EECS 349 (Machine Learning) Homework 2

WHAT TO HAND IN

A PDF document containing answers to the homework questions. Show your reasoning in your answers.

The source code for the program you write (in Python 2.7 using SciPy, if needed).

HOW TO HAND IT IN

1. Compress all of the files specified into a .zip file.
2. Name the file firstname_lastname_hw1.zip. For example, Bryan_Pardo_hw1.zip.
3. Submit this .zip file via Canvas

DUE DATE: See the course calendar

DUE DATE: the start of class as specified in the course calendar.

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? Explain the results in terms of bias and variance (you don't have to calculate bias or variance in the statistical sense).

C. (0.5 points) 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.

EECS 349 (Machine Learning) Homework 2

2) 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*.

EECS 349 (Machine Learning) Homework 2

3) Collaborative Filtering: Looking at the Data (1 point)

When doing machine learning, it is important to have an understanding of the dataset that you will be working on. On the course website under "links" there is a link to the MovieLens dataset. You will be using the MovieLens 100K dataset to build a collaborative filter to predict the rating a user may give to a movie they haven't seen. This dataset has 100,000 ratings from 943 users on 1682 movies.

A. (1/2 point) Go through each pair of users. For each pair of users, count how many movies both have reviewed in common. What is the mean number of movies two people have reviewed in common? What is the median number? Plot a histogram of this data where each bar is some number of movies reviewed in common and the height of the bar is the number of user pairs who have reviewed that many movies in common. Clearly label your dimensions. Explain any choices you made in the generation of this histogram.

B. (1/2 point) Go through the movies. For each movie, measure how many reviews it has. What movie had the most reviews? How many reviews did it have? What had the fewest? Order the movies by the number of reviews each one has. Now, plot a line where the vertical dimension is the number of reviews and the horizontal dimension is the movie's number in order by the number of reviews. Clearly label your dimensions. Now that you have this data, do you think the number of reviews a movie has follows Zipf's law? (http://en.wikipedia.org/wiki/Zipf's_law)

4) Collaborative Filtering: Distance measures (1 point)

A. (1/2 point) Assume a user will be characterized by a vector that has that user's ratings for all the movies in the MovieLens data. This means each user is described by a vector of size 1682 (the number of movies). Assume the distance between two users will be their Manhattan distance. Here is the problem: most users have not rated most of the movies. Therefore, you can't use Manhattan distance until you decide how to deal with this. One approach (call it approach A) is to put a 0 in for every missing rating. Another approach is to find the average rating chosen by that user on the movies he/she DID rate. Then, substitute that value in for all missing movie ratings (call it approach B). Which approach do you think is better? Say why. Give a toy example situation (i.e. 3 users, 5 movies, a few missing ratings) that illustrates your point.

B. (1/2 point) You are trying to decide between two different distance measures for movies: Euclidean distance and Pearson's correlation. Each movie is characterized by a vector of 943 (the number of users) ratings. Assume that all missing ratings values have the value 3 filled in as a place-holder. Which distance measure do you think would be better for item-based collaborative filtering? Why do you think that? Illustrate your point with a toy example.

5) Collaborative Filters (4 points)

You will now build two collaborative filters, each callable the same way

```
python user_cf.py <datafile> <userID> <movieID> <distance> <k> <i>
python item_cf.py <datafile> <userID> <movieID> <distance> <k> <i>
```

The input variables are as follows:

<datafile> - a fully specified path to a file formatted like the MovieLens100K data file *u.data*

<userID> - a userId in the MovieLens100K data

<movieID> - a movieID in the MovieLens 100K data set

<distance> - a Boolean. If set to 0, use Pearson's correlation as the distance measure. If 1, use Manhattan distance.

<k> - The number of nearest neighbors to consider

<i> - A Boolean value. If set to 0 for user-based collaborative filtering, only users that have actual (ie non-0) ratings for the movie are considered in your top K. For item-based, use only movies that have actual ratings by the user in your top K. If set to 1, simply use the top K regardless of whether the top K contain actual or filled-in ratings.

Filter *user_cf.py* must be a user-based collaborative filter and *item_cf.py* must be an item-based collaborative filter. All distances will be in the user/movie ratings space. Therefore movies will each have a 943-element vector (one element per user) and the ith element will contain the rating of that movie by user i. People will have 1682 element vectors, where the jth element contains that user's rating for movie j. Missing ratings must be filled in by the value 0. The predicted rating returned will be the mode of the top K neighbors.

Each call of your collaborative filters should create an output on the command line that contains the following information, in the order below, formatted as shown in the example below. In real use, the numbers here would vary depending on userID, movieID, etc.

> userID:123 movieID: 456 trueRating: 2 predictedRating: 4 distance:1 K:3 I:1

6) Test your movie ratings predictor (3 points)

Your final job is to determine the relative effectiveness of your collaborative filters. You will test things with cross-validation. Split the MovieLens 100K data in *u.data* into 100 subsets of size 1000 each. Select a subset of 1000 ratings you will use as a test set. Use the other 99,000 ratings as your prior data for collaborative filtering. For each of the 1000 movie-ratings in your data, hand the movieID-userID pair as input to a variant of your collaborative filter and get back the predicted rating. You can then compare it to the actual rating. If you do this for all 1000 items in your test set, you can get an average error on this set. Select another subset of 1000 and repeat the process. You can do this 100 times to get 100 averages. Now you have lots of data on the performance of this variant of your filter. You can then repeat this for other variants and compare their performance. This is the basic idea. You will be expected to do 100-fold cross validation when performing the experiments used to answer the questions.

A. (1/2 point) Clearly define an error measure for collaborative filtering that you will use in your experiments. Use a formula, text and an example to explain your error measure. Explain why you made this choice. *(Consider the following: Should the grade on each prediction be right/wrong? Or, does the degree of error on each rating matter? Maybe predicting a 4 when the truth is 1 is worse than predicting a 2 when the truth is 1…or maybe not)*

B. (1/2 point) Define the statistical test you will use in subsequent experiments to determine whether the difference between two system variants is statistically significant. Explain why you chose the test you selected. What is the null hypothesis? What level of confidence is sufficient to call the results from two filters statistically different?

For Parts C, D, E, F you are required to explain the assumptions you made in answering each question and state how you set the parameters you were NOT varying in that particular question. You must show a graph for each one that illustrates the difference (or lack of difference) between your variants. You must say whether the difference is statistically meaningful and back that up with the results of your statistical test(s).

EECS 349 (Machine Learning) Homework 2

C. (1/2 point) How does your choice for <distance> affect item-based collaborative filtering on the MovieLens 100K data? Does this correspond to the intuition you developed in problem 4?

D. (1/2 point) How does your choice for <i> affect user-based collaborative filtering on the MovieLens 100K data? Does this correspond with the intuition you developed in problem 4?

E. (1/2 point) Use the best settings learned from parts C and D. Varying <k> by powers of two: 1,2,4,8,16,32. What value for <k> is best for user-based collaborative filtering?

F. (1/2 point) Use the best settings learned from parts C, D, E for both item-based and user-based filters. Compare user-based to item-based collaborative filtering. Which is better? How confident are you in this result? Why or why not?