Pattern Analysis and Applications manuscript No.
(will be inserted by the editor)

Creating ensemble classifiers through order and incremental data selection in a stream

Application to the online learning of road safety indicators

Nicolas Saunier · **Sophie Midenet**

Received: September 16th 2010 / Accepted: December 26th 2011

Abstract This paper presents an original time-sensitive traffic management application for road safety diagnosis in signalized intersections. Such applications require to deal with data streams that may be subject to concept drift over various time scales. The method for road safety analysis relies on the estimation of severity indicators for vehicle interactions based on complex and noisy spatial occupancy information. An expert provides imprecise labels based on video recordings of the traffic scenes. In order to improve the performance - overall and for each class - and the stability of learning in a stream, this paper presents new ensemble methods based on incremental algorithms that rely on their sensitivity to the processing order of instances. Different data selection criteria, many used in active learning methods, are studied in a comprehensive experimental evaluation, including benchmark datasets from the UCI Machine Learning Repository and the prediction of severity indicators. The best performance is obtained with a criterion that selects instances which are misclassified by the current hypothesis. The proposed ensemble methods using this criterion and AdaBoost have similar principles and performance, while the proposed methods have a smaller computational training cost.

Keywords Road safety · Traffic management · Ensemble methods · Incremental algorithms

Supported by the French “Region Ile de France”

N. Saunier
Civil, Mining and Geological Engineering Department, École Polytechnique de Montréal, C.P. 6079, succ. Centre-Ville, Montréal Québec, Canada, H3C 3A7
Tel.: +1 (514) 340-4711 (ext. 4962)
Fax: +1 (514) 340-3981
E-mail: nicolas.saunier@polymtl.ca

S. Midenet
Université Paris-Est, IFSTTAR, GRETTIA, F-93166 Noisy-le-Grand, France
E-mail: sophie.midenet@ifsttar.fr

1 Introduction

1.1 Traffic Management Systems

Road traffic management systems (TMS) seek to improve road traffic flow in urban environments through a better utilization of existing infrastructures. This work targets time-sensitive applications that must adapt in the short to medium term respectively at the operational and tactical levels. Indeed, road traffic management requires at the operational level to continuously monitor traffic variables (flow, speed and density) and other potentially disrupting events (incidents), and to continuously make decisions based on the most recent data. Traffic management also relies on decision support systems at the tactical level to provide diagnosis over a longer time frame (in the order of hours to days). In addition, the traffic data input to TMS shows strong temporal patterns at different time scales (daily, weekly, seasonal) but also evolves in time: traffic tends to increase over time, but is also regularly affected by unforeseen events such as economic crises. Such data may therefore suffer from *concept drift*, i.e. when data, its distribution and its processing (interpretation) change over time [18].

In the scope of a research project on advanced traffic management methods for signalized intersections, we had access to an experimental intersection where multiple video sensors are installed and supply a video processing tool. This tool computes traffic data measurements like occupancy ratios, queue lengths and traffic flows that are used for several traffic management applications such as traffic control, incident detection, red light running detection and road safety diagnosis [4, 24, 25]. In this context, we use machine learning techniques to automatically learn and predict traffic indicators, which take a small number of values (classes).

We favour incremental learning methods because they are particularly suited for TMS, where the data stream may be subject to concept drift. Indeed, incremental algorithms construct a classifier with only one pass over the data and update the classifier at each step with the new data, without accessing the data seen previously [36]. The ability to update classifiers on new data is very advantageous in case of concept drift, instead of forgetting what was learnt previously or re-training classifiers on the whole augmented dataset.

1.2 Road Safety Diagnosis in Signalized Intersections

This paper focuses on a specific module of our research project devoted to road safety diagnosis in signalized intersections. Our approach relies on the detection of vehicle interactions and the evaluation of their severity. Vehicles are interacting if they are close enough in time and space. The severity, or proximity to a potential collision, is measured by the spatio-temporal distance between the interacting vehicles, and is related to their probability of collision. Several severity indicators have been proposed in the road safety literature [1, 31]. This work relies on a generic method based on an expert's judgment.

Our system uses the spatial occupancy measurements of the experimental intersection provided by the video processing tool as input data. The intersection surface is virtually covered by a grid, each grid unit covering approximately one sixth of the size of a standard vehicle. Through video analysis, each grid unit is assigned one of six possible states representing the dynamics of the vehicles that passed on the corresponding part of the roadway over one second (Figure 1). This data and the video traffic scenes were recorded for a complex intersection in the suburbs of Paris over a period of eight months (Figure 2). The resulting database provides the data used in this work.

The task is twofold: the detection of vehicle interactions and the evaluation of their severity. Vehicle interactions are detected and categorized by a first rule-based component, described in [28, 29]. The second component, described in this paper, predicts a severity indicator (a class of severity) for each interaction instance on the basis of the examples labeled by the expert.

Fig. 2 The experimental intersection.

As illustrated in Figure 1, the input is a nominal vector concatenating the states of all grid units in the conflict, upstream and downstream zones of the approach with the green light; based on the vehicle speeds, the severity of the detected interaction in the example has been labeled as maximum by the expert watching the video recordings. Even though the state of the grid units may seem explicit, the limited spatio-temporal definition, the fact that vehicles are not individually identified and the noise make their interpretation difficult. Supervised machine learning techniques are used to tackle these issues and provide robust solutions over manually tuned rule-based systems.

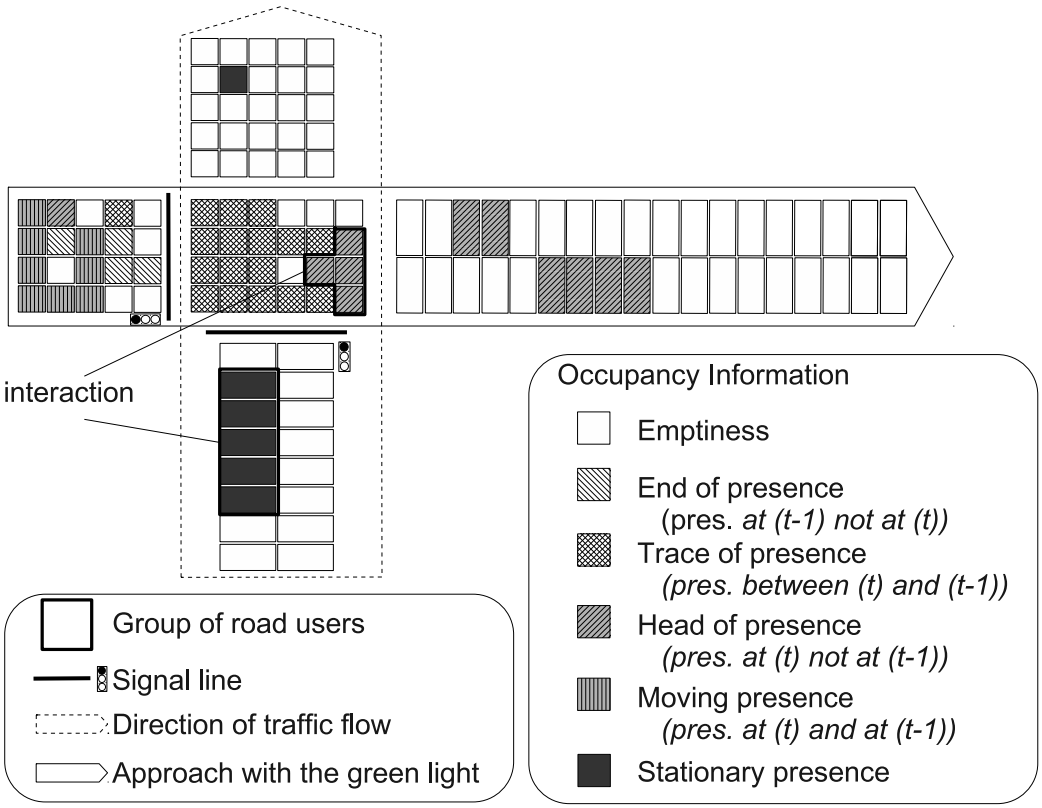

Fig. 1 Example of input data, with one detected interaction of the highest severity.

