Effective Feature Learning with Unsupervised Learning for Improving the Predictive Models in Massive Open Online Courses

Mucong Ding
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China
mcding@connect.ust.hk

Kai Yang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China
yangkai@cse.ust.hk

Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China
dyyeung@cse.ust.hk

Ting-Chuen Pong
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China
tcpong@cse.ust.hk

ABSTRACT

The effectiveness of learning in massive open online courses (MOOCs) can be significantly enhanced by introducing personalized intervention schemes which rely on building predictive models of student learning behaviors such as some engagement or performance indicators. A major challenge that has to be addressed when building such models is to design handcrafted features that are effective for the prediction task at hand. In this paper, we make the first attempt to solve the feature learning problem by taking the unsupervised learning approach to learn a compact representation of the raw features with a large degree of redundancy. Specifically, in order to capture the underlying learning patterns in the content domain and the temporal nature of the clickstream data, we train a modified auto-encoder (AE) combined with the long short-term memory (LSTM) network to obtain a fixed-length embedding for each input sequence. When compared with the original features, the new features that correspond to the embedding obtained by the modified LSTM-AE are not only more parsimonious but also more discriminative for our prediction task. Using simple supervised learning models, the learned features can improve the prediction accuracy by up to 17% compared with the supervised neural networks and reduce overfitting to the dominant low-performing group of students, specifically in the task of predicting students' performance. Our approach is generic in the sense that it is not restricted to a specific supervised learning model nor a specific prediction task for MOOC learning analytics.

CCS CONCEPTS

• **Computing methodologies → Unsupervised learning**; **Neural networks**; **Learning latent representations**; • **Applied computing → E-learning**;

KEYWORDS

Feature Learning, Learning Behavior, Unsupervised Learning, Dimensionality Reduction, Autoencoder, Long Short-Term Memory

ACM Reference Format:
Mucong Ding, Kai Yang, Dit-Yan Yeung, and Ting-Chuen Pong. 2019. Effective Feature Learning with Unsupervised Learning for Improving the Predictive Models in Massive Open Online Courses. In *The 9th International Learning Analytics & Knowledge Conference (LAK19), March 4–8, 2019, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3303772.3303795

1 INTRODUCTION

With the advancement in digital technology, massive open online courses (MOOCs) have become popular over the past decade and provide alternative ways of learning. With the current one-size-fits-all approach of MOOCs, the students need to be self-motivated and self-disciplined throughout the whole learning period without much individual guidance from the instructor, and as a consequence, many are prone to drop out of courses [5]. In terms of the high attrition rate in MOOCs, there has been a great deal of interest in providing personalized learning guidance with the help of predictive models, to improve students' motivation and the effectiveness of learning which should result in an increased retention rate [20]. The predictive models proposed in the literature [4, 6, 13], help depict the learning progress, achieve a better understanding of the learning abilities, identify the at-risk students, and provide proactive interventions. However, when building such models with supervised learning, it is still challenging to design handcrafted features that characterize students' learning behaviors such as some engagement or performance indicators, and are effective for the prediction tasks.

In MOOCs, students' learning behavior is complicated in both the time domain and the content domain (the sequence of how course materials are ordered). An instructor usually designs the course schedule according to the knowledge graph which depicts the inherent dependence and hierarchical structure among course materials. The content-based representation is important, as the course materials are what a student interacts with, and the information of the course materials can only be encoded within the content domain. However, the learning pattern hidden in the content domain is usually missed in the current research, as all the models of the prevalent dropout prediction [4, 6–8, 13, 19, 21] are devised to capture the learning pattern in the time domain. If only the temporal features are utilized, dropout prediction can only provide preliminary interventions such as sending email alerts to at-risk students, let alone providing effective pedagogic advice and support with regard to the course materials for each student to improve the effectiveness of learning. Actually, the MOOC platform allows students' learning activities including video watching, page navigation, quiz participation to be recorded as the clickstream logs. Each clickstream is a collection of records, while each record consists of student ID, interaction time and accessed course materials so that we can extract both time and content domain features from the clickstream. Because of the strong inter-dependencies between the course materials with close locations in the knowledge graph, the content-domain features exhibit contextual locality, i.e., features with close locations in the sequence are more correlated. This enables us to utilize the underlying local patterns in the content domain for a variety of prediction problems.

The knowledge provided in the course content is the most important part of MOOCs. When attaining the content mastery of a specific concept, a student needs to go to the relevant pages, watch videos and answer questions with a sequence of actions recorded in the clickstream. Therefore, in terms of the many interactions related to knowledge mastery, there is a large degree of information redundancy in the clickstream data, so that it is challenging to design a set of effective handcrafted features to discriminate between the learning patterns of the high and low performing students using the raw information in the clickstream data. Feature learning with unsupervised models is a state-of-the-art approach to this problem [2]. As it is capable of learning an effective and compact representation of the raw features, which not only simplifies the architectures of the predictive models but also helps improve the prediction performance.

In this paper, we prepare the features in the time-content domain and make the first attempt to solve the feature learning problem by taking the unsupervised learning approach to learning a compact and effective representation from the highly redundant information in the raw records. Specifically, we design a modified auto-encoder (AE) combined with the long short-term memory (LSTM) networks to learn the new features that correspond to the representation of the embedding layer. The key points of this paper are summarized as follows:

(1) We prepare raw features and analyze the learning behavior in the time-content domain.

(2) We show that neural networks outperform logistic regressions by a large margin in predicting the next chapter grade.

(3) We propose a modified auto-encoder (AE) combined with the long short-term memory (LSTM) network (**modified LSTM-AE**) and two variational auto-encoders (VAEs) to learn compact and effective representations from the raw features.

(4) Compared with the VAEs, the representation in **modified LSTM-AE** is better at discriminating the low and high performing students.

(5) With the **modified LSTM-AE**, the learned features help reduce the overfitting to the dominant low-performing group of students and improve the performance in the specific task of predicting the next chapter grade by up to 17% compared with the completely supervised neural network baselines.

2 BACKGROUND

In this section, we introduce how the course materials are managed in edX, and how to formulate the learning performance prediction problem as a supervised learning task. Then, we introduce logistic regression and design some specific neural networks as the baselines to predict students' learning performance. We analyze the high-degree of redundancy problem of handcrafted features and point out that representation learning is a potent solution. Finally, we discuss restrictions of real-time predictions, i.e., predicting alongside with the progression of a course.

