Developing a Multi-fidelity Approach to Numerically Optimize SVRG

Neymika Jain[*], Elizabeth Qian[†], Pan Xu[‡]

November 1, 2022

Abstract

Stochastic gradient descent (SGD) is a commonly used optimization method for regression problems but has a sublinear convergence rate due to variance introduced by random data selection. Stochastic variance reduced gradient descent (SVRG) achieves linear convergence for strongly convex functions by correcting SGD's update rule using the full gradient. This method of correction is similar to principles used in multifidelity Monte Carlo methods, which reduce Monte Carlo estimator variance. We empirically test different theoretically-supported choices of SVRG hyperparameters to achieve the optimal convergence results for a given computational budget. We then compare these results with SGD and other SVRG estimators with different hyperparameters for a strongly convex function using a linear loss function. We found that using a multifidelity approach resulted in an improvement of 58.3%, i.e. 7 fewer total gradient computations, compared to the optimal standard SVRG results.

1 Introduction

Stochastic gradient descent (SGD) is a common algorithm solving large-scale optimization problems where computation over large data is expensive. However, due to the randomness induced by stochastic selection of data, the algorithm may often not fully converge to the global minima. To reduce this variance introduced by the SGD estimator, a newer approach known as stochastic variance reduced gradient descent (SVRG) was developed. SVRG reduces the variance introduced by stochasticity by introducing a snapshot estimator that only varies every epoch instead of within the batch. Thus by changing the update rule within the batch update, SVRG can use a larger step size and converge faster than SGD for convex problems [1]. Current research by technical mentor Pan Xu with random batch based SVRG demonstrates faster convergence rates for non-convex problems as well. Convergence rates could potentially be improved using a batch sizing approach inspired by ideas from multifidelity Monte Carlo.

Multilevel discretization methods, such as multilevel Monte Carlo, have been useful in minimization problems where computational allocations are limited [2]. Current research by mentor Elizabeth Qian uses multifidelity Monte Carlo methods (MFMC) to reduce computational expense and improved convergence rates of variance and sensitivity indices for the Ishigami function [3]. For a predetermined computational budget, MFMC uses a high-fidelity model and an optimal combination of surrogate models to reduce runtime and estimator error. Unlike multilevel methods, the surrogate models can be any general type instead of dependent on the high-fidelity model. By applying multifidelity principles to the SVRG method, we achieved the goal to empirically study convergence rates and runtime improvements of this method through numerical examples of gaussian least-squares problems. This research provides a basis that can extend to general convex problems.

[*]Arthur Adams SURF Fellow at Caltech
[†]Mentor and Instructor (CMS) at Caltech
[‡]Technical Mentor and Researcher (CMS) at Caltech

2 Methods

2.1 Problem Formulation

Regression techniques are methods that use data to define a function describing a relationship between an independent variable or input and dependent variables or outputs. In particular, linear regression is the most commonly used technique to determine a relationship between the independent variable, x with data points $\{x^i\}_{i=1}^n$, and the dependent variable, y with data points $\{y^i\}_{i=1}^n$ with some error $\{e^i\}_{i=1}^n$ [4]. In linear regression, we consider modeling the following linear function, $f(x^i; w)$, with linear weight parameters $\{w_j\}_{j=1}^k$ where x_j^i is the j-th component of data point x^i:

$$y^i = f(x^i; w) + e^i \text{ where } f(x^i; w) = w_0 + w_1 x_1^i + ... + w_k x_k^i. \tag{1}$$

We can solve for w by minimizing the mean square error (MSE) function, $\mathcal{L}(w)$:

$$\mathcal{L}(w) = \frac{1}{n}\sum_{i=1}^{n}(e^i)^2 = \frac{1}{n}\sum_{i=1}^{n}\psi^i(w). \tag{2}$$

Particularly, we define in (2), $\mathcal{L}(w)$ as the *total loss* function and $\psi^i(w)$ as the *component loss* function. With this method, we select weights that minimize the sum of the squared residuals, where the component loss function for each i-th data point is denoted by $\psi^i(w) = (y^i - f(x^i; w))^2$. We can define the minimization problem over $\mathbb{R}^k$ for w, the $k \times 1$ coefficient vector; y, an $n \times 1$ vector of the dependent variables; and X, a $n \times (k-1)$ matrix of independent variables:

$$w = ([X\ \mathbf{1}]^T[X\ \mathbf{1}])^{-1}[X\ \mathbf{1}]^T y \implies \arg\min_{w \in \mathbb{R}^k} \mathcal{L}(w). \tag{3}$$

Thus, for all linear regression problems, we can solve for w by noting the relationship reaches its minimum value when the slope is 0. We can solve the exact same problems and more general problems where the optimal values, w^*, cannot be found by setting the slope to 0 by using gradient descent.

2.2 Descent Methods

2.2.1 Gradient Descent

While (3) defines the analytical solution, we can determine a numerical solution by using gradient descent to determine optimal values, w^*. In this iterative approach, we calculate the gradient of the total loss function at a random initial point, w_{t-1}, using all x_i, i.e. $\nabla\mathcal{L}(w_{t-1})$. We then determine the current parameter w_t from the prior parameter w_{t-1} by moving in the direction of the negative gradient by a step size η_t:

$$w_t = w_{t-1} - \eta_t \nabla\mathcal{L}(w_{t-1}) = w_{t-1} - \frac{\eta_t}{n}\sum_{i=1}^{n}\nabla\psi^i(w_{t-1}). \tag{4}$$

We continue this process until the gradient reaches an approximate optima, i.e. when $||\nabla\mathcal{L}(w_t)||_2 < \tau$ for some small real constant τ. The main challenge in using gradient descent for learning models from data is for every iteration, gradients for all n component losses must be computed, resulting in large per-iteration cost as n increases.

2.2.2 Stochastic Gradient Descent (SGD)

SGD seeks to address the challenge of high computational cost of gradient descent by only calculating a random set of $B \ll n$ component loss gradients per epoch. Instead of computing the gradient of the total loss function, either a single random index i_t or a random set of B indices I_t is drawn, and an estimate

of the total gradient is computed using just the single random index or the smaller set of random indices. These indices are uniformly draw iid samples from a set of total possible indices, which ensures unbiased estimation of the gradient as $\mathbb{E}[\nabla\psi^{i_t}(w_{t-1})|w_{t-1}] = \nabla\mathcal{L}(w_{t-1})$ [5]. Thus, in the case of a single random selection, we can write our iterative equation using the SGD estimator, v_t, as

$$w_t = w_t - \eta_t v_t = w_{t-1} - \eta_t \nabla\psi^{i_t}(w_{t-1}). \tag{5}$$

With SGD, as opposed to N evaluations, we only need to compute B gradients per iteration, reducing the cost of the algorithm. However, SGD suffers from a drawback since this reduction of computation introduces variance because the estimator, v_t, is a random variable with nonzero variance. For any t, we define

$$\sigma^2 \geq \mathbb{E}[||\nabla\psi^{i_t}(w_t) - \nabla\mathcal{L}(w_t)||_2^2], \tag{6}$$

and the mini-batch SGD estimator, v_t, such that the variance of the SGD gradient estimator can be bounded by

$$\mathbb{E}[||v_t||^2] \leq 2||\nabla\mathcal{L}(w_{t-1})||^2 + \frac{2\sigma^2}{B}. \tag{7}$$

Due to the introduced variance as a result of random sampling, the original step size constant, η_t must sublinearly decay to zero over several epochs in order to ensure convergence to optimum w^*. This decay slows down SGD's convergence rate from gradient descent's linear rate, $O(\log t)$, to a sublinear one, $O(1/t)$ [5]. To increase the step size thereby increasing our convergence rate, a newer algorithm, SVRG, was developed [1].

2.2.3 Stochastic Variance Reduced Gradient (SVRG)

In SVRG, after every epoch of m SGD iterations, a "snapshot" w_s is calculated and stored. The *update rule* or method by which we update the snapshot at the end of the epoch can vary. We will focus on two update rule choices: 1) a naive update where at the end of the epoch $w_s = w_t$ where w_t is the last iteration in batch size B_t^{small} to be updated, and, 2) a random update where $w_s = w_i$ where $w_i \in \{w_1, ..., w_t\}$ is randomly selected from the batch of the iteration updates. We can then define two separate random sets of indices, B_t^{small} and B_s^{large} where $|B_t^{small}| \ll |B_s^{large}|$. Essentially, we redefine B_s^{large} every epoch while redefining B_t^{small} every iteration. We can write the iteration rule for the SVRG algorithm as

$$w_t = w_{t-1} - \eta_t u_t \tag{8}$$

where u_t is defined as a linear combination of the loss functions over different random indices:

$$u_t = \nabla\mathcal{L}^{small}(w_{t-1}) - \nabla\mathcal{L}^{small}(w_s) + \nabla\mathcal{L}^{large}(w_s) \tag{9}$$

$$= \frac{1}{B_t^{small}}\Big(\sum_{m\in B_t^{small}} \nabla\psi^m(w_{t-1}) - \sum_{m\in B_t^{small}} \nabla\psi^m(w_s)\Big) + \frac{1}{B_s^{large}} \sum_{m\in B_s^{large}} \nabla\psi^m(w_s) \tag{10}$$

where we note that we can reduce the induced randomness from SGD and use a larger constant or monotonically decreasing learning rate, which increases the convergence rate [1]. Furthermore, based on our update rule for our snapshot, our convergence rate may change. For instance, if we naively enforce $w_s = w_t$, then we have a slower convergence than if we choose a random $i \in [1, t]$ and set $w_s = w_i$ [1]. Similar to the SGD variance bound, we can also define the SVRG variance upper bound based on [1] as

$$\mathbb{E}[||u_t||^2] \leq 2||\nabla\mathcal{L}(w_{t-1})||^2 + \frac{2\sigma^2}{B_s^{large}} + \frac{2}{B_t^{small}}||w_{t-1} - w_s||^2. \tag{11}$$

Consequently, we can directly compare the upper bounds of the gradient estimators' variances between the two algorithms. For $|B_s^{large}| > n$, the SVRG variance is smaller than the SGD variance. When the snapshot is close to the optima, i.e. when $\nabla\psi^i(w_s) \to \nabla\psi^i(w^*)$, we note a constant, non-decaying learning

rate can be used in the SVRG algorithm. Thus, with SVRG, we can leverage the computational efficiency of SGD while reducing the number of required iterations such that we achieve a faster, linear convergence rate of $O(\log t)$ [1].

2.3 Multifidelity Methods

Multifidelity methods, such as multifidelity Monte Carlo (MFMC), exploit a collection of lower-fidelity surrogate models, $f^{(2)}, ..., f^{(k)}$, to estimate the expected outputs of a high-fidelity model, $s = \mathbb{E}[f^{(1)}(Z)]$, given random variable inputs, Z. Unlike multilevel methods, such as multilevel Monte Carlo (MLMC), multifidelity methods have no hierarchy of the lower-fidelity models, so any type of general surrogate model can be used [2]. To reduce the MSE of the multifidelity estimator described in (2), both bias and variance must be reduced. To ensure unbiasedness, the multifidelity estimator is occasionally corrected using the high-fidelity model [2]. Thus, we focus on the use the surrogate models as control variates to reduce MSE of the variance of the multifidelity estimator [3].

Specifically, for the MFMC method with k models, we define indices $m = [\lfloor m_i \rfloor]^T \in \mathbb{N}^k$ for $i = 1, ..., k$ where $m_j > m_l > 0$ for $k \geq j > l \geq 1$. We draw m_k iid realizations of Z, $z_1, ..., z_{m_k} \in \mathcal{Z}$, and for model evaluations $f^{(i)}(z_1), ..., f^{(i)}(z_{m_i})$, we derive the Monte Carlo estimator,

$$\bar{y}^{(i)}_{m_i} = \frac{1}{m_i} \sum_{j=1}^{m_i} f^{(i)}(z_j), \tag{12}$$

to calculate the MFMC estimator $\hat{s}$ of s:

$$\hat{s} = \bar{y}^{(1)}_{m_1} + \sum_{j=2}^{k} \alpha_j (\bar{y}^{(j)}_{m_j} - \bar{y}^{(j)}_{m_{j-1}}). \tag{13}$$

In order to determine the optimal coefficients α_j and number of realizations m_1, m_j which minimize the MSE of the variance of the MFMC estimator, we first define the variance of the MFMC estimator using σ_k, the standard deviation of $f^{(k)}(Z)$, and the Pearson's correlation coefficient, $\rho_{j,l} = \frac{\text{Cov}[f^{(j)}(Z), f^{(l)}(Z)]}{\sigma_j \sigma_l}$, as:

$$\text{Var}[\hat{s}] = \frac{\sigma_1^2}{m_1} + \sum_{j=2}^{k} (\frac{1}{m_{j-1}} - \frac{1}{m_i})(\alpha_j^2 \sigma_j^2 - 2\alpha_j \rho_{1,j} \sigma_1 \sigma_j). \tag{14}$$

Since the MFMC estimator is unbiased [1], we can define the minimization problem for a fixed computational budget, $p = w^T m \in \mathbb{R}_+$, and computational costs of models $f^{(1)}, ..., f^{(k)}$, $c = [c_1, ..., c_k]$, over $\alpha_2, ..., \alpha_k \in \mathbb{R}^k$ and $m = [m_1, ..., m_k]^T \in \mathbb{R}^k_+$ as

$$\alpha_j = \frac{\rho_{1,j} \sigma_1}{\sigma_j}, m_1 = \frac{p}{c^T r}, m_j = m_1 r_j \implies \arg \min_{m, \alpha_2, ..., \alpha_k} \text{Var}[\hat{s}] \tag{15}$$

where for the global solutions we define $r = [r_i]^T = [r_1, ..., r_k]^T \in \mathbb{R}^k_+$ as:

$$r_i = \sqrt{\frac{c_1(\rho^2_{1,i} - \rho^2_{1,i+1})}{c_i(1 - \rho^2_{1,2})}}. \tag{16}$$

Based on these definitions, we note that the MSE of the MFMC estimator is smaller than the MSE Monte Carlo estimator for any given cost, and the multifidelity estimator is computationally faster [1].

