Combining Word Embeddings of Protein Sequences with Evolutionary Information for Secondary Structure Prediction Using Deep Neural networks
Bachelor's thesis in Informatics WS18/19

Anna Reithmeir

November 26, 2018

Contents

Abstract

Abstract goes here

DSSP states		
DSSP-3	DSSP-8	Description
E	E	β sheet
	B	β bridge
H	H	α helix
	I	π helix
	G	3_{10} helix
C	S	bend
	T	turn
	-	other

Table 1: Table of DSSP-3 and DSSP-8 states

1 Introduction

1.1 Protein Secondary Structure Prediction

Today the number of known protein sequences outnumber the amount of known protein secondary and tertiary structures due to the so-called sequence-structure-gap. While the secondary struc- ture implies the tertiary structure, this itself determines the function of the protein. It is important to know the functionality of a protein in order to e.g. create new medicine. Even though the secondary structure can be inferred from measurements od the 3d structure, the measurements themselves are time- and cost-consuming and thus scientists are searching for methods to predict the secondary structure. One way of doing so is with the help of machine learning techniques. The secondary and ter- tiary structure is determined by the sequence of the single amino acids (Anfinson experiment). Various methods have been found which hold promising results in protein secondary structure prediction, such as [still need to look up related work]. In my the- sis I will explore the application of convolutional neural networks, as well as LSTM networks on amino acid sequences. Importantly I will encorporate evolutionary information into the input data and use ProtVec on single residue basis for representing the input. I will not only predict the secondary structure but also the flexibility and solvent accessibility for each residue position. Thus I will make a multi-task prediction. Table 1 shows the DSSP states for 3 and 8 classes of secondary structures.

1.2 ProtVec

In 2015, [Asgari and Mafrad from the Lawrence Berkeley National Lab have adapted a method called Word2vec which is usually used by the NLP community to represent words by vectors in such a way that the learnt vectors describe amino acids in a protein. The vector embeddings were learnt by training on xyz protein sequences from the SwissProtDataBase] a representation of protein sequences, called ProtVec, which has been learned from an unsupervised neural network trained on 546 790 protein se- quences from the Swiss-Prot databank.

The basic idea behind their work is to use the so called Skip-Gram model introduced by Mikolov et al, taken from the field of NLP, and apply it on biological sequences like protein sequences. In- stead of using the Skip-gram model directly for feature extraction like in NLP, the authors use it to find a distributed representation for biological sequences, including protein sequences. The model can be learned only once and the resulting represenation can then be used for a variety of problems. In their pa- per the authors use 3-grams to represent the input sequences through summing over the ProtVec representation of the whole sequence. Furthermore Word2Vec negative sampling is used in the training process. As a result, each 3-gram of the input is represented as a dense vector of the length 100 in the paper. The au- thors also have shown in their paper that the representation they found actually holds information on biophysical and -chemical properties. The model achieves an accuracy of 93% in family classification, which is more accurate than pre- viously introduced methods while only learning from the sequence itself. Like the authors propose I will use the ProtVec representation as a pre-trained rep- resentation for the input of the neural networks I will work on, but instead of summing over the representation vectors I will use ProtVec on the basis of each single residue in the sequence. I will do this by considering the left and right neighbor of one residue as its' 'context' and represent each 3-gram of the sequence as a vector of length 100, given by ProtVec.

1.3 Evolutionary Information

2 Related Work

2.1 Secondary Structure Prediction with Deep Neural Networks

2.2 Multi-target Prediction And Evolutionary Information

3 Combining Word Embeddings of Protein Sequences with Evolutionary Information for Secondary Structure Prediction Using Deep Neural networks

3.1 The Dataset

The dataset I use consists of 3270 protein sequences and their according atomic coordinates taken from PDB with an average length of 235 amino acids, the longest one being 968 amino acids long. I will only consider proteins with a resolution higher than 2.5 Å. The DSSP algorithm by Kabsch and Sander is used to identify the secondary structure information corresponding to each of the residues in the sequence. I have 3-state and 8-state structure classification. The 8-state DSSP structures are G(3_{10} helix), H(α helix), I(π helix) , B(β bridge), E(β sheet) , T (turn), S(bend) and - (if no rule applies). These 8 classes can also be grouped into 3 larger classes: C (S, T, -), H (H, G, I) and E (E, B) which correspond to the states in 3-state DSSP classification. I will start out with the 3-state classification and then also use the 8-state DSSP structures to hopefully make accurate predictions in a more detailed way.