1.3 Proposed Approach

A supervised machine learning task is defined by three components [12]: the environment that generates the data, an expert that supervises and a learner that builds hypotheses (the term hypothesis is used here as a synonym to classifier).

This paper presents original ensemble methods based on incremental algorithms in which the learner iteratively selects informative instances in order to improve the classification accuracy, overall and for each class. These new methods rely on a feature of many incremental algorithms: their sensitivity to the processing order of instances. Most incremental algorithms build different hypotheses if the training instances are presented in different orders. These hypotheses can then be combined through a vote. This paper explores the use of order to create ensembles of classifiers. To the authors' knowledge, sensitivity to the processing order of instances has not been used explicitly in ensemble methods.

The incremental data selection algorithms are introduced in the next section. After a review of ensemble methods, the new ensemble methods are presented in section 3. The experimental evaluation on benchmark datasets and the road safety application is performed in section 4 and section 5 concludes this paper.

2 Incremental Data Selection in a Stream

2.1 Learning Severity Indicators

As no tracking is performed by the video processing tool, an early choice was made to interpret the data $x_t \in X$ at each time t, independently from the other instants. This data at time t contains occupancy information over the period of time between $t-1$ and t which makes it possible to estimate vehicle dynamics (see Figure 1). The task is to learn how to associate a severity indicator y_t to each x_t. Based on the video recordings of the traffic scenes, the judgement of the expert is uncertain and imprecise. The expert can distinguish only a few ordered severity levels. Let Y be the set of these levels or classes ($y_t \in Y$). Since evaluating the severity indicator may be difficult, the expert may hesitate and therefore not be able to label the data, in which case it cannot be used for learning. Besides, it is observed that instances can look very similar and be labeled in different classes with certainty by the expert: there seems to be a significant class overlap. Finally, the classes are unbalanced, and it is important for the application to maintain good performance for each class.

Instances are available only up to time t in a stream, which defines an online setting [3], as opposed to an offline or batch setting, in which all the data is available. As

mentioned, incremental algorithms are the natural solution to deal with data streams, and learning in one pass over the data is also very appealing when dealing with very large datasets in batch mode. An algorithm is strictly incremental if its requirements for memory and per example computation do not increase with the number of instances [18]. An incremental algorithm is lossless if it produces a hypothesis at time t functionally equivalent to the corresponding hypothesis trained on the same data up to time t in one batch.

2.2 First Approach

This section revisits the work presented in [30] that aims at improving the accuracy of severity indicator prediction. Given the complexity of the task, in particular the noise in the data, specific methods are needed to improve the accuracy of the hypotheses in difficult parts of the input space, e.g. where classes overlap at their boundaries: the approach relies on selecting a subset of the available instances for learning. Learning with a data stream allows the learner to make an iterative selection of the most informative instances, without knowing the data distribution, in order to specify the boundaries between the classes. A method that selects instances involves an active learner, i.e. a learner that has some control over the learning process [7, 34]. Data selection in a stream is thus cast in the active learning framework.

Most active learning algorithms deal with membership queries, where the learner can select instances and ask an expert for their labels. Asking for the "right" instances can greatly reduce the number of labeled instances. Most methods are applied to the "pool-based" setting, where a set of unlabeled instances is available from the beginning, and the learner picks some iteratively for labeling at each step. Many pool-based algorithms choose instances maximizing or minimizing some utility attached to each instance, such as the confidence of the current hypothesis in its prediction of the instance class [22]. In the "stream-based" setting, the learner decides whether to ask or not the label for each incoming instance of the stream. In both settings, an incremental hypothesis learning algorithm is a great advantage since during the active learning process the hypotheses are iteratively used for prediction and then updated with the newly labeled instances. In the remainder, the component learning algorithm A is supposed to be incremental (Naive Bayes is used for A in our experiments): it builds a hypothesis $h : X \rightarrow Y$ on a training dataset. If requested, an instance is labeled by the expert if he can.

Algorithm 1 Generic Simple algorithm

Input: A stream of data $S = [x_t]$
A labeled dataset for initialization L^{init}
Output: Final hypothesis h_{final}

begin
  Run A on L^{init} with output hypothesis h_0
  $t = 0$
  repeat
    if $SelectionCriterion(x_t, h_t)$ is true **then**
      ($y_t \in Y$ is obtained in $SelectionCriterion$)
      Update h_t on (x_t, y_t) with A to produce h_{t+1}
    else
      $h_{t+1} = h_t$
    $t = t + 1$
  until $StoppingCriterion$
  $h_{final} = h_t$
end

Algorithm 1 describes a classical generic incremental learning algorithm for data selection in a stream: in addition to the learning algorithm, it is characterized by a selection and a stopping criterion. Instances can be selected either before or after labeling. Instances are selected before labeling in most active learning approaches, which aim at reducing the number of labeled instances. If the expert cannot label the instance, *SelectionCriterion* returns automatically $false$ (this is not repeated in the following selection criteria to simplify their presentation). In the stream-based setting, the most famous technique is the Query-by-Committee (QBC) [8, 33], which relies on a set, or committee, of hypotheses H.

$SelectionCriterion\_QBC(x, h)$
Draw randomly 2 hypotheses h_1 and h_2 from H.
if $h_1(x) \neq h_2(x)$ **then**
  ask the expert for the label y of x and return $true$.
else
  return $false$.

In the pool-based setting, criteria for data selection often rely on a definition of the utility $utility_h(x)$ of a pool instance x with respect to the current hypothesis h. They can be adapted to the online setting by using a threshold [15] or biased random decision [8]. A utility commonly used in the literature is derived from the confidence of a hypothesis h on its prediction for an instance x denoted $confidence_h(x)$ and typically normalized to $[0,1]$: the objective is to select instances where the hypothesis is uncertain, i.e. $utility_h(x) = 1 - confidence_h(x)$.

$SelectionCriterion\_CThres(x, h)$
if $utility_h(x) \geq threshold$ **then**
  ask the expert for the label y of x and return $true$.
else
  return $false$.

SelectionCriterion_CRand(x,h)
Draw a random variable *rand* following a uniform distribution on $[0,1]$.
if $rand \leq utility_h(x)$ **then**
  ask the expert for the label y of x and return $true$.
else
  return $false$.

Yet, the primary goal of this work is not to minimize the number of labeled instances, but to build the hypotheses with the best accuracy on all classes. This is why the selection of labeled instances was tested through the "windowing" technique that selects instances that are misclassified by the current hypothesis [14]. Misclassified instances are likely to be close to the boundaries between classes and to help better classify in these areas. It was shown experimentally that the criterion that selects labeled instances that are misclassified by the current hypothesis, called *MC* (for *M*is*C*lassified), yields the best results over the other criteria on the road safety application [30].

SelectionCriterion_MC(x,h)
Ask the expert for the label y of x.
if $h(x) \neq y$ **then**
  return $true$.
else
  return $false$.

The version of the generic incremental data selection algorithm (Algorithm 1) with the MC criterion is called *SimpleMC* (the algorithms with the other criteria are respectively called SimpleQBC, SimpleCThres and SimpleCRand). Specific stopping criteria exist, e.g. for Support Vector Machines (SVMs) [32], but the only generic stopping criterion is an extra labeled dataset kept for performance evaluation. In any case, the availability of the expert implies a finite number of labeled instances, and thus constitutes the ultimate stopping criterion of this algorithm. It should also be kept in mind that the learning is never really finished when dealing with a data stream, and it should be possible to later update the hypotheses if necessary, in particular in the case of concept drift.

The main shortcoming of this approach based on data selection is stability. A learning algorithm is unstable when small changes in the training set lead to significantly different learnt hypotheses and big performance fluctuations [9]. Ensemble methods are a common solution to reduce the instability of learning algorithms. A simple ensemble method that combined the last versions of the hypothesis was introduced in [30]. Although it offered the advantage of a small computational cost by using previous versions of the hypotheses (already computed), the key limitation was the diversity in this set of hypotheses. As each hypothesis is the previous hypothesis updated with only one new instance, the ensemble diversity is bound to decrease as the number of instances used for updating increases. The next section presents new ensemble methods to address this issue.

