Tracking and predicting student performance in university

INF7370
UQAM
Winter 2018
By
Fares Benslimane
and
Khalid MOUSTAPHA ASKIA

23 April 2018

Abstract

By continuously tracking students' academic performance and accurately predicting their future performance, we can ensure students graduation and help them succeed their courses. Predicting Student performance faces two challenges, mainly due to the diversity of the students' background and the necessity of continuously tracking the students' evolving progress. In this paper, we develop and compare two approaches: a naive and static approach that does a one-time prediction and uses classical machine learning algorithms like Decision trees, SVM and a sequential approach that enables the progressive prediction of students' performance and which is based on Recurrent neural networks.

1 Introduction

University tuition can be excessively expensive. According to a recent report from Statistics Canada [6], tuition to a Canadian university costs an average of $6,571 for undergraduate programs per year. Students usually require loan plans in order to pay for their tuition. According to Canadian statistics [2], a $2.7 billion was disbursed to approximately 489,000 full-time students in 2014-2015. Meanwhile, recent studies show that in American higher-education system: only 59 percent of four-year college students graduate within six years [1].

To make university affordable and worthwhile, it is important to ensure that most of the students enrolled succeed their programs or even graduate on time. Therefore, early interventions for students who most likely will fail their graduation can help these save both money and time.

A possible solution for making early interventions for students is by building an automatic system that would successfully assess students' early performance and pre-university traits, and accurately predicts their future outcome, such as if they will graduate or not. However, predicting student performance at universities faces two main challenges.

Firstly, The attributes that have been frequently used by researchers to predict students' performance are the cumulative grade point average (CGPA),

internal assessment and students' demographic (gender, age, family background, and disability). It have been shown that they are considered as an indication of realized academic potential [7]. Thus, students can excessively differ in terms of pre-university traits, especially since many students come from different backgrounds.

Secondly, predicting student performance is usually not a one time task. You can not accurately predict a student's academic potential only through his early grades or even through his static demographic traits. Rather, this requires a continuous tracking of the student's performance over a period of time, in order to reveal his true potential. For that reason, the predicting task needs to incorporate not only the historical student accomplishments but also the evolution of his academic progress.

In this paper, we try to address these challenges and we compare both approaches : the temporal approach and the static one-time prediction approach. The proposed temporal method is based on recurrent neural network that treats the task of predicting student performance (whether or not the student will graduate) as a temporal/sequential task.

2 Related work

Many machines learning techniques have been used for the problem of predicting student performance. Many researchers have used Decision trees like (Romero and al 2010), for their simplicity and comprehensibility; Neural networks (wang et al 2002), for their ability to learn complex nonlinear relationships between variables; Naive Bayes (Suljic et al 2008) and K-Nearest Neighbor (Mayilvaganan et al 2014). Most of the works done in this field treat the prediction as a one time task, it ignores the temporal effect that students can improve their knowledge over time. To take the temporal effect into account, (Mihaela et al 2017) [8] use an ensemble learning technique based on the Exponontially Weighted Average Forecaster (EWAF) (Lugosi et al 2006), in order to enable progressive prediction of students performance. Also (Fei et al) [3] use a temporal model based on recurrent neural network (RNN) with long short term memory (LSTM) cells, to predict student dropout in online courses. They argue that since the features are captured continuously for

students, therefore the dropout prediction is essentially a time series prediction problem. Which means that temporal models like RNNs usually works better in such problems than simple machine learning methods like support vector machines or decision trees.

Our approach uses recurrent neural network (RNN), in particular, the long short-term memory (LSTM), to capture the temporal correlation of the student's performance over the academic terms. For that, we prove that by using a continuous approach we can guarantee a much better performance than using a static one-time prediction method.

3 Database description

In this paper we use a database from the University of Quebec at Montreal (UQAM). The data are collected by the Planning and Institutional Research Service and contain information about the students since its creation in 1969.

