Project 14: Text Centering

Joseph Carmack

December 9, 2016

Abstract

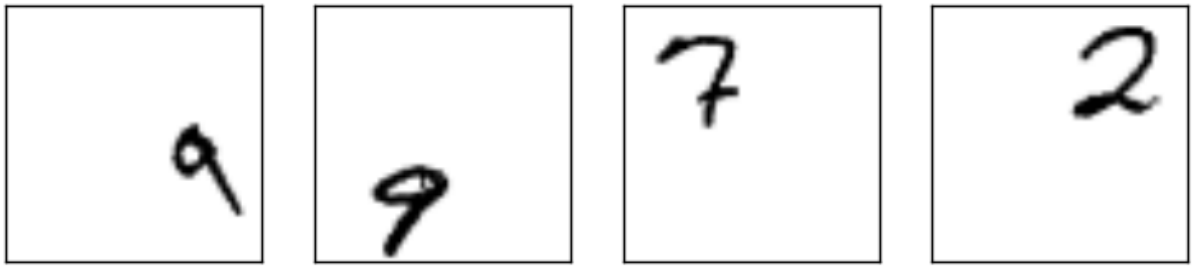
The goal of this project was to make some progress in using machine learning to read and make sense of digital micrographs of old, hand-written documents. Because data collection and preprocessing to get the data into a format suitable for training is a huge task, the mnist data set was used as a substitute. The challenge addressed was that of centering hand-written text. the Mnist data set was modified for this task by adding additional pixels to make the images bigger followed by a random shift in the hand-written number to off-center the hand-written numbers. The shifts were stored as labels to be predicted by the machine learning system. A feed-forward neural network was then trained and tuned to get optimal results. The number of hidden layers and units per layer were varied. L1, L2, and no regularization were also experimented with. Results will be discussed in the body of the report.

1 Modifying Mnist

In modifying the Mnist data set I decided the images needed to be enlarged so that they could be shifted significantly away from the center in a random way. So the first thing I did was to write some code to enlarge the images by adding 14 extra pixels around the existing images. After this I gave the original 28×28 pixels a random pixel shift in the horizontal and vertical directions with a maximum shift in either direction of 14 pixels. Examples are shown in Figure 1. The final images were 56×56 pixels resulting in a one-dimensional vector representation with 3136 elements.

2 Neural Network Trianing and Tuning

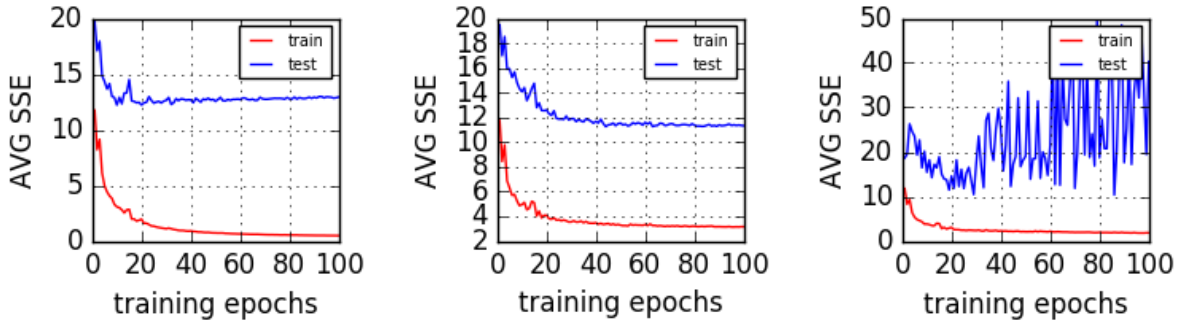
In order to be able to get rapid feedback on changing meta parameters, I used a subset of the training and testing shifted mnist data (training on all the training data and testing on all the testing data took 4+ hours to run on my workstation). For the training subset I used the first 1000 training patterns as opposed to the entire 60,000. For the testing subset I used the first 100 patterns from the testing data out of the 10,000 total patterns. After getting a feel for the optimal meta parameters I then scaled up and trained on the entire training data and tested on the entire testing data.



