İçindekiler

YAPAY ZEKA

Zeka

Karmaşık bir problemi çözmek için gerekli bilgileri toplayıp birleştirme kabiliyetidir. Karmaşık bir problemi, çözüm arama alanını daraltarak kısa yoldan çözebilme kabiliyetidir.

Hedef: Bir problemi, etkin ve kısa yoldan çözmek

Zekâ'nın sözlük anlamı: İnsanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma yeteneklerinin tümüdür. Ayrıca: Soyutlama, öğrenme ve yeni durumlara uyma gibi yetenekler de zeka kapsamı içindedir.

American Psychological Association Zekâ Tanımı:

> "Bireyler, karmaşık düşünceleri anlama yetenekleri ile etkin bir şekilde çevreye uyum sağlayabilmeleri ile, deneyim kazanarak öğrenmeleri ile, değişik şekillerde akıl yürütmeleri ile, düşünerek engelleri aşabilmeleri ile birbirlerinden ayrılırlar. Bu bireysel farklılıklar güçlü olmakla birlikte tamamıyla yeterli değildir: Herhangi bir kişinin bireysel performansı farklı şartlarda ve farklı sahalarda değişmektedir. Zeka üzerine yapılmakta olan çalışmalar, bu karmaşık olaylar kümesini aydınlatmayı ve düzenlemeyi amaçlamaktadır."

1994 yılında 52 yapay zeka araştırmacısı tarafından imzalanan bir tanım:

> Akıl yürütme, planlama, problem çözme, soyut düşünme, karmaşık düşünceleri anlama, hızlı öğrenme ve deneyimlerle öğrenmeyle birlikte birçok elemandan oluşan çok genel zihinsel yetenek.

Yapay Zekâ

Yapay zekâ; organik olmayan sistemlerdeki zekâdır. İnsan zekâsının bilgisayar tarafından taklit edilmesini sağlamaya yönelik metotlarla ilgilenen çalışma alanıdır.

Yapay zekâ, insan zihninin problem çözme ve karar verme yeteneklerini taklit etmek için bilgisayarlardan ve makinelerden yararlanır.

Yapay zekâ (AI), makinelerin deneyimden öğrenmesini, yeni girdilere uyum sağlamasını ve insan benzeri görevleri gerçekleştirmesini mümkün kılar. Bugün duyduğunuz çoğu AI örneği - satranç oynayan bilgisayarlardan kendi kendine giden arabalara kadar - derin öğrenme ve doğal dil işlemeye dayanmaktadır. Bu teknolojileri kullanarak bilgisayarlar, büyük miktarda veri işleyerek ve verilerdeki kalıpları tanıyarak belirli görevleri yerine getirecek şekilde eğitilebilir.

Yapay zeka asıl teknolojinin adıdır. İngilizcede "machine learning" olarak tanımlanan makine öğrenmesi ise yapay zekayı oluşturan unsurlardan biridir. Yine İngilizcede "deep learning" şeklinde ifade edilen ve dilimize "derin öğrenme" olarak tercüme edilebilen teknoloji ise makine öğrenmesinin unsurlarından biridir. Coğrafi bir örnekle pekiştirmek gerekirse; yapay zeka dünyadır, makine öğrenmesi dünyayı oluşturan ülkelerden biridir, derin öğrenme ise ülkeleri meydana getiren şehirleri temsil etmektedir. Derin öğrenme için "derin makine öğrenmesi" tanımı da kullanılmaktadır.

Yapay Zekanın Amaçları

- İnsanların zor yaptığı işleri yapabilecek sistemler üretmek.
- İnsan beyninin fonksiyonlarını, bilgisayar modelleri yardımıyla anlamaya çalışmak.
- İnsanın bilgi kazanma, öğrenme ve buluş yapma gibi zihinsel yeteneklerini araştırmak.
- Öğrenme metotlarını bilgisayar sistemlerine aktarmak.
- İnsan bilgisayar iletişimini kolaylaştıran kullanıcı arabirimleri geliştirmek.
- Yapay uzman sistemler oluşturmak.
- Yapay zekaya sahip robotlar geliştirmek (İşbirliği)
- Bilgisayarları, bilimsel araştırma ve buluşlarda kullanmak.

Sezgisel Algoritmalar

- Sezgisel ya da buluşsal (heuristic) bir problem çözme tekniğidir.
- Sonucun doğruluğunun kanıtlanabilir olup olmadığını önemsenmez (en iyi sonucu bulacaklarını garanti etmezler).
- Çeşitli alternatif hareketlerden etkili olanlara karar vererek iyiye yakın çözüm yolları elde etmeyi amaçlar.
- Makul bir süre içerisinde bir çözüm elde edeceklerini garanti ederler.
- Genellikle en iyiye yakın olan çözüm yoluna hızlı ve kolay bir şekilde ulaşırlar.

Sezgisel Yöntemlerden Bazıları:

- **Genetik Algoritma (Genetic Algorithm-GA)**
- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization-ACO)**
- **Parçacık Sürü Optimizasyonu (Particle Swarm Optimization-PSO)**
- **Yapay Arı Kolonisi (Artificial Bee Colony-ABC)**
- Diferansiyel Gelişim Algoritması (Differential Evolution Algorithm-DEA)
- **Benzetim Tavlama(Simulated Annealing-SA)**
- Yerçekimi Arama Algoritması (Gravity Search Algorithm-GSA)
- Isı Transferi Arama (Heat transfer search-HTS)
- **Yasak Arama (Tabu Search-TS)**
- Ağırlıklı Süperpozisyon Çekimi (Weighted Superposition Attraction-WSA)
- Orman Optimizasyonu Algoritması (Forest Optimization Algorithm-FOA)
- Kasırga Temelli Optimizasyon Algoritması (Hurricane Based Optimization Algorithm)
- Ağaç-Tohum Algoritması (Tree-Seed Algorithm-TSA)

Sürü Zekası:

Bazen tek başlarına hiçbir iş yapamayan varlıklar, toplu hareket ettiklerinde çok zekice davranışlar sergileyebilmektedir. Bir topluluğa ait bireyler, en iyi bireyin davranışından ya da diğer bireylerin davranışlarından ve kendi deneyimlerinden yararlanarak yorum yapmakta ve bu bilgileri ileride karşılaşacakları problemlerin çözümleri için bir araç olarak kullanmaktadırlar. Örneğin, bir canlı sürüsünü oluşturan bireylerden birisi bir tehlike sezdiğinde bu tehlikeye karşı tepki verir ve bu tepki sürü içinde ilerleyip tüm bireylerin tehlikeye karşı ortak bir davranış sergilemesini sağlar. Canlıların sürü içerisindeki bu hareketleri gözlemlenerek sürü zekâsı tabanlı optimizasyon algoritmaları geliştirilmiştir. Örnek sürü optimizasyon algoritmalarına Kedi Sürüsü Optimizasyonu (KSO) ve Yapay Arı Koloni Algoritması (YAKA) örnek gösterilebilir.

MAKİNE ÖĞRENMESİ

Çok büyük miktarlardaki verinin elle işlenmesi ve analizinin yapılması mümkün değildir. Amaç geçmişteki verileri kullanarak gelecek için tahminlerde bulunmaktır. Bu problemleri çözmek için Makine Öğrenmesi (machine learning) yöntemleri geliştirilmiştir. Makine öğrenmesi yöntemleri, geçmişteki veriyi kullanarak yeni veri için en uygun modeli bulmaya çalışır. Verinin incelenip, içerisinden işe yarayan bilginin çıkarılmasına da Veri Madenciliği (data mining) adı verilir.

> Makine Öğrenimi "**örnekler ve verilerden öğrenilen algoritmaları birleştirir**".
>
> Bu, "**örnek olarak sunulan veri kümelerindeki değerleri tahmin etmeyi**" mümkün kılar.
>
> Sonuç: Sonuçların kalitesi, **öğrenme sistemine verilen verilerin kalitesine** bağlıdır.

Makine Öğrenmesinin öncüsü kabul edilen Arthur Samuel bunu "**bilgisayarlara açıkça programlanmadan öğrenme kabiliyeti kazandıran bir çalışma alanı**" olarak tanımlamıştır.

- **Denetimsiz öğrenme**, sisteme pek çok örnek sunmaktır, ancak bu sefer iyi cevaplar vermeden. Müşterilerin gruplarını benzerliklere göre ayırma, anormallikleri tespit etme (bankacılık sahtekarlığı) veya korelasyonları tespit etme (örneğin iki ürünün bir mağaza rafına yan yana yerleştirmek için) makinenin ne öğrenmesi.
- **Yarı denetimli öğrenme**, sisteme birçok örnek sunmak ve bazıları için doğru cevabı vermektir. Google'ın veya Facebook'un bu hizmetleri barındırdığı fotoğraflarda bulunan kişileri "tanımak" için yeterli.
- **Takviye öğrenme**, bir sistemin fiziksel (yani, bir robot) veya sanal ortamda gelişmesine izin verir. Sistem cezalar ve ödüller ile gelişiyor. Bir robotun tek başına yürümeyi öğrenmesi budur ya da bir video oyunundaki bot gelişir.

Denetimli Öğrenme

Denetimli öğrenme, "etiketlenmiş" verilerle ilgilenir veya öğrenir. Bu, bazı verilerin zaten doğru yanıtla etiketlendiği anlamına gelir.

Sistemi öğrenmek ve doğru cevabı vermek için örnekler sağlar. Bu tür yapay zekâ uygulamaları, video içeriğini tanımlamak, bir evin geçmişe dayanan satış fiyatını tahmin etmek veya tıbbi riskleri tahmin etmek için kullanılır.

Denetimli öğrenmede veri kümesinin ne olduğu ve bu verilerden istenen çıktının ne olması gerektiği bellidir. Makinelerin veriler arasındaki ilişkiyi öğrenmesi beklenmektedir.

Denetimli öğrenme problemleri **regresyon** ve **sınıflandırma** olarak ikiye ayrılır.

Sınıflandırma: Çıktı değişkeni sözel bir veriyse "siyah" ya da "beyaz" veya "evli" ya da "bekar" gibi bir kategori olduğunda bir sınıflandırma problemidir. Bu biraz da istatistikteki sınıflandırmaya da benzemektedir. Buna örnek olarak cinsiyet, kan grubu, öğrenim durumu vs. gibi liste çoğaltılabilir.

Regresyon: Çıkış değişkeni sayısal bir veriyse "para birimi", "ağırlık" , "yaş" gibi bir gerçek değer olduğunda ise regresyon problemidir.

Örnek-1:

(A) Regresyon: Bir firmanın reklam harcamalarının satışlarını nasıl etkilediğini tahmin etmek
(B) Sınıflandırma: Gelen mailleri spam ve spam olmayan e-posta olarak ayırmak

Örnek-2:

(A) Gayrimenkul piyasasındaki evlerin büyüklüğü hakkında veriler verildiğinde, fiyatların önceden tahmin edilmesi. Fiyatı büyüklüğün bir fonksiyonu olarak sürekli bir çıktı olması nedeniyle, bu bir **regresyon** problemidir.
(B) Bunun yerine, evin "öngörülen fiyattan daha fazla veya daha az bir miktarda satılıp satılmadığını" öğrenmek içinse bu örneği bir **sınıflandırma** problemine dönüştürebiliriz. Burada, satış fiyatlarına dayalı evleri iki ayrı kategoriye atarız.

Örnek-3:

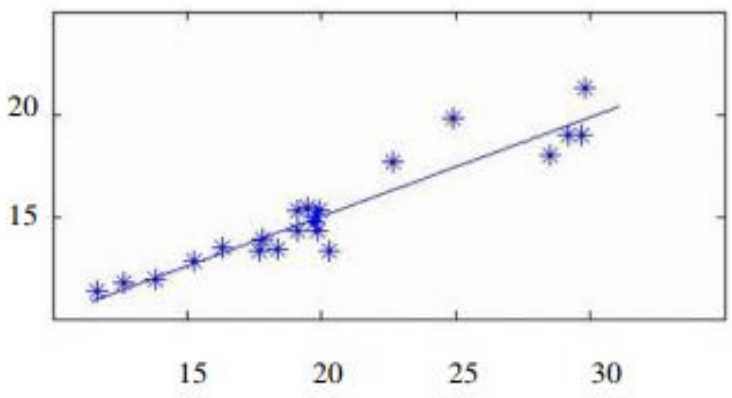
(A) Regresyon – Bir kişinin resmini verildiğinde, verilen resmi temel alarak yaşlarını tahmin etmek
(B) Sınıflandırma – Tümörlü bir hasta göz önüne alındığında, tümörün iyi huylu olup olmadığını öngörmek

Regresyon

Regresyon analizi, iki veya daha fazla değişken arasındaki ilişkiyi tahmin etmek için istatistiksel bir araçtır. Her zaman bir yanıt değişkeni ve bir veya daha fazla tahmin değişkeni vardır.

Bağımlı/tepki değişkenini ve bağımsız/tahmin edici değişkeni kullanarak işletmelerin ve kuruluşların ürünlerinin pazardaki davranışını öğrenmelerine yardımcı olur.

Boya bağlı kilo tahmini - Hava sıcaklığına bağlı deniz suyu sıcaklığı tahmini



x ekseni hava sıcaklığını, y ekseni de deniz suyu sıcaklığını göstermektedir.

Bizden istenen hava sıcaklığına bağlı olarak deniz suyu sıcaklığının tahmin edilmesidir.

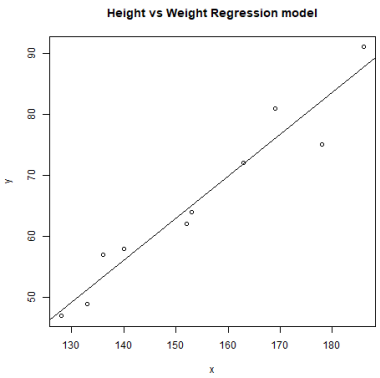
Giriş ile çıkış arasındaki fonksiyonun eğrisi bulunur.

R Linear Regression (Doğrusal Regresyon)

Doğrusal regresyonda, ilişki iki değişken, yani bir yanıt değişkeni ve bir öngörücü değişken arasında tahmin edilir. Doğrusal regresyon, grafikte düz bir çizgi oluşturur.

