Deep Model for Dropout Prediction in MOOCs

Wei Wang
LILY, Interdisciplinary Graduate School, Nanyang Technological University
Singapore
wwang008@e.ntu.edu.sg

Han Yu
Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University
Singapore
han.yu@ntu.edu.sg

Chunyan Miao
School of Computer Science and Engineering, Nanyang Technological University
Singapore
ascymiao@ntu.edu.sg

ABSTRACT

Dropout prediction research in MOOCs aims to predict whether students will drop out from the courses instead of completing them. Due to the high dropout rates in current MOOCs, this problem is of great importance. Current methods rely on features extracted by feature engineering, in which all features are extracted manually. This process is costly, time consuming, and not extensible to new datasets from different platforms or different courses with different characters. In this paper, we propose a model that can automatically extract features from the raw MOOC data. Our model is a deep neural network, which is a combination of Convolutional Neural Networks and Recurrent Neural Networks. Through extensive experiments on a public dataset, we show that the proposed model can achieve results comparable to feature engineering based methods.

CCS CONCEPTS

• **Computing methodologies** → *Supervised learning by classification*; *Neural networks*; • **Applied computing** → *E-learning*;

KEYWORDS

Deep Learning, MOOCs, Dropout Prediction

ACM Reference Format:
Wei Wang, Han Yu, and Chunyan Miao. 2017. Deep Model for Dropout Prediction in MOOCs. In *Proceedings of ICCSE'17, Beijing, China, July 6–9, 2017,* 7 pages.
https://doi.org/10.1145/3126973.3126990

1 INTRODUCTION

Massive Open Online Courses (MOOCs) [6, 20] have witnessed fast development in recent years. Through MOOCs, students around the world have the opportunities of taking online courses via the Internet. This breaks limitations of traditional courses in classrooms. In MOOCs, students have more flexibility in terms of when and where to take the courses. In this way, MOOCs have attracted more and more students and becoming increasingly popular.

ICCSE'17, July 6–9, 2017, Beijing, China

ACM ISBN 978-1-4503-5375-5/17/07…$15.00
https://doi.org/10.1145/3126973.3126990

Despite their rapid development and successes, there are still some problems within MOOCs. One prominent problem is the high dropout rates. Most of the students taking online courses do not complete the courses and drop out halfway. This could be a potential factor hindering the development of MOOCs. Making effective prediction of whether a student will drop out is of great value for MOOC platforms.

In the dropout prediction problem, the data we have are raw activity records of students in the online course platform over a period of time. The prediction we need to make is whether these students will drop out from the courses in the future. This is a binary classification problem.

To solve this problem, some methods [1, 16, 21, 25, 27] have been proposed in recent years. Most of these methods follow the standard way of solving classification problems. In the first step, features are extracted from the raw activity records. In the second step, classification is accomplished via classification algorithms.

Although inspiring results have been achieved by these methods, there still exist some problems in them. One salient problem is the way of extracting features in the first step. In these methods, the step of feature extraction is usually accomplished via feature engineering [19, 29]. Feature engineering is a process of manually extracting features from the raw data. This process is carried out in a heuristic manner and all features are extracted by hand. In this process, people who extract features need to be familiar with the dataset and have some corresponding domain knowledge. When extracting features by feature engineering, it needs several iterations of feature extraction and testing. This makes this process very time consuming and inconsistent. On the other hand, in feature engineering, strategies for extracting features are specific to the characteristics of datasets. Strategies effective for one kind of datasets may not be effective for another kind of datasets. If there are new kinds of datasets, new strategies of extracting features must be developed manually.

These drawbacks of feature engineering are especially serious in dropout prediction in MOOCs. In this problem, activity records coming from different platforms and different courses often have different characteristics in both format and content. Effective strategies of extracting features in one dataset may not be effective in another dataset. In practice, we often meet new datasets coming from new platforms or new courses. In this case, we need to develop new strategies of extracting features.

Because of the drawbacks of feature engineering, approaches that can automatically extract features are required. One promising character of deep learning models is they can extract features automatically from the raw input [3]. As a type of deep learning

model, Convolutional Neural Networks (CNN) [17] are widely used in feature extracting in lots of areas.

