Final Project Instructions

Deadline: April 16th, at 11:59pm

Submission: Through MarkUs.

Late Submission: 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Collaboration: You should form teams of 1-3 students. Your final report should list the contributions of each team member.

Two Project Format Options:

Because this course is the graduate version of an intro course, we want to support both those who want guided, hands-on experience, and those who want support doing machine learning research. To support students in both situations, we are allowing each group the choice between two project formats:

- The default project: This project is designed to give you hands-on experience using some of the methods covered in the course, with a bit of research and exploration on top. The default project takes most of the guesswork out of choosing a doable and educational set of tasks.
- An open-ended project: This project format is designed to support a machine learning research agenda. This format requires you to choose your own research idea, and requires more planning and research. This route is probably harder, but has two extra chances for feedback: a project proposal, and project presentations. This extra feedback is to support you and make sure your project doesn't go off the rails.

The two projects formats have different grading rubrics, detailed below. You can change your mind about which type of project you'd like to do at any time. However, for the open-ended project format only, you have to give a presentation on April 9th.

1 Default Project

1.1 Introduction

One of CSC2515's objectives is to prepare you to apply machine learning algorithms to real-world tasks. The default project aims to help you get started in this direction. You will be performing the following tasks:

- Try out existing algorithms to real-world tasks.
- Modify an existing algorithm to improve performance.
- Write a short report analyzing the result.

1.2 Background & 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 that students understand the core topics. The assessment component is often composed of diagnostic questions, each a multiple choice question with one correct answer. The diagnostic question is designed so that each of the incorrect answers 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.

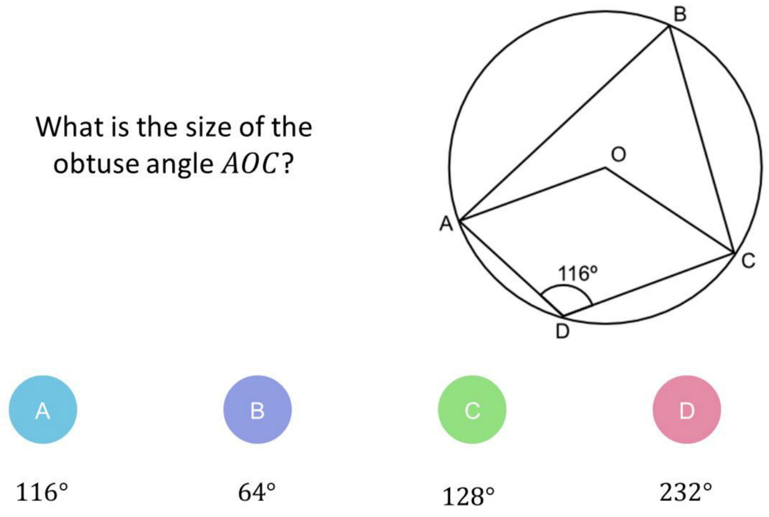

Figure 1: An example diagnostic question [1].

In this project, you will build machine learning algorithms to predict whether a student can correctly answer a specific diagnostic question based on the student's previous answers to other questions and other students' responses. Predicting the correctness of students' answers to as yet 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 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 existing machine learning algorithms you learned in this course. You will then compare the performances of different algorithms and analyze their advantages and disadvantages. Next, you will modify existing algorithms to predict students' answers with higher accuracy. Lastly, you will experiment with your modification and write up a short report with the results.

