Détection de concept drift

Thèse de MASTER

Master Sciences, Technologies, Santé

Mention, Sciences et Technologies de l'Information et de la Communication

Spécialité, Systèmes Intelligents pour les Transports

Présentée et soutenue publiquement par :

Yunhui Hou

Le 13 Juillet 2012

À l'Université de Technologie de Compiègne.

Devant le jury :

Bertrand Ducourthial,	Professeur
Philippe Bonnifait,	Responsable de SIT
Pierre Morizet,	Responsable de master

Directeur de thèse : Vincent Lemaire

Stage effectué à Orange Labs, Lannion

Acknowledgement

I would like to thank all people who worked with me during my internship at Orange Labs, especially my supervisor Vincent Lemaire for inspiration and guide on methodology, and Christophe Salperwyck for helping me implementing our method on MOA.

Abstract:

Concept drift describes changes in statistical properties of incoming data instances for online classification over time. It causes a decrease in classifiers' accuracy as time passes (non stationary data stream). By monitoring online classifier's performance measures over time, we would like to detect concept drift using MODL discretization method on a mixture of two observation time windows: one representing properties of stationary state, the other representing current state. If MODL method is able to distingue measure values from the two windows, our method will detect a concept drift, otherwise the targeted data stream is still stationary.

Résumé:

En classification, les propriétés statistiques de la variable cible, que le modèle essaie de prédire, peuvent évaluer au cours du temps d'une manière imprévue. On parle alors de changement de concept (Concept drift) pose des problèmes parce que les prédictions deviennent moins exactes au fur et à mesure que le temps passe (données non stationnaires). En surveillant les mesures de performance en ligne du classifier, on applique la méthode de discrétisation MODL sur le mélange de deux fenêtres d'observation : l'une de référence représentant les propriétés historiques d'un état stationnaire, l'autre pour celles actuelles supposées être différentes s'il y a un concept drift. Si la discrétisation MODL réussit à discriminer/séparer les variables des deux fenêtres, notre méthode détectera un concept drift, sinon on reste à l'état stationnaire.

Contents

Notations

S, S_i data generating source

P probability distribution

n size of dataset

N training data set size

c_1 … c_k class labels

$X_i$attributes of ith instance in memory

X_t attributes of instance coming at time t

y_t, y_i class label/ attribute of X_t, X_i

W number of “recent” examples (time window size)

$I = \{1, \dots, n\}$ set of indexes of all available instances in memory

$\hat{y}$ estimated label

p_t probability of misclassifiying of error rate at time t

s_t standard deviation of error rate at time t

1 Orange and France Telecom Group

France Telecom-Orange is one of the world's leading telecommunications operators with 171,000 employees worldwide, including 105,000 employees in France, and sales of 10 billion Euros in the first three months of 2012. Present in 33 countries, the group had a customer base of 225 million customers at 31 March 2012, including 181 million customers under the Orange brand, the Group's single brand for internet, television and mobile services in the majority of countries where the company operates. Orange is one of the main European operators for mobile and broadband internet services and, under the brand Orange Business Services, is one of the world leaders in providing telecommunication services to multinational companies. (http://www.orange.com/en/group)

2 Introduction of our problem

2.1 Concept drift

Concept drift refers to a non stationary machine learning problem over time. A sequence of instances is observed, one instance at a time, not necessarily in equal spaced-time intervals. $X_t \in R^p$ is a vector in p-dimensional feature space observed at time t and $y_t \in \{c_1 \dots c_k\}$ its corresponding label. The concept or data source is defined as a set of prior probabilities of classes and class-conditional probability density functions.

$$S = \{(P(c_i), p(X|c_i)), i = 1 \dots k\}$$

There is a concept drift if for any two time points i and j, $S_i \neq S_j$. In another way, a concept drift indicates changes in the posterior distributions of data $p(y|X)$.

Assuming that the concept of a sequence of instance changes from S_i to S_j, we call a drift a **sudden drift**, if at time t, S_i is suddenly replaced by S_j.

Instead of a sudden change which means that the source is switched during a very short period of time after the change point t, gradual drift refers to a change with a period both sources S_i and S_j are active. As time passes, the probability of sampling from source S_i decreases, probability of sampling from source S_j increases until S_i is completely replaced by S_j. There is another type of **gradual drift** called incremental or stepwise drift. With more than two sources, the difference between the sources is very small, thus the drift is noticed only when looking at a longer time period. **Reoccurring** context means that previously active concept reappears after some time. It is not certainly periodic. So there is not any predictive information on when the source will reappear. In Figure 1, we can see how these changes happen in the mean of a dataset.

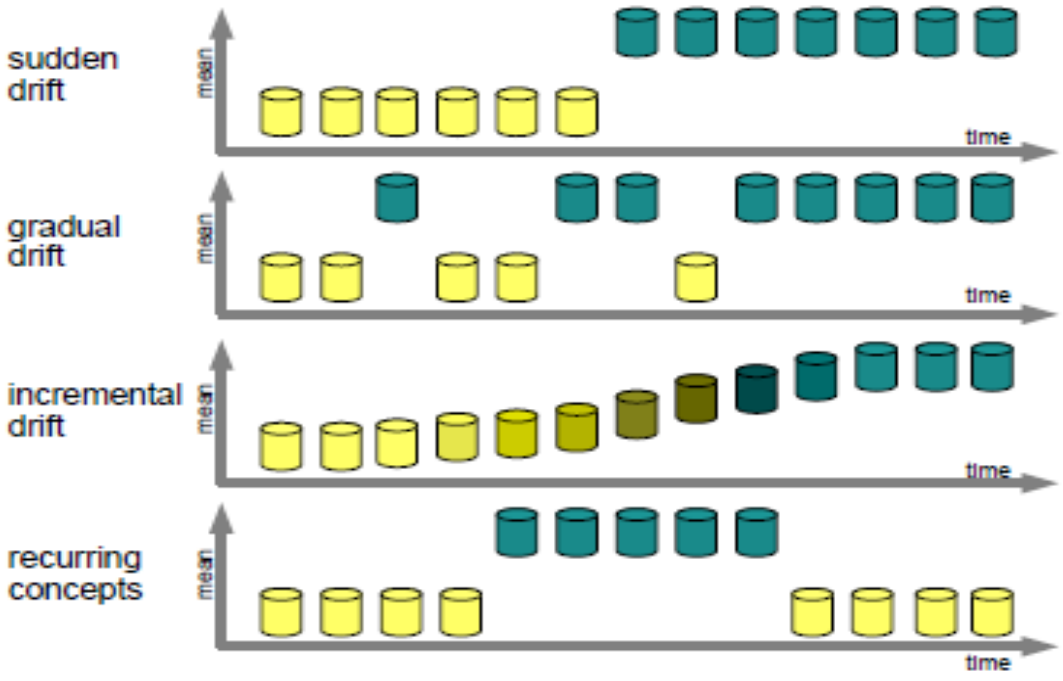


Figure 1 Concept Changes in the mean of a sequential data (Žliobaite, 2010)

2.2 Strategies / categories

Strategies for handling concept drift can be presented as table below in two dimensions.

	Trigger-based	Evolving
Single classifier	Detectors(variable time windows)	Forgetting(fixed time windows, instance weighting)
Ensemble	Contextual (dynamic integration, meta learning)	Dynamic ensemble(adaptive combination rules)

Trigger-based means that there is a signal which indicates a need for model change. The trigger directly influences how the new model should be constructed. Most often change detectors are used as triggers.

Evolving methods, on the contrary, do not maintain an explicit link between data progress and model construction, and usually do not detect changes. Evolving learners adapt at every step and it often corresponds to a forgetting mechanism. There is not explicit change detection. They provide neither indication about change points nor dynamics of the process generation data.

Ensemble algorithm consists of a set of different single classifiers whose decisions are aggregated by a voting rule or given by a properly chosen classifier according to the observed incoming data (contextual). The decision produced by the former strategy is usually more accurate than the decision given by a single classifier. It is necessary to diversify ensemble members from each other.

2.3 Incremental learning

2.3.1 Adaptive methods

The adaption model is characterized by changing the decision model to adapt to the most recent examples. We can overview this part in two dimensions: “when” and “how” the adaption is applied.

“Blind methods” or evolving methods reconfigure the learner at regular intervals without considering whether changes have really occurred. Another type of adaptive models is “informed method” usually used on trigger-based classifiers. They only change the decision model after a change is detected.

The adaptivity can be achieved by adjusting training set with forgetting mechanism which punishes irrelevant data. Window technique with fixed or variable size is widely used. With change detector, the window is cut at the change point and the data before the change alarm are deleted. “Instance weighting” also represents the forgetting mechanism by giving each stored instance a time related similarity measure(weight) which signifies its importance on decision making procedure. Another group of methods use instance selection. The incoming testing instances (unlabeled) are inspected. Based on the similarity between the testing instance and profiled partitions of the training set, one is chosen and used for building the classifier for prediction. After updating training set to concept drift, it is necessary to retrain the learner. For some learners such as Naïve Bayes and SVM, reconstruction of the whole decision model is required. On the contrary, for granular decision models like decision rules and decision trees, we just need to rebuild part of the decision model or adjust decision model’s parameters by resuming recent data.

2.3.2 Ensemble update strategies

For classifier ensembles, adaptive method is more flexible, because it is able to adjust fusion rules (e.g. base learner weights), update training data, delete or add ensemble members, and modify ensemble structure(decision trees).

2.3.3 Base online learners

2.3.3.1 VFDT, Hoeffding Tree

Hoeffding Tree (Pedro Domingos, 2000) is an incremental decision tree algorithm using Hoeffding bound as split decision function, which only a small subset of incoming instances passing through is needed to find the best split.

2.3.3.2 Naive Bayes

Naïve Bayes classifier (Langley, 1992) is a conditional probabilistic model classifier based on Bayes’ theorem and assuming that variables are independent.

According to Bayes’ theorem, the conditional probability for an instance to be classified into class Y given its variable values becomes

$$p(Y|X_1, X_2, \dots, X_n) = \frac{p(Y)p(X_1, X_2, \dots, X_n|Y)}{p(X_1, X_2, \dots, X_n)}$$

Supposing that all these variables are independent, the decision/prediction of the classifier $\hat{y}$ can be written as

$$\hat{y} = \text{argmax}_y P(Y = y) \prod_{i=1}^{n} p(X_i = x_i | Y = y)$$

2.3.3.3 Perceptron

Perceptron (Rosenblatt, 1957) is a linear binary classifier which can be regarded as a single layer artificial neural network, whose decision model is :

$$f(x) = \begin{cases} 1 \; if \; wx + b = 0 \\ 0 \quad otherwise \end{cases}$$

Where x is the input variable, *w* is a vector of real-valued weight, *wx* is the dot product, and b is the "bias".

2.3.3.4 SGD (Stochastic Gradient Descent)

Stochastic Gradient Descent (SGD) (Bottou, 2004) classifier uses linear model whose parameters are estimated by gradient descent optimization method for minimizing its object function, usually least squares or least mean squares.

3 Related Work

3.1 Concept Drift Detection

Change detectors monitor real-time measures and send alarms as soon as drifts are detected. A good test reduces detection delay (i.e., time needed to detect the change) and minimizes both false positive and negative detection numbers. Multiple or hierarchical detectors are sometimes applied to deal with different types of concept drifts and reduce false positive number. Measures monitored are usually classifier's performance indicators or data properties in order to indicate the change point in time.

3.1.1 Classifier's performance measures

When there is a change in examples class-distribution, classifier's actual model does not correspond to the actual distribution any more. Meanwhile, its error-rate increases. The basic idea of change detection in predictive learning is monitoring its prediction quality.

FLORA family algorithms (Kubat., 1996) monitor accuracy and model size. It includes a window adjustment heuristic for a rule-based classifier. Flora3 deals with recurring concepts. (Klinkenberg, 1998) proposes to monitor performance indicators (accuracy, recall and precision over time), and compare them to a confidence interval of standard sample errors for a moving average value using the last M batches for each particular indicator.

Statistical Test of Equal Proportions **(STEPD**) (Kyosuke Nishida, 2007) is based on the assumption that classifier's accuracy on recent W examples will be equal to the overall accuracy from the beginning of the learning if the target concept is stationary; a significant decrease of recent accuracy suggests that the concept is changing. It calculates a measure *T* which can be regarded equivalent to chi-square test with Yates' continuity correction and compares it to the percentile of the standard normal distribution to obtain the observed significance level, *P*. It stores examples in short term memory while $P < \alpha_w$; rebuilds the classifier from the stored examples and resets all variables if $P < \alpha_d$. Stored examples are removed if $P \geq \alpha_w$.

3.1.2 Model properties

Model evaluation can be monitored to detect concept drift using indicators such as model complexity and memory size. For decision trees, it could be tree size, attribute-class statistics (Bifet, 2009), or misclassification number for a node (Gama J. R., 2003).

3.1.3 Data properties

Another assumption is that after a concept drift occurs, current incoming data distribution differs from the one before. So we just need to monitor the data distribution variance using statistical methods. For numeric sequences, we can apply a sequential analysis. **CUSUM test** (Page, 1954) detects significant increase or decrease in successive observation of a random variable. **GMA** (geometric moving average test) (Roberts., 2000) is an alternative of CUSUM test. **SPRT** (Sequential probability ration test) (Wald, 1947) calculates the cumulative sum of the log-likelihood ratio as new data arrive with specific sequential hypothesis. For example, we have two hypotheses $\mathrm{H}_0, \mathrm{H}_1$ represented by their probability distribution functions f_0, f_1. As the *i*th instance x_i arrives, we calculate the cumulative sum of the log-likelihood ratio.

$$\Lambda_i = \Lambda_{i-1} + log\frac{f_1(x_i)}{f_0(x_i)}$$

Then compare the sum with two thresholds *a, b* decided beforehand.

$$a < \Lambda_i < b : continue\ monitoring$$
$$\Lambda_i \geq b : accept\ H_1$$
$$\Lambda_i \leq a : accept\ H_0$$

The changes are detected by the passage from one hypothesis to the other.

Another option is to monitor distributions on two or multiple different time-windows.

Mann-Kendall test evaluates the signs of observed values differences.

Density estimation on data analysis (Alexis Bondu B. G.-L., 2010) summarizes the input data stream by micro-clustering algorithm "Denstream", and estimates the underlying distribution of incoming data stream periodically using a variant of Parzen window. It sends an alarm if the distance between the current estimated distribution and a reference distribution evaluated by the Kullback-Leibler divergence is significant. Thus there is a diagnostic evaluating the cause of concept drift. **ACED** (Nishida, 2005) calculates q_t, the predictive accuracy of the online classifier for recent *W* examples, and the $1 - \alpha_d$ confidence interval for q_t whose lower endpoint is $q^l{}_t$. . When $q^l{}_{t-W} > q_t$, it declares a concept drift and initializes the classifier after *2W* examples arrived. It is able to detect concept drift quickly when *W* is small; however, such small window size often causes misdetection.

ADWIN change detector keeps a sliding time-window with the most recent examples divided into 2 sub-windows. When these "large enough" sub windows exhibit "distinct enough" averages, we can conclude that there is a concept change and the older portion of the window is dropped. Statistical test on different distributions is to check whether the observed average in both sub windows is more than the threshold.

Test on p-value (Anton Dries, 2009) assumes that a small p-value suggests that a concept drift is likely. The system signals the discovery of a new concept if a peak in the sequence of p-values is detected.

4 Testing/validation methodology

4.1 Test Protocol

For offline classifiers:

Holdout evaluation: Holdout an independent test set. Apply the current decision model to the test set at regular time intervals or set of examples. The loss estimated in the holdout is an unbiased estimator.

For online classifiers:

Interleave Test-Then-Train: Each individual example is used to test the model before it is used for training, and from this the accuracy is incrementally updated. Its advantage is that the model is always being tested on unseen examples and no holdout set is needed so that available data are fully used. It also produces smooth plot accuracy over time.

4.2 Indicators

A good learner handling concept drift should have high accuracy, low computational cost (memory space and operational time) and fast reliable change detection.

Change detection performance could be evaluated on:

- **False positives (FP)** counts number of detections that a test sends in the sequence when there is not. The ideal drift detection algorithm should not be too much influenced by noise.
- **False negatives (FN)** counts the times that a test does not detect a change when there is one.
- **Recognition delay (RD)** measures delay time for change detection.
- **Computational time (CT)** provides execution time needed to perform the test detector on a reference platform.
- **Memory occupation** is the size of memory occupied for storing examples and model parameters over time.
- **Ram-Hour** (Albert Bifet G. H., 2010) counts cost per hour and memory used at the same time. Every GB of RAM deployed for 1 hour equals to 1 RAM-Hour. It is a measure on resource used by classifiers during learning procedure. It fits practically pricing strategies for main cloud computing service providers, such as Amazon EC2 and Microsoft Azure.

