

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2024–2025 FALL SEMESTER**

**PROJECT-1 REPORT**

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

**DELIVERY DATE**

17/11/2024

**PREPARED BY**

05220000277, Mert Kerem Dalkılınç

05220000315, Görkem Ege Akıncı

05220000329, Yiğit Çakar

İçindekiler

[1) KARAYOLLARI UZAKLIK HESAPLAMALARI 2](#_Toc180684140)

[1.a Uzaklık Matrisi Oluşturma ve 10 Şehir Çifti İçin Şehir Uzaklıkları 2](#_Toc180684141)

[1.a.1 Kodlar 2](#_Toc180684142)

[1.a.2 Ekran görüntüleri 2](#_Toc180684143)

[1.b Uzaklık Matrisinin Komşu İllere Göre Düzenlenmesi 2](#_Toc180684144)

[1.b.1 Kodlar 2](#_Toc180684145)

[1.b.2 Ekran görüntüleri 2](#_Toc180684146)

[1.c Komşu Olmayan Şehir Çiftleri Arasındaki En Kısa Yolların Dijkstra Algoritması ile Hesaplanması ve Fark Değerlerinin Hesaplanması 2](#_Toc180684147)

[1.c.1 Kodlar 2](#_Toc180684148)

[1.c.2 Ekran görüntüleri 2](#_Toc180684149)

[1.d İzmir’ in İlçelerinin Uzaklık Matrisinin Oluşturulması 2](#_Toc180684150)

[1.d.1 Kodlar 2](#_Toc180684151)

[1.d.2 Ekran görüntüleri 2](#_Toc180684152)

[1.e İzmir’in İlçeleri için Komşu Olmayan İlçe Çiftleri Arasındaki En Kısa Yolların Dijkstra Algoritması ile Hesaplanması ve Fark Değerlerinin Hesaplanması 2](#_Toc180684153)

[1.e.1 Kodlar 2](#_Toc180684154)

[1.e.2 Ekran görüntüleri 3](#_Toc180684155)

[2) IMAGE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS 4](#_Toc180684156)

[2.a Veri Seti Oluşturma 4](#_Toc180684157)

[2.a.1 Kaynak Kod 4](#_Toc180684158)

[2.a.2 Açıklama 4](#_Toc180684159)

[2.a.3 Ekran Görüntüleri 4](#_Toc180684160)

[2.b Neuron (Sinir Hücresi) Sınıfı 4](#_Toc180684161)

[2.b.1 Kaynak Kod 4](#_Toc180684162)

[2.b.2 Açıklama 4](#_Toc180684163)

[2.c Neural Network (Yapay Sinir Ağı) Sınıfı 4](#_Toc180684164)

[2.c.1 Kaynak Kod 4](#_Toc180684165)

[2.c2 Açıklama 4](#_Toc180684166)

[2.d Eğitim Metodunun Yazılması 4](#_Toc180684167)

[2.d.1 Kaynak Kod 4](#_Toc180684168)

[2.d.2 Açıklama 4](#_Toc180684169)

[2.e Eğitim 4](#_Toc180684170)

[2.e.1 Kaynak Kod 4](#_Toc180684171)

[2.e.2 Ekran Görüntüleri / Tablolar 4](#_Toc180684172)

[2.f Modelin Görmediği Bir Matristen Sayıyı Tahminleme 4](#_Toc180684173)

[2.d.1 Kaynak Kod 4](#_Toc180684174)

[2.f.2 Sonuçlar / Ekran Görüntüleri 5](#_Toc180684175)

[Öz değerlendirme Tablosu 5](#_Toc180684176)

