HW 05- Neural Networks

ISHAN JOSHI, USC ID 3451334123

07/11/2020

Neural Networks

Introduction:

A neural network is modelled after the structure and functioning of the neurons in the human brain. It is based on perceptrons and linear classifications and works on the principles of dynamic programming and stochastic gradient descent. Just like a neuron in the brain, a node in the neural network functions in two stages- the first stage involves the assimilation of the information received i.e., the calculation of the products of weights and input nodes. The second stage involves using a non-linear activation function on the output of the previous stage to present the final output of the node.

The perceptron algorithm requires the data to be linearly separable. However, this is not a necessity for neural networks. This makes it more robust for complex datasets.

For this assignment, we have a training list of images of the pgm format, which is the lowest common denominator grayscale format. The images are of the dimension 32x30. This is concatenated into a 960x1 vector, does making our input layer of the neural network at 960 nodes.

The hidden layer consists of 100 layers as mentioned in the assignment. For both these layers, we add a bias value.

Our final output layer will have just the one node, and thus the overall architecture of the neural network will have three layers including the input layer, with 960, 100 and 1 node respectively.

Stochastic Gradient Descent

While gradient descent considers a cumulative error while updating weights, stochastic gradient descent considers errors of only a specific instance. This makes it easier to compute the gradients and simplifies the process of tweaking the hyperparameters to achieve maximum accuracy on the test set. Choosing the specific instance involves randomizing the selection of the instances, which also helps local search.

Feedforward Neural Network

Feedforward Network

The feedforward network involves propagating the weights and outputs of each only in one direction. The values of the outputs cannot travel within a layer or from layer L to layer L-1. The sigmoid function used in this assignment is $Theta(S) = \frac{1}{1+\exp(-S)}$.

Where S = $\sum_{i=1}^{d(l-1)} wij(l) * xi(l-1)$, with xij being the outputs of layer L-1, wij being weights of layer L and d(l-1) being the dimensions of layer L-1.

We add a bias to both the input node and the hidden layer nodes. Thus, the weights in input layer are of dimensions (961 x 100) and the weights in hidden layer are (100 x 1).

Backpropagate Errors

We use backpropagation to update the values of weights. The error of the final layer of the network is propagated through the layers back to the input layer weights.

Mathematically, for layer L:

$$wij = wij - learning\ rate * (delta(j) * xi(L-1))$$

Member Functions

1. **Sigmoid ():** Calculated the sigmoid or activation values.
2. **Error_term_derivative ():** Calculated the derivative of the squared error.
3. **Sigmoid_derivative ():** Calculated the derivative of the activation function.
4. **Feedforward ():** Computes the feedforward stage of the neural network. It returns the outputs of hidden layer and final layer.
5. **Backprop ():** Computes the deltas and errors for backpropagation stage.
6. **Execute ():** It trains the neural network over 1000 epochs.

Outputs

The network is trained on the training set, and then we test the performance of the network on the test set. The hyperparameters as specified in the assignment present a testing accuracy of 77.10843373493977 for a specific run of the code. Training accuracy is 73.36956521739131

Weights and other hyperparameters can be modified to try and achieve a higher accuracy. The network is trained across all 184 instances for 1000 epochs.

Code Challenges

1. The neural network takes a long time to train over 1000 epochs, and accuracy achieves a maximum of 79.51807228915662 if the input layer weights are randomized from -100 to 100 for specific runs of the code, while keeping the other hyperparameters the same.
2. Calculation of the deltas and error values for backpropagation are complicated.
3. The program flow needs to be created with the objective of streamlining the computations for both the feedforward and backpropagate stages.