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

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

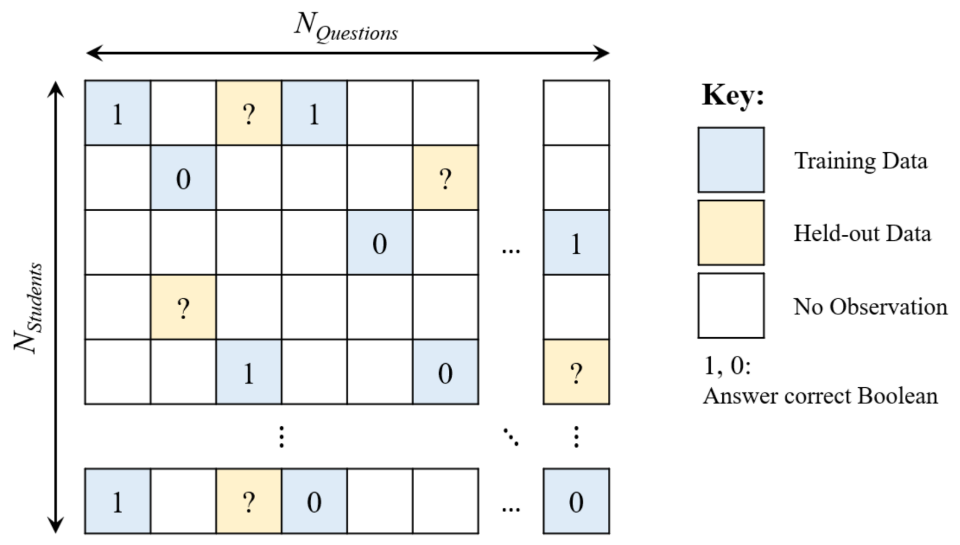

Figure 2: An example sparse matrix [1].

1.3 Data

We subsampled 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`.

1.3.1 Primary Data

The primary data, `train_data.csv`, is the main dataset you will be using to train the learning algorithms throughout the project. There is also a validation set `valid_data.csv` that you should use for model selection and a test set `test_data.csv` that you should use for reporting the final performance. All primary data `csv` files are composed of 3 columns:

- **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, `sparse_matrix.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`).

1.3.2 Question Metadata

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

[1] https://eedi.com/

- **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`.

1.3.3 Student Metadata

Lastly, we provide the student metadata, `student_meta.csv`, that is composed of 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:** Birth date of the student, when available.
- **premium_pupil:** Student's eligibility for free school meals or pupil premium due to being financially disadvantaged, when available.

1.4 Part A

In the first part of the project, you will implement and apply various machine learning algorithms you studied in the course to predict students' correctness of a given diagnostic question. Review the course notes if you don't recall the details of each algorithm. For this part, you will only be using the primary data: `train_data.csv`, `sparse_matrix.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. **[5pts] k-Nearest Neighbor.** In this problem, using the provided code at `part_a/knn.py`, you will experiment with k-Nearest Neighbor (kNN) algorithm.

   The provided kNN code performs collaborative filtering that uses other students' answers to predict whether the specific student can correctly answer some diagnostic questions. In particular, the starter code implements user-based collaborative filtering: given a user, kNN finds the closest user that similarly answered other questions and predicts the correctness based on the closest student'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 a function `main` located at `knn.py` that runs kNN for different values of $k \in \{1, 6, 11, 16, 21, 26\}$. Plot and report the accuracy on the validation data as a function of k.

   (b) Choose k^* that has the highest performance on validation data. Report the chosen k^* and the final test accuracy.

   (c) Implement a function `knn_impute_by_item` on the same file that performs item-based collaborative filtering instead of user-based collaborative filtering. Given a question, kNN finds the closest question that was answered similarly, and predicts the correctness basted on the closest question's correctness. State the underlying assumption on item-based collaborative filtering. Repeat part (a) and (b) with item-based collaborative filtering.

(d) Compare the test performance between user- and item- based collaborative filtering. State which method performs better.

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

2. **[15pts] Item Response Theory.** In this problem, you will implement an Item-Response Theory (IRT) model to predict students' correctness to diagnostic questions.

   The IRT assigns each student an ability value and each question a difficulty value to formulate a probability distribution. In the one-parameter IRT model, β_j represents the difficulty of the question j, and θ_i that represents the i-th students ability. Then, the probability that the question j is correctly answered by student i is formulated as:

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

   We provide the starter code in `part_a/item_response.py`.

   (a) Derive the log-likelihood $\log p(\mathbf{C}|\boldsymbol{\theta}, \boldsymbol{\beta})$ for all students and questions. Here $\mathbf{C}$ is the sparse matrix. Also, show 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 missing functions in `item_response.py` that performs alternating gradient descent on $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ to maximize the log-likelihood. Report the hyperparameters you selected. With your chosen hyperparameters, report the training curve that shows the training and validation log-likelihoods as a function of iteration.

   (c) With the implemented code, report the final validation and test accuracies.

   (d) Select five questions j_1, j_2, j_3, j_4, and j_5. Using the trained $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, plot five curves on the same plot that shows the probability of the correct response $p(c_{ij})$ as a function of θ given a question j. Comment on the shape of the curves and briefly describe what these curves represent.

