Machine Learnig Algorithm ,Deep Learing And CNN

Hazırlayan:Levent Elçiçek

İçerik

Machine Learning Algorithm

Deep Learning

CNN Model

13.08.2024

Machine Learning Algorithm Tanımı

Makine Öğrenmesinin Temel Prensibi

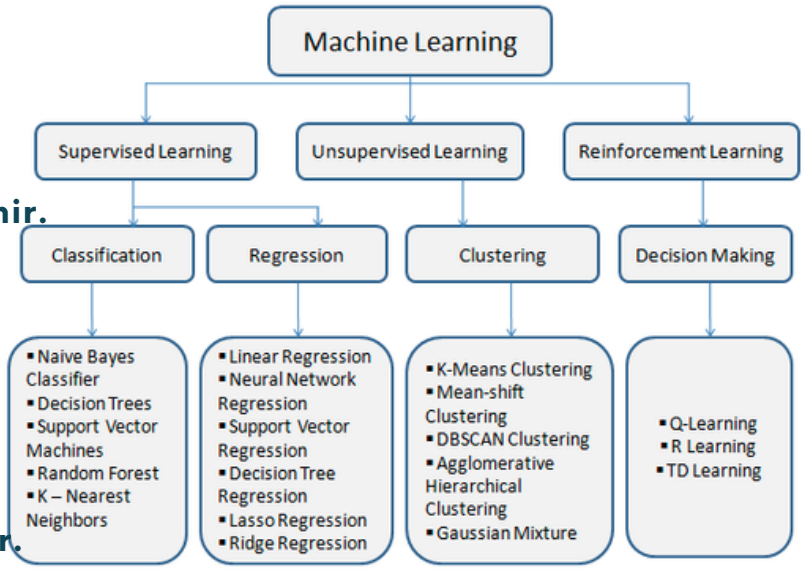
Geleneksel yazılım geliştirmede, bir bilgisayara belirli görevleri yerine getirmesi için açık talimatlar verilir. Örneğin, bir e-ticaret sitesinde arama yapmak için bir algoritma yazıldığında, bu algoritma belirli kurallar ve koşullarla tanımlanır. Ancak makine öğrenmesi bu yaklaşımı tersine çevirir. Bunun yerine, bilgisayarın belirli görevleri gerçekleştirebilmesi için gerekli bilgiyi ve kuralları veriden öğrenmesine olanak tanır.(1)

Verilerden Öğrenme Süreci

Makine öğrenmesi algoritmaları, büyük miktarda veri üzerinde çalışarak model adı verilen matematiksel yapılar oluşturur. Bu modeller, verinin yapısını anlamak, kalıpları tanımak ve bu kalıplar üzerinden genelleme yapmak için kullanılır. Örneğin, bir makine öğrenmesi modeli bir dizi resim üzerinde eğitildiğinde, resimlerdeki belirli özellikleri (örneğin, şekiller, renkler, desenler) öğrenir ve bu bilgilere dayanarak daha önce görmediği yeni resimleri tanıyabilir.

Öğrenme Türleri (1)

Makine öğrenmesi, öğrenme sürecini farklı şekillerde gerçekleştirir: Görsel 0.0



Görsel 0.0 (1)

-> Denetimli Öğrenme:
Model, etiketlenmiş veriler üzerinde eğitilir. Yani, her girdiye karşılık gelen doğru çıktı bilinir ve model, bu eşleşmelerden öğrenir. Örneğin, bir model, bir resim setindeki her bir resmin köpek mi kedi mi olduğunu öğrenir ve daha sonra yeni bir resmi sınıflandırabilir.

-> Denetimsiz Öğrenme:
Model, etiketlenmemiş veriler üzerinde çalışır. Yani, veriler arasında doğal olarak oluşan kalıpları ve grupları bulmaya çalışır. Örneğin, müşteri verilerini analiz ederek müşteri segmentlerini belirleyebilir.

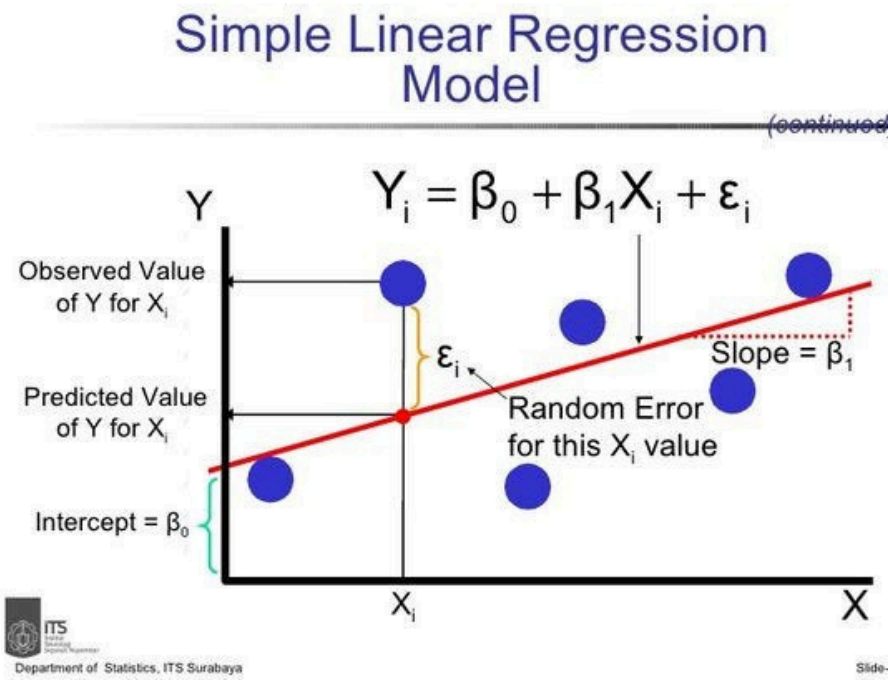
-> Pekiştirmeli Öğrenme: Bir ajan (bilgisayar), bir ortamda hareket eder ve eylemlerinin sonucunda ödül veya ceza alır. Amaç, maksimum ödülü elde etmek için en iyi stratejiyi öğrenmektir. Bu yaklaşım, oyun oynama ve robotikte yaygın olarak kullanılır.

1. Doğrusal Regresyon (Linear Regression):

Tanım: Doğrusal regresyon, bağımlı bir değişken ile bir veya daha fazla bağımsız değişken arasındaki doğrusal ilişkiyi modellemek için kullanılır. Model, bağımsız değişkenlerin bir kombinasyonu olarak bağımlı değişkenin tahmin edilmesine olanak tanır.

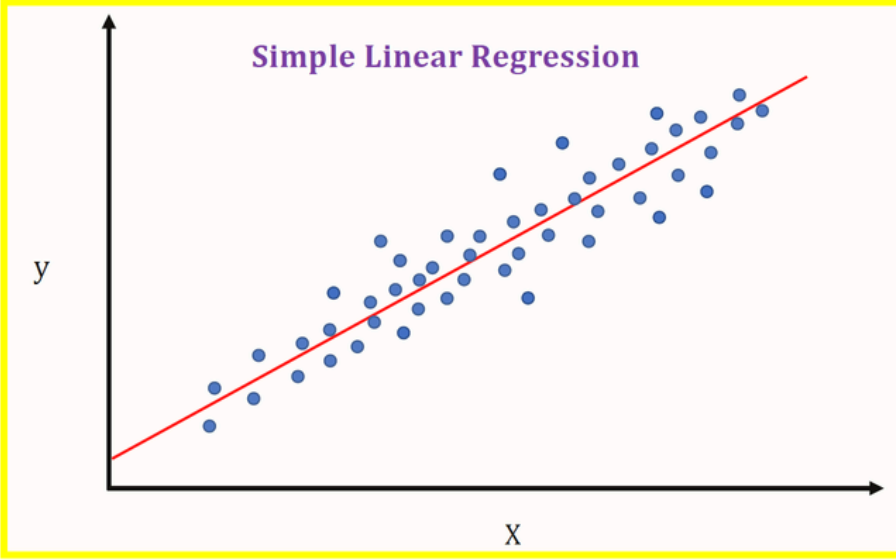
Örnek Kullanım: Ev fiyatlarını tahmin etmek için evin büyüklüğü, oda sayısı gibi özellikler kullanılarak bir model oluşturulur. Bu model, bu özelliklere dayanarak ev fiyatlarını tahmin eder. (2)

Doğrusal regresyonun temel amacı, bağımlı değişken (y) ile bir veya daha fazla bağımsız değişken (x) arasındaki ilişkiyi modellemektir. Bu ilişkiyi tanımlamak için kullanılan matematiksel formül şudur: Görsel 0.1



Görsel 0.1

Doğrusal Lineer regresyon algoritması çalıştırıldığında Görsel 0.2 deki gibi bir grafik elde ederiz



Görsel 0.2

2. Neural Network Regression:

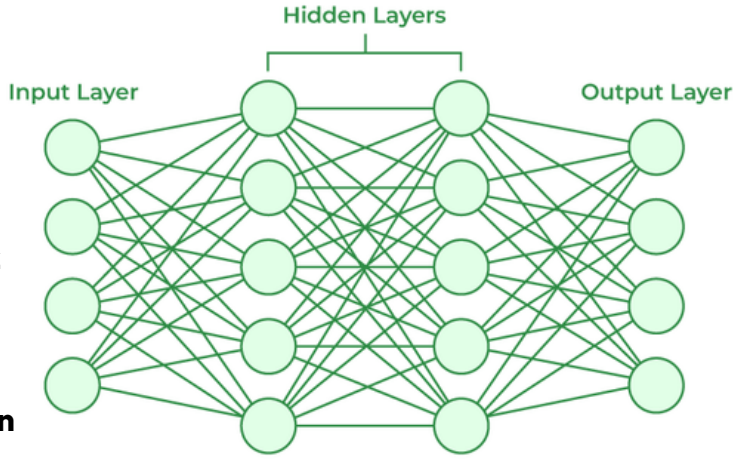
Tanım:Neural network regresyonu sinir ağları (neural networks) kullanarak sürekli (sayısal) verileri tahmin etmeye yönelik bir yaklaşımdır. Geleneksel doğrusal regresyondan farklı olarak, sinir ağları doğrusal olmayan ve karmaşık ilişkileri modellemek için kullanılır. Bu da, özellikle veri setindeki karmaşık kalıpları yakalamada ve yüksek boyutlu verilerle çalışmada daha etkili olmalarını sağlar. (2)

Neural Network Regresyonunun Temel Bileşenleri (2)

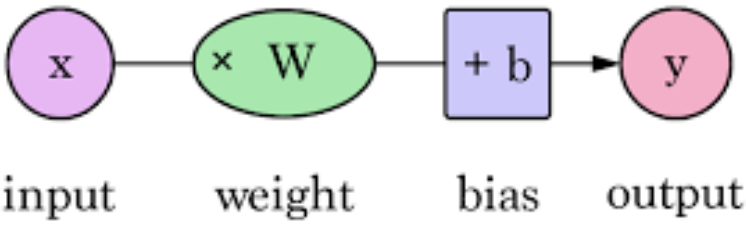
-->Giriş Katmanı (Input Layer):
Girdi verisinin model tarafından alındığı katmandır.
Bu katmandaki her nöron (düğüm), bir bağımsız değişkeni temsil eder.

