Learning Sparse Confidence-Weighted Classifier on Very High Dimensional Data

Mingkui Tan†, Yan Yan‡, Li Wang§, Anton Van Den Hengel†, Ivor W. Tsang‡, Qinfeng (Javen) Shi†
†ACVT, The University of Adelaide, Australia
‡QCIS, University of Technology Sydney, Australia
§Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, USA
†{mingkui.tan, anton.vandenhengel, javen.shi}@adelaide.edu.au, ‡{yan.yan,ivor.tsang}@uts.edu.au, §liwang8@uic.edu

Abstract

Confidence-weighted (CW) learning is a successful online learning paradigm which maintains a Gaussian distribution over classifier weights and adopts a covariance matrix to represent the uncertainties of the weight vectors. However, there are two deficiencies in existing full CW learning paradigms, these being the sensitivity to irrelevant features, and the poor scalability to high dimensional data due to the maintenance of the covariance structure. In this paper, we begin by presenting an online-batch CW learning scheme, and then present a novel paradigm to learn sparse CW classifiers. The proposed paradigm essentially identifies feature groups and naturally builds a *block diagonal covariance structure*, making it very suitable for CW learning over very high-dimensional data. Extensive experimental results demonstrate the superior performance of the proposed methods over state-of-the-art counterparts on classification and feature selection tasks.

Introduction

Online learning, which starts from the perceptron and relies on few statistical assumptions, has been successfully applied to many applications (Rosenblatt 1958; Cesa-Bianchi and Lugosi 2006; Duchi, Hazan, and Singer 2011; Crammer et al. 2006; Crammer and Singer 2003; Ma et al. 2009b). It is one of the most popular topics in machine learning community in the past decades (Yang, Jin, and Ye 2009; Wang, Zhao, and Hoi 2012; Xu et al. 2014; Zhang et al. 2015). Confidence-weighted (CW) learning (Crammer, Dredze, and Pereira 2008; Dredze, Crammer, and Pereira 2008; Wang, Zhao, and Hoi 2012), which generalizes the margin-based learning methods, has become a very powerful online learning paradigm for constructing linear classifiers.

Operating with a passive-aggressive rule (Crammer et al. 2006), CW learning receives a d-dimensional example $\mathbf{x}_i$ and its label y_i in the round i, and seeks to make the smallest possible change to the weight vector $\boldsymbol{\mu}$. Different from traditional margin-based learning methods like PA (Crammer et al. 2006), CW learning maintains a probabilistic constraint over classifier weights. Basically, the classifier weight $\mathbf{w}$ in CW learning is assumed to be drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and the label of $\mathbf{x}_i$ is assigned by $\mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i)$. In this sense, the margin M is viewed as a random variable that follows a univariate Gaussian distribution: $M \sim \mathcal{N}\left(y_i(\boldsymbol{\mu}^\top \mathbf{x}_i), \mathbf{x}_i^\top \boldsymbol{\Sigma} \mathbf{x}_i\right)$. Then the probability of a correct prediction (when $M \geq 0$) is given by $\Pr[M \geq 0] = \Pr_{\mathbf{w}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[y_i(\boldsymbol{\mu}^\top \mathbf{x}_i) \geq 0]$.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

To ensure good generalization performance, it is essential to maintain a large margin on the example. This can be achieved by ensuring that the probability of a correct prediction for a training example is no smaller than a confidence level δ (Crammer, Dredze, and Pereira 2012): $\Pr[y_i(\boldsymbol{\mu}^\top \mathbf{x}_i) \geq 0] \geq \delta$. Given Φ the cumulative function of a normal distribution and let $\phi = \Phi^{-1}(\delta)$, this probabilistic constraint can be written explicitly as

$$y_i(\boldsymbol{\mu}^\top \mathbf{x}_i) \geq \phi\sqrt{\mathbf{x}_i{}^\top \boldsymbol{\Sigma} \mathbf{x}_i}. \tag{1}$$

To update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, CW classifier tries to make the least changes to previous distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Using the KL divergence to measure the distribution difference (Dredze, Crammer, and Pereira 2008), the update can be made by

$$\begin{aligned}(\boldsymbol{\mu}_{i+1}, \boldsymbol{\Sigma}_{i+1}) &= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)),\\ &\text{s.t. } y_i(\boldsymbol{\mu}^\top \mathbf{x}_i) \geq \phi\sqrt{\mathbf{x}_i^\top \boldsymbol{\Sigma} \mathbf{x}_i},\end{aligned} \tag{2}$$

where function D_{KL} can be formulated explicitly as: $\mathrm{D}_{\mathrm{KL}} := \frac{1}{2}\log\left(\frac{\det\boldsymbol{\Sigma}_i}{\det\boldsymbol{\Sigma}}\right) + \frac{1}{2}\mathrm{Tr}(\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\Sigma}) + \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}) - \frac{d}{2}$. Applying the KKT conditions of the problem w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we obtain the updating rules of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \mathbf{y}_i \boldsymbol{\Sigma}_i \mathbf{x}_i,\ \boldsymbol{\Sigma}_{i+1} = \boldsymbol{\Sigma}_i + \beta_i \boldsymbol{\Sigma}_i \mathbf{x}_i \mathbf{x}_i^\top \boldsymbol{\Sigma}_i. \tag{3}$$

where α_i is the Lagrangian multiplier w.r.t. the constraint, and β_i is a scalar related to α_i. In last several years, many variants of CW method have been proposed (Crammer, Dredze, and Pereira 2008; Crammer, Kulesza, and Dredze 2009a; 2009b; Crammer, Dredze, and Pereira 2012; Wang, Zhao, and Hoi 2012) (see Section 2 for more details), but most of the them follow the above rules in (3) to update $\boldsymbol{\mu}_{i+1}$ and $\boldsymbol{\Sigma}_{i+1}$, and the difference lies on different calculations of α_i and β_i.

By modeling a full covariance structure, CW learning provides significant advantages for classification tasks (Crammer, Dredze, and Pereira 2012; Ma et al. 2010; Wang, Zhao, and Hoi 2012). However, there are two deficiencies in existing full CW learning methods. First, it is computationally infeasible to maintain a full covariance structure for high-dimensional data. The diagonalization technique (Crammer, Dredze, and Pereira 2012) can be applied to address this issue, but it ignores feature interactions and often leads to degraded performance (Duchi, Hazan, and Singer 2011). Second, the learning performance might be severely degraded if the data contain many irrelevant features to the output (Ma et al. 2009a).

The main contributions of this paper are as follows.

