Journal of Big Data
a SpringerOpen Journal

CASE STUDY Open Access

Comparative study between incremental and ensemble learning on data streams: Case study

Wenyu Zang[1,2*], Peng Zhang[2], Chuan Zhou[2] and Li Guo[2]

*Correspondence: zangwenyu@software.ict.ac.cn
[1] Institute of Computing Technology, Chinese Academy of Science, South Road of Chinese Academy of Science, Beijing, China
[2] Institute of Information Engineering, Chinese Academy of Science, MinZhuang Road, Beijing, China

Abstract

With unlimited growth of real-world data size and increasing requirement of real-time processing, immediate processing of big stream data has become an urgent problem. In stream data, hidden patterns commonly evolve over time (i.e.,concept drift), where many dynamic learning strategies have been proposed, such as the incremental learning and ensemble learning. To the best of our knowledge, there is no work systematically compare these two methods. In this paper we conduct comparative study between theses two learning methods. We first introduce the concept of "concept drift", and propose how to quantitatively measure it. Then, we recall the history of incremental learning and ensemble learning, introducing milestones of their developments. In experiments, we comprehensively compare and analyze their performances *w.r.t.* accuracy and time efficiency, under various concept drift scenarios. We conclude with several future possible research problems.

Keywords: Incremental learning; Ensemble learning; Concept drift; Big data streams

Background

We are now entering the era of big data. In government, business and industry domains, big data are generated rapidly and steadily, with a constant growth speed at a magnitude of million records per day. Moreover, these data are often related in temporal and spatial correlations. Typical examples include the wireless sensor data, RFID data and Web traffic data. These data often arrives unboundedly and rapidly, which forms a new class of data called "big stream data".

The focus on learning from big stream data is how to addressing the concept drifting challenge. Concept drift was first introduced by Wdimer and Kubat [1], where they noticed that the concept (the classification boundary or clustering centers) continuously changes with time elapsing. Based on the changing speed of concept, we formally divide the concept drifting into loose concept drift and rigorous concept drift [2]. In the former, concepts in adjacent data chunks are sufficiently close to each other; in the latter, genuine concepts in adjacent data chunks may randomly and rapidly changed.

Incremental learning [3] and ensemble learning [4] are two fundamental methods in learning from big stream data with concept drift. Incremental learning follows a machine learning paradigm where the learning process taking place whenever new examples emerge, and then adjusts to what has been learned from the new examples. While the ensemble learning employs multiple base learners and combines their predictions. The fundamental principle of dynamic ensemble learning is to dividing large data-stream into small data chunks and training classifiers on each data chunk independently. The most prominent difference of incremental learning from traditional machine learning is that incremental learning does not assume the availability of a sufficient training set before the learning process, but the training example appears over time. Moreover, the biggest difference between incremental learning and ensemble learning is that ensemble learning may discard training data outdated but incremental learning may not.

Although these two types of methods have their own strengths in data streams mining. However, the comparisons between them are rare. A.Tsymbal [5] described some types of concept drift and related works to handle it. Nevertheless, it not clearly categorizes the incremental and ensemble learning algorithms. In addition, they did no experiments on different learning framework.

In this paper we comparative study the incremental learning and ensemble learning algorithms. In addition, we compare performance between them in both accuracy and efficiency. Furthermore, some suggestions are given for choosing a better classifier.

This paper is organized as follows. In section Incremental learning, review and summarize the incremental learning algorithms. In section Ensemble learning, ensemble learning algorithms are learned and classified. In section Experiment results, incremental learning and ensemble learning algorithms are analysis and compare in a unified standard. The experiment results and discussions are given in section Conclusion and section 6.

Incremental learning

Generally, classification problem is defined as follows. A set of N training examples of the form (x, y) is given, where y is a discrete class label and x is a vector of d attributes (each of which may be symbolic or numeric). The goal is to produce from these examples a model $y = f(x)$ which will predict the classes y of future examples x with high accuracy.

To solve this problem, traditional statistic analysis method would load all training data into memory at once. However, compared to the explosive growth of today's information, the storage capacity is far from desirable. Moreover, when it comes to temporal series traditional data mining algorithms have showed limitations. Incremental learning algorithms are efficient method to these problems.

According to the differences of basic data learning method, incremental learning method can be sorted as there categories: incremental decision tree, incremental Bayesian and incremental SVM. According to the number of new instances to be added in a model at a time, it can be sorted as instance-by-instance learning and block-by-block learning.

Incremental decision tree

VFDT (very fast decision tree) [6] and CVFDT (concept-adapting very fast decision tree) [7] are two classical and impactive algorithms in incremental decision tree algorithms.

VFDT (very fast decision tree) Algorithm was first proposed by Domingos and Hulte in 2000. The author used hoeffding bounds verified that we can use a small sample of the available examples when choosing the split attribute at any given node and the output is asymptotically nearly identical to that of a conventional learner.

According to Hoeffding bounds, n independent observations of a real-valued random variable r with range R, with confidence $1 - \delta$, the true mean of r is at least $\overline{r} - \varepsilon$, where $\overline{r}$ is the observed mean of the samples and

$$\varepsilon = \sqrt{\frac{R^2 ln\left(\frac{1}{\delta}\right)}{2n}} \tag{1}$$

Select $G(X_i)$ be the heuristic measure used to choose test attributes. Let X_a be the attribute with best heuristic measure and X_a be the second best attribute. Let $\triangle\overline{G} = G(X_a) - G(X_b)$. Applying the Hoeffding bound to $\triangle\overline{G}$, if $\triangle\overline{G} > \varepsilon$, we can confidently select X_a as the split attributes. So VFDT is a real-time system and able to learn from large amount of data within practical time and memory constraints.

But comes to rigorous concept drift, VFDT has its own limitations. In order to solve this problem, Hulten and Spencer proposed CVFDT (concept-adapting very fast decision tree) algorithm [7] in 2001 based on VFDT. In CVFDT, each internal node has a list of alternate sub-trees being considered as replacements for the sub-tree rooted at the node. It also supports a parameter which limits the total number of alternate trees being grown at any one time. Each node with a non-empty set of alternate sub-trees,l_{test}, enters a testing mode to determine if it should be replaced by one of its alternate sub-trees.l_{test} collects the next m training examples that arrives to compare the accuracy of the sub-tree it roots with the accuracies of all of its alternative sub-trees. If the most accurate alternate sub-tree is more accurate than the l_{test}, l_{test} is replaced by the alternate. CVFDT also prunes alternate sub-trees during the test phase. For each alternative sub-tree of l_{test}, l^i_{all}, CVFDT remembers the smallest accuracy difference ever achieved between the two, $\triangle min\left(l_{test}, l^i_{all}\right)$. CVFDT prunes any alternate whose current test phase accuracy difference is at least $\triangle min\left(l_{test}, l^i_{all}\right) + 1\%$. By this means of sub-tree, CVFDT can adapt itself to concept drift well than VFDT.

