Pairwise combination of classifiers for ensemble learning on data streams

Conference Paper · April 2015

DOI: 10.1145/2695664.2695754

CITATIONS
3

READS
189

3 authors:

Heitor Murilo Gomes
The University of Waikato
58 PUBLICATIONS **968** CITATIONS
SEE PROFILE

Jean Paul Barddal
Pontifícia Universidade Católica do Paraná (PUC-PR)
57 PUBLICATIONS **880** CITATIONS
SEE PROFILE

Fabrício Enembreck
Pontifícia Universidade Católica do Paraná (PUC-PR)
143 PUBLICATIONS **1,418** CITATIONS
SEE PROFILE

Pairwise Combination of Classifiers for Ensemble Learning on Data Streams

Heitor Murilo Gomes Jean Paul Barddal Fabrício Enembreck
Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155
Curitiba, Brazil
{hmgomes, jean.barddal,fabricio}@ppgia.pucpr.br

ABSTRACT

This work presents two different voting strategies for ensemble learning on data streams based on pairwise combination of component classifiers. Despite efforts to build a diverse ensemble, there is always some degree of overlap between component classifiers models. Our voting strategies are aimed at using these overlaps to support ensemble prediction. We hypothesize that by combining pairs of classifiers it is possible to alleviate incorrect individual predictions that would otherwise negatively impact the overall ensemble decision. The first strategy, Pairwise Accuracy (PA), combines the shared accuracy estimation of all possible pairs in the ensemble, while the second strategy, Pairwise Patterns (PP), record patterns of pairwise decisions during training and use these patterns during prediction. We present empirical results comparing ensemble classifiers with their original voting methods and our proposed methods in both real and synthetic datasets, with and without concept drifts. Our analysis indicates that pairwise voting is able to enhance overall performance for PP, especially on real datasets, and that PA is useful whenever there are noticeable differences in accuracy estimates among ensemble members, which is common during concept drifts.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning, Induction*

General Terms

Algorithms

Keywords

Data Stream Mining, Concept Drift, Ensemble Classifiers, Machine Learning, Supervised Learning

1. INTRODUCTION

Data stream mining has become an important research area during recent years due to the great volume of real-time data generated by mobile phones, social networks and various sensors currently available. Despite its benefits, data stream mining is a difficult task, since it is subject to, virtually, all problems that affect traditional data mining, e.g., absent values, and to specific issues that arise in a data stream configuration, most notably concept drifts.

A data stream classification algorithm is presented to a large, possibly infinite, amount of instances, each of which are made available to the algorithm in a serialized fast-paced way. On top of that, it is assumed that the underlying concept is unstable, i.e., concept drifts are expected to occur. Also, more recently [26] it was identified that some data streams exhibit temporal dependences, such that instance i_{t+1} label is not independent of instance i_t label. Therefore, a data stream classification algorithm must deal with concept drifts, consider temporal dependences and process each instance at least as fast as instances are made available to it. Also, the algorithm should not store instances for future processing, because it is not useful, nor practical to do so, as older instances are less likely to represent the current concept and the amount of instances tends to surpass available memory.

To cope with these challenges, researches have thoroughly adapted or wrapped existing classification methods to the data stream context. Many of these methods are based on ensembles of classifiers. Ensemble classifiers achieve high accuracy through the combination of a diverse set of component classifiers, such that (ideally) incorrect predictions are obfuscated while correct predictions are highlighted. In a data stream scenario, ensemble classifiers also have the advantageous characteristic of being flexible, i.e., it is possible to replace (or remove) component classifiers based on drift detector algorithms [5,6] or other methods [8,12,13,18]. Even though there is not a solid proof of a strong correlation between accuracy and diversity [19,20], it is possible to verify that a set of homogeneous classifiers that always decide for the same labels cannot achieve better (or worse) results than one of them alone. It is important to note that diversity is favorable as long as the individual decisions are properly combined. Existing ensemble classifiers for data streams use various techniques for combining predictions, such as majority vote, weighted majority [21] and other methods [2,8,13] that attempt to incorporate characteristics, mainly to account for concept drifts, that benefit classifiers adapted to the current concept over others.

