Personalized Student Performance Prediction for Online Learning Platforms

Members: Qingyang Bao, Yufei Chen

November 2024

We decide to modify a basic one-layer neural net in an attempt to improve the accuracy of predictions. We will first describe our modified algorithm and show our effort using figures and compare it with the baseline model.

1.1 Description of our modified Neural network

Our model incorporates additional meta-information about the questions by using Principal Component Analysis (PCA) to reduce their dimensionality. PCA works by summarizing the data with fewer features while retaining the most significant aspects, the ones with highest variance. This helps avoid overfitting by discarding less important information. Starting with a subject-question matrix, X, of size $1774 * 388$, since there are 1774 questions and 388 different subjects, the matrix will initially contain ones and zeros: each row corresponds to a question, and each column indicates whether the question is associated with a specific subject. PCA selects the top k components, reducing X to a matrix Z of size $1774 * k$, where each question is represented by a compact k-dimensional embedding. These embeddings capture latent relationships between subject, not only simplifying data, but also enhancing the interpretability by summarizing important patterns. For instance, question number i might have an embedding of $z_i = [z_{i1}, z_{i2}, z_{ik}]$, capturing both its subject composition and its relationships to other subjects. As a result, this helps to model student performance: a student who performs well on questions with such embeddings can be inferred to perform decent on similar embeddings. As a result, the input data to the model will change. For each question i, we weight its embedding by the student's response (rather -1, 0, or 1), through scalar multiplication. Thus, the model's input for each student is a personalized embedding with dimension of **number of questions** $* k$ that combines the student's response profile across questions with question-level metadata reduced by PCA. By incorporating both question metadata and student response data, the model aims to learn richer representations of student knowledge.

We have made several key modifications to the autoencoder architecture to enhance the model's ability to generalize. We added 4 additional layers to both the encoder and decoder networks respectively. These added layers enable the model to learn hierarchical features, which are crucial for capturing complex, non-linear relationships between users and questions. The encoder is now designed with a progressive compression mechanism, where the initial layers capture broad trends in the data, and the deeper layers focus on finer, more detailed representations. This multi-layer architecture allows the model to approximate highly non-linear mappings between inputs (user-question interactions) and outputs, leading to improved generalization and the ability to model intricate patterns in the data. Specifically, the first layer of the encoder captures more general patterns, while each subsequent deeper layer extracts more abstract features.
In order to improve generalization and avoid overfitting, we introduced dropout regularization layers throughout both the encoder and decoder. Given the sparsity of the student performance data, the model is at risk of overfitting and memorizing noise in the training data. Dropout layers, introduced after each linear layer, randomly "drop" a fraction of the neurons during training. This encourages the model to rely on a distributed set of neurons, reducing its dependence on any single feature and improving its ability to generalize to new, unseen data. By preventing overfitting, dropout ensures the model learns more robust and redundant features, enhancing its performance on sparse

and noisy datasets.
Moreover, the generalization error of the model is reduced by using the Kaiming initialization technique for weight initialization. The Kaiming initialization, proposed by He et al. (2015), sets the weights of the network according to a specific distribution that ensures proper scaling of the variance, mitigating issues such as vanishing gradients and exploding gradients that can hamper the training of deep networks. For more complex neural network models, it is necessary to introduce Kaiming initialization to help model get better results. The weights are initialized as

$$W \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

where n_{in} represents the number of input units.
Becuase we added more hidden layers, performing back propagation could lead to more problems. To prevent the gradient from vanishing, we decided to use ReLU activation functions instead of sigmoid ones.
1. The sigmoid activation function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Its gradient is:

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \tag{2}$$

For large positive or negative values of x, the gradient approaches zero, leading to the vanishing gradient problem.
2. In contrast, the ReLU activation function is defined as:

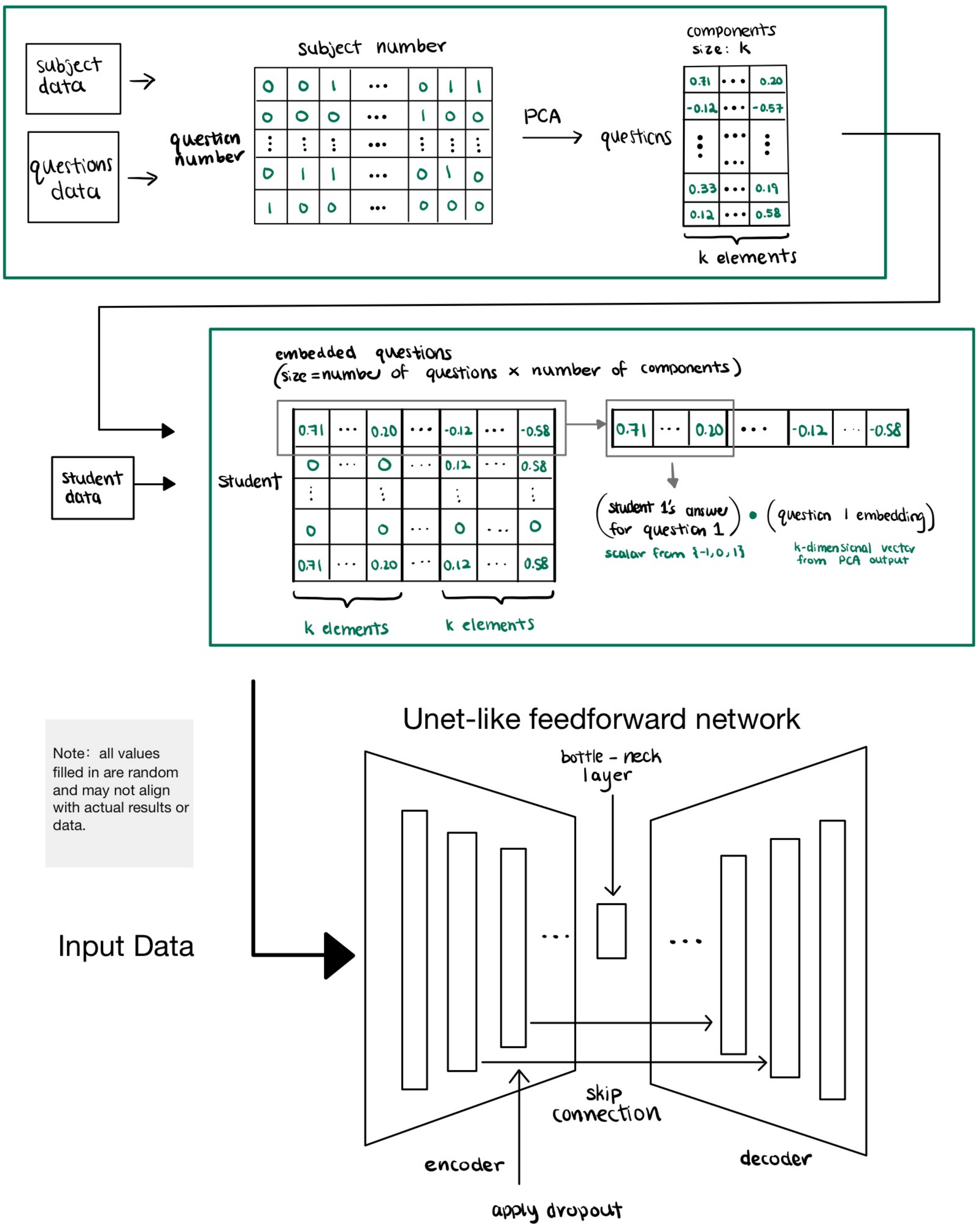
$$\text{ReLU}(x) = \max(0, x) \tag{3}$$

Its gradient is:

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{4}$$

This avoids vanishing gradients for positive inputs, allowing gradients to propagate more effectively through deep networks.
We also modified the architecture of the autoencoder model by incorporating skip connections in the decoder part, as shown in the diagram on the next page. Initially, the autoencoder was designed without skip connections, where the decoder would reconstruct the data solely based on the encoded latent representation. However, this approach often leads to a loss of fine-grained information during the decoding process, especially in cases where we have sparse data, containing NAN values. Skip connections adds the input x of a layer directly to its output $f(x)$, forming a combination $x + f(x)$. By adding skip connections, the decoder now has direct access to both low-level and high-level features, enabling it to make more accurate predictions and fill in gaps in the data more effectively.

Theoretically, these changes will enhance our model, making it predict better.



Note: all values filled in are random and may not align with actual results or data.

Figure 1: Diagram Incorporating Modifications

1.2 Analysis

Model Description	Hyperparameters	Validation Accuracy	Test Accuracy
Model From Part A (one hidden layer and no metadata)	k = 50, lr = 0.01, epochs = 50	0.6908	0.6822
Original data without skip connections but 7 hidden layers + using ReLU in inner layers	k = 50, lr = 0.01, epochs = 12	0.5164	0.5196
Original data with skip connections and 7 hidden layers + using ReLU in inner layers	k = 50, lr = 0.01, epochs = 12	0.5319	0.5461
Metadata PCA with skip connections and 7 hidden layers + using ReLU in inner layers	**k = 50, lr = 0.01, epochs = 12**	**0.6715**	**0.6819**
Metadata PCA with skip connections and 7 hidden layers + using Sigmoid in inner layers	k = 50, lr = 0.01, epochs = 12	0.5155	0.5199
Metadata PCA with one hidden layer	k = 50, lr = 0.01, epochs = 12	0.6537	0.6534
Metadata PCA without skip connection layer but 7 hidden layers + using ReLU in inner layers	k = 50, lr = 0.01, epochs = 12	0.5846	0.5854

