A NOVEL CONVERGENCE ANALYSIS FOR THE STOCHASTIC PROXIMAL POINT ALGORITHM

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the stochastic proximal point algorithm (SPPA) for general empirical risk minimization (ERM) problems as well as deep learning problems. We present an efficient implementation of SPPA with minor modification for different problem definitions and we observe that efficiently implemented SPPA has faster and more stable convergence than the celebrated stochastic gradient descent (SGD) algorithm, and its many variations, for both convex and non-convex problems. Due to the fact that per-iteration update of SPPA is defined abstractly and has long been considered expensive, its convergence proof has not been well-studied until recently. In this paper, we close the theoretical gap by providing its convergence for convex problems. Our proof technique is different from some of the recent attempts. As a result, we present a surprising result that SPPA for convex problems may converge *arbitrarily fast*, depending on how the step sizes are chosen. As a second contribution, we also show that for some of the canonical ERM problems and deep learning problems, each iteration of SPPA can be efficiently calculated either in closed form or closed to closed form via bisection—the resulting complexity is exactly the same as that of SGD. Real data experiments showcase its effectiveness in terms of convergence compared to SGD and its variants.

1 INTRODUCTION

It has been widely accepted that when training large-scale machine learning models, the training algorithm should act in a sample-by-sample manner in order to reduce computational and memory overhead—the size of the data set may be too large that calculating the full gradient information is too costly. Moreover, most machine learning problems does not have to be solved with very high accuracy, since the ultimate goal is not to fit the training data but rather to generalize the algorithm well such that the performance is decent on unseen data.

Most existing stochastic algorithms are based upon the stochastic gradient descent (SGD) framework (Bottou et al., 2018). SGD is extremely easy to implement and provides asymptotic convergence, although the convergence rate is generally slow (and subject to careful choice of step sizes). Various approaches have been proposed to accelerate the plain vanilla SGD. Reducing the variance of the stochastic gradient and introducing adaptive learning schemes are two main lines of research. SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014) (and their follow-up works such as (Defazio, 2016)) focus on reducing the variance of the stochastic gradient descent, at the cost of increased time or memory complexities (to the order of the entire data set). On the other hand, AdaGrad (Duchi et al., 2011) and Adam (Diederik P. Kingma, 2014) introduce adaptive learning schemes and effectively keep the algorithm fully stochastic and light-weight. Besides these practical improvements, theoretical progress has been made by on quantifying the best possible convergence rate using first-order information (Lei et al., 2017; Allen-Zhu, 2017; 2018a;b).

1.1 STOCHASTIC PROXIMAL POINT ALGORITHM (SPPA)

In this paper, we explore a different type of stochastic algorithm called the stochastic proximal point algorithm (SPPA), also known as stochastic proximal iterations (Ryu & Boyd, 2014) or incremental proximal point method (Bertsekas, 2011a;b). Consider the following optimization problem with the

objective function in the form of a finite sum of component functions

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}) = F(\boldsymbol{w}). \tag{1}$$

SPPA takes the following simple form:

Algorithm 1 Stochastic proximal point algorithm (SPPA)

1: **repeat**
2: randomly draw i_t uniformly from $\{1, \ldots, n\}$
3: $\boldsymbol{w}_{t+1} \leftarrow \arg\min_{\boldsymbol{w}} \lambda_t f_{i_t}(\boldsymbol{w}) + (1/2)\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 = \mathrm{Prox}_{\lambda_t f_{i_t}}(\boldsymbol{w}_t)$
4: **until** convergence

Line 3 of Algorithm 1 calculates the proximal operator of the function $\lambda_t f_i$ evaluated at $\boldsymbol{w}_t$, denoted as $\mathrm{Prox}_{\lambda_t f_i}(\boldsymbol{w}_t)$. This is the stochastic version of the proximal point algorithm, which dates back to Rockafellar (1976).

SPPA has an abstract per-iteration update rule and it acquires more information from the problem than solely the first order derivatives, which makes it not as universally applicable as SGD. Yet, thanks to the more information inquired, it is possible to obtain faster and more robust convergence guarantees. While some 'accelerated' versions of SGD demand additional time or space overhead to go over the entire data set, SPPA does not have any overhead and it is suitable to be used in a completely online setting, it performs well even the data samples are not revisited again.

To the best of our knowledge, people started studying the convergence behavior of SPPA only recently (Bertsekas, 2011a; Ryu & Boyd, 2014; Bianchi, 2016; Pătraşcu, 2020; Toulis et al., 2021). Somewhat surprisingly, the convergence analysis of SPPA draws little resemblance to its deterministic counterpart, the proximal point algorithm. This is unlike the case for SGD, of which the convergence analysis follows almost line-by-line to that of the subgradient method. Moreover, existing analysis of SPPA shows no improvement in terms of convergence rate, which seems counter-intuitive due to the nature of the updates. Most authors also accept the premise that the proximal operator is sometimes difficult to evaluate, and thus proposed variations to the plain vanilla version to handle more complicated problem structures (Wang & Bertsekas, 2013; Duchi & Ruan, 2018; Asi & Duchi, 2019; Davis & Drusvyatskiy, 2019).

1.2 Contributions

The main contribution of this paper is to provide a completely novel convergence analysis of SPPA for general convex problems. This contribution, together with the efficient implementation strategies discussed in the appendix, results in great practical results in almost all of the classical empirical risk minimization (ERM) problems in large-scale machine learning. While there exists some convergence results of SPPA for convex problems, the novel analysis provided in this work requires minimal assumptions (nothing but the convexity of the loss functions). The new convergence analysis also shows great resemblance to its deterministic counterpart, which is not the case from other works.

In the appendix we will discuss how to efficiently compute the abstract per-iteration update rule for SPPA with complexity that is comparable to SGD-type methods. An interesting observation is that in a lot of ERM examples the resulting update has a similar form like SGD, but with a smartly chosen step size derived from the proximal update. For a class of smooth risks such as the logistic loss, we appeal to reformulate the optimization problem such that we get close to closed form solution via bisection. We also briefly discuss how to modify the algorithm to have the ability to handle nonsmooth regularization terms such as the L_1 norm, using the stochastic Douglas-Rachford splitting approach.

Finally, we apply SPPA with efficient implementations to a large variety of classification and regression problems. Numerical experiments are conducted not only for linear classification and regression (in the appendix) models, but also nonconvex deep neural network problems. Although the convergence analysis provided in this paper does not cover nonconvex cases, empirical results suggest that it is still worth treating SPPA as an effective alternative for deep learning.

2 CONVERGENCE ANALYSIS

In this section we provide convergence analysis of SPPA for general convex loss functions equation 1 to a global minimum in expectation. In recent years there have been some work tackling the same problem, e.g., Bertsekas (2011a); Pătraşcu (2020); Toulis et al. (2021). In this paper, however, we provide new convergence analysis that is much easier to understand while requiring nearly no assumptions other than convexity. For this reason we believe the theoretical contribution is significant enough as it broadens the applicability of SPPA.

There is a well-known resemblance between proximal methods and gradient descent, assuming the loss function is differentiable: while a full gradient descent step takes the form $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \lambda_t \nabla F(\boldsymbol{w}_t)$, the definition of a full proximal step guarantees that $\lambda_t \nabla F(\boldsymbol{w}_{t+1}) = \boldsymbol{w}_{t+1} - \boldsymbol{w}_t$, meaning that $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \lambda_t \nabla F(\boldsymbol{w}_{t+1})$. Therefore, one might expect that a well-established convergence analysis of SGD for nonconvex problems, for example (Bottou et al., 2018, §4.3), can be seamlessly applied to SPPA. However, when applied in a stochastic fashion, the situation is a little more complicated.

Consider $\mathbb{E}[\nabla f_i(\boldsymbol{w}_t) \mid \boldsymbol{w}_t]$, where the expectation is taken over the sampling procedure conditioned on $\boldsymbol{w}_t$, for SGD we typically require $\nabla f_i(\boldsymbol{w}_t)$ to be an unbiased estimator of $\nabla F(\boldsymbol{w}_t)$, which is easy to satisfy if i_t is uniformly sampled from $\{1, \dots, n\}$ given $\boldsymbol{w}_t$. For SPPA then one needs to consider $\mathbb{E}[\nabla f_i(\boldsymbol{w}_{t+1}) \mid \boldsymbol{w}_t]$, again over the sampling procedure conditioned on $\boldsymbol{w}_t$. This is in fact difficult to quantify because the update $\boldsymbol{w}_{t+1}$ depends on the sample that is drawn from the data set. The equation $\mathbb{E}[\nabla f_i(\boldsymbol{w}_{t+1}) \mid \boldsymbol{w}_t] = \nabla F(\boldsymbol{w}_{t+1})$ does not make sense because $\boldsymbol{w}_{t+1}$ on the right-hand-side is still random conditioned on $\boldsymbol{w}_t$. It is for this reason that existing analysis of SPPA differs drastically from its deterministic counterpart PPA (Bertsekas, 2011a; Ryu & Boyd, 2014; Bianchi, 2016).

