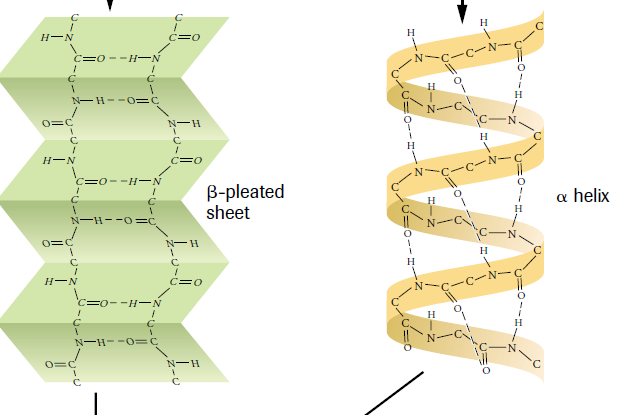
# Predicting Secondary Structure of Proteins

# from Amino Acid Sequences

Proteins are the main building blocks of the living cell and they drive almost all molecular processes inside it. The specific function of each protein is determined by its three-dimensional shape. Therefore, one of the main challenges in bioinformatics is the prediction of the protein structures. Besides providing fundamental knowledge of the functioning of the cell mechanisms, the structure prediction has important applications in the pharmaceutical industry.

In the structure of the proteins three different levels can be distinguished. The proteins are made of smaller molecules called amino acids, which are 20 in number. Each protein consists on average of around 100 amino acids which are arranged in a sequence. This sequence of amino acids is encoded in the DNA and represents the *primary structure* of the protein.

Under the influence of the environment this linear structure is folded into a three-dimensional form which determines the physico-chemical properties of the protein. This spatial structure is also called a *tertiary structure*. There is also a *secondary structure*, which distinguishes certain standard subunits of the proteins. There are two main subunit types: *alpha-helices*, a spiral-like chains, and *beta-sheets*, flat plain structures, as shown in the figure below.



The goal of this assignment is to predict the secondary structure corresponding to a given primary structure sequence.

We will do this in three different ways using different approaches.

We use protein domains (subunits) from the CATH database (https://www.cathdb.info/).

A pre-processed version of the database is available from:

<https://raw.githubusercontent.com/dbosnacki/HelisDeepLearningCourse/main/cath-domain-description-file-v2_4ProcessedForNN.tsv>

for each protein domain its name, class, label (encoding of the class) and amino acid sequence.

# Exercise 1

In this exercise we represent the sequence based on the relative frequencies of occurrence of each pair of neighbouring amino acids in it. To obtain the relative frequency the number of occurrences of each pair is divided by the length of the sequence. Since there are 20x20=400 possible pairs the input vector consists of 400 numbers. For example, in the sequence

KRCLXRCGCR

the pair occurrences are as follows:

* KR, CL, LX, XR, CG, GC, CR occur once,
* RC has two occurrences,
* all other pairs have zero occurrences.

Since the length of the sequence is 10 these frequencies translate into relative frequencies of 0.1, 0.2, and 0.0 respectively.

Translating the sequences into frequency vectors (arrays) can be considered as a feature extraction. The frequency of each amino acid pair is a separate feature of the sequence. In this way we “help” our model in the classification.

Hence, the task is to design a deep neural network which takes such a frequency vector as an input and classifies the corresponding sequence in one of the four secondary structure classes called *mainly-alpha*, *mainly-beta*, *(mixed) alpha-beta* and *irregular*.

# Exercise 2

In this exercise we do not try to “help” the neural network in the classification by selecting relevant features. Instead, we want to predict the class based directly on the amino acid sequence.

Before feeding them into the network the sequences need to be pre-processed. We first extend (pad) the sequences with spaces in order to obtain the same length for all inputs. The length of the input is equal to the length of the longest sequence After that the padded sequences are encoded using one-hot encoding. In one-hot encoding each letter corresponding to an amino acid as well as the space character used for padding, are encoded with a binary vector of 21 elements having 1 in only one position.

# Exercise 3

This exercise is similar to the previous one in the sense that we use directly the sequences which are pre-processed in the same or similar way, i.e., with padding and one-hot encoding. Only we use a Recurrent Neural Network instead of the standard deep neural network.

# General remarks

Since the main goal of the exercises is not to learn python programming, please feel free to ask the instructors for solutions of some of the auxiliary tasks, like finding maximal sequence length or padding strings with spaces. Alternatively, please feel free to look in the provided solutions.

To improve the performance of Google Colab, in particular when using Recurrent Neural Networks, you can set the corresponding menu option

Runtime -> Change runtime type->Hardware accelerator

by choosing GPU.

You can write your programs from scratch or use the Google Colab notebooks in which you have parts of the exercises worked out:

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/Exercise1.ipynb>

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/Exercise2.ipynb>

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/Exercise3.ipynb>

The solutions of the exercises are also available:

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/ExerciseModelTrainTestProteinDomainsWithFrequencyPairs.ipynb>

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/ExerciseModelTrainTestProteinDomains.ipynb>

<https://github.com/dbosnacki/HelisDeepLearningCourse/blob/main/exercises/ExerciseModelTrainTestProteinDomainsRNN.ipynb>

Other useful links are to the Python library Tensorflow tutorial

<https://www.tensorflow.org/tutorials/quickstart/beginner>

and the Python API doe Deep Learning:

<https://keras.io/>

Good luck!