ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220974771

Learning with Drift Detection

CONFERENCE PAPER *in* INTELLIGENT DATA ANALYSIS · SEPTEMBER 2004

Impact Factor: 0.61 · DOI: 10.1007/978-3-540-28645-5_29 · Source: DBLP

CITATIONS
257

READS
534

4 AUTHORS, INCLUDING:

João Gama
University of Porto
280 PUBLICATIONS **3,095** CITATIONS
SEE PROFILE

Gladys Castillo
University of Aveiro
30 PUBLICATIONS **533** CITATIONS
SEE PROFILE

Pedro Pereira Rodrigues
University of Porto
87 PUBLICATIONS **857** CITATIONS
SEE PROFILE

Available from: João Gama
Retrieved on: 08 February 2016

Learning with Drift Detection

João Gama[1,2], Pedro Medas[1], Gladys Castillo[1,3], and Pedro Rodrigues[1]

[1] LIACC - University of Porto
Rua Campo Alegre 823, 4150 Porto, Portugal
[2] Fac. Economics, University of Porto
[3] University of Aveiro
jgama@liacc.up.pt,pmedas@liacc.up.pt, gladys@mat.ua.pt

Abstract. Most of the work in machine learning assume that examples are generated at random according to some stationary probability distribution. In this work we study the problem of learning when the class-probability distribution that generate the examples changes over time. We present a method for detection of changes in the probability distribution of examples. A central idea is the concept of *context*: a set of contiguous examples where the distribution is stationary. The idea behind the drift detection method is to control the online error-rate of the algorithm. The training examples are presented in sequence. When a new training example is available, it is classified using the actual model. Statistical theory guarantees that while the distribution is stationary, the error wil decrease. When the distribution changes, the error will increase. The method controls the trace of the online error of the algorithm. For the actual context we define a warning level, and a drift level. A new context is declared, if in a sequence of examples, the error increases reaching the warning level at example k_w, and the drift level at example k_d. This is an indication of a change in the distribution of the examples. The algorithm learns a new model using only the examples since k_w. The method was tested with a set of eight artificial datasets and a real world dataset. We used three learning algorithms: a *perceptron*, a *neural network* and a *decision tree*. The experimental results show a good performance detecting drift and also with learning the new concept. We also observe that the method is independent of the learning algorithm.

Keywords: Concept Drift, Incremental Supervised Learning, Machine Learning

1 Introduction

In many applications, learning algorithms acts in dynamic environments where the data flows continuously. If the process is not strictly stationary (as most of real world applications), the target concept could change over time. Nevertheless, most of the work in machine learning assume that training examples are generated at random according to some stationary probability distribution.

In this work we present a direct method to detect changes in the distribution of the training examples. The method will be presented in the on-line learning

model, where learning takes place in a sequence of trials. On each trial, the learner makes some kind of prediction and then receives some kind of feedback.

A important concept through out this work is the concept of *context*. We define context as a set of examples where the function generating examples is stationary. We assume that the data stream is composed by a set of contexts. Changes between contexts can be gradual - when there is a smothed transition between the distributions; or abrupt - when the distribution changes quickly. The aim of this work is to present a straightforward and direct method to detect the several moments when there is a change of context. If we can identify contexts, we can identify which information is outdated and re-learn the model only with relevant information to the present context.

The paper is organized as follows. The next section presents related work in detecting concept drifting. In section 3 we present the theoretical basis of the proposed method. Section 4 we evaluate the method using several algorithms on artificial and real datasets. Section 5 concludes the paper and present future work.