3 Using the Order of Training Instances to Create Classifier Ensembles

3.1 Ensemble Methods

Ensemble methods for classification and regression have attracted a great deal of interest in the recent years. A good introductory survey is provided in [27]. An ensemble is a collection of hypotheses, called base or component hypotheses, whose predictions are combined by weighted average or vote [17]. Based on a substantial amount of theoretical and empirical evidence, and on the availability of smart training methods for ensembles, an ensemble of hypotheses is generally more accurate than a single hypothesis.

An intuitive key characteristic of ensemble methods is the diversity of the set of combined hypotheses [20, 19]. There is obviously no point in combining identical hypotheses, or no need for ensemble if the "perfect" hypothesis is available. When ensemble components are imperfect, they should be different so that at least some of them are correct where the others are wrong. In the special issue of Information Fusion on Diversity in Multiple Classifier Systems, [6] provides a comprehensive survey of diversity creation methods, introducing the idea of implicit and explicit diversity creating methods. A technique such as Bagging [5] is an implicit method, since it randomly samples the training data to produce different sets for each hypothesis learning, without any measurement to influence diversity. Boosting [13] is an explicit method, since it directly manipulates the training data distributions to ensure some form of diversity. Numerous theoretical studies explain the success of Boosting by proving bounds and margins on its error. Bagging and Boosting are the most famous and successful ensemble methods. This categorization is similar to the difference made in [9] between independent and coordinated constructions of hypotheses.

Diversity can be induced along three dimensions: 1) the starting point in hypothesis space, 2) the set of accessible hypotheses, 3) the traversal of hypothesis space. Hypotheses are made accessible through the manipulation of training data. The space of possible training sets is characterized by three dimensions: the training instances, the features that describe the instances, and the pre-processing of features to create a different representation of the instances ("distortion" methods). This category of methods, which manipulate training data, include implicit methods such as Bagging and k-fold cross-validation, and explicit methods such as Boosting, Input Decimation [35], DECORATE [23] and the ma-

nipulation of output targets using Error Correcting Output Codes [10]. While some methods rely on measures of the diversity of hypotheses, there are "doubts about the usefulness of diversity measures in building classifier ensembles in real-life pattern recognition problems" [20]. This paper presents methods using the sensitivity of incremental algorithms to the processing order of instances to create diverse sets of hypotheses. To the authors' knowledge, this mechanism has not been used explicitly in previous ensemble methods. It can be argued that algorithms like Bagging and Boosting, by manipulating the training sets, may take advantage of this characteristic, but they do not require explicitly base classifiers that are sensitive to the processing order of instances.

3.2 Batch Mode

If the component algorithm A of SimpleMC is deterministic, so is SimpleMC. The hypothesis produced by SimpleMC is determined by the initialization dataset L_{init} and the data stream S. Diversity can be created through the manipulation of one of these two elements. In the batch mode, the manipulation of the processing order of instances is possible. Although a non-deterministic component algorithm A could alternatively be used, this work relies on a deterministic algorithm, the Naive Bayes classifier (NB).

In batch mode, a dataset $L = \{x \in X\}$ is available, labeled or not. When instances are used for training by A, their label is available, by asking an expert if necessary. The proposed algorithm initializes each hypothesis h^k on a dataset L^k_{init} randomly drawn from the training dataset L (all L^k_{init} have the same size n^k_{init}). The rest $L \backslash L^k_{init}$ is then presented in a random order to each hypothesis and processed according to the selection criterion.

Algorithm 2 Generic BatchVote algorithm

Input: A dataset $L = \{x \in X\}$
Output: Final hypothesis obtained by the votes of $\{h^k_{final} | 0 \leq k \leq K-1\}$.
begin
  for $k = 0 \ldots K-1$ **do**
    Draw a random subset L^k_{init} of L
    Run A on L^k_{init} with output hypothesis h^k_0
    $i = 0$
    for each x_i drawn in a random order in $L \backslash L^k_{init}$ **do**
      if $SelectionCriterion(x_i, h^k_i)$ is true **then**
        (y_i is obtained in $SelectionCriterion$)
        Update h^k_i on (x_i, y_i) with A to produce h^k_{i+1}
      else
        $h^k_{i+1} = h^k_i$
      $i = i + 1$
    $h^k_{final} = h^k_i$
end

Algorithm 2 is generic with respect to the selection criterion and is called *BatchVote* (the version of this algorithm with MC is called *BatchVoteMC*). The parameters of BatchVote are the number of hypotheses K and the size n^k_{init} of the initialization datasets L^k_{init} for each hypothesis. When using the MC criterion, n^k_{init} should be small in order to take advantage of its ability to select instances for learning and its sensitivity to the processing order of instances to create diversity. BatchVoteMC follows similar principles to Boosting techniques, namely distorting the training data distribution towards misclassified instances. While each hypothesis of BatchVoteMC will independently process instances in a given order, some instances should be misclassified by many hypotheses, e.g. at the class boundaries, and thus be present in many selected training datasets.

3.3 Online Mode

When learning online, with a data stream S, the order of the data instances cannot be controlled. The diversity can then be created by manipulating the initialization of the component hypotheses. BatchVote can be used for that purpose. The proposed algorithm initializes a set of diverse hypotheses on the initialization labeled dataset L_{init} of size n_{init} with BatchVote. Each instance in the stream S is then considered for selection by each hypothesis: since the selection criterion depends on each hypothesis, which have a different initialization, it will pick different instances from the same data stream for the different hypotheses.

Algorithm 3 Generic StreamVote algorithm

Input: A stream of data $S = [x_t]$
  An initialization labeled dataset L_{init}
Output: Final hypothesis obtained by the votes of $\{h^k_{final} | 0 \leq k \leq K-1\}$.
begin
  Initialize K hypotheses $\{h^k_0 | 0 \leq k \leq K-1\}$
  with BatchVote on L_{init}
  $t = 0$
  repeat
    for $k = 0 \ldots K-1$ **do**
      if $SelectionCriterion(x_t, h^k_t)$ is true **then**
        (y_t is obtained in $SelectionCriterion$)
        Update h^k_t on (x_t, y_t) with A to produce h^k_{t+1}
      else
        $h^k_{t+1} = h^k_t$
    $t = t + 1$
  until $StoppingCriterion$
  for $k = 0 \ldots K-1$ **do**

$h^k_{final} = h^k_t$
end

Algorithm 3 is generic with respect to the selection and stopping criteria and is called *StreamVote* (the version of this algorithm with MC is called *StreamVoteMC*). For a given initialization dataset L_{init} and data stream S, the parameters of StreamVote are the number of hypotheses K and the size of the initialization datasets per hypothesis n^k_{init} ($n^k_{init} \leq n_{init}$). If the component algorithm A is incremental, this algorithm is incremental and updates the hypotheses on each incoming instance x_t. Like BatchVoteMC, StreamVoteMC follows similar principles to Boosting techniques. While each hypothesis of StreamVoteMC has a different initialization, some instances of the stream should be misclassified and selected by many hypotheses.

3.4 Computational Cost of the Proposed Algorithms

BatchVoteMC and StreamVoteMC have the same complexity: the comparison is made between BatchVoteMC and AdaBoost, the most common Boosting algorithm, since they are both non-incremental. Training an ensemble of K hypotheses on a dataset of N instances, is more expensive with the AdaBoost algorithm [27] than the BatchVoteMC algorithm (assuming an incremental base algorithm A). Each hypothesis is trained on the whole dataset for AdaBoost, but only on the subset of iteratively misclassified instances for BatchVoteMC. For both algorithms, each hypothesis predicts the label of each instance that is compared to the true label: in addition, AdaBoost computes the total error and a weight for each hypothesis, which is also used to update the probability distribution of the instances. AdaBoost therefore involves more computations than BatchVoteMC per instance and per hypothesis: the extra cost is constituted of at least K computations of the total weighted error and $K \times N$ updates of instance weights.