Syntax:
lm(formula)
Parameter:
formula: *represents the formula on which data has to be fitted*
To know about more optional parameters, use below command in console: help("lm")

```
# Linear Regression
# Height vector
x <- c(153, 169, 140, 186, 128, 136, 178, 163, 152, 133)
# Weight vector
y <- c(64, 81, 58, 91, 47, 57, 75, 72, 62, 49)
# Create a linear regression model
model <- lm(y~x)
# Print regression model
print(model)
# Find the weight of a person
# With height 182
df <- data.frame(x = 182)
res <-  predict(model, df)
cat("\nPredicted value of a person with height = 182")
print(res)
# Output to be present as PNG file
png(file = "linearRegGFG.png")
# Plot
plot(x, y, main = "Height vs Weight Regression model")
abline(lm(y~x))
# Save the file.
dev.off()
```



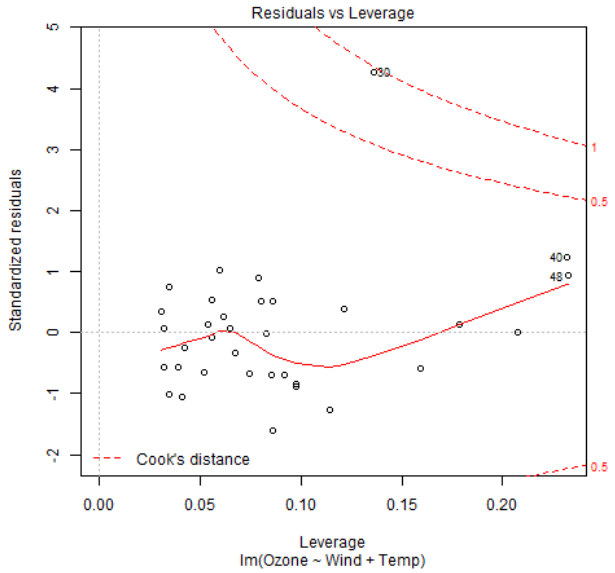
R-Multiple Regression (Çoklu Regresyon)

Çoklu regresyon, modeli oluşturmak için birden fazla tahmin değişkeni kullandığından, doğrusal regresyon modelinin bir uzantısı olan başka bir regresyon analizi tekniği türüdür.

Syntax:
lm(formula, data)
Parameters:
formula: *represents the formula on which data has to be fitted*
data: *represents dataframe on which formula has to be applied*

R temel paketinde bulunan hava kalitesi veri setinin çoklu regresyon modelini oluşturalım ve modeli grafik üzerinde çizelim.

```
# Multiple Linear Regression
# Using airquality dataset
# Using airquality dataset
input <- airquality[1:50,c("Ozone", "Wind", "Temp")]
# Create regression model
model <- lm(Ozone~Wind + Temp, data = input)
# Print the regression model
cat("Regression model:\n")
print(model)
png(file="multipleRegression.png")
#Plot
plot(model)
#Save the File
dev.off()
```



R-Logistic Regression (Lojistik Regresyon)

Regresyon metodudur fakat sınıflandırma işlemi gerçekleştirir. Lojistik Regresyon, değeri bir aralıkla tahmin eder. Kategorik bir değişkenin değerini tahmin eden bir regresyon algoritmasıdır. Yalnızca iki olası değer alabilen bir değişkenin değerini bulur (örn: başarılı veya başarısız). Lojistik regresyon, kategorik bir bağımlı değişken ile birkaç bağımsız değişken arasındaki ilişkiyi bulur. Algoritma yalnızca bir nesne için bir sınıf bulmakla kalmaz, aynı zamanda o nesnenin belirli bir sınıfta olması için gerekçe ve sebep verir.

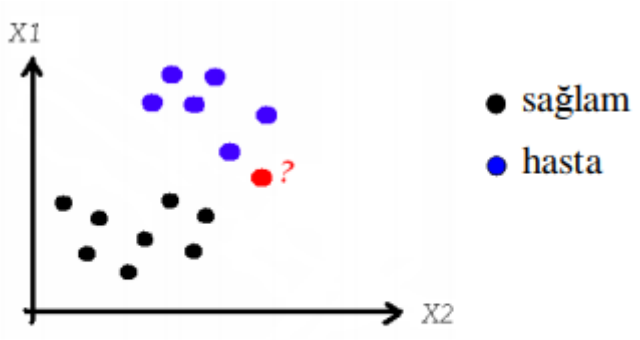
Örneğin, E-posta istenmeyen postadır veya istenmeyen posta değildir, kazanan veya kaybeden, erkek veya kadın vb.

> ***Syntax:****glm(formula, data, family)*
> ***Parameters:***
> ***formula:*** *represents a formula on the basis of which model has to be fitted*
> ***data:*** *represents dataframe on which formula has to be applied*
> ***family:*** *represents the type of function to be used. "binomial" for logistic regression*

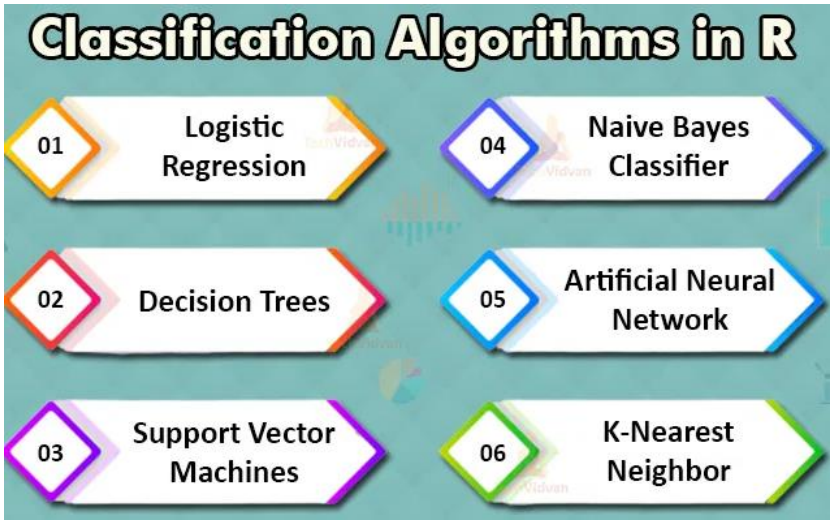
```
# Logistic Regression
# Using mtcars dataset    wt: Weight (1000 lbs)  vs: Engine (0 = V-shaped, 1 = straight)
# To create the logistic model
summary(mtcars)
model <- glm(formula = vs ~ wt, family = binomial, data = mtcars)

# Creating a range of wt values
x <- seq(min(mtcars$wt),
         max(mtcars$wt),
         0.01)

# Predict using weight
y <- predict(model, list(wt = x),
             type = "response")

# Print model
print(model)

# Output to be present as PNG file
png(file = "LogisticRegression.png")

# Plot
plot(mtcars$wt, mtcars$vs, pch = 16,
     xlab = "Weight", ylab = "VS")
lines(x, y)

# Saving the file
dev.off()
```

Sınıflandırma

Geçmiş bilgileri hangi sınıftan olduğu biliniyorsa, yeni gelen verinin hangi sınıfa dahil olacağının bulunmasıdır.



Kırmızı hangi sınıfa dahildir ?



Sınıflandırma, özelliklerine dayalı olarak bir veri nesnesinin kategorik bir etiketini tahmin etme sürecidir. Sınıflandırmada, belirli bir etikete veya kategoriye karşılık gelen tanımlayıcıları veya sınır koşulları bulunur. Daha sonra tanımlayıcıları kullanarak çeşitli bilinmeyen nesneleri bu kategorilere yerleştirmeye çalışılır. Bunun bir örneği, saflığına ve mineral içeriğine bağlı olarak suyun türünü (maden, musluk, akıllı vb.) tahmin etmek olabilir.

Kümeleme ve Sınıflandırma Arasındaki Fark

Kümelemede benzer nesneleri birlikte gruplandırmaya çalışılır. Bir gruptaki nesnelerin o kümedeki diğer nesnelere benzer olması ve farklı gruplardaki hiçbir nesnenin birbirine benzer olmamasıdır.

Sınıflandırmada, bir hedef sınıfı tahmin etmeye çalışılır. Olası sınıflar ve tüm sınıfların tanımlayıcı özellikleri bilinmektedir. Algoritmanın, bir veri nesnesinin hangi sınıfa ait olduğunu tanımlaması gerekir.

Temel Sınıflandırma Terminolojisi

1. **Classifier:** Sınıflandırıcı, girdi verilerini çıktı kategorilerine sınıflandıran bir algoritmadır.
2. **Classification model:** Sınıflandırma modeli, veri nesnelerini çeşitli kategorilere ayırmak için bir sınıflandırıcı kullanan bir modeldir.
3. **Feature:** Bir özellik, bir veri nesnesinin ölçülebilir bir özelliğidir.
4. **Binary classification:** İkili sınıflandırma, iki olası çıktı kategorisine sahip bir sınıflandırmadır.
5. **Multi-class classification:** Çok sınıflı sınıflandırma, ikiden fazla olası çıktı kategorisine sahip bir sınıflandırmadır.
6. **Multi-label classification:** Çoklu etiket sınıflandırması, bir veri nesnesine birden çok etiket veya çıktı sınıfı atanabildiği bir sınıflandırmadır.

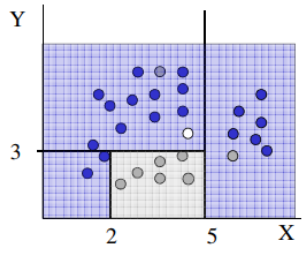
Decision Tree Classifier (Karar Ağaçları)

Karar ağaçları, bir ağaç şeklinde bir dizi karar ve seçeneği temsil eder. Nesnenin hangi sınıfta olduğuna karar vermek için bir nesnenin özelliklerini kullanırlar. Karar ağaçları ikili veya çok sınıflı sınıflandırıcılar olabilir. Karar ağaçları, son bir karar verilinceye kadar nesneleri tekrar tekrar bölmek için kuralları kullandıklarından, böl ve yönet algoritmalarına bir örnektir.

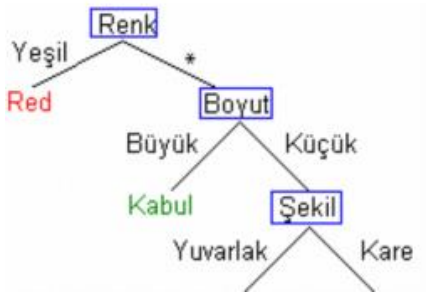
Temelde seçimleri temsil eden bir grafiktir. Grafikteki düğümler veya köşeler bir olayı temsil eder ve grafiğin kenarları karar koşullarını temsil eder. Yaygın olarak Makine Öğrenimi ve Veri Madenciliği uygulamalarında kullanılır.

E-postanın Spam olup olmadığının sınıflandırması, bir tümörün kanserli olup olmadığının sınıflandırılması gibi. Genellikle, eğitim veri kümesi olarak da adlandırılan not edilmiş verilerle bir model oluşturulur. Ardından, modeli doğrulamak ve geliştirmek için bir dizi doğrulama verisi kullanılır.

Böl ve yönet stratejisini kullanır. Peki nasıl bölünecek?



if X > 5 then blue
else if Y > 3 then blue
else if X > 2 then grey
else blue



Ürettikleri kurallar anlaşılır. Karar düğümleri ve yapraklardan oluşan hiyerarşik bir yapıdadır.

Karar Ağaçları Oluşturma

- Tüm veri kümesiyle başlanır.
- Bir özelliğin bir değerine göre veri kümesi iki alt kümeye bölünür. Bölmede kullanılan özellikler ve değerler karar düğüme yerleştirilir.
- Her alt küme için aynı prosedür, her alt kümede sadece tek bir sınıfa ait örnekler kalıncaya kadar uygulanır.

Karar Düğümlerini Bulma

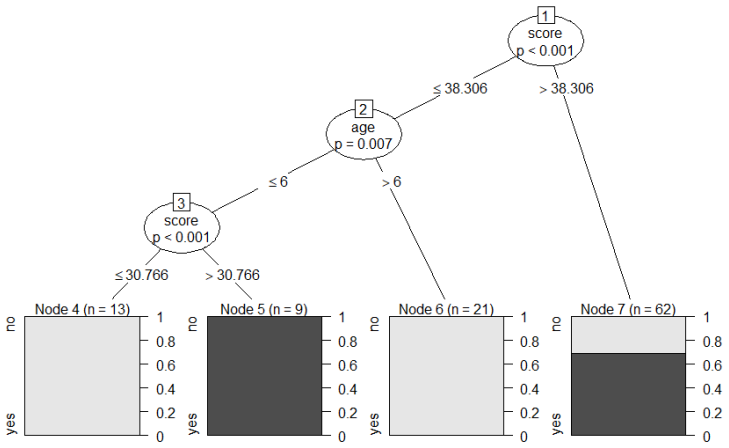
- Karar düğümlerinde yer alan özelliğin ve eşik değerinin belirlenmesinde genel olarak entropi kavramı kullanılır.
- Eğitim verisi her bir özelliğin her bir değeri için ikiye bölünür. Oluşan iki alt kümenin entropileri toplanır. En düşük entropi toplamına sahip olan özellik ve değeri karar düğümüne yerleştirilir.

Karar Ağaçlarıyla Sınıflandırma

- En tepedeki kök karar düğümünden başla.
- Bir yaprağa gelinceye kadar karar düğümlerindeki yönlendirmelere göre dallarda ilerle (Karar düğümlerinde tek bir özelliğin adı ve bir eşik değeri yer alır. O düğüme gelen verinin hangi dala gideceğine verinin o düğümdeki özelliğinin eşik değerinden büyük ya da küçük olmasına göre karar verilir).
- Verinin sınıfı, yaprağın temsil ettiği sınıf olarak belirle.

R- Desicion Tree Örneği

```
# Load the party package. It will automatically load other
# dependent packages.
install.packages("party")
library(party)
input.dat <- readingSkills[c(1:105), ]
# Give the chart file a name.
png(file = "decision_tree.png")
# Create the tree.
output.tree <- ctree(nativeSpeaker ~ age + shoeSize + score, data = input.dat)
# Plot the tree.
plot(output.tree)
# Save the file.
dev.off()
```



Naive Bayes Classifier

Naive Bayes sınıflandırıcısı, Bayes teoremine dayalı bir sınıflandırma algoritmasıdır. Bir veri nesnesinin tüm özelliklerini birbirinden bağımsız olarak kabul eder. Büyük veri kümeleri için çok hızlı ve kullanışlıdırlar. Çok az eğitimle çok doğru sonuçlar elde ederler.

Naive Bayes sınıflandırması, olasılık yaklaşımını kullanan genel bir sınıflandırma yöntemidir, dolayısıyla özellikler arasında bağımsızlık varsayımıyla Bayes teoremine dayalı olasılıksal bir yaklaşım olarak da bilinir.

Model, predict() işleviyle tahminler yapmak için eğitim veri kümesi üzerinde eğitilir.

Genellikle duygusal analizlerde kullanılır.

