bir hücre dizisinin bir hücresinde saklanır.

3.5.2. Matlab Uygulaması

Bu uygulamada ilk olarak yine Matlab'da yüklü olan fisheriris veri seti çalışma alanına (Workspace) yüklenmektedir (Matlab, 2020).

```
clc;

clear all;

close all;

load fisheriris;

X = meas(:,3:4);

Y = species;
```

- X = meas(:,3:4);

komutu ile tüm sınıfların taç yapraklarına ait uzunluk ve genişlik değerleri X vektörüne atanmaktadır.

- Y = species(inds);

komutu ile de, Y vektörüne Setosa, Versicolor ve Virginia sınıf etiketleri atanmaktadır.

```
DVMModels = cell(3,1);
classes = unique(Y);
rng(1);
```

Veri setinde üç farklı sınıf olmasından dolayı, üç adet DVM modeli oluşturulmaktadır. Bu amaç doğrultusunda,

- DVMModels = cell(3,1);

ifadesi kod satırına yazılmaktadır. Herbir sınıf etiketinin eşsiz olması gerektiğinden dolayı,

- classes = unique(Y);

komutu kullanılmaktadır.

Veri setinde, üç adet farklı sınıf olması nedeniyle, bir döngü içerisinde üç farklı DVM modeli oluşturulmaktadır.

```
for j = 1:numel(classes)
    indx = strcmp(Y,classes(j));
    DVMModels{j} = fitcSVM(X,indx, ...
                                    'ClassNames',[false true], ...
                                    'Standardize',true,...
                                    'KernelFunction','rbf', ...
                                    'BoxConstraint',1);
end
```

- numel(classes)

ifadesi ile classes değişkeni içerisinde bulunan Setosa, Versicolor ve Virginia sınıflarının etiket sayısı olan üç elde edilerek, döngünün üç kez tekrar etmesi sağlanmaktadır.

- indx = strcmp(Y,classes(j));

ifadesi ile, tüm sınıf etiketlerinin bulunduğu Y vektörünün içerisinde, classes vektörünün içerisinde bulunan j. değer aranmakta ve bulunan değerlerin index numaraları indx değişkenine aktarılmaktadır.

- fitcSVM(X,indx, 'ClassNames',[false true], 'Standardize', true,

  'KernelFunction','rbf', 'BoxConstraint',1);

ifadesinde diğer uygulamalardan farklı olarak kullanılan,

'ClassNames',[false true]
ifadesi ile sınıflara sıfır ve bir etiket değerleri verilmektedir.

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
                        min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1);
Scores = zeros(N,numel(classes));
```

Her bir sınıfın gösteriminde, iki boyutlu ızgaralı bir yapı kullanılmaktadır. Bu yapıda ızgara boyutunu, taç yaprağın genişlik ve yükseklik değerlerinin en küçük ve en büyük değerleri belirlenmektedir. Oluşturulan ızgaranın aralık değerleri d değişkenine verilen 0.02 değeri ile belirlenmektedir. Böylece her bir sınıf için xGrid matrisi boyutu kadar değer belirlenmiş olmaktadır. Bunun sonucunda xGrid matrisine herbir sınıftan elde edilen taç yaprakların genişlik ve yükseklik değerleri aktarılabilmektedir.

- N = size(xGrid,1);

ifadesi ile, ızgarada bulunması gereken toplam değer sayısı N değişkenine aktarılmaktadır.

- Scores = zeros(N,numel(classes));

ifadesi ile de, toplam sınıf sayısı ve herbir sınıf için ayrı olmak üzere tüm sınıflar için NxSınıf_Sayısı boyutunda bir ‘sıfır’ matrisi oluşturulmaktadır. Oluşturulan bu matrise sınıf değerleri girilmektedir.