In summary these two algorithms are both real-time method for data-stream mining. CVFDT is faster than VFDT and also adapts better to concept drift. While VFDT cost less memory than CVFDT.

Incremental Bayesian algorithm

Besides the advantages such as feasible, accurate and fast shared by all incremental learning algorithms, Incremental Bayesian Algorithms [8-10] can handle training instances without labels. Generally speaking, Bayesian Algorithm implement incremental learning by constantly updating the priori probability according to incoming training instants. As it illustrates in Figures 1 and 2.

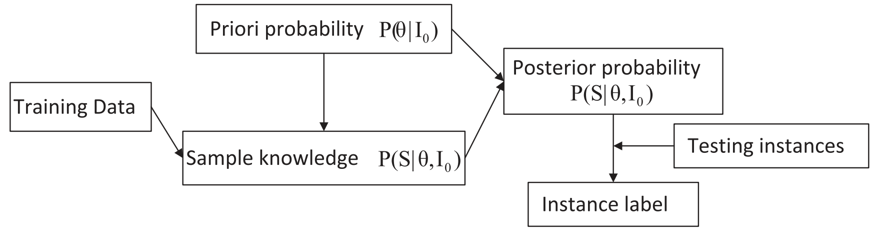
Priori probability $P(\theta|I_0)$
Posterior probability $P(S|\theta,I_0)$
Training Data
Sample knowledge $P(S|\theta,I_0)$
Testing instances
Instance label

Figure 1 Traditional Bayesian algorithm.

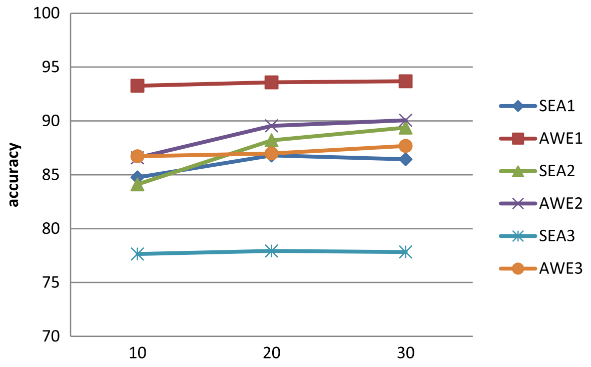

Figure 2 Incremental Bayesian algorithm.

In Bayesian algorithm priori probability $P(\theta|S, I_0)$ is a known quantity. While in the incremental Bayesian the priori probability change into $P(\theta|S, I_0)$ considering incoming new training instances. What we are concerning with is how to update a priori probability incrementally.

Firstly, make the following stipulation to some marks. The sample's space S is composed of attribute space I and class space C. Which is denote $S = \{S_1, S-2, \ldots, \S_n\} = < I, C >$. Each sample $S_i = \{a_1, a_2, \ldots, a_m, c_l\}$, the attribute is denoted by A_i, whose value is $\{a_{ik}\}$, and class attribute C is composed by I discrete values $(c_1, c_2, \ldots, c_l)$. The task of classifier is to learning the attribute space I and class space C, then finding out the mapping relation between them. Only one c_i in class attribute set $C = (c_1, c_2, \ldots, c_l)$ will be found to correspond given any one sample $s_i = \{a_1, a_2, \ldots, a_m\} \in I$. That is to say existing only one c_i for each instance $x = (a_1, a_2, \ldots, a_m) \in I$, let $P(c = c_i|x) \geq \left(j = 1, 2, \ldots, l\right)$.

For the training samples $D = \{x_1, x_2, \ldots, x_n\}$, assume that the priori probability follows dirichlet distribution. We can estimate the parameters as follows.

$$\theta_{ik|r} = P(A_{ik}|c_r; \theta) = \frac{1 + count(A_{ik} \wedge c_r)}{|A_i| + count\,(c_r)} \tag{2}$$

$$\theta_r = P(c_r|\theta) = \frac{1 + count\,(c_r)}{|C| + |D|} \tag{3}$$

Where A_{ik} is the k_{th} value of attribute A_i, $|A_i|$ is the number of values in attribute A_i. $|D|$ is the size of training samples.

According to incoming instances $T = \{x'_1, x'_2, \ldots, x'_m\}$, we consider two different situations: labeled instances and unlabeled instances. For labeled instances, we can update the parameters as follows:

$$\theta'_{ik|r} = P(A_{ik}|c_r; \theta')$$

$$= \frac{1 + count\,(A_{ik} \wedge c_r) + count'\,(A_{ik} \wedge c_r)}{|A_i| + count\,(c_r) + count'\,(c_r)} \tag{4}$$

$$\theta'_r = P(c_r|\theta') = \frac{1 + count\,(c_r) + count'\,(c_r)}{|C| + |D| + |D'|} \tag{5}$$

For the unlabeled instances, we can update the parameters as follows:

$$\theta'_r = \begin{cases} P(c_r|\theta') = \frac{\delta}{1+\delta}\theta_r & c_r \neq c'_p \\ P(c_r|\theta') = \frac{\delta}{1+\delta}\theta_r + \frac{\delta}{1+\delta} & c_r = c'_p \end{cases} \tag{6}$$

Where $\delta = |C| + |D|$

$$\theta'_{ik|r} = \begin{cases} P\left(A_{ik}|c_r;\theta'\right) = \frac{\delta}{1+\delta}\theta_{ik|r} & c_r = c'_p \wedge A_{ik} \neq A'_{ip} \\ P\left(A_{ik}|c_r;\theta'\right) = \frac{\delta}{1+\delta}\theta_{ik|r} + \frac{\delta}{1+\delta} & c_r \neq c'_p \wedge A_{ik} \neq A'_{ip} \\ \theta_{ik|r} & c_r \neq c'_p \end{cases} \tag{7}$$

In summary, Bayesian Algorithm itself has incremental property. For the incoming training instances with labels, it is easy to complement an incremental algorithm. Otherwise, with instances without labels, we discusses the sampling policy and various classifying loss expressions to simplifies and improves the classifiers.