R- Naive Bayes Örneği

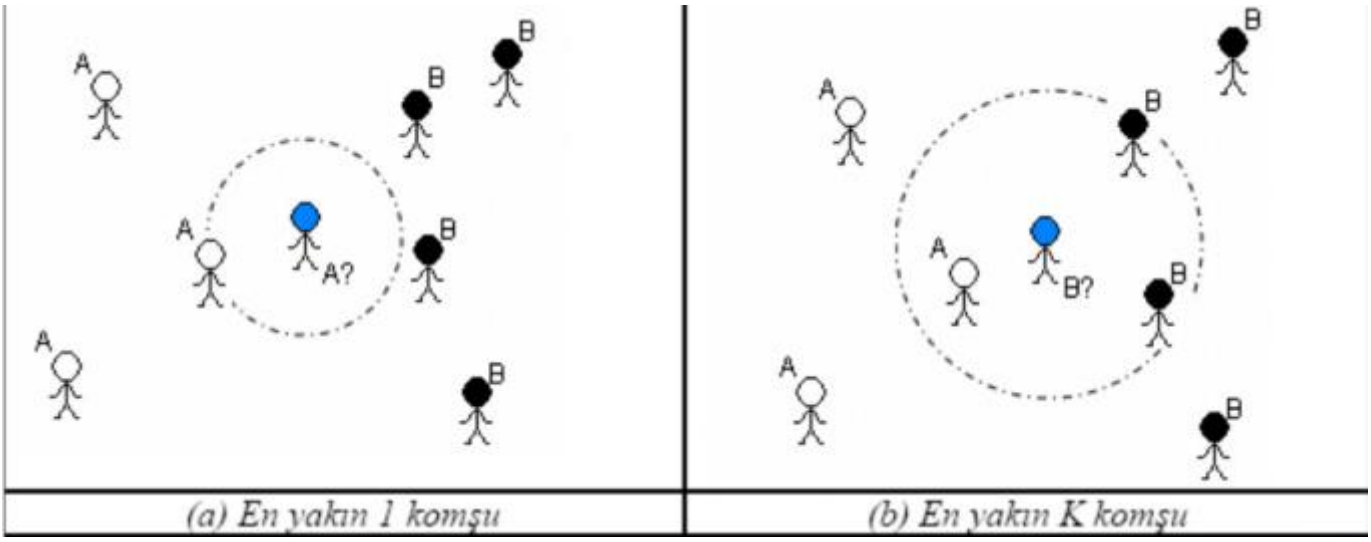
```
library(caret)
## Warning: package 'caret' was built under R version 3.4.3
set.seed(7267166)
trainIndex = createDataPartition(mydata$prog, p = 0.7)$Resample1
train = mydata[trainIndex, ]
test = mydata[-trainIndex, ]

## check the balance
print(table(mydata$prog))
##
## academic    general vocational
## 105         45          50
print(table(train$prog))
```

K-Nearest Neighbor Classifier (K-En Yakın Komşuluk - KNN)

K-en yakın komşu tembel bir öğrenme algoritmasıdır. Eğitim kümesindeki tüm nesneleri n boyutlu bir uzayda eşler ve saklar. Henüz etiketlenmemiş veya sınıflandırılmamış diğer nesneleri etiketlemek için etiketlenmiş nesneleri kullanır. Yeni bir nesneyi etiketlemek için en yakın k komşusuna bakar. Daha sonra bir sayım yapılır ve komşuların çoğunluğu tarafından taşınan etiket, etiketlenmemiş nesneye atanır. Bu algoritma gürültülü veriler için çok sağlamdır ve büyük veri kümeleri için de iyi çalışır. Bununla birlikte, diğer sınıflandırma tekniklerinden daha hesaplamalı olarak ağırdır. k-NN sınıflandırmasında çıktı bir sınıf üyeliğidir.

K-En Yakın Komşu / K-NN algoritması, eğiticili ve örnek tabanlı (instance based) bir sınıflandırma algoritmasıdır. Bu tip algoritmalarda eğitim işlemi yapılmaz. Test edilecek örnek, eğitim kümesindeki her bir örnek ile bire bir işleme alınır.

(Bana arkadaşını söyle, sana kim olduğunu söyleyeyim)



Bir test örneğinin sınıfı belirlenirken eğitim kümesinde o örneğe en yakın K adet örnek seçilir. Seçilen örnekler içerisinde en çok örneği bulunan sınıf, test örneğinin sınıfı olarak belirlenir.

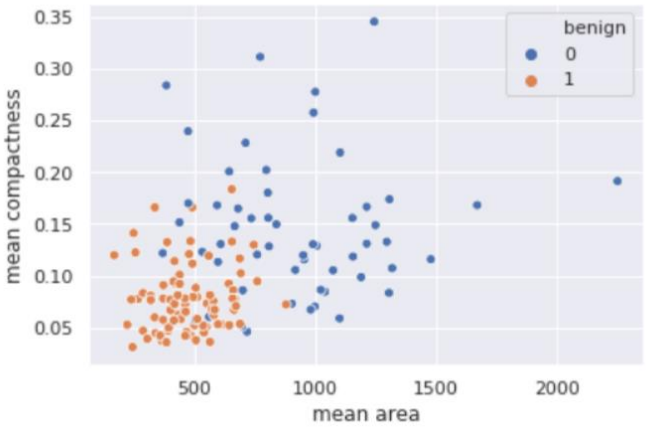
$$y(x_q) = \arg\max_{t \in C} \sum_{j=1}^{k} \delta(x_j, c_t)$$

Örnekler arasındaki uzaklık hesaplanırken eulidean distance kullanılır.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

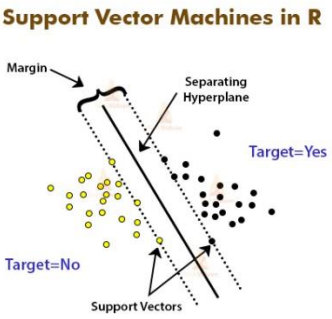
```
# Write Python3 code here
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set()
breast_cancer = load_breast_cancer()
X = pd.DataFrame(breast_cancer.data, columns = breast_cancer.feature_names)
```

```
X = X[['mean area', 'mean compactness']]
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
y = pd.get_dummies(y, drop_first = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 1)
sns.scatterplot(
    x ='mean area',
    y ='mean compactness',
    hue ='benign',
    data = X_test.join(y_test, how ='outer')
)
```



Support Vector Machine (Destek Vektör Makineleri - SVM)

Bir destek vektör makinesi, veri nesnelerini uzaydaki noktalar olarak temsil eder. Ardından, alanı hedef çıktı sınıflarına göre bölebilen bir işlev tasarlar. SVM, uzaydaki nesneleri çizmek ve alanı bölen fonksiyona ince ayar yapmak için eğitim setini kullanır. İşlev sona erdiğinde, nesneleri hangi sınıfa ait olduklarına bağlı olarak uzayın farklı bölümlerine yerleştirir. SVM'ler çok hafiftir ve yüksek boyutlu alanlarda oldukça verimlidir.



Destek vektör makinesi (SVM), iki grup sınıflandırma sorunları için sınıflandırma algoritmalarını kullanan denetimli bir ikili makine öğrenme algoritmasıdır. Metin sınıflandırma problemlerinde ağırlıklı olarak SVM kullanılmaktadır. Görünmeyen verileri sınıflandırır. Naive Bayes'ten daha yaygın olarak kullanılır. SVM genellikle sınırlı miktarda veriyle çok iyi performans gösteren hızlı ve güvenilir bir sınıflandırma algoritmasıdır.

SVM'lerin, genleri sınıflandırmak için Biyoinformatik gibi çeşitli alanlarda bir takım uygulamaları vardır.

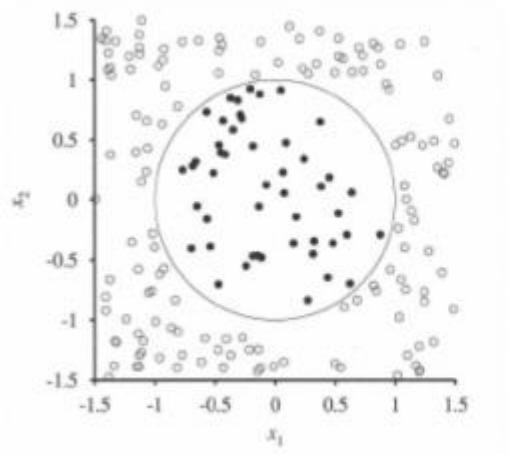
Sınıfları birbirinden ayıran özel bir çizginin (hyperplane) bulunmasını amaçlar.

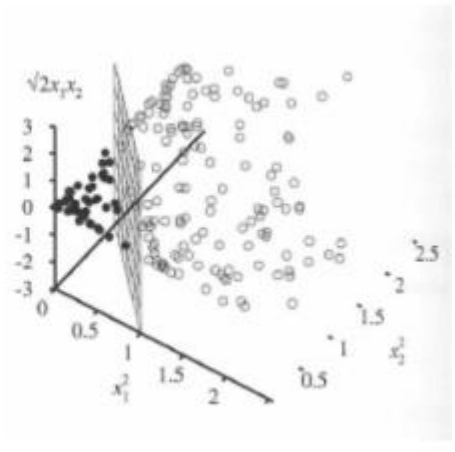
• SVM, her iki sınıfa da en uzak olan hyperplane bulmayı amaçlar.

• Eğitim verileri kullanılarak hyperplane bulunduktan sonra, test verileri sınırın hangi tarafında kalmışsa o sınıfa dâhil edilir.

• Lineer olarak ayrılamayan örneklerde veriler daha yüksek boyutlu başka bir uzaya taşınır ve sınıflandırma o uzayda yapılır.

• Soldaki şekilde örnekler lineer olarak ayrılamaz iken, sağdaki şekilde üç boyutlu uzayda (x12, x22, sqrt(2x1x2)) ayrılabilmektedir.

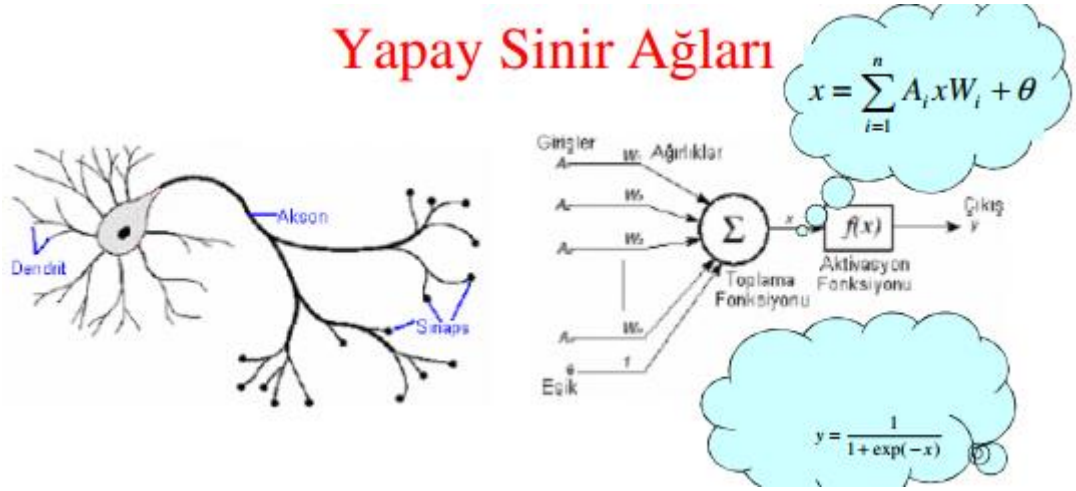




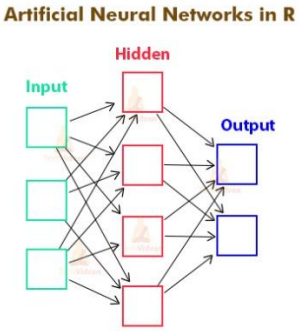
Yapay Sinir Ağları

Canlılardaki sinir hücreleri ve ağları modellenerek yapay sinir ağları oluşturulmuştur.

Gerçek sinir hücreleri, dentritlerden gelen sinyaller belirli bir eşik değerinin üzerine çıktığında akson'lar yardımıyla komşu hücrelere iletilir. Yapay hücrelerde de bu modellenir. Sinyal girişleri (Ai) ve bunları toplayan bir birim giriş sinyallerinin (Ai) ağırlıkları ile (Wi) çarpımlarını toplayan ve bu toplama eşik değerini de ekleyip bir aktivasyon fonksiyonundan geçirerek çıkış elde eder.

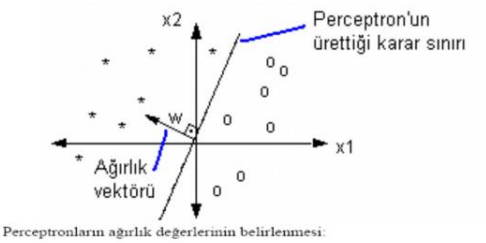


Yapay sinir ağları, çeşitli düğümleri birbirine bağlayan bağlantıların veya nöronların koleksiyonlarıdır. Bir sinir ağındaki iki düğüm arasındaki her bağlantı, bu iki düğüm arasındaki bir ilişkiyi temsil eder. Bu nöronlar, bir yapay sinir ağında katmanlar halinde düzenlenir. Her düğüm, girdiye uyguladığı ve ardından çıktıyı bir sonraki katmana ileten doğrusal olmayan bir işlev içerir. Bunlar, nihai bir sonuç elde edilene kadar her katmanın çıktıyı bir sonraki katmana ilettiği anlamına gelen ileri beslemeli ağlardır. Modelin eğitimi sırasında farklı katmanlara, bağlantılara ve düğümlere farklı ağırlıklar atanır. Bu ağırlıklar, hangi değerlerin ve çıktıların daha çok ve ne kadar tercih edileceğini söyler. Sinir ağları, gürültülü verilerle uğraşırken çok iyidir.



Yapay sinir ağları (YSA) çalışması, kısmen biyolojik öğrenme sistemlerinin beyinlerdeki birbirine bağlı nöronların çok karmaşık ağlarından oluştuğu gözleminden ilham almıştır. Genel olarak, YSA'lar, her birimin bir dizi gerçek değerli girdi aldığı ve tek bir gerçek değerli çıktı ürettiği, birbirine yoğun şekilde bağlı basit birimlerden oluşur.

Tek yapay sinir hücresine perceptron denir.



