Temporal Models for Predicting Student Dropout in Massive Open Online Courses

Fei Mi
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
Email: fmi@cse.ust.hk

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

***Abstract*—Over the past few years, the rapid emergence of massive open online courses (MOOCs) has sparked a great deal of research interest in MOOC data analytics. Dropout prediction, or identifying students at risk of dropping out of a course, is an important problem to study due to the high attrition rate commonly found on many MOOC platforms. The methods proposed recently for dropout prediction apply relatively simple machine learning methods like support vector machines and logistic regression, using features that reflect such student activities as lecture video watching and forum activities on a MOOC platform during the study period of a course. Since the features are captured continuously for each student over a period of time, dropout prediction is essentially a time series prediction problem. By regarding dropout prediction as a sequence classification problem, we propose some temporal models for solving it. In particular, based on extensive experiments conducted on two MOOCs offered on Coursera and edX, a recurrent neural network (RNN) model with long short-term memory (LSTM) cells beats the baseline methods as well as our other proposed methods by a large margin.**

I. Introduction

The rapid emergence of some massive open online course (MOOC) platforms and many MOOCs provided on them has opened up a new era of education by pushing the boundaries of education to the general public. Due to the open and online nature of MOOCs, they have attracted many people around the globe to take advantage of this new learning opportunity. However, unlike in traditional brick-and-mortar classrooms, students enrolled in MOOCs often show a much wider range of motivations and engagement styles with many of them not motivated enough to complete a course. Consequently, an undeniable fact is that MOOCs generally show very high dropout rate [1], [2]. This problem puts a major hurdle to the transformative potential of MOOCs.

The dropout prediction problem is challenging for several reasons. First, students engage in a MOOC for different purposes. A student who does not access the course content often may indicate underperformance and hence need help, but it is also possible that she already knows the topic concerned very well. As such, the data for dropout prediction is very *noisy*. Second, MOOC platforms typically log a wide range of student activities but only a very small fraction might be relevant to dropout prediction. Consequently the data appears very *sparse*. Third, as the dropout rate is often as high as 60-80%, the data is typically quite *imbalanced* with dropout students significantly more than those who stay through. One should pay special attention to this class imbalance issue when training a machine learning model for dropout prediction.

Before we proceed to look at the detailed formulation and analysis of the dropout prediction task, let us remark that there is no universally accepted definition of dropout for MOOCs. Different definitions have been adopted by different research groups in previous work. In this paper, we do not stick to just one definition but consider three different ones to capture different contexts of the student status in a course, including both final and intermediate stages. We contend that a powerful and robust dropout prediction model should be verified thoroughly based on different dropout definitions. To set the stage for presenting different dropout prediction models in later sections, we note that dropout prediction can be seen as a sequence labeling problem [3] or a time series prediction problem [4]. Recognizing this nature of the problem can help to justify our approach of adopting temporal models, which include the input-output hidden Markov model (IOHMM) [5] and the recurrent neural network (RNN) model [6].

The main contribution of this paper as follows:

- We view dropout prediction from a sequence labeling perspective and apply temporal models to solve the problem.
- Using an RNN model with long short-term memory (LSTM) cells, we obtain significantly better dropout prediction results than other methods for all three definitions of dropout.

To the best of our knowledge, our work makes the first attempt on both aspects for the dropout prediction task.

II. Related Work

We first review some features commonly used for this task. Although different research groups often use different features, they usually correspond to the same set of underlying characteristics. Some raw behavioral features that most existing MOOC platforms keep track of include *Video Watching, Page Viewing, Forum Viewing and Posting, Wiki Page Viewing and Editing, Video Click-stream*. Most researchers used a flat feature structure [7], [8], [9], [10], [2]. For example, if we predict dropout on a weekly basis, the features logged in the second week will be directly concatenated or added to the features logged in the first week, and similarly for later weeks.

In terms of the machine learning models used, some systems [7], [8], [11] applied SVM to the features extracted

to predict the last week of active participation of a student. Another system [10] applied logistic regression to similar features to predict student dropout according to the last activity of a student in the course. An attempt was also made [2] to modify logistic regression by adding a regularization term to reduce the difference in failure probabilities between consecutive weeks to predict whether a student will fail a course at the end. A decision tree was used to predict dropout and perform feature analysis [9]. Using probabilistic soft logic (PSL), the dropout prediction task was formulated in an intuitive, logic-like language [1]. In [12], a survival model was developed to measure the influence of factors related to student behavior and social positioning within discussion forums using standard social network analysis techniques. Besides using a single model, a composition and ensemble of classification results given by the naive Bayes (NB) classifier, multilayered perceptron (MLP), SVM, and decision table (DT) was also used to give the final dropout prediction [13].

