COS 314
Assignment 3 Report

U17047880

Morne Roos

Assignment:

Backpropagation Neural Network for Classification

Date:

12 June 2020

The Neural Network

The neural network chosen for the Iris classification problem consisted of an input layer, a hidden layer and an output layer. The number of neurons in each layer is as follows

Input Layer: 4 Neurons

Hidden Layer: 5 Neurons

Output Layer: 3 Neurons

Functions

Normalization

To avoid irregular input data causing irregular behavior within the program, all input data was normalized using Z score normalization which is calculated as follows

Z = Input – Mean / Standard Deviation

Where Z will be the normalized value of 1 of the 4 input values.

Input will be the value of 1 of the 4 input values being normalized

Mean is the Mean of the 4 input values

Standard deviation is the Standard Deviation of the 4 input values

Neuron Input

To calculate the input value of a neuron in the hidden or output layer, the following function was used:

$Z(i) = \sum((x)\ A[x]*W[x][i]) + B[i]$

Z(i) : The input value in a layer (hidden/output) at neuron i

A[x] : The activation value of the neuron x in the previous layer

W[x][i] : The weight from neuron x in the previous layer to neuron i in the current layer.

B[i] : Bias to the neuron i.

Hidden Layer

The Activation function used for the hidden layer is the Rectified Linear Units function (ReLU function).ReLU was chosen as it provides constant gradients during learning and thus improving the efficiency of the learning process. ReLU is defined as follows:

ReLU (Z) = max(Z,0)

The function returns the input value Z if it is greater than 0 (any positive Z input).

The function returns 0 if the input value Z is negative.

Output Layer

The activation function used for the output layer is the Softmax function. The softmax function outputs a vector of probability distribution over the target classes. Effectively the softmax function outputs the probability of an input instance belonging to a class. The sum of the output vector will be equal to 1 as all probabilities must sum to 1.The softmax function was chosen at it provides constant gradients even for a high error model, allowing for faster correction of model. The softmax function is defined as follows:

Softmax(Z[i]) = e^Z[i] / (∑(j) e^Z[j])

Z[i] is the input for output node i

The softmax value for an input Z at i is equal to e to the power of z[i] divided by the sum of e^j where j will be the number of inputs.

Error

The error function used was Cross Entropy Loss.

CrossEntropyLoss(Actual,Predicted) = - (∑x Actual[x] * log(Predicted[x]))

Actual[x] : target probability distribution at x

Predicted[x] : predicted probability distribution at x

This is the loss for a single training instance.

To calculate the cost of the entire model with a full input data set , we would calculate the average of Cross Entropy Loss for all input instances.

Learning Rules

Mini batch Gradient Descent was applied to the training data, weights and biases were updated after each mini-batch.

Weight update:

$W[j][k]new = W[j][k] – (n * \Delta W[j][k])$

$W[j][k]new$: the updated weight value for a weight from neuron j to neuron k

$W[j][k]$: the weight value from neuron j to neuron k being updated

n: Learning Rate of the Neural Network

$\Delta W[j][k]$: The change in weight from neuron j to neuron k to minimize the error of the model.

Bias update:

$B[k]new = B[k] – (n * \Delta B[k])$

$B[k]new$: the updated bias value for a bias to neuron k.

$B[k]$: the bias value to neuron k being updated

n: Learning Rate of the Neural Network

$\Delta B[k]$: The change in weight from neuron j to neuron k to minimize the error of the model.

$\Delta W[j][k]$ and $\Delta B[k]$ are the average negative gradients to the cost function with respect to inputs (weights and bias respectively). Effectively the average changes to weights and biases in a minibatch that would minimize the cost function. The changes are averaged over the mini batch and the update occurs after each mini batch. Mini batch gradient descent is a combination of stochastic gradient descent and batch gradient descent. The benefits of mini-batch gradient descent is that less computing is required as values are averaged over a smaller data set and because of the increased number of updates, the model will converge faster.

To calculate the number of batches and the number of inputs in a batch:

```
For(x =10; x>0 ; x++)
        If(numInputs % x ==0)
                numBatches = x
                batchSize = numInputs / numBatches
        endIf
endFor
```

For the given input data of 150 inputs, the batch size would be 150 / 10 = 15 and the number of batches = 10.

Convergence Conditions

The model used multiple convergence conditions, if any of the conditions were met at any point in the training, the training was stopped and the weights and biases after this were the final weights and biases which would be used for testing.

The convergence conditions were:

- Maximum number of epochs: if the program had performed 15000 epochs it was considered to have converged
- If any of the 2 following conditions were met 15 times in total during training, the model was considered to have converged
    - If the cost of the model at epoch x was greater than the lowest cost of any epoch prior
    - OR if the change in lowest cost of any epoch was smaller than 0.0002%

Learning Rate

As previously mentioned the updating of weights and biases is affected by a learning rate n.

Weight update: W[j][k]new = W[j][k] – (n * ΔW[j][k])

Bias Update: B[k]new = B[k] – (n * ΔB[k])

Numerous iterations of the program with differing learning rates resulted in an observation that the program was most effective and stable with a learning rate of 0.001

The learning rate affects the extent to which newly calculated gradients will impact existing weights and biases, therefore the learning rate effects the speed at which learning will take place. Learning can be too fast or too slow with errors resulting from both instances, in this model specifically it was observed that 0.001 was optimal for the speed of learning.

Parameter Values

Parameter 1: Learning rate = 0.001

Parameter 2: Batch Size = 15

Parameter 3: Maximum Epochs = 15 000