2 Tracking Drifting Concepts

There are several method in machine learning to deal with changing concepts,[4, 5, 3, 11]. In machine learning drifting concepts are often handled by time windows or weighted examples according to their age or utility. In general, approaches to cope with concept drift can be classified into two categories: *i*) approaches that adapt a learner at regular intervals without considering whether changes have really occurred; *ii*) approaches that first detect concept changes, and next, the learner is adapted to these changes. Examples of the former approaches are *weighted examples* and *time windows* of fixed size. Weighted examples are based on the simple idea that the importance of an example should decrease with time (references about this approach can be found in [4, 5, 7, 8, 11]). When a time window is used, at each time step the learner is induced only from the examples that are included in the window. Here, the key difficulty is how to select the appropriate window size: a small window can assure a fast adaptability in phases with concept changes but in more stable phases it can affect the learner performance, while a large window would produce good and stable learning results in stable phases but can not react quickly to concept changes. In the latter approaches,with the aim of detecting concept changes, some indicators (e.g. performance measures, properties of the data, etc.) are monitored over time (see [4] for a good classification of these indicators). If during the monitoring process a concept drift is detected, some actions to adapt the learner to these changes can be taken. When a time window of adaptive size is used these actions usually lead to adjusting the window size according to the extent of concept drift [4]. As a general rule, if a concept drift is detected the window size decreases, otherwise the window size increases. An example of work relevant to this approach is the FLORA family of algorithms developed by Widmer and Kubat [11]. For instance, FLORA2 includes a window adjustment heuristic for a rule-based clas-

sifier. To detect concept changes the accuracy and the coverage of the current learner are monitored over time and the window size is adapted accordingly.

Other relevant works are the works of R.Klinkenberg and C.Lanquillon, both of them in information filtering. For instance, Klinkenberg and Renz in [4], in order to detect concept drift, propose monitoring the values of three performance indicators: *accuracy*, *recall* and *precision* over time, and then, comparing it to a confidence interval of standard sample errors for a moving average value (using the last M batches) of each particular indicator. Although these heuristics seem to work well in their particular domain, they have to deal with two main problems: *i*) to compute performance measures, user feedback about the true class is required, but in some real applications only partial user feedback is available; *ii*) a considerable number of parameters are needed to be tuned. Afterwards, in [5] Klinkenberg and Joachims present a theoretically well-founded method to recognize and handle concept changes using support vector machines. The key idea is to select the window size so that the estimated generalization error on new examples is minimized. This approach uses unlabeled data to reduce the need for labeled data, it doesn't require complicated parameterization and it works effectively and efficiently in practice. However, it is not independent of the hypothesis language (a support vector machine) and therefore it is not generally applicable.

3 The Drift Detection Method

In most of real-world applications of machine learning data is colected over time. For large time periods, it is hard to assume that examples are independent and identically distributed. At least in complex environments its highly provable that class-distributions changes over time.

In this work we assume that examples arrive one at a time. The framework could be easly extended to situations where data comes on batches of examples. We consider the online learning framework. In this framework when an example becomes available, the decision model must take a decision (e.g. an action). Only after the decision has been taken the environment react providing feedback to the decision model (e.g. the class label of the example).

Suppose a sequence of examples, in the form of pairs $(\boldsymbol{x_i}, y_i)$. For each example, the actual decision model predicts $\hat{y_i}$, that can be or True or False. For a set of examples the error is a random variable from Bernoulli trials. The Binomial distribution gives the general form of the probability for the random variable that represents the number of errors in a sample of n examples. For each point i in the sequence, the error-rate is the probability of observe False, p_i, with standard deviation given by $s_i = sqrt(p_i(1 - p_i)/i)$.

Statiscal theory [9] guarantees that while the class distribution of the examples is stationary, the error rate of the learning algorithm (p_i) will decrease when i increases. A significant increase in the error of the algorithm, suggest a change in the class distribution, and that the actual decision model is not appropriate.

For sufficient large values of the example size, the Binomial distribution is closely approximated by a Normal distribution with the same mean and variance. Considering that the probability distribution is unchanged when the context is static, then the $1 - \alpha/2$ confidence interval for p with $n > 30$ examples is approximately $p_i \pm \alpha * s_i$. The parameter α depends on the confidence level. The drift detection method manages two registers during the training of the learning algorithm, p_{min} and s_{min}. Every time a new example i is processed those values are updated when $p_i + s_i$ is lower than $p_{min} + s_{min}$.

