Modeling Student Learning in Large-Scale Online Settings

Zhaolei Shi, Susan Athey

May 8, 2021

Abstract

The rise of popular mobile education applications produced data where a large number of students each answers a small subset of questions from a large question bank. Traditional approaches from the education measurement literature face important limitations in this context where data is large but sparse. We propose models based on latent factorization and Bayesian variational inference to address these challenges. Our models retrieve true parameters with greater fidelity than traditional models in simulations. They also scale well computationally to industrial-size datasets. Compared to traditional specifications, latent factorization models can make more accurate predictions on the hold-out test set in general. More latent factors and adding hierarchical dependence on question attributes contribute to better predictive performance in lower-frequency content areas. We conclude by describing a real-world application of our models in personalizing homework assignments. In a future study, we plan to run experiments with this application to quantify the impact of personalization.

1. Introduction

Online education has been transformed in recent years by the rise of mobile learning. Around the world, popular mobile learning apps offer students personalized paths of engagement. In India, the tutoring app Byju's personalizes students' learning journeys using a large knowledge graph (Bhatia, 2017). The US-based Khan Academy offers teachers the option of personalizing student assignments (Khan Academy, 2020b) and also allows students to choose their own pathway through practice exercises (Khan Academy, 2020c). Similar to Khan Academy, 17Zuoye offers choices to teachers and students in China (Sunny Education Inc., 2018). Together, these three apps account for more than 180 million users (Singh, 2020; Khan Academy, 2020a; Sunny Education Inc., 2018), and many other apps offer similar features to their users.

Choice over one's learning path is heralded as an essential part of a broader effort to improve learning through personalization. While this movement towards personalized learning is received with great fanfare in the online education industry, they also created important challenges for documenting student progress. Compared to the traditional setting of standardized tests, students are not exposed to the same items in personalized learning paths. Typically, students are only exposed to a small fraction of items from a large question bank. Furthermore, new interaction data is generated in real-time and new students and new items are frequently added to the system.

As an example, in 17Zuoye's database, there are 1.5 million unique questions exposed to students in December of 2017. However, among active users, the median student is only exposed to 398 questions over the same month. The high-frequency user at 95 percentile only logs 1115 questions in the same period.

Traditionally, education measurement tools based on Item response theory (IRT) are designed for standardized tests with dense data. They are unable to scale to this setting where the student-item matrix is large and sparse. The literature on linking has procedures for accommodating data sparsity. But existing methods are ad hoc, heavily dependent on model specification, and do not computationally scale to large data.

In this paper, we propose new models based on latent factorization and Bayesian variational inference to address these challenges. We find that our models retrieve true parameters with greater fidelity than traditional models in small data settings. In addition, our models also scale well computationally to industrial-size datasets. Finally, compared to the two-parameter model, our factorization models are able to make more accurate predictions on the hold-out test set in general. More latent factors and hierarchical dependence on question attributes contribute to better predictive performance in lower-frequency content areas.

In section 8 of our paper, we describe a real-world application of our models. *Smart Homework* uses predictions from our models to make personalized question recommendations to students that are not too hard nor too easy. We plan to experimentally test the effect of personalization on student outcomes in a future study.

Section 9 concludes the paper by discussing other potential applications of our models.

2. Related Work

Our methods are inspired by recent advances in inference techniques and machine learning models that employ user/item factorization. Variational inference techniques allow Bayesian inference to be recast as an optimization problem, lending it to stochastic optimization techniques (e.g. stochastic gradient descent) which allows the algorithm to scale to large datasets. See Blei et al. (2017) for a review. Recent engineering work allows models using stochastic variational inference to be implemented through off-the-shelf machine learning packages (Tran et al., 2016; Bingham et al., 2019).

One central idea behind our proposed models is that students and items have latent vector representations whose dot product influences the probability of answering a question correctly. This approach draws from recent work on latent factorization that have been applied to providing online recommendations (Gopalan et al., 2015; Donnelly et al., 2020), analyzing complementarity and substitutability in consumer choice (Ruiz et al., 2017; Donnelly et al., 2019), and geographical preferences of restaurant-goers (Athey et al., 2018).

These works draw from the large literature on recommender systems where the standard approach is to find a try to find an approximation of the full matrix of user-item interactions using the product of two lower-rank matrices. Nonetheless, these recent works extend the latent vector representation approach to allow latents to depend on observed characteristics and to account for time-varying effects. We will also incorporate these innovations into our approach.

Our work is also related to the large literature on linking in educational measurement. Linking refers to the practice that compares student performance across different tests (see Kolen and Brennan (2004) for a review). Similar to our goal, linking can be interpreted as a way to overcome sparsity in the combined student-item matrix of different tests that share common persons or common items (Reardon et al., 2019). A major difference, however, is that the linking literature is generally focused on the design of tests for particular settings where researchers control the recruiting of examinees and the administration of exams. As such, procedures for accommodating data sparsity are mostly ad hoc (e.g. estimating parameters from test A first, then keeping these parameters fixed when estimating test B) Kolen and Brennan (2004). There is no prevailing consensus on which methods should be used and typical works in this literature are also only applicable to certain model specifications. Our work differs from this literature in the scale of data and the highly sparse nature of our context and the generality of our approach.

Our work is also related to new literature that casts the model of student learning as solutions to a predictive problem (see Pardos (2017) for a review). In a purely predictive setting, many researchers have taken advantage of the recent developments in training deep neural networks and have applied them to the problem (see Hernández-Blanco et al. (2019) for a review). As a response to the proliferation of deep learning work, recent papers find that simpler models with psychological interpretations can behave just as well as deep learning approaches when structured to fit a few regularities (Khajah et al., 2016; Wilson et al., 2016a,b). Our approach takes the middle ground where we produce interpretable parameters while using a factorization approach to flexibly model latent regularities. We are also unique in our utilization of Bayesian variational inference which allows us to scale to large data while maintaining the flexibility of a Bayesian model to update parameters.

3. Model specification

We use Bayesian variational inference for all parameters in our proposed models. See section A for an summary of Bayesian variational inference. We place Gaussian posteriors on every parameter with flexible mean and scale. We also initialize most parameters with Gaussian priors with mean 0 and a standard deviation of 1. In practice, our optimization methods also use stochastic gradient descent and various computational optimizations implemented by Bingham et al. (2019).

