

**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**

06/12/2021

**PREPARED BY**

05180000075, Emircan BAHADIR

05180000063 , Burak KIZILAY

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

We used the Visual Studio Version 2019. And we used C# for programming language.

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

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

{

double pointHeight = Math.Truncate(random.NextDouble() \* int.Parse(height) \* 100) / 100;

double pointWidth = Math.Truncate(random.NextDouble() \* int.Parse(width) \* 100) / 100;

noktalar[i, 0] = pointWidth;

noktalar[i, 1] = pointHeight;

Console.WriteLine("x: " + pointWidth + " " + "y: " + pointHeight);

}

### Console.WriteLine();

### 1.a.2 Ekran görüntüleri

### 

### 1.a.3 Açıklama

Double tipinde pointHeight ve pointWidth olacak şekilde iki adet random nokta değeri elde eden değişken oluşturduk. Bunu bir for döngüsünün içinde yazdık. Üretilen random değerleri height ve width değerleriyle de çarptıktan sınra 100 ile çarpıp 100 ile bölerek formatın düzgün çıkmasını sağladık. Kod bloğumuzun sonunda da ekrana width ve height değerlerini sırayla yazdırıyor.

## 

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

static double uzaklıkhesap(double x1, double y1, double x2, double y2)

{

double hesap = Math.Sqrt(Math.Pow(x2 - x1, 2) + Math.Pow(y2 - y1, 2));

hesap = Math.Truncate(hesap \* 10) / 10;

return hesap;

}

static Random random = new Random();

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

{

for (int j = 0; j < int.Parse(n); j++)

{

double temp = uzaklıkhesap(noktalar[i, 0], noktalar[i, 1], noktalar[j, 0], noktalar[j, 1]);

uzaklıklar[i, j] = temp;

string tempStr = String.Format("{0:000.00}", temp);

Console.Write(tempStr + " | ");

}

Console.WriteLine();

### }

### 1.b.2 Ekran görüntüleri

### 

### 1.b.3 Açıklama

Bir adet uzaklıkhesap fonksiyonu ile Uzaklık Matrisi hesaplamanın matematiksel formülünü tanımladık.

Bu kod bloğunda iç içe iki adet for döngüsünden yararlanarak uzaklık matrisi elde ettik. Double tipinde bir temp değişkeni oluşturarak bu değişkene uzaklıkhesap fonksiyonunu çağırarak, bu fonksiyona, önceden tanımlanmış noktaları atamaktadır.

## 1.c En yakın komşu yöntemi ile dolaşma

### 1.c.1 Kodlar

for (int i = 0; i < 10; i++)

{

int başlangıç = random.Next(0, int.Parse(n));

Console.WriteLine((i + 1) + ". tur");

Console.Write("Uğradığı noktalar: " + başlangıç);

double totalUzaklık = 0;

List<int> gezilmisler = new List<int>();

gezilmisler.Add(başlangıç);

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

{

var uzaklik = (double)int.MaxValue;

int node = 0;

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

{

if (uzaklıklar[başlangıç, k] < uzaklik && !gezilmisler.Contains(k))

{

uzaklik = uzaklıklar[başlangıç, k];

node = k;

}

}

Console.Write("-" + node);

gezilmisler.Add(node);

başlangıç = node;

totalUzaklık += uzaklik;

}

Console.WriteLine("\nUzaklıklar toplamı; " + Math.Truncate(totalUzaklık \* 100) / 100);

}

### 1.c.2 Ekran görüntüleri

### 

### 1.c.3 Açıklama

Bu kod bloğunda bir for döngüsü içerisinde dolaşarak dolaştığı noktaları bir listeye atayan ve sonrasında başka bir for döngüsü ile bu noktalar arası uzaklıkları da alarak toplamıyla birlikte gezilenlerin olduğu listeyi ekrana yazdıran bir yapı oluşturmuş olduk.

# 2) DEVELOPING A SIMPLE ARTIFICIAL NEURON and CLASSIFICATION

## We used the Visual Studio Version 2019. And we used C# for programming language.

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

double[,] dataSet = { { .6, .5, 1 }, { .2, .4, 1 }, { -.3, -.5, -1 }, { -.1, -.1, -1 }, { .1, .1, 1 }, { -.2, .7, 1 }, { -.4, -.2, -1 }, { -.6, .3, -1 } };

double[,] dataTest = { { .3, .1, -1 }, { -.5, .4, 1 }, { -.4, -.2, 1 }, { .3, -.7, 1 }, { -.2, .3, -1 } };

double[] weights;

static readonly double lambda = 0.05;

public Neuron()//rastgele weightler atanır

{

weights = new double[4];

Random random = new Random();

weights[0] = random.NextDouble() \* 2 - 1;

weights[1] = random.NextDouble() \* 2 - 1;

weights[2] = random.NextDouble() \* 2 - 1;

weights[3] = random.NextDouble() \* 2 - 1;

}

## 2.b Eğitim

### 2.b.1 Kaynak Kod

public void Train()

{

double accurate = 0;

Console.WriteLine("Eğitim aşaması");

for (int i = 0; i < 100; i++)

{

accurate = 0;

for (int j = 0; j < dataSet.GetLength(0); j++)

{

double v = dataSet[j, 0] \* weights[0] + dataSet[j, 1] \* weights[1];

int y = 1;

if (v < 0.5)