What we can show, however, is that the distribution of i_t is still uniform conditioned on $\boldsymbol{w}_{t+1}$ instead of $\boldsymbol{w}_t$, as formalized in the following proposition.

Lemma 1. *At every iteration of SPPA (Algorithm 1), we have the conditional probability*

$$p(i_t | \boldsymbol{w}_{t+1}) = 1/n.$$

The proof of Lemma 1 is relegated to the supplementary material. What Lemma 1 suggests is that given the current iterate $\boldsymbol{w}_{t+1}$, without knowing its predecessors $\boldsymbol{w}_t$ and beyond, every component function f_i is equally likely to be picked as the proximal update that leads to $\boldsymbol{w}_{t+1}$. In other words, conditioned on the current iterate without knowing the past, we do not gain additional information about which component function f_i is more likely to have been selected. As it turns out, Lemma 1 significantly simplifies the following convergence analysis while showing resemblance to that of deterministic PPA. What is more, the resulting analysis requires no assumptions other than convexity: typical assumptions such as Lipschitz smoothness or bounded variance are not required.

Remark. Lemma 1 holds for $t = 0$ if the initialization $\boldsymbol{w}_0$ is random, and the distribution from which it is drawn has the sample space the same as the domain of the objective function equation 1. In practice $\boldsymbol{w}_0$ is usually drawn from a distribution that is independent from the cost function, say $\mathcal{N}(0, \boldsymbol{I})$. In this case, for a given value of $\boldsymbol{w}_1$, there is no chance of eliminating possible $\boldsymbol{w}_0$'s because any corresponding $\boldsymbol{w}_0$ is in the sample space of the initialization distribution.

2.1 GENERIC CONVEX CASE WITHOUT STRONG CONVEXITY

Under Lemma 1, we have the following proposition, which serves as the stepping stone for our main convergence results.

Proposition 1. *Suppose each $f_1, \dots, f_n$ is convex, then at iteration t of SPPA (Algorithm 1) we have*

$$2\lambda_t \left(F(\boldsymbol{w}_{t+1}) - F(\boldsymbol{w}_\star)\right) \leq \mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 \mid \boldsymbol{w}_{t+1}] - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2, \tag{2}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1.

Proof. We start by the equation

$$\begin{aligned} \|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 &= \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1} + \boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 \\ &= \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2(\boldsymbol{w}_t - \boldsymbol{w}_{t+1})^\top (\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star) + \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|^2. \end{aligned} \tag{3}$$

According to the definition $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \lambda_t f_{i_t}(\boldsymbol{w}) + (1/2)\|\boldsymbol{w} - \boldsymbol{w}_t\|^2$, we know that $\boldsymbol{w}_t - \boldsymbol{w}_{t+1}$ is a subgradient of $\lambda_t f_{i_t}$ at $\boldsymbol{w}_{t+1}$. Therefore

$$\lambda_t f_{i_t}(\boldsymbol{w}_{t+1}) + (\boldsymbol{w}_t - \boldsymbol{w}_{t+1})^\top(\boldsymbol{w} - \boldsymbol{w}_{t+1}) \le \lambda_i f_{i_t}(\boldsymbol{w})$$

at any $\boldsymbol{w}$. Let $\boldsymbol{w} = \boldsymbol{w}_\star$ and substitute it in equation 3, we have

$$\begin{aligned}\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 &\ge \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2\lambda_t(f_{i_t}(\boldsymbol{w}_{t+1}) - f_{i_t}(\boldsymbol{w}_\star)) + \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|^2 \\ &\ge \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2\lambda_t(f_{i_t}(\boldsymbol{w}_{t+1}) - f_{i_t}(\boldsymbol{w}_\star)).\end{aligned}$$

Finally, taking condition expectations on both sides given $\boldsymbol{w}_{t+1}$ (but not $\boldsymbol{w}_t$ nor i_t)

$$\mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 \mid \boldsymbol{w}_{t+1}] \ge \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2\lambda_t \mathbb{E}[(f_i(\boldsymbol{w}_{t+1}) - f_i(\boldsymbol{w}_\star) \mid \boldsymbol{w}_{t+1}].$$

According to Lemma 1, the conditional distribution over i_t is uniform, thus

$$\mathbb{E}[f_{i_t}(\boldsymbol{w}_{t+1}) - f_{i_t}(\boldsymbol{w}_\star) \mid \boldsymbol{w}_{t+1}] = F(\boldsymbol{w}_{t+1}) - F(\boldsymbol{w}_\star),$$

and we obtain equation 2. □

With the help of Proposition 1, the rest of the results follows straight-forwardly.

Theorem 1. *If all $f_1, \ldots, f_n$ are convex, and an initialization $\boldsymbol{w}_0$ is picked from a distribution with the sample space the same as the domain of F, then the sequence $\{\boldsymbol{w}_t\}$ generated by SPPA (Algorithm 1) satisfies*

$$\lim_{T\to\infty} \inf_{t\le T} \mathbb{E}[F(\boldsymbol{w}_t)] - F(\boldsymbol{w}_\star) \le \frac{1}{2\sum_{t=1}^{\infty}\lambda_t}\mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2, \tag{4}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1. The term $\mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2$ is a constant for any approach of initializing $\boldsymbol{w}_0$.

Proof. Taking total expectation over equation 2, we have

$$2\lambda_t\left(\mathbb{E}[F(\boldsymbol{w}_{t+1})] - F(\boldsymbol{w}_\star)\right) \le \mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2] - \mathbb{E}[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2].$$

Sum over all $t = 1, 2, \ldots, T$, we have

$$2\sum_{t=1}^{T}\lambda_t\left(\mathbb{E}[F(\boldsymbol{w}_{t+1})] - F(\boldsymbol{w}_\star)\right) \le \mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2 - \mathbb{E}\|\boldsymbol{w}_T - \boldsymbol{w}_\star\|^2 \le \mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2.$$

On the left-hand-side we have (by definition) $\inf \mathbb{E}[F(\boldsymbol{w}_t)] \le \mathbb{E}[F(\boldsymbol{w}_\tau)]$ for any τ. Applying that, dividing both sides by $2\sum_t \lambda_t$, and letting $T \to \infty$, we get equation 4. □

Remark. The proofs here take conditional expectations *backwards*, i.e., conditioned on $\boldsymbol{w}_{t+1}$ and average over the *previous* iteration. This may be a little counter-intuitive. However, there is nothing wrong mathematically—expectation is a linear operator so $\mathbb{E}[A + B] = \mathbb{E}[A] + \mathbb{E}[B]$ under all circumstances; if an inequality $f(x) \le g(x)$ holds for all values of x in its sample space, then $\mathbb{E}[f(x)] \le \mathbb{E}[g(x)]$ for any distribution over x since it is just a nonnegative sum on both sides.

Theorem 1 states a generic results on the convergence of SPPA. The left-hand-side of equation 4 is, by definition, nonnegative; the right-hand-side, however, goes to zero if the infinite sum $\sum_t \lambda_t \to \infty$. This implies that the infimum if $\mathbb{E}[F(\boldsymbol{w}_t)]$ goes to zero for appropriately chosen step sizes. Somewhat surprisingly, the only assumption we made was that the functions are convex. The celebrated SGD, on the other hand, requires at least two more assumptions: Lipschitz smoothness and that the stochastic gradients have bounded variance. What is more, the flexible choice of λ_t means that we can make the convergence rate *arbitrarily* fast, by allowing λ_t to be increasing (rather than decreasing in most gradient-based methods). Of course this may lead to large variance of the sequence, which we may want to avoid in practice.

Here we provide the convergence rate for two commonly used step size rules.

Corollary 1. *If all $f_1, \ldots, f_n$ are convex, then the sequence $\{\boldsymbol{w}_t\}$ generated by SPPA (Algorithm 1) with a constant step size $\lambda_t = \lambda$ satisfies*

$$\inf_{t\le T} \mathbb{E}[F(\boldsymbol{w}_t)] - F(\boldsymbol{w}_\star) \le \frac{1}{2\lambda T}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2, \tag{5}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1.

The proof is straight-forward by substituting λ_t with λ in equation 4. This shows that a constant step size (regardless of its value) gives a $\mathcal{O}(1/T)$ sublinear convergence rate.

Corollary 2. *If all $f_1, \ldots, f_n$ are convex, then the sequence $\{\boldsymbol{w}_t\}$ generated by SPPA (Algorithm 1) with an increasing step size rule $\lambda_t = t\lambda$ satisfies*

$$\inf_{t \le T} \mathbb{E}[F(\boldsymbol{w}_t)] - F(\boldsymbol{w}_\star) \le \frac{1}{\lambda T(T+1)} \|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2, \tag{6}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1.

The proof comes from the arithmetic series $1+2+\cdots+T = T(T+1)/2$. This shows that the increasing step size rule $\lambda_t = t\lambda$ gives a $\mathcal{O}(1/T^2)$ sublinear convergence rate, the same as Nesterov's optimal gradient algorithm (2013).

