When and How to Update Online Analytical Models for Predicting Students Performance?

Chahrazed Labba, Anne Boyer

When and How to Update Online Analytical Models For Predicting Students Performance?

Chahrazed Labba and Anne Boyer

University of Lorraine, CNRS, LORIA, Nancy, France
{chahrazed.labba,anne.boyer}@loria.fr

Abstract. One of the main concerns in online learning environments is the identification of students with learning difficulties. Conventionally, analytical models trained offline on pre-prepared datasets are used to predict student performance. However, as learning data become progressively available over time, this learning method is no longer sufficient in real-world applications. Nowadays, incremental learning strategies are increasingly applied to update online analytical models by re-training them on newly received data. Various online incremental learning approaches have been proposed to overcome different issues such as catastrophic forgetting and concept drift. However, no approach addresses the question of when to update the model and how to determine whether the new data provide important information that the model should learn. In this paper, we propose a method for determining when an online classifier that predicts student performance and receives a real-time data stream, should be updated. In addition, we use a typical approach that maintains balanced old and new data examples to re-train the model when necessary. As a proof of concept, we applied our method on real data of k-12 learners enrolled in an online physics-chemistry module.

Keywords: Incremental Learning · Distance learning · K-12 learners · Machine Learning · classification

1 Introduction

Learning from anywhere, at any time and at one's own pace has become a reality through the use of e-learning platforms. One of the main concerns in such a context is the high failure rate among the learners. Multiple research works focused on elaborating analytical models to predict students performance. Conventionally, most of these models operate in batch mode by reading and processing the entire training set, with the strong assumption that the data is static and always available in advance. Indeed, the learning data become progressively available over time. It is impossible to collect all relevant training examples at once, and the models must therefore be updated to incorporate the unlearned knowledge encoded in the new data received over time. Thus, the traditional methods of training and evaluating models are no more sufficient in real-world applications.

To address this challenge, incremental learning is increasingly used to ensure continuous adaptation of online analytical models based on newly received data.

The use of online incremental learning has revealed many challenges, including concept drift and catastrophic forgetting. Both of these problems have been widely addressed and many approaches have been proposed [1,8,12] to overcome their impact on the model efficiency. However, none of the incremental learning approaches addresses the question of when to update the model, which in turn raises the question of how to determine whether the newly received data provides relevant information that the model should learn. Addressing the frequency of updating an online model is of high importance. In the distance education, each student has his own pace to learn, which results in variations in the students engagement, regularity and reactivity. There are periods during the school year when most students are active, while the rest of the time only a few of them use the e-learning platform continuously. This variation in the learning behavior has an impact on the quantity and the quality of the generated data over time. According to the existing definitions [4,5,13], incremental learning is a dynamic strategy that consists in processing the stream data as soon as it becomes available due to limited memory resources. This method can lead to frequent and unnecessary updates of the models.

In this paper, we propose an incremental learning process for determining when an online classifier that predicts student performance and receives a real-time data stream, should be updated. Our process invokes the retraining process: i) when new classes are detected in the newly received data ; ii) when the forgetting value in each detected class is below a certain threshold and iii) when a class label is seen but never predicted. The forgetting value within a class is the difference between the two accuracy values over two successive time intervals. To overcome the problems related to concept drift and catastrophic forgetting, our process uses a typical approach that consists in maintaining a balanced training set of old and new data to train the model when necessary. An algorithm is proposed to update the exemplar set continuously as long as the data is generated to re-train the model when necessary. As a proof of concept, we used a real-world scenario of k-12 learners adopting 100% online education. Our process is applied with an Artificial Neural Network (ANN) to predict students at risk of failure.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 introduces the proposed incremental learning process. In Section 4 and Section 5, we present respectively the case study description and the experimental results. The Section 6 presents the conclusions and the future works.

2 Related Work

Online incremental learning is an Artificial intelligence (AI) technique that refers to the circumstance of a permanent online adaptation of the analytical model according to the constantly received data flow over time [4,5,13]. This technique has been used to fulfill adequately various learning analytics objectives among which predicting students performance [1, 6, 7], image classification [10, 14] and text classification [11]. Most of the existing works focus either on solving the

problem of catastrophic forgetting and concept drift, or on comparing incremental online algorithms. When it comes to predicting student performance incrementally, most of the research is oriented towards the comparison of incremental algorithms. In [7],the authors compared four classifiers that can run incrementally. The aim is to recommend the suitable algorithm to use in assessing students performance within an incremental learning context. In [1], the authors compared three approaches of incremental learning to determine the suitable way to handle students stream data. The used approaches include instance-based, batch-based and ensembling of instance-based incremental learning. In [6], the authors proposed an incremental learning technique that combines an incremental version of Naive Bayes, the 1-NN and the WINNOW algorithms. The aim is to predict the student's performance within a distance education environment by using incremental ensemble based on a voting methodology.

