Assignment 1

Due on November 03, 2018 (23:59:59)

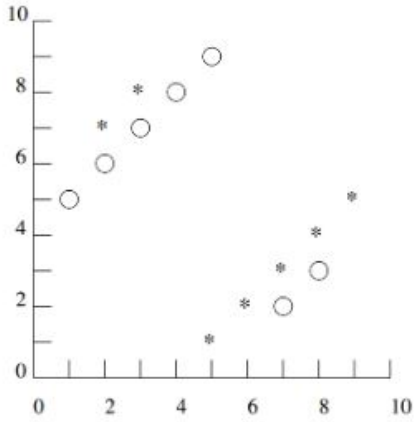
Click here to accept your Assignment 1

Instructions. There are two parts in 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 kernel regression algorithm.

Part I: Theory Questions

k-Nearest Neighbor Classification

1. Let k-NN(S) be the k Nearest Neighbor classification algorithm on sample set S, which takes the majority of the closest k points where there are 2 classes (positive and negative).
   - Show that if in both 1-NN(S_1) and 1-NN(S_2) the label of point x is positive, then in 1-NN($S_1 \cup S_2$) the label of x is positive.
   - Show an example such that in both 3-NN(S_1) and 3-NN(S_2) the label of x is positive, and in 3-NN($S_1 \cup S_2$) the label of x is negative.
2. One of the problems with k-nearest neighbor learning is selecting a value for k. Say you are given the following data set. This is a binary classification task in which the instances are described by two real-valued attributes.



   - What value of k minimizes training set error for this data set, and what is the resulting training set error? Why is training set error not a reasonable estimate of test set error, especially given this value of k?

- What value of k minimizes the leave-one-out cross-validation error for this data set, and what is the resulting error? Why is cross-validation a better measure of test set performance?
- Why might using too large values k be bad in this dataset? Why might too small values of k also be bad?
- Sketch the 1-nearest neighbor decision boundary for this dataset.

Linear Regression

1. Suppose you have m=23 training examples with n=5 features (excluding the additional all-ones feature for the bias term, which you should add).
   The closed form solution is $\theta = (X^T X)^{-1} X^T y$. For the given values of m and n, what are the dimensions of X, y, θ in this equation?
2. Suppose you have m=50 training examples which are represented with n=200,000 dimensional feature vectors. You want to use multivariate linear regression to fit paremeters θ to our data. Should you prefer gradient descent or the closed form solution?
3. Which of the following are valid reasons for using feature scaling?
   (a) It speeds up solving for θ using the normal equation.
   (b) It prevents the matrix $X^T X$ (used in the normal equation) from being non-invertable (singular/degenerate).
   (c) It speeds up gradient descent by making it require fewer iterations to get to a good solution.
   (d) It is necessary to prevent gradient descent from getting stuck in local optima.

PART II: Book Recommendation System

In this part, you will implement a nearest neighbor algorithm to recommend books to readers best suited to their tastes and traits. Similarly, your algorithm will also be able to recommend books 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 book ratings, and make predictions using the average rating of top-k nearest neighbors. You will also extend your implementation as weighted k-NN algorithm.

You are provided with a book ratings dataset of 1.1 million ratings of 27M books by 90K 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.

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
- Item-based collaborative filtering

A dataset is provided for your training phase. The test set will be provided later and announced from Piazza group. Since test set will be provided later, 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.

Dataset
Book-Crossings is a book rating dataset compiled by Cai-Nicolas Ziegler [2]. As mentioned before, it contains 1.1 million ratings of 270,000 books by 90,000 users. The ratings are on a scale from 1 to 10. You can download it from `https://drive.google.com/file/d/1qDj_Jah2LmPrsmaGrfxEQbDPGmFOD2A4/view?usp=sharing`

- The data consists of three tables: ratings, books info, and users info.
- The ratings data set provides a list of ratings that users have given to books. It includes 1,149,780 records and 3 fields: userID, ISBN, and bookRating.
- The books dataset provides book details. It includes 271,360 records and 8 fields: ISBN, book title, book author, publisher and so on.
- The user dataset provides the user demographic information. It includes 278,858 records and 3 fields: user id, location, and age.

Similarities

- There is no limitation about similarities. You can use any similarity that you think it's proper for your classification assignment. Some similarities are listed below: [1]
  - **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

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\left\|\vec{i}\right\|_2 \left\|\vec{j}\right\|_2}$$

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

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

- **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

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

Steps to follow

1. Combine book data with rating data. In order to improve computing speed, and not run into the 'MemoryError' issue, limit user data to those in the **US** or **Canada** (for **US** or **Canada** data count = 66961). You can ignore some data from dataset if it is more proper for your task. You can extract more than one data.
2. Use user-based collaborative filtering.
3. Calculate similarities (cosine-based, correlation-based, adjusted cosine,...) between rating vectors to find the nearest neighbors.
4. For a given test sample, you will try to recommend books.
5. Finally you will measure your recommender algorithm performance for each setting you have used:

   Mean Absolute Error (MAE) = $\frac{1}{n}\sum_{i=1} n|d_i| - |\hat{d}_i|$
   d_i is the actual rating
   $\hat{d}_i$ is the predicted rating
   n is the amount of ratings

Submit

You are required to submit all your code (*all your code should be written in Jupyter notebook* long with a report in ipynb format (should be prepared using Jupyter notebook). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. You can include pseudocode or figures to highlight or clarify

certain aspects of your solution. Finally, prepare a ZIP file named **name-surname-pset1.zip** containing

- report.ipynb (PDF file containing your report)
- code/ (directory containing all your codes as Python file .py)

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

NOTE: To enter the competition, you have to register kaggle in Class with your department email account. The webpage of the competition will be announced later. Top 5 assignment will earn extra points (10p).

Grading

- Code (60): k-NN: 20, Weighted k-NN: 40
- Report(40): 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 to the results. You can create a table to report your results.

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.

References

[1] Guanwen Yao and Lifeng Cai. User-based and item-based collaborative filtering recommendation algorithms design. *University of California, San Diego*, 2017.

[2] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.