



# Student dropout prediction in massive open online courses by convolutional neural networks

Lin Qiu<sup>1,2</sup> · Yanshen Liu<sup>3</sup> · Quan Hu<sup>3</sup> · Yi Liu<sup>3</sup>

Published online: 17 October 2018  
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

## Abstract

Massive open online courses (MOOCs) have given global learners access to quality educational resources, but the persistent high dropout rates problem has a serious impact on their educational effectiveness. Therefore, how to predict the dropout in MOOCs and make advance intervention is a hot topic in the research of MOOCs in recent years. Traditional methods rely on handcrafted features, the workload is heavy, and it is difficult to ensure the final prediction effect. In order to solve this problem, this paper proposes an end-to-end dropout prediction model based on convolutional neural networks to predict the student dropout problem in MOOCs and it integrates feature extraction and classification into a single framework, which transforms the original timestamp data according to different time windows and automatically extracts features to achieve better feature representation. Extensive experiments on a public dataset show that our approach can achieve results comparable to other dropout prediction methods on precision, recall, F1 score, and AUC score.

**Keywords** Convolutional neural networks · Feature extraction · Dropout prediction · Massive open online courses

## 1 Introduction

With the further popularization of the Internet and the improvement in mobile communication technology, the number of Internet users in the world has increased dramatically, and the form of people's learning has changed greatly. From 2012, massive open online courses (MOOCs) as the exten-

sion of E-Learning have been favored by educators all over the world, and have been vigorously promoted. MOOC platforms have sprung up in the world. Coursera,<sup>1</sup> edX<sup>2</sup> and other online platforms have gathered various course resources from all over the world and made great contributions to the promotion of global education.

Although the development of MOOCs is growing and the number of learners is increasing, the effectiveness of MOOCs is indeed questioned in many aspects. In particular, the statistics show extremely low completion rate is one of the main problems that affect the development of MOOCs (Wang 2013). According to Jordan (2014), the average completion rate of the MOOCs is close to 6.5%, which means that the most students quit their studies before the end of the courses. Therefore, how to reduce the dropout rate and improve the final passing rate of the course is a must for the MOOC builders. Analyzing the learners so as to predict whether or not to quit the course can find out the possibility of dropout early, and then intervene in advance, it is more feasible at present.

In dropout prediction in MOOCs, given various data generated by students in the learning activities, such as click-

---

Communicated by V. Loia.

---

✉ Lin Qiu  
linqiu100@hotmail.com  
Yanshen Liu  
yanshenliu@126.com  
Quan Hu  
henry\_hq@163.com  
Yi Liu  
liuyi136@126.com

- <sup>1</sup> National Engineering Research Center for E-Learning, Central China Normal University, Wuhan 430079, Hubei, China
- <sup>2</sup> School of Computer Science, Yangtze University, Jingzhou 434023, Hubei, China
- <sup>3</sup> Educational Informatization Research Center of Hubei, Central China Normal University, Wuhan 430079, Hubei, China

---

<sup>1</sup> <https://www.coursera.org/>.

<sup>2</sup> <https://www.edx.org/>.

stream data (Hung et al. 2017), assignment submission and performance information (Halawa et al. 2014), curriculum forum posts and interactive information (Chaplot et al. 2015), demographic information of students (Onah et al. 2014), its basic task is to predict whether the students will dropout in the future. This is a typical binary classification task. Most of the forecasting studies use clickstream data as the processing object and apply the general process of data mining to analyze it. There are two main categories in the realization method. One is to consider the problem as a general binary classification problem, and use the traditional machine learning methods to predict, such as logistic regression (LR) (Taylor et al. 2014), support vector machine (SVM) (Kloft et al. 2014) and so on. One is based on the time series characteristics of clickstream data. It treats the problem as a time series classification problem, and uses sequence mining methods to predict, for example, hidden Markov model (HMM) (Balakrishnan and Coetzee 2013). As deep learning has achieved better results in various fields than ever before, deep learning has also been applied to dropout prediction in MOOCs, for example, recurrent neural network (RNN) (Fei and Yeung 2015). These methods are all feature-based classification methods because their final performances usually depend on the quality of the features extracted manually.

Convolutional neural network (CNN) has become a research hot spot with great achievements in many fields. One of the biggest advantages of CNN is that it can automatically learn all kinds of invisible features from the raw data, and does not need too much manual feature processing (Glorot et al. 2011). Therefore, it is natural to think that CNN can be introduced into the problem of dropout prediction in MOOCs to reduce the complexity of features extraction and improve the quality of features. However, there is not much research on the application of CNN to solve the problem of dropout prediction in MOOCs at present.

In this paper, we propose an end-to-end dropout prediction model based on convolutional neural networks (DP-CNN), and try to use this model to directly analyze the clickstream data and make the final prediction. The significance of this work can be summarized as the following:

- Unlike other machine learning-based dropout prediction methods available in the literature, the proposed model processes directly on the raw clickstream data without the need for preprocessing or manual feature extraction.
- Traditional machine learning dropout prediction methods use handcrafted features which are needed for domain knowledge, and the effects are unstable. But the proposed CNN-based model applies the best features learned by CNN to maximize the classification effect, which reduces the complexity and improves the stability.
- Finally, this study designs experiments to compare our approach with other related research methods and

demonstrate the effectiveness of our approach, and then extends the research methods of dropout prediction in MOOCs.

The rest of this paper is organized as follows: In the next section, the related work is highlighted. The research methodology is presented in Sect. 3, and the experimental process and results from the study are given in Sect. 4. Section 5 concludes the study.

## 2 Related work

### 2.1 Dropout in MOOCs

As a new way of education, MOOCs have received extensive attention both in the social media and in academic research since 2012. Breslow et al. (2013) systematically analyzed the first MOOC, “Circuits and Electronics” (6.002x) on edX, from the time for students to visit each course module, the students background, ability, the factors that affect the final grade, etc., and proposed that the students learning effect can be improved by log mining. DeBoer et al. (2013) used a dataset of the course 6.002x on edX to analyze the relationships between students background, learning behaviors, and the final achievements. On the basis of these studies, the researchers have found that MOOCs retention rate is very low and a large number of students have not completed their courses. Learners retention is an important measure of the success of MOOCs as only those who adhere to the curriculum have the opportunity to derive the expected educational benefits from their learning experience.

Some researchers then expect to find out the possible reasons for students to drop out of course by analyzing various records of the MOOCs. Yang et al. (2013) used the survival model to mine the social relationships of learners with the forum interactive data in a Coursera MOOC, and explored the impact of social relations on students dropping out. Onah et al. (2014) summarized the reasons for dropout through literature research, and then analyzed the relationships between dropouts and students demographic data of a MOOC on the Moodle<sup>3</sup> platform. Stein and Allione (2014) analyzed students retention of a MOOC on Coursera using the Cox proportional hazard model in combination with students demographic data and week 1 learning activity data. Wen et al. (2014) conducted an emotional analysis of forum posts of three MOOCs and studied the relationships between daily dropouts and the emotional proportion of the forum posts. Hone and Said (2016) concluded that the contents of the course and teachers are the key factors for students to keep learning through a detailed investigation and study of

<sup>3</sup> <https://www.moodle.org/>.

379 students participating in a MOOC. These studies focus on the various reasons why students drop out of a MOOC from different data sources, the relationships between students learning behaviors and dropouts, and indicate that dropouts can be predicted using such behavioral data. But these studies do not give a specific prediction model.

