To Better Handle Concept Change and Noise: A Cellular Automata Approach to Data Stream Classification

Sattar Hashemi[1], Ying Yang[1], Majid Pourkashani[2], Mohammadreza Kangavari[2]

[1] Clayton School of Information Technology
Monash University, Australia
{Sattar.Hashemi,Ying.Yang}@infotech.monash.edu.au

[2] Computer Engineering Department, Iran University
Of Science and Technology, Tehran, Iran
{mpkashani, kangavari}@iust.ac.ir

Abstract: A key challenge in data stream classification is to detect changes of the concept underlying the data, and accurately and efficiently adapt classifiers to each concept change. Most existing methods for handling concept changes take a windowing approach, where only recent instances are used to update classifiers while old instances are discarded indiscriminately. However this approach can often be undesirably aggressive because many old instances may not be affected by the concept change and hence can contribute to training the classifier, for instance, reducing the classification variance error caused by insufficient training data. Accordingly this paper proposes a cellular automata (CA) approach that feeds classifiers with most *relevant* instead of most *recent* instances. The strength of CA is that it breaks a complicated process down into smaller adaptation tasks, for each a single automaton is responsible. Using neighborhood rules embedded in each automaton and emerging time of instances, this approach assigns a relevance weight to each instance. Instances with high enough weights are selected to update classifiers. Theoretical analyses and experimental results suggest that a good choice of local rules for CA can help considerably speed up updating classifiers corresponding to concept changes, increase classifiers' robustness to noise, and thus offer faster and better classifications for data streams.

Keywords: Data Stream Classification, Cellular Automata, Concept Change, Noise Suppression

1. Introduction

Nowadays, there are many applications in which data are not static but streaming, such as sensor network data and credit card transactions. In data streams, the concept underlying the data may change over time, which can cause the accuracy of current classifiers to decrease. Meanwhile, real-world data are seldom perfect and often suffer from significant amount of noise, which may affect the accuracy of induced classifiers. Dealing with concept changes and differentiating them from noise has become an interesting and challenging task in the machine learning and data mining community [8,3,24].

Instances in data streams often arrive in huge volume and at high speed. It is virtually impossible to apply general data mining algorithms to streaming data as a whole. Hence, methods proposed for mining data streams are mainly based on windowing approaches [8,9,10]. In windowing approaches, a frame of recent instances are selected as a window and are constantly used to train and evaluate the classifiers. Some of these windows are fixed-size [21,22], while others are resized adaptively according to the validity of the learner [8], for example, whenever a concept change happens the window size is reduced so that the learner is supplied with a smaller portion of instances. In either case, the implicit idea underlying the method is that *the more recent the data, the more relevant they are to test/train the current learner.*

However, due to concept change and noise in data streams, classifiers learned from a small portion of recent instances may vary significantly in performance [24]. Furthermore, it has been shown that the arrival time of an instance is not always the best relevance criterion [11,20,23]. The schemes proposed to deal with the mentioned problems and find the most relevant instances are mainly based on the ensemble approach [9,11,22]. The key idea is to divide data stream into sequential windows and assign a weight to each window so that the estimated generalization error is minimized. Based on authors claim, the approach shares the assumption that newer instances are always more important than older ones which is not the best selection strategy in noisy environments. Moreover the approach meets instances more than once to assign the windows' corresponding weights what obviously affects efficiency of the model for mining high speed data streams [9,11].

Loureiro et al. have suggested that the nearness of instances according to the attributes used for learning can be a good measure of relevance [18]. Other experiments have revealed that clustering instances can aid the learning process in the sense that a group of "near" (according to some distance measure) instances with the same class label in neighborhood are more likely to be relevant [19].

This paper proposes a cellular automata (CA) approach that takes advantages of the above-mentioned approaches to handle both concept change and noise problems. Using simple neighborhood rules, it retains instances corresponding to the current concept for learning even if they are old. This property provides the leaner with more relevant training instance, which can reduce its classification variance. Meanwhile, noisy instances can be discarded even when they are most recent data.

