Signal peptide prediction

Timothy Bergström

January 12, 2018

Contents

Abstract

A neural network classifier was created to predict signal peptides in proteins. The model was trained with X amount of data, which yielded a classification accuracy of 94% of the test set. The model was used to predict the total amount of signal peptides in the Humans and Mouse proteome, which yielded Z and Q SPs respectively, an error rate of .... This is showing that the model is capable of learning general structures of signal peptides ...

1 Introduction

Signal peptides (SPs) are short regions in peptides that are on average between 16 to 30 amino acids long, but can be as short as 8 and as long as 65 amino acids [1]. SPs function is to translocate proteins to different parts of the cell, such as insertion into membranes, moving proteins to organelles or to help the cell excrete the proteins [2]. When analyzing an organisms proteome, the SPs can give an indication where in the cell the proteins are translocated to, which could be of an interest when researching an organism. Not all proteins contain SPs, which is why signal peptide prediction is useful for determining the layout of the cell and for annotating organisms proteomes.

Due to the large amount of peptide data available, manually annotating each peptide would require many man-hours, while creating and using a classifier for SPs would be more efficient.

2 Methods and models

2.1 The structure of an Signal Peptide

SPs are located near the N-terminal of a protein and have three regions; the n-region, the h-region and the c-region.

The n-region is the first region, which usually contains positively charged amino acids, making the region very polar and hydrophilic. Then comes the h-region, which contains mostly hydrophobic amino acids that forms an α-helix and lastly the c-region, which is usually a sequence of 3 to 8 non-charged polar amino acids. After the c-region, there is usually a cleavage site that have a tendency to be surrounded by small, uncharged amino acids.

The problem with signal peptide prediction is that polar regions in trans-membrane proteins can mistakenly be predicted as the h-region of a SP, which makes predicting SPs in trans-membrane proteins difficult.

2.2 Design choices

The neural network was written in python with Keras 2.0 and Tensorflow 1.4 as the backend. Keras can be used to quickly prototype and develop neural networks while the Tensorflow backend can utilize a gpu to run the calculations.

2.3 Data sets

A small sample of region-annotated proteins were provided by the course in `fasta` format, containing mostly eukaryotic proteins. The samples also contained information if the proteins were transmembrane proteins or not. The data set contained both positive (contains a SP) and negative samples (does not contain a SP). Some samples was extracted from NCBI ftp server, by extracting proteins annotated with signal peptide. Few positive samples and a large set of negative samples were obtained, mostly bacterial proteins. Many of the positive samples were scraped from Signal Peptide database. A majority of the samples were bacterial proteins. Most samples were extracted from Swissprot protein database. Mostly eukaryotic proteins, both negative and positive.

Proteomes from Homo sapiens and Mus musculus were obtained from Swissprot, a database of high-quality manually annotated proteins [3].

Type	Positive	Negative
TM	10	30
non-TM	1	2
Unknown	a	b

Table 1: Data table.

Typical signal peptide from dataset:

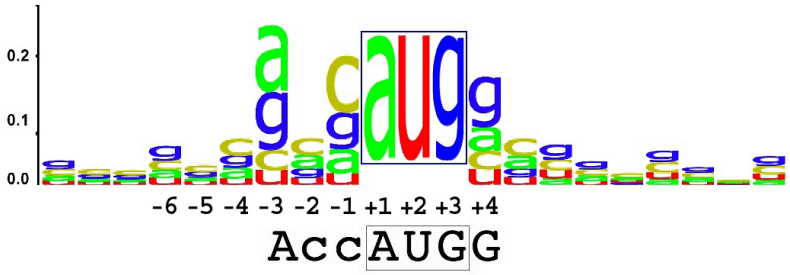

Figure 1: Extracted data from all positive samples

2.4 Data preprocessing

The protein sequences (samples) were contained in `.fasta` files. The first amino acid was removed, due to high bias towards some amino acids, mainly Methionine, see Figure 1. The following 30 amino acids from each sample were taken and sequences that were too short were padded with X. Then, the samples were cleared from all amino acids that were not one of the 20 standard amino acids. Unknown amino acids were labeled as X. Each amino acids were then one-hot encoded and merged into one 2d array for each sample, creating a 3d array for all the samples. Negative samples were labeled as 0 and positive samples were labeled as 1. The data was lastly undersampled, making the ratio between negative and positive samples 50/50.

2.5 Neural Network structure

The neural network was based on bidirectional LSTMs, due to LSTMs effectiveness with sequences and pattern recognition [4]. The bidirectionality of the network was added due to its success in other areas of machine learning with sequences and time-series predictions [5] [6]. The input was one 2d array and the output was one float, ranging from 0 to 1. A regularization layer was added between the LSTM layers to prevent overfittning.