The MOOC activity records are time series data. When making predictions about dropout, this characteristic should also be considered. To make use of this characteristic, another type of deep learning model, Recurrent Neural Networks (RNN) is promising.

In this paper, we propose a deep learning model that combines CNN and RNN in a bottom-up manner. In our model, features are extracted automatically from raw records by convolutional and pooling layers in the lower part of the model. Characteristics of time series data are considered by recurrent layer in the upper part of the model. As our model can be seen as a combination of CNN and RNN, we call our model ConRec Network. The advantage of our model is that, features are automatically extracted. In this way, no feature engineering is needed. This saves time and human efforts in model training. To evaluate the effectiveness of the proposed model, we conduct experiments on a public dataset. Experimental results show that our model can achieve comparable results to those obtained by feature engineering based methods.

2 RELATED WORK

In this section, we give a brief summary of related work in areas of MOOCs dropout prediction and deep learning.

2.1 Dropout Prediction in MOOCs

A general definition of "dropout" in MOOCs is not having activity records in a period of time. In current literature, according to the differences in datasets and prediction purposes adopted by different papers, the specific definition varies in different papers.

Also in these works, different classification algorithms and different approaches of extracting features were adopted. In [16], features were extracted from click-stream data, and a linear SVM was used to predict dropout in each week. In [1], features were also extracted from click-stream data, and a SVM with RBF kernel was used to predict dropout. In [27], logistic regression was used to predict dropout by the last activity on the corresponding course of a student. Sinha et al. [25] only used video click-stream interaction data to extract features. Mi and Yeung [21] regarded dropout prediction as a sequential prediction problem, and used RNN for prediction under different definitions of dropout. Li et al. [18] converted dropout prediction into a semi-supervised learning problem and used multi-training to solve the problem.

In MOOCs, a similar problem to dropout prediction is completion prediction. In this problem, instead of predicting whether a student will drop out from a course, it aims to predict whether a student can complete a course and get the corresponding certificate. He et al. [13] used logistic regression to identify students who seem to be not able to complete the course. Qiu et al. [24] used latent dynamic factor graph model for the prediction.

There are also some works that comprehensively considered activity records and completion of the course, and proposed the prediction problem suitable for their problem setting [23, 31].

2.2 Deep Learning

Deep learning [2, 8, 11] is a subfield of machine learning. It has been successfully applied in a lot of areas in recent years. In our model, we mainly utilize two types of deep learning models, CNN and RNN.

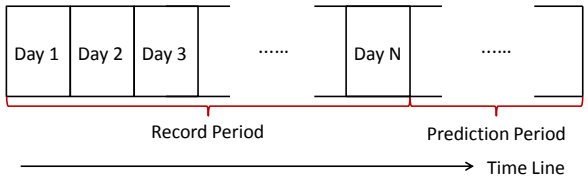

Figure 1: The Dropout Prediction Problem

CNNs are a kind of neural networks used to deal with data which have grid-like topology [11]. They have been successfully used in many areas, such as image object recognition [17], video classification [15], natural language processing [4], speech recognition [7] and human activity recognition [32]. In CNNs, each convolutional layer only finds relationships among adjacent elements. As a CNN is of a layer-wise architecture, lower layers can find some local patterns, and upper layers can find patterns in the whole scale of the input. CNN is often used as a tool for extracting features from raw data.

RNNs are a kind of neural networks used to process sequential data. They have been successfully used in many areas, such as speech recognition [12], question answering [30] and machine translation [26]. In RNN, the input is a sequence, such as time series, biological sequence, speech or language sequence. RNN process the elements in the input sequentially to acquires the information contained in the whole sequence.

By combining CNN and RNN into a new architecture, we propose a framework to address the MOOC dropout prediction problem.

3 PROPOSED MODEL

3.1 Problem Formulation

In an online course platform, there are many registered students and many online courses. Each student can take several different courses, and each course can be taken by many different students. In the platform, the courses usually last for several weeks. Students taking these courses have some activity records in the platform. In these records, each record is a time-stamped log, recording information of an event in the platform.

In this paper, we formulate the problem of dropout prediction as: given activity records of some students on some courses in a period of time (as shown in Figure 1, it is from Day 1 to Day N, we call it "record period"), we aim to predict whether these students will drop out the courses in a period in the future (as shown in Figure 1, we call it "prediction period"). In the prediction period, if a student has activity records on a course, we consider this student has not dropped out from the corresponding course; otherwise, we consider this student has dropped out from this course.

