EURO Journal on Computational Optimization manuscript No.
(will be inserted by the editor)

Hyper-parameter Optimization for Support Vector Machines using Stochastic Gradient Descent and Dual Coordinate Descent

Wei Jiang · **Sauleh Siddiqui**

Received: date / Accepted: date

Abstract We developed a gradient-based method to optimize the regularization hyper-parameter, C, for support vector machines in a bilevel optimization framework. On the upper level, we optimized the hyper-parameter C to minimize the prediction loss on validation data using stochastic gradient descent. On the lower level, we used dual coordinate descent to optimize the parameters of support vector machines to minimize the loss on training data. The gradient of the loss function on the upper level with respect to the hyper-parameter, C, was computed using the implicit function theorem combined with the optimality condition of the lower-level problem, i.e., the dual problem of support vector machines. We compared our method with the existing gradient-based method in the literature on several datasets. Numerical results showed that our method converges faster to the optimal solution and achieves better prediction accuracy for large-scale support vector machine problems.

Keywords Hyper-parameter, support vector machines, gradient-based methods

1 Introduction

Machine learning models usually contain hyper-parameters that need to be optimized to prevent overfitting and minimize generalization error. Currently, the standard approach for assessing the generalization error of machine learning models is cross-validation (Friedman et al. (2001), Ron Kohavi (1995)). The standard practice to select the hyper-parameters is using grid-search or random search combined with cross-validation. Specifically, these methods search through a fixed (grid-search) or random (random search) set of values for the hyper-parameters and choose the one that yields the best model performance evaluated by cross-validation. Grid search and random search are computationally expensive especially

Wei Jiang
Department of Civil Engineering, Johns Hopkins System Institute, Johns Hopkins University, 3400 N Charles St, Baltimore MD 21218, USA
Tel.: +1-443-240-4268
E-mail: wjiang1990@gmail.com; wjiang16@jhu.edu

Sauleh Siddiqui
Department of Civil Engineering, Johns Hopkins University, Baltimore, MD 21218, USA
E-mail: siddiqui@jhu.edu

given high-dimensional hyper-parameter space. Moreover, a locally optimal solution is not even guaranteed.

Support vector machines (SVM) are popular and efficient methods for supervised learning, specifically, classification tasks (Cortes and Vapnik (1995)). They contain a regularization hyper-parameter C to control for model complexity and overfitting. SVM can also have additional kernel hyper-parameters if we decide to use non-linear kernels, such as a commonly used kernel, Gaussian radial basis function kernels or RBF kernel. In practice, often grid-search or random search is used to choose the hyper-parameters. For more complex machine learning models, particularly, deep neural networks (deep learning models), the training process often involves tuning the hyper-parameter (most importantly, the learning rate, i.e., the step size in gradient descent method) manually while babysitting the training process. As the dimensionality of dataset becomes very large, its computationally expensive and inefficient to use those empirical ad hoc methods for hyper-parameter tuning. On the other hand, choosing good hyper-parameters is very important to obtain trained machine learning models with good predictive performance on unseen test data. It is often the critical step for building complex nonlinear machine learning models such as deep neural networks, and SVM with nonlinear kernels.

Researchers have started to create more efficient hyper-parameter tuning methods for machine learning algorithms in general. Hyper-parameter optimization has been naturally formulated as a bilevel optimization problem. On the outer or upper level, these methods minimize the prediction error on unseen test data, also called the generalization error. On the inner or lower level, these methods minimize the error of model fitting on training data. Gradient-based methods for optimizing hyper-parameters have emerged as early as 1999 by Bengio (1999). Those gradient-based methods mostly rely on the implicit function theorem. Precisely, those methods differentiate the upper-level loss function with respect to the hyper-parameters using an implicit equation (the optimality condition of inner optimization problem). For instance, Bengio (1999) used a gradient-based method by computing the gradient of the loss function with respect to hyperparameters using the implicit function theorem to optimize multiple hyper-parameters for a general smooth training loss function, particularly the quadratic loss function. Do et al. (2007) used gradient-based methods with the implicit function theorem to efficiently optimize multiple hyper-parameters for log-linear models. To avoid computing a demanding exact gradient in the implicit function theorem, Pedregosa (2016) optimized the hyper-parameters with approximate gradients and provided numerical results on l^2 -regularized logistic regression and kernel Ridge regression models.

Recently, new gradient-based methods have been emerging for hyper-parameter optimization for deep neural network models. These methods make the hyper-parameter tuning process faster by overcoming expensive memory requirement while retaining good model performance (Maclaurin et al. (2015), Fû et al. (2016), Jules (2017), Franceschi et al. (2017)). Typical hyper-parameters for deep neural networks include learning rates and regularization parameters. Computing the hyper-parameter gradient using reverse-mode differentiation usually requires iterations of forward and backward pass of computations. It also requires storing the entire training trajectory for millions of intermediate parameters in memory, which is unmanageable. Maclaurin et al. (2015) showed that, instead of storing the entire training trajectory, we could recompute the learning trajectory during backpropagation on the fly by storing few auxiliary bits. They demonstrated the idea for training procedure of stochastic gradient descent (SGD) with momentum. Fû et al. (2016) demonstrated that a shortcut could be created to approximate the reverse-mode differentiation step by extracting the knowledge from the forward pass. Their algorithm is 45 times faster and requires 100 times less memory compared to standard methods evaluated on two image datasets.

SVM has a unique optimization problem structure, which can be explored and utilized to optimize its hyper-parameters. Existing methods mostly optimize the regularization hyper-parameter C, not the kernel hyper-parameter due to the nonlinearity and computation complexity induced by . Hastie et al. (2004) created a method that can trace the entire regularization path of SVM solutions for different values of the hyper-parameter C. Exploiting the fact that the dual variables of SVM are piecewise-linear in C, their method has the same computational cost as solving one SVM problem. Bennett et al. (2008) and Kunapuli et al. (2008) formulated hyper-parameter optimization as a bilevel optimization problem. They further reduced the problem to a single optimization problem by replacing the lower-level SVM with its optimality conditions, i.e., Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker (1951)). The resulting single optimization problem is mathematical programming with equilibrium constraints (MPEC). The authors explored various general optimization methods for solving MPEC problems with limited success. The disadvantage of this approach is that the number of constraints grows linearly with the training data size, which makes the optimization problem intractable for large data. Moreover, it doesnt exploit the particular structure of SVM to achieve a more efficient method. Couellan and Wang (2015) took the same bilevel formulation as Bennett et al. (2008) but used SGD with the implicit function theorem to optimize the hyper-parameter C. Numerical results showed that their method is efficient for large datasets by performing cheap gradient estimates using SGD. However, SGD itself requires additional hyper-parameters, i.e., the learning rates. As Couellan and Wang (2015)'s bilevel-SGD method requires performing SGD both on the upper-level and lower-level problems, it requires two additional hyper-parameters to tune.