-->Gizli Katmanlar (Hidden Layers):
Giriş katmanıyla çıkış katmanı arasında bulunan katmanlardır.
Her bir gizli katman, girdi verilerinin doğrusal olmayan dönüşümlerini gerçekleştirir.
Sinir ağı, birden fazla gizli katmana sahip olabilir; bu katmanlar ağırlıklar ve aktivasyon fonksiyonları ile işlev görür.

-->Çıkış Katmanı (Output Layer):
Modelin tahmin ettiği sonuçların çıktısını verir.
Regresyon probleminde genellikle tek bir nöron bulunur ve bu nöron, sürekli bir değer döner.
(Görsel 0.3)



Görsel 0.3



Görsel 0.4

Ağırlıklar (Weights) ve Biaslar (Biases): (2)
Her bir bağlantı üzerinde, girdiyi dönüştüren bir ağırlık değeri bulunur.
Bias terimi, her nörona eklenen sabit bir değerdir ve modelin daha esnek olmasını sağlar.

Aktivasyon Fonksiyonları (Activation Functions):
Gizli katmanlarda, girdinin doğrusal olmayan bir dönüşüme uğramasını sağlar.
ReLU (Rectified Linear Unit), sigmoid, tanh gibi çeşitli aktivasyon fonksiyonları kullanılır.

Görsel 0.4

3. Support Vector Regression (SVR):

Support Vector Regression (SVR), destek vektör makineleri (Support Vector Machines, SVM) yönteminin regresyon problemleri için uyarlanmış bir versiyonudur. SVR, doğrusal ve doğrusal olmayan regresyon problemlerini çözmek için kullanılabilir ve özellikle yüksek boyutlu ve karmaşık veri setlerinde etkili olabilir.
Veriyi sınıflandırmak için en iyi hiper-düzlemi bulmaya çalışır.
İkili sınıflandırma problemlerinde sıklıkla kullanılır. (4)

SVR'nin Matematiksel Tanımı
SVR, doğrusal regresyonu şu şekilde tanımlar: Görsel 0.5

$$\hat{y} = \begin{cases} 0 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b < 0, \\ 1 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases}$$

Görsel 0.5

Burada:
w: Ağırlık vektörü
x: Girdi özellikleri
b: Bias terimi

SVR'nin hedefi, hatayı belirli bir ε-tolerans içinde tutarak en iyi uygunluğu sağlamaktır. Matematiksel olarak, SVR'nin optimizasyon problemi şu şekilde tanımlanabilir: (Görsel 0.6)

$$\text{Minimize} \quad \frac{1}{2}\|w\|^2$$

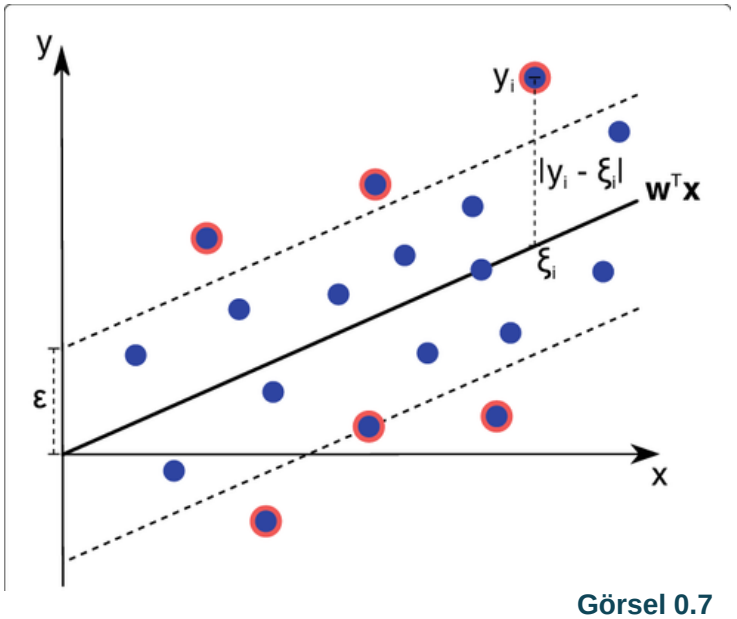
$$\text{subject to} \quad |y_i - (w^T x_i + b)| \leq \epsilon + \xi_i$$

$$\xi_i \geq 0$$

Görsel 0.6

Burada:
y_i: Gerçek değer
xi_i: Hata terimleri (slack variables),
ε-tolerans sınırının ötesindeki hatayı temsil eder.

SVR algoritması çalıştırıldığında Görsel 0.7 **deki gibi bize bir grafik sunar.**



Görsel 0.7

4. Decision Tree Regression :

Decision Tree Regression, sürekli hedef değişkenleri tahmin etmek için kullanılan bir karar ağaçları yöntemidir. Karar ağacı, veri setini özelliklere dayalı olarak dallara ayırarak, her bir dalda tahminler yapar. Sonuç olarak, ağaç yapısındaki yapraklar, hedef değişkenin tahmin değerlerini temsil eder. (5)

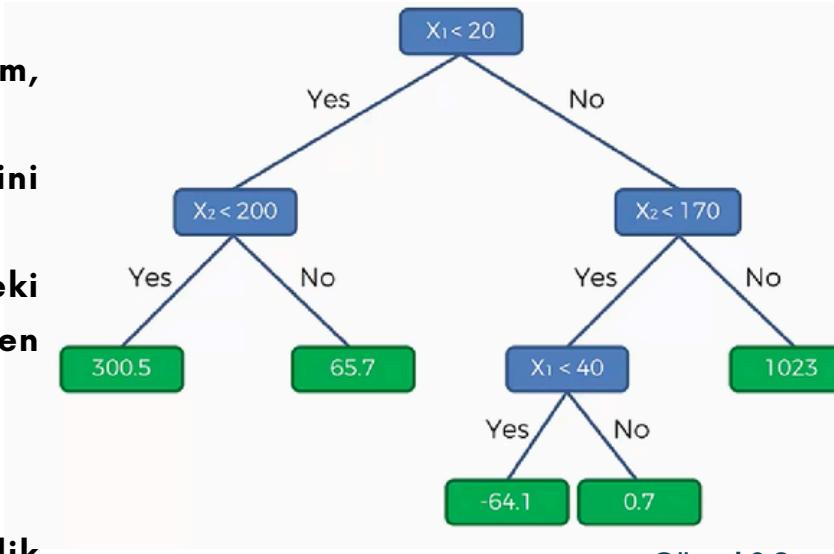
Karar Ağacı Regresyonunun Temel Özellikleri

Ağaç Yapısı:

Kök Düğüm (Root Node): Ağaçtaki ilk düğüm, veri setini ilk bölümlere ayırır.

İç Düğümler (Internal Nodes): Veri setini çeşitli özelliklere göre ayıran düğümlerdir.

Yaprak Düğümler (Leaf Nodes): Veri setindeki final tahmin değerlerini temsil eden düğümlerdir. (5)



Görsel 0.8

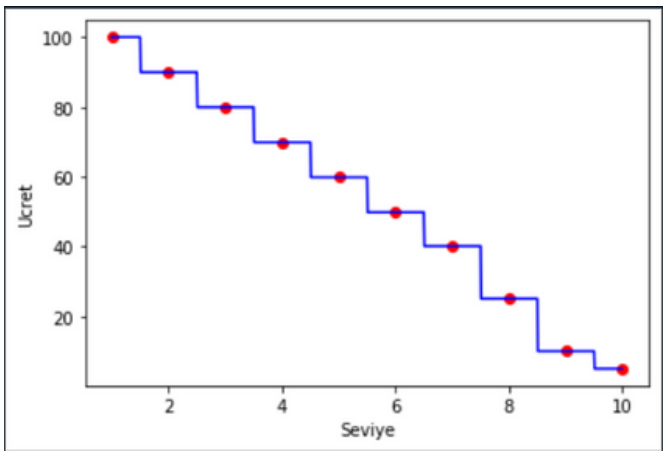
Bölme (Splitting):

Ağaç, her düğümde veriyi belirli bir özellik üzerinde iki veya daha fazla gruba böler. Bu bölme işlemi, her grubun homojenliğini artıracak şekilde yapılır.