Incremental SVM

The two core concepts of SVM algorithm are mapping input vectors into a high dimensional feature space and structural risk minimization. There is a useful property in SVM algorithm: classification equivalence on SV set and the whole training set. Based on this property, Incremental SVM [11-18] can be trained by preserving only the SVs at each step, and add them to the training set for the next step. According to different situations, there are different ways to select training set at each step.

The problems discussed in Incremental SVM algorithm are how to discarding history samples optimally and how to selecting new training instances in successive learning procedure. But there is still some intrinsic difficulties. Firstly, Support vectors (SVs) is highly depended on kernel functions you selected. Secondly, when concept drift happens, previous support vectors could be useless.

Decision tree algorithms, Bayesian learning algorithms and SVM algorithms are three main algorithms in data mining. The problem we discussed in incremental algorithm is how to using old training result accelerating the successive learning procedure. Incremental decision tree (hoffding tree or VFDT) uses a statistic result (hoffding bounds) to guaranteeing that we can learn from abundant data within practical time and memory constraints. Incremental Bayesian algorithm updates the prior probability dynamically according to the incoming instances. Incremental SVM is based on the classification equivalence of SV set and the whole training set. So we can add only support vectors (SVs) to the incoming training set for incrementally training a new model. In these three algorithms, Incremental decision tree and Incremental Bayesian algorithms are based on experience risk minimization. While Incremental SVM is based on structural risk minimization. Incremental decision tree and Incremental Bayesian algorithm is faster and Incremental SVM algorithm has better a generalization ability.

All of these algorithms above update a classifier dynamically using the new coming data. On one hand, we need not to load all data into memory at once. On the other hand, we can real-time modify the classification model according to the new training instances. Moreover, the classifier can adapt to concept drift via real-time updating to new data. However, there are still shortcomings and limitations in incremental learning algorithms. For example, it can only unceasing absorb new data-streams, it cannot remove old instances in the classification model. Because of these shortcomings, incremental algorithms will be helpless when comes to rigorous concept drift.

Ensemble learning

The fundamental principle of dynamic ensemble learning is to dividing large data-stream into small data chunks. Then training classifiers on each data chunk independently. Finally, it develops heuristic rules to organize these classifiers into one super classifier.

This structure has many advantages. Firstly, each data chunk is relatively small so that the cost of training a classifier on it is not high. Secondly, we saved a well trained classifier instead of the whole instances in the data chunk which cost much less memory. Thirdly, it can adapt to various concept drifts via different weighing policies. So the dynamic ensemble learning models can cope with both unlimited increasing amounts of data and concept drift problems in data-stream mining.

There are many heuristic algorithms for ensemble learning. According to the ways of forming the base classifiers, it can be roughly divided into two classes: horizontal ensemble framework and vertical ensemble framework.

Horizontal ensemble framework

Horizontal ensemble framework tends to selecting the same type of classifiers and train them independently on different data-chunks, then using a heuristic algorithm to organize them together. It can be illustrated in Figure 3.

In this framework, almost all researches develop center on three issues: weighting policy, data selection and the choice of base classifiers. It can be formulized as:

$$f_{HE} = \Sigma_{i=1}^{N} \alpha_i f_i\left(x\right) \tag{8}$$

Where α_i is the weighting value assigned to the i_{th} data-chunk. $f_i\left(x\right)$ is the classifier trained on the i_{th} data-chunk. And the $1\ to\ N$ is the data-chunks selected.

Weighting policy is the most important method in ensemble learning to guarantee accuracy. Street [19] proposed a SEA algorithm, which combined all the decision tree models using majority-voting. In this algorithm $\alpha_i = \frac{1}{N}\left(i = 1, 2, \ldots, N\right)$. Kolter [20] also proposed a Dynamic Weighted Majority (DWM) algorithm. Yeon [21] proved majority-voting is the optimum solution in the case of no concept drift. In order to tracing the concept drift, Wang [22] proposed an accuracy-weighted ensemble algorithm, in which they assign each classifier a weight reversely proportional to the classifier's accuracy on the up-to-data chunk. In this algorithm $\alpha_i = -\left(MSE_i - MSE_r\right)$, where $MSE_i = \frac{1}{|S_n|}\Sigma_{(x,c)\in S_n}\left(1 - f_c^i\left(x\right)\right)^2$ is the mean square error of $f_i\left(x\right)$. S_n is the training set. $MSE_r = \Sigma_c p\left(c\right)\left(1 - p\left(c\right)\right)^2$ is the mean square error of a random classifier. C is the labels

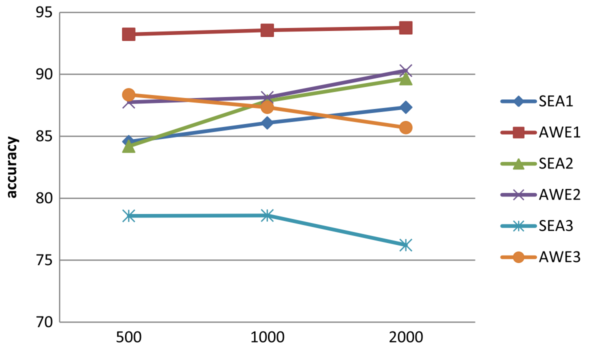

Figure 3 Horizontal ensemble framework.

of all instances. Tsymbal [5] proposed a dynamic integration of classifiers in which base classifier is given a weight proportional to its local accuracy. Zhang [23] develop a kernel mean matching (KMM) method to minimize the discrepancy of the data chunks in the kernel space for smooth concept drift and an Optimal Weight values for classifiers trained from the most recent data chunk for abrupt concept drift. Yeon [21] proposed an ensemble model has a form of a weighted average and ridge regression combiner. In this proposed algorithm a angle between the estimated weights and optimal weight is used to estimate concept drift, when concept drift is smooth $\alpha_i = \frac{1}{N}\,(i = 1, 2, \ldots, N)$ otherwise $\alpha_i = arg_w min \Sigma_{i=1}^{n} \left(y_i - \Sigma_{j=1}^{m} \alpha_j f_j\left(x_i\right)\right)^2 + \lambda \Sigma_{j=1}^{m} \alpha_j^2$ subject to $\Sigma_{j=1}^{m} \alpha_j = 1, \alpha_j > 0$ where y_i is the label of instance. m is the number of classifiers and n is the number of instances. In this algorithm a penalty coefficient is employed to trace different level of concept drift.

