CSE 158/258, Fall 2023: Midterm

Instructions

Midterm is due at 9am, on **Wednesday November 8**. Submissions should be made via gradescope.

The midterm is worth 25 marks.

You can base your solution on the stub code provided here:
`https://cseweb.ucsd.edu/classes/fa23/cse258-a/files/Midterm_stub.ipynb`
`https://cseweb.ucsd.edu/classes/fa23/cse258-a/files/Midterm_stub.html`

You should submit two files:

`answers_midterm.txt` should contain a python dictionary containing your answers to each question. Its format should be like the following:

```
{ "Q1": 1.5, "Q2": [3,5,17,8], "Q2": "b", (etc.) }
```

The provided code stub demonstrates how to prepare your answers and includes an answer template for each question.

`midterm.py` A python file containing working code for your solutions. The autograder *will not execute your code*; this file is required so that we can assign partial grades in the event of incorrect solutions, check for plagiarism, etc. Your solution should **clearly document which sections correspond to each question and answer**. We may occasionally run code to confirm that your outputs match submitted answers, so **please ensure that your code generates the submitted answers.**

All questions in this midterm will make use of the video game review data from *Steam*: `https://cseweb.ucsd.edu/classes/fa23/cse258-a/files/train.json.gz`

Each data point consists of a record of a video game review. A few relevant fields include:

`userID`, `gameID` The ID of the user and the game.

`hours` The amount of time the user played the game.

`hours_transformed` $log_2(\texttt{hours} + 1)$; i.e., the above, log-transformed.

`text` The text of the user's review.

Note: none of these experiments should require more than 1-2 seconds to run on modest hardware. Please only use a smaller fraction if absolutely stuck; we may only give partial credit if we cannot verify the correctness of your solution.

Section 1: Regression

1. Fit a linear regressor of the form

   $$\texttt{review\_length} = \theta_0 + \theta_1 \dot{(}\texttt{hours})$$

   where the review length is the number of characters in the review (i.e., `len(d['text'])`) and 'hours' is the original (i.e., not transformed) hours variable.

   Report the value of θ_1 and the Mean Squared Error of the prediction (2 marks).

2. Incorporate several *transforms* of the 'hours' variable into your model: fit

   $$\texttt{review\_length} = \theta_0 + \theta_1(\texttt{hours}) + \theta_2 \log_2(\texttt{hours} + 1) + \theta_3 \sqrt{(}\texttt{hours}) + \theta_4 \delta(\texttt{hours} > \mathit{median}),$$

   where $\delta(\texttt{hours} > \mathit{median})$ is a binary indicator indicating whether 'hours' is above (1) or below (0) the median number of hours played across the entire dataset.

   Report the MSE of your predictor (1 mark).

3. Fit a model that uses a sequence of binary indicators of the form:

   $$\texttt{review\_length} = \theta_0 + \theta_1\delta(h > 1) + \theta_2\delta(h > 5) + \theta_3\delta(h > 10) + \theta_4\delta(h > 100) + \theta_5\delta(h > 1000)$$

   (where h is shorthand for the number of hours played).

   Report the MSE of your predictor (1 mark).

4. Modify your predictor to estimate

   $$\texttt{hours} = \theta_0 + \theta_1 \dot{(}\texttt{review\_length}).$$

   Report both the MSE and the Mean Absolute Error (MAE) of your predictor. Briefly comment on which of MSE or MAE might be more suitable based on the characteristics of this dataset (2 marks).

5. Fit a model which predicts

   $$\log_2(\texttt{hours} + 1) = \theta_0 + \theta_1 \dot{(}\texttt{review\_length})$$

   (i.e., the transformed version of the hours variable). Compute:

   - The MSE of this model (i.e., the MSE when trying to estimate the transformed predictions);
   - The MSE of the model compared to the original (i.e., untransformed, from Question 4) 'hours' values, by transforming the model's predictions back into the original scale (i.e., by inverting the $\log_2(x + 1)$ transform)

   (2 marks).