First, we propose an efficient **online-batch** CW learning scheme for low- and medium-dimensional data, motivated by two common observations: 1) Most computers (even laptops) nowadays allow us to load a batch of examples one time. 2) In many applications, the data are often collected in online-batch. For example, in (Ma et al. 2009b), data for malicious web sites detection are collected by day. The batch learning scheme significantly reduces the overall complexity by avoiding repetitive covariance updating, and may also help to learn a more accurate model. Empirical studies show that it is much faster than existing methods while achieving similar or even better accuracy.

Second, the proposed online-batch CW scheme however cannot handle very high-dimensional data if maintaining a full covariance structure. To address this and choose the most relevant features, we further propose a novel paradigm to learn sparse CW classifiers, which automatically chooses groups of features and builds a block diagonal covariance structure.

Related Studies

The original CW learning (Dredze, Crammer, and Pereira 2008) has theoretical guarantees in the mistake-bound (Crammer, Dredze, and Pereira 2012; Dredze, Crammer, and Pereira 2008), but the aggressive update rules in original CW learning in (Dredze, Crammer, and Pereira 2008) may incur severe over-fitting when the data or labels are noisy. To address this issue, *Adaptive Regularization of Weights* (AROW) (Crammer, Kulesza, and Dredze 2009a) employs an adaptive regularization for each example by replacing the probabilistic constraint in (1) with a squared hinge loss plus a confidence penalty:

$$\begin{aligned}(\boldsymbol{\mu}_{i+1}, \boldsymbol{\Sigma}_{i+1}) = &\arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) || \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right) \\ &+ C_1\xi_i^q + C_2\mathbf{x}_i^\top\boldsymbol{\Sigma}\mathbf{x}_i,\end{aligned} \tag{4}$$

where $\xi_i = \max\left(0, 1 - y_i\boldsymbol{\mu}^\top\mathbf{x}_i\right)$, and C_1 and C_2 are trade-off parameters. Another method, called *Soft Confidence-Weighted* (SCW) (Wang, Zhao, and Hoi 2012), replaces the constraint (i.e. equation (1)) with $\xi_i = \max(0, \phi\sqrt{\mathbf{x}^\top\boldsymbol{\Sigma}\mathbf{x}} - y_i\boldsymbol{\mu}^\top\mathbf{x}_i)$. There are some other variants of CW methods, see (Crammer and Lee 2010; Orabona and Crammer 2010).

CW learning is related to second-order methods like second-order perceptron (Cesa-Bianchi, Conconi, and Gentile 2005; Gentile, Vitale, and Brotto 2008; Duchi, Hazan, and Singer 2011), ellipsoid methods (Yang, Jin, and Ye 2009), and so on. All these methods update $\boldsymbol{\Sigma}$ in each round, which cannot be applied to high-dimensional data. To extend second-order methods to high-dimensional data, Ma *et al.* developed a low-rank approximation to the covariance matrix so it can be cheaply stored (Ma et al. 2010). However, this method might be sensitive to irrelevant features (Wu, Hoi, and Mei 2014).

Recently, a CW learning based second-order feature selection (SOFS) method is proposed in (Wu, Hoi, and Mei 2014). However, this method relies on sparse data. More feature selection methods regarding classification problems can be found in (Shalev-Shwartz and Zhang 2012; Tan, Tsang, and Wang 2014; Yuan, Ho, and Lin 2012; Chang et al. 2014; Han and Zhang 2015).

Online-Batch Confidence-Weighted Learning

Notation. Let the superscript $^\top$ denote the transpose of a vector/matrix, $\mathbf{0}$ be a vector/matrix with all zeros, $\mathrm{diag}(\mathbf{v})$ be a diagonal matrix with diagonal elements equal to $\mathbf{v}$, and $\|\mathbf{v}\|_p$ be the ℓ_p-norm of a vector $\mathbf{v}$, $\mathbf{A} \odot \mathbf{B}$ be the element-wise product of two matrices $\mathbf{A}$ and $\mathbf{B}$. Let $[n] = \{1, ..., n\}$.

Let $\widehat{\boldsymbol{\mu}}$ and $\widehat{\boldsymbol{\Sigma}}$ be model parameters at the $(h-1)^{\mathrm{th}}$ time slot (where $h > 1$). Given a batch of N_h training examples $\{\mathbf{X}_h, \mathbf{y}_h\}$ at the time slot h, where $\mathbf{X}_h \in \mathbb{R}^{d\times N_h}$ and $\mathbf{y}_h \in \{1, -1\}^{N_h}$, we seek to estimate parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ regarding the h^{th} time slot in an alternating scheme.

Procedure 1: Learning $\boldsymbol{\Sigma}$. Motived by AROW (Crammer, Kulesza, and Dredze 2009a), given $\boldsymbol{\mu}$, we learn $\boldsymbol{\Sigma}$ by addressing the following problem:

$$\min_{\boldsymbol{\Sigma}} \ \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})) + \tfrac{C}{2}\textstyle\sum_{i=1}^{N_h}\mathbf{x}_i^\top\boldsymbol{\Sigma}\mathbf{x}_i, \tag{5}$$

where $C > 0$ is a trade-off parameter. By applying the KKT condition, we have $\boldsymbol{\Sigma}^{-1} = \widehat{\boldsymbol{\Sigma}}^{-1} + C\mathbf{X}_h\mathbf{X}_h^\top$. Using the Woodbury identity, we can update $\boldsymbol{\Sigma}$ by

$$\begin{aligned}\boldsymbol{\Sigma} &= \left(\widehat{\boldsymbol{\Sigma}}^{-1} + C\mathbf{X}_h\mathbf{X}_h^\top\right)^{-1} \\ &= \widehat{\boldsymbol{\Sigma}} - \widehat{\boldsymbol{\Sigma}}\mathbf{X}_h(\frac{1}{C}\mathbf{I}_{N_h} + \mathbf{X}_h^\top\widehat{\boldsymbol{\Sigma}}\mathbf{X}_h)^{-1}\mathbf{X}_h^\top\widehat{\boldsymbol{\Sigma}}.\end{aligned} \tag{6}$$

In (6), the first matrix inverse takes $O(d^3 + N_h^2 d)$ complexity, which is unbearable when d is large; while the second one takes $O(N_h^3 + N_h d^2)$ complexity, which is unbearable when N_h is large. Nevertheless, we only need to compute once for the instances in the same batch. Moreover, as we consider all examples in the same batch, this $\boldsymbol{\Sigma}$ should be more accurate compared to the online updating (where the accuracy of $\boldsymbol{\Sigma}$ is improved gradually w.r.t. rounds).