To improve the bilevel-SGD method, we further exploited the structure of SVM and developed a method that converges faster with better predictive performance. Precisely, we adopted a dual coordinate descent method (DCD) (Hsieh et al. (2008)) for the lower-level problem and combined it with SGD on the upper-level problem. Our approach avoids introducing additional hyper-parameters for the lower-level problem. Numerical results on multiple benchmark datasets show our method converges fast and achieves good generalization performance. First, we present our problem formulation. Then we explain our approach and algorithms used in detail. Finally, we compare our approach with the bilevel-SGD method and show the empirical results on multiple benchmark datasets.

2 Problem Setting

2.1 SVM Optimization Problem

For a binary classification problem, lets define the training data as $\{x_i \in R^p, i = 1,...,N\}$ and the binary labels as $\{y_i \in \{-1,1\}, i = 1,...,N\}$, where p is the number of features and N is the number of samples. We denote vectors in bold symbols, matrices in capital letters and scalars in unbolded lower letters. SVM is a maximum margin classifier that uses a hyperplane $x : f(x) = x^T\beta + \beta_0 = 0$ to separate the feature space, where β and β_0 are the parameters to be learned. The decision function or classification rule is $G(x) = \text{sign}[f(x)] = \text{sign}[x^T\beta + \beta_0]$.

For the hard-margin case, where the data is linearly separable, the SVM optimization problem is:

$$\begin{aligned} &\underset{\beta,\beta_0}{\text{Min}} \|\beta\| \\ \text{subject to} \quad & y_i({x_i}^T\beta+\beta_0) \geq 1, i \in \{1,...N\} \end{aligned} \tag{1}$$

where $\frac{1}{\|\beta\|}$ is the margin size between the data points and the separating hyperplane. The constraints ensure the training data was classified to be on the correct side of the two margins, i.e. $|({x_i}^T\beta+\beta_0)| = 1$. The two margins are two separating hyperplanes $\{x : (x^T\beta+\beta_0) = 1\}$ and $\{(x^T\beta+\beta_0) = -1\}$ that separates the training data into two classes and $\frac{2}{\|\beta\|}$ is the margin size between the two separating hyperplanes.

For the linearly non-separable case, hinge loss is introduced into Eq. 1. The optimization problem becomes minimizing the sum of the hinge loss and the inverse of margin size as follows:

$$\underset{\beta,\beta_0}{\text{Min}} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \max\{0, 1-y_i({x_i}^T\beta+\beta_0)\} \tag{2}$$

where $\max\{0, 1-y_i({x_i}^T\beta+\beta_0)$ is the hinge loss for classifying the training example. The loss is positive when the training example is incorrectly classified, i.e., when $y_i({x_i}^T\beta+\beta_0) < 1$. Various SGD methods have been proposed to efficiently solve formulation (2) for large-scale datasets (Shalev-Shwartz et al. (2011), Zhang and Tong (2004)).

The hinge loss function in (2) makes the objective function non-smooth. Classically, the problem (2) has been reformulated into a constrained quadratic programming problem by introducing a slack variable ξ. The optimization problem can be reformulated as:

$$\begin{aligned} &\underset{\beta,\beta_0,\xi_i}{\text{Min}} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i({x_i}^T\beta+\beta_0) \geq 1-\xi_i \quad \text{(dual variable:}\alpha_i\text{)} \\ & \xi_i \geq 0, i \in \{1,...N\} \quad \text{(dual variable:}\gamma_i\text{)} \end{aligned} \tag{3}$$

2.2 SVM Dual Problem

Eq. (3) is the primal problem for SVM. Its dual problem is a more straightforward quadratic programming problem that only involves one set of decision variables α_i. The dual problem is typically solved instead. Efficient decomposition and coordinate descend methods exit for solving it with large-scale datasets (Hsieh et al. (2008), Platt (1998)). The constraints for the primal problems are all linear constraints, and its objective function is convex. Therefore, Slaters condition is satisfied which implies strong duality holds (Boyd and Vandenberghe (2004)). As a result, we can solve the dual problem and obtain the same solution as if we solve the primal problem. Moreover, the KKT conditions are sufficient and necessary for optimality.

The dual problem for SVM in Eq. (3) is:

$$\begin{aligned} \underset{\alpha_i}{\text{Max}} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{N} \alpha_i y_i = 0, \quad i, j \in 1, ..., N \end{aligned} \tag{4}$$

Platt (1998) created one of the first decomposition methods, sequential minimal optimization (SMO), to efficiently solve the dual problem (Platt (1998)).

By including the bias term β_0 into the vector and append another constant column filled with 1 to the input data, the problem (4) can be further simplified (Hsieh et al. (2008)). Specifically, after the following operation:

$${x_i}^T \leftarrow [{x_i}^T, 1], \quad \beta^T \leftarrow [\beta, \beta_0] \tag{5}$$

The dual problem is reformulated as follows:

$$\begin{aligned} \underset{\alpha_i}{\text{Max}} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \\ & = e^T \alpha - \frac{1}{2} \alpha^T Q \alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i, j \in 1, ..., N \end{aligned} \tag{6}$$

where $Q_{ij} = y_i y_j {x_i}^T x_j$, and e is a vector of one.

Hsieh et al. (2008) proposed a dual coordinate descent method to solve the slightly reformulated problem in (6). Their approach is well suited for large-scale linear SVM problems. It takes $O(log(1/\varepsilon))$ iterations for their DCD method to reach an ε-accurate solution.

