# EECS 349 (Machine Learning) Homework 3

## *How to submit your homework*

1. Create a **PDF document** containing answers to the homework questions. Show your reasoning in your answers.
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_hw3.zip*.
3. Submit this .zip file via Canvas by the date specified on Canvas.

## *1) Linear Regression (3 points)*

Load the *linearreg.csv* file. This is the file you will use for this problem. There are two vectors in the file ***X*** and ***Y***. ***X*** consists of 30 instances of a univariate attribute vector, and ***Y*** is the response vector. The intent of this problem is to get hands on experience doing polynomial regression (and its limits) and to use cross-validation to get an idea of how model complexity relates to bias/variance/MSE.

A. (2 point) Using n-fold cross-validation (the value of n is your choice, but must be explained in your answer) with $k^{th}$ polynomial regression, fit a function to the data for values of ***k*** between 0 and 9. In your homework, show the plot of the mean square error on the validation set(s), averaged over all of the folds, as a function of ***k***. Also, plot the best function overlaying a scatterplot of the data. The code for your work must be in a single file called *nfoldpolyfit.py*. The stub for this file has been provided to you as an example. Below is the function signature, as well as how it will be run from the command line

```
def nfoldpolyfit(X,Y,maxK,n, verbose)
python nfoldpolyfit.py <csvfile> <maxdegree> <numberoffolds> <verbose>
```

B. (0.5 points) Which value of ***k*** yielded the best results, in terms of accuracy of the prediction?

C. (0.5 point) Predict the response for a new query, ***x***=3. Given the performance during cross-validation, do you think this is an accurate prediction? Is there anything about the value 3 that should give you pause, given the training data? Explain your reasoning.

## *2) Classification with Regression (2 points)*

A. (1 point) Explain how to do classification via regression. Be clear. Use a graph to illustrate.

B. (1 point) Explain a weakness of classification via regression. Be clear. Use a graph and a concrete example to illustrate.

# EECS 349 (Machine Learning) Homework 3

## *3) Linear Discriminant Analysis (2 points)*

Linear Discriminant Analysis (LDA) is a common technique for learning (you guessed it) a linear discriminant. Look at chapter 4 of the Elements of Statistical Learning (that's a free book, linked to repeatedly from the course calendar). Look at the lecture slides (there may be more on the slides than was covered in class). Now answer some questions about LDA.

A. (0.5 points) When there are 3 or more classes to be distinguished, what is a situation where LDA will do a better job than classification via regression?

B. (0.5 points) What assumptions does LDA make about the data distribution(s)?

C.(0.5 points) What assumptions does Quadratic Discriminant Analysis (QDA) relax compared to Linear Discriminant Analysis? What are the upsides and downsides of relaxing those assumptions?

D.(0.5 points) How could I find a decision surface that could be modeled with a polynomial using LDA, not QDA.

## *4) Perceptron Linear Discriminants (3 points)*

Load the *linearclass.csv* file. There are four vectors in the file: ***X1***, ***Y1***, and ***X2***, ***Y2***. Vectors ***X1*** and ***X2*** both consist of 200 instances of a univariate attribute vector. ***Y1*** and ***Y2*** are the respective output labels {-1,1} for each input vector, e.g. the $k^{th}$ labeled instance for ***X1*** is <X1(k),Y1(k)>.

**begin** **initialize** $\vec{w}, k \leftarrow 0$
  **do** $k \leftarrow k + 1 \bmod m$
    **if** $\vec{x}_k$ is misclassified using $\vec{w}$
    **then** $\vec{w} \leftarrow \vec{w} + \vec{x}_k y_k$
  **until** all examples are properly classified
**return** $\vec{w}$
**end**

*Figure 1. Pseudo code for Sequential Perceptron Algorithm*

Figure 1 contains pseudo code for the Sequential Perceptron Algorithm. Here...
***w*** is the parameter vector (weight vector and threshold value)
***m*** is the number of training examples
$x_k$ is the $k^{th}$ training example
$y_k$ is the $k^{th}$ training example class label {-1,1}

**A. (1 point)** Implement the sequential perceptron algorithm to learn the parameters for a linear discriminant function that correctly assigns X1 to class -1 or 1. The algorithm should terminate when the classification error is 0. Output the number of iterations of that the algorithm performed before convergence and the learned parameters. Name your file *perceptrona.py* and include it with your homework submission. Comment this code to the level you saw in the provided stub for *nfoldpolyfit.py* Below is how the function will be run from the command line, as well as the function signature. We have provided some starter code for reading in the csv file in perceptrona.py

```
def perceptrona(w_init,X,Y):
      #return a tuple (w, e)
      return (w, e)
python perceptrona.py <csvfile>
```

where...
***w*** is the parameter vector (weight vector and threshold value)
***e*** is the number of epochs (one epoch is one iteration through all of X) the algorithm performed
***w_init*** is the parameter vector (weight vector and threshold value)
***X*** is the matrix of training examples (i.e. each row is an attribute vector with a prepended '1' for the threshold term)
***Y*** is the column vector of class {-1,1} class labels for the training examples

**B. (1 point)** Using the same sequential perceptron algorithm from part A, learn the parameters for a linear discriminant function that correctly assigns ***X2*** to class -1 or 1. What happened and why?

**C. (1 point)** How can you transform the attributes of ***X2*** so that your algorithm can correctly classify the data? Add this transformation into your algorithm, and describe your changes. Name your function (and file) "perceptronc(.py)" and include with your homework submission. This function should have the exact same input and output parameters (in the same order) as *perceptrona*.