3.1. Model 1: Two-Parameter model

Our first model takes the form of the classic Two-Parameter model from the item response literature. The only difference to traditional models is that we estimate the parameters through Bayesian variational inference.

In this model θ_i is the student parameter and α_j, β_j are question parameters. For the probability of student i answering question j correctly ($Y_{ij} = 1$), the two-parameter model is defined as:

$$P(Y_{ij} = 1|\theta_i, \alpha_j, \beta_j) = \frac{1}{1 + \exp(-\alpha_j(\theta_i - \beta_j))}$$

3.2. Model 2: Latent factorization model

In this model, we map students and questions into latent vectors $(\boldsymbol{\theta}_i, \boldsymbol{\alpha}_j)$ and allow their inner product to influence the probability of correctness. The model is given by:

$$P(Y_{ij} = 1|\boldsymbol{\theta}_i, \boldsymbol{\alpha}_j, \beta_j) = \frac{1}{1 + \exp(-(\boldsymbol{\theta}_i^\top \boldsymbol{\alpha}_j - \beta_j))}$$

The main benefit of latent factorization in this model is analogous to that of methods commonly found in recommendation systems. These types of models allow the parameters to learn from the structure of the student-question matrix. The richness of the latent factors allows us to quantify student ability in multi-dimensional ways as the predictions for correctness will be different for items with different latent factors.

3.3. Model 3: Hierarchical factorization model

In this model, we retain the structure of the latent factorization model. However, we replace a simple latent vector $\boldsymbol{\alpha}_j$ with a concatenation of the latent $\boldsymbol{\alpha}_j$ and a latent transformation of observed question covariates X_j through the transformation matrix H_α. We denote this concatenated vector as $\boldsymbol{\alpha}_j \oplus H_\alpha X_j$. Accordingly, we make $\boldsymbol{\theta}_i$ to be the same length as the result of the concatenation.

The trainable parameters of this model are $\boldsymbol{\theta}_i, \boldsymbol{\alpha}_j, H_\alpha, \beta_j$. The model is given by:

$$P(Y_{ij} = 1 | \boldsymbol{\theta}_i, \boldsymbol{\alpha}_j, H_\alpha, \beta_j, X_j) = \frac{1}{1 + \exp(-(\boldsymbol{\theta}_i^\top (\boldsymbol{\alpha}_j \oplus H_\alpha X_j) - \beta_j))}$$

By adding a flexible dependency on the observed characteristics of questions, we are allowing question attributes to influence the probability of correctness directly. Similar to Athey et al. (2018), this hierarchical structure may allow the model to perform better, especially for questions that appear in the data with low frequency.

3.4. Software implementation

Our code base[1] uses the probabilistic programming package Pyro (Bingham et al., 2019) for Bayesian stochastic variational inference. Pyro is built on top of Pytorch (Paszke et al., 2017) and uses data structure, automatic differentiation, and optimizers from the latter.

[1]Code access is available at `https://github.com/henrishi/bm_model`.

4. Data collection

We apply our models to data generated on the 17Zuoye platform. The data comes from homework assignments and exams in three subject areas–English, math, and Chinese. Records are logged at the students-question level.

Exam data on the 17Zuoye platform are also tagged with questions attributes. Attributes include the appropriate grade level of the question. They also include two types of domain knowledge tags. One system maps questions to competencies. The other maps questions to skills. These tags are manually labeled by content specialists.

5. Parameter retrieval

In this section, our goal is to compare how well our proposed models and estimation strategies retrieve true data generating parameters. In addition, we also compare the parameters retrieved by our approach to those retrieved by a widely-used traditional item response model package *ltm* (Rizopoulos, 2006). *ltm* results are labeled as *traditional_2param* in the following figures.

We focus on the two-parameter models for this exercise because the factorization models are under determined systems and there are multiple parameter arrangements that can yield the same prediction. *ltm* produces frequentist point estimates and standard errors. For the sake of comparison, with our Bayesian models, we take the mean of the posterior distribution as our estimator and the standard deviation of the posterior distribution as our standard error.

5.1. Simulated dense data

We fit *ltm* and *Bayesian two-parameter model* on a small simulated data set. Our data generating process samples θ and β from a normal distribution with mean 0 and standard deviation of 1. α is sampled from a normal distribution with mean 1.2 and standard deviation of 0.5 while constrained to be positive. The student-question response data is then drawn from a Bernoulli distribution where the probability of for a correct answer from student i and question j is $P(Y_{ij} = 1) = \frac{1}{1+\exp(-\alpha_j(\theta_i - \beta_j))}$. We have 30 questions and 50 students in the simulated data and every student has a response for every question.

We focus on the estimate for the β parameter since it is the easiest among the three parameters to estimate. Figure 1 compare the estimates from *ltm* and *Bayesian two-parameter model* alongside the true data-generating parameter. Table 1 presents the correlation (both linear and ranking correlations) between the estimates and the true parameter values.

We see that the *Bayesian two-parameter model* is recovering parameters better than *ltm* . *ltm* has large devious estimates for the parameter with the lowest value and large error ranges for certain parameters with mid-range values. Not only are the point estimates more devious than *Bayesian two-parameter model* , but the standard errors are also bigger for *ltm* in general. This is especially pronounced for the devious estimates. These deviations hurt the correlations of the estimates from *ltm* with the true parameter across all the correlation metrics we report (Pearson, Kendall, and Spearman).