3 Bilevel Problem Formulation

In general, a bilevel problem is expressed as follows:

$$\begin{aligned} \underset{x,\bar{y}}{\text{Min}} \; & F(x, \bar{y}) \\ \text{subject to} \quad & G(x, \bar{y}) \leq 0 \\ & \bar{y} = \underset{x,\bar{y}}{argmin} \{ f(x, y) : g(x, y) \leq 0 \} \end{aligned} \tag{7}$$

where $\bar{y}$ is the optimal solution for the lower-level optimization problem and x and $\bar{y}$ both decision variables for the upper-level optimization problem.

3.1 One-Fold Validation

Usually, the hyperparameter is tuned by maximizing model performance on the hold-out dataset using k-fold cross-validation (Ron Kohavi (1995)). Lets first consider a simple case using the same framework where we optimize the hyperparameter to maximize the model prediction performance on one separate validation set (N samples). Precisely, on the upper level, we are minimizing prediction loss on the validation set. On the lower level, we are solving the SVM problem itself. Lets again ignore the bias term β_0 using operation (5) and take the original SVM primal problem formulation, Eq. (2), as the lower-level problem. The bilevel problem formulation for hyper-parameter optimization for SVM becomes:

$$\begin{aligned} \underset{C,\bar{\beta}}{\text{Min}} & F(C,\bar{\beta}) = \sum_{j=1}^{N} \max\{0, 1 - y_i^v(\bar{\beta}^T x_j{}^v)\} \\ \text{subject to} \quad & C_{\max} \geq C \geq 0 \\ & \bar{\beta} = \underset{\beta}{argmin}\{G(C,\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{L} \max\{0, 1 - y_i(x_i{}^T\beta)\}\} \end{aligned} \tag{8}$$

where L is the sample size of training set, N is the sample size of validation set, x^v and y^v are data in the validation set.

3.2 Related Work

Couellan and Wang (2015) solved the above problem (8) using SGD to optimize both the hyper-parameter C on the upper level and the SVM parameter β on the lower level. The key challenge is to use SGD is to obtain the gradient estimate of $F(C,\bar{\beta})$ with respect to hyper-parameter C, i.e., $\nabla_C F(C,\bar{\beta}(C))$. To deal with this, they used the implicit function theorem and chain rule for deriving derivatives. Using chain rule, we obtain:

$$\nabla_C F(C,\bar{\beta}(C)) = \nabla_C F(C,\bar{\beta})^T + \nabla_{\bar{\beta}} F(C,\bar{\beta})^T \nabla_C \bar{\beta}(C) \tag{9}$$

When the lower-level problem reaches the optimum solution, the sub-gradient equals zero, i.e.,

$$\nabla_\beta G(C,\beta) = \beta - C\sum_{l=1}^{L_e} y_l x_l = 0 \tag{10}$$

where l is the index of the sample whose training loss is positive, i.e., $\{l = 1,...,L_e | y_l \beta^T x_l < 1\}$, as the for the other correctly classified samples, the sub-gradient is just zero. They used SGD to solve the lower-level problem and randomly sampled one data instance to estimate the sub-gradient for each iteration. Eq. (10) was further approximated by:

$$\nabla_\beta G_l(C,\beta) = \beta - C y_l x_l = 0 \tag{11}$$

Using this optimality condition for the lower-level problem, they applied implicit function theorem to obtain $\nabla_C \bar{\beta}(C)$.

Using implicit function theorem: assume we have a continuously differentiable function $f(C,\beta)$ and value C in open set $U \in R^n$, β in open set V^m, such that $f(C,\beta) = 0$. Then there is a unique continuously differetiable function $y : U \to V$, such that $\beta = y(C)$

and $f(C,y(C))=0, \forall C \in U$. Further, differentiating equation $f(C,\beta)=0$ with respect to C yields:

$$\nabla_\beta f(C,\beta)\nabla_C \beta + \nabla_C f(C,\beta) = 0 \tag{12}$$

Therefore, we have:

$$\nabla_C \beta = -(\nabla_\beta f(C,\beta))^{-1}\nabla_C f(C,\beta) \tag{13}$$

Replacing $f(C,\beta)$ in Eq. (13) with $\nabla_\beta G_l(C,\beta)$ in Eq. (11), we obtain $\nabla_C \bar{\beta}(C)$ as follows:

$$\nabla_C \bar{\beta}(C) = -\nabla^2_{\bar{\beta}} G_l(C,\bar{\beta})^{-1}\nabla^2_{\bar{\beta}C} G_l(C,\bar{\beta}) \tag{14}$$

From Eq. (11) we can derive $\nabla^2_{\bar{\beta}} G_l(C,\bar{\beta}) = 1$ and $\nabla^2_{\bar{\beta}C} G_l(C,\bar{\beta}) = y_l x_l$. Eq. (14) can be simplified as:

$$\nabla_C \bar{\beta}(C) = y_l x_l \tag{15}$$

They also used SGD to optimize the upper-level problem and estimated the sub-gradient $\nabla_{\bar{\beta}} F(C,\bar{\beta})$ using one random data instance as an unbiased noisy gradient estimate. More specifically:

$$\nabla_{\bar{\beta}} F(C,\bar{\beta}) = -y^v_q {x_q}^v, \text{where } q \in \{1,...,N_e | y^v_q \beta^T {x_q}^v < 1\} \tag{16}$$

Inserting Eq. (16) and $\nabla_C F(C,\bar{\beta}) = 0$ into Eq. (9), they obtained:

$$\nabla_C F(C,\bar{\beta}(C)) = -y^v_q x^v_q y_l x_l \tag{17}$$

Then they optimized both the hyper-parameter C on the upper level and model parameters on the lower level iteratively using simple SGD (Couellan and Wang (2015)). The disadvantage of their approach is that they introduced two new hyper-parameters, the learning rates for the upper-level SGD and lower-level SGD parameter updates. The learning rates themselves are hard to tune as well.

3.3 Our Approach

To further improve their method, we propose an approach to combine SGD and DCD methods to optimize the hyper-parameter C. We choose to optimize the SVM dual problem formulation on the lower level using a DCD method proposed by Hsieh et al. (2008), while keep using SGD to optimize the hyper-parameter C on the upper level.

