Target-based Surrogates for Stochastic Optimization

Jonathan Wilder Lavington[*1] **Sharan Vaswani**[*2] **Reza Babanezhad**[3] **Mark Schmidt**[1 4] **Nicolas Le Roux**[5 6]

Abstract

We consider minimizing functions for which it is expensive to compute the (possibly stochastic) gradient. Such functions are prevalent in reinforcement learning, imitation learning and adversarial training. Our target optimization framework uses the (expensive) gradient computation to construct surrogate functions in a *target space* (e.g. the logits output by a linear model for classification) that can be minimized efficiently. This allows for multiple parameter updates to the model, amortizing the cost of gradient computation. In the full-batch setting, we prove that our surrogate is a global upper-bound on the loss, and can be (locally) minimized using a black-box optimization algorithm. We prove that the resulting majorization-minimization algorithm ensures convergence to a stationary point of the loss. Next, we instantiate our framework in the stochastic setting and propose the `SSO` algorithm, which can be viewed as projected stochastic gradient descent in the target space. This connection enables us to prove theoretical guarantees for `SSO` when minimizing convex functions. Our framework allows the use of standard stochastic optimization algorithms to construct surrogates which can be minimized by any deterministic optimization method. To evaluate our framework, we consider a suite of supervised learning and imitation learning problems. Our experiments indicate the benefits of target optimization and the effectiveness of `SSO`.

[*]Equal contribution [1]University of British Columbia [2]Simon Fraser University [3]Samsung - SAIT AI Lab, Montreal [4] Canada CIFAR AI Chair (Amii) [5]Microsoft Research [6] Canada CIFAR AI Chair (MILA). Correspondence to: Jonathan Wilder Lavington <wilderlavington@gmail.com>, Sharan Vaswani <vaswani.sharan@gmail.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Stochastic gradient descent (SGD) (Robbins and Monro, 1951) and its variants (Duchi et al., 2011; Kingma and Ba, 2015) are ubiquitous optimization methods in machine learning (ML). For supervised learning, iterative first-order methods require computing the gradient over individual mini-batches of examples, and that cost of computing the gradient often dominates the total computational cost of these algorithms. For example, in reinforcement learning (RL) (Williams, 1992; Sutton et al., 2000) or online imitation learning (IL) (Ross et al., 2011), policy optimization requires gathering data via potentially expensive interactions with a real or simulated environment.

We focus on algorithms that access the expensive gradient oracle to construct a sequence of surrogate functions. Typically, these surrogates are chosen to be global upper-bounds on the underlying function and hence minimizing the surrogate allows for iterative minimization of the original function. Algorithmically, these surrogate functions can be minimized efficiently *without additional accesses to the gradient oracle*, making this technique advantageous for the applications of interest. The technique of incrementally constructing and minimizing surrogate functions is commonly referred to as *majorization-minimization* and includes the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) as an example. In RL, common algorithms (Schulman et al., 2015; 2017) also rely on minimizing surrogates.

Typically, surrogate functions are constructed by using the convexity and/or smoothness properties of the underlying function. Such surrogates have been used in the stochastic setting (Mairal, 2013; 2015). The prox-linear algorithm (Drusvyatskiy, 2017) for instance, uses the composition structure of the loss function and constructs surrogate functions in the parametric space. Unlike these existing works, we construct surrogate functions over a well-chosen *target space* rather than the parametric space, leveraging the *composition structure* of the loss functions prevalent in ML to build better surrogates. For example, in supervised learning, typical loss functions are of the form $h(\theta) = \ell(f(\theta))$, where ℓ is (usually) a convex loss (e.g. the squared loss for regression or the logistic loss for classification), while f corresponds to a transformation (e.g. linear or high-dimensional, non-convex as in the case of neural networks) of the inputs. Similarly, in IL, ℓ measures the divergence between the policy being learned and the ground-truth expert policy, whereas f corresponds to a specific parameterization of the policy being learned. More formally, if Θ is the feasible set of parameters, $f : \Theta \to \mathcal{Z}$ is a potentially non-convex mapping from the *parametric space* $\Theta \subseteq \mathbb{R}^d$ to the *target space* $\mathcal{Z} \subseteq \mathbb{R}^p$ and $\ell : \mathcal{Z} \to \mathbb{R}$ is a convex loss function. For example, for linear regression, $h(\theta) = \frac{1}{2} \left\| X\theta - y \right\|_2^2$, $z = f(\theta) = X\theta$ and $\ell(z) = \frac{1}{2} \left\| z - y \right\|_2^2$. In our applications of interest, computing $\nabla_z \ell(z)$ requires accessing the expensive gradient oracle, but $\nabla_\theta f(\theta)$ can be computed efficiently. Unlike Nguyen et al. (2022) who exploit this composition structure to prove global convergence, we will use it to construct surrogate functions in the target space. Johnson and Zhang (2020) also construct surrogate functions using the target space, but require access to the (stochastic) gradient oracle for each model update, making the algorithm proposed inefficient in our setting. Moreover, unlike our work, Johnson and Zhang (2020) do not have theoretical guarantees in the stochastic setting. Concurrently with our work, Woodworth et al. (2023) also consider minimizing expensive-to-evaluate functions, but do so by designing "proxy" loss function which is similar to the original function. We make the following contributions.

Target smoothness surrogate: In Section 3, we use the smoothness of ℓ with respect to z in order to define the *target smoothness surrogate* and prove that it is a global upper-bound on the underlying function h. In particular, these surrogates are constructed using tighter bounds on the original function. This ensures that additional progress can be made towards the minimizer using multiple model updates before recomputing the expensive gradient.

Target optimization in the deterministic setting: In Section 3, we devise a majorization-minimization algorithm where we iteratively form the target smoothness surrogate and then (locally) minimize it using any black-box algorithm. Although forming the target smoothness surrogate requires access to the expensive gradient oracle, it can be minimized without additional oracle calls resulting in multiple, computationally efficient updates to the model. We refer to this framework as *target optimization*. This idea of constructing surrogates in the target space has been recently explored in the context of designing efficient off-policy algorithms for reinforcement learning (Vaswani et al., 2021). However, unlike our work, Vaswani et al. (2021) do not consider the stochastic setting, or provide theoretical convergence guarantees. In Algorithm 1, we instantiate the target optimization framework and prove that it converges to a stationary point at a sublinear rate (Lemma C.1 in Appendix C).

Stochastic target smoothness surrogate: In Section 4, we consider the setting where we have access to an expensive stochastic gradient oracle that returns a noisy, but unbiased estimate of the true gradient. Similar to the deterministic setting, we access the gradient oracle to form a *stochastic* target smoothness surrogate. Though the surrogate is constructed by using a stochastic gradient in the target space, it is a deterministic function with respect to the parameters and can be minimized using any standard optimization algorithm. In this way, our framework disentangles the stochasticity in $\nabla_z \ell(z)$ (in the target space) from the potential non-convexity in f (in the parametric space).

Target optimization in the stochastic setting: Similar to the deterministic setting, we use m steps of GD to minimize the stochastic target smoothness surrogate and refer to the resulting algorithm as stochastic surrogate optimization (`SSO`). We then interpret `SSO` as inexact projected SGD in the target space. This interpretation of `SSO` allows the use of standard stochastic optimization algorithms to construct surrogates which can be minimized by any deterministic optimization method. Minimizing surrogate functions in the target space is also advantageous since it allows us to choose the space in which to constrain the size of the updates. Specifically, for overparameterized models such as deep neural networks, there is only a loose connection between the updates in the parameter and target space. In order to directly constrain the updates in the target space, methods such as natural gradient (Amari, 1998; Kakade, 2001) involve computationally expensive operations. In comparison, `SSO` has direct control over the updates in the target space and can also be implemented efficiently.

Theoretical results for `SSO`: Assuming $h(\theta)$ to be smooth, strongly-convex, in Section 4.1, we prove that `SSO` (with a constant step-size in the target space) converges linearly to a neighbourhood of the true minimizer (Theorem 4.2). In Proposition 4.3, we prove that the size of this neighbourhood depends on the noise (σ_z^2) in the stochastic gradients in the target space and an error term ζ^2 that depends on the dissimilarity in the stochastic gradients at the optimal solution. In Proposition 4.4, we provide a quadratic example that shows the necessity of this error term in general. However, as the size of the mini-batch increases or the model is over-parameterized enough to interpolate the data (Schmidt and Le Roux, 2013; Vaswani et al., 2019a), ζ_t^2 becomes smaller. In the special case when interpolation is exactly satisfied, we prove that `SSO` with $O\left(\log(1/\epsilon)\right)$ iterations is sufficient to guarantee convergence to an ϵ-neighbourhood of the true minimizer. Finally, we argue that `SSO` can be more efficient than using parametric SGD for expensive gradient oracles and common loss functions (Section 4.2).

Experimental evaluation: To evaluate our target optimization framework, we consider online imitation learning (OIL) as our primary example. For policy optimization in OIL, computing $\nabla_z \ell$ involves gathering data through interaction with a computationally expensive simulated environment. Using the Mujoco benchmark suite (Todorov et al., 2012) we demonstrate that SSO results in superior empirical performance (Section 5). We then consider standard supervised learning problems where we compare SSO with different choices of the target surrogate to standard optimization methods. These empirical results indicate the practical benefits of target optimization and SSO.

2. Problem Formulation

We focus on minimizing functions that have a composition structure and for which the gradient is expensive to compute. Formally, our objective is to solve the following problem: $\min_{\theta \in \Theta} h(\theta) := \ell(f(\theta))$ where $\Theta \subseteq \mathbb{R}^d$, $\mathcal{Z} \subseteq \mathbb{R}^p$, $f : \Theta \to \mathcal{Z}$ and $\ell : \mathcal{Z} \to \mathbb{R}$. Throughout this paper, we will assume that h is L_θ-smooth in the parameters θ and that $\ell(z)$ is L-smooth in the targets z. For all generalized linear models including linear and logistic regression, $f = X^\top \theta$ is a linear map in θ and ℓ is convex in z.

Example: For logistic regression with features $X \in \mathbb{R}^{n \times d}$ and labels $y \in \{-1, +1\}^n$, $h(\theta) = \sum_{i=1}^n \log\left(1 + \exp(-y_i \langle X_i, \theta \rangle)\right)$. If we "target" the logits[1], then $\mathcal{Z} = \{z | z = X\theta\} \subseteq \mathbb{R}^n$ and $\ell(z) = \sum_{i=1}^n \log\left(1 + \exp(-y_i z_i)\right)$. In this case, L_θ is the maximum eigenvalue of $X^\intercal X$, whereas $L = \frac{1}{4}$. A similar example follows for linear regression.

In settings that use neural networks, it is typical for the function f mapping X to y to be non-convex, while for the loss ℓ to be convex. For example in OIL, the target space is the space of parameterized policies, where ℓ is the cumulative loss when using a policy distribution $\pi := f(\theta)$ who's density is parameterized by θ. Though our algorithmic framework can handle non-convex ℓ and f, depending on the specific setting, our theoretical results will assume that ℓ (or h) is (strongly)-convex in θ and f is an affine map.

For our applications of interest, computing $\nabla_z \ell(z)$ is computationally expensive, whereas $f(\theta)$ (and its gradient) can be computed efficiently. For example, in OIL, computing the cumulative loss ℓ (and the corresponding gradient $\nabla_z \ell(z)$) for a policy involves evaluating it in the environment. Since this operation involves interactions with the environment or a simulator, it is computationally expensive. On the other hand, the cost of computing $\nabla_\theta f(\theta)$ only depends on the policy parameterization and does not involve additional interactions with the environment. In some cases, it is more natural to consider access to a *stochastic* gradient oracle that returns a noisy, unbiased gradient $\nabla \tilde{\ell}(z)$ such that $\mathbb{E}[\nabla \tilde{\ell}(z)] = \nabla \ell(z)$. We consider the effect of stochasticity in Section 4.

[1]The target space is not unique. For example, we could directly target the classification probabilities for logistic regression resulting in different f and ℓ.

If we do not take advantage of the composition structure nor explicitly consider the cost of the gradient oracle, iterative first-order methods such as GD or SGD can be directly used to minimize $h(\theta)$. At iteration $t \in [T]$, the *parametric GD* update is: $\theta_{t+1} = \theta_t - \eta \nabla h_t(\theta_t)$ where η is the step-size to be selected or tuned according to the properties of h. Since h is L_θ-smooth, each iteration of parametric GD can be viewed as exactly minimizing the quadratic surrogate function derived from the smoothness condition with respect to the parameters. Specifically, $\theta_{t+1} := \arg\min g_t^p(\theta)$ where g_t^p is the *parametric smoothness surrogate*: $g_t^p(\theta) := h(\theta_t) + \langle \nabla h(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta} \|\theta - \theta_t\|_2^2$.

The quadratic surrogate is tight at θ_t, i.e $h(\theta_t) = g_t^p(\theta_t)$ and becomes looser as we move away from θ_t. For $\eta \leq \frac{1}{L_\theta}$, the surrogate is a global (for all θ) upper-bound on h, i.e. $g_t^p(\theta) \geq h(\theta)$. Minimizing the global upper-bound results in descent on h since $h(\theta_{t+1}) \leq g_t^p(\theta_{t+1}) \leq g_t^p(\theta_t) = h_t(\theta_t)$. Similarly, the *parametric SGD* update consists of accessing the stochastic gradient oracle to obtain $\left(\tilde{h}(\theta), \nabla \tilde{h}(\theta)\right)$ such that $\mathbb{E}[\tilde{h}(\theta)] = h(\theta)$ and $\mathbb{E}[\nabla \tilde{h}(\theta)] = \nabla h(\theta)$, and iteratively constructing the *stochastic parametric smoothness surrogate* $\tilde{g^p}_t(\theta)$. Specifically, $\theta_{t+1} = \arg\min \tilde{g^p}_t(\theta)$, where $\tilde{g^p}_t(\theta) := \tilde{h}(\theta_t) + \langle \nabla \tilde{h}(\theta_t), \theta - \theta_t \rangle + \frac{1}{2\eta_t} \|\theta - \theta_t\|_2^2$. Here, η_t is the iteration dependent step-size, decayed according to properties of h (Robbins and Monro, 1951). In contrast to these methods, in the next section, we exploit the smoothness of the losses with respect to the target space and propose a majorization-minimization algorithm in the deterministic setting.

3. Deterministic Setting

We consider minimizing $\ell(f)$ in the deterministic setting where we can exactly evaluate the gradient $\nabla_z \ell(z)$. Similar to the parametric case in Section 2, we use the smoothness of $\ell(z)$ w.r.t the target space and define the *target smoothness surrogate* around z_t as: $\ell(z_t) + \langle \nabla_z \ell(z_t), z - z_t \rangle + \frac{1}{2\eta} \|z - z_t\|_2^2$, where η is the step-size in the target space and will be determined theoretically. Since $z = f(\theta)$, the surrogate can be expressed as a function of θ as: $g_t(\theta) := \left[\ell(z_t) + \langle \nabla_z \ell(z_t), f(\theta) - z_t \rangle + \frac{1}{2\eta} \|f(\theta) - z_t\|_2^2\right]$, which in general is not quadratic in θ.

Example: For linear regression, $f(\theta) = X^\top \theta$ and $g_t(\theta) = \frac{1}{2} \|X\theta_t - y\|_2^2 + \langle [X\theta_t - y], X(\theta - \theta_t) \rangle + \frac{1}{2\eta} \|X(\theta - \theta_t)\|_2^2$.

Similar to the parametric smoothness surrogate, we see that $h(\theta_t) = \ell(f(\theta_t)) = g_t(\theta_t)$. If ℓ is L-smooth w.r.t the target space $\mathcal{Z}$, then for $\eta \leq \frac{1}{L}$, we have $g_t(\theta) \geq \ell(f(\theta)) = h(\theta)$ for all θ, that is the surrogate is a global upper-bound on h. Since g_t is a global upper-bound on h, similar to GD, we can minimize h by minimizing the surrogate at each iteration i.e. $\theta_{t+1} = \arg\min_\theta g_t(\theta)$. However, unlike GD, in general, there is no closed form solution for the minimizer of g_t, and we will consider minimizing it approximately.

Algorithm 1 (Stochastic) Surrogate optimization

```
Input: $\theta_0$ (initialization), $T$ (number of iterations), $m_t$ (number of inner-loops), $\eta$ (step-size for the target space), $\alpha$ (step-size for the parametric space)
  for $t = 0$ to $T-1$ do
    Access the (stochastic) gradient oracle to construct $\tilde{g_t}(\theta)$
    Initialize inner-loop: $\omega_0 = \theta_t$
    for $k \leftarrow 0$ to $m_{t-1}$ do
      $\omega_{k+1} = \omega_k - \alpha \nabla_\omega \tilde{g_t}(\omega_k)$
    end for
    $\theta_{t+1} = \omega_m$   ;   $z_{t+1} = f(\theta_{t+1})$
  end for
  Return $\theta_T$
```

This results in the following meta-algorithm: for each $t \in [T]$, at iterate θ_t, form the surrogate g_t and compute θ_{t+1} by (approximately) minimizing $g_t(\theta)$. This meta-algorithm enables the use of any black-box algorithm to minimize the surrogate at each iteration. In Algorithm 1, we instantiate this meta-algorithm by minimizing $g_t(\theta)$ using $m \geq 1$ steps of gradient descent. For $m = 1$, Algorithm 1 results in the following update: $\theta_{t+1} = \theta_t - \alpha \nabla g_t(\theta_t) = \theta_t - \alpha \nabla h(\theta_t)$, and is thus equivalent to parametric GD with step-size α.

Example: For linear regression, instantiating Algorithm 1 with $m = 1$ recovers parametric GD on the least squares objective. On the other hand, minimizing the surrogate exactly (corresponding to $m = \infty$) to compute θ_{t+1} results in the following update: $\theta_{t+1} = \theta_t - \eta\,(X^\intercal X)^{-1}\,[X^\intercal(X\theta_t - y)]$ and recovers the Newton update in the parameter space. In this case, approximately minimizing g_t using $m \in (1, \infty)$ steps of GD interpolates between a first and second-order method in the parameter space. In this case, our framework is similar to quasi-Newton methods (Nocedal and Wright, 1999) that attempt to model the curvature in the loss without explicitly modelling the Hessian. Unlike these methods, our framework does not have an additional memory overhead.

In Lemma C.1 in Appendix C, we prove that Algorithm 1 with any value of $m \geq 1$ and appropriate choices of α and η results in an $O(1/T)$ convergence to a stationary point of h. Importantly, this result only relies on the smoothness of ℓ and g_t^z, and does not require either $\ell(z)$ or $f(\theta)$ to be convex. Hence, this result holds when using a non-convex model such as a deep neural networks, or for problems with non-convex loss functions such as in reinforcement learning. In the next section, we extend this framework to consider the stochastic setting where we can only obtain a noisy (though unbiased) estimate of the gradient.

4. Stochastic Setting

In the stochastic setting, we use the noisy but unbiased estimates $(\tilde{\ell}(z), \nabla\tilde{\ell}(z))$ from the gradient oracle to construct the *stochastic target surrogate*. To simplify the theoretical analysis, we will focus on the special case where ℓ is a finite-sum of losses i.e. $\ell(z) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(z)$. In this case, querying the stochastic gradient oracle at iteration t returns the individual loss and gradient corresponding to the loss index i_t i.e. $\left(\tilde{\ell}(z), \nabla\tilde{\ell}(z)\right) = (\ell_{i_t}(z), \nabla\ell_{i_t}(z))$. This structure is present in the use-cases of interest, for example, in supervised learning when using a dataset of n training points or in online imitation learning where multiple trajectories are collected using the policy at iteration t. In this setting, the deterministic surrogate is $g_t(\theta) := \frac{1}{n}\left[\sum\left[\ell_i(z) + \langle\nabla\ell_i(z_t), f(\theta) - z_t\rangle\right] + \frac{1}{2\eta_t}\left\|f(\theta) - z_t\right\|_2^2\right]$.

In order to admit an efficient implementation of the stochastic surrogate and the resulting algorithms, we only consider loss functions that are separable w.r.t the target space, i.e. for $z \in \mathcal{Z}$, if $z^i \in \mathbb{R}$ denotes coordinate i of z, then $\ell(z) = \frac{1}{n}\sum_i \ell_i(z^i)$. For example, this structure is present in the loss functions for all supervised learning problems where $\mathcal{Z} \subseteq \mathbb{R}^n$ and $z^i = f_i(\theta) := f(X_i, \theta)$. In this setting, $\frac{\partial \ell_i}{\partial z^j} = 0$ for all i and $j \neq i$ and the *stochastic target surrogate* is defined as $\tilde{g}_t(\theta) := \ell_{i_t}(z_t) + \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}\left[f_{i_t}(\theta) - z_t^{i_t}\right] + \frac{1}{2\eta_t}\left[f_{i_t}(\theta) - z_t^{i_t}\right]^2$, where η_t is the target space step-size at iteration t. Note that $\tilde{g}_t(\theta)$ only depends on i_t and only requires access to $\partial\ell_{i_t}(z_t)$ (meaning it can be constructed efficiently). We make two observations about the stochastic surrogate: (i) unlike the parametric stochastic smoothness surrogate $\tilde{g}^p$ that uses i_t to form the stochastic gradient, $\tilde{g}_t(\theta)$ uses i_t (the same random sample) for both the stochastic gradient $\partial\ell_{i_t}(z_t)$ and the regularization $\left[f_{i_t}(\theta) - z_t^{i_t}\right]^2$; (ii) while $\mathbb{E}_{i_t}[\tilde{g}_t(\theta)] = g_t(\theta)$, $\mathbb{E}_{i_t}[\arg\min \tilde{g}_t(\theta)] \neq \arg\min g_t(\theta)$, in contrast to the parametric case where $\mathbb{E}[\arg\min \tilde{g}_t^p(\theta)] = \arg\min g_t^p(\theta)$.