The raw database contain 288 variables that are multivariate (numerical and categorical) and more than 300.000.000 (tree hundreds millions) records. These variables stores personal information about the student (such as age, address, citizenship etc) and information about the activities and the program in which he is registered. It is important to note that each record is activity based.

4 System model

For a given university program, a student must complete C credits within T terms to graduate. However, students can still graduate after or even before the T terms limit. Let x_i^t denote the student i's academic performance state at term t, which includes the earned credits, grades..etc. Notice that different students can have different sequence lengths. Therefore, our system model needs to accept students with different sequence lengths (number of terms). Let $\hat{y}$ denote the final prediction output (1: graduated or 0: not), $\hat{y}$ will depend on all the performance states of the student across all the previous terms. At which, $\hat{y}_i = f(x_i^1, x_i^2, ..., x_i^t)$ represent the predicted output given the academic states up to term t of student i. Given student i's static background and his evolving academic state, at term t, the model will be

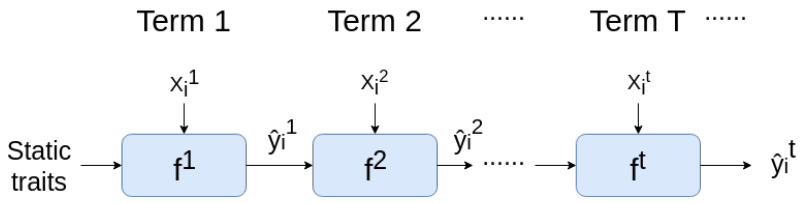

Figure 1: Prediction the evolving academic state of the student

able to predict whether or not he will eventually graduate. Therefore, the predictor will not only use the current performance state of the student but all the previous academic states from all the previous terms. This idea is clearly illustrated in Figure 1.

Let's say that we want to predict if a student A that already finished 4 terms out of 6, whether or not he will graduate. We do the same thing with another student B that has finished only 1 term out of 6. It is reasonable to assume that our model will have a better, more reliable prediction for student A than for student B; Since the input sequence of A has more information than those of B's.

5 Prediction methods

In the interest of showing the advantages of using temporal models on sequential data over using machine learning algorithms that perform one-time prediction; we compare between the two approaches and prove the effectiveness of using temporal models.

5.1 One-time prediction model

A naive approach is to consider the prediction of the student graduation as a one-time prediction task. Where, we combine all the academic states of the student across all terms. Then, we use all the information of the student across all the terms together for prediction. Off-the-shelf machine learning algorithm can then be used to perform the prediction task. We used five straight forward machine learning algorithm: k-nearest neighbors (KNN), Decision tree, Random Forest, Support Vector Machine (SVM) And Neural

Network. The reason behind this choice of algorithms is because they are the most used by researchers for predicting student performance (PSP) problems [7]. We also used an ensemble learning algorithm based on the random forest, the SVM and the decision tree.

5.2 Feature Vector

The feature vector is composed of two types of features. First, static features that include the student's pre-university traits (student's demographic, age, gender, citizenship ..etc). These features don't usually change from a term to another. Second, the student's academic state (dynamic features) that includes courses that he has taken with their grades and the earned credits in every term.

Besides the static features, we construct a dynamic feature vector that assemble all the available courses that students have taken where their values correspond to the course grade. Suppose that the total number of courses is N. Therefore, we end up with a feature vector of size N. (Mihaela et al 2017) [9] used the same technique but ended up with a feature vector of 2N since they are using both the grade and the earned credit of every course. An obvious drawback of this is that the feature vector can grow exponentially with the total number of courses. Besides, the feature vector will be very sparse where there will be many NULL values since the students will choose to take only a small portion of the total available courses. To fix this problem, (Mihaela et al 2017) used educational domain knowledge to cluster the courses, in order to reduce the feature vector size. They also used feature selection techniques to further reduce the feature vector and discover the most relevant features that affect the prediction task. We carry out the same solution where we divide our courses into different clusters (Computer Science (CS), Computer Engineering (CE), Mathematics (MATH)..etc). We only chose the first two, since most of the students only take CS and CE courses.