Stats	Pearson correlation (linear)	Kendall correlation (ranking)	Spearman correlation (ranking)
Bayesian two-parameter model - True parameters	0.95	0.83	0.93
ltm - True parameters	0.91	0.69	0.83

Table 1: Small dense data: correlation stats for beta

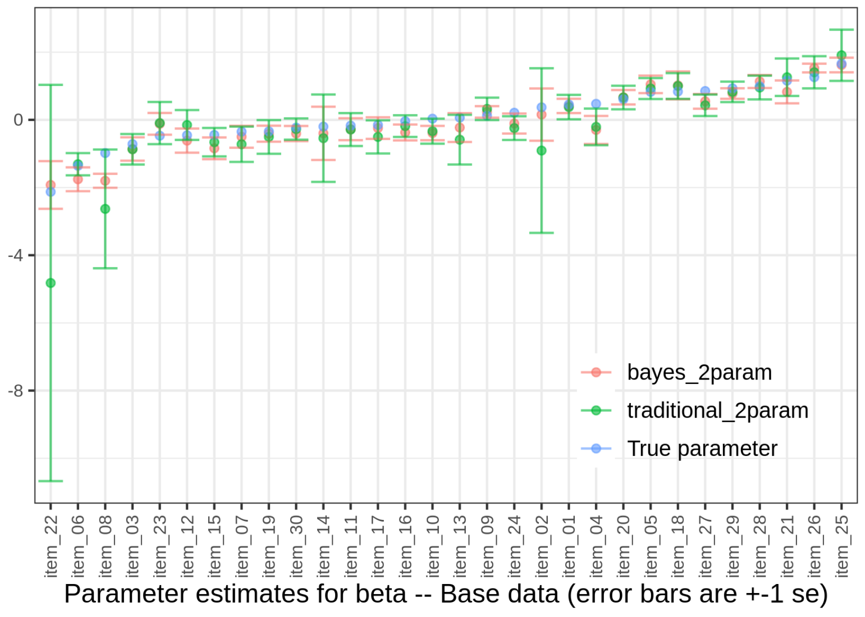

Figure 1: Small dense data: true and estimated beta

5.2. Simulated overlap data with missings

In the current exercise, we generate a simulated dataset where two groups of students share some overlapping questions but not others. The way missing data is structured is shown in Figure 2. Our data generating process samples θ from a normal distribution with mean 0 and standard deviation of 1. α is sampled from a normal distribution with mean 1.2 and a standard deviation of 0.5 while constrained to be positive.

Different from the section 5.1, we sample the β for overlap questions and those only available for students 1 - 50 from a normal distribution with mean 0 and standard deviation of 1. However, we sample the β for questions only available for students 51 - 100 from a normal distribution with mean 0.5 and standard deviation of 0.7. We sampled the β parameters this way to mimic real-world settings where question difficulty is usually different for different groups of students. As in 5.1 the student-question response data is then drawn from a Bernoulli distribution where the probability of a correct answer is a logistic function of the parameters.

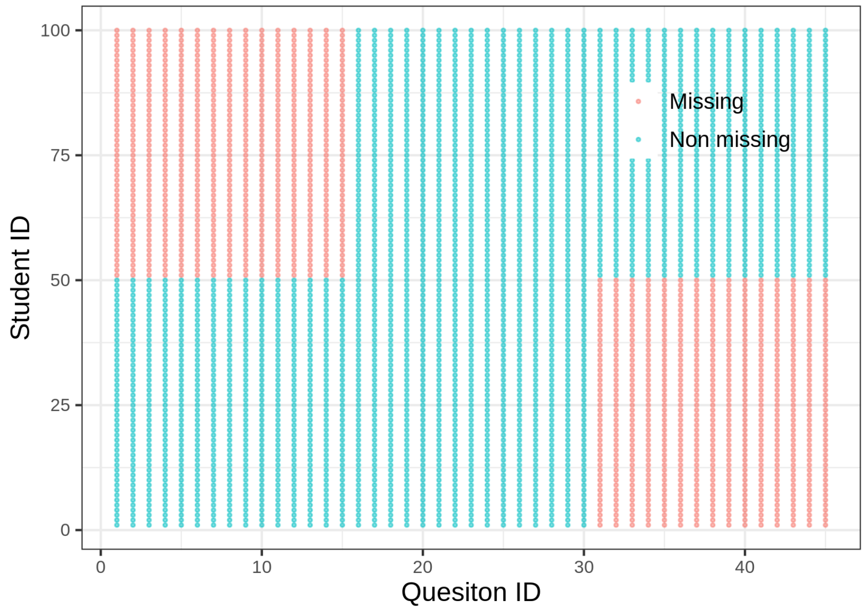

Figure 2: Overlap data missing pattern

In this setting, we again compare the performance of the *ltm* model to the *Bayesian two-parameter model* model in estimates of β. We find that the problem of devious estimates is greatly exacerbated for *ltm* . Figure 3 shows that the error ranges are excessively large for some estimates from *ltm* .

For a more informative figure, we take out the standard error bars of the outliers from the *ltm* model and re-plot in Figure 4. We see that while the majority of parameters have similar under both models, *ltm* is yielding very devious estimates for items 11, 31, and 37. These items are not overlapping items (see point map above) and are only taken by a single group of students.

Bayesian two-parameter model produces point estimates that are much closer to the true parameters for the parameters that *ltm* failed to estimate accurately. This corroborates the Bayesian inference property that the prior distribution serves as a regularizer and allows the model to be more numerically stable. Finally, we see that *Bayesian two-parameter model* again dominates *ltm* in correlation stats with the true parameters by a large margin in this data setting.

Stats	Pearson correlation (linear)	Kendall correlation (ranking)	Spearman correlation (ranking)
Bayesian two-parameter model - True parameters	0.96	0.84	0.96
ltm - True parameters	0.41	0.76	0.91