3.2 Data Preprocessing and Input Features

3.3 Neural Network Architecture

3.3.1 CNN

3.3.2 LSTM

4 Results

What I have done first was the data analysis and pre-processing which is a crucial step in any machine learning pipeline. Luckily Michael has already provided the raw amino acid sequences and their corresponding structure in- formation parsed from the DSSP output for both classification types. He also introduced two new classes, Y and X, where the first one describes structures which could not be measured in DSSP and Y describes the residues for which the different sequence parts did not output the same structure but different ones. The X class does not hold any information for my network and I will mask its' occurences out by saving the indices of where a Y occures in the structure sequence, creating a vector with zeros at these indices and ones elsewhere and then multiply the non-reduced loss of my neural networks by this vector. Tech- nically I am ignoring any misprediction at these indices. While proceeding in the same matter at first for the Y class too, I might take a closer look later on at the probabilities of the structure classes at that position and then, instead of multiplying with zero at this index, multiply the loss with the probability of the classes. However, the amount of X and Y occurences is rather small so I do not expect that to have a big effect on the performance of my networks. To get familiar with the data I computed the distribution of the class occurences and also the chain length of the different classes. In the 3-state classification, the C-class has the highest average chain length with 9.8 residues in a row, while in the 8-state classification class H has an average of 11.1 residues following in a row. To represent the data appropriately I first encoded each residue sequences as a 1-hot matrix of size sequencelength $\times$ 20 and the targets as a 1-hot ma- trix of size sequencelength $\times$ 4 (or 9), the 4 classes being C,H,E and X/Y (or H,E,I,S,T,G,B,-,X/Y). As soon as I make a basic CNN pipeline work for that, the 1-hot representation will give way for a representation through ProtVec, i.e. I will encode each input sequence as a dense sequencelength $\times$ 100 matrix and leave the targets like before. I then have to think about if i keep the fourth class for X and Y or if I will not. I have created a validation set consisting of 20into a training set

(80information and encorporate this into the input of the network. An idea is to average over the aligned sequences, but I will have to think further on how to use this information in the most effective way. After that I will concentrate on extending the predictions to the solvent accessibility and flexibility. Once I programmed this and found an architecture and hyperparameters which predict well, I will look as LSTM networks, which have been developed by Hochreiter and Schmidhuber. They consist of multiple modules of RNNs and allow learning information over arbitrary time intervals and being regulated by its' input gate, output gate and forget gate. LSTMs have found application mainly in NLP and are able to capture context information in text also over big intervals which seems to be useful for secondray structure prediction, because a residue at location i could be near a residue at location i + 100 in the 3d structure and thus influence the tertiary structure and functionality.

4.1 Code overview

4.2 CNN results

4.3 LSTM results

4.4 DSSP3 and DSSP8 results

5 Discussion

5.1 Runtime And Memory

5.2 Problems

5.3 Improvement

6 Outlook

7 Notes

Today the number of known protein sequences outnumber the amount of known protein structures by far. Scientists try to detect the structures with the help of machine learning techniques and by assuming that the secondary and tertiary structure is determined by the sequence of amino acids. Various methods have been found which hold promising results in protein secondary structure prediction, such as. Neural networks have also proven to be helpful in finding the structures. In my thesis I will explore the application of convolutional neural networks, as well as LSTM networks on amino acid sequences and importantly I will encorporate evolutionary information into the input data and use ProtVec on the residue basis for representing the input. I will not only predict the secondary structure but also the flexibility and solvent accessibility by only working on the sequences themselves. Thus I will make a multi-task prediction.

In 2015, Asgari and Mafrad from the Lawrence Berkeley National Lab have published a representation of protein sequences, called ProtVec, which has been learned from an unsupervised neural network trained on 546 790 protein sequences from the Swiss-Prot databank. The basic idea behind their work is to use the so called Skip-Gram model introduced by Mikolov et al, taken from the field of NLP, and apply it on biological sequences like protein sequences. The Skip-Gram model learns a vector representation for predicting the surrounding words of one word in a document or sentence, encoding linguistic regularities and patterns while being extremely efficient. the objective function is

$$\frac{1}{T}\sum_{l=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{l+j}|w_t) \tag{1}$$

