# Makine Öğrenimi Algoritmalarında Ustalaşın

## Nasıl Çalıştıklarını Keşfedin ve Sıfırdan Uygulayın

Jason Brownlee

MACHINE LEARNING MASTERY

Jason Brownlee

# Makine Öğrenimi Algoritmalarında Ustalaşın

## Nasıl Çalıştıklarını Keşfedin ve Sıfırdan Uygulayın

**Makine Öğrenimi Algoritmalarında Ustalaşın**



Baskı, v1.1
http://MachineLearningMastery.com

# İçindekiler

# Önsöz

Makine öğrenimi algoritmaları uygulamalı makine öğrenimine hakimdir. Algoritmalar makine öğreniminin çok büyük bir parçası olduğu için onlara aşina olmak ve nasıl çalıştıklarını gerçekten anlamak için zaman harcamalısınız. Bu kitabı, bu yolculuğa başlamanıza yardımcı olmak için yazdım.

Makine öğrenimi algoritmalarını istatistik, olasılık ve lineer cebir kullanarak tanımlayabilirsiniz. Matematiksel tanımlamalar çok kesin ve genellikle nettir. Ancak makine öğrenimi algoritmalarını tanımlamanın tek yolu bu değildir. Bu kitabı yazarken, makine öğrenimi algoritmalarını geliştiriciler (benim gibi) için tanımlamak üzere yola çıktım. Geliştiriciler olarak tekrarlanabilir prosedürlerle düşünürüz. Bizim için bir makine öğrenimi algoritmasını tanımlamanın en iyi yolu şudur:

1. Algoritma tarafından kullanılan temsil açısından (bir dosyada saklanan gerçek sayılar).
2. Algoritma tarafından verilerden bir model öğrenmek ve daha sonra modelle tahminler yapmak için kullanılan soyut tekrarlanabilir prosedürler açısından.
3. Gerçek sayıların denklemlere tam olarak nasıl girdiğini ve çıktı olarak hangi sayıların bekleneceğini gösteren net çalışma örnekleri ile.

Bu kitap, makine öğrenimi algoritmalarıyla ilgili matematiksel konuşmaları kesip atıyor ve size tam olarak nasıl çalıştıklarını gösteriyor, böylece bunları bir elektronik tabloda, en sevdiğiniz programlama diliyle kodda veya istediğiniz şekilde kendiniz uygulayabilirsiniz. Bu samimi bilgiye bir kez sahip olduğunuzda, her zaman yanınızda olacaktır. Algoritmaları tekrar tekrar uygulayabilirsiniz. Daha da önemlisi, bir algoritmanın davranışını altta yatan prosedüre geri çevirebilir ve neler olup bittiğini ve ondan en iyi şekilde nasıl yararlanacağınızı gerçekten bilebilirsiniz.

Bu kitap sizin makine öğrenimi algoritmaları turunuz ve ben de tur rehberiniz olmaktan heyecan ve onur duyuyorum. Hadi başlayalım.

Jason Brownlee
Melbourne,
Avustralya
2016

# Bölüm I Giriş

# Bölüm 1 Hoş geldiniz

*Makine Öğrenimi Algoritmalarında Ustalaşmaya* hoş geldiniz. Bu kitap size sıfırdan 10 güçlü makine öğrenimi algoritması öğretecek.

Geliştiriciler en iyi algoritma açıklamaları ve pratik örneklerin bir karışımı ile öğrenirler. Bu kitap, geliştiricilere makine öğrenimi algoritmalarını öğretmek için özenle tasarlanmıştır. Yapı, hem makine öğrenimi algoritmalarının prosedürel açıklamalarını hem de sayıları çeşitli denklemlere tam olarak nasıl ekleyeceğinizi ve diğer tarafta tam olarak hangi sayıları bekleyeceğinizi gösteren adım adım öğreticileri içerir. Bu kitap, hiçbir şeyin gizli kalmaması için makine öğrenimi algoritmalarının üzerindeki perdeyi sizin için kaldırmak üzere yazılmıştır. Bu kitaptaki algoritma açıklamalarını ve öğreticileri okuduktan sonra şunları yapabileceksiniz:

1. En iyi makine öğrenimi algoritmalarının nasıl çalıştığını anlayın ve açıklayın.
2. Algoritma prototiplerini tercih ettiğiniz dilde veya araçta uygulayın.

Bu kitap, makine öğrenimi algoritmalarının iç kısımlarına yapacağınız rehberli bir turdur.

## 1.1 İzleyici

Bu kitap geliştiriciler için yazılmıştır. İstatistik, olasılık veya lineer cebir konusunda bir geçmişe sahip olduğunuzu varsaymaz. Biraz istatistik ve olasılık biliyorsanız, ortalamalar, standart sapmalar ve Gauss dağılımları gibi kavramlardan bahsedeceğimiz için yardımcı olabilir. Paslanmışsanız veya emin değilseniz endişelenmeyin, hepsini bir araya getirebilmek için denklemlere ve çalışılmış örneklere sahip olacaksınız.

Bu kitap ayrıca makine öğrenimi konusunda bir geçmişe sahip olduğunuzu varsaymaz. Genel hatları bilmeniz yardımcı olur, ancak bu kitabın amacı size makine öğrenimi algoritmalarını sıfırdan öğretmektir. Özellikle, tahmine dayalı modelleme olarak adlandırılan yeni veriler üzerinde tahminler yapmak için modeller oluşturduğumuz makine öğrenimi türü ile ilgileniyoruz. Bu sizin için yeniyse endişelenmeyin, yakında makine öğrenimi algoritmalarının türlerinin ayrıntılarına gireceğiz.

Son olarak, bu kitap kodlamayı bildiğinizi veya iyi kodladığınızı varsaymaz. Tüm örnekleri bir elektronik tabloda takip edebilirsiniz. Aslında bir hesap tablosunda takip etmeniz şiddetle tavsiye edilir. Eğer bir programcıysanız, öğrenme sürecinin bir parçası olarak örnekleri favori programlama dilinize de aktarabilirsiniz.

## 1.2 Algoritma Açıklamalar

Bu kitaptaki algoritmaların tanımı ve sunumu dikkatlice tasarlanmıştır. Her algoritma üç temel özellik açısından tanımlanmıştır:

1. Bir dosyada saklanabilecek gerçek sayılar ve yapı açısından algoritma tarafından kullanılan temsil.
2. Algoritma tarafından eğitim verilerinden öğrenmek için kullanılan prosedür.
3. Algoritma tarafından öğrenilen bir modele göre tahminler yapmak için kullanılan prosedür.

Bu kitapta çok az matematik kullanılacaktır. Dahil edilen denklemler, bir fikri aktarmanın en iyi yolu oldukları için dahil edilmiştir. Mümkün olduğunda, her denklem metinsel olarak da açıklanacak ve tam olarak nasıl kullanılacağını göstermek için çalışılmış bir örnek verilecektir.

Son olarak ve en önemlisi, bu kitapta açıklanan her algoritma adım adım bir öğretici içerecektir. Bu sayede öğrenme ve tahmin prosedürlerinin gerçek sayılarla nasıl çalıştığını tam olarak görebilirsiniz. Her öğretici, bir elektronik tabloda veya seçtiğiniz bir programlama dilinde takip edebilmeniz için yeterli ayrıntıda sağlanmıştır. Bu, ham girdi verilerini ve her denklemin çıktısını, tüm kanlı hassasiyet dahil olmak üzere içerir. Hiçbir şey gizlenmemiş veya saklanmamıştır. Her şeyi göreceksiniz.

## 1.3 Kitap Yapısı

Bu kitap dört bölüme ayrılmıştır:

1. Makine öğrenimi algoritmaları hakkında arka plan.
2. Doğrusal makine öğrenimi algoritmaları.
3. Doğrusal olmayan makine öğrenimi algoritmaları.
4. Makine öğrenimi algoritmaları topluluğu.

Beş bölümün her birine daha yakından bakalım:

### 1.3.1 Algoritmalar Arka Planı

Bu bölüm size makine öğrenimi algoritmaları konusunda bir temel sağlayacaktır. Size tüm makine öğrenimi algoritmalarının nasıl bağlantılı olduğunu ve aynı temel sorunu çözmeye çalıştığını öğretecektir. Bu size herhangi bir makine öğrenimi algoritmasını anlayabilmeniz için gerekli bağlamı sağlayacaktır. Keşfedeceksiniz:

Makine öğreniminde verileri tanımlarken kullanılan terminoloji.

Tüm makine öğrenimi algoritmaları tarafından çözülen problemi anlamak için çerçeve.

Parametrik ve parametrik olmayan algoritmalar arasındaki önemli farklar.

Denetimli, denetimsiz ve yarı denetimli makine öğrenimi problemleri arasındaki kontrast.

Yanlılık ve varyanstan kaynaklanan hatalar, bu kaygılar arasındaki dengeyi bozmaktadır.

Verilere aşırı uyum sağlama sorununun üstesinden gelmek için uygulamalı makine öğreniminde savaş.

### 1.3.2 Doğrusal Algoritmalar

Bu bölüm, daha basit doğrusal algoritmalarla başlayarak makine öğrenimi algoritmalarına giriş yapmanızı kolaylaştıracaktır. Bunlar basit algoritmalar olabilir, ancak aynı zamanda daha güçlü teknikleri anlamak için önemli bir temel oluştururlar. Aşağıdaki doğrusal algoritmaları keşfedeceksiniz:

Birçok makine öğrenimi algoritmasının kalbinde kullanılabilen gradyan inişi optimizasyon prosedürü.

Gerçek değerleri tahmin etmek için doğrusal regresyon, iki öğretici ile gerçekten battığından emin olmak için.

İki kategorili problemlerde sınıflandırma için lojistik regresyon.

İkiden fazla kategoriye sahip problemlerde sınıflandırma için doğrusal diskriminant analizi.

### 1.3.3 Doğrusal Olmayan Algoritmalar

Bu bölümde doğrusal algoritmaların üzerine inşa edilen daha güçlü doğrusal olmayan makine öğrenimi algoritmaları tanıtılacaktır. Bunlar, probleminiz hakkında daha az varsayımda bulunan ve çok çeşitli problem türlerini öğrenebilen tekniklerdir. Ancak bu gücün dikkatli kullanılması gerekir çünkü çok iyi öğrenebilir ve eğitim verilerinize aşırı uyum sağlayabilirler. Aşağıdaki doğrusal olmayan algoritmaları keşfedeceksiniz:

Sınıflandırma ve regresyon ağaçları temel karar ağacı algoritması.

Naive Bayes, bu tekniğin kullanılabileceği yararlı yolları gösteren iki öğretici ile sınıflandırma için olasılığı kullanır.

Veri setiniz dışında herhangi bir model gerektirmeyen K-En Yakın Komşular.

Eğitim veri kümenizin boyutunu küçültmeyi öğrenerek K-En Yakın Komşuları genişleten Vektör Nicelemeyi Öğrenme.

Destek vektör makineleri belki de en popüler ve en güçlü algoritmalardan biridir.

### 1.3.4 Topluluk Algoritmaları

Makine öğrenimi algoritmasının güçlü ve daha gelişmiş bir türü de topluluk algoritmalarıdır. Bunlar, daha doğru tahminler sağlamak için birden fazla modelden gelen tahminleri birleştiren tekniklerdir. Bu bölümde en çok kullanılan iki topluluk yöntemini tanıyacaksınız:

Mevcut en güçlü algoritmalar arasında yer alan Bagging ve Random Forests.

Boosting topluluğu ve daha zayıf modellerin tahminlerini art arda düzelten AdaBoost algoritması.

## 1.4 Bu Kitap Ne Değil

Bu bir makine öğrenimi ders kitabı değildir. İşlerin neden çalıştığının arkasındaki teoriye veya denklemlerin türetilmesine girmeyeceğiz. Bu kitap, makine öğrenimi algoritmalarının neden çalıştığını değil, nasıl çalıştığını öğretmekle ilgilidir.

Bu bir makine öğrenimi programlama kitabı değildir. Üretim veya operasyonel kullanım için makine öğrenimi algoritmaları tasarlamayacağız. Bu kitaptaki tüm örnekler sadece gösterim amaçlıdır.

## 1.5 Bu Kitabı En İyi Nasıl Kullanılır?

Bu kitap bir uçtan diğerine doğrusal olarak okunmak üzere tasarlanmıştır. Bu kitabı okumak yeterli değildir. Kavramların kalıcı olmasını sağlamak ve makine öğrenimi algoritmalarını gerçekten öğrenmek için öğreticiler üzerinde çalışmanız gerekir. Kitabın yanında bir elektronik tablo açarsanız ve her bir öğretici üzerinde çalışırsanız bu kitaptan en iyi şekilde yararlanacaksınız.

Eğitimler boyunca çalışmak, her algoritma için açıklanan temsil, öğrenme ve tahmin prosedürlerine bağlam kazandıracaktır. Buradan, fikirleri kendi programlarınıza ve bu algoritmaları pratikte kullanımınıza aktarabilirsiniz.

Öğrendiklerinizi hemen uygulayabilmeniz için günde bir bölümü, ideal olarak akşamları bilgisayar başında tamamlamanızı öneririm. Günden güne kaldığınız yerden devam edebilmeniz için temel denklemleri ve açıklamaları bilinçli olarak tekrarladım.

## 1.6 Özet

Nihayet makine öğrenimini anlamanın zamanı geldi. Bu kitap makine öğrenimi algoritmalarına giriş biletiniz. Bundan sonra, tüm makine öğrenimi algoritmalarının çözmeye çalıştığı temel sorunu anlamak için bir temel oluşturacaksınız.

# Bölüm II Arka Plan

# Bölüm 2

# Makine Öğreniminde Veri Hakkında Nasıl Konuşulur?

Veri, makine öğreniminde büyük bir rol oynar. Veriler hakkında konuşurken doğru terminolojiyi anlamak ve kullanmak önemlidir. Bu bölümde makine öğreniminde verilerin tam olarak nasıl tanımlanacağını ve veri hakkında nasıl konuşulacağını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Genel olarak elektronik veri tabloları hakkında konuşurken kullanılan standart veri terminolojisi.

İstatistikte kullanılan veri terminolojisi ve makine öğrenimine istatistiksel bakış.

Makine öğreniminin bilgisayar bilimi perspektifinde kullanılan veri terminolojisi.

Bu, genel olarak makine öğrenimi algoritmalarını anlamanıza büyük ölçüde yardımcı olacaktır. Hadi başlayalım.

## 2.1 Bildiğiniz Gibi Veri It

Veriler hakkında nasıl düşünürsünüz? Bir elektronik tablo düşünün. Sütunlarınız, satırlarınız ve hücreleriniz var.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | Column 1 | Column 2 | Column 3 |
| 2 | Row 1 | 2.2 | 2.3 | 1 |
| 3 | Row 2 | 2.3 | 2.6 | 0 |
| 4 | Row 3 | 2.1 | 2 | 1 |
| 5 | | | | |

Şekil 2.1: Makine Öğreniminde Veri Terminolojisi.

**Sütun**: Sütun, tek bir türdeki verileri tanımlar. Örneğin, ağırlıklar, yükseklikler veya fiyatlardan oluşan bir sütununuz olabilir. Bir sütundaki tüm veriler aynı ölçeğe sahip olacak ve birbirlerine göre anlam taşıyacaktır.

**Satır**: Bir satır tek bir varlık veya gözlemi, sütunlar ise bu varlık veya gözlemle ilgili özellikleri tanımlar. Ne kadar çok satırınız varsa, problem alanından o kadar çok

örneğiniz vardır.

**Hücre**: Hücre, bir satır ve sütundaki tek bir değerdir. Gerçek bir değer (1,5) bir tamsayı olabilir (2) veya bir kategori (*kırmızı* ).

Veriler, sütunlar, satırlar ve hücreler hakkında muhtemelen bu şekilde düşünüyorsunuz. Genel olarak bu tür verilere tablosal veriler diyebiliriz. Bu tür verilerle makine öğreniminde çalışmak kolaydır. Makine öğrenimi alanında farklı bakış açıları sunan farklı yaklaşımlar vardır. Örneğin, istatistiksel bakış açısı ve bilgisayar bilimi bakış açısı vardır. Daha sonra, bildiğiniz gibi veriye atıfta bulunmak için kullanılan farklı terimlere bakacağız.

## 2.2 İstatistiksel Öğrenme Perspektif

İstatistiksel bakış açısı, verileri makine öğrenimi algoritmasının öğrenmeye çalıştığı varsayımsal bir fonksiyon ($f$ ) bağlamında çerçeveler. Yani, bazı girdi değişkenleri (*girdi*) verildiğinde, tahmin edilen çıktı değişkeni (*çıktı*) nedir?

$$Çıkış = f(Giriş) \tag{2.1}$$

Girdi olan sütunlar girdi değişkenleri olarak adlandırılır. Her zaman sahip olamayabileceğiniz ve gelecekte yeni girdi verileri için tahmin etmek istediğiniz veri sütununa ise çıktı değişkeni denir. Buna yanıt değişkeni de denir.

$$Çıktı\ Değişkenleri = f(Girdi\ Değişkenleri) \tag{2.2}$$

| | A | B | C |
|---|---|---|---|
| 1 | X1 | X2 | Y |
| 2 | 2.2 | 2.3 | 1 |
| 3 | 2.3 | 2.6 | 0 |
| 4 | 2.1 | 2 | 1 |
| 5 | | | |

Şekil 2.2: Makine Öğreniminde Verilerin İstatistiksel Öğrenme Perspektifi.

Tipik olarak, birden fazla girdi değişkeniniz vardır. Bu durumda girdi değişkenleri grubu girdi vektörü olarak adlandırılır.

$$Çıktı\ Vektörü = f(Girdi\ Vektörü) \tag{2.3}$$

Geçmişinizde biraz istatistik yaptıysanız, daha geleneksel bir terminoloji biliyor olabilirsiniz. Örneğin, bir istatistik metni girdi değişkenlerinden bağımsız değişkenler, çıktı değişkeninden ise bağımlı değişken olarak bahsedebilir. Bunun nedeni, tahmin probleminin ifadesinde çıktının bağımlı ya da girdi veya bağımsız değişkenlerin bir fonksiyonu olmasıdır.

$$Bağımlı\ Değişken = f(Bağımsız\ Değişkenler) \tag{2.4}$$

Veriler, denklemlerde ve makine öğrenimi algoritmalarının açıklamalarında kısa bir el kullanılarak tanımlanır. İstatistiksel perspektifte kullanılan standart kısaltma, girdi değişkenlerine sermaye *x* (*X)* ve çıktı değişkenlerine sermaye *y* (Y ) olarak atıfta bulunmaktır.

$$Y = f(X) \tag{2.5}$$

Birden fazla girdi değişkeniniz olduğunda, girdi vektöründeki sıralamalarını belirtmek için bir tamsayı ile referansları kaldırılabilir, örneğin ilk üç sütundaki veriler için *X1*, X2 ve X3.

## 2.3 Bilgisayar Bilimi Perspektif

Bilgisayar bilimleri terminolojisinde veri ile istatistiksel perspektif arasında pek çok örtüşme vardır. Temel farklılıklara bakacağız. Bir satır genellikle bir varlığı (bir kişi gibi) veya bir varlıkla ilgili bir gözlemi tanımlar. Bu nedenle, bir satırın sütunları genellikle gözlemin nitelikleri olarak adlandırılır. Bir sorunu modellerken ve tahminlerde bulunurken, girdi özniteliklerine ve çıktı öznitelik lerine başvurabiliriz.

$$OutputAttribute = Program(InputAttributes) \tag{2.6}$$

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | Attribute 1 | Attribute 2 | Output Attribute |
| 2 | Instance 1 | 2.2 | 2.3 | 1 |
| 3 | Instance 2 | 2.3 | 2.6 | 0 |
| 4 | Instance 3 | 2.1 | 2 | 1 |
| 5 | | | | |

Şekil 2.3: Makine Öğreniminde Verinin Bilgisayar Bilimleri Perspektifi.

Sütunların bir diğer adı da özelliktir ve öznitelikle aynı nedenle kullanılır; burada bir özellik gözlemin bazı özelliklerini tanımlar. Bu, bir gözlem oluşturmak için ham verilerden özelliklerin çıkarılması gereken verilerle çalışırken daha yaygındır. Buna örnek olarak görüntü, ses ve video gibi analog veriler verilebilir.

$$Çıktı = Program(InputFeatures) \tag{2.7}$$

Bir başka bilgisayar bilimi ifadesi de bir veri satırı veya bir gözlemin örnek olarak kullanılmasıdır. Bunun kullanılmasının nedeni, bir satırın problem alanı tarafından gözlemlenen veya üretilen verilerin tek bir örneği veya tek bir örneği olarak kabul edilebilmesidir.

$$Tahmin = Program(Örnek) \tag{2.8}$$

## 2.4 Modeller ve Algoritmalar

Önemli olan son bir açıklama notu vardır ve bu da algoritmalar ve modeller arasındadır. Hem algoritma hem de model birbirinin yerine kullanılabildiği için bu durum kafa karıştırıcı olabilir.

A

Benim sevdiğim bakış açısı, modeli verilerden öğrenilen belirli bir temsil olarak ve algoritmayı da bunu öğrenme süreci olarak düşünmektir.

$$Model = Algoritma(Veri) \tag{2.9}$$

Örneğin, bir karar ağacı veya bir katsayılar kümesi bir modeldir ve C5.0 ve En Küçük Kareler Doğrusal Regresyon, bu ilgili modelleri öğrenmek için kullanılan algoritmalardır.

## 2.5 Özet

Bu bölümde makine öğreniminde verileri tanımlamak için kullanılan temel terminolojiyi keşfettiniz.

Bir elektronik tabloda sütunlar, satırlar ve hücreler olarak görülen standart tablo veri anlayışıyla başladınız.

$X$ olarak gösterilebilecek girdi ve çıktı değişkenlerinin istatistiksel terimlerini öğrendiniz. ve $sY$ sırasıyla.

Nitelik, özellik ve örnek gibi bilgisayar bilimi terimlerini öğrendiniz.

Son olarak, modellerden ve algoritmalardan bahsetmenin öğrenilen temsil ve öğrenme süreci olarak ayrılabileceğini öğrendiniz.

Artık makine öğreniminde veri hakkında nasıl konuşulacağını biliyorsunuz. Bir sonraki bölümde tüm makine öğrenimi algoritmalarının altında yatan paradigmayı keşfedeceksiniz.

# Bölüm 3

# Algoritmalar Girdiden Çıktıya Bir Eşleme Öğrenir

Makine öğrenimi algoritmaları nasıl çalışır? Tahmine dayalı modelleme için tüm denetimli makine öğrenimi algoritmalarının altında yatan ortak bir ilke vardır. Bu bölümde, tüm algoritmaların altında yatan ortak prensibi anlayarak makine öğrenimi algoritmalarının gerçekte nasıl çalıştığını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Tüm denetimli makine öğrenimi algoritmalarının çözmeyi amaçladığı eşleme problemi.

Makine öğreniminin tahmin yapmaya odaklanan alt alanına tahmine dayalı modelleme denir.

Farklı makine öğrenimi algoritmaları, eşleme fonksiyonunu öğrenmek için farklı stratejileri temsil eder.

Hadi başlayalım.

## 3.1 İşlevini Öğrenme

Makine öğrenimi algoritmaları, girdi değişkenlerini ($X$) bir çıktı değişkenine ($Y$) en iyi şekilde eşleyen bir hedef fonksiyonun ($f$) öğrenilmesi olarak tanımlanır.

$$Y = f(X) \tag{3.1}$$

Bu, girdi değişkenlerinin ($X$) yeni örnekleri verildiğinde gelecekte ($Y$) tahminler yapmak istediğimiz genel bir öğrenme görevidir. Fonksiyonun ($f$) neye benzediğini veya formunu bilmiyoruz. Bilseydik, onu doğrudan kullanırdık ve makine öğrenimi algoritmalarını kullanarak verilerden öğrenmemize gerek kalmazdı. Bu düşündüğünüzden daha zor. Ayrıca girdi verisinden ($X$) bağımsız olan bir hata (e) vardır.

$$Y = f(X) + e \tag{3.2}$$

Bu hata, *X*'ten *Y'ye* en iyi eşlemeyi yeterince karakterize etmek için yeterli özniteliğe sahip olmamak gibi bir hata olabilir. Bu hataya indirgenemez hata denir çünkü hedef fonksiyonu ($f$) tahmin etmede ne kadar iyi olursak olalım, bu hatayı azaltamayız. Yani, verilerden bir fonksiyon öğrenme problemi zor bir problemdir ve makine öğrenimi alanının ve makine öğrenimi algoritmalarının var olma sebebi de budur.

## 3.2 Tahminleri Yapmak İçin Bir Fonksiyon Öğrenme

En yaygın makine öğrenimi türü, yeni $X$ için $Y$ tahminleri yapmak üzere $Y = f(X)$ eşlemesini öğrenmektir. Buna tahmine dayalı modelleme veya tahmine dayalı analitik denir ve amacımız mümkün olan en doğru tahminleri yapmaktır.

Bu nedenle, öğrenmekte olduğumuz fonksiyonun ($f$) şekli ve biçimi ile gerçekten ilgilenmiyoruz, sadece doğru tahminler yapması ile ilgileniyoruz. Verilerdeki ilişki hakkında daha fazla bilgi edinmek için $Y = f(X)$ eşlemesini öğrenebiliriz ve buna istatistiksel çıkarım denir. Amaç bu olsaydı, daha basit yöntemler kullanır ve öğrenilen modeli ve ($f$)'nin biçimini anlamaya doğru tahminler yapmaktan daha fazla değer verirdik.

Bir fonksiyonu ($f$) öğrendiğimizde, elimizdeki verilerden onun biçimini tahmin ederiz. Bu nedenle, bu tahminde hata olacaktır. Uygulamalı makine öğreniminde zamanın çoğu, altta yatan işlevin tahminini iyileştirmeye ve terim olarak model tarafından yapılan tahminlerin performansını iyileştirmeye çalışmak için harcanır.

## 3.3 Fonksiyonunu Öğrenme Teknikleri

Makine öğrenimi algoritmaları, girdi değişkenleri ($X$) verildiğinde çıktı değişkenini ($Y$) tahmin etmek için hedef fonksiyonu ($f$) tahmin etme teknikleridir. Farklı temsiller, öğrenilen fonksiyonun biçimi hakkında, örneğin doğrusal mı yoksa doğrusal olmayan mı olduğu gibi farklı varsayımlarda bulunur.

Farklı makine öğrenimi algoritmaları, fonksiyonun şekli ve yapısı ve buna yaklaşmak için bir temsilin en iyi nasıl optimize edileceği hakkında farklı varsayımlarda bulunur. Bir makine öğrenimi problemi üzerinde farklı algoritmalar denemek bu yüzden çok önemlidir, çünkü hangi yaklaşımın yaklaştırmaya çalıştığımız temel fonksiyonun yapısını tahmin etmede en iyi olacağını önceden bilemeyiz.

## 3.4 Özet

Bu bölümde, tahmine dayalı modelleme için tüm makine öğrenimi algoritmalarının amacını açıklayan temel ilkeyi keşfettiniz.

Makine öğrenimi algoritmalarının, girdi değişkenleri ($X$) verildiğinde çıktı değişkenlerinin ($Y$) eşleme fonksiyonunu ($f$) veya $Y = f(X)$ tahmin etmek için çalıştığını öğrendiniz.

Ayrıca, farklı makine öğrenimi algoritmalarının temel fonksiyonun biçimi hakkında farklı varsayımlarda bulunduğunu öğrendiniz.

Hedef fonksiyonun biçimi hakkında fazla bir şey bilmediğimizde, neyin en iyi sonucu verdiğini görmek için bir dizi farklı algoritma denememiz gerekir.

Artık tüm makine öğrenimi algoritmalarının altında yatan prensibi biliyorsunuz. Bir sonraki bölümde makine öğrenimi algoritmalarının iki ana sınıfını keşfedeceksiniz: parametrik ve parametrik olmayan algoritmalar.

# Bölüm 4

# Parametrik ve Parametrik Olmayan Makine Öğrenimi Algoritmaları

Parametrik makine öğrenimi algoritması nedir ve parametrik olmayan makine öğrenimi algoritmasından farkı nedir? Bu bölümde parametrik ve parametrik olmayan makine öğrenimi algoritmaları arasındaki farkı keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Bu parametrik makine öğrenimi algoritmaları sadece bilinen bir fonksiyonel forma eşleme yapar.

Parametrik olmayan algoritmaların girdilerden çıktılara herhangi bir eşlemeyi öğrenebilmesi.

Tüm algoritmaların parametrik veya parametrik olmayan gruplar halinde organize

edilebileceği. Hadi başlayalım.

## 4.1 Parametrik Makine Öğrenimi Algoritmalar

Varsayımlar öğrenme sürecini büyük ölçüde basitleştirebilir, ancak öğrenilebilecekleri de sınırlayabilir. Fonksiyonu bilinen bir forma basitleştiren algoritmalara parametrik makine öğrenimi algoritmaları denir.

> Verileri sabit büyüklükte (eğitim örneklerinin sayısından bağımsız) bir dizi parametre ile özetleyen bir öğrenme modeline parametrik model denir. Parametrik bir modele ne kadar veri atarsanız atın, kaç parametreye ihtiyacı olduğu konusunda fikrini değiştirmeyecektir.

- Yapay Zeka: A Modern Approach, sayfa 737

Algoritmalar iki adımdan oluşmaktadır:

1. İşlev için bir form seçin.
2. Eğitim verilerinden fonksiyon için katsayıları öğrenin.

Eşleme fonksiyonu için anlaşılması kolay bir fonksiyonel form, doğrusal regresyonda kullanıldığı gibi bir çizgidir:

$$B0 + B1 \times X1 + B2 \times X2 = 0 \quad (4.1)$$

Burada *B0*, *B1* ve *B2* kesişme ve eğimi kontrol eden doğrunun katsayıları, X1 ve X2 ise iki girdi değişkenidir. Bir doğrunun fonksiyonel formunu varsaymak öğrenme sürecini büyük ölçüde basitleştirir. Şimdi tek yapmamız gereken doğru denkleminin katsayılarını tahmin etmek ve problem için bir tahmin modelimiz var.

Genellikle varsayılan fonksiyonel form, girdi değişkenlerinin doğrusal bir kombinasyonudur ve bu nedenle parametrik makine öğrenimi algoritmaları genellikle *doğrusal makine öğrenimi algoritmaları* olarak da adlandırılır. Sorun şu ki, altta yatan gerçek bilinmeyen fonksiyon bir doğru gibi doğrusal bir fonksiyon olmayabilir. Neredeyse bir doğru olabilir ve doğru çalışması için giriş verilerinde bazı küçük dönüşümler gerektirebilir. Ya da doğruya hiç benzemeyebilir, bu durumda varsayım yanlıştır ve yaklaşım kötü sonuçlar üretecektir.

Parametrik makine öğrenimi algoritmalarına bazı örnekler daha verilebilir:

Lojistik Regresyon

Doğrusal Diskriminant Analizi

Perceptron

Parametrik Makine Öğrenimi Algoritmalarının Faydaları:

**Daha basit**: Bu yöntemlerin anlaşılması ve sonuçların yorumlanması daha kolaydır.

**Hız**: Parametrik modellerin verilerden öğrenmesi çok hızlıdır.

**Daha Az Veri**: Çok fazla eğitim verisi gerektirmezler ve verilere uyum mükemmel olmasa bile iyi çalışabilirler.

Parametrik Makine Öğrenimi Algoritmalarının Sınırlamaları:

**Kısıtlı**: Bir fonksiyonel form seçerek bu yöntemler belirtilen formla yüksek oranda kısıtlanır.

**Sınırlı Karmaşıklık**: Yöntemler daha basit problemler için daha uygundur.

**Zayıf Uyum**: Pratikte yöntemlerin altta yatan eşleme fonksiyonuna uyması pek olası değildir.

## 4.2 Parametrik Olmayan Makine Öğrenimi Algoritmalar

Eşleme fonksiyonunun biçimi hakkında güçlü varsayımlarda bulunmayan algoritmalara parametrik olmayan makine öğrenimi algoritmaları denir. Varsayım yapmadıkları için, eğitim verilerinden herhangi bir fonksiyonel formu öğrenmekte özgürdürler.

Parametrik olmayan yöntemler, çok fazla veriniz olduğunda ve ön bilginiz olmadığında ve sadece doğru özellikleri seçme konusunda çok fazla endişelenmek istemediğinizde iyidir.

- Yapay Zeka: Modern Bir Yaklaşım, sayfa 757

Parametrik olmayan yöntemler, eşleme fonksiyonunu oluştururken eğitim verilerine en iyi şekilde uymaya çalışırken, görülmeyen verilere genelleme yapma yeteneğini de korur. Bu nedenle, çok sayıda fonksiyonel forma uyum sağlayabilirler. Anlaşılması kolay bir parametrik olmayan model, yeni bir veri örneği için en benzer k eğitim modeline dayalı tahminler yapan k-en yakın komşu algoritmasıdır. Bu yöntem, birbirine yakın örüntülerin benzer bir çıktı değişkenine sahip olma olasılığı dışında eşleme fonksiyonunun biçimi hakkında herhangi bir varsayımda bulunmaz. Parametrik olmayan popüler makine öğrenimi algoritmalarına bazı örnekler daha verilebilir:

CART ve C4.5 gibi Karar Ağaçları

Naive Bayes

Destek Vektör Makineleri

Sinir Ağları

Parametrik Olmayan Makine Öğrenimi Algoritmalarının Faydaları:

**Esneklik**: Çok sayıda fonksiyonel forma uyum sağlayabilir.

**Güç**: Temel işlev hakkında varsayım yok (veya zayıf varsayımlar).

**Performans**: Tahmin için daha yüksek performanslı modellerle

sonuçlanabilir. Parametrik Olmayan Makine Öğrenimi Algoritmalarının

Sınırlamaları:

**Daha fazla veri**: Eşleme fonksiyonunu tahmin etmek için çok daha fazla eğitim verisi gerekir.

**Daha yavaş**: Genellikle eğitilecek çok daha fazla parametreye sahip olduklarından eğitilmeleri çok daha yavaştır.

**Aşırı uyum**: Eğitim verilerine aşırı uyum sağlama riski daha fazladır ve belirli tahminlerin neden yapıldığını açıklamak daha zordur.

## 4.3 Özet

Bu bölümde parametrik ve parametrik olmayan makine öğrenimi algoritmaları arasındaki farkı keşfettiniz.

Parametrik yöntemlerin girdi değişkenlerinin çıktı değişkeniyle eşleştirilmesi konusunda büyük varsayımlarda bulunduğunu ve bu nedenle eğitilmelerinin daha hızlı olduğunu, daha az veri gerektirdiğini ancak o kadar güçlü olmayabileceğini öğrendiniz.

Ayrıca parametrik olmayan yöntemlerin hedef fonksiyon hakkında çok az varsayımda bulunduğunu veya hiç varsayımda bulunmadığını ve bunun sonucunda çok daha fazla veri gerektirdiğini, eğitilmesinin daha yavaş olduğunu ve model karmaşıklığının daha yüksek olduğunu ancak daha güçlü modellerle sonuçlanabileceğini öğrendiniz.

Artık parametrik ve parametrik olmayan makine öğrenimi algoritmaları arasındaki farkı biliyorsunuz. Bir sonraki bölümde makine öğrenimi algoritmalarını öğrenme şekillerine göre gruplandırmanın başka bir yolunu keşfedeceksiniz: denetimli ve denetimsiz öğrenme.

# Bölüm 5

# Gözetimli, Gözetimsiz ve Yarı Gözetimli Öğrenme

Denetimli makine öğrenimi nedir ve denetimsiz makine öğrenimi ile nasıl bir ilişkisi vardır? Bu bölümde denetimli öğrenmeyi, denetimsiz öğrenmeyi ve yarı denetimli öğrenmeyi keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Sınıflandırma ve regresyon denetimli öğrenme problemleri hakkında.

Kümeleme ve ilişkilendirme denetimsiz öğrenme problemleri hakkında.

Denetimli ve denetimsiz problemler için kullanılan örnek algoritmalar.

Yarı denetimli öğrenme olarak adlandırılan denetimli ve denetimsiz öğrenme arasında yer alan bir problem. Hadi başlayalım.

## 5.1 Denetimli Makine Öğrenme

Pratik makine öğreniminin çoğunluğu denetimli öğrenmeyi kullanır. Denetimli öğrenme, girdi değişkenleriniz ($X$) ve bir çıktı değişkeniniz ($Y$ ) olduğu ve girdiden çıktıya eşleme işlevini öğrenmek için bir algoritma kullandığınız yerdir.

$$Y = f(X) \tag{5.1}$$

Amaç, eşleme fonksiyonuna o kadar iyi yaklaşmaktır ki, yeni girdi verilerine sahip olduğunuzda ($X$) bu veriler için çıktı değişkenlerini ($Y$ ) tahmin edebilirsiniz. Buna denetimli öğrenme denir çünkü bir algoritmanın eğitim veri setinden öğrenme süreci, öğrenme sürecini denetleyen bir öğretmen olarak düşünülebilir. Doğru cevapları biliyoruz, algoritma eğitim verileri üzerinde yinelemeli olarak tahminler yapar ve öğretmen tarafından düzeltilir. Algoritma kabul edilebilir bir performans seviyesine ulaştığında öğrenme durur. Denetimli öğrenme problemleri ayrıca regresyon ve sınıflandırma problemleri olarak gruplandırılabilir.

**Sınıflandırma**: Bir sınıflandırma problemi, çıktı değişkeninin *kırmızı* veya *mavi* ya da *hastalık* var veya *yok* gibi bir kategori olduğu durumdur.

**Regresyon**: Bir regresyon problemi, çıktı değişkeninin gerçek bir değer olduğu durumdur, örneğin *dolar* veya *ağırlık*.

Sınıflandırma ve regresyon üzerine inşa edilen bazı yaygın problem türleri, sırasıyla tavsiye ve zaman serisi tahminini içerir.

Denetimli makine öğrenimi algoritmalarının bazı popüler örnekleri şunlardır:

Regresyon problemleri için doğrusal regresyon.

Sınıflandırma ve regresyon problemleri için rastgele orman.

Sınıflandırma problemleri için destek vektör makineleri.

## 5.2 Denetimsiz Makine Öğrenme

Denetimsiz öğrenme, yalnızca girdi verilerine ($X$) sahip olduğunuz ve karşılık gelen çıktı değişkenlerinin olmadığı durumdur. Denetimsiz öğrenmenin amacı, veriler hakkında daha fazla bilgi edinmek için verilerin altında yatan yapıyı veya dağılımı modellemektir.

Bunlara denetimsiz öğrenme denir çünkü yukarıdaki denetimli öğrenmeden farklı olarak doğru cevaplar yoktur ve öğretmen yoktur. Algoritmalar, verilerdeki ilginç yapıyı keşfetmek ve sunmak için kendi tasarımlarına bırakılır. Denetimsiz öğrenme problemleri ayrıca kümeleme ve ilişkilendirme problemleri olarak gruplandırılabilir.

**Kümeleme**: Bir kümeleme problemi, müşterileri satın alma davranışlarına göre gruplandırmak gibi verilerdeki doğal gruplamaları keşfetmek istediğiniz durumdur.

**Birliktelik**: Bir birliktelik kuralı öğrenme problemi, $A$ satın alan kişilerin $B$ satın alma eğiliminde olması gibi verilerinizin büyük bölümünü tanımlayan kuralları keşfetmek istediğiniz durumdur.

Denetimsiz öğrenme algoritmalarının bazı popüler örnekleri şunlardır:

kümeleme problemleri için k-ortalamalar.

Birliktelik kuralı öğrenme problemleri için Apriori algoritması.

## 5.3 Yarı Denetimli Makine Öğrenme

Büyük miktarda girdi verisine ($X$) sahip olduğunuz ve verilerin yalnızca bir kısmının etiketlendiği ($Y$ ) problemlere yarı denetimli öğrenme problemleri denir. Bu problemler hem denetimli hem de denetimsiz öğrenme arasında yer alır. Sadece bazı görüntülerin etiketlendiği (örneğin köpek, kedi, insan) ve çoğunluğun etiketsiz olduğu bir fotoğraf arşivi buna iyi bir örnektir. Birçok gerçek dünya makine öğrenimi problemi bu alana girer. Bunun nedeni, alan uzmanlarına erişim gerektirebileceğinden verileri etiketlemenin pahalı veya zaman alıcı olabilmesidir. Oysa etiketsiz verilerin toplanması ve saklanması ucuz ve kolaydır.

Giriş değişkenlerindeki yapıyı keşfetmek ve öğrenmek için denetimsiz öğrenme tekniklerini kullanabilirsiniz. Ayrıca etiketsiz veriler için en iyi tahmin tahminlerini yapmak için denetimli öğrenme tekniklerini kullanabilir, bu verileri eğitim verileri olarak denetimli öğrenme algoritmasına geri besleyebilir ve modeli yeni görünmeyen veriler üzerinde tahminler yapmak için kullanabilirsiniz.

## 5.4 Özet

Bu bölümde denetimli, denetimsiz ve yarı denetimli öğrenme arasındaki farkı öğrendiniz. Artık şunu biliyorsunuz:

Denetimli: Tüm veriler etiketlenir ve algoritmalar girdi verilerinden çıktıyı tahmin etmeyi öğrenir.

Denetimsiz: Tüm veriler etiketsizdir ve algoritmalar girdi verilerinden içsel yapıyı öğrenir.

Yarı denetimli: Bazı veriler etiketlidir ancak çoğu etiketsizdir ve denetimli ve denetimsiz tekniklerin bir karışımı kullanılabilir.

Artık makine öğrenimi algoritmalarını denetimli, denetimsiz ve yarı denetimli öğrenme olarak gruplandırabileceğinizi biliyorsunuz. Bir sonraki bölümde, verilerden öğrenirken en büyük iki hata kaynağını, yani önyargı ve varyansı ve bu iki endişe arasındaki gerilimi keşfedeceksiniz.

# Bölüm 6

# Önyargı-Varyans Ödünleşimi

Denetimli makine öğrenimi algoritmaları en iyi şekilde yanlılık-varyans değiş tokuşu merceğinden anlaşılabilir. Bu bölümde Bias-Variance Trade-Off'u keşfedecek ve makine öğrenimi algoritmalarını daha iyi anlamak ve verileriniz üzerinde daha iyi performans elde etmek için nasıl kullanacağınızı öğreneceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz.

Tüm öğrenme hatalarının önyargı veya varyans hatası olarak ayrıştırılabileceği.

Bu önyargı, problemin çözümünü kolaylaştırmak için algoritma tarafından yapılan basitleştirici varsayımları ifade eder.

Bu varyans, bir modelin eğitim verilerindeki değişikliklere olan duyarlılığını ifade eder.

Tahmin modeline yönelik tüm uygulamalı makine öğrenimi en iyi şekilde önyargı ve varyans çerçevesinde anlaşılabilir.

Hadi başlayalım.

## 6.1 Önyargıya Genel Bakış ve Varyans

Denetimli makine öğreniminde bir algoritma eğitim verilerinden bir model öğrenir. Herhangi bir denetimli makine öğrenimi algoritmasının amacı, girdi verileri ($X$) göz önüne alındığında çıktı değişkeni ($Y$) için eşleme fonksiyonunu ($f$) en iyi şekilde tahmin etmektir. Eşleme fonksiyonu genellikle hedef fonksiyon olarak adlandırılır çünkü belirli bir denetimli makine öğrenimi algoritmasının yaklaşmayı amaçladığı fonksiyondur. Herhangi bir makine öğrenimi algoritması için tahmin hatası üç bölüme ayrılabilir:

Önyargı Hatası

Varyans Hatası

İndirgenemez Hata

İndirgenemez hata, hangi algoritma kullanılırsa kullanılsın azaltılamaz. Problemin seçilen çerçevesinden kaynaklanan hatadır ve girdi değişkenlerinin çıktı değişkenine eşlenmesini etkileyen bilinmeyen değişkenler gibi faktörlerden kaynaklanabilir. Bu bölümde makine öğrenimi algoritmalarımızla etkileyebileceğimiz iki kısma odaklanacağız. Yanlılık hatası ve varyans hatası.

## 6.2 Bias Hata

Yanlılık, hedef fonksiyonun öğrenilmesini kolaylaştırmak için bir model tarafından yapılan basitleştirici varsayımlardır. Genel olarak parametrik algoritmalar yüksek önyargıya sahiptir, bu da onları hızlı öğrenilir ve anlaşılması daha kolay hale getirir ancak genellikle daha az esnektir. Buna karşılık, algoritmaların önyargılarının basitleştirici varsayımlarını karşılayamayan karmaşık problemler üzerinde daha düşük tahmin performansına sahiptirler.

**Düşük Yanlılık**: Hedef fonksiyonun biçimi hakkında daha fazla varsayım önerir.

**High-Bias**: Hedef fonksiyonun biçimi hakkında daha az varsayım önerir.

Düşük önyargılı makine öğrenimi algoritmalarına örnek olarak şunlar verilebilir: Karar Ağaçları, k-En Yakın Komşu- bors ve Destek Vektör Makineleri. Yüksek önyargılı makine öğrenimi algoritmalarına örnek olarak şunlar verilebilir: Doğrusal Regresyon, Doğrusal Diskriminant Analizi ve Lojistik R e g r e s y o n .

## 6.3 Varyans Hata

Varyans, farklı eğitim verileri kullanıldığında hedef fonksiyonun tahmininin değişeceği miktardır. Hedef fonksiyon bir makine öğrenimi algoritması tarafından eğitim verilerinden tahmin edilir, bu nedenle algoritmanın bir miktar varyansa sahip olmasını beklemeliyiz. İdeal olarak, bir eğitim veri setinden diğerine çok fazla değişmemelidir, bu da algoritmanın girdiler ve çıktı değişkenleri arasındaki gizli altta yatan eşlemeyi seçmede iyi olduğu anlamına gelir. Yüksek varyansa sahip makine öğrenimi algoritmaları, eğitim verilerinin özelliklerinden büyük ölçüde etkilenir. Bu, eğitimin özelliklerinin eşleme fonksiyonunu karakterize etmek için kullanılan parametrelerin sayısını ve türlerini etkilediği anlamına gelir.

**Düşük Varyans**: Eğitim veri setindeki değişikliklerle hedef fonksiyonun tahmininde küçük değişiklikler önerir.

**Yüksek Varyans**: Eğitim veri setindeki değişikliklerle hedef fonksiyonun tahmininde büyük değişiklikler o l d u ğ u n u gösterir.

Genellikle çok fazla esnekliğe sahip parametrik olmayan makine öğrenimi algoritmaları yüksek bir yanlılığa sahiptir. Örneğin karar ağaçları yüksek bir yanlılığa sahiptir ve ağaçlar kullanılmadan önce budanmazsa bu daha da y ü k s e k t i r . Düşük varyanslı makine öğrenimi algoritmalarına örnek olarak şunlar verilebilir: Doğrusal Regresyon, Doğrusal Diskriminant Analizi ve Lojistik Regresyon. Yüksek varyanslı makine öğrenimi algoritmalarına örnek olarak şunlar verilebilir: Karar Ağaçları, k-En Yakın Komşular ve Destek Vektör Makineleri.

## 6.4 Önyargı-Varyans Takası- Kapalı

Herhangi bir denetimli makine öğrenimi algoritmasının amacı düşük yanlılık ve düşük varyans elde etmektir. Bunun karşılığında algoritma iyi bir tahmin performansı elde etmelidir. Yukarıdaki örneklerde genel bir eğilim görebilirsiniz:

Parametrik veya doğrusal makine öğrenimi algoritmaları genellikle yüksek bir önyargıya ancak düşük bir varyansa sahiptir.

Parametrik olmayan veya doğrusal olmayan makine öğrenimi algoritmaları genellikle düşük bir önyargıya ancak yüksek bir varyansa sahiptir.

Makine öğrenimi algoritmalarının parametrelendirilmesi genellikle önyargı ve varyansı dengelemek için verilen bir mücadeledir. Aşağıda, belirli algoritmalar için yanlılık-varyans dengesinin yapılandırılmasına ilişkin iki örnek verilmiştir:

K-en yakın komşu algoritması düşük yanlılığa ve yüksek varyansa sahiptir, ancak bu denge, k değerini artırarak değiştirilebilir, bu da tahmine katkıda bulunan komşu sayısını artırır ve sonuç olarak modelin yanlılığını artırır.