Procedure 2: Learning $\boldsymbol{\mu}$. After finding $\boldsymbol{\Sigma}$, we learn the classifier weight $\boldsymbol{\mu}$ by solving the following problem:

$$\min_{\boldsymbol{\mu}} \ \tfrac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \tfrac{C}{q}\textstyle\sum_{i=1}^{N_h}\xi_i^q, \tag{7}$$

where $\xi_i = \max(0, 1 - y_i\boldsymbol{\mu}^\top\mathbf{x}_i)$ is the hinge loss, and q is either 1 or 2. By applying the KKT condition on $\boldsymbol{\mu}$, we have $\boldsymbol{\mu} = \sum_{i=1}^{N_h}\alpha_i y_i\boldsymbol{\Sigma}\mathbf{x}_i$, where α_i is the Lagrangian multiplier regarding the ith example. Based on this formula, we can compute $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t\boldsymbol{\Sigma}\mathbf{x}_t$ sequentially starting from $\boldsymbol{\mu}_0 = 0$, as in (Crammer, Kulesza, and Dredze 2009a). However, this method cannot deal with hinge loss (i.e $q = 1$).

We now present a method to deal with general loss functions. Let $\boldsymbol{\Upsilon}$ be the squared root of $\boldsymbol{\Sigma}$ (e.g., $\boldsymbol{\Upsilon}^2 = \boldsymbol{\Sigma}$),

$\mathbf{w} := \boldsymbol{\Upsilon}^{-1}\boldsymbol{\mu}$ and $\widehat{\mathbf{x}}_i := \boldsymbol{\Upsilon}\mathbf{x}_i$. Problem (7) can be reformulated as follows:

$$\min_{\mathbf{w}} \ \frac{1}{2}||\mathbf{w}||_2^2 + \frac{C}{q}\sum_{i=1}^{N_h} \xi_i^q, \qquad (8)$$

where $\xi_i = \max(0, 1 - y_i\mathbf{w}^\top\widehat{\mathbf{x}}_i)$. This problem is a standard linear SVM problem (Hsieh et al. 2008), and can be efficiently solved through dual coordinate descent(DCD) method as in (Hsieh et al. 2008; Shalev-Shwartz and Zhang 2012; 2013). Following (Hsieh et al. 2008), we conduct the online-batch CW learning in Algorithm 1.

Algorithm 1: Learning confidence-weighted classifiers in online-batch.

```
Require: Parameters $r, C > 0$, and $\widehat{\boldsymbol{\mu}} = \mathbf{0}$ and $\widehat{\boldsymbol{\Sigma}} = \mathbf{I}$.
  for $h = 1 : H$ do
    Receive a batch of data $\{\mathbf{X}_h, \mathbf{y}_h\}$, where $\mathbf{X}_h \in \mathbb{R}^{d \times N_h}$.
    Compute $\boldsymbol{\Sigma}$ by (6) and $\boldsymbol{\Upsilon}$ by the eigen-decomposition of
    $(\widehat{\boldsymbol{\Sigma}}^{-1} + C\mathbf{X}_h\mathbf{X}_h^\top)$.
    Compute $\widehat{\mathbf{X}} = \boldsymbol{\Upsilon}\mathbf{X}_h$, and initialize $\mathbf{w}^0 = \boldsymbol{\Upsilon}^{-1}\widehat{\boldsymbol{\mu}}$ and $\boldsymbol{\alpha} = \mathbf{0}$.
    for $i = 1 : N_h$ do
      Compute $\xi_i = \max(1 - y_i\mathbf{w}^\top\widehat{\mathbf{x}}_i, 0)$.
      if $\xi_i > 0$ then
        Compute $\alpha_i = \min(\xi_i/||\widehat{\mathbf{x}}_i||_2^2, C)$ for $q = 1$ or
        $\alpha_i = \xi_i/(||\widehat{\mathbf{x}}_i||_2^2 + 0.5/C)$ for $q = 2$.
        Compute $\mathbf{w} = \mathbf{w} + \alpha_i y_i \widehat{\mathbf{x}}$.
    Compute $\boldsymbol{\mu} = \boldsymbol{\Upsilon}\mathbf{w}$. Let $\widehat{\boldsymbol{\Sigma}} := \boldsymbol{\Sigma}$, $\widehat{\boldsymbol{\mu}} := \boldsymbol{\mu}$.
```

The inner *for loop* in Algorithm 1 is implemented in an online setting, i.e., it only scans the data once; while DCD method (Hsieh et al. 2008) which is a batch method may need to scan the data many times to converge. By considering the covariance structure in learning $\mathbf{w}$, Algorithm 1 often converges faster and achieves comparable or even better prediction performance than batch methods (See comparisons with DCD in Table 2).

Algorithm 1 is related to the *whitened Perceptron* algorithm (Cesa-Bianchi, Conconi, and Gentile 2005) (See SOP method in Table 2). However, the updating of weight vector in Algorithm 1 is different from SOP. In addition, in Algorithm 1, one needs to compute $\boldsymbol{\Sigma}$ once for each batch, thus the expensive parts, namely $\mathbf{X}_h\mathbf{X}_h^\top$ and $\boldsymbol{\Upsilon}\mathbf{X}$, can be efficiently computed in parallel. So our algorithm has big advantages over existing second-order learning methods.

Learning Sparse CW Classifiers[1]

The proposed online-batch CW learning scheme, however, cannot deal with very high-dimensional data due to the maintaining of the full covariance matrix. However, in general, for high-dimensional data, the number of relevant features is often very small. What's more, although there might be many second-order feature interactions, only those interactions involving relevant features are significant. These two observations regarding the very high-dimensional data motivate us to learn sparse confidence-weighted classifiers and sparse covariance structures.

[1]All the proofs can be found in the supplementary file which is available from the author's website.

To identify the most relevant features w.r.t. the output $\mathbf{y}$, we consider to introduce an index vector $\boldsymbol{\eta} \in \{0,1\}^d$ to scale each instance $\mathbf{x}$ by $(\boldsymbol{\eta} \odot \mathbf{x})$. Here, a feature j is chosen if $\eta_j = 1$; otherwise it will be dropped. Accordingly, the hinge loss can be expressed as

$$\xi_i(\boldsymbol{\eta}) = \max\left(0, 1 - y_i\boldsymbol{\mu}^\top(\mathbf{x}_i \odot \boldsymbol{\eta})\right).$$

Without loss of generality, in the following, we assume that $\boldsymbol{\mu} = \mathbf{0}$ at the beginning, i.e., no feature is selected when $h = 0$. By introducing the new variable $\boldsymbol{\eta}$, we will do the CW learning under the following two goals. Firstly, we prefer to select the least number of features that would fit the data well. Secondly, only the features selected by $\boldsymbol{\eta}$ will be considered in the second order feature interactions.