3.2 Preprocessing of Input

3.2.1 Characteristics of the Raw Records. The dataset we have in this problem are raw activity records. These records are structured

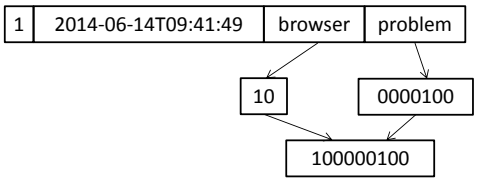

Figure 2: An Example of Converting a Raw Record to a One-hot Vector

time-stamped logs ordered chronologically and consist of different attributes. These attributes have different physical meanings and contain information on different aspects. For example, in one dataset, the records may contain attributes like student ID (denoting the student which each record belongs to), time (denoting the time when each event occurs), activity type (denoting activity type of the event) and so forth. Each record in the dataset records information of an event. It contains specific values on each of the attributes.

The time span of record period in one dataset usually lasts for several weeks. In this paper, for the convenience of the description of our model, we divide the time span of record period by several different scales. For each record, it records an event in the platform at a certain time point. We refer to the time in each record as a time point. Several adjacent time points form a time unit. Several adjacent time units form a time slice. All time slices form the record period.

3.2.2 Converting to One-hot Vectors. The raw activity records in the dataset are in their original text format, can not be directly used as inputs to our model. To use these data, we need to convert them into the format that can be processed by deep neural networks. In this paper, we convert them into vectors.

In these records, each attribute can take several different values. For each record, we convert the specific value of each attribute to be a one-hot vector. The number of digits of each one-hot vector equals the number of different values this attribute can take. This method of converting has been used by [14] in the area of natural language processing. In their work, they converted each word into a one-hot vector and concatenated vectors belonging to the same sentence into a long vector to represent this sentence. In our model, as the example[1] shown in Figure 2, for one record, we convert some of the attributes into one-hot vectors, and concatenate these vectors into a long vector to represent this record.

Each record in the dataset has a time attribute denoting the time the event occurred. The occurring time of each record corresponds to a time point in record period. In the dataset, not every time point in record period has a corresponding record. There exists a large number of time points that do not have records. For these time points, we pad zero vectors at the corresponding positions.

3.2.3 Combining Vectors. In activity records, the scale of each time point can be accurate to second, but the time span of record period can last for several weeks. Compared with the time scale of each time point, the time span of record period is quite large. If we directly use the original vectors at every time point as input, the volume of input will be very large. For this reason, it is necessary to combine some vectors and reduce the volume of the input. In our model, we combine all vectors belonging to the same time unit into one vector. To combine these vectors, we add up all these vectors in a bitwise manner. For example, if we have two vectors [0, 1, 0, 0, 1, 0, 1] and [0, 0, 1, 0, 1, 0, 0], by adding them, we get [0, 1, 1, 0, 2, 0, 1].

After adding up, for each time unit, we have one vector to represent it. Then, we concatenate all vectors in the same time slice into a matrix, each row to be the vector of one time unit. After that, for each student taking one course, we have several matrices representing records in different time slices. We use them as inputs to the model.

3.3 Overview of the Model

To solve the dropout prediction problem, we need to consider the characteristics of the input. The characteristics can be seen from the macro and the micro views, respectively.

In the macro view, activities of a student in a certain time will be affected by his/her states or activities in the past. The effect of temporally closer ones is greater than temporally further ones. In our model, this characteristic is used by the RNN among different time slices.

In the micro view, for each time slice, we have a matrix to represent records within it. Features can be extracted from each of the matrices. To extract features, we use a CNN, which is the most widely adopted neural network as feature extracting tool.

In our model, we combine CNN and RNN together to make use of the characters stated above. As our model is a combination of these two kinds of neural networks, we call our model ConRec Network. In this model, in the lower part, there are convolutional and pooling layers to extract features in each time slice. In the upper part, there is a recurrent layer to combine information in each time slice. The combined information is used to make predictions.