2.4 Application of Multifidelity Principles in SVRG

In order to apply the results from the results from the multifidelity minimization solution, we need to determine an equivalent expression within the SVRG algorithm for the problem and metrics used. We note that we can rewrite the high-fidelity function, $f^{(1)}$, and low-fidelity functions, $f^{(j)}$ as

$$f^{(1)}(Z) = \nabla \mathcal{L}^{large}(w) = \frac{1}{B_s^{large}} \sum_{m \in B_s^{large}} \nabla \psi^m(w_s) \tag{17}$$

$$f^{(j)}(Z) = \mathcal{L}^{small}(w) = \frac{1}{B_t^{small}} \sum_{m \in B_t^{small}} \nabla \psi^m(w), \tag{18}$$

where the computational costs are the number of component gradients calculated. Thus, for the high-fidelity function, $c_1 = B_s^{large}$, and the low-fidelity ones, $c_j = B_t^{small}$. We calculate the standard deviations using the snapshot vector, w_s, for σ_1 and the current vectors, w_t, for σ_j. Finally, we note for vector spaces, the Pearson's correlation coefficient, ρ can be rewritten as

$$\rho_{1,j} = \frac{\text{Cov}[f^{(1)}(Z), f^{(j)}(Z)]}{\sigma_1 \sigma_j} \tag{19}$$

$$= \frac{\langle f^{(1)}(Z) - \bar{f^{(1)}}, f^{(j)}(Z) - \bar{f^{(j)}} \rangle}{||f^{(1)}(Z) - \bar{f^{(1)}}|| \cdot ||f^{(j)}(Z) - \bar{f^{(j)}}||} \tag{20}$$

$$= \frac{\langle f^{(1)}(Z), f^{(j)}(Z) \rangle}{||f^{(1)}(Z)|| \cdot ||f^{(j)}(Z)||} \tag{21}$$

$$= \cos f^{(1)}(Z), f^{(j)}(Z), \tag{22}$$

since as $w_s \to w^*$, $f^{(1)}(w_s) \to \bar{f^{(1)}}$ and $f^{(1)}(w_t) \to \bar{f^{(j)}}$. Thus, for a fixed number of gwe can use multifidelity principles in SVRG to determine the optimal step sizes, $\eta_t = \alpha_j$, and number of iterations per epoch, $M = m_2$.

3 Results

3.1 Current Results

3.1.1 Multifidelity Monte Carlo results

Similar to ongoing work, we can analytically and numerically compare the MSE of Monte Carlo and MFMC estimators using Ishigami functions [3]. Using a random uniform sample, $Z \sim \mathcal{U}(-\pi, \pi)$ we define the high fidelity as $f^{(1)}$ in the original work,

$$f^{(1)} = \sin(Z_1) + a \sin^2(Z_2) + b Z_3^4 \sin Z_1 \tag{23}$$

and the low fidelity model as $f^{(3)}$,

$$f^{(3)} = \sin(Z_1) + .6a \sin^2(Z_2) + 9b Z_3^2 \sin Z_1 \tag{24}$$

where constants $a = 5$ and $b = 0.1$. We used the same multifidelity definition as (14) and we analytically determined the global solutions for (15) for computational budget $p = 40$: $\alpha = .9455$, $m_1 = 36$, and $m_3 = 3395$ [3]. Furthermore, we numerically generated the distributions of $\hat{E}_{(\text{mf})}$, $\hat{E}_{m_1}^{(1)}$, and $\hat{E}_{m_3}^{(3)}$, where

$$\hat{E}_{(\text{mf})} = \hat{E}_{m_1}^{(1)} + \alpha(\hat{E}_{m_3}^{(3)} - \hat{E}_{m_1}^{(3)}), \tag{25}$$

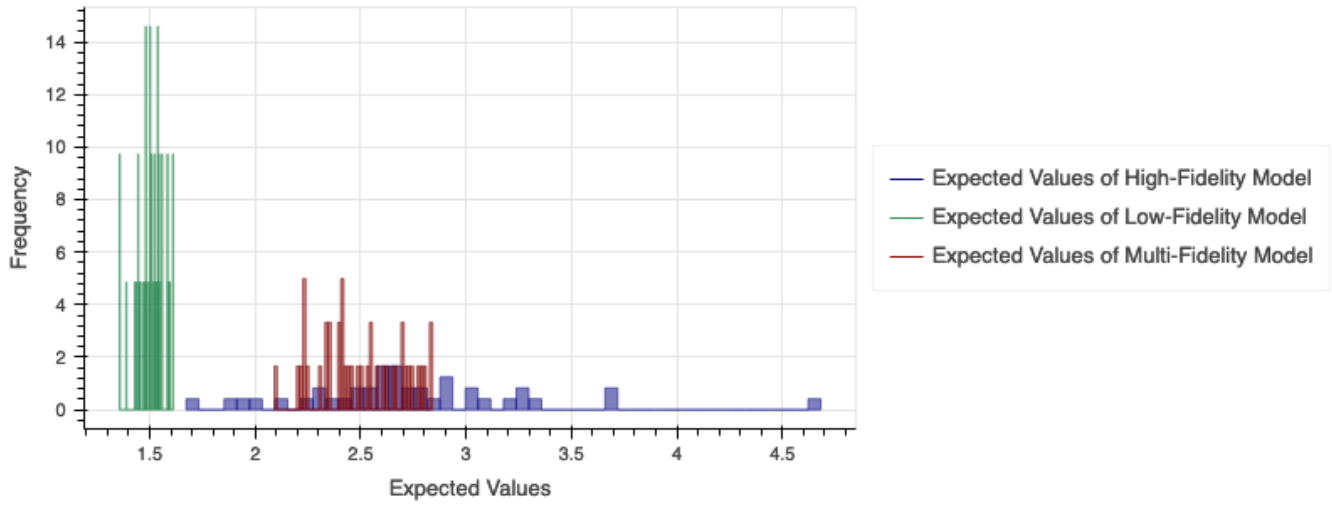

Figure 1: Distribution of expected model outputs, $\hat{E}_{(\text{mf})}$, $\hat{E}^{(1)}_{m_1}$, and $\hat{E}^{(3)}_{m_3}$, for randomly sampled Z

Table 1: Model Definitions and Statistics

models	μ_k	σ_k	m_1	m_3	MSE
$f^{(1)}$	2.73	1.23	36	-	0.291
$f^{(3)}$	1.50	0.0388	-	395	0.00378
$f_{(\text{mf})}$	2.50	0.392	36	3395	0.040
$f_{(\text{mc})} = f^{(1)}$	2.69	0.545	40	-	0.227

in Figure 1.
Based on these results, we note the MFMC estimator with variance optimization closely approximates the average output of the true model, $f^{(1)}$. Indeed, since the sample size of the low fidelity model, 9070, is much larger than the high fidelity one, 8, the distribution of $w_k^{(3)}$ is narrower than $w_k^{(1)}$'s. Furthermore, we note the following model statistics and MSE results tabulated in Table 1.

3.1.2 Gradient Descent Methods' Results

Separately, gradient descent methods were also explored for two different datasets. In one, we generated the data using $y^i = .5x^i + e^i$ for $x_i \sim \mathcal{N}(0, 1)$ and $e^i \sim \mathcal{N}(0, .1)$. In the other, we took from an ACM 213 homework problem with an unknown true function [6]. Gradient Descent, SGD, Naive SVRG, and Random SVRG were all compared for both datasets. For gradient descent, if the terminating condition, $||\nabla\mathcal{L}(w)||_2 < 10^{-4}$, was not met, the maximum number of iterations was fixed at 200 for the generated dataset and 2000 for the homework one. For the other three algorithms, if the terminating condition, $||\Delta w||_2 < 10^{-4}$, was not met, the maximum number of epochs was fixed at 200 for the generated dataset and 2000 for the homework one. All stochastic methods had a terminating condition when $||\Delta w||_2 < 10^{-4}$ because in theory, for stochastic methods $\nabla\mathcal{L}(w)$ is unknown. For SGD and SVRG methods, the internal iteration batch sizes for the number random component gradients calculated was varied. Thus, for $\eta_t = 1/t$ and $w_0 = [2.25...2.25]$, Figure 10 displays the convergence results generated for the known true function dataset and Figure 13 displays the convergence results generated for the homework dataset. The other figures in the Appendix show the difference in convergence results within the first 50 # of $\nabla\psi^i/m$.

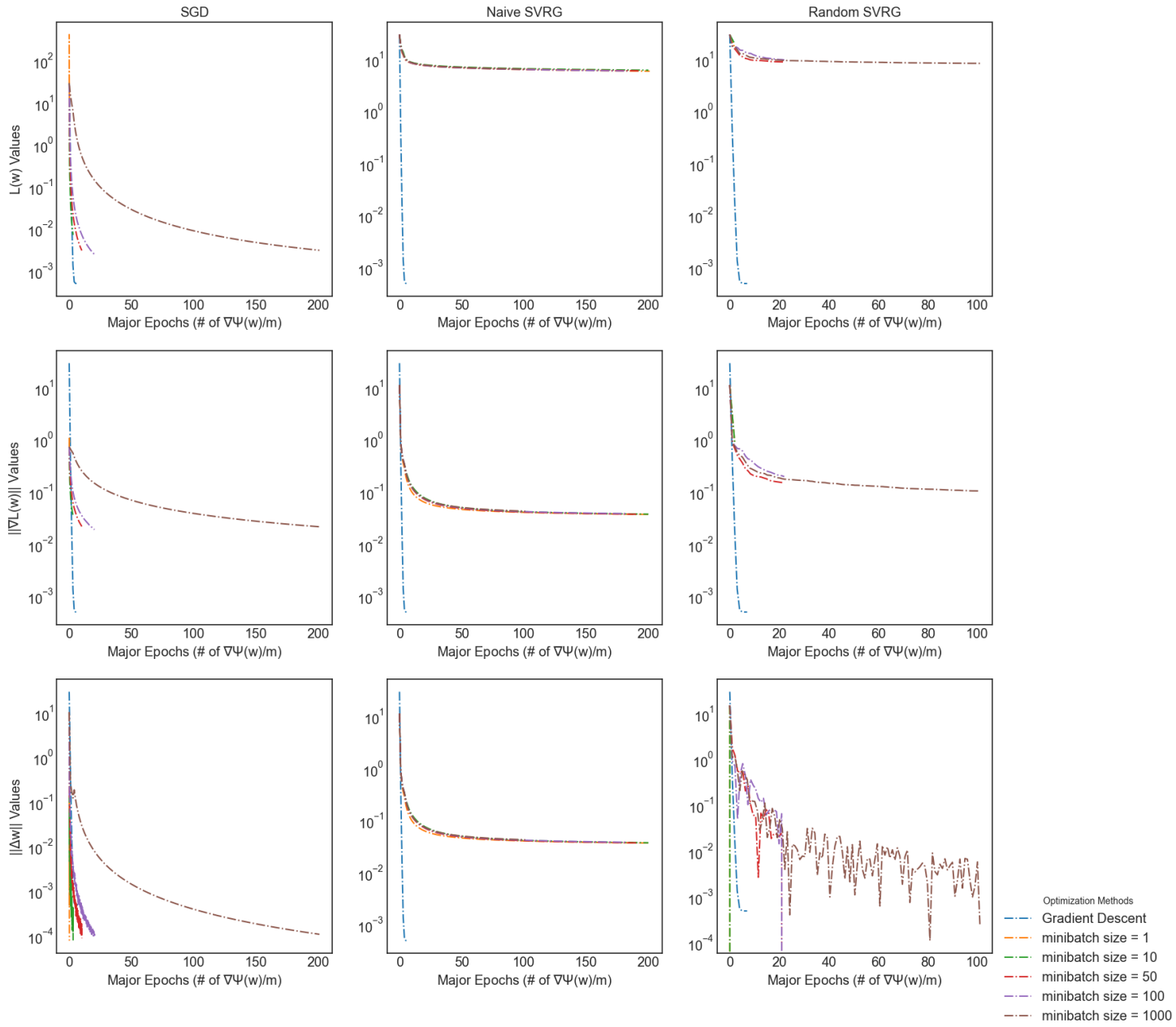

Figure 2: Comparisons of Optimization Methods for generated dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$

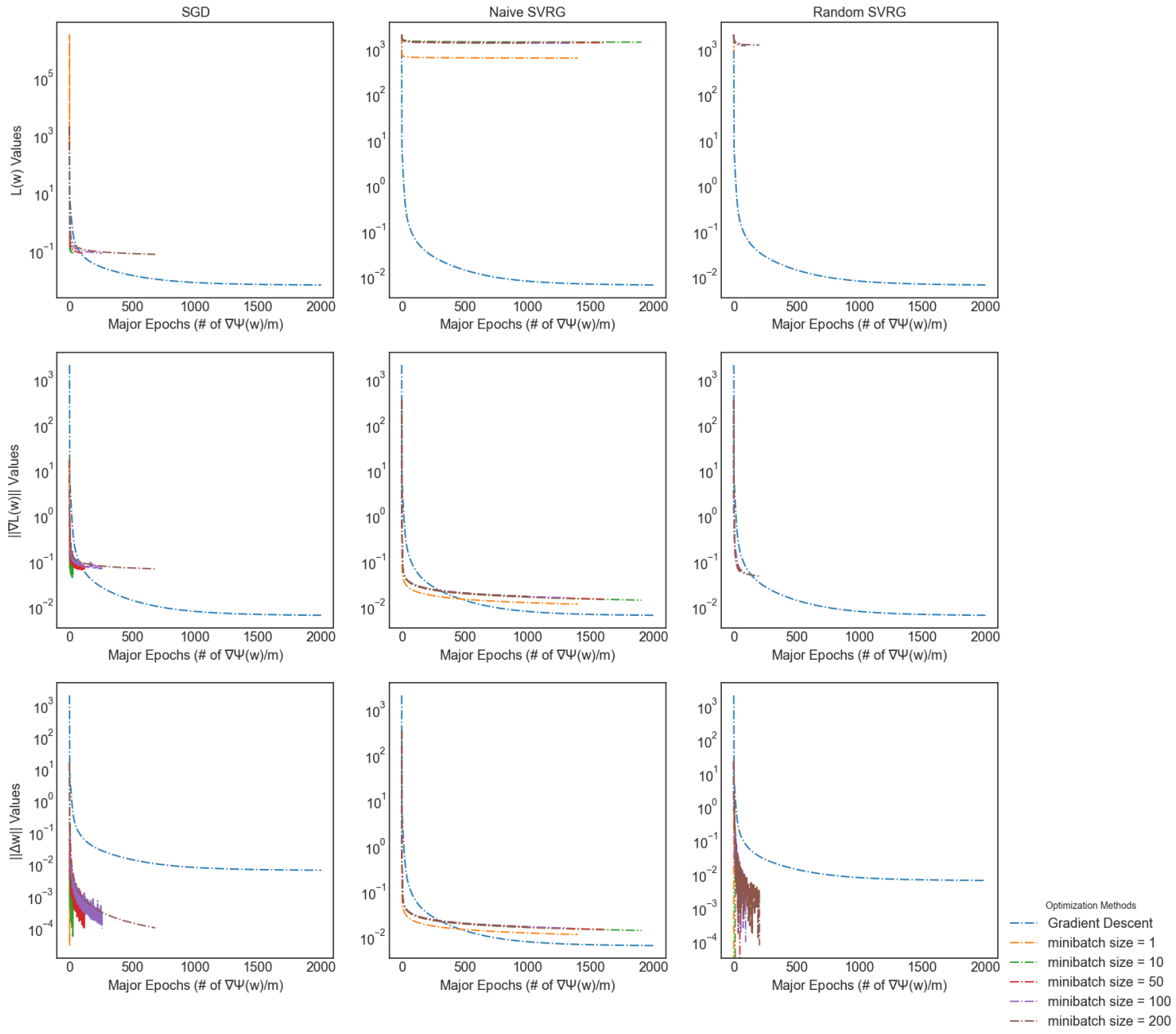

Figure 3: Comparisons of Optimization Methods for homework dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$

We note when comparing methods with equivalent batch sizes, SGD is faster than naive SVRG but slower than random SVRG. Also, we note there seems to be a tradeoff among batch sizes. In order to avoid saddle points, the batch size must be small but not too small otherwise the resulting w is extremely random. Furthermore, we note that the SVRG methods converge poorly with regards to $||\nabla\mathcal{L}(w)||$ when compared to SGD and gradient descent. However, these results were using optimization methods that all used different step size methods. For gradient descent, we used a line search method to choose the optimal step size each iteration. For SGD and SVRG, we used a decay schedule after the first three epochs, where $\eta_{t>3} = 1/s$ where s is the number of epochs. Thus, we redid our comparison with the same decay schedule where $\eta_{1-3} = 1/100$ and $\eta_{t>3} = 1/t$ where t is the outermost loop index. For gradient descent, t represents the number of iterations while for SVRG and SGD, t is the number of epochs. We defined SVRG as random SVRG since it had theoretically [1] and empirically better results as seen in Figure 10. Furthermore, we focused on the simpler, generated dataset for more intuitive results.