Destek vektör makinesi algoritması düşük yanlılığa ve yüksek varyansa sahiptir, ancak eğitim verilerinde izin verilen marjın ihlal sayısını etkileyen C parametresini artırarak değiş tokuş değiştirilebilir, bu da yanlılığı artırır ancak varyansı azaltır.

Makine öğreniminde önyargı ve varyans arasındaki ilişkiden kaçış yoktur.

Sapmanın artırılması varyansı azaltacaktır.

Varyansın artırılması yanlılığı azaltacaktır.

Bu iki kaygı arasında bir denge vardır ve seçtiğiniz algoritmalar ve bunları yapılandırma şekliniz, probleminiz için bu dengede farklı dengeler bulmaktadır. Gerçekte gerçek yanlılık ve varyans hata terimlerini hesaplayamayız çünkü altta yatan asıl hedef fonksiyonu bilmiyoruz. Bununla birlikte, bir çerçeve olarak önyargı ve varyans, tahmin performansı arayışında makine öğrenimi algoritmalarının davranışını anlamak için araçlar sağlar.

## 6.5 Özet

Bu bölümde makine öğrenimi algoritmaları için önyargı, varyans ve önyargı-varyans değiş tokuşunu keşfettiniz. Artık şunu biliyorsunuz:

Önyargı, hedef fonksiyonun daha kolay tahmin edilebilmesi için model tarafından yapılan basitleştirici varsayımlardır.

Varyans, farklı eğitim verileri verildiğinde hedef fonksiyonun tahmininin değişeceği miktardır.

Değiş tokuş, yanlılığın getirdiği hata ile varyans arasındaki gerilimdir.

Verilerden öğrenirken iki hata kaynağı olan önyargı ve varyansı biliyorsunuz. Bir sonraki bölümde, makine öğrenimini problemlere uygularken yanlılık ve varyansın pratik sonuçlarını, yani aşırı uyum ve yetersiz uyumu keşfedeceksiniz.

# Bölüm 7

# Aşırı Uyum ve Yetersiz Uyum

Makine öğreniminde düşük performansın nedeni ya aşırı uyum ya da veriye yetersiz uyumdur. Bu bölümde makine öğreniminde genelleme kavramını ve bununla birlikte gelen aşırı uyum ve yetersiz uyum sorunlarını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Bu aşırı uyum, yeni verilere iyi genelleme yapamama pahasına eğitim verilerini çok iyi öğrenmek anlamına gelir.

Bu yetersiz uyum, eğitim verilerinden problemi yeterince öğrenememek anlamına gelir.

Bu aşırı uyum, uygulamada en sık karşılaşılan sorundur ve yeniden örnekleme yöntemleri ve geri tutulan bir doğrulama veri seti kullanılarak ele alınabilir.

Hadi başlayalım.

## 7.1 Makinede Genelleştirme Öğrenme

Makine öğreniminde, hedef fonksiyonun eğitim verilerinden öğrenilmesini tümevarımsal öğrenme olarak tanımlarız. Tümevarım, belirli örneklerden genel kavramları öğrenmeyi ifade eder ki bu da tam olarak denetimli makine öğrenimi problemlerinin çözmeyi amaçladığı sorundur. Bu, tam tersi olan ve genel kurallardan belirli kavramları öğrenmeye çalışan tümdengelimden farklıdır.

Genelleme, bir makine öğrenimi modeli tarafından öğrenilen kavramların, modelin öğrenirken görmediği belirli örneklere ne kadar iyi uygulandığını ifade eder. İyi bir makine öğrenimi modelinin amacı, eğitim verilerinden problem alanındaki herhangi bir veriye iyi bir şekilde genelleme yapmaktır. Bu, gelecekte modelin hiç görmediği veriler üzerinde tahminler yapmamızı sağlar. Bir makine öğrenimi modelinin yeni verileri ne kadar iyi öğrendiği ve genelleştirdiği hakkında konuşurken makine öğreniminde kullanılan bir terminoloji vardır, yani aşırı uyum ve yetersiz uyum. Aşırı uyum ve yetersiz uyum, makine öğrenimi algoritmalarının düşük performans göstermesinin en büyük iki nedenidir.

## 7.2 İstatistiksel Uyum

İstatistikte uyum, bir hedef fonksiyona ne kadar iyi yaklaştığınızı ifade eder. Bu, makine öğreniminde kullanmak için iyi bir terminolojidir, çünkü denetimli makine öğrenimi algoritmaları, girdi değişkenleri göz önüne alındığında çıktı değişkenleri için bilinmeyen temel eşleme işlevine yaklaşmaya çalışır.

İstatistikler genellikle, fonksiyon yaklaşımının hedef fonksiyonla ne kadar iyi eşleştiğini tahmin etmek için kullanılan ölçüleri ifade eden uyum iyiliğini tanımlar. Bu yöntemlerden bazıları makine öğreniminde kullanışlıdır (örneğin artık hataların hesaplanması), ancak bu tekniklerden bazıları yaklaştırdığımız hedef fonksiyonun biçimini bildiğimizi varsayar, ki makine öğreniminde durum böyle değildir. Hedef fonksiyonun formunu bilseydik, gürültülü eğitim verisi örneklerinden bir yaklaşım öğrenmeye çalışmak yerine, tahminler yapmak için doğrudan kullanırdık.

## 7.3 Makinede Aşırı Uyum Öğrenme

Aşırı uyum, eğitim verilerini çok iyi modelleyen bir model anlamına gelir. Aşırı uyum, bir modelin eğitim verilerindeki ayrıntıları ve gürültüyü, modelin yeni veriler üzerindeki performansını olumsuz etkileyecek ölçüde öğrenmesi durumunda ortaya çıkar. Bu, eğitim verilerindeki gürültü veya rastgele dalgalanmaların model tarafından kavramlar olarak alındığı ve öğrenildiği anlamına gelir. Sorun, bu kavramların yeni veriler için geçerli olmaması ve modellerin genelleme yeteneğini olumsuz etkilemesidir. Bir hedef fonksiyonu öğrenirken daha fazla esnekliğe sahip olan parametrik olmayan ve doğrusal olmayan modellerde aşırı uyum daha olasıdır. Bu nedenle, birçok parametrik olmayan makine öğrenimi algoritması, modelin ne kadar ayrıntı öğreneceğini sınırlamak ve kısıtlamak için parametreler veya teknikler de içerir. Örneğin, karar ağaçları çok esnek olan ve eğitim verilerine aşırı uyuma tabi olan parametrik olmayan bir makine öğrenimi algoritmasıdır. Bu sorun, bir ağaç budanarak ele alınabilir öğrendikten sonra, aldığı bazı ayrıntıları ortadan kaldırmak için.

## 7.4 Makinede Yetersiz Uyum Öğrenme

Yetersiz uyum, eğitim verilerini modelleyemeyen ve yeni verilere genelleme yapamayan bir modeli ifade eder. Yetersiz uyum gösteren bir makine öğrenimi modeli uygun bir model değildir ve eğitim verileri üzerinde düşük performans göstereceği açıktır. İyi bir performans ölçütü verildiğinde tespit edilmesi kolay olduğu için yetersiz uyum genellikle tartışılmaz. Çözüm, yola devam etmek ve alternatif makine öğrenimi algoritmalarını denemektir. Bununla birlikte, aşırı uyum kavramı sorununa iyi bir karşıtlık sağlar.

## 7.5 Makinede İyi Bir Uyum Öğrenme

İdeal olarak, yetersiz uyum ile aşırı uyum arasındaki tatlı noktada bir model seçmek istersiniz. Amaç budur, ancak pratikte bunu yapmak çok zordur.

Bu hedefi anlamak için, bir makine öğrenimi algoritmasının bir eğitim verisini öğrenirken zaman içindeki performansına bakabiliriz. Hem eğitim verileri üzerindeki beceriyi hem de eğitim sürecinden geri tuttuğumuz bir test veri kümesi üzerindeki beceriyi çizebiliriz. Zaman içinde, algoritma öğrendikçe, eğitim verisi üzerindeki modelin hatası azalır ve test veri kümesindeki hata da azalır. Çok uzun süre eğitim yaparsak, eğitim veri kümesindeki performans düşmeye devam edebilir çünkü model aşırı uyum sağlar ve eğitim veri kümesindeki alakasız ayrıntıları ve gürültüyü öğrenir. Aynı zamanda, modelin genelleme yeteneği azaldıkça test kümesi için hata tekrar yükselmeye başlar.

Tatlı nokta, test veri kümesindeki hatanın artmaya başlamasından hemen önce, modelin hem eğitim veri kümesinde hem de görünmeyen test veri kümesinde iyi beceriye sahip olduğu noktadır. Bu deneyi favori makine öğrenimi algoritmalarınızla gerçekleştirebilirsiniz. Bu genellikle

Uygulamada faydalı bir tekniktir, çünkü test veri kümesindeki beceriyi kullanarak eğitim için durma noktasını seçmek, test kümesinin artık *görülmediği* veya bağımsız bir nesnel ölçüt olmadığı anlamına gelir. Bu verilerle ilgili bazı bilgiler (çok sayıda faydalı bilgi) eğitim prosedürüne sızmıştır. Uygulamada tatlı noktayı bulmaya yardımcı olmak için kullanabileceğiniz iki ek teknik vardır: yeniden örnekleme yöntemleri ve bir doğrulama veri kümesi.

## 7.6 Aşırı Uyum Nasıl Sınırlandırılır

Hem aşırı uyum hem de yetersiz uyum kötü model performansına yol açabilir. Ancak uygulamalı makine öğreniminde açık ara en yaygın sorun aşırı uyumdur. Aşırı uyum böyle bir sorundur çünkü makine öğrenimi algoritmalarının eğitim verileri üzerindeki değerlendirmesi, aslında en çok önem verdiğimiz değerlendirmeden, yani algoritmanın görünmeyen veriler üzerinde ne kadar iyi performans gösterdiğinden farklıdır. Makine öğrenimi algoritmalarını değerlendirirken aşırı uyumu sınırlamak için kullanabileceğiniz iki önemli teknik vardır:

1. Model doğruluğunu tahmin etmek için bir yeniden örnekleme tekniği kullanın.
2. Bir doğrulama veri kümesini geride tutun.

En popüler yeniden örnekleme tekniği k-kat çapraz doğrulamadır. Modelinizi eğitim verilerinin farklı alt kümeleri üzerinde k kez eğitmenize ve test etmenize ve bir makine öğrenimi modelinin görünmeyen veriler üzerindeki performansına ilişkin bir tahmin oluşturmanıza olanak tanır.

Doğrulama veri kümesi, projenizin sonuna kadar makine öğrenimi algoritmalarınızdan uzak tuttuğunuz eğitim verilerinizin bir alt kümesidir. Makine öğrenimi algoritmalarınızı eğitim veri kümeniz üzerinde seçip ayarladıktan sonra, modellerin görünmeyen veriler üzerinde nasıl performans gösterebileceğine dair nihai bir nesnel fikir edinmek için öğrenilen modelleri doğrulama veri kümesi üzerinde değerlendirebilirsiniz. Çapraz doğrulama kullanmak, görünmeyen veriler üzerinde model doğruluğunu tahmin etmek için uygulamalı makine öğreniminde altın bir standarttır. Elinizde veri varsa, bir doğrulama veri kümesi kullanmak da mükemmel bir uygulamadır.

## 7.7 Özet

Bu bölümde makine öğreniminin problemleri tümevarım yöntemiyle çözmek olduğunu keşfettiniz. Genellemenin, bir model tarafından öğrenilen kavramların yeni verilere ne kadar iyi uygulandığının bir açıklaması olduğunu öğrendiniz. Son olarak, makine öğreniminde aşırı uyum ve yetersiz uyum genelleme terminolojisini öğrendiniz:

Aşırı uyum: Eğitim verilerinde iyi performans, diğer verilere zayıf genelleme.

Yetersiz uyum: Eğitim verilerinde düşük performans ve diğer verilere zayıf genelleme.

Artık verilere aşırı ve yetersiz uyum sağlamanın risklerini biliyorsunuz. Bu bölüm, makine öğrenimi algoritmaları hakkındaki arka planınızı sona erdiriyor. Bir sonraki bölümde doğrusal algoritmalarla başlayarak makine öğrenimi algoritmaları hakkında bilgi edinmeye başlayacaksınız.

# Bölüm III

# Doğrusal Algoritmalar

# Bölüm 8

# Elektronik Tablo Matematiğinde Crash-Course

Bu kitaptaki eğitimler, bir elektronik tablo programı kullanarak tamamlamanız için tasarlanmıştır. Bu bölümde, bu kitaptaki eğitimleri tamamlamak için bilmeniz gereken bazı matematiksel fonksiyonlar hakkında hızlı bir kurs verilmektedir. Bu bölümü tamamladıktan sonra şunları öğreneceksiniz:

Bir elektronik tabloda temel aritmetik işlemlerin nasıl gerçekleştirileceği.

Verileri özetlemek için istatistiksel fonksiyonların nasıl kullanılacağı.

Test verisi olarak kullanmak için rastgele sayılar nasıl oluşturulur?

Eğitimleri tamamlamak için hangi elektronik tablo programını kullandığınız önemli değildir. Kullanılan tüm fonksiyonlar elektronik tablo programları arasında geneldir. Kullanabileceğiniz bazı önerilen programlar şunlardır:

Excel ile Microsoft Office.

Calc ile LibreOffice.

Mac'te Numbers.

Google Drive'da Google E-Tablolar.

Bir hesap tablosu programı kullanma konusunda zaten yetkinseniz, bu bölümü atlayabilirsiniz. Alternatif olarak, bir hesap tablosu kullanmanıza gerek yoktur ve öğreticileri doğrudan seçtiğiniz programlama dilinde uygulayabilirsiniz. Hadi başlayalım.

## 8.1 Aritmetik

Bazı temel elektronik tablo navigasyonu ve aritmetik ile başlayalım.

Bir hücre, eşittir (=) ve ardından ifadeyi kullanarak bir ifadeyi değerlendirebilir. Örneğin =1+1 ifadesi 2 olarak değerlendirilecektir.

SUM() fonksiyonunu kullanarak birden fazla hücredeki değerleri toplayabilirsiniz. Örneğin =SUM(A7:C7) ifadesi, A7 hücresinden C7 hücresine kadar olan aralıktaki değerlerin

toplamını veya değerlerini değerlendirecektir. Genellikle *i* yineleyici değişkeni kullanılarak bir aralık üzerinde, örneğin 1 ile *n* arasında toplama işlemi yazılır matematiksel olarak $\sum_{i=1}^{n}$.

COUNT() fonksiyonunu kullanarak bir aralıktaki hücreleri sayabilirsiniz. Örneğin şu ifade =COUNT(A7:C7) 3 olarak değerlendirilir çünkü aralıkta 3 hücre vardır.

Üslü sayılarla çalışmayı deneyelim.

Bir sayıyı ^ operatörünü kullanarak bir kuvvete yükseltebilirsiniz. Örneğin =2^2 ifadesi 2 sayısının karesini alır ve 4 olarak değerlendirir. Bu genellikle $2^2$ şeklinde yazılır.

LOG() fonksiyonunu kullanarak, varsayılan olarak 10 tabanına göre bir sayının logaritmasını hesaplayabilirsiniz. Logun, bir sayıyı bir kuvvete yükseltmenin ters işlemi olduğunu unutmayın. Örneğin =LOG(2,2) ifadesi, 2 tabanını kullanarak 4'ün logaritmasını hesaplayacak ve 2 olarak değerlendirilecektir.

SQRT()√ fonksiyonunu kullanarak bir sayının karekökünü hesaplayabilirsiniz. Örneğin, =SQRT(4) ifadesi 2 olarak değerlendirilir. Bu genellikle şu şekilde yazılır 4.

Matematiksel sabit Euler sayısı (*e*) ile çalışmayı deneyelim.

EXP() fonksiyonunu kullanarak bir sayıyı *e*'ye yükseltebiliriz. Örneğin =EXP(2) ifadesi 7.389056099 olarak değerlendirilecektir. Bu aynı zamanda $e^2$ olarak da yazılabilir.

LN() fonksiyonunu kullanarak bir sayının doğal logaritmasını hesaplayabiliriz. Doğal logaritmanın *e*'yi bir kuvvete yükseltmenin ters işlemi olduğunu unutmayın. Örneğin =LN(7.389056099) ifadesi 2 olarak değerlendirilecektir.

Diğer bazı faydalı şeyler:

PI() fonksiyonunu kullanarak matematiksel sabit PI'yi hesaplayabilirsiniz. Örneğin, =PI() ifadesi 3.141592654 olarak değerlendirilir. PI genellikle $\pi$ olarak yazılır.

## 8.2 İstatistiksel Özetler

Birçok makine öğrenimi algoritmasının girdi verilerinin istatistiksel özetlerini kullanması gerekir. Verileri nasıl özetleyebileceğimize bir göz atalım.

AVERAGE() fonksiyonunu kullanarak bir sayı listesinin ortalamasını veya ortalamasını hesaplayabilirsiniz. Ortalamanın bir sayı listesinin ortası ya da merkezi eğilimi olduğunu unutmayın. Örneğin, =AVERAGE(1,2,3) ifadesi 2 olarak değerlendirilecektir. Genellikle ortalama $\mu$ (mu) olarak adlandırılır.

MODE() fonksiyonunu kullanarak bir sayı listesinin modunu hesaplayabilirsiniz. Bir sayı listesinin modunun listedeki en yaygın değer olduğunu unutmayın. Örneğin =MODE(2,2,3) ifadesi 2 olarak değerlendirilecektir.

STDEV() fonksiyonunu kullanarak bir sayı listesinin standart sapmasını hesaplayabilirsiniz. Standart sapmanın, noktaların ortalama değerden ortalama yayılımı olduğunu unutmayın. Örneğin =STDEV(1,2,3) ifadesi 1 olarak değerlendirilir. Genellikle standart sapma $\sigma$ (sigma) olarak adlandırılır.

PEARSON() fonksiyonunu kullanarak iki sayı listesi arasındaki korelasyonu hesaplayabilirsiniz. Korelasyonun 1 ve -1 olmasının sırasıyla mükemmel pozitif ve negatif korelasyonu gösterdiğini unutmayın. Örneğin =PEARSON( 2,3,4 ,{4,5,6}){ifadesi}1 (mükemmel pozitif korelasyon) olarak değerlendirilir.

Bu örneklerin tümünde satır içi sayı listeleri kullanılmıştır, ancak girdi olarak hücre aralıkları da kolayca kullanılabilir.

## 8.3 Rastgele Sayılar

Makine öğrenimi algoritmalarını bir elektronik tabloda uygularken örnek verilere ihtiyacınız vardır. Kontrollü örnek veriler için en iyi kaynak rastgele sayılar kullanmaktır.

kullanarak 0 ile 1 aralığında düzgün rastgele bir sayı hesaplayabilirsiniz.
RAND() işlevi.

NORMINV() fonksiyonunu kullanarak bir Gauss rastgele sayısını hesaplayabilirsiniz. Gauss'un çan şekline sahip bir dağılım anlamına geldiğini unutmayın. Birçok doğrusal makine öğrenimi algoritması Gauss dağılımı varsayar. Örneğin, =NORMINV(RAND(), 10, 1) ifadesi, ortalaması 10 ve standart sapması 1 olan Gauss rastgele sayıları üretecektir.

## 8.4 Akış Kontrol

Elektronik tablonuzda temel akış kontrolü yapabilirsiniz.

IF() fonksiyonunu kullanarak bir hücreyi koşullu olarak değerlendirebilirsiniz. Üç bağımsız değişken alır; birincisi değerlendirilecek koşul, ikincisi koşul doğru olarak değerlendirilirse kullanılacak ifade ve son bağımsız değişken de koşul yanlış olarak değerlendirilirse kullanılacak ifadedir. Örneğin =IF(1>2, "YES", "NO") ifadesi NO olarak değerlendirilir.

## 8.5 Daha fazla Yardım

Bu bölümde sunulan fonksiyonlar konusunda uzman olmanız gerekmez, ancak bunları kullanma konusunda rahat olmalısınız. Kitaptaki eğitimlerde ilerledikçe, hangi fonksiyonları kullanmanız gerektiğini size hatırlatacağım. Eğer emin değilseniz, bu bölüme geri dönün ve referans olarak kullanın.

Elektronik tablolar mükemmel yardıma sahiptir. Bu hızlandırılmış kursta kullanılan işlevler veya diğer işlevler hakkında daha fazla bilgi edinmek istiyorsanız, lütfen elektronik tablo programınızdaki işlevler için yerleşik yardıma bakın. Yardım mükemmeldir ve fonksiyonları küçük test tablolarında kullanarak ve test verilerini bu tablolarda çalıştırarak daha fazla bilgi edinebilirsiniz.

## 8.6 Özet

Artık bu kitaptaki tüm eğitimleri tamamlamak için bir elektronik tablodaki matematiksel işlevleri yeterince biliyorsunuz. Öğrendiniz:

Bir elektronik tabloda sayılar, toplamlar, logaritmalar ve üsler gibi temel aritmetik işlemlerin nasıl gerçekleştirileceği.

Ortalama, mod ve standart sapma gibi veri özetlerini hesaplamak için istatistiksel fonksiyonların nasıl kullanılacağı.

Test verisi olarak kullanmak için tekdüze ve Gauss rastgele sayıları nasıl oluşturulur.

Bir elektronik tabloyu nasıl kullanacağınızı biliyorsunuz. Bunun da ötesinde, bir elektronik tabloda herhangi bir makine öğrenimi algoritmasını uygulamak ve bunlarla oynamak için kullanabileceğiniz temel araçlara sahipsiniz. Bir sonraki bölümde gradyan inişi adı verilen makine öğrenimindeki en yaygın optimizasyon algoritmasını keşfedeceksiniz.

# Bölüm 9

# Makine Öğrenimi İçin Gradyan İnişi

Optimizasyon, makine öğreniminin büyük bir parçasıdır. Neredeyse her makine öğrenimi algoritmasının özünde bir optimizasyon algoritması vardır. Bu bölümde herhangi bir makine öğrenimi algoritması ile kullanabileceğiniz basit bir optimizasyon algoritması keşfedeceksiniz. Anlaşılması ve uygulanması kolaydır. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Gradyan iniş optimizasyon algoritması hakkında.

Gradyan inişinin doğrusal regresyon gibi algoritmalarda nasıl kullanılabileceği.

Gradyan inişi çok büyük veri kümelerine nasıl ölçeklenebilir?

Pratikte gradyan inişinden en iyi şekilde yararlanmak için

ipuçları. Hadi başlayalım.

## 9.1 Gradyan İniş

Gradyan inişi, bir maliyet fonksiyonunu (*maliyet*) en aza indiren bir fonksiyonun ($f$) parametrelerinin (katsayılarının) değerlerini bulmak için kullanılan bir optimizasyon algoritmasıdır. Gradyan inişi, parametreler analitik olarak (örneğin doğrusal cebir kullanılarak) hesaplanamadığında ve bir optimizasyon algoritması tarafından aranması gerektiğinde en iyi şekilde kullanılır.

### 9.1.1 Gradyan İniş için Sezgi

İçinde seri yemek yiyeceğiniz veya meyve saklayacağınız büyük bir kase düşünün. Bu kase maliyet fonksiyonunun ($f$) bir grafiğidir. Kasenin yüzeyindeki rastgele bir konum, katsayıların mevcut değerlerinin maliyetidir (maliyet). Kasenin alt kısmı, fonksiyonun minimum değeri olan en iyi katsayı kümesinin maliyetidir.

Amaç, katsayılar için farklı değerler denemeye devam etmek, maliyetlerini değerlendirmek ve biraz daha iyi (daha düşük) maliyete sahip yeni katsayılar seçmektir. Bu işlemin yeterince tekrarlanması kasenin dibine ulaşmanızı sağlayacak ve minimum maliyetle sonuçlanan katsayıların değerlerini bileceksiniz.

### 9.1.2 Gradyan İniş Prosedürü

Prosedür, fonksiyonun katsayısı veya katsayıları için başlangıç değerleriyle başlar. Bunlar 0.0 veya küçük rastgele bir değer olabilir.

$$\text{katsayısı} = 0.0 \tag{9.1}$$

Katsayıların maliyeti, bunları fonksiyona yerleştirerek ve maliyeti hesaplayarak değerlendirilir.

$$\begin{aligned} &maliyet = f(\text{katsayı}) \\ &maliyet = değerlendir(f(\text{katsayı})) \end{aligned} \tag{9.2}$$

Maliyetin türevi hesaplanır. Türev, kalkülüsten gelen bir kavramdır ve belirli bir noktada fonksiyonun eğimini ifade eder. Eğimi bilmemiz gerekir, böylece bir sonraki iterasyonda daha düşük bir maliyet elde etmek için katsayı değerlerini taşıyacağımız yönü (işareti) biliriz.

$$delta = türev(maliyet) \tag{9.3}$$

Artık türevden hangi yönün yokuş aşağı olduğunu bildiğimize göre, şimdi katsayı değerlerini güncelleyebiliriz. Her güncellemede katsayıların ne kadar değişebileceğini kontrol eden bir öğrenme oranı parametresi (*alpha*) belirtilmelidir.

$$\text{katsayı} = \text{katsayı} - (alfa \times delta) \tag{9.4}$$

Bu işlem, katsayıların maliyeti (maliyet) 0,0 olana veya maliyette daha fazla iyileşme sağlanamayana kadar tekrarlanır. Gradyan inişinin ne kadar basit olduğunu görebilirsiniz. Maliyet fonksiyonunuzun veya optimize ettiğiniz fonksiyonun gradyanını bilmenizi gerektirir, ancak bunun dışında çok basittir. Daha sonra bunu makine öğrenimi algoritmalarında nasıl kullanabileceğimizi göreceğiz.

## 9.2 Toplu Gradyan İniş

Tüm denetimli makine öğrenimi algoritmalarının amacı, girdi verilerini ($X$) çıktı değişkenlerine ($Y$) eşleyen bir hedef fonksiyonu ($f$) en iyi şekilde tahmin etmektir. Bu, tüm sınıflandırma ve regresyon problemlerini tanımlar. Bazı makine öğrenimi algoritmaları, hedef fonksiyon ($f$) için algoritma tahminini karakterize eden katsayılara sahiptir. Farklı algoritmalar farklı temsillere ve farklı katsayılara sahiptir, ancak birçoğu hedef fonksiyonun en iyi tahminiyle sonuçlanan katsayı kümesini bulmak için bir optimizasyon süreci gerektirir. Gradyan inişi kullanılarak optimize edilebilen katsayılara sahip algoritmaların yaygın örnekleri Doğrusal Regresyon ve Lojistik Regresyondur.

Bir makine öğrenimi modelinin hedef fonksiyonu ne kadar yakın tahmin ettiğinin değerlendirilmesi, genellikle makine öğrenimi algoritmasına özgü olmak üzere bir dizi farklı yolla hesaplanabilir. Maliyet fonksiyonu, veri kümesindeki her bir eğitim örneği için bir tahmin hesaplayarak ve tahminleri gerçek çıktı değerleriyle karşılaştırarak makine öğrenimi modelindeki katsayıların değerlendirilmesini ve ardından bir toplam veya ortalama hatanın (doğrusal regresyon durumunda Karesel Kalıntıların Toplamı veya SSR gibi) hesaplanmasını içerir.

Maliyet fonksiyonundan her katsayı için bir türev hesaplanabilir, böylece tam olarak yukarıda açıklanan güncelleme denklemi kullanılarak güncellenebilir. Maliyet, gradyanın her iterasyonu için tüm eğitim veri kümesi üzerinden bir makine öğrenimi algoritması için hesaplanır

iniş algoritmasıdır. Algoritmanın bir iterasyonu bir yığın olarak adlandırılır ve bu gradyan inişi biçimi yığın gradyan inişi olarak adlandırılır. Toplu gradyan inişi, makine öğreniminde tanımlanan en yaygın gradyan inişi biçimidir.

## 9.3 Stokastik Gradyan İniş

Gradyan inişinin çok büyük veri kümelerinde çalıştırılması yavaş olabilir. Gradyan inişi algoritmasının bir iterasyonu eğitim veri kümesindeki her örnek için bir tahmin gerektirdiğinden, milyonlarca örneğe sahip olduğunuzda uzun zaman alabilir. Büyük miktarda veriye sahip olduğunuz durumlarda, gradyan inişinin stokastik gradyan inişi adı verilen bir varyasyonunu kullanabilirsiniz. Bu varyasyonda, yukarıda açıklanan gradyan inişi prosedürü çalıştırılır, ancak katsayıların güncellenmesi, örnek yığınının sonunda değil, her eğitim örneği için gerçekleştirilir.

Prosedürün ilk adımı, eğitim veri kümesinin sırasının rastgele olmasını gerektirir. Bu, katsayılara yapılan güncellemelerin sırasını karıştırmak içindir. Katsayılar her eğitim örneğinden sonra güncellendiğinden, güncellemeler gürültülü olacak, her yere atlayacak ve ilgili maliyet fonksiyonu da öyle olacaktır. Katsayıların güncellenme sırasını karıştırarak, bu rastgele yürüyüşten faydalanır ve takılıp kalmayı önler.

Katsayılar için güncelleme prosedürü yukarıdaki ile aynıdır, ancak maliyet tüm eğitim örüntüleri üzerinden toplanmaz veya ortalaması alınmaz, bunun yerine bir eğitim örüntüsü için hesaplanır. Çok büyük eğitim veri kümeleri için stokastik gradyan inişi ile öğrenme çok daha hızlı olabilir ve genellikle iyi veya yeterince iyi bir katsayı kümesine ulaşmak için veri kümesinden yalnızca az sayıda geçişe ihtiyacınız vardır, örneğin veri kümesinden 1 ila 10 geçiş.

## 9.4 Gradyan için İpuçları İniş

Bu bölümde, makine öğrenimi için gradyan iniş algoritmasından en iyi şekilde yararlanmak için bazı ipuçları ve püf noktaları listelenmektedir.

**Zamana Karşı Maliyet Çizimi**: Her bir iterasyonda algoritma tarafından hesaplanan maliyet değerlerini toplayın ve çizin. İyi performans gösteren bir gradyan inişi çalışması için beklenti, her iterasyonda maliyetin azalmasıdır. Eğer azalmazsa, öğrenme oranınızı azaltmayı deneyin.

**Öğrenme Oranı**: Öğrenme oranı değeri 0,1, 0,001 veya 0,0001 gibi küçük bir gerçek değerdir. Sorununuz için farklı değerler deneyin ve hangisinin en iyi sonucu verdiğini görün.

**Girdileri Yeniden Ölçeklendirin**: Maliyet fonksiyonunun şekli çarpık ve bozuk değilse algoritma minimum maliyete daha hızlı ulaşacaktır. Bunu, tüm girdi değişkenlerini ($X$) aynı aralıkta, örneğin 0 ile 1 arasında yeniden ölçeklendirerek başarabilirsiniz.

**Birkaç Geçiş**: Stokastik gradyan inişi genellikle iyi veya yeterince iyi katsayılara yakınsamak için eğitim veri kümesinden 1 ila 10 geçişten fazlasına ihtiyaç duymaz.

**Ortalama Maliyet Çizimi**: Stokastik gradyan inişi kullanılırken her bir eğitim veri kümesi örneği için yapılan güncellemeler, zaman içinde maliyetin gürültülü bir şekilde çizilmesine neden olabilir. Ortalamayı 10, 100 veya 1000 güncelleme üzerinden almak, algoritmanın öğrenme eğilimi hakkında size daha iyi bir fikir verebilir.

## 9.5 Özet

Bu bölümde makine öğrenimi için gradyan inişini keşfettiniz. Bunu öğrendiniz:

Optimizasyon, makine öğreniminin büyük bir parçasıdır.

Gradyan inişi, birçok makine öğrenimi algoritması ile kullanabileceğiniz basit bir optimizasyon prosedürüdür.

Toplu gradyan inişi, bir güncellemeyi hesaplamadan önce tüm eğitim verilerinden türevin hesaplanması anlamına gelir.

Stokastik gradyan inişi, her bir eğitim verisi örneğinden türevin hesaplanması ve güncellemenin hemen hesaplanması anlamına gelir.

Artık birçok makine öğrenimi algoritmasının temeli olan gradyan inişi optimizasyon algoritmasını biliyorsunuz. Bir sonraki bölümde, gerçek değerli veriler için tahminler yapmak için doğrusal regresyon algoritmasını keşfedeceksiniz.

# Bölüm 10

# Doğrusal

# Regresyon

Doğrusal regresyon, istatistik ve makine öğreniminde belki de en iyi bilinen ve en iyi anlaşılan algoritmalardan biridir. Bu bölümde doğrusal regresyon algoritmasını, nasıl çalıştığını ve makine öğrenimi projelerinizde en iyi şekilde nasıl kullanabileceğinizi keşfedeceksiniz. Bu bölümde şunları öğreneceksiniz:

Doğrusal regresyon neden hem istatistik hem de makine öğrenimine aittir?

Doğrusal regresyonun bilindiği birçok isim.

Doğrusal bir regresyon modeli oluşturmak için kullanılan temsil ve öğrenme algoritmaları.

Doğrusal regresyon kullanarak modelleme yaparken verilerinizi en iyi

şekilde nasıl hazırlayabilirsiniz? Hadi başlayalım.

## 10.1 Doğrusal Regresyon İstatistiklerinden değil mi?

Doğrusal regresyonun ayrıntılarına girmeden önce, kendinize neden bu algoritmaya baktığımızı soruyor olabilirsiniz. Bu bir istatistik tekniği değil mi?

Makine öğrenimi, daha spesifik olarak tahmine dayalı modelleme alanı, öncelikle bir modelin hatasını en aza indirmek veya açıklanabilirlik pahasına mümkün olan en doğru tahminleri yapmakla ilgilenir. Uygulamalı makine öğreniminde, istatistik de dahil olmak üzere birçok farklı alandan algoritmalar ödünç alacak, yeniden kullanacak ve çalacağız ve bunları bu amaçlar doğrultusunda kullanacağız.

Bu nedenle, doğrusal regresyon istatistik alanında geliştirilmiş ve girdi ve çıktı sayısal değişkenler arasındaki ilişkiyi anlamak için bir model olarak çalışılmıştır, ancak makine öğrenimi tarafından ödünç alınmıştır. Hem istatistiksel bir algoritma hem de bir makine öğrenimi algoritmasıdır. Şimdi, bir doğrusal regresyon modeline atıfta bulunmak için kullanılan bazı yaygın isimleri gözden geçirelim.

## 10.2 Doğrusal Regresyonun Birçok Adı

Doğrusal regresyonu incelemeye başladığınızda, işler çok kafa karıştırıcı olabilir. Bunun nedeni, doğrusal regresyonun çok uzun süredir (200 yıldan fazla) var olmasıdır. Mümkün olan her açıdan incelenmiştir ve genellikle her açının yeni ve farklı bir adı vardır.

Doğrusal regresyon doğrusal bir modeldir, örneğin girdi değişkenleri (*x*) ile tek çıktı değişkeni (*y*) arasında doğrusal bir ilişki olduğunu varsayan bir modeldir. Daha spesifik olarak, *y,* girdi değişkenlerinin (*x*) doğrusal bir kombinasyonundan hesaplanabilir. Tek bir girdi değişkeni (*x*) olduğunda, yöntem basit doğrusal regresyon olarak adlandırılır. Birden fazla girdi değişkeni olduğunda, istatistik literatürü genellikle yöntemi çoklu doğrusal regresyon olarak adlandırır. Doğrusal regresyon denklemini verilerden hazırlamak veya eğitmek için farklı teknikler kullanılabilir, bunlardan en yaygın olanı Sıradan En Küçük Kareler olarak adlandırılır. Bu nedenle, bu şekilde hazırlanan bir modele Sıradan En Küçük Kareler Doğrusal Regresyon veya sadece En Küçük Kareler Regresyonu olarak atıfta bulunmak yaygındır. Artık doğrusal regresyonu tanımlamak için kullanılan bazı isimleri bildiğimize göre, daha yakından bakalım kullanılan temsile bakın.

## 10.3 Doğrusal Regresyon Modeli Gösterimi

Doğrusal regresyon çekici bir modeldir çünkü gösterimi çok basittir. Temsil, belirli bir girdi değerleri kümesini (*x*) birleştiren ve çözümü bu girdi değerleri kümesi (*y*) için tahmin edilen çıktı olan doğrusal bir denklemdir. Bu nedenle, hem girdi değerleri (*x*) hem de çıktı değeri sayısaldır.

Doğrusal denklem, her girdi değerine veya sütununa, genellikle Yunan harfi Beta (*β*) ile temsil edilen katsayı adı verilen bir ölçek faktörü atar. Ayrıca, çizgiye ek bir serbestlik derecesi veren (örneğin iki boyutlu bir grafikte yukarı ve aşağı hareket eden) ve genellikle kesişim veya sapma katsayısı olarak adlandırılan bir katsayı daha eklenir. Örneğin, basit bir regresyon probleminde (tek bir *x* ve tek bir *y*), modelin şekli şöyle olacaktır:

$$y = B0 + B1 \times x \tag{10.1}$$

Daha yüksek boyutlarda, birden fazla girdimiz (*x*) olduğunda, doğruya düzlem veya hiper-düzlem denir. Dolayısıyla temsil, denklemin biçimi ve katsayılar için kullanılan belirli değerlerdir (örneğin yukarıdaki örnekte *B0* ve *B1*). Doğrusal regresyon gibi bir regresyon modelinin karmaşıklığı hakkında konuşmak yaygındır. Bu, modelde kullanılan katsayıların sayısını ifade eder.

Bir katsayı sıfır olduğunda, girdi değişkeninin model üzerindeki etkisini ve dolayısıyla modelden yapılan tahmini etkili bir şekilde ortadan kaldırır ($0 \times x = 0$). Regresyon modellerinin karmaşıklığını azaltmak için öğrenme algoritmasını değiştiren, katsayıların mutlak büyüklüğü üzerinde baskı uygulayarak bazılarını sıfıra çeken düzenlileştirme yöntemlerine bakarsanız bu durum önem kazanır. Doğrusal bir regresyon modeli için kullanılan temsili anladığımıza göre, bu temsili verilerden öğrenebileceğimiz bazı yolları gözden geçirelim.

## 10.4 Doğrusal Regresyon Modelini Öğrenme

Doğrusal bir regresyon modeli öğrenmek, elimizdeki verilerle gösterimde kullanılan katsayıların değerlerini tahmin etmek anlamına gelir. Bu bölümde doğrusal bir regresyon modeli hazırlamak için dört tekniğe kısaca göz atacağız. Bu, onları sıfırdan uygulamak için yeterli bilgi değildir, ancak ilgili hesaplama ve ödünleşimlerin tadını almak için yeterlidir.

Model çok iyi çalışıldığı için daha birçok teknik vardır. Genel olarak kullanılan en yaygın yöntem olduğu için Sıradan En Küçük Kareler'e dikkat edin. Ayrıca, makine öğrenimi perspektifinden öğretilen en yaygın teknik olduğu için Gradient Descent'i de not edin.

### 10.4.1 Basit Doğrusal Regresyon

Tek bir girdimiz olduğunda basit doğrusal regresyon ile katsayıları tahmin etmek için istatistikleri kullanabiliriz. Bu, verilerden ortalama, standart sapma, korelasyon ve kovaryans gibi istatistiksel özellikleri hesaplamanızı gerektirir. Tüm veriler, istatistikleri dolaşmak ve hesaplamak için kullanılabilir olmalıdır. Bu, bir elektronik tabloda alıştırma olarak eğlencelidir, ancak pratikte pek kullanışlı değildir.

### 10.4.2 Sıradan En Küçük Kareler

Birden fazla girdimiz olduğunda, katsayıların değerlerini tahmin etmek için Sıradan En Küçük Kareler yöntemini kullanabiliriz. Sıradan En Küçük Kareler prosedürü, karesel artıkların toplamını en aza indirmeyi amaçlar. Bu, veriler üzerinden bir regresyon doğrusu verildiğinde, her veri noktasından regresyon doğrusuna olan mesafeyi hesapladığımız, karesini aldığımız ve tüm karesel hataları topladığımız anlamına gelir. Bu, sıradan en küçük karelerin en aza indirmeye çalıştığı miktardır.

Bu yaklaşım verileri bir matris olarak ele alır ve katsayılar için en uygun değerleri tahmin etmek için doğrusal cebir işlemlerini kullanır. Bu, tüm verilerin mevcut olması gerektiği ve verileri sığdırmak ve matris işlemlerini gerçekleştirmek için yeterli belleğe sahip olmanız gerektiği anlamına gelir. Doğrusal cebir alıştırması olmadığı sürece Sıradan En Küçük Kareler prosedürünü kendiniz uygulamanız alışılmadık bir durumdur. Doğrusal cebir kütüphanesindeki bir prosedürü çağırmanız daha olasıdır. Bu prosedürün hesaplanması çok hızlıdır.

## 10.5 Gradyan İniş

Bir veya daha fazla girdi olduğunda, eğitim verileriniz üzerinde modelin hatasını yinelemeli olarak en aza indirerek katsayıların değerlerini optimize etme işlemini kullanabilirsiniz. Bu işlem Gradyan İnişi olarak adlandırılır ve her katsayı için sıfır değerle başlayarak çalışır. Karesel hataların toplamı her bir girdi ve çıktı değeri çifti için hesaplanır. Ölçek faktörü olarak bir öğrenme oranı kullanılır ve katsayılar hatayı en aza indirecek yönde güncellenir. İşlem, minimum karesel hata toplamına ulaşılana veya daha fazla iyileştirme mümkün olmayana kadar tekrarlanır.

Bu yöntemi kullanırken, prosedürün her iterasyonunda atılacak iyileştirme adımının boyutunu belirleyen bir öğrenme oranı (*alfa*) parametresi seçmeniz gerekir. Gradyan inişi genellikle doğrusal bir regresyon modeli kullanılarak öğretilir çünkü anlaşılması nispeten kolaydır. Pratikte, satır sayısı ya da sütun sayısı bakımından belleğe sığmayacak kadar büyük bir veri kümeniz olduğunda kullanışlıdır.

### 10.5.1 Düzenlenmiş Doğrusal Regresyon

Düzenli hale getirme yöntemleri olarak adlandırılan doğrusal model eğitiminin uzantıları vardır. Bunlar hem eğitim verileri üzerinde modelin karesel hata toplamını en aza indirmeyi (Sıradan En Küçük Kareler kullanarak) hem de modelin karmaşıklığını azaltmayı (sayı veya

modeldeki tüm katsayıların toplamının mutlak büyüklüğü). Doğrusal regresyon için iki popüler düzenlileştirme prosedürü örneği şunlardır:

Kement Regresyonu: Sıradan En Küçük Kareler'in katsayıların mutlak toplamını da en aza indirecek şekilde değiştirildiği durumdur (*L1* düzenlemesi olarak adlandırılır).

Ridge Regresyonu: Sıradan En Küçük Kareler'in katsayıların karesel mutlak toplamını da en aza indirecek şekilde değiştirildiği durumdur (*L2* düzenlemesi olarak adlandırılır).

Bu yöntemler, girdi değerlerinizde eş doğrusallık olduğunda ve sıradan en küçük kareler eğitim verilerine aşırı uyum sağladığında kullanmak için etkilidir. Artık bir doğrusal regresyon modelindeki katsayıları öğrenmek için bazı teknikler bildiğinize göre, yeni veriler üzerinde tahminler yapmak için bir modeli nasıl kullanabileceğimize bakalım.

## 10.6 Doğrusal Regresyon ile Tahmin Yapma

Temsilin doğrusal bir denklem olduğu düşünüldüğünde, tahmin yapmak, belirli bir girdi kümesi için denklemi çözmek kadar basittir. Bunu bir örnekle somutlaştıralım. Boydan (*x*) yola çıkarak ağırlığı (*y*) tahmin ettiğimizi düşünün. Bu problem için doğrusal regresyon modeli gösterimimiz şöyle olacaktır:

$$\begin{aligned} y &= \mathrm{B0} + \mathrm{B1} \times \mathrm{X1} \\ \textit{ağırlık} &= \mathrm{B0} + \mathrm{B1} \times \textit{yükseklik} \end{aligned} \tag{10.2}$$

Burada *B0* yanlılık katsayısı ve *B1* yükseklik sütunu için katsayıdır. İyi bir katsayı değerleri kümesi bulmak için bir öğrenme tekniği kullanırız. Bulduktan sonra, ağırlığı tahmin etmek için farklı boy değerlerini ekleyebiliriz. Örneğin, $B0 = 0{,}1$ ve $B1 = 0{,}5$ kullanalım. Bunları girelim ve 182 santimetre boyundaki bir kişinin ağırlığını (kilogram cinsinden) hesaplayalım.

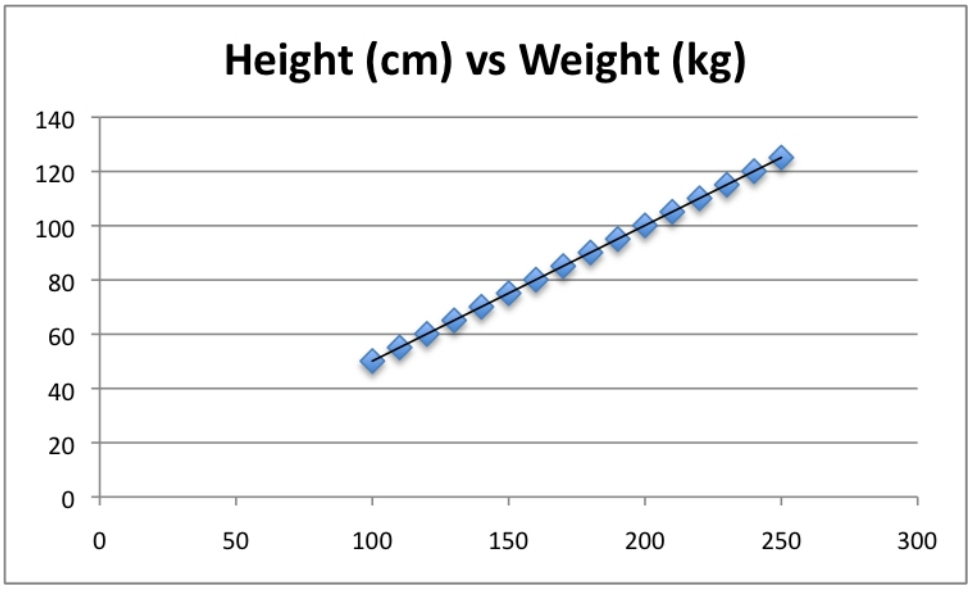
$$\begin{aligned} \textit{ağırlık} &= 0{,}1 + 0{,}05 \times 182 \\ \textit{ağırlık} &= 91{,}1 \end{aligned} \tag{10.3}$$

Yukarıdaki denklemin iki boyutlu bir çizgi olarak çizilebileceğini görebilirsiniz. B0, hangi yüksekliğe sahip olursak olalım başlangıç noktamızdır. Çizgimizi oluşturmak için 100 ila 250 santimetre arasında bir dizi yükseklikten geçebilir ve bunları denkleme ekleyerek ağırlık değerlerini elde edebiliriz.

Artık öğrenilmiş bir doğrusal regresyon modeli ile nasıl tahminler yapacağımızı bildiğimize göre, bu tür bir modelden en iyi şekilde yararlanmak için verilerimizi hazırlamaya yönelik bazı temel kurallara bakalım.

## 10.7 Doğrusal Regresyon için Veri Hazırlama

Doğrusal regresyon üzerinde uzun uzadıya çalışılmıştır ve modelden en iyi şekilde yararlanmak için verilerinizin nasıl yapılandırılması gerektiğine dair çok sayıda literatür bulunmaktadır. Bu nedenle, bu gereksinimler ve beklentiler hakkında konuşurken göz korkutucu olabilecek çok fazla karmaşıklık vardır. Pratikte, doğrusal regresyonun en yaygın uygulaması olan Sıradan En Küçük Kareler Regresyonunu kullanırken bu kuralları daha çok temel kurallar olarak kullanabilirsiniz. Bu sezgisel yöntemleri kullanarak verilerinizin farklı hazırlıklarını deneyin ve sorununuz için en iyi sonucu neyin verdiğini görün.




Şekil 10.1: Örnek Boy ve Ağırlık Doğrusal Regresyonu.

