Cost-Sensitive Perceptron Decision Trees for Imbalanced Drifting Data Streams

Bartosz Krawczyk[1] and Przemysław Skryjomski[1]

Department of Computer Science
Virginia Commonwealth University
Richmond, VA 23284, USA
bkrawczyk@vcu.edu

Abstract. Mining streaming and drifting data is among the most popular contemporary applications of machine learning methods. Due to the potentially unbounded number of instances arriving rapidly, evolving concepts and limitations imposed on utilized computational resources, there is a need to develop efficient and adaptive algorithms that can handle such problems. These learning difficulties can be further augmented by appearance of skewed distributions during the stream progress. Class imbalance in non-stationary scenarios is highly challenging, as not only imbalance ratio may change over time, but also relationships among classes. In this paper we propose an efficient and fast cost-sensitive decision tree learning scheme for handling online class imbalance. In each leaf of the tree we train a perceptron with output adaptation to compensate for skewed class distributions, while McDiarmid's bound is used for controlling the splitting attribute selection. The cost matrix automatically adapts itself to the current imbalance ratio in the stream, allowing for a smooth compensation of evolving class relationships. Furthermore, we analyze characteristics of minority class instances and incorporate this information during the model update process. It allows our classifier to focus on most difficult instances, while a sliding window keeps track of changes in class structures. Experimental analysis carried out on a number of binary and multi-class imbalanced data streams indicate the usefulness of the proposed approach.

Keywords: machine learning, data streams, imbalanced data, concept drift, online learning, multi-class imbalance

1 Introduction

Modern machine learning systems must take into account the phenomenon of data in motion, a scenario in which instances arrive rapidly and continuously. This has given birth to the notion of data streams, potentially unbounded and ordered data collections. They impose new challenges on learning systems, due to their ever-growing size, speed of incoming instances and difficulties or latencies for obtaining true class labels. Additionally, data characteristics may change over time, leading to a phenomenon known as concept drift [2]. When dealing

with non-stationary data, the previously trained classifier may lose its competence over time, as the new concepts are vastly different from the previously seen ones. By using only incremental learning we will accommodate new instances, but do not take into account the changes in the relevance of older cases. Therefore, mining data streams require methods that are able to detect the potential presence of drift in order to reset the learning model, or smoothly adapt to incoming data with properly tuned forgetting mechanism. This is built on the top of need for high responsiveness, low computational resource consumption and highly limited data storage.

The discussed scenario becomes even more complex when we consider the potential presence of difficulty known as class imbalance [4]. Skewed distributions pose significant challenge for classifiers, leading to their bias towards the majority class. At the same time in many real-life applications minority instances are usually of higher significance. While learning from imbalanced data has gained broad attention from the research community over last two decades, online class imbalance is still an emerging topic [12]. Here the imbalance ratio may change dynamically with the stream progress. Furthermore, the concept drift may lead to changes in class relationships, allowing for minority class to become the majority one and vice versa. While most works in online imbalance consider binary cases, one must be aware that new classes may appear over time and old ones disappear, causing further complications for the learning process. As the number of classes may change over time, so may their mutual relationships. Multi-class imbalance, while difficult on itself in static scenarios, is an important direction to be addressed in order to obtain robust data stream classifiers. Additionally, we need proper performance metrics that can take into account streaming and multi-class skewed nature of analyzed data.

In this paper we propose a novel decision tree learning approach for handling imbalanced and drifting data streams. As a base for our model, we use fast perceptron trees and improve them to become skew-insensitive by using a moving threshold solution. It aims at re-balancing the supports for each class during the decision making step, thus alleviating the skew bias with almost no additional computational cost. This is achieved by weighting support functions for each class according to a specified cost function. In our solution the cost matrix evolves over time and adapts to the current state of the stream. This allows us to propose an adaptive cost-sensitive solution that is able to learn from both binary and multi-class imbalanced data streams. We augment it with drift detection and use McDiarmids bound for controlling the splitting attribute selection. Additionally, we show how to analyze the structure of minority classes in an online manner by using a sliding window. This allows us to estimate the difficulty of incoming minority class instances, giving an additional insight into the current state of the stream. We propose an efficient method of incorporating this background information into the update process of the proposed decision tree in order to better capture the minority class characteristics. Experimental study carried out on a number of drifting and imbalanced binary and multi-class data streams shows the usefulness of the proposed learning algorithm.

2 Learning from imbalanced data streams

In this section, we will discuss the necessary background for this paper. This includes the area of mining drifting data streams, the problem of skewed class distributions and online learning in the presence of class imbalance.

2.1 Data stream mining

Let us define data stream as an ordered sequence of instances that arrive over time and can be of unbounded size. This leads to a set of learning characteristics specific to this problem that must be accounted for when designing data stream mining algorithms. Here, we do not have a predefined training set, but the instances become available at various time intervals sequentially with the stream progress. Additionally, due to the unknown and potentially massive size of the stream we cannot store it in memory and must use each instance a limited number of times before discarding it to limit the computational resources being used. Furthermore, characteristics of the stream are subject to change and we must accommodate this fact during the continuous learning process [2].

