Abstract In streaming data analysis where underlying data distribution may be changed or the concept of interest can drift with the progress of time, the ability to adapt to concept drift can be very powerful especially in the process of incremental learning. In this paper, we develop a general framework for an adaptive incremental classifier on data stream with concept drift. A distribution, representing the performance pattern of a classifier, is constructed by utilizing the distance between the confidence score of a classifier and a class indicator vector. A hypothesis test is then performed for concept drift detection. Based on the estimated p-value, the weight of outdated data is set automatically in updating the classifier. We apply our proposed method for two types of linear discriminant classifiers. The experimental results on streaming data with concept drift demonstrate that the proposed adaptive incremental learning method improves the prediction accuracy of an incremental classifier highly.

Keywords: adaptive incremental classifier, concept drift, streaming data, linear discriminant analysis

1. Introduction

A data stream is a sequence of data samples which is continuously generated as time goes on. In streaming data analysis, not all the data samples can be stored and it is not expected that a batch of data samples will be processed several times. Moreover, underlying data distribution may be changed or the concept of interest can drift from one to another as the data are streaming. Under the assumption that the true class label of a data sample is revealed immediately after the prediction is made, classification algorithms on concept-drifting streaming data...
are divided into two categories: the first uses one classifier and updates it whenever a new data sample or a chunk of data samples arrive [1-3]. The other uses the ensemble of classifiers so that a new member of the ensemble can reflect recent data distribution better than old members [4-7].

Recently, incremental algorithms of linear discriminant analysis (LDA), which is a well-known dimension reduction method, were proposed [8-11]. In particular, a rank-one update method of least squares LDA (linear discriminant analysis) adapts itself to concept drift by decreasing the dependency on outdated data, through a forgetting factor in the learning process [11]. However, a forgetting factor which is fixed in advance cannot be suitably applied to both cases: when a change of data distribution occurs, or when a change does not occur. In [12], an online linear discriminant classifier with adaptive learning rate was proposed. It modifies the learning rate according to the change of classification error rate which is estimated by the difference of error rates in two adjacent data blocks. Since the learning rate is updated whenever the current data point is misclassified, it is not easy to control the learning rate optimally.

In this paper, we propose an adaptive incremental classifier for data stream with concept drift. A distribution representing the performance of a classifier is constructed by utilizing the distance between the confidence score of a classifier and a class indicator vector, and hypothesis testing for concept drift detection is performed. Using the estimated p-value we set the weight of outdated data in updating the classifier. We apply the proposed method for two types of a linear discriminant classifier in order to demonstrate the performance: a linear classifier by minimum squared error and a linear discriminant classifier using common class covariance matrix.

The rest of the paper is organized as follows. In Section 2, a brief review of several learning algorithms on streaming data is given. In Section 3, our approach for an adaptive incremental classifier is presented. Experimental results comparing the proposed method and other methods are given in Section 4 and discussions follow in Section 5.

2. Related Work

Updating a decision tree incrementally as data samples arrive is one of the well-known methods for classification on streaming data [1,13]. It stores statistical information on nodes and grows the tree by splitting a node as necessary. In order to confront the concept drift problem, many variants of incremental decision tree have been proposed [2,3].

Using an ensemble of classifiers has been shown to be very powerful, especially when a base classifier is weak and unstable [14]. In a concept-drifting data stream, a new member of the ensemble family is built on a chunk of recent data samples and an outdated member is removed. By assigning weights to ensemble members depending on the estimated error rate, concept drift can be dealt with [4]. Various approaches for improved performance include adaptive window size [5], dynamic ensemble construction [6], combination with concept drift detection [7].

