PSPS: Preconditioned Stochastic Polyak Step-size method for badly scaled data

Farshed Abdukhakimov[1] FARSHED888@GMAIL.COM
Chulu Xiang[1] CHULU.XIANG@MBZUAI.AC.AE
Dmitry Kamzolov[1] KAMZOLOV.OPT@GMAIL.COM
Robert Gower[2] GOWERROBERT@GMAIL.COM
Martin Takáč[1] TAKAC.MT@GMAIL.COM
[1]*Mohamed bin Zayed University of Artificial Intelligence* [2]*Flatiron Institute*

Abstract

The family of Stochastic Gradient Methods with Polyak Step-size offers an update rule that alleviates the need of fine-tuning the learning rate of an optimizer. Recent work [5] has been proposed to introduce a *slack* variable, which makes these methods applicable outside of the interpolation regime. In this paper, we combine preconditioning and *slack* in an updated optimization algorithm to show its performance on badly scaled and/or ill-conditioned datasets. We use *Hutchinson's* method to obtain an estimate of a Hessian which is used as the preconditioner.

1. Introduction

In this paper we consider a finite-sum optimization problem

$$w^* \in \arg\min_{w\in\mathbb{R}^d}\{f(w) := \frac{1}{n}\sum_{i=1}^n f_i(w)\}, \tag{1}$$

where $w \in \mathbb{R}^d$ is the weight parameter and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a smooth and twice differentiable objective function. Problems of this structure constitute one of the core parts of machine learning where it is known as Empirical Risk Minimization. The loss function $f_i(w)$ computes the difference between the prediction of a model with weights parameters w and a target value. The objective is to then minimize the average loss $f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)$ over a given dataset with n elements.

Stochastic Gradient Descent. Since this minimization problem can be particularly complicated to solve due to complexity of the dataset or the type of loss function, it attracts substantial research every year. One of the most basic optimization methods to solve this problem is Stochastic Gradient Descent (*SGD*) [2, 11] which despite its simplicity remains one of the most often used algorithms for this purpose. SGD updates the weight parameter w in a loop as following:

$$w_{t+1} = w_t - \gamma\nabla f_i(w_t), \tag{2}$$

where $\nabla f_i(w_t)$ is the gradient of a loss function on a mini-batch $\{i\}$ of a dataset at point w_t, γ is the step-size (or learning rate) of the update and t is the iteration counter. Using mini-batches of a large dataset to compute the gradient significantly helps to reduce the the time needed for convergence to an optimal point w^*. However, every loss function and dataset combination requires special tuning of a step-size γ to find a minimum, which turns γ into a hyperparameter. This issue was one of the motivations behind methods with adaptive learning rate, where γ is replaced by an expression which does not need a fine-tuning process. One of the recent representatives of this family of methods is Stochastic Gradient Descent with a Polyak Step-size (*SPS*) [5, 8–10].

1.1. Notation and Assumptions.

For a positive definite matrix $B \in \mathbb{R}^{d\times d}$, we can endow the primal space $w \in \mathbf{E} \subseteq \mathbb{R}^d$ and $g \in \mathbf{E}^* \subseteq \mathbb{R}^d$ by the conjugate Euclidean norms: $\|w\|_B = \langle Bw, w\rangle^{1/2}$ and $\|g\|_{B^{-1}} = \langle g, B^{-1}g\rangle^{1/2}$, where $\nabla f(w) \in \mathbf{E}^*$ and $\nabla^2 f(w) \in \mathbf{E}^*$. The operator $\odot$ is defined as a component-wise product between two vectors, also known as the Hadamard product. We use $\mathrm{diag}(v)$ as a diagonal matrix of a given vector v and a vector $\mathrm{diagonal}(\mathbf{H}) \in \mathbb{R}^d$ as the diagonal of a matrix $\mathbf{H} \in \mathbb{R}^{d\times d}$.

Assumption 1.1

We assume that the *interpolation* condition holds for a set of functions $\{f_i(w)\}_{i=1}^n$ with a non-negative loss functions, $f_i(w) \geq 0$, when

$$\exists w^* \in \mathbb{R}^d \quad \text{s.t.} \quad f_i(w^*) = 0 \quad \forall i \in \{1, 2, \ldots, n\}. \tag{3}$$

1.2. Related Work

If we assume that the interpolation condition holds, then we can solve (1) by sampling $i \in \{1, \ldots, n\}$ i.i.d at each iteration t and then solving the nonlinear equation

$$w_{t+1} = \arg\min_{w\in\mathbb{R}^d} \|w - w^t\|^2 \quad \text{s.t.} \quad f_i(w) = 0. \tag{4}$$

While the above projection might have a closed form solution for some simple loss functions, for most nonlinear models like Deep Neural Networks (DNNs) there is no closed-form solution of (4). So instead of solving (4) exactly, we can linearize the $f_i(w)$ around the current iterate w^t to obtain

$$w_{t+1} = \arg\min_{w\in\mathbb{R}^d} \|w - w^t\|^2 \quad \text{s.t.} \quad f_i(w^t) + \langle \nabla f_i(w^t), w - w^t\rangle = 0. \tag{5}$$