3.4 Details of ConRec Network

This model contains an input layer, an output layer and six hidden layers. The structure of the model is shown in Figure 3. The first and third hidden layers are convolutional layers, the second and forth hidden layers are pooling layers, the fifth hidden layer is a fully connected layer, the sixth hidden layer is a recurrent layer. The nonlinear function we applied in each hidden layer is rectified linear unit (ReLU) function.

In this model, a student taking a course is treated as an instance, it contains activity records in some time slices. In this model, we assume matrices of each time slice are of the same size, and the numbers of matrices in each instance are also the same. For each instance, it contains T matrices, representing records in T time slices. Thus, for each instance, the input of the model is T matrices $X^{(1)}, X^{(2)} \cdots X^{(t)} \cdots X^{(T)}$, each of size $Q^{(0)} \times H^{(0)}$, where $Q^{(0)}$ is the number of time units in each time slice, $H^{(0)}$ is the length of the vectors at each time unit. As illustrated above, in the lower layers, we extract features among adjacent elements.

3.4.1 Convolutional Layers. In this model, the first and the third hidden layers are convolutional layers. In these two layers, the convolutional mode is valid convolutional, i.e. it does not pad zero

[1]This example comes from the data provided by KDD Cup 2015.

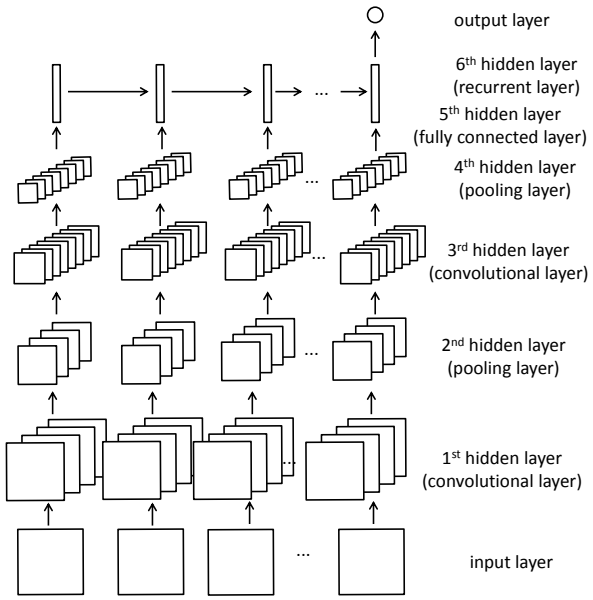

Figure 3: The Proposed ConRec Network

vectors at the boundary of the matrix when performing convolution operation. We set the stride to be 1. For each instance, the input is $T \times K^{(p-1)}$ matrices, each of size $Q^{(p-1)} \times H^{(p-1)}$. The output is $T \times K^{(p)}$ matrices, each of size $Q^{(p)} \times H^{(p)}$,where $Q^{(p)} = Q^{(p-1)} - L^{(p)} + 1$ and $H^{(p)} = H^{(p-1)} - D^{(p)} + 1$. Operations in these two layers are:

$$\begin{aligned} X^{(p),(k^{(p)}),(t)} =& ReLU(\sum_{k^{(p-1)}=1}^{K^{(p-1)}} X^{(p-1),(k^{(p-1)}),(t)} \\ & * W^{(p),(k^{(p)}),(k^{(p-1)})} + b^{(p),(k^{(p)})}), \\ & \forall t = 1, 2 \cdots T, \quad \forall k^{(p)} = 1, 2 \cdots K^{(p)} \end{aligned} \tag{1}$$

where T is the number of time slices, p is the index of the layer. $X^{(p-1),(k^{(p-1)}),(t)}$ is the matrix representing the $(k^{(p-1)})^{th}$ feature map of $(p-1)^{th}$ layer at t^{th} time slice, it is a input of the p^{th} layer. $W^{(p),(k^{(p)}),(k^{(p-1)})}$ is a filter in the p^{th} layer, it is a matrix that generates the $(k^{(p)})^{th}$ feature map in the p^{th} layer from the $(k^{(p-1)})^{th}$ feature map from the $(p-1)^{th}$ layer, it is of size $L^{(p)} \times D^{(p)}$. $b^{(p),(k^{(p)})}$ is the bias of the generated $(k^{(p)})^{th}$ feature map in the p^{th} layer, it is added to all elements of the corresponding feature map matrices. Operator "$*$" represents convolution operation.