While traditional classification algorithms such as a nearest neighbor classifier and support vector machines are also used on a concept-drifting data stream [15,16], a linear function is attractive to use due to its simplicity and intuitiveness. A linear classifier is also competent compared with more complicated classification models. In [12], error rate is used for adaptive learning rate for an online linear discriminant classifier. Error rate increased largely may indicate that a special event such as concept drift or a change in data distribution happens. When the learning rate $\lambda (0 < \lambda < 1)$ accounts for the weight of a new data point and $1 - \lambda$ is for old ones, large increase in error rates changes the learning rate in direction of forgetting old data by the updating formula $\lambda \leftarrow \lambda (1 + \delta)\Delta$ where $\delta$ is the difference of error rates in two adjacent windows. In [11], incremental updating of least squares linear discriminant analysis (LSLDA) adopted a forgetting factor which decreased dependency on the outdated data in the updating process. However, a fixed forgetting factor is not suitable when concept drift does not occur, since it cannot fully utilize information from past data samples. It is also difficult to choose an optimal forgetting factor in advance. Moreover, LSLDA is a dimension reduction method rather than a classifier,
therefore a nearest neighbor classifier was used after
dimension reduction in [11]. Motivated by incre-
mental updating with a forgetting factor, in this paper
we develop a general framework for an adaptive
incremental classifier on data stream with concept drift.

3. An Adaptive Incremental Linear Classifier
for Data Stream with Concept Drift

Error rate is often used for concept drift detection
on streaming data[17-19]. An error value for each
data sample is a binary value, which is set as 0 if
the prediction is correct and 1 if the prediction is
wrong. The error can be a random variable of Ber-
noulli distribution, and a significant increase in error
rate is monitored to warn and signal about the event
of concept drift[17,18]. It is based on the assumption
that if the distribution of the data samples is sta-
tionary, the error rate of the classifier will decrease
when the number of samples increases. However,
describing the prediction results with only binary
values can cause useful information about classifier
behavior to be lost. Instead of error rates, we pro-
pose a method to monitor the classifier behavior by
using confidence scores given by the classifier.

Let $f$ be a classifier model and $C_i(x)$, $i=1,...,r$, be confidence scores or probabilities for assigning a
data sample $x$ to the class $i$ by the classifier $f$. An
ideal case for the normalized confidence scores is

$$C_i(x) = \begin{cases} 1 & \text{if the data sample } x \text{ belongs to class } i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Let $y \in \mathbb{R}^{1 \times r}$ be a class indicator vector for a data
sample $x$ belonging to the class $i$, where the $i$-th com-
ponent of $y$ is 1 and other components are 0. Con-
sider a random variable $X$ defined by

$$X(x) = \| C(x) - y \|_2^2 \quad (2)$$

for a data sample $x$, where $C(x) = [C_1(x),...,C_r(x)]$.

Our proposed method is based on the supposition
that a significant change in $X(x)$ can imply the change
in classifier behavior by concept drift. If the value $X(x)$
by the next incoming data sample deviates significantly from the distribution of $X$, it may indicate the occurrence of concept drift and therefore
the weight of recently received data sample in the
incremental classifier updating needs to be increased.
However, it is difficult to estimate the distribution of
$X$ in a data streaming environment. In order to
circumvent the problem, we simply assume the dis-
tribution of $X$ as normal distribution. When $x_{n+1}$ is
a just incoming data sample, we need to measure the
degree of concept drift implied by $x_{n+1}$. In order to
estimate mean $\mu$ and variance $\sigma^2$ of $X$, we use the
values $X(x_{n-t+1}),...,X(x_n)$ by recently received data
samples. Now hypothesis testing is designed for
concept drift detection. The $p$-value for a new data
sample $x_{n+1}$ is computed by

$$p\text{-value} = P(Z > \frac{X(x_{n+1}) - \mu}{\sigma}) \quad (3)$$

for a standard normal distribution $Z$. If the $p$-
value is smaller than a significance level, it indicates a
change in data distribution and the weight on an old
classifier model is adjusted based on $p$-value in the
updating process of a classifier.

The function which maps $p$-value to the weight $\beta$-
value needs to be an increasing function, since a small $p$-
value under the significance level requires the effect of past data samples to be reduced. The
value $\beta$ is used as the weight on the current model
$f_{cur}$ in an updating formula $f_{\text{new}} \leftarrow \text{Update}(\beta, f_{cur}, x_{n+1})$.