We will assume that data stream is composed of a set of states $\mathbf{S} = \{\mathrm{S}_1, \mathrm{S}_2, \cdots, \mathrm{S}_n\}$, where given state S_i comes from a distribution D_i. We may deal with online case (each state is a separate instance) or chunk case (each state is a set of instances). The simplest learning scenario is a stationary data stream, where transition between states $\mathrm{S}_j \rightarrow \mathrm{S}_{j+1}$ holds $\mathrm{D}_j = \mathrm{D}_{j+1}$. In most real-life scenarios the stream characteristics evolve over time, leading to the notion of non-stationary data stream and concept drift. It may affect various aspects of incoming data, thus leading to a number of views on the discussed phenomenon. From the point of view of influencing the existing decision boundaries, we may distinguish real and virtual concept drifts. The former has effect on posterior probabilities and may impact unconditional probability density functions. This forces the learning system to adapt to change in order not to lose the competence. The latter drift does not have any effect on posterior probabilities, but only on conditional probability density functions. It may still cause difficulties for the learning system, leading to false alarms and unnecessary computational expenses on rebuilding the classifier. Another view on concept drift comes form the severity of ongoing changes. Sudden concept drift appears when S_j is being suddenly replaced by S_{j+1}, where $\mathrm{D}_j \neq \mathrm{D}_{j+1}$. Gradual concept drift is a transition phase where examples in S_{j+1} are generated by a mixture of D_j and D_{j+1} with their proportions continuously changing. Incremental concept drift is characterized by a smooth and slow transition between distributions, where the differences between D_j and D_{j+1} are not significant. Additionally, we may face recurring concept drift, in which a state from k-th previous iteration may suddenly reemerge $\mathrm{D}_{j+1} = \mathrm{D}_{j-k}$, which may take place once or periodically.

In order to tackle the presence of concept drift one may chose among three main approaches: (a) rebuilding classifier from a scratch whenever new instances become available; (b) use a specific tool to detect changes and guide the model reconstruction; and (c) use adaptive classifier that will naturally follow the changes

in the stream. The first approach is completely unsuitable due to prohibitive computational requirements, especially in case of online stream processing. Following the remaining two directions, we may distinguish four main approaches to handling drifting data streams. First one relies on using a concept drift detector - an external tool that monitors the characteristics and informs when a change is expected to appear. This allows for rebuilding the classifier only when it is deemed as necessary. Second one uses a sliding window in order to keep a track of most recent instances, as they should be most representative to the current state of the stream. Such a window follows the stream, discarding old instances and replacing them with most recent ones. However, the size of the window is a crucial factor affecting the performance of this approach. Third solution uses online or incremental classifiers that are able to incorporate new instances by updating the classification model without a need for a complete retraining. A forgetting mechanism is required in order to allow for better adaptation to changes and reduced model complexity. Finally, ensemble solutions are popularly used for mining drifting data streams. Here, new instances may be used to control diversity of the base learners, allowing them to better adapt to changes, while offering improved predictive capabilities.

2.2 Online class imbalance

Learning from imbalanced data is continuously challenging topic despite over two decades of developments in this domain [4]. Skewed distributions pose challenges to most of classifiers, as their will lead to a bias towards the majority class, while minority is usually the more important one. This has lead to a number of solutions that aims at alleviating this disproportion that can be grouped into three categories: data-level, algorithm-level and hybrid solutions. The first one uses preprocessing algorithms to balance class distributions. It involves oversampling the minority class, undersampling the majority one, or both at the same time. Second group focuses on identifying what causes a given classifier to fail in an imbalanced scenario and modifying its learning procedure in order to make it skew-insensitive. Third solution is a combination of one of the two previous ones with another learning paradigm, most commonly ensemble solution [14].

While there is a plethora of works devoted to binary imbalanced problems, its multi-class version still requires a significant attention [9]. One cannot view it as a simple extension from two to many classes, as the complexities go far beyond it. In binary cases the relationships between classes are easily defined and the bias is easy to be identified. In multi-class scenario we deal with much more complex dynamics among classes, leading to a notion of multi-minority and multi-majority cases. Using a simple decomposition into a set of pairwise relations leads to loss of useful information, as it is easy to gain performance on some of classes, while losing it on others. Furthermore, difficulties embedded in the nature of imbalanced data, such as noisy instances and class overlapping, become much more difficult to tackle.

The problem of imbalance becomes even more challenging when being considered from online and non-stationary perspective [12]. Here not only we must

deal with skewed data distributions, but also with the fact that the underlying imbalance ratio is not know from the beginning and is subject to continuous change during the stream progress. As instances arrive one by one there is a need to monitor the relationship between classes and update the learning model accordingly. In this scenario two types of changes are bound to appear. First one is evolving imbalance ratio and class properties. Here incoming instances may influence the distribution skew, either strengthening or weakening it. The role of classes in no longer stationary and in time minority and majority distributions may swap places. Therefore, online class imbalance learning requires algorithms that are not fixed on a given minority class, but can adapt themselves to constantly evolving class dynamics. This may be accompanied with the concept drift, where class boundaries may be affected. A proper solution to this problem should be able to react to both types of changes in order to achieve good adaptability, generalization and minority-majority concept description. There is a number of works for imbalanced data streams that work under a much simpler assumption that the role of classes do not change over time or that data arrives in chunks and we must handle only local skewness. There is still little works devoted to actual online class imbalance and the most efficient approaches include a neural network-based solutions [12], combination of Hoeffding decision tree with Hellinger distance splitting criterion [5], and a Bagging-based ensemble solution [12].