# 1) KARAYOLLARI UZAKLIK HESAPLAMALARI

The platform: Visual Studio 2022

Version: 17.11.5

Programming Language Used: C#

## 1.a Uzaklık Matrisi Oluşturma ve 10 Şehir Çifti İçin Şehir Uzaklıkları

### 1.a.1 Kodlar

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

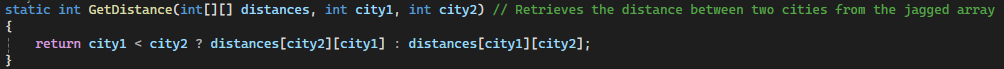
ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Jagged array that gives distances of the 81 cities in Türkiye.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.a.2 Ekran görüntüleri

ekran görüntüsü, siyah, siyah beyaz, monokrom, tek renkli içeren bir resim

Açıklama otomatik olarak oluşturuldu

## metin, yazı tipi, ekran görüntüsü içeren bir resim Açıklama otomatik olarak oluşturuldu

## 1.b Uzaklık Matrisinin Komşu İllere Göre Düzenlenmesi

### 1.b.1 Kodlar

### ekran görüntüsü, metin içeren bir resim Açıklama otomatik olarak oluşturuldu

### metin, ekran görüntüsü içeren bir resim Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.b.2 Ekran görüntüleri

## metin, ekran görüntüsü, siyah beyaz, tasarım içeren bir resim Açıklama otomatik olarak oluşturuldu

## metin, ekran görüntüsü, siyah beyaz içeren bir resim Açıklama otomatik olarak oluşturuldu

## 1.c Komşu Olmayan Şehir Çiftleri Arasındaki En Kısa Yolların Dijkstra Algoritması ile Hesaplanması ve Fark Değerlerinin Hesaplanması

### 1.c.1 Kodlar

### metin, ekran görüntüsü, yazılım içeren bir resim Açıklama otomatik olarak oluşturuldu

### metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.c.2 Ekran görüntüleri

## 

## metin, ekran görüntüsü, siyah beyaz, kalıp, desen, düzen içeren bir resim Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, kalıp, desen, düzen, siyah beyaz, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, kalıp, desen, düzen, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu



## 1.d İzmir’ in İlçelerinin Uzaklık Matrisinin Oluşturulması

### 1.d.1 Kodlar

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazılım, multimedya yazılımı, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 

### 1.d.2 Ekran görüntüleri

## ekran görüntüsü, metin, kalıp, desen, düzen, kumaş, doku içeren bir resim Açıklama otomatik olarak oluşturuldu 1.e İzmir’in İlçeleri için Komşu Olmayan İlçe Çiftleri Arasındaki En Kısa Yolların Dijkstra Algoritması ile Hesaplanması ve Fark Değerlerinin Hesaplanması

### 1.e.1 Kodlar

### metin, ekran görüntüsü içeren bir resim Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.e.2 Ekran görüntüleri

# metin, ekran görüntüsü, yazı tipi, menü içeren bir resim Açıklama otomatik olarak oluşturuldu

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

Açıklama otomatik olarak oluşturuldu

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

Açıklama otomatik olarak oluşturuldu

kalıp, desen, düzen, ekran görüntüsü, siyah beyaz, monokrom, tek renkli içeren bir resim

Açıklama otomatik olarak oluşturuldu

kalıp, desen, düzen, ekran görüntüsü, monokrom, tek renkli, kumaş, doku içeren bir resim

Açıklama otomatik olarak oluşturuldu

kalıp, desen, düzen, ekran görüntüsü, metin, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

Açıklama otomatik olarak oluşturuldu

# **2) IMAGE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS**

The Platform: Visual Studio, Version: 17.11.5, C# used

## **2.a Veri Seti Oluşturma**

### **2.a.1 Kaynak Kod**

// 1 Matrices

double[][] ornek\_matris1 = new double[][]

{

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 0, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 }

};

double[][][] matrices\_1 = new double[10][][];

for (int m = 0; m < matrices\_1.Length; m++) // Selecting all of the matrices

{

matrices\_1[m] = (double[][])ornek\_matris1.Clone();

for (int i = 0; i < ornek\_matris1.Length; i++)

{

matrices\_1[m][i] = (double[])ornek\_matris1[i].Clone(); // Copy every line

}

for (int i = 0; i < ornek\_matris1.Length; i++)

{

for (int j = 0; j < ornek\_matris1[0].Length; j++)

{

matrices\_1[m][i][j] = ornek\_matris1[i][j]; // Copy the data

if (i == 4 && (m == 9 || m == 8 || m == 6 || m == 5))