3. **[15pts] Matrix Factorization OR Neural Networks.** In this question, please read both option (i) and option (ii), but you only need to do one of the two.

   (i) **Option 1: Matrix Factorization.** In this problem, you will be implementing matrix factorization methods. The starter code is located at `part_a/matrix_factorization`.

   (a) Using a function `svd_reconstruct` that factorizes the sparse matrix using singular-value decomposition, try out at least 5 different k and select the best k using the validation set. Report the final validation and test performance with your chosen k.

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

   (c) Implement functions `als` and `update_u_z` located at the same file that performs alternating least square method (ALS). This will be covered in week 10's lecture. As a reminder, the objective for ALS 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. Tune learning rate and number of iterations. Report your chosen hyperparameters. Try at least 5 different values of k and select the best k^* that achieves the lowest validation accuracy.

(e) With your chosen k^*, plot and report how the training and validation squared-error-losses change as a function of iteration. Also report final validation accuracy and test accuracy.

(f) For ALS, you trained the model as a regression problem; your loss function was a squared-error-loss. How would you modify the loss function if you would like to train the model as a binary classification problem? Describe your modified loss function.

(ii) **Option 2: Neural Networks.** In this problem, you will implement neural networks to predict students' correctness on a diagnostic question. Specifically, you will design an autoencoder model. Given a user $\mathbf{v} \in \mathbb{R}^{N_{questions}}$ 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:

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

for some activation functions h and g. In this question, you will be using sigmoid activation functions for both. Here, $\mathbf{W}^{(1)} \in \mathbb{R}^{k \times N_{questions}}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{N_{questions} \times k}$, where $k \in \mathbb{N}$ is the latent dimension. We provide the starter code written in `PyTorch` at `part_a/neural_network`.

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

(b) Implement a class `AutoEncoder` that performs a forward pass of the autoencoder following the instructions in the docstring.

(c) Train the autoencoder using latent dimensions of $k \in \{10, 50, 100, 200, 500\}$. Also, tune optimization hyperparameters such as learning rate and number of iterations. Select k^* that has the highest validation accuracy.

(d) With your chosen k^*, plot and report how the training and validation objectives changes as a function of epoch. Also, report the final test accuracy.

(e) Modify a 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 a 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 final validation and test accuracy. Does your model perform better with the regularization penalty?

4. **[15pts] Ensemble.** In this problem, you will be implementing bagging ensemble to improve the stability and accuracy of your base models. Select and train 3 base models with bootstrapping the training set. You may use the same or different base models. Your implementation should be completed in `part_a/ensemble.py`. To predict the correctness, generate 3 predictions by using the base model and average the predicted correctness. Report the final validation and test accuracy. Explain the ensemble process you implemented. Do you obtain better performance using the ensemble? Why or why not?

1.5 Part B of Default Project

In the second part of the project, you will modify one of the algorithms you implemented in part A to hopefully predict students' answers to the diagnostic question with higher accuracy. In particular, consider the results obtained in part A, reason about what factors are limiting the performance of one of the methods (e.g. overfitting? underfitting? optimization difficulties?) and come up with a proposed modification to the algorithm which could help address this problem. 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. its accuracy); rather, your grade will be based on the quality of your analysis. Try to be creative! You may also optionally use the provided metadata (`question_meta.csv` and `student_meta.csv`) to improve the accuracy of the model. At last, you are free to use any third-party ideas or code as long as it is publicly available. You must properly provide references to any work that is not your own in the write-up.

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

1. **[15pts] Formal Description:** Precisely define the way in which you are extending the algorithm. You should provide equations and possibly an algorithm box. Describe way your proposed method should be expected to perform better. For instance, are you intending it to improve the optimization, reduce overfitting, etc.?
2. **[10pts] Figure or Diagram:** that shows the overall model or idea. The idea is to make your report more accessible, especially to readers who are starting by skimming your report.
3. **[15pts] Comparison or Demonstration:** Include:
   - A comparison of the accuracies obtained by your model to those from baseline models. Include a table or a plot for an illustrative comparison.
   - Based on the argument you gave for why your extension should help, design and carry out an experiment to test your hypothesis. E.g., consider how you would disentangle whether the benefits are due to optimization or regularization.
4. **[15pts] Limitations:** of your approach.
   - Describe some settings in which we'd expect your approach to perform poorly, or where all existing models fail.
   - Try to guess or explain why these limitations are the way they are.
   - Give some examples of possible extensions, ways to address these limitations, or open problems.

