Short summary (notes)

Anna Reithmeir - Bachelorarbeit WS18/19

November 22, 2018

Today the number of known protein sequences outnumber the amount of known protein secondary and tertiary structures. While the secondary structure determines 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. Eventhough measuring the secondary structure is possible, it is time- and cost-consuming and thus scientists arelooking for methods to predict the secondary structure. One way of doing so is with the help of machine learning techniques and by the assumtion that the secondary and tertiary structure is determined by the sequence of single amino acids. 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 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 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.

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. Instead 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. 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 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.

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 $G(3_{10}helix)$, $H(\alpha helix)$, $I(\pi helix)$, $B(\beta bridge)$, $E(ladder)$, $T(turn)$, $S(bend)$ 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 make accurate predictions in a more detailed way.

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). 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 20% of the samples, for training and testing I will split the remaining smaples into a training set (80%) and a test set (20%). After that I will use evolutional information 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.