Predicting Math Marks Using Math: A Machine Learning Analysis of Predicting STEM Question Correctness

Eric Zhu and Felix Liu

15/12/2020

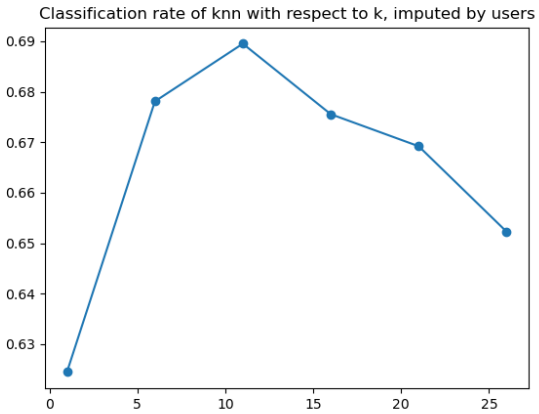
Contents

Part A

Question 1 - k-Nearest Neighbor

Part a

1. Validation Accuracy (users), $k = 1$: 0.6244707874682472
2. Validation Accuracy (users), $k = 6$: 0.6780976573525261
3. Validation Accuracy (users), $k = 11$: 0.6895286480383855
4. Validation Accuracy (users), $k = 16$: 0.6755574372001129
5. Validation Accuracy (users), $k = 21$: 0.6692068868190799
6. Validation Accuracy (users), $k = 26$: 0.6522720858029918

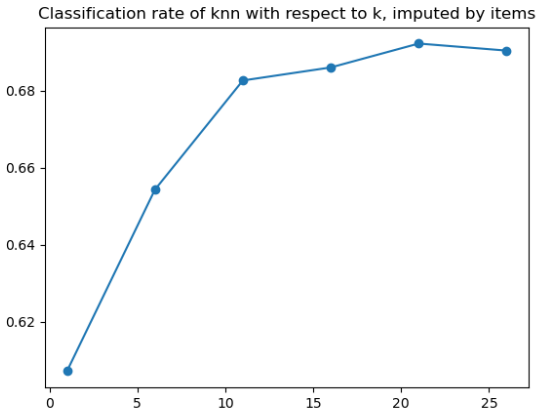


Part b

The best performing k^* was $k^* = 11$. The corresponding test accuracy is: 0.6841659610499576.

Part c

1. Validation Accuracy (items), $k = 1$: 0.607112616426757
2. Validation Accuracy (items), $k = 6$: 0.6542478125882021
3. Validation Accuracy (items), $k = 11$: 0.6826136042901496
4. Validation Accuracy (items), $k = 16$: 0.6860005644933672
5. Validation Accuracy (items), $k = 21$: 0.6922099915325995
6. Validation Accuracy (items), $k = 26$: 0.69037538808919



The best performing k^* on the validation data was $k^* = 21$. The corresponding test accuracy is: 0.6816257408975445.

Part d

From our testing, we find that the better performing method was user-based collaborative filtering as we have 0.6841659610499576 vs 0.6816257408975445 (user vs item based). The accuracies between the two methods are very close though.

Part e

1. The first potential limitation is the curse of dimensionality, as we have high dimensional data. There are 1774 questions, so each sample is a 1774 dimensional vector, which could mean that the knn model would be impacted by the curse of dimensionality.
2. This dataset is relatively large, with many thousands of samples and many questions per sample. KNN requires that we compute the distance of each point we wish to classify with the other points (no training period), which makes the algorithm relatively slow at test time compared to a classification model that trains the model beforehand, e.g., neural network with softmax classification.

Question 2 - Item Response Theory

Part a

$$p(C|\theta,\beta) = \prod_{i=1}^{N}\prod_{j=1}^{D} p(c_{ij}=1|\theta_i,\beta_j)^{c}\left(1-p(c_{ij}=1|\theta_i,\beta_j)\right)^{1-c}$$

$$\log p(c|\theta,\beta) = \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log p(c_{ij}=1|\theta_i,\beta_j) + (1-c_{ij})\log\left(1-p(c_{ij}=1|\theta_i,\beta_j)\right)$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log\left(\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right) + (1-c_{ij})\log\left(1-\frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}\right)$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\left[\log\exp(\theta_i-\beta_j) - \log(1+\exp(\theta_i-\beta_j))\right] + (1-c_{ij})\log\left(\frac{1}{1+\exp(\theta_i-\beta_j)}\right)$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}(\theta_i-\beta_j) - c_{ij}\log(1+\exp(\theta_i-\beta_j)) + (1-c_{ij})\left[\log 1 - \log(1+\exp(\theta_i-\beta_j))\right]$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}(\theta_i-\beta_j) - c_{ij}\log(1+\exp(\theta_i-\beta_j)) - \log(1+\exp(\theta_i-\beta_j)) + c_{ij}\log(1+\exp(\theta_i-\beta_j))$$