2.1 Learning Activity and Assessment

The materials of a MOOC in edX are managed in a hierarchical way: $\{$ *videos*, *problems*, *others* $\} \in$ *verticals* $\in$ *sequentials* $\in$ *chapters* $\in$ *course*, where a sequential corresponds to a subsection in a chapter. In each sequential, a set of problems, videos, and other course materials are listed in the verticals. A set of assignments in the problem verticals constitute the assessments of a sequential, and the sequential grade is accordingly calculated by aggregating the weighted scores of the problem verticals according to the grading policy. The chapter grades and the course grade can be obtained similarly. The numerical grades obtained indicate the students' mastery levels of the knowledge associated with the course content.

In the content domain, video and navigation activities are the two most important learning behaviors and activities related to knowledge mastery. In this paper, we extract several raw features which aim to characterize specific behaviors related to video and navigation from the clickstream data, and summarize them in Table 1. Each feature is two-fold in the time domain to capture activities before and after the split time when the student completed the corresponding assignment. For example, the feature `load-video` is split into `load-video-prior` and `load-video-post`, where the former is the number of loading video events for a assignment before its submission, and latter is corresponding count after the submission. This two-fold separation in the time domain is useful for capturing different behaviors of students with different performances. As shown in Figure 1, the high and low performing students show different behaviors before and after the completion of assignments. Students with low performance (blue dots) have not completed enough prior-quiz studies (as they are located to the left of the blue line, their prior features are relatively small), while students with high grades (red dots) have at least completed some post-quiz studies (as they are located above the red line, their post features are large). As described above, we attempt to prepare the features

Table 1: List of Features

Feature		Explanation (in the content domain)
Navigation	navigate-forward	Number of forward navigation events for a specific assignment
	navigate-backward	Number of backward navigation events for a specific assignment
Video	load-video	Number of loading video events for a specific assignment
	play-video	Number of playing video events for a specific assignment
	pause-video	Number of pausing video events for a specific assignment
	stop-video	Number of stopping video events for a specific assignment
	seek-backward	Number of seeking video backward events for a specific assignment
	seek-forward	Number of seeking video forward events for a specific assignment
	show-subtitle	Number of showing subtitle events for a specific assignment
	hide-subtitle	Number of hiding subtitle events for a specific assignment

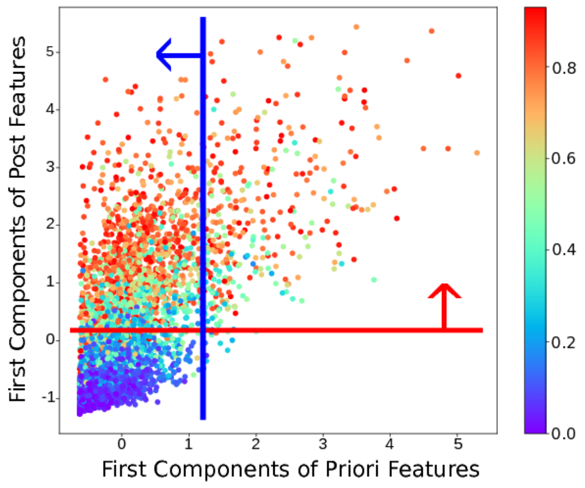

Figure 1: Principal component analysis (PCA) on the prior and post assignment features, which capture student activities before and after the time when the student completed the corresponding assignment respectively. The color shows each student's average chapter grade.

in the content-time domain, with the two-fold representation in the time domain.

The learning progress of a student in edX can be organized as sequences. Suppose there are N sets of course materials, all of which have the corresponding assessments, whose order in the course is pre-defined. For a specific student, we define the sequence of learning behaviors $(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ as the raw features, where $\mathbf{x}_i \in \mathbb{R}^F$ is a list of features characterizing his activities in the i-th set of materials, and define the sequence of grades $(y_1, ..., y_N)$ as the labels. In this paper, we prepare the set of course materials at the chapter level, and N is the total number of chapters. The sequence of assignment grades is not independent owing to the prerequisite dependency in the knowledge graph. A student who gets a good grade in the current assessment is likely to do well in the future assessments.

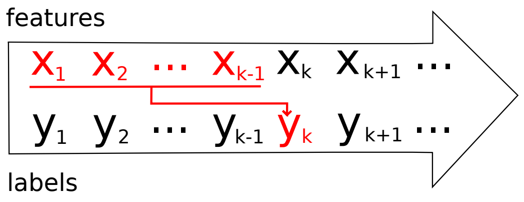
features
labels
x_1 x_2 ··· x_{k-1} x_k x_{k+1} ···
y_1 y_2 ··· y_{k-1} $\hat{y}_k$ y_{k+1} ···

Figure 2: Formulation of the performance prediction problem. For the predicted grade in chapter k, only features up to chapter $k-1$ are available.

2.2 Formulation of the Performance Prediction Problem

The performance prediction is essentially a sequence labeling problem, which uses the raw features of a student before a specific chapter k, i.e. $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}]$, to predict the student's grade in the next chapter, i.e. y_k, where $1 < k \leq N$, as shown in Figure 2. We train a list of $(N-1)$ independent models, $[f_2, \ldots, f_N]$, one at a time, so that f_k specifically predicts the grades of chapter k. The loss function of model f_k for a specific sample is defined as $\ell = (y_k - \hat{y}_k)^2$ where $\hat{y}_k$ is the predicted chapter grade and y_k is the ground truth label.

2.3 The Baselines of Performance Prediction

2.3.1 Logistic Regression. In statistics, the logistic model is a statistical model with input (independent variable) a continuous variable and output (dependent variable) a binary variable, where a unit change in the input multiplies the odds of the two possible outputs by a constant factor. Logistic regression (**LR**) is widely used for binary classification, and we consider it as a baseline model to evaluate the performance of neural networks.

2.3.2 Multi-layer Perceptron. The multi-layer perceptron (MLP) is a class of artificial neural networks based on a collection of units called artificial neurons, an analogous to biological neurons in an animal brain. With the neuron connection, MLP focuses on approximating the complex relationship between the input and output. An MLP consists of multiple fully connected layers of neurons and is able to learn the dependency between a collection of distinguishing features and the target labels. In this paper, we design the fully connected networks with three hidden layers (**FC3**) as a baseline.

