Data Min Knowl Disc (2009) 19:95–131
DOI 10.1007/s10618-009-0130-9

Flexible decision tree for data stream classification in the presence of concept change, noise and missing values

Sattar Hashemi · Ying Yang

Received: 8 February 2008 / Accepted: 3 March 2009 / Published online: 17 March 2009

Abstract In recent years, classification learning for data streams has become an important and active research topic. A major challenge posed by data streams is that their underlying concepts can change over time, which requires current classifiers to be revised accordingly and timely. To detect concept change, a common methodology is to observe the online classification accuracy. If accuracy drops below some threshold value, a concept change is deemed to have taken place. An implicit assumption behind this methodology is that any drop in classification accuracy can be interpreted as a symptom of concept change. Unfortunately however, this assumption is often violated in the real world where data streams carry noise that can also introduce a significant reduction in classification accuracy. To compound this problem, traditional noise cleansing methods are incompetent for data streams. Those methods normally need to scan data multiple times whereas learning for data streams can only afford one-pass scan because of data's high speed and huge volume. Another open problem in data stream classification is how to deal with missing values. When new instances containing missing values arrive, how a learning model classifies them and how the learning model updates itself according to them is an issue whose solution is far from being explored. To solve these problems, this paper proposes a novel classification algorithm, flexible decision tree (FlexDT), which extends fuzzy logic to data stream classification. The advantages are three-fold. First, FlexDT offers a *flexible* structure to effectively and efficiently handle concept change. Second, FlexDT is robust to noise.

Responsible editor: Bart Goethals.

S. Hashemi
School of Electrical Engineering and Computer Sciences, Shiraz University, Shiraz, Iran
e-mail: s_hashemi@shirazu.ac.ir

Y. Yang (✉)
Australian Taxation Office, Melbourne, VIC, Australia
e-mail: ying.yang@ato.gov.au

Hence it can prevent noise from interfering with classification accuracy, and accuracy drop can be safely attributed to concept change. Third, it deals with missing values in an elegant way. Extensive evaluations are conducted to compare FlexDT with representative existing data stream classification algorithms using a large suite of data streams and various statistical tests. Experimental results suggest that FlexDT offers a significant benefit to data stream classification in real-world scenarios where concept change, noise and missing values coexist.

Keywords Classification learning · Data stream classification · Decision tree learning · Fuzzy learning

1 Introduction

Nowadays many applications deal with data that grow without limit at a rate of millions of records per day, such as sensor networks, stock exchanges, internet communications and intrusion detections. These data are called data streams and they differ from traditional static data in two ways. First, the data arrive at high speed and in huge volume (Domingos and Hulten 2000). For example, large banks often need to process millions of ATM or credit card transactions over a very short period of time. In this case, data as a whole can seldom be buffered into the main memory due to the size limitations. Storing the data into the hard disk is not advisable either because of the response time constraints. These conditions imply that an efficient classifier for data streams should meet each instance only once and should learn in an online manner. Second, the underlying concept of a data stream is often subject to changes along time (Tsymbal 2004; Widmer and Kubat 1996; Yang et al. 2005, 2006). For example, a slow wearing piece of factory equipment might cause a gradual change in the quality of output parts. For another example, customers' purchase patterns may change depending on factors such as the current day of the week and availability of alternatives. Hence, different from classifiers for static data, an effective classifier for data streams should be able to detect each concept change and revise itself according to arriving instances of the new concept. Many algorithms have been proposed to cope with the two unique and challenging features of data streams. Some representative ones are introduced as follows.

The concept-adapting very fast decision tree (CVFDT) (Hulten et al. 2001) is a well-known incremental classifier. It starts with a single leaf and collects labeled instances from a data stream. Using the Hoeffding bound, when it has enough data to determine with high confidence which attribute is the best to partition the data with, it turns the leaf into an internal node, splits on that attribute and starts learning at the new leaves recursively. CVFDT maintains a window of training instances and keeps its learned tree up to date with this window by monitoring the tree's accuracy as instances move into and out of the window. CVFDT periodically scans the internal nodes of the tree, looking for those where the chosen split attribute would no longer be selected because an alternative attribute now offers higher information gain. If this happens, CVFDT detects a concept change. It then starts growing an alternate subtree in parallel which is rooted at the new best attribute. When the alternate subtree is more

accurate on new data than the original one, the original is replaced by the alternate and freed.

The streaming ensemble algorithm (SEA) (Street and Kim 2001) takes an ensemble approach. It builds individual C4.5 decision trees (Quinlan 1993) from sequential chunks of instances in the stream. These classifiers are combined into an ensemble whose maximum size is fixed. To predict the class label of a new instance, the ensemble uses an unweighed majority vote, similar to bagging. Over time new classifiers can be added into the ensemble while old ones can be deleted from the ensemble according to a heuristic that favors classifiers that can correctly classify instances on which the ensemble is nearly undecided.

The weighed classifier ensemble (WCE) (Wang et al. 2003) is based on the authors' proof that a carefully weighed classifier ensemble built on a set of data partitions $S_1, S_2, \ldots, S_n$ is more accurate than a single classifier built on $S_1 \cup S_2 \ldots \cup S_n$. To classify a new instance, WCE divides its previous data into sequential chunks of fixed size, builds from each chunk a classifier such as the C4.5 decision tree, the RIPPER rule learner (Cohen 1995) or the naive Bayesian classifier (Mitchell 1997). These classifiers compose an ensemble where each classifier is weighed proportional to its classification accuracy on the chunk most recent to the instance to be classified.

With all due respect to previous achievements, this paper suggests that some problems remain. The first problem is how to deal with noise. All the above-mentioned algorithms assume that the decrease of classification accuracy can only be attributed to concept change. Hence they observe the online classification accuracy. Once the accuracy drops below some threshold value, they deem a concept change has taken place and start revising the classifiers. Unfortunately however real-world data streams often contain noise from a variety of corrupting processes, such as acquisition, transmission and transcription. Noise usually has adverse impact on interpretations of the data and introduces a significant reduction in classification accuracy. To compound this problem, traditional noise cleansing approaches are incompetent for data streams. Those methods normally need to scan data multiple times whereas learning in data streams can only afford one-pass scan because of data's high speed and huge volume. As a result, the above-mentioned representative classification algorithms can perform poorly facing noise because they cannot differ concept change from noise and will trigger a concept change detection regardlessly (Hashemi et al. 2007). A second issue is how to handle uncertainty in data streams due to missing values. This issue is very common in the real world as well because of factors including privacy and unavailability. All the above-mentioned algorithms do not explicitly address this issue. Meanwhile, conventional methods of handling missing values such as imputation, k-nearest neighborhood and prediction techniques were designed for static data and are unsuitable for streaming data because they incur expensive time complexities and often require multiple scans of the data (Saar-Tsechansky and Provost 2007; Mundfrom and Whitcomb 1998; Chan and Dunn 1972). Hence when new instances containing missing values arrive, how a learning model classifies them and how the learning model updates itself according to them is a problem whose solution is far from being explored.

Nonetheless some preliminary research has been reported on how to handle these open problems. One representative algorithm is the dynamic classifier selection (DCS)

(Zhu et al. 2004, 2006). DCS employs an ensemble of base classifiers and explores each classifier's domain expertise. It then dynamically selects a single "best" classifier to classify each incoming instance in a data stream. According to statistical information from attribute values, DCS uses each attribute to partition an evaluation data set (the most recent training instances) into disjoint subsets. It then evaluates the classification accuracy of each base classifier on these subsets. Given a new instance, its attribute values determine which subsets in the evaluation set it belongs to. The classifier with the highest classification accuracy among those subsets is selected to classify this new instance. Besides, DCS treats a missing value as a specific additional value of the corresponding attribute. Although DCS is proposed as a method especially targeting at noisy data streams, it can suffer from the following disadvantages. First, the size of the evaluation set critically affects the system performance whose optimal value is however difficult to determine. If the number of instances in the evaluation set is small, the model becomes very sensitive to noise. If the number is big, DCS cannot react to concept change fast. Second, to classify an instance, DCS has to go through the whole evaluation set in order to find the corresponding best classifier. As a result, its time complexity is very expensive for data streams. Third, DCS has difficulty in discovering each base classifier's domain expertise whenever attributes are numeric. When numeric attributes are involved in learning, DCS partitions them before the whole online learning process starts, and will never change the cut points again. However, because of concept changes, the appropriate cut points of the values of a numeric attribute are very likely to change accordingly. As a result, the original attribute partitioning can be no longer valid. Please note that DCS strongly relies on attributes' values to explore the domain expertise of each base classifier. Hence, having improper cut points is extremely misleading for DCS.

Inspired by these understandings, this paper suggests to integrate the representative power of decision trees with the knowledge component inherent in fuzzy sets in order to classify noisy data streams. The result of this integration, flexible decision tree (FlexDT), is a noise-robust model that handles concept changes in an effective and efficient way. FlexDT includes two phases, a forward phase and a backward phase. In the forward phase it classifies new arriving instances. In the backward phase it adapts to emergence of new concepts by tuning fuzzy parameters, where the parameter tuning is performed by using a novel efficient back-propagation method. Moreover, fuzzy sets enable FlexDT to effectively handle imprecise context such as when data contain missing values.

In summary the contributions of this paper and the novelties of FlexDT are as follows.

1. Our research for the first time extends fuzzy logic to data stream classification both in theory and in practice. The result is a novel fuzzy decision tree FlexDT that efficiently and effectively handles concept-change, possibly-noisy, large-volume and fast-speed streaming data.
2. FlexDT handles concept change using a novel back propagation approach, which is specially tailored for the decision tree structure and is significantly different from the neural network context. This new back propagation approach guaranties

that the classification error monotonically decreases with time going, and is very efficient in order to cope with streaming data.
3. FlexDT has the novelty of preventing noise from interfering with classification accuracy, and hence accuracy drop can be safely attributed to concept change.
4. FlexDT has the unique ability of not only classifying instances with missing values, but also learning from them.
5. The Hoeffding bound for growing data stream classification trees is for the first time adapted to a fuzzy learning context.
6. FlexDT adopts a novel binary fuzzy partitioning of numeric attributes together with the sigmoid membership function.

The rest of this paper is organized as follows. Section 2 introduces background knowledge of fuzzy sets and fuzzy decision trees. First, fuzzy sets are explained. Then fuzzy decision trees are studied in general while fuzzy ID3 (Umanol et al. 1994), a representative algorithm for classifying static data, is discussed in particular. Section 3 proposes the FlexDT algorithm and presents its advantages. FlexDT has been tested on a large suit of data streams, whose results are presented in Sect. 4. Section 5 summarizes the main contributions of this paper, and points out some future work.

2 Background knowledge

This section introduces background knowledge of fuzzy sets and fuzzy decision trees. First, fuzzy sets are explained. Then fuzzy decision trees are studied in general while fuzzy ID3 (Umanol et al. 1994), a representative algorithm for classifying static data, is discussed in particular.

2.1 Fuzzy sets

Set and *element* are two primitive notions of the *set theory*. In the *crisp* set theory, an element $\mathbf{x}$ can either be a member of a set S or not. Therefore, we can define the membership of the element $\mathbf{x}$ in the crisp set S as μ_S: $\mathbf{x} \rightarrow \{0, 1\}$, that is,

$$\mu_S(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in S, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

However, because of factors such as imprecise measurements and vagueness (Janikow 1998), many real-world sets are inherently fuzzy. For example, if we want to form a crisp set for the 'old' concept, it is unlikely to find an exact boundary point for the age of people to decide whether they belong to the 'old' concept or not. Instead there is a gradual change of membership (fuzziness) with respect to the age. The *fuzzy* set theory is a natural solution to this problem. It is a mathematical construct that allows an element to have partial membership in more than one set. The membership degree of the instance $\mathbf{x}$ in the fuzzy set S is defined as μ_S: $\mathbf{x} \rightarrow [0, 1]$, that is, $0 \leq \mu_S(\mathbf{x}) \leq 1$.

