Evolving Systems (2013) 4:119–131
DOI 10.1007/s12530-013-9076-7

ORIGINAL PAPER

Recursive least square perceptron model for non-stationary and imbalanced data stream classification

Adel Ghazikhani · Reza Monsefi · Hadi Sadoghi Yazdi

Received: 20 September 2012 / Accepted: 23 January 2013 / Published online: 6 February 2013

Abstract Classifying non-stationary and imbalanced data streams encompasses two important challenges, namely concept drift and class imbalance. "Concept drift" (or non-stationarity) is changes in the underlying function being learnt, and class imbalance is vast difference between the numbers of instances in different classes of data. Class imbalance is an obstacle for the efficiency of most classifiers and is usually observed in two-class datasets. Previous methods for classifying non-stationary and imbalanced data streams mainly focus on batch solutions, in which the classification model is trained using a chunk of data. Here, we propose an online perceptron model. The main contribution is a new error model inspired from the error model of recursive least square (RLS) filter. In the proposed error model, non-stationarity is handled with the forgetting factor of RLS error model and for handling class imbalance two different errors weighting strategies are proposed. These strategies are verified with convergence and tracking theories from adaptive filters theory. The proposed methods is evaluated on two synthetic and six real-world two-class datasets and compared with seven previous online perceptron models. The results show statistically significant improvement to previous methods.

Keywords Data stream classification · Online perceptron · Concept drift · Imbalanced data · Recursive least square · Adaptive filter

A. Ghazikhani (✉) · R. Monsefi · H. Sadoghi Yazdi
Computer Engineering Department, Ferdowsi University of Mashhad (FUM), P.O. Box: 91775-1111, Mashhad, Iran
e-mail: a_ghazikhani@yahoo.com;
Adel.Ghazikhani@stu-mail.um.ac.ir

R. Monsefi
e-mail: monsefi@um.ac.ir

H. Sadoghi Yazdi
e-mail: sadoghi@um.ac.ir

H. Sadoghi Yazdi
Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad (FUM), P.O. Box: 91775-1111, Mashhad, Iran

1 Introduction

Data stream classification (DSC) is gaining increasing attention in pattern recognition and machine learning community. This is related to numerous applications such as credit card operations, weather, sensor networks, fraud detection, network attacks, web data, etc. The main issue in DSC is "concept drift", which is shifts in the underlying function generating the data (Gama 2010). In addition to concept drift, there are a number of other challenges reported in previous literature, such as concept evolution, feature evolution, limited labels and noise (Masud 2009).

Previous literature has reported three categories of methods for DSC that is: (1) instance selection, (2) instance weighting and (3) ensemble model. In the "instance selection" (or sliding window) category, the algorithms have two phases, which are drift detection and classification model retraining. The classification model is updated whenever drift is detected. These algorithms maintain a buffer of instances (usually recent instances) to retrain the classifier. Alippi et al. (2011) proposed an adaptive classification system which utilizes the intersection of confidence intervals rule (Goldenshluger and Nemirovski 1997) to detect drift and adapt the data window. Sun and Li (2011) utilized a sliding window scheme for financial distress prediction. The second category of methods is "instance weighting". In this category, an incrementing

weight is assigned to every incoming instance to make newer instances more effective in the classification model. Pavlidis et al. (2011) proposed an adaptive forgetting factor, based on the recursive least square adaptive filter, for instance weighting. Martinez-Rego et al. (2012) proposed a one-layer neural network that incorporates forgetting factor for handling drifts. The last category of methods is "ensemble models". Ensemble models are the most frequent used method for DSC in the literature. In batch ensemble models the data stream is divided into chunks, and a learner is built on each incoming chunk. The learners are weighted according to their error on the current chunk. Elwell and Polikar (2011) proposed a recent dynamic ensemble method that handles different kinds of concept drift. Abdulsalam et al. (2011) proposed an ensemble method that considers time constraints. Masud et al. (2011) proposed an ensemble model that handles concept drift and concept evolution concurrently.

Another challenge for DSC is class imbalance. In imbalanced data, the number of data in different classes differs vastly. In two-class imbalanced data, the class with less data is called the minority, and the other is called the majority class. Imbalanced data cause imbalanced classification results in different classes. Class imbalance has been studied in various paradigms thoroughly (He and Garcia 2009; Soda 2011; Tahir et al. 2012; Wang et al. 2012). Previous literature has made little attention to imbalanced data stream classification (IDSC). Algorithms for IDSC are composed of two main parts that are responsible for handling concept drift and class imbalance. Gao et al. (2007) utilized ensemble model for handling concept drift and oversampling for dealing with class imbalance. Chen and He (2010) use the same strategy as Gao with some improvements in both parts. Ditzler and Polikar (2010) proposed an algorithm that again utilizes an ensemble model for handling concept drift, but to handle class imbalance he proposed a modification in the ensemble learning scheme. Most of the previous methods on IDSC use batch solutions for training the classifiers. In these methods, the classifier is trained using a batch of data. Unlike batch processing, online processing has a per-instance view on the data stream (Tsymbal 2004).

Building on previous studies on IDSC, we propose an online perceptron classifier for non-stationary and imbalanced data streams. The proposed perceptron classifier utilizes the recursive least square (RLS) adaptive filter error model. The perceptron is trained with gradient descent method. The main contribution is a new error model suitable for classifying non-stationary and imbalanced data streams. The error model is composed of two parts for handling non-stationarity and class imbalance. Non-stationarity is handled with the inherent forgetting behavior of the RLS error model. An error weighting ability is embedded into the RLS error model to handle class imbalance for two-class datasets. This error weighting strategy is verified theoretically by the convergence and tracking theories from adaptive filters theory.

2 Background

2.1 Recursive least square adaptive filter

Adaptive filters are systems that are adapted based on input/output pairs of an unknown system which we tend to approximate its behavior (Haykin 2001). Recursive least square is a type of adaptive filter in which the parameters are adapted by minimizing the sum of the errors in the all the previous instants. The original RLS adaptive filter error model is shown below

$$E(\beta, t) = \sum_{i=1}^{t} \lambda^{t-i} \times e(\beta, i) \tag{1}$$

In Eq. (1), λ is the forgetting factor that is a constant in [0, 1], β is the parameters of the adaptive filter, and $e(\beta, i)$ is the error at instant i. The forgetting factor adjusts the overall error so that error of more recent instants is more important than previous ones.

2.2 Class imbalance learning

Class imbalance is vast difference between the number of instances in different classes and it is a challenge for most supervised classifiers (He and Garcia 2009). This problem is usually investigated in two-class datasets and causes weak learning in the minor class. There are two main strategies to strike this challenge, namely preprocessing

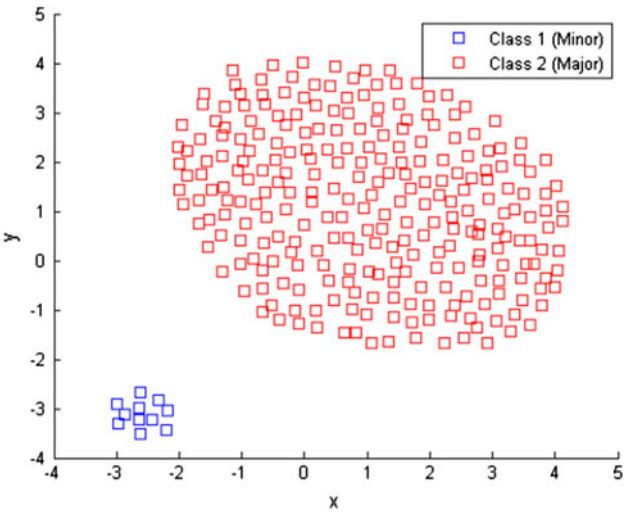

Fig. 1 Incremental learning curve for the SEA dataset

and classifier modification. Figure 1 shows a two-class imbalanced dataset.

