

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2021–2022 FALL SEMESTER**

**PROJECT-1 REPORT**

**(Arrays, Matrices, Methods, Random Numbers)**

**DELIVERY DATE**

07/12/2021

**PREPARED BY**

05200000099, Arda Bozkurt

05190000106, Duhan Boblanlı

05190000062, Adem Soydan

İçindekiler

[1) POINTS IN A 2D PLANE 2](#_Toc87438605)

[1.a Rastgele Nokta Üretimi 2](#_Toc87438606)

[1.a.1 Kodlar 2](#_Toc87438607)

[1.a.2 Ekran görüntüleri 2](#_Toc87438608)

[1.a.3 Açıklama 2](#_Toc87438609)

[1.b Uzaklık Matrisi 2](#_Toc87438610)

[1.b.1 Kodlar 2](#_Toc87438611)

[1.b.2 Ekran görüntüleri 2](#_Toc87438612)

[1.b.3 Açıklama 2](#_Toc87438613)

[1.c En yakın komşu yöntemi ile dolaşma 2](#_Toc87438614)

[1.c.1 Kodlar 2](#_Toc87438615)

[1.c.2 Ekran görüntüleri 2](#_Toc87438616)

[1.c.3 Açıklama 2](#_Toc87438617)

[2) DEVELOPING A SIMPLE ARTIFICIAL NEURON and CLASSIFICATION 2](#_Toc87438618)

[2.a Neuron (Sinir Hücresi) sınıfı kaynak kodu 2](#_Toc87438619)

[2.b Eğitim 2](#_Toc87438620)

[2.b.1 Kaynak Kod 2](#_Toc87438621)

[2.b.2 Sonuçlar/Ekran görüntüleri 3](#_Toc87438622)

[2.b.3 Açıklama 3](#_Toc87438623)

[2.c Test 3](#_Toc87438624)

[2.c.1 Test Verisi 3](#_Toc87438625)

[2.c.2 Kaynak Kod 3](#_Toc87438626)

[2.c.2 Sonuçlar/Ekran görüntüleri 3](#_Toc87438627)

[2.c.3 Açıklama 3](#_Toc87438628)

[Öz değerlendirme Tablosu 3](#_Toc87438629)

# 1) POINTS IN A 2D PLANE

//Microsoft Visual Studio, Community edition, C#

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

public static double[,] MatrisOlustur(double Genislik, double Yukseklik, Random random)

{

Console.WriteLine(" Kaç tane nokta oluşturulacağını giriniz ");

int n = Convert.ToInt32(Console.ReadLine());

double[,] noktalarMatrisi = new double[n, 2];

for (int i = 0; i < n; i++)

{

double x = random.NextDouble() \* Genislik;

double y = random.NextDouble() \* Yukseklik;

noktalarMatrisi[i, 0] = x;

noktalarMatrisi[i, 1] = y;

}

return noktalarMatrisi;

}

### 1.a.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.a.3 Açıklama

//Kullanılan veri yapıları ve algoritmanın kısaca anlatımını burada gerçekleştiriniz

Algoritmada kullanıcıdan aldığımız x ve y değer aralığı, random sınıfı ve oluşturulan noktaları saklayacağımız iki boyutlu liste kullanılmaktadır. X ve y değerleri double şeklinde kullanıcıdan alınır ve random sınıfı kullanılarak oluşturulan 0.0 ve 1.0 arasındaki sayı değerleri ile çarpılır ki oluşturulacak noktanın x ve y koordinatları belirlenebilsin. Daha sonrasında koordinatları belli olan noktaların 2 boyutlu liste içinde saklanması için liste indekslerine noktalar tek tek eklenir.

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

public static double[,] UzaklikMatrisiOlustur(double[,] noktalarMatrisi)

{

int n = noktalarMatrisi.Length / 2;

double[,] uzaklikMatrisi = new double[n, n];

for (int i = 0; i < n; i++)

{

for (int k = 0; k < n; k++)

{

double xlerFarki = noktalarMatrisi[i, 0] - noktalarMatrisi[k, 0];

double ylerFarki = noktalarMatrisi[i, 1] - noktalarMatrisi[k, 1];

double xlerfarkiKaresi = xlerFarki \* xlerFarki;

double ylerFarkiKaresi = ylerFarki \* ylerFarki;

double sonuc = Math.Sqrt(xlerfarkiKaresi + ylerFarkiKaresi);

uzaklikMatrisi[i, k] = sonuc;

}

### }

### 1.b.2 Ekran görüntüleri

siyah, elektronik eşyalar içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.b.3 Açıklama

Noktalar matrisinin içindeki noktaların birbirleri arasındaki uzaklığı öklid teoremi kullanarak hesaplar. Hesapladığı uzaklıkları yeni oluşturduğu nokta sayısı kadar satırı ve sütunu olan yeni bir iki boyutlu listeye ekler.