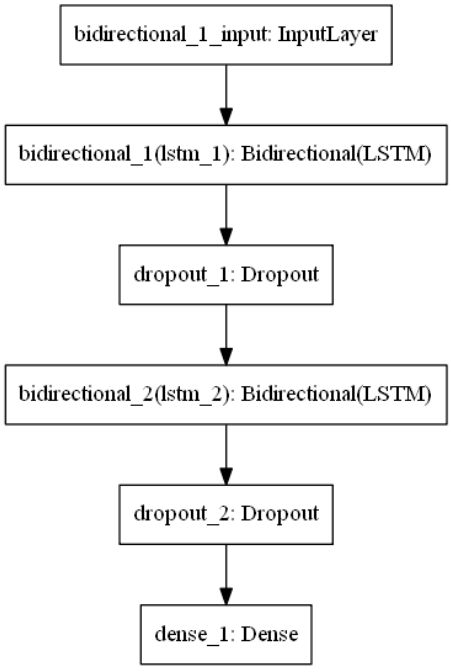

Figure 2: Prediction model

2.6 Related works

There are multiple softwares and websites that predicts signal peptides, such as SignalP, SignalBlast and Predisi to name a few [7] [8] [9]. Neural Networks (NN) are the most used methods while some sites use Hidden Markov Models (HMM) instead. Similar methods can used to predict other biological problems, such as motifs and trans-membrane regions in proteins.