**Doğrusal Varsayım**. Doğrusal regresyon, girdiniz ve çıktınız arasındaki ilişkinin doğrusal olduğunu varsayar. Başka hiçbir şeyi desteklemez. Bu açık olabilir, ancak çok sayıda özelliğiniz olduğunda hatırlamakta fayda vardır. İlişkiyi doğrusal hale getirmek için verileri dönüştürmeniz gerekebilir (örneğin, üstel bir ilişki için log dönüşümü).

**Gürültüyü Kaldırın**. Doğrusal regresyon, girdi ve çıktı değişkenlerinizin gürültülü olmadığını varsayar. Verilerinizdeki sinyali daha iyi ortaya çıkarmanızı ve netleştirmenizi sağlayan veri temizleme işlemlerini kullanmayı düşünün. Bu en çok çıktı değişkeni için önemlidir ve mümkünse çıktı değişkenindeki ($y$) aykırı değerleri kaldırmak istersiniz.

**Bağlantılılığı Kaldırın**. Yüksek korelasyonlu girdi değişkenleriniz olduğunda doğrusal regresyon verilerinize aşırı uyum sağlayacaktır. Girdi verileriniz için ikili korelasyonları hesaplamayı ve en korelasyonlu olanları kaldırmayı düşünün.

**Gauss Dağılımları**. Girdi ve çıktı değişkenleriniz Gauss dağılımına sahipse doğrusal regresyon daha güvenilir tahminler yapacaktır. Dağılımlarını daha Gauss görünümlü hale getirmek için değişkenlerinizde dönüşümler (örn. log veya BoxCox) kullanarak bazı faydalar elde edebilirsiniz.

**Girdileri Yeniden Ölçeklendirin**: Standardizasyon veya normalizasyon kullanarak girdi değişkenlerini yeniden ölçeklendirirseniz doğrusal regresyon genellikle daha güvenilir tahminler yapacaktır.

## 10.8 Özet

Bu bölümde makine öğrenimi için doğrusal regresyon algoritmasını keşfettiniz. Aşağıdakiler dahil birçok konuyu ele aldınız:

Doğrusal regresyon modellerini tanımlarken kullanılan yaygın isimler.

Model tarafından kullanılan temsil.

Modeldeki katsayıları tahmin etmek için kullanılan öğrenme algoritmaları.

Doğrusal regresyon ile kullanılmak üzere veri hazırlarken dikkate alınması gereken temel kurallar.

Artık gerçek değerli tahminler yapmak için doğrusal regresyon algoritmasını biliyorsunuz. Bir sonraki bölümde basit doğrusal regresyon algoritmasını sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 11

# Basit Doğrusal Regresyon Eğitimi

Doğrusal regresyon çok basit bir yöntemdir ancak çok sayıda durum için çok yararlı olduğu kanıtlanmıştır. Bu bölümde doğrusal regresyonun tam olarak nasıl çalıştığını adım adım keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Basit bir doğrusal regresyon adım adım nasıl hesaplanır.

Modelinizi kullanarak yeni veriler üzerinde nasıl tahminlerde bulunabilirsiniz?

Hesaplamayı büyük ölçüde basitleştiren bir kısayol.

Hadi başlayalım.

## 11.1 Öğretici Veri Seti

Kullandığımız veri seti tamamen uydurmadır. Aşağıda ham veriler yer almaktadır.

```
x   y
1   1
2   3
4   3
3   2
5   5
```

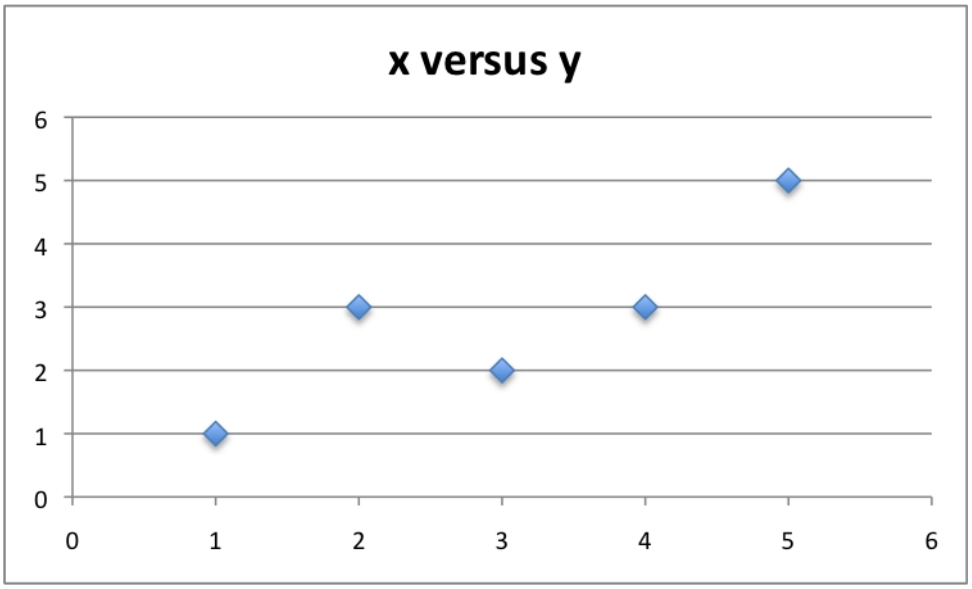
Liste 11.1: Öğretici Veri Seti.

*x* özniteliği girdi değişkenidir ve *y* tahmin etmeye çalıştığımız çıktı değişkenidir. Daha fazla veriye sahip olsaydık, sadece *x* değerlerine sahip olurduk ve *y değerlerini* tahmin etmekle ilgilenirdik. Aşağıda *x*'e karşı *y'nin* basit bir dağılım grafiği yer almaktadır.

*x* ve *y* arasındaki ilişkinin doğrusal gibi göründüğünü görebiliriz. Veriler arasındaki ilişkiyi genel olarak tanımlamak için muhtemelen grafiğin sol altından sağ üst tarafına çapraz olarak bir çizgi çizebiliriz. Bu, doğrusal regresyon kullanmanın bu küçük veri kümesi için uygun olabileceğinin iyi bir göstergesidir.

## 11.2 Basit Doğrusal Regresyon

Tek bir girdi özniteliğimiz (*x*) olduğunda ve doğrusal regresyon kullanmak istediğimizde, buna basit doğrusal regresyon denir. Birden fazla girdi özniteliğimiz olsaydı (örneğin *X1*, *X2*, *X3*, vb.) Bu




Şekil 11.1: Basit Doğrusal Regresyon Veri Kümesi.

çoklu doğrusal regresyon olarak adlandırılabilir. Doğrusal regresyon prosedürü çoklu doğrusal regresyondan farklı ve daha basittir, bu nedenle başlamak için iyi bir yerdir. Bu bölümde, eğitim verilerimizden basit bir doğrusal regresyon modeli oluşturacağız, ardından modelin verilerdeki ilişkiyi ne kadar iyi öğrendiğine dair bir fikir edinmek için eğitim verilerimiz için tahminler yapacağız. Basit doğrusal regresyon ile verilerimizi aşağıdaki gibi modellemek istiyoruz:

$$y = \mathrm{B0} + \mathrm{B1} \times x \tag{11.1}$$

Bu, y'nin tahmin etmek istediğimiz çıktı değişkeni, *x*'in bildiğimiz girdi değişkeni ve *B0* ve *B1*'in çizgiyi hareket ettiren tahmin etmemiz gereken katsayılar olduğu bir çizgidir. Teknik olarak B0'a kesişim denir çünkü doğrunun y eksenini nerede kestiğini belirler. Makine öğreniminde buna önyargı diyebiliriz, çünkü yaptığımız tüm tahminleri dengelemek için eklenir. *B1* terimi eğim olarak adlandırılır çünkü doğrunun eğimini ya da biz yanlılığımızı eklemeden önce *x*'in *y* değerine nasıl dönüştüğünü tanımlar.

Amaç, *x*'ten *y'yi* tahmin etmedeki hataları en aza indirmek için katsayılar için en iyi tahminleri bulmaktır. Basit regresyon harikadır, çünkü değerleri deneme yanılma yoluyla aramak veya daha gelişmiş doğrusal cebir kullanarak analitik olarak hesaplamak yerine, bunları doğrudan verilerimizden tahmin edebiliriz. B1 değerini şu şekilde tahmin ederek başlayabiliriz:

$$\mathrm{B1} = \frac{\sum_{i=1}^{n} (x_i - ortalama(x)) \times (y_i - ortalama(y))}{\sum_{i=1}^{n} (x_i - ortalama(x))^2} \tag{11.2}$$

Burada *mean*() veri kümemizdeki değişken için ortalama değerdir. $x_i$ ve $y_i$ bu hesaplamaları veri setimizdeki tüm değerler için tekrarlamamız gerektiğini ve *i x* veya *y'nin i*'inci değerini ifade eder. *B0*'ı *B1*'*i* ve veri setimizdeki bazı istatistikleri kullanarak aşağıdaki gibi hesaplayabiliriz:

$$B0 = ortalama(y) - B1 \times ortalama(x) \tag{11.3}$$

O kadar da kötü değil, değil mi? Bunları doğrudan elektronik tablomuzda hesaplayabiliriz.

### 11.2.1 Eğimin Tahmin Edilmesi (B1)

Denklemin en üst kısmı olan pay ile başlayalım. Önce *x* ve *y'nin* ortalama değerini hesaplamamız gerekir. Ortalama şu şekilde hesaplanır:

$$\frac{1}{n} \times \sum_{i=1}^{n} x_i \tag{11.4}$$

Burada *n* değer sayısıdır (bu durumda 5). Elektronik tablonuzda AVERAGE() fonksiyonunu kullanabilirsiniz. Şimdi *x* ve *y* değişkenlerimizin ortalama değerini hesaplayalım:

$$\begin{aligned} &ortalama(x) \\ &= 3 \\ &ortalama(y) \\ &= 2{,}8 \end{aligned} \tag{11.5}$$

Şimdi her bir değişkenin ortalamadan hatasını hesaplamamız gerekiyor. Bunu önce *x* ile yapalım:

```
x   ortalama   x -
    (x)        ortalama(x)
1   3          -2
2              -1
4              1
3              0
5              2
```

Liste 11.2: Her x değerinin ortalamadan kalıntısı.

Şimdi bunu *y* değişkeni için yapalım.

```
y   ortalama   y -
    (y)        ortalama(y)
1   2.8        -1.8
3              0.2
3              0.2
2              -0.8
5              2.2
```

Liste 11.3: Her bir y değerinin ortalamadan kalıntıları.

Artık payı hesaplamak için gerekli parçalara sahibiz. Tek yapmamız gereken her *x* için hatayı her *y* için hata ile çarpmak ve bu çarpımların toplamını hesaplamaktır.

```
x - ortalama   y -           Çarpma İşlemi
    (x)        ortalama(y)
-2             -1.8          3.6
-1             0.2           -0.2
1              0.2           0.2
0              -0.8          0
2              2.2           4.4
```

Liste 11.4: x ve y artıklarının ortalamalarından çarpılması.

Son sütunu toplayarak payımızı 8 olarak hesapladık. Şimdi *B1*'i hesaplamak için denklemin alt kısmını veya paydayı hesaplamamız gerekiyor. Bu, her bir *x* değerinin ortalamadan karesel farklarının toplamı olarak hesaplanır. Her bir *x* değerinin ortalamadan farkını zaten hesapladık, tek yapmamız gereken her bir değerin karesini almak ve toplamı hesaplamak.

```
x - ortalama(x)     kare
-2                  4
-1                  1
```

```
1           1
0           0
2           4
```

Liste 11.5: Her bir x değerinin ortalamadan karesel artığı.

Bu kareli değerlerin toplamını hesaplamak bize 10'luk bir payda verir. Şimdi eğimimizin değerini hesaplayabiliriz.

$$\begin{aligned} B1 &= \frac{8}{10} \\ B1 &= 0.8 \end{aligned} \tag{11.6}$$

### 11.2.2 Kesişimin Tahmin Edilmesi (B0)

İlgili tüm terimlerin değerlerini zaten bildiğimiz için bu çok daha kolaydır.

$$\begin{aligned} B0 &= ortalama(y) - B1 \times ortalama(x) \\ B0 &= 2{,}8 - 0{,}8 \times 3 \\ B0 &= 0.4 \end{aligned} \tag{11.7}$$

## 11.3 Tahminlerde Bulunmak

Artık basit doğrusal regresyon denklemimiz için katsayılara sahibiz.

$$\begin{aligned} y &= B0 + B1 \times x \\ y &= 0{,}4 + 0{,}8 \times x \end{aligned} \tag{11.8}$$

Eğitim verilerimiz için tahminler yaparak modeli deneyelim.

```
x   Tahmini Y
1   1.2
2   2
4   3.6
3   2.8
5   4.4
```

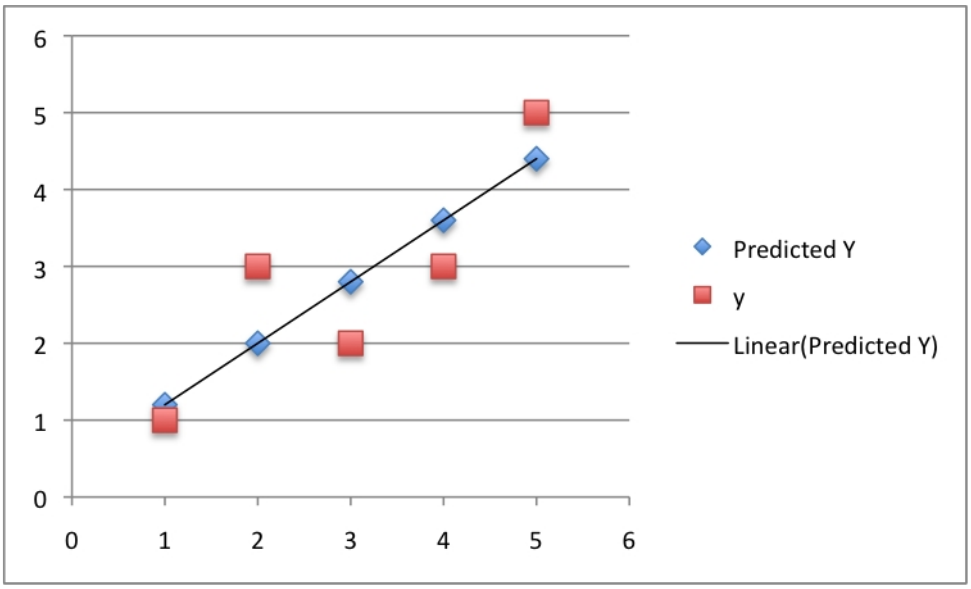
Liste 11.6: Her x giriş değeri için tahmin edilen y değeri.

Bu tahminleri verilerimizle birlikte bir doğru olarak çizebiliriz. Bu bize doğrunun verilerimizi ne kadar iyi modellediğine dair görsel bir fikir verir.

## 11.4 Tahmin Etme Hata

Tahminlerimiz için Kök Ortalama Karesel Hata veya RMSE adı verilen bir hata puanı hesaplayabiliriz.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (p_i - y_i)^2}{n}} \tag{11.9}$$




Şekil 11.2: Basit Doğrusal Regresyon Tahminleri.

Karekökü hesaplamak için elektronik tablonuzda SQRT() işlevini kullanabileceğiniz yerde, *p* tahmin edilen değer ve *y* gerçek değerdir, *i* belirli bir örnek için indekstir, çünkü tahmin edilen tüm değerlerdeki hatayı hesaplamamız gerekir. Öncelikle her bir model tahmini ile gerçek *y* değerleri arasındaki farkı hesaplamalıyız.

```
Tahmini    y   Tahmin edilen
                - y
1.2        1   0.2
2          3   -1
3.6        3   0.6
2.8        2   0.8
4.4        5   -0.6
```

Liste 11.7: Tahmin edilen değerler için hata.

Bu hata değerlerinin her birinin karesini (*hata* × *hata* veya *hata*$^2$) kolayca hesaplayabiliriz.

```
Tahmin edilen -   karesel hata
y
0.2               0.04
-1                1
0.6               0.36
0.8               0.64
-0.6              0.36
```

Liste 11.8: Tahmin edilen değerler için karesel hata.

Bu hataların toplamı 2,4 birimdir, 5'e bölünür ve karekökü alınır:

$$RMSE = 0.692820323 \tag{11.10}$$

Ya da her bir tahmin ortalama olarak yaklaşık 0,692 birim yanlıştır.

## 11.5 Kısayol

Bitirmeden önce size katsayıları hesaplamak için hızlı bir kısayol göstermek istiyorum. Basit doğrusal regresyon, regresyonun en basit şeklidir ve en çok çalışılanıdır. B0 ve *B1* değerlerini hızlı bir şekilde tahmin etmek için kullanabileceğiniz bir kısayol vardır. Aslında bu *B1*'i hesaplamak için bir kısayoldur. *B1*'in hesaplanması şu şekilde yeniden yazılabilir:

$$\text{B1} = corr(x, y) \times \frac{stdev(y)}{stdev(x)} \tag{11.11}$$

Burada *corr*(*x*) *x* ve *y* arasındaki korelasyondur ve *stdev*() bir değişken için standart sapmanın hesaplanmasıdır. Korelasyon (Pearson korelasyon katsayısı olarak da bilinir), iki değişkenin -1 ila 1 aralığında ne kadar ilişkili olduğunun bir ölçüsüdür. 1 değeri, iki değişkenin tamamen pozitif ilişkili olduğunu, her ikisinin de aynı yönde hareket ettiğini ve -1 değeri ise tamamen negatif ilişkili olduğunu, biri hareket ettiğinde diğerinin de diğer yönde hareket ettiğini gösterir.

Standart sapma, verilerin ortalamadan ne kadar farklı olduğunu gösteren bir ölçüdür. Elektronik tablonuzda PEARSON() fonksiyonunu kullanarak *x* ve *y* arasındaki korelasyonu 0,852 (yüksek korelasyon) olarak hesaplayabilir ve STDEV() fonksiyonunu kullanarak *x*'in standart sapmasını 1,5811 ve *y'nin* standart sapmasını 1,4832 olarak hesaplayabilirsiniz. Bu değerleri girerek şunu elde ederiz:

$$\begin{aligned} \text{B1} &= 0.852802865 \times \frac{1.483239697}{1.58113883} \\ \text{B1} &= 0.8 \end{aligned} \tag{11.12}$$

## 11.6 Özet

Bu bölümde basit doğrusal regresyonun bir elektronik tabloda adım adım nasıl uygulanacağını keşfettiniz. Şunları öğrendiniz:

Eğitim verilerinizden basit bir doğrusal regresyon modeli için katsayılar nasıl tahmin edilir.

Öğrendiğiniz modeli kullanarak nasıl tahminler yaparsınız?

Artık basit doğrusal regresyon algoritmasını sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde, stokastik gradyan inişi kullanarak sıfırdan doğrusal regresyonu nasıl uygulayabileceğinizi keşfedeceksiniz.

# Bölüm 12

# Gradient Descent Kullanarak Doğrusal Regresyon Eğitimi

Stokastik Gradyan İnişi, makine öğreniminde önemli ve yaygın olarak kullanılan bir algoritmadır. Bu bölümde, bir eğitim veri kümesi üzerindeki hatayı en aza indirerek basit bir doğrusal regresyon modelinin katsayılarını öğrenmek için Stokastik Gradyan İnişini nasıl kullanacağınızı keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Bir regresyon modelinin katsayılarını aramak için stokastik gradyan inişinin nasıl kullanılabileceği.

Gradyan inişinin tekrarlanan iterasyonları nasıl doğru bir regresyon modeli

oluşturabilir? Hadi başlayalım.

## 12.1 Öğretici Veri Seti

Veri seti, Basit Doğrusal Regresyon ile ilgili bir önceki bölümde kullanılanla aynıdır. Bütünlük için tekrar listelenmiştir.

```
x   y
1   1
2   3
4   3
3   2
5   5
```

Liste 12.1: Öğretici Veri Seti.

## 12.2 Stokastik Gradyan İniş

Gradyan İnişi, maliyet fonksiyonunun gradyanlarını takip ederek bir fonksiyonu minimize etme işlemidir. Bu, maliyetin biçimini ve türevini bilmeyi gerektirir, böylece belirli bir noktadan gradyanı bilirsiniz ve bu yönde, örneğin yokuş aşağı minimum değere doğru hareket edebilirsiniz. Makine öğreniminde, maliyet fonksiyonunu değerlendiren ve güncelleyen bir teknik kullanabiliriz.

Eğitim verilerimiz üzerinde bir modelin hatasını en aza indirmek için stokastik gradyan inişi adı verilen her iterasyonda katsayılar.

Bu optimizasyon algoritmasının çalışma şekli, her bir eğitim örneğinin modele teker teker gösterilmesidir. Model bir eğitim örneği için bir tahminde bulunur, hata hesaplanır ve bir sonraki tahminde hatayı azaltmak için model güncellenir. Bu prosedür, eğitim verileri üzerinde model için en küçük hatayla sonuçlanan bir modeldeki katsayılar kümesini bulmak için kullanılabilir. Her iterasyonda, makine öğrenimi dilinde ağırlıklar (*w*) olarak adlandırılan katsayılar denklem kullanılarak güncellenir:

$$w = w - alfa \times delta \tag{12.1}$$

Burada *w* optimize edilen katsayı veya ağırlık, *alpha* yapılandırmanız gereken bir öğrenme oranı (örn. 0,1) ve gradyan da ağırlığa atfedilen eğitim verileri üzerindeki modelin hatasıdır.

# 12.3 Stokastik Gradyan ile Basit Doğrusal Regresyon İniş

Basit doğrusal regresyonda kullanılan katsayılar stokastik gradyan inişi kullanılarak bulunabilir. Stokastik gradyan inişi, veri seti geleneksel En Küçük Kareler yönteminin kullanılmasını engellemediği sürece (örneğin çok büyük bir veri seti) pratikte doğrusal regresyon katsayılarını hesaplamak için kullanılmaz. Bununla birlikte, doğrusal regresyon, makine öğrenimi algoritmaları tarafından maliyet fonksiyonlarını en aza indirmek için kullanılan önemli bir algoritma olan stokastik gradyan inişini uygulamak için yararlı bir alıştırma sağlar. Önceki bölümde belirtildiği gibi, doğrusal regresyon modelimiz aşağıdaki şekilde tanımlanmıştır:

$$y = B0 + B1 \times x \tag{12.2}$$

## 12.3.1 Gradyan İniş İterasyonu #1

Her iki katsayı için de 0,0 değeriyle başlayalım.

$$\begin{aligned} B0 &= 0.0 \\ B1 &= 0.0 \\ y &= 0.0 + 0.0 \times x \end{aligned} \tag{12.3}$$

Bir tahmin için hatayı aşağıdaki şekilde hesaplayabiliriz:

$$hata = p(i) - y(i) \tag{12.4}$$

Burada $p(i)$ veri setimizdeki $i$'inci örnek için tahmin ve $y(i)$ veri setindeki örnek için $i$'inci çıktı değişkenidir. Şimdi ilk eğitim örneği için başlangıç noktası katsayılarımızı kullanarak y için öngörülen değeri hesaplayabiliriz: $x = 1$, $y = 1$.

$$\begin{aligned} p(i) &= 0.0 + 0.0 \times 1 \\ p(i) &= 0 \end{aligned} \tag{12.5}$$

Tahmin edilen çıktıyı kullanarak hatamızı hesaplayabiliriz:

$$hata = (0 - 1)$$
$$hata = -1 \tag{12.6}$$

Şimdi bu hatayı ağırlıkları güncellemek için gradyan inişi denklemimizde kullanabiliriz. Önce kesişme noktasını güncellemekle başlayacağız, çünkü bu daha kolay. B0'ın tüm hatadan sorumlu olduğunu söyleyebiliriz. Bu, ağırlığın güncellenmesinde gradyan olarak sadece hatanın kullanılacağı anlamına gelir. *B0* katsayısı için güncellemeyi aşağıdaki gibi hesaplayabiliriz:

$$B0(t + 1) = B0(t) - alfa \times hata \tag{12.7}$$

Burada $B0(t + 1)$ bir sonraki eğitim örneğinde kullanacağımız katsayının güncellenmiş versiyonudur, $B0(t)$ *B0* için mevcut değerdir, *alfa* öğrenme oranımızdır ve hata eğitim örneği için hesapladığımız hatadır. Şimdi 0,01'lik küçük bir öğrenme oranı kullanalım ve B0'ın yeni ve biraz optimize edilmiş değerinin ne olacağını bulmak için değerleri denkleme yerleştirelim:

$$B0(t + 1) = 0.0 - 0.01 \times -1.0$$
$$B0(t + 1) = 0,01 \tag{12.8}$$

Şimdi, *B1* değerini güncellemeye bakalım. Küçük bir değişiklikle aynı denklemi kullanacağız. Hata, buna neden olan girdi tarafından filtrelenir. Denklemi kullanarak B1'i güncelleyebiliriz:

$$B1(t + 1) = B1(t) - alfa \times hata \times x \tag{12.9}$$

Burada $B1(t + 1)$ güncelleme katsayısı, $B1(t)$ ise katsayının mevcut versiyonudur, *alfa* yukarıda açıklanan öğrenme oranının aynısı, hata yukarıda hesaplanan hatanın aynısı ve *x* girdi değeridir. Sayılarımızı denkleme ekleyebilir ve *B1* için güncellenmiş değeri hesaplayabiliriz:

$$B1(t + 1) = 0,0 - 0,01 \times -1 \times 1$$
$$B1(t + 1) = 0,01 \tag{12.10}$$

Gradyan inişinin ilk iterasyonunu henüz tamamladık ve ağırlıklarımızı $B0 = 0.01$ ve $B1 = 0.01$ olacak şekilde güncelledik. Bu işlem, veri kümemizdeki kalan 4 örnek için tekrarlanmalıdır. Eğitim veri kümesinden bir geçiş bir epok olarak adlandırılır.

## 12.3.2 Gradyan İniş İterasyonu #20

İleriye atlayalım. Bu işlemi 19 kez daha tekrarlayabilirsiniz. Bu, eğitim verilerinin modele maruz kaldığı ve katsayıların güncellendiği 4 tam epoktur. İşte görmeniz gereken 20 iterasyon boyunca katsayılar için tüm değerlerin bir listesi:

```
B0          B1
0.01        0.01
0.0397      0.0694
0.066527    0.176708
0.08056049  0.21880847
0.118814462 0.410078328
0.123525534 0.4147894
0.14399449  0.455727313
```

```
0.154325453    0.497051164
```

```
0.157870663    0.507686795
0.180907617    0.622871563
0.182869825    0.624833772
0.198544452    0.656183024
0.200311686    0.663251962
0.19841101     0.657549935
0.213549404    0.733241901
0.21408149     0.733773988
0.227265196    0.760141398
0.224586888    0.749428167
0.219858174    0.735242025
0.230897491    0.79043861
```

Liste 12.2: Basit doğrusal regresyon katsayıları 20 iterasyondan sonra.

Bence 20 iterasyon veya 4 epok güzel bir yuvarlak sayı ve durmak için iyi bir yer. İsterseniz devam edebilirsiniz. Değerleriniz birbirine yakın olmalıdır, ancak farklı elektronik tablo programları ve farklı hassasiyetler nedeniyle küçük farklılıklar olabilir. Her bir katsayı çiftini basit doğrusal regresyon denklemine geri takabilirsiniz. Bu kullanışlıdır çünkü her bir eğitim örneği için bir tahmin hesaplayabilir ve karşılığında hatayı hesaplayabiliriz.

Aşağıda, öğrenme süreci ilerledikçe her bir katsayı kümesi için hatanın bir grafiği yer almaktadır. Bu, bize hatanın her iterasyonda azaldığını ve sona doğru biraz sıçramaya başladığını göstermesi açısından faydalı bir grafiktir.


Şekil 12.1: Yinelemeye Karşı Basit Doğrusal Regresyon Performansı.

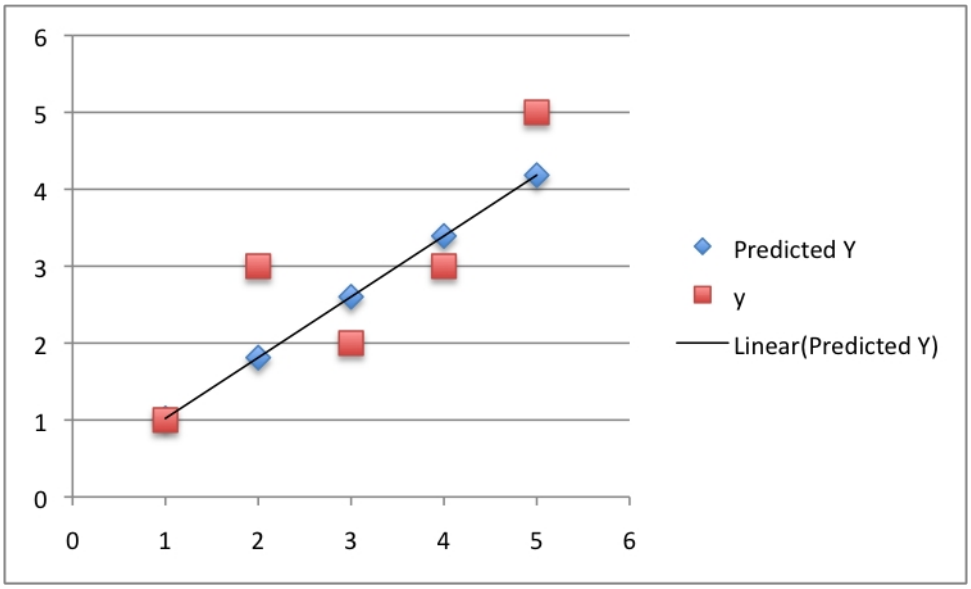
Nihai katsayılarımızın *B0* = 0.230897491 ve *B1* = 0.79043861 değerlerine sahip olduğunu görebilirsiniz. Bunları basit doğrusal Regresyon modelimize ekleyelim ve eğitim veri setimizdeki her nokta için bir tahmin yapalım.

```
x   Tahmin
1   1.021336101
2   1.811774711
4   3.392651932
3   2.602213322
```

```
5   4.183090542
```

Liste 12.3: Eğitim veri kümesi için basit doğrusal regresyon tahminleri.

Veri setimizi bu tahminleri üst üste bindirerek tekrar çizebiliriz (*x* vs *y* ve *x* vs *tahmin*). Bu 5 tahmin üzerinden bir çizgi çizmek, modelin eğitim verilerine ne kadar iyi uyduğu konusunda bize bir fikir verir.




Şekil 12.2: Basit Doğrusal Regresyon Tahminleri.

Önceki bölümde yaptığımız gibi bu tahminler için RMSE'yi hesaplayabiliriz. Sonuç *RMSE* = 0.720626401 olarak ortaya çıkar.

## 12.4 Özet

Bu bölümde basit doğrusal regresyon modelini ve stokastik gradyan inişi kullanarak bu modeli nasıl eğiteceğinizi keşfettiniz. Şunları öğrendiniz:

Gradyan inişi için güncelleme kuralının uygulanması yoluyla nasıl çalışılır.

Öğrenilmiş bir doğrusal regresyon modeli kullanarak tahminler nasıl yapılır.

Artık stokastik gradyan inişi kullanarak doğrusal regresyonun nasıl uygulanacağını biliyorsunuz. Bir sonraki bölümde ikili sınıflandırma için lojistik regresyon algoritmasını keşfedeceksiniz.

# Bölüm 13 Lojistik Regresyon

Lojistik regresyon, makine öğrenimi tarafından istatistik alanından ödünç alınan bir başka tekniktir. İkili sınıflandırma problemleri (iki sınıf değeri olan problemler) için başvurulan bir yöntemdir. Bu bölümde makine öğrenimi için lojistik regresyon algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Lojistik regresyonu tanımlarken kullanılan birçok isim ve terim (log odds ve logit gibi).

Lojistik regresyon modeli için kullanılan gösterim.

Bir lojistik regresyon modelinin katsayılarını verilerden öğrenmek için kullanılan teknikler.

Öğrenilmiş bir lojistik regresyon modeli kullanarak gerçekte nasıl tahminler yapılır.

Biraz daha derine inmek isterseniz daha fazla bilgi için nereye

gideceğiniz. Hadi başlayalım.

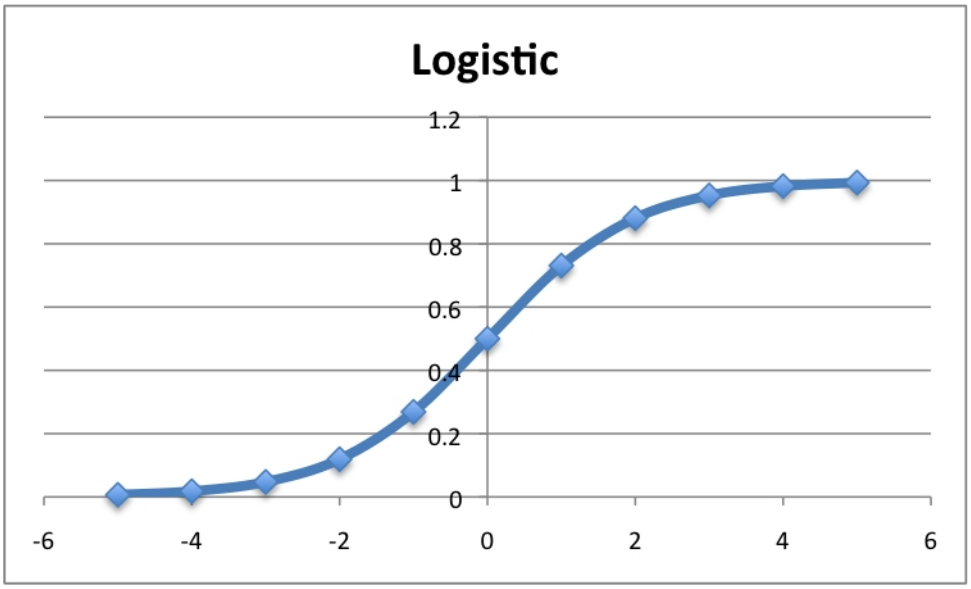
## 13.1 Lojistik Fonksiyonu

Lojistik regresyon, adını yöntemin temelinde kullanılan fonksiyondan, lojistik fonksiyondan almaktadır. Sigmoid fonksiyon olarak da adlandırılan lojistik fonksiyon, ekolojide nüfus artışının özelliklerini tanımlamak için istatistikçiler tarafından geliştirilmiştir, hızla yükselir ve ortamın taşıma kapasitesinde maksimuma ulaşır. Herhangi bir gerçek değerli sayıyı alıp 0 ile 1 arasında bir değere eşleyebilen, ancak asla tam olarak bu sınırlarda olmayan *S* şeklinde bir eğridir.

$$\frac{1}{1 + e^{-value}} \tag{13.1}$$

Burada *e* doğal logaritma tabanıdır (Euler sayısı veya elektronik tablonuzdaki EXP() işlevi) ve *değer* dönüştürmek istediğiniz gerçek sayısal değerdir. Aşağıda -5 ile 5 arasındaki sayıların lojistik fonksiyon kullanılarak 0 ile 1 aralığına dönüştürülmüş bir grafiği yer almaktadır.

Artık lojistik fonksiyonun ne olduğunu bildiğimize göre, lojistik regresyonda nasıl kullanıldığını görelim.




Şekil 13.1: Lojistik Fonksiyon.

## 13.2 Lojistik Regresyon için Kullanılan Temsil

Lojistik regresyon, doğrusal regresyona çok benzer şekilde gösterim olarak bir denklem kullanır. Girdi değerleri ($x$), bir çıktı değerini ($y$) tahmin etmek için ağırlıklar veya katsayı değerleri kullanılarak doğrusal olarak birleştirilir. Doğrusal regresyondan önemli bir farkı, modellenen çıktı değerinin sayısal bir değer yerine ikili bir değer (0 veya 1) olmasıdır.

Aşağıda örnek bir lojistik regresyon denklemi verilmiştir:

$$y = \frac{e^{B0+B1\times x}}{1 + e^{B0+B1\times x}} \tag{13.2}$$

Burada $y$ tahmin edilen çıktı, $B0$ sapma veya kesişme terimi ve $B1$ tek girdi değeri ($x$) için katsayıdır. Girdi verilerinizdeki her sütun, eğitim verilerinizden öğrenilmesi gereken ilişkili bir $B$ katsayısına (sabit bir gerçek değer) sahiptir. Modelin bellekte veya bir dosyada saklayacağınız gerçek temsili, denklemdeki katsayılardır (beta değeri veya $B$'ler).

## 13.3 Lojistik Regresyon Olasılıklarını Tahmin Ediyor

Lojistik regresyon varsayılan sınıfın (örneğin birinci sınıf) olasılığını modeller. Örneğin, insanların cinsiyetini boylarına göre erkek veya kadın olarak modelliyorsak, ilk sınıf erkek olabilir ve lojistik regresyon modeli, bir kişinin boyu verildiğinde erkek olma olasılığı olarak veya daha resmi olarak yazılabilir:

$$P\,(cinsiyet = erkek|boy) \tag{13.3}$$

Başka bir şekilde yazarsak, bir girdinin ($X$) varsayılan sınıfa ($Y = 1$) ait olma olasılığını modelliyoruz, bunu resmi olarak şu şekilde yazabiliriz:

$$P\,(X) = P\,(Y = 1|X) \tag{13.4}$$

Olasılıkları mı tahmin ediyoruz? Lojistik regresyonun bir sınıflandırma algoritması olduğunu sanıyordum? Kesin bir tahminde bulunabilmek için olasılık tahmininin ikili değerlere (0 veya 1) dönüştürülmesi gerektiğini unutmayın. Bu konuya daha sonra tahmin yapmaktan bahsederken değineceğiz. Lojistik regresyon doğrusal bir yöntemdir, ancak tahminler lojistik fonksiyon kullanılarak dönüştürülür. Bunun etkisi, tahminleri artık doğrusal regresyonda olduğu gibi girdilerin doğrusal bir kombinasyonu olarak anlayamayacağımızdır, örneğin

Yukarıdaki model şu şekilde ifade edilebilir:

$$p(X) = \frac{e^{B0+B1\times X}}{1 + e^{B0+B1\times X}} \tag{13.5}$$

Matematiğe çok fazla dalmak istemiyorum, ancak yukarıdaki denklemi aşağıdaki gibi çevirebiliriz (diğer tarafa bir *ln*() ekleyerek bir taraftan *e*'yi kaldırabileceğimizi unutmayın):

$$ln(\frac{p(X)}{1 - p(X)}) = \text{B0} + \text{B1} \times X \tag{13.6}$$

Bu kullanışlıdır çünkü sağdaki çıktının hesaplanmasının yine doğrusal olduğunu (tıpkı doğrusal regresyon gibi) ve soldaki girdinin varsayılan sınıfın olasılığının doğal logaritması olduğunu görebiliriz. Soldaki bu orana varsayılan sınıfın olasılığı denir (tarihsel olarak olasılıkları kullanırız, örneğin at yarışlarında olasılıklar yerine oranlar kullanılır). Oranlar, olayın gerçekleşme olasılığının gerçekleşmeme olasılığına bölünmesiyle hesaplanır, Örneğin. $\frac{0.8}{1-0.8}$ Yani bunun yerine şöyle yazabiliriz:

$$ln(odds) = \text{B0} + \text{B1} \times X \tag{13.7}$$

Oranların log dönüşümü yapıldığından, bu sol tarafa log-odds veya probit adını veriyoruz. Dönüşüm için başka türde fonksiyonlar kullanmak mümkündür (ki bu kapsam dışındadır), ancak doğrusal regresyon denklemini olasılıklarla ilişkilendiren dönüşümü bağlantı fonksiyonu olarak adlandırmak yaygındır, örneğin logit bağlantı fonksiyonu. Üsteli sağa kaydırabilir ve şu şekilde yazabiliriz:

$$odds = e^{B0+B1\times X} \tag{13,8}$$

Tüm bunlar, aslında modelin hala girdilerin doğrusal bir kombinasyonu olduğunu, ancak bu doğrusal kombinasyonun varsayılan sınıfın log-odds'iyle ilgili olduğunu anlamamıza yardımcı olur.

## 13.4 Lojistik Regresyon Modelini Öğrenme

Lojistik regresyon algoritmasının katsayıları eğitim verilerinizden tahmin edilmelidir. Bu, maksimum-olabilirlik tahmini kullanılarak yapılır. Maksimum olabilirlik tahmini, çeşitli makine öğrenimi algoritmaları tarafından kullanılan yaygın bir öğrenme algoritmasıdır, ancak verilerinizin dağılımı hakkında varsayımlarda bulunur (verilerinizi hazırlamaktan bahsederken bu konuda daha fazla bilgi vereceğiz).

En iyi katsayılar, varsayılan sınıf için 1'e çok yakın bir değer (örneğin erkek) ve diğer sınıf için 0'a çok yakın bir değer (örneğin kadın) öngören bir modelle sonuçlanacaktır. Lojistik regresyon için maksimum olabilirlik sezgisi, bir arama prosedürünün, model tarafından tahmin edilen olasılıklardaki hatayı verilerdekilere göre en aza indiren katsayılar için değerler aramasıdır (örneğin, veriler birincil sınıf ise 1 olasılığı).

Maksimum olabilirliğin matematiğine girmeyeceğiz. Eğitim verileriniz için katsayıların en iyi değerlerini optimize etmek için bir minimizasyon algoritmasının kullanıldığını söylemek yeterlidir. Bu genellikle pratikte verimli sayısal optimizasyon algoritması (Quasi-newton yöntemi gibi) kullanılarak uygulanır. Lojistik öğrenirken, çok daha basit olan gradyan iniş algoritmasını kullanarak bunu sıfırdan kendiniz uygulayabilirsiniz.

## 13.5 Lojistik Regresyon ile Tahmin Yapma

Lojistik regresyon modeli ile tahmin yapmak, sayıları lojistik regresyon denklemine yerleştirmek ve bir sonuç hesaplamak kadar basittir. Bunu belirli bir örnekle somutlaştıralım.

Diyelim ki bir kişinin boyuna göre (tamamen hayali) kadın mı erkek mi olduğunu tahmin edebilen bir modelimiz var. Boyu 150 cm olan bir kişi erkek midir yoksa kadın mıdır? *B0* katsayılarını öğrendik = -100 ve *B1* = 0,6. Yukarıdaki denklemi kullanarak 150 cm veya daha uzun boylu bir erkeğin olasılığını resmi olarak *P* (*erkek boyu* = 150) hesaplayabiliriz. *e* için EXP() kullanacağız, çünkü bu örneği elektronik tablonuza yazarsanız bunu kullanabilirsiniz:

$$y = \frac{e^{B0+B1\times X}}{1+e^{B0+B1\times X}}$$

$$y = \frac{EXP(-100+0{,}6\times 150)}{1+EXP(-100+0{,}6\times X)} \qquad (13.9)$$

$$y = 0.0000453978687$$

Ya da kişinin erkek olma olasılığı sıfıra yakındır. Pratikte olasılıkları doğrudan kullanabiliriz. Bu bir sınıflandırma olduğundan ve net bir cevap istediğimizden, olasılıkları örneğin ikili bir sınıf değerine bağlayabiliriz:

$$tahmin = 0 \text{ EĞER } p(erkek) < 0{,}5$$
$$tahmin = 1 \text{ EĞER } p(erkek) \geq 0{,}5 \qquad (13.10)$$

Artık lojistik regresyon kullanarak nasıl tahminler yapacağımızı bildiğimize göre, teknikten en iyi şekilde yararlanmak için verilerimizi nasıl hazırlayabileceğimize bakalım.

## 13.6 Verileri Lojistik Regresyon için Hazırlama

Verilerinizdeki dağılım ve ilişkiler hakkında lojistik regresyon tarafından yapılan varsayımlar, doğrusal regresyonda yapılan varsayımlarla hemen hemen aynıdır. Bu varsayımların tanımlanması için çok fazla çalışma yapılmış ve kesin olasılıksal ve istatistiksel dil kullanılmıştır. Benim tavsiyem, bunları kılavuz veya temel kurallar olarak kullanmanız ve farklı veri hazırlama şemaları ile denemeler yapmanızdır. Nihayetinde tahmine dayalı modelleme makine öğrenimi projelerinde sonuçları yorumlamaktan ziyade doğru tahminler yapmaya odaklanırsınız. Bu nedenle, model sağlam olduğu ve iyi performans gösterdiği sürece bazı varsayımları çiğneyebilirsiniz.

**İkili Çıktı Değişkeni:** Daha önce de belirttiğimiz gibi bu açık olabilir, ancak lojistik regresyon ikili (iki sınıflı) sınıflandırma problemleri için tasarlanmıştır. Bir örneğin varsayılan sınıfa ait olma olasılığını tahmin edecektir, bu da 0 veya 1 sınıflandırmasına bölünebilir.

**Gürültüyü Kaldırın**: Lojistik regresyon, çıktı değişkeninde (*y*) hata olmadığını varsayar, eğitim verilerinizden aykırı değerleri ve muhtemelen yanlış sınıflandırılmış örnekleri kaldırmayı düşünün.

**Gauss Dağılımı**: Lojistik regresyon doğrusal bir algoritmadır (çıktı üzerinde doğrusal olmayan bir dönüşüm ile). Girdi değişkenleri ile çıktı arasında doğrusal bir ilişki olduğunu varsayar. Girdi değişkenlerinizin bu doğrusal ilişkiyi daha iyi ortaya çıkaran veri dönüşümleri daha doğru bir modelle sonuçlanabilir. Örneğin, bu ilişkiyi daha iyi ortaya çıkarmak için log, kök, Box-Cox ve diğer tek değişkenli dönüşümleri kullanabilirsiniz.

**İlişkili Girdileri Kaldırın:** Doğrusal regresyonda olduğu gibi, birden fazla yüksek korelasyonlu girdiniz varsa model aşırı uyum sağlayabilir. Tüm girdiler arasındaki ikili korelasyonları hesaplamayı ve yüksek korelasyonlu girdileri kaldırmayı düşünün.

**Yakınsama Başarısızlığı**: Katsayıları öğrenen beklenen olabilirlik tahmin sürecinin yakınsamada başarısız olması mümkündür. Verilerinizde çok sayıda yüksek korelasyonlu girdi varsa veya veriler çok seyrekse (örneğin, girdi verilerinizde çok sayıda sıfır varsa) bu durum ortaya çıkabilir.

## 13.7 Özet

Bu bölümde makine öğrenimi ve tahmine dayalı modelleme için lojistik regresyon algoritmasını keşfettiniz. Çok yol kat ettiniz ve öğrendiniz:

Lojistik fonksiyonun ne olduğu ve lojistik regresyonda nasıl kullanıldığı.

Lojistik regresyondaki temel temsilin, tıpkı doğrusal regresyonda olduğu gibi katsayılar olduğu.

Lojistik regresyondaki katsayıların maksimum olabilirlik tahmini adı verilen bir süreç kullanılarak tahmin edildiğini.

Lojistik regresyon kullanarak tahmin yapmak o kadar kolaydır ki bunu bir elektronik tabloda yapabilirsiniz.

Lojistik regresyon için veri hazırlamanın doğrusal regresyona çok benzediği.

Artık ikili sınıflandırma için lojistik regresyon algoritmasını biliyorsunuz. Bir sonraki bölümde stokastik gradyan inişi kullanarak lojistik regresyonu sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 14

# Lojistik Regresyon Eğitimi

Lojistik regresyon, ikili sınıflandırma için en popüler makine öğrenimi algoritmalarından biridir. Bunun nedeni, çok çeşitli problemlerde çok iyi performans gösteren basit bir algoritma olmasıdır. Bu bölümde ikili sınıflandırma için lojistik regresyon algoritmasını adım adım keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Lojistik fonksiyon nasıl hesaplanır.

Stokastik gradyan inişi kullanarak bir lojistik regresyon modeli için katsayılar nasıl öğrenilir.

Lojistik regresyon modeli kullanarak tahminler nasıl yapılır.

Hadi başlayalım.

