Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Efficient Online Learning for Large-Scale Sparse Kernel Logistic Regression

Lijun Zhang
zljzju@zju.edu.cn
Zhejiang Provincial Key Laboratory of Service Robot
College of Computer Science
Zhejiang University, Hangzhou 310027, China

Rong Jin
rongjin@cse.msu.edu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA

Chun Chen and **Jiajun Bu**
{chenc,bjj}@zju.edu.cn
Zhejiang Provincial Key Laboratory of Service Robot
College of Computer Science
Zhejiang University, Hangzhou 310027, China

Xiaofei He
xiaofeihe@cad.zju.edu.cn
State Key Lab of CAD&CG
College of Computer Science
Zhejiang University, Hangzhou 310058, China

Abstract

In this paper, we study the problem of large-scale Kernel Logistic Regression (KLR). A straightforward approach is to apply stochastic approximation to KLR. We refer to this approach as *non-conservative online learning* algorithm because it updates the kernel classifier after every received training example, leading to a dense classifier. To improve the sparsity of the KLR classifier, we propose two conservative online learning algorithms that update the classifier in a stochastic manner and generate sparse solutions. With appropriately designed updating strategies, our analysis shows that the two conservative algorithms enjoy similar theoretical guarantee as that of the non-conservative algorithm. Empirical studies on several benchmark data sets demonstrate that compared to batch-mode algorithms for KLR, the proposed conservative online learning algorithms are able to produce sparse KLR classifiers, and achieve similar classification accuracy but with significantly shorter training time. Furthermore, both the sparsity and classification accuracy of our methods are comparable to those of the online kernel SVM.

Introduction

Compared to other kernel methods, such as kernel Support Vector Machine (SVM) (Burges 1998), Kernel Logistic Regression(KLR) (Jaakkola and Haussler 1999; Roth 2001) is advantageous in that it outputs posterior probabilities in addition to the classification decision, and it is able to handle multi-class problems naturally. In the past, KLR has been successfully applied to several domains, such as cancer diagnosis (Koo et al. 2006) and speaker identification (Yamada, Sugiyama, and Matsui 2010).

Due to the data explosion in recent years, there has been an increasing demand of applying logistic regression to large data sets (Keerthi et al. 2005). The key challenge in developing efficient algorithms for large-scale KLR is that since negative log-likelihood is used as the loss function in KLR, the resulting kernel classifier is inherently non-sparse, leading to high computational cost in both training and testing. Although several methods have been proposed for large-scale KLR (Jaakkola and Haussler 1999; Keerthi et al. 2005; Shalev-Shwartz, Singer, and Srebro 2007), none of them addresses this challenge. To the best of our knowledge, (Zhu and Hastie 2001) was the only effort that aims to obtain sparse KLR. It proposed an algorithm, named Import Vector Machine (IVM), that constructs the kernel classifier using a fraction of training examples. However, since it is computationally expensive to identify the import vectors, IVM is generally impractical for large data sets.

In this paper, we address the challenge of large-scale sparse KLR by developing *conservative* online learning algorithms. Unlike a non-conservative online learning algorithm (Crammer and Singer 2003) that updates the classifier for every received training example, the conservative approaches will update the classifier only for a subset of training examples, leading to sparse kernel classifiers and consequently high computational efficiency for both training and testing. Specifically, for each received training example, we introduce a Bernoulli random variable to decide whether the current classifier should be updated. By appropriately choosing the probability distribution of the Bernoulli random variable, the conservative algorithms tend to update the classifier only when the loss is large. Our analysis shows that despite the stochastic updating, the conservative algorithms enjoy similar theoretical guarantee as the non-conservative algorithm. Empirical studies also confirm both the efficiency and effectiveness of the proposed methods for sparse KLR.

Related Work

We will first review the related work on kernel logistic regression, and then the developments in online learning.

Kernel Logistic Regression (KLR)

Let $\mathcal{H}_\kappa$ be a reproducing kernel Hilbert space (RKHS) endowed with kernel function $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. KLR

aims to learn a function $f \in \mathcal{H}_\kappa$ such that the posterior probability for any $\mathbf{x} \in \mathbb{R}^d$ to take the class $y \in \{1, -1\}$, is computed as

$$p(y|f(\mathbf{x})) = 1/\big(1 + \exp(-yf(\mathbf{x}))\big).$$

Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the set of training data. The regularized optimization problem of KLR is given by:

$$\min_{f \in \mathcal{H}_\kappa} \quad \frac{\lambda}{2}|f|^2_{\mathcal{H}_\kappa} + \frac{1}{n}\sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)), \tag{1}$$

where $\ell(z) = \ln(1 + e^{-z})$ measures the negative log-likelihood of a data point. Besides the primal problem in Eq. (1), several studies (Jaakkola and Haussler 1999; Keerthi et al. 2005) consider the dual problem of KLR.

The key challenge of learning a KLR model is that it uses the negative of log-likelihood as the loss function, which inadvertently leads to non-sparse kernel classifiers regardless of the optimization methods. Import Vector Machine (IVM) (Zhu and Hastie 2001) aims to reduce the complexity of the kernel classifier by selecting part of training examples as the import vectors to construct the classifier. However, the computational cost of selecting q import vectors is $O(n^2q^2)$, making it impractical for large data sets.

Online Learning

Since the invention of Perceptron (Agmon 1954; Rosenblatt 1958; Novikoff 1962), tremendous progress has been made in online learning. Early studies focused on the linear classification model, and were later extended to nonlinear classifier by using the kernel trick (Freund and Schapire 1999; Kivinen, Smola, and Williamson 2004). Inspired by the success of maximum margin classifiers, various algorithms have been proposed to incorporate classification margin into online learning (Gentile 2001; Li and Long 2002; Crammer and Singer 2003). Many recent studies explored the optimization theory for online learning (Kivinen and Warmuth 1997; Zinkevich 2003; Shalev-Shwartz and Singer 2006). More detailed discussion can be found in (Cesa-Bianchi and Lugosi 2006).

Our work is closely related to budget online learning (Cavallanti, Cesa-Bianchi, and Gentile 2007; Dekel, Shalev-Shwartz, and Singer 2008; Orabona, Keshet, and Caputo 2008) that aims to build a kernel classifier with a constrained number of support vectors. Unlike these studies that focus on the hinge loss, we focus on the negative of log-likelihood and kernel logistic regression. In addition, we apply stochastic updating to improve the sparsity of kernel classifiers while most of the budget online learning algorithms deploy deterministic strategies for updating. Finally, we distinguish our work from sparse online learning. Although sparse online learning (Langford, Li, and Zhang 2009) also aim to produce sparse classifiers, they focus on linear classification and sparse regularizers such as the ℓ_1 regularizer are used to select features not training examples.

Online Learning for KLR

In this study, we consider the constrained KLR

$$\min_{f \in \Omega} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)), \tag{2}$$

where $\ell(z) = \ln(1 + e^{-z})$ is the logit loss, $\Omega = \{f \in \mathcal{H}_\kappa : |f|_{\mathcal{H}_\kappa} \leq R\}$ and R specifies the maximum function norm for the classifier f. Throughout the paper, we assume $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any $\mathbf{x} \in \mathbb{R}^d$.

Non-conservative Online Learning for KLR (NC)

As a starting point, we apply the stochastic gradient descent approach (Kivinen, Smola, and Williamson 2004) to the problem in Eq. (2). At each iteration of gradient descent, given a training example $(\mathbf{x}_t, y_t)$, we update the current classifier $f_t(\mathbf{x})$ by

$$f_{t+1}(\mathbf{x}) = f_t(\mathbf{x}) - \eta \nabla_f \ell(y_t f_t(\mathbf{x}_t)), \tag{3}$$

where η is the step size. ∇_f denotes the gradient with respect to function f and is given by

$$\nabla_f \ell(y_t f_t(\mathbf{x}_t)) = y_t \ell'(y_t f_t(\mathbf{x}_t))\kappa(\mathbf{x}_t, \cdot), \tag{4}$$

where $\ell'(y_t f_t(\mathbf{x}_t)) = p(y_t|f_t(\mathbf{x}_t)) - 1$.

Algorithm 1 shows the detailed procedure. We follow (Crammer and Singer 2003) and call it non-conservative online learning algorithm for KLR, or **NC** for short. At step 7 of Algorithm 1, we use the notation $\pi_\Omega(f)$ to represent the projection of f into the domain Ω, which is calculated as $f' = \pi_\Omega(f) = \frac{R}{\max(R, |f|_{\mathcal{H}_\kappa})} f$. One of the key disadvantages of Algorithm 1 is that it updates the classifier $f(\cdot)$ after receiving every example, leading to a very dense classifier.

Define the generalization error $\overline{\ell}(f)$ for a kernel classifier f as

$$\overline{\ell}(f) = \mathrm{E}_{(\mathbf{x},y)}[\ell(yf(\mathbf{x}))],$$

where $\mathrm{E}_{(\mathbf{x},y)}[\cdot]$ takes the expectation over the unknown distribution $P(\mathbf{x}, y)$. We denote by f^* the optimal solution within domain Ω that minimizes the generalization error, i.e.,

$$f^* = \min_{f \in \Omega} \overline{\ell}(f).$$

Theorem 1. *After running Algorithm 1 over a sequence of iid sampled training examples $\{(\mathbf{x}_t, y_t), t = 1, \ldots, T\}$, we have the the following bound for $\widehat{f}$*

$$\mathrm{E}_T[\overline{\ell}(\widehat{f})] \leq \overline{\ell}(f^*) + \frac{R^2}{2\eta T} + \frac{\eta}{2},$$

where $\mathrm{E}_T[\cdot]$ takes expectation over the sequence of examples $\{(\mathbf{x}_t, y_t), t = 1, \ldots, T\}$. By choosing $\eta = R/\sqrt{T}$, we have

$$\mathrm{E}_T[\overline{\ell}(\widehat{f})] \leq \overline{\ell}(f^*) + \frac{R}{\sqrt{T}}.$$

We skip the proof because it follows the standard analysis and can be found in (Cesa-Bianchi and Lugosi 2006).

The above theorem shows that the difference in prediction between the learned classifier $\widehat{f}$ and the optimal classifier f^*

Algorithm 1 A non-conservative online learning algorithm for large-scale KLR (**NC**)

Input: the step size $\eta > 0$
Initialization: $f_1(\mathbf{x}) = 0$
for $t = 1, 2, \ldots, T$ **do**
  Receive an example $(\mathbf{x}_t, y_t)$
  $\ell'(y_t f_t(\mathbf{x}_t)) = p(y_t|f_t(x_t)) - 1$
  $f'_{t+1}(\mathbf{x}) = f_t(\mathbf{x}) - \eta y_t \ell'(y_t f_t(\mathbf{x}_t))\kappa(\mathbf{x}_t, \mathbf{x})$
  $f_{t+1} = \pi_\Omega(f'_{t+1})$
end for
Output: $\widehat{f}(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x})/T$

decreases at the rate of $O(1/\sqrt{T})$. The next theorem shows that the difference in prediction could decrease at the rate of $O(1/T)$, but at the price that we will compare $\overline{\ell}(\widehat{f})$ to $2/(2-\eta)\overline{\ell}(f^*)$, instead of $\overline{\ell}(f^*)$. This result indicates that the generalization error may be reduced fast when the corresponding classification model is easy to learn, i.e., $\overline{\ell}(f^*)$ is small.

Theorem 2. *Repeat the conditions of Theorem 1. We have*

$$\mathrm{E}_T[\overline{\ell}(\widehat{f})] \leq \frac{2}{2-\eta}\overline{\ell}(f^*) + \frac{R^2}{\eta(2-\eta)T}.$$

Proof. First, according to (Cesa-Bianchi and Lugosi 2006), we have

$$\begin{aligned}&\ell(y_t f_t(\mathbf{x}_t)) - \ell(y_t f(\mathbf{x}_t))\\ \leq&\frac{|f - f_t|^2_{\mathcal{H}_\kappa} - |f - f_{t+1}|^2_{\mathcal{H}_\kappa}}{2\eta} + \frac{\eta(1 - p(y_t|f_t(\mathbf{x}_t))^2}{2}.\end{aligned}$$

Since

$$\begin{aligned}\ell(y_t f_t(\mathbf{x}_t)) &= -\ln p(y_t|f_t(\mathbf{x}_t))\\ &\geq 1 - p(y_t|f_t(\mathbf{x}_t)) \geq (1 - p(y_t|f_t(\mathbf{x}_t))^2,\end{aligned}$$

we have

$$\begin{aligned}&\ell(y_t f_t(\mathbf{x}_t))\\ \leq&\frac{2\ell(y_t f(\mathbf{x}_t))}{2-\eta} + \frac{1}{\eta(2-\eta)}\left(|f - f_t|^2_{\mathcal{H}_\kappa} - |f - f_{t+1}|^2_{\mathcal{H}_\kappa}\right).\end{aligned}$$

We complete the proof by following the standard analysis of online learning. □

Conservative Online Learning for KLR

To overcome the drawback of Algorithm 1, we developed online learning algorithms with conservative updating strategies that aims to improve the sparsity of kernel classifier. The simplest approach is to update $f(\mathbf{x})$ only when it misclassifies a training example, a common strategy employed by many online learning algorithms (Cesa-Bianchi and Lugosi 2006). We note that in most previous studies, this simple strategy is used to derive mistake bound, not the generalization error bound.

Instead of having a deterministic procedure, we propose a stochastic procedure to decide if the classifier should be updated for a given example. In particular, after receiving the training example $(\mathbf{x}_t, y_t)$, we introduce a Bernoulli random variable Z_t with probability p_t being 1; the classifier will be updated only when $Z_t = 1$. Below, we introduce two types of approaches for stochastic updates, one based on the classification margin and the other based on the auxiliary function.

Algorithm 2 A classification margin based conservative algorithm for large-scale KLR (**Margin**)

Input: the step size $\eta > 0$
Initialization: $f_1(\mathbf{x}) = 0$
for $t = 1, 2, \ldots, T$ **do**
  Receive an example $(\mathbf{x}_t, y_t)$
  Compute probability p_t in Eq. (5)
  Sample Z_t from a Bernoulli distribution with probability p_t being 1
  if $Z_t = 1$ **then**
    $\ell'(y_t f_t(\mathbf{x}_t)) = p(y_t|f_t(x_t)) - 1$
    $f'_{t+1}(\mathbf{x}) = f_t(\mathbf{x}) - \eta y_t \ell'(y_t f_t(\mathbf{x}_t))\kappa(\mathbf{x}_t, \mathbf{x})$
    $f_{t+1}(\mathbf{x}) = \pi_\Omega(f'_{t+1})$
  else
    $f_{t+1} = f_t$
  end if
end for
Output: $\widehat{f}(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x})/T$

Classification margin based approach In this approach, we introduce the following sampling probability p_t for stochastically updating $f_t(\mathbf{x})$

$$p_t = \frac{2-\eta}{2-\eta+\eta p(y_t|f_t(\mathbf{x}_t))}. \tag{5}$$

It is obvious that p_t is monotonically decreasing in $p(y_t|f_t(\mathbf{x}_t))$, i.e., the larger the chance to classify the training example correctly, the smaller the sampling probability is, which is clearly consistent with our intuition. Since the $p(y_t|f_t(\mathbf{x}_t))$ is determined by the classification margin $y_t f_t(\mathbf{x}_t)$ of $\mathbf{x}_t$, we refer to this algorithm as classification margin based approach, and the details are given in Algorithm 2. It is interesting to see, although we introduce a sampling probability to decide if the classifier should be updated, the generalization error bound for Algorithm 2 remains the same as Theorem 2, as stated below.

Theorem 3. *After running Algorithm 2 over a sequence of iid sampled training examples $\{(\mathbf{x}_t, y_t), t = 1, \ldots, T\}$, we have the the following bound*

$$\mathrm{E}_T[\overline{\ell}(\widehat{f})] \leq \frac{2}{2-\eta}\overline{\ell}(f^*) + \frac{R^2}{\eta(2-\eta)T}.$$

Proof. Following the same analysis of Theorem 2, we have

$$\begin{aligned}&Z_t[\ell(y_t f_t(\mathbf{x}_t)) - \ell(y_t f(\mathbf{x}_t))] - \frac{|f - f_t|^2_{\mathcal{H}_\kappa} - |f - f_{t+1}|^2_{\mathcal{H}_\kappa}}{2\eta}\\ \leq&\frac{\eta Z_t}{2}\ell(y_t f_t(\mathbf{x}_t))(1 - p(y_t|f_t(\mathbf{x}_t))),\end{aligned}$$

and therefore

$$Z_t\ell(y_t f_t(\mathbf{x}_t))\left(1-\frac{\eta}{2}(1-p(y_t|f_t(\mathbf{x}_t)))\right)$$
$$\leq Z_t\ell(y_t f(\mathbf{x}_t))+\frac{1}{2\eta}\left(|f-f_t|^2_{\mathcal{H}_\kappa}-|f-f_{t+1}|^2_{\mathcal{H}_\kappa}\right).$$

By taking the expectation over Z_t, we obtain

$$\ell(y_t f_t(\mathbf{x}_t))-\frac{2\ell(y_t f(\mathbf{x}_t))}{2-\eta}$$
$$\leq\frac{1}{\eta(2-\eta)}\left(|f-f_t|^2_{\mathcal{H}_\kappa}-|f-f_{t+1}|^2_{\mathcal{H}_\kappa}\right).$$

We complete the proof by using the standard analysis of online learning. □

Auxiliary function based approach One drawback with the stochastic approach stated above is that the sampling probability p_t is associated with the step size η. When η is small, it will result in a sampling probability close to 1, leading to a small reduction in the number of updates. To overcome this limitation, we introduce an auxiliary function $h(z)$ that is (i) convex in z and (ii) $h(z)\geq\ell(z)$ for any z. Define the maximum gradient of $h(z)$ as $M(h)=\max_z|h'(z)|$.

Given the auxiliary function, we update the classifier $f(\mathbf{x})$ based on the gradient of the auxiliary function $h(z)$, not the gradient of the loss function $\ell(z)$. We define the sampling probability p_t as

$$p_t=\frac{\ell(y_t f_t(\mathbf{x}_t))}{h(y_t f_t(\mathbf{x}_t))}. \quad (6)$$

It is easy to verify the following theorem regarding the generalization error of the classifier found by Algorithm 3.

Theorem 4. *After running Algorithm 3 over a sequence of iid sampled training examples* $\{(\mathbf{x}_t,y_t),t=1,\ldots,T\}$*, we have*

$$\mathrm{E}_T[\bar{\ell}(\widehat{f})]\leq\min_{f\in\Omega(f)}\mathrm{E}_{(\mathbf{x},y)}[h(yf(\mathbf{x}))]+\frac{RM(h)}{\sqrt{T}}.$$

In the following, we list several possible types of auxiliary functions:

- $h(z)=\ln(\gamma+e^{-z})$, with $\gamma\geq 1$. The corresponding sampling probability p_t is computed as

$$p_t=\frac{\ln(1+\exp(-y_t f_t(\mathbf{x}_t)))}{\ln(\gamma+\exp(-y_t f_t(\mathbf{x}_t)))}.$$

  It is clear that p_t is monotonically decreasing in $y_t f_t(\mathbf{x}_t)$, and approaches zero when $y_t f_t(\mathbf{x}_t)\to+\infty$.

- $h(z)=\ln(1+\gamma e^{-z})$, with $\gamma\geq 1$. And the sampling probability p_t becomes

$$p_t=\frac{\ln(1+\exp(-y_t f_t(\mathbf{x}_t)))}{\ln(1+\gamma\exp(-y_t f_t(\mathbf{x}_t)))}.$$

  This auxiliary function is similar to the above one.

Algorithm 3 An auxiliary function based conservative algorithm for large-scale KLR (**Auxiliary**)

Input: the step size $\eta>0$ and the auxiliary function $h(z)$
Initialization: $f_1(\mathbf{x})=0$
for $t=1,2,\ldots,T$ **do**
  Receive an example $(\mathbf{x}_t,y_t)$
  Compute probability p_t in Eq. (6)
  Sample Z_t from a Bernoulli distribution with probability p_t being 1
  if $Z_t=1$ **then**
    Compute the gradient $h'(y_t f_t(\mathbf{x}_t))$
    $f'_{t+1}(\mathbf{x})=f_t(\mathbf{x})-\eta y_t h'(y_t f_t(\mathbf{x}_t))\kappa(\mathbf{x}_t,\mathbf{x})$
    $f_{t+1}=\pi_\Omega(f'_{t+1})$
  else
    $f_{t+1}=f_t$.
  end if
end for
Output: $\widehat{f}(\mathbf{x})=\sum_{t=1}^T f_t(\mathbf{x})/T$

Table 1: Data statistics

Data Sets	# of Examples	# of Features
mushrooms	8,124	112
a9a	32,561	123
ijcnn1	141,691	22
rcv1.binary	697,641	47,236

- $h(z)=\max(\ell(z),\ell(\delta))$. If $\ell(y_t f_t(\mathbf{x}_t))\geq\ell(\delta)$ (i.e., $y_t f_t(\mathbf{x}_t)\leq\delta$), the sampling probability $p_t=1$, and the gradient $h'(y_t f_t(\mathbf{x}_t))=\ell'(y_t f_t(\mathbf{x}_t))$. Otherwise,

$$p_t=\frac{\ln(1+\exp(-y_t f_t(\mathbf{x}_t)))}{\ln(1+\exp(-\delta))}.$$

  and the gradient $h'(y_t f_t(\mathbf{x}_t))=0$. Thus, using this function, our method becomes a hard cut-off algorithm, which updates the classifier only when the margin $y_t f_t(\mathbf{x}_t)$ is smaller than δ.

One advantage of the auxiliary function based approach is that it allows us to control the sparsity of the classifier. For the above auxiliary functions, the sparsity can be turned by varying the value of γ or δ.

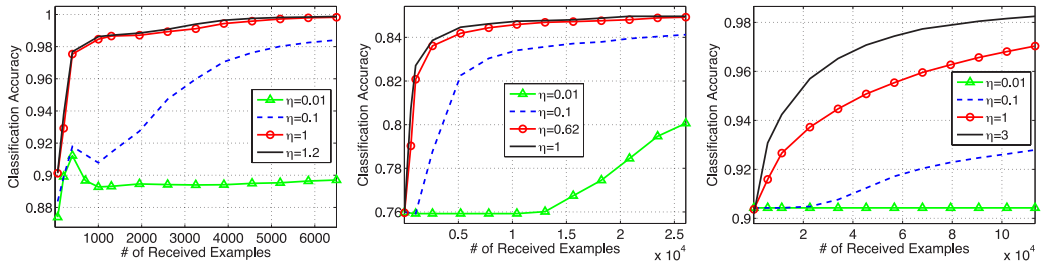
Experiments

Data Sets

Four benchmark data sets (mushrooms, a9a , ijcnn1, and rcv1.binary) from LIBSVM (Chang and Lin 2011) are used in this evaluation. Table 1 summarizes the statistics of them.

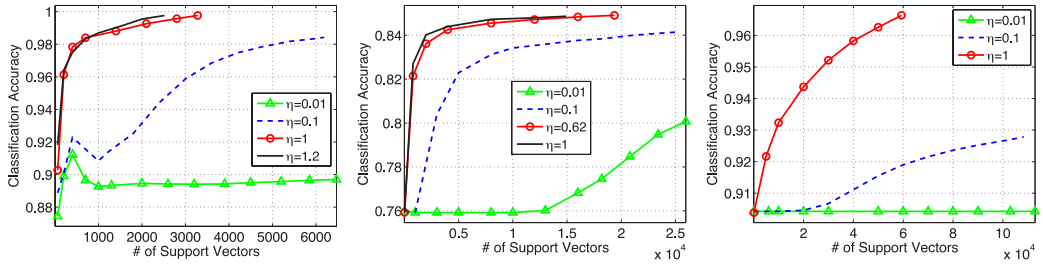
Experiments on medium-size data sets

In this section, we perform classification experiments on the first three data sets to evaluate our methods comprehensively. We choose the Gaussian kernel $\kappa(\mathbf{x}_i,\mathbf{x}_j)=\exp(\|\mathbf{x}_i-\mathbf{x}_j\|^2/(2\sigma^2))$, and set the kernel width σ to the 5-th percentile of the pairwise distances (Mallapragada, Jin,



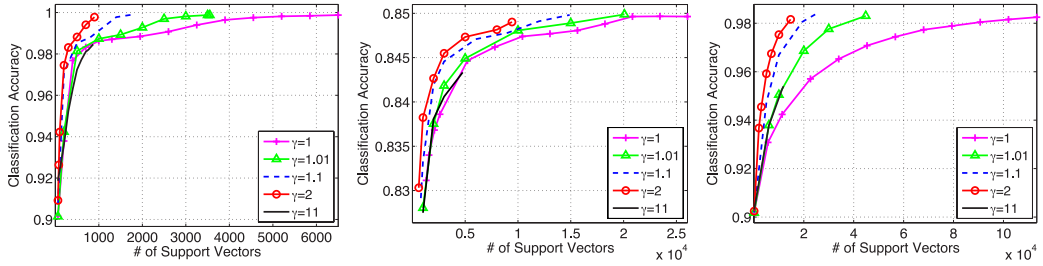
(a) The mushrooms data set. (b) The a9a data set. (c) The ijcnn1 data set.

Figure 1: Classification accuracy of the non-conservative online learning algorithm (NC) using different η.



(a) The mushrooms data set. (b) The a9a data set. (c) The ijcnn1 data set.

Figure 2: Classification accuracy of the classification margin based conservative algorithm (Margin) using different η.



(a) The mushrooms data set. (b) The a9a data set. (c) The ijcnn1 data set.

Figure 3: Classification accuracy of the auxiliary function based conservative algorithm (Auxiliary) using different γ.

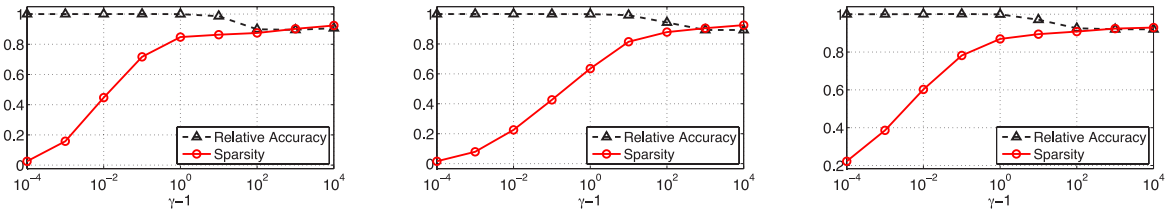
and Jain 2010). The parameters R in Eq. (2) and λ in Eq. (1) are determined by Cross Validation (CV), and searched in the range of $\{1, 1e1, \ldots, 1e5\}$ and $\{1e{-}5, 1e{-}4, \ldots, 1\}$, respectively. We perform 5-fold CV on each data set, and report the average classification accuracy.

Experimental Results for the Non-conservative Online Learning Method (NC) We evaluate the performance of NC for KLR with $\eta = \{R/\sqrt{T}, 1e{-}2, 1e{-}1, 1\}$, where T is the number of training examples. Fig. 1 shows the average classification accuracy versus the number of received training examples on the three data sets. We observe that the performance of the non-conservative method improves rapidly when the number of training examples is small, and the performance levels off after receiving enough training examples. Besides, the step size $\eta = R/\sqrt{T}$ yields good classification accuracy for all the data sets.

Experimental Results for Conservative online learning Methods For conservative methods, we refer to as *support vectors* the training examples used to update the classifier.

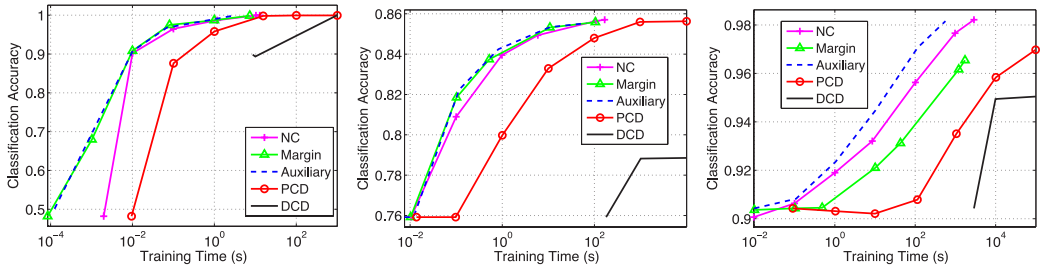
Fig. 2 shows the average classification accuracy of the classification margin based conservative algorithm (Margin) versus the number of support vectors. Compared to Fig. 1, we observe that the final classification accuracy of Margin is almost the same as that of NC, which is consistent with the generalization error bound in Theorem 3. Note that since the sampling probability defined in Eq. (5) is valid only when $\eta < 2$, there is no curve for $\eta = 3$ in Fig. 2(c). We also observe that the overall reduction in the number of support vectors is not very significant. In most cases, less than 50% of support vectors are removed by Margin.

For the auxiliary function based approach (Auxiliary), we set $h(z) = \ln(\gamma + e^{-z})$. Fig. 3 shows the average classification accuracy of this approach with different γ, where we use the best η found from Fig. 1. Note that NC is equivalent to the case of $\gamma = 1$. We observe that by choosing an appropriate γ, we can reduce the number of support vectors dramatically without sacrificing the performance. For example, for the mushrooms and ijcnn1 data sets, with $\gamma = 2$, Auxiliary is able to achieve the same performance as the non-conservative approach but with 20% of the training data as support vectors. On the other hand, the gains of this ap-



(a) The mushrooms data set. (b) The a9a data set. (c) The ijcnn1 data set.

Figure 4: The sparsity and relative accuracy of the final classifier obtained by Auxiliary versus $\gamma - 1$.



(a) The mushrooms data set. (b) The a9a data set. (c) The ijcnn1 data set.

Figure 5: Comparison between online learning methods and batch-model methods.

Table 2: Comparison between Auxiliary and Pegasos on the rcv1.binary data set

Method	Auxiliary ($h(z) = \ln(\gamma + e^{-z})$)			Pegasos		
	$\gamma = 1.01$	$\gamma = 2$	$\gamma = 101$	$\lambda = 1e{-}5$	$\lambda = 1e{-}7$	$\lambda = 1e{-}9$
# of Support Vectors	24,654	23,894	23,753	47,150	24,381	24,034
Sparsity	0.9636	0.9647	0.9649	0.9304	0.9640	0.9645
Accuracy	0.9794	0.9783	0.9792	0.9778	0.9746	0.9653
Training Time (s)	20,779	20,060	19,974	34,098	20,214	19,980

proach on the a9a data set is less obvious. This may be attributed to the fact that this data set is more difficult to be classified and therefore has a higher sampling probability.

To further show the tradeoff between the sparsity and the performance, Fig. 4 plots the sparsity and relative accuracy of the final classifier obtained by Auxiliary versus $\gamma - 1$. The sparsity is defined as the ratio between the number of non-support vectors and the number of received training examples, and the relative accuracy is computed by comparing to that of the non-conservative approach. We observe that for all the data sets, there is a large range of γ which can be used to produce a sparse and competitive classifier.

Comparison with Batch-model Methods Since there are no off-the-shelf packages available for KLR, we develop two batch-model methods based on the coordinate descent algorithms described in (Keerthi et al. 2005).

- **PCD**: the coordinate descent method for solving the primal problem of KLR.
- **DCD**: the coordinate descent method for solving the dual problem of KLR.

The best step size is used for the three online learning methods, and γ is set to 2 for the auxiliary function based approach. Fig. 5 shows the classification accuracy versus the training time for all the algorithms. We observe that the three online learning algorithms are considerably more efficient than the two batch-model algorithms (PCD and DCD). For example, Auxiliary is about 10 times faster than PCD on the ijcnn1 data set.

Experiments on one large data set

In this section, we compare our methods with Pegasos (Shalev-Shwartz, Singer, and Srebro 2007), which is the state-of-the-art online kernel SVM algorithm, on the rcv1.binary data set. Using the split provided by (Chang and Lin 2011), 677,399 samples are used for training and the rest are used for testing. We choose the polynomial kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2$, which is commonly used for text corpus (Joachims 1998). Since Auxiliary gives the best result in terms of sparsity among the three proposed algorithms, in this study, we only report the result of Auxiliary with $h(z) = \ln(\gamma + e^{-z})$. For this large data set, it is time consuming to apply CV to determine the best parameters. As an alternative, we empirically set $R = 1e5$ and $\eta = R/\sqrt{T}$ for Auxiliary, and then vary the value of γ to make the tradeoff between the sparsity and accuracy. For Pegasos, its parameter λ is varied in the range of $\{1e{-}10, \ldots, 1e{-}1\}$.

Table 2 shows the sparsity and the testing accuracy of the kernel classifier learned from the entire set of training examples, as well as the training time. For brevity, we just show the partial results around the best parameters for each method. As can be seen, with suitable parameters, both the sparsity and the accuracy of Auxiliary is comparable to those

of the Pegasos method. This is consistent with the observation made by (Keerthi et al. 2005), in which KLR and kernel SVM yield similar classification performance. Besides classification accuracy, we observe that both Pegasos and the proposed algorithm learn the kernel classifier with similar sparsity. In particular, for this data set, less than 4% of the training examples are used as the support vectors for constructing the kernel classifier. Since the sparsity is similar, the training time of the two methods is similar too.

Conclusions

In this paper, we present conservative online learning algorithms for large-scale sparse KLR. The key idea is to adopt stochastic procedures for updating the classifiers. Compared to the non-conservative approach that requires updating the classifier for every received training example, we show that the conservative algorithms enjoy similar theoretical guarantee, which is further confirmed by our empirical studies.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Grant No: 61125203, 90920303, 61173186), National Basic Research Program of China (973 Program) under Grant No. 2009CB320801, National Science Foundation (IIS-0643494), Office of Naval Research (ONR N00014-09-1-0663), Fundamental Research Funds for the Central Universities, Program for New Century Excellent Talents in University (NCET-09-0685), Zhejiang Provincial Natural Science Foundation under Grant No. Y1101043, and Scholarship Award for Excellent Doctoral Student granted by Ministry of Education.

References

Agmon, S. 1954. The relaxation method for linear inequalities. *Canadian Journal of Mathematics* 6(3):382–392.

Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2):121–167.

Cavallanti, G.; Cesa-Bianchi, N.; and Gentile, C. 2007. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning* 69(2-3):143–167.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2(3):27:1–27:27.

Crammer, K., and Singer, Y. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research* 3:951–991.

Dekel, O.; Shalev-Shwartz, S.; and Singer, Y. 2008. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing* 37(5):1342–1372.

Freund, Y., and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37(3):277–296.

Gentile, C. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research* 2:213–242.

Jaakkola, T. S., and Haussler, D. 1999. Probabilistic kernel regression models. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, 137–142.

Keerthi, S.; Duan, K.; Shevade, S.; and Poo, A. 2005. A fast dual algorithm for kernel logistic regression. *Machine Learning* 61(1-3):151–165.

Kivinen, J., and Warmuth, M. K. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation* 132(1):1–63.

Kivinen, J.; Smola, A. J.; and Williamson, R. C. 2004. Online learning with kernels. *IEEE Transactions on Signal Processing* 52(8):2165–2176.

Koo, J.-Y.; Sohn, I.; Kim, S.; and Lee, J. W. 2006. Structured polychotomous machine diagnosis of multiple cancer types using gene expression. *Bioinformatics* 22(8):950–958.

Langford, J.; Li, L.; and Zhang, T. 2009. Sparse online learning via truncated gradient. *Journal of Machine Learning Research* 10:777–801.

Li, Y., and Long, P. M. 2002. The relaxed online maximum margin algorithm. *Machine Learning* 46(1-3):361–387.

Mallapragada, P.; Jin, R.; and Jain, A. 2010. Non-parametric mixture models for clustering. In *Proceedings of the 2010 Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition*, 334–343.

Novikoff, A. 1962. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume XII, 615–622.

Orabona, F.; Keshet, J.; and Caputo, B. 2008. The projectron: a bounded kernel-based perceptron. In *Proceedings of the 25th International Conference on Machine Learning*, 720–727.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65:386–407.

Roth, V. 2001. Probabilistic discriminative kernel classifiers for multi-class problems. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, 246–253.

Shalev-Shwartz, S., and Singer, Y. 2006. Online learning meets optimization in the dual. In *Proceedings of the 19th Annual Conference on Learning Theory*, 423–437.

Shalev-Shwartz, S.; Singer, Y.; and Srebro, N. 2007. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, 807–814.

Yamada, M.; Sugiyama, M.; and Matsui, T. 2010. Semi-supervised speaker identification under covariate shift. *Signal Processing* 90(8):2353–2361.

Zhu, J., and Hastie, T. 2001. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems 13*, 1081–1088.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 928–936.