The strength of CA is that it breaks a complicated process down into smaller adaptation tasks, for each a single automaton is responsible. Each individual automaton is only involved with the simple rules which are applied locally to update the overall automata's states. Takac has studied utilization of a Cellular Genetic Programming method in static data mining applications [5, 6]. He shows that using GP in a cellular framework can improve scalability and efficiency of the previous algorithm utilized Genetic Programming to evolve decision trees [16, 17]. Similar research can be found in Folioni et al.'s research [4]. Maji et al. have studied the application of cellular automata theory in several pattern recognition fields [7]. They propose a cellular automata classifier, which can improve classification accuracy, minimize memory overhead, and has linear complexity in the training phase.

As far as we know, the theory of cellular automata has not been studied in the context of data stream mining. To do so, our model keeps and updates a lattice of automata as data stream in. Upon arrival, each instance is "placed" on its corresponding automaton on the lattice and affects its neighbor instances according to some simple local rules. The instance corresponding to each automaton is assigned with a relevance *weight*, which represents its chance of being "selected" for training the classifier. By choosing appropriate local rules to be applied in each automaton, we can improve classifiers' classification accuracy, adaptation speed to concept change and robustness to noise.

The rest of the paper is organized as follows. Section 2 introduces the theory of cellular automata, emphasizing on the concepts we will use in our method. Section 3 offers a formal representation of the CA approach to suppress noise and handle concept change in data streams. Section 4 presents experimental results and analyses. Section 5 gives concluding remarks and suggestions for future work.

2. Background Knowledge

The theory of cellular automata (CA) was originally presented by Von Neumann in the 1960's in hope of simulating complex biophysical behaviors [3]. It was then further explored by numerous researchers [1,2,3]. A cellular automata is a discrete dynamical system and includes a regular spatial lattice (gird) of points. Each point, called a cell or an automaton, is a finite state machine. The input to a cell are the states of all cells in its neighborhood. The states of the cells in the lattice are updated according to a local rule. That is, the state of a cell at a given time depends only on its own state and its nearby neighbors' states one time step before. All cells on a lattice are updated synchronously. Thus the state of the entire lattice advances in discrete time steps.

A 1-dimensional cellular automata takes as its underlying space the infinite strip S^Z where *S* is a finite state machine and *Z* are integers (infinite in both positive and negative directions). Overall the state of the automata is updated according to a global function f: $S^Z \rightarrow S^Z$, whose dynamics are determined "locally" as defined below. Let *r* be the radius of neighborhood. A local function *f* is defined on a finite region

$$f : S^{2r+1} \rightarrow S \qquad (1)$$

which uses the fact that the next state of a single cell in a 1-dimentional CA depends on the states of the *r* cells to its right, the *r* cells to its left and itself. By applying function *f* to each cell and using the neighbors as input, the next state of each cell is calculated. For higher dimension cellular automata, there exist a variety of neighborhood rules among which Moore's and Von Neumann's neighborhood [3] are the most widely used. It is expected that CA generates an overall solution to the complicated process.

3. Proposed Method

Concept change is an intrinsic characteristic of data streams that makes re-learning necessary. Furthermore, noisy instances often appear in data streams, which may falsely be considered as a concept change, trigger unnecessary (time-consuming) re-learning, and increase classification error. An ideal algorithm is one that handles concept changes in an efficient manner and differentiates real changes from noise. In the following subsections we will define our method based on CA for mining data streams, and study how it suppresses noise and adapts to concept changes.