{

matrices\_1[m][i][m - 5] = 0;

}

else if (i == 4 && (m == 4 || m == 3))

{

matrices\_1[m][i][m] = 0;

matrices\_1[m][i][4 - m] = 0;

}

else if (i == 4 && (m == 2 || m == 1))

{

matrices\_1[m][i][5 - m] = 0;

matrices\_1[m][i][4 - m] = 0;

matrices\_1[m][i][3 - m] = 0;

matrices\_1[m][i][2 - m] = 0;

}

else if (i == 4 && m == 0)

{

matrices\_1[m][i][m] = 0;

matrices\_1[m][i][m + 1] = 0;

matrices\_1[m][i][m + 2] = 0;

matrices\_1[m][i][m + 3] = 0;

matrices\_1[m][i][m + 4] = 0;

}

}

}

}

// 2 Matrices

double[][] ornek\_matris2 = new double[][]

{

new double[] { 1, 0, 0, 0, 1 },

new double[] { 0, 1, 1, 1, 0 },

new double[] { 1, 1, 1, 0, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 0, 0, 0, 0, 0 }

};

double[][][] matrices\_2 = new double[10][][];

for (int m = 0; m < matrices\_2.Length; m++) // Selecting all of the matrices

{

matrices\_2[m] = (double[][])ornek\_matris2.Clone();

for (int i = 0; i < ornek\_matris2.Length; i++)

{

matrices\_2[m][i] = (double[])ornek\_matris2[i].Clone(); // Copy every line

}

for (int i = 0; i < ornek\_matris2.Length; i++)

{

for (int j = 0; j < ornek\_matris2[0].Length; j++)

{

matrices\_2[m][i][j] = ornek\_matris2[i][j]; // Copy the data

if (i == 0 && (m == 0 || m == 4))

{

matrices\_2[m][i][m] = 0;

}

else if (i == 1 && (m == 1 || m == 3))

{

matrices\_2[m][i][m] = 0;

}

else if (i == 2 && (m == 5 || m == 7))

{

matrices\_2[m][i][m - 3] = 0;

}

else if (i == 3 && (m == 6 || m == 8))

{

matrices\_2[m][i][m - 5] = 0;

}

else if (i == 4 && m == 9)

{

matrices\_2[m][i][m - 9] = 1;

matrices\_2[m][i][m - 5] = 1;

}

}

}

### }

// inputs matrice to store 1 matrices (10 matrices) and 2 matrices (10 matrices) (total 20 matrices)

double[][][] inputs = new double[20][][];

// Displaying all of the 20 matrices

nn.DisplayMatrices(inputs);

### **2.a.2 Açıklama**

At this stage, this code piece creates 10 1 matrices and 10 2 matrices. These 1 and 2 matrices are created from base 1 and 2 matrices with for loops. Base matrices are 2D array and other 20 2D matrices created with for loops are stored in the 2 separated (one for 1 matrices, one for 2 matrices) 3D arrays. Finally, code displays these 20 matrices from the method of neural network class (this class will be discussed later) (nn is an object of this class).

### **2.a.3 Ekran Görüntüleri**

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

Açıklama otomatik olarak oluşturulduekran görüntüsü, metin, siyah, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

a.3. First 5 of the “2 matrices”

a.4. Second 5 of the “2 matrices”

a.2. Second 5 of the “1 matrices”

a.1. First 5 of the “1 matrices”

## 

## **2.b Neuron (Sinir Hücresi) Sınıfı**

### **2.b.1 Kaynak Kod**

public class Neuron

{

private double[][] inputs;

private double[] weights;

private static Random rand = new Random();

// constructor

public Neuron(int numInputs)

{

inputs = new double[numInputs][];

weights = new double[numInputs];

// Random weights between 0 and 1

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

{

weights[i] = rand.NextDouble();

}

}

// Computing outputs using inputs and weights related to that input

public double ComputeOutput(double[][] inputs)

{

this.inputs = inputs;

double weightedSum = 0;

// Selecting all matrices and their variables to calculate weight

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

{

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

{

weightedSum += this.inputs[i][j] \* weights[i];