Assume that three fuzzy sets ('young', 'middle-age' and 'old') are assigned to the 'age' variable. The membership functions are defined by the widely used trapezoidal

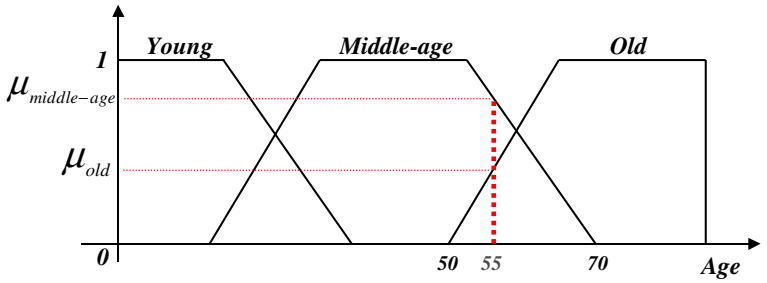

Fig. 1 Three fuzzy sets ('young', 'middle-age' and 'old') are assigned to the 'age' variable. For the age value 55, its membership degree belonging to the 'young' fuzzy set is 0, to the 'middle-age' fuzzy set is $\mu_{middle\text{-}age}(55)$, and to the 'old' fuzzy set is $\mu_{old}(55)$

functions (Zimmermann 2001). The fuzzy sets usually overlap to reflect the fuzziness of the 'age' concept. This is illustrated in Fig. 1 where a 55-year-old person belongs to both the 'middle-age' and 'old' sets with different membership degrees.

Given a set of n training instances $S = \{\langle \mathbf{x}^i, y^i \rangle | i = 1, \ldots, n\}$ where $\mathbf{x}^i = \{x^i_j | j = 1, \ldots, p\}$ is the ith instance and $y^i \in \{c_k | k = 1, \ldots, q\}$ is the class label of $\mathbf{x}^i$, the classification problem is to find the decision tree that minimizes the mapping error from input vectors $\mathbf{x}$ to class labels y for unlabeled instances. In fuzzy representation each attribute $\mathbf{x}_j$ $(j = 1, \ldots, p)$ is fuzzified into m fuzzy sets $F_1, \ldots, F_m$ (the value of m may vary from attribute to attribute). In contrast to crisp sets, fuzzy sets are not mutually exclusive, that is, each instance belongs to all sets each with a membership degree $\mu()$. The representation $\mu_{jF_v}(\mathbf{x}^i)$ $(v = 1, \ldots, m)$ is the degree to which x^i_j belongs to the fuzzy set F_v (the vth fuzzy set of the jth attribute). Similarly, the membership degree of the instance $\mathbf{x}^i$ belonging to the class c_k can be expressed by $\mu()$ as follows:

$$\mu_k(y^i) = y^i_k = \begin{cases} 1 & \text{if } y^i \text{ belogs to } k\text{th class,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2.2 Fuzzy decision tree

A fuzzy decision tree combines fuzzy sets with symbolic decision trees. This fusion enhances the representative power of decision trees with the knowledge component inherent in fuzzy logic, leading to better robustness (noise immunity) and applicability in imprecise context (Janikow and Kawa 2005; Mitra et al. 2002). Each fuzzy decision tree has three major components:

1. to partition each attribute into fuzzy sets and to assign membership degrees to its original values according to membership functions;
2. to use assigned fuzzy membership to induce an explicit fuzzy decision tree;
3. to infer from the tree in order to classify an instance.

In this section we use fuzzy ID3 (Maher and Clair 1993; Umanol et al. 1994), a representative popular fuzzy decision tree for classifying static data, as a vehicle of illustration to explain these three components in detail.

2.2.1 Fuzzy partitioning of attributes

In the fuzzy ID3 system, fuzzy sets and membership degrees for every attribute is provided by users.

2.2.2 Growing method

Regarding its growing method, FID3 is very similar to the crisp decision tree ID3 (Quinlan 1993) except that FID3 uses membership degrees of instances rather than their crisp values to compute the information gain at each node. Assume that we have a fuzzy set S of data with p attributes $\mathbf{x}_1, \ldots, \mathbf{x}_p$, a class label in $\{c_1, \ldots, c_q\}$, and fuzzy sets $F_1, \ldots, F_m$ for each attribute $\mathbf{x}_j$. Let S^{c_k} be the fuzzy subset in S whose class is c_k and let $|S|$ be the sum of the membership degrees in the fuzzy set S. Then, the second component to build the FID3 is shown in Algorithm 1.

The information gain $G(\mathbf{x}_j, S)$ of the attribute $\mathbf{x}_j$ in the fuzzy set S is defined by

$$G(\mathbf{x}_j, S) = I(S) - E(\mathbf{x}_j, S) \tag{3}$$

where

$$I(S) = -\sum_{k=1}^{q} (P(c_k, S) \cdot \log_2 P(c_k, S)), \tag{4}$$

$$E(\mathbf{x}_j, S) = \sum_{v=1}^{m} (P(S_{jv}, S) \cdot I(S_{jF_v})), \tag{5}$$

Algorithm 1 FID3 learning (Umanol et al. 1994)

1: Generate the root node with a fuzzy set comprising all instances with the membership degree 1;
2: **if** a node t with a fuzzy set of instances S satisfies one of the following conditions:
(**a**) the proportion of instances of a class c_k is greater than or equal to a threshold τ_1, that is, $\frac{S^{c_k}}{|S|} \geq \tau_1$;
(**b**) the number of instances is less than a threshold τ_2, that is, $|S| < \tau_2$;
(**c**) there are no attributes for further splitting;
then
3: The node t is a leaf and is assigned class labels with membership degrees;
4: **else**
5: **for** each $\mathbf{x}_j (j = 1, \ldots, p)$ **do**
6: Calculate the information gain $G(\mathbf{x}_j, S)$ following Eq. 3;
7: Select the test attribute $\mathbf{x}_{max}$ that maximizes the information gain;
8: **end for**
9: Divide S into fuzzy subsets $S_1, \ldots, S_m$ according to $\mathbf{x}_{max}$, where the membership degree of every instance $\mathbf{x}^i$ in S_v is the product of $\mathbf{x}^i$'s membership degree in S and the value of $\mu_{max,F_v}(\mathbf{x}^i)$;
10: **for** each fuzzy subset $S_v (v = 1, \ldots, m)$ **do**
11: Generate a new node t_v corresponding to S_v;
12: Label the membership functions $\mu_{max,v}$ on the edge that connects t_v to t;
13: Replace S by S_v;
14: Repeat from Line 2 recursively;
15: **end for**
16: **end if**

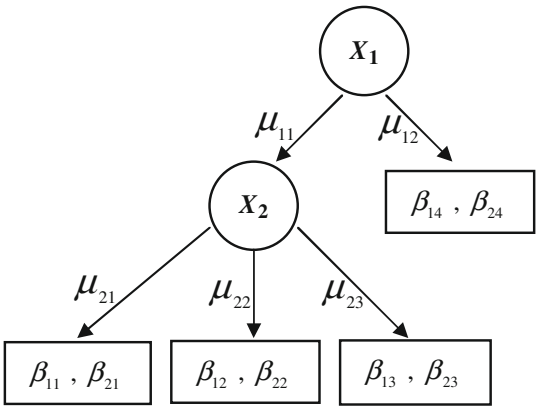

Fig. 2 An example fuzzy decision tree. The attribute $\mathbf{x}_1$ is the root and the attribute $\mathbf{x}_2$ is an internal node. The vth membership function of the jth attribute is defined by μ_{jv}. The probability of the kth class at the lth leaf is expressed by β_{kl}

$$P(c_k, S) = \frac{|S^{c_k}|}{|S|}, \tag{6}$$

$$P(S_{jv}, S) = \frac{|S_{jF_v}|}{\sum_{v=1}^{m} |S_{jF_v}|}. \tag{7}$$

If at least one of the conditions (**a**), (**b**) or (**c**) at Line 2 of Algorithm 1 is satisfied, the algorithm refrains from further splitting the node, and the node is assigned class labels together with membership degrees. This is in contrast to traditional decision trees that assign just one class label to each leaf. For assigning each membership degree $\beta_{kl} (0 \leq \beta_{kl} \leq 1)$ to the kth class at the lth leaf, FID3 takes into account all data:

$$\beta_{kl} = \frac{\sum_{i=1}^{n} \prod_{j \in path_l} \mu_{jl}(\mathbf{x}^i) \mu_{kl}(y^i)}{\sum_{i=1}^{n} \prod_{j \in path_l} \mu_{jl}(\mathbf{x}^i)} \tag{8}$$

where $path_l$ is composed of the attributes that $\mathbf{x}^i$ has past when traversing from the root node to the lth leaf, and $\mu_{kl}(y^i)$ is the membership degree of $\mathbf{x}^i$ in the class k at the leaf l.

Figure 2 shows a basic fuzzy decision tree structure. The tree consists of two nodes $\mathbf{x}_1$ and $\mathbf{x}_2$, four leaves (rectangles, $l = 4$), and a set of branches leading to fuzzy sets of each corresponding attribute for which the membership function is μ_{jv}. The number of classes in the shown tree is 2 ($q = 2$). Thus each leaf ℓ contains two fuzzy variables $\beta_{1\ell}$ and $\beta_{2\ell}$.

2.2.3 Classifying new instances

When an unlabeled instance arrives to be classified, it traverses the tree from the root downward to the leaves of the tree. If the instance has a nonzero membership in at least one of the fuzzy sets associated with a node, the membership degree(s) is calculated and the instance is sent along the path(es) corresponding to the fuzzy set(s). The process is followed recursively such that the instance meets all probable leaves. The third component, to calculate the probability of the instance $\mathbf{x}^i$ belonging to the class k, sums up the probability values across all leaves:

$$\hat{y}_k^i = \sum_{\ell=1}^{l} \mu_{path_\ell}(\mathbf{x}^i) \times \beta_{k\ell} \tag{9}$$

where $0 \leq \hat{y}_k^i \leq 1$ $(k = 1, \ldots, q)$. Then, the final class label estimated by the learning model for the $\mathbf{x}^i$) is $\hat{y}^i = \arg \max_{k=1,\ldots,q} \{\hat{y}_k^i\}$.

3 Flexible decision tree

We suggest that an ideal classification model for data streams classification should

- be able to learn in an online manner, that is, meet each instance only once;
- handle concept change and distinguish concept change from noise;
- be applicable in imprecise context, that is, deal with uncertainty such as missing values.

We propose a novel data stream classifier, flexible decision tree (FlexDT), which can meet these requirements. FlexDT has the following advantages. First, it adopts incremental learning and only needs to process each instance once. Second, it has a flexible structure that can adapt to concept change effectively and efficiently. Third, inherited from fuzzy logic, FlexDT is a noise-robust model. Moderate noise turbulence in streaming data hardly affects FlexDT's overall classification performance. As a result, FlexDT can distinguish noise from concept change, prevent noise from interfering with classification accuracy, and hence accuracy drop can be safely attributed to concept change. Fourth, FlexDT is adept at handling data uncertainty such as missing values. If an instance arrives with a missing value for any attribute, that instance will belong to all fuzzy sets of that attribute with equal membership degrees. To the best of our knowledge, FlexDT is the first algorithm ever developed that extends fuzzy logic's merits to data stream classification.

As fuzzy decision trees in general, FlexDT has three major components (Sect. 2.2). *The first two components, fuzzy partitioning and tree growing, are different from the traditional fuzzy ID3. The third component, classifying an instance, is the same as fuzzy ID3* (Eq. 9). In the following sections, we focus on the first two components of FlexDT. Meanwhile, how FlexDT handles emergence of new concepts, noise and missing values is also explored.

3.1 Fuzzy partitioning of attributes

The traditional fuzzy ID3 assumes that the fuzzy partitions are a prior definition provided by users. In contrast, we provide an automatic scheme for assigning fuzzy sets to each attribute. Our scheme is based on binary partitioning of attributes and the sigmoidal membership function.

Generally there are two options for fuzzy partitioning of attributes, binary or multiple. We select binary fuzzy partitioning for the following reasons. First, binary partitioning is more efficient. This is critical in order to cope with streaming data's

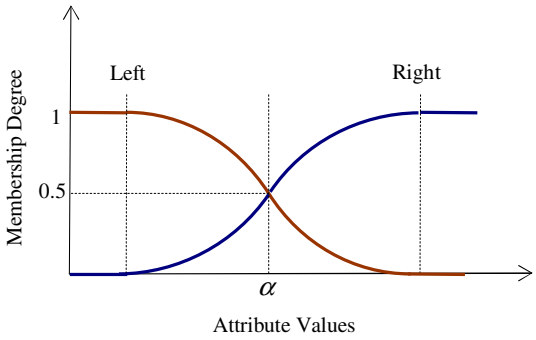

Fig. 3 Binary fuzzy partitioning in FlexDT. The cut point α as well as the left and right sigmoidal functions can take varieties of values, which gives FlexDT desired flexibility

high speed and large volume. Second, binary partitioning offers FlexDT more flexibility since the parameters of adopted fuzzy sets can change in wider ranges with more optional values (σ and α as in Fig. 3).

