EECS 349 (Machine Learning) Homework 2 Solutions

Name: Dipendra Jha (NetID: dkj755)

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.

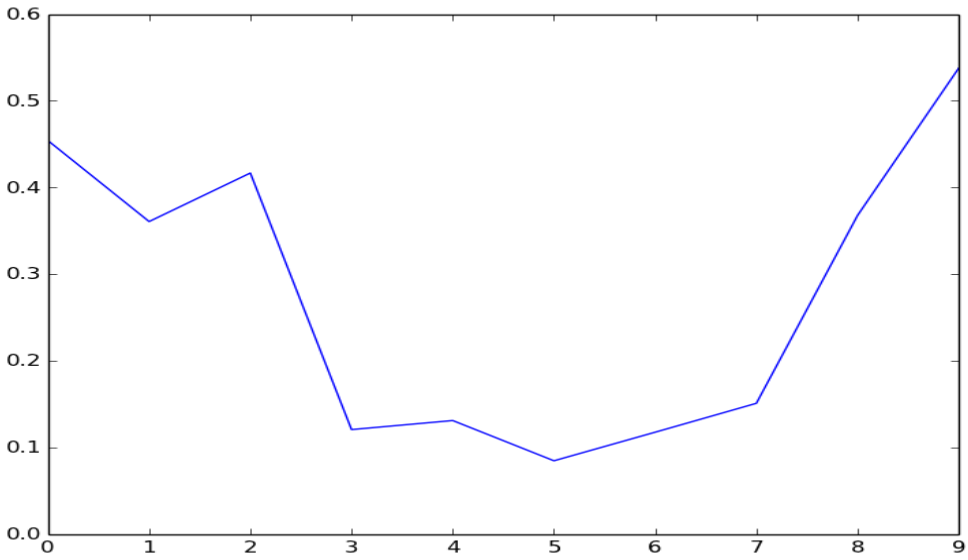
A. (2 point) Using n-fold cross-validation (the value of n is your choice, but must be explained in your answer) with k^{th} polynomial regression, fit a function to the data for values of ***k*** between 0 and 9. In your homework, show the plot of the mean square error on the validation set(s), averaged 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>

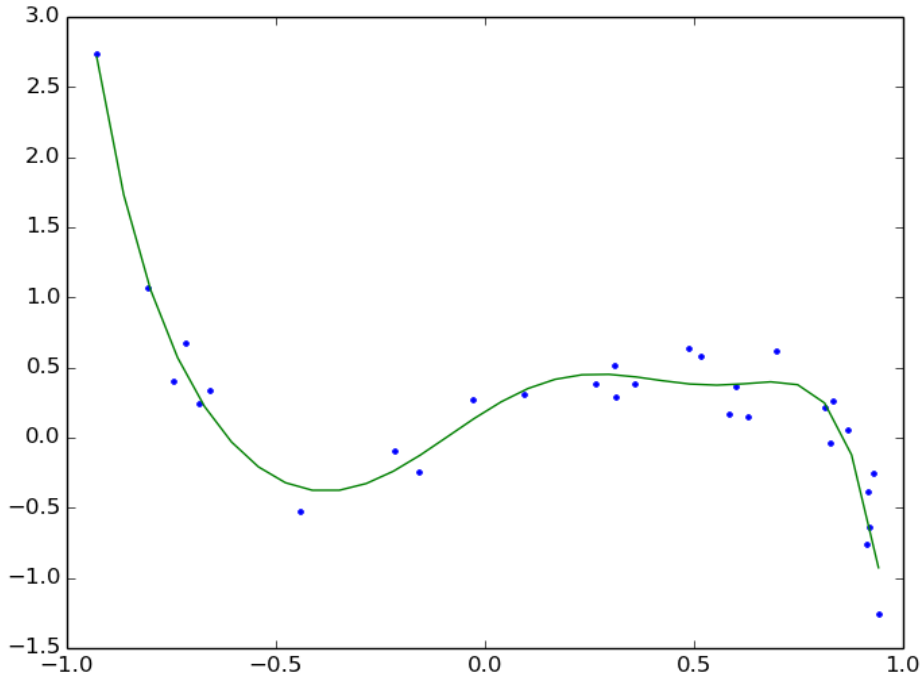
Solution:

Code is included in the file.