Table 1: Model Results with Different Configurations

Results Analysis
Addition of Metadata:
From the results, the addition of new metadata to the model from Part A decreased the test accuracy from 0.6822 to 0.6534. A possible explanation could be that the input data becomes more complex and may hence require more sophisticated models to match and process the new information. The integration of additional questions metadata increases the dimensionality of the input space, making it harder for simpler models to extract relevant patterns and generalize effectively. Consequently, the model may struggle to balance learning both the original features and the new metadata, leading to a decline in overall performance.
Addition of Skip-Connection Layer:
From the results, it can be observed that the addition of the skip connection layer successfully improved the model's performance. When the model is performed on no additional metadata without skip connections, it has a test accuracy of 0.5196, compared to with skip connections' test accuracy of 0.5461. This hypothesis is also supported by the difference between the accuracy of the additional metadata without skip connections (0.5854) and the additional metadata with skip connections (0.6819). The addition of hidden layers and skip connections to the autoencoder significantly improves the model's ability to handle the augmented input data by facilitating the flow of gradients through deeper layers. The skip connection layer plays a crucial role by mitigating the vanishing gradient problem, allowing the model to retain and propagate essential features from earlier layers. This helps the model converge more effectively during training, aiding in finding

an optimal solution and improving generalization to unseen data. Thus, the additional layers with the skip connections together help the model leverage the more complex input information provided by the metadata, leading to improved performance on the test sets.
Addition of Hidden Layers:
The results indicate that the direct addition of hidden layers to the augmented input data led to a decrease in test accuracy from 0.6534 to 0.5854. While one may argue that the addition of layers led to excessive overfitting and worsened the model's predictions, we can explain this phenomenon otherwise using our results. The model, with the added complexity from the extra layers, may have struggled to converge to an optimal solution during backpropagation because the model had to navigate a more complex loss surface, which can initially hinder convergence. However, these additional layers are valuable because the data itself becomes more complex, and the additional layers help the model capture intricate patterns and dependencies within the data. Once the skip connections were introduced, the model's performance improved significantly, increasing the test accuracy to 0.6819. This suggests that the gradual reduction in dimensions during the encoding process, followed by an increase in dimensions during decoding, enables the model to effectively capture and reconstruct the input data, facilitating better learning and generalization. It can better capture the more advanced feature patterns.
Use of ReLU instead of Sigmoid Activation:
Our hypothesis that ReLU will perform better is validated by the results, with the model achieving a test accuracy of 0.6819 compared to 0.5199 with Sigmoid. This improvement is likely due to the fact that ReLU mitigates the vanishing gradient problem during backpropagation, allowing gradients to flow more effectively through deeper layers. In contrast, the Sigmoid function causes gradients to shrink, leading to slower learning and reduced performance in deeper networks.
Use of PCA:
Without dimension reduction, we tried to train the model on the dataset. However, we encountered the error message "CUDA out of memory," indicating that the system was unable to handle the large size of the input data. Specifically, with each question embedding having a dimension of 1774, the input data matrix becomes extremely large, with a shape of (542) * (1774 * 388), which translates to over 365 million elements. This size exceeds the memory capacity of the available hardware, making it impossible to train the model efficiently. This issue underscores the importance and relevance of techniques like PCA, which helps reduce the dimensionality of the data while retaining as much of the variance as possible. By applying PCA, we can significantly reduce the size of the input data, allowing the model to fit within the available memory and train effectively, without losing important information.

1.3 Limitations of Our Approach

- The addition of multiple encoder and decoder layers, along with skip connections, while may allow the model to learn more meaningful information, also introduces challenges that can limit the effectiveness of our model. While additional layers allow the network to learn hierarchical and complex patterns, they also significantly increase the model's capacity. This higher capacity can lead to over-fitting, where the model memorizes the training data rather than generalizing to unseen data. Even with dropout regularization, the depth of the network makes it more prone to relying on spurious correlations, particularly in sparse data settings. Skip connections, while useful for preserving low-level information, can also create limitations in this context. In the context of sparse data, noise and irrelevant information can be present in the lower layers. Skip connections allow this noise to propagate through the network, potentially leading to poor generalization. The model might inadvertently "memorize" these noisy patterns rather than learning meaningful relationships.

- PCA assumes linearity in the relationships among features. In the case where the underlying relationships among features are non-linear, PCA may fail to effectively reduce dimensionality, and instead discard important information. In our case, the selection of the number of components, k, was based on practical constraints such as runtime and GPU limitations, we did not perform exhaustive testing across all possible values of k (388 possibilities). As a result, the chosen value may not be the most optimal one. Hence, in such a context, using PCA may also limit the model's performance.

- Sensitivity to Hyperparameters: The performance of feed-forward neural network depends heavily on the choice of hyperparameters (e.g. number of layers, number of neurons). It requires extensive tuning. Due to time limitation, we may not find the optimal choice of hyperparameters.

Hence, some improvement that could be possibly made to the model are:

- Data Agumentation: Address the sparsity of the data by performing data augmentation methods or smoothign techniques to artificially increase the amount of training data. For instance, adding noise or perturbing the input features slightly could make the model more robust.

- Can try t-SNE or other autoencoders to perform dimensionality reduction instead of PCA to capture more complex, non-linear relationships in data. These methods may preserve more meaningful variance, especially in the case where linear assumptions of PCA fail.

- Potentially using attention mechanisms or gating mechanisms could help the model better decide when to use lower-level features, reducing the possibility of propagating irrelevant noise.