3 The proposed method

An online perceptron classifier is proposed for IDSC. The proposed classifier borrows ideas from the RLS adaptive filter. The proposed error model of the perceptron is shown below

$$E(\beta,t)=\sum_{i=1}^{t} w(i)\times\lambda^{t-i}\times e(\beta,i) \tag{2}$$

$$e(\beta,t)=\frac{1}{2}\left(\mathrm{y(t)}-\varphi(\beta^{T}x)\right)^{2} \tag{3}$$

In Eq. (3) y is desired output and φ is activation function of the neuron. The main contribution is an error model that handles drift and imbalance concurrently. λ is the forgetting factor, that is used to handle concept drift. $w(i)$ is the error weight at instant i, that is responsible for handling imbalance. $w(i)$ should be adjusted such that the error of the class with minor data is more important than the major class. Strategies for adapting $w(i)$ are detailed in the next section.

The training procedure of perceptron classifier is online, i.e. the parameters of the perceptron are updated with each incoming instance. Here similar to (Pavlidis et al. 2011) gradient descent is utilized as the method for optimizing the perceptron parameters. In Eq. (4) gradient descent is written for adapting the parameters of the error function in Eq. (2)

$$\beta_j(t+1)=\beta_j(t)-\eta\frac{\partial E(\beta_j(t),t)}{\partial\beta_j(t)} \tag{4}$$

where η is a positive constant called the learning rate. This parameter influences the rate of convergence, i.e. higher values of this parameter causes faster learning in the algorithm; which is not always desired. Therefore, usually lower values are selected. The partial derivative in Eq. (4) could be written

$$\frac{\partial E(\beta_j(t),t)}{\partial\beta_j(t)}=\sum_{i=1}^{t} w(i)\times\lambda^{t-i}\times\frac{\partial e(\beta_j(t),t)}{\partial\beta_j(t)} \tag{5}$$

by eliminating the summation, Eq. (5) could be rewritten

$$E(\beta,t)=w(i)e(\beta,i)+\lambda E(\beta,t-1) \tag{6}$$

If we define

$$\frac{\partial E(\beta_j(t),t)}{\partial\beta_j(t)}=Gr_{\beta_j}(t)$$

by taking derivative from both sides of Eq. (6) we have

$$Gr_{\beta_j}(t)=w(t)\frac{\partial e(\beta(t),t)}{\partial\beta_j(t)}+\lambda Gr_{\beta_j}(t-1) \tag{7}$$

and substituting Eq. (7) in Eq. (4)

$$\beta_j(t+1)=\beta_j(t)-\eta Gr_{\beta_j}(\mathrm{t}) \tag{8}$$

Therefore the parameters of the perceptron model are adapted using Eq. (8). The gradient descent method claims that the obtained parameters converge to the optimal parameters for the perceptron error model.

3.1 Adapting error weight

Adaptation is a required action in non-stationary environments. One aspect of this adaptation is error weights. In Eq. (5), there are two types of error weights, namely λ and $w(i)$. λ is the forgetting factor, that gives more weight to error of recent instances. In (Pavlidis et al. 2011) a classifier is proposed in which λ is adapted. Here we tend to adapt $w(i)$, that is the error weight for handling class imbalance.

3.1.1 Classifiers and adaptive filters

It should be noted that a classifier has similar behavior as an adaptive filter. In both paradigms, a mapping is constructed between input and output data. Therefore concepts used in adaptive filter theory could be used for classifiers. The proposed strategy for adapting error weights for handling class imbalance is based on theoretical concepts in adaptive filters. In adaptive filter theory, filters are assessed by different criteria. Two common criteria are convergence and tracking. In the next two sections we define convergence and tracking and use these theories to analyze the proposed method.

3.1.2 Convergence analysis

In pattern recognition, convergence ensures that an iterative algorithm will eventually come to an optimum state (Duda et al. 2000). In adaptive filter theory, convergence analysis aims to analyze the convergence behavior of a filter. A classifier has a very similar role as an adaptive filter. Both of them approximate an unknown function. Therefore we can legally analyze the convergence of the online perceptron model in Eq. (2) with theories of adaptive filter. In adaptive filters, the RLS filter's convergence analysis is accomplished using the wiener filter (Haykin 2001). In the wiener filter, optimum parameters are

$$\beta_O=R^{-1}P \tag{9}$$

where $R=E[X^{T}X],\ P=E[YX]$

The wiener filter has optimum parameter values for mean square error. In Eq. (9) X is input data and Y is desired values of X.

In Eq. (2) the proposed online perceptron model is defined, in which β_j's are the perceptron parameters, which are to be analyzed. In other words, we want to analyze

$$\lim_{t\to\infty} \mathrm{E}[\beta(t)] \tag{10}$$

From Eqs. (7) and (8) we have

$$\beta_j(n+1) = \beta_j(n) - \eta G_{\beta_j}(n) \tag{11}$$

$$G_{\beta_j}(n) = w(n)e'(n) + \lambda G_{\beta_j}(n-1)$$

$$G_{\beta_j}(n) = \sum_{i=0}^{n} \lambda^{n-i} w(i) e'(i) \tag{12}$$

by substituting Eq. (12) in Eq. (11) we have

$$\beta_j(n+1) = \beta_j(n) - \eta \sum_{i=0}^{n} \lambda^{n-i} w(i) e'(i)$$

$$\beta_j(n+1) = -\eta \sum_{k=0}^{n} \sum_{i=0}^{k} \lambda^{k-i} w(i) e'(i)$$

by taking expectation from both sides

$$E\left[\beta_j(n+1)\right] = -\eta \sum_{k=0}^{n} \sum_{i=0}^{k} \lambda^{k-i} w(i) E[e'(i)] \tag{13}$$

if we assume a linear activation function we have

$$e'(i) = -x(i)(y(i) - \beta^T x(i)) \tag{14}$$

by substituting Eq. (14) in Eq. (13) we have

$$E[\beta_j(n+1)] = -\eta \sum_{k=0}^{n} \sum_{i=0}^{k} \lambda^{k-i} w(i) E[-x(i)(y(i) - \beta^T x(i))] \tag{15}$$

by assuming

$$\gamma_i = \sum_{k=0}^{n} \sum_{i=0}^{k} \lambda^{k-i} w(i) \tag{16}$$

by substituting Eq. (16) in Eq. (15) we have

$$E[\beta_j(n+1)] = -\eta\gamma_i E[-x(i)(y(i) - \beta^T x(i))] \tag{17}$$

by assuming $E[\beta_j(n+1)] = \beta(n)$ and doing some simplification

$$\beta_n = (I + \eta^{-1}\gamma_i^{-1} R^{-1})^{-1} R^{-1} P \tag{18}$$

From Eq. (18) it is obvious if $\eta^{-1}\gamma_i^{-1}R^{-1} < I$, the expectation of perceptron parameters converge to optimum wiener parameters. From this analysis it could be concluded that in Eq. (2), the strategy for adapting error weights should assign weights in an incrementing manner i.e. $w(i+1) \geq w(i)$. It should be noted, the same result could be achieved with sigmoid activation function and linearization with taylor series.