While the expected convergence looks excellent, we notice that in practice the performance also depends on the variance of the sequence generated by SPPA, which could be large if the expected convergence rate is too fast. We can even have linear convergence rate by letting λ_t increase exponentially, but the variance would be so large that it makes little practical sense. As we will see in the experiment section, a constant step size rule still gives the best performance in most cases.

2.2 Strongly convex case

Similar to most other algorithms, convergence can be significantly improved if the loss functions are strongly convex.

Proposition 2. *Suppose each $f_1, \ldots, f_n$ is strongly convex with parameter μ, and an initialization $\boldsymbol{w}_0$ is picked from a distribution with the sample space the same as the domain of F, then at iteration t of SPPA (Algorithm 1) we have*

$$2\lambda_t \left(F(\boldsymbol{w}_{t+1}) - F(\boldsymbol{w}_\star)\right) \le \mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 \mid \boldsymbol{w}_{t+1}] - (1 + \lambda_t/\mu)\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2, \tag{7}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1.

Proof. Following equation equation 3, we again have that $\boldsymbol{w}_t - \boldsymbol{w}_{t+1}$ is a subgradient of f_{i_t}, which satisfies the following inequality if it is strongly convex

$$\lambda_t f_{i_t}(\boldsymbol{w}_{t+1}) + (\boldsymbol{w}_t - \boldsymbol{w}_{t+1})^\top(\boldsymbol{w} - \boldsymbol{w}_{t+1}) + \frac{\lambda_t}{2\mu}\|\boldsymbol{w} - \boldsymbol{w}_{t+1}\|^2 \le \lambda_i f_{i_t}(\boldsymbol{w})$$

at any $\boldsymbol{w}$. Let $\boldsymbol{w} = \boldsymbol{w}_\star$ and substitute it in equation 3, we have

$$\begin{aligned}\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 &\ge \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2\lambda_t(f_{i_t}(\boldsymbol{w}_{t+1}) - f_{i_t}(\boldsymbol{w}_\star)) + \frac{\lambda_t}{\mu}\|\boldsymbol{w} - \boldsymbol{w}_{t+1}\|^2 + \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|^2 \\ &\ge (1 + \lambda_t/\mu)\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 + 2\lambda_t(f_{i_t}(\boldsymbol{w}_{t+1}) - f_{i_t}(\boldsymbol{w}_\star)).\end{aligned}$$

Taking condition expectations on both sides given $\boldsymbol{w}_{t+1}$ (but not $\boldsymbol{w}_t$ nor i_t) and invoking Lemma 1, we obtain equation 7. □

The main convergence of SPPA for strongly convex functions is the following.

Theorem 2. *If all $f_1, \ldots, f_n$ are strongly convex with parameter μ, then the sequence $\{\boldsymbol{w}_t\}$ generated by SPPA (Algorithm 1) with constant step sizes $\lambda_t = \lambda$ satisfies*

$$\inf_{t \le T} \mathbb{E}[F(\boldsymbol{w}_t)] - F(\boldsymbol{w}_\star) \le \frac{1/(2\mu)}{(1 + \lambda/\mu)^T - 1}\mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2, \tag{8}$$

where $\boldsymbol{w}_\star$ denotes an optimal solution of equation 1. The term $\mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2$ is a constant for any approach of initializing $\boldsymbol{w}_0$. For large enough T, the denominator on the right-hand-side of equation 7 is almost $(1 + \lambda/\mu)^T$, which indicates a linear convergence rate.

Proof. Letting $\lambda_t = \lambda$, taking total expectation over equation 7 and multiplying both sides by $(1 + \lambda/\mu)^t$, we have

$$2\lambda(1 + \lambda/\mu)^t \left(\mathbb{E}[F(\boldsymbol{w}_{t+1})] - F(\boldsymbol{w}_\star)\right) \le (1+\lambda/\mu)^t \mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2] - (1+\lambda/\mu)^{t+1}\mathbb{E}[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2].$$

Sum over all $t = 1, 2, \ldots, T$, we have

$$2\lambda \sum_{t=1}^{T} (1+\lambda/\mu)^t \left(\mathbb{E}[F(\boldsymbol{w}_{t+1})] - F(\boldsymbol{w}_\star)\right) \leq \mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2 - (1+\lambda/\mu)^T \mathbb{E}\|\boldsymbol{w}_T - \boldsymbol{w}_\star\|^2 \leq \mathbb{E}\|\boldsymbol{w}_0 - \boldsymbol{w}_\star\|^2.$$

On the left-hand-side we have (by definition) $\inf \mathbb{E}[F(\boldsymbol{w}_t)] \leq \mathbb{E}[F(\boldsymbol{w}_\tau)]$ for any τ. Replacing every $\mathbb{E}[F(\boldsymbol{w}_t)]$ with the infimum and applying the geometric series

$$\sum_{t=0}^{T} (1+\lambda/\mu)^t = \frac{(1+\lambda/\mu)^T - 1}{\lambda/\mu},$$

we get equation 8. □

Final remark. In this section, we provided convergence of SPPA *in expectation*, following the strategies utilized in the tutorial (Bottou et al., 2018) which is easy to understand. In the supplementary we also provide a stronger *almost sure* convergence using Martingales theorem.

2.3 Extension to stochastic Douglas-Rachford Splitting

In many cases the ERM formulation includes a regularization term $g(\boldsymbol{w})$ as

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}) + g(\boldsymbol{w}). \tag{9}$$

The most widely adopted regularization include the Tikhonov regularization $\rho\|\boldsymbol{w}\|^2$, the lasso regularization $\rho\|\boldsymbol{w}\|_1$ to promote sparsity, and the group-lasso regularization to promote group-sparsity. The Tikhonov regularization can easily be incorporated into the ERM parts without affecting the efficiency of the proximal updates. However, for the nonsmooth ones it may not be ideal to spread it into the stochastic terms as $f_i(\boldsymbol{w}) + g(\boldsymbol{w})$. We propose to modify the Douglas-Rachford splitting in order to handle situation like this, which keeps track of two sequences $\{\boldsymbol{w}_t\}$ and $\{\tilde{\boldsymbol{w}}_t\}$:

Algorithm 2 Stochastic Douglas-Rachford splitting (SDRS)

1: **repeat**
2: $\quad \boldsymbol{w}_{t+1} \leftarrow \mathrm{Prox}_{\lambda_t g}(\tilde{\boldsymbol{w}}_t)$
3: $\quad$ randomly draw i_t uniformly from $\{1, \ldots, n\}$
4: $\quad \tilde{\boldsymbol{w}}_{t+1} \leftarrow \tilde{\boldsymbol{w}}_t + \mathrm{Prox}_{\lambda_t f_{i_t}}(2\boldsymbol{w}_{t+1} - \tilde{\boldsymbol{w}}_t) - \boldsymbol{w}_{t+1}$
5: **until** convergence

There has been some attempts to study the convergence of SDRS in the name of stochastic ADMM (Ouyang et al., 2013; Zhong & Kwok, 2014; Huang et al., 2019) or online ADMM (Wang & Banerjee, 2012). However, most of them do not consider a full proximal update but rather a 'linearized' update similar in the form of a (stochastic) gradient step; Wang & Banerjee (2012) considers the full proximal update but in an online setting using regret bounds to prove convergence, which resulted in a slower convergence rate as is typical comparing incremental verses stochastic methods.

3 Experiments

We show some real data experiments to demonstrate effectiveness of the proposed SPPA implementations in linear regression and binary classification problems. We compare our proposed algorithms with SGD and its three variants: SGD with momentum (Sutskever et al., 2013), AdaBelief (Zhuang et al., 2020), and Adam (Diederik P. Kingma, 2014), with the default settings suggested in their original papers. We used the PyTorch (Paszke et al., 2019) implementation of the algorithms and used the same platform for implementing our customized optimization algorithm. We ran SPPA with our proposed implementations with various λ values among $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. The performance was consistent with different settings. We ran all the experiments with 100 different seeds, 0-100, to show the robustness of the algorithms to different stochastic settings. We ran all the experiments in Google Colab Pro environment. [1]

[1] `https://colab.research.google.com/notebooks/intro.ipynb`

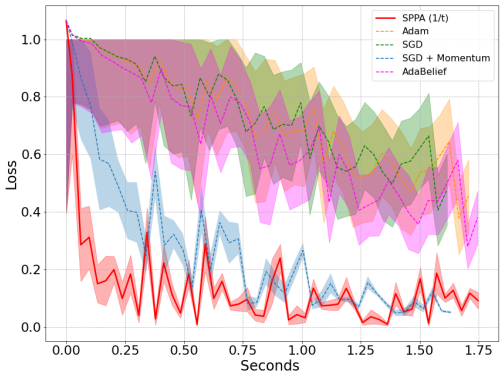

Figure 1: SVM on Bank Note Authentication: regularized hinge loss per seconds, batch size 1

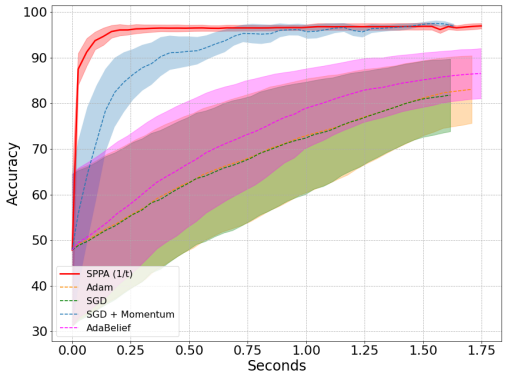

Figure 2: SVM on Bank Note Authentication: classification accuracy per seconds, batch size 1

3.1 Linear classification

We perform binary classification using SVM and logistic regression. Our aim is to show the performance of our proposed SPPA algorithms compared to other state of the art optimization algorithms using two different datasets, Bank Note Authentication and IMDB. The inner loop in SPPA algorithm is run at most 20 times for the results presented below.