Regarding the first goal, we impose an ℓ_0-norm constraint on $\boldsymbol{\eta}$ to induce the sparsity, i.e., $||\boldsymbol{\eta}||_0 \le r$ (where $r \ll d$). For convenience, let $\Lambda := \{\boldsymbol{\eta} | \boldsymbol{\eta} \in \{0,1\}^d, ||\boldsymbol{\eta}||_0 \le r\}$ be the set of all feasible $\boldsymbol{\eta}$'s. As $|\Lambda| = \sum_{i=0}^{r}\binom{d}{i}$, i.e., there are $|\Lambda|$ feasible $\boldsymbol{\eta}$'s in total. Then the feature selection task can be cast as an optimization problem to pick up an optimal $\boldsymbol{\eta}$ from Λ. Focusing on the fitness of data only, we may find an optimal $\boldsymbol{\eta}$ by solving the following optimization problem:

$$\min_{\boldsymbol{\eta} \in \Lambda} \min_{\boldsymbol{\mu}, \boldsymbol{\xi}} \ \frac{C}{q}\sum_{i=1}^{N_h} \xi_i(\boldsymbol{\eta})^q. \qquad (9)$$

For each feasible $\boldsymbol{\eta}$, solving the inner minimization problem w.r.t. $\boldsymbol{\mu}$ and $\boldsymbol{\xi}$ will obtain the fitness of the associated features. However, this objective may over-fit and ignores the feature interactions. Therefore, we need to consider the second goal, that is, we will maintain a covariance matrix w.r.t. the features indexed by $\boldsymbol{\eta}$. If a good $\boldsymbol{\eta}$ is given, motivated by the batch CW learning, we can compute the covariance matrix $\boldsymbol{\Sigma}$ by solving the following problem:

$$\min_{\boldsymbol{\Sigma}} \mathrm{D_{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) || \mathcal{N}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}})) + \frac{C}{2}\sum_{i=1}^{N_h}(\mathbf{x}_i \odot \boldsymbol{\eta})^\top \boldsymbol{\Sigma}(\mathbf{x}_i \odot \boldsymbol{\eta}).$$

Let $\mathbf{X}_h^r = \mathrm{diag}(\boldsymbol{\eta})\mathbf{X}_h$. By the KKT condition on $\boldsymbol{\Sigma}$, we achieve a closed solution of $\boldsymbol{\Sigma}$ by $\boldsymbol{\Sigma}(\boldsymbol{\eta}) = \left(\widehat{\boldsymbol{\Sigma}}^{-1} + C(\mathbf{X}_h^r)(\mathbf{X}_h^r)^\top\right)^{-1} = \widehat{\boldsymbol{\Sigma}} - \widehat{\boldsymbol{\Sigma}}\mathbf{X}_h^r\left(\frac{1}{C}\mathbf{I}_{N_h} + (\mathbf{X}_h^r)^\top\widehat{\boldsymbol{\Sigma}}(\mathbf{X}_h^r)\right)^{-1}(\mathbf{X}_h^r)^\top\widehat{\boldsymbol{\Sigma}}$.

Remark 1. *The matrix inverse above is operated on r features only, thus the computation of* $\boldsymbol{\Sigma}(\boldsymbol{\eta})$ *takes* $O(r^3 + N_h^2 r)$ *complexity.* $\boldsymbol{\Sigma}(\boldsymbol{\eta})$ *can be cheaply computed since* $r \ll d$.

Once $\boldsymbol{\Sigma}(\boldsymbol{\eta})$ is computed, we incorporate it into formulation (9). Similar in (7), we obtain the following problem:

$$\min_{\boldsymbol{\eta} \in \Lambda} \min_{\boldsymbol{\mu}} \ \frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}(\boldsymbol{\eta})^{-1}\boldsymbol{\mu} + \frac{C}{q}\sum_{i=1}^{N_h}\xi_i(\boldsymbol{\eta})^q. \qquad (10)$$

The integer r denotes the number of features we intend to select, which actually represents our basic prior knowledge about the data. For example, setting $r = 1$ means that there might be one relevant feature. However, as will be shown, our later proposed optimization scheme guarantees to find more features (if relevant) even when we set $r = 1$. Nevertheless, a relatively large r is useful and also necessary to detect the correlated features via our optimization paradigm.

Problem (10) is a mixed integer programming problem which is hard to solve. However, we can transform it into a standard convex programming problem via convex relaxation (Tan, Wang, and Tsang 2010):

$$\max_{\theta \in \mathbb{R}, \boldsymbol{\alpha} \in \mathcal{A}} \quad \theta, \quad \text{s.t.} \;\; \theta \leq f(\boldsymbol{\alpha}, \boldsymbol{\eta}), \quad \forall \boldsymbol{\eta} \in \Lambda. \tag{11}$$

Here, the function $f(\boldsymbol{\alpha}, \boldsymbol{\eta})$ is defined as $f(\boldsymbol{\alpha}, \boldsymbol{\eta}) := -\frac{1}{2}\boldsymbol{\mu}(\boldsymbol{\alpha}, \boldsymbol{\eta})^\top \boldsymbol{\Sigma}(\boldsymbol{\eta}) \boldsymbol{\mu}(\boldsymbol{\alpha}, \boldsymbol{\eta}) - (q-1)\frac{\boldsymbol{\alpha}^\top \boldsymbol{\alpha}}{2C} + \sum_{i=1}^{N_h} \alpha_i$, where $\boldsymbol{\mu}(\boldsymbol{\alpha}, \boldsymbol{\eta}) := \sum_{i=1}^{N_h} \alpha_i y_i (\mathbf{x}_i \odot \boldsymbol{\eta})$, $\boldsymbol{\alpha} \in \mathbb{R}^{N_h}$ is the dual variable regarding constraint $\xi_i(\boldsymbol{\eta}) = \max\big(0, 1 - y_i \boldsymbol{\mu}^\top (\mathbf{x}_i \odot \boldsymbol{\eta})\big), \forall i \in [N_h]$, and $\mathcal{A} := \{\boldsymbol{\alpha} \in \mathbb{R}^{N_h} | 0 \leq \alpha_i \leq U\}$ is the domain of $\boldsymbol{\alpha}$ (here, $U = C$ for $q = 1$ and $U = \infty$ for $q = 2$). Recall that each feasible $\boldsymbol{\eta} \in \Lambda$ corresponds to a constraint, thus problem (11) has exponentially many constraints as there are $\sum_{i=0}^{r} \binom{d}{i}$ elements in Λ, making it hard to address directly.

General Optimization Paradigm

Note that there are exponentially many constraints in problem (11). Motivated by the cutting-plane approach (Kortanek and No 1993), we propose to address it by Algorithm 2. As this algorithm is based on the online-batch CW model, hereafter we refer to it as sparse BCW (SBCW).