where c is the size of the training context and T the amount of words in the input data. Each word will be embedded into a vector of the length n, similar words will have similar vectors and the representation is found by looking at the context, i.e. the neighboring words. Astonishingly, it is possible to meaningfully combine words by elementwise addition of the resulting vetor representations. Instead of using the n-gram model directly for feature extraction like in NLP, the authors use it to find a distributed representation for biological sequences, including protein sequences. Thus, the model can be learned only once and the resulting represenation can then be used for a variety of problems. In their paper the authors use 3-grams to represent the input sequences through summing over the ProtVec representation of the whole sequence. Furthermore Word2Vec negative sampling is used in the training process. As a result, each 3-gram of the input is represented as a dense vector of the length 100 in the paper. The authors also have shown in their paper that the representation they found actually holds information on biophysical and -chemical properties. The model achieves an accuracy of 93% in family classification, which is more accurate than previously introduced methods while only learning from the sequence itself. Like the authors propose I will use the ProtVec representation as a pre-trained representation for the input of the neural networks I will work on, but instead of making use of the summation I will use ProtVec on the basis of one residue. In a given amino acid sequence, the n-gram model considers sub-sequences of the length l as one word and its' neighboring words are considered as the context, following the assumtion that the secodary structure of a protein relies on the order of the acids in the sequence.

The dataset I use consists of 3270 protein sequences and their according atomic coordinates taken from PDB with an average length of 235 amino acids, the longest one being 447 amino acids long. I will only consider proteins with a resolution higher than 2.5 Å. The DSSP algorithm by Kabsch and Sander is used to identify the secondary structure information corresponding to each of the residues in the sequence. I have 3-state and 8-state structure classification. The 8-state DSSP structures are $G(3_{10}helix)$, $H(\alpha helix)$, $I(\pi helix)$, $B(\beta bridge)$, $E(ladder)$, $T(turn)$, $S(bend)$ and - (if no rule applies). These 8 classes can also be grouped into 3 larger classes: C (S, T, -), H (H, G, I) and E (E, B) which correspond to the states in 3-state DSSP classification. I will start out with the 3-state classification and then also use the 8-state DSSP structures to hopefully increase the accuracy of my predictions.

What I have done first was the data analysis and pre-processing which is a crucial step in any machine learning pipeline. Luckily Michael has already provided the raw amino acid sequences and their corresponding structure information parsed from the DSSP output for both classification types. He also introduced two new classes, Y and X, where the first one describes structures which could not be measured in DSSP and Y describes the residues for which the different sequence parts did not output the same structure but different ones. The X class does not hold any information for my network and I will mask its' occurences out by saving the indices of where a Y occures in the structure sequence, creating a vector with zeros at these indices and ones elsewhere and then multiply the non-reduced loss of my neural networks by this vector. Technically I am ignoring any misprediction at these indices. While proceeding in the same matter at first for the Y class too, I might take a closer look later on at the probabilities of the structure classes at that position and then, instead of multiplying with zero at this index, multiply the loss with the probability of the classes. However, the amount of X and Y occurences is rather small so I do not expect that to have a big effect on the performance of my networks. To get familiar with the data I computed the distribution of the class occurences and also the chain length of the different classes. In the 3-state classification, the C-class has the highest average chain length with 9.8 residues in a row, while in the 8-state classification class H has an average of 11.1 residues following in a row. To represent the data appropriately I first encoded each residue sequences as a 1-hot matrix of size $sequencelength \times 20$ and the targets as a 1-hot matrix of size $sequencelength \times 4$ (or 9), the 4 classes being C,H,E and X/Y (or H,E,I,S,T,G,B,-,X/Y). In the following weeks the 1-hot representation willgive way for a representation through ProtVec, i.e. I will encode each input sequence as a dense $sequencelength \times 100$ matrix and leave the targets like before. I then have to think about if i keep the fourth class for X and Y or if I will not. I have created a validation set consisting of 20% of the samples, for training and testing I will split the remaining smaples into a training set (80%) and a test set (20%).

As a first attempt I use a CNN architecture and the 3-state classification, represented as 1-hot matrices. CNNs are mainly used on image data. As soon as I have the pipeline working, I will encorporate the ProtVec representation and then include the evolutionary information, which I have from Michael. It shows the sequnce aligned related protein sequences and I will still have to find a way of how to use this information. Michaels idea is to average over all related sequences. In the future I want to explore the use of so-called LSTM networks, which have been developed by Hochreiter and Schmidhuber. They consist of multiple modules of RNNs and allow learning information over arbitrary time intervals and being regulated by its' input gate, output gate and forget gate. LSTMs have found application mainly in NLP and are able to capture context information in text.

Then I will compare how these two architectures perform, play around with their hyper parameters like the dropout rate, number of layers, activation functions (e.g. SELU and RELU) and so on.