- **SGD with Polyak Step-size (SPS).** Vanilla SPS is a method with an adaptive learning rate defined as $\gamma_t = \frac{f_i(w_k)}{\|\nabla f_i(w_k)\|^2}$. It requires only the stochastic gradient and the function at the current iterate. Conveniently enough this update serves as an exact closed-form solution for (5).
- **Adding Slack.** Outside of the interpolation regime there might not exist a solution for (4). So instead of trying to set all losses functions to zero, we can instead try to make them all small by minimizing a *slack* variable as follows

$$\min_{w\in\mathbb{R}^d, s\geq 0} \quad s \quad \text{s.t.} \quad f_i(w) \leq s, \quad \text{for} \quad i = 1, \ldots, n, \tag{6}$$

$$\min_{w\in\mathbb{R}^d, s\geq 0} \quad s^2 \quad \text{s.t.} \quad f_i(w) \leq s, \quad \text{for} \quad i = 1, \ldots, n, \tag{7}$$

  which are referred to as *L1* and *L2* slack minimization [5], respectively. One can note that the goal of this method is to force s to be as small as possible which allows to solve problems where the interpolation assumption does not hold or the model is under-parameterized.
- **SPS with Slack.** In a recent work [5] a new variant of SPS was proposed which samples a single inequality in (6) or (7), linearizes that inequality, and solves the following problem iteratively

$$w_{t+1}, s_{t+1} = \arg\min_{w\in\mathbb{R}^d, s\geq 0} \tfrac{1}{2}\|w - w_t\|^2 + \tfrac{1}{2}(s - s_t)^2 + \lambda s,$$
$$\text{s.t.} \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t\rangle \leq s, \tag{8}$$

  for (6), or

$$w_{t+1}, \quad s_{t+1} = \arg\min_{w\in\mathbb{R}^d, s\in\mathbb{R}} \|w - w_t\|^2 + (s - s_t)^2 + \lambda s^2,$$
$$\text{s.t.} \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t\rangle \leq s, \tag{9}$$

  for (7), which are referred to as *SPSL1* and *SPSL2*, respectively. The slack parameter $\lambda > 0$ controls the trade-off between finding a small slack variable s, and not moving too far from the previous iterate. The closed-form solutions to (8) and (9) are given in [5].

- **Preconditioning.** Data can be badly scaled and/or ill-conditioned and preconditioning is one way to improve the convergence speed of algorithms. Algorithms that take advantage of preconditioning have a generic update rule as following,
$$w_{t+1} = w_t - \gamma_t \hat{D}_t^{-1} \nabla f_i(w_t), \tag{10}$$
where $\hat{D}_t \in \mathbb{R}^{d\times d}$ is an invertible positive definite matrix. A textbook example of a method that utilizes this technique is Newton's method where $\hat{D}_t = \nabla^2 F(w_t)$ and $\gamma_t = 1$. More recent and practical methods include AdaHessian, Adagrad and OASIS [4, 7, 12]. These methods incorporate curvature of the loss function via adaptive estimates of the Hessian. They use *Hutchinson's* method (described later in detail) to obtain an estimate of the Hessian diagonal.

1.3. Contributions

Here we combine preconditioning and variants of slack regularized SPS methods. We then demonstrate that these new preconditioned methods perform well on badly scaled and ill-conditioned data.

- **Updated SPS.** We extend the SPS methods and present 3 updated algorithms *PSPS*, *PSPSL1* and *PSPSL2* which use *Hutchinson's* method to precondition search directions and include the scaling of Polyak step-size with a weighted Euclidean norm. Closed-form updates to our methods are described later.
- **PyTorch Implementation.** We develop practical variants of our methods as *PyTorch* optimizers and make the code publicly available at our GitHub repository[1].
- **Empirical Results.** Several experiments are conducted in 2 different settings to compare our results to SGD, Adam and to variants of SPS that are not applying any preconditioning techniques. We demonstrate the proposed algorithms exhibit noticeable improvements on badly scaled data.

2. Diagonal Preconditioning

Hutchinson's method. Hutchinson's method [6] is used to estimate the diagonal of the Hessian matrix. To compute this estimate, the Hutchinson method uses only a few Hessian-vector products, which in turn can be computed efficiently using backpropagation [3]. Indeed, the product of a Hessian matrix $\mathbf{H} = \nabla^2 f(w)$ and a vector z can be computed through a directional derivative of the gradient since $\frac{d}{dt} \nabla f(w + tz)|_{t=0} = \mathbf{H}z$. The Hutchinson's method uses a Hessian-vector products to estimate the diagonal via $\text{diagonal}(\mathbf{H}) = \mathbb{E}[z \odot (\mathbf{H}z)]$, where z is a random vector with Rademacher distribution[2] or a normal distribution, see [1] or Lemma 4 in the appendix. Using this identity, we estimate the Hessian diagonal from a given D_0 by sampling a vector z at each iteration, and iteratively updatie our estimate using a weighted average as follows,
$$D_t = \beta D_{t-1} + (1-\beta)\,\text{diag}(z \odot \mathbf{H}z), \tag{11}$$
where $\beta \in (0,1)$ is a momentum parameter and $D_0 = \frac{1}{m}\sum_{i=1}^m \text{diag}(z_i \odot \mathbf{H}_i z_i)$, where $\mathbf{H}_i$ denotes a Hessian at the initial point w_0 of a randomly sampled batch. Finally, to ensure that D_t remains Positive Definite, despite possible non-convexity of the loss functions, we use truncation and keep only absolute values of elements as follows $(\hat{D}_t)_{i,i} = \max\{\alpha, |D_t|_{i,i}\}$.

1. https://github.com/fxrshed/ScaledSPS.
2. $z_i \in \{-1, +1\}$ with equal probability.

3. Preconditioning SPS methods

Here we present our new preconditioned SPS methods. To develop each of these new methods, we change the norm in the projection to a weighted norm based on the preconditioning matrix $H \succ 0$. For instance the PSPS (Preconditioned SPS) is given in the next lemma.

