YAPAY SİNİR AĞLARI "PERSEPTRON"

HAFTA 5 PROJESİ

23435004013-MELIKA JIBRIL SEID

17/11/2024 | **BURSA TEKNIK UNIVERSITESI**

İÇİNDEKİLER

PROJENİN TANITIMI

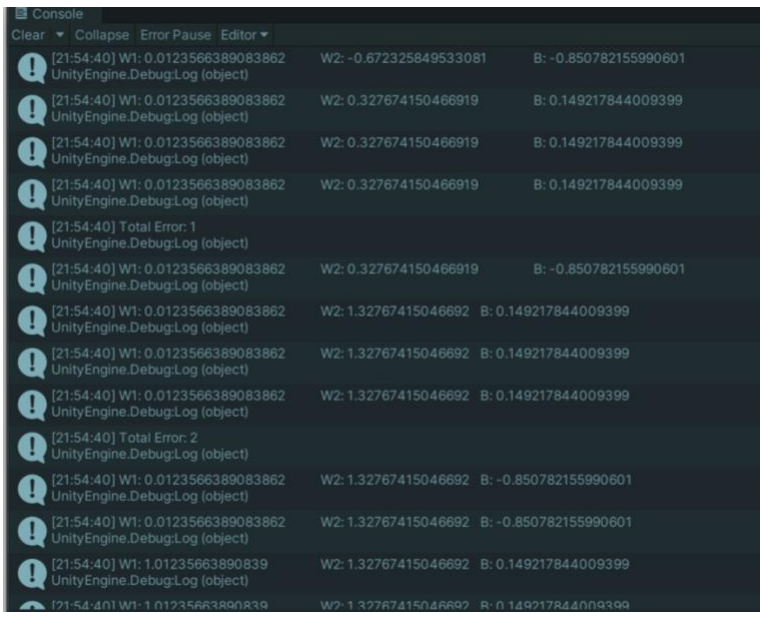
Perseptron, yapay sinir ağlarının en basit türlerinden biridir ve daha karmaşık sinir ağı yapılarının temelini oluşturur. Genellikle ikili sınıflandırma (binary classification) görevlerinde kullanılır. Bu projede ise VE ve VEYA lojik kapılarını temsil edebilen basit bir makine öğrenmesi gerçekleştirmek için kullanılmıştır. Proje unityde yapılmış olup görselliği olmamakla birlikte çıktıları debug bölümünden alınmıştır. Projenin kodları linkte bulunur.
Github linki: https://github.com/m3likaj/GameProgramming/tree/main/Hafta5

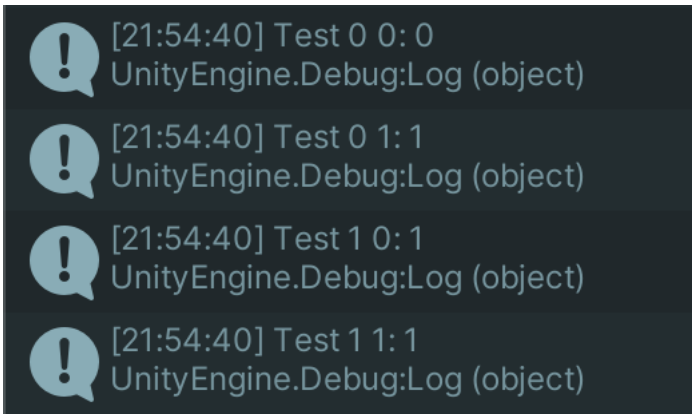
PERCEPTRON MANTIĞI

Perseptronlar, dijital devrelerin ve makine öğrenmesinin temel yapı taşlarından olan **AND**, **OR** ve **NOT** gibi mantık kapılarını modellemek için kullanılabilir. Bu kapılar, ikili sınıflandırma problemlerinde çok yaygın olarak kullanılır. Bir **OR** kapısı, en az bir giriş 1 olduğunda 1 çıkar. Yalnızca her iki giriş de 0 olduğunda çıkış 0 olur. OR kapısının doğruluk tablosu şu şekildedir:

Giriş 1	Giriş 2	Çıkış
0	0	0
0	1	1
1	0	1
1	1	1

Bu projenin amacı perceptronu bu çıktıyı sağlayabilmesi için gerekli olan ağırlık(weight) ve bias değerlerini öğretmektir. Perseptron, denetimli öğrenme (supervised learning) yöntemini kullanarak öğrenir. Bu süreç, modelin hatalarını düzeltmek için ağırlıkları ve bias'ı ayarlamayı içerir. Her durumda, perseptron, girişlerin ağırlıklı toplamını hesaplar ve bir adım fonksiyonu (aktivasyon function) kullanarak çıktıyı belirler. Bu sayede perseptron, bu temel mantık kapılarını doğru şekilde modelleyebilir. Beklenen çıktı ve öğrenme aşamaları aşağıdaki resimlerde bulunur.





İLK HAZIRLIKLAR

Projeyi başlamak için bu adımları takip etmek gerekmektedir.

1. Unity Hubı açıp 2D veya 3D proje oluşturunuz
2. Heirarchyde sağ tıklayıp “create empty”ı tıklayarak boş bir obje oluşturunuz ve ona “perceptron” adını veriniz
3. Assets kısmında sağ tıklayıp “create”, sonra ise “folder”ı basarak script adlı bir dosya oluşturunuz.
4. Script dosyanın içinde sağ tıklayıp “create”, sonra ise “C# script”i basarak “Perceptron_sc” isimli script oluşturunuz.
5. Perceptron_sc scriptini hierarchyde aynı isimli obje üzerinde sürükle bırak ile atayınız.

TRAINING SET SINIFI

Bu sınıf bizim öğrenme setimizi temsil eder. Perceptron denetimli öğrenme yöntemini kullanılmaktadır. Bu bizim öğrenme setimizde etiketli veri eklememiz gerektiği anlamına gelmektedir. VEYA mantık kapısı açısından etiketli veri 2 giriş ve beklenen çıkıştan oluşmaktadır.

```
public class TrainingSet{
    public double[] input;
    public double output;
}
```