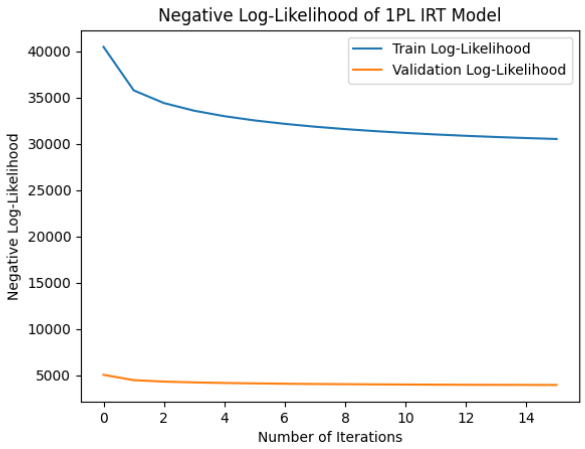
$$\log(c|\theta,\beta) = \sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}(\theta_i-\beta_j) - \log(1+\exp(\theta_i-\beta_j))$$

$$\ell(\theta,\beta) = \log(c|\theta,\beta)$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{j=1}^{D} c_{ij} - \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}$$

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{N} -c_{ij} + \frac{\exp(\theta_i-\beta_j)}{1+\exp(\theta_i-\beta_j)}$$

Part b

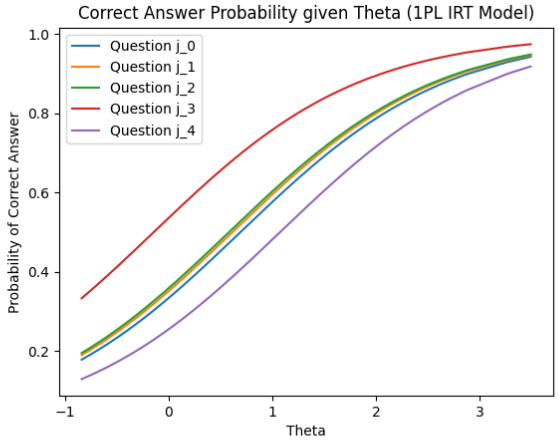


For the hyperparameters, $\boldsymbol{\theta}, \boldsymbol{\beta}$ were initialized as a sample from a uniform distribution. The learning rate was chosen to be 0.01, and the number of iterations was 16.

Part c

1. Final Validation Score: 0.7087214225232854
2. Final Test Score: 0.7039232289020604

Part d



The shape of the curves is similar to that of the sigmoid function, which is representative of the sigmoid function being used as the activation function in the model. These curves describe the positive correlation between the ability of students and their predicted probability of answering a given question correctly. As student ability increases, the model expects that the chances of them answering the question correctly increases as well.

Question 3 - Option 2: Neural Networks

Part a

Here are the following differences between ALS and neural networks:

1. Alternating least squares uses matrix factorization, meaning that it uses the product of a user matrix and an item matrix to help make predictions. An autoencoder projects the input layer onto lower dimensional subspace(s), namely a latent dimension k, in encoding and then decodes the latent layer back to the original dimensions.

2. The hyperparameters between ALS and neural networks are different. For instance, ALS does not use have an `epoch` hyperparameter.
3. Unless the objective function of a neural network is necessarily convex, we will always be minimizing a non-convex cost function. With the ALS model, when either $\mathbf{u}_n$ or $\mathbf{z}_m$ is fixed, we minimize a convex function.

Part b

Done in code. Please consult `neural_network.py`.

