CS612 - Algorithms in Bioinformatics

Protein Structure Detection Methods

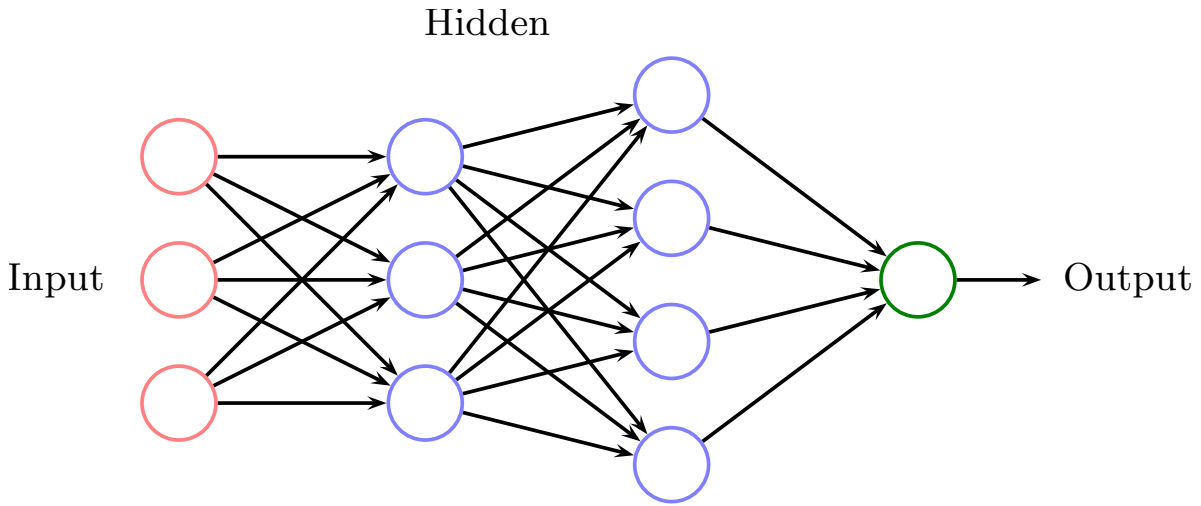
December 7, 2020

Secondary Structure Prediction

- Assignment of secondary structure is a typical annotation problem that can be addressed with various machine learning techniques
- HMMs can be used to annotate an amino-acid sequence with secondary structure information – HMMs are an example of generative models
- There are other methods that rely on neural networks, SVMs, and other machine learning techniques
- The current state of the art achieves accuracy rates of 70%-80%
- All approaches capture key amino-acid level signals present in alpha-helices and beta-strands
- Since coils, loops, turns do not have such well-defined signals, they are usually predicted as "other" and are more difficult to pin down

Artificial Neural Networks

- Artificial Neural Networks (ANN) are computational models inspired by the biological neural networks that constitute animal brains.
- They are used for classification problems and pattern recognition.
- The network is represented as a weighted directed graph. The graph has a layered structure:
- An input layer, one or more "hidden layers" and an output layer.

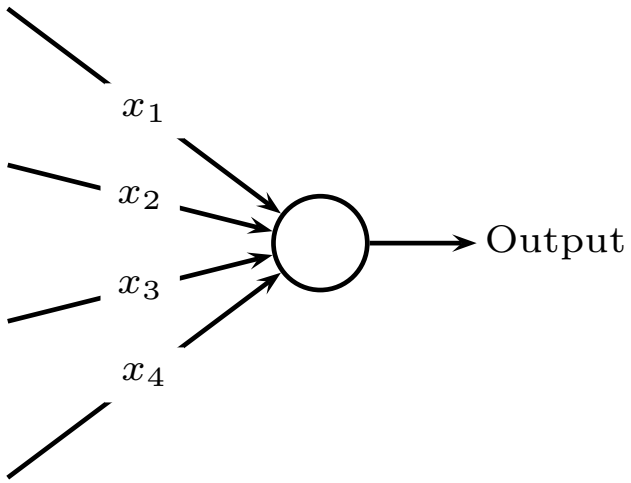


Artificial Neural Networks

- Every node represents a neuron and every edge represents a connection (synapse) between neurons.
- Different layers may perform different functions on their inputs.
- Signals travel from the input layer, to the last output layer, possibly after traversing the layers multiple times.
- The input layer receives input from the outside in the form of a vector. The output layer transmits output to the outside.
- The hidden layers are not connected directly to the outside, only to other layers.
- Each input is multiplied by its edge weight, representing the strength of the interconnection between neurons inside the network.