3.4.2 Pooling Layers. The second and the forth hidden layers are pooling layers. In this model, we use max pooling. For each instance, the input is $T \times K^{(p-1)}$ matrices, each of size $Q^{(p-1)} \times H^{(p-1)}$. The output is $T \times K^{(p)}$ matrices, each of size $Q^{(p)} \times H^{(p)}$, where $K^{(p)} = K^{(p-1)}, Q^{(p)} = \frac{Q^{(p-1)}}{L^{(p)}}, H^{(p)} = \frac{H^{(p-1)}}{D^{(p)}}$. Operations in these two layers are:

$$\begin{aligned} & X_{i,j}^{(p),(k^{(p)}),(t)} = \\ & \max_{1 \le l^{(p)} \le L^{(p)}, 1 \le d^{(p)} \le D^{(p)}} (X_{(i-1)\times L^{(p)}+l^{(p)},(j-1)\times D^{(p)}+d^{(p)}}^{(p-1),(k^{(p-1)}),(t)}), \\ & \forall t = 1, 2 \cdots T, \quad \forall k^{(p)} = k^{(p-1)} = 1, 2 \cdots K^{(p-1)} \end{aligned} \tag{2}$$

in these two layers, pooling areas do not overlap, they are of size $L^{(p)} \times D^{(p)}$.

3.4.3 Fully Connected Layer. The fifth hidden layer is fully connected layer. This layer combines features extracted by the convolutional and pooling layers, and generate one vector for each time slice. For each instance, the inputs are $T \times K^{(4)}$ matrices, each of size $Q^{(4)} \times H^{(4)}$. The outputs are T vectors, each of size M. In this layer, for each matrix in the input, we firstly flatten it into a vector, each vector is represented as $\mathbf{x}^{(4),(k^{(4)}),(t)}$. Operation in this layer is:

$$\begin{aligned} \mathbf{x}^{(5),(t)} = ReLU(\sum_{k^{(4)}=1}^{K^{(4)}} W^{(4),(5),(k^{(4)})} \mathbf{x}^{(4),(k^{(4)}),(t)} + \mathbf{b}^{(5),(t)}), \\ \forall t = 1, 2 \cdots T \end{aligned} \tag{3}$$

where $W^{(4),(5),(k^{(4)}),(m)}$ is a weight matrix generating the vector from the $(k^{(4)})^{th}$ feature map of the forth layer. $\mathbf{b}^{(5),(t)}$ is the bias.

3.4.4 Recurrent Layer. In the first five layers, we extract features in each time slice. In the sixth hidden layer, we combine information in each time slice. This layer is a recurrent layer. Operation in this layer is:

$$\begin{aligned} \mathbf{s}^{(t)} = ReLU(W^{(6)} \mathbf{x}^{(5),(t)} + U^{(6)} \mathbf{s}^{(t-1)} + \mathbf{b}^{(6)}), \\ \forall t = 1, 2 \cdots T \end{aligned} \tag{4}$$

where $\mathbf{s}^{(t)}$ is the hidden state at time t, $W^{(6)}$ and $U^{(6)}$ are weight matrices, $\mathbf{b}^{(6)}$ is the bias.

This prediction problem is a binary classification problem. We denote class "not drop out" as "0" and class "drop out" as "1". The output of this model is

$$\hat{y} = \frac{1}{1 + e^{-(V\mathbf{s}^{(T)} + \mathbf{c})}} \tag{5}$$

where $\mathbf{s}^{(T)}$ is the hidden state in time slice T, it is a combination of all information in previous time slices. V is the weight matrix and $\mathbf{c}$ is the bias. The output $\hat{y}$ is a scaler between 0 and 1 denoting the probability the output belongs to class "1".

The loss function of this model is the mean of negative log-likelihood of training instances

$$L = \frac{1}{N} \sum_{i=1}^{N} (-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)) \tag{6}$$

where N is the number of training instances.

4 EXPERIMENTAL EVALUATION

In this section, we first introduce the dataset used in the experiments, then describe the settings of the experiments. Finally, we report the results of the experiments and discuss the implications.