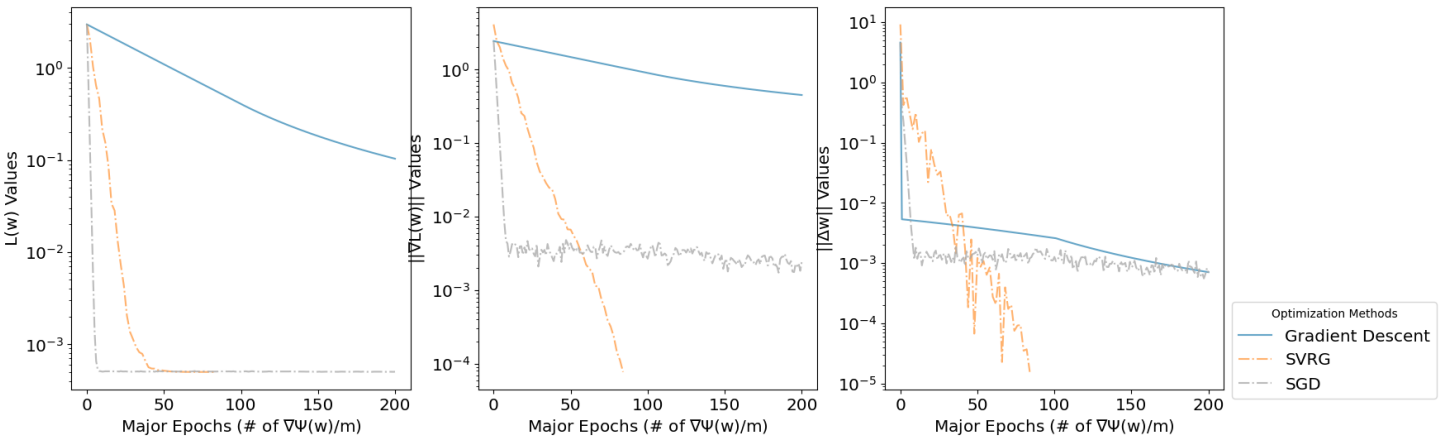

Figure 4: Comparisons of Optimization Methods for generated dataset with same stepsize schedule using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$

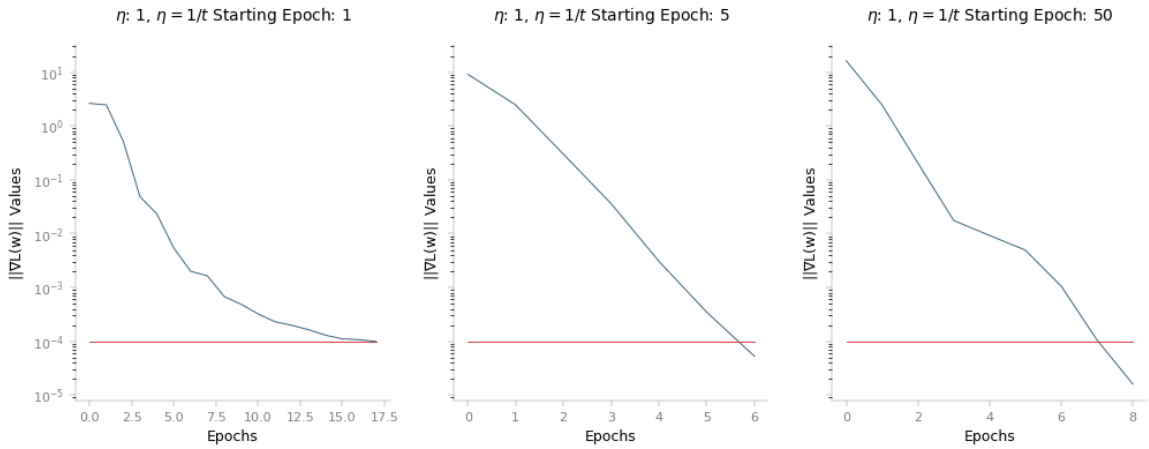

Figure 5: Effect of decay scheduling on $||\nabla\mathcal{L}(w)||$ for SVRG with initial fixed value $\eta_t = 1$ (blue). We note that decaying η_t too early (left plot) slows down convergence to below the convergence line $||\nabla\mathcal{L}(w)|| < \tau = 10^{-4}$ (orange). Furthermore, there seems to be limited improvement when comparing the later decay schedule (middle plot) with the fully fixed η_t (right plot).

We noted for these smaller step sizes, gradient descent performed worse in comparison to SVRG and SGD. SVRG performed worse initially wrt all metrics compared to SGD; however, it converged faster than SGD to lower $||\nabla\mathcal{L}(w)||$ and $||\Delta w||$ values as seen in 4. To be more precise about the seeming tradeoff in convergence results between initial fixed step size and beginning decay schedule, we varied the initial step sizes among $\tilde{\eta} = [1, .1, ..., 10^{-5}]$ and the scheduled decay epoch/iteration among $\tilde{t} = [1, 5, ..., 100]$ in Appendix 5.2. In particular, we found for SVRG, we found for larger initial η_t, such as $\eta_t = 1$ or $\eta_t = .1$, having having some decay schedule improved convergence results wrt $||\nabla\mathcal{L}(w)||$. As shown in Figure 5, we found for $\eta_t = 1$, keeping the step size fixed for the initial epochs (right two columns) resulted in a faster convergence compared to immediately decaying η_t (leftmost column). Additionally, adding any sort of decay schedule provided limited to no improvement regardless of initial η_t as seen in Figure 6. When comparing optimal schedules for all four methods, we note that multifidelity SVRG outperforms all other stochastic methods by over 50%; however, it performs slightly worse than gradient descent. Further testing is needed to determine the significance of these results and the effectiveness of multifidelity SVRG on other

convex problems.

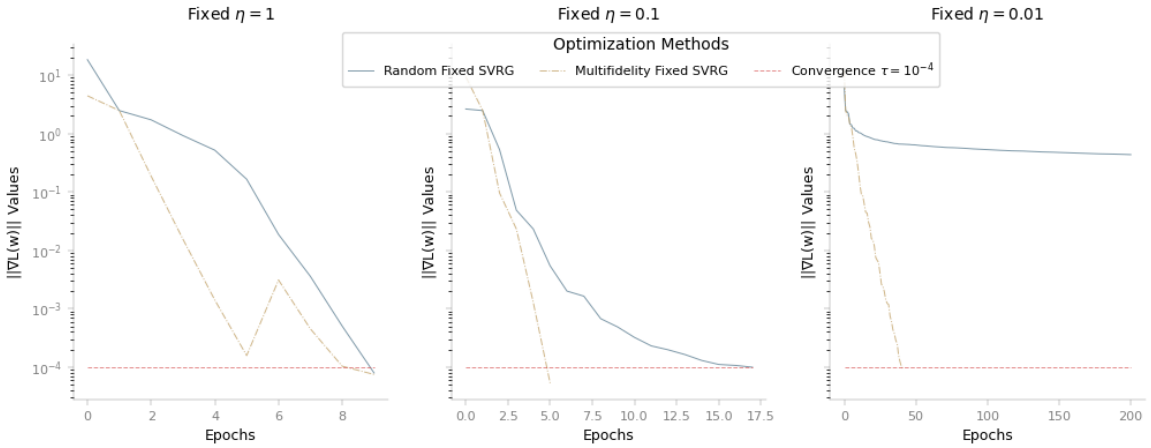

Figure 6: Effect of step size on $||\nabla\mathcal{L}(w)||$ for Random SVRG (blue) with fixed η_t and Multifidelity SVRG (yellow). we note as we decrease the initial fixed value $\eta_t = 0.1$ to $\eta_t = 0.01$, both methods take more epochs to converge. Additionally, regardless of the step size, the multifidelity method consistently converges below the convergence line $||\nabla\mathcal{L}(w)|| < \tau = 10^{-4}$ (red) in fewer epochs than the fixed method.

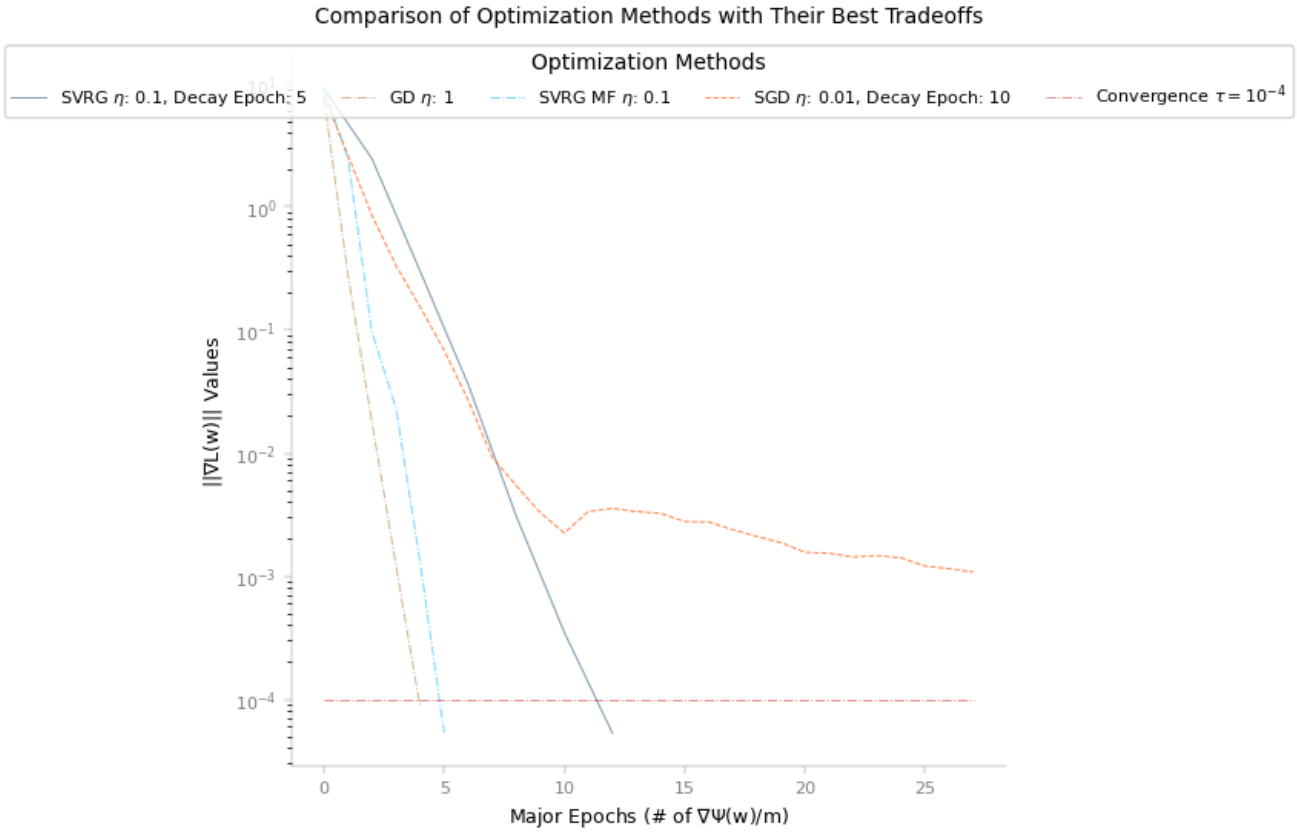

Figure 7: Comparison of all Gradient Descent methods using $||\nabla\mathcal{L}(w)||$.

Thus, we have accomplished some of the main objectives discussed in the SURF proposal. We have studied numerical examples for multifidelity and gradient descent methods. We have implemented existing models and compared performance using gaussian regression problems. Furthermore, we have determined potential performance gains using multifidelity SVRG.

References

[1] R. Johnson and T. Zhang,"Accelerating Stochastic Gradient Descent using Predictive Variance Reduction," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[2] B. Peherstorfer, K. Willcox, and M. Gunzburger,"Optimal Model Management for Multifidelity Monte Carlo Estimation," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A3163–A3194, Oct. 2016.

[3] E. Qian, B. Peherstorfer, D. O'Malley, V. V. Vesselinov, and K. Willcox, "Multifidelity Monte Carlo estimation of variance and sensitivity indices," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 6, no. 2, pp. 683–706, 2018.

[4] A. V. Srinivasan, "Stochastic Gradient Descent — Clearly Explained !!," *Towards Data Science*, 06-Sep-2019. .

[5] B. H, "Stochastic Variance Reduced Gradient (SVRG)," *Bingo.H's blog*, 08-Jun-2017. .

