Cormick Hnilicka
Garritt Moede
CS638 - Deep Learning with Neural Networks

Lab 2 Report

- Introduction -

Lab 2 proved to be a very challenging but educational project for us. Neither of us had previously written a multilayered neural network before so there was a large domain of background information needed to be researched. One-hot encoding was a new concept for the both of us that ultimately led to the most frustration during development. We were unaware of how to represent the feature vector as an output of one of the seventeen input nodes. Our first approach was to pass the index of the one hot “1.0” as the output of the input node but quickly realized that this was the incorrect way of doing it as they were no longer categorical variables but now numerical variables. This gave some amino acids a higher value than others which prevented our neural network from being trained properly. Upon realizing that we were to represent the 17 input nodes as 21 nodes, one for each possible value of the amino acid, we were able to properly train, tune, and test our neural network.

Once our neural network was working properly and our backpropagation algorithm was training our network, we began to experiment with the parameters of our basic backpropagation algorithm. Small variances with the parameters, such as raising the learning rate from 5% to 10% did not seem to have a large effect on the ability for the network to be properly changed given a large enough time. However, adding terms like ‘momentum’ and ‘drop out’ had a large effect on the overall output and behavior of the network. Below we go into a bit more detail on the effects these terms had on our network.

- Preliminary Notes -

These notes are just to discuss some different configurations we ran with our network that we did not end up pursuing, namely 1000 Hidden Unit layers and ReLU for hidden units.

Using **ReLU** for our hidden layer substantially reduced accuracy across all network configurations, so we decided to remove these trials from this report. Below is confusion matrix test set data using ReLUs for our hidden layer.

Confusion Matrix Data run on TEST

3200.0 | 2.0 | 0.0
1491.0 | 1.0 | 0.0
1124.0 | 2.0 | 0.0

Accuracy: 0.5453151618398637
Error Rate: 0.4546848381601363
Recall alpha-helix: 0.0
Recall beta-strand: 6.702412868632708E-4
Recall coil: 0.9993753903810119
Precision alpha-helix: NaN
Precision beta-strand: 0.2
Precision coil: 0.5503009458297506

Additionally, due to the processing power available to us, we were not able to get timely data for as many **1000 hidden unit trials** as we would have liked. We have included confusion matrix data for a 1000 hidden unit trial using our best network configuration below.

Confusion Matrix Data run on TEST

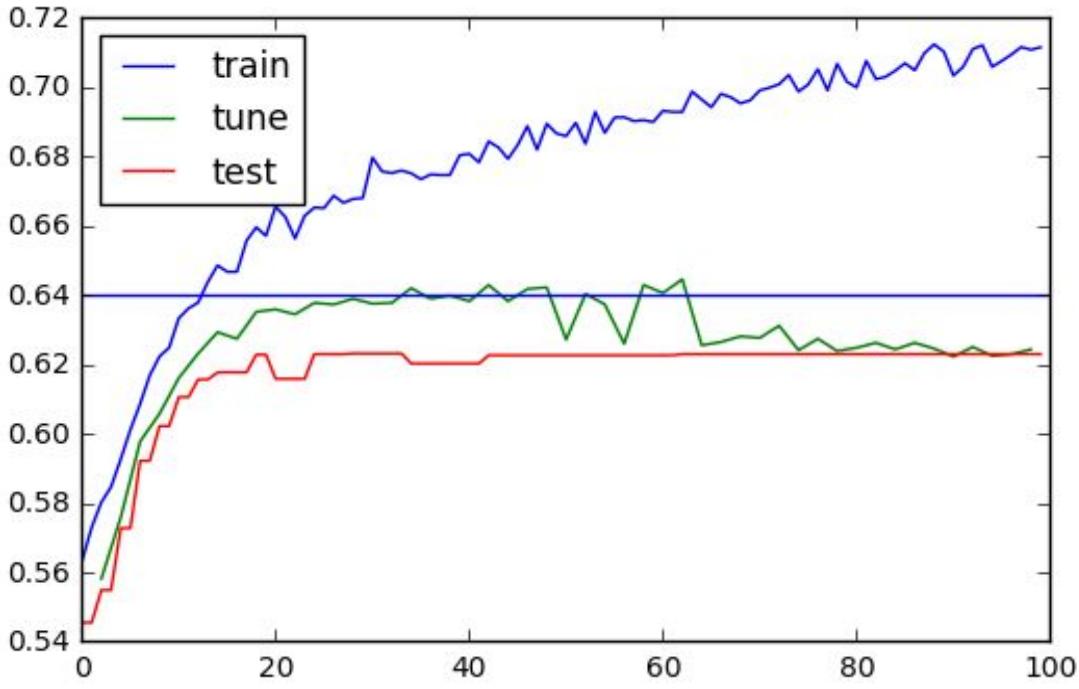
2780.0 | 213.0 | 209.0
885.0 | 520.0 | 87.0
612.0 | 192.0 | 322.0

