Budgeted Semi-supervised Support Vector Machine

Trung Le*, **Phuong Duong, Mi Dinh**
Faculty of Information Technology
University of Pedagogy, Vietnam
{trunglm, phuongdth, midtt}@hcmup.edu.vn

Tu Dinh Nguyen, Vu Nguyen, Dinh Phung
Centre for Pattern Reccognition and Data Analytics
Deakin University, Australia
{tu.nguyen, v.nguyen, dinh.phung}@deakin.edu.au

Abstract

Due to the prevalence of unlabeled data, semi-supervised learning has drawn significant attention and has been found applicable in many real-world applications. In this paper, we present the so-called Budgeted Semi-supervised Support Vector Machine (BS3VM), a method that leverages the excellent generalization capacity of kernel-based method with the adjacent and distributive information carried in a spectral graph for semi-supervised learning purpose. The fact that the optimization problem of BS3VM can be solved directly in the primal form makes it fast and efficient in memory usage. We validate the proposed method on several benchmark datasets to demonstrate its accuracy and efficiency. The experimental results show that BS3VM can scale up efficiently to the large-scale datasets where it yields a comparable classification accuracy while simultaneously achieving a significant computational speed-up compared with the baselines.

1 Introduction

Supervised learning constitutes one of the most fundamental problems in machine learning. While in no doubt this theory has been applied successfully to many real-world applications, a key limitation of this theory lies in the requirement of annotated labels during training. However, manual labeling process for large-scale data is labor intensive and error-prone. Consequently, the collected datasets frequently consist of a collection of labeled data jointly with a larger collection of unlabeled data. Semi-supervised learning (SSL) involves utilizing the intrinsic information carried in unlabeled data to enhance the generalization capacity of the learning algorithms. During the past decade, SSL has attracted significant attention and has been found applicable in a variety of problems including text categorization [Joachims, 1999], and bioinformatics [Kasabov et al., 2005] to name a few.

A notable approach to semi-supervised learning paradigm is to employ a spectral graph for representing the adjacent and distributive information in data. Several existing works have leveraged on the expressiveness of spectral graphs for SSL, including mincut [Blum et al., 2004], graph random walk [Azran, 2007], manifold regularization [Belkin et al., 2006], and spectral graph [Duong et al., 2015].

Inspired from the seminal work of [Joachims, 1999], a large body of works has attempted to advance kernel methods such as Support Vector Machine (SVM) [Cortes and Vapnik, 1995] within the semi-supervised learning paradigm. The underlying idea is to *solve standard SVM problem while treating the unknown labels as optimization variables* [Chapelle et al., 2008]. This leads to a non-convex optimization problem with a combinatorial explosion of label assignments. A wide spectrum of techniques have been proposed to solve this optimization problem including local combination search [Joachims, 1999], gradient descent [Chapelle and Zien, 2005], convex-concave procedures [Collobert et al., 2006], and deterministic annealing [Sindhwani et al., 2006, Le et al., 2013, Nguyen et al., 2014]. Although these approaches can somehow deal with the combinatorial intractability, their requirement of repeated retraining the model renders them impractical for many real-world problems.

At the intersection between kernel method and the spectral graph theory, several existing works have attempted to incorporate information carried in a spectral graph to build a better kernel function [Smola and Kondor, 2003, Zhu et al., 2004]. These works basically employed the Laplacian matrix induced from the spectral graph to construct the kernel function which can capture the features of ambient space. Manifold regularization framework [Belkin et al., 2006] exploited the geometry property of the probability distribution that generates data and incorporated it as an additional regularization term. Two regularization terms

*This work was carried out when the first author was supported and visiting the Centre for Pattern Recognition and Data Analytics, Deakin University, Australia in 2015.

were introduced to control the complexity of the classifier in the ambient space and to control the complexity measured by the geometry property of the data distribution. However, the computational complexity for manifold regularization approach is cubical in the training size. Hence, other approaches have been proposed to scale it up [Tsang and Kwok, 2006, Melacci and Belkin, 2011]. In particular, the work of [Melacci and Belkin, 2011] (the Laplacian Support Vector Machine or LapSVM) made use of preconditioned conjugate gradient to solve the optimization problem of manifold regularization approach in the primal form. This allows the computational complexity to be scaled up to quadratic. However, this approach was to solve the corresponding optimization problem in the first dual layer instead of in the primal form, hence renders the solution infeasible for online setting. Furthermore, LapSVM requires storing a full Hessian matrix of the training size in the memory, resulting in a quadratic memory complexity.

Recently, stochastic gradient descent (SGD) method [Shalev-Shwartz and Singer, 2007, Kakade and Shalev-Shwartz, 2008, Lacoste-Julien et al., 2012, Rakhlin et al., 2012, Hazan and Kale, 2014] has emerged as a promising framework to scale up fast and online learning algorithms. SGD possesses three key advantages: i) very fast; ii) capacity to run in online mode; and iii) economic memory usage. However, SGD-based methods are vulnerable to the *curse of kernelization* [Wang et al., 2012], that is, the model size linearly grows up with the training size accumulated over time. To bound the model size, budget-based algorithms limit the model size to a predefined budget B. When the model size exceeds the budget, a budget maintenance procedure is invoked to decrease the model size by one. Three widely-used budget maintenance strategies are removal, projection, and merging. In the removal strategy, the most redundant support vector is simply discarded [Crammer et al., 2004, Cavallanti et al., 2007, Wang and Vucetic, 2010, Le et al., 2016]. In the projection strategy, the information of the most redundant support vector is partly preserved through its projection onto the linear span of the remaining support vectors [Wang and Vucetic, 2010, Wang et al., 2012, Le et al., 2016]. The merging strategy first selects two vectors, and then merges them into one before discarding them [Wang et al., 2012].