The use of incremental learning is more developed, especially for image classification. In [5], the authors proposed an incremental learning framework to overcome the problem of catastrophic forgetting when learning new classes and the problem of data distribution over time referred as concept drift. The framework was tested to classify images using the CIFAR-100 and ImageNet-1000 datasets. In [12] the authors presented a novel framework that can incrementally learn to identify various chest abnormalities by using few training data. the framework is based on an incremental learning loss function that infers Bayesian theory to recognize structural and semantic inter-dependencies between incrementally learned knowledge representations. In [8], the authors compared eight popular incremental methods representing different algorithm classes using stationary and non-stationary datasets. A set of metrics including the accuracy, the robustness and the error classification rate are used to assess the algorithms.

Existing incremental learning methods address a variety of issues, such as catastrophic forgetting, but none address the issue of when to update a model. In this paper, we propose a new incremental learning method that considers the optimal time to update a model by reducing the number of unnecessary updates while maintaining good performance stability.

3 Proposed Approach for incremental Learning

This section starts with a formal introduction to the problem of when and how to update an online analytical model using stream data (Section 3.1). Then it presents an overview of the proposed incremental learning process (Section 3.2).

3.1 Problem Formalization

Online incremental learning [8] is a subset of incremental learning, which is further constrained by runtime and the ability to provide lifelong learning with limited data when compared to offline learning. In general, these constraints are related to real-world applications in which new data is generated sequentially over time, thereby contradicting the strong assumption of total data availability.

Assume $\mathcal{M}_1$, $\mathcal{M}_2$,.., $\mathcal{M}_k$ a sequence of models that is computed on stream data (D_1, Y_1), (D_2, Y_2),.., (D_k, Y_k) as shown in the Figure 1.

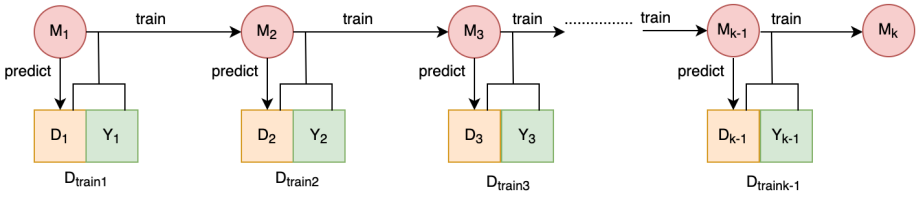

Fig. 1. An Incremental Online Scenario

Each D_i represents a block of new data ($x_i | i \in \{1, m\}$), which it has at least one element and no more than m elements. Usually, the size of the block is limited due to memory constraints [5]. Each Y_i represents the set of true labels. $D_{traini} = (D_i, Y_i)$ represents the data used to update the model $\mathcal{M}_i$ to create the model $\mathcal{M}_{i+1}$ that will be used to predict D_{i+1}.

As shown in Fig.1, the principle of incremental learning is that each time new data is available, the model update is invoked. This, however, may necessitate frequent updates of the online model, which is both time and resource intensive. Defining *when to invoke the re-train* is of high importance. If the model $\mathcal{M}_i$ can accurately predict D_i, there is no need to update it; it can still be used to process the data D_{i+1}. In other words, if the model maintains a certain level of performance stability, it means that the new data does not contain any new knowledge that the model is unable to handle.

Further, one of the difficult challenges in incremental learning is catastrophic forgetting. Suppose the model $\mathcal{M}_i$ is trained on j classes and we invoke its train on $D_{traini} = (D_i, Y_i)$ that contains p new classes. In theory, the model can predict all classes well (j+p), but in practice, the model's stability on the old j classes decreases significantly due to a lack of representation of these classes when training on new ones.

In this work, we propose a new online incremental learning process that aims to reduce the frequency of updating a model while maintaining a certain performance stability over time. To address the problems related to catastrophic forgetting, our method uses a typical approach [3, 5, 9] that consists in maintaining a balanced training set of old and new data to train the model when necessary.