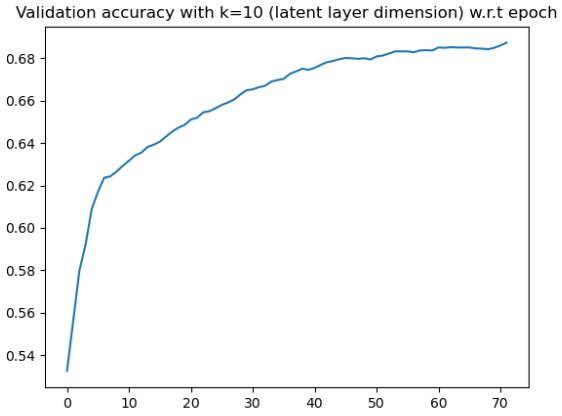
Part c

We got the following validation accuracies:

1. $k = 10$: 0.6862828111769687
2. $k = 50$: 0.6816257408975445
3. $k = 100$: 0.6747106971493085
4. $k = 200$: 0.6703358735534858
5. $k = 500$: 0.6603161162856337

We then see that we should choose k^* to be $k^* = 10$.

Part d



The test accuracy with $k = 10$: 0.6821902342647473.

Part e

With L_2 regularization, we obtain the following validation accuracies with respect to λ:

1. $\lambda = 0.001$: 0.68077900084674
2. $\lambda = 0.01$: 0.6809201241885408
3. $\lambda = 0.1$: 0.6711826136042901
4. $\lambda = 1$: 0.6690657634772792

As such we choose our λ to be $\lambda = 0.01$, and we get a final validation and test accuracy of 0.6809201241885408 and 0.676545300592718, respectively.

We note that our model did not perform better with the regularization penalty (0.676545300592718 vs 0.6821902342647473).

Question 4 - Ensemble

This question tasked us with creating an ensemble model through bagging. We chose to create an ensemble model using the KNN model we created in **Question 1** with $k = 11$ - our best performing k. We went through the bagging procedure, creating 3 bootstrap samples with `sklearn` and its `resample` function (`sklearn.utils.resample()`) with the same amount of samples as the training dataset. Note that `resample()` samples with replacement as required by the bagging method. We then use either the validation set or the training set and get predictions for every data point in the selected datasets for all 3 base models. We then take the elementwise average of the 3 lists of predictions from the base models. Using the vector of averaged predictions, we call `evaluate()` to evaluate the accuracy of the model on either the validation set or test set.

Evaluating our ensemble model, we find the following accuracies on the validation and test sets:

1. validation accuracy: 0.6065481230595541
2. test accuracy: 0.6130397967823878

We see that the ensemble model performs worse than the base model (0.6130397967823878 vs 0.6821902342647473). A potential reason for this is because the bagging does not reduce bias in the case of squared error, which is our loss function. It seems that we may still have high correlation between classifiers, while our base model already has relatively low variance (validation accuracy is very close to test accuracy on the base model). So it is possible for us to see this as we have relatively the same accuracy with the similar relatively low variance (close test/validation accuracies).

Part B

Question 1 - Formal Description

We are extending the `1-pl IRT` model by adding a pseudo-guessing parameter γ in order to leverage the `2-pl IRT` model.

Our motivation to explore the `2-pl IRT` model came from brainstorming various potential models, such as various changes to our neural network model, introducing more features from the student metadata file, and other possible models/changes. However, we kept coming back to the idea of a `2-pl IRT` model because as students (who sometimes have long and brain-teasing finals!), we know that exam performance can seemingly be harder with more multiple choice options. We felt that capturing this intuition could very well improve the performance of our original `IRT` model as it could help decrease the variance and improve generalize by capturing more of what goes on during an exam.

Ultimately, we believed that our pseudo-guessing parameter would be the probability of randomly guessing the correct answer. In particular, after examining handout further, we discovered that the questions came from the Eedi NeurIPS 2020 Education Challenge (Eedi 2020), which consists of 4 option multiple choice questions as described by their "Tasks 1 & 2: Prediction" description section.

Given that it is fair to assume one option isn't favoured over another, e.g., option A doesn't have more intrinsic weight over option D, we assign this pseudo-guessing parameter the value of 0.25.

Upon further research, we discovered "cognitive theory suggests that when an examinee does not know the correct response, the individual will guess"(Johnson 2007). The `1-pl IRT` model also assumes that the examinee ability approaches 0 if they do not know the answer, i.e., $\theta_i \to -\infty \Rightarrow p(c_{ij} = 1|\theta_i, \beta_j) = 0$, which is a false assumption due to guessing (Johnson 2007). So in effect, considering γ allows us to "regularize" the objective function by ensuring that even the most incapable student for a given question has some chance of getting the question correct. In other words, we penalize the model, when the model has overfit such that a student is deemed to be too incapable.