MSE vs K:



Best Poly fit, k=7:



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).

Solution:

Best K:
In 90% of the cases, the best values for K is 7 or 5. Few of the times 3 was also outputted.

Bias vs Variance:
During cross-validation and graph plotting, I observed that at first mean squared error decreases with K. So, performance increases. But, after a certain value of K (=7), this mean squared error again starts to increase, thus performance decreases. We can see this phenomena from the graph in 1(a).
This is caused by the increase of variance and decrease of bias value as K increases. There is optimal region where bias and variance both is sufficiently less (K=3 to K=7), and then variance increases rapidly.

When k=low, bias is high and variance is low, causing under-fitting.
When k=high, bias is low and variance is high, causing over-fitting.

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.

Solution:

When a new query arrives, with the optimal polynomial weights using K=57, the prediction for (X=3) is = - 20646.0735.

Analysis:
As seen in cross-validation and other values from data set, this prediction is not accurate, as no Y-value in the dataset has this much negative value.
In the training data, the values of X range from [-1, 1]. This causes us to think and pause, when we get X=3 to predict, as there is no value of X that is quite near to this. This prevents the accuracy of the predictor when predicting a Y-value for X=3.

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 $w, k \leftarrow 0$

do $k \leftarrow k + 1 \bmod m$

if $x_!$ is misclassified using w

then $w \leftarrow w + x_! y_!$

until all examples are properly classified

return 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$\boldsymbol{x_k}$ is the k^{th} training example

$\boldsymbol{y_k}$ 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

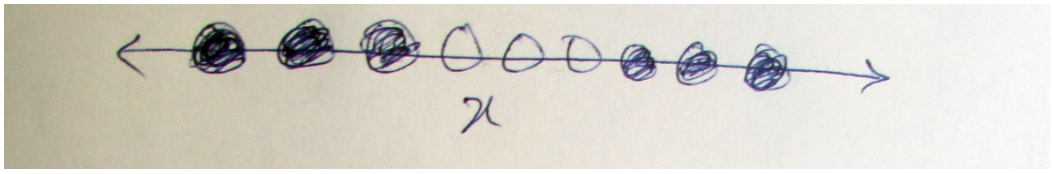
Solution:
Code is included in the file.
Learned values: **W:** W_0 (Threshold) = -11, W_1 = 4.06.

Here, $g(x) = W_1 * X + W_0$.

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?

Solution:

Result:
The algorithm could not learn the parameters for a linear discriminant for X2 dataset. It never onverged.
Reason:
X2 dataset is not linearly separable. If we plot the values of X2 along one axis, and see the corresponding Y2 values, we will get something like this:



As, from the dataset, we can see:
X2 < 1.0: -1
1.0 < x2 < 2.0: 1

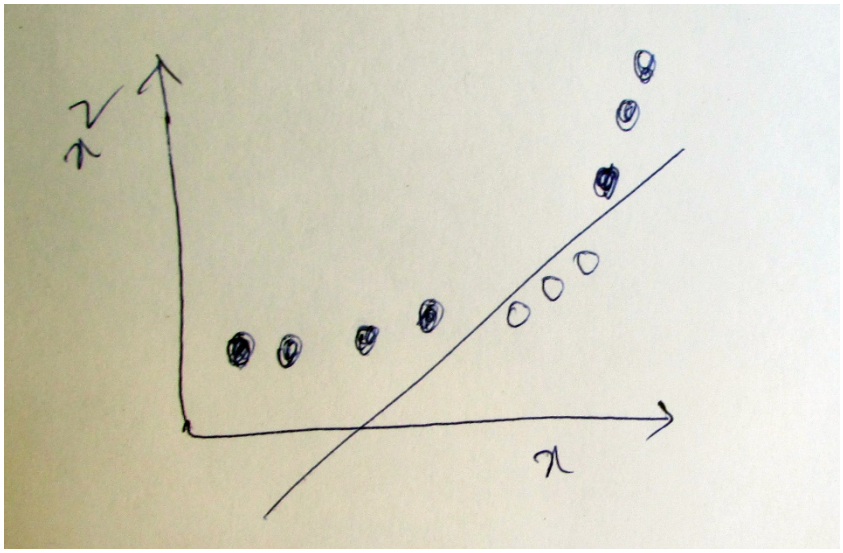
X2 > 2.0: -1
Thus, this is not linearly separable.