3.1.3 Tracking analysis

Non-stationary signals are signals with changing statistics. In adaptive filter, non-stationary behavior is evaluated with tracking analysis (Haykin 2001). In tracking analysis different criteria have been defined to asses tracking ability. One of these criteria is Mean Square Deviation (MSD), which is the mean square deviation between the optimum parameter vector ($\beta_o(n)$) and the learned parameter vector ($\widehat{\beta}(n)$) of the filter

$$MSD = E\left[\left\|\widehat{\beta}(n) - \beta_o(n)\right\|^2\right] \tag{19}$$

In Eq. (19) the optimum parameters may change through time and MSD determines the difference between learned parameters and optimum parameters. Therefore MSD should have minimum value for good tracking behavior. Here we analyzed the tracking ability of the proposed perceptron model with MSD.

From Eq. (17) we have

$$\beta_j(n+1) = -\eta\gamma_n(-x(n)(y(n) - \beta^T x(n))) \tag{20}$$

similar to (Haykin 2001), we replace $y(i)$ with $\beta_o^T x + v$

$$\beta_j(n+1) = -\eta\gamma_n(-x(n)(\beta_o^T(n)x(n) + v - \beta^T(n)x(i))) \tag{21}$$

similar to (Haykin 2001), we define

$$\varepsilon(n) = \beta^T(n) - \beta_o^T(n) \tag{22}$$

from Eqs. (21) and (22) we have

$$\beta_j(n+1) = -\eta\gamma_n(-x(n)(\beta_o^T(n)x(n) + \mathrm{v} - (\varepsilon(n) - \beta_o(n))^T x(n))) \tag{23}$$

similar to (Haykin 2001), we define

$$\beta_o^T(n+1) = a\beta_o^T(n) + \omega(n) \tag{24}$$

from Eqs. (23) and (24) we have

$$\varepsilon(n+1) = -\eta\gamma_n(-x(n)(\beta_o^T(n)x(n) + \mathrm{v} - (\varepsilon(n) - \beta_o(n))^T x(n))) - a\beta_o^T(n) + \omega(n) \tag{25}$$

assuming

$$a = 2n\gamma_n \mathrm{x}^T \mathrm{x} \tag{26}$$

with some simplification

$$\varepsilon(n+1) = \eta\gamma_n x(n) v - \eta\gamma_n \varepsilon(n) R + \omega(n) \tag{27}$$

Mean square deviation is

$$E[\varepsilon(n+1)^T\varepsilon(n+1)] = \eta\gamma_n\sigma_v^2 R + \eta\gamma_n K(n)RR + Q \qquad (28)$$

Equation (28) should be minimized, therefore it could be concluded that in Eq. (2), the strategy for adapting error weights should assign weights in a decrementing manner i.e. $w(i+1) \leq w(i)$. Again it should be noted the same result could be reached with sigmoid activation function and linearization with Taylor series.

The convergence analysis stated that the error weights for handling class imbalance should increase whereas the tracking analysis implies they should decrease. This conflict is prevalent in adaptive filter theory (Haykin 2001). We may conclude that there are two choices, the first choice is to increase the error weights in a controlled manner and the second choice is to make little changes (increase and/or decrease) in the error weights. These two choices verify that the method considers both convergence and tracking criteria.

3.1.4 Strategies for adapting error weights

Based on convergence and tracking analysis presented in previous sections two different schemes was proposed for adapting error weights to handle class imbalance. The proposed methods are shown in the pseudo code below

Adaptive error weighting I

$$\begin{aligned}
&\textit{if } \textit{classification_rate}(t, \textit{minor}) \\
&\quad < \textit{classification_rate}(t-1, \textit{minor}) \\
&\quad w_{\textit{minor}}(t) = w_{\textit{minor}}(t-1) + \alpha_1 \\
&\quad w_{\textit{major}}(t) = w_{\textit{major}}(t-1) + \beta_1 \\
&\textit{elseif } \textit{classification_rate}(t, \textit{major}) \\
&\quad < \textit{classification_rate}(t-1, \textit{major}) \\
&\quad w_{\textit{major}}(t) = w_{\textit{major}}(t-1) + \beta_2 \\
&\quad w_{\textit{minor}}(t) = w_{\textit{minor}}(t-1) + \alpha_2 \\
&\textit{conditions}: \\
&\quad w_{\textit{minor}}(1) > w_{\textit{major}}(1) \\
&\quad \alpha_1 > \beta_1 \,\&\, \beta_2 > \alpha_2
\end{aligned}$$

Adaptive error weighting II

$$\begin{aligned}
w_{\textit{Minor}}(t) &= (\textit{num_minor}(t) + \textit{num_major}(t))/\gamma \\
&\quad \times \textit{num_minor}(t) \\
w_{\textit{Major}}(t) &= (\textit{num_minor}(t) + \textit{num_major}(t)) \\
&\quad /\textit{num_major}(t) \\
\gamma &= 1/(\textit{imbalance_ratio} \times w_{\textit{Minor}}(1))
\end{aligned}$$

In the first strategy, error weights are adapted based on classifier results in different classes. In this strategy, the classifier results are checked at certain instants (for example every 500 instants) and weights are incremented depending on classifier results. This issue is not in conflict with the online nature of the model, because the NN weights are still updated with each incoming instance. On the other hand since class imbalance does not change in all instants, the error weights are updated in certain instants.

The pseudo code shows that the error weights are updated in both classes but with different amounts. This issue is related to the nature of handling imbalanced data; when we weight errors of instances from one class, we are decreasing the error weight of instances from the other class. The error weights are updated in both classes so that increasing accuracy in the minority class doesn't result in decreasing the overall accuracy. This strategy has six parameters, namely $w_{minor}(1)$, $w_{major}(1)$, α_1, β_1, β_2, α_2. The first two parameters are the initial error weights of minority and majority class and the four other parameters are the error weight incrementation values. The proposed RLS perceptron model with this error weighting strategy is named recursive least square adaptive cost perceptron I (RLSACP I).

The second strategy for adapting error weights is a simple one, in which weights are adjusted based on number of instances in different classes. In this strategy the number of instances in the minority and majority class are counted for a fixed number of the most recent data and the weights are assigned accordingly. This strategy has two parameters, namely $w_{minor}(1)$, $w_{major}(1)$which are the initial error weights of minority and majority class. γ is a scaling factor which depends on the imbalance ratio and initial error weight in minority class. This parameter controls the amount of difference between error weights in different classes. The proposed RLS perceptron model with this error weighting strategy is named recursive least square adaptive cost perceptron II (RLSACP II). The pseudo code of the proposed method is shown next.

