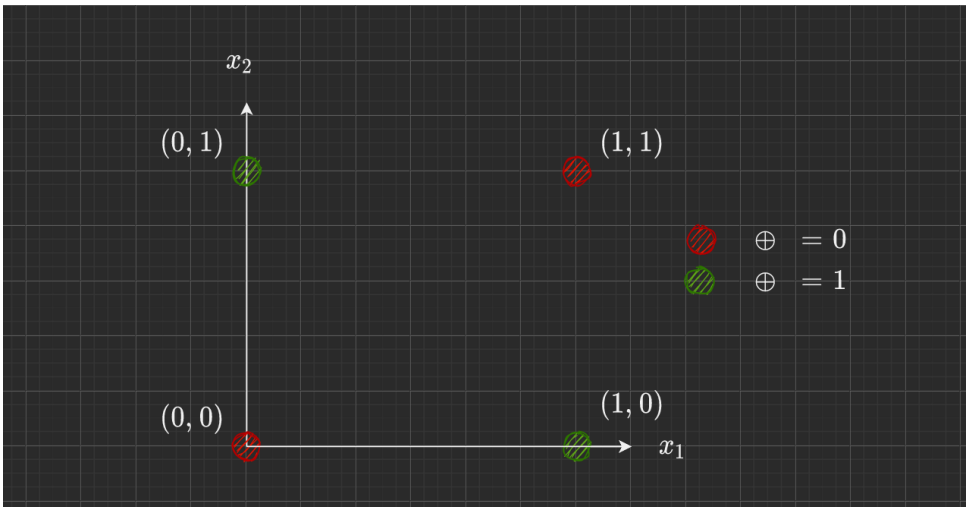
Selim CAN ERKAN

185260019

ÖDEV

1) XOR KAPISI

Bizden istenilen XOR kapısını tekli lineer sınıflandırma yöntemi ile elde etmek ama XOR'un doğruluk tablosunu 2D bir tabloda gösterecek olursak Şekil1deki gibi bir sonuç elde ediyoruz.

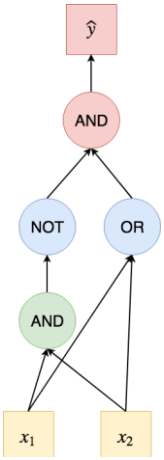


Şekil 1

Şekil1de de görüldüğü gibi tek bir doğru ile sınıflandırma yapmak bize XOR kapısını elde etmeye yetmiyor. Çünkü doğruyu nereden kesersek keselim muhakkak bir değer yanlış sınıflandırılmış oluyor. Bu durumu araştırdığımda elde ettiğim çözüm XOR kapısına girilen giriş değerlerini belirli bir formüle göre AND, OR ve NOT kapıları ile işleyip sonucu elde etmek oldu. Burada bahsetmiş olduğum formül ise şöyledir;

$$XOR(x1, x2) = AND(NOT(AND(x1, x2)), OR(x1, x2))$$

Şekil 2



Şekil 3

Şayet bu formülden faydalanacak olursam AND,OR ve NOT kapılarını verilen kod ile eğitip bunlarla girilen değeri tahmin edip doğru sonucu bulabilirim. Şekil3 deki adımları bu formülün algoritması olarak düşünebiliriz. Gerekli işlemleri bu adımlardan faydalanarak yapmak bizi doğru sonuca götürecektir.

KODLARIN AÇIKLAMASI

```
[ ]  import matplotlib.pyplot as plt
     import numpy as np
```

```
[ ]  giris = np.array([[1, 1], [1, 0], [0, 1], [0, 0]])
     cikis_AND = np.array([1, 0, 0, 0]) # and
     cikis_OR = np.array([1, 1, 1, 0]) #or

     giris_NOT= np.array([[1],[0]])
     cikis_NOT= np.array([0,1])
```