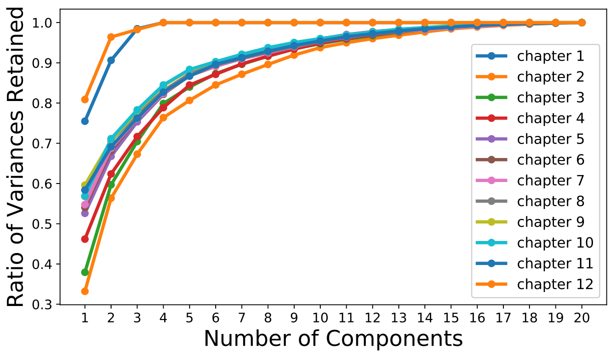

Figure 3: The dependency of the ratio of variances retained on the number of output components with the PCA for each chapter.

2.3.3 Convolution Neural Networks. The convolutional neural network (CNN) applies a group of neurons (a kernel) across a specific dimension of the data. The neurons thus are capable of learning features that are defined by local patterns possibly occurring anywhere in the input. When the data is heterogeneous, it is meaningless to apply the convolutional filter along a list of distinguishing features. For our content-domain features, because the order of course materials preserves the implicit dependence on the knowledge graph, we can apply one-dimensional convolution to the content domain to find the implicit learning patterns. For most of the MOOCs, since the number of chapters is limited to $N \leq 12$, we always set the kernel size to 3. We design a neural network with two convolutional layers followed by a fully connected layer (**CNN2-FC1**) as a baseline.

2.3.4 Long Short-Term Memory Networks. Recurrent neural networks (RNNs) are a class of network structures in which the neurons at each layer are not only connected to neurons of adjacent layers but also receive the input from the same layer at the previous step in a sequence. With these connections across steps in sequential data, RNN is capable of learning patterns that change dynamically over that dimension. The RNNs with long short-term memory (LSTM) cells are used to overcome the gradient vanishing problem in the vanilla RNN and is now a widely used standard in sequence learning problems. In our case, LSTM networks are potent to capture the hidden learning patterns within the content domain. Here, we design a neural network with one LSTM layer (**LSTM1**), and another model consists of a one-dimensional CNN of kernel size 1 followed by an LSTM layer (**CNN1-LSTM1**) as baselines.

2.4 Feature Redundancy and Representation Learning

Feature redundancy is a common problem of hand-crafted features in the context of MOOC data analytics. Even for the simple purpose of learning specific content, students need to go through a collection of relevant pages (e.g., watching videos and answering quizzes), which triggers a sequence of interdependent actions recorded in the clickstream. In this regard, the underlying correlations might be significant among different types of interaction events, and the manually extracted features could be highly redundant. This gives rise to the increasing difficulty of training a robust predictive model. For the specific set of features we extracted in this paper, the redundancy problem can be shown by performing principal component analysis (PCA) directly on the 20 features of a specific chapter k. From Figure 3 we can see that 5 components can retain over 80% of the variances for all chapters.

Unsupervised learning using auto-encoders (AEs) for finding a distinctive representation of the raw features is a potential solution to the feature redundancy problem. Since our features and labels are essentially sequences, the auto-encoding of features is a sequence-to-sequence mapping problem. And thus we should search for an unsupervised architecture which is capable of extracting the underlying representations of general sequences. Outside of the educational context, a general sequence-to-sequence learning framework is used in natural language processing and video representation learning, where a long short-term memory (LSTM) network is used to encode a sequence into a fixed-length representation, and then another LSTM network is used to decode a sequence out of that representation [17]. Following this well-established scheme, we explore AEs combined with LSTM networks to obtain a compact embedding of MOOC data in the educational context, which also aims to be efficient for prediction problems.

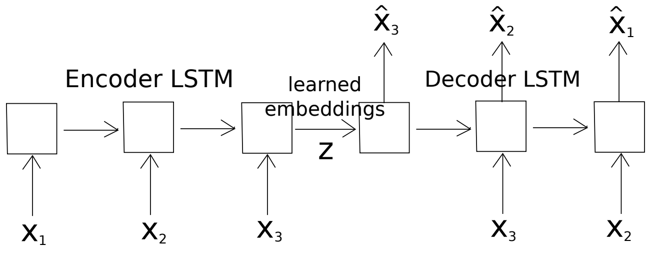

Figure 4: Architecture of the LSTM AE when $k = 4$.

3 REPRESENTATION LEARNING

In this section, we propose our modified LSTM auto-encoder for representation learning. We also list two other designs of variational auto-encoders (VAEs) with the symmetric and asymmetric structures for comparison purposes.

3.1 The Modified LSTM Auto-encoder

Inspired by the unsupervised LSTM auto-encoder which has achieved a big success in video representation learning [16], we propose a modified LSTM auto-encoder (denoted by **Modified LSTM-AE**) for learning efficient and compact representations of feature sequences in the context of MOOC data analytics.

We first describe the LSTM auto-encoder model, which combines the well-established sequence-to-sequence learning framework [17] and the idea of using auto-encoders for representation learning. It consists of two LSTM networks, the encoder, and the decoder LSTM, as shown in Figure 4. The features in the input sequence $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{k-1})$ are fed into the encoder LSTM one at a time. Right after the last input $\mathbf{x}_{k-1}$ has been read, the last output from the encoder, z, is the learned embedding and is forwarded to the decoder LSTM as the first input. The decoder then reads in the original sequence (except from $\mathbf{x}_1$) in reverse order, $(\mathbf{x}_{k-1}, \ldots, \mathbf{x}_2)$. Its output $(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-2}, \cdots, \hat{\mathbf{x}}_1)$ aims to reconstruct the input sequence. Because of the contextual locality of the content domain (features with close locations in the sequence are more correlated), it is much easier for the decoder to reconstruct the input sequence in reverse order. And hence the entire LSTM auto-encoder adopts the last-in-first-out (LIFO) scheme. Note that no matter how long the input sequence is, the learned representation has a fixed length, which prohibits the auto-encoder to learn a trivial identity mapping.

We could think of our LSTM auto-encoder as first recursively performing an LSTM operation (defined by the weights of the encoder LSTM) to encode, and then recursively performing another LSTM operation to decode. In this sense, features close to the turning point (chapter k) has the potential to dominate the learned embedding, as some features in the early chapters decay exponentially during the recursive encoding. This characteristic helps the LSTM auto-encoder learn an efficient representation for real-time predictions (predicting labels in the next chapter) tasks. We expect features close to chapter k are more useful in prediction because of the contextual locality, while the local patterns in early sequences could be meaningless. Note that the decoder is conditioned on the ground truth features (i.e., we input the ground truth features to the decoder again), which enables this LSTM auto-encoder to capture multiple modes in a sequence.