Multi-class online imbalance is even more difficult to handle, as classes may swap their multi-minority and multi-majority roles [11]. Therefore, not only we need to model the evolving multi-distribution imbalance ratio, changing class relationships (together with overlapping levels or noisy instances), but also take into account the fact that number of minority and minority classes may change at any stage of processed data stream. So far there are only two solutions discussed in the literature to this problem, based on ensemble of neural networks and multi-class oversampling / undersampling with online Bagging ensemble.

3 Cost-sensitive perceptron decision trees

In this section, we will discuss in details the proposed cost-sensitive perceptron decision trees with adaptive threshold for online mining of imbalanced drifting data streams, as well as the usage of McDiarmid's bound for controlling the splitting attribute selection and online analysis of the minority class structures for gaining additional information to improve the performance of classifiers.

3.1 McDiarmid's Perceptron Decision Tree

We propose to build our learning algorithm for imbalanced and drifting data streams on top of the Fast Perceptron Decision Tree [1], as it provides both high accuracy and update speed, making it highly suitable for the task at hand. Its main advantage lies in using a linear perceptron at each leaf. This allows to speed-up the decision making process, as well as improve the overall accuracy. This

hybrid solution combines the advantages of trees and neural models, allowing for efficient processing of data streams.

We use an online perceptron approach with sigmoid activation function (as suggested by Bifet et al. [1]) with squared error optimization. Let us assume that instances from the stream arrive in a form of $\langle \mathbf{x_i}, y_i \rangle$, where x_i is a feature vector for i-th instance and y_i is a class label associated with it. The perceptron learning scheme aims at minimizing the number of misclassified instances. We will annotate the learning hypothesis function of given perceptron for i-th instance as $h_{\mathbf{w}}(\mathbf{x_i})$. To evaluate the learning process, we apply the mean-square error defined as $J(\mathbf{w}) = \frac{1}{2}\sum (y_i - h_{\mathbf{w}}(\mathbf{x_i}))^2$. Bifet et al. [1] proposed to use sigmoid activation function instead of a traditional threshold and we follow this approach. The sigmoid activation function for hypothesis $h_{\mathbf{w}} = \sigma(\mathbf{w}^T \mathbf{x_i})$ is expressed as $\sigma(x) = 1/(1 = e^{-x})$, being differentiable as $\sigma'(x) = \sigma(x)\,(1 - \sigma(x))$. This allows us to calculate the error function gradient as follows:

$$\bigtriangledown J = -\sum_i (y_i - h_{\mathbf{w}}(\mathbf{x_i})) \bigtriangledown h_{\mathbf{w}}(\mathbf{x_i}), \tag{1}$$

where sigmoid hypothesis:

$$\bigtriangledown h_{\mathbf{w}}(\mathbf{x_i}) = h_{\mathbf{w}}(\mathbf{x_i})(1 - h_{\mathbf{w}}(\mathbf{x_i})), \tag{2}$$

which allows us to compute the following weight update rule:

$$\mathbf{w} = \mathbf{w} + \eta \sum_i (y_i - h_{\mathbf{w}}(\mathbf{x_i}))\, h_{\mathbf{w}}(\mathbf{x_i})(1 - h_{\mathbf{w}}(\mathbf{x_i}))\mathbf{x_i}. \tag{3}$$

As we deal with online learning, a stochastic gradient descent is being used with weights updated after each instance [1]. A single perceptron is trained per each class, making it suitable for both binary and multi-class problems. To obtain a final prediction regarding the class of new instance we select the highest value of support functions returned by each perceptron $\arg\max_k (h_{\mathbf{w_1}}(\mathbf{x}), \cdots, h_{\mathbf{w_K}}(\mathbf{x}))$, where K is the number of classes.

Original implementation of Fast Perceptron Decision Tree used Hoeffding inequality to determine the amount of instances needed for conducting a split [1]. However, recent study discussed flaws in the Hoeffding bound [8]. In this work, we propose to modify the underlying base of the original Fast Perceptron Decision Tree and use a McDiarmid's inequality for controlling the splitting criteria. It is a generalization of the Hoeffding's inequality, being applicable to both numerical and non-numerical data, as well as better describing the split measures. Let us now present the McDiarmid's theorem.

Theorem 1 (McDiarmid's Theorem) *Let $X_1, \cdots, X_n$ be a set of independent random variables and $f(x_1, \cdots, x_n)$ be a function fulfilling the following inequality:*

$$\sup_{x_1,\cdots,x_i,\cdots,x_n,\hat{x_i}} |f(x_1, \cdots, x_i, \cdots, x_n) - f(x_1, \cdots, \hat{x_i}, \cdots, x_n)| \leq c_i, \forall_{i=1,\cdots,n}. \tag{4}$$

Then for any given $\epsilon > 0$ the following inequality is true:

$$Pr\left(f(X_1, \cdots, X_n) - E\left[f(X_1, \cdots, X_n)\right] \geq \epsilon\right) \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right) = \delta. \quad (5)$$

One may apply McDiarmid's inequality to any split measure. We use it in combination with the popular Gini gain in order to estimate the minimal number of instances n to conduct a split during data stream processing [8]. One may defined the Gini gain as follows:

$$\Delta g_i^G(S) = g^G(S) - \sum_{q \in \{L,R\}} \frac{n_{q,i}(S)}{n(S)} \left(1 - \sum_{k}^{K} \left(\frac{n_{q,i}^k(S)}{n_{q,i}(S)}\right)^2\right), \quad (6)$$

where S is a set of instances in analyzed node, L and R stand for children left and right nodes, $n_{q,i}(S)$ is the number of elements in given node that will be passed to q-th child node for split made on i-th attribute, and $n_{q,i}(S)$ is the number of instances belonging to k-th class that will be passed to q-th child node for split made on i-th attribute.