{

y = -1;

}

if (y == -1 && dataSet[j, 2] == 1)//hedefe ulaşılıp ulaşılamama durumuna göre weight üzerinde oynama yapılır ya da accurate olanlara eklenir

{

weights[0] += lambda \* 2 \* dataSet[j, 0];

weights[1] += lambda \* 2 \* dataSet[j, 1];

}

else if (y == 1 && dataSet[j, 2] == -1)

{

weights[0] -= lambda \* 2 \* dataSet[j, 0];

weights[1] -= lambda \* 2 \* dataSet[j, 1];

}

else

{

accurate += 1;

}

}

if (i == 9)//10.epokta yazdırılması gerekeni yazdırır

{

Console.WriteLine("10 epok sonu accuracy değeri: %" + (accurate / dataSet.GetLength(0) \* 100));

}

}

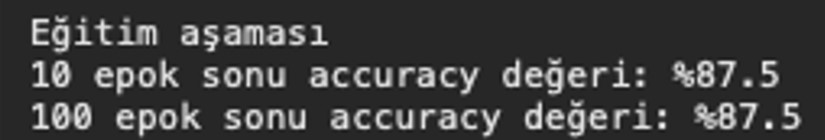
Console.WriteLine("100 epok sonu accuracy değeri: %" + (accurate / dataSet.GetLength(0) \* 100));

### }

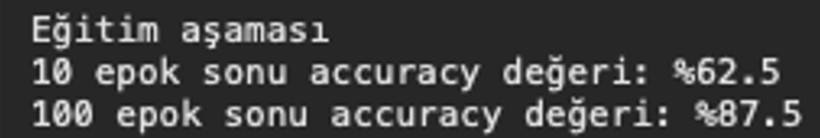
### 2.b.2 Sonuçlar/Ekran görüntüleri

### 

--------------------------------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------------------------------



### 2.b.3 Açıklama

Bu kod bloğunda void tipinde bir Train() tanımladık. Başta dictionary kullanmayı denedik fakat çok uygun olmadığına karar verip array list kullanımı gerçekleştirdik. İç içe iki for döngüsü ile beraber kullandığımız if blokları ile eğitim yapısını oluşturduk. En sonunda da 10 ve 100 epok için değerleri ekrana yazdırdık.

## 2.c Test

### 2.c.1 Test Verisi

double[,] dataTest = { { .3, .1, -1 }, { -.5, .4, 1 }, { -.4, -.2, 1 }, { .3, -.7, 1 }, { -.2, .3, -1 } };

### 2.c.2 Kaynak Kod

public void Test()

{

double accurate = 0;

Console.WriteLine("Test aşaması");

for (int i = 0; i < 100; i++)

{

accurate = 0;

for (int j = 0; j < dataTest.GetLength(0); j++)

{

double v = dataTest[j, 0] \* weights[2] + dataTest[j, 1] \* weights[3];

int y = 1;

if (v < 0.5)

{

y = -1;

}

if (y == -1 && dataTest[j, 2] == 1)

{

weights[2] += lambda \* 2 \* dataTest[j, 0];

weights[3] += lambda \* 2 \* dataTest[j, 1];

}

else if (y == 1 && dataTest[j, 2] == -1)

{

weights[2] -= lambda \* 2 \* dataTest[j, 0];

weights[3] -= lambda \* 2 \* dataTest[j, 1];

}

else

{

accurate += 1;

}

}

if (i == 9)

{

Console.WriteLine("10 epok sonu accuracy değeri: %" + (accurate / dataTest.GetLength(0) \* 100));

}

}

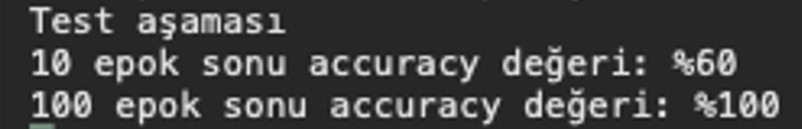
Console.WriteLine("100 epok sonu accuracy değeri: %" + (accurate / dataTest.GetLength(0) \* 100));

}

### 2.c.2 Sonuçlar/Ekran görüntüleri

### 

--------------------------------------------------------------------------------------------------------------------------------------



### 2.c.3 Açıklama

Yapılabilecek işlemlerin en etkilisi, uygulanacak olan epok sayısının arttırılması olacaktır. Daha fazla epok tutarlılık oranını arttırmaya yarayacaktır.

Bir başka yol da ağırlıkları rastgele atamak yerine x1, x2 ve target’ın kullanıldığı bir formülle atamak accuracy değerimizi arttırmamızda yardımcı olabilir.

Bir başka yol ise veri setimizi değiştirmek olabilir.

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 10 | 10 | Yapıldı |
| 1.b | 10 | 10 | Yapıldı |
| 1.c | 15 | 15 | Yapıldı |
| 2.a | 15 | 15 | Yapıldı |
| 2.b | 10 | 10 | Yapıldı |
| 2.c | 10 | 7 | Okuduğumuzu geç anlamamız ve bu şıkkın son güne kalması nedeniyle aşırı etkin bir blok oluşturamadık. Hem 2.b hem 2.c için tek bir fonksiyon yazılabilirdi. |
| Rapor | 20 | 20 | Yapıldı |
| Öz değerlendirme Tablosu | 10 | 10 | Yapıldı |

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