(a) (b) (c) (d)

Figure 1: Sample of enlarged and shifted mnist images

The meta parameters that I tweeked included the number of layers, the number of units per layer, L1 or L2 or no regularization, and regularization strength. The error metric I used was the average summed squared error. I chose this error metric so that results would be comparable when switching between the data subsets and the full data sets. I plotted the avg SSE versus number of training epochs for both the training data and the testing data. This plot would allow me to see how the different sets of meta parameters were influencing how well the model was generalizing and also be able to see when overfitting was occuring.

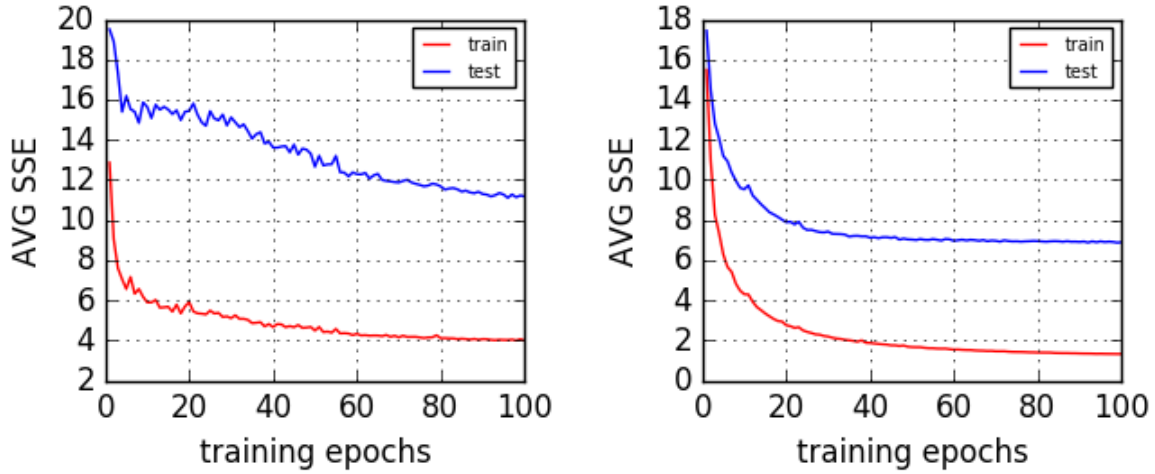


(a) no regularization (b) L_2 regularization (c) L_1 regularization

Figure 2: Comparing Regularizations using one hidden layer with 100 units.

I started with a single hidden layer with 100 units. I used this topology to explore what was best for regularization. I tried a number of different regularization strengths, λ, for both L1 and L2 regularization and the best results are depicted in Figure 2. As can be seen from these plots it seems that L2 regularization seems to work best. L1 regularization achieves fairly good results but is very unstable. I think this is because some weights get so small that there is either numercal cancellation going on or some weights become alternate between being essentially zero and non-zero. I am not sure the exact reason. Regardless, I chose to stick with L2 regularization.

Next I moved on to varying the number of layers and the number of units per layer. One question of I had going into this next step of tuning my neural network was, "Why do we use more than one hidden layer at all?" I did did some analysis, and discovered that



(a) Single hidden layer with 268 activation units. (b) Three hidden layers with a (200,60,8) topology.

Figure 3: Comparing the benefit of hidden layers with a total of 268 activation units.

if you keep the total number of units in the neural network constant, the total number of weights was also constant regardless of how many layers you organized those units into. I also spent some time playing around with plotting either summed or nested Hyperbolic Tangent functions. When using a single hidden layer the result is a large sum of activation functions evaluated at their individual net values. On the other hand, for more than one hidden layer with the same amount of activation units, you get a sum that has fewer terms because the activation functions get nested. From my qualitative observations it seemed that a sum of Hyperbolic Tangent functions resulted in greater flexibility than the same number where they were nested. This lead me to wonder if I increased the total number of activation units in my neural network, then would a single hidden layer work better than arranging the units into additional hidden layers.