4 Experimental Evaluation

4.1 Experimental Setup

The learning algorithms are evaluated on benchmarks composed of classical real datasets from the UCI Machine Learning Repository Database [2], and on the severity prediction task. Naive Bayes classifiers are used as the component learning algorithm in all presented ensemble methods. This choice was made on account of the good accuracy of this incremental and fast learning algorithm, especially in the severity prediction task. NB handle nominal data, have no parameter and are not sensitive to the processing order of instances. They have shown good performance on a wide range of tasks [11, 16], and are frequently used in ensemble methods [26].

Table 1 The algorithms compared in the experimental evaluation are classified depending on three characteristics: algorithms that build ensemble of hypotheses ("Ensemble"), algorithms that construct a hypothesis with only one pass over the data and update the hypothesis at each step with the new data ("Incremental"), and algorithms that select instances as a form of active learning ("Data selection").

Algorithm	Ensemble	Incremental	Data selection
NB		×	
SimpleMC		×	×
SimpleCThres		×	×
SimpleCRand		×	×
StreamVoteMC	×	×	×
QBC	×	×	×
StreamVoteCThres	×	×	×
StreamVoteCRand	×	×	×
BatchVoteMC	×		×
Bagging	×	×	
Adaboost	×		
SVM			

The algorithms proposed in this paper, SimpleMC, BatchVoteMC and StreamVoteMC, are compared to other algorithms (see the full list and their main characteristics in Table 1). The Weka toolbox was used for the implementation of the algorithms and the experimental evaluation [37]. The incremental lossless version of Bagging is preferred [21]. The version of QBC is derived from Bagging, where the set of hypotheses learnt independently in the Bagging algorithm is used to select instances. An incremental non-lossless version of AdaBoost exists [26], but for practical reasons, the non-incremental implementation of Weka is used as it is reported to yield equivalent or slightly better results.

The performance of a hypothesis is evaluated through the global rate of correctly classified instances or accuracy (Acc), the recall (R_c) and precision (P_c) for each class c. Based on the confusion matrix of a hypothesis h, which holds at the position (i, j) the number of instances of the class c_i with predicted class c_j by h, with notation $M_{i,j}$, and the number of classes N_c, the accuracy, recall and precision for each class are defined as

$$Acc = \frac{\sum_{1 \leq i \leq N_c} M_{i,i}}{\sum_{1 \leq i \leq N_c,\ 1 \leq j \leq N_c} M_{i,j}}$$

$$R_c = \frac{M_{c,c}}{\sum_{1 \leq j \leq N_c} M_{c,j}}; \quad P_c = \frac{M_{c,c}}{\sum_{1 \leq i \leq N_c} M_{i,c}}$$

In the severity prediction task, the number of instances in each class can vary greatly. With the recall and precision for each class, it can be checked that a hypothesis performs well on all classes.

4.2 Benchmarks

4.2.1 Description

Table 2 Description of the 11 benchmark datasets [2], with the number of instances, attributes and classes ("cmc" stands for "contraceptive method choice", "derma" for "dermatology" and "disc" for discretized). All their attributes are nominal.

Dataset	Number of instances	Number of attributes	Number of classes
mushroom	8124	23	2
soybean	683	36	19
vote	435	17	2
tic-tac-toe	958	10	2
iris disc	150	5	3
car	1728	7	4
cmc	1473	10	3
connect-4	67557	43	3
derma	366	35	6
promoters	106	58	2
audiology	226	70	24

The benchmarks described in the table 2 provide information about the performance of the algorithms on other datasets than the severity of vehicle interactions. None of these benchmarks have a natural order. Incremental algorithms process them in one pass in a given order (an initialization dataset L_{init} is drawn if necessary). The attributes describing the data are all nominal, after discretization if necessary, for comparison with the road safety data and direct processing by NB. These datasets are very different. The performance of the algorithms is evaluated with ten-fold cross-validation (the partitions are identical for all algorithms). The results for SVM are not available for the largest dataset connect-4 due to computation time.

4.2.2 Sensitivity of StreamVoteMC to its Parameters

The influence of the parameters of StreamVoteMC (the size of the initialization datasets per hypothesis n^k_{init}, and the number of combined hypotheses K) on accuracy is studied. The results are shown for the "tic-tac-toe" benchmark binary classification task. The size n_{init} of the initialization dataset L_{init} is kept small with respect to the size of the datasets to be closer to a data stream situation. For $n_{init} = 10$, the accuracy is plotted as a function of n^k_{init}, varying from 1 to n_{init}, for different values of K in Figure 3. When n^k_{init} nears n_{init}, there is a drop of accuracy: this is expected for $n^k_{init} = n_{init}$ as there is no more diversity in the set of hypotheses in this limit case, and this diversity is reduced for values close to n_{init}.

Accuracy is also plotted as a function of K for different values of n^k_{init} in Figure 3. As expected, accuracy increases

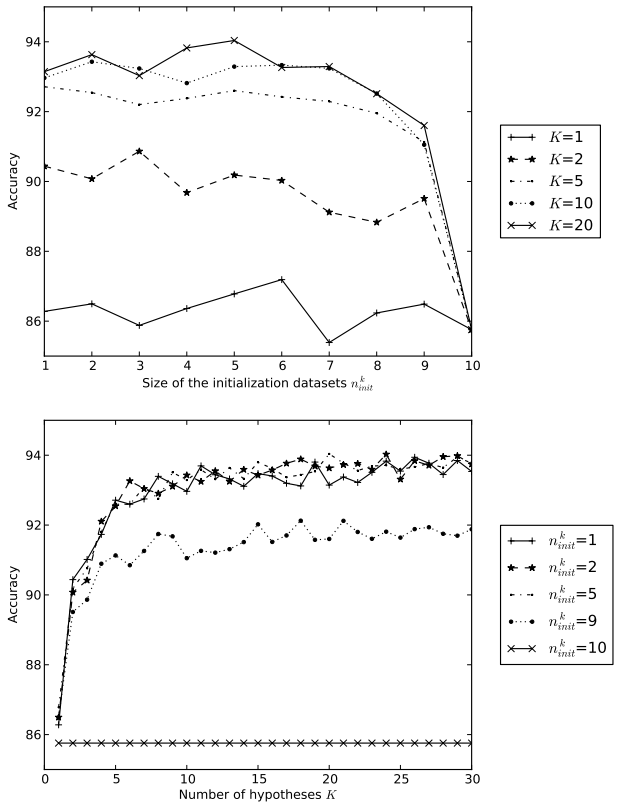

Fig. 3 Accuracy of StreamVoteMC for the vote benchmark dataset ($n_{init} = 10$). The results are averaged over 10 random partitions of the datasets for cross-validation. On the first row, accuracy is plotted as a function of n^k_{init}, varying from 1 to $n_{init} = 10$, for different values of K. On the second row, accuracy is plotted as a function of K for different values of n^k_{init}.

with the number of hypotheses. The standard deviation is not plotted since it shows no dependence on the parameters. For the extensive comparison of all algorithms on all benchmark datasets, the following parameter values are chosen: $n^k_{init} = 6$ (for $n_{init} = 10$), and $K = 10$. A small default value $n^k_{init} = 1$ is chosen for BatchVoteMC to take full advantage of its characteristics.

4.2.3 Accuracy Results

The complete accuracy results for all algorithms are presented in the Figures 4 and 5. BatchVoteMC and StreamVoteMC appear to have consistently the best accuracy or close to the best among the incremental algorithm relying on data selection. More importantly, if some data selection algorithms perform well in some cases, such as QBC, CThres and StreamVoteCThres, there is always a dataset on which they have particularly weak accuracy, such as tic-tac-toe and car.