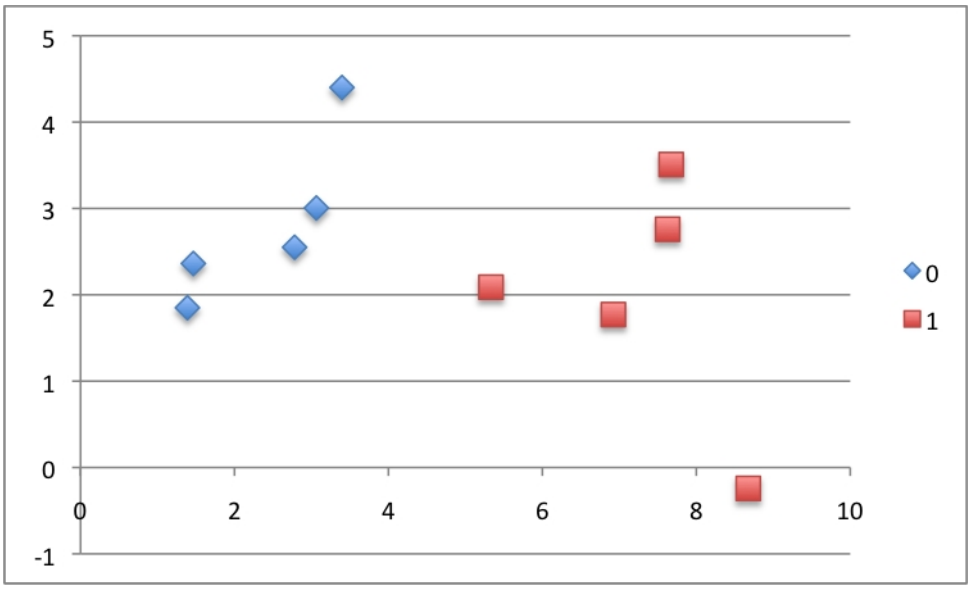
## 14.1 Öğretici Veri Kümesi

Bu eğitimde uydurma bir veri kümesi kullanacağız. Bu veri kümesinde iki girdi değişkeni (*X1* ve *X2*) ve bir çıktı değişkeni (*Y* ) bulunmaktadır. Giriş değişkenleri Gauss dağılımından çekilen gerçek değerli rastgele sayılardır. Çıktı değişkeninin iki değeri vardır, bu da problemi ikili sınıflandırma problemi yapar. Ham veriler aşağıda listelenmiştir.

```
X1            X2            Y
2.7810836     2.550537003   0
1.465489372   2.362125076   0
3.396561688   4.400293529   0
1.38807019    1.850220317   0
3.06407232    3.005305973   0
7.627531214   2.759262235   1
5.332441248   2.088626775   1
6.922596716   1.77106367    1
8.675418651   -0.242068655  1
7.673756466   3.508563011   1
```

Liste 14.1: Öğretici Veri Seti.

Aşağıda veri kümesinin bir grafiği yer almaktadır. Tamamen yapmacık olduğunu ve sınıfları ayırmak için kolayca bir çizgi çizebileceğimizi görebilirsiniz. Lojistik regresyon modeli ile yapacağımız şey tam olarak budur.




Şekil 14.1: Lojistik Regresyon Veri Kümesi.

## 14.2 Lojistik Regresyon Modeli

Lojistik regresyon modeli gerçek değerli girdileri alır ve girdinin varsayılan sınıfa (sınıf 0) ait olma olasılığına ilişkin bir tahmin yapar. Olasılık 0,5'ten büyükse, çıktıyı varsayılan sınıf (sınıf 0) için bir tahmin olarak alabiliriz, aksi takdirde tahmin diğer sınıf (sınıf 1) içindir. Bu veri kümesi için lojistik regresyonun, örneğin doğrusal regresyon gibi üç katsayısı vardır:

$$çıktı = B0 + B1 \times X1 + B2 \times X2 \tag{14.1}$$

Öğrenme algoritmasının görevi, eğitim verilerine dayanarak katsayılar (*B0*, *B1* ve *B2*) için en iyi değerleri keşfetmek olacaktır. Doğrusal regresyonun aksine, çıktı lojistik fonksiyon kullanılarak bir olasılığa dönüştürülür:

$$p(class = 0) = \frac{1}{1 + e^{-çıktı}} \tag{14.2}$$

Hesap tablonuzda bu şu şekilde yazılacaktır:

$$p(sınıf = 0) = \frac{1}{1 + EXP\ (-çıktı)} \tag{14.3}$$

## 14.3 Stokastik Gradyan ile Lojistik Regresyon İniş

Stokastik gradyan inişi kullanarak katsayıların değerlerini tahmin edebiliriz. Lojistik regresyon modeli için katsayıları bulma problemine stokastik gradyan inişi uygulayabiliriz.

### 14.3.1 Tahmini Hesapla

Her katsayıya 0.0 atayarak ve 0 sınıfına ait ilk eğitim örneğinin olasılığını hesaplayarak başlayalım.

$$\begin{aligned} B0 &= 0.0 \\ B1 &= 0.0 \\ B2 &= 0.0 \end{aligned} \tag{14.4}$$

İlk eğitim örneği şöyledir: $X1 = 2.7810836$, $X2 = 2.550537003$, $Y = 0$. Yukarıdaki denklemi kullanarak tüm bu sayıları girebilir ve bir tahmin hesaplayabiliriz:

$$\begin{aligned} tahmin &= \frac{1}{1 + e^{-(B0+B1\times X1+B2\times X2)}} \\ tahmin &= \frac{1}{1 + e^{-(0,0+0,0\times 2,7810836+0,0\times 2,550537003)}} \\ tahmin &= 0,5 \end{aligned} \tag{14.5}$$

### 14.3.2 Yeni Katsayıları Hesaplayın

Basit bir güncelleme denklemi kullanarak yeni katsayı değerlerini hesaplayabiliriz.

$$b = b + alfa \times (y - öngörü) \times öngörü \times (1 - öngörü) \times x \tag{14.6}$$

Burada $b$ güncellediğimiz katsayıdır ve tahmin, modeli kullanarak bir tahmin yapmanın çıktısıdır. *Alfa*, eğitim çalışmasının başında belirtmeniz gereken bir parametredir. Bu, öğrenme oranıdır ve katsayıların (ve dolayısıyla modelin) her güncellendiğinde ne kadar değiştiğini veya öğrendiğini kontrol eder. Daha büyük öğrenme oranları çevrimiçi öğrenmede kullanılır (her eğitim örneği için modeli güncellediğimizde). İyi değerler 0,1 ila 0,3 aralığında olabilir. Şimdi 0,3 değerini kullanalım.

Denklemdeki son terimin $x$ olduğunu fark edeceksiniz, bu katsayı için giriş değeridir. B0'ın bir girdisi olmadığını fark edeceksiniz. Bu katsayı genellikle önyargı veya kesişim olarak adlandırılır ve her zaman 1.0 giriş değerine sahip olduğunu varsayabiliriz. Bu varsayım, algoritmayı vektörler veya diziler kullanarak uygularken yardımcı olabilir. Önceki bölümdeki tahmin (0.5) ve katsayı değerlerini (0.0) kullanarak katsayıları güncelleyelim.

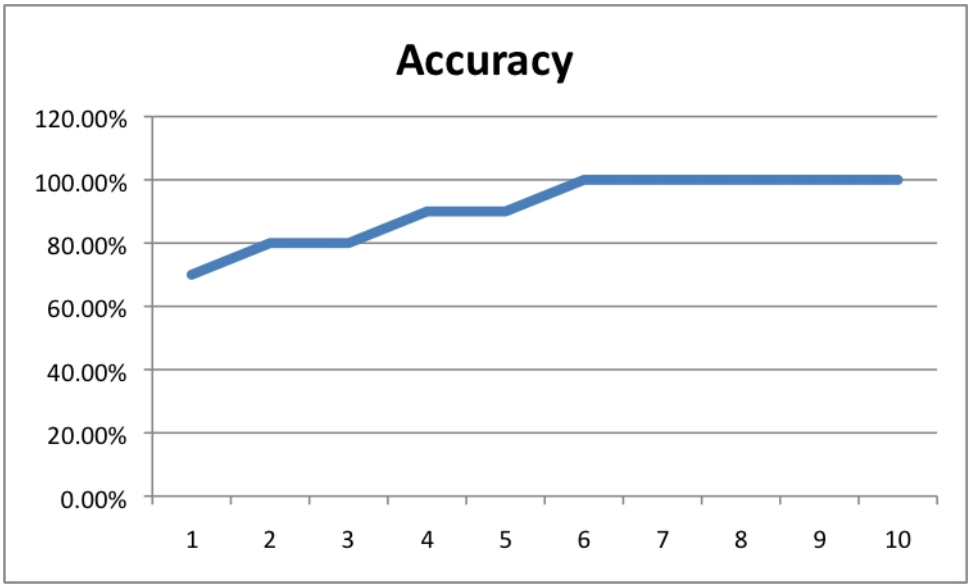
$$\begin{aligned} B0 &= 0.0 + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 1.0 \\ B1 &= B1 + 0,3 \times (0 - 0,5) \times 0,5 \times (1 - 0,5) \times 2,7810836 \\ B2 &= B2 + 0,3 \times (0 - 0,5) \times 0,5 \times (1 - 0,5) \times 2.550537003 \end{aligned} \tag{14.7}$$

veya

$$\begin{aligned} B0 &= -0.0375 \\ B1 &= -0.104290635 \\ B2 &= -0.095645138 \end{aligned} \tag{14.8}$$

### 14.3.3 Süreci Tekrarlayın

Bu işlemi tekrarlayabilir ve veri kümesindeki her eğitim örneği için modeli güncelleyebiliriz. Eğitim veri kümesi üzerinden yapılan tek bir yinelemeye epok denir. Stokastik gradyan inişi prosedürünü sabit sayıda epok için tekrarlamak yaygındır. Epok sonunda model için hata değerlerini hesaplayabilirsiniz. Bu bir sınıflandırma problemi olduğundan, modelin her yinelemede ne kadar doğru olduğuna dair bir fikir edinmek güzel olacaktır. Aşağıdaki grafik, modelin 10 epok üzerindeki doğruluk grafiğini göstermektedir.




Şekil 14.2: Yinelemeye Karşı Gradyan İniş Doğruluğu ile Lojistik Regresyon.

Modelin eğitim veri kümesinde çok hızlı bir şekilde %100 doğruluğa ulaştığını görebilirsiniz. Stokastik gradyan inişinin 10 epokundan sonra hesaplanan katsayılar şunlardır:

$$\begin{aligned} B0 &= -0.406605464 \\ B1 &= 0.852573316 \\ B2 &= -1.104746259 \end{aligned} \tag{14.9}$$

### 14.3.4 Tahminler Yapın

Artık modeli eğittiğimize göre, tahminler yapmak için kullanabiliriz. Eğitim veri kümesi üzerinde tahminler yapabiliriz, ancak bu kolayca yeni veriler de olabilir. Yukarıdaki 10 epoktan sonra öğrenilen katsayıları kullanarak, her eğitim örneği için çıktı değerlerini hesaplayabiliriz:

```
X1              X2              Tahmin
2.7810836       2.550537003     0.298756986
1.465489372     2.362125076     0.145951056
3.396561688     4.400293529     0.085333265
1.38807019      1.850220317     0.219737314
3.06407232      3.005305973     0.247059
7.627531214     2.759262235     0.954702135
5.332441248     2.088626775     0.862034191
6.922596716     1.77106367      0.971772905
```

```
8.675418651    -0.242068655   0.999295452
7.673756466    3.508563011    0.905489323
```

Liste 14.2: Ham Lojistik Regresyon Tahminleri.

Bunlar, her bir örneğin $Y = 0$'a ait olma olasılıklarıdır. Bunları kullanarak net sınıf değerlerine dönüştürebiliriz:

$$tahmin = \text{EĞER } (çıktı < 0{,}5) \text{ Sonra } 0 \text{ Yoksa } 1 \tag{14.10}$$

Bu basit prosedür ile tüm çıktıları sınıf değerlerine dönüştürebiliriz:

```
Tahmin         Gevre
               k
0.298756986    0
0.145951056    0
0.085333265    0
0.219737314    0
0.247059       0
0.954702135    1
0.862034191    1
0.971772905    1
0.999295452    1
0.905489323    1
```

Liste 14.3: Net Lojistik Regresyon Tahminleri.

Son olarak, eğitim veri kümesi üzerinde model için doğruluğu hesaplayabiliriz:

$$\begin{aligned} doğruluk &= \frac{DoğruTahminler}{ToplamTahminler} \times 100 \\ doğruluk &= \frac{10}{10} \times 100 \\ doğruluk &= \%100 \end{aligned} \tag{14.11}$$

# 14.4 Özet

Bu bölümde lojistik regresyonu sıfırdan adım adım nasıl uygulayabileceğinizi keşfettiniz. Öğrendiniz:

Lojistik fonksiyon nasıl hesaplanır.

Stokastik gradyan inişi kullanarak bir lojistik regresyon modeli için katsayılar nasıl öğrenilir.

Lojistik regresyon modeli kullanarak tahminler nasıl yapılır.

Artık stokastik gradyan inişi kullanarak lojistik regresyonu sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde sınıflandırma için doğrusal diskriminant analizi algoritmasını keşfedeceksiniz.

# Bölüm 15

# Doğrusal Diskriminant Analizi

Lojistik regresyon, geleneksel olarak yalnızca iki sınıflı sınıflandırma problemleriyle sınırlı bir sınıflandırma algoritmasıdır. Eğer ikiden fazla sınıfınız varsa, Doğrusal Diskriminant Analizi tercih edilen doğrusal sınıflandırma tekniğidir. Bu bölümde, sınıflandırma tahmin modelleme problemleri için Doğrusal Diskriminant Analizi (LDA) algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz.

Lojistik regresyonun sınırlamaları ve doğrusal diskriminant analizine duyulan ihtiyaç.

Verilerden öğrenilen ve dosyaya kaydedilebilen modelin temsili.

Verilerinizden modelin nasıl tahmin edildiği.

Öğrenilmiş bir LDA modelinden tahminler nasıl yapılır.

LDA modelinden en iyi şekilde yararlanmak için verilerinizi nasıl

hazırlarsınız? Hadi başlayalım.

## 15.1 Lojistik Regresyonun Sınırlamaları

Lojistik regresyon basit ve güçlü bir doğrusal sınıflandırma algoritmasıdır. Ayrıca, alternatif doğrusal sınıflandırma algoritmalarına ihtiyaç duyulduğunu gösteren sınırlamaları da vardır.

**İki Sınıflı Problemler**. Lojistik regresyon iki sınıflı veya ikili sınıflandırma problemleri için tasarlanmıştır. Çok sınıflı sınıflandırma için genişletilebilir, ancak bu amaçla nadiren kullanılır.

**İyi Ayrılmış Sınıflarda Kararsız**. Sınıflar iyi ayrıldığında lojistik regresyon kararsız hale gelebilir.

**Az Örnekle Kararsız**. Lojistik regresyon, parametrelerin tahmin edileceği az sayıda örnek olduğunda kararsız hale gelebilir.

Doğrusal diskriminant analizi bu noktaların her birini ele alır ve çok sınıflı sınıflandırma problemleri için başvurulacak doğrusal yöntemdir. İkili sınıflandırma problemlerinde bile hem lojistik regresyonu hem de doğrusal diskriminant analizini denemek iyi bir fikirdir.

## 15.2 LDA'nın Gösterimi Modeller

LDA'nın gösterimi oldukça basittir. Her sınıf için hesaplanan verilerinizin istatistiksel özelliklerinden oluşur. Tek bir girdi değişkeni ($x$) için bu, her sınıf için değişkenin ortalaması ve varyansıdır.

Birden fazla değişken için bu, çok değişkenli Gauss üzerinden hesaplanan aynı özelliklerdir, yani ortalamalar ve kovaryans matrisi (bu, varyansın çok boyutlu bir genellemesidir). Bu istatistiksel özellikler verilerinizden tahmin edilir ve tahminler yapmak için LDA denklemine eklenir. Bunlar, modeliniz için dosyaya kaydedeceğiniz model değerleridir. Şimdi bu parametrelerin nasıl tahmin edildiğine bakalım.

## 15.3 LDA Modellerini Öğrenme

LDA, verileriniz hakkında bazı basitleştirici varsayımlarda bulunur

Verilerinizin Gauss olduğunu, her bir değişkenin çizildiğinde çan eğrisi şeklinde olduğunu.

Her bir özelliğin aynı varyansa sahip olması, her bir değişkenin değerlerinin ortalama etrafında ortalama olarak aynı miktarda değişiklik göstermesi.

Bu varsayımlarla, LDA modeli her sınıf için verilerinizden ortalama ve varyansı tahmin eder. Bunu iki sınıflı tek değişkenli (tek girdi değişkeni) durumda düşünmek kolaydır. Her bir sınıf ($k$) için her bir girdinin ($x$) *ortalama* (*ortalama*) değeri, değerlerin toplamının toplam değer sayısına bölünmesiyle normal şekilde tahmin edilebilir.

$$ortalama_k = \frac{1}{n_k} \times \sum_{i=1} x_i \tag{15.1}$$

Burada $ortalama_k$ $k$ sınıfı için $x$'in ortalama değeridir, $n_k$ sınıflı örneklerin sayısıdır $k$. Varyans, tüm sınıflar genelinde her bir değerin ortalamadan ortalama karesel farkı olarak hesaplanır.

$$sigma^2 = \frac{1}{\sum - K} \times \sum_{i=1}^{n} (x_i - ortalama_k)^2 \tag{15.2}$$

Burada $sigma^2$ tüm girdiler ($x$) arasındaki varyans, $n$ örnek sayısı, $K$ sınıf sayısı ve $ortalama_k$ $x$'in$_i$ ait olduğu sınıf için $x$'in *ortalamasıdır*. Başka bir deyişle, her bir değerin sınıf grupları içindeki ortalamadan karesel farkını hesaplarız ancak bu farkların tüm sınıf grupları genelinde ortalamasını alırız. Varyans hakkında konuştuğumuzda (*varyans*$^2$ , *sigma*$^2$ veya $\sigma^2$ ) birimlerin kareli birimler olduğunu, varyans değerinin karesini almamız gerekmediğini unutmayın.

## 15.4 LDA ile Tahmin Yapma

LDA, yeni bir girdi kümesinin her bir sınıfa ait olma olasılığını tahmin ederek tahminler yapar. En yüksek olasılığı alan sınıf çıktı sınıfıdır ve bir tahmin yapılır. Model, olasılıkları tahmin etmek için Bayes Teoremini kullanır. Kısaca Bayes Teoremi şöyle ifade edilebilir

her bir sınıfın olasılığını ve her bir sınıfa ait verilerin olasılığını kullanarak girdi ($x$) göz önüne alındığında çıktı sınıfının ($k$) olasılığını tahmin etmek için kullanılır:

$$P(Y = k|X = x) = \frac{P(k) \times P(x|k)}{\sum_{l=1}^{K} (P(l) \times P(x|l))} \quad (15.3)$$

Nerede
?

$P(Y = k|X = x)$, $x$ giriş verisi göz önüne alındığında $Y = k$ sınıfının olasılığıdır.

$P(k)$, dikkate aldığımız belirli bir $k$ sınıfının temel olasılığıdır ($Y = k$), örneğin eğitim veri kümesinde bu sınıfa sahip örneklerin oranı.

$P(x|k)$, $x$'in $k$ sınıfına ait olma tahmini olasılığıdır.

Payda, her bir $l$ sınıfı için, örneğin $P(l)$ sınıfının olasılığı ve $P(x|l)$ sınıfı göz önüne alındığında girdinin olasılığı arasında normalleştirme yapar.

$P(x\ k)$ değerini tahmin etmek için bir Gauss dağılım fonksiyonu kullanılabilir. Gauss'u yukarıdaki denkleme yerleştirir ve basitleştirirsek aşağıdaki denklemi elde ederiz. Bazı terimleri attığımız için bu artık bir olasılık değildir. Bunun yerine $k$ sınıfı için diskriminant fonksiyonu olarak adlandırılır. Her k sınıfı için hesaplanır ve en büyük diskriminant değerine sahip olan sınıf, çıktı sınıflandırmasını ($Y = k$) yapacaktır:

$$D_k(x) = x \times \frac{ortalama_k}{sigma^2} - \frac{ortalama_k^2}{2 \times sigma^2} + ln(P(k)) \quad (15.4)$$

$D_k(x)$, $x$ girdisi verilen $k$ sınıfı için ayrım fonksiyonudur, $ortalama_k$, $sigma^2$ ve $P(k)$ verilerinizden tahmin edilir. $ln()$ fonksiyonu doğal logaritmadır.

## 15.5 Verilerin LDA İçin Hazırlanması

Bu bölümde, verilerinizi LDA ile kullanmak üzere hazırlarken göz önünde bulundurabileceğiniz bazı öneriler listelenmektedir.

**Sınıflandırma Problemleri**. Bunu söylemeye gerek yok, ancak LDA çıktı değişkeninin kategorik olduğu sınıflandırma problemleri için tasarlanmıştır. LDA hem ikili hem de çok sınıflı sınıflandırmayı destekler.

**Gauss Dağılımı**. Modelin standart uygulaması girdi değişkenlerinin Gauss dağılımına sahip olduğunu varsayar. Her bir özelliğin tek değişkenli dağılımlarını gözden geçirmeyi ve bunları daha Gauss görünümlü hale getirmek için dönüşümler kullanmayı düşünün (örneğin, üstel dağılımlar için log ve kök ve çarpık dağılımlar için Box-Cox).

**Aykırı Değerleri Kaldırın**. Verilerinizden aykırı değerleri kaldırmayı düşünün. Bunlar, LDA'da sınıfları ayırmak için kullanılan ortalama ve standart sapma gibi temel istatistikleri çarpıtabilir.

**Aynı Varyans**. LDA, her girdi değişkeninin aynı varyansa sahip olduğunu varsayar. LDA'yı kullanmadan önce verilerinizi standartlaştırmak neredeyse her zaman iyi bir fikirdir, böylece ortalaması 0 ve standart sapması 1 olur.

## 15.6 LDA Uzantıları

Doğrusal Diskriminant Analizi, sınıflandırma için basit ve etkili bir yöntemdir. Basit olması ve çok iyi anlaşılması nedeniyle, yöntemin birçok uzantısı ve varyasyonu vardır. Bazı popüler uzantılar şunlardır:

Kuadratik Diskriminant Analizi: Her sınıf kendi varyans tahminini (veya birden fazla girdi değişkeni olduğunda ortak varyans tahminini) kullanır.

Esnek Diskriminant Analizi: Spline gibi doğrusal olmayan girdi kombinasyonlarının kullanıldığı durumlardır.

Düzenlenmiş Diskriminant Analizi: Varyans (veya kovaryans) tahminine düzenlilik getirerek farklı değişkenlerin LDA üzerindeki etkisini ılımlı hale getirir.

Orijinal geliştirme Doğrusal Diskriminant veya Fisher'in Diskriminant Analizi olarak adlandırılmıştır. Çok sınıflı versiyonu ise Çoklu Diskriminant Analizi olarak adlandırılmıştır. Bunların hepsi artık basitçe Doğrusal Diskriminant Analizi olarak anılmaktadır.

## 15.7 Özet

Bu bölümde, sınıflandırma tahmin modelleme problemleri için Doğrusal Diskriminant Analizini keşfettiniz. Öğrendiniz:

LDA için model gösterimi ve öğrenilen bir model hakkında gerçekte neyin farklı olduğu.

LDA modelinin parametrelerinin eğitim verilerinden nasıl tahmin edilebileceği.

Modelin yeni veriler üzerinde tahminler yapmak için nasıl kullanılabileceği.

Yöntemden en iyi şekilde yararlanmak için verilerinizi nasıl hazırlayacağınız.

Artık sınıflandırma için doğrusal diskriminant analizi algoritması hakkında bilgi sahibisiniz. Bir sonraki bölümde, sınıflandırma için LDA algoritmasını sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 16

# Doğrusal Diskriminant Analizi Eğitimi

Doğrusal Diskriminant Analizi, sınıflandırma tahmin modelleme problemleri için doğrusal bir yöntemdir. Bu bölümde, basit bir veri kümesi için algoritmayı sıfırdan adım adım uygulayarak Doğrusal Diskriminant Analizinin (LDA) nasıl çalıştığını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

LDA tarafından eğitim verileriniz hakkında yapılan varsayımlar.

Verilerinizden LDA modelinin gerektirdiği istatistikler nasıl hesaplanır?

Öğrenilen LDA modeli kullanılarak tahminler nasıl yapılır.

Hadi başlayalım.

## 16.1 Öğretici Genel Bakış

Bir girdi ve bir çıktı değişkenine sahip basit bir veri kümesi için bir LDA modelinin nasıl hesaplanacağını adım adım inceleyeceğiz. Bu, LDA için en basit durumdur. Bu eğitim şunları kapsayacaktır:

1. **Veri kümesi**: Modelleyeceğimiz veri kümesini tanıtın. Eğitim ve test veri kümesi olarak aynı veri kümesini kullanacağız.
2. **Modeli Öğrenme**: Tahmin yapmak için gereken tüm istatistikler dahil olmak üzere veri kümesinden LDA modelinin nasıl öğrenileceği.
3. **Tahmin Yapma**: Eğitim veri kümesindeki her bir örnek için tahminler yapmak üzere öğrenilen modelin nasıl kullanılacağı.

## 16.2 Öğretici Veri Seti

LDA, verileriniz hakkında bazı varsayımlarda bulunur:

Girdi değişkenleri Gauss (çan eğrisi) dağılımına sahiptir.

Sınıf gruplamasına göre her bir girdi değişkeni için hesaplanan varyans (ortalamadan ortalama karesel fark) aynıdır.

Eğitim setinizdeki sınıfların karışımının sorunu temsil etmesi.

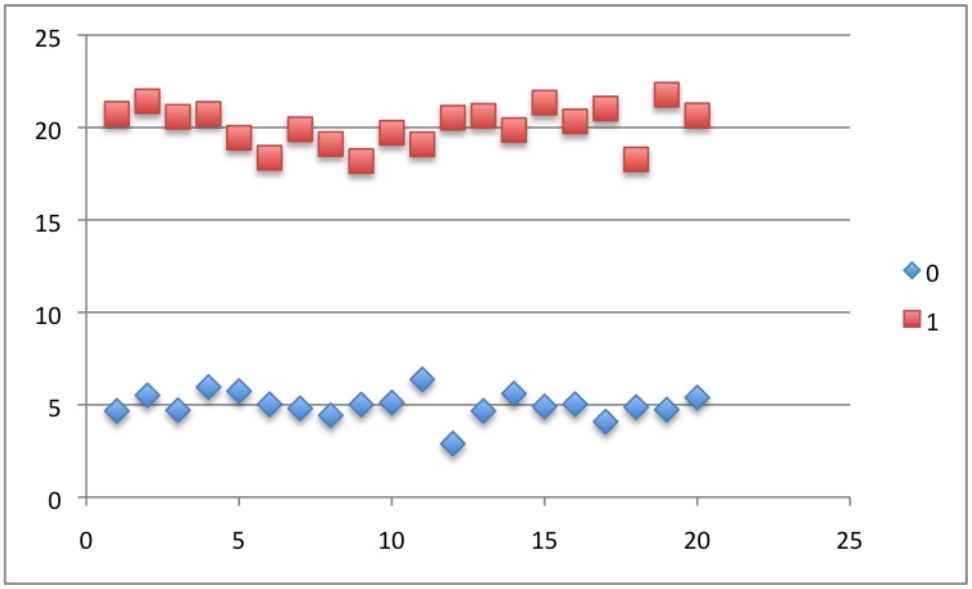
Aşağıda *X* girdi değişkenini ve *Y* çıktı sınıf değişkenini içeren iki boyutlu basit bir veri kümesi bulunmaktadır. *X* için tüm değerler Gauss dağılımından alınmıştır ve *Y* sınıf değişkeni 0 veya 1 değerine sahiptir. Tahmin problemini daha basit hale getirmek için iki sınıftaki örnekler ayrılmıştır. Sınıf 0'daki tüm örnekler ortalaması 5 ve standart sapması 1 olan bir Gauss dağılımından çekilmiştir. Sınıf 1'deki tüm örnekler ortalaması 20 ve standart sapması 1 olan bir Gauss dağılımından çekilmiştir.

Sınıflar etkileşime girmez ve LDA gibi doğrusal bir modelle ayrılabilir olmalıdır. Verilerin gerçek istatistiksel özelliklerini bilmek de kullanışlıdır çünkü daha fazla test üretebiliriz daha sonraki örnekler LDA'nın modeli ne kadar iyi öğrendiğini görmek için. Veri setinin tamamı aşağıdadır.

```
X             Y
4.667797637   0
5.509198779   0
4.702791608   0
5.956706641   0
5.738622413   0
5.027283325   0
4.805434058   0
4.425689143   0
5.009368635   0
5.116718815   0
6.370917709   0
2.895041947   0
4.666842365   0
5.602154638   0
4.902797978   0
5.032652964   0
4.083972925   0
4.875524106   0
4.732801047   0
5.385993407   0
20.74393514   1
21.41752855   1
20.57924186   1
20.7386947    1
19.44605384   1
18.36360265   1
19.90363232   1
19.10870851   1
18.18787593   1
19.71767611   1
19.09629027   1
20.52741312   1
20.63205608   1
19.86218119   1
21.34670569   1
20.333906     1
21.02714855   1
18.27536089   1
21.77371156   1
20.65953546   1
```

Liste 16.1: LDA Öğretici Veri Seti.

Aşağıda, iki sınıfın ayrımını gösteren veri kümesinin bir grafiği yer almaktadır.




Şekil 16.1: LDA Öğretici veri seti.

## 16.3 Modelini Öğrenme

LDA modeli, eğitim verilerinden istatistiklerin tahmin edilmesini gerektirir:

1. Her sınıf için her girdi değerinin ortalaması
2. Bir örneğin her bir sınıfa ait olma olasılığı.
3. Her sınıf için girdi verileri için kovaryans.

### 16.3.1 Sınıf Ortalamalarını Hesaplayın

Ortalama, her sınıf için şu şekilde hesaplanabilir:

$$ortalama(x) = \frac{1}{n} \times \sum_{i=1}^{n} x_i \tag{16.1}$$

Burada $x$ bir sınıf için girdi değerleridir ve $n$ girdi değerlerinin toplam sayısıdır. Elektronik tablonuzda AVERAGE() işlevini kullanabilirsiniz. Her sınıf için $x$'in ortalama değeri şöyledir:

$Y = 0$: 4.975415507

$Y = 1$ 1: 20.06447458

### 16.3.2 Sınıf Olasılıklarını Hesaplayın

Daha sonra, belirli bir örneğin $Y = 0$ veya $Y = 1$'e ait olma olasılığını tahmin etmemiz gerekir. Bu şu şekilde hesaplanabilir:

$$\begin{aligned} P(y = 0) &= \frac{count(y = 0)}{count(y = 0) + count(y = 1)} \\ P(y = 1) &= \frac{count(y = 1)}{count(y = 0) + count(y = 1)} \end{aligned} \tag{16.2}$$

veya

$$\begin{aligned} P(y = 0) &= \frac{20}{20 + 20} \\ P(y = 1) &= \frac{20}{20 + 20} \end{aligned} \tag{16.3}$$

Bu, her sınıf için 0,5'tir. Veri kümesini oluşturduğumuz için bunu zaten biliyorduk, ancak model öğrenme sürecinin her adımında çalışmak iyi bir fikirdir.

### 16.3.3 Varyansı Hesaplayın

Şimdi her sınıf için girdi değişkeninin varyansını hesaplamamız gerekiyor. Varyansı, her bir örneğin ortalamadan farkı olarak anlayabilirsiniz. Farkın karesi alınır, bu nedenle varyans genellikle bu birimleri içerecek şekilde yazılır. Bu, varyans değerini kullanırken karesini almanız gerektiği anlamına gelmez. Veri kümemiz için varyansı iki adımda hesaplayabiliriz:

1. Her bir girdi değişkeni için grup ortalamasından karesel farkı hesaplayın.
2. Karesel farkın ortalamasını hesaplayın.

Veri kümesini 0 ve 1 $Y$ değerlerine göre iki gruba ayırabiliriz. Daha sonra her bir $X$ giriş değeri için bu grubun ortalamasından farkı hesaplayabiliriz. Her bir girdi değerinin ortalamadan farkını şu şekilde hesaplayabiliriz:

$$KareselFark = (x - ortalama)_k^2 \tag{16.4}$$

Burada $ortalama_k$ , $x$'in ait olduğu $k$ sınıfı için $x$'in ortalama değeridir. Bu değerleri toplayabiliriz, bu da bize (4 basamağa yuvarlanmış olarak) verir:

$Y = 0$: 10.15823013

$Y = 1$: 21.49316708

Daha sonra varyansı ortalamadan ortalama karesel fark olarak şu şekilde hesaplayabiliriz:

$$varyans = \frac{1}{count(x) - count(classes)} \times \sum_{i=1}^{n} KareselFark(x_i) \tag{16.5}$$

$$varyans = \frac{1}{20 - 2} \times (10.15823013 + 21.49316708)$$

$$varyans = 0.832931506$$

Kullandığımız iki sınıf değeri veya iki grup için iki serbestlik derecesini çıkarmamız gerektiğinden, bunun yukarıdaki ortalamayı kullanmaktan biraz farklı olduğuna dikkat edin. Bunun tam olarak neden böyle olduğunun açıklaması bu eğitimin ötesinde bir konudur.

## 16.4 Tahminlerde Bulunmak

Artık tahminler yapmaya hazırız. Tahminler, her sınıf için diskriminant fonksiyonu hesaplanarak ve en büyük değere sahip sınıf tahmin edilerek yapılır. Bir girdi ($x$) verilen bir sınıf için diskriminant fonksiyonu şu şekilde hesaplanır:

$$diskriminant(x) = x \times \frac{ortalama}{varyans} - \frac{ortalama^2}{2 \times varyans} + ln(olasılık) \quad (16.6)$$

Burada $x$ giriş değeridir, ortalama, varyans ve olasılık yukarıda ayrıştırdığımız sınıf için hesaplanır. Her sınıf için diskriminant değeri hesaplandıktan sonra, en büyük diskriminant değerine sahip sınıf tahmin olarak alınır. İlk örnek için her bir sınıfın diskriminant değerinin hesaplanmasını adım adım inceleyelim. Veri kümesindeki ilk örnek şöyledir: $X = 4.667797637$ ve $Y = 0$. $Y = 0$ için diskriminant değeri aşağıdaki gibi hesaplanır:

$$diskriminant(Y = 0|x) = 4.667797637 \times \frac{4.975415507}{0.832931506} - \frac{4.975415507^2}{2 \times 0.832931506} + ln(0.5) \quad (16.7)$$
$$diskriminant(Y = 0|x) = 12.3293558$$

$Y = 1$ için diskriminant değerini de hesaplayabiliriz:

$$diskriminant(Y = 1|x) = 4.667797637 \times \frac{20.08706292}{0.832931506} - \frac{20.08706292^2}{2 \times 0.832931506} + ln(0.5) \quad (16.8)$$
$$diskriminant(Y = 1|x) = -130.3349038$$

$Y = 0$ için diskriminant değerinin (12.3293558) $Y = 1$ için diskriminat değerinden (-130.3349038) daha büyük olduğunu görebiliriz, bu nedenle model $Y = 0$'ı tahmin eder. Veri kümesindeki her bir örnek için $Y$ değerlerini aşağıdaki gibi hesaplamaya devam edebilirsiniz:

| X | Disk. Y=0 | Disk. Y=1 | Tahmin |
|---|---|---|---|
| 4.667797637 | 12.3293558 | -130.3349038 | 0 |
| 5.509198779 | 17.35536365 | -110.0435863 | 0 |
| 4.702791608 | 12.53838805 | -129.4909856 | 0 |
| 5.956706641 | 20.02849769 | -99.25144007 | 0 |
| 5.738622413 | 18.72579795 | -104.510782 | 0 |
| 5.027283325 | 14.47670003 | -121.6655095 | 0 |
| 4.805434058 | 13.15151029 | -127.0156496 | 0 |
| 4.425689143 | 10.88315002 | -136.1736175 | 0 |
| 5.009368635 | 14.3696888 | -122.0975421 | 0 |
| 5.116718815 | 15.0109321 | -119.5086739 | 0 |
| 6.370917709 | 22.50273735 | -89.26228283 | 0 |
| 2.895041947 | 1.740014309 | -173.0868646 | 0 |
| 4.666842365 | 12.3236496 | -130.3579413 | 0 |
| 5.602154638 | 17.91062422 | -107.8018531 | 0 |
| 4.902797978 | 13.73310188 | -124.6676111 | 0 |

```
5.032652964    14.50877492    -121.5360148   0
4.083972925    8.841949563    -144.4144814   0
4.875524106    13.57018471    -125.3253507   0
4.732801047    12.7176458     -128.7672748   0
5.385993407    16.61941128    -113.0148198   0
20.74393514    108.3582168    257.3589021    1
21.41752855    112.3818455    273.603351     1
20.57924186    107.3744415    253.3871419    1
20.7386947     108.3269137    257.2325231    1
19.44605384    100.60548      226.0590616    1
18.36360265    94.1395889     199.954556     1
19.90363232    103.3387697    237.0940718    1
19.10870851    98.59038855    217.9236065    1
18.18787593    93.08990662    195.7167121    1
19.71767611    102.2279828    232.6095325    1
19.09629027    98.5162097     217.6241269    1
20.52741312    107.0648488    252.1372346    1
20.63205608    107.6899208    254.6608151    1
19.86218119    103.0911664    236.0944321    1
21.34670569    111.9587938    271.8953795    1
20.333906      105.9089574    247.4705967    1
21.02714855    110.0499579    264.1889062    1
18.27536089    93.61248743    197.8265085    1
21.77371156    114.5094616    282.1930975    1
20.65953546    107.8540656    255.3235107    1
```

Liste 16.2: Veri kümesi için LDA Tahminleri.

Tahminleri veri setiyle karşılaştırırsanız, LDA'nın %100'lük bir doğruluk elde ettiğini görebilirsiniz (hata yok). Veri kümesinin $Y = 0$ ve $Y = 1$ gruplarının açıkça ayrılabilir olması için tasarlandığı düşünüldüğünde bu şaşırtıcı değildir.

## 16.5 Özet

Bu bölümde LDA algoritmasının nasıl çalıştığını basit bir çalışma örneği ile adım adım keşfettiniz. Öğrendikleriniz

LDA modelinin gerektirdiği istatistiklerin veri setinizden nasıl hesaplanacağı.

Her sınıf için bir diskriminant değeri hesaplamak ve bir tahmin yapmak için LDA modelinin nasıl kullanılacağı.

Artık sınıflandırma için sıfırdan doğrusal diskriminant analizi algoritmasının nasıl uygulanacağını biliyorsunuz. Bu, doğrusal makine öğrenimi algoritmalarına girişinizi tamamlıyor. Bir sonraki bölümde karar ağaçları ile başlayarak doğrusal olmayan makine öğrenimi algoritmalarını keşfedeceksiniz.

# Bölüm IV

# Doğrusal Olmayan Algoritmalar

# Bölüm 17

# Sınıflandırma ve Regresyon Ağaçları

Karar Ağaçları, tahminsel modelleme makine öğrenimi için önemli bir algoritma türüdür. Klasik karar ağacı algoritmaları onlarca yıldır kullanılmaktadır ve rastgele orman gibi modern varyasyonları mevcut en güçlü teknikler arasındadır. Bu bölümde, Sınıflandırma ve Regresyon Ağaçları anlamına gelen daha modern adıyla CART olarak bilinen mütevazı karar ağacı algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Makine öğrenimi için CART algoritmasını tanımlamak için kullanılan birçok isim.

Öğrenilen CART modelleri tarafından kullanılan ve aslında diskte depolanan temsil.

Bir CART modeli eğitim verilerinden nasıl öğrenilebilir?

Öğrenilmiş bir CART modelinin görünmeyen veriler üzerinde tahminler yapmak için nasıl kullanılabileceği.

CART ve ilgili algoritmalar hakkında daha fazla bilgi edinmek için kullanabileceğiniz ek kaynaklar.

Eğer bir algoritma ve veri yapıları dersi aldıysanız, bu basit ve güçlü algoritmayı uygulamaktan sizi alıkoymak zor olabilir. Ve oradan, kendi Rastgele Orman uygulamanızdan küçük bir adım uzaktasınız. Hadi başlayalım.

## 17.1 Karar Ağaçlar

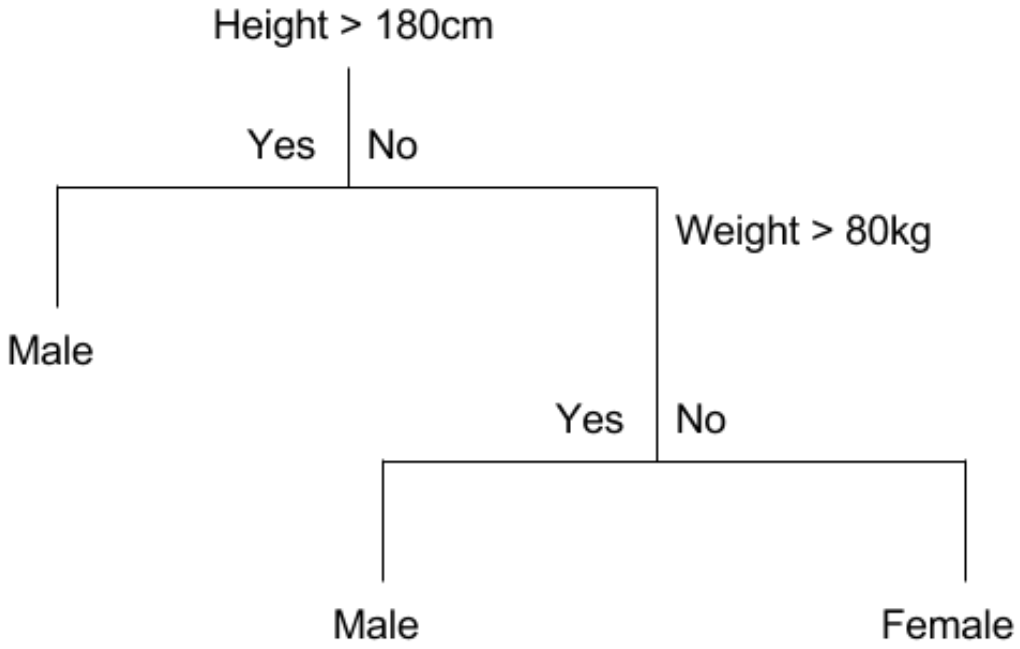
Sınıflandırma ve Regresyon Ağaçları veya kısaca CART, Leo Breiman tarafından sınıflandırma veya regresyon tahmini modelleme problemleri için kullanılabilen Karar Ağacı algoritmalarına atıfta bulunmak için ortaya atılan bir terimdir. Klasik olarak bu algoritma *karar ağaçları* olarak adlandırılır, ancak R gibi bazı platformlarda daha modern bir terim olan CART ile anılırlar. CART algoritması, torbalanmış karar ağaçları, rastgele orman ve güçlendirilmiş karar ağaçları gibi önemli algoritmalar için bir temel sağlar.

## 17.2 CART Modeli Temsil

CART modelinin temsili bir ikili ağaçtır. Bu, algoritmalardan ve veri yapılarından gelen ikili ağacınızdır, çok süslü bir şey değildir. Her düğüm tek bir girdi değişkenini ($x$) ve bu değişken üzerindeki bir bölünme noktasını temsil eder (değişkenin sayısal olduğu varsayılırsa). Ağacın

yaprak düğümleri

bir tahmin yapmak için kullanılan bir çıktı değişkeni (*y*) içerir. İki girdisi santimetre cinsinden boy ve kilogram cinsinden ağırlık, çıktısı ise erkek veya kadın cinsiyeti olan bir veri kümesi göz önüne alındığında, aşağıda ikili bir karar ağacının kaba bir örneği verilmiştir (yalnızca gösterim amacıyla tamamen kurgusaldır).




Şekil 17.1: Örnek Karar Ağacı.

Ağaç, dosyaya bir grafik veya bir kurallar kümesi olarak kaydedilebilir. Örneğin, aşağıda yukarıdaki karar ağacı bir kurallar kümesi olarak verilmiştir.

```
Boy > 180 cm ise Erkek
Boy <= 180 cm VE Kilo > 80 kg ise Erkek Boy <= 180
cm VE Kilo <= 80 kg ise Kadın CART Modelleri ile
Tahmin Yapın
```

Liste 17.1: Karar Ağacının Kural Gösterimi Örneği.

## 17.3 Tahminlerde Bulunmak

Yukarıda açıklanan CART modelinin ikili ağaç temsili ile tahminler yapmak nispeten kolaydır. Yeni bir girdi verildiğinde, ağacın kök düğümünde başlatılan belirli girdi değerlendirilerek ağaçta dolaşılır. Öğrenilmiş bir ikili ağaç aslında girdi uzayının bir bölümlemesidir. Her bir girdi değişkenini p-boyutlu bir uzayda bir boyut olarak düşünebilirsiniz. Karar ağacı bunu dikdörtgenlere ($p$ = 2 girdi değişkeni olduğunda) veya daha fazla girdiye sahip hiper-dikdörtgenlere böler. Yeni veriler ağaç boyunca filtrelenir ve dikdörtgenlerden birine düşer ve

bu dikdörtgen için çıktı değeri, model tarafından yapılan tahmindir. Bu size bir CART modelinin verebileceği kararların türü hakkında bir fikir verir, örneğin kutu gibi karar sınırları. Örneğin, *boy* = 160 cm ve *ağırlık* = 65 kg girdisi verildiğinde, yukarıdaki ağacı aşağıdaki şekilde dolaşırız:

1. Yükseklik > 180 cm: Hayır
2. Ağırlık > 80 kg: Hayır
3. Bu nedenle: Kadın

# 17.4 Verilerinden Bir CART Modeli Öğrenin

İkili bir karar ağacı oluşturmak aslında girdi uzayını bölme işlemidir. Uzayı bölmek için özyinelemeli ikili bölme adı verilen açgözlü bir yaklaşım kullanılır. Bu, tüm değerlerin sıralandığı ve farklı bölme noktalarının bir maliyet fonksiyonu kullanılarak denendiği ve test edildiği sayısal bir prosedürdür. En iyi maliyete (maliyeti en aza indirdiğimiz için en düşük maliyet) sahip bölme seçilir. Tüm girdi değişkenleri ve tüm olası bölme noktaları değerlendirilir ve açgözlü bir şekilde seçilir (örneğin, her seferinde en iyi bölme noktası seçilir). Regresyon tahmini modelleme problemleri için, ayrık noktaları seçmek üzere minimize edilen maliyet fonksiyonu, dikdörtgenin içine düşen tüm eğitim örneklerinin karesel hata toplamıdır:

$$\sum_{i=1} (y_i - tahmin)_i^2 \tag{17.1}$$

Burada *y* eğitim örneği için çıktıdır ve tahmin dikdörtgen için öngörülen çıktıdır. Sınıflandırma için yaprak düğümlerinin ne kadar *saf* olduğunu (her bir düğüme atanan eğitim verilerinin ne kadar karışık olduğunu) gösteren Gini maliyet fonksiyonu kullanılır.

$$G = \sum_{k=1} p_k \times (1 - p)_k \tag{17.2}$$

Burada G tüm sınıflar üzerindeki Gini maliyetidir, $p_k$ ilgilenilen dikdörtgende *k* sınıfına sahip eğitim örneklerinin sayısıdır. Tüm sınıfları aynı türde olan bir düğüm (mükemmel sınıf saflığı) $G = 0$ olurken, ikili sınıflandırma problemi için sınıfların yarı yarıya bölündüğü bir *G* (en kötü saflık) $G = 0{,}5$ olacaktır.

## 17.4.1 Durdurma Kriteri

Yukarıda açıklanan özyinelemeli ikili bölme prosedürünün, eğitim verileriyle ağaçta aşağı doğru ilerlerken bölmeyi ne zaman durduracağını bilmesi gerekir. En yaygın durdurma prosedürü, her bir yaprak düğüme atanan eğitim örneklerinin sayısında minimum bir sayı kullanmaktır. Eğer sayı belli bir minimumdan azsa, bölünme kabul edilmez ve düğüm son yaprak düğüm olarak alınır. Eğitim üyelerinin sayısı veri kümesine göre ayarlanır, örneğin 5 veya 10. Ağacın eğitim verilerine ne kadar özel olacağını tanımlar. Çok spesifik (örneğin 1 sayısı) ve ağaç eğitim verilerine aşırı uyacak ve muhtemelen test setinde düşük performans gösterecektir.