As to instance selection, weighted instance and data discarded policy et al. are discussed. Fan [24] proposed a benefit-based greedy approach which can safely remove more than 90% of the base models and guarantee the acceptable accuracy. Fan [25] proposed a simple, efficient and accurate cross-validation decision tree ensemble method to discard old data and combine with new data to construct the optimal model for evolving concept. Zhao [26] proposed a pruning method (PMEP) to obtain the ensembles at a proper size. Lu [27] proposed a heuristic metric that considers the trade-off in accuracy and diversity to select the top p percent of ensemble members, depending on their resource availability and tolerable waiting time. Kuncheva [28] proposed a concept of "forgetting" by ageing at a variable rate.

Vertical ensemble framework

Vertical ensemble framework tends to selecting different type of classifiers and training it independently on the up-to-data data-chunk. Then it uses a heuristic algorithm to organizing them together. This algorithm often uses in a situation of rigorous concept drift, with little or no correlation of the decision concepts between data chunk. It can be illustrated in Figure 4.

In this frame work, we focus more on classifier diversity and a suited weighting policy. It can be formulized as:

$$f_{VE}^{n}\left(x\right) = \Sigma_{i=1}^{m} \beta_i f_{in}\left(x\right) \tag{9}$$

Where β_i is the weighting value assigned to the i_{th} classifier. And $f_{in}\left(x\right)$ is the i_{th} classifier trained on the n_{th} data-chunk.

In vertical ensemble framework, classifier diversity is a primary factor to guarantee accuracy. Zhang [29] proposed a semi-supervised ensemble method: $U_D EED$. It works by maximizing accuracies of base learners on labeled data while maximizing diversity among

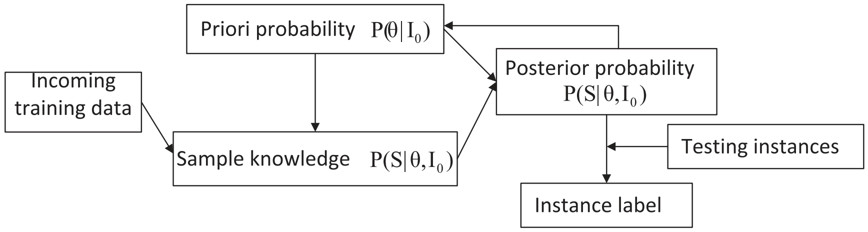
Priori probability $P(\theta|I_0)$
Incoming training data
Posterior probability $P(S|\theta,I_0)$
Sample knowledge $P(S|\theta,I_0)$
Testing instances
Instance label

Figure 4 Vertical ensemble framework.

them on unlabeled data. Zhang [2] proposed an Optimal Weight values for classifiers in the case of abrupt concept drift, in this algorithm all classifiers using different learning algorithms, e.g., Decision Tree, SVM, LR, and then builds prediction models on and only on the up-to-data data chunk. Minku [30] show that low diverse ensemble obtain low error in the case of smooth concept drift while high diverse ensemble is better when abrupt concept drift happens.

The weighting policy in horizontal framework is almost commonly used in the vertical framework. It is also method like voting majority, weighted based on accuracy and weighted through a regression algorithm and so on.

Horizontal ensemble frame work building classifiers on different data-chunks, in this way it robust to noisy stream and concept drift because the final decisions are based on the classifiers trained from different chunks. Even if noisy data chunks and concept drift may deteriorate some base classifiers, the ensemble can still maintain relatively stable prediction accuracy. While vertical ensemble framework building classifiers using different learning algorithms on the same data-chunk, in this way it can decrease the expected bias error compared to any single classifiers. When we have no prior knowledge on the incoming data, it is difficult to determine which type of classifier is better, so combining multiple types of classifiers is likely to be a better solution than simply choosing either of them. We can also aggregate these two frameworks together. We can combine these base classifiers to form an aggregate ensemble through model defined in Eq. 10.

$$f_{AB} = \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} \alpha_i \beta_j f_{ij}\left(x\right) \tag{10}$$

In a word, the core idea of ensemble learning is to organizing different weak classifiers into one strong classifier. The main method used in ensemble learning is divide-and-conquer. In ensemble learning large data-stream is divided into small data-chunks, and we train classifiers on each chunk independently. The difficult problems we discussed mostly in ensemble learning are as follows. First, what base classifier should we choose? Second, how to set the size of a data-chunk? Third, how to assign weighting values to different classifiers? Finally, how to discard previous data? As to setting the size of a data-chunk, large data-chunk is more robust while small data-chunk adapts better to concept drift. And the weighting policy direct influence on accuracy.

Experiment results

The aim of the experiments is to comparing the incremental learning with the ensemble learning algorithms. In incremental learning algorithms incremental decision tree (include VFDT and CVFDT), incremental Bayesian algorithm and incremental SVM were experimental verified. In ensemble learning algorithms horizontal framework and vertical ensemble framework were implemented. AWE was chosen to represent horizontal ensemble framework. In all the compared algorithms we compare basic characteristics on popular synthetic and real life data sets.

All of the tested algorithms were implemented in Java as part of the MOA and Weka framework. We implemented the AWE algorithms and implement incremental SVM in Libsvm, while all the other algorithms were already a part of MOA or Weka. The experiments were done on a machine equipped with an AMD Athlon (tm) II X3 435 @2.89 GHz Processor and 3.25 GB of RAM. To make the experimental more reliable, we experiment every algorithm on each data stream (from different starting point) for 10 times and calculated the mean and variance based on these values in the experimental. T-test was used for Significance Testing. Classification accuracy was calculated using the data block evaluation method, which works similarly to test-then-train paradigm. This method reads incoming examples without processing them, until they form a data block of size d. Each new data block is first used to test the existing classifier, and then it updates the classifier.

Synthetic and real data streams in experiment

In this part all five data-streams used in the experiment will be listed. There are four synthetic data-streams (Hyperplane 1, Hyperplane 2, Hyperplane 3 and KDDcup99) and one real data-streams (sensor data-stream).