Lemma 1 (PSPS) *Let $B_t \succ 0$ for any $t \geq 0$. Then the iterative update of the following problem*

$$w_{t+1} = \arg\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w - w_t\|^2_{B_t} \quad s.t. \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t \rangle = 0$$

is given by

$$w_{t+1} = w_t - \frac{f_i(w_t)}{||\nabla f_i(w_t)||^2_{B_t^{-1}}} B_t^{-1} \nabla f_i(w_t). \tag{12}$$

We can use the same trick for introducing a preconditioner into the slack based methods.

Lemma 2 (PSPS with L1 Slack - (PSPSL1)) *Let $B_t \succ 0$ for any $t \geq 0$ and $\mu, \lambda > 0$. Then the closed-form update for the following problem*

$$w_{t+1}, \quad s_{t+1} = \arg\min_{w \in \mathbb{R}^d, s \geq 0} \frac{1}{2}\|w - w_t\|^2_{B_t} + \mu(s - s_t)^2 + \lambda s$$
$$s.t. \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t \rangle \leq s, \tag{13}$$

is given by $\gamma_t^{L1} = \frac{(f_i(w_t) - s_t + \frac{\lambda}{2\mu})_+}{\frac{1}{2\mu} + \|\nabla f_i(w_t)\|^2_{B_t^{-1}}}$, $\gamma_t = \min\{\gamma_t^{L1}, \frac{f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}}\}$, $w_{t+1} = w_t - \gamma_t B_t^{-1} \nabla f_i(w_t)$, $s_{t+1} = (s_t - \frac{1}{2\mu}(\lambda + \gamma_t^{L1}))_+$. *Here slack parameter λ forces s to be closer to 0 while μ does not allow s_{t+1} to be far from s_t.*

Lemma 3 (PSPS with L2 Slack - (PSPSL2)) *Let $B_t \succ 0$ for any $t \geq 0$ and $\mu, \lambda > 0$. Then the closed form update for the following problem*

$$w_{t+1}, s_{t+1} = \arg\min_{w \in \mathbb{R}^d, s \in \mathbb{R}} \|w - w_t\|^2_{B_t} + \mu(s - s_t)^2 + \lambda s^2$$
$$s.t. \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t \rangle \leq s, \tag{14}$$

is given by $w_{t+1} = w_t - \frac{(f_i(w_t) - \mu\hat{\lambda} s_t)_+}{\hat{\lambda} + \|\nabla f_i(w_t)\|^2_{B_t^{-1}}} B_t^{-1} \nabla f_i(w_t)$, $s_{t+1} = \hat{\lambda}(\mu s_t + \frac{(f_i(w_t) - \mu\hat{\lambda} s_t)_+}{\hat{\lambda} + \|\nabla f_i(w_t)\|^2_{B_t^{-1}}})$, *where* $\hat{\lambda} = \frac{1}{\mu + \lambda}$.

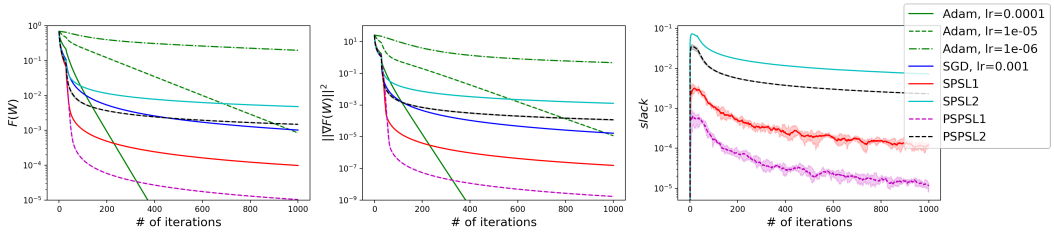
4. Numerical Experiments

By combining methods from Section 3 with Hutchinson's preconditioning, such that $B_t = \hat{D}_t$, we get our new methods *PSPS*, *PSPSL1* and *PSPSL2*. In this section, we present our experiments on binary classification problem with *logistic regression* and *non-linear least squares* loss functions. We selected these settings to demonstrate the performance of our methods on both convex and non-convex environments. We compare our results to original *SPS*, *SPSL1*, *SPSL2*, *SGD*, and *Adam*. All experiments were run with 5 different seeds using *PyTorch 1.11.0*.

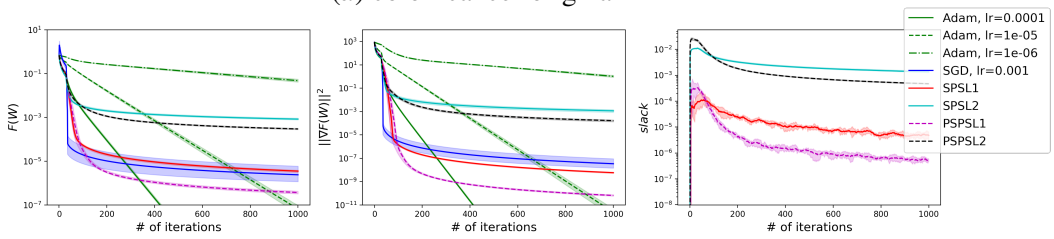
Loss Functions. Let $\{(x_i, y_i)\}_{i=1}^n$ be our dataset, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. *Logistic regression* is defined as $f_{LogReg}(w) = \frac{1}{n}\sum_{i=1}^n \log(1 + \exp(-y_i x_i^T w))$ while *non-linear least squares* is given by $f_{NLLSQ}(w) = \frac{1}{n}\sum_{i=1}^n (y_i - 1/(1 + \exp(-x_i^T w)))^2$, where $y_i \in \{0, 1\}$.

Binary Classification on LibSVM Datasets. We test our methods on 2 binary classification datasets from LibSVM[3], namely **mushrooms** and **colon-cancer**. To simulate badly scaled data we introduce *scaled* version of each datasets where its columns are multiplied by a vector $e = \{\exp(x_i)\}_{i=1}^d$ where x_i is generated from a uniform distribution on the interval $[-k, k]$. For comparison, we also train *SGD* and *Adam* with a constant step-size.

