# EECS 349 (Machine Learning) Homework 7

## *How to submit your homework*

1. Create a **PDF document** containing answers to the homework questions.
2. Include source code for the program you write.
3. Compress all of the files specified into a .zip file.
4. Name the file in the following manner, *firstname_lastname_hw7.zip*.
3. Submit this .zip file via Canvas by the date specified on Canvas.

## *1) Perceptrons (2 points)*

Read Chapter 4 of "Machine Learning" (on the course calendar) and answer the following.

**A. (1/2 point):** Is a one-layer perceptron capable of representing non-linear decision surfaces? Why or why not?

**B. (1/2 point):** Is it possible use a multi layer perceptron with linear activation functions to represent a non-linear decision surfaces? Why or why not?

**C. (1/2 point):** In a multi layer perceptron, is there any advantage to using sigmoid activation functions compared to linear activation functions? If so, what is it? If not, why not?

**D. (1/2 point):** If the sigmoid function in a multi-layer perceptron were replaced with the following function, how would this affect the back propagation of error training method?

$$output = sign\left(\frac{1}{1+e^{-net}} - 0.5\right) \qquad sign(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ -1 \text{ else} \end{cases}$$

## *2) Restricted Boltzman Machines (1 point)*

From the course calendar, follow the link to the tutorial on Deep Belief Networks and read the paper called "Scaling Learning Algorithms towards AI"…and there is always the Wikipedia. These resources will help you answer the following questions.

**A. (1/2 point):** Explain what a Restricted Boltzman Machine (RBM) is. Don't just give a sentence.

**B. (1/2 point):** What relationship do RBMs have to Deep Belief Networks?

## *3) Scaling Learning Algorithms towards AI (1 point)*

Read **"**Scaling Learning Algorithms towards AI" and answer the following questions.

**A. (1/2 point):** What are Kernel Machines? Describe two limitations of Kernel Machines.

**B. (1/2 point):** What are Deep Architectures? .How do the authors expect Deep Architectures will get around the limits of Kernel Machines you describe in part A?

# EECS 349 (Machine Learning) Homework 7

## *4) Activation functions (1 point)*

Modern neural nets use many different kinds of activation functions in their nodes. You've already learned about *linear*, *perceptron* and *sigmoid* activation functions. Two other popular ones are *softmax*, and *ReLU*. Research these on the web.

**A. (1/2 point):** Write the formula for a ReLU activation function. Show a qualitative plot that shows the shape of the ReLU function. Don't forget to cite your sources.

**B. (1/2 point):** Write the formula for a softmax activation function. Give one reason for using a softmax function instead of normalizing a distribution in a standard way (i.e., dividing by the sum of all values)? Don't forget to cite your sources.

## *5) Learning tensorflow/tflearn (2 points)*

For this problem and problem 6 we will be using Google's open source, deep learning framework, TensorFlow (see www.tensorflow.org). More specifically, we will be using a wrapper library called tflearn (www.tflearn.org), a simpler interface for implementing neural networks.

TensorFlow and tf learn only work on 64-bit machines, so we have uploaded a new VM that is 64-bit Ubuntu and has TensorFlow and tflearn pre-installed. See Piazza for instructions on how to use this. You also have the option to download and install them on your own computer. (installation instructions for both packages are in the aforementioned links. But in summation, use python's command-line software manager, `pip`, to install these libraries. Note: do **not** download the GPU versions, **you will only need the CPU version**.) We will be available to help with any installation issues during recitation and office hours this week.

A good way to check if these packages are installed correctly on your machine is to run the following from the command line:

```
$ python
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> import tflearn
>>>
```

Figure 1. TensorFlow and tflearn installed successfully. The lines with the three carrots (>>>) are the lines we use to test the installation of these packages. They don't throw errors so we know that these packages are installed correctly.

In the above example, the first line is calling python in the interactive shell from the command line. The next three lines about the specific version of python being used, but what's important here is importing tensorflow and tflearn. If no errors get thrown then tensorflow and tflearn are installed correctly. See below for an example of what a fail to find tensorflow looks like.

```
$ python
Python 2.7.10 (default, Aug 22 2015, 20:33:39)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named tensorflow
>>>
```

Figure 2. TensorFlow installed incorrectly.

# EECS 349 (Machine Learning) Homework 7

Once TensorFlow and tflearn are installed correctly, read over the tflearn tutorial located here: https://github.com/tflearn/tflearn/blob/master/tutorials/intro/quickstart.md to get a sense of what using tensor flow with tflearn is like. For this problem, you will be using the code we have provided to you in `titanic_predictor.py` to answer the following questions. You do not need to alter this code for this problem.