Accuracy: 0.6170357751277683
Error Rate: 0.3829642248722317
Recall alpha-helix: 0.28596802841918295
Recall beta-strand: 0.3485254691689008
Recall coil: 0.868207370393504
Precision alpha-helix: 0.5210355987055016
Precision beta-strand: 0.5621621621621622
Precision coil: 0.6499883095627776

- Early Stopping -

Our early stopping process was used to reduce overfitting when training on our training set. Our implementation of it used a threshold of min_error to break out of training. Every 2 epochs we would evaluate the current network weights on our tuning set and compare to our threshold. Once our network performed better than our min_error threshold of **.358**, we continued to evaluate on our test set.

Below is a figure representing our best network configuration (see Network Trials below) settings against our early stopping criteria. X-axis = epochs, Y-axis = accuracy.



- Network Trials -

Default

These trials were run on our network using no momentum term and no drop out. The trials for each were run over a maximum of 100 epochs.

A. ANN with 5 Hidden Units and a learning rate of 0.05
    a. Accuracy: 0.5981260647359455
    b. Error Rate: 0.40187393526405446
B. ANN with 10 Hidden Units and a Learning Rate of 0.05
    a. Accuracy: 0.6037478705281091
    b. Error Rate: 0.3962521294718909
C. ANN with 30 Hidden Units and a Learning Rate of 0.05
    a. Accuracy: 0.6076660988074958
    b. Error Rate: 0.3923339011925042
D. ANN with 100 Hidden Units and a Learning Rate of 0.05
    a. Accuracy: 0.5761499148211243
    b. Error Rate: 0.4238500851788757

Average Accuracy with a learning rate of 0.05 and no momentum or drop out.

Average Accuracy = **0.59642248722317 => 59.64%**

Momentum

These trials were on the same network as before, however, this time we implemented a momentum term in order to better control the fluctuation of the weight changes during our backpropagation algorithm. This helps the network escape deceiving local minima as well as continue to train in a seemingly flat portion of the error space. These trials were again ran over a maximum of 100 epochs unless our early stopping condition was met.

A. ANN with 5 Hidden Units, a Learning Rate of 0.05, and a Momentum Term of 0.9
    a. Accuracy: 0.6078364565587734
    b. Error Rate: 0.3921635434412266
B. ANN with 10 Hidden Units, a Learning Rate of 0.05, and a Momentum Term of 0.9
    a. Accuracy: 0.6129471890971039
    b. Error Rate: 0.3870528109028961
C. ANN with 30 Hidden Units, a Learning Rate of 0.05, and a Momentum Term of 0.9
    a. Accuracy: 0.6076660988074958
    b. Error Rate: 0.3923339011925042
D. ANN with 100 Hidden Units, a Learning Rate of 0.05, and a Momentum Term of 0.9
    a. Accuracy: 0.5838160136286201
    b. Error Rate: 0.4161839863713799

Average Accuracy with a learning rate of 0.05 and a momentum term of 0.9:

Average Accuracy = **0.603066439523 => 60.31%**

Drop Out

These trials were run on our original network but instead of momentum, we implemented a dropout term. The dropout term is used in attempts to further reduce the chances of overfitting by removing a percentage of the hidden units during its computations. This removes the corresponding weights and outputs from the delta weight calculations which prevents overfitting to the training set. It also provides us with a more robust network to run our training set on by giving us variance in its calculations which can better generalize the target output. These trials were run over 100 epochs unless our stopping condition was met.