Mathematical formulation of algorithm

We will now formally define our `2-pl IRT` model. Note that our starting point was from the Stanford CS398 slides on IRT (Information Response Theory) (Piech 2019).

We will also use the same variables for parameters and random variables as defined in **question 2** of **Part A**, i.e., $\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{c}$. We will additionally define N to be the number of samples in our training dataset, with D to be the number of features in a particular sample. Recall that γ is our pseudo-guessing parameter, which has a value of 0.25.

We begin with the likelihood function ($L(\boldsymbol{\theta}, \boldsymbol{\beta})$):

$$\prod_{i=1}^{N}\prod_{j=1}^{D} p(c_{ij} = 1|\theta_i, \beta_j)^c(1 - p(c_{ij}|\theta_i, \beta_j))^{1-c}$$

In the same vein of our derivation for `1-pl IRT`, we will need to first derive the log-likelihood ($\ell(\boldsymbol{\theta}, \boldsymbol{\beta})$), making it easier to find the MLEs for θ_i and β_j.

For brevity sake, we will omit some steps in the derivation of the log-likelihood. First, we redefine $p(c_{ij} = 1|\theta_i, \beta_j)$ to be:

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

It follows that the log-likelihood can be written as:

$$\sum_{i=1}^{N}\sum_{j=1}^{D} c_{ij}\log(\gamma + (1 - \gamma)\frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)}) + (1 - c_{ij})\log(1 - (\gamma + (1 - \gamma)\frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)}))$$

We then arrive at the simplified log-likelihood:

$$c_{ij}\log(\gamma + exp(\theta_i - \beta_j)) + (1 - c_{ij})\log(1 - \gamma) - \log(1 + exp(\theta_i - \beta_j))$$

Our next step was computing the MLEs for θ_i and β_j, i.e., $\frac{\partial \ell}{\partial \theta_i}$ and $\frac{\partial \ell}{\partial \beta_j}$. Ultimately, we arrive at:

$$\begin{cases} \frac{\partial \ell}{\partial \theta_i} = \frac{c_{ij} exp(\theta_i - \beta_j)}{\gamma + exp(\theta_i - \beta_j)} - \frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} \\ \frac{\partial \ell}{\partial \beta_j} = \frac{-c_{ij} exp(\theta_i - \beta_j)}{\gamma + exp(\theta_i - \beta_j)} + \frac{exp(\theta_i - \beta_j)}{1 + exp(\theta_i - \beta_j)} \end{cases}$$

From there we will use alternating gradient descent. We first calculate new θ_i value using:

$$\theta_i' \leftarrow \theta_i - \alpha \cdot \frac{\partial \ell(\theta_i, \beta_j)}{\partial \theta_i}$$

Next we use the updated θ_i to calculate the new value of β_j:

$$\beta_j' \leftarrow \beta_j - \alpha \cdot \frac{\partial \ell(\theta_i', \beta_j)}{\partial \beta_j}$$

Note that in both of the above gradient descent rules, α is the learning rate, and that upon initialization, $\boldsymbol{\theta}, \boldsymbol{\beta}$ are randomly assigned from a uniform distribution. We will also have an `iterations` hyperparameters.

A mathematical note about γ

Finally, we'll make a more rigourous note about γ's regularizing properties.

From the definition of $p(c_{ij}|\theta_i, \beta_j)$ for the 2-PL IRT model, we have that:

$$\begin{aligned} p(c_{ij}|\theta_i, \beta_j) &= \gamma + (1 - \gamma) * \sigma(\theta_i, \beta_j) \\ &= \frac{(\gamma + exp(\theta_i, \beta_j))}{(1 + exp(\theta_i, \beta_j))} \end{aligned}$$

This is simply the formula for the sigmoid function with a lower horizontal asymptote shifted by γ. Thus, we can see that the pseudo guessing parameter sets an asymptotic lower bound of γ on the conditional probability of a correct answer, i.e.,$p(c_{ij}|\theta_i, \beta_j)$, which prevents students with low ability from being assigned probabilities close to 0, and rectifies the false assumption discussed earlier.