The bilevel hyper-parameter optimization problem we are solving becomes:

$$\begin{aligned}
\min_{C,\bar{\alpha}} F(C,\bar{\alpha}) &= \sum_{j=1}^{N} \max\left\{0, 1 - y^v_j(\beta^T x^v_j)\right\} \\
&= \sum_{j=1}^{N} \max\{0, 1 - y^v_j((\sum_{i=1}^{L} \bar{\alpha}_i y_i x_i)^T x^v_j)\} \\
&= \sum_{j=1}^{N} \max\{0, 1 - y^v_j((M\bar{\alpha})^T x^v_j)\} \\
\text{subject to} \quad & C_{\max} \geq C \geq 0 \\
& \bar{\alpha} = \underset{\alpha}{argmax}\left\{ f(\alpha) = e^T\alpha - \frac{1}{2}\alpha^T Q \alpha : 0 \leq \alpha_i \leq C, \forall i \in \{1,...,L\} \right\}
\end{aligned} \tag{18}$$

where $Q_{ij} = y_i y_j x_i^T x_j$, and e is a vector of one, $M_{*,j} = y_j x_j$ (in training set).

Using chain rule, we have:

$$\nabla_C F(C, \alpha(C)) = \nabla_C F(C, \alpha)^T + \nabla_\alpha F(C, \alpha(C))^T J_\alpha(C) \tag{19}$$

Following the framework by Couellan and Wang (2015), stochastic gradient descent can be used to obtain a noisy unbiased estimate of the gradient by randomly choose one data sample q in $F(C, \alpha)$.

$$\nabla_C F_q(C, \alpha(C)) = \nabla_C F_q(C, \alpha)^T + \nabla_\alpha F_q(C, \alpha(C))^T J_\alpha(C) \tag{20}$$

From the objective function of problem (18), we have $\nabla_C F_q(C, \alpha)^T = 0$ and $\nabla_\alpha F_q(C, \alpha(C))^T = -y_q^v (x_q^v)^T M$.

Therefore, Eq. (20) becomes:

$$\nabla_C F_q(C, \alpha(C)) = -y_q^v (x_q^v)^T M J_\alpha(C) \tag{21}$$

The key is how to obtain $J_\alpha(C)$. As we use the dual coordinate descent method by Hsieh et al. (2008) to solve the lower level problem, the optimality condition for the lower problem is that the projected gradient is zero, i.e., $\nabla^P f(\bar{\alpha}) = 0$, where $f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - e^T \alpha$. According to Hsieh et al. (2008), we have:

$$\nabla_i^P f(\alpha) = \begin{cases} \nabla_i f(\alpha) & \text{if } 0 < \alpha_i < C \\ \min(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = 0 \\ \max(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = C \end{cases} \tag{22}$$

The only α_i that are a function of C are the ones that are equal to C, i.e., the constraint $\alpha_i \leq C$ is active when optimality condition is satisfied. Therefore,

$$J_\alpha(C)_i = \begin{cases} 1, & \text{if } \alpha_i = C \\ 0, & \text{if } 0 \leq \alpha_i < C \end{cases} \tag{23}$$

The advantage of our approach is that we did not introduce any additional hyper-parameters to the lower level optimization problem by using the DCD method (Hsieh et al. (2008)). As the DCD method is very efficient for large-scale SVM problems, we believe the new approach to be computational fast, while achieving the optimal solution for hyper-parameter C. The specific algorithm for our approach is shown in Algorithm 1 below. As we combined SGD with DCD in Algorithm 1, we also call Algorithm 1 the SGD+DCD method in the following section.

Algorithm 1: Stochastic gradient descent combined with dual coordinate descent method for Linear SVM, SGD+DCD

Set C_{min}, C_{max}, initialize C, α and the corresponding $\beta = \sum_1^L \alpha_i y_i x_i$, $M_{*,j} = y_j x_j$;
while *Stopping criteria not satisfied* **do**
  Pick l randomly from training set $T_e = \{1, ..., L\}$;
  if α *is not optimal* **then**
    $\bar{\alpha}_l \leftarrow \alpha_l$;
    $\nabla_l f(\alpha) = y_l \beta^T x_l - 1$;
    $\nabla_l^P f(\alpha) = \begin{cases} min(\nabla_l f(\alpha), 0) & \text{if } \alpha_l = 0 \\ max(\nabla_l f(\alpha), 0) & \text{if } \alpha_l = C \\ \nabla_l f(\alpha) & \text{if } 0 < \alpha_l < C \end{cases}$;
    if $|\nabla_l^P f(\alpha)| \neq 0$ **then**
      $\alpha_l \leftarrow min(max(\alpha_l - \nabla_l f(\alpha)/Q_{ll}, 0), C)$;
      $\beta \leftarrow \beta + (\alpha_l - \bar{\alpha}_l) y_l x_l$;
  $J_\alpha(C)_i = \begin{cases} 1, & \text{if } \alpha_i = C \\ 0, & \text{if } 0 \leq \alpha_i < C \end{cases}$
  if $J_\alpha(C)$ *are not all zeros* **then**
    Pick q randomly from validation set $V_e = \{j = 1, ..., N | y_j^v \beta^T x_j{}^v < 1\}$;
    Compute $\nabla_\alpha F_q(C, \alpha(C)) = -y_q^v (x_q^v)^T M J_\alpha(C)$;
    $C \leftarrow C - \eta_C \nabla_\alpha F_q(C, \alpha(C))$;
    if $C < C_{min}$ **then**
      $C = C_{min}$;
    if $C > C_{max}$ **then**
      $C = C_{max}$;

3.4 K-fold Cross-validation

In practice, k-fold cross-validation is the standard approach to tune hyper-parameters (Ron Kohavi (1995)). We can easily extend the one-fold validation problem formulation in the previous section to the k-fold cross-validation case. Lets assume we randomly split the dataset into k folds. Denote the data in the k^{th} validation fold as $(x_i^{v_k}, y_i^{v_k}), i \in \{1, 2, , N\}$, where N is the number of samples in the validation fold. The rest of the dataset, i.e., $k-1$ folds are the training data. Let's denote the training data as $(x_i^{t_k}, y_i^{t_k}), i \in \{1, 2, , L\}$, where L is the number of samples in the training data.