This allows us to formulate McDiarmid's inequality for comparing Gini gains for any two features.

Theorem 2 (McDiarmid's Inequality for Gini Gain) *Let $S = s_1, \cdots, s_n$ be a set of instances and let $\Delta g_i^G(S)$ and $\Delta g_j^G(S)$ be the Gini gain values (see Eq. 6) for i-th and j-th feature. If the following condition is satisfied by them:*

$$\Delta g_i^G(S) - \Delta g_j^G(S) > \sqrt{\frac{8\ln(1/\delta)}{n(S)}}, \quad (7)$$

then the following inequality holds with probability of $1 - \delta$ or higher:

$$E[\Delta g_i^G(S)] > E[\Delta g_j^G(S)]. \quad (8)$$

Corollary 1 (McDiarmid's Splitting Criterion for Gini Gain) *Let us assume that $\Delta g_{i_1}^G(S)$ and $Deltag_{i_2}^G(S)$ are the metric values for features with respectively highest and second highest Gini gain. If the following condition is satisfied:*

$$\Delta g_{i_1}^G(S) - \Delta g_{i_2}^G(S) > \sqrt{\frac{8\ln(1/\delta)}{n(S)}}, \quad (9)$$

then following Theorem 2, with the probability equal to $(1-\delta)^{d-1}$ the following statement is true:

$$i_1 = \arg\max_{i=1,\cdots,d} \left\{E[g_i^G(S)]\right\}, \quad (10)$$

where d is the number of features and i_1-th feature is selected to split the current node.

3.2 Cost-sensitive modification with adaptive output

The decision tree learning algorithm discussed above will be further modified in order to make it suitable for learning from online imbalanced data. While decision trees are popular both in static imbalanced or balanced streaming data mining areas [13], for online skewed data there exists only a modification of Hoeffding Tree using Hellinger distance for conducting splits [5]. This metric, although skew insensitive, may still fail for difficult imbalanced datasets with complex class structures. On the other hand, it imposes minimal additional computational cost on the classifier - a highly desirable property in data stream mining. In non-stationary scenarios using data preprocessing is challenging and may lead to a prohibitively increased computational complexity. Therefore, algorithm-level solutions are worth pursuing and we will concentrate on them in this paper.

We propose to take an advantage of using perceptrons in leafs of the decision tree and enhance them with cost-sensitive approach. This will be achieved by modifying the output of each perceptron, instead of changing the structure of the training data or the training algorithm.

We will introduce the cost-sensitive modification in the prediction step. In the previous section for a K-class problem we denoted the continuous output of k-th perceptron for object $\mathbf{x}$ as $h_{\mathbf{w_k}}(\mathbf{x})$. In the proposed cost-sensitive approach, we will calculate the output of k-th perceptron in a leaf of our decision tree as:

$$h^*_{\mathbf{w_k}}(\mathbf{x}) = \sum_{l=1}^{K} h_{\mathbf{w_k}}(\mathbf{x}) \cdot cost[k, l], \tag{11}$$

where $cost[k, l]$ is the misclassification cost between k-th and l-th class, provided by the user.

Output modification approaches for neural classifiers have proved themselves to be efficient in tackling stationary imbalanced data [15], yet this is the first work on their usage for online class imbalance. This solution is highly compatible with data stream mining requirements, as it does not impose significant additional computational needs, do not rely on data preprocessing and can be easily included in the proposed decision tree learning scheme, taking the advantage of McDiarmid's inequality. Additionally, it is easily applicable for both binary and multi-class data streams, making it a versatile approach.

3.3 Adaptive online cost matrix

The used misclassification costs have significant influence on the performance of this method. Too low cost would not alleviate the bias, while too high cost would degrade the performance over the majority classes. We need to strive for a balanced performance. In optimal scenario the cost would be provided by a domain expert or embedded in the nature of the problem. However, in most of real-life cases we do not have a supplied cost matrix and thus a manual setting is required. There exist some automatic and semi-automatic methods applicable to stationary data, yet they cannot be used for data streams.

As the imbalanced stream will evolve over time, it is to be expected that it will affect relationships among classes. Therefore, a static cost matrix will quickly become outdated, not being able to properly reflect the current concept. On the other hand tuning it for each incoming set of instances would impose additional computational requirements, as well as a need for validation instances to select the best settings. Therefore, a lightweight solution is needed that will be able to keep a track of changes in the stream.