There are also a variety of alternative membership functions that can be used for fuzzification of input attributes. We chose the sigmoidal membership function. The sigmoidal membership function has the following characteristics that lend itself as a suitable choice for our model.

1. It is differentiable. This is an essential property for handling concept changes to be detailed such as in Eq. 15 of Sect. 3.4.
2. It is complete. For each value it provides at least one non-zero membership degree (fuzzy set). The completeness implies that the resulting fuzzy sets cover all values of an attribute. It also provides the inference system a smooth transition form one set to the other, which in turn can improve classification accuracy (Jang 1993).
3. It is consistent. If a value's membership degree to one fuzzy set equals to 1, its membership degree to all other fuzzy sets is 0. The consistency implies that if a value of an attribute certainly belongs to one set, it shall not belong to any other set (consistent with the crisp set theory).
4. It is normal. The maximum value of membership degree in each fuzzy set is 1. Normal fuzzy sets offer the model computational efficiency (Zimmermann 2001).

Our partitioning model is depicted in Fig. 3. The membership functions for the left and right terms of the jth attribute of the instance $\mathbf{x}^i$ are defined as follows:

$$\mu_{j_L}(\mathbf{x}^i) = \frac{1}{1 + e^{(\frac{x^i_j - \alpha_j}{\sigma_{jL}})}}, \tag{10}$$

$$\mu_{j_R}(\mathbf{x}^i) = \frac{1}{1 + e^{-(\frac{x^i_j - \alpha_j}{\sigma_{jR}})}} \tag{11}$$

where α_j and σ_j are respectively the center and standard deviation of each sigmoidal membership function. The value of α_j is obtained in a way similar to the cut point selection in traditional crisp decision trees (Fayyad and Irani 1993). Given a set of

fuzzy instances S and an attribute $\mathbf{x}_j$, the class information entropy of the left and right partitions (S_L, S_R) induced by every probable cut point α is calculate as

$$E(\mathbf{x}_j, S) = \frac{|S_L|}{|S|} I(S_L) + \frac{|S_R|}{|S|} I(S_R) \tag{12}$$

where $I(S) = \sum_{k=1}^{q} P(c_k, S) \log(P(c_k, S))$. The cut point α_j is selected whose $E(\mathbf{x}_j, \alpha, S)$ is minimal amongst all candidates. Keeping the cut point α_j installed, the initial values of σ_j for the left and right fuzzy sets are obtained as the standard deviation of instances in S_L and S_R respectively. *The values of α_j and σ_j are updated during the learning task to give FlexDT the flexibility to adapt to changing concepts, as to be detailed in Sect. 3.4.*

It is worth mentioning that if the corresponding attribute is nominal, we use the crisp set theory to partition the attribute, that is, partitioning is performed according to the attribute's values. This fact does not affect the generality of our proposed algorithm because the adopted fuzzy sets are consistent with the crisp set theory according to the sigmoidal membership function's characteristic 3.

The proposed partitioning scheme is different from that of DCS's as discussed in Sect. 1. First, FlexDT's attribute partitioning is performed on demand (whenever it is necessary at different levels of the tree) whereas DCS's is carried out as a preprocess before the whole learning procedure starts. Producing cut point selection at different levels of the tree enables the algorithm to better focus on the current concept, that is, cut points are more durable. Second, cut points can change in FlexDT when concept change happens. However in DCS once partitioning is performed cut points can no longer change.

3.2 Growing method

Instances in a data stream arrive at high speed and in huge volume. As a result, it is impossible to store all instances in memory and learn from them as a whole. One common solution is to make decision based on a portion of data at each time.

Given the first portion of instances, FlexDT selects the attribute that maximizes the information gain, assigns to it the membership functions and installs it at the root node of the tree. Upon arrival of new instances to the recently installed node, they are past down along all branches with the corresponding membership degrees. The same story is repeated for the rest of nodes in a recursive manner.

The question arises here is how many instances are enough for turning a leaf to an internal node and growing the tree downward. FID3 uses the predefined conditions (2.a, 2.b and 2.c in Algorithm 1) to decide the number of instances needed for this purpose. FlexDT on the other hand utilizes the Hoeffding bound (Domingos and Hulten 2000; Hulten et al. 2001) to decide whether to expand the tree or not. Consider a real-valued random variable R whose range is $\Re$ (for information gain, the range is $\log c$ where c is the number of classes). Suppose we have made n' independent observations of an attribute, and computed its mean value $\bar{R}$. The Hoeffding bound states that, with probability $1 - \delta$, the true mean value of this attribute is at least $\bar{R} - \epsilon$, where

$$\epsilon = \sqrt{\frac{\Re^2 ln(1/\delta)}{2n'}}. \tag{13}$$

The Hoeffding bound is extended to select the attribute with the highest information gain at each node after observing enough number of instances. It ensures that, with the probability $(1-\delta)$, the attribute chosen using a small portion of data with n' instances is the same as that would be chosen using the entire data stream (n instances). Let $G(\mathbf{x}_j)$ be the information gain used to choose the test attribute $\mathbf{x}_j$ at each node. Assume G is to be maximized. Let $\mathbf{X}_a$ be the attribute with highest observed $\bar{G}$ after seeing n' instances, and let $\mathbf{X}_b$ be the second best attribute. Let $\Delta\bar{G} = \bar{G}(\mathbf{X}_a) - \bar{G}(\mathbf{X}_b) \geq 0$ be the difference between their information gains after observing n' instances. Then, given a desired δ, the Hoeffding bound guarantees that $\mathbf{X}_a$ is the correct choice with the probability $1-\delta$ if n' instances have been seen at this node and $\Delta\bar{G} > \epsilon$. It is important to note that, given the probability δ, the value of ϵ is obtained using Eq. 13. Therefore, a node needs to first accumulate instances from the stream until ϵ becomes smaller than $\Delta\bar{G}$ (notice that ϵ is a monotonically decreasing function of n'). Then the node can be split using the current best attribute, and succeeding instances will be past to the new leaves.

3.3 Being robust to noise and dealing with uncertainty

Most real-world sets are inherently fuzzy. If one tries to find crisp boundary point to differentiate one set from the others, that point would be very fragile. In particular, the crisp cut point selection invoked by the traditional algorithms may have very negative impacts on the instances whose attribute values are close to the cut point. Two instances that are close to each other in the attribute space may follow separate branches and therefore locate faraway from each other in the output space. In such a case, the presence of noise worsens the problem. Whenever noise occurs, the boundary instances follow different branches in a stochastic manner depending on the amount and direction of noise. This is why the accuracy of traditional algorithms in noisy domain decreases dramatically. On the contrary, FlexDT treats two instances close to each other in the attribute space in a similar fashion. In such a case, it assigns almost the same membership function to the instances and hence the bias of the instances to one particular branch decreases, which eliminates the effect of noise. Moreover, in classification phase the effect of all leaves the instance has met are taken into account. Even if one branch is followed mistakenly by a particular instance due to noise in data, this mistake can be compensated by other leaves' classifications.

The same property, sending an instance along more than one branch with corresponding membership degrees, also enables FlexDT to deal with uncertainty in an elegant way. In the case that an instance has an unknown value, it is evenly sent to both left and right branches. Therefore, an instance with missing values can be classified following Eq. 9, and moreover, can be fully used to contribute to building the classifier which in turn improves the performance of FlexDT. This is in contrast to DCS that treats a missing value as a specific additional value of the corresponding attribute and only classifies instances with missing values rather than learn from them.

3.4 Adapting to emergence of new concepts

To handle concept change, FlexDT adopts the back-propagation approach previously proposed for neural networks and neuro-fuzzy models (Basak 2006; Bhatt and Gopal 2006; Haykin 1994). However the traditional back propagation models are not directly suitable for mining high-speed data streams because they incur high cost and are inefficient. This is mainly due to the fact that they comprise several forward and backward iterations to tune the parameters and minimize the error of mapping from the input space to the output class vectors, that is, each instance is met many times during the training phase.

To solve these problems, FlexDT proposes to have one forward and one backward cycle for each instance, that is, each instance is met only once. During the forward phase, the instance is sent along the tree to be classified at the corresponding leaves using Eq. 9. In the backward phase the error being made at the classification phase is propagated toward the root to tune the fuzzy parameters so as to minimize classification error. It is important to note that once FlexDT has classified a new instance, the membership degrees of all nonzero paths from the root to the leaves are computed. As a result, the tuning is merely traversing the tree from its leaves upward and updating β, σ and α using Eqs. 15–21 at each node in sequence. By this scheme, the change of each fuzzy variable at a particular node is simply a function of cumulative changes in its right and left children which is performed very efficiently.

We define the error of FlexDT for classifying an instance $\mathbf{x}_i$ as the differentiable square error:

$$Err(\mathbf{x}^i) = \frac{1}{2}\sum_{k=1}^{q}(y_k^i - \hat{y}_k^i)^2 \tag{14}$$

where $y_k \in \{0, 1\}$ depends on whether the instance belongs to the class k or not, and $\hat{y}_k \in [0, 1]$, $k = (1, \ldots, q)$ is the fuzzy classification output of the model.

To minimize the error *Err*, it is essential that the differentiations of *Err* with respect to α, σ and β approach toward zero. Hence, subject to gradient descent, the parameters are changed as follow

$$\Delta\beta_{kl} = -\eta\frac{\partial Err(\mathbf{x}^i)}{\partial\beta_{kl}}, \tag{15}$$

$$\Delta\sigma_{jL} = -\eta\frac{\partial Err(\mathbf{x}^i)}{\partial\sigma_{jL}}\ ,\ \Delta\sigma_{jR} = -\eta\frac{\partial Err(\mathbf{x}^i)}{\partial\sigma_{jR}}, \tag{16}$$

$$\Delta\alpha_j = -\eta\frac{\partial Err(\mathbf{x}^i)}{\partial\alpha_j}, \tag{17}$$

where η is the learning rate and upon arrival of new instances, changes in an adaptive fashion according to the variation in the learning parameters:

$$\eta = \frac{1}{\sqrt{\Delta\beta^2 + \Delta\alpha^2 + \Delta\sigma^2}}.$$

Having the above equations along with Eqs. 9 and 14, it is straightforward to obtain the update rules for learning parameters so that *Err* is minimized. The update rule for β at the leaves of FlexDT with the sigmoidal membership function is

$$\Delta\beta_{kl} = \eta \left(y_k^i - \hat{y}_k^i\right) \mu_{path_l}(\mathbf{x}^i). \tag{18}$$

Also, the update rules for σ are obtained using partial derivation:

$$\begin{aligned}
\Delta\sigma_{jL} &= -\eta \frac{\partial Err(\mathbf{x}^i)}{\partial \sigma_{jL}} \\
&= -\eta \frac{\partial Err(\mathbf{x}^i)}{\partial \mu_{jL}(\mathbf{x}^i)} \frac{\partial \mu_{jL}(\mathbf{x}^i)}{\partial \sigma_{jL}} \\
&= -\eta \left(\frac{x_j^i - \alpha_j}{{\sigma_{jL}}^2}\right) (1 - \mu_{jL}(\mathbf{x}^i)) \sum_{\ell \in LSL_j} \left(\mu_{path_\ell}(\mathbf{x}^i) \sum_{k=1}^{q} \beta_{k\ell}(y_k^i - \hat{y}_k^i)\right),
\end{aligned} \tag{19}$$

$$\begin{aligned}
\Delta\sigma_{jR} &= -\eta \frac{\partial Err(\mathbf{x}^i)}{\partial \sigma_{jR}} \\
&= -\eta \frac{\partial Err(\mathbf{x}^i)}{\partial \mu_{jR}(\mathbf{x}^i)} \frac{\partial \mu_{jR}(\mathbf{x}^i)}{\partial \sigma_{jR}} \\
&= \eta \left(\frac{x_j^i - \alpha_j}{{\sigma_{jR}}^2}\right) (1 - \mu_{jR}(\mathbf{x}^i)) \sum_{\ell \in RSL_j} \left(\mu_{path_\ell}(\mathbf{x}^i) \sum_{k=1}^{q} \beta_{k\ell}(y_k^i - \hat{y}_k^i)\right),
\end{aligned} \tag{20}$$

where LSL_j and RSL_j are the left and right subtree leaves of the node associated with the attribute x_j respectively.

