SVM-Based Just-in-Time Adaptive Classifiers*

Cesare Alippi[1,2], Li Bu[1], and Dongbin Zhao[1]

[1] State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation Chinese Academy of Sciences, Beijing, 100190 China
bulipolly@gmail.com, dongbin.zhao@ia.ac.cn
[2] Dipartimento di Elettronica e Informazione, Politecnico, di Milano, 20133 Milano, Italy
alippi@elet.polimi.it

Abstract. Aging of sensors, faults in the read-out electronics and environmental changes are some immediate examples of time variant mechanisms violating that stationarity hypothesis mostly assumed in the design of classification systems. Such changes, known in the related literature as concept drift, modify the probability density function of measurements, hence impairing the accuracy of the classifier. To cope with these mechanisms, active classifiers such as the Just-in-time adaptive ones, are needed to detect a change in stationarity and modify the classifier configuration accordingly to track the process evolution. At the same time, when the process is stationary, new available supervised information is integrated in the classifier to improve over time its classification accuracy. This paper introduces a JIT adaptive classifier based on support vector machines able to track changes in the process generating the data with computational complexity and memory requirements well below that of current JIT classifiers integrating k-nearest neighbor solutions.

Keywords: Change Detection Tests, Concept Drifts, adaptive SVM.

1 Introduction

Not rarely, data coming from real applications are characterized by a time-variant nature whereas applications built on the top of them mostly assume the stationarity hypothesis. As a consequence of nonstationarity, the accuracy of the application solution degrades over time. To mitigate this problem we need to intervene by checking at first if the stationary hypothesis holds for a given datastream and, when it does not, provide the application solution with adaptation mechanisms for tracking changes and keep performance at an acceptable level. In the following, we focus the attention on active classifiers [1]–[3], i.e., classifiers incorporating a change-detection test for identifying concept drift and react accordingly. Readers interested in passive classifiers, i.e., classifiers adapting online without the need to detect a change, e.g., those based on incremental learning and ensemble of classifiers, can refer to [4], [5].

* This work was supported partly by National Natural Science Foundation of China (Nos. 61273136, 61034002, 60921061), Beijing Natural Science Foundation (4122083) and visiting professorship of Chinese Academy of Sciences.

T. Huang et al. (Eds.): ICONIP 2012, Part II, LNCS 7664, pp. 664–672, 2012.

The active classification mechanism has become very effective with the design of powerful sequential Change Detection Tests (CDT). For instance, the hypothesis test which uses the computational intelligence-based CUSUM (CI-CUSUM) proposed by Alippi [1] can be also used in a multidimensional analysis without requiring any strong a prior knowledge about the process, provided that data are independent and identically distributed (i.i.d.). Moreover, needed test parameters can be learned from available data. In Just-In-Time (JIT) classifiers the CI-CUSUM test is naturally coupled with a k-nearest neighbor (KNN) classifier which does not require a proper training phase making it an ideal classification system candidate to deal with an evolving environment. In fact, information can be easily inserted in (and at the same time removed from) the knowledge base (KB) without the need to retrain the classifier. A main result is that the classifier tends asymptotically to the optimal Bayes one in stationary conditions. However, the computational complexity of the KNN scales badly with the number of samples in the KB, which increases as more supervised samples are made available and, unfortunately, condensing techniques are very computational demanding.

This paper, starting from [1][2], proposes a novel JIT adaptive classifier based on a SVM as a classification core substituting the native KNN. The use of SVM, also well grounded from the theory point of view, allows us for mitigating the main problems posed by the KNN, namely the computational complexity of the algorithm and the amount of memory requested to store the samples in the KB.

Adaptive SVM classifier solutions can be found in the literature. In [6], an online SVM is proposed, which uses fresh supervised samples only for online training; previous samples are discarded. Xiao et al. [7] provide the -ISVM method, which combines support vectors and misclassification samples to build the new training set; at the same time some less relevant data are selectively removed. Blanzieri et al. [8] proposed an alternative method based on KNN and SVM.

The paper inherits the main advantages of the above methods and insert them in the JIT classifier framework.