The data mining technology to dropout prediction has become another focus of research. Many experts and scholars have applied educational data mining techniques to predict student failure at school (Mrquez-Vera et al. 2013) and dropout (Mrquez-Vera et al. 2016). These techniques have also been utilized to predict the dropout in MOOCs. Balakrishnan and Coetzee (2013) used HMM to process courses clickstream data, forum threads and comments, and homework assignments, and mine the students behavior patterns to predict whether students will drop out next week. Halawa et al. (2014) developed two rule models and used the characteristics of students' learning activities to predict the dropouts. Jiang et al. (2014) used logistic regression to predict the probability of a student completing a course and obtaining a certificate with the students task performance and social interaction during the first week of a MOOC on Coursera. Kloft et al. (2014) trained SVM with clickstream data of different weeks gathered from a MOOC to predict whether the students will drop out finally. Ramesh et al. (2014) used the probabilistic soft logic framework to make probabilistic relational modeling on students learning activity records of three MOOCs on Coursera, and then applied this model to predict the completion of these courses. Sinha et al. (2014) adopted logistic regression on the interaction data of all the students video playing in a course on Coursera, and ultimately predicted the click behaviors and dropout situations of students later. Taylor et al. (2014) applied clickstream data and forum submission data of the course 6.002x on edX to train logistic regression to predict whether students will stop learning in the next week. Chaplot et al. (2015) used a Coursera MOOC's clickstream log and forum posting data to train artificial neural networks to predict whether students will drop out the following week. Fei and Yeung (2015) predicted whether students will drop out using RNN and HMM with clickstream data of different weeks. He et al. (2015) utilized logistic regression to analyze the completion of students courses, homework, and grading in Coursera MOOCs to make early dropout prediction for early intervention. Wang and Chen (2016) used a nonlinear state space model with clickstream data of different weeks to predict student dropouts. Xing et al. (2016) applied general Bayesian network (GBN) and decision tree (C4.5) to model the interaction data of a MOOC, so as to achieve early identification students who are at risk of dropping out. Hung et al. (2017) used the time series clustering method to classify students learning behaviors and explored how to identify students at risk as early as possible.

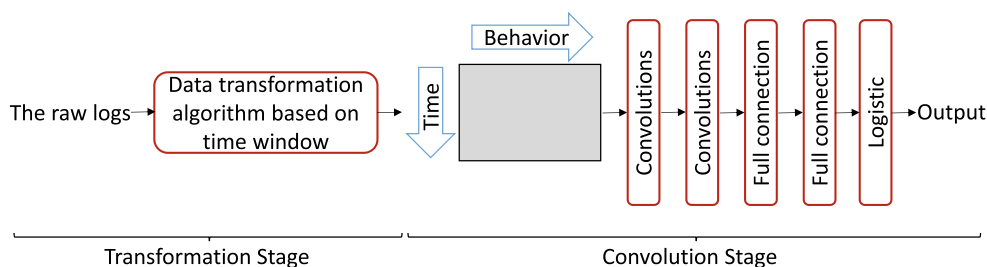
We can summarize the study of dropout prediction from two aspects: data sources, dropout prediction models in MOOCs based on clickstream data.

*Data Sources* As a MOOC platform, it will have a large number of registered students and a lot of courses. Each student has the freedom to choose a number of different courses, and each course has many different students to learn. Generally, each course lasts for a period of time. Students will have a lot of activity records when they learn these courses, including personal information, access to various course modules, watching videos, participating in discussion forums, completing quizzes, and scoring. This information can be divided into clickstream data (Balakrishnan and Coetzee 2013; Halawa et al. 2014; Kloft et al. 2014; Ramesh et al. 2014; Sinha et al. 2014; Stein and Allione 2014; Taylor et al. 2014; Chaplot et al. 2015; Fei and Yeung 2015; Wang and Chen 2016; Xing et al. 2016; Hung et al. 2017), assignment submission and performance information (Balakrishnan and Coetzee 2013; He et al. 2015; Xing et al. 2016), curriculum forum posts and interactive information (Balakrishnan and Coetzee 2013; Yang et al. 2013; Jiang et al. 2014; Taylor et al. 2014; Wen et al. 2014; Chaplot et al. 2015; Xing et al. 2016), student demographic information (Onah et al. 2014; Stein and Allione 2014), and so on. Although the integration of different types of data to forecast is the best way, now the learning activities of various data records are very unbalanced, for example, only a small percentage of the entire student population will communicate in the forum, so this type of data will be less to narrow. The most widely covered data is the log information of students' learning activities, that is, the clickstream data. Most of the research focuses on the analysis and prediction of clickstream data. This paper also studies the click stream data as the basis.

*Dropout prediction models in MOOCs based on clickstream data* Existing models of using clickstream data to predict dropout in MOOCs are mainly implemented by data mining technology based on features. According to the understanding of the problem, these methods can be divided into two categories: binary classification and time series classification.

The binary classification approach assumes that the dropout probability is independent, and treats the problem as an ordinary binary classification problem. Firstly, it manually extracts various features from the data, and then uses diverse machine learning methods or statistical analysis methods to predict. Typical methods include: SVM (Kloft et al. 2014), LR (Jiang et al. 2014; Sinha et al. 2014; Taylor et al. 2014; He et al. 2015), and so on.

According to the temporal characteristics of clickstream data, the time series classification method considers that the students' state will be affected by the previous state, and regards the final prediction as a time series classification forecasting problem. It first extracts the temporal features from



**Fig. 1** Overall architecture of OD-CNN

the data, and then uses sequence models such as HMM (Balakrishnan and Coetzee 2013), RNN (Fei and Yeung 2015), and nonlinear state space model (Wang and Chen 2016) to analyze and predict.

## 2.2 CNN for dropout prediction in MOOCs

In recent years, deep learning has shown very good performance in solving many problems (LeCun et al. 2015). Among different types of deep neural networks, the research on convolutional neural networks has emerged in large numbers and achieved first-class results. They have been successfully applied in many fields such as computer vision (Krizhevsky et al. 2012; Karpathy et al. 2014), speech recognition (Sainath et al. 2015) and natural language processing (Kim 2014). Convolutional neural networks are often used to process data that have the characteristics of a grid-like topology. And time series data can be viewed as one-dimensional grid data sampled at fixed time intervals (Goodfellow et al. 2016). Zheng et al. (2014) used convolutional neural networks to classify time series data and achieve good results. Inspired by these applications, we try to use CNNs to analyze the clickstream data of students, so as to make the prediction of dropout. Based on Google Scholars search results with the keyword dropout prediction and MOOCs in the period of 2012–2017, CNN is rarely used in dropout prediction in MOOCs.

## 3 Methodology

In this section, we first define the research problem, and then describe the implementation details of our dropout prediction model in MOOCs.

### 3.1 Problem definition

This paper uses learning activities of a student in a course that have already happened to judge whether the student will stop learning the course in the future; specifically, if the student does not have any learning activity in the course in a period of time, then the student drops out of this course. Therefore, the problem of dropout prediction in this paper can be defined as

**Table 1** An example of a raw record in MOOCs

Enrollment id	Time	Event
1	2014-06-14T09:38:29	Navigate

follows: Given the activity records of a student in a course in a given period, it is necessary to predict whether this student will drop out in the future.

### 3.2 Architecture of DP-CNN

The overall architecture of DP-CNN, as shown in Fig. 1, contains two sequential stages: transformation and convolution.

- (1) The transformation stage converts the raw logs into meaningful data that can be learned by the second stage. The research data object is clickstream behavior data, and such data are generally stored as a log file. Table 1 shows an example of a raw record in MOOCs used in this article. These original activity records are in text format and cannot be used directly as input for the second stage of our model. In order to use these data, we need to transform them into a format that can be fed into deep neural networks. To this end, we propose a data transformation algorithm based on time window (DTTW) for data transformation. See Sect. 3.3 for details.
- (2) In the convolution stage, we use several convolutional layers to extract features of the data output from the first stage and then apply the fully connected layers and logistic function to generate the final prediction results. This is a complete end-to-end system, and all parameters are co-trained by backpropagation. The details are shown in Sect. 3.4.