3.1 Cellular-Automata-Based Learning

Suppose a data stream consists of instances with *d* attributes, each taking discrete values $\alpha_i \in V_i,\ i = 0,1,...,d$ where $V_i = \{v_1^i, v_2^i, ..., v_{n_i}^i\}$ and n_i is the number of possible values of the *i*th attribute. Without loosing generality, we assume that $n_1 = n_2 = ... = n_d = n$. In this manner, each instance s_i can be identified by the *d*-tuple of attribute values $(\alpha_1, \alpha_2, ..., \alpha_d)$. This *d*-tuple represents a point in a *d*-dimensional problem space. Let $G = \{c_k = (D,t,w,c), D = (\alpha_1, \alpha_2, ..., \alpha_d) \mid \alpha_i \in V_i,\ for\ i = 1..d\}$ be a d-dimensional grid (lattice) of cellular automata, which has a one-to-one relation with the instances in the problem space, and c_k represent the kth cell or automaton in a sequential indexing. Each automaton is assigned three parameters that describe the state of its corresponding data point: *t* as *timetag*, *w* as *weight*, and *c* as *class*. *Timetag* is a monotonically decreasing value that shows the recency of the data. *Weight* is a parameter that accounts for the relevance of the automaton to the current concept. *Class* is the true label of the instance that is used afterwards to validate the learnt concept. We use the super-index notation to refer to parameters of a cell. Thus, c_k^t means the timetag of the cell c_k, c_k^D stands for its corresponding instance, and c_k^w for its weight. Whenever a new instance streams in, the corresponding cell is activated and its timetag and weight are initialized to a pre-defined constant *T* and *W* respectively. These parameters for other active cells are decreased by a pre-defined forgetting factor λ ;

$$(c_k^t \leftarrow c_k^t - \lambda\ and\ \ c_k^w \leftarrow c_k^w - \lambda)\ \ \forall k\ \ s.t.\ \ c_k\ \ is\ \ active\ \ cell \qquad (2)$$

As soon as the timetag and weight of an active cell reach zero, the cell is deactivated. It is worth mentioning that the weight of each cell can also be altered by the local rules of the cells in its neighborhood. Local rules are simple heuristics that aim at feeding classifiers with more relevant instances by taking into account the local situations of the instances in the current concept with respect to each other. By adding local rules, our CA based approach is more appropriate than the naïve recency-based windowing approach. Different rules can be defined depending on the purpose, for example, do we need to deal with noise, detect concept changes or both, which is to be discussed in Sections 3.2 and 3.3. The only restriction on these rules is the one that has been set by the CA theory: locality. In other words, the next state of an automaton should only depend on its current state, and the states of its neighbors in the grid. We have adopted a generalized Moore neighborhood definition [3], which includes cells in a hyper-sphere that is centered at the base cell and with a radius of *n* cells. Formally, the neighborhood is a function *N*, which maps each cell to a group of cells:

$$N:\{c_k\}\rightarrow P(\{c_k\}),\quad where\ P(A)\ denotes\ power\ set\ of\ A.$$
$$N_n(c_k)=\left\{c_p \middle| c_k^D=(\alpha_1,\alpha_2,\cdots,\alpha_d)\ and\ c_p^D=(\alpha_1+\delta_1,\alpha_2+\delta_2,\cdots,\alpha_d+\delta_d),\quad |\delta_i|\le n \quad 1\le i\le d\right\} \tag{3}$$

where δ_j , $j=1...d$ is a signed integer representing the distance of a neighbor to the center cell of the *j*th attribute.

The algorithm runs as follows. The main loop waits until a new instance arrives. The new instance is classified using base learner and the classification accuracy of the learner is returned. The new instance is then placed on the grid and CA states are updated according to the local rules to be defined next. Any cell whose weight is above a threshold (usually *W/2*) is considered relevant and its corresponding instance is selected for learning whenever learner accuracy drops below predefined threshold.

