Assignment 1

Due on October 30, 2023 (23:59:59)

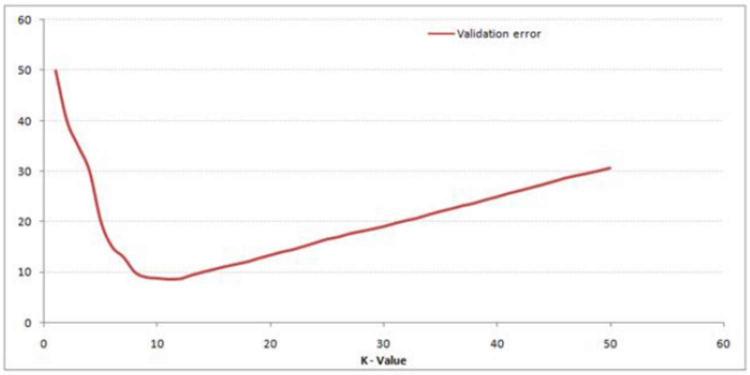
Instructions There are two parts on this assignment. The first part involves a series of theory questions and the second part involves coding. The goal of this problem set is to make you understand and familiarize with the kernel regression algorithm.

PART I: Theory Questions

k-Nearest Neighbor Classification

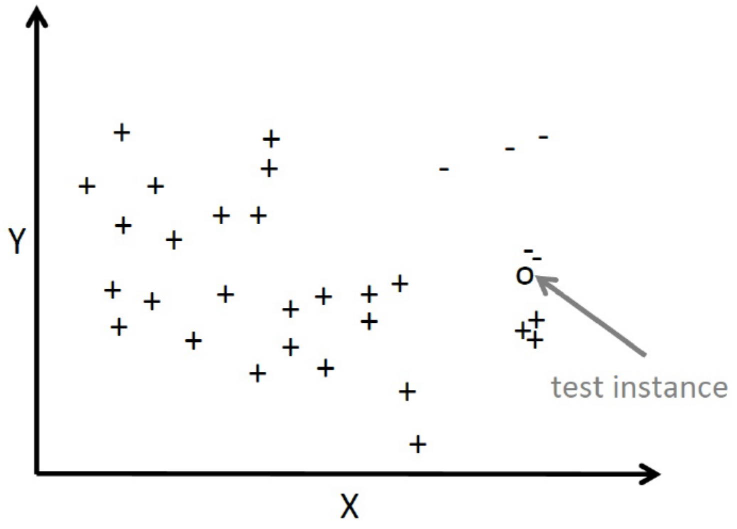
1. Assume that you have a large training dataset. Specify a disadvantage of the k-Nearest Neighbor method when using it during testing. State also your reason about your answer.

2. Considering the image below, state an optimal k-value depending on that the algorithm you are using is k-Nearest Neighbor. State also your reason behind the optimal value you preferred.



3. Assume that you have the following training set of positive (+), negative (-) instances and a single test instance (o) in the image below (Figure 1). Assume also that the Euclidean metric is used for measuring the distance between instances. Finally consider that every nearest neighbor instance affects the final vote equally.

   - What is the class appointed to the test instance for K=1? State also reason behind your answer.
   - What is the class appointed to the test instance for K=3? State also reason behind your answer.
   - What is the class appointed to the test instance for K=5? State also reason behind your answer.



4. Fill the blanks with T (True) or F (False) for the statements below:

   - If all instances of the data have the same scale then k-Nearest Neighbor's performance increases drastically. (_)
   - While k-Nearest Neighbor performs well with a small number of input variables, it's performance decreases when the number of inputs becomes large. (_)
   - k-Nearest Neighbor makes supposes nothing about the functional form of the problem it handles. (_)

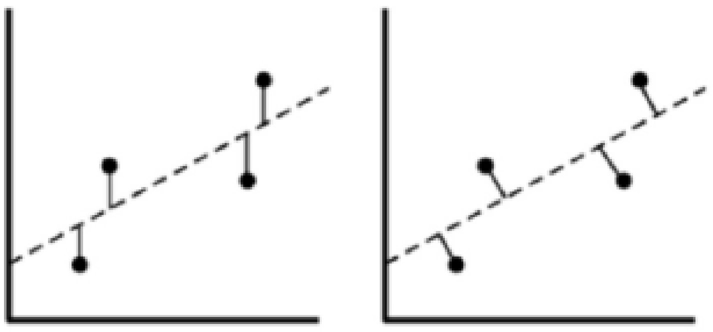
Linear Regression

1. Assume that you have five students have registered to a class and the class have a midterm and the final exam. You have obtained a set of their marks on two exams, which is in the table below:

Student	Midterm Score	(Midterm Score)^2	Final Score
1	87	7569	94
2	70	4900	72
3	92	8464	85
4	67	4489	76
5	45	2025	51

   You plan to a model which form's is $f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ for fitting the data above. The x_1 shows midterm exam score while x_2 shows square of the midterm score. Besides you plan to use feature scaling (using divide operation by the "max-min", or range, of a feature) and mean normalization. What is the normalized value of the feature $x_2^{(5)}$?

2. Considering the figure below, which of the offsets used in linear regressions least square line fit? Assume that horizontal axis represents independent variable and vertical axis represents dependent variable. State your answer with your proper explanation.



3. Considering the table below, consisting of four training examples:

x	y
1	0.5
2	1
4	2
0	0

   Assume that you are trying to fit the data above to the linear regression model $f_\theta(x) = \theta_0 + \theta_1 x_1$. Find the θ_0 and θ_1 values by using closed form solution ($\theta = \left(X^T X\right)^{-1} X^T y$). Also state dimension values of X, y, and θ matrices. Finally, show your calculations step by step.

4. State a valid reason for feature scaling and explain why it is a valid reason with respect to your reasoning.

PART II: Anime Recommendation System

In this part, you will implement a nearest-neighbor algorithm to recommend animes to viewers best suited to their tastes and traits. Similarly, your algorithm can also recommend animes to a user based on user-item ratings.

Specifically, you are going to implement a KNN algorithm to find sets of similar users based on common anime ratings and make predictions using the average rating of top-k nearest neighbors. You will also extend your implementation as a weighted k-NN algorithm.

You are provided with an anime ratings dataset of more than 49,360 ratings of 24,905 animes by 3,000 users. The ratings are on a scale between 1 to 10. Your algorithm will try to make predictions using the average rating of top-k nearest neighbors.

A dataset is provided for your training phase. The test set will be provided for final evaluation. You should use a subset of the training set to validate the performance of your model.

In other words, you should split your training dataset into two: the training set which will be used to learn model and the validation set which will be used to measure the success of your model. You can use k-fold cross-validation method which is explained in the class.

Collaborative Filtering

Collaborative filtering based systems collect and analyze users' behavioral information in the form of their feedback, ratings, preferences and activities. Based on this information, these systems then exploit similarities amongst several users/items to predict missing ratings and hence make suitable recommendations. They can discover and learn features on its own without the need of explicit features to profile items or users. Types of collaborative filtering are usually grouped into two as:

- User-based collaborative filtering: In user-based collaborative filtering, recommendations are made based on the similarity of user profiles. The idea is to find users who are similar to the target user and then recommend items that those similar users have liked. The similarity between users can be measured using various metrics, such as cosine similarity, etc.
- Item-based collaborative filtering: In item-based collaborative filtering, recommendations are made by identifying similar items to the ones the target user has shown an interest in. The algorithm looks for items that are often interacted with together by other users and suggests these related items to the user.

Dataset

The Anime dataset is collected from the MyAnimeList platform. The dataset contains 49,360 ratings of 24,905 animes by 3,000 users. The ratings are on a scale from 1 to 10. You can download the dataset from Drive link. Test data including 50 user ratings to evaluate the model.

(a) Anime 1 (b) Anime 2 (c) Anime 3 (d) Anime 4

Figure 1: Example of some animes on the dataset.

- The dataset consists of three tables: animes, user rates train and user rates test.

