CSC 311 Winter 2024 Final Project

Deadline: Friday, April 5, 2024, at 11:59 pm

Please submit the following files on MarkUs:

- `project_report.pdf` contains your answers to Part A and B. Make sure that the file is readable. We highly recommend typesetting the file using LaTeXor Microsoft Word, scanner).
- `project_code.zip` contains the Python codes you used for both Part A and B. You may exclude the folder `/data` from the starter code.

You should form teams of 2-4 students. In your final report, **please describe each team member's contributions in at most one paragraph**.

1 Introduction

One of CSC311's main objectives is to prepare you to apply machine learning algorithms to real-world tasks. The final project aims to help you get started in this direction.

You will be performing the following tasks:

- Implement several algorithms to solve a real-world problem.
- Modify an existing algorithm to improve its performance.
- Write a short report analyzing the result.

The final project is not intended to be a stressful experience. It is a good chance for you to experiment, think, play, and hopefully have fun. These tasks are what you will be doing daily as a data analyst/scientist or machine learning engineer.

2 Background and Task

Online education services, such as Khan Academy and Coursera, provide a broader audience with access to high-quality education. On these platforms, students can learn new materials by watching a lecture, reading course material, and talking to instructors in a forum. However, one disadvantage of the online platform is that it is challenging to measure students' understanding of the course material. To deal with this issue, many online education platforms include an assessment component to ensure students understand the core topics. The assessment component often consists of diagnostic questions, each a multiple-choice question with one correct answer. The diagnostic question is designed so that each incorrect answer highlights a common misconception. An example of the diagnostic problem is shown in figure 1. When students incorrectly answer the diagnostic question, it reveals the nature of their misconception, and by understanding these misconceptions, the platform can offer additional guidance to help resolve them.

In this project, you will build machine learning algorithms to predict whether a student can correctly answer a diagnostic question based on the student's previous answers to other questions and other students' responses. Predicting the correctness of students' answers to unseen diagnostic questions helps estimate the student's ability level in a personalized education platform. Moreover, these predictions form the groundwork for many advanced customized tasks. For instance, using

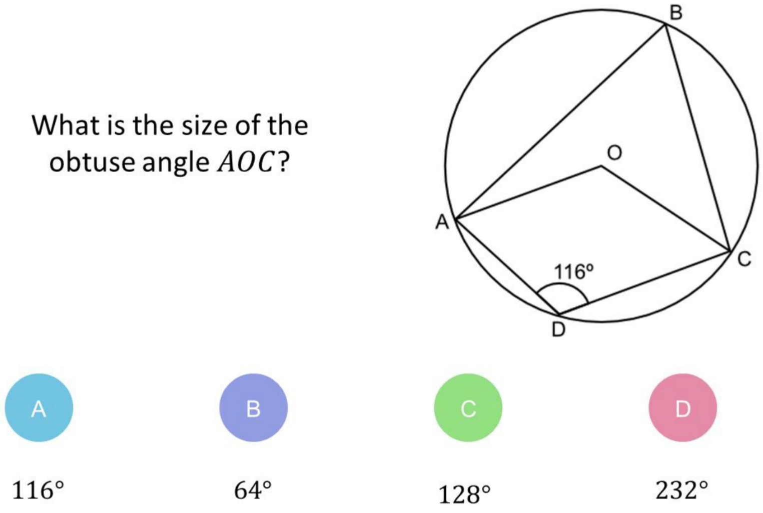

Figure 1: An example diagnostic question [1].

the predicted correctness, the online platform can automatically recommend a set of diagnostic questions of appropriate difficulty that fit the student's background and learning status.

You will begin by applying a few machine learning algorithms you learned in this course, comparing their performances, and analyzing their advantages and disadvantages. Next, you will modify existing algorithms to predict students' answers more accurately. Lastly, you will experiment with your modification and write a short report with the results.

You will measure the performance of the learning system in terms of prediction accuracy. However, you are welcome to include other metrics in your report if you believe they provide additional insight:

$$\text{Prediction Accuracy} = \frac{\text{The number of correct predictions}}{\text{The number of total predictions}}$$

3 Data

We sub-sampled the answers of 542 students to 1774 diagnostic questions from the dataset provided by Eedi[1], an online education platform that is currently being used in many schools [1]. The platform offers crowd-sourced mathematical diagnostic questions to students from primary to high school (between 7 and 18 years old). The truncated dataset is provided in the folder `/data`.