Example: For linear regression, $z^i = f_i(\theta) = X_i\theta$ and $\ell_i(z^i) = \frac{1}{2}(z^i - y_i)^2$. In this case, the stochastic target surrogate is equal to $\tilde{g}_t(\theta) = \frac{1}{2}(X_{i_t}\theta_t - y_{i_t})^2 + [X_{i_t}\theta_t - y_{i_t}] \cdot X_{i_t}(\theta - \theta_t) + \frac{1}{2\eta_t}\|X_{i_t}(\theta - \theta_t)\|_2^2$.

Similar to the deterministic setting, the next iterate can be obtained by (approximately) minimizing $\tilde{g}_t(\theta)$. Algorithmically, we can form the surrogate $\tilde{g}_t$ at iteration t and minimize it approximately by using any black-box algorithm. We refer to the resulting framework as *stochastic surrogate optimization* (SSO). For example, we can minimize $\tilde{g}_t$ using m steps of GD. The resulting algorithm is the same as Algorithm 1 but uses $\tilde{g}_t$ (the changes to the algorithm are highlighted in green). Note that the surrogate depends on the randomly sampled i_t and is therefore random. However, once the surrogate is formed, it can be minimized using any deterministic algorithm, i.e. there is no additional randomness in the inner-loop in Algorithm 1. Moreover, for the special case of $m = 1$, Algorithm 1 has the same update as parametric stochastic gradient descent. Previous work like the retrospective optimization framework in Newton et al. (2021) also considers multiple updates on the same batch of examples. However, unlike Algorithm 1, which forces proximity between consecutive iterates in the target space, Newton et al. (2021) use a specific stopping criterion in every iteration and consider a growing batch-size.

In order to prove theoretical guarantees for SSO, we interpret it as projected SGD in the target space. In particular, we prove the following equivalence in Appendix D.1.

Lemma 4.1. *The following updates are equivalent:*

$$(1)\quad \tilde{\theta}_{t+1} = \arg\min_{\theta} \tilde{g}_t(\theta); \qquad (\texttt{SSO})$$
$$\tilde{z}^{(1)}_{t+1} = f(\tilde{\theta}_{t+1})$$
$$(2)\quad z_{t+1/2} = z_t - \eta_t \nabla_z \ell_{i_t}(z_t)\,; \qquad \text{(Target-space SGD)}$$
$$\tilde{z}^{(2)}_{t+1} = \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\left\|z_{t+1/2} - z\right\|^2_{\mathcal{P}_t}$$

where, $\mathcal{P}_t \in \mathbb{R}^{p\times p}$ is a random diagonal matrix such that $\mathcal{P}_t(i_t, i_t) = 1$ and $\mathcal{P}_t(j,j) = 0$ for all $j \neq i_t$. That is, SSO (1) and target space SGD (2), result in the same iterate in each step i.e. if $z_t = f(\theta_t)$, then $\tilde{z}_{t+1} := \tilde{z}^{(1)}_{t+1} = \tilde{z}^{(2)}_{t+1}$.

The second step in target-space SGD corresponds to the projection (using randomly sampled index i_t) onto $\mathcal{Z}$[2].

Using this equivalence enables us to interpret the inexact minimization of $\tilde{g}_t$ (for example, using m steps of GD in Algorithm 1) as an inexact projection onto $\mathcal{Z}$ and will be helpful to prove convergence guarantees for SSO. The above interpretation also enables us to use the existing literature on SGD (Robbins and Monro, 1951; Li et al., 2021; Vaswani et al., 2019b) to specify the step-size sequence $\{\eta_t\}_{t=1}^T$ in the target space, completing the instantiation of the stochastic surrogate. Moreover, alternative stochastic optimization algorithms such as follow the regularized leader (Abernethy et al., 2009) and adaptive gradient methods like AdaGrad (Duchi et al., 2011), online Newton method (Hazan et al., 2007), and stochastic mirror descent (Bubeck et al., 2015, Chapter 6) in the target space result in different stochastic surrogates (refer to Appendix B) that can then be optimized using a black-box deterministic algorithm. Next, we prove convergence guarantees for SSO when $\ell(z)$ is a smooth, strongly-convex function.

[2]For linear parameterization, $z^{i_t} = \langle X_{i_t}, \theta\rangle$ and the set $\mathcal{Z}$ is convex. For non-convex f, the set $\mathcal{Z}$ can be non-convex and the projection is not well-defined. However, $\tilde{g}_t$ can still be minimized, albeit without any guarantees on the convergence.

4.1. Theoretical Results

For the setting where $\ell(z)$ is a smooth and strongly-convex function, we first analyze the convergence of inexact projected SGD in the target space (Section 4.1.1), and then bound the projection errors in in Section 4.1.2.

4.1.1. Convergence Analysis

For the theoretical analysis, we assume that the choice of f ensures that the projection is well-defined (e.g. for linear parameterization where $f = X^\intercal\theta$). We define $\bar{z}_{t+1} := f(\bar{\theta}_{t+1})$, where $\bar{\theta}_{t+1} := \arg\min \tilde{q}_t(\theta)$ and $\tilde{q}_t := \ell_{i_t}(z_t) + \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}\left[f_{i_t}(\theta) - z_t^{i_t}\right] + \frac{1}{2\eta'_t}\left\|f(\theta) - z_t\right\|_2^2$. In order to ensure that $\mathbb{E}[\tilde{q}_t(\theta)] = g_t(\theta)$, we will set $\eta'_t = \eta_t\, n$. Note that $\tilde{q}_t$ is similar to $\tilde{g}_t$, but there is no randomness in the regularization term. Analogous to $\tilde{z}_{t+1}$, $\bar{z}_{t+1}$ can be interpreted as a result of projected (where the projection is w.r.t ℓ_2-norm) SGD in the target space (Lemma D.1). Note that $\tilde{q}_t$ is only defined for the analysis of SSO.

For the theoretical analysis, it is convenient to define $\epsilon_{t+1} := \left\|z_{t+1} - \bar{z}_{t+1}\right\|_2$ as the projection error at iteration t. Here, $z_{t+1} = f(\theta_{t+1})$ where θ_{t+1} is obtained by (approximately) minimizing $\tilde{g}_t$. Note that the projection error incorporates the effect of both the random projection (that depends on i_t) as well as the inexact minimization of $\tilde{g}_t$, and will be bounded in Section 4.1.2. In the following lemma (proved in Appendix D.2), we use the equivalence in Lemma 4.1 to derive the following guarantee for two choices of η_t: (a) constant and (b) exponential step-size (Li et al., 2021; Vaswani et al., 2022).

Theorem 4.2. *Assuming that (i) ℓ_{i_t} is L-smooth and convex, (ii) ℓ is μ-strongly convex, (iii) $z^* = \arg\min_{z\in\mathcal{Z}} \ell(z)$ (iv) and that for all t, $\epsilon_t \leq \epsilon$, T iterations of SSO result in the following bound for $z_T = f(\theta_T)$,*

$$\mathbb{E}\left\|z_{T+1} - z^*\right\| \leq \left(\left(\prod_{i=1}^{T} \rho_i\right)\left\|z_1 - z^*\right\|_2^2 + 2\sigma^2 \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i {\eta'_t}^2\right)^{1/2} + 2\epsilon \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i\,,$$

where $\rho_t = (1 - \mu\eta'_t)$, $\sigma^2 := \mathbb{E}\left[\left\|\nabla\ell(z^*) - \nabla\ell_t(z^*)\right\|_2^2\right]$,
(a) ***Constant step-size****: When $\eta_t = \frac{1}{2L\,n}$ and for $\rho = 1 - \frac{\mu}{2L}$,*

$$\mathbb{E}\left\|z_{T+1} - z^*\right\| \leq \left\|z_1 - z^*\right\|\left(1 - \frac{1}{2\kappa}\right)^{\frac{T}{2}} + \frac{\sigma}{\sqrt{\mu L}} + \frac{2\epsilon}{1 - \sqrt{1 - \frac{1}{2\kappa}}}\,.$$

(b) ***Exponential step-size****: When $\eta_t = \frac{1}{2L\,n}\alpha^t$ for $\alpha =$*

$(\frac{\beta}{T})^{\frac{1}{T}}$,

$$\mathbb{E}\,\|z_{T+1} - z^*\| \leq c_1 \, \exp\left(-\frac{T}{4\kappa}\frac{\alpha}{\ln(T/\beta)}\right) \|z_1 - z^*\| + \frac{4\kappa c_1 (\ln(T/\beta))}{Le\alpha\sqrt{T}}\sigma + 2\epsilon\, c_2\,.$$

where $c_1 = \exp\left(\frac{1}{4\kappa}\,\frac{2\beta}{\ln(T/\beta)}\right)$, *and* $c_2 = \exp\left(\frac{\beta \ln(T)}{2\kappa \ln(T/\beta)}\right)$

In the above result, σ^2 is the natural analog of the noise in the unconstrained case. In the unconstrained case, $\nabla \ell(z^*) = 0$ and we recover the standard notion of noise used in SGD analyses (Bottou et al., 2018; Gower et al., 2019). Unlike parametric SGD, both σ^2 and κ do not depend on the specific model parameterization, and only depend on the properties of ℓ in the target space. For both the constant and exponential step-size, the above lemma generalizes the SGD proofs in Bottou et al. (2018) and Li et al. (2021); Vaswani et al. (2022) to projected SGD and can handle inexact projection errors similar to Schmidt et al. (2011). For constant step-size, SSO results in convergence to a neighbourhood of the minimizer where the neighbourhood depends on σ^2 and the projection errors ϵ_t^2. For the exponential step-sizes, SSO results in a noise-adaptive (Vaswani et al., 2022) $O\big(\exp\left(\frac{-T}{\kappa}\right) + \frac{\sigma^2}{T}\big)$ convergence to a neighbourhood of the solution which only depends on the projection errors. In the next section, we bound the projection errors.

4.1.2. Controlling the Projection Error

In order to complete the theoretical analysis of SSO, we need to control the projection error $\epsilon_{t+1} = \|\bar{z}_{t+1} - z_{t+1}\|_2$ that can be decomposed into two components as follows: $\epsilon_{t+1} \leq \|\bar{z}_{t+1} - \tilde{z}_{t+1}\|_2 + \|\tilde{z}_{t+1} - z_{t+1}\|_2$, where $\tilde{z}_{t+1} = f(\tilde{\theta}_{t+1})$ and $\tilde{\theta}_{t+1}$ is the minimizer of $\tilde{g}_t$. The first part of this decomposition arises because of using a stochastic projection that depends on i_t (in the definition of $\tilde{z}_{t+1}$) versus a deterministic projection (in the definition of $\bar{z}_{t+1}$). The second part of the decomposition arises because of the inexact minimization of the stochastic surrogate, and can be controlled by minimizing $\tilde{g}_t$ to the desired tolerance. In order to bound ϵ_{t+1}, we will make the additional assumption that f is L_f-Lipschitz in θ.[3] The following proposition (proved in Appendix D.3) bounds ϵ_{t+1}.

Proposition 4.3. *Assuming that (i) f is L_f-Lipschitz continuous, (ii) $\tilde{g}_t$ is μ_g strongly-convex and L_g smooth with $\kappa_g := L_g/\mu_g$[4], (iii) $\tilde{q}_t$ is μ_q strongly-convex (iv) $\zeta_t^2 := \frac{8}{\min\{\mu_g,\mu_q\}}\left([\min\{\mathbb{E}_{i_t}[\tilde{g}_t]\} - \mathbb{E}_{i_t}[\min\{\tilde{g}_t\}]]\right) + \left(\frac{8}{\min\{\mu_g,\mu_q\}}\,[\min\{\mathbb{E}_{i_t}[\tilde{q}_t]\} - \mathbb{E}_{i_t}[\min\{\tilde{q}_t\}]]\right)$, (v) $\sigma_z^2 := \min_{z\in\mathcal{Z}}\{\mathbb{E}_{i_t}[\ell_{i_t}(z)]\} - \mathbb{E}_{i_t}[\min_{z\in\mathcal{Z}}\{\ell_{i_t}(z)\}]$, if $\tilde{g}_t$ is minimized using m_t inner-loops of GD with the appropriate step-size, then, $\mathbb{E}[\epsilon_{t+1}^2] \leq L_f^2\,\zeta_t^2 + \frac{4L_f^2}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\left[\mathbb{E}[\ell(z_t) - \ell(z^*)] + \sigma_z^2\right]\right]$.*

[3]For example, when using a linear parameterization, $L_f = \|X\|$ This property is also satisfied for certain neural networks.

[4]We consider strongly-convex functions for simplicity, and it should be possible to extend these results to when $\tilde{g}$ is non-convex but satisfies the PL inequality, or is weakly convex.

Note that the definition of both ζ_t^2 and σ_z^2 is similar to that used in the SGD analysis in Vaswani et al. (2022), and can be interpreted as variance terms. In particular, using strong-convexity, $\min\{\mathbb{E}_{i_t}[\tilde{g}_t]\} - \mathbb{E}_{i_t}[\min\{\tilde{g}_t\}] \leq \frac{2}{\mu_g}\,\mathbb{E}[\left\|\nabla\tilde{g}_t(\theta'_{t+1}) - \mathbb{E}[\nabla\tilde{g}_t(\theta'_{t+1})]\right\|_2^2]$ which is the variance in $\nabla\tilde{g}_t(\theta'_{t+1})$. Similarly, we can interpret the other term in the definition of ζ_t^2. The first term $L_f^2\,\zeta_t^2$ because of the stochastic versus deterministic projection, whereas the second term $4L_f^2/\mu_g\,\exp\left(-m_t/\kappa_g\right)\,\left[\mathbb{E}[\ell(z_t) - \ell(z^*)] + \sigma_z^2\right]$ arises because of the inexact minimization of the stochastic surrogate. Setting $m_t = O(\log(1/\epsilon))$ can reduce the second term to $O(\epsilon)$. When sampling a mini-batch of examples in each iteration of SSO, both ζ_t^2 and σ_z^2 will decrease as the batch-size increases (because of the standard sampling-with-replacement bounds (Lohr, 2019)) becoming zero for the full-batch. Another regime of interest is when using over-parameterized models that can *interpolate* the data (Schmidt and Le Roux, 2013; Ma et al., 2018; Vaswani et al., 2019a). The interpolation condition implies that the stochastic gradients become zero at the optimal solution, and is satisfied for models such as non-parametric regression (Liang and Rakhlin, 2018; Belkin et al., 2019) and over-parametrized deep neural networks (Zhang et al., 2017). Under this condition, $\zeta_t^2 = \sigma_z^2 = 0$ (Vaswani et al., 2020; Loizou et al., 2021). From an algorithmic perspective, the above result implies that in cases where the noise dominates, using large m for SSO might not result in substantial improvements. However, when using a large batch-size or over-parameterized models, using large m can result in the superior performance.

Given the above result, a natural question is whether the dependence on ζ_t^2 is necessary in the general (non-interpolation) setting. To demonstrate this, we construct an example (details in Appendix D.4) and show that even for a sum of two one-dimensional quadratics, when minimizing the surrogate exactly i.e. $m = \infty$ and for any sequence of convergent step-sizes, SSO will converge to a neighborhood of the solution.

Proposition 4.4. *Consider minimizing the sum $h(\theta) := \frac{h_1(\theta)+h_2(\theta)}{2}$ of two one-dimensional quadratics, $h_1(\theta) := \frac{1}{2}(\theta - 1)^2$ and $h_2(\theta) = \frac{1}{2}\left(2\theta + 1/2\right)^2$, using SSO with $m_t = \infty$ and $\eta_t = c\,\alpha_t$ for any sequence of α_t and any constant $c \in (0, 1]$. SSO results in convergence to a neighbourhood of the solution, specifically, if θ^* is the minimizer of h and $\theta_1 > 0$, then, $\mathbb{E}(\theta_T - \theta^*) \geq \min\left(\theta_1, \frac{3}{8}\right)$.*

In order to show the above result, we use the fact that for quadratics, SSO (with $m = \infty$) is equivalent to the sub-sampled Newton method and in the one-dimensional case, we can recover the example in Vaswani et al. (2022). Since the above example holds for $m = \infty$, we conclude that this bias is not because of the inexact surrogate minimization. Moreover, since the example holds for all step-sizes including *any* decreasing step-size, we can conclude that the optimization error is not a side-effect of the stochasticity. In order to avoid such a bias term, sub-sampled Newton methods use different batches for computing the sub-sampled gradient and Hessian, and either use an increasing batch-size (Bollapragada et al., 2019) or consider using over-parameterized models (Meng et al., 2020).

Agarwal et al. (2020) also prove theoretical guarantees when doing multiple SGD steps on the same batch. In contrast to our work, their motivation is to analyze the performance of data-echoing (Choi et al., 2019). From a technical perspective, they consider (i) updates in the parameteric space, and (ii) their inner-loop step-size decreases as m increases. Finally, we note our framework and subsequent theoretical guarantees would also apply to this setting.

4.2. Benefits of Target Optimization

In order to gain intuition about the possible benefits of target optimization, let us consider the simple case where each h_i is μ_θ-strongly convex, L_θ-smooth and $\kappa_\theta = {}^{L_\theta}/_{\mu_\theta}$. For convenience, we define $\zeta^2 := \max_{t \in [T]} \zeta_t^2$. Below, we show that under certain regimes, for ill-conditioned least squares problems (where $\kappa_\theta >> 1$), target optimization has a provable advantage over parametric SGD.

Example: For the least squares setting, κ_θ is the condition number of the $X^\intercal X$ matrix. In order to achieve an ϵ sub-optimality, assuming complete knowledge of $\sigma_\theta^2 := \mathbb{E}\left\|\nabla h_i(\theta^*)\right\|_2^2$ and all problem-dependent constants, parametric SGD requires $T_{\text{param}} = O\big(\max\left\{\kappa_\theta \log\left({}^{1}/_{\epsilon}\right), {}^{\sigma_\theta^2}/_{\mu^2\epsilon}\right\}\big)$ iterations (Gower et al., 2019, Theorem 3.1) where the first term is the bias term and the second term is the effect of the noise. In order to achieve an ϵ sub-optimality for SSO, we require that $\zeta^2 \leq \epsilon$ (for example, by using a large enough batch-size) and $O(\kappa_\theta \log(1/\epsilon))$ inner iterations. Similar to the parametric case, the number of outer iterations for SSO is $T_{\text{target}} = O\big(\max\left\{\log\left({}^{1}/_{\epsilon}\right), {}^{\sigma^2}/_{\epsilon}\right\}\big)$, since the condition number w.r.t the targets is equal to 1.

In order to model the effect of an expensive gradient oracle, let us denote the cost of computing the gradient of ℓ as τ and the cost of computing the gradient of the surrogate as equal to 1. When the noise in the gradient is small and the bias dominates the number of iterations for both parametric and target optimization, the cost of parametric SGD is dominated by $\tau T_{\text{param}} = O(\kappa_\theta \tau)$, whereas the cost of target optimization is given by $T_{\text{target}} \times [\tau + \kappa_\theta \log(1/\epsilon)] = O(\tau + \kappa_\theta)$. Alternatively, when the noise dominates, we need to compare the $O\big(\tau {}^{\sigma_\theta^2}/_{\mu^2\epsilon}\big)$ cost for the parametric case against the $O\big(\frac{\sigma^2}{\epsilon}\left[\tau + \kappa_\theta \log(1/\epsilon)\right]\big)$ cost for target optimization. Using the definition of the noise, $\sigma_\theta^2 = \mathbb{E}_i \left\|X_i^\intercal (X_i\theta^* - y_i)\right\|_2^2 \leq L\sigma^2$. By replacing σ_θ^2 by $L\sigma^2$, we again see that the complexity of parametric SGD depends on $O(\kappa_\theta \tau)$, whereas that of SSO depends on $O(\kappa_\theta + \tau)$.

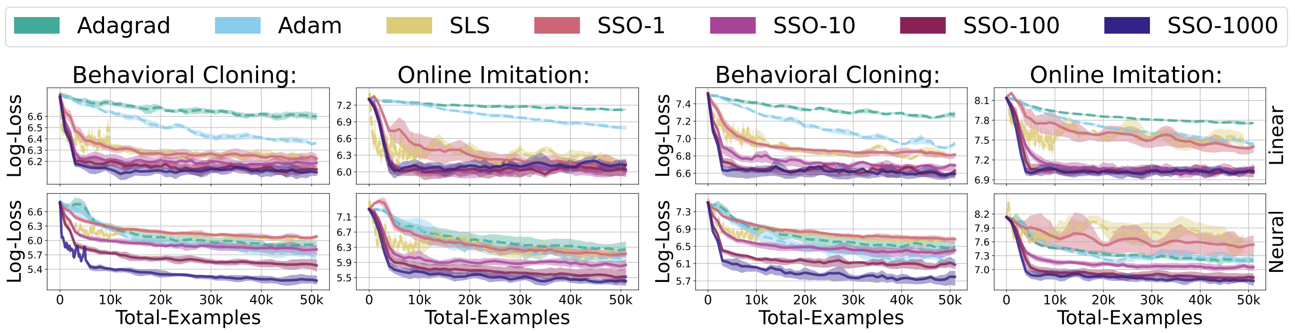
A similar property can also be shown for the logistic regression. Similarly, we could use other stochastic optimization algorithms to construct surrogates for target optimization. See Appendix B and Appendix E for additional discussion.