3.2 Noise Suppression

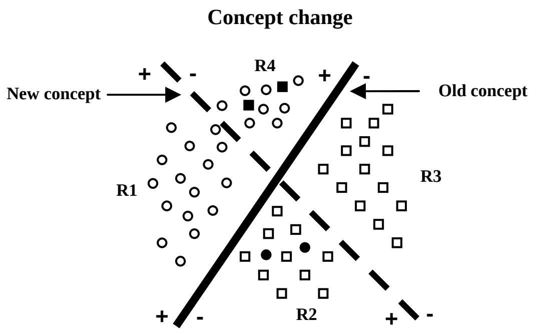
To deal with noise, our approach simply checks for local class disagreements. For each instance, its true class is checked against the class of its neighbors. If most of the active neighbors have a different class, this instance is considered as noisy. Its weight is suppressed to an amount below the selection threshold. As a consequence, this instance is moved out of the selection basket of learning. This strategy is based on the heuristic "A positive sample will rarely emerge in-between many negative samples, and vice versa".

3.3 Handling Concept Changes

Both concept change and noise affect classifiers' performance but in different ways. Effect of the concept change in a particular neighborhood is consistent while it is not the case for noise. Our proposed method adopts this local heuristic to detect concept change: "A newly misclassified instance, whose class label is different from most neighboring instances but is supported by coming instances from the stream, indicates a concept change".

To distinguish between concept change and noise, a two-tailed binomial sign test is applied on misclassification sequence of the classifier. If the result is less than the critical level of 0.05, the drop in accuracy is the effect of concept change, otherwise it is due to noise. Whenever a concept change is detected, the misclassified new instances are considered as representatives of the new concept, while their nearby instances with a different class are considered as representatives of the outdated concept. In particular, whenever an instance of the new concept is detected by a cell, the cell looks for the neighbor cells that have a different class from that instance. These cells are thereafter suppressed, that is, their weight state is decreased by a large amount so as to reduce the probability that they are selected into the basket for learning. By suppressing instances of the older concept, the selection basket is left with instances that are more relevant to the new concept (both new and old instances).

To illustrate CA's advantage, let's consider a generic concept change scenario as depicted in Figure 1. The feature space is a 2-d plane with two linearly separable classes, positive and negative, whose instances are shown by circles and squares respectively. The solid line represents the “old” concept before the concept changes, where regions R1 and R4 are positive, and regions R2 and R3 are negative. Instances of the old concept are shown by “empty” shapes. The dashed line represents the “new” concept after the concept changes, where regions R1 and R2 are positive, and R3 and R4 are negative. Instances of the new concept are shown by “filled” shapes, in contrast to the older “empty” ones.



Regions	R1	R2	R3	R4
Labels in old concept	+	-	-	+
Labels in new concept	+	+	-	-

Figure 1. Regions R1 and R3 are not affected by the concept change. Their instances are still valid for the new concept and hence should be utilized to update classifiers.

Now assume that the old concept is in effect (the solid line) and the algorithm detects concept change when new instances (filled shapes) fall into regions R2 or R4 with unexpected class labels. Please note that it is impossible to detect the concept change if all new instances fall into the “unaffected regions” (R1 and R3). Once the concept change is detected, the CA approach will identify those negative instances (empty squares) in R2 and those positive instances (empty circles) in R4 are no longer relevant to the learning task. Those instances will be removed while instances in regions R1 and R3 will be retained for updating classifiers. In contrast, the windowing approach will indiscriminately remove instances from every region as long as they are not most recent regardless of whether they are still valid for the new concept. As a result, the windowing approach will unwisely reduce the amount of training data, undermine the learner’s ability to approximate the new concept and hence increase the classification error [14, 15].

4. Experiments

Experiments are conducted to evaluate the cellular automata approach for dealing with noise and handling concept change using both synthetic and real-world data streams. Evaluations involve decision trees (DT) [27], support vector machines (SVM) [12] and adaptive neuro-fuzzy system (ANFIS) [26].

4.1. Datasets

Two public benchmark datasets are used for evaluation. They are relatively large, have concept changes, and are commonly used in published research of data stream classification.