In these data-streams Hyperplane 1, Hyperplane 2 and Hyperplane 3 are generated by Hyperplane generator in moa. They all have 9 attributes and one label with 2 classes, and there are 800,000 instances in each of the data-streams. The difference between these three synthetic data-streams is that they have different level of concept drifts. Hyperplane 1 has no concept drift. Hyperplane 2 has median level of concept drift and Hyper plane 3 has abrupt concept drift. Kddcup99 stream was collected from the KDD CUP challenge in 1999, and the task is to build predictive models capable of distinguishing between intrusions and normal connections. Clearly, the instances in the stream do not flow in similar way as the genuine stream data. In this data-stream each instance has 41 attributes and one label with 23 classes. Sensor stream contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley Research Lab. The whole stream contains consecutive information recorded over a 2 months period (1 reading per 1-3 minutes). sensor ID is used as the class label, so the learning task of the stream is to correctly identify the sensor ID (1 out of 54 sensors) purely based on the sensor data and the corresponding recording time. While the data stream flow over time, so does the concepts underlying the stream. For example, the lighting during the working hours is generally stronger than the night, and the temperature of specific sensors (conference room) may regularly rise during the meetings. So there are 5 attributes and a label of 54 classes in this data-stream.

In order to make a visual representation of the concept drift, we divided these data-streams into small data-chunks. And then, we train a C4.5 decision tree on the first data-chunk. Next, we use this classifier predict the labels of the following data-chunks and record the accuracy. If there is no concept drift, the accuracies will be stable. Otherwise the accuracies will changes dramatically.

As it show in Figure 5, we can see that KDDcup99 and HGstream1 data streams have no concept drift. The sensor data stream has the most rigorous concept drift. HGstream

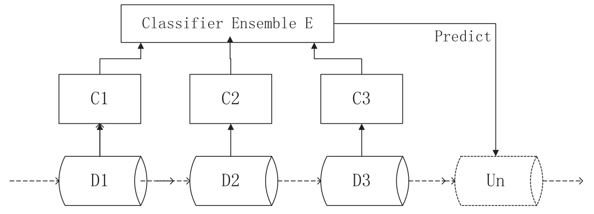

Figure 5 Different level of concept drift.

Table 1 The accuracy of four kinds of incremental learning algorithms

	Hyperplane 1	Hyperplane 2	Hyperplane 3	Sensor	KDDcup99
VFDT	$90.42_{\pm 0.13}$	$78.93_{\pm 1.7}$	$82.73_{\pm 3.08}$	$92.21_{\pm 2.19}$	$99.69_{\pm 0.01}$
CVFDT	$90.44_{\pm 0.14}$	$80.22_{\pm 1.55}$	$84.51_{\pm 2.68}$	$92.22_{\pm 2.12}$	
Incremental Bayesian	$93.8_{\pm 0.001}$	$73.54_{\pm 2.89}$	$81.65_{\pm 0.023}$	$93.29_{\pm 0.22}$	$98.50_{\pm 0.002}$
Incremental SVM	$90.8_{\pm 0.14}$	$70.5_{\pm 3.79}$	$80.12_{\pm 2.96}$	$91.89_{\pm 1.21}$	$97.96_{\pm 0.97}$

2 and HGstream 3 have median level of concept drift. While HGstream2 has a relatively rigorous concept drift and HGtream3 has a relatively loose concpet dirft.

Competitive study

Incremental learning and ensemble learning are two major solutions to large-scale data and concept drift in big stream data mining. Incremental learning is a style of learning where the learner updates its model of the environment where a new significant experience becomes available. And ensemble learning adopts a divide-and-conquer method to organize different base classifiers into one super classifier. They both can handle infinitely increasing amount of data and time series. Moreover, they both meet the real-time demands. Besides the above advantages they shared together each algorithm has its own relative merits. It will be discussed in detail in the followings.

Competitive study on accuracy and efficiency

Incremental learning and ensemble learning are two major solutions to large-scale data and concept drift in today's data stream mining. Incremental learning is a style of learning where the learner updates its model of the environment where a new significant experience becomes available. And ensemble learning adopts a divide-and-conquer method to organize different base classifiers into one super classifier. They both can handle infinitely increasing amount of data and time series. Moreover, they both meet the real-time demands. Besides the above advantages they shared together each algorithm has its own relative merits. It will be discussed in detail in the followings.

Competitive study on various concept drift

Incremental algorithm cannot adapt well to sudden concept drift. That is because almost of the incremental algorithms update its model according to incoming data-streams but it never discard history knowledge. For examples, in incremental Bayesian algorithms, priori probability is updated smoothly according to incoming instances. In incremental

Table 2 The mean of the accuracy on data stream Hyperplane1

	Data-chunk size	500	1000	2000
Classifier number	Algorithm			
10	SEA	$83.66_{\pm 2.49}$	$84.23_{\pm 2.01}$	$86.39_{\pm 0.35}$
	AWE	$92.88_{\pm 0.11}$	$93.34_{\pm 0.05}$	$93.61_{\pm 0.10}$
20	SEA	$85.82_{\pm 3.14}$	$87.06_{\pm 0.53}$	$87.49_{\pm 0.79}$
	AWE	$93.33_{\pm 0.12}$	$93.63_{\pm 0.06}$	$93.79_{\pm 0.09}$
30	SEA	$84.23_{\pm 2.01}$	$86.94_{\pm 1.62}$	$88.14_{\pm 0.27}$
	AWE	$93.49_{\pm 0.10}$	$93.70_{\pm 0.06}$	$93.85_{\pm 0.08}$

SEA get the optimal accuracy at a 2,000 data chunk size and 30 base classifiers, while AWE algorithm get optimal accuracy at a 2,000 data chunk size and 30 base classifiers.

Table 3 The mean of the accuracy on data stream Hyperplane2

	Data-chunk size	500	1000	2000
Classifier number	Algorithm			
10	SEA	$77.06_{\pm 0.97}$	$87.91_{\pm 1.97}$	$87.36_{\pm 0.78}$
	AWE	$84.94_{\pm 3.87}$	$85.72_{\pm 0.53}$	$89.09_{\pm .012}$
20	SEA	$86.32_{\pm 1.21}$	$87.17_{\pm 0.99}$	$91.15_{\pm 0.29}$
	AWE	$87.96_{\pm 0.52}$	$89.27_{\pm 0.96}$	$91.42_{\pm 0.26}$
30	SEA	$89.22_{\pm 1.56}$	$88.49_{\pm 0.5}$	$90.44_{\pm 0.08}$
	AWE	$90.36_{\pm 0.43}$	$89.44_{\pm 0.46}$	$90.39_{\pm 0.04}$

SEA get the optimal accuracy at a 2,000 data chunk size and 20 base classifiers, while AWE algorithm get optimal accuracy at a 2,000 data chunk size and 20 base classifiers.