The Simplest Artificial Neural Network

- Let us first look at what is perhaps the simplest ANN, a *perceptron*.
- A perceptron takes binary inputs, $x_1 \ldots x_n$ and produces a single binary output.
- The inputs are weighted by real numbers: $w_1 \ldots w_n$, scaling the importance of each input to the output.
- Overall, the input is a weighted sum $\sum_{1}^{n} w_i x_i$.



A perceptron

- The output is either 0 or 1, depending on whether $\sum_{1}^{n} w_i x_i$ is below or above a given threshold. respectively:

$$Output = \begin{cases} 0, & \text{if } \sum_{1}^{n} w_i x_i < threshold \\ 1, & \text{otherwise} \end{cases}$$

- This is called a step function. You can think about it as a very simple decision making device, weighing up evidence from the input neurons.

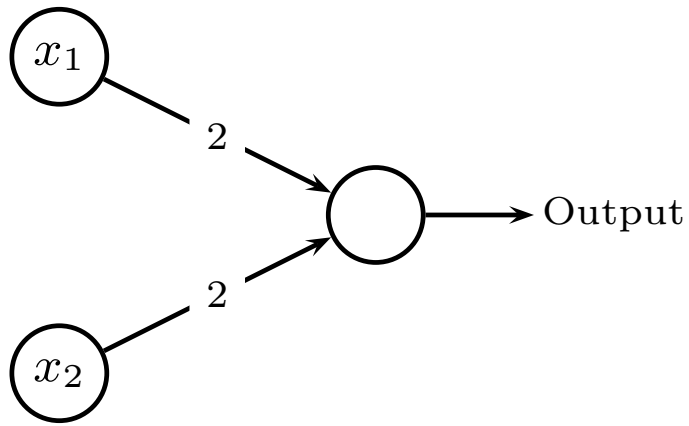
An or perceptron

Here is a simple example, a perceptron with two input neurons that calculates logical OR:

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

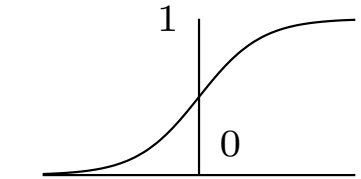
The activation function is:

$$Output = \begin{cases} 0, & \text{if } \sum_{1}^{n} w_i x_i < 2 \\ 1, & \text{otherwise} \end{cases}$$



A Perceptron

- The input neurons are binary in this case and the weights are 2.
- It is easy to see that the output is 2 if and only if both inputs are 0, and 1 otherwise.
- The step function from above is often replaced by a sigmoidal function with a smoother threshold: $Output = \frac{1}{1+e^{-x}}$.



Sigmoidal function

- We can use this simple perceptron model to build increasingly complex networks.
- Each of the perceptrons in a given layer makes a decision based on the input from the previous layer.
- This way, a many-layer network of perceptrons can engage in sophisticated decision making.

Types of Neural Networks

Definition (feed forward network)

A feed forward network is a network where the output is only propagated in one direction: From the input to the output.

- This is the simplest neural network model.
- There are no feedback connections in which outputs of the model are fed back into itself.

