Nonparametric Budgeted Stochastic Gradient Descent

Trung Le **Vu Nguyen** **Tu Dinh Nguyen** **Dinh Phung**
Pattern Recognition and Data Analytics, Deakin University, Australia

Abstract

One of the most challenging problems in kernel online learning is to bound the model size. Budgeted kernel online learning addresses this issue by bounding the model size to a predefined budget. However, determining an appropriate value for such predefined budget is arduous. In this paper, we propose the Nonparametric Budgeted Stochastic Gradient Descent that allows the model size to automatically grow with data in a principled way. We provide theoretical analysis to show that our framework is guaranteed to converge for a large collection of loss functions (e.g. Hinge, Logistic, L2, L1, and ε-insensitive) which enables the proposed algorithm to perform both classification and regression tasks without hurting the ideal convergence rate $\mathrm{O}\left(\frac{1}{T}\right)$ of the standard Stochastic Gradient Descent. We validate our algorithm on the real-world datasets to consolidate the theoretical claims.

1 Introduction

In machine learning, online learning represents a family of efficient and scalable learning algorithms for building a predictive model incrementally from a sequence of data examples [19]. Different from the conventional learning algorithms which usually require a costly procedure to retrain the entire dataset when a new instance arrives [11, 2], online learning algorithms aim at adapting the model using new incoming instances without hurting the predictive performance, making them more suitable for large-scale online applications wherein data usually arrive sequentially and evolve rapidly.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

The seminal line of work in online learning, referred to as *linear online learning* [19, 3, 7], aims to learn a linear predictor in the input space. The key limitation of this approach lies in its over-simplified assumption of linearity, hence fail to deal with nonlinear data. This motivates the work of *kernel-based online learning* [8, 12] in which a linear model in the feature space corresponding to a nonlinear model in the input space is capable of modeling complex and nonlinear functions.

Recently, Stochastic gradient descent (SGD) method [18] has gained increasing attention in developing scalable online learning methods which can efficiently handle large-scale dataset. It is currently regarded as one of the simplest and most effective first-order method to solve convex optimization problems. Given a convex loss function and a training set of N examples, SGD can be used to obtain a sequence of T predictors, whose average has a generalization error which converges (with T) to the optimal one in the class of predictors we are considering [17]. A typical convergence rate of SGD is known to be $\mathrm{O}\left(\frac{\log T}{T}\right)$ [9, 20]; and several works have recently attained a better convergence rate at $\mathrm{O}\left(\frac{1}{T}\right)$ for strongly convex case [10, 13, 17]. However, the kernelization of a standard SGD setting is vulnerable to the *curse of kernelization* which may cause a linear growth in the model size accumulated over time and hence increases training time significantly [21].

To resolve this issue, Budgeted Stochastic Gradient Descent (BSGD) has been proposed in [21] to bound the model size using a predefined and fixed budget B. Particularly, when the current model size exceeds this budget, a budget maintenance strategy (e.g., removal, projection, or merging) is triggered to recover the model size back to the budget B. The convergence analysis presented in [21] reveals that there is an error gap between the approximate and optimal solutions. Furthermore, determining a suitable value for the predefined budget in a principled way is important, but challenging, since setting a small budget makes the learning faster but may suffer underfitting phenomenon, whereas a large budget makes the

model fitter to data but may dramatically slow down the training process.

In this paper, we propose Nonparametric Budgeted Stochastic Gradient Descent (NBSGD) to address the problem of model selection and adaptation in BSGD wherein the model size can automatically grow with the data. We provide convergence analysis of our proposed method. Our analysis covers widely used loss functions in classification (Hinge and Logistic) and regression (L2, L1, and ε-insensitive). We also would like to point out that the original BSGD proposed in [21] only covers Hinge loss case. Moreover, we provide an insightful analysis of the error gap which enables us to quantify the influences of the budget maintenance rate and gradient error to this gap. In light of this result, we introduce a Bernoulli random variable to control the budget maintenance rate. The resulting NBSGD simultaneously achieves two major aims: (1) adjusting the model size in a principled way and (2) eradicating the error gap to recover the ideal convergence rate $\mathrm{O}\left(\frac{1}{T}\right)$ for the proposed algorithm.

Another issue of budget online learning is that although the number of support vectors is bounded for each intermediate classifier f_t, it is usually not the case for the final solution, which is computed as the average of the intermediate classifiers [24]. We overcome this obstacle by theoretically showing that if we output the last decision boundary, with a high confidence level we still approximate the optimal solution with a good convergence rate (cf. Thm. 6 and 12). We are aware that the last decision boundary is used in the previous work of budgeted SGD [21] without any theoretical guarantee of convergence to the optimal solution.

2 Related Work

In this section, we review existing literature on budgeted online kernel learning that are closely related to our work. The most popular approach is to bound the model size using one of three following budget maintenance strategies:

Support vector removal. Forgetron [6] is the first budgeted online learning method that employs removal strategy for budget maintenance. At each iteration, if the classifier makes a mistake, it conducts a three-step update: first, running the standard Perceptron [19] update; second, shrinking the coefficients of support vectors with a scaling factor; and lastly, removing the support vector which has the smallest coefficient. Another method is Randomized Budget Perceptron (RBP) [1] that randomly removes a support vector when the model size exceeds the budget. Besides, several methods including Budget Perceptron [4], Budgeted Passive Aggressive (BPA-S) algorithm [23], and Budgeted Stochastic Gradient Descent with removal strategy [21] attempt to discard the most redundant support vector (SV).

