

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2023–2024 FALL SEMESTER**

**PROJECT-1 REPORT**

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

**DELIVERY DATE**

19/11/2023

**PREPARED BY**

05210000261, Bahrihan Torpil

05210000260, Kutlu Çağan Akın

05210000292, Beyza Çelik

İçindekiler

[1) KARAYOLLARI UZAKLIK HESAPLAMALARI 2](#_Toc149748651)

[1.a Uzaklık Matrisi Oluşturma 2](#_Toc149748652)

[1.a.1 Kodlar 2](#_Toc149748653)

[1.a.2 Ekran görüntüleri 2](#_Toc149748654)

[1.b Verilen İlden Belli Bir Uzaklığa Kadar Olan İllerin ve Uzaklıklarının Listelenmesi 2](#_Toc149748655)

[1.b.1 Kodlar 2](#_Toc149748656)

[1.b.2 Ekran görüntüleri 2](#_Toc149748657)

[1.c Türkiye’deki Birbirine En Yakın İki İlin ve En Uzak İki İlin Bulunması 2](#_Toc149748658)

[1.c.1 Kodlar 2](#_Toc149748659)

[1.c.2 Ekran görüntüleri 2](#_Toc149748660)

[1.d Verilen İlden Verilen Mesafe Kullanılarak En fazla Kaç İl Dolaşılabildiğinin Bulunması 2](#_Toc149748661)

[1.d.1 Kodlar 2](#_Toc149748662)

[1.d.2 Algoritma ve Açıklama 2](#_Toc149748663)

[1.d.3 Ekran görüntüleri 2](#_Toc149748664)

[1.e Matris Şeklinde İllerin Adlarıyla Birlikte Ekrana Listelenmesi 2](#_Toc149748665)

[1.e.1 Kodlar 2](#_Toc149748666)

[1.e.2 Ekran görüntüleri 2](#_Toc149748667)

[2) DEVELOPING A PERCEPTRON MODEL and IMPLEMENTATION of a REGRESSION EXAMPLE 3](#_Toc149748668)

[2.a Neuron (Sinir Hücresi) Sınıfı 3](#_Toc149748669)

[2.a.1 Kaynak Kod 3](#_Toc149748670)

[2.a.2 Açıklama 3](#_Toc149748671)

[2.b Eğitim 3](#_Toc149748672)

[2.b.1 Kaynak Kod 3](#_Toc149748673)

[2.b.2 Açıklama 3](#_Toc149748674)

[2.b.3 Ekran Görüntüleri 3](#_Toc149748675)

[2.c Modelin Görmediği Veriden Sınav Sonucu Tahminleme 3](#_Toc149748676)

[2.c.1 Kaynak Kod 3](#_Toc149748677)

[2.c.2 Sonuçlar/Ekran görüntüleri 3](#_Toc149748678)

[2.d Deneyler 3](#_Toc149748679)

[2.d.1 Kaynak Kod 3](#_Toc149748680)

[2.d.2 Sonuçlar 3](#_Toc149748681)

[Öz değerlendirme Tablosu 4](#_Toc149748682)

# 1) KARAYOLLARI UZAKLIK HESAPLAMALARI

MICROSOFT VISUAL STUDIO , VERSION 17.7.6, C# .NET 4.8

## 1.a Uzaklık Matrisi Oluşturma

### 1.a.1 Kodlar

//İlgili kod parçasını buraya ekleyiniz

// Program başladığında liste ve matrisi otomatik oluşturur.

//----------------------------------

int[][] distances = new int[81][]; // jagged matris

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

{

distances[i] = new int[81];

}

String[] cities = new string[81]; // şehir list

Operations.DistanceOfCities(distances); // il mesafesini textten oku ve jagged matrise at.

String[] insertedCities = Operations.InsertCities(cities); // şehirleri dosyadan oku ve diziye at.