## 1.c En yakın komşu yöntemi ile dolaşma

### 1.c.1 Kodlar

public static void NoktalariDolas(double[,] uzaklikmatrisi, Random random, List<int> basRandomNoktalar, ref int turSayisi)

{

Console.WriteLine("Tur Sayısı = " + turSayisi);

turSayisi++;

int n = (int)Math.Sqrt(uzaklikmatrisi.Length);

int randomNoktaIndex = random.Next(0, n);

while (basRandomNoktalar.Contains(randomNoktaIndex))

{

randomNoktaIndex = random.Next(0, n);

}

basRandomNoktalar.Add(randomNoktaIndex);

Console.WriteLine("Başlangıç Noktamız = " + randomNoktaIndex);

//Console.Write(randomNoktaIndex + "Random Index");

double[] uzakliklarDizisi = new double[n];

double[,] enYakinNoktaIndexVeUzaklik = null;

int enYakinNoktaIndex = 0;

double uzaklik;

int[] gidilenNoktlalar = new int[n];

// boş olan dizi elemanları sıfır olduğu için

for (int i = 0; i < gidilenNoktlalar.Length; i++)

{

gidilenNoktlalar[i] = n + 5;

}

double[] uzakliklar = new double[n];

gidilenNoktlalar[0] = randomNoktaIndex;

for (int j = 0; j < n - 1; j++)

{

// uzaklik dizisini doldurmak icin

for (int i = 0; i < n; i++)

{

if (j == 0)

{

uzakliklarDizisi[i] = uzaklikmatrisi[i, randomNoktaIndex];

}

else

{

uzakliklarDizisi[i] = uzaklikmatrisi[i, enYakinNoktaIndex];

}

}

enYakinNoktaIndexVeUzaklik = EnYakinNoktayiBul(n, uzakliklarDizisi, gidilenNoktlalar);

enYakinNoktaIndex = (int)enYakinNoktaIndexVeUzaklik[0, 0];

gidilenNoktlalar[j + 1] = enYakinNoktaIndex;

uzakliklar[j] = enYakinNoktaIndexVeUzaklik[0, 1];

uzaklik = enYakinNoktaIndexVeUzaklik[0, 1];

}

Console.WriteLine("Noktalara uğranma sırası: ");

for (int i = 1; i < gidilenNoktlalar.Length; i++)

{

if (!(i == gidilenNoktlalar.Length - 1))

{

Console.Write(gidilenNoktlalar[i] + " - ");

}

else

{

Console.Write(gidilenNoktlalar[i]);

}

}

double toplamUzaklik = 0.0;

foreach (double item in uzakliklar)

{

toplamUzaklik += item;

}

Console.WriteLine("");

Console.WriteLine("Gidilen yolun uzunluğu = " + Convert.ToDouble(String.Format("{0:0.0}", toplamUzaklik)) + "\n");

}

public static double[,] EnYakinNoktayiBul(int n, double[] uzakliklarDizisi, int[] gidilenNoktalar)

{

double minimunUzaklik = 500;

int minUzaklikIndex = n + 1;

for (int i = 0; i < n; i++)

{

if (uzakliklarDizisi[i] == 0)

{

continue;

}

else

{

if (uzakliklarDizisi[i] < minimunUzaklik && !gidilenNoktalar.Contains(i))

{

minimunUzaklik = uzakliklarDizisi[i];

minUzaklikIndex = i;

}

}

}

double[,] indexVeUzaiklik = new double[1, 2];

indexVeUzaiklik[0, 0] = minUzaklikIndex;

indexVeUzaiklik[0, 1] = minimunUzaklik;

return indexVeUzaiklik;

### }

### 1.c.2 Ekran görüntüleri

### metin içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.c.3 Açıklama

10 kez yapılacak bu işlemde ilk başta 0 ile nokta sayısı arasında bir sayı seçilir. Seçilen bu sayı noktaları tuttuğumuz listedeki o noktayı belirten indeks sayısıdır. Rastgele seçilen bu noktaya en yakın nokta uzaklık matrisi üzerinden aranır ve bulunduğunda en yakın olan noktanın indeksi kaydedilir. Daha sonrasında indeksi kaydedilen noktaya en yakın ve daha önce uğranmamış nokta uzaklık matrisi içinde aranır ve bu işlem bütün noktalara uğranılana kadar devam eder. En sonra uğranılan noktaları kaydettiğimiz liste sırasıyla yazdırılır. Böylece noktalara uğrama sıramız belli olur, uzaklık matrisindeki verileri bu sıraya göre toplayarak da gidilen yolun uzunluğu belirlenir. Bu değerler kullanıcın görmesi için konsola yazdırılır.