4.3 Dataset

4.3.1 Concept drift generators

A general way to generate concept drifts is to mix data with different characteristics into one stream and modify its composition to create various types of concept drifts.

In order to test concept drift detectors, we can use a single labeled dataset. We regroup all examples by label. Each new dataset contains all the attributes except the class label column. They are assumed to be generated by different concept. We usually use two or three classes having the most members. It's easy for pre-treatment but not possible to evaluate classifiers' performance. When several

datasets are available, we concatenate the lists of attributes from all of them, and the number of classes is set to the maximum number of classes to fit all dataset.

To create sudden drift, we just need to use one dataset before the breakpoint and switch to another after it. Gradual drifts are more complicated. (Bifet, 2009) suggests to join two datasets using the sigmoid function by modifying their proportion on the new data stream gradually. The function can be parameterized by the location of change point and the length of change interval. Reoccurring drifts are achieved by repeating this procedure to replace the current dataset by the "previous" one. It's also possible to introduce multiple concepts sequentially into our test data stream. This method builds a labeled data stream which is available for evaluating classifiers and can be controlled so that it's possible to create divers situations. However, joining datasets from sources with little nature alike may lead to insensible change detectors, as it's not the common case in reality.

4.3.2 Synthetic data generators

MOA (Massive Online Analysis) contains a serial of artificial stream generators as SEA concepts, STAGGER, rotating hyperplane, random tree, and random radius based functions. (Albert Bifet G. H., 2011)

- **SEA Concept generator** (Kim, 2001) contains abrupt drift and generates points with 3 attributes between 0 and 10 where only the first 2 attributes are relevant. These points are divided into 4 blocks with different concepts by giving each block a threshold value which is the upper bound to the sum of the first two attributes.
- **STAGGER** (Granger, 1986) is a collection of elements, where each individual element is a Boolean function of attribute-valued pairs represented by a disjunct of conjuncts.
- **Rotating hyperplane** (G. Hulten, 2001) uses a hyperplane in *d*-dimension as the one in SVM and determines the sign of labels. It's useful for simulating time-changing concepts, because we can change the orientation and position of the hyperplane in a smooth manner by changing the relative size of weights.
- **Random RBF** (Bifet, 2009) creates numerical dataset whose classes are represented in hyperphere of examples with random centers. Population of each class depends on its weight. The probability that a point is distributed in a position is calculated by a radical basis function on its distance from its centers.
- **LED** (Leo Breiman, 1984) (LED Display Domain Data Set) is composed of 7 Boolean attributes which are predicted LED displays (the light is on or not) and 10 concepts. Each attribute value has the 10% of its value inverted. It has an optimal Bayes classification rate of 74%.
- **Waveform** (Waveform Database Generator (Version 2) DataSet) (Leo Breiman, 1984) produces 40 attributes including noise, and the latter 19 attributes are all noise attributes with mean 0 and variance 1. It differentiates between 3 different classes of waves, each of which is generated from a combination of two or three base waves.
- **Function generator** (Rakesh Agrawal, 1993) produces a stream containing 9 attributes, 6 numeric and 3 categorical, describing hypothetical loan applications. The classes (whether the loan should be approved) are presented in a Boolean label defined by 10 functions.

4.3.3 Real world data

In many articles, the UCI Machine Learning Repository is cited as real world data source. Here we introduce several most used datasets on concept drift handling.

- **Forest Cover type dataset** (Covertype) contains 581012 instances of 54 categorical and integer attributes describing wildness areas' condition collected from US Forest Service (USFS) Region 2 Resource Information System (RIS) and US Geological Survey (USGS) and USFS data. The class set consists of is 7 forest cover types.
- **Poker-Hand dataset** (Poker Hand)**:** Each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. There is one Class attribute that describes the "Poker Hand". The order of cards is important, which is why there are 480 possible Royal Flush hands as compared to 4.
- **Electricity dataset** (Harries, 1999)**:** This data was collected from the Australian New South Wales Electricity Market. It contains 45312 instances drawn from 7 May 1996 to 5 December 1998 with one instance for each half hour. The class label (DOWN or UP) identifies the change of the price related to a moving average of the last 24 hours. The attributes are time, electricity demands and scheduled power transfer between states, all of which are numeric values.

5 Inspiration

5.1 Drift Detection Method (DDM)

(Gama J. M., 2004) introduces Drift Detection Method (DDM) monitoring incremental global error rate of an online learner. The error rate at time t is p_t, probability of misclassifying;

$$p_t = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predicted\ instances}$$

$$p_{t+1} = \begin{cases} p_t + \frac{1 - p_t}{t} & if\ the\ prediction\ for\ the\ tth\ instances\ is\ incorrect \\ p_t + \frac{p_t}{t} & otherwise \end{cases}$$

And its standard deviation is calculated by

$$s_t = \sqrt{p_t(1 - p_t)/t}$$

It is assumed that p_t decreases as time advances if the target concept keeps stationary; otherwise, significant increase of p_t is observed.

The values of p_t and s_t are stored when $p_t + s_t$ reaches its minimum. If the performance indictor for each prediction z_t equals to 0, if the prediction is false; 1, if it's correct. Then after reaching the classifier's stable state on stationary data stream, z_t follows Bernoulli distribution with $p = p_{min}$. So the sum of z_t follows Gaussian distribution with *pmin* as mean and *smin* as standard deviation. This hypothesis can be available at least if there are a large number of examples (n>30 by default proposed by the authors). Two thresholds are fixed by estimating p-value. New examples are kept in short term memory while $p_t + s_t \geq p_{min} + 2s_{min}$. The concept drift is detected and we need to rebuild the classifier from the stored examples and reset all variables if $p_t + s_t \geq p_{min} + 3s_{min}$.

This method performs well for detecting sudden changes. But it has difficulties detecting gradual changes. The hypothesis is not available because theoretically z_t is not identically independently distributed, because for incremental learners keeps learning even if incoming instances are iid, we can't say its performance is not temporal /dynamic.

DDM Algorithm

Parameters: MinNbForDetection=30 (default value)
Initialize :p=1, n=0, s =1, pmin=inf, smin=inf
while (an instance (X_t, y_t) comes in){
n=n+1
Learner predicts on X_t
if (the prediction is false) {
p=p+(1-p)/n
}
else {
p=p-p/n
}
s=$\sqrt{p(1-p)/n}$
if (p+s<= pmin+ smin){ pmin=p smin=s}
if (n>MinNbForDetection){
ddm=$(p_t + s_t - p_{min})/s_{min}$
If(ddm≥3){
Drift detected
}
Else if(ddm≥2) {
Warning sent
}
}
If neither drift nor warning : stationary state
}

After observing figures of Annex 3, we notice that DDM does not work well with learners like Hoeffding Tree, which intern modification while learning is not gradual, and perceptron, which only keeps short term memory.

5.2 Early drift detection method

EDDM (early drift detection method) (Baena-Garcia, 2006) monitors the time interval (distance) between two occurrences of classification errors. The assumption is that any significant decrease in the distance suggests changes on concept. It calculates the average distance between two errors p'_t and its standard s'_t ; keeps in memory values of p'_t and s'_t when $p'_t + 2s'_t$ reaches its maximum; stores examples in short-term memory while $\frac{p_t+2s_t}{p_{min}+2s_{min}} < \alpha$; rebuilds the classifier from the stored examples and resets all variables if $\frac{p_t+2s_t}{p_{min}+2s_{min}} < \beta$. The detection starts after 30 errors have occurred. The method performs well for gradual changes; but it is not good at detection drift in noisy examples.

EDDM Algorithm

Parameters: MinNbForDetection=30 (default value), α, β
Initialize :p=1, n=0, s =1, pmin=inf, smin=inf
while (an instance (X_t, y_t) comes in){
n=n+1
Learner predicts on X_t

if (the prediction is false) {
p=p+(1-p)/n
}
else {
p=p-p/n
}
s=$\sqrt{p(1-p)/n}$

if (p+s<= pmin+ smin){ pmin=p smin=s}

if (n>MinNbForDetection){
eddm=$\frac{p_t+2s_t}{p_{min}+2s_{min}}$
If($eddm < \beta$)) Drift detected
Else if($eddm < \alpha$) Warning sent
}
If neither drift nor warning : Stationary state
}

5.3 Alexis Bondu's Supervised approach for change detection proposition

5.3.1 MODL : a supervised discretization approach and a supervised grouping approach

In this section we discuss the MODL method for discretization on continuous attributes and grouping on categorical variables.

The MODL method is based on a Bayesian approach of the discretization problem which means that the optimized method is obtained by maximizing the probability P(Model|Data) of the model given data. According to Bayesian rule and considering that the probability P(Data) is constant while the model varies, we just need to maximize P(Model)P(Data|Model)

or minimize C(Model)=-[log(P(Model))+log(P(Data|Model)] .

The MODL model is defined by

- I: the number of intervals
- $\{N_i\}_{1\leq i\leq I}$: the number of examples located in the interval i
- $\{N_{ij}\}_{1\leq i\leq I, 1\leq j\leq J}$: the number of examples labeled by the class j located in the interval i

Then the problem is transferred into a model selection problem. Then C(Model) becomes

$$\mathrm{C}(model) = \log \mathrm{N} + \log \binom{\mathrm{N}+\mathrm{I}-1}{\mathrm{I}-1} + \sum_{\mathrm{i}=1} \log \binom{N_i + J - 1}{\mathrm{J}-1} + \sum_{\mathrm{i}=1} \log \frac{N_i!}{N_{i1}!\, N_{i2}! \ldots N_{iJ}!}$$

where

$$-\log\big(P(model)\big) = \log \mathrm{N} + \log \binom{\mathrm{N}+\mathrm{I}-1}{\mathrm{I}-1} + \sum_{\mathrm{i}=1} \log \binom{N_i + J - 1}{\mathrm{J}-1}$$

$$-\log\left(\mathrm{P}(\mathrm{Data}|\mathrm{Model})\right) = \sum_{\mathrm{i}=1} \log \frac{N_i!}{N_{i1}! N_{i2}! \ldots N_{iJ}!}$$

$$\mathrm{N} = \sum N_i = \sum N_{ij}$$

The quality of the optimal MODL model, called Map (Maximum A Posteriori), is evaluated by

$$\mathrm{Gain} = 1 - \frac{\mathrm{C}(\mathrm{Map})}{\mathrm{C}(\mathrm{M}_\emptyset)}$$

where $\mathrm{M}_\emptyset$ is the null model which discretizes the variable into only one interval.

Gain (compression rate) is equal to 0 if the variable can not be discriminated, indicating that the distributions of variables with different label is similar or the same. Otherwise, the Gain value is strictly positive and increases as the difference in distribution becomes more and more significant. Finally, the Gain can reach to 1, when the variable can be perfectly discriminated by the model and indicates there's no cross-over between the labels' distribution.

In this way, the MODL discretization method becomes a univariate hypothesis test for similarity like Chi2 test on distribution.

MODL is also used for grouping categorical variables, following a simple idea, which if the final discretization "interval" number is smaller than variable's possible value number, we can group the

categories in the same “interval” into one. Finally, MODL is capable to calculate both continue and categorical variables similarity.

5.3.2 Change detection proposition

Our proposition is inspired by Alexis Bondu’s article “A Supervised Approach for Change Detection in Data Streams” (Alexis Bondu M. B., 2011). His proposition uses two time windows containing historical examples labeled “Class 1” and the most recent examples labeled “Class 2”. The detector discriminates on each attribute with MODL method. MODL gives a positive value when there is a significant difference in distribution from the two classes. And then it uses the Gain(model) as a criterion for detection reliability.

His proposition fits our aim partially by introducing MODL discretization method to describe concept drifts automatically on instance attributes’ distribution from two time windows.

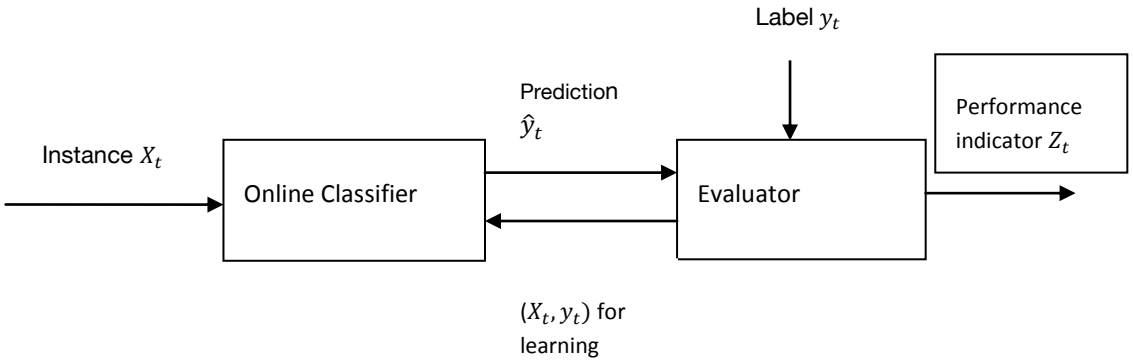
The MODL drift detection based on incoming instance variable distribution change is proved useful. However, it becomes more and more complicated. The number of instance attributes increases as discretization on each dimension is required. In order to reduce detection complexity bought by MODL, the usage of performance indicators should be considered.

6 Our Proposition & Preliminary experiences

6.1 Objective

Our initial objective is automation for drift detection without any preliminary knowledge on performance indicators' distribution for online learners. Absence of parameterization is a critical element for drift detection automation, because the phase of parameters adjustment is very complicated and expensive, and products uncertainty. We also hoped that it can be used with a numerous types of classifiers and be as much as possible independent from data stream properties.

Our proposition is to use a performance indicator Z_t as random variable for MODL method and detect concept drifts.



6.2 Our hypothesis

First of all we need to introduce several hypotheses for our method.

The first hypothesis is that the incoming instances variables are iid when the source is stationary which is proved by the definition of "concept".

According to the PAC modeling, the classifier's performance keeps getting better while more instances come from stationary source, and it obtains more information about the targeted data stream. When there is a change in the class-distribution of the examples, the actual model does not correspond any more to the actual distribution. There will be a difference in an incremental classifier's prediction quality after a concept drift occurs, and would be reflected on indicators' value, for example, the error-rate increases. In other words, if there is no influence on prediction quality after a (slight) concept drift, we have no motivation to handle it. Our basic idea of change detection in predictive learning is to monitor the quality of the process. So an online classifier's performance evaluation indicator on supervised data stream can be considered as time series whose member Z_t depends on other instances arriving before t as the classifier keeps learning on incoming examples and shows dependency on them.

Our second hypothesis is that the prediction quality indicator follows a certain pattern. During a period when incoming data is stable, it can be reflected on distribution of value (See Annexes 1 and 2). There will be a change in its behavior pattern and value distribution after a concept drift. In this case, the MODL method can be introduced to test similarity on indicator's value distribution from two

time-windows, one representing an older state as reference, the other representing current classifier prediction quality. The advantage of MODL is that it does not need any information from the indicator value distribution. However, we need to separate the classifier's amelioration by incremental learning from changes caused by concept drift, which is impossible to get any knowledge only by observing simple MODL level value. For example, in case using error-rate, we consider there is no drift when error rate decreases over time.

6.3 Classifier adaptation

To valid our second hypothesis and figure out our method's classifier compatibility, we looked into classifiers' evolution on prediction quality when there is a concept drift. The classifiers tested are Naïve Bayes, Hoeffding Tree, SDG and Perceptron. The data stream is generated by SEA generator on MOA (Section 4.3.2). For the first three classifiers, there are 150000 instances and the generator changes its label decision function from "Function 1" to "Function 2" in the original article at 25,000 within a window of 1000 instances. As Perceptron gets stable slowly, the experimental settings are different: there are 200000 instances and the same change happens at 100000. Figure 2 to Figure 5 show accuracy on every 1000 instances over time for all of them.

