Report

Wayne Chew Ming Chan (9071997606)
Sparsh Agarwal (9075905142)

CS 638
Lab-2

Introduction

The aim was to predict the protein structure from a set of protein strands.
We used back propagation along with one-hot-encoding and implemented early stopping over it. Further for experimentation we tried to implement weight decay, dropout, and momentum but the accuracy takes a hit on doing so.
We implemented the backpropagation using multiple hidden layers to train our weights of multiple HUs. In backpropagation, the activation function used is of two types: Sigmoid and rectified linear. If we use sigmoid for all the layers including the output layer, the accuracy increases but on use of rectified linear the accuracy becomes stagnant after a certain value.
The accuracy varies mainly on following factors:

1. Learning rate
2. Number of hidden units

Data:

There are 128 proteins protein strands and we have divided the data into 3 sections of train, test and tune. The training happens on trainset, early stopping is implemented based on accuracy of tune set and final result is obtained from test set.

Amino acids:
Train Set: Used all except the one used in tune and test set …. Amino acids and their neighbours for train set
Tune Set: Used the 5th, 10th, …. Amino acids and their neighbours for tune set
Test Set: Used the 6th, 11th, …. Amino acids and their neighbours for test set

Features:(encoded using one-hot encoding)
[a, c, d, e, f, PADDING_AMINO_ACID, g, h, i, k, l, m, n, p, q, r, s, t, v, w, y]
There are 20 + 1 features. We use “PADDING_AMINO_ACID” as a padding at the front and back of the protein. Max size of padding is 8.

Labels:(e-beta, h- helix , _-coil)
[e, h, _]

Window Size:
The window size is 17, with [8 padding/data, amino acid, 8 padding/data].

Neural Network Design:

The most basic unit of the neural network is a perceptron node.

We design a Perceptron Class with the functionality:

1) **FeedForward Function**
    a. Multiply the inputs with the weights of the perceptron
    b. The result is then pass through an activation function and the output is returned
    c. You can choose between a ReLU or Sigmoid function
2) **BackPropagation Function**
    a. Derivative of the error is pass into the function as a parameter
    b. Derivative is multiplied with the derivative of the activation function and the derivative of input_x for weight_x
    c. The weight is updated with the derivative
    d. Part of the derivative is then multiplied with the weight and returned by the function. This derivative is going to be passed to the perceptron before it

We also have a Neural Network Class:

1) Stores the hidden layer and the output layer
2) Connect the input with the hidden layer and the output layer
3) **Train Function**
    a) Given a list of examples, feedforward the examples and backpropagate the error
4) **Test Function**
    a) Given a list of examples, predict the labels for the examples and print out the statistics (accuracies of beta, helix, coil and overall)
5) **Export and Import Weights**
    a) Export allow the users to obtain a copy of all the weights in the neural network, stored in a specific data structure that store weights
    b) Import allow the users to set all the weights in the neural network by call the function with a specific data structure that store weights
6) **Predict**
    a) Given an example, predict the label

Early Stopping

The training of the neural network uses early stopping. There are two variables for the early stopping method. (p = Patience, e = Epoch)

The neural network is trained for (p) passes. For each pass, the neural network train itself with a training set (e) times. At the end of each pass, the neural network is tested with a tuning set. If the accuracy is higher than the previous result, the (p) is resetted back to 0 and the optimal weights are stored.

This method prevents overfitting with a patience. The accuracy of the training set will keep increasing when it is trained. However, when it starts to overfit, the accuracy of the tuning set will decrease. As the accuracy of the tuning set decreases for (p = patience) times, the training stops and the optimal weights are restored. This is the method we use to prevent overfitting and it works nicely.

1-of-N-encoding versus 1-hot-encoding

1-of-N-encoding: [0, 1, …, N] for N distinct feature values
1-hot-encoding: [1, 0, 0, …] for the 1st distinct value, [0, 1, 0, …] for the 2nd and so on … N
We implemented both 1-of-N-encoding and 1-hot-encoding. 1-of-N-encoding results in a maximum accuracy of around 56%, it never goes higher than that. As for 1-hot-encoding, the maximum accuracy is around 62%.

In the end, we chose the 1-hot-encoding as it gives a better and stable accuracy.

Rectified Linear Function versus Sigmoid Function

At the first place, we used ReLU for the hidden layer and sigmoid for the output layer. But, the accuracy never increases and stay around 58% most of the time. We then switch to sigmoid function for the hidden layer and this gives us an accuracy of 62% most of the time.

In the end, both the hidden layer and the output layer uses the sigmoid function.

Number of Hidden Units and Learning Rate

After running a couple of tests, we found out that the optimal number of hidden units is in the range of (2 - 10) with a learning rate of (0.01 - 0.005). We believe that this is the main factor that affects our accuracy.

HU	2			4			6			8		
Accuracy of Beta:	0.663230245	0.5143184422	0	0.7789232532	0.5418098511	0	0.5246277205	0.5085910653	0	0.5899198167	0.6036655212	0
Accuracy of Helix:	0	0.4104712042	0	0.02827225131	0.2691099476	0.009424083 77	0.1078534031	0.3089005236	0.001047120419	0.08586387435	0.2439790576	0
Accuracy of Coil:	0.750976562 5	0.7622070313	1	0.5776367188	0.796875	0.997558593 8	0.7338867188	0.7807617188	0.9990234375	0.6625976563	0.7207031 25	1
Overall accuracy:	0.546181630 5	0.6197110423	0.528379773	0.4876160991	0.6093911249	0.529411764 7	0.5325077399	0.6031991744	0.528121775	0.504127967	0.5768833849	0.528379773
learning rate	0.5	0.05	0.005	0.5	0.05	0.005	0.5	0.05	0.005	0.5	0.05	0.005
time in millisecond	1979	2383	865	1826	3027	2653	7739	3883	2327	8694	6654	3772

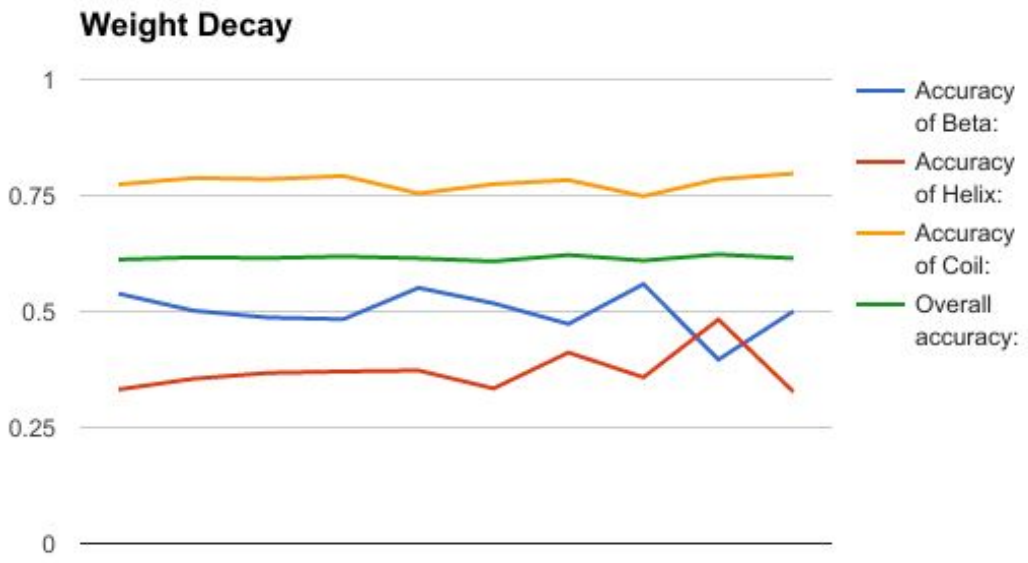
Since we have best accuracy while using 2 hidden units and learning rate as 0.05. That is the most favorable accuracy of 61.9%

Experimental Implementation:

Implementing Weight Decay:

On reducing weights by some random amount between 90-99%, we get following results