The parameter α_j on the other hand, affects the error being made at leaves of both left and right subtrees of the attribute x_j (according to Eqs. 10 and 11). Consequently, we have

$$\begin{aligned}
\Delta\alpha_j &= -\eta \frac{\partial Err(\mathbf{x}^i)}{\partial \alpha_j} \\
&= -\eta \left(\frac{\partial Err(\mathbf{x}^i)}{\partial \mu_{jL}(\mathbf{x}^i)} \frac{\partial \mu_{jL}(\mathbf{x}^i)}{\partial \alpha_j} + \frac{\partial Err(\mathbf{x}^i)}{\partial \mu_{jR}(\mathbf{x}^i)} \frac{\partial \mu_{jR}(\mathbf{x}^i)}{\partial \alpha_j}\right) \\
&= \eta \left(\frac{1}{\sigma_{jR}}\right) (1 - \mu_{jR}(\mathbf{x}^i)) \sum_{\ell \in RSL_j} \left(\mu_{path_\ell}(\mathbf{x}^i) \sum_{k=1}^{q} \beta_{k\ell}(y_k^i - \hat{y}_k^i)\right) \\
&\quad -\eta \left(\frac{1}{\sigma_{jL}}\right) (1 - \mu_{jL}(\mathbf{x}^i)) \sum_{\ell \in LSL_j} \left(\mu_{path_\ell}(\mathbf{x}^i) \sum_{k=1}^{q} \beta_{k\ell}(y_k^i - \hat{y}_k^i)\right).
\end{aligned} \tag{21}$$

Our derived formulae also give FlexDT the capability to induce from nominal data. To do so, FlexDT splits the node of concern based on all values of the nominal

attribute. The instances attached to the node then are past along the branches with crisp membership function (0 or 1) depending on their attribute's value. The rest of the procedure is the same as that of numeric attributes. However, there are no parameters to be tuned at the nodes corresponding to nominal attributes and hence the flexibility of FlexDT decreases. The higher the level of a nominal attribute is in the tree structure, the less flexible FlexDT becomes.

Now the question remains to answer is how to handle the situations in which a concept change causes a node of the tree and its corresponding subtree to become obsolete. In such a case, FlexDT deploys a strategy similar to that of CVFDT's. It grows an alternative subtree whenever some node that previously past the Hoeffding test can no longer do so because an alternative attribute now offers higher information gain at the node. The old subtree is replaced by the new one when the new one becomes more accurate.

Algorithm 2 presents the pseudo codes of learning FlexDT in streaming data.

4 Experiments and analysis

Experiments are conducted to evaluate FlexDT's classification performance on handling concept change, noise and missing values. FlexDT is compared with popular existing data stream classification algorithms including CVFDT, SEA and WCE, as well as the representative algorithm for classifying noisy data streams, DCS. All the algorithms have some parameters to be tuned. This paper deploys typical parameter settings used by each algorithm's corresponding paper. CVFDT uses $\delta = 0.01$, $f = 200$, $n_{min} = 100$, $\tau = 0.05$ and $w = 1000$. SEA has 25 base classifiers ($ensemble\_size$), each trained on 500 instances ($chunk\_size$). WCE comprises 10 base classifiers, each trained on 1000 instances. DCS contains 5 base classifiers, each trained on 1000 instances. DCS considers the most recent instances as the evaluation set because they are most likely to be consistent with the current test instances. The base classifier of all the ensemble models is the C4.5 decision tree (Quinlan 1993) with its default parameter settings.[1] FlexDT's parameters are the same as those of CVFDT's with the additional parameter $\eta = 0.1$.

4.1 Experimental data and design

A large suite of 24 data streams are used as listed in Table 1. Each of the data streams is commonly used by previous published research on data stream classification learning.

4.1.1 Simulating concept change

Hyperplane data stream (Hulten et al. 2001; Wang et al. 2003, 2005; Yang et al. 2005) is a synthetic data stream commonly used by published research on data stream classification. A hyperplane in a d-dimensional space is denoted by $\sum_{i=1}^{d} w_i x_i = w_0$,

[1] The original DCS uses C4.5 rules (Quinlan 1993). We use C4.5 decision trees (Quinlan 1993) here in order to impose a fair comparison among rival approaches.

Algorithm 2 FlexDT learning in streaming data

Let $FlexDT$ be the flexible decision tree to be induced;
while a new instance $\mathbf{x}^i$ arrives **do**
  if there is no node in $FlexDT$ other than the root **then**
    Classify $\mathbf{x}^i$ using given β in the root;
    Compute classification error according to Eq. 14;
    Add $\mathbf{x}^i$ to the root and update its statistics. Also update β following Eq. 18 (path's membership degree at this step equals to 1, that is, $\mu_{path}(\mathbf{x}^i) = 1$);
  else
    if for the attribute x_j installed at this node, the value of x_j^i is known **then**
      Send $\mathbf{x}^i$ along the left and right branches with the membership degrees $\mu_{jL}(\mathbf{x}^i)$ and $\mu_{jR}(\mathbf{x}^i)$ respectively;
    else
      Send the instance $\mathbf{x}^i$ evenly along both children;
    end if
    Repeat the previous step for both left and right sub-trees in recursive manner until $\mathbf{x}^i$ reaches corresponding leaves;
    Compute $\hat{y}_k^i$ $(k = 1, \ldots, q)$ using Eq. 9 and classify $\mathbf{x}^i$ accordingly;
    Calculate the classification error being made by $FlexDT$ to classify $\mathbf{x}^i$, which is $Err(\mathbf{x}^i)$ (Eq. 14);
    Update the statistics of all leaves $\mathbf{x}^i$ has reached;
    Change β at leaves followed by δ and α at internal nodes by the small distance η in a direction to decrease $Err(\mathbf{x}^i)$; (Eqs. 18, 19, 20 and 21);
    Set $\eta = \frac{1}{\sqrt{\Delta\beta^2 + \Delta\alpha^2 + \Delta\sigma^2}}$;
  end if
  while there exists a leaf ℓ that satisfies the following conditions:
  (**a**) the instances seen so far at ℓ are not from the same class, that is, given a predefined threshold $0 < \tau < 1, \tau \leq \beta_{k\ell} \leq 1 - \tau$ $(k = 1, \ldots, q)$;
  (**b**) the difference between the attribute with the highest information gain and the attribute with the second highest information gain at ℓ passes the Hoeffding bound (Eq. 13) due to arrival of the new instance $\mathbf{x}^i$;
  do
    Let $\mathbf{x}_j$ be the attribute with the highest information gain at ℓ;
    Find the cut point α_j that minimizes entropy (Eq. 12);
    Assign left and right sigmoidal membership functions to the attribute $\mathbf{x}_j$ (Eqs. 10 and 11);
    Replace ℓ by internal node, split on α_j, and attach new leaves ℓ_L and ℓ_R to ℓ;
    Send ℓ's instances to ℓ_L and ℓ_R by multiplying their membership functions with $\mu_{jL}(\mathbf{x}^i)$ and $\mu_{jR}(\mathbf{x}^i)$ respectively;
    Compute $\beta_{k\ell_L}$ and $\beta_{k\ell_R}$ $(k = 1, \ldots, q)$ according to fuzzy instances assigned to ℓ_L and ℓ_R following Eq. 8;
  end while
end while

where each vector of variables $\langle x_1, x_2, \ldots, x_d \rangle$ is a randomly generated instance and is uniformly distributed in the space $[0.0, 1.0]^d$. Instances satisfying $\sum_{i=1}^{d} w_i x_i \geq w_0$ 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. Altogether this data stream has 50,000 instances.

SEA data stream (Kolter and Maloof 2003; Street and Kim 2001) is another popular synthetic data stream. It has three attributes $x_i \in [0.0, 10.0]$. The underlying concept is if $x_1 + x_2 \leq b$ and x_3 is an irrelevant attribute, the instance is positive; otherwise

Table 1 Each experimental data stream's origin, name, number of numeric attributes (NumAtt.), number of nominal attributes (NomAtt.), number of classes (Cls.), and number of instances (Ins.)

Origin	Data set	NumAtt.	NomAtt.	Cls.	Ins.
Synthetic data	Hyperplane	d	0	2	50000
	SEA	3	0	2	50000
	USPSHandwritten	256	0	10	50000
UCI data	Adult	6	8	2	48842
	Annealing	6	32	6	898
	AustralianSignLanguage	8	0	3	12546
	Balance	4	0	3	625
	BreastCancerWisconsin	9	0	2	683
	CreditScreening	6	9	2	690
	Diabetes	8	0	2	768
	German	7	13	2	1000
	Glass	9	0	6	214
	HeartDisease	7	6	2	270
	Hepatitis	6	13	2	155
	HorseColic	7	14	2	368
	Ionosphere	34	0	2	351
	Iris	4	0	3	150
	LiverDisorder	6	0	2	345
	SatelliteImage	36	0	6	4435
	Sonar	60	0	2	208
	Thyroid	7	21	4	3772
	Vehicle	18	0	4	846
	Waveform	21	0	3	5000
	Wine	13	0	3	178

the instance is negative. There are four alternative values for b: 7.0, 8.0, 9.0, 9.5. To compose a data stream, five segments each with 10,000 instances are queued in sequence. To simulate concept change, the value of b changes from one segment to the next.

USPS handwritten digits data stream (Ho 2005) contains handwritten image data in 10 classes with 256 attributes. It has 9298 instances. A concept-change data stream is formed by queuing ten different data segments, each corresponding to a fixed set of three different digits $\{x, y, z\}$ in a random fashion. The three-digit set changes from one segment to the next. For each segment, 5,000 instances each belonging to x, y or z are randomly selected. More details of this data stream are listed in Table 2.

UCI data streams employed in this study stem from public benchmark data sets from the UCI Machine Learning Repository (Newman et al. 1998) that have been commonly used by previous published research, as listed in Table 1. Following the practice of previous papers (Yang et al. 2006), to produce a data stream, given a data

Table 2 USPS data stream is composed of ten data segments

Segment	Digit x	Digit y	Digit z
I	0 (1767)	1 (1502)	2 (1731)
II	0 (1597)	3 (1658)	4 (1745)
III	1 (1856)	5 (1755)	6 (1389)
⋮	⋮	⋮	⋮

Each segment corresponds to the concept of a three-digit set $\{x, y, z\}$. Each row corresponds to a segment and indicates its values of x, y and z as well as the total number of instances belonging to each of them in parentheses

set, one attribute is randomly selected. If the attribute is nominal, each of its values is treated as a state in the Markov Chain. For each state 2000 instances possessing the same attribute value are randomly selected from the data set, and are queued into the data stream. If the selected attribute is numeric, all instances are ascendingly sorted according to its value and queued into the data stream. All together, each data stream has 50,000 instances.

4.1.2 Simulating noise

Real-world data often contain noise such as typos and corruptions. In classification learning, there are two common types of noise, attribute noise and class noise (Zhu and Wu 2004). In this paper it is assumed that data streams suffer from both types of noise. To add noise into a data stream, the pairwise scheme (Zhu et al. 2003, 2006) is adopted. Given the noise level γ and a pair of instances (I_x, I_y) with different class labels x and y, the value of one randomly-selected attribute (including the class label) of I_x has γ probability to be replaced by that of I_y's, and vice versa. However for each instance this scheme only introduces noise into one randomly-selected attribute (including the class label). Thus it will have less effect on a data stream with a bigger number of attributes. To have a consistent effect of noise across data streams with different numbers of attributes, we have ranked the attributes according to their correlation with the class. The higher the rank of an attribute, the higher chance it has to be selected and corrupted. Please note that under this new scheme, the class label has the highest probability to be corrupted. This is a desirable property because it balances class noise and attribute noise in data streams considering that every data stream contains one class label and multiple attributes.

4.1.3 Having missing values

UCI data sets sometimes naturally have missing values. We use the top three data sets that contain the most number of missing values in our data suite to explore FlexDT's ability of classifying under missing values. These top three data sets are Annealing, Horse Colic and Credit Screening.

4.1.4 Performance measured

We record our experimental results as follows.