### 3.3 DTTW

For dealing with the clickstream data in MOOCs, it is easy to think of integrating all the behavior records of a student in a course into a record for one-dimensional feature extraction and processing. It has been widely applied in the classical classifiers, such as LR, SVM and so on, which

all receive one-dimensional features for training and prediction. However, this method ignores the time information that may be useful for prediction, and the successful applications of existing CNNs, such as image classification and text classification, are mostly implemented by two-dimensional convolution. Therefore, we propose a two-dimensional feature design scheme based on the temporal characteristics of the existing data:

*Time dimension* It describes the time sequence relationship of the time period of the record, and the length of the time period is defined by the question or the user. In this problem, it can be defined as a day, a week, etc. The number of time segments after split is the size of the dimension.

*Behavior dimension* In order to extract advanced features from the original data for CNNs, we have not do all kinds of complex feature extraction and transformation of the original data, only some basic statistics. Since we are dealing with the behavior log data of the clickstream, we count the number of various behaviors in the corresponding time period as the value of this dimension, which size is consistent with the number of types of behaviors.

By analyzing the students learning behavior logs and combining the proposed two-dimensional feature design scheme, we propose a data transformation algorithm based on time window (DTTW) to transform the original records. Although currently there is no uniform standard for learning activity records of students on MOOC platforms, the records of each platform vary. However, from the records of the two mainstream MOOC platforms, Coursera (Veeramachaneni et al. 2014) and edX (Breslow et al. 2013), these learning activity records are generally stored as structured log data. Each record at least consists of the operator (a student who attended a course), the time, and the type of an event. Algorithm 1 summarizes the transform process of DTTW.

### 3.4 Convolution stage

After processing through DTTW, each record in the output dataset  $F$  is an instance, and we select the event count set containing two dimensions in the instance as the input that we need to feed into the convolution stage.

The convolution stage, shown in Fig. 1, is a four-layer convolutional neural network with two convolutional layers, and two fully connected layers. The output of the final fully connected layer is passed into a logistic classifier to generate the probability of two classifications. In order to preserve all the feature information, no pooling layer is used.

As we input two-dimensional data, we use a two-dimensional convolution kernel for processing. The role of the convolution kernel in each layer is to extract the different feature maps of the input. In CNN, the feature map  $a_j^l$  for

---

#### ALGORITHM 1: Data transformation Algorithm Based on Time Window (DTTW)

---

```

Input: Courses set  $C$ , students set  $S$ , event types set  $ET$ , events set  $E$  and predicted time period  $t$ .
Output: the dataset  $F$  with two-dimensional features.
1 Set  $F = \emptyset$ 
2 foreach course  $c_i \in C$  do
3   Find events set  $E^i \in E$  about  $c_i$ 
4   Get the start time  $TS_i$  and the end time  $TE_i$  of  $E^i$ 
5   Set course  $c_i$ 's time windows set  $W^i = \emptyset$ 
6   for  $j = 0; j \leq \frac{TE_i - TS_i}{t}; j++$  do
7      $StartTime_j^i = TS_i + j * t$ 
8      $EndTime_j^i = \min(StartTime_j^i + t, TE_i)$ 
9      $W^i = W^i \cup \{(StartTime_j^i, EndTime_j^i)\}$ 
10  end
11  Find students set  $S^i \in S$  of  $c_i$ 
12  foreach student  $s_k^i \in S^i$  do
13    Get events set  $E_k^i \in E^i$  of  $s_k^i$ 
14    Set event count set  $ET_k^i = \{ET_{k,m,n}^i = 0 | m = \text{length}(W^i), n = \text{length}(ET)\}$ 
15    foreach event  $e_{kl}^i \in E_k^i$  do
16      if  $e_{kl}^i$ 's time in  $w_m^i \in W^i$  and  $e_{kl}^i$ 's event type is  $ET_n \in ET$  then
17         $ET_{k,m,n}^i = ET_{k,m,n}^i + 1$ 
18      end
19    end
20     $F = F \cup \{(c_i, s_k^i, ET_k^i)\}$ 
21  end
22 end

```

---

each output of the convolutional layer  $l$  is:

$$a_j^l = f \left( b_j^l + \sum_{i \in M_j} w_{ji}^l * a_i^{l-1} \right). \tag{1}$$

where  $M_j$  represents the selected combination of input feature maps,  $w_{ji}^l$  is the convolution kernel weight used for the connection between the input  $i$ th feature map and the output  $j$ th feature map in the  $l$ th layer,  $*$  denotes the convolution operation,  $b_j^l$  is the bias corresponding to the  $j$ th feature map in the  $l$ th layer,  $f$  is a nonlinear activation function such as tanh-function or rectified linear unit (ReLU) function. At present, most of the convolutional neural networks basically use the ReLU function, because it can make the convolution neural network achieve better results (Goodfellow et al. 2016). In this model, the ReLU function is also used. The ReLU function can be represented as follows:

$$g(z) = \max(0, z). \tag{2}$$

The third and the fourth layers are fully connected. The third layer accepts the output of the upper layer and produces the input of the fourth layer. The fourth layer accepts the

input for the final classification calculation. The calculation expression can be shown as follows:

$$a_j^l = f \left( b_j^l + \sum_{i=1}^{N_{l-1}} w_{ji}^l a_i^{l-1} \right), \quad (3)$$

where  $a_j^l$  is the output of the  $j$ th neuron of the  $l$ th layer, which is also the input of the neurons of the  $(l+1)$ th layer.  $N_{l-1}$  is the number of neurons in the  $(l-1)$ th layer.  $w_{ji}^l$  is the weight from the  $i$ th neuron in the  $(l-1)$ th layer to the  $j$ th neuron in the  $l$ th layer,  $b_j^l$  is the bias of the  $j$ th neuron in the  $l$ th layer, and  $f$  is a nonlinear activation function. In the third level,  $f$  is the ReLU function. In this problem, the final problem is a binary classification problem, so in the fourth level, we use sigmoid function (also called logistic function) to produce the final classification result. We define “0” as “not dropout” and “1” as “dropout”. Sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (4)$$

The model uses cross-entropy as a cost function that is defined as

$$C = -\frac{1}{m} \sum_{k=1}^m (y_k \log \hat{y}_k + (1 - y_k) \log(1 - \hat{y}_k)). \quad (5)$$

where  $m$  is the number of training instances,  $y_k$  is the expectation output of the  $k$ th training instance and  $\hat{y}_k$  is the prediction results of the  $k$ th training instance.

We use the backpropagation (BP) algorithm to calculate the gradient of the model parameters. Finally, mini-batch stochastic gradient descent is used to train the parameters of the network. The main parameters are connection weight between neurons  $w^l$  in the  $l$ th layer, and each neuron's own bias  $b^l$  in the  $l$ th layer. The updating expression of the gradient descent method is shown as follows:

$$w_{\text{new}}^l = w_{\text{old}}^l - \eta \frac{\partial C}{\partial w_{\text{old}}^l}, \quad (6)$$

$$b_{\text{new}}^l = b_{\text{old}}^l - \eta \frac{\partial C}{\partial b_{\text{old}}^l}. \quad (7)$$

Here,  $\eta$  is the gradient descent learning rate.

In order to obtain the partial derivative of the error cost function associated with layer  $l$ , node sensitivities  $\delta^l$ , also called residuals, are introduced to compute, which are defined as how much impact the node has on the residuals of the final output value.

$$\delta^l = \frac{\partial C}{\partial z^l}. \quad (8)$$

In the fully connected layer,

$$z^l = b^l + w^l a^{l-1}. \quad (9)$$

Here,  $a^{l-1}$  is the output of the layer  $l-1$ , which is the input of layer  $l$ ,  $w^l$  and  $b^l$  are the weight and bias of layer  $l$ .