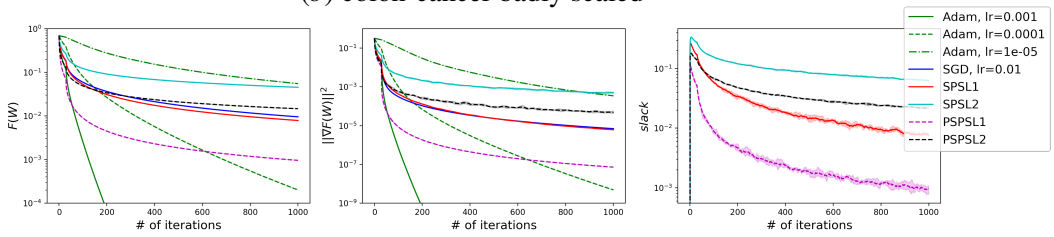
3. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/



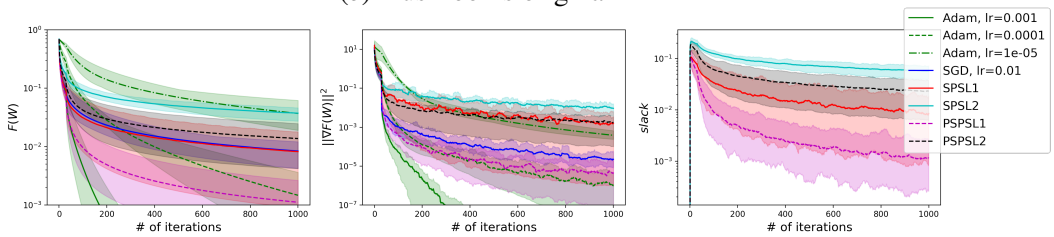
(*a*) colon-cancer original



(*b*) colon-cancer badly scaled



(*c*) mushrooms original



(*d*) mushrooms badly scaled

Figure 1: Performance comparison of PSPSL1 and PSPSL2 to non-scaled versions of SPS, SGD and Adam on original and badly scaled datasets. All methods are trained on Logistic Regression. In badly scaled cases the scaling factor $k = 3$.

5. Conclusion and Future work

In this paper we studied the effect of preconditioning on the family of SPS(Stochastic Gradient Descent with Polyak Step-size) methods. We showed modified update rules *PSPS, PSPSL1, PSPSL2* in Section 3. In our solution a new parameter μ is introduced which helps to control the step direction of slack s. Experiments were conducted in both convex and non-convex settings with 3 different datasets.

Future work. This paper lacks theoretical analysis of our proposed methods which can be done as a follow up research work. On top of that, it is highly encouraged to extend experiments to a realm of Deep Neural Networks.

References

[1] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11):1214–1229, 2007. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2007.01.003. URL https://www.sciencedirect.com/science/article/pii/S0168927407000244. Numerical Algorithms, Parallelism and Applications (2).

[2] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[3] Bruce Christianson. Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12(2):135–150, 1992.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[5] Robert M Gower, Mathieu Blondel, Nidham Gazagnadou, and Fabian Pedregosa. Cutting some slack for sgd with adaptive polyak stepsizes. *arXiv preprint arXiv:2202.12328*, 2022.

[6] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[7] Majid Jahani, Sergey Rusakov, Zheng Shi, Peter Richtárik, Michael W Mahoney, and Martin Takáč. Doubly adaptive scaled algorithm for machine learning using second-order information. *arXiv preprint arXiv:2109.05198*, 2021.

[8] Shuang Li, William J Swartworth, Martin Takáč, Deanna Needell, and Robert M Gower. Using quadratic equations for overparametrized models. 2022.

[9] Shuang Li, William J Swartworth, Martin Takáč, Deanna Needell, and Robert M Gower. SP2: a second order stochastic polyak method. *arXiv preprint arXiv:2207.08171*, 2022.

[10] Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1306–1314. PMLR, 2021.

[11] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[12] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10665–10673, 2021.

Appendix A. Derivation of solutions

Lemma 4 *Let $\mathbf{I} \in \mathbb{R}^{d\times d}$ be the identity matrix. Let $\mathbf{H} \in \mathbb{R}^{d\times d}$ and let $z \in \mathbb{R}^d$ be a random vector with a distribution such that*

$$\mathbb{E}[zz^\top] = \mathbf{I}. \tag{15}$$

It follows that

$$\text{diagonal}(\mathbf{H}) = \mathbb{E}\left[z \circ \mathbf{H}z\right]. \tag{16}$$

Furthermore if z has Rademacher or a normal distribution, then (15) holds.

Proof Taking expectation and expanding the Hadamard product we have that

$$\mathbb{E}\left[z \odot (\mathbf{H}z)\right] = \mathbb{E}\left[\sum_i z_i (\sum_j \mathbf{H}_{ij} z_j) e_i\right] = \sum_i \sum_j \mathbf{H}_{ij}\mathbb{E}[z_j z_i] e_i \tag{17}$$

Since $\mathbb{E}[zz^\top] = \mathbf{I}$ we have that

$$\mathbb{E}[z_j z_i] = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

Using the above in (18) we have that

$$\mathbb{E}[z \odot (\mathbf{H}z)] = \sum_i \mathbf{H}_{ii} e_i \tag{18}$$

which is the diagonal of the Hessian matrix.