- *Incremental learning curve* of classification accuracy for an algorithm A on a data stream D is a two-dimensional curve whose horizontal axis corresponds to the number of instances in D that have been classified by A by the time t, and whose vertical axis corresponds to A's classification accuracy averaged across all already-classified instances by the time t.
- *Overall classification accuracy* of an algorithm A on a data stream D is recorded when A finishes classifying every instance in D. It equals to the number of correctly classified instances divided by the total number of instances in D.
- *Run time* measures an algorithm's efficiency. It equals to an algorithm's total running time averaged over a data stream with different noise levels. In the online learning context of data stream classification, run time composes learning time and classification time simultaneously.
- *Convergence time* is the time the classification system remains unstable during updating to a concept change, averaged over a data stream with different noise levels. Due to concept change or noise, there can sometimes be a significant drop in classification accuracy. The convergence time is the time the classifier takes to return to its original accuracy.

4.1.5 Statistical tests employed

A variety of statistical tests are employed to evaluate measured performance of each competing scheme.

- *Mean of ranks* Following the practice of Friedman (1937, 1940), for each data stream, we rank competing algorithms. The one that attains the best performance (such as the highest overall classification accuracy, the shortest run time or the shortest convergence time) is ranked 1, the second best ranked 2, so on and so forth. A method's mean rank is obtained by averaging its ranks across all data streams. Compared with mean value (the arithmetic mean of measured performance, such as accuracy, averaged across all data streams), mean rank can reduce the susceptibility to outliers that, for instance, allows a classifier's excellent performance on one data stream to compensate for its overall bad performance (Demsar 2006).
- *Friedman test* As recommended by Demsar (2006), the Friedman test is effective for comparing multiple algorithms across multiple data streams. It compares mean ranks of schemes to decide whether to reject the null-hypothesis, which states that all the schemes are equivalent and so their ranks should be equal.
- *Nemenyi test* If the Friedman test rejects its null-hypothesis, we can proceed with a post-hoc test, the Nemenyi test (Demsar 2006). It can be applied to mean ranks of competing schemes and indicate whose performances have statistically significant differences (here we use the 0.05 critical level).
- *Win/lose/tie record and binomial sign test* A win/lose/tie record can be calculated for each pair of competing algorithms A and B with regard to a performance measure M. The record represents the number of data streams in which A respectively

beats, loses to or ties with B on M. A one-tailed binomial sign test can be applied to wins versus losses. If its result is less than the critical level of 0.05, the wins against losses are statistically significant, supporting the claim that the winner has a systematic (instead of by chance) advantage over the loser.

4.2 Observations and analysis

Rival algorithms' learning accuracy and efficiency are presented and analyzed in this section.

4.2.1 Classification accuracy facing noise

This section evaluates how well FlexDT performs when classifying noisy data streams. Four rounds of experiments are conducted, each with a different level of noise across the 24 data streams. The overall classification accuracy of each algorithm on each data stream is detailed in Table 6 in the Appendix to avoid distraction from the main text.

First we examine the experimental results when the noise level $\gamma = 0$. According to Table 6, one can calculate that the mean ranks of CVFDT, DCS, FlexDT, SEA and WCE are respectively 3.5417, 3.0833, 2.1250, 3.6667 and 2.5833. When we apply the Friedman test, with 5 algorithms and 24 data streams, F_F is distributed according to the F distribution with $(5-1) = 4$ and $(5-1) \times (24-1) = 92$ degrees of freedom. The critical value of $F(4, 92)$ at the 0.05 critical level is 2.47. F_F calculated from the mean ranks is 4.66. Since $4.66 > 2.47$, we can reject the null hypothesis and infer that there exists significant difference among rival schemes.

To find out exactly which schemes are significantly different, we proceed to the Nemenyi test, whose results are illustrated in Fig. 4a. The mean rank of each scheme is pointed by a circle. The horizontal bar across each circle indicates the é6critical differenceć6. The performance of two methods is significantly different if their corresponding mean ranks differ by at least the critical difference. In other words, two methods are significantly different if their horizontal bars are not overlapping. For instance, Fig. 4a reveals that when the noise level $\gamma = 0.0$, FlexDT is ranked best and is significantly better than CVFDT and SEA.

The same statistical test procedure is repeatedly applied to rival methods' classification accuracy at each noise level. The results are illustrated in Fig. 4. It is observed that FlexDT outperforms rival algorithms at every noise level (ranked as the best). With noise increasing in data streams, FlexDT's advantage over others becomes stronger because its rank keeps increasing. DCS is ranked second best when $\gamma = 0.1$ and $\gamma = 0.2$. The main reason why FlexDT and DCS outperform the rest of approaches is because both methods have mechanisms especially designed to handle noisy data while other methods have none. However, when noise becomes serious ($\gamma = 0.3$), DCS is overtaken by WCE. One possible reason is that when noise increases, each of DCS's base classifiers becomes weaker. In the case that all base classifiers have low confidence, combing results from multiple base classifiers (WCE) becomes more reasonable than using a single unconfident classifier.

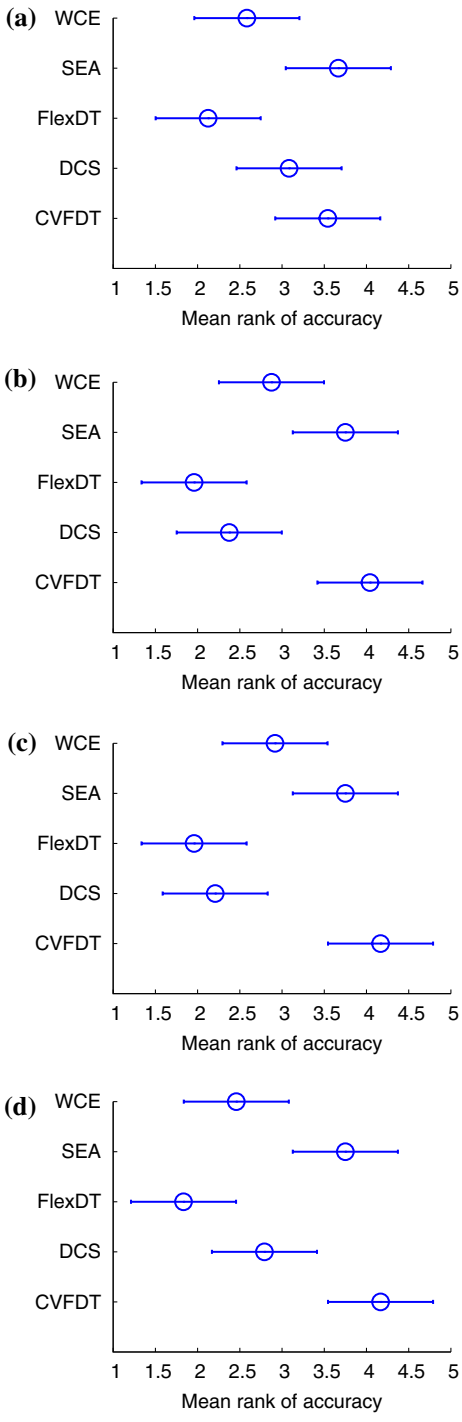

Fig. 4 Apply the Nemenyi test to alternative schemes' mean ranks of classification accuracy. **a** Noise level $\gamma = 0.0$; **b** Noise level $\gamma = 0.1$; **c** Noise level $\gamma = 0.2$; **d** Noise level $\gamma = 0.3$

In contrast, although FlexDT also uses a single fuzzy decision tree, it is still the best facing large amount of noise. We suggest two reasons for this good performance. One reason is that FlexDT uses fuzzy partitioning. All other algorithms select crisp cut points. As explained in Sect. 3.3, crisp partitioning can have very negative impacts on the instances with attribute values close to the cut point. This is why the accuracy of CVFDT in noisy domain decreases dramatically. When CVFDT's accuracy decreases, it will start to prune old branches and grow new ones despite that the underlying concept actually remains unchanged. WCE and SEA suffer from the same problem because their base classifiers are likewise very sensitive to noise. Therefore, their base classifiers are frequently replaced with new ones even when no concept change happens. DCS can alleviate this problem by using statistical information of attribute values. FlexDT is most robust to noise by smoothing each attribute's cut points.

Another reason for FlexDT's better accuracy relates to its capability of outputting class probability estimates beyond simple 0/1-loss classifications (Eq. 9). As a consequence, FlexDT can be considered as a continuous parametric function. This enables the system to minimize classification error through the backward phase, where the fuzzy parameters are tuned to adapt to a concept change and improve the predictive accuracy in face of the concept change. This kind of adaption cannot be achieved by any other competing method in this study because they only carry out 0/1-loss classification.

To further explore FlexDT's performance on handling concept change, we present the incremental learning curves of rival algorithms on two data streams, the Satellite Image data stream and the SEA data stream at different levels of noise. Figure 5 depicts the incremental learning curve for each rival method on the Satellite Image data stream. FlexDT's accuracy is always on top of others with time going on as well as with noise increasing. CVFDT as a single incremental classifier always performs worse than the ensemble methods (WCE and SEA). This difference increases when noise becomes more severe, which reveals CVFDT's instability particularly in noisy environments. DCS is surprisingly less accurate than most rival methods although its accuracy gets closer to the others with noise increasing in data. One possible reason is that Satellite Image has a large number (36) of attributes. Hence the complexity of finding domain expertise of each base classifier (the search space of DCS) increases and DCS has more difficulty in pinpointing the correct domain expertise of the base classifiers. Therefore it tends to be suboptimal. Nonetheless, facing increasing noise, DCS's improvement in relative accuracy suggests the statistical information of attributes used by DCS helps DCS to be more robust to noise than other previous methods. Figure 6 depicts the incremental learning curve for each rival method on the SEA data stream. Again, FlexDT is the method of choice with time going on as well as with noise increasing. This time DCS becomes the second best at most of the time because the SEA data stream has a small number (3) of attributes. However, when noise becomes extremely serious, DCS' base classifiers become weaker and selecting a single one weak classifier turns out to be an inadvisable solution. That is why the ensemble methods (SEA and WCE) take over DCS in Fig. 6d when $\gamma = 0.3$. Meanwhile, CVFDT remains susceptible to noise. Its accuracy drops dramatically when the noise level increases from $\gamma = 0.0$ to $\gamma = 0.3$.

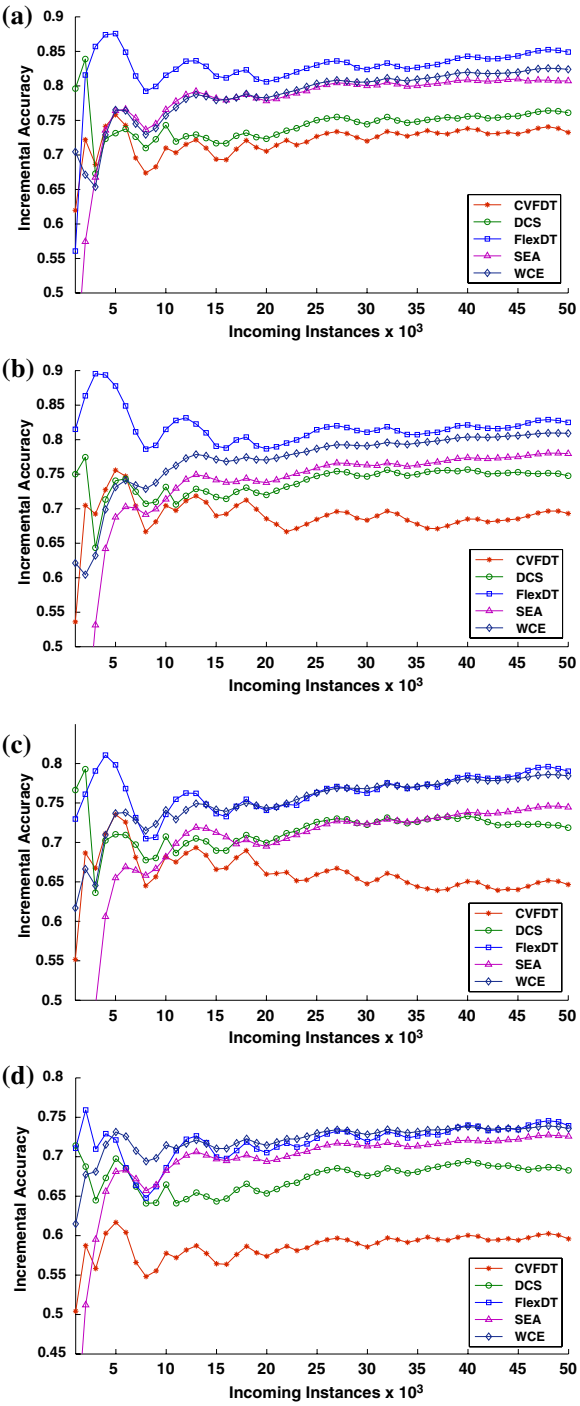

Fig. 5 Incremental learning curves of rival algorithms on the Satellite Image data stream at different levels of noise. **a** Noise level $\gamma = 0.0$; **b** Noise level $\gamma = 0.1$; **c** Noise level $\gamma = 0.2$; **d** Noise level $\gamma = 0.3$