III. Formulation of Dropout Prediction Problem

A. Sequence Labeling Perspective to Dropout Prediction

A MOOC spans over a period of time usually no more than 10 weeks. During the period, lectures are released to students in the form of online videos while assignments and in-class discussions are conducted online. All student activities in the course are recorded by the MOOC platform as the course progresses. For the dataset we use, the course materials, including lecture videos and quizzes, are organized on a weekly basis so they are spread out quite evenly over the course period. Therefore, we regard one week as a time unit and aggregate all student activities within a week for each student. According to the dropout definition (to be detailed later) adopted, a dropout label indicating whether the student drops the course is generated for each student at each time step. Suppose a course spans over t weeks as depicted in Figure 1. For a specific student, we obtain her week-by-week activities in the course, denoted by an input sequence $(\boldsymbol{x_1}, \ldots, \boldsymbol{x_t})$, and the corresponding dropout labels forming an output sequence $(y_1, \ldots, y_t)$. As the course progresses, our goal is to predict the dropout label of a student week by week by using the weekly activities of this student up to the current week, that is, the model trained and the prediction made at week k only use features up to $\boldsymbol{x_k}$.

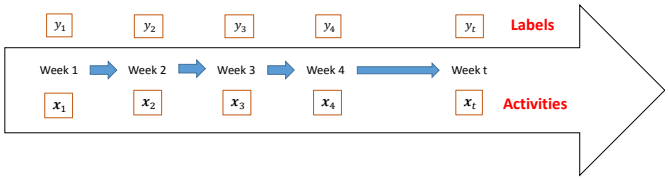

Fig. 1: Formulation of dropout prediction problem

We view the dropout prediction task as a sequence labeling or classification problem [14], [3] since the activities recorded week by week form an input sequence and the corresponding dropout labels assigned according to the chosen dropout definition form an output sequence. While a conventional classification problem assumes that the inputs are independent and identically distributed (i.i.d.), the inputs forming the input sequence in a sequence classification task are actually dependent. For example, if a student participates actively in the early weeks of a course, it is reasonable to assume that she will continue to participate actively in the following weeks. On the contrary, if a student never shows up in the early weeks, it is unlikely to expect her to participate actively or even just show up at all in the following weeks. In this sense, applying sequence labeling techniques to the dropout prediction task enables us to model temporal dependency within the input sequence. Specifically, we will propose several temporal models in Section IV.

B. Dropout Definitions

DEF1	**Participation in the final week:** whether a student will stay to the end of the course [12], [1], [2]
DEF2	**Last week of engagement:** whether the current week is the last week the student has activities [7], [8], [11], [9], [10]
DEF3	**Participation in the next week:** whether a student has activities in the comming week

TABLE I: Three dropout definitions

Time	Week 1	Week 2	Week 3	Week 4	Week 5
Feature	[7,34,9,2,0,7,5]	*Zeros*	[6,3,12,4,1,8,3]	*Zeros*	*Zeros*
DEF1	1	1	1	1	1
DEF2	0	0	1	1	null
DEF3	1	0	1	1	null

TABLE II: An illustrative example for **DEF1**-**DEF3**

We consider three definitions of dropout, as summarized in Table I, to capture different contexts of the student status in a course. They will all be used later when evaluating different dropout prediction models. **DEF1** and **DEF2** are similar in that they both correspond to some *final* status of a student and hence the dropout label cannot be defined until the end of the course. **DEF3**, on the other hand, corresponds to some *ongoing* status of a student and the dropout label can be defined with only one unit (one week in our case) time lag. Based on it, the course instructor could provide relatively timely feedback or intervention. **DEF1** and **DEF3** are easy to understand, yet **DEF2** may need more elaboration. To be precise for **DEF2**, a prediction should be made every week for each student to predict whether the student will have no activities after the current week, i.e., whether she will drop out of the course from the next week onwards. For all three dropout definitions, we use the label 1 to refer to dropout or no participation since our goal is to identify those students who have a high chance of dropping out. Table II gives a concrete example to illustrate the three dropout definitions. The student has some course activities in week 1 and 3 but not in all other weeks (shown as '*Zeros*' for the features). As such, she is considered finally dropping out of the course and thus the labels for **DEF1** are 1's for all weeks. As for **DEF2**, since the student has no activities after week 3, the labels from week 3 onwards are all 1's. On the other hand, based on **DEF3**, the labels are all 1's except week 2 because she has activities in week 3. The 'null' labels are assigned to **DEF2** and **DEF3** in the final week because they cannot be defined for that week. Consequently, the dropout prediction experiments reported in Section V will not include the final week of each course.

C. Datasets for Dropout Prediction

The datasets used in this paper are from two MOOCs, one offered on the Coursera platform[1], called "The Science of Gastronomy", and the other on the edX platform[2], called "Introduction to Java Programming". More details about these two courses are provided separately below.

1) Coursera Course: This is a six-week course with a total of 85394 students enrolled. Among the enrolled students, only 39877 of them have at least one record in terms of the seven features we keep track of as shown in Table III. Since most students have no activities at all or simply dropped out, a simple classifier that labels every single student as a dropout (by definition 1) could achieve a high accuracy, yet we are not particularly interested in this setting. For our dropout prediction task, we will only focus on the subset of 39877 students who are candidates to switch their status from active to dropout.