Table 2: Overlap data: correlation stats for beta

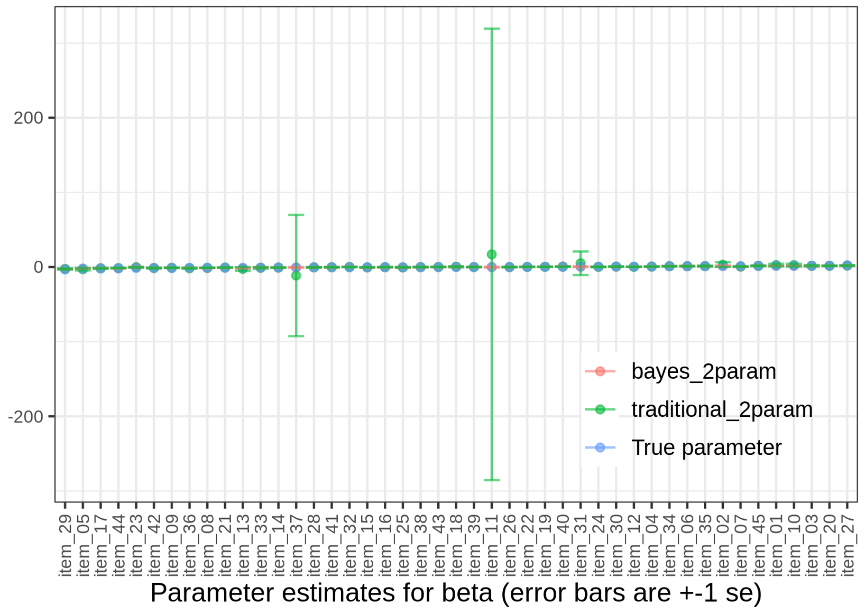

Figure 3: Overlap data: true and estimated beta

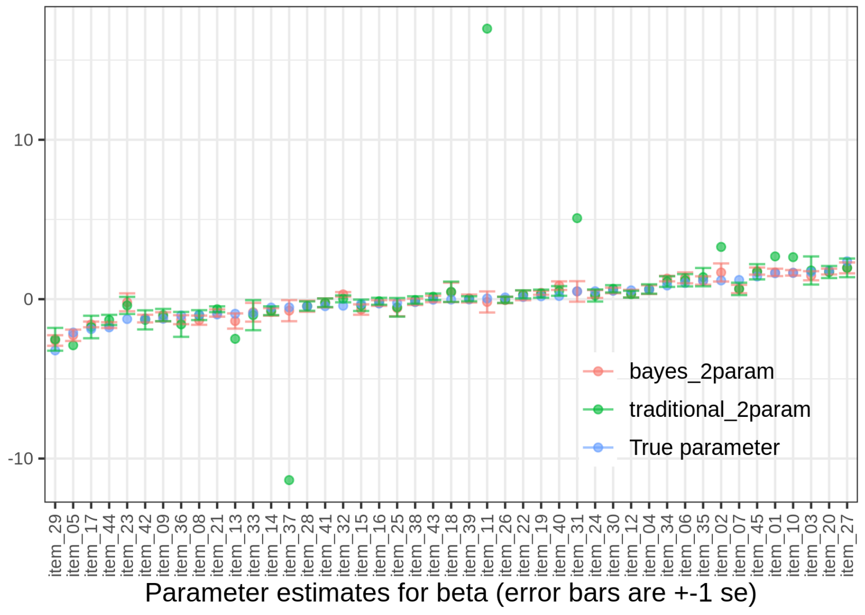

Figure 4: Overlap data: true and estimated beta (zoomed in)

6. Computation performance

In this section, we report the computation performance of our proposed models using an industrial-scale dataset with 88 million responses. The purpose of this section is to show that our proposed Bayesian models *Bayesian two-parameter model* and *Bayesian factorization model* are computationally tractable for real-world applications. Traditional methods such as *ltm* cannot be compared here because they could not handle data of this size. Specifically, *ltm* is unable to produce any useful estimates for even very small datasets when sparsity is at this level. [2]

This dataset comes from homework records of 1000 schools over a period of 2 months. In this dataset, we have 334K questions and 162K students. The overall density of the student-question matrix is 0.16%. The training of our models terminates when convergence has been achieved. We define convergence as the change in loss averaged over the last 5 iterations falls below 0.1% of the change in loss from the first to the second iterations.

In Figure 5 we show the training loss for different model specifications over time. Even though we adopt stochastic gradient descent for optimization, the overall loss curve is smooth due to the large amount of data used for training. The more complex models, factorization models with longer latent vectors, tend to take marginally longer to train. Most models converge within 30 minutes, the longest model to train took less than 40 minutes (see Figure 6).

[2] *ltm* produces numerical errors for even very small samples from this dataset (e.g. a sample of 1000 records). For example, a random sample of 1000 records translated into a matrix of 639 students by 717 questions. *ltm* returns numerical errors for the resulting matrix. One can get *ltm* to run if the missing entries are replaced values (e.g. change all missing values to 0), but running *ltm* on the resulting data takes 16 minutes to converge. My testing shows that *ltm* convergence time is roughly $O(n^2)$ meaning that doubling the amount of data takes 4 times as long to converge. This is an exorbitant amount of time considering that 1000 records are a mere 0.00011% of the full dataset with 88 million records.

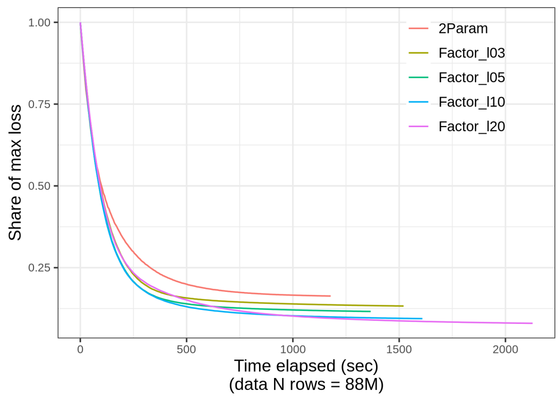

Figure 5: Training loss

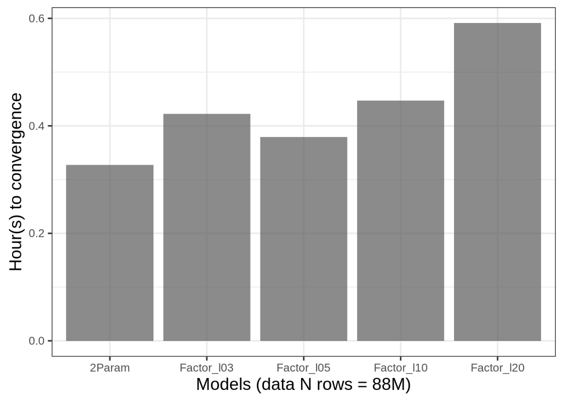

Figure 6: Convergence time

7. Predictive performance

We compare the prediction performance of our proposed models using a large exam dataset where we have access to question attributes. We want to compare the performance of all three models, *Bayesian two-parameter model* , *Bayesian factorization model* , and *Bayesian hierarchical factorization model* . Since *Bayesian hierarchical factorization model* needs question attributes as inputs, we needed to test our predictive performance on a dataset that has question attributes. The exams are also higher stakes than typical homework assignments, as such the data may be more reflective of actual competencies and less noisy as a result.