For the bias  $b^l$  in the parameters, since  $\frac{\partial z^l}{\partial b^l} = 1$ , by the chain rule:

$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial z^l} \frac{\partial z^l}{\partial b^l} = \delta^l, \quad (10)$$

$$\frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial z^l} \frac{\partial z^l}{\partial w^l} = \delta^l (a^{l-1})^T. \quad (11)$$

Then, we have obtained the gradient needed for each layer to update the weights and biases by using the gradient descent. We can see from Eqs. (10) and (11) that they are related to the sensitivity.

In the output layer  $L$ , the components of  $\delta^L$  are given by

$$\delta^L = \nabla_a C \odot f'(z^L). \quad (12)$$

Here,  $\nabla_a C$  is defined to be a vector whose components are the partial derivatives  $\frac{\partial C}{\partial a_k^L}$ ,  $a_k^L$  is the  $k$ th output.

And in the other fully connected layer, the sensitivity can be calculated by Eq. (13).

$$\begin{aligned} \delta^L &= \frac{\partial C}{\partial z^l} = \frac{\partial C}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial z^l} \\ &= \delta^{l+1} \frac{\partial (b^{l+1} + w^{l+1} f(z^l))}{\partial z^l} \\ &= \delta^{l+1} (w^{l+1})^T \odot f'(z^l), \end{aligned} \quad (13)$$

where “ $\odot$ ” denotes element-wise multiplication.

And in the convolutional layer, the sensitivity can be calculated by Eq. (14).

$$\delta^L = \delta_j^{l+1} * \text{rot180}(w_j^{l+1}) \odot f'(z^l). \quad (14)$$

Here,  $\text{rot180}(\cdot)$  indicates that the convolution kernel  $w_j^{l+1}$  has been rotated  $180^\circ$ . Then, the partial derivative of the error cost function to  $b^l$  and  $w_{ij}^l$  can be written as

$$\frac{\partial C}{\partial b_j^l} = \sum_{u,v} (\delta_j^l)_{uv}, \quad (15)$$

$$\frac{\partial C}{\partial w_{ij}^l} = \sum_{u,v} (\delta_j^l)_{uv} (p_i^{l-1})_{uv}. \quad (16)$$

where  $(\cdot)_{uv}$  traverses all the elements of  $\cdot$ ,  $(p_i^{l-1})_{uv}$  is a matrix of related elements of  $z_i^{l-1}$  in the layer  $l-1$  connected by the  $(\delta_j^l)$ .

**Table 2** Attributes in logs

Attribute	Description
Enrollment_id	ID of the enrollment record
Time	Time when the event occurs
Source	Event source (server or browser)
Event	Event type
Object	The object the student access or navigate to (Only for navigate and access event )

**Table 3** Statistics of dataset for the experiment

Item	Statistical description
Courses	39
Students	79,186
Enrollments	120,542
Activity logs	8,157,277

## 4 Experiment

### 4.1 Dataset

The dataset used in our experiment is a public dataset provided by KDD Cup 2015. This dataset contains 39 courses information on the MOOC platform XuetangX within a period of time. During this period, 79,186 students participated in these courses, and created a total of 120,542 enrollment information. As the course progresses, all student activities in these courses are recorded as events in a log file by the MOOC platform, as shown in Table 2, which includes seven different types of activity records of students, such as viewing lecture navigation, watching course videos, doing exercises, reading Wikipedia of the course, participating in discussion forums, accessing other course modules, and closing the web page. The basic statistical information of the dataset is shown in Table 3. There are altogether 8,157,277 event records, and the interval between each course is only 30 days.

Because of the need for data desensitization, all course names have been hashed, and in order to facilitate presentation and post-processing, we use the course number instead of the course name. Figure 2 shows the number of students enrolled in these 39 courses for a given period of time, the number of dropouts in the future, and the dropout rate of each course. From the dropout rate of each course, it can be seen that the dropout rate of students in MOOCs is generally high and the average dropout rate reaches more than 80%. This once again proves that the dropout problem is indeed an important issue in MOOCs.

In order to verify that the student's activity record is related to the final dropout, we counted the total number of access

days for each enrolled student in the course learning periods, then counted the number of enrollments and dropouts corresponding to different access days, and calculated the corresponding students' dropout ratio, as shown in Fig. 3. From it, we can see that the proportion of students dropping out of courses decreases with the increase in days they visit, which means the more students who visit the course, the less likely they are to drop out. This can indicate that the students participating in the study are indeed related to the final dropout. It also shows that it is reasonable to select the student activity records for the dropout prediction.

### 4.2 Prediction goal

With regard to the definition of dropout, different literatures have different descriptions. To facilitate verification and testing, we adopt the definition of the problem by KDD Cup 2015: If a student has no record of course activity within the next 10 days, then he/she is regarded as dropout.

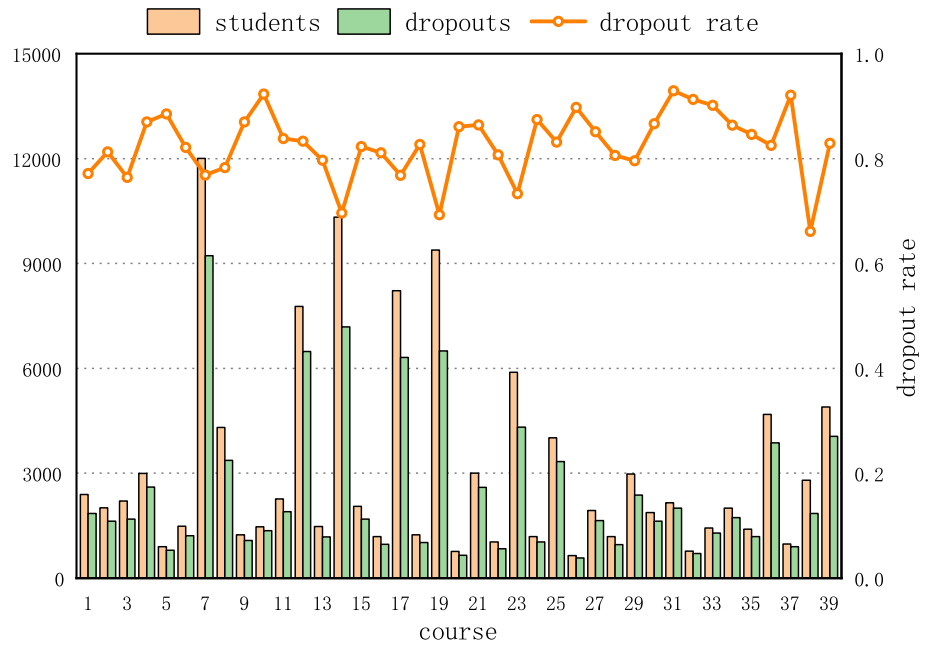
### 4.3 Evaluation metrics

In this experiment, *precision*, *recall*, *F1 score*, and area under the receiver operating characteristic curve (*AUC*) score are used as evaluation metrics. They are often used to evaluate the performance of a binary classifier. *Precision* is a measure of accuracy and indicates the proportion of actual positive cases in examples classified as positive. *Recall* rate is a metric of coverage, which measures how much actual positives are classified as positive examples. The *F1 score* is the harmonic average of accuracy and recall. From Fig. 3, we can see that the proportion of "dropout" in the dataset is much larger than that of "not dropout". Therefore, this experiment is to solve a binary classification problem with unbalanced data. The receiver operating characteristic (ROC) curve has a good characteristic: When the distribution of positive and negative samples in the test set is changed, the ROC curve can remain unchanged; that is, the ROC curve is not affected by the proportion of positive and negative categories, while other evaluation metrics will be affected (Davis and Goadrich 2006).