Genelleştirme ve Aşırı Öğrenme:

Karar ağaçları genellikle aşırı öğrenme (overfitting) riskine sahip olabilir. Bu nedenle, ağaçları kesmek (pruning) veya maksimum derinlik gibi hiperparametrelerle sınırlamak yaygın bir uygulamadır. (5)(2)

Decides Tree Regression Algoritması çalıştırıldığında bizlere değerlerin kesin hatlarla ayrılmış grafiğini sunar. Görsel 0.9



Görsel 0.9

5. Naive Bayes Classifier:

Naive Bayes algoritması, makine öğrenmesinde sınıflandırma problemlerini çözmek için kullanılan etkili bir olasılık tabanlı algoritmadır. Bu algoritma, Bayes teoremine dayanır ve veri kümesindeki özellikleri kullanarak bir örneğin hangi sınıfa ait olduğunu tahmin eder.Her özelliğin, diğer özelliklerden bağımsız olduğunu kabul eder. Bu, gerçek hayatta her zaman geçerli olmasa da, Naive Bayes'in genellikle oldukça iyi sonuçlar verdiği görülmüştür. (2)

Naive Bayes, Bayes Teoremi'ne dayanır. Bayes Teoremi şu şekilde ifade edilir: Görsel 0.9

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

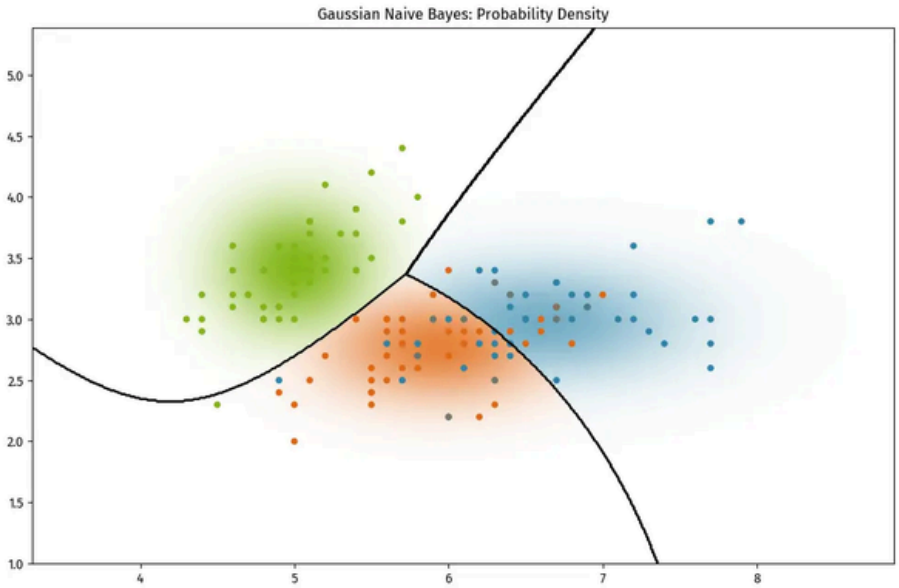
Görsel 0.9

Burada:
P(A\B): B olayı gerçekleştiğinde A olayının gerçekleşme olasılığıdır (posterior olasılık).
P(B\A): A olayı gerçekleştiğinde B olayının gerçekleşme olasılığıdır (likelihood).
P(A):A olayının gerçekleşme olasılığıdır (prior olasılık).
P(B):B olayının gerçekleşme olasılığıdır (evidence).

Naive Bayes Türleri: (2)

-->Gaussian Naive Bayes: Sürekli veri ile çalışır ve her özelliğin normal dağıldığını varsayar.
-->Multinomial Naive Bayes: Belirli olayların (örneğin, kelime sayısı) sayımı ile çalışır ve metin sınıflandırmasında sıklıkla kullanılır.
-->Bernoulli Naive Bayes: İkili (binary) veri ile çalışır. Özelliklerin sadece iki değer alabileceğini varsayar (örneğin, kelimenin varlığı veya yokluğu).

Naive Bayes algoritması çalıştırıldığında verileri olduğu gibi farklı reklerle gruplandırır Görsel 1.0



Görsel 1.0

6. Random Forest Classifier:

Random Forest, makine öğreniminde sıklıkla kullanılan güçlü ve esnek bir denetimli öğrenme (supervised learning) algoritmasıdır. Hem sınıflandırma hem de regresyon problemlerinde kullanılır. Temel olarak, Random Forest birden fazla karar ağacının (decision tree) bir araya getirilmesiyle oluşur ve bu ağaçların tahminlerini birleştirerek (ensemble) daha doğru ve genelleştirilebilir sonuçlar elde eder. (6)

Görsel 1.1

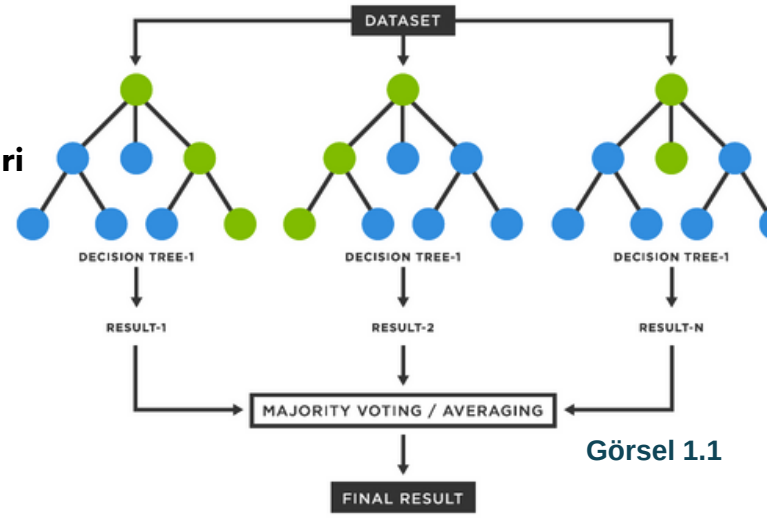
Random Forest'in Temel Mantığı:

Karar Ağaçları:

Karar ağaçları, veri setini farklı özelliklere(features) göre dallara ayıran ve her bir yaprakta (leaf node) bir karar (sınıf) veya sürekli bir değer (regresyon) veren, ağaç benzeri yapılardır. Ancak, tek bir karar ağacı aşırı uyum (overfitting) yapabilir ve eğitildiği veri setine çok fazla uyum sağlayabilir, bu da genelleme yeteneğini düşürür.

Bagging (Bootstrap Aggregating):(6)

Random Forest, veri setinden birçok farklı alt veri seti (bootstrap örnekleri) oluşturur ve her bir alt veri seti üzerinde bir karar ağacı eğitir.
Bu işlem, her bir ağacın farklı bir alt veri setiyle eğitilmesini sağlar ve modelin çeşitliliğini artırır.



Görsel 1.1

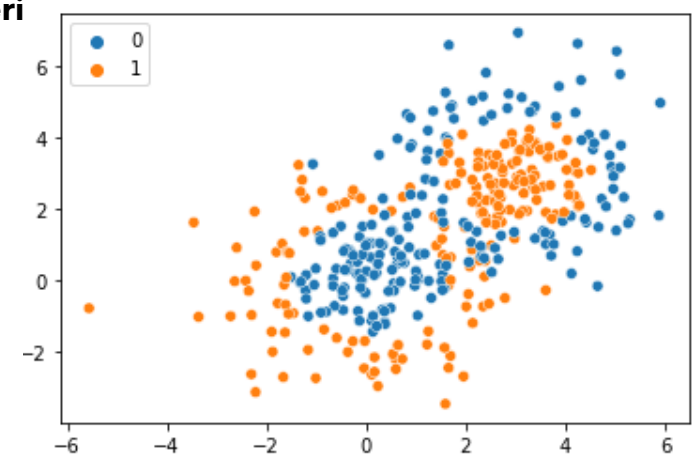
Rastgele Özellik Seçimi: (6)

Her karar ağacı oluşturulurken, her bir düğümde (node) veri setinin tamamı yerine sadece rastgele seçilmiş bir özellik alt kümesi (subset of features) üzerinde bölme yapılır.
Bu da, ağaçlar arasındaki korelasyonu azaltır ve modelin genelleme yeteneğini artırır.

Tahmin ve Birleştirme: (6)

Sınıflandırma probleminde, her bir ağacın tahmini alınır ve en çok oy alan sınıf (majority voting) final tahmini olarak seçilir.
Regresyon probleminde ise, her bir ağacın tahminleri olarak kullanılır.

Random Forest Classifier algoritmasının çıktısı şekil Görsel 1.2 deki gibidir.



Görsel 1.2

7. K-Nearest Neighbors (k-NN) :

K-Nearest Neighbors (k-NN) algoritması, hem sınıflandırma hem de regresyon problemlerinde kullanılan basit ve sezgisel bir makine öğrenimi algoritmasıdır. K-NN algoritması, yeni bir veri noktasının sınıfını veya değerini, bu noktaya en yakın olan k komşusunun sınıflarına veya değerlerine göre tahmin eder. (6)

K-NN Algoritmasının Temel Mantığı: (6)

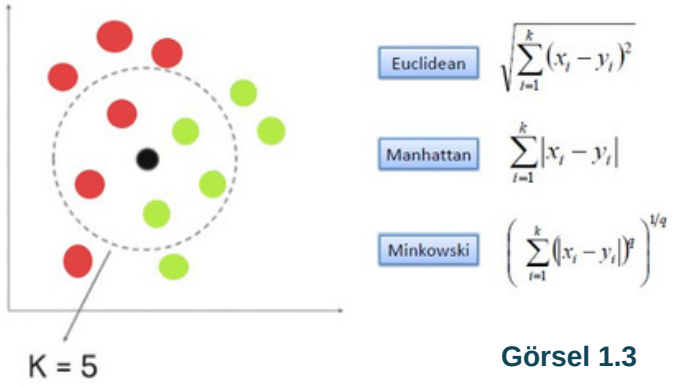
Veri Noktalarının Mesafesi:

K-NN algoritması, bir veri noktasının komşularını belirlemek için mesafe ölçümlerini kullanır. En yaygın kullanılan mesafe ölçümleri şunlardır:

Öklidyen Mesafe (Euclidean Distance): İki nokta arasındaki düz bir hattaki mesafeyi ölçer.

Manhattan Mesafesi (Manhattan Distance): İki nokta arasındaki mesafeyi, dik açılı bir ızgara boyunca ölçer.

Minkowski Mesafesi: Genel bir formül olup, hem Öklidyen hem de Manhattan mesafesini kapsar.



Görsel 1.3

k Değeri: (6)

Algoritma, yeni bir veri noktasının sınıfını veya değerini belirlerken, ona en yakın k tane komşunun sınıfını veya değerini göz önünde bulundurur.

kdeğeri, modelin karmaşıklığını ve genelleştirme yeteneğini etkiler:

Küçük bir k değeri (örneğin k=1), modelin aşırı uyum (overfitting) yapmasına neden olabilir.