The objective function on the upper level becomes minimizing the average loss over the k validation folds. On the lower level, we are minimizing the loss for each of the k

corresponding training data. The bilevel hyper-parameter optimization problem becomes:

$$\begin{aligned}
\underset{C,\bar{\alpha}_k}{\text{Min}} F(C,\bar{\alpha}_k) &= \frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{N_k}\max\{0,1-y_i^{v_k}((M_k\bar{\alpha}_k)^T x_j^{v_k})\} \\
&= \frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{N_{e_k}}(1-y_i^{v_k}((M_k\bar{\alpha}_k)^T x_j^{v_k})) \\
\text{subject to} \quad & C_{\max}\geq C\geq 0 \\
& \bar{\alpha}_k = \underset{\alpha_k}{argmin}\{f(\alpha_k)=e^T\alpha_k-\frac{1}{2}\alpha_k^T Q_k\alpha_k : 0\leq\alpha_{k_i}\leq C\}, \\
& \forall i\in\{1,...,L_k\}, k\in\{1,...,K\}
\end{aligned} \tag{24}$$

where $Q_{k_{ij}} = y_i^{t_k} y_j^{t_k} x_i^{t_k T} x_j^{t_k}$, $M_{k_{*,j}} = y_j^{t_k} x_j^{t_k}$ and e is a vector of one.

Given the value of C, all the lower-level optimization problems are independent. Therefore, we can update the variable α_k for each lower-level problem independently. The only modification of the SGD+DCD method needed to perform K-fold cross-validation is at step 15 in Algorithm 1. Here we update C using the averaged gradient from all the training folds instead, i.e., using $\frac{1}{K}\sum_{k=1}^{K}\nabla_{\alpha_k}F_p(C,\alpha_k(C))$.

Next, we are going to illustrate the effectiveness of our approach with numerical results on multiple benchmark datasets. All the experiments were conducted using 5-fold cross-validation.

4 Numerical Experiment

There are two main implementation issues that need to be clarified. First, a stopping criterion needs to be chosen. In practice, several stopping criteria are used. For instance, stop the algorithm when the accuracy exceeds a predefined threshold. Or simply stop the algorithm when the maximal number of iterations is reached. To compare the two methods, we used the second stopping criterion.

Another implementation issue is how to initialize all the variables. For our SGD+DCD method, we initialized as zero vector as suggested by Hsieh et al. (2008) because the solution of α typically has only a few nonzero elements, which are the support vectors. For the bilevel-SGD method, we initialized all the variables following Couellan and Wang (2015)'s study. Specifically, we randomly initialized the β from a $[0,1]$ uniform distribution. Learning rate for the lower lever SGD was chosen to be $1/t$, and learning rate for the upper level SGD was set as $\frac{1}{t\sqrt{p}|\nabla_C F_p(C,\beta(C))|}$, where t is the t^{th} iteration and p is the number of features. Our SGD+DCD method also used $\frac{1}{t\sqrt{p}|\nabla_C F_p(C,\alpha(C))|}$ as the learning rate η_C for the upper level SGD. For both methods, we set $C_{min}=10^{-4}$, $C_{max}=10^{6}$, and initialize hyper-parameter C to be same as C_{min}.

The datasets we used are the following:

1. Cancer: Wisconsin diagnostic breast cancer dataset. The two classes are a benign or malignant diagnosis. Number of samples: 699, number of features 9. Source: UCI Machine Learning Repository (1995b).
2. Pima: Diabetes data. The prediction target is whether a patient has positive or negative diabetes diagnosis. Number of samples: 768, number of features: 8. Source: UCI Machine Learning Repository (1998).

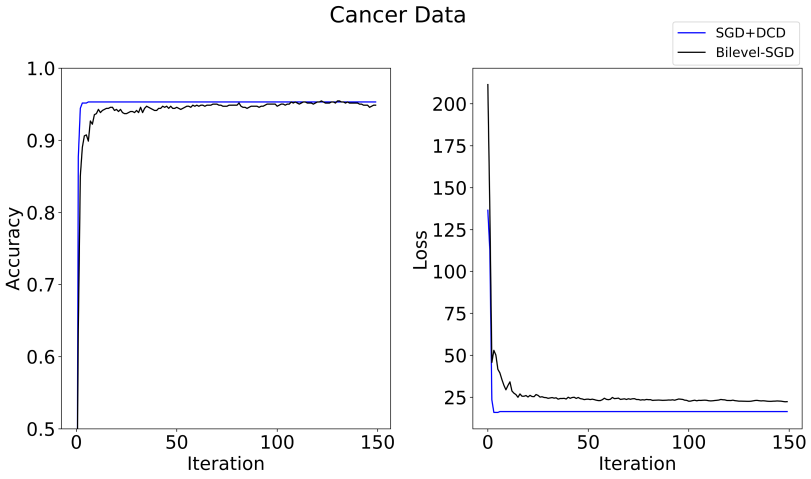
3. SVMguide1: astroparticle dataset with a binary class label. Number of samples: 3089, number of features: 4. Source: LIBSVM a Library for Support Vector Machines (Chang and Lin (2013)).
4. Connect: connect-4 game dataset. The prediction target is win or loss. Number of samples: 67557, number of features: 42. Source: UCI Machine Learning Repository (1995a).
5. Magic04: gamma telescope dataset. The prediction target is whether a signal is high energy gamma signal or background signal. Number of samples: 19020, number of features: 10. Source: UCI Machine Learning Repository (2007).
6. Xerostomia: head and neck cancer radiotherapy side effect dataset. The prediction target is whether a patient develops acute xerostomia after radiotherapy. Number of samples: 551, number of features: 943. Source: Oncospace JHU (McNutt, Wong J, Purdy J, et al. (2010))
7. Xerostomia recovery: head and neck cancer radiotherapy side effect dataset. The prediction target is whether a patient developed xerostomia recovered at 18 months post radiotherapy. Number of samples: 146, number of features: 943. Source: Oncospace JHU (McNutt, Wong J, Purdy J, et al. (2010))

All dataset features were preprocessed and scaled to be in the range of $[-1,1]$. The class labels were coded as $y \in \{-1,1\}$. For 5-fold cross-validation, we created the five folds using stratified sampling to ensure each fold has the same distribution of the prediction labels. Table 1 reports the numerical results of the two methods. Accuracy is the prediction accuracy on the validation folds (hold-out data) by doing 5-fold cross-validation. C is the final value obtained for the hyper-parameter. Running time is the total CPU time (Intel Core i7-7700HQ) spent on training the SVM model. We obtained the results presented in Table 1 by running each of the methods for 150 iterations.