StreamVoteMC is consistently more accurate than SimpleMC. However, the accuracy of NB is not systematically

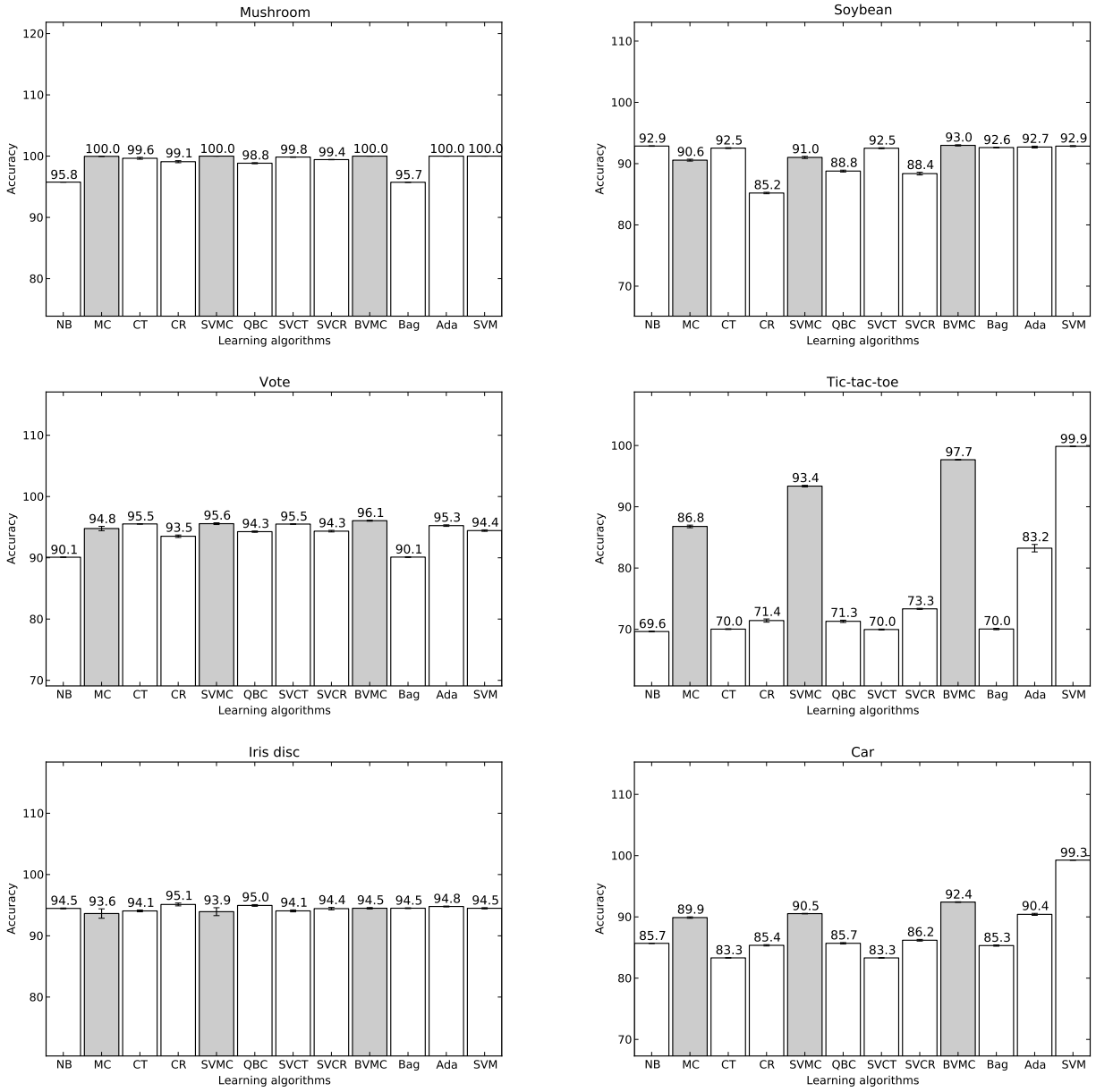

Fig. 4 Accuracy averaged over 50 tenfold cross-validation, with standard deviation, on the 6 first benchmark datasets for all algorithms (see the list of shortened names at the bottom right of Figure 5).

improved by SimpleMC and StreamVoteMC on five datasets, soybean, cmc, connect-4, dermatology and promoters. The fact that AdaBoost and Bagging have similar weaknesses may imply that the performance of NB cannot be easily improved upon. StreamVoteMC accuracy is not so far from AdaBoost accuracy, and is even better on two datasets. BatchVoteMC shows remarkable results, having the best accuracy on all but the last three datasets. StreamVoteMC is less accurate, but has the advantage of being incremental. As can be seen in Figure 6, BatchVoteMC and StreamVoteMC require about the same amount of time for learning, and about K times more than SimpleMC, as expected. More importantly, they require much less processing time than AdaBoost, i.e. consistently about half the computation time of AdaBoost in these experiments.

SVM is the most accurate classifier on four datasets, but the learning algorithm is not incremental and its training time is several orders of magnitude higher than all other learning algorithms, which make it unsuitable to deal with data streams and concept drift.

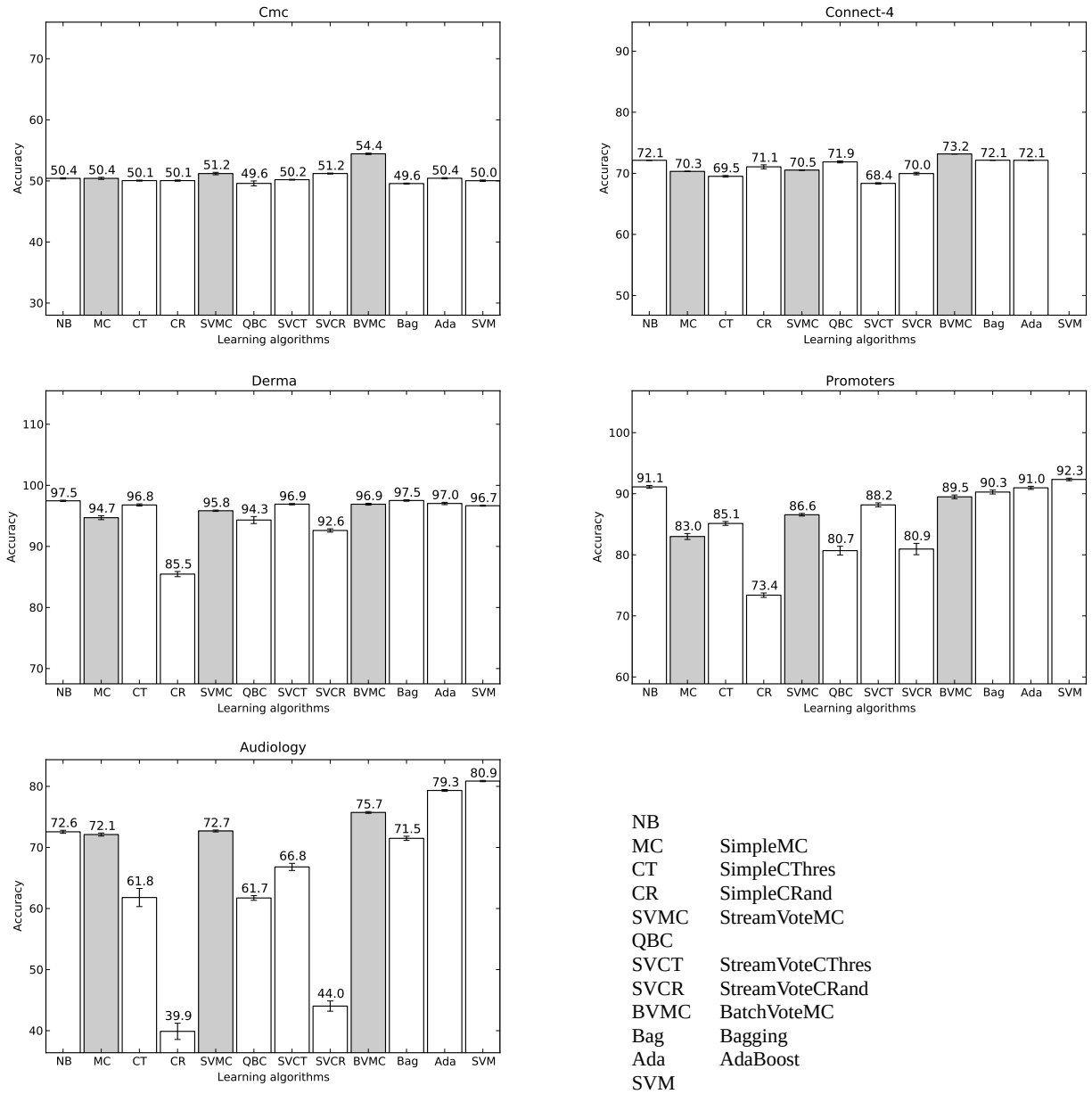

Fig. 5 Accuracy averaged over 50 tenfold cross-validation, with standard deviation, on the 5 last benchmark datasets for all algorithms (see the list of shortened names at the bottom right).