### 17.4.2 Ağacın Budanması

Durdurma kriteri, ağacınızın performansını büyük ölçüde etkilediği için önemlidir. Performansı daha da artırmak için ağacınızı öğrendikten sonra budama işlemini kullanabilirsiniz. Bir karar ağacının karmaşıklığı, ağaçtaki bölünme sayısı olarak tanımlanır. Daha basit ağaçlar tercih edilir. Anlaşılmaları kolaydır (çıktılarını alıp konu uzmanlarına gösterebilirsiniz) ve verilerinize aşırı uyma olasılıkları daha düşüktür.

En hızlı ve en basit budama yöntemi, ağaçtaki her bir yaprak düğümü üzerinde çalışmak ve bir bekletme test seti kullanarak çıkarmanın etkisini değerlendirmektir. Yaprak düğümler yalnızca tüm test setinde genel maliyet fonksiyonunda bir düşüşe neden oluyorsa kaldırılır. Daha fazla iyileştirme yapılamadığında düğümleri çıkarmayı bırakırsınız. Alt ağacın boyutuna bağlı olarak düğümlerin kaldırılıp kaldırılamayacağını tartmak için bir öğrenme parametresinin (*alfa*) kullanıldığı maliyet karmaşıklığı budaması (en zayıf bağlantı budaması olarak da adlandırılır) gibi daha gelişmiş budama yöntemleri kullanılabilir.

## 17.5 Verilerin CART İçin Hazırlanması

CART, problemin iyi bir temsili dışında herhangi bir özel veri hazırlığı gerektirmez.

## 17.6 Özet

Bu bölümde makine öğrenimi için Sınıflandırma ve Regresyon Ağaçlarını (CART) keşfettiniz. Öğrendiniz:

Algoritmanın klasik adı Karar Ağacı, daha modern adı ise CART'tır.

CART için kullanılan gösterim ikili bir ağaçtır.

CART ile tahminler, yeni bir girdi kaydı verildiğinde ikili ağacın çaprazlanmasıyla yapılır.

Ağaç, ağaçtaki bölünmeleri seçmek için eğitim verileri üzerinde açgözlü bir algoritma kullanılarak öğrenilir

Durdurma kriterleri, bir ağacın ne kadar öğrendiğini tanımlar ve budama, öğrenilen bir ağaçta genellemeyi iyileştirmek için kullanılabilir.

Artık sınıflandırma ve regresyon için Sınıflandırma ve Regresyon Ağaçları makine öğrenimi algoritmasını biliyorsunuz. Bir sonraki bölümde CART'ı sıfırdan nasıl uygulayabileceğinizi keşfedeceksiniz.

# Bölüm 18

# Sınıflandırma ve Regresyon Ağaçları Eğitimi

Karar ağaçları, regresyon ve sınıflandırma tahmin modelleme problemleri için esnek ve çok güçlü bir makine öğrenimi yöntemidir. Bu bölümde CART makine öğrenimi algoritmasını sıfırdan adım adım nasıl uygulayacağınızı keşfedeceksiniz. Bu bölümü tamamladıktan sonra şunları öğreneceksiniz:

Bir karar ağacında belirli bir bölünme için Gini endeksi nasıl hesaplanır.

Bir karar ağacı oluştururken farklı bölünme noktaları nasıl değerlendirilir?

Öğrenilmiş bir karar ağacı ile yeni veriler üzerinde tahminler nasıl

yapılır. Hadi başlayalım.

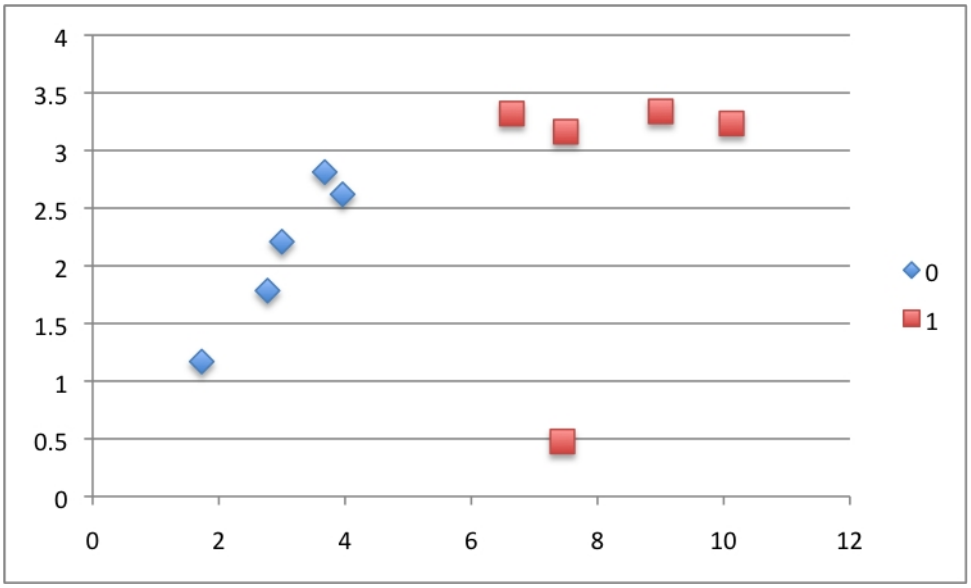
## 18.1 Öğretici Veri Seti

Bu eğitimde CART için basit bir ikili (iki sınıflı) sınıflandırma problemi üzerinde çalışacağız. İşleri basit tutmak için iki girdi değişkeni (X1 ve *X2*) ve tek bir çıktı değişkeni (*Y* ) ile çalışacağız. Bu gerçek bir problem değil, CART modelinin nasıl uygulanacağını ve tahminlerin nasıl yapılacağını göstermek için uydurulmuş bir problemdir. Örnek, algoritmanın eğitim veri setini en iyi şekilde sınıflandırmak için en az iki ayrık nokta bulacağı şekilde tasarlanmıştır. Bu problem için ham veriler aşağıdaki gibidir:

```
X1            X2            Y
2.771244718   1.784783929   0
1.728571309   1.169761413   0
3.678319846   2.81281357    0
3.961043357   2.61995032    0
2.999208922   2.209014212   0
7.497545867   3.162953546   1
9.00220326    3.339047188   1
7.444542326   0.476683375   1
10.12493903   3.234550982   1
6.642287351   3.319983761   1
```

Liste 18.1: CART Öğretici Veri Seti.

İki boyutlu bir dağılım grafiği olarak görselleştirildiğinde veri kümesi aşağıdaki gibi görünür:




Şekil 18.1: Sınıflandırma ve Regresyon Ağaçları Öğretici Veri Kümesi.

## 18.2 Bir CART Modeli Öğrenme

CART modeli, verilerdeki bölünme noktaları aranarak öğrenilir. Bölünme noktası, tek bir özniteliğin tek bir değeridir, örneğin *X1* özniteliğinin ilk değeri 2.771244718. Bir bölünme noktasındaki verilerin bölümlenmesi, o düğümdeki tüm verilerin bölünme noktasının solunda ve bölünme noktasının sağında olmak üzere iki gruba ayrılmasını içerir. Ağaçtaki ilk bölünme noktası üzerinde çalışıyorsak, veri kümesinin tamamı etkilenir. Diyelim ki bir seviye derindeki bir bölünme noktası üzerinde çalışıyorsak, yalnızca yukarıdaki düğümlerden ağaca süzülen ve o düğümde bulunan veriler bölünme noktasından etkilenir.

Seçilen ayırma noktasının sınıf değerinin ne olduğuyla ilgilenmiyoruz. Sadece bölünme noktasının SOL ve SAĞ alt düğümlerine atanan verilerin bileşimiyle ilgileniyoruz. Bölünmenin her bir tarafına atanan eğitim verilerinin sınıflarının karışımını değerlendirmek için bir maliyet fonksiyonu kullanılır. Sınıflandırma problemlerinde Gini endeksi maliyet fonksiyonu kullanılır.

### 18.2.1 Gini Endeksi Maliyet Fonksiyonu

Bir bölünme noktası için Gini endeksini aşağıdaki şekilde hesaplıyoruz:

$$Gini(\text{bölünmüş}) = \sum_{k=1} (p_k \times (1 - p_k)) \quad (18.1)$$

Her sınıf ($k$) için, her grup (sol ve sağ) için. Burada $p$, belirli bir grupta belirli bir sınıfa sahip eğitim örneklerinin oranıdır. İkili bir ağaç kullandığımız için her zaman sol ve sağ olmak üzere iki grubumuz olacaktır. Ve veri setimizden sadece iki sınıfımız olduğunu biliyoruz. Bu nedenle, veri setimizdeki herhangi bir bölünme noktasının Gini endeksini aşağıdakilerin toplamı olarak hesaplayabiliriz:

$$\begin{aligned} Gini(\text{bölünmüş}) = {} & (sol(0) \times (1 - sol(0))) + (sağ(0) \times (1 - sağ(0)) \\ & + (sol(1) \times (1 - sol(1))) + (sağ(1) \times (1 - sağ(1)) \end{aligned} \quad (18.2)$$

Burada sol(0), sol gruptaki 0 sınıflı veri örneklerinin oranıdır; sağ(0), sağ gruptaki 0 sınıflı veri örneklerinin oranıdır ve bu böyle devam eder. Bir sınıfa ait veri örneklerinin oranını hesaplamak kolaydır. Eğer bir SOL grupta 0 sınıfına sahip 3 örnek varsa ve sınıf 1'e sahip 4 örnek varsa, sınıf 0'a sahip veri örneklerinin oranı $\frac{3}{7}$ veya 0.428571429. Gini endeksi puanları hakkında fikir edinmek için aşağıdaki tabloya bir göz atın. Tek bir gruptaki 0 ve 1 sınıflarının karışımları için 7 farklı senaryo sunmaktadır.

| Sınıf 0 | Sınıf 1 | Saymak | Sınıf 0/Sayı | Sınıf 1/Sayı | Gini |
|---|---|---|---|---|---|
| 10 | 10 | 20 | 0.5 | 0.5 | 0.5 |
| 19 | 1 | 20 | 0.95 | 0.05 | 0.095 |
| 1 | 19 | 20 | 0.05 | 0.95 | 0.095 |
| 15 | 5 | 20 | 0.75 | 0.25 | 0.375 |
| 5 | 15 | 20 | 0.25 | 0.75 | 0.375 |
| 11 | 9 | 20 | 0.55 | 0.45 | 0.495 |
| 20 | 0 | 20 | 1 | 0 | 0 |

Liste 18.2: Örnek Gini hesaplamaları.

Bu senaryoları tek bir grup için görselleştirmek kolaydır, ancak kavramlar SOL ve SAĞ gruplardaki oranların toplanmasına dönüşür. İlk satırda grubun 50-50 karışımına sahip olduğu durumda Gini'nin 0,5 olduğunu görebilirsiniz. Bu mümkün olan en kötü bölünmedir. Ayrıca, grubun son satırda yalnızca 0 sınıfına sahip veri örneklerine sahip olduğu ve Gini endeksinin 0 olduğu bir durumu da görebilirsiniz. Bu mükemmel bir bölünme örneğidir. Bir bölme noktası seçerken amacımız, tüm olası bölme noktalarının Gini endeksini değerlendirmek ve açgözlülükle en düşük maliyetli bölme noktasını seçmektir. Bölme noktamız olarak farklı veri noktalarını seçmenin maliyetini hesaplayarak bunu daha somut hale getirelim.

### 18.2.2 Birinci Aday Bölünme Noktası

İlk adım, karar ağacımızın kütüğü veya kök düğümü olacak bir bölünme seçmektir. İlk aday bölme noktası olan X1 niteliği ve ilk örnekte X1'in değeri ile başlayacağız: *X1* = 2.771244718.

X1 < 2.771244718 **ISE** SOL

X1 ≥ 2.771244718 **ISE** DOĞRU

Bu kuralı eğitim veri setimizdeki her bir *X1* değerine uygulayalım. Veri kümesindeki her bir numaralı örnek için elde ettiğimiz yanıt aşağıdadır:

| X1 | Y | Grup |
|---|---|---|
| 2.771244718 | 0 | DOĞRU |
| 1.728571309 | 0 | SOL |
| 3.678319846 | 0 | DOĞRU |
| 3.961043357 | 0 | DOĞRU |
| 2.999208922 | 0 | DOĞRU |
| 7.497545867 | 1 | DOĞRU |
| 9.00220326 | 1 | DOĞR |
| 7.444542326 | 1 | U |
| 10.12493903 | 1 | SAĞ |
| 6.642287351 | 1 | SAĞ |
| | | SAĞ |

Liste 18.3: Eğitim Veri Kümesinin Aday Bölme ile Ayrılması.

Bu bölünme ne kadar iyiydi? SOL ve SAĞ düğümlerin her birindeki sınıfların karışımını, kök düğümümüz için bu bölme noktasını seçmenin tek bir maliyeti olarak değerlendirebiliriz. SOL grubunun yalnızca bir üyesi varken SAĞ grubunun 9 üyesi vardır. SOL gruptan başlayarak, her bir sınıfa sahip eğitim örneklerinin oranını hesaplayabiliriz:

$Y = 0$: $\frac{1}{1}$ veya 1.0

$Y = 1$: $\frac{0}{1}$ veya 0.0

SAĞ grup, sınıfların bir karışımından oluştuğu için daha ilginçtir (muhtemelen yüksek bir Gini endeksi elde edeceğiz).

$Y = 0$: $\frac{7}{9}$ veya 0.444444444

$Y = 1$: $\frac{5}{9}$ veya 0.555555556

Artık bu bölünme için Gini endeksini hesaplamak için yeterli bilgiye sahibiz:

$$Gini(X1 = 2.7712) = (1.0 \times (1 - 1.0)) + (0.0 \times (1 - 0.0)) + (0.444444444 \times (1 - 0.444444444) + (0.555555556 \times (1 - 0.555555556)) \quad (18.3)$$

veya

$$Gini(X1 = 2.771244718) = 0.49382716 \quad (18.4)$$

### 18.2.3 En İyi Aday Bölünme Noktası

Her bir aday ayırma noktasını *X1* ve *X2* değerleriyle yukarıdaki süreci kullanarak değerlendirebiliriz. Verilerin grafiğine bakarsak, sınıfları ayırmak için muhtemelen dikey bir çizgi çizebileceğimizi görebiliriz. Bu, *X1* için yaklaşık bir değere sahip bir ayırma noktasına dönüşecektir
0.5. Son örnekteki X1 değeri yakın bir uyum sağlayacaktır: X1 = 6.642287351.

X1 < 6.642287351 **ISE** SOL

X1 ≥ 6.642287351 **ISE** DOĞRU

Bu kuralı tüm örneklere uygulayalım, aşağıda her bir numaralı veri örneği için atanmış grubu alıyoruz:

```
X1             Y   Grup
2.771244718    0   SOL
1.728571309    0   SOL

9.00220326     1   DOĞR
7.444542326    1   U
10.12493903    1   SAĞ
6.642287351    1   SAĞ
                   SAĞ
```

Liste 18.4: Eğitim Veri Kümesinin Bahis Bölünmesine Göre Ayrılması.

```
3.678319846    0    SOL
3.961043357    0    SOL
2.999208922    0    SOL
7.497545867    1   DOĞRU
```

```
9.00220326     1   DOĞR
7.444542326    1   U
10.12493903    1   SAĞ
6.642287351    1   SAĞ
                   SAĞ
```

Liste 18.4: Eğitim Veri Kümesinin Bahis Bölünmesine Göre Ayrılması.

Her grupta 5 örnek vardır, bu iyi bir dağılım gibi görünmektedir. SOL gruptan başlayarak, her bir sınıfa sahip eğitim örneklerinin oranını hesaplayabiliriz:

$Y = 0$: $\frac{5}{5}$ veya 1.0

$Y = 1$: $\frac{0}{5}$ veya 0.0

SAĞ grup ise tam tersi oranlara sahiptir.

$Y = 0$: $\frac{0}{5}$ veya 0.0

$Y = 1$: $\frac{5}{5}$ veya 1.0

Artık bu bölünme için Gini endeksini hesaplamak için yeterli bilgiye sahibiz:

$$\begin{aligned} Gini(\text{X1} = 6{,}642287351) = &(1{,}0 \times (1 - 1{,}0)) + \\ &(0.0 \times (1 - 0.0)) + \\ &(1.0 \times (1 - 1.0)) + \\ &(0.0 \times (1 - 0.0)) \end{aligned} \tag{18.5}$$

veya

$$Gini(\text{X1} = 6.642287351) = 0.0 \tag{18.6}$$

Bu, sınıflar mükemmel bir şekilde ayrıldığı için saf bir Gini endeksi ile sonuçlanan bir bölünmedir. SOL alt düğüm örnekleri 0. sınıf ve sağ alt düğüm de 1. sınıf olarak sınıflandıracaktır. Burada durabiliriz. Daha zorlu bir veri kümesi için daha karmaşık bir ağaç oluşturmak üzere bu işlemin her bir alt düğüm için nasıl tekrarlanabileceğini görebilirsiniz.

## 18.3 Verileri Üzerinden Tahmin Yapma

Şimdi bu karar ağacını tüm eğitim örnekleri için bazı tahminler yapmak üzere kullanabiliriz. Ancak yukarıda Gini endeksini hesapladığımızda bunu zaten yapmıştık. Bunun yerine, aynı dağılımı kullanarak her sınıf için oluşturulan bazı yeni verileri sınıflandıralım. İşte test veri kümesi:

```
X1            X2            Y
2.343875381   2.051757824   0
3.536904049   3.032932531   0
2.801395588   2.786327755   0
3.656342926   2.581460765   0
2.853194386   1.052331062   0
8.907647835   3.730540859   1
9.752464513   3.740754624   1
8.016361622   3.013408249   1
6.58490395    2.436333477   1
7.142525173   3.650120799   1
```

Liste 18.5: Test Veri Kümesi.

X1 = 6.642287351 değerinde tek bir bölünme ile karar ağacını kullanarak test örneklerini aşağıdaki gibi sınıflandırabiliriz:

```
Y   Tahmin
0   0
0   0
0   0
0   0
0   0
1   1
1   1
1   1
1   0
1   1
```

Liste 18.6: Test Veri Kümesi için Tahminler.

Yine, bu mükemmel bir sınıflandırmadır veya %100 doğrudur.

## 18.4 Özet

Bu bölümde bir CART modelinin tam olarak nasıl oluşturulacağını ve tahminler yapmak için nasıl kullanılacağını keşfettiniz. Nasıl yapılacağını öğrendiniz:

Bir karar ağacında aday bir bölünme için Gini endeksini hesaplayın.

Gini endeksini kullanarak herhangi bir aday bölünme noktasını değerlendirin.

Tahminler yapmak için sıfırdan bir karar ağacı nasıl oluşturulur?

Öğrenilmiş bir karar ağacı kullanarak yeni veriler üzerinde tahminler nasıl yapılır?

Artık CART'ı sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde sınıflandırma için Naive Bayes makine öğrenimi algoritmasını keşfedeceksiniz.

# Bölüm 19 Naive

# Bayes

Naive Bayes, tahminsel modelleme için basit ama şaşırtıcı derecede güçlü bir algoritmadır. Bu bölümde sınıflandırma için Naive Bayes algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Naive Bayes tarafından kullanılan ve bir model bir dosyaya yazıldığında gerçekte depolanan temsil.

Öğrenilmiş bir modelin tahminlerde bulunmak için nasıl kullanılabileceği.

Eğitim verilerinden bir naif Bayes modelini nasıl öğrenebilirsiniz?

Verilerinizi naif Bayes algoritması için en iyi şekilde nasıl hazırlayabilirsiniz?

Naif Bayes hakkında daha fazla bilgi için nereye

gitmeli. Hadi başlayalım.

## 19.1 Bayes'in Teoremine Hızlı Giriş

Makine öğreniminde genellikle veri (*d*) verildiğinde en iyi hipotezi (*h)* seçmekle ilgileniriz. Bir sınıflandırma probleminde, hipotezimiz (*h*) yeni bir veri örneği için atanacak sınıf olabilir (*d*). Problemle ilgili ön bilgimiz olarak kullanabileceğimiz elimizdeki veriler göz önüne alındığında en olası hipotezi seçmenin en kolay yollarından biri. Bayes Teoremi, ön bilgilerimiz göz önüne alındığında bir hipotezin olasılığını hesaplayabileceğimiz bir yol sağlar. Bayes Teoremi şu şekilde ifade edilir:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)} \tag{19.1}$$

Nerede
?

$P(h|d)$, $d$ verileri göz önüne alındığında $h$ hipotezinin olasılığıdır.

$P(d|h)$, $h$ hipotezinin doğru olması durumunda $d$ verisinin olasılığıdır.

$P(h)$, h hipotezinin doğru olma olasılığıdır (verilerden bağımsız olarak). Buna $h$'nin önceki olasılığı denir.

$P(d)$ verilerin olasılığıdır (hipotezden bağımsız olarak).

$P(D)$ ve $P(d\ h)$ ile önceki olasılık $p(h)$'den $P(h\ d)$'nin sonsal olasılığını hesaplamakla ilgilendiğimizi görebilirsiniz. Bir dizi farklı hipotez için sonsal olasılığı hesapladıktan sonra, en yüksek olasılığa sahip hipotezi seçebilirsiniz. Bu maksimum olası hipotezdir ve resmi olarak maksimum a posteriori (MAP) hipotezi olarak adlandırılabilir. Bu şu şekilde yazılabilir:

$$\begin{aligned} MAP(h) &= maks(P(h|d)) \\ MAP(h) &= maks(\frac{P(d|h) \times P(h)}{P(d)}) \\ MAP(h) &= maks(P(d|h) \times P(h)) \end{aligned} \tag{19.2}$$

$P(d)$, olasılığı hesaplamamızı sağlayan normalleştirici bir terimdir. Sabit olduğu ve yalnızca normalleştirmek için kullanıldığı için en olası hipotezle ilgilendiğimizde bunu bırakabiliriz. Sınıflandırmaya geri dönersek, eğitim verilerimizde her sınıfta eşit sayıda örneğimiz varsa, her sınıfın olasılığı (örneğin $P(h)$) her sınıf için aynı değerde olacaktır (örneğin 50-50 bölünme için 0,5). Yine, bu denklemimizde sabit bir terim olacaktır ve bunu düşürebiliriz, böylece şu sonuca ulaşırız:

$$MAP(h) = maks(P(d|h)) \tag{19.3}$$

Bu yararlı bir alıştırmadır, çünkü Naive Bayes hakkında daha fazla okuma yaparken teoremin tüm bu biçimlerini görebilirsiniz.

# 19.2 Naive Bayes Sınıflandırıcı

Naive Bayes, ikili (iki sınıflı) ve çok sınıflı sınıflandırma problemleri için bir sınıflandırma algoritmasıdır. Teknik, ikili veya kategorik girdi değerleri kullanılarak tanımlandığında anlaşılması en kolay yöntemdir. Naive Bayes veya idiot Bayes olarak adlandırılır çünkü her bir hipotez için olasılıkların hesaplanması, hesaplanmalarını izlenebilir hale getirmek için basitleştirilmiştir. Her bir öznitelik değerinin $P(d1, d2, d3\ h)$ değerlerini hesaplamaya çalışmak yerine, hedef değer verildiğinde bunların koşullu olarak bağımsız olduğu varsayılır ve $P(d1\ h) \times P(d2\ h)$ ve benzeri şekilde hesaplanır. Bu, gerçek verilerde pek olası olmayan, yani niteliklerin etkileşime girmediği çok güçlü bir varsayımdır. Bununla birlikte, yaklaşım bu varsayımın geçerli olmadığı verilerde şaşırtıcı derecede iyi performans göstermektedir.

## 19.2.1 Naive Bayes Modelleri Tarafından Kullanılan Temsil

Naif Bayes için gösterim olasılıklardır. Öğrenilmiş bir naif Bayes modeli için dosyaya bir olasılıklar listesi kaydedilir. Bu şunları içerir:

Sınıf Olasılıkları: Eğitim veri kümesindeki her bir sınıfın olasılıkları.

Koşullu Olasılıklar: Her bir sınıf değeri verildiğinde her bir girdi değerinin koşullu olasılıkları.

## 19.2.2 Verilerden Naive Bayes Modeli Öğrenme

Eğitim verilerinizden bir naive Bayes modeli öğrenmek hızlıdır. Eğitim hızlıdır çünkü yalnızca her sınıfın olasılığının ve farklı girdi ($x$) değerleri verilen her sınıfın olasılığının hesaplanması gerekir. Optimizasyon prosedürleri ile katsayıların uydurulmasına gerek yoktur.

### Sınıf Olasılıklarını Hesaplama

Sınıf olasılıkları basitçe her bir sınıfa ait örneklerin sıklığının toplam örnek sayısına bölünmesiyle elde edilir. Örneğin ikili bir sınıflandırmada bir örneğin sınıf 1'e ait olma olasılığı şu şekilde hesaplanır:

$$P\,(sınıf = 1) = \frac{count(class = 1)}{count(class = 0) + count(class = 1)} \tag{19.4}$$

En basit durumda, her sınıfta aynı sayıda örnek bulunan ikili bir sınıflandırma problemi için her sınıf 0,5 veya %50 olasılığa sahip olacaktır.

### Koşullu Olasılıkların Hesaplanması

Koşullu olasılıklar, belirli bir sınıf değeri için her bir öznitelik değerinin sıklığının o sınıf değerine sahip örneklerin sıklığına bölünmesiyle elde edilir. Örneğin, bir hava durumu özniteliği güneşli ve yağmurlu değerlerine sahipse ve sınıf özniteliği dışarı çık ve evde kal sınıf değerlerine sahipse, her bir sınıf değeri için her bir hava durumu değerinin koşullu olasılıkları şu şekilde hesaplanabilir:

$$\begin{aligned}
P\,(weather = sunny\,|class = \text{ go-out}) &= \frac{count(weather = sunny \ \wedge \ \text{class=go-out})}{count(\text{class=go-out})} \\
P\,(weather = sunny\,|class = \text{ stay-home}) &= \frac{count(weather = sunny \ \wedge \ class = \text{stay-home})}{count(class = \text{evde kal})} \\
P\,(weather = rainy\,|class = \text{ go-out}) &= \frac{count(weather = rainy \ \wedge \ class = \text{go-out})}{count(class = \text{go-out})} \\
P\,(weather = rainy\,|class = \text{ stay-home}) &= \frac{count(weather = rainy \ \wedge \ class = \text{stay-home})}{count(class = \text{evde kal})}
\end{aligned} \tag{19.5}$$

Burada ∧ bağlaç (ve) anlamına gelir.

## 19.2.3 Naive Bayes Modeli ile Tahminler Yapın

Naif bir Bayes modeli verildiğinde, Bayes teoremini kullanarak yeni veriler için tahminlerde bulunabilirsiniz.

$$MAP\,(h) = maks(P\,(d|h) \times \ P\,(h)) \tag{19.6}$$

Yukarıdaki örneğimizi kullanarak, hava durumu güneşli olan yeni bir örneğimiz olsaydı, hesaplayabilirdik:

$$\begin{aligned}
&\text{dışarı çık} = P\,(hava = güneşli|sınıf = \text{dışarı çık}) \times \ P\,(sınıf = \text{dışarı çık}) \\
&\text{evde kal} = P\,(hava\ durumu = güneşli|sınıf = \text{evde kal}) \times \ P\,(sınıf = \text{evde kal})
\end{aligned} \tag{19.7}$$

Hesaplanan en büyük değere sahip olan sınıfı seçebiliriz. Bu değerleri aşağıdaki gibi normalize ederek olasılıklara dönüştürebiliriz:

$$P(\text{dışarı } \textit{çıkma} | \textit{hava} \text{ durumu} = \textit{güneşli}) = \frac{\text{ÇIKIŞ}}{\text{dışarı çıkma} + \text{stay-home}}$$
$$P(\text{evde } \textit{kal} | \textit{hava} \text{ durumu} = \textit{güneşli}) = \frac{\text{EVDE KAL}}{\text{dışarı çık} + \text{stay-home}} \quad (19.8)$$

Daha fazla girdi değişkenimiz olsaydı yukarıdaki örneği genişletebilirdik. Örneğin, çalışan ve bozuk değerlerine sahip bir araba özelliğimiz olduğunu varsayalım. Bu olasılığı denklemle çarpabiliriz. Örneğin aşağıda, çalışmaya ayarlanmış araba girdi değişkeninin eklenmesiyle dışarı çıkma sınıfı etiketi için hesaplama yer almaktadır:

$$\begin{aligned} \text{dışarı çık} = & P(\textit{hava durumu} = \textit{güneşli} | \textit{sınıf} = \text{dışarı çık}) \times \\ & P(\textit{araba} = \textit{çalışma} | \textit{sınıf} = \text{dışarı çıkma}) \times \\ & P(\textit{sınıf} = \text{dışarı çıkma}) \end{aligned} \quad (19.9)$$

# 19.3 Gaussian Naive Bayes

Naive Bayes, çoğunlukla bir Gauss dağılımı varsayarak gerçek değerli özniteliklere genişletilebilir. Naive Bayes'in bu uzantısına Gaussian Naive Bayes adı verilir. Verilerin dağılımını tahmin etmek için başka fonksiyonlar kullanılabilir, ancak Gauss (veya Normal dağılım) ile çalışmak en kolay olanıdır, çünkü yalnızca eğitim verilerinizden ortalama ve standart sapmayı tahmin etmeniz gerekir.

## 19.3.1 Gauss Naive Bayes için Temsil

Yukarıda, frekans kullanarak her bir sınıf için girdi değerlerinin olasılıklarını hesapladık. Gerçek değerli girdilerle, dağılımı özetlemek için her sınıf için girdi değerlerinin ($x$) ortalamasını ve standart sapmasını hesaplayabiliriz. Bu, her sınıf için olasılıklara ek olarak, her sınıf için her girdi değişkeninin ortalama ve standart sapmalarını da saklamamız gerektiği anlamına gelir.

## 19.3.2 Verilerden Gauss Naive Bayes Modeli Öğrenme

Bu, her bir girdi değişkeninin ortalama ve standart sapma değerlerini hesaplamak kadar basittir ($x$) her bir sınıf değeri için.

$$ortalama(x) = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (19.10)$$

Burada $n$ örnek sayısıdır ve $x$ eğitim verilerinizdeki bir girdi değişkeninin değerleridir. Standart sapmayı aşağıdaki denklemi kullanarak hesaplayabiliriz:

$$StandartDeğişim(x) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - ortalama(x))^2} \quad (19.11)$$

Bu, $x$'in her bir değerinin $x$'in ortalama değerinden ortalama karesel farkının kareköküdür; burada $n$ örnek sayısıdır, $x_i$ $i$'inci örnek için $x$ değişkeninin belirli bir değeridir ve *ortalama*($x$) yukarıda açıklanmıştır.

### 19.3.3 Gauss Naive Bayes Modeli ile Tahminler Yapın

Yeni $x$ değerlerinin olasılıkları Gauss Olasılık Yoğunluk Fonksiyonu (PDF) kullanılarak hesaplanır. Tahminler yapılırken bu parametreler değişken için yeni bir girdi ile Gauss PDF'sine yerleştirilebilir ve karşılığında Gauss PDF'si o sınıf için yeni girdi değerinin olasılığının bir tahminini sağlayacaktır.

$$pdf(x, ortalama, sd) = \frac{1}{\sqrt{2 \times \pi \times sd}} \times e^{-\left(\frac{(x-ortalama)^2}{2 \times sd^2}\right)} \tag{19.12}$$

Burada *pdf* ($x$) Gauss PDF'si, *ortalama* ve *sd* yukarıda hesaplanan ortalama ve standart sapma, $\pi$ sayısal sabit PI, $e$ sayısal sabit Euler sayısı ve $x$ girdi değişkeni için girdi değeridir. Daha sonra gerçek değerli girdilerle tahminler yapmak için olasılıkları yukarıdaki denkleme ekleyebiliriz. Örneğin, yukarıdaki hesaplamalardan birini hava durumu ve araba için sayısal değerlerle uyarlamak:

$$\begin{aligned} \text{dışarı çık} = & P(pdf(hava\ durumu) | sınıf = \text{dışarı çık}) \times \\ & P(pdf(car) | class = \text{go-out}) \times \\ & P(sınıf = \text{dışarı çıkma}) \end{aligned} \tag{19.13}$$

## 19.4 Naive Bayes İçin Veri Hazırlama

Bu bölümde verilerinizi Naive Bayes için hazırlamaya yönelik bazı ipuçları verilmektedir.

**Kategorik Girdiler**: Naive Bayes ikili, kategorik veya nominal gibi etiket niteliklerini varsayar.

**Gauss Girdileri**: Girdi değişkenleri gerçek değerliyse Gauss dağılımı varsayılır. Bu durumda, verilerinizin tek değişkenli dağılımları Gauss veya Gauss'a yakınsa algoritma daha iyi performans gösterecektir. Bu, aykırı değerlerin (örneğin ortalamadan 3 veya 4 standart sapmadan daha fazla olan değerler) kaldırılmasını gerektirebilir.

**Sınıflandırma Problemleri**: Naive Bayes, ikili ve çok sınıflı sınıflandırma için uygun bir sınıflandırma algoritmasıdır.

**Log Olasılıkları**: Farklı sınıf değerlerinin olasılığının hesaplanması, çok sayıda küçük sayının birlikte çarpılmasını içerir. Bu durum sayısal hassasiyetin azalmasına yol açabilir. Bu nedenle, bu düşük akışı önlemek için olasılıkların log dönüşümünü kullanmak iyi bir uygulamadır.

**Çekirdek Fonksiyonları**: Sayısal girdi değerleri için Gauss dağılımı varsaymak yerine, çeşitli çekirdek yoğunluk fonksiyonları gibi daha karmaşık dağılımlar kullanılabilir.

**Olasılıkları Güncelleyin**: Yeni veriler elde edildiğinde, modelinizin olasılıklarını kolayca güncelleyebilirsiniz. Veriler sık sık değişiyorsa bu yararlı olabilir.

## 19.5 Özet

Bu bölümde sınıflandırma için Naive Bayes algoritmasını keşfettiniz. Hakkında bilgi edindiniz:

Bayes Teoremi ve pratikte nasıl hesaplanacağı.

Naive Bayes algoritması; temsil, tahmin yapma ve modelin öğrenilmesini içerir.

Naive Bayes'in gerçek değerli girdi verileri için uyarlanması Gaussian Naive Bayes olarak adlandırılır.

Naive Bayes için veri nasıl hazırlanır.

Artık sınıflandırma için Naive Bayes algoritması hakkında bilgi sahibisiniz. Bir sonraki bölümde kategorik değişkenler için Naive Bayes'i sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 20

# Naive Bayes Eğitimi

Naive Bayes, her bir girdi değişkeninin bağımsızlığı hakkında bazı güçlü varsayımlarda bulunan çok basit bir sınıflandırma algoritmasıdır. Bununla birlikte, çok sayıda problem alanında etkili olduğu gösterilmiştir. Bu bölümde kategorik veriler için Naive Bayes algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz.

Naive Bayes için kategorik verilerle nasıl çalışılır.

Bir Naive Bayes modeli için sınıf ve koşullu olasılıklar nasıl hazırlanır.

Tahmin yapmak için öğrenilmiş bir Naive Bayes modeli nasıl

kullanılır. Hadi başlayalım.

## 20.1 Öğretici Veri Seti

Veri kümesi iki kategorik girdi değişkenini ve iki çıktısı olan bir sınıf değişkenini tanımlamaktadır.

```
Hava Durumu  Araba      Sınıf
güneşli      çalışıyor  ÇIKIŞ
yağmurlu     kırık      ÇIKIŞ
güneşli      çalışıyor  ÇIKIŞ
güneşli      çalışıyor  ÇIKIŞ
güneşli      çalışıyor  ÇIKIŞ
yağmurlu     kırık      EVDE KAL
yağmurlu     kırık      EVDE KAL
güneşli      çalışıyor  EVDE KAL
güneşli      kırık      EVDE KAL
yağmurlu     kırık      EVDE KAL
```

Liste 20.1: Naive Bayes Öğretici Veri Seti.

Bunu sayılara dönüştürebiliriz. Her girdinin yalnızca iki değeri vardır ve çıktı sınıfı değişkeninin iki değeri vardır. Her bir değişkeni aşağıdaki gibi ikiliye dönüştürebiliriz:

Hava durumu: güneşli = 1, yağmurlu = 0

Araba: çalışıyor = 1, bozuk = 0

Sınıf: dışarı çıkmak = 1, evde kalmak = 0

Bu nedenle, veri kümesini şu şekilde yeniden ifade edebiliriz:

```
Hava       Araba   Sınıf
Durumu
1          1       1
0          0       1
1          1       1
1          1       1
1          1       1
0          0       0
0          0       0
1          1       0
1          0       0
0          0       0
```

Liste 20.2: Basitleştirilmiş Naive Bayes Öğretici Veri Seti.

Bu, takip ediyorsanız verilerin bir elektronik tabloda veya kodda çalışılmasını kolaylaştırabilir.

# 20.2 Bir Naive Bayes Modeli Öğrenin

Naif Bayes modeli için veri setinden hesaplanması gereken iki tür büyüklük vardır:

Sınıf Olasılıkları.

Koşullu Olasılıklar.

Sınıf olasılıkları ile başlayalım.

## 20.2.1 Sınıf Olasılıklarını Hesaplayın

Veri kümesi iki sınıflı bir problemdir ve veri kümesini oluşturduğumuz için her bir sınıfın olasılığını zaten biliyoruz. Bununla birlikte, 0 ve 1 sınıfları için sınıf olasılıklarını aşağıdaki gibi hesaplayabiliriz:

$$P\,(sınıf = 1) = \frac{count(class = 1)}{count(class = 0) + count(class = 1)}$$
$$P\,(sınıf = 0) = \frac{count(class = 0)}{count(class = 0) + count(class = 1)} \quad (20.1)$$

veya

$$P\,(sınıf = 1) = \frac{5}{5 + 5}$$
$$P\,(sınıf = 0) = \frac{5}{5 + 5} \quad (20.2)$$

Bu, herhangi bir veri örneğinin sınıf 0 veya sınıf 1'e ait olma olasılığının 0,5 olduğu anlamına gelir.

### 20.2.2 Koşullu Olasılıkları Hesaplayın

Koşullu olasılıklar, her bir sınıf değeri verildiğinde her bir girdi değerinin olasılığıdır. Veri kümesi için koşullu olasılıklar aşağıdaki gibi hesaplanabilir:

**Hava Durumu Girdi Değişkeni**

$$
\begin{aligned}
P(weather = sunny \mid class = \text{go-out}) &= \frac{count(weather = sunny \ \wedge \ class = \text{go-out})}{count(class = \text{go-out})} \\
P(weather = rainy \mid class = \text{go-out}) &= \frac{count(weather = rainy \ \wedge \ class = \text{go-out})}{count(class = \text{go-out})} \\
P(weather = sunny \mid class = \text{stay-home}) &= \frac{count(weather = sunny \ \wedge \ class = \text{stay-home})}{count(class = \text{evde kal})} \\
P(weather = rainy \mid class = \text{stay-home}) &= \frac{count(weather = rainy \ \wedge \ class = \text{stay-home})}{count(class = \text{evde kal})}
\end{aligned}
\tag{20.3}
$$

Unutmayın ki ∧ sembolü sadece bağlaç (AND) için bir kısaltmadır. Rakamları yerleştirirsek elde ederiz:

$$
\begin{aligned}
&P(hava = güneşli \mid sınıf = \text{dışarı çık}) = 0{,}8 \\
&P(hava = yağmurlu \mid sınıf = \text{dışarı çık}) = 0{,}2 \\
&P(hava = güneşli \mid sınıf = \text{evde kal}) = 0{,}4 \\
&P(hava = yağmurlu \mid sınıf = \text{evde kal}) = 0{,}6
\end{aligned}
\tag{20.4}
$$

**Araç Giriş Değişkeni**

$$
\begin{aligned}
P(car = \text{çalışma} \mid class = \text{dışarı çıkma}) &= \frac{count(araba = \text{çalışma} \ \wedge \ sınıf = \text{dışarı çıkma})}{count(class = \text{go-out})} \\
P(car = broken \mid class = \text{go-out}) &= \frac{count(car = broken \ \wedge \ class = \text{go-out})}{count(class = \text{go-out})} \\
P(car = \text{çalışma} \mid class = \text{evde kalma}) &= \frac{count(araba = \text{çalışma} \ \wedge \ sınıf = \text{evde kalma})}{count(class = \text{evde kal})} \\
P(car = broken \mid class = \text{stay-home}) &= \frac{count(car = broken \ \wedge \ class = \text{stay-home})}{count(class = \text{evde kal})}
\end{aligned}
\tag{20.5}
$$

Rakamları girdiğimizde:

$$
\begin{aligned}
&P(araba = \text{çalışma} \mid sınıf = \text{dışarı çıkma}) \\
&= 0{,}8 \; P(araba = bozuk \mid sınıf = \text{dışarı çıkma}) = 0{,}2 \\
&P(araba = \text{çalışma} \mid sınıf = \text{evde kalma}) = 0{,}2 \\
&P(araba = bozuk \mid sınıf = \text{evde kalma}) = 0{,}8
\end{aligned}
\tag{20.6}
$$

Artık Naive Bayes modelini kullanarak tahminler yapmak için ihtiyacımız olan her şeye sahibiz.

## 20.3 Naive Bayes ile Tahminler Yapın

Bir önceki bölümde tanımlanan ve açıklanan Bayes Teoremini kullanarak tahminlerde bulunabiliriz.

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)} \tag{20.7}$$

Aslında, yeni bir veri örneği için en olası sınıfı tahmin etmek için bir olasılığa ihtiyacımız yoktur. Yalnızca pay ve en büyük yanıtı veren sınıfa ihtiyacımız vardır, bu da tahmin edilen çıktı olacaktır.

$$MAP(h) = maks(P(d|h) \times P(h)) \tag{20.8}$$

Veri kümemizden ilk kaydı alalım ve hangi sınıfa ait olduğunu düşündüğümüzü tahmin etmek için öğrenilmiş modelimizi kullanalım. İlk örnek: *hava = güneşli, araba = çalışıyor*.

Modelimizin olasılıklarını her iki sınıf için de giriyoruz ve yanıtı hesaplıyoruz. Çıktı çıkışı için yanıtla başlıyoruz. Koşullu olasılıkları birlikte çarpıyoruz ve herhangi bir örneğin sınıfa ait olma olasılığı ile çarpıyoruz.

$$\begin{aligned} \text{dışarı çık} = & P(\textit{hava durumu} = \textit{güneşli} | \textit{sınıf} = \text{dışarı çık}) \times \\ & P(\textit{araba} = \textit{çalışma} | \textit{sınıf} = \text{dışarı çıkma}) \times \\ & P(\textit{sınıf} = \text{dışarı çıkma}) \end{aligned} \tag{20.9}$$

veya

$$\begin{aligned} \text{çıkış} &= 0{,}8 \times 0{,}8 \times 0{,}5 \\ \text{çıkış} &= 0{,}32 \end{aligned} \tag{20.10}$$

Aynı hesaplamayı evde kalma durumu için de yapabiliriz:

$$\begin{aligned} \text{evde kalma} = & P(\textit{hava} = \textit{güneşli} | \textit{sınıf} = \text{evde kalma}) \times \\ & P(\textit{araba} = \textit{çalışma} | \textit{sınıf} = \text{evde kalma}) \times \\ & P(\textit{sınıf} = \text{evde kalma}) \end{aligned} \tag{20.11}$$

veya

$$\begin{aligned} \text{evde kalma} &= 0{,}4 \times 0{,}2 \times 0{,}5 \\ \text{evde kalmak} &= 0{,}04 \end{aligned} \tag{20.12}$$

0,32'nin 0,04'ten büyük olduğunu görebiliyoruz, bu nedenle bu örnek için go-out tahmininde bulunuyoruz, ki bu doğru. Bu işlemi tüm veri kümesi için aşağıdaki gibi tekrarlayabiliriz:

```
                                                                    n
Hava       Araba      Sınıf      Dışarı çıkmak mı? Evde
güneşli    çalışıyor  çıkışı     kalmak mı? Tahmin
                                 0.32   0.04       ÇIKIŞ
yağmurlu    kırık     ÇIKIŞ      0.02   0.24       EVDE KAL
güneşli    çalışıyor  ÇIKIŞ      0.32   0.04       ÇIKIŞ
güneşli    çalışıyor  ÇIKIŞ      0.32   0.04       ÇIKIŞ
güneşli    çalışıyor  ÇIKIŞ      0.32   0.04       ÇIKIŞ
yağmurlu    kırık     EVDE KAL   0.02   0.24       EVDE KAL
yağmurlu    kırık     EVDE KAL   0.02   0.24       EVDE KAL
güneşli    çalışıyor  EVDE KAL   0.32   0.04       ÇIKIŞ
güneşli     kırık     EVDE KAL   0.08   0.16       EVDE KAL
yağmurlu    kırık     EVDE KAL   0.02   0.24       EVDE KAL
```

Liste 20.3: Veri Kümesi için Naive Bayes Tahminleri.

Tahminleri gerçek sınıf değerleriyle karşılaştırırsak, veri kümesinde çelişkili örnekler olduğu göz önüne alındığında mükemmel olan %80'lik bir doğruluk elde ederiz.

## 20.4 Özet

Bu bölümde Naive Bayes'in sıfırdan nasıl uygulanacağını keşfettiniz. Öğrendiniz:

Naive Bayes ile kategorik verilerle nasıl çalışılır.

Eğitim verilerinden sınıf olasılıkları nasıl hesaplanır?

Eğitim verilerinden koşullu olasılıklar nasıl hesaplanır.

Yeni veriler üzerinde tahminler yapmak için öğrenilmiş bir Naive Bayes modeli nasıl kullanılır?

Artık Naive Bayes'i kategorik veriler için sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde Naive Bayes'i gerçek değerli veriler için sıfırdan nasıl uygulayabileceğinizi keşfedeceksiniz.

# Bölüm 21

# Gauss Naive Bayes Eğitimi

Naive Bayes, yeni veriler üzerinde tahminler yapmak için eğitim verilerinizden hesaplanan olasılıkları kullanan basit bir modeldir. Temel Naive Bayes algoritması kategorik verileri varsayar. Gerçek değerli veriler için basit bir uzantı Gaussian Naive Bayes olarak adlandırılır. Bu bölümde Gaussian Naive Bayes'i sıfırdan nasıl uygulayacağınızı keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Gauss Olasılık Yoğunluk Fonksiyonu ve gerçek değerlerin olasılığının nasıl hesaplanacağı.

Eğitim verilerinizden bir Gauss Naive Bayes modeli için özellikler nasıl öğrenilir.

Yeni veriler üzerinde tahminler yapmak için öğrenilmiş bir Gauss Naive Bayes modeli

nasıl kullanılır. Hadi başlayalım.

## 21.1 Öğretici Veri Kümesi

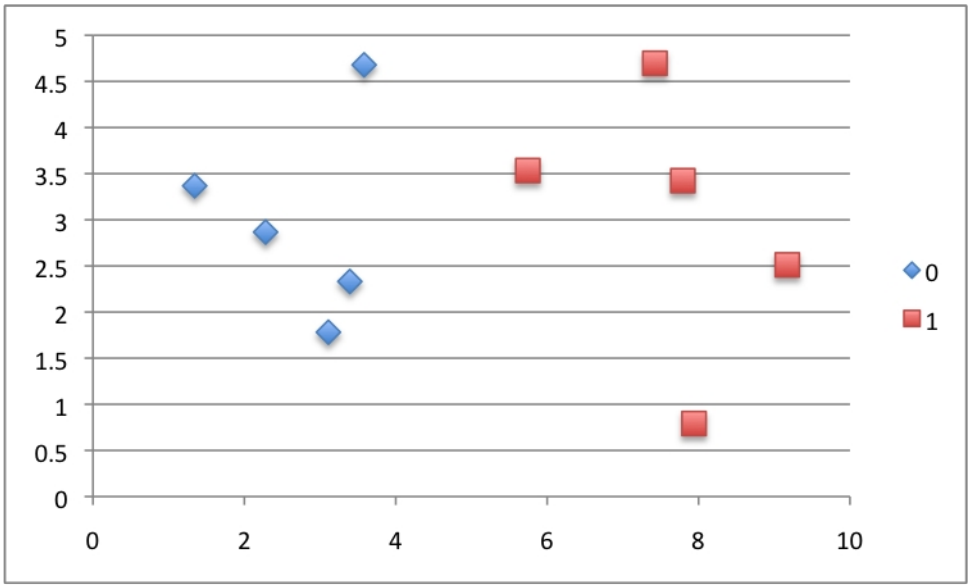
Amaçlarımız doğrultusunda basit bir veri kümesi oluşturulmuştur. *X1* ve *X2* olmak üzere iki girdi değişkeni ve *Y olmak* üzere bir çıktı değişkeninden oluşmaktadır. Girdi değişkenleri, Gauss Naive Bayes tarafından yapılan bir varsayım olan Gauss dağılımından alınmıştır. Sınıf değişkeninin 0 ve 1 olmak üzere iki değeri vardır, bu nedenle problem ikili bir sınıflandırma problemidir.