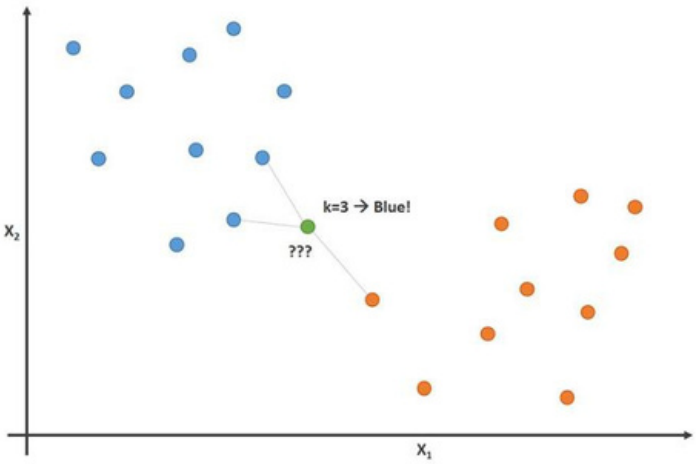
Büyük bir k değeri, modelin genelleme yeteneğini artırır, ancak çok büyük bir k değeri, modelin performansını olumsuz etkileyebilir.

Sınıflandırma: (6)

Sınıflandırma problemi için, K-NN algoritması, bir veri noktasını en çok oy alan sınıfa atar. Örneğin, eğer k=5 ve komşuların 3'ü sınıf A'ya, 2'si sınıf B'ye aitse, yeni nokta sınıf A'ya atanır (majority voting).

Regresyon:

Regresyon problemi için, K-NN algoritması, komşuların ortalama değerini alarak yeni veri noktasının değerini tahmin eder.



Görsel 1.4

Görsel 1.4

Denetimsiz Öğrenme (Unsupervised Learning)

K-Means Clustering:

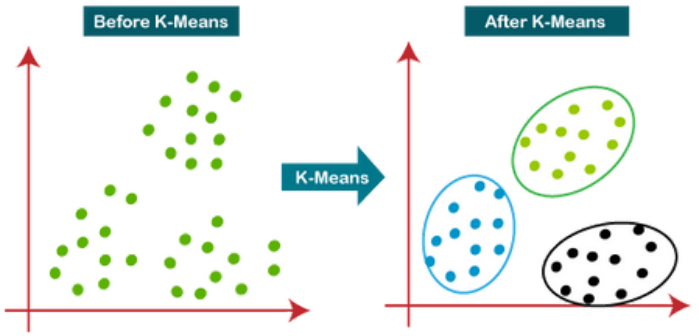
K-Means Clustering, denetimsiz öğrenme (unsupervised learning) yöntemlerinden biri olan ve verileri önceden tanımlanmamış gruplara (kümelere) ayırmak için kullanılan bir kümeleme algoritmasıdır. K-Means, veri setini K sayıda küme veya grup içine böler ve her bir küme için bir "merkez" veya "centroid" belirler.(6)

K-Means Algoritmasının Temel Mantığı: (6)

Başlangıç Centroidleri Seçimi:

Algoritma dirsek yöntemi ile belirlenen K sayıda küme için rastgele veya belirli bir stratejiye göre başlangıç merkezlerini (centroid) seçer.

Bu merkezler, başlangıçta rastgele veri noktaları olabilir.



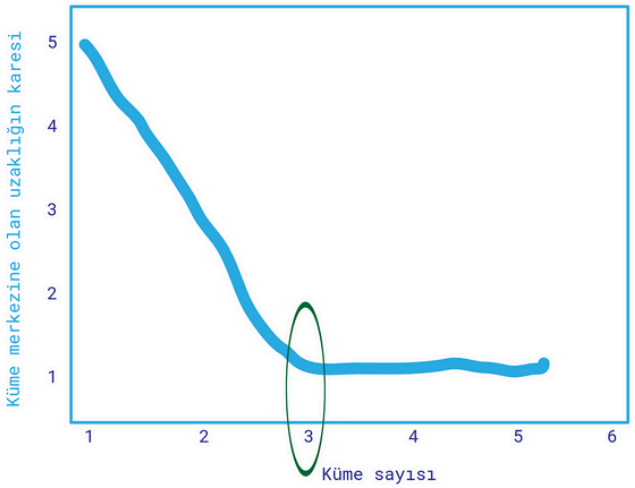
Görsel 1.5

Görsel 1.5

Dirsek Yöntemi Nedir? (7)

Dirsek yöntemi, K-means algoritmasında en uygun küme sayısını (K) belirlemek için kullanılan bir yöntemdir. Amaç, K değerini seçerken, kümeler arasındaki ayrışmayı en iyi şekilde sağlayan ve veri setini en iyi temsil eden K sayısını bulmaktır.

Görsel 1.6



Görsel 1.6

Atama Adımı (Assignment Step): (7)

Her veri noktası, en yakın merkeze (centroid) atanır. "En yakın" burada genellikle Öklidyen mesafe ile ölçülür, ancak diğer mesafe ölçümleri de kullanılabilir.

Böylece her veri noktası, K kümelerden birine atanmış olur

Merkezleri Güncelleme (Update Step): (7)

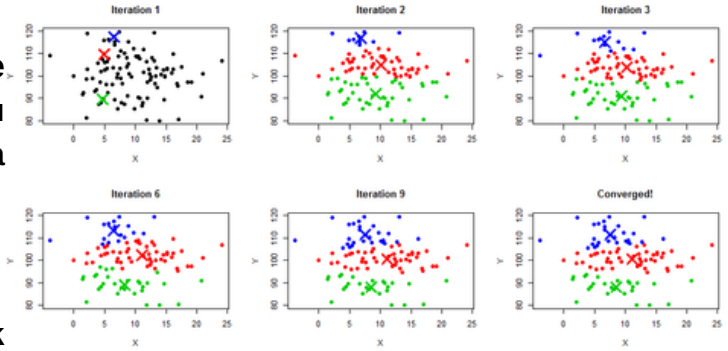
Her bir küme için yeni merkezler hesaplanır. Bu, kümedeki tüm noktaların ortalaması alınarak yapılır.

Bu yeni merkezler, mevcut kümelerin gerçek merkezi olarak işlev görür.

Tekrar Et (Repeat): (7)

Atama ve güncelleme adımları, merkezler stabil hale gelene kadar, yani veri noktalarının küme atamaları değişmeyene kadar veya belirli bir iterasyon sayısına ulaşana kadar tekrar edilir.

Görsel 1.6



Görsel 1.6

Durma Kriteri: (7)

Algoritma, merkezlerde artık anlamlı bir değişiklik olmadığında veya belirli bir iterasyon sayısına ulaşıldığında durur.

Öğrenme Süreci ve Karşılaştığım Zorluklar

Öncelikle Kendi yoğunlaşmaya karar verdiğin sektörden farklı bir sektöre geçiş yapmak herkes için biraz yorucu ve kafa karıştırıcı olabilir. Ancak her şeyin çözümü çok çalışmak.

Öncelikle nereden başlayacağını bilememek biraz boşluğa düşmememe sebep oldu. Birçok farklı kaynak ve yöntem varken olayı en doğru şekilde öğrenmeye çalışmak biraz beklememe yol açtı.

İlk olarak yazacağım kodlar için gerekli olan bazı sistem gereksinimlerini, kendi cihazım tarafından karşılayamamak bazı sorunlara yol açtı.
Cihazımın donanımından tam olarak haberdar olmamam da biraz zaman kaybına sebep oldu.
Bununla beraber veri analizi ,veri görselleştirme için kullanılan birçok kütüphane ile ilk kez tanıştım. Öğrenme sürecimde başlarda ezber mantığı ile hızlıca yol alma fikri doğsa da bu zamanla yerini algoritmik mantığa bıraktı.
Algoritmaların kodsal mantığını inceledikçe edindiğim bilgiler bir sonuca varmaya başladı.
Konuyu ve kavramları öğrendikçe kendi modellerimi oluşturup farklı veri setleri üzerinde uygulamaya başladım.

Sürecin devamında tekrar tekrar farklı problemler oluştu ,ancak araştırarak veya fikir alarak bu sürecini de iyi bir şekilde atlattığımı düşünüyorum.

Deep Learning Tanımı

Deep learning (derin öğrenme), yapay sinir ağlarını kullanarak verilerden yüksek seviyede özellikler öğrenmeyi amaçlayan bir makine öğrenimi alt alanıdır. Derin öğrenme, özellikle büyük ve karmaşık veri setleri üzerinde çalışmak için kullanılan çok katmanlı sinir ağlarına dayanır. İşte derin öğrenmenin temel unsurları: (2)

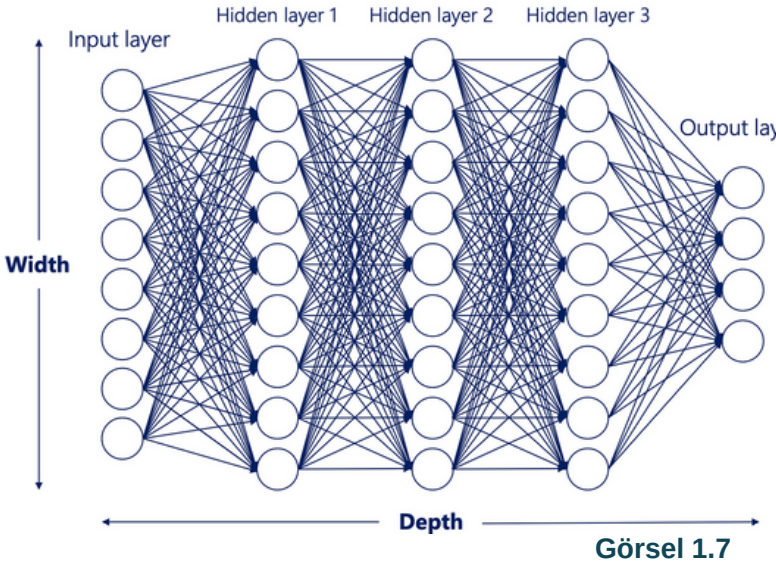
Sinir Ağları (Neural Networks) (2)

Derin öğrenme, yapay sinir ağları üzerine kuruludur. Bir sinir ağı, genellikle katmanlar halinde düzenlenmiş nöronlardan (düğümlerden) oluşur. Bu nöronlar, biyolojik sinir hücrelerinin işleyişini taklit eder.