An Algorithm Box

Algorithm 1: 2-pl IRT with pseudo-guessing parameter γ

Input : $iterations \in \mathbb{N}, \alpha \in \mathbb{R}, \boldsymbol{c} \in \{0, 1\}$

Output: $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}, validation\_accuaracy\_list := list$

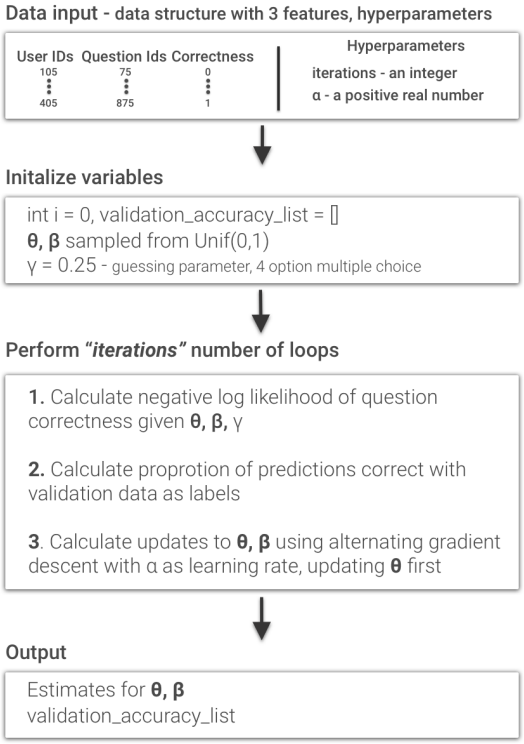
Initialization: $\boldsymbol{\theta}, \boldsymbol{\beta} \sim$ Unif(0.0, 1.0), i := 0, γ:=0.25;

while $i < iterations$ **do**

- $negative\_log\_likelihood := -\log(p(\boldsymbol{c}|\boldsymbol{\theta}, \boldsymbol{\beta}, \gamma))$;
- $score := calculate\_classification\_rate(validation\_data, \boldsymbol{\theta}, \boldsymbol{\beta})$;
- $validation\_accuracy\_list.append(score)$;
- $\boldsymbol{\theta'} := \boldsymbol{\theta} - \alpha \cdot \frac{\partial \ell(\theta_i, \beta_j)}{\partial \boldsymbol{\theta}}$;
- $\boldsymbol{\beta'} := \boldsymbol{\beta} - \alpha \cdot \frac{\partial \ell(\theta_i', \beta_j)}{\partial \boldsymbol{\beta}}$;
- $i := i + 1$;

end

Question 2 - Figure or Diagram



Question 3 - Comparison or Demonstration

We compare the 2PL model results to the 1PL baseline model below:

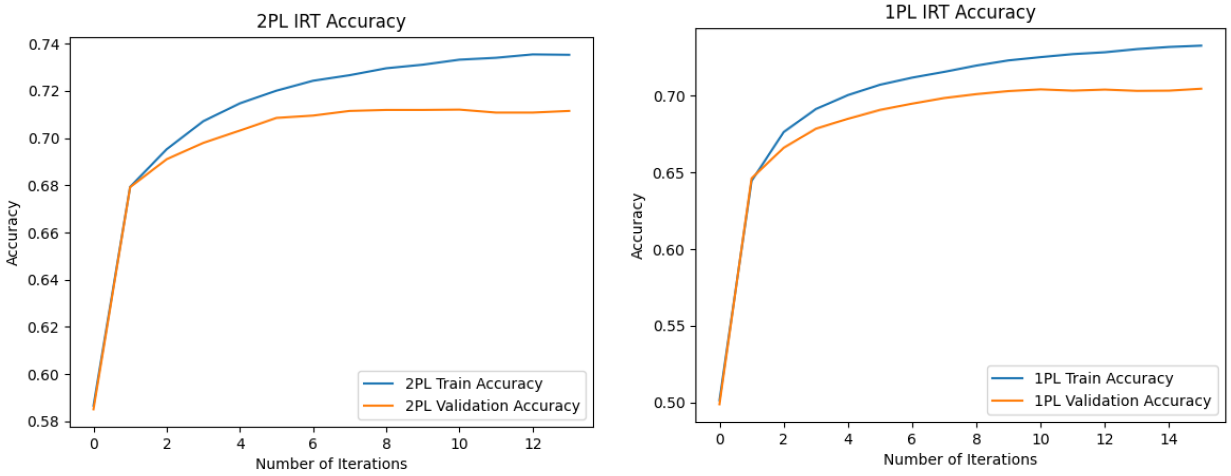
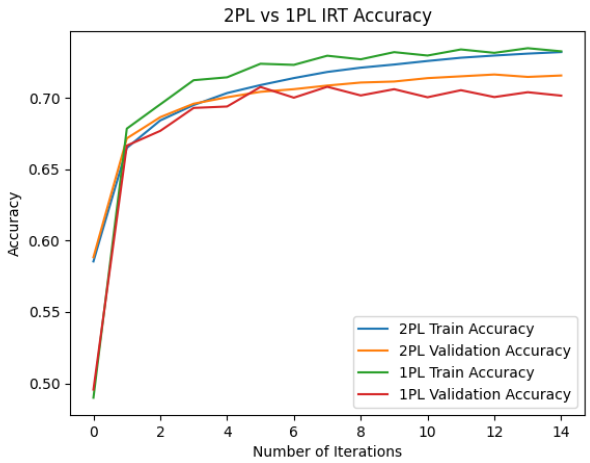