The structure of the paper is as follows. Section 2 introduces the computational intelligence-based CUSUM (CI-CUSUM) test for concept drift detection. The general framework design for JIT adaptive classifiers based on SVM, including the update mechanism of the Knowledge Base (KB) is presented in Section 3. Experimental results are provided in Section 4 and conclusions given in Section 5.

2 CI-CUSUM Change Detection Test

The traditional CUSUM test [1], evaluates the difference between two known probability density functions (pdf) at time t

$$R(t) = \sum_{\tau=1}^{t} ln \frac{p\Theta^1(x(\tau))}{p\Theta^0(x(\tau))} \tag{1}$$

where $x(t)$ is an i.i.d random sample and $p\Theta^i$, i=0,1 are the two pdfs parameterized in the parameter vector $\Theta = \{\theta_1, \theta_2, \ldots, \theta_\lambda\}$, $\lambda = |\Theta|$ referring to the working condition before the concept drift and after the change, respectively.

Then, the method estimates the minimum $m(t)=min_{1\le\tau\le t}(R(\tau))$ on the train set. When the discrepancy $g(\bar{t})=R(\bar{t})-m(\bar{t})$ at time $\bar{t}$ is larger than a given threshold h a change is detected. This method is rather effective in detecting changes but requires the availability of the pdfs, parameters Θ^0, Θ^1 and h, information which is rarely available.

The above problems can be solved by considering the CI-CUSUM [1], which extends the traditional CUSUM test, in the following aspects

1) When sequence $X=\{x(t)\},t=1,...,N$, is composed of scalar, real, i.i.d. random samples and n is sufficiently large, thanks to the central limit theorem the transformation $y(t)=\frac{1}{n}\sum_{\tau=(t-1)n+1}^{tn}(\tau),t=1\ldots N/n$ is ruled by a Gaussian pdf with mean and variance parameters $\Theta=\{\mu,\sigma^2\}$. The parameters can be directly estimated on the sequence $Y=\{y(i)\},i=1\ldots N/n$;
2) The threshold value h can be learned from the training set instead of been asked to the designer;
3) Designers can consider their favorite features for constructing a new and application tailored CDT; a principal component analysis (PCA) can be considered to reduce the complexity of the feature space;
4) Configurations for the alternative hypothesis Θ^1 can be automatically generated by balancing change detection performance and computational complexity.

In the following we consider the CI-CUSUM test as a CDT for the JIT classifier.

3 JIT Adaptive SVM Classifier

We briefly introduce the SVM and show how the technique can be modified to be integrated within the JIT framework.

Let $\{x_i,y_i\}$, $x_i(t)\in R^N$, $y_i\in\{-1,1\},i=1,2\ldots,N$ be the training set and

$$f(x)=sgn(\omega\cdot x+b) \tag{2}$$

the SVM classifier with parameters ω obtained by solving the optimization problem

$$min(\frac{1}{2}\omega^2+C\sum_{i=1}^{N}\xi_i) \tag{3}$$

subject to $y_i((\omega\cdot x_i)+b)\ge 1-\xi_i$, $\xi_i\ge 0$, $i=1,...,N.$ ω is the normal vector to the hyperplane, b is the hyperplane offset, $C>0$ is a cost parameter and ξ_i the slack variable. The (3) can be cast in the dual form [9]

$$max\left(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K\left(x_i, x_j\right)\right) \tag{4}$$

subject to

$$\sum_{i=1}^{N} y_i\alpha_i = 0, C \geq \alpha_i \geq 0$$

where $K\left(x_i, x_j\right) = exp\left(-\frac{\left\|x_i, -x_j\right\|^2}{\sigma^2}\right)$ is a kernel function, here chosen to be the radial basis function (RBF) parameterized in σ. Finally, the classifier has form

$$f\left(x\right) = sgn(\omega \cdot x + b) = sgn\left(\sum_{i=1}^{N}\alpha_i y_i K\left(x_i, x\right) + b\right) \tag{5}$$

Parameters α_i and b are determined by solving the dual problem: the support vectors (SV) are those samples for which $\alpha_i \neq 0$.

Here, we consider the Grid-Search method [10] to select the best parameters *C* and σ; performances were evaluated with a K-fold cross-validation to mitigate the fact that a limited data set is available.