Projection. The work in this category first projects the incoming instance onto the linear span of support vectors in the feature space to compute the corresponding distance. If this distance exceeds a predefined threshold, the new instance is then added to the support set, otherwise the coefficients of support vectors are incremented by the projection coefficients. Typical examples are Projectron [16], Budgeted Passive Aggressive Nearest Neighbor (BPA-NN) [23], and Budgeted Stochastic Gradient Descent with projection strategy [21] .

Support vector merging. The goal of this strategy is to maintain the budget by merging two existing support vectors into a new one. Typical methods include Twin Support Vector Machine (TVM) algorithm [22] and Budgeted Stochastic Gradient Descent with merging strategy [21] .

3 Problem Settings

We consider two following optimization problems for batch (Eq. (1)) and online (Eq. (2)) settings:

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{(x,y)\sim\mathbb{P}_N}\left[l(\mathbf{w};x,y)\right] \triangleq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{w};x_i,y_i) \quad (1)$$

$$\min_{\mathbf{w}} f(\mathbf{w}) \triangleq \frac{\lambda}{2}\|\mathbf{w}\|^2 + \mathbb{E}_{(x,y)\sim\mathbb{P}_{X,Y}}\left[l(\mathbf{w};x,y)\right] \quad (2)$$

where $l(\mathbf{w};x,y)$ is a convex loss function, $\mathbb{P}_{X,Y}$ is the joint distribution of (x,y) over $\mathcal{X}\times\mathcal{Y}$, and $\mathbb{P}_N$ specifies the uniformly empirical distribution over the training set $\mathcal{D} = \{(x_1,y_1),\ldots,(x_N,y_N)\}$. Furthermore, we assume that the convex loss function $l(\mathbf{w};x,y)$ satisfies the following property, that is, there exists two positive numbers A, B such that $\left\|l'(\mathbf{w};x,y)\right\| \leq A\|\mathbf{w}\|^{1/2} + B$, $\forall \mathbf{w}, x, y$. As demonstrated in Sec. 1 of supplementary material accompanied with this paper, this condition is valid for all typical loss functions. It is evident that given any $\mathbf{w}$, there exists a random variable g such that $\mathbb{E}[g|\mathbf{w}] = f'(\mathbf{w})$. In particular, we can specify $g = \lambda\mathbf{w} + l'(\mathbf{w};x_t,y_t)$ where $(x_t,y_t) \sim \mathbb{P}_{X,Y}$ or $\mathbb{P}_N$. Let us further define $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$.

We recap the standard stochastic gradient descent (SGD) in Alg. 1. In Alg. 1, we use the standard learning rate $\eta_t = \frac{1}{\lambda t}$ and α_t is a scalar such that $l'(\mathbf{w}_t;x_t,y_t) = \alpha_t\Phi(x_t)$ (this scalar exists for all typical loss functions) where $\Phi(.)$ is the transformation from the input space to the feature space. In the case

Algorithm 1 Stochastic Gradient Descent algorithm.

Input: λ, $K(.,.)$

1: $\mathbf{w}_1 = \mathbf{0}$
2: **for** $t = 1, 2, \ldots T$ **do**
3: Receive (x_t, y_t) //$(x_t, y_t) \sim \mathbb{P}_{X,Y}$ *or* $\mathbb{P}_N$
4: $g_t = \lambda \mathbf{w}_t + l^{'}(\mathbf{w}_t; x_t, y_t)$
5: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t g_t = \frac{t-1}{t}\mathbf{w}_t - \eta_t \alpha_t \Phi(x_t)$
6: **end for**

Output: $\overline{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$ or $\mathbf{w}_{T+1}$

of kernelization, this standard SGD algorithm is vulnerable to the curse of kernelization, that is, in online setting or batch setting with large-scale dataset, the size of support set or the model size, almost linearly grows with data accumulated over time. Consequently, the computation gradually becomes slower or even infeasible when the data size grows rapidly.

4 Budgeted SGD

To break the curse of kernelization for SGD, [21] proposes to perform a budget maintenance procedure (e.g., removal, projection, or merging), whenever the model size exceeds the predefined budget B. The analysis in [21] further discloses that there exists an error gap between the approximate solution of SGD with budget maintenance and that of standard SGD. However, this analysis only focuses on the Hinge loss case and hence restricts itself for only classification task.

In this paper, we propose a new theoretical analysis for all typical loss functions (i.e., Hinge, Logistic, L2, L1, ε-insensitive), and hence handles both classification and regression tasks. In addition, our analysis offers a deeper insightful view which enables us to quantify the influences of the frequency of budget maintenance and the gradient error to the gap. The proposed algorithm is presented in Alg. 2.

Let Z_t be a binary random variable which indicates if budget maintenance is performed. By its definition, $Z_t = 1$ only when one support vector is added and the number of support vectors b exceeds the budget B, i.e., $\left\|l^{'}(\mathbf{w}_t; x_t, y_t)\right\| > 0$ and $b > B$. The update rule is

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \eta_t g_t - Z_t \Delta_t \\ &= \frac{t-1}{t}\mathbf{w}_t - \eta_t l^{'}(\mathbf{w}_t; x_t, y_t) - Z_t \Delta_t\end{aligned}$$

The instance $(x_{t'}, y_{t'})$ arriving in at the time t' will possess the coefficient at the time t as

$$\frac{-l^{'}(\mathbf{w}_{t'}; x_{t'}, y_{t'})}{\lambda t'} \times \frac{t'}{t'+1} \times \ldots \times \frac{t-1}{t} = \frac{-l^{'}(\mathbf{w}_{t'}; x_{t'}, y_{t'})}{\lambda t}$$

Algorithm 2 Budgeted SGD algorithm.