Hyperplane dataset [20-23] is created artificially to give a more elaborate view of algorithms' performance. A hyperplane in a *d*-dimensional space is denoted by $\sum_{i=0}^{d} w_i x_i = w_0$, where each vector of variables $< x_1, \cdots, x_d >$ is a randomly generated instance and is uniformly distributed. Instances satisfying $\sum_{i=0}^{d} w_i x_i \geq w_0$ are labeled as positive, and otherwise negative. The value of each coefficient w_i is continuously changed so that the hyperplane is gradually drifting in the space. Besides, the value of w_0 is always set as $\frac{1}{2}\sum_{i=1}^{d} w_i$ so that roughly half of the instances are positive and the other half are negative. Noise is introduced in the data by toggling the class label of a certain number of instances. We generate random instances uniformly distributed in $[0,1]^d$ and then discretize each dimension into 10 intervals of equal size. The data stream contains 50,000 instances.

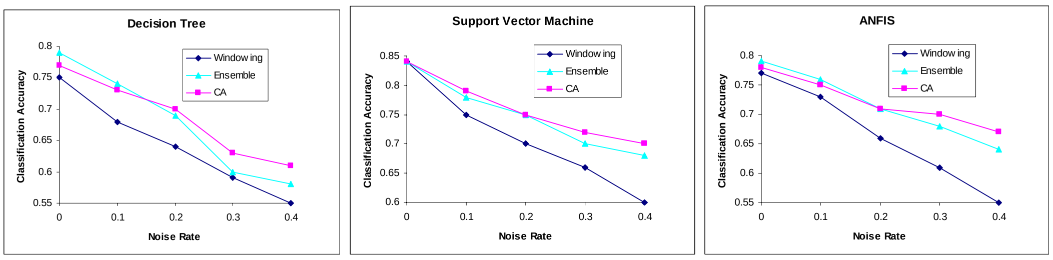
Car dataset [24] is from the UCI Machine Learning Repository [25]. This dataset consists of 6 nominal ordered attributes and 1728 instances in 4 classes. To simulate concept change, an attribute is randomly selected, and each of its values is treated as a state in the Markov Chain [20,23]. For each state 100 instances possessing the same attribute value are randomly selected from the data set, and are queued into the data stream. Altogether the data stream comprises 50,000 instances. To add class noise, we adopt a pairwise noise previously used in [24]

4.2 Experiments with Different Levels of Noise

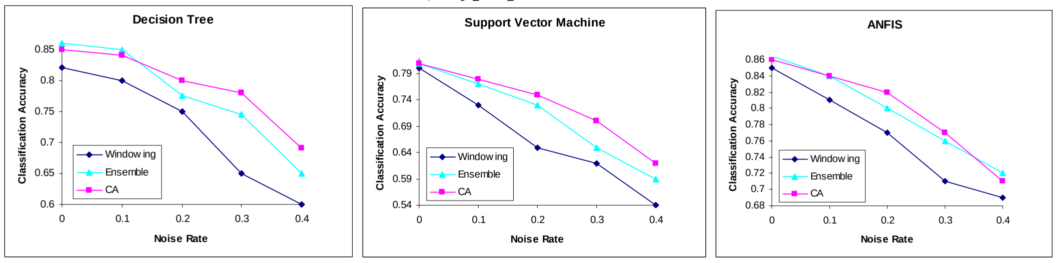
To observe the role that local rules play in dealing with noise, we add different levels of noise into the data. We have used noise rates of 0% (noiseless), 10%, 20%, 30% and 40% in this experiment, where a *x*% level means *x*% instances have their class labels toggled. It is worth noting that although the main focus of this subsection is how different models deal with noise, there are also concept changes in both datasets. Hence, the presented results demonstrate alternative methods' behavior in face of both concept changes and different levels of noise.

We deploy the adaptive windowing, the ensemble and the CA approaches into the noisy datasets. The CA removes instances that it has identified as noise and feed other instances to classifiers. The predictive performance of each classification algorithm (DT, SVM and ANFIS) under the windowing, the ensemble and the CA strategy respectively are compared in Figure 2.