The real-time framework does not forbid us to input the feature sequences after chapter k into the decoder part (see Sec.3.4 for our detailed argument). Thus a notable short-coming of the LSTM auto-encoder is that it does not utilize this subsequence of ground truth features, $(\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \cdots, \mathbf{x}_N)$, to enhance the predictive power of the embedding. As we merely require the model to reconstruct the input sequence, the learned representation only contains information collected before chapter k, and this is the main restriction of the effectiveness of our learned representation.

We could improve the design by adding another predicting decoder parallel to the original reconstructing decoder, which aims to predict the sequences of features in the later chapters at the same time (see Figure 5). If this decoder could predict the features in the next few chapters correctly, the encoder must have captured some latent patterns of the input sequences, and such information is contained in the learned embedding. This prohibits the encoder to ignore all of the useful information in the early part of the sequence. Moreover, during the training process, some information from the subsequence following chapter k could be captured by the weights of encoder LSTM though stochastic gradient descent based optimization. Hence, the embedding can find useful patterns from the entire sequence now, even though we only input the subsequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1})$ to the encoder. This greatly eases the difficulty of predicting with the embedding, since our encoder now has some prediction capability as well. Similarly, we also provide the ground truth features to the reconstructing decoder to help it extract multiple modes from the sequence.

One small defect of our current design is that our embedding, $\mathbf{z}$, must have exactly the same size as the feature units, $\mathbf{z} \in \mathbb{R}^Z$ and $Z = F$, otherwise the decoders cannot take both of them as input. This is a restriction that we want to get rid of. Actually, we can

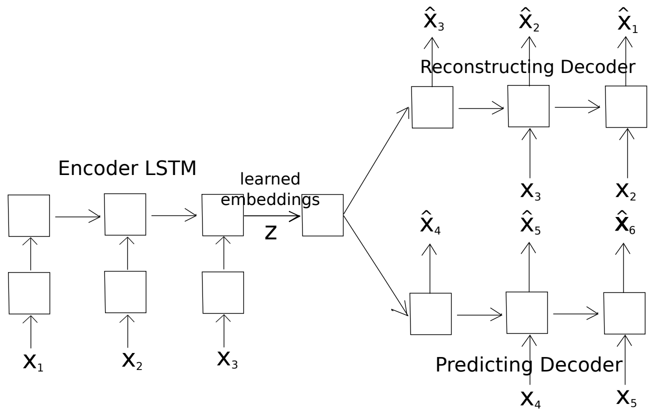

Figure 5: Architecture of the Modified LSTM-AE model when $k = 4$ and $N = 6$.

further shrink the size of the embedding space by applying a one-dimensional convolutional layer of kernel size 1 before the LSTM layer in the encoder. This layer further reduces the dimensionality of the inputs before extracting the underlying patterns. At the same time, we add one fully connected layer before the two decoders to map the small embedding space $\mathbb{R}^Z$ to $\mathbb{R}^F$, where $Z < F$. The complete design is shown in Figure 5. Note that this architecture is also capable of enlarging the embedding space to $\mathbb{R}^Z$ where $Z > F$ under the situation where F is too small for learning a good representation.

The loss of this **Modified LSTM-AE** model is generally defined as the weighted mean squared error (MSE) between the complete input sequence $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ and the output $[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N]$. In order to further encourage our model to learn a representation which utilizes the contextual locality around the turning point, we assign a set of Gaussian weights $\exp[(k-n)^2/2\sigma^2]$ to each of the MSEs between the pair $(\mathbf{x}_n, \hat{\mathbf{x}}_n)$, $(1 \le n \le N)$, where σ is usually set to 3. For a single sample, the loss is defined as $\ell = \sum_{n=1}^{N} \exp[(k-n)^2/2\sigma^2](\mathbf{x}_n - \hat{\mathbf{x}}_n)^2$. You can see that we assign larger weights to MSEs around chapter k. This forces our model to treat learning the local representation as the first priority.

3.2 The Baselines of Representation Learning

Variational auto-encoder (VAE) is a popular approach to learning robust embeddings for various types of data. Here, we build two VAEs as baselines to compare with the proposed **Modified LSTM-AE**. The first VAE we build (denoted by **Symmetric-VAE**) processes a symmetric structure [2]. As shown in Figure 6, both the encoder and the decoder of it is one layer of the bi-directional LSTM network. Where in a bi-directional LSTM, we connect two hidden layers of opposite directions to the same output. Thus future input information is reachable from the current state. the another VAE, the encoder consists of three one-dimensional convolutional layers (Figure 7). We call it **Asymmetric-VAE**, since it has an asymmetric structure and is capable of extracting more low-level feature embeddings [12].

The biggest difference between the VAEs and our **Modified LSTM-AE** model is that they do not have a fixed-length embedding.

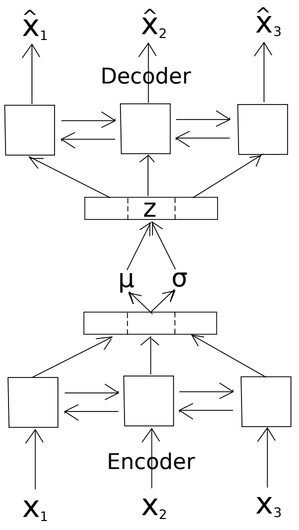

Figure 6: Architecture of the Symmetric-VAE model when $k = 4$.

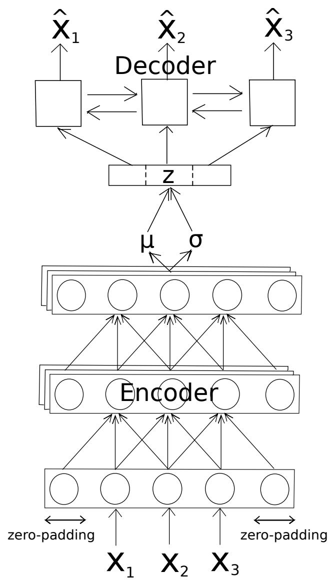

Figure 7: Architecture of the Asymmetric-VAE model when $k = 4$.

In the VAEs, the content domain is preserved in the embedding layer with the same sequence length. This approach has the advantage of capturing useful patterns anywhere in the feature sequence and makes the reconstruction of the original input easier. However, the embeddings learned by those VAEs may contain many useless local patterns in the content domain which are relatively far from the chapter we are predicting. In this sense, we argue that the two VAEs are not as powerful as the **Modified LSTM-AE** model in finding efficient representations for real-time predictions. This is what we see in the experiments.