Bank Note Authentication Dataset. The bank note authentication dataset is a publicly available dataset in UCI machine learning repository Dua & Graff (2017). Data consists of genuine and counterfeit banknote images. The dataset has 1372 instances. Each instance has 5 attributes out of which 4 are the features and one is the target attribute. The target attribute contains two values: 0 and 1, where 0 represents genuine note and 1 represents fake note. The ratio of the two classes are balanced 55/45 (genuine/counterfeit). We pre-process the data to have target values to be -1 and 1. We perform classification using SVM and logistic regression. Considering the per iteration complexity of SPPA, we plotted the loss-time, accuracy-time figures, Figure 1 and Figure 2 respectively. The Figure 1 demonstrates how the regularized hinge loss value decreases with respect to time (seconds) using different algorithms. The Figures 1 and 2 show that with our efficient implementation, SPPA takes less amount of time to reach to a smaller loss, and a greater accuracy despite of its per iteration complexity. The range shown in the plots clearly shows that SPPA is more robust to change in stochastic settings.Due to lack of space we present further experiments on the performance of SPPA using logistic regression on Bank Note Authentication dataset in supplementary.

IMDB Dataset. IMDB is large movie review dataset, used for binary sentiment classification Maas et al. (2011). There are 25,000 movie reviews for training, and 25,000 movie reviews for testing. We used the processed bag of words format to run our experiments. The ratio of the two classes are balanced, 50/50 (positive sentiment/negative sentiment). The target attribute contains two values: -1 and 1, where -1 represents negative sentiment whereas 1 represents positive sentiment. Similar to Bank Note Authentication dataset, we perform classification using SVM and logistic regression with IMDB dataset. SPPA outperforms the other algorithms when we use logistic regression to do the classification. Surprisingly, SGD Momentum diverged with this particular data set for logistic regression problem. Even though SGD Momentum performs similar to our proposed SPPA in most cases, this experiment shows that it is not as stable as SPPA. We tried different parameter settings for all algorithms and the presented results are the best performance of each algorithm. The Figure 3 shows the logistic loss, and the Figure 4 shows the accuracy per seconds. Focusing on the Figure 3, we see that the logistic loss follows a decreasing trend with SPPA whereas with other algorithms the decrease is not too visible. Moreover, Figure 4 shows that SPPA reaches to a higher test accuracy compared to the other algorithms.The performance of SPPA is similar, but not particularly better than the state of the art methods when we use SVM. Related experiment results are available in supplementary.

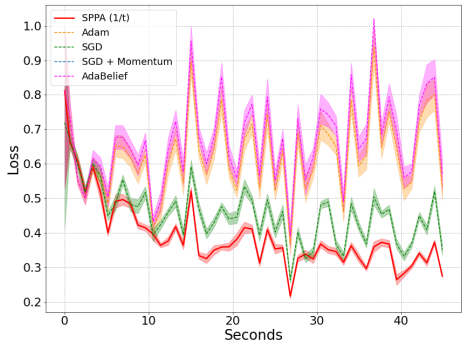

Figure 3: Logistic Regression on IMDB: loss per seconds

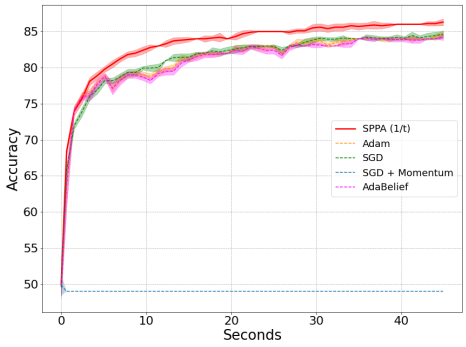

Figure 4: Logistic Regression on IMDB: classification accuracy per seconds

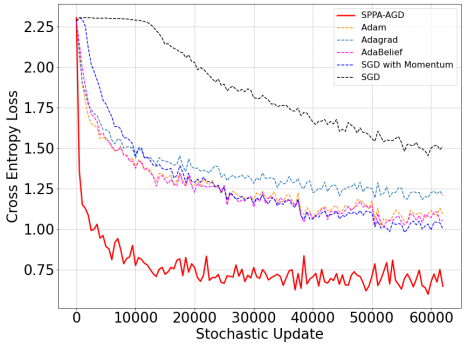

Figure 5: CNN on CIFAR10: cross entropy loss on test data

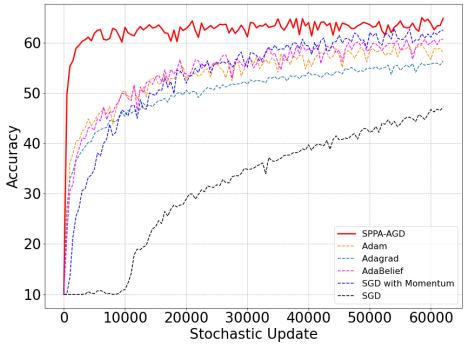

Figure 6: CNN on CIFAR10: prediction accuracy on test data

3.2 Deep neural net classification

CIFAR10. CIFAR10 (Krizhevsky et al., 2010) consists of 50,000 training and 10,000 test images of size 32×32 in 10 classes. The network we use for this experiment is based on convolutional deep neural network. It consists of 5 layers, each of the first 2 being a combination of 5×5 convolutional filters and 2×2 max pooling and last 3 being fully connected. The loss function is cross entropy loss, and the activation function is RELU. Batch size is 4. To show the behaviour of the algorithms more in detail we recorded the loss values at every 500 stochastic update, the experiment is run for 5 epochs and in total 125 updates are presented ($25 \times 500 \times 4$ giving the number of training samples). The accuracy is calculated on the test data at every 500 stochastic update overall for 5 epochs, using the accuracy calculation in PyTorch tutorial. Convergence plots are shown in Figures 5 and 6. To show that SPPA works not only on the CNN architecture, we also tried training a residual neural network (ResNet) on CIFAR10. For the details about the network, one can refer to 20 layer ResNet architecture on CIFAR10 (He et al., 2016). In our settings the batch size is 32. The results are shown in Figures 7 and 8. As we can see, the performance is indeed consistent as in the CNN case.

MNIST. MNIST Handwritten Digits Data set (LeCun et al., 2010) is a common data set used for showing the effectiveness of many state of art the algorithms. It consists of 60,000 training, 10,000 test images of size 28×28 in 10 classes. The network we used for this experiment is based on convolutional deep neural network. It consists of 5 layers, each of the first 3 being combination of 3×3 convolutional filters and 2×2 max pooling with stride of 2 and last 2 being fully connected. The loss function is cross entropy loss, and the activation function is RELU. Batch size is 64. To show the behaviour of the algorithms more in detail we recorded the loss values at every 25 stochastic update, the experiment is run for 1 epoch and in total 38 updates are presented ($25 \times 38 \times 64$ giving the number of training samples). The accuracy is calculated on the test data at every 500 stochastic update, using the accuracy calculation in PyTorch tutorial. [2] As observed in Figures 9 and 10, SPPA-based methods

[2] https://github.com/pytorch/tutorials/blob/master/beginner_source/blitz/cifar10_tutorial.py