[6] Machine Learning Course Materials by Various Authors, including David Rosenberg, available at https://github.com/davidrosenberg/mlcourse and licensed under the Creative Commons Attribution 4.0 license, which allows sharing and adapting with appropriate attribution.

[7] B. O'Connor, "Cosine similarity, Pearson correlation, and OLS coefficients," *AI and Social Science*, 2012. .

[8] S. Borgatti, "Distance and Correlation," Measures of similarity and distance, 2013. [Online]. Available: https://cmci.colorado.edu/classes/INFO-1301/files/borgatti.htm#:~:text=Euclidean%20distance%20is%20only%20appropriate,standardized%20versions%20of%20the%20data. .

4 Acknowledgements

I acknowledge and thank Elizabeth Qian for her mentorship and guidance throughout the project. I thank Pan Xu for his assistance in explaining stochastic gradient descent methods. I acknowledge and thank SURF donors for the Arthur R. Adams SURF Fellowship.

5 Appendix

5.1 Gradient Descent Methods' Initial Results

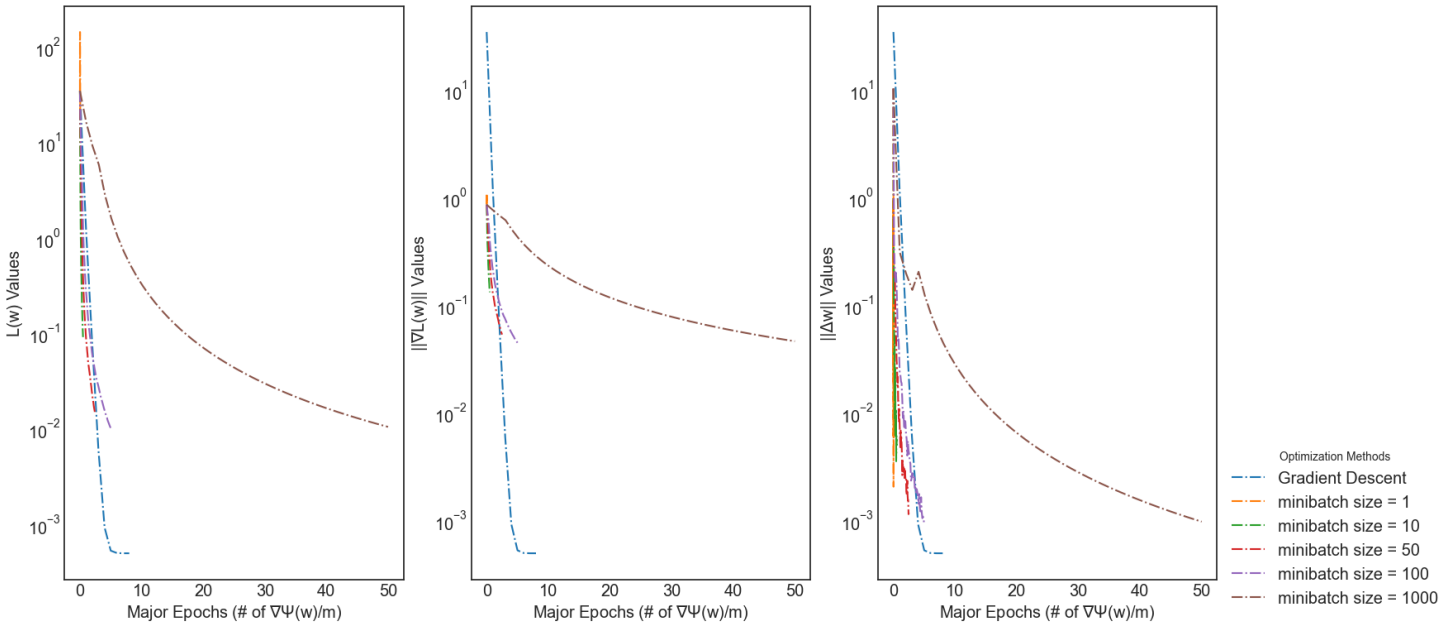

Figure 8: Comparisons of SGD performance at different batch sizes for generated dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show SGD converges slightly slower than Gradient Descent. We also note as the batch size increases, SGD converges in a smoother, slower fashion.

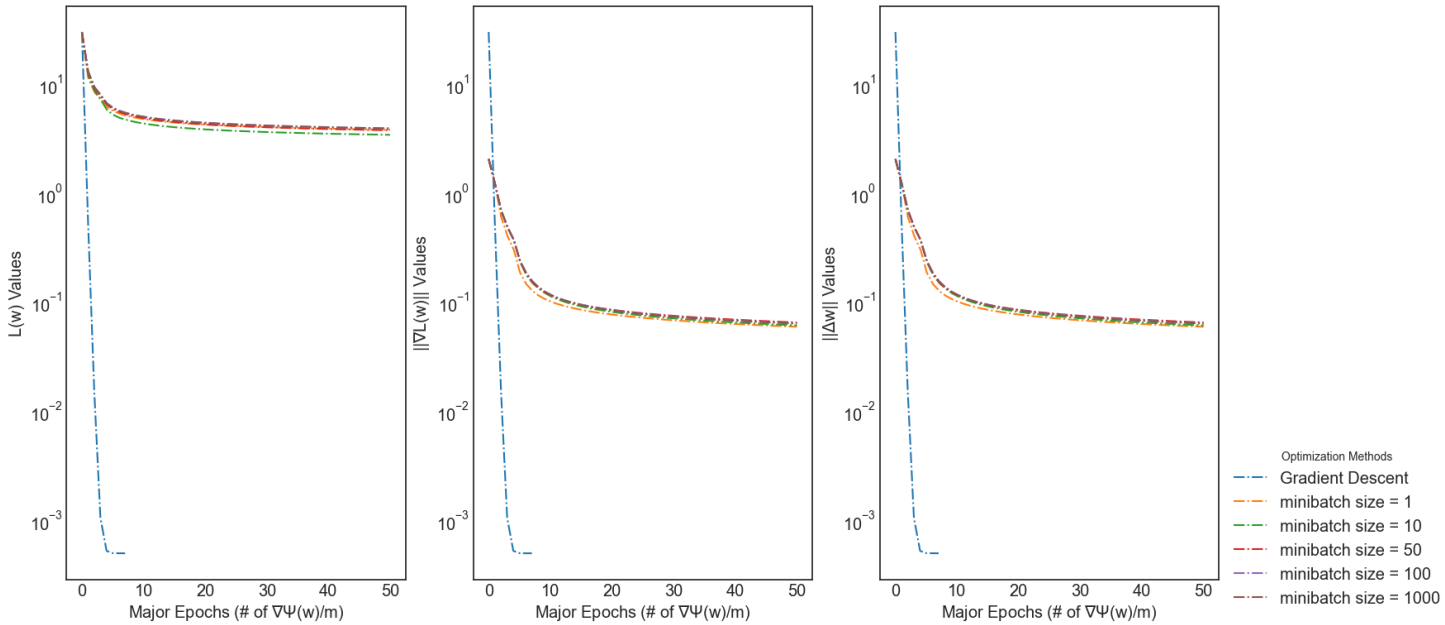

Figure 9: Comparisons of Naive SVRG performance at different batch sizes for generated dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show Naive SVRG converges slower than Gradient Descent. We also note there is no visible difference in convergence as batch size increases.

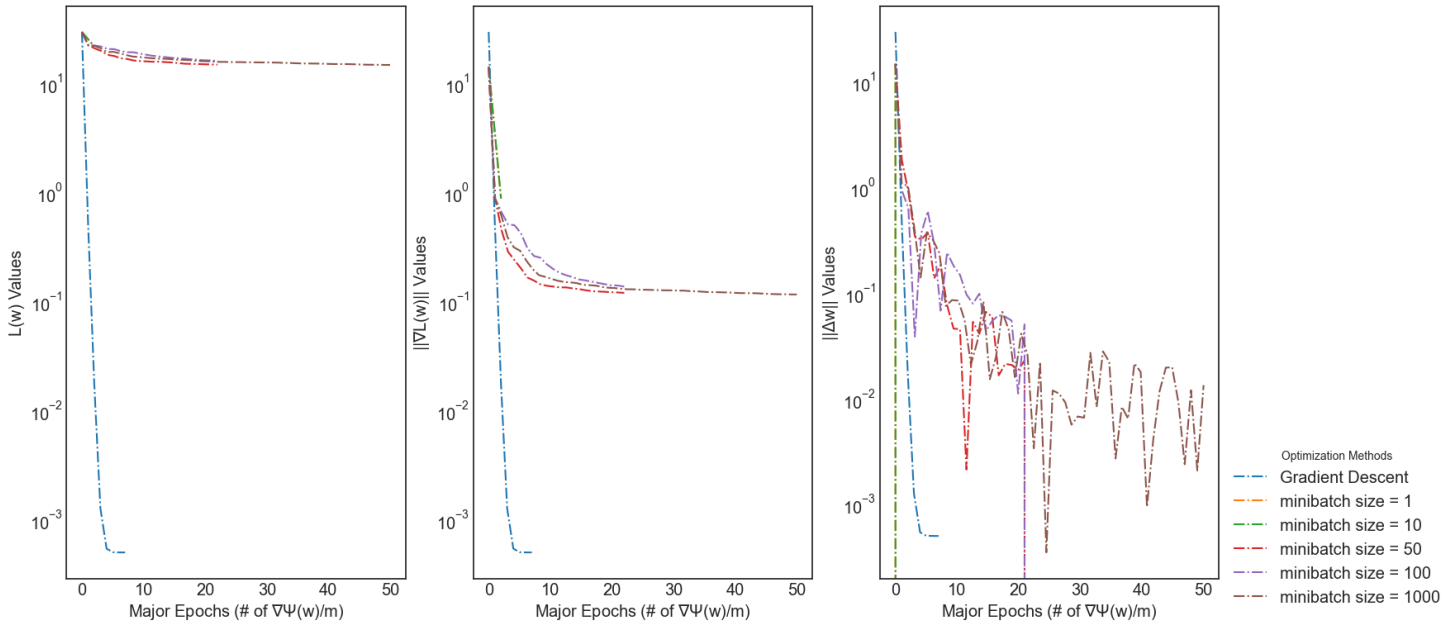

Figure 10: Comparisons of Random SVRG performance at different batch sizes for generated dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show Random SVRG converges slower than Gradient Descent. We also note as the batch size increases, Random SVRG converges in a smoother, slower fashion.

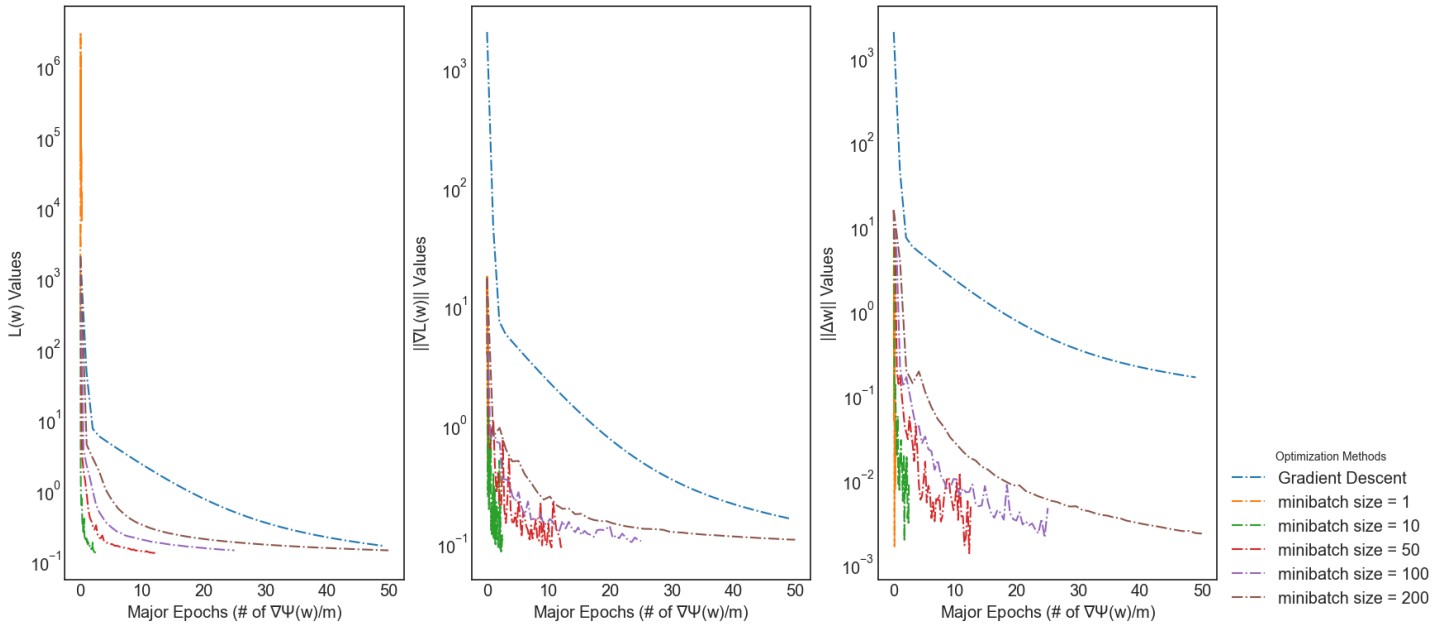

Figure 11: Comparisons of SGD performance at different batch sizes for the homework dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show SGD converges faster than Gradient Descent in this case. We also note as the batch size increases, SGD converges in a smoother, slower fashion.

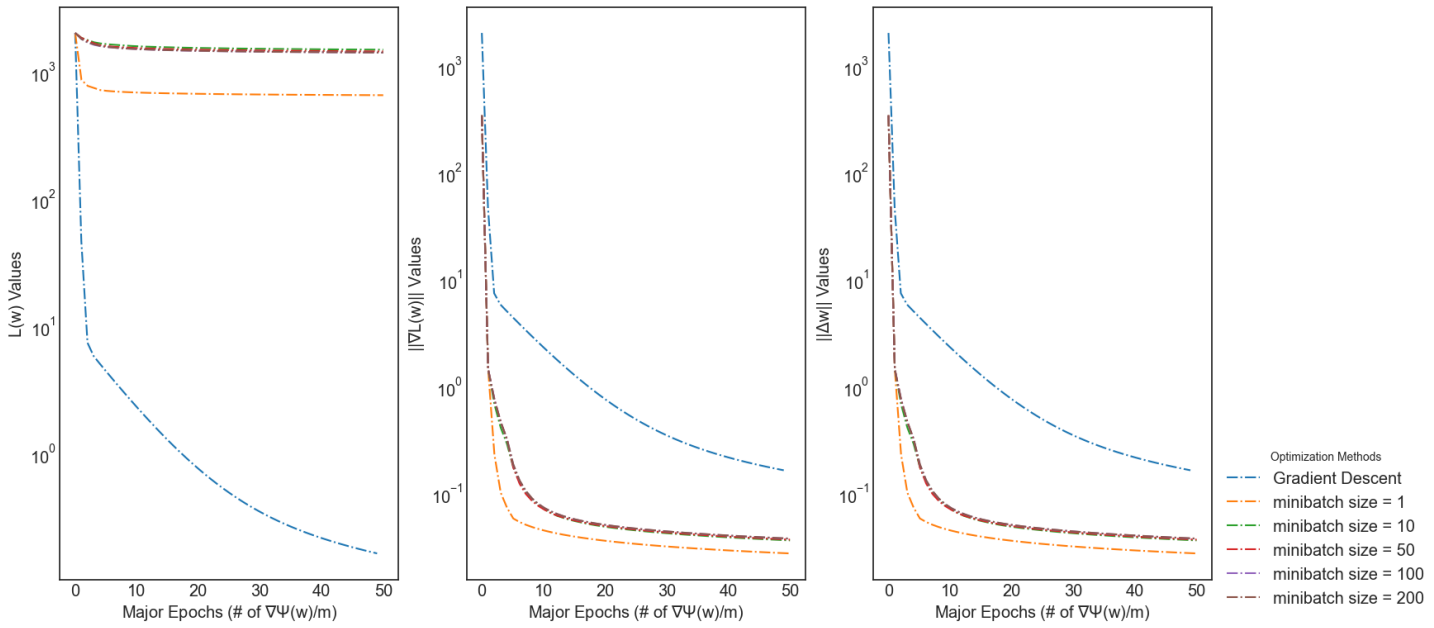

Figure 12: Comparisons of Naive SVRG performance at different batch sizes for homework dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show Naive SVRG converges faster than Gradient Descent except for $\mathcal{L}(w)$. We also note there is no visible difference in convergence as the batch size increases. However, there seems to be a slightly faster convergence when the batch size is 1 for the Naive SVRG method.

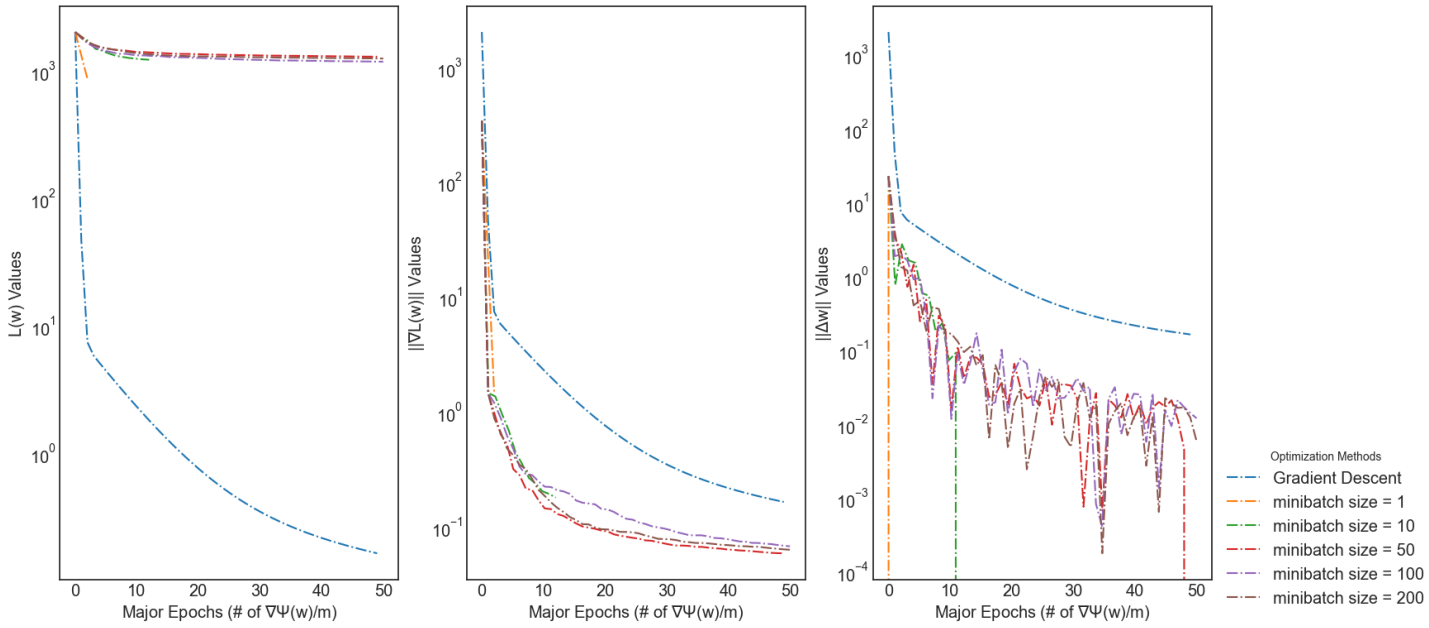

Figure 13: Comparisons of Random SVRG performance at different batch sizes for homework dataset using $\mathcal{L}(w)$, $||\nabla\mathcal{L}(w)||$, and $||\Delta w||$. Particularly, we focused on the first 50 # of $\nabla\psi^i/m$ in this plot in order to show Random SVRG converges faster than Gradient Descent except for $\mathcal{L}(w)$. We also note there is no visible difference in convergence as the batch size increases.

5.2 Full Tradeoff Results Between Fixed Step Size and Decaying Step Size

Initially, we wanted to examine how choosing when to decay $\eta_t = 1/t$ from a fixed value $\eta_t \in \{1, .01, ...\}$ affects convergence for wrt $\mathcal{L}(w)$ and $||\nabla\mathcal{L}(w)||$. We set the We keep the same convergence condition where $||\nabla\mathcal{L}(w)|| < \tau = 10^{-4}$ must be true. We examined each gradient descent method individually. For Gradient Descent and SGD, we examined initially fixing then varying when we decay $\eta_t = 1/t$ where t is the number of iterations or epochs respectively. For SVRG, we focused on

- varying fixed η_t,
- fixed η_t and varying when we decay $\eta_t = 1/t$ where t is the number of epochs, and
- fixed η_t and varying when we decay $\eta_t = 1/\sqrt{t}$ where t is the number of epochs.

For the sake of comparison, we adjusted the algorithms as follows:

Algorithm 1: Gradient Descent

Input: number of iterations T, initial step size $\tilde{\eta}$, decay start $\tilde{t}$, gradient loss function $\mathcal{L}$, initial estimator $w_t = w_0$, tolerance τ

$t = 0$
while $||\nabla\mathcal{L}(w_t)|| > \tau \ \ \& \ \ t < T$ **do**
  $p_t = -\nabla\mathcal{L}(w_t)$
  if $t \leq \tilde{t}$ **then**
    $\eta_t = \tilde{\eta}$
  else
    $\eta_t = \frac{\tilde{\eta}}{t-\tilde{t}}$
  $w_t = w_{t-1} + \eta_t p_t$
  $t = t + 1$
return w_t

Algorithm 2: Stochastic Gradient Descent (SGD)

Input: number of epochs S, minibatch size B, minibatch iterations M, initial step size $\tilde{\eta}$, decay start $\tilde{t}$, gradient component loss functions $\nabla\psi$, initial estimator $w_t = w_0$, tolerance τ

$t = 0$
while $||w_t - w_{t-1}|| > \tau \ \ \& \ \ t < S$ **do**
  if $t \leq \tilde{t}$ **then**
    $\eta_t = \tilde{\eta}$
  else
    $\eta_t = \frac{\tilde{\eta}}{t-\tilde{t}}$
  for $m \leftarrow 1$ **to** M **do**
    Randomly sample B number of component indices I_t
    $w_t = w_t - \frac{\eta_t}{B} \sum_{i_t=0}^{I_t} \nabla\psi^{i_t}(w_t)$
  $w_{t-1} = w_t$
  $t = t + 1$
return w_t

Algorithm 3: Stochastic Variance Reduced Gradient (SVRG)

Input: number of epochs S, epoch size B_s^{large}, minibatch size B_t^{small}, minibatch iterations M, initial step size $\tilde{\eta}$, decay start $\tilde{s}$, gradient component loss functions $\nabla\psi$, initial estimator $w_s = w_t = w_0$, tolerance τ

3.1 $s = 0$
3.2 **while** $||w_s - w_{s-1}|| > \tau$ & $s < S$ **do**
3.3 **if** $s \leq \tilde{s}$ **then**
3.4 $\eta_t = \tilde{\eta}$
3.5 **else**
3.6 $\eta_t = \frac{\tilde{\eta}}{s - \tilde{s}}$
3.7 Randomly sample B_s^{large} number of component indices I
3.8 $\nabla\tilde{\mu} = \frac{1}{B_s^{large}} \sum_{i=0}^{I} \nabla\psi^i(w_s)$
3.9 $w_{t-1} = w_s$
3.10 **for** $m \leftarrow 1$ **to** M **do**
3.11 Randomly sample B_t^{small} number of component indices I_t
3.12 $u_t = \nabla\tilde{\mu} + \frac{1}{B_t^{small}} \sum_{i_t=0}^{I_t} \nabla\psi^{i_t}(w_{t-1}) - \frac{1}{B_t^{small}} \sum_{i_t=0}^{I_t} \nabla\psi^{i_t}(w_s)$
3.13 $w_t = w_{t-1} - \eta_t u_t$
3.14 Randomly select a $\tilde{w}_t$ from inner loop
3.15 $w_s = \tilde{w}_t$
3.16 $s = s + 1$
3.17 **return** w_s

Algorithm 4: Stochastic Variance Reduced Gradient (SVRG) with Multifidelity Principles

Input: number of epochs S, cost of "higher" function/epoch size B_s^{large}, cost of "lower" function/minibatch size B_t^{small}, initial step size $\tilde{\eta}$, decay start $\tilde{s}$, gradient component loss functions $\nabla\psi$, initial estimator $w_s = w_t = w_0$, tolerance τ, computational budget p

4.1 $s = 0$
4.2 **while** $||w_s - w_{s-1}|| > \tau$ & $s < S$ **do**
4.3 Randomly sample B_s^{large} number of component indices I
4.4 $\nabla\tilde{\mu} = \frac{1}{B_s^{large}} \sum_{i=0}^{I} \nabla\psi^i(w_s)$
4.5 Calculate $\rho_{1,2} = \cos(\theta) = \frac{\langle \nabla\mu\, , \nabla\tilde{\mu}\, \rangle}{||\nabla\mu|| \cdot ||\nabla\tilde{\mu}||}$
4.6 Calculate $r_1 = 1$ and $r_2 = \sqrt{\frac{B_s^{large}\rho_{1,2}}{B_t^{small}(1-\rho_{1,2})}}$
4.7 Calculate $m = [m_1, m_2]$
4.8 Calculate $\alpha = [\alpha_2]$ **if** $s \leq \tilde{s}$ **then**
4.9 $\eta_t = \tilde{\eta}$
4.10 **else**
4.11 $\eta_t = \alpha_2$
4.12 $w_{t-1} = w_s$
4.13 **for** $j \leftarrow 1$ **to** m_2 **do**
4.14 Randomly sample B_t^{small} number of component indices I_t
4.15 $u_t = \nabla\tilde{\mu} + \frac{1}{B_t^{small}} \sum_{i_t=0}^{I_t} \nabla\psi^{i_t}(w_{t-1}) - \frac{1}{B_t^{small}} \sum_{i_t=0}^{I_t} \nabla\psi^{i_t}(w_s)$
4.16 $w_t = w_{t-1} - \eta_t u_t$
4.17 Randomly select a $\tilde{w}_t$ from inner loop
4.18 $w_s = \tilde{w}_t$
4.19 $s = s + 1$
4.20 **return** w_s

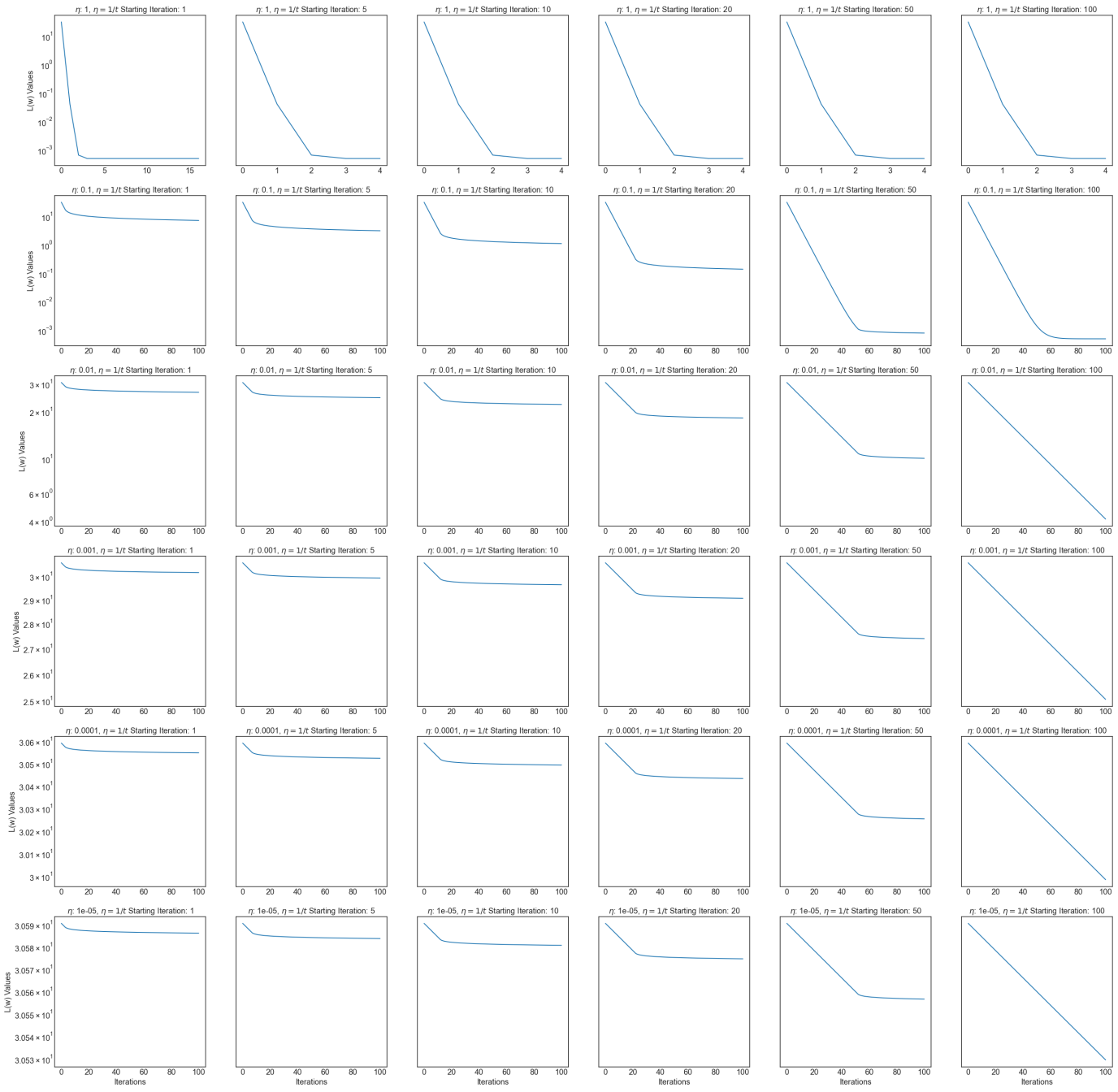

Figure 14: η_t Tradeoff analysis for Gradient Descent using $\mathcal{L}(w)$. We note that for $\eta_t = 1$, the estimator rapidly converges within four iterations, so no decay schedule is necessary. Aside from $\eta_t = 1$, we note performance worsens as fixed η_t decreases (seen in all columns) and improves when we do not decay η_t at all (seen in the final column). This result shows that decaying an already small η_t worsens performance.

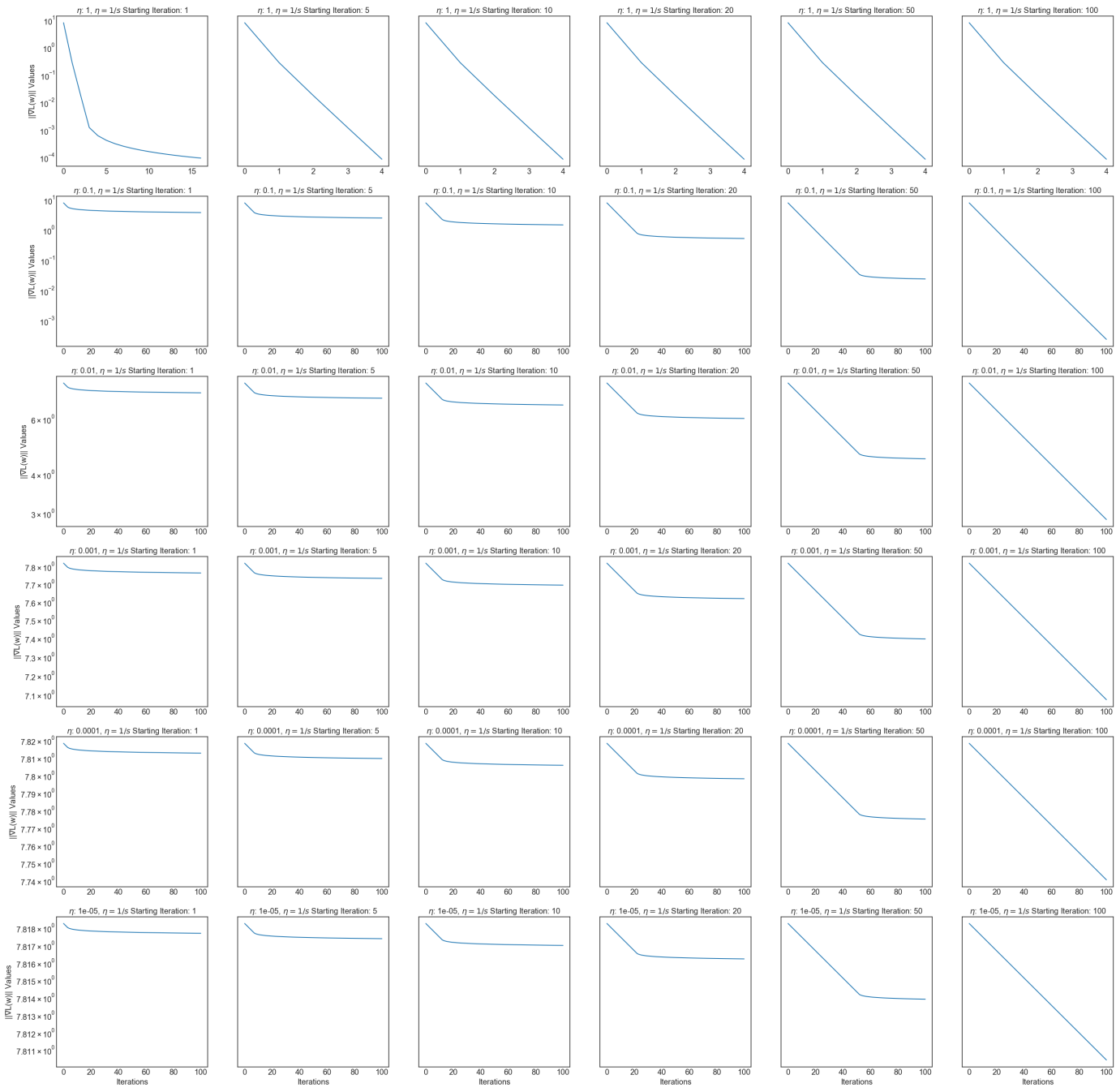

Figure 15: η_t Tradeoff analysis for Gradient Descent using $||\nabla\mathcal{L}(w)||$. We note results consistent with Figure 14. Furthermore, we see a pattern of improvement as we increase the scheduled decay iteration (seen in all but the first rows).

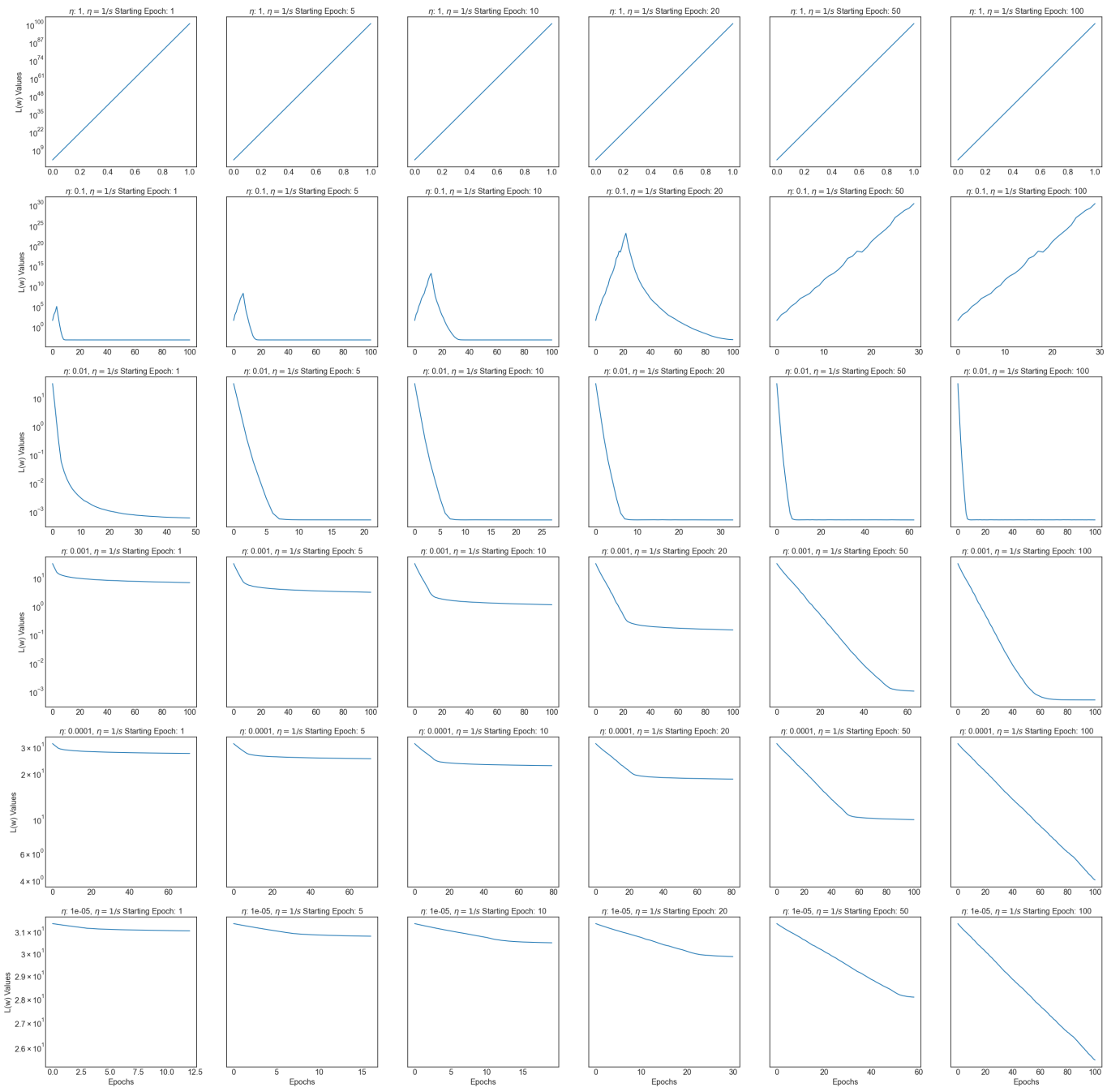

Figure 16: η_t Tradeoff analysis for SGD using $\mathcal{L}(w)$. We note that for $\eta_t = 1$, the estimator rapidly overflows within one epoch, so no decay schedule is shown. Furthermore, SGD is sensitive to the initial fixed η_t values because for larger $\eta_t = 1, \eta_t = .01$, we note the estimator overflows without any decay schedule (rows 1 and 2). However, for smaller $\eta_t = .001, .0001$, SGD is able to converge without any decay schedule (last column, rows 3 and 4). Similar to Gradient Descent, we note that the SGD estimator is unable to converge for too small $\eta_t = 10^{-5}$. For most fixed η_t, the SGD estimator minimizes $\mathcal{L}(w)$ faster when we do not decay η_t at all (seen in the final column). However, in the case of $\eta_t = .001$, some decay actually produced the best convergence results (row 2 column 2). This result shows that decaying for particular fixed η_t values improves performance.

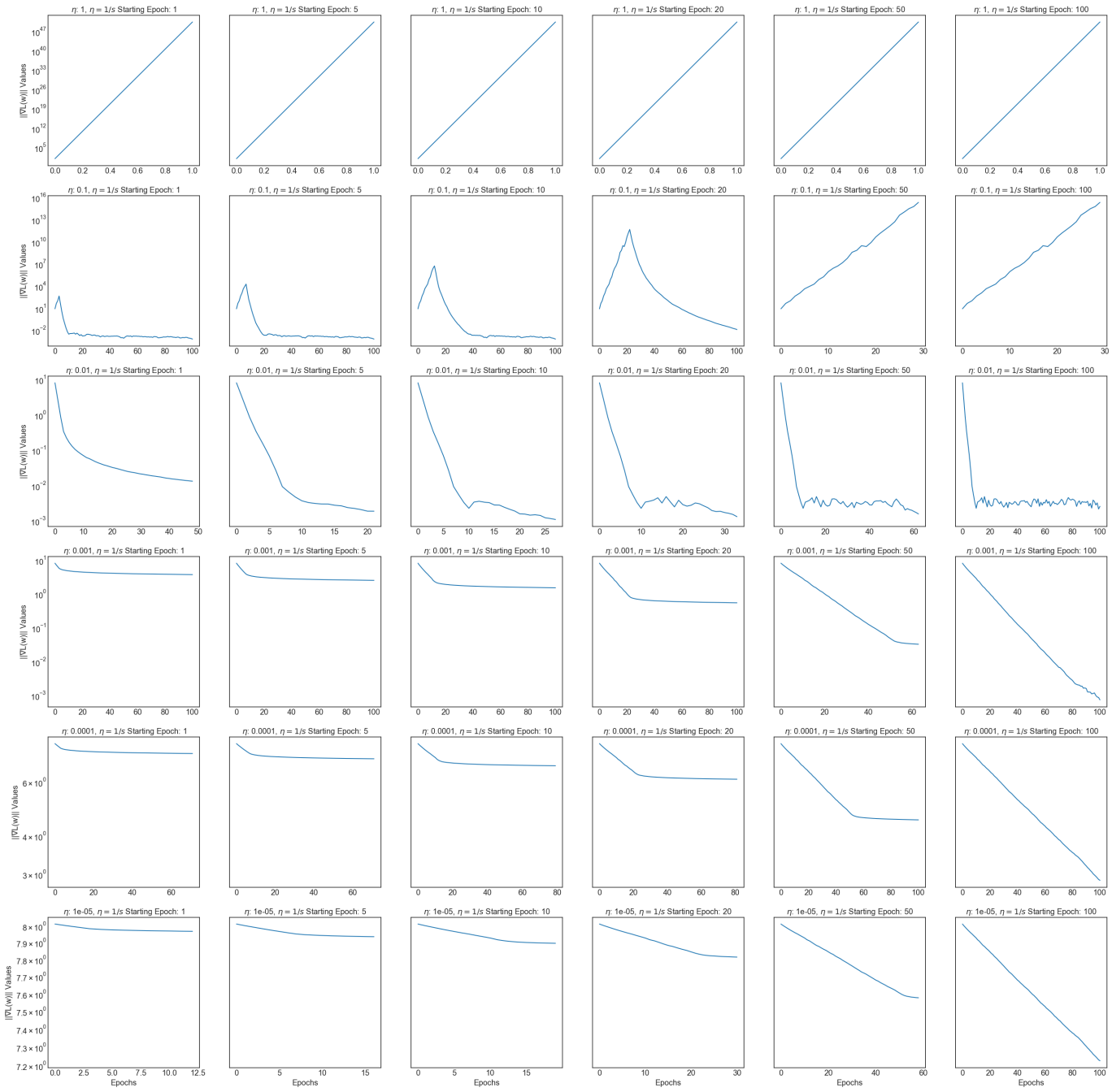

Figure 17: η_t Tradeoff analysis for SGD using $||\nabla\mathcal{L}(w)||$. We note results consistent with Figure 16. However, we note $\eta_t = .0001$ reduces $||\nabla\mathcal{L}(w)||$ at a much slower rate than $\eta_t = .001$.

