Margarita Geleta, Ibrar Malik

Mathematical Optimization | Data Science

Pattern Recognition

WITH SINGLE LAYER NEURAL NETWORK

round()

**Figure 2:** Architecture of a perceptron (SLNN) with a sigmoidal activation function.

**About the project**

The aim of the project is to build a *Single Layer Neural Network* (abbreviated as *SLNN*) from zero in *Python* that would be capable of recognizing a set of target numbers. The numbers from the [0,9] interval will be used as targets. Each number has a 7*x*5 pixel matrix representation, with values 0 and 1 (*figure 1*).

|  |  |  |
| --- | --- | --- |
| /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/135484DD.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/A1CFCD3A.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/DFC87F3.tmp |
| /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/9AB8B958.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/8ED27979.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/F9E76126.tmp |
| /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/5E2B12EF.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/CB123024.tmp | /var/folders/sc/wk95ry317zj7gtv9yxmtgxl00000gp/T/com.microsoft.Word/Content.MSO/429EC9D5.tmp |

**Figure 1:** Pixel matrix representation of [0,9] numbers.

**Network architecture**

Our neural network is a *perceptron*, that is, a single neuron model, in our case, with 35 entries – all the pixels of a number stored in the input vector . Then, combined with the vector of weights and finally, the *sigmoid function* is applied to obtain a binary output 0/1.

So, our neural network () is defined as:

0/1

|  |
| --- |
| *x1* |
| *x2* |
|  |
| *xd* |

We define in *Python* as follows:

|  |
| --- |
| **def** **sigmoid**(x):  **return** 1. / (1. + np.exp(-x))  **def** **y**(X, w):  **return** **sigmoid**((**sigmoid**(X) @  w.T)) |

**Network training**

Now the big question is: how do we get our neural network to learn? It is going to learn from errors. Thus, we need to define an “*error/loss function*”. Before formulating that function, note that we are going to use a set of numbers to train our network and a separate test for testing the predictions. These sets are called *training ()* and *test* () sets, respectively.

The loss function is defined as follows:

being the size of the training set. We can also add a regularization parameter (of type *L2*):

We define them in *Python* in one method, being p =. If p =, then we return , otherwise with its corresponding regularization parameter :

|  |
| --- |
| **def** **loss**(w, X, ytr, p=0):  **return** np.linalg.norm(y(X, w) –  ytr)\*\*2 + p/2 \*  np.linalg.norm(w)\*\*2 |

To train the network, i.e. find the optimum , we need to minimize with some optimization algorithm. To minimize the loss function we need its gradient: