Project lab notes

2016

Project

Exploration and implementation of algorithm for classification of signal-peptides.

---

19 December 2016

We have decided to mainly focus on implementing a classifier using some sort of HMM approach. If there is time we will also try to do something using an RNN. The first step will be writing code to handle and import the data.

20 December 2016

Today we implemented the first attempt at using an HMM. We used the hidden states from the data to train two models. One on the positive data and one on the negative data. We then used these to score the data points in the test set, choosing the model that had the highest probability score.

```
Number of positive samples labeled positive 528, total number of positive 528
Number of negative samples labeled negative 118, total number of negative 534
precission: 55.932203389830505
recal: 100.0
accuracy: 60.8286252354049
avrg positive neg prob: -4.320928254863987
avrg positive pos prob: -1.8246559833753646
avrg positive neg prob: -3.372393241005475
avrg positive pos prob: -3.1031834877775832
```

Figure 1: Results from first run.

This approach did not fare so well. It seems that the mean probability of the negative model is much lower, creating a classifier that is overly prone to positive classification.

We will now try a different approach, were we instead train a model on all the samples, and have it predict a hidden state sequence for the given protein sequence. We then use the hidden state sequence to predict the class of the data by looking for "C".

I have also played around with an RNN but not quite fully grasped how to use it

```
Epoch 1/3
2388/2388 [==============================] - 2725s - loss: 0.6689 - acc: 0.5917
Epoch 2/3
2388/2388 [==============================] - 2827s - loss: 0.6630 - acc: 0.6591
Epoch 3/3
2388/2388 [==============================] - 2889s - loss: 0.6408 - acc: 0.6642
Accuracy: 66.17%
```

Figure 2: Results from first run.

21 December 2016

Today we finished an single HMM approach to the problem. In this we use the hidden-state data to create a markow model for all the peptides in the training data. Then to classify new sequences we use the model to predict the the hidden state of the sequence, and then look at the produced hiden-state to decide if the sequence is a signal peptide. To decide if it is an signal peptide we simply look for a C in the state sequence.

```
Evaluated on full data set:
Predicting.
Done.
--------------------
All true 127
All positive 128
True positive 117
True negative 128
Precission 0.9140625
Recal 0.92125984252
Accuracy on test data: 92.11%
--------------------
Evaluated on non-tm:
Predicting.
Done.
--------------------
All true 120
All positive 118
True positive 110
True negative 106
Precission 0.932203389831
Recal 0.916666666667
Accuracy on test data: 92.31%
--------------------
Evaluated on tm:
Predicting.
Done.
--------------------
All true 7
All positive 10
True positive 7
True negative 22
Precission 0.7
Recal 1.0
Accuracy on test data: 90.62%
--------------------
```

Figure 3: Stats from the second run

22 December 2016

We are now trying to use our model to analyze the proteome. We realized that our model had no way of handling errors in the data, or '*' and had to adjust for this.

I also continued playing with the rnn. I have realized that the first attempt was implemented badly. It would be interesting to try it but i would have to preprocess the data by first adding beginning and end of sentence markers and then concatenating all the sequences. I am still unclear of weather i should send the data in as onehot vectors or as integers. For now i will put this on ice. Concentrating on the hmm.

27 December 2016

We have made some runs on the proteom and the result seems satisfactory

```
HUMAN

human_prediction = evaluate(model, encoded_data_human, human_data_labels, states_map)

Predicting.
Done.
--------------------
All 215929
All true 12931
All positive 17181
True positive 10626
True negative 196443
Precission 0.618473895582
Recal 0.821746191323
Accuracy on test data: 95.9%
--------------------


MOUSE

mouse_prediction = evaluate(model, encoded_data_mouse, mouse_data_labels, states_map)

Predicting.
Done.
--------------------
All 124168
All true 7756
All positive 9966
True positive 6246
True negative 112692
Precission 0.626730885009
Recal 0.805312016503
Accuracy on test data: 95.79%
--------------------
```

Figure 4: Stats from the run on human and mouse proteom

28 december 2016

Today we implemented controls for our experiment to ensure the validity of our results. We did this by generating random codon sequences of different length and assuming these should be negative and by adding noise at the end of positive and negative sequences. In these test we found that the classifier was overly prone to classify long random sequences as positive. Looking at the state sequences that the classifier created we identified that when the sequences were getting longer we were occasionally predicting "C" states in non signal peptide sequences of states. To adjust for this we changed the final step of the classifier to look for a valid sequence of states instead of just looking for a "C". This made accuracy on randomly created sequences of length 10,000 go from about 10 % to about 90 %. It also slightly improved the stats on the test data and proteoms. The other test did not find any anomalies.