Table 1: Attributes in Files Containing Information about Enrollments

Attribute	Meaning
Enrollment ID	ID of the enrollment record
User Name	Name of the student
Course ID	ID of the course

Table 2: Attributes in Files Containing Activity Records of the Enrollments

Attribute	Meaning
Enrollment ID	ID of the enrollment record
Time	Time when the event occurs
Source	Event source ("server" or "browser")
Event	Event type (there are 7 types: "problem" "video" "access" "wiki" "discussion" "navigate" and "close page")
Object	The object the student access or navigate to.(Only for "navigate" and "access" event).

4.1 Dataset

The dataset we used for the experiments comes from KDD Cup 2015[2]. It contains information about 39 courses in the online course platform XuetangX. For each course, the record period is 30 days. Information contained in the dataset includes information about which student enrolls in which course, activity records of the students, and other information about the courses. Table 1 is the attributes in files ("enrollment_train.csv" and "enrollment_test.csv") containing information about enrollment records[3]. Table 2 is the attributes in files ("log_train.csv" and "log_test.csv") containing activity records of the students.

In this dataset, only the enrollment records in the file "enrollment_train.csv" have ground truth (in file "truth_train.csv") of whether a student drops out (enrollment records in file "enrollment_test.csv" have no corresponding ground truth). Thus in this experiment, we only use data of enrollment records from "enrollment_train.csv". This file contains 120,542 enrollment records. In the corresponding activity record file, there are 8,157,277 records. In these enrollment records, 95,581 of them dropped out, the other 24,961 enrollment records did not drop out.

In this experiment, we follow the standard way of splitting training and testing datasets, with proportion of training and testing instances to be 4:1. In this way, we randomly separate these enrollments into 96,434 as training instances and 24,108 as testing instances. In the training and testing sets, the proportions of dropout and not dropout enrollment records are nearly the same.

4.2 Experimental Settings

4.2.1 Prediction Goal. In KDD CUP 2015, the goal is to predict whether the students will drop the courses in the following 10 days. In our experiment, we follow this definition of dropout and the same prediction goal.

4.2.2 Comparison Methods. In the experiments, we aim to evaluate the effectiveness of our model compared with feature engineering based methods. Thus, in the experiments, we select several classification methods as baseline methods and provide them features extracted via feature engineering.

In the experiments, we extract 186 features from the raw records via feature engineering. These features mainly come from five aspects: (1) number of different kinds of events, (2) number of the days that have different kinds of events, (3) amount of operating time, (4) time of the last event of an enrollment item, and (5) number of registered students of the course in an enrollment item. All of these features are normalized by min-max scaling with the maximum and minimum values of each feature in training dataset.

The baseline methods are Linear SVM [16], SVM with RBF kernel [1], Logistic Regression [27], Decision Tree [22], AdaBoost [9], Gradient Tree Boosting [10], Random Forest [5] and Gaussian Naive Bayes [33]. For Linear SVM, Decision Tree and Random Forest, we perform probability calibration to obtain the probabilities of belonging to class "dropout".

4.2.3 Implementation Details. In our model, we select three attributes (attributes "source", "event" in activity records and "course ID" in enrollment information records) from the dataset as input. In this implementation, the scale of time point is second. We set the time unit to be one hour, and the time slice to be one day. For each time point, we convert values in these three attributes into vectors and concatenate them to get a vector of size 48. Then, we add vectors in the same hour, and combine vectors in the same day into one matrix. The input of each enrollment item is 30 matrices, each of size 24×48. In the first hidden layer, we use 20 filters and biases to generate 20 feature maps. The filters are all of size 5×5. The output of this layer is 30×20 matrices of size 20×44. In the second hidden layer, the pooling size is 2×2. The output is 30×20 matrices of size 10×22. In the third hidden layer, the filters are also of size 5×5, and the number of filters and biases is 50. The output is 30×50 matrices of size 6×18. In the forth hidden layer, the pooling size is also 2×2, the output is 30×50 matrices of size 3×9. The fifth hidden layer is the fully connected layer. It generates a vector for each day. The output is 30 vectors of size 20. The sixth hidden layer is the recurrent layer. The input is 30 vectors of size 20. The hidden state vector in this layer is of size 50. The final output is a number between 0 and 1, indicating the probability that this student will drop out the corresponding course. In this model, we denote "0" as "not dropout" and "1" as "dropout".