Leveraging on the advantages of kernel method, spectral graph theory and stochastic gradient descent, we propose in this paper a novel approach to semi-supervised learning, termed as *Budgeted Semi-supervised Support Vector Machine* (BS3VM). To devise BS3VM, we first conjoin the theory of kernel method with the framework of spectral-graph-based semi-supervised learning. This allows us to form a specific optimization problem which involves the core optimization problem of kernel method and simultaneously enables the label propagation. We then apply SGD method to solve the incurred optimization problem directly in the primal form. To avoid the curse of kernelization, we employ two budgets B_l and B_u for the labeled and unlabeled portions. When either the labeled or unlabeled portion in the model exceeds its budget, the corresponding budget maintenance strategy will be invoked accordingly. We also establish a rigorous convergence analysis for BS3VM. The theoretical result shows that there exists a gap between the proposed and optimal solutions. This gap can be explicitly quantified and crucially depends on the budget maintenance rates and the coefficients accompanied with the removed vectors. We further establish the extensive experiments on several real-world datasets. The experimental results show that our proposed BS3VM can offer a comparable predictive performance while simultaneously achieving a significant computational speed-up comparing with the state-of-the-art baselines.

2 Spectral-graph-based Semi-supervised Learning

2.1 Spectral Graph

Spectral graph is a useful tool to capture the geometrical and distributive information carried in data. It is usually an undirected graph whose vertices are data instances. In the context of semi-supervised learning, we are given a training set $X = X_l \cup X_u$ where $X_l = \{(x_i, y_i)\}_{i=1}^{l}$ specifies labeled data and $X_u = \{x_i\}_{i=l+1}^{l+u}$ identifies unlabeled data. We can start constructing the spectral graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with the vertex set $\mathscr{V}$ including all labeled and unlabeled instances (i.e., $\mathscr{V} = \{x_i\}_{i=1}^{l+u}$). An edge $e_{ij} = \overline{x_i x_j} \in \mathscr{E}$ between two vertices x_i, x_j represents the similarity of these two instances. Let μ_{ij} be the weight of this edge. The principle is that if μ_{ij} is sufficiently large then two labels y_i, y_j are expected to be the same. The set of edges $\mathscr{G}$ and its weighs can be established using the following ways:

- *Fully connected graph*: every pair of vertices x_i, x_j is connected by an edge. The edge weight decreases when the distance $\left\|x_i - x_j\right\|$ increases. The Gaussian kernel weight function widely used is given by

$$\mu_{ij} = e^{-\left\|x_i - x_j\right\|^2/(2\sigma^2)}$$

  where σ is known as the bandwidth parameter and controls how quickly the weight decreases.

- *k-NN*: each vertex x_i defines its k nearest neighbors (k-NN) and makes an edge with one of its k-NN. The Gaussian kernel weight function can be used for the edge weight. Empirically, k-NN graphs with small k tend to perform well.

- *ε-NN*: we connect x_i and x_j if $\left\|x_i - x_j\right\| \leq \varepsilon$. Again the Gaussian kernel weight function can be used to weight the connected edges. In practice, ε-NN graphs are easier to construct than k-NN graphs.

It is noteworthy that when constructing the spectral graph, we avoid connecting the edge of two labeled instances since we do not need to propagate the label between them.

2.2 Label Propagation

After constructing the spectral graph, a semi-supervised learning problem is cast to assign labels to the unlabeled vertices. To this end, we need a mechanism to rationally propagate labels from the labeled vertices to the unlabeled ones. The key idea is that if μ_{ij} is large, then the two labels y_i, y_j are expected to be the same.

To assign labels to the unlabeled instances, it is desirable to learn a map $f : \mathscr{X} \longrightarrow \mathscr{Y}$ where $\mathscr{X}$ and $\mathscr{Y}$ are domains of data and label, respectively such that

- $f(x_i)$ is as closest to its label y_i as possible for all labeled instances x_i $(1 \leq i \leq l)$.
- f should be smooth on the whole graph $\mathscr{G}$, i.e., if x_i is very close to x_j (i.e., x_i, x_j are very similar or μ_{ij} is large), the discrepancy between f_i and f_j (i.e., $|f_i - f_j|$) is small.

Therefore, the following optimization problem is proposed to solve

$$\min_f \left(\infty . \sum_{i=1}^{l} (f_i - y_i)^2 + \sum_{(i,j)\in\mathscr{E}} \mu_{ij} \left| f_i - f_j \right| \right) \quad (1)$$

where by convention we define $\infty.0 = 0$ and $f_i = f(x_i)$.

The optimization problem in Eq. (1) peaks its minimum as the first term is exactly 0 and the second term is as smallest as possible. It is therefore rewritten as follows

$$\min_f \left(\sum_{(i,j)\in\mathscr{E}} \mu_{ij} \left| f_i - f_j \right| \right) \quad (2)$$
$$\text{s.t.} : \forall_{i=1}^{l} : f_i = y_i$$

To extend the representation ability of the prediction function f, we relax the discrete function f to be a real-valued. The drawback of the relaxation is that in the solution, $f(x)$ is now real-valued, hence does not directly correspond to a label. This can however be addressed by thresholding $f(x)$ at zero to produce discrete label predictions, i.e., if $f(x) \geq 0$, predict $y = 1$, and if $f(x) < 0$, predict $y = -1$.

3 Budgeted Semi-supervised Support Vector Machine

In this section, we present our proposed Budgeted Semi-supervised Support Vector Machine (BS3VM). We start this section with the introduction of the optimization problem of BS3VM. We then propose SGD-based solution for BS3VM with two budgets for the labeled and unlabeled portions, followed by the convergence analysis.