Let z be a Rademacher random variable. That is

$$z_i = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases}$$

Thus for $i, j \in \{1, \ldots, d\}$ and $i \neq j$, we have that

$$\mathbf{E}[z_i] = \frac{1}{2} \times 1 - \frac{1}{2} \times 1 = 0$$
$$\mathbf{E}[z_i^2] = \frac{1}{2} \times 1 + \frac{1}{2} \times 1$$
$$\mathbf{E}[z_i z_j] = \mathbf{E}[z_i]\mathbf{E}[z_j] = 0.$$

The same result follows for $z \in \mathcal{N}(0, 1)$ since by definition $\mathbf{E}[z_i] = 0$ and $\mathbf{E}[z_i^2] = 1$. ■

A.1. Proof of Lemma 1

Proof

Let

$$w^* = \arg\min_{w\in\mathbb{R}^d} \frac{1}{2}\|w - w_t\|_{B_t}^2$$
$$\textit{s.t.} \quad f_i(w_t) + \langle \nabla f_i(w_t), w - w_t\rangle = 0.$$

To simplify notation, we denote that

$$a = \frac{1}{\sqrt{2}}(w - w_t),$$

Then,

$$\min \frac{1}{2}||a||^2_{B_t}$$
$$s.t. \quad f_i(w_t) + \langle \nabla f_i(w_t), a \rangle = 0.$$

We introduce λ, and let $L = \frac{1}{2}||a||^2_{B_t} + \lambda(f_i(w_t) + \langle \nabla f_i(w_t), a \rangle)$, taking the partial derivation with respect to a and λ ,

$$\frac{\partial L}{\partial a} = B_t a + \lambda \nabla f_i(w_t) = 0$$
$$\frac{\partial L}{\partial \lambda} = f_i(w_t) + \langle \nabla f_i(w_t), a \rangle = 0.$$

To solve these equations, we get

$$a = -\lambda B_t^{-1} \nabla f_i(w_t)$$
$$\lambda = \frac{f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}}.$$

Finally,

$$\hat{w} = w_t - \frac{f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}} B_t^{-1} \nabla f_i(w_t).$$

■

A.2. Proof of Lemma 2

Proof We can rewrite the slack part of the objective function in (13) as

$$\lambda s + \mu(s - s_t)^2 = \frac{1}{2} \cdot 2\mu \left(s - s_t + \frac{\lambda}{2\mu} \right)^2 + constants \ w.r.t \ w \ and \ s. \tag{19}$$

Dropping constants independent of s and w and let $s^0 = s_t - \frac{\lambda}{2\mu}$ we have that () is equivalent to solving

$$w_{t+1}, s_{t+1} = \arg\min_{w \in \mathbb{R}^d, s \geq 0} \|w - w_t\|^2_{B_t} + 2\mu(s - s^0)^2$$
$$f_i(w_t) + \langle \nabla f_i(w_t), w - w_t \rangle - (s - s^0) \leq s^0$$
$$s \geq 0. \tag{20}$$

(1) If $s^0 \geq f_i(w_t)$ holds then the solution is simply $(w_{t+1}, s_{t+1}) = (w_t, s^0)$.
(2) If $s^0 \leq f_i(w_t)$. And at least one of the inequality constraints must be active at the optimal point as the problem is an L2 projection onto the intersection of two halfspace.
(i) If the constraints $f_i(w_t) + \langle \nabla f_i(w_t), w - w_t \rangle = s$ is active, and let $w - w_t = \alpha$, $s - s^0 = \beta$, then our problem reduces to

$$\alpha_{t+1}, \beta_{t+1} = \arg\min_{w \in \mathbb{R}^d, s \geq 0} \|\alpha\|^2_{B_t} + 2\mu\beta^2$$
$$f_i(w_t) + \langle \nabla f_i(w_t), \alpha \rangle - \beta = s^0$$
$$-s^0 - \beta \leq 0 \tag{21}$$

Let $L = \|\alpha\|^2_{B_t} + 2\mu\beta^2 + \theta(f_i(w_t) + \langle \nabla f_i(w_t), \alpha\rangle - \beta - s^0) + \gamma(-s^0 - \beta)$, and take the derivative with respect to α, β. And get KKT conditions:

$$\begin{cases} \frac{\partial L}{\partial \alpha} = 2B_t\alpha + \theta\nabla f_i(w_t) = 0 \\ \frac{\partial L}{\partial \beta} = 4\mu\beta - \theta - \gamma = 0 \\ \gamma \geq 0 \\ f_i(w_t) + \langle \nabla f_i(w_t), \alpha\rangle - \beta - s^0 = 0 \\ \gamma(-s^0 - \beta) = 0, \end{cases} \tag{22}$$

which is equivalent to

$$\begin{cases} \alpha = -\frac{\theta B_t^{-1}\nabla f_i(w_t)}{2} \\ \beta = \frac{\theta+\gamma}{4\mu} \\ \theta = \frac{4\mu(f_i(w_t) - \frac{\gamma}{4\mu} - s^0)}{1+2\mu\|\nabla f_i(w_t)\|^2_{B_t^{-1}}} \\ \gamma \geq \frac{-2s^0(2\mu\|\nabla f_i(w_t)\|^2_{B_t^{-1}}) - 2f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}} \\ \gamma \geq 0. \end{cases} \tag{23}$$

So when condition $2\mu\|\nabla f_i(w_t)\|^2_{B_t^{-1}} s^0 + f_i(w_t) \geq 0$ holds, then the solution is given by

$$\begin{cases} \beta = \frac{\theta}{4\mu} \\ \alpha = -\frac{\theta B_t^{-1}\nabla f_i(w_t)}{2} \\ \theta = \frac{4\mu(f_i(w_t) - s^0)}{1+2\mu\|\nabla f_i(w_t)\|^2_{B_t^{-1}}}. \end{cases} \tag{24}$$

which is equivalent to

$$\begin{cases} w - w_t = -\frac{(f_i(w_t) - s^0)}{\frac{1}{2\mu} + \|\nabla f_i(w_t)\|^2_{B_t^{-1}}} B_t^{-1}\nabla f_i(w_t) \\ s - s^0 = \frac{(f_i(w_t) - s^0)}{1+2\mu\|\nabla f_i(w_t)\|^2_{B_t^{-1}}}. \end{cases} \tag{25}$$