We implemented this model in Theano [28], which is a python library used to realize deep learning methods. In the experiment, we use mini-batch stochastic gradient descent method to train the parameters. We set the learning rate to be 0.1, and the batch size to be 20. The model is run for 15 epochs.

4.2.4 Evaluation Metrics. The problem solved in this paper is a classification problem. In this problem, number of instances belonging to the two classes differ a lot. This is a phenomenon called class imbalance. In this case, accuracy is not an appropriate evaluation metric. In this paper, we use Precision, Recall, F1-score [18] and

[2]http://kddcup2015.com

[3]In this dataset, they refer to a student taking a course as an enrollment record. It corresponds to an instance in our model and the baseline methods.

Table 3: Performances of Different Methods under Different Metrics (in %)

Method	Precision	Recall	F1-score	AUC
Linear SVM [16]	88.91	96.34	92.48	87.83
SVM(with RBF kernel) [1]	88.06	97.01	92.32	84.51
Logistic Regression [27]	89.17	96.12	92.52	87.70
Decision Tree [22]	84.57	97.87	90.74	80.03
AdaBoost [9]	89.39	95.59	92.39	87.96
Gradient Tree Boosting [10]	89.61	95.71	92.56	88.26
Random Forest [5]	88.86	96.21	92.39	86.71
Gaussian Naive Bayes [33]	89.89	92.52	91.18	77.40
ConRec Network	88.62	96.55	92.41	87.42

Table 4: Relative Difference between ConRec Network and Various Baseline Methods under Different Metrics (in %)

Method	Precision	Recall	F1-score	AUC
Linear SVM [16]	-0.32	0.21	-0.07	-0.46
SVM(with RBF kernel) [1]	0.63	-0.47	0.09	3.44
Logistic Regression [27]	-0.61	0.44	-0.11	-0.31
Decision Tree [22]	4.78	-1.34	1.84	9.23
AdaBoost [9]	-0.86	1.00	0.02	-0.61
Gradient Tree Boosting [10]	-1.10	0.87	-0.16	-0.95
Random Forest [5]	-0.27	0.35	0.02	0.81
Gaussian Naive Bayes [33]	-1.41	4.35	1.34	12.94

Area Under the Receiver Operating Characteristic Curve (AUC) score [21] as evaluation metrics.

4.3 Experimental Results and Analysis

Experimental results of our model and baseline methods are shown in Table 3. From the table we can see, compared with baseline classification methods (they all use features extracted via feature engineering), our model can obtain comparable results on all of the four metrics. This illustrates the effectiveness of our model.

To evaluate the differences between these results more accurately, we calculate relative differences between the performances of ConRec Network and each of the baseline methods on the four metrics. Here, we define "relative difference" as: for a performance value a from ConRec Network and a performance value b from a baseline method, the relative difference between them is $(a - b)/b$. This criteria takes the performance of baseline method as standard, and measures the significance of difference between the performances of ConRec Network and this method.

The values of relative differences are shown in Table 4. In this table, the value in each cell denotes the relative difference between ConRec Network and the corresponding baseline method under the corresponding metric. Positive values mean ConRec Network outperforms the corresponding baseline method; negative values represent the opposite meaning. The absolute value indicates the significance of difference. From the table, we can see, our proposed ConRec Network can achieve comparable results with most of the baseline methods under the four metrics. The performance of ConRec Network is much better than Decision Tree and Gaussian Naive Bayes.

From the results we can conclude: (1) for classification problems based on raw activity records, there exist efficient way of automatically extracting features from the raw data. (2) the approach adopted by our method to extract features is effective. (3) our proposed model ConRec Network is an efficient model to solve dropout prediction problem in MOOCs.

5 CONCLUSION

In this paper, we propose a deep neural network for solving dropout prediction problem in MOOCs. The key advantage of our model is that features used in the model are automatically extracted from the raw records. No manual feature engineering is needed. In this way, this method saves a lot of time and human efforts, and eliminates the potential inconsistency introduced by the manual process. Experimental results on a large scale public dataset show that the proposed model can achieve comparable performance to approaches relying on feature engineering performed by experts.