SVM algorithms, support vectors (SVs) are directly related to decision plane and kernel function. So it is very sensitive to concept drift. Only CVFDT an incremental decision algorithm can process time-changing concept by growing an alternative sub-tree. But it costs additional space to save alternative paths which decrease its efficiency dramatically.

In compared with incremental algorithms, ensemble learning algorithms is more flexible to concept drift. Firstly, it can set the size of data chunk to fit different level of concept drift: small data chunk for sudden concept drift and large data chunk for smooth concept drift. Secondly, it can assign different weighting values to different base classifiers to satisfy various concept drift. Thirdly different policy to select and discard base classifiers also helped.

As a result, ensemble learning algorithms adapt much better to concept drift than incremental learning algorithms.

Generally speaking, incremental algorithms is faster and has better anti-noise capacity than ensemble algorithms. While ensemble algorithms is more flexible and adapt itself better to concept drift. Moreover, incremental algorithms has more restrictions than ensemble algorithms. Not all classification algorithms can be used in incremental learning, but almost every classification algorithms can be used in an ensemble algorithms.

Therefore, when there is no concept drift or concept drift is smooth, an incremental algorithm is recommended. While huge concept drift or abrupt concept drift exist, ensemble algorithms are recommended to guarantee accuracy. Otherwise, in case of relatively simple data-stream or a high level of real-time processing is demanded incremental learning is a better choice. And in case of complicated or unknown distribution data-stream ensemble learning is a better choice.

Table 4 The mean of the accuracy on data stream Hyperplane3

	Data-chunk size	500	1000	2000
Classifier number	Algorithm			
10	SEA	$77.08_{\pm 9.59}$	$78.44_{\pm 11.07}$	$77.42_{\pm 4.03}$
	AWE	$88.04_{\pm 2.52}$	$86.99_{\pm 2.97}$	$85.13_{\pm 2.20}$
20	SEA	$79.4_{\pm 18.41}$	$78.58_{\pm 5.78}$	$75.8_{\pm 4.98}$
	AWE	$88.26_{\pm 2.61}$	$87.17_{\pm 3.12}$	$85.53_{\pm 2.04}$
30	SEA	$79.26_{\pm 6.22}$	$78.82_{\pm 8.45}$	$75.44_{\pm 1.84}$
	AWE	$88.75_{\pm 2.36}$	$87.86_{\pm 2.70}$	$86.44_{\pm 1.84}$

SEA get the optimal accuracy at a 500 data chunk size and 20 base classifiers, while AWE algorithm get optimal accuracy at a 500 data chunk size and 30 base classifiers.

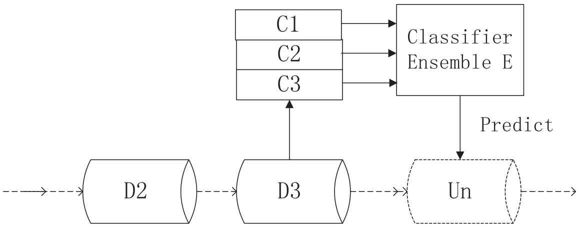

Figure 6 Different accuracy on number of classifiers.

Experiments on incremental algorithms

In this part we will experiment on different incremental algorithms.

Table 1 shows the accuracy of four kinds of incremental learning algorithms: VFDT (very fast decision tree), CVFDT (concept-adapting very fast decision tree), incremental bayesian algorithm and incremental SVM.We can see that the accuracy is decreased as the concept drift increase. CVFDT relatively adapts better to concept drift, but we can see in Table 1 when there is a large number of attributes in the data sets (KDDcup99) CVFDT can not work properly. In a word, majority of incremental algorithms can meet the requirement of real-time processing but not adapt well to abrupt concept drift.

Experiments on ensemble algorithms

In this section, horizontal ensemble framework is firstly discussed. And then we talked about vertical ensemble framework. In the end we compared these two ensemble frameworks.

Table 2, Table 3 and Table 4 show the relations between the size of data chunk, number of classifiers and classification accuracy on different concept drift. two kinds of most popular and most representative horizontal framework ensemble algorithms are tested. In all ensemble classifiers decision tree is selected as base classifier. We can point out that in a smooth concept drift we tend to select a relatively large data chunk and small size of classifiers while in the case of abrupt concept drift a small data chunk is better.

In these tables we can see that AWE algorithm is better than SEA algorithm, especially in case of concept drift (HGstream3). Moreover, we can see that different weighting policy directly lead to different accuracy on test instances. And many papers about ensemble algorithms are discussed on different weighting policies. Beside weighting policy, data

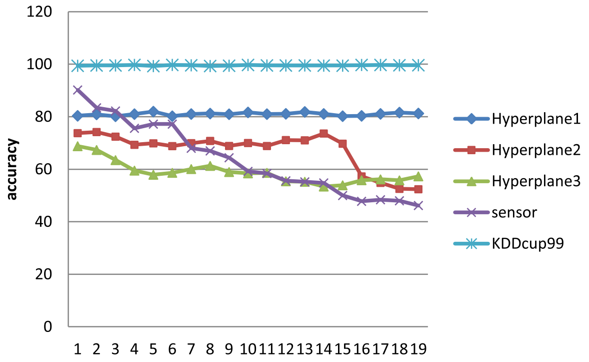

Figure 7 Different accuracy on size of data chunk.

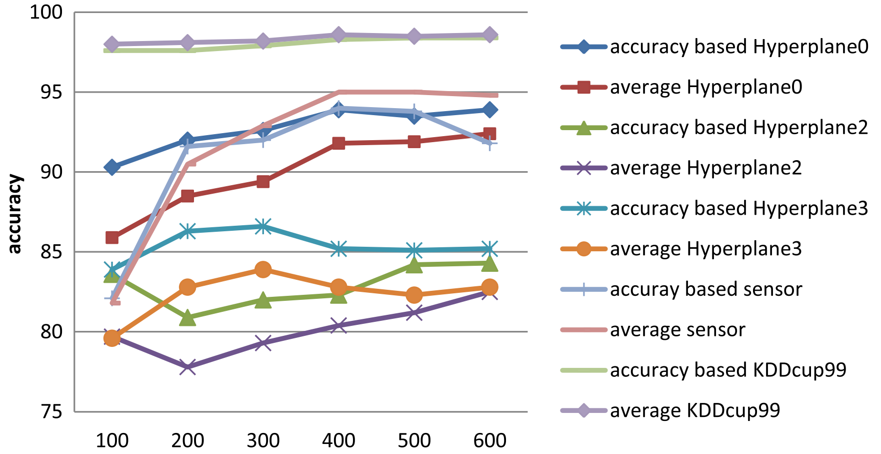

Figure 8 Different variance on number of classifiers.

chunk size and classifier number are other two influential factors on the performance of ensemble algorithms.