Şekil 4

İlk olarak eğitim için kullanacağım kapıların giriş ve çıkış değerlerini np kütüphanesinden faydalanarak Şekil 4de görüldüğü gibi belirli değişkenlere atıyorum.

```
[ ]  class Perceptron(object):
         def __init__(self, ogrenme_orani=0.1, iter_sayisi=10):
             self.ogrenme_orani = ogrenme_orani
             self.iter_sayisi = iter_sayisi

         def ogren(self, X, y):
             self.w = np.zeros(1 + X.shape[1])
             self.hatalar = []
             for _ in range(self.iter_sayisi):
                 hata = 0
                 for xi, hedef in zip(X, y):
                     degisim = self.ogrenme_orani * (hedef - self.tahmin(xi))
                     self.w[1:] += degisim * xi
                     self.w[0] += degisim
                     hata += int(degisim != 0.0)
                 self.hatalar.append(hata)
             return self

         def net_input(self, X):
             return np.dot(X, self.w[1:]) + self.w[0]

         def tahmin(self, x):
             a = np.where(self.net_input(x) >= 0.0, 1, 0)
             return a
```

Şekil 5

Şekil5de ise öncelikle sınıftan bir nesne oluşturulduğunda iterasyon sayısı ile öğrenme oranını parametre olarak alınıyor. Şayet nesneyi oluştururken parametre girilmezse diye de kurucu metotta sabit değerler atanıyor.

“ogren” metodunda ağırlıklar için giriş değerinin bir fazlası olacak şekilde bir “w” dizisi oluşturuluyor. Bu dizide bias değeri ve ağırlıklarını değişimi saklanıyor. Bu hesaplama ise şu şekilde oluyor;

$$y = f(x_1 \cdot w_1 + x_2 \cdot w_2 + b)$$

Şekil 6

Şekil6da görülen formülü kodlamada “net_input” metodu ile yapıyoruz. Buradan elde edilen değer “threshold” olarak adlandırılan bir yöntem ile belli bir değerin üstünde ise 0 değilse 1 gibi biri değer olarak işleniyor. Bu işlemi de “tahmin” metodundan faydalanarak yapıyoruz.(Şekil 7)

```
def net_input(self, X):
    return np.dot(X, self.w[1:]) + self.w[0]

def tahmin(self, x):
    a = np.where(self.net_input(x) >= 0.0, 1, 0)
    return a
```

Şekil 7

```
degisim = self.ogrenme_orani * (hedef - self.tahmin(xi))
self.w[1:] += degisim * xi
self.w[0] += degisim
```

Şekil 8

Şekil 8deki kod parçasında ise modelin doğruluk oranın arttırmaya yaran Bias değerini ve ağırlıkların değişimlerini hesaplıyor ve güncelliyor. Buradaki “degisim” değişkeni eğitimdeki hatayı hesaplamaktadır.

```
def XOR(x):
  siniflandirici_AND = Perceptron(ogrenme_orani=0.1, iter_sayisi=10)
  siniflandirici_NOT = Perceptron(ogrenme_orani=0.1, iter_sayisi=10)
  siniflandirici_OR = Perceptron(ogrenme_orani=0.1, iter_sayisi=10)


  siniflandirici_AND.ogren(giris, cikis_AND)
  siniflandirici_NOT.ogren(giris_NOT,cikis_NOT )
  siniflandirici_OR.ogren(giris, cikis_OR)

  sonuc_AND = siniflandirici_AND.tahmin(x)
  sonuc_AND_NOT = siniflandirici_NOT.tahmin(sonuc_AND)
  sonuc_OR = siniflandirici_OR.tahmin(x)

  new_x = np.array([sonuc_AND_NOT[0], sonuc_OR])
  return siniflandirici_AND.tahmin(new_x)
```

Şekil 9

Şekil 9de ise önce kullanacağım her bir kapı için bir sınıflandırıcı tanımlıyor ve gerekli giriş, çıkış değerleri ile eğitiyorum. Daha sonra Şekil 3de görülen işlem adımlarını takip ederek parametre olarak alına değeri işliyor ve XOR kapısına bu parametre girdiğinde elde edilecek sonucu, sınıflandırıcılar yardımı ile elde etmiş oluyorum. Bu işlem adımları ise şöyledir;