Feature	Explanation (feature aggregated on a weekly basis)
Lecture view	Number of lecture videos viewed by a student
Lecture download	Number of lecture videos downloaded by a student
Quiz attempt	Number of quizzes attempted by a student
Forum view	Number of times a student views forum contents
Forum thread	Number of forum threads created by a student
Forum post	Number of times a student posts on forum
Forum comment	Number of times a student comments on forum

TABLE III: Feature set of Coursera course

2) edX Course: Compared with the Coursera course described above, this is a longer course spanning over 10 weeks. Similarly, many students simply "disappeared" and have no interactions at all after enrollment, so we still only focus on the subset of 27629 students that have at least one record in terms of the five features in Table IV. The features in Table IV are collected from the clickstream logs of each student, and the five features in Table IV are aggregated by similar clickstream logs according to the edX developer guide. For example, the 'video' feature corresponds to the aggregation of clicking 'play video', 'pause video', 'stop video', 'seek video', 'speed change', 'load video', 'show transcript', and 'hide transcript'. Similar aggregation is also applied to other features.

Feature	Explanation (feature aggregated on a weekly basis)
Navigate	Number of times a student navigates through the course page
Forum	Number of times a student interacts with course forum
Video	Number of course video activities (click-stream) by a student
Problem	Number of course problem activities by a student
Access	Number of activities with other course objects (besides above)

TABLE IV: Feature set of edX course

IV. Temporal Models

In the rest of this section, we present some temporal models, from the sequence labeling perspective, that we will apply to the dropout prediction task. Section IV-A presents a variant of HMM called IOHMM, while Section IV-B and Section IV-C present two RNN models which, for convenience, are referred to as vanilla RNN network and long short-term memory (LSTM) network, respectively.

Compared with static, non-temporal models, temporal models with temporal structures can in principle learn from the entire history of the previous inputs and the current input and hence can capture the temporal dynamics in a more flexible way. That is, the temporal pathway allows a "memory" of the previous inputs to persist in the internal state which then influences the output. Here we consider two temporal models:

- **State space models:** two variants of IOHMM with continuous state space.
- **Recurrent neural networks:** vanilla RNN and RNN with LSTM cells as hidden units.

A. Input-Output Hidden Markov Models

IOHMM was proposed by Bengio and Frasconi [5] for learning problems involving sequentially structured data. While it is originated from HMM, it is more general in that it can learn to map input sequences to output sequences. For both formulations of IOHMM to be presented below, a hidden node in week t is a state vector $\boldsymbol{h_t}$ for that week.

1) First IOHMM Formulation (IOHMM 1): The model formulation of the first IOHMM, referred to as IOHMM 1, is summarized in Equation 1 and Figure 2 below. The temporal dependency is captured by the transitions between hidden states which are shown as connections between the hidden states in Figure 2. For week t, the hidden state $\boldsymbol{h_t}$ is linearly related to the input features $\boldsymbol{x_t}$ and the previous hidden state $\boldsymbol{h_{t-1}}$, subject to some Gaussian noise. The dropout label y_t is linearly related to the current hidden state $\boldsymbol{h_t}$, again subject to some Gaussian noise. The covariance matrices of the zero-mean normal distributions of the noise processes for the state and output are $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. We note that the state transition and observation models defined by the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are linear and time invariant. As such, this IOHMM is a linear time-invariant dynamical system [15]. Moreover, unlike the standard discrete-state HMM, the state space in our IOHMM formulation is continuous so that the state space can in principle bear more representation power compared with enumerating discrete states.

$$\begin{aligned} \boldsymbol{h_t} &= \boldsymbol{A}\boldsymbol{h_{t-1}} + \boldsymbol{B}\boldsymbol{x_t} + \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}) \\ y_t &= \boldsymbol{C}\boldsymbol{h_t} + \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}) \end{aligned} \tag{1}$$

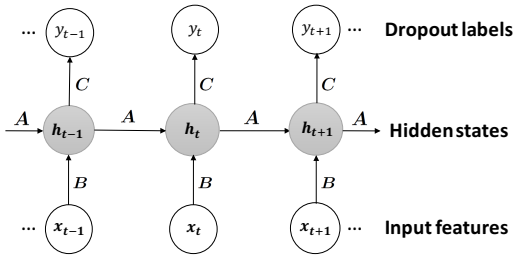

Fig. 2: IOHMM 1

[1] www.coursera.org
[2] www.edx.org

2) Second IOHMM Formulation (IOHMM 2): The model formulation of the second IOHMM, referred to as IOHMM 2, is given in Equation 2. Compared with IOHMM 1, IOHMM 2 has one additional linear dependency which is between the input features and the output labels. This is the same as the original model proposed in [5]. The direct connection from the input features to the output labels can in principle compensate for some potential ineffectiveness of the hidden state learned for the current week.

$$\begin{aligned} \boldsymbol{h_t} &= \boldsymbol{A h_{t-1}} + \boldsymbol{B x_t} + \mathcal{N}(\mathbf{0}, \boldsymbol{Q}) \\ y_t &= \boldsymbol{C h_t} + \boldsymbol{D x_t} + \mathcal{N}(\mathbf{0}, \boldsymbol{R}) \end{aligned} \tag{2}$$

B. Vanilla Recurrent Neural Network (Vanilla RNN)

We now move on to look at RNN models [6] as the second type of temporal models with recurrent structures for dropout prediction. Unlike feedforward neural networks such as the MLP, RNN allows the network connections to form cycles. Despite this relatively small difference, the implications for sequence learning tasks are far-reaching. Indeed, the equivalent result to the universal approximation theory for MLP is that an RNN with a sufficient number of hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy [16]. This makes it a well-fit candidate for our dropout prediction task which is a sequence learning problem. Many variants of RNN have been proposed, including the Elman network [17], Jordan network [18], and vanilla network [19]. In what follows, we first introduce how to apply the vanilla RNN to our dropout prediction task.

