Tracking Sparse Linear Classifiers

Tingting Zhai, Frederic Koriche, Hao Wang, Yang Gao

Tracking Sparse Linear Classifiers

Tingting Zhai, Frédéric Koriche, Hao Wang, Yang Gao, *Member, IEEE*

***Abstract*—In this paper, we investigate the problem of sparse online linear classification in changing environments. We first analyze the tracking performance of standard online linear classifiers, which use gradient descent for minimizing the regularized hinge loss. The derived shifting bounds highlight the importance of choosing appropriate step-sizes in presence of concept drifts. Notably, we show that a better adaptability to concept drifts can be achieved using constant step-sizes rather than the state-of-the-art decreasing step-sizes. Based on these observations, we then propose a novel sparse approximated linear classifier, called SALC, which uses a constant step-size. In essence, SALC simply rounds small weights to zero for achieving sparsity, and controls the truncation error in a principled way for achieving a low tracking regret. The degree of sparsity obtained by SALC is continuous and can be controlled by a parameter which captures the tradeoff between the sparsity of the solution and the regret performance of the algorithm. Experiments on nine stationary datasets shows that SALC is superior to the-state-of-the-art sparse online learning algorithms especially when the solution is required to be sparse; on seven groups of non-stationary datasets with various total shifting amount, SALC also presents a good ability to track drifts. When wrapped with a drift detector, SALC achieves a remarkable tracking performance regardless of the total shifting amount.**

***Index Terms*—concept drift, sparse online learning, online gradient descent, shifting regret.**

I. Introduction

ONLINE linear classification is a well-studied problem in machine learning, with various practical applications, ranging from anti-spam filtering and fraud detection, to medical diagnosis and sentiment analysis. For example, in the anti-spam filtering task, the learning algorithm receives at each round t a vector representation $\boldsymbol{x}_t \in \mathbb{R}^d$ of an incoming email, and it is required to predict whether $\boldsymbol{x}_t \in \mathbb{R}^d$ is a spam, or not, using its current separating hyperplane $\boldsymbol{w}_t \in \mathbb{R}^d$. Once the learner has committed to its prediction, the true label y_t is revealed, and the learner incurs a loss $f(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))$ that measures the inaccuracy of the prediction. In light of this information, the learner is allowed to find another separating hyperplane for the next round, in the hope of improving its predictive performance.

In contrast with batch algorithms which are trained and tested on separate data sets, online algorithms process instances one-by-one, and are evaluated on the whole sequence of instances. Thus, online learners can be significantly more efficient and more practical than batch learners for handling large-scale, and possibly streaming, data sets. Furthermore, online algorithms do not require any distributional assumption about data instances; they are analyzed within an adversarial scenario. Specifically, the usual metric advocated in the literature of online learning is the *static regret*, defined as the maximum difference in cumulative loss between the online learner and the best separating hyperplane chosen with the benefit of hindsight, on any possible sequence of data instances. However, for modern applications involving high-dimensional data streams, the online learning framework is faced with two major challenges.

1) *Curse of dimensionality*: this issue is related to the fact that in high-dimensional application domains, many features are irrelevant for predicting the output label. For example, in the anti-spam filtering task, the feature vector of associated with an electronic message many involve tens or even hundreds of thousands features, and many of them are irrelevant for deciding whether the message is a spam, or not. For such domains, standard linear classifiers which predict using a separating hyperplane defined for all features, are prone to overfit the data [9], [10].
2) *Concept drift*: this is related to gradual changes of the underlying relation between the input data and the target label [2]–[7]. For example, in personalized anti-spam filtering tasks [8], a concept drift indicates that the user is changing her opinion about separating "spam" messages from "ham" ones. Similar concept drifts arise in fraud detection, sentiment analysis, and many other applications in which the best predictive model is susceptible to change other time. Standard online algorithms which, according to the notion of static regret, aims at competing with a single model, are likely to perform poorly on drifting data streams, due to the fact that the target model is susceptible to change with concept drifts.

So far, a wide variety of online linear classification algorithms have been proposed in the literature [11]–[16]. In particular, gradient descent (GD) techniques are endowed with an attractive combination of predictive accuracy and computational efficiency. Namely, GD algorithms in [11], [12], [14] are defined over the same weight updating rule, but differ in the choice of the learning rate, which yields different convergence guarantees. These techniques, however, cannot handle the curse of dimensionality in high-dimensional data streams. On the other hand, sparse models provide an elegant solution to the curse of dimensionality, by reducing the number of non-zero coordinates [9], [10]. Sparsity also promotes interpretability of models and decreases the prediction time. Informally, the sparsity of a predictor $\boldsymbol{w} \in \mathbb{R}^d$ is measured by its ℓ_0 pseudo-norm, that is, the number of non-zero coordinates of $\boldsymbol{w}$. Since the ℓ_0 pseudo-norm is non-convex, achieving sparsity using this metric is far from easy. In the literature, this difficulty is circumvented by three main techniques: (1)

T. Zhai, H. Wang and Y. Gao (**Corresponding Author*) are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, 210023. E-mail: zhtt.go@gmail.com; wanghao@nju.edu.cn; gaoy@nju.edu.cn.

F. Koriche is with Center of Research in Information in Lens, Université d'Artois, Lens, France, 62307. E-mail: koriche@cril.fr.

imposing an ℓ_1 norm constraint, (2) adding an ℓ_1 norm regularization to the loss function, or (3) using ℓ_0 truncation.

Methods using an ℓ_1 norm constraint project the solution $\boldsymbol{w}_t$ obtained by gradient descent onto an ℓ_1 ball, using efficient projection methods [17], [18]. The shortcoming of this approach is that the ℓ_1 norm constraint does not always guarantee sparsity [19]. Methods using ℓ_1 regularization consider an ℓ_1 norm regularized problem at each iteration. Duchi *et al.* [20] proposed the FOBOS algorithm, which at each iteration, first performs a sub-gradient descent to get an intermediate solution, and then finds a new solution that has a low ℓ_1 norm complexity and that stays close to the intermediate solution. The second stage can be implemented efficiently by truncating coefficients below a threshold in the intermediate solution. Langford *et al*. [21] argued that such truncation operation is too aggressive and proposed a truncated gradient algorithm (TrunGrad), which gradually shrinks the coefficients to zero by a small amount. In contrast with the above approaches, Jin *et al.* [22] proposed an ℓ_0-based truncation method that satisfies a predefined ℓ_0 pesudo-norm constraint at each iteration. Their OFS algorithm [22] first projects the predictor $\boldsymbol{w}_t$ (obtained from gradient descent) onto an ℓ_2 norm ball, so that most of the numerical values of $\boldsymbol{w}_t$ are concentrated to their largest elements, and then keeps only the B largest elements in $\boldsymbol{w}_t$, where B is a pre-defined constant.

Yet, all the aforementioned sparse algorithms have been developed and analyzed for stationary data streams. The goal of this paper is to study *sparse* online linear classification in *drifting data*. In such environments, the best predictor is not fixed, but is susceptible to evolve over time, and the challenge is to investigate the tracking behavior of sparse online classifiers using a *shifting regret* metric. Informally, the shifting regret captures the difference in cumulative loss between the online algorithm and the best *sequence* of separating hyperplanes, which reflect the gradual changes to be made for dealing with shifting data. To this point, very few is known about the theoretical and practical tracking performance of popular classifiers, such as Pegasos [14] and Norma [11]. Even less known is the tracking ability of popular sparse classifiers. In this paper, we make the following contributions:

1) We derive shifting regret bounds for the classic gradient-descent technique, which provides a new insight that the step-size choice takes a crucial part in the tracking performance of the learner. Specifically, for gradient-descent techniques, a better adaptability to concept drifts can be achieved using constant step-sizes rather than standard decreasing step-sizes.
2) We propose a novel sparse approximated linear classification algorithm, abbreviated as SALC, which uses a constant step-size. SALC achieves sparsity by truncating small elements in the model at each round and controls the truncation error in a natural way for ensuring low regret. The sparsity parameter in SALC can control the degree of sparsity from no sparsity to total sparsity. A concise shifting regret bound of SALC is also provided and analyzed.
3) Empirical evidence for the influence of the step-size choice on the tracking performance of the learner is provided. Moreover, extensive experiments on nine stationary datasets demonstrate the superiority of SALC especially when high sparsity level of the solution is desired. On seven groups of non-stationary datasets, SALC presents good capability to track concept drifts. Wrapped with a drift detector, SALC outperforms the state-of-the-art sparse online learning algorithms significantly in tracking performance no matter how large the amount of drifts is.

The remaining paper is organized as follows. Section II introduces the notation and the problem definition. Section III presents the GD-based techniques and derives the shifting regret bounds for three classic step-size schedulings. The proposed sparsity strategy and its corresponding regret analysis are provided in Section IV. Comprehensive experiments on both stationary and non-stationary datasets are given in Section V. Section VI provides related works in online linear classification and regret metrics in static and dynamic environments. Finally, Section VII concludes the paper.

II. Notation and problem definition

Let $||\boldsymbol{w}||$ and $||\boldsymbol{w}||_1$ denote ℓ_2 norm and ℓ_1 norm of vector $\boldsymbol{w}$ respectively. Let $||\boldsymbol{w}||_0$ denote ℓ_0 pseudo-norm of $\boldsymbol{w}$, that is, the number of non-zeros elements in $\boldsymbol{w}$. We use $\mathbb{1}[a]$ to denote the indicator function, that is, $\mathbb{1}[a] = 1$ if a is true, otherwise $\mathbb{1}[a] = 0$.

Online linear classification can be viewed as a repeated game between the learning algorithm and its environment. During each trial t of the game, the environment supplies a new instance vector $\boldsymbol{x}_t \in \mathbb{R}^d$, where we assume that there exists a constant R such that $||\boldsymbol{x}_t|| \leq R$ for any t, then the learner is required to choose a predictor $\boldsymbol{w}_t \in \mathbb{R}^d$ to classify $\boldsymbol{x}_t$. Once the learner has committed to its choice, the true label $y_t \in \{-1, +1\}$ of $\boldsymbol{x}_t$ is revealed by the environment, and the learner suffers a loss $f(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))$ that measures the discrepancy between the predicted label and the true label. For binary classification tasks, a natural metric is the 0-1 loss function, that is, $f(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))$ = 1 if $\mathrm{sgn}(\boldsymbol{w}_t^\top \boldsymbol{x}_t) \neq y_t$, and 0 otherwise. However, the 0-1 loss function is non-convex, and hence, hard to optimize. To alleviate this issue, we shall use the convex hinge loss $f(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t)) = \max\{0, 1 - y_t \boldsymbol{w}^\top \boldsymbol{x}_t\}$, which is a well-known surrogate of the 0-1 loss. In what follows, we shall often use $f_t(\boldsymbol{w})$ as an abbreviation of $f(\boldsymbol{w}; (\boldsymbol{x}_t, y_t))$.

The tracking performance of the learning algorithm is measured using the notion of *shifting regret* [23]–[26], which compares the cumulated loss of the learner against an arbitrary *time-varying* sequence of separating hyperplanes $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_T \in (\mathbb{R}^d)^T$ chosen collectively in hindsight. Formally, the shifting regret is defined as

$$\text{Regret}^T \triangleq \sum_{t=1}^{T} f_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{u}_t).$$

A low shifting regret means that the learner can compete with a broad family of time-varying classifiers. In order to provide bounds on the shifting regret, we use the the maximum norm of the comparator sequence, i.e. $U = \max_t ||\boldsymbol{u}_t||$, together

with the sum of the variation distances between each pair of the time-adjacent comparators [23], [24], that is,

$$\mathcal{S}_T = \sum_{t=1}^{T-1} ||\boldsymbol{u}_t - \boldsymbol{u}_{t+1}||.$$

Intuitively, $\mathcal{S}_T$ captures the drifting behavior of the environment; a regret bound with a small dependence on $\mathcal{S}_T$ is preferable, as it captures a better robustness to changes.