}

}

return weightedSum;

}

// Weights getter method

public double[] GetWeights()

{

return this.weights;

}

// Weights setter method

public void SetWeights(double[] newWeights)

{

if (newWeights.Length == weights.Length)

{

this.weights = newWeights;

}

else

{

Console.WriteLine("Error: Weights array size mismatch.");

}

}

// Pulling the weights to chech the calcualtions' accuracy

public void DisplayWeights()

{

Console.WriteLine("Weights: ");

foreach (var weight in weights)

{

Console.WriteLine(weight);

}

}

}

### **2.b.2 Açıklama**

In order to create neural networks, we need neurons. This led us to create neuron class. A primitive structure used (This means neuron will have input and weight that corresponding to it). This class have two objects inputs (2D array) and weights (1D array). Weights will be given randomly with “Random Library”. Neuron object computes the output with the “ComputeOutput” method. In order to that, there are some additional weight methods to detect any issues if there are any.

## **2.c Neural Network (Yapay Sinir Ağı) Sınıfı**

### **2.c.1 Kaynak Kod**

public class NeuralNetwork

{

private Neuron n1;

private Neuron n2;

private double learningRate;

private int[] output = new int[20];

// Constructor: Creating neurons in the network

public NeuralNetwork(int numInputs, double learningRate)

{

this.learningRate = learningRate;

n1 = new Neuron(numInputs); // create the n1 neuron

n2 = new Neuron(numInputs); // create the n2 neuron

}

// Training the neurons to get more suitable weigthts

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

{

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

{

Console.WriteLine($"Epoch {epoch + 1}/{epochs}"); // Display the eopch's number

for (int m = 0; m < inputs.Length; m++) // Loop on every input

{

for (int i = 0; i < inputs[m].GetLength(0); i++)

{

double[][] input = inputs[m];

int target = targets[m]; // Pull the target

// Calculating the outputs with the realted input

double n1Output = n1.ComputeOutput(input);

double n2Output = n2.ComputeOutput(input);

// Determining the neuron that has the bigger output

int maxOutputNeuron = (n1Output > n2Output) ? 1 : 2;

// Comparing the bigger output with the target

if ((target == 1 && maxOutputNeuron != 1) || (target == 2 && maxOutputNeuron != 2))

{

// If they are not same update neurons

if (maxOutputNeuron == 1)

{

// n1's output is wrong and n2's true

UpdateWeights(n1, -1, input); // decrease the n1's weights

UpdateWeights(n2, 1, input); // increase the n2's weights

}

else

{

// n2's output is wrong and n1's true

UpdateWeights(n2, -1, input); // decrease the n2's weights

UpdateWeights(n1, 1, input); // increase the n1's weights

}

}

}

}

}

}

// Weight update function

private void UpdateWeights(Neuron neuron, int direction, double[][] inputs)

{

double[] weights = neuron.GetWeights(); // Get the weights of the neuron

for (int m = 0; m < inputs.Length; m++) // Loop on every input

{

for (int i = 0; i < inputs[m].GetLength(0); i++)

{

weights[m] += direction \* learningRate \* inputs[m][i]; // Update the weights

}

}

neuron.SetWeights(weights); // Set the updated weights

}

// Displaying weights in the neurons

public void DisplayWeights()

{

Console.WriteLine("N1 Weights:");

n1.DisplayWeights(); // Show the n1's weights

Console.WriteLine("N2 Weights:");

n2.DisplayWeights(); // Show the n2's weights

}

// Test method, it is similar to train but not updating weights at this method

public void Test(double[][][] inputs)

{

Console.WriteLine("\nTest Results:");

for (int a = 0; a < inputs.Length; a++) // Loop on all of the matrices

{

double[][] input = inputs[a];

// Calculating outputs with the inputs

double n1Output = n1.ComputeOutput(input);

double n2Output = n2.ComputeOutput(input);

// Print the outputs

Console.WriteLine("Input:");

for (int i = 0; i < input.Length; i++) // Loop on all of the matrices

{

for (int b = 0; b < input[i].Length; b++)