Temporal Accuracy on every 1000 instances over time on SEA data stream with concept drift

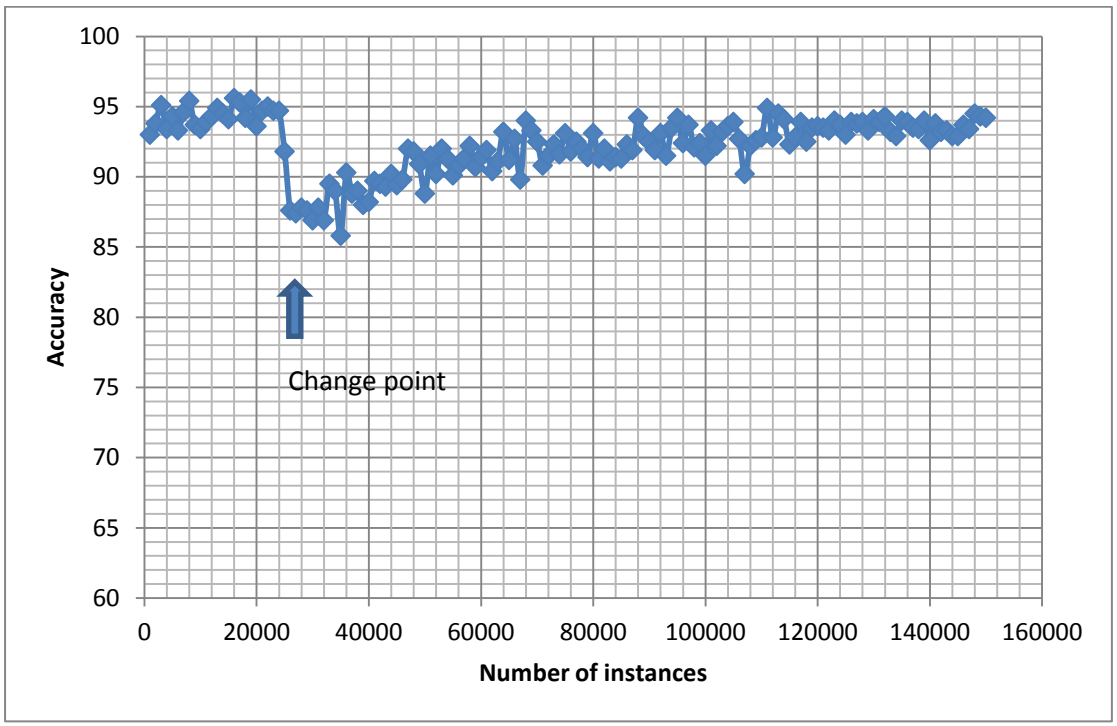


Figure 2 Naive Bayes

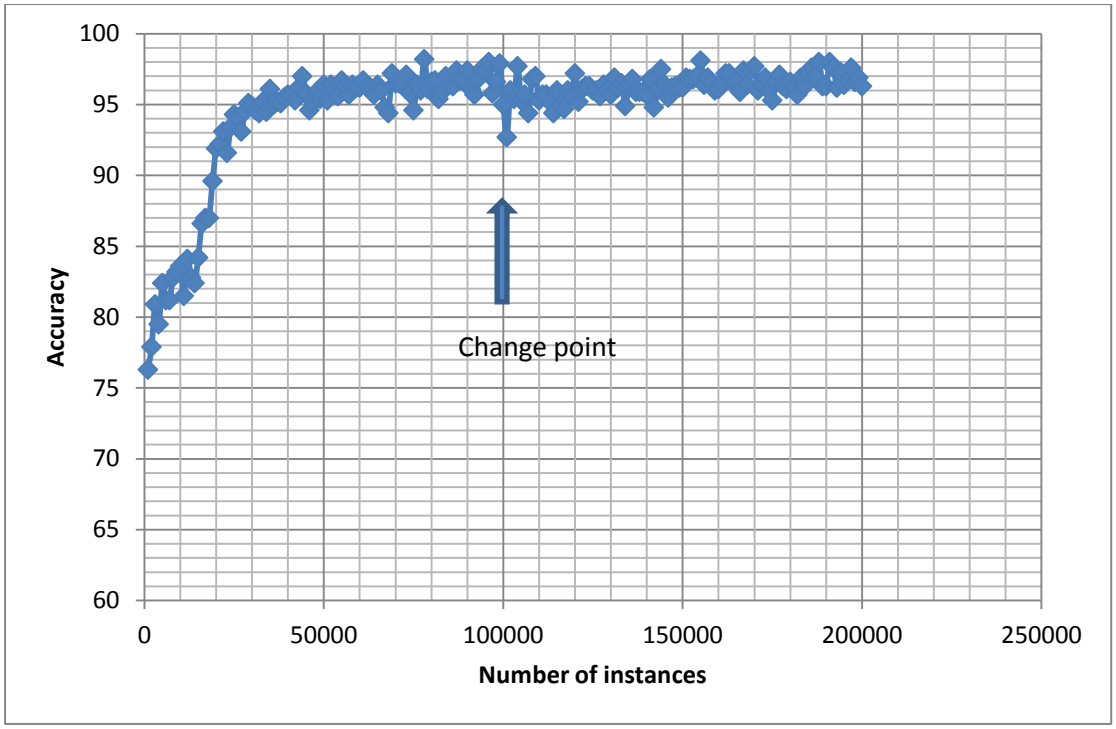


Figure 3 Perceptron

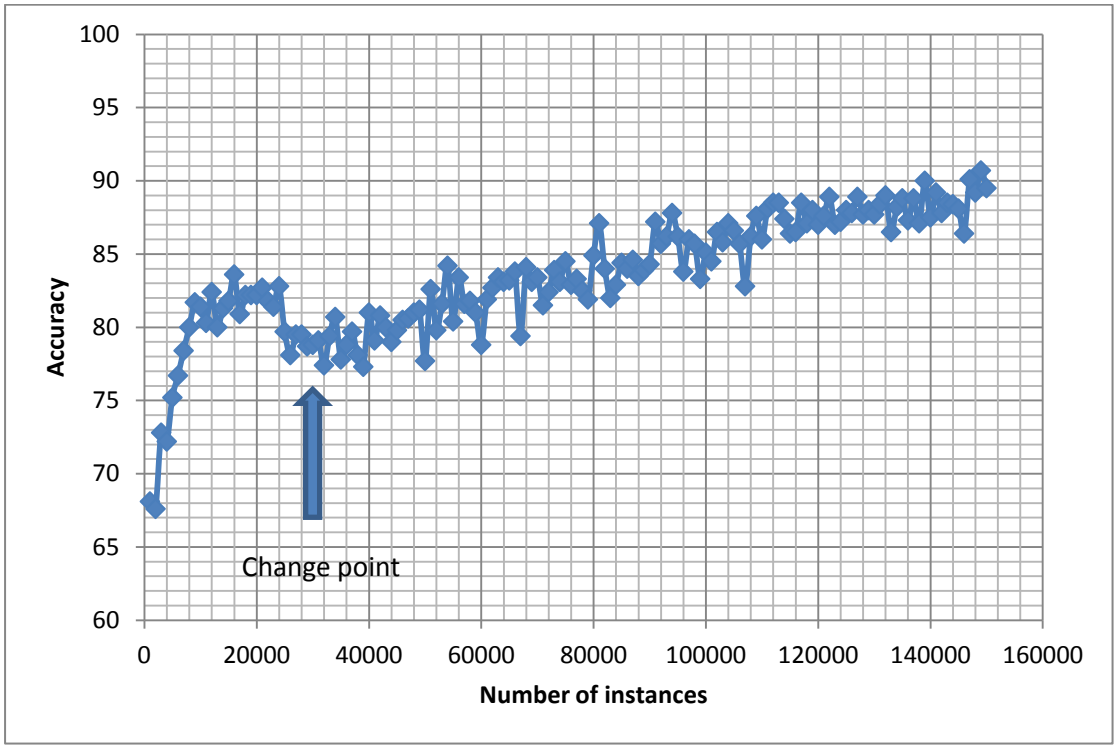


Figure 4 SGD

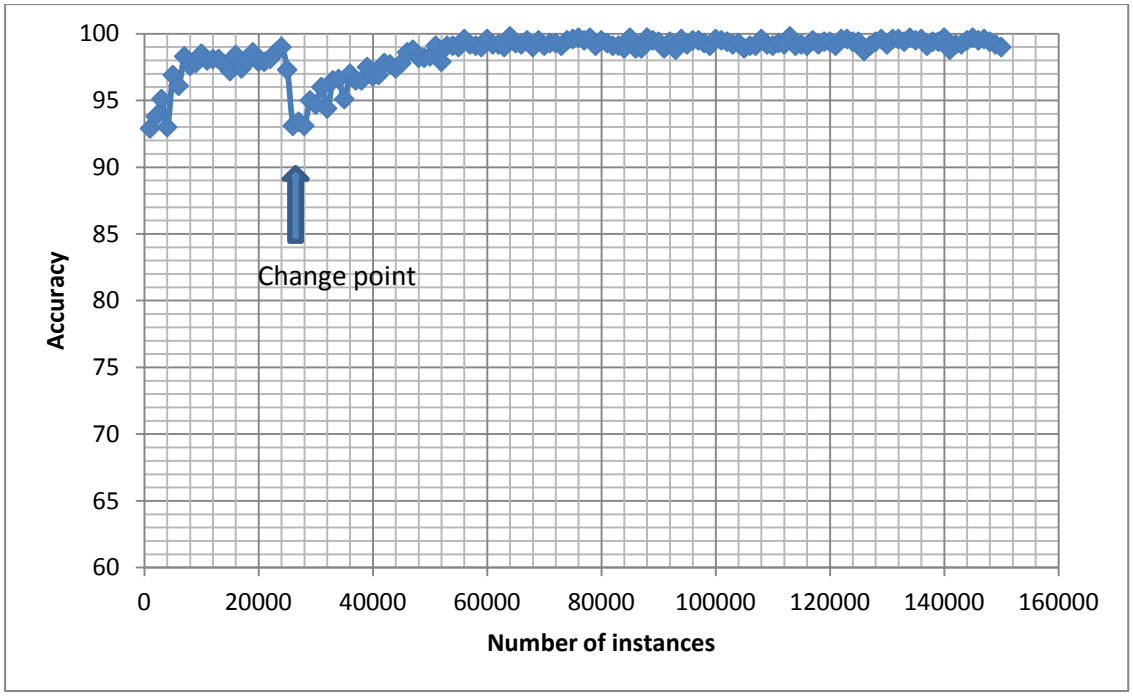


Figure 5 Hoeffding Tree

From these experiences, we concluded that the magnitude of change on prediction quality depends not only on the difference between two data stream sources but also on classifier's historical example forgetting mechanism. For classifiers which keep statistics on data for a long period of time such as Hoeffding Tree and Naïve Bayes classifier, the drift causes a significant decrease on prediction quality. The "recovery" takes more time than initial learning because it has to firstly efface the influence of the "old" source instances. For those with less memory on historical instances such as Perceptron and SGD, are less influenced by the drift and quickly adapt to the new source. Since our method is based on supervised prediction result/quality monitoring, we were more interested in the first cases where there is real need for detecting in time concept drifts and evident abnormal behaviors (quality declines) on reaching drift.

6.4 Time window strategy

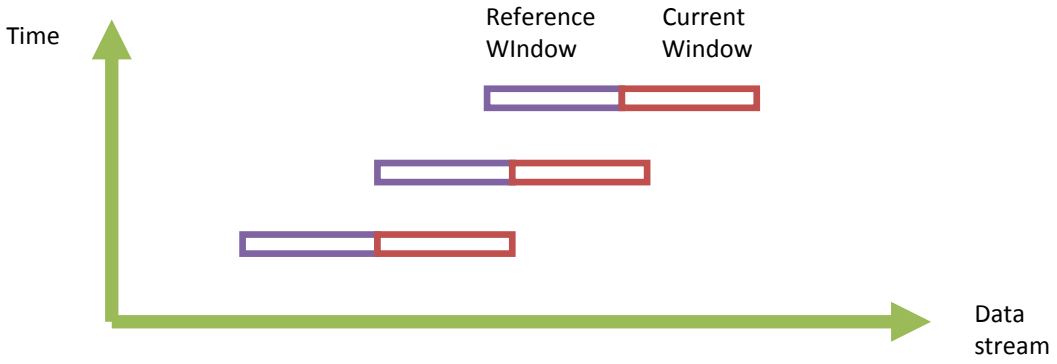
Time window technique takes arrival time as main measure for incoming examples. Because of concept change, the old observations become irrelevant to the current period and cause performance drop. The examples are deleted from the memory and totally forgotten according to some time criteria. Examples are stored in a FIFO data structure. At each step the learner induces a decision model using only the examples that are includes in the time window.

The key difficulty is how to select an appropriate window size. Small windows can assure a fast adaptability in phases with concept changes. In more stable phases they affect the learner performance. Large windows produce good and stable learning results in stable phases but they can not react quickly to concept changes.

Besides using fixed size window, many propose variable size window whose number of examples in the window is variable. They are often used in conjunction with a detection model. So the size of the window is decreased whenever the detection model signals drift and increasing otherwise.

We use time windows for storing performance indicator(s) instead of examples' attributes from the data stream. A two side-by-side equal-size sliding window strategy is employed whose size is Wc for

current window and Wr for reference window. As time passes, the instance prediction evaluation enters firstly the current window, after Wc instances, it moves into the reference window, and (Wr+Wc) after it is dumped by the detector and is no more in the memory. We chose this solution because for online learners, keeping the reference window fixed is not available. The performance is not stable (generally amelioration) and shows data dependency before reaching its error-rate's "lower bound" at the beginning of prediction procedure. The detector is easy to be disturbed by their instable behavior that is why we need to update the reference window. The other reason is that we believe the classifier can treat "slight/tiny" but not negligible changes on incoming data stream by learning incrementally on new incoming instance or/and its own forgetting mechanism. In this case, the reference window can not be left too far from the current observation window.



Another proposition once considered is to use the solution from Alexis Bondu's article, a reference window which is larger than the current window. The reference window is fixed and the current window is sliding over time. But there are two modifications: first, to cross classifier's learning period, initially let the reference window slide after the current window as the previous proposition; once the error rate stop increasing (the error rate in the current window equals to the one in the reference window), fix the reference window and the other one keeps sliding over time. However, this proposition is not capable to localize concept drift.

As in DDM we keep the global error p and its standard deviation s, and storing the minimal values. Our method starts detecting after (*p+s>pmin+2smin*) (See Annex 3) in order to separate prediction quality amelioration from declines probably caused by concept drift. Even if we used a part of the hypothesis of DDM, since the "bound" is rather loose, there should not be any problem for it to be the first filter for our proposition.

Proposed algorithm

Algorithm
Parameters: W
Initialize :p=1, n=0, s =1, pmin=inf, smin=inf
Wc={}, Wr={},LevelValue=0,OldLevelValue=0

while (an instance (X_t, y_t) comes in){
n=n+1
Learner predicts on X_t
Calculate z_t

if (the prediction is false) { p=p+(1-p)/n}
else {p=p-p/n}
s=$\sqrt{p(1-p)/n}$

if (p+s<= pmin+ smin){ pmin=p smin=s}

If(n<=W) Wr=Wr+$\{z_t\}$
Else if (n<=2W) Wc=Wc+$\{z_t\}$
Else
Wr=Wr+$\{z_{t-W-1}\}$-$\{z_{t-2W-1}\}$
Wc=Wc+$\{z_t\}$-$\{z_{t-W-1}\}$

If (p+s>pmin+2smin&& Wr and Wc are full){
Contingency Table=

value\class	*Wr*	*Wc*

LevelValue=Level.group(Contingency Table) or
LevelValue=Level.discretize(Contingency Table)

If(LevelValue> 0){
Drift detected
}
Else Suspect drift detected(Warning)
}
If no drift detected : Stationary state
}

6.5 Choice of performance indicator

All a proper performance indicator for our method has to be chosen and must satisfy conditions below:

- We can easily evaluate the learner's performance for the whole window time interval, so that it's possible to distingue changes caused by learning procedure and ones caused by concept drifts.
- Within small time window, we can get enough number of values for the MODL method.

6.5.1 "Right prediction"

Our first suggestion is inspired by DDM /EDDM method, which introduces a Boolean indicator: true if the prediction is correct; false otherwise. According to our hypothesis, it approaches to an iid random variable. We can easily get enough samples whose number is equal to the number of instances with a small window size. The learner's prediction improvement can be decided by comparing the error rate of each window.

6.5.2 Global error rate

Our second choice of indicator is global error rate (the p from DDM and EDDM). This indicator is very smooth (See Annexes 4 and 5) so that we do not need anymore the warning level and it is possible that a reference window larger than the current one would be more effective.