In our experiments, we used the function shown in
Fig.1 with a significance level $\alpha = 0.05$. When $p$
-value by $X(x_{n+1})$ is smaller than 0.05, $\beta$-value is
set as a value between $s$ and 1. Otherwise, no
concept drift occurs and $\beta$ is set as 1. Instead of the
function in Fig.1 any function which reflects a user’s
prior knowledge can be used. Maybe it would be
better to say that the $p$-value below the significance
level gives a warning for concept drift, rather than
rather than concept drift detection, since it does not cause
learning of a new classifier but only reduces the
effect of past data samples in updating a current
classifier. Whenever a new data sample $x_{n+1}$ arrives,
the list $(X(x_{n-t+1}),...,X(x_n))$ is updated by adding
$X(x_{n+1})$ and removing the oldest element $X(x_{n-t+1})$.
In order to recompute $\mu$ and $\sigma$, only $\sum X(x_k)$ and
$\sum X(x_k)^2$ need to be kept updated. The proposed
approach is summarized in Algorithm 1.

In the next two subsections, we apply our proposed
method for linear discriminant classifiers of two
types : a linear classifier by minimum squared error
and a discriminant classifier with common class covariance.

Algorithm 1. General framework for adaptive incremental learning

Input:

\[ F = \{ X(x_{n-t+1}), \ldots, X(x_n) \} : \text{records by the recently received data samples} \]
\[ x_{n-1}, x_{n-2}, x_{n-3}, \ldots : \text{incoming data stream} \]

Algorithm:

1. Let \( k = n+1 \).
2. While (incoming data sample \( x_k \) is available)
3. Compute \( X(x_k) = \| C(x_k) - y_k \|_2^2 \)
4. Let \( \mu \) and \( \sigma \) be the mean and standard deviation of \( F \)
5. Compute p-value \( = P(Z > \frac{X(x_k) - \mu}{\sigma}) \)
6. Compute \( \beta \) using the function given in Fig.1
7. Update the classifier by the updating formula \( f_{n+1} \leftarrow \text{Update}(\beta, f_{n+1} \cdot x_k) \)
8. Remove the oldest element and add \( X(x_k) \) to \( F \).
9. \( k = k + 1 \).
10. end of while

3.1 A Linear Classifier by Minimum Squared Error

A linear function represents a separating hyper-plane between two classes[20]. For a multi-class problem where a data sample belongs to one of the classes \( 1, \ldots, r \), a classifier composed of a set of discriminant functions \( \{ g_i(x), i = 1, \ldots, r \} \) assigns a data sample \( x \) to the class \( j \) corresponding to the largest discriminant value such that

\[ g_j(x) > g_i(x) \quad \text{for all} \quad i \neq j. \]  (4)

In many cases, the discriminant values \( \{ g_i(x) \} \) \( 1 \leq i \leq r \) after suitable normalization, can be considered to be confidence scores for class prediction. Higher discriminant value means high confidence for the prediction.

Assume that a set of data samples \( \{ x_1, \ldots, x_n \} \) is given, where \( x_i \in \mathbb{R}^{d \times 1} \) denotes a data sample in a \( d \) -dimensional space belonging to one of classes \( 1, \ldots, r \). A set of discriminant functions \( \{ g_i(x) = w_i^T x | 1 \leq i \leq r \} \) is to be found[20], which is a solution to the problem

\[ g_i(x) = w_i^T x = \begin{cases} 1 & \text{if} \ x \in \text{class } i \\ 0 & \text{otherwise} \end{cases} \]  (5)