4.3 The Road Safety Application

4.3.1 Description

Data was labeled for two different sub-intersections of the complex intersection under study. For these two tasks, the expert can distinguish three severity levels, called MIN, MED and MAX. There are many more instances in the class MED. The results are based on three labeled datasets (initialization, training and test) for each sub-intersection, described in table 3. The data streams were labeled sequentially for 35 and 50 minutes. To be more representative, each test dataset contains data from four separate periods, under peak and fluid traffic conditions.

Table 3 Characteristics of the severity prediction task: number of instances in the initialization dataset in which L_{init} is drawn (to average results with different initialization datasets), in the data stream S and in the test set; number of attributes (i.e. number of grid units, see Figure 1).

Sub-intersection	Number of instances Initialization / S / Test	Number of attributes
1	254 / 755 / 311	80
2	66 / 410 / 349	96

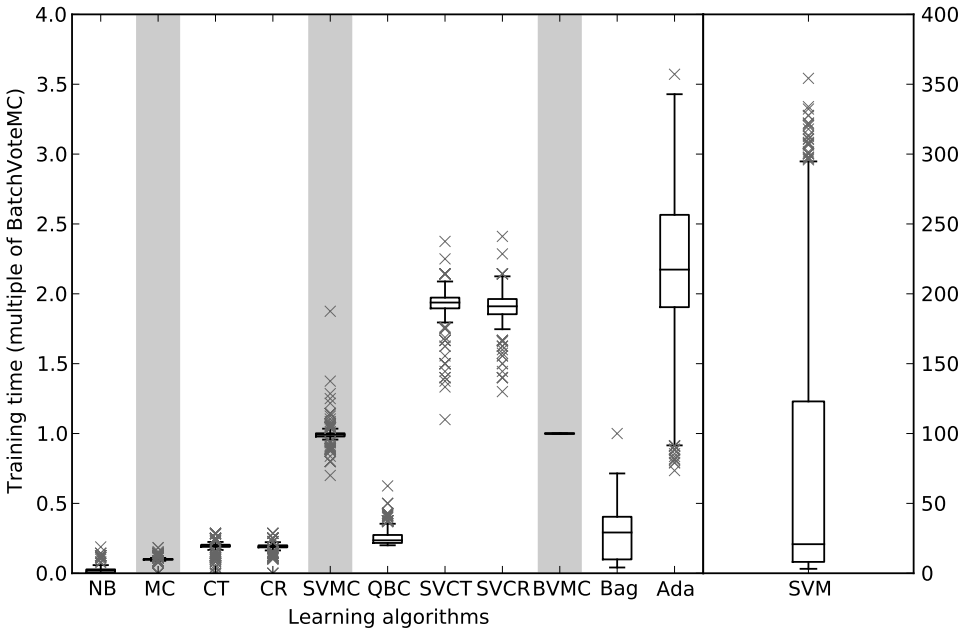

Fig. 6 Distributions of the computational time taken by the learning phase of each algorithm over 50 runs for each dataset, as a proportion of the time for BatchVoteMC (pay attention to the different scale for SVM on the right) (see the list of shortened algorithm names at the bottom right of Figure 5).

4.3.2 Sensitivity of StreamVoteMC to its Parameters

In a typical learning task with a data stream, the size of the initialization dataset L_{init} should be small, but sufficient to create diversity among hypotheses. For $n_{init} = 25$, the influence of n^k_{init} and K on the performance of StreamVoteMC is studied in Figure 7. Accuracy does not appear to depend on n^k_{init}, except for values close to n_{init}. However, on the contrary to the benchmark datasets, the standard deviation does depend on n^k_{init}, and a small value should be chosen. K has a greater impact on accuracy than n^k_{init}: the average accuracy increases and its standard deviation decreases as K increases. The chosen values are $n^k_{init} = 6$ and $K = 15$.

4.3.3 Learning Results

The classification results at the end of the stream for the two sub-intersections are presented in Figure 8 (boxplots are used because several result distributions cannot be represented well with only the mean and standard deviation, e.g. for the algorithms SimpleCThres and SimpleCRand). BatchVoteMC is not tested on the severity prediction task since incremental algorithms are favoured for this online application. It is satisfying to notice that SimpleMC has better accuracy and results for almost all classes than NB, and that it is the same situation for StreamVoteMC with respect to SimpleMC. The learning task seems to be more difficult for sub-intersection 1 than for sub-intersection 2. StreamVoteMC has good accuracy for the two sub-intersections, slightly better than AdaBoost but inferior to SVM for sub-intersection 1, whereas these three algorithms are tied together for sub-intersection 2. It should be noted that the good accuracy results of SVM are achieved at the expense of the results on some classes, especially the MIN and MAX classes which have much fewer instances than the MOY class (see for example the MAX class for sub-intersection 2). AdaBoost shows more balanced results, but they are still weak for the MIN and MAX classes for sub-intersection 1. The results of the other incremental algorithms for data selection are significantly worse than the best results, degrading most NB results, except for QBC.

4.3.4 Learning Curves

The accuracy of the algorithms are estimated on the test datasets every five processed instances of the stream (selected or not) and the learning curves are displayed in Figure 9. Results are erratic at the beginning of the stream. The accuracy standard deviation is not displayed to clarify the plots: it decreases as more instances are processed. Similarly to the previous results at the end of the stream it appears that StreamVoteMC improves SimpleMC, which itself already improves NB. Similar learning curves per class, not displayed here, show also that StreamVoteMC has more balanced results, when other algorithms like SVM and AdaBoost sacrifice the results on the MIN or MAX classes.