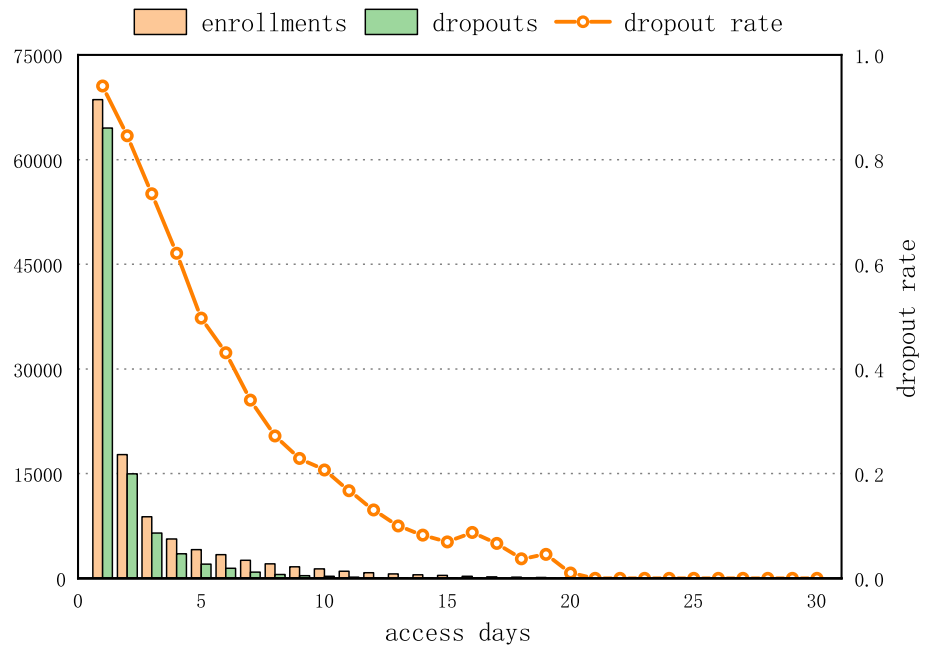
Consider a binary classification question, dividing an instance into a positive or a negative, there will be four situations.

- *True Positive (TP)* The number of instances that are correctly classified as positive, that is, the number of instances that are actually positive and classified as positive by the classifier;
- *False Positive (FP)* The number of instances that are incorrectly classified as positive, i.e., the number of instances that are actually negative but classified as positive by the classifier;

**Fig. 2** The number of students, of dropouts, and dropout rates in different courses



**Fig. 3** The number of enrollments, dropouts, and dropout rates for different access days



- *False Negative (FN)* The number of cases that are incorrectly divided into negative cases, that is, the number of instances that are actually positive but classified into negative cases by the classifier;
- *True Negative (TN)* The number of instances correctly classified as negative, i.e., the number of instances that are actually negative and classified by the classifier as negative.

From these, we can calculate various evaluation criteria for the model. The formulas for precision  $P$ , recall  $R$ , and F1 score  $F1$  are defined as follows:

$$P = \frac{TP}{TP + FP}, \tag{17}$$

$$R = \frac{TP}{TP + FN}, \tag{18}$$

$$F1 = \frac{2TP}{2TP + FP + FN}. \tag{19}$$

The ROC curve is based on a series of different binary classifications with different decision threshold, plotted with the true positive rate (TPR) as the vertical axis and the false positive rate (FPR) as the horizontal axis. The calculation formulas of TPR and FPR are



$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \quad (20)$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (21)$$

AUC is defined as the area under the ROC curve, and it is clear that the area is not more than 1. Moreover, since the ROC curve is generally above the  $y = x$  line (the classifier using random guess strategy), the range of AUC is between 0.5 and 1. Using AUC score as an evaluation metric is because ROC curves often do not clearly indicate which classifier is better, and as a numerical value, AUC score can be used to show which classifier is better more clearly (Fawcett 2006).

#### 4.4 Compared methods

In order to verify the effectiveness of our proposed model, we chose seven mainstream methods in researches of educational data mining as baseline methods and provide them with features via manual feature extraction. The experimental results of these methods based on feature engineering are then compared with the experimental results of the DP-CNN, which directly uses the log data.

The baseline methods are:

- (1) *Logistic regression (LR)* Logistic regression is a classic binary classification method. It assumes that the data obey the Bernoulli distribution and uses gradient descent to solve the parameters by maximizing the likelihood function to achieve the purpose of classification. Logistic regression with features via manual extraction is usually used to predict whether the students will stop learning in MOOCs (Jiang et al. 2014; Sinha et al. 2014; Taylor et al. 2014; He et al. 2015).
- (2) *Naive Bayes* (Ng and Jordan 2002) The Naive Bayes method is a statistical classification method based on Bayes theorem and the independence assumption of feature conditions. It uses the Bayes formula to obtain the conditional probability of each class for the sample and takes the class corresponding to the maximum conditional probability as the class corresponding to the sample. In this paper, we choose Gaussian Naive Bayes to classify whether students drop out of school.
- (3) *Decision Tree* (Murthy 1998) The decision tree is based on the tree structure for decision making. Each non-leaf node in the tree represents a test on a feature attribute, each branch represents the output of the feature attribute on a certain range, and each leaf node stores a category. The decision process using the decision tree is to start from the root node, test the corresponding feature attributes in the item to be classified, and select the output branch according to its value until the leaf node is reached, and the category stored by the leaf node is used as the decision result. The information gain, gain rate, and Gini index can be selected as the basis for optimal attribute partitioning. This paper chooses the Gini index as the basis for the division.
- (4) *AdaBoost* (Freund and Schapire 1997) AdaBoost adopts the idea of iteration. Each iteration only trains one weak classifier. The trained weak classifier will participate in the next iteration. Generally, a single-layer decision tree is used as its weak classifier. There are two kinds of weights in the AdaBoost algorithm, one is the weight of the data, and the other is the weight of the weak classifier. Among them, the weight of the data is mainly used to build a new weak classifier to find the decision point with the minimum classification error, and then use this minimum error to calculate the weight of the weak classifier. Finally, all weak classifiers are combined to get a final ensemble classifier.
- (5) *Gradient Tree Boosting* (Friedman 2001) Gradient tree boosting is similar to AdaBoost, whose base classifier is a decision tree, but each new classifier is built to reduce the residual of the previous classifier to the gradient direction.
- (6) *Random Forest* (Breiman 2001) Random forest is based on constructing Bagging ensemble with decision tree as a base learner and introduces random attribute selection in the decision tree training process to improve the generalization performance of the final ensemble classifier.
- (7) *Support vector machine (SVM)* Support vector machine is a supervised machine learning method commonly used in binary classification applications. The support vector machine method is based on the VC dimension theory of statistical learning theory and the minimum structure risk principle. It has many unique advantages in solving small sample, nonlinear, and high-dimensional pattern recognition. It has the input space nonlinearly mapped to the high-dimensional feature space, which is different from the commonly used methods of dimensionality reduction in traditional pattern recognition, so that the linearly inseparable classes in the low-dimensional space can be linearly separable in the high-dimensional space. Kloft et al. (2014) used SVC with the radial basis function (RBF) kernel to predict the dropout. And we also use SVC with RBF kernel in our experiments.

#### 4.5 Implementation details

In our algorithm, we feed the log dataset into DP-CNN for processing. In the transformation stage, we select the representative time window commonly used in the existing literatures, a week, and the time window which is consistent with the above problem definition, 10 days, and then use these time windows to transform the data through DTTW.