By denoting an augmented feature vector with 1 such as \( \tilde{x} = [1 \ x^T]^T \) and an augmented vector \( \tilde{w} = [w_1^T \ w_r^T]^T \), let us rewrite \( g_i(x) = w_i^T \tilde{x} \). Discriminant functions are found by minimizing the squared error which is expressed using the Frobenius norm as

\[ \sum_{k=1}^{n} \| g(x_k) - y_k \|_2^2, \]  (6)

where \( g(x) = [g_1(x) \cdots g_r(x)] \) and \( y_k \) is a class indicator vector for a data sample \( x_k \). Denoting

\[ A = \begin{bmatrix} \tilde{w}_1^T \\ \vdots \\ \tilde{w}_r^T \end{bmatrix}, \quad W = \begin{bmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_n^T \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \]

a solution which minimizes the squared error (6) can be computed by using the pseudo-inverse \( A^+ \) of \( A \) such as

\[ W = A^+ Y = (A^T A)^{-1} A^T Y \]  (7)

If the matrices \( W, A, Y \) and \( M = A^T A \) for data samples \( \{ x_1, \ldots, x_n \} \) are denoted with a subscript \( n \), Eq. (7) becomes

\[ W_n = (A_n^T A_n)^{-1} A_n^T Y_n = M_n^{-1} A_n^T Y_n \]  (8)

When a new data sample \( x_{n+1} \) is received, we need to compute \( W_{n+1} \) by using

\[ A_{n+1} = \begin{bmatrix} A_n \\ x_{n+1} \end{bmatrix} \quad \text{and} \quad Y_{n+1} = \begin{bmatrix} Y_n \\ y_{n+1} \end{bmatrix}. \]  (9)

Now we have

\[ W_{n+1} = M_n^{-1} A_n^T Y_n = M_n^{-1} (A_n Y_n + x_{n+1} y_{n+1}) \]  (10)

\[ = M_n^{-1} (A_n Y_n + x_{n+1} y_{n+1}) \]  (11)

\[ = M_n^{-1} (M_n W_n + x_{n+1} y_{n+1}) \]  (12)

\[ = M_n^{-1} (M_n x_{n+1}^T W_n + x_{n+1} y_{n+1}) \]  (13)

\[ = W_n + M_n^{-1} x_{n+1}^T (y_{n+1} + x_{n+1} W_n) \]  (14)

\[ M_n^{-1} \] can be computed from \( M_n^{-1} \) by applying the Woodbury matrix identity[21] for \( M_n^{-1} = (M_n + x_{n+1} x_{n+1}^T)^{-1} \).
as in [11]. The Woodbury matrix identity is
\[(B+UDV)^{-1} = B^{-1} - B^{-1}U(D^{-1}+VB^{-1}U)^{-1}VB^{-1},\]
where B, U, D and V all denote matrices of the correct size. Hence we have
\[M_{n+1} = (M_n + \hat{x}_{n+1}x_{n+1}^T)^{-1} = M_n^{-1} - \frac{M_n^{-1}\hat{x}_{n+1}x_{n+1}^T}{I + \hat{x}_{n+1}^TM_n^{-1}\hat{x}_{n+1}}M_n^{-1},\]
where \(B, U, D\) and \(V\) all denote matrices of the

Under the change in data distribution, reducing the effect of past data samples on updating a classifier is realized by the introduction of a weight factor \(\beta\) as in [11] such as
\[M_{n+1} = \beta M_n + \hat{x}_{n+1}\hat{x}_{n+1}^T\]
and
\[A_n^T Y_n = \beta A_n^T Y_n + \hat{x}_{n+1}y_{n+1}^T,\]
\(\beta=1\) means no concept drift and low \(\beta\) value reflects concept drift. By the introduction of the weight \(\beta\), Eq.(10) is changed as
\[W_{n+1} = M_{n+1}^{-1}(\beta A_n^T Y_n + x_{n+1}y_{n+1}) = M_{n+1}^{-1}(\beta M_n W_n + x_{n+1}y_{n+1}) = M_{n+1}^{-1}((M_{n+1}^{-1}x_{n+1}^T)W_n + x_{n+1}y_{n+1}) = W_n + M_{n+1}^{-1}x_{n+1}y_{n+1} - x_{n+1}^TW_n,\]
where \(M_{n+1}^{-1}\) is obtained from \(M_n^{-1}\) such as
\[M_{n+1}^{-1} = (\beta M_n + x_{n+1}x_{n+1}^T)^{-1} = \frac{1}{\beta}M_n^{-1} - \frac{1}{\beta}M_n^{-1}\hat{x}_{n+1}x_{n+1}^T\frac{1}{I + \hat{x}_{n+1}^TM_n^{-1}\hat{x}_{n+1}}M_n^{-1}.\]