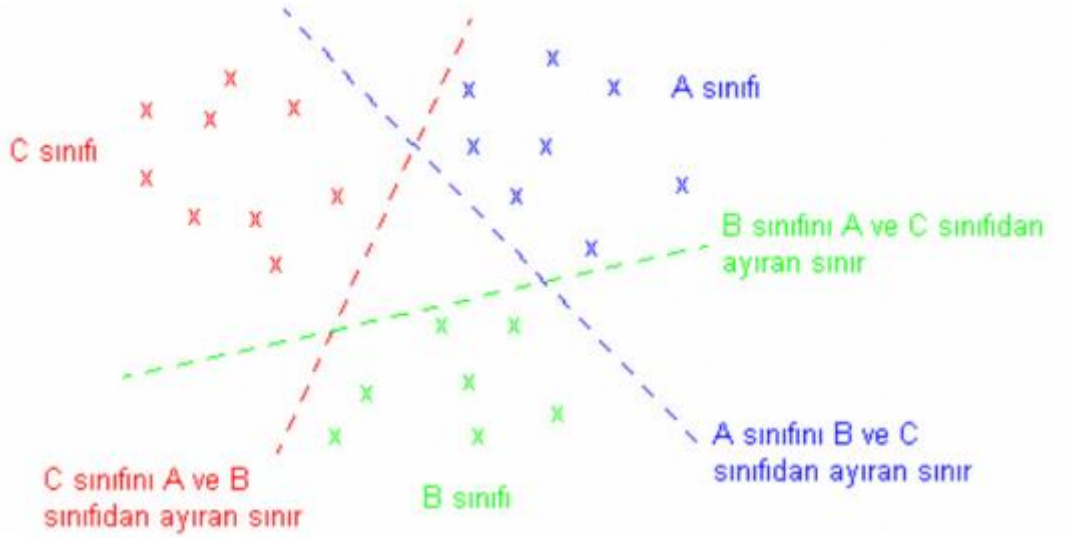
Perceptronların ağırlık değerlerinin belirlenmesi:

1. Ağırlıklara rasgele ilk değerler atanır.
2. (0-1) arasında öğrenme katsayısı (μ) seçilir.
3. Ağırlıklar değiştiği sürece Her bir eğitim örneği için (x, t):

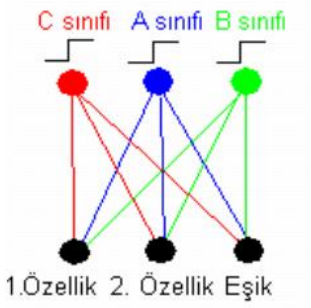
(x→ örneğin değerlerini; t→ sınıfını, w→ ağırlıkları gösterir)

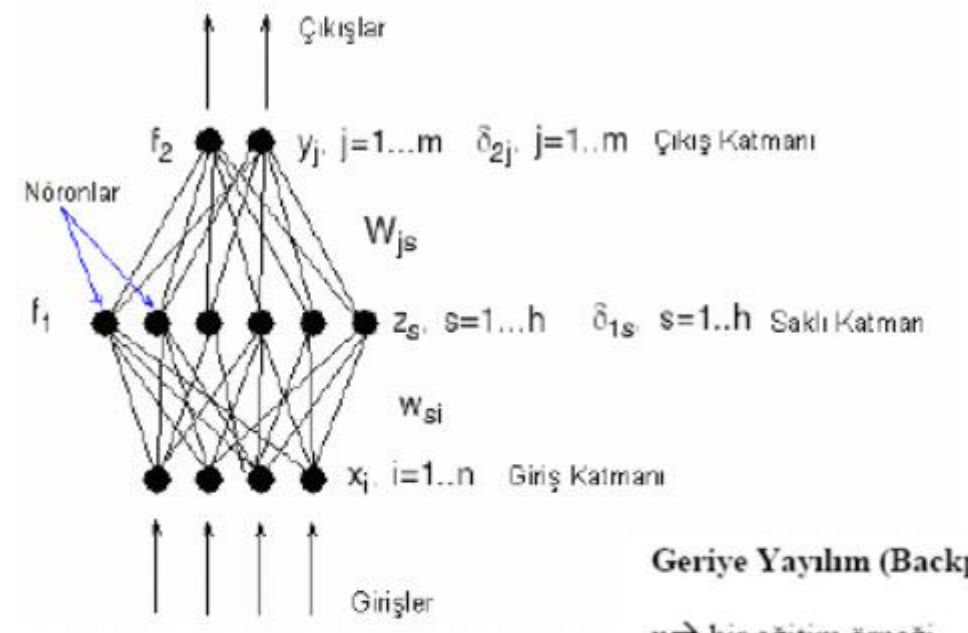
- Çıkışı hesapla. y = f(w*x+esik)
- Çıkışla (perceptronun cevabıyla) gerçek sınıf aynı ise (y = t) ağırlıkları değiştirme
- Farklılarsa (y != t), ağırlıkları w(yeni) = w(eski) + μ (t - y) x güncelle.

İkiden fazla sınıfı birbirinden ayırmak için perceptron katmanı oluşturmak gerekir. Şekilde 3 sınıftan oluşan bir veri kümesi ve bu veriyi sınıflandıran percepton katmanı görülmektedir. Herbir sınıfı diğer sınıflardan ayırt edebilmek için perceptron kullanılmıştır.



Doğrusal olmayan karar sınırları üretebilmek için çok katmanlı perceptronlar kullanılır. Çok katmanlı perceptronlar genelikle geriye yayılım algoritması ile eğitilirler.





Geriye Yayılım (Backpropagation) algoritması

x→ bir eğitim örneği
n→ örneklerin boyutu
h→ saklı katmandaki nöron sayısı
m→ çıkış katmanındaki nöron sayısı
f1→ saklı katmandaki aktivasyon fonksiyonu
f2→ çıkış katmanındaki aktivasyon fonksiyonu
z→ saklı katmanın çıkışları
y→ ağın çıkışları

Eğitim setindeki her bir örnek için aşağıdaki 3 adımın tekrarlanmasına bir çevrim (epoch) adı verilir. Sistemin eğitimine önceden belirlenmiş bir hata değerine ulaşılıncaya kadar ya da maksimum çevrim sayısına erişilinceye kadar devam edilir.

1.Adım: İleri Yayılım

Her saklı nöron için net_i ve y_i hesaplanır, i=1,..., h:

$$net_i = \sum_{r=1}^{n} w_{ri} x_r \qquad z_i = f_1(net_i)$$

Her çıkış nöronu için net_j ve y_j hesaplanır, j=1,...,m:

$$net_j = \sum_{i=1}^{h} w_{ij} z_i \qquad y_j = f_2(net_j)$$

Step 2: Geri Yayılım

Her çıkış nöronu için hata hesaplanır, j=1,...,m:

$$\delta_{2j} = (t_j - y_j) f'_2(net_j)$$

Her saklı nöron için hata hesaplanır, i=1,...,h:

$$\delta_{1i} = f'_1(net_i) \sum_{j=1}^{m} W_{ij} \delta_{2j}$$

Step 3: Ağırlıklar güncellenir:

$$W_{ij}(yeni) = W_{ij}(eski) - \mu \delta_{2j} z_i$$

$$w_{ri}(yeni) = w_{ri}(eski) - \mu \delta_{1i} x_r$$

CNN Kullanım Alanları

- Her yeni teknoloji bir öncekinden yardıma ihtiyaç duyar, yani öncekilerden gelen veriler ve bu veriler analiz edilir, böylece her artı ve eksi doğru şekilde çalışılmalıdır. Bütün bunlar ancak sinir ağı yardımıyla mümkündür.
- Sinir ağı, Hayvan davranışı, yırtıcı/avcı ilişkileri ve popülasyon döngüleri araştırmaları için uygundur.
- Neural Network, at yarışlarında, spor müsabakalarında ve en önemlisi borsada uygulamalarında kullanılabilir.
- Girdi olarak büyük bir suç detayı verisi ve çıktı olarak ortaya çıkan cümleler kullanılarak herhangi bir suç için doğru kararı tahmin etmek için kullanılabilir.
- Veri madenciliği adı verilen verinin analizi ve hangi veride hatanın (peerlerden ayrılan dosyalar) olduğu belirlenerek, yapay sinir ağları üzerinden temizleme ve doğrulama yapılabilir.
- Sinir Ağı, sonar, radar, sismik ve manyetik aletlerden elde ettiğimiz eko kalıpları yardımıyla hedefleri tahmin etmek için kullanılabilir.
- Tıbbi Araştırmalarda geniş bir uygulama alanına sahiptir.
- Geçmiş kayıtları analiz ederek kredi kartları, sigorta veya vergilerle ilgili Dolandırıcılık Tespiti için kullanılabilir.

Single - Multi Layered Neural Network

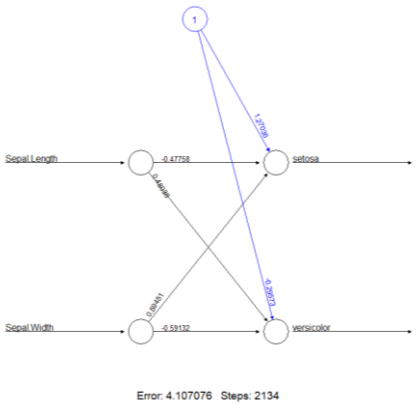
Tek Katmanlı Sinir Ağı: Tek katmanlı bir sinir ağı, giriş ve çıkış katmanını içerir. Giriş katmanı giriş sinyallerini alır ve çıkış katmanı buna göre çıkış sinyallerini üretir. Genellikle perceptron olarak adlandırılan tek katmanlı bir sinir ağı, giriş ve çıkış katmanlarından oluşan bir tür ileri beslemeli sinir ağıdır. Sağlanan girdiler çok boyutludur. Ağırlıkların ve girdilerin toplamı her düğümde hesaplanır. Giriş katmanı, sinyalleri çıkış katmanına iletir. Çıktı katmanı hesaplamaları gerçekleştirir. Perceptron yalnızca doğrusal bir işlevi öğrenebilir ve daha az eğitim çıktısı gerektirir. Çıktı bir veya iki değerde (0 veya 1) temsil edilebilir.

Syntax:
neuralnet(formula, data, hidden)

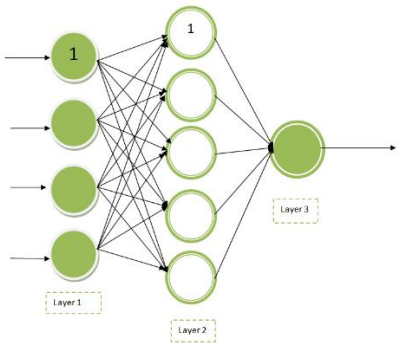
Parameters:
formula: *represents formula on which model has to be fitted*
data: *represents dataframe*
hidden: *represents number of neurons in hidden layers*
To know about more optional parameters of the function, use below command in console: help("neuralnet")

Bu örnekte, sepal uzunluk ve sepal genişlik temelinde setosa ve versicolor iris bitki türlerinin tek katmanlı sinir ağını veya algılayıcısını oluşturalım.

```
# Install the required package
install.packages("neuralnet")

# Load the package
library(neuralnet)

# Load dataset
df <- iris[1:100, ]


#Fitting neural network

nn = neuralnet(Species ~ Sepal.Length
               + Sepal.Width, data = df,
               hidden = 0, linear.output = TRUE)

# Output to be present as PNG file
png(file = "neuralNetworkGFG.png")

# Plot
plot(nn)

# Saving the file
dev.off()
```



Çok Katmanlı Sinir Ağı: Çok katmanlı sinir ağı, girdi, çıktı ve bir veya birden fazla gizli katman içerir. Gizli katmanlar, girdiyi çıktı katmanına yönlendirmeden önce ara hesaplamalar yapar. Doğru olması için, tam bağlantılı(fully connected) Çok Katmanlı Sinir Ağı, Çok Katmanlı Algılayıcı olarak bilinir. Sinir Ağı, birden çok yapay nöron veya düğüm katmanından oluşur.



Burada "1" olarak işaretlenen düğümler, önyargı birimleri olarak bilinir. Katman 1, giriş katmanıdır, Katman 2, gizli katmandır ve Katman 3, çıktı katmanıdır. Yukarıdaki diyagramın 3 giriş birimi (önyargı biriminden ayrılarak), 1 çıkış birimi ve 4 gizli birimi olduğu söylenebilir.

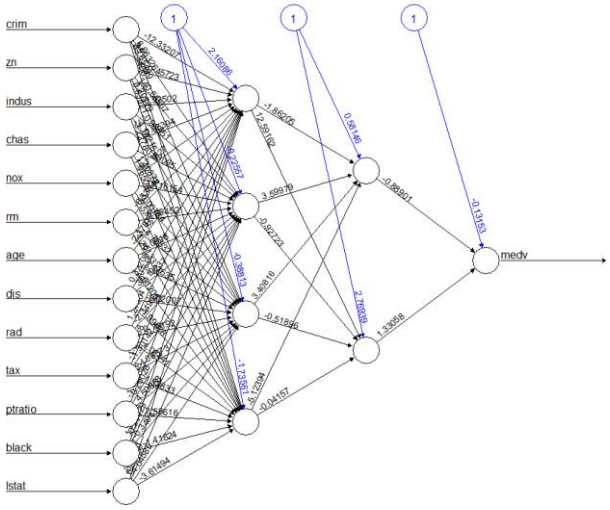
Çok Katmanlı Sinir Ağı, İleri Beslemeli Sinir Ağının tipik bir örneğidir. Nöron sayısı ve katman sayısı, Yapay Sinir Ağlarının ayarlanması gereken hiperparametrelerinden oluşur. Hiperparametreler için ideal değerleri bulmak için bazı çapraz doğrulama teknikleri kullanılmalıdır. Geri Yayılım tekniği kullanılarak ağırlık ayarlama eğitimi yapılır.