- Parametre olarak alınan değerlerin AND kapısı değerleri ile eğitilmiş olan değişkenden faydalanarak çıkış değerini tahmin et ve sonucu bir değişkene ata. Bu değişkene burada “sonuc_AND” dedim.
- Elde ettiğimiz sonucu NOT kapısı değerleri ile eğitilmiş olan değişkenden faydalanarak çıkış değerini tahmin et ve sonucu bir değişkene ata. (Bu değişkene burada “sonuc_AND_NOT” dedim.)
- Parametre olarak alınan değerleri OR kapısı değerleri ile eğitilmiş olan değişkenden faydalanarak çıkış değerini tahmin et ve bir değişkene ata. Bu değişkene burada “sonuc_OR” dedim.
- “sonuç_AND_NOT ile sonuç_OR değişkenini np kütüphanesinden faydalanarak bir ikili değer olacak şekilde birleştir ve AND kapısı değerleri ile eğitilmiş olan değişkenden faydalanarak çıkış değerini tahmin et ve sonucu geri gönder.

Bu adımları uygulayarak XOR kapısını elde etmiş oldum.

2) Veri Kümesi Eğitimi

Okul numarasının son hanesine göre eğitmem gerek veri kümesinin linkini kodda tanımlayıp veri kümesi değerlerini alıyorum.

	0	1	2	3	4	5	6	7	8
0	NaN	ozellik-1	ozellik-2	ozellik-3	ozellik-4	ozellik-5	ozellik-6	ozellik-7	cikis
1	0.0	-1.4080093652163415	-7.6656452619446025	-7.238232614974349	8.27005204886454	2.316332076510735	7.78069725994181	8.746140758288522	1
2	1.0	-2.3607238576457528	9.518346596580436	1.8483606179808003	9.62115643470663	-8.37008940851254	1.6157250616275787	7.208178980641894	0
3	2.0	-3.6768645865711664	9.766196950795905	0.8032334518724678	7.475693012048605	-8.519747478526474	4.371835533376637	5.475342832122751	0
4	3.0	-3.5360115449773155	7.943393253823311	2.466308517360952	7.327269280782561	-8.002647507690526	2.893346486968285	6.109700648700899	0
...	...	...	...	...	...	...	...	...	...
138	137.0	-4.2828928159065995	9.610162187361816	1.1204592530576047	7.917081727180424	-6.93999247648068	6.1208119781494705	4.27027460709047	0
139	138.0	-0.49861610166772463	-8.679843404948713	-10.801956967184243	8.672718658806925	3.7473853639137893	7.374688422411102	11.400780726988502	1
140	139.0	-2.500175785722042	-7.915534378731315	-6.377663194153808	7.059154817380378	3.5377739302046507	6.958497655165187	8.028975801492967	1
141	140.0	-2.939196147021269	10.093756985699981	0.27979067121290424	9.513667229485844	-8.557074490770898	2.188721060308475	6.379760417603098	0
142	141.0	-3.26251606248002	-7.744204542274318	-7.214319157387278	8.790517592118007	3.638784783065677	7.315731729918048	8.117473507323892	1

Şekil 10

Şekil 10da da görüleceği gibi ilk satırın değerlerden oluşmamasından dolayı veri kümesini okurken buna uygun olarak düzenliyorum daha sonra giriş ve çıkış değerlerini belirliyorum.(Şekil 11)

```
[ ]  cikis = df.iloc[1:100, 8].values
     giris = df.iloc[1:100, [2,3, 7]].values

     giris = giris.astype(float)
     cikis = cikis.astype(float)
```

Şekil 11

Eğitim kümesi doğru bir şekilde okuduktan ve verilerin lineer sınıflandırmaya uygun bir kümeleme sergilediğinden emin olduktan 2,3 ve 7 özellikleri giriş olarak alıyor sonra eğitimi yapıyorum. Burada 3 özellik almamın sebebi 2 özellik ile kümeleme lineer sınıflandırmaya uygun olmadığındandır. Kodları 1. Ödev ile aynı olduğundan hesaplamaları anlatırken tekrara binmek istemiyorum. Temel olarak Şekil 6daki formülü kullanıyoruz.

