BLM4520 Yapay Sinir Ağlarına Giriş
1. Ödev Raporu

Mert Şamil Gül

November 28, 2020

1 Perceptron Mimarisi

Uygulanan yapay sinir ağı mimarisinde perceptron algoritması kullanılmıştır. Perceptron, ikili sınıflandırıcılar için supervised bir öğrenme algoritmasıdır. Lineer ayrılabilir olan bütün problemlerde Perceptron öğrenme algoritmasının sonlu iterasyon ile problemi çözecek olan uygun ağırlıkları bulduğu kanıtlanmıştır.

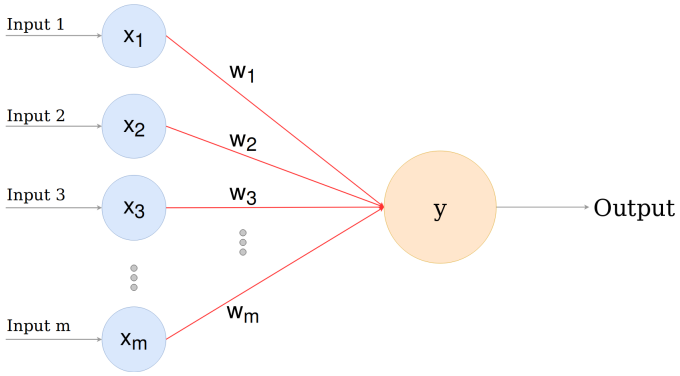

Figure 1: Perceptron Mimarisi

Perceptron algoritması Figür 1 deki gibi gösterilebilir. Yapılan programda şekildeki mimariye ek olarak ağırlıklı toplama, değeri her zaman +1 olan bias değeri kendi ağırlığı b ile çarpılarak eklenmiştir. Eğer i'inci input x_i, ağırlığı ise w_i ile gösterilirse toplama işlemi formül olarak aşağıdaki gibidir.

$$y_{in} = b + \sum_i w_i x_i \tag{1}$$

Ayrıca bias eklenmiş ağırlıklı toplam daha sonra bir aktivasyon fonksiyonundan geçirilerek fonksiyon sonucu çıktı olarak verilmektedir. Bu mimarideki aktivasyon fonksiyonunda daha önceden belirlenmiş bir θ (threshold) değeri kullanılmıştır. Kullanılan aktivasyon fonksiyonu formül 2'de gibidir.

$$y = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > \theta \\ 0, & \text{if } -\theta \leq y_{in} \leq \theta \\ -1, & \text{if } y_{in} < -\theta \end{cases} \tag{2}$$

Son olarak oluşan y çıktısının -1 veya +1 olan t hedef değeri ile karşılaştırılarak ağırlıkların güncellenip güncellenmeyeceği belirlenir. $y \neq t$ durumunda yine daha önceden belirlenmiş η (learning rate) değeri kullanılarak formül 3'de gösterildiği gibi ağırlıklar güncellenir. Aynı şekilde bias'ın ağırlığı olan b değeri de formül 4'deki gibi güncellenir.

$$w_i(new) = w_i(old) + \eta t x_i \tag{3}$$

$$b(new) = b(old) + \eta t \tag{4}$$

Tüm nöronların outputu target değere eşit olduğu durumda kısaca tüm çıkışlar için $y = t$ durumunda ise ağırlıklarda artık bir değişim olmayacağı için maksimum epoch sayısına ulaşılmadan eğitim sonlanır.

2 Veri

Ödevde kullanılan 7x9 matriste kodlanmış 3 farklı fontta 7 adet harfin binary ve bipolar biçimde değerleri vektor olarak alınmıştır. Projede amaç, verilen 21 harfin perceptron mimarisinde eğitimi ve hangi harfin hangi harf olduğunun tanınmasını sağlamaktır. Binary vektörde boş pixeller için 0, dolu pixeller için 1, bipolar vektörde ise dolu pixeller 1 iken boş pixeller -1 alınmıştır. Binary ve bipolar değerler sonuçta farklılıklar oluşturabileceği için 2 durum için de testler yapılmıştır.

