K-NN ve MLP ile İris Çiçeği Analizi

Onur Kantar   
*Kocaeli Üniversitesi*  
*Bilişim Sistemleri Mühendisliği*Umuttepe Kampüsü, Kocaeli, Türkiye  
onurkantr@gmail.com

*Özet*—Bu çalışma, iris çiçeğinin türünü belirlemek için K-En Yakın Komşu (K-NN) ve Çok Katmanlı Algılayıcılar (MLP) kullanılarak geliştirilmiş makine öğrenmesi projesidir.

Anahtar Kelimeler—K-NN, MLP, iris, makine öğrenmesi

# Giriş

Makine öğrenmesi, verilerin analiz edilip bunlardan bir sonuç elde edilmesine yarayan ve şu anda hayatımızın neredeyse her yerinde kullanılan teknolojidir. Bu projemizde KNN ve MLP algoritmalarını kullanarak iris veri setini kullanarak iris çiçeğinin türünü belirleyeceğiz. Bu sayede makine öğrenmesi, algoritmalar ve veri setleri üzerinde bilgi sahibi olacağız.

# K-NN

## Nedir?

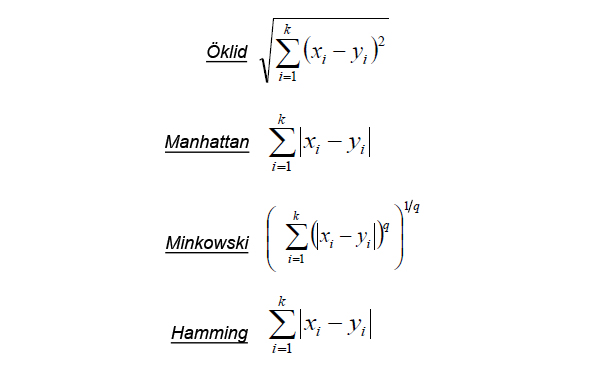
## K-NN (K-Nearest Neighbor) algoritması en basit ve en çok kullanılan sınıflandırma algoritmalarından biridir. K-NN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır.

### Non-parametric: Haritalama fonksiyonunun şekli hakkında güçlü varsayımlar yapmayan algoritmalara non-parametric makine öğrenme algoritmaları denir.

### Lazy: Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini “ezberler”.

## Nasıl Çalışır?

Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır. Öklid fonksiyonuna alternatif olarak Manhattan, Minkowski ve Hamming fonksiyonları da kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.



1. Fonksiyonların formülleri.

# MLP

## Nedir?

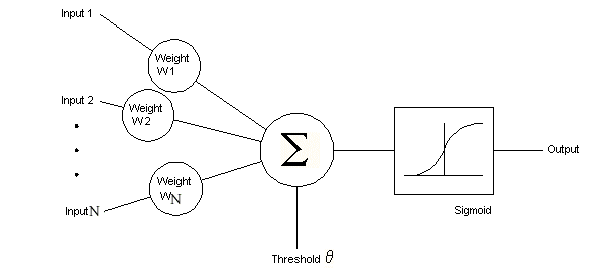
## Çok Katmanlı Algılayıcılar (MLP) XOR problemini çözmek için yapılan çalışmalar sonucu ortaya çıkmıştır. MLP özellikle sınıflandırma ve genelleme yapma durumlarında etkin çalışır. Çok Katmanlı Ağların yapısı aşağıdaki gibidir.

## 

1. MLP’nin yapısı.

## Nasıl Çalışır?

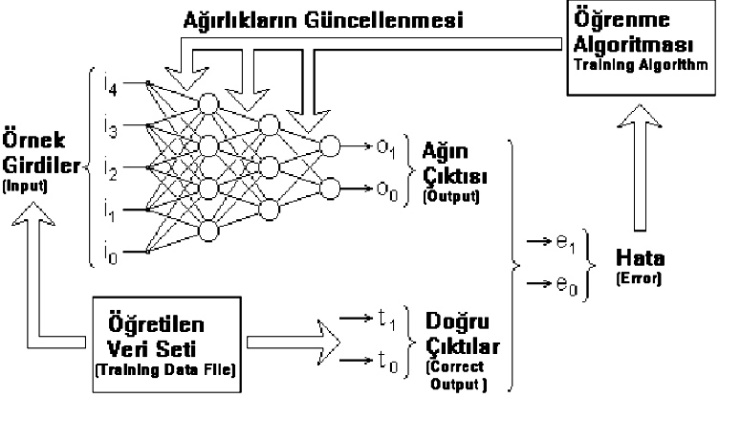
Paralel işlem yapan birden fazla nörona ihtiyaç duyulduğunda katman kavramı devreye girer. Görüldüğü üzere Single Perceptron Model’den farklı olarak arada gizli katman bulunmaktadır. Giriş katmanı gelen verileri alarak ara katmana gönderir. Gelen bilgiler bir sonraki katmana aktarılırlar. Ara katman sayısı en az bir olmak üzere probleme göre değişir ve ihtiyaca göre ayarlanır. Her katmanın çıkışı bir sonraki katmanın girişi olmaktadır. Böylelikle çıkışa ulaşılmaktadır. Her işlem elemanı yani nöron bir sonraki katmanda bulunan bütün nöronlara bağlıdır. Ayrıca katmandaki nöron sayısı da probleme göre belirlenir. Çıkış katmanı önceki katmanlardan gelen verileri işleyerek ağın çıkışını belirler. Sistemin çıkış sayısı çıkış katmanında bulunan eleman sayısına eşittir.