Algorithm 2: Sparse CW learning in online-batch.

```
Require: Parameters $r, C > 0, H, \widehat{\boldsymbol{\mu}} = \mathbf{0}$ and $\widehat{\boldsymbol{\Sigma}} = \mathbf{I}$.
  for $h = 1 : H$ do
      Load data $\{\mathbf{X}_h, \mathbf{y}_h\}$, where $\mathbf{X}_h \in \mathbb{R}^{d \times N_h}$ and $\mathbf{y}_h \in \{1, -1\}^{N_h}$,
      $\Lambda_0 = \emptyset$ and $T_h = \sum_{i=0}^{r} \binom{d}{i}$.
      for $t = 1 : T_h$ do
          Find $\widehat{\boldsymbol{\eta}}$ by solving the problem (12). Let $\Lambda_t = \Lambda_{t-1} \bigcup \{\widehat{\boldsymbol{\eta}}\}$.
          Update $\boldsymbol{\alpha}^t$ by solving the subproblem in (13) w.r.t. $\Lambda_t$.
          Terminate if stopping conditions are achieved.
      Let $\Lambda^h = \Lambda_t$.
```

Instead of dealing with all $T = \sum_{i=0}^{r} \binom{d}{i}$ constraints, we iteratively find a constraint until some stopping conditions are achieved. Given $\boldsymbol{\alpha}^{t-1}$, the most-violated constraint can be found by solving the following optimization problem:

$$\begin{aligned} \widehat{\boldsymbol{\eta}} &= \arg\min_{\boldsymbol{\eta} \in \Lambda} \;\; f(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\eta}) \\ &= \arg\max_{\boldsymbol{\eta} \in \Lambda} \;\; \boldsymbol{\mu}(\boldsymbol{\alpha}, \boldsymbol{\eta})^\top \boldsymbol{\Sigma}(\boldsymbol{\eta}) \boldsymbol{\mu}(\boldsymbol{\alpha}, \boldsymbol{\eta}). \end{aligned} \tag{12}$$

where $\boldsymbol{\mu}(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\eta}) := \sum_{i=1}^{N_h} \alpha_i^{t-1} y_i (\mathbf{x}_i \odot \boldsymbol{\eta})$. This problem, unfortunately, is non-trivial to solve due to the involvement of $\boldsymbol{\eta}$ in $\boldsymbol{\Sigma}(\boldsymbol{\eta})$.

After obtaining an active constraint, associated with $\widehat{\boldsymbol{\eta}}$, we add it into the active set $\Lambda_t = \Lambda_{t-1} \bigcup \{\widehat{\boldsymbol{\eta}}\}$, and then address the following subproblem w.r.t. constraints defined by Λ_t:

$$\theta_t := \max_{\theta \in \mathbb{R}, \boldsymbol{\alpha} \in \mathcal{A}} \quad \theta, \quad \text{s.t.} \;\; \theta \leq f(\boldsymbol{\alpha}, \boldsymbol{\eta}), \quad \forall \boldsymbol{\eta} \in \Lambda_t. \tag{13}$$

In this following, we discuss the optimization of (12) and (13).

Consider (12) first, which is difficult. However, on one hand, for feature selection, we focus on choosing the most relevant features to the output. On the other hand, $\boldsymbol{\Sigma}(\boldsymbol{\eta})$ is not in the loss $\xi_i(\boldsymbol{\eta}) = \max\big(0, 1 - y_i \boldsymbol{\mu}^\top (\mathbf{x}_i \odot \boldsymbol{\eta})\big)$. We thus assume that $\boldsymbol{\Sigma}(\boldsymbol{\eta})$ is an identity matrix, and reduce problem (12) to the following problem:

$$\widehat{\boldsymbol{\eta}} = \arg\max\nolimits_{\boldsymbol{\eta} \in \Lambda} \sum_{j=1}^{d} \eta_j \sum_j s_j^2, \tag{14}$$

where $\mathbf{s} = \sum_{i=1}^{N_h} \alpha_i y_i \mathbf{x}_i$. For this problem, its optimal solution can be easily obtained by finding the r features with the largest score (e.g. s_j), and then setting the corresponding η_j to 1 and the rest to 0. In this sense, s_j^2 can be considered as the feature score for the jth feature, i.e., s_j^2 essentially measures the importance of the feature.

Theorem 1. *Let $T = |\Lambda| < \infty$, assume that both problems (12) and (13) can be addressed, then $\{\theta_t\}_{t=1}^{T}$ is monotonically decreasing and Algorithm 2 stops after a finite number of iterations with a global solution of problem (11).*

Remark 2. *The solution to problem (14), however, is not necessary to be the optimal solution of problem (12). In Algorithm 2, finding the solution of problem (12) is to update the constraint set Λ_t. It is easy to see that any update of set Λ_t will make $\{\theta_t\}_{t=1}^{T}$ monotonically decreasing, but the updating by solving (14) will make the decreasing faster.*

Stopping Conditions

We may use following stopping criteria. First, as the objective θ_t decreases monotonically (see Theorem 1), the algorithm can be stopped if $|\theta_t - \theta_{t-1}|/|\theta_t| \leq \epsilon$ is true, where ϵ is a small tolerance value. Second, given limited memory, we stop the algorithm if the memory is not enough. Third, given a fixed r, the algorithm can be stopled after $m_{iter} = \lceil p/r \rceil$ iterations in order to choose p features. For example, in our experiments on feature selection task, we stop SBCW after $m_{iter} = 15$ iterations, and set $r = \lceil p/m_{iter} \rceil$ in order to select p features, which often produces satisfactory results.

Optimization of Subproblem (13)

Solving the subproblem in (13) w.r.t. $\boldsymbol{\alpha}$ is computationally expensive. In the following, we study how to solve it efficiently by applying a proximal primal-dual coordinate ascent method. Let $K = |\Lambda_t|$ be the number of active constraints. For each $\boldsymbol{\eta}_k \in \Lambda_t$, where $k \in [K]$, we let $\mathbf{x}_i^k \in \mathbb{R}^r$ denote the ith instance w.r.t. features associated with $\boldsymbol{\eta}_k \in \Lambda_t$, and $\boldsymbol{\mu}_k \in \mathbb{R}^r$ and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{r \times r}$ be associated model parameter and covariance matrix, respectively. Let $\boldsymbol{\Upsilon}_k$ be the square root of $\boldsymbol{\Sigma}_k$, $\mathbf{w}_k := \boldsymbol{\Upsilon}_k^{-1} \boldsymbol{\mu}_k$, $\widehat{\mathbf{x}}_i^k := {\boldsymbol{\Upsilon}_k}^\top \mathbf{x}_i^k$.