In this dataset, we have 8.6 million student-question records. There are 3.6k questions and 261k students in total. The overall density of the student-question matrix is 0.92%.

The data were randomly divided into 80% training, 10% validation, and 10% test sets at the level of student-question interactions. This means data for a single student may appear in any of the three sets. The same goes for data from a single question. Model training would stop automatically once convergence is achieved. To get a better metric of the models' capabilities, we define a stricter convergence criterion – as average per-iteration changes in loss becoming 0.05% the initial change.

We compare *Bayesian factorization model* against *Bayesian two-parameter model* in sections 7.1 and 7.2. Having established the superiority of factorization models, we move on to quantify the gains from adding hierarchical dependency in *Bayesian hierarchical factorization model* in section 7.3.

7.1. Overall predictive performance

We first document the overall predictive accuracy for *Bayesian two-parameter model* and *Bayesian factorization model* in Figures 7 and 8. For *Bayesian factorization model* , we show results from three models where the length of the latent vectors $\boldsymbol{\theta}, \boldsymbol{\alpha}$ are taken to be 3, 5, 10, and 20 respectively. We see that *Bayesian factorization model* models enjoy higher AUC in the test set than *Bayesian two-parameter model* . The F1 statistic tells the same story where *Bayesian factorization model* models are superior predictive performance.[3] We note that in all models, training set accuracy is higher than the test set. This is partially attributable to the sparse nature of the student-question matrix. Some students or questions may only show up in the training set or the test set, making test set predictions less accurate than the training set.

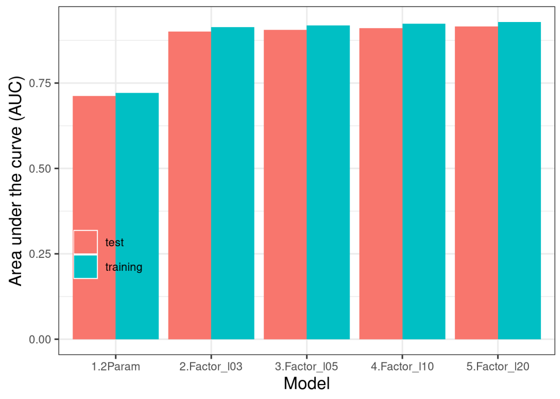

Figure 7: Area under the curve (AUC) by model

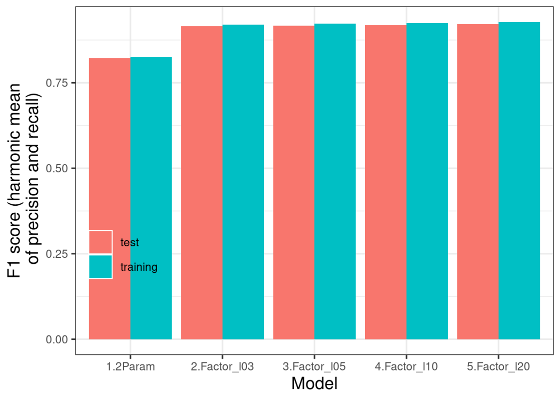

Figure 8: F1 statistic by model

[3]In calculating the F1 statistic, we set the predictive threshold at $P > 0.5$ for a positive prediction.

7.2. Predictive performance by content area

Do factorization models with more latent factors perform better than simpler models on predictive accuracy in less frequent content areas. In this section, we answer this question by examining the predictive performance of candidate models by question knowledge labels. We have 22 knowledge labels in our data. For example, a knowledge label for a math question may be "arithmetic" or "geometry", one for an English question may be "English spelling" or "English pronunciation".

Figure 9 shows the distribution of student-question records by knowledge labels. From left to right, we see knowledge labels in descending popularity. We note that some knowledge labels have significantly lesser data than the most popular knowledge labels.

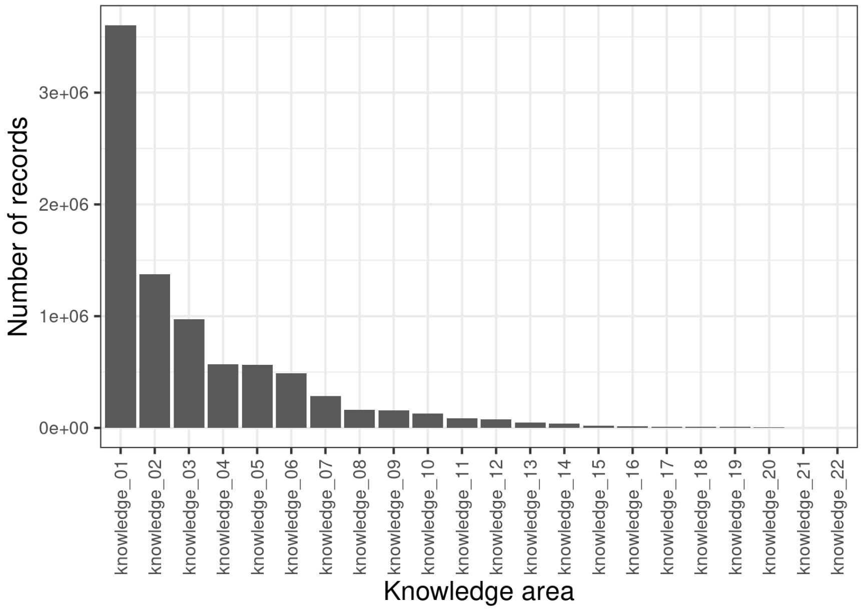

Figure 9: Count of student-question records by knowledge label

We document the predictive performance of different models across knowledge labels in 10 and 11. We see that confirm that *Bayesian factorization model* dominates *Bayesian two-parameter model* in both AUC and F1 across the knowledge labels. Interestingly, the more complicated factorization models (the 10-factor and 20-factor models) have better performance than simpler factorization models (the 3-factor and 5-factor models) in predictive performance for the less frequent knowledge labels. This suggests that additional latent factors may have picked up additional heterogeneity useful in predicting less frequent knowledge labels.

