به نام اندیشه پاک

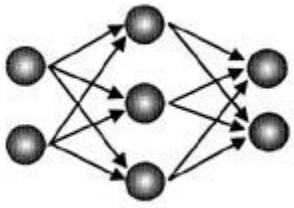
گزارش پروژه درس شبکه عصبی

ایجاد یک برنامه تشخیص کاراکترهای نوری

استاد محترم:

اعضای گروه:

در این برنامه از کتابخانه *Neuro.net* استفاده شده تا یک برنامه ساده *OCR* به کمک شبکه عصبی *MLP* بنویسیم.



همچنین انتظار می‌رود با اصول پردازش تصویر نیز آشنایی داشته باشیم. از کلاس کتابخانه‌ای `BackPropagationRPROPNetwork` استفاده می‌کنیم تا شبکه *OCR* خود را بسازیم.

```
//Inherit form Backpropagation neural network
public class OCRNetwork: BackPropagationRPROPNetwork
{
    //Override method of the base class in order to implement our
    //own training method
    public override void Train(PatternsCollection patterns)
    {
        ...
    }
}
```

متد `Train` از کلاس پایه را *Override* می‌کنیم تا متد آموزش شبکه مورد نظر خود را پیاده‌سازی کنیم و همچنین کیفیت برنامه نیز به سرعت آموزش شبکه وابسته می‌باشد. شرایط توقف آموزش هم به این صورت در در نظر می‌گیریم که شبکه تا زمانی که توانایی تشخیص صحیح همه‌ی الگوها را بدون حتی یک خطا نداشته باشد، متوقف نشود.

```
public override void Train(PatternsCollection patterns)
{   //Current iteration number
    if (patterns != null)
    {
        double error = 0;
        int good = 0;
        // Train until all patterns are correct
        while (good < patterns.Count)
         {
            good = 0;
            for (int i = 0; i<patterns.Count; i++)
            {
                //Set the input values of the network
                for (int k = 0; k<NodesInLayer(0); k++)
                    nodes[k].Value = patterns[i].Input[k];
                //Run the network
                this.Run();
                //Set the expected result
                for (int k = 0;k< this.OutputNodesCount;k++)
                    this.OutputNode(k).Error = patterns[i].Output[k];
                //Make the network to remember corresponding output
                //values. (Teach the network)
                this.Learn();
                //See if network did produced correct result during
                //this iteration
                if (BestNodeIndex == OutputPatternIndex(patterns[i]))
                                good++;
             }
            //Adjust weights of the links in the network to their
            //average value. (An epoch training technique)
            foreach (NeuroLink link in links)
                ((EpochBackPropagationLink)link).Epoch(patterns.Count);
```

```
        }
    }
}
```

همچنین یک *Property* به‌نام `BestNodeIndex` را پیاده‌سازی کردیم که ایندکس گره‌ای که بیشترین ارزش و کمترین خطا را دارد برمی‌گرداند. متد `OutputPatternIndex` ایندکس المان الگوی خروجی که مقدار ۱ دارد را برمی‌گرداند. اگر این مقادیر برابر باشند شبکه نتیجه صحیحی را تولید کرده است.

```
public int BestNodeIndex
{
    get {
        int result = -1;
        double aMaxNodeValue = 0;
        double aMinError = double.PositiveInfinity;
        for (int i = 0; i< this.OutputNodesCount;i++)
        {
            NeuroNode node = OutputNode(i);
            //Look for a node with maximum value or lesser error
            if ((node.Value > aMaxNodeValue)||
                  ((node.Value >= aMaxNodeValue)&&(node.Error <aMinError)))
            {
                aMaxNodeValue = node.Value;
                aMinError = node.Error;
                result = i;
            }
        }
        return result;
     }
}
```

تابع سازنده شبکه تعداد نرون‌ها در هر مرحله را مشخص می‌کند. لایه اول لایه ورودی می‌باشد . برابر با تعداد المان‌ها در ماتریس تصویر دیجیتال شده می‌باشد که در ادامه بحث خواهد شد.

```
NodesNumber = (InputsCount+OutputsCount) / 2
```

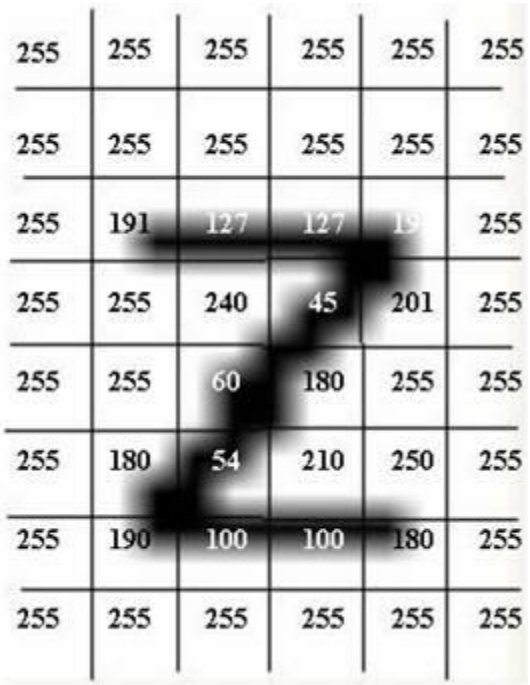
لایه آخر، لایه خروجی می‌باشد و در این لایه به‌دنبال خروجی‌ها می‌گردیم. در اینجا تعداد *Node* های لایه خروجی را برابر با تعداد حروف و کارکترهایی در نظر می‌گیریم که می‌خواهیم تشخیص دهیم.

```
//Create an instance of the network
backpropNetwork = new OCRNetwork(new int[3] {aMatrixDim * aMatrixDim,
       (aMatrixDim * aMatrixDim + aCharsCount)/2, aCharsCount});
```

