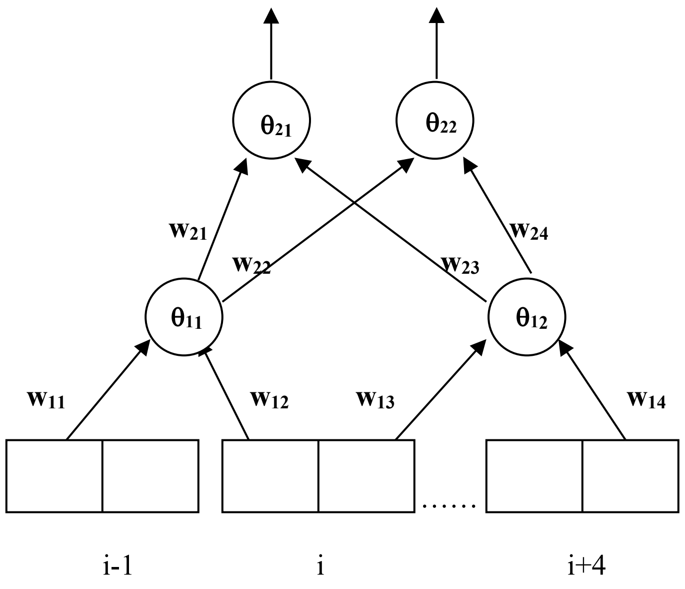
277B: Machine Learning Algorithms

Homework assignment #3: Computing with neurons
Assigned Feb. 13 and Due Feb. 23

1. Artificial Neural Networks. In lecture we saw that ANN can be used for secondary structure prediction of alpha-helix, beta-sheet, or random coil given the amino acid sequence as input. Each amino acid is represented by two numbers, the first representing its propensity to be hydrophobic (+1) or hydrophilic (-1), while the second is its propensity to form helix (+1) or not helix (-1). A helix is predicted by the network if the output is (1,-1), β–sheet if output is (-1,1), and coil if (-1,-1). We could design two simple Boolean functions as part of a bigger network with the following connectivity:



Please code up your own little neural network using NumPy! If you choose to do it with code, to make life easier you can also treat this architecture as a fully connected network of shape (6,2,2).

(a) Initialize the weights to random values between 0 and 1.0

(b) Given the following input values for one pattern:

i-1=(-1,1) i=(-1,-1) i+4=(1,-1)

Feedforward through the above network and give the calculated output for this pattern and corresponding secondary structure definition (assume a hyperbolic tangent activation function, and its derivative is given as: $\frac{d\tanh x}{dx} = 1 - \tanh^2 x$).

(c) The actual *observed* output for this one pattern is (-1,-1). Define the error and calculate it for all nodes that are not in the input layer.

(d) Using back-propagation, give a formula for the weight update of all of the w_{ij}'s, and then calculate weight adjustments with a learning parameter.

Reference Reading: http://neuralnetworksanddeeplearning.com/chap2.html

2. Logistic regression using a simple perceptron. For whether your status is sufficient to make it off the Titanic, download the Titanic data set from Files.

(a) Process the dataset. Filter out data with missing features. Use one-hot encoders to transform the categorical features and the output survival status.

Hint: Here are some brief descriptions for the dataset if you are unsure what each descriptor means.

(b) Fill in the code for a simple perceptron. Initialize your weights and biases between 0 and 0.05 using your random number generator. Use the mean squares error (MSE) to adjust your weights through back-propagation. Fill in the code for k-fold validation. Use 80% of the data for training and 20% of the data for testing and do 5-fold validation. Can we predict who will survive? Play around with the features to determine which ones give you a better chance to get back to shore.

3. Nonlinear regression using a simple perceptron and a simple ANN. Let's try to fit to a simple sine function y = 3 sin(x) + 5 with the simple perceptron and a simple fully connected network.

(a) Use the generate_data() function provided in the reference file to generate the training data (5000 points). Do 5-fold cross validation with the simple perceptron model. How well is the prediction? Generate 1000 new points as your test data, and use the show correlation() function to see how well your model agree with the test data.

(b) Use the multilayer perceptron regressor of scikit-learn as a simple fully connected neural network. Use one hidden layer first. Do 5-fold cross validation with this simple ANN, and report the MSE on each fold. Visualize the correlation of your model prediction and the true test data. Is the result better than a simple perceptron?

(c) Play with the hyperparameters for the ANN, can you improve the performance by changing the architecture or any other hyperparameters?