# Lab 7: Multiclass Classification with a Multilayer Neural Network

CMPS 3560 Artificial Intelligence

Alberto C. Cruz, Ph.D.

Department of Computer and Electrical Engineering and Computer Science

Document version 4/4/2018

## Prelab

There is no prelab for this lab. Though, you may want to re-read the following book chapters:

- Negnevitsky 6.4
- Russel and Norvig 18.7.3-18.7.5

## Datasets

A repository of many data sets can be found here: https://github.com/DrAlbertCruz/CMPS-356-Datasets. The data here is multi-class, do not use the old CSV files.

You should start with the Fisher's Iris Dataset (iris.csv), because we have worked with it before. We know that it's an easy dataset to work with, so you should get close to 100%. If you do not, something is wrong. Later, you can experiment with the other datasets.

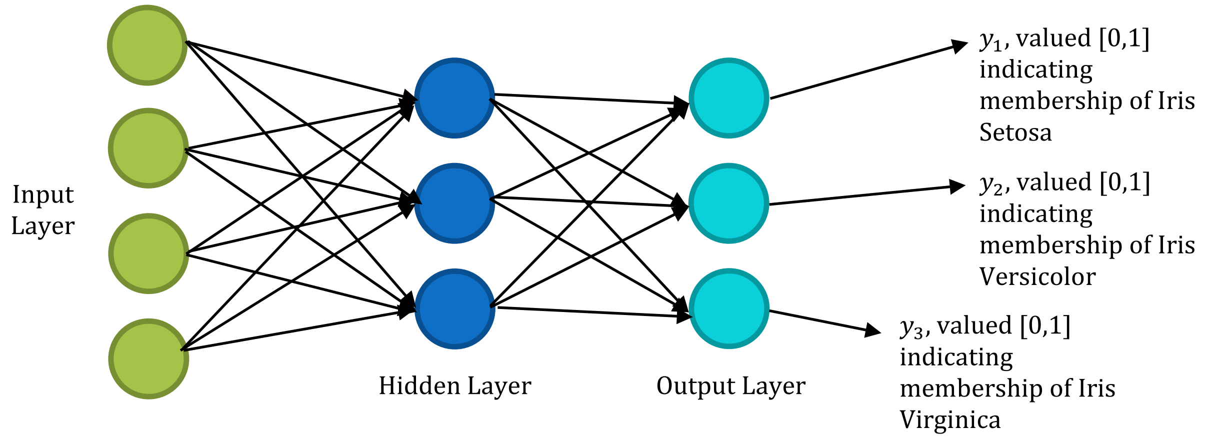
## Introduction

### Goal

- Encapsulate the perceptron from the previous lab into a neuron class
- Implement backpropagation
- Try other datasets

### Background

In this lab we will expand upon the previous lab by implementing a multilayer neural network. Implementing the previous lab's neuron as a class/struct will reduce your work. Each node in a graph representing a neural network is an instance of a neuron. Consider the following multilayer neural network:


Input Layer
Hidden Layer
Output Layer
$y_1$, valued [0,1] indicating membership of Iris Setosa
$y_2$, valued [0,1] indicating membership of Iris Versicolor
$y_3$, valued [0,1] indicating membership of Iris Virginica

*This graph omits the bias term. Every neuron must have a bias that updates with other weights.* Nodes in the input layer represent features, they are not proper neurons. Each node in the hidden layer(s) have their own weight vector $\boldsymbol{w}_j$. Each neuron outputs a weighted linear combination of the features $h(\boldsymbol{w}_j \cdot \boldsymbol{x})$. The nodes in the output layer also each have their own weight vector $\boldsymbol{w}_k$, and output a weighted linear combination of the features:

$$y = h\left(\begin{vmatrix}\text{Output of Hidden Layer Node 1}\\ \text{Output of Hidden Layer Node 2}\\ \vdots\end{vmatrix} \cdot \boldsymbol{w}_k\right)$$

$h$ is the activation function. Previously we were using a step function. Now we will use a sigmoid:

$$y = h(X) = \frac{1}{1 + e^{-X}}$$

In short, compute the dot product and plug it into the equation for $h(x)$. This is the output of a neuron.

## Multiclass Classification

In the last lab we implemented a perceptron. It predicted only one of two classes. This is binary classification. We used this algorithm to predict if a sample was I. Virginica or not. In this lab, we will frame a neural network to output three values. Each value corresponds to your confidence that a sample belongs to one of our three classes. A classifier that can output predictions at once is a multi-class classifier. A new version of the iris dataset has been uploaded:

| Column | Info |
|---|---|
| **1 - 4** | Features as usual |
| **5** | 0 – Not Iris setosa<br>1 – Iris setosa |
| **6** | 0 – Not Iris versicolor<br>1 – Iris versicolor |
| **7** | 0 – Not Iris virginica<br>1 – Iris virginica |

See the "Datasets" section for where to download it.

## Measuring Performance

Note that our output has changed to a three-length vector. Our prediction has three numbers for each of the three columns in the data. You can compute the error for a specific sample as:

$$\|\boldsymbol{y}_d - \hat{\boldsymbol{y}}\|_1$$

This is a 1-norm (Manhattan distance). For example, if the first row has: $y_d = [1\ 0\ 0]^t$. Suppose $\hat{\boldsymbol{y}}$, the concatenated (joined) output values) of $y = [0.02\ 0.54\ 0.87]^t$:

$$\left\| \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 0.02 \\ 0.54 \\ 0.87 \end{vmatrix} \right\|_1 = |1 - 0.02| + |0 - 0.54| + |0 - 0.87| = 2.39$$

The performance of the entire epoch should be calculated as the *mean absolute deviation* which is average error of each sample for the entire epoch:

$$\frac{1}{\text{\# of Samples in Epoch}} \left( \sum_{\text{Each Sample in Epoch}} \|\boldsymbol{y}_d - \hat{\boldsymbol{y}}\|_1 \right)$$

## Technical Approach

Implement a multilayer neural network using the framework you developed in the last lab. A single hidden layer with three hidden layer neurons, and an output layer with three output neurons should suffice. This is a parameter that you can investigate with other datasets if you finish early. Like the last lab, you should output some values. Outputting all of the weights would be too much information to take in at once, so just output the 1-hot prediction vector:

```
...
Epoch 19 Results: MAD = 2.249
...
Epoch 20, Iteration 149: Prediction is [0.01, 0.54, 0.98].
Epoch 20, Iteration 150: Prediction is [0.74, 0.87, 0.40].
Epoch 20 Results: MAD = 1.732
```

This is a mock-up of the results you will get. Your output may vary.

## Check Off

For full credit, demonstrate that your MAD value decreases from epoch to epoch. With the Iris dataset, you should get close to 100% accuracy. If you finish early, you should also look at the wine and breast cancer datasets. Note that the structure of your network may need to change as the number of output neurons must match the number of possible classes.