Class prediction of a new data sample \(x\) is performed by assigning it to the class \(j\), when the \(j\)-th component in \(W_{n+1}\) is the maximum value. However, fixing the value \(\beta\) in advance cannot fit both situations of concept drift and no concept drift. Hence, we propose to adjust the value of \(\beta\) adaptively according to the deviation of a current data sample from the distribution of past data samples.

Now we define confidence values as discriminant values such as
\[C_i(x) = g_i(x), i = 1, \cdots, r.\]

Due to the minimization of Eq. (6), \(g_i(x)\) is naturally scaled around the range from 0 to 1. Consider \(X(x_i) = g(x_i) - y_i\). The \(p\)-value by \(X(x_{n+1})\) for a new data sample \(x_{n+1}\) gives the \(\beta\)-value for the updating process of \(W_{n+1}\) in Eqs. (11) and (12). We denote the proposed adaptive incremental linear classifier by minimum squared error as ADIMSE.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>The components for ADIMSE required in Algorithm 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a classifier : (f(x) = W_n^T\hat{x}), where (\hat{x} = [1 \ x^T]^T), (M_n = A_n^TA_n) and (W_n = M_n^{-1}A_n^TY_n), a confidence score vector : (C(x) = W_n^T\hat{x}) an updating formula of a classifier (f_{new} \leftarrow Update(\beta, f_{cur}, x)) :</td>
<td></td>
</tr>
<tr>
<td>(M_{n+1} = \frac{1}{\beta}M_n^{-1} - \frac{1}{\beta}M_n^{-1}\hat{x}<em>{n+1}x</em>{n+1}^T\frac{1}{I + \hat{x}<em>{n+1}^TM_n^{-1}\hat{x}</em>{n+1}}M_n^{-1})</td>
<td>(W_{n+1} = W_n + M_{n+1}^{-1}\hat{x}<em>{n+1}(y</em>{n+1} - x_{n+1}^TW_n))</td>
</tr>
</tbody>
</table>

When \(\beta\) is fixed as 1, we call it as IMSE. Table 1 summarizes the components for ADIMSE which are needed in Algorithm 1.

3.2 A Discriminant Classifier with Common Class Covariance

Assuming that conditional density \(p(x|i)\) has normal distribution, Bayes classifier gives a discriminant function
\[g_i(x) = \ln(P(i)p(x|i)) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) \tag{14}\]
\[-\frac{d}{2}\ln(2\pi) - \frac{d}{2}\ln|\Sigma| + \ln P(i).\]

\(P(i)\) is the prior probability of class \(i\), \(d\) is a data dimension, and \(\mu_i\) and \(\Sigma_i\) are the mean and covariance of class \(i\) respectively[20]. When the covariance matrices for all of the classes are assumed to be identical to the within-class scatter matrix
\[\Sigma = \frac{1}{n} \sum_{i=1}^{r}n_i\Sigma_i = \frac{1}{n} \sum_{i=1}^{r} \sum_{x \in class_i} (x-\mu_i)(x-\mu_i)^T \tag{15}\]
it terms independent of \(i\) are ignored, the discriminant function in Eq. (14) is simplified to
\[g_i(x) = \mu_i^T\Sigma^{-1}x - \frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \ln P(i). \tag{16}\]