Input: $B > 0$, λ, $K(.)$

1: $\mathbf{w}_1 = \mathbf{0}$
2: **for** $t = 1, 2, \ldots T$ **do**
3: Receive (x_t, y_t) //$(x_t, y_t) \sim \mathbb{P}_{X,Y}$ *or* $\mathbb{P}_N$
4: $\mathbf{w}_{t+1} = \frac{t-1}{t}\mathbf{w}_t$
5: **if** $\left(\left\|l^{'}(\mathbf{w}_t; x_t, y_t)\right\| > 0\right)$ **then**
6: $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \eta_t l^{'}(\mathbf{w}_t; x_t, y_t)$
7: $b = b + 1$
8: **if** $(b > B)$ **then**
9: $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \Delta_t$ //*maintenance*
10: $b = b - 1$
11: **end if**
12: **if** *L2 is used* and $\lambda \leq 1$ **then**
13: $\mathbf{w}_{t+1} = \prod_{\mathcal{B}(\mathbf{0}, y_{\max}\lambda^{-1/2})}(\mathbf{w}_{t+1})$
14: **end if**
15: **end if**
16: **end for**

Output: $\overline{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$ or $\mathbf{w}_{T+1}$

Therefore, we have $\Delta_t = \frac{-l^{'}(\mathbf{w}_{t'}; x_{t'}, y_{t'})}{\lambda t}$. The update rule is rewritten as follows

$$\mathbf{w}_{t+1} = \frac{t-1}{t}\mathbf{w}_t - \frac{l^{'}(\mathbf{w}_t; x_t, y_t)}{\lambda t} + Z_t \frac{l^{'}(\mathbf{w}_{t'}; x_{t'}, y_{t'})}{\lambda t}$$

It is noteworthy that to ensure $\|\mathbf{w}_t\|$ is bounded, in Alg. 2 and 3 with L2 loss, if $\lambda \leq 1$ then we project $\mathbf{w}_t$ onto the hypersphere with centre origin and radius $y_{\max}\lambda^{-1/2}$ (i.e., $\mathcal{B}\left(\mathbf{0}, y_{\max}\lambda^{-1/2}\right)$), since it can be shown that the optimal solution $\mathbf{w}^*$ lies in $\mathcal{B}\left(\mathbf{0}, y_{\max}\lambda^{-1/2}\right)$ in this case (cf. Thm. 1). In addition, with L2 loss and $\lambda > 1$, we do not need to perform a projection to bound $\|\mathbf{w}_t\|$, since according to Thm. 2, $\|\mathbf{w}_t\|$ is bounded by $\frac{y_{\max}}{\lambda - 1}$. We further define $y_{\max} = \max_y |y|$.

Theorem 1. *If*

$$\mathbf{w}^* = \underset{\mathbf{w}}{argmin}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^{\mathsf{T}}\Phi(x_i)\right)^2\right)$$

then $\|\mathbf{w}^*\| \leq y_{max}\lambda^{-1/2}$.

Theorem 2. *If* $\lambda > 1$ *then* $\|\mathbf{w}_{T+1}\| \leq \frac{y_{max}}{\lambda - 1}\left(1 - \frac{1}{\lambda^T}\right) < \frac{y_{max}}{\lambda - 1}$ *for all* T.

The following theorem quantifies the influences of the budget maintenance rate and the gradient error to the gap between the BSGD and standard SGD.

Theorem 3. *In Alg. 2, let define the gradient error as* $M_t = \frac{\Delta_t}{\eta_t} = -l^{'}(\mathbf{w}_{t'}; x_{t'}, y_{t'})$. *We have the following*

inequality where Q, W are two positive constants.

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{Q\left(\log T + 1\right)}{2\lambda T} + \frac{1}{T} W^{1/2} \sum_{t=1}^{T} \mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2} \mathbb{P}\left(Z_t = 1\right)^{1/2} \quad (3)$$

$$\leq \frac{Q\left(\log T + 1\right)}{2\lambda T} + \frac{1}{T} W^{1/2} \sum_{t=1}^{T} \mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2} \quad (4)$$

Remark 4. The inequality in Eq. (3) shows the influences of the budget maintenance rate (i.e., $\mathbb{P}\left(Z_t = 1\right)$) and the gradient error $\mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2}$ to the error gap. It is evident that to demote this gap, we need to reduce either the probability $\mathbb{P}\left(Z_t = 1\right)$ or the gradient error $\mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2} = \mathbb{E}\left[\alpha_t^2\right]^{1/2} \left\|\Phi\left(x_t\right)\right\|$. This observation supports us in designing the budget maintenance strategies.

Remark 5. Our analysis also encompasses the standard analysis. Precisely, if $\mathbb{P}\left(Z_t = 1\right) = 0$ (i.e., the budget maintenance is never performed), the inequality in Eq. (4) becomes $\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{Q(\log T+1)}{2\lambda T}$.

