Term Project, part I

Fall 2020

Introduction

In this project you will implement an Artificial Neural Network (ANN) to predict protein secondary structures, based on the work by Holley, L.H. and Karplus, M., 1989. Protein secondary structure prediction with a neural network. Proceedings of the National Academy of Sciences, 86(1), pp.152-156. The full article is available here: https://www.pnas.org/content/pnas/86/1/152.full.pdf

The subject is also covered by Arthur Lesk in the textbook, pages 246–249.

Artificial Neural Networks (ANN)

See class notes for an intro. It is very basic and if you have never worked with ANN before you may need more materials. You can (and probably should!) do the following tutorials:

- https://people.revoledu.com/kardi/tutorial/Python/Practice+Neural+Network+in+Python.html (very basic)
- https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/
- https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/

The algorithm

ANN background is discussed in class and detailed in the class notes. Details are given in the paper.

Here are the specs:

1. Every amino acid should be converted into a binary vector of size 21, with zeros everywhere except the amino acid's index. 21 stand for 20 types of amino acids + a null category when a window overlaps with a terminus.
2. A window of size 17 is defined around each amino acid, containing the amino acid ± 8 on each side.
3. The network has three layers – input, hidden and output.

Figure 1 shows the example from the paper. Steps:

- The input layer is a sliding window on the amino acid sequence. The window size is 17, where the prediction is made for the central residue in the window.
- Each amino acid at each window position is encoded by a group of 21 inputs, one for each possible amino acid type and one to provide a null input when the window overlaps with the N- or C- terminus. In each group of 21 inputs, the input corresponding to the amino acid type at that window position is set to 1 and all other inputs are set to 0.
- Thus, the input layer consists of 17 groups of 21 inputs each, and for any given 17 amino acid window, 17 network inputs are set to 1 and the rest are set to 0.

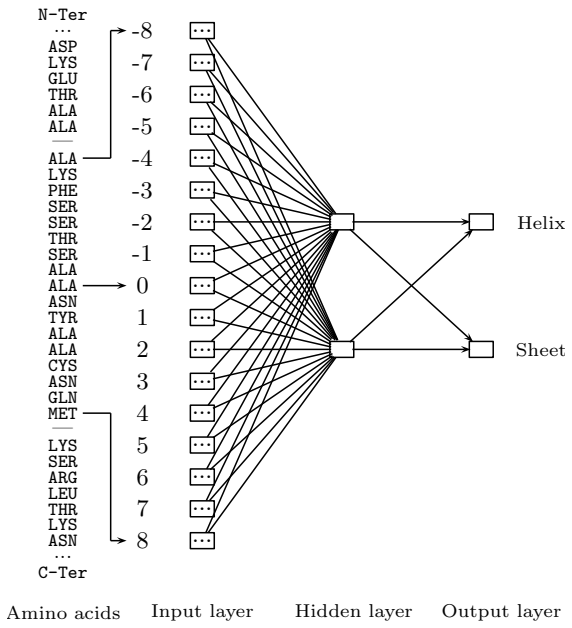

Figure 1: The architecture of a Neural Network for secondary structure prediction.

- The hidden layer consists of two units. The output layer also consists of two units. Secondary structure is encoded in these output units as follows: $(1, 0)$ = helix, $(0, 1)$ = sheet, and $(0, 0)$ = coil. Actual computed output values are in the range $0.0 - 1.0$ and are converted to predictions with the use of a threshold t.
- Initial weights are set to random numbers between -0.1 and 0.1 both for the edges from input to hidden, and from hidden to output.
- Train the network for as many cycles as you need to achieve convergence (see paper for more details).
- Helix is assigned to any group of four or more contiguous residues having helix output values greater than sheet outputs and greater than t. β-Strand is assigned to any group of two or more contiguous residues, having sheet output values greater than helix outputs and greater than t. Residues not assigned to helices or sheets are assigned to coil.
- The value of t is adjusted by maximizing the accuracy of secondary structure assignment for the training set.
- The accuracy can be estimated by constructing a confusion matrix, which gives the predicted vs. actual secondary structure elements:

	Actual		
predicted	helix	sheet	coil
helix			
sheet			
coil			

Where true positives are the diagonal elements.

You may use any software package you want (including implementing the whole thing by yourself from scratch although I would advise against it), but there are several tools that can help you:

- It is better to use a newer dataset than the one in the paper. If you use python (for which you will need `numpy` and `scipy` for handling matrices for example), `sklearn` can be used to construct the network.
- For secondary structure assignments you can use the `dsppkeras` dataset at https://github.com/PeptoneInc/dspp-keras. It contains amino acid sequence and a numeric value corresponding to its SS. Individual, residue-specific scores are bound between 1.0 and 3.0. A propensity of 1.0 Implicates the sampling of beta-sheet conformations. A score of 2.0 indicates behavior found in disordered proteins, whereas 3.0 is an indicator

of properly folded alpha-helix. A score of 2.5 should be understood as a situation when 50% of ensemble members form a helix and the remaining part samples different conformations.
- Usually, 70-80% of the data is used for training and the rest for testing.
- For training and testing parameters and specs, see original paper or modify as you wish.
- If you prefer R you can use the neuralnet package https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf.

Delivery

The project should be submitted by the end of the exam period at the latest – December 21, 2020. You should plan accordingly and start working as soon as possible. I will ask you to submit a bi-weekly report about your progress to keep you in check. The following is required:

1. The code.
2. A document (2-3 pages) specifying your choice of implementation, algorithm, specs - data set, training process, confusion matrix and results.

Rough (suggested) timeline

By mid-November you should have the following:

- Know how to create a simple neural network, train and test it (see examples above and understand them).
- Be able to convert the amino acids into categorical vectors of 0 and 1 – I suggest to define a vector of size 21, where every position is dedicated to one amino acid, say in alphabetical order of their one-letter code. So for example – if the amino acid is 'A' (Alanine), position 0 is 1, the other position 0. If the if the amino acid is 'C' (Cysteine) , position 1 is 1, the other position 0 etc. You can do it as an enum or any other hard coded definition.
- Following that – you should be able to map a window of 17 amino acids into a matrix of size 17×21 (if you use python, Numpy can help).

By early December you should have the following:

- Learn how to read protein data and convert them into a sliding window of matrices (As above). Don't forget the window of ± 8 residues before and after the protein, for which you have the 21st symbol.
- Feed the data into a neural network.
- Train the network and determine the weights.

By mid-December you should have the following:

- Determine parameters, test the network.
- Have results and verify accuracy.

The last week can be dedicated to writing the report and finalizing.