CSE 158/258, Fall 2020: Midterm

Instructions

Midterm is due at 6:30pm, PST, on Tuesday November 10. Submissions should be made via gradescope.

The midterm is worth 26 marks.

You can base your solution on the stub code provided here:
`https://cseweb.ucsd.edu/classes/fa20/cse258-a/files/Midterm_stub.ipynb`
`https://cseweb.ucsd.edu/classes/fa20/cse258-a/files/Midterm_stub.html`

Submissions should take the form of a single pdf; we suggest submitting a modified printout of the stub code, though will accept other formats. **Please carefully label which pages correspond to which questions when turning in your work.**

Section 1: Regression

For our first exercise, we'll consider estimating the *length* of a review (i.e., the number of characters) based on a few simple features. The code provided in the stub estimates review length for comic book reviews on *Goodreads*, using features based on the rating, number of comments, and day of week (though the specific features are not particularly important for this exercise).

1. When introducing the Mean Squared Error (Lecture 2), we argued that it may be a poor choice of loss in datasets that have significant *outliers*. For example, when predicting the review length as in this exercise, our predictor may be dominated by a few very long reviews.

   In practice, there are several strategies we might use to handle the presence of outliers in datasets:

   (a) Simply delete outlying instances from the dataset, i.e., define some range $[y_{\min}, y_{\max}]$ and discard all instances (x, y) outside of that range.

   (b) Transform the variable y; e.g. a transformation such as $y' = \log(y)$ might be less prone to outliers.

   (c) Reframe the problem as a classification problem, rather than regression. For example, predict whether y is above or below the median value (a binary outcome) observed in the training set.

   (d) (Hard) Use an objective that is less sensitive to outliers rather than the Mean Squared Error (such as the Mean Absolute Error).

   Describe advantages and disadvantages of each approach. When considering disadvantages, it may be useful to think beyond this specific problem, and describe a hypothetical setting where (e.g.) converting the problem to classification would be poorly motivated (**4 marks**).

2. (a-d) Implement each of the four techniques from Question 1, following the starter code. For each, briefly explain:

   - Your reasoning behind any model choices you make (e.g. how do you choose $y_{\max}$ in Question 1(a)).
   - How you evaluate the quality or measure the performance of your proposed solution.

   Your code should document these choices and report the performance measures that you select. Note that some problems are hard (e.g. Question 1(d)); you may still describe evaluation strategies even if you are unable to implement a working solution for partial marks (**8 marks**).

3. (Theory) In class (Lecture 2) we argued that the best possible trivial predictor $y = \theta_0$ in terms of the Mean Squared Error was found by taking $\theta_0 = \bar{y}$ (i.e., the mean value of the label). Show that when optimizing the *Mean Absolute Error* with a trivial predictor, i.e.,

   $$\frac{1}{N} \sum_i |\theta_0 - y_i|,$$

   that the best possible trivial predictor is found by taking θ_0 as the *median* value of y (**1 mark**).

4. (Design) Consider building a prediction pipeline to estimate tip amounts from taxicab trips. This task has sometimes been attempted for Assignment 2, e.g. using open data from trips in NYC.[1] Here we'll consider designing a pipeline for this task. You may provide each of your answers as plain text. (**3 marks**)

   (a) Suggest what features might be useful for prediction (and could realistically be collected). Consider, for example, features associated with temporal and geographical information.

   (b) How should the above features be represented or transformed? For example, how can the timestamp be represented to capture variation at the level of time of day (etc.); how might you represent (and extract features from) the start and end locations? Are there any useful derived features (i.e., transformations or combinations of existing features) that are useful for prediction?

   (c) How might you set up the problem as a predictive task? Would you cast the task as regression or classification? Would it be useful to transform output variable (e.g. as in Question 1(b))? Discuss the merits of a few alternatives.

[1] `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

Section 2: Classification

In the classification lectures (Lecture 3 and Lecture 4), we developed various classification schemes that make predictions of the form:

$$y_i = \begin{cases} 1 & \text{if } x_i \cdot \theta > 0 \\ \text{-1} & \text{otherwise} \end{cases} \tag{1}$$

(for example, *logistic regression* was an example of such a classifier).[2]

A naïve solution to developing a classifier might be to simply train a regressor directly. That is, given a classification problem with labels $y_i \in \{-1, 1\}$, just use ordinary regression (as in Lectures 1 and 2) to estimate θ. Then at test time use Equation 1 to estimate the labels.

5. Argue why this form of classifier is unlikely to work as well as the types of classification scheme we designed in class. If useful, feel free to include a simple diagram or similar to support your argument (**2 marks**).

6. Perform a simple experiment to demonstrate that this naïve model does not work as well as logistic regression. That is, select a dataset (which could be e.g. the dataset from Section 1, or any homework exercise), a few features, a label to predict, and an appropriate classifier evaluation metric, to show that the naïve classifier is outperformed by logistic regression (**2 marks**).

Section 3: Recommender Systems

7. In Lecture 7 we introduced simple recommender systems based on the notions of item-to-item and user-to-user similarity. Furthermore, we showed how they can be used for rating prediction. One such model makes predictions by

$$r(u, i) = \frac{\sum_{j \in I_u} R_{u,j} \cdot \text{Sim}(i, j)}{\sum_{j \in I_u} \text{Sim}(i, j)}, \tag{2}$$

where $R_{u,j}$ is the rating u gave to item j (note that when evaluating, i itself should not be included in the summation).

Code for this solution is in the stub and is covered in more depth on `http://cseweb.ucsd.edu/classes/fa20/cse258-a/code/workbook4.html`. Note that the code only evaluates accuracy on a sample of 1,000 ratings for efficiency reasons; you are free to modify or maintain this sample size.

There are several potential variants of this model, some of which may work better on a particular dataset. Some examples include:

- Interchange users and items in the above equation. That is, predict ratings in terms of user-to-user similarity $\text{Sim}(u, v)$ rather than item-to-item similarity.
- Use a different similarity measure other than the Jaccard similarity (as in the stub code). In principle you could use *any* item-to-item or user-to-user similarity function, including one specifically designed for this dataset.
- You could first subtract the mean rating, so that you are weighting deviations from the mean, i.e.,

$$r(u, i) = \bar{R}_i + \frac{\sum_{j \in I_u} (R_{u,j} - \bar{R}_j) \cdot \text{Sim}(i, j)}{\sum_{j \in I_u} \text{Sim}(i, j)},$$

where $\bar{R}_i$ is the average rating for item i.

Implement three model variants (either following the three ideas above, or some combinations of the above). Document the choices you made for each, and report their performance. **At least one of your variants should outperform the model of Equation 2.** (**6 marks**)

[2] Note that for the purposes of this problem, our labels are -1 and 1, rather than 0 or 1, though the basic idea is unchanged.