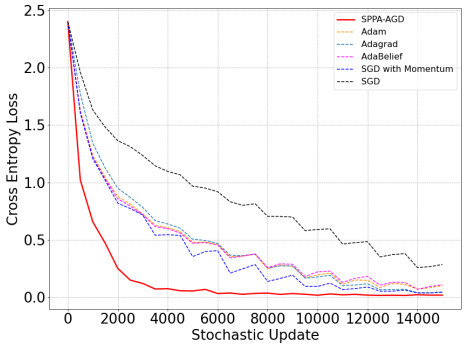

Figure 7: ResNet on CIFAR10: cross entropy loss, batch size 32

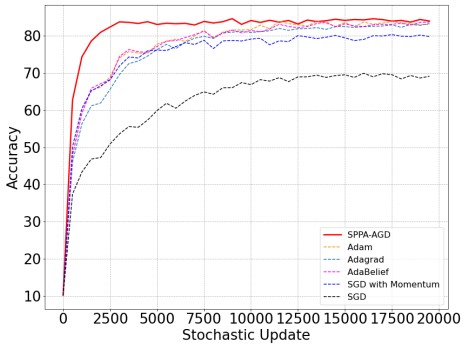

Figure 8: ResNet on CIFAR10: prediction accuracy, batch size 32

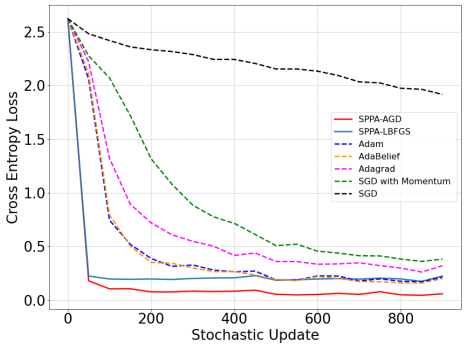

Figure 9: CNN on MNIST: cross entropy loss, batch size 64

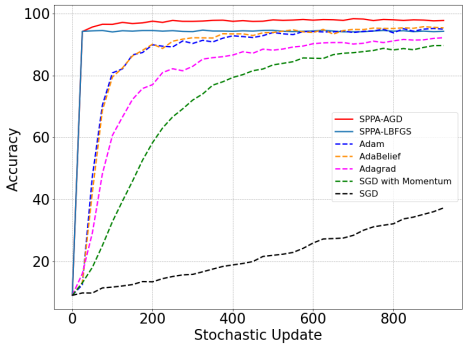

Figure 10: CNN on MNIST: prediction accuracy, batch size 64

outperform all SGD-based algorithms in terms of both the cross entropy loss and prediction accuracy on the test set.

4 Conclusion

In this paper we presented efficient implementations of the stochastic proximal point algorithm (SPPA) for general ERM problems and selected deep learning problems. We showed that for each of the ERM problem, one iteration of SPPA can be expressed in closed form or even as a simple optimization algorithm that can be easily solved by bisection method. We also proved that with our proposed efficient implementation, SPPA has same order of complexity as SGD. There do exist some convergence results of SPPA for convex problems different than ours, but our approach require only convexity of loss functions. The new convergence analysis also shows great resemblance to its deterministic counterpart, which is not the case from other works. The resulting algorithm is more robust in different stochastic settings and has cheap per-iteration complexity, while enjoying convergence under milder conditions. We demonstrated the performance of SPPA with our proposed efficient implementation on some well-known classification and regression data sets for ERM problems, and showed that they outperform many existing methods.

References

Zeyuan Allen-Zhu. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. In *STOC*, 2017.

Zeyuan Allen-Zhu. Katyusha X: Practical Momentum Method for Stochastic Sum-of-Nonconvex Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, ICML '18, 2018a.

Zeyuan Allen-Zhu. How To Make the Gradients Small Stochastically. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, NeurIPS '18, 2018b.

Nuno Antonio, Ana de Almeida, and Luis Nunes. Hotel booking demand datasets. *Data in brief*, 22: 41–49, 2019.

Hilal Asi and John C Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019.

Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical programming*, 129(2):163, 2011a.

Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010(1-38):3, 2011b.

Pascal Bianchi. Ergodic convergence of a stochastic proximal point algorithm. *SIAM Journal on Optimization*, 26(4):2235–2260, 2016.

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

Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Aaron Defazio. A simple practical accelerated method for finite sums. *Advances in neural information processing systems*, 29:676–684, 2016.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

John C Duchi and Feng Ruan. Stochastic methods for composite and weakly convex optimization problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pp. 1–15, 2013. ISSN 2192-6352. doi: 10.1007/s13748-013-0040-3. URL `[WebLink]`.

Simon Haykin. *Adaptive Filtering Theory*. Prentice Hall, 2002.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Feihu Huang, Songcan Chen, and Heng Huang. Faster stochastic alternating direction method of multipliers for nonconvex optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2839–2848. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/huang19a.html`.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5, 2010.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010.

Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pp. 2348–2358, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 80–88, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/ouyang13.html`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Andrei Pătraşcu. New nonasymptotic convergence rates of stochastic proximal point algorithm for stochastic convex optimization. *Optimization*, pp. 1–29, 2020.

David Pollard. *A user's guide to measure theoretic probability*. Number 8. Cambridge University Press, 2002.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Ernest K Ryu and Stephen Boyd. Stochastic proximal iteration: a non-asymptotic improvement upon stochastic gradient descent. *Preprint*, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Panos Toulis, Thibaut Horel, and Edoardo M Airoldi. The proximal Robbins–Monro method. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 83(1):188–212, 2021.

Huahua Wang and Arindam Banerjee. Online alternating direction method. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1699–1706, 2012.

Mengdi Wang and Dimitri P Bertsekas. Incremental constraint projection-proximal methods for nonsmooth convex optimization. *SIAM J. Optim.(to appear)*, 2013.

Wenliang Zhong and James Kwok. Fast stochastic alternating direction method of multipliers. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 46–54, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/zhong14.html`.

Juntang Zhuang, Tommy Tang, Sekhar Tatikonda, Nicha Dvornek, Yifan Ding, Xenophon Papademetris, and James S Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *arXiv preprint arXiv:2010.07468*, 2020.

A EFFICIENT IMPLEMENTATION

In this section we introduce several efficient methods to calculate the proximal operator update in Algorithm 1 line 3. At first glance it may seem as hard as solving the batch problem itself, but we will see that there are some interesting properties when f_i involve the loss evaluation of only one or a few data points in a data fitting scenario. On the other hand, it is not clear how to parallelize SPPA with mini-batches, which leaves room for future work.

A.1 LEAST SQUARES LOSS

For the least squares loss $f_i = (y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2$, the proximal update is a linear least squares problem with closed-form solution $\boldsymbol{w}_{t+1} = (\boldsymbol{I} + \lambda_t \boldsymbol{x}_i \boldsymbol{x}_i^\top)^{-1}(\boldsymbol{w}_t + \boldsymbol{x}_i y_i)$. It can be efficiently calculated by invoking the Sherman-Morrison formula and avoid directly inverting a $d \times d$ matrix for $\boldsymbol{w} \in \mathbb{R}^d$ as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\boldsymbol{x}_i^\top \boldsymbol{w}_t - y_i}{\|\boldsymbol{x}_i\|^2 + 1/\lambda} \boldsymbol{x}_i.$$

The update rule looks like, but is not exactly the same as, the normalized least mean squares (NLMS) algorithm (Haykin, 2002). It has appeared in (Pătraşcu, 2020), but we include it here for completeness.

A.2 LOGISTIC (CROSS-ENTROPY) LOSS

In general, the one-sample loss function can be written as $f_i(\boldsymbol{w}) = \boldsymbol{x}_i^\top \boldsymbol{w} = \ell(\boldsymbol{x}_i^\top \boldsymbol{w})$. For example, the one-sample loss for logistic regression is $f_i(\boldsymbol{w}) = \log(1 + \exp(-y_i \boldsymbol{x}_i^\top \boldsymbol{w}))$. Suppose it is differentiable, then by applying the chain rule, we have $\nabla f_i(\boldsymbol{w}) = \boldsymbol{x}_i \ell'(\boldsymbol{x}_i^\top \boldsymbol{w})$, where ℓ' is the derivative of the scalar function $\ell(\cdot)$. Plugging this into the optimality condition $\lambda_t \nabla f_i(\boldsymbol{w}_{t+1}) + \boldsymbol{w}_{t+1} - \boldsymbol{w}_t = 0$, we see that $\boldsymbol{w}_{t+1}$ has the form

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \boldsymbol{x}_i,$$

where α is some scalar. The proximal update reduces to finding the scalar α that solves the following convex problem,

$$\underset{\alpha}{\text{minimize}} \quad \ell(\boldsymbol{x}_i^\top \boldsymbol{w}_t - \alpha \|\boldsymbol{x}_i\|^2) + \frac{\alpha^2}{2\lambda_t} \|\boldsymbol{x}_i\|^2.$$

This can be done by setting the derivative equal to zero. Due to convexity, its derivative is a monotonic function, so the root can be obtained via bisection.

As a concrete example, consider the logistic loss $f_i(\boldsymbol{w}) = \log(1 + \exp(-y_i \boldsymbol{x}_i^T \boldsymbol{w}))$ with $y_i = \pm 1$. According to the aforementioned arguments, we need to solve the nonlinear equation

$$\alpha = y_i \lambda_t \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{w}_t - \alpha \|\boldsymbol{x}_i\|^2)}{1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{w}_t - \alpha \|\boldsymbol{x}_i\|^2)}.$$

Notice that the right-hand-side is a number between 0 and $\pm\lambda_t$, which gives the initial upper and lowerbound on α. Furthermore, we see that $\boldsymbol{x}_i^\top \boldsymbol{w}_t$ and $\|\boldsymbol{x}_i\|^2$ only need to be calculated once, with $\mathcal{O}(d)$ flops for $\boldsymbol{w} \in \mathbb{R}^d$; each bisection step takes constant time and it needs no more than $20 \sim 30$ scalar computations to render an accurate-enough solution.