A. ANN with 5 Hidden Units, a Learning Rate of 0.05, and a Drop Rate of 0.5
    a. Accuracy: 0.6158432708688245
    b. Error Rate: 0.3841567291311755
B. ANN with 10 Hidden Units, a Learning Rate of 0.05, and a Drop Rate of 0.5
    a. Accuracy: 0.6173764906303236
    b. Error Rate: 0.3826235093696764

C. ANN with 30 Hidden Units, a Learning Rate of 0.05, and a Drop Rate of 0.5
    a. Accuracy: 0.6224872231686541
    b. Error Rate: 0.3775127768313459
D. ANN with 100 Hidden Units, a Learning Rate of 0.05, and a Drop Rate of 0.5
    a. Accuracy: 0.6224872231686541
    b. Error Rate: 0.3775127768313459

Average Accuracy with a learning rate of 0.05 and a drop rate of 0.5:

Average Accuracy = **0.61954855195911 => 61.96%**

Momentum and Drop Out

These trials were run over our network with both drop out and momentum implemented. Based on the results above when using one or the other of these techniques, we would expect that our network would net a higher accuracy than when not implemented. These trials were run over 100 epochs unless our early stopping criteria was met.

A. ANN with 5 Hidden Units, a Learning Rate of 0.05, a Drop Rate of 0.3, and a Momentum Term of 0.9
    a. Accuracy: 0.6170357751277683
    b. Error Rate: 0.3829642248722317
B. ANN with 10 Hidden Units, a Learning Rate of 0.05, a Drop Rate of 0.3, and a Momentum Term of 0.9
    a. Accuracy: 0.6170357751277683
    b. Error Rate: 0.3829642248722317
C. ANN with 30 Hidden Units, a Learning Rate of 0.05, a Drop Rate of 0.3, and a Momentum Term of 0.9
    a. Accuracy: 0.6202725724020443
    b. Error Rate: 0.37972742759795575
D. ANN with 100 Hidden Units, a Learning Rate of 0.05, a Drop Rate of 0.3, and a Momentum Term of 0.9
    a. Accuracy: 0.617206132879046
    b. Error Rate: 0.382793867120954

Average Accuracy with a learning rate of 0.05 and a momentum term of 0.9 and a Drop Rate of 0.3:

Average Accuracy = **0.61788756388416 => 61.79%**

Variances on Network Parameters

The trials above were reproduced with different variances on the parameters. We did this in order to get a better understanding of the effect that each parameter had on the network. Their configurations and average accuracies are as follows:

A. ANN with a Learning Rate of 0.05 and a Drop Rate of 0.7
    a. Average Accuracy = 0.61686541737649 => 61.68%
B. ANN with a Learning Rate of 0.1 and a Drop Rate of 0.5
    a. Average Accuracy = 0.61852640545145 => 61.85%

Analysis

Our neural network produced the best results when utilizing either drop out or momentum and generally were affected the most by changes to the dropout term. This follows our prediction that a more robust network to train on can prevent overfitting to the training data. Our data also suggests that the best configuration of the network was to use 30 hidden units as opposed to a higher or lower amount. We believe this is because a large number of hidden units could over-generalize the data which in turn reduces the overall accuracy of the network. Overall this project was a great learning experience for the both of us in terms of getting a better understanding of multilayer neural networks as well as the actual implementation of one from scratch. As we move forward, we plan to use techniques that we learned from this lab in future experiments and networks.