Giriş Katmanı (Input Layer): Girdileri (verileri) alır.

Gizli Katmanlar (Hidden Layers): Verilerin işlenmesi ve anlamlı temsillerin çıkarılması bu katmanlarda gerçekleşir. Derin öğrenmede, bu gizli katmanlar birden fazla olabilir, bu nedenle "derin" olarak adlandırılır.

Çıkış Katmanı (Output Layer): Sonuçları verir (örneğin, bir sınıflandırma problemi için hangi sınıfa ait olduğunu belirtir).



Görsel 1.7

Görsel 1.7

İleri Besleme (Feedforward) ve Geri Yayılım (Backpropagation) (2)

İleri Besleme (Feedforward): Veriler ağın giriş katmanından başlayarak gizli katmanlara doğru ilerler ve nihayetinde çıkış katmanına ulaşır.

Geri Yayılım (Backpropagation): Ağın çıktısı ile gerçek sonuçlar arasındaki hatayı minimize etmek için ağırlıklar güncellenir. Bu süreç, hata geri yayılımı (backpropagation) adı verilen bir teknikle gerçekleştirilir.

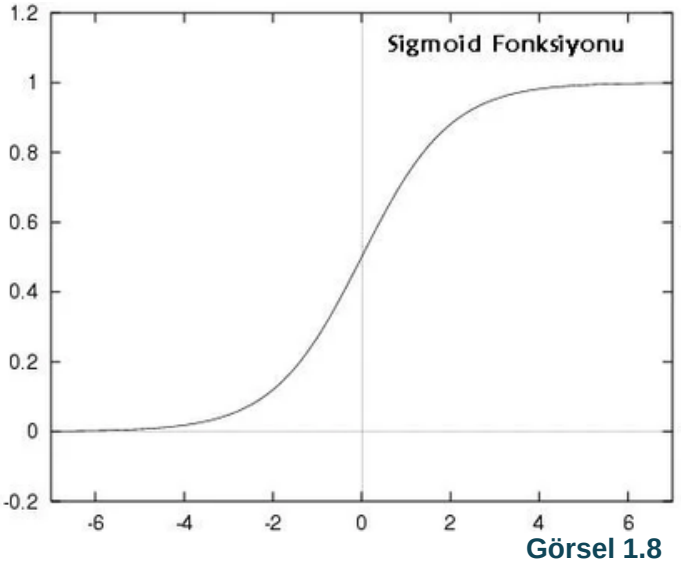
Aktivasyon Fonksiyonları: (2)

Aktivasyon fonksiyonları, bir nöronun çıkışını hesaplamak için kullanılır. Bu fonksiyonlar, ağa doğrusal olmayanlık ekleyerek daha karmaşık fonksiyonları öğrenmesini sağlar. Yaygın aktivasyon fonksiyonları arasında ReLU (Rectified Linear Unit), sigmoid, ve tanh bulunur.

Activation Functions

Sigmoid Fonksiyonu: (8)

Tanım: Sigmoid fonksiyonu, giriş değerini 0 ile 1 arasında bir değere dönüştüren S-şekilli bir eğriye sahiptir. Görsel 1.8



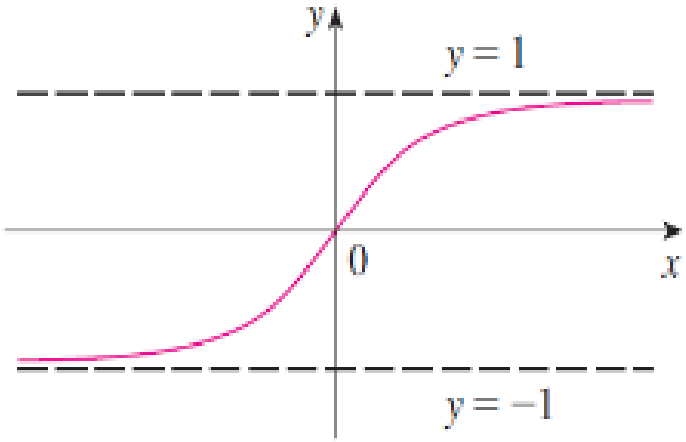
Görsel 1.8

Özellikler:

Çıkış aralığı: (0, 1)
Yaygın olarak çıktı katmanlarında kullanılır (örneğin, ikili sınıflandırma problemlerinde).
Dezavantajı: Büyük veya küçük giriş değerlerinde gradyanların çok küçük olması nedeniyle gradyan kaybolması (vanishing gradient) problemi olabilir.

Tanh (Hiperbolik Tanjant) Fonksiyonu:(8)

Tanım: Tanh, sigmoid fonksiyonuna benzer, ancak çıktılar -1 ile 1 arasında olur. Görsel 1.9



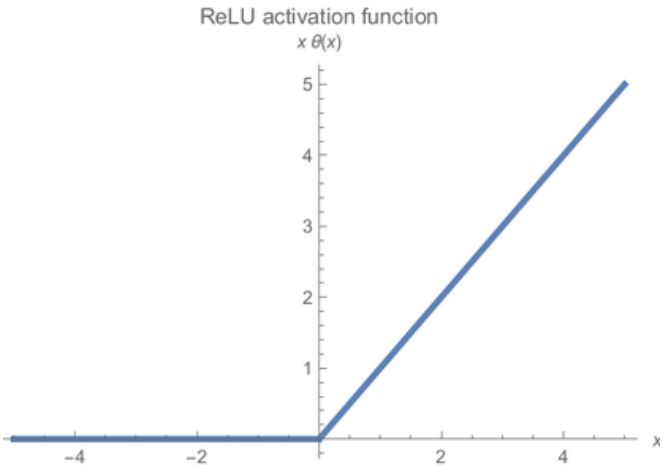
Görsel 1.9

Özellikler:

Çıkış aralığı: (-1, 1)
Daha iyi bir merkezlenmiş çıktı (0 etrafında) sağlar, bu da bazı durumlarda öğrenmeyi hızlandırabilir.
Sigmoid gibi, gradyan kaybolması problemi yaşayabilir.

ReLU (Rectified Linear Unit): (8)

Tanım: ReLU, pozitif girişleri olduğu gibi bırakırken, negatif girişleri 0 yapar. Görsel 2.0



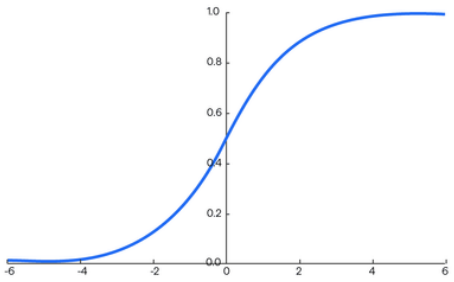
Görsel 2.0

Özellikler:

Çıkış aralığı: [0, sonsuz)
Gradyan kaybolması problemini büyük ölçüde ortadan kaldırır, bu nedenle çok derin ağlarda yaygın olarak kullanılır.
Dezavantajı: Ölü nöronlar (dead neurons) problemi, yani sürekli negatif değerler alan nöronların bir daha asla aktif hale gelmemesi.

Softmax Fonksiyonu: (8)

Tanım: Softmax, çok sınıflı sınıflandırma problemlerinde çıktı katmanında kullanılır ve her sınıf için bir olasılık dağılımı üretir. Görsel 2.1



Görsel 2.1

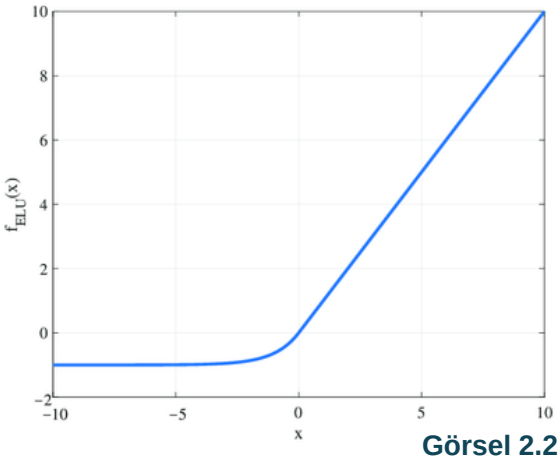
Özellikler:

Çıkış aralığı: (0, 1) ve tüm çıkışların toplamı 1'e eşittir.

Genellikle çok sınıflı sınıflandırma problemlerinde kullanılır.

ELU (Exponential Linear Unit): (8)

Tanım: ELU, negatif değerler için eksponansiyel bir eğriye sahip olup, pozitif değerler için ReLU gibi davranır. Görsel 2.2



Görsel 2.2

Özellikler:

Çıkış aralığı: (-α, sonsuz)

Gradyan kaybolması problemini çözmeye yardımcı olur ve ReLU'ya göre daha iyi öğrenme performansı gösterebilir.

Aktivasyon Fonksiyonlarının Seçimi (8)igmoid ve Tanh:

Genellikle küçük sinir ağlarında veya çıktı katmanında kullanılır. Ancak, büyük ağlar için gradyan kaybolması problemi nedeniyle genellikle tercih edilmez.

ReLU ve Türevleri (Leaky ReLU, ELU):

Derin sinir ağlarında en yaygın kullanılan aktivasyon fonksiyonlarıdır. Gradyan kaybolması problemini azaltır ve hızlı hesaplama avantajı sağlar.

Softmax:

Çıktı katmanında, özellikle çok sınıflı sınıflandırma problemleri için kullanılır.

Deep Learning Models

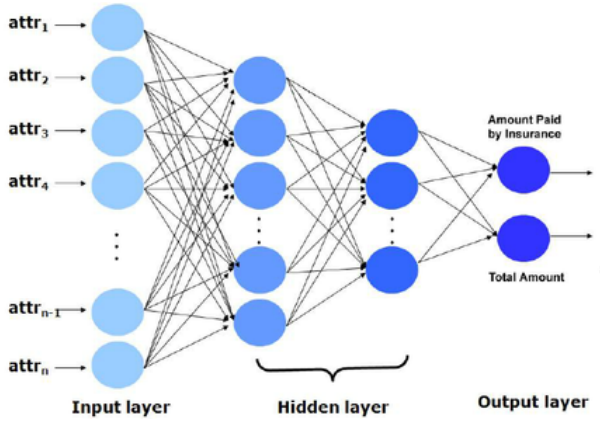
Derin öğrenme modelleri, farklı türde verileri işlemek ve belirli problemleri çözmek için tasarlanmış çeşitli yapay sinir ağı mimarileridir. Her model, belirli veri türlerine ve görev türlerine uygun olacak şekilde tasarlanmıştır. (9)

Yapay Sinir Ağı (Artificial Neural Network - ANN)

Tanım: ANN'ler, en temel sinir ağı yapılarıdır ve bir veya birkaç gizli katmandan oluşur. Her nöron, bir önceki katmandan gelen girdilerle ağırlıklandırılmış toplama işlemi yapar ve aktivasyon fonksiyonu ile çıkış üretir.