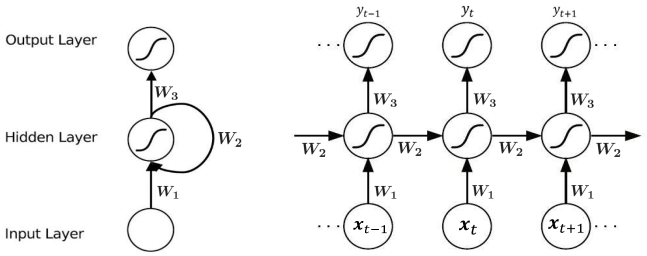

Fig. 3: **Left:** Vanilla RNN structure; **Right:** Vanilla RNN unfolded

The network structure of vanilla RNN is shown in the left part of Figure 3. The recurrent connections from the hidden layer to itself help the network "memorize" the previous inputs in the internal state. It is worth noting that each node in Figure 3 actually represents a layer of network units. As such, each hidden node represents a certain number of hidden units as in the conventional neural network.

1) Network unfolding: A useful way that facilitates visualizing and understanding RNNs is to consider the structure resulted from unfolding the network along the temporal domain. In so doing, the recurrent steps are unfolded to make the resulting network structure acyclic. The right part of Figure 3 shows part of an unfolded vanilla RNN. We can see that the unfolded graph contains no cycles. Consequently, this representation of vanilla RNN can be viewed as a special type of feedforward neural network with shared input weights $\boldsymbol{W_1}$, transitional weights $\boldsymbol{W_2}$, and output weights $\boldsymbol{W_3}$. Moreover, the underlying structure of the unfolded vanilla RNN is actually the same as IOHMM 1 as shown in Figure 2.

The model formulation of vanilla RNN is as follows:

$$\begin{aligned} \boldsymbol{h_t} &= \mathcal{H}(\boldsymbol{W_1 x_t} + \boldsymbol{W_2 h_{t-1}} + b_h) \\ y_t &= \mathcal{F}(\boldsymbol{W_3 h_t} + b_y) \end{aligned} \tag{3}$$

where $\mathcal{H}(\cdot)$ is the hidden layer activation function, $\mathcal{F}(\cdot)$ is the output layer activation function, $\boldsymbol{h_t}$ is the activation of the hidden node for week t, and b_h, b_y are the bias terms.

2) Activation of hidden layer: For the network structure with one hidden layer in our model, each node in the hidden layer computes a weighted sum of the values from the input layer and those from the previous output of the hidden layer. An activation function $\mathcal{H}(\cdot)$ is then applied to the weighted sum to obtain the corresponding activation value. A common choice for $\mathcal{H}(\cdot)$ is the logistic sigmoid function due to its differentiability for training and nonlinearity for stronger representation power.

3) Activation of output layer: The output label of the RNN for each week is computed as the activation of the weighted sum of the activations from the hidden layer. Since dropout prediction is a binary classification problem, we use the logistic sigmoid function for the activation function $\mathcal{F}(\cdot)$ of the output layer. The corresponding cross-entropy loss function is used in defining the objective function for the learning problem. Since the range of the logistic sigmoid function is the open interval $(0, 1)$, the activation of the output unit may be interpreted as the probability that the input vector belongs to the target class (and conversely, one minus the activation gives the probability that it belongs to the other class).

4) Network learning: The optimization problem for RNN learning seeks to minimize the cross-entropy loss with respect to the network weights. This can be done using a variant of the back-propagation learning algorithm called back-propagation through time (BPTT) [20]. Optimization is done through a gradient descent procedure.

5) Vanishing gradient problem: While an important property of RNNs is their ability to use contextual information in learning the mapping between the input and output sequences, a subtlety is that, for basic RNN models, the range of temporality that can be accessed in practice is usually quite limited so that the dynamic states of RNNs are considered as short-term memory. This is because the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the recurrent connections. More precisely, running back-propagation based on gradient descent requires computing the product of a large number of Jacobians if the number of time steps is large and thus the product can easily go to infinity or zero. This effect is often referred to in the literature as the vanishing gradient problem [21].

C. Long Short-Term Memory RNN (LSTM Network)

In this section, we first introduce the basic structure of an LSTM memory cell [22] and then explain how it makes the short-term memory of RNNs last for longer so as to tackle the vanishing gradient problem. After that, we will propose our temporal model for the dropout prediction task based on this

more powerful RNN model with LSTM memory cells as units in the hidden layer.

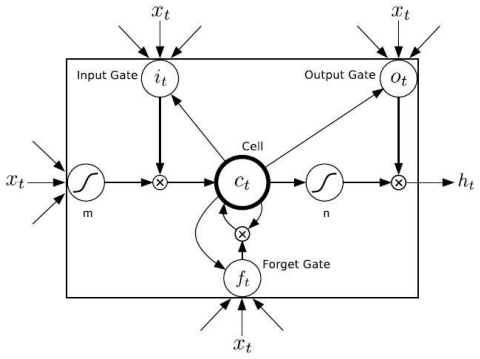

Fig. 4: One LSTM memory cell. This figure is from [23].