3.1 Primary Data

`train_data.csv` is the primary data set you will use to train the learning algorithms. `valid_data.csv` is the validation set you can use to tune hyper-parameters or perform model selection. Finally, `test_data.csv` is the test set you should use to report the performance of the final model.

Each of the three `csv` data files has 3 columns:

[1] https://eedi.com/

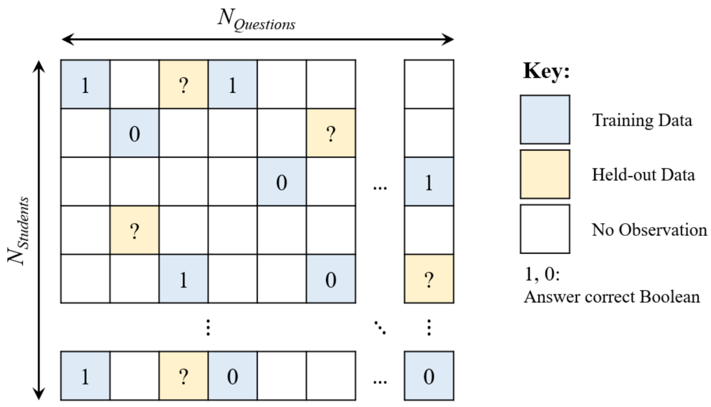

Figure 2: An example sparse matrix [1].

- **question_id:** ID of the question answered (starts from 0).
- **user_id:** ID of the student who answered the question (starts from 0).
- **is_correct:** Binary indicator whether the student's answer was correct (0 is incorrect, 1 is correct).

We also provide a sparse matrix, `train_sparse.npz`, where each row corresponds to the **user_id** and each column corresponds to the **question_id**. An illustration of the sparse matrix is shown in figure 2. The correct answer given a pair of (**user_id**, **question_id**) will have an entry 1, and an incorrect answer will have an entry 0. Answers with no observation and held-out data (that will be used for validation and test) will have an entry `NaN` (`np.NaN`).

3.2 Question Metadata

We also provide the question metadata, `question_meta.csv`, which contains the following columns:

- **question_id:** ID of the question answered (starts from 0).
- **subject_id:** The subject of the question covered in an area of mathematics. The text description of each subject is provided in `subject_meta.csv`.

3.3 Student Metadata

Lastly, we provide the student metadata, `student_meta.csv`, that has the following columns:

- **user_id:** ID of the student who answered the question (starts from 0).
- **gender:** Gender of the student, when available. 1 indicates a female, 2 indicates a male, and 0 indicates unspecified.
- **data_of_birth:** The student's birth date, when available.
- **premium_pupil:** Student's eligibility for free school meals or pupil premium due to being financially disadvantaged, when available.

4 Part A (20 marks)

In the first part of the project, you will implement and apply various machine learning algorithms in the course to predict the correctness of a student's answer to a diagnostic question.

For this part, you will only be using the primary data: `train_data.csv`, `train_sparse.npz`, `valid_data.csv`, and `test_data.csv`. Moreover, you may use the helper functions provided in `utils.py` to load the dataset and evaluate your model. You may also use any functions from packages `NumPy`, `Scipy`, `Pandas`, and `PyTorch`. Make sure you understand the code instead of using it as a black box.

1. **(5 marks) Collaborative filtering with k-Nearest Neighbor**

   In this part, you will implement collaborative filtering using the k-Nearest Neighbor (kNN) algorithm. The starter code is in `part_a/knn.py`.

   The `knn_impute_by_user` function in `knn.py` implements **user-based collaborative filtering**. Given student A, kNN finds another student B who is most similar to student A in terms of the correctness of their answers and predicts student A's correctness using student B's correctness. The core underlying assumption is that if student A has the same correct and incorrect answers on other diagnostic questions as student B, A's correctness on specific diagnostic questions matches that of student B.

   (a) Complete the `main` function in `knn.py` to run user-based collaborative filtering.

   Run user-based collaborative filtering for $k \in \{1, 6, 11, 16, 21, 26\}$. Plot the validation accuracy as a function of k.

   Report the k value you selected and the test accuracy of the final model.