Görsel 2.3

Kullanım Alanları: Basit sınıflandırma ve regresyon problemleri, temel veri işleme görevleri. (8)



Görsel 2.3

Evrik Sinir Ağı (Convolutional Neural Network - CNN) (9)

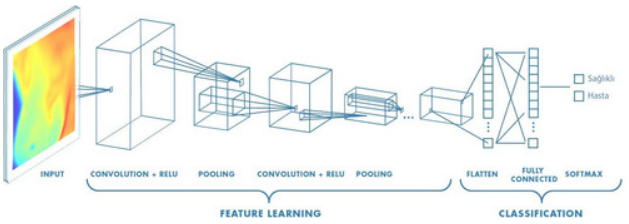
Tanım: CNN'ler, özellikle görüntü verilerini işlemek için tasarlanmıştır. Temel olarak, verilerdeki uzamsal ilişkileri ve kalıpları öğrenmek için evrişim katmanları kullanılır.

Görsel 2.4

Özellikler:

Evrişim Katmanları: Girdiden belirli özellikleri çıkarmak için filtreler (kernels) kullanılır.

Havuzlama Katmanları (Pooling): Boyutları küçülterek ağın hesaplama yükünü azaltır ve önemli özellikleri korur.



Görsel 2.4

Tekrarlayan Sinir Ağı (Recurrent Neural Network - RNN) (9)

Tanım: RNN'ler, sıralı verileri (zaman serileri, metin) modellemek için tasarlanmıştır. Bu ağlar, zaman adımları arasında bilgi taşımak için geri besleme döngüleri kullanır.

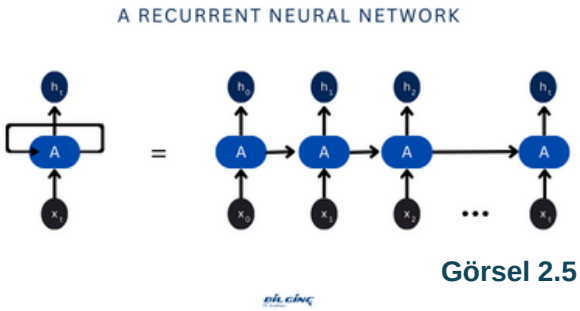
Görsel 2.5

Özellikler:

Bellek (Memory): Geçmiş bilgiyi hatırlayarak sıralı verilerdeki bağıntıları öğrenir.

Dezavantaj: Uzun dizilerde gradyan kaybolması problemi yaşanabilir.

Kullanım Alanları: Doğal dil işleme, konuşma tanıma, zaman serisi tahmini.

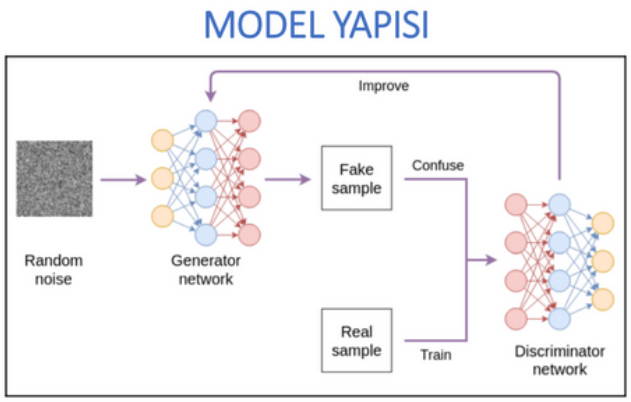


Görsel 2.5

Generative Adversarial Networks (GAN) (9)

Tanım: GAN'lar, iki sinir ağından (üretici ve ayırt edici) oluşur. Üretici ağ, sahte veriler üretir; ayırt edici ağ ise gerçek ve sahte verileri ayırt etmeye çalışır. Bu iki ağ birbirine karşı öğrenir. Görsel 2.6

Kullanım Alanları: Görüntü oluşturma, stil transferi, sahte veri üretimi.

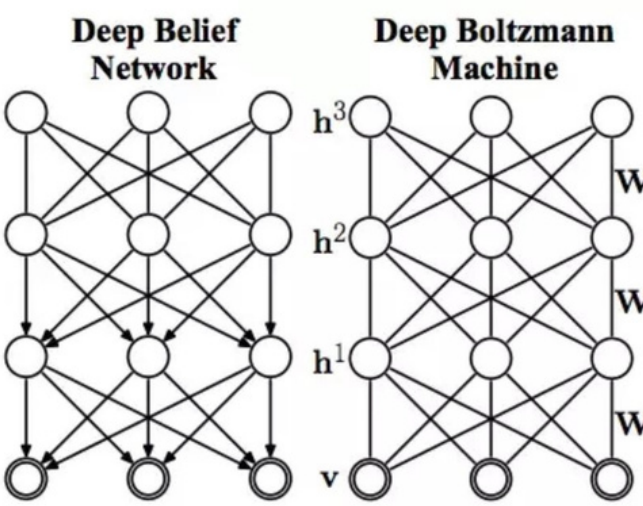


Görsel 2.6

Boltzmann Makinesi (Restricted Boltzmann Machine - RBM) (9)

Tanım: RBM'ler, derin inanç ağları (DBN) gibi unsurların yapı taşı olarak kullanılır ve olasılıksal grafik modellerdir. Öğrenme, verileri temsil eden gizli bir yapı öğrenmeye dayanır. Görsel 2.7

Kullanım Alanları: Boyut indirgeme, özellik çıkarma, öneri sistemleri.



Görsel 2.8

Görsel 2.7

Derin İnanç Ağı (Deep Belief Network - DBN):

Tanım: DBN'ler, birkaç RBM'nin istiflenmesiyle oluşturulan derin öğrenme modelidir. Her RBM, bir sonraki için ön eğitim yaparak daha iyi bir öğrenme sağlar.

Kullanım Alanları: Görüntü tanıma, boyut indirgeme, özellik çıkarma.(9) Görsel 2.8

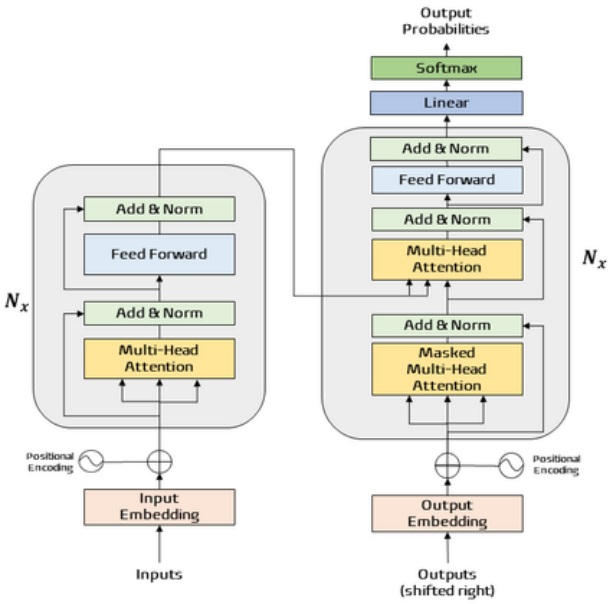
Transformer Ağları: (8)

Tanım: Transformer'lar, özellikle doğal dil işleme görevlerinde yaygın olarak kullanılır. RNN'ler gibi sıralı işlemlere ihtiyaç duymadan, dikkat (attention) mekanizması ile tüm girdi dizisini bir seferde işler. Görsel 2.9

Özellikler:

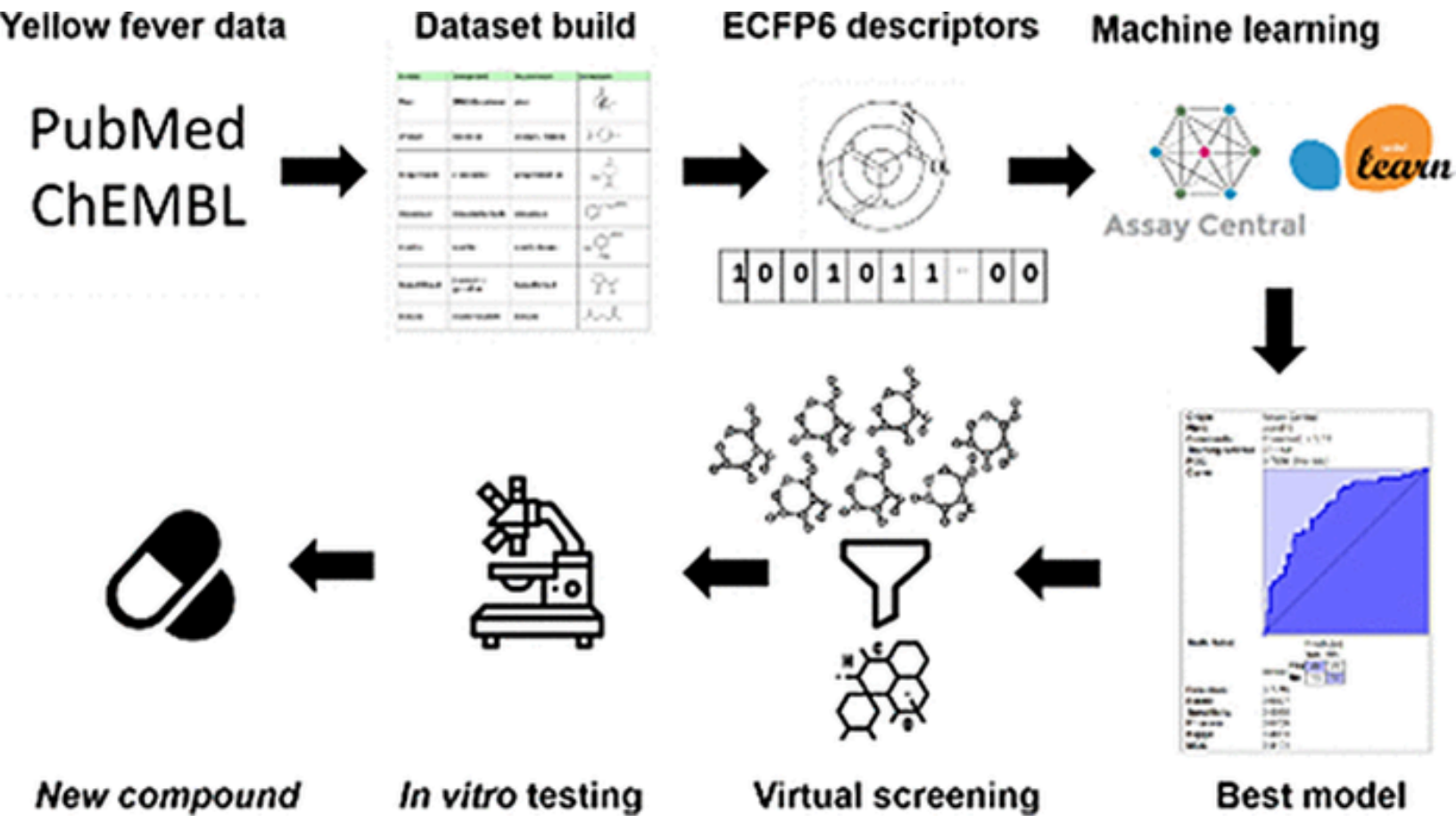
Dikkat Mekanizması: Her kelimenin diğer tüm kelimelerle olan ilişkisini öğrenir, bu sayede uzun bağımlılıkları yakalar.