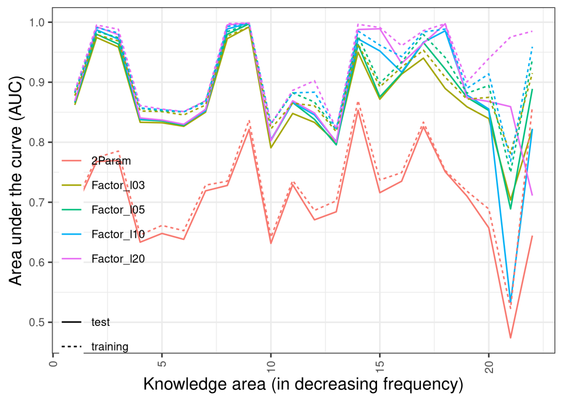

Figure 10: Area under the curve (AUC) by knowledge label

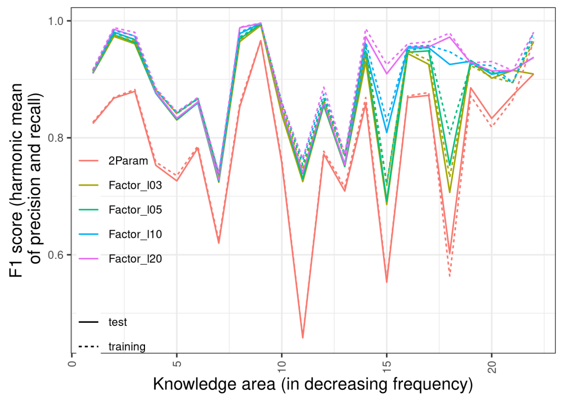

Figure 11: F1 statistic by knowledge label

7.3. Performance gains from adding hierarchical dependency

Having established the superiority of factorization models over *Bayesian two-parameter model* , we move on to quantify the gains from adding a hierarchical dependency in *Bayesian hierarchical factorization model* . To make models more comparable, we hold the size of latent vector constant and compare the *Bayesian factorization model* model with the same model that adds different hierarchical dependency.

We use two types of question attributes, one of which is the one-hot vector of knowledge label described in section 7.2. The other attribute is a multi-hot vector of skills involved in the question. The attribute is encoded as a multi-hot vector because a single question can be associated with multiple skills. For our candidate models with hierarchical dependency, we have freedom in choosing the size of the latent matrix H_α. Following Athey et al. (2018)

we pick H_α such that some entries are 0 so that certain types of question attributes can only contribute to certain parts of the resulting latent vector $H_\alpha X_j$. As specified in 3.3, the latent vector $H_\alpha X_j$ is a linear combination of these representations. We choose two model specifications, the first structures H_α such that the knowledge labels and skills each map to one latent scalar so the resulting $H_\alpha X_j$ is a length 2 vector. The second is slightly more complex in that the knowledge labels and skills each map to two latent scalars so the resulting $H_\alpha X_j$ is a length 4 vector.

In Figure 12 we compare the factorization model with a length 3 latent vector to a hierarchical model with the same setup. We see that the hierarchical models outperform the factorization models slightly overall, but the improvement is more significant for certain low-frequency knowledge labels. The same is true when we look at factorization model with a length 5 latent vector and hierarchical models with the same set up (see Figure 13).[4]

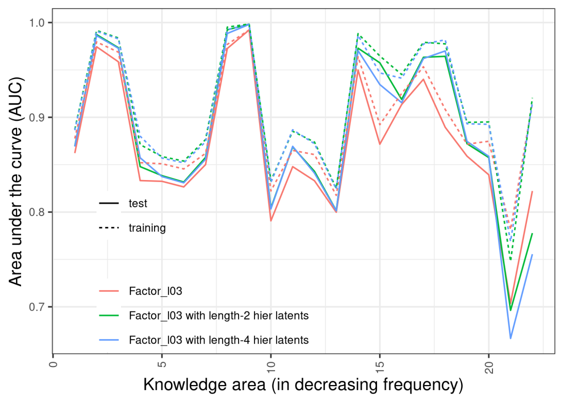

Figure 12: Area under the curve (AUC) by knowledge label (models where latent vector size = 3)

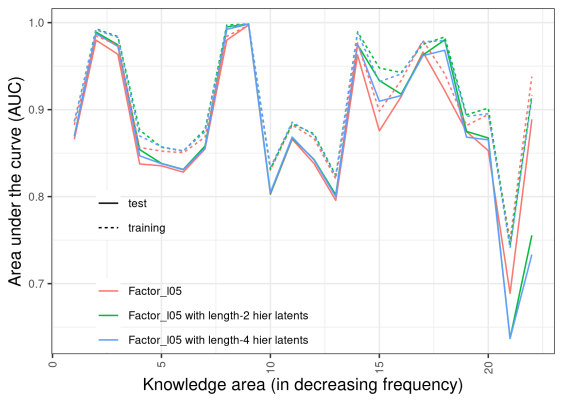

Figure 13: Area under the curve (AUC) by knowledge label (models where latent vector size = 5)

[4]For the sake of brevity, we focus on AUC for our comparison, but the F1 statistics results are substantively the same as the AUC results.

8. Model application: recommender system for personalized questions

This section describes a practical application of our proposed models. Our models were put into use at our partner company to power a recommender system producing personalized questions for students. The recommender system is one piece of a broader product, *Smart Homework* , which combines personalized questions with immediate feedback when the student incorrectly answers a question. Figure 14 shows the relationship between the different components of the system.

The recommender system aims to produce questions that are not too difficult nor too easy. This aligns with a long line of research in educational psychology stemming from the Zone of Proximal Development proposed by Vygotsky (1980). The predicted probabilities for student i getting question j correct is computed using our proposed models from historical data.

Once the probabilities are calculated, the question pool is filtered by taking out questions that the student has already attempted in the past. Then the pool goes through a ranking algorithm for each student. The algorithm ranking questions based on a weighted sum of different factors. The most prominent factor is how close the predicted correctness probability is to 0.7. Other factors include how recent was the question created, whether the question includes a picture or a table. The top 10 questions are selected to form the *Smart Homework* .

As the student works through her *Smart Homework* she moves from one question to the next if she answers correctly. When she answers a question incorrectly, she is taken through a series of explanations and exercises also known as the learn-practice-explain (LPE) module. The student is guided through a series of questions and animations that explains one knowledge point for the triggering question (see Figure 15). The typical time to complete this module is between 2 and 4 minutes. As the student answers the questions, only a correct answer would allow the student to move forward. A wrong answer would trigger a hint and the student is directed to answer the question again.