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*.

Solution:

Transform:

We can transform the X2 attribute to higher dimensional space, in which they will become linearly separable. In this case, we can map attribute X2 to $\{X2^2, X2\}$. Then we will get a linear separator for the dataset. As an example:



Now, we need to learn:

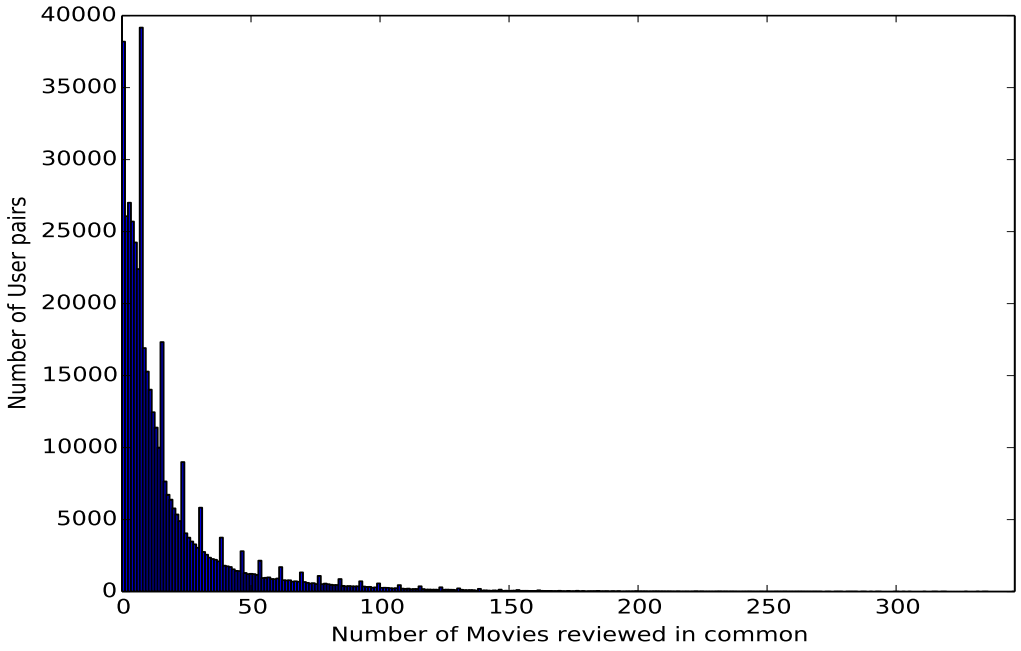
$G(x)= W_0 + W_1*X2^2 + W_2*(X2)$

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.

Solution:



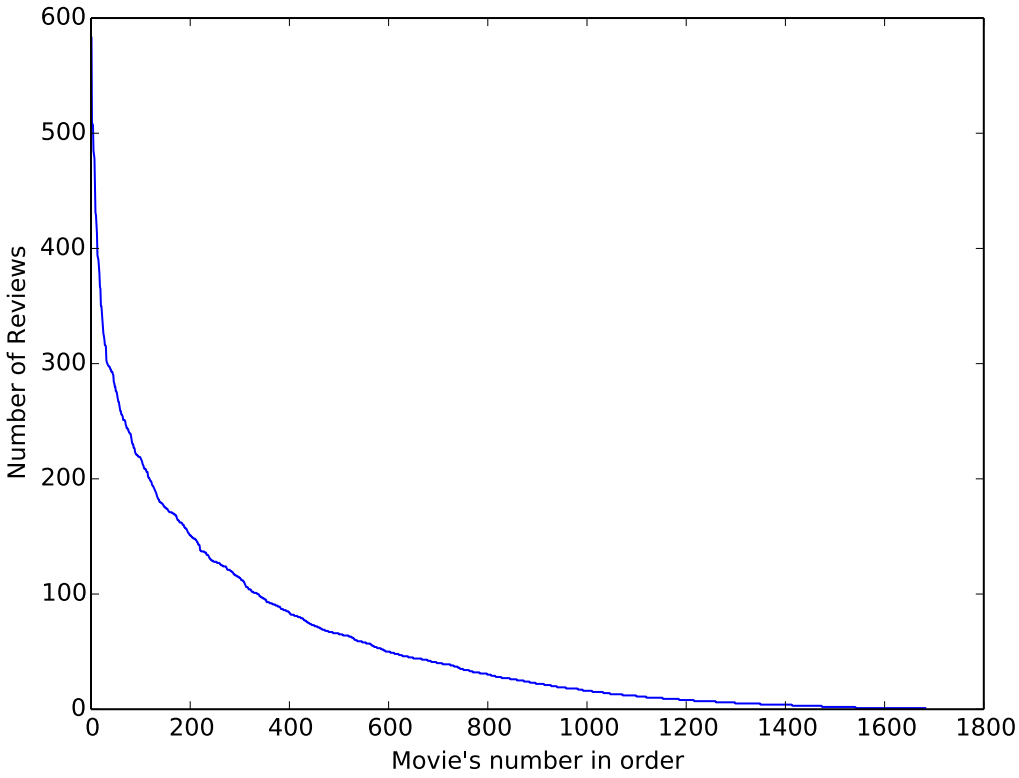
Mean number of movies two people have reviewed in common: 18

Median number of movies two people have reviewed in common: 10

I have used plot.hist() function of matplotlib for plotting the 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)

Solution:



Movie number **50** got most number of reviews. It got **583** reviews.

First movie (in increasing sequence of movieID) with least number of movie was **599**, getting **1**

review. There were another movies after 599, that got 1 review each too.

I have plotted the movies in decreasing order of their reviews. We see that the number of reviews decreases exponentially with its increase in rank on x axis. This shows that the number of reviews a movie has follows 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.

Solution:

It will be better to use the average value than using 0 to replace missing values.

Lets assume the data is:

	Movies:		M1	M2	M3	M4	M5
Users:	U1	5	3	4	4	4	
	U2	5	4	3	?	?	
	U3	4	2	?	2	3	

Now, lets look at both approach to predict the rating for user U1 for M4:

Approach A: Filling with 0:

	Movies:		M1	M2	M3	M4	M5
Users:	U1	5	3	4	4	4	
	U2	5	4	3	0	0	
	U3	4	2	0	2	3	

Now distance using Manhattan distance:

Distance (U1, U2) = 0 +1 +1 + 4 + 4 = 10

Distance (U1, U3) = 1 + 1 + 4+ 2 + 1= 9

Approach B: Filling with average:

	Movies:		M1	M2	M3	M4	M5
Users:	U1	5	3	4	4	4	
	U2	5	3	4	4	4	
	U3	4	2	3	2	3	

Using Manhattan again,

Distance (U1, U2) = 0 + 0 + 0 + 0 + 0 = 0

Distance (U1, U3) = 1 + 1 +1 + 2 + 1 = 7

Here, we see that the U1 and U2 are more related to each other than U1 and U3. However, when we use the zero value, to replace two missing values of U2, its distance increases linearly with the increase in missing values compared to U3 that has just one missing value in this case. So, we see that if we put 0 for missing values, it will greatly affect our results. However, when using average

value, it gives a better measure. As we can see, U1 and U2 have same rating when using mean value for missing values. That sounds more accurate than in case of 0. So, I think replacing missing value with average is better than replacing with 0. This gives more accurate and close neighbors for predictions.