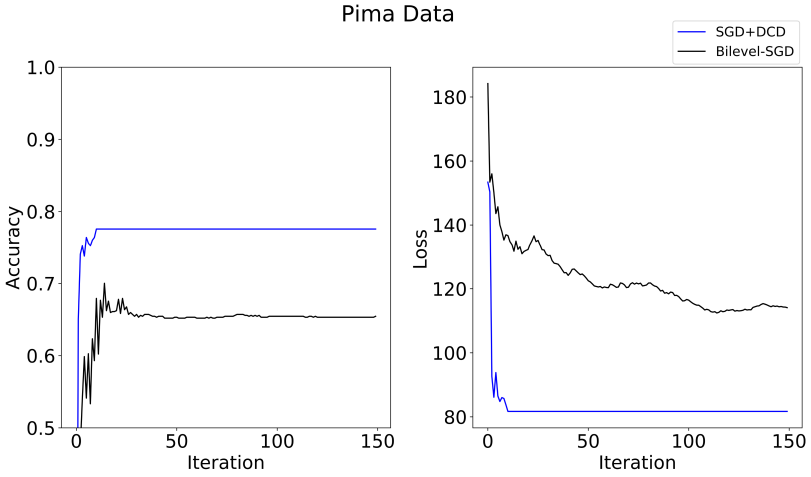
Table 1: Numerical results of the bilevel-SGD method and the SGD+DCD method.

Dataset	Datasize	Bilevel-SGD			Algorithm 1, SGD+DCD		
		Accuracy (%)	C	Running time (s)	Accuracy (%)	C	Running time (s)
Cancer	(699, 9)	94.29	1.24	0.30	**95.02**	0.16	0.49
Pima	(768, 8)	65.58	0.38	0.33	**76.53**	0.21	0.91
SVMguide1	(3089, 4)	72.81	1.06	1.21	**84.88**	0.29	2.43
Connect	(67557, 42)	72.77	0.0001	89.36	**83.68**	0.06	148.39
Magic04	(19020, 10)	66.97	0.26	22.80	**78.55**	0.54	47.39
Xerostomia	(551, 943)	68.97	0.10	2.56	**69.51**	0.01	2.57
Xerostomia recovery	(146, 943)	**81.58**	0.12	0.39	78.75	0.01	0.45

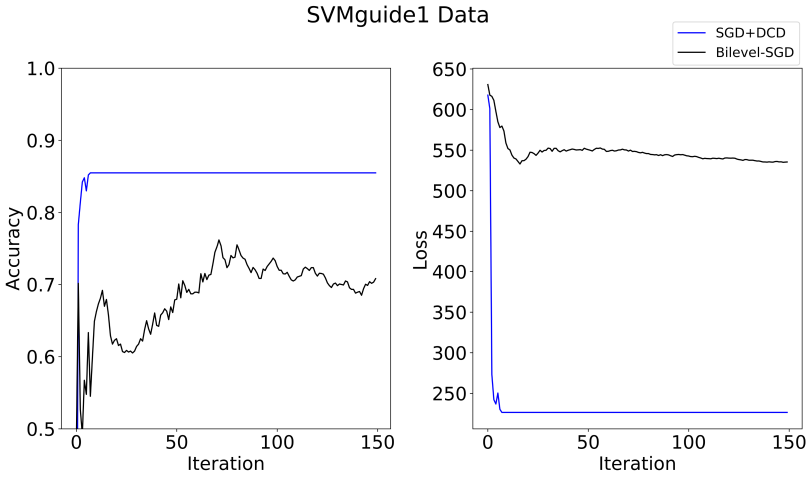
From Table 1, it's easy to see that, with a slight increase of running time, the SGD+DCD method achieved higher accuracy on the validation folds for all datasets except the xerostomia recovery dataset. For the Cancer dataset, xerostomia and xerostomia recovery dataset, the accuracies result is about the same. For all the other datasets, the SGD+DCD methods


Cancer Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

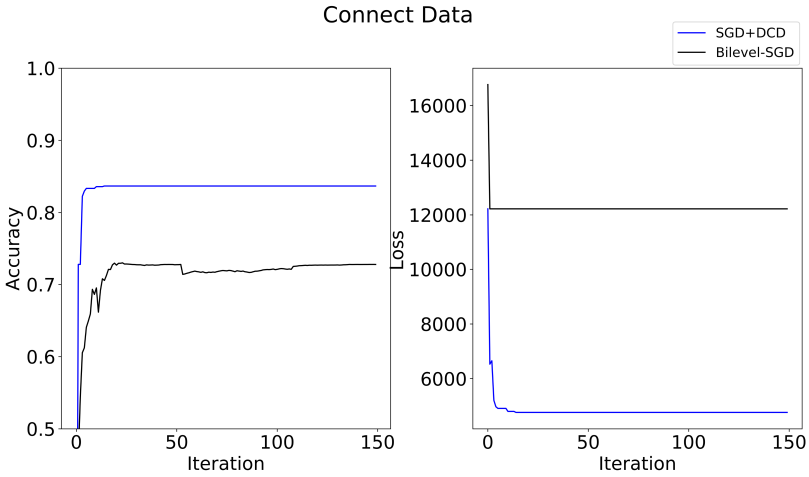
(a)