3.2 Learning from a Train Exemplar

To address the common issues of incremental learning (e.g catastrophic forgetting), we adopt a common approach [3, 5, 9] that has been widely applied: we use a small exemplar of both old and new data. In our work, the exemplar set

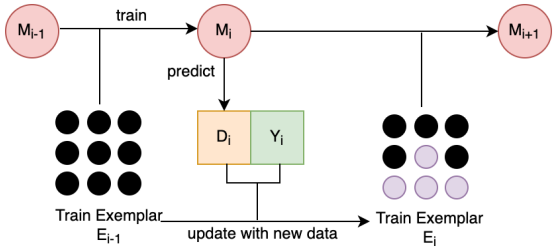

Fig. 2. Exemplar Set

is updated each time new data are received. The new data may contain both new data classes and new observations of old classes. As shown in Fig.2, we no longer use only newly received data to train the model; instead, we use an exemplar set created previously and updated it with new samples from the received stream data. Exemplar samples are selected at random, but with each update, we seek to preserve an equal representation of all learned data classes. Algorithm 1 depicts the entire process of updating the exemplar set.

The Algorithm 1 takes as input the exemplar to update $\mathcal{E}$, the received data (D, Y), the number of samples m to store in $\mathcal{E}$ and the ratio of the classes R. This ratio defines the representation of the classes within the exemplar (e.g if we have 3 class labels, R is equal to 1/3). The Algorithm provides as output an updated version of the exemplar $\mathcal{E}$.

It all starts with cleaning up the old exemplar $\mathcal{E}$ (Line 1 - Line 8). The algorithm checks the number of samples in $\mathcal{E}$ for each old class (Line 3). If this number is strictly greater than the new representation ratio, the algorithm removes the extra samples at random in order to meet the class representation condition (Line 4). Otherwise, there is no need to delete the old observations (Line 6).

The next step is to update the exemplar with the new data. We distinguish between two types of updates: i) the exemplar is updated with the new detected class labels (if any) (Line 9 - Line 16), and ii) the exemplar is updated with the new observations for the old class labels (Line 17 - Line 25). For the first kind of update, the algorithm first determines whether the number of samples in the new data exceeds the allowed representation ratio (Line 10 - Line 11). If this is the case, the algorithm selects samples at random to store in $\mathcal{E}$. The number of selected samples must meet the representation condition. Else, all samples are kept in $\mathcal{E}$ (Line 14). For the second type of update the algorithm checks, for each class, whether the number of samples for old observations meets the representation ratio condition (Line 18 - Line 19). If this is the case, all of the old data that have new observations are updated (Line 20). Otherwise, the algorithm updates the old data with new observations (Line 22). Then, it selects samples at random from the new observations of old classes to store in $\mathcal{E}$ while satisfying the representation condition (Line 23).

Algorithm 1 Build the exemplar set

Require: $\mathcal{E}, (D, Y), m, R$
Ensure: $\mathcal{E}$
1: $(D_{old}, Y_{old}) \leftarrow get\_old\_class(\mathcal{E})$
2: **for** $(c \in Y_{old})$ **do**
3: **if** $(|D_{old_c}| > (m * R))$ **then**
4: $\mathcal{E} \leftarrow remove\_extra\_observations(\mathcal{E}, (D_{old_c}, c))$
5: **else**
6: No need to remove
7: **end if**
8: **end for**
9: $(D_{new}, Y_{new}) \leftarrow get\_new\_data((D, Y))$
10: **for** $(c \in Y_{new})$ **do**
11: **if** $(|D_{new_c}| > (m * R))$ **then**
12: $\mathcal{E} \leftarrow put(\mathcal{E}, select\_random((D_{new}, Y_{new}), R, m))$
13: **else**
14: $\mathcal{E} \leftarrow put(\mathcal{E}, (D_{new_i}, c))$
15: **end if**
16: **end for**
17: $(D_{new_{obs}}, Y_{old_{obs}}) \leftarrow get\_new\_observations\_for\_old\_class((D, Y))$
18: **for** $(c \in Y_{old_{obs}})$ **do**
19: **if** $(|D_{new_{obs_c}}| == (m * R))$ **then**
20: $\mathcal{E} \leftarrow update(\mathcal{E}, D_{old_c}, D_{new_{obs_c}})$
21: **else**
22: $\mathcal{E} \leftarrow update(\mathcal{E}, D_{old_c}, D_{new_{obs_c}})$
23: $\mathcal{E} \leftarrow put(\mathcal{E}, select\_random(D_{new_{obs_c}}, Y_{old_{obs}}))$
24: **end if**
25: **end for**

In the next section, we present how the use of the exemplar fits into the overall incremental learning process.