R-Örnek Uygulama

Boston veri seti tipik olarak Boston'un kenar mahallelerindeki veya banliyölerindeki konut değerleriyle ilgilenir. Amaç, mevcut diğer tüm sürekli değişkenleri kullanarak, sahibinin oturduğu evlerin ortalama veya ortanca değerlerini bulmaktır.

```
set.seed(500)
library(MASS)
data <- Boston

#Adım 2: Ardından, veri kümesindeki eksik değerleri veya veri noktalarını kontrol edin.
#Varsa, eksik olan veri noktalarını düzeltin
apply(data, 2, function(x) sum(is.na(x)))


#Adım 3: Hiçbir veri noktası eksik olmadığından veri setini hazırlamaya devam edin.
#Şimdi verileri rastgele iki kümeye ayırın, Eğitim seti ve Test seti.
#Verileri hazırlarken, verileri doğrusal bir regresyon modeline sığdırmaya çalışın ve
ardından test setinde test edin.

index <- sample(1 : nrow(data),
                round(0.75 * nrow(data)))
train <- data[index, ]
test <- data[-index, ]
lm.fit <- glm(medv~., data = train)
summary(lm.fit)
pr.lm <- predict(lm.fit, test)
MSE.lm <- sum((pr.lm - test$medv)^2) / nrow(test)


#Adım 4: Şimdi bir Sinir ağını eğitmeden önce veri setini normalleştirin. Bu nedenle,
verileri ölçekleme ve bölme.

maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data,
                              center = mins,
                              scale = maxs - mins))
train_ <- scaled[index, ]
test_ <- scaled[-index, ]
```

```
#Adım 5: Şimdi verileri Sinir ağına sığdırın. Neuralnet paketini kullanın


library(neuralnet)
n <- names(train_)
f <- as.formula(paste("medv ~",
                      paste(n[!n %in% "medv"],
                            collapse = " + ")))
nn <- neuralnet(f,
                data = train_,
                hidden = c(4, 2),
                linear.output = T)

#Artık modelimiz Çok Katmanlı Sinir ağına uyarlanmıştır. Şimdi tüm adımları birleştirin
ve çıktıyı
#görselleştirmek için sinir ağını çizin. Bunu yapmak için plot() işlevini kullanın.


# R program to illustrate
# Multi Layered Neural Networks

# Use the set.seed() function
# To generate random numbers
set.seed(500)

# Import required library
library(MASS)

# Working on the Boston dataset
data <- Boston
apply(data, 2, function(x) sum(is.na(x)))
index <- sample(1 : nrow(data),
                round(0.75 * nrow(data)))
train <- data[index, ]
test <- data[-index, ]
lm.fit <- glm(medv~., data = train)
summary(lm.fit)
pr.lm <- predict(lm.fit, test)
MSE.lm <- sum((pr.lm - test$medv)^2) / nrow(test)
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data,
                              center = mins,
                              scale = maxs - mins))
train_ <- scaled[index, ]
test_ <- scaled[-index, ]

# Applying Neural network concepts
library(neuralnet)
n <- names(train_)
f <- as.formula(paste("medv ~",
                      paste(n[!n %in% "medv"],
                            collapse = " + ")))
nn <- neuralnet(f, data = train_,
                hidden = c(4, 2),
                linear.output = T)

# Plotting the graph
plot(nn)
```


crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
1
1
1
medv

Denetimsiz Öğrenme

Denetimsiz öğrenmede, tahmin edebileceğiniz gibi, eğitim verileri etiketsizdir. Sistem öğretmen olmadan öğrenmeye çalışır. Ham verileri organize verilere dönüştüren bir Makine Öğrenimi türüdür.

Bilgisayar, etiketlenmemiş verilerle eğitilir. Çoğunlukla desen algılama ve tanımlayıcı modellemede kullanılır. Yalnızca giriş verilerinizin (X) olduğu ve karşılık gelen çıkış değişkenlerinin olmadığı öğrenme biçimidir.

Denetimsiz öğrenme, verilerimizden elde etmek istediğimiz çıktının nasıl göründüğü hakkında çok az ya da hiç fikir sahibi olmadığımızda kullandığımız yaklaşımdır. Değişkenlerin etkisini bilmediğimiz veriden modeli oluşturabiliriz.

"Denetimsiz öğrenmede" sadece veriler vardır onlar hakkında bilgi verilmez. Bu verilerden sonuçlar çıkarılmaya çalışılır. En baştan veriler hakkında herhangi bir bilgi verilmediği için çıkartılan sonuçların kesinlikle doğru olduğu söylenemez.

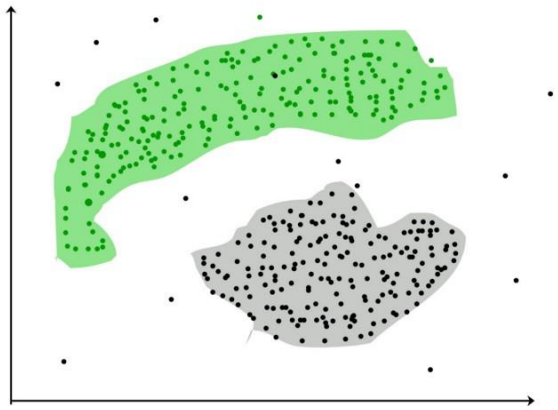
Veriyi değişkenler arasındaki ilişkilere dayalı olarak kümeleyerek çeşitli modeller, yapılar oluşturabiliriz.

Kümeleme

Geçmişteki verilerin sınıfları/etiketleri verilmediği/bilinmediği durumlarda verilerin birbirlerine yakın benzerliklerinin yer aldığı kümelerin bulunmasıdır.

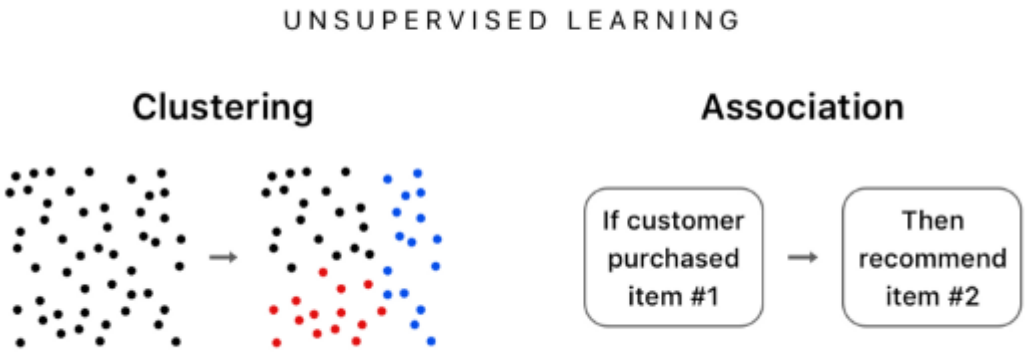
Kümeleme sorunu, satın alma davranışı yoluyla müşterileri gruplama gibi verilerdeki doğal gruplamaları keşfetmek istediğimiz yerdir.

Verilerin yakınlık, uzaklık, benzerlik gibi ölçütlere göre analiz edilerek sınıflara ayrılmasına kümeleme denir.



Örneğin, blogunuzun ziyaretçileri hakkında çok fazla veriniz olduğunu varsayalım. Benzer ziyaretçilerin gruplarını algılamaya çalışmak için bir kümeleme algoritması çalıştırdığımızı varsayalım . Algoritmamıza ziyaretçilerin hangi gruba ait olduğunu söylemeyiz, algoritmamız bu bağlantıları bizim yardımımız olmadan bulur.

Başka bir örnek verelim, ziyaretçilerimizin % 30'unın makine öğrenmesiyle ilgilenen ve genellikle akşamları blogumuzu okuyan kızlar olduğunu, % 50'sinin ise hafta sonları ziyaret eden genç yazılım severler olduğunu fark edebiliriz. Hiyerarşik bir kümeleme algoritması kullanırsanız, her grubu daha küçük gruplara ayırabiliriz. Bu, yayınlarımızı her grup için farklı hedeflememize yardımcı olabilir.



- 256 rengi 16 renge nasıl indiririz ?

- Kümeleme algoritmaları eğiticisiz öğrenme metotlarıdır.
- Örneklere ait sınıf bilgisini kullanmazlar.
- Temelde verileri en iyi temsil edecek vektörleri bulmaya çalışırlar.
- Verileri temsil eden vektörler bulunduktan sonra artık tüm veriler bu yeni vektörlerle kodlanabilirler ve farklı bilgi sayısı azalır.
- Bu nedenle birçok sıkıştırma algoritmasının temelinde kümeleme algoritmaları yer almaktadır.

Elimizde tek boyutlu 10 örnek içeren bir verimiz olsun.

12 15 13 87 4 5 9 67 1 2

Bu 10 farklı veriyi 3 farklı veri ile temsil etmek istersek

12 12 12 77 3 3 3 77 3 3 şeklinde ifade ederiz.

Gerçek değerler ile temsil edilen değerler arasındaki farkı minimum yapmaya çalışır.

Yukarıdaki örnek için 3 küme oluşmuştur.

12-15-13 örnekleri 1. kümede

87-67 örnekleri 2. kümede

4-5-1-2-9 örnekleri 3. kümede yer almaktadır.

Kümelemede Yaygın Kullanılan İki Yöntem Vardır:

K-Means

K-Ortalama algoritması bir kümeleme algoritmasıdır. Buradaki "K" kaç kümeye ayrılacağını temsil eder. Örnek 2 dediğimizi varsayarsak elimizdeki veride ortak 2 tane merkez nokta bulunmaya çalışılır. Ortak noktalar bulunduktan sonra diğer veriler yakında olduğu noktaya göre kümelenir. Ortak noktalar en verimli şekili alana kadar bu algoritma devam eder.

• Kümeleme algoritmalarının en basitidir. Veriyi en iyi ifade edecek K adet vektör bulmaya çalışır. K sayısı kullanıcı tarafından verilir. Nümerik değerler için çalışır.

i adet merkez belirlemek için :

• Rasgele K adet küme merkezi atanır (C1, C2, ...,Ck)

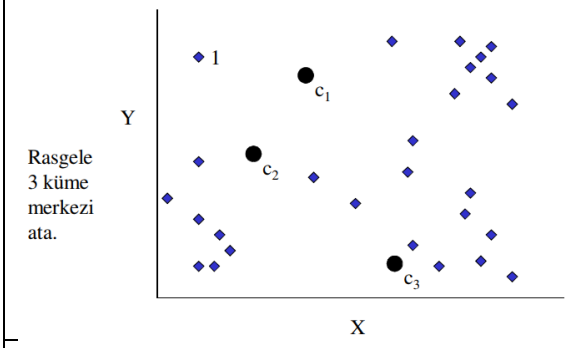
• Her örnek en yakınındaki merkezin kümesine atanır

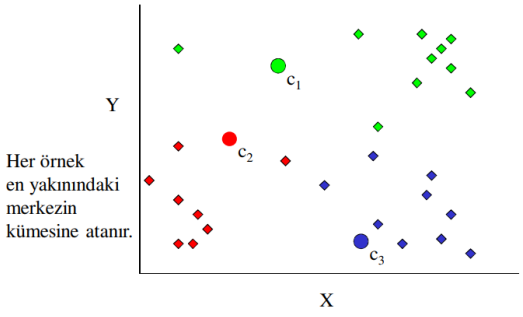
• Ci'ler tekrar hesaplanır (her kümedeki örneğin ortalaması alınır)

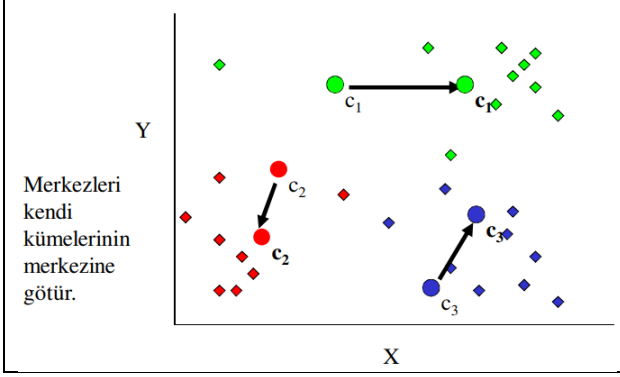
• Ci' lerde değişiklik olmuş ise 2. ve 3. adımlar tekrar edilir. Bu işleme küme değiştiren örnek kalmayıncaya kadar devam edilir, aksi taktirde işlem sonlandırılır.

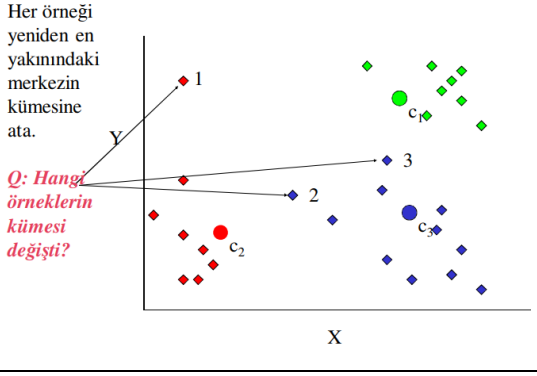
Solda 256 renkle ifade edilen resim, sağda da K-Means kullanılarak 16 renge indirilmiş resim görülmektedir.

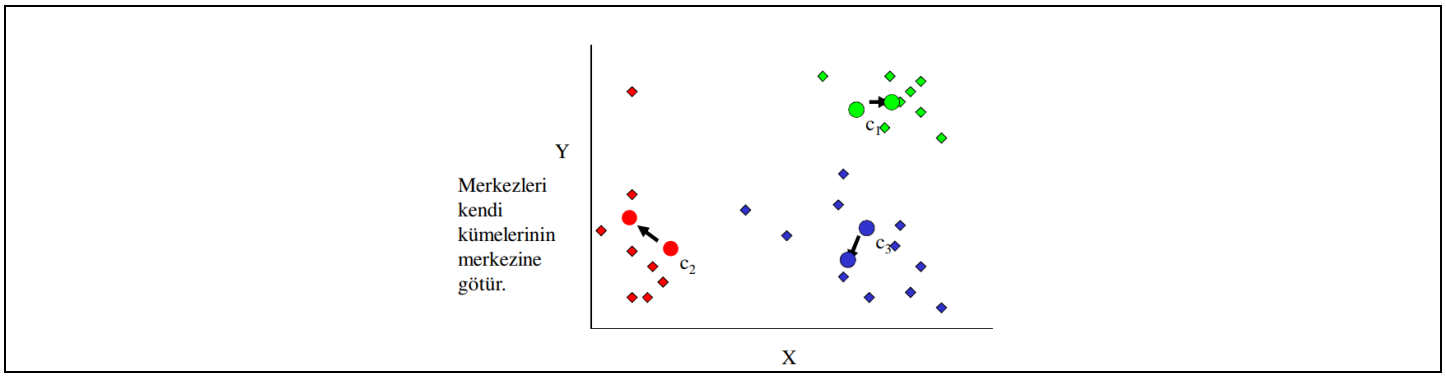
K-Means İşlem Basamakları







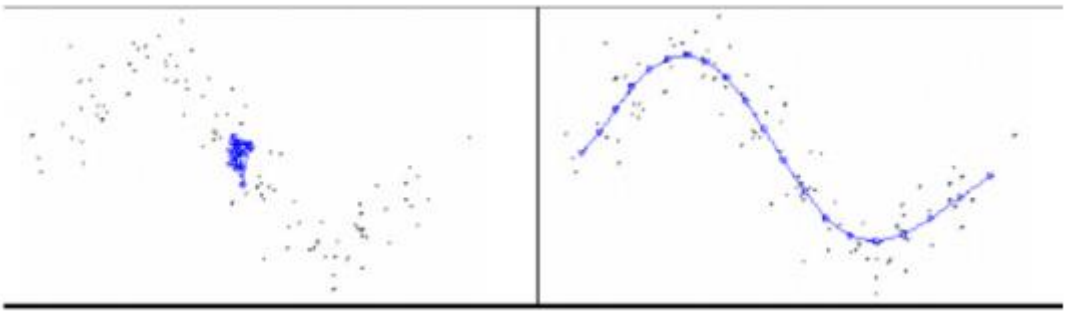




Kendi Kendini Düzenleyen Haritalar (SOM)

• K-Means algoritmasında merkez noktalar arasında herhangi bir ilişki yok iken SOM'da merkez noktalar 1 veya 2 boyutlu dizi içerisinde yer alırlar.