Kullanım Alanları: Dil modelleme, metin çevirisi, metin sınıflandırma, sohbet botları.



Görsel 2.9

Deep Leaarning Graphicak Abstract



Görsel 3.0

Öğrenme Sürecinde Karşılaştığım Zorluklar

Deep learning konusunu kavramam ve ilerlemem sürecin başlangıcından çok daha hızlı oldu. Yavaş yavaş konulara hakimiyet kazandığımı ve örnek kodlara bakmadan kendi kodlarımı clean bir şekilde yazabildiğimi düşünüyorum.
Ancak kod yazmanın dışında teorik bilgiler e biraz daha ağırlık vermem gerektiğini anladım.

Kodlarımı yazarken küçük otlar tutmaya başladım ve bu şekilde biraz daha teoriye hakim olabileceğimi düşünüyorum.

Her şeye rağmen süreç benim için çok heyecan verici ve umarım heyecanımı hiç kaybetmem.

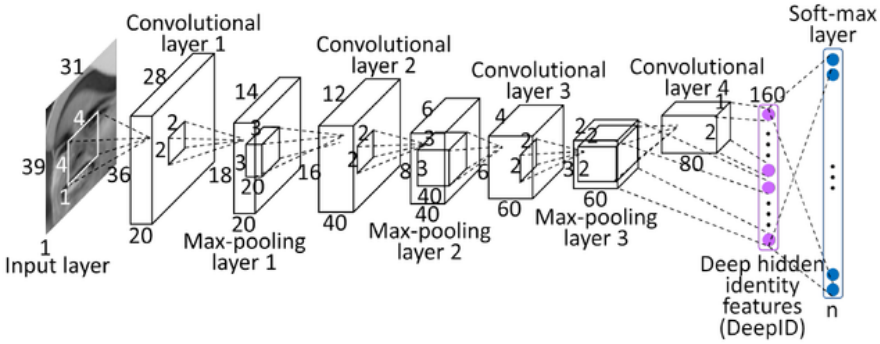
Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) veya Türkçe'de Evrişimli Sinir Ağları, özellikle görüntü verilerini işlemek için tasarlanmış derin öğrenme modelleridir. CNN'ler, verilerdeki uzamsal ve zamansal bağıntıları öğrenme yeteneği sayesinde görüntü tanıma, nesne algılama, yüz tanıma gibi görevlerde son derece başarılıdır. (2)

CNN Mimarisi

CNN'ler genellikle üç ana katmandan oluşur:
Evrişim (Convolutional) Katmanı
Havuzlama (Pooling) Katmanı
Tam Bağlantılı (Fully Connected) Katman

Görsel 3.1



Görsel 3.1

Evrişim Katmanı (Convolutional Layer) (2)

Görev: Evrişim katmanı, girdideki özellikleri çıkarmak için kullanılan temel katmandır.

Nasıl Çalışır:

Filtreler (Kernels): Küçük boyutlu matrislerdir (örneğin, 3x3 veya 5x5). Bu filtreler, görüntünün tamamında kaydırılarak (slide) her bir bölgeden belirli özellikler çıkarır.

Evrişim İşlemi: Filtre, giriş görüntüsünün üzerine yerleştirilir ve eleman bazında çarpım yapılarak sonuçlar toplanır. Bu işlem, görüntü üzerinde belirli bir özelliği (örneğin, kenarlar, köşeler) tespit eder.

Özellik Haritası (Feature Map): Evrişim işlemi sonucunda elde edilen çıktılar, özellik haritaları olarak adlandırılır ve her biri belirli bir özelliği temsil eder.

Özellikler:

Stride: Filtrenin her adımda ne kadar kaydırılacağını belirler.

Padding: Görüntü boyutlarını korumak için kenarlara eklenen sıfırlar. "Same padding" ve "valid padding" olmak üzere iki türü vardır.

Havuzlama Katmanı (Pooling Layer) (2)

Görev: Havuzlama katmanı, özellik haritalarının boyutunu küçültmek (downsampling) ve hesaba katılacak parametre sayısını azaltmak için kullanılır.

Nasıl Çalışır:

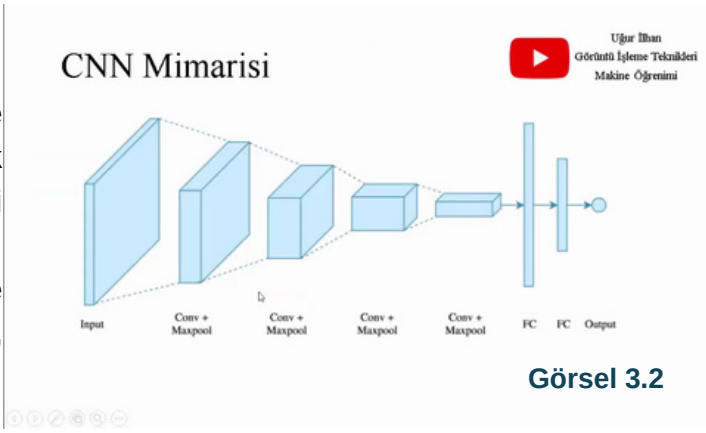
Maksimum Havuzlama (Max Pooling): Küçük bir pencere (örneğin, 2x2) seçilir ve bu pencere içerisindeki en büyük değer alınır. Bu işlem, boyutları küçültür ve önemli özellikleri korur.

Ortalama Havuzlama (Average Pooling): Pencere içerisindeki tüm değerlerin ortalaması alınır. Bu yöntem, daha yumuşak bir küçültme sağlar.

Özellikler:

Havuzlama Boyutu: Pencere boyutu, örneğin 2x2 veya 3x3.

Stride: Pencerenin her adımda ne kadar kaydırılacağını belirler.



Görsel 3.2

Görsel 3.2

Tam Bağlantılı Katman (Fully Connected Layer):(2)

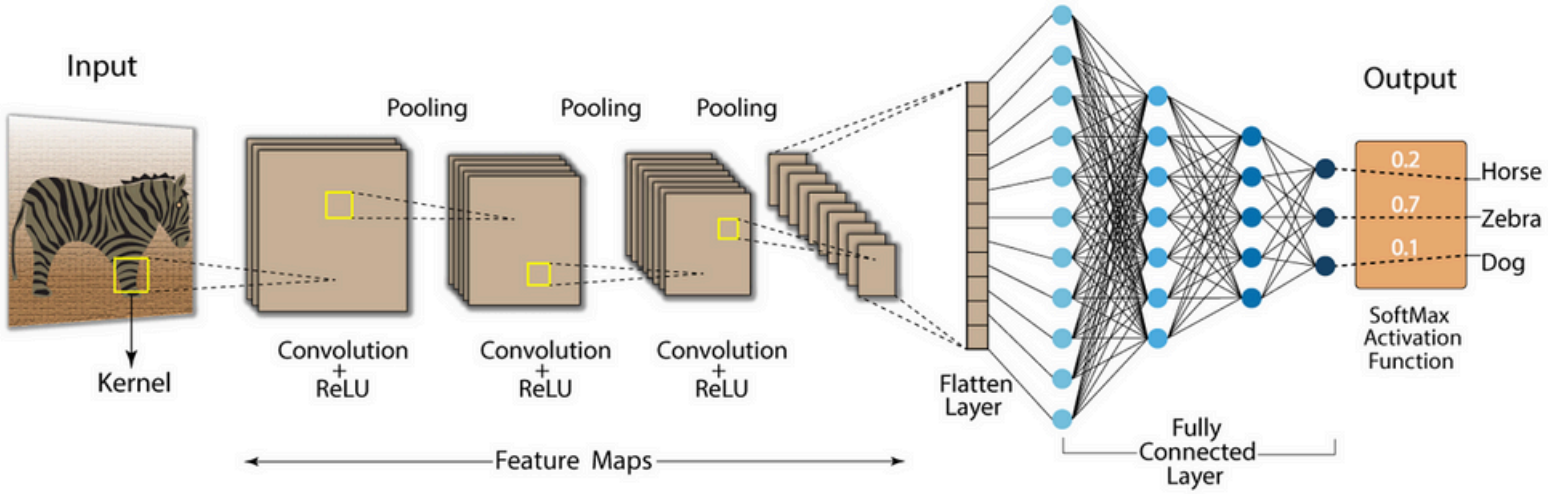
Görev: Tam bağlantılı katman, ağın son katmanıdır ve sınıflandırma veya regresyon gibi çıktıları üretir.