3.3 Incremental Learning Process to Update an Online Model

Our incremental learning process (Algorithm 2) takes as input: i) the stream data $D = (D_1, .., D_n)$ as it arrives over time; ii) the true label $Y = (Y_1, .., Y_n)$ associated to the stream data [1]; iii) the ML model ($\mathcal{M}$) , iv) the allowed forgetting value ($\mathcal{F}$) and v) m the number of samples to store in the exemplar trainset.

The Algorithm 2 starts by iterating over the prediction times (Line 1). If the prediction time corresponds to the beginning of the time interval (Line 2), we train the model on the received data during that time (Line 3). This first moment corresponds to the beginning of the school year, when all students are given a class label. Indeed, to overcome the cold start problem, students can be

[1] e.g Y_1 represents the set of true labels for the stream data D_1

considered all successful, all at risk of failure, or their historical information can be used to assign them to a specific class among the predefined ones.

Algorithm 2 Incremental Learning Process

Require: $\mathcal{D} = ((D_1, Y_1), .., (D_n, Y_n)), \mathcal{M}, \mathcal{F}, m$

1: **for** i in $(1..n)$ **do**
2: **if** $(i == 1)$ **then**
3: $\mathcal{M} \leftarrow$ **fit**$(\mathcal{M}, (D_i, Y_i))$
4: $\mathcal{C} \leftarrow$ **get-seen-class**(Y_i)
5: $\mathcal{E}_i \leftarrow$ **build-trainset**$((D_i, Y_i), m, 1/|\mathcal{C}|)$
6: $A_{last} \leftarrow \emptyset$
7: **else**
8: $c \leftarrow |\mathcal{C}|$
9: **List_preds** $\leftarrow$ **predict** $(\mathcal{M}, D_i)$
10: $A_i \leftarrow$ **Score-Accuracy**$(List\_preds, Y_i)$
11: $\mathcal{C}_i \leftarrow$ **get_current_detected_class**(Y_i)
12: $\mathcal{C} \leftarrow$ **unique_class** $(\mathcal{C} \cup \mathcal{C}_i)$
13: $\mathcal{E}_i \leftarrow$ **update Exemplar set**$(\mathcal{E}_{i-1}, (D_i, Y_i), m, 1/|\mathcal{C}|)$
14: **if** $(|\mathcal{C}| > c)$ **then**
15: $\mathcal{M} \leftarrow$ **fit**$(\mathcal{M}, \mathcal{E}_i)$
16: **else**
17: $OK \leftarrow$ **true** , $j \leftarrow 0$
18: **while** (**OK and** $j < |\mathcal{C}|$) **do**
19: **if** $(A_{ij} == 0)$ **then**
20: $\mathcal{M} \leftarrow$ **fit**$(\mathcal{M}, \mathcal{E}_i)$
21: $OK \leftarrow$ **false**
22: **else if** $(A_{last_j} > A_{ij})$ **then**
23: $a_j \leftarrow$ **compute_forget** (A_{last_j}, A_{ij})
24: **if** $(a_j > \mathcal{F})$ **then**
25: $\mathcal{M} \leftarrow$ **fit**$(\mathcal{M}, \mathcal{E}_i)$
26: $OK \leftarrow$ **false**
27: **end if**
28: **end if**
29: $j \leftarrow j + 1$
30: $A_{last} \leftarrow A_i$
31: **end while**
32: **end if**
33: **end if**
34: **end for**

Then (line 4), we recuperate the learned classes during the first training time. Later, we build the first trainset (Line 5). The *build-trainset* function takes as parameters the received data (D_1), the true labels (Y_1) , the number of samples to store (m) and the ratio of each learned class $(1/|\mathcal{C}|)$. The samples are selected randomly, but the learned classes are equally represented in order to address the issue of under-represented classes. The algorithm uses the list A_{last} to save the accuracy values of the model for the most recent prediction time (Line 6). For

