1. (0.5 points) Assume you've used a perceptron to learn a good decision boundary model $\mathbf{w^Tx}$ for a two-way classification problem. What is the time complexity to select a class label for a new point using this model? Give your answer as a function of the number of points, n, in the data set the decision boundary was learned from. What is the space complexity of the model, in terms of n and the number of dimensions d in the vector representing each data point? Explain your reasoning.

   Time complexity:O(d), each data only takes O(d) because each data can simply get label by feeding data to function which only takes O(d). Therefore, time complexity is O(d)

   Space complexity:O(d), each data only takes O(d) because each data has dimension d. Therefore, space complexity is O(d)

2. (0.5 points) Assume you have a K-nearest neighbor (KNN) classifier for doing a 2-way classification. Assume it uses an L^p norm distance metric (e.g. Euclidean, Manhattan, etc.). Assume a naive implementation, like the one taught to you in class. What is the time complexity to select a class label for a new point using this model? Give your answer as a function of the number of points, n, in the data set. What is the space complexity of the model, in terms of n and the number of dimensions d in the vector representing each data point? Explain your reasoning.

   Time complexity: O(n*d + K), when predicting, each new data first need to know the distance between it and other data, the distance between two data takes O(d) , n data takes O(n*d) and select the nearest K data to do predict. Therefore, time complexity is O(n*d + K)

   Space complexity: O(n + d), we need to store the distance between the new data and others, so is O(n). Then we can get the label by learning from nearest K neighbors to store the prediction O(d). therefore, space complexity is O(n+d)

3. (0.5 points) What is the time-complexity of training a KNN classifier, in terms of the number of points in the training data, n, and the number of dimensions d in the vector representing each data point? Is this greater or less than the time-complexity of training a perceptron on the same data? Explain your reasoning.

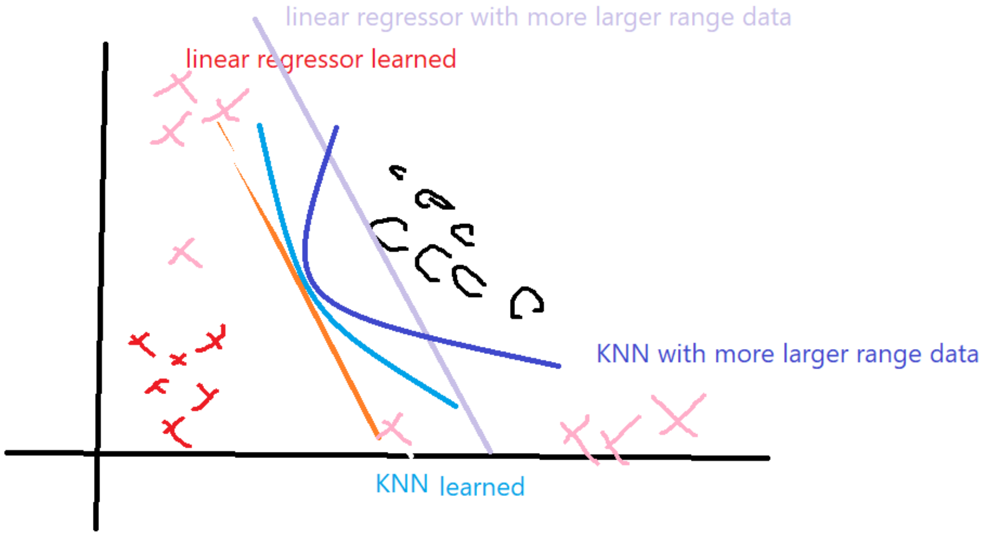
   Time complexity: O(n * d), to store n data and each data has dimension d.

KNN takes less time on training because almost everything has to be done during the prediction.

4. (0.5 points) Is a KNN classifier able to learn a non-linear decision surface without using a data transformation prior to inputting the data to the classifier? Why or why not?

   Yes, because KNN can classify non-linear separable data. KNN doesn't need to use a data transformation prior to inputting the data to the classifier in this case.

5. (0.5 points) Assume a Euclidean distance metric. Give an illustration of where the function produced by KNN regression would be very similar to that of a learned linear regressor, but very different when considered on a larger range of input values than exists in the training data. Provide a graph and an explanation.


linear regressor with more larger range data
linear regressor learned
KNN with more larger range data
KNN learned

   When input data range are small, data can be linear separable. Therefore, linear regressor is similar with KNN regression. However, when input data has a larger range, data can't be linear separable. The result will be different between KNN regression and linear regressor. In this case, we should transform data first and then do training and testing.