Pairwise combination methods have been widely used as alternatives to decompose multiclass into binary problems [15, 23]. The voting strategies presented in this paper differ from these, as we do not decompose the problem based on the possible labels. Instead, our methods combine the outputs of a set of classifiers based on their shared estimated accuracy (Pairwise Accuracy) or on weighted prediction patterns incrementally updated during training (Pairwise Patterns). In order to test these methods, we use an ensemble structure that combines existing strategies for data stream learning and an adapted version of the Leveraging Bagging algorithm [5].

The remainder of this work is structured as follows. In Section 2 we present ensemble methods for data stream classification, with focus on their combination and voting strategies. Section 3 presents our voting strategies along with the ensemble adaptations to use them. Section 4 presents empirical results and evaluation of the proposed methods. Finally, Section 5 concludes this paper and presents future work.

2. ENSEMBLE METHODS FOR DATA STREAM CLASSIFICATION

The Online Bagging algorithm was introduced in [22] as an adaptation of the batch ensemble classifier Bagging [7]. Originally, a bagging ensemble is composed of k classifiers, which are trained with subsets (bootstraps) N_j of the whole training set N. However, sampling usually is not feasible in a data stream configuration, since it requires storing all instances before creating subsets. Authors in [22] observe that the probability of each instance to be selected for a given subset is approximated by a Poisson distribution with $\lambda = 1$, thus it is feasible to "simulate" bagging in an online fashion by training each classifier k times on each instance, such that $k = poisson(1)$.

In [6] authors present two new ensemble methods, the Adaptive-Size Hoeffding Trees (ASHT) Bagging and ADWIN Bagging. Both methods combine online bagging with a strategy to cope with concept drift. ASHT Bagging uses decision trees of varying sizes (levels) to adapt to concept drifts (smaller trees), without compromising accuracy during stable periods (bigger trees). ADWIN Bagging combines online bagging with the Adaptive Window (ADWIN) [3] algorithm for drift detection. Both algorithms use majority vote.

The Leveraging Bagging algorithm [5] is another variant of online bagging that adds more randomization to the input of the ensemble. This randomization improvement is achieved by increasing the resampling (online bagging with $\lambda > 1$). Leveraging bagging uses ADWIN to detect concept drifts, just as the methods presented in [6]. In the original paper, authors present experiments that show that leveraging bagging improves accuracy, but demands more memory and processing time, when compared to other bagging variants. To increase randomization, one can choose to use random error correcting codes at the output of leveraging bagging.

The Weighted Majority (WM) algorithm [21] weights classifiers votes based on past performance, such that every classifier has a weight β, which is decreased every time it predicts incorrectly. In order to account for evolving data streams Kolter and Maloof introduced the Dynamic Weighted Majority (DWM) [18] algoritm, which uses the same voting strategy than WM, with the addition of update heuristics such as removing classifiers if their weight β is below a given user threshold α.

The Online Accuracy Updated Ensemble (OAUE) [8] maintains a weighted set of component classifiers, such that the weighting is given by an adaptation to incremental learning of the weight function presented in [25]. Every d instances, the least accurate classifier is replaced by a candidate classifier, which has been trained only in the last d instances. Since there is no drift detection algorithm, OAUE relies on this periodic reconstructs of the ensemble to adapt to concept drifts.