```
for j = 1:numel(classes)
    [~,score] = predict(DVMModels{j},xGrid);
    Scores(:,j) = score(:,2);
end
```

- predict(DVMModels{j},xGrid);

ifadesi ile xGrid matrisindeki her bir değer, yukarıdaki parametreler girilerek oluşturulan üç adet DVM'ye sırasıyla girilmektedir. Elde edilen score değerleri de her bir DVM modeli için Scores matrisine aktarılmaktadır. Burada özellikle;

- score(:,2)

ifadesi ile, predict fonksiyonu ile elde edilen tahmin değerlerinden ikinci sütundaki değer alınmakta ve j. DVM modeli için Scores matrisine atanmaktadır.

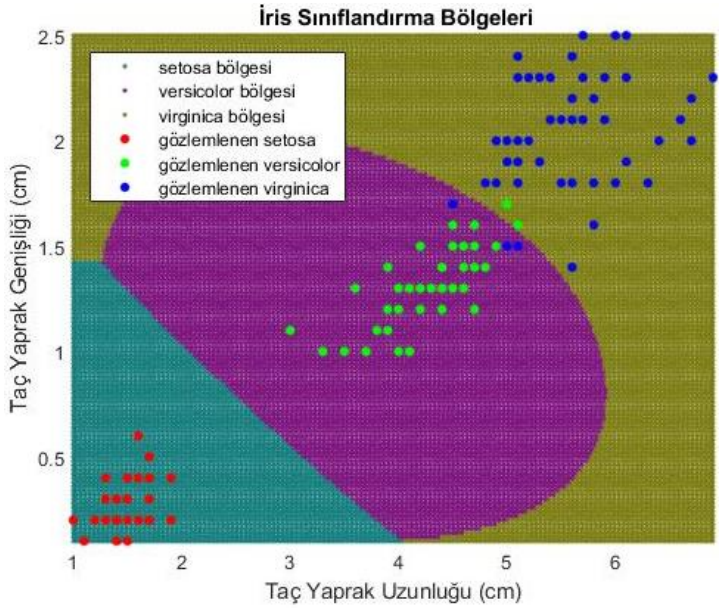
```
[~,maxScore] = max(Scores,[],2);
```

- [~,maxScore] = max(Scores,[],2);

ifadesi ile, üç farklı modelden elde edilen Score matrisindeki herbir satırdaki değerler içerisindeki en büyük değerin bulunduğu sütun, sınıf bilgisi olarak kabul edilmektedir. Bu sınıf bilgisi de maxScore matrisine aktarılmaktadır.

Aşağıdaki kod parçacıkları ile elde edilen değerler de grafiksel olarak Şekil 36'da gösterilmektedir.