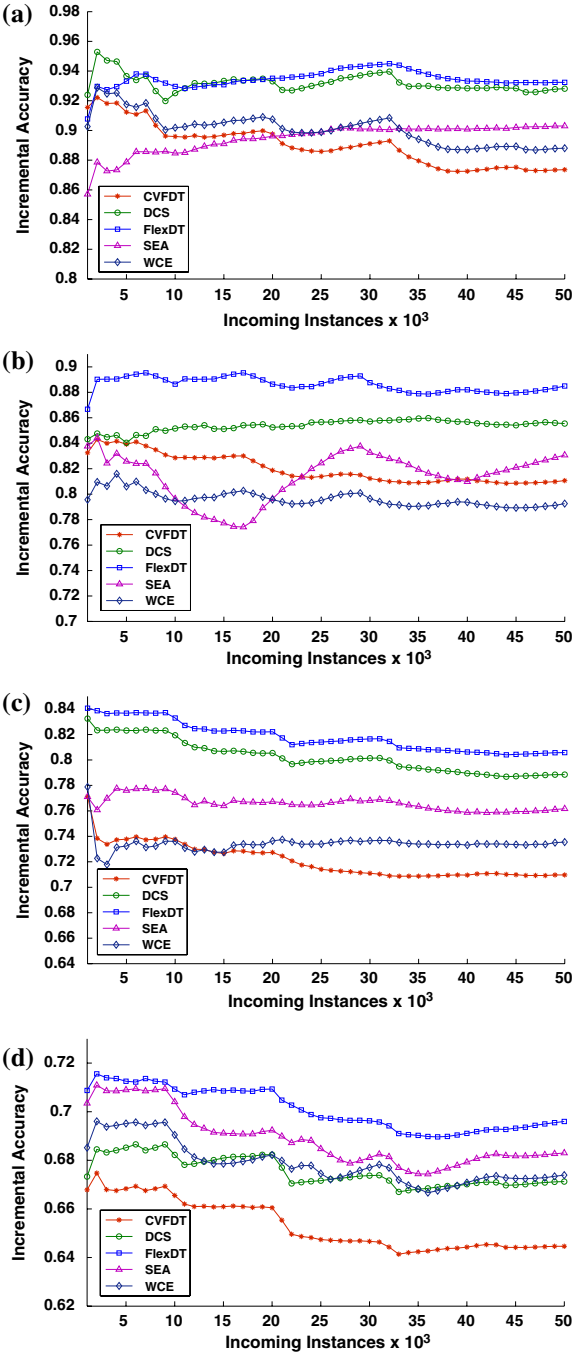

Fig. 6 Incremental learning curves of rival algorithms on the SEA data stream at different levels of noise. **a** Noise level $\gamma = 0.0$; **b** Noise level $\gamma = 0.1$; **c** Noise level $\gamma = 0.2$; **d** Noise level $\gamma = 0.3$

One interesting scenario to mention is the German data stream. According to Table 6, FlexDT only offers mediocre performance at different levels of noise. Exploring the reason we have found that German data stream comprises 13 nominal attributes and 7 numeric ones. Whenever instances stream in, the nominal attributes are more frequently selected at the higher level of the decision tree. As explained in Sect. 3.4, this decreases the flexibility of the final FlexDT model and hence affects its accuracy. Moreover, there is no missing values in the German data stream that otherwise can give FlexDT opportunity to make up its accuracy compared with other approaches.

To complete the picture, we have listed in Table 3 the statistical win/lose/tie records among rival methods at different levels of noise. A bold face entry indicates that the wins against losses are statistically significant using a one-tailed binomial sign test at the critical level 0.05. The records attest that very often FlexDT's wins versus

Table 3 Rival algorithms' win/lose/tie records with regard to their overall classification accuracy across 24 data streams

Method	CVFDT	DCS	FlexDT	SEA	WCE
(a) Noise level $\gamma = 0.0$					
CVFDT	0/0/24	9/15/0	5/19/0	12/12/0	9/15/0
DCS	15/9/0	0/0/24	7/17/0	14/10/0	10/14/0
FlexDT	**19/5/0**	**17/7/0**	0/0/24	**18/6/0**	15/9/0
SEA	12/12/0	10/14/0	6/18/0	0/0/24	4/20/0
WCE	15/9/0	14/10/0	9/15/0	**20/4/0**	0/0/24
(b) Noise level $\gamma = 0.1$					
CVFDT	0/0/24	4/20/0	2/22/0	10/14/0	7/17/0
DCS	**20/4/0**	0/0/24	10/14/0	**19/5/0**	14/10/0
FlexDT	**22/2/0**	14/10/0	0/0/24	**19/5/0**	**18/6/0**
SEA	14/10/0	5/19/0	5/19/0	0/0/24	6/18/0
WCE	**17/7/0**	10/14/0	6/18/0	**18/6/0**	0/0/24
(c) Noise level $\gamma = 0.2$					
CVFDT	0/0/24	3/21/0	2/22/0	9/15/0	6/18/0
DCS	**21/3/0**	0/0/24	10/14/0	**19/5/0**	**17/7/0**
FlexDT	**22/2/0**	14/10/0	0/0/24	**20/4/0**	**17/7/0**
SEA	15/9/0	5/19/0	4/20/0	0/0/24	6/18/0
WCE	**18/6/0**	7/17/0	7/17/0	**18/6/0**	0/0/24
(d) Noise level $\gamma = 0.3$					
CVFDT	0/0/24	4/20/0	3/21/0	9/15/0	4/20/0
DCS	**20/4/0**	0/0/24	6/18/0	**17/7/0**	10/14/0
FlexDT	**21/3/0**	**18/6/0**	0/0/24	**22/2/0**	15/9/0
SEA	15/9/0	7/17/0	2/22/0	0/0/24	6/18/0
WCE	**20/4/0**	14/10/0	9/15/0	**18/6/0**	0/0/24

Each entry indicates that the algorithm of the row compares against the algorithm of the column. A statistically significant record (at the 0.05 critical level) is indicated in a bold face

losses against rival methods are of statistically significant frequency across the 24 data streams.

4.2.2 Classification accuracy facing missing values

To evaluate FlexDT's performance facing missing values, we conduct experiments on the top three data streams that have the most number of inherent missing values: Annealing, Horse Colic and Credit Screening (as explained in Sect. 4.1.3).

First, we observe rival algorithms' behaviors in a noisy free environment. Thus the advantage of a learner can be purely attributed to its strength on coping with missing values. Figure 7 illustrates alternative algorithms' incremental learning curves on each data stream. These figures are arranged according to the ascending order of the amount of missing values in each data stream. In other words, from Fig. 7a–c, the amount of uncertainty in the data stream increases. It is observed that in each data stream FlexDT always achieves the best accuracy among alternatives. With the amount of missing values increases, previous algorithms' performance zigzags a lot. It is because when new instances containing missing values arrive, these previous learners may have difficulty in classifying them as well as updating itself according to them. Hence, their accuracy goes up and drops down frequently. The more uncertain a data stream is, the more variation in accuracy and the more erratic behavior a learner would have. In contrast, FlexDT largely maintains a smooth pattern while the uncertainty increases, which attests its strength on coping with missing values for data stream classification.

Second, we test rival algorithms on data streams where both noise and missing values exist. Figure 8 depicts the results. The horizontal axis corresponds to the noise level. The vertical axis corresponds to each algorithm's overall classification accuracy on each data stream at some noise level. It is observed that in general FlexDT and DCS are the best two algorithms. The main reason is that FlexDT and DCS have strategies especially developed to handle missing values while other algorithms have none. In addition, FlexDT has advantage over DCS because FlexDT can both classify uncertain instances and learn from them, while DCS only classifies these instances without learning from them. This advantage becomes stronger when the amount of missing values increases.

4.2.3 Learning efficiency

To assess rival algorithms' learning efficiency, we consider two criteria as defined in Sect. 4.1.4, *run time* that is the total running time of each algorithm averaged over a data stream with different levels of noise, and *convergence time* that is the average time the classification system remains unstable during updating to a concept change. All the experiments are conducted on a 2.4GHz Pentium IV machine with 1GB of main memory. The results of run time and convergence time are respectively presented in Tables 7 and 8 in the Appendix to avoid distraction from the main text. Please note that the less the time an algorithm takes, the better it should be ranked.

To statistically analyze the results, we again apply the Friedman test and the Nemenyi test on the results. As for run time, according to Table 7, we can reject the null

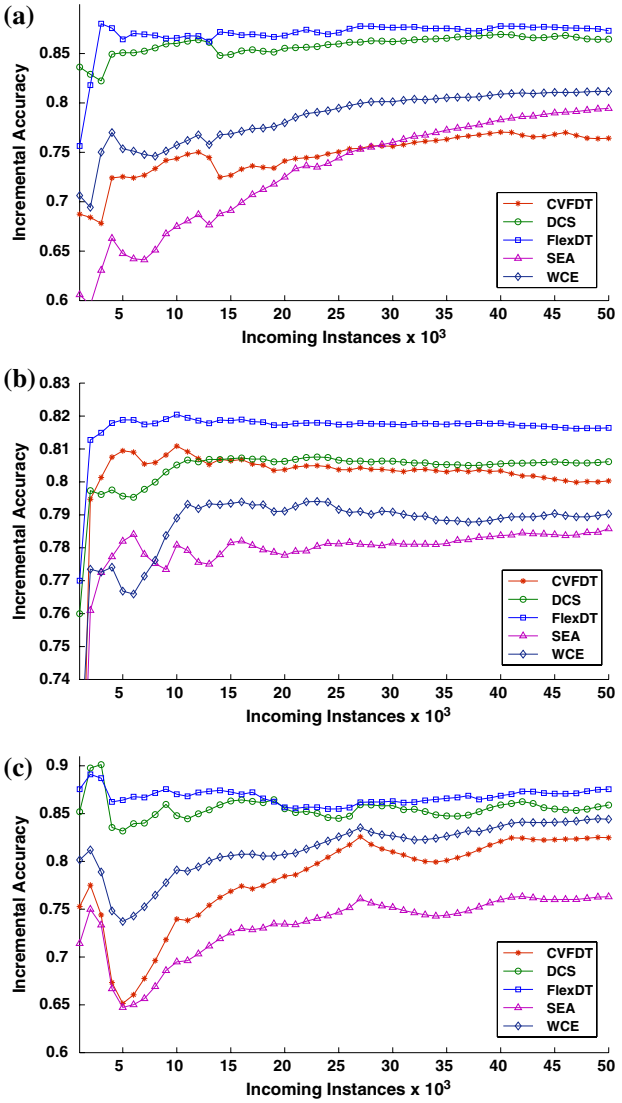

Fig. 7 Incremental learning curves of rival algorithms in data streams containing missing values. Figures are arranged according to the ascending order of the amount of missing values in each data stream. FlexDT often outperforms alternatives. **a** CreditScreening; **b** HorseColic; **c** Annealing

hypothesis of the Friedman test and infer that there exists significant difference among rival schemes. Then we proceed to the Nemenyi test, whose results are illustrated in Fig. 9. The win/lose/tie records for the run time are presented in Table 4.