3.1 BS3VM Optimization Problem

Let $\Phi : \mathscr{X} \longrightarrow \mathscr{H}$ be a transformation from the input space $\mathscr{X}$ to a Reproducing Hilbert Kernel Space (RHKS) $\mathscr{H}$. We use the function $f(x) = \mathbf{w}^{\mathsf{T}}\Phi(x) - \rho = \sum_{i=1}^{l+u} \alpha_i K(x_i, x) - \rho$, where $\mathbf{w} = \sum_{i=1}^{l+u} \alpha_i \Phi(x_i)$ and $K(.,.)$ is kernel function, to predict label. Inspired from the optimization problem in Eq. (2), the following optimization problem is proposed

$$\min_{\mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{l} \sum_{i=1}^{l} \xi_i + \frac{C'}{|\mathscr{E}|} \sum_{(i,j)\in\mathscr{E}} \mu_{ij} \left| f_i - f_j \right| \right) \quad (3)$$
$$\text{s.t.} : \forall_{i=1}^{l} : y_i \left(\mathbf{w}^{\mathsf{T}}\Phi(x_i) - \rho \right) \geq 1 - \xi_i$$
$$\forall_{i=1}^{l} : \xi_i \geq 0$$

where $f_i = \mathbf{w}^{\mathsf{T}}\Phi(x_i) - \rho$.

In the optimization formulation of Eq. (3), we minimize $\frac{1}{2}\|\mathbf{w}\|^2$ to maximize the margin for motivating the generalization capacity. At the same time, we minimize $\sum_{(i,j)\in\mathscr{E}} \mu_{ij} \left| f_i - f_j \right|$ to make the prediction function smoother on the spectral graph.

We rewrite the optimization problem in Eq. (3) in the primal form as follows[1]

$$\min_{\mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{l} \sum_{i=1}^{l} l(\mathbf{w}; z_i) + \frac{C'}{|\mathscr{E}|} \sum_{(i,j)\in\mathscr{E}} \mu_{ij} l_1 \left(\mathbf{w}^{\mathsf{T}}\Phi_{ij} \right) \right) \quad (4)$$

where $z_i = (x_i, y_i)$, $l(\mathbf{w}; x, y) = \max\left\{0, 1 - y\mathbf{w}^{\mathsf{T}}\Phi(x)\right\}$, $\Phi_{ij} = \Phi(x_i) - \Phi(x_j)$, $l_p(t) = |t|^p$ with $t \in \mathbb{R}$, and $p \geq 1$.

3.2 Budgeted SGD-based Solution for BS3VM

We now present the SGD-based solution for the optimization problem in Eq. (4). To resolve the curse of kernelization, we employ two budgets for the labeled and unlabeled portions whose budget sizes are B_l and B_u, respectively. When either the size of labeled or unlabeled portion in the current model exceeds its budget, the corresponding budget maintenance strategy is executed to maintain the model.

Let us denote the objective function in Eq. (4) by $J(\mathbf{w})$. At the iteration t, we construct the instantaneous objective function $J_t(\mathbf{w})$ which is defined as

$$J_t(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C l\left(\mathbf{w}; x_{i_t}, y_{i_t}\right) + C' \mu_{u_t v_t} l_1 \left(\mathbf{w}^{\mathsf{T}}\Phi_{u_t v_t} \right)$$

where i_t is uniformly sampled from $\{1, ..., l\} = [l]$ and the edge (u_t, v_t) connected x_{u_t} and x_{v_t} is uniformly sampled from the set of edges $\mathscr{E}$.

[1]We can eliminate the bias ρ by simply adjusting the kernel.

Inspired from the SGD method, we update $\mathbf{w}_t$ as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t = \mathbf{w}_t - \frac{1}{t} J^{'}(\mathbf{w}_t) = \frac{t-1}{t}\mathbf{w}_t + \frac{C\alpha_t y_{i_t}\Phi(x_{i_t})}{t} + C' \frac{\mu_{u_t v_t}\beta_t(\Phi(x_{u_t}) - \Phi(x_{v_t}))}{t} \quad (5)$$

where $\alpha_t = -l_o^{'}(\mathbf{w}_t; x_{i_t}, y_{i_t})$, $\beta_t = -\text{sign}\left(\mathbf{w}_t^{\mathsf{T}}\Phi_{u_t v_t}\right)$, the learning rate $\eta_t = \frac{1}{t}$, and $g_t = J^{'}(\mathbf{w}_t)$.

It is noteworthy that we denote $o = \mathbf{w}^{\mathsf{T}}\Phi(x)$ which implies $l(\mathbf{w}; x, y)$ is now a function of o, and $l_o^{'}(\mathbf{w}; x, y)$ is the derivative of the loss function w.r.t the variable o.

The update formula shown in Eq. (5) is vulnerable to the curse of kernelization, that is, the model size linearly grows with the data size accumulated over time. To address this issue, we propose to use two budgets for the labeled and unlabeled portions whose sizes are B_l and B_u, respectively.

Algorithm 1 Algorithm for training BS3VM.

```
Input: $B_l, B_u, K(.,.), C, C^{'}, \sigma$
 1: $\mathbf{w}_1 = \mathbf{0}$
 2: $b_l = 0$
 3: $b_u = 0$
 4: for $t = 1$ to $T$ do
 5:     Uniformly sample $i_t$ from $[l]$
 6:     Uniformly sample the edge $(u_t, v_t)$ from $\mathcal{E}$
 7:     Update
            $\mathbf{w}_{t+1} = \frac{t-1}{t}\mathbf{w}_t + \frac{C}{t}\alpha_t y_{i_t}\Phi(x_{i_t}) + \frac{C'}{t}\mu_{u_t v_t}\beta_t(\Phi(x_{u_t}) - \Phi(x_{v_t}))$
 8:     $b_l = b_l + 1 + \mathbb{I}_{u_t \le l} + \mathbb{I}_{v_t \le l}$
 9:     $b_u = b_u + \mathbb{I}_{u_t > l} + \mathbb{I}_{v_t > l}$
10:     if $b_l > B_l$ then
11:         $BM(\mathbf{w}_{t+1}, 'l')$          // labeled portion
12:         $b_l = B_l$
13:     end if
14:     if $b_u > B_u$ then
15:         $BM(\mathbf{w}_{t+1}, 'u')$          // unlabeled portion
16:         $b_u = B_u$
17:     end if
18: end for
Output: $\overline{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$ or $\mathbf{w}_{T+1}$
```