5.3 Features selection

Because the dataset contains so many attributes (288 variables), it is essential to reduce dimensionality. We use feature selection methods to extract the most important features that are the most correlated with the success of the

•yn_temps_complet_debut_inscription_a	•nb_score_type_programme_a
•nb_cote_r	•nb_trimestre_a
•nb_moyenne_acad_admission_a	•nb_moyenne_acad_programme_a
•de_ville	•informatique
•moyenne_credits_activite_trimestre	•ratio_reussi_activite

Table 1: The 10 features selected

student. We perform two techniques of feature selection: the information gain and the symmetrical uncertainty.

For the information gain, we should first compute the entropy of the data set S : $Entropy(S) = -\sum_{i=1}^{c} p_i \times \log_2(p_i)$ where c is the number of classes (in our case two class : success or failure) and p_i is the percentage of records belonging to the class. Then, the information gain for the variable A is computed by this formula : $Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$ where S_v describe the different values of the variable.

The symmetrical uncertainty also determine the correlation between two variables. We compute it by using the mutual information of two variables X and Y : $I(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) log(\frac{p(x,y)}{p(x)p(y)})$ where $p(x, y)$ is the conjoint probability of x and y and $p(x)$, the probability of x. Knowing the mutal information and the entropy of each variables, the formula of the symmetrical uncertainty U(X,Y) is : $U(X, Y) = 2 \times \frac{I(X,Y)}{Entropie(X)+Entropie(Y)}$

Both of these techniques give us a quantitative value to determine how much a feature is correlated with the output variable. So, we took the 10 most correlated features regarding the both techniques. These ten features are shown in Table 1

5.4 Temporal model

We use a temporal model to capture the temporal correlation of the sequential data in which we'll be able to learn the evolution state of the student across his education period.

5.4.1 Recurrent Neural Network

A recurrent neural network (RNN) handles sequential data. Unlike a normal neural network, an RNN can have inputs and outputs with variant lengths. Also, RNN shares features and information learned across different position of

the data, where in each time step the output is calculated with consideration of both the past information and the current information as shown in Figure 3.

The RNN parameters are trained by the Backpropagation through time (BPTT), where the error between the ground truth and the predicted value is propagated through the different time steps. Although RNN hypothetically can use all the past information learned across the past time steps, in fact, learning to store information over long time intervals takes a very long time because of the decaying error backflow [4].Then, the error is not able to propagate to further past time steps. Which means that the current output is only influenced by close information values. Therefore, RNN usually tends to not capture long-range dependencies.

To address this problem, long short term memory (LSTM) have been designed to solve the above issue. They do so using several gates that control the proportion of the input to give to the memory cell, and the proportion from the previous state to forget [4]. Every LSTM cell uses 3 gates, a memory cell that stores long-range dependencies and another memory cell that works as a replacement for the memory cell.

$$
\begin{aligned}
c^{*<t>} &= tanh(w_c[a^{<t-1>}, x^{<t>}] + b_c) \\
\Gamma_u &= sigmoid(w_u[a^{<t-1>}, x^{<t>}] + b_u) \\
\Gamma_f &= sigmoid(w_f[a^{<t-1>}, x^{<t>}] + b_f) \\
\Gamma_o &= sigmoid(w_o[a^{<t-1>}, x^{<t>}] + b_o) \\
c^{<t>} &= \Gamma_u * c^{*<t>} + \Gamma_f * c^{<t-1>} \\
a^{<t>} &= \Gamma_o * c^{<t>}
\end{aligned}
$$

Where $c^{<t>}$ is the memory cell, $c^{*<t>}$ is the memory cell replacement, $\Gamma_u, \Gamma_f, \Gamma_o$ are the update, forget and output gates respectively. * is the element-wise product.