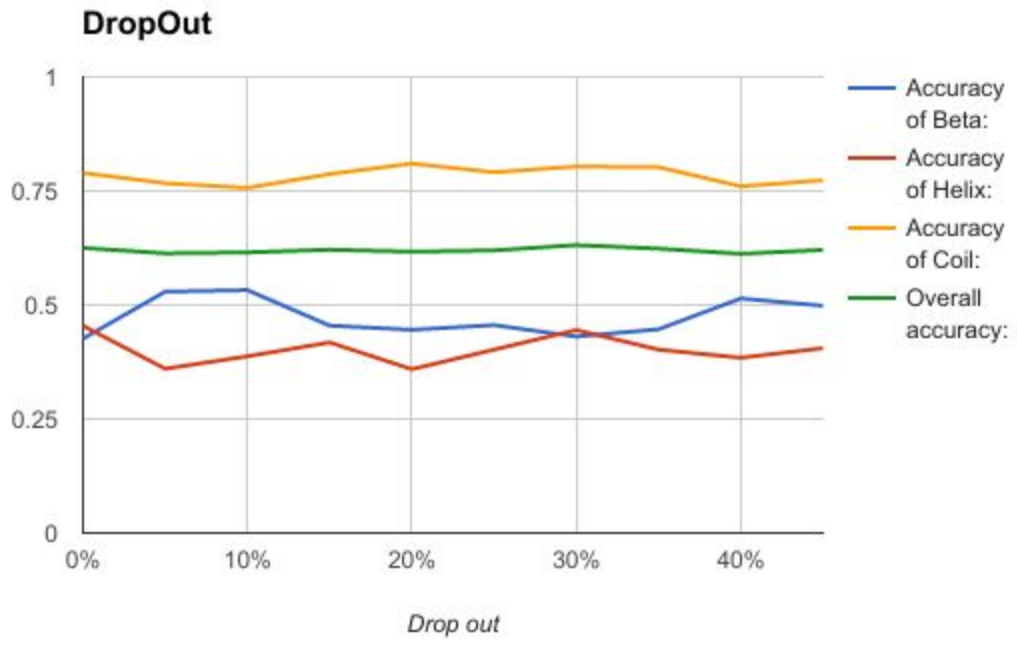
Weight decay (random decay between 90% and 99%)										
Accuracy of Beta:	0.53837 3425	0.50171 82131	0.48682 70332	0.48339 06071	0.55097 36541	0.51775 48683	0.47308 13288	0.55899 19817	0.39633 44788	0.50057 27377
Accuracy of Helix:	0.33193 71728	0.35497 3822	0.36753 9267	0.37068 06283	0.37277 48691	0.33403 14136	0.41151 83246	0.35811 51832	0.48272 25131	0.32670 15707
Accuracy of Coil:	0.77392 57813	0.78759 76563	0.78515 625	0.79199 21875	0.75439 45313	0.77441 40625	0.78320 3125	0.74853 51563	0.78515 625	0.79736 32813
Overall accuracy:	0.61197 11042	0.61661 50671	0.61506 70795	0.61867 90506	0.61455 10836	0.60810 11352	0.62177 50258	0.60964 91228	0.62306 50155	0.61455 10836



Implementing DropOut:

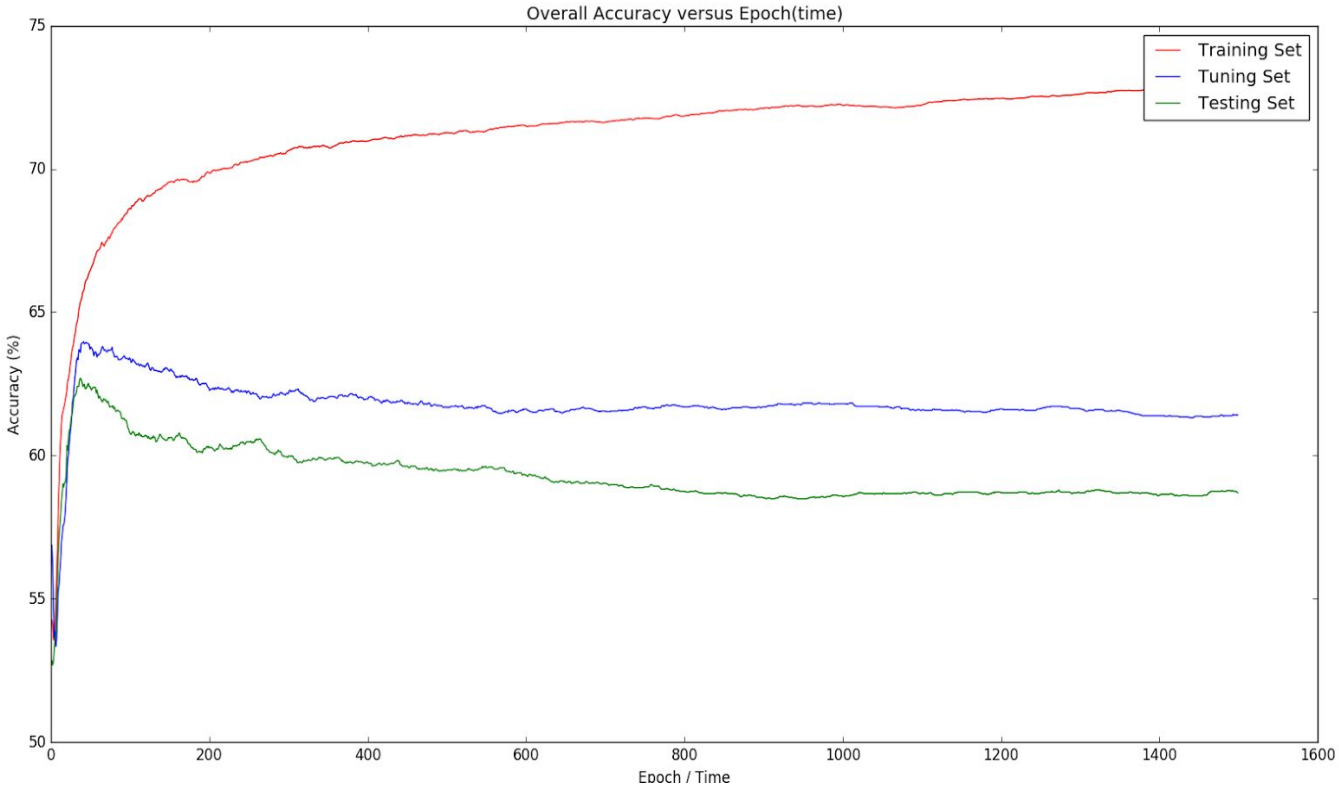
On dropping out weights on random basis as below percentage

Drop out	45%	40%	35%	30%	25%	20%	15%	10%	5%	0%
Accuracy of Beta:	0.4982817869	0.5143184422	0.4467353952	0.43069874	0.4558991982	0.4455899198	0.4547537228	0.5326460481	0.529209622	0.4261168385
Accuracy of Helix:	0.4052356021	0.3842931937	0.4020942408	0.445026178	0.4020942408	0.3591623037	0.4178010471	0.387434555	0.3602094241	0.4544502618
Accuracy of Coil:	0.7734375	0.7602539063	0.8022460938	0.8037109375	0.791015625	0.8100585938	0.787109375	0.7568359375	0.7670898438	0.7895507813
Overall accuracy:	0.6207430341	0.6122291022	0.6235810114	0.6313209494	0.6197110423	0.61687 3065	0.6212590299	0.6153250774	0.6132610939	0.625128999



We are not using either dropout, momentum or weight decay because all of them reduce the accuracy of the test set.

