COL774: Machine Learning- Assignment 3

Neural Networks

Rajat Jaiswal (2017CS50415)

26th April 2020

Neural Networks

(a) **Neural Network Architecture:** In this part, I implemented a **generic fully connected neural network architecture** in which each unit in a hidden layer is connected to every unit in the next layer. This architecture is further used to learn a model for multi-class classification using one-hot encoding. To train the network, I implemented **backpropogation algorithm** and used **mini-batch Stochastic Gradient Descent**. Mean Squared Error(MSE) over each mini-batch was used as the loss function. The architecture is generic enough to train a model by **varying Mini-Batch Size(M), Number of features(n), Hidden layer architecture(h), and Number of target classes(r).**

(b) **Constant Learning Rate:** In this part, different neural networks having a **single hidden layer** consisting of {1, 5, 10, 50, 100} units respectively were trained and their accuracy, and training time was compared. The learning rate used was constant and set to **0.1**, and a mini-batch size of 100 was used.

No. of units in hidden layer	Training Set Accuracy(in %)	Test Set Accuracy(in %)	Time taken to train(in s)	No. of Epochs taken
1	7.63	7.69	3.18	42
5	26.10	26.09	43.99	460
10	81.17	75.77	103.54	999
50	93.85	88.44	145.52	907
100	95.59	90.00	198.86	918

Table 1:Training set accuracy, Test set accuracy, time taken to train, and no. of epochs, with varying no. of units in hidden layer for constant learning rate

Stopping Criteria: If the loss between two consecutive epochs didn't improve by a particular `threshold`, or the accuracy of the model didn't improve for n `no_of_iterations` consecutively, or the model has reached `max_epoch`, then the training stops, and the model is said to have converged. In this case, the value of `threshold` was set to $4e-5$, `no_of_iterations` was set to **15**, and `max_epoch` was set to **2500**.

The plots obtained are as follows:

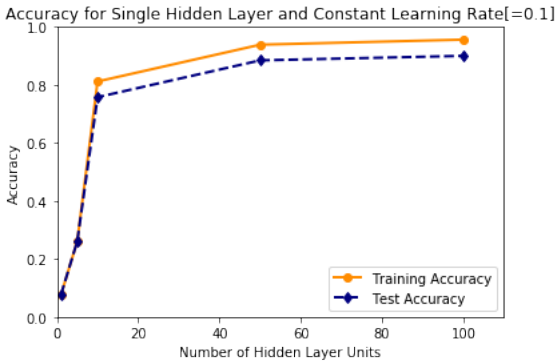

Fig 1: Accuracy vs no of units in hidden layer (Constant Learning Rate)

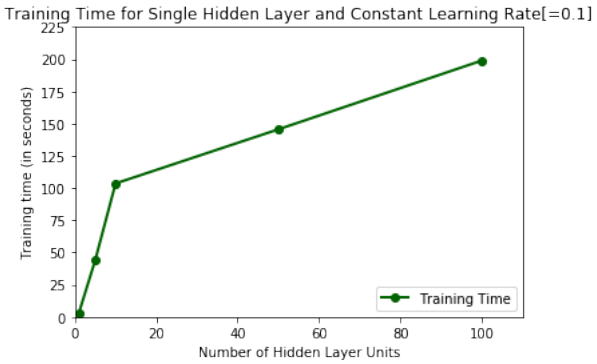

Fig 2: Training Time vs no of units in hidden layer (Constant Learning Rate)

Observations:

- It can be concluded that both training set accuracy and test set accuracy increases as the number of units in the hidden layer increases because the model becomes more adaptive and will learn smaller details. And the time taken to train the model also increases. As number of units in hidden layer increases, which means more parameters to learn, hence it leads to higher accuracy and larger training time.
- The number epochs taken for smaller neural networks(1, 5 units in hidden layer) is less. However, the neural network with 10 units in hidden layer took larger number of epochs to converge as compared to the larger networks(50, 100 units in hidden layer).

(c) **Adaptive Learning Rate:** In this part, different neural networks having a **single hidden layer** consisting of {1, 5, 10, 50, 100} units respectively were trained and their accuracy, and training time was compared. The learning rate used was adaptive and initially it was set to $\eta_0 = 0.5$. It was decreased as $\eta = \frac{\eta_0}{\sqrt{t}}$, where t is the current epoch number.

No. of units in hidden layer	Training Set Accuracy(in %)	Test Set Accuracy(in %)	Time taken to train(in s)	No. of Epochs taken
1	5.77	5.55	1.54	19
5	14.08	14.20	6.19	73
10	79.64	75.35	145.70	1413
50	91.34	87.00	158.97	964
100	92.75	88.28	220.72	1024

Table 2:Training set accuracy, Test set accuracy, time taken to train, and no. of epochs, with varying no. of units in hidden layer for adaptive learning rate

Stopping Criteria: The stopping criteria was kept the **same as before** in order **to draw better comparison**. The stopping criteria was chosen in such a way that the models in part(c) converge, and then the same stopping criteria was used in part(b) above. In fact, the same stopping criteria is used in part(d) as well.

The plots obtained are as follows:

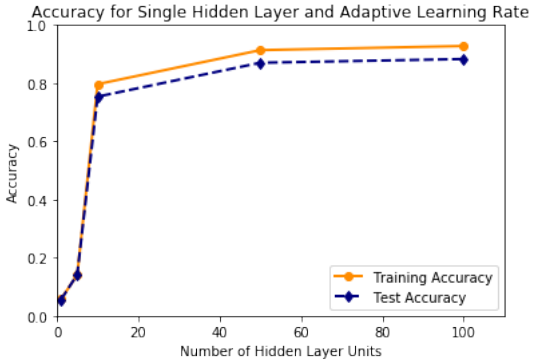

Fig 3: Accuracy vs no of units in hidden layer (Adaptive Learning Rate)

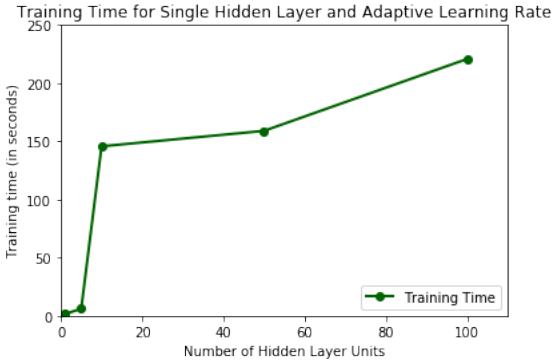

Fig 4: Training Time vs no of units in hidden layer (Adaptive Learning Rate)

Observations:

- The convergence for hidden units {1, 5} is faster, but for hidden units {10, 50, 10}is slower, as compared to part(b). The convergence for hidden unit 10 is the slowest, same as above. The models also converge to a point of lower test and train accuracy. In this part as well, accuracy improves with number of units in hidden layer.
- It may owe to the fact that even when learning rate was a constant = 0.1 in part(b), the loss was always decreasing across epochs which means that the network was always moving towards the optimum point, so using adaptive learning rate here doesn't help. In fact, the learning rate grows smaller in this case as the number of epochs increase, and hence models converges slowly, and eventually the learning rate dies out. It also diminishes the difference of loss in consecutive epochs leading to a point which is away from the optimum.
- In the beginning, there is steeper rate for convergence than the above part(b), but it becomes slower only after 25 epochs, and convergence rate goes on getting slower.
- It can be concluded that the adaptive learning rate is not very useful and doesn't make convergence any faster for this problem. The model in the previous part i.e. (b) is better. Perhaps, the hyperparameter η_0 needs tuning.

(d) **ReLU Activation Unit:** In this part, two neural networks, each having **two hidden layers** and **100** units in both the layers were trained. One model used ReLU activation function in its hidden layers while the other one used Sigmoid activation function in its hidden layers. The accuracy, and training time was compared with each other as well as with the model with single hidden layer and 100 hidden units, learnt in part(b) above. The learning rate used was adaptive, as in part(c), and initially it was set to $\eta_0 = 0.5$. It was decreased as $\eta = \frac{\eta_0}{\sqrt{t}}$, where t is the current epoch number.

Stopping Criteria: The stopping criteria was kept the **same as before** in part(b) and (c), to draw better comparison.

Activation type (No of Hidden layers)	Training Set Accuracy(in %)	Test Set Accuracy(in %)	Time taken to train(in s)	No. of Epochs taken
Sigmoid(Single, Constant)	95.59	90.00	198.86	918
ReLU(Double, Adaptive)	96.03	90.78	101.69	297
Sigmoid(Double, Adaptive)	92.99	88.72	319.37	1296

Table 3:Training set accuracy, Test set accuracy, time taken to train, and no. of epochs, for different models

Observations:

- Between the two models that uses two hidden layers, the one with ReLU activation function in hidden layers has a better accuracy and trains faster, than the one with sigmoid activation function in its hidden layers.
- Among the above three models, the one with two hidden layers and ReLU activation function in its hidden layers is the best, as it has higher accuracy than the rest of two and is learnt/trained significantly faster. The accuracy of double-layered model with ReLU activation units appears to be similar to the single-layered model with sigmoid activation units, but convergence happens in half the time, which means that ReLU activation performs better than sigmoid.
- The double-layered model with sigmoid activation units gives a lower accuracy than the one in part(b), but that happens possibly because of the adaptive learning rate used in this part which leads the network to converge to a point of lower accuracy. It is also slower because it has more parameters to be learnt and uses adaptive learning rate to do so.
- ReLU has its own advantage. It has a reduced chance of gradient to vanish, Sigmoid has a flat plateau, which is in a small range near zero, and there the gradient is also very small. Due to this backpropagation algorithm works better for ReLU as it has a large domain for non-zero gradients, and zero gradients will not pass on. Also, the computation of ReLU and its gradient is much faster as compared to sigmoid, thus makes the training faster.

(e) **MLP Classifier:** In this part, I used scikit-learn's `MLPClassifier` to build the neural network, similar to that built in the previous part(d) with ReLU activation units in hidden layers.

Activation type (implementation)	Training Set Accuracy(in %)	Test Set Accuracy(in %)	Time taken to train(in s)	No. of Epochs taken
ReLU[Scratch-Part(d)]	96.03	90.78	101.69	297
ReLU[sklearn]	100.00	92.18	46.71	152

Table 4:Training set accuracy, Test set accuracy, training time, and no. of epochs, for 2 implementations

Scikit-learn's `MLPClassifier` with modified loss function leads to a model with higher training and test set accuracies, and it also takes much less time as compared to our similar implementation in part(d). It gave the best accuracies for both training and test sets amongst all the models. And also its training time is significantly less than any other similar implementation of our own.