3.3 Prediction with Learned Representations

Equipped with the unsupervised model which learns a good representation of the feature sequences, we could build a relatively simple predictor which takes the learned representations as the new feature inputs. For the **Modified LSTM-AE** model, we use a fully connected network with one hidden layer as the predictor. For the two VAEs, since the embedding is still a sequence, we use one layer of LSTM network as the predictor.

When training these predictors, it is possible to fine-tune the weights of the pre-trained encoders to enhance the representation for a specific prediction task further. After fine-tuning, the portion of useful features for this specific task in the embedding space could be further enlarged. During such a training process, a much smaller learning rate should be used for the encoder part, as we expect the pre-trained encoder can already find a very efficient representation for a variety of prediction problems.

3.4 Real-time Interventions Based on the Predictive Model

Prediction along with the progression of a course (i.e., predicting students' behaviors in the next chapter k when they are learning chapter $k - 1$) is crucial for real-time interventions. This one-step ahead foresight can provide students with the effective pedagogic support on the current course material at the right time. However, real-time prediction also imposes a strong restriction on the training setups, in the sense that only features and labels up till chapter $k - 1$ of the current course are available to us when we are making predictions for chapter k. As a consequence, the typical training procedure which requires labels defined in chapter k does not work on a course in progress. Instead, we have to learn the model retrospectively on data generated from a finished course [3]. Under this transfer learning scenario, although all the data is actually available, when we train a predictor on a completed course, we are not allowed to input features in or after chapter k to the model, because we cannot do the same thing when making predictions using this trained predictor on an ongoing course. However, when we train an unsupervised model on a finished course, we can input the ground truth features in or later than chapter k to the decoder part (like what we do in **Modified LSTM-AE**), as long as its encoder part only reads in the set of features before chapter k. This is because only the encoder which generates the learned embedding is used when making predictions, and as long as we do not make use of the features in and after week k when predicting on an ongoing course, we do not break the causality.

4 EXPERIMENTS

In this section, we first list some details of the experimental setups and then present the complete model comparison results.

4.1 Experimental Setups

4.1.1 The Data Set. We train all of the baseline predictors and unsupervised models on a 12-chapter-long Open edX MOOC, named "Introduction to Computing with Java", which was held by the *Hong Kong University of Science and Technology* from June 2014 to September 2014. There are 44,920 students enrolled but only around one-ninth of them completed the entire course. As described in

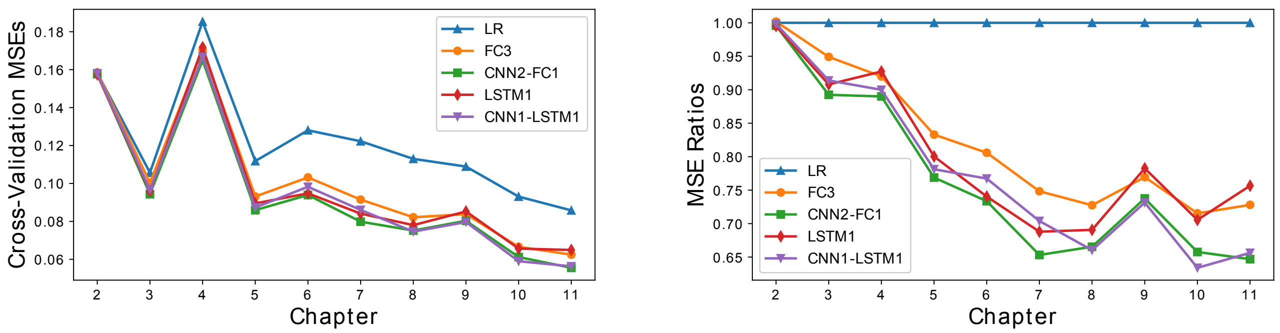

Figure 8: The left figure shows the comparison of prediction performance among supervised baselines. The right figure shows the relative prediction performance compared to logistic regression (LR).

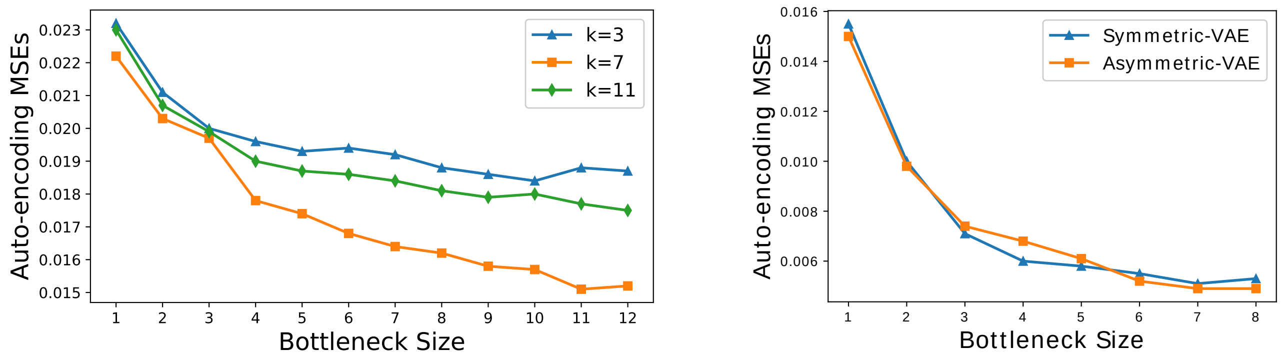

Figure 9: The left figure shows the auto-encoding mean-squared errors (MSEs) of Modified LSTM-AE with different bottleneck sizes for chapter $k = 3, 7$ and 11. The right figure shows the auto-encoding MSEs of Symmetric-VAE and Asymmetric-VAE with different bottleneck sizes for chapter $k = 12$.

Sec.2.1, we prepared 20 features ($F = 20$) and 1 label for each student in each chapter. All of the features and labels are rescaled to $[0, 1]$ respectively. The normalization is for each feature/label across all students. For each student, the sequence of features are $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, where $N = 12$ is the number of chapters. For $1 \leq n \leq N$, we have $\mathbf{x}_n \in [0, 1]^F$, where $F = 20$ is the number of features. The labels also form a sequence $[y_1, y_2, \ldots, y_N]$, where each $y_n \in [0, 1]$ (except from y_{12} which is undefined, since there is no assessment in the last chapter of the course). Only students with valid labels at the chapter level are considered. Under such a criterion, our dataset contains 5,739 students.

4.1.2 Model Comparison Scheme. We compare the effectiveness of the learned embeddings of the three unsupervised models by means of the principal component analysis (PCA), the t-distributed stochastic neighbor embedding (t-SNE) [14] and the final examination by evaluating the performance improvements on a specific grade prediction.