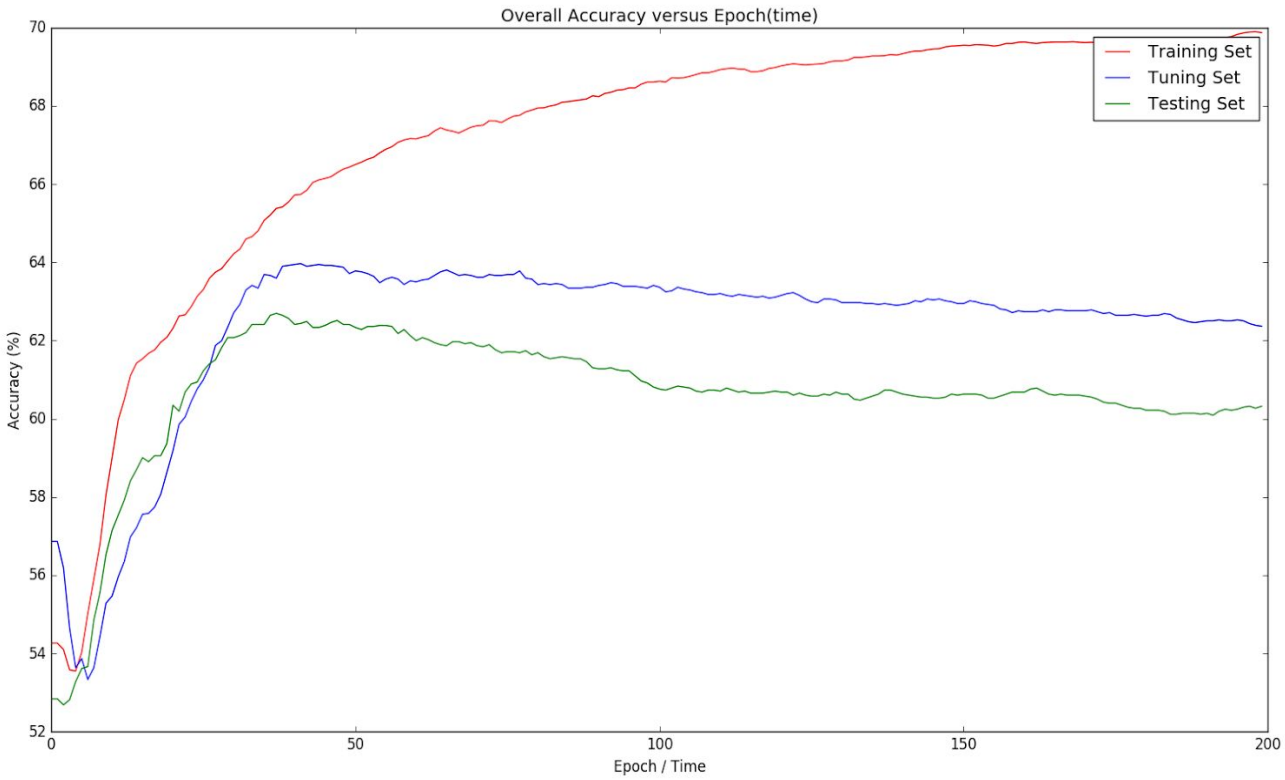
Best Results



Overall Accuracy versus Epoch

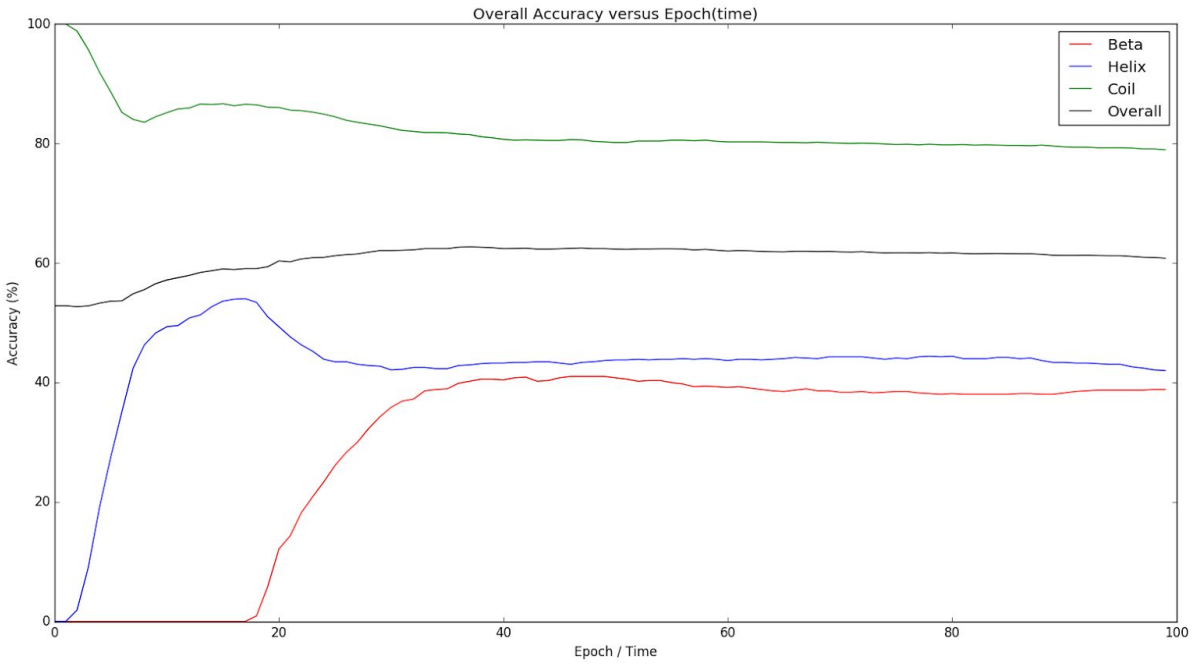
Results of training the neural network with the training set for 1500 epoch. The accuracy for the training (red), tuning (blue) and testing (green) set is plotted against epoch. This uses a learning rate of 0.025 and 3 hidden units.