6. (0.5 points) We will build up a *collaborative filter* over the next few questions. The questions presume that you have implemented the `collaborative_filtering` function and passed the corresponding test

case `test_collaborative_filtering`. Collaborative filters are essentially how recommendation algorithms work on sites like Amazon ("people who bought blank also bought blank") and Netflix ("you watched blank so you might also like blank"). They work by comparing distances between users. If two users are similar, then items that one user has seen and liked but the other hasn't seen are recommended to the other user. First, frame this problem as a KNN regression problem. How are users compared to each other? Given the K nearest neighbors of a user, how can these K neighbors be used to estimate the rating for a movie that the user has not seen?
First, calculate users' distance between each other by known rating in both users. After knowing the k nearest neighbors, we can fill those unrated space with mean, mode, or median of neighbors' rating.

7. (0.5 points) The MovieLens dataset contains 100k ratings on 1682 movies given by 942 users. First, explore the MovieLens data. For each pair of users, what is the mean number of movies that two users have reviewed in common? What is the median number of movies that two reviewers have reviewed in common? How would the number of movies that two users have both reviewed affect the quality of your collaborative filter? What occurs for those movies that no user has rated?

   Mean: 5.94068470622068

   Median: 3.0

   With more number of movies that two users have both reviewed can have more features to get the similarity of two users. Therefore, we should be able to find the one similar to each other and can help the prediction more accurate.

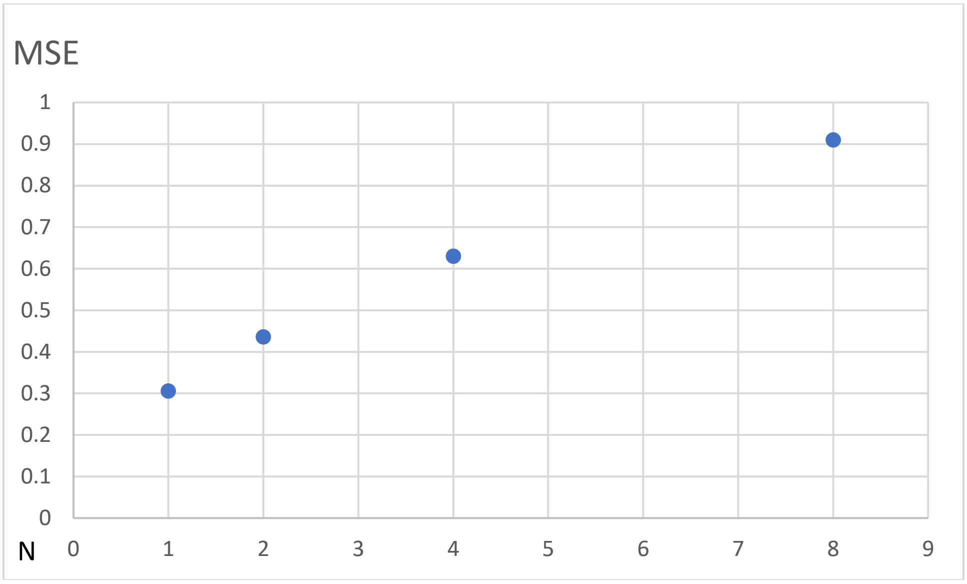
   If there is movie no user has rated, then we can't recommend to anyone. Or, we can decide to recommend it to user or nor by ourselves, because we can think users are equally interested in(or not interested in) this movie. However, this movie won't affect any other recommendation.

The following questions presume that you have implemented the `collaborative_filtering` function and passed the corresponding test case `test_collaborative_filtering`. You will now need to create additional code to let you explore the effect of different design choices on the performance of a collaborative filter. We will not evaluate this code. We will evaluate the experimental results you produce using this code. The process for this is as follows (note the free parameters `N, K, D, A`):

- For each user in MovieLens, take `N` ratings at random that are *known* (e.g. nonzero ratings) and replace them with 0. Save this altered data as a new matrix. Be sure to keep the original data around to evaluate your approach.
- Use the `collaborative_filtering` function to impute values for this new matrix with `n_neighbors = K`, `distance_measure = D`, and `aggregator = A`.
- To evaluate the quality of the collaborative filter, use mean squared error For each *known* rating that you estimated, do: (original_rating - estimated_rating) ** 2. Take the mean of all of these, and then take the square root of that. This is the mean squared error evaluation: error = sqrt(mean((r_i - rhat_i)^2)) where r_i is the original rating (before you substitued a 0) and rhat_i is the rating your system produced, and i iterates over all of the ratings you replaced with 0. Do not estimate the error on ratings that were 0 before you altered them.