{

Console.Write($"{input[i][b],1} ");

}

Console.WriteLine();

}

Console.WriteLine($"=> N1 Output: {n1Output:F2}, N2 Output: {n2Output:F2}");

Console.WriteLine();

// Decide the output

if (n1Output > n2Output)

{

Console.WriteLine("Çıktı: 1 (N1 is greater)\n");

output[a] = 1;

}

else

{

Console.WriteLine("Çıktı: 2 (N2 is greater)\n");

output[a] = 2;

}

}

}

public void DisplayMatrices(double[][][] inputs)

{

Console.WriteLine("\nMatrices: ");

for (int a = 0; a < inputs.Length; a++) // Loop on all of the matrices

{

double[][] input = inputs[a];

// Print the outputs

Console.WriteLine($"Input{a+1}:");

for (int i = 0; i < input.Length; i++) // Loop on all of the matrices

{

for (int b = 0; b < input[i].Length; b++)

{

Console.Write($"{input[i][b],1} ");

}

Console.WriteLine();

}

}

}

// Table to check accuracy of inputs to targets

public void TargetOutputTable(double[][][] inputs, int[] targets)

{

double[][] table = new double[targets.Length][];

Console.WriteLine("\nTarget - Output and Accuracy table:");

int trueOutputs = 0;

for (int a = 0; a < inputs.Length; a++) // Loop on all of the matrices

{

double[][] input = inputs[a];

// Calculating outputs with the inputs

double n1Output = n1.ComputeOutput(input);

double n2Output = n2.ComputeOutput(input);

// Determining if the output is correct to find accuracy

if (output[a] == targets[a])

{

trueOutputs += 1;

}

//adding output-target pair to the table

table[a] = new double[2];

table[a][0] = targets[a];

table[a][1] = output[a];

}

// Printing the table

Console.WriteLine("\nTarget - Output Table:");

Console.WriteLine("{");

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

{

Console.Write($" {{ {table[i][0]}, {table[i][1]} }}");

if (i < table.Length - 1)

Console.Write(", ");

Console.Write("");

if ((i + 1) % 5 == 0)

{

Console.WriteLine();

}

}

Console.WriteLine("}");

// Printing the accuracy

Console.WriteLine($"Accuracy: {trueOutputs}/{targets.Length} = %{trueOutputs / (double)targets.Length \* 100}");

}

### }

### **2.c2 Açıklama**

In this class we suppose to create 2 neurons that is Neuron type. One of the neurons will be trained to recognize 1 matrices and the other will be trained to recognize 2 matrices, then we will get the output 1 or 2. These neurons will scan the matrices one by one. At every matrix they will scan 25 data. There is also a learningRate (double type) object created to affect train part (It will be discussed later). Finally, there are some methods that will be discussed later.

## **2.d Eğitim Metodunun Yazılması**

### **2.d.1 Kaynak Kod**

“NeuralNetwork()” class provied in the “**2.c.1 Kaynak Kod”** is used for this article.

### **2.d.2 Açıklama**

In the NeuralNetwork class, Train method is written. In this method 2 neurons created in this class will be trained to recognize 1 and 2’s more accurately. This method takes the 20 1 and 2 matrices as training inputs, also takes their targets, and epochs number (Epochs stands for how many times this training will happen.). Every epoch, neurons will compute an output, and these outputs will be compared with target (Bigger output will be the final output). If the target and output are not same, then corresponding weights of these 2 neurons will be changed with the formula. In this formula inputs, learningRate are used to increase or decrease the neurons weight.