The original Social Adaptive Ensemble (SAE) [12] algorithm and its improved version (SAE2) [13] are focused on three aspects of ensemble classifiers in a data stream scenario: diversity, combination and adaptation. SAE and SAE2, through different means, attempt to quantify diversity, and combine individual predictions in a way that similar classifiers predictions are grouped. Similarity between two classifiers is measured as a function of their recent individual predictions over the same instances. Subgroups of classifiers are identified through Weakly Connected Components (SAE) or Maximal Cliques (SAE2). SAE2 weights classifiers predictions in two levels. First, similar classifiers predictions are weighted and combined to reflect the subgroup prediction. The predictions of all subgroups are then weighted and combined to obtain the overall ensemble prediction. To weighed individual classifiers, SAE2 uses a weighting function based on accuracy during the current window (period). The subgroup prediction is weighted according to the average weight of its members.

The Scale-free Network Classifier (SFNC) [2] is an ensemble method that weights classifiers predictions based on an adaptation of the scale-free network construction model. In SFNC, classifiers are arranged in a graph structure, such that classifiers with higher accuracy are more likely to connect to recently added classifiers. Classifiers weighting is directly proportional to a user given centrality metric α, e.g., eigenvector. Since high accuracy classifiers usually receives most of the connections, these are expected to have higher influence on the overall decision. Although, this process is non-deterministic and, theoretically, low accuracy classifiers can become prominent as well. The worst classifier, in terms of accuracy, is removed from the network every p instances, and that triggers a rewiring process to maintain the graph connected.

3. PAIRWISE BASED VOTING

Other works usually focus on building highly diverse ensembles [5, 12, 13, 22], although it is always expected some degree of overlap between classifiers. The voting strategies presented here are aimed at using these overlaps to support ensemble prediction. In the following sections we present the Pairwise Accuracy (PA) and the Pairwise Pattern (PP) voting strategies, along with the adaptations needed for an ensemble to incorporate them. PA and PP are based on the hypothesis that by combining pairs of classifiers it is possible to alleviate incorrect individual predictions that would impact the overall ensemble decision.

3.1 Pairwise Accuracy

Pairwise Accuracy (PA) combine classifiers into pairs, and weights the predictions of these pairs based on their shared estimated accuracy in the most recent instances. To esti-

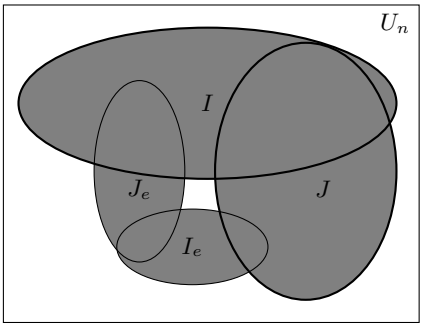

Figure 1: Venn diagram representation of window n and classifiers c_i and c_j correctly and incorrectly classified subsets of instances

mate accuracy, we use a similar approach to the weighting function in [13], such that individual and pairs of classifiers are weighted incrementally and reset periodically according to a fixed window (period). Formally, let U_n be the set of all instances from window n, I and J be subsets of U_n, which classifiers c_i and c_j were able to correctly classify, respectively, and I_e and J_e instances drawn from U_n that c_i and c_j incorrectly classified, respectively. Also, $U_n^t = I \cup J \cup I_e \cup J_e$ is the set of all instances from window n already presented to c_i and c_j up to instance t. We define the pairwise estimated accuracy, namely $S_{acc}(c_i, c_j)$ (Equation 1), of classifiers c_i and c_j, as the ratio of instances from U_n^t that both correctly classifies. Conversely, c_i and c_j shared estimated error rate, namely $S_{err}(c_i, c_j)$ (Equation 2), is defined as the ratio of instances from U_n^t that both incorrectly classifies.

$$S_{acc}(c_i, c_j) = \frac{|I \cap J|}{|U_n^t|} \tag{1}$$

$$S_{err}(c_i, c_j) = \frac{|I_e \cap J_e|}{|U_n^t|} \tag{2}$$

We also define the accuracy estimation for a single classifier c_i during window n up to instance t, namely $acc(c_i)$ (Equation 3), as the ratio of instances from U_n^t that c_i was able to correctly classify.

$$acc(c_i) = \frac{|I|}{|U_n^t|} \tag{3}$$

Figure 1 shows a graphical representation of how sets I, J, I_e, J_e could overlap.