We use a warning level to define the optimal size of the context window. The context window will contain the old examples that are on the new context and a minimal number of examples on the old context. Suppose that in the sequence of examples that traverse a node, there is an example i with correspondent p_i and s_i. In the experiments described below the confidence level for warning has been set to 95%, that is, the warning level is reached if $p_i + s_i \geq p_{min} + 2 * s_{min}$. The confidence level for drift has been set to 99%, that is, the drift level is reached if $p_i + s_i \geq p_{min} + 3 * s_{min}$. Suppose a sequence of examples where the error of the actual model increases reaching the warning level at example k_w, and the drift level at example k_d. This is an indication of a change in the distribution of the examples. A new context is declared starting in example k_w, and a new decision model is induced using only the examples starting in k_w till k_d. It is possible to observe an increase of the error reaching the warning level, followed by a decrease. We assume that such situations corresponds to a *false alarm*, without changing the context. Figure 1 details the dynamic window structure. With this method of learning and forgetting we ensure a way to continuously keep a model better adapted to the present context.

This method could be applied with any learning algorithm. It could be directly implemented inside online and incremental algorithms, and could be implemented as a wrapper to batch learners. The goal of the proposed method is to detect sequences of examples with a stationary distribution. We denote those sequences of examples as *context*. From the pratical point of view, what the method does is to choose the training set more appropriate to the actual class-distribution of the examples.

4 Experimental Evaluation

In this section we describe the evaluation of the proposed method. We used three distinct learning algorithms with the drift detection algorithm: a *Perceptron*, a neural network and a decision tree [10]. These learning algorithms use different representations to generalize examples. The simpler representation is linear, the *Perceptron*. The neural networks example representation is a non linear combination of attributes. The decision tree uses DNF to represent generalization of the examples.

We have used eight artificial datasets, previously used in concept drift detection [6] and a real-world problem [2]. The artificial datasets have several different characteristics that allow us to assess the performance of the method in various

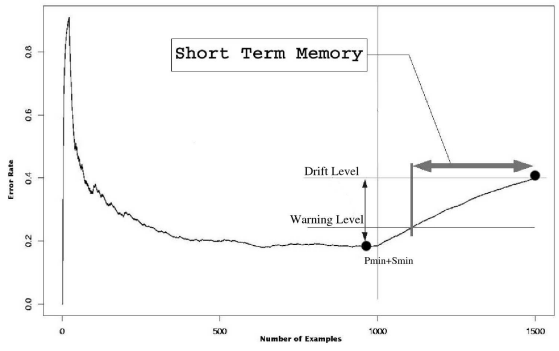

Fig. 1. Dynamically constructed Time Window. The vertical line marks the change of concept.

conditions - abrupt and gradual drift, presence and absence of noise, presence of irrelevant and symbolic attributes, numerical and mixed data descriptions.

4.1 Artificial Datasets

The eight artificial datasets used are briefly described. All the problems have two classes. Each class is represented by 50% of the examples in each context. To ensure a stable learning environment within each context, the positive and negative examples in the training set are interchanged. Each dataset embodies at least two different versions of a target concept. Each context defines the strategy to classify the examples. Each dataset is composed of 1000 random generated examples in each context.

1. SINE1. `Abrupt concept drift, noise-free examples`. The dataset has two relevant attributes. Each attributes has values uniformly distributed in $[0, 1]$. In the first context all points below the curve $y = \sin(x)$ are classified as positive. After the context change the classification is reversed.
2. SINE2. The same two relevant attributes. The classification function is $y < 0.5 + 0.3 \sin(3\pi x)$. After the context change the classification is reversed.
3. SINIRREL1. `Presence of irrelevant attributes`. The same classification function of SINE1 but the examples have two more random attributes with no influence on the classification function.
4. SINIRREL2. The same classification function of SINE2 but the examples have two more random attributes with no influence on the classification function.
5. CIRCLES. `Gradual concept drift, noise-free examples`. The same relevant attributes are used with four new classification function. This dataset has four contexts defined by four circles:

center	$[0.2, 0.5]$	$[0.4, 0.5]$	$[0.6, 0.5]$	$[0.8, 0.5]$
radius	0.15	0.2	0.25	0.3