Nasıl Çalışır:

Düzleştirme (Flattening): Havuzlama katmanından gelen özellik haritaları düzleştirilir ve bir vektör haline getirilir.

Bağlantılar: Her nöron, bir önceki katmandaki tüm nöronlarla tam bağlantılıdır. Bu bağlantılar, ağırlıklar ve aktivasyon fonksiyonları ile işlem görür.

Çıkış: Son katmandan elde edilen sonuçlar, genellikle bir softmax aktivasyon fonksiyonu ile sınıflandırılır ve her sınıfın olasılık değerlerini verir. Görsel 3.3



Görsel 3.3

CNN'in Çalışma Prensibi (2)

1, Bir CNN, genellikle yukarıdaki katmanları bir araya getirerek çalışır:

2. Giriş Verisi: Örneğin, 28x28 boyutunda bir gri tonlamalı görüntü.

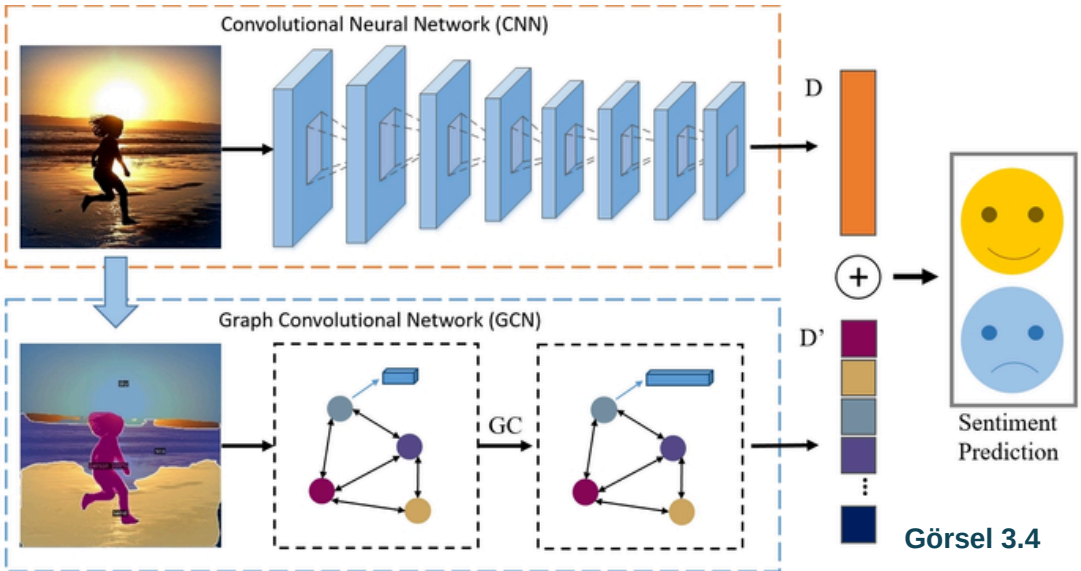
3. Evrişim Katmanı: Farklı filtreler, giriş görüntüsünden çeşitli özellikler çıkarır.

4. Aktivasyon Fonksiyonu: Evrişim sonrası çıkışlar, genellikle ReLU gibi doğrusal olmayan bir aktivasyon fonksiyonundan geçirilir.

5. Havuzlama Katmanı: Boyutlar küçültülür ve önemli özellikler korunur.

6. Tam Bağlantılı Katman: Düzleştirilmiş veriler, sınıflandırma veya regresyon görevini gerçekleştiren bir tam bağlantılı katmana beslenir.

7. Çıktı: Sonuçlar, örneğin hangi sınıfa ait olduğunu gösteren bir olasılık dağılımı olabilir



Görsel 3.4

Hiper Parameters

CNN (Convolutional Neural Networks) modellerinde kullanılan hiperparametreler, modelin performansını ve eğitiminin verimliliğini doğrudan etkileyen önemli ayarlardır. Bu hiperparametrelerin doğru seçilmesi, modelin daha iyi genelleştirme yapmasını ve daha hızlı eğitilmesini sağlar. (3)

1. Filtre Sayısı (Number of Filters): (3)

Tanım: Her bir konvolüsyon katmanında kaç tane farklı filtre (kernels) kullanılacağını belirler. Filtreler, görüntüdeki özellikleri çıkarmak için kullanılır. Daha fazla filtre, daha fazla özellik çıkarılmasına olanak tanır.

Etkisi: Filtre sayısının artırılması, modelin daha karmaşık özellikleri öğrenmesini sağlar, ancak hesaplama maliyetini de artırır.

2. Filtre Boyutu (Filter Size): (3)

Tanım: Her bir filtrenin boyutunu belirler. Örneğin, 3x3 veya 5x5 gibi. Filtre boyutu, bir görüntüdeki komşu piksel ilişkilerini incelemek için kullanılır.

Etkisi: Küçük filtreler (örneğin 3x3), ince detayları çıkarırken, büyük filtreler (örneğin 5x5) daha geniş alanları kapsar.

3. Adım Sayısı (Stride): (3)

Tanım: Filtrenin görüntü üzerinde kaç piksel kayacağını belirler. Örneğin, stride=1 ise filtre her seferinde bir piksel kayar.

Etkisi: Adım sayısının artırılması, çıktı boyutunu küçültür ve hesaplama süresini azaltır, ancak bilgi kaybına yol açabilir.

4. Havuzlama (Pooling) Türü ve Boyutu: (3)

Tanım: Havuzlama katmanında hangi tür havuzlamanın (genellikle MaxPooling veya AveragePooling) ve havuzlamanın boyutunun kullanılacağını belirler. Havuzlama, özellik haritalarının boyutunu küçültmek için kullanılır.

Etkisi: Havuzlama boyutu ve türü, modelin uzamsal bilgileri nasıl koruduğunu ve özetlediğini belirler.

5. Aktivasyon Fonksiyonu (Activation Function): (3)

Tanım: Her katmandaki nöronların çıkışını hesaplamak için kullanılan fonksiyondur. CNN'lerde genellikle ReLU (Rectified Linear Unit) kullanılır.

Etkisi: Aktivasyon fonksiyonu, modelin doğrusal olmayan ilişkileri öğrenmesine olanak tanır.

Öğrenme Oranı (Learning Rate): (3)

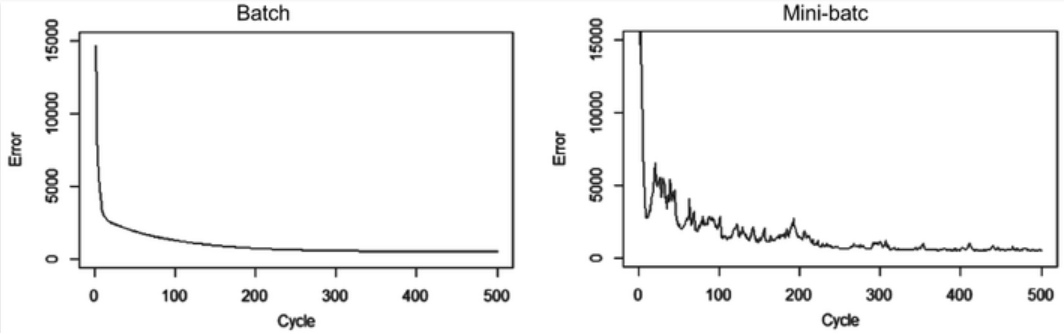
Tanım: Modelin her adımda ne kadar büyük bir güncelleme yapacağını belirler. Gradient Descent gibi optimizasyon algoritmaları için temel bir hiperparametredir.

Etkisi: Öğrenme oranının düşük olması, modelin daha yavaş öğrenmesine neden olurken, yüksek olması modelin dengesiz öğrenmesine yol açabilir.

7. Batch Boyutu (Batch Size): (3)

Tanım: Modeli eğitirken her bir adımda kaç veri örneğinin kullanılacağını belirler. Tüm veri seti yerine küçük gruplar halinde eğitim yapılır.

Etkisi: Küçük batch boyutları daha gürültülü, ancak daha genelleyici güncellemeler sağlar; büyük batch boyutları ise daha stabil, fakat daha az genelleyici olabilir. **Görsel 3.5**



Görsel 3.5

Epoch Sayısı (Number of Epochs): (3)

Tanım: Modelin tüm eğitim veri seti üzerinden kaç kez geçeceğini belirler.

Etkisi: Daha fazla epoch, modelin daha uzun süre öğrenmesine olanak tanır, ancak aşırı öğrenmeye (overfitting) yol açabilir.

9. Dropout Oranı (Dropout Rate): (3)

Tanım: Eğitim sırasında bazı nöronların rastgele devre dışı bırakılma oranını belirler. Aşırı öğrenmeyi önlemek için kullanılır.

Etkisi: Dropout, modelin genelleştirme kabiliyetini artırır, ancak çok yüksek bir oran öğrenme kapasitesini azaltabilir.

Öğrenme Sürecinde Karşılaştığım Zorluklar

Öğrenme Sürecimin daha da hızlandığını düşünüyorum.
Çözümlerini bulamadığım bazı problemler biraz yavaşlamama sebep oldu.
Problemlerin çözümü için farklı bir ortamda kod yazmaya başladım.
Halen çözemediğim ancak beni yavaşlatmayan problemler var ancak hepsinin üstesinden geleceğime inancım tam.
Sonraki aşamalar için çok heyecanlıyım.

Kaynakça

1 https://alper-bayram.medium.com/makina-%C3%B6%C4%9Frenmesi-ml-algoritmalar%C4%B1-nelerdir-2d58d6d109c4

2 https://chatgpt.co

3 https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4

4 https://medium.com/@niousha.rf/support-vector-regressor-theory-and-coding-exercise-in-python-ca6a7dfda92

5 https://en.wikipedia.org/wiki/Decision_tre

6 **https://www.youtube.com/watch?v=MOeodrg7ceA&list=PLkJUWWxr1XL6tm7mTGlIXpSfyi6256_nZ**

7 https://medium.com/@anilguven1055/k-means-k%C3%BCmeleme-y%C3%B6ntemi-ed1c537046c

8 **https://www.youtube.com/watch?v=jztwpsIzEGc**

9 https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorith