052300 VU Foundations of Data Analysis
Assignment 4: SVM and Neural Networks
Appendix

November 26, 2017

Question 3 (Neural Network Training): The Neural Network formulation for assignment 4 has caused some confusions. There were a few typos that are highlighted here in red. We also use some more detailed equations that might be helpful to follow the flow of error backpropagation. It is very similar to the variables used in the textbook and the course slides. The variable names in the code are slightly different, which is part of the assignment to solve. To get more details, you can read chapter 5.3 of Bishop book.

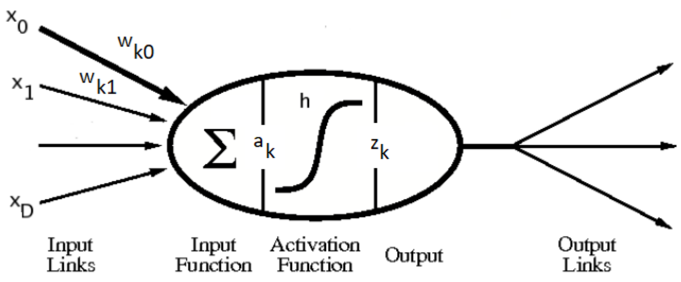

Figure 1: A single neuron in our neural network. It shows activation variable a_k and output z_k for node k. Weights w_{kj} is the weight for the node k from the node j in previous level. We also used $w_{j \to k}$ for this in the assignment description.

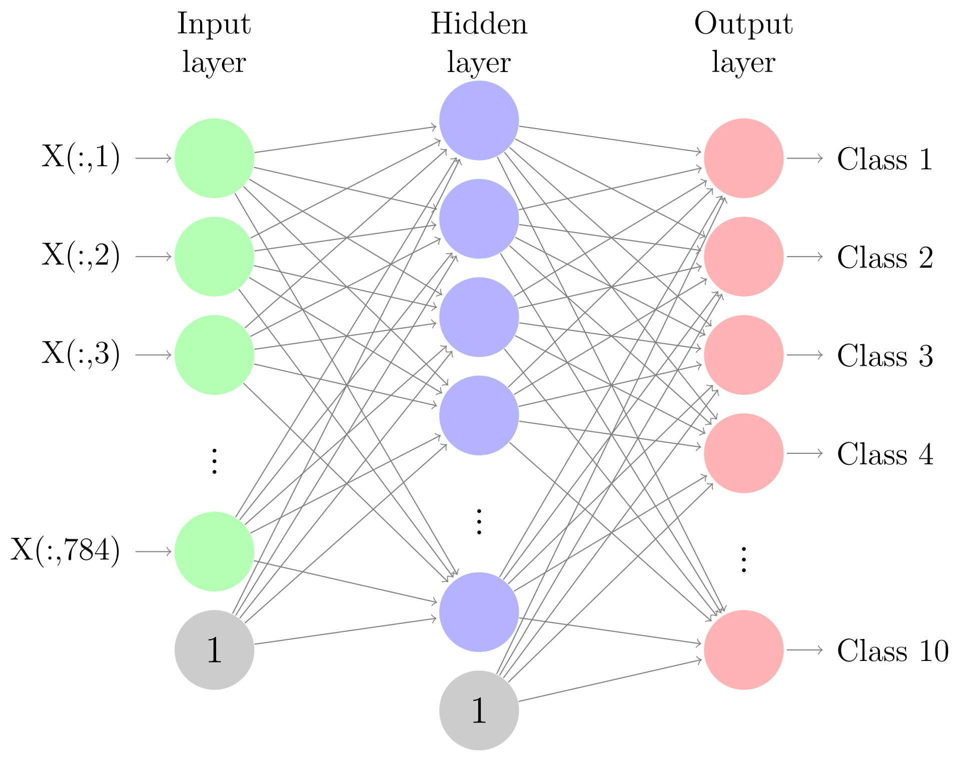

Figure 2: The Neural Network model in Question 3. There are ($D = 784$) input nodes, ($H = 500$) hidden nodes, and ($K = 10$) output nodes. The weights between i^{th} input node and j^{th} hidden node is `weights[[1]][j,i]`, and between j^{th} hidden node and k^{th} output node is `weights[[2]][k,j]`. The activation value for n^{th} node at m^{th} layer is `a[[m]][n]`, i.e. `a[[1]]` is the input layer and `a[[3]]` is the output layer.

The structure of each neurons is shown in Figure 1. Figure 2 shows the full structure of our neural network. We use super-index to show the level. For example, for the k^{th} node in the input layer, we use $x_t^{(0)}$, $a_i^{(1)}$, and $z_i^{(1)}$ for the input, activation, and the output values. We assume that $z_i^{(1)} = a_i^{(1)} = x_i^{(0)}$, i.e. the input layer is just assigning a node to each input variable. Since the network is connected, x from each level is connected to z from the previous level; e.g. $x_j^{(2)} = z_j^{(2)}$. So, using Equation 5.43 from the Bishop book:

$$\frac{\partial}{\partial w_{j\to k}^{(2)}} E_n \quad = \quad \frac{\partial}{\partial w_{kj}^{(2)}} E_n = \delta_k^{(2)} x_j^{(2)} = (y_k - t_k) x_j^{(2)} \tag{1}$$

$$= \quad (y_k - t_k) z_j^{(2)} \tag{2}$$

In other words, the partial derivative of the error with respect to weight $w_{kj}^{(2)}$, for input j of node k in the output layer is the delta between the predicted and the real output, times the j^{th} input of the node, which is equivalent to the output of the node j from the hidden layer. The same rule applies to the previous level:

$$\frac{\partial}{\partial w_{i \to j}^{(1)}} E_n = \delta_j^{(1)} x_i^{(1)} = \delta_j^{(1)} z_i^{(1)} \tag{3}$$

$$\delta_j^{(1)} = g'(a_j^{(1)}). \sum_k w_{kj}^{(2)} \delta_k^{(2)} \tag{4}$$

Note that the notations are a little different in the source code. Variable a is used to show the output values instead of the activation values. Also, $a[1]$ represents the input data and $a[3]$ represents the output of the network, as shown in Figure 2.