```
figure
h(1:3) = gscatter(xGrid(:,1),xGrid(:,2),maxScore,...
            [0.1 0.5 0.5; 0.5 0.1 0.5; 0.5 0.5 0.1]);
hold on
h(4:6) = gscatter(X(:,1),X(:,2),Y);
title('{\bf İris Sınıflandırma Bölgeleri}');
xlabel('Taç Yaprak Uzunluğu (cm)');
ylabel('Taç Yaprak Genişliği (cm)');
legend(h,{'setosa bölgesi','versicolor bölgesi','virginica bölgesi',...
            'gözlemlenen setosa ','gözlemlenen versicolor', ...
            'gözlemlenen virginica'}, 'Location','Northwest');
axis tight
hold off
```



Şekil 36: Iris Veri Setinin Çoklu Sınıflandırılması

3.6. Uygulama 5

Gerçekleştirilen bu uygulamada, özel bir çekirdek kullanılarak DVM'nin eğitimi amaçlanmaktadır. Bu amaç doğrultusunda, uygulamada çekirdek fonksiyon olarak kendi hazırladığımız mysigmoid fonksiyonu tercih edilmiştir.

3.6.1. Veri Seti

Uygulamada kullanılmak üzere, birim çember içinde rastgele bir nokta kümesi oluşturulmaktadır. Bu birim çember üzerindeki birinci ve üçüncü çeyrekteki noktaları pozitif, ikinci ve dördüncü çeyrekteki noktaları ise negatif sınıf etiketi ile etiketlenmektedir.

3.6.2. Matlab Uygulaması

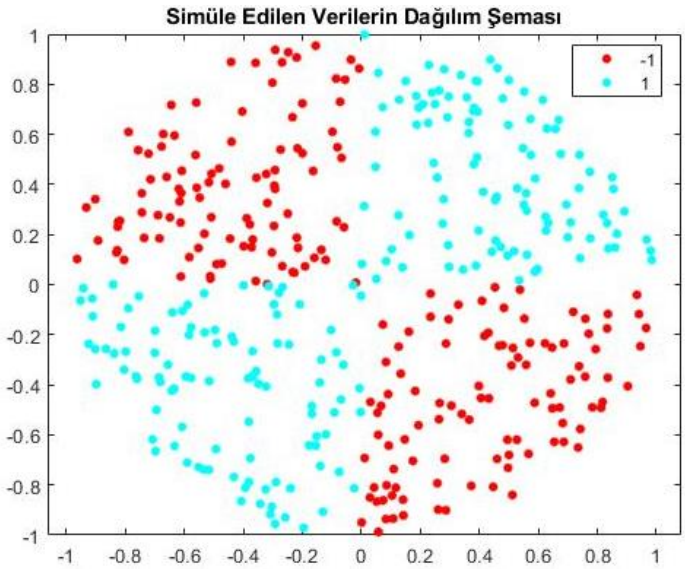
İlk adımda, uygulamada kullanılmak üzere aşağıdaki kod bloğu ile veriler oluşturulmaktadır (Matlab, 2020).

```
clear all;
close all;
clc;
n = 100;                    % Birim çemberin her çeyreğinde bulunan
                            % nokta sayısı
r1 = sqrt(rand(2*n,1));     % Yarıçap ölçülerinin rastgele belirlenmesi
t1 = [pi/2*rand(n,1); (pi/2*rand(n,1)+pi)];
                            % Birim çemberin 1. ve 3. Bölgesi için
                            % rastgele açıların belirlenmesi
X1 = [r1.*cos(t1) r1.*sin(t1)]; % Kutup Kartezyen dönüşümünden X
                                % eksenindeki noktanın orjine olan
                                % uzaklığının bulunması
```

```
r2 = sqrt(rand(2*n,1));
t2 = [pi/2*rand(n,1)+pi/2; (pi/2*rand(n,1)-pi/2)];
                                % Birim çemberin 2. ve 4. Bölgesi için
                                % rastgele açıların belirlenmesi
X2 = [r2.*cos(t2) r2.*sin(t2)];
X = [X1; X2];                   % Örnek noktalar
Y = ones(4*n,1);
Y(2*n + 1:end) = -1;            % Etiketler
```

Veriler hazırlandıktan sonra etiketlenmiş veriler Şekil 37'de gösterilmektedir.

```
figure;
gscatter(X(:,1),X(:,2),Y);
title('Simüle Edilen Verilerin Dağılım Şeması')
```



Şekil 37: Simule Edilen Veriler

Giriş parametresi olarak iki matrisi kabul eden ve içerisinde de sigmoid çekirdek fonksiyonunu kullanarak Gram matrisine dönüştüren bir fonksiyon aşağıdaki kod bloğunda gösterildiği üzere hazırlanmaktadır. Hazırlanan fonksiyon mysigmoid ismi ile kaydedilmektedir.

```
function G = mysigmoid(U,V)
gamma = 1;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

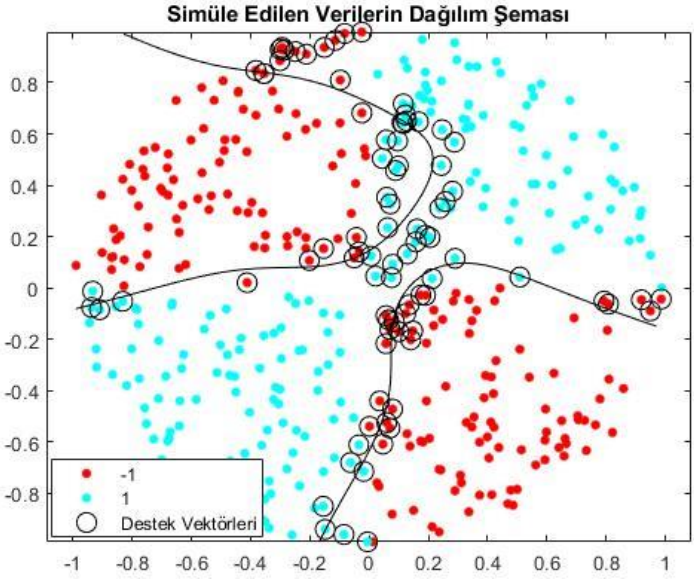
Kullanılan DVM modeli yukarıda tanımlanan mysigmoid çekirdek fonksiyonu kullanılarak eğitilmektedir. Eğitim esnasında kullanılan veriler aşağıdaki kod bloğunda gösterildiği üzere fitcSVM fonksiyonu içerisinde standart hale getirilmiştir.

```
Mdl1 = fitcSVM(X,Y, …
                    'KernelFunction','mysigmoid',…
                    'Standardize',true);
```

Son aşamada destek vektörleri ve karar sınırları Şekil 38'de gösterilmektedir. Bu işlem içinde kullanılan komutlar aşağıdaki kod bloğunda gösterilmektedir.

```
d = 0.02;                                % Izgaranın sıklığı için kullanıldı
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
                                min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];        % Izgara
[~,scores1] = predict(Mdl1,xGrid); % Tahmin skoru
```

```
figure;
h(1:2) = gscatter(X(:,1),X(:,2),Y);
hold on
h(3) = plot(X(Mdl1.IsSupportVector,1),...            % Destek Vektörleri
   X(Mdl1.IsSupportVector,2),'ko','MarkerSize',10);
contour(x1Grid,x2Grid,reshape(scores1(:,2),…        % Karar sınırları
                                    size(x1Grid)),[0 0],'k');
 title('Simüle Edilen Verilerin Dağılım Şeması')
legend({'-1','1','Destek Vektörleri'},'Location','Best');
hold off
```



Şekil 38: Simule Edilmiş Verilerin Destek Vektörleri ile Gösterimi

KAYNAKÇA

Akbulut, S. (2006). Veri madenciliği teknikleri ile bir kozmetik markanın ayrılan müşteri analizi ve müşteri segmentasyonu. *Yüksek Lisans Tezi*.

Alkan, Ö. (2008). Temel bileşenler analizi ve bir uygulama örneği. *Atatürk Üniversitesi, Sosyal Bilimler Enstitüsü, İşletme Anabilimdalı, Yüksek Lisans Tezi, Erzurum, 125s*.

Alma, Ö. G., & Vupa, Ö. (2008). Regresyon analizinde kullanılan en küçük kareler ve en küçük medyan kareler yöntemlerinin karşılaştırılması. *SDÜ Fen Edebiyat Fakültesi Fen Dergisi (E-Dergi)*, *3*(2), 219–229.

Anderson, E. (1936). The species problem in Iris. *Annals of the Missouri Botanical Garden*, *23*(3), 457–509.

Andrew, A. M. (2000). An introduction to support vector machines and other kernel-based learning methods by Nello Christianini and John Shawe-Taylor, Cambridge University Press, Cambridge, 2000. *Robotica*, *18*(6), 687–689.

Baker, K. B., Greenwald, R. A., Villain, J. P., & Wing, S. (1987). *Spectral characteristics of high frequency backscatter from high latitude ionospheric irregularities: A statistical survey*. Tech. Rep. RADC-TR-87.