the following intervals, the algorithm starts by saving the number of the classes already seen (Line 8). Then, the last calculated model is used to predict the classification of the received stream data (D_i) (Line 9). Then it calculates the current accuracy scores for the seen class labels (Line 10). C_i that corresponds to the list of class labels detected in Y_i is identified (Line 11) and the set of seen classes $\mathcal{C}$ is updated (Line 12). Later, the exemplar train-set is updated using the Algorithm 1 (Line 13). The train-set is updated each time new data is received, regardless of whether or not a model is updated. The aim is to maintain an up-to-date train-set that will serve to train the model when necessary. There are three cases to start model training: If new classes are detected in the labeled new received data (Line 14), the model's train is invoked using the recently updated exemplar set ($\mathcal{E}_i$). If no new classes are found, the model's accuracy per class is checked: if the current accuracy score equals zero than the training is invoked (Line 19 - line 21). Else, the accuracy is compared to that computed during the last prediction time to see if it has improved or decreased(Line 22). If the second case, the algorithm verifies if the the forgetting value within the learned classes does not exceed a given threshold ($\mathcal{F}$) (Line 23 - Line 24). If it is so, the model's train is invoked (Line 25). It is sufficient to detect a drop in accuracy in one class to start the training phase.

4 Case Study

The Cned[2] offers a diverse range of courses entirely online to k-12 students located all over the world (173 countries). These students come from a variety of demographic backgrounds and are unable to attend regular schools for a variety of reasons. The Cned offers the courses through a Learning Management System (LMS) and provides with it a set of applications such as an education management system that allows administrative tracking of the students. Our case study in this work consists of K-12 students enrolled in the physics-chemistry course during the 2017-2018 school year. There are 46 weeks in the school year and 671 enrolled students.

To predict students performances on weekly-basis, the problem is formalized as a n-classification problem. The classification consists of three classes: high risk ($<= 8$), medium risk (< 8 and $<= 12$) and success (> 12). On each week w_i, a student is defined by a tuple $X = (f_1, .., f_m, y)$ where $f_1, .., f_m$ are the features and y is the class label. The student class may vary from one week to another based on his/her performance. The selected features are extracted from the two data sources including the LMS (moodle) and the education management system (GAEL).

We distinguish the following indicators [2] calculated based on the used features:

- Demographic data: it represents information such as the gender, the age, has or not a scholarship, and repeating or not the year. These data are provided by the education management system.

[2] Centre national d'enseignement à distance: https://www.cned.fr

- Performance: this indicator denotes the submitted exams and the grades.
- Engagement: it described the learner activity on the LMS. The only way to track learners engagement is through their interaction with the LMS content.
- Regularity: it denotes the progress made by the learner in terms of achieved LMS activities and the number of submitted exams through GAEL.
- Reactivity: It is denoted by the time taken to submit an exam as well as the time between successive connections to the LMS.

The aim is to predict students at risk of failure as early as possible while taking into account the progressive availability of data over time. To address the issue of a cold start, all students are classified as having a high risk of failing during the first week. This classification will evolve over time based on the students performance.

5 Experiments

As a proof of concept, the incremental learning process was tested with the ANN model. Prior to the assessment, a set of experiments were performed to determine the suitable parameters for our model, including defining the optimizer (SGD) and the learning rate (0.01). Several configurations were used to evaluate the effect of process parameters on the number of model updates as well as its accuracy. Furthermore, to demonstrate our process's efficiency in reducing the number of model updates while maintaining good performance stability, we compared it to an incremental process that has full access to all previous data and is trained each week. The second process is ideal for an incremental model because all data is available and training is performed on a weekly basis.

5.1 Impact of the Forgetting Value and Exemplar set Size

Our incremental learning process is based on two key parameters including the exemplar set size and the forgetting value (see Section 3). The first specifies the number of the samples to be used when re-training the model. While the second shows the rate of forgetting we can tolerate per class label.

Various configurations were used to test the proposed incremental learning process (see Table 1). Each configuration differs in the size of the exemplar set and the forgetting value. Overall, three exemplar sizes (80, 100 and 150) were used, each with ten forgetting values (from 1% till 10%).