• K-Means algoritmasında sadece kazanan merkez güncellenirken, SOM 'da bütün merkezler kazanan nöronun komşuluklarına göre güncellenirler. Yakın komşular uzak komşulara göre daha fazla hareket eder.



SOM merkezleri 1 boyutlu bir dizide birbirlerine komşudur, başlangıçta rasgele atandıkları için yığılma mevcuttur ancak eğitim tamamlandıktan sonra SOM merkezleri düzgün dağılmıştır.

Birliktelik Kuralları Keşfi

Elimizdeki verilerin aralarındaki birliktelik ilişkisi çözmeye yönelik algoritmadır. Çok sık alışveriş takiplerinde kullanılır. Örnek bir ürünü alan kullanıcıların yanında aldıkları ürünlerin yoğunluğuna göre diğer kullanıcılara ürün tavsiye edilmesi gibi.

Association kuralı öğrenme, X satın alan kişiler de Y satın alma eğiliminde olduğu gibi verilerimizin büyük bölümlerinde nitelikler arasındaki ilginç ilişkileri keşfetmek olduğu ilişkilendirme kuralı öğrenmesidir. Örneğin, bir süpermarkete sahip olduğumuzu varsayalım. Satış günlüklerimizde bir association kuralı çalıştırmak, barbekü sosu ve patates cipsi satın alan kişilerin de biftek alma eğiliminde olduğunu ortaya çıkarabilir. Bu nedenle, bu öğeleri birbirine yakın yerleştirmek isteyebiliriz.

Bir süpermarkette, x ürününü alan müşterilerin %80'i y ürününü de alıyorsa, x ürününü alıp, y ürününü almayan müşteriler, y ürününün potansiyel müşterileridir. Müşterilerin sepet bilgilerinin bulunduğu bir veri tabanında potansiyel y müşterilerini bulma işlemi türündeki problemler ilişki belirme yöntemleri ile çözülür.

- Sepet analizi
- Raf düzenlemesi

- Promosyonlar

Apirior Algoritması

Birliktelik kurallarının algoritmalarından biridir. Kısacası "müşteriler hangi ürünleri birlikte alıyorlar?" sorusunu cevabını vermeye çalışır. Müşterilerin aldığı ürünlerin birlikteliğini inceleyerek bir sonraki müşteri aynı ürünü aldığında tavsiye algoritması çalıştırarak ona tavsiye edilecek ürünlerin listesini çıkarır. "Priori" anlamı "önce" anlamına gelir ve kendinden önceki ürünlerin durumuna göre sonuçlar çıkarır.

Özellik Seçimi

Veriye ait olan birçok özellikten bazıları ilgili verinin kümesini/sınıfını belirlemede önemli rol oynar. Bu gibi durumlarda özellik kümesinin bir alt kümesi seçilir (özellik seçimi) veya bu özelliklerin birleşiminden yeni özellikler elde edilebilir (özellik çıkarımı).

Kısaca, Makineler, isanlığın işgücüne sağladıkları yararı, makine öğrenmesi yöntemleri ile birleştirdiklerinde beyin gücünü de sağlamayı başarmışlardır. Uygulama alanı ne olursa olsun, çok miktardaki verinin analiz edilerek gelecek ile ilgili tahminlerde bulunması ve bizim karar vermemize yardımcı olması ile makine öğrenmesi yöntemlerinin her geçen gün önemi artmaktadır.

Genel Konular

Makine Öğrenmesi Nedir?

Makine Öğrenmesi, bilgisayarların kendi başlarına öğrenmelerini sağlar. Bu tür öğrenme, büyük veri setlerini kolayca işleyebilen modern bilgisayarların işlem gücünden yararlanır.

Denetimli Öğrenme, girdilere ve beklenen çıktılara sahip etiketli veri setlerini kullanmayı içerir. Denetimli öğrenmeye örnek, hava durumu belirleyici Yapay Zeka'dır. Geçmiş verilerini kullanarak hava durumunu tahmin etmeyi öğrenir. Bu eğitim verilerinde girdiler (basınç, nem, rüzgar hızı) ve çıktılar (sıcaklık) bulunur.

Denetimsiz Öğrenme, belirli bir yapıya sahip olmayan veri kümelerini kullanan makine öğreniminin görevidir. Denetlenmemiş öğrenmeyi kullanarak bir Yapay Zekayı eğitirseniz, Yapay Zeka'ya verilerin mantıksal sınıflandırmasını yapma izin verirsiniz. Denetimsiz öğrenmenin bir örneği, bir e-ticaret web sitesi için tahmin yapan yapay zeka örnek verilebilir. Çünkü burada etiketli bir girdi ve çıktı veri seti kullanılarak öğrenilmez. Bunun yerine girdi verileri kullanarak kendi sınıflandırmasını oluşturacaktır. Hangi tür kullanıcıların daha fazla farklı ürün alabileceklerini size söyleyecektir.

Sınıflandırma ve Kümeleme arasındaki fark nedir?

Sınıflandırma, gözetimli(supervised); kümeleme gözetimsiz(unsupervised) öğrenme metodudur.

Sınıflandırmada	Kümelemede
-Verilerin etiketi vardır. -Verileri bir gruba dahil etmek için bir kural oluşturulmasını bekler. -Veri setini eğitim ve test olarak ayırmak gereklidir.	-Verilerin etiketi yoktur. -Verileri bir gruba yakınlığa/benzerliğe/alakaya/hiyerarşiye göre dahil eder. -Verideki örüntülerini ve yapıları tespit eder.

Derin Öğrenme Nedir?

Derin Öğrenme bir makine öğrenme yöntemidir. Verilen bir veri kümesi ile çıktıları tahmin edecek yapay zekayı eğitmemize olanak sağlar. Yapay zekayı eğitmek için hem denetimli hem de denetimsiz öğrenme kullanılabilir.

Derin Öğrenme ile uçak bileti fiyat tahmini örneği üzerinden çalışma mantığını anlatmaya çalışacağım. Bu örnekte denetimli öğrenme yöntemi kullanacağız.

Uçak bileti fiyatlarını tahmin ederken, aşağıdaki girdileri kullanmak istiyoruz diyelim (şuan için tek yön uçuşları düşünüyoruz):

Kalkış Havalimanı

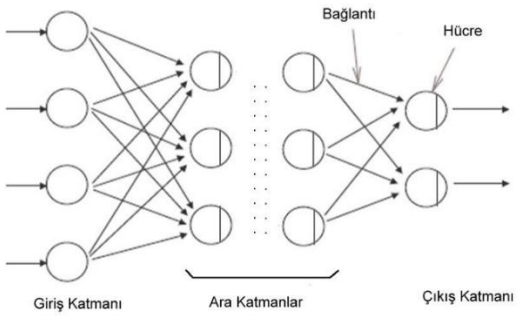
Varış Havalimanı

Kalkış Tarihi

Firma

Yapay Sinir Ağları

Yapay sinir ağları tıpkı insan beyni gibi nöronlardan oluşur. Tüm nöronlar birbirine bağlıdır ve çıktıyı etkilemektedir.



Nöronlar üç farklı katmana ayrılır:

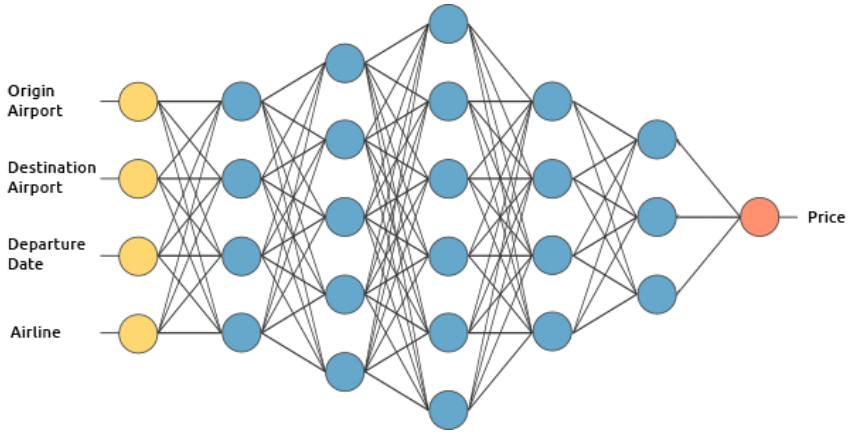
1. Giriş Katmanı
2. Gizli Katman(lar)
3. Çıkış Katmanı

Giriş katmanı giriş verilerini alır. Bizim örneğimizde, giriş katmanında dört nöron var: Kalkış Havaalanı, Varış Havaalanı, Kalkış Tarihi ve Firma. Giriş katmanı, girişleri ilk gizli katmana gönderir.

Gizli katmanlar girdilerimizde matematiksel hesaplamalar yapar. Yapay sinir ağları oluşturmadaki zorluklardan biri, her bir katman için nöronların sayısının yanı sıra gizli katmanların sayısına da karar vermektir.

Derin Öğrenmedeki "Derin", birden fazla gizli katmana sahip olmayı ifade eder.

Çıktı katmanı, çıktı verilerini döndürür. Bizim örneğimizde, bize fiyat tahmini verir.



Peki fiyat tahmini nasıl yapılır?

Derin Öğrenmenin başladığı yer burasıdır.

Nöronlar arasındaki her bağlantı bir ağırlık(weight) ile ilişkilidir. Bu ağırlık, girdi değerinin önemini belirler. İlk ağırlıklar rastgele ayarlanır.

Bir uçak biletinin fiyatını tahmin ederken, en önemli (ağır) faktörlerden biri kalkış tarihidir. Bu nedenle, kalkış tarihi nöron bağlantıları büyük bir ağırlığa sahip olacak.

Her nöron bir ***Aktivasyon Fonksiyonuna*** sahiptir. Aktivasyon foksiyonunun amaçlarından biri nörondan elde edilen çıktıları "***standartlaştırmak***" tır.

Bir veri kümesi sinir ağının tüm katmanlarından geçtikten sonra, çıktı katmanından sonuç olarak döner.

Yapay Sinir Ağını Eğitme

Derin öğrenmenin en zor kısımlarından biri yapar sinir ağını eğitmektir.

1. Büyük bir veri kümesine ihtiyaç var.
2. Çok fazla miktarda hesaplama gücüne ihtiyacınız var.

Uçak bileti fiyat tahmini uygulaması için, bilet fiyatlarının geçmiş verilerini bulmamız gerekiyor. Çok sayıda muhtemel havalimanı ve kalkış tarihi kombinasyonundan dolayı çok geniş bir bilet fiyatı listesine ihtiyacımız var.

Yapay zekayı eğitmek için, veri kümemizde ki girdileri vermemiz ve çıktılarını veri kümemizde ki çıktılarla karşılaştırmamız gerekir. Yapay zeka hala eğitimsiz olduğundan, çıktıları yanlış olacaktır.

Tüm veri kümesini incelediğimizde, yapay zeka çıktılarının gerçek çıktılardan ne kadar yanlış olduğunu gösteren bir fonksiyon oluşturabiliriz. Bu fonksiyona ***Maliyet Fonksiyonu*** denir.

Eğitim süresince yapmak istediğimiz maliyet fonksiyonu değerinin 0 olmasıdır. Böylece yapay zeka çıktıları ile veri kümesinde ki çıktıların aynı olduğu anlamına gelir.

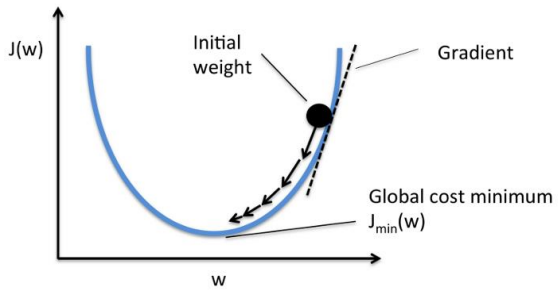
Maliyet Fonksiyonunu nasıl azaltabiliriz?

Nöronlar arasındaki ağırlıkları değiştiririz. Maliyet fonksiyonu düşük olana kadar ağırlıkları rastgele değiştirebiliriz, ancak bu çok verimli değil.

Bunun yerine, Gradient Descent adlı bir teknik kullanacağız.

Gradient Descent, bir fonksiyonun minimumunu bulmamızı sağlayan bir optimizasyon algoritmasıdır. Gradyan İniş tekniği ile ağ üzerinde bir ileri bir geri gidip hatayı azaltmak için model parametreleri ayarlanır. Bu ileri ve geri gidişler, çözüme yaklaşıncaya kadar tekrar eder. Bizim örneğimizde, maliyet fonksiyonunun minimumunu arıyoruz.

Gradient Descent her veri seti yinelemesinden sonra ağırlıkları küçük artışlarla değiştirerek çalışır. Maliyet fonksiyonunun türevini (veya gradyanını) belirli bir ağırlık kümesinde hesaplayarak, minimumun hangi yönde olduğunu görebiliriz.



Maliyet fonksiyonunu en aza indirmek için, veri kümenizde birçok kez yineleme yapmanız gerekir. Bu yüzden büyük miktarda hesaplama gücüne ihtiyacınız var.

Ağırlıklar, Gradient Descent kullanılarak otomatik olarak güncellenir. İşte bu, Derin Öğrenmenin sihridir!

Uçak bileti fiyat tahmini için yapay zekamızı eğitip hata oranını en aza indirdikten sonra, gelecekteki fiyatları tahmin etmek için kullanabiliriz.

Özetle:

1. **Derin Öğrenme, hayvan zekasını taklit etmek için Yapay Sinir Ağlarını kullanır.**
2. **Bir sinir ağında üç tür nöron katmanı vardır: Giriş Katmanı, Gizli Katmanlar ve Çıkış Katmanı.**
3. **Nöronlar arasındaki bağlantılar, giriş değerinin önemini belirleyen bir ağırlıkla ilişkilendirilir.**
4. **Nöronlar, nörondan çıkan çıktıyı "standartlaştırmak" için verilerde bir Aktivasyon Fonksiyonu kullanırlar.**
5. **Yapay sinir ağını eğitmek için büyük bir veri kümesine ihtiyaç vardır.**
6. **Veri kümesi boyunca yineleme yapmak ve çıktıları karşılaştırmak, yapay zekanın gerçek çıktılardan ne kadar uzakta olduğunu gösteren bir Maliyet Fonksiyonu üretecektir.**
7. **Veri kümesindeki her yinelemeden sonra, maliyet fonksiyonunu azaltmak için nöronlar arasındaki ağırlıklar Gradient Descent kullanılarak ayarlanır.**

Makine Öğrenmesinde Overfitting, Underfitting, Cross Validation, Bias Kavramlarını açıklayınız?