6. GAUSS. `Abrupt concept drift, noisy examples`. Positive examples with two relevant attributes from the domain $R \times R$ are normally distributed around the center $[0, 0]$ with standard deviation 1. The negative examples are normally distributed around center $[2, 0]$ with standard deviation 4. After each context change, the classification is reversed.
7. STAGGER.`Abrupt concept drift,symbolic noise-free examples`. The examples have three symbolic attributes - size (small, medium, large), color (red, green), shape (circular, non-circular). In the first context only the examples satisfying the description *size=small* $\wedge$ *color=red* are classified positive. In the second context, the concept description is defined by two attributes, *color=green* $\vee$ *shape=circular*. With the third context, the examples are classified positive if *size=medium* $\vee$ *size= large*.
8. MIXED. `Abrupt concept drift, boolean noise-free examples`. Four relevant attributes, two boolean attributes v, w and two numeric attributes from $[0, 1]$. The examples are classified positive if two of three conditions are satisfied:$v, w, y < 0.5 + 0.3 * \sin(3\pi x)$. After each context change the classification is reversed.

4.2 Results on Artificial Domains

The propose of this experiments is to study the effect of the proposed drift detection method on the generalization capacity of each learning algorithm. We also show the method independence of the learning algorithm. The results of different learning algorithms are not comparable.

Figure 2 compare the results of the application of the drift detection method with the results without detection. These are the results for the three learning algorithms used and two artificial datasets. The use of artificial datasets allow us to control the points where the concept drift. The points where the concept drift are signaled by a vertical line. We can observe the performance curve of the learning algorithm without drift detection. During the first concept the learning algorithm error systematically decreases. After the first concept drift the error strongly increases and never drops to the level of the first concept. When the concept drift is detected the error rate grows dramatically compared to the gradual growth of the model without drift detection. But the drift detection method overcomes this and with few examples can achieve a much better performance level, as can be seen with figure 2, than the method without drift detection. While the error rate still grows with the non detection algorithm, the drift detection curve falls to a lower error rate. Both with the neural network and the decision tree it is relevant the application of the detection method over the flat application of the learning algorithm on the learning efficiency.

Table 1 shows the final values for the error rate by dataset and learning algorithm. There is a significant difference of results when the drift detection is used. We can observe that the method is effective with all learning algorithms.

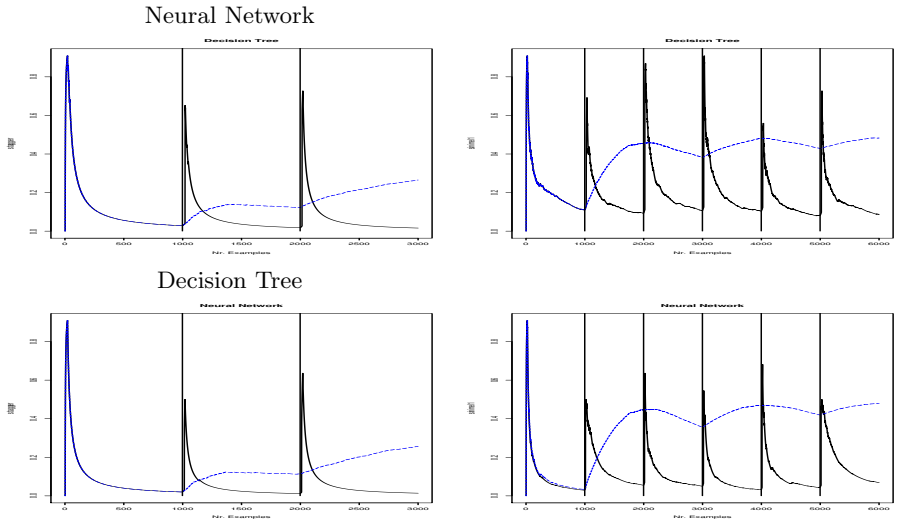

Fig. 2. Abrupt Concept Drift, noise-free examples. Left column: STAGGER dataset, wright column: MIXED dataset.