   (b) Implement the function `knn_impute_by_item` to perform **item-based collaborative filtering**. Given question A, kNN finds the closest question B that was answered similarly and predicts the correctness of answering question A based on the correctness of answering question B.

   State the underlying assumption on item-based collaborative filtering.

   (c) Repeat part (a) with item-based collaborative filtering.

   (d) Which one of user-based and item-based collaborative filtering is better? Why?

   (e) List two potential limitations of kNN for the task you are given.

2. **(7 marks) The Item Response Theory Model**

   In this part, you will implement an Item-Response Theory (IRT) model. The starter code is in `part_a/item_response.py`.

   In the one-parameter IRT model, student i has an ability θ_i and question j has the difficulty β_j. Then, the probability that the student i answers question j correctly is given by:

   $$p(c_{ij} = 1|\theta_i, \beta_j) = \frac{\exp(\theta_i - \beta_j)}{1 + \exp(\theta_i - \beta_j)}$$

   (a) Derive the log-likelihood $\log p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ for all students and questions, where $\mathbf{C}$ is the sparse matrix.

   Then, derive the derivative of the log-likelihood with respect to θ_i and β_j
   (Hint: recall the derivative of the logistic model with respect to the parameters).

   (b) Implement all the missing functions in `item_response.py`.

   (c) Tune the hyperparameters (the learning rate and number of iterations) and report the hyperparameters you selected.

   With your chosen hyperparameters, plot the training and validation log-likelihoods as a function of iteration.

   Report the validation and test accuracies for your final model.

   (d) Select three questions j_1, j_2, and j_3. For each question, plot a curve showing the probability of the correct response $p(c_{ij} = 1)$ as a function of the ability θ using the trained $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. Put all three curves on the same figure.

   Comment on the shapes of the curves and briefly describe what these curves represent.

3. **(8 marks) Matrix Factorization OR Neural Networks.** Please read both options below, but only need to complete one of the two.

   (i) **Option 1: Matrix Factorization.**

   In this part, you will implement matrix factorization methods.
   The starter code is located at `part_a/matrix_factorization.py`.

   (a) We have provided the function `svd_reconstruct`, which factorizes a sparse matrix using singular-value decomposition (SVD).

   Run SVD with at least 5 different k values and report the k value you selected.

   Report the validation and test accuracies of the final model with your chosen k value.

   (b) State one limitation of SVD for this task.
   (Hint: how are you treating the missing entries?)

   (c) Implement the functions `als` and `update_u_z` to perform alternating updates. You need to use Stochastic Gradient Descent (SGD) to update $\mathbf{u}_n \in \mathbb{R}^k$ and $\mathbf{z}_m \in \mathbb{R}^k$ as described in the docstrings. Note that this is **different** from the alternating least squares (ALS) we introduced in the lecture slides, where we used the direct solution. As a reminder, the objective is as follows:

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(n,m)\in O} \left(C_{nm} - \mathbf{u}_n^\top \mathbf{z}_m\right)^2,$$

where $\mathbf{C}$ is the sparse matrix and $O = \{(n, m) : \text{entry } (n, m) \text{ of matrix } \mathbf{C} \text{ is observed}\}$.

(d) Learn the representations $\mathbf{U}$ and $\mathbf{Z}$ using ALS with SGD. Tune the hyperparameters (k, the learning rate, and the number of iterations). Make sure to try at least 5 different k values. Report the hyperparameters that you selected.

(e) With your chosen hyperparameters, plot the training and validation squared-error losses as a function of iteration. Also, report the validation accuracy and test accuracies for your final model.

(ii) **Option 2: Neural Networks.**

In this part, you will design an autoencoder model. Given a user $\mathbf{v} \in \mathbb{R}^{N_{\text{questions}}}$ from a set of users $\mathcal{S}$, our objective is:

$$\min_{\boldsymbol{\theta}} \sum_{\mathbf{v}\in\mathcal{S}} \|\mathbf{v} - f(\mathbf{v}; \boldsymbol{\theta})\|_2^2,$$

where f is the reconstruction of the input $\mathbf{v}$. The network computes the following function:

$$f(\mathbf{v}; \boldsymbol{\theta}) = h(\mathbf{W}^{(2)} g(\mathbf{W}^{(1)}\mathbf{v} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \in \mathbb{R}^{N_{\text{questions}}}$$

where g and h are the logistic (sigmoid) activation function.