4 Algorithm: online perceptron for non-stationary and imbalanced data streams

Inputs: $x_p = (x_{1p}, x_{2p}, \ldots, x_{ip},),\ p = 1 \ldots P$
$t_p = (t_{1p}, t_{2p}, \ldots, t_{ip},)$

1: Initialize perceptron parameters randomly
2: For each sample p $(p=1,\ldots,P)$
3: calculate the output of the perceptron
4: adapt error weights
5: update parameters of perceptron using Eq. (7)
6: end of for

5 Experimental results

In this section we present experiments to evaluate the effectiveness of the proposed method. Experiments are performed on artificial and real world data streams. All the datasets have two classes. Since our research is on imbalanced datasets, similar to a recent research (Chen and He 2010), a preprocessing was done on the datasets to have a high imbalance ratio. In this preprocessing, data is randomly removed from the dataset until the specified imbalance ratio is achieved. In all the experiments the imbalance ratio is set to 0.05. The datasets are show in Table 1.

This preprocessing is repeated ten times for each dataset and the obtained results are the average results for these ten random samples. The activation function for RLSACP1 and RLSACP2 is sigmoid ($f(z) = 1/(1 + e^{-z})$), the learning rate (η) is set to 0.1 and the forgetting factor (λ) is 0.9. In RLSACP I,

$$w_{minor}(1) = 5 \text{ and } w_{major}(1) = 1,\ \alpha_1 = 4 \times \frac{w_{minor}(1)}{100}$$
$$\alpha_2 = \frac{w_{minor}(1)}{100},\ \beta_1 = \frac{w_{minor}(1)}{100}, \beta_2 = 2 \times \frac{w_{minor}(1)}{100}$$

w_{minor} is higher than w_{major} because errors in the minor class are more important. The same is true for α_1 and β_2. $\alpha_1, \beta_1, \alpha_2$ and β_2 are scaled to control the adaptation of error weights. Previous methods on IDSC have mostly focused on batch solutions (Gao et al. 2007; Chen and He 2010; Ditzler and Polikar 2010), therefore we compared the proposed algorithm with previous online perceptron models. The compared methods are listed in Table 2.

In Table 2, PAI and PAII are large margin perceptrons with particular adaptive learning rates. LBP and RBP are perceptrons with budgets (memory). The budget stores previous mistakes which are used in the model training. In LBP, when the budget is full the oldest instance and in RBP a random instance is removed. LBSOP and RBSOP are combination of LBP and RBP with second-order perceptron. SOP is a perceptron which buffers mistakes and shifts the hyperplane based on the correlation of previous mistakes with current instance. RLSP is a perceptron based on the RLS adaptive filter with fixed forgetting factor. All of these methods have some kind of approach for handling drift.

Table 1 Datasets

Name	Minority	Majority	Total	Attributes
SEA (Street and Kim 2001)	45,520	2,400	47,920	3
STAGGER (Cesa-Bianchi et al. 2005)	11,340	600	11,940	3
Electricity Market (Widmer and Kubat 1996)	15,988	812	16,800	4
Weather (Harries 1999)	12,461	655	13,116	8
Haberman (NOAA 2010)	47,673	2,327	50,000	3
Pima (NOAA 2010)	47,528	2,472	50,000	8
Yeast (NOAA 2010)	48,281	1,719	50,000	8
Ecoli (NOAA 2010)	47,609	2,391	50,000	7

Table 2 Compared methods

Algorithm	Description
PA I	Passive-aggressive I (Crammer et al. 2006)
PA II	Passive-aggressive II (Crammer et al. 2006)
RBP	Randomised budget perceptron (Cavallanti et al. 2007)
LBP	Least recent budget perceptron (Cavallanti et al. 2007)
LBSOP	Least recent budget second-order perceptron (Cavallanti et al. 2007)
RBSOP	Randomised budget second-order perceptron (Cavallanti et al. 2007)
RLSP	Recursive least square perceptron with fixed forgetting factor
RLSACP I	First proposed algorithm RLS + adaptive cost
RLSACP II	Second proposed algorithm RLS + adaptive cost

5.1 Evaluation

In this section we present some issues related to the evaluation scheme.

5.1.1 *Evaluation metrics for imbalanced data*

An important issue in classification problems is the evaluation metric. Usually classification accuracy, which is the percent of correct predictions, is used to evaluate the result of classifiers. Classification accuracy or error is not suitable for imbalanced datasets, because a classifier has imbalanced results in different classes and this could not be noticed by the accuracy or error metric. Therefore in class imbalance learning, other metrics that represent the accuracy in each class are used (He and Garcia 2009). Two common metrics are geometric mean and f-measure.

$$Geometric\ mean = \sqrt{TPR \times TNR} \tag{29}$$

$$F - measure = \frac{(1+\beta)^2 \times recall \times precision}{\beta^2 \times recall + precision}$$
$$TPR = \frac{TP}{TP + FN},\ TNR = \frac{TN}{TN + FP},$$
$$recall = \frac{TP}{TP + FN},\ precision = \frac{TP}{TP + FP} \tag{30}$$

In Eq. (29), TP is true positive, FN is false negative, TN is true negative and FP is false positive. In this study we use the geometric mean metric for evaluating the algorithms because it is simple and unlike f-measure takes into account notion of true negatives.

5.1.2 Evaluation method

Online classifiers could be evaluated with two methods. The first method is the overall evaluation, in which the metric is evaluated on the whole data stream. The second evaluation method is the incremental learning curve. This curve is a two-dimensional curve whose horizontal axis corresponds to the number of instances that have been classified by the time t and the vertical axis corresponds to a metric. This curve is plotted for all the instances of the dataset. This metric determines the incremental efficiency of the classifier.

5.2 Synthetic datasets

In this section we present experiments on artificial data streams. All of these data streams have been used in previous research.

5.2.1 Sea

SEA is a frequently used synthetic dataset in previous stream mining research (Street and Kim 2001). This dataset contains three features with random values in [0, 10]. The label of the instances is specified using a threshold for the sum of the first two features. Concept drift is added to the dataset by adjusting the threshold periodically. This dataset simulates an abrupt change in the class concepts.

Similar to the original design of the SEA dataset, we categorize the whole data stream into four blocks and for each of these blocks the threshold is fixed and set to 8, 9, 7 and 9.5. In each block, 12,000 instances are generated. Examples that have the sum of two features greater than the threshold, belong to the majority class and others belong to the minority class. The results are detailed in Table 3 and Fig. 2.

Table 3 Results for SEA dataset

Algorithm	TPR	TNR	GM
PA I	0	1.0000	0
PA II	0	1.0000	0
RBP	0.0045	0.9998	0.0610
LBP	0.0053	0.9998	0.0646
LBSOP	0.0976	0.9997	0.3054
RBSOP	0.0966	0.9996	0.3077
RLSP	0.0000	0.9999	0.0020
RLSACP I	0.0194	0.9908	0.1354
RLSACP II	0.0093	0.9953	0.0903

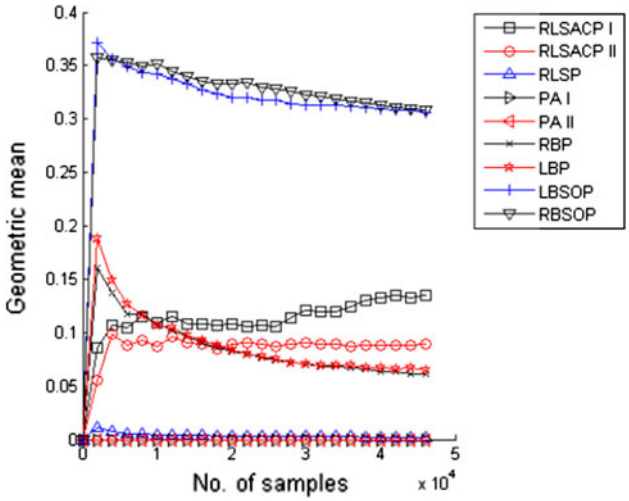

Fig. 2 Incremental learning curve for the SEA dataset

Table 4 Feature values of STAGGER dataset

Feature	First value	Second value	Third value
Size	Small	Medium	Large
Color	Red	Green	Blue
Shape	Square	Circular	Triangular
Classes			
1	Size = small and color = red		
2	(color = green or shape = circular) or (size = medium or size = large)		

In this dataset RBSOP and LBSOP have better results than other methods which is related to the usage of second order information in these methods compared to their counterparts LBP and RBP (Cesa-Bianchi et al. 2005).