Another problem with filling with 0 is that the predicted value will be zero if its missing in the closest neighbor considered for prediction.

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.

Solution:

I think Pearson' correlation is better for item-based collaborative filtering. Pearson correlation captures the variance in ratings between two items better than Euclidean distance. Euclidean only considers the distance between two items, independent of how they are distributed.

Lets consider a toy example:

	Movies:		M1	M2	M3
Users:	U1	2	4	5	
	U2	2	3	4	
	U3	2	2	3	

Now, using Euclidean distance:

(M1, M4) = 2 + 0 + 1 = 3

(M2, M3) = 1 + 1 + 1 = 3

Using Pearson's correlation:

(M1, M2): avg1 = 3, avg2 = 3 sim (M1, M2) = (-1 + 0 + 1)/(1.414 * 1.414) = 0

(M2, M3): avg2 = 3, avg3 = 3 sim(M2, M3) = (2 + 0 + -1)/(1.414 * 2.36) = 0.316

From the data, we can see that M2 and M3 are more correlated as difference in their ratings is 1 in all case. But M2'ratings differs more from M1. So, in this case Euclidean distance fails to capture this closeness of two neighbors but Pearson's correlation captures it well.

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

Solution:

Codes is included in files item_cf.py and user_cf.py.

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)*

Solution: The error measure I will be using in my experiments for cross validation is absolute difference between true rating and predicted rating. This can be defined as:

Error (e) = | true rating – predicted rating |

In the case of collaborative filters, the values of outputs range from 1 to 5. As it is not binary (true and false) classifier, it won't be a good idea to define error to be prediction right or wrong. Suppose neighbor A has prediction 3 and neighbor B has prediction 5 for true prediction for C being 2. Now the distance between A and C is less than distance between C and B. And A is closer to C than B, so it would make sense to define error as the difference in predicted and true rating. Also, this error is a metric.

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?

Solution: For this experiment, I am using 50-fold cross validation. For each cross validation, I randomly pick 10 items from 2000 data in test set and use the 98000 as prior data. I reduced it from what's said above due to time constraints.

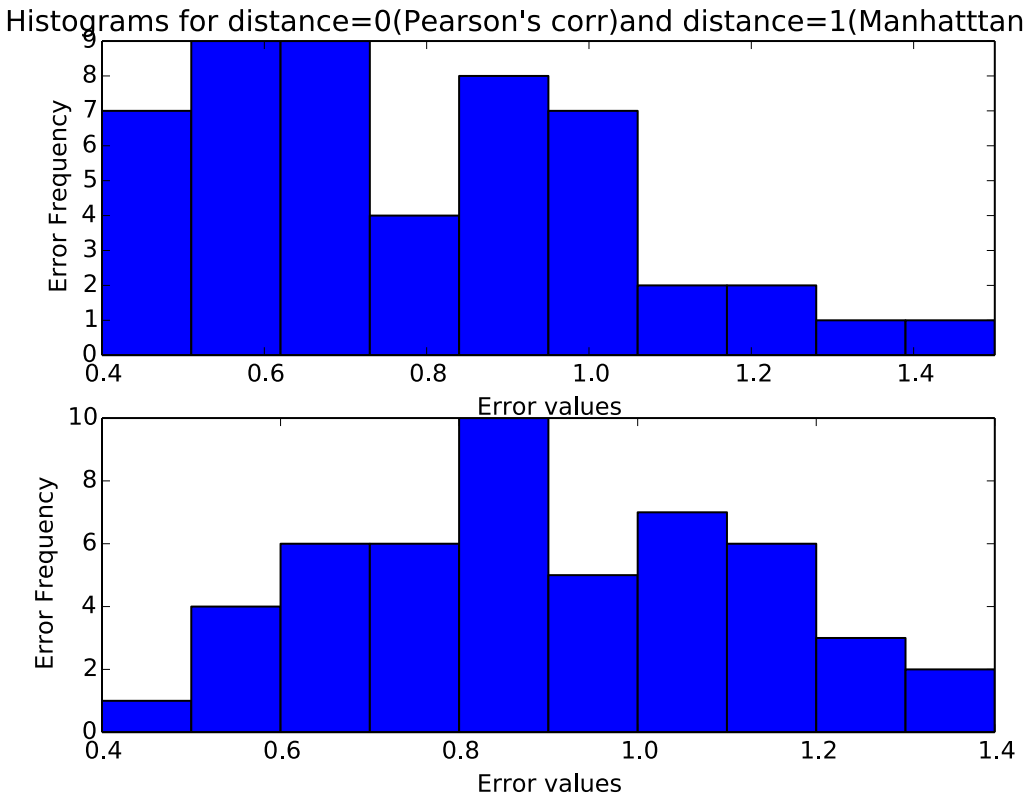
As I have more than 30 error average in each case and each data is independent and identically distributed, I assume it forms a normal distribution by Central Limit Theorem. With two independent normally distributed data, I will use independent samples t-test to see if the difference between two systems is statistically significant.