PCA is a technique that offers the reduction of a large set of correlated variables to a smaller number of uncorrelated hypothetical components [10]. It is widely used for feature extraction and selection. The set of output components are ranked in decreasing order of variances, and thus the first several components are most informative. In our experiments, we evaluate the effectiveness of the learned embeddings in the two-dimensional projected space composed of the first two components output by PCA, by visualizing how samples are distributed in the embedding space. If there are clearly formed clusters and our labels are discriminative for distinguishing samples in different clusters, the learned embedding should be discriminative for predicting the label as well. We also compute the percentage of variance retained by the first several components obtained from PCA, to have a sense of how many components are useful and how compact our embeddings could be.

t-SNE as a machine learning algorithm for dimensionality reduction which has also been widely and successfully applied to visualize the learned representations. It is a variation of Stochastic Neighbor Embedding [9] that produces visualizations by reducing the tendency to crowd points together in the center of the map. As a nonlinear dimensionality reduction technique, it is particularly well-suited for embedding high-dimensional data into a space of two or three dimensions. In this paper, we use t-SNE as another method to check the effectiveness of the representation learning.

The comparison of prediction performance is carried out as follows. Recall that for each predictor, we trained a sequence of independent models $f = [f_2, \ldots, f_N]$. For any $1 < k \leq N$, the model f_k of chapter k takes the subsequence $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}]$ as inputs and gives the prediction $\hat{y}_{k;f} = f_k([\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}])$. When we compare the prediction performance of two models f and g, we actually compare each pair of mean squared errors (MSEs) $\text{MSE}_{f,k}$ and $\text{MSE}_{g,k}$ given a specific chapter k. Where $\text{MSE}_{f,k}$ is the average cross-validation MSE of the model f in chapter k. In this paper, we always perform five-fold cross-validation and this ensures that the average validation MSEs reflect the actual performance of a model.

4.1.3 Experimental Configurations. Since both our features and labels are scaled to $[0, 1]$, we always use the *sigmoid* function as the output activation function. For the baseline predictors defined in Sec.2.3: **FC3**, **CNN2-FC1**, **LSTM1** and **CNN1-LSTM1**, we choose a unified learning rate $l = 0.001$. While the learning rates for unsupervised models: **Modified LSTM-AE**, **Symmetric-VAE**, **Asymmetric-VAE** are larger $l = 0.004$. Following the recommendations in [1], we increase the number of nodes per layer and the number of epochs until a good fit of the data is achieved. After that, we slightly regularize our network using dropout [15] until the model no longer overfits the training data. Here we do not change the number of layers during hyper-parameter tuning, as they are our parameters of interest for model comparison. Since all of our models are not very deep, slight regularization is often adequate and we are able to fit the data well within 200 epochs. We use the *Adam* optimizer [11] for all networks without an LSTM layer, and use *RMSprop* optimizer [18] for all LSTM-based models. All networks are implemented using the *Keras* framework with *TensorFlow™* back-end.

4.2 Performance Comparison

To prove our proposed **Modified LSTM-AE** model learns a good representation, we compare our approach with the start-of-the-art techniques on finding efficient embeddings (**Symmetric-VAE** and **Asymmetric-VAE**) and with the supervised baselines on prediction performance.

4.2.1 Baseline Predictors. In the first experiment, we compare the mean squared errors (MSEs) of grade prediction of the baseline predictors. Figure 8 illustrates the relation of average validation MSEs versus the chapter number to predict for all baselines. We can clearly see that neural network models (**CNN2-FC1**, **LSTM1** and **CNN1-LSTM1**) outperform logistic regression (**LR**) by a large margin. This motivates us using neural networks since traditional regression-based models fail to predict accurately. **LSTM1** performs weaker than **CNN2-FC1** when the chapter number k is large, this is because LSTM networks which fit the data best when $k \approx 7$ tends to overfit when k is larger. Adding a dimensionality reduction layer (one-dimensional CNN of kernel size 1) (**CNN1-LSTM1**) solves this problem, where we can see the performance of it and **CNN2-FC1** are similar for large k. We identify that **CNN2-FC1** is the best predictor architecture having an excellent and stable prediction performance on nearly all chapters. We then choose it as the baseline to compare to predictors working on the embedding representations.

4.2.2 Bottleneck Sizes. Before training the unsupervised models, we should carefully choose an important hyper-parameter, the bottleneck size. For a **Modified LSTM-AE**, the bottleneck size Z is just the dimension of the embedding space $z \in \mathbb{R}^Z$. In Figure 9, we plot the cross-validation auto-encoding MSEs of the **Modified LSTM-AE** model versus the bottleneck size Z on the left-hand side, for three chapters $k = 3$, 7 or 11. The results show that for the middle chapters $k \approx 7$, the MSEs are more sensitive to the bottleneck size. For all situations, setting the bottleneck size to $Z = 8$ is enough for retaining its learning capability on our data set.

Things are completely different for the two VAEs since their embedding features are one-dimensional sequences of size $z \in \mathbb{R}^{Z \times k}$, where $1 < k \leq N$, it is expected that a small bottleneck size per unit Z should be enough. When $k = 12$ (which is the only situation that they output the entire sequence X as does **Modified LSTM-AE**), we also plot the curves of auto-encoding MSEs versus bottleneck sizes, as shown in Figure 9. We can see that $Z = 4$ is generally enough.

4.2.3 Analysis of the Embedding Representations. A direct visualization of the embedding representations learned by the **Modified LSTM-AE** model is shown in Figure 10 and Figure 11. We can clearly see that at the beginning of a course, the input features are relatively noisy and the learned embedding can hardly separate students with different grades. This situation is improved along with the progression of the course. When $k = 11$, we can identify three clusters of students corresponding to the low, medium and high performing groups respectively, with the naked eye. This serves as direct proof that our **Modified LSTM-AE** model can learn an efficient representation for the chapter grade prediction task.

We also calculate the percentages of variances retained after PCA (when the number of out components is 4) for all of the three unsupervised models in the case of $k = 8$. We keep the sizes of the embedding spaces to be exactly the same so that these ratios are comparable. For the **Modified LSTM-AE** model, the percentage of variances in the first 4 components is as high as 97.85%, while for the **Symmetric-VAE** and **Asymmetric-VAE**, they are 79.84% and 84.72% respectively. A higher retained variance ratio implies a smaller effective size (i.e., the number of important components) of the learned embedding. Thus we conclude that **Modified LSTM-AE** also learns the most compact representation.

