IV. Artificial Neural Network (ANN) to determine protein secondary structure

1. Why secondary structure -> important information about the protein, used in fold recognition and to improve accuracy of tertiary structure prediction
2. Artificial Neural Network -> background, structure, how they learn
3. ANN for secondary structure prediction

* Requirements: Multiclass classification problem
* Dataset
* Specifications:
  + input (amino acid representation),
  + output (hot one encoding of classes),
  + parameters (window size, number of hidden layers, number of neurons/hidden layer, number of epochs, batch size, eta)
  + testing strategy (jack knife vs. training + testing sets)
* Design: ANN structure, activation function, forward propagation, backpropagation, error computation
* Implementation: ANN class, important functions, use of matrix multiplications, UI
* Performance: performance measure (Q3, sensitivity, specificity), impact of parameters on performance
* Further improvements: larger dataset, include physico-chemical properties of amino acids, different amino acid encoding, regularization, different activation functions

window\_size = 15  
minibatch\_size = 200  
epochs = 5000

eta = 0.03

hidden = 15, 15

[ 0] Training error: 0.73794 Test error: 0.75107

[4999] Training error: 0.49934 Test error: 0.53214

another run:

[9999] Training error: 0.44875 Test error: 0.46071

1. Theory on ANNs

Neural networks [1] were originally conceived as computational models of the way in which the human brain works. Like the human brain, they consist of many units (analogous to neurons and sometimes called by the same name) connected to each other by variable strength links (analogous to axons in the brain). These variable strength links are abstract representations of the way that most neurons actually communicate with each other in the brain: through changes in the rate or frequency of electrical or chemical messages. As with a number of the techniques described in this book, this technique has been inspired by the way biological organisms (in particular humans) solve the problems of computation in nature. As mathematical models, they have found a large number of applications in science and commerce, particularly in the area of finance and market prediction. The attraction of neural networks is that they can ‘learn’ relationships between sets of variables taken from a system. Once trained, the network can then be shown new examples and asked to predict the outcome of the new data based on the previous examples it has learnt. This quality, known as generalization, is the ability to infer the underlying relationships in the data and being able to apply them to new situations and is the staple reason for their use in such a wide variety of contexts. This may sound similar to the method by which humans learn and, to a very limited extent, this is true. A further property which distinguishes this technique from other computational methods is that of ‘graceful degradation’. The knowledge learnt is encoded in the network as a set of ‘weights’, the individual strength of these weights determines the behavior of the network. Should any of these weights or units be removed, the network can still function but with reduced performance, a little like the human brain. This is in contrast to most other computational techniques which cannot function at all if one or more parts of their decision making process is faulty. Neural networks should not, however, be seen as constituting biologically significant models of human brain activity, although some studies are conducted into the simulation of human brain activity (under the umbrella of connectionism) for the purposes of this book, they are merely useful computational tools.

Therefore, as computational tools, neural networks represent somewhat of a departure from many of the other artificial intelligence techniques which have a more symbolic flavor. They have a step-by-step algorithm of operation, but the resulting neural structure has a little more in common with biology than other methods.

A neural network consists of interconnected units, often arranged in layers. The configuration of these units is known as the architecture, and can vary widely depending on the application for which it is used. In the simplest neural networks there are only two layers – one ‘input’ layer and one ‘output’ layer. These networks are only able to discriminate linear relationships between variables because they possess only one layer of weights. The more sophisticated ‘multi-layer’ network adds a number of ‘hidden’ layers of units and therefore the two sets of weights increase the power of the network to infer non-linear relationships between variables. There is no theoretical limit to the number of layers a network can possess, although these two are among the most popular.

1. Dataset

The dataset used contains a training set and a test set. The training set will be used to teach the neural network and the second one will be used to compute the performance of the application. In total, the dataset contains 106 proteins separated by a specific symbol, each protein having the sequence of amino acids and the corresponding secondary structure element to which each amino acid will belong: α-helix, β-sheet or coil. This set was used in an older study [2] and was constructed to contain an equal amount of the three elements in the training set, as well as in the test set.