Table 1. Configurations

Exemplar set size	Forgetting Value
80	1%, 2%, 3%,4%, 5%, 6%, 7%, 8%, 9%, 10%
100	
150	

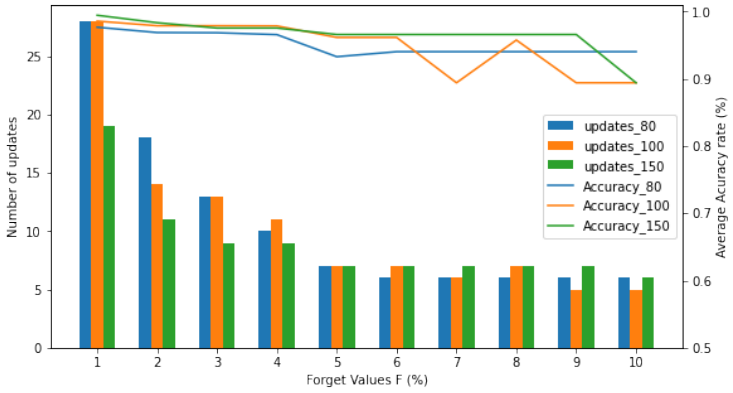

Fig. 3. Number of updates and Average accuracy Per exemplar Size and Forgetting Value

The Fig 3 depicts the variation in the number of model updates as well as the average of accuracy as a function of exemplar size and forgetting value. The weekly accuracy values are used to calculate the average accuracy (over a period of 46 weeks).

Regardless the exemplar size, we notice, in overall, that the number of updates decreases while the forgetting value increases. This is to be expected, as increasing forgetting values give the model a lot more space to forget what it has learned. While a minor forgetting value may result in frequent updates. As shown in Fig 3, for allowed values of 1%, we find the highest number of updates (28, 28, and 19 updates respectively for exemplar sizes 80, 100 and 150). While for a value of 10%, we notice the smallest number of updates (6, 5 and 6 updates respectively for exemplar sizes 80, 100 and 150).

The average accuracy associated with the lowest forgetting values and thus frequent updates is, indeed, the highest. However, for fewer updates, the average accuracy remains high ($\geq 90\%$), even though it gradually decreases as the forgetting value increases.

Despite the decrease in the number of updates as the forgetting value increases, the model has maintained good stability, which can be attributed to the use of an updated exemplar set. As explained in Section 3, the exemplar set is used to store observations for old and new classes over time. Furthermore, when creating this exemplar, we consider an equal representation of all classes to allow the model to learn the knowledge gained over time more effectively. Equal class representation is considered, since the received data over time already present imbalances with respect to the "medium risk failure" class. Consequently, with a non-equal representation this class is not well detected, especially when the samples are selected randomly when building the exemplar set. The number of samples in the exemplar influences both the number of updates and the average accuracy. Increasing this number does not always ensure the smallest number

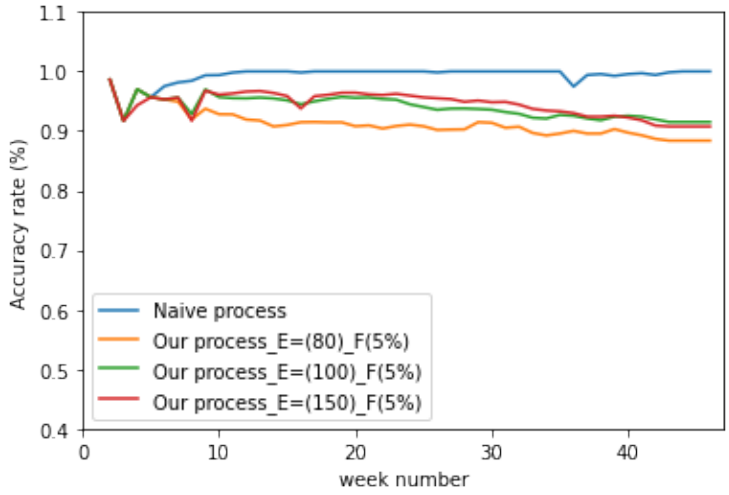

Fig. 4. Accuracy over weeks

of updates and the highest average accuracy. For example, using an exemplar set of size 100, for half of the time resulted in an equal or higher number of updates than using an exemplar set of size 80. Furthermore, it demonstrated a high variation in average accuracy when compared to the rest, even though this variation was not significant. While, in overall, the use of the exemplar set with a size 150 samples resulted in less number of updates and better average accuracy. Furthermore, for each exemplar set size, we observe that the number of updates is stable or only slightly varies on an interval of forgetting values for each exemplar set size. For example, for the exemplar set with a size 80, on the interval [6%, 10%], the average accuracy is stable, and the number of updates is equal to six. This can be explained by the fact that most of the detected forgetting values were less than 6%, requiring no model update. Thus, in this case, the number of updates is mostly identified when a new class is detected or when the accuracy of a given class equals zero.