In order to use S_{acc} and S_{err} during voting, we employ a weighting function that prioritizes equal pairwise predictions over individual predictions. For all c_i classifiers in C at window n, voting is performed pairwise, such that $\binom{C}{2}$ pairs are formed. Using a vector $\vec{v}$ to store weights during voting, such that every position in $\vec{v}$ corresponds to a possible label and is initialized with 0, and assuming individual predictions are denoted by $h(x)$, each pair of classifiers (c_i, c_j) contributes to the overall prediction through Equation 4, if c_i prediction matches c_j prediction, i.e., $h_i(x) = h_j(x)$, or Equations 5 and 6, if c_i and c_j predictions diverge for instance x, i.e., $h_i(x) \neq h_j(x)$.

$$\vec{v}(h_i(x)) := \vec{v}(h_i(x)) + S_{acc}(c_i, c_j) - S_{err}(c_i, c_j) \tag{4}$$

$$\vec{v}(h_i(x)) := \vec{v}(h_i(x)) + acc(c_i) - S_{acc}(c_i, c_j) \tag{5}$$

$\vec{p}$	Corr. 0	Corr. 1	...	Corr. (k-1)
(0,0)	12	4	...	0
(0,1)	3	16	...	1
⋮	⋮	...	...	...
(k-1,k-1)	3	5	...	18

Figure 2: Example of the data stored for a given pair of classifiers c_i and c_j for a classification problem with k classes. Every entry in $\vec{p}$ has a one-to-one relation to a line in M.

$$\vec{v}(h_j(x)) := \vec{v}(h_j(x)) + acc(c_j) - S_{acc}(c_i, c_j) \tag{6}$$

The overall ensemble prediction is obtained through Equation 7.

$$\arg\max_i \vec{v}(i) \tag{7}$$

Through Equations 4, 5 and 6, it is possible to observe that given two classifiers with high S_{acc}, their split predictions will be degraded, while their equal predictions will obtain a high weight, which in turn is decreased by their shared mistakes S_{err}. If pairs disagree on their predictions, their individual decisions will be taken into account, but their weights will be decreased according to their estimated shared accuracy (S_{acc}).

In comparison to existing weighting methods, PA resembles OAUE and SAE2 weighting mechanism as it formulation is concerned about how changes in the data distribution may impact classifiers' weights. The main difference is that PA combines classifiers pairwise, which may reduce the impact of classifiers committed to previous concepts on the overall decision.

3.2 Pairwise Patterns

Similarly to PA, Pairwise Patterns (PP) generates all possible pairs of classifiers $\binom{C}{2}$, but instead of estimating shared accuracy, PP records predictions patterns, during training, and use these patterns to weight decisions while predicting the label of an unknown instance x.

PP maintains a vector $\vec{p}$ with all possible $d = k^2$ prediction patterns given two classifiers and k classes, and a matrix $M_{d,k}$ which is updated during training and used to determine which labels corresponds to each pattern. $M_{d,k}$ has one column for each possible label and one line for each pattern. During training, classifiers c_i and c_j predict the label of an instance x independently, their predictions are then combined into a pattern which is used to find the corresponding line index in $M_{d,k}$, while the correct label y determines the column index that must be incremented in $M_{d,k}$. The overall ensemble prediction for an unknown instance x is the label that receives more votes based on the observed patterns from all pairs of classifiers for instance x.

Figure 2 presents an example of how $M_{d,k}$ and $\vec{p}$ are related for a given pair. It is important to notice that the whole ensemble just needs one vector $\vec{p}$, but every pair of classifiers must have a distinct matrix $M_{d,k}$.