Finally, two different datasets are obtained, and then, these datasets are sent to the convolution stage for feature extraction and prediction. The first layer of the convolution stage of DP-CNN uses 32 filters and biases, and the second layer uses 64 filters and biases. The size of these filters is  $3 \times 3$ , the step is set to 1, and when the convolution calculation is carried out, the boundaries are filled with “0”. Then, the first layer outputs 32 feature maps of the same size as the input, and the second layer outputs 64 feature maps of the same size as the input. The third layer has 128 neurons, and some neurons are dropout randomly at a probability of 50% when training to prevent overfitting (Srivastava et al. 2014). The fourth layer outputs the prediction result, and since the final issue to be solved is the binary classification problem, the number of neurons in this layer is set to 2. Here, we denote “1” as “dropout” and “0” as “not dropout”. In the training process, the mini-batch stochastic gradient descent optimization algorithm is used for parameter learning (Ruder 2016), and the learning rate is set to 0.001, batch size is set to 256, and the number of iterations is 20,000.

For experimental comparison, we extract 78 features from the raw records through feature engineering. These features mainly include three aspects: (1) various statistics of learning time; (2) various statistics of different events; and (3) statistics of operation object categories. All features are normalized by min-max scaling to convert the range of features to the 0 to 1 range. These features will be used for training and prediction of the above baseline methods.

We use fivefold cross-validation to select the parameters for the baseline methods. The maximum number of iterations  $n\_estimators$  for AdaBoost and Gradient tree boosting is 100, and the learning rate  $learning\_rate$  is 1 and 0.1, respectively. The parameter  $n\_estimators$  of random forest is 60, and the maximum depth  $max\_depth$  of the decision tree is 13. The penalty coefficient  $C$  of SVC with RBF kernel is 10, and the kernel function parameter  $\gamma$  is 1.

In order to train and test these models, we randomly divide each dataset into a training set and a testing set in the proportion of 3:1. Finally, the implementation and comparison of various models are carried out on Sklearn and Tensorflow.

## 4.6 Results and discussion

**Single Course** Taking into account the course arrangements and the learner’s interest are different between different courses, we first model each course separately. In order to reduce the amount of calculation and ensure the training data of the model, we only choose courses with a registration number greater than 5000 for processing. Finally, we select the six courses numbered 7, 12, 14, 17, 19, and 23 in Fig. 2 to carry out the specific experiments, with the number of registrations being 12,004, 7775, 10,322, 8223, 9382, and 5890. Then, the corresponding data of the six courses are

extracted, respectively, from the original dataset and the feature set which contains 78 features for training and testing. The test results of these experiments in different algorithms with the six courses are shown in Table 4. The best values for the experimental results in each course are indicated in bold. Here, the baseline algorithms use the feature set which contains 78 features that are extracted from the corresponding courses.

As can be seen from Table 4, the comparison of DP-CNN with other baseline algorithms does not lead to an intuitive conclusion. In order to accurately describe the differences between DP-CNN and the baseline algorithms, we use the Friedman test to analyze the experimental results. We applied the Friedman test to detect whether there exist significant differences among DP-CNN and other baseline algorithms ( $\alpha = 0.05$ ).  $H_0$  hypothesis assumes that the results of the algorithms are equivalent. Table 5 shows the average ranking of all algorithms for different metrics, and Table 6 reports the results of the Friedman test on the single courses for different metrics. Friedman statistic is distributed according to chi-square with 8 degrees of freedom.

From Table 6, we could conclude that the four tests find significant differences among the considered algorithms because  $p$  values  $< \alpha$ . ( $H_0$  hypothesis of equivalence of rankings is rejected.) Hence, we perform the post hoc Nemenyi test to compare all algorithms to each other. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \quad (22)$$

where critical values  $q_\alpha$  are based on the studentized range statistic divided by  $\sqrt{2}$  (Table 7) (Demar 2006).

With nine algorithms and six courses datasets, the critical value for  $\alpha = 0.05$  is 3.102 and the corresponding CD is  $3.102 \sqrt{\frac{9 \times 10}{6 \times 6}} = 4.905$ . We can identify that the performance of the four metrics of decision tree is significantly worse than that of LR, random forest, and gradient tree boosting, and decision tree performs also significantly worse than DP-CNN with different window size on *F1 Score* and *AUC*. Gaussian Naive Bayes performs significantly worse than LR on *precision* and *recall*, and worse than gradient tree boosting and random forest on *AUC*.

At  $\alpha = 0.10$ , CD is  $2.855 \sqrt{\frac{9 \times 10}{6 \times 6}} = 4.514$ . Then, DP-CNN with different window size performs significantly better than decision tree on *precision* and *recall*, and better than Gaussian Naive Bayes on *AUC*. Gaussian Naive Bayes performs significantly worse than random forest on *precision* and worse than LR on *AUC*.

From the above results of the statistical analysis, the performance of the LR algorithm is relatively prominent; random forest, gradient tree boosting, and our algorithm DP-

**Table 4** Results of the experiments in different algorithms with one course

Course	Algorithm	Precision	Recall	F1 Score	AUC
7	LR	0.8469	0.8534	0.8411	0.8551
	Gaussian Naive Bayes	0.8228	0.8251	0.8238	0.7903
	Decision Tree	0.7777	0.7801	0.7788	0.6840
	Random Forest	<b>0.8480</b>	<b>0.8547</b>	0.8436	<b>0.8668</b>
	Gradient Tree Boosting	0.8477	<b>0.8547</b>	<b>0.8442</b>	0.8651
	AdaBoost	0.8365	0.8454	0.8357	0.8533
	SVM (with RBF kernel)	0.8444	0.8517	0.8404	0.8503
	DP-CNN (with window size = 7)	0.842	0.8494	0.8368	0.8607
	DP-CNN (with window size = 10)	0.8441	0.8507	0.8375	0.8550
	12	LR	0.8882	0.8951	0.8863
Gaussian Naive Bayes		0.8666	0.8627	0.8645	0.8126
Decision Tree		0.8400	0.8390	0.8395	0.7135
Random Forest		0.8899	0.8966	0.8898	0.8810
Gradient Tree Boosting		0.8887	0.8951	0.8898	0.8783
AdaBoost		<b>0.8931</b>	<b>0.8992</b>	<b>0.8936</b>	0.8608
SVM (with RBF kernel)		0.8856	0.8920	0.8802	0.8715
DP-CNN (with window size = 7)		0.8876	0.8945	0.8857	0.8796
DP-CNN (with window size = 10)		0.8871	0.8935	0.8827	<b>0.8836</b>
14		LR	0.8538	<b>0.8566</b>	0.8534
	Gaussian Naive Bayes	0.8230	0.8276	0.8230	0.8313
	Decision Tree	0.7725	0.7726	0.7725	0.7278
	Random Forest	0.8489	0.8520	0.8487	0.9006
	Gradient Tree Boosting	<b>0.8540</b>	<b>0.8566</b>	<b>0.8545</b>	<b>0.9072</b>
	AdaBoost	0.8397	0.8431	0.8402	0.8957
	SVM (with RBF kernel)	0.825	0.8194	0.8216	0.8946
	DP-CNN (with window size = 7)	0.8453	0.8481	0.8428	<b>0.9072</b>
	DP-CNN (with window size = 10)	0.8462	0.8493	0.8448	0.9055
	17	LR	<b>0.8356</b>	<b>0.8371</b>	0.8178
Gaussian Naive Bayes		0.8029	0.8118	0.8056	0.7420
Decision Tree		0.7409	0.7432	0.7420	0.6487
Random Forest		0.8252	0.8322	0.8162	<b>0.8194</b>
Gradient Tree Boosting		0.8179	0.8273	0.8131	0.8188
AdaBoost		0.8086	0.8191	0.8012	0.8062
SVM (with RBF kernel)		0.8247	0.8157	0.7804	0.8132
DP-CNN (with window size = 7)		0.8197	0.8259	0.8218	0.8131
DP-CNN (with window size = 10)		0.8244	0.8307	<b>0.8264</b>	0.8153
19		LR	<b>0.8419</b>	<b>0.8444</b>	<b>0.8387</b>
	Gaussian Naive Bayes	0.8091	0.8142	0.8095	0.8099
	Decision Tree	0.7621	0.7621	0.7621	0.7220
	Random Forest	0.8332	0.8363	0.8301	0.8788
	Gradient Tree Boosting	0.8392	0.8423	0.8372	0.8792
	AdaBoost	0.8329	0.8359	0.8294	0.8748
	SVM (with RBF kernel)	0.8241	0.8257	0.8248	0.8763
	DP-CNN (with window size = 7)	0.8358	0.8376	0.8300	0.8771
	DP-CNN (with window size = 10)	0.8353	0.8359	0.8270	0.8774
	23	LR	0.8428	0.8466	0.8382
Gaussian Naive Bayes		0.8347	0.8377	0.8358	0.8247
Decision Tree		0.7513	0.7502	0.7507	0.6898