In summary, the forgetting value and the size of the exemplar set are relevant parameters for reducing the number of updates and increasing the stability of the model performance in the context of incremental learning. The goal of this article is not to identify and fix these parameters, but rather to demonstrate how they can be incorporated into a full incremental learning process to reduce unnecessary updates while maintaining good stability.

5.2 Assess our Proposal to an Incremental Process with Full Data Access

In this experiment, we compared the efficiency of our proposal to an incremental learning process that has full access to old data and trains the model weekly. The

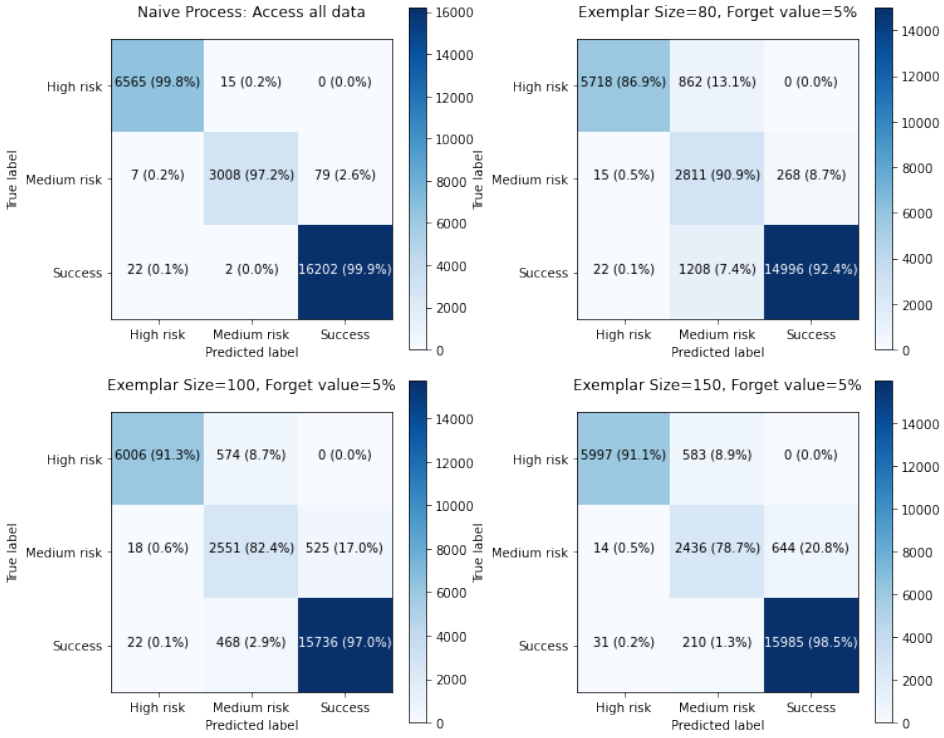

Fig. 5. confusion matrix

second procedure does not take into account the use of the forgetting value and the exemplar set for training. The model update is invoked 46 times (over the 46 weeks). For the rest of the paper, this second process is denoted as the naive process. For this experiment, we consider the results of the model trained with a forgetting value of 5% for each of the exemplar set sizes (80, 100, 150). The Fig 4 presents the evolution of the accuracy of the four ANN models over the weeks. The model with the highest accuracy values over time is the one that was trained weekly using a process that has full access to all of the data. However, the rest of the models, which were only trained 7 times over the 46 weeks using our incremental learning process, were able to maintain high accuracy values of 90% or higher.

The overall accuracy does not reflect the actual performance of a classifier. Thus, in the Fig 5, we present the confusion matrix of the four models over all weeks. The model trained using the naive incremental process is represented by the first confusion matrix. It has the highest accuracy values across all classes, and we used it as a reference to evaluate the efficacy of our incremental learning process when training the rest of the models with different exemplar set sizes.