III. GD-BASED LINEAR CLASSIFIERS (GD-LC)

Linear classification algorithms in [11], [12], [14] use GD to produce a sequence of predictors $(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots) \in (\mathbb{R}^d)^\infty$. In each trial t, the regularized loss function is given by

$$f_t(\boldsymbol{w}) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + \max\{0, 1 - y_t \boldsymbol{w}^\top \boldsymbol{x}_t\},$$

where λ is the regularization parameter for controlling the complexity of the predictor. GD works as follows: Initially, $\boldsymbol{w}_1 = \boldsymbol{0}$. At trial t, the new predictor $\boldsymbol{w}_{t+1}$ is constructed by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla f_t(\boldsymbol{w}_t),$$

where $\eta_t < 1/\lambda$ is the step-size and $\nabla f_t(\boldsymbol{w}_t)$ is the gradient of $f_t(\boldsymbol{w})$ at $\boldsymbol{w}_t$. Given that

$$\nabla f_t(\boldsymbol{w}_t) = \lambda \boldsymbol{w}_t - \gamma_t y_t \boldsymbol{x}_t, \tag{1}$$

where $\gamma_t = \mathbb{1}[y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t < 1]$, the gradient descent update can be rewritten as

$$\boldsymbol{w}_{t+1} = (1 - \lambda\eta_t)\boldsymbol{w}_t + \eta_t \gamma_t y_t \boldsymbol{x}_t.$$

Different choices of step-size η_t yield different instantiations of GD-LC. Originally, the theoretical performance of GD-LC was analyzed in stationary streams. We will extend its analysis to non-stationary streams. The next two lemmas will help to bound the shifting regret of GD-LC. Note that Lemma 1 is based on one lemma from [27], however we still provide the proof in Appendix A for completeness.

Lemma 1. *Let $f_1, f_2, \cdots$ be a sequence of λ-strongly convex functions w.r.t. $\boldsymbol{w}$. Let $\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots$ be a sequence of vectors in $\mathbb{R}^d$ such that for $t \geq 1, \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla f_t(\boldsymbol{w}_t)$. Assume that there exists a constant G such that $||\nabla f_t(\boldsymbol{w}_t)|| \leq G$ for all t. For an arbitrary comparator sequence $(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots) \in (\mathbb{R}^d)^\infty$, the following inequality holds:*

$$\sum_{t=1}^{T}(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}_t)) \leq \sum_{t=1}^{T}\left(\frac{D_t}{2\eta_t} - \frac{\lambda}{2}||\boldsymbol{w}_t - \boldsymbol{u}_t||^2\right) + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t,$$

where $D_t = ||\boldsymbol{w}_t - \boldsymbol{u}_t||^2 - ||\boldsymbol{w}_{t+1} - \boldsymbol{u}_t||^2$ is the relative progress towards $\boldsymbol{u}_t$ in round t.

Lemma 2. *GD-LC satisfies $||\boldsymbol{w}_t|| \leq \frac{R}{\lambda}$ and $||\nabla f_t(\boldsymbol{w}_t)|| \leq 2R$ for all t.*

Proof. We prove by induction that $||\boldsymbol{w}_t|| \leq \frac{R}{\lambda}$. The base case $||\boldsymbol{w}_1|| = 0 \leq \frac{R}{\lambda}$. Assume that $||\boldsymbol{w}_k|| \leq \frac{R}{\lambda}$ for $t = k$, then

$$\begin{aligned}||\boldsymbol{w}_{k+1}|| &\leq (1 - \lambda\eta_k)||\boldsymbol{w}_k|| + \eta_k||\gamma_k y_k \boldsymbol{x}_k|| \\ &\leq (1 - \lambda\eta_k)\frac{R}{\lambda} + \eta_k R = \frac{R}{\lambda}.\end{aligned}$$

Therefore, $||\boldsymbol{w}_t|| \leq \frac{R}{\lambda}$ for all t. Proving $||\nabla f_t(\boldsymbol{w}_t)|| \leq 2R$ is trivial since

$$||\nabla f_t(\boldsymbol{w}_t)|| \leq \lambda||\boldsymbol{w}_t|| + ||\boldsymbol{x}_t|| \leq \lambda \cdot \frac{R}{\lambda} + R = 2R$$

is a direct result from Eq. 1. □

We are now ready to present the main theorem on GD-LC, that different choices of the step-size η_t yields very different regret guarantees. Therefore, to handle non-stationary streams, it is important to choose η_t wisely.

Theorem 1. *GD-LC can achieve different shifting regret bounds using different schedulings of the step-size η_t:*

1) *If $\eta_t = 1/(\lambda t)$, then*

$$\textit{Regret}^T_{\textit{GD-LC}} \leq 2RT\mathcal{S}_T + \frac{2R^2}{\lambda}(1 + \ln T).$$

2) *If $\eta_t = \eta/\sqrt{t}$, where $0 < \eta < 1/\lambda$, then*

$$\textit{Regret}^T_{\textit{GD-LC}} \leq \frac{R\sqrt{T}}{\lambda\eta}(\mathcal{S}_T + U) + 4\eta R^2\sqrt{T}.$$

Hence if we know $\mathcal{S}_T$ in advance, setting $\eta = \Theta(\sqrt{\mathcal{S}_T})$ leads to a shifting regret bound of $O(\sqrt{T\mathcal{S}_T})$.

3) *If $\eta_t = \eta \in (0, 1/\lambda)$, then*

$$\textit{Regret}^T_{\textit{GD-LC}} \leq \frac{R}{\lambda\eta}(\mathcal{S}_T + U) + 2\eta R^2 T.$$

Hence, setting $\eta = \Theta(\sqrt{\mathcal{S}_T/T})$ produces a bound of order $O(\sqrt{T\mathcal{S}_T})$ which is in fact better than the bound from $\eta_t = \eta/\sqrt{t}$ by a constant factor.

Proof. See Appendix B. □

Remarks The above bounds quantify how GD-LC with various step-sizes are affected by changes in the environment. On the one hand, when $\mathcal{S}_T \approx 0$ and T is large enough, GD-LC with $\eta_t = 1/(\lambda t)$ produces the smallest regret upper bound of $O(\ln T)$. On the other hand, when $\mathcal{S}_T \gg 0$, GD-LC with $\eta_t = \eta$ is more robust due to the fact that the dependence on $\mathcal{S}_T$ is sublinear in T. Thus, the step-size $\eta_t = \eta/\sqrt{t}$ produces a natural compromise between stationary case $\mathcal{S}_T \approx 0$, and non-stationary case $\mathcal{S}_T \gg 0$. We provide empirical evidence in Section V-B, showing that for GD-LC, a better concept tracking capability can be achieved using $\eta_t = \eta$ than using $\eta_t = 1/(\lambda t)$ or $\eta_t = \eta/\sqrt{t}$. Based on these observations, we shall adopt a constant step-size in our proposed sparse online learning algorithm, in order to achieve a better tracking performance.

IV. A SPARSE APPROXIMATED LINEAR CLASSIFIER (SALC)

Sparse models can alleviate the *curse of dimensionality* by providing hyperplanes with few non-zero coordinates. In this section, we explore a new option for achieving sparsity. As mentioned earlier, constant step-size scheduling is used for better tracking performance. Our idea for achieving sparsity is to obtain a sparse *approximated* solution after the GD update at each online round. The key of our approach is thus to control the error between the exact solution and the approximated

one in a principled way so that the algorithm can converge. Our method, SALC (sparse approximated linear classifier), is presented in Algorithm 1.

Algorithm 1: SALC

Input: Data $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{\infty}$, regularization parameter λ, step-size $\eta \in (0, 1/\lambda)$, error tolerance ϵ, the acceptable minimum number of features $\kappa \geq 1$

Output: $\{\boldsymbol{w}_t\}_{t=1}^{\infty}$

1 $\boldsymbol{w}_1 = \boldsymbol{0}$;
2 **for** $t = 1, 2, \cdots$ **do**
3 Receive $\boldsymbol{x}_t$;
4 Predict the label of $\boldsymbol{x}_t$ using $\boldsymbol{w}_t$;
5 Receive y_t and suffer loss $f_t(\boldsymbol{w}_t)$;
6 $\boldsymbol{z}_{t+1} = (1 - \lambda\eta)\boldsymbol{w}_t + \eta\gamma_t y_t \boldsymbol{x}_t$;
7 **if** $||\boldsymbol{z}_{t+1}||_0 > \kappa$ **then**
8 $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} ||\boldsymbol{w}||_0$ subject to $||\boldsymbol{w} - \boldsymbol{z}_{t+1}|| \leq \epsilon$ and $||\boldsymbol{w}||_0 \geq \kappa$;

As we can see, Step 6 of Algorithm 1 computes a vector $\boldsymbol{z}_{t+1}$ following the update rule of GD-LC. After that, sparsity is achieved by Step 8, which essentially searches for the most sparse vector $\boldsymbol{w}_{t+1}$ that is close to $\boldsymbol{z}_{t+1}$ and that has acceptable number of non-zero features. The value of truncation error ϵ controls how close $\boldsymbol{w}_{t+1}$ to $\boldsymbol{z}_{t+1}$, thus affects the sparsity of $\boldsymbol{w}_{t+1}$. The parameter κ is a small integer, which helps to avoid any improper situations where the value of ϵ is set too large.

Note that the general form of the optimization problem at Step 8: $\min_{\boldsymbol{w}} ||\boldsymbol{w}||_0$ subject to $||\boldsymbol{w} - \boldsymbol{H}\boldsymbol{z}|| \leq \epsilon$, for a given matrix $\boldsymbol{H}$, a vector $\boldsymbol{z}$ and an error ϵ, is NP-hard [28]. However, we find that when $\boldsymbol{H}$ is the identity matrix, the problem can be efficiently solved using a simple greedy algorithm. Specifically, at the t-th round, to implement Step 8 we do the following: first let $\boldsymbol{w}_{t+1}$ be a copy of $\boldsymbol{z}_{t+1}$; then repeat setting the non-zero element with the smallest absolute value in $\boldsymbol{w}_{t+1}$ to 0 until no more elements can be set to 0 without violating the constraints $||\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1}|| \leq \epsilon$ and $||\boldsymbol{w}_{t+1}||_0 \geq \kappa$.

Lemma 3. *$\boldsymbol{w}_{t+1}$ obtained by the above greedy procedure is an optimal solution of the minimization problem of Step 8 of Algorithm 1.*

Proof. The implementation process of Step 8 implies that $\boldsymbol{w}_{t+1}$ is a feasible solution. Let C be the set of non-zero elements in $\boldsymbol{z}_{t+1}$. Let Q be a subset of C, of which elements are set to 0 in $\boldsymbol{w}_{t+1}$ and set $\bar{Q} = C - Q$. Let v_j be an any element in $\bar{Q}$.