6.5.3 Score for a target class

Another indicator produced each time an instance arrives is the score/vote for each prediction class, which could be P(Y|X) for Naïve Bayes classifier, *wx+b* for Perceptron, etc. On MOA, these votes are provided by all classifiers. As our proposition monitors only one indicator, and the number of scores equals to the number of prediction classes for an instance, a target class must be chosen. The advantage of this indicator is that it reflects not only to instances' class distribution (higher the score is, more probably it is that the instance will be classified in the targeted class), but also the classifier's performance evolution (when the portion between classes is the same and there is a change in target class score).

7 Test

7.1 Test Protocol

Test protocols for learners handling concept drift have to look in two dimensions. First the classifier must have good performance in stationary phase which means little false positive drift alarm. Then its drift detection ability, indicated by number of false negative and detection delay, should be evaluated. We evaluated and discussed first the classifier's behavior and performance in the stationary case and then in the non stationary one. Moreover, we introduced a global evaluation by observing classifier's overall performance on dataset with concept drifts.

Data source

All datasets introduced here contain 100,000 instances. The stationary datasets used were generated by SEA generator's Function 1 and LED generator with 10% of noise.

The datasets with sudden drift were generated by STAGGER and SEA from MOA with a single drift at t=50000 whose width equals to 1000. As the drift change position and the change rate are known, drift detection is correct if it is sent in the drift/change window, the interval (drift position –drift window size, drift position + drift window size). The criterion to observe the precision of our detector is the rate of correct drift detections which equals to number of correct detection divided by total detection number.

Correct Detection Rate= detection number over change window/ total detection number

We calculated the rate but not only right or false detection number because a good change detector need to balance the sensibility of changes and the robustness to noises and classifier's instability. We supposed that performance of our algorithm would get better if the reference window size increases. We also compared the final accuracy to observe contribution of drift detection.

Dataset used with gradual drift was generated by MOA's rotating hyperplane generator, as it is difficult to identify a drift position; we only looked at the total accuracy. Higher the accuracy gets more the classification benefits from drift detection.

Classifiers

The classifiers used in our experiments are Naïve Bayes and Perceptron supposed to fit our drift detection strategy for handling concept drifts.

Drift detection handling

If a warning is sent, the classifier starts to store instances. If the drift detection is confirmed, the classifier and the detector are initialized and the classifier starts learning first of all from these instances. If detection is directly launched, the classifier and the detector are initialized and the classifier starts learning from new incoming instances.

Parameters estimation

Parameters (reference and current window sizes) used are shown in table below.

Reference window size (Wr)	Current window size (Wc)
50	50
100	50
450	50
100	100
200	100

Alternative methods

The alternative method is DDM, which is based on classifier's performance monitoring.

7.2 “Right prediction”

Without drift

Classifier		Naïve Bayes						Perceptron					
		LED			SEA			LED			SEA		
Wr	Wc	Number of false detections	Accuracy	Number of suspect drifts	Number of false detections	Accuracy	Number of suspect drifts	Number of false detections	Accuracy	Number of suspect drifts	Number of false detections	Accuracy	Number of suspect drifts
50	50	0	0.73949	183	0	0.88139	16	0	0.72876	0	0	0.82322	215
100	50	0	0.73949	183	0	0.88139	16	0	0.72877	0	0	0.8161	30
450	50	0	0.73949	183	0	0.88139	16	0	0.72873	0	0	0.82308	102
100	100	0	0.73949	183	0	0.88139	16	0	0.72873	0	0	0.82308	102
200	100	0	0.73949	183	0	0.88139	16	0	0.72873	0	0	0.818	35
DDM		0			0			0			0		

Sudden drift

Classifier		Naive Bayes						Perceptron					
Detector		Correct detection number/ number of detections		Accuracy		Number of suspect drifts		Correct detection number/ number of detections		Accuracy		Number of suspect drifts	
Wr	Wc	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER
50	50	0/2	0/1	0.94376	0.77973	0	46182	0/0	0/11	0.91952	0.8495	16	0
100	50	0/3	0/1	0.94772	0.9105	0	17197	0/0	0/1	0.92417	0.85503	87	12211
450	50	0/1	0/0	0.91645	0.7599	49706	51169	0/0	0/0	0.92576	0.85481	38	50907
100	100	0/1	0/1	0.92717	0.91035	0	17223	0/0	0/3	0.92703	0.85472	73	0
200	100	0/0	0/1	0.91629	0.91046	49592	17201	0/0	0/2	0.92748	0.85458	44	0
DDM		0/2	6/6	0.94743	0.99699			0/0	4/4	0.92379	0.8717		

Gradual drift

Classifier		Naive Bayes			Perceptron		
		Accuracy	Number of detections	Number of suspect drifts	Accuracy	Number of detections	Number of suspect drifts
Wr	Wc	Rotating hyperplane			Rotating hyperplane		
50	50	0.84235	4	0	0.89731	5	0
100	50	0.83411	5	0	0.89992	8	0
450	50	0.86167	5	0	0.9048	2	0
100	100	0.84731	2	0	0.90555	4	0
200	100	0.82785	3	17272	0.90396	3	0
DDM		0.90177	34		0.90501	7	

Although there was not detection on stationary datasets, it did not produce an acceptable correct detection rate on sudden drift datasets. When the window sizes are rather small (50/50, 100/50), there were detections as soon as the DDM criterion passes its threshold, meaning that our method is too sensitive to noise. On the contrary, when windows are large, there was few detection and we lost a lot on accuracy.

To figure out the problem, we plotted the difference in error rate between the two windows mentioned in our proposed algorithm applied on SEA sudden drift dataset, as MODL level value, when both windows sizes are fixed, is related to error rate change between them. We noticed that there was a very evident drop around the change point in accuracy for every 1000 examples. However, it showed only a slight increase in average of error rate between two windows in our algorithm during the drift interval. It was hard to figure out the change point among the other peaks. In order to leave out noise and rapid changes, a smoothing method would be useful, but would require supplemental parameters.

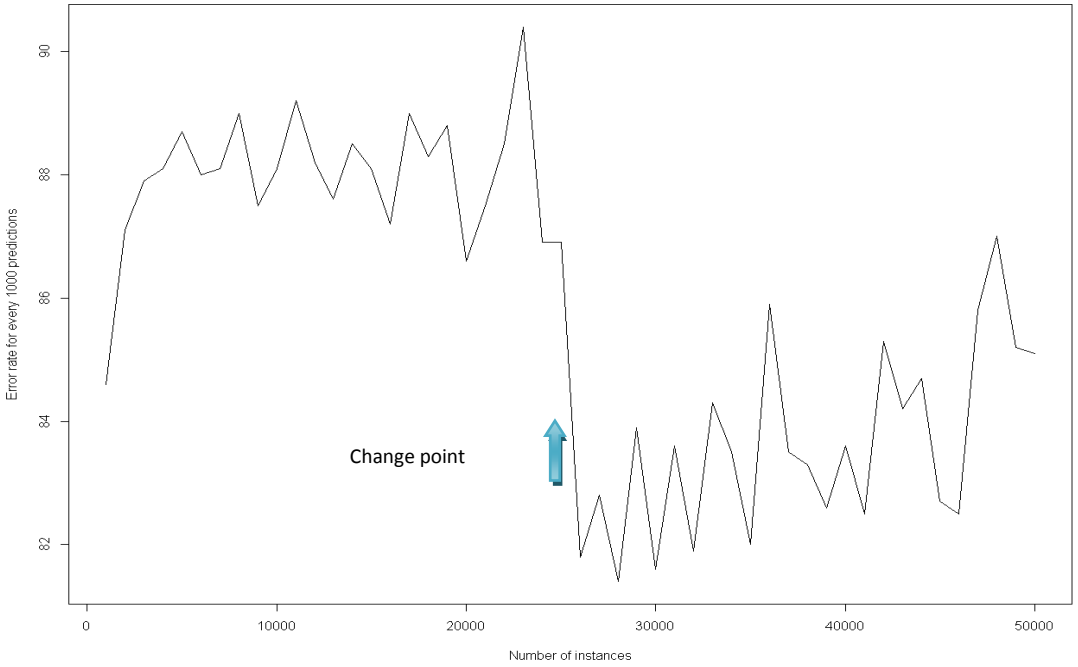


Figure 6 Accuracy (on percentage) on SEA sudden drift dataset with Naïve Bayes classifier

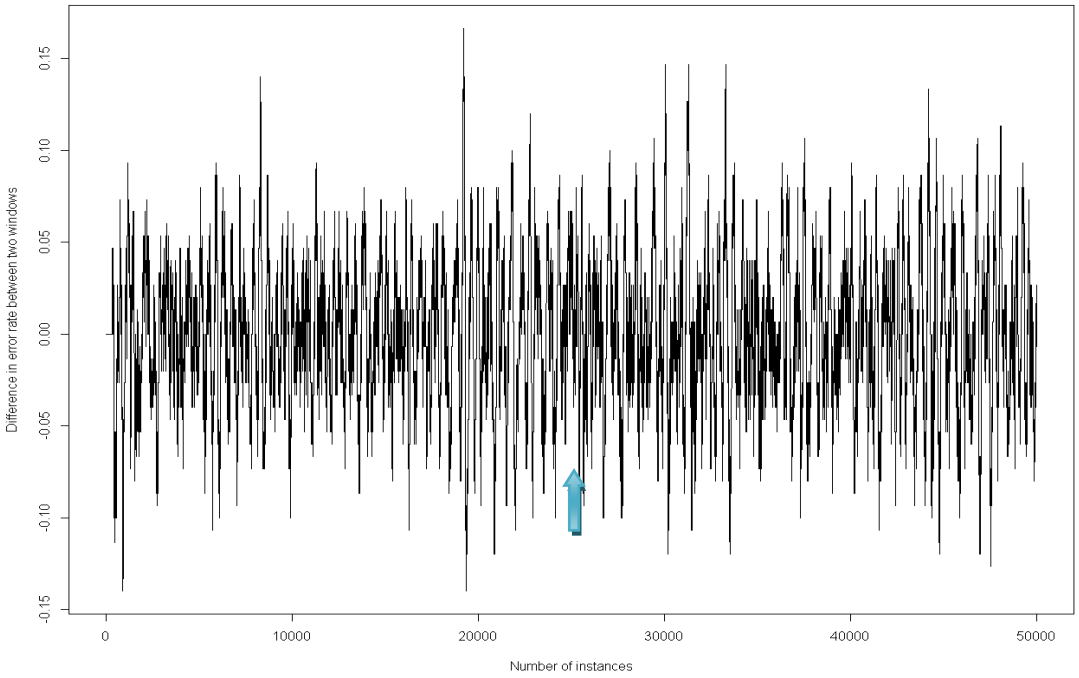


Figure 7 Difference in error rate between two windows on SEA sudden drift dataset with Naïve Bayes classifier

7.3 Global error rate

Without drift

Classifier		Naïve Bayes		Perceptron	
Detector		Number of false detections		Number of false detections	
Wr	Wc	LED	SEA	LED	SEA
50	50	1	0	0	0
100	50	1	0	0	2
450	50	1	0	0	0
100	100	1	0	0	10
200	100	1	0	0	0
DDM		0	0	0	0

Sudden drift

Classifier		Naive Bayes				Perceptron			
Detector		Correct detection number/Total detection number		Accuracy		Correct detection number/Total detection number		Accuracy	
Wr	Wc	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER
50	50	1/4	2/4	0.94729	0.99662	0/0	7/7	0.92845	0.87133
100	50	1/4	2/4	0.94729	0.99662	0/5	1/6	0.92416	0.87143
450	50	1/4	2/3	0.94729	0.9969	0/0	4/10	0.92397	0.87089
100	100	1/4	2/4	0.94729	0.99662	0/4	4/4	0.92675	0.87151
200	100	1/4	2/4	0.94729	0.99658	0/0	4/4	0.92691	0.87141
DDM		0/2	6/6	0.94743	0.99699	0/0	4/4	0.92379	0.8717

Gradual drift

Classifier		Naive Bayes		Perceptron	
Dataset		Rotating hyperplane		Rotating hyperplane	
Wr	Wc	Accuracy	Detection number	Accuracy	Detection number
50	50	0.904	42	0.90332	10
100	50	0.90367	43	0.90345	10
450	50	0.90427	33	0.90127	11
100	100	0.90367	42	0.90254	13
200	100	0.9028	46	0.90135	14
DDM		0.90177	34	0.90501	7

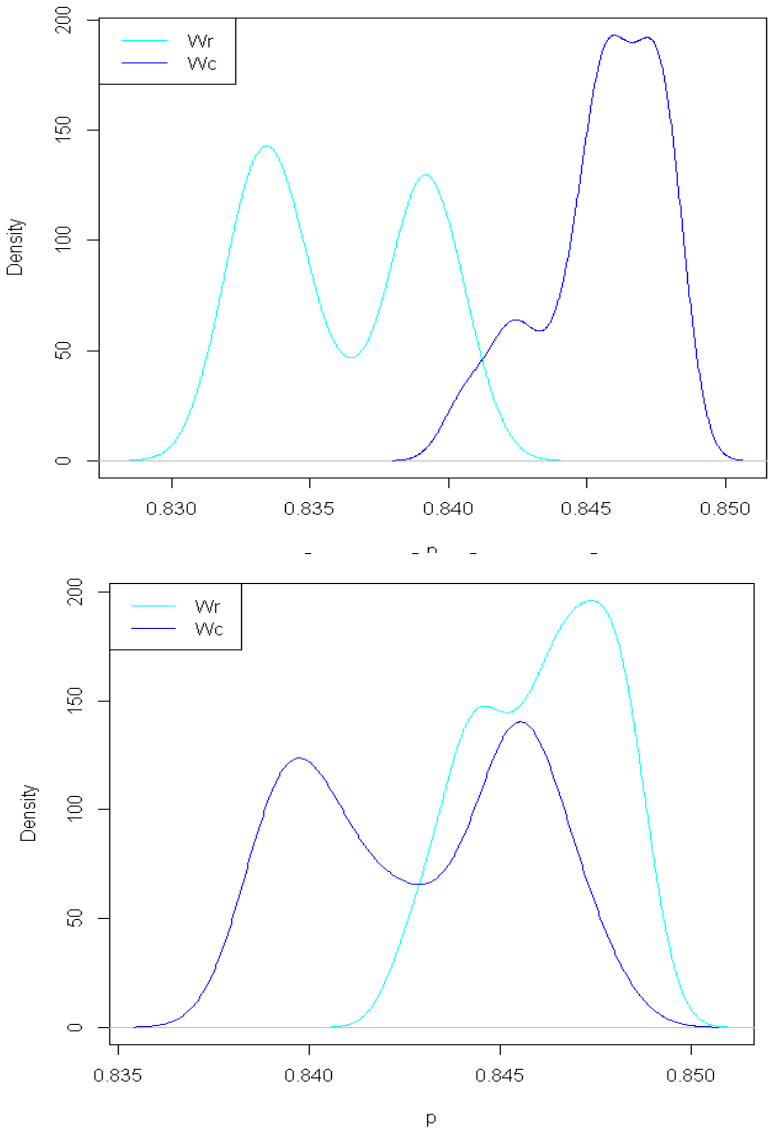


Figure 8 Distribution of value p in two windows when data stream is stationary

We got rather good results on stationary dataset using our method, mainly because our first test on DDM criterion filtered a lot of noises. The incremental global accuracy/ error rate moves more and more slightly as number of instances increases. It caused MODL method to focus on difference in distribution from tiny value interval (in our case between 0.83 and 0.85 for accuracy) comparing with their absolute values (Figure 9). MODL noticed slight changes and then our algorithm became too sensitive. Since detections were alarmed, the classifier was reinitialized and the detector faced the same situation again, so false drift detections frequently appeared. That is why there were still several false detections and we got much more detections on gradually changing dataset.

It seems that large window sizes produce less detection alarms. It takes more instances to fill both windows. There is not an evident difference between different window sizes used for our method.

As shown in Figure 9, this criterion reacted to concept drift so slowly for the previous SEA dataset with drift. In this case, our method monitors evaluation of global incremental error rate as indicator; the delay is more significant than DDM. But the difference in accuracy between tested and alternative detectors is almost negligible, indicating that it is not MODL discretization that caused the delay.