It should be noted that each newly made available supervised sample $\{x_i, y_i\}$ satisfies the Karush-Kuhn-Tucker (KKT) condition [11]

$$\begin{aligned} &\alpha_i = 0 \Rightarrow y_i f(x_i) \geq 1; \\ &0 < \alpha_i < C \Rightarrow y_i f\left(x_i\right) = 1; \\ &\alpha_i = C \Rightarrow y_i f(x_i) \leq 1 \end{aligned} \tag{6}$$

A sample satisfying the KKT conditions does not change the SVs and, as such, the structure of the classifier. Within an incremental learning strategy only those samples violating the KKT condition should be kept as relevant and stored in KB; the others can be discarded since do not provide any additional information to the current space partitioning.

SVs fully describe the classification problem given the available data KB. Since in most cases the number of SVs is a small fraction of the number of samples in KB, we recommend to use the SVs instead of the original training samples in the KB for classification purposes. The final effect is that by using SVs, we save memory w.r.t to a solution envisaging a KNN classifier.

When we are working in a stationary environment a new supervised sample violating the KKT must be added to the KB. However, inserting a sample in the SV set must be seen as a perturbation to the method. As such, when the number of inserted samples in the incremental knowledge base (IKB) exceeds a threshold, we need to fuse KB with IKB and re-train the SVM.

Differently, when a concept drift is detected, data contained in KB become obsolete, hence negatively impacting on performance. As such they must be removed and only the most recent ones kept to compose a short memory classifier.

The joint use of the CI-CUSUM CDT and the dynamic management of the knowledge base of the classifier leads to the JIT adaptive SVM classifier of Algorithm 1. The Algorithm operates both in stationary and nonstationary conditions. In stationarity conditions (also following an abrupt concept drift) the performance improves asymptotically; in non-stationary conditions the classifier adapts to track the change and keep performance high.

JIT adaptive SVM classifier

1. Configure the test CDT and train classifier SVM on the training set KB_0;
2. $KB = support\ vectors\ of\ KB_0$; $n = |KB|$; $IKB = 0$;
3. While (1) {
   Acquire sample x;
4. if(new supervised knowledge is available) {
5. Insert samples (x, y) violating the KKT condition in IKB }
6. if(CDT (x) = nonstationary) {
7. $KB = last\ N\ samples\ of\ new\ supervised\ samples$
8. Train SVM and CDT on KB ;
9. $KB = support\ vectors\ of\ KB$; $n = |KB|$;}
10. else{
11. $n_{IKB} = |IKB|$;
12. if($n_{IKB} / n > Th$){
13. $KB = KB \bigcup IKB$;
14. Train SVM and CDT on KB;
15. $KB = support\ vectors\ of\ KB$
16. $n = |KB|$; $IKB = 0$;
    }
    }
17. Classification=SVM (x , KB , C , σ);
}

Algorithm 1. The Just-in-time adaptive classifier with a SVM core

The performance of the JIT adaptive SVM classifier largely depends on the ability of the CDT to detect a concept drift. As such, when the process undergoes a concept drift but the CDT does not recognize it (false positive), the classifier will continue to follow the stationary mode, with and obvious loss in classification accuracy. However, before or later the CDT will detect the change if the "magnitude" of the concept drift evolves over time (for instance, a gradual concept drift will be seen by the CDT as a sum of concept drift).

4 Experimental Validation

In order to verify the performance of the suggested adaptive classifier we considered two applications. The first provides synthetically generated data taken from [2], the second are real acquisitions from photodiodes [3].

Application D1 contains four classification datasets characterized by different concept drift: abrupt, transient, stairs and drift. Each dataset is composed of 50 sequences of 10000 real-valued observations drawn from two Gaussian-distributed classes initially distributed (stationary condition) as $P(x|\omega_0) = N(0,3)$, $P(x|\omega_1) = N(4,3)$. In the abrupt case, the concept drift occurs in the middle of each sequence by providing an additive perturbation ($\delta = 3$) inducing an increment of the mean of both classes. In the transient dataset, the mean of both classes increases after one third of the dataset and returns to the original value after two thirds of the dataset samples. The sum of concept drift dataset increases its mean at one fourth, two fourths, and three fourths of the sequence. The drift dataset is configured so that the drift starts at sample 5000 and reaches a perturbation $\delta = 3$, on the expectations at the end of it (the distributions of last samples are $P(x|\omega_0) = N(3,3)$, $P(x|\omega_1) = N(7,3)$).