// ---------------------------------

METHODLAR

public static int[][] DistanceOfCities(int[][] distances)

{

string file = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\proje1\\proje1\\ilMesafe.txt";

using (StreamReader sr = new StreamReader(file))

{

string satir;

int i = 0;

while ((satir = sr.ReadLine()) != null)

{

String[] tempList = satir.Split(' '); // string to int dönüşümü

for (int j = 0; j < tempList.Length; j++)

{

distances[i][j] = Convert.ToInt32(tempList[j]);

}

i++;

}

sr.Close();

}

return distances;

}

public static String[] InsertCities(String[] cities)

{

string file = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\proje1\\proje1\\şehirler.txt";

using (StreamReader sr = new StreamReader(file, Encoding.GetEncoding("iso-8859-9")))

{

string row;

int i = 0;

while ((row = sr.ReadLine()) != null)

{

cities[i] = row;

i++;

}

}

return cities;

}

### 1.a.2 Ekran görüntüleri

Indexe göre şehir mesafeleri text dosyası

ekran görüntüsü, kalıp, desen, düzen, dikiş içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.b Verilen İlden Belli Bir Uzaklığa Kadar Olan İllerin ve Uzaklıklarının Listelenmesi

### 1.b.1 Kodlar

public static void RequestedCity(String[] cities, int[][] distances)

{

Console.Write("İl plakasını veya adını giriniz: ");

String cityNum = Console.ReadLine();

Console.Write("O ilden gezilecek mesafeyi giriniz: ");

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

int plateNumber = 1;

if (cityNum.Length <= 2) //plaka kodu en fazla 2 karakter olabilir.

{

int intInput = Convert.ToInt32(cityNum);

if (intInput <= 0 || intInput > 81)

{

Console.WriteLine("Şehir bulunamadı.");

return;

}

else

{

plateNumber = intInput - 1;

}

}

else

{

int i = 0;

while (i < cities.Length)

{

if (cities[i] == cityNum.ToUpper())

{

plateNumber = i;

break;

}

i++;

}

if (plateNumber == 1)

{

Console.WriteLine("Şehir bulunamadı: " + cityNum);

return; // şehir bulunamadığı için fonksiyonu sonlandır.

}

}

Console.WriteLine("İstenilen şehir ve plaka kodu: " + cities[plateNumber] + "," + (plateNumber + 1));

Console.WriteLine("-------------------------------------------");

int counter = 0;

for (int j = 0; j < cities.Length; j++)

{

if (distances[plateNumber][j] <= distance)

{

if (distances[plateNumber][j] != 0)

{

counter++;

Console.WriteLine("Şehir ismi: " + cities[j] + " Uzaklık: " + distances[plateNumber][j]);

}

}

}

Console.WriteLine("Gezilebilecek toplam il sayısı: " + counter);

}

### 1.b.2 Ekran görüntüleri

metin, ekran görüntüsü, yazı tipi, menü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, siyah beyaz, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

İnput1: 35

İnput2: İzMiR(hata kontrollü)

## 1.c Türkiye’deki Birbirine En Yakın İki İlin ve En Uzak İki İlin Bulunması

### 1.c.1 Kodlar

public static void MinDistance(String[] cities, int[][] distances)

{

int min = int.MaxValue;

int minCity1 = 0; // index city

int minCity2 = 0;

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

{

for (int j = 0; j < distances[i].Length; j++)

{

if (distances[i][j] != 0)

{

if (min > distances[i][j])

{

min = distances[i][j];

minCity1 = i;

minCity2 = j;

}

}

}

}

Console.WriteLine("Aralarındaki mesafenin en kısa olduğu 2 il : " + cities[minCity1]

+ " ve " + cities[minCity2] + "\nAralarındaki mesafe: " + min + " km");

}

public static void MaxDistance(String[] cities, int[][] distances)

{

int max = 0;

int maxCity1 = 0; // index city

int maxCity2 = 0;

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

{

for (int j = 0; j < distances[i].Length; j++)

{

if (distances[i][j] != 0)

{

if (max < distances[i][j])

{

max = distances[i][j];

maxCity1 = i;

maxCity2 = j;

}

}

}

}

Console.WriteLine("Aralarındaki mesafenin en uzun olduğu 2 il : " + cities[maxCity1]

+ " ve " + cities[maxCity2] + "\nAralarındaki mesafe: " + max + " km");

}

### 1.c.2 Ekran görüntüleri

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.d Verilen İlden Verilen Mesafe Kullanılarak En fazla Kaç İl Dolaşılabildiğinin Bulunması

### 1.d.1 Kodlar

public static void MaxCitiesGivenDistance(string[] plakaList, int[][] km\_list)

{

Console.WriteLine("Hangi il için bakmak istiyorsunuz(plaka): ");

int ilk\_il\_input = Convert.ToInt32(Console.ReadLine());

int il\_input = ilk\_il\_input;

int gercek\_il\_input = il\_input - 1;

Console.WriteLine("Kaç km içinde bakıyorsunuz: ");

int km\_input = Convert.ToInt32(Console.ReadLine());

int[][] km\_listKopya = CopyJaggedArray(km\_list);

string[] plakaListKopya = new string[plakaList.Length];