Using the example from Figure 2, while predicting the label of an unknown instance x, if $h_i(x) = 1$ and $h_j(x) = 0$, then pair p_{c_i,c_j} combined prediction is going to be 10 for label 0 and 4 for label 1.

3.3 Ensemble adaptations for PA and PP

To make it possible to test PA and PP, we use an ensem-

ble structure based on existing methods. We denote this ensemble by Generic Ensemble (GE) and briefly describe it.

GE updates its component classifiers periodically based on a predefined window, such that the worst or oldest classifier is replaced by a background learner (also known as candidate [8, 13]) trained during the last window. The default method and the adaptation for PA replaces the worst classifier, according to Equation 3, while the adaptation for PP replaces the oldest classifier, since the latter does not assess classifiers individually. During the first window GE only has one active classifier and a background classifier. In the second window the background classifier becomes active and a new background classifier is created. This process continues until the maximum number of classifiers has been reached (user threshold), and after that new classifiers can only replace existing classifiers. There are mainly two reasons to perform this process. First, classifiers are trained on different chunks of data and this assists in creating a diverse set. Second, the background classifier is adapted to the latest concept, and therefore it can gradually "adapt" the ensemble if drifts occurs. The background classifier do not take part of voting during predictions, since it could potentially degrade the overall performance as it may have a "weaker" hypothesis. The default voting strategy of GE is based on the combination of the individual classifiers weight, such that weights are assigned according to Equation 3.

Besides using GE to test PP, we have also adapted the leveraging bagging [5] algorithm to use PP for voting. The changes made to leveraging bagging includes the addition of a vector of patterns $\vec{p}$ to the ensemble, and for each pair of classifiers a matrix M. In order to account for the fact that leveraging bagging uses ADWIN to detect concept drifts, whenever a drift is detected, and a classifier c is reset, all matrices M corresponding to classifier c are reset as well. As a consequence, values in different matrices M may vary in scale. To adjust for that, each pair vote is weighted according to the total amount of instances that both classifiers have predicted together. We have not adapted leveraging bagging to work with PA, since PA weight functions depends on a fixed window size.

4. EXPERIMENTS

In our experimental framework we assess mainly the accuracy of classifiers. To evaluate accuracy we apply the Prequential [11] evaluation procedure with sample frequency = 1/10 of the data stream length. We use Prequential evaluation since it gives accuracy estimates that approximates those of a holdout evaluation and allows using all instances for testing and training. To account for random decisions in SAE2, Leveraging Bagging, ADWIN Bagging, SFNC and GE, the experiments were repeated third times varying the algorithms random seeds. The overall accuracy measure was obtained from the average of the third repetitions. Even though, other aspects such as processing time and memory are important for data stream evaluation, we do not present these here due to the limited space, and because the overhead caused by the voting strategies only marginally affects memory and processing time. All experiments were configured and executed on MOA (Massive Online Analysis) framework [4].

We compare our modified ensemble classifiers with PA and PP to their default methods (without pairwise voting), and also to other ensemble classifiers. The reason to compare the modified ensembles against default implementations is to check whether (and when) these pairwise voting strategies are able to enhance the overall ensemble accuracy.

ID	Data stream configuration		
	Data generator	# drifts	Type of drift
RTG	RTG	-	-
AGR1	AGRAWAL	2	A/A
AGR2	AGRAWAL	2	G/G
SEA1	SEA	2	A/A
SEA2	SEA	2	G/G
HYPE	Hyperplane	-	I

Table 1: Synthetic data streams configurations (A: Abrupt Drift, G: Gradual Drift, I: Incremental Drift)

The other ensemble parameters were set according to their original publications, except for the maximum number of classifiers and the window (period), which were set to 10 and 1% of the data stream total length, respectively. For example, the window size for a data stream with 1 million instances was set to 10 thousand instances.