Model girdilerin çıktılara eşlenmesi için kullanılan bir sistemdir. Örneğin amacımız ev fiyatlarını tahmin etmek olsun. Bunun için evin metrekare bilgisini girdi olarak kullanan bir model oluştururuz ve çıktı olarak da evin fiyatını

elde ederiz. Bir model bir problem hakkında teori üretir. Buradaki teori evin metrekare bilgisi ile fiyatı arasında bir ilişki olmasıdır. Eğitim veri çalışmaları sonucunda bu ilişkiyi öğrenmiş olan bir model oluşturmuş oluruz.

Makine öğreniminde gelecek veriler hakkında tahmin yapabilmek için verilerimizi eğitim verileri (train) ve test verileri olmak üzere iki alt kümeye ayırıyoruz. Modelimizi eğitim verilerinden elde edilen örüntülere göre oluşturuyoruz. Bu işlem sonucunda iki şeyden biri olabilir; modelimiz aşırı öğrenebilir veya eksik öğrenebilir. Bu durumda modelimiz yeterli öngörüde bulunamayacak ve tahminlerimizde hata oranı yüksek olacaktır.

Overfitting: Eğer modelimiz, eğitim için kullandığımız veri setimiz üzerinde gereğinden fazla çalışıp ezber yapmaya başlamışsa ya da eğitim setimiz tek düze ise overfitting olma riski büyük demektir. Eğitim setinde yüksek bir skor aldığımız bu modele, test verimizi gösterdiğimizde muhtemelen çok düşük bir skor elde edeceğiz. Çünkü model eğitim setindeki durumları ezberlemiştir ve test veri setinde bu durumları aramaktadır. En ufak bir değişiklikte ezberlenen durumlar bulunamayacağı için test veri setinde çok kötü tahmin skorları elde edebilirsiniz. Overfitting problemi olan modellerde yüksek varyans, düşük bias durumu görülmektedir.

Bu genellikle model çok karmaşık olduğunda (yani gözlem sayısına kıyasla çok fazla özellik / değişken varsa) gerçekleşir. Bu model eğitim verilerinde çok yüksek tahmin doğruluğuna sahip olacaktır, ancak eğitimsiz veya yeni verilerde muhtemelen çok doğru tahminleme yapamayacaktır. Bu sorun modelin genelleştirme yapamamasından kaynaklanmaktadır. Bu tip modeller verilerdeki değişkenler arasındaki gerçek ilişkiler yerine eğitim verilerindeki "gürültüyü" öğrenir veya açıklar.

Overfitting problemi aşağıdaki yöntemler uygulanarak çözülebilmektedir;

- Öz nitelik sayısını azaltmak,
- Daha fazla veri eklemek
- Regularization (Düzenleme)

Underfitting: Aşırı öğrenmenin aksine, bir model yetersiz öğrenmeye sahipse, modelin eğitim verilerine uymadığı ve bu nedenle verilerdeki trendleri kaçırdığı anlamına gelir. Ayrıca modelin yeni veriler için genelleştirilemediği anlamına da gelir. Tahmin ettiğiniz gibi bu problem genellikle çok basit bir modelin sonucudur (yetersiz tahminleyici bağımsız değişken eksikliği).

Underfitting sorunu olan modellerde hem eğitim hem de test veri setinde hata oranı yüksektir. Düşük varyans ve yüksek bias'a sahiptir. Bu modeller eğitim verilerini çok yakından takip etmek yerine, eğitim verilerinden alınan dersleri yok sayar ve girdiler ile çıktılar arasındaki temel ilişkiyi öğrenemez.

Varyans: model eğitim veri setinde iyi performans gösterdiğinde, ancak test veri kümesinde iyi performans göstermediğinde ortaya çıkar. Varyans, gerçek değerden tahmin edilen değerin ne kadar dağınık olduğunu söyler.

Bias: gerçek değerlerden tahmin edilen değerlerin ne kadar uzak olduğudur. Tahmin edilen değerler gerçek değerlerden uzaksa, bias yüksektir.

- **Yüksek Bias Düşük Varyans:** Modeller tutarlıdır, ancak ortalama hata oranı yüksektir.
- **Yüksek Bias Yüksek Varyans:** Modeller hem hatalı hem de tutarsızdır.
- **Düşük Bias Düşük Varyans:** Modeller ortalama olarak doğru ve tutarlıdır. Modellerimizde bu sonucu elde etmek için çabalamaktayız.
- **Düşük Bias Yüksek Varyans:** Modeller bir dereceye kadar doğrudur ancak ortalamada tutarsızdır. Veri setinde ufak bir değişiklik yapıldığında büyük hata oranına neden olmaktadır.

Cross-validation: makine öğrenmesi modelinin görmediği veriler üzerindeki performansını mümkün olduğunca objektif ve doğru bir şekilde değerlendirmek için kullanılan istatistiksel bir yeniden örnekleme(resampling) yöntemidir. İkinci bir kullanım alanı ise modelde hiperparametre optimizasyonu yapmaktır.

Derin Öğrenmedeki Katmanlar Nelerdir?

Derin öğrenme katmanları normal bir durumda makineye bir bilginin öğretilmesini insanlar sağlıyor. Bu sayede cihazlar çalışabiliyor. Derin öğrenmede ise durum çok farklı. Makineler öğrendikleriyle verileri analiz ediyor, yeni veriler çıkarıyor.

Bu işlemlerin her biri ise katmanlar sayesinde oluyor. Bir beyin gibi işleyen sistem milyarlarca yöntemi karşılaştırıyor, sınıflandırıyor, olasılığını hesaplıyor ve bir sonuç çıkarıyor. Derin öğrenme katmanları ise 9 gruba ayrılıyor. Bu katmanlar kendi içerisinde aldığı veriyi işleyerek diğerine gönderiyor ve net bir sonuç çıkarabiliyor. Daha yakından inceleyecek olursak;

Giriş Katmanı: Bu kısıma hazırlama alanı diyebiliriz. Veri seti ağının mimarisine göre girdisi yapılan veriyi hazırlar ve gönderir. Giriş katmanına girilen verinin boyutuna göre işlem için gerekli olan ağ hızı ve bellek ihtiyacı gibi temel bileşen ihtiyacını arttırabilmektedir.

Konvolüsyon Katmanı: Girilen verinin belirgin özelliklerini ön plana almaya yarayan katmandır. Bu özellikler belirlenen filtrelerden geçer. Burada kullanılan filtre projenin başarı oranında direkt etki etmektedir. Evrişimli katmanı bir dizi "filtreden" oluşan bir katmandır. Filtreler, bir seferde giriş (input) verilerinin bir alt kümesini alır, ancak tüm girdiye uygulanır.

Aktivasyon Katmanı: Matematik işlemlerinin gerçekleştiği alanda denilebilir. Veri gerekli fonksiyonlardan (tanjant, sinüs, step ve benzeri) geçerek bir değer alır. Elde edilen değer negatif ise 0, pozitif ise 1 değerini alır.

Havuzlama Katmanı: Verilerin belli kısımları çıkarılarak kullanılacak bellek ve ağ miktarını azaltmayı hedefler. Bu işlem veri üzerinde kayıplar oluşturabilmektedir.

Tam Bağlı Katman: Nesneyi belirleyemek için kullanılan özelliklerin hangi sınıf(lar)a ait olduğunu anlamak için kullanılır.

Dropout Katmanı: Çok katmanlı yapay sinir ağları aşırı öğrenme adı verilen bir durumla karşılaşmaktadır. Bu istenmediği için dropout katmanında ağda ezber yapan özelliklerin kaldırılması gerçekleşir.

Sınıflandırma Katmanı: Sınıflandırılması yapılacak öğe kadar sonuç üretir. Her biri bir sınıfı temsil eden bu öğeleri genelde softmax sınıflandırıcısı ile sınıflandırır.

Yumuşatma Katmanı: Gelen verileri olasılık hesaplaması yaparak hangi sınıfa ait olup olmadığını belirler.

Normalizasyon Katmanı: Eğitim süresinin azalması için büyük ya da küçük olan girdileri düzenler.

Derin Öğrenme katmanları tam olarak mantığı açıklamaya çalışırsak işlem şu şekilde yürüyor: Veri alınır, ayrılır ve küçültülür. Sonrasında her biri birbirinden farklı fonksiyonlarda sınıflandırılarak net bir çıktı alınır. Bu çıktının alınması için oldukça güçlü bir bilgisayar gereklidir.

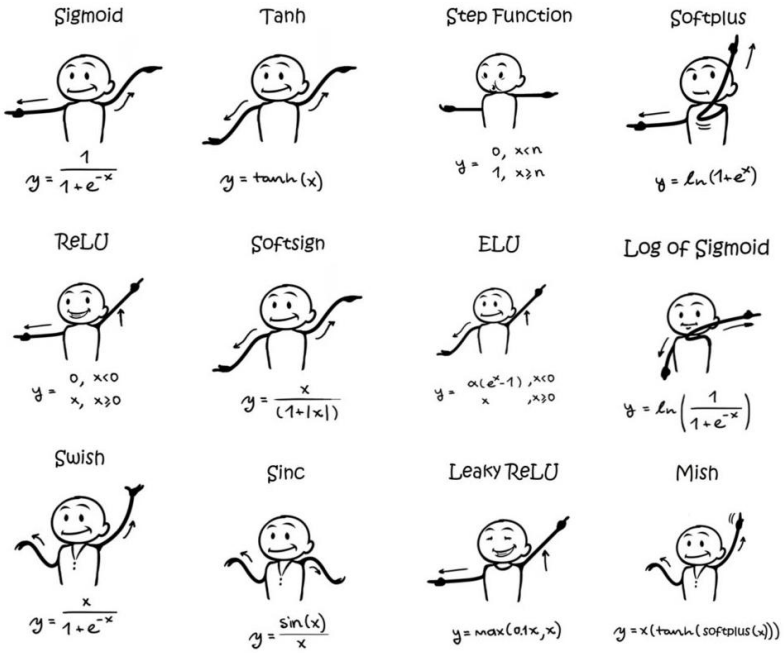
Derin Öğrenmedeki Aktivasyon Fonksiyonu Nedir?

Her nöron bir Aktivasyon Fonksiyonuna sahiptir. Aktivasyon foksiyonunun amaçlarından biri nörondan elde edilen çıktıları "standartlaştırmak" tır.

Eğer aktivasyon fonksiyonu uygulanmazsa çıkış sinyali basit bir doğrusal fonksiyon olur. Aktivasyon fonksiyonu doğrusalsızlıkları arttırmak için kullanılır.

Aktivasyon fonksiyonu kullanılmayan bir sinir ağı sınırlı öğrenme gücüne sahip bir doğrusal regresyon (linear regression) gibi davranacaktır. Ama biz sinir ağımızın doğrusal olmayan durumları da öğrenmesini istiyoruz. Çünkü sinir ağımıza öğrenmesi için görüntü, video, yazı ve ses gibi karmaşık gerçek dünya bilgileri vereceğiz. Çok katmanlı derin sinir ağları bu sayede verilerden anlamlı özellikleri öğrenebilir.

Her zaman yapılacak iş: Girişler ile ağırlıkları çarp, bias ile topla ve aktivasyon uygula!



İleri Yayılım ve Geri Yayılım

Bu girdilerin ileriye doğru geçişine forward pass denir. İleriye doğru geçiş ile en son bir tahmin bulunur ama geri geçişte kullanılmak için ara geçiş sonuçları saklanır. Algoritma çıkan tahmin ile gerçek sonuç değerini karşılaştırır ve loss fonksiyonu ile hatayı ölçer.

Daha sonra algoritma geriye doğru giderek her bir çıktının hataya katkısını hesaplar. Son olarak, algoritma, ağdaki bütün bağlantı ağırlıklarını tekrar ayarlayarak hata azaltılır.

Başlangıçta atanan ağırlıkların rasgele olması önemlidir aksi takdirde eğitim başarısız olur. Ağırlık toplamları her nöronda bir aktivasyon fonksiyonundan geçirilir. Sigmoid, ReLU gibi çeşitli aktivasyon fonksiyonları vardır.

Geri Yayılım Algoritması

Bir yapay sinir ağı, her birinde özel hesaplamaların yapıldığı nöronlardan oluşmaktadır ve bu nöronların özelleştiği 3 tür katmandan bahsedilmektedir. Bunlar; girdi katmanı, gizli katman ve çıktı katmanıdır. Girdi katmanı, göreve özel olarak türü değişen verilerin sinir ağına sunulduğu katmandır. Veriler, bir ses kaydı (Doğal Dil İşleme), bir görüntü (Görüntü İşleme) veya bir metin (Duygu Analizi) gibi farklı türlerde olabilir.

Verinin türüne göre, yapay sinir ağının gerçekleştireceği işlemler farklılık gösterecektir. Örneğin bir görüntü verisinde, öncelikle renk ayrıştırma veya nesnelerin kenarlarını ayırt etme işlemleri yapılırken; metinsel bir veride öncelikle kelimeleri köklerine ayrıştırma işlemi gerçekleştirilir.

Bir sinir ağındaki katmanlar arasında, her bir katmandaki nöronu bir sonraki katmandaki nöronlara bağlayan bağlantılar ve her bir bağlantının sayısal olarak bir değeri vardır. Bu değerlere ağırlık denir. Ağırlık değerleri, bağlı oldukları her bir nöronun, eğitimin sonucunda alınacak çıktı değeri için ne kadar öneme sahip olduğunu gösterir.

Girdi katmanına ait nöronlardaki her bir değer, kendisine bağlı olan ağırlık değeriyle çarpılır. Bir sonraki katmanda yer alan nöronların her birine denk gelen çarpım değerleri toplanır. Gizli katmanda, o katmana özel olarak belirlenmiş aktivasyon fonksiyonuna dayalı olarak bir takım hesaplamalar yapılmakta ve her bir nöronda hesaplanan değer, bir sonraki katmana yine ağırlık değerleriyle çarpılarak aktarılmaktadır.Bir yapay sinir ağında, girdi katmanından çıktı katmanına kadar gerçekleştirilen bütün bu ileri yönlü hesaplama akışına İleri Yayılım Algoritması adı verilmektedir.