5. Experimental Evaluation

We evaluate the target optimization framework for online imitation learning and supervised learning[5]. In the subsequent experiments, we use either the theoretically chosen step-size when available, or the default step-size provided by Paszke et al. (2019). We do not include a decay schedule for any experiments in the main text, but consider both the standard ${}^{1}/\sqrt{t}$ schedule (Bubeck et al., 2015), as well as the exponential step-size-schedule (Vaswani et al., 2022; Orabona, 2019) in Appendix E. For SSO, since optimization of the surrogate is a deterministic problem, we use the standard back-tracking Armijo line-search (Armijo, 1966) with the same hyper-parameters across all experiments. For each experiment, we plot the average loss against the number of calls to the (stochastic) gradient oracle. The mean and the relevant quantiles are reported using three random seeds.

Online Imitation Learning: We consider a setting in which the losses are generated through interaction with a simulator. In this setting, a behavioral policy gathers examples by observing a state of the simulated environment and taking an action at that state. For each state gathered through the interaction, an expert policy provides the action that it would have taken. The goal in imitation learning is to produce a policy which imitates the expert. The loss measures the discrepancy between the learned policy π and the expert policy. In this case, $z = \pi$ where π is a distribution over actions given states, $\ell_{i_t}(z) = \mathbb{E}_{s_t}\left[\mathbf{KL}(\pi(\cdot|s_t)||\pi_{\text{expert}}(\cdot|s_t))\right]$ where the expectation is over the states visited by the behavioral policy and $f(\theta)$ is the policy parameterization. Since computing ℓ_{i_t} requires the behavioural policy to interact with the environment, it is expensive. Furthermore, the KL divergence is 1-strongly convex in the ℓ_1-norm, hence OIL satisfies all our assumptions. When the behavioral policy is the expert itself, we refer to the problem as *behavioral cloning*. When the learned policy is used to interact with the environment (Florence et al., 2022), we refer to the problem as *online imitation learning* (OIL) (Lavington et al., 2022; Ross et al., 2011).

[5]The code is available at `http://github.com/WilderLavington/Target-Based-Surrogates-For-Stochastic-Optimization`.



(a) Hopper-v2 Environment (b) Walker2d-v2 Environment

Figure 1: Comparison of log policy loss (mean-squared error between the expert labels and the mean action produced by the policy model) incurred by `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO` as a function of the total interactions (equal to t in Algorithm 1). `SSO-m` in the legend indicates that the surrogate has been minimized for m GD steps. The bottom row shows experiments where the policy is parameterized by a neural network, while the top row displays an example where the policy is linear model. Across all environments, behavioral policies, and model types, `SSO` outperforms all other online-optimization algorithms. Additionally, as m increases, so does the performance of `SSO`.

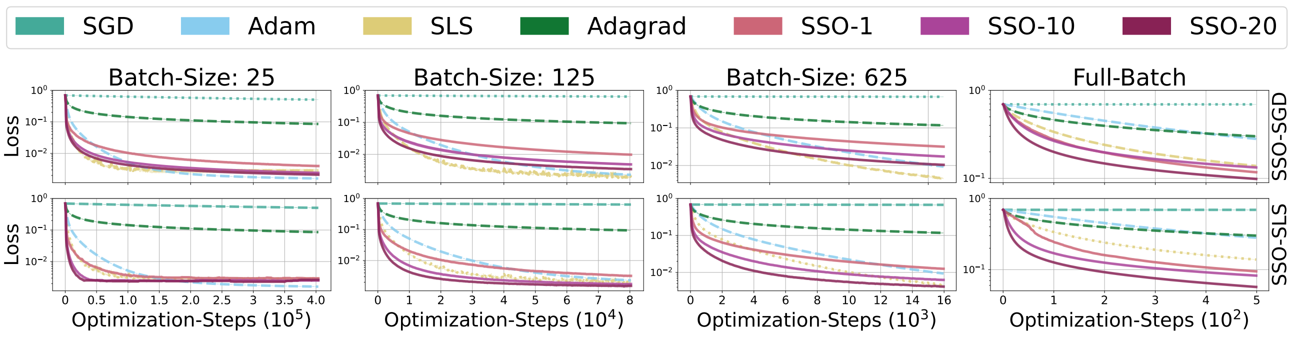


Figure 2: Comparison of `SGD` and its SSO variant (top row), `SLS` and it SSO variant (bottom row) over the rcv1 dataset (Chang and Lin, 2011) using a logistic loss. `Adam` and `Adagrad` are included as baselines. All plots are in log space, where the x-axis defines optimization steps (equal to t in Algorithm 1). We note that `SGD` with its theoretical step-size is outperformed by more sophisticated algorithms like `SLS` or `Adam`. In contrast, `SSO` with the theoretical step-size is competitive with both `SLS` and `Adam` with default hyper-parameters. Notably, the SSO variant of both SLS and SGD outperforms its parametric counterpart across both m and batch-size.

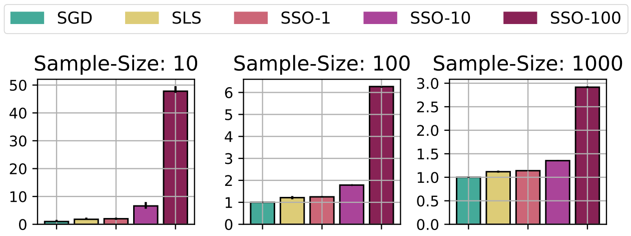


Figure 3: Comparison of run-times normalized with respect to the cost of a single SGD update) between SSO and relevant baselines. These plots illustrate that when data collection is expensive, SSO will be as fast as SGD, even for relatively large m. For Fig. 1, $\tau \approx 1000$, and the ratio of the time required to compute ℓ vs f is approximately 0.001.

In Fig. 1, we consider continuous control environments from the Mujoco benchmark suite (Todorov et al., 2012). The policy corresponds to a standard normal distribution whose mean is parameterized by either a linear function or a neural network. For gathering states, we sample from the stochastic (multivariate normal) policy in order to take actions. At each round of environment interaction 1000 states are gathered, and used to update the policy. The expert policy, defined by a normal distribution and parameterized by a two-layer MLP is trained using the Soft-Actor-Critic Algorithm (Haarnoja et al., 2018). Fig. 1 shows that `SSO` with the theoretical step-size drastically outperforms standard optimization algorithms in terms the log-loss as a function of environment interactions (calls to the gradient oracle). Further, we see that for both the linear and neural network parameterization, the performance of the learned policy consistently improves as m increases.

Runtime Comparison: In Fig. 3, we demonstrate the relative run-time between algorithms. Each column represents the average run-time required to take a single optimization step (sample states and update the model parameters) normalized by the time for SGD. We vary the number of states gathered (referred to as sample-size) per step and consider sample-sizes of 10, 100 and 1000. The comparison is performed on the `Hopper-v2` environment using a two layer MLP. We observe that for small sample-sizes, the time it takes to gather states does not dominate the time it takes to update the model (for example, in column 1, `SSO-100` takes almost 50 times longer than SGD). On the other hand, for large sample-sizes, the multiple model updates made by `SSO` are no longer a dominating factor (for example, see the right-most column where `SSO-100` only takes about three times as long as SGD but results in much better empirical performance). This experiment shows that in cases where data-access is the major bottleneck in computing the stochastic gradients, target optimization can be beneficial, matching the theoretical intuition developed in Section 4.2. For applications with more expensive simulators such as those for autonomous vehicle (Dosovitskiy et al., 2017), `SSO` can result in further improvements.

Supervised Learning: In order to explore using other optimization methods in the target optimization framework, we consider a simple supervised learning setup. In particular, we use the the `rcv1` dataset from libsvm (Chang and Lin, 2011) across four different batch sizes under a logistic-loss. We include additional experiments over other data-sets, and optimization algorithms in Appendix E.[6]

Extensions to the Stochastic Surrogate: We consider using a different optimization algorithm – stochastic line-search (Vaswani et al., 2019b) (`SLS`) in the target space, to construct a different surrogate. We refer to the resulting algorithm as `SSO-SLS`. For `SSO-SLS`, at every iteration, we perform a backtracking line-search in the target space to set η_t that satisfies the Armijo condition: $\ell_{i_t}(z_t - \eta_t \nabla_z \ell_{i_t}(z_t)) \leq \ell_{i_t}(z_t) - \frac{\eta_t}{2} \left\|\nabla_z \ell_{i_t}(z_t)\right\|_2^2$. We use the chosen η_t to instantiate the surrogate $\tilde{g}_t$ and follow Algorithm 1. In Fig. 2, we compare both `SSO-SGD` (top row) and `SSO-SLS` (bottom row) along with its parametric variant. We observe that the `SSO` variant of both SGD and SLS (i) does as well as its parametric counterpart across all m, and (ii), improves it for m sufficiently large.

[6]We also compared `SSO` against SVRG (Johnson and Zhang, 2013), a variance reduced method, and found that `SSO` consistently outperformed it across the batch-sizes and datasets we consider. For an example of this behavior see Fig. 15 in Appendix E.

6. Discussion

In the future, we aim to extend our theoretical results to a broader class of functions, and empirically evaluate target optimization for more complex models. Since our framework allows using any optimizer in the target space, we will explore other optimization algorithms in order to construct better surrogates. Another future direction is to construct better surrogate functions that take advantage of the additional structure in the model. For example, Taylor et al. (2016); Amid et al. (2022) exploit the composition structure in deep neural network models, and construct local layer-wise surrogates enabling massive parallelization. Finally, we also aim to extend our framework to applications such as RL where ℓ is non-convex.

7. Acknowledgements

We would like to thank Frederik Kunstner and Reza Asad for pointing out mistakes in the earlier version of this paper. This research was partially supported by the Canada CIFAR AI Program, the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants RGPIN-2022-03669 and RGPIN-2022-04816.

References

Abernethy, J. D., Hazan, E., and Rakhlin, A. (2009). Competing in the dark: An efficient algorithm for bandit linear optimization. *COLT*. (cited on 5)

Agarwal, N., Anil, R., Koren, T., Talwar, K., and Zhang, C. (2020). Stochastic optimization with laggard data pipelines. *Advances in Neural Information Processing Systems*, 33:10282–10293. (cited on 7)

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*. (cited on 2)

Amid, E., Anil, R., and Warmuth, M. (2022). Locoprop: Enhancing backprop via local loss optimization. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 9626–9642. PMLR. (cited on 9)

Armijo, L. (1966). Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3. (cited on 7)

Belkin, M., Rakhlin, A., and Tsybakov, A. B. (2019). Does data interpolation contradict statistical optimality? In *AISTATS*. (cited on 6)

Bollapragada, R., Byrd, R. H., and Nocedal, J. (2019). Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578. (cited on 7)

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311. (cited on 6)

Bubeck, S. et al. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357. (cited on 5, 7)

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27. (cited on 8, 9, 30, 36)

Choi, D., Passos, A., Shallue, C. J., and Dahl, G. E. (2019). Faster neural network training with data echoing. *arXiv preprint arXiv:1907.05550*. (cited on 7)

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22. (cited on 1)

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16. (cited on 9)

Drusvyatskiy, D. (2017). The proximal point method revisited. *arXiv preprint arXiv:1712.06038*. (cited on 1)

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*. (cited on 1, 5)

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In Faust, A., Hsu, D., and Neumann, G., editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 158–168. PMLR. (cited on 7)

Gower, R. M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., and Richtárik, P. (2019). Sgd: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR. (cited on 6, 7)

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR. (cited on 8)

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192. (cited on 5)

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems, NeurIPS*. (cited on 9, 36)

Johnson, R. and Zhang, T. (2020). Guided learning of nonconvex models through successive functional gradient optimization. In *International Conference on Machine Learning*, pages 4921–4930. PMLR. (cited on 2)

Kakade, S. M. (2001). A natural policy gradient. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. (cited on 2)

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*. (cited on 1, 30)

Lavington, J. W., Vaswani, S., and Schmidt, M. (2022). Improved policy optimization for online imitation learning. *arXiv preprint arXiv:2208.00088*. (cited on 7, 37)

Li, X., Zhuang, Z., and Orabona, F. (2021). A second look at exponential and cosine step sizes: Simplicity, adaptivity, and performance. In *International Conference on Machine Learning*, pages 6553–6564. PMLR. (cited on 5, 6)

Liang, T. and Rakhlin, A. (2018). Just interpolate: Kernel" ridgeless" regression can generalize. *arXiv preprint arXiv:1808.00387*. (cited on 6)

Lohr, S. L. (2019). *Sampling: Design and Analysis: Design and Analysis*. Chapman and Hall/CRC. (cited on 6)

Loizou, N., Vaswani, S., Laradji, I. H., and Lacoste-Julien, S. (2021). 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. (cited on 6)

Ma, S., Bassily, R., and Belkin, M. (2018). The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In *ICML*. (cited on 6)

Mairal, J. (2013). Stochastic majorization-minimization algorithms for large-scale optimization. *Advances in Neural Information Processing Systems*, 26. (cited on 1)

Mairal, J. (2015). Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855. (cited on 1)

Meng, S. Y., Vaswani, S., Laradji, I. H., Schmidt, M., and Lacoste-Julien, S. (2020). Fast and furious convergence: Stochastic second order methods under interpolation. In *International Conference on Artificial Intelligence and Statistics*, pages 1375–1386. PMLR. (cited on 7)

Nesterov, Y. (2003). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media. (cited on 22)

Newton, D., Bollapragada, R., Pasupathy, R., and Yip, N. K. (2021). Retrospective approximation for smooth stochastic optimization. *arXiv preprint arXiv:2103.04392*. (cited on 5)

Nguyen, L. M., Tran, T. H., and van Dijk, M. (2022). Finite-sum optimization: A new perspective for convergence to a global solution. *arXiv preprint arXiv:2202.03524*. (cited on 2)

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer. (cited on 4)

Orabona, F. (2019). A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*. (cited on 7, 30)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. (cited on 7, 33, 34, 35, 38)

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407. (cited on 1, 3, 5)

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of machine learning research*, pages 627–635. (cited on 1, 7)

Schmidt, M. and Le Roux, N. (2013). Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*. (cited on 2, 6)

Schmidt, M., Roux, N., and Bach, F. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. *Advances in neural information processing systems*, 24. (cited on 6, 28)

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897. (cited on 1)

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347. (cited on 1)

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1057–1063. (cited on 1, 38)

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. (2016). Training neural networks without gradients: A scalable admm approach. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2722–2731, New York, New York, USA. PMLR. (cited on 9)

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. (cited on 3, 8, 37)

Vaswani, S., Bach, F., and Schmidt, M. (2019a). Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR. (cited on 2, 6)

Vaswani, S., Bachem, O., Totaro, S., Müller, R., Garg, S., Geist, M., Machado, M. C., Castro, P. S., and Roux, N. L. (2021). A general class of surrogate functions for stable and efficient reinforcement learning. *arXiv preprint arXiv:2108.05828*. (cited on 2)

Vaswani, S., Dubois-Taine, B., and Babanezhad, R. (2022). Towards noise-adaptive, problem-adaptive (accelerated) stochastic gradient descent. In *International Conference on Machine Learning*, pages 22015–22059. PMLR. (cited on 5, 6, 7, 20, 25, 30)

Vaswani, S., Laradji, I., Kunstner, F., Meng, S. Y., Schmidt, M., and Lacoste-Julien, S. (2020). Adaptive gradient methods converge faster with over-parameterization (but you should do a line-search). *arXiv preprint arXiv:2006.06835*. (cited on 6)

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. (2019b). Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, pages 3727–3740. (cited on 5, 9, 30, 33, 34, 35, 38)

Ward, R., Wu, X., and Bottou, L. (2020). Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *The Journal of Machine Learning Research*, 21(1):9047–9076. (cited on 34)

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256. (cited on 1)

Woodworth, B., Mishchenko, K., and Bach, F. (2023). Two losses are better than one: Faster optimization using a cheaper proxy. *arXiv preprint arXiv:2302.03542*. (cited on 2)

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *ICLR*. (cited on 6)

Organization of the Appendix

A. Definitions

Our main assumptions are that each individual function f_i is differentiable, has a finite minimum f_i^*, and is L_i-smooth, meaning that for all v and w,

$$f_i(v) \leq f_i(w) + \langle \nabla f_i(w),\, v - w\rangle + \frac{L_i}{2}\left\|v - w\right\|_2^2 , \tag{Individual Smoothness}$$

which also implies that f is L-smooth, where L is the maximum smoothness constant of the individual functions. A consequence of smoothness is the following bound on the norm of the stochastic gradients,

$$\|\nabla f_i(w) - \nabla f_i^*\|^2 \leq 2L(f_i(w) - f_i^* - \langle \nabla f_i^*, w - w_i^*\rangle). \tag{1}$$

We also assume that each f_i is convex, meaning that for all v and w,

$$f_i(v) \geq f_i(w) + \langle \nabla f_i(w),\, v - w\rangle, \tag{Convexity}$$

Depending on the setting, we will also assume that f is μ strongly-convex, meaning that for all v and w,

$$f(v) \geq f(w) + \langle \nabla f(w),\, v - w\rangle + \frac{\mu}{2}\left\|v - w\right\|_2^2 , \tag{Strong Convexity}$$

B. Algorithms

In this section, we will formulate the algorithms beyond the standard SGD update in the target space. We will do so in two ways – (i) extending SGD to the online Newton step that uses second-order information in Appendix B.1 and (ii) extend SGD to the more general stochastic mirror descent algorithm in Appendix B.2. For both (i) and (ii), we will instantiate the resulting algorithms for the squared and logistic losses.

B.1. Online Newton Step

Let us consider the online Newton step w.r.t to the targets. The corresponding update is:

$$z_{t+1/2} = z_t - \eta_t [\nabla_z^2 \ell_t(z_t)]^{-1} \nabla_z \ell_t(z_t) \quad ; \quad \bar{z}_{t+1} = \arg\min_{z \in \mathcal{Z}} \frac{1}{2}\left\|z - z_{t+1/2}\right\|_{\mathcal{P}_t}^2 \tag{2}$$

$$z_{t+1} = f(\theta_{t+1}) \quad ; \quad \theta_{t+1} = \arg\min_{\theta} \left[\langle \nabla_z \ell_t(z_t), f(\theta) - z_t\rangle + \frac{1}{2\eta_t}\left\|f(\theta) - z_t\right\|_{\nabla^2 \ell_t(z_t)}^2\right] \tag{3}$$

where $\nabla_z^2 \ell_t(z_t)$ is the Hessian of example of the loss corresponding to sample i_t w.r.t z. Let us instantiate this update for the squared-loss. In this case, $\ell_t(z) = \frac{1}{2}\left\|z - y_t\right\|_2^2$, and hence, $\nabla \ell_t(z) = z - y_t$, $[\nabla^2 \ell_t(z)]_{i_t, i_t} = 1$ and $[\nabla^2 \ell_t(z)]_{j,j} = 0$ for all $j \neq i_t$. Hence, for the squared loss, Eq. (2) is the same as GD in the target space.

For the logistic loss, $\ell_t(z) = \log\left(1 + \exp\left(-y_t z\right)\right)$. If i_t is the loss index sampled at iteration t, then, $[\nabla \ell_t(z)]_j = 0$ for all $j \neq i_t$. Similarly, all entries of $\nabla^2 \ell_t(z)$ except the $[i_t, i_t]$ are zero.

$$[\nabla \ell_t(z)]_{i_t} = \frac{-y_t}{1 + \exp(y_t\, z_t)} \quad ; \quad [\nabla^2 \ell(z)]_{i_t, i_t}, = \frac{1}{1 + \exp(y_t z_t)}\, \frac{1}{1 + \exp(-y_t z_t)} = (1 - p_t)\, p_t \, ,$$

where, $p_t = \frac{1}{1+\exp(-y_t z_t)}$ is the probability of classifying the example i_t to have the $+1$ label. In this case, the surrogate can be written as:

$$\tilde{g}_t^z(\theta) = \frac{-y_t}{1+\exp(y_t\, z_t)} \left(f_{i_t}(\theta) - z_t^{i_t}\right) + \frac{(1-p_t)\, p_t}{2\eta_t} \left(f_{i_t}(\theta) - z_t^{i_t}\right)^2 \tag{4}$$

As before, the above surrogate can be implemented efficiently.