Nevertheless, the differences are more significant with the neural network and the decision tree.

4.3 The Electricity Market Dataset

The data used in this experiments was first described by M. Harries [2]. The data was collected from the Australian New South Wales Electricity Market. In this market, the prices are not fixed and are affected by demand and supply of the market. The prices in this market are set every five minutes. Harries [2] shows the seasonality of the price construction and the sensitivity to short-term events such as weather fluctuations. Another factor on the price evolution was the time evolution of the electricity market. During the time period described in the data the electricity market was expanded with the inclusion of adjacent areas. This allowed for a more elaborated management of the supply. The excess production of one region could be sold on the adjacent region. A consequence of this expansion was a dampener of the extreme prices. The ELEC2 dataset contains 45312 instances dated from 7 May 1996 to 5 December 1998. Each example of the dataset refers to a period of 30 minutes, i. e. there are 48 instances for each time period of one day. Each example on the dataset has 5 fields, the day of week, the time stamp, the NSW electricity demand, the Vic electricity demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflect deviations of the price on a one day average and removes the impact of longer term price trends.

Dataset	*Perceptron*		Neural Network		Decision Tree	
	No Detection	Detection	No Detection	Detection	No Detection	Detection
STAGGER	0.048	**0.029**	0.351	**0.002**	0.265	**0.016**
SINE1	0.126	**0.115**	0.489	**0.019**	0.490	**0.081**
SINIRREL1	0.159	**0.139**	0.479	**0.068**	0.483	**0.088**
SINE2	0.271	**0.262**	0.492	**0.118**	0.477	**0.100**
SINIRREL2	0.281	0.281	0.477	**0.059**	0.485	**0.084**
MIXED	0.100	0.111	0.240	**0.065**	0.491	**0.465**
GAUSS	0.384	0.386	0.395	**0.150**	0.380	**0.144**
CIRCLES	0.410	0.413	0.233	**0.225**	0.205	**0.109**

Table 1. Final learning algorithm error rate with and without drift detection active.

The interest of this dataset is that it is a real-world dataset. We dont know when drift occurs or if there is drift.

Experiments with ELEC2 data. We have considered two problems. The first problem consists in short term prediction: predict the changes in the prices relative to the last day. The other problem consists in predicting the changes in the prices relative to the last week of examples recorded. In both problems the learning algorithm, the implementation of CART available in R, learns a model from the training data. We have used the proposed method as a wrapper over the learning algorithm. After seeing all the training data, the final model classifies the test data.

As we have pointed out we dont know if and when drift occurs. In a first set of experiments we run a decision tree using two different training sets: all the available data (e.g. except the test data), and the examples relative to the last year. These choices corresponds to *ad-hoc* heuristics. Our method makes an intelligent search of the appropriate training stes. These heuristics have been used to define *upper bounds* to the generalization ability of the learning algorithm.

A second set of experiments was designed to find a *lower bound* for the predictive accuracy. We made an extensive search to look for the segment of the training dataset with the best prediction performance on the test set. There should be noted that this is not feasible in practice, because we are looking for the class in the test set. This result can only be seen as a *lower bound*. Starting with all the training data, the learning algorithm generates a model that classifies the test set. Each new experiment uses a subset of the last dataset which excludes the data of the oldest week, that is, it removes the first 336 examples of the previous experiment. In each experiment, a decision tree is generated from the training set and evaluated on the test set. The smallest test set error is chosen as a *lower bound* for comparite purposes. We made 134 experiments with the 1-day test set problem, and 133 with the 1-week test set problem, using in each a different partition of the train dataset.