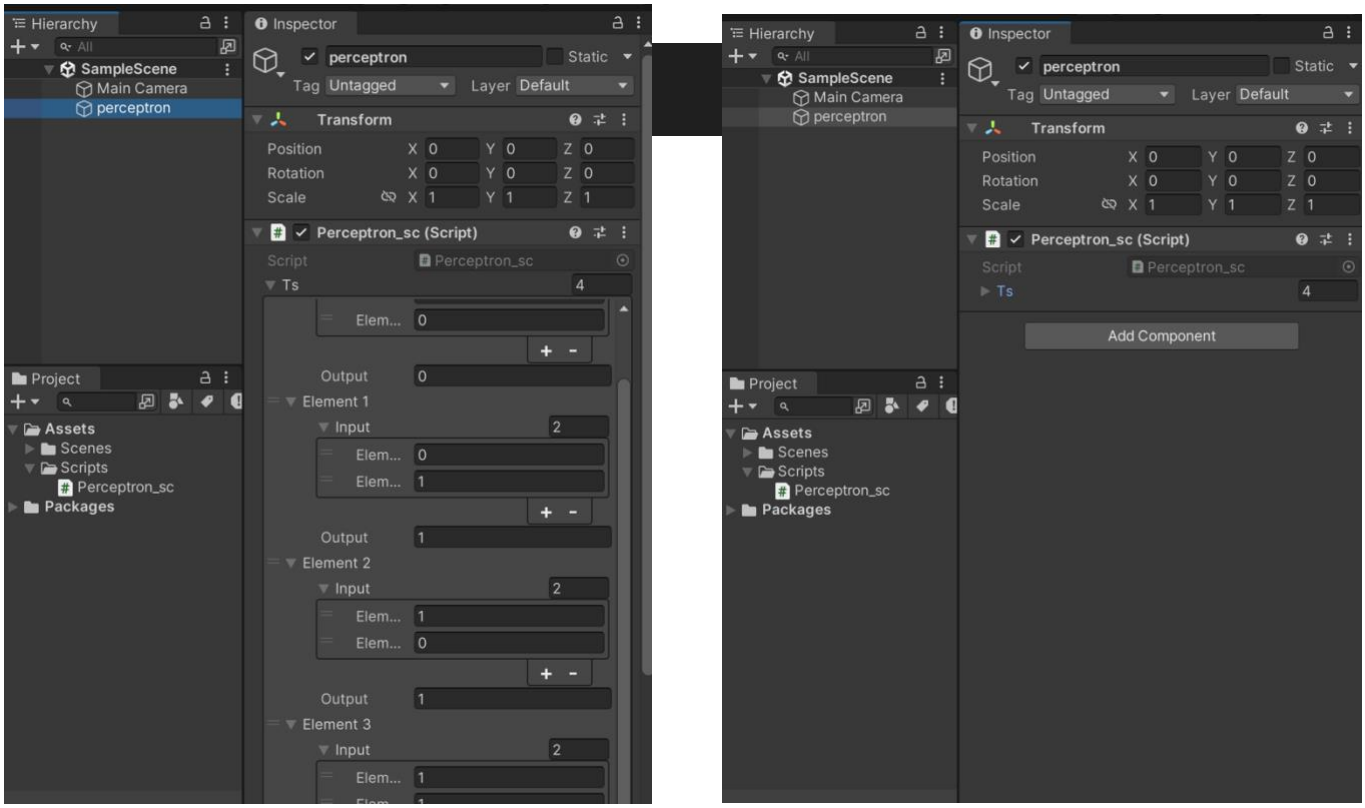
PERCEPTRON SINIFI

Bu script adından anlaşılacağı üzere perceptron nesnemize ait kodları bulunduran dosyadır. İçinde perseptronu eğitmek için kullanılacak çeşitli fonksiyonları var

DEĞİŞKENLER

Burada az önce oluşturduğumuz öğrenme seti sınıfından bir öğrenme set listesi, girişleri çıkışa dönüştürürken kullanacağımız ağırlık listesi ve bias (sapma) değerleri tanımlanmıştır. TotalError ts içindeki tüm veriler üzerinden geçtikten sonra kaç kez hata oluştuğunu tutan bir değişkendir. Bunu tanımladıktan sonra kayıt edip unityde perceptron nesnesi üzerinde tıklayınca skript altında ts adlı bir değişken tanımlandığını görürsünüz. Değişkenin yanındaki kutu içerisinde başta sıfır yazar. O kutuyu tıklayıp ts değişkenimizin uzunluğunu 4 yazarak belirlyebilirsiniz. Daha sonra üstünde tıklayarak içindeki değerleri tanıtımda olan tabloya göre doldurmanız gerekir.

```
    public TrainingSet[] ts;
    double [] weights = {0,0};
```



START FONKSİYONU

Bu fonksiyon Train fonsiyonunu çağırıp 8 kez öğrenmeyi gerçekleştirdikten sonra sonuçları debug kısmında yazdırır.

```
void Start (){
    Train(8);
    Debug.Log("Test 0 0: " + CalcOutPut(0,0));
    Debug.Log("Test 0 1: " + CalcOutPut(0,1));
    Debug.Log("Test 1 0: " + CalcOutPut(1,0));
    Debug.Log("Test 1 1: " + CalcOutPut(1,1));
}
```

TRAIN FONKSİYONU

Bu fonksiyon parametre olarak kaç kez eğitileceğini alır ve ilk olarak ağırlık değerleri rastgele sayılardan başlatır. Eğitim verileri üzerinde birden fazla kez (epoch) işlem yapılır, ağırlıklar ve bias her seferinde ayarlanır. Bu süreç, model tüm eğitim örneklerini doğru şekilde sınıflandırana kadar devam eder. Daha sonra öğrenme setinin her elemanının

ağırlığını ayarlayarak konsola yazdırır. En sonda yapılan hata sayısını de yazar ve toplam hata sayısını tekrardan sıfırlar.

```
void Train(int epochs){
    InitializeWeights();

    for (int i = 0; i < epochs; i++)
    {
        totalError = 0;
        for (int j = 0; j < ts.Length; j++)
        {
            UpdateWeights(j);
            Debug.Log("W1: " + weights[0] + "\t W2: " + weights[1] + "\t B: " + bias);
        }
        Debug.Log("Total Error: " + totalError);
    }
}
```