B.2. Stochastic Mirror Descent

If ϕ is a differentiable, strictly-convex mirror map, it induces a Bregman divergence between x and y: $D_\phi(y,x) := \phi(y) - \phi(x) - \langle \nabla\phi(x), y - x\rangle$. For an efficient implementation of stochastic mirror descent, we require the Bregman divergence to be separable, i.e. $D_\phi(y,x) = \sum_{j=1}^p D_{\phi_j}(y^j, x^j) = \sum_{j=1}^p \phi_j(y^j) - \phi_j(x^j) - \frac{\partial \phi_j(x)}{\partial x^j}[y^j - x^j]$. Such a separable structure is satisfied when ϕ is the Euclidean norm or negative entropy. We define stochastic mirror descent update in the target space as follows,

$$\nabla\phi(z_{t+1/2}) = \nabla\phi(z_t) - \eta_t \nabla_z \ell_t(z_t) \quad ; \quad \bar{z}_{t+1} = \operatorname*{arg\,min}_{z\in\mathcal{Z}} \sum_{j=1}^p \mathbb{I}(j = i_t) D_{\phi_j}(z^j, z^j_{t+1/2})) \tag{5}$$

$$\implies \bar{z}_{t+1} = \operatorname*{arg\,min}_{z\in\mathcal{Z}} \left[\langle \nabla_z \ell_t(z_t), z - z_t\rangle + \frac{1}{\eta_t} \sum_{j=1}^p \mathbb{I}(j = i_t) D_{\phi_j}(z^j, z^j_{t+1/2}) \right] \tag{6}$$

where $\mathbb{I}$ is an indicator function and i_t corresponds to the index of the sample chosen in iteration t. For the Euclidean mirror map, $\phi(z) = \frac{1}{2}\|z\|_2^2$, $D_\phi(z, z_t) = \frac{1}{2}\|z - z_t\|_2^2$ and we recover the SGD update.

Another common choice of the mirror map is the negative entropy function: $\phi(x) = \sum_{i=1}^K x^i \, \log(x^i)$ where x^i is coordinate i of the $x \in \mathbb{R}^K$. This induces the (generalized) KL divergence as the Bregman divergence,

$$D_\phi(x,y) = \sum_{k=1}^K x^k \, \log\left(\frac{x^k}{y^k}\right) + \sum_{k=1}^K x^k - \sum_{k=1}^K y^k \,.$$

If both x and y correspond to probability distributions i.e. $\sum_{k=1}^K x^k = \sum_{k=1}^K y^k = 1$, then the induced Bregman divergence corresponds to the standard KL-divergence between the two distributions. For multi-class classification, $\mathcal{Z} \subseteq \mathbb{R}^{p\times K}$ and each $z^i \in \Delta_K$ where Δ_K is K-dimensional simplex. We will refer to coordinate j of z^i as $[z^i]_j$. Since $z^i \in \Delta_K$, $[z^i]_k \geq 0$ and $\sum_{k=1}^K [z^i]_k = 1$.

Let us instantiate the general SMD updates in Eq. (5) when using the negative entropy mirror map. In this case, for $z \in \Delta_K$, $[\nabla\phi(z)]_k = 1 + \log([z]_k)$. Denoting $\nabla_t := \nabla_z \ell_t(z_t)$ and using $[\nabla_t]_k$ to refer to coordinate k of the K-dimensional vector ∇_t. Hence, Eq. (5) can be written as:

$$[z^{i_t}_{t+1/2}]_k = [z_t{}^{i_t}]_k \, \exp\left(-\eta_t [\nabla_t]_k\right) \tag{7}$$

For multi-class classification, $y^i \in \{0,1\}^K$ are one-hot vectors. If $[y^i]_k$ refers to coordinate k of vector y^i, then corresponding log-likelihood for n observations can be written as:

$$\ell(z) = \sum_{i=1}^n \sum_{k=1}^K [y^i]_k \, \log([z^i]_k)\,.$$

In our target optimization framework, the targets correspond to the probabilities of classifying the points into one of the classes. We use a parameterization to model the vector-valued function $f_i(\theta) : \mathbb{R}^d \to \mathbb{R}^K$. This ensures that for all i, $\sum_{k=1}^K [f_i(\theta)]_k = 1$. Hence, the projection step in Eq. (5) can be rewritten as:

$$\min_{z\in\mathcal{Z}} D_\phi(z, z_{t+1/2}) = \sum_{k=1}^K [f_{i_t}(\theta)]_k \, \log\left(\frac{[f_{i_t}(\theta)]_k}{[z^{i_t}_{t+1/2}]_k}\right)$$

where $[z^{i_t}_{t+1/2}]_k$ is computed according to Eq. (7). Since the computation of $[z^{i_t}_{t+1/2}]_k$ and the resulting projection only depends on sample i_t, it can be implemented efficiently.

C. Proofs in the Deterministic Setting

Lemma C.1. *Assuming that $g_t^z(\theta)$ is β-smooth w.r.t. the Euclidean norm and $\eta \leq \frac{1}{L}$, then, for $\alpha = {}^{1}\!/\!{}_{\beta}$, iteration t of Algorithm 1 guarantees that $h(\theta_{t+1}) \geq h(\theta_t)$ for any number $m \geq 1$ of surrogate steps. In this setting, under the additional assumption that h is lower-bounded by h^*, then Algorithm 1 results in the following guarantee,*

$$\min_{t \in \{0,\ldots,T-1\}} \|\nabla h(\theta_t)\|_2^2 \leq \frac{2\beta\,[h(\theta_0) - h^*]}{T}.$$

Proof. Using the update in Algorithm 1 with $\alpha = \frac{1}{\beta}$ and the β-smoothness of $g_t^z(\theta)$, for all $k \in [m-1]$,

$$g_t^z(\omega_{k+1}) \leq g_t^z(\omega_k) - \frac{1}{2\beta}\|\nabla g_t^z(\omega_k)\|_2^2$$

After m steps,

$$g_t^z(\omega_m) \leq g_t^z(\omega_0) - \frac{1}{2\beta}\sum_{k=0}^{m-1}\|\nabla g_t^z(\omega_k)\|_2^2$$

Since $\theta_{t+1} = \omega_m$ and $\omega_0 = \theta_t$ in Algorithm 1,

$$\implies g_t^z(\theta_{t+1}) \leq g_t^z(\theta_t) - \frac{1}{2\beta}\|\nabla g_t^z(\theta_t)\|_2^2 - \sum_{k=1}^{m-1}\|\nabla g_t^z(\omega_k)\|_2^2$$

Note that $h(\theta_t) = g_t^z(\theta_t)$ and if $\eta \leq \frac{1}{L}$, then $h(\theta_{t+1}) \leq g_t^z(\theta_{t+1})$. Using these relations,

$$h(\theta_{t+1}) \leq h(\theta_t) - \left[\underbrace{\frac{1}{2\beta}\|\nabla g_t^z(\theta_t)\|_2^2 + \sum_{k=1}^{m-1}\|\nabla g_t^z(\omega_k)\|_2^2}_{\geq 0}\right] \implies h(\theta_{t+1}) \leq h(\theta_t).$$

This proves the first part of the Lemma. Since $\sum_{k=1}^{m-1}\|\nabla g_t^z(\omega_k)\|_2^2 \geq 0$,

$$h(\theta_{t+1}) \leq h(\theta_t) - \frac{1}{2\beta}\|\nabla g_t^z(\theta_t)\|_2^2 \implies \|\nabla h(\theta_t)\|_2^2 \leq 2\beta\,[h(\theta_t) - h(\theta_{t+1})] \qquad \text{(Since } \nabla h(\theta_t) = \nabla g_t^z(\theta_t)\text{)}$$

Summing from $k = 0$ to $T-1$, and dividing by T,

$$\frac{\|\nabla h(\theta_t)\|_2^2}{T} \leq \frac{2\beta\,[h(\theta_t) - h^*]}{T} \implies \min_{t \in \{0,\ldots,T-1\}} \|\nabla h(\theta_t)\|_2^2 \leq \frac{2\beta\,[h(\theta_t) - h^*]}{T}$$

□

D. Proofs in the Stochastic Setting

D.1. Equivalence of SSO and SGD in Target Space

Lemma 4.1. *The following updates are equivalent:*

$$(1) \quad \tilde{\theta}_{t+1} = \arg\min_{\theta} \tilde{g}_t(\theta); \qquad (\texttt{SSO})$$
$$\tilde{z}^{(1)}_{t+1} = f(\tilde{\theta}_{t+1})$$
$$(2) \quad z_{t+1/2} = z_t - \eta_t \nabla_z \ell_{i_t}(z_t)\,; \qquad \text{(Target-space SGD)}$$
$$\tilde{z}^{(2)}_{t+1} = \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\left\|z_{t+1/2} - z\right\|^2_{\mathcal{P}_t}$$

where, $\mathcal{P}_t \in \mathbb{R}^{p\times p}$ is a random diagonal matrix such that $\mathcal{P}_t(i_t,i_t) = 1$ and $\mathcal{P}_t(j,j) = 0$ for all $j \neq i_t$. That is, SSO *(1) and target space SGD (2), result in the same iterate in each step i.e. if $z_t = f(\theta_t)$, then $\tilde{z}_{t+1} := \tilde{z}^{(1)}_{t+1} = \tilde{z}^{(2)}_{t+1}$.*

Proof. Since ℓ is separable, if i_t is the coordinate sampled from z, we can rewrite the target-space update as follows:

$$z_{t+1/2}{}^{i_t} = z_t{}^{i_t} - \eta_t \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}$$
$$z_{t+1/2}{}^{j} = z_t{}^{j} \quad \text{when } j \neq i_t\ .$$

Putting the above update in the projection step we have,

$$\begin{aligned}
\tilde{z}_{t+1} &= \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\{\left\|z_{t+1/2}{}^{i_t} - z^{i_t}\right\|_2^2\} \\
&= \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\{\left\|z_t{}^{i_t} - \eta_t \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}} - z^{i_t}\right\|_2^2\} \\
&= \arg\min_{z\in\mathcal{Z}} \left\{\left[\frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}\,[z^{i_t} - z_t^{i_t}] + \frac{1}{2\eta_t}\left\|z^{i_t} - z_t^{i_t}\right\|_2^2\right]\right\} && \text{(Due to separability of } \ell)
\end{aligned}$$

Since for all $z \in \mathcal{Z}$, $z = f(\theta)$ and $z^i = f_i(\theta)$ for all i. Hence $\tilde{z}_{t+1} = f(\tilde{\theta}_{t+1})$ such that,

$$\tilde{\theta}_{t+1} = \arg\min_{\theta\in\Theta} \left\{\frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}\,[f_{i_t}(\theta) - f_{i_t}(\theta_t)] + \frac{1}{2\eta_t}\left\|f_{i_t}(\theta) - f_{i_t}(\theta_t)\right\|_2^2\right\} = \arg\min_{\theta\in\Theta} \tilde{g}^z_t(\theta)$$

□

Lemma D.1. *Consider the following updates:*

$$\bar{\theta}_{t+1} = \arg\min_{\theta} \tilde{q}_t(\theta) \quad ; \quad \bar{z}^{(1)}_{t+1} = f(\bar{\theta}_{t+1}) \qquad (\texttt{SSO})$$
$$z_{t+1/2} = z_t - \eta'_t \nabla_z \ell_{i_t}(z_t)\,; \qquad \text{(Target-space SGD)}$$
$$\bar{z}^{(2)}_{t+1} = \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\left\|z_{t+1/2} - z\right\|^2_2$$

SSO *and target space SGD result in the same iterate in each step i.e. if $z_t = f(\theta_t)$, then $\bar{z}_{t+1} := \bar{z}^{(1)}_{t+1} = \bar{z}^{(2)}_{t+1}$.*

Proof.

$$\begin{aligned}
\bar{z}_{t+1} &= \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\{\left\|z_{t+1/2} - z\right\|_2^2\} \\
&= \arg\min_{z\in\mathcal{Z}} \frac{1}{2}\{\|z_t - \eta'_t \nabla_z \ell_{i_t}(z_t) - z\|_2^2\} \\
&= \arg\min_{z\in\mathcal{Z}} \left\{\left[\frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}}\,[z^{i_t} - z_t^{i_t}] + \frac{1}{2\eta'_t}\|z - z_t\|_2^2\right]\right\} && \text{(Due to separability of } \ell)
\end{aligned}$$

Since for all $z \in \mathcal{Z}$, $z = f(\theta)$. Hence $\bar{z}_{t+1} = f(\bar{\theta}_{t+1})$ such that,

$$\begin{aligned}\bar{\theta}_{t+1} &= \arg\min_{\theta \in \Theta} \left\{ \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}} \left[f_{i_t}(\theta) - f_{i_t}(\theta_t) \right] + \frac{1}{2\eta'_t} \left\| f(\theta) - f(\theta_t) \right\|_2^2 \right\} \\ &= \arg\min_{\theta \in \Theta} \tilde{q}_t(\theta)\end{aligned}$$

□

D.2. Proof for Strongly-convex Functions

We consider the case where $\ell(z)$ is strongly-convex and the set $\mathcal{Z}$ is convex. We will focus on SGD in the target space, and consider the following updates:

$$\begin{aligned}z_{t+1/2} &= z_t - \eta'_t \nabla \ell_t(z_t) \\ \bar{z}_{t+1} &= \Pi_{\mathcal{Z}}[z_{t+1/2}] := \arg\min_{z \in \mathcal{Z}} \frac{1}{2} \left\| z - z_{t+1/2} \right\|_2^2 \\ \|z_{t+1} - \bar{z}_{t+1}\| &\leq \epsilon_{t+1}\end{aligned}$$

Lemma D.2. *Bounding the suboptimality (to z^*) of z_{t+1} based on the sub-optimality of $\bar{z}_{t+1}$ and ϵ_{t+1}, we get that*

$$\|z_{t+1} - z^*\|_2^2 \leq \|\bar{z}_{t+1} - z^*\|_2^2 + 2\epsilon_{t+1} \|z_{t+1} - z^*\| \ . \tag{8}$$

Proof.

$$\begin{aligned}\|z_{t+1} - z^*\|_2^2 &= \|z_{t+1} - \bar{z}_{t+1} + \bar{z}_{t+1} - z^*\|_2^2 \\ &= \|z_{t+1} - \bar{z}_{t+1}\|_2^2 + \|\bar{z}_{t+1} - z^*\|_2^2 + 2\langle z_{t+1} - \bar{z}_{t+1}, \bar{z}_{t+1} - z^* \rangle \\ &= \|z_{t+1} - \bar{z}_{t+1}\|_2^2 + \|\bar{z}_{t+1} - z^*\|_2^2 + 2\langle z_{t+1} - \bar{z}_{t+1}, \bar{z}_{t+1} - z_{t+1} + z_{t+1} - z^* \rangle \\ &= \|z_{t+1} - \bar{z}_{t+1}\|_2^2 + \|\bar{z}_{t+1} - z^*\|_2^2 + 2\langle z_{t+1} - \bar{z}_{t+1}, z_{t+1} - z^* \rangle - 2\|z_{t+1} - \bar{z}_{t+1}\|_2^2 \\ &\leq \|\bar{z}_{t+1} - z^*\|_2^2 + 2\|z_{t+1} - \bar{z}_{t+1}\| \|z_{t+1} - z^*\| \\ &\leq \|\bar{z}_{t+1} - z^*\|_2^2 + 2\epsilon_{t+1} \|z_{t+1} - z^*\|\end{aligned}$$

□

Now we bound the exact sub-optimality at iteration $t+1$ by the inexact sub-optimality at iteration t to get a recursion.

Lemma D.3. *Assuming (i) each ℓ_t is L-smooth and (ii) ℓ is μ-strongly convex and (iii) $\eta'_t \leq \frac{1}{2L}$, we have*

$$\mathbb{E}\|\bar{z}_{t+1} - z^*\|_2^2 \leq (1 - \mu\eta'_t)\mathbb{E}\|z_t - z^*\|_2^2 + 2{\eta'_t}^2 \sigma^2 \tag{9}$$

where $\sigma^2 = \mathbb{E}\|\nabla \ell(z^*) - \nabla \ell_t(z^*)\|_2^2$.

Proof. Since $z^* \in \mathcal{Z}$ and optimal, $z^* = \Pi_{\mathcal{Z}}[z^* - \eta'_t \nabla \ell(z^*)]$.

$$\begin{aligned}\|\bar{z}_{t+1} - z^*\|_2^2 &= \|\Pi_{\mathcal{Z}}[z_t - \eta'_t \nabla \ell_t(z_t)] - \Pi_{\mathcal{Z}}[z^* - \eta'_t \nabla \ell(z^*)]\|_2^2 \\ &\leq \|[z_t - \eta'_t \nabla \ell_t(z_t)] - [z^* - \eta'_t \nabla \ell(z^*)]\|_2^2 \qquad \text{(Since projections are non-expansive)} \\ &= \|[z_t - \eta'_t \nabla \ell_t(z_t)] - [z^* - \eta'_t \nabla \ell_t(z^*)] + \eta'_t [\nabla \ell(z^*) - \nabla \ell_t(z^*)]\|_2^2 \\ &= \|z_t - z^*\|_2^2 + {\eta'_t}^2 \|\nabla \ell(z^*) - \nabla \ell_t(z^*)\|_2^2 + {\eta'_t}^2 \|\nabla \ell_t(z_t) - \nabla \ell_t(z^*)\|_2^2 + \underbrace{2\eta'_t \langle z_t - z^*, \nabla \ell(z^*) - \nabla \ell_t(z^*) \rangle}_{A_t} \\ &\quad - 2\eta'_t \langle z_t - z^*, \nabla \ell_t(z_t) - \nabla \ell_t(z^*) \rangle + 2{\eta'_t}^2 \underbrace{\langle \nabla \ell_t(z^*) - \nabla \ell(z^*), \nabla \ell_t(z_t) - \nabla \ell_t(z^*) \rangle}_{B_t} \\ &\leq \|z_t - z^*\|_2^2 + 2{\eta'_t}^2 \|\nabla \ell(z^*) - \nabla \ell_t(z^*)\|_2^2 + 2{\eta'_t}^2 \|\nabla \ell_t(z_t) - \nabla \ell_t(z^*)\|_2^2 \\ &\quad + A_t - 2\eta'_t \langle z_t - z^*, \nabla \ell_t(z_t) - \nabla \ell_t(z^*) \rangle \qquad \text{(Young inequality on } B_t\text{)}\end{aligned}$$

Taking expectation w.r.t i_t, knowing that $\mathbb{E} A_t = 0$ and using that $\sigma^2 = \mathbb{E}\left\|\nabla \ell(z^*) - \nabla \ell_t(z^*)\right\|_2^2$.

$$\begin{aligned}
\mathbb{E}\left\|\bar{z}_{t+1} - z^*\right\|_2^2 &\leq \mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2{\eta'_t}^2 \mathbb{E}\left\|\nabla \ell_t(z_t) - \nabla \ell_t(z^*)\right\|_2^2 - 2\eta'_t \langle z_t - z^*, \nabla \ell(z_t) - \nabla \ell(z^*)\rangle + 2{\eta'_t}^2 \sigma^2 \\
&\leq \mathbb{E}\left\|z_t - z^*\right\|_2^2 + 4{\eta'_t}^2 L\, \mathbb{E}\left\{\ell_t(z_t) - \ell_t(z^*) - \langle \nabla \ell_t(z^*), z_t - z^*\rangle\right\} - 2\eta'_t \langle z_t \\
&\qquad - z^*, \nabla \ell(z_t) - \nabla \ell(z^*)\rangle + 2\eta_t'^2 \sigma^2 \qquad (10) \\
&\leq \mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2\eta'_t\left\{\ell(z_t) - \ell(z^*) - \langle \nabla \ell(z^*), z_t - z^*\rangle\right\} - 2\eta'_t\langle z_t - z^*, \nabla \ell(z_t) - \nabla \ell(z^*)\rangle + 2\eta_t'^2\sigma^2 \\
&\qquad (\eta'_t < \tfrac{1}{2L}) \\
&\leq \mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2\eta'_t\left\{\ell(z_t) - \ell(z^*) - \langle \nabla \ell(z_t), z_t - z^*\rangle\right\} + 2\eta_t'^2\sigma^2 \\
&\leq \mathbb{E}\left\|z_t - z^*\right\|_2^2 - \mu\eta'_t \mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2\eta_t'^2\sigma^2 \qquad \text{(strong convexity of } \ell) \\
&\leq (1 - \mu\eta'_t)\mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2\eta_t'^2\sigma^2
\end{aligned}$$

where in Eq. (10) we use the smoothness of ℓ_t. □

Theorem 4.2. *Assuming that (i) ℓ_{i_t} is L-smooth and convex, (ii) ℓ is μ-strongly convex, (iii) $z^* = \arg\min_{z \in \mathcal{Z}} \ell(z)$ (iv) and that for all t, $\epsilon_t \leq \epsilon$, T iterations of SSO result in the following bound for $z_T = f(\theta_T)$,*

$$\begin{aligned}
\mathbb{E}\left\|z_{T+1} - z^*\right\| \leq &\left(\left(\prod_{i=1}^{T} \rho_i\right)\left\|z_1 - z^*\right\|_2^2 \right. \\
&\left. + 2\sigma^2 \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i {\eta'_t}^2\right)^{1/2} \\
&+ 2\epsilon \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i\,,
\end{aligned}$$

where $\rho_t = (1 - \mu\eta'_t)$, $\sigma^2 := \mathbb{E}\left[\left\|\nabla \ell(z^*) - \nabla \ell_t(z^*)\right\|_2^2\right]$,
*(a) **Constant step-size**: When $\eta_t = \frac{1}{2L\,n}$ and for $\rho = 1 - \frac{\mu}{2L}$,*

$$\begin{aligned}
\mathbb{E}\left\|z_{T+1} - z^*\right\| \leq &\left\|z_1 - z^*\right\|\left(1 - \frac{1}{2\kappa}\right)^{\frac{T}{2}} \\
&+ \frac{\sigma}{\sqrt{\mu L}} + \frac{2\epsilon}{1 - \sqrt{1 - \frac{1}{2\kappa}}}\,.
\end{aligned}$$

*(b) **Exponential step-size**: When $\eta_t = \frac{1}{2L\,n}\alpha^t$ for $\alpha = (\frac{\beta}{T})^{\frac{1}{T}}$,*

$$\begin{aligned}
\mathbb{E}\left\|z_{T+1} - z^*\right\| \leq &\; c_1 \exp\left(-\frac{T}{4\kappa}\frac{\alpha}{\ln(T/\beta)}\right)\left\|z_1 - z^*\right\| \\
&+ \frac{4\kappa c_1(\ln(T/\beta))}{Le\alpha\sqrt{T}}\sigma + 2\epsilon\, c_2\,.
\end{aligned}$$

where $c_1 = \exp\left(\frac{1}{4\kappa}\,\frac{2\beta}{\ln(T/\beta)}\right)$, and $c_2 = \exp\left(\frac{\beta\ln(T)}{2\kappa\ln(T/\beta)}\right)$

Proof. Using Lemma D.2 and Lemma D.3 we have

$$\begin{aligned}
\mathbb{E}\left\|z_{t+1} - z^*\right\|_2^2 &\leq \mathbb{E}\left\|\bar{z}_{t+1} - z^*\right\|_2^2 + 2\mathbb{E}[\epsilon_{t+1}\left\|z_{t+1} - z^*\right\|] \\
&\leq \underbrace{(1 - \mu\eta'_t)}_{\rho_t}\mathbb{E}\left\|z_t - z^*\right\|_2^2 + 2\eta_t'^2\sigma^2 + 2\mathbb{E}[\epsilon_{t+1}\left\|z_{t+1} - z^*\right\|]
\end{aligned}$$

Recursing from $t = 1$ to T,

$$\mathbb{E}[\left\|z_{T+1} - z^*\right\|_2^2] \leq \left(\prod_{t=1}^{T}\rho_t\right)\left\|z_1 - z^*\right\|_2^2 + 2\sigma^2\sum_{t=1}^{T}\prod_{i=t+1}^{T}\rho_i{\eta'_t}^2 + 2\sum_{t=1}^{T}\prod_{i=t+1}^{T}\rho_i\mathbb{E}[\epsilon_{t+1}\left\|z_{t+1} - z^*\right\|]$$

Denote $u_t := \mathbb{E}\|z_t - z^*\|$. By applying Jensen's inequality we know that $u_T^2 \leq \mathbb{E}[\|z_T - z^*\|_2^2]$.

$$\implies u_{T+1}^2 \leq \left(\prod_{t=1}^{T} \rho_t\right) u_1^2 + 2\sigma^2 \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i {\eta'_t}^2 + 2\epsilon \sum_{t=1}^{T} \prod_{i=t+1}^{T} \rho_i \, u_{t+1}$$

Dividing both sides in the previous inequality by $\prod_{i=1}^{T} \rho_i$ leads to:

$$\left(\prod_{i=1}^{T} \rho_i\right)^{-1} u_{T+1}^2 \leq u_1^2 + 2\sigma^2 \left(\prod_{i=1}^{T} \rho_i\right)^{-1} \sum_{t=1}^{T} {\eta'_t}^2 \left(\prod_{i=t+1}^{T} \rho_i\right) + 2\epsilon \left(\prod_{i=1}^{T} \rho_i\right)^{-1} \sum_{t=1}^{T} u_{t+1} \left(\prod_{i=t+1}^{T} \rho_i\right)$$

Simplify the above inequality for a generic $\tau \leq T$,

$$\left[\left(\prod_{i=1}^{\tau} \rho_i\right)^{-\frac{1}{2}} u_{\tau+1}\right]^2 \leq u_1^2 + 2\sigma^2 \sum_{t=1}^{\tau} {\eta'_t}^2 \left(\prod_{i=1}^{t} \rho_i\right)^{-1} + \sum_{t=1}^{\tau} 2\epsilon \left(\prod_{i=1}^{t} \rho_i\right)^{-\frac{1}{2}} \left[\left(\prod_{i=1}^{t} \rho_i\right)^{-\frac{1}{2}} u_{t+1}\right]$$

Let $v_\tau := \left(\prod_{i=1}^{\tau} \rho_i\right)^{-\frac{1}{2}} u_{\tau+1}$ and $S_\tau := u_1^2 + 2\sigma^2 \sum_{t=1}^{\tau} {\eta'_t}^2 \left(\prod_{i=1}^{t} \rho_i\right)^{-1}$. Let us also denote $\lambda_t := 2\epsilon \left(\prod_{i=1}^{t} \rho_i\right)^{-\frac{1}{2}}$. Observe that $S_0 = u_1^2 = v_0^2$ and $S_{\tau+1} = S_\tau + 2\sigma^2 {\eta'_{t\tau+1}}^2 \left(\prod_{i=1}^{\tau+1} \rho_i\right)^{-1}$. Therefore S_τ is an increasing sequence. Re-writing the previous inequality using the new variables leads to the following inequality:

$$v_\tau^2 \leq S_\tau + \sum_{t=1}^{\tau} \lambda_t v_t$$

Using the result from Lemma D.10 we have:

$$\begin{aligned} v_\tau &\leq \frac{1}{2} \sum_{t=1}^{\tau} \lambda_t + \left(S_\tau + \left(\frac{1}{2} \sum_{t=1}^{\tau} \lambda_t\right)^2\right)^{\frac{1}{2}} \\ &\leq \sum_{t=1}^{\tau} \lambda_t + \sqrt{S_\tau} && \text{(using } \sqrt{a+b} \leq \sqrt{a} + \sqrt{b} \text{ for } a, b \geq 0) \end{aligned}$$

Writing the inequality above using the original variables results in:

$$\left(\prod_{i=1}^{\tau} \rho_i\right)^{-\frac{1}{2}} u_{\tau+1} \leq \sum_{t=1}^{\tau} 2\epsilon \left(\prod_{i=1}^{t} \rho_i\right)^{-\frac{1}{2}} + \left(u_1^2 + 2\sigma^2 \sum_{t=1}^{\tau} {\eta'_t}^2 \left(\prod_{i=1}^{t} \rho_i\right)^{-1}\right)^{\frac{1}{2}}$$

$$u_{\tau+1} \leq 2\epsilon \sum_{t=1}^{\tau} \left(\prod_{i=1}^{t} \rho_i\right)^{-\frac{1}{2}} \left(\prod_{i=1}^{\tau} \rho_i\right)^{\frac{1}{2}} + u_1 \left(\prod_{i=1}^{\tau} \rho_i\right)^{\frac{1}{2}} + \sqrt{2}\sigma \left(\sum_{t=1}^{\tau} {\eta'_t}^2 \left(\prod_{i=1}^{t} \rho_i\right)^{-1}\right)^{\frac{1}{2}} \left(\prod_{i=1}^{\tau} \rho_i\right)^{\frac{1}{2}}$$

(a) **Constant step size**: Choosing a constant step size $\eta'_t = \eta = \frac{1}{2L}$ implies $\rho_i = \rho = 1 - \mu\eta = 1 - \frac{\mu}{2L}$. Plugging this into the previous inequality leads to:

$$\begin{aligned} u_{\tau+1} &\leq 2\epsilon \rho^{\frac{\tau}{2}} \sum_{t=1}^{\tau} \rho^{-\frac{t}{2}} + u_1 \rho^{\frac{\tau}{2}} + \sqrt{2}\sigma\eta \, \rho^{\frac{\tau}{2}} \sqrt{\sum_{t=1}^{\tau} \rho^{-t}} \\ &\leq \frac{2\epsilon}{1 - \sqrt{\rho}} + u_1 \rho^{\frac{\tau}{2}} + \frac{\sqrt{2}\sigma\eta}{\sqrt{1-\rho}} && \text{(applying the formula for finite geometric series)} \end{aligned}$$

$$\mathbb{E}\|z_{T+1} - z^*\| \leq \|z_1 - z^*\| \left(1 - \frac{1}{2\kappa}\right)^{\frac{T}{2}} + \frac{\sigma}{\sqrt{\mu L}} + \frac{2\epsilon}{1 - \sqrt{1 - \frac{1}{2\kappa}}}.$$

(b) **Exponential step size.**

Starting from Eq. (10) and using the proof from Vaswani et al. (2022), we have

$$\begin{aligned}
\mathbb{E}\left\|\bar{z}_{t+1}-z^*\right\|_2^2 &\leq \mathbb{E}\left\|z_t-z^*\right\|_2^2+4\eta_t'^2 L \mathbb{E}\left\{\ell_t(z_t)-\ell_t(z^*)-\langle\nabla \ell_t(z^*), z_t-z^*\rangle\right\}-2\eta_t'\langle z_t-z^*, \nabla \ell(z_t) \\
&\quad -\nabla \ell(z^*)\rangle+2\eta_t'^2 \sigma^2 \\
&\leq \mathbb{E}\left\|z_t-z^*\right\|_2^2+\frac{\alpha^{2t}}{L} \mathbb{E}\left\{\ell_t(z_t)-\ell_t(z^*)-\langle\nabla \ell_t(z^*), z_t-z^*\rangle\right\}-\frac{\alpha_t}{L}\langle z_t-z^*, \nabla \ell(z_t) \\
&\quad -\nabla \ell(z^*)\rangle+2\eta_t'^2 \sigma^2 && (\text{ using } \eta_t'=\tfrac{\alpha_t}{2L}) \\
&\leq \mathbb{E}\left\|z_t-z^*\right\|_2^2+\frac{\alpha_t}{L}\left\{\ell(z_t)-\ell(z^*)-\langle\nabla \ell(z^*), z_t-z^*\rangle\right\}-\frac{\alpha_t}{L}\langle z_t-z^*, \nabla \ell(z_t)-\nabla \ell(z^*)\rangle+2\eta_t'^2 \sigma^2 \\
& && (\text{ using } \alpha_t \leq 1) \\
&= \mathbb{E}\left\|z_t-z^*\right\|_2^2+\frac{\alpha_t}{L}\left\{\ell(z_t)-\ell(z^*)-\langle\nabla \ell(z_t), z_t-z^*\rangle\right\}+2\eta_t'^2 \sigma^2 \\
&\leq \mathbb{E}\left\|z_t-z^*\right\|_2^2-\frac{\mu\alpha_t}{2L} \mathbb{E}\left\|z_t-z^*\right\|_2^2+2\eta_t'^2 \sigma^2=\left(1-\frac{1}{2\kappa}\alpha_t\right) \mathbb{E}\left\|z_t-z^*\right\|_2^2+2\eta_t'^2 \sigma^2 \\
& && (\text{using strong convexity of } \ell)
\end{aligned}$$

Combining the above with Lemma D.2 we get:

$$\begin{aligned}
\mathbb{E}\left\|z_{t+1}-z^*\right\|_2^2 &\leq\left(1-\frac{1}{2\kappa}\alpha_t\right) \mathbb{E}\left\|z_t-z^*\right\|_2^2+2\eta_t'^2 \sigma^2+2\mathbb{E}[\epsilon_{t+1}\left\|z_{t+1}-z^*\right\|] \\
&\leq \exp\left(-\frac{1}{2\kappa}\alpha_t\right)\mathbb{E}\left\|z_t-z^*\right\|_2^2+\frac{\sigma^2}{2L^2}\alpha^{2t}+2\mathbb{E}[\epsilon_{t+1}\left\|z_{t+1}-z^*\right\|] && (1-x\leq\exp(-x)) \\
&\leq \exp\left(-\frac{1}{2\kappa}\alpha_t\right)\mathbb{E}\left\|z_t-z^*\right\|_2^2+\frac{\sigma^2}{2L^2}\alpha^{2t}+2\epsilon\mathbb{E}[\left\|z_{t+1}-z^*\right\|] && (\epsilon_{t+1}\leq\epsilon)
\end{aligned}$$

Unrolling the recursion starting from $t=1$ to T, denoting $u_t := \mathbb{E}\left\|z_t-z^*\right\|$ and applying Jensen's inequality to deduce that that $u_{T+1}^2 \leq \mathbb{E}\left\|z_{T+1}-z^*\right\|_2^2$, we get that,

$$u_{T+1}^2 \leq u_1^2 \exp\left(-\frac{1}{2\kappa}\sum_{t=1}^T \alpha_t\right)+\frac{\sigma^2}{2L^2}\sum_{t=1}^T \alpha^{2t}\exp\left(-\frac{1}{2\kappa}\sum_{i=t+1}^T \alpha_i\right)+2\epsilon\sum_{t=1}^T \exp\left(-\frac{1}{2\kappa}\sum_{i=t+1}^T \alpha_i\right)u_{t+1}$$

By multiplying both sides by $\exp\left(\frac{1}{2\kappa}\sum_{t=1}^T \alpha_t\right)$ we have:

$$\left(\exp\left(\frac{1}{4\kappa}\sum_{t=1}^T \alpha_t\right)u_{T+1}\right)^2 \leq u_1^2+\frac{\sigma^2}{2L^2}\sum_{t=1}^T \alpha^{2t}\exp\left(\frac{1}{2\kappa}\sum_{i=1}^t \alpha_i\right)+2\epsilon\sum_{t=1}^T \exp\left(\frac{1}{2\kappa}\sum_{i=1}^t \alpha_i\right)u_{t+1}.$$

Now let us define $v_\tau := \exp\left(\frac{1}{4\kappa}\sum_{t=1}^\tau \alpha_t\right)u_{\tau+1}$, $S_\tau := u_1^2+\frac{\sigma^2}{2L^2}\sum_{t=1}^\tau \alpha^{2t}\exp\left(\frac{1}{2\kappa}\sum_{i=1}^t \alpha_i\right)$ and $\lambda_t := 2\epsilon\exp\left(\frac{1}{4\kappa}\sum_{i=1}^t \alpha_i\right)$. Note that S_τ is increasing and $S_0=u_1^2=v_0^2\geq 0$ and $\lambda_t>0$, $v_\tau>0$. By applying Lemma D.10, similar to the fixed step-size case, for $\tau=T$, we get:

$$\exp\left(\frac{1}{4\kappa}\sum_{t=1}^T \alpha_t\right)u_{T+1} \leq \left(u_1^2+\frac{\sigma^2}{2L^2}\sum_{t=1}^T \alpha^{2t}\exp\left(\frac{1}{2\kappa}\sum_{i=1}^t \alpha_i\right)\right)^{1/2}+2\epsilon\sum_{t=1}^T \exp\left(\frac{1}{4\kappa}\sum_{i=1}^t \alpha_i\right)$$

Multiplying both sides by $\exp\left(-\frac{1}{4\kappa}\sum_{t=1}^{T}\alpha_t\right)$ gives us

$$u_{T+1} \leq \left(\exp\left(-\frac{1}{2\kappa}\underbrace{\sum_{t=1}^{T}\alpha_t}_{:=A}\right)u_1^2 + \frac{\sigma^2}{2L^2}\underbrace{\sum_{t=1}^{T}\alpha^{2t}\exp\left(\frac{1}{2\kappa}\sum_{i=t+1}^{T}\alpha_i\right)}_{:=B_T}\right)^{1/2} + 2\epsilon\underbrace{\sum_{t=1}^{T}\exp\left(-\frac{1}{4\kappa}\sum_{i=t+1}^{T}\alpha_i\right)}_{C_T}$$

$$= \left(u_1^2\exp\left(-\frac{1}{2\kappa}A\right) + \frac{\sigma^2}{2L^2}B_T\right)^{1/2} + 2\epsilon\, C_T$$

To bound A, we use Lemma D.6 and get

$$u_1^2\,\exp\left(-\frac{1}{2\kappa}A\right) \leq \|z_1 - z^*\|_2^2\underbrace{\exp\left(\frac{1}{2\kappa}\,\frac{2\beta}{\ln(T/\beta)}\right)}_{:=c_1^2}\exp\left(-\frac{T}{2\kappa}\frac{\alpha}{\ln(T/\beta)}\right)$$

To bound B_T we use Lemma D.7

$$B_T \leq \frac{16\kappa^2 c_1^2(\ln(T/\beta))^2}{e^2\alpha^2 T}$$

Finally using Lemma D.8 to bound C_T we get

$$\mathbb{E}\,\|z_{T+1} - z^*\| \leq \left(c_1^2\exp\left(-\frac{T}{2\kappa}\frac{\alpha}{\ln(T/\beta)}\right)\|z_1 - z^*\|_2^2 + \frac{16\kappa^2 c_1^2(\ln(T/\beta))^2}{2L^2e^2\alpha^2 T}\sigma^2\right)^{1/2} + 2\epsilon\underbrace{\exp\left(\frac{\beta\ln(T)}{2\kappa\ln(T/\beta)}\right)}_{c_2}$$

$$\implies \mathbb{E}\,\|z_{T+1} - z^*\| \leq c_1\,\exp\left(-\frac{T}{4\kappa}\frac{\alpha}{\ln(T/\beta)}\right)\|z_1 - z^*\| + \frac{4\kappa c_1(\ln(T/\beta))}{Le\alpha\sqrt{T}}\sigma + 2\epsilon\, c_2\,.$$

□

D.3. Controlling the Projection Error

Let us recall the following definitions for the theoretical analysis:

$$\tilde{\theta}_{t+1} := \arg\min_{\theta} \tilde{g}_t(\theta); \quad \tilde{g}_t(\theta) := \ell_{i_t}(z_t) + \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}} \left[f_{i_t}(\theta) - z_t^{i_t}\right] + \frac{1}{2\eta_t} \left[f_{i_t}(\theta) - z_t^{i_t}\right]^2 \quad ; \quad \tilde{z}_{t+1} = f(\theta_{t+1})$$

$$\bar{\theta}_{t+1} := \arg\min_{\theta} \tilde{q}_t(\theta) \quad ; \quad \tilde{q}_t(\theta) := \ell_{i_t}(z_t) + \frac{\partial \ell_{i_t}(z_t)}{\partial z^{i_t}} \left[f_{i_t}(\theta) - z_t^{i_t}\right] + \frac{1}{2\eta'_t} \left\|f(\theta) - z_t\right\|_2^2 \quad ; \quad \bar{z}_{t+1} = f(\bar{\theta}_{t+1})$$

$$\theta'_{t+1} := \arg\min_{\theta} g_t(\theta) \quad ; \quad g_t(\theta) := \frac{1}{n}\left[\left[\sum_i \ell_i(z_t) + \langle \nabla \ell_i(z_t), f(\theta) - z_t\rangle\right] + \frac{1}{2\eta_t}\left\|f(\theta) - z_t\right\|_2^2\right]$$

$$z_{t+1} = f(\theta_{t+1}),$$

where θ_{t+1} is obtained by running m_t iterations of GD on $\tilde{g}_t(\theta)$. We will use these definitions to prove the following proposition to control the projection error in each iteration.

Proposition 4.3. *Assuming that (i) f is L_f-Lipschitz continuous, (ii) $\tilde{g}_t$ is μ_g strongly-convex and L_g smooth with $\kappa_g := L_g/\mu_g$[7], (iii) $\tilde{q}_t$ is μ_q strongly-convex (iv) $\zeta_t^2 := \frac{8}{\min\{\mu_g, \mu_q\}} \left(\left[\min\left\{\mathbb{E}_{i_t}\left[\tilde{g}_t\right]\right\} - \mathbb{E}_{i_t}\left[\min\left\{\tilde{g}_t\right\}\right]\right]\right) + \left(\frac{8}{\min\{\mu_g, \mu_q\}} \left[\min\left\{\mathbb{E}_{i_t}\left[\tilde{q}_t\right]\right\} - \mathbb{E}_{i_t}\left[\min\left\{\tilde{q}_t\right\}\right]\right]\right)$, (v) $\sigma_z^2 := \min_{z \in \mathcal{Z}} \left\{\mathbb{E}_{i_t}\left[\ell_{i_t}(z)\right]\right\} - \mathbb{E}_{i_t}\left[\min_{z\in\mathcal{Z}}\left\{\ell_{i_t}(z)\right\}\right]$, if $\tilde{g}_t$ is minimized using m_t inner-loops of GD with the appropriate step-size, then, $\mathbb{E}[\epsilon_{t+1}^2] \leq L_f^2 \zeta_t^2 + \frac{4L_f^2}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\left[\mathbb{E}[\ell(z_t) - \ell(z^*)] + \sigma_z^2\right]\right]$.*

Proof. Since we obtain θ_{t+1} by minimizing $\tilde{g}_t(\theta)$ using m_t iterations of GD starting from θ_t, using the convergence guarantees of gradient descent (Nesterov, 2003),

$$\left\|\tilde{\theta}_{t+1} - \theta_{t+1}\right\|_2^2 \leq \exp\left(-m_t/\kappa_g\right)\left\|\tilde{\theta}_{t+1} - \theta_t\right\|_2^2$$

$$\left\|\theta_{t+1} - \bar{\theta}_{t+1}\right\|_2^2 = \left\|\theta_{t+1} - \tilde{\theta}_{t+1} + \tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2 \leq 2\left\|\theta_{t+1} - \tilde{\theta}_{t+1}\right\|_2^2 + 2\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$
$$(\|a+b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2)$$
$$\leq 2\exp\left(-m_t/\kappa_g\right)\left\|\tilde{\theta}_{t+1} - \theta_t\right\|_2^2 + 2\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$
$$\leq \frac{4}{\mu_g}\exp\left(-m_t/\kappa_g\right)\left[\tilde{g}_t(\theta_t) - \tilde{g}_t(\tilde{\theta}_{t+1})\right] + 2\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$

Taking expectation w.r.t i_t,

$$\mathbb{E}_{i_t}\left\|\theta_{t+1} - \bar{\theta}_{t+1}\right\|_2^2 \leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\mathbb{E}_{i_t}[\tilde{g}_t(\theta_t) - \tilde{g}_t(\tilde{\theta}_{t+1})]\right] + 2\mathbb{E}\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$

Let us first simplify $\mathbb{E}\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$.

$$\mathbb{E}\left\|\tilde{\theta}_{t+1} - \bar{\theta}_{t+1}\right\|_2^2 = \mathbb{E}\left\|\tilde{\theta}_{t+1} - \theta'_{t+1} + \theta'_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$
$$\leq 2\mathbb{E}\left\|\tilde{\theta}_{t+1} - \theta'_{t+1}\right\|_2^2 + 2\mathbb{E}\left\|\theta'_{t+1} - \bar{\theta}_{t+1}\right\|_2^2$$
$$\leq \frac{4}{\mu_g}\mathbb{E}[\tilde{g}_t(\theta'_{t+1}) - \tilde{g}_t(\tilde{\theta}_{t+1})] + \frac{4}{\mu_q}\mathbb{E}[\tilde{q}_t(\theta'_{t+1}) - \tilde{q}_t(\bar{\theta}_{t+1})]$$
$$= \frac{4}{\mu_g}\left[\min\left\{\mathbb{E}_{i_t}\left[\tilde{g}_t\right]\right\} - \mathbb{E}_{i_t}\left[\min\left\{\tilde{g}_t\right\}\right]\right] + \frac{4}{\mu_q}\left[\min\left\{\mathbb{E}_{i_t}\left[\tilde{q}_t\right]\right\} - \mathbb{E}_{i_t}\left[\min\left\{\tilde{q}_t\right\}\right]\right]$$
$$(\text{Since } \mathbb{E}[\tilde{q}] = \mathbb{E}[\tilde{g}] = g)$$

[7]We consider strongly-convex functions for simplicity, and it should be possible to extend these results to when $\tilde{g}$ is non-convex but satisfies the PL inequality, or is weakly convex.

$$2\mathbb{E}\left\|\tilde{\theta}_{t+1}-\bar{\theta}_{t+1}\right\|_2^2 \leq \underbrace{\frac{8}{\min\{\mu_g,\mu_q\}}\left([\min\{\mathbb{E}_{i_t}[\tilde{g}_t]\}-\mathbb{E}_{i_t}[\min\{\tilde{g}_t\}]]+[\min\{\mathbb{E}_{i_t}[\tilde{q}_t]\}-\mathbb{E}_{i_t}[\min\{\tilde{q}_t\}]]\right)}_{:=\zeta_t^2}$$

$$\implies 2\mathbb{E}\left\|\tilde{\theta}_{t+1}-\bar{\theta}_{t+1}\right\|_2^2 \leq \zeta_t^2$$

Using the above relation,

$$\begin{aligned}
\mathbb{E}_{i_t}\left\|\theta_{t+1}-\bar{\theta}_{t+1}\right\|_2^2 &\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\mathbb{E}_{i_t}[\tilde{g}_t(\theta_t)-\tilde{g}_t(\tilde{\theta}_{t+1})]\right]+\zeta_t^2\\
&\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\mathbb{E}_{i_t}[h_t(\theta_t)-h_t(\tilde{\theta}_{t+1})]\right]+\zeta_t^2\\
&\qquad\qquad\text{(Since } \tilde{g}_t(\theta_t)=h_t(\theta_t) \text{ and } \tilde{g}_t(\theta)\geq h_t(\theta) \text{ for all } \theta)\\
&= \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\mathbb{E}_{i_t}[h_t(\theta_t)-h_t^*+h_t^*-h_t(\tilde{\theta}_{t+1})]\right]+\zeta_t^2 \qquad (h_t^*:=\min_\theta h_t(\theta))\\
&\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\mathbb{E}_{i_t}[h_t(\theta_t)-h_t^*]+\zeta_t^2\right] \qquad \text{(Since } h_t^*\leq h_t(\theta) \text{ for all } \theta)\\
&= \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,[\mathbb{E}_{i_t}[h_t(\theta_t)-h_t(\theta^*)]+\mathbb{E}_{i_t}[h_t(\theta^*)-h_t^*]]\right]+\zeta_t^2\\
&= \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,[\mathbb{E}_{i_t}[\ell_t(z_t)-\ell_t(z^*)]+\mathbb{E}_{i_t}[\ell_t(z^*)-\ell_t^*]]\right]+\zeta_t^2\\
&\qquad\qquad\text{(Since } h(\theta)=\ell(f(\theta))=\ell(z))\\
\mathbb{E}_{i_t}[\left\|\theta_{t+1}-\bar{\theta}_{t+1}\right\|_2^2] &\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,[\mathbb{E}_{i_t}[\ell_t(z_t)-\ell_t(z^*)]+\mathbb{E}_{i_t}[\ell_t(z^*)-\ell_t^*]]\right]+\zeta_t^2\\
&\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\left[[\ell(z_t)-\ell(z^*)]+\underbrace{\mathbb{E}_{i_t}[\ell_t(z^*)-\ell_t^*]}_{:=\sigma_z^2}\right]\right]+\zeta_t^2\\
&\qquad\text{(Since both } z_t \text{ and } z^* \text{ are independent of the randomness in } \ell_t \text{ and } \mathbb{E}_{i_t}[\ell_t]=\ell)\\
\implies \mathbb{E}[\left\|\theta_{t+1}-\bar{\theta}_{t+1}\right\|_2^2] &\leq \frac{4}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\left[\ell(z_t)-\ell(z^*)+\sigma_z^2\right]\right]+\zeta_t^2
\end{aligned}$$

Now, we will bound $\mathbb{E}[\epsilon_{t+1}^2]$ by using the above inequality and the Lipschitzness of f.

$$\mathbb{E}[\epsilon_{t+1}^2]=\left\|z_{t+1}-\bar{z}_{t+1}\right\|_2^2=\left\|f(\theta_{t+1})-f(\bar{\theta}_{t+1})\right\|_2^2\leq L_f^2\left\|\theta_{t+1}-\bar{\theta}_{t+1}\right\|_2^2 \qquad \text{(Since } f \text{ is } L_f\text{-Lipschitz)}$$

$$\implies \mathbb{E}[\epsilon_{t+1}^2]\leq \frac{4L_f^2}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\left[\ell(z_t)-\ell(z^*)+\sigma_z^2\right]\right]+L_f^2\,\zeta_t^2$$

Taking expectation w.r.t the randomness from iterations $k=0$ to t,

$$\mathbb{E}[\epsilon_{t+1}^2]\leq \frac{4L_f^2}{\mu_g}\left[\exp\left(-m_t/\kappa_g\right)\,\left[\mathbb{E}[\ell(z_t)-\ell(z^*)]+\sigma_z^2\right]\right]+L_f^2\,\zeta_t^2$$

□

D.4. Example to show the necessity of ζ^2 term

Proposition 4.4. *Consider minimizing the sum $h(\theta) := \frac{h_1(\theta)+h_2(\theta)}{2}$ of two one-dimensional quadratics, $h_1(\theta) := \frac{1}{2}(\theta-1)^2$ and $h_2(\theta) = \frac{1}{2}\left(2\theta + 1/2\right)^2$, using* `SSO` *with $m_t = \infty$ and $\eta_t = c\,\alpha_t$ for any sequence of α_t and any constant $c \in (0,1]$.* `SSO` *results in convergence to a neighbourhood of the solution, specifically, if θ^* is the minimizer of h and $\theta_1 > 0$, then, $\mathbb{E}(\theta_T - \theta^*) \geq \min\left(\theta_1, \frac{3}{8}\right)$.*

Proof. Let us first compute $\theta^* := \arg\min h(\theta)$.

$$h(\theta) = \frac{1}{4}(\theta-1)^2 + \frac{1}{4}\left(2\theta + \frac{1}{2}\right)^2 = \frac{5}{4}\theta^2 + \frac{1}{4} + \frac{1}{16} \Rightarrow \theta^* = 0$$

For h_1,

$$\ell_1(z) = \frac{1}{2}(z-1)^2 \quad \text{where } z = \theta$$
$$\tilde{g}_t(\theta) := \frac{1}{2}(\theta_t-1)^2 + (\theta_t-1)\,(\theta_t-\theta) + \frac{1}{2\eta_t}(\theta-\theta_t)^2$$

If $m_t = \infty$, `SSO` will minimize $\tilde{g}_t$ exactly. Since $\nabla\tilde{g}_t(\theta_{t+1}) = 0$,

$$\implies \frac{1}{\eta_t}(\theta_{t+1}-\theta_t) = -(\theta_t - 1) \implies \theta_{t+1} = \theta_t - \eta_t\,(\theta_t-1) \implies \theta_{t+1} = \theta_t - \eta_t \nabla h_1(\theta_t)$$

Similarly, for h_2,

$$\ell_2(z) = \frac{1}{2}(z + 1/2)^2 \quad \text{where } z = 2\theta$$
$$\tilde{g}_t(\theta) := \frac{1}{2}(2\theta_t + 1/2)^2 + (2\theta_t + 1/2)\,(2\theta_t - 2\theta) + \frac{1}{2\eta_t}(2\theta - 2\theta_t)^2$$

If $m_t = \infty$, `SSO` will minimize $\tilde{g}_t$ exactly. Since $\nabla\tilde{g}_t(\theta_{t+1}) = 0$,

$$\implies \frac{4}{\eta_t}(\theta_{t+1}-\theta_t) = -(4\theta_t+1) \implies \theta_{t+1} = \theta_t - \eta_t\,(\theta_t + 1/4) \implies \theta_{t+1} = \theta_t - \frac{\eta_t}{4}\nabla h_2(\theta_t)$$

If $i_t = 1$ and $\eta = c\,\alpha_t$

$$\theta_{t+1} = \theta_t - c\,\alpha_t\,(\theta_t - 1) = c\,\alpha_t + (1 - c\alpha_t)\theta_t$$

If $i_k = 2$,

$$\theta_{t+1} = \theta_t - c\,\alpha_t\frac{2}{4}(2\theta_t + \frac{1}{2}) = (1 - c\,\alpha_t)\theta_t - \frac{1}{4}c\,\alpha_t$$

Then

$$\mathbb{E}\theta_{t+1} = (1-c\,\alpha_t)\theta_t + \frac{1}{2}c\,\alpha_t - \frac{1}{8}c\,\alpha_t = (1-c\,\alpha_t)\theta_t + \frac{3}{8}c\,\alpha_t$$

and

$$\mathbb{E}\theta_T = \mathbb{E}(\theta_T - \theta^*) = (\theta_1 - \theta^*)\prod_{t=1}^{T}(1 - c\,\alpha_t) + \frac{3}{8}\sum_{t=1}^{T}2(1-c)\alpha_t \prod_{i=t+1}^{T}(1 - c\,\alpha_i)$$

Using Lemma D.9 and the fact that $c\,\alpha_t \leq 1$ for all t, we have that if $\theta_1 - \theta^* = \theta_1 > 0$, then,

$$\mathbb{E}(\theta_T - \theta^*) \geq \min\left(\theta_1, \frac{3}{8}\right).$$

□

D.5. Helper Lemmas

The proofs of Lemma D.4, Lemma D.6, and Lemma D.7 can be found in (Vaswani et al., 2022).

Lemma D.4. *For all $x > 1$,*

$$\frac{1}{x-1} \le \frac{2}{\ln(x)}$$

Proof. For $x > 1$, we have

$$\frac{1}{x-1} \le \frac{2}{\ln(x)} \iff \ln(x) < 2x - 2$$

Define $f(x) = 2x - 2 - \ln(x)$. We have $f'(x) = 2 - \frac{1}{x}$. Thus for $x \ge 1$, we have $f'(x) > 0$ so f is increasing on $[1, \infty)$. Moreover we have $f(1) = 2 - 2 - \ln(1) = 0$ which shows that $f(x) \ge 0$ for all $x > 1$ and ends the proof. □

Lemma D.5. *For all $x, \gamma > 0$,*

$$\exp(-x) \le \left(\frac{\gamma}{ex}\right)^{\gamma}$$

Proof. Let $x > 0$. Define $f(\gamma) = \left(\frac{\gamma}{ex}\right)^{\gamma} - \exp(-x)$. We have

$$f(\gamma) = \exp\left(\gamma \ln(\gamma) - \gamma \ln(ex)\right) - \exp(-x)$$

and

$$f'(\gamma) = \left(\gamma \cdot \frac{1}{\gamma} + \ln(\gamma) - \ln(ex)\right) \exp\left(\gamma \ln(\gamma) - \gamma \ln(ex)\right)$$

Thus

$$f'(\gamma) \ge 0 \iff 1 + \ln(\gamma) - \ln(ex) \ge 0 \iff \gamma \ge \exp\left(\ln(ex) - 1\right) = x$$

So f is decreasing on $(0, x]$ and increasing on $[x, \infty)$. Moreover,

$$f(x) = \left(\frac{x}{ex}\right)^{x} - \exp(-x) = \left(\frac{1}{e}\right)^{x} - \exp(-x) = 0$$

and thus $f(\gamma) \ge 0$ for all $\gamma > 0$ which proves the lemma. □

Lemma D.6. *Assuming $\alpha < 1$ we have*

$$A := \sum_{t=1}^{T} \alpha^t \ge \frac{\alpha T}{\ln(T/\beta)} - \frac{2\beta}{\ln(T/\beta)}$$

Proof.

$$\sum_{t=1}^{T} \alpha^t = \frac{\alpha - \alpha^{T+1}}{1-\alpha} = \frac{\alpha}{1-\alpha} - \frac{\alpha^{T+1}}{1-\alpha}$$

We have

$$\frac{\alpha^{T+1}}{1-\alpha} = \frac{\alpha\beta}{T(1-\alpha)} = \frac{\beta}{T} \cdot \frac{1}{1/\alpha - 1} \le \frac{\beta}{T} \cdot \frac{2}{\ln(1/\alpha)} = \frac{\beta}{T} \cdot \frac{2}{\frac{1}{T}\ln(T/\beta)} = \frac{2\beta}{\ln(T/\beta)} \tag{11}$$

where in the inequality we used Lemma D.4 and the fact that $1/\alpha > 1$. Plugging back into A we get,

$$\begin{aligned} A &\geq \frac{\alpha}{1-\alpha} - \frac{2\beta}{\ln(T/\beta)} \\ &\geq \frac{\alpha}{\ln(1/\alpha)} - \frac{2\beta}{\ln(T/\beta)} && (1 - x \leq \ln(\tfrac{1}{x})) \\ &= \frac{\alpha T}{\ln(T/\beta)} - \frac{2\beta}{\ln(T/\beta)} \end{aligned}$$

□

Lemma D.7. *For $\alpha = \left(\frac{\beta}{T}\right)^{1/T}$ and any $\kappa > 0$,*

$$\sum_{t=1}^{T} \alpha^{2t} \exp\left(-\frac{1}{2\kappa}\sum_{i=t+1}^{T}\alpha^i\right) \leq \frac{16\kappa^2 c_2 (\ln(T/\beta))^2}{e^2\alpha^2 T}$$

where $c_2 = \exp\left(\frac{1}{2\kappa}\frac{2\beta}{\ln(T/\beta)}\right)$

Proof. First, observe that,

$$\sum_{i=t+1}^{T}\alpha^i = \frac{\alpha^{t+1} - \alpha^{T+1}}{1-\alpha}$$

We have

$$\frac{\alpha^{T+1}}{1-\alpha} = \frac{\alpha\beta}{T(1-\alpha)} = \frac{\beta}{T}\cdot\frac{1}{1/\alpha - 1} \leq \frac{\beta}{T}\cdot\frac{2}{\ln(1/\alpha)} = \frac{\beta}{T}\cdot\frac{2}{\frac{1}{T}\ln(T/\beta)} = \frac{2\beta}{\ln(T/\beta)}$$

where in the inequality we used D.4 and the fact that $1/\alpha > 1$. These relations imply that,

$$\begin{aligned} &\sum_{i=t+1}^{T}\alpha^i \geq \frac{\alpha^{t+1}}{1-\alpha} - \frac{2\beta}{\ln(T/\beta)} \\ &\implies \exp\left(-\frac{1}{2\kappa}\sum_{i=t+1}^{T}\alpha^i\right) \leq \exp\left(-\frac{1}{2\kappa}\frac{\alpha^{t+1}}{1-\alpha} + \frac{1}{2\kappa}\frac{2\beta}{\ln(T/\beta)}\right) = c_2\exp\left(-\frac{1}{2\kappa}\frac{\alpha^{t+1}}{1-\alpha}\right) \end{aligned}$$

We then have

$$\begin{aligned} \sum_{t=1}^{T}\alpha^{2t}\exp\left(-\frac{1}{2\kappa}\sum_{i=t+1}^{T}\alpha^i\right) &\leq c_2\sum_{t=1}^{T}\alpha^{2t}\exp\left(-\frac{1}{2\kappa}\frac{\alpha^{t+1}}{1-\alpha}\right) \\ &\leq c_2\sum_{t=1}^{T}\alpha^{2t}\left(\frac{2(1-\alpha)2\kappa}{e\alpha^{t+1}}\right)^2 && \text{(Lemma D.5)} \\ &= \frac{16\kappa^2 c_2}{e^2\alpha^2}T(1-\alpha)^2 \\ &\leq \frac{16\kappa^2 c_2}{e^2\alpha^2}T(\ln(1/\alpha))^2 \\ &= \frac{16\kappa^2 c_2(\ln(T/\beta))^2}{e^2\alpha^2 T} \end{aligned}$$

□

Lemma D.8. *For* $\alpha = \left(\frac{\beta}{T}\right)^{1/T}$ *and any* $\zeta > 0$*,*

$$\sum_{t=1}^{T} \exp\left(-\frac{1}{\zeta}\sum_{i=t}^{T}\alpha^i\right) \leq \exp\left(\frac{2\beta\ln(T)}{\zeta\ln(T/\beta)}\right)$$

Proof. First, observe that,

$$\sum_{i=t}^{T}\alpha^i = \frac{\alpha^t - \alpha^{T+1}}{1-\alpha} \geq -\frac{\alpha^{T+1}}{1-\alpha}$$

We have

$$\frac{\alpha^{T+1}}{1-\alpha} = \frac{\alpha\beta}{T(1-\alpha)} = \frac{\beta}{T}\cdot\frac{1}{1/\alpha - 1} \leq \frac{\beta}{T}\cdot\frac{2}{\ln(1/\alpha)} = \frac{\beta}{T}\cdot\frac{2}{\frac{1}{T}\ln(T/\beta)} = \frac{2\beta}{\ln(T/\beta)}$$

where in the inequality we used Lemma D.4 and the fact that $1/\alpha > 1$. Using the above bound we have

$$\sum_{t=1}^{T} \exp\left(-\frac{1}{\zeta}\sum_{i=t}^{T}\alpha^i\right) \leq \sum_{t=1}^{T}\exp\left(\frac{2\beta}{\zeta\ln(T/\beta)}\right) = \exp\left(\frac{2\beta\ln(T)}{\zeta\ln(T/\beta)}\right)$$

□

Lemma D.9. *For any sequence* α_t

$$\prod_{t=1}^{T}(1-\alpha_t) + \sum_{t=1}^{T}\alpha_t\prod_{i=t+1}^{T}(1-\alpha_i) = 1$$

Proof. We show this by induction on T. For $T = 1$,

$$(1-\alpha_1) + \alpha_1 = 1$$

Induction step:

$$\begin{aligned}
\prod_{t=1}^{T+1}(1-\alpha_t) + \sum_{t=1}^{T+1}\alpha_t\prod_{i=t+1}^{T+1}(1-\alpha_i) &= (1-\alpha_{T+1})\prod_{t=1}^{T}(1-\alpha_t) + \left(\alpha_{T+1} + \sum_{t=1}^{T}\alpha_t\prod_{i=t+1}^{T+1}(1-\alpha_i)\right) \\
&= (1-\alpha_{T+1})\prod_{t=1}^{T}(1-\alpha_t) + \left(\alpha_{T+1} + (1-\alpha_{T+1})\sum_{t=1}^{T}\alpha_t\prod_{i=t+1}^{T}(1-\alpha_i)\right) \\
&= (1-\alpha_{T+1})\left(\underbrace{\prod_{t=1}^{T}(1-\alpha_t) + \sum_{t=1}^{T}\alpha_t\prod_{i=t+1}^{T}(1-\alpha_i)}_{=1}\right) + \alpha_{T+1} \qquad \text{(Induction hypothesis)} \\
&= (1-\alpha_{T+1}) + \alpha_{T+1} = 1
\end{aligned}$$