Note that when the greedy procedure terminates, both constraints of the problem remain satisfied, but the marginal case of at least one constraint is reached. In the first case, when the marginal case of the constraint $||\boldsymbol{w}_{t+1}||_0 \geq \kappa$ is reached, we have $||\boldsymbol{w}_{t+1}||_0 = \kappa$. In the second case, when the marginal case of the constraint $||\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1}|| \leq \epsilon$ is reached, we have $\sum_{vi \in Q} v_i^2 + v_j^2 > \epsilon^2$. In the first case, it is obvious that $\boldsymbol{w}_{t+1}$ is an optimal solution. We next prove that in the second case, $\boldsymbol{w}_{t+1}$ is also an optimal solution.

Let $H = \{\boldsymbol{w} \in \mathbb{R}^d : \kappa \leq ||\boldsymbol{w}||_0 \text{ and } ||\boldsymbol{w}||_0 < \|\boldsymbol{w}_{t+1}\|_0\}$. Let $I = \{1, 2, \cdots, ||\boldsymbol{w}_{t+1}||_0 - \kappa\}$. For any $k \in I$, let $E_k = \{\boldsymbol{w} \in H : ||\boldsymbol{w}||_0 = k + \kappa - 1\}$. For any $k \in I$, also let $\boldsymbol{w}_k^* = \arg\min_{\boldsymbol{w} \in E_k} ||\boldsymbol{w} - \boldsymbol{z}_{t+1}||$. It is clear that for any $\boldsymbol{w}_k^*$, all the non-zero elements of $\boldsymbol{w}_k^*$ must belong to C. Let $s = \arg\min_{k \in I} ||\boldsymbol{w}_k^* - \boldsymbol{z}_{t+1}||$, then we can obtain that $||\boldsymbol{w}_s^* - \boldsymbol{z}_{t+1}|| \leq ||\boldsymbol{w} - \boldsymbol{z}_{t+1}||$ holds for any $\boldsymbol{w} \in H$, and $||\boldsymbol{w}_s^*||_0$ must be equal to $||\boldsymbol{w}_{t+1}||_0 - 1$. Given the fact that all the non-zero elements of $\boldsymbol{w}_s^*$ belong to C, $\boldsymbol{w}_s^*$ must be obtained by setting the element in $\boldsymbol{w}_{t+1}$ with the smallest absolute value to 0. Therefore, we have $||\boldsymbol{w}_s^* - \boldsymbol{z}_{t+1}|| > \epsilon$, which means that $||\boldsymbol{w} - \boldsymbol{z}_{t+1}|| > \epsilon$ holds for any $\boldsymbol{w} \in H$. So any $\boldsymbol{w} \in H$ is infeasible and $\boldsymbol{w}_{t+1}$ is an optimal feasible solution in the second case.

□

Lemma 4. *SALC satisfies $||\boldsymbol{w}_t|| \leq \frac{R}{\lambda}$ and $||\nabla f_t(\boldsymbol{w}_t)|| \leq 2R$ for all t.*

Proof. By using $\|\boldsymbol{w}_{t+1}\| \leq \|\boldsymbol{z}_{t+1}\|$, the proof is similar to that of Lemma 2. □

We can now prove a shifting regret bound of SALC.

Theorem 2. *Let $\epsilon = \rho\eta^2$, where ρ is a pre-defined parameter, then SALC can achieve the following shifting regret bound:*

$$Regret_{SALC}^T \leq \frac{R}{\lambda\eta}(\mathcal{S}_T + U) + (2R^2 + \frac{\rho R}{\lambda} + \rho U)\eta T + \frac{\rho^2\eta^3 T}{2}.$$

When $\mathcal{S}_T \approx 0$, by setting $\eta = \Theta\left(1/\sqrt{T}\right)$, we obtain a regret bound of $O\left(\sqrt{T}\right)$. When $\mathcal{S}_T > 0$, setting $\eta = \Theta\left(\sqrt{\mathcal{S}_T/T}\right)$ and assuming $\mathcal{S}_T = o(T)$, a sublinear regret bound $o(T)$ is achieved; in particular, when $\mathcal{S}_T = O\left(T^{\frac{1}{3}}\right)$, a bound of order $O\left(T^{\frac{2}{3}}\right)$ can be achieved.

Proof. We consider the difference between $\boldsymbol{w}_{t+1}$ and $\boldsymbol{u}_t$.

$$\begin{aligned}
&||\boldsymbol{w}_{t+1} - \boldsymbol{u}_t||^2 \\
=\ &||\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1} + \boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 \\
=\ &||\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1}||^2 + ||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 \\
&+ 2(\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1})^\top(\boldsymbol{z}_{t+1} - \boldsymbol{u}_t) \\
\leq_1\ &\epsilon^2 + ||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 + 2||\boldsymbol{w}_{t+1} - \boldsymbol{z}_{t+1}|| \cdot ||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t|| \\
\leq\ &\epsilon^2 + ||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 + 2(||\boldsymbol{z}_{t+1}|| + ||\boldsymbol{u}_t||)\epsilon \\
\leq_2\ &\epsilon^2 + ||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 + 2(\frac{R}{\lambda} + U)\epsilon \\
\leq_3\ &\epsilon^2 + ||\boldsymbol{w}_t - \boldsymbol{u}_t||^2 + 4\eta^2R^2 - 2\eta(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}_t)) \\
&+ 2(\frac{R}{\lambda} + U)\epsilon.
\end{aligned}$$

In the above calculation, $\leq_1$ is due to the feasibility of $\boldsymbol{w}_{t+1}$ and Cauchy-Schwarz inequality. $\leq_2$ is due to Lemma 4 and the boundedness assumption $\|\boldsymbol{u}_t\| \leq U$ for any t. The last inequality $\leq_3$ is due to the fact that

$$\begin{aligned}
&||\boldsymbol{z}_{t+1} - \boldsymbol{u}_t||^2 = ||\boldsymbol{w}_t - \boldsymbol{u}_t - \eta\nabla f_t(\boldsymbol{w}_t)||^2 \\
=&||\boldsymbol{w}_t - \boldsymbol{u}_t||^2 + \eta^2||\nabla f_t(\boldsymbol{w}_t)||^2 - 2\eta(\boldsymbol{w}_t - \boldsymbol{u}_t)^\top\nabla f_t(\boldsymbol{w}_t) \\
\leq&||\boldsymbol{w}_t - \boldsymbol{u}_t||^2 + 4\eta^2R^2 - 2\eta(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}_t)).
\end{aligned}$$

Rearranging the terms and then summing over t, we have

$$\begin{aligned}\text{Regret}^T_{\text{SALC}} &\leq \frac{1}{2\eta}\sum_{t=1}^{T}\left(||\boldsymbol{w}_t-\boldsymbol{u}_t||^2-||\boldsymbol{w}_{t+1}-\boldsymbol{u}_t||^2\right)\\ &\quad+2\eta R^2T+\frac{1}{2\eta}\sum_{t=1}^{T}\epsilon^2+(\frac{R}{\lambda}+U)\sum_{t=1}^{T}\frac{\epsilon}{\eta}\\ &= A_1+A_2+A_3+A_4.\end{aligned}$$

Using directly the proof for Theorem 1, we can obtain

$$A_1 \leq \frac{R}{\lambda\eta}(\mathcal{S}_T+U).$$

Next, we only need to bound A_3 and A_4 caused by the sparsity operation. Considering $\epsilon=\rho\eta^2$, we obtain

$$A_3=\frac{1}{2\eta}\sum_{t=1}^{T}(\rho\eta^2)^2=\frac{\rho^2\eta^3T}{2},$$

$$A_4=(\frac{R}{\lambda}+U)\sum_{t=1}^{T}\rho\eta=(\frac{R}{\lambda}+U)\rho\eta T.$$

Combining these results, we get the stated bound. □

Remarks The bound on $\text{Regret}^T_{\text{SALC}}$ suggests that if the total variation distance $\mathcal{S}_T$ of an arbitrary time-varying classifier sequence is sublinear, SALC can achieve average accumulated loss as small as that of such a sequence as T approaches infinity. The parameter ρ serves as an adjustor between sparsity and regret. A larger value of ρ generally leads to a larger approximation error ϵ, thus yields a sparser solution and more regret. Conversely, a smaller value of ρ produces a denser solution and less regret. If $\rho=0$, we obtain immediately the same bound as the non-sparse GD-LC with step-size $\eta_t=\eta$.

V. Experiments

In order to study the empirical performance of our learning algorithms, we have performed four experiments. The first experiment is to investigate the adaptability of GD-LC under different step-size schedulings, aiming to provide empirical evidence for Theorem 1. The second experiment demonstrates the effectiveness of SALC in producing discriminative sparse solutions. The third and last experiments aim to check how well SALC performs compared with existing state-of-the-art sparse online classifiers on stationary and non-stationary datasets respectively.

A. Datasets

Stationary datasets. We used 9 stationary two-class classification datasets from various domains and with various feature sparsity levels. The characteristics of stationary datasets are presented in Table I, where data density is the maximal number of non-zero features of instances divided by the number of all attributes. The task of *arcene* is to distinguish cancer versus normal patterns from mass-spectrometric data. Both *dexter* and *farm_ads* are text classification problems in a bag-of-words representation. The aim of *gisette* is to separate the highly confusable digits '4' and '9'. These four datasets are from the UCI repository[1]. The datasets *pcmac* and *basehock* are subsets extracted from *20Newsgroups*[2], which contains news messages of 20 different newsgroups. The task of *pcmac* is to separate documents from "ibm.pc.hardware" and "mac.hardware"; and *basehock* is to distinguish "baseball" versus "hockey". *Luad_Brca*, *prostate_GE*, and *SmkCan187* are from biological domains for disease state analysis, where *Luad_Brca* is extracted from the gene expression dataset *RNA-Seq* in the UCI repository, while *prostate_GE* and *SmkCan187* are from the scikit-feature selection repository[3].

TABLE I
A Summary of Stationary Datasets.

Dataset	# features	# train	# test	density
arcene	10000	100	100	71.25%
dexter	20000	300	300	1.65%
gisette	5000	6000	1000	29.6%
basehock	26214	1197	796	6.48 %
pcmac	26214	1168	777	4.5%
farm_ads	54877	3313	830	4.19%
Luad_Brca	20532	331	110	90.57%
prostate_GE	5966	81	21	100%
SmkCan187	19993	149	38	100%

Nonstationary datasets. Using *20Newsgroups*, 7 groups of non-stationary datasets were created.[4] Each group contains 10 datasets (thus there are 70 datasets in total), each simulating the evolution of a particular user's interests in news messages. We used the first 760 news messages from the following 6 newsgroups to create datasets: "ibm.pc.hardware", "mac.hardware", "rec.autos", "rec.motorcycles", "sci.electronics" and "sci.med", thus each dataset created contains 4,560 instances and 26,214 features. The first group, *zeroDt*, was created to have no concept drifts. The news messages of 6 newsgroups are uniformly distributed in each dataset of *zeroDt*, and the labels of the news messages in each dataset is obtained by randomly assigning 3 newsgroups as interesting (positive) and the others as not interesting (negative). The remaining 6 groups of datasets were then created by injecting concept drifts into the datasets in *zeroDt*. Specifically, each dataset in *zeroDt* was split into p equally-sized chunks; then for the adjacent two chunks, we obtained the labels of the news message in the latter chunk by turning randomly q interested newsgroups in the former chunk into uninterested ones and also turning q uninterested newsgroups in the former chunk into interested ones. Therefore, for each concept drift, the user changes her interests in $2q$ newsgroups. The specific value for p and q on each group can be seen in Table II. We can see that $\mathcal{S}_T$ in different groups differs in the number of concept drifts and the amount of drifts; a larger value of p and q lead to a larger $\mathcal{S}_T$.

[1] https://archive.ics.uci.edu/ml/index.php

[2] http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

[3] http://featureselection.asu.edu/datasets.php

[4] In fact, the datasets in *zeroDt* are stationary. Nonetheless, due to the strong connection between *zeroDt* and the other groups, we often discuss these seven groups of datasets together.

TABLE II
A SUMMARY OF NON-STATIONARY DATASETS.

Group Name	# concepts	# drifting newsgroups
zeroDt	$p = 1$	$q = 0$
oneDtTwo	$p = 2$	$q = 1$
oneDtFour	$p = 2$	$q = 2$
oneDtSix	$p = 2$	$q = 3$
threeDtTwo	$p = 4$	$q = 1$
threeDtFour	$p = 4$	$q = 2$
threeDtSix	$p = 4$	$q = 3$

B. Adaptability analysis of GD-LC

With this set of experiments, we investigated how the three step-size schedulings presented in Theorem 1 affect the tracking performance of GD-LC. We also compared GD-LC with the shifting perceptron algorithm (SPA) [23], which is an improved perceptron for dealing with drifting concept and uses polynomial vector decaying scheme to diminish the importance of early update stages. Experiments were performed on the datasets in Table II. The optimal parameter setting for each algorithm on each dataset was manually searched. Table III displays the average online classification accuracy of GD-LC and SPA on *zeroDt* datasets under one-pass (epoch $= 1$) and multi-pass (epoch $= 5, 10, 50, 100$) learning respectively. Note that multiple passes are allowed only on the stationary datasets, of which the purpose is to *simulate* the one-pass learning on a much longer data stream. Table IV presents the results on the non-stationary datasets with $\mathcal{S}_T > 0$. The best result on each dataset are shown in bold.

TABLE III
ONLINE CLASSIFICATION ACCURACY AND STANDARD DEVIATION [%] ON ZERODT DATASETS UNDER MULTIPLE PASSES.

Epoch	GD-LC			SPA
	$\eta_t = 1/(\lambda t)$	$\eta_t = \eta/\sqrt{t}$	$\eta_t = \eta$	
1	87.87 (2.86)	85.62 (4.07)	**88.57** (2.84)	85.89 (2.57)
5	95.05 (1.21)	95.19 (1.43)	**95.43** (1.17)	95.18 (1.13)
10	97.18 (0.79)	96.85 (0.94)	97.03 (0.83)	**97.34** (0.65)
50	99.36 (0.17)	98.83 (0.37)	98.77 (0.40)	**99.44** (0.15)
100	99.67 (0.09)	99.17 (0.26)	99.00 (0.34)	**99.72** (0.07)

TABLE IV
ONLINE CLASSIFICATION ACCURACY AND STANDARD DEVIATION [%] ON NON-STATIONARY DATASETS UNDER ONE-PASS CONSTRAINT.

Dataset	GD-LC			SPA
	$\eta_t = 1/(\lambda t)$	$\eta_t = \eta/\sqrt{t}$	$\eta_t = \eta$	
oneDtTwo	84.22 (2.03)	82.46 (2.50)	**84.74** (2.10)	83.25 (1.40)
oneDtFour	81.86 (2.61)	81.65 (2.46)	**82.64** (2.68)	81.32 (2.36)
oneDtSix	79.81 (3.42)	80.88 (3.36)	**81.28** (3.43)	80.09 (3.12)
threeDtTwo	79.93 (1.21)	80.17 (0.91)	**81.00** (0.91)	80.14 (0.73)
threeDtFour	74.58 (1.12)	77.04 (1.60)	**77.35** (1.53)	76.43 (1.72)
threeDtSix	69.99 (4.11)	75.03 (3.65)	**75.26** (3.27)	74.24 (3.69)

We can observe from Table III that when one-pass learning is required, GD-LC with $\eta_t = \eta$ achieves the best online performance; when multi-pass learning is allowed and epoch ≥ 10, GD-LC with $\eta_t = 1/(\lambda t)$ and SPA outperform GD-LC with $\eta_t = \eta$. From Table IV, we can see that GD-LC with $\eta_t = \eta$ always beats SPA and GD-LC with the other two step-size choices, exhibiting the best online performance in spite of the number of drifts and the amount of drifts. In conclusion, in terms of the three step-size choices of GD-LC, $\eta_t = 1/(\lambda t)$ should be preferred on stationary data provided that the number of instances is sufficient, and $\eta_t = \eta$ should be adopted on non-stationary data. The empirical evidence is consistent with the theoretical results presented in Theorem 1.

C. Sparsity analysis of SALC

In this section, we demonstrate that SALC can work towards a sparse solution and meanwhile does not degrade the learning performance significantly. Experiments were performed on the stationary datasets in Table I. For SALC, the regularization parameter was set to $\lambda = 0.01$, the acceptable minimum number of features was set to $\kappa = 1$, and the learning step-size η on each dataset was chosen by minimizing the regularized regret on that dataset. As in Section V-B, we use the multi-pass trick to simulate the one-pass learning on a much longer data stream. Specifically, SALC was run 10 times on each training set, each time with τ passes and each pass with a new random permutation on the set. The specific values of η and τ on each dataset are shown in Table V. Fig. 1 & 2 plot the number of non-zero features (namely, $||\boldsymbol{w}_t||_0$) and the average cumulated loss $\frac{1}{t}\sum_{i=1}^{t} f_i(\boldsymbol{w}_i)$ achieved by SALC with different values of sparsity parameter ρ during the online learning process on each dataset. The results are averaged on 10 runs.

TABLE V
LEARNING STEP-SIZE η AND THE NUMBER OF PASSES τ ON EACH DATASET.

Dataset	η	τ	Dataset	η	τ
arcene	10^{-8}	500	dexter	10^{-5}	100
gisette	10^{-8}	100	basehock	10^{-3}	100
farm_ads	10^{-2}	100	pcmac	10^{-2}	100
Luad_Brca	$10^{-5.5}$	200	prostate_GE	10^{-4}	1500
SmkCan187	10^{-6}	10000			

We can observe from Fig. 1 & 2 that as the number of instances increases, the number of features kept by SALC with different values of ρ first decreases then tends to stabilize. Meanwhile, the corresponding average cumulated loss diminishes to zero gradually. We also notice that large reduction in the number of features only leads to small additional average regret. Moreover, generally the larger ρ, the less features kept and the more loss produced. Therefore, by adjusting the value of ρ, solutions with different sparsity levels and different regret performance can be obtained. These results suggest clearly that SALC is an effective sparse online learning algorithm.

To further demonstrate that SALC can keep discriminative sparse solutions, we compared SALC with GD-LC on the above datasets. GD-LC used the same parameter settings as SALC. For both algorithms, we recorded $||\boldsymbol{w}_t||_0$ at the end of training and the corresponding test accuracy obtained by evaluating $\boldsymbol{w}_t$ on a separate testing set. The results are displayed in Table VI. The bold font indicates that SALC can achieve similar or better test accuracy even if it uses much fewer non-zero features than GD-LC.

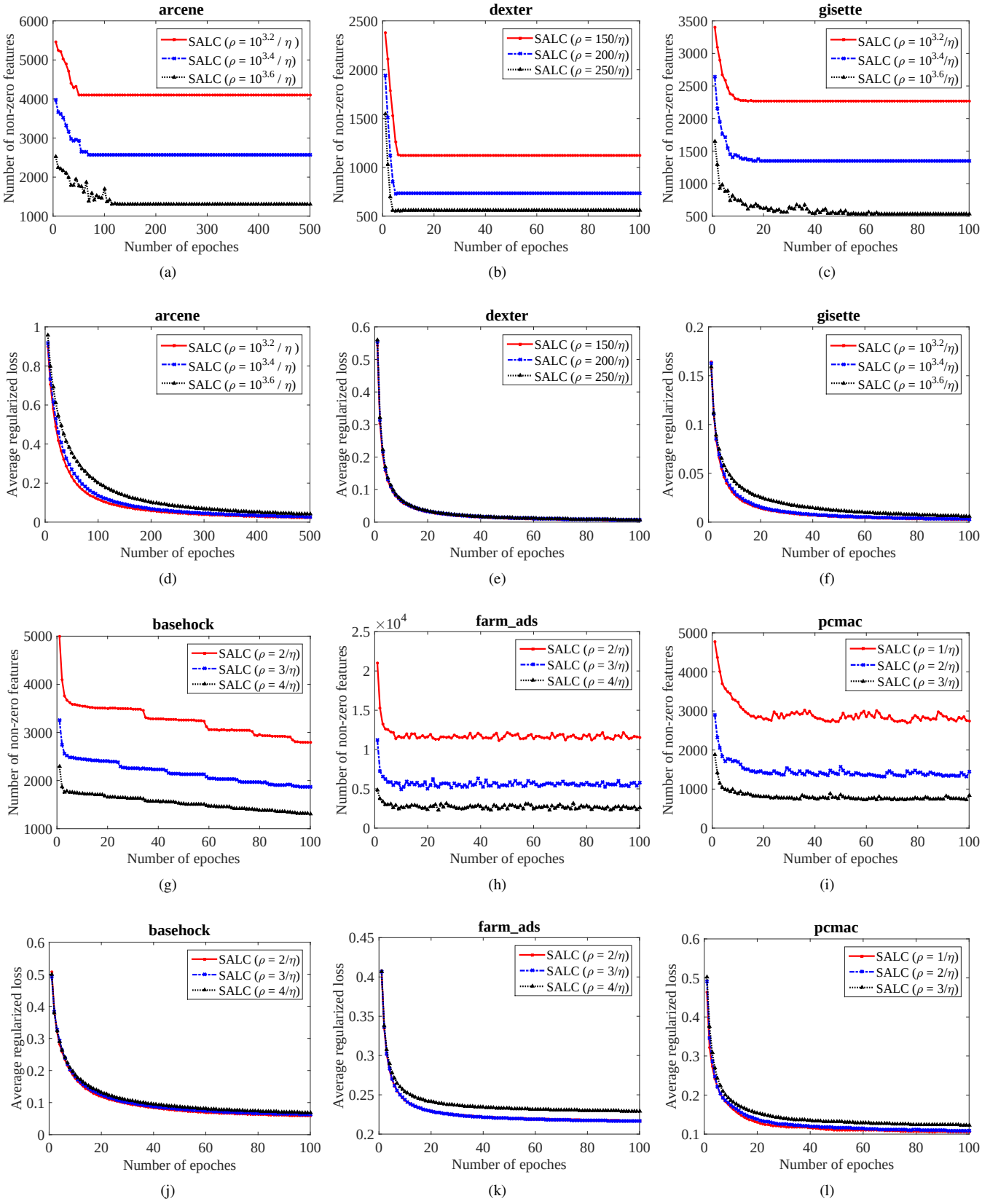

Fig. 1. Number of non-zero features and average cumulated loss achieved by SALC during the online learning process.

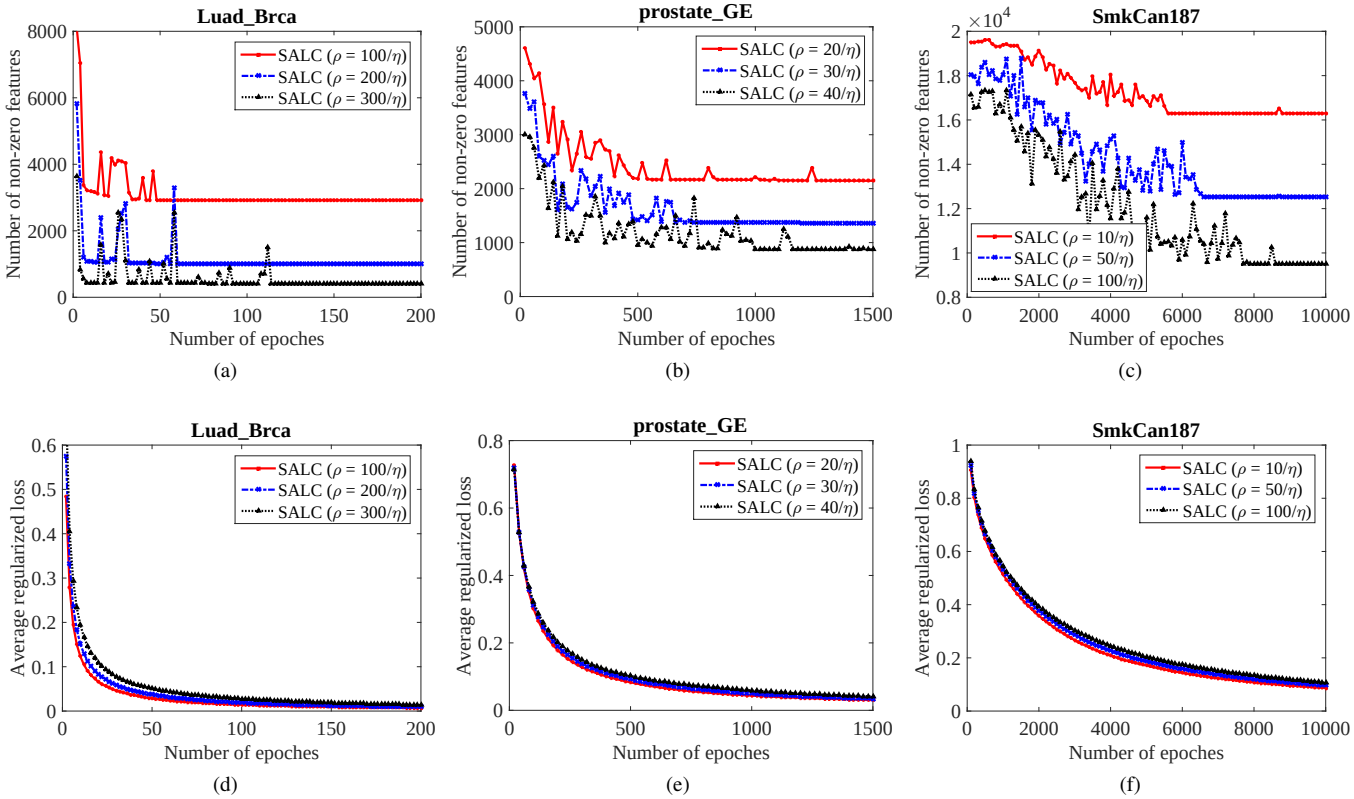

Fig. 2. Number of non-zero features and average cumulated loss achieved by SALC during the online learning process.

TABLE VI
NUMBER OF NON-ZERO FEATURES AT THE END OF TRAINING AND THE CORRESPONDING TEST ACCURACY [%]. STANDARD DERIVATIONS ARE IN PARENTHESES.

Dataset	SALC			GD-LC	
	ρ	$\|\|\boldsymbol{w}_t\|\|_0$	test acc.	$\|\|\boldsymbol{w}_t\|\|_0$	test acc.
arcene	$10^{3.2}/\eta$	4100 (56)	82.80 (1.93)	9912 (8)	83.30 (1.25)
	$10^{3.4}/\eta$	2575 (60)	82.10 (1.73)		
	$10^{3.6}/\eta$	1297 (37)	81.40 (1.35)		
dexter	$150/\eta$	1121 (47)	**91.67 (1.23)**	4260 (131)	91.23 (1.44)
	$200/\eta$	733 (17)	90.93 (1.53)		
	$250/\eta$	557 (14)	90.37 (1.44)		
gisette	$10^{3.2}/\eta$	2270 (29)	**97.74 (0.18)**	4693 (14)	97.54 (0.24)
	$10^{3.4}/\eta$	1351 (28)	**97.60 (0.21)**		
	$10^{3.6}/\eta$	530 (18)	97.28 (0.40)		
basehock	$2/\eta$	2791 (68)	**95.28 (0.16)**	8083 (178)	94.84 (0.12)
	$3/\eta$	1865 (67)	**95.35 (0.55)**		
	$4/\eta$	1307 (31)	**95.68 (0.36)**		
farm_ads	$2/\eta$	11553 (566)	90.39 (0.59)	44220 (280)	90.40 (0.36)
	$3/\eta$	5749 (867)	90.36 (0.57)		
	$4/\eta$	2636 (971)	90.22 (0.87)		
pcmac	$1/\eta$	2742 (144)	**88.29 (0.46)**	7150 (45)	88.08 (1.25)
	$2/\eta$	1441 (207)	**88.31 (0.96)**		
	$3/\eta$	836 (302)	87.75 (2.15)		
Luad_Brca	$100/\eta$	2914 (164)	**100.00 (0.00)**	20148 (12)	100.00 (0.00)
	$200/\eta$	998 (134)	**100.00 (0.00)**		
	$300/\eta$	415 (30)	**100.00 (0.00)**		
prostate_GE	$20/\eta$	2146 (91)	**90.00 (1.51)**	5966 (0)	84.76 (4.38)
	$30/\eta$	1360 (55)	**90.48 (0.00)**		
	$40/\eta$	874 (88)	**90.48 (0.00)**		
SmkCan187	$10/\eta$	16295 (49)	72.89 (1.27)	19993(0)	73.42(0.83)
	$50/\eta$	12543 (212)	72.11 (1.36)		
	$100/\eta$	9494 (191)	**75.26 (2.54)**		

In Table VI, it is clear that on some datasets, SALC can greatly reduce the number of non-zero features while keeping slight test performance degradation, while on other datasets, using much fewer non-zero features, SALC can enhance the test performance. For example, on *arcene*, using nearly 40% of the features kept by GD-LC, SALC suffers 0.5% performance loss; on *dexter*, using 26% of the features kept by GD-LC, SALC achieves 0.44% performance gain, however, using 17% of the features by GD-LC leads to 0.3% performance loss; similar slight performance loss or gain can also be observed on *gisette*, *basehock*, *farm_ad*, *pcmac* and *SmkCan187*; on *Luad_Brca*, SALC achieves the same test accuracy using only 2% of the features by GD-LC; on *prostate_GE*, SALC outperforms GD-LC significantly, since using about 36%, or 23%, or 15% of the features kept by GD-LC, SALC all enjoys more than 5% performance gain, which maybe is due to that the dataset contains lots of noisy features and SALC can reduce noisy features.

D. Comparison with sparse online classifiers on stationary datasets

We compared SALC with the state-of-the-art sparse online learning algorithms, including TrunGrad [21], FOBOS [20] and the method that projects the solution after gradient descent update onto an ℓ_1 ball at each online step (Proj_L1, for short) [17]. Experiments were performed on the stationary datasets. The parameter settings of these algorithms were as follows.

1) *TrunGrad*. After every K rounds of GD update, TrunGrad shrinks the coefficients in $(-\theta, -\alpha)$ and (α, θ) by a small

amount α and rounds the coefficients in $[-\alpha, \alpha]$ to 0, where $\alpha = \eta K g$. We fixed $\theta = +\infty$ and $K = 10$, as suggested in [21] and searched the optimal η that yields the minimum ℓ_1 regularized loss on the training set from $\{10^{-9}, 10^{-8.5}, \cdots, 10^{-1}\}$. After θ, K and η were determined, we varied the gravity parameter g for different ranges to observe how much test accuracy could be obtained under different sparsity levels.

2) *FOBOS*. We searched the optimal η as what we did for TrunGrad, then fixed η and varied the ℓ_1 regularization parameter to tune the sparsity of the solution and to observe the test accuracy.
3) *Proj_L1*. The optimal η was searched in the same way as that for TrunGrad, then η was fixed and the radius r of ℓ_1 ball was varied to see the test accuracy.
4) *SALC*. We fixed $\lambda = 10^{-2}$ and $\kappa = 1$, then searched the optimal η that yields the minimum l_2 regularized loss on the training set from the same range as TrunGrad. When λ, κ and η were determined, we varied the value of the sparsity parameter ρ to see the test performance.

In order to obtain reliable results, under a given parameter setting, each algorithm was run 10 times, each time with τ passes on the training data and each pass with a new random permutation, and then the models learnt were evaluated on a separated test data. The value of τ on each dataset is in Table V. Fig. 3 displays the average test accuracy of these algorithms under different sparsity levels, where the number of non-zero features of the solution is obtained at the end of training.

From Fig. 3, we see that, when the number of non-zero features kept is larger, the performance gap between SALC and the others is insignificant; however, as the number of non-zero features decrease, the gap is growing larger and larger. We also notice that both SALC and TrunGrad achieve good performance on *Luad_Brca* even if they keep very few features, which maybe is due to that the dataset contains a large number of redundant features and SALC and TrunGrad can reduce such features. On *prostate_GE*, as the number of non-zero features kept increase, the test performance of SALC degrades gradually, while that of TrunGrad fluctuates a lot, which is probably caused by lots of noisy features contained in the dataset. On *prostate_GE*, FOBOS cannot achieve continuous sparse solutions, we therefore just provide its scatter plot. Moreover, note that FOBOS and Proj_L1 almost cannot achieve sparse solutions on the three dense biological datasets, *Luad_Brca*, *prostate_GE* and *SmkCan187*, neither can TrunGrad on *SmkCan187*. In conclusion, the promising results suggest clearly that SALC has significant superiority when high sparsity level of the solution is desired.

E. Comparison with sparse online classifiers on non-stationary datasets

To observe how algorithms are affected by concept drifting, we performed experiments on the non-stationary datasets in Table II. Besides TrunGrad, FOBOS and Proj_L1, SALC was also compared with algorithms equipped with a drift detector. Specifically, we wrapped each compared algorithm with a drift detector DDM [29]. DDM tracks the online error of its wrapped algorithm, issues a warning when it observes an increase of error reaching the warning level, and actually alarms a drift when an increase of error reaches the drift level. Algorithms wrapped with DDM will discard their old model when a drift is alarmed and learn from scratch a new model using the instances after warning level. Each algorithm uses the same parameter setting as in Section V-D except the learning step-size, which was set to 0.01 for all algorithms.

Different from the experiments on stationary datasets where multi-pass was allowed, in the non-stationary experiment, only one-pass of data was allowed. Fig. 4 plots online classification accuracy against the number of non-zero features at the end of learning on *zeroDt* datasets with $\mathcal{S}_T = 0$. The same plots on the datasets with $\mathcal{S}_T > 0$ are presented in Fig. 5. The experimental results are averaged on 10 datasets in each group. Note that in order to avoid clutter, we did not display in Fig. 5 the plots for FOBOS and FOBOS+DDM since we observe that on these datasets, the plot of FOBOS coincides with that of Proj_L1, while the plot of FOBOS+DDM coincides with that of Proj_L1+DDM.

When $\mathcal{S}_T = 0$, we can see that SALC outperforms significantly TrunGrad, FOBOS and Proj_L1 when the number of non-zero features kept is less; as the number of non-zero features increases, the gap in online performance grows smaller and eventually vanishes. This phenomenon is consistent with that in Section V-D.

When $\mathcal{S}_T > 0$, we have the following observations:

1) When $\mathcal{S}_T$ is smaller on *oneDtTwo* and *oneDtFour*, SALC performs better than the other algorithms, just like the observations in stationary case; as $\mathcal{S}_T$ grows on *threeDtFour* and *threeDtSix*, the other algorithms overtake SALC when the number of non-zero features kept is larger, however, SALC can beat them when smaller number of non-zero features is kept.
2) As the amount of drifts increases, seen from Fig. 5a-5c or Fig. 5d-5f, algorithms wrapped with DDM achieve increasing performance superiority over their counterparts without DDM. The reason is that when the divergence of time-adjacent concepts is growing large, discarding the old model directly and learning from scratch a new model on the most recent data is superior to incrementally updating the old model.
3) We notice that the maximum number of non-zero features achieved by algorithms with DDM on *threeDtFour* and *threeDtSix* datasets is around 7,200, which is less than their counterparts without DDM. This phenomenon is due to that algorithms without DDM cannot forget the old concept quickly and thus keep features both in the old and new concepts while algorithms with DDM only keep discriminative features in the new concept. The phenomenon does not occur on *threeDtTwo* datasets owing to the less divergence of time-adjacent concepts: most of discriminative features in the old concept remain discriminative in the new concept and thus are kept.
4) SALC with DDM is superior to all the other algorithms no matter how large $\mathcal{S}_T$ is. The advantage is significant when the number of non-zero features kept is smaller.

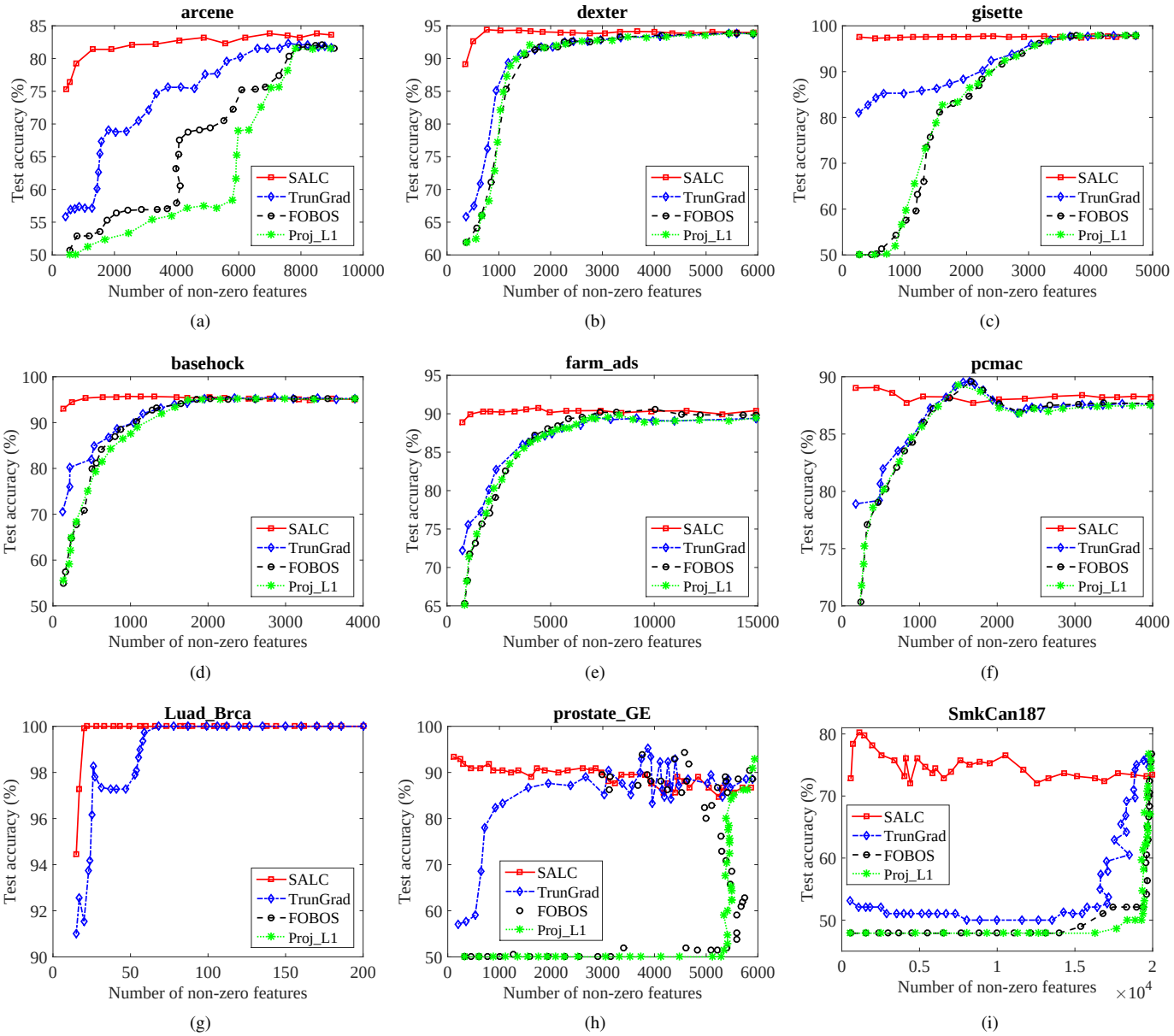

Fig. 3. The test performance under different sparsity levels on stationary datasets. Note that the plots for FOBOS and Proj_L1 do not appear on Luad_Brca since their plots fall outside the specified range.

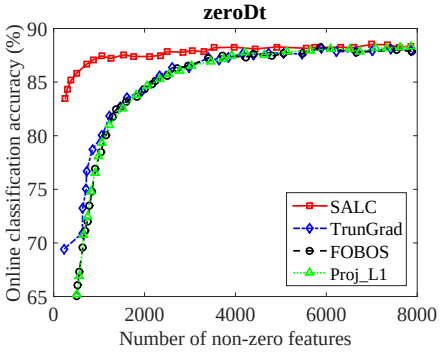

Fig. 4. Online classification accuracy against the number of non-zero features on zeroDt datasets.

In conclusion, we can see that when $\mathcal{S}_T = 0$, SALC achieves the best online performance when high sparsity level of the solution is required, meanwhile when $\mathcal{S}_T > 0$, SALC wrapped with DDM still can keep the performance superiority.

F. Comparison with online feature selection

We also compared SALC with the online feature selection algorithm OFS [22] in terms of online tracking capability. SALC differs from OFS in that SALC, like most of online sparse learning algorithms, imposes only soft restrictions on the number of non-zero features and does not explicitly address the feature selection problem. In spite of the differences, we compared both algorithms in the online classification tasks. Experiments were performed on the non-stationary datasets with $\mathcal{S}_T > 0$. Both algorithms used the regularization pa-

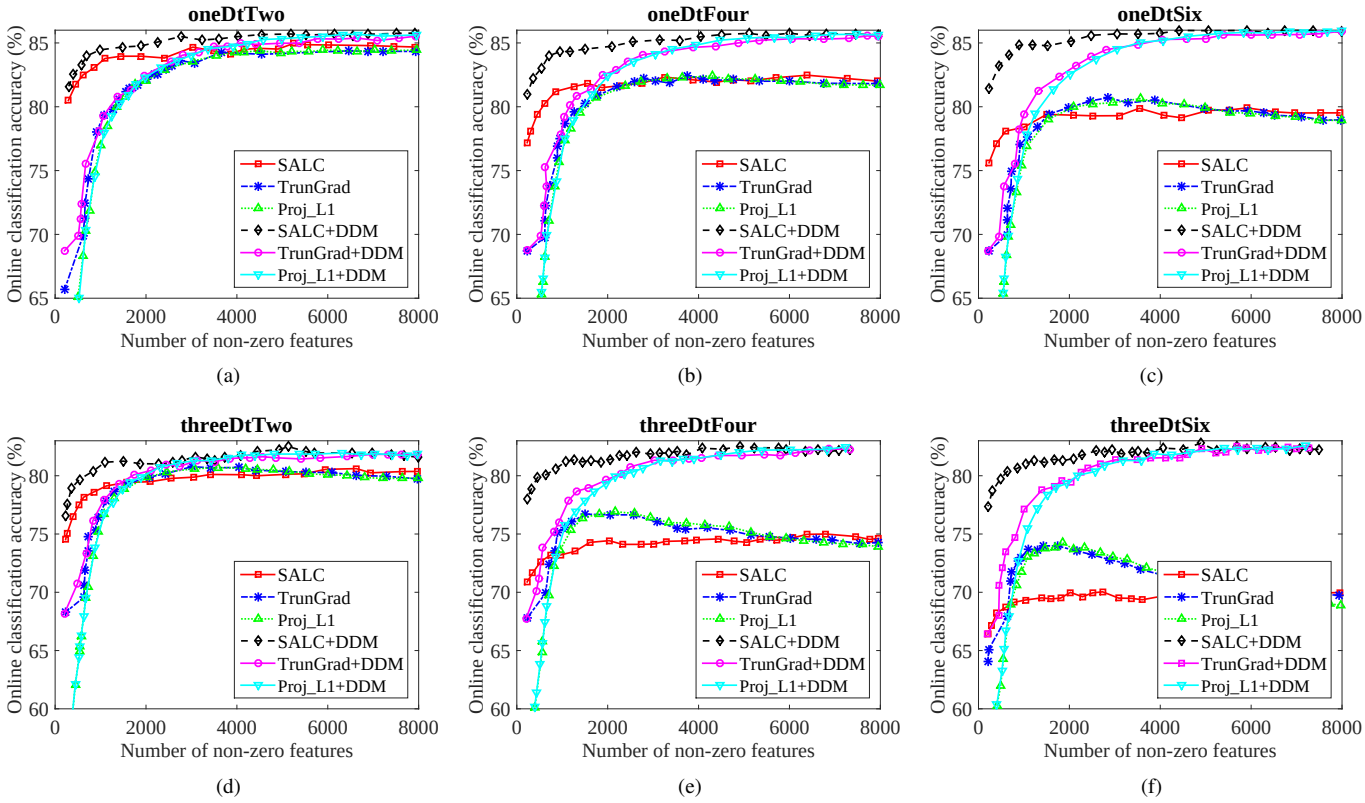

Fig. 5. Online classification performance against the number of non-zero features on the datasets with $\mathcal{S}_T > 0$.

rameter $\lambda = 0.01$ and step-size $\eta = 0.01$; SALC also set $\kappa = 1$; we varied the sparsity parameter ρ in SALC to obtain a solution with various sparsity level and let OFS select the exact number of features as SALC did. Fig. 6 presents the online classification accuracy when the number of non-zero features kept is less than 2000. Note that as the number of features kept continues to grow, the gap in online accuracy between both algorithms will disappear gradually. From Fig. 6, we can clearly see that when the amount of drifts is smaller, SALC is slightly better than OFS, however, as the amount of drifts increases, the difference in online accuracy is growing. These observations demonstrate the superiority of SALC over OFS in dynamic environments.

VI. Related Work

Our study is closely related to online linear classification and regret metrics in static and dynamic environments. Next we review important related works in these research areas.

Online linear classification According to the information used during the updating, online linear classifiers can be divided as first-order and second-order methods. First-order methods only use the sub-gradient information, such as [11]–[14], [30], which have low updating cost and low memory requirement and thus are very suitable for large-scale applications. In spite of the slow convergence rate, first-order methods can yield good generalization performance in classification tasks [15]. Second-order methods, such as [16], [31], [32], exploit the second derivation information of the objective function, so they have expensive updating cost and memory requirement but can often achieve faster convergence rate. Despite extensive research in online linear classifiers, most of studies only focus on stationary environments where an optimal fixed classifier with hindsight is pursued. In contrast, our work also focuses on non-stationary environments where an optimal classifier is allowed to change with time gradually, which is also a more challenging problem than traditional online classification.

Static and dynamic regrets The main performance metric for traditional online learning is the *static regret*, defined as the difference between the cumulated loss of the algorithm and that of the best fixed predictor, chosen from a given hypothesis class to minimize the cumulated loss over all trials of learning. Formally, the static regret of the algorithm over T trials is defined as

$$\text{Regret}^T \triangleq \sum_{t=1}^{T} f_t(\boldsymbol{w}_t) - \min_{\boldsymbol{u}} \sum_{t=1}^{T} f_t(\boldsymbol{u}).$$

Yet, in changing environments, the static regret is no longer appropriate for assessing the performance of online classifiers, since the best predictor is not constant, but is susceptible to evolve over time. We must therefore turn to a more general notion of *dynamic regret*. So far, two dynamic regret metrics have been proposed or adopted, including *shifting regret* [24], [26] and *adaptive regret* [33]–[35]. *Shifting regret* [24], [26] measures the cumulated loss of the algorithm against an arbitrary time-varying sequence of comparators in a given hypothesis class. Such metric is valuable for evaluating how the algorithm detects and adapts to concept drifting. *Adaptive*

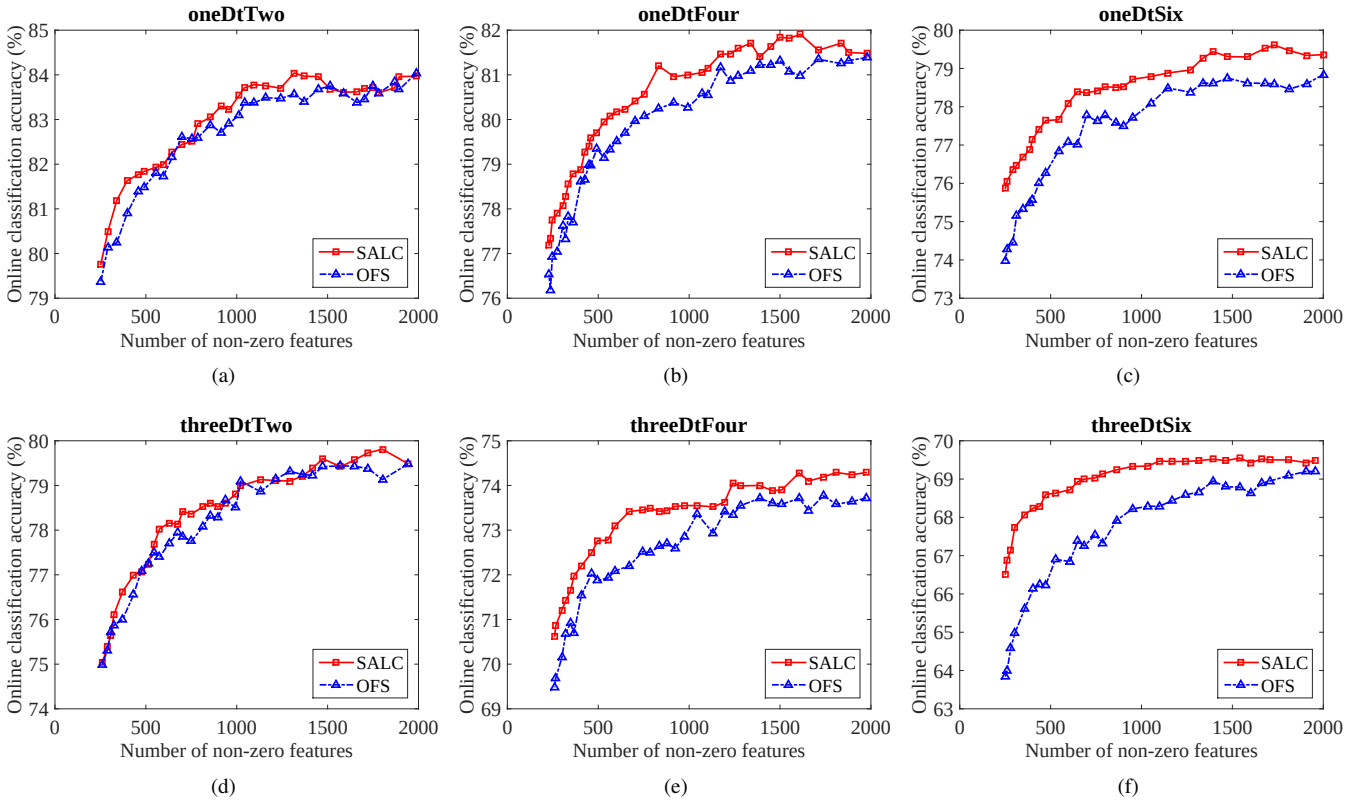

Fig. 6. Online classification performance with different number of features on the non-stationary datasets with $\mathcal{S}_T > 0$.

regret [33] is defined as the maximum regret the algorithm achieves over any contiguous time intervals. Formally,

$$\text{Regret}^T \triangleq \max_{[r,s]\in[1,T]} \left\{ \sum_{t=r}^{s} f_t(\boldsymbol{w}_t) - \min_{\boldsymbol{u}} \sum_{t=r}^{s} f_t(\boldsymbol{u}) \right\}.$$

Since the optimal predictor varies with the time interval, adaptive regret actually evaluates how well the algorithm approximates the best predictor locally. Shifting regret and adaptive regret are closely related and the relationship between the two has been discussed in [24].

VII. Conclusion

This paper extends the convergence analysis of GD-LC to changing environments and establishes its shifting regret bounds under different step-size choices. The derived bounds suggest that the step-size choice plays an important part in the tracking performance of GD-LC. Specifically, the step-size $\eta_t = 1/(\lambda t)$ helps GD-LC to achieve the least regret, given enough instances in the stationary environments, while the step-size $\eta_t = \eta$ helps GD-LC to adapt to the dynamics in the non-stationary environments. Experimental results are shown to be consistent with the theoretical results.

According to the above theoretical results, a novel sparse approximated linear classifier, called SALC, is proposed. SALC adopts the constant step-size scheduling so as to achieve good tracking performance. SALC achieves sparsity by rounding small model coefficients to zero at each online round, and controls the truncation error in a very smart way. By controlling the value of the sparsity parameter in SALC, the sparsity level of the solution can change continuously from no sparsity to total sparsity. We also provide the shifting regret bound of SALC. Experiments have shown that SALC can keep discriminative features and meanwhile does not hurt the learning performance much; in some cases, SALC even improves the generalization by reducing noisy features. We also demonstrate the superiority of SALC compared with the state-of-the-art sparse online classifiers on nine stationary datasets. On seven groups of non-stationary datasets with various shifting amount, SALC presents good tracking performance when the shifting amount is not that large; however, when wrapped with a drift detector, SALC outperforms all the other algorithms regardless of the shifting amount. We are further working on extending our sparsity strategy to online kernel learning.

Appendix A
Proof of Lemma 1

Proof.

$$\begin{aligned} D_t &= ||\boldsymbol{w}_t - \boldsymbol{u}_t||^2 - ||\boldsymbol{w}_t - \eta_t \nabla f_t(\boldsymbol{w}_t) - \boldsymbol{u}_t||^2 \\ &= 2\eta_t (\boldsymbol{w}_t - \boldsymbol{u}_t)^\top \nabla f_t(\boldsymbol{w}_t) - \eta_t^2 ||\nabla f_t(\boldsymbol{w}_t)||^2 \\ &\geq 2\eta_t (\boldsymbol{w}_t - \boldsymbol{u}_t)^\top \nabla f_t(\boldsymbol{w}_t) - \eta_t^2 G^2. \end{aligned}$$

Rearranging the terms, we get

$$(\boldsymbol{w}_t - \boldsymbol{u}_t)^\top \nabla f_t(\boldsymbol{w}_t) \leq \frac{D_t}{2\eta_t} + \frac{\eta_t G^2}{2}.$$

Given that f_t is a λ-strongly convex function, we obtain

$$(\boldsymbol{w}_t - \boldsymbol{u}_t)^\top \nabla f_t(\boldsymbol{w}_t) \geq f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}_t) + \frac{\lambda}{2} ||\boldsymbol{w}_t - \boldsymbol{u}_t||^2.$$

Combining the above two inequalities and then summing over t, we get the stated inequality. □

APPENDIX B
PROOF OF THEOREM 1

Proof. The pre-conditions in Lemma 1 hold for GD-LC, thus the inequality stated in Lemma 1 holds. Now, for three different step-size schedulings we prove Theorem 1 respectively.

Case 1: $\eta_t = 1/(\lambda t)$.

Substituting $\eta_t = 1/(\lambda t)$ and $G = 2R$ into the inequality in Lemma 1, we obtain

$$\begin{aligned}\text{Regret}^T_{\text{GD-LC}} &\le \frac{\lambda}{2}\sum_{t=1}^{T}\left((t-1)||\boldsymbol{w}_t-\boldsymbol{u}_t||^2 - t||\boldsymbol{w}_{t+1}-\boldsymbol{u}_t||^2\right) \\ &\quad + \frac{2R^2}{\lambda}\sum_{t=1}^{T}\frac{1}{t} \\ &= \frac{\lambda}{2}\sum_{t=1}^{T}((t-1)||\boldsymbol{w}_t||^2 - t||\boldsymbol{w}_{t+1}||^2) - \frac{\lambda}{2}\sum_{t=1}^{T}||\boldsymbol{u}_t||^2 \\ &\quad + \lambda\sum_{t=1}^{T}(t\boldsymbol{w}_{t+1}^\top\boldsymbol{u}_t - (t-1)\boldsymbol{w}_t^\top\boldsymbol{u}_t) + \frac{2R^2}{\lambda}\sum_{t=1}^{T}\frac{1}{t} \\ &= -\frac{\lambda T}{2}||\boldsymbol{w}_{T+1}||^2 - \frac{\lambda}{2}\sum_{t=1}^{T}||\boldsymbol{u}_t||^2 + \lambda T\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T \\ &\quad + \lambda\sum_{t=1}^{T-1}t\boldsymbol{w}_{t+1}^\top(\boldsymbol{u}_t-\boldsymbol{u}_{t+1}) + \frac{2R^2}{\lambda}\sum_{t=1}^{T}\frac{1}{t} \\ &\le -\frac{\lambda T}{2}||\boldsymbol{w}_{T+1}||^2 - \frac{\lambda}{2}\sum_{t=1}^{T}||\boldsymbol{u}_t||^2 + \lambda T\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T \\ &\quad + R\sum_{t=1}^{T-1}t||\boldsymbol{u}_t-\boldsymbol{u}_{t+1}|| + \frac{2R^2}{\lambda}(1+\ln T).\end{aligned}$$

The last inequality is due to (1) Cauchy-Schwarz inequality $\boldsymbol{a}^\top\boldsymbol{b} \le ||\boldsymbol{a}||\cdot||\boldsymbol{b}||$ for any $\boldsymbol{a}$ and $\boldsymbol{b}$, (2) the fact that $||\boldsymbol{w}_{t+1}|| \le R/\lambda$ and (3) the fact that $\sum_{t=1}^{T}\frac{1}{t} \le 1+\ln T$.

Furthermore, we have

$$\begin{aligned}&-\frac{\lambda T}{2}||\boldsymbol{w}_{T+1}||^2 - \frac{\lambda}{2}\sum_{t=1}^{T}||\boldsymbol{u}_t||^2 + \lambda T\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T \\ \le_1 &-\frac{\lambda T}{2}\left(||\boldsymbol{w}_{T+1}||^2 + ||\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t||^2\right) + \lambda T\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T \\ \le_2 &-\lambda T||\boldsymbol{w}_{T+1}||\cdot||\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t|| + \lambda T||\boldsymbol{w}_{T+1}||\cdot||\boldsymbol{u}_T|| \\ = &\ \lambda T||\boldsymbol{w}_{T+1}||\cdot(||\boldsymbol{u}_T|| - ||\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t||) \\ \le_3 &\ R\sum_{t=1}^{T-1}t||\boldsymbol{u}_t-\boldsymbol{u}_{t+1}||,\end{aligned}$$

where $\le_1$ is due to Jensen's inequality

$$\frac{1}{T}\sum_{t=1}^{T}||\boldsymbol{u}_t||^2 \ge ||\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t||^2,$$

$\le_2$ is due to $||\boldsymbol{a}||^2+||\boldsymbol{b}||^2 \ge 2||\boldsymbol{a}||\cdot||\boldsymbol{b}||$ and $\boldsymbol{a}^\top\boldsymbol{b} \le ||\boldsymbol{a}||\cdot||\boldsymbol{b}||$, and $\le_3$ is due to $||\boldsymbol{w}_{t+1}|| \le R/\lambda$ and

$$\begin{aligned}||\boldsymbol{u}_T|| - ||\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t|| &\le ||\boldsymbol{u}_T - \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}_t|| \le \frac{1}{T}\sum_{t=1}^{T}||\boldsymbol{u}_T-\boldsymbol{u}_t|| \\ &\le \frac{1}{T}\sum_{t=1}^{T}\sum_{i=t}^{T-1}||\boldsymbol{u}_i-\boldsymbol{u}_{i+1}|| = \frac{1}{T}\sum_{t=1}^{T-1}t||\boldsymbol{u}_t-\boldsymbol{u}_{t+1}||.\end{aligned}$$

Combing the above inequalities, we finally get

$$\begin{aligned}\text{Regret}^T_{\text{GD-LC}} &\le 2R\sum_{t=1}^{T-1}t||\boldsymbol{u}_t-\boldsymbol{u}_{t+1}|| + \frac{2R^2}{\lambda}(1+\ln T) \\ &\le 2RT\mathcal{S}_T + \frac{2R^2}{\lambda}(1+\ln T).\end{aligned}$$

Case 2: $\eta_t = \eta/\sqrt{t}$.

Substituting $\eta_t = \eta/\sqrt{t}$ and $G = 2R$ into the inequality in Lemma 1, we have

$$\begin{aligned}\text{Regret}^T_{\text{GD-LC}} &\le \frac{1}{2\eta}\sum_{t=1}^{T}\sqrt{t}(||\boldsymbol{w}_t-\boldsymbol{u}_t||^2 - ||\boldsymbol{w}_{t+1}-\boldsymbol{u}_t||^2) \\ &\quad + 2\eta R^2\sum_{t=1}^{T}\frac{1}{\sqrt{t}}.\end{aligned}$$

The first sum can be expanded and bounded as

$$\begin{aligned}&\frac{1}{2\eta}\sum_{t=1}^{T}\sqrt{t}(||\boldsymbol{w}_t||^2 - ||\boldsymbol{w}_{t+1}||^2) + \frac{1}{\eta}\sum_{t=1}^{T}\sqrt{t}(\boldsymbol{w}_{t+1}^\top\boldsymbol{u}_t - \boldsymbol{w}_t^\top\boldsymbol{u}_t) \\ \le &\frac{\sqrt{T}}{2\eta}\sum_{t=1}^{T}(||\boldsymbol{w}_t||^2 - ||\boldsymbol{w}_{t+1}||^2) + \frac{\sqrt{T}}{\eta}\sum_{t=1}^{T}(\boldsymbol{w}_{t+1}^\top\boldsymbol{u}_t - \boldsymbol{w}_t^\top\boldsymbol{u}_t) \\ = &-\frac{\sqrt{T}}{2\eta}||\boldsymbol{w}_{T+1}||^2 + \frac{\sqrt{T}}{\eta}\left(\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T + \sum_{t=1}^{T-1}\boldsymbol{w}_{t+1}^\top(\boldsymbol{u}_t-\boldsymbol{u}_{t+1})\right) \\ \le &\frac{R\sqrt{T}}{\lambda\eta}(U + \sum_{t=1}^{T-1}||\boldsymbol{u}_t-\boldsymbol{u}_{t+1}||).\end{aligned}$$

The second sum can be bounded as

$$\begin{aligned}2\eta R^2\sum_{t=1}^{T}\frac{1}{\sqrt{t}} &\le 2\eta R^2(1+\int_1^T\frac{1}{\sqrt{x}}dx) \\ &= 2\eta R^2(2\sqrt{T}-1) < 4\eta R^2\sqrt{T}.\end{aligned}$$

Combining the two sums, we obtain

$$\text{Regret}^T_{\text{GD-LC}} \le \frac{R\sqrt{T}}{\lambda\eta}(\mathcal{S}_T + U) + 4\eta R^2\sqrt{T}.$$

Case 3: $\eta_t = \eta$.

Substituting $\eta_t = \eta$ and $G = 2R$ into the inequality in Lemma 1, we get

$$\text{Regret}^T_{\text{GD-LC}} \le \frac{1}{2\eta}\sum_{t=1}^{T}\left(||\boldsymbol{w}_t-\boldsymbol{u}_t||^2 - ||\boldsymbol{w}_{t+1}-\boldsymbol{u}_t||^2\right) + 2\eta R^2 T.$$

The first sum can be expanded and bounded as

$$\frac{1}{2\eta}\sum_{t=1}^{T-1}\left(||\boldsymbol{w}_t||^2-||\boldsymbol{w}_{t+1}||^2\right)+\frac{1}{\eta}\sum_{t=1}^{T}(\boldsymbol{w}_{t+1}^\top\boldsymbol{u}_t-\boldsymbol{w}_t^\top\boldsymbol{u}_t)$$
$$=-\frac{1}{2\eta}||\boldsymbol{w}_{T+1}||^2+\frac{1}{\eta}\boldsymbol{w}_{T+1}^\top\boldsymbol{u}_T+\frac{1}{\eta}\sum_{t=1}^{T-1}\boldsymbol{w}_{t+1}^\top(\boldsymbol{u}_t-\boldsymbol{u}_{t+1})$$
$$\leq\frac{R}{\lambda\eta}(\mathcal{S}_T+U).$$

Hence, we have proved that in this case

$$\text{Regret}_{\text{GD-LC}}^T\leq\frac{R}{\lambda\eta}(\mathcal{S}_T+U)+2\eta R^2T.$$

□

ACKNOWLEDGMENT

The authors would like to acknowledge support for this project from the National Key R&D Program of China (2017YFB0702600, 2017YFB0702601), the National Natural Science Foundation of China (Nos. 61432008, 61503178) and the Natural Science Foundation of Jiangsu Province of China (BK20150587) and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.
[2] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.
[3] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 27–39, 2014.
[4] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 81–94, 2014.
[5] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowl. Inf. Syst.*, vol. 46, no. 3, pp. 567–597, 2016.
[6] T. Zhai, Y. Gao, H. Wang, and L. Cao, "Classification of high-dimensional evolving data streams via a resource-efficient online ensemble," *Data Min. Knowl. Discov.*, vol. 31, no. 5, pp. 1242–1265, 2017.
[7] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.
[8] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, 2010.
[9] M. A. T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1150–1159, 2003.
[10] N. S. Rao, R. D. Nowak, C. R. Cox, and T. T. Rogers, "Classification with the sparse group lasso," *IEEE Trans. Signal Process.*, vol. 64, no. 2, pp. 448–463, 2016.
[11] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.
[12] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. ICML*, 2004, p. 116.
[13] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, 2006.
[14] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.
[15] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proc. IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.
[16] K. Crammer, M. Dredze, and F. Pereira, "Confidence-weighted linear classification for text categorization," *J. Mach. Learn. Res.*, vol. 13, pp. 1891–1926, 2012.
[17] J. C. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the ℓ_1-ball for learning in high dimensions," in *Proc. ICML*, 2008, pp. 272–279.
[18] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *Math. Program.*, vol. 158, no. 1-2, pp. 575–585, 2016.
[19] S. Shalev-Shwartz, N. Srebro, and T. Zhang, "Trading accuracy for sparsity in optimization problems with sparsity constraints," *SIAM J. Optim.*, vol. 20, no. 6, pp. 2807–2832, 2010.
[20] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Mach. Learn. Res.*, vol. 10, pp. 2899–2934, 2009.
[21] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. Mach. Learn. Res.*, vol. 10, pp. 777–801, 2009.
[22] J. Wang, P. Zhao, S. C. Hoi, and R. Jin, "Online feature selection and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 698–710, 2014.
[23] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, no. 2-3, pp. 143–167, 2007.
[24] N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz, "A new look at shifting regret," *CoRR abs/1202.3323*, 2012.
[25] E. C. Hall and R. Willett, "Dynamical models and tracking regret in online convex programming," in *Proc. ICML*, 2013, pp. 579–587.
[26] A. György and C. Szepesvári, "Shifting regret, mirror descent, and matrices," in *Proc. ICML*, 2016, pp. 2943–2951.
[27] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, 2012.
[28] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.
[29] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *Proc. SBIA*, 2004, pp. 286–295.
[30] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.
[31] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proc. ICML*, 2008, pp. 264–271.
[32] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *J. Mach. Learn. Res.*, vol. 10, pp. 1737–1754, 2009.
[33] E. Hazan and C. Seshadhri, "Efficient learning algorithms for changing environments," in *Proc. ICML*, 2009, pp. 393–400.
[34] D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk, "A closer look at adaptive regret," *J. Mach. Learn. Res.*, vol. 17, no. 23, pp. 1–21, 2016.
[35] A. Daniely, A. Gonen, and S. Shalev-Shwartz, "Strongly adaptive online learning," in *Proc. ICML*, 2015, pp. 1405–1411.

Tingting Zhai is a Ph.D. candidate with the State Key Laboratory for Novel Software Technology, Nanjing University, China. Her current research interests include online learning, data stream mining and stochastic optimization.

Frédéric Koriche received his Habilitation thesis in Computer Science from University of Montpellier in 2010. Currently, he is a professor at the CRIL lab (UMR CNRS 8188), Université d'Artois, Lens, France. His research interests include learning and inference algorithms in combinatorial domains, and graphical models.

Hao Wang received the Ph.D. degree from the Department of Computer Science, the University of Hong Kong (HKU), China, in 2014. Currently, he is an assistant researcher at the Department of Computer Science and Technology, Nanjing University. His research interests include reinforcement learning, transfer learning, recommender systems and rank-aware query processing.

Yang Gao received the Ph.D. degree in Computer Software and Theory from the Department of Computer Science and Technology, Nanjing University, China, in 2000. He is a professor at the Department of Computer Science and Technology, Nanjing University. His research interests include artificial intelligence and machine learning. Prof. Gao has published more than 100 papers in top conferences and journals in and out of China.