Array.Copy(plakaList, plakaListKopya, plakaList.Length);

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

int toplam = 0; // gidilen toplam km

int il\_say = 0;

int min = km\_list.Min(x => x[gercek\_il\_input]);

var result = FindMinValueAndIndex(km\_list[gercek\_il\_input], ziyaretEdilenIller);

int minValue = result.Item1;

int minIndex = result.Item2;

for (int j = 0; j < 81; j++)

{

if (result != null)

{

minValue = result.Item1;

minIndex = result.Item2;

toplam += minValue;

if (toplam < km\_input)

{

// index 0-80 --> plaka kodu 1-81

Console.WriteLine($"{plakaListKopya[il\_input - 1]}' den {plakaListKopya[minIndex - 1]} 'a Uzaklık: {minValue + " km"}");

ziyaretEdilenIller.Add(il\_input);

il\_input = minIndex;

result = FindMinValueAndIndex(km\_listKopya[il\_input - 1], ziyaretEdilenIller);

il\_say++;

Console.WriteLine();

}

}

}

Console.WriteLine($"{plakaListKopya[ilk\_il\_input - 1]}' den {km\_input} km mesafe ile gidilebilen maksimum il sayısı: {il\_say}");

}

static Tuple<int, int> FindMinValueAndIndex(int[] array, List<int> yasaklist)

{

if (array == null || array.Length == 0)

{

return null;

}

int minValue = int.MaxValue;

int minIndex = 0;

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

{

if (array[i] <= minValue && array[i] != 0 && !yasaklist.Contains(i + 1))

{

minValue = array[i];

minIndex = i;

}

}

return new Tuple<int, int>(minValue, minIndex + 1);

}

static int[][] CopyJaggedArray(int[][] originalArray)

{

int[][] copiedArray = new int[originalArray.Length][];

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

{

copiedArray[i] = new int[originalArray[i].Length];

for (int j = 0; j < originalArray[i].Length; j++)

{

copiedArray[i][j] = originalArray[i][j];

}

}

return copiedArray;

}

### 1.d.2 Algoritma ve Açıklama

Bu kod, bir dizi plaka numarası ve her bir plakaya bağlı olarak bir dizi uzaklık (km) içeren bir matris üzerinde işlem yapmaktadır. Kod, girilen bir ilin başlangıç noktası olarak alındığı ve belirli bir maksimum uzaklık içinde gidilebilen diğer illeri bulmaya çalıştığı bir algoritmayı içerir. Bu kodda kullanılan bazı önemli veri yapıları ve fonksiyonlar:

1. int[][] km\_list: İki boyutlu bir jagged array olan km\_list, her bir ilin diğer illere olan uzaklıklarını içerir.

2. string[] plakaList: İndexlerine göre il plakaları içeren bir dizi.

3. List<int> ziyaretEdilenIller: Ziyaret edilen illerin indekslerini içeren bir liste.

4. Tuple<int, int> FindMinValueAndIndex(int[] array, List<int> yasaklist): Verilen bir dizide minimum değeri ve bu değerin indeksini bulan bir fonksiyon.

5. static int[][] CopyJaggedArray(int[][] originalArray): İki boyutlu jagged array'in bir kopyasını oluşturan bir fonksiyon.

### 1.d.3 Ekran görüntüleri

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.e Matris Şeklinde İllerin Adlarıyla Birlikte Ekrana Listelenmesi

### 1.e.1 Kodlar

public static void Random5Cities(String[] cities, int[][] distances)

{

int[] randomCities = new int[5];

Random random = new Random();

for (int i = 0; i < 5; i++)

{

int j = 0;

randomCities[i] = random.Next(81);

while (j < i) // same value control

{

while (randomCities[j] == randomCities[i])

{

randomCities[i] = random.Next(81);

}

j++;

}

}

int[,] citiesLength = new int[5, 5];

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

{

for (int j = 0; j < randomCities.Length; j++)

{

citiesLength[i, j] = distances[randomCities[i]][randomCities[j]];

}

}

Console.Write(" ");

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

{

String b = String.Format("{0, -15}", cities[randomCities[i]]);

Console.Write(b);

}

Console.WriteLine();

Console.WriteLine();

for (int i = 0; i < citiesLength.GetLength(0); i++)

{

String s = String.Format("{0, -15}", cities[randomCities[i]]);

Console.Write(s);

for (int j = 0; j < citiesLength.GetLength(1); j++)