Null hypothesis is that the means of the two normally distributed populations are equal. To say that the two filters are statistically different, a confidence interval of 95% will be good. For this, the p-value should be less than 0.05. Otherwise, we can't infer that the two systems are 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).

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?

Solution:



I observed the following results for independent samples t-test:

Tstat = -1.182569, pvalue = 0.2398

And I calculated the average errors:

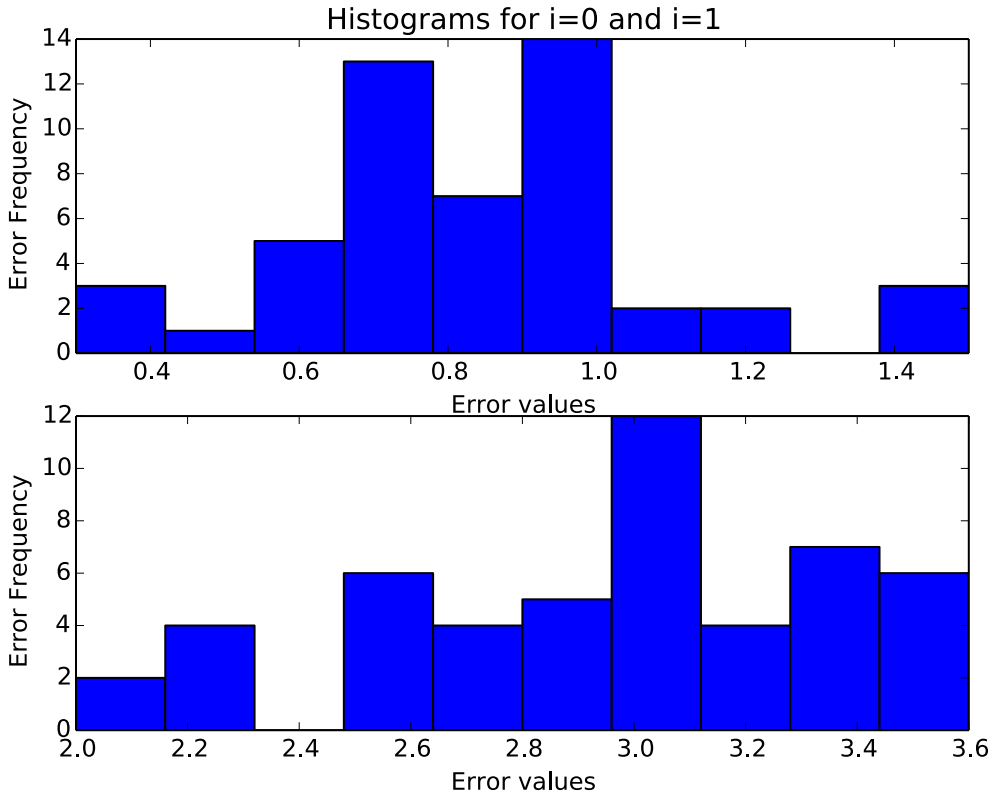
Average errors: 0.798 (distance = 0) and 0.854 (distance=1)