After completing the module, the student is asked to answer another question similar to the original question she missed. Regardless of whether she answered this question correctly, she is taken back to other questions in the *Smart Homework* . *Smart Homework* exits when the student has completed every question.

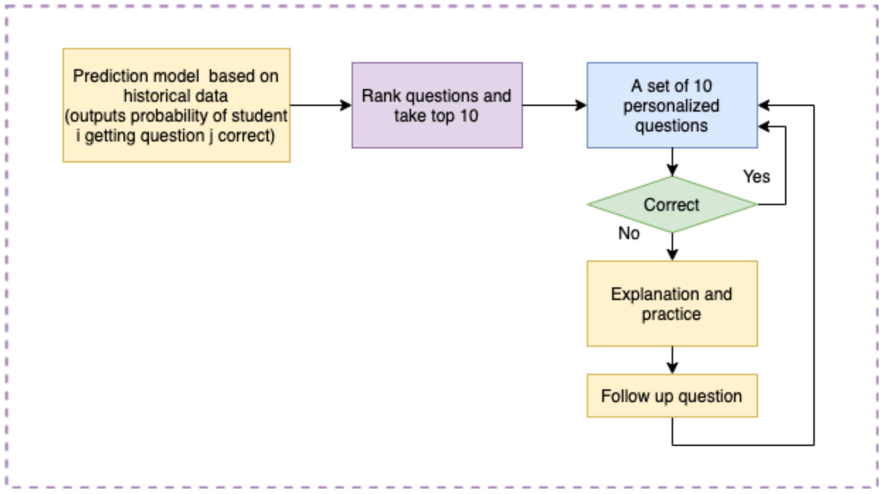
Prediction model based on historical data (outputs probability of student i getting question j correct)
Rank questions and take top 10
A set of 10 personalized questions
Correct
Yes
No
Explanation and practice
Follow up question

Figure 14: *Smart Homework* is powered by a recommeder system and immediate feedback

8.1. Batch updates and parameter freeze

We added two features to the implementation of our models to address practical challenges for building *Smart Homework* . Firstly, we implemented batch updating of parameters. Since our models are based on Bayesian inference, making updates to parameter estimates is very straightforward. Our features load pre-existing parameters into the model in the form of prior distributions. The prior distributions encapsulate all the information the model learned from previous data. Thus, instead of re-fitting the model every time new data comes in, the model needs only to use incremental data to update itself using the prior distributions as its starting point. This helps reduce the computational costs of keeping models up-to-date. In

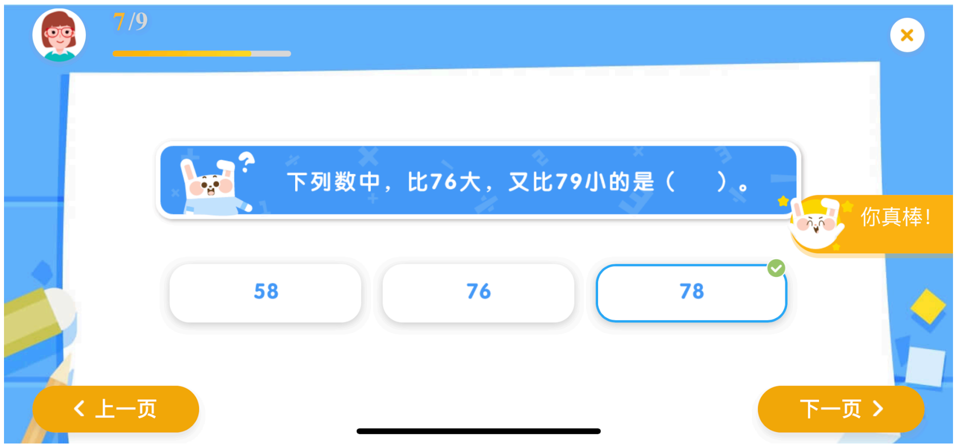

Figure 15: Question in an LPE module

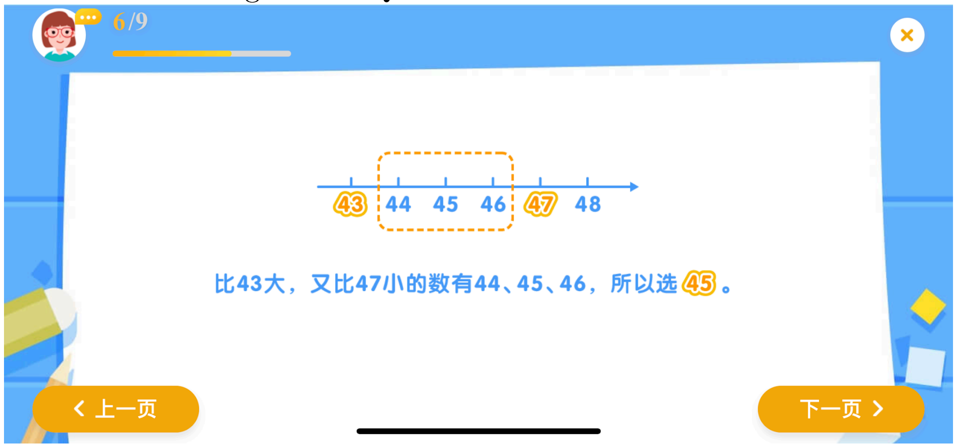

Figure 16: Explanatory text/audio in an LPE module

practice, our model updates itself with incremental data once a week.

Secondly, we added a feature to allow parameters to be fixed. The fitting of the model will only alter trainable parameters but not fixed parameters. This allows question parameters, typically estimated over a large amount of historical data, to remain unchanged in the model fitting process. This lessens the burden of optimization and can further reduce computational costs when a large number of questions are involved.

8.2. Future experimentation

In a future study, we plan to experimentally test whether *Smart Homework* improves student outcomes. Namely, we will assign three versions of *Smart Homework* as the treatment condition and compare its effects on student engagement and achievement against a control group that received a curated (but non-personalized) homework. The three versions of *Smart Homework* will be 1) the full version with both personalization and feedback, 2) a partial version with only personalization, and 3) a partial version with only feedback.