**Table 4** continued

Course	Algorithm	Precision	Recall	F1 Score	AUC
	Random Forest	0.8336	0.8391	0.8327	0.8662
	Gradient Tree Boosting	0.8406	0.8452	0.8381	0.8692
	AdaBoost	0.8249	0.8310	0.8222	0.8634
	SVM (with RBF kernel)	0.8328	0.8364	0.8253	0.8667
	DP-CNN (with window size = 7)	<b>0.8452</b>	<b>0.8493</b>	<b>0.8421</b>	<b>0.8785</b>
	DP-CNN (with window size = 10)	0.8428	0.8473	0.8403	0.8759

**Table 5** Average rankings of the algorithms for different metrics

Algorithm	Precision	Recall	F1 Score	AUC
LR	2.25	2.1667	2.6667	3.3333
Gaussian Naive Bayes	7.5	7.5	7	8
Decision Tree	9	9	9	9
Random Forest	3.1667	2.9167	3.4167	3
Gradient Tree Boosting	3	2.75	2.5833	2.5833
AdaBoost	5.8333	5.5833	5.6667	6.6667
SVM (with RBF kernel)	5.8333	6.6667	6.8333	6
DP-CNN (with window size = 7)	4.1667	4.1667	3.8333	3.25
DP-CNN (with window size = 10)	4.25	4.25	4	3.1667

**Table 6** Results of the Friedman test on the single courses for different metrics ( $\alpha = 0.05$ )

Evaluation metrics	Value in $\chi^2$	$p$ value
Precision	31.8555	< 0.0001
Recall	35.2444	< 0.0001
F1 Score	31.9666	< 0.0001
AUC	38.2555	< 0.0001

CNN are also close to it, while decision tree and Gaussian Naive Bayes perform poorly. It shows that DP-CNN is effective under a small amount of data of a single course.

*All Courses* Although it is the best way to predict whether or not a student will drop out of a course by using the same course previous data to train the model, the training of the model is likely to be overfitting due to the limited amount of data in a course, especially for the newly opened courses. Therefore, this problem can be solved by training and forecasting with data from multiple courses. For verification, we select all the data to experiment with. The specific values of *precision*, *recall*, *F1 score*, and *AUC* of different models in the experiments are shown in Table 8. The best values are indicated in bold. Also here the baseline algorithms use a collection of features that are manually extracted from all courses.

Comparing the experimental results shown in Tables 4 and 8, it can be seen that the performance of the baseline algorithms under each metric is closer to the mean value of the performance with a single course, which shows that it is feasi-

ble to train and predict using the data of different courses, and it can excavate the essence of students' learning behaviors when the data volume is large, so as to achieve more generalization performance than the training with single class.

We have used the relative differences between the performances of DP-CNN (with window size = 10) and each of the other methods on the four metrics. This relative difference is defined as  $\frac{v_1 - v_2}{v_2}$ , where  $v_1$  is a performance value of DP-CNN (with window size = 10) and  $v_2$  is a performance value of a compared algorithm on the same metric. If the relative difference is bigger, it shows that the difference between DP-CNN (with window size = 10) and the compared algorithm is more significant.

The values of relative differences are shown in Table 9. From this table, we can see that our DP-CNN algorithm is superior to the selected baseline methods at two time windows of different size in the case of a large amount of data. But from the results analysis in the above experiments, we also know that the performance of LR is prominent when using a single course dataset. Comparing the experiments of these two different situations, we can see that the biggest difference is that the datasets used are different. The maximum number of instances of datasets for a single course is 12,004, while the number of instances for all courses is 120,542, a difference of more than ten times. DP-CNN is essentially a deep learning method. Deep learning requires a large amount of data, the higher the amount of data, the better the performance, while the performance of traditional machine learning including LR tends to plateau after reaching a certain amount of data. Therefore, LR performs better

**Table 7** Critical values for the two-tailed Nemenyi test after the Friedman test

Number of classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.96	2.343	2.569	2.728	2.85	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.78	2.855	2.92

**Table 8** Results of the experiments in different algorithms with all courses

Algorithm	Precision	Recall	F1 Score	AUC
LR	0.8179	0.8241	0.8118	0.8435
Gaussian Naive Bayes	0.7935	0.8038	0.7922	0.7574
Decision Tree	0.7525	0.7523	0.7524	0.6890
Random Forest	0.8238	0.8304	0.8216	0.8510
Gradient Tree Boosting	0.8293	0.8354	0.8276	0.8600
AdaBoost	0.8277	0.8338	0.8257	0.8550
SVM (with RBF kernel)	0.8118	0.8160	0.7980	0.8426
OD-CNN (with window size = 7)	<b>0.8652</b>	0.8708	0.8577	0.8660
OD-CNN (with window size = 10)	0.8644	<b>0.8717</b>	<b>0.8627</b>	<b>0.8675</b>

**Table 9** Relative difference between DP-CNN (with window size = 10) and various other algorithms on different metrics (in %)

Algorithm	Precision	Recall	F1 Score	AUC
LR	5.69	5.78	6.27	2.85
Gaussian Naive Bayes	8.94	8.45	8.90	14.54
Decision Tree	14.87	15.87	14.66	25.91
Random Forest	4.93	4.97	5.00	1.94
Gradient Tree Boosting	4.23	4.35	4.24	0.87
AdaBoost	4.43	4.55	4.48	1.46
SVM (with RBF kernel)	6.48	6.83	8.11	2.96
DP-CNN (with window size = 7)	-0.09	0.10	0.58	0.17

when the amount of data is small, and DP-CNN performs better when the amount of data is large. So the performance of LR is outstanding when using a small dataset of a course and DP-CNN performs better in a big dataset of all courses.

It also can be seen from Table 9 that DP-CNN (with window size = 10) is better than DP-CNN (with window size = 7) on *recall*, *F1 score*, and *AUC* in the results with all courses. One possible reason is that the size of the time window selected by DP-CNN (with window size = 10) is consistent with the size of the time period to be predicted in the problem definition. This provides a certain reference for how to deal with the raw dataset so as to better solve the problem.

From the above results, we can conclude: (1) our proposed DP-CNN model is effective in solving dropout prediction problem in MOOCs with large amount of data and achieves better results than the baseline methods. (2) In the transformation stage, time windows of different scales have a certain impact on the prediction effect of the whole method, and it is best to set the size of time windows according to the problem to be solved. (3) Different students have some commonalities

in learning behavior patterns in different courses, so existing models that are trained with clickstream data of all existing courses can be used to predict the dropout for new courses.