İleri yayılım algoritması sonucunda elde edilen değerler, sinir ağının oluşturduğu tahmin değerleridir. Bu yüzden tahmin değerleri çoğu zaman %100 doğrulukta olamamaktadır ve tahmin değeriyle gerçek değer arasında bir fark olacaktır. Bu hatanın olabildiğince küçültülmesi için farklı yöntemler geliştirilmiştir. Geri yayılım algoritmasına göre, ileri yayılım algoritması sonucunda hesaplanan hata değeri, sinir ağı üzerinde çıktı katmanından girdi katmanına

doğru olacak şekilde çeşitli türev işlemlerine dâhil edilerek “geriye doğru yayılmış” olur. Bunun için öncelikle hata değerinin hesaplanması gerekir.

Apriori Algoritmasını kısaca açıklayınız

Veri Madenciliği alanında Birliktelik Kuralları Yönetimi (Association Rule Mining) konusu altında yer alan ve sıkça kullanılan apriori algoritmasının ismi, tümdengelim ile edinilen anlamına gelir. Büyük veri kümeleri arasındaki bağlantıyı ortaya çıkarmak için kullanılır. Apriori algoritması destek ve güven değerleri olmak üzere iki önemli kritere sahiptir. Destek değerleri ürün grubunun bütün veri içerisindeki oranını belirler. Güven değerleri ise ürünler arasındaki bağlantıyı bize verir. Bu algoritma kısaca bir veri kümesini adım adım keşfetmek ve veri kümesi içerisinde bulunan verilerin birliktelik kurallarını incelemek için kullanılan bir algoritmadır.

Apriori algoritmasının en kolay anlaşılır örneği alışveriş sepeti analizidir. Bir alışveriş sepetinde bulunan ürünlerden ya da fişlerden yola çıkarak hangi ürünün ne sıklıkla satıldığı, hangi ürünlerin birlikte satıldığı gibi önemli bilgilerin öğrenebileceği bir algoritmadır.

Doğal Dil İşleme Sırasındaki Temel Adımlar Nelerdir?

Doğal Dil İşleme (NLP), makinelerin insan diliyle nasıl etkileşime girdiğini inceleyen yapay zekanın bir parçasıdır. NLP, sohbet robotları, yazım denetleyicileri veya dil çevirmenleri gibi her gün kullandığımız birçok aracı geliştirmek için perde arkasında çalışır.

NLP, makine öğrenimi algoritmalarıyla birleştirildiğinde, görevleri kendi başına gerçekleştirmeyi öğrenen ve deneyim yoluyla daha iyi hale gelen sistemler oluşturur.

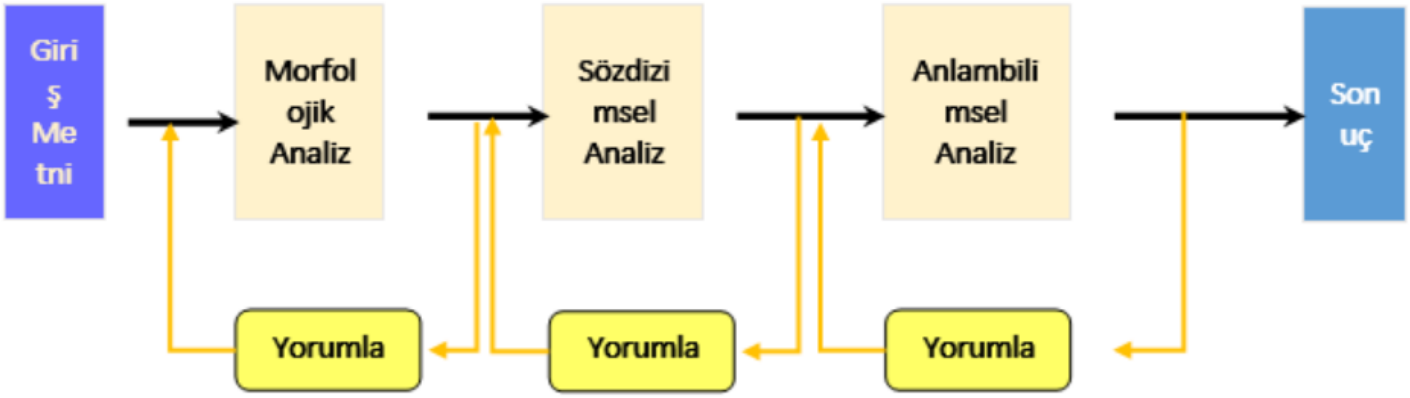
Doğal dil işleme (natural language processing), bilgisayarlara insan dilini okuma, anlama ve yorumlama yeteneği veren bir yapay zeka uygulamasıdır. Bilgisayarların, insan duygularını ölçmesine ve insan dilinin hangi bölümlerinin önemli olduğunu belirlemesine yardımcı olur.

Doğal dil işlemenin (NLP) amacı, metni anlamlandırabilen ve çeviri, dilbilgisi denetimi veya konu sınıflandırması gibi görevleri gerçekleştirebilen sistemler oluşturmaktır.Doğal dil işlemenin amacı yalnızca kurallı bir dil yapısının çözümlenerek anlaşılmasını olmayıp aynı zamanda çözümlenebilen bir dile ait yapının yeniden üretilebilmesini sağlamaktır. Google Assist, Siri ve Alexa gibi sanal asistanlar, doğal dil işlemenin hayatımızdaki uygulama alanlarının en popülerler örnekleri arasında yer alır.

Doğal dil ile üretilen metin ve konuşmalar (ses) belirli ön işlemlerden sonra benzer metodlarla işlenmektedir. Yazılı metinler doğal dil işleme modüllerine uygun bir hale getirildikten sonra bilgisayar sistemleri tarafından doğrudan işlenip analiz edilmektedir. Konuşmalar (ses), elektromanyetik dalgaların algılanması ile elde edildikten sonra öncelikle ses verileri yazılı metinlere dönüştürülmektedir. Yazılı metinlere dönüşümü tamamlanan ses verileri bu metinler üzerinden gerekli modülasyon işlemlerine tabi tutulduktan sonra işlenip analiz edilmektedir.

Genel anlamda NLP; dili daha kısa, temel parçalara böler, parçalar arasındaki ilişkileri anlamaya çalışır ve parçaların anlam yaratmak için birlikte nasıl çalıştığını keşfeder.

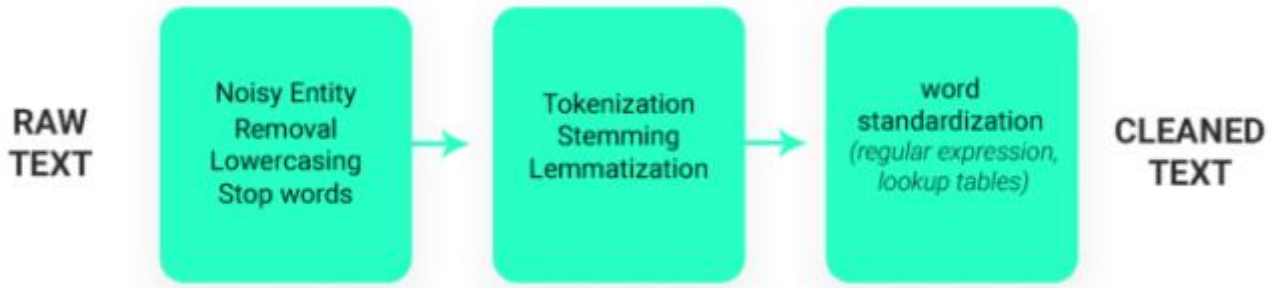
Doğal dil işlenmesi sırasında metnin gereksiz ifadelerden ve noktalamalardan ayrıştırılması, morfolojik analiz, söz dizimsel analiz ve anlam bilimsel analiz adımları sırasıyla farklı yorumlayıcı sistemler tarafından yapılmaktadır. Bu sürece dilin matematiksel analiz süreci adı verilmektedir.



(Doğal Dil İşleme Süreci)

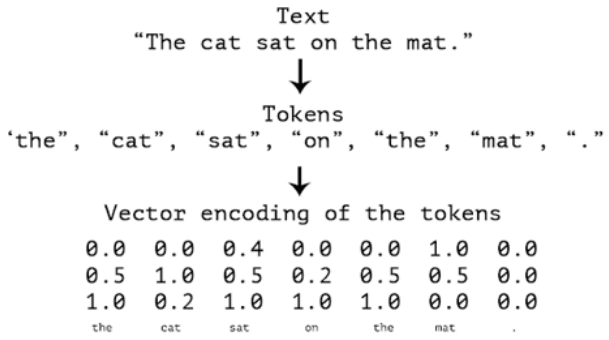
Doğal Dil İşleme Adımları

Veri ön işlemleri dışında doğal dil işleme sürecinde verinin(metnin) çeşitli işlemlerden geçmesi gerekmektedir.



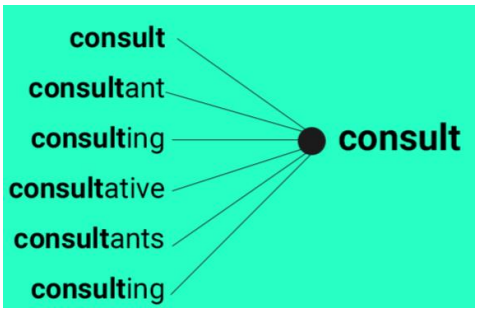
Bu adımlardan bazıları,

- **StopWord,** metnin içerisinde tekrar eden kelimelerin çıkartılması.
- **Lowercasing,** kelimelerin küçük harfe dönüştürülmesi. Noktalama ve anlamsız vs. karakterlerden kurtarılması (**noisy entity removal**).
- T**okenization,** metnin parçalara ayrılması.



Tokenization & Vector Encoding

- **Stemming**, kelime köklerinin ayrıştırılması ve (**lemmatization**) ile sözcüğün temel biçimlerini ortak bir forma indirgeme gibi adımlar bulunmakta.



Doğal Dil İşleme Teknikleri

Sözdizimsel Analiz

Sözdizimsel analiz veya ayrıştırma; cümle yapısını, kelimelerin nasıl düzenlendiğini ve kelimelerin birbirleriyle ilişkisini tespit etmek için temel dilbilgisi kurallarından yararlanarak metni analiz eder.

Anlamsal Analiz

Anlamsal analiz, metnin anlamını bulmaya odaklanır. İlk olarak, her bir kelimenin anlamını inceler (sözcüksel anlambilim). Ardından, kelimelerin kombinasyonuna ve bağlam içinde ne anlama geldiklerine bakar.

Büyük Verinin Temel Özellikleri Nelerdir?

Çeşitlilik (Variety), Hacim (Volume), Hız (Velocity), Doğruluk (Veracity), Değer (Value) özelliklerini barındırması gerekmektedir.

Çeşitlilik: Sağlıklı bir çalışma için çeşitli formattaki bilgilerin, birbirine dönüştürülebiliyor olmaları önemlidir.

Hacim: Veri akışı sürekli ve büyük hacimlerde olmalı.

Hız: Verinin devamlılık arzetmesi ve çok hızlı olması gerekmektedir.

Doğruluk: Verinin güvenilir ve doğru bilgiler içermesi gerekir. Veriler içinde doğru olmayan ve anlamsız kayıtların sağlıklı sonuç alabilmek adına temizlenmesi gerekmektedir.

Değer: Mevcut verilerin veri madenciliği metotlarıyla işlenmesi ve anlamlı sonuçlar üretilebilmesi gerekir.

Veri Madenciliği

Veri madenciliği gelişen teknolojiyle elde edilen büyük veri yığınlarından anlamlı bilgiler çıkartmak amacıyla gündeme gelen önemli kavramlardan birisidir. Veri madenciliğinin birçok tanımı bulunmaktadır. Genel olarak elde edilen birçok veriden, öncesinde bilinmeyen, gizli olan, sezgi veya tecrübe ile tahmin edilemeyen verilerden anlamlı bilgilere ulaşılarak, belirli kalıplara göre karar verilebilmesi için işlenmesi anlamına gelmektedir (Silahtaroğlu, 2016). Veri madenciliği belirlenen veri üzerinde; sınıflama, kümeleme, birliktelik kuralları vb. yöntemler ve bu yöntemlere uygun algoritmalar kullanılarak çeşitli örüntülerin elde edilmesi maksadıyla veri tabanından bilgi keşfi sürecinin bir aşamasıdır (Demir & Dinçer, 2020).

Metin Madenciliği Nedir?

Dünyada her yeni günde %80'inin metin formalarından oluştuğu yaklaşık olarak 2,5 milyar GB veri üretildiği tahmin edilmektedir. Bu üretilen veride doğal olarak internetin son zamanlarda çok yaygınlaşması (sosyal medyalar, bulut depolama çözümleri vb.) da sebep olarak incelenebilmektedir. İnsanoğlu veriyi, kendisinin anlayacağı en kolay şekil olan genellikle doğal dil ile kaydetmeyi tercih etmektedir (Atan, 2020).

Metin madenciliği veri madenciliğinin bir alt dalı olarak madencilik işleminin daha çok metinsel ifadelerle yapılması üzerine çalışmaktadır. Veri tipi metin olan değerlerin, metin madenciliği kullanılarak özel olarak incelenip ve yorumlanması bu alanın ortaya çıkmasına neden olmuştur (Atan, 2020). Metin madenciliği hacimsel olarak büyük verilerin bulunduğu, büyük veri yığınları arasından anlamlı yapıların ortaya çıkarıldığı ve bu anlamlandırma sürecinde birçok farklı uygulama ve analizin kullanıldığı yapıların bütünüdür. Metin madenciliği büyük veri içindeki özelliklerin ortaya çıkarılması, kümelenmelerin sağlanması, sınıflandırmaların gerçekleştirilmesi, özetlemelerin sağlanması, trend analizinin ortaya çıkarılması ve görselleştirmelerin yapılması amacıyla kullanılan temeli anahtar kelimelerin kaç kez tekrar ettiği bağlamında frekansın sunulduğu uygulamalardır (Artsın, 2020). Bu bilgiler ışığında metin madenciliğinin yapılandırılmamış metinlerden ilgi çekici bilgi veya iç görü elde etme süreci olduğu görülmektedir (Chen, 2011).

Metin madenciliği daha çok yapılandırılmamış veriler üzerinde yoğun olarak çalışmaktadır. Gelen verinin belirli bir standardı yoktur. Bu aşamada örnek olarak belirli mail yazışmalarında, raporlarda, haberlerde, elektronik belge sistemlerindeki yazışmalarda konuların saptanmasının metin madenciliği ile gerçekleştirilebileceği düşünülebilir. Bu belgelerdeki veriler doğal dil kullanılarak yazılmıştır belirli bir şablonu standardı veri tipi yoktur, bu yüzden verileri yapılandırılmış veri gibi bir analize sokmak çok mümkün değildir. Dünya genelindeki yapılandırılmamış veriler (ses, video, fotoğraf, metinsel ifadeler), tüm verilerin %80'ine karşılık geldiği ortaya atılmaktadır (Gupta & Lehal, 2009).