{

String a = String.Format("{0, -15}", citiesLength[i, j]);

Console.Write(a);

}

Console.WriteLine();

Console.WriteLine();

}

}

### 1.e.2 Ekran görüntüleri

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 2) DEVELOPING A PERCEPTRON MODEL and IMPLEMENTATION of a REGRESSION EXAMPLE

MICROSOFT VISUAL STUDIO , VERSION 17.7.6, C# .NET 4.8

## 2.a Neuron (Sinir Hücresi) Sınıfı

### 2.a.1 Kaynak Kod

public class Neuron

{

private double weight1;

private double weight2;

private double learningRate = 0.05;

public Neuron()

{

Random random = new Random();

weight1 = random.NextDouble();

weight2 = random.NextDouble();

### }

### 2.a.2 Açıklama

Derse devamlılık ve Çalışma sürelerini çentikli matris içinde tutan bir liste ve sınav sonuçlarını da tek boyutlu liste içinde tutan 2 nesnemiz var.

Nöron sınıfı içerisinde double veri tipinde 2 ağırlık tutar ve bunları Train methodu ile eğitir.

Nöron sınıfındaki Activate methodu da ağırlıkları Train methodunda sırasıyla çentikli matristeki değerlerle çarpar.

Main sınıftaki Normalize fonksiyonu değerleri istenilen sonuçlar için yapılacak işlemleri içerir. Method overloading yöntemi kullanılmıştır.

CalculateMSE methodu da hata değerini ölçmek için gerekli formül ile verileri kullanarak hata payını bulur.

## 2.b Eğitim

### 2.b.1 Kaynak Kod

public double Activate(double input1, double input2)

{

return input1 \* weight1 + input2 \* weight2;

}

public void Train(double[][] inputSet, double[] targets, int epochs)

{

for (int epoch = 0; epoch < epochs; epoch++)

{

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

{

double output = Activate(inputSet[i][0], inputSet[i][1]);

double error = targets[i] - output;

// Ağırlıkları güncelle

weight1 += learningRate \* error \* inputSet[i][0];

weight2 += learningRate \* error \* inputSet[i][1];

}

}

### }

### 2.b.2 Açıklama



Mean Squared Error (MSE), öğrenme algoritmalarında sıkça kullanılan bir hata ölçüsüdür. Bu ölçü, bir modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu ölçen bir metriktir. MSE, özellikle **regresyon** problemlerinde kullanılır.

MSE'nin formülü:

yazı tipi, metin, el yazısı, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada:

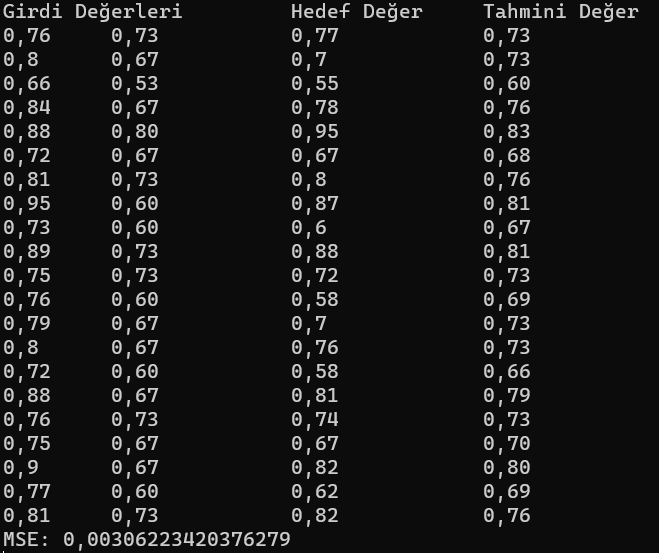
-n veri noktasının toplam sayısını temsil eder.

- y(i), gerçek değeri temsil eder.

- Şapkalı y(i) ise modelin tahmin ettiği değeri temsil eder.

MSE'nin yüksek bir değeri, modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu gösterir. Düşük bir MSE değeri, modelin daha iyi performans gösterdiğini gösterir.

### 2.b.3 Ekran Görüntüleri



## 2.c Modelin Görmediği Veriden Sınav Sonucu Tahminleme

### 2.c.1 Kaynak Kod

// trial inputs

// weight1 and weight2 are trained.