# 2) DEVELOPING A SIMPLE ARTIFICIAL NEURON and CLASSIFICATION

//Microsoft Visual Studio, Community edition, C#

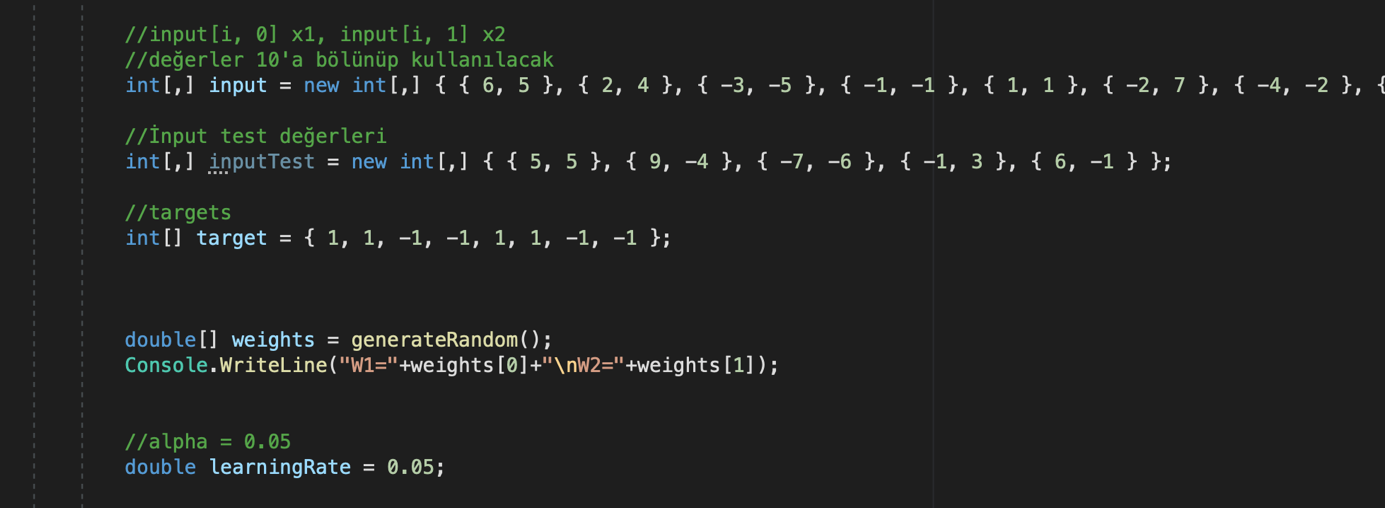
## 2.a Neuron (Sinir Hücresi) sınıfı kaynak kodu

Neuron sınıfındaki fonksiyonlarca kullanılacak 2 boyutlu ve 1boyutlu listeler -->

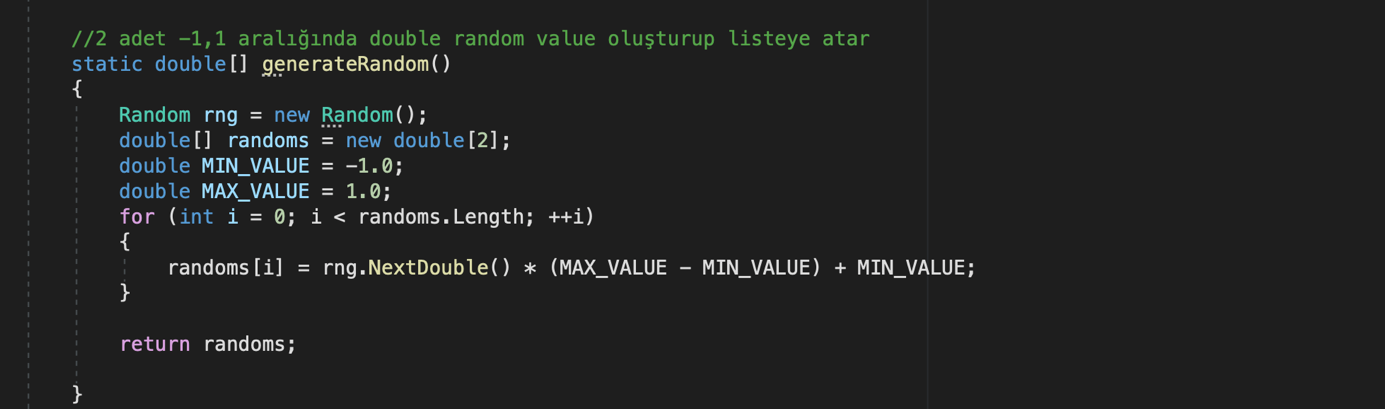
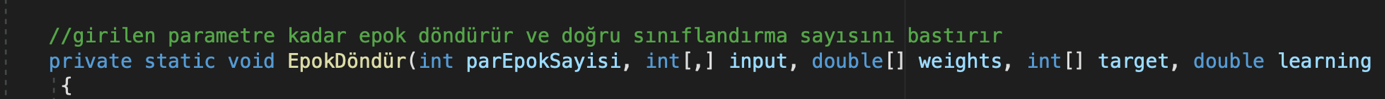
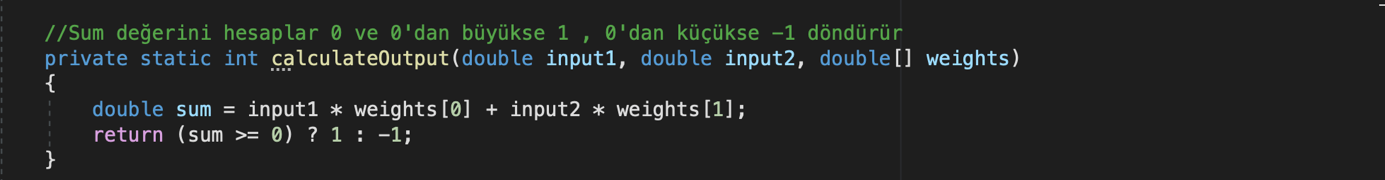
input ,inputTest,Target