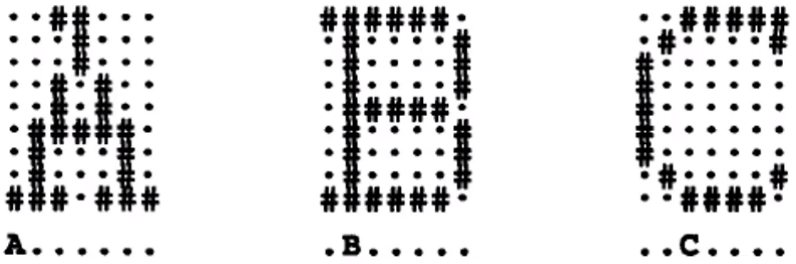

Figure 2: Girdiler

3 Eğitim Algoritması

Verilerin eğitilmesi için hazırlanan algoritmada 7*9 boyutunda vektörlerde tutulan 21 adet karakter bulunuğundan 21 tane 63'lük x girişi tanımlanmıştır. Bu 7 karakterin her birinin çıkışının farklı olması beklendiğinden 7 adet de y çıkış değeri ve her çıkış değerin için de bir adet b(bias ağırlığı) tanımlanmıştır. Programda kullanılacak diğer parametreler de değiştirilebilir şekilde programın başında tanımlanıştır.

```
double x[21][63]; // karakter vektorleri
double w[63][7]; // agirliklar
double y_in=0; // agirlikli toplam sonucu
double y; // aktivasyon fonksiyonu sonucu
double lr = 1; // learning rate degeri
double b[7]; // bias degerleri
```

```
double th=0.2; // threshold degeri
int max_epoch = 10; // maksimum epoch sayisi
```

Karakterler verilen txt dosyalarından okunup x matrisine atandıktan sonra ağırlıklar ve bias değerleri 0 lanır. Ardından maximum epoch değerine ulaşana kadar veya hiç bir değişim olmayana kadar dönecek olan döngünün içinde perceptron algoritması adımlarına başlanır. Her epoch içinde tüm inputlar birer kere perceptrona girdiği için 21 adımlı bir döngü daha tanımlanmıştır. 2 adet döngünün içinde aşağıdaki kod parçası ile ağırlıklı toplama işlemi yapılmaktadır.

```
y_in=b[z];
for(int k=0;k<63;k++)
    y_in += x[j][k]*w[k][z];
```

Daha sonra çıkan sonuç(y_{in}) aşağıdaki gibi formül 2'te bahsedilen aktivasyon fonksiyonuna girerek fonksiyon sonucunda perceptronun çıkış değeri hesaplanmaktadır.

```
if(y_in > th)
    y=1;
else if(y_in < th)
    y=-1;
else
    y=0;
```

Çıkış değeri, belirlenen target(t) değeri ile farklı olduğunda oluşan hatadan öğrenim gerçekleşmesi için ilgili çıkışlara bağlı olan tüm ağırlık değerle formül 3 kullanılarak güncellenir.

```
if(y!=t){
    for(int k=0;k<63;k++)
        w[k][z] = w[k][z] + lr*t*x[j][k];
    b[z] = b[z] + lr*t;
    same=false;
}
```

Döngünün başında true atanmış same boolean değişkeni herhangi bir çkışta eşitsizlik olduğunda false değerini alarak döngünün devam etmesini sağlar. Fakat hiçbir ağırlıkta değişiklik olmazsa max_epoch değerine ulaşmadan döndüden çıkılır ve eğitim tamamlanmış olur.

4 Analiz

Yapılan testler sonucunda bipolar değerlerde $\eta = 0.01$ ve $\theta = 0.2$ için perceptronun tamamen öğrenmesi 5 epoch sürmüştür. Aynı değerlerde binary inputlarda ise sonuç 11 olmuştur. Sonucun daha net anlaşılabilmesi için her bir farklı harf için yapılan testlerde tüm çıkışların değerleri ekrana yazdırılmıştır.

```
5.epoch 1. adim  input: Font_1_A.txt
A: 1 | B: -1 | C: -1 | D: -1 | E: -1 | J: -1 | K: -1
5.epoch 2. adim  input: Font_1_B.txt
A: -1 | B: 1 | C: -1 | D: -1 | E: -1 | J: -1 | K: -1
5.epoch 3. adim  input: Font_1_C.txt
A: -1 | B: -1 | C: 1 | D: -1 | E: -1 | J: -1 | K: -1
```

Figure 3: Bipolar input 5. epoch çıktısı

Figür 3'te bipolar değerler ile 5. epochtaki ilk 3 adımda, Font 1 A,B ve C harfleri için çıkan sonuçlar gösterilmektedir. Perceptron çıktılarına bakıldığında her harf için sadece kendi olması gereken çıkış değerinin +1 diğer çıkış değeğerlerinin ise -1 olduğu gözlemlenmiştir . Tüm değerlerde aynı sonuca ulaşıldığı için program 5. epochta sonlanmıştır. 4. epoch çıkışlarına bakılırsa, 9. adım çıktısında Font 2 B karakterinin E karakteri ile karıştırıldığı bu yüzden E çıkışının -1 yerine 0 olduğu görülmüştür. 3. epochta ise Font 1 C karakteri ile yine E karakterinin karıştırıldığı gözlenmiştir. Bu hatalar sayesinde perceptron öğrenmesi gerçekleştirilmiş ve 5. epochta tamamen öğrenilmiştir. Binary ve bipolar karşılaştırması Figür 4'te gösterilmiştir. Learning rate (η) değerini 0.01 yerine 0.001 yapıldığında ise öğrenme hızı yavaşladığı için bipolar öğrenme 5 den 16 epocha, binary öğrenme ise 9'dan 66 epocha yükselmiştir. Daha küçük learning rate değerlerinde bipolar ve binary değerlerin arasındaki performans farkı daha iyi anlaşılmaktadır.