Proposition 1. *Let $\xi_i = \max(0, 1 - \sum_{k=1}^{t} \mathbf{w}_k^\top \widehat{\mathbf{x}}_i^k)$. The subproblem (13) is the dual of the following problem:*

$$\min\nolimits_{\mathbf{w}, \boldsymbol{\xi}} \;\; \frac{1}{2}\Big(\sum_{k=1}^{K} \|\mathbf{w}_k\|\Big)^2 + \frac{C}{q} \sum_{i=1}^{N_h} \xi_i^q. \tag{15}$$

The $\ell_{2,1}^2$-norm regularizer $\frac{1}{2}(\sum_{k=1}^{K} \|\mathbf{w}_k\|)^2$ in problem (15) is non-smooth, and would encourage **sparsity** among $\mathbf{w}_k$'s. Focusing on large-scale data, we apply a proximal primal-dual coordinate ascent method (Shalev-Shwartz and Zhang 2012) to find a nearly accurate solution of problem

(15) efficiently. To apply this algorithm, we add a small regularization term $\frac{\sigma}{2}\|\mathbf{w}\|^2$ (i.e., $\sigma \ll 1$), and address the following optimization problem instead:

$$\min_{\mathbf{w}} \frac{\sigma}{2}\|\mathbf{w}\|^2 + \frac{1}{2}(\sum_{k=1}^K \|\mathbf{w}_k\|)^2 + C\sum_{i=1}^{N_h} L_i(\mathbf{w}^\top \widehat{\mathbf{x}}_i)^q \tag{16}$$

where $\mathbf{w} = [\mathbf{w}_k]_{k=1}^K, \widehat{\mathbf{x}}_i = [\widehat{\mathbf{x}}_i^k]_{k=1}^K$ and $L_i(\mathbf{w}^\top \widehat{\mathbf{x}}_i) := \xi_i^2/q$.

Remark 3. *Let $\mathbf{w}^*$ be an $\frac{\epsilon}{2}$-accurate minimizer of (16). By choosing a sufficiently small σ, $\mathbf{w}^*$ is also an ϵ-accurate solution of (15) (Shalev-Shwartz and Zhang 2012). Therefore, the optimal solutions of problems (15) and (16) are close.*

Let $\Omega(\mathbf{w}) := \frac{\sigma}{2}\|\mathbf{w}\|^2 + \frac{1}{2}(\sum_{k=1}^K \|\mathbf{w}_k\|)^2$ which is strongly convex. Note that L_i is γ-Lipschitz for some $\gamma > 0$ (Shalev-Shwartz and Zhang 2012). Let $\Omega^*(\mathbf{z}) = \max_{\mathbf{w}} \mathbf{w}^\top \mathbf{z} - \Omega(\mathbf{w})$ be the conjugate of $\Omega(\mathbf{w})$, and L_i^* be the conjugate of L_i. The conjugate dual of problem (16) can be written as: $\max_{\boldsymbol{\alpha} \in [0,1]^{N_h}} D(\boldsymbol{\alpha})$, where $D(\alpha_i) = -\Omega^*\left(C\sum_{i=1}^{N_h} \alpha_i \widehat{\mathbf{x}}_i\right) - C\sum_{i=1}^{N_h} L_i^*(-\alpha_i)$. Here, $L_i^*(-\alpha_i) = -\alpha_i y_i$ for hinge loss, and $L_i^*(-\alpha_i) = -\frac{1}{2}\alpha_i^2 - \alpha_i y_i$ for squared hinge loss. Following (Shalev-Shwartz and Zhang 2012), we define

$$\mathbf{w}(\boldsymbol{\alpha}) = \nabla^*\Omega(\mathbf{z}(\boldsymbol{\alpha})) \text{ and } \mathbf{z}(\boldsymbol{\alpha}) = C\sum_{i=1}^{N_h} \alpha_i \widehat{\mathbf{x}}_i, \tag{17}$$

where $\nabla^*\Omega(\mathbf{z}(\boldsymbol{\alpha}))$ denotes the gradient of the conjugate Ω, and is also the minimizer of problem $\Omega^*(\mathbf{z}) = \max_{\mathbf{w}} \ \mathbf{w}^\top \mathbf{z} - \Omega(\mathbf{w})$.[2] The proximal primal-dual coordinate ascent for solving problem (15) is shown in Algorithm 3. Note that it is implemented in the online setting.

Algorithm 3: Online proximal primal-dual coordinate ascent for solving problem (13)

```
Require: Parameters $C > 0$, input data $\{\widehat{\mathbf{X}}_h^k\}_{k=1}^K$ and $\mathbf{y}_h$, and $\{\boldsymbol{\Upsilon}_k\}_{k=1}^K$.
    Initialize $\mathbf{z}^0 = \mathbf{0}$ and $\mathbf{w}^0 = \mathbf{0}$.
    for $i = 1 : N_h$ do
        Compute loss $\xi_i = \max\left(0, 1 - y_i \sum_{k=1}^K \mathbf{w}_k^\top \widehat{\mathbf{x}}_i^k\right)$.
        if $\xi_i > 0$ then
            Compute $\alpha_i = \min(\xi_i/(C\|\widehat{\mathbf{x}}_i\|_2^2), 1)$ for $q = 1$ or
            $\alpha_i = \xi_i/(C\|\widehat{\mathbf{x}}_i\|_2^2 + 0.5)$ for $q = 2$.
            Compute $\mathbf{z}^i = \mathbf{z}^{i-1} + C\alpha_i y_i \widehat{\mathbf{x}}_i$.
            Compute $\mathbf{w}^i = \nabla^*\Omega(\mathbf{z}^i)$.
```

Complexity Analysis

Based on Algorithm 3, we summarize the complexity of SBCW as follows.

Proposition 2. *Let K be the number of active $\boldsymbol{\eta}$'s being chosen. Depending on the permutation of features via $\boldsymbol{\eta}$, SBCW essentially maintains a **block diagonal covariance structure** $\boldsymbol{\Sigma} \in \mathbb{R}^{Kr \times Kr}$ regarding all selected features with K submatrices $\boldsymbol{\Sigma}_k$ on the diagonal. The overall time and space complexities thus are $O(dr^2 + dN_h r + dN_h)$ and $O(dr + dN_h)$, respectively.*

[2]The computation of $\nabla\Omega^*(\mathbf{z})$ is put in the supplementary file.

Experiments

We conduct two sets of experiments to verify our methods. First, we compare the proposed BCW method with several state-of-the-art online learning methods on several low-dimensional data sets. Second, we verify the feature selection performance of our proposed SBCW method on several high-dimensional data sets. The sources of our methods are available from http://www.tanmingkui.com/sbcw.html.

All data sets are widely used benchmarks in machine learning, and are summarized in Table 1. `URL` is originally from http://sysnet.ucsd.edu/projects/url/ for identifying suspicious URLs, `astro-ph` is from (Hsieh et al. 2008), and others are from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. From Table 1, all data sets are sparse data sets except `epsilon`. For each data set, we load all instances into the memory as our machine can afford enough memory, which is consistent with our earlier claim that Most computers nowadays allow us to load a batch of examples one time. All experiments are conducted on a PC installed a 64-bit operating system with an Intel(R) Core(TM) Xeon CPU 3.00GHz and 32GB memory.

