Projekt lab notes

2016

Project

Exploration and implementation of algorithm for classifcation 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 hiden 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, chosing the model that had the higthes 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 hiden state sequence for the given protein sequence. We then use the hiden state sequence to predict the class of the data by looking for "C".

21 december 2016

Using the single model approach mentioned earlier

This model was trained without dropout and with an embedding layer that was probably unneccesary. The run took about 2 hours.

```
model = Sequential()
model.add(Embedding(27, 24, input_length=max_len))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train_padded, Y_train, nb_epoch=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test_padded, Y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Figure 2: Code for the first run.

```
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 3: Stats from the first run

21 december 2016

Today we finished an 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 hiden state of the sequence, and then look at the produced hiden-state to decide if the sequence is a signal peptide.

22 december 2016

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