Programming a simple MLP

A. Mustafi

October 22, 2017

Abstract

In this sessional we shall write a simple program to simulate a MLP, which shall utilize the *Backpropogation* training algorithm. Our target is to reduce the classification error on a simple dataset. Since we only have a single layer of hidden nodes this simplifies the calculation. Instead of having to loop over the layers this allows to calculate just the required number of variables e.g. $Input_{hidden}$, $Output_{hidden}$ etc. The computations are also simplified by the fact that we are assuming there is only one output layer node.

1 Introduction

We shall simulate the most basic MLP having a single hidden layer and one which shall perform a binary classification.

2 Inputs

The inputs to your program are the following:

1. A dataset containing M numeric attributes, and one binary class attribute. You can use the *iris* dataset, where the target class is assumed to be one of the three classes present. Ensure that the data attributes are **normalized**.
2. The number of hidden layer nodes to use i.e. K

3 Methodology

THE FORWARD PROPOGATION STEP:

1. Add a bias attribute to the dataset.
2. Initialize a weight matrix for the weights from the input layer to the hidden layer. Ensure the matrix has a size $(\mathbf{M}+\mathbf{1})\mathbf{xK}$. *Justify*.

3. Initialize a weight matrix for the weights from the hidden layer to the output layer. Remembering that the hidden layer also has a bias unit the weight matrix should have a size $(\mathbf{K}+\mathbf{1})\mathbf{x1}$. *Justify*.

4. Calculate the input to any node in layer j, as follows:

$$z_j = \sum_{i=0}^{K+1} w_{ij} x_i \tag{1}$$

. Remember the summation includes the bias which has a constant value of 1. Also remember the inputs to all nodes can be calculated at once if we use a dot product method to calculate it.

5. Calculate the output of any node in layer j as follows:

$$y_i = sigmoid(z_j) \tag{2}$$

THE BACK PROPOGATION STEP:

1. Calculate the error at the output layer node as follows:

$$Err_j = O_j(1-O_j)(T_j - O_j) \tag{3}$$

where T_j is the actual value of the class label (i.e. either 1 or 0) and O_j is the computed value, from the forward propogation step.

2. The error at any other node (in this case only nodes of the hidden layer) can be calculated as

$$Err_j = O_j(1-O_j) \sum_{k \epsilon output} Err_k w_{jk} \tag{4}$$

Remember we have already calculated the Err_k in the previous step. O_j has its usual meaning but for the node under consideration.

3. The update of any weight w_{ij} is done as follows:

$$\Delta w_{ij} = \alpha Err_j O_i \tag{5}$$

where α is the learning rate.

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{6}$$

4 Output

Report the classification accuracy you are able to obtain over a train set/ test set.

5 Extra credit

Try and generalize the algorithm for any arbitrary number of hidden layers, containing different number of nodes. Also try to generalize the program so that the output layer has multiple nodes.