Table 1: Data sets used in the experiments.

Data set	d	N_{train}	N_{te}	# nonzeros per inst
`covtype`	54	581,012	–	12
`epsilon`	2,000	400,000	100,000	2,000
`real-sim`	20,958	32,309	40,000	52
`rcv1`	47,236	677,399	20,242	74
`astro-ph`	99,757	62,369	32,487	77
`URL`	3,231,961	2,396,130	–	115

Experiments on Low-dimensional Data Sets

We compare the proposed BCW method on three low-dimensional data sets (see Table 1) with several well-known online learning methods, including PA (Crammer et al. 2006), SOP (Cesa-Bianchi, Conconi, and Gentile 2005), AROW (Crammer, Kulesza, and Dredze 2009a), NAROW (Orabona and Crammer 2010), SCW (with hinge loss) (Wang, Zhao, and Hoi 2012), SCW2 (with squared hinge loss) (Wang, Zhao, and Hoi 2012). Except PA, others are second-order methods, and NAROW, SCW and SCW2 are considered as the state-of-the-arts. Since BCW is closely related to batch DCD method (hinge loss) (Hsieh et al. 2008), we also include DCD as a baseline. For all competing methods, we apply 5-cross-validation to choose their parameters. All compared online methods are from http://libol.stevenhoi.org; while DCD is from the Liblinear package http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

For `covtype`, we sample 450k examples from the entire set as training data and the rests as testing data. The full `epsilon` data set contains 400k training samples, which is too huge to other CW methods. We thus sample $50k$ examples randomly for the comparison. Nevertheless, our BCW, PA and DCD can handle the entire data set (See Table 1).

Online learning methods are often affected by sample orders. Therefore, we run the experiments 10 times with random orders, and record the mean and standard variance of

Table 2: Experimental results on low-dimensional data sets, where time is reported in seconds.

Method	`covtype 450k`		`epsilon 50k`		`epsilon (entire set)`	
	Test error rate	Time	Test error rate	Time	Test error rate	Time
AROW (Crammer, Kulesza, and Dredze 2009a)	0.2461 ± 0.0000	21.72	0.1105 ± 0.0004	3506.0	–	–
CW (Dredze, Crammer, and Pereira 2008)	0.4330 ± 0.0335	19.13	0.1522 ± 0.0023	3405.1	–	–
NAROW (Orabona and Crammer 2010)	0.3703 ± 0.0352	22.05	0.1423 ± 0.0019	3468.6	–	–
PA (Crammer et al. 2006)	0.3443 ± 0.0433	13.87	0.1751 ± 0.0221	5.77	0.1821	68.60
SCW (Wang, Zhao, and Hoi 2012)	$\mathbf{0.2382 \pm 0.0003}$	20.75	0.1136 ± 0.0003	3381.4	–	–
SCW2 (Wang, Zhao, and Hoi 2012)	0.2964 ± 0.0159	20.40	0.1126 ± 0.0004	3452.7	–	–
SOP (Cesa-Bianchi, Conconi, and Gentile 2005)	0.4653 ± 0.0298	20.77	0.3705 ± 0.0522	3511.6	–	–
DCD	0.2460	29.85	0.1102	2.62	0.1021	81.52
BCWL1	$\mathbf{0.2402 \pm 0.0003}$	0.41(0.06)	$\mathbf{0.1093 \pm 0.0000}$	5.73(0.39)	0.1019	32.06(1.19)
BCWL2	0.2462 ± 0.0005	0.47(0.07)	0.1102 ± 0.0002	6.35(0.41)	**0.1016**	30.87(1.22)

testing errors, and the training times in Table 2. For the proposed BCWL1 and BCWL2, we need to compute the covariance $\mathbf{\Sigma}$ before learning $\mathbf{w}$. For ease of comparison, in Table 2, we record the time in computing $\mathbf{\Sigma}$ (former one) and conducting Algorithm 1 (in parentheses) separately. Note that the results for some methods on entire `epsilon` are absent since these methods require more than 15,000 seconds for training.

From Table 2, BCW achieves the fastest training speed while achieving comparable or better testing accuracy than others. In particular, BCW is much faster than all second-order methods, including all CW methods. There are several reasons for this: BCW only need to compute $\mathbf{\Sigma}$ once, which can be done very efficiently (See computation time in parentheses); while other CW methods need to update $\mathbf{\Sigma}$ for each example. For PA, the covariance structure is not considered, thus the convergence is very slow, leading to worse accuracy. It is worth mentioning that, BCW is faster than DCD, a batch training method, with comparable or even better accuracy. This demonstrate the significance of maintaining covariance structure $\mathbf{\Sigma}$ in BCW.

Feature Selection on High-dimensional Data

We compare the proposed SBCW method with four state-of-the-art feature selection methods, namely ℓ_1-norm SVM by Liblinear (Yuan et al. 2010) (L1SVM), ℓ_1-norm SVM by dual coordinate ascent method (similar to Algorithm 3 and denoted sparDCA) (Shalev-Shwartz and Zhang 2012), feature generating machine (FGM) method (Tan, Tsang, and Wang 2014) which employs similar feature selection strategy to SBCW but does not consider covariance structure, and SOFS (Wu, Hoi, and Mei 2014), a CW based second-order feature selection method. The comparison is performed on four high-dimensional data sets in Table 1. For `url`, we employ two settings: 1) We train on Day0 and predict Day1; 2) We train on data of even days and predict the data of odd days. More details of experimental settings are put in supplementary file. We only show figures of three data sets.

Figure 1 shows the feature selection performance of various methods w.r.t. different number of selected features. From Figure 1, SBCW shows comparable or better performance over competing algorithms in terms of both training speed and prediction accuracy under the same number of features. The reasons are as follows. First, compared to other methods (except SOFS), SBCW considers covariance structure, thus it can capture feature interactions, which will in turn contribute to finding better features. Second, by maintaining the block covariance structure, Algorithm 3 can well and efficiently address the optimization subproblem in an online manner, which contributes to both faster convergence speed and better accuracy than FGM. For SOFS, it considers the covariance structure, but it essentially solves an ℓ_0-norm non-convex problem by hard thresholding, which possibly leads to the relatively worse results.

Conclusion

In this paper, we start by proposing an efficient online-batch CW learning scheme for medium-dimensional problems. To deal with high-dimensional data and choose the most relevant features, we propose a novel SBCW method to learn sparse CW classifiers. SBCW can effectively find groups of relevant features, and naturally maintain a *block diagonal covariance matrix*. Empirical studies on several data sets show that the superior performance of the proposed BCW and SBCW methods over compared state-of-the-art methods on two sets of tasks.