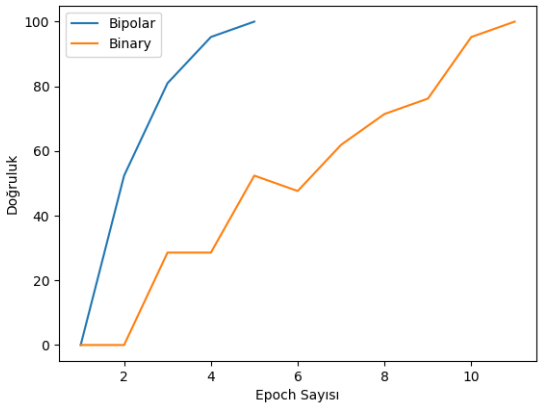

Figure 4: Binary Bipolar Farkı

Eğitim tamamlandıktan sonra eğitimde kullanılmış olan test verisi üzerinde belli yüzdelerde bozma işlemi yapılıp bozulmuş veri ile testler yapılmıştır. Figür 5'de görüldüğü gibi %10luk bir bozulmada harfler hala ayırt edilebilmektedir. Daha yüksek yüzdelerde tamamen ayırt edilemediği gözlemlenmiştir. Bu yüzden %30 bozulmadan sonra perceptron harfleri tanıyamamaktadır ve doğruluğu Figür 6'ta gösterildiği gibi %0'a düşmektedir.

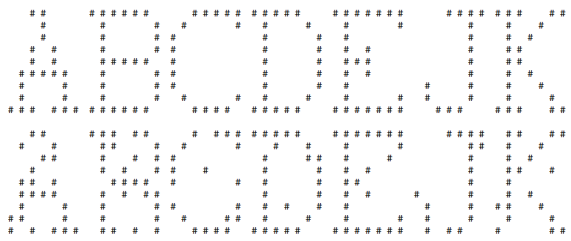

Figure 5: Karakterlerin %10 bozulma öncesi ve sonrası

Binary ve Bipolar için yapılan bozma testlerinde Figür 6 de görülebileceği gibi bipolar değerlerin bozulmalara göre direnci daha yüksektir. Binary değerlerde %28 bozulmadan sonra perceptron %0 başarıya ulaşırken bipolar değerlerde %40'lardan sonra ulaşmaktadır.

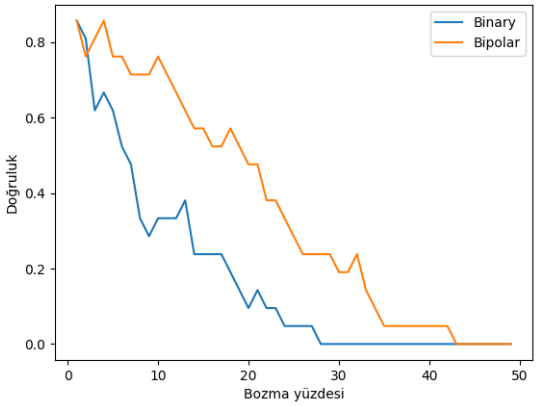

Figure 6: Farklı karakter bozma yüzdeleri için perceptron doğruluğu grafiği

Aynı zamanda ağırlıklar güncelelnirken kullanılan formül 3 ve formül 4'teki perceptron öğrenme kuralı dışında sonuca oluşan çıktının dahil edildiği perceptron öğrenme kuralındaki t çarpanı yerine $t - y$ çarpanı kullanılan formül 5'deki delta kuralı da eğitim aşamasında denenmiştir.

$$w_i(new) = w_i(old) + \eta(t - y)x_i \tag{5}$$

Bipolar veriler ile yapılan testlerde $\eta = 0.01$ ve $\theta = 0.5$ değerleri için perceptron öğrenme kuralı ile eğitim gerçekleştiğimde, eğitim 21 epoch sürmektedir. Aynı veriler ve aynı değerler ile öğrenme kuralı olarak delta kuralı kullanıldığında figür 7 de görüldüğü gii eğitimin sadece 16 epoch sürdüğü gözlemlenmiştir. Bu durum da delta kuralının perceptron öğrenme kuralından daha verimli olduğunu kanıtlamaktadır.

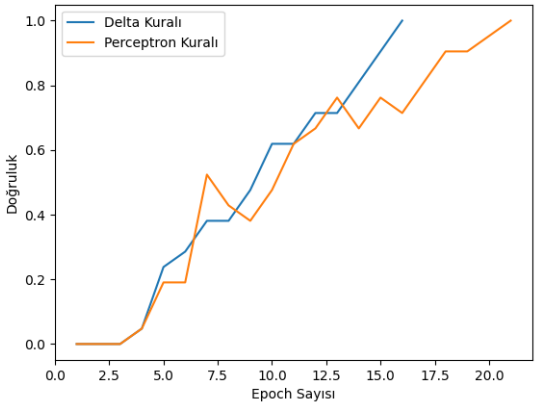

Figure 7: Delta kuralı ve Perceptron kuralı karşılaştırması