According to Fig. 9 and Table 4, CVFDT is the fastest algorithm because it only deploys a single classifier and it does not contain any backward phase like FlexDT. However, CVFDT's efficiency is at the cost of its suboptimal accuracy facing noise. Please note that we have observed that the running time of CVFDT increases with

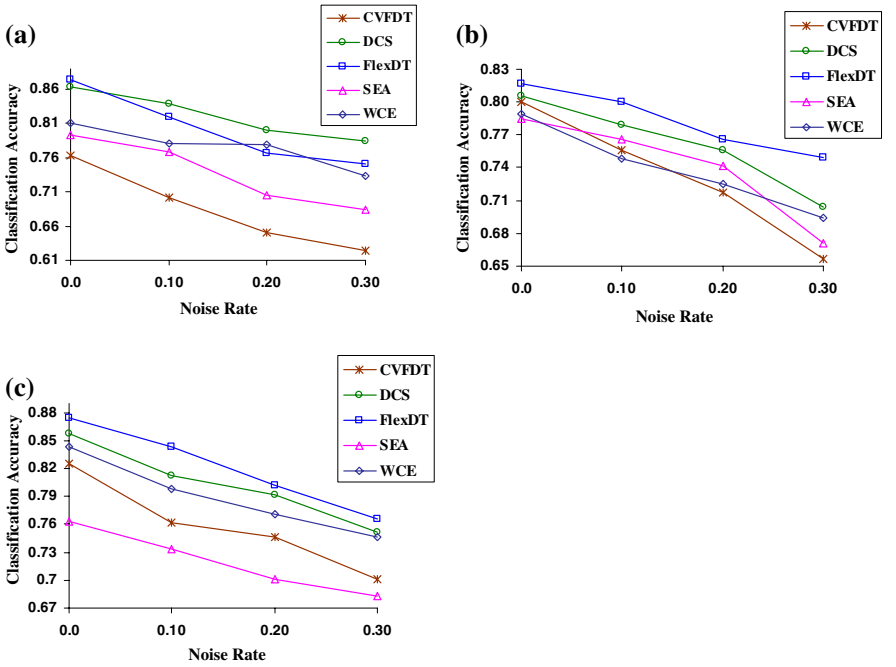

Fig. 8 Overall classification accuracy of rival algorithms on data streams containing uncertainty at different levels of noise. Figures are arranged according to the ascending order of the amount of missing values in each data stream. FlexDT and DCS outperform other rival methods. **a** CreditScreening; **b** HorseColic; **c** Annealing

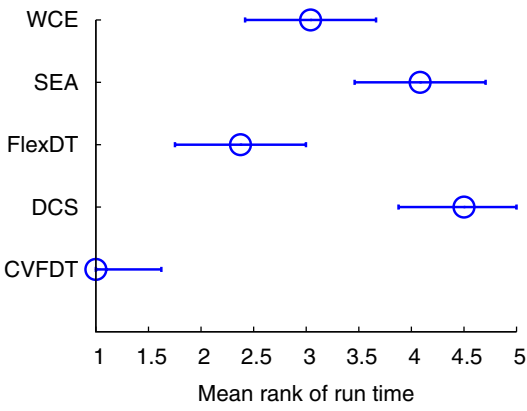

Fig. 9 Apply the Nemenyi test to alternative schemes' mean ranks of run time

the noise level increasing although its average running time is still lower than alternative approaches. FlexDT is the second fastest followed by WCE, SEA and DCS respectively. FlexDT's wins versus losses against WCE, SEA and DCS are all of statistically significant frequency. Between the two ensemble approaches, WCE is faster than SEA because the former involves only 10 component classifiers whereas the latter includes 25 classifiers. DCS is the slowest approach over the data streams studied in this research because of two reasons. First, it is very time consuming to find domain

Table 4 Rival algorithms' win/lose/tie records across 24 data streams regarding *run time*

Method	CVFDT	DCS	FlexDT	SEA	WCE
CVFDT	0/0/24	**24/0/0**	**24/0/0**	**24/0/0**	**24/0/0**
DCS	0/24/0	0/0/24	4/20/0	5/19/0	3/21/0
FlexDT	0/24/0	**20/4/0**	0/0/24	**22/2/0**	**21/3/0**
SEA	0/24/0	**19/5/0**	2/22/0	0/0/24	1/23/0
WCE	0/24/0	**21/3/0**	3/21/0	**23/1/0**	0/0/24

Each entry indicates the result when the algorithm of the row compares against the algorithm of the column. A statistically significant record (at the 0.05 critical level) is indicated in a bold face

expertise of each classifier in the ensemble during the learning phase. Second, during the classification phase, DCS has to go through the whole evaluation set to find the corresponding classifier to classify an instance. Nonetheless, it is worth mentioning that whenever there are only a small number of attributes in the data streams such as Balance, Iris and SEA, DCS provides acceptable efficiency.

As for convergence time, according to Table 8, we can reject the null hypothesis of the Friedman test and infer that there exists significant difference among rival schemes. Then we proceed to the Nemenyi test, whose results are illustrated in Fig. 10. The win/lose/tie records for the convergence time are presented in Table 5.

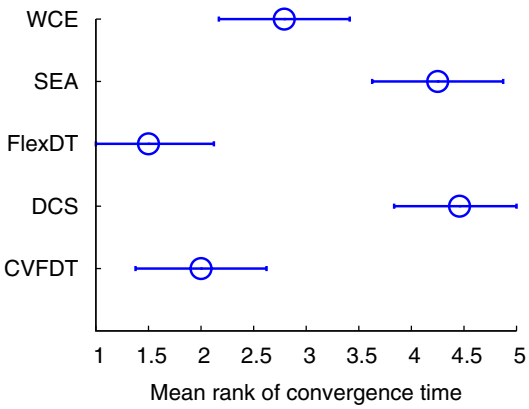

Fig. 10 Apply the Nemenyi test to alternative schemes' mean ranks of convergence time

Table 5 Rival algorithms' win/lose/tie records across 24 data streams regarding *convergence time*

Method	CVFDT	DCS	FlexDT	SEA	WCE
CVFDT	0/0/24	**21/3/0**	7/17/0	**23/1/0**	**21/3/0**
DCS	3/21/0	0/0/24	2/22/0	7/17/0	1/23/0
FlexDT	**17/7/0**	**22/2/0**	0/0/24	**24/0/0**	**21/3/0**
SEA	1/23/0	**17/7/0**	0/24/0	0/0/24	0/24/0
WCE	3/21/0	**23/1/0**	3/21/0	**24/0/0**	0/0/24

Each entry indicates the result when the algorithm of the row compares against the algorithm of the column. A statistically significant record (at the 0.05 critical level) is indicated in a bold face

According to Fig. 10 and Table 5, FlexDT offers the fastest convergence among all alternatives and its wins versus losses against rival algorithms are all of statistically significant frequency. CVFDT is the second fastest followed by WCE, SEA and DCS respectively. It is worth mentioning that, although CVFDT is the second best in term of convergence time, for example, it converges to a new concept faster than WCE does, CVFDT delivers lower accuracy compared to WCE. The reason lies in the sensitivity of CVFDT. CVFDT converges to every change very fast even when it is actually due to noise or uncertainty rather than a true concept change. This characteristic is not desirable for mining real-world data streams where concept change, noise and missing values coexist. In contrast, FlexDT is a noise-robust model that can identify each true concept change and adapt to it efficiently and effectively. Compared with DCS that also explicitly deal with noise, FlexDT as a single interpretable classifier is statistically significantly more efficient for data stream classification.

5 Conclusion and future work

This paper focuses on improving classification performance for data streams when concept change, noise and missing values coexist. This is an important problem to solve because noise and missing values are pervasive in real-world data streams, and yet a problem that has not received well-deserved attention from the data mining community. On the other hand, traditional fuzzy learning methodologies have the merit of being robust to noise and missing values. This paper for the first time extends fuzzy logic's strength to data stream classification both in theory and in practice. The result is an online data stream classification algorithm FlexDT that has the following capabilities.

- FlexDT is robust to noise. This is achieved by fuzzy partitioning of attributes, sending each instance along more than one branch with corresponding membership degrees, and taking into account of all leaves to classify an instance. As a result, moderate noise turbulence in streaming data hardly affects FlexDT's overall classification performance. Thus FlexDT can prevent noise from interfering with classification accuracy, and accuracy drop can be safely attributed to concept change.
- FlexDT can effectively and efficiently adapt to a new concept. This is achieved by outputting classification results as continuous values (class probability estimates instead of simple 0/1-loss classifications), and then using a novel efficient back-propagation method to tune the tree's parameters so as to minimize the classification error. Besides, binary partitioning of attributes offers FlexDT more flexibility since the parameters of adopted fuzzy sets can change in wider ranges with more optional values.
- FlexDT can not only classify instances with missing values, but also learn from them. If an instance has an unknown value for an attribute associated with a node, this instance is evenly sent to all branches with corresponding membership degrees. Hence, each instance with missing values can be classified following Eq. 9, and moreover, can fully participate in building the classifier.

To verify FlexDT's efficacy and efficiency, we have conducted extensive experiments, using 24 benchmark data sets as well as a variety of established statistical tests. Empirical evidence attests that FlexDT can statistically significantly outperform the state-of-the-art data stream classification algorithms including CVFDT, SEA, WCE and DCS in the presence of concept change, noise and missing values.

Some future work is named below.

- The flexibility of FlexDT decreases when a data stream contains nominal attributes in addition to numeric attributes. Our future work will extend the model in such a way that it provides desired flexibility for nominal attributes as well.
- It has been shown that in the batch mode of decision tree learning from static data sets, multiple partitioning of attributes can improve the classification accuracy compared with binary partitioning (Fayyad and Irani 1993; Olaru and Wehenkel 2003). However whether this conclusion still stands in the context of data stream mining is a question yet to be answered. In our current study we have employed binary partitioning to maximize FlexDT's efficiency and flexibility. Using multiple partitioning instead is expected to incur higher time complexity and higher classification variance, that is, the cut points need to be revised more frequently along time when new instances become available. Our future work will investigate the performance of FlexDT with multiple attribute partitioning for data stream classification.
- In order to evaluate alternative algorithms' efficacy on handling missing values, our current study tests and compares algorithms on data sets that have natural missing values. We deem this as a more fair approach than using synthetic missing values because the latter can (unfairly) bias towards or against some classifiers depending on the artificial distribution of the missing values, which can have many options. Nonetheless, we think it would be interesting to further explore this issue of artificially producing missing values and observing the behaviors of alternative algorithms with the missing value amount change.

Appendix

See Tables 6, 7, and 8

Table 6 Each rival algorithm's overall classification accuracy (%) on each data stream

Data stream	CVFDT	DCS	FlexDT	SEA	WCE
Noise level $\gamma = \mathbf{0.0}$					
Adult	81.56	80.76	82.16	77.21	80.32
Annealing	82.52	85.71	87.5	76.27	84.37
AustralianSign	63.36	73.82	66.63	70.19	76.11
Balance	81.28	83.14	86.59	85.83	86.21
BreastCancerWisconsin	95.45	97.75	97.16	98.33	97.51
CreditScreening	76.39	86.44	87.5	79.39	81.13

Table 6 continued

Data stream	CVFDT	DCS	FlexDT	SEA	WCE
Diabetes	78.49	80.78	82.24	81.37	83.9
German	78.92	82.78	79.94	79.34	81.49
Glass	78.62	89.26	86.33	88.19	89.11
HeartDisease	90.23	89.31	90	87.12	83.24
Hepatitis	84.43	92.77	88.9	71.25	79.04
HorseColic	80	80.59	81.63	78.47	78.94
Hyperplane	82.05	87.11	85.8	91.3	93.16
Ionosphere	82.75	85.3	86.55	89.65	89.67
Iris	98.56	97	98.67	98	98.21
LiverDisorders	88.3	85.28	87.31	88.08	86.54
SatelliteImage	73.84	76.34	85.14	80.71	82.51
SEA	87.34	92.78	93.24	90.3	88.78
Sonar	95.89	91	95.33	91.64	93.99
Thyroid	94.4	93.43	92.74	90.24	96.37
USPSHandwritten	73.1	72.3	76.1	69	74.23
Vehicle	86.11	83.24	85.82	82.08	85.26
Waveform	77.91	83.11	85.78	84.37	87.42
Wine	94.35	93.43	97.45	91.33	94.24
Noise level $\gamma = \mathbf{0.1}$					
Adult	76.34	77.41	80.88	74.35	77.82
Annealing	76.16	81.26	84.33	73.4	79.78
AustralianSign	59.73	71.61	63.79	67.02	72.58
Balance	77.18	82.3	83.73	80.81	81.93
BreastCancerWisconsin	89.73	96.01	95.06	95.19	95.71
CreditScreening	70.13	83.93	81.89	76.93	78.01
Diabetes	76.34	78.39	80.99	77.87	80.63
German	76.9	78.3	77.71	77.21	78.65
Glass	73.22	85.94	84.78	86.43	86.66
HeartDisease	84.58	84.2	83.49	83.12	75.92
Hepatitis	81.21	88.2	86.31	68.02	76.03
HorseColic	75.61	77.91	79.98	76.58	74.84
Hyperplane	73.97	80.49	79.28	85	81.92
Ionosphere	77.72	84.33	83.43	83.34	84.87
Iris	94.51	95.42	96.53	93.12	94.27
LiverDisorders	80.81	82.51	85.74	84.29	84.59
SatelliteImage	69.66	75.08	82.77	78.04	80.96
SEA	80.98	85.59	88.33	82.87	79.12
Sonar	92.48	90.61	92.19	88.45	88.05
Thyroid	88.9	88.86	90.35	87.24	88.94