7.1 Overview of files

matrix_1hot_3_val.npy : validation set 3-state, 1-hot

targets_3_val.npy : structure information encoded as memmap 3-state

targets_8_val.npy : structure information encoded as memmap 8-state

matrix_1hot_3_train.npy : remaining samples 3-state, 1-hot for active use

targets_3_train.npy : corresponding targets as memmaps 3-state

targets_8_train.npy : corresponding targets as memmaps 8-state

7.2 More Notes

Proteins secondary structure can be divided into 3 or 8 classes - DSSP classification. 'turn' and 'bridge', repeating turns are 'helices', repeating bridges are 'ladders'. Connected ladders are 'sheets'. There is a relation of the amino acid sequence to the secondary structure of course. H=H(4-turn helix)G(3.turn helix)I(5-turn helix), E=E(ladder)B(bridge), C=S(bend)T(turn)C(coil)

PDB: Protein Data Bank, lists of atomic coordinates located through X-ray cristallography, data is sometimes incomplete and subjective.

non standard proteins X and O occur 8 and 1 times. Just ignored them and encoded them as zero vector.

UniProt: Universal Protein Resource. UniParc database contains most of the known protein sequences

Å (ångstrom): unit of length, 0.1 nanometre. Used to express sizes of atoms and molecules. Resolution of the protein acid models, 1 Å is high resolution, 3Å low.

The data I am working on: sequences from UniProt, corresponding structures taken from PDB where resolution higher than 2.5 Å.

What I will do: Encode each protein sequence as a 20xlength one-hot matrix where each position indicates which protein can be found here. Look at the neighbors of each protein and feed each position including it's neighbors - like 'QWP' - to an NN - that would be a 20x3 matrix.The NN should classify the input into 3 or 8 structure classes. But instead we want to use ProtVec and thus have an input size of 3x100 for each 3-sequence. The targets are 3-sequences of structures like 'HHH'. After that I will include the proteon evolution into the input data, by sequence alignment methods. Rost idea: first consider evolution, then combine it with ProtVec later?

ProtVec: Table where each possible 3-sequence corresponds to a vector of length 100. Trained on 546 790 smaples of the Swiss-Prot database to encode biophysical and biochemical properties. NLP used. Each word represented as n-dim Vector, close words in terms of semantics and syntax should have close vectors. Skip-gram modelProtein family classification by using neighboring words as context, specifically an overlapping sliding window of 3 amino acids are seen as one word are used to train an NN on maximizing over the probability of observed word sentences. As an output each 3-gram can be identified by an 100 vector. These vectors are then projected onto two dimensions through Stochastic Neighbor embedding. Properties tested were: mass, volume, polarity, hydrophobicity, charge, Van-der-Waals volume. Eventhough the model is trained on sequences only, the authors obtain an weighted accuracy of 93+-0.06% which is better than the existing methods for protein family classification. It can distinguish between various classes of sequences and is a dense representation of the input protein sequences. Was able to identify disordered sequences with nearly 100% accuracy. We will consider these vectors as 'pre-training data' from our input sequences. Skip-gram model: Used to learn high-quality vector representaion of unstructured words. NLP model. The resulting vectors can then be used for simple vector calculations which yield the expected results.

Skip-gram model:

The NN: CNN and/or LSTM intuitively. More than one hidden layer because some 3-sequences like 'HEH' cannot be and the hidden layer then pools the sequence like it would have been 'HHH'. Each sequence end cannot be looked at because one neighbor missing. Thus insert a 0-vector there or explore what else could be done here.

Data statistics: number of class occurences and locations, avg length, length of distribution.

Overall structure:

section 1	Introduction: secondary structure, multi target prediction, NLP, NN, My work
section 2	Related Work on structure prediction with ML using evolutionary information
section 3	The dataset and plots (maybe also classification of biophysical properties?)
section 4	Incorporating evolution and ProtVec
section 5	
section 6	My network and code overview
section 7	Outcome, accuracy, plots of input and prediction data
section 8	Discussion and problems
section 9	Outlook
section 10	References

7.3 The dataset

The data set I will be using for training is taken from the PDB - the RCSB Protein Data Bank [1] which holds more than 1 TB of structure data of proteins, DNA and RNA and is considered the leading global resource for protein data in research. The secondary structure is determined through X-ray cristallography which is expensive and sometimes incomplete. It is described through a list of 3d coordinates. To have accurate input data, I will only consider proteins of PDB which are presented with a resolution better than 2.5Å - 3Å is already considered a high resolution. On the one hand I will use the amino acid sequences as an input. On the other hand I will use sequences of secondary structures, each one corresponding to one amino acid in the input, as the targets. To obtain these the DSSP algorithm was used, which was developed in 1983 by Kabsch and Sander. This algorithm takes raw 3d coordinates of the atoms of a protein as an input and classifies them either into 3 or 8 classes of structures. I will use both classification types. Besides the structure classes C, H and E or H, E, I, S, T, G, B, - there are also the two classes X and Y which Michael has inserted while parsing the DSSP files. While X is a placeholder for amino acids where the structure could not be measured (that is, in the DSSP file there are locations missing), Y is a placeholder for a structure which the algorithm could not identify with certainty (that is, the structure prediction differ in the different parts of the sequence). Thus, the class X is not important for us and will be masked out in my work. The class Y still has predicted structure information, eventhough not a clear one. To make this information useful I will count the MLE of the structure at the Y location and insert the probability of that structure being at this location into the target vector, instead of masking it out. In total I have 3313 protein sequences and their according structures with an average length of 235 amino acids, the longest one being 447 amino acids long.