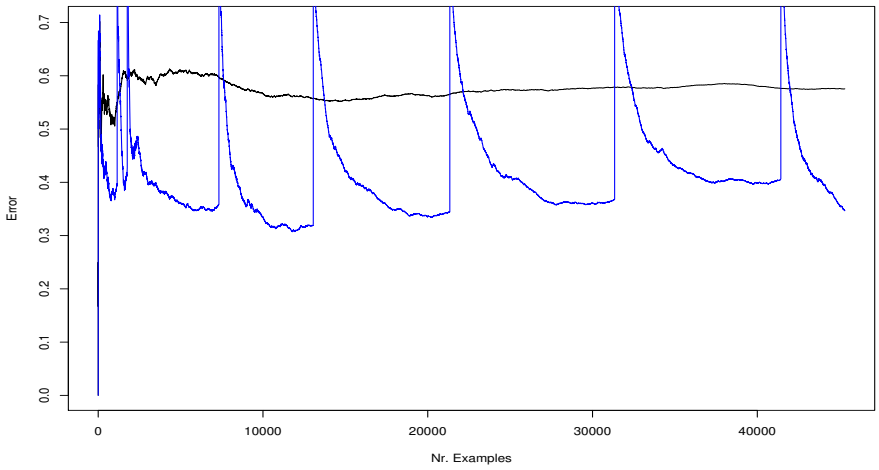

Fig. 3. Trace of the on-line error using the Drift Detection Method applied with a Decision Tree on ELEC2 dataset.

	Lower	*Upper Bound*		Drift
Test Set	*Bound*	*All Data*	*Last Year*	Detection
Last Day	0.104	0.187	0.125	0.104
Last Week	0.190	0.235	0.247	0.199

Table 2. Results of the error rate for two different Test Sets.

The figure 3 presents the trace of the error rate of the drift detection method using the full ELEC2 dataset. The figure also presents the trace of the decision tree without drift detection.

The third set of experiments was the application of the drift detection method with a decision tree to the training dataset defined for each of the test datasets, 1-day and 1-week test dataset. With the 1-day dataset the trees are built using only the last 3836 examples on the training dataset. With the 1-week dataset the trees are built with the 3548 most recent examples. This is the data collected since 1998/09/16. Table 2 shows the error rate obtained with the 1-day and 1-week prediction for the three set of experiments. We can see that the 1-day prediction error rate of the Drift Detection Method is equal to the lower bound and the 1-week prediction is very close to the lower bound. This is a excellent indicator of the drift detection method performance.

We have also tested the method using the dataset *ADULT* [1]. This dataset was created using census data in a specific point of time. The concept should be stable. Using a decision tree as inducer, the method never detects drift. This is

an important aspect, because it presents evidence that the method is robust to false alarms.

5 Conclusions

We present a method for detection of concept drift in the distribution of the examples. The method is simple, with direct application and is computationally efficient. The Drift Detection Method can be applied to problems where the information is available sequentially over time. The method is independent of the learning algorithm. It is more efficient when used with learning algorithms with greater capacity to represent generalizations of the examples. This method improves the learning capability of the algorithm when modeling non-stationary problems. We intend to proceed with this research line with other learning algorithms and real world problems. We already started working to include the drift detection method in an incremental decision tree. Preliminary results are very promising. The algorithm could be applied with any loss-function given appropriate values for α. Preliminary results in regression domain using *mean-squared error* loss function confirm the results presented here.

Acknowledgments: The authors reveal its gratitude to the financial contribution of project ALES (POSI/SRI/39770/2001).

References

1. C. Blake, E. Keogh, and C.J. Merz. UCI repository of Machine Learning databases, 1999.
2. Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.
3. R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 2004.
4. R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts, 1998.
5. Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.
6. M. Kubat and G. Widmer. Adapting to drift in continuous domain. In *Proceedings of the 8th European Conference on Machine Learning*, pages 307–310. Spinger, 1995.
7. C. Lanquillon. *Enhancing Test Classification to Improve Information Filtering*. PhD thesis, University of Madgdeburg, Germany, 2001.
8. M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.
9. Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.
10. W.N. Venables, D.M. Smith, and R development Core Team. An introduction to R, 2002.
11. Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.