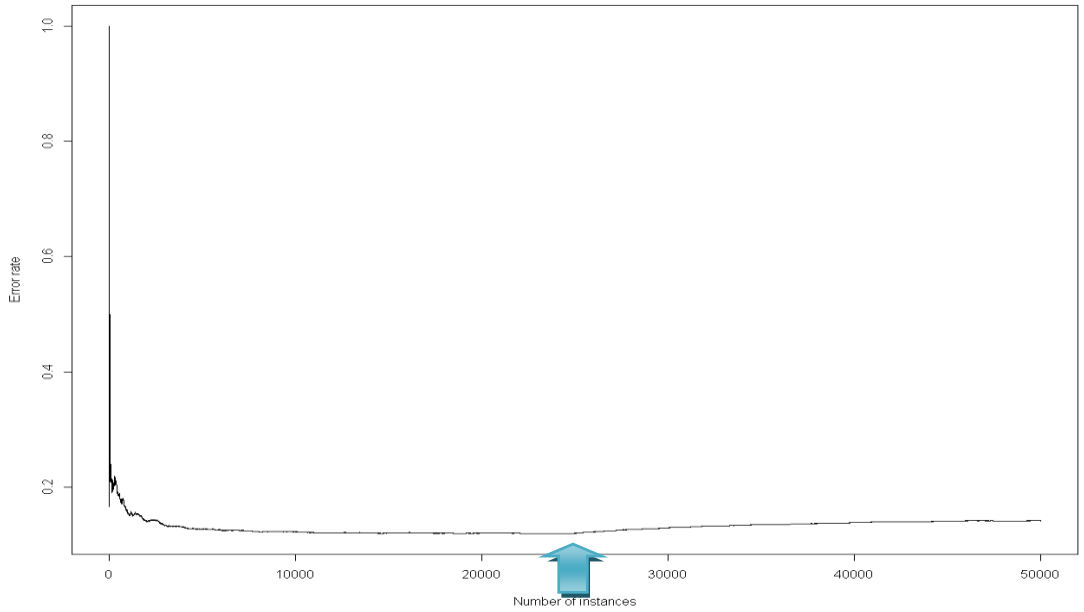


Figure 9 Global/ incremental error rate SEA without drift detection

7.4 Score for a target class

Without drift

Classifier		Naïve Bayes		Perceptron	
		Number of false detections		Number of false detections	
Wr	Wc	LED	SEA	LED	SEA
50	50	2	0	0	1
100	50	0	0	0	1
450	50	0	0	0	0
100	100	0	0	0	9
200	100	0	0	0	0
DDM		0	0	0	0

Sudden drift

Classifier		Naive Bayes				Perceptron			
Detector		Correct detection number/Total detection number		Accuracy		Correct detection number/Total detection number		Accuracy	
Wr	Wc	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER	SEA	STAGGER
50	50	1/46	3/5	0.94702	0.99722	0/0	6/6	0.92408	0.87153
100	50	0/15	5/7	0.94535	0.99751	0/0	10/10	0.92484	0.87133
450	50	0/6	3/3	0.94682	0.99743	0/0	3/3	0.92009	0.87172
100	100	1/30	5/7	0.94687	0.99727	0/0	3/3	0.92395	0.87165
200	100	0/7	5/7	0.94424	0.99707	0/0	3/3	0.92539	0.87163
DDM		0/2	6/6	0.94743	0.99699	0/0	4/4	0.92379	0.8717

Gradual drift

Classifier		Naive Bayes		Perceptron	
		Accuracy	Detection number	Accuracy	Detection number
Wr	Wc	Rotating hyperplane		Rotating hyperplane	
50	50	0.89764	100	0.90219	17
100	50	0.88836	95	0.90373	10
450	50	0.87093	26	0.9028	12
100	100	0.8862	30	0.90264	13
200	100	0.88977	53	0.90218	13
DDM		0.90177	34	0.90501	7

We also got rather good results on stationary dataset using our method, mainly because our first filter on DDM criterion eliminates a lot of noises and just a few level values were calculated.

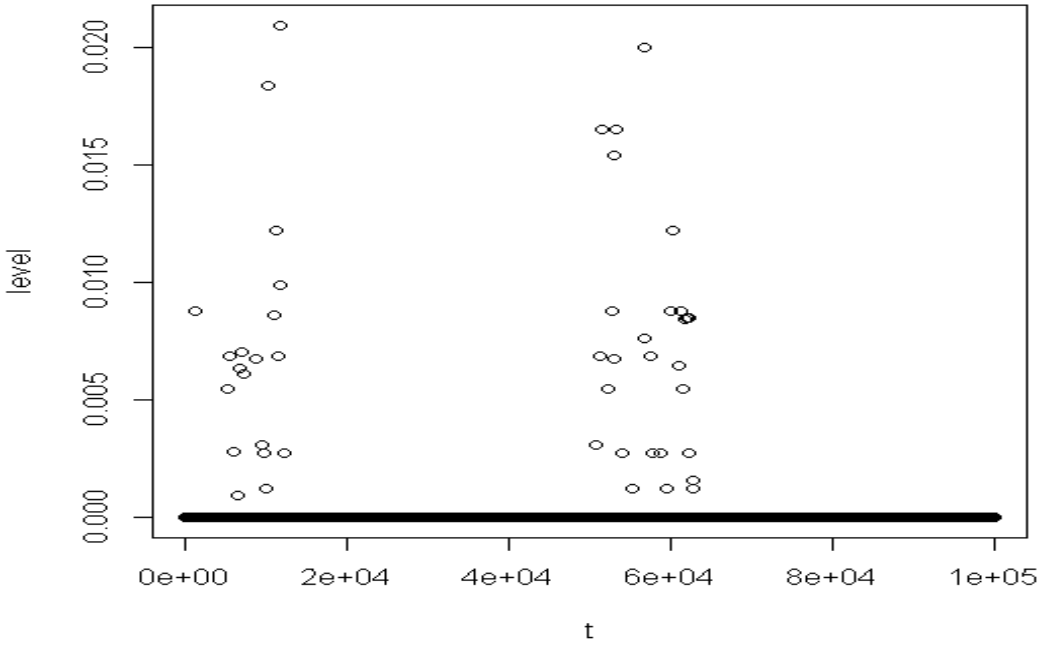


Figure 10 MODL level value over time with detector (reference window size=50 ; current window size=50) and Naive Bayes classifier on SEA sudden drift dataset

On the dataset with a sudden drift shown in Figure 10, our method still had problem on dealing with classifier's instable state. It detected many false drifts at the beginning of the learning process. Even though the performance of the classifier becomes stable, the classifier's structure inside and parameters are still changing, which influences method's indicator value.

On dataset with gradual drift, our method produced many unnecessary and badly positioned drift detections disturbing classifier's learning process. The indicator is influenced by not only incoming instance's properties but also parameters of the classifier (which are always changing over time in case of gradual drifts).

7.5 Conclusion

On general, our method did not work better on detection than DDM method using the three indicators directly in our experiences. The “right prediction” indicator is too robust to changes; the second one, global incremental error rate, produced results almost the same as DDM’s; the third one score of a certain class is too variant which can not be treated by our first filter.

Our method also showed an important dependency on dataset and classifier types. Classifiers keeping short example memory like Perceptron do not fit method’s hypothesis of stable performance state. And for example, in our experiences, MODL level detector worked much better on STAGGER dataset than SEA dataset with sudden drift.

8 Conclusion

During this internship, firstly we briefly introduced our problem “concept drift” and studied several strategies and methods widely used to solve it. Then, we concentrated our research direction on method monitoring classifier’s prediction performance. Inspired by a method calculating MODL level on instances’ variable distribution change, we developed our method using MODL on two observation windows of different performance indicators.

MODL is a good method for discretization and a measure to detect difference on distribution with no matter which kind of variable. However, in our case our indicator values are not variants. Considering the reaction time and detector’s complexity, the sample size is rather small. MODL is much more sensitive to noises. We had to use a preliminary filter which is the same as DDM method’s suspect drift detection, to reduce number of false alarms. Our method is too dependant to the DDM filter and few MODL level values were calculated. In the same time we lost our very first advantage that our method has not a hypothesis on indicator value’s distribution.

Several propositions for future research:

- Apply smoothing on indicator value or MODL level value
- Use several different indicators at the same time
- Use larger reference window size and do resume on the reference window then calculate the MODL level

Personally, this internship is my first experience on research domain. I got deeper comprehension on machine learning especially on online/incremental classification. Concept detection is a very difficult problem where we met a lot of problems during my internship. And I have learnt a lot on getting over them and on research methodology, which will be useful for my future career.

9 Bibliography

Albert Bifet, G. H. (2010). Fast Perceptron Decision Tree Learning from Evolving Data Streams. *PAKDD 2010, Part II, LNAI 6119*, (pp. 299-310).

Albert Bifet, G. H. (2011, May). *http://heanet.dl.sourceforge.net/project/moa-datastream/documentation/StreamMining.pdf.* Retrieved from DATA STREAM MINING A Practical Approach.

Albert Bifet, G. H. (2009). New ensemble methods for evolving data streams. *In John F. Elder IV, Fran¸coise Fogelman-Soulie, Peter A. Flach, and Mohammed Javeed Zaki, editors, KDD* (pp. 139–148). ACM.

Alexis Bondu, B. G.-L. (2010). Density estimation on data streams : an application to Change Detection. *EGC (Extraction et Gestion de Connaissances).* Hammamet, Tunisie.

Alexis Bondu, B. G.-L. (2010). Density estimation on data streams : an application to Change Detection. *EGC (Extraction et Gestion de Connaissances).* Hammamet, Tunisie.

Alexis Bondu, M. B. (2011). A Supervised Approach for Change Detection in Data Stream. *IJCNN (International Joint Conference on Neural Networks)* .

Anton Dries, U. R. (2009). Adaptive concept drift detection. *Statistical Analy Data Mining, Vol. 2, No. 5-6*, (pp. 311-327).

Arlot, S., & Celisse, A. (2010). Segmentation in the mean of heteroscedastic data by cross-validation. *Statistics and Computing* .

Baena-Garcia, M. d.-A.-B. (2006). Early drift detection method. *Proc. ECML/PKDD 2006, Work. Knowledge Discovery from Data Streams*, (pp. 77–86).

Bifet, A. (2009). *Thesis: Adaptive Learning and Mining for Data Streams and Frequent Patterns.* Universitat Politècnica de Catalunya.

Bottou, L. (2004). Stochastic Learning. *Advanced Lectures on Machine Learning* , 146-168.

C. Alippi, G. B. (2010). Change Detection Tests Using the ICI rule. *Proc. of IJCNN* , 1-7.

C.Alippi, M. (2008). Just-in-time Adaptive Classifiers. Part I. Detecting non-stationary Changes. *IEEE-Transactions on Neural Networks, Vol. 19, No. 7* , 1145 - 1153.

C.Alippi, M. (2008). Just-in-time Adaptive Classifiers. Part II. Designing the Classifier. *IEEE Transactions on Neural Networks, Volume 19, Issue 12*, (pp. 2053 - 2064).

Covertype. (n.d.). Retrieved from http://archive.ics.uci.edu/ml/datasets/Covertype

G. Hulten, L. S. (2001). Mining timechanging data streams. *7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 97–106). San Francisco, CA: ACM Press.

Gama, J. M. (2004). Learning with drift detection. *Proc. 17th Brazilian Symp. Artificial Intelligence,*, (pp. 285–295).

Gama, J. R. (2003). Accurate decision trees for mining high-speed data streams. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 523–528). ACM New York, NY, USA.

Gama, J., Medas, P., & Rocha, R. (2004). Forest trees for on-line data. *SAC '04: Proceedings of the 2004 ACM symposium* (pp. 632–636). New York, NY, USA: ACM Press.

Gavalda, A. B. (2007). Learning from time-changing data with adaptive windowing. *Proc. of SIAM int. conf. on Data Mining (SDM'07)*, (pp. 443–448).

Granger, J. C. (1986). Incremental learning from noisy data. *Machine Learning, 1(3)*, (pp. 317–354).

Haixun Wang, W. F. (2003). Mining concept-drifting data streams using ensemble classifiers. *In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos* (pp. 226–235). ACM.

Harries, M. (1999). *Splice-2 comparative evaluation: Electricity pricing.* Technical report, The University of SouthWales.

http://archive.ics.uci.edu/ml/datasets.html. (n.d.).

http://www.orange.com/en/group. (n.d.).

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 97–106). San Francisco, CA: ACM Press.

Kasaei, R. D. (2008). On Automatic Threshold Selection in Regression Method for Change Detection in Remote Sensing Images. *In The 4th International Symposium on Telecommunications, IST, Tehran, Iran, Aug.* .

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting Change in Data Streams. *Proceedings of the 30th VLDB Conference.* Toronto, Canada.

Kim, W. N. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. *KDD '01:Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 377–382). New York, NY, USA: ACM Press.

Kim., N. S. (2001). A streaming ensemble algorithm (sea) for largescale classification. *KDD '01: Proc. of the 7th ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining* (pp. 377–382). ACM.

Klinkenberg, R. a. (1998). Adaptive Information Filtering: Learning in the Presence of Concept Drifts. *Workshop Notes of the ICML/AAAI-98 Workshop Learning for Text Categorization* (pp. 33--40). Menlo Park, CA, USA: Sahami, Mehran and Craven, Mark and Joachims, Thorsten and McCallum, Andrew, AAAI Press.

Kubat., G. a. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning,23(1)* , 69–101.

Kyosuke Nishida, K. Y. (2007). Detecting concept drift using statistical testing. *Proceeding DS'07 Proceedings of the 10th international conference on Discovery science.* Springer-Verlag Berlin, Heidelberg.

Langley, P. a. (1992). An analysis of bayesian classifiers. *International conference on Artificial Intelligence, AAAI* , 223-228.

Last., M. (2002). Online classification of nonstationary data streams. *Intelligent Data Analysis* , 6(2):129–147.

LED Display Domain Data Set . (n.d.). Retrieved from UCI Machine Learning repository: http://archive.ics.uci.edu/ml/datasets/LED+Display+Domain

Leo Breiman, J. H. (1984). Classification and Regression Trees. Wadsworth International Group: Belmont, California.

Maloof, J. K. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research, 8* , 2755–2790.

Nishida, K. Y. (2005). ACE: Adaptive classifiers-ensemble system for concept-drifting environments. *Proc. 6th Int. Work. Multiple Classifier Systems*, (pp. 176–185).

Page, E. (1954). Continuous Inspection Scheme. *Biometrika 41* .

Pedro Domingos, G. H. (2000). Mining HighSpeed Data Streams. *KDD' 00 Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* , 71-80.

Poker Hand. (n.d.). Retrieved from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Poker+Hand

Rakesh Agrawal, T. I. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering, 5(6):*, (pp. 914-925).

Roberts., S. W. (2000). Control chart tests based on geometric moving. *Technometrics, 42(1)* , 97–101.

Rosenblatt, F. (1957). The Perceptron--a perceiving and recognizing automaton. *Cornell Aeronautical Laboratory* , Report 85-460-1.

Russell, N. C. (2001). Online Bagging and Boosting. *Artificial Intelligence and Statistics 2001* (pp. 105-112). Morgan Kaufmann.

Stefanowski, D. B. (2011). Accuracy Updated Ensemble for Data Streams. *Hybrid Artificial Intelligence Systems, 6th International Conference, HAIS 2011, Proceedings, Part II*, (pp. 155-163). Wrocław, Poland.

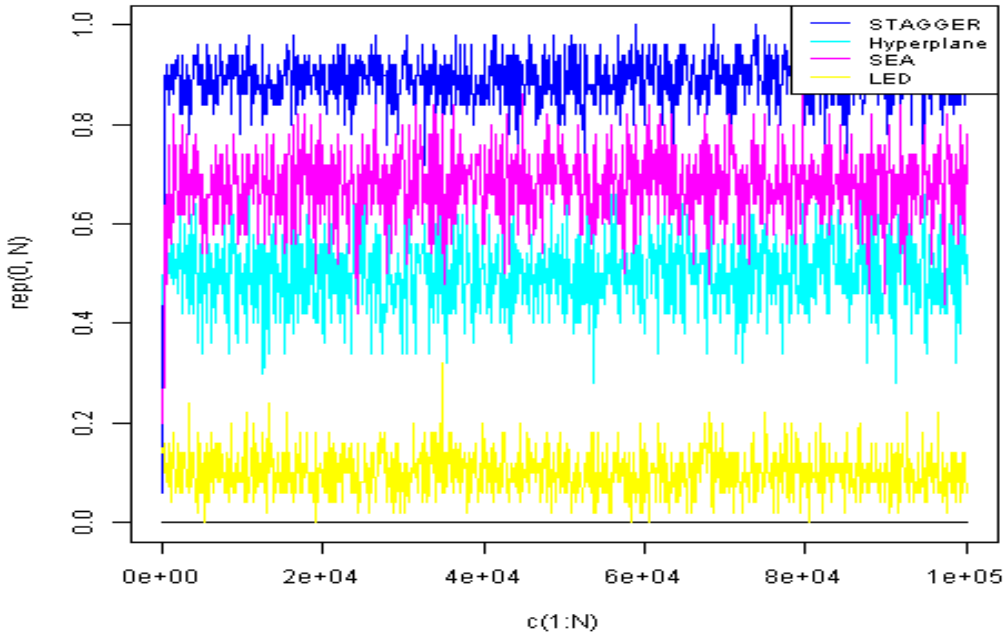
Wald, A. (1947). *Sequential analysis.* John Wiley and Sons, Inc.