A.3 HINGE LOSS

Regarding nonsmooth optimization problems, it turns out many of the widely used loss functions admit closed form updates. The main idea is to consider the subgradient calculus, and find the point where 0 is in the subdifferential. Take support vector machine (SVM) as an example, in which the loss function is the hinge loss $f_i(\boldsymbol{w}) = [1 - y_i \boldsymbol{x}_i^\top \boldsymbol{w}]_+$. Its subdifferential is

$$\partial f_i(\boldsymbol{w}) = \begin{cases} \{0\} & 1 - y_i \boldsymbol{x}_i^\top \boldsymbol{w} < 0, \\ \{-y_i \boldsymbol{x}_i\} & 1 - y_i \boldsymbol{x}_i^\top \boldsymbol{w} > 0, \\ \{-\alpha y_i \boldsymbol{x}_i | 0 \le \alpha \le 1\} & 1 - y_i \boldsymbol{x}_i^\top \boldsymbol{w} = 0. \end{cases}$$

When added with a proximal term, the subdifferential set is added with $\boldsymbol{w} - \boldsymbol{w}_t$. As a result, the update rule is simply

$$\boldsymbol{w}_{t+1} = \begin{cases} \boldsymbol{w}_t & y_i \boldsymbol{x}_i^\top \boldsymbol{w}_t > 1, \\ \boldsymbol{w}_t + \lambda y_i \boldsymbol{x}_i & y_i \boldsymbol{x}_i^\top \boldsymbol{w}_t < 1 - \lambda^{(t)} \|\boldsymbol{x}_i\|^2, \\ \boldsymbol{w}_t + \lambda y_i \boldsymbol{x}_i \frac{1 - y_i \boldsymbol{x}_i^\top \boldsymbol{w}_t}{y_i \|\boldsymbol{x}_i\|^2} & \text{otherwise.} \end{cases}$$

An interesting observation here is that it looks like the perceptron algorithm with an adaptively chosen step-size.

A.4 Absolute error loss

Robust regression using absolute error loss $f_i(\boldsymbol{w}) = |y_i - \boldsymbol{x}_i^\top \boldsymbol{w}|$, is another example to nonsmooth convex loss function. The derivation is similar to that of SVM: the subdifferential for the one-sample loss is

$$\partial f_i(\boldsymbol{w}) = \begin{cases} \{\boldsymbol{x}_i\} & y_i - \boldsymbol{x}_i^\top \boldsymbol{w} < 0, \\ \{-\boldsymbol{x}_i\} & y_i - \boldsymbol{x}_i^\top \boldsymbol{w} > 0, \\ \{\alpha \boldsymbol{x}_i | -1 \le \alpha \le 1\} & y_i - \boldsymbol{x}_i^\top \boldsymbol{w} = 0. \end{cases}$$

After adding the proximal term, the update rule is

$$\boldsymbol{w}_{t+1} = \begin{cases} \boldsymbol{w}_t - \lambda \boldsymbol{x}_i & y_i - \boldsymbol{x}_i^\top \boldsymbol{w}_t < -\lambda \|\boldsymbol{x}_i\|^2, \\ \boldsymbol{w}_t + \lambda \boldsymbol{x}_i & y_i - \boldsymbol{x}_i^\top \boldsymbol{w}_t > \lambda \|\boldsymbol{x}_i\|^2, \\ \boldsymbol{w}_t + \frac{y_i - \boldsymbol{x}_i^\top \boldsymbol{w}_t}{\|\boldsymbol{x}_i\|^2} \boldsymbol{x}_i & \text{otherwise.} \end{cases}$$

A.5 Generic losses

Even for generic nonlinear programming problems, it is still possible to efficiently evaluate the proximal update beyond merely a simple gradient step. The idea is to apply the limited-memory BFGS algorithm (Nocedal & Wright, 2006), or L-BFGS for short. L-BFGS is a memory-efficient implementation of the famous quasi-Newton algorithm BFGS. In a nut shell, L-BFGS is an iterative algorithm that makes use of the second-order information from the optimization loss function, but does not require solving matrix inverses and only requires explicitly evaluating first-order gradients. For a prescribed number of iterations, it requires one matrix-vector multiplication and multiple vector multiplications. As a result, the overall complexity is again $\mathcal{O}(d)$ if the initial guess of the Hessian matrix is diagonal.

While there are certain limitations for applying L-BFGS to general nonlinear programming problems, we reckon that it fits perfectly in the context of SPPA implementations.

- L-BFGS has to specify a good initial guess of the Hessian approximation matrix. While in many cases people simply use the identity matrix to start, it may result in very poor approximation. Fortunately, thanks to the proximal term, the identity matrix is in fact a very good initial guess for the Hessian matrix for SPPA updates.
- In order to save memory consumption, L-BFGS has to prescribe the number of iterations before running the algorithm. Obviouly, if the prescribed number of iteration is too large, we incur unnecessary computations, while if it is too small we need to invoke another round with a new estimated Hessian matrix. However, again in the context of SPPA, the proximal term naturally provides a good initialization $\boldsymbol{w}_t$. Our experience show that prescribing 10 iterations of L-BFGS updates is more than enough to obtain accurate solutions.

On the other hand, it perhaps makes more sense to simply use some more advanced first-order methods such as Nesterov's accelerated gradient descent (1983; 2013) to calculate the proximal update. Both L-BFGS and accelerated gradient descent evaluates the gradient of the loss function with $\mathcal{O}(d)$ complexity, and the question is how to leverage convergence rate versus sophistication.

B Proof of Lemma 1

At every iteration of SPPA, a component function f_i is picked uniformly at random to obtain the update. Therefore $p(i_t | \boldsymbol{w}_t) = 1/n$, obviously. Lemma 1, however, states that the conditional

probability of i_t given the next update $\boldsymbol{w}_{t+1}$ is still uniform. Recall that the update rule is

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \lambda_t f_{i_t}(\boldsymbol{w}) + (1/2)\|\boldsymbol{w} - \boldsymbol{w}_t\|^2. \tag{10}$$

What Lemma 1 implies is that if we know $\boldsymbol{w}_{t+1}$ is obtained at iteration t, without knowing the previous iterate $\boldsymbol{w}_t$, we gain no additional information regarding which component function f_i is more likely to have rendered the update equation 10.

Consider the sequence of indexes $i_1, i_2, \ldots$. At every iteration of SPPA, an index is independently drawn from $\{1, \ldots, n\}$ with equal probabilities. We may also treat this as a Markov chain with the transition probability $p(i_t|i_{t-1}) = 1/n$. This seems unnecessary at the moment, since an independent sequence is a trivial special case of a Markov chain. However, this implies $p(i_t|\boldsymbol{w}_t) = p(i_t|i_{t-1})$ and subsequently $p(i_t|\boldsymbol{w}_{t+1}) = p(i_t|i_{t+1})$. The question then boils down to studying the transition probability of the *reversed* Markov chain.

Kolmogorov's criterion gives a necessary and sufficient condition for a Markov chain to be identical to its time-reversed version (Kelly, 2011).

Theorem 3. *A stationary Markov chain is reversible if and only if its transition probability satisfies*

$$p(j_1|j_2)p(j_2|j_3)\cdots p(j_{m-1}|j_m)p(j_m|j_1) = p(j_1|j_m)p(j_2|j_1)\cdots p(j_m|j_{m-1})$$

for any finite sequence of states $j_1, \ldots, j_m$.

It is easy to see that in our case the Kolmogorov's criterion trivially holds since the transition probability is $1/n$ in all cases. This proves that $p(i_t|i_{t+1}) = p(i_t|i_{t-1}) = 1/n$ and thus $p(i_t|\boldsymbol{w}_{t+1}) = 1/n$.

C Almost Sure Convergence

In the main paper we presented the result that *the infimum of* $\mathbb{E}[F(\boldsymbol{w}_t)]$ goes to zero, following the techniques presented by Bottou et al. (2018). What this implies is that for some iteration s we may have a very small $\mathbb{E}[F(\boldsymbol{w}_s)]$, but for $t > s$ the expected loss may go up again. To make a stronger claim that the random variable $F(\boldsymbol{w}_t)$ converges *almost surely*, we are going to use martingales convergence results. The difference in this case is that we are going to use the *backwards* version of martingale convergence. Here we list the key results referenced from (Pollard, 2002).

Definition 1. Let $\{X_t : t \in \mathbb{N}\}$ be a sequence of integrable random variables, adapted to a decreasing filtration $\{\mathcal{G}_t : t \in \mathbb{N}\}$. We call $\{(X_t, \mathcal{G}_t) : t \in \mathbb{N}\}$ a *reversed supermartingale* if $\mathbb{E}[X_t|\mathcal{G}_{t+1}] \geq X_{t+1}$ for all t.