**A. (1/2 point)** Describe the architecture of the neural network provided in this tutorial. Including the input and output layers, how many total layers does it have? How many of these layers are "hidden layers"? How many nodes does each layer have? What are the activation functions used in each layer? How many epochs does this network train for? *Hint:* You may have to research the tflearn documentation to find which activation functions are used.

Run the provided code in titanic_predictor.py without altering it. Tflearn will output information at each "snapshot" step of the gradient descent. There is important information here about how well your neural network has learned; things like training loss and how accurate the network is on the training data in its current state. (Note: One epoch is one presentation of all the training data to the learner. One training step is one presentation of a single data point to the learner.) Now use the output of titanic_predictor.py to answer the following questions.

**B. (1/2 point)** What is the number of the first training step where your system shows non-zero accuracy on the training data? What is the final accuracy value on the training data?

**C. (1/2 point)** What is the final accuracy after you increase the number of epochs to 20? What about 100? What does this tell you about the power of this network architecture?

**D. (1/2 point)** What is the chance that Leonardo DiCaprio survives? What about Kate Winslet?

## *6) Using tensorflow/tflearn (3 points)*

Just as in Problem 5, you will be using TensorFlow and tflearn to complete problem 6.

In spiral_data.csv you will find a dataset that represents four different spirals emitting from the origin. The file has three columns: an x coordinate, a y coordinate, and a class (0, 1, 2, or 3). Each row specifies a single data point. If you plot this data, it is plain to see the spiral arms and their classifications (see next paragraph on how to visualize this data).

For this problem you will be asked to use tflearn to build a multiclass classifier to determine which coordinate point is which spiral arm. The starter code for this problem is provided in `spiral_classifier.py`. Include all of your code for this problem in this file. We have provided two plotting functions: one that will plot the data without your model's classifications (`plot_spiral()`), and another that will plot the data and your neural network's prediction (`plot_spiral_and_predicted_class()`). You should use the first plotting function to visualize the data before you start, but you are not required to include this plot in your write up.

**A. (1/2 point)** Using tflearn, build a classifier for the data in spiral_data.csv that will classify a data point into one of the 4 classes. The first layer of your neural network is the input layer that should input an `x` and `y` coordinate. The next layer should be a fully connected layer with four nodes and a softmax activation function (in tflearn, a "fully connected layer" means that every node in this layer is connected with every node from the previous and next layers, but there are no connections between the nodes of this layer). The loss function for your regression object should be '`categorical_crossentropy`' (this is the name of the input parameter that tflearn accepts). Your code should go in spiral_classifier.py under the following function definition:

```
def linear_classifier(position_array, class_array, n_classes):
```

And should be callable like this:

```
$ python spiral_classifier.py <path/to/data.csv> <0>
```

Where the '0' means that spiral_classifier.py will call your `linear_classifier()` function and '1' means that `spiral_classifier.py` will call `non_linear_classifier()` function (below). Your `spiral_classifier.py` program should not output anything other than what tflearn outputs by default.

**B. (1/2 point)** Now train your classifier and report results. Describe any testing/training choices you made on the data. Decide how many epochs you want to train the linear classifier. Include a justification for this choice. Include a plot of the data with your classification results in your write up (use the second function for this). What was the accuracy of the classifier after training? Is it possible to correctly classify all of the datapoints with just one layer? Why or why not?

**C. (1/2 point)** Now you are going to add a layer to your neural network. Build a neural network that has an input layer and **two** fully connected layers (meaning that the middle layer is a *hidden layer*). The last fully connected layer is the output layer (so it should have 4 nodes and a softmax activation function, as it did in part A, above). You get to decide how many nodes the middle layer has and the activation function that it uses. The goal is to make a better classifier than the one from Part A of this problem. Your code should go in `spiral_classifier.py` under the following function definition:

```
def non_linear_classifier(position_array, class_array, n_classes):
```

And should be callable like this:

```
$ python spiral_classifier.py <path/to/data.csv> <1>
```

As described above.

**D. (1 point)** Now train up the two_layer_classifier from part C and report on how well it learned. How did the number of epochs impact on how well it learned? Include a plot of your classification results in your write up (again, using second plotting function).

**E. (1/2 point)** Explain your decisions for the choices you made when making the network from part C and the choices you made in training the network. Details you should give include the number of nodes each layer has, the activation functions (e.g. ReLU, softmax, linear) for those layers, how many epochs you let your network train, etc. Also describe any testing/training choices you made on the data.