## **2.e Eğitim**

### **2.e.1 Kaynak Kod**

Also “NeuralNetwork()” class provied in the “**2.c.1 Kaynak Kod”** is used for this article. After this class there is Main() class in the following lines:

static void Main(string[] args)

{

// 1 Matrices

double[][] ornek\_matris1 = new double[][]

{

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 0, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 }

};

double[][][] matrices\_1 = new double[10][][];

for (int m = 0; m < matrices\_1.Length; m++) // Selecting all of the matrices

{

matrices\_1[m] = (double[][])ornek\_matris1.Clone();

for (int i = 0; i < ornek\_matris1.Length; i++)

{

matrices\_1[m][i] = (double[])ornek\_matris1[i].Clone(); // Copy every line

}

for (int i = 0; i < ornek\_matris1.Length; i++)

{

for (int j = 0; j < ornek\_matris1[0].Length; j++)

{

matrices\_1[m][i][j] = ornek\_matris1[i][j]; // Copy the data

if (i == 4 && (m == 9 || m == 8 || m == 6 || m == 5))

{

matrices\_1[m][i][m - 5] = 0;

}

else if (i == 4 && (m == 4 || m == 3))

{

matrices\_1[m][i][m] = 0;

matrices\_1[m][i][4 - m] = 0;

}

else if (i == 4 && (m == 2 || m == 1))

{

matrices\_1[m][i][5 - m] = 0;

matrices\_1[m][i][4 - m] = 0;

matrices\_1[m][i][3 - m] = 0;

matrices\_1[m][i][2 - m] = 0;

}

else if (i == 4 && m == 0)

{

matrices\_1[m][i][m] = 0;

matrices\_1[m][i][m + 1] = 0;

matrices\_1[m][i][m + 2] = 0;

matrices\_1[m][i][m + 3] = 0;

matrices\_1[m][i][m + 4] = 0;

}

}

}

}

// 2 Matrices

double[][] ornek\_matris2 = new double[][]

{

new double[] { 1, 0, 0, 0, 1 },

new double[] { 0, 1, 1, 1, 0 },

new double[] { 1, 1, 1, 0, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 0, 0, 0, 0, 0 }

};

double[][][] matrices\_2 = new double[10][][];

for (int m = 0; m < matrices\_2.Length; m++) // Selecting all of the matrices

{

matrices\_2[m] = (double[][])ornek\_matris2.Clone();

for (int i = 0; i < ornek\_matris2.Length; i++)

{

matrices\_2[m][i] = (double[])ornek\_matris2[i].Clone(); // Copy every line

}

for (int i = 0; i < ornek\_matris2.Length; i++)

{

for (int j = 0; j < ornek\_matris2[0].Length; j++)

{

matrices\_2[m][i][j] = ornek\_matris2[i][j]; // Copy the data

if (i == 0 && (m == 0 || m == 4))

{

matrices\_2[m][i][m] = 0;

}

else if (i == 1 && (m == 1 || m == 3))

{

matrices\_2[m][i][m] = 0;

}

else if (i == 2 && (m == 5 || m == 7))

{

matrices\_2[m][i][m - 3] = 0;

}

else if (i == 3 && (m == 6 || m == 8))

{

matrices\_2[m][i][m - 5] = 0;

}

else if (i == 4 && m == 9)

{

matrices\_2[m][i][m - 9] = 1;

matrices\_2[m][i][m - 5] = 1;

}

}

}

}

// Independent examples

double[][] example3\_1 = new double[][]

{

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 }

};

double[][] example3\_2 = new double[][]

{

new double[] { 1, 1, 0, 1, 1 },

new double[] { 0, 0, 0, 0, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 0, 0, 0, 0 }

};

double[][] example3\_3 = new double[][]

{

new double[] { 1, 0, 1, 0, 1 },

new double[] { 0, 1, 0, 1, 0 },

new double[] { 1, 1, 1, 0, 1 },

new double[] { 1, 1, 0, 1, 1 },

new double[] { 1, 0, 0, 0, 1 }

};

// A single neuron example

// Creating a neuron with 4 examples

Neuron neuron = new Neuron(4);

// 4 input example

double[][] neuron\_inputs = new double[][]

{

new double[] {0.9},

new double[] {0.2},

new double[] {0.5},

new double[] {0.7}

};

// Calculating outputs

double output = neuron.ComputeOutput(neuron\_inputs);

// Writing output in the console

Console.WriteLine("Output: " + output);

// Displaying weights used to calculate these 4 inputs

neuron.DisplayWeights();