We see a small difference between the two. Pearson's correlation (distance=0) is better than Manhattan distance (distance = 1). This corresponds to the intuition I developed in problem 4.

However, from the statistical result of t-test we see that the p-value is 0.2398. For the two mean errors to be statistically different for a confidence interval of 95%, the p-value should be less than 0.05. So, from this experiment we find that null hypothesis is valid and our choice for distance does not have any statistically valid affect on item-based collaborative filtering on the MovieLens 100k data.

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?

Solution:



I observed the following results from independent samples t-test:

Tstat = -30.93, pvalue = 5.3e-47

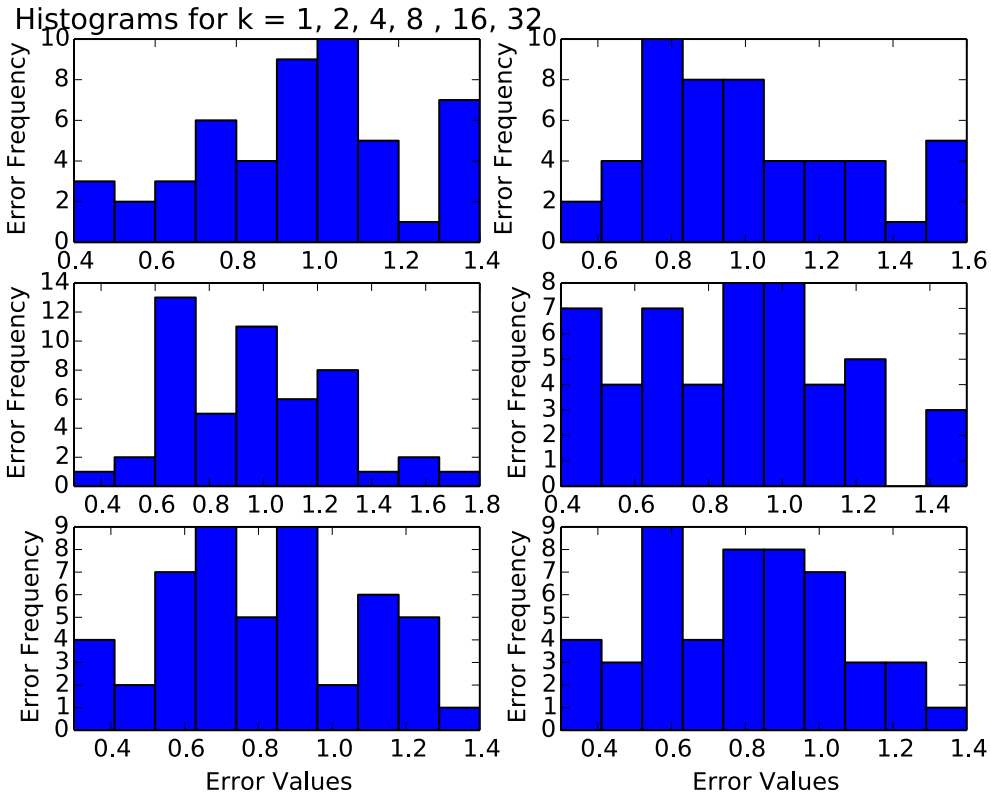
I calculated the average errors as:

Average errors: 0.83 (i = 1) and 2.953(i=1)

We see a large difference between the two. With i=1, placing 0 for missing values, there is a large error 2.95 compared to i = 0 when error is just 0.83. Also, the statistical result of t-test with p-value of 5.3e-47 and t-stat is 30.93. This p-value means the there is almost 100% confidence interval and if we consider our threshold of 0.05, p-value is almost zero. Hence, we can say that the null hypothesis does not hold. Choice of i affect user-based collaborative filtering. Yes, this also corresponds to the intuition I 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?