Algorithm 1 presents the pseudocode of BS3VM. The model of BS3VM is represented through the labeled and unlabeled portions whose current sizes are b_l and b_u, respectively. When either b_l or b_u exceeds its budget, a budget maintenance procedure is triggered to maintain the model size (cf. lines 11 and 15 in Algorithm 1). We have two kinds of budget maintenance (BM) which involve the labeled and unlabeled portions, respectively. To differentiate these two kinds of BM, we employ the second argument in BM procedure wherein $'l'$ involves BM for the labeled portion and $'u'$ involves BM for the unlabeled portion. In addition, two options (i.e., $'l'$ or $'u'$) involve the same functional activity. The only difference is that they refer to either labeled or unlabeled portions. In addition, we utilize the fully connected spectral graph wherein the edge weights are computed on the fly as necessary.

3.3 Budget Maintenance Strategy

In this section, we present the BM strategies used in this paper which are removal and projection. The original projection strategy can partly preserve the information of the removed vectors and hence, usually offers better predictive performance than removal strategy. However, it requires a costly computation of the inverse of a matrix whose dimension is either B_l or B_u. To resolve this computational burden, we propose two special projection strategies which are nearest-neighbor projection (NNP) and random-neighbor projection (RNP). At the outset of this section, we define the index sets of the labeled and unlabeled portions at the iteration t as $I_t^l \subset [l]$[2] and $I_t^u \subset [l+1 : l+u]$[3], respectively. Hence, the current model $\mathbf{w}_t$ can be written as

$$\mathbf{w}_t = \sum_{i \in I_t^l} \delta_i \Phi(x_i) + \sum_{i \in I_t^u} \delta_i \Phi(x_i)$$

Both the removal and projection strategies involve the vectors whose coefficients have smallest absolute values in the labeled and unlabeled portions. We now define

$$l_t = \underset{i \in I_t^l}{\text{argmin}} \; |\delta_i| \text{ and } u_t = \underset{i \in I_t^u}{\text{argmin}} \; |\delta_i|$$

3.3.1 Removal

In the removal strategy, we simply remove $\Phi(x_{l_t})$ or $\Phi(x_{u_t})$. This strategy is efficient, but the information of the removed vectors are completely vanished.

3.3.2 Projection

To keep the information of the removed vector, the original projection strategy performs a projection of this vector onto the linear span of the remaining vectors. Although this full projection can efficiently preserve the information of the removed vector, it requires a costly computation of the inverse of B_l (or B_u) by B_l (or B_u) matrix which costs cubically over the budget sizes. Furthermore, decreasing the budget sizes to reduce the computational cost may significantly compromise the learning performance. To speed up the computation and omit the computational dependence of the projection on the budget sizes, we propose two variations of the projection which are nearest-neighbor projec-

[2]We denote $[l] = \{1, 2, \ldots, l\}$.

[3]We denote $[l+1 : l+u] = \{l+1, l+2, \ldots, l+u\}$.

tion (NNP) and random-neighbor projection (RNP) strategies. For brevity in presentation, we denote

$$x_{r_t} = \begin{cases} x_{l_t} & \text{for the option 'l' of BM} \\ x_{u_t} & \text{for the option 'u' of BM} \end{cases}$$

Nearest-Neighbor Projection (NNP). To efficiently preserve the information of x_{r_t}, before removing it, we find k-NN of x_{r_t} and do projection of x_{r_t} onto the linear span of this set. Our intuition is that if the vector x falls into the k-NN of x_{r_t} then x is close to x_{r_t}; consequently the induced dot product $K(x, x_{r_t}) = \Phi(x)^{\mathsf{T}} \Phi(x_{r_t})$ is high and hence, $\Phi(x)$ can largely keep the information of $\Phi(x_{r_t})$.

Random-Neighbor Projection (RNP). To further speed up the NNP strategy, we propose random-neighbor projection (NNP) wherein we first randomly choose k vectors from the support set and then project $\Phi(x_{r_t})$ onto the linear span of these vectors to preserve its information.

3.4 Convergence Analysis

In what follows we present the convergence analysis for BS3VM. Given an instance x, in a BM procedure, we replace this instance by its approximation $A(x)$ which incurs the difference vector $D(x) = \Phi(x) - A(x)$. In particular, with the removal strategy, $A(x) = 0, \forall x$, with the full projection strategy, $A(x) = \mathbb{P}_L(x), \forall x$ where $\mathbb{P}_L(x)$ specifies the linear span of the remaining vectors in the support set, and with NNP or RNP strategy, $A(x) = \mathbb{P}_L(x), \forall x$ where $\mathbb{P}_L(x)$ specifies the linear span of k corresponding vectors. We further define $\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}}\, J(\mathbf{w})$. For simplification, we assume that $\|\Phi(x)\| = K(x,x)^{1/2} = 1, \forall x$.

Let us denote two Bernoulli random variables which indicate whether the budget maintenances for the labeled portion (i.e., the option 'l') and for the unlabeled portion (i.e., the option 'u') are performed by Z_t^l, Z_t^u. The update rule is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t - Z_t^l \delta_{l_t} D(x_{l_t}) - Z_t^u \delta_{u_t} D(x_{u_t}) \qquad (6)$$

It is noteworthy that the update rule in Eq. (6) covers all BM strategies. In addition, our convergence analysis can be applied to all aforementioned BM strategies but for comprehensibility, we present the theoretical results for the removal strategy.

Lemma 1 establishes an upper bound on $\|\mathbf{w}_t\|$, followed by Lemma 2 which establishes an upper bound on $\|g_t\|$.