We propose a simple, yet effective approach of monitoring the current imbalance ratio among classes and setting the cost according to local pairwise imbalance ratios. This will allow for an easy modeling of multi-minority and multi-majority cases. The costs will change with the progress of the stream, as labels of incoming instances will be recorded and used to update the current skewness levels. Please note that this solution does not require to keep the instances in memory, as only counters for each class are needed. However, as the stream evolve over time one cannot keep all of previous information regarding class relationships. Therefore, we propose to use a fixed time threshold, as well as time stamps with each recorded label and use them to remove outdated cases from imbalance ratio counting. This allows for dynamically adapting our cost matrix to changes and drifts in the data stream.

3.4 Drift and imbalance detection

In order to efficiently learn from imbalanced and drifting data streams, we require tools that will be able to monitor the imbalance ratio and the appearance of concept drifts. We propose to combine our Cost-Sensitive Perceptron Decision Tree with Drift Detection Method for Online Class Imbalance (DDM-OCI) [10]. It is based on monitoring the recall in the minority class and if there is a significant drop in it (as evaluated by drift detector), it reports a drift. Following other works in drift detection, it may also be used to output a two-stage decision: drift warning and drift detection. This will be very useful for the next solution to online class imbalance proposed in this paper. DDM-OCI was proposed for binary online imbalance, but can be easily extended to multi-class cases. Here, we monitor the averaged recall over all of minority classes. In our case that means all of classes with an exception to the current most frequent one, allowing for taking into account that in multi-class online imbalance roles of classes are also subject to change.

3.5 Online analysis of minority class structure

Imbalance ratio among classes is not the sole source of learning difficulty. The underlying class structures, overlapping and noisy instances have significant impact on the decision boundaries being estimated. Therefore, one may assume that minority class instances may pose a different level of difficulty to the learning procedure. Recent works for static imbalanced data propose to take this factor into account and analyze the types of minority instances [6]. However,

Table 1: Six levels of instance difficulty for minority class.

	safe	borderline	borderline+	rare	rare+	outlier
ρ	5	4	3	2	1	0
λ	1	2	3	4	5	6

there is still a need for approaches that will directly incorporate this information into training procedures. Additionally, no such analysis have been done for data streams and online class imbalance.

In this work, we propose to analyze the difficulty of incoming minority class objects, while taking into account the evolving structure of classes. Firstly, let us define the types of minority instances. For their identification, we will use a neighborhood search with $k = 5$, similar as in works dealing with static data. Based on that, we propose six levels of difficulty λ that can be assigned to each new minority instance based on how contaminated is its neighborhood. This is measured by parameter ρ that states how many of k neighbors belong to the same minority class. Details are presented in Table 1. This analysis may be extended to multi-class scenario by considering each multi-minority class separately, as we have shown for static scenarios in our previous work [9].

We propose to label each new minority class instance based on this analysis. As we deal with an online scenario, we cannot nor want to keep the entire stream in the memory. Therefore, we propose to analyze the types of minority instances using a small sliding window that will keep only the most recent instances, allowing for a fast neighborhood search within it. Additionally, we will incorporate the information from the drift detector. When a warning signal is being raised, the window will be reduced to 1/4 of its original size. This will allow to accommodate the change that starts to appear by taking into account a reduced subset of recent instances. When a drift is being detected, we reset the window in order to not include instances from the previous concepts into the analysis after the change. When minority classes switch places with majority, we also reset the window.

Another issue lies in how to utilize this information regarding minority class structure during the online learning process. We propose to take advantage of perceptrons in the introduced cost-sensitive decision tree model. Their learning procedure can be influenced by the number of iterations they are allowed to spend on each instance. Therefore, each new minority instance will be presented to the cost-sensitive perceptron tree λ times during online learning, where λ is the difficulty level associated with this instance. This will shift our classifier towards concentrating on difficult instances, which in turn should lead to a better predictive performance.

4 Experimental study

This experimental study was designed in order to answer the following research questions:

- Does the proposed cost-sensitive perceptron decision tree is able to efficiently learn from online binary and imbalanced drifting data streams?
- Does the introduced online analysis of minority instance difficulties can lead to better understanding the learning difficulties in online imbalance and thus improving the performance of an underlying classifier?
- Do the proposed modifications significantly influence the memory and time requirements of the learning model?

Following subsections will describe used datasets, experimental set-up, as well as present obtained results with their discussion.

4.1 Datasets

As there are no standard benchmarks for online class imbalance learning, we selected a diverse set of both artificially generated and real-life datasets with various levels of class imbalance. Let us now describe them shortly.

- **Binary data streams**. We have created six artificial datasets with varying proportions of minority instance types. They are generated by mixing one of five predefined states (described in Table 2), each consisting of 10000 instances and 7 features created using Clover data generator [7]. Therefore, each artificial dataset has 50000 instances. Artificial1 is $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S1 \rightarrow S2$. Artificial1 is $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S1 \rightarrow S2$. Artificial2 is $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S2 \rightarrow S3$. Artificial3 is $S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S1 \rightarrow S3$. Artificial4 is $S_3 \rightarrow S_5 \rightarrow S_2 \rightarrow S3 \rightarrow S4$. Artificial6 is $S_1 \rightarrow S_5 \rightarrow S_3 \rightarrow S3 \rightarrow S4$. Additionally, we include Twitter dataset as described in [12]. We use MOA benchmarks, including RBF (10 classes), Hyperplane (5 classes), LED (10 classes), Random Tree (3 classes), and Poker (10 classes), generated using standard settings with 50000 instances each. They were transformed into binary problems by randomly selecting one class as minority and merging remaining ones as majority. After each 10000 instances minority class is swapped with one of the remaining ones.
- **Multi-class data streams**. Here, we used Chess and Tweet datasets, as described in [11]. Additionally, we used three MOA generators. RBF dataset consisted of 50000 instances, 20 features and 10 classes with random proportions and gradual drift. Hyperplane dataset consisted of 50000 instances, 10 features and 5 classes with proportions 1:5:10:20:50:100 and incremental concept drift. Random Tree dataset consisted of 50000 instances, 10 features and 3 classes with proportions 10:30:100 and sudden concept drift. Additionally, we use Yeast dataset with 8 features and 10 classes due to its difficult multi-class structure [9]. It has been copied and randomly shuffled to create 50000 instances.

Table 2: Five predefined states for generating artificial binary data streams with respect to composition [in %] of different minority instance types.

	safe	borderline	borderline+	rare	rare+	outlier
S_1	100	0	0	0	0	0
S_2	50	30	20	0	0	0
S_3	30	30	20	15	5	0
S_4	10	20	40	10	10	10
S_5	0	10	20	30	20	20

4.2 Set-up

There are but few methods for online class imbalance learning for both binary and multi-class cases. Most of them are either rooted in neural networks or ensemble approaches [12], thus not making a fair nor suitable reference for our single tree learning procedure. Therefore, as reference we have selected a standard Fast Perceptron Decision Tree (PDT) [1], to evaluate how our modifications increase its skew-insensitivity and Hellinger Hoeffding Decision Tree (HHT) [5], as this is another decision tree algorithm for imbalanced data streams. Furthermore, we evaluate the performance of the proposed cost-sensitive algorithm without taking into account the types of minority instances (CSPT) and with this extension included (CSPT+). Our algorithm is directly applicable for both binary and multi-class data streams, while for multi-class cases we modify HHT to conduct binary decompositions in each split in an identical fashion as its static version [3].

As evaluation metric, we use prequential G-mean for binary streams and prequential Averaged Recall for multi-class ones (where $AvRec = \frac{\sum_{k=1}^{K} TPR_k}{K}$). Decay factor is set to 0.995 for both metrics.

Experiments were conducted in R language using RMOA package on a machine equipped with an Intel Core i7-4700MQ Haswell @ 2.40 GHz processor and 32.00 GB of RAM.

4.3 Experiment 1: binary imbalanced data streams

Firstly, let us analyze the performance of our method on binary imbalanced data streams. We wanted to check how does the size of sliding window influence the performance of our method. Results for window size $\in [25, \cdots, 300]$ instances are reported in Table 3. From it one can see that in most cases 100 instances are enough to efficiently calculate both current imbalance ratio and minority instances types. Smaller windows cut-off too many instances, thus preventing us from gaining a more global view on the current state of the stream. Bigger

Table 3: Averaged prequential G-mean [%] for varying sizes of sliding windows used to analyze the difficulty of minority instances over binary data streams. Best trade-off between size and predictive performance bolded.

	25	50	75	100	125	150	175	200	225	250	275	300
Artificial1	77.94	78.14	81.34	**82.59**	82.59	82.59	82.59	82.59	82.59	82.59	80.43	78.67
Artificial2	78.60	78.80	81.86	**83.18**	83.18	83.18	83.18	83.18	83.18	83.18	81.37	79.61
Artificial3	78.88	79.17	82.23	**83.59**	83.59	83.59	83.59	83.46	83.42	83.60	81.65	79.94
Artificial4	79.16	79.41	82.42	**84.18**	84.18	83.78	83.83	83.60	83.60	83.93	81.79	80.13
Artificial5	78.22	78.61	81.29	**82.99**	82.99	82.99	82.99	82.99	82.99	82.99	80.52	78.86
Artificial6	78.50	78.94	81.67	**83.94**	83.94	83.94	83.94	83.94	83.94	83.23	80.85	79.19
Twitter	46.43	48.92	**50.27**	50.27	50.27	50.27	50.27	50.27	50.27	50.27	46.47	45.18
RBF	90.08	92.34	94.18	**94.51**	94.51	94.51	94.51	94.51	94.51	94.51	94.51	92.01
Hyperplane	77.54	79.89	81.02	**81.87**	81.87	81.87	81.87	81.87	81.87	81.87	80.18	79.83
LED	53.19	55.88	**56.11**	56.11	56.11	56.11	56.11	56.11	56.11	56.11	54.36	52.19
RTree	49.75	**52.12**	52.12	52.12	52.12	52.12	52.12	52.12	52.12	52.12	52.12	52.12
poker	61.34	62.67	66.03	**67.69**	67.69	67.69	67.69	67.69	67.69	67.69	67.69	67.69

windows do not contribute to the predictive power, yet significantly increase the computational complexity. Therefore, we may conclude that for estimating the imbalance ratios and minority class structures sliding window of size 100 is sufficient. This setting will be used in following experimental comparison.