First I will extract some statistical data of the dataset like the number of occurences of each structure and the average length of a chain of one structure. Then I will encode the available data into a 1-hot matrix of the size 20xsequnce length and feed it to different nns. Then I will encode the amino acid sequences with the use of ProtVec into a matrix of size 100xsequence length and again feed it to the saem networks, so I will have a direct comparison.

7.4 Protein evolution and ProtVec

The key feature of my work is that I will incorporate the envolution of the protein sequences into the input data.

I will use ProtVec [**?**]. ProtVec protein vectors represent one protein sequence by a n-dimensional vector, here a vector of length 100, which encodes information on different biochemical and biophysical features. ProtVec was determined by learning represenations in an unsupervised fashion on 546 790 sequencesand adapting methods from existing NLP (Natural Language Processing) methods. Each sequence is split into 'words' of three residues that is into one residue and it'scontext of left and right neighbors. This method is adapted from the Skip-gram model but instead of using n-grams directly for feature extraction they used it for a distributed representation of sequences, attempting to maximize the probability of observed word sequences resulting in a list of 3-grams and their representation as a 100x1 vector. It shows a weighted average accuracy of 93 +- 0.06% in protein family classification where each sequence is represented as the summation of the representation vectors for each overlapping 3-gram. By down projection of the representation vectors onto a 2 dimensional space via stochastic neighbor embedding the authors show that the vector hold significant information about the distribution of properties like mass, volume, hydrophobicity or charge.ProtVec only needs to be trained once and the resulkting vector representations can then be used as a pre-trained input for mahcine learning tasks, which is what I will do in the following.

7.5 My work

I looked at the data and analysed it. Then I split the data into validation set and training set (20/80 percent) with same ratio of structure classes (classification mode=3). Created 1-hot matrix.

First attempt: Simple CNN in 2 dimensions with 1-hot mtrix as input. The average distance between the beta sheets determine the kernel size because we want to have two sheets in one frame on average. The CNN uses log-softmax and NLL-loss. Padding is done in a way that all the samples are being zero-padded at the end up to the maximum length occuring in the dataset. So the dimensions are: Input samüple: [batch size, 21, 1, 447], targets: [batch size, 1, 447], output of the network: [batch size, 4, 1, 447] which I will transform to [batch size, 1, 447] by taking the maximum of the probabilities in each feature row. X and Y are not dssp classes but Michael has inserted them whenever the structure information was missing(X) or when there were different structures at the same place in the A/B/C... sequences (Y). In the first attempt I will treat X and Y as a separate class and mask all these locations out. This is done by saving the indexes of the locations and then multiplying the loss, which is not being reduced in my code, with the mask so that the loss at the specified locations will turn to zero and will not be relevant. Also there are some outlier AAs like O and X as residues which I will also mask out(todo!)

In the second attempt I will classify X into one class and Y into another and then measure the probability of what structure is at one location where Y is and insert a probability vector at this location instead of a null-vector, while the X locations are still being masked out. I have to parse the dssp file for that to get the MLE(still to do!).

8 TODOs

- Input vector normalization
- Average distance between beta sheets
- reliability and impossible combinations. filter?
- encorporate parsed percentages from dssp files and put that in input vector
- vectorgraphics of the final CNN and LSTM architecture

DSSP-3 states	
state	DSSP-8 state
E	E, B
H	H, G, I
C	S, T, -

DSSP states		
DSSP-3	DSSP-8	Description
E	E	β sheet
	B	β bridge
H	H	α helix
	I	π helix
	G	3_{10} helix
C	S	bend
	T	turn
	-	other

References

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000. [Online available at `www.rcsb.org`; accessed 23 Oct 2018].

[2] E. Asgari and M. R. Mofrad, "Protvec: A continuous distributed representation of biological sequences for deep proteomics and genomics," *PLoS ONE*, vol. 10, no. 11, 2015.

Appendices

A Appendix A

Appendix A

B Appendix B

Appendix B