هر الگوی آموزش برای تشخیص تصاویر از ۲ آرایه یک بعدی از جنس *Float* برای ورودی و خروجی تشکیل شده.

```
/// <summary>
/// A class representing single training pattern and is used to train a
/// neural network. Contains input data and expected results arrays.
/// </summary>
public class Pattern: NeuroObject
{
    private double[] inputs, outputs;
    ...
}
```

آرایه `Inputs` داده‌های ورودی را در بر می‌گیرد که در این مورد نمایش دیجیتال شده تصویر کارکتر می‌باشد. منظور از دیجیتال شده همان فرآیند مشخص کردن روشنایی می‌باشد. برای این منظور تصویر را به خانه‌هایی تقسیم کرده و مقادیر را در آن‌ها قرار می‌دهیم.



255	255	255	255	255	255
255	255	255	255	255	255
255	191	127	127	19	255
255	255	240	45	201	255
255	255	60	180	255	255
255	180	54	210	250	255
255	190	100	100	180	255
255	255	255	255	255	255

از متد `CharToDoubleArray` برای این کار استفاده می‌کنیم. سپس مقادیر تولید شده را نرمال می‌کنیم تا مقادیر بین ۱ و ۱- قرار بگیرند و برای این کار نیز از متد `Scale` استفاده می‌کنیم.

```
//aSrc – an image of the character
//aArrayDim – dimension of the pattern matrix
//calculate image quotation X step
double xStep = (double)aSrc.Width/(double)aArrayDim;
//calculate image quotation Y step
double yStep = (double)aSrc.Height/(double)aArrayDim;
double[] result = new double[aMatrixDim*aMatrixDim ];
for (int i=0; i<aSrc.Width; i++)
    for (int j=0;j<aSrc.Height;j++)
    {
        //calculate matrix address
        int x = (int)(i/xStep);
        int y = (int)(j/yStep);
        //Get the color of the pixel
        Color c = aSrc.GetPixel(i,j);
        //Absolute value of the color, but I guess, it is possible to
        //use the B component of Alpha color space too...
        result[y*x+y]+=Math.Sqrt(c.R*c.R+c.B*c.B+c.G*c.G);
    }
//Scale the matrix to fit values into a range from 0..1 (required by
//ANN) In this method we look for a maximum value of the element
//and then divide all elements of the matrix by this maximum value.
return Scale(result);
```

آرایه خروجی از الگوها، نتایج مورد انتظار را نمایش می‌دهد. نتایجی که شبکه در طول آموزش استفاده می‌کند. در آنجا المان‌های زیادی در آرایه می‌باشد که همان کارکترهایی می‌باشند که می‌خواهیم تشخیص دهیم. سپس شبکه از روی ورودی و مجموعه عکس‌های دیجیتال شده، خروجی را تشخیص خواهد داد که ورودی شبیه کدامیک از حروف می‌باشد. متد `CreateTrainingPatterns` این کار را برای ما انجام می‌دهد.

```
public PatternsCollection CreateTrainingPatterns(Font font) {
//Create pattern collection
// As many inputs (examples) as many elements in digitized image matrix
// As many outputs as many characters we going to recognize.
```

```
PatternsCollection result = new PatternsCollection(aCharsCount,
                              aMatrixDim * aMatrixDim, aCharsCount);
// generate one pattern for each character
for (int i= 0; i<aCharsCount; i++)
{
      //CharToDoubleArray creates an image of the character and digitizes it.
      //You can change this method to pass actual the image of the character
      double[] aBitMatrix = CharToDoubleArray(Convert.ToChar(aFirstChar + i),
                                                    font, aMatrixDim, 0);
      //Assign matrix value as input to the pattern
       for (int j = 0; j<aMatrixDim * aMatrixDim; j++)
            result[i].Input[j] = aBitMatrix[j];
      //Output value set to 1 for corresponding character.
      //Rest of the outputs are set to 0 by default.
           result[i].Output[i] = 1;
     }
       return result;
}
```

حال که الگوها را ایجاد کردیم، از آن‌ها برای آموزش شبکه استفاده می‌کنیم.

برای آموزش شبکه باید متد زیر را فراخوانی کرد و الگوهای ایجاد شده را به عنوان ورودی به آن داد.

```
//Train the network
backpropNetwork.Train(trainingPatterns);
```

همچنین می‌توان نتایج این متد را برای استفاده‌های بعدی ذخیره نمود.

حال می‌توانیم نتیجه یادگیری شبکه را مشاهده کنیم. تکه کد زیر نشان می‌دهد چطور شبکه آموزش دیده می‌تواند در برنامه *OCR* ما استفاده شود

```
//Get your input data
 double[] aInput = ... (your digitized image of the character)
//Load the data into the network
for (int i = 0; i< backpropNetwork.InputNodesCount;i++)
    backpropNetwork.InputNode(i).Value = aInput[i];
//Run the network
backpropNetwork.Run();
//Get result from the network and convert it to a character
return Convert.ToChar(aFirstChar + backpropNetwork.BestNodeIndex).ToString();
```

به منظور استفاده از شبکه ما باید داده‌ها را در لایه ورودی بارگذاری کنیم. سپس از تابع `Run` استفاده کنیم تا اجازه دهیم شبکه داده ها را پردازش کند. در نهایت می‌توان نتایج را از *Node*های خروجی گرفت و تحلیل نمود(توسط `BestNodeIndex` در کلاس `OCRNetwork`).