3 Evaluating model

3.1 Model performance

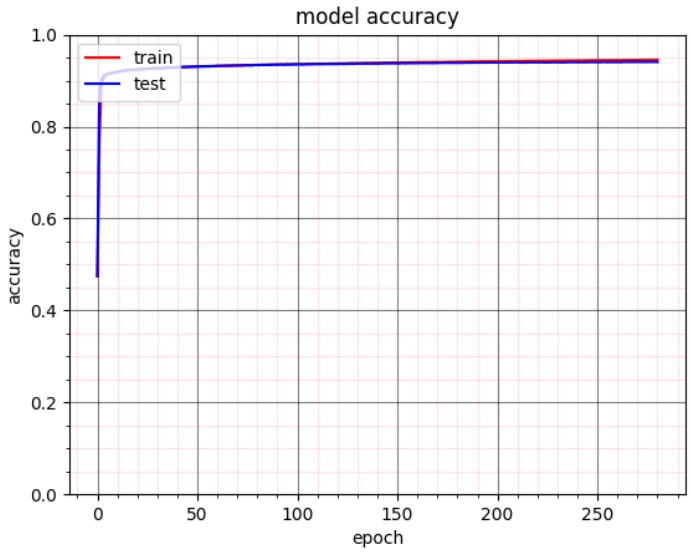

Figure 3: Model accuracy during training

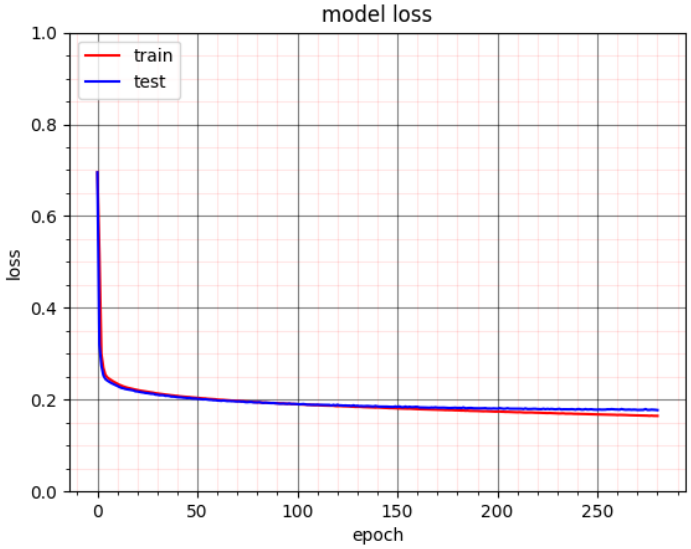

Figure 4: Model loss during training

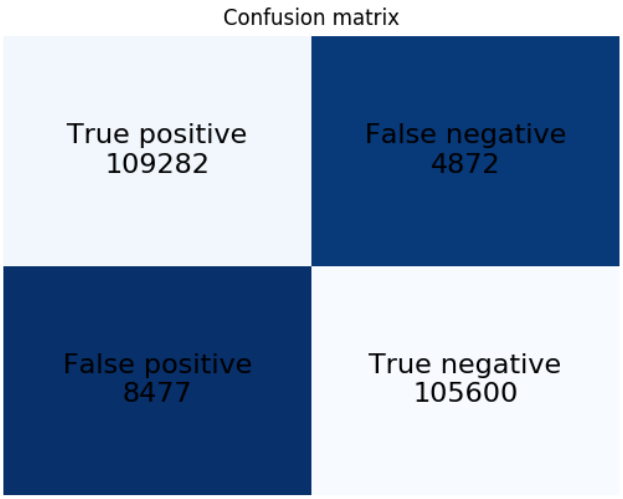

Figure 5: Confusion matrix on test set after training

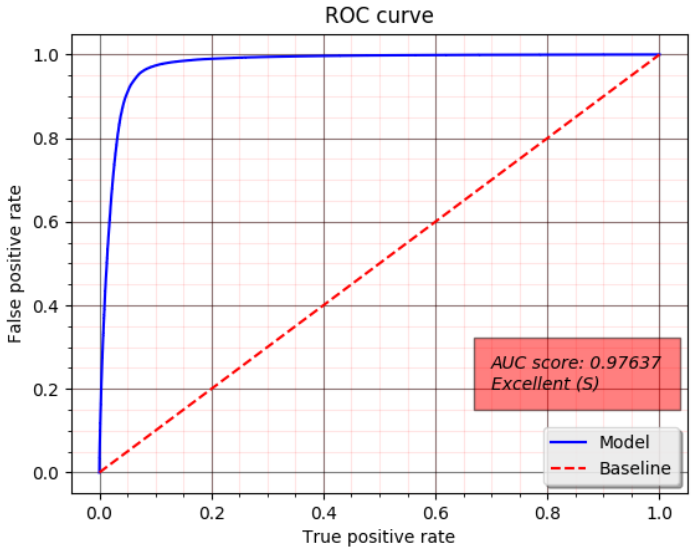

Figure 6: ROC curve on test set after training

3.2 Signal peptide detection in transmembrane proteins

Evaluate transmembrane and non transmembrane acc/loss. CM for TM and non-TM

3.3 Signal peptide detection in proteomes

Test two organisms, correct for CM

3.4 Generative model

A generative model was created that can create SPs. See in appendix 5.4.

4 Conclusion

Is it good enough? Can the model be improved? Can you use other methods (ex HMM, CNN, grouping etc...)

References

[1] Henrik Nielsen http://www.cbs.dtu.dk/services/SignalP-1.1/sp_lengths.html

[2] Signal peptides wikipedia https://en.wikipedia.org/wiki/Signal_peptide

[3] Swissprot database http://www.uniprot.org/uniprot/?query=reviewed:yes

[4] ??? http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[5] ??? https://web.stanford.edu/class/cs224n/reports/2760320.pdf

[6] ??? http://www.cs.cmu.edu/afs/cs/user/zhouyu/www/ASRU.pdf

[7] Henrik Nielsen, *Predicting Secretory Proteins with SignalP*, In Kihara, D (ed): Protein Function Prediction (Methods in Molecular Biology vol. 1611), pp. 59-73, Springer 2017, doi: 10.1007/978-1-4939-7015-5_6, PMID: 28451972, http://www.cbs.dtu.dk/services/SignalP/

[8] Karl Frank; Manfred J. Sippl, *High Performance Signal Peptide Prediction Based on Sequence Alignment Techniques*, Bioinformatics, 24, pp. 2172-2176 (2008), http://sigpep.services.came.sbg.ac.at/signalblast.html

[9] Karsten Hiller, *PrediSi*, Institute for Microbiology Technical University of Braunschweig, http://www.predisi.de/