Application D2 refers to a dataset composed of 28 sequences of measurements taken from couples of photodiodes. Each sequence is composed of 12000 16-bit measurements (6000 per sensor). We test the algorithms by classifying the observations according to the sensor. From figure 1 we see that the photodiodes are subject to a sequence of abrupt concept drift.

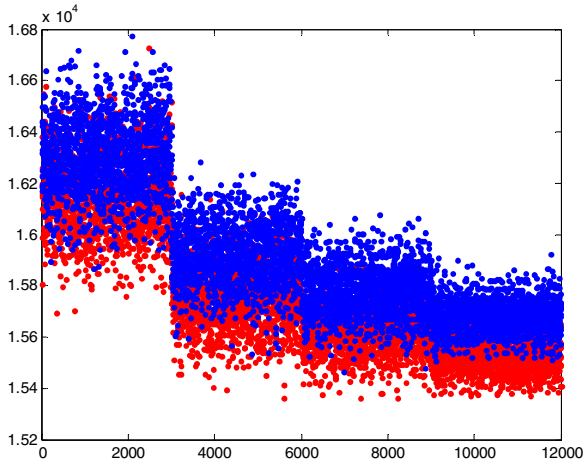

Fig. 1. Some data from application D2

Table 1. JIT adaptive SVM classifier and the KNN classifier: application D1

Samples \ Classifiers		JIT adaptive KNN classifier		JIT adaptive SVM classifier	
		ACC(%)	CT(s)/M(S)	ACC(%)	CT(s)/M(S)
Abrupt	$\delta = 3$	73.90	18.67/1351	74.35	15.44/846
Transient	$\delta = 3$	72.33	12.70/1323	73.58	10.06/931
Stairs	$\delta = 3$	72.39	12.63/969	73.35	11.60/671
	$\delta = 1$	73.62	14.27/1284	73.95	10.46/734
Drift	N(3,3)	73.61	14.42/1496	73.93	10.57/912

Table 2. JIT adaptive SVM classifier and the KNN classifier: application D2

Samples \ Classifiers	JIT adaptive KNN classifier		JIT adaptive SVM classifier	
	ACC(%)	CT(s) /M(S)	ACC(%)	CT(s) /M(S)
Application D2 (average)	71.87	20.29/1419	71.56	19.87/1220