In other words, $\{(X_t, \mathcal{G}_t) : t \in \mathbb{N}\}$ is a reversed supermartingale if and only if $\{(X_{-t}, \mathcal{G}_{-t}) : t \in -\mathbb{N}\}$ is a supermartingale. In particular, for each fixed t, the finite sequence $X_t, X_{t-1}, \ldots, X_0$ is a supermartingale with respect to the filtration $\mathcal{G}_t \subseteq \mathcal{G}_{t-1} \subseteq \cdots \subseteq \mathcal{G}_0$.

The theory for reversed positive supermartingale is analogous to the regular supermartingale theory.

Theorem 4. *For every reversed, positive supermartingale* $\{(X_t, \mathcal{G}_t) : t \in \mathbb{N}\}$*, there exists an* X_∞ *for which* $X_t \to X_\infty$ *almost surely.*

Equipped with the reversed supermartingale theory, we have the following:

Theorem 5. *If all* $f_1, \ldots, f_n$ *are convex and* λ_t *is lowerbounded, then the sequence* $\{\boldsymbol{w}_t\}$ *generated by SPPA (Algorithm 1) satisfies that* $F(\boldsymbol{w}_t) \to F(\boldsymbol{w}_\star)$ *almost surely.*

Proof. Define the filtration $\mathcal{G}_t = \{\boldsymbol{w}_t, \boldsymbol{w}_{t+1}, \ldots\}$ for all $t = 0, 1, \ldots$. Consider inequality equation 2 from Proposition 1. Notice that

$$\mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 \mid \boldsymbol{w}_{t+1}] = \mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 \mid \mathcal{G}_{t+1}].$$

This means $\{(\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2, \mathcal{G}_t) : t \in \mathbb{N}\}$ is a reversed supermartingale. Applying Theorem 4, we have that $\{\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2\}$ converges almost surely; in other words

$$\left|\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2\right| \to 0$$

almost surely.

Notice that Proposition 1 also implies

$$\|\boldsymbol{w}_t - \boldsymbol{w}_\star\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_\star\|^2 \geq 2\lambda_t(F(\boldsymbol{w}_{t+1}) - F(\boldsymbol{w}_\star)).$$

Combining this with the previous result shows that

$$\lambda_t(F(\boldsymbol{w}_{t+1}) - F(\boldsymbol{w}_\star)) \to 0$$

almost surely. Since we assume that λ_t is lowerbounded for all t, this means $F(\boldsymbol{w}_{t+1}) \to F(\boldsymbol{w}_\star)$ almost surely. □

D Additional Experiments

D.1 Classification

In the main paper, we showed the performance of our unique implementation of SVM on Bank Note Authentication dataset and Logistic Regression on IMDB dataset. For the sake of completeness, we will show the performance of SPPA with our implementation for Logistic Regression on Bank Note Authentication dataset. Figure 11 shows the logistic loss value per second using Bank Note Authentication dataset. The performance of SPPA with our implementation is comparable with SGD with momentum. It still performs better than most of the other algorithms and it is more robust to change in seeds. Figure 12 shows that SPPA with our implementation reaches to the highest accuracy value on test data in less than a second, whereas the other algorithms take around 3 seconds to reach to an even smaller accuracy value. The robustness of SPPA with our implementation is consistent in this experiment too.

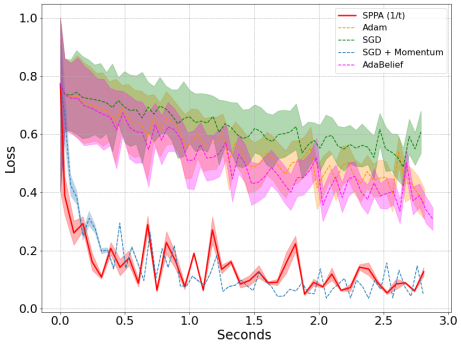

Figure 11: Logistic Regression on Bank Note Authentication: loss per seconds

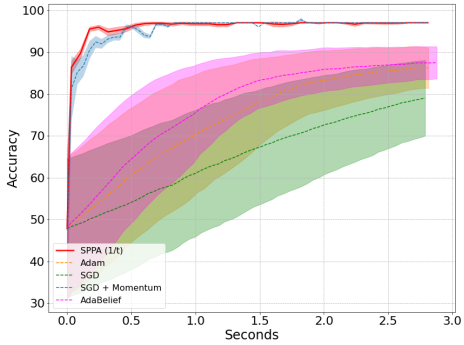

Figure 12: Logistic Regression on Bank Note Authentication: classification accuracy per seconds

D.1.1 Classification with L1 Regularization

We perform binary classification using SVM and logistic regression with L1 regularization. Our aim is to show the performance of our proposed SPPA algorithms in presence of L1 regularization term compared to other state of the art optimization algorithms using two different datasets, Bank Note Authentication and IMDB. In the results presented below, for each algorithm every stochastic update is followed by soft-thresholding to impose the effect of L1 regularization term. The inner loop in SPPA algorithm is run at most 20 times for the results presented below.

Bank Note Authentication Dataset. We perform classification using SVM and logistic regression with L1 regularization. Considering the per iteration complexity of SPPA, we plotted the loss-time, accuracy-time figures, Figure 13 and Figure 14 respectively. The Figure 13 demonstrates how the regularized hinge loss value decreases with respect to time (seconds) using different algorithms. The Figures 13 and 14 show that with our efficient implementation, SPPA takes less amount of time to reach to a smaller loss, and a greater accuracy despite of its per iteration complexity. The range shown in the plots clearly shows that SPPA is more robust to change in stochastic settings.

We present further experiments on the performance of SPPA using logistic regression with L1 regularization on Bank Note Authentication dataset loss-time, accuracy-time figures, Figure 15 and

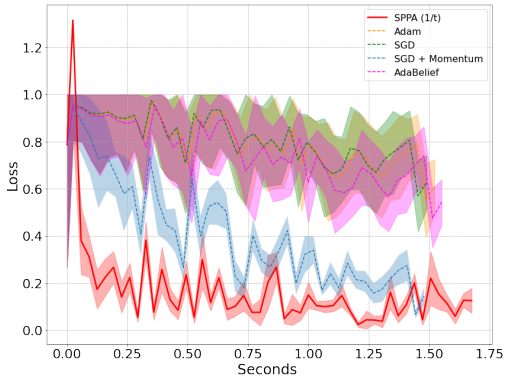

Figure 13: SVM with L1 Regularization on Bank Note Authentication: loss per seconds, batch size 1

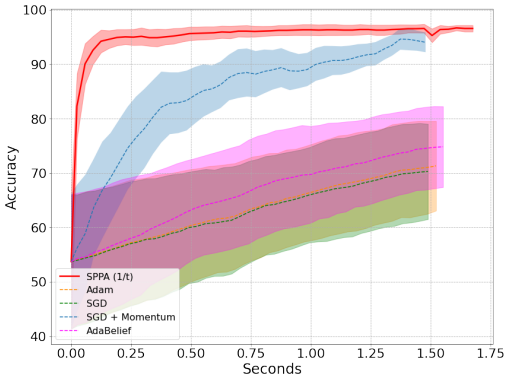

Figure 14: SVM with L1 Regularization on Bank Note Authentication: classification accuracy per seconds, batch size 1

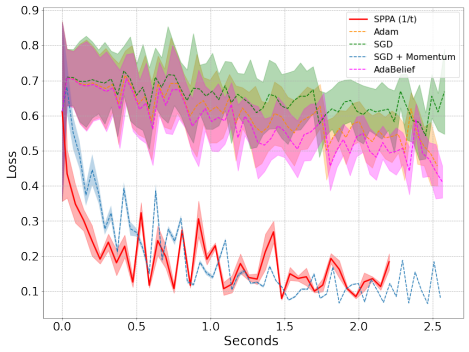

Figure 15: Logistic Regression with L1 Regularization on Bank Note Authentication: loss per seconds

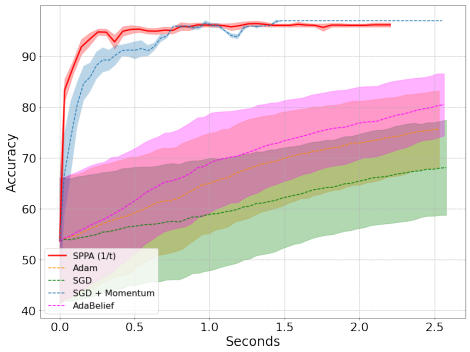

Figure 16: Logistic Regression with L1 Regularization on Bank Note Authentication: classification accuracy per seconds

Figure 16 respectively. In Figure 15 SPPA reaches to a smaller loss value in shorter amount of time compared to other algorithms, the closest performance is by Stochastic Gradient with Momentum algorithm. In Figure 16 SPPA reaches to the largest accuracy value in shortest amount of time and it has smallest variance compared to the other algorithms.