Waveform Database Generator (Version 2) DataSet. (n.d.). Retrieved from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+2%29

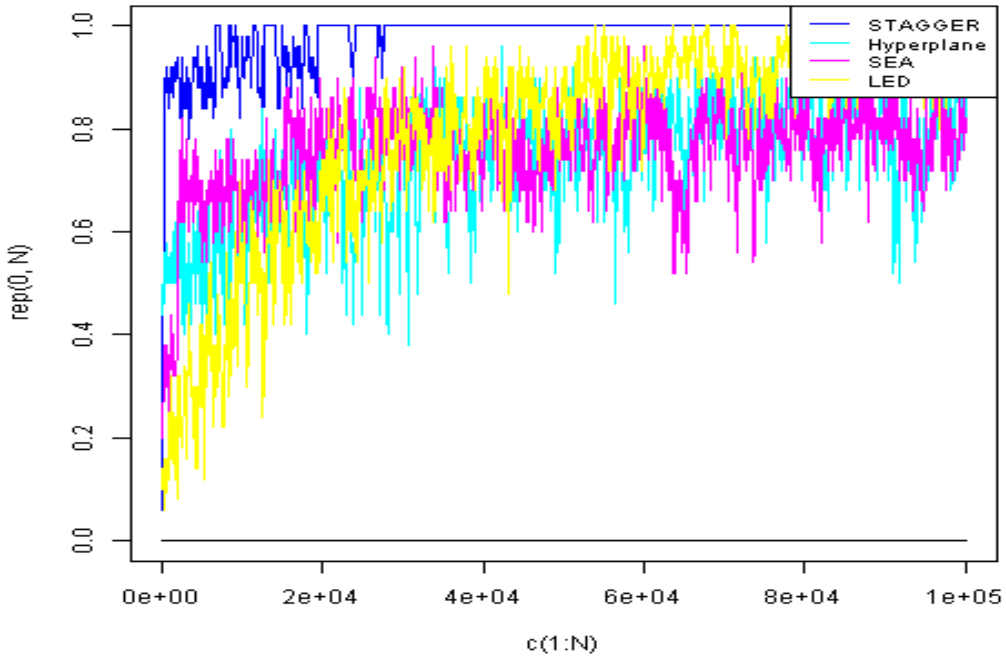
Žliobaite, I. (2010). ADAPTIVE TRAINING SET FORMATION. *VILNIUS UNIVERSITY.* Vilnius.

10 Annexes

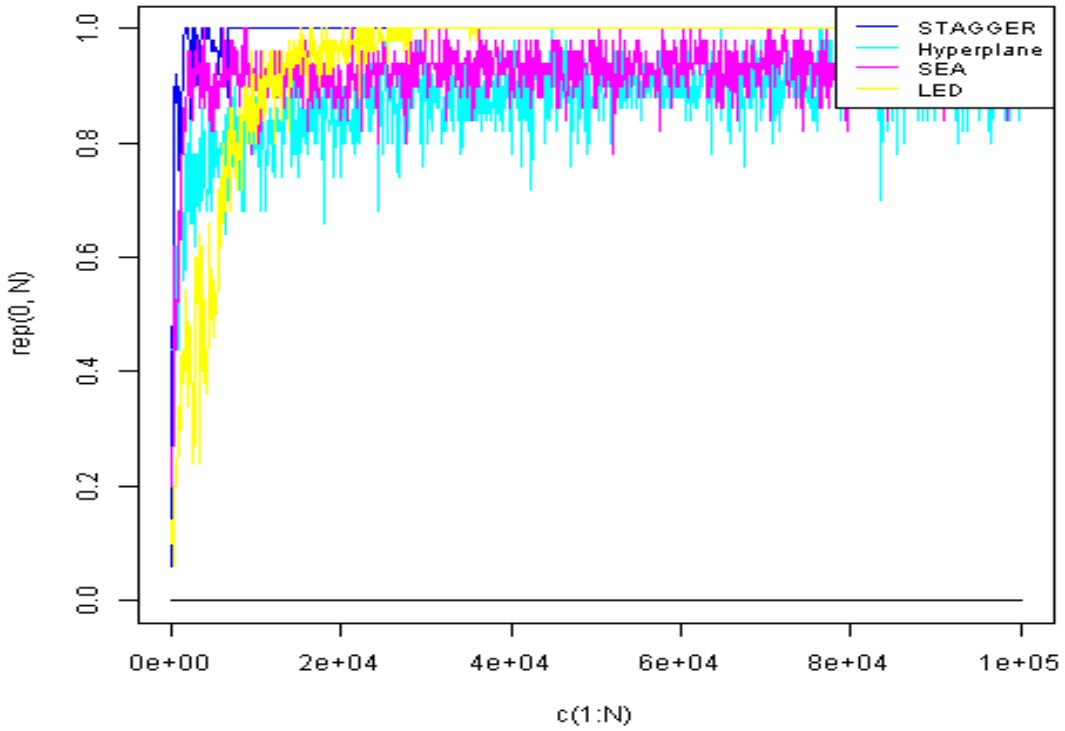
10.1 Accuracy of incremental learner applied on source without drift



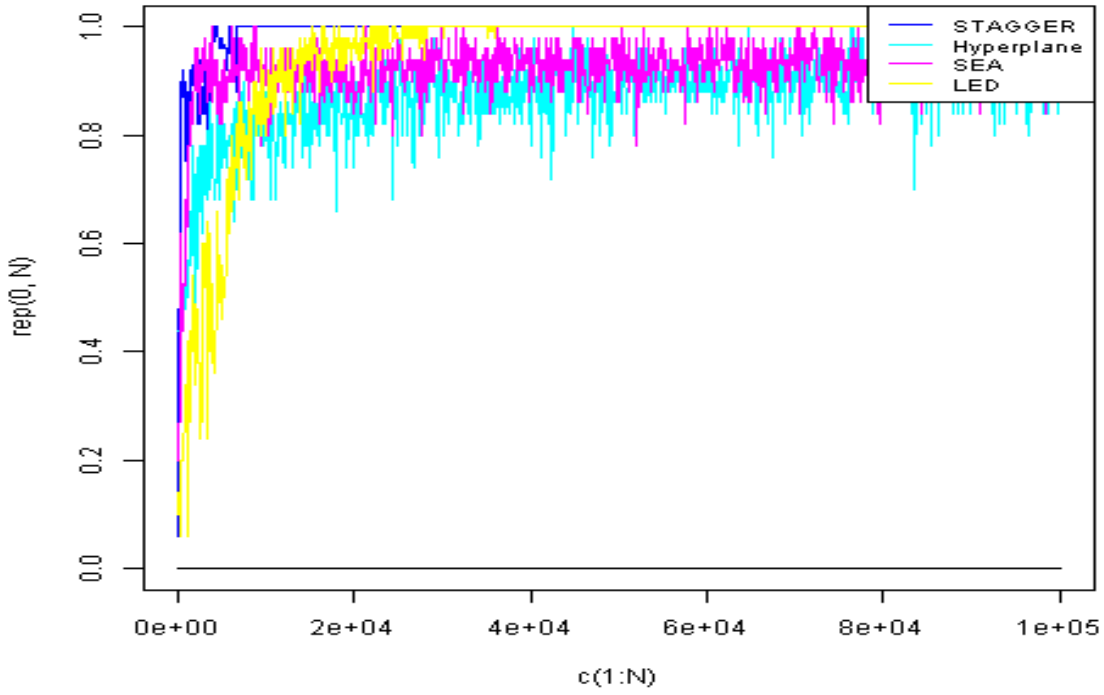
SDG



Perceptron

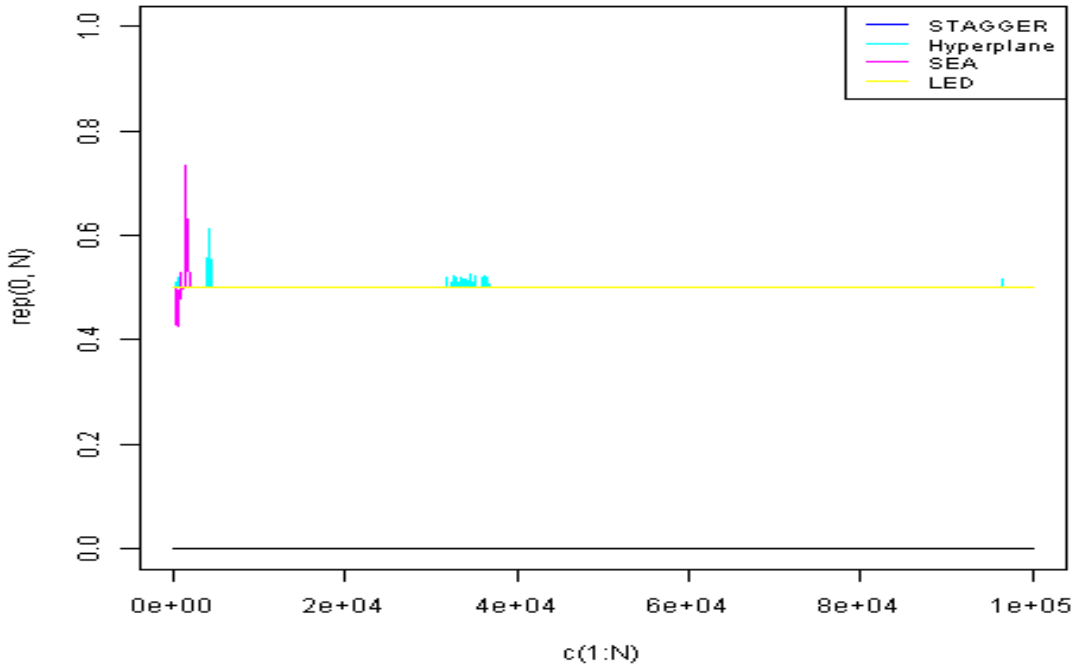


Naive Bayes

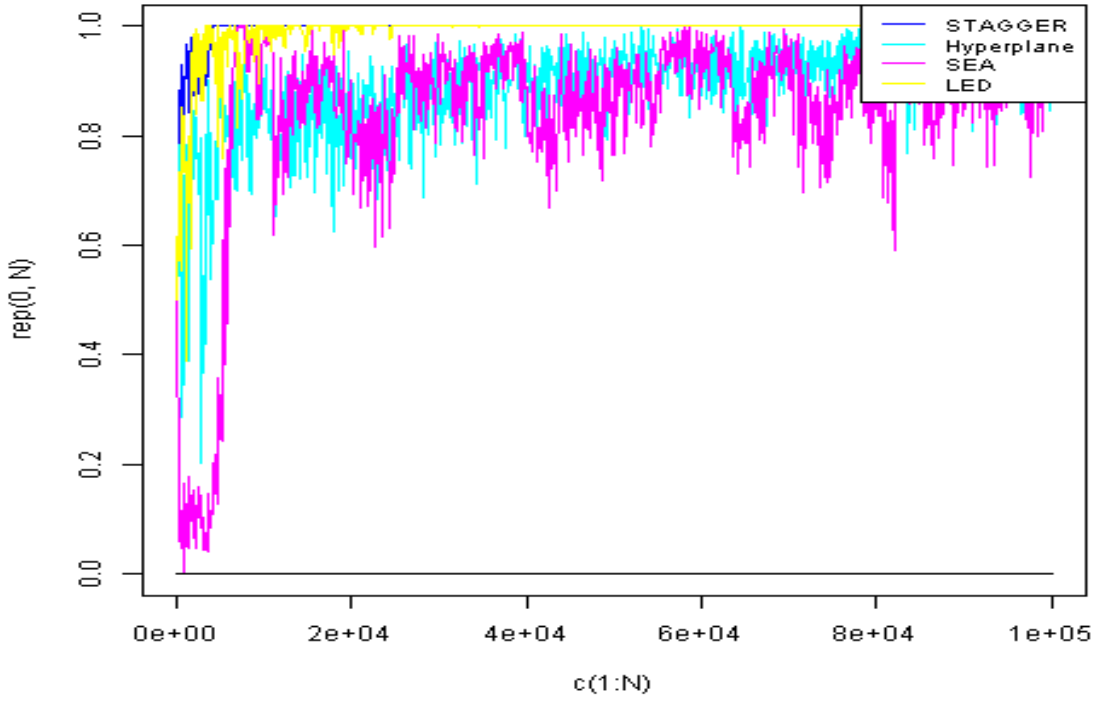


Hoeffding Tree

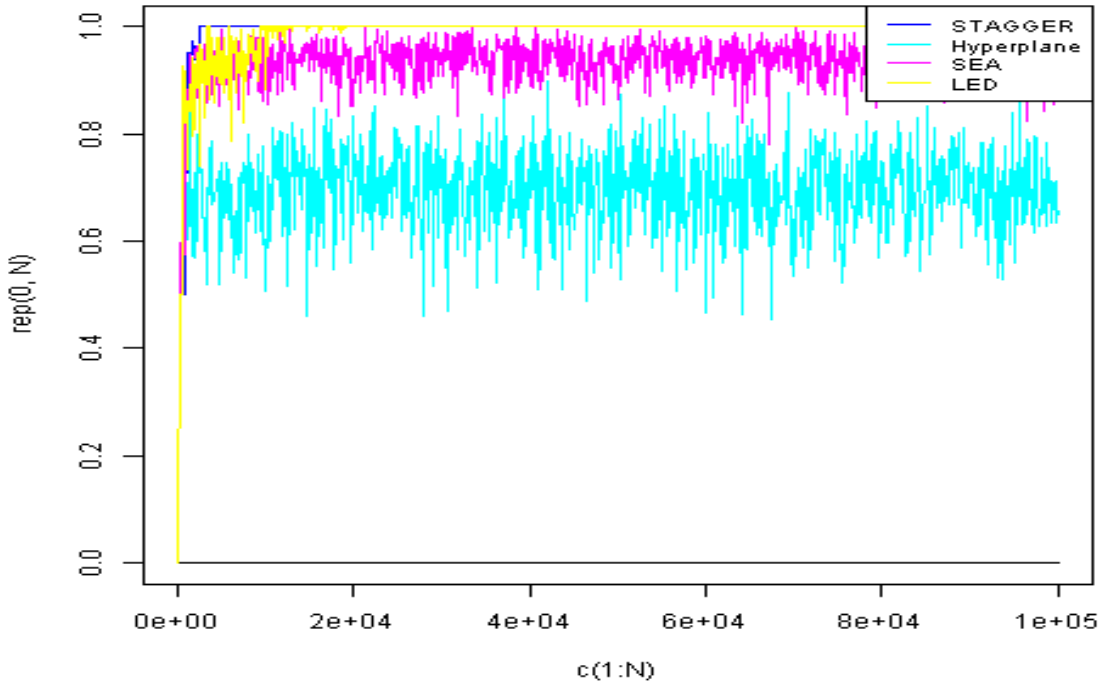
10.2 AUC of incremental learner applied on source without drift