Baş, N. (2006). Yapay sinir ağları yaklaşımı ve bir uygulama. *Yayınlanmamış Yüksek Lisans Tezi. Mimar Sinan Güzel Sanatlar Üniversitesi Fen Bilimleri Enstitüsü. İstanbul*.

Birbil, İ. (2018). Veri defteri, tahmin ve çıkarım, destek vektör makineleri. Retrieved August 10, 2020, from http://www.veridefteri.com/2018/12/13/tahmin-ve-cikarim-9-destek-vektor-makineleri/

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L., & Cutler, A. (n.d.). Random Forest.

Copeland, M. (2020). What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? Retrieved October 10, 2020, from

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Derman Dildar, A. (2019). *Çoklu bağlantı durumunda yanlı regresyon yöntemlerinin incelenmesi*. Ordu Üniversitesi, Fen Bilimleri Enstitüsü.

Dua, Dheeru and Graff, C. (2017). Machine learning repository. Retrieved October 1, 2020, from University of California, Irvine, School of Information and Computer Sciences website: http://archive.ics.uci.edu/ml

Elmas, Ç. (2003). *Yapay sinir ağları:(kuram, mimari, eğitim, uygulama)* (1. Baskı). Seçkin Yayıncılık.

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, *6*(Dec), 1889–1918.

Fisher, R. A. (1936a). Machine learning repository. Retrieved October 1, 2020, from University of California, Irvine, School of Information and Computer Sciences website: http://archive.ics.uci.edu/ml/datasets/Iris/

Fisher, R. A. (1936b). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188.

Gour, R. (2019). Unique Real-Life Applications of SVM. Retrieved October 24, 2020, from https://medium.com/@rinu.gour123/8-unique-real-life-applications-of-svm-8a96ca43313

Greenwald, R. A., Baker, K. B., Hutchins, R. A., & Hanuise, C. (1985). An HF phased-array radar for studying small-scale structure in the high-latitude ionosphere. *Radio Science*, *20*(1), 63–79.

Hamerly, G., & Elkan, C. (2004). Learning the k in k-means. *Advances in Neural Information Processing Systems*, 281–288.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Haton, J. P., & Haton, M. C. (1989). Yapay Zeka (Çev.: Ayşen Ekmekçi-Alev Türker). *İstanbul: İletişim Yayınları*.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

Karakaya, E. (2011). *Ridge ve Liu tahmincilerinin etkinliklerinin ve yanlılıklarının karşılaştırılması* (Anadolu Üniversitesi). Retrieved from http://193.140.22.72/xmlui/handle/11421/5546

Karaküçük, Y., & Eker, S. (2020). Oftalmolojide yapay zeka ve derin öğrenme uygulamaları. In S. Şahin, Ahmet Rıza;Doğan, Kamil;Sivri (Ed.), *Yapay Zeka ve Derin Öğrenme Uygulamaları* (1st ed., pp. 123–143). Ankara: Akademisyen Kitabevi.

Kavzoğlu, T., & Çölkesen, İ. (2010). Destek vektör makineleri ile uydu görüntülerinin sınıflandırılmasında kernel fonksiyonlarının etkilerinin incelenmesi. *Harita Dergisi*, *144*(7), 73–82.

Kecman, V., Huang, T.-M., & Vogt, M. (2005). Iterative single data algorithm for training kernel machines from huge data sets: Theory and performance. In *Support vector machines: Theory and Applications* (pp. 255–274). Springer.

Kolhe, S. R., & Ranjana, S. Z. (2010). Clustering Iris Data using Supervised and Unsupervised Learning. *International Journal of Computer Science and Application*, (2010), 767–974.

Manning, C. D., Raghavan, P., & Schutze, H. (n.d.). Introduction to infor-mation retrieval? cambridge university press 2008. *Ch*, *20*, 405–416.