Table 6 continued

Data stream	CVFDT	DCS	FlexDT	SEA	WCE
USPSHandwritten	65.94	68.94	69.6	58.25	66.34
Vehicle	82.76	84.59	84.54	81.54	83.88
Waveform	72.48	82.4	83.5	83.59	82.19
Wine	93.34	92.41	96.31	89.69	91.9
Noise level $\gamma = \mathbf{0.2}$					
Adult	73.45	76.12	77.32	71.14	75.39
Annealing	74.69	79.22	80.25	70.17	77.14
AustralianSign	56.59	67.02	62.59	61.98	68.39
Balance	73.75	77.92	76.45	74.01	76.19
BreastCancerWisconsin	85.85	93.42	94.34	92.84	94.29
CreditScreening	65.13	80.1	76.64	70.46	77.83
Diabetes	73.14	77.62	78.63	74.31	77.37
German	74.9	76.28	75.94	73.18	75.03
Glass	71.43	84.5	81.84	84.25	84.57
HeartDisease	81	82.17	78.03	77.14	70.15
Hepatitis	73.33	83.39	82.21	65.25	71.13
HorseColic	71.7	75.56	76.64	74.14	72.52
Hyperplane	67.31	76.07	74.04	76.11	75.2
Ionosphere	73.81	81.71	81.28	80.86	81.38
Iris	91	93.92	95.16	91.05	92.29
LiverDisorders	76.46	80.33	79.3	81.16	81.84
SatelliteImage	65.07	72.16	79.36	74.59	78.57
SEA	71.02	78.86	80.59	76.35	73.78
Sonar	87.31	86.16	86.19	87.21	86.74
Thyroid	85.99	84.26	89.49	84.81	85.37
USPSHandwritten	62.94	64.52	66.71	55.14	60.93
Vehicle	77.8	82.31	83.53	80.59	81.84
Waveform	70.73	81.02	81.83	80.23	79.07
Wine	91.26	91.14	94.76	87.99	89.2
Noise level $\gamma = \mathbf{0.3}$					
Adult	71.7	75.76	76.51	68.13	74.33
Annealing	70.14	75.14	76.56	68.31	74.69
AustralianSign	55.36	62.9	60.75	61.39	65.93
Balance	70.5	74.66	76.13	72.41	76.28
BreastCancerWisconsin	82.44	88.35	92.13	90.84	92.35
CreditScreening	62.33	78.47	75.14	68.4	73.25
Diabetes	71.93	73.91	73.48	70.92	74.85
German	72.9	74.47	74.22	72.83	75.86
Glass	70.39	77.88	78.15	82.13	82.28

Table 6 continued

Data stream	CVFDT	DCS	FlexDT	SEA	WCE
HeartDisease	78.12	77.49	76.69	70.23	68.13
Hepatitis	64.47	76.33	80	61.27	66.78
HorseColic	65.67	70.37	74.91	67.13	69.37
Hyperplane	59.23	68.31	69.43	67.09	64.04
Ionosphere	68.59	79.13	78.52	77.86	79.49
Iris	90.85	90.59	93.03	89.88	90.12
LiverDisorders	70.3	74.76	76.66	75.57	79.2
SatelliteImage	60.06	68.6	74.42	72.66	73.8
SEA	64.45	67.14	69.67	68.4	67.51
Sonar	78.48	83.8	86.9	86.18	85.99
Thyroid	79.93	83.29	88.42	81.24	80.34
USPSHandwritten	60.94	58.01	60.34	51.23	57.11
Vehicle	71.13	77.78	79.75	75.27	78.63
Waveform	67.81	76.46	78.98	78.8	78.01
Wine	89.85	87.94	88.39	86.31	88.59

The presented results comprise four separate rounds of experiments with different amount of noise (γ)

Table 7 The efficiency of rival algorithms regarding *run time* (in seconds)

Data stream	Run time (s)				
	CVFDT	DCS	FlexDT	SEA	WCE
Adult	2.91	25.14	4.05	17.11	13.16
Annealing	0.75	21.02	1.78	7.33	6.62
AustralianSign	1.12	11.10	4.25	6.78	7.52
Balance	2.74	5.05	5.24	6.02	5.32
BreastCancerWisconsin	1.96	17.95	2.18	12.26	9.43
CreditScreening	2.27	17.12	3.21	11.54	8.88
Diabetes	0.81	9.42	1.78	6.51	5.01
German	1.72	14.06	8.45	9.35	7.69
Glass	1.32	15.67	5.98	10.84	8.34
HeartDisease	3.67	22.43	4.52	14.68	11.29
Hepatitis	1.89	12.02	2.64	8.00	6.15
HorseColic	1.23	15.27	5.28	10.23	8.85
Hyperplane	0.67	3.33	3.73	3.47	2.67
Ionosphere	4.25	73.24	6.29	22.57	17.36
Iris	0.92	2.08	3.56	2.91	2.24
LiverDisorders	1.74	8.24	3.17	9.65	7.42
SatelliteImage	3.73	87.47	14.12	33.11	25.47
SEA	1.56	3.24	3.56	6.55	5.04

Table 7 continued

Data stream	Run time (s)				
	CVFDT	DCS	FlexDT	SEA	WCE
Sonar	5.69	80.58	15.12	53.35	48.77
Thyroid	1.61	28.65	3.38	13.59	10.45
USPSHandwritten	8.22	1143.34	45.29	179.08	137.75
Vehicle	2.53	20.51	3.31	13.90	10.69
Waveform	3.52	76.56	8.69	25.13	19.33
Wine	2.13	18.86	6.24	12.56	10.46

Table 8 The efficiency of rival algorithms regarding *Convergence time* (in miliseconds)

Data stream	Convergence time (ms)				
	CVFDT	DCS	FlexDT	SEA	WCE
Adult	54.05	157.26	45.17	106.80	65.06
Annealing	15.63	93.24	11.93	67.25	36.05
AustralianSign	35.93	83.57	21.31	73.12	39.08
Balance	47.25	45.37	48.62	50.46	35.56
BreastCancerWisconsin	27.13	97.11	35.79	102.08	53.55
CreditScreening	35.56	130.25	33.14	120.38	79.75
Diabetes	19.09	102.34	16.97	75.83	50.10
German	33.64	160.74	77.08	124.38	41.65
Glass	50.40	87.20	24.76	57.71	49.24
HeartDisease	43.66	141.93	58.28	105.43	99.83
Hepatitis	36.67	110.60	40.59	87.97	51.75
HorseColic	43.33	136.21	20.21	160.59	94.47
Hyperplane	24.73	39.37	11.55	44.07	25.92
Ionosphere	73.54	393.10	57.28	189.32	167.48
Iris	23.54	18.63	19.86	21.76	16.03
LiverDisorders	32.02	112.67	42.01	126.25	74.26
SatelliteImage	124.63	503.26	64.31	350.23	233.12
SEA	39.96	38.43	36.90	89.98	52.93
Sonar	158.99	610.32	98.10	280.93	224.50
Thyroid	34.14	122.34	47.76	97.74	57.50
USPSHandwritten	237.27	1261.15	141.72	537.84	316.38
Vehicle	43.43	286.34	42.62	207.04	121.79
Waveform	87.78	368.68	55.69	289.59	170.35
Wine	46.79	169.06	31.72	125.33	113.94

References

Basak J (2006) Online adaptive decision trees: pattern classification and function approximation. Neural Comput 18(9):2062–2101

Bhatt RB, Gopal M (2006) Neuro-fuzzy decision trees. Int J Neural Syst 16(1):63–78

Chan P, Dunn OJ (1972) The treatment of missing values in discriminant analysis. J Am Stat Assoc (6):473–477

Cohen W (1995) Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning (ICML), pp 115–123

Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Domingos P, Hulten G (2000) Mining high speed data streams. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 71–80

Fan W (2004) Systematic data selection to mine concept-drifting data streams. In: Proceedings of the 10th ACM international conference on knowledge discovery and data mining (SIGKDD), pp 128–137

Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th international joint conference of artificial intelligence, pp 1022–1027

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32:675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11:86–92

Hashemi S, Yang Y, Pourkashani M, Kangavari M (2007) To better handle concept change and noise: a cellular automata approach to data stream classification. In: Australian joint conference on artificial intelligence, pp 669–674

Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA

Ho SS (2005) A martingale framework for concept change detection in time-varying data streams. In: Proceedings of the 22nd international conference on machine learning (ICML), pp 321–327

Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 97–106

Jang J-SR (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23:665–684

Janikow CZ (1998) Fuzzy decision trees: issues and methods. IEEE Trans Syst Man Cybern B Cybern 28(1):1–14

Janikow CZ, Kawa K (2005) Fuzzy decision tree fid. In: Annual meeting of the north American fuzzy information processing society, IEEE, pp 379–384

Kolter JZ, Maloof MA (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the 3rd IEEE international conference on data mining (ICDM), p 123

Maher PE, Clair DCS (1993) Uncertain reasoning in an id3 machine learning framework. In: 2nd IEEE international conference on fuzzy systems, pp 7–12

Mitchell TM (1997) Machine learning. McGraw Hill

Mitra S, Konwar KM, Pal SK (2002) Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: generation and evaluation. IEEE Trans Syst Man Cybern C Appl Rev 32(4):328–339

Mundfrom DJ, Whitcomb A (1998) Imputing missing values: The effect on the accuracy of classification. Multiple Linear Regre Viewp 25(1):13–19

Newman DJ, Hettich S, Blake C, Merz C (1998) UCI repository of machine learning databases

Olaru C, Wehenkel L (2003) A complete fuzzy decision tree technique. Fuzzy Sets Syst 138:221–254

Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers

Quinlan JR (1993) Induction of decision trees, pp 349–361

Saar-Tsechansky M, Provost F (2007) Handling missing values when applying classification models. J Mach Learn Res 8:1625–1657

Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 377–382

Tsymbal A (2004) The problem of concept drift: definitions and related work. Technical report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Ireland

Umanol M, Okamoto H, Hatono I, Tamura H, Kawachi F, Umedzu S, Kinoshita J (1994) Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In: IEEE world congress on computational intelligence, pp 2113–2118

Wang H, Fan W, Yu PS, Han J (2003) Mining concept drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 226–235

Wang P, Wang H, Wu X, Wang W, Shi B (2005) On reducing classifier granularity in mining concept-drifting data streams. In: Proceedings of the 5th IEEE international conference on data mining (ICDM), pp 474–481

Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Machine Learn 23:69–101

Yang Y, Wu X, Zhu X (2005) Combining proactive and reactive predictions for data streams. In: Proceedings of the 11th ACM international conference on knowledge discovery and data mining (SIGKDD), ACM Press, pp 710–715

Yang Y, Wu X, Zhu X (2006) Mining in anticipation for concept change: proactive-reactive prediction in data streams. Data Mining Knowl Discov 13(3):261–289

Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study. Artif Intell Rev 22(3):177–210

Zhu X, Wu X, Chen Q (2003) Eliminating class noise in large datasets. In: Proceedings of the 20th internationl conference in machine learning (ICML), pp 920–927

Zhu X, Wu X, Yang Y (2004) Dynamic classifier selection for effective mining from noisy data streams. In: Proceedings of the 4th IEEE international conference on data mining (ICDM), pp 305–312

Zhu X, Wu X, Yang Y (2006) Effective classification of noisy data streams with attribute-oriented dynamic classifier selection. Knowl Inf Syst 9(3):339–363

Zimmermann HJ (2001) Fuzzy set theory and its applications. Kluwer Academic Publishers, Dordrecht, Boston