Theorem 6. *We denote the gap*

$$d_T = \frac{1}{T} W^{1/2} \sum_{t=1}^{T} \mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2} \mathbb{P}\left(Z_t = 1\right)^{1/2}$$

Let r be an integer picked uniformly at random from $\{1, 2, \ldots, T\}$. Then, with probability of at least $1 - \delta$ we have

$$f\left(\mathbf{w}_r\right) \leq f\left(\mathbf{w}^*\right) + d_T + \frac{Q\left(\log T + 1\right)}{2\lambda\delta T}$$

Remark 7. Thm. 6 reveals that when outputting any $\mathbf{w}_r$ with a high confidence level, we can approximate the optimal solution with the convergence rate $\mathrm{O}\left(\frac{\log T}{T}\right)$.

5 Nonparametric Budgeted SGD

Motivated from Thm. 3, another way to narrow the gap is to control the rate of budget maintenance. In what follows, we prove that if the rate of budget maintenance downgrades proportional to $\frac{1}{t}$ (i.e., $\mathbb{P}\left(Z_t = 1\right) \sim \mathrm{O}\left(\frac{1}{t}\right)$), the gap vanishes. Moreover, using the γ-suffix averaging [17] we can obtain the convergence rate $\mathrm{O}\left(\frac{1}{T}\right)$ (cf. Thm. 11). We start this section with an important corollary.

Corollary 8. *If $\mathbb{E}\left[Z_t^2\right] = \mathbb{P}\left(Z_t = 1\right) \sim O\left(\frac{1}{t}\right)$ then $\mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right] \sim O\left(\frac{1}{t}\right)$, i.e., there exists $D > 0$ such that $\mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right] < \frac{D}{t}$ for all t.*

In Alg. 3, to control the budget maintenance rate, we employ a Bernoulli binary random variable Z_t where $Z_t = 1$ indicates that budget maintenance is performed and specifically establish $\mathbb{P}\left(Z_t = 1\right) = \min\left(\frac{\beta}{t}, 1\right)$.

Algorithm 3 Nonparametric BSGD algorithm.

Input: B, β, γ, λ, $K\left(.,.\right)$

$\mathbf{w}_1 = \mathbf{0}$
for $t = 1, 2, \ldots T$ **do**
  Receive $\left(x_t, y_t\right)$ //$\left(x_t, y_t\right) \sim \mathbb{P}_{X,Y}$ *or* $\mathbb{P}_N$
  $\mathbf{w}_{t+1} = \frac{t-1}{t}\mathbf{w}_t$
  if $\left(\left\|l'\left(\mathbf{w}_t; x_t, y_t\right)\right\| > 0\right)$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \eta_t l'\left(\mathbf{w}_t; x_t, y_t\right)$
    $b = b + 1$
    if $\left(b > B\right)$ **then**
      Sample $Z_t \sim \mathrm{Ber}\left[\min\left(\frac{\beta}{t}, 1\right)\right]$
      if $\left(Z_t = 1\right)$ **then**
        $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} - \Delta_t$
        $b = b - 1$
      end if
    end if
    if *L2 is used* and $\lambda \leq 1$ **then**
      $\mathbf{w}_{t+1} = \prod_{\mathcal{B}\left(\mathbf{0}, y_{\max}\lambda^{-1/2}\right)}\left(\mathbf{w}_{t+1}\right)$
    end if
  end if
end for

Output: $\overline{\mathbf{w}}_T^{\gamma} = \frac{\mathbf{w}_{(1-\gamma)T+1} + \ldots + \mathbf{w}_T}{\gamma T}$ or $\mathbf{w}_{T+1}$

Remark 9. In Alg. 3, β is used to govern the budget maintenance rate. When β rises up, the budget maintenance is performed more frequently and the sparsity is encouraged.

Theorem 10. *In Alg. 3, let define the gradient error as $M_t = \frac{\Delta_t}{\eta_t} = -l'\left(\mathbf{w}_{t'}; x_{t'}, y_{t'}\right)$. We have the following inequality*

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^{\gamma}\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{D\lambda^2 + Q\log\left(1/\left(1-\gamma\right)\right)}{2\gamma T} + \frac{\beta D^{1/2}}{\gamma T} \sum_{t=(1-\gamma)T+1}^{T} \frac{\mathbb{E}\left[\left\|M_t\right\|^2\right]^{1/2}}{t^{3/2}}$$

Theorem 11. *Consider the running of Alg. 3 , we have the following inequality*

$$\mathbb{E}\left[f\left(\overline{\mathbf{w}}_T^{\gamma}\right) - f\left(\mathbf{w}^*\right)\right] \leq \frac{D\lambda^2 + Q\log\left(1/\left(1-\gamma\right)\right)}{2\gamma T} + \frac{2\beta L D^{1/2}\log\left(1/\left(1-\gamma\right)\right)}{2\gamma T}$$

Theorem 12. *Let r be an integer randomly picked from $\{(1-\gamma)T + 1, \ldots, T\}$. Then, with probability at least $1 - \delta$ we have*

$$f\left(\mathbf{w}_r\right) \leq f\left(\mathbf{w}^*\right) + \frac{R}{2\gamma\delta T}$$

where we have defined

$$R = D\lambda^2 + Q\log\left(1/\left(1-\gamma\right)\right) + 2\beta L D^{1/2}\log\left(1/\left(1-\gamma\right)\right)$$

Remark 13. Thm. 11 shows the convergence rate $\mathrm{O}\left(\frac{1}{T}\right)$ if we employ γ-suffix average output. However, as a consequence, model size is increased. Thm. 12 points out that if we take the last hyperplane, with a high confidence level, we can approach the optimal solution with convergence rate $\mathrm{O}\left(\frac{1}{T}\right)$. In fact, the model size is not bounded. However, we will demonstrate in the experiment that tuning β efficiently inspires the sparsity and NBSGD always gains much better sparsity than the standard SGD.