GenerateRandom methodu ile oluşturulan weights listesi -->

2 adet (-1,1) aralağında double değer barındırır.



Neuron Sınıfı Fonksiyonları



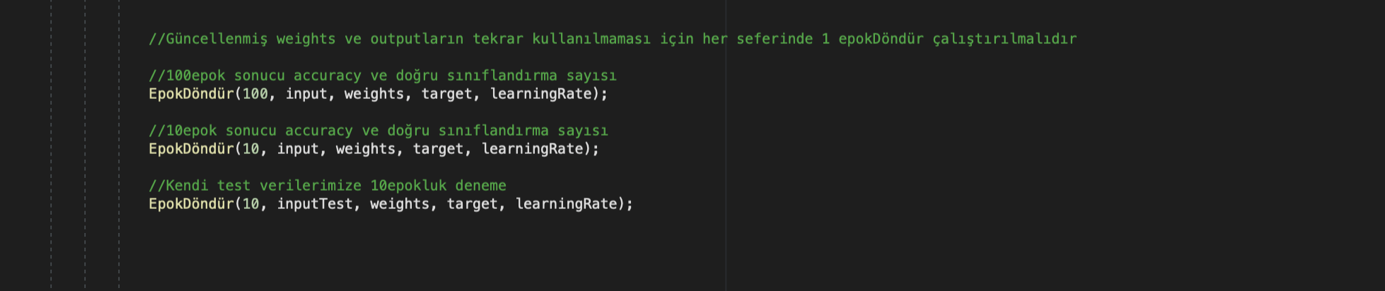
## 2.b Eğitim

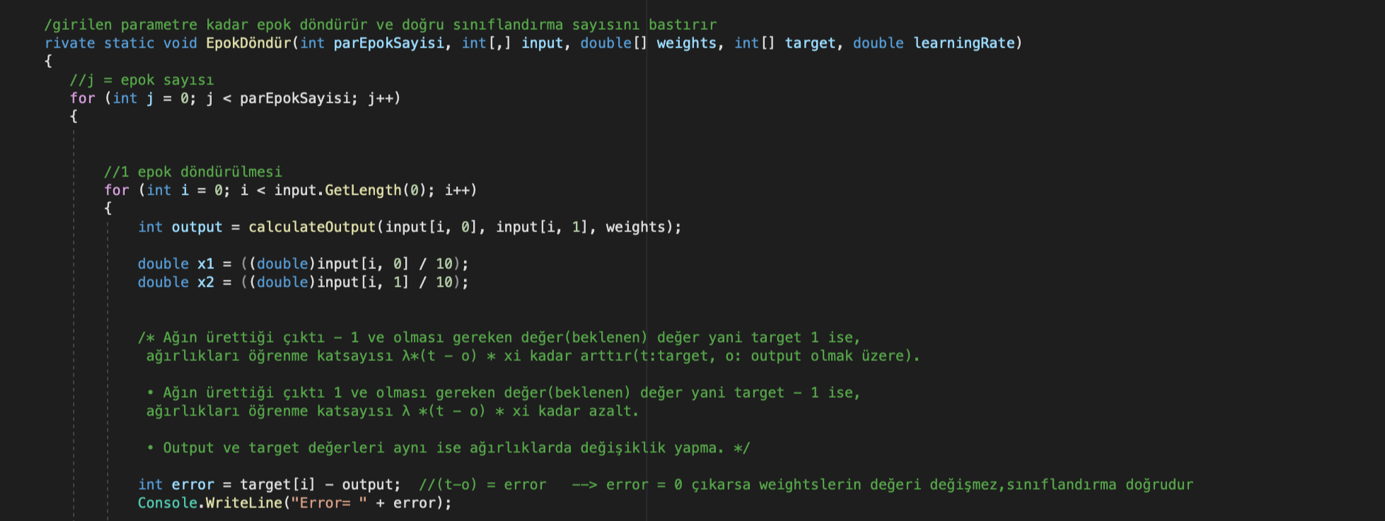
### 2.b.1 Kaynak Kod

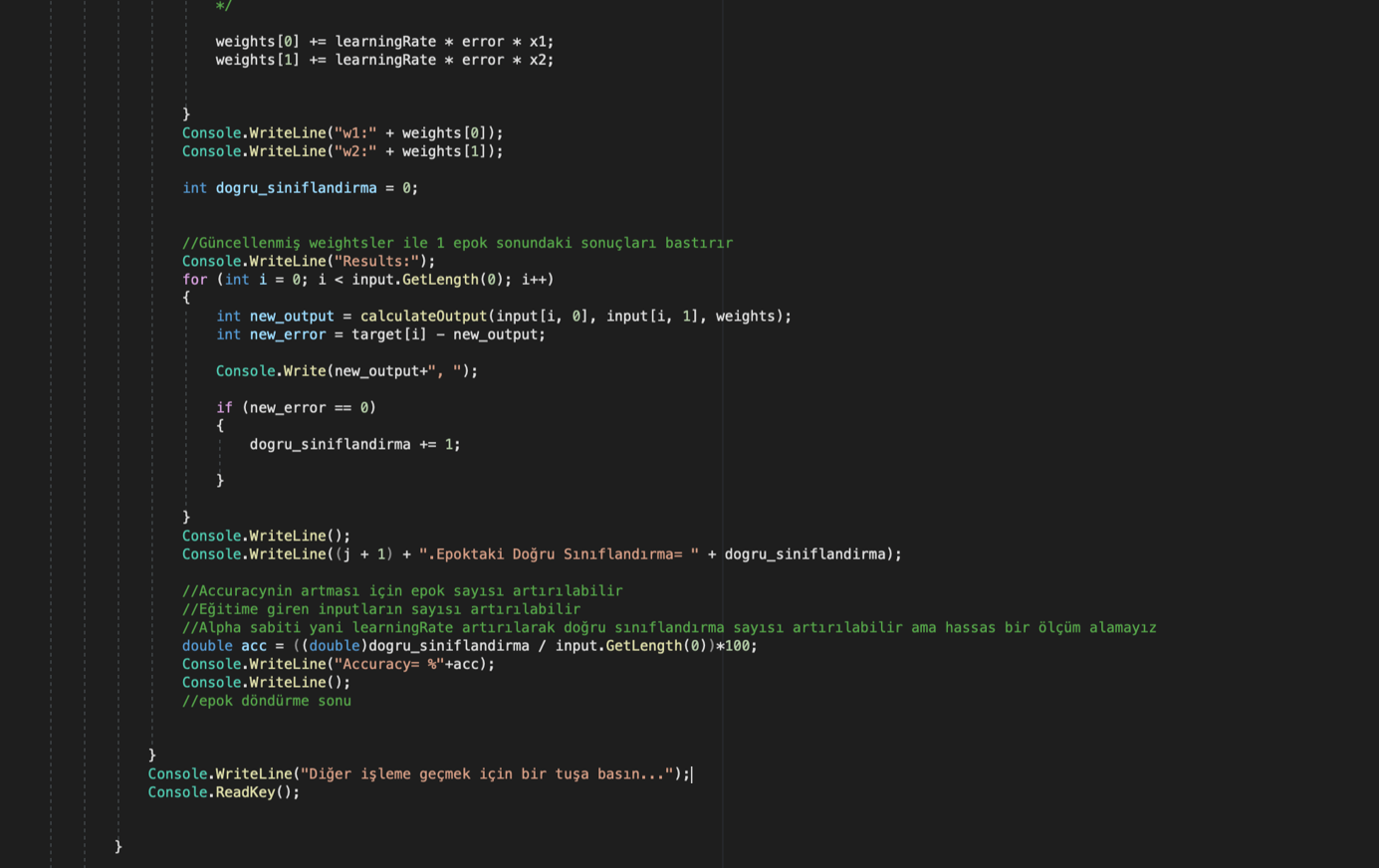
Epok döndürün main method'da çağrılması

Not: 100 epok ve 10 epok eğitimleri için her seferinde EpokDöndür 1 kere çağrılmalıdır. Aksi taktirde güncellenmiş weights değerleri ile işlem yapılır.

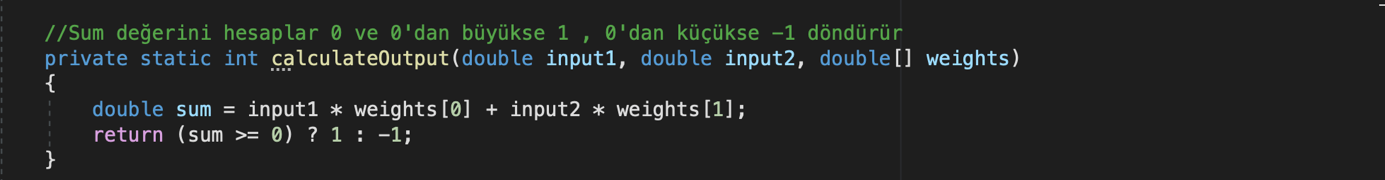
Örnek olarak 100epok üzerine 10 epok daha eğitim yapılarak 110 epokluk sonuç çıkar.