1) Long Short-Term Memory Cell: Figure 4 shows a single LSTM memory cell. The state of the LSTM memory cell is stored in c_t. The input and output gates, i_t and o_t, multiply the input and output of the cell while the forget gate f_t multiplies the previous state of the cell. The gate activation functions i_t, o_t, f_t are usually the logistic sigmoid so that the gate activations are between 0 (gate closed) and 1 (gate open). The input and output activation functions, m and n, are usually the hyperbolic tangent function. In effect, the three nonlinear summation gates collect the activations from inside and outside the block and control the activation of the cell via the element-wise multiplication operation $\otimes$. The self-connection normally has a weight of 1 and ensures that, barring any outside interference, the state c_t can remain constant from one time step to another.

The equations below describe the hidden layer activation function $\mathcal{H}(\cdot)$ for an LSTM memory cell at every time step t. In these equations:

- x_t and h_t are the input and output of the memory cell at time t
- $W_{xi}, W_{hi}, W_{xi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xc}, W_{hc}, W_{xo}$, W_{ho}, W_{co} are weight matrices
- b_i, b_f, b_c, b_o are the bias of different gates
- $\otimes$ denotes element-wise multiplication

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\
h_t &= o_t \otimes \tanh(c_t)
\end{aligned} \tag{4}
$$

In effect, the input gate can allow the incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on the other units or prevent it. Finally, the forget gate modulates the self-recurrent connections of the memory cell, allowing the cell to remember or forget its previous state as needed. In essence, the multiplicative gates allow the LSTM memory cells to store and access information over a longer period of time when compared with the units in conventional RNNs, thereby mitigating the vanishing gradient problem. A memory cell contains four equivalent inputs (x_t) and the only output (h_t) from the cell to the rest of the network emanated from the output gate multiplication. Essentially, each memory cell in an LSTM network has the same input and output as a hidden unit in the vanilla RNN, but has more parameters and a system of gating units that control the flow of information.

Over the past decade, LSTM has been applied to various real-world problems, such as protein structure prediction [24], speech recognition [25], [26] and handwriting recognition [27], [28]. As expected, its advantages are most pronounced for problems that require the use of a relatively long range of contextual or temporal information.

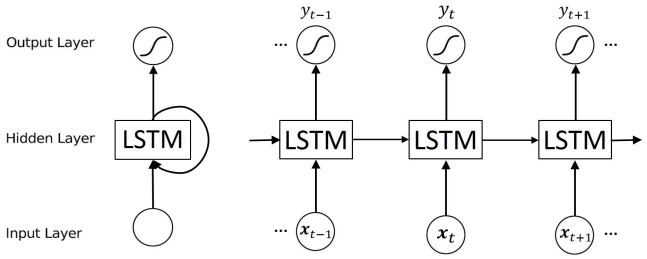

Fig. 5: **Left:** Hybrid of LSTM and RNN (LSTM network); **Right:** LSTM network unfolded

2) Hybrid of LSTM Memory Cells and RNN (LSTM Network): The structure of the proposed LSTM network is shown in Figure 5, with the recurrent view on the left and the unfolded view on the right. Each hidden node of the LSTM network is an LSTM memory block which is composed of a layer of LSTM memory cells. An LSTM network structure is the same as a vanilla RNN in Figure 3, with the hidden node replaced by the LTSM memory block and the standard units within the hidden node replaced by the LSTM memory cells. Suppose each LSTM memory block in the hidden layer has k memory cells. All k memory cells at time t are fully connected to the k memory cells each at time $t-1$ and time $t+1$ through recurrent connections, i.e., hidden layer connections, and will be connected to the corresponding input feature vector and output label. Such connectivity relationship also holds for a vanilla RNN with the memory cells replaced by regular hidden units. Moreover, the update of one memory cell of LSTM is described in Equation 4, while the update function of a regular hidden unit of a vanilla RNN is just a logistic sigmoid as we choose. Similar to vanilla RNN, the activation function of the output layer is logistic sigmoid.

V. Experiments

A. Evaluation Criterion

We choose to use the area under the Receiver Operating Characteristic curve (ROC), called the AUC score, to be the evaluation criteria through our experiments. An ROC curve is a graphic plot created by plotting the true positive rate (TPR) against the false positive rate (FPR). In a dropout prediction task, the true positive rate is the fraction of students predicted as "dropouts" by a classifier who actually dropped this course, and false positive rate is the fraction of students predicted as "dropouts" by a classifier who actually did not drop this course. In order to plot an ROC curve, we need to vary the discrimination or classification threshold to generate the corresponding TPR and FPR, so that the AUC measure is

invariant to the classification threshold. Numerically speaking, an AUC score is a number in the range [0,1], and the closer the number is to 1, the better the classification performance.

B. Baseline Models

Two machine learning models are used as baseline models in our experiments. Since these models are commonly used in different machine learning applications, we will only briefly present them here and focus on the settings for our dropout prediction experiments.