## 5 Conclusion

In this paper, we propose a dropout prediction model based on two-dimensional convolutional neural networks (DP-CNN), which is used to predict students' dropouts in MOOCs based on clickstream data of students learning behaviors. The model DP-CNN can directly process clickstream data and automatically extract features for final prediction. It is a complete end-to-end system that reduces the complexity of handling this dropout prediction issue. The proposed model is validated on the dataset of 39 courses collected from XuetangX. The experimental results show that our prediction model has achieved better results than the related baseline methods under the sufficient data and it is an effective method. For future work, we will study how to enhance dropout prediction effect in MOOCs by combining other

information from multiple sources, such as assignment submission and performance information, course forum posts and interaction information, and then further expand the research and application of CNN in MOOCs.

**Acknowledgements** This work is supported by the National Social Science Fund of China for Young Project (13CYY037) and Educational Informatization Research Center of Hubei, Central China Normal University. We would like to gratefully acknowledge the organizers of KDD Cup 2015 as well as XuetangX for making the datasets available.

## Compliance with Ethical Standards

**Conflict of interest** The Authors declare that they have no conflict of interest.

## References

- Balakrishnan G, Coetzee D (2013) Predicting student retention in massive open online courses using hidden Markov models. *Electrical Engineering and Computer Sciences, University of California at Berkeley*
- Breiman L (2001) Random forests. *Mach Learn* 45(1):5–32
- Breslow L, Pritchard DE, DeBoer J, Stump GS, Ho AD, Seaton DT (2013) Studying learning in the worldwide classroom: research into edX's first MOOC. *Res Pract Assess* 8:13–25
- Chaplot DS, Rhim E, Kim J (2015) Predicting Student attrition in MOOCs using sentiment analysis and neural networks. In: *Proceedings of the 2015 AIED workshop on intelligent support for learning in groups*, pp 7–12
- Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. In: *Proceedings of the 23rd international conference on Machine learning, ACM*, pp 233–240
- DeBoer J, Stump GS, Seaton D, Breslow L (2013) Diversity in MOOC students backgrounds and behaviors in relationship to performance in 6.002 x. In: *Proceedings of the sixth learning international networks consortium conference*, vol 4
- Demar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7(1):1–30
- Fawcett T (2006) An introduction to ROC analysis. *Pattern Recogn Lett* 27(8):861–874
- Fei M, Yeung DY (2015) Temporal models for predicting student dropout in massive open online courses. In: *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, pp 256–263
- Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 55(1):119–139
- Friedman JH (2001) Greedy function approximation: a gradient boosting machine. *Ann Stat* 29:1189–1232
- Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp 315–323
- Goodfellow I, Bengio Y, Courville A (2016) *Deep learning*. MIT Press, Cambridge
- Halawa S, Greene D, Mitchell J (2014) Dropout prediction in MOOCs using learner activity features. *Exp Best Pract Around MOOCs* 7:3–12
- He J, Bailey J, Rubinstein BI, Zhang R (2015) Identifying at-risk students in massive open online courses. In: *Proceedings of the 29th AAAI conference on artificial intelligence*, pp 1749–1755
- Hone KS, Said GRE (2016) Exploring the factors affecting MOOC retention: a survey study. *Comput Educ* 98(Supplement C):157–168
- Hung JL, Wang MC, Wang S, Abdelrasoul M, Li Y, He W (2017) Identifying at-risk students for early interventions—a time-series clustering approach. *IEEE Trans Emerg Top Comput* 5(1):45–55
- Jiang S, Williams A, Schenke K, Warschauer M, O'dowd D (2014) Predicting MOOC performance with week 1 behavior. In: *Proceedings of the 7th international conference on educational data mining*
- Jordan K (2014) Initial trends in enrolment and completion of massive open online courses. *Int Rev Res Open Distance Learn* 15(1):133–160
- Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Li FF (2014) Large-scale video classification with convolutional neural networks. In: *IEEE conference on computer vision and pattern recognition*, pp 1725–1732
- Kim Y (2014) Convolutional neural networks for sentence classification. [arXiv:1408.5882](https://arxiv.org/abs/1408.5882) [cs]
- Kloft M, Stiehler F, Zheng Z, Pinkwart N (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*, pp 60–65
- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: *International conference on neural information processing systems*, pp 1097–1105
- LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444
- Mrquez-Vera C, Cano A, Romero C, Ventura S (2013) Predicting student failure at school using genetic programming and different data mining approaches with high dimensional and imbalanced data. *Appl Intell* 38(3):315–330
- Mrquez-Vera C, Cano A, Romero C, Noaman AYM, Mousa Fardoun H, Ventura S (2016) Early dropout prediction using data mining: a case study with high school students. *Expert Syst* 33(1):107–124
- Murthy SK (1998) Automatic construction of decision trees from data: a multi-disciplinary survey. *Data Min Knowl Discov* 2(4):345–389
- Ng AY, Jordan MI (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. *Adv Neural Inf Process Syst* 16:841–848
- Onah DFO, Sinclair JE, Boyatt R (2014) Dropout rates of massive open online courses: behavioural patterns. In: *International conference on education and new learning technologies*, pp 5825–5834
- Ramesh A, Goldwasser D, Huang B, Daume III H, Getoor L (2014) Learning latent engagement patterns of students in online courses. In: *Proceedings of the 28th AAAI conference on artificial intelligence*. AAAI Press
- Ruder S (2016) An overview of gradient descent optimization algorithms. [arXiv preprint arXiv:1609.04747](https://arxiv.org/abs/1609.04747)
- Sainath TN, Kingsbury B, Saon G, Soltau H, Mohamed AR, Dahl G, Ramabhadran B (2015) Deep convolutional neural networks for large-scale speech tasks. *Neural Netw* 64:39
- Sinha T, Jermann P, Li N, Dillenbourg P (2014) Your click decides your fate: inferring information processing and attrition behavior from MOOC video clickstream interactions. [arXiv preprint arXiv:1407.7131](https://arxiv.org/abs/1407.7131)
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
- Stein RM, Allione G (2014) Mass attrition: an analysis of drop out from a principles of microeconomics MOOC. *Social Science Research Network*, pp 1–19
- Taylor C, Veeramachaneni K, O'Reilly UM (2014) Likely to stop? Predicting stopout in massive open online courses. [arXiv preprint arXiv:1408.3382](https://arxiv.org/abs/1408.3382)

- Veeramachaneni K, Halawa S, Derroncourt F, O'Reilly UM, Taylor C, Do C (2014) Moomdb: developing standards and systems to support MOOC data science. arXiv preprint [arXiv:1406.2015](https://arxiv.org/abs/1406.2015)
- Wang Y (2013) Exploring possible reasons behind low student retention rates of massive online open courses: a comparative case study from a social cognitive perspective. In: Proceedings of the 1st workshop on massive open online courses at the 16th annual conference on artificial intelligence in education, p 58
- Wang F, Chen L (2016) A nonlinear state space model for identifying at-risk students in open online courses. In: Proceedings of the 9th international conference on educational data mining, pp 527–532
- Wen M, Yang D, Rose C (2014) Sentiment analysis in MOOC discussion forums: what does it tell us? In: Proceedings of educational data mining
- Xing W, Chen X, Stein J, Marcinkowski M (2016) Temporal predication of dropouts in MOOCs: reaching the low hanging fruit through stacking generalization. *Comput Human Behav* 58(Supplement C):119–129
- Yang D, Sinha T, Adamson D, Ros CP (2013) 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 11, p 14
- Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL (2014) Time series classification using multi-channels deep convolutional neural networks. *Web-Age Information Management. Springer, Cham, (Lecture notes in computer science)*, pp 298–310

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.