If not, we have solution as following:

$$\begin{cases} \beta = -s^0 \\ f_i(w_t) + \langle \nabla f_i(w_t), \alpha\rangle = 0. \end{cases} \tag{26}$$

This problem can be solved similarly as proof of Lemma 1, and its solution is given by

$$\begin{cases} w - w_t = -\frac{f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}} B_t^{-1}\nabla f_i(w_t) \\ s - s^0 = -s^0. \end{cases} \tag{27}$$

(ii) If the constraints $s_{t+1} = 0$ is active then our problem reduces to

$$\begin{aligned} &\min_{w\in\mathbb{R}^d} \|w - w_t\|^2_{B_t} \\ &f_i(w_t) + \langle \nabla f_i(w_t), w - w_t\rangle \leq 0 \end{aligned} \tag{28}$$

which is a projection onto a halfspace, and its solution is given by

$$w - w_t = -\frac{f_i(w_t)}{\|\nabla f_i(w_t)\|^2_{B_t^{-1}}} B_t^{-1}\nabla f_i(w_t) \tag{29}$$

To sum up all these above cases can be written as solution which is given by lemma2 (2).

■

Lemma 5 *Let $\delta > 0$,$c \in \mathbb{R}$ and $w, w^0, a \in \mathbb{R}^d$. The closed-form solution to*

$$w', s' = \arg\min_{w\in\mathbb{R}^d, s\in\mathbb{R}^b} \|w - w^0\|_{B_t}^2 + \delta(s - s^0)^2$$

s.t.

$$a^T(w - w^0) + c \leq s, \tag{30}$$

is given by

$$w' = w^0 - \delta\frac{(c - s^0)_+}{1 + \delta\|a\|_{B_t^{-1}}^2} B_t^{-1}a,$$

$$s' = s^0 + \frac{(c - s^0)_+}{1 + \delta\|a\|_{B_t^{-1}}^2}. \tag{31}$$

Proof Let $\alpha = w - w^0, \beta = s - s^0$, then our question becomes

$$\alpha, \beta = \arg\min_{\alpha\in\mathbb{R}^d, \beta\in\mathbb{R}^b} \|\alpha\|_{B_t}^2 + \delta\beta^2$$

s.t.

$$a^T\alpha - \beta + c - s^0 \leq 0. \tag{32}$$

(1)If $w = w^0$ and $s = s^0$ satisfies in the linear inequality constraint, that is if $c \leq s^0$, in which case the solution is simply $w' = w^0$ and $s' = s^0$.
(2) But if $c \geq s^0$, (w^0, s^0) is out of the feasible set, then we need to project (w^0, s^0) onto the boundary of the halfspace. Let $L = \|\alpha\|_{B_t}^2 + \delta\beta^2 + \lambda(a^T\alpha - \beta + c - s^0)$, take the derivative with respect to α , β and λ , make them equal to zero, we get

$$\begin{cases} \frac{\partial L}{\partial \alpha} = 2B_t\alpha + \lambda\alpha = 0 \\ \frac{\partial L}{\partial \beta} = 2\delta\beta - \lambda = 0 \\ \frac{\partial L}{\partial \lambda} = a^T\alpha - \beta + c - s^0 = 0. \end{cases} \tag{33}$$

To solve these problems we have:

$$\lambda = \frac{2(c - s^0)}{\frac{1}{\delta} + \|a\|_{B_t^{-1}}}, \ \alpha = -\frac{\lambda}{2}B_t^{-1}a, \ \beta = \frac{\lambda}{2\delta}. \tag{34}$$

By plugging in and enumerating all possible cases, we get the closed solution (31). ■

A.3. Proof of Lemma 3

Proof The slack variables in the objective function of (14) can be re-written as

$$\mu(s - s_t)^2 + \lambda s^2 = \frac{1}{\hat{\lambda}}(s - \mu\hat{\lambda}s_t)^2 + constant \ w.r.t. \ s,$$

where $\hat{\lambda} = \frac{1}{\mu+\lambda}$. After dropping constants, solving (14) is equivalent to

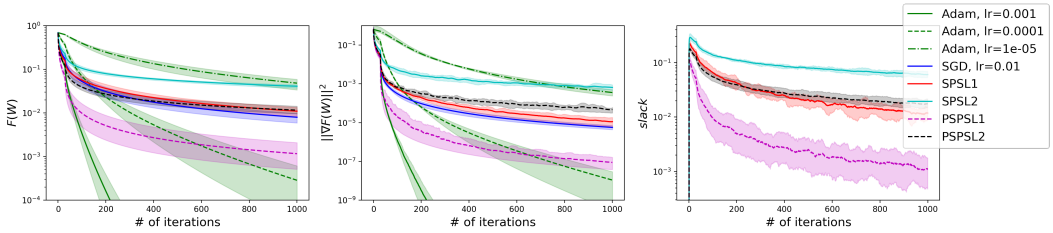
$$w_{t+1}, s_{t+1} = \arg\min_{w\in\mathbb{R}^d, s\in\mathbb{R}} \|w - w_t\|_{B_t}^2 + \frac{1}{\hat{\lambda}}(s - \mu\hat{\lambda}s_t)^2$$