- Animes data include details of animes. It includes 24,905 records and 8 fields: "anime_id", "Name", "Genres", Type", "Studios", "Source", "Duration", and "Image URL".
- User rates train data include details of users rates for animes. It includes 49,360 records and 5 fields: "user_id", "Username", "anime_id", "Anime Title", and "rating".
- User rates test data include details of users rates for animes. It includes 877 records and 5 fields: "user_id", "Username", "anime_id", "Anime Title", and "rating"

Similarities

There is no limitation about similarities. You can use any similarity that you think it's proper for your regression assignment. Some similarities are listed below:

- **Cosine-based Similarity** Cosine-based Similarity In this case, two items are thought of as two vectors in m dimensional user-space. The similarity between them is measured by computing cosine angle between two vectors. Similarity between items i and j, denoted by

$$\operatorname{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2\|\vec{j}\|_2} \tag{1}$$

- **Correlation-based Similarity** Similarity between two items is measured by computing correlation $corr_{ij}$. Denoting the set of users who both rate i and j as U, the correlation similarity is given by

$$\operatorname{sim}(i,j) = \frac{\sum_{u\in U}\left(R_{u,i}-\overline{R_i}\right)\left(R_{u,j}-\overline{R_j}\right)}{\sqrt{\sum_{u\in U}\left(R_{u,i}-\overline{R_i}\right)^2}\sqrt{\sum_{u\in U}\left(R_{u,j}-\overline{R_j}\right)^2}} \tag{2}$$

- **Adjusted Cosine Similarity** Computing similarity by using basic cosine measure in item-based case has one obvious drawback, and it is the difference in rating scale between different users. We can subtract this kind of bias, so the similarity using this scheme is given by

$$\operatorname{sim}(i,j) = \frac{\sum_{u\in U}\left(R_{u,i}-\overline{R_u}\right)\left(R_{u,j}-\overline{R_u}\right)}{\sqrt{\sum_{u\in U}\left(R_{u,i}-\overline{R_u}\right)^2}\sqrt{\sum_{u\in U}\left(R_{u,j}-\overline{R_u}\right)^2}} \tag{3}$$

Steps to follow

1. Combine movie data with rating data. You can extract anime names, user names, and Image URLs from the dataset.
2. Use k-fold cross-validation for training the model (you can use k = 5).
3. Use **user-based collaborative filtering** to find similar users.

4. Calculate similarities (cosine-based, correlation-based, adjusted cosine,...) between rating vectors to find the nearest neighbors. Your kNN method can use one of the similarity methods. For that step, you need to implement the algorithms (kNN, similarity function) by yourself.

5. For a given test sample, you will try to find similar users.

6. Finally you will measure your recommender algorithm performance for each setting you have used:

   Mean Absolute Error (MAE) $= \frac{1}{n}\sum_{i=1}^{n} |d_i| - \left|\hat{d}_i\right|$ (MAE function also will written by yourself.)

   d_i is the actual rating

   $\hat{d}_i$ is the predicted rating

   n is the amount of ratings

7. You need the report the results for k values for 3, 5, and 7.

8. Analyze your results and why you get these results etc.

9. **[Extra]** You can implement an item-based collaborative filter for recommended animes.

Bonus

As a bonus task, you will also given pre-trained VGG16 visual features (Keras) on poster images of the anime dataset. You may try to combine anime posters features with tabular features to obtain better results for recommending.

Submit

You are required to submit all your code in a Jupyter notebook, along with a report in ipynb format, which should also be prepared using Jupyter notebook. The code you submit should be thoroughly commented. Your report should be self-contained and include a concise overview of the problem and the details of your implemented solution. Feel free to include pseudocode or figures to highlight or clarify specific aspects of your solution. Finally, prepare a ZIP file named name-surname-a1.zip containing:

- assignment_1.ipynb (including your report and code)
- assignment_1.py (py file version of your ipynb file)
- If you used pictures in Part I, please send the pictures as well.
- Do not send the dataset.

The ZIP file will be submitted via Google Classroom. Click here to accept your Assignment 1.

Grading

- k-NN: 20 points, Weighted k-NN: 20 points, other codes parts (well-written): 20 points
- Theory part: 12 points, Analysis of the results for prediction: 28 points.

Note: Preparing a good report is important as well as the correctness of your solutions! You should explain your choices and their effects on the results. You can create a table to report your results.

Late Policy

You may use up to four extension days (in total) over the course of the semester for the three problem sets you will take. Any additional unapproved late submissions will be weighted by 0.5. You have to submit your solution in (the rest of your late submission days + 4 days), otherwise, it will not be evaluated.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.