As the graphs show, the proposed CA approach yields better classification accuracy, which suggests that the local rule heuristic is more robust to noise. Although the ensemble approach performs better at first, but, it is outperformed by the CA approach in noisy environment what approves our early discussion that relying on the most recent instances is not always the best strategy in noisy environments.



a) Hyperplane dataset



b) Car dataset

Figure 2. Classification accuracy of different classification algorithms under the windowing, the ensemble and the CA approaches as a function of noise level.

4.3 Experiments with Different Amount of Concept Changes

In this section we compare the performance of the different approaches in data streams with different complexities of concept changes. As the complexity of concept changes is fixed in real-word data such as the Car dataset, we conduct the experiments only on the Hyperplane dataset [21]. In this dataset, the complexity of concept changes relates to the number of dimensions *k* whose weights are changing over time. Using four different values of *k*, we produce several Hyperplane datasets to evaluate the performance of our approach from different points of view. Three measures of the performance are calculated for each experiment: 1) convergence time, *ct*, which is the average time the system remains unstable during updating to a concept change, measured by the number of instances; 2) classification accuracy, *acc*, which is the percentage of the classifier's correct predictions, and 3) re-learning requirement, N_ℓ, as the total number of times the classifier is re-learned from scratch.

Results are shown in Table 1. It is observed that almost in every case the CA approach outperforms the windowing and ensemble approaches: it achieves higher classification accuracy; spends less time on updating classifiers upon concept changes; and require less frequently building classifiers from scratch. Lower *ct* and N_ℓ are desirable because data streams demand fast online classification. In other words, whenever the other approaches present almost equal classification accuracy, it is at the cost of using more time resources.

Table 1. Experiment results in learning drifting concepts

Learner	k	Windowing Approach			Ensemble Approach			CA Approach		
		acc	*ct*	N_ℓ	*acc*	*ct*	N_ℓ	*acc*	*ct*	N_ℓ
Decision Tree	1	%75	43	70	%79	29	85	%77	31	58
	2	%71.2	47	79	%73	34	92	%73.3	37	67
	3	%60.8	72	95	%65	61	111	%64.8	57	74
	4	%56	75	104	%60	67	123	%61.3	60	86
SVM	1	%84	47	66	%84	34	80	%84	35	54
	2	%73	70	80	%78	47	89	%77.7	45	62
	3	%69.7	77	85	%74	50	100	%76	48	66
	4	%58	100	96	%67	67	119	%68.3	63	73
ANFIS	1	%77	33	69	%79	15	72	%78.1	20	56
	2	%71.3	40	77	%74.5	23	80	%75	23	73
	3	%70.9	41	86	%72.9	26	98	%73.4	25	78
	4	%63.8	57	102	%65	44	113	%66	40	83

5. Conclusion and Future Work

A cellular automata (CA) approach is proposed for data stream classifications. Its advantage is to use most relevant instances instead of most recent instances to update classifiers in face of concept changes. By using neighboring instances and simple local rules, the CA approach can be more robust to noise, achieve higher classification accuracy, adapt faster to changed concepts, and less frequently require building classifiers from scratch.

Using cellular automata for data stream classification is a new topic and there are various interesting issues to further investigate. For example, as cellular automata are distributed along a grid with just interactions between neighboring cells, they can be consider as a means for parallel stream mining. For another example, cellular automata have a discrete nature. Hence, an effective online method for discretizing continuous attributes in data streams may be considered for future work.