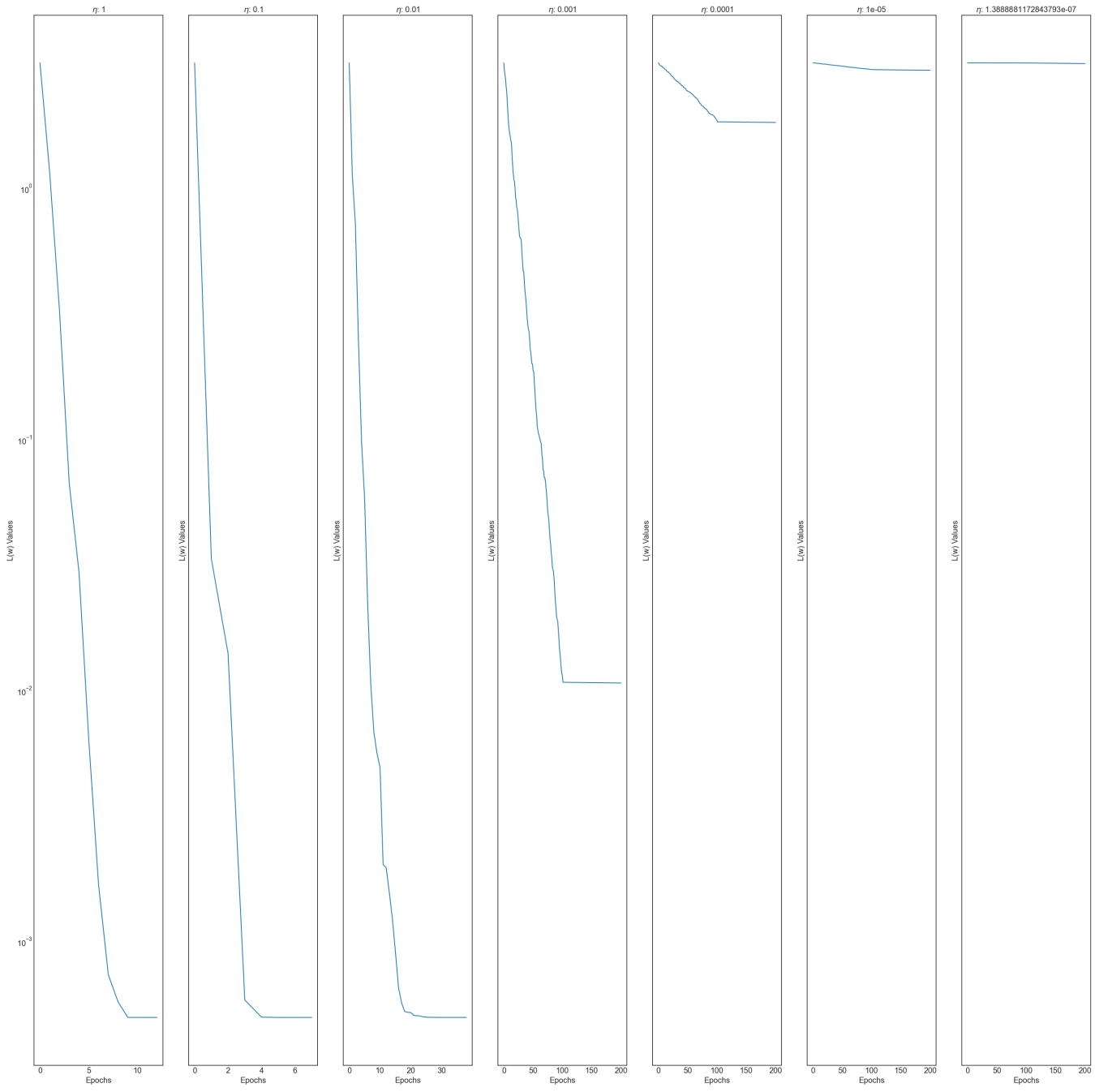

Figure 18: Fixed η_t analysis for SVRG using $\mathcal{L}(w)$. We note similar to Gradient Descent, SVRG converges only for larger η_t, with the best results at $\eta_t = 1$ (20 epochs). Furthermore, we note performance worsens as fixed η_t decreases.

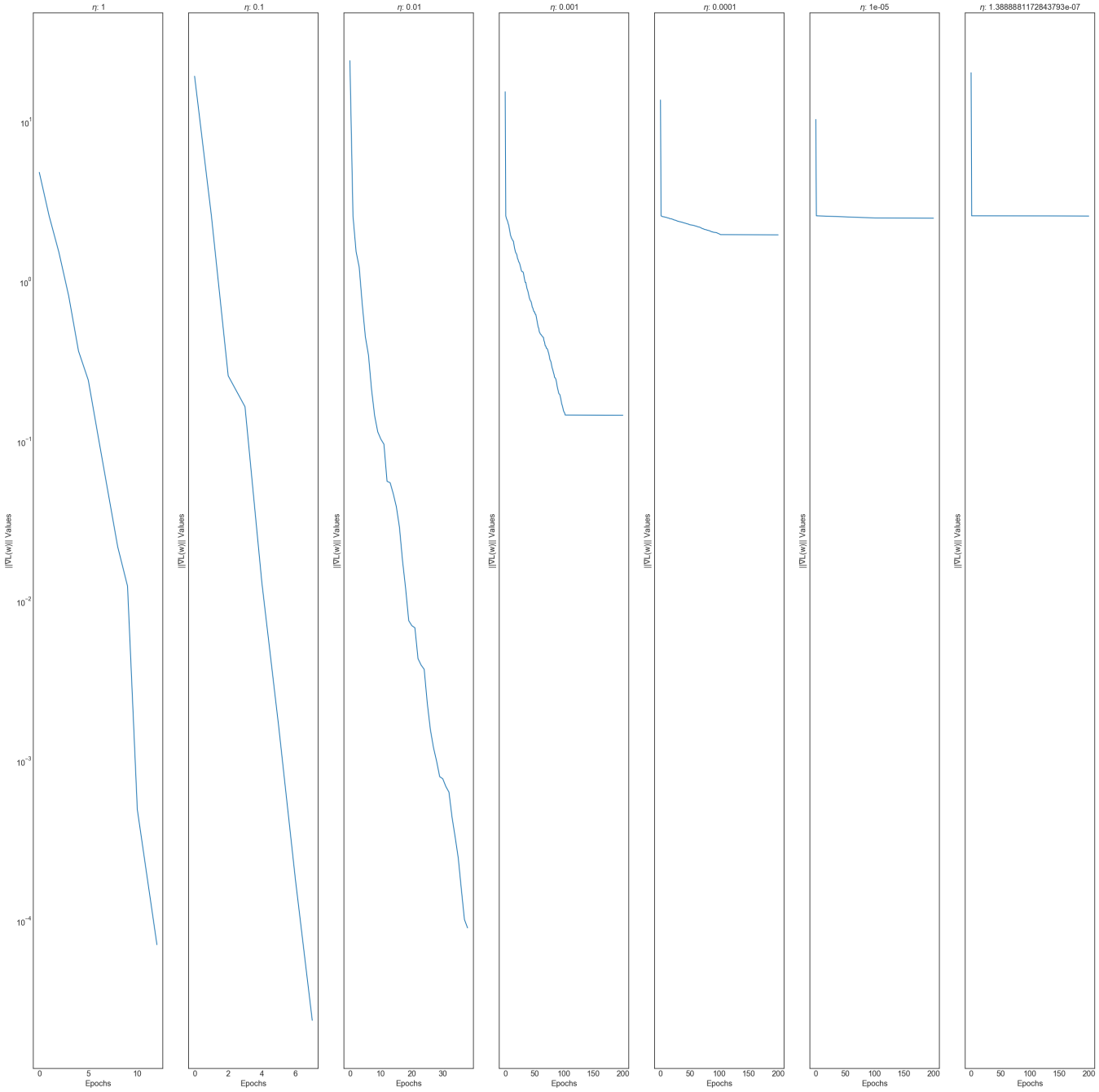

Figure 19: Fixed η_t analysis for SVRG using $||\nabla \mathcal{L}(w)||$. We note results consistent with Figure 18

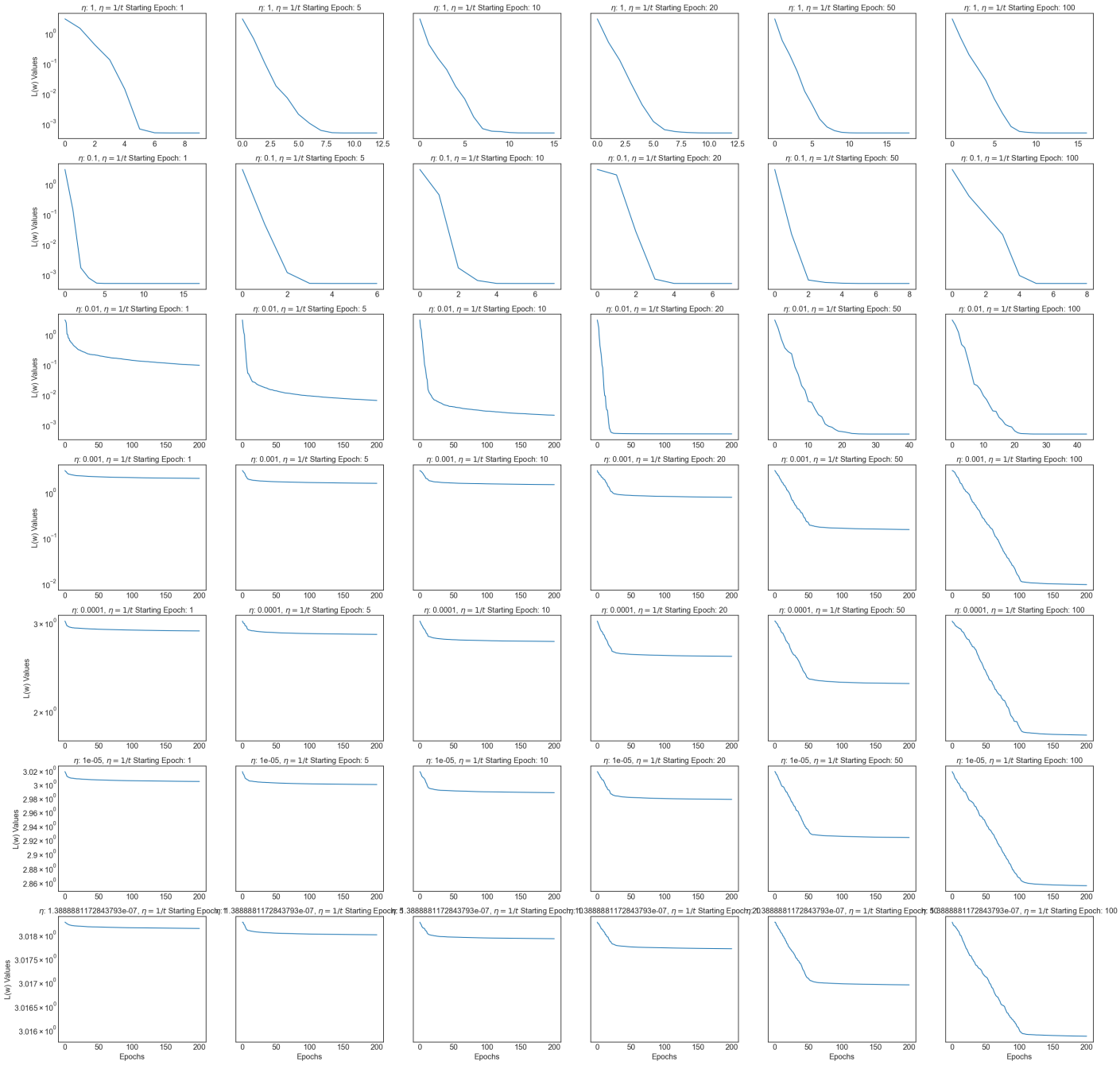

Figure 20: η_t Tradeoff analysis for SVRG using $\mathcal{L}(w)$. We note similar to Gradient Descent, SVRG converges only for larger η_t, with the best results at $\eta_t = .1$ (6 epochs). Furthermore, we note performance worsens as fixed η_t decreases regardless of when we start the decay schedule (see all columns). Similar to other methods, $\mathcal{L}(w)$ is minimized faster when we do not decay η_t at all (seen in the final column). However, in the case of $\eta_t = 1$, $\mathcal{L}(w)$ is minimized faster when we decay η_t after the first epoch (seen in the first row and first column). This result shows that decaying for particular fixed η_t values improves performance for SVRG.

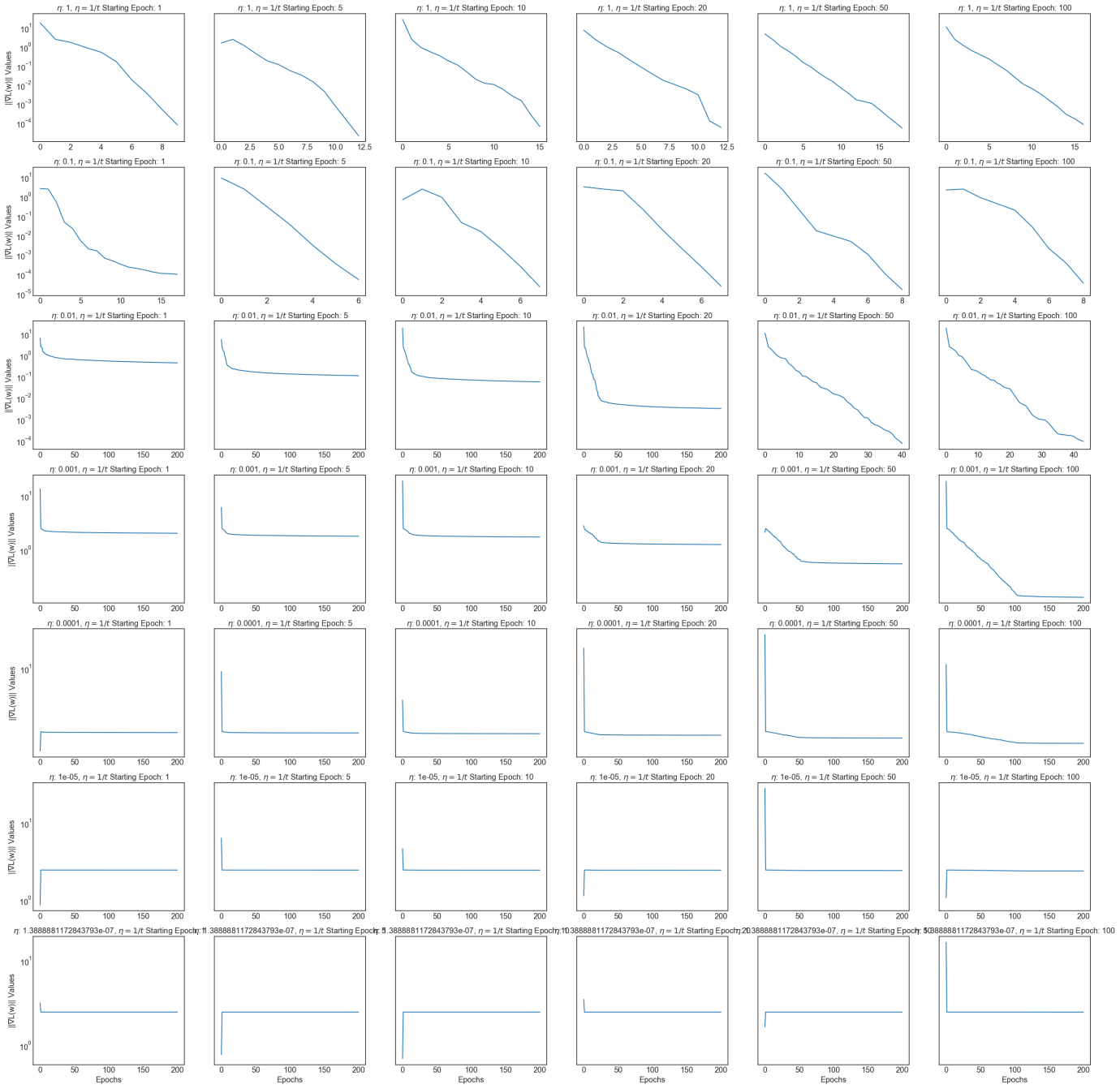

Figure 21: η_t Tradeoff analysis for SVRG using $||\nabla\mathcal{L}(w)||$. We note results consistent with Figure 20. We note the longer we keep η_t fixed, $||\nabla\mathcal{L}(w)||$ decreases longer at a linear rate (see first and last columns).