$$f_i(w_t) + \langle\nabla f_i(w_t), w - w_t\rangle \leq s. \tag{35}$$

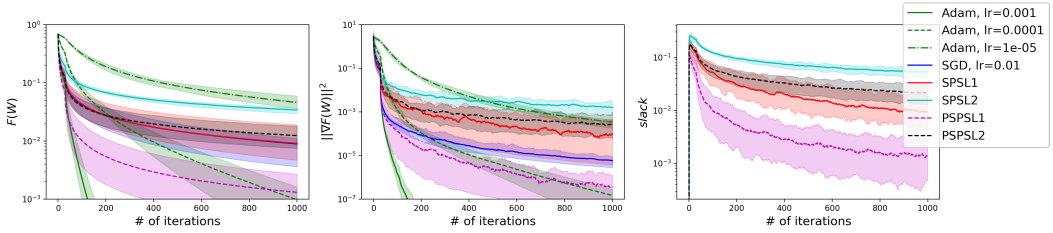
By Lemma (5) with $a \leftarrow \nabla f_i(w_t)$, $c \leftarrow f_i(w_t)$, $s^0 \leftarrow \mu\hat{\lambda}s_t$ and $\delta \leftarrow \frac{1}{\hat{\lambda}}$, we have the solution given by Lemma (3). ■

Appendix B. Additional Experiments

In all experiments the slack parameters $\lambda = 0.01$, $\mu = \frac{1}{2}$ for PSPSL1 and $\mu = 1$ for PSPSL2. Batch sizes of 64 and 1 were used with datasets *mushrooms* and *colon-cancer* respectively. Scaling vector $e = \{\exp(x_i)\}_{i=1}^{d}$ where x_i is generated from a uniform distribution on the interval $[-k, k]$.

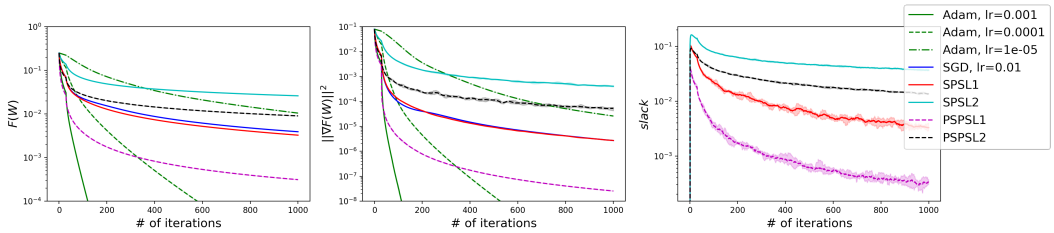


(*a*) $k = 1$

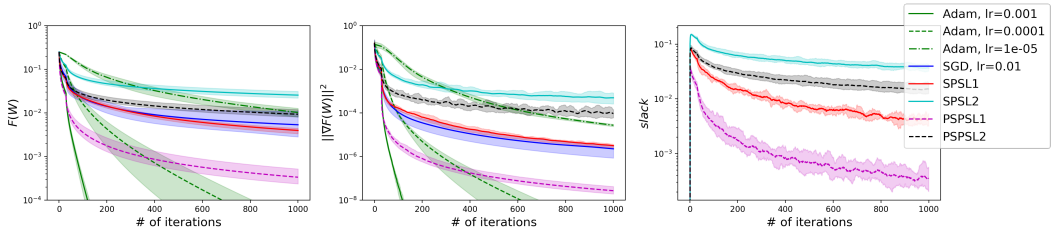


(*b*) $k = 2$

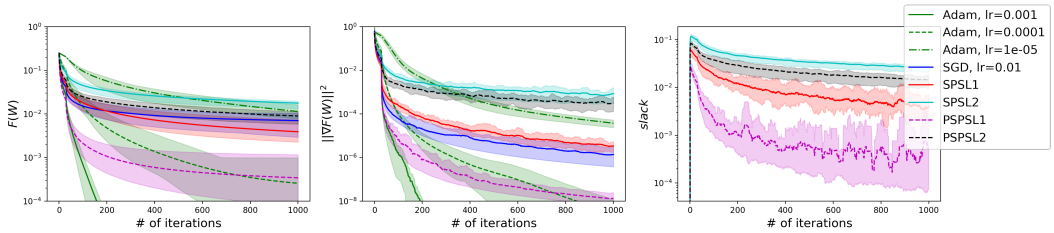
Figure 2: Performance comparison of PSPSL1 and PSPSL2 to non-scaled versions of SPS, SGD and Adam on badly scaled *mushrooms* datasets. All methods are trained with Logistic Regression.