Sınıf 0'dan gelen veriler, X1 ve *X2* için standart sapması 1.0 olan bir Gauss dağılımından rastgele çekilmiştir. Sınıf 1'den gelen veriler, *X1* için ortalaması 7,5 ve *X2* için 2,5 olan bir Gauss dağılımından rastgele çekilmiştir. Bu, girdi verilerini bir dağılım grafiğine çizdiğimizde sınıfların güzel bir şekilde ayrıldığı anlamına gelir. Ham veri kümesi aşağıda listelenmiştir:

```
X1            X2            Y
3.393533211   2.331273381   0
3.110073483   1.781539638   0
1.343808831   3.368360954   0
3.582294042   4.67917911    0
2.280362439   2.866990263   0
7.423436942   4.696522875   1
5.745051997   3.533989803   1
9.172168622   2.511101045   1
7.792783481   3.424088941   1
7.939820817   0.791637231   1
```

Liste 21.1: Gaussian Naive Bayes Öğretici Veri Seti.

Aşağıdaki grafikte sınıfların ayrımını açıkça görebilirsiniz. Bu, bir Gauss Naive Bayes modelini uygularken ve test ederken verilerle çalışmayı nispeten kolaylaştıracaktır.




Şekil 21.1: Gaussian Naive Bayes Öğretici Veri Kümesi.

## 21.2 Gauss Olasılık Yoğunluğu Fonksiyonu

Gauss Olasılık Yoğunluk Fonksiyonu (PDF), geldiği dağılımın ortalaması ve standart sapması verilen bir değerin olasılığını hesaplayacaktır. Gauss PDF aşağıdaki gibi hesaplanır:

$$pdf(x, ortalama, sd) = \frac{1}{\sqrt{2 \times \pi \times sd}} \times e^{-\left(\frac{(x-ortalama)^2}{2 \times sd^2}\right)} \quad (21.1)$$

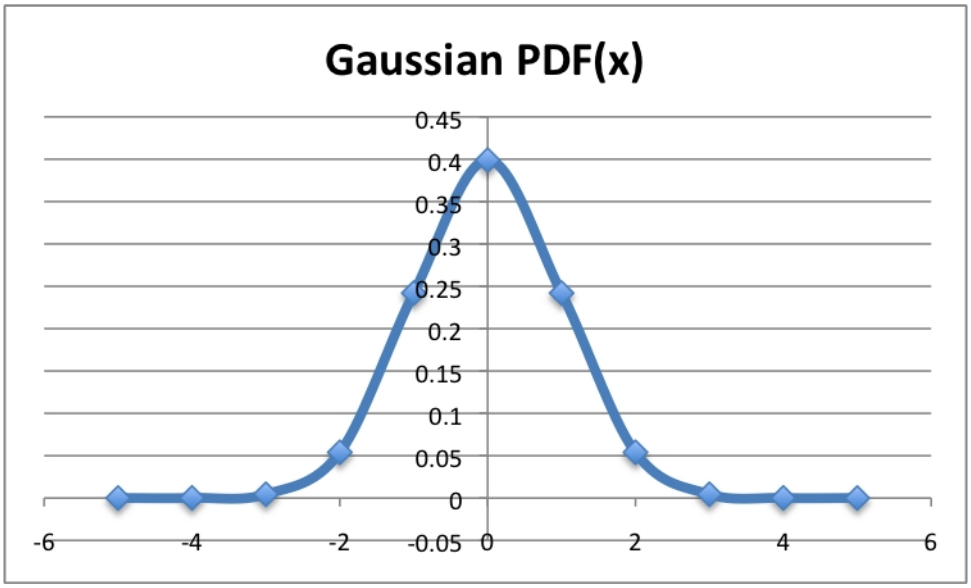
Burada *pdf* (*x*) Gauss PDF'si, *ortalama* ve *sd* yukarıda hesaplanan ortalama ve standart sapma, $\pi$ sayısal sabit PI, e Euler'in bir güce yükseltilmiş sayısı ve *x* giriş değişkeni için giriş değeridir. Bir örneğe bakalım. Ortalaması 0 ve standart sapması 1 olan bir popülasyondan alınan gerçek değerlere sahip olduğumuzu varsayalım. Gauss PDF'sini kullanarak değer aralığının olasılığını tahmin edebiliriz.

```
X       PDF(x)
-5      1.48672E-06
-4      0.00013383
-3      0.004431848
-2      0.053990967
-1      0.241970725
0       0.39894228
1       0.241970725
2       0.053990967
```

```
3       0.004431848
4       0.00013383
5       1.48672E-06
```

Liste 21.2: Gauss Olasılık Yoğunluk Fonksiyonu Testi.

Ortalamanın yaklaşık 0,4 (%40) ile en yüksek olasılığa sahip olduğunu görebilirsiniz. Ayrıca -5 ve +5 (ortalamadan 5 standart sapma) gibi ortalamadan uzak değerlerin çok düşük bir olasılığa sahip olduğunu görebilirsiniz. Aşağıda olasılık değerlerinin bir grafiği bulunmaktadır.




Şekil 21.2: Gauss Olasılık Yoğunluk Fonksiyonu.

Bu fonksiyon Naive Bayes için gerçekten kullanışlıdır. Girdi değişkenlerinin her birinin bir Gauss dağılımından çekildiğini varsayabiliriz. Eğitim verilerinden her bir girdi değişkeninin ortalamasını ve standart sapmasını hesaplayarak, her bir sınıf için her bir değerin olasılığını tahmin etmek için Gauss PDF'sini kullanabiliriz. Bunun koşullu olasılıkları hesaplamak için nasıl kullanıldığını bir sonraki bölümde göreceğiz.

## 21.3 Bir Gauss Naive Bayes Modeli Öğrenin

Naif Bayes modeli için eğitim verilerimizden özetlememiz gereken iki tür olasılık vardır:

Sınıf Olasılıkları.

Koşullu Olasılıklar.

### 21.3.1 Sınıf Olasılıkları

Veri kümesi iki sınıflı bir problemdir ve veri kümesini oluşturduğumuz için her bir sınıfın olasılığını zaten biliyoruz. Bununla birlikte, 0 ve 1 sınıfları için sınıf olasılıklarını hesaplayabiliriz.

1 aşağıdaki gibi:

$$P\,(Y=1) = \frac{count(Y=1)}{count(Y=0) + count(Y=1)}$$
$$P\,(Y=0) = \frac{count(Y=0)}{count(Y=0) + count(Y=1)} \tag{21.2}$$

veya

$$P\,(Y=1) = \frac{5}{5+5}$$
$$P\,(Y=0) = \frac{5}{5+5} \tag{21.3}$$

Bu, herhangi bir veri örneğinin sınıf 0 veya sınıf 1'e ait olma olasılığının 0,5 (%50) olduğu anlamına gelir.

### 21.3.2 Koşullu Olasılıklar

Koşullu olasılıklar, her bir sınıf değeri verildiğinde her bir girdi değerinin olasılıklarıdır. Eğitim verilerinden toplanması gereken koşullu olasılıklar aşağıdaki gibidir:

$$P\,(X1|Y=0)$$
$$P\,(X1|Y=1)$$
$$P\,(X2|Y=0)$$
$$P\,(X2|Y=1) \tag{21.4}$$

*X1* ve *X2* girdi değişkenleri gerçek değerlerdir. Bu nedenle onları bir Gauss dağılımından çekilmiş olarak modelleyeceğiz. Bu, yukarıda açıklanan Gauss PDF'sini kullanarak belirli bir değerin olasılığını tahmin etmemizi sağlayacaktır. Gauss PDF'si, olasılığın tahmin edildiği değere ek olarak iki parametre gerektirir: ortalama ve standart sapma. Bu nedenle, ihtiyaç duyduğumuz her koşullu olasılık grubu için ortalama ve standart sapmayı tahmin etmeliyiz. Bunları doğrudan veri setinden tahmin edebiliriz. Sonuçlar aşağıda özetlenmiştir.

```
       P(X1|Y=0)      P(X1|Y=1)      P(X2|Y=0)      P(X2|Y=1)
Ortalama               7.614652372    3.005468669    2.991467979
       2.742014401
Stdev 0.926568329      1.234432155    1.107329589    1.454193138
```

Liste 21.3: Sınıfa göre nüfus istatistiklerinin özeti.

Artık eğitim verileri ve hatta yeni bir veri kümesi için tahminler yapmak için yeterli bilgiye sahibiz.

## 21.4 Gaussian Naive Bayes ile Tahmin Yapın

Önceki bölümde tanıtılan ve açıklanan Bayes Teoremini kullanarak tahminler yapabiliriz. Yeni bir veri örneği için en olası sınıfı tahmin etmek için bir olasılığa ihtiyacımız yoktur. Sadece paya ihtiyacımız vardır ve en büyük yanıtı veren sınıf tahmin edilen yanıttır.

$$MAP\,(h) = maks(P\,(d|h) \times P\,(h)) \tag{21.5}$$

Veri kümemizden ilk kaydı alalım ve hangi sınıfa ait olduğunu düşündüğümüzü tahmin etmek için öğrenilen modelimizi kullanalım. Örnek: *X1* = 3.393533211, *X2* = 2.331273381, *Y* = 0. Modelimizin olasılıklarını her iki sınıf için de girebilir ve yanıtı hesaplayabiliriz. Çıktı sınıfı 0 için yanıtla başlayalım. Koşullu olasılıkları birlikte çarpıyoruz ve herhangi bir örneğin sınıfa ait olma olasılığı ile çarpıyoruz.

$$\begin{aligned} &\text{sınıf } 0 = P\,(pdf\,(X1)|sınıf = 0) \times P\,(pdf\,(X2)|sınıf = 0) \times P\,(sınıf = 0) \\ &\text{sınıf } 0 = 0.358838152 \times 0.272650889 \times 0.5 \\ &\text{sınıf } 0 = 0.048918771 \end{aligned} \tag{21.6}$$

Aynı hesaplamayı sınıf 1 için de yapabiliriz:

$$\begin{aligned} &\text{sınıf } 1 = P\,(pdf\,(X1)|sınıf = 1) \times P\,(pdf\,(X2)|sınıf = 1) \times P\,(\text{sınıf} = \\ &1)\ \text{sınıf } 1 = 4.10796E - 07 \times 0.173039018 \times 0.5 \\ &\text{sınıf } 1 = 3.55418E - 08 \end{aligned} \tag{21.7}$$

0,048918771 değerinin 3,55418E-08 değerinden büyük olduğunu görebiliyoruz, dolayısıyla bu örnek için sınıfı 0 olarak tahmin ediyoruz ki bu doğru. Bu işlemi veri kümesindeki tüm örnekler için tekrarlayarak sınıf 0 ve sınıf 1 için aşağıdaki çıktıları elde ederiz. En yüksek çıktıya sahip sınıfı seçerek eğitim veri kümesindeki tüm örnekler için doğru tahminler yaparız.

```
X1            X2            Çıkış Y=0      Çıkış Y=1     Tahmin
3.393533211   2.331273381   0.048918771    3.55418E-08   0
3.110073483   1.781539638   0.02920928     1.82065E-09   0
1.343808831   3.368360954   0.030910813    3.7117E-15    0
3.582294042   4.67917911    0.010283134    9.08728E-09   0
2.280362439   2.866990263   0.069951664    1.67539E-11   0
7.423436942   4.696522875   1.10289E-06    0.001993521   1
5.745051997   3.533989803   0.001361494    0.002264317   1
9.172168622   2.511101045   1.30731E-09    0.00547065    1
7.792783481   3.424088941   1.22227E-06    0.0355024     1
7.939820817   0.791637231   3.53132E-08    0.000245214   1
```

Liste 21.4: Gaussian Naive Bayes kullanarak tahminler.

Tahmin doğruluğu, sınıfların net bir şekilde ayrılması göz önüne alındığında beklendiği gibi %100'dür.

## 21.5 Özet

Bu bölümde Gaussian Naive Bayes sınıflandırıcısını sıfırdan nasıl uygulayacağınızı keşfettiniz. Şunları öğrendiniz:

Verilen herhangi bir gerçek değerin olasılığını tahmin etmek için Gauss Olasılık Yoğunluk Fonksiyonu.

Bir eğitim veri setinden Naive Bayes modelinin gerektirdiği olasılıklar nasıl tahmin edilir.

Tahmin yapmak için öğrenilen Naive Bayes modeli nasıl kullanılır?

Artık Gaussian Naive Bayes'i gerçek değerli veriler için sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde sınıflandırma ve regresyon için K-En Yakın Komşular algoritmasını keşfedeceksiniz.

# Bölüm 22

# K-En Yakın Komşular

Bu bölümde sınıflandırma ve regresyon için k-En Yakın Komşular (KNN) algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz.

KNN tarafından kullanılan model gösterimi.

KNN kullanılarak bir model nasıl öğrenilir (ipucu, değil).

KNN kullanarak tahminler nasıl yapılır

Farklı alanların KNN'ye nasıl atıfta bulunduğu da dahil olmak üzere KNN için birçok isim.

KNN'den en iyi şekilde yararlanmak için verilerinizi nasıl hazırlarsınız?

KNN algoritması hakkında daha fazla bilgi edinmek için

nereye bakmalısınız? Hadi başlayalım.

## 22.1 KNN Modeli Temsil

KNN için model temsili tüm eğitim veri kümesidir. Bu kadar basittir. KNN'nin tüm veri kümesini depolamaktan başka bir modeli yoktur, bu nedenle öğrenme gerekmez. Verimli uygulamalar, tahmin sırasında yeni modellerin aranmasını ve eşleştirilmesini verimli hale getirmek için k-d ağaçları gibi karmaşık veri yapıları kullanarak verileri depolayabilir. Eğitim veri setinin tamamı depolandığından, eğitim verilerinizin tutarlılığı hakkında dikkatlice düşünmek isteyebilirsiniz. Bu verileri düzenlemek, yeni veriler elde edildikçe sık sık güncellemek ve hatalı ve aykırı verileri kaldırmak iyi bir fikir olabilir.

## 22.2 KNN ile Tahmin Yapma

KNN doğrudan eğitim veri setini kullanarak tahminler yapar. Tahminler yeni bir veri noktası için tüm eğitim kümesinde en benzer K örnek (komşular) aranarak ve bu K örnek için çıktı değişkeni özetlenerek yapılır. Regresyon için bu ortalama çıktı değişkeni olabilir, sınıflandırmada ise bu mod (veya en yaygın) sınıf değeri olabilir.

Eğitim veri kümesindeki K örneklerinden hangilerinin yeni bir girdiye en çok benzediğini belirlemek için bir uzaklık ölçüsü kullanılır. Gerçek değerli girdi değişkenleri için en popüler uzaklık ölçütü Öklid uzaklığıdır. Öklid mesafesi, *i* tüm girdi öznitelikleri boyunca bir *a* noktası ile *b* noktası arasındaki karesel farkların toplamının karekökü olarak hesaplanır.

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1} (a_i - b_i)^2} \quad (22.1)$$

Diğer popüler mesafe ölçümleri şunlardır:

Hamming Mesafesi: İkili vektörler arasındaki mesafeyi hesaplar.

Manhattan Uzaklığı: Mutlak farklarının toplamını kullanarak gerçek vektörler arasındaki mesafeyi hesaplar. Şehir Blok Mesafesi olarak da adlandırılır.

Minkowski Uzaklığı: Öklid ve Manhattan uzaklığının genelleştirilmesi.

Tanimoto, Jaccard, Mahalanobis ve kosinüs mesafesi gibi kullanılabilecek başka birçok mesafe ölçüsü vardır. Verilerinizin özelliklerine göre en iyi mesafe ölçütünü seçebilirsiniz. Emin değilseniz, farklı mesafe ölçümlerini ve farklı K değerlerini birlikte deneyebilir ve hangi karışımın en doğru modellerle sonuçlandığını görebilirsiniz. Öklid, girdi değişkenleri benzer türdeyse (örneğin, ölçülen tüm genişlikler ve yükseklikler) kullanmak için iyi bir mesafe ölçüsüdür. Manhattan mesafesi, girdi değişkenleri tür olarak benzer değilse (yaş, cinsiyet, boy vb.) kullanmak için iyi bir ölçüdür.

K değeri algoritma ayarlaması ile bulunabilir. K için birçok farklı değer denemek (örneğin 1'den 21'e kadar olan değerler) ve probleminiz için en iyi sonucu veren değeri görmek iyi bir fikirdir. KNN'nin hesaplama karmaşıklığı eğitim veri kümesinin boyutuyla birlikte artar. Çok büyük eğitim setleri için KNN, K-en benzer örneklerin hesaplanacağı eğitim veri setinden bir örnek alınarak stokastik hale getirilebilir. KNN uzun zamandır kullanılmaktadır ve çok iyi çalışılmıştır. Bu nedenle, farklı disiplinler bunun için farklı isimlere sahiptir, örneğin:

**Örnek Tabanlı Öğrenme**: Tahmin yapmak için ham eğitim örnekleri kullanılır. Bu nedenle KNN genellikle örnek tabanlı öğrenme veya vaka tabanlı öğrenme (her bir eğitim örneğinin problem alanından bir vaka olduğu) olarak adlandırılır.

**Tembel Öğrenme**: Modelin öğrenilmesi gerekmez ve tüm çalışma bir tahmin istendiği anda gerçekleşir. Bu nedenle KNN genellikle tembel öğrenme algoritması olarak adlandırılır.

**Parametrik** değildir: KNN, çözülmekte olan problemin fonksiyonel formu hakkında hiçbir varsayımda bulunmaz. Bu nedenle KNN, parametrik olmayan bir makine öğrenimi algoritması olarak adlandırılır.

KNN regresyon ve sınıflandırma problemleri için kullanılabilir.

### 22.2.1 Regresyon için KNN

KNN regresyon problemleri için kullanıldığında, tahmin K-en benzer örneklerin ortalamasına veya medyanına dayanır.

### 22.2.2 Sınıflandırma için KNN

KNN sınıflandırma için kullanıldığında, çıktı K-en benzer örneklerden en yüksek frekansa sahip sınıf olarak hesaplanabilir. Özünde her örnek kendi sınıfı için oy verir ve en çok oy alan sınıf tahmin olarak alınır. Sınıf olasılıkları, yeni bir veri örneği için K en benzer örnek kümesindeki her bir sınıfa ait örneklerin normalleştirilmiş frekansı olarak hesaplanabilir. Örneğin, ikili bir sınıflandırma probleminde (sınıf 0 veya 1'dir):

$$p(sınıf = 0) = \frac{count(class = 0)}{count(class = 0) + count(class = 1)} \quad (22.2)$$

K kullanıyorsanız ve çift sayıda sınıfınız varsa (örneğin 2), eşitliği önlemek için tek sayılı bir K değeri seçmek iyi bir fikirdir. Tersi de geçerlidir, tek sayıda sınıfınız varsa K için çift sayı kullanın. K değerini 1'e genişleterek ve eğitim veri kümesindeki bir sonraki en benzer örneğin sınıfına bakarak bağlar tutarlı bir şekilde kırılabilir.

## 22.3 Boyutluluğun Laneti

KNN az sayıda girdi değişkeniyle (*p*) iyi çalışır, ancak girdi sayısı çok büyük olduğunda zorlanır. Her bir girdi değişkeni, p boyutlu bir girdi uzayının bir boyutu olarak düşünülebilir. Örneğin, *X1* ve *X2* olmak üzere iki girdi değişkeniniz varsa, girdi uzayı 2 boyutlu olacaktır. Boyut sayısı arttıkça girdi uzayının hacmi üstel bir oranda artar. Yüksek boyutlarda, benzer olabilecek noktalar çok büyük mesafelere sahip olabilir. Tüm noktalar birbirinden uzakta olacaktır ve basit 2 ve 3 boyutlu uzaylardaki mesafelere ilişkin sezgilerimiz bozulacaktır. Bu ilk başta sezgisel gelmeyebilir, ancak bu genel soruna *Boyutluluk Laneti* denir.

## 22.4 Verilerin KNN İçin Hazırlanması

**Verileri Yeniden Ölçeklendirin**: KNN, tüm veriler aynı ölçeğe sahipse çok daha iyi performans gösterir. Verilerinizi 0 ile 1 arasında normalleştirmek iyi bir fikirdir. Gauss dağılımına sahipse verilerinizi standartlaştırmak da iyi bir fikir olabilir.

**Eksik Verileri Ele Alın**: Eksik veriler, örnekler arasındaki mesafenin hesaplanamayacağı anlamına gelecektir. Bu örnekler hariç tutulabilir veya eksik değerler atfedilebilir.

**Düşük Boyutluluk**: KNN düşük boyutlu veriler için uygundur. Yüksek boyutlu verilerde (yüzlerce veya binlerce girdi değişkeni) deneyebilirsiniz ancak diğer teknikler kadar iyi performans göstermeyebileceğini unutmayın. KNN, girdi özellik uzayının boyutluluğunu azaltan özellik seçiminden faydalanabilir.

## 22.5 Özet

Bu bölümde KNN makine öğrenimi algoritmasını keşfettiniz. Şunları öğrendiniz:

KNN, temsili olarak kullandığı tüm eğitim veri kümesini saklar.

KNN herhangi bir model öğrenmez.

KNN, bir girdi örneği ile her bir eğitim örneği arasındaki benzerliği hesaplayarak tahminleri tam zamanında yapar.

Girdi verilerinizin yapısına uyması için aralarından seçim yapabileceğiniz birçok mesafe ölçüsü vardır.

KNN kullanırken normalleştirme gibi yöntemlerle verilerinizi yeniden ölçeklendirmenin iyi bir fikir olduğu.

Artık sınıflandırma ve regresyon için K-En Yakın Komşular algoritmasını biliyorsunuz. Bir sonraki bölümde sınıflandırma için KNN'yi sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 23

# K-En Yakın Komşular Eğitimi

K-En Yakın Komşular (KNN) algoritması çok basit ve çok etkilidir. Bu bölümde, adım adım sıfırdan nasıl uygulanacağını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Gerçek değerli vektörler arasındaki Öklid uzaklığı nasıl hesaplanır.

Yeni verilere yönelik tahminler yapmak için Öklid mesafesi ve eğitim veri kümesi nasıl kullanılır? Hadi başlayalım.

## 23.1 Öğretici Veri Seti

Problem ikili (iki sınıflı) bir sınıflandırma problemidir. Bu problem bu eğitim için oluşturulmuştur. Veri kümesi iki girdi değişkeni (*X1* ve *X1*) ve 0 ve 1 değerlerine sahip sınıf çıktı değişkeni içerir. Veri kümesi, her sınıfa ait 5 olmak üzere 10 kayıt içerir.

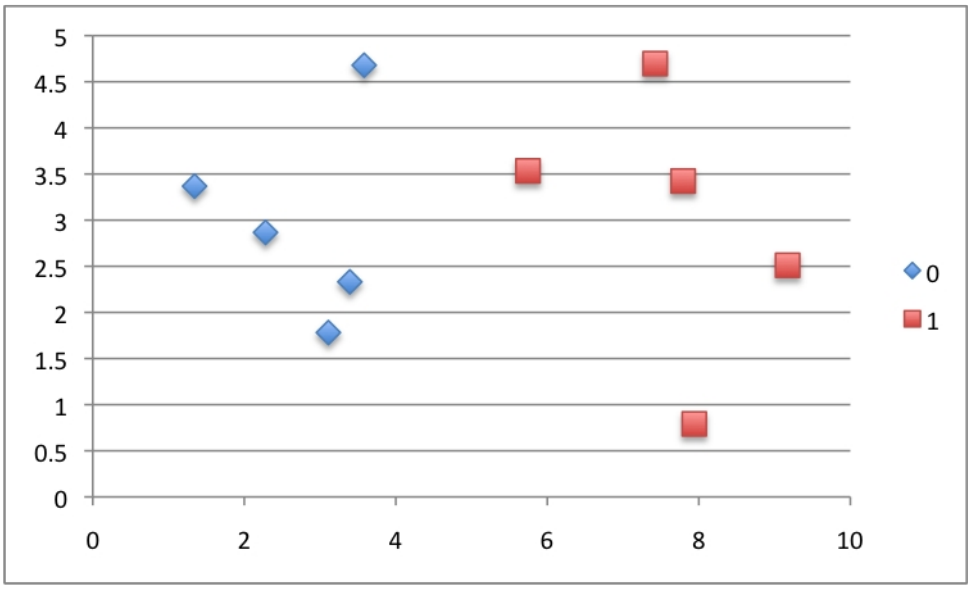
```
X1            X2            Y
3.393533211   2.331273381   0
3.110073483   1.781539638   0
1.343808831   3.368360954   0
3.582294042   4.67917911    0
2.280362439   2.866990263   0
7.423436942   4.696522875   1
5.745051997   3.533989803   1
9.172168622   2.511101045   1
7.792783481   3.424088941   1
7.939820817   0.791637231   1
```

Liste 23.1: KNN Öğretici Veri Seti.

Her sınıf için verilerin oldukça ayrı olduğunu görebilirsiniz. Bu, öğrenme algoritmasına odaklanabilmemiz için problemle çalışmayı kolaylaştırmak amacıyla yapılmıştır.

## 23.2 KNN ve Öklid Mesafesi

KNN, bir tahminde bulunurken eğitim veri kümesinden en benzer K örneği bulmak için bir mesafe ölçüsü kullanır. Seçilen mesafe ölçütü, veri kümesinin yapısına saygı göstermelidir.




Şekil 23.1: KNN Öğretici Veri Seti Dağılım grafiği.

Böylece uzaklık ölçüsüne göre birbirine yakın olan veri örnekleri de aynı sınıfa ait olur. Aynı birimlere veya ölçeğe sahip gerçek değerler için en yaygın uzaklık ölçüsü Öklid uzaklığıdır. Öklid mesafesi, *i* tüm girdi nitelikleri boyunca bir *a* noktası ile *b* noktası arasındaki karesel farkların toplamının karekökü olarak hesaplanır.

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1} (a_i - b_i)^2} \tag{23.1}$$

Bunu somutlaştırmak için, veri setimizden iki örnek için Öklid mesafesinin hesaplanması üzerinde çalışacağız.

| Örnek | X1 | X2 |
|---|---|---|
| 1 | 3.393533211 | 2.331273381 |
| 2 | 3.110073483 | 1.781539638 |

Liste 23.2: Öklid Uzaklığını Hesaplamak İçin İki Örnek.

İlk adım, her bir özellik için karesel farkı hesaplamaktır:

$$\begin{aligned} KareliFark1 &= (X1_1 - X1)_2^2 \\ KareliFark2 &= (X2_1 - X2)_2^2 \end{aligned} \tag{23.2}$$

veya

$$\begin{aligned} KareselFark1 &= (3.393533211 - 3.110073483)^2 \\ KareliFark2 &= (2.331273381 - 1.781539638)^2 \end{aligned} \tag{23.3}$$

veya

$$\begin{aligned} KareselFark1 &= 0.08034941698 \\ KareselFark2 &= 0.3022071889 \end{aligned} \tag{23.4}$$

Bu karesel farkların toplamını şu şekilde hesaplarız:

$$\begin{aligned} &ToplamKareFarkı = \text{ToplamKareFarkı1} + ToplamKareFarkı2 \\ &ToplamKareFarkı = 0.080349417 + 0.302207188 \\ &SumSquaredDifference = 0.382556606 \end{aligned} \tag{23.5}$$

Son olarak, toplamın karekökünü almamız gerekir. Bu, veri örnekleri (gerçek vektörler) arasındaki farkın birimlerini kareli birimlerden orijinal birimlerine dönüştürecektir.

$$\begin{aligned} Mesafe &= \sqrt{SumSquaredDifference} \\ Mesafe &= 0.618511605 \end{aligned} \tag{23.6}$$

Bu son adım performans nedenleriyle atlanabilir. Muhtemelen gerçek birimlerdeki mesafeye ihtiyacınız yoktur ve karekök işlevi diğer işlemlere kıyasla nispeten pahalıdır ve sınıflandırılacak yeni veri örneği başına birçok kez gerçekleştirilecektir. Artık Öklid mesafesi ölçüsünün nasıl hesaplandığını bildiğinize göre, yeni veriler için tahminler yapmak üzere veri kümesiyle birlikte kullanabiliriz.

## 23.3 KNN ile Tahmin Yapma

Tahmin yapmak istediğimiz yeni bir veri örneği verildiğinde, yeni veri örneğine en küçük mesafeye sahip K örneği bu tahmine katkıda bulunmak üzere seçilir. Sınıflandırma için bu, K üyelerinin her birinin yeni veri örneğinin hangi sınıfa ait olduğunu oylamasına izin vermeyi içerir. Bunu somutlaştırmak için, eğitim veri kümesini model olarak kullanarak yeni bir veri örneği için tahmin yapmaya çalışacağız. Yeni veri örneği aşağıda listelenmiştir. Veri kümesini biz oluşturduğumuz için hile yapıyoruz, örneğin hangi sınıfa tahsis edilmesi gerektiğini biliyoruz. Örnek: $X1 = 8.093607318$, $X2 = 3.365731514$, $Y = 1$.

İlk adım, yeni girdi örneği ile eğitim veri kümesindeki tüm örnekler arasındaki Öklid mesafesini hesaplamaktır. Aşağıdaki tabloda her bir eğitim örneği ile yeni veri arasındaki mesafe listelenmektedir.

| Hayır. | X1 | X2 | Y | (X1-X1)^2 | (X2-X2)^2 | Toplam | Mesafe |
|---|---|---|---|---|---|---|---|
| 1 | 3.393533211 | 2.331273381 | 0 | 22.09069661 | 1.070103629 | 23.16080024 | 4.812566908 |
| 2 | 3.110073483 | 1.781539638 | 0 | 24.83560948 | 2.5096639 | 27.34527338 | 5.229270827 |
| 3 | 1.343808831 | 3.368360954 | 0 | 45.55977962 | 6.91395E-06 | 45.55978653 | 6.749798999 |
| 4 | 3.582294042 | 4.679179110 | 0 | 20.35194747 | 1.725144587 | 22.07709206 | 4.698626614 |
| 5 | 2.280362439 | 2.866990263 | 0 | 33.79381602 | 0.248742835 | 34.04255886 | 5.834600146 |
| 6 | 7.423436942 | 4.696522875 | 1 | 0.449128333 | 1.771005647 | 2.220133979 | 1.490011402 |
| 7 | 5.745051997 | 3.533989803 | 1 | 5.515712096 | 0.028310852 | 5.544022948 | 2.354574897 |
| 8 | 9.172168622 | 2.511101045 | 1 | 1.163294486 | 0.730393239 | 1.893687725 | 1.376113268 |
| 9 | 7.792783481 | 3.424088941 | 1 | 0.090494981 | 0.003405589 | 0.09390057 | 0.306431999 |
| 10 | 7.939820817 | 0.791637231 | 1 | 0.023650288 | 6.625961377 | 6.649611665 | 2.578684096 |

Liste 23.3: Eğitim Verileri için Yeni Örneğe Öklid Uzaklıkları.

K değerini 3 olarak ayarlayacağız ve veri örneğine en çok benzeyen 3 komşuyu seçeceğiz. $K = 3$ değeri küçüktür ve bu örnekte kullanımı kolaydır, aynı zamanda tek sayıdır, yani komşular çıktı sınıfı üzerinde oylama yaptığında eşitlik olması mümkün değildir. Yeni veri örneğinin $K = 3$ en benzer komşuları şunlardır:

```
Hayır   Mesafe          Y
.
9       0.306431999     1
8       1.376113268     1
6       1.490011402     1
```

Liste 23.4: K=3 Yeni Örneğe En Benzer Komşular.

Tahmin yapmak, komşulardaki çoğunluk sınıfını seçmek kadar kolaydır. Sınıf değerleri için 0 ve 1 kullandığımızdan, en sık rastlanan değeri döndürmek için bir elektronik tablodaki MODE() istatistiksel işlevini kullanabiliriz.

$$tahmin = mod(sınıf(i)) \tag{23.7}$$

Bu durumda 3 komşunun da sınıfı 1'dir, dolayısıyla bu örnek için tahmin 1'dir, ki bu doğru.

## 23.4 Özet

Bu bölümde, ikili bir sınıflandırma probleminde tahminler yapmak için K-En Yakın Komşuları nasıl kullanabileceğinizi keşfettiniz. Şunları öğrendiniz:

Öklid uzaklık ölçüsü ve adım adım nasıl hesaplanacağı.

Yeni bir veri örneğinin en yakın komşularını bulmak için Öklid mesafesi nasıl kullanılır?

K-en yakın komşulardan bir tahmin nasıl yapılır.

Artık sınıflandırma için sıfırdan K-En Yakın Komşuları nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde sınıflandırma için KNN'nin bir uzantısı olan Vektör Niceleme Öğrenimini keşfedeceksiniz.

# Bölüm 24

# Vektör Niceleme Öğrenimi

K-En Yakın Komşular'ın bir dezavantajı, tüm eğitim veri setinize bağlı kalmanız gerektiğidir. Öğrenme Vektörü Niceleme algoritması (ya da kısaca LVQ), kaç eğitim örneğine bağlı kalacağınızı seçmenize olanak tanıyan ve bu örneklerin tam olarak neye benzemesi gerektiğini öğrenen bir yapay sinir ağı algoritmasıdır. Bu bölümde Öğrenme Vektörü Niceleme algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

LVQ algoritması tarafından kullanılan ve aslında bir dosyaya kaydettiğiniz temsil.

Öğrenilmiş bir LVQ modeli ile tahminler yapmak için kullanabileceğiniz prosedür.

Eğitim verilerinden bir LVQ modeli nasıl öğrenilir.

LVQ algoritmasından en iyi performansı elde etmek için kullanılacak veri hazırlığı.

LVQ hakkında daha fazla bilgi için nereye

bakmalısınız. Hadi başlayalım.

## 24.1 LVQ Modeli Temsil

LVQ için gösterim, kod kitabı vektörlerinin bir koleksiyonudur. LVQ bir sınıflandırma algoritması olarak geliştirilmiştir ve en iyi şekilde anlaşılmaktadır. Hem ikili (iki sınıflı) hem de çok sınıflı sınıflandırma problemlerini destekler. Kod kitabı vektörü, eğitim verilerinizle aynı girdi ve çıktı özelliklerine sahip sayıların bir listesidir. Örneğin, probleminiz 0 ve 1 sınıfları ve genişlik, uzunluk yükseklik girdileri olan ikili bir sınıflandırma ise, bir kod kitabı vektörü dört özniteliğin tümünden oluşacaktır: genişlik, uzunluk, yükseklik ve sınıf.

Model temsili, eğitim verilerinden öğrenilen sabit bir kod kitabı vektörleri havuzudur. Eğitim örneklerine benzerler, ancak her bir özniteliğin değerleri öğrenme prosedürüne göre uyarlanmıştır. Sinir ağları dilinde, her bir kod kitabı vektörüne nöron, bir kod kitabı vektöründeki her bir özniteliğe ağırlık ve kod kitabı vektörleri koleksiyonuna da ağ adı verilebilir.

## 24.2 LVQ Modeli ile Tahminlerde Bulunmak

Tahminler, K-En Yakın Komşular ile aynı şekilde LVQ kod kitabı vektörleri kullanılarak yapılır. Yeni bir örnek için tahminler, en benzer K örnek için tüm kod kitabı vektörlerinde arama yapılarak ve bu K örnek için çıktı değişkeni özetlenerek yapılır. Sınıflandırma için bu, mod (veya en yaygın) sınıf değeridir. Tipik olarak tahminler $K = 1$ ile yapılır ve eşleşen kod kitabı vektörüne En İyi Eşleşen Birim (BMU) denir.

Eğitim veri kümesindeki K örneklerinden hangilerinin yeni bir girdiye en çok benzediğini belirlemek için bir uzaklık ölçüsü kullanılır. Gerçek değerli girdi değişkenleri için en popüler uzaklık ölçütü Öklid uzaklığıdır. Öklid mesafesi, *i* tüm girdi öznitelikleri boyunca bir *a* noktası ile *b* noktası arasındaki karesel farkların toplamının karekökü olarak hesaplanır.

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1} (a_i - b_i)^2} \tag{24.1}$$

## 24.3 Verilerinden Bir LVQ Modeli Öğrenme

LVQ algoritması kod kitabı vektörlerini eğitim verilerinden öğrenir. Kullanılacak kod kitabı vektörlerinin sayısını seçmelisiniz, örneğin 20 veya 40. Eğitim veri kümeniz üzerinde farklı yapılandırmaları test ederek kullanılacak en iyi kod kitabı vektör sayısını bulabilirsiniz. Öğrenme algoritması rastgele kod kitabı vektörlerinden oluşan bir havuzla başlar. Bunlar eğitim verilerinden rastgele seçilmiş örnekler veya eğitim verileriyle aynı ölçekte rastgele oluşturulmuş vektörler olabilir. Kod kitabı vektörleri eğitim verileriyle aynı sayıda girdi niteliğine sahiptir. Ayrıca bir çıktı sınıfı değişkenine sahiptirler.

Eğitim veri kümesindeki örnekler teker teker işlenir. Belirli bir eğitim örneği için, havuzdan en benzer kod kitabı vektörü seçilir. Kod kitabı vektörü eğitim örneği ile aynı çıktıya sahipse, kod kitabı vektörü eğitim örneğine yaklaştırılır. Eşleşmezse, daha uzağa taşınır. Vektörün taşındığı miktar, öğrenme oranı (*alfa*) adı verilen bir algoritma parametresi tarafından kontrol edilir. Örneğin, bir kod kitabı vektörünün girdi değişkeni ($x$), sınıflar aşağıdaki gibi eşleşirse öğrenme oranındaki (*alfa*) miktar kadar eğitim girdi değerine ($t$) yaklaştırılır:

$$x = x + alfa \times (t - x) \tag{24.2}$$

Bir kod kitabı değişkeninin girdi değişkenlerini bir eğitim örneğinden uzaklaştırmanın tersi durumu şu şekilde hesaplanır:

$$x = x - alfa \times (t - x) \tag{24.3}$$

Bu işlem her bir girdi değişkeni için tekrarlanacaktır. Her eğitim örneği için değişiklik yapılmak üzere bir kod kitabı vektörü seçildiğinden, algoritma kazanan hepsini alır algoritması veya bir tür rekabetçi öğrenme olarak adlandırılır. Bu işlem eğitim veri kümesindeki her örnek için tekrarlanır. Eğitim veri kümesinin bir iterasyonuna epok denir. İşlem, 200, 2000 veya 20.000 gibi seçmeniz gereken (MaxEpoch) bir epok sayısı için tamamlanır.

Ayrıca bir başlangıç öğrenme oranı da seçmelisiniz (örneğin *alpha* = 0.3). Öğrenme oranı, kod kitabı vektörlerinde en fazla değişikliği yapan epok 1'de belirlediğiniz büyük değerden başlayarak ve son epokta sıfıra yakın küçük bir değerle bitirerek epokla birlikte azalır.

epoch kod kitabı vektörlerinde çok küçük değişiklikler yapar. Her bir epok için öğrenme oranı şu şekilde hesaplanır:

$$LearningRate = alpha \times \left(1 - \frac{Epoch}{MaxEpoch}\right) \quad (24.4)$$

Burada *LearningRate* mevcut dönem için öğrenme oranıdır (0 ila *MaxEpoch-1*), alpha eğitim çalışmasının başlangıcında algoritmaya belirtilen öğrenme oranıdır ve *MaxEpoch* çalışmanın başlangıcında da belirtilen algoritmanın çalıştırılacağı toplam dönem sayısıdır. Öğrenme sürecinin sezgisi, kod kitabı vektörleri havuzunun, eğitim veri kümesinin sınıfların ayrılmasını en iyi karakterize eden noktalara sıkıştırılmasıdır.

## 24.4 Verilerin LVQ İçin Hazırlanması

Genel olarak, LVQ için verileri K-En Yakın Komşular için hazırladığınız şekilde hazırlamak iyi bir fikirdir.

**Sınıflandırma**: LVQ, hem ikili (iki sınıflı) hem de çok sınıflı sınıflandırma algoritmaları için çalışan bir sınıflandırma algoritmasıdır. Bu teknik regresyon için uyarlanmıştır.

**Çoklu Geçişler**: LVQ ile iyi bir teknik, kod kitabı vektörleri üzerinde eğitim veri kümesinin birden fazla geçişini gerçekleştirmeyi içerir (örneğin, birden fazla öğrenme çalışması). İlki, kod kitabı vektörleri havuzunu yerleştirmek için daha yüksek bir öğrenme oranıyla ve ikincisi vektörlere ince ayar yapmak için küçük bir öğrenme oranıyla çalıştırılır.

**Çoklu En İyi Eşleşmeler:** LVQ'nun uzantıları, öğrenme sırasında değiştirmek için aynı sınıftan bir tane ve sırasıyla bir eğitim örneğine doğru ve ondan uzağa çekilen farklı bir sınıftan bir tane gibi birden fazla en iyi eşleşen birim seçer. Diğer uzantılar her bir kod kitabı vektörü için özel bir öğrenme oranı kullanır. Bu uzantılar öğrenme sürecini iyileştirebilir.

**Girdileri Normalleştirin**: Geleneksel olarak, girdiler 0 ile 1 arasındaki değerlere normalize edilir (yeniden ölçeklendirilir). Bu, bir niteliğin mesafe ölçüsüne hakim olmasını önlemek içindir. Girdi verileri normalize edilirse, kod kitabı vektörleri için başlangıç değerleri 0 ile 1 arasında rastgele değerler olarak seçilebilir.

**Özellik Seçimi**: Girdi değişkenlerinin boyutsallığını azaltabilen özellik seçimi, yöntemin doğruluğunu artırabilir. LVQ, tahmin yaparken K-En Yakın Komşular ile aynı boyutsallık lanetinden muzdariptir.

## 24.5 Özet

Bu bölümde LVQ algoritmasını keşfettiniz. Öğrendiniz:

LVQ için temsil, eğitim veri kümesinden daha küçük olan küçük bir kod kitabı vektör havuzudur.

Kod kitabı vektörleri, K-En Yakın Komşular ile aynı tekniği kullanarak tahminler yapmak için kullanılır.

Kod kitabı vektörleri, iyi eşleştiklerinde yakınlaştırılarak ve kötü eşleştiklerinde uzaklaştırılarak eğitim veri kümesinden öğrenilir.

Kod kitabı vektörleri, sınıfları en iyi şekilde ayırmak için eğitim verilerinin sıkıştırılmasıdır.

Veri hazırlama geleneksel olarak giriş değerlerinin 0 ile 1 aralığına normalleştirilmesini içerir.

Artık sınıflandırma için Öğrenme Vektörü Niceleme algoritması hakkında bilgi sahibisiniz. Bir sonraki bölümde LVQ'yu sıfırdan nasıl uygulayabileceğinizi keşfedeceksiniz.

# Bölüm 25

# Vektör Niceleme Öğreticisini Öğrenme

Öğrenme Vektörü Niceleme (LVQ) algoritması K-En Yakın Komşu algoritmasına çok benzer, ancak öğrenmeyi içerir. Bu bölümde LVQ algoritmasını sıfırdan nasıl uygulayacağınızı keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Bir LVQ modeli nasıl başlatılır.

Bir eğitim örneği için En İyi Eşleşen Birim Nasıl Güncellenir?

Bir ve birden fazla dönem için LVQ modeli nasıl güncellenir.

Tahmin yapmak için öğrenilmiş bir LVQ modeli nasıl

kullanılır? Hadi başlayalım.

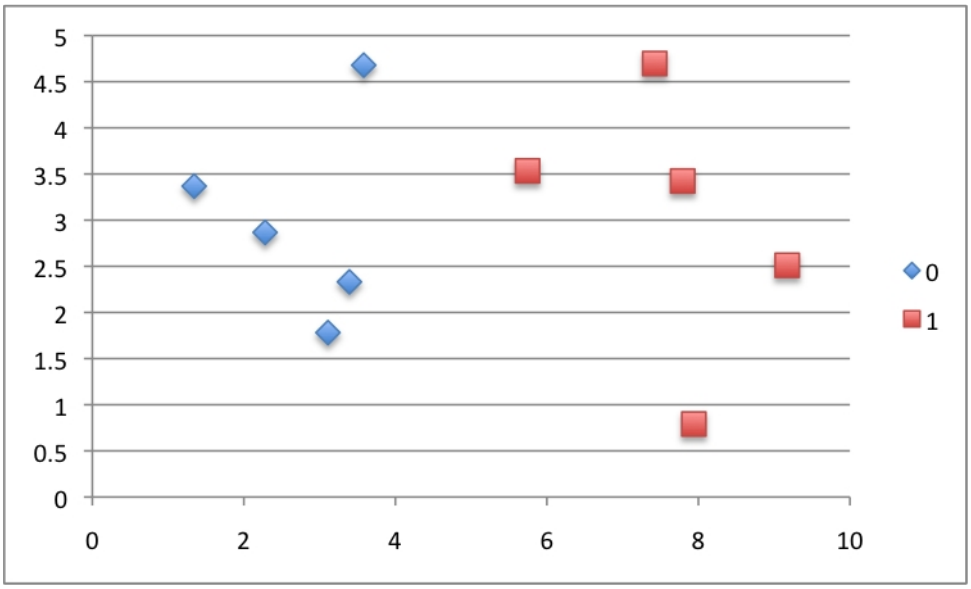
## 25.1 Öğretici Veri Seti

Problem ikili (iki sınıflı) bir sınıflandırma problemidir. Problem bu eğitim için oluşturulmuştur. Veri kümesi iki girdi değişkeni (*X1* ve *X2*) ve 0 ve 1 değerlerine sahip sınıf çıktı değişkeni içerir. Veri kümesi, her sınıfa ait 5 olmak üzere 10 kayıt içerir.

```
X1            X2            Y
3.393533211   2.331273381   0
3.110073483   1.781539638   0
1.343808831   3.368360954   0
3.582294042   4.67917911    0
2.280362439   2.866990263   0
7.423436942   4.696522875   1
5.745051997   3.533989803   1
9.172168622   2.511101045   1
7.792783481   3.424088941   1
7.939820817   0.791637231   1
```

Liste 25.1: LVQ Öğretici Veri Seti.

Her sınıf için verilerin oldukça ayrı olduğunu görebilirsiniz. Bu, öğrenme algoritmasına odaklanabilmemiz için problemle çalışmayı kolaylaştırmak amacıyla yapılmıştır.




Şekil 25.1: LVQ Öğretici Veri Seti Dağılım grafiği.

## 25.2 LVQ Modelini öğrenin

LVQ, eğitim verilerinden bir kod kitabı vektörleri popülasyonu öğrenir. Bu bölüm 3 kısma ayrılmıştır:

1. İlk Kod Kitabı Vektörleri.
2. Bir Desen için Kod Kitabı Vektörlerini Güncelleyin.
3. Bir Dönem İçin Güncelleme.

### 25.2.1 İlk Kod Kitabı Vektörleri

Kod kitabı vektörlerinin sayısı genellikle problemin boyutuna ve karmaşıklığına bağlıdır. Basit bir problemle çalıştığımız için ve gösterim amaçlı olarak her sınıftan iki tane olmak üzere az sayıda 4 kod kitabı vektörü seçeceğiz. Vektörlerin değerleri rastgele seçilebilir veya giriş verilerinden seçilebilir. Biz ikincisini kullanacağız. Aşağıdaki tabloda seçilen kod kitabı vektörleri listelenmektedir:

```
X1            X2            Y
3.582294042   0.791637231   0
7.792783481   2.331273381   0
7.939820817   2.866990263   1
3.393533211   4.67917911    1
```

Liste 25.2: İlk LVQ kod kitabı Vektörleri.