5.2.2 Stagger

The STAGGER dataset is a popular artificial dataset used in previous stream mining research (Widmer and Kubat 1996). This dataset contains three categorical features, each having three distinct values. The features are size, color and shape and the class concepts are specified by conditions on features. The values of the feature values and the conditions for concepts are detailed in Table 4.

In the original STAGGER dataset there are three concepts, but here we used the condition of the third concept

Table 5 Results for the STAGGER dataset

Algorithm	TPR	TNR	GM
PA I	0	1.0000	0
PA II	0	1.0000	0
RBP	0.0028	0.9988	0.0394
LBP	0.0027	0.9988	0.0351
LBSOP	0.0027	0.9976	0.0378
RBSOP	0.0025	0.9977	0.0335
RLSP	0.0003	0.9995	0.0082
RLSACP I	0.0270	0.9777	0.1583
RLSACP II	0.0335	0.9711	0.1773

Table 6 Results for the Electricity Market dataset

Algorithm	TPR	TNR	GM
PA I	0	1.0000	0
PA II	0	1.0000	0
RBP	0.0010	0.9991	0.0240
LBP	0.0010	0.9991	0.0236
LBSOP	0.0037	0.9980	0.0568
RBSOP	0.0031	0.9981	0.0502
RLSP	0.0011	0.9995	0.0198
RLSACP I	0.0446	0.9682	0.2058
RLSACP II	0.0913	0.9242	0.2803

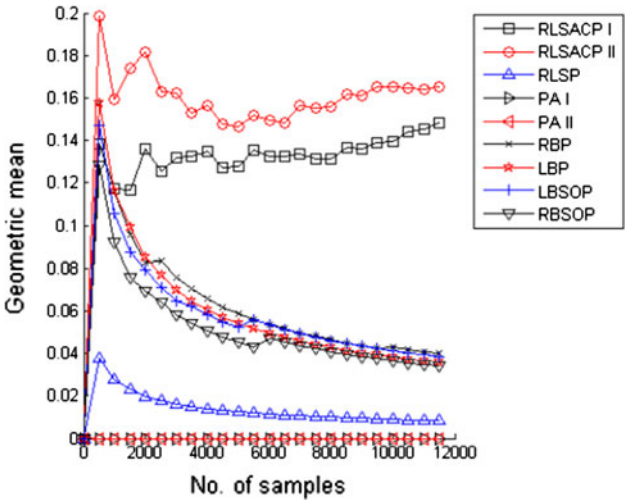

Fig. 3 Incremental learning curve for the STAGGER dataset

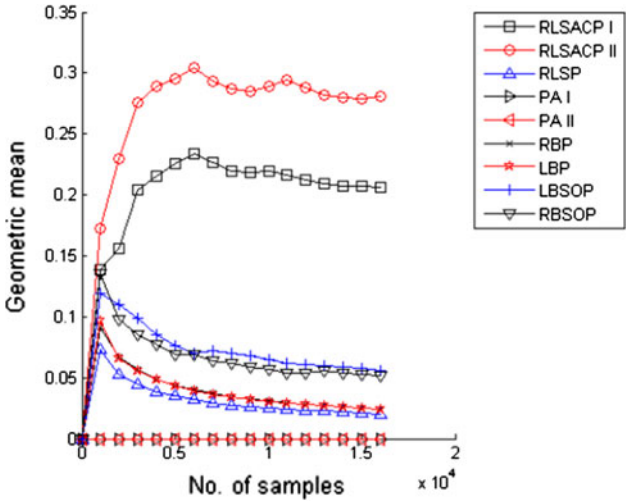

Fig. 4 Incremental learning curve for the Electricity Market dataset

for the second concept to have two concepts only. The class of each instance is specified with some conditions on the values of the features. The concepts are changed by alternating the condition. The results are detailed in Table 5 and Fig. 3.

In this dataset RLSACPI and RLSACPII have better results than other methods which is related to the adaptive error weighting strategy. LBSOP, RBSOP, RLSP, PA I and PA II have lower results because they have no mechanism for handling class imbalance. PA I and PA II are large margin online methods therefore have no mechanism for handling drift. On the other hand RLSP has a forgetting ability, causing better results than PA I and PA II. LBP and RBP have a better mechanism for handling drift than RLSP in which the influence of particular instances is eliminated. Finally LBSOP and RBSOP are better than LBP and RBP because they take into account the features of second-order perceptron in which advances the drift handling ability.

5.3 Real world datasets

5.3.1 Electricity Market

The electricity market dataset is a real world data stream collected from the Australian New South Wales Electricity Market between the years 1996 to 1998 (Harries 1999). The original dataset contains 45,312 records, but since some instances have missed feature values, we only retain the records after 11 May 1997, which sums up to 27,549 records. The original dataset has eight features, but here, similar to (Chen and He 2010) we only use the last four features, which are NSW electricity demand, the VIC Price, the VIC electricity demand, and the scheduled transfer between state. In this dataset the target class is electricity price fluctuations (up/down). The results are detailed in Table 6 and Fig. 4.

In this dataset RLSACPI and RLSACPII have better results than other methods. The results in other methods are downgrading because of the geometric mean metric.

Other methods have lower results because they are only capable of handling drifts. PA I and PA II have no mechanism for handling drifts.

5.3.2 Weather

The Weather dataset was created by the National Oceanic and Atmospheric Administration (NOAA), part of the United States Department of Commerce (USDC) (NOAA 2010). This dataset was generated by compiling a database of weather measurements from over 7,000 weather stations worldwide between 1949 and 1999. This dataset contains 18,159 data points each represented by eight features which are related to the weather condition (for example temperature, dew point and etc). The target class is the weather status which is 'rain' or 'no rain'. The imbalance ratio of the whole dataset is set to 0.05. The results are detailed in Table 7 and Fig. 5.