Figure 6 shows the algorithm accuracy on different number of classifiers. We can see that in a case of little or no concept drift more classifiers is better. While in a case of abrupt concept drift less classifiers is better. But the difference of this influential factor is not that obvious.

Figure 7 shows the algorithm accuracy on different size of data chunk. We can see that in a case of little or no concept drift a large data chunk is better and in a case of concept drift small data chunk is better. The influence of data chunk size is obvious. In both Figure 6 and Figure 7 we can see that data stream with less concept drift have better performance.

Figure 8 shows the algorithm variance on different number of classifiers. We can see that the influence of classifier numbers is not that obvious. Only in the case of high level concept drift, we can see that more base classifiers more stable.

Figure 9 shows the algorithm variance on different size of data chunk. We can see that the bigger the data chunk is the more table the algorithm performance is. And this tendency is very obvious. In both Figure 8 and Figure 9 we can see that AWE algorithm is more stable than SEA algorithm. And less concept drift directly lead to better performance. Meanwhile the influence on variance is more obvious than that on accuracy.

As show in Figure 10, we can see that besides in data stream KDDcup99, in data stream sensor vertical ensemble has better performance than in other data-streams. KDDcup99

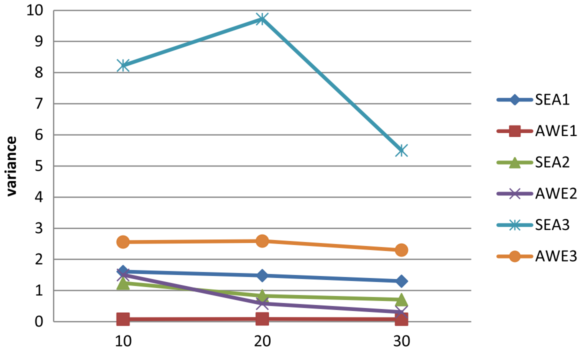

Figure 9 Different variance on size of data chunk.

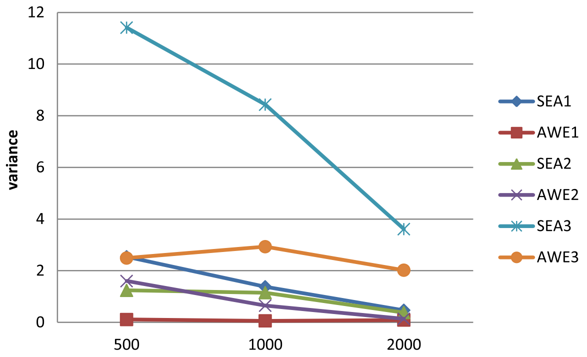

Figure 10 Different accuracy on size of data chunk for all datasets.

is a data-stream every classify algorithm will achieve a outstanding result and sensor is a data-stream with the highest level of concept drift. That is to say in a case of large concept drift vertical ensemble algorithm is a better choice. And generally in vertical ensemble algorithms we tend to selected a data chunk not larger than 1000 instances. Moreover a data chunk less than 500 instances is not stable and can not achieve a good performance.

As we show in Table 5 we can see that horizontal ensemble framework do a better job when concept drift is relatively smooth while the vertical do a better job in the case of abrupt concept drift. we also can regard the vertical ensemble framework as an extreme case of horizontal ensemble framework where only one base classifier is selected and trained on the latest data chunk.

Experiments on competitive learning

In this section, we will competitively discussed the advantages and disadvantages between incremental and ensemble algorithms. Consider the comparability, in each algorithm we selected decision tree as a base classifier. So we chose VFDT as representative of incremental algorithm and we used a accuracy based weighting algorithm in both horizontal ensemble and vertical ensemble algorithms.

Table 6 shows that ensemble algorithm is more accuracy than incremental algorithm. And in a case of high level concept drift, vertical ensemble algorithm has better performance while in smooth concept drift or no concept drift horizontal ensemble algorithm is better. But in a case when a single classifier also can perform very well in classification the ensemble learning algorithm is not as good as incremental learning algorithms.

Table 7 shows the cost time of different algorithms. In all these data-streams Hyperplane 1, Hyperplane 2 and Hyperplane 3 contains 300,000 instances and sensor contains 10,000 instances. KDDcup99 contains 100,000 instances. We can see that incremental algorithm is obvious faster than ensemble algorithms. While horizontal ensemble and vertical ensemble algorithms has a similar cost time.

Table 5 Competitive study on accuracy

	Hyperplane 1	Hyperplane 2	Hyperplane 3	Sensor	KDDcup99
Average vote horizontal ensemble	$88.14_{\pm 0.27}$	$91.15_{\pm 0.29}$	$78.82_{\pm 8.45}$	$89.05_{\pm 3.85}$	$99.44_{\pm 0.01}$
Accuracy based horizontal ensemble	$93.85_{\pm 0.08}$	$91.42_{\pm 0.26}$	$88.75_{\pm 2.36}$	$87.34_{\pm 2.17}$	$99.31_{\pm 0.04}$
Average vote vertical ensemble	$92.4_{\pm 0.01}$	$82.5_{\pm 0.07}$	$83.9_{\pm 0.04}$	$95_{\pm 0.02}$	$98.6_{\pm 0.01}$
Accuracy based vertical ensemble	$93.6_{\pm 0.01}$	$84.3_{\pm 0.06}$	$86.6_{\pm 0.04}$	$94_{\pm 0.05}$	$98.4_{\pm 0.01}$

Table 6 The accuracy of different algorithms

	Hyperplane 1	Hyperplane 2	Hyperplane 3	Sensor	KDDcup99
VFDT	$90.42_{\pm 0.13}$	$78.93_{\pm 0.17}$	$82.73_{\pm 3.08}$	$92.21_{\pm 2.19}$	$99.69_{\pm 0.01}$
Horizontal ensemble	$93.85_{\pm 0.08}$	$86.28_{\pm 0.74}$	$88.75_{\pm 2.63}$	$87.34_{\pm 2.17}$	$99.31_{\pm 0.043}$
Vertical ensemble	$93.9_{\pm 0.013}$	$84.3_{\pm 0.67}$	$86.6_{\pm 0.04}$	$95.1_{\pm 0.35}$	$98.4_{\pm 0.005}$

In a word, we can say that ensemble learning is more accuracy than incremental learning algorithms and incremental algorithm is more efficiency than ensemble algorithms.