1) Support Vector Machine: SVM was recently applied to the dropout prediction task [7], [8], [11]. We use two variants of SVM, linear SVM and SVM with the radial basis function (RBF) kernel, on the LIBSVM package [29]. The parameter C of the linear SVM and the parameters C, γ of the SVM with RBF kernel are tuned via 5-fold cross-validation on the entire training set. Every week we train a separate SVM classifier from scratch using the observed data up to the current week. This ensures that data only available after the current week will not be used for training the classifier and making prediction.

2) Logistic Regression: Logistic regression has also been applied to the dropout prediction task [10], [2]. This simple classifier models the posterior probability of observing the target class for a single trial as a logistic function of the weighted sum of the input features:

$$P(y \mid \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{W}\boldsymbol{x}}} \tag{5}$$

where y is the class label, $\boldsymbol{x}$ is the input feature vector, and $\boldsymbol{W}$ is the regression weight vector. Since the range of the logistic function is $(0, 1)$, its output can naturally be interpreted as a probability. By regarding logistic regression as a special type of generalized linear model, we use a standard MATLAB toolbox to implement it. The model is trained iteratively using a gradient-descent procedure. Similar to SVM, a separate model is trained from scratch every week.

For the baseline models, we consider two different ways of representing the features:

- Stacked features: The feature vector used in week t is formed by stacking the features of all previous weeks on top of those of the current week, which captures the cumulative activity-related information of a student since the beginning of the course.
- Non-stacked features: The second way is simply to use the features of the current week as input to the model. It does not capture any information from the previous weeks. Unlike feature stacking which results in an increasing feature dimensionality as the course progresses, it is fixed for this scheme.

C. Experimental Settings of Temporal Models

In this section, we describe the experimental settings of our temporal models which include the two variants of IOHMM, vanilla RNN and LSTM network. The results reported below are all average AUC scores based on 5-fold cross-validation. We emphasize that all our models are trained in an incremental manner, that is, for the models trained and predictions made in week t, only features up to week t are used without using any future information. To ensure this, a separate model is trained for each week using only features up to the current week, with its parameters initialized by using those of the trained model from the previous week.

For both formulations of IOHMM, we train the models using an expectation-maximization (EM) algorithm similar to that for linear dynamical systems as described in [30]. The major hyperparameter to determine for these models is the dimensionality of the state space. From our investigation, a dimensionality higher than three hurts the predictive performance. The results reported below are based on IOHMM models of a continuous one-dimensional hidden state space which can achieve the best performance for both datasets.

As for the vanilla RNN and LSTM network, we use the RECURRENNT toolbox [31] which is a C++ implementation of RNN models. It supports the use of graphics processing units (GPUs) based on NVIDIA's CUDA programming model. It can be used to construct RNN models that use either standard hidden units in feedforward networks or LSTM memory cells. Training is done using stochastic gradient descent based on BPTT. For a one-layer architecture of either the vanilla RNN or the LSTM network, two major types of parameters need to be determined:

- The number of units in a hidden node of the vanilla RNN or the number of memory cells within an LSTM block.
- Initialization of the bias terms and network weights in the hidden and output layers.

From our experimental investigation, significantly increasing the number of units in a hidden node of the vanilla RNN or the number of cells in each LSTM block does not bring about much improvement in the predictive performance. In our experiments, we fix the number to 20 which is reasonably large considering that the input feature dimensionality is only seven for the Coursera course and five for the edX course. Moreover, network weights are commonly initialized using the uniform distribution or the Gaussian distribution. By slightly changing the interval of the uniform distribution or the variance of the Gaussian distribution, we found that initialization does not affect much the final results as long as there are sufficient iterations. So we choose to use a simple uniform distribution in the range $[-0.1, 0.1]$ for initializing the network weights of both the vanilla RNN and the LSTM network. Besides, we tune the initialization of the bias terms of the hidden layer and output layer within the range $[-0.2, 0.2]$. We found that having positive bias terms improves the performance slightly for the first dropout definition while having zero bias terms is slightly better for the second and third dropout definitions. So the experiments below regarding both vanilla RNN and LSTM networks will initialize a bias of 0.2 to the hidden and output layers for the first dropout definition, and a bias of 0 for the second and third dropout definitions.

D. Nonlinear Models Help

Figure 6 and 7 show the results of vanilla RNN and two IOHMM formulations. As the results suggest, vanilla RNN outperforms the IOHMMs for all cases. While both vanilla RNN and IOHMM have the same temporal structures, the crucial difference is that both the latent representation and the classifier of RNN are nonlinear, but they are both linear in our IOHMM formulations. This demonstrates again that