1. Aktivasyon Fonksiyonu (Sigmoid).

Modelde aktivasyon fonksiyonu olarak herhangi bir matematiksel fonksiyon kullanılabilir. Ancak Sigmoid, tang, lineer, threshold ve hard limiter fonksiyonları en çok kullanılan fonksiyonlardır.

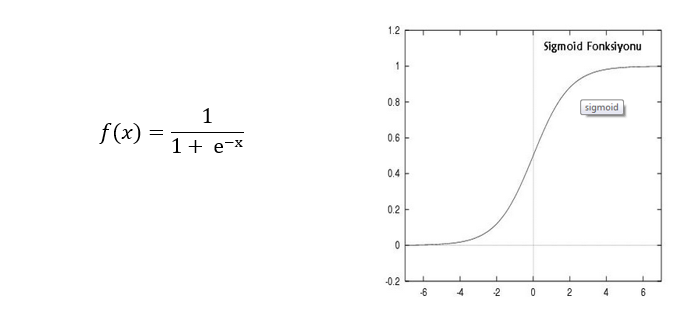
Çok katmanlı ağda öğrenme Delta Öğrenme Kuralı tabanlıdır.



1. Genelleştirilmiş Delta Öğrenme Kuralı’nın yapısı.

Ağın öğrenebilmesi için örnek giriş ve çıkışlardan oluşan eğitim seti şarttır. Geri Yayılımlı Yapay Sinir Ağları’nda öğrenme işlemi bir anlamda örnek setindeki giriş değerleriyle, çıkış değerlerini eşleştiren fonksiyonu bulma işlemidir. Sistemin öğrenme metodu genel olarak iki aşamadan oluşur. Birinci kısım ileri doğru hesaplamadır. İkinci kısım ise geri doğru hesaplamadır.

### İleri doğru hesaplama: sisteme verilen girdi ara katmanlardan geçerek çıkışa ulaşır. Her işlem elemanına gelen girdiler toplanılarak net girdi hesaplanır. Bu net girdi aktivasyon fonksiyonundan geçirilerek mevcut işlem elemanının çıktısı bulunur ve bu çıktı değeri bir sonraki katmanda bulunan işlem elemanlarına gönderilir. Bu işlemler tekrar edilerek en son çıktı katmanından çıktılar elde edilir. En çok kullanılan aktivasyon fonksiyonu olan sigmoid fonksiyonu şekildedir.

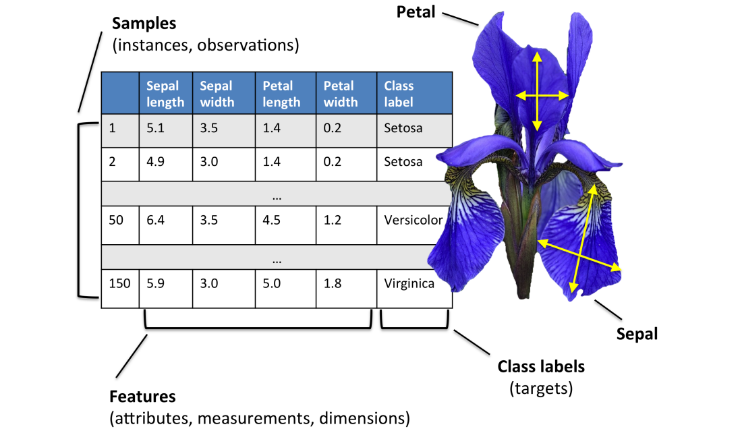


1. Sigmoid fonksiyonu.

### Geri doğru hesaplama: Ağdan çıktı alınmasıyla öğrenmenin ilk aşaması bitirilmiş olur. İkinci aşama hatanın dağıtılması olacaktır. Beklenen çıktı değeri ile elde ettiğimiz birbirinden farklı ise hata vardır. Geriye doğru hesaplama aşamasında hata ağırlık değerlerine dağıtılarak her iterasyonda azaltılması beklenir. Sisteme başlangıçta random olarak verilen ağırlık değerleri, hataların ağırlıklara dağıtılmasıyla her iterasyonda güncellenmiş olur.

