COL774 Assignment 3B

Siddhant Mago, 2017CS50419

April 2020

Neural Networks

(a) In this part, I created a class for the neural network which worked on the Mean Squared Error (MSE) loss.

$$J^b(\theta) = \frac{1}{2M} \sum_{i=(b-1)M}^{bM} \sum_{l=1}^{r} (y_l^{(i)} - o_l^{(i)})^2$$

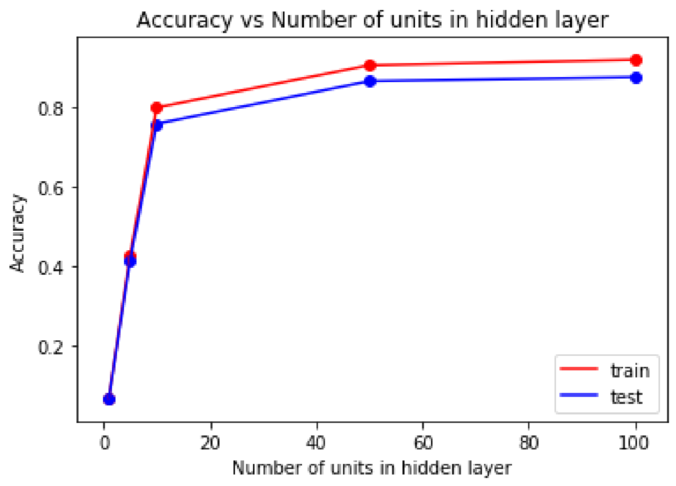
The network was trained using mini batch stochastic gradient descent and the back propagation algorithm was implemented using first principles.

(b) In this part, the neural networks with a single hidden layer consisting of 1, 5, 10, 50, 100 units respectively were trained and their accuracy was compared.

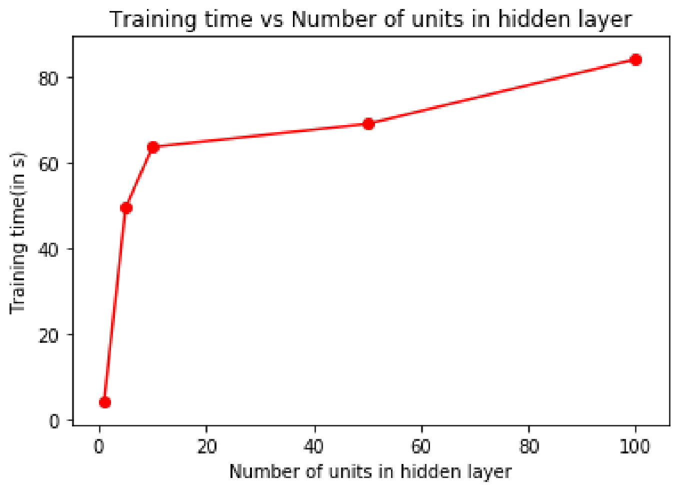
Stopping criteria: The neural network was trained using stochastic gradient descent. If the loss function didn't improve by at-least `Epsilon` for `n_iter_no_change` consecutive iterations, the convergence was assumed to be reached and the algorithm stopped. Here, `Epsilon` was taken as 10^{-4}, `MAX_EPOCHS` as 1000 and `n_iter_no_change` as 10. The data and plots obtained are:

Units in hidden layer	Train Accuracy	Test Accuracy	Time(in s)	Epochs
1	6.8 %	6.7 %	4.31s	53
5	42.5 %	41.2 %	49.5s	596
10	79.8 %	75.8 %	63.63s	681
50	90.5 %	86.5 %	69.025s	437
100	91.9 %	87.5 %	84.1s	467

The plots obtained were:



(a) Accuracy



(b) Training Time

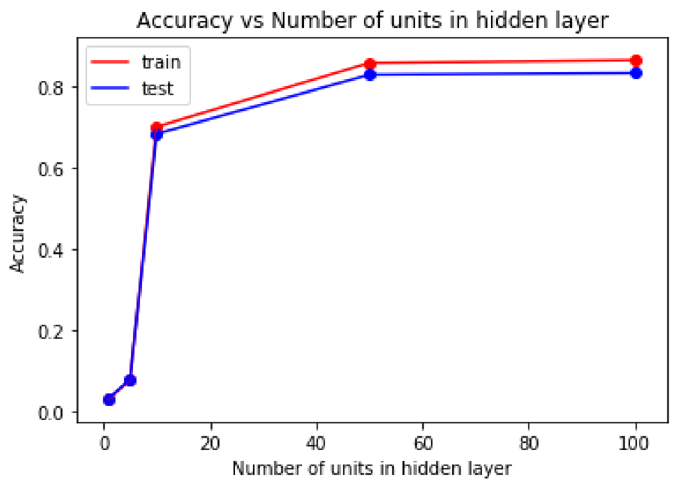
Figure 1: Desirable units in hidden layer

- From the above graph, we can conclude that the accuracy(both test and training) increases as the number of units in the hidden layer increases. However, the time taken to train the network also increases. This behaviour is expected as more number of units means more parameters which leads to higher accuracy and larger training time.
- The model with 10 units in hidden layer takes a very large number of epochs to converge. This maybe because, although the model at 10 units in hidden layer is expressive enough to lead a large increase in accuracy(over the 5 units), it is still a small network and thus there are only a few choices of parameters which give good decision boundaries, so it takes large number of epochs for this network to train.

