CSE 446 HW3

Rohan Mukherjee

May 15, 2024

A1.

(a) In general deep neural networks are non-linear optimization problems, and often times tend to be non-convex and have many local minima. Gradient descent provides great results but not provably (nor likely) the best.

(b) When training deep neural networks, it is much better to initialize the weights according to some random uniform distribution, instead of all zeros because all zeros could get stuck immediately in local mimina and not be able to escape them with gradient descent.

(c) Yes, using non-linear activation functions allows the model to be much more complex and have non-linear decision boundaries.

(d) No. Using DP and the theorem that calculating gradients can be done up to small constants in the same time complexity as just evaluating a function, backpropogation runs in the same time complexity as the forward pass.

(e) Sometimes neural networks can be too complex leading to too much bias, or simply have too many parameters to train in a reasonable amount of time. In this case the neural network may not be the best model to use.

A2.

Notice that,

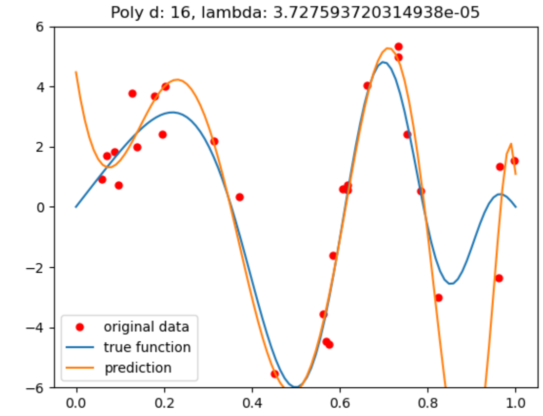
$$
\begin{aligned}
K(x,x') &= \sum_{i=0}^{\infty} \frac{1}{\sqrt{i!}} e^{-x^2/2} x^i \cdot \frac{1}{\sqrt{i!}} e^{-x'^2/2} x'^i = \sum_{i=0}^{\infty} \frac{1}{i!} e^{-(x^2+x'^2)/2} \cdot (xx')^i \\
&= e^{-(x^2+x'^2)/2} \cdot e^{xx'} = e^{-(x^2+x'^2-2xx')/2} = e^{-(x-x')^2/2}
\end{aligned}
$$

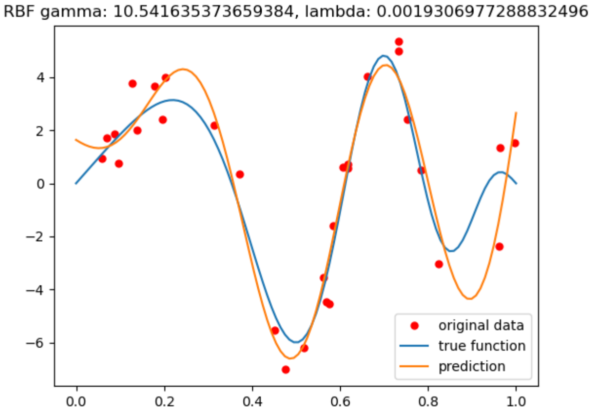
Where we used that,

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

A3.

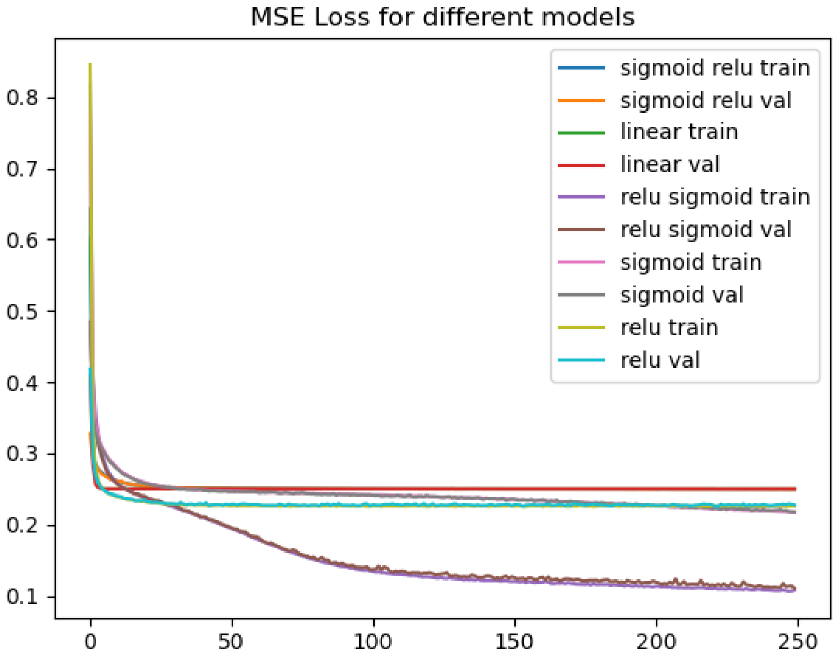
The optimal hyperparameters used are in the title of the below graphs:

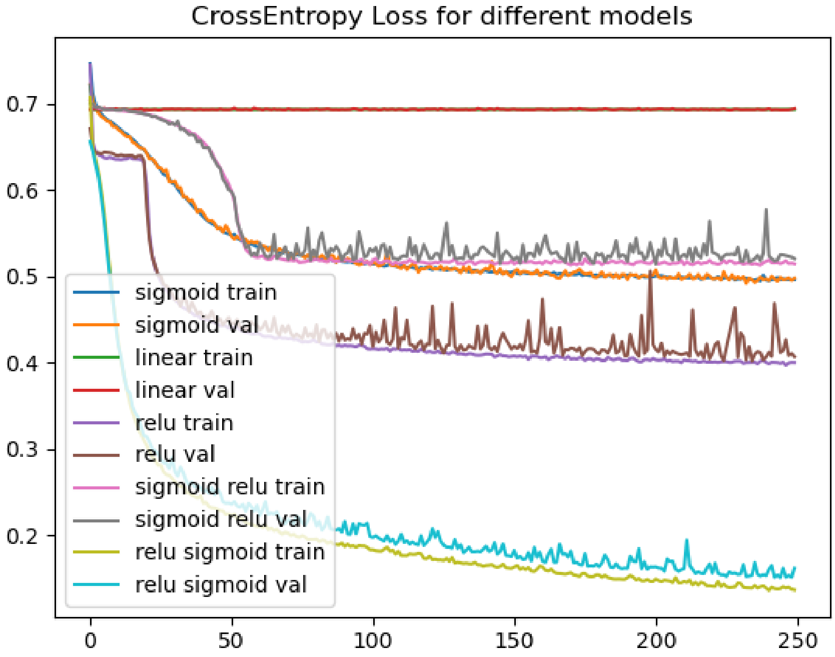




A4.

b. Here are the plots of the train and validation set losses for each loss function:





c. The best performing architectures are seen from above to be relu sigmoid in both cases. It is honestly a little shocking how much it leaves the competition in the dust! Here are the accuracy plots for each:

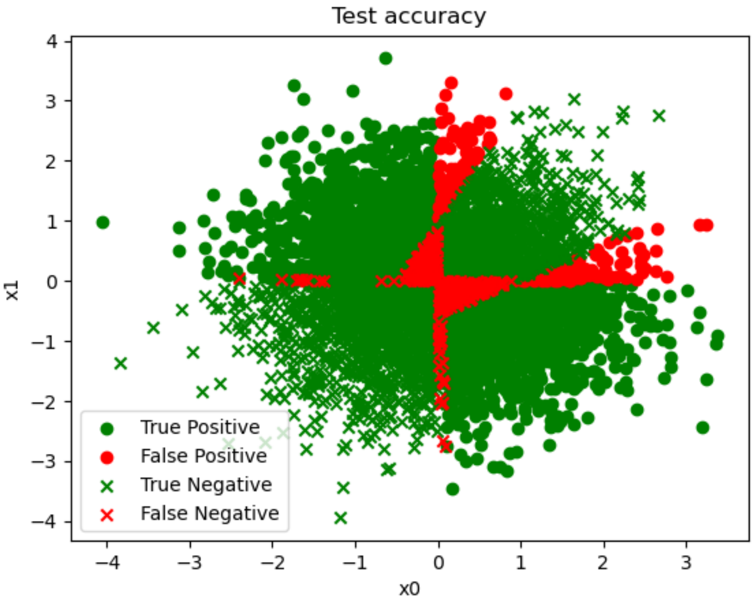


Figure 1: MSE

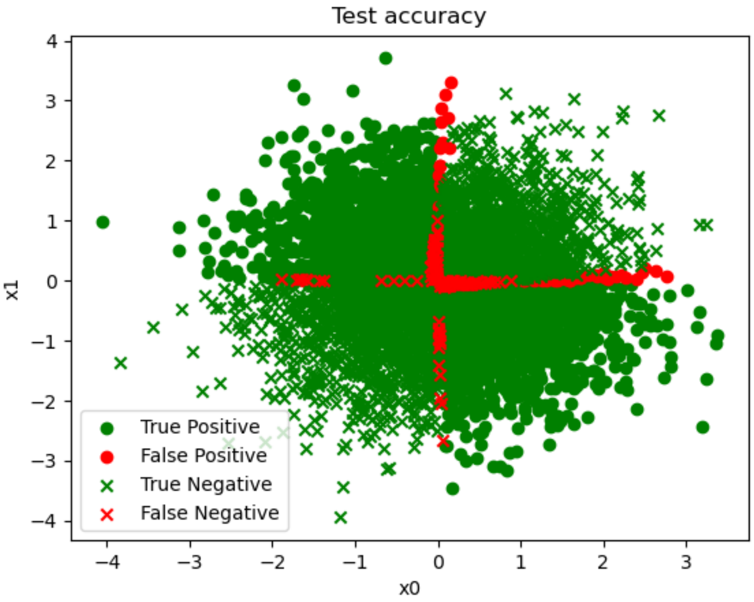
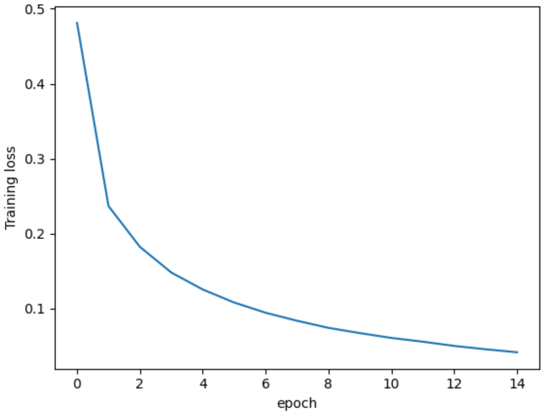


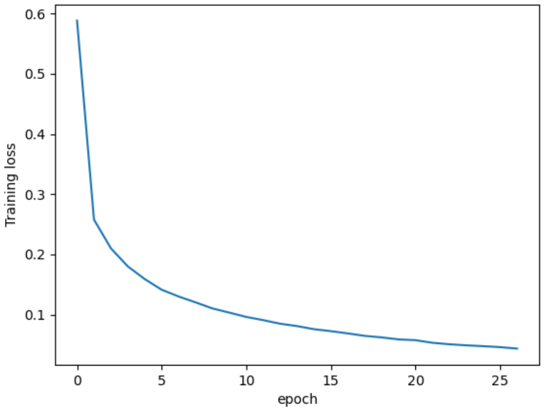
Figure 2: Cross Entropy

A5.

(a) I get an accuracy of 0.97557% and a loss of 0.0816% on the test data. Here is the graph of the training loss:



(b) On the test data, after training to 99% accuracy on the training data, I get an accuracy of 96.668% and a loss of 0.113. The graph I get is as follows:



(c) The first model, F1, has 50,890 parameters while the second has only 26,506 parameters. The first one is seen to perform better as per the above stats. I believe this is because it has a lot more parameters and thus can train a more complex model, yielding a higher accuracy. One would guess that if you have more complexity then you would be better able to represent the data.

A6.

This homework assignment took me significantly longer than the previous, probably around 25-30 hours. I really enjoyed it!