5.4.2 Proposed method

We propose an architecture based on LSTM that enables the progressive prediction. For that, we only choose dynamic features that change across different terms, which are the average grade of the cluster courses (CS and CE), the earned credits in the term and the average grade of the term.

Data Preparation .
Students have a different number of terms which means that sequences that correspond to students are of different lengths. Sequences should be grouped together by the same length before being fed to the LSTM model. Since LSTM accepts a matrice of sequences, therefore, all sequences must have the same dimensionality. To fix this problem, there are three possible solutions:

1- The sequences can be padded with 0s.
2- We do batches of size 1.
3- We regroup sequences of the same length into batches.

We chose the first solution, where, we take 9 (maximum number of terms) as a static number of terms, as a result, all students will have 9 terms. For example, a student that only completed 6 terms, will have a sequence of 9 terms which 6 of them are filled with his earned grades and credits and the rest of the terms are filled with 0s.

Problem Formulation .
Predicting weather or not the student will graduate is a binary classification problem where the output is a binary label $y \in \{0, 1\}$ indicating whether or not the student will graduate. For a single example in the training set, we optimize the binary cross entropy loss.

$$L(X, y) = -y.log.p(Y = 1|X) - (1 - y).log.p(Y = 0|X)$$

Where $p(Y = i|X)$ is the probability for the assigned label i and $p(Y = 0|X) = (1 - p(Y = 1|X))$.

Model Architecture and Training .
Our architecture is composed of one LSTM network that accepts inputs with a sequence length of 9 and a number of features of 4, and it returns a sequence of values. The output values are then flattened to be fed to a neural

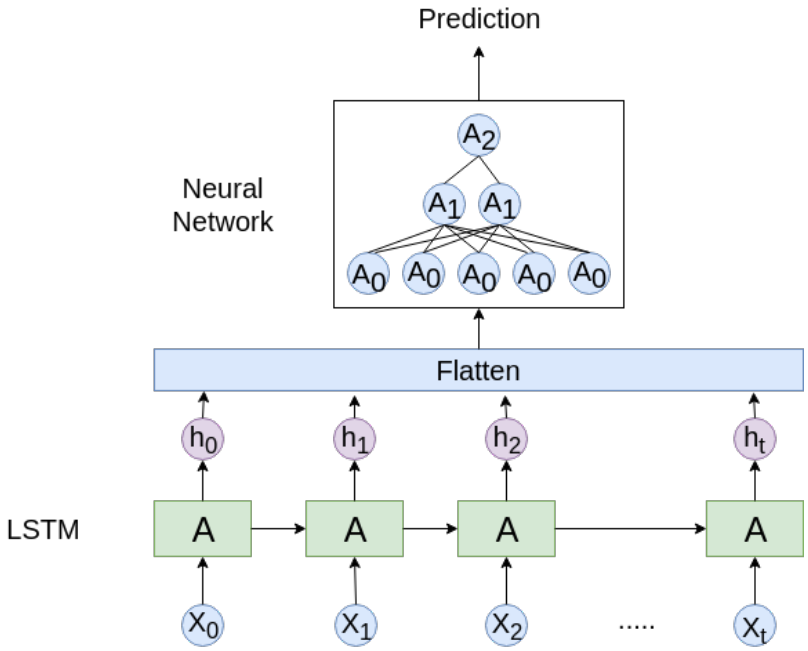

Figure 2: The sequential model that is based on LSTM

network with 3 hidden layers: the first hidden layer has 5 neurones, the second one has two and the last one has only one, they all use the sigmoid activation function. We train the model using mini-batches of size 10 and a total number of iteration of 100. The model is trained end-to-end using Adam (adaptive moment estimation). Adam is an optimization algorithm for stochastic gradient descent that can handle sparse gradients on noisy problems [5]. It is used to update network weights using the training data. It uses 4 hyperparameters: alpha or the learning rate (the proportion that weights are updated, larger values result in larger updates, smaller values result in smaller updates which mean slower convergence. beta1, the exponential decay rate for the first moment estimates. beta2, the exponential decay rate for the second-moment estimates. epsilon, is a very small number to prevent any division by zero in the implementation. We use default parameters that are recommended by the paper (Diederik et al 2014).