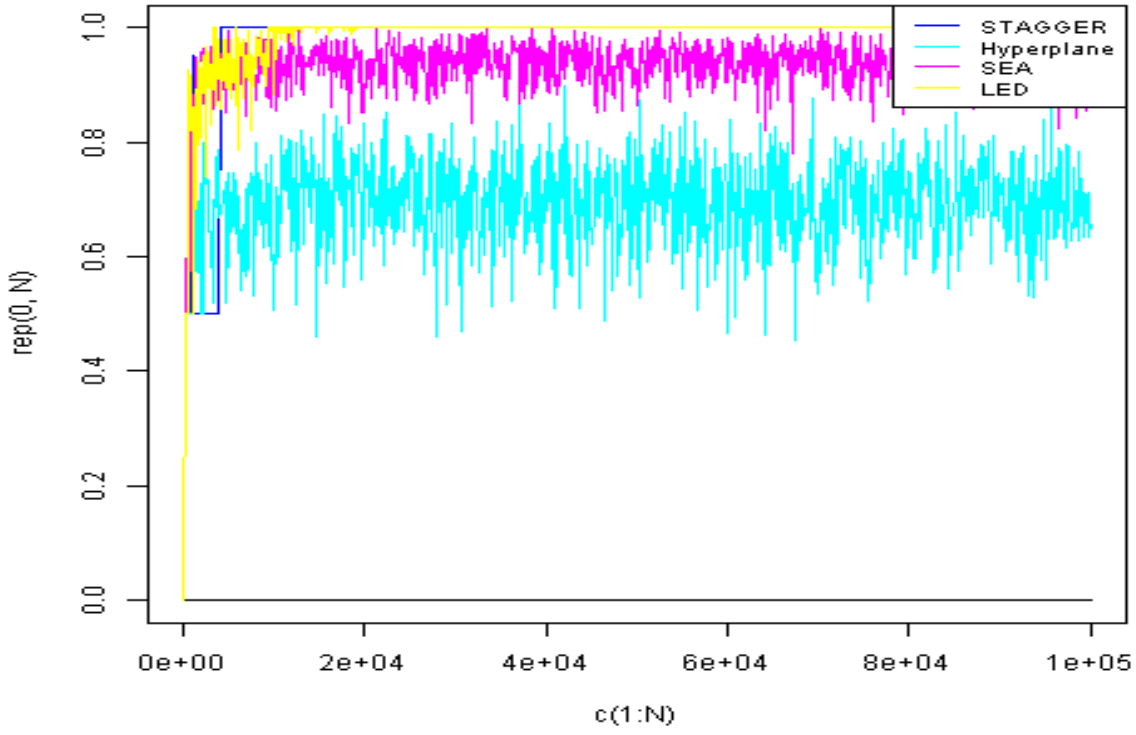
SGD



Perceptron

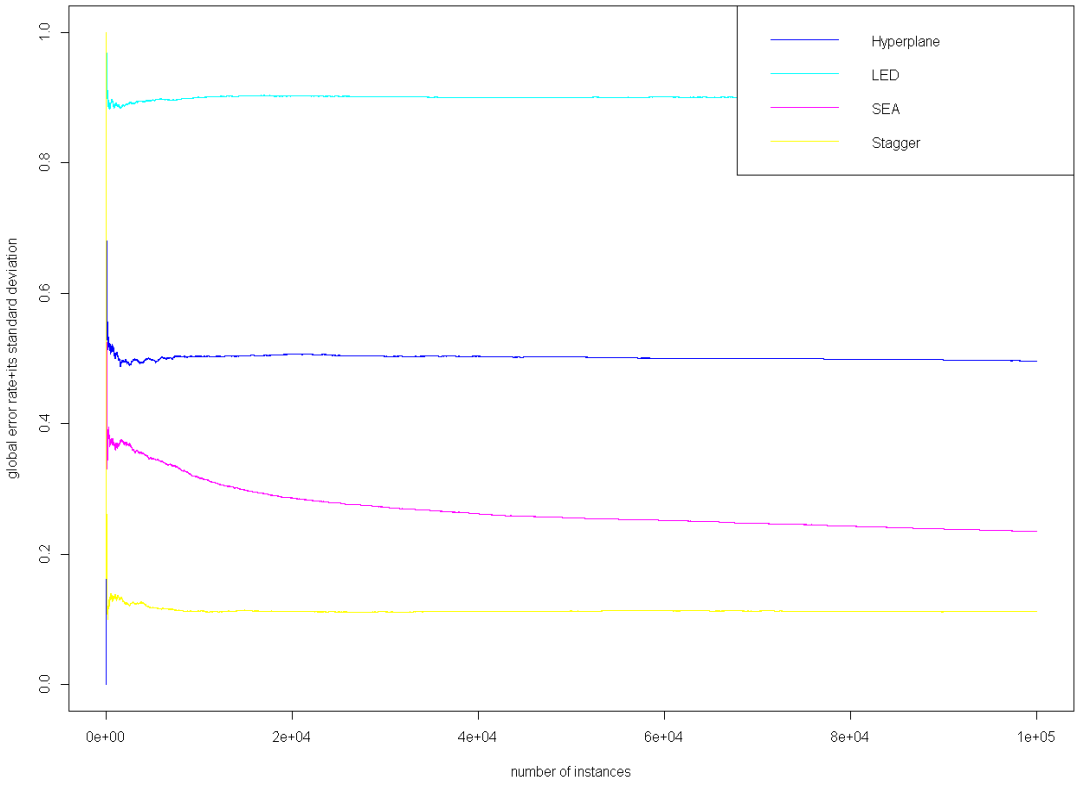


Naive Bayes

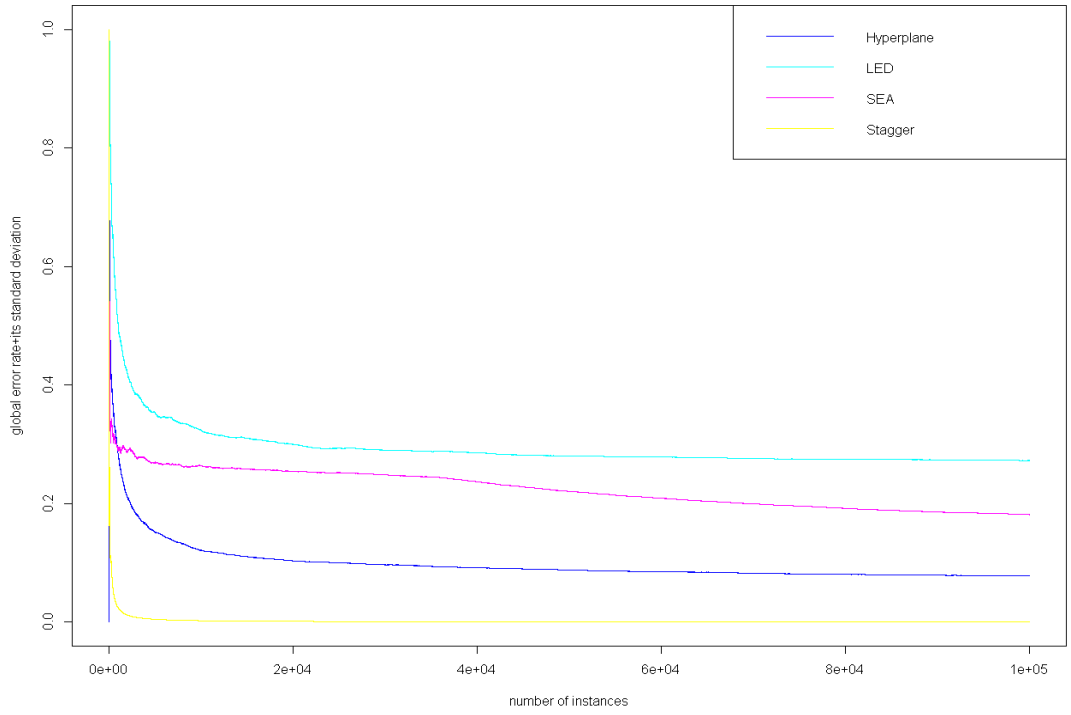


Hoeffding Tree

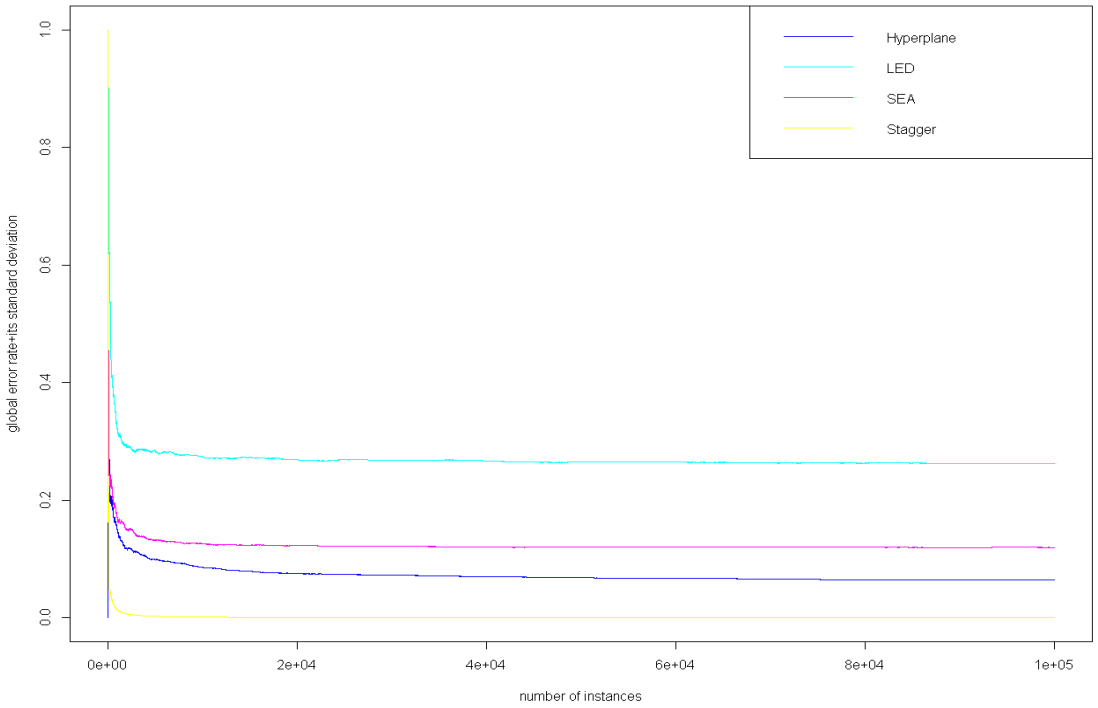
10.3 DDM p+s of incremental SGD learner applied on source without drift