□

Lemma D.10. *(Schmidt et al., 2011, Lemma 1). Assume that the non-negative sequence $\{v_\tau\}$ for $\tau \geq 1$ satisfies the following recursion:*

$$v_\tau^2 \leq S_\tau + \sum_{t=1}^{\tau} \lambda_t v_t \,,$$

where $\{S_\tau\}$ is an increasing sequence, such that $S_0 \geq v_0^2$ and $\lambda_t \geq 0$. Then for all $\tau \geq 1$,

$$v_\tau \leq \frac{1}{2}\sum_{t=1}^{\tau} \lambda_t + \left(S_\tau + \left(\frac{1}{2}\sum_{t=1}^{\tau}\lambda_t\right)^2\right)^{1/2}.$$

Proof. We prove this lemma by induction. For $\tau = 1$ we have

$$v_1^2 \leq S_1 + \lambda_1 v_1 \implies (v_1 - \frac{\lambda_1}{2})^2 \leq S_1 + \frac{\lambda_1^2}{4} \implies v_1 \leq \left(S_1 + \left(\frac{1}{2}\lambda_1\right)^2\right)^{1/2} + \frac{1}{2}\lambda_1$$

Inductive hypothesis: Assume that the conclusion holds for $\tau \in \{1, \dots, k\}$. Specifically, for $\tau = k$,

$$v_k \leq \frac{1}{2}\sum_{t=1}^{k} \lambda_t + \left(S_k + \left(\frac{1}{2}\sum_{t=1}^{k}\lambda_t\right)^2\right)^{1/2}.$$

Now we show that the conclusion holds for $\tau = k+1$. Define $\psi_k := \sum_{t=1}^{k} \lambda_t$. Note that $\psi_{k+1} \geq \psi_k$ since $\lambda_t \geq 0$ for all t. Hence, $v_k \leq \frac{1}{2}\psi_k + \left(S_k + \left(\frac{1}{2}\psi_k\right)^2\right)^{1/2}$. Define $k^* := \arg\max_{\tau \in \{0,\dots,k\}} v_\tau$. Using the main assumption for $\tau = k+1$,

$$\begin{aligned}
& v_{k+1}^2 \leq S_{k+1} + \sum_{t=1}^{k+1} \lambda_t v_t \\
\implies\ & v_{k+1}^2 - \lambda_{k+1} v_{k+1} \leq S_{k+1} + \sum_{t=1}^{k} \lambda_t v_t \\
\implies\ & \left(v_{k+1} - \frac{\lambda_{k+1}}{2}\right)^2 \leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \sum_{t=1}^{k} \lambda_t v_t \\
& \leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + v_{k^*}\sum_{t=1}^{k} \lambda_t && \text{(since } v_{k^*} \text{ is the maximum)} \\
& = S_{k+1} + \frac{\lambda_{k+1}^2}{4} + v_{k^*}\psi_k && \text{(based on the definition for } \psi_k\text{)}
\end{aligned}$$

Since $k^* \leq k$, by the inductive hypothesis,

$$\begin{aligned}
\left(v_{k+1} - \frac{\lambda_{k+1}}{2}\right)^2 &\leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \psi_k \left\{\frac{1}{2}\psi_{k^*} + \left(S_{k^*} + \left(\frac{1}{2}\psi_{k^*}\right)^2\right)^{1/2}\right\} \\
&= S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \frac{1}{2}\psi_k\psi_{k^*} + \psi_k\left(S_{k^*} + \left(\frac{1}{2}\psi_{k^*}\right)^2\right)^{1/2} \\
&\leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \frac{1}{2}\psi_k^2 + \psi_k\left(S_{k^*} + \left(\frac{1}{2}\psi_{k^*}\right)^2\right)^{1/2} && \text{(Since } \{\psi_\tau\} \text{ is non-decreasing and } k^* \leq k\text{)}
\end{aligned}$$

Furthermore, since $\{S_\tau\}$ is increasing and $k^* < k+1$,

$$\begin{aligned}
\left(v_{k+1} - \frac{\lambda_{k+1}}{2}\right)^2 &\leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \frac{1}{2}\psi_k^2 + \psi_k \left(S_{k+1} + \left(\frac{1}{2}\psi_{k^*}\right)^2\right)^{1/2} \\
&\leq S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \frac{1}{2}\psi_k^2 + \psi_k \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} \\
&\qquad \text{(Since } \{\psi_\tau\} \text{ is non-decreasing and } k^* < k+1\text{)} \\
&= S_{k+1} + \frac{\lambda_{k+1}^2}{4} + \frac{1}{4}\psi_k^2 + \psi_k \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{4}\psi_k^2 \\
&\leq S_{k+1} + \frac{1}{4}\psi_{k+1}^2 + \psi_k \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{4}\psi_k^2 \\
&\qquad (a^2 + b^2 \leq (a+b)^2 \text{ for } a = \psi_k > 0 \text{ and } b = \lambda_{k+1} > 0) \\
&= \underbrace{S_{k+1} + \frac{1}{4}\psi_{k+1}^2}_{:=x^2} + \underbrace{\psi_k \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2}}_{=2xy} + \underbrace{\frac{1}{4}\psi_k^2}_{:=y^2} \\
&= \left(\left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{2}\psi_k\right)^2 \\
\implies v_{k+1} - \frac{\lambda_{k+1}}{2} &\leq \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{2}\psi_k \\
\implies v_{k+1} &\leq \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{2}\psi_k + \frac{\lambda_{k+1}}{2} \\
&= \left(S_{k+1} + \left(\frac{1}{2}\psi_{k+1}\right)^2\right)^{1/2} + \frac{1}{2}\psi_{k+1}
\end{aligned}$$

where replacing ψ_{k+1} with its definition gives us the required result. □

E. Additional Experimental Results

Supervised Learning: We evaluate our framework on the LibSVM benchmarks (Chang and Lin, 2011), a standard suite of convex-optimization problems. Here consider two datasets – `mushrooms`, and `rcv1`, two losses – squared loss and logistic loss, and four batch sizes – {25, 125, 625, full-batch} for the linear parameterization ($f = X^\top \theta$). Each optimization algorithm is run for 500 epochs (full passes over the data).

E.1. Stochastic Surrogate Optimization

Comparisons of `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO` evaluated on three SVMLib benchmarks `mushrooms`, `ijcnn`, and `rcv1`, two losses – squared loss and logistic loss, and four batch sizes – {25, 125, 625, full-batch}. Each run was evaluated over three random seeds following the same initialization scheme. All plots are in log-log space to make trends between optimization algorithms more apparent. All algorithms and batch sizes are evaluated for 500 epochs and performance is represented as a function of total optimization steps. We compare stochastic surrogate optimization (`SSO`), against `SGD` with the standard theoretical $1/2L_\theta$ step-size, `SGD` with the step-size set according to a stochastic line-search (Vaswani et al., 2019b) `SLS`, and finally `Adam` (Kingma and Ba, 2015) using default hyper-parameters. Since `SSO` is equivalent to projected SGD in the target space, we set η (in the surrogate definition) to $1/2L$ where L is the smoothness of ℓ w.r.t z. For squared loss, L is therefore set to 1, while for logistic it is set to 2. These figures show (i) `SSO` improves over `SGD` when the step-sizes are set theoretically, (ii) `SSO` is competitive with `SLS` or `Adam`, and (iii) as m increases, on average, the performance of `SSO` improves as projection error decreases. For further details see the attached code repository. Below we include three different step-size schedules: constant, $\frac{1}{\sqrt{t}}$ (Orabona, 2019), and $(1/T)^{t/T}$ (Vaswani et al., 2022).

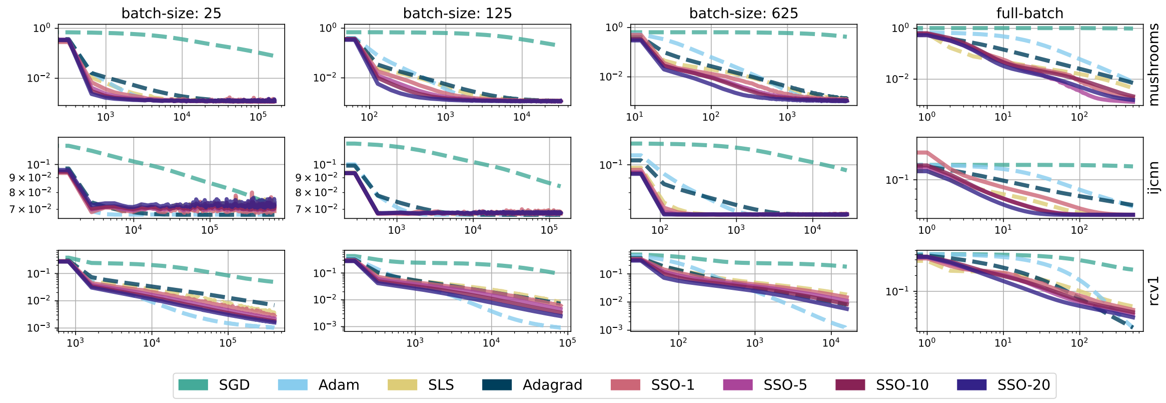


Figure 4: **Constant step-size:** comparison of optimization algorithms under a **mean squared error loss**. We note, `SSO` significantly outperforms its parametric counterpart, and maintains performance which is on par with both `SLS` and `Adam`. Additionally we note that taking additional steps in the surrogate generally improves performance.

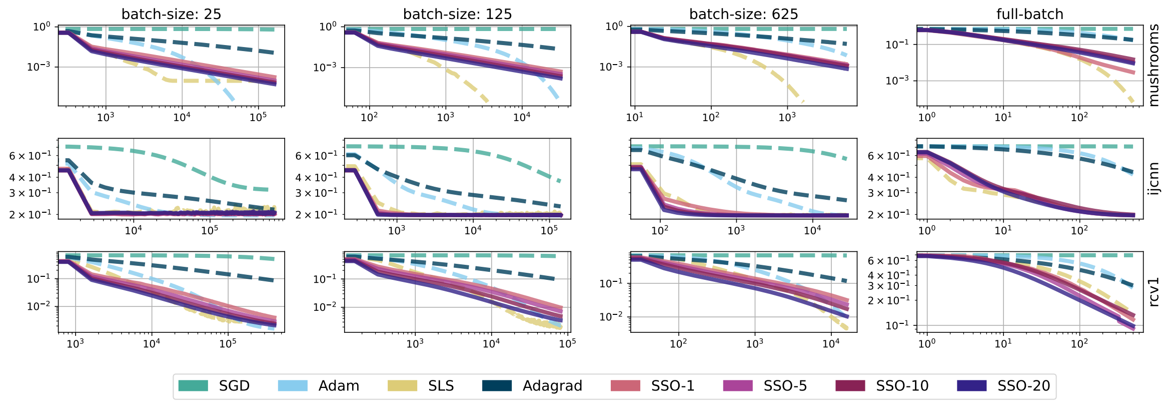


Figure 5: **Constant step-size:** comparison of optimization algorithms under a average **logistic loss**. We note, SSO significantly outperforms its parametric counterpart, and maintains performance which is on par with both SLS and Adam. Additionally we note that taking additional steps in the surrogate generally improves performance

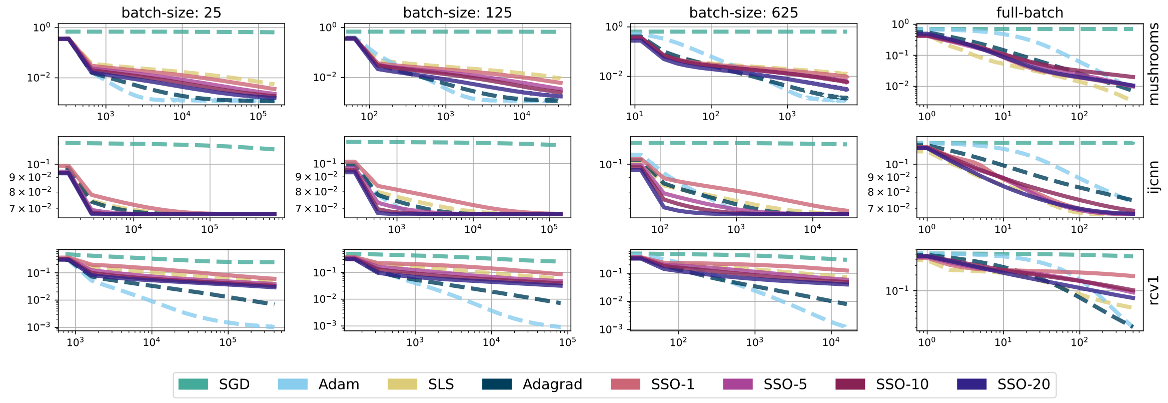


Figure 6: **Decreasing step-size:** comparison of optimization algorithms under a **mean squared error loss**. We compare examples which include a decaying step size of $\frac{1}{\sqrt{t}}$ alongside both SSO as well as SGD and SLS. Adam (and Adagrad). Again, we note that taking additional steps in the surrogate generally improves performance. Additionally the decreasing step-size seems to help maintain strict monotonic improvement.

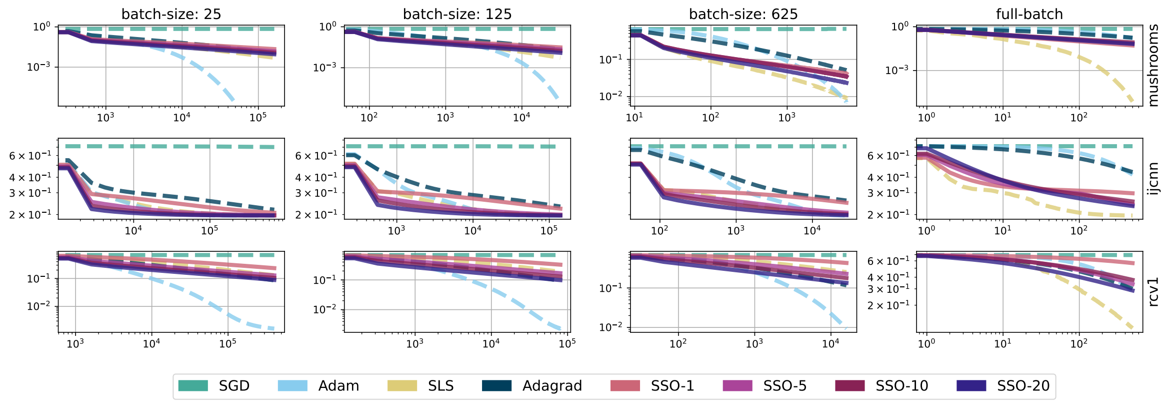


Figure 7: **Decreasing step-size:** comparison of optimization algorithms under a **logistic loss**. We compare examples which include a decaying step size of $\frac{1}{\sqrt{t}}$ alongside both SSO as well as SGD and SLS. Adam (and Adagrad). Again, we note that taking additional steps in the surrogate generally improves performance. Additionally the decreasing step-size seems to help maintain strict monotonic improvement.

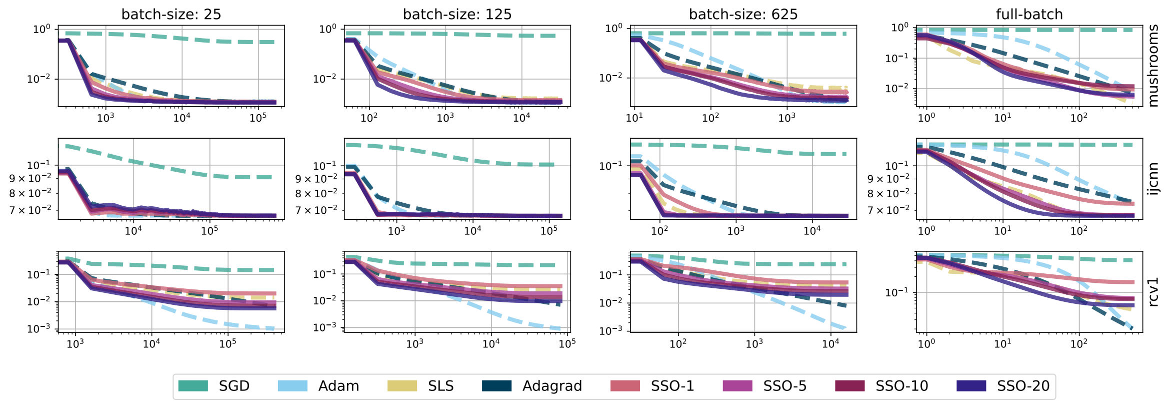


Figure 8: **Exponential step-size:** comparison of optimization algorithms under a **mean squared error loss**. We compare examples which include a decaying step size of $(\frac{1}{T})^{t/T}$ alongside both SSO as well as SGD and SLS. Adam. Again, we note that taking additional steps in the surrogate generally improves performance. Additionally the decreasing step-size seems to help maintain strict monotonic improvement. Lastly, because of a less aggressive step size decay, the optimization algorithms make more progress then their stochastic $\frac{1}{\sqrt{t}}$ counterparts.

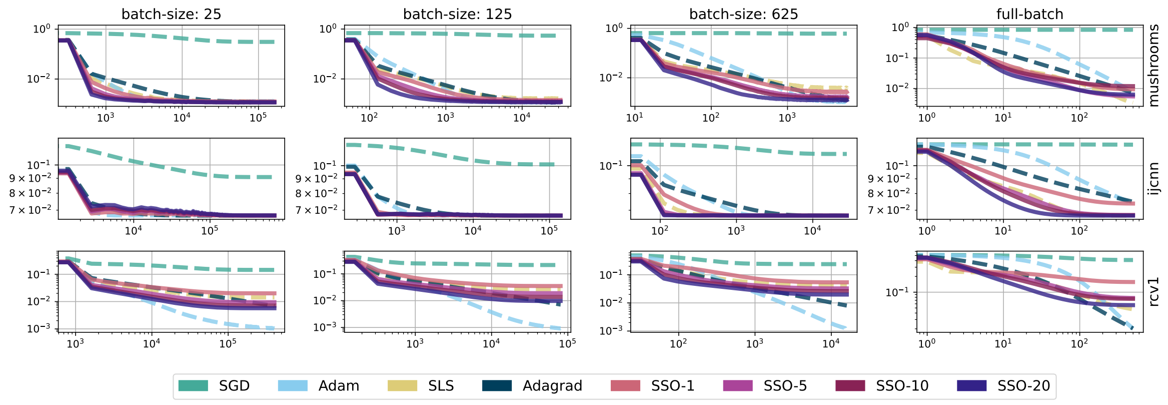


Figure 9: **Exponential step-size:** comparison of optimization algorithms under a **logistic loss**. We compare examples which include a decaying step size of $(\frac{1}{T})^{t/T}$ alongside both SSO as well as SGD and SLS. Adam remains the same as aboves. Again, we note that taking additional steps in the surrogate generally improves performance. Additionally the decreasing step-size seems to help maintain strict monotonic improvement. Lastly, because of a less aggressive step size decay, the optimization algorithms make more progress then their stochastic $\frac{1}{\sqrt{t}}$ counterparts.

E.2. Stochastic Surrogate Optimization with a Line-search

Comparisons of SGD, SLS, Adam, Adagrad, and SSO-SLS evaluated on three SVMLib benchmarks mushrooms, ijcnn, and rcv1. Each run was evaluated over three random seeds following the same initialization scheme. All plots are in log-log space to make trends between optimization algorithms more apparent. As before in all settings, algorithms use either their theoretical step-size when available, or the default as defined by (Paszke et al., 2019). The inner-optimization loop are set according to line-search parameters and heuristics following Vaswani et al. (2019b). All algorithms and batch sizes are evaluated for 500 epochs and performance is represented as a function of total optimization steps.

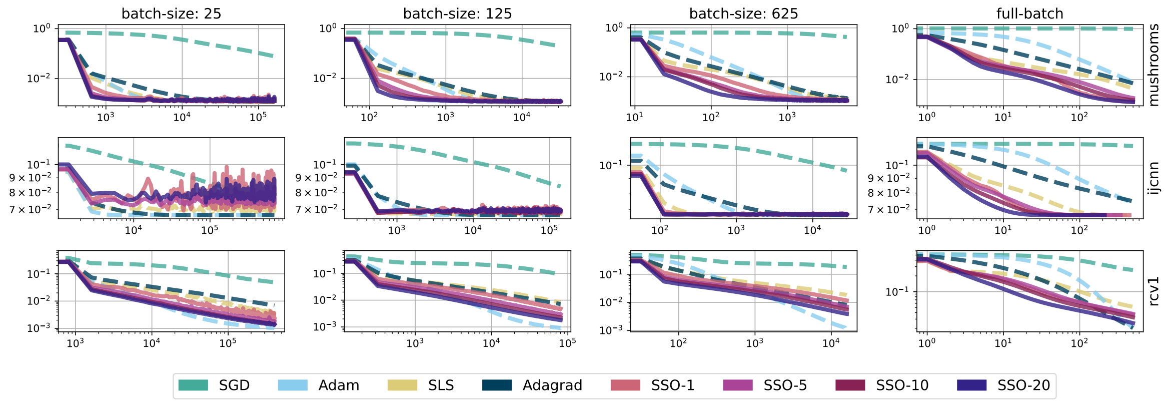


Figure 10: **Constant step-size:** comparison of optimization algorithms under a **mean squared error loss**. We note, SSO-SLS outperforms its parametric counterpart, and maintains performance which is on par with both SLS and Adam. Additionally we note that taking additional steps in the surrogate generally improves performance, especially in settings with less noise (full-batch and batch-size 625).