Here, $\mathbf{W}^{(1)} \in \mathbb{R}^{k\times N_{\text{questions}}}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{N_{\text{questions}}\times k}$, where $k \in \mathbb{N}$ is the latent dimension. The starter code in `PyTorch` is at `part_a/neural_network.py`.

(a) Describe at least three differences between ALS and neural networks.

(b) Complete the `AutoEncoder` class to perform a forward pass of the autoencoder.

(c) Train the autoencoder for $k \in \{10, 50, 100, 200, 500\}$, and tune other hyperparameters (the learning rate and number of iterations). Report the hyperparameters that you selected.

(d) With your chosen hyperparameters, plot the training and validation objectives as a function of epoch. Also, report the test accuracy for the final model.

(e) Modify the function `train` so that the objective adds the L_2 regularization. The objective is as follows:

$$\min_{\boldsymbol{\theta}} \sum_{\mathbf{v}\in\mathcal{S}} \|\mathbf{v} - f(\mathbf{v}; \boldsymbol{\theta})\|_2^2 + \frac{\lambda}{2}(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2)$$

You may use the method `get_weight_norm` to obtain the regularization term.

Using the k and other hyperparameters selected from part (d), tune the regularization penalty $\lambda \in \{0.001, 0.01, 0.1, 1\}$.

With your chosen λ, report the validation and test accuracies of the final model. Does your model perform better with the regularization penalty?

5 Part B (30 marks)

In the second part of the project, you will modify one algorithm you implemented in part A to hopefully predict the correctness of students' answers to the diagnostic question with higher accuracy. In particular, consider the results obtained in part A, analyze the factors limiting the performance of one of the methods (e.g., overfitting, underfitting, or optimization difficulties) and propose a modification to the algorithm which could help address this limitation. Rigorously test the performance of your modified algorithm, and write up a report summarizing your results as described below.

You will **not** be graded on how well the algorithm performs (i.e., accuracy); your grade will be based on the quality of your analysis. Try to be creative! You may also use the provided metadata (`question_meta.csv` and `student_meta.csv`) and any third-party ideas or code that are publicly available. You must cite any work that is not your own in your report.

Your report for part B should be **3-4 pages** long. Don't be afraid to keep the text short and include large illustrative figures. The guidelines and marking schemes are as follows:

1. **(7 marks) Formally Describe Your Modified Algorithm:**

   Define how you modified the algorithm, possibly by providing equations and/or a precise description of the steps. Describe why one would expect your modified algorithm to perform better than the ones in part (a). For instance, did you design it to improve the optimization, reduce over-fitting, etc.?

2. **(5 marks) Figure or Diagram:**

   Use figures or diagrams to illustrate your modified algorithm. The goal is to make your report accessible, especially to readers who are starting by skimming it.

3. **(8 marks) Your Model versus Baseline Models:**

   Compare your model with the baseline models in part (a). Does your model perform better or worse than the baseline ones? Make sure to compare your model and the baseline models by testing them rigorously. Show the comparison using a table or a plot.

4. **(5 marks) Explain Your Model's Performance:**

   If your model performed better than the baseline models, try to explain why. For example, you can try to disentangle whether the benefits are due to optimization or regularization.

   If your model performed worse than the baseline models, try to explain why. Propose one idea that may improve your model's performance.

5. **(5 marks) Analyze One Limitation of Your Model:**

   State one limitation of your model that you haven't mentioned previously. Describe one setting where we expect your model to perform poorly. Try to explain why this limitation exists. Give one idea of a possible modification to address this limitation.

6 Tips for Success

- **Read carefully!** Read this document carefully. Ask questions on Piazza and visit office hours if you are unsure of any requirements.
- **Be honest!** You are not being marked on how good the results are. It doesn't matter if your method is better or worse than the ones you compare to. What matters is that you clearly describe the problem, your method, what you did, and the results. Just be scientific.
- **Be careful!** Don't do things like evaluating your model on the training data, choosing hyperparameters using test accuracy, comparing unfairly against other methods, including plots with unlabeled axes, using undefined symbols in equations, etc. Do sensible crosschecks like running your algorithms several times to understand the between-run variability, performing gradient checking, etc.

References

[1] Wang, Zichao, et al. *Diagnostic Questions: The NeurIPS 2020 Education Challenge.* arXiv preprint arXiv:2007.12061 (2020)