Indeed, with our incremental process, we find that increasing the sample size does not always improve the model's accuracy across all classes. When detecting

the medium risk class, training the model with 80 samples outperforms training it with larger numbers of samples (100, 150). Indeed, this could be a result of the sample selection strategy used when creating the exemplar set. When it comes to class representation, the data distribution is not homogeneous during the first few weeks. As a result, the total number of samples determined by the fixed rate cannot always be guaranteed (e.g. 30 % of the number of samples should be in the medium-risk category or only 10% are available). However, with a smaller number of samples we can reach the full proportions of the different classes more quickly than by using a larger number of samples. The rapidity is addressed in terms of the number of the week at which we begin to have a complete representation of all classes of students in the selected samples with respect to the predefined rate for each class. We believe it is important to determine the appropriate threshold that should be used as the size of the exemplar set. Since the goal of our experiments is to detect students in difficulty (high and medium risk), we can say that for a fixed forgetting value (5%), the appropriate size of the example set is 80. Indeed, high-risk students have the lowest accuracy value when compared to the rest (100, 150), but students who are not well detected are classified as medium risk. As a result, they will be notified in both cases. Furthermore, with 80 as the exemplar size, the proportion of students who are actually at medium risk and were classified as successful is low (only 8.7%), compared to the rest (100: 17%, 150: 20.8%).

6 Conclusion

In this paper, we addressed the question of when to update online analytical models and how to determine whether the new data provide important information that the model should learn. We proposed an incremental learning process that determines when an online classifier that predicts student performance and receives a real-time data stream, should be updated. Our method invokes the retraining process: i) when new classes are detected in the newly received data ; ii) when the forgetting value in each detected class is below a certain threshold and iii) when a class label is seen but never predicted. In addition, we use a typical approach that maintains balanced old and new data examples to re-train the model when necessary. As a proof of concept, we applied our method on real data of k-12 learners enrolled in an online physics-chemistry module. The experimental results show that the forgetting value and the size of the exemplar set are relevant parameters for reducing the number of updates and maintaining the stability of the model performance in the context of incremental learning. Further, we found that increasing the exemplar set size does not always improve the classifier's accuracy across all the classes. Both parameters can be set based on the requirements and the desired outcome.

The current work presents some limitations that we tried to mitigate when possible: i) currently, the proposed incremental process has been evaluated using only the ANN, as the method, rather than the model, makes the most significant contribution and ii) we defined fixed rates for the samples representing each of

the class labels when creating the exemplar set for training. This representation, however, cannot always be insured because the number of samples available may be less than what is required.

In the future, we plan to compare the use of our incremental learning process with other classifiers, such as the random forest. Furthermore, we are interested in improving the process of building the exemplar set, particularly as it's currently based on a random selection of samples.

References

1. Ade, R., Deshmukh, P.: Instance-based vs batch-based incremental learning approach for students classification. International Journal of Computer Applications **106**(3) (2014)
2. Ben Soussia, A., Roussanaly, A., Boyer, A.: An in-depth methodology to predict at-risk learners. In: European Conference on Technology Enhanced Learning. pp. 193–206. Springer (2021)
3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the European conference on computer vision (ECCV). pp. 233–248 (2018)
4. Gepperth, A., Hammer, B.: Incremental learning algorithms and applications. In: European symposium on artificial neural networks (ESANN) (2016)
5. He, J., Mao, R., Shao, Z., Zhu, F.: Incremental learning in online scenario. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13926–13935 (2020)
6. Kotsiantis, S., Patriarcheas, K., Xenos, M.: A combinational incremental ensemble of classifiers as a technique for predicting students' performance in distance education. Knowledge-Based Systems **23**(6), 529–535 (2010)
7. Kulkarni, P., Ade, R.: Prediction of student's performance based on incremental learning. International Journal of Computer Applications **99**(14), 10–16 (2014)
8. Losing, V., Hammer, B., Wersing, H.: Incremental on-line learning: A review and comparison of state of the art algorithms. Neurocomputing **275**, 1261–1274 (2018)
9. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020)
10. Ristin, M., Guillaumin, M., Gall, J., Van Gool, L.: Incremental learning of random forests for large-scale image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(3), 490–503 (2016). https://doi.org/10.1109/TPAMI.2015.2459678
11. Shan, G., Xu, S., Yang, L., Jia, S., Xiang, Y.: Learn#: A novel incremental learning method for text classification. Expert Systems with Applications **147**, 113198 (2020)
12. Sirshar, M., Hassan, T., Akram, M.U., Khan, S.A.: An incremental learning approach to automatically recognize pulmonary diseases from the multi-vendor chest radiographs. Computers in Biology and Medicine **134**, 104435 (2021)
13. Yang, Q., Gu, Y., Wu, D.: Survey of incremental learning. In: 2019 chinese control and decision conference (ccdc). pp. 399–404. IEEE (2019)
14. Zhao, H., Wang, H., Fu, Y., Wu, F., Li, X.: Memory efficient class-incremental learning for image classification. IEEE Transactions on Neural Networks and Learning Systems (2021)