The experiments include four real datasets and six variations of synthetic data streams. The synthetic data includes evolving (abrupt, gradual and incremental drifts) and stationary (no drifts) data streams. Table 1 presents which types of drifts were simulated for every synthetic data stream used. We refrain from providing a detailed explanation on each dataset for brevity, thus we refer the reader to the original publications, reviews or web resources on each dataset: Spam Corpus (SPAM) [17], Forest Covertype (COVT)[1], Airlines (AIRL)[2], Electricity (ELEC) [14], SEA generator (SEA) [24], Agrawal generator (AGR) [1], Random tree generator (RTG) [10], Hyperplane generator (Hyper) [16].

Tables 2 and 3 presents the results for the adapted methods (LevBag-PP, GE-PA and GE-PP) with their default methods (LevBag and GE). LevBag-PP was able to boost LevBag accuracy most notably in 3 real datasets (SPAM, AIRL and COVT). We observed that the best results for LevBag-PP were achieved in situations where classifiers with low individual accuracy for a given class, when combined translated their prediction pattern into the most often correct label. For example, classifiers c_i and c_j predict label 1 for an unknown instance x, but it is more likely that the correct label is 0 when both decide for 1.

Through analysis of Table 3 it is noticeable that GE-PA was able to achieve higher or comparable accuracy in most datasets, while GE-PP was more unstable. In order to understand in which situations PA voting surpassed that of GE we analyzed in details experiments AGR1 and AGR2. Figures 3 and 4 shows that GE-PA was able to adapt itself to the second concept drift (around instance 666,666) faster than GE. GE-PA performance reported at 0.7×10^6 (see Figures 3 and 4) cannot be credited solely to background classifiers replacing older classifiers, as this same mechanism took place on GE during the same period.

By analyzing the period comprehended between instances 666,000 and 700,000 for AGR1, we observed a scenario, unusual during other periods, in which the amount of disagree-

[1] `https://archive.ics.uci.edu/ml/datasets/Covertype`
[2] `http://kt.ijs.si/elena_ikonomovska/data.html`

Dataset	**LevBag-PP**	**LevBag**
AGR1	93.64 ± 0.15	**93.69** ± 0.83
AGR2	90.95 ± 1.17	**91.07** ± 1.29
AIRL	**63.7** ± 0.28	62.67 ± 0.25
COVT	**92.95** ± 0.26	92.19 ± 0.28
ELEC	90.67 ± 0.24	**90.82** ± 0.21
RTS	97.99 ± 0.1	**98.21** ± 0.09
SEA1	**89.91** ± 0.04	89.64 ± 0.34
SEA2	**90.46** ± 0.03	90.45 ± 0.04
SPAM	**93.89** ± 0.55	93.11 ± 0.35
HYPE	**90.75** ± 0.11	90.29 ± 0.12

Table 2: Average accuracy for LevBag and LevBag-PP. The best accuracies per data stream are indicated in boldface.

Dataset	**GE-PA**	**GE-PP**	**GE**
AGR1	**94.2** ± 0.27	87.43	92.04 ± 0.08
AGR2	**92.42** ± 0.41	82.7	90.87 ± 0.01
AIRL	**66.38** ± 0.15	62.67	66.21 ± 0.02
COVT	87.72 ± 0.43	**89.67**	88.31 ± 0.16
ELEC	**86.03** ± 0.17	84.76	85.97 ± 0.15
RTS	95.13 ± 0.08	**96.57**	95.19 ± 0.02
SEA1	89.32 ± 0.22	86.54	**89.33** ± 0
SEA2	89.41 ± 0.07	86.51	**89.43** ± 0
SPAM	87.19 ± 0.06	**88.76**	87.1 ± 0.02
HYPE	**91.16** ± 0.06	86.42	91.15 ± 0.06

Table 3: Average accuracy for GE, GE-PA and GE-PP. The best accuracies per data stream are indicated in boldface. GE-PP standard deviation was below 10^{-9} for all experiments.