Matlab. (2020). Fitcsvm. Retrieved September 8, 2020, from https://www.mathworks.com/help/stats/fitcsvm.html

Mengi, D. F., & Metlek, S. (2020). Türkiye'nin Akdeniz Bölgesine ait rüzgâr ekserjisinin çok katmanli yapay sinir aği ile modellenmesi. *International Journal of Engineering and Innovative Research*, *2*(2), 102–120.

Metlek, S., & Kayaalp, K. (2020). Otistik spektrum bozukluğunun makine öğrenme

algoritmaları ile tespiti. *Zeki Sistemler Teori ve Uygulamaları Dergisi*, *3*(2), 60–68.

Metlek, S., & Özkan, T. (2019). Analysis of perceived service quality and customer satisfaction in the aviation sector with artificial neural networks. *Techno-Science 2nd Internatioanl Conference on Technology and Science*, 583–864. Burdur, Türkiye.

Muratlar, E. R. (2019). Toprağin tarla kapasitesi değişiminin toprak türüne göre temel bileşenler regresyon analizi ile modellenmesi. Retrieved August 11, 2020, from https://www.veribilimiokulu.com/makine-ogrenmesine-cok-degiskenli-istatistiksel-yaklasimlar-temel-bilesenler-analizi

Özkan, K. (2009). Toprağin tarla kapasitesi değişiminin toprak türüne göre temel bileşenler regresyon analizi ile modellenmesi. *Turkish Journal of Forestry; Sayı: 2 (2009); 1-9.*

Pirim, A. G. H. (2006). Yapay zeka. *Journal of Yaşar University*, *1*(1), 81–93.

Rencher, A. C. (2003). *Methods of multivariate analysis* (Second Edi). John Wiley & Sons Inc.

Sigillito, V. G., Wing, S. P., Hutton, L. V, & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, *10*(3), 262–266.

Souza, C. R. (2010). Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, *3*, 29.

Tan, P.-N., Steinbach, M., & Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.

Vapnik, V. (1998). The support vector method of function estimation. In *Nonlinear Modeling* (pp. 55–85). Springer.

Walker, A. D. M., Greenwald, R. A., & Baker, K. B. (1987). Determination of the fluctuation level of ionospheric irregularities from radar backscatter measurements. *Radio Science*, *22*(05), 689–705.

ÖZGEÇMİŞLER

Dr. Öğr. Üyesi. Sedat METLEK

Dr. Sedat METLEK, Lisans eğitimini Marmara Üniversitesi Elektronik ve Bilgisayar eğitiminde, Yüksek Lisans eğitimini Süleyman Demirel Üniversitesi Elektronik Bilgisayar Eğitimi bölümünde ve Doktora eğitimini de Süleyman Demirel Üniversitesi Elektronik Haberleşme Mühendisliğinde tamamlamıştır. Halen Burdur Mehmet Akif Ersoy Üniversitesinde Dr. Öğretim Üyesi olarak çalışmaktadır. Yazarın Derin Öğrenme, Yapay Zekâ, Örüntü İşleme ve Makine Otomasyonu konularında birçok akademik ve ticari çalışması bulunmaktadır.

Dr. Öğr. Üyesi Kıyas KAYAALP

Dr. Kıyas KAYAALP, Lisans eğitimini Marmara Üniversitesi Elektronik ve Bilgisayar eğitiminde ve Süleyman Demirel Üniversitesi Bilgisayar Mühendisliği bölümünde, Yüksek Lisans eğitimini Süleyman Demirel Üniversitesi Elektronik Bilgisayar Eğitimi bölümünde ve Doktora eğitimini de Süleyman Demirel Üniversitesi Fizik bölümünde tamamlamıştır. Halen Isparta Uygulamalı Bilimler Üniversitesinde Dr. Öğretim Üyesi olarak çalışmaktadır. Yazarın, Derin Öğrenme, Veri Madenciliği, Yapay Zekâ, Nesnelerin İnterneti konularında, ulusal ve uluslararası düzeyde bildirileri, makaleleri ve kitapları bulunmaktadır.

ISBN: 978-625-7279-74-1