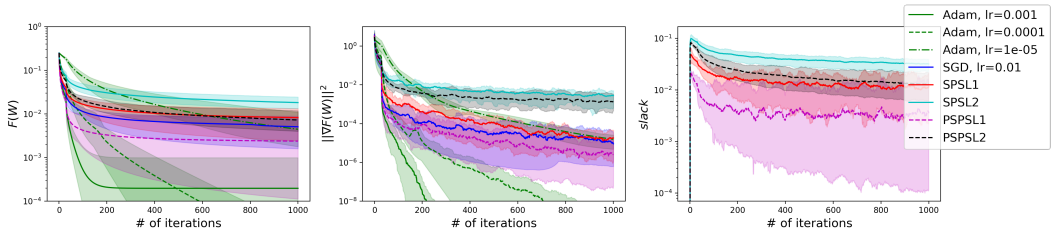
(*a*) original



(*b*) $k = 1$

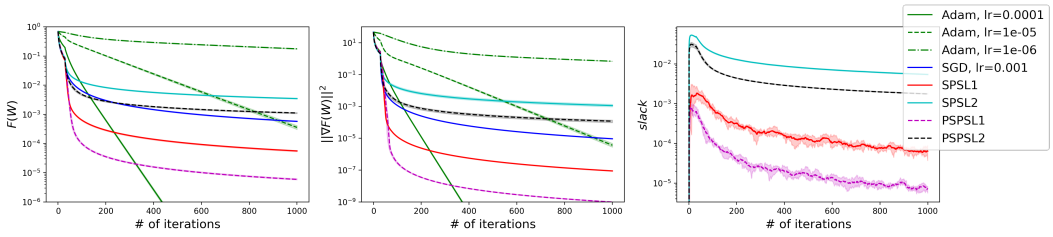


(*c*) $k = 2$

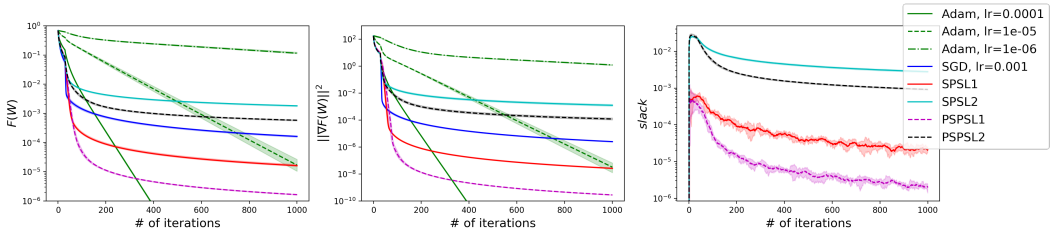


(*d*) $k = 3$

Figure 3: Performance comparison of PSPSL1 and PSPSL2 to non-scaled versions of SPS, SGD and Adam on original badly scaled *mushrooms* datasets. All methods are trained with NLLSQ.

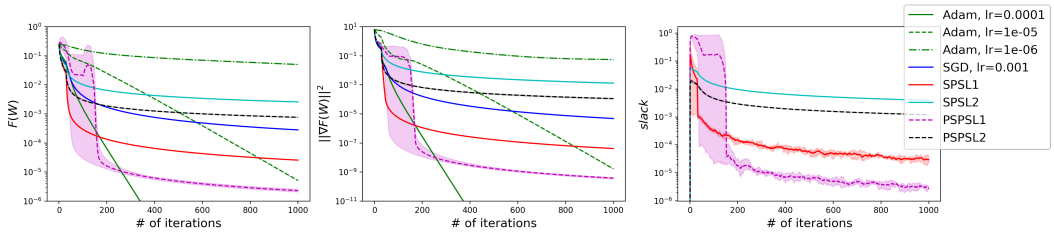


(*a*) $k = 1$

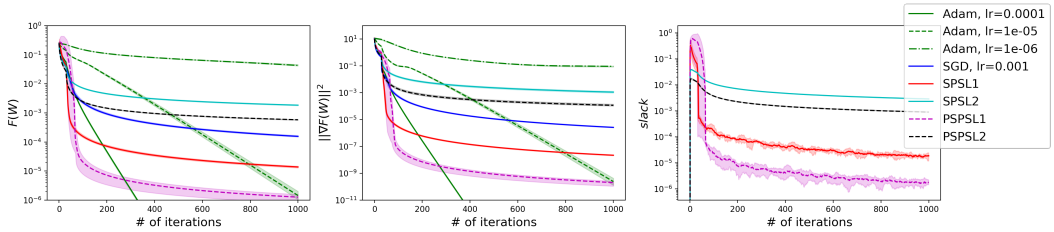


(*b*) $k = 2$

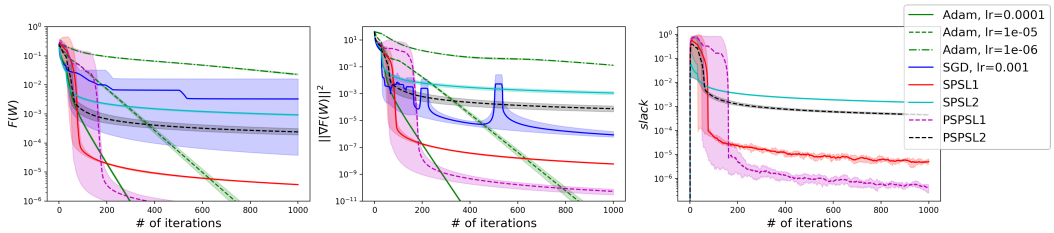
Figure 4: Performance comparison of PSPSL1 and PSPSL2 to non-scaled versions of SPS, SGD and Adam on original badly scaled *colon-cancer* datasets. All methods are trained with Logistic Regression.



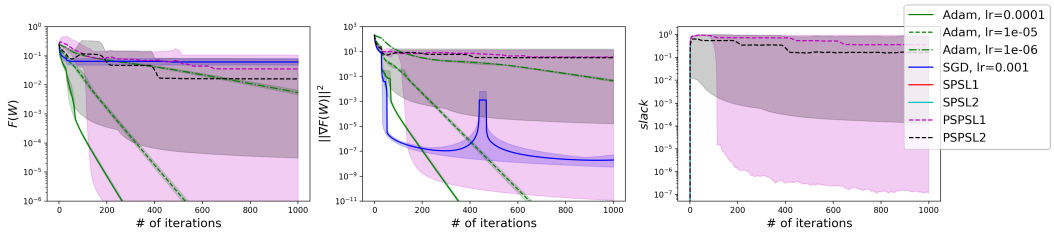
(a) original



(b) $k = 1$



(c) $k = 2$



(d) $k = 3$

Figure 5: Performance comparison of PSPSL1 and PSPSL2 to non-scaled versions of SPS, SGD and Adam on original badly scaled *colon-cancer* datasets. All methods are trained with NLLSQ.