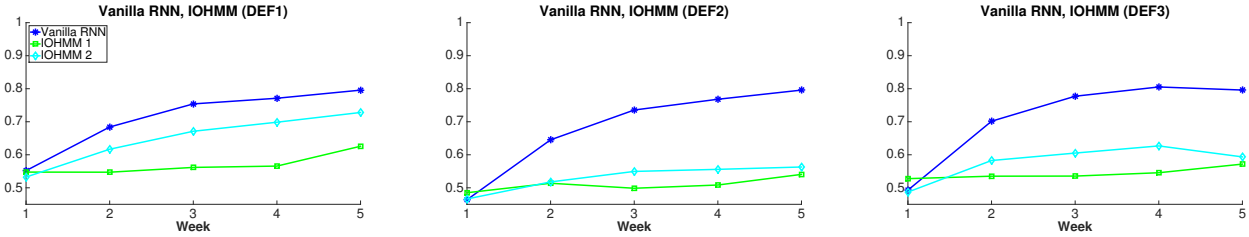

Fig. 6: AUC scores of vanilla RNN and IOHMMs for Coursera course

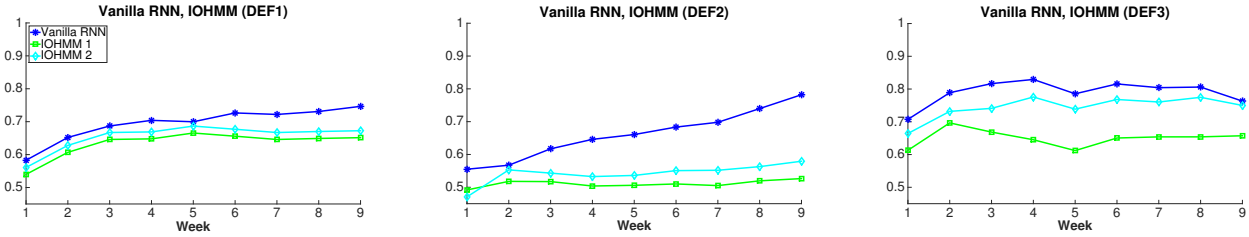

Fig. 7: AUC scores of vanilla RNN and IOHMMs for edX course

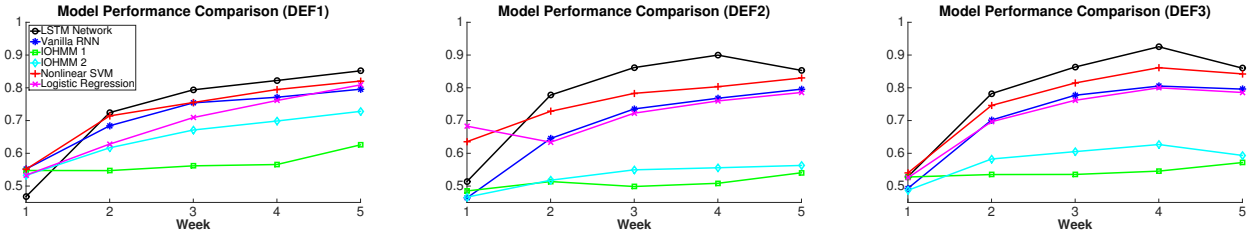

Fig. 8: AUC scores of all models for Coursera course

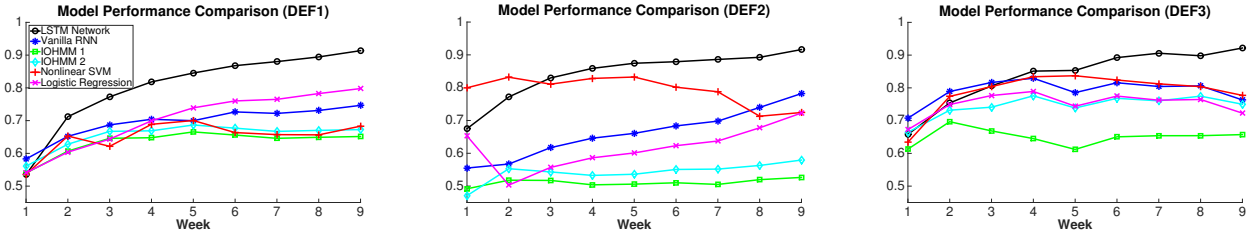

Fig. 9: AUC scores of all models for edX course

nonlinearity in the model can help the dropout prediction task. Moreover, the fact that IOHMM 1 is worse than IOHMM 2 shows that the hidden state learned by IOHMM 1 through linear state transition and the observation matrix are not powerful enough to predict the dropout label well. For baseline models, we compare nonlinear SVM using RBF kernel with linear SVM on both datasets for all dropout definitions, and we also found that nonlinearity does help to improve the performance for SVM. The result comparisons between nonlinear SVM and linear SVM are not included here due to page limitation.

E. Model Performance Comparison

Figure 8 and 9 compare all the temporal models described above and the two baseline models on both the Coursera and edX courses. For the baseline models, we report the settings that give the best performance, namely, stacked features for the first dropout definition and non-stacked features for the last two dropout definitions. For SVM, results of the nonlinear SVM are reported since it outperforms the linear SVM.

1) LSTM Network Wins: In terms of the AUC score, Figure 8 and 9 show that the LSTM network outperforms the other models compared except for some dropout definitions during the early weeks when the AUC scores for all models are relatively low. In particular, the LSTM network performs consistently and significantly better than the vanilla RNN, showing that the long-term memory retained by the LSTM block is very crucial and effective for boosting the dropout prediction performance.