6 Budget Maintenance Strategy

Driven by Thm. 3 and 10, to speed up the reduction of the error gap, we should concentrate on the support vector x_p in the support set such that

$$p = \underset{j\in\mathcal{I}_t}{\operatorname{argmin}} \ \|M_j\| = \underset{j\in\mathcal{I}_t}{\operatorname{argmin}} \ \left\|\alpha^{(j)}\right\|^2 \|\Phi\left(x_j\right)\|^2 \quad (5)$$

where $\mathcal{I}_t$ specifies the support indices and without loss of generality, we assume that $\mathcal{I}_t = \{1, 2, \ldots, t\}$.

As in [21], we use two budget maintenance strategies which are removal and projection. Regarding the projection strategy, different from [21], we examine two different cases: 1) $|\mathcal{I}_t| \leq B$ where the matrix inversion is incrementally updated as in [5, 16] and 2) $|\mathcal{I}_t| = B + 1$ to maintain the inversion of the Gram matrix $K_t = \left[K\left(x_i, x_j\right)\right]_{i,j\in\mathcal{I}_t}$. We note that the projection strategy proposed in [21] did not cover the second case and hence, cannot result in the exact computation for the inverse matrix. Moreover, to speed up the projection strategy, we propose to use a heuristic procedure as follows. Before removing x_p, we project $\Phi\left(x_p\right)$ onto the k most-correlated vectors in $\{\Phi\left(x_1\right), \ldots, \Phi\left(x_t\right)\}$ and preserve information of $\Phi\left(x_p\right)$ using these k principle components. The advantage is twofold. First, the k most-correlated vectors turn out to be the k nearest neighbors of x_p in the input space. Second, to find the projection, we only need to compute the inversion of an $k \times k$ matrix, where k is usually a small number (e.g., $k = 4$). Our experiment shows that this heuristic procedure is very efficient. It offers comparable accuracy with the exact projection while achieving a significant speed-up.

6.1 Removal strategy

We choose to remove x_p according to Eq. (5). It is noteworthy that if a radial kernel is used then $\|\Phi\left(x_t\right)\|^2 = K\left(x_t, x_t\right) = 1$ and only $\left\|\alpha^{(j)}\right\|^2$ is involved in selecting x_p.

6.2 Projection strategy

We also choose to remove x_p according to Eq. (5). Before removing x_p, we project $\Phi\left(x_p\right)$ onto the $\mathrm{span}\left(\{\Phi\left(x_i\right) : i \in \mathcal{I}_t \,\mathrm{and}\, i \neq p\}\right)$ and then substitute $\Phi\left(x_p\right)$ by this projection. Let us further denote the projection by $\mathcal{P}_t x_p = \sum_{i\in\mathcal{I}_t\setminus\{p\}} d_i x_i$. We can find the projection by $d = K_t^{-1}\boldsymbol{k}_p$ where $K_t = \left[K\left(x_i, x_j\right)\right]_{i,j\in\mathcal{I}_t\setminus\{p\}}$ and $\boldsymbol{k}_p = \left[K\left(x_p, x_i\right)\right]_{i\in\mathcal{I}_t\setminus\{p\}}^{\mathsf{T}}$.

6.2.1 Exact projection

$|\mathcal{I}_t| \leq B$: we use the incremental formula as in [5, 16] for maintaining K_t^{-1}. We note that the budget maintenance is not carried out in this stage.

$|\mathcal{I}_t| = B + 1$: for updating K_t^{-1} from K_{t-1}^{-1}, we observe that the two matrices K_{t-1} and K_t are distinct in one row and one column. To transform K_{t-1} to K_t, we can substitute the column and row $\boldsymbol{k}_p$ by $\boldsymbol{k}_t$. Therefore, we can formulate $K_t = K_{t-1} + L$ where L is a sparse matrix of all zeros except for the p-th column and row which are L_p and L_p^{T} respectively where $L_p = \boldsymbol{k}_t - \boldsymbol{k}_p$. It is apparent that $rank\left(L\right) = 2$. To update K_t^{-1} from K_{t-1}, we rely on Thm. 14 (cf. [15]). The derivation is provided in the supplementary material.