Before inputting the data into the network, we normalize based on the mean and standard deviation of the input values. The system block diagram of the proposed architecture is illustrated in figure 2.

6 Experimentation

The data that we used to train and test our algorithms have been extracted from the original dataset. It has 300 anonymized students which are enrolled in the same program of masters in Computer Science. Each student can have one to many terms. wherein every term he can be enrolled in one to many courses, If the student fails a course, he will be assigned a 0. Each student contains static features (student pre-university traits) like his age, sex, address, citizenship...etc and dynamic features which are his courses with their grades and credits.

6.1 Static approach

The static model is designed to use directly classic machine learning algorithms. We used and compared six different algorithms : the K Nearest Neighbor (KNN) with K (number of neighbors) = 8, the decision tree, the random forest with 16 trees, the Suport Vector Machine (SVM) with a degree 2 polynomial kernel, a neural network with 3 layers and an ensemble learning based on the random forest, the SVM and the decision tree.

We perform two kind of validation. A static split of 80% for the training set and 20% for the test set and a cross validation using 10 folds. We use both the F-score metric and the accuracy to evaluate our resulting models. Figures 3 and 4 illustrates the different results of the three algorithms. We can see that the Random Forest is performing the best with both precision and F-measure of 83%.

6.2 Sequential approach

For the sequential model, we split the data to 80% for the training and 20% for the test. Unlike the first static model, in the sequential model, we use only the dynamic features that include the courses with their grades and credits. We get an overall accuracy of 78.72%. Figure 5 shows the prediction accuracy over terms where the accuracy is evolving and improving over the terms.