4.2.4 Predictors on Learned Embedding. Now to examine how much the prediction performance is improved by the learned embeddings. For each unsupervised model, we train a predictor on a pre-trained encoder and fine-tune the encoder's parameters with a relatively small learning rate (one-tenth of the learning rate on the predictor part of the model). This training methodology is equivalently used for all of the three predictors on embeddings. Figure 12 illustrates the average validation MSEs for grade prediction in each chapter for the three predictors with different embeddings, and also the results of the supervised baseline, **CNN2-FC1**. Where we can see the representation learned by **Modified LSTM-AE** performs the best, as it outperforms the other two VAEs for all chapters by a large margin. The improvements in the early chapters are limited for all of the three unsupervised models because further reducing the

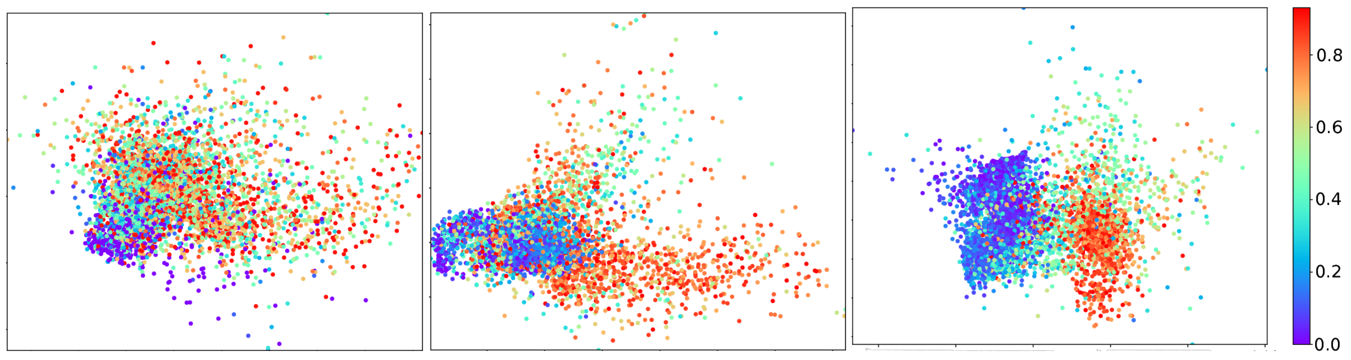

Figure 10: PCA projections of the embeddings learned by Modified LSTM-AE. From left to right, chapter k is 3, 7 and 11 respectively. The color labels the student's average chapter grade up till chapter k.

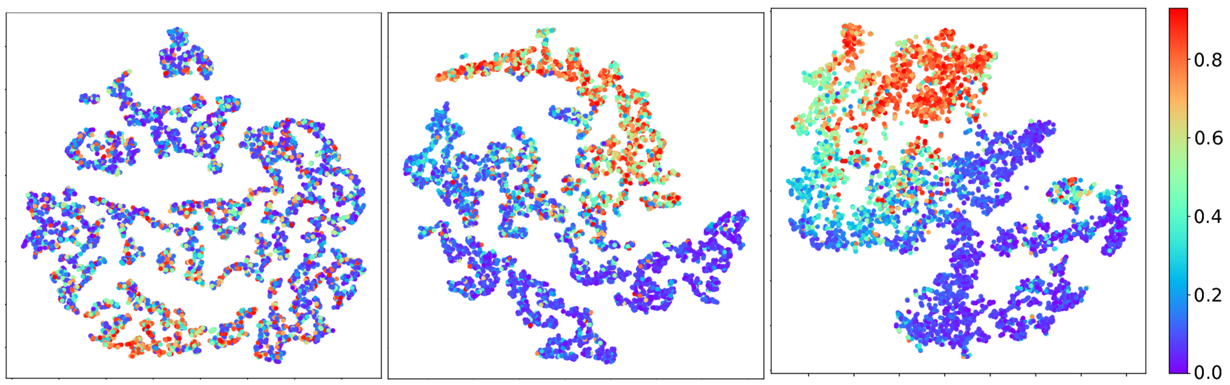

Figure 11: t-SNE projections of the embeddings learned by Modified LSTM-AE. From left to right, chapter k is 3, 7 and 11 respectively. The color labels the student's average chapter grade up till chapter k.

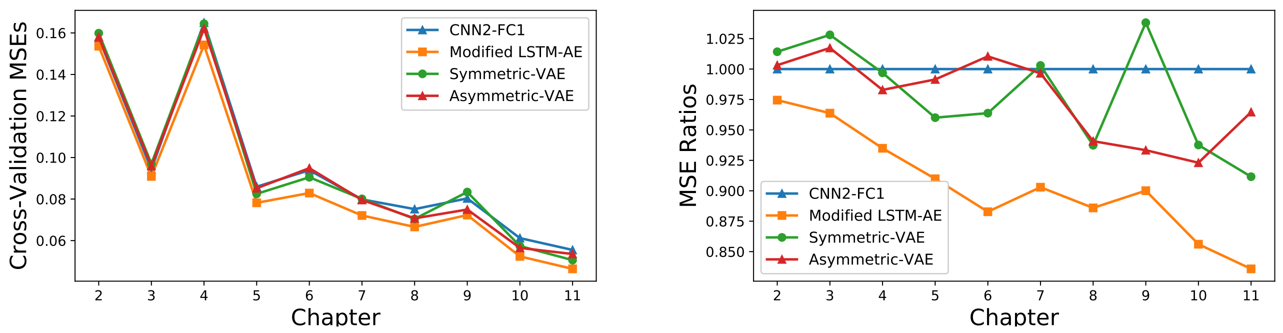

Figure 12: The left-hand side figure shows the difference between prediction performance using either the learned features or the raw features. The right-hand size figure shows the relative prediction performance compared to CNN2-FC1 using the raw features.

dimension of the short feature sequences might remove some useful information that is effective for the prediction problem. We also examine the performance on students with different average grades, as depicted in Figure 13 where we can see that the predictor using **Modified LSTM-AE** embedding features achieves significantly smaller MSEs for the medium and high performing groups, compared with **CNN2-FC1**. Although these two groups have much smaller numbers of students, the **Modified LSTM-AE** does not overfit to the dominant low-performing group. It helps reduce the modeling overfit of the supervised methods. We summarize that **Modified LSTM-AE** performs consistently and significantly better