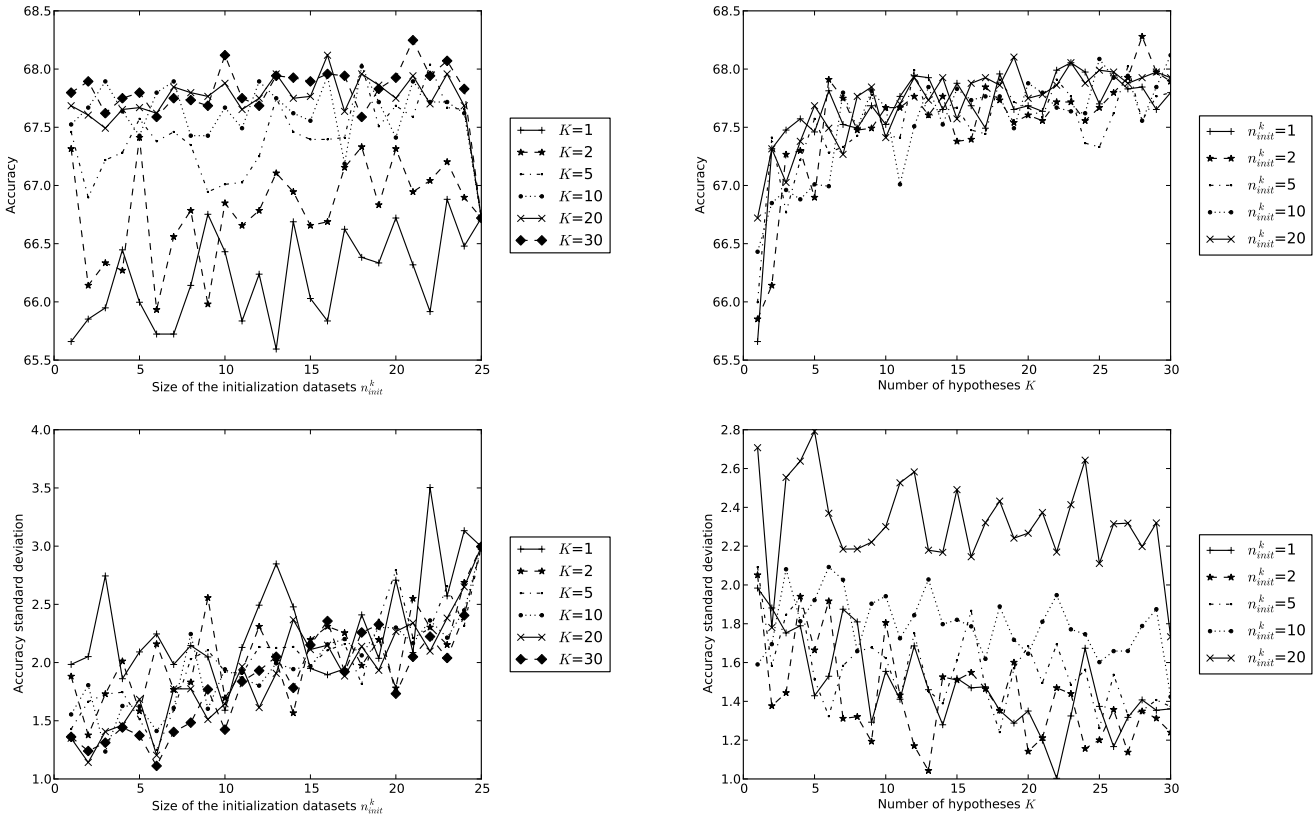

Fig. 7 Accuracy averaged over 20 random drawings of L_{init} with $n_{init} = 25$ (top) and its standard deviation (bottom) for one sub-intersection. They are plotted as a function of the number of initialization instances per hypothesis n^k_{init} for different values of the number of hypotheses K (on the left), and as a function of the number of hypotheses K for different values of the number of initialization instances per hypothesis n^k_{init} (on the right).

5 Conclusion

This paper has presented new learning methods that meet the demand of time-sensitive traffic management systems: based on their sensitivity to the processing order of instances, the new learning algorithms create diverse ensemble of hypotheses. These ensemble methods show good results on a variety of classification tasks, especially the incremental generic StreamVote algorithm that deals with data streams in the severity prediction task for the road safety application that motivated this work. The comparison of the proposed algorithms to AdaBoost is particularly encouraging, with close or better results on most datasets at a smaller computational cost.

For lack of space, the possibility for the expert to refuse labelling some instances was not presented in details. Tests on the road safety application were made using fuzzy sets to model the expert judgement, permitting the use of "fuzzy labels" to label instances at the boundaries between classes. Preliminary results using instances with fuzzy labels through methods similar to Error Correcting Output Codes [10] did not show any improvements. It should still be noted that fuzzy labels are a useful tool for the expert, that could make a difference for other tasks, especially with more than three classes.

The generic algorithms BatchVote and StreamVote are also interesting pool-based and stream-based active learning ensemble methods that should be investigated as such. An important unknown point of this work is the impact of the Naive Bayes classifiers as base hypothesis for the ensemble methods. In further steps, the dependence of the results on the base algorithm and the way the selection criteria interact with the base algorithm should be analyzed. A natural extension will be to deal explicitly with concept drift, in which case the issue of selecting a stopping criterion becomes the issue of detecting changes.

Acknowledgment

The authors wish to thank the reviewers for their constructive comments that helped to significantly improve the paper.

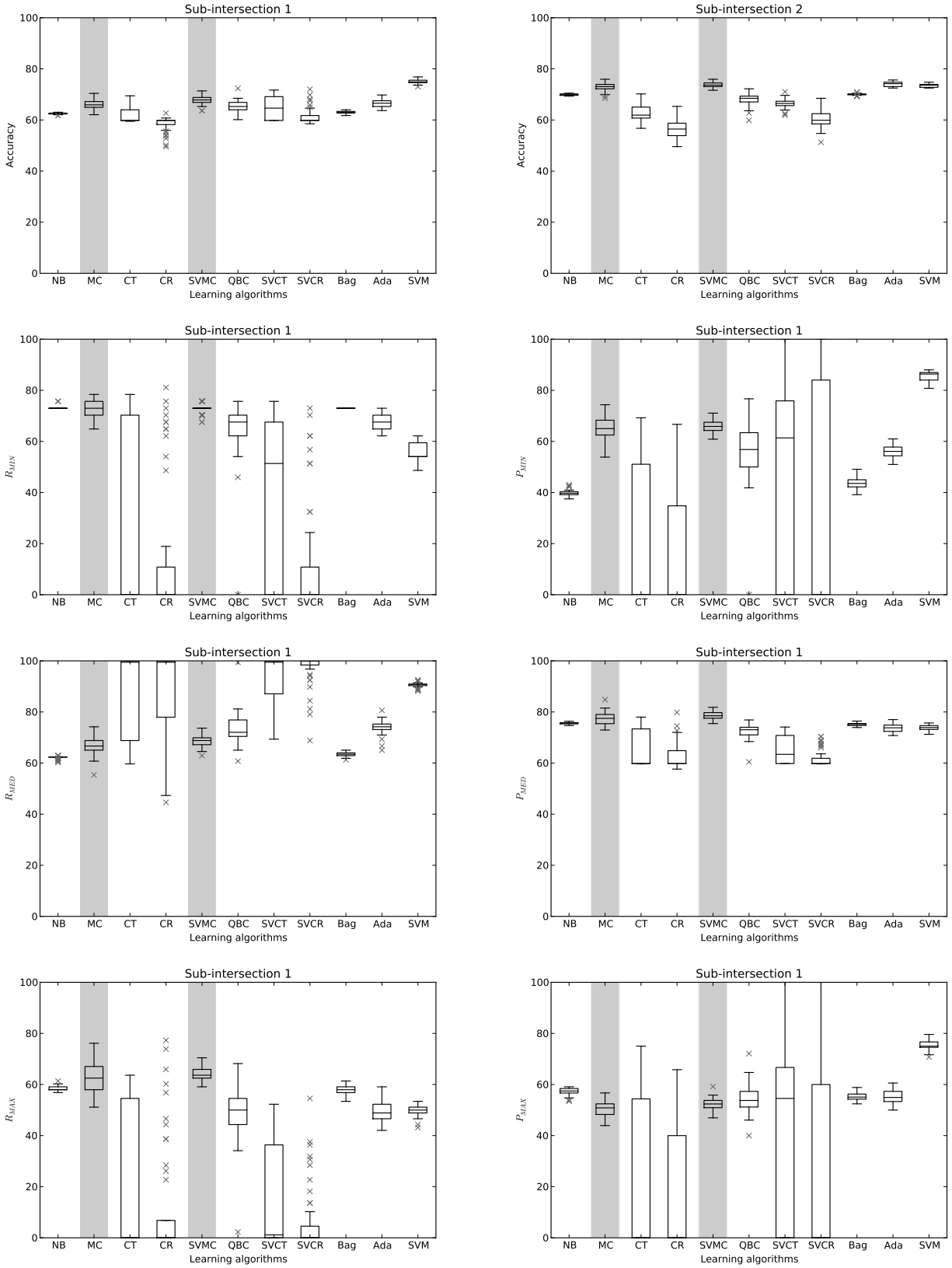

Fig. 8 Distributions of all final results obtained from 50 random drawings of L_{init} for each algorithm: overall accuracy for sub-intersections 1 and 2 (respectively top left and right), recall R_c and precision P_c for each class c (on each row) on the sub-intersection 1 (see the list of shortened algorithm names at the bottom right of Figure 5).