Lemma 1. *The following statement holds*

$$\|\mathbf{w}_t\| \le C + 2C', \forall t$$

Lemma 2. *The following statement holds*

$$\|g_t\| \le G = 2\left(C + 2C'\right), \forall t$$

In Algorithm 1, the labeled-vertex sampling (cf. line 5) updates the coefficient of one labeled support vector while the edge sampling (cf. line 6) updates two coefficients of two support vectors. To proceed the convergence analysis, we assume that before removed, the coefficient of the labeled vector $\Phi(x_{l_t})$ is updated at most m times via the labeled-vertex sampling and n times via the edge sampling and the coefficient of the unlabeled vector $\Phi(x_{u_t})$ is updated at most p times via the edge sampling. Particularly, in the context of online learning, the labeled vector $\Phi(x_{l_t})$ might be sampled from a continuous distribution and so might be the edge. It follows that $m = n = p = 1$ and the assumption is naturally valid.

Lemma 3. *Given two positive integer numbers m, n, assume that before removed, the coefficient of $\Phi(x_{l_t})$ is updated at most m times via the labeled-vertex sampling and n times via the edge sampling. We then have*

$$|\delta_{l_t}| \le \left(mC + nC'\right)/t, \forall t$$

Lemma 4. *Given a positive integer number p, assume that before removed, the coefficient of $\Phi(x_{u_t})$ is updated at most p times via the edge sampling. We then have*

$$|\delta_{u_t}| \le pC'/t, \forall t$$

Lemma 5 establishes an upper bound on $\|h_t\|$, followed by Lemma 6 establishing an upper bound on $\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2}$.

Lemma 5. *We define $\rho_i = \frac{\delta_i}{\eta_t} = t\delta_i$ and $h_t = Z_t^l \rho_{l_t} D(x_{l_t}) + Z_t^u \rho_{u_t} D(x_{u_t})$. Then we have*

$$\|h_t\| \le H = mC + (n+p)C', \forall t$$

Lemma 6. *The following statement holds*

$$\mathbb{E}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right]^{1/2} \le W = H + \sqrt{H^2 + (G+H)^2}, \forall t$$

We can now state Theorem 7 which establishes an upper bound on the regret. This theorem also reveals that there exists a gap between the rendered and optimal solutions. This gap crucially depends on the budget maintenance rates for the labeled and unlabeled portions.

Theorem 7. *Let us consider the running of Algorithm 1. The following statement holds*

$$\begin{aligned}\mathbb{E}[J(\overline{\mathbf{w}}_t)] - J(\mathbf{w}^*) &\le \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[J(\mathbf{w}_t)] - J(\mathbf{w}^*) \\ &\le \frac{(G+H)^2(\log T + 1)}{2T} \\ &+ \frac{W}{T}\sum_{t=1}^{T} \mathbb{P}\left(Z_t^l = 1\right)\mathbb{E}\left[\rho_{l_t}^2\right]^{1/2} \\ &+ \frac{W}{T}\sum_{t=1}^{T} \mathbb{P}(Z_t^u = 1)\mathbb{E}\left[\rho_{u_t}^2\right]^{1/2}\end{aligned}$$

where $\rho_{l_t} = \delta_{l_t}/\eta_t$ *and* $\rho_{u_t} = \delta_{u_t}/\eta_t$.

Remark 8. The theoretical result gained in Theorem 7 also encompasses the standard analysis. In particular, if the BM procedures never happen (i.e., $\mathbb{P}\left(Z_t^l = 1\right) = \mathbb{P}\left(Z_t^u = 1\right) = 0, \forall t$), we achieve the logarithm regret bound $\frac{(G+H)^2(\log T+1)}{T}$. Furthermore, to minimize the gap, we should choose to remove the vectors with the smallest absolute coefficients (since $\rho_{l_t} = \frac{\delta_{l_t}}{\eta_t}$ and $\rho_{u_t} = \frac{\delta_{u_t}}{\eta_t}$).

4 Experiments

We establish quantitative experiments to investigate the influence of the budget sizes (i.e., B_l and B_u) to the accuracy and training time, and to prove the accuracy and efficiency of our proposed BS3VM on several benchmark datasets. The data statistics is given in Table 1. To simulate the semi-supervised learning context, we randomly remove 80% and 90% data labels in each dataset. We create three versions of our approach: BS3VM with the removal strategy (BS3VM-R), BS3VM with the nearest-neighbor projection strategy (BS3VM-NNP), and BS3VM with the random-neighbor projection strategy (BS3VM-RNP).

Baselines. In order to investigate the efficiency and accuracy of BS3VM, we compare with the following baselines:

- LapSVM [Melacci and Belkin, 2011]: Laplacian Support Vector Machine is a state-of-the-art method in semi-supervised classification based on manifold regularization framework. It can reduce the computational complexity from $\mathrm{O}\left(n^3\right)$ to $\mathrm{O}\left(n^2\right)$ where n is the training size using the preconditioned conjugate gradient and an early stopping strategy.

- CCCP-TSVM [Collobert et al., 2006]: A kernel-based semi-supervised method was proposed to solve the optimization problem using convex-concave procedures.

All codes of the baselines are achieved from the corresponding authors. All compared methods run on a Windows machine with the configuration of 24-vcore CPU Xeon 3.47 GHz and 96GB RAM.