Figure 3 shows the error plots for two cases. The only difference between the two cases is the arrangement of activation units into hidden layers. In the first case, Figure 3a, only a single hidden layer was used with a total of 268 activation units. In this graph the test error gets down to a value between 11 and 12. It also might not have fully converged because you don't see any evidence of the onset of overfitting yet. However, I do think it is quite close to convergence and would be very surprised to see the test error improve below a value of 10. In the second case, Figure 3b, the same number of activation units is used, 268, except that they are organized into three hidden layers as opposed to only one. The layer topology is 200, 60, and 8 activation units in the first, second, and third hidden layers. In the plot you can see that the test error gets down to a value of about 7 which is much better than the first case. At least for this data set, I would say that this is conclusive evidence that using multiple hidden layers works better than simply piling all the activation units into a single hidden layer.

I tried several other topologies for one, two, and three hidden layers. Interestingly, I found that the 200, 60, 8 three layer topology worked best out of all my different topology selections which included topologies with as much as 388 total activation units. Consequently

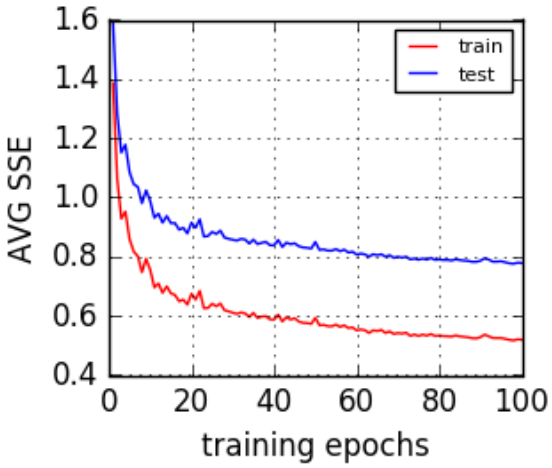
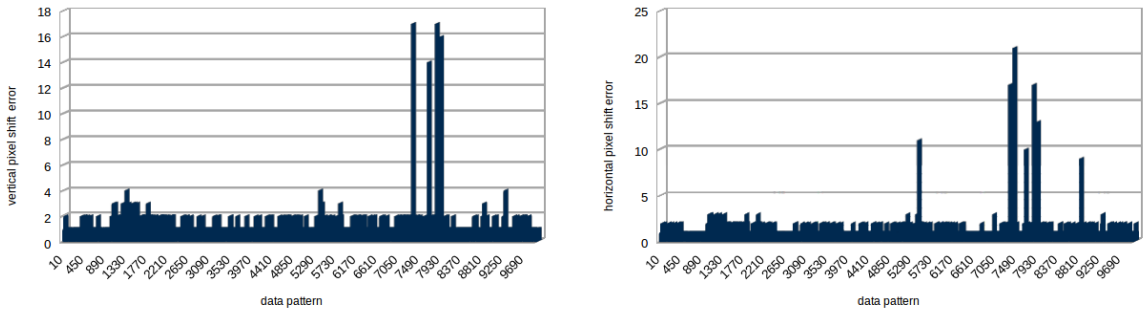


Figure 4: Training and testing on all the data (60,000 training patterns and 10,000 testing patterns).

I decided to use this topology, along with L2 regularization to train and test on the entire data. The error plot for running on the entire data is shown in Figure 4. As can be seen from the plot, the neural network achieves a pretty good generalization with the average pixel shift error being less than a single pixel.

3 Analysing Final Results

I spent some time analysing the predictions made by my trained network on the testing data. First of all the predictided labels were not integral but had non-zero decimal values. The actual labels were integer numbers representing the number of pixels the original mnist image was shifted vertically and horizontally. Consequently, I decided to round all the predictions to their closest integer values. Next I decided to plot the absolute difference between the predictions and the actual labels. Figure 5 shows the results for the vertical and horizontal absolute differences. I sorted the patterns when generating these plots by the



(a) Vertical pixel shift error.

(b) Horizontal pixel shift error.