1. Requirements

The application will read the input training set from file and will start with random weights. These weights will be adjusted in the teaching phase by analyzing the secondary structure element to which each amino acid belongs. After that, the testing set will be read from another file and every amino acid will be categorized in one of the three structural classes. In the end, the performance of the network will be assessed, knowing the actual category for each amino acid in every protein in the test set.

1. Specifications

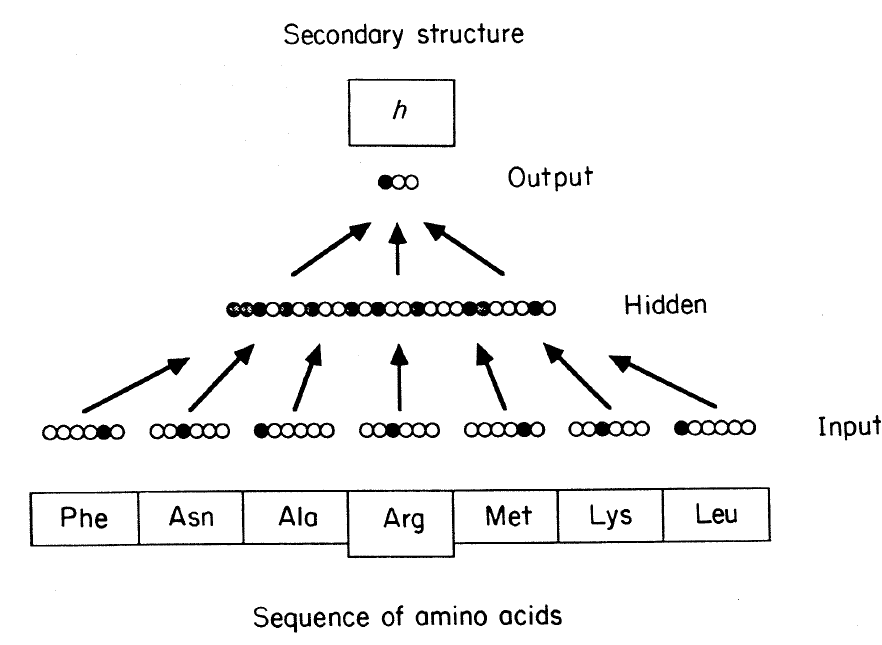
The input consists of 106 proteins, divided into two sets: training set, for which we know the secondary structure of each protein and the testing set, which contains only the amino acid sequence of the proteins. Each protein is delimited by the next using a specified symbol (“<>”) and each amino acid is written on a different line, with the corresponding category next to it, separated by a dash. The categories are: “h” for α-helix, “e” for β-sheet and “\_” for coil structure.

The output will represent the category of secondary structure element in which every amino acid falls into, according to the trained neural network. The categories are the same as for the training set.

Performance criteria + mathematical expr

1. Design

A representation of the structure of the neural network is shown in Figure IV.1, with the input neurons at the bottom of the picture and output neurons at the top. The units on the input layer have connections to the units on the intermediate, hidden layer, which in turn have connections to the units on the output layer. The network is given a contiguous sequence of 13 amino acids and the network will predict the secondary structure for the middle amino acid in the sequence.



**Figure IV.1**. The layers of the neural network for predicting protein secondary structure

The input layer is arranged in 13 groups, with each group having 21 units (20 amino acids and the spacer symbol). The output group has 3 units, each representing one of the possible secondary structures for the central amino acid. For a given input and set of weights, the output of the network will be a set of numbers from zero to one. The secondary structure chosen will be the output unit with the highest activity level.

[1] Keedwell, E., Narayanan, A.: Intelligent Bioinformatics: The application of artificial intelligence techniques to bioinformatics problems, John Wiley & Sons Ltd, Chichester, 2005.

[2] Qian, N., Sejnowski, T.J.: Predicting the secondary structure of globular proteins using neural network models, Journal of Molecular Biology, 202(1988), 865-884.