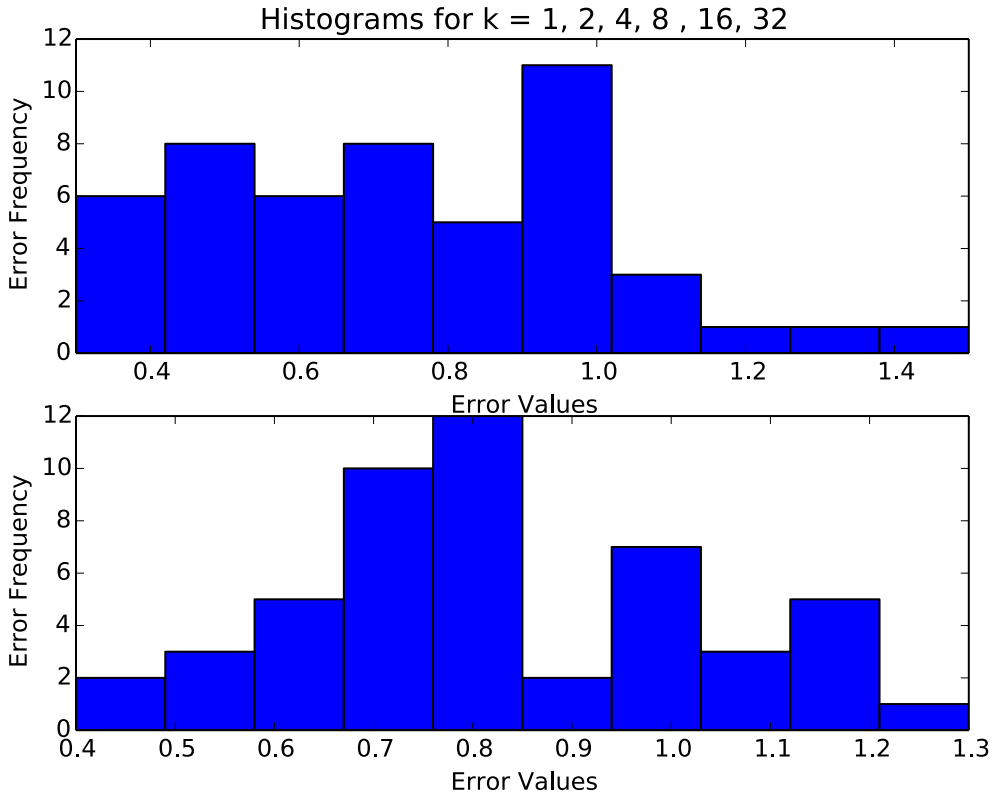
Solution:



Using the distance = 0 and i = 0, I plotted some graphs as shown in figure 6egraphs.pdf. The average errors went on decreasing from 0.908 (k=1)to 0.802(k = 32). Also, the t-stat and p-value for k = (1,32) was 2.12, 0.03. This makes the two error means statistically different with confidence interval more than 95%. So, the filter improves from k=0 to k=32. For other pairs of k, the p-value was above 0.05. p-value was 0.054 for (k=1,2), 0.45 for (k=1,4), 0.49(k=1,8) and 0.08 for (k=1,16) So, the best choice for k is 32.

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?

Solution:



I observed the following results from independent samples t-test:

Tstat = -1.75, p-value = 0.0822

I calculated the average errors as:

Average errors = 0.74(item-based), 0.826(user-based)

Here, as the p-value is 0.08, we can say with 92% confidence that the two means are different. As the mean of item-based is less than that of user-based, item-based is better than user-based. However, we can't say with 95% confident that they are different as the p-value is more than 0.05.