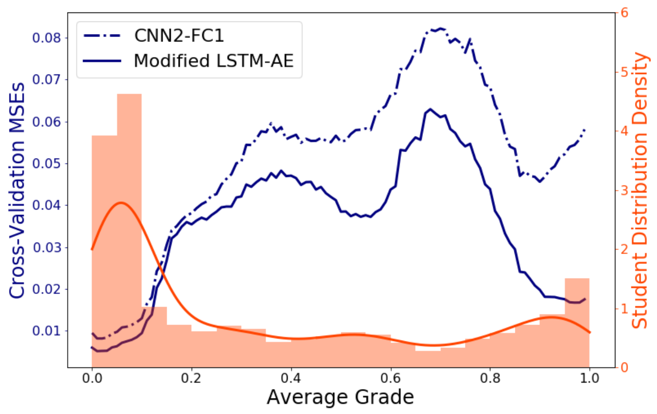

Figure 13: Average prediction performance of CNN2-FC1 (blue dashed line) and the predictor using the Modified LSTM-AE embedding (blue solid line) for students with different average grades in all chapters. The three peaks of the distribution density curve (orange line) refer to the low, medium and high performing groups respectively.

than the other unsupervised models on improving the real-time grade prediction, where up to 17% MSEs could be reduced on our data set.

5 CONCLUSION

In this paper, we report the effective feature learning with unsupervised learning approach for improving the predictive models in MOOCs. In the field of learning analytics, there is a large degree of freedom of designing the handcrafted features for improving the prediction performance in a specific task with an abundance of the clickstream data. Different from those traditional approaches of feature engineering, we attempt to describe students' learning activities in the time-content domain with the raw interaction records in the clickstream. We design the **Modified LSTM-AE** model which can learn a compact representation that is discriminable for the target labels of performance indicators. In our experiment, the **Modified LSTM-AE** successfully gives the effective features, which are indeed helpful for improving the prediction performance of the task of predicting students' learning performance, and reducing the modeling overfit to the low-performing majority. An open source release of our pipeline will be published.

For future work, we will examine our approach on more courses when the data is available. Although a specific prediction objective is used to prove that the learned features are effective, our feature learning pipeline is not restricted to a specific supervised learning model nor a specific prediction task. We will also test the representation learning methods on different prediction tasks in the future. To extend our work further, we can attempt other approximations for preparing the feature in the time-content domain. Our features are two-fold in the time domain as a reasonable approximation, some learning patterns in the time domain cannot be identified. A new unsupervised learning approach is needed to learn the effective features that characterize the learning patterns in both the time domain and the content domain.

REFERENCES

[1] Yoshua Bengio. 2012. *Practical Recommendations for Gradient-Based Training of Deep Architectures.* Springer Berlin Heidelberg, Berlin, Heidelberg, 437–478. https://doi.org/10.1007/978-3-642-35289-8_26

[2] Nigel Bosch. 2017. Unsupervised Deep Autoencoders for Feature Extraction with Educational Data. In *Proceedings of the EDM 2017 Workshops and Tutorials co-located with the 10th International Conference on Educational Data Mining.* EDM, Urbana, IL, USA.

[3] Sebastien Boyer and Kalyan Veeramachaneni. 2015. Transfer Learning for Predictive Models in Massive Open Online Courses. In *Artificial Intelligence in Education.* Springer International Publishing, Massachusetts Institute of Technology, 54–63.

[4] Devendra Singh Chaplot, Eunhee Rhim, and Jihie Kim. 2015. Predicting Student Attrition in MOOCs using Sentiment Analysis and Neural Networks. In *AIED Workshops.* AIED, Seoul, South Korea.

[5] T. Daradoumis, R. Bassi, F. Xhafa, and S. Caballé. 2013. A Review on Massive E-Learning (MOOC) Design, Delivery and Assessment. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing.* 3PGCIC, Mytilini, Greece, 208–213. https://doi.org/10.1109/3PGCIC.2013.37

[6] M. Fei and D. Y. Yeung. 2015. Temporal Models for Predicting Student Dropout in Massive Open Online Courses. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW).* ICDMW, Hong Kong, China, 256–263. https://doi.org/10.1109/ICDMW.2015.174

[7] Sherif Halawa, Daniel Greene, and John Mitchell. 2014. Dropout prediction in MOOCs using learner activity features. *Proceedings of the Second European MOOC Stakeholder Summit* 37, 1 (2014), 58–65.

[8] Jiazhen He, James Bailey, Benjamin IP Rubinstein, and Rui Zhang. 2015. Identifying At-Risk Students in Massive Open Online Courses. In *AAAI.* AAAI, Melbourne, Australia, 1749–1755.

[9] Geoffrey E Hinton and Sam T Roweis. 2003. Stochastic neighbor embedding. In *Advances in neural information processing systems.* NIPS, Toronto, Canada, 857–864.

[10] I. T. Jolliffe. 1986. *Principal Component Analysis and Factor Analysis.* Springer, New York, NY, 115–128. https://doi.org/10.1007/978-1-4757-1904-8_7

[11] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[12] Severin Klingler, Rafael Wampfler, Tanja Käser, Barbara Solenthaler, and Markus Gross. 2017. Efficient Feature Embeddings for Student Classification with Variational Auto-encoders. In *Proceedings of the 10th International Conference on Educational Data Mining.* EDM, ETH Zurich, Switzerland, 72–79.

[13] Marius Kloft, Felix Stiehler, Zhilin Zheng, and Niels Pinkwart. 2014. Predicting MOOC dropout over weeks using machine learning methods. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs.* EMNLP, Berlin, Germany, 60–65.

[14] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[16] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised Learning of Video Representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research),* Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 843–852.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27.* Curran Associates, Inc., Mountain View, CA, USA, 3104–3112.

[18] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[19] Jacob Whitehill, Kiran Mohan, Daniel Seaton, Yigal Rosen, and Dustin Tingley. 2017. MOOC Dropout Prediction: How to Measure Accuracy?. In *Proceedings of the Fourth (2017) ACM Conference on Learning@Scale.* ACM, L@S, Worcester, MA, USA, 161–164.

[20] Jacob Whitehill, Joseph Jay Williams, Glenn Lopez, Cody Austun Coleman, and Justin Reich. 2015. Beyond prediction: First steps toward automatic intervention in MOOC student stopout. In *Proceedings of the 8th International Conference on Educational Data Mining.* EDM, Worcester, MA, USA.

[21] Cheng Ye and Gautam Biswas. 2014. Early prediction of student dropout and performance in MOOCs using higher granularity temporal information. *Journal of Learning Analytics* 1, 3 (2014), 169–172.