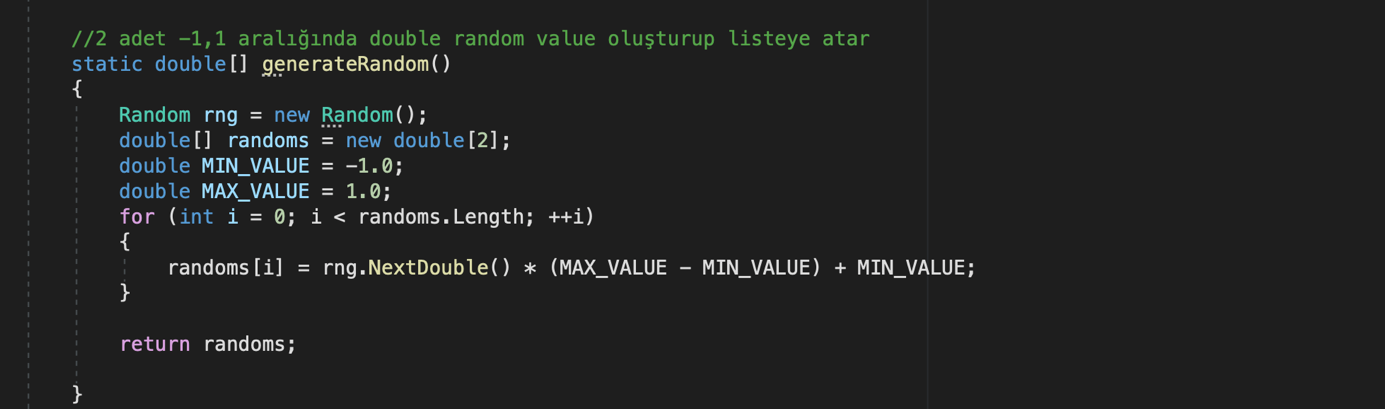






Epok Döndür methodu içinde calculateOutput methodu çağrılır



CalculateOutput methodu içinde de generateRandom methodunda üretilen weights değerleri kullanılır.

### 

### 2.b.2 Sonuçlar/Ekran görüntüleri

10 epok eğitim sonucu örnek ekran çıktıları

### 

### 2.b.3 Açıklama

Bize kullanmamız için verilen input değerleri ve generateRandom() ile oluşturulan weights değerleri calculateOutput fonksiyonunda x1.w1 , x2\*w2 olacak şekilde sırasıyla çarpılır ve sonuç 0'dan büyükse output değerimiz 1 , küçük ise -1 olarak belirlenir. Çıkan output değeri target değeri ile karşılaştırılır. Eğer ikisi de aynı ise weightslerde değişiklik yapılmaz. Target değerinden az çıkarsa weightslere artırma işlemi , fazla çıkarsa weightslere azaltma işlemi yapılır. Bu da algoritmada error değişkeninde hesaplanır. Target - output ikisinin eşit olma durumunda 0 çıkarak . Weightler üzerinde toplamanın etkisiz elemanı olarak işlem yapar. (Target - output) 2 ya da -2 çıkma durumlarında weightslere gerekli artırma ve azalma işlemleri yapılır. Tüm inputlar kullanılana kadar işleme devam edilir ve 1 epok sonunda en iyi sonucu veren weights değerleri bulunmuş olur. Epok işlemlerine devam edildikçe input değerleri en uygun weigths değerleri ile çarpılarak targetdaki değerlere yaklaşmaya başlar ve accuracy oranımız artar.