Table 7 Results for the weather dataset

Algorithm	TPR	TNR	GM
PA I	0	1.0000	0
PA II	0	0.9999	0
RBP	0.0021	0.9988	0.0347
LBP	0.0017	0.9988	0.0285
LBSOP	0.0105	0.9962	0.0999
RBSOP	0.0102	0.9967	0.0999
RLSP	0.0012	0.9980	0.0244
RLSACP I	0.0716	0.9556	0.2607
RLSACP II	0.1144	0.9208	0.3229

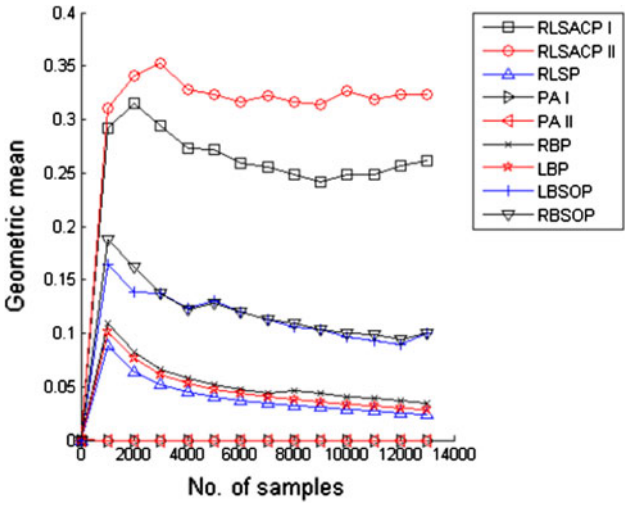

Fig. 5 Incremental learning curve for the weather dataset

In this dataset RLSACPI and RLSACPII have better results than other methods. LBSOP and RBSOP have the next better results. PA I and PA II have the worst result because they have no mechanisms for handling neither drift nor imbalance.

Table 8 Results for the UCI dataset

Metrics	TPR	TNR	GM
Dataset/algorithms			
Haberman			
PA I	0	1.0000	0
PA II	0	1.0000	0
RBP	0.0008	0.9997	0.0229
LBP	0.0004	0.9997	0.0143
LBSOP	0.0015	0.9994	0.0361
RBSOP	0.0012	0.9994	0.0330
RLSP	0.0000	0.9999	0.0021
RLSACP I	0.0264	0.9792	0.1602
RLSACP II	0.0348	0.9691	0.1822
Pima			
PA I	0.0000	1.0000	0.0020
PA II	0.0000	1.0000	0.0020
RBP	0.0006	0.9996	0.0234
LBP	0.0005	0.9996	0.0204
LBSOP	0.0023	0.9986	0.0474
RBSOP	0.0020	0.9986	0.0440
RLSP	0.0001	0.9997	0.0049
RLSACP I	0.0748	0.9539	0.2660
RLSACP II	0.1315	0.9139	0.3452
Yeast			
PA I	0	1.0000	0
PA II	0	1.0000	0
RBP	0.0025	0.9997	0.0455
LBP	0.0032	0.9997	0.0523
LBSOP	0.0143	0.9990	0.1186
RBSOP	0.0126	0.9991	0.1111
RLSP	0.0015	0.9988	0.0314
RLSACP I	0.1569	0.9087	0.3713
RLSACP II	0.7647	0.3092	0.4406
Ecoli			
PAI	0.0005	1.0000	0.0071
PAII	0.0005	1.0000	0.0071
RBP	0.0821	0.9997	0.0821
LBP	0.0810	0.9997	0.0810
LBSOP	0.4995	0.9993	0.4995
RBSOP	0.4926	0.9993	0.4926
RLSP	0.0708	0.9976	0.0708
RLSACP I	0.4856	0.4416	0.4856
RLSACP II	0.4996	0.5253	0.5032

5.3.3 UCI datasets

In this section we report results on data streams generated from UCI datasets (UCI Repository of Machine Learning Database 2007). Yang et al. (2006) proposed an iterative method to convert a normal dataset to a non stationary data stream using the Markov chain concept. In this method, a random attribute is selected in each iteration. If the attribute is nominal, each of its values is treated as a state in the Markov chain. For each state 2,000 instances with the same attribute value are selected and added to the final data stream. On the other hand if the attribute is numeric, the instances are sorted in ascending order according to the attribute values and then added to the final data stream. This procedure is repeated until a desired number of data, for example 50,000 is reached. We use this method to convert some imbalanced datasets from the UCI repository to imbalanced data streams with concept drift. The results are detailed in Table 8 and Fig. 6.

In all the UCI datasets except for Ecoli, RLSACPI and RLSACPII have better results than other methods. In the Ecoli dataset, LBSOP and RBSOP have better incremental results but the overall result is similar.

5.3.4 Main achievements

By observing the experiment results we could conclude that all methods except for RLSACP I and RLSACP II have downgrading results on the datasets, which is related to the imbalance nature of the datasets and geometric mean (GM) metric. The GM metric is the product of true positive and false positive rate. The minor class is the positive and

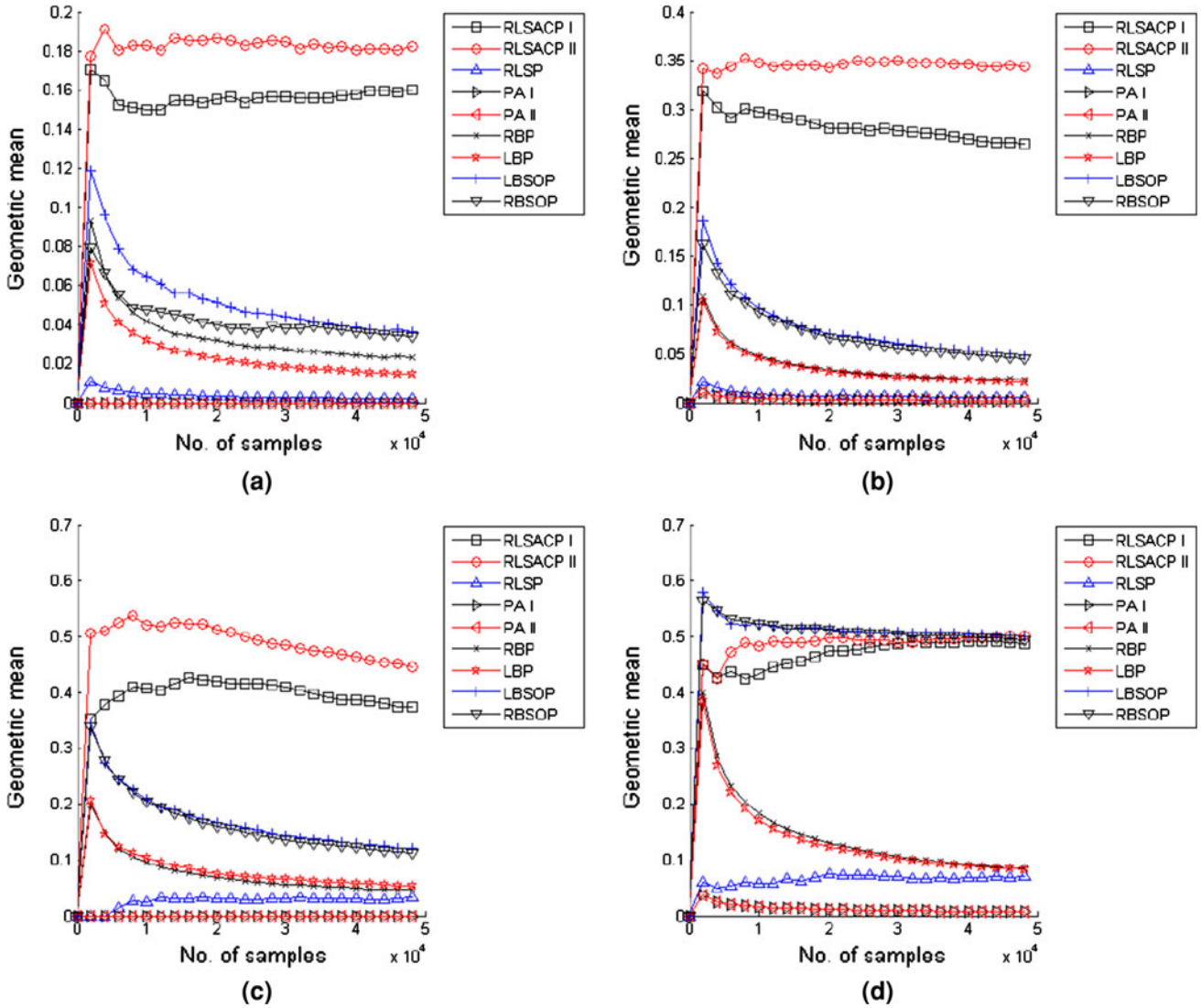

Fig. 6 Incremental learning curve (geometric mean) for UCI datasets, **a** Haberman, **b** Pima, **c** Yeast, **d** Ecoli