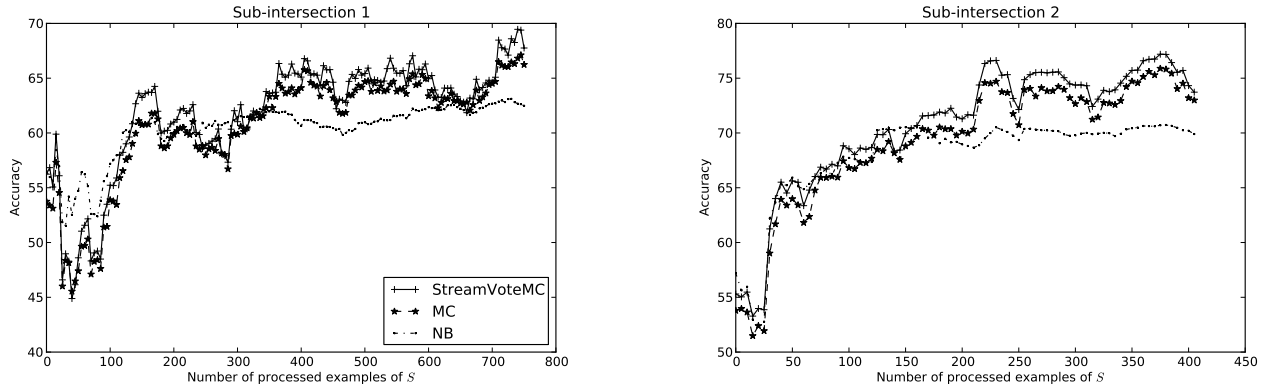

Fig. 9 Learning curves for the algorithms NB, SimpleMC and StreamVoteMC (averaged over the same 50 random drawings of L_{init} as the results in Figure 8): plot of the accuracy as a function of the number of processed instances from the stream *S* for sub-intersections 1 and 2 (respectively left and right).

References

1. Archer J (2004) Methods for the assessment and prediction of traffic safety at urban intersections and their application in micro-simulation modelling. Academic thesis, Royal Institute of Technology, Stockholm, Sweden, URL http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-143
2. Blake C, Merz C (1998) UCI repository of machine learning databases. URL http://www.ics.uci.edu/ mlearn/MLRepository.html
3. Blum A (1998) On-line algorithms in machine learning. In: Fiat A, Woeginger G (eds) Online Algorithms, Lecture Notes in Computer Science, vol 1442, Springer Berlin / Heidelberg, pp 306–325, URL 10.1007/BFb0029575
4. Boillot F, Midenet S, Pierrelée JC (2006) The real-time urban traffic control system cronos: Algorithm and experiments. Transportation Research Part C: Emerging Technologies 14(1):18–38, DOI 10.1016/j.trc.2006.05.001
5. Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140, URL citeseer.ist.psu.edu/breiman96bagging.html
6. Brown G, Wyatt J, Harris R, Yao X (2005) Ensemble diversity creation methods: A survey and categorisation. Information Fusion Journal (Special issue on Diversity in Multiple Classifier Systems) 6(1):5–20
7. Cohn D, Atlas L, Ladner RE (1994) Improving generalization with active learning. Machine Learning 15(2):201–221, URL citeseer.ist.psu.edu/cohn92improving.html
8. Dagan I, Engelson SP (1995) Committee-based sampling for training probabilistic classifiers. In: Proceedings of the 12^{th} International Conference on Machine Learning (ML-95), pp 150–157, URL citeseer.ist.psu.edu/17150.html
9. Dietterich TG (2002) The Handbook of Brain Theory and Neural Networks, Second edition, The MIT Press, Cambridge, MA, chap Ensemble Learning, pp 405–408
10. Dietterich TG (2002) Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science, vol 2396, Springer-Verlag, chap Machine Learning for Sequential Data: A Review, pp 15–30
11. Domingos P, Pazzani M (1997) On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning 29:103–130
12. Duda RO, Hart PE (2000) Pattern Classification. Wiley-Interscience
13. Freund Y (1995) Boosting a weak learning algorithm by majority. Information and Computation 121(2):256–285
14. Fürnkranz J (1998) Integrative windowing. Journal of Artificial Intelligence Research 8:129–164, DOI doi:10.1613/jair.487
15. Ho SS, Wechsler H (2004) Learning from data streams via online transduction. In: Ma S, Li T, Perng CS (eds) Workshop Proceedings, Temporal Data Mining: Algorithms, Theory and Applications, ICDM 2004, Brighton, pp 45–52
16. Hoare Z (2008) Landscapes of nave bayes classifiers. Pattern Analysis & Applications 11:59–72, DOI 10.1007/s10044-007-0079-5
17. Kittler J (1998) Combining classifiers: A theoretical framework. Pattern Analysis & Applications 1:18–27, DOI 10.1007/BF01238023
18. Kuncheva LI (2004) Classifier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (eds) Proceedings 5th International Workshop on Multiple Classifier Systems, MCS2004, Cagliari, Italy, Lecture Notes

in Computer Science, vol 3077, pp 1–15
19. Kuncheva LI (2005) Guest editorial. Information Fusion Journal (Special issue on Diversity in Multiple Classifier Systems) 6(1):3–4
20. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles. Machine Learning 51:181–207
21. Lee HKH, Clyde MA (2004) Lossless online bayesian bagging. Journal on Machine Learning Research 5:143–151
22. Lewis DD, Catlett J (1994) Heterogeneous uncertainty sampling for supervised learning. In: Cohen WW, Hirsh H (eds) Proceedings of ICML-94, 11th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US, New Brunswick, US, pp 148–156, URL citeseer.nj.nec.com/135290.html
23. Melville P, Mooney R (2004) Diverse ensembles for active learning. In: Proceedings of the 21st International Conference on Machine Learning (ICML-2004), Banff, Canada, pp 584–591, URL citeseer.ist.psu.edu/melville04diverse.html
24. Midenet S, Boillot F, Pierrelée JC (2004) Signalized intersection with real-time adaptative control: on-field assessment of CO_2 and pollutant emission reduction. Transportation Research Part D: Transport and Environment 9:29–47, DOI 10.1016/S1361-9209(03)00044-0
25. Midenet S, Saunier N, Boillot F (2011) Exposure to lateral collision in signalized intersections with protected left turn under different traffic control strategies. Accident Analysis & Prevention 43:1968–1978, DOI 10.1016/j.aap.2011.05.015
26. Oza N (2001) Online ensemble learning. PhD thesis, University of California, Berkeley
27. Polikar R (2006) Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3):21–45, DOI 10.1109/MCAS.2006.1688199
28. Saunier N, Midenet S (2010) Automatic Estimation of the Exposure to Lateral Collision in Signalized Intersections using Video Sensors. Tech. rep., arXiv, URL http://arxiv.org/abs/1012.4776v1
29. Saunier N, Midenet S, Grumbach A (2003) Automatic detection of vehicle interactions in a signalized intersection. In: 16^{th} International Cooperation on Theories and Concepts in Traffic Safety Workshop, Soesterberg, The Netherlands, URL http://www.ictct.org/Workshops/03-Soesterberg/Saunier.pdf
30. Saunier N, Midenet S, Grumbach A (2004) Stream-based Learning through Data Selection in a Road Safety Application. In: Onaindia E, Staab S (eds) STAIRS 2004, Proceedings of the Second Starting AI Researchers' Symposium, IOS Press, Valencia, Spain, Frontiers in Artificial Intelligence and Applications, vol 109, pp 107–117
31. Saunier N, Sayed T, Ismail K (2010) Large scale automated analysis of vehicle interactions and collisions. Transportation Research Record: Journal of the Transportation Research Board 2147:42–50, DOI 10.3141/2147-06
32. Schohn G, Cohn D (2000) Less is more: Active learning with support vector machines. In: Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, pp 839–846, URL citeseer.nj.nec.com/schohn00less.html
33. Seung HS, Opper M, Sompolinsky H (1992) Query by committee. In: Computational Learning Theory, pp 287–294, URL citeseer.nj.nec.com/seung92query.html
34. Tong S (2001) Active learning: Theory and applications. PhD thesis, Department of Computer Science of Stanford University
35. Tumer K, Oza NC (2003) Input decimated ensembles. Pattern Analysis & Applications 6:65–77, DOI 10.1007/s10044-002-0181-7
36. Utgoff PE (1989) Incremental induction of decision trees. Machine Learning 4:161–186
37. Witten IH, Frank E (2005) Data Mining: Practical machine learning tools and techniques, 2^{nd} edn. Morgan Kaufmann, San Francisco