// inputs matrice to store 1 matrices (10 matrices) and 2 matrices (10 matrices) (total 20 matrices)

double[][][] inputs = new double[20][][];

for (int m = 0; m < (inputs.Length) / 2; m++)

{

inputs[m] = matrices\_1[m];

}

for (int m = 0; m < (inputs.Length) / 2; m++)

{

inputs[m + 10] = matrices\_2[m];

}

// targets of that 1 and 2 matrices

int[] targets =

{

1, 1, 1, 1, 1,

1, 1, 1, 1, 1,

2, 2, 2, 2, 2,

2, 2, 2, 2, 2,

};

// inputs matrice to store 3 independent matrice

double[][][] testDifferentInputs = new double[3][][]

{

example3\_1, example3\_2, example3\_3

};

// Neural Network that gets 20 matrices to train and has 0.03 λ value

NeuralNetwork nn = new NeuralNetwork(inputs.Length, 0.000005);

// Displaying all of the 20 matrices

nn.DisplayMatrices(inputs);

// Training the network with these 20 matrices

nn.Train(inputs, targets, 40);

// Testing the network with the same inputs

nn.Test(inputs);

// Writing the output-target and accuracy rate

nn.TargetOutputTable(inputs, targets);

// Testin with 3 different matrices

nn.Test(testDifferentInputs);

Console.ReadKey();

}

### **2.e.2 Ekran Görüntüleri / Tablolar**

siyah beyaz, ekran görüntüsü, monokrom, tek renkli, kalıp, desen, düzen içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, menü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

c.1. Outputs of the test (Matrix and neuron outputs).

b. Epochs that represents every successful training.

metin, ekran görüntüsü, menü, siyah beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

c.2. Outputs of the test (Matrix and neuron outputs).

c.3. Outputs of the test (Matrix and neuron outputs).

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

Açıklama otomatik olarak oluşturuldu

c.4. Outputs of the test (Matrix and neuron outputs).

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

Açıklama otomatik olarak oluşturuldu

d. Table of outputs and related targets with the accuracy

## 

## **2.f Modelin Görmediği Bir Matristen Sayıyı Tahminleme**

### **2.d.1 Kaynak Kod**

“NeuralNetwork()” class provied in the “**2.c.1 Kaynak Kod”** and “Main()” class provided in the “**2.e.1 Kaynak Kod”** are used for this article.

### **2.f.2 Sonuçlar / Ekran Görüntüleri**

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

Açıklama otomatik olarak oluşturuldu

e. Independent inputs and their outputs

# 

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a) | 5 | 5 | It is done. We created a jagged array that contains 81 cities and their distances between them. Then, for 10 random city pairs we printed the distances. |
| 1.b | 7 | 7 | It is done. We created a dictionary that contains all the cities and their neighbors. Then made all connection values on the distance matrix infinite except the connections between neighboring cities. |
| 1.c | 10 | 10 | It is done. For each pair of cities, we printed the direct distance, Dijkstra distance and their differences. |
| 1.d | 8 | 8 | It is done. We created a 30x30 matrix for districts of İzmir. |
| 1.e | 5 | 5 | It is done. We made the operations in 1.c for districts of Izmir. |
| 2.a | 5 | 5 | It is done. 20 matrices created successfully and stored in an array. Also shown in the console. |
| 2.b | 5 | 5 | It is done. Neuron class and its objects are created. It has the appropriate methods. Tested successfully. |
| 2.c | 10 | 10 | It is done. NeuralNetwork class and its objects are created. It has the appropriate methods. Tested successfully. |
| 2.d | 5 | 5 | It is done. Train method in NeuralNetwork class is created. Tests showed that the function does its job correctly. |
| 2.e | 10 | 10 | It is done. Training with the 20 matrices is done and the test is done. In the console outputs and their table with accuracy is shown. |
| 2.f | 5 | 5 | It is done. Testing 3 independent matrices after the training is successfully shown. |
| Rapor | 15 | 15 | Related information added correctly. |
| Öz değerlendirme Tablosu | 10 | 10 | Points and their reasons are explained clearly. |
| **Toplam** | **100** | **100** | **Every step done properly** |