Hyperparameter Setting. The standard RBF kernel, given by $K\left(x,x'\right) = e^{-\gamma\left\|x-x'\right\|^2}$, is used in the experiments. For LapSVM, we use the parameter settings proposed in [Melacci and Belkin, 2011], wherein the parameters γ_A and γ_I are searched in the range $\left\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10, 100\right\}$. In all experiments with LapSVM, we utilize the preconditioned conjugate gradient version, which is more suitable for the LapSVM optimization problem [Melacci and Belkin, 2011]. For CCCP-TSVM, we use the setting CCCP-TSVM$|_{UC*=LC}^{s=0}$. The trade-off parameter C is tuned in the range $\left\{2^{-5}, 2^{-3}, \dots, 2^3, 2^5\right\}$ and the width of kernel δ is varied in the range $\left\{2^{-5}, 2^{-3}, \dots, 2^3, 2^5\right\}$. Regarding our proposed BS3VM, the bandwidth σ of Gaussian kernel weight function is set as to $\frac{1}{2\sigma^2} = \gamma$, and the second trade-off parameter C' is set to be equal the first trade-off parameter C. We employ the standard training-testing split with 90% of data for training and 10% of data for testing. We run 5-fold cross-validation, and then select the parameter set that yields the highest classification accuracy. We set the number of iterations T in BS3VM to $\lceil 0.01 \times (l+u) \rceil$ for the large-scale datasets such as MUSHROOMS, W5A, W8A, COD-RNA, and COVTYPE, and to $\lceil 0.1 \times (l+u) \rceil$ for the remaining datasets. Each experiment is carried out five times to compute the average of the reported measures.

Dataset	**Dimension**	**Size**
G50C	50	551
COIL20	1,014	145
USPST	256	601
AUSTRALIAN	14	690
A1A	123	1,605
MUSHROOM	112	8,124
SVMGUIDE3	21	1,243
SVMGUIDE2	20	391
W5A	300	9,888
W8A	300	49,749
COR-RNA	8	59,535
COVTYPE	54	100,945

Table 1: The statistics of the experimental datasets.

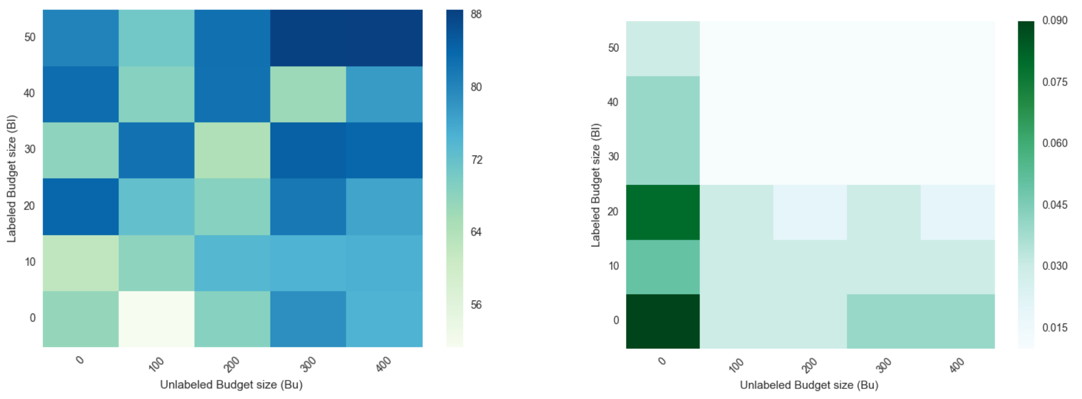
Experimental Results. The experimental results are reported in Tables 2 and 3. For readability, we emphasize in boldface the highest accuracy and in italics the shortest training time. Regarding the classification accuracy, our proposed BS3VMs are comparable with other baselines and CCCP-TSVM is slightly better than others. However, our BS3VMs scale impressively with the large-scale datasets whilst CCCP-TSVM scale unsatisfactorily. The version BS3VM-R wins the shortest training time over all experimental datasets, except for the dataset W5A under 80%-unlabeled setting. Besides, two other versions BS3VM-NNP and BS3VM-RNP also scale efficiently with the training size, and their training times only slightly exceed those of BS3VM-R on all datasets. This implies that the simplified projection strategies (i.e., NNP and RNP) do not incur a significant computational burden. Interestingly, BS3VM-R always offers comparable accuracies comparing with BS3VM-NNP and BS3VM-RNP, which indicates that the information loss occurring in the removal of vector in BS3VM-R is tolerant. It is noteworthy that although we only set small budgets for all datasets (i.e., 50 or 100), the classification accuracies attained by three versions on all datasets are still remarkable. This fact confirms the effectiveness of our proposed budget maintenance strategies in eliminating the redundant vectors and in keeping the core vectors which sufficiently characterize the training set.

Datasets [*B*]	**BS3VM-R**		**BS3VM-NNP**		**BS3VM-RNP**		**LapSVM**		**CCCP**	
	Acc	**Time**	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**
G50C [50]	95.45	*0.131*	95.45	0.14	95.45	0.133	96.2	0.29	**98.18**	0.141
COIL20 [50]	**100**	*0.083*	**100**	0.094	**100**	0.088	**100**	0.39	98.1	1.078
USPST [50]	99.17	*0.091*	99.17	0.217	99.17	0.134	99.2	0.28	**99.58**	0.61
AUSTRALIAN [50]	**88.41**	0.041	87.68	0.098	86.96	0.085	85.9	0.94	81.88	*0.002*
A1A [50]	**80.1**	*0.146*	78.19	0.192	79.44	0.167	**80.1**	0.21	79.75	0.953
MUSHROOM [50]	96.12	*0.506*	96.86	0.874	97.05	0.566	98.8	5.25	**100**	28.078
SVMGUIDE3 [50]	78.23	*0.056*	77.02	0.289	78.23	0.116	75.8	0.33	**81.45**	1.421
SVMGUIDE2 [50]	76.12	*0.02*	79.1	0.03	86.57	0.024	85.1	0.41	**90.27**	0.078
W5A [50]	96.46	1.732	96.97	2.471	97.27	2.348	97	*1.18*	**98.33**	146.28
W8A [100]	97.02	*18*	97.02	18.15	96.8	18.1	**97.4**	26.15	97.1	1,380.16
COR-RNA [100]	85.53	*0.545*	85.95	0.937	86.53	0.836	85.7	13.14	**88.47**	3,900.43
COVTYPE [100]	**87.07**	*8.273*	84.56	8.931	84.12	8.51	81.8	19.75	85.91	5,958.07

Table 2: Cross-validation accuracies (in %) and training times (in second) on the experimental datasets when 80% of data labels are removed. We set the same value for B_l and B_u which is the notation [B] next to the dataset name.