In future work, we plan to evaluate the effectiveness of this model on other kinds of activity records or on other classification problems based on activity records. Also for the upper layers of the model, we plan to evaluate other types of RNN, such as LSTM and GRU. For the format of input, instead of one-hot vectors, we plan to explore other formats.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative; the Interdisciplinary Graduate School (IGS), NTU; the Lee Kuan Yew Post-Doctoral Fellowship Grant; and the NTU-PKU Joint Research Institute, a collaboration between Nanyang Technological University and Peking University that is sponsored by a donation from the Ng Teng Fong Charitable Foundation. We would like to gratefully acknowledge the organizers of KDD Cup 2015 as well as XuetangX for making the datasets available.

REFERENCES

[1] Bussaba Amnueypornsakul, Suma Bhat, and Phakpoom Chinprutthiwong. 2014. Predicting attrition along the way: The UIUC model. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. 55–59.
[2] Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.
[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[4] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.
[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[6] John Daniel. 2012. Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. *Journal of interactive Media in education* 2012, 3 (2012).

[7] Li Deng, Ossama Abdel-Hamid, and Dong Yu. 2013. A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6669–6673.

[8] Li Deng, Dong Yu, et al. 2014. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing* 7, 3–4 (2014), 197–387.

[9] Yoav Freund and Robert E Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. System Sci.* 55, 1 (1997), 119–139.

[10] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),*. IEEE, 6645–6649.

[13] Jiazhen He, James Bailey, Benjamin IP Rubinstein, and Rui Zhang. 2015. Identifying At-Risk Students in Massive Open Online Courses. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 1749–1755.

[14] Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 103–112.

[15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1725–1732.

[16] 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*. 60–65.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[18] Wentao Li, Min Gao, Hua Li, Qingyu Xiong, Junhao Wen, and Zhongfu Wu. 2016. Dropout prediction in MOOCs using behavior features and multi-view semi-supervised learning. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 3130–3137.

[19] Chen Lin, Helena Canhao, Timothy Miller, Dmitriy Dligach, Robert M Plenge, Elizabeth W Karlson, and Guergana K Savova. 2012. Feature engineering and selection for rheumatoid arthritis disease activity classification using electronic medical records. In *ICML Workshop on Machine Learning for Clinical Data Analysis*.

[20] Tharindu Rekha Liyanagunawardena, Andrew Alexandar Adams, and Shirley Ann Williams. 2013. MOOCs: A systematic study of the published literature 2008-2012. *The International Review of Research in Open and Distributed Learning* 14, 3 (2013), 202–227.

[21] Fei Mi and Dit-Yan Yeung. 2015. Temporal models for predicting student dropout in massive open online courses. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 256–263.

[22] Sreerama K Murthy. 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery* 2, 4 (1998), 345–389.

[23] Saurabh Nagrecha, John Z Dillon, and Nitesh V Chawla. 2017. MOOC Dropout Prediction: Lessons Learned from Making Pipelines Interpretable. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 351–359.

[24] Jiezhong Qiu, Jie Tang, Tracy Xiao Liu, Jie Gong, Chenhui Zhang, Qian Zhang, and Yufei Xue. 2016. Modeling and predicting learning behavior in MOOCs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 93–102.

[25] Tanmay Sinha, Patrick Jermann, Nan Li, and Pierre Dillenbourg. 2014. Your click decides your fate: Inferring information processing and attrition behavior from mooc video clickstream interactions. In *2014 Empirical Methods in Natural Language Processing Workshop on Modeling Large Scale Social Interaction in Massively Open Online Courses*.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[27] Colin Taylor, Kalyan Veeramachaneni, and Una-May O'Reilly. 2014. Likely to stop? Predicting stopout in massive open online courses. *arXiv preprint arXiv:1408.3382* (2014).

[28] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688

[29] Kalyan Veeramachaneni, Una-May O'Reilly, and Colin Taylor. 2014. Towards feature engineering at scale for data from massive open online courses. *arXiv preprint arXiv:1407.5238* (2014).

[30] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* (2015).

[31] 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 *International Educational Data Mining Society*.

[32] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 3995–4001.

[33] Harry Zhang. 2004. The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*. 562–567.