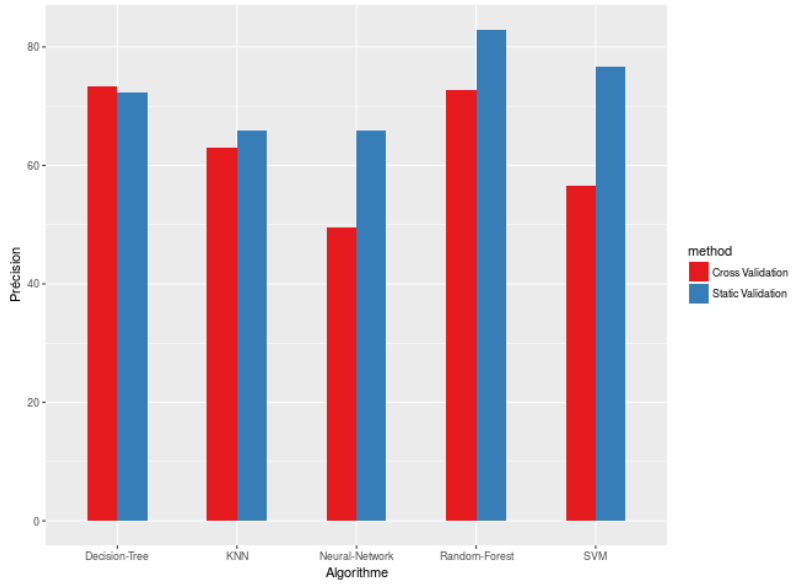

Figure 3: Précision des différents algorithmes

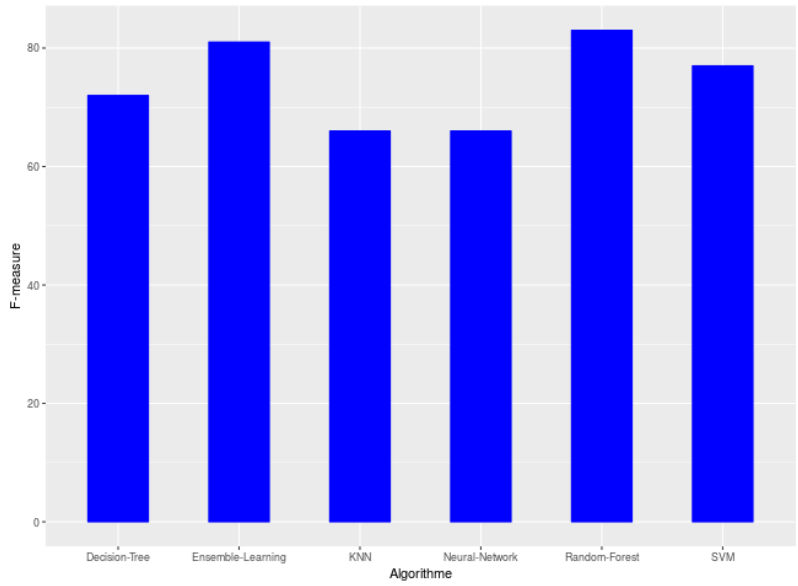

Figure 4: F-measure des différents algorithmes

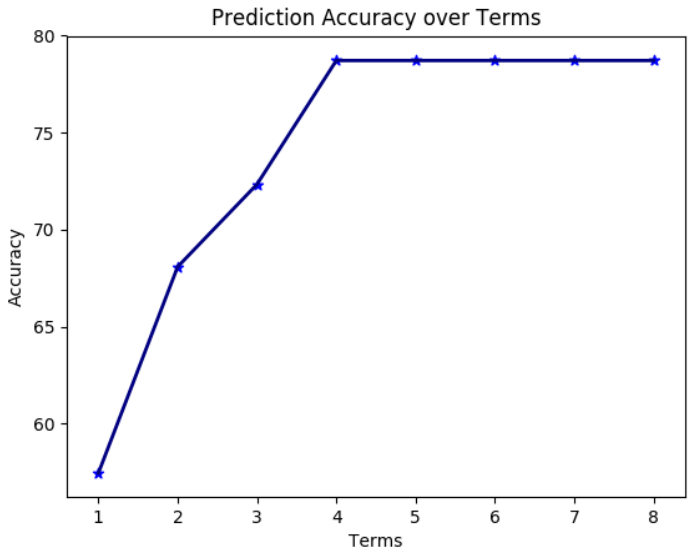

Figure 5: Accuracy of the sequential model

7 Conclusion

In this paper, we propose an architecture that is based on the recurrent neural network that predicts the student's performance whether or not he will graduate given his current academic records. We compared our proposed method with the naive approach of using basic machine learning algorithms that perform one-time prediction. And We prove the effectiveness of using our proposed approach.

References

[1] Preston Cooper. Colleges fail two-thirds of students, 2018.

[2] Employment and Social Development Canada. Canada student loans program statistical review 2014-2015, 2018.

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

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Kathleen Harris · CBC News. University tuition fees in canada jump 3.1% on average, statscan reports, 2018.

[7] Amirah Mohamed Shahiri, Wahidah Husain, et al. A review on predicting student's performance using data mining techniques. *Procedia Computer Science*, 72:414–422, 2015.

[8] Jie Xu, Yuli Han, Daniel Marcu, and Mihaela Van Der Schaar. Progressive prediction of student performance in college programs. In *AAAI*, pages 1604–1610, 2017.

[9] Jie Xu, Kyeong Ho Moon, and Mihaela Van Der Schaar. A machine learning approach for tracking and predicting student performance in degree programs. *IEEE Journal of Selected Topics in Signal Processing*, 11(5):742–753, 2017.