## 2.c Test

### 2.c.1 Test Verisi

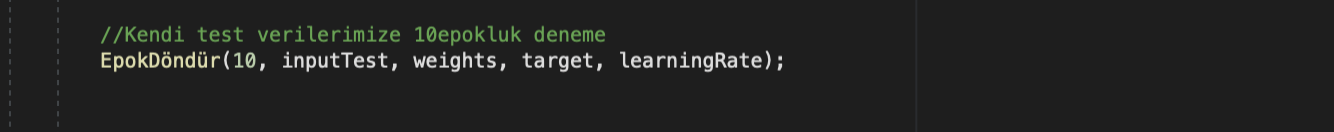
### //İnput test değerleri

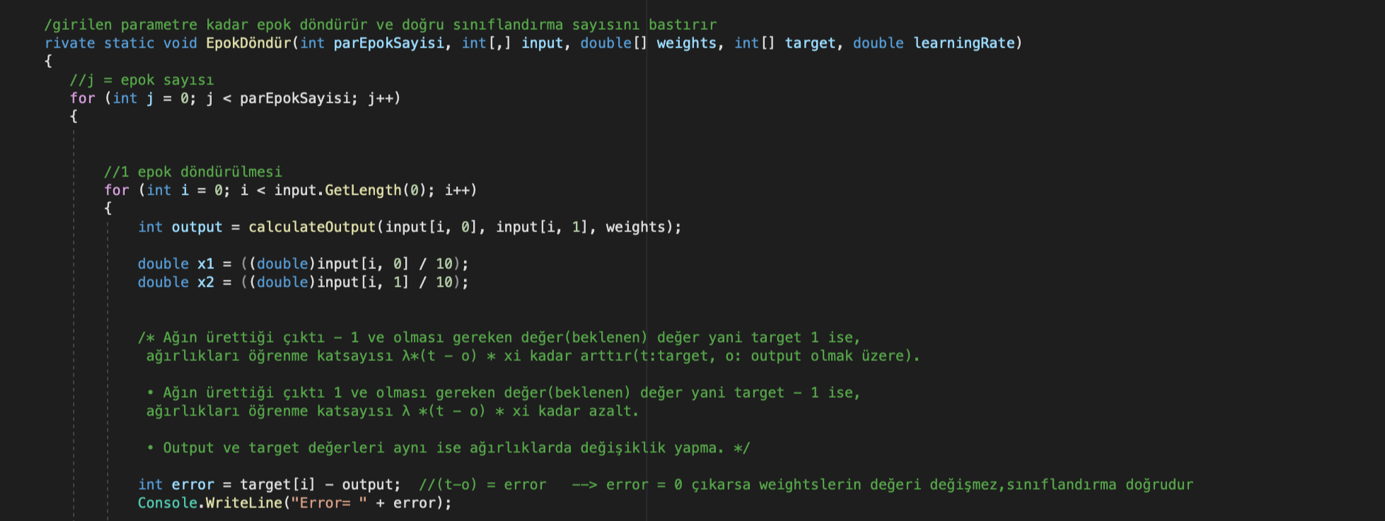
### int[,] inputTest = new int[,] { { 5, 5 }, { 9, -4 }, { -7, -6 }, { -1, 3 }, { 6, -1 } };