Theorem 14. *Let A and $A+B$ be non-singular matrices, and let B have rank $r > 0$. Let $B = B_1 + \cdots + B_r$, where each B_i has rank 1, and each $C_{k+1} = A + B_1 + \cdots + B_k$ is non-singular. Setting $C_1 = A$, then $C_{k+1}^{-1} = C_k^{-1} - g_k C_k^{-1} B_k C_k^{-1}$ where $g_k = \frac{1}{1+trace(C_k^{-1}B_k)}$. In particular, $(A+B)^{-1} = C_r^{-1} - g_r C_r^{-1} B_r C_r^{-1}$.*

6.2.2 Approximate projection

Although we can incrementally update the inverse matrix, the matrix inversion still costs $\mathrm{O}\left(B^3\right)$ which circumvents the application of the projection strategy when the budget B is large. To address this issue, we propose to use the heuristic procedure that instead of projecting $\Phi\left(x_p\right)$ onto $\mathrm{span}\left(\{\Phi\left(x_i\right) : i \in \mathcal{I}_t \,\mathrm{and}\, i \neq p\}\right)$, we project it onto the linear span of the images of k most-correlated vectors which turns out to be the k nearest neighbors of x_p in the input space. Our intuition behind this heuristic is that if a vector x_i lies too far x_p, their similarity $K\left(x_p, x_i\right)$ in the feature space is small. Consequently, they are nearly orthogonal in the feature space, and thereby the projection of $\Phi\left(x_p\right)$ onto $\Phi\left(x_i\right)$ (in the feature space) is negligible.

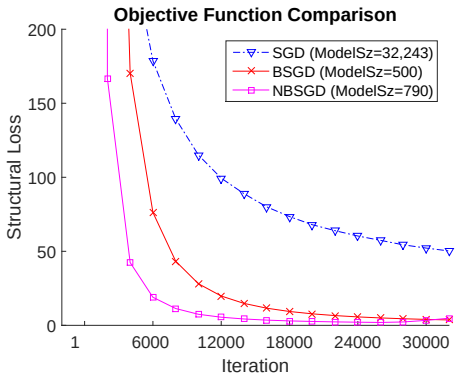

Figure 1: Objective function comparison.

7 Experiment

In this experiment, we demonstrate the efficiency of the proposed method in automatically identifying the suitable model size for the kernel online learning task. We conduct the experiments with five loss functions: Hinge and Logistic for classification and L2, L1 and ε-insensitive for regression.

As almost all algorithms are random, we run each experiment 5 times and then average the performance. We carry out the experiments using Matlab on the Windows machine equipped with Core i7 2.60GHz CPU, 24GB RAM. All models are trained a single pass for fair comparisons.

We use 10 datasets downloaded from LIBSVM site[1] to perform a number of tasks including binary, multiclass classification and regression.

Baselines

- Kernel Stochastic Gradient Descend [20].
- Budgeted Stochastic Gradient Descend [21].
- Random Perceptron (RBP) [1]: Budgeted Perceptron algorithm with random support vector removal strategy.
- Forgetron [6]: Forgetron maintains the budget size by discarding the oldest support vectors.
- Projectron++ [16]: The aggressive version of Projectron algorithm that updates with both margin error and mistake case.
- Budgeted Passive Aggressive (BPAs) [23] and Budgeted Online Gradient Descent (BOGD) [25]: The budgeted versions of the original PA and OGD with simple support removal strategy.

[1]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

The RBP, Forgetron, Projectron++ and BPAs are obtained from LSOKL library [14].

Hyperparameters setting Throughout the experiment, we utilize RBF kernel, $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \times ||\boldsymbol{x} - \boldsymbol{x}'||^2\right)$. We use cross-validation with 5 folds for selecting the best parameters γ and λ whose ranges are in $\left\{2^{-8}, 2^{-4}, 1, 2^4, 2^{10}\right\}$. In all experiments, we heuristically set $\beta = 0.6 \times \#train$ which can be again selected using cross-validation.

7.1 Analysis of NBSGD learning behavior

To gain deeper understanding into model behavior, we analyze NBSGD on the objective function, the impact of β and the learning with different maintenance schemes.

Objective function We compare the objective function (defined in Eq. (1)) of NBSGD with that of the standard SGD and the Budgeted SGD in Fig. 1. As our NBSGD can discover the appropriate model size in such a way that we minimize the empirical loss $\mathbb{E}_{(x,y)\sim\mathbb{P}_N}\left[l\left(\mathbf{w}; x, y\right)\right]$ and control the regularization quantity $\frac{\lambda}{2}||\mathbf{w}||^2$, we secure the best objective function. Budgeted SGD's objection function converges more slowly than NBSGD, but more rapidly than SGD. SGD produces a larger model size or heavier structural loss and converges more slowly than the budgeted counterparts. The reason is that NBSGD results in sparser model than SGD while its empirical loss is comparable.

Impact of β on model size We investigate the effect of β on model size in Fig. 2 (right). We vary β from 1 to $\#train$ of 70,000, then record the model size on SenseIT Vehicle dataset. We observe that the larger β yields smaller model size and decreases classification accuracy. Reversely, smaller β yields larger model size and increases classification accuracy. Using this analysis, we can learn the trade-off between the model size and accuracy through β. In addition, we can use the plot on the right Fig. 2 for model selection, e.g., choosing β from 40,000 to 50,000 (given the total 70,000 data points) would result in the optimal performance which has a relatively good accuracy and a low model size.