major negative. All of the methods except for RLSACP I and RLSACP II have a high true negative rate and very weak true positive rate which causes downgrading GM value. In all experiments RLSACP II has better results compared to RLSACP I. Figures 3 and 4 are two representative graphs that show this behavior precisely. Other methods have different strategies for handling drift which causes distinct results.

6 Discussion

6.1 Statistical significance

In this section we use the analysis of variance (ANOVA) test to verify the statistical significance of the empirical results (Alpaydın 2010). ANOVA is a statistical test used for comparing the response of several algorithms to single or multiple factors. Similar to other statistical tests, ANOVA uses hypothesis testing to verify the statistical significance. In hypothesis testing we determine a null hypothesis which we tend to reject in the statistic test. In ANOVA the null hypothesis is when all the algorithms have similar average results. We use the results of all the experiments to perform the ANOVA test. We set the significance factor (α) to 0.05. The ANOVA test rejected the null hypothesis. The box plot of the results is shown in Fig. 7.

Following this step we use the Turkey's honestly significance difference (HSD) statistical test to compare the results of each pair of algorithms. The result of the HSD test is detailed in Table 9. RLSACP I and RLSACP II have significantly better results than all other algorithms. It should be noted that the average result of RLSACP II is better than RLSACP I.

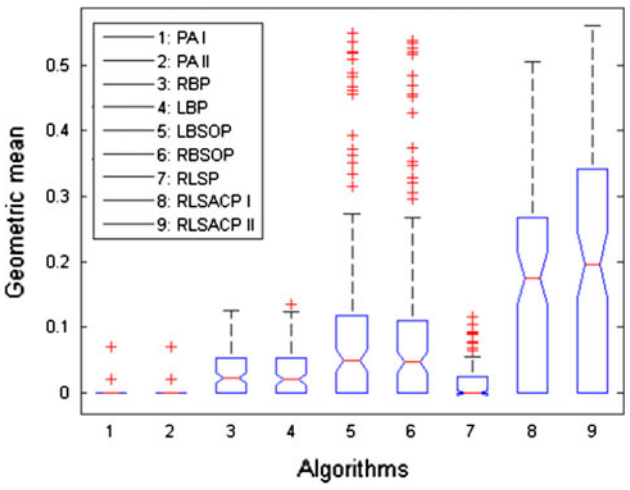

Fig. 7 ANOVA test result, 1: PAI, 2: PAII, 3: RBP, 4: LBP, 5: LBSOP, 6: RBSOP, 7: RLSP, 8: RLSACP I, 9: RLSACP II

Table 9 Results of HSD test

Algorithm	Average geometric mean	Significant from
PA I	0.0008	–
PA II	0.0008	–
RBP	0.0303	–
LBP	0.0291	–
LBSOP	0.1092	PA I, PA II, RBP, LBP, RLSP
RBSOP	0.1065	PA I, PA II, RBP, LBP, RLSP
RLSP	0.0149	–
RLSACP I	0.1858	PA I, PA II, RBP, LBP, RLSP, LBSOP, RBSOP
RLSACP II	0.2129	PA I, PA II, RBP, LBP, RLSP, LBSOP, RBSOP

6.2 Sensitivity to imbalance ratio

Imbalance ratio specifies the rate of imbalance of a dataset. In this section the sensitivity of results to imbalance ratio in RLSACP I and RLSACP II e better results among other algorithm is analyzed. We changed the imbalance ratio from 0.01 to 0.1 and checked the results for SEA and Electricity Market datasets. The results are shown in Fig. 8.

Figure 8 shows RLSACP I and RLSACP II have similar results in both datasets. In both algorithms the results are improved as the imbalance ratio becomes higher (0.1 is high) which is sensible for classifiers that handle class imbalance.

7 Conclusion

We have proposed online perceptron models for classifying non-stationary and imbalanced data streams. Imbalanced data stream classification (IDSC) embraces two significant challenges namely non-stationarity and class imbalance. Non-stationarity is changes in the underlying function generating the data and class imbalance is vast difference between the numbers of instances in different classes. Previous research on IDSC have mostly focused on batch solutions (Gao et al. 2007; Chen and He 2010; Ditzler and Polikar 2010). Here we proposed online methods. By online we mean the classifier model is updated with every incoming instance (Tsymbal 2004). The proposed method is an online perceptron model. The main contribution is a new perceptron error model for which is inspired from the recursive least square (RLS) filter error model. In the proposed error model non-stationarity is handled with the forgetting factor (λ) existent in the RLS error model and for handling class imbalance adaptive error weighting strategies are proposed. These strategies are verified with