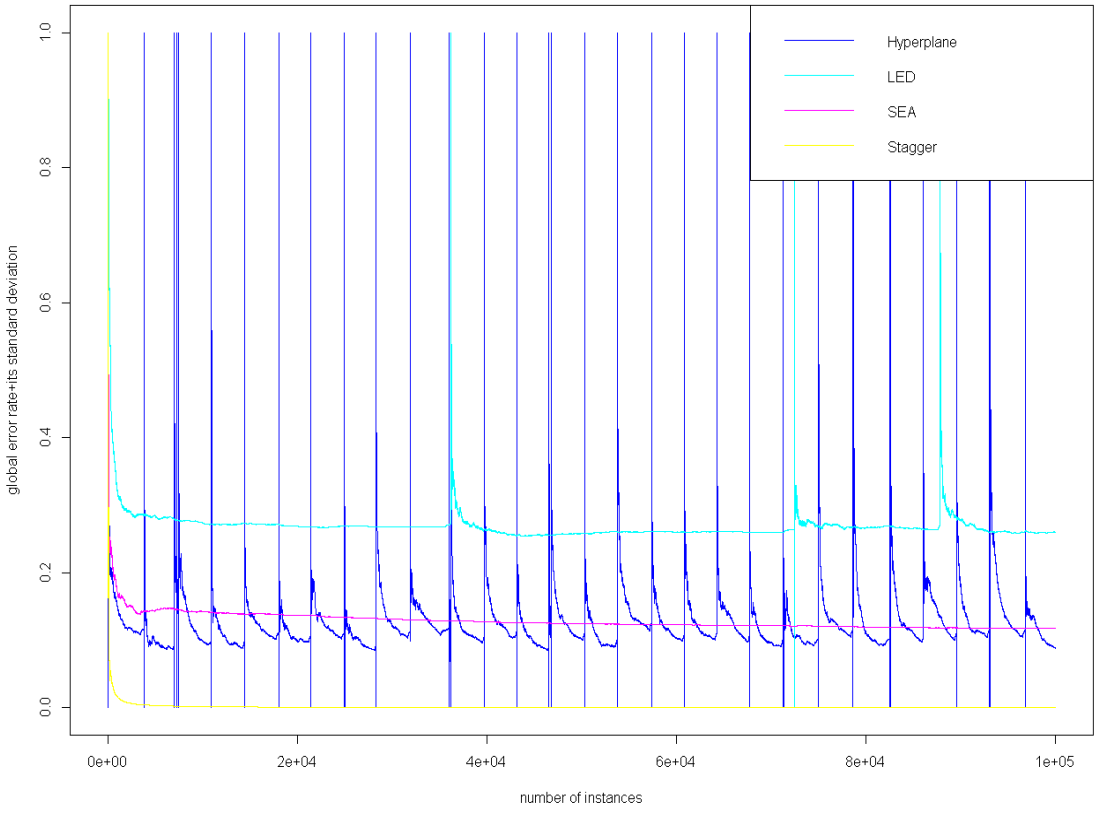
SGD



Perceptron

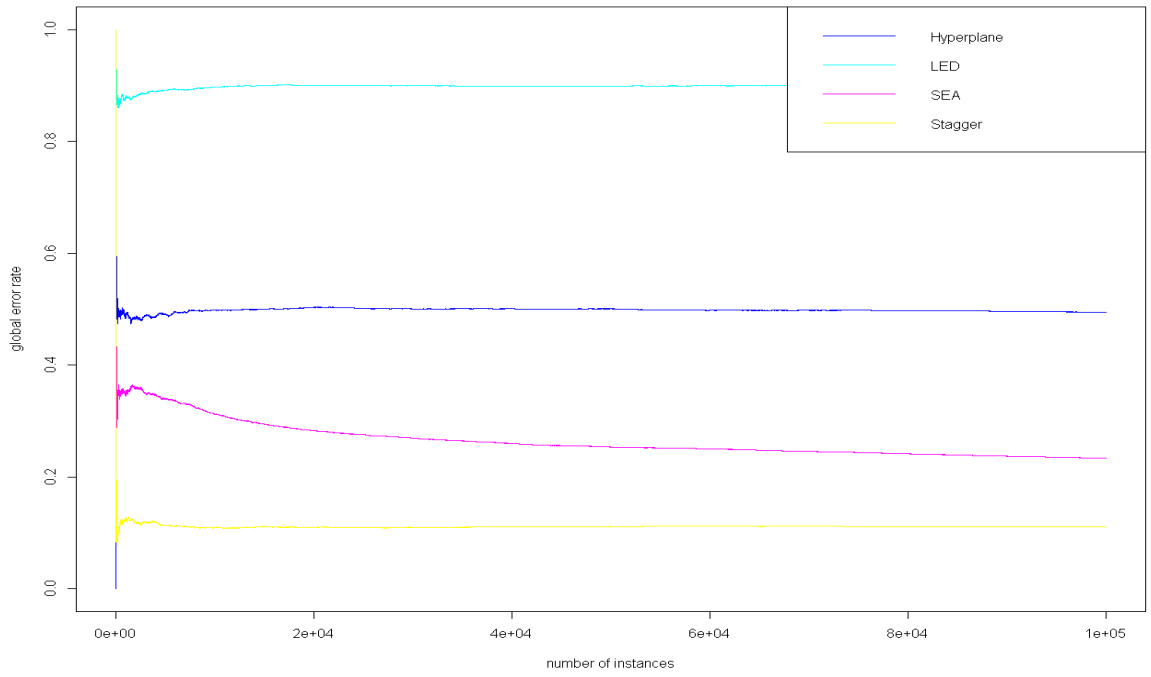


Naive Bayes

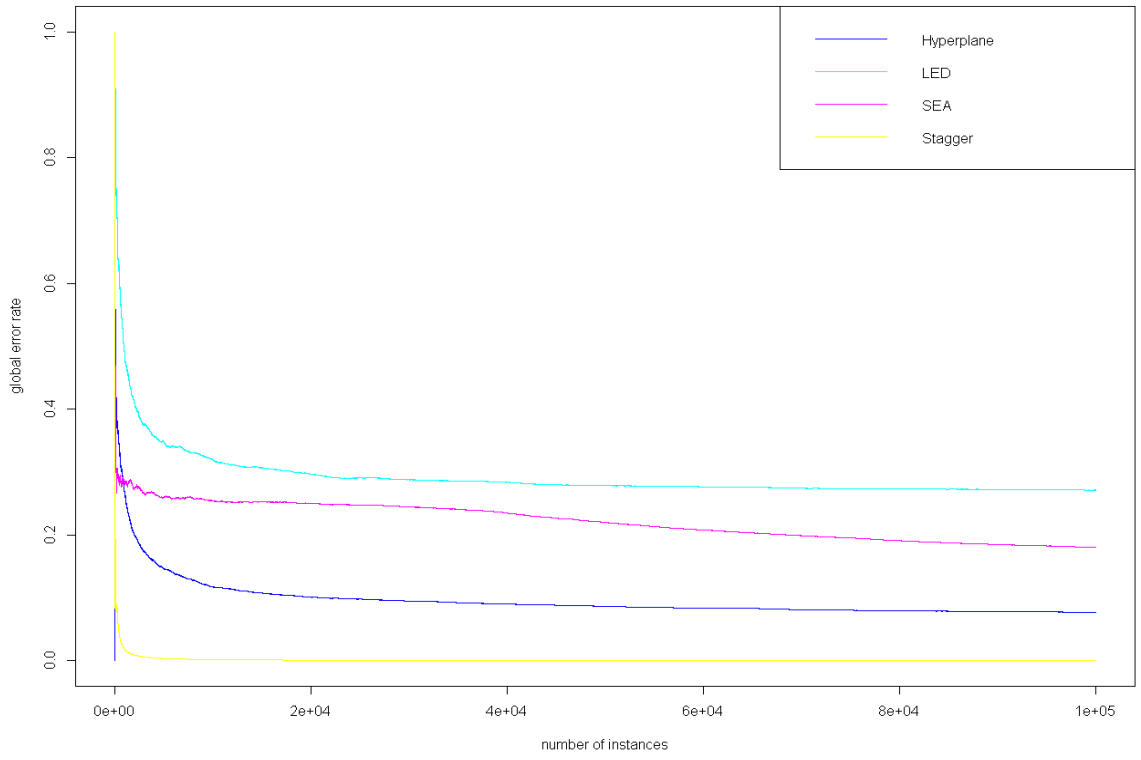


Hoeffding Tree

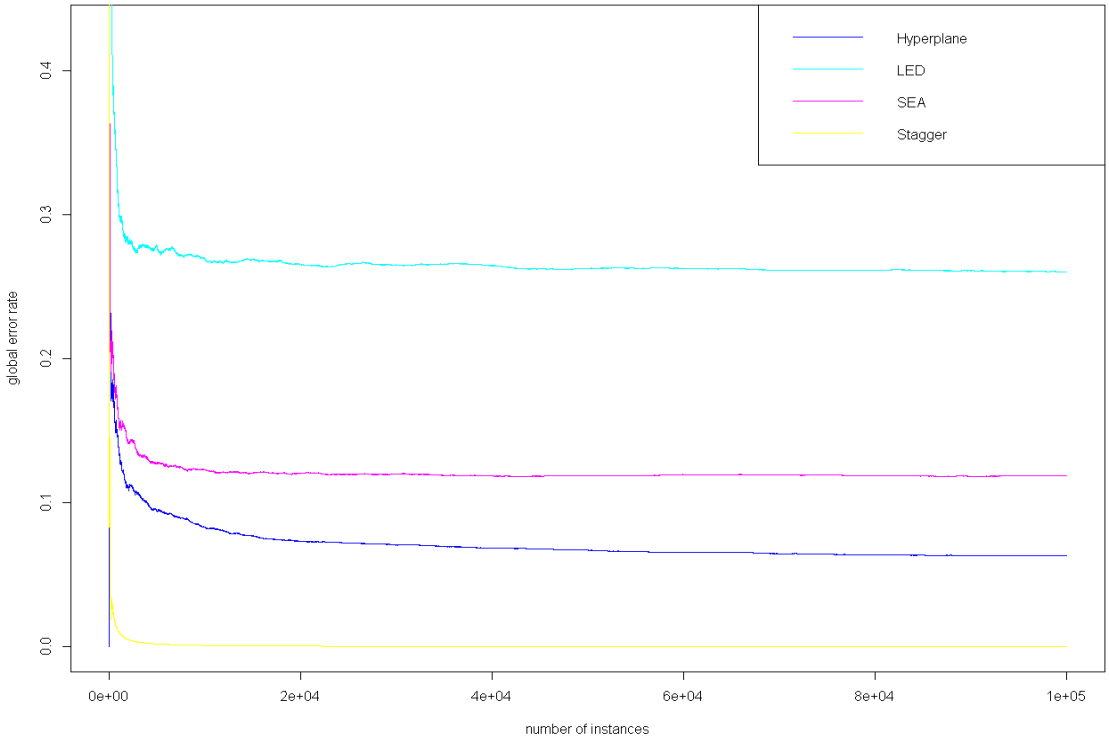
10.4 Global/incremental error rate of incremental learner applied on source without drift



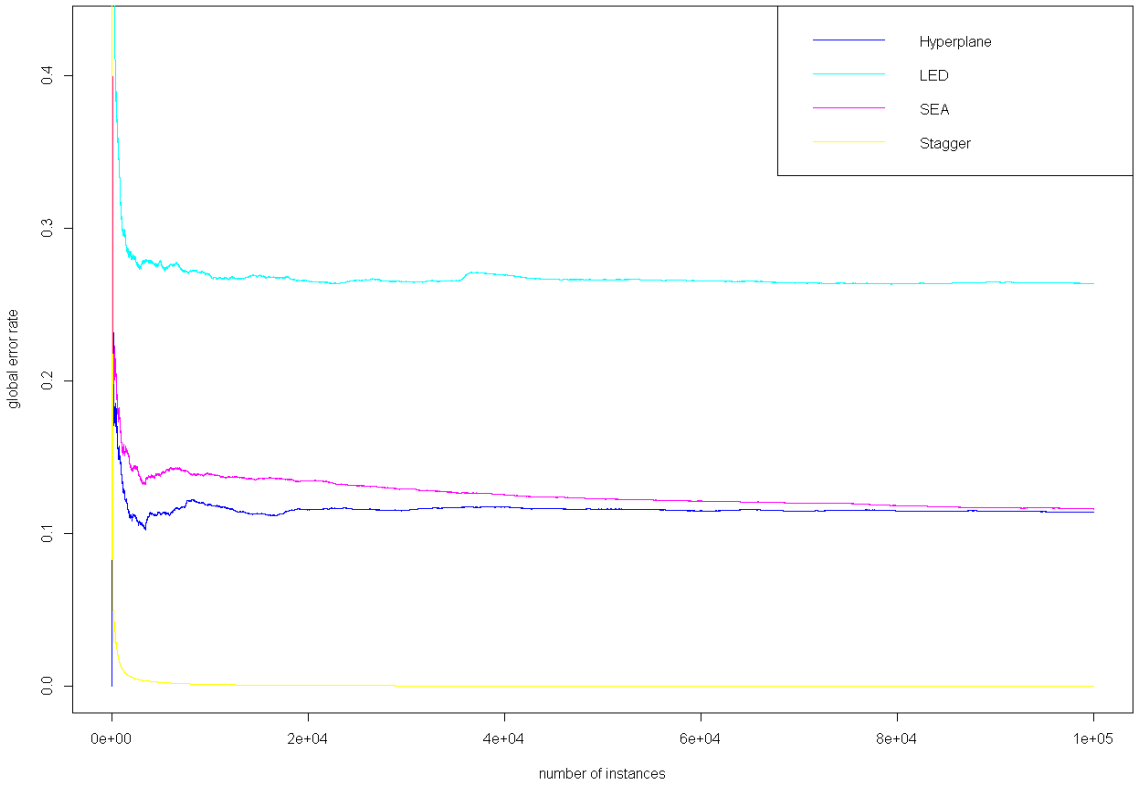
SGD



Perceptron

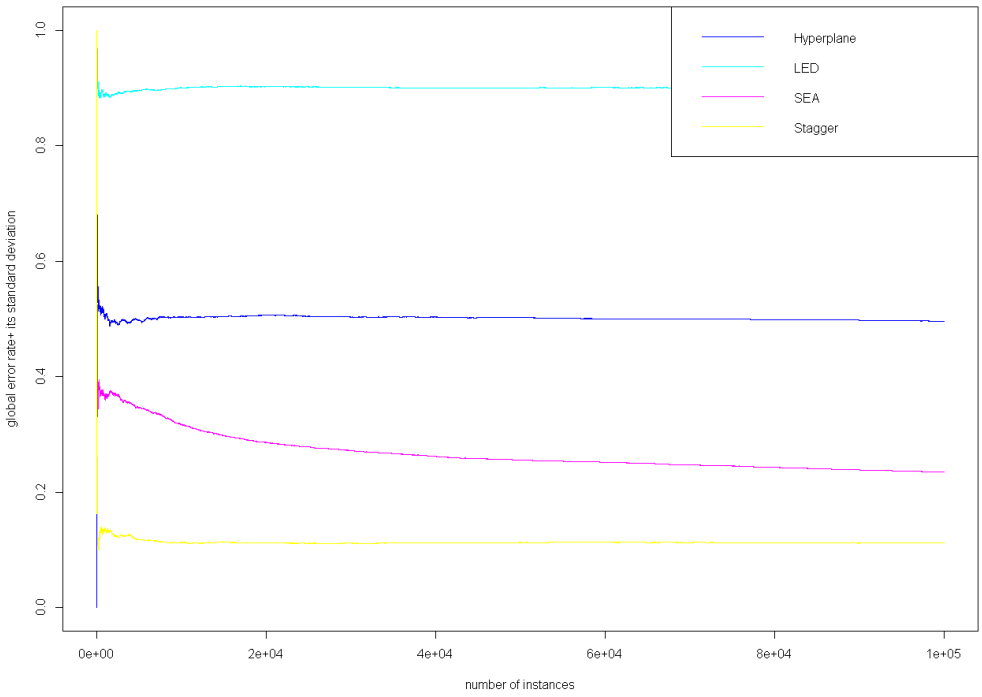


Naive Bayes

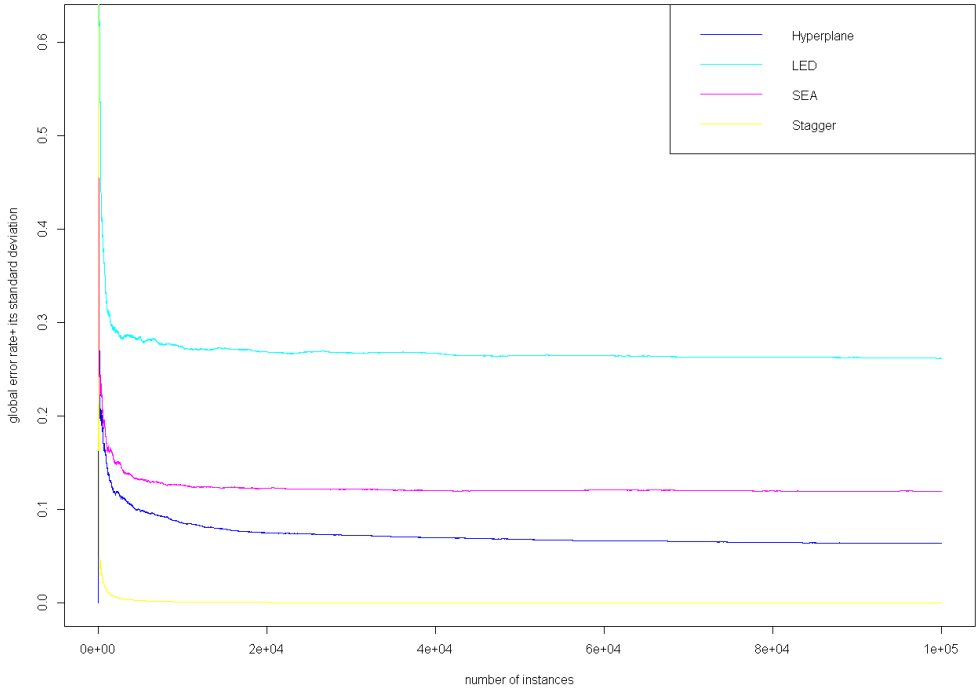


Hoeffding Tree

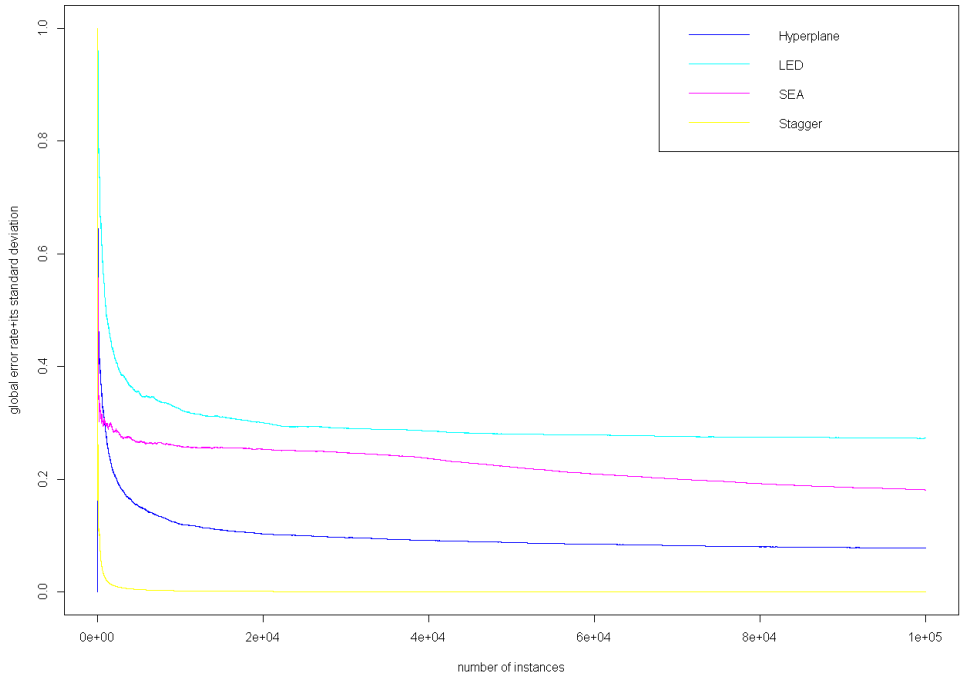
10.5 Error rate+its standard deviation of incremental learner applied on source without drift



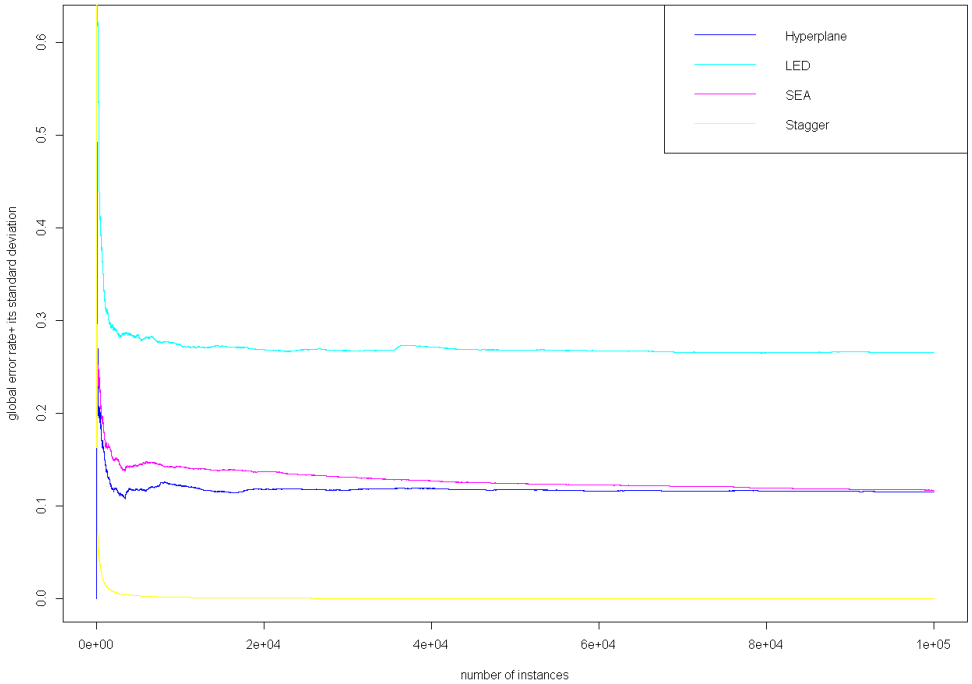
SGD



Naive Bayes



Perceptron



Hoeffding Tree