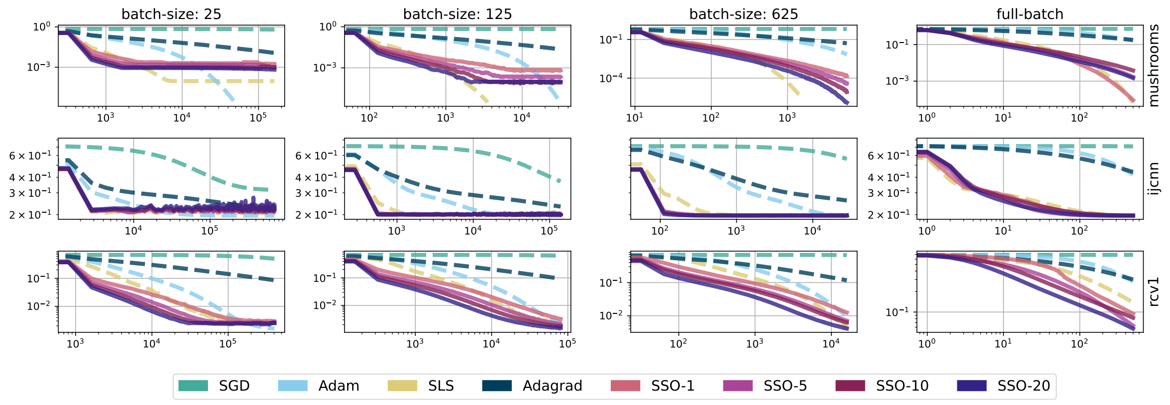


Figure 11: **Constant step-size:** comparison of optimization algorithms under a average **logistic loss**. We note, `SSO-SLS` outperforms its parametric counterpart, and maintains performance which is on par with both `SLS` and `Adam`. Additionally we note that taking additional steps in the surrogate generally improves performance.

E.3. Combining Stochastic Surrogate Optimization with Adaptive Gradient Methods

Comparisons of `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO-Adagrad` evaluated on three SVMLib benchmarks `mushrooms`, `ijcnn`, and `rcv1`. Each run was evaluated over three random seeds following the same initialization scheme. All plots are in log-log space to make trends between optimization algorithms more apparent. As before in all settings, algorithms use either their theoretical step-size when available, or the default as defined by (Paszke et al., 2019). The inner-optimization loop are set according to line-search parameters and heuristics following Vaswani et al. (2019b). All algorithms and batch sizes are evaluated for 500 epochs and performance is represented as a function of total optimization steps. Here we update the η according to the same schedule as scalar `Adagrad` (termed AdaGrad-Norm in Ward et al. (2020)). Because Adagrad does not have an easy to compute optimal theoretical step size, for our setting we set the log learning rate (the negative of $\log \eta$) to be 2.. For further details see the attached coding repository.

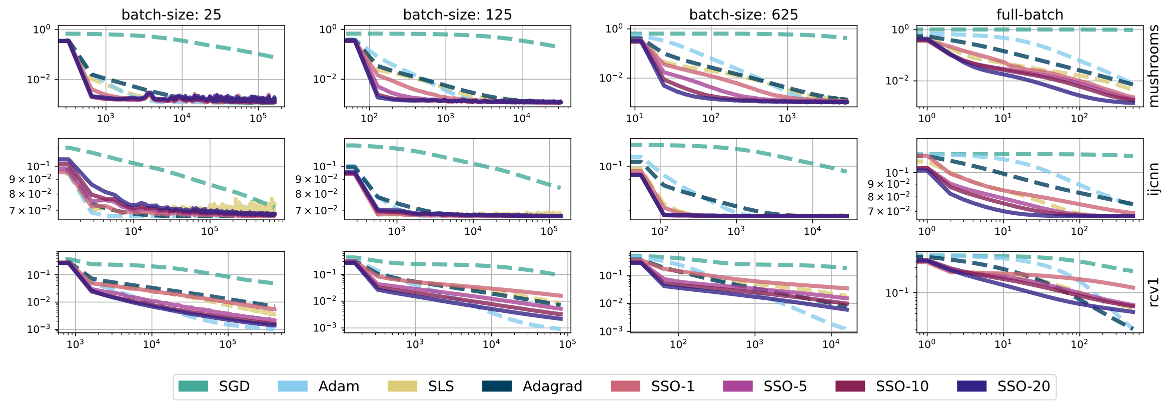


Figure 12: Comparison in terms of average MSE loss of `SGD`, `SLS`, `Adam`, and `SSO-Adagrad` evaluated under a **mean squared error loss**. These plots show that `SSO-Adagrad` outperforms its parametric counterpart, and maintains performance which is on par with both `SLS` and `Adam`. Additionally, we again find that taking additional steps in the surrogate generally improves performance.

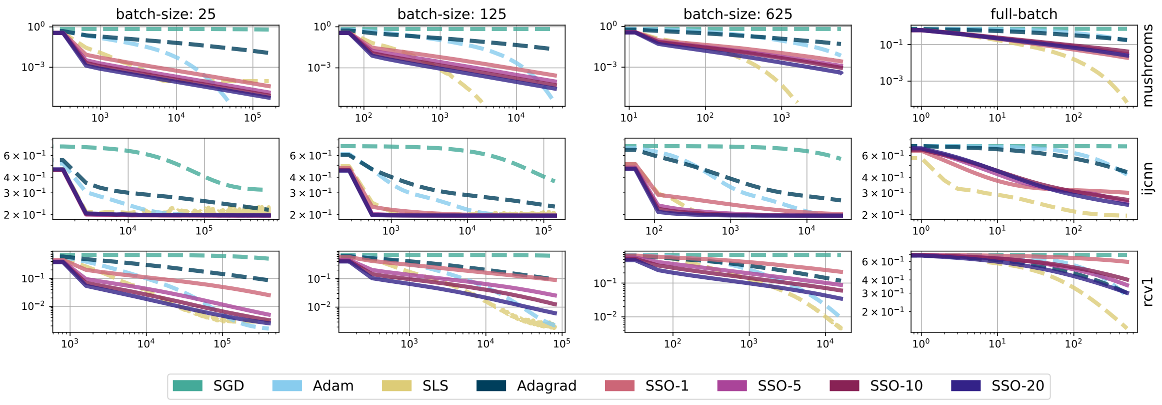


Figure 13: Comparison in terms of average MSE loss of `SGD`, `SLS`, `Adam`, and `SSO-Adagrad` evaluated under a **logistic loss**. These plots show that `SSO-Adagrad` outperforms its parametric counterpart, and maintains performance which is on par with both `SLS` and `Adam`. Additionally, we again find that taking additional steps in the surrogate generally improves performance.

E.4. Combining Stochastic Surrogate Optimization With Online Newton Steps

Comparisons of `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO-Newton` evaluated on three SVMLib benchmarks `mushrooms`, `ijcnn`, and `rcv1`. Each run was evaluated over three random seeds following the same initialization scheme. All plots are in log-log space to make trends between optimization algorithms more apparent. As before, in all settings, algorithms use either their theoretical step-size when available, or the default as defined by (Paszke et al., 2019). The inner-optimization loop are set according to line-search parameters and heuristics following Vaswani et al. (2019b). All algorithms and batch sizes are evaluated for 500 epochs and performance is represented as a function of total optimization steps. Here we update the η according to the same schedule as `Online Newton`. We omit the MSE example as `SSO-Newton` in this setting is equivalent to `SSO`. In the logistic loss setting however, which is displayed below, we re-scale the regularization term by $(1-p)p$ where $p = \sigma(f(x))$ where σ is this sigmoid function, and f is the target space. This operation is done per-data point, and as can be seen below, often leads to extremely good performance, even in the stochastic setting. In the plots below, if the line vanishes before the maximum number of optimization steps have occurred, this indicates that the algorithm has converged to the minimum and is no longer executed. Notably, `SSO-Newton` achieves this in for multiple data-sets and batch sizes.

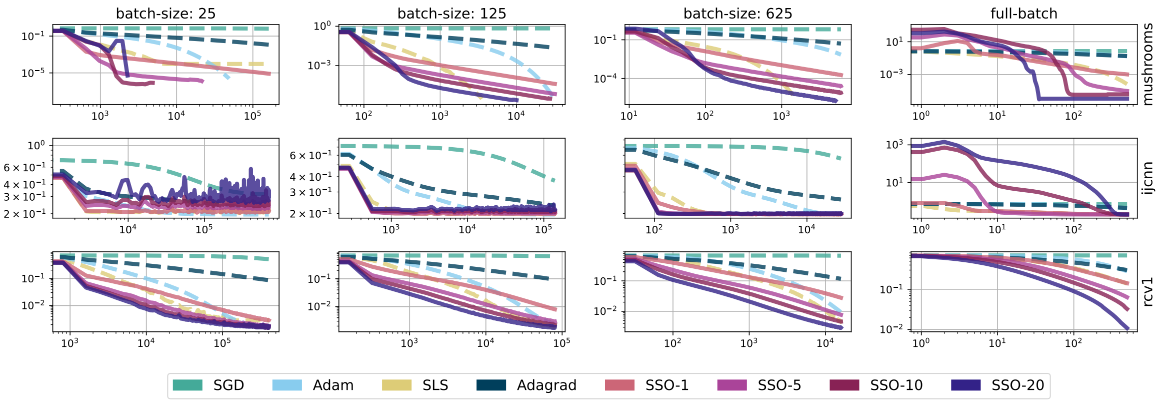


Figure 14: Comparison in terms of average logistic loss of `SGD`, `SLS`, `Adam`, and `SSO-Newton` evaluated on the **logistic loss**. This plot displays that significant improvement can be made at no additional cost by re-scaling the regularization term correctly. Note that in the case of mushrooms, `SSO-Newton` in all cases for $m = 20$ reaches the stopping criteria before the 500th epoch. Second, even in many stochastic settings, `SSO-Newton` outperforms both `SLS` and `Adam`.

E.5. Comparison with SVRG

Below we include a simple supervised learning example in which we compare the standard variance reduction algorithm SVRG (Johnson and Zhang, 2013) to the SSO algorithm in the supervised learning setting. Like the examples above, we test the optimization algorithms using the Chang and Lin (2011) rcv1 dataset, under an MSE loss. This plot shows that SSO can outperform SVRG for even small batch-size settings. For additional details on the implementation and hyper-parameters used, see `https://github.com/WilderLavington/Target-Based-Surrogates-For-Stochastic-Optimization`. We note that it is likely, that for small enough batch sizes SVRG may become competitive with SSO, however we leave such a comparison for future work.

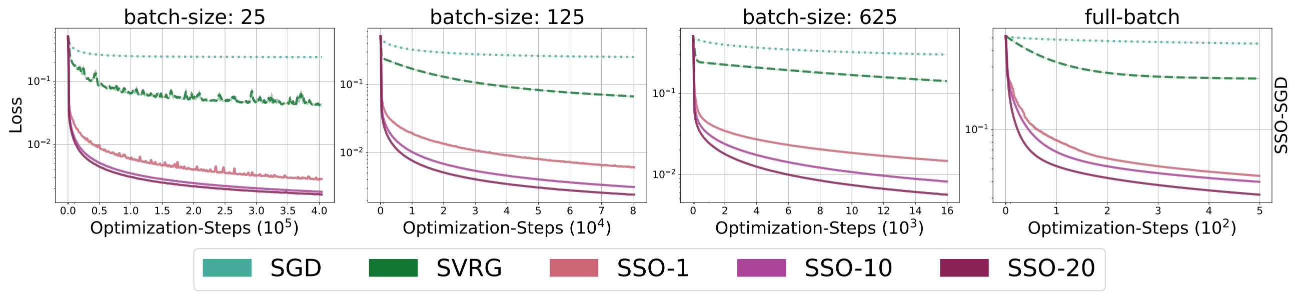
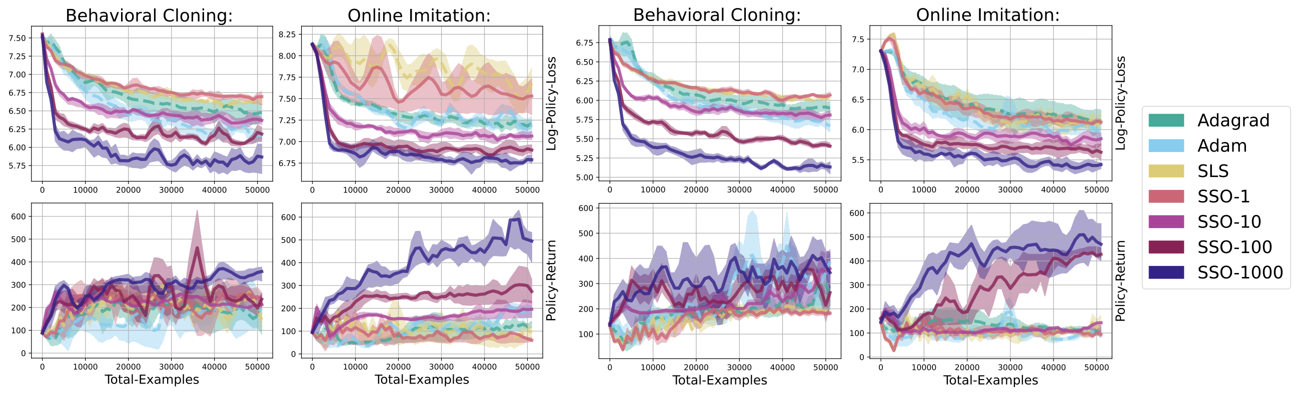


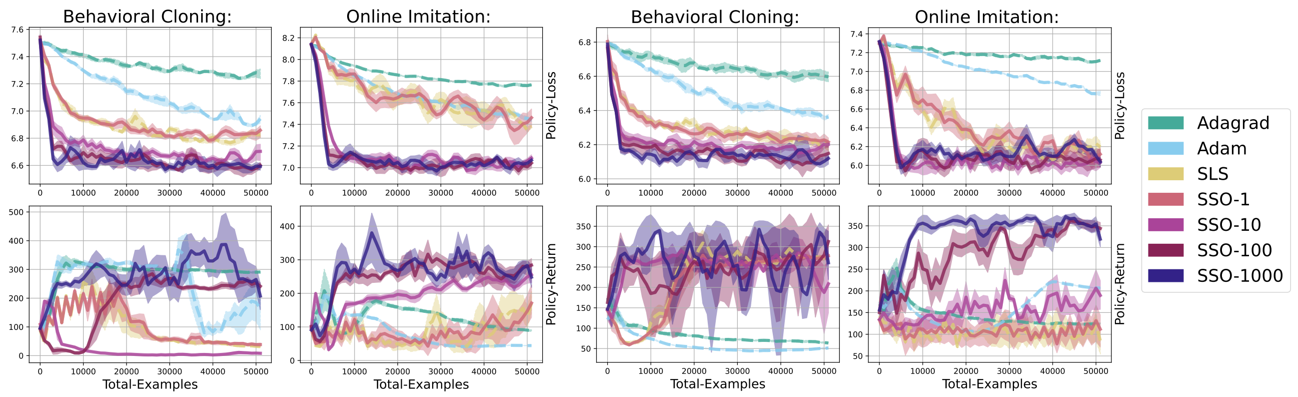
Figure 15: Comparison of the average mean-squared error between `SGD`, `SVRG`, and `SSO-SGD`. This plot displays that even in settings where the batch-size is small, `SSO` can even improve over variance reduction techniques like SVRG.

E.6. Imitation Learning



(a) Walker2d-v2 Environment (b) Hopper-v2 Environment

Figure 16: Comparison of policy return, and log policy loss incurred by `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO` as a function of the total interactions. Unlike Section 5, the mean of the policy is parameterized by a **neural network model**. In both environments, for both behavioral policies, `SSO` outperforms all other online-optimization algorithms. Additionally, as m in increases, so to does the performance of `SSO` in terms of both the return as well as the loss.



(a) Walker2d-v2 Environment (b) Hopper-v2 Environment

Figure 17: Comparison of policy return, and log policy loss incurred by `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO` as a function of the total interactions. Unlike Section 5, the mean of the policy is parameterized by a **linear model**. In both environments, for both behavioral policies, `SSO` outperforms all other online-optimization algorithms. Additionally, as m in increases, so to does the performance of `SSO` in terms of both the return as well as the loss.

Comparisons of `Adagrad`, `SLS`, `Adam`, and `SSO` evaluated on two Mujoco (Todorov et al., 2012) imitation learning benchmarks (Lavington et al., 2022), `Hopper-v2`, and `Walker-v2`. In this setting training and evaluation proceed in rounds. At every round, a behavioral policy samples data from the environment, and an expert labels that data. The goal is guess at the next stage (conditioned on the sampled states) what the expert will label the examples which are gathered. Here, unlike the supervised learning setting, we receive a stream of new data points which can be correlated and drawn from following different distributions through time. Theoretically this makes the optimization problem significantly more difficult, and because we must interact with a simulator, querying the stochastic gradient can be expensive. Like the example in Appendix E, in this setting we will interact under both the experts policy distribution (behavioral cloning), as well as the policy distribution induced by the agent (online imitation). We parameterize a standard normal distribution whose mean is learned through a mean squared error loss between the expert labels and the mean of the agent policy. Again, the expert is trained following soft-actor-critic. All experiments were run using an NVIDIA GeForce RTX 2070 graphics card. with a AMD Ryzen 9 3900 12-Core Processor.

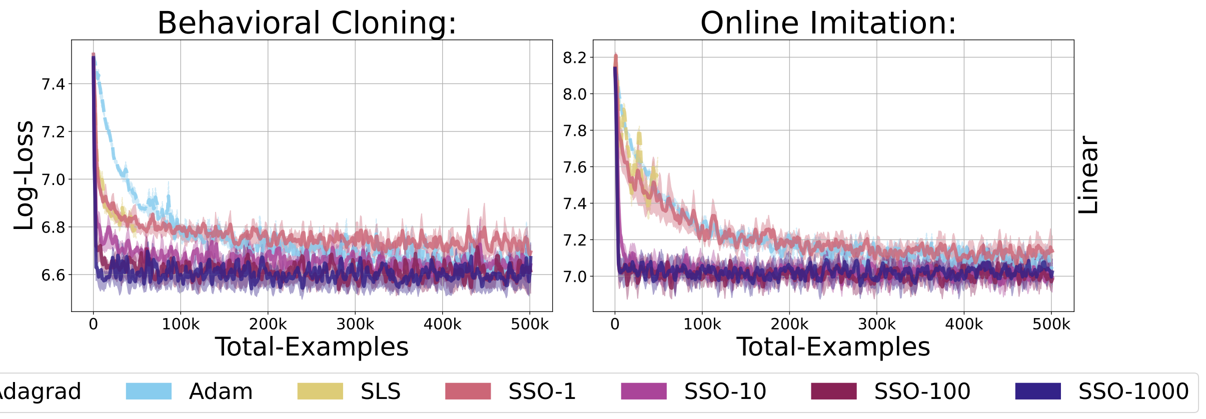


Figure 18: Comparison of log policy loss incurred by `SGD`, `SLS`, `Adam`, `Adagrad`, and `SSO` as a function of the total interactions for the Walker2d gym environment. Unlike Section 5, the mean of the policy is parameterized by a **linear model**. Unlike Fig. 17 and Fig. 16,this plot displays 500 thousand iterations instead of only 50. We again see that `SSO` outperforms all other online-optimization algorithms, *even* for a very large number of iterations. Additionally, as m in increases, we again see the performance of `SSO` log loss improves as well.

In this this setting we evaluate two measures: the per-round log-policy loss, and the policy return. The log policy loss is as described above, while the return is a measure of how well the imitation learning policy actually solves the task. In the Mujoco benchmarks, this reward is defined as a function of how quickly the agent can move in space, as well as the power exerted to move. Imitation learning generally functions by taking a policy which has a high reward (e.g. can move through space with very little effort in terms of torque), and directly imitating it instead of attempting to learn a cost to go function as is done in RL (Sutton et al., 2000).

Each algorithm is evaluated over three random seeds following the same initialization scheme. As before, in all settings, algorithms use either their theoretical step-size when available, or the default as defined by (Paszke et al., 2019). The inner-optimization loop is set according to line-search parameters and heuristics following Vaswani et al. (2019b). All algorithms and batch sizes are evaluated for 50 rounds of interaction (accept in the case of Fig. 18, which is evaluated for 500) and performance is represented as a function of total interactions with the environment. Below we learn a policy which is parameterized by a two layer perception with 256 hidden units and relu activations (as was done in the main paper). For further details, please see the attached code repository.