IMDB Dataset. We perform classification using SVM and logistic regression with L1 regularization on IMDB dataset. SPPA outperforms the other algorithms when we use logistic regression to do the classification. We aim to see the change in performances of the algorithms in presence of L1 regularization. According to the experiments presented there is not a big difference in the performance of algorithms with L1 regularization except for the slightly increase on the loss values. The Figure 17 shows the loss, and the Figure 18 shows the accuracy per seconds when logistic regression model is used with different optimization algorithms to perform classification on IMDB dataset.

D.2 Regression

Bike Sharing. Bike Sharing is a publicly available data set containing hourly or daily count of rental bikes as well as environmental and seasonal settings. The data set we are using is from UC Irvine Machine Learning Repository. [3] The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bike share system, Washington D.C., USA (Fanaee-T & Gama, 2013). It consists of 17379 samples (12512 training, 3476 test, 1391 validation)

[3] https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

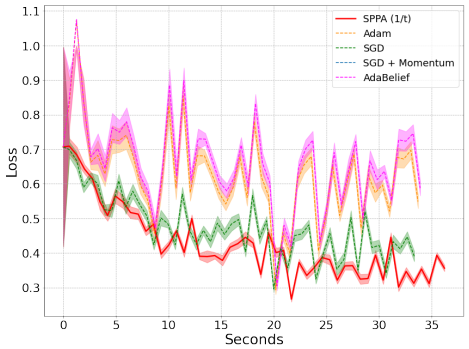

Figure 17: Logistic Regression with L1 Regularization on IMDB: loss per seconds

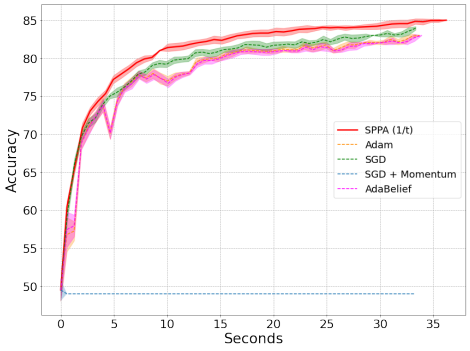

Figure 18: Logistic Regression with L1 Regularization on IMDB: classification accuracy per seconds

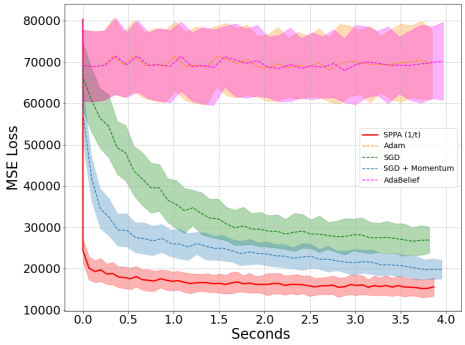

Figure 19: Linear Regression on Bike Sharing: mean squared error loss per seconds, batch size 16

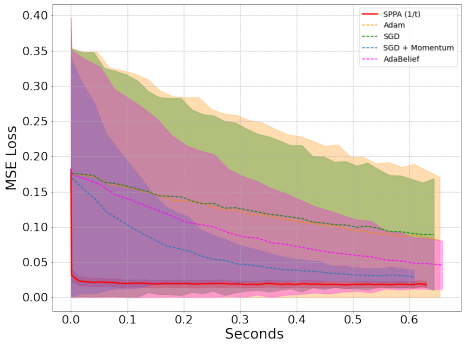

Figure 20: Linear Regression on Hotel ADR: mean squared error loss per seconds, batch size 1

with 16 features each to predict hourly count of rental bikes. We perform linear regression with two different loss functions, MSE loss. The Figure 19 shows mean squared error loss per seconds. We used 100 seeds to test the robustness of algorithms. SPPA reaches to a smaller loss value in less number of iterations. Batch size is 16.

Hotel Average Daily Rates (ADR). Hotel ADR is a publicly available data set consisting of hotel demand data for two different types of hotels; resort hotels and city hotels. Each hotel data set consists of 40,060 samples with 31 features each to predict average daily rate values (Antonio et al., 2019). In our experiments, we used resort hotels with 40,060 samples (30045 training, 10015 test) with 8 features. We perform linear regression with mean squared error (MSE). The Figure 20 shows mean squared error loss per seconds. SPPA reaches to a smaller loss value in less number of iterations and in less time. Batch size is 16. In Figure 20, MSE loss goes to very small value only in first few iterations. Our experiments show that overall performance of SPPA is better in linear regression problems using Hotel ADR dataset.

D.3 Regression with L1 Regularization

We perform linear regression with L1 regularization and show the results of different algorithms including our efficient implementation on both Bike Sharing [4] and Hotel Average Daily Rates (ADR) datasets (Antonio et al., 2019).

Bike Sharing. We perform linear regression with L1 regularization on Bike sharing dataset and compare the performances of different optimization algorithms. The Figure 21 shows the decrease in the loss values over time for different algorithms. SPPA outperforms the other algorithms and reaches to a smaller value in a shorter time. We tried different settings for the algorithms, in the results presented in Figure 21 the learning rate for SPPA is $1/sqrt(t)$ where t is the number of iterations.

[4] https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

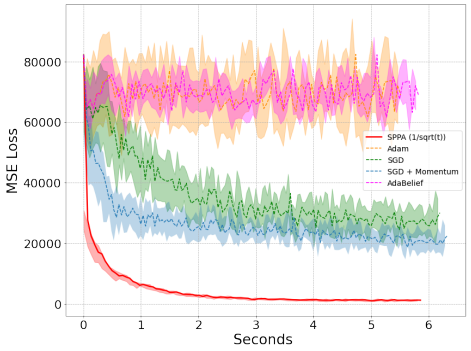

Figure 21: Linear Regression with L1 regularization on Bike Sharing: mean squared error loss per seconds, batch size 16

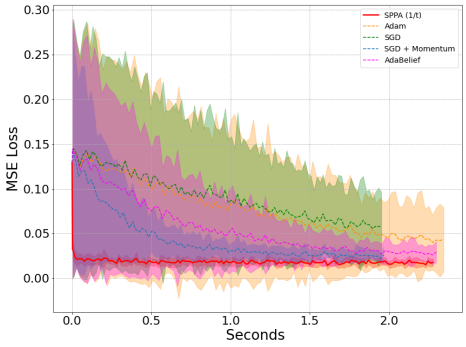

Figure 22: Linear Regression with L1 regularization on Hotel ADR: mean squared error loss per seconds, batch size 1

The other advantage, stability of SPPA is also showcased in Figure 21. Compared to the results without the L1 regularizer, the L1 regularizer helps SPPA to get smaller loss values.

Hotel Average Daily Rates (ADR). We perform linear regression with L1 regularization on Hotel ADR dataset and compare the performances of different optimization algorithms. The Figrure 22 shows the decrease in loss values over time for different algorithms. SPPA converges to a very small loss value much faster, in less than a second it converges to the minimum loss. We tried different settings for the algorithms, in the results presented in Figure 22 the learning rate for SPPA is $1/t$ where t is the number of iterations.

D.4 Mean-absolute-error Regression

We showcased the performance of our algorithm with mean squared error loss in regression problem with two real data-sets in the main paper. Here, we will present our results on the same data sets using absolute error loss. Figure 23 shows that, SPPA with our unique implementation reaches to a smaller loss value in less amount of time and significantly outperforms the alternative methods on Bike Sharing dataset. However, the error range of the algorithms are similar. Figure 24 shows the performance on Hotel ADR dataset with absolute error loss, our implementation of SPPA reaches to a smaller loss value in a significantly less amount of time, in the first few iterations. In addition to its great performance, looking at the error bar one can see that it is much more robust to different seed settings compared to other algorithms.

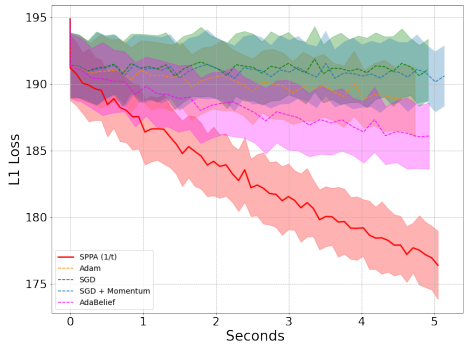

Figure 23: Linear Regression on Bike Sharing: absolute error loss per seconds, batch size 16

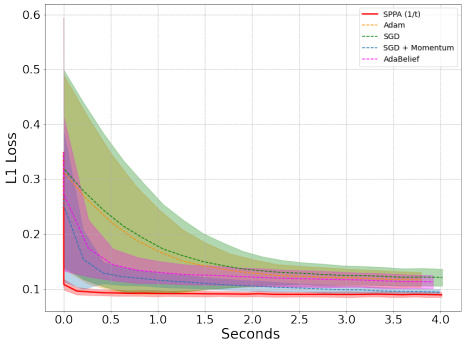

Figure 24: Linear Regression on Hotel ADR: absolute error loss per seconds, batch size 1