In [12], incremental updating of a linear discriminant function in (16) was proposed by using an adaptive learning rate \(\lambda\). Instead, we derive incremental updating by the weight \(\beta\) for the outdated data. Update formulas of the means, prior probability and covariance matrix for a new data sample \(x_{n+1}\) belonging to the class \(i\) are given as
\[\mu_i \leftarrow \beta\mu_i + x_{n+1}/(\beta n_i + 1) \tag{17}\]
\[P(i) \leftarrow \beta P(i)/\beta n_i + 1, \quad P(j) \leftarrow \beta P(j)/\beta n_i + 1, \quad j \neq i \]
\[\Sigma^{-1} \leftarrow \frac{\beta n_i + 1}{\beta n_i} \Sigma^{-1} - \frac{1}{\beta n_i + 1} \Sigma^{-1}xx^T\Sigma^{-1}x^T.\]
where \( z = x_{n+1} - \mu_i \) and \( n_j \) denotes the number of data samples in the class \( j \).

In this classifier, we use the posterior probability \( p(i|x) = p(x|i)p(i)/p(x) \) as the confidence score \( C_i(x) \) for the prediction to the class \( i \) of the data sample \( x \). By using the covariance \( \Sigma \) and ignoring terms which are independent of \( I \), \( C_i(x) \) is simplified as

\[
C_i(x) \sim P(i) \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) \right).
\]

After the normalization process by

\[
C_i(x) \leftarrow C_i(x)/\sum_j C_j(x),
\]

we consider the distribution of \( X(x) = \| C(x) - y_k \|^2 \).

As data samples arrive continuously, the sample size \( n \) gets bigger and bigger and the weight multiplied by \( n \) does not make a role in reducing the effect of outdated data. In order to maintain effective sample size, we update the sample size \( n \) and \( n_j \) recursively such that

\[ n_i \leftarrow \beta n_i + 1, \quad n_j \leftarrow \beta n_j \quad \text{for} \quad j \neq i, \quad \text{and} \quad n \leftarrow \beta n + 1, \]

when a data sample \( x \) belongs to the class \( i \). Let us denote this method as ADILDA (adaptive incremental linear discriminant analysis). When \( \beta \) is fixed as 1, we call it as ILDA (incremental linear discriminant analysis).

4. Experimental Results

For the comparative evaluation for the proposed method, we used three artificial data sets and two real data sets. Three artificial data sets were generated by using MOA data stream software[22], which is an open source software framework in Java designed for online settings as data streams. We used three data generators which have been most commonly used in the literature.

- **Random RBF Generator** : This generator creates normally distributed hyperspheres of data samples surrounding each central point with varying densities. Each center has a random position, a single standard deviation, a class label and weight. Drift is introduced by moving the centers with constant speed. We generated two data sets by setting the number of classes (and centroids) as 10 and 50 respectively. Drift speed was set as 0.001 for both data sets. Other parameters were set as default.

- **SEA Concepts Generator** : This data set is generated using three attributes, where only two attributes are relevant. Different concepts are defined by using \( f_1 + f_2 \leq \theta \), where \( f_1 \) and \( f_2 \) are two relevant attributes and \( \theta \) is a threshold value. Concept drift happens at the interval of 100,000 data samples. The default values were used for other parameters.

- **LED Generator** : The goal of this data is to predict the digit displayed on a seven segment LED display with drift. Concept drift happens at the interval of 100,000 data samples. The default values were used for other parameters.

Two real data sets used in our experiments are Electricity[23] and Forest Covertype[24]. Electricity data describes changes of electricity prices based on the electricity market in the Australian state of New South Wales. It consists of 45,312 instances with 8 attributes and the class label “UP” and “DOWN” identify the change of the price related to a moving average of the last 24 hours. We used the data Elec2-3 which has five attributes, Day of week, Time, NSW electricity demand, Victorian electricity demand, and Scheduled interstate electricity transfer. After excluding data instances with missing values, 27,549 instances were left. Forest covertype data contains the forest cover type for 30x30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The goal is to predict the forest cover type from cartographic variables. It contains 581,012 instances and 54 attributes (10 numeric attributes + 44 nominal attributes) and we used 10 numeric attributes.