Table 4 presents comparison with reference decision tree algorithms for binary online data streams. Standard PDT cannot tackle skewed distributions and becomes easily biased towards the majority class. HHT performs much better, yet CSPT and CSPT+ outperform it on 10 out of 12 data streams. This can be explained by Hellinger split criterion not being enough to counter severe class imbalance and difficult minority class structures, which is especially visible in case of six artificial datasets. In most cases CSPT+ returns the superior performance, showing that the proposed online analysis of minority instances difficulty can be beneficial to the learning process. It is interesting to notice that this happens on most of datasets, not only on six artificial ones that had explicitly generated such structures. We may conclude that difficult minority instances are bound to happen in online learning scenarios, especially when the minority class structure is constantly evolving. Therefore, it is worthwhile to incorporate such information during online classifier updating.

When taking into account both time and memory resources being used, one can see that perceptron-based solutions are faster than HHT. CSPT displays almost identical resource usage as the native PDT, proving that the proposed cost-sensitive modification and adaptive cost matrix does not impose any significant additional costs. CSPT+ displays slightly higher computational requirements, which was to be expected as for each new minority instance it analyzes its type and needs to store instances in a sliding window. However, this search is conducted only for minority instances, leading only to a slight increase in overall resource consumption which is far from being prohibitive.

Table 4: Averaged prequential G-mean [%], together with update time [s.] and memory consumption [RAM] per 1000 instances for binary data streams.

	PDT			HHT			CSPT			CSPT+		
Dataset	G-mean	Time	Memory	G-mean	Time	Memory	G-mean	Time	Memory	G-mean	Time	Memory
Artificial1	72.34	0.98	2.13	81.32	2.00	2.33	**82.59**	1.04	2.15	**82.59**	1.20	2.40
Artificial2	69.17	1.03	2.17	80.14	1.95	2.40	82.34	1.10	2.19	**83.18**	1.27	2.47
Artificial3	65.28	1.05	2.25	79.85	1.56	2.37	81.52	1.12	2.28	**83.59**	1.28	2.42
Artificial4	61.03	1.06	2.28	78.02	2.09	2.50	80.05	1.17	2.32	**84.18**	1.26	2.46
Artificial5	58.51	1.11	2.30	76.98	2.31	2.48	78.83	1.20	2.31	**82.99**	1.30	2.52
Artificial6	52.18	1.17	2.36	73.97	2.24	2.53	75.33	1.25	2.38	**83.94**	1.34	2.48
Twitter	37.37	0.99	1.78	47.24	1.43	1.98	48.42	1.04	1.82	**50.27**	1.13	2.00
RBF	70.38	1.36	2.03	87.04	2.56	2.21	92.18	1.42	2.05	**94.51**	1.51	2.19
Hyperplane	65.19	1.28	2.78	**81.26**	2.09	3.01	79.58	1.34	2.79	80.87	1.45	3.07
LED	21.89	2.17	3.01	**56.89**	3.05	3.23	54.29	2.23	3.04	56.11	2.31	3.32
RTree	19.76	2.02	2.32	47.51	2.74	2.49	51.15	2.08	2.35	**52.12**	2.18	2.46
poker	39.04	0.78	1.34	62.98	1.95	1.44	65.66	0.83	1.37	**67.69**	1.00	1.67

Table 5: Averaged prequential AvRec for varying sizes of sliding windows used to analyze the difficulty of minority instances over multi-class data streams. Best trade-off between size and predictive performance bolded.

	25	50	75	100	125	150	175	200	225	250	275	300
Chess	23.19	23.21	23.29	23.97	24.28	**25.72**	25.72	25.72	25.72	25.14	24.73	24.28
Tweet	28.54	28.93	28.99	29.36	30.42	**31.93**	31.93	31.93	31.93	31.93	31.74	31.02
RBF	37.56	37.87	37.89	40.05	**41.25**	41.25	41.25	41.25	41.25	41.25	40.51	39.99
Hyperplane	46.45	46.65	46.39	47.54	47.89	47.92	**48.31**	48.31	48.31	48.31	48.31	47.69
RTree	67.34	68.39	**70.18**	70.18	70.18	70.18	70.18	70.18	70.18	69.17	68.92	68.38
Yeast	78.23	78.78	79.02	79.43	79.58	**80.98**	80.98	80.98	80.98	80.02	78.93	78.75

4.4 Experiment 2: multi-class imbalanced data streams

We will now switch to multi-class imbalanced data streams. Let us once again check how does the size of sliding window influence the performance of our method, this time when higher number of classes is being taken into consideration. Results for window size $\in [25, \cdots, 300]$ instances are reported in Table 5. From it one can see that multi-class scenarios prefer slightly bigger sliding windows. This can be contributed the need for capturing more complex relationships among a number of distributions. Thus, more instances are needed as they will be divided among a number of classes. We will use the window size of 150 in the following experiments.

Table 6 presents comparison with reference decision tree algorithms for multi-class online data streams. Once again PDT fails to deliver satisfactory performance. However, we can see much bigger discrepancies between HHT and CSPT/CSPT+. As Hellinger distance is a binary metric, to adapt it for multi-class problems one must use a binary decomposition at each node and ten average the metric results when conducting splits. Our experiments show that this fails for multi-class imbalanced data streams. CSPT+ always returns the supe-

Table 6: Averaged prequential AvRec [%], together with update time [s.] and memory consumption [RAM] per 1000 instances for multi-class data streams.