Conclusion

Unlimited growth of big stream data and concept drift has been two most difficult problems in data-stream mining. There are two mainstream solutions to these problems: incremental learning and ensemble learning algorithms. Incremental learning algorithms employ a method of updating a single model by incorporating newly arrived data. While ensemble learning algorithms use the divide-and-conquer method to cutting up large data into small data chunks and training classifiers on each data chunk independently, then a heuristic algorithm is used to ensemble these classifiers together. In incremental algorithms, we talked mostly about how to recording previous knowledge and adapting to new knowledge. In ensemble learning algorithms, we discuss mostly about how to making a weighting policy for each base classifiers.

Both of these algorithms can handle big stream data and concept drift problems, and each of them has its own properties. Incremental learning algorithms have better performance on efficiency and ensemble learning adapts better to concept drift. Moreover, ensemble learning algorithms are more stable than incremental algorithms. The size of data chunk is another important factor in ensemble algorithms, which influences the algorithm performance. Generally, a better way to achieving high accuracy is that the higher levels a concept drift is the smaller a data chunk will be.

Therefore, in a case of loose concept drift or no concept drift an incremental algorithm is recommended and in a case of rigorous concept drift an ensemble algorithm is a better choice. Otherwise, when efficiency is first considering factor we tend to selecting incremental algorithm and when accuracy is the most important factor we choose an ensemble algorithm. We can employ different algorithms according to the real data-stream distributions.

Weighting policy, instances selection, classifier diversity and so on is the main rules discussed in previously researches. With the very fast development of information industry, we have to face the reality of information explosion. In that situation, more and more classifiers will be trained and real-time processing will become a challenge. Therefore, the next step is how to effectively managing large amount of classifiers. We can consider some pruning method or index technology on the classifiers. We can also consider some parallel algorithms to organizing the classifiers.

Table 7 Cost time of different algorithms

	Hyperplane 1	Hyperplane 2	Hyperplane 3	Sensor	KDDcup99
VFDT	3254.9	3310.6	3246.9	203.2	3479
Horizontal ensemble	67760	67913	64237	5426	348114
Vertical ensemble	63145	59876	62110	5897	30146

Authors' contributions

WZ and PZ have made substantial contributions to conception and design. WZ has been involved in drafting the manuscript. PZ and CZ revising it critically for important intellectual content; LG has given final approval of the version to be published. All authors read and approved the final manuscript.

Acknowledgments

This work was supported by the NSFC (No. 61370025), 863 projects (No.2011AA01A103 and 2012AA012502), 973 project (No. 2013CB329605 and 2013CB329606), and the Strategic Leading Science and Technology Projects of Chinese Academy of Sciences (No.XDA06030200).

Received: 13 December 2013 Accepted: 20 January 2014
Published: 24 June 2014

References

1. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23(1): 69–101
2. Zhang P, Zhu X, Shi Y (2008) Categorizing and mining concept drifting data streams. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 812–820
3. Giraud-Carrier C (2000) A note on the utility of incremental learning. AI Commun 13(4): 215–223
4. Polikar R (2006) Ensemble based systems in decision making. Circ Syst Mag 6(3): 21–45
5. Tsymbal A (2004) The problem of concept drift: definitions and related work. Computer Science Department, Trinity College Dublin
6. Domingos P, Hulten G (2000) Mining high-speed data streams, pp 71–80
7. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams, pp 97–106
8. Hua C, Xiao-gang Z, Jing Z, Li-hua D (2009) A simplified learning algorithm of incremental Bayesian
9. Deng L, Droppo J, Acero A (2003) Incremental Bayes learning with prior evolution for tracking nonstationary noise statistics from noisy speech data
10. Alcobé JR (2004) Incremental augmented Naive Bayes classifiers 16: 539
11. Laskov P, Gehl C, Krüger S, Müller K-R (2006) Incremental support vector learning: analysis, implementation and applications. J Mach Learn Res 7: 1909–1936
12. Syed NA, Liu H, Huan S, Kah L, Sung K (1999) Handling concept drifts in incremental learning with support vector machines, pp 317–321
13. Fung G, Mangasarian OL (2002) Incremental support vector machine classification, pp 247–260
14. Zheng J, Yu H, Shen F, Zhao J (2010) An online incremental learning support vector machine for large-scale data. In: Artificial neural networks–ICANN, pp 76–81
15. Ruping S (2001) Incremental learning with support vector machines, pp 641–642
16. Xiao R, Wang J, Zhang F (2000) An approach to incremental SVM learning algorithm, pp 268–273
17. Tseng C-Y, Chen M-S (2009) Incremental SVM model for spam detection on dynamic email social networks 4: 128–135
18. Cauwenberghs G, Poggio T (2001) Incremental and decremental support vector machine learning. Advances in neural information processing systems: 409–415
19. Street WN, Kim Y A Streaming Ensemble Algorithm (SEA) for large-scale classification
20. Kolter JZ, Maloof MA (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift, pp 123–130
21. Yeon K, Song MS, Kim Y, Choi H, Park C (2010) Model averaging via penalized regression for tracking concept drift. J Comput Graph Stat 19(2)
22. Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers, pp 226–235
23. Zhang P, Zhu X, Shi Y (2008) Categorizing and mining concept drifting data streams, pp 812–820
24. Fan W, Chu F, Wang H, Yu PS (2002) Pruning and dynamic scheduling of cost-sensitive ensembles, pp 146–151. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, 1999
25. Fan W Systematic data selection to mine concept-drifting data streams
26. Zhao Q-L, Jiang Y-H, Xu M (2009) A fast ensemble pruning algorithm based on pattern mining process. Data Min Knowl Discov 19(2): 277–292
27. Lu Z, Wu X, Zhu X, Bongard J Ensemble pruning via individual contribution ordering
28. Kuncheva LI (2004) Classifier ensembles for changing environments. In: Multiple classifier systems, pp 1–15
29. Zhang P, Zhu X, Tan J, Guo L (2010) Classifier and cluster ensembles for mining concept drifting data streams, pp 1175–1180
30. Minku LL, White AP, Yao X (2010) The impact of diversity on online ensemble learning in the presence of concept drift. IEEE Trans Knowl Data Eng 5: 730–742

doi:10.1186/2196-1115-1-5
Cite this article as: Zang *et al.*: **Comparative study between incremental and ensemble learning on data streams: Case study.** *Journal of Big Data* 2014 **1**:5.