### 25.2.2 Bir Desen için Kod Kitabı Vektörlerini Güncelleme

Kod kitabı vektörlerinin sayısını seçmenin yanı sıra, bir başlangıç öğrenme oranı da belirtilmelidir. İyi bir varsayılan değer 0,3'tür, ancak değerler genellikle 0,1 ile 0,5 arasındadır. Öğrenme oranı

oranı kod kitabı vektörlerini güncellemek için kullanılır. Bu bölümde, bir eğitim örüntüsü için kod kitabı vektörlerini güncellemek için kullanılan kurala bakacağız. İlk eğitim örüntüsünü ele alalım: X1 = 3.393533211, X2 = 2.331273381, *Y* = 0. Bir örüntü için güncelleme kuralı aşağıdaki gibidir:

1. Eğitim örüntüsünden her bir kod kitabı vektörüne olan uzaklığı hesaplayın.
2. En İyi Eşleşen Birim (BMU) olarak adlandırılan en benzer kod kitabı vektörünü seçin.
3. En iyi eşleşen birimi, aynı sınıfa sahipse eğitim modeline daha yakın, aksi takdirde daha uzak olacak şekilde güncelleyin.

### Mesafe Hesaplama

Bir eğitim örneği ile kod kitabı vektörü arasındaki mesafeyi Öklid mesafesini kullanarak hesaplayabiliriz. Bu, tüm girdi nitelikleri aynı ölçeğe sahip olduğunda en yaygın mesafe ölçüsüdür, ki bu durumda öyledir. Öklid mesafesi, *i* tüm girdi öznitelikleri boyunca bir *a* noktası ile *b* noktası arasındaki karesel farkların toplamının karekökü olarak hesaplanır.

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1} (a_i - b_i)^2} \qquad (25.1)$$

Her bir kod kitabı vektörü ile eğitim örneği arasındaki Öklid mesafesini hesaplayalım. Sonuçlar aşağıda listelenmiştir.

| (X1-X1)^2 | (X2-X2)^2 | Toplam | Mesafe | BMU MU? |
|---|---|---|---|---|
| 0.035630651 | 2.370479474 | 2.406110125 | 1.551164119 | EVET |
| 19.35340294 | 0 | 19.35340294 | 4.39925027 | |
| 20.668731 | 0.286992578 | 20.95572357 | 4.577742192 | |
| 0 | 5.512661312 | 5.512661312 | 2.347905729 | |

Liste 25.3: Kod Kitabı Vektörlerinin Örnekten Öklid Uzaklıkları.

Mesafelerden, 1 numaralı kod kitabı vektörünün en küçük mesafeye sahip olduğunu ve bu nedenle BMU olduğunu görebiliriz.

### En İyi Eşleşen Birimi Güncelleyin

En iyi eşleşen birimin, aynı sınıfsa eğitim örneğine yaklaştırılması veya sınıflar farklıysa daha uzağa taşınması gerekir. BMU için sınıf 0'dır ve bu eğitim örneğinin sınıfıyla eşleşir 0. Bu nedenle, öğrenme oranıyla sınırlı olmak üzere, öznitelikleri eğitim örneğine daha yakın olacak şekilde güncellememiz gerekir. Öğrenme oranı, tek bir güncelleme için kod kitabı vektörlerinde ne kadar değişiklik yapılabileceğini kontrol eder. Bu basit problemde hızlı öğrenmeyi göstermek için normalden çok daha büyük olan 0,7'lik bir başlangıç öğrenme oranı kullanalım. Bu nedenle tek bir nitelik için güncelleme prosedürü şöyledir:

$$bmu_j = bmu_j + alfa \times (eğitim_j - bmu)_j \qquad (25.2)$$

Burada *j* güncellenmekte olan kod kitabı vektöründeki özniteliktir (örneğin *X1*), *alfa* öğrenme oranıdır ve $eğitim_j$ eğitim örneğindeki aynı özniteliktir (örneğin *X1*). Eğer

BMU farklı bir sınıfa sahip olsaydı, BMU'yu eğitim örneğinden daha uzağa itmek için negatif bir işaret kullanmamız dışında güncelleme neredeyse aynı olurdu. Örneğin:

$$bmu_j = bmu_j - alfa \times (eğitim_j - bmu)_j \tag{25.3}$$

Öğrenme kuralını uygulayalım ve BMU'nun niteliklerini güncelleyelim.

$$\begin{aligned} &X1 = 3.582294042 + 0.7 \times (3.393533211 - 3.582294042) \\ &X2 = 0.791637231 + 0.7 \times (2.331273381 - 0.791637231) \end{aligned} \tag{25.4}$$

veya

$$\begin{aligned} X1 &= 3.45016146 \\ X2 &= 1.869382536 \end{aligned} \tag{25.5}$$

Bir eğitim örneği için güncellemeyi henüz tamamladık.

### 25.2.3 Bir Dönem İçin Güncelleme

Bir epok, tüm eğitim veri setinden (10 örneğin tamamı) bir geçiştir. Bu, her eğitim örneği için yukarıdaki prosedürün uygulanmasını, BMU'nun bulunmasını ve güncellenmesini içerir. Çalıştırmaya başlamadan önce modeli eğitmek için gerçekleştirilecek epok sayısını seçmeliyiz. Bu sayı, problemin zorluğuna bağlı olarak yüzlerce, binlerce ve hatta on binlerce epok olabilir. Her epokta, öğrenme oranı başlangıç değerinden azaltılır. Bu, çalışmanın başlangıcında modelin çok fazla öğrenme yaptığı ve çalışmanın sonuna doğru zaten öğrenilmiş olan kod kitabı vektörlerinde yalnızca çok küçük ayarlamalar yaptığı anlamına gelir. Öğrenme oranı belirli bir epok için aşağıdaki gibi hesaplanabilir:

$$LearningRate = alpha \times \left(1 - \frac{Epoch}{MaxEpoch}\right) \tag{25.6}$$

Burada *LearningRate* mevcut epok için öğrenme oranıdır (0 ila *MaxEpoch-1*), alpha eğitim çalışmasının başlangıcında algoritmaya belirtilen öğrenme oranıdır ve *MaxEpoch* çalışmanın başlangıcında da belirtilen algoritmanın çalıştırılacağı toplam epok sayısıdır.

Sonunda aşağıdaki kod kitabı vektörlerini elde ederiz:

```
X1            X2            Y
2.55988367    2.549260936   0
6.048389028   3.195023766   0
7.343461045   3.512289796   1
5.700572642   6.239052716   1
```

Liste 25.4: Güncellenmiş Kod Kitabı Vektör Değerleri.

Kod kitabı vektörlerinin oturması için sürecin muhtemelen daha düşük bir öğrenme oranıyla (örneğin 0,3) 10 ila 20 epok daha tekrarlanması gerekir.

## 25.3 LVQ ile Tahminler Yapın

Kod kitabı vektörleri öğrenildikten sonra, tahminler yapmak için kullanılabilirler. Tahmin yapmak için KNN ile aynı prosedürü kullanabiliriz, ancak K 1 olarak ayarlanır. Tıpkı öğrenmede olduğu gibi

sürecinde, yeni bir veri örneği için BMU'yu bulmak için bir mesafe ölçüsü kullanıyoruz. BMU'yu güncellemek yerine, modelimiz için tahmin haline gelen sınıf değerini döndürüyoruz. Yukarıda hazırlanan kod kitabı vektörlerini ve Öklid mesafe ölçüsünü kullanarak, veri kümesindeki her örnek için aşağıdaki tahminleri yapabiliriz:

```
X1            X2            Tahmin     Y
3.393533211   2.331273381   0          0
3.110073483   1.781539638   0          0
1.343808831   3.368360954   0          0
3.582294042   4.67917911    0          0
2.280362439   2.866990263   0          0
7.423436942   4.696522875   1          1
5.745051997   3.533989803   0          1
9.172168622   2.511101045   1          1
7.792783481   3.424088941   1          1
7.939820817   0.791637231   1          1
```

Liste 25.5: Kod Kitabı Vektörleri ile Tahminler.

Bunu *Y* için veri kümesindeki gerçek değerlerle karşılaştırırsak bazı hatalar görebiliriz. Sınıflandırma doğruluğu şu şekilde hesaplanabilir:

$$\begin{aligned} accuracy &= \frac{count(correct)}{count(\text{instances})} \times 100 \\ doğruluk &= \frac{9}{10} \times 100 \\ doğruluk &= \%90 \end{aligned} \tag{25.7}$$

Biraz daha eğitimle kod kitabı vektörleri daha doğru hale gelecektir.

## 25.4 Özet

Bu bölümde, bir ikili sınıflandırma problemi için LVQ makine öğrenimi algoritmasının sıfırdan nasıl uygulanacağını keşfettiniz. Şunları öğrendiniz:

Bir LVQ modeli nasıl başlatılır.

Bir eğitim örneği için En İyi Eşleşen Birim Nasıl Güncellenir?

Bir ve birden fazla dönem için LVQ modeli nasıl güncellenir.

Tahmin yapmak için öğrenilmiş bir LVQ modeli nasıl kullanılır?

Artık sınıflandırma için sıfırdan Öğrenme Vektör Nicelemesini nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde sınıflandırma için Destek Vektör Makinesi makine öğrenimi algoritmasını keşfedeceksiniz.

# Bölüm 26

# Destek Vektör Makineleri

Destek Vektör Makineleri belki de en popüler ve hakkında en çok konuşulan makine öğrenimi algoritmalarından biridir. Bu algoritmalar 1990'larda geliştirildikleri dönemde son derece popülerdi ve az ayarlama ile yüksek performanslı bir algoritma için başvurulan yöntem olmaya devam ediyor. Bu bölümde Destek Vektör Makinesi (SVM) makine öğrenimi algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Destek vektör makinelerine atıfta bulunmak için kullanılan birçok isim nasıl ayrıştırılır?

Model aslında diskte depolandığında SVM tarafından kullanılan temsil.

Öğrenilmiş bir SVM model temsilinin yeni veriler için tahminler yapmak üzere nasıl kullanılabileceği.

Eğitim verilerinden bir SVM modeli nasıl öğrenilir.

SVM algoritması için verilerinizi en iyi şekilde nasıl hazırlayabilirsiniz?

SVM hakkında daha fazla bilgi almak için nereye

bakabilirsiniz. Hadi başlayalım.

## 26.1 Maximal-Margin Sınıflandırıcı

Maximal-Margin Sınıflandırıcı, DVM'nin pratikte nasıl çalıştığını en iyi açıklayan varsayımsal bir sınıflandırıcıdır. Verilerinizdeki (sütunlar) sayısal girdi değişkenleri ($x$) n boyutlu bir uzay oluşturur. Örneğin, iki girdi değişkeniniz olsaydı, bu iki boyutlu bir uzay oluşturacaktı. Bir hiper düzlem, girdi değişkeni uzayını bölen bir çizgidir. DVM'de, girdi değişkeni uzayındaki noktaları sınıflarına göre (sınıf 0 veya sınıf 1) en iyi şekilde ayırmak için bir hiper düzlem seçilir. İki boyutta bunu bir çizgi olarak görselleştirebilirsiniz ve tüm girdi noktalarımızın bu çizgi tarafından tamamen ayrılabileceğini varsayalım. Örneğin:

$$B0 + (B1 \times X1) + (B2 \times X2) = 0 \quad (26.1)$$

Burada doğrunun eğimini ve kesişme noktasını (*B0*) belirleyen katsayılar (B1 ve *B2*) öğrenme algoritması tarafından bulunur ve X1 ve X2 iki girdi değişkenidir. Bu doğruyu kullanarak sınıflandırmalar yapabilirsiniz. Girdi değerlerini doğru denklemine yerleştirerek, yeni bir noktanın d o ğ r u n u n üstünde mi yoksa altında mı olduğunu hesaplayabilirsiniz.

Çizginin üzerinde, denklem 0'dan büyük bir değer döndürür ve nokta birinci sınıfa (sınıf 0) aittir.

Çizginin altında, denklem 0'dan küçük bir değer verir ve nokta ikinci sınıfa (sınıf 1) aittir.

Çizgiye yakın bir değer sıfıra yakın bir değer verir ve noktanın sınıflandırılması zor olabilir.

Değerin büyüklüğü büyükse, model tahmine daha fazla güvenebilir.

Çizgi ile en yakın veri noktaları arasındaki mesafe marj olarak adlandırılır. İki sınıfı ayırabilecek en iyi veya en uygun çizgi, en büyük marj olarak çizgidir. Buna Maksimal-Marjin hiper düzlemi denir. Marjin, çizgiden sadece en yakın noktalara olan dik mesafe olarak hesaplanır. Doğrunun tanımlanmasında ve sınıflandırıcının oluşturulmasında yalnızca bu noktalar önemlidir. Bu noktalar destek vektörleri olarak adlandırılır. Bunlar hiper düzlemi destekler veya tanımlar. Hiper düzlem, marjı maksimize eden bir optimizasyon prosedürü kullanılarak eğitim verilerinden öğrenilir.

## 26.2 Yumuşak Marj Sınıflandırıcı

Pratikte, gerçek veriler dağınıktır ve bir hiper düzlem ile mükemmel bir şekilde ayrılamaz. Sınıfları ayıran çizginin marjını maksimize etme kısıtlaması gevşetilmelidir. Bu genellikle yumuşak marj sınıflandırıcısı olarak adlandırılır. Bu değişiklik, eğitim verilerindeki bazı noktaların ayırma çizgisini ihlal etmesine izin verir. Her bir boyutta marjda kıpırdanma payı sağlayan ek bir katsayı seti eklenir. Bu katsayılar bazen gevşek değişkenler olarak adlandırılır. Bu karmaşıklığı sağlamak için modelin verilere uydurması gereken daha fazla parametre olduğundan bu durum modelin karmaşıklığını artırır.

Tüm boyutlarda izin verilen kıpırdanmanın büyüklüğünü tanımlayan ve basitçe *C* olarak adlandırılan bir ayarlama parametresi eklenmiştir. *C* parametresi, izin verilen marjın ihlal miktarını tanımlar. $C = 0$ olduğunda ihlal olmaz ve yukarıda açıklanan esnek olmayan Maksimal Kenar Boşluğu Sınıflandırıcısına geri döneriz. *C* değeri büyüdükçe hiper düzlemin daha fazla ihlaline izin verilir. Hiper düzlemin verilerden öğrenilmesi sırasında, marj mesafesi içinde kalan tüm eğitim örnekleri hiper düzlemin yerleşimini etkileyecek ve destek vektörleri olarak adlandırılacaktır. *C*, marjin içinde kalmasına izin verilen örneklerin sayısını etkilediğinden, *C* model tarafından kullanılan destek vektörlerinin sayısını etkiler.

*C* değeri ne kadar küçük olursa, algoritma eğitim verilerine o kadar duyarlı olur (daha yüksek varyans ve daha düşük yanlılık).

*C* değeri ne kadar büyük olursa, algoritma eğitim verilerine o kadar az duyarlı olur (daha düşük varyans ve daha yüksek yanlılık).

## 26.3 Destek Vektör Makineleri (Kerneller)

DVM algoritması pratikte bir kernel kullanılarak uygulanır. Doğrusal DVM'de hiper düzlemin öğrenilmesi, problemin bazı doğrusal cebirler kullanılarak dönüştürülmesiyle yapılır.

DVM'ye bu girişin kapsamı. Doğrusal DVM'nin, gözlemlerin kendileri yerine verilen herhangi iki gözlemin iç çarpımı kullanılarak yeniden ifade edilebileceği güçlü bir içgörüdür. İki vektör arasındaki iç çarpım, her bir girdi değeri çiftinin çarpımının toplamıdır. Örneğin, [2, 3] ve [5, 6] vektörlerinin iç çarpımı 2 × 5 + 3 × 6 veya 28.

Girdi ($x$) arasındaki nokta çarpımını kullanarak yeni bir girdi için tahminde bulunma denklemi ve her bir destek vektörü ($x_i$) aşağıdaki gibi hesaplanır:

$$f(x) = B0 + \sum_{i=1}^{n} (a_i \times (x \times x_i)) \tag{26.2}$$

Bu, yeni bir girdi vektörünün ($x$) eğitim verilerindeki tüm destek vektörleriyle iç çarpımlarının hesaplanmasını içeren bir denklemdir. *B0* ve $a_i$ katsayıları (her bir girdi için) öğrenme algoritması tarafından eğitim verilerinden tahmin edilmelidir.

## 26.3.1 Doğrusal Çekirdek DVM

Nokta çarpımı çekirdek olarak adlandırılır ve şu şekilde yeniden yazılabilir:

$$K(x, x_i) = \sum (x \times x)_i \tag{26.3}$$

Çekirdek, yeni veriler ile destek vektörleri arasındaki benzerliği veya bir mesafe ölçüsünü tanımlar. Nokta çarpımı, doğrusal DVM veya doğrusal bir çekirdek için kullanılan benzerlik ölçüsüdür çünkü mesafe, girdilerin doğrusal bir kombinasyonudur. Polinom Kernel ve Radyal Kernel gibi girdi uzayını daha yüksek boyutlara dönüştüren başka kerneller de kullanılabilir. Buna Çekirdek Hilesi denir. Daha karmaşık çekirdeklerin kullanılması arzu edilir çünkü bu sayede çizgiler eğri veya daha karmaşık olan sınıfları ayırabilir. Bu da daha doğru sınıflandırıcılara yol açabilir.

## 26.3.2 Polinom Çekirdek DVM

Nokta çarpımı yerine, örneğin bir polinom çekirdeği kullanabiliriz:

$$K(x, x_i) = 1 + \sum (x \times x)_i^d \tag{26.4}$$

Burada polinomun derecesi öğrenme algoritmasına elle belirtilmelidir. $d = 1$ olduğunda bu doğrusal çekirdek ile aynıdır. Polinom çekirdeği, girdi uzayında eğri çizgilere izin verir.

## 26.3.3 Radyal Çekirdek DVM

Son olarak, daha karmaşık bir radyal çekirdeğe de sahip olabiliriz. Örneğin:

$$K(x, x_i) = e^{-gamma \times \sum ((x - x^2)_i} \tag{26.5}$$

Burada gama, öğrenme algoritmasına belirtilmesi gereken bir parametredir. Gama için iyi bir varsayılan değer 0,1'dir; burada gama genellikle $0 < gama < 1$'dir. Radyal çekirdek çok yereldir ve iki boyutlu bir uzayda kapalı çokgenler gibi özellik uzayında karmaşık bölgeler oluşturabilir.

## 26.4 SVM Modeli Nasıl Öğrenilir?

SVM modelinin bir optimizasyon prosedürü kullanılarak çözülmesi gerekir. Hiper düzlemin katsayılarını aramak için sayısal bir optimizasyon prosedürü kullanabilirsiniz. Bu verimsizdir ve LIBSVM gibi yaygın olarak kullanılan SVM uygulamalarında kullanılan yaklaşım değildir. Algoritmayı bir alıştırma olarak uyguluyorsanız, alt gradyan inişi adı verilen gradyan inişinin bir varyasyonunu kullanabilirsiniz.

Optimizasyon problemini bir Karesel Programlama problemi olarak yeniden formüle eden özel optimizasyon prosedürleri vardır. DVM'yi uydurmak için en popüler yöntem, çok verimli olan Sıralı Minimal Optimizasyon (SMO) yöntemidir. Bu yöntem, problemi sayısal olarak (arama veya optimizasyon yoluyla) çözmek yerine analitik olarak (hesaplama yoluyla) çözülebilecek alt problemlere ayırır.

## 26.5 Verileri SVM İçin Hazırlama

Bu bölümde, bir SVM modeli öğrenirken eğitim verilerinizi en iyi şekilde nasıl hazırlayacağınıza ilişkin bazı öneriler listelenmektedir.

**Sayısal Girdiler**: SVM girdilerinizin sayısal olduğunu varsayar. Kategorik girdileriniz varsa bunları ikili kukla değişkenlere (her kategori için bir değişken) dönüştürmeniz gerekebilir.

**İkili Sınıflandırma:** Bu bölümde açıklanan temel DVM ikili (iki sınıflı) sınıflandırma problemleri için tasarlanmıştır. Bununla birlikte, regresyon ve çok sınıflı sınıflandırma için uzantılar geliştirilmiştir.

## 26.6 Özet

Bu bölümde makine öğrenimi için Destek Vektör Makinesi Algoritmasını keşfettiniz. Hakkında bilgi edindiniz:

SVM'yi anlamak için basit bir teorik model sağlayan Maximal-Margin Sınıflandırıcısı.

Yumuşak Marj Sınıflandırıcısı, gerçek verilerdeki gürültülü sınıf sınırlarını ele almak için marjı gevşetmek üzere Maksimal Marj Sınıflandırıcısının bir modifikasyonudur.

Destek Vektör Makineleri ve öğrenme algoritmasının bir nokta-çarpım çekirdeği olarak nasıl yeniden formüle edilebileceği ve Polinom ve Radyal gibi diğer çekirdeklerin nasıl kullanılabileceği.

Hiper düzlemi öğrenmek için sayısal optimizasyonu nasıl kullanabileceğinizi ve verimli uygulamaların Sıralı Minimal Optimizasyon adı verilen alternatif bir optimizasyon şeması kullandığını.

Artık Destek Vektör Makinesi algoritması hakkında bilgi sahibisiniz. Bir sonraki bölümde alt gradyan inişi kullanarak SVM'yi sıfırdan nasıl uygulayabileceğinizi keşfedeceksiniz.

# Bölüm 27

# Destek Vektör Makinesi Eğitimi

Destek Vektör Makineleri esnek, parametrik olmayan bir makine öğrenimi algoritmasıdır. Bu bölümde, alt gradyan inişi kullanarak Destek Vektör Makinesi algoritmasını adım adım nasıl uygulayacağınızı keşfedeceksiniz. Bu bölümü tamamladıktan sonra şunları öğreneceksiniz:

Bir SVM modelinin katsayılarını güncellemek için alt gradyan inişi nasıl kullanılır.

Eğitim verileri için bir SVM modeli öğrenmek üzere alt gradyan iniş algoritması nasıl yinelenir.

Öğrenilmiş bir SVM modeli ile tahminler nasıl yapılır. Hadi

başlayalım.

## 27.1 Öğretici Veri Seti

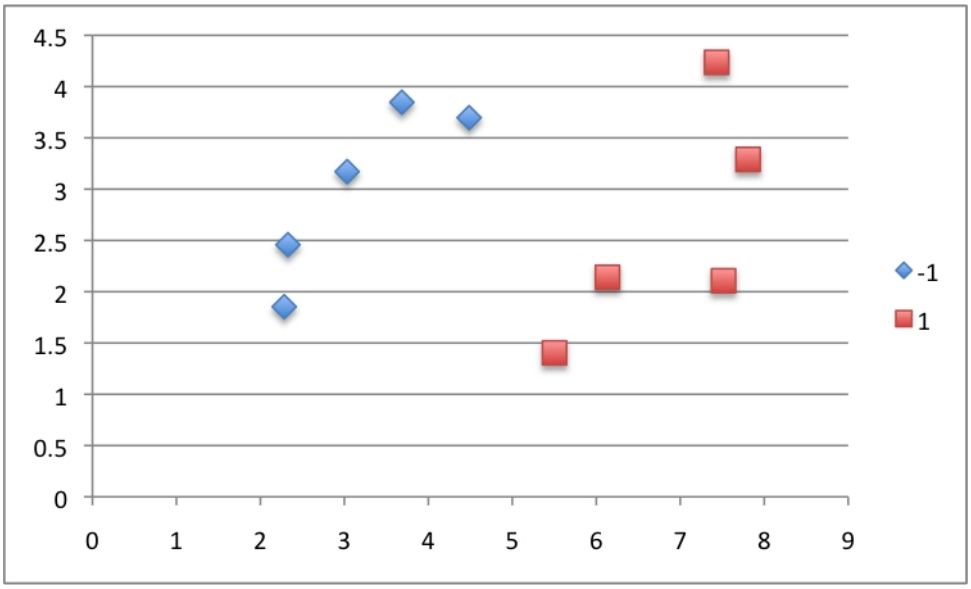
Sınıfların doğrusal olarak ayrılabilir olması için bir test problemi geliştirilmiştir. Bu, sınıfları ayırmak için düz bir çizgi çizilebileceği anlamına gelir. Bu, doğrusal bir çekirdeğe (düz çizgi) sahip bir DVM'nin nasıl uygulanacağını keşfedebilmemiz için kasıtlıdır. DVM algoritması, birinci sınıf değerinin -1 ve ikinci sınıf değerinin +1 olmasıdır.

```
X1            X2            Y
2.327868056   2.458016525   -1
3.032830419   3.170770366   -1
4.485465382   3.696728111   -1
3.684815246   3.846846973   -1
2.283558563   1.853215997   -1
7.807521179   3.290132136   1
6.132998136   2.140563087   1
7.514829366   2.107056961   1
5.502385039   1.404002608   1
7.432932365   4.236232628   1
```

Liste 27.1: SVM Öğretici Veri Seti.

Aşağıdaki görselleştirme, veri kümesinin bir dağılım grafiğini

sunmaktadır.




Şekil 27.1: SVM Öğretici Veri Seti Dağılım grafiği.

# 27.2 Gradyan ile DVM Eğitimi İniş

Bu bölümde DVM modelinin biçimi ve gradyan inişi optimizasyon prosedürünün bir çeşidi kullanılarak nasıl öğrenilebileceği açıklanmaktadır.

## 27.2.1 Doğrusal DVM Modelinin Şekli

Doğrusal DVM modeli bir çizgidir ve öğrenme algoritmasının amacı, katsayılar için sınıfları en iyi şekilde ayıran değerleri bulmaktır. Çizgi tipik olarak şu şekildedir (okunabilirlik için terimleri gruplandırarak):

$$B0 + (B1 \times X1) + (B2 \times X2) = 0 \quad (27.1)$$

Burada *B0*, B1 ve B2 katsayılar, X1 ve X2 ise girdi değişkenleridir. Bu, küçük bir değişiklikle kullanacağımız denklemin şekli olacaktır, ofset veya kesişme olarak da adlandırılan önyargı terimini (*B0*) çıkaracağız. Örneğin $B0 = 0$.

$$(B1 \times X1) + (B2 \times X2) = 0 \quad (27.2)$$

Bu, doğrunun orijinden geçeceği anlamına gelir ($X1 = 0$ ve $X2 = 0$). Bu sadece öğreticiyi takip etmeyi kolaylaştırmak içindir ve basit problemimiz gerçekten buna ihtiyaç duymadığından, isterseniz önyargı terimini geri ekleyebilirsiniz.

## 27.2.2 SVM Optimizasyon Yöntemi

Katsayıları bulmak için optimizasyon algoritması ikinci dereceden bir programlama problemi olarak ifade edilebilir. Bu, hızlı çözücülerin kullanılabileceği bir tür kısıt optimizasyonudur. Bu derste bu yaklaşımı kullanmayacağız. Doğrusal DVM için katsayı değerlerini bulmak için kullanılabilecek bir başka yaklaşım da alt gradyan inişidir. Bu yöntemde her iterasyonda rastgele bir eğitim örüntüsü seçilir ve katsayıları güncellemek için kullanılır. Çok sayıda sonra

Algoritma, binlerce veya yüz binlerce iterasyondan sonra kararlı bir katsayı kümesine yerleşecektir. Katsayı güncelleme denklemi aşağıdaki gibi çalışır. İlk olarak bir çıktı değeri şu şekilde hesaplanır:

$$çıktı = Y \times (\text{B1} \times X1) + (\text{B2} \times X2) \tag{27.3}$$

Çıktı değerine bağlı olarak iki farklı güncelleme prosedürü kullanılır. Çıktı değerinin 1'den büyük olması, eğitim örneğinin bir destek vektörü olmadığını gösterir. Bu, örneğin çıktının hesaplanmasında doğrudan yer almadığı anlamına gelir, bu durumda ağırlıklar biraz azaltılır:

$$b = (1 - \frac{1}{t}) \times b \tag{27.4}$$

Burada b güncellenmekte olan ağırlıktır (*B1* veya *B2* gibi), *t* mevcut iterasyondur (örneğin ilk güncelleme için 1, ikincisi için 2 vb.). Çıktı 1'den küçükse, eğitim örneğinin bir destek vektörü olduğu ve verileri daha iyi açıklamak için güncellenmesi gerektiği varsayılır.

$$b = (1 - \frac{1}{t}) \times b + \frac{1}{lambda \times t} \times (y \times x) \tag{27.5}$$

Burada *b* güncellenmekte olan ağırlık, *t* mevcut iterasyon ve lambda öğrenme algoritmasının bir parametresidir. Lambda bir öğrenme parametresidir ve genellikle 0.0001 veya daha küçük gibi çok küçük değerlere ayarlanır. Prosedür, hata oranı istenen bir seviyeye düşene kadar veya çok büyük bir sabit iterasyon sayısı için tekrarlanır. Daha küçük öğrenme oranları genellikle çok daha uzun eğitim süreleri gerektirir. Yineleme sayısı bu öğrenme algoritmasının bir dezavantajıdır.

# 27.3 Eğitim Verilerinden SVM Modeli Öğrenme

Bu bölümde SVM öğrenme algoritmasını göstermek için katsayılarda birkaç güncelleme yapacağız. Çok büyük bir lambda değeri kullanacağız: *lambda* = 0.45. Bu alışılmadık derecede büyüktür ve her güncellemede çok fazla değişikliği zorlayacaktır. Normalde *lambda* değerleri çok küçüktür.

## 27.3.1 Öğrenme Yinelemesi #1

Katsayıları 0.0 olarak ayarlayarak başlayacağız.

$$\begin{aligned} \text{B1} &= 0.0 \\ \text{B2} &= 0.0 \end{aligned} \tag{27.6}$$

Ayrıca hangi iterasyonda olduğumuzu da takip etmemiz gerekiyor: *t* = 1. Modeli eğitim örüntülerinin sırasını kullanarak eğiteceğiz. İdeal olarak, öğrenme algoritmasının takılıp kalmasını önlemek için kalıpların sırası rastgele olacaktır. Güncellemek için kullanacağımız ilk eğitim modeli

katsayılarıdır: Örnek: *X1* = 2.327868056, *X2* = 2.458016525, *Y* = - 1. Şimdi bu iterasyon için çıktı değerini hesaplayabiliriz.

$$\begin{aligned} &\text{çıktı} = Y \times (B1 \times X1) + (B2 \times X2) \\ &\text{çıktı} = -1 \times (0{,}0 \times 2{,}327868056) + (0{,}0 \times 2{,}458016525) \\ &\text{çıktı} = 0.0 \end{aligned} \tag{27.7}$$

Yeterince kolay. Çıktı 1.0'dan azdır, bu nedenle eğitim modelinin bir destek vektörü olduğunu varsayan daha karmaşık güncelleme prosedürünü kullanacağız:

$$\begin{aligned} &b = (1 - \frac{1}{t}) \times b + \frac{1}{lambda \times t} \times (y \times x) \\ &B1 = (1 - \frac{1}{1}) \times 0.0 + \frac{1}{0.5 \times 1} \times (-1 \times 2.327868056) \\ &B1 = -5.173040124 \end{aligned} \tag{27.8}$$

Ve B2 için bu:

$$\begin{aligned} &B2 = (1 - \frac{1}{1}) \times 0{,}0 + \frac{1}{0.5 \times 1} \times (-1 \times 2{,}458016525) \\ &B2 = -5.462258944 \end{aligned} \tag{27.9}$$

## 27.3.2 Öğrenme Yinelemesi #2

Artık eğitim veri kümesinden ikinci örnekle öğrenme algoritmasının bir sonraki yinelemesinde kullanabileceğimiz güncellenmiş katsayılara sahibiz: Örnek: *X1* = 3.032830419, X2 = 3.170770366, *Y* = 1. Yine, iterasyonu takip etmeliyiz: *t* = 2. İşlemi tekrarlayalım.

$$\begin{aligned} B1 &= -5.173040124 \\ B2 &= -5.462258944 \end{aligned} \tag{27.10}$$

Çıkış değeri şu şekilde hesaplanır:

$$\begin{aligned} &\text{çıktı} = -1 \times (-5.173040124 \times 3.032830419) + (-5.462258944 \times 3.170770366) \\ &\text{çıkış} = 33.00852224 \end{aligned} \tag{27.11}$$

Çıktı değerinin 1,0'dan büyük olması, bu eğitim örneğinin bir destek vektörü olmadığını gösterir. *B1* katsayısını buna göre güncelleyebiliriz:

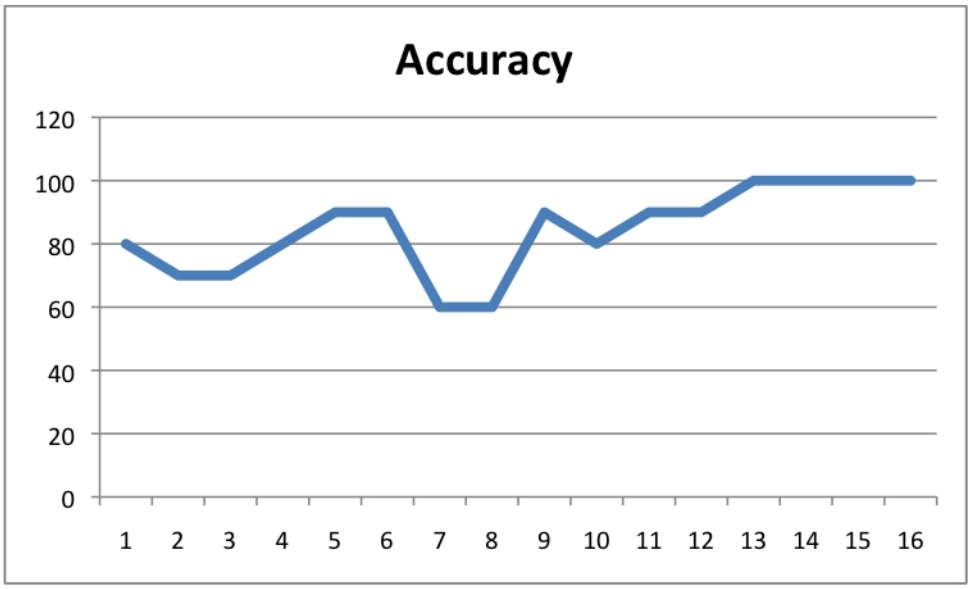
$$\begin{aligned} &b = (1 - \frac{1}{t}) \times b \\ &B1 = (1 - \frac{1}{2}) \times -5.173040124 \\ &B1 = -2.586520062 \end{aligned} \tag{27.12}$$

Ve B2 katsayısı için:

$$\begin{aligned} &B2 = (1 - \frac{1}{2}) \times -5.462258944 \\ &B2 = -2.731129472 \end{aligned} \tag{27.13}$$

### 27.3.3 Daha Fazla Yineleme

Veri kümesinin geri kalanı için bu işlemi tekrarlayın. Veri kümesi üzerinden bir geçiş bir dönem olarak adlandırılır. Şimdi toplam 160 iterasyon için işlemi 15 epok daha tekrarlayın (16 epok 10
epok başına güncelleme). Her dönem için modelin kaybını veya doğruluğunu takip etmek mümkündür. Bu, algoritmanın yakınsayıp yakınsamadığı veya uygulamada bir hata olup olmadığı konusunda fikir edinmenin harika bir yoludur. Her dönemin sonunda modelin doğruluğunu çizerseniz, aşağıdaki grafiğe benzeyen bir şey görmelisiniz:




Şekil 27.2: Destek Vektör Makinesi Model Doğruluğu.

Eğitim verilerinde 16 epoktan sonra %100'lük bir doğruluk elde ettiğimizi görebilirsiniz. Katsayılar için aşağıdaki gibi görünen nihai değerlere ulaşmalısınız:

$$\begin{aligned} B1 &= 0.552391765 \\ B2 &= -0.724533592 \end{aligned} \tag{27.14}$$

Bu nedenle öğrenilen hiper düzlemin şekli şöyledir:

$$0 + (0.552391765 \times X1) + (-0.724533592 \times X2) = 0 \tag{27.15}$$

## 27.4 SVM Modeli ile Tahminler Yapın

Artık çizgi için katsayılarımız olduğuna göre, tahminler yapabiliriz. Bu bölümde eğitim verileri için tahminler yapacağız, ancak bu yeni veriler için tahminler yapmak için kolayca uyarlanabilir. Tahminler aşağıdaki denklem kullanılarak yapılabilir:

$$\begin{aligned} &çıktı = (B1 \times X1) + (B2 \times X2) \\ &Y = -1 \text{ EĞER } çıktı < \\ &0 \text{ ise} \\ &Y = +1 \text{ EĞER } çıkış > \\ &0 \text{ ise} \end{aligned} \tag{27.16}$$

Yukarıdaki katsayıları ve bu tahmin kurallarını kullanarak, eğitim veri kümesindeki her örnek için bir tahmin yapabiliriz:

```
Çıktı          Gevrek  Y
-0.495020399   -1      -1
-0.622019096   -1      -1
-0.20066956    -1      -1
-0.75170826    -1      -1
-0.081298299   -1      -1
1.928999147    1       1
1.836907801    1       1
2.624496306    1       1
2.022225129    1       1
1.036597783    1       1
```

Liste 27.2: Öğrenilmiş DVM Modelinin Doğruluğu.

Net tahmini (*Crisp*) beklenen çıktı sütunuyla ($Y$ ) karşılaştırdığımızda, modelimizin %100 doğruluk elde ettiğini görebiliriz.

## 27.5 Özet

Bu bölümde, alt gradyan iniş optimizasyon tekniğini kullanarak Destek Vektör Makinesi algoritmasını sıfırdan nasıl uygulayacağınızı keşfettiniz. Şunları öğrendiniz:

Alt gradyan inişi kullanarak SVM için katsayı değerleri nasıl güncellenir.

İyi katsayı değerleri bulmak için alt gradyan iniş prosedürü nasıl yinelenir.

Öğrenilmiş bir SVM modeli kullanarak tahminler nasıl yapılır.

Artık Destek Vektör Makinesi algoritmasını alt gradyan inişi kullanarak sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bu, doğrusal olmayan makine öğrenimi algoritmalarına girişinizi tamamlıyor. Bir sonraki bölümde torbalama işleminden başlayarak topluluk makine öğrenimi algoritmalarını keşfedeceksiniz.

# Bölüm V

# Topluluk Algoritmaları

# Bölüm 28

# Bagging ve Rastgele Orman

Random Forest en popüler ve en güçlü makine öğrenimi algoritmalarından biridir. Bootstrap Aggregation veya bagging olarak adlandırılan bir tür topluluk makine öğrenimi algoritmasıdır. Bu bölümde, tahminsel modelleme için Bagging ensemble algoritmasını ve Random Forest algoritmasını keşfedeceksiniz. Bu bölümü okuduktan sonra aşağıdakiler hakkında bilgi sahibi olacaksınız:

Örneklerden istatistiksel büyüklükleri tahmin etmek için bootstrap yöntemi.

Tek bir eğitim veri setinden birden fazla farklı model oluşturmak için Bootstrap Aggregation algoritması.

Rastgele Orman algoritması, Bagging'de küçük bir değişiklik yapar ve çok güçlü bir sınıflandırıcı ile sonuçlanır.

Hadi başlayalım.

## 28.1 Bootstrap Yöntemi

Bagging konusuna geçmeden önce, bootstrap adı verilen önemli bir temel tekniğe hızlıca göz atalım. Bootstrap, bir veri örneğinden bir niceliği tahmin etmek için kullanılan güçlü bir istatistiksel yöntemdir. Bunu anlamak için en kolay yol, bu niceliğin ortalama veya standart sapma gibi tanımlayıcı bir istatistik olmasıdır. Elimizde 100 değerden ($x$) oluşan bir örneklem olduğunu ve örneklemin ortalamasına ilişkin bir tahmin elde etmek istediğimizi varsayalım. Ortalamayı doğrudan örneklemden şu şekilde hesaplayabiliriz:

$$ortalama(x) = \frac{1}{100} \sum_{i=1}^{100} x_i \tag{28.1}$$

Örneklemimizin küçük olduğunu ve ortalamamızda hata olduğunu biliyoruz. Bootstrap prosedürünü kullanarak ortalamamızın tahminini iyileştirebiliriz:

1. Veri setimizden çok sayıda (örneğin 1000) rastgele alt örnek oluşturun (yani aynı değeri birden çok kez seçebiliriz).
2. Her bir alt örneğin ortalamasını hesaplayın.

3. Topladığımız tüm ortalamaların ortalamasını hesaplayın ve bunu veriler için tahmini ortalamamız olarak kullanın.

Örneğin, diyelim ki 3 yeniden örnekleme kullandık ve 2,3, 4,5 ve 3,3 ortalama değerlerini elde ettik. Bunların ortalamasını alarak verilerin tahmini ortalamasını 3,367 olarak alabiliriz. Bu işlem, standart sapma gibi diğer nicelikleri ve hatta öğrenilen katsayılar gibi makine öğrenimi algoritmalarında kullanılan nicelikleri tahmin etmek için kullanılabilir.

## 28.2 Bootstrap Aggregation (Bagging)

Bootstrap Aggregation (veya kısaca Bagging), basit ve çok güçlü bir topluluk yöntemidir. Topluluk yöntemi, herhangi bir modelden daha doğru tahminler yapmak için birden fazla makine öğrenimi algoritmasının tahminlerini bir araya getiren bir tekniktir. Bootstrap Aggregation, yüksek varyansa sahip algoritmaların varyansını azaltmak için kullanılabilecek genel bir prosedürdür. Yüksek varyansa sahip bir algoritma, sınıflandırma ve regresyon ağaçları (CART) gibi karar ağaçlarıdır.

Karar ağaçları, üzerinde eğitildikleri belirli verilere karşı hassastır. Eğitim verileri değiştirilirse (örneğin bir ağaç eğitim verilerinin bir alt kümesi üzerinde eğitilirse) ortaya çıkan karar ağacı oldukça farklı olabilir ve buna bağlı olarak tahminler de oldukça farklı olabilir. Bagging, Bootstrap prosedürünün yüksek varyanslı bir makine öğrenimi algoritmasına, tipik olarak karar ağaçlarına uygulanmasıdır. Elimizde 1000 örnekten oluşan bir veri kümesi olduğunu ve CART algoritmasını kullandığımızı varsayalım. CART algoritmasının torbalanması aşağıdaki gibi çalışacaktır.

1. Veri setimizden çok sayıda (örneğin 100) rastgele alt örnek oluşturun.
2. Her örnek üzerinde bir CART modeli eğitin.
3. Yeni bir veri kümesi verildiğinde, her modelin ortalama tahminini hesaplayın.

Örneğin, bir girdi örneği için şu sınıf tahminlerini yapan 5 torbalanmış karar ağacımız olsaydı: mavi, mavi, kırmızı, mavi ve kırmızı, en sık görülen sınıfı alır ve maviyi tahmin ederdik. Karar ağaçları ile torbalama yaparken, tek tek ağaçların eğitim verilerine aşırı uyum sağlaması konusunda daha az endişe duyarız. Bu nedenle ve verimlilik için, bireysel karar ağaçları derin büyütülür (örneğin, ağacın her yaprak düğümünde birkaç eğitim örneği) ve ağaçlar budanmaz. Bu ağaçlar hem yüksek varyansa hem de düşük yanlılığa sahip olacaktır. Bunlar, torbalama kullanarak tahminleri birleştirirken alt modellerin önemli özellikleridir.

Karar ağaçlarını torbalarken tek parametre oluşturulacak ağaç sayısıdır. Bu, doğrulukta iyileşme görülmemeye başlayana kadar (örneğin çapraz doğrulama test demetinde) çalıştırmadan sonra ağaç sayısı artırılarak seçilebilir. Çok sayıda karar ağacı oluşturmak uzun zaman alabilir, ancak eğitim verilerine aşırı uyum sağlamayacaktır. Tıpkı karar ağaçlarının kendileri gibi, Torbalama da sınıflandırma ve regresyon problemleri için kullanılabilir.

## 28.3 Rastgele Orman

Rastgele Ormanlar, torbalanmış karar ağaçlarına göre bir gelişmedir. CART gibi karar ağaçlarının bir sorunu da açgözlü olmalarıdır. Hangi değişkene bölüneceklerini bir

hatayı en aza indiren açgözlü algoritma. Bu nedenle, Torbalama ile bile, karar ağaçları çok fazla yapısal benzerliğe sahip olabilir ve bu da tahminlerinde yüksek korelasyona neden olabilir. Alt modellerden gelen tahminler korelasyonsuz veya en iyi ihtimalle zayıf korelasyonluysa, birden fazla modelden gelen tahminleri topluluklarda birleştirmek daha iyi sonuç verir.

Rastgele orman, alt ağaçların öğrenilme şekli için algoritmayı değiştirir, böylece tüm alt ağaçlardan elde edilen tahminler daha az korelasyona sahip olur. Bu basit bir değişikliktir. CART'ta, bir ayırma noktası seçerken, öğrenme algoritmasının en uygun ayırma noktasını seçmek için tüm değişkenleri ve tüm değişken değerlerini incelemesine izin verilir. Rastgele orman algoritması bu prosedürü değiştirir, böylece öğrenme algoritması aranacak rastgele bir özellik örneği ile sınırlandırılır. Her bölme noktasında aranabilecek özellik sayısı (m) algoritmaya bir parametre olarak belirtilmelidir. Farklı değerler deneyebilir ve çapraz doğrulama kullanarak ayarlayabilirsiniz.

Sınıflandırma için iyi bir varsayılan değer: $m = \sqrt{p}$'dir.

Regresyon için iyi bir varsayılan değer: $m = p/3$

Burada $m$, bir bölünme noktasında aranabilecek rastgele seçilmiş özelliklerin sayısı ve $p$ ise girdi değişkenlerinin sayısıdır. Örneğin, bir veri kümesi bir sınıflandırma problemi için 25 girdi değişkenine sahipse, o zaman:

$$\begin{aligned} m &= \sqrt{25} \\ m &= 5 \end{aligned} \tag{28.2}$$

## 28.4 Tahmini Performans

Eğitim verilerinden alınan her bootstrap örneği için, geride dahil edilmeyen örnekler kalacaktır. Bu örneklere Torba Dışı örnekler veya OOB denir. Ortalaması alındığında her modelin dışarıda kalan örneklerdeki performansı, torbalanmış modellerin tahmini doğruluğunu sağlayabilir. Bu tahmini performans genellikle OOB tahmini olarak adlandırılır. Bu performans ölçümleri test hatasının güvenilir bir tahminidir ve çapraz doğrulama hata tahminleriyle iyi korelasyon gösterir.

## 28.5 Değişken Önem

Torbalı karar ağaçları oluşturulurken, her bir bölünme noktasında bir değişken için hata fonksiyonunun ne kadar düştüğünü hesaplayabiliriz. Regresyon problemlerinde bu, toplam karesel hatadaki düşüş olabilir ve sınıflandırmada bu Gini puanı olabilir. Hatadaki bu düşüşlerin tüm karar ağaçlarında ortalaması alınabilir ve her bir girdi değişkeninin önemine ilişkin bir tahmin sağlamak için çıktı alınabilir. Değişken seçildiğinde düşüş ne kadar büyükse, önem de o kadar büyüktür.

Bu çıktılar, problemle en çok veya en az ilgili olabilecek girdi değişkenlerinin alt kümelerinin belirlenmesine yardımcı olabilir ve bazı özelliklerin veri kümesinden çıkarıldığı olası özellik seçimi deneylerini önerebilir.

## 28.6 Torbalı CART İçin Veri Hazırlama

Torbalı CART, problemin iyi bir temsili dışında herhangi bir özel veri hazırlığı gerektirmez.

## 28.7 Özet

Bu bölümde Bagging topluluk makine öğrenimi algoritmasını ve Random Forest adı verilen popüler varyasyonunu keşfettiniz. Şunları öğrendiniz:

Bir veri örneğinden istatistiksel büyüklükler nasıl tahmin edilir.

Torbalama kullanarak birden fazla yüksek varyanslı modelden gelen tahminler nasıl birleştirilir?

Rastgele Ormanlar adı verilen bir teknikle, tahminlerini korelasyondan arındırmak için torbalama yaparken karar ağaçlarının yapısında nasıl değişiklik yapılır.

Artık torbalama topluluğu algoritması, torbalanmış karar ağaçları ve rastgele ormanlar hakkında bilgi sahibisiniz. Bir sonraki bölümde torbalanmış karar ağaçlarını sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 29

# Torbalı Karar Ağaçları Eğitimi

Torbalama, neredeyse her zaman temel modellere göre performansta artış sağlayan basit bir topluluk yöntemidir. Bu bölümde torbalanmış karar ağaçlarını adım adım nasıl uygulayacağınızı keşfedeceksiniz. Bu bölümü okuduktan sonra şunları keşfedeceksiniz:

Eğitim veri setinizin bootstrap örneklerinden karar ağaçları nasıl oluşturulur?