Types of Neural Networks

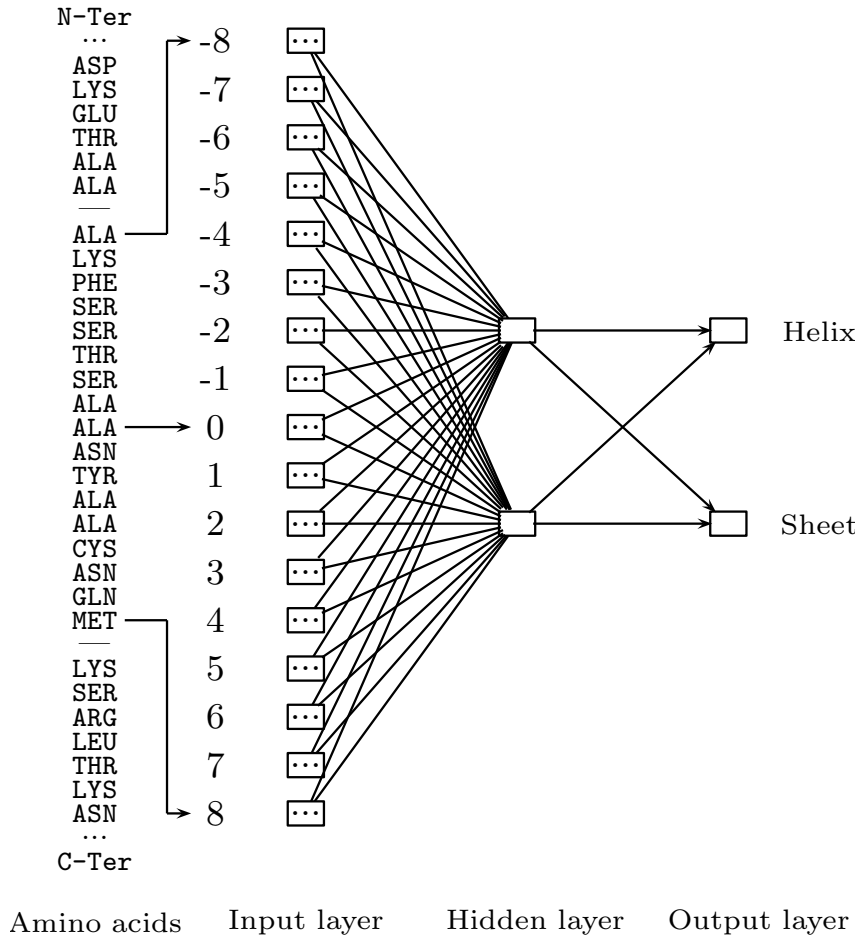
Definition (backpropagation network)

A backpropagation network is a network where an output can be fed back through the network in order to minimize the output error.

- Arbitrary weights are initially assigned and the output values are compared with the correct answer (target output) to compute the value of some predefined error-function.
- The error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount.
- The process continues for a pre-defined number of rounds or until the output converges to a good enough value.

ANN for Secondary Structure Prediction

- A representative example goes back to 1989.
- The input was a set of 62 proteins. 48 were used as the training set, and the remaining 14 were the test set.
- The network consisted of one input layer, a single hidden layer and an output layer.



ANN for Secondary Structure Prediction

- The input layer was a sliding window of size 17 on the amino acid sequence.
- 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 is 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.

ANN 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.
- 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.

Basics of Network

- The input layer is of size 21 (NOT 17), since every amino acid is a vector of size 21.
- Therefore, every training cycle feeds a 21×17 matrix to the network.
- There are two sets of weights - input to hidden (of size 21×2) and hidden to output (of size 2×2)
- The weights can be organized in a matrix, where each row is the weights associated with a single neuron.
- This matrix will be multiplied by a vector that contains the input neurons.
- Optionally, you can add bias.

Feed-Forward

- The weights can be organized in a matrix, where each row is the weights associated for a single neuron.
- This matrix will be multiplied by a vector that contains the input neurons.
- This is the input to the next layer.
- The activation function maps the result into $[0, 1]$ through multiplication by an activation function.
- Sigmoid is a popular choice.
- The process repeats from one layer to the next.

Backpropagation

- This stage feeds the output of the Feed-forward back to the network and tries to reduce the error.
- First, measure the difference between the output and label (known output).
- Don't forget that the output of one layer is the input to the next.
- We work our way backwards – calculate the output error rate (derivative of error with respect to output), subtract the error from the weights.
- Repeat for all the layers – the error from layer n+1 is fed to layer n.
- Some more background here: https://towardsdatascience.com/math-neural-network-from-scratch-in-python-d6da9f29ce65
- This is minimization with gradient descent.

Generating the Data

- There are many ways to slice the data into windows of 17.
- This implementation slices off the entire sequence at once:
- A hankel matrix is a square matrix in which each ascending skew-diagonal from left to right is constant, e.g.

$$\begin{bmatrix} a & b & c & d & e \\ b & c & d & e & f \\ c & d & e & f & g \\ d & e & f & g & h \\ e & f & g & h & i \end{bmatrix}$$

- Details:

  https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.hankel.html
- In this implementation the matrix is built for every 17 consecutive amino acids (notice that the last 16 rows are sliced off since they are superfluous).
- Each window is one-hot encoded and trained.
- Notice that only one output, for the central amino acid, is given for backpropagation.