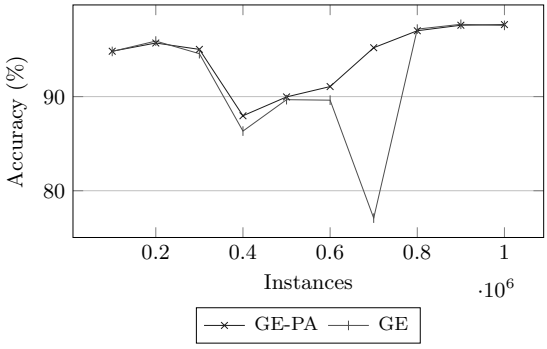

Figure 3: Accuracy on the AGR1 experiment (2 abrupt concept drifts around instances 3.33×10^5 and 6.66×10^5)

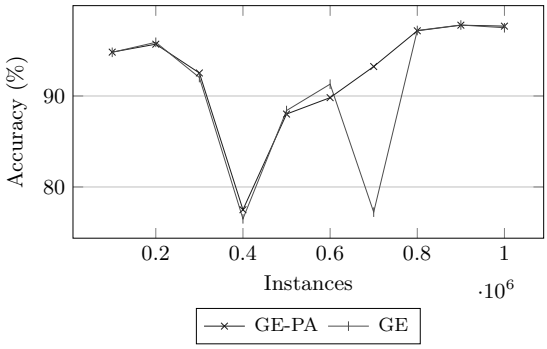

Figure 4: Accuracy on the AGR2 experiment (2 gradual concept drifts around instances 3.33×10^5 and 6.66×10^5)

ment between pairs was high, and it was for the better. In this scenario, during prediction, classifiers with low individual weight (adapted to the previous concept) "transferred" their condition to other classifiers that predicted the same label as them, since their estimated shared accuracy (S_{acc}) were low as well, while classifiers with high individual weight (adapted to the current concept) were able to achieve significant weight in split decisions when combined with these low weight classifiers.

Table 4 presents the average accuracy for LevBag-PP and GE-PA in comparison to other ensemble methods. To verify if there were statistically significant differences between algorithms, we performed non-parametric tests using the methodology from [9]. First, we used the Friedman test with $\alpha = 0.05$ and the null hypothesis "there were no statistical difference between given algorithms" was rejected. We proceeded with the Nemenyi post-hoc test to identify these differences. The Nemenyi test indicates, with a confidence level of 95%, that for the given experiments {LevBag-PP, OAUE} $\succ$ {DWM}.

5. CONCLUSION

In this work we have presented two voting strategies that weights classifiers predictions based on their shared accuracy (PA) or prediction patterns (PP). Intuitively, using pairs instead of single classifiers weights adds more information to what is already known from each component classifier to the ensemble. It allows for insights such as "if classifier A decides for 0 and B for 1, most likely A is wrong". Through empirical experiments we have shown that even though our current pairwise strategies were not able to provide huge accuracy improvements, they were able to achieve intermediate results on most datasets and, when compared to the default voting methods, for some datasets improvements were identified. Also, we identify the most important contribution of this work not as the specific pairwise voting methods, but as the possibility that by combining classifiers into pairs one may be able to model more sophisticated weighting mechanisms that take into account agreements (and disagreements) between classifiers.

For future work, we plan to extend PP for more than two classifiers, thus generating more complex patterns, and to use other weighting functions for PA that take into account temporal dependences and unbalanced classes. Currently, we are investigating the impacts of ensemble size on PA and PP, and the adaptation of the voting methods to other ensemble methods. Another interesting analysis regards the correlation between diversity metrics and "good" interactions observed throughout the experiments.

6. REFERENCES

[1] R. Agrawal, T. Imilielinski, and A. Swani. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, Dec. 1993.