1.6 Submission for Default Project

If you choose the default project, you need to submit:

- Your answers to Part A and B, as a PDF file titled `final_report.pdf`. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.
- The Python codes you used for both Part A and B, as a zip file `code.zip`. You may exclude the folder `/data` from the starter code.

2 Open-ended project

2.1 Introduction

The purpose of the open-ended project format is to give you some experience working on a piece of original research and writing up your results in a paper style format. You can get feedback on your research idea/application clearly with a project proposal, and from the project presentations. You will document the project progress in a final report.

If you choose the open-ended project, there are two important dates besides the final deadline::

- If you get us a project proposal by April 1st we will have time to give you useful feedback. The earlier the better. This project proposal is ungraded.
- Project presentations, woth 20% of the open-ended project grade, will happen on April 9th. If the timezone is bad for you, you can send a pre-recorded presentation. If you choose the default project, there's no need to do a presentation.

2.2 Writing format

All submissions must be in PDF format. You may include algorithm blocks, tables, and figures. The write-ups should be prepared in the NeurIPS paper format: https://neurips.cc/Conferences/2019/PaperInformation/StyleFiles. You may find online editors such as Overleaf helpful for writing the reports: https://www.overleaf.com/latex/templates/neurips-2019/tprktwxmqmgk

Project Proposal : The project proposal is ungraded. It's limited to two pages including references. As a suggestion, it should probably contain roughly the following sections:

- 1/2 page abstract and introduction
- 1/2 page related works
- 1/2 page method / algorithm

The point of the proposal is mainly for us to give you feedback and formulate a plan for the final report. You can email your proposal report to `duvenaud@cs.toronto.edu`

Final report : You will expand out your project proposal to include experiments and comprehensive method sections. You are expected to discuss the experimental results in detail, and highlight any interesting findings. You must also submit code sufficient to reproduce your experiments.

2.3 How to Choose a Research Project

Any course project to do with machine learning is in scope. There are two categories of projects to choose from.

- Understanding and analysis: For the students who would like to have a more in-depth understanding of the course material, it is often a good idea to re-implement an existing method and re-evaluate the implementation against some standard benchmarks.
- Reproduce the experimental results from some existing papers. Perform sensitivity analysis on hyper-parameters.
- Apply / extend existing algorithms to a new application / task / dataset.
- Do a lit review of an area, comparing the properties of existing approaches and identifying open questions in that area.
- Improve / fix an existing algorithm. Evaluate the improvement on benchmark environments.
- Develop novel model architectures / algorithms for a new application / area / environment.

If you decide to work on a research idea, you will need to implement and compare the performance of your method against at least one existing approach in your problem. Here is some advice on picking a good research problem from Bill Freeman: `https://billf.mit.edu/sites/default/files/documents/cvprPapers.pdf` and from David Patterson's slides part III and IV: `https://people.eecs.berkeley.edu/pattrsn/talks/BadCareer.pdf`.

You are welcome to do a project related to your research. In this case, your project proposal and final report must each clearly explain the relationship to your research, what work was already done prior to the course, and what work (if any) was done by people not on the project team. Our expectations will be higher in this case.

2.4 Project Presentation Grading Scheme

For the open-ended projects, 20% of the project grade will come from the project presentations. If you're not very far along in your project yet, don't panic! The presentation is about explaining your project idea clearly and concisely, and is only 5 minutes long. If you don't yet have results to show, simply explaining what you're planning to do and why is 100% OK. It's a chance for us to give feedback, and for you to practice research presentation skills.

- Motivate and define the problem: 20% What new thing will we be able to or understand do if you succeed?
- Briefly mention related work: 20% You can leave this till the end, if you like. Explain at least one the main ideas of your project clearly: 20% It's enjoyable and a good use of time to listen to presentations if they also educate the audience.
- Show a draft of main figure: 10% Or show some sort of visual representation of the main idea.
- Explain planned experiments or show results: 20% Try to explain what we might learn from these experiments.
- Finish under time (5 minutes): 10% I will cut you off at 5 minutes sharp!!