	PDT			HHT			CSPT			CSPT+		
Dataset	AvRec	Time	Memory	AvRec	Time	Memory	AvRec	Time	Memory	AvRec	Time	Memory
Chess	2.74	1.18	3.03	14.98	2.31	3.26	23.43	1.24	3.17	25.72	1.44	4.44
Tweet	3.19	1.39	2.87	22.12	2.25	2.97	28.71	1.48	2.99	31.93	1.59	4.33
RBF	3.89	3.03	4.03	33.91	3.68	4.22	40.59	3.08	4.15	41.25	3.19	5.02
Hyperplane	5.19	3.89	5.48	37.19	4.48	5.69	47.31	3.94	5.63	48.31	4.23	6.62
RTree	20.98	2.15	2.78	58.13	2.59	3.02	67.28	2.21	2.88	70.18	2.45	3.90
Yeast	21.48	4.02	6.48	61.93	5.01	6.71	72.19	4.14	6.58	80.98	4.25	8.05

rior performance, showing that taking into account minority class structures in multi-class online imbalance plays a very important role for the learning process.

When analyzing the resource usage, we can see that perceptron-based solutions increased their costs. This is due to higher number of perceptrons being trained at each leaf. Additionally, CSPT+ needs to store more instances in the sliding window and conduct more instance difficulty analyses, as minority instances may arrive from multiple classes. However, the displayed complexity does is not prohibitive and shows that CSPT+ can be used in real-life scenarios with multi-class imbalanced data streams.

5 Conclusions and future works

In this paper, we have introduced a novel decision tree learning algorithm for online learning from binary and multi-class data streams in presence of class imbalance and concept drift, using McDiarmid's inequality. A cost-sensitive improvement to Fast Perceptron Decision Trees was introduced. It modified the outputs of perceptrons in each leaf that were used to predict a class for new instances. Cost-sensitive weighting of support functions allowed to alleviate the bias towards the majority class without introducing additional computational costs associated with data preprocessing techniques. We proposed a simple, yet effective method for calculating cost matrix dynamically using pairwise imbalance ratios measured over most recent examples. This allowed for our cost matrix to swiftly adapt to changes in class distributions during the stream progress. Furthermore, we proposed to incorporate information regarding the types and difficulties of minority class instances into the learning process. We used a sliding window solution to store a small batch of most recent instances and use them to label types of incoming minority class instances by measuring how contaminated was their neighborhood. We proposed six levels of difficulty that were used to determine how many times a given instance is used by our cost-sensitive perceptron tree during the learning process. This allowed for more difficult instances to have a greater influence over the formed decision boundaries.

Obtained results encourage us to further pursue this direction. We envision a potential of using the proposed decision tree in ensemble set-up to improve its

predictive power and drift handling capacities, as well as a need for evaluating alternative approaches to analyzing structure of minority classes.

References

1. Bifet, A., Holmes, G., Pfahringer, B., Frank, E.: Fast perceptron decision tree learning from evolving data streams. In: Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II. pp. 299–310 (2010)
2. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. 46(4), 44:1–44:37 (2014)
3. Hoens, T.R., Qian, Q., Chawla, N.V., Zhou, Z.: Building decision trees for the multi-class imbalance problem. In: Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29-June 1, 2012, Proceedings, Part I. pp. 122–134 (2012)
4. Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in AI 5(4), 221–232 (2016)
5. Lyon, R.J., Brooke, J.M., Knowles, J.D., Stappers, B.W.: Hellinger distance trees for imbalanced streams. In: 22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014. pp. 1969–1974 (2014)
6. Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. J. Intell. Inf. Syst. 46(3), 563–597 (2016)
7. Napierala, K., Stefanowski, J., Wilk, S.: Learning from imbalanced data in presence of noisy and borderline examples. In: Rough Sets and Current Trends in Computing - 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30,2010. Proceedings. pp. 158–167 (2010)
8. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision trees for mining data streams based on the mcdiarmid's bound. IEEE Trans. Knowl. Data Eng. 25(6), 1272–1279 (2013)
9. Sáez, J.A., Krawczyk, B., Woźniak, M.: Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognition 57, 164–178 (2016)
10. Wang, S., Minku, L.L., Ghezzi, D., Caltabiano, D., Tiño, P., Yao, X.: Concept drift detection for online class imbalance learning. In: The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013. pp. 1–10 (2013)
11. Wang, S., Minku, L.L., Yao, X.: Dealing with multiple classes in online class imbalance learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. pp. 2118–2124 (2016)
12. Wang, S., Minku, L.L., Yao, X.: A systematic study of online class imbalance learning with concept drift. CoRR abs/1703.06683 (2017), http://arxiv.org/abs/1703.06683
13. Wozniak, M.: A hybrid decision tree training method using data streams. Knowl. Inf. Syst. 29(2), 335–347 (2011)
14. Woźniak, M., Graña, M., Corchado, E.: A survey of multiple classifier systems as hybrid systems. Information Fusion 16, 3–17 (2014)
15. Zhou, Z., Liu, X.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans. Knowl. Data Eng. 18(1), 63–77 (2006)