[2] J. P. Barddal, H. M. Gomes, and F. Enembreck. Sfnclassifier: A scale-free social network method to handle concept drift. In *Proceedings of the 29th*

Dataset	**LevBag-PP**	**GE-PA**	**ADWBag**	**DWM**	**OAUE**	**SFNC**	**SAE2**
AGR1	93.64 ± 0.15	94.2 ± 0.27	94.36 ± 0.2	86.5	93.77	93.33	**94.68** ± 0.15
AGR2	90.95 ± 1.17	92.42 ± 0.41	90.69 ± 1.32	82.41	**93.24**	92.31	89.79 ± 2.09
AIRL	63.7 ± 0.28	66.38 ± 0.15	66.05 ± 0.32	61.46	64.48	**66.42**	60.8 ± 0.58
COVT	92.95 ± 0.26	87.72 ± 0.43	85.67 ± 0.25	91.28	**93.55**	85.85	86.59 ± 0.53
ELEC	**90.67** ± 0.24	86.03 ± 0.17	85.05 ± 0.33	84.69	89.38	85.38	85.8 ± 0.48
RTS	**97.99** ± 0.1	95.13 ± 0.08	95.6 ± 0.09	93.64	97.35	95.09	95.06 ± 0.12
SEA1	89.91 ± 0.04	89.32 ± 0.22	88.63 ± 0.48	88.6	**90.02**	89.54	89.86 ± 0.17
SEA2	**90.46** ± 0.03	89.41 ± 0.07	90.15 ± 0.08	88.63	90.25	89.16	90.12 ± 0.1
SPAM	**93.89** ± 0.55	87.19 ± 0.06	88.34 ± 0.9	88.21	67.23	86.5	87.76 ± 0.42
HYPE	90.75 ± 0.11	**91.16** ± 0.06	90.5 ± 0.12	88.2	90.41	90.91	90.88 ± 0.12
Avg. Rank	2.6	3.4	4.2	6.1	3	4.4	4.3

Table 4: Comparison of average accuracy. The best accuracies per data stream are indicated in boldface. SFNC standard deviation were below 10^{-14} for all experiments.

Annual ACM Symposium on Applied Computing, SAC '14, pages 786–791. ACM, 2014.

[3] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *SIAM*, 2007.

[4] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[5] A. Bifet, G. Holmes, and B. Pfahringer. Leveraging bagging for evolving data streams. In *PKDD*, pages 135–150, 2010.

[6] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *15th ACM SIGKDD*, pages 139–148, 2009.

[7] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[8] D. Brzezinski and J. Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50–67, 2014.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, Dec. 2006.

[10] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM SIGKDD, Sep. 2000.

[11] J. Gama and P. Rodrigues. Issues in evaluation of stream learning algorithms. In *15th ACM SIGKDD*, pages 329–338. ACM SIGKDD, June 2009.

[12] H. M. Gomes and F. Enembreck. Sae: Social adaptive ensemble classifier for data streams. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 199–206, 2013.

[13] H. M. Gomes and F. Enembreck. Sae2: Advances on the social adaptive ensemble classifier for data streams. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14. ACM, 2014.

[14] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, 1999.

[15] T. Hastie, R. Tibshirani, et al. Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998.

[16] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[17] I. Katakis, G. Tsoumakas, and I. Vlahavas. An adaptive personalized news dissemination system. *Journal of Intelligent Information Systems*, 32(2):191–212, Apr. 2009.

[18] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. In *The Journal of Machine Learning Research*, pages 123–130. JMLR, Jan. 2007.

[19] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[20] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6(1):22–31, 2003.

[21] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[22] N. C. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, pages 105–112. Society for Artificial Intelligence and Statistics, Jan. 2001.

[23] B. Quost, T. Denoeux, and M.-H. Masson. Pairwise classifier combination using belief functions. *Pattern Recognition Letters*, 28(5):644–653, 2007.

[24] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-classification. In *Proc. of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM SIGKDD, Aug. 2001.

[25] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM, 2003.

[26] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, pages 1–28, 2014.