Learning with different maintenance schemes To maintain a low model size, NBSGD performs the budget maintenance procedure when a support vector is added into the model, the current model size exceeds the budget, and the Bernoulli variable Z_t is sampled with 1. We note that the budget maintenance strategies described in Sec. 6 and in [21] may have different effects on speed and accuracy. We examine the

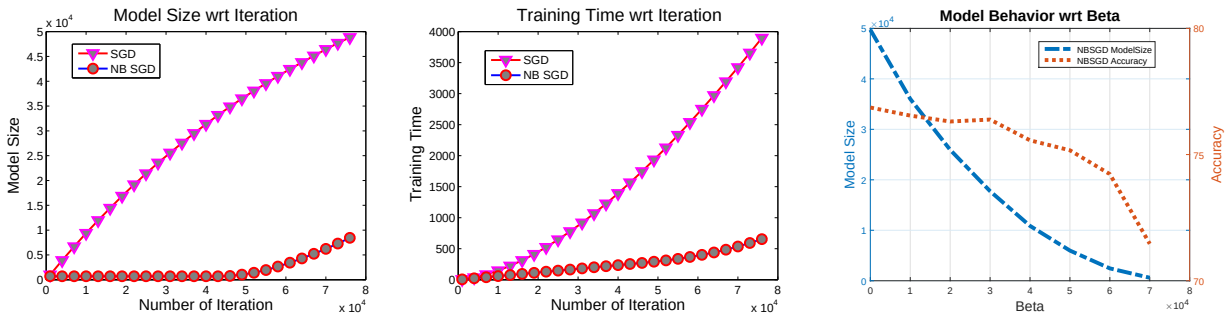

Figure 2: Model size, training time behavior w.r.t. number of iterations and parameter β.

performance using both two Removal and Projection strategies with Hinge and Logistic in Table 1.

As Removal is pretty simpler and requires less operation than Projection, Removal is much faster although we have sped up the Projection using approximate projection described in Sec. 6.2.2. For learning quality with NBSGD, both maintenance approaches are comparable and neither outperforms the other. Particularly, Removal with Logistic loss holds the best performance in all three datasets (Gisette, Skin and DNA) while Projection obtains the best in two out of three datasets.

7.2 NBSGD for Kernelized Budgeted Online Regression

SGD is vulnerable to the curse of kernelization which may cause unbounded linear growth in the model size and the training time as a consequence. Therefore, the proposed NBSGD is a promising candidature for the kernel online regression problem wherein the model size is much sparser.

We conduct experiments on the kernel budgeted online regression task where the budget size is data-drivenly identified. As it is hard to specify the suitable number of budget size in advance, we design two settings for BSGD as Under Budgeted SGD with a smaller budget size than NBSGD and Over Budgeted SGD with a larger budget size than NBSGD. Then, we show that our NBSGD is automatically adapted to the growth of the data and produces better performances than the fixed budget counterparts [21].

We present the results in Table 2. SGD outperforms the budgeted counterparts in terms of regression quality with the lowest RMSE score since SGD utilizes almost all training data points as the support vectors for learning (2,795 in Space GA and 18,575 in Cadata). In contrast, our NBSGD and BSGD with fixed budgets run faster and have smaller model size while we also secure the comparable RMSE scores.

The performance may depend on the loss function and the individual dataset. In Space GA and Cadata, ε-insensitive and L1 attain the best performance with the lowest RMSE score.

7.3 Classification Comparison with Budgeted Online Competitors

To demonstrate the advantage of NBSGD in the budgeted online learning paradigm, we further compare our model with the other baselines in Table 3. NBSGD consistently gains much sparser model, thus run significantly faster while still outperforming SGD in accuracy (except A9A). This observation is consistent with our experiment of objective function comparison in Fig. 1. For fixed budget algorithms including RBP, Forgetron, BPAs, BOGD, although we use Over Budgeted setting, NBSGD outperforms them on all experimental datasets. Comparing with Projectron++, a method that can adapt its model size, NBSGD surpasses it in accuracy, sparsity, and training time on A9A and IJCNN. For DNA, NBSGD achieves comparable accuracy and a little higher model size. Moreover, NBSGD's training time is lower due to the high computation demand of projection in Projectron++.

8 Conclusion

In this paper, we have proposed Nonparametric Budgeted Stochastic Gradient Descent which can automatically discover the model size that fits data in a principled way. We have analyzed the convergence property of NBSGD for common loss functions. In addition, the proposed model does not hurt the ideal convergence rate $O\left(\frac{1}{T}\right)$. The experiment shows that our method can gain much sparser model while simultaneously achieving the comparable accuracy.

Dataset [#train][#feat]			Gisette [6,000][5,000]			Skin [220,551][3]			DNA [1,400][180]		
	Scheme	Loss	Acc	Time	ModSz	Acc	Time	ModSz	Acc	Time	ModSz
SGD	None	Hinge	**0.92**	748	3,096	**1.00**	4,062	139,445	**82.4**	1.74	1,399
		Logistic	0.89	906	3,793	**1.00**	3,865	139,434	82.4	1.76	1,399
BSGD	Removal	Hinge[21]	0.86	30.55	100	**1.00**	3,378	30,000	81.0	1.56	1,000
		Logistic	0.86	30.59	100	**1.00**	3,357	30,000	85.6	1.56	1,000
	Projection	Hinge[21]	0.86	35.59	100	0.95	4,145	30,000	81.0	1.75	1,000
		Logistic	0.86	35.62	100	**1.00**	4,145	30,000	85.6	1.75	1,000
NBSGD	Removal	Hinge	**0.92**	101.7	774.6	0.99	2,494	4,984	77.2	0.60	267.4
		Logistic	**0.92**	98.7	758.8	**1.00**	2,541	5,008	**88.3**	0.62	272.0
	Projection	Hinge	**0.92**	103.9	765.6	0.99	2,688	4,963	76.1	1.17	275.4
		Logistic	**0.92**	105.1	757.8	**1.00**	2,686	4,977	87.7	1.19	273.2

Table 1: Classification performance with different maintenance schemes of Removal and Projection. The number of training points and feature size are indicated in the brackers. Blue and magenta colors indicate the best performance for Logistic and Hinge losses, respectively.