6. Implement a validation pipeline. 50%/25%/25% train/validation/test splits are provided in the stub. Fit a model that predicts the length of the review text (i.e., as in Question 1); your feature should be a *one-hot encoding* of the hours played. Your one-hot encoding should have 100 dimensions corresponding to (rounded down) integer values of the number of hours played (e.g. if the user played less a game for less than 1 hour, the first dimension of your one-hot encoding would be 1); if a user played a game for more than 99 hours, the last entry should be 1.

   Fit a regularized regression model (e.g. `sklearn.linear_model.Ridge`) with regularization strengths of $\alpha \in \{1, 10, 100, 1000, 10000\}$.

   Report (a) the best value of the regularization parameter; (b) the corresponding model's MSE on the validation set; its MSE on the test set (4 marks).

Section 2: Classification

7. Compute (a) the median value of the `hours_transformed` variable; and (b) the number of interactions with less than one hour played (1 mark).

8. Fit a logistic regressor that estimates whether the (transformed) number of hours played (`hours_transformed`) is above or below the median value using a single feature based on the review length (i.e., `len(d['text'])`). Fit a *balanced* classifier, setting any other parameters to default values.

   Report (a) the number of True Positives; (b) the number of True Negatives; (c) the number of False Positives; (d) the number of False Negatives; and (e) the Balanced Error Rate (BER) of the classifier (2 marks).

9. Compute the *precision@k* of the above classifier for $k \in \{5, 10, 100, 1000\}$. Instead of our normal procedure for computing the precision@k, we must be careful in this case to avoid 'ties,' since multiple data points may have identical features (and therefore identical predictions) in spite of having different labels. When there are multiple values tied at a particular value of k, you must

   - Compute the corresponding decision threshold for that value of k (e.g. the probability score of the classifier);
   - Compute the precision for all entries whose confidence matches or exceeds that threshold.

   In other words, you must increase k by enough to account for all tied values (2 marks).

10. An alternative strategy to training a classifier as we did above would be to simply to train a regressor (much as we did in Question 5), and threshold its output directly (i.e., returning 'True' if the regressor predicts above a certain threshold). See if you can find a threshold on the regression model from Question 5 that has a lower Balanced Error Rate than the model from Question 8. Report the threshold you selected and the corresponding BER (2 marks).

Section 3: Recommendation

For these questions you should use the first 90% of samples as a training set and the last 10% as a test set (though there are no models to fit). Code to split the data is provided in the stub.

11. Our goal in the following questions is to recommend games that a user is likely to play for a long amount of time. We will treat games played above the median amount of time as positives (label 1) and other games as negatives (label 0).

    First, compute the per-user and per-item median values, *using only entries from the training set.* Report the item and user medians for the item/user in the first entry in the training set (1 mark).[1]

12. Fit a trivial hand-crafted model that given a user/item pair (u, i) does the following:

    - Returns 1 if the median time played for item i is above the global median (i.e., the median across all interactions);

[1]The autograder should work whether you use `hours` or `hours_transformed`, as long as you are consistent throughout.

- If the item hasn't been seen before, returns 1 if the time played for user u is above the global median;
- returns 0 otherwise.

Report its accuracy on the test set (1 mark).

You may use the *entire dataset* for the final three questions:

13. Compute the 10 items most similar to first item in the dataset (i.e., the item from the first entry) in terms of Jaccard similarity. Report the Jaccard similarity of the first and tenth most similar items (2 marks).
14. Instead of the Jaccard similarity, compute *Cosine* similarities where each user/item has a label of 1 if the game was played more than the median amount of time and -1 otherwise. Again report the Cosine similarities of the first and 10th most similar items (1 mark).
15. Repeat the above question, but instead of using a binary indicator, compute the Cosine similarity using the `hours_transformed` values directly. Again report the Cosine similarities of the first and 10th most similar items (1 mark).