5 Appendix

5.1 Model details

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_1 (Bidirection (None, 30, 128)           44032
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 128)           0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128)               98816
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 142,977
Trainable params: 142,977
Non-trainable params: 0
_________________________________________________________________
Input shape: (None, 30, 21)
Output shape: (None, 1)
```

12.3 hours trained with 729 710 samples on a GTX 970. Batch size of 4096, dropout of 0.5 for each layer and using ADAM optimizer (default learning rate, 0.0001)

More stuff

5.2 Pitfalls when training a model

Concept of under and overfitting and how to prevent it. Example pictures of bad models.

Overfitting ...
other stuff too
more stuff

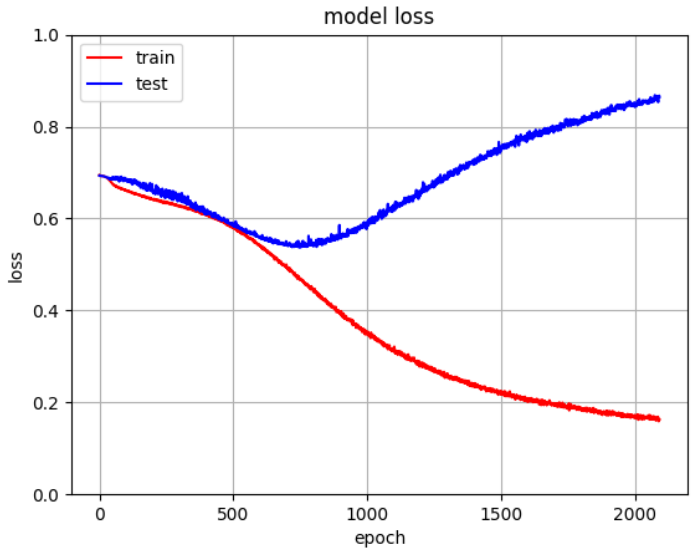

Figure 7: Overfitted

Too high learning rate ...
other stuff too
more stuff

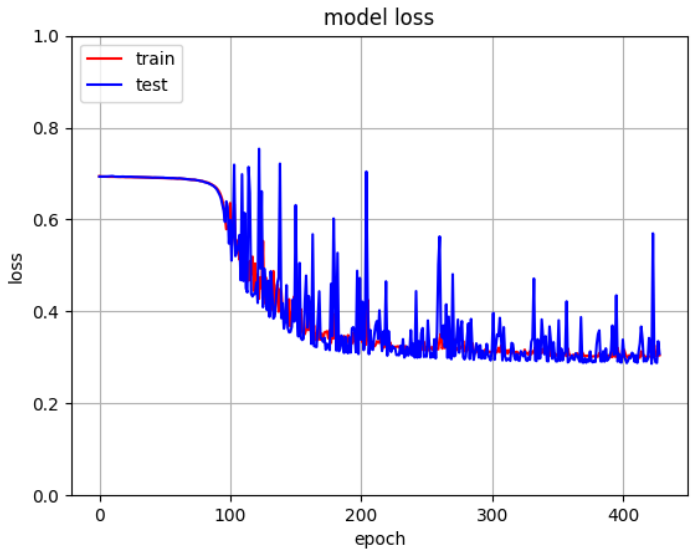

Figure 8: Too high learning rate

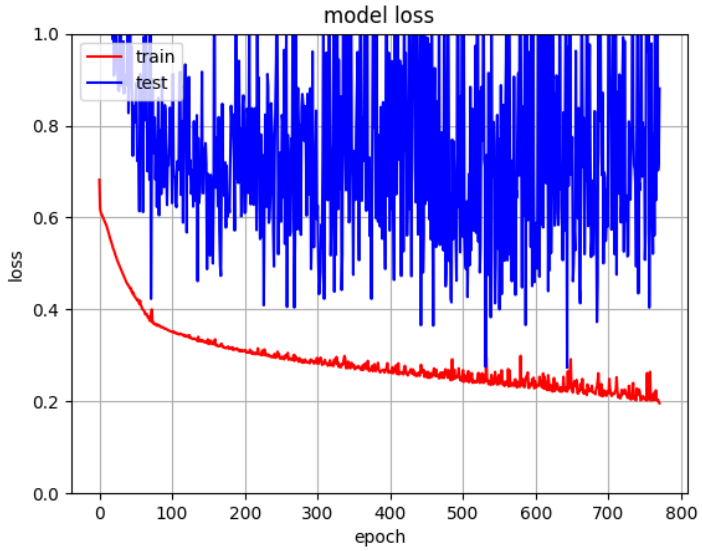

Figure 9: Data not shuffled

5.3 Additional features

List of things

1. Data augmentation,
2. Data conversion

- Things to add,
- Other things

The data can be loaded in many ways. One way is to augment the data by dividing each sequence into many chunks, labeling the chunk closest to the N-terminal as positive and the rest as negative.

Due to the large amount of data obtained, data augmentation were not used.

Advantages with data augmentation:

- Each positive sample would yield multiple negative samples, so smaller data sets can be used for training.
- Using positive samples and the negative samples from different organisms could make the model overfit to a particular amino acid bias. Using other regions of the positive samples would preserve the amino acid bias, making the data more consistent.
- Some trans-membrane regions of the proteins might be very similar to signal peptide regions, which would force the model to improve the prediction of trans-membrane proteins.

Disadvantages with data augmentation:

- The first amino acid in the N-terminal has an high bias towards some amino acids, mainly Methionine, see Figure 1. The model would then have a high affinity for the first amino acids, creating a poor model that would only detect if the sequence is located in the N-terminal or not. The first amino acids of each sequence were removed, preventing the problem if data augmentation were to be used.
- The N-terminal of peptides both with and without SPs might have a similar structure compared to other regions of protein, which would make the model as stated above, only be able to distinguish if the region is at the N-terminal or not. To prevent this problem, using both negative samples from N-regions and other regions should be used.

5.4 Generative Model