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

{

double output = trialInputSet[i][0] \* weight1 + trialInputSet[i][1] \* weight2;

trialOutputs.Add(output);

}

### }2.c.2 Sonuçlar/Ekran görüntüleri

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

## 2.d Deneyler

### 2.d.1 Kaynak Kod

(Main Class)

int epochs = 10; // 50, 100

neuron.Train(inputSet, targets, epochs, trialInputSet, trialOutputs);

(Neuron Class)

private double weight1;

private double weight2;

private double learningRate = 0.05; // 0.01, 0.025

public Neuron()

{

Random random = new Random();

weight1 = random.NextDouble(); // 0 between 1

weight2 = random.NextDouble();

}

### 2.d.2 Sonuçlar

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Deney 1 | 10 Epok | 50 Epok | 100 Epok | | λ = 0.01 | 0,003694 | 0,003172 | 0,003573 | | λ=0.025 | 0,003081 | 0,003040 | 0,003068 | | λ = 0.05 | 0,003271 | 0,003942 | 0,003087 | | | | | | |  |  |  |  | | --- | --- | --- | --- | | Deney 2 | 10 Epok | 50 Epok | 100 Epok | | λ = 0.01 | 0,003450 | 0,003038 | 0,003079 | | λ = 0.025 | 0,003059 | 0,003128 | 0,003314 | | λ = 0.05 | 0,003486 | 0,003175 | 0,003043 | | | | | |
| Deney 3 | 10 Epok | 50 Epok | 100 Epok | Deney 4 | | 10 Epok | 50 Epok | 100 Epok |
| λ = 0.01 | 0,003714 | 0,003172 | 0,003233 | λ = 0.01 | | 0,003454 | 0,003136 | 0,003573 |
| λ =0.025 | 0,003053 | 0,003310 | 0,003068 | λ = 0.025 | | 0,003231 | 0,003120 | 0,003068 |
| λ = 0.05 | 0,003369 | 0,003942 | 0,003125 | λ = 0.05 | | 0,003762 | 0,003044 | 0,003087 |

Değerler, ağırlıklar rastgele bir değer tuttuğu için deney sonuçları tutarlı olmayabiliyor. Fakat 100 devir dönüldüğünde mse değerimiz daha stabil gözüküyor. Ayrıca sonuçların daha stabil olması için ağırlıkların değer aralığını azalttığımızda mse değerimizdeki sapmalar görünür ölçüde azalıyor.(ağırlıkların ilk değerlerini 0.1 ile çarpmak gibi..)

|  |  |  |  |
| --- | --- | --- | --- |
| Deney 5 | 10 Epok | 50 Epok | 100 Epok |
| λ = 0.01 | 0,004576 | 0,003212 | 0,003016 |
| λ = 0.025 | 0,003325 | 0,004301 | 0,003054 |
| λ = 0.05 | 0,004193 | 0,003218 | 0,003141 |

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 10 | 10 | Yapıldı. Textten veriler çekildi. |
| 1.b | 5 | 5 | Yapıldı. İstenilen şekilde listelendi ve çıktı üretildi. |
| 1.c | 5 | 5 | Yapıldı. İstenilen şekilde çıktı üretildi. |
| 1.d | 15 | 10 | Yapıldı. Ancak en kısa mesafeye göre yapıldığı için başarı oranı yaklaşık 90% seviyesinde (DFS algoritması kullanılamadı.) |
| 1.e | 5 | 5 | Yapıldı. Rastgele şehirler listelendi. Ayrıca aynı şehirler tekrar gelmeyecek şekilde kodlandı. |
| 2.a | 10 | 10 | Yapıldı. Nöron sınıfı ve nesneler istenilen şekilde kodlandı. |
| 2.b | 15 | 15 | Yapıldı. Eğitim tüm verileri kapsayacak şekilde ağırlıkları güncelleyerek sağlandı. |
| 2.c | 5 | 5 | Yapıldı. Farklı 5 verinin olduğu liste kullanılarak tahmini sonuçlar sağlandı. |
| 2.d | 10 | 9 | Yapıldı. Tablolar istenilen öğrenme katsayısı ve devirler ile deneyler gerçekleştirildi. Genel yorum yapıldı. Deney hakkında en doğru yorum yapılmamış olabilir. |
| Rapor | 10 | 9 | Yapıldı. Bazı kısımlar çok özenli olmayabilir. |
| Öz değerlendirme Tablosu | 10 | 10 | Yapıldı. Dürüst olundu. :) |
| **Toplam** | **100** | **93** |  |

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

**Not: Raporu teslim edilmeyen projeler değerlendirmeye alınmayacaktır.**