5 Örnek

A) **X1=** 10.25368143 **w1=** -1

X2= -0.87653154 **w2=** -0.15

X3 = 5.74261898 **w3=** -0.70

Şekil 6daki formüle göre hesaplarsak ;

(X1*w1)+(x2*w2)+(x3*w3)=y

(10.25368143*-1)+(-0.87653154*-0.15)+(5.74261898*0.70)=y

(-10.25368143)+(0.131479731)+(4.019833286)= -6.102368413

Elde ettiğimiz değeri -6.102368413 tahmin metodunda olduğu gibi 0dan büyük ya da küçük olduğuna göre değerlendirecek olursak "**0**" değerini alıyor.

B) **X1=** -9.58811909 **w1=** -1

X2= -9.13315552 **w2=** -0.15

X3 = 6.95728263 **w3=** -0.70

Şekil 6daki formüle göre hesaplarsak ;

(X1*w1)+(x2*w2)+(x3*w3)=y

(-9.58811909*-1)+(-9.13315552*-0.15)+(6.95728263*0.70)=y

(9.58811909)+(1.369973328)+(4.870097841)= 15.828190259

Elde ettiğimiz değeri 15.828190259 tahmin metodunda olduğu gibi 0dan büyük ya da küçük olduğuna göre değerlendirecek olursak "**1**" değerini alıyor.

C) **X1=** 11.07593915 **w1=** -1

X2= 0.54337096 **w2=** -0.15

X3 = 5.94963275 **w3=** -0.70

Şekil 6daki formüle göre hesaplarsak ;

(X1*w1)+(x2*w2)+(x3*w3)=y

(11.07593915*-1)+(0.54337096*-0.15)+(5.94963275*0.70)=y

(-11.07593915)+(-0.081505644)+(4.164742925)= -6.992701869

Elde ettiğimiz değeri -6.992701869 tahmin metodunda olduğu gibi 0dan büyük ya da küçük olduğuna göre değerlendirecek olursak "**0**" değerini alıyor.

D) **X1=** -7.62109444 **w1=** -1

X2= -9.4831335 **w2=** -0.15

X3 = 8.89551378 **w3=** -0.70

Şekil 6daki formüle göre hesaplarsak ;

(X1*w1)+(x2*w2)+(x3*w3)=y

(-7.62109444*-1)+(-9.4831335 * -0.15)+(8.89551378 * 0.70)=y

(7.62109444)+(1.422470025)+(6.226859646)= 15.270424111

Elde ettiğimiz değeri 15.270424111 tahmin metodunda olduğu gibi 0dan büyük ya da küçük olduğuna göre değerlendirecek olursak "**1**" değerini alıyor.

[10.31884752 0.74405258 6.71506055]

E) **X1=** 10.31884752 **w1=** -1

X2= 0.74405258 **w2=** -0.15

X3 = 6.71506055 **w3=** -0.70

Şekil 6daki formüle göre hesaplarsak ;

(X1*w1)+(x2*w2)+(x3*w3)=y

(10.31884752 *-1)+(0.74405258 *-0.15)+(6.71506055*0.70)=y

(-10.31884752)+(-0.111607887)+(4.700542385)= -5.729913022

Elde ettiğimiz değeri -5.729913022tahmin metodunda olduğu gibi 0dan büyük ya da küçük olduğuna göre değerlendirecek olursak "**0**" değerini alıyor.

NOT: Hesaplamalarda bias değeri dahil edilmemiştir.

Kaynaklar

- https://towardsdatascience.com/perceptrons-logical-functions-and-the-xor-problem-37ca5025790a
- https://towardsdatascience.com/how-neural-networks-solve-the-xor-problem-59763136bdd7
- https://sefiks.com/2020/01/04/a-step-by-step-perceptron-example/