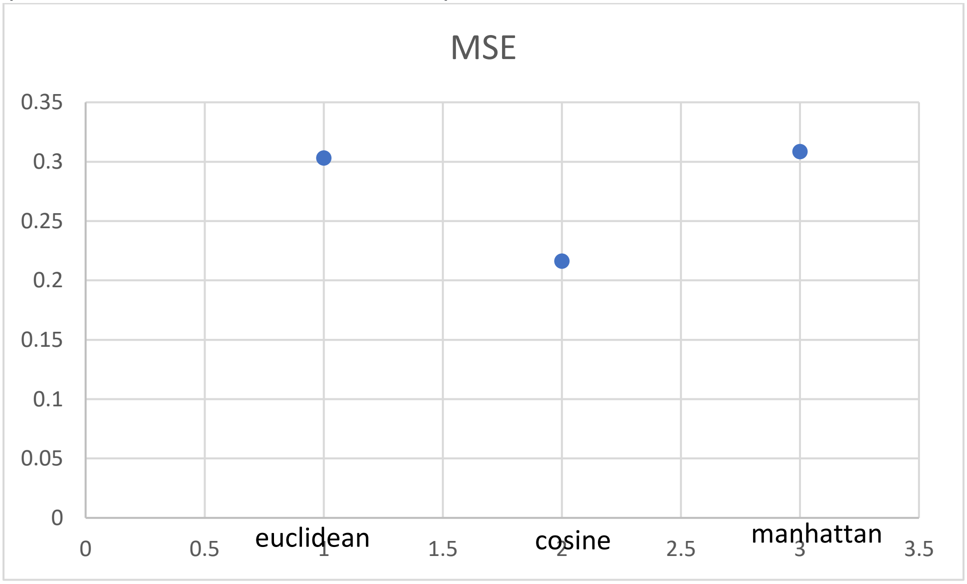
HINT: Write ONE function that takes in `(N, K, D, A)` as arguments and returns the error of a collaborative filter on the MovieLens dataset. You will be able to use this function for each of the following questions.
WARNING: Some of these plots can take a very long time to generate as the collobarative filter process is *expensive*. Start early so your computations finish before the deadline! The runtime of the solution (run on a Macbook Pro 2018 laptop) is noted at the end of each question. One tip is to first run the experiments on a small subset of the data to make sure everything works, say just the first 100 users. Once everything works, move up to the entire dataset of 943 users.

8. (0.25 points) For this question, we will measure the effect of changing `N`, the number of ratings we blank out per user. Report the error of your collaborative filter for the values `N` = 1, 2, 4, and 8. Keep `K`, `D`, and `A` fixed at 3, `'euclidean'`, and `'mean'`, respectively. Report the error via a **plot**. On the x-axis should be the value of N and on the y-axis should be the mean squared error of imputating missing values. What happens as N increases? (Estimated runtime: 95 seconds)



When N increases, MSE increases.
MSE 1 = 0.30588970922796704
MSE 2 = 0.4360959120894191
MSE 4 = 0.6299602337118468
MSE 8 = 0.9097259190422512

9. (0.25 points) For this question, we will measure the effect of changing `D`, the distance measure we use to compare users. Report the error of your collaborative filter via a **table**. For each possible distance measure, report the error of the filter. Keep `N`, `K`, and `A` fixed at 1, 3, and `'mean'`, respectively. Report the error for each possible distance measure `'euclidean'`, `'cosine'`, and `'manhattan'` distance. (Estimated runtime: 52 seconds)



euclidean MSE = 0.3032345669859837
cosine MSE = 0.21630516854892612
manhattan MSE = 0.30852460950935295

For D, cosine > `'euclidean' > 'manhattan'` . This makes sense, because using more complicated measurement considering more detail including physical distance and angle will produce a better distance measurement.

10. (0.25 points) Once you find the best performing distance measure, measure the effect of `K`. Report the error of your collaborative filter via a **plot**. On the x-axis should be the value of `K` and on the y-axis should be the mean squared error of the filter. Do this for `K` = 1, 3, 7, 11, 15, 31. Keep `N`, `D`, and `A` fixed at 1, the best performing distance measure found in the previous question, and `'mean'`, respectively. (Estimated runtime: 95 seconds)

11. (0.25 points) Finally, measure the effect of `A`, the aggregator used. Report the error of your collaborative filter via a **table**. For each possible aggregator, report the error of the filter. Keep `N`, `D`, and `K` fixed at 1, the best performing distance measure you found, and the best performing value of `K` that you found, respectively. Set A to be either `'mean'`, `'mode'`, or `'median'`. (Estimated runtime: 110 seconds).

12. (0.5 points) Now, discuss your results. What were the best settings you found for `D`, `A`, and `K`? For `D`, why do you think that distance measure worked best? For `A`, what about the best setting made it more suitable for your dataset? How did the value of `K` affect your results? Engage critically with the results of your experiments. For each graph you generated, propose an explanation for the behavior of that graph. For each table of values, propose an explanation for the variation in your results.

With higher N, which means replace more reviewed rating will increase MSE. This makes sense because the predicted rating won't be always correct. Therefore, MSE will increase.

For D, cosine > `'euclidean' > 'manhattan'` . This makes sense, because using more complicated measurement considering more detail including physical distance and angle will produce a better distance measurement.

Before overfitting, increasing K will reduce MSE. However, after reach the best MSE, the MSE will increase because of overfitting.

Aggregator choosing mean is the best, because it will consider all nearest neighbors unless there is some extremely value(have a big standard deviation). Therefore, mean is the best choice.

The best setting will be using cosine distance to be distance measurement and choosing k = 7 and aggregator chooses mean.