Project #4 COMP-135
Prof. Roni Khardon
Sydney Strzempko

Backpropagation on Neural Networks

Results

Part 1:
Running 3000 iterations on 838 dataset

See Attachment A

Final Node Output:

[0: 0.42741799158306287, 1: 0.44672312745262921, 2: 0.46841232304226849, 3: 0.42941048920134939,
4: 0.44159111329778711, 5: 0.4192103965210342, 6: 0.48529478027339584, 7: 0.45246965922670995]

Part 2:
Running 200 iterations on varying depths and widths of network on optdigits test & train datasets

d = 5, w = 5 - See Attachment B
d = 5, w = 10 - See Attachment C
d = 5, w = 15 - See Attachment D
d = 5, w = 20 - See Attachment E
d = 5, w = 30 - See Attachment F
d = 5, w = 40 - See Attachment G
d = 5, w = 50 - See Attachment H
w over d for this set - See Attachment I

d = 0, w = 10 - See Attachment J
d = 1, w = 10 - See Attachment K
d = 2, w = 10 - See Attachment L
d = 3, w = 10 - See Attachment M
d = 4, w = 10 - See Attachment N
d = 5, w = 10 - See Attachment O
d over w for this set - See Attachment P

Observations & Conclusions

On the first set of iterations (10-attribute dataset 1-hot wired to 10-class output), the test set accuracy takes about 1500 iterations to begin to accurately predict the classification of the following iteration over the train set. I predict that given more iterations, the network would increasingly correctly classify further examples from a train or test set. This makes a lot of sense given the conditions of the network being built; the learning rate variable is set to a rather low number, there are only about 8 or so examples in the set to build off of, and there are only 8 features across the entire set, the same as the number of values, with an example of each class alone in the set. Thus, the network being built (see the hidden node representation above) averaged to a little less than .5 across the whole set, which makes sense given the inputs of each class and the corresponding rounding of that prediction to a number within the classifying range [0,1] on output nodes Xn. Although the node structure is perfectly matched to the input range of features and the output range of numbers, the small number of examples given in each train set severely limits the ability of the algorithm to learn and build with each iteration; this is reflected by the number of iterations needed to begin to produce accurate results.

On the set of iterations with a fixed d=5 and ranged w=[5,10,15,20,30,40], there is a general pattern of a learning curve followed by a stabilization of error - ie, the neural network's equivalent of overfitting. These learning curves and error plateaus become increasingly pronounced as the width of the neural network increases. In the d=5,w=5 instance, for example, this learning curve dominates the 200 iterations with a slow and steady decline of error rate - we don't really see a plateau, however we do see a slow uptick of error rate around 150 iterations that might suggest a sinusoidal steadying out of the error rate over more iterations. On the opposite end of the spectrum, in the d=5,w=40 instance, the learning curve manifests as a dipping of error rate below 40% by the 15th iteration, followed by an increase then an almost-perfect plateau of error hovering around 80% for the following 180 or so iterations. Although these two examples are very different, by examining the ranged ws from 5 leading up to 40, we see this manifestation of a dip in the error rate, followed by a plateau or even slow rise in error rate suggesting that after the initial error dip, the neural network no longer benefits from iterative applications of the train set. Thus, the larger the width, the more efficiently the corresponding neural network is able to correctly classify new training instances. However, after a certain optimal number of iterations is reached, the training set overfits the neural network and its classifying error rate actually increases. This is reflected in Attachment H, where seemingly, the best test error happens at the d=1 and d=3 cases, when in reality this corresponds with the overfitting of the corresponding sets as w increases. This is because the test errors reflected on this graph are the test error of the 3000th iteration - as we have seen above, some of the best test errors for this range of widths happen earlier on in the neural network learning iterations.

On the set of iterations with a fixed w=10 and d=[0,1,2,3,4,5], in contrast, the final graph of test sets (Attachment O) *does* accurately reflect the overall error rate, and almost non-curve of the varying iterative graphs (preceding attachments). The best depths are d=1,d=2,d=3; these indicate that a lower depth will test better (at least in the case where we have an input of 64 features, mapped to 10 different output nodes). The d=4 and d=5 sets don't ever correctly classify the train or test sets, unfortunately; I would hazard that these show that a moderate change in depth can completely affect the neural network's ability to correctly predict certain sets significantly more than a change in the width of nodes. Thus, although slightly miscalibrated here with a depth of d=4 compared to d=3 and w=10, the learning curve is massively affected, versus having a width of w=5 compared to w=15 and d=10. It seems that in finding the correct widths and depths to parameterize a neural network by, the best method would be to find a depth or few depths which minimize error, then find a width which minimizes error on top of that.