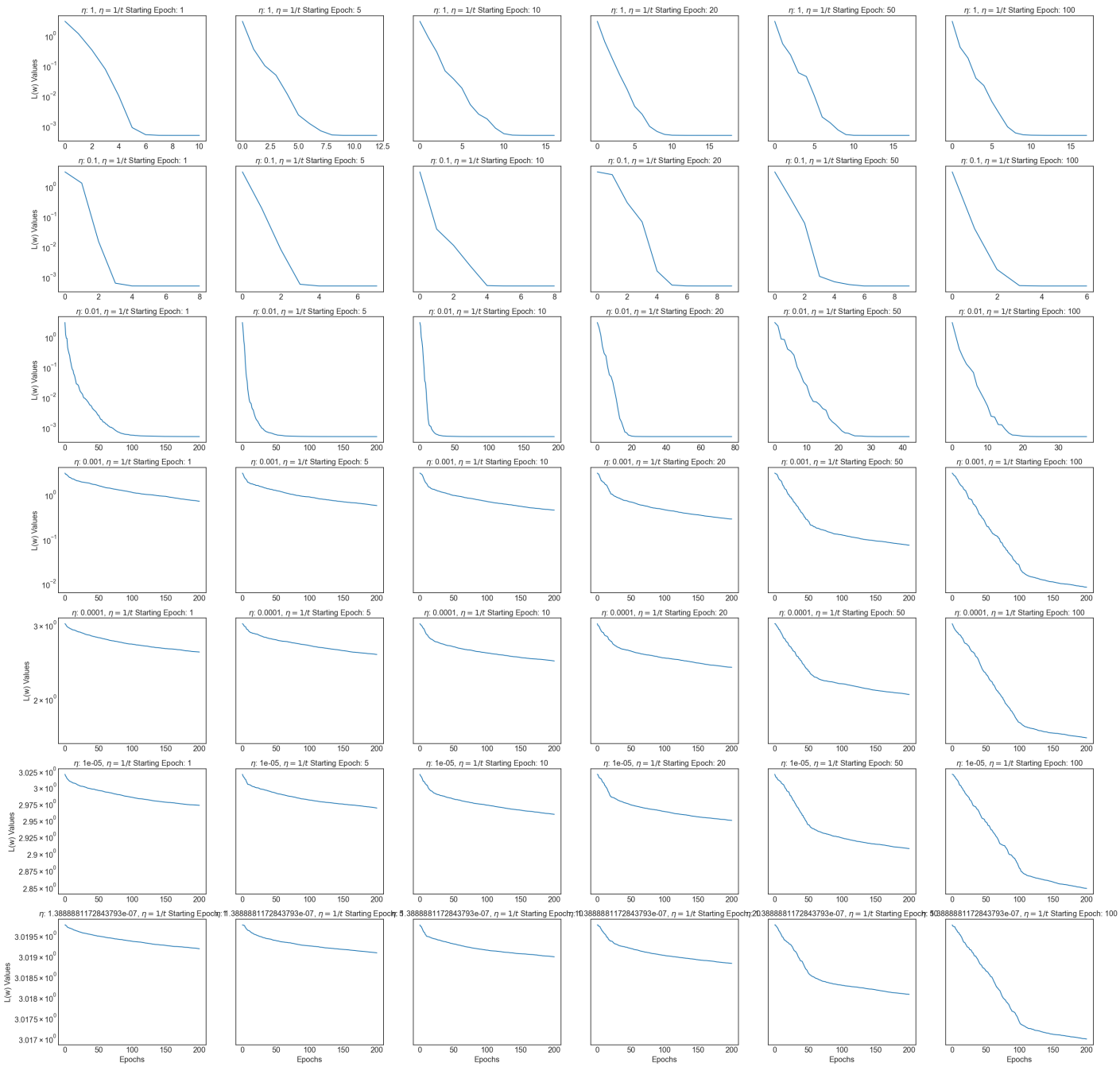

Figure 22: η_t Tradeoff analysis for SVRG using $\mathcal{L}(w)$. In this case, we used a slower decay schedule $\eta_t = 1/\sqrt{t}$ instead of the standard decay schedule $\eta_t = 1/t$ used prior. We note similar to Gradient Descent, SVRG converges only for larger η_t, with the best results at $\eta_t = .1$ (6 epochs). Furthermore, we note performance worsens as fixed η_t decreases regardless of when we start the decay schedule (see all columns). Similar to other methods, $\mathcal{L}(w)$ is minimized faster when we do not decay η_t at all (seen in the final column). However, in the case of $\eta_t = 1$, $\mathcal{L}(w)$ is minimized faster when we decay η_t after the first epoch (seen in the first row and first column). This result shows that decaying for particular fixed η_t values improves performance for SVRG, but a slower decay rate may provide a slight improvement for smaller η_t (seen when comparing third rows to Figure 20)

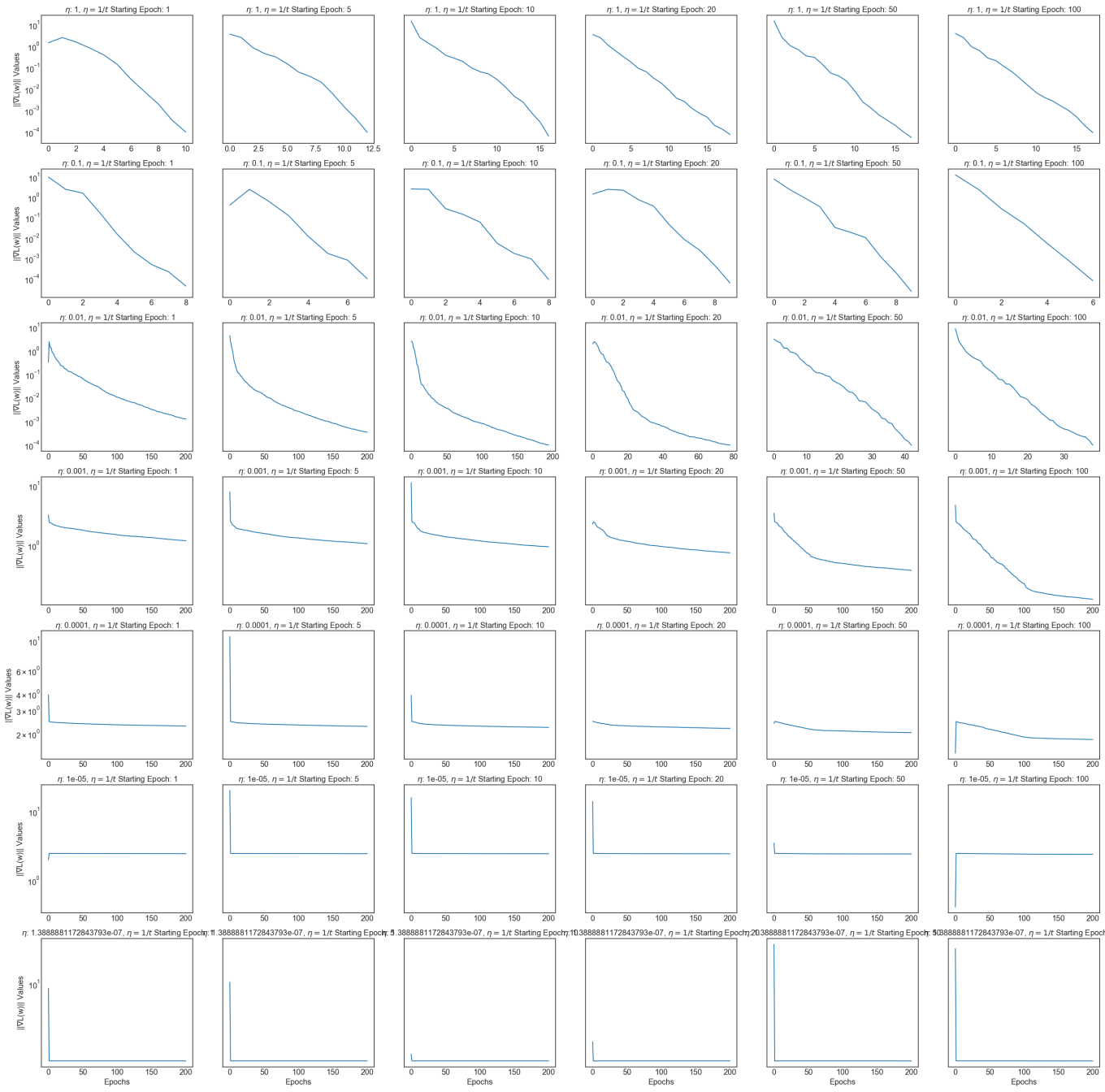

Figure 23: η_t Tradeoff analysis for SVRG using $||\nabla\mathcal{L}(w)||$. In this case, we used a slower decay schedule $\eta_t = 1/\sqrt{t}$ instead of the standard decay schedule $\eta_t = 1/t$ used prior. We note results consistent with Figure 22. We note the longer we keep η_t fixed, $||\nabla\mathcal{L}(w)||$ decreases longer at a linear rate (see first and last columns).

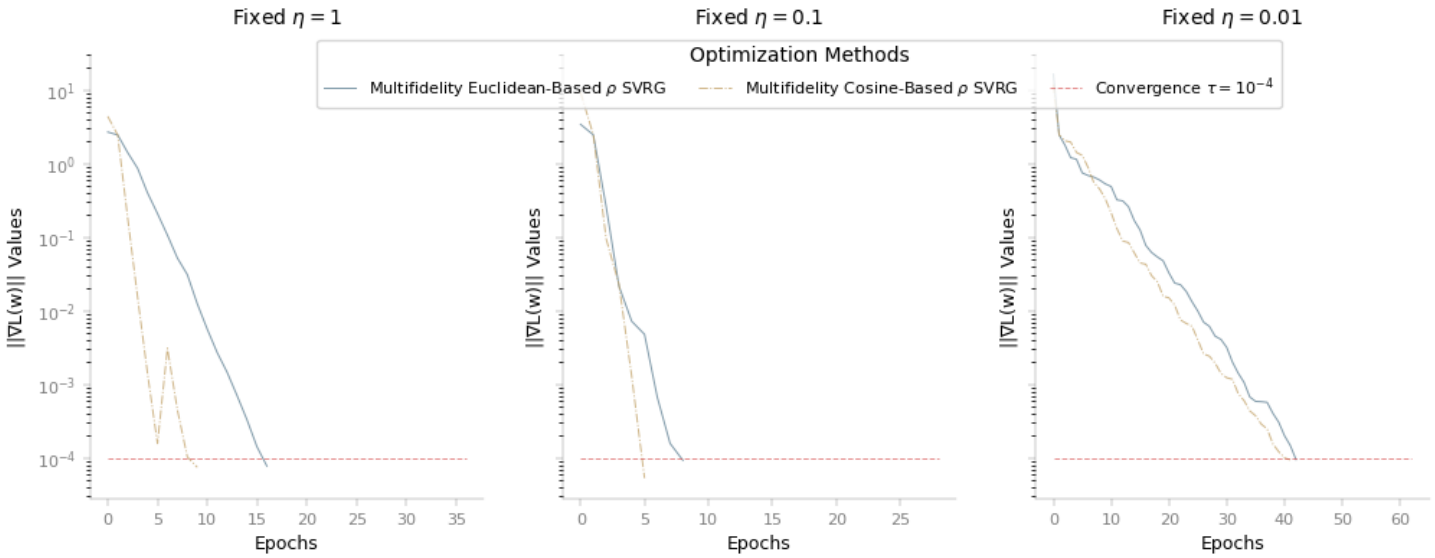

Figure 24: η_t Tradeoff analysis for different Pearson correlation substitutes for multifidelity SVRG using $||\nabla\mathcal{L}(w)||$. For vector spaces, there exists different substitutes for Pearson's correlation coefficient based on the inner product [7] and vector norms [8]. As a result, the true gradient vector $\nabla\mu$ and sampled gradient vector$\nabla\tilde{\mu}$ were compared using the Euclidean and Cosine Similarity distance metrics. Based on these empirical results, we note the Cosine Similarity metric is likely better to use than the Euclidean one. Since the Euclidean metric, $\rho = \frac{||\nabla\mu - \nabla\tilde{\mu}||}{||\nabla\mu|| \cdot ||\nabla\tilde{\mu}||}$ only measured on relative magnitude, the Cosine Similarity metric, $\rho = \frac{\langle \nabla\mu\, , \nabla\tilde{\mu}\, \rangle}{||\nabla\mu|| \cdot ||\nabla\tilde{\mu}||}$ was preferred for SVRG as it measured direction.