Our partner company is currently (as of May 2021) conducting a pilot of *Smart Homework* with 30K student participants. The pilot stage aims to reveal bugs and inefficiencies in the product's engineering. The pilot stage also aims to establish an understanding of the baseline student behaviors including open rate, completion rate, and utilization of feedback. The main study will proceed after the issues revealed in the pilot stage are addressed.

9. Conclusion

We have shown that latent factorization and hierarchical modeling powered by Bayesian variational inference can make important gains in modeling student-question interactions using large datasets from online education. This set of modern approaches perform well in parameter retrieval and predictive accuracy. They are also numerically stable and can be applied to industrial-scale datasets with computational ease. Finally, they also allow for desirable features such as batch updates and parameter freeze to be implemented with ease.

Smart Homework is only one of the many applications that our models can power. Our models can be used to identify learning gaps. In particular, our factorization approach can identify content domains where a student is weaker than average. Early warning systems and remedial content recommendations can be generated based on such information.

Our models can also improve the informational landscape of parents and teachers. One example can be building learning progress dashboards. For this application, the predictions from our models will be used to derive statistics about student progress. An example of this is using trained models to make predictions about the average rate at which the student will get a particular set of pre-selected domain-representative questions correct. This statistic can be used as an indicator for proficiency.

Finally, applications can make use of the latent parameters the models learned from data. One such application is the automatic labeling of questions based on the latent parameters produced by our models. Another application is discovering the preferences of teachers in terms of which types of question a particular teacher likes to assign.

10. References

References

Athey, S., Blei, D., Donnelly, R., Ruiz, F., and Schmidt, T. (2018). Estimating heterogeneous consumer preferences for restaurants and travel time using mobile location data. *arXiv preprint arXiv:1801.07826*.

Bhatia, R. (2017). A look at how Byju's personalization engine is driving one-on-one learning experience.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2019). Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Donnelly, R., Kanodia, A., and Morozov, I. (2020). A unified framework for personalizing product rankings. *Available at SSRN 3649342*.

Donnelly, R., Ruiz, F. R., Blei, D., and Athey, S. (2019). Counterfactual inference for consumer choice across many product categories. *arXiv preprint arXiv:1906.02635*.

Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical poisson factorization.

Hernández-Blanco, A., Herrera-Flores, B., Tomás, D., and Navarro-Colorado, B. (2019). A systematic review of deep learning approaches to educational data mining. *Complexity*, 2019.

Khajah, M., Lindsey, R. V., and Mozer, M. C. (2016). How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*.

Khan Academy (2020a). Khan Academy 2019 Annual Report.

Khan Academy (2020b). Using Khan Academy for personalized practice and mastery.

Khan Academy (2020c). Using Khan Academy for self-paced practice.

Kolen, M. J. and Brennan, R. L. (2004). Test equating, scaling, and linking.

Pardos, Z. A. (2017). Big data in education and the models that love them. *Current opinion in behavioral sciences*, 18:107–113.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Reardon, S. F., Kalogrides, D., and Ho, A. D. (2019). Validation methods for aggregate-level test scale linking: A case study mapping school district test score distributions to a common scale. *Journal of Educational and Behavioral Statistics*, page 1076998619874089.

Rizopoulos, D. (2006). ltm: An r package for latent variable modeling and item response theory analyses. *Journal of statistical software*, 17(5):1–25.

Ruiz, F. J., Athey, S., and Blei, D. M. (2017). Shopper: A probabilistic model of consumer choice with substitutes and complements. *arXiv preprint arXiv:1711.03560*.

Singh, M. (2020). Indian education startup Byju's is fundraising at a $10B valuation.

Sunny Education Inc. (2018). 17ZUOYE Raises US$250 Million to Consolidate K-12 Edtech Market Leader Position in China.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. (2016). Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard university press.

Wilson, K. H., Karklin, Y., Han, B., and Ekanadham, C. (2016a). Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. *arXiv preprint arXiv:1604.02336*.

Wilson, K. H., Xiong, X., Khajah, M., Lindsey, R. V., Zhao, S., Karklin, Y., Van Inwegen, E. G., Han, B., Ekanadham, C., Beck, J. E., et al. (2016b). Estimating student proficiency: Deep learning is not the panacea. In *In Neural Information Processing Systems, Workshop on Machine Learning for Education*, page 3.

Appendices

A. Bayesian Variational Inference

The main method of inference for parameters in our proposed models is Bayesian variational inference (see (Blei et al., 2017) for a review). Bayesian variational inference is increasingly popular in estimation tasks involving large amounts of data. It has superior runtime performance compared to traditional Bayesian estimation techniques such as Markov Chain Monte Carlo. Variational inference has been shown to work well in mean-fields approximation tasks, but existing methods are less well suited to recover correlations between parameters.

Variational inference proposes a parameterized class of posterior distributions and reduces the Bayesian inference task to one of optimizing a divergence value between the proposed posterior and the true posterior. For this inference task, I use the Kullback-Leibler divergence defined by

$$\begin{aligned}\lambda^* &= \arg\min_{\lambda} KL(q(\boldsymbol{z};\lambda)||p(\boldsymbol{z}|\boldsymbol{x})) \\ &= \arg\min_{\lambda} E_{q(\boldsymbol{z};\lambda)}[\log(q(\boldsymbol{z};\lambda)) - \log(p(\boldsymbol{z}|\boldsymbol{x}))]\end{aligned}$$

where $\boldsymbol{z}$ is the parameter vector of interest, $\boldsymbol{x}$ is the data, and λ is the parameter vector characterizing the proposed posterior distribution. However, directly optimizing this expression is not possible since it involves the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. However, since we can reformulate the divergence as

$$KL(q(\boldsymbol{z};\lambda)||p(\boldsymbol{z}|\boldsymbol{x})) = \log(p(\boldsymbol{x})) - \text{ELBO}(\lambda)$$

where ELBO(λ) is the evidence lower bound defined by

$$\text{ELBO}(\lambda) = E_{q(\boldsymbol{z};\lambda)}[\log(p(\boldsymbol{x}, \boldsymbol{z})) - \log(q(\boldsymbol{z}; \lambda))]$$

we can derive gradients to perform optimization on the ELBO.