2) Other Temporal Models and Baselines: Several findings regarding the other models are worth noting as well. First, as another RNN structure, although the vanilla RNN is inferior to the LSTM network, it is still usually among the top 3 methods. Second, both IOHMM models generally give the worst performance, with IOHMM 2 better than IOHMM 1. Even though they have temporal structures, their linear nature makes them incapable of learning a hidden state that is effective for dropout prediction. At last, the two baseline models are comparable in performance to the vanilla RNN, yet they still fall behind the

LSTM network quite a lot especially during the later weeks of a course. For most cases, nonlinear SVM performs better than logistic regression, which is consistent with our conclusion that nonlinearity can help the dropout prediction task since logistic regression is a linear model. Besides, we also note that the baseline methods do not give consistent performance in the two courses as they generally perform better on the Coursera course than on the edX course in which the performance drops for some cases as the course progresses.

VI. Conclusion

For the dropout prediction task, our research reported here is novel, though quite natural, to view it as a sequence classification problem and apply temporal models to solve it. Our extensive experimental validation shows that nonlinearity for both the latent representation and the classifier, as well as the longer-term memory retained by the LSTM memory cells, play crucial roles in delivering superior performance. The LSTM network consistently and significantly outperforms all other methods for all three dropout definitions.

To take this work further, one possible extension to the model is to add a max-pooling layer before the output layer. A max-pooling layer is essentially a *max-out* operation [32] in which the output of a unit is activated by the *maximum* or *maximum subset* of the inputs. The rationale behind this is that a strong activation in a hidden unit learned from the input features will not be suppressed by other possibly noisy dimensions. We expect this extension to further improve the robustness of our model.

References

[1] A. Ramesh, D. Goldwasser, B. Huang, H. Daume III, and L. Getoor, "Learning latent engagement patterns of students in online courses," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[2] J. He, J. Bailey, B. I. Rubinstein, and R. Zhang, "Identifying at-risk students in massive open online courses," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, vol. 385.

[4] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994, vol. 2.

[5] Y. Bengio and P. Frasconi, "An input output HMM architecture," *Advances in Neural Information Processing Systems*, pp. 427–434, 1995.

[6] L. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, 2001.

[7] B. Amnueypornsakul, S. Bhat, and P. Chinprutthiwong, "Predicting attrition along the way: The UIUC model," *EMNLP 2014*, p. 55, 2014.

[8] M. Kloft, F. Stiehler, Z. Zheng, and N. Pinkwart, "Predicting MOOC dropout over weeks using machine learning methods," *EMNLP 2014*, p. 60, 2014.

[9] M. Sharkey and R. Sanders, "A process for predicting MOOC attrition," *EMNLP 2014*, p. 50, 2014.

[10] C. Taylor, K. Veeramachaneni, and U.-M. O'Reilly, "Likely to stop? predicting stopout in massive open online courses," *arXiv preprint arXiv:1408.3382*, 2014.

[11] T. Sinha, N. Li, P. Jermann, and P. Dillenbourg, "Capturing "attrition intensifying" structural traits from didactic interaction sequences of MOOC learners," *arXiv preprint arXiv:1409.5887*, 2014.

[12] D. Yang, T. Sinha, D. Adamson, and C. P. Rose, "Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses," in *Proceedings of the 2013 NIPS Data-Driven Education Workshop*, vol. 10, 2013, p. 13.

[13] L. M. B. Manhães, S. M. S. da Cruz, and G. Zimbrão, "Evaluating performance and dropouts of undergraduates using educational data mining," in *Proceedings of the Twenty-Ninth Symposium on Applied Computing*, 2014.

[14] N. Nguyen and Y. Guo, "Comparisons of sequence labeling algorithms and extensions," in *Proceedings of the 24th International Conference on Machine Learning*. ACM, 2007, pp. 681–688.

[15] J. C. Willems, "From time series to linear system part i. finite dimensional linear time invariant systems," *Automatica*, vol. 22, no. 5, pp. 561–580, 1986.

[16] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, no. 1, pp. 107–123, 2000.

[17] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[18] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Artificial Neural Networks*. IEEE Press, 1990, pp. 112–127.

[19] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," 2015, book in Preparation for MIT Press. [Online]. Available: http://www.iro.umontreal.ca/~bengioy/dlbook

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, pp. 323–533, 1986.

[21] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with dradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6645–6649.

[24] S. Hochreiter, M. Heusel, and K. Obermayer, "Fast model-based protein homology detection without alignment," *Bioinformatics*, vol. 23, no. 14, pp. 1728–1736, 2007.

[25] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4. IEEE, 2005, pp. 2047–2052.

[26] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 369–376.

[27] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber, "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks," in *Proceedings of 9th International Conference on Document Analysis and Recognition*, vol. 1, 2007, pp. 367–371.

[28] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández, "Unconstrained on-line handwriting recognition with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2008, pp. 577–584.

[29] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.

[30] Z. Ghahramani, "Parameter estimation for linear dynamical systems," Dept. Comp. Sci., Univ. Toronto, Tech. Rep., 1996.

[31] F. Weninger, J. Bergmann, and B. Schuller, "Introducing CURRENNT– the munich open-source CUDA recurrent neural network toolkit," *Journal of Machine Learning Research*, vol. 15, 2014.

[32] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.