Birden fazla bootstrap örneğinden elde edilen tahminler nasıl toplanır?

Torbalama, bireysel yüksek varyanslı modellerden nasıl daha doğru tahminler

oluşturabilir? Hadi başlayalım.

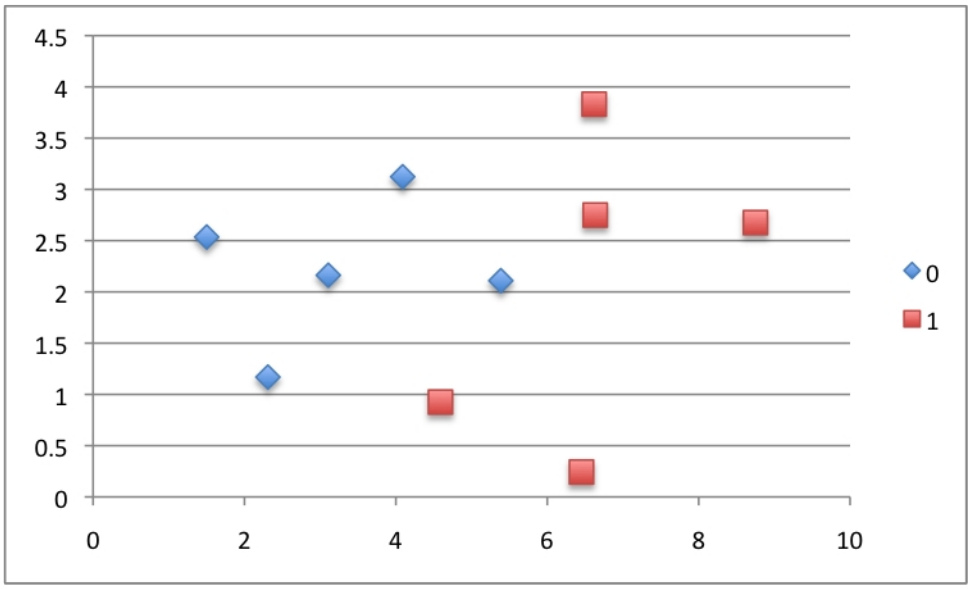
## 29.1 Öğretici Veri Seti

Bu eğitimde iki girdi değişkeni (X1 ve *X2*) ve bir çıktı değişkeni (*Y* ) içeren veri kümesi kullanacağız. Giriş değişkenleri Gauss dağılımından çekilen gerçek değerli rastgele sayılardır. Çıktı değişkeninin iki değeri vardır, bu da problemi ikili sınıflandırma problemi yapar. Ham veriler aşağıda listelenmiştir.

```
X1            X2            Y
2.309572387   1.168959634   0
1.500958319   2.535482186   0
3.107545266   2.162569456   0
4.090032824   3.123409313   0
5.38660215    2.109488166   0
6.451823468   0.242952387   1
6.633669528   2.749508563   1
8.749958452   2.676022211   1
4.589131161   0.925340325   1
6.619322828   3.831050828   1
```

Liste 29.1: Torbalama Öğretici Veri Seti.

Aşağıda veri kümesinin bir grafiği yer almaktadır. Sınıfları ayırmak için düz bir çizgi çizemediğimizi görüyorsunuz.




Şekil 29.1: Torbalama Eğitimi Veri Seti Dağılım grafiği.

## 29.2 Torbalı Karar Ağacı Modelini öğrenin

Bootstrap Aggregation (veya kısaca Bagging), basit ve çok güçlü bir topluluk yöntemidir. Torbalama, eğitim veri setinizin bir alt örneğini alarak (değiştirme ile) ve bir model, genellikle yüksek varyansa sahip oldukları için bir karar ağacı oluşturarak çalışır. Bu işlem tekrarlanarak istediğiniz kadar ağaç oluşturulur. Daha sonra yeni veriler için tahminler yapılırken, her bir ağaçtan elde edilen çıktılar ortalama alınarak birleştirilir.

CART gibi karar ağaçları, sınıflandırma için Gini endeksi gibi bir maliyet fonksiyonunu en aza indirmeyi amaçlayan açgözlü bir algoritma kullanarak bölünme noktalarını seçer. Ağaç, eğitim verilerinin rastgele bir örneği üzerinde oluşturulduğunda, algoritmanın farklı ayrım noktaları seçmesi muhtemeldir, bu da farklı ağaçlara ve dolayısıyla farklı tahminlere neden olur. Torbalama işleminin gücü, probleme çok farklı bakış açıları olan modellerden gelen tahminleri birleştirmekten gelir. Torbalanmış karar ağaçlarının ilginç kısmı, topluluk tahminleri yapmak için nasıl birleştirildikleridir. Bunu akılda tutarak, eğitim verilerinden her biri düşük doğruluğa sahip 3 karar ağacı oluşturacağız. Bölünme noktaları, aynı soruna ilişkin farklı bakış açılarının daha yüksek performans sağlamak için nasıl birleştirilebileceğini tam olarak göstermek için manuel olarak seçilmiştir.

Bir bölünme noktası olan bir karar ağacına karar kütüğü denir, bunun nedeni k o n u ş u l a c a k çok az ağaç olmasıdır. Üç karar kütüğü kullanacağız. Her biri için bölünme noktaları aşağıdaki gibidir:

$$\begin{aligned} &\text{Model1} : X1 \leq 5.38660215 \\ &\text{Model2} : X1 \leq 4.090032824 \\ &\text{Model3} : X2 \leq 0.925340325 \end{aligned} \tag{29.1}$$

Bu örneği kendi probleminize uyarlıyorsanız, CART algoritmasını eğitim veri kümesindeki her bir bootstrap örneğine uygulayabilirsiniz. Bu bir sınıflandırma problemi olduğundan, ayırma noktalarını seçmek için Gini endeksi maliyet fonksiyonunu minimize etmek üzere kullanabilirsiniz. Ayrıca, bu örnekte yalnızca 3 model kullanıyoruz, torbalama yaparken genellikle onlarca, yüzlerce ve hatta binlerce model oluşturacaksınız. Artık bootstrap modellerimiz olduğuna göre, bunları nasıl bir araya getirebileceğimize bakalım.

## 29.3 Torbalı Karar Ağaçları ile Tahminler Yapın

Bootstrap modellerinden gelen tahminler daha doğru tahminler üretmek için birleştirilebilir. Bu bölümde, her bir önyükleme modeliyle tahminler yapmayı ve son olarak alt model tahminlerini daha doğru topluluk tahminleri halinde toplamayı inceleyeceğiz.

### 29.3.1 Karar Güdük Modeli 1

İlk model için ayırma noktası X1 5 . 38≤660215'dur. Bu ayırma noktasını kullanarak eğitim verilerini iki gruba ayırabiliriz:

```
Y   Grup
0   SOL
0   SOL
0   SOL
0   SOL
0   SOL
1   DOĞRU
1   DOĞRU
1   DOĞRU
1   SOL
1   DOĞRU
```

Liste 29.2: Model 1 için Örneklerin Gruplandırılması.

SOL grubun çoğunlukla 0 sınıfına ait örnekler içerdiğini ve SAĞ grubun çoğunlukla 1 sınıfına ait örnekler içerdiğini görebiliriz. Bu nedenle, iki gruba atanan örnekler sınıflandırılacaktır:

SOL: sınıf 0

SAĞ: sınıf 1

Bu nedenle her bir eğitim örneği için tahmini hesaplayabiliriz.

```
Tahmin iyon       Doğruluk
0          Hata   %90
           ı 0
0          0
0          0
0          0
0          0
1          0
1          0
1          0
0          1
1          0
```

Liste 29.3: Model 1 için Örneklerin Tahmini.

Modelin yalnızca bir eğitim örneğini (sondan ikinci) yanlış aldığını görebiliriz.
Bu da modele %90'lık bir doğruluk oranı veriyor. Hiç fena değil, ama daha iyisini yapabiliriz.

### 29.3.2 Karar Güdük Modeli 2

İkinci model X1 bölünme noktasını kullanır≤ 4.090032824. Eğitim verilerini iki gruba ayırabiliriz:

```
Y   Grup
0   SOL
0   SOL
0   SOL
0   SOL
0   DOĞRU
1   DOĞRU
1   DOĞRU
1   DOĞRU
1   DOĞRU
1   DOĞRU
```

Liste 29.4: Model 2 için Örneklerin Gruplandırılması.

Yukarıda olduğu gibi, SOL grup örnekleri sınıf 0 ve SAĞ grup sınıf 1 olarak sınıflandıracaktır. Bu modeli kullanarak, eğitim veri kümesindeki tüm örnekler için tahminler yapabiliriz:

```
Tahmin iyon       Doğruluk
0          Hata   %90
           ı 0
0          0
0          0
0          0
1          1
1          0
1          0
1          0
1          0
1          0
```

Liste 29.5: Model 2 için Örneklerin Tahmini.

Yine, ilk modelde olduğu gibi, yalnızca bir örneğin yanlış sınıflandırılmasıyla doğruluğun %90 olduğunu görebiliriz. Bu sefer 5. örnek. Sadece bu iki modelin birleştirilmesi, modellerin tahminlerinde çeliştiği bazı belirsiz tahminlerle sonuçlanır. Üçüncü bir model eklemek işleri açıklığa kavuşturacaktır.

### 29.3.3 Decision Stump Model 3

Üçüncü model tarafından kullanılan ayırma noktası X2 ≤ 0.925340325. Bu ayırma noktasını kullanarak, eğitim veri setini tekrar iki gruba ayırabiliriz.

```
Y   Grup
0   DOĞRU
0   DOĞRU
0   DOĞRU
0   DOĞRU
0   DOĞRU
1   SOL
1   DOĞRU
1   DOĞRU
1   SOL
```

```
1   DOĞRU
```

Liste 29.6: Model 3 için Örneklerin Gruplandırılması.

Bu modelde SAĞ grup örnekleri sınıf 0 ve SOL grup sınıf 1 olarak sınıflandıracaktır. Bu basit modeli kullanarak, her eğitim örneği için tekrar tahminler yapabiliriz.

```
Tahmin iyon       Doğruluk
0          Hata   %70
           ı 0
0          0
0          0
0          0
0          0
1          0
0          1
0          1
1          0
0          1
```

Liste 29.7: Model 3 için Örneklerin Tahmini.

Bu model %70 ile biraz daha kötü bir doğruluğa sahiptir. Daha da önemlisi, hem ikinci son örneği hem de 5. örneği doğru bir şekilde tahmin ederek, önceki iki modelin birleştirilmesi durumunda ortaya çıkabilecek belirsizlikleri ortadan kaldırmaktadır.

## 29.4 Final Tahminler

Artık bootstrap modellerinden tahminlerimiz olduğuna göre, tahminleri bir topluluk tahmininde toplayabiliriz. Bunu, her bir eğitim örneği için her bir model tahmininin modunu alarak yapabiliriz. Mod, en yaygın değeri seçen istatistiksel bir işlevdir. Bu durumda, tahmin edilen en yaygın sınıf değeri. Bu basit prosedürü kullanarak aşağıdaki tahminleri elde ederiz:

```
Tahmin      Y   Hata    Doğruluk
0           0   0       100
0           0   0
0           0   0
0           0   0
0           0   0
1           1   0
1           1   0
1           1   0
1           1   0
1           1   0
```

Liste 29.8: Torbalı Model için Örneklerin Tahmini.

Tüm 10 eğitim örneğinin %100 doğrulukla doğru sınıflandırıldığını görebilirsiniz.

## 29.5 Özet

Bu bölümde torbalanmış karar ağaçlarını keşfettiniz. Öğrendiniz.

Bootstrap veri örneklerinden çoklu karar güdük modelleri nasıl oluşturulur.

Birden fazla karar ağacı modelinden gelen tahminler nasıl toplanır?

Artık torbalanmış karar ağaçlarını sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bir sonraki bölümde boosting ensemble yöntemini ve AdaBoost makine öğrenimi algoritmasını keşfedeceksiniz.

# Bölüm 30

# Boosting ve AdaBoost

Boosting, bir dizi zayıf sınıflandırıcıdan güçlü bir sınıflandırıcı oluşturmaya çalışan bir topluluk tekniğidir. Bu bölümde makine öğrenimi için AdaBoost Ensemble yöntemini keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Güçlendirme topluluğu yönteminin ne olduğu ve genel olarak nasıl çalıştığı.

AdaBoost algoritmasını kullanarak karar ağaçlarını güçlendirmeyi öğrenme.

Öğrenilen AdaBoost modeli kullanılarak tahminler nasıl yapılır.

Verilerinizi AdaBoost algoritması ile kullanmak için en iyi şekilde nasıl

hazırlayabilirsiniz? Hadi başlayalım.

## 30.1 Boosting Ensemble Yöntemi

Boosting, bir dizi zayıf sınıflandırıcıdan güçlü bir sınıflandırıcı oluşturan genel bir topluluk yöntemidir. Bu, eğitim verilerinden bir model oluşturarak ve ardından ilk modelden gelen hataları düzeltmeye çalışan ikinci bir model oluşturarak yapılır. Eğitim seti mükemmel bir şekilde tahmin edilene veya maksimum sayıda model eklenene kadar modeller eklenir. AdaBoost, ikili sınıflandırma için geliştirilen gerçekten başarılı ilk boosting algoritmasıdır. Boosting'i anlamak için en iyi başlangıç noktasıdır. Modern boosting yöntemleri AdaBoost üzerine inşa edilmiştir, özellikle de stokastik gradyan boosting makineleri.

## 30.2 Verilerinden AdaBoost Modeli Öğrenme

AdaBoost en iyi ikili sınıflandırma problemlerinde karar ağaçlarının performansını artırmak için kullanılır. AdaBoost başlangıçta tekniğin geliştiricileri tarafından AdaBoost.M1 olarak adlandırılmıştır. Son zamanlarda regresyondan ziyade sınıflandırma için kullanıldığından ayrık AdaBoost olarak adlandırılabilir. AdaBoost, herhangi bir makine öğrenimi algoritmasının performansını artırmak için kullanılabilir. En iyi zayıf öğrenicilerle kullanılır.

Bunlar, bir sınıflandırma probleminde rastgele şansın biraz üzerinde doğruluk elde eden modellerdir. AdaBoost ile kullanılan en uygun ve dolayısıyla en yaygın algoritma tek seviyeli karar ağaçlarıdır. Çünkü bu ağaçlar çok kısadır ve sınıflandırma için yalnızca bir karar içerir,

genellikle karar kütükleri olarak adlandırılırlar. Eğitim veri kümesindeki her örnek ağırlıklandırılır. Başlangıç ağırlığı şu şekilde ayarlanır:

$$ağırlık(x_i) = \frac{1}{n} \tag{30.1}$$

Burada $x_i$ $i$'inci eğitim örneğidir ve $n$ eğitim örneklerinin sayısıdır.

## 30.3 Bir Modeli Nasıl Eğitilir

Ağırlıklı örnekler kullanılarak eğitim verileri üzerinde zayıf bir sınıflandırıcı (karar kütüğü) hazırlanır. Yalnızca ikili (iki sınıflı) sınıflandırma problemleri desteklenir, bu nedenle her karar kütüğü bir giriş değişkeni üzerinde bir karar verir ve birinci veya ikinci sınıf değeri için +1.0 veya -1.0 değerini verir. Yanlış sınıflandırma oranı, eğitilen model için hesaplanır. Geleneksel olarak bu şu şekilde hesaplanır:

$$error = \frac{doğru - N}{N} \tag{30.2}$$

Burada *hata* yanlış sınıflandırma oranı, doğru model tarafından doğru tahmin edilen eğitim örneği sayısı ve $N$ toplam eğitim örneği sayısıdır. Örneğin, model 100 eğitim örneğinden 78'ini doğru tahmin ettiyse hata veya yanlış sınıflandırma oranı ol $\frac{78-100}{100}$ veya 0,22. Bu, eğitim örneklerinin ağırlıklandırmasını kullanmak için değiştirilmiştir:
m
ak

$$hata = \frac{\sum_{i=1}^{n} (w_i \times perror)_i}{\sum_{i=1}^{n} w} \tag{30.3}$$

Bu, yanlış sınıflandırma oranının ağırlıklı toplamıdır; burada $w$, $i$ eğitim örneği için ağırlık ve *perror*, $i$ eğitim örneği için yanlış sınıflandırılmışsa 1 ve doğru sınıflandırılmışsa 0 olan tahmin hatasıdır. Örneğin, ağırlıkları 0,01 olan 3 eğitim örneğimiz olsaydı,
0,5 ve 0,2. Tahmin edilen değerler -1, -1 ve -1 ve örneklerdeki gerçek çıktı değişkenleri -1, 1 ve -1 olsaydı, *hata* değerleri 0, 1 ve 0 olurdu. Yanlış sınıflandırma oranı şu şekilde hesaplanırdı:

$$error = \frac{0.01 \times 0 + 0.5 \times 1 + 0.2 \times 0}{0.01 + 0.5 + 0.2}$$
$$hata = 0{,}704 \tag{30.4}$$

Eğitilen model için bir aşama değeri hesaplanır ve bu değer modelin yaptığı tahminler için bir ağırlıklandırma sağlar. Eğitilmiş bir model için aşama değeri aşağıdaki gibi hesaplanır:

$$aşama = ln(\frac{1 - error}{Hata}) \tag{30.5}$$

Burada aşama, modelden gelen tahminleri ağırlıklandırmak için kullanılan aşama değeridir, *ln*() doğal logaritmadır ve hata, model için yanlış sınıflandırma hatasıdır. Aşama ağırlığının etkisi, daha doğru modellerin nihai tahmine daha fazla ağırlığa veya katkıya sahip olmasıdır. Eğitim ağırlıkları, yanlış tahmin edilen örneklere daha fazla ağırlık ve doğru tahmin edilen örneklere daha az ağırlık verilerek güncellenir. Örneğin, bir eğitim örneğinin ağırlığı ($w$) şu şekilde güncellenir:

$$w = w \times e^{sahne \times perror} \tag{30.6}$$

Burada *w* belirli bir eğitim örneği için ağırlık, *e* bir güce yükseltilmiş sayısal sabit Euler sayısı, stage zayıf sınıflandırıcı için yanlış sınıflandırma oranı ve *perror* zayıf sınıflandırıcının eğitim örneği için çıktı değişkenini tahmin ederken yaptığı hata olarak değerlendirilir:

perror = 0 **EĞER** y == p

perror = 1 **EĞER** y != p

Burada *y* eğitim örneği için çıktı değişkeni ve *p* zayıf öğreniciden gelen tahmindir. Bu, eğitim örneği doğru sınıflandırılmışsa ağırlığı değiştirmeme ve zayıf öğrenici örneği yanlış sınıflandırmışsa ağırlığı biraz daha artırma etkisine sahiptir.

# 30.4 AdaBoost Ensemble

Zayıf modeller sırayla eklenir ve ağırlıklı eğitim verileri kullanılarak eğitilir. İşlem, önceden ayarlanmış sayıda zayıf öğrenici oluşturulana (bir kullanıcı parametresi) veya eğitim veri kümesinde daha fazla iyileştirme yapılamayana kadar devam eder. İşlem tamamlandığında, her biri bir aşama değerine sahip zayıf öğrenicilerden oluşan bir havuzunuz olur.

# 30.5 AdaBoost ile Tahmin Yapma

Tahminler, zayıf sınıflandırıcıların ağırlıklı ortalaması hesaplanarak yapılır. Yeni bir girdi örneği için her bir zayıf öğrenici +1.0 veya -1.0 olarak tahmin edilen bir değer hesaplar. Tahmin edilen değerler her bir zayıf öğrenicinin aşama değeri ile ağırlıklandırılır. Topluluk modeli için tahmin, ağırlıklı tahminlerin toplamı olarak alınır. Toplam pozitifse birinci sınıf, negatifse ikinci sınıf tahmin edilir.

Örneğin, 5 zayıf sınıflandırıcı 1.0, 1.0, -1.0, 1.0, -1.0 değerlerini tahmin edebilir. Çoğunluk oylamasına göre, model 1.0 değerini veya birinci sınıfı tahmin edecek gibi görünmektedir. Aynı 5 zayıf sınıflandırıcı sırasıyla 0,2, 0,5, 0,8, 0,2 ve 0,9 aşama değerlerine sahip olabilir. Bu tahminlerin ağırlıklı toplamı hesaplandığında -0,8'lik bir çıktı elde edilir, bu da -1,0'lık bir topluluk tahmini veya ikinci sınıf olur.

# 30.6 Verilerin AdaBoost İçin Hazırlanması

Bu bölümde, verilerinizi AdaBoost için en iyi şekilde hazırlamaya yönelik bazı sezgisel yöntemler listelenmektedir.

**Kaliteli Veri**: Topluluk yöntemi, eğitim verilerindeki yanlış sınıflandırmaları düzeltmeye çalışmaya devam ettiğinden, eğitim verilerinin yüksek kalitede olmasına dikkat etmeniz gerekir.

**Aykırı** değerler: Aykırı değerler, topluluğu gerçekçi olmayan vakaları düzeltmek için çok çalışmaya zorlayacaktır. Bunlar eğitim veri setinden çıkarılabilir.

**Gürültülü Veri**: Gürültülü veriler, özellikle de çıktı değişkenindeki gürültü sorunlu olabilir. Mümkünse, bunları eğitim veri setinizden izole etmeye ve temizlemeye çalışın.

## 30.7 Özet

Bu bölümde makine öğrenimi için Boosting ensemble yöntemini keşfettiniz. Hakkında bilgi edindiniz:

Boosting ve sınıflandırma hatalarını düzeltmek için zayıf öğrenicileri eklemeye devam eden genel bir tekniktir.

İkili sınıflandırma problemleri için ilk başarılı boosting algoritması olarak AdaBoost.

Eğitim örneklerini ve zayıf öğrenicilerin kendilerini ağırlıklandırarak AdaBoost modelini öğrenme.

Zayıf öğrenicilerden gelen tahminleri ağırlıklandırarak AdaBoost ile tahmin etme.

AdaBoost algoritması hakkında daha fazla teorik arka plan için nereye bakmalısınız?

Artık boosting ensemble yöntemi ve AdaBoost makine öğrenimi algoritması hakkında bilgi sahibisiniz. Bir sonraki bölümde AdaBoost algoritmasını sıfırdan nasıl uygulayacağınızı keşfedeceksiniz.

# Bölüm 31

# AdaBoost Eğitimi

AdaBoost, ilk boosting ensemble makine öğrenimi algoritmalarından biridir. Bu bölümde makine öğrenimi için AdaBoost'un adım adım nasıl çalıştığını keşfedeceksiniz. Bu bölümü okuduktan sonra şunları öğreneceksiniz:

Ağırlıklı eğitim verileri kullanılarak eğitim verilerinden karar kütükleri nasıl oluşturulur?

Sınıflandırılması zor örneklere daha fazla dikkat edilmesi için eğitim verilerindeki ağırlıkların nasıl güncelleneceği.

Birbiri ardına 3 modelden oluşan bir dizi nasıl oluşturulur.

Tahmin yapmak için üç zayıf modelli bir AdaBoost modeli nasıl kullanılır. Hadi

başlayalım.

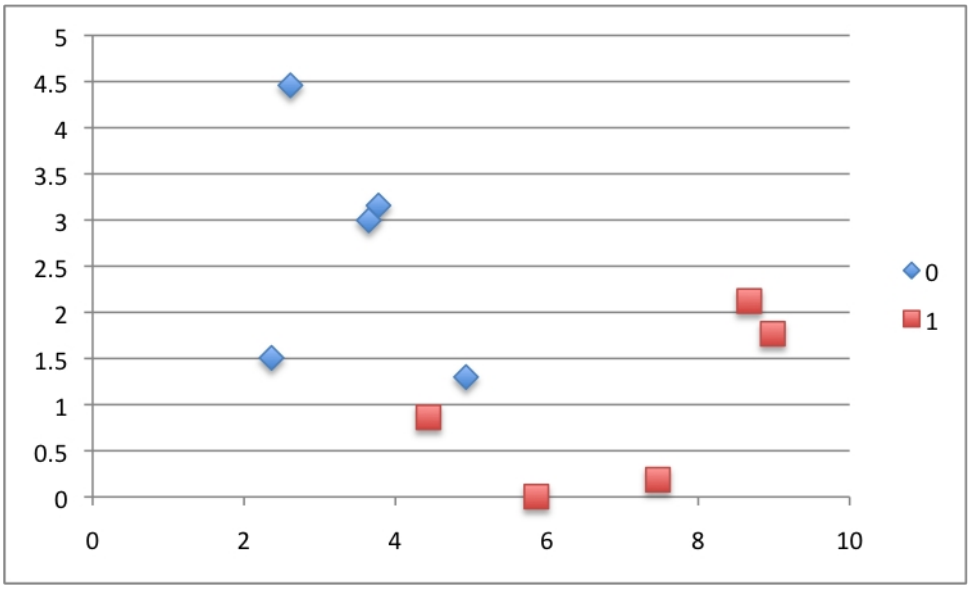
## 31.1 Sınıflandırma Problemi Veri Kümesi

Bu eğitimde iki girdi değişkeni (*X1* ve *X2*) ve bir çıktı değişkeni (*Y* ) olan bir veri kümesi kullanacağız. Giriş değişkenleri Gauss dağılımından çekilen gerçek değerli rastgele sayılardır. Çıktı değişkeninin iki değeri vardır, bu da problemi ikili sınıflandırma problemi yapar. Ham veriler aşağıda listelenmiştir.

```
X1            X2            Y
3.64754035    2.996793259   0
2.612663842   4.459457779   0
2.363359679   1.506982189   0
4.932600453   1.299008795   0
3.776154753   3.157451378   0
8.673960793   2.122873405   1
5.861599451   0.003512817   1
8.984677361   1.768161009   1
7.467380954   0.187045945   1
4.436284412   0.862698005   1
```

Liste 31.1: Öğretici Veri Setini Güçlendirme.

Ham verileri çizdiğinizde, sınıfların net bir şekilde ayrılmadığını görebilirsiniz.




Şekil 31.1: Boosting Öğretici Veri Seti Dağılım grafiği.

## 31.2 AdaBoost Modelini Verilerinden Öğrenin

Bu bölümde eğitim verilerinden bir AdaBoost modeli öğreneceğiz. Bu, birbiri ardına 3 model öğrenmeyi içerecektir, böylece eğitim örneklerine verilen ağırlıklar üzerindeki etkiyi ve iki modelin her birinden gelen tahminlerin nasıl birleştirildiğini gözlemleyebiliriz. AdaBoost, dahili model olarak karar kütüğü (tek düğümlü karar ağaçları) kullanır. Bu karar ağaçları için bölünme noktalarını seçmek üzere CART algoritması veya benzerini kullanmak yerine, bölünme noktalarını manuel olarak seçeceğiz. Yine, sınıflandırma hataları oluşturmak ve ikinci modelin ilk modeli nasıl düzeltebileceğini göstermek için bu bölünme noktaları zayıf bir şekilde seçilecektir ve bu böyle devam edecektir.

## 31.3 Karar Kütüğü: Model #1

İlk model, 3.64754035 değerinde bölünmüş X1 girdi değişkeni için bir karar kütüğü olacaktır.

X1 ≤ 4.932600453 **ISE** SOL

X1 > 4.932600453 **ISE** SAĞA

Bu, bu veriler için zayıf bir ayırma noktasıdır ve bu eğitimde bazı yanlış sınıflandırılmış örnekler oluşturmak için kasıtlı olarak seçilmiştir. İdeal olarak, maliyet fonksiyonu olarak Gini endeksi ile bu veri kümesi için bir ayırma noktası seçmek üzere CART algoritmasını kullanırsınız. Bu ayırma noktasını eğitim verilerine uygularsak, verilerin aşağıdaki iki gruba ayrıldığını görebiliriz:

```
Y   Grup
0   SOL
0   SOL
0   SOL
0   SOL
0   SOL
1   DOĞRU
```

```
1   DOĞRU
1   DOĞRU
1   DOĞRU
1   SOL
```

Liste 31.2: Eğitim Veri Kümesinin Model 1'e Göre Bölünmesi.

Bu grubun bileşimine baktığımızda, sağ grubun ağırlıklı olarak sınıf 1'de ve sol grubun ağırlıklı olarak sınıf 0'da olduğunu görebiliriz. Bu sınıf değerleri her grup için yapılan tahminler olacaktır.

SOL: Sınıf 0

SAĞ: Sınıf 1

Artık eğitilmiş bir karar güdük modelimiz olduğuna göre, bunu eğitim veri kümesi için tahminler yapmak üzere kullanabiliriz. Tahmindeki hata şu şekilde hesaplanabilir:

hata = 0 **EĞER** Tahmin == Y

hata = 1 **EĞER** Tahmin != Y

Karar kütüğünü kullanarak yapılan tahminleri ve bunların hatalarını aşağıdaki tabloda özetleyebiliriz.

```
Y       Tahmin          Hata
0       0               0
0       0               0
0       0               0
0       0               0
0       0               0
1       1               0
1       1               0
1       1               0
1       1               0
1       0               1
```

Liste 31.3: Model 1 ile Eğitim Veri Kümesi Tahminleri.

10 tahminde 0 hata ya da 0,9 veya %90 doğruluk oranı görebiliriz. Fena değil, ancak ideal değil. AdaBoost kullanırken, aşağıdaki denklemi kullanarak yukarıdaki karar kütüğü gibi zayıf bir model için yanlış sınıflandırma oranını hesaplarız:

$$\textit{Yanlış Sınıflandırma Oranı} = \frac{\sum_{i=1}^{n} (w_i \times hata)_i}{\sum_{i=1}^{n} w} \tag{31.1}$$

Eğitim veri kümesindeki her örneğin başlangıç ağırlığı $\frac{1}{N}$ Burada $N$ eğitim örneği, bu durumda 10 yani $\frac{1}{10} = 0.1$.

$$\begin{aligned} ağırlık &= \frac{1}{N} \\ ağırlık &= \frac{1}{10} \\ ağırlık &= 0.1 \end{aligned} \tag{31.2}$$

Bu başlangıç ağırlığını ve yukarıdaki tahmin hatalarını kullanarak, her bir tahmin için ağırlıklı hatayı hesaplayabiliriz.

$$WeightedError = ağırlık \times \ hata \tag{31.3}$$

Sonuçlar aşağıda listelenmiştir.

```
Ağırlı Hata   Ağırlıklı Hata
k
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    0      0
0.1    1      0.1
```

Liste 31.4: Model 1 için Ağırlık Hataları.

Şimdi yanlış sınıflandırma oranını şu şekilde hesaplayabiliriz:

$$\begin{aligned} Yanlış\ Sınıflandırma\ Oranı &= \frac{\Sigma(AğırlıklıHata)}{\Sigma(ağırlık)} \\ Yanlış\ Sınıflandırma\ Oranı &= \frac{0.2}{1.0} \\ Yanlış\ Sınıflandırma\ Oranı &= 0{,}2 \end{aligned} \tag{31.4}$$

Son olarak, bu zayıf modelin aşamasını hesaplamak için yanlış sınıflandırma oranını kullanabiliriz. Aşama, daha sonra bu modeli gerçekten tahminler yapmak için kullandığımızda bu model tarafından yapılan herhangi bir tahmine uygulanan ağırlıktır. Aşama şu şekilde hesaplanır:

$$\begin{aligned} stage &= ln(\frac{1 - Yanlış\ Sınıflandırma\ Oranı}{Yanlış\ Sınıflandırma\ Oranı}) \\ aşama &= ln(\frac{1 - 0.1}{0.1}) \\ aşama &= 2.197224577 \end{aligned} \tag{31.5}$$

Bu sınıflandırıcı tahminler üzerinde çok fazla ağırlığa sahip olacaktır, bu da %90 doğru olduğu için iyidir.

### 31.3.1 Örnek Ağırlıklarını Güncelleme

İkinci bir güçlendirilmiş model hazırlamadan önce örnek ağırlıklarını güncellememiz gerekir. Bu, boosting'in özüdür. Ağırlıklar güncellenir, böylece oluşturulan bir sonraki model, önceki modellerin yanlış yaptığı eğitim örneklerine daha fazla, doğru yaptığı örneklere ise daha az eğitim örneği. Artık aşamayı da biliyoruz. Her eğitim örneği için ağırlıkların güncellenmesi

dikkat eder. Bir eğitim örneğinin ağırlığı şu şekilde güncellenir:

$$ağırlık = ağırlık \times e^{stage \times error}$$

(31.6) Her eğitim örneği için mevcut ağırlığı (0.1) ve her eğitim örneğinde yapılan hataları biliyoruz.

eğitim örneği. Artık aşamayı da biliyoruz. Her eğitim örneği için ağırlıkların güncellenmesi

bu nedenle oldukça basittir. Aşağıda her bir eğitim örneği için güncellenmiş ağırlıklar yer almaktadır. Yalnızca ilk modelin yanlış yaptığı bir örnek için ağırlıkların farklı olduğunu fark edeceksiniz. Aslında daha büyüktür, böylece oluşturulan bir sonraki model bunlara daha fazla dikkat eder.

```
Ağırlık
0,1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.124573094
```

Liste 31.5: Güncellenmiş Örnek Ağırlıkları Eğitim Modeli 1.

## 31.4 Karar Kütüğü: Model #2

Şimdi ağırlıklandırılmış eğitim örneklerinden ikinci zayıf modelimizi oluşturabiliriz. Bu ikinci model de bir karar kütüğü olacaktır, ancak bu kez X2 değişkeninde 2.122873405 değerinde bir bölünme yapacaktır. Yine, bu zayıf doğruluğa sahip olması amaçlanan yapmacık bir modeldir. İdeal olarak, iyi kalitede bir ayırma noktası seçmek için CART algoritması ve Gini endeksi gibi bir şey kullanırsınız. CART algoritmasının örnek ağırlığını dikkate alacağı ve farklı bir karar kütüğüyle sonuçlanmak için daha büyük ağırlığa sahip örneklerde bölmeye odaklanacağı unutulmamalıdır. Aksi takdirde, algoritma modelden modele aynı karar kütüğünü oluşturacak ve önceki modellerden yapılan tahminleri düzeltme şansı olmayacaktır. Bu yeni bölme noktasını kullanarak aynı süreçten geçeceğiz.

X2 ≤ 2.122873405 **ISE** SOL

X2 > 2.122873405 **ISE** SAĞA

Bu da aşağıdaki gruplarla sonuçlanır:

```
Y   Grup
0   DOĞRU
0   DOĞRU
0   SOL
0   SOL
0   DOĞRU
1   SOL
1   SOL
1   SOL
1   SOL
1   SOL
```

Liste 31.6: Eğitim Veri Kümesinin Model 2'ye Göre Bölünmesi.

Her bir grup için kompozisyona bakıldığında, SOL grup 1 sınıf değerine sahiptir ve

SAĞ grup 0 sınıf değerine sahiptir.

SOL: Sınıf 1

SAĞ: Sınıf 0

Bu basit karar ağacını kullanarak, artık her eğitim örneği için bir tahmin ve bu tahminler için hata ve ağırlıklı hatayı hesaplayabiliriz.

| Y | Tahmin | Hata | Ağırlıklı Hata |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0.1 |
| 0 | 1 | 1 | 0.1 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |

Liste 31.7: Model 2 için Tahminler ve Ağırlıklı Hata.

Hızlı bir zarf kontrolü, bu modelin eğitim verilerinde %80 oranında doğru olduğunu göstermektedir. İlk modelden biraz daha kötü. Ayrıca farklı hatalar yapmış ve ilk modelin yanlış tahmin ettiği örnekleri doğru tahmin etmiş gibi görünüyor. Bu sınıflandırıcı için aşama değerini hesaplayabiliriz:

$$\begin{aligned} &\Sigma \\ &\Sigma \; ağırlık = 1.024573094 \\ &\Sigma \\ &\quad WeightedError = 0{,}2 \\ &Yanlış\ Sınıflandırma\ Oranı = \\ &0.1952032521 \\ &aşama = 1.416548424 \end{aligned} \tag{31.7}$$

Henüz işimiz bitmedi. Eğer burada durursak, çok doğru bir AdaBoost modeline sahip olamayız. Aslında hızlı bir kontrol, modelin eğitim verilerinde yalnızca %60'lık bir doğruluğa sahip olacağını göstermektedir. Bir sonraki bölümü yalnızca ilk iki modelden gelen tahminleri birleştirecek şekilde değiştirerek bunu daha sonra kendiniz kontrol edebilirsiniz. Bazı son düzeltmeleri yapmak için bir modele daha ihtiyacımız var.

## 31.5 Karar Kütüğü: Model #3

Yine, yukarıda açıklanan aynı güncelleme prosedürünü kullanarak örnekler için ağırlıkları güncelleyebiliriz. Bu bize aşağıdaki ağırlıkları verir:

| Ağırlık |
|---|
| 0,1 |
| 0.1 |
| 0.11521789 |
| 0.11521789 |
| 0.1 |
| 0.1 |
| 0.1 |
| 0.1 |
| 0.1 |
| 0.124573094 |

Liste 31.8: Model 3'ü Eğitmeden Önceki Ağırlık Değerleri.

Bu model X2 için 0.862698005 adresinde bir ayırma noktası seçecektir.

X2 ≤ 0.862698005 **ISE** SOL

X2 > 0.862698005 **ISE** SAĞA

Bu ayırma noktası veri kümesini aşağıdaki gruplara ayırır:

| Y | Grup |
|---|---|
| 0 | DOĞRU |
| 0 | DOĞRU |
| 0 | DOĞRU |
| 0 | DOĞRU |
| 0 | DOĞRU |
| 1 | DOĞRU |
| 1 | SOL |
| 1 | DOĞRU |
| 1 | SOL |
| 1 | SOL |

Liste 31.9: Eğitim Veri Kümesinin Model 3'e Göre Bölünmesi.

Yine, her bir grup için kompozisyona bakıldığında, SOL grup 1 sınıf değerine ve SAĞ grup 0 sınıf değerine sahiptir.

SOL: Sınıf 1

SAĞ: Sınıf 0

Bu son basit karar ağacını kullanarak, artık her eğitim örneği için bir tahmin ve bu tahminler için hata ve ağırlıklı hatayı hesaplayabiliriz.

| Y | Tahmin | Hata | Ağırlıklı Hata |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0.1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0.1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |

Liste 31.10: Model 3 için Tahminler ve Ağırlıklı Hata.

Bu model de 2 hata yapar (veya %80 doğruluğa sahiptir). Yine, bu sınıflandırıcı için aşama değerini hesaplayabiliriz:

$$\begin{aligned} &\sum(ağırlık) = 1.024573094 \\ &\sum WeightedError = 0{,}2 \\ &Yanlış\ Sınıflandırma\ Oranı = 0.1952032521 \\ &aşama = 1.416548424 \end{aligned} \tag{31.8}$$

Şimdi işimiz bitti. Bu güçlendirilmiş modeller serisini tahminler yapmak için nasıl kullanabileceğimize bakalım.

## 31.6 AdaBoost Modeli ile Tahminler Yapın

Artık AdaBoost modelini kullanarak eğitim veri kümesi için tahminler yapabiliriz. Tek bir sınıflandırıcı için tahminler +1 veya -1'dir ve model için aşama değeri ile ağırlıklandırılır. Örneğin bu problem için kullanacağız:

$$tahmin = aşama \times \text{ (IF (}output == 0\text{) THEN -1 ELSE +1)} \quad (31.9)$$

Yukarıdaki üç modeli kullanarak, eğitim verilerinden alınan girdi değerleri ile ağırlıklı tahminler yapabiliriz. Bu, tahminler yapmak istediğimiz yeni veriler de olabilir.

| X1 | X2 | Model 1 | Model 2 | Model 3 |
|---|---|---|---|---|
| 3.64754035 | 2.996793259 | -2.197224577 | -1.416548424 | -1.45279448 |
| 2.612663842 | 4.459457779 | -2.197224577 | -1.416548424 | -1.45279448 |
| 2.363359679 | 1.506982189 | -2.197224577 | 1.416548424 | -1.45279448 |
| 4.932600453 | 1.299008795 | -2.197224577 | 1.416548424 | -1.45279448 |
| 3.776154753 | 3.157451378 | -2.197224577 | -1.416548424 | -1.45279448 |
| 8.673960793 | 2.122873405 | 2.197224577 | 1.416548424 | -1.45279448 |
| 5.861599451 | 0.003512817 | 2.197224577 | 1.416548424 | 1.45279448 |
| 8.984677361 | 1.768161009 | 2.197224577 | 1.416548424 | -1.45279448 |
| 7.467380954 | 0.187045945 | 2.197224577 | 1.416548424 | 1.45279448 |
| 4.436284412 | 0.862698005 | -2.197224577 | 1.416548424 | 1.45279448 |

Liste 31.11: Her 3 Modelden Tahminler.

Nihai bir sonuç vermek için her modelin tahminlerini toplayabiliriz. Bir sonuç 0'dan küçükse 0 sınıfı tahmin edilir ve bir sonuç 0'dan büyükse 1 sınıfı tahmin edilir. Şimdi AdaBoost modelinden nihai tahminleri hesaplayabiliriz.

| Toplam | Tahmin | Y | Hata | Doğruluk |
|---|---|---|---|---|
| -5.066567482 | 0 | 0 | 0 | 100 |
| -5.066567482 | 0 | 0 | 0 | |
| -2.233470634 | 0 | 0 | 0 | |
| -2.233470634 | 0 | 0 | 0 | |
| -5.066567482 | 0 | 0 | 0 | |
| 2.160978521 | 1 | 1 | 0 | |
| 5.066567482 | 1 | 1 | 0 | |
| 2.160978521 | 1 | 1 | 0 | |
| 5.066567482 | 1 | 1 | 0 | |
| 0.672118327 | 1 | 1 | 0 | |

Liste 31.12: AdaBoost Modelinden Topluluk Tahmini.

Tahminlerin beklenen *Y* değerleriyle mükemmel bir şekilde eşleştiğini görebiliriz. Ya da başka bir şekilde ifade etmek gerekirse, AdaBoost modeli eğitim verilerinde %100 doğruluk elde etmiştir.

## 31.7 Özet

Bu bölümde bir AdaBoost modelini sıfırdan adım adım uygulayarak nasıl çalışacağınızı keşfettiniz. Şunları öğrendiniz:

Ağırlıklı eğitim verileri kullanılarak eğitim verilerinden karar kütükleri nasıl oluşturulur?

Sınıflandırılması zor örneklere daha fazla dikkat edilmesi için eğitim verilerindeki ağırlıkların nasıl güncelleneceği.

Birbiri ardına 3 modelden oluşan bir dizi nasıl oluşturulur.

Tahmin yapmak için üç zayıf modele sahip bir AdaBoost modeli nasıl kullanılır?

Artık AdaBoost makine öğrenimi algoritmasını sıfırdan nasıl uygulayacağınızı biliyorsunuz. Bu, topluluk makine öğrenimi algoritmalarına girişinizi tamamlıyor.

# Bölüm VI

# Sonuçlar

# Bölüm 32

# Ne Kadar Uzağa Geldiniz

*Başardın. Aferin sana.* Bir an durun ve geriye dönüp ne kadar yol kat ettiğinize bakın.

1. Makine öğrenimi algoritmalarına ilgi duyarak başladınız.
2. Tüm denetimli makine öğrenimi algoritmalarının altında yatan prensibi öğrendiniz; bu algoritmalar girdi değişkenlerinden çıktı değişkenlerine bir eşleme fonksiyonu tahmin etmektedir.
3. Parametrik ve parametrik olmayan algoritmalar, denetimli ve denetimsiz algoritmalar ile yanlılık ve varyanstan kaynaklanan hatalar arasındaki farkı keşfettiniz.
4. Doğrusal regresyon, lojistik regresyon ve doğrusal diskriminant analizi dahil olmak üzere doğrusal makine öğrenimi algoritmalarını keşfettiniz ve uyguladınız.
5. Sınıflandırma ve regresyon ağaçları, naif Bayes, k-en yakın komşular, öğrenme vektörü niceleme ve destek vektör makineleri gibi doğrusal olmayan makine öğrenimi algoritmalarını sıfırdan uyguladınız.
6. Son olarak, karar ağaçları ile torbalama ve adaboost ile güçlendirme gibi en popüler iki topluluk algoritmasını keşfettiniz ve uyguladınız.

Bunu hafife almayın. Kısa sürede uzun bir yol kat ettiniz. Makine öğrenimi algoritmalarını bir elektronik tabloda sıfırdan uygulayabilmek gibi önemli ve değerli bir beceri geliştirdiniz. Bir hesap tablosunda saklanacak hiçbir yer yoktur, ya algoritmayı anlarsınız ve çalışır ya da anlamazsınız ve sonuç alamazsınız.

Makine öğrenimi algoritmaları ile yolculuğunuza başlamanıza yardımcı olmama izin verdiğiniz için bir dakikanızı ayırmak ve içtenlikle teşekkür etmek istiyorum. Umarım öğrenmeye devam eder ve makine öğreniminde ustalaşmaya devam ederken eğlenirsiniz.

# Bölüm 33 Daha

# Fazla Yardım

# Almak

Bu, makine öğrenimi algoritmaları ile yolculuğunuzun sadece başlangıcıdır. Yeni algoritmalar üzerinde çalışmaya başladığınızda veya mevcut algoritma bilginizi genişlettiğinizde yardıma ihtiyacınız olabilir. Bu bölüm, makine öğrenimi algoritmaları konusunda bulabileceğiniz en iyi yardım kaynaklarından bazılarına işaret etmektedir.

## 33.1 Makine Öğrenimi Kitaplar

Bu kitap, makine öğrenimi algoritmalarına başlamak için ihtiyacınız olan her şeyi içeriyor, ancak benim gibiyseniz, kitapları seviyorsunuz demektir. Çok sayıda makine öğrenimi kitabı mevcut, ancak aşağıda bir sonraki adım olarak önerdiğim küçük bir seçki var.

**İstatistiksel Öğrenmeye Giriş**. Makine öğrenimi algoritmalarının istatistiksel bir bakış açısıyla mükemmel bir şekilde ele alınması. Bir sonraki adım olarak önerilir.
http://amzn.to/1pgirl0

**Uygulamalı Tahmine Dayalı Modelleme**. Çok sayıda algoritmayı kapsayan tahmine dayalı modellemeye mükemmel bir giriş. Bu kitap, herhangi bir algoritma üzerinde derinlikten ziyade genişlik için daha iyidir.
http://amzn.to/1n5MSsq

**Yapay Zeka: Modern Bir Yaklaşım**. Genel olarak yapay zeka üzerine mükemmel bir kitap, ancak makine öğrenimi ile ilgili bölümler bu kitapta ele alınan algoritmalara mükemmel bir bilgisayar bilimi perspektifi sunuyor.
http://amzn.to/1TGk1rr

## 33.2 Forumlar ve Soru-Cevap Web Siteleri

Soru-cevap siteleri, makine öğrenimi hakkındaki özel teknik sorularınıza yanıt almanın belki de en iyi yoludur. Benzer sorular için arama yapabilir, ortak sorunların çözümlerini öğrenmek için konulara göz atabilir ve kendi teknik sorularınızı sorabilirsiniz. Makine öğrenimi algoritması sorularınız için tavsiye edebileceğim en iyi soru-cevap siteleri şunlardır:

Çapraz Doğrulama: http://stats.stackexchange.com/

Stack Overflow: http://stackoverflow.com/questions/tagged/machine-learning

Veri Bilimi: http://datascience.stackexchange.com/

Reddit Makine Öğrenimi: http://www.reddit.com/r/machinelearning

Quora Makine Öğrenimi: https://www.quora.com/topic/Machine-Learning/top_stories

Bu sitelerdeki arama özelliğini yoğun bir şekilde kullanın. Ayrıca Cross Validated'da belirli bir soruyu görüntülerken sağ taraftaki gezinti çubuğunda yer alan *İlgili* sorular listesine dikkat edin. Bunlar genellikle alakalı ve faydalıdır.

## 33.3 İletişim Yazar

Makine öğrenimi algoritmaları veya bu kitap hakkında herhangi bir sorunuz olursa, lütfen doğrudan benimle iletişime geçin. Yardımcı olmak için elimden geleni yapacağım.

**Jason Brownlee**
Jason@MachineLearningMastery.com

# Errata

Bu bölüm, bu kitabın farklı revizyonları arasındaki değişiklikleri listeler.

## Revizyon 1.0

Bu kitabın ilk baskısı.

## Revizyon 1.1

Biçimlendirme düzeltildi Bölüm 2, sayfa 7-8.