Datasets [*B*]	**BS3VM-R**		**BS3VM-NNP**		**BS3VM-RNP**		**LapSVM**		**CCCP**	
	Acc	**Time**	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**
G50C [50]	**95.45**	*0.129*	94.55	0.128	**95.45**	0.131	94.5	0.29	94.55	0.509
COIL20 [50]	92.86	*0.011*	96.43	0.025	**100**	0.015	**100**	0.16	**100**	0.366
USPST [50]	**100**	*0.012*	**100**	0.032	**100**	0.017	99.6	0.38	**100**	2.17
AUSTRALIAN [50]	86.23	*0.004*	85.51	0.013	85.51	0.009	86.2	0.32	**89.85**	0.031
A1A [50]	82.24	*0.018*	81.26	0.035	81.62	0.024	81.6	0.24	**82.37**	0.047
MUSHROOM [50]	91.38	*0.09*	91.02	0.141	94.29	0.105	97.5	0.334	**99.96**	8.82
SVMGUIDE3 [50]	77.82	*0.005*	77.42	0.017	77.02	0.005	77.9	0.28	**83.37**	0.054
SVMGUIDE2 [50]	82.09	*0.004*	79.1	0.007	79.1	0.005	80.6	0.38	**85.12**	0.02
W5A [50]	97.17	*0.329*	97.27	0.412	91.17	0.323	**97.5**	0.521	97.39	7.41
W8A [100]	97.03	*3.215*	96.93	3.982	97.01	3.79	**97.32**	9.15	97.18	379.06
COR-RNA [100]	83.33	*0.519*	82.73	0.82	**92.92**	0.682	86.1	11.42	89.74	326.72
COVTYPE [100]	80.98	*4.628*	86.89	5.142	**86.38**	4.897	80.2	34.02	85.75	1,275.22

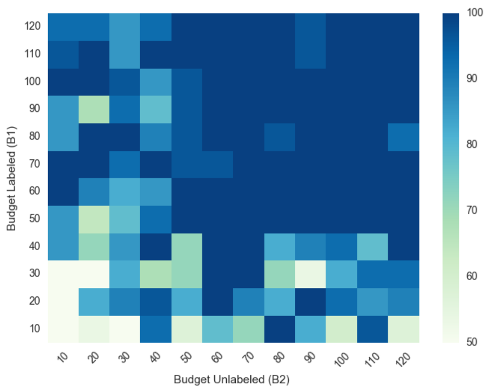
Table 3: Cross-validation accuracies (in %) and training times (in second) on the experimental datasets when 90% of data labels are removed. We set the same value for B_l and B_u which is the notation [B] next to the dataset name.



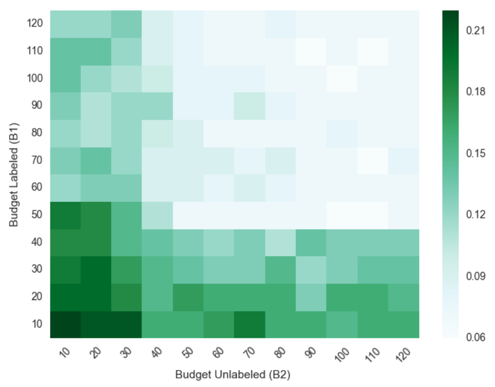
(a) The variation of the classification accuracy.

(b) The variation of the training time.

Figure 1: The variations of the classification accuracy and training time on the dataset AUSTRALIAN when two budget sizes B_l and B_u are varied.

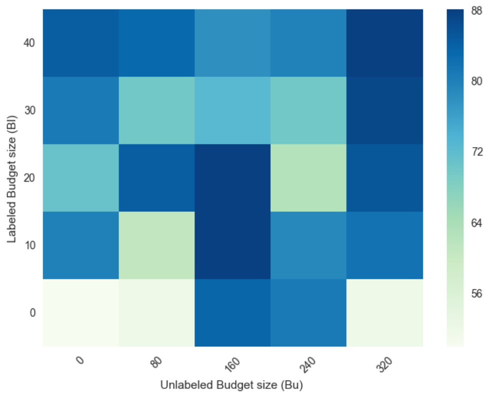


(a) The variation of the classification accuracy.

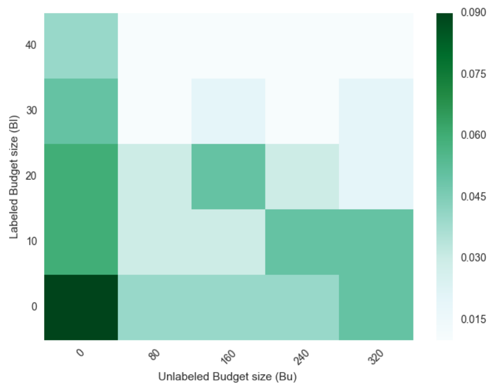


(b) The variation of the training time.

Figure 2: The variations of the classification accuracy and training time on the dataset COIL20 when two budget sizes B_l and B_u are varied.

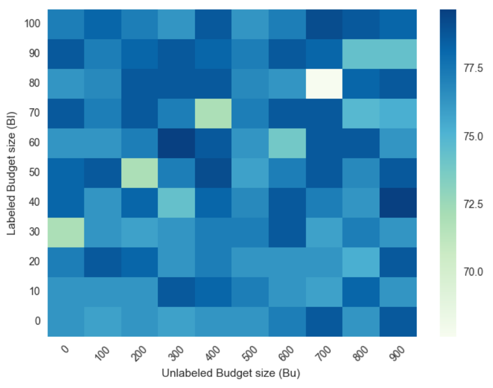


(a) The variation of the classification accuracy.