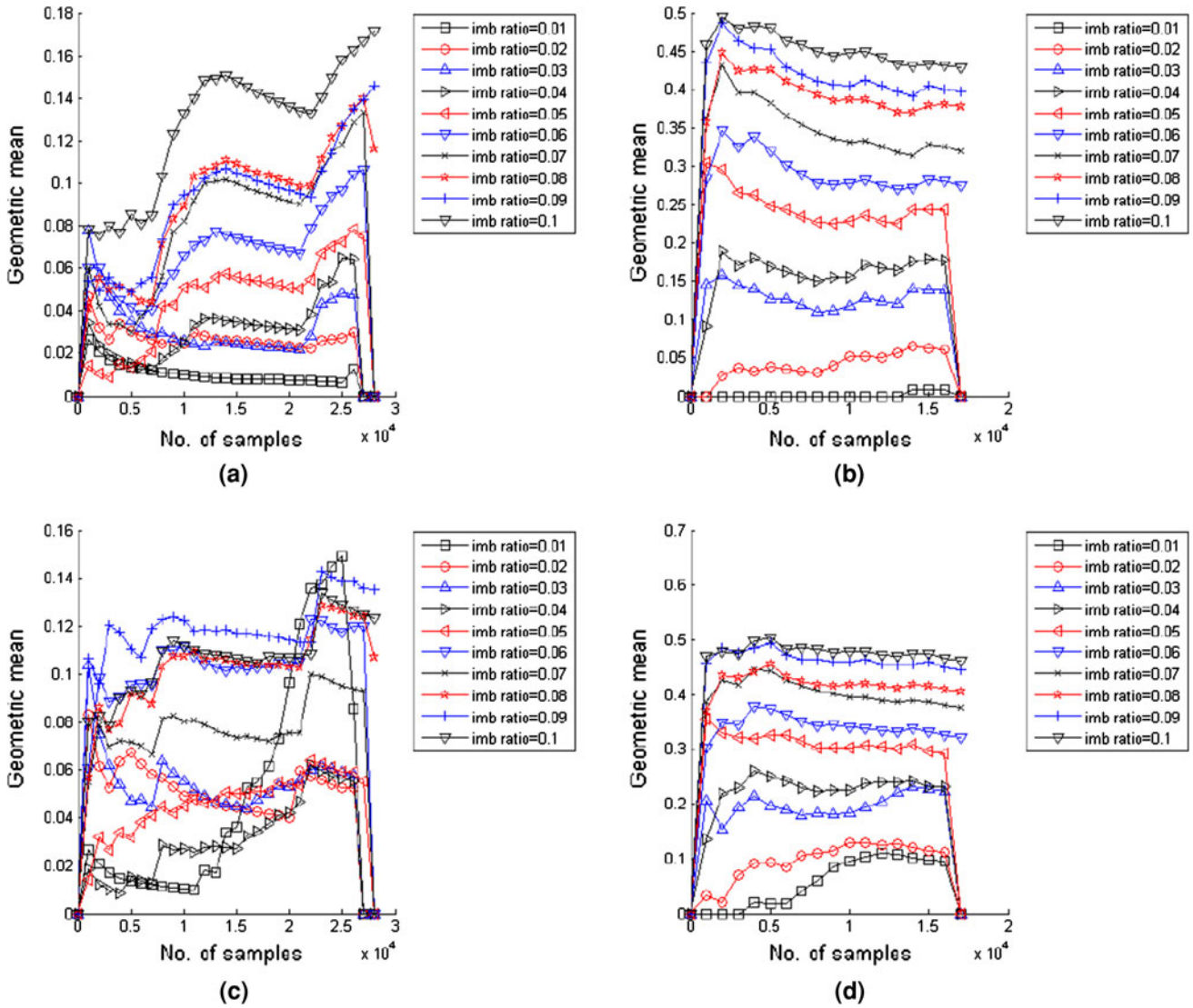

Fig. 8 Sensitivity of results to imbalance ratio, **a** RLSACP I in SEA dataset, **b** RLSACP I in Electricity Market dataset, **c** RLSACP II in SEA dataset, **d** RLSACP II in Electricity Market dataset

convergence and tracking theories in adaptive filters. In the first proposed algorithm (RLSACP I) error weights are adapted based on classifier results and in the second algorithm (RLSACP II) the imbalance ratio is proposed as the method for adapting error weights.

The proposed method has been evaluated using the geometric mean metric, which is a metric for evaluating methods classifying imbalanced datasets (He and Garcia 2009). Similar to previous research on data stream classification, the evaluation has been done in two modes, namely overall result and incremental result. Since previous research on IDSC has focused on batch solutions (Gao et al. 2007; Chen and He 2010; Ditzler and Polikar 2010), the proposed method has been compared with seven other online perceptron models in previous research.

The experimental results showed statistically significance of the proposed methods. Statistical significance was verified using the ANOVA test followed by the HSD test with a significance factor (α) of 0.05. Results also show the advancement of RLSACP II to RLSACP I.

An important issue in algorithms for handling class imbalance is the imbalance ratio of the datasets. Experiments were performed to show the influence of this issue. The experiments showed RLSACP I and RLSACP II have similar results. In both algorithms the results are improved as the imbalance ratio becomes higher (higher imbalance ratio means less imbalance).

For future research we aim to propose extensions such as Multi Layer Perceptron (MLP) and kernel classifier with the RLS error function. These extensions will improve the results on nonlinear and/or non-separable datastes.

References

Abdulsalam H, Skillicorn DB, Martin P (2011) Classification using streaming random forests. Knowl Data Eng IEEE Trans 23(1):22–36

Alippi C, Boracchi G, Roveri M (2011) A just-in-time adaptive classification system based on the intersection of confidence intervals rule. Neural Netw 24(8):791–800

Alpaydın E (2010) Introduction to machine learning, 2nd edn. The MIT Press, Cambridge

Cavallanti G, Cesa-Bianchi N, Gentile C (2007) Tracking the best hyperplane with a simple budget Perceptron. Mach Learn 69(2):143–167. doi:10.1007/s10994-007-5003-0

Cesa-Bianchi N, Conconi A, Gentile C (2005) A second-order perceptron algorithm. SIAM J Comput 34(3):640–668. doi:10.1137/s0097539703432542

Chen S, He H (2010) Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evol Syst 2(1):35–50

Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y (2006) Online passive-aggressive algorithms. J Mach Learn Res 7:551–585

Ditzler G, Polikar R (2010) An ensemble based incremental learning framework for concept drift and class imbalance. Paper presented at the world congress on computational intelligence (WCCI), 18–23 July 2010, Barcelona, Spain

Duda RO, Hart PE, Stork DG (2000) Pattern classification. Wiley-Interscience, New York

Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. Neural Netw IEEE Trans 22(10):1517–1531

Gama J (2010) Knowledge discovery from data streams. Chapman & Hall/CRC Press, Boca Raton

Gao J, Fan W, Han J, Yu PS (2007) A general framework for mining concept-drifting data streams with skewed distributions. Paper presented at the SIAM international conference, Minneapolis, MN

Goldenshluger A, Nemirovski A (1997) On spatial adaptive estimation of nonparametric regression. Math Methods Stat 6:135–170

Harries M (1999) Splice-2 comparative evaluation: electricity pricing. University of South Wales, NSW, Australia

Haykin S (2001) Adaptive filter theory, 4th edn. Prentice Hall, Englewood Cliffs

He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284

Martínez-Rego D, Fontenla-Romero O, Alonso-Betanzos A (2012) Nonlinear single layer neural network training algorithm for incremental, nonstationary and distributed learning scenarios. Pattern Recogn 45(12):4536–4546

Masud MM (2009) Adaptive classification of scarcely labeled and evolving data streams. University of Texas, Dallas

Masud MM, Jing G, Khan L, Jiawei H, Thuraisingham BM (2011) Classification and novel class detection in concept-drifting data streams under time constraints. Knowl Data Eng IEEE Trans 23(6):859–874

NOAA (2010) "Weather data". http://users.rowan.edu/~polikar/research/NSE/

Pavlidis NG, Tasoulis DK, Adams NM, Hand DJ (2011) Landa perceptron: an adaptive classifier for data streams. Pattern Recogn 44(1):78–96

Soda P (2011) A multi-objective optimisation approach for class imbalance learning. Pattern Recogn 44(8):1801–1810

Street NW, Kim Y (2001) A streaming ensemble algorithm (SEA) for large-scale classification. Paper presented at the 7th ACM SIGKDD international conference on knowledge discovery and data mining, 26–29 Aug 2001, San Francisco, California, USA

Sun J, Li H (2011) Dynamic financial distress prediction using instance selection for the disposal of concept drift. Expert Syst Appl 38(3):2566–2576

Tahir MA, Kittler J, Yan F (2012) Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recogn 45(10):3738–3750

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

UCI Repository of Machine Learning Database (2007) School of information and computer science, Irvine, CA: University of California. http://www.ics.uci.edu/~mlearn/MLRepository.html

Wang J, You J, Li Q, Xu Y (2012) Extract minimum positive and maximum negative features for imbalanced binary classification. Pattern Recogn 45(3):1136–1145

Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23:60–101

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