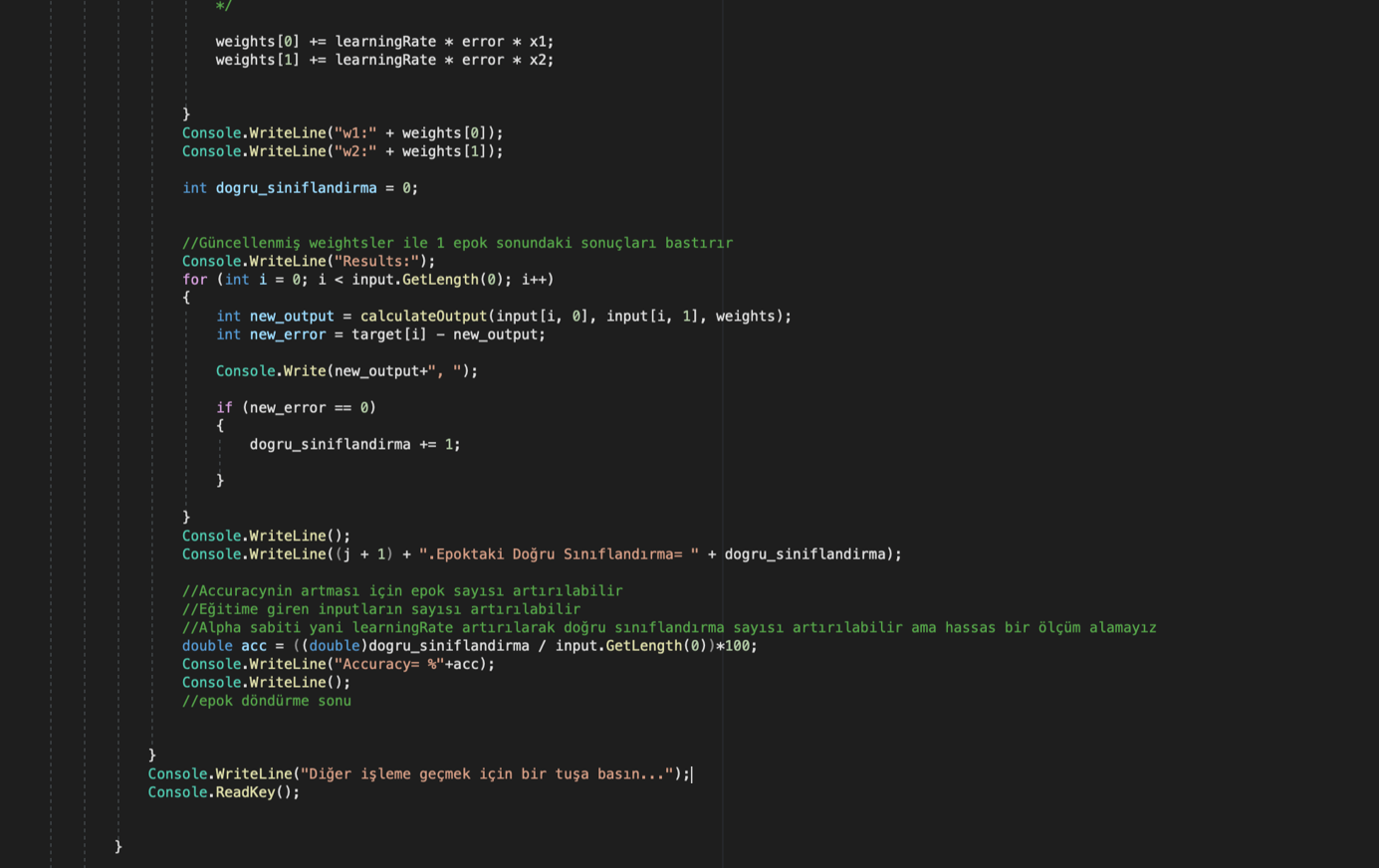
### 2.c.2 Kaynak Kod

EpokDöndür methodu bizim verdiğimiz inputTest listesi ile çalıştırılır.

Önceden eğitilen güncellenmiş weightsler kullanılmak isteniyorsa EpokDöndür methodu 2. kez çağırılıyor demektir. Yani başlangıçta verilen inputlar ile 10 epokluk eğitimden sonra , bizim inputlarımız ile 10 epokluk bir eğitim daha yapılır. Böylece eğittilmiş model test edilir.







### 2.c.2 Sonuçlar/Ekran görüntüleri

### 

### 2.c.3 Açıklama

# 1-Accuracynin artması için epok sayısı artırılabilir.

# 2-Eğitime giren inputların sayısı artırılabilir.

# 3-Alpha sabiti yani learningRate artırılarak doğru sınıflandırma sayısı artırılabilir ama hassas bir ölçüm alamayız. Çünkü weightslerin güncellenme aralığı büyür.

4-Kodda yorum satırına alınan -= 'li kısım çıkartılarak azaltma ve artırma işlemleri tek bir kod parçasında += ile yapıldığında accuracy'de artma gözlemledik. Çünkü örneğin (target - outputun) -2 çıkması durumunda iki - birbirini götürerek sadece x1 ve x2 değerlerinin negatif olması durumuna göre azaltma yapılıyordu. Yani buradaki -= azaltma yapılması için ayarlanan if koşulunda artırma işlemi yaparak hataya yol açıyordu. Dolayısıyla (target - output) durumlarına özel if blokları açmak yerine

alttaki kodu kullanarak error ve x1 ,x2 değerlerinin negatif pozitifliğine göre otomatik olarak azaltma, artırma işlemleri gerçekleştirilmiş oldu ve hata oranı azaldı.

weights[0] += learningRate \* error \* x1;

weights[1] += learningRate \* error \* x2;

/\* else if (error == -2)

{

weights[0] -= learningRate \* error \* x1;

weights[1] -= learningRate \* error \* x2;

} \*/

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 10 | 10 | Noktalar istenilen şekilde oluşturulup listelerde saklandı. |
| 1.b | 10 | 10 | Uzaklık matrisi dökümana uygun olarak çalışır şekilde hazırlanıp, örnek tabloya benzer şekilde konsola yazdırıldı. |
| 1.c | 15 | 15 | En yakın komşuya uğrama algoritması dökümana uygun olarak ve çalışır şekilde yazıldı. İstenilen veriler konsola yazdırıldı. |
| 2.a | 15 | 15 | **Bir Neuron sınıfı** oluşturuldu. Girdiler ve ağırlıkları tutmak için **uygun veri yapıları tanımlandı. Hesaplamaları ve gerekli işlemleri yapan metotlar oluşturuldu.** |
| 2.b | 10 | 10 | 10 epok ve 100 epok (epoch) sonunda veri seti üzerindeki doğruluk (accuracy) değerleri hesaplandı ve ekrana bastırıldı. |
| 2.c | 10 | 10 | Eğitim verileri dışında farklı 5 tane test verisi oluşturuldu. Test verileri ve başarı değerleri rapora eklendi. Başarı değerlerini artırabilmek için neler yapabileceği rapora yazıldı. |
| Rapor | 20 | 20 | Rapor yazılan algoritmaların işleyişini anlatarak hazırlandı |
| Öz değerlendirme Tablosu | 10 | 10 | Öz değerlendirme tablosu rapor ile birlikte dolduruldu. |

**Açıklama kısmında yapıldı, yapılmadı bilgisi veya hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır.**