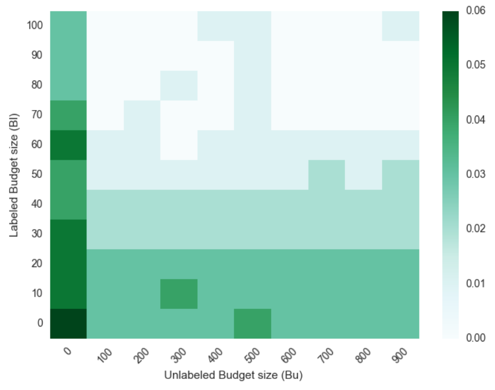


(b) The variation of the training time.

Figure 3: The variations of the classification accuracy and training time on the dataset G50C when two budget sizes B_l and B_u are varied.



(a) The variation of the classification accuracy.



(b) The variation of the training time.

Figure 4: The variations of the classification accuracy and training time on the dataset SVMGUIDE3 when two budget sizes B_l and B_u are varied.

Influence of Budget Sizes to Learning Performance. We investigate the influence of the budget sizes (i.e., B_l and B_u) to the learning performance. We choose to conduct a simulation study on 4 datasets AUSTRALIAN, COIL20, G50C, and SVMGUIDE3. For each dataset, we simultaneously vary the labeled budget size B_l and the unlabeled budget size B_u to measure the classification accuracy and the training time. We visualize the classification accuracies and the training times using heat maps shown in Figures 1, 2, 3, and 4. It can be observed that when increasing the budget sizes (i.e., B_l and B_u), the classification accuracy tends to increase and the training time tends to decrease or fluctuate. The reason is that large budget sizes enrich the expressiveness of the model, and hence boost the accuracy. In the meanwhile, when increasing the budget sizes, there appears a trade-off between the computational cost in each iteration and the budget maintenance rate which fluctuates the training time depending on which factor dominates. In practice, using these heat maps, one can conveniently find the optimal pair (B_l, B_u) that balances the classification accuracy and the training time for example $(50, 300)$ for AUSTRALIAN, $(50, 50)$ for COIL20, $(40, 320)$ for G50C, and $(60, 300)$ for SVMGUIDE3. Another observation is that the increases of B_l and B_u fairly equally affect the classification accuracy while increasing B_u strongly affects to the training time than increasing B_l. Finally, the training time becomes worst if we set one budget size to a small value and gradually increase another.

5 Conclusion

In this paper, we have proposed Budgeted Semi-supervised Support Vector Machine (BS3VM) for semi-supervised learning purpose. We first leverage the theory of kernel method with the framework of spectral-graph-based semi-supervised learning to form a specific optimization problem, which involves the core optimization problem of kernel method for learning on labeled data and simultaneously allows the label propagation. We then apply the SGD method to directly solve such optimization problem in the primal form. To resolve the curse of kernelization, we employ two budgets for the labeled and unlabeled portions in the model. We further establish a rigorous convergence analysis for BS3VM. The theoretical results reveal that there exists a gap between the rendered and optimal solutions. This gap crucially depends on the budget maintenance rates and the coefficients accompanied with the removed vectors. Finally, we conduct extensive experiments on several benchmark datasets. The experimental results show that BS3VM yields a comparable classification accuracy while simultaneously achieving a significant computational speed-up comparing with the state-of-the-art baselines.

References

A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th International Conference on Machine Learning*, ICML 2007, pages 49–56, 2007.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7: 2399–2434, December 2006.

A. Blum, J. D. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML*, volume 69, 2004.

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

O. Chapelle and A. Zien. Semi-supervised classification by low density separation, 2005.

O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, June 2008.

R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims. Large scale transductive svms. *Journal of Machine Learning Research*, 2006.

C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

P. Duong, V. Nguyen, M. Dinh, T. Le, D. Tran, and W. Ma. Graph-based semi-supervised support vector data description for novelty detection. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2015.

E. Hazan and S. Kale. Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization. *J. Mach. Learn. Res.*, 15(1):2489–2512, January 2014. ISSN 1532-4435.

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)*, pages 200–209, Bled, Slowenien, 1999.

S. Kakade and Shalev-Shwartz. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, 2008.

N. Kasabov, D. Zhang, and P.S. Pang. Incremental learning in autonomous systems: evolving connectionist systems for on-line image and speech recognition. In *Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on*, pages 120 – 125, june 2005.

S. Lacoste-Julien, M. W. Schmidt, and F. Bach. A simpler approach to obtaining an o(1/t) convergence rate for the projected stochastic subgradient method. *CoRR*, 2012.

T. Le, D. Tran, T. Tran, K. Nguyen, and W. Ma. Fuzzy entropy semi-supervised support vector data description. In *2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5, Aug 2013.

T. Le, V. Nguyen, D. T. Nguyen, and D. Phung. Nonparametric budgeted stochastic gradient descent. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 654–662, 2016.

S. Melacci and M. Belkin. Laplacian support vector machines trained in the primal. *J. Mach. Learn. Res.*, 12: 1149–1184, jul 2011.

V. Nguyen, T. Le, T. Pham, M. Dinh, and T. H. Le. Kernel-based semi-supervised learning for novelty detection. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 4129–4136, July 2014.

A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML 2012*, pages 449–456, 2012.

S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. In *The Hebrew University*, 2007.

V. Sindhwani, S.S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the 23rd international conference on Machine learning*, ICML 2006, pages 841–848, 2006.

A. J. Smola and I. R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2003.

I. W. Tsang and J. T. Kwok. Large-scale sparsified manifold regularization. pages 1401–1408. MIT Press, 2006.

Z. Wang and S. Vucetic. Online passive-aggressive algorithms on a budget. In *AISTATS*, volume 9, pages 908–915, 2010.

Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *Journal of Machine Learning Research*, 13(1):3103–3131, 2012.

X. Zhu, J. S. Kandola, Z. Ghahramani, and J. D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *NIPS 2004*, pages 1641–1648, 2004.