Figure 5: Absolute difference between predicted pixel shifts and actual shifts.

patterns corresponding number (0-9). This way I could see if there was a particular number that gave the network more trouble at recentering than the others. From examining the Figure 5, it is clear that sevens, ones, and fives seem to have the worst errors, in that order.

I also generated the actual images for the worst 20 contenders. These are shown in Figure 6. Interestingly I noticed that all of the images in Figure 6 had very large shifts in at least

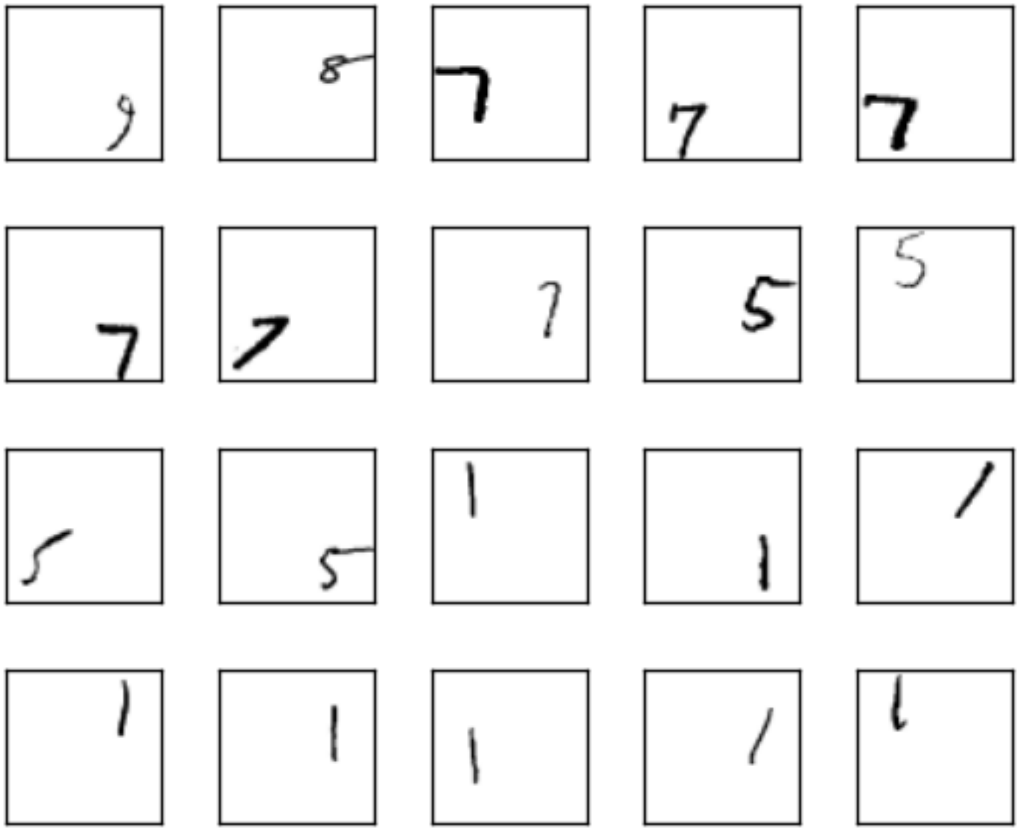


Figure 6: Top 20 patterns that the neural network struggles most to recenter.

one of the horizontal or vertical directions but usually both (being placed in a corner). So for some reason the neural network struggles most to recenter sevens, ones, and fives, and especially those that are in corners. I am not exactly sure why this is.

4 Lessons Learned

From this project I have learned a lot. First of all I experienced that the size of the meta parameter space for neural networks is staggeringly huge. There are so many ways I could have continued to try to tune this network that I didn't try. There are a huge number of regularization methods and network topologies that I didn't try. I also only tried one activation function and there are many others that could have been tested.

I did learn that when keeping the number of activation units constant, using more than one hidden layer produced better results than just sticking all the units in a single hidden

layer. I am not sure how well this generalizes to other datasets but I am guessing there are many data sets where this is also true. Of course their may be those where this is not true.