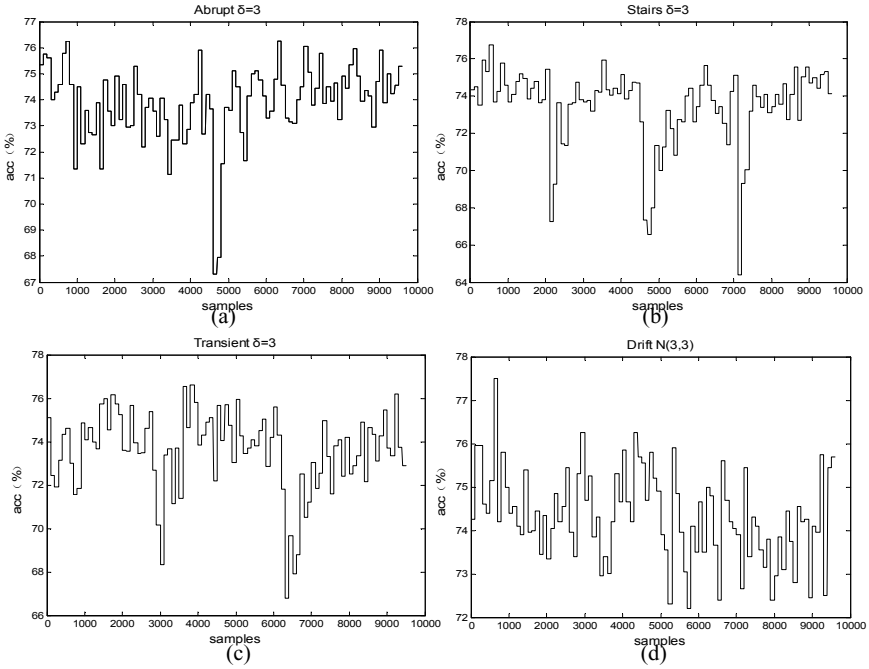

Fig. 2. Classification accuracy: the JIT adaptive SVM classifier, application D1

Tables 1 and 2 show the performance of the JIT adaptive SVM classifier and the JIT KNN classifier of [2] on applications D1 and D2, respectively. Results represents the average over 50 sets of experiments. We immediately observe how the accuracies of the JIT-SVM are perfectly aligned with those of the JIT-KNN, which, de facto, represents the optimal classifier (in [2] the KNN has been optimized). We appreciate the fact that, on average, the JIT-SVM is 16% faster than the KNN-SVM and its memory consumption (in samples S) is 32% less than its counterpart. Fig.2 shows the classification accuracy over incoming samples for application D1 (accuracy is averaged every 100 samples). We comment that, when there are no changes (i.e., we are in stationarity conditions), new samples added to the KB improve the accuracy of the JIT adaptive SVM classifier. Of course, when a concept drift occurs accuracy drops but the classifier reacts as new supervised samples come in.

5 Conclusions

The paper presents an adaptive SVM mechanism within a Just-in-Time classification framework. It comes out that the computational complexity of the traditional JIT, which relies on the use of a KNN core for classification, can be reduced by considering an incremental SVM classifier instead of about 15%. The same results with the saving memory which shows an improvement of about 32% over the KNN.

Surely, space for improvement is here. In particular, a more effective computational-aware mechanism can be considered to train the SVM starting from an already configured situation with available support vectors (changes in the Bayes classification border should be contained in the stationary and the gradual concept drift case). Another improvement issue resides in assessing the proportion of old and new samples to be kept to deal with the concept drift.

References

1. Alippi, C., Roveri, M.: Just-In-Time Adaptive Classifiers—Part I: Detecting Nonstationarity Changes. IEEE Transactions on Neural Networks 19, 1145–1153 (2008)
2. Alippi, C., Roveri, M.: Just-In-Time Adaptive Classifiers—Part II: Designing the Classifier. IEEE Transactions on Neural Networks 19, 2053–2064 (2008)
3. Alippi, C., Boracchi, G., Roveri, M.: A Just-In-Time Adaptive Classification System Based on the Intersection of Confidence Intervals Rule. Neural Networks 24, 791–800 (2011)
4. Zliobaite, I.: Learning under Concept Drift: An Overview. Technical Report, Faculty of Mathematics and Informatics (2009)
5. Elwell, R., Polikar, R.: Incremental Learning of Concept Drift in Nonstationary Environments. IEEE Transactions on Neural Networks 22, 1517–1531 (2011)
6. Oskoei, M.A., Gan, J.Q., Hu, H.S.: Adaptive Schemes Applied to Online SVM for BCI Data Classification. In: 31st Annual International Conference of the IEEE EMBS, Minneapolis, Minnesota, pp. 2600–2603 (2009)
7. Xiao, R., Wang, J.C., Sun, Z.X., Zhang, F.Y.: An Incremental SVM Learning Algorithm α-ISVM. Journal of Software 12, 1818–1824 (2001)

8. Blanzieri, E., Melgani, F.: An Adaptive SVM Nearest Neighbor Classifier for Remotely Sensed Imagery. In: Geoscience and Remote Sensing Symposium, Trento, pp. 3931–3934 (2006)
9. Bian, Z.Q., Zhang, X.G.: Pattern Recognition, pp. 285–300. Tsinghua University Press, Beijing (1999)
10. Shen, H.J., Xi, H., Xie, G.: The Improved Grid-Search Algorithm Used in the Fault Diagnosis by SVM. Mechanical Engingeering & Automation, 108–110 (2012)
11. Zeng, W.H., Ma, J.: A New Algorithm to Incremental Learning with Support Vector Machine. Journal of Xiamen University (Natural Science) 41, 687–691 (2002)