INITIALIZE WEIGHTS FONKSIYONU

Bu fonksiyon başta tanımlanan weight listenin değerlerini 0 – 1 arasında bir rastgele değer olarak atar. Ağırlıkların sayısı giriş değerlerimizin sayısına eşit olmalıdır. Bu örnek için 2 giriş ve 1 çıktımız olduğuna göre weightin uzunluğu de 2 dir.

```
void InitializeWeights(){
    for (int i = 0; i < weights.Length; i++)
    {
        weights[i] =  Random.Range(-1.0f, 1.0f);
    }
    bias =  Random.Range(-1.0f, 1.0f);
}
```

UPDATE WEIGHTS FONKSİYONU

Bu fonksiyon asıl öğrenmenin gerçekleştiği yerdir. Burada asıl Her eğitim örneği için, model tahmin edilen çıktıyı hesaplar; bunun için ağırlıklı toplam ve adım fonksiyonu kullanılır. Tahmin edilen çıktı ile gerçek hedef çıktıyı karşılaştırılır. Bu hata, perseptronun doğru

sınıflandırmadan ne kadar uzak olduğunu gösterir. Eğer perseptron yanlış tahminde bulunursa ağırlıklara hata ve giriş değerinin çarpımı eklenir. Bias de üzerinde hatayı ekleyerek güncellenir. Eğer hata 0 ise hiç güncelleme yapılmaz ve değerler aynı kalır.

```
void UpdateWeights(int j){
    double error = ts[j].output - CalcOutPut(j);
    totalError += Mathf.Abs((float)error);
    for (int i = 0; i < weights.Length; i++)
    {
        weights[i] =  weights[i] + error* ts[j].input[i];
    }
    bias += error;
}
```

CALCULATE OUTPUT FONKSİYONLARI

Bu projede aynı isimli iki fonksiyon var. İkisi de çıktıyı hesaplamaktadır. Biri bir indisi parametre alarak ağırlıklı toplama fonksiyona o indisteki setin giriş ve ağırlık değerlerini gönderirken, ikincisi ise giriş değerleri parametre olarak alıp kendisi bir input listesi oluşturduktan sonra o listeyi ve ağırlıkları ağırlıklı toplama fonksiyona gönderir. Sonucun 0'dan büyük olup olmamasına bağlı olarak da 0 veya 1 değerini döndürür. Bu fonksiyon bir aktivasyon fonksiyonudur.

```
double CalcOutPut(int i){
    double dp = DotProductBias(weights, ts[i].input);
    if (dp>0) return 1;
    return 0;
}
 double CalcOutPut(double i1, double i2){
    double[] input = new double[] {i1, i2};
    double dp = DotProductBias(weights, input);
    if(dp>0) return 1;
    return 0;
}
```

DOT PRODUCT BIAS FONKSİYONU

Bu fonksiyon ağırlıklı toplamayı vektör dot product şeklinde gerçekleştirir. Bunu yapabilmesi için önce iki vektörün de boş olmaması ve aynı uzunlukta olması bekleniyor. Aksi takdirde -1 değeri döndürür ve aktivasyon fonksiyonumuz 0 olarak yorumlar. Girişleri ağırlıklarıyla çarıpıp topladıktan sonra en sonda sapma değerini (bias) ekleyerek geri döndürür.

```
double DotProductBias(double[] v1, double[] v2){
    if(v1==null || v2==null)
        return -1;

    if(v1.Length != v2.Length)
        return -1;
    double d=0;
    for (int i = 0; i < v1.Length; i++)
    {
        d+= v1[i] * v2[i];
    }
    d+= bias;
    return d;
}
```

VEYA KAPISI

Veya kapısını modellemek için kodta hiçbir değişiklik yapmaya gerek yok. Sadece unitye girip perseptron nesnesinde bulunan öğrenme set (ts) değişkenin içeriklerini veya kapısına uygun olacak şekilde güncellemek yeterlidir. Bu şekilde eğittiğinizde yine doğru sonuçlar üretecektir.