The proposed methods ADIMSE and ADILDA were compared with the well-known classifiers HAT, and DDM+NB which were implemented in MOA. The compared methods are as follows:

- **Hoeffding Adaptive Tree (HAT)** [25]: It uses ADWIN to monitor performance of branches on the tree and to replace them with new branches as needed.

- **DDM+NB** [17] : The drift detection method DDM proposed in [17] is used with Naive Bayes classifier.

- **IMSE** : Incremental linear classifier by Minimum Squared Error.

- **ILDA** : Incremental discriminant classifier with common class covariance.
Table 2 The accuracy (%) of the compared methods

<table>
<thead>
<tr>
<th></th>
<th>RBF(10)</th>
<th>RBF(50)</th>
<th>SEA</th>
<th>LED</th>
<th>Electricity</th>
<th>Covertype</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAT</td>
<td>55.35</td>
<td>14.46</td>
<td>88.33</td>
<td>73.20</td>
<td>70.0</td>
<td>82.63</td>
</tr>
<tr>
<td>DDM+NB</td>
<td>85.59</td>
<td>64.43</td>
<td>85.91</td>
<td>68.63</td>
<td>71.26</td>
<td>81.32</td>
</tr>
<tr>
<td>IMSE</td>
<td>31.74</td>
<td>10.43</td>
<td>84.75</td>
<td>29.32</td>
<td>64.61</td>
<td>63.12</td>
</tr>
<tr>
<td>ILDA</td>
<td>31.87</td>
<td>10.40</td>
<td>84.78</td>
<td>29.96</td>
<td>64.57</td>
<td>67.66</td>
</tr>
<tr>
<td>ADIMSE</td>
<td>84.21</td>
<td>52.0</td>
<td>87.15</td>
<td>69.82</td>
<td>74.68</td>
<td>83.43</td>
</tr>
<tr>
<td>ADILDA</td>
<td>91.33</td>
<td>71.94</td>
<td>86.86</td>
<td>72.74</td>
<td>75.49</td>
<td>85.96</td>
</tr>
</tbody>
</table>

- ADIMSE: The proposed adaptive incremental linear classifier by Minimum Squared Error.
- ADILDA: The proposed adaptive incremental discriminant classifier with common class covariance.

The evaluation methodology used in our experiments is the Interleaved Test-and-Train approach. Each data sample is first used for testing, and then it is used to train the model. The accuracy was measured as the final percentage of data samples which were correctly classified over the interleaved evaluation. For each artificial data set 1,000,000 data samples were generated and it was repeated 50 times. The average accuracies achieved from 50 runs are presented in Table 2. ADIMSE and ADILDA were tested with $s=0.9$, where $s$ is the parameter for the function shown in Fig. 1. For real data sets, the final accuracy by the interleaved Test-and-Train approach is reported.

Table 2 summarizes the experimental results for both artificial and real data sets. Since decision tree, naive bayesian and linear discriminant analysis are classifiers which have different properties, it is difficult to compare the performance of them directly. But, there are several noticeable observations. Overall, adaptive versions ADIMSE and ADILDA improve the performance of incremental linear classifiers IMSE and ILDA, due to the use of p-value implying occurrence of concept drift. The gaps between them are big on all the data sets except SEA data. HAT and DDM+NB show poor performance on two data sets by the Random RBF generator which represent gradual drift. On the other hand, HAT gave good performance on SEA and LED data where drift occurred abruptly every interval of 100,000 data samples.

5. Discussions

We proposed a method to learn a learning rate adaptively for the incremental learning of a linear classifier. The proposed method can adjust a classifier quickly to the new trend, when the concept changes on the streaming data. This is done by monitoring the performance of a classifier by means of the distribution of the distance between a confidence vector and class indicator vector of past data samples. Concept drift is indicated by a p-value below a significance level. The learning rate is changed according to the p-value. The proposed method in Fig. 1 can be applied for any classification algorithm if it gives confidence score or probability values as an output and an update formula $f_{new} \leftarrow U(\beta, f_{old}, x)$ for a current classification model $f_{old}$ can be devised.

Experimental results demonstrated the competent performance of the proposed method.

References