Implementation of Back Propagation

Language Used: Python 3.7

Reference TextBook: Machine Learning by Tom Mitchell

Feature of the Code

1. **Data Structures:**The dictionary and list data structures have been used to implement the Algorithm. The various functions and data structures used are stored as member methods and member variables of the **Network Class** in the code**.** The functionalities of the Neural Network are simulated as follows:

    a. Layers of the Network: A nested dictionary is used to store every layer of the Network. The key of the dictionary is the name of the layer i.e “Input” ,”hidden_1” ,”Output” etc. A list storing the layer names in the order of the feed forward network is maintained. This list is referenced during forward propagation and it’s reversed version is referenced during back propagation. The value of the layer dictionary is another dictionary with the following key, value pairs:

        i. Nodes : Corresponding to the Nodes key a list of node values are stored. They are initialised as zero and are later assigned values based on the inputs from the previous layer and the corresponding weights.

        ii. Weights : Each layer is assigned a Weight Matrix which is a 2D list of float values. These are initialised randomly. For every layer the weight matrix denotes the value of weights for the edges between the nodes of the previous layer and the current layer.
        For example : If the previous layer has 5 nodes and the current layer has 4 nodes. The weight matrix stored in the current layer denoting the weights of the edges between the two layers would be of dimension (4,5), where W[i][j] denotes the weight of the edge connecting the i^{th} Node of the current layer with the j^{th} node of the previous layer. If the node from the previous layer is not connected to the current layer node, the corresponding weight is stored as a string “ZERO”.

iii. Deltas: Denote the differential of output wrt weights, corresponding to every node. It is observed that output of the node is dependent on input from the previous layer. The activation function used at every Node is the sigmoid function. For each node the deltas are stored as a list

Deltas for the Output Layer Nodes are calculated as follows:

$$\delta_k = O_k (1 - O_k) (T_k - O_k)$$

where:
δ_k denotes delta value for the k^{th} Node
O_k denotes Output of the k^{th} Node in Output Layer
T_k denotes the Target value for k^{th} Node in Output Layer

Deltas for the Hidden Layer Nodes are calculated as follows:

$$\delta_h = O_h (1 - O_h) \Sigma_{k \in M} W_{kh} \delta_k$$

where:
M denotes the set of Nodes of the next layer connected to the current node.
δ_h denotes delta value at the hidden Node
O_h denotes the Output value at the hidden Node
δ_k denotes the delta value of the Node is the next layer connected to the current Node
W_{kh} denotes the the Weight of the edge connecting the current Node with the Nodes of the next layer

Note: Derivative of Sigmoid $\Delta y = y(1-y)$

Note: Deltas for the input layer will be zero.

b. Feed Forward: The Neural Network built is a feed forward network, to obtain the Output values at the Output layer the input is fed to the Input layer which is propagated forward through the layers in order. The value at each Node is obtained by finding a dot product of the inputs and corresponding weights along with the addition of bias. These values are then provided as Input to the next layers till we reach the Output layer using the list of layer names in order.

c. Activation Function: The activation function used at every Node is the sigmoid function.

d. Error Function: The error function used at the Output Layer is sum of squares error function as shown:

Error(for a training example) : $\frac{1}{2} \Sigma_k (T_k - O_k)^2$ for the k nodes of the Output Layer

e. Back Propagation: After obtaining the error, the delta values are calculated for Output Layer as shown above. The algorithm used to minimize the error is gradient descent. Since the Back Propagation and Weight Updation functions are called after the error is calculated for every training sample(rather than after summation of error for all training samples) , the approach used is known as **Stochastic Gradient Descent**. These delta values of the Output layer are used to calculate the delta values of the previous hidden layer. The process is repeated using the list of layer names in reverse order, till the input layer is reached.

f. Weight Updation: After back propagation the weights are updated as follows:

$$\Delta W_{ji} = \eta \, \delta_j \, X_i$$
$$W_{ji}(new) = W_{ji}(old) + \Delta W_{ji}$$

where:

ΔW_{ji}: denotes the change in weight

W_{ji} : denotes the Weight of the edge connecting Node j of previous layer to Node i of next layer.

η : Learning rate of the Network.

δ_j : Delta value for the j^{th} node in the next layer.

X_i : Input at the i^{th} Node of the current layer.

2. **Configurations of the Network:** The network can be configured by the user by stating the following properties of the Network:
    a. Structure: A list named structure contains the number of nodes in every layer starting from input till the output layer.

        Ex: structure = [5, 4, 3]
        It denotes input_layer has 5 nodes, hidden layer has 4 nodes, and output layer has 3 nodes.

    b. Connectivity: The network may not be a fully connected network. To configure the connectivity between nodes, the user can specify the 2D "links" array as shown wrt to the above mentioned structure of [5, 4, 3]:

        Ex links [["11011", " 11101", "11111", "11000"], ["1101", "1111", "1001"]]

        where
        links[i][j] is a binary string(where 0 denotes node is not connected and 1 denotes not is connected) , which resembles the links between the nodes of $(i-1)^{th}$ layer with the j^{th} Node of the current i^{th} layer.

        Note: 0^{th} layer is the Input Layer.

    c. Model Parameters. While creating an object of the Network class, the learning rate and the bias values are specified as parameters to the Constructor.

3. **Training:** For training the model the input consists of 2 parameters.

   - ➢ The input X is a 2D array of dimension (m,n) where m denotes number of training samples, and n denotes the input dimension as per the structure of Network. For the previous example n = 5

   - ➢ The input Y denotes is also a 2D array denoting the Y labels(target values) of (m,n) where m in the number of training examples, and Y is the output dimension based on the structure of Network. For the previous example n = 3

The number of epochs for training can be specified when calling the train Function

In the code the model is trained using the **iris dataset**.

Dataset : The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Thus the dataset in total contains 150 samples. The iris dataset was stored as a csv as shown in Table 1.1, the features along with the labels were extracted as a part of preprocessing before feeding the data to the model.

Sr No	sepal_length	sepal_width	petal_length	petal_width	Species
1	5.1	3.5	1.3	0.2	setosa
2	7.1	3	6.6	2.1	virginica
3	6.4	3.2	4.5	1.5	versicolor

Table 1.1

Input X is a list of 4 features
Y labels include 3 lists i.e [1,0,0], [0,1,0] [0,0,1] denoting the species “setosa”, “versicolor”, “virginica” respectively.

The final training accuracy as well as error is displayed after every epoch during training.

4. **Testing :** The trained model is tested by providing only the input X array. It outputs the Y array(predicted labels) which is compared with the actual Y array for the given input to obtain the accuracy of the model.

   The model returns the list of float values as the output array. We determine the index with the max value in the array.

   Ex: Output of Model : [0.234, 0.738]
   Y_label : [0,1]

   Using the above approach the index obtained for the Output Array is 1, the index for Y_label is also 1. The species or category can be obtained by using the mapping of index to category which is stored as a global dictionary. The final accuracy is obtained by comparing the predicted category with the golden truth category. The Predicted Value and Golden Truth values are displayed in tabular format as the results along with the accuracy as shown.

   Prediction Results for 4 Test Samples

Predicted Value	Golden Truth
virginica	virginica
virginica	virginica
virginica	versicolor
setosa	setosa

   The Accuracy is: 0.75