Data	Exam [#train 3,653][#test 406][#feature 8]											
Method	SGD [*M=3,653*]			UBSGD [*M=200*]			OBSGD [*M=2,000*]			NBSGD [*M=1,444*]		
Loss	L2	L1	Eps	L2	L1	Eps	L2	L1	Eps	L2	L1	Eps
RMSE	**0.90**	**0.90**	**0.90**	0.98	1.04	1.04	0.90	0.95	0.95	**0.90**	0.96	0.96
Time	1.23	1.20	1.21	0.84	0.82	0.92	1.14	1.14	1.15	0.98	0.96	0.96
Data	**Space GA** [#train 3,759][#test 418][#feature 8]											
Method	**SGD** [*M=2,795*]			**UBSGD** [*M=200*]			**OBSGD** [*M=2,000*]			**NBSGD** [*M=1,249*]		
Loss	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**
RMSE	**0.70**	**0.61**	**0.61**	0.84	0.80	0.81	0.71	0.68	0.70	0.73	0.68	0.69
Time	0.98	0.92	0.98	0.74	0.66	0.61	0.93	0.81	0.98	0.73	0.75	0.66
Data	**Cadata** [#train 18,576][#test 2,064][#feature 8]											
Method	**SGD** [*M=18,576*]			**UBSGD** [*M=500*]			**OBSGD** [*M=10,000*]			**NBSGD** [*M=1,027*]		
Loss	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**	**L2**	**L1**	**Eps**
RMSE	**0.95**	**0.96**	**0.96**	0.98	0.98	0.98	0.99	0.99	0.99	0.98	0.98	0.98
Time	24.3	25.1	23.5	4.80	4.99	4.49	326	328	328	5.04	4.79	5.33

Table 2: Regression performance comparison. Due to the property of L2, L1 and ε-insensitive losses in regression task, SGD uses all of data points as a support vectors. The BSGD is specified in two settings as Under Budgeted SGD (smaller budget size than NBSGD) and Over Budgeted SGD (larger budget size than NBSGD). The model size is emphasized in the bracket as [M]. Magenta, blue and red colors indicate the best performance for L2, L1 and ε -insensitive losses, respectively.

Dataset	A9A			DNA			IJCNN		
Data Size	[#train 32,561][#feat 123]			[#train 2,000][#feat 180]			[#train 35,000][#feat 22]		
Methods	Acc	Time	ModSz	Acc	Time	ModSz	Acc	Time	ModSz
RBP	78.9(1)	484(2.5)	8,000	67.7(18)	1.29	300	94.9(1)	**25.1**	4,000
Forgetron	78.9(1)	485(1.5)	8,000	87.6(2)	1.25	300	94.8(1)	27.1	4,000
Projectron++	81.7(1)	7,363(99)	7,125	**88.4(4)**	1.62	245.6	95.7	676	2,218
BPAs	82(2)	713(3.5)	8,000	85.8(2)	11.84	300	96.1	28.2	4,000
BOGD	76.4	705(5)	8,000	76.3(7)	1.79	300	90.5	27.7	4,000
SGD	**84.1**	1,076	20,579	82.2	1.74	1,399	*96.4*	894	34,999
NBSGD	*83.6*	**60.4**	3,559	*88.1(2)*	**0.62**	272	**97.3**	121	759

Table 3: Classification with baselines budgeted methods. SGD and NBSGD use Hinge loss and Removal scheme. The standard deviation, where is significant, is in the parenthesis ().

References

[1] 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.

[2] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[3] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, 2006.

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

[5] L. Csato. Sparse online gaussian processes. *Neural Computation*, 2002.

[6] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems*, pages 259–266, 2005.

[7] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 264–271, New York, NY, USA, 2008. ACM.

[8] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, December 1999.

[9] E. Hazan, A. Agarwal, and S Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[10] 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.

[11] T. Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[12] J. Kivinen, A. J. Smola, and R. C. Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52:2165–2176, August 2004.

[13] 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.

[14] J. Lu, S. C.H. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu. Large scale online kernel learning. *J. Mach. Learn. Res.*, 2015.

[15] K. S. Miller. On the Inverse of the Sum of Matrices. *Mathematics Magazine*, 54(2):67–72, 1981.

[16] F. Orabona, J. Keshet, and B. Caputo. Bounded kernel-based online learning. *J. Mach. Learn. Res.*, 10:2643–2666, December 2009.

[17] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *International Conference on Machine Learning (ICML-12)*, pages 449–456, 2012.

[18] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[19] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[20] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated subgradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[21] Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *J. Mach. Learn. Res.*, 13(1):3103–3131, 2012.

[22] Z. Wang and S. Vucetic. Twin vector machines for online learning on a budget. In *Proceedings of the SIAM International Conference on Data Mining*, pages 906–917, 2009.

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

[24] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He. Online kernel learning with a near optimal sparsity bound. In *International Conference on Machine Learning*, volume 28, pages 621–629, 2013.

[25] P. Zhao, J. Wang, P. Wu, R. Jin, and S. C. H. Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. *CoRR*, 2012.