Pima Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

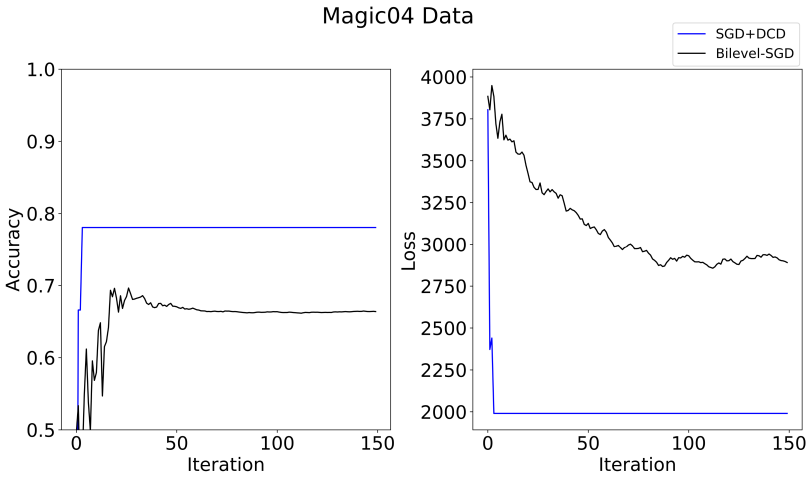
(b)


SVMguide1 Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

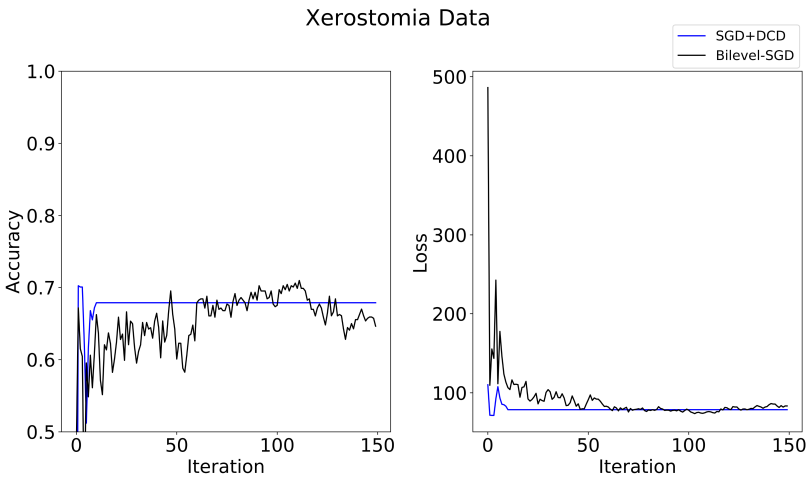
(c)


Connect Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

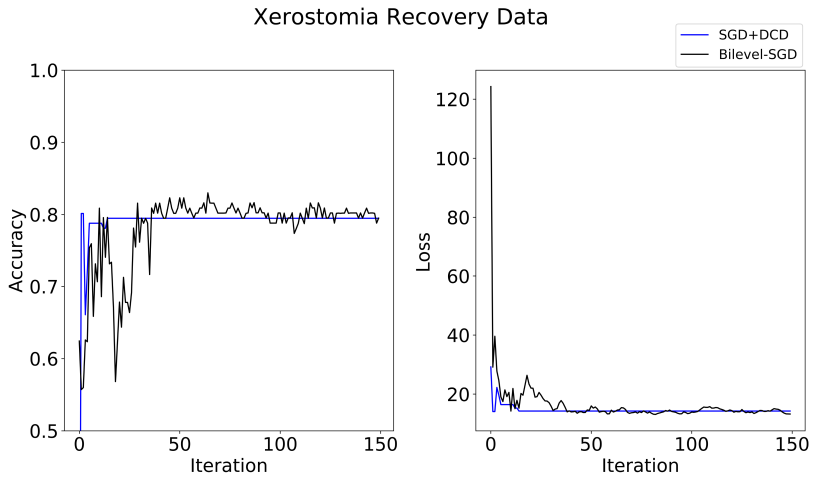
(d)


Magic04 Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

(e)


Xerostomia Data
SGD+DCD
Bilevel-SGD
Accuracy
Loss
Iteration
Iteration

(f)



(g)

Fig. 1: The numerical results of prediction accuracy and loss on the validation folds for Algorithm 1 (our SGD+DCD) method and the bilevel-SGD method on seven datasets: (a) Cancer data, (b) Pima data, (c) SVMguide1 data, (d) Connect data, (e) Magic04 data, (f) Xerostomia data, (g) Xerostomia recovery data.

accuracies are more than 10% higher than the bilevel-SGD method. For high dimensional datasets, xerostomia and xerostomia recovery, the SGD+DCD has about the same running time with the bilevel-SGD method. For the other non-high dimensional datasets, the bilevel-SGD is relatively faster.

To further compare the convergence results, we plotted the accuracy and loss profile on the three datasets for both methods in Fig. 1. We can see that the SGD+DCD method converges to the optimal solution much faster than the bilevel-SGD method. Also, the SGD+DCD has higher accuracy and a lower loss on the validation fold than the bilevel-SGD method for all three datasets.

5 Discussion

Even though the SGD+DCD converges to an optimal solution very fast, its computationally more expensive than the bilevel-SGD method. Bilevel-SGD is much faster because it uses SGD to optimize both the lower and upper-level problems. As we know, SGD is fast due to cheap gradient updates. For our SGD+DCD method, we optimized the lower-level problem using DCD. DCD method iterates through all the variables on the lower level for each outer iteration. For DCD, the number of variables on the lower-level equals the number of training examples. Therefore, when the number of training examples is much larger than the number of features, DCD requires longer running time (Hsieh et al. (2008)). All the datasets we used (except the Xerostomia and Xerostomia Recovery data) are all low dimensional datasets. The largest number of features is 42 (the Connect data), which is much smaller than its number of samples (699). DCD is well suited for high dimensional datasets where the number of features is larger than the number of samples (Hsieh et al. (2008)). As a re-

sult, we believe the SGD+DCD method will be computationally more efficient to tune the hyper-parameter for a high dimensional dataset. This was further shown by the results on the high dimensional Xerostomia and Xerostomia Recovery datasets.

There are two main reasons that our SGD+DCD method achieves better accuracy performance than the bilevel-SGD method. Using SGD for optimizing both the hyper-parameter C on the upper level and SVM parameters on the lower level is fast due to cheap gradient estimates. However, there are assumptions and approximations behind this approach. The first reason is that, for each iteration of updating hyper-parameter C, we are assuming the lower-level SVM problem has already reached its optimal solution. This is apparently not true, especially in the beginning iterations when the lower-level SVM problem is far from converging to the optimal solution. To follow the bilevel problem formulation exactly, before each iteration of updating the value of C using SGD, we should let the lower-level SVM problem reaches its optimal solution. Regarding this assumption, our method is more appropriate than Couellan and Wang (2015)'s method as the DCD method converges much faster than the SGD method for solving the lower-level SVM problem.