## İris Veri Seti

İris veri seti, örüntü tanıma literatüründe bulunan belki de en iyi bilinen veri setidir. Veri seti, her biri 50 türden oluşan 3 sınıf içerir, burada her sınıf bir tür iris bitkisini belirtir. Ayrıca her bir örnek için 4 özellik tanımlanmıştır.



1. İris veri seti özet gösterimi.

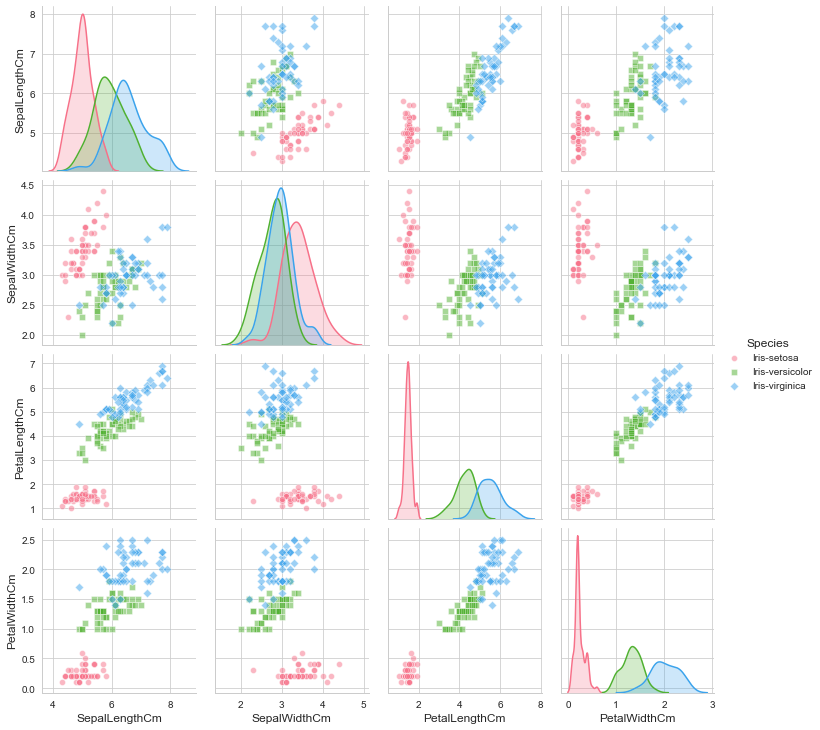
### Özellik bilgileri:

* Sepal Length: Çanak yaprak uzunluğu.
* Sepal Width: Çanak yaprak genişliği.
* Petal Length: Taç yaprak uzunluğu.
* Petal Width: Taç yaprak genişliği.

### Sınıf bilgileri:

* Iris Setosa (50)
* Iris Versicolor (50)
* Iris Virginica (50)

### Grafiksel gösterim:



1. İris veri setinin grafiksel gösterimi.

## K-NN İşlem Adımları

### “iris.csv“ dosyasındaki veri küme yüklenir.

### Veriler özellikler ve sınıflar x ve y diye ayrılır.

### X verileri normalleştirilir.

### Y verine one hot encoding uygulanır.

### Train ve test işlemleri için ayrıl gerçekleştirilir.

### KNeighborsClassifier yardımıyla 1 ile 30 arasındaki tüm k değerlerinin doğruluk değerleri hesaplanır.

### En yüksek doğruluk değeri veren k değeri bulunur ve yazdırılır.

### Sonuçlar confusion matrix yardımıyla yorumlanır.

## MLP İşlem Adımları

### “iris.csv“ dosyasındaki veri küme yüklenir.

### Veriler özellikler ve sınıflar x ve y diye ayrılır.

### X verileri standartlaştırılır.

### Y verine one hot encoding uygulanır.

### Train ve test işlemleri için ayrıl gerçekleştirilir.

### Katmanlar oluşturulup eğitime başlanır.

### Çıkan sonuçlar grafikler ve confusion matrix yardımıyla yorumlanır

## Doğruluk Sonuçları

|  |  |
| --- | --- |
| **Sınıflandırıcı** | **Doğruluk Sonucu** |
| K-NN | 1.00 |
| MLP | 1.00 |

##### Kaynakça

1. https://github.com/xexuew/MLP-Keras-Iris-Dataset/blob/master/Iris%20MLP.ipynb
2. <https://medium.com/@contactsunny/why-do-we-need-feature-scaling-in-machine-learning-and-how-to-do-it-using-scikit-learn-d8314206fe73>
3. <https://stackoverflow.com/questions/28064634/random-state-pseudo-random-number-in-scikit-learn>
4. <https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/>
5. <https://machinelearningmastery.com/how-to-develop-multilayer-perceptron-models-for-time-series-forecasting/>
6. <https://medium.com/@ayyucekizrak/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd>
7. https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4