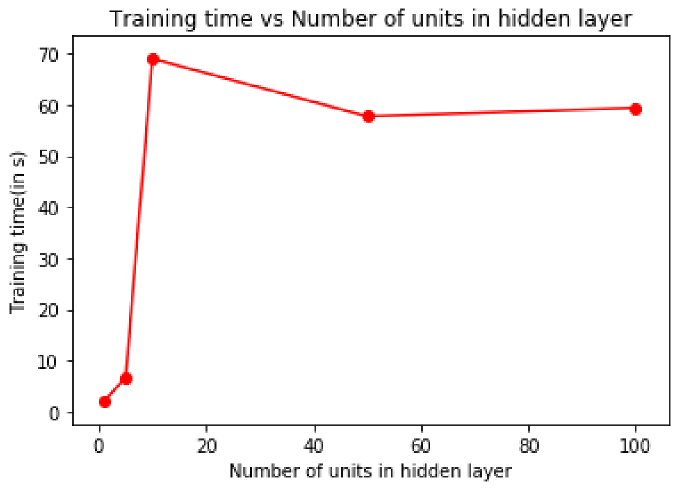
(c) In this part, adaptive learning rate was used. The learning rate was decreased as $\frac{\eta_0}{\sqrt{t}}$, where $\eta_0 = 0.5$ and t is the current epoch number. The stopping criteria was kept the same as before in order to draw better comparison.

Units in hidden layer	Train Accuracy	Test Accuracy	Time(in s)	Epochs
1	3.1 %	3.3 %	2.02s	23
5	7.7 %	7.9 %	6.60s	77
10	70.1 %	68.4 %	68.99s	751
50	85.9 %	83 %	57.52s	355
100	86.6 %	83.4 %	59.39s	322

The plots obtained were:



(a) Accuracy



(b) Training Time

Figure 2: Desirable units in hidden layer

- In this part as well, accuracy improves with number of units in hidden layer.
- As compared to part b, the convergence for all the cases (except 10) is faster but they also converge to a point of lower test and train accuracy. This happens because even when learning rate was a constant = 0.1, the loss was always decreasing across epochs which means that the network was always moving towards the local optimum, so using adaptive learning rate here doesn't help.

- The network where number of units in hidden layer is 10 takes even more time than before now. This is because, the number of epochs for convergence for that particular model is high and with adaptive learning rate, as the number of epochs increases the updates to parameters keep getting smaller even when the gradients are relatively large, leading to slower training.

(d) In this part, `ReLU` activation units were used instead of sigmoid in the hidden layers. The derivative of `ReLU` at $x = 0$ was considered to be 1. A network containing of 2 hidden layers each consisting of 100 units was trained in order to draw comparisons.

Activation type	Train Accuracy	Test Accuracy	Time(in s)	Epochs
Sigmoid(1 layer, Part B)	91.9 %	87.5 %	84.1s	467
Sigmoid(2 layer)	84.56 %	82.10 %	164s	562
ReLU(2 layer)	92 %	87.17 %	49.9s	199

- We observe that ReLU activation gives higher accuracy(both test and train) as compared to sigmoid activation and also converges faster. The accuracy of 2-layer ReLU appears to be similar to the 1 layer sigmoid (part B) but convergence happens faster, which means that ReLU activation performs better than sigmoid. This is because in ReLU the problem of vanishing gradients does not arise.
- The 2-layer sigmoid gives a lower accuracy than the one in part B, but that happens maybe because of the adaptive learning rate used in this part which leads the network to converge earlier but to a point of lower accuracy.

(e) In this part, I used Sklearn's `MLPClassifier` to build the neural network, similar to that built in the previous part. The accuracy with the modified loss were:

Network type	Learning rate	Train Accuracy	Test Accuracy	Time(in s)	Epochs
ReLU(Part D)	0.5(invscaling)	92 %	87.17 %	49.9s	199
ReLU(Sklean)	0.5(invscaling)	87.6 %	85.0 %	163s	1025
ReLU(Sklean)	0.1(constant)	100 %	92.4 %	20s	124

Training using sklearn's `MLPClassifier` with a high inverse scaling learning rate leads to poor results, but with a constant learning rate we get much better results. This is because:

- In this part as well, the model keeps on moving towards the optimum value (without oscillating), so using an adaptive learning rate doesn't help, and makes the model converge to a point of lower accuracy.
- The Binary cross entropy loss is $L_{BCE} = -y\log(x) - (1-y)\log(1-x)$ which means that the gradient shoots up when x = 1 or 0, so BCE usually leads to larger gradients and thus the learning rate should be kept low.