The second reason is that Couellan and Wang (2015)'s method approximated the optimality condition of the lower-level problem using only one randomly sampled data instance when using SGD to optimize C. Precisely, they estimated the stochastic gradient using Eq. (11), which is an approximation of the true optimality condition (Eq. (10)). Here the stochastic gradient was computed using the implicit function theorem on the optimality condition. This is different from a regular SGD problem, where the stochastic gradient is usually computed from a loss function. We utilized the implicit function theorem as well to obtain the gradient of lower-level SVM parameter α with respect to C, i.e., $J_\alpha(C)$. However, to obtain $J_\alpha(C)$, we didn't approximate the optimality condition of the lower-level SVM problem. Instead, we computed the exact gradient of $J_\alpha(C)$ using Eq. (22) and Eq. (23). This is both an exact and computationally cheap gradient calculation as most of the elements in $J_\alpha(C)_i$ is zero when the lower-level SVM problem reaches its optimal solution.

In summary, we developed an efficient hyper-parameter optimization method to optimize the regularization parameter C for supoort vector machine by combining stochastic gradient descent and dual coordinate desent method. Several benchmark datasets show that our method yields consistently higher out-of-sample accuracy performance than an existing bilevel-SGD method with slightly increased running time.

References

Bengio Y (1999) Gradient-Based Optimization of Hyper-Parameters URL https://pdfs.semanticscholar.org/0b2b/806201fd1146f16b4c20fc3dd696e22c3c88.pdf

Bennett KP, Kunapuli G, Hu J, Pang JS (2008) Bilevel optimization and machine learning. In: IEEE World Congress on Computational Intelligence, Springer, pp 25–47

Boyd SP, Vandenberghe L (2004) Convex optimization. Cambridge University Press

Chang CC, Lin CJ (2013) LIBSVM: A Library for Support Vector Machines URL https://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf

Cortes C, Vapnik V (1995) Support-vector networks. Machine Learning 20(3):273–297, DOI 10.1007/BF00994018, URL http://link.springer.com/10.1007/BF00994018

Couellan N, Wang W (2015) Bi-level stochastic gradient for large scale support vector machine. Neurocomputing 153:300–308, DOI 10.1016/J.NEUCOM.2014.11.025, URL http://www.sciencedirect.com/science/article/pii/S0925231214015586

Do CB, Foo CS, Ng AY (2007) Efficient multiple hyperparameter learning for log-linear models URL http://ai.stanford.edu/ chuongdo/papers/learn_reg.pdf

Franceschi L, Donini M, Frasconi P, Pontil M (2017) Forward and Reverse Gradient-Based Hyperparameter Optimization URL https://arxiv.org/pdf/1703.01785.pdf

Friedman J, Hastie T, Tibshirani R (2001) The elements of statistical learning, vol 1. Springer series in statistics New York

Fû J, Luô H, Fen J, Hsiang Lo K, Chuâ TS (2016) DrMAD: Distilling Reverse-Mode Automatic Differentiation for Optimizing Hyperparameters of Deep Neural Networks URL https://arxiv.org/pdf/1601.00917.pdf

Hastie T, Edu H, Rosset S, Tibshirani R, Zhu J, Edu J (2004) The Entire Regularization Path for the Support Vector Machine. Journal of Machine Learning Research 5:1391–1415

Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S (2008) A dual coordinate descent method for large-scale linear svm. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 408–415

Jules MS (2017) Experiments With Scalable Gradient-based Hyperparameter Optimization for Deep Neural Networks URL https://pdfs.semanticscholar.org/b694/d478bc9b7e4828c8fca4ff23433f9bf5e9d3.pdf

Kuhn HW, Tucker AW (1951) NONLINEAR PROGRAMMING. In: Proc. Second Berkeley Symp. on Math. Statist. and Prob., pp 481–492, DOI 10.1007/978-3-0348-0439-4, URL https://link.springer.com/content/pdf/10.1007/978-3-0348-0439-4_11.pdf

Kunapuli G, Bennett KP, Hu J, Pang Js (2008) Bilevel Model Selection for Support Vector Machines. Notes 45:129–158

Maclaurin D, Duvenaud D, Adams RP (2015) Gradient-based Hyperparameter Optimization through Reversible Learning URL https://arxiv.org/pdf/1502.03492.pdf

McNutt, Wong J, Purdy J, et al (2010) OncoSpace: A new paradigm for clinical research and decision support in radiation oncology. In: Proceedings of the XVIth International Conference on Computers in Radiation Therapy, Amsterdam, Netherlands

Pedregosa F (2016) Hyperparameter optimization with approximate gradient URL http://arxiv.org/abs/1602.02355, 1602.02355

Platt JC (1998) Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Advances in kernel methods pp 185 – 208, DOI 10.1.1.43.4376, arXiv:1011.1669v3

Ron Kohavi (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, International Joint Conferences on Artificial Intelligence, Inc., pp 1137–1143, URL https://dl.acm.org/citation.cfm?id=1643047

Shalev-Shwartz S, Singer Y, Srebro N, Cotter A (2011) Pegasos: primal estimated sub-gradient solver for SVM. Mathematical Programming 127(1):3–30, DOI 10.1007/s10107-010-0420-4, URL http://link.springer.com/10.1007/s10107-010-0420-4

UCI Machine Learning Repository (1995a) Connect-4 Data Set. URL http://archive.ics.uci.edu/ml/datasets/connect-4

UCI Machine Learning Repository (1995b) UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set. URL https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

UCI Machine Learning Repository (1998) UCI Machine Learning Repository. URL https://archive.ics.uci.edu/ml/index.php

UCI Machine Learning Repository (2007) MAGIC Gamma Telescope Data Set. URL https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope

Zhang T, Tong (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Twenty-first international conference on Machine learning - ICML '04, ACM Press, New York, New York, USA, p 116, DOI 10.1145/1015330.1015332, URL http://portal.acm.org/citation.cfm?doid=1015330.1015332