We can see that the accuracy of the training (red) set keeps on increasing and its accuracy is getting close to 75%. And the accuracy of the tuning (blue) set maintains at an accuracy of around 62.5% while the testing (green) set maintains at an accuracy of around 60%. This is because the neural network starts to overfit at around 100 epochs, which is the point where the accuracy of the tuning (blue) and test (green) set starts to decrease.



Overall Accuracy versus Epoch (Zoom in version at 0-200 epoch)
We're zooming in to examine the graph more clearly. If you compare the tuning (blue) set and the testing (green) set, you can see that although both the tuning and the testing set reach their peak at about the same epoch. The gap between the two lines increase more as epoch/time increases. This is because the accuracy of the testing set is decreasing faster than the tuning set when the neural network starts to overfits. Hence, we have to be aware when using early stopping as we might overfit the tuning set. Since, the tuning set is used to estimate early stopping, a large patience might cause the neural network to overfit the tuning set.

Based on the graphs, a patience of about 30 epoch seems to work best.



Accuracy (Beta, Helix, Coil, Overall) versus Epoch
This is a very interesting graph.
First, we define the accuracy of beta as:

The number of correctly predicted beta protein / Total number of beta protein

This goes the same for Helix and Coil. While Overall is the combined accuracy for all three of them.

More than 50% of the data are labelled as Coil. This can be seen in the graph where the prediction of Coil starts with an accuracy of around 100%. This makes sense as most of the protein strands are made of coil. Only a small part of these are made of beta and helix.

The accuracy of helix (Blue) increases first, then the beta (Red) starts to increase while the helix (Blue) decreases slightly. And then both helix and beta maintain a stable accuracy of around 40%. At the same time, the coil (Green) maintain a high accuracy of 80% because most prediction are coils. The overall accuracy increases from the baseline (53%) to around 62%. The accuracy should never go below 53% as most of the protein strands are made of coils and coils prediction should be easy. If it goes below 53%, something must be very wrong.

The prediction of helix and beta are hard compare to coil because we have very little data of helix and beta. The dataset is skewed towards coil which takes up at least 50% of the data. Hence, we don't have enough positive example of helix and beta to make more accurate prediction.

Final Result:

Setup:

2 hidden units, 0.05 learning rate, 25 patience, 1 epoch, sigmoid function for both layers
This is the best/optimal setup that we come up with for this data set. We are aware that it might not be accurate for other protein datasets. Might have to change the number of hidden units depending on the dataset.

Average Result:

Average beta accuracy: 47.29782359679267%
Average helix accuracy: 41.13298429319373%
Average coil accuracy: 77.8662109375%
Average overall accuracy: 61.930598555211 54%

Best Result:

Max beta accuracy: 55.555555555556%
Max helix accuracy: 49.63350785340314%
Max coil accuracy: 82.275390625%
Max overall accuracy: 63.02889576883385%