References

1. A. Adamatzky. Identification of Cellular Automata. Taylor and Francis, London, Bristol, 1994.
2. J. D. Farmer, T. Toffoli, and S. Wolfram, editors. Cellular Automata : Proceedings of an Interdisciplinary Workshop. Los Alamos, New Mexico, March 7-11, 1983,
3. S. Wolfram, editor. Theory and Applications of Cellular Automata. World Scientific, Singapore, 1986. Collection of papers on CA's.
4. G. Folioni, C. Pizzuti and G.Spezzano. A Cellular Genetic Programming Approach to Classification. In the Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1015-1020, 1999
5. A. Takac, Application of Cellular Genetic Programming in Data Mining, Proceedings of Conference Knowledge 2004, Brno, Czech Republic.
6. A. Takac, Genetic Programming in Data Mining: Cellular Approach, Ms-Thesis for the faculty of mathematics, physics and informatics, Institute of informatics, Comenius University, Bratislava, Slovakia, 2003

7. P. Maji, C. Shaw, N. Ganguly, B. K. Sikdar and P.P. Chaudhuri, , Theory and Application of Cellular Automata for data mining. Fundamenta Informaticae, Issue 58, pp. 321-354, 2003
8. G. Widmer and M. Kubat, Learning is presence of concept drift and hidden contexts, Machine Learning, 23 (1), pp. 69-101, 1996
9. R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, Vol. 8 (3), 2004.
10. A. Tsymbal, The problem of concept drift: definitions and related work, 2004. Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Ireland.
11. Wei Fan, StreamMiner: A Classifier Ensemble-based Engine to Mine Concept-Drifting Data Streams, 2004, Very Large Databases, 1257-1260, 2004.
12. V. Vapnik, The Nature of Statistical Learning Theory, Springer, N.Y., 1995. ISBN 0-387-94559-8.
13. M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining Data Streams, A Review, ACM SIGMOD Record, Vol. 34 (2), pp. 18-26, June 2005.
14. P.L. Bartlett, S.B. David, and S.R. Kulkarni, Learning Changing Concepts by Exploiting the Structure of Change, Computational Learning Theory, pp. 131-139, 1996.
15. D.P. Helmbold and P.M. Long, Tracking Drifting Concepts by Minimizing Disagreement, Machine Learning, vol. 14 (1), pp. 27-45. 1994.
16. A.A. Freitas, A Genetic Programming Framework for two Data Mining Tasks: Classification and Generalized Rule Induction, Proceedings of the 2nd Annual Conference on Genetic Programming, pp.96-101, 1997, Stanford University, CA, USA.
17. M.D. Ryan and V.J. Rayward-Smith, The Evolution of Decision Trees, Proceedings of the 3rd Annual Conference on Genetic Programming, 1998, Morgan Kaufmann.
18. A. Loureiro, L. Torgo and C. Soares, Outlier Detection Using Clustering Methods: a Data Cleaning Application, In Proceedings of the Data Mining for Business Workshop, 2005, Porto, Portugal
19. X. Li and N. Ye. A supervised clustering and classification algorithm for mining data with mixed variables, IEEE Transactions on Systems, Man, and Cybernetics-Part A, Vol. 26, No. 2, 2006.
20. Ying Yang, Xidong Wu & Xingquan Zhu, Combining Proactive Reactive Predictions for Data Streams, Proceeding of the 11th ACM SIGKDD international conference on knowledge discovery in data mining, pp. 710-715, 2005.
21. G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 97–106, 2001.
22. H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept drifting data streams using ensemble classifiers. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 226–235, 2003.
23. Ying Yang, Xindong Wu and Xingquan Zhu, Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams. In Data Mining and Knowledge Discovery (DMKD), Volume 13, Number 3, 261-289, 2006
24. Xingquan Zhu, Xindong Wu and Ying Yang, Effective Classification of Noisy Data Streams with Attribute-Oriented Dynamic Classifier Selection. In Knowledge and Information Systems An International Journal (KAIS), Volume 9, Number 3, 339-363, 2006
25. D. J. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998
26. Jang JSR. ANFIS: Adaptive-network-based fuzzy inference systems. IEEE Transaction on System, Man and Cybernetic 23(3): 665–685, 1993
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)