Fig 1: The training and validation accuracy for both the 1PL and 2PL models with tuned hyperparameters.

The hyperparameters used were iterations = 16, learning rate = 0.01 for 1PL and iterations = 14, learning rate = 0.035 for 2PL. 2PL offers an increase in both test and validation accuracy compared to 1PL, as shown by the accuracy scores below:

1. Final 1PL Validation Score: 0.7023708
2. Final 1PL Test Score: 0.7022297
3. Final 2PL Validation Score: 0.7125317
4. Final 2PL Test Score: 0.7056167

We hypothesized that the regularizing effect of the pseudo guessing parameter will improve accuracy scores regardless of the other regularization techniques applied to the two models. Additionally, we predicted that the 2PL model would reduce overfitting due to the lower bound on correctness probabilities introduced by gamma, which prevents the model from overfitting on low ability students in the training data. To test this hypothesis, we designed an experiment that compared the accuracy of the two models and kept hyperparameters fixed. The hyperparameters were set to the average between the tuned hyperparameters chosen for each model to minimize bias. This controlled for the benefits due to regularization and optimization differences between the two models, since learning rate affected the model stability and the number of iterations was used to regularize through early stopping. The results of the experiment demonstrated that our 2PL model objectively outperformed our 1PL model, as shown in the figure on the next page.



The regularization effect that we predicted in our hypothesis was also confirmed since the training accuracy of our 2PL model was lower than the 1PL model, but the validation accuracy was higher. This showed that the 2PL reduced overfitting of the training data and improved generalization performance.

Question 4 - Limitations

The limitations of our model are related to the gamma parameter and the characteristics of the questions used at training and test time. First, the gamma parameter we used is hardcoded and needs to be changed to adapt to datasets with more or less multiple choice answers than the current, which would change the probability of randomly guessing the correct answer. As such, this limits the performance of our model to datasets that have questions with a varying number of choices. Heuristics employed by students to improve their guessing chances, like ruling out purposefully absurd or illogical answers immediately, would also affect the ability for the pseudo-parameter to account for this behaviour. A possible extension to the model that could address the limitations related to guessing is allowing the model to fit the guessing pseudo-parameter as a real parameter, using maximum likelihood estimation and alternating gradient descent. However, it is possible that this would introduce more variance, which leads to overfitting.

Additionally, the model ignores the discrimination factor of the questions, which seeks to quantify the degree that the question favours high ability students over low ability students. This information is captured by the discrimination parameter of a 3PL IRT model (Johnson 2007). As with the previous extension proposal, including this parameter makes the model more complex and can introduce overfitting.

Contributions

This project was completed by Eric Zhu and Felix Liu. We both did an equal amount of work and worked on every problem together either through brainstorming, coding, tuning hyperparameters, trouble shooting, and or generating graphs. The report content was similarly equally split, we both touched all aspects of the written report, having written content for all parts, proofread all parts, etc. . . .

References

Eedi. 2020. "NeurIPS 2020 Education Challenge." *NeurIPS Education Challenge*. https://eedi.com/projects/neurips-education-challenge.

Johnson, Matthew S. 2007. "Marginal Maximum Likelihood Estimation of Item Response Models in R." *Journal of Statistical Software, Articles* 20 (10): 1–24. https://doi.org/10.18637/jss.v020.i10.

Piech, Chris. 2019. "Item Response Theory Cs398." *Stanford*, September. https://web.stanford.edu/class/cs398/lectures/lecture2/2%20-%20IRT.pdf.