Acknowledgements

This work was in part funded by the Data to Decisions Cooperative Research Centre, Australia, ARC Grant DP140102270, National Natural Science Foundation Key Project #61231016. Ivor W. Tsang is supported by the ARC Future Fellowship FT130100746 and ARC grant LP150100671.

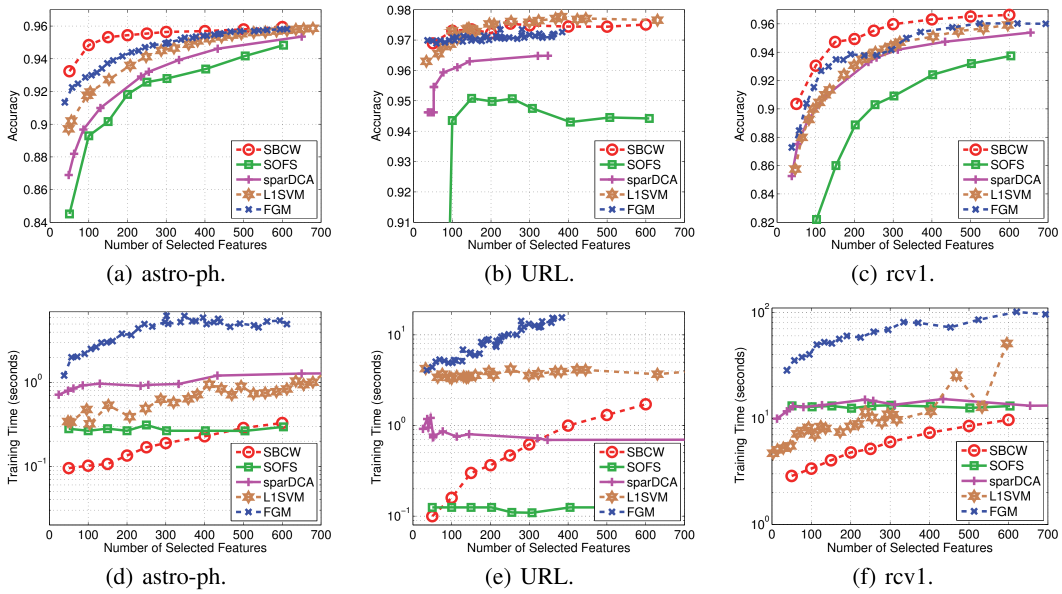
References

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge University Press.

Cesa-Bianchi, N.; Conconi, A.; and Gentile, C. 2005. A second-order perceptron algorithm. *SIAM Journal on Computing*.

Chang, X.; Nie, F.; Yang, Y.; and Huang, H. 2014. A convex formulation for semi-supervised multi-label feature selection. In *AAAI*.

Crammer, K., and Lee, D. 2010. Learning via gaussian herding. In *NIPS*.



(a) astro-ph. (b) URL. (c) rcv1.

(d) astro-ph. (e) URL. (f) rcv1.

Figure 1: Feature selection performance of various methods on high-dimensional data sets in terms of accuracy (see (a)(b)(c)) and training time (see (d)(e)(f)) v.s. number of selected features.

Crammer, K., and Singer, Y. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Y, S. 2006. Online passive-aggressive algorithms. *JMLR*.

Crammer, K.; Dredze, M.; and Pereira, F. 2008. Exact convex confidence-weighted learning. In *NIPS*.

Crammer, K.; Dredze, M.; and Pereira, F. 2012. Confidence-weighted linear classification for text categorization. *JMLR*.

Crammer, K.; Kulesza, A.; and Dredze, M. 2009a. Adaptive regularization of weight vectors. In *NIPS*.

Crammer, K.; Kulesza, A.; and Dredze, M. 2009b. Multi-class confidence weighted algorithms. In *EMNLP*.

Dredze, M.; Crammer, K.; and Pereira, F. 2008. Confidence-weighted linear classification. In *ICML*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Gentile, C.; Vitale, F.; and Brotto, C. 2008. On higher-order perceptron algorithms. In *NIPS*.

Han, L., and Zhang, Y. 2015. Discriminative feature grouping. In *AAAI*.

Hsieh, C.-J.; Chang, K.-W.; Lin, C.-J.; Keerthi, S.; and Sundararajan, S. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*.

Kortanek, K. O., and No, H. 1993. A central cutting plane algorithm for convex semi-infinite programming problems. *SIAM J. on Optimization* 3:4.

Ma, J.; Saul, L.; Savage, S.; and Voelker, G. 2009a. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *KDD*.

Ma, J.; Saul, L.; Savage, S.; and Voelker, G. 2009b. Identifying suspicious urls: an application of large-scale online learning. In *ICML*.

Ma, J.; Kulesza, A.; Dredze, M.; Crammer, K.; Saul, L.; and Pereira, F. 2010. Exploiting feature covariance in high-dimensional online learning. In *AISTATS*.

Orabona, F., and Crammer, K. 2010. New adaptive algorithms for online classification. In *NIPS*.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psych. rev.*

Shalev-Shwartz, S., and Zhang, T. 2012. Proximal stochastic dual coordinate ascent.

Shalev-Shwartz, S., and Zhang, T. 2013. Stochastic dual coordinate ascent methods for regularized loss. *JMLR*.

Tan, M.; Tsang, I. W.; and Wang, L. 2014. Towards ultrahigh dimensional feature selection for big data. *JMLR* 15:1371–1429.

Tan, M.; Wang, L.; and Tsang, I. W. 2010. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, 1047–1054.

Wang, J.; Zhao, P.; and Hoi, S. 2012. Exact soft confidence-weighted learning. In *ICML*.

Wu, Y.; Hoi, S. C. H.; and Mei, T. 2014. Massive-scale online feature selection for sparse ultra-high dimensional data. *arXiv preprint arXiv:1409.7794*.

Xu, T.; Gao, J.; Xiao, L.; and Regan, A. C. 2014. Online classification using a voted rda method. In *AAAI*.

Yang, L.; Jin, R.; and Ye, J. 2009. Online learning by ellipsoid method. In *ICML*.

Yuan, G.-X.; Chang, K.-W.; Hsieh, C.-J.; and Lin, C.-J. 2010. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *JMLR* 11:3183–3234.

Yuan, G.-X.; Ho, C.-H.; and Lin, C.-J. 2012. An improved glmnet for l1-regularized logistic regression. *JMLR* 13(1):1999–2030.

Zhang, L.; Yang, T.; Jin, R.; and Zhou, Z. 2015. Online bandit learning for a special class of non-convex losses. In *AAAI*.