2.5 Open-Ended Project Report Grading Scheme

The idea is that this project report should be a manageable amount of work, but that if you want to turn your project into a paper, everything in the project report will need to be done anyways. If you feel that your project won't fit into this rubric, please talk to me. There are many ways to make contributions to a field!

Length: 4 to 8 pages, not including appendices or bibliography. Don't be afraid to keep the text short and to the point, and to include large illustrative figures. You can include as many proofs, extra details, experiments, etc. as you want in the appendices.

- Abstract (4 points) that summarizes the main idea of the project and its contributions. Should be understandable to anyone in the course. You don't need to say everything you did, just what the main idea was and one or two takeaways. Introduction (4 points) that states the problem being addressed and why we might want to solve it.
- Figure or diagram (8 points) that shows the overall model or idea. The idea is to make your paper more accessible, especially to readers who are starting by skimming your paper. You must create a new figure, not just use someone else's, even with attribution!
  - For the project, taking a picture of a hand-drawn diagram is fine, as long as it's legible.
  - For camera-ready diagrams, we recommend using Tikz, a LaTeX package.
  - Try to be clear whether arrows indicate computational flow, or conditional dependencies, or both.
- Formal description (16 points) of the model / loss function / conjecture / problem domain. Include at least one of:
  - An algorithm box.
  - Equations describing your model.
  - A theorem or formally stated conjecture.
  - A formal description of a problem domain. Highlight how your model is different from other approaches, or what the main relevant considerations are for the domain. This can be done by comparing it to an existing model, perhaps by using another diagram or in words. E.g. if you are proposing a new algorithm that only changes one line in an existing algorithm, highlight that one line, or do a side-by-side comparison.
- Related work (16 points) section and bibliography. If your project builds on previous work, clearly distinguish what they did from what your new contribution is. Also, include a 1-2 sentence summary of other closely related papers. I realize you might not know about all related papers (or have time to carefully read all related papers), and that's OK for this project. Using bibtex is annoying at first, but Google Scholar can give you the bibtex entries.
- Comparison or demonstration (16 points). Include at least one of:
  - A demonstration of a theorem or conjecture. For example, an example or counter-example.
  - A comparison of data generated by your model to a baseline model. Qualititative evaluation is OK.

- An experiment demonstrating a property that your model has that a baseline model does not. Experiments should also include a description of how you prepared your datasets, how you trained your model, and any tricks you used to get it to work.
- If doing a review, include a table comparing the properties of the different approaches. Toy data is OK! The point is to help the reader understand why or when we would want to use one approach over another, or to understand something better. Try to summarize the main takeaways.

- Limitations (12 points) of your approach. Describe some settings in which we'd expect your approach to perform poorly, or where all existing models fail. Try to guess or explain why these limitations are the way they are. Give some examples of possible extensions, ways to address these limitations, or open problems.
- Conclusions (4 points) State the results achieved in relation to the problem described in the introduction. Repeat the main takeaways from your paper.
- Novelty (20 points) Novelty is graded on a sliding scale and is of course somewhat subjective, but here are a few guidelines:
  - 0 points: A literature review with no new content. This is completely OK as a project, but it won't earn an A+.
  - 5 to 15 points: A novel combination of method and type of data. You get novelty points for making tweaks to a method to take advantage of the structure of a new dataset or domain. Think of a reason why the experiment turned out the way it did, and check if this was actually the reason. Alternatively: Motivate and formally pose a new problem, such as characterizing the difficulty of some task, or the learnability of some domain.
  - 20 points: Novel method with motivation - not just making a random tweak for no reason. For full marks, you need to check if the reason you said it should work better is actually the reason why it worked better. Alternatively, do a theoretical analysis with a new result.

  Extra guidelines:

  - You're free to play with the format of your paper, as long as all the above content is there and easy to find.
  - Put long mathematical derivations that interrupt the flow of the paper in an appendix. This is also a good place to put extra generated examples.
  - Text in figures should be about the same size as the text in the rest of the paper.
  - For figures, try to use a vector graphics format such as pdf, eps or svg so that your figures don't look blurry.

3 Friendly Advice for Either Project Format

- **Read carefully!** Make sure you are following the guidelines. Read this document carefully to understand the dataset, what you are asked to implement and demonstrate, etc. 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 what the results were. Just be scientific.
- **Be careful!** Don't do things like test on your training data, set hyperparameters using test accuracy, compare unfairly against other methods, include plots with unlabeled axes, use 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)