Mini-project Resubmission

Group Partner: Meng Lai

1 Introduction:

Stochastic gradient descent (SGD) is an important concept in computational optimization and machine learning. The properties and characteristics of SGD have been of great interest and investigated in many literature. In this paper, we consider using SGD for the optimization of variables in a high-dimensional linear regression problem with the error terms t-distributed.

For SGD optimization, we consider two types of commonly used algorithms, one with constant step size and one with decaying step size, and a modification of the constant step size based on the average of the iterates. Our goal is to: (1) explore different configurations (based on modifying variables) of SGD to fit a regression model, (2) examine how SGD performance depends on hyperparameter choices, and (3) examine the effects of violations of SGD assumptions in theory on model performance, specifically by looking at the mean squared errors between the fitted values and the true parameters.

There are several assumptions we need to make to define the error bounds on the expected values of iterates in SGD including smoothness and convexity of the loss function. However, in practice, it is difficult to guarantee to meet those assumptions. In this paper, we conduct empirical experiments to examine whether Chapter 4 error analyses are applicable when the smoothness and convexity assuptions are violated. Different scenarios of changing configurations of several parameters such as batch sizes, epochs, initializations, etc. are considered to investigate SGD properties such as the convergence rate and noise, which allow us to demonstrate that SGD algorithms converge for some "non-convex" and "not smooth" cases of the loss function with appropriate adjustments to the parameters in the distribution. The experiments prove that the error analyses and bounds in Chapter 4 can still be used as an assessment of the SGD algorithms' performances with varying configurations and they also show us the relationships among the parameters that are insightful for our analysis of their accuracy and efficiency.

2 Background:

Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function that starts from a chosen initialization and progresses in steps to the optimal fitted parameters. SGD estimates the loss and its gradient at each iteration by using a small batch of data selected uniformly chosen randomly. Consider each batch has B observations and at each iteration k, we call each data point is $n(k,1),...,n(k,B)$ and hence, the batch loss is defined as

$$\mathcal{L}_k(\boldsymbol{x}) := \frac{1}{B}\sum_{b=1}^{B} \ell_{(n(k,b))}(\boldsymbol{x}) + r(\boldsymbol{x})$$

and so $\mathbb{E}\{\mathcal{L}_k(\boldsymbol{x})\} = \mathcal{L}(\boldsymbol{x})$. Then, the SGD update algorithm is

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \eta_k \nabla \mathcal{L}_k(\boldsymbol{x}) \tag{1}$$

The next natural question is how to quantify the error that the SGD algorithm produces to ensure our inferences. Error analysis of SGD is an essential tool that helps us understand the limitations of the results we obtain from using SGD and identify the appropriate uses for the results to achieve our goals of the analysis. Define a minimizer, $\boldsymbol{x}_*$, of our function of interest, f to be

$$\boldsymbol{x}_* = \arg\min_{x} f(\boldsymbol{x})$$

Assume that we have a sequence of independent, unbiased stochastic gradient estimates $\{\hat{f'}_k\}$ such that for any $\boldsymbol{x} \in \mathcal{A}$, $f'(\boldsymbol{x}) = \mathbb{E}\{\widehat{f}_k(\boldsymbol{x})\}$. Thus, the SGD update algorithm becomes

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \eta_{k+1}\hat{f}_{k+1}(\boldsymbol{x}_k).$$

A shorthand notation we use is $\mathbb{E}_k$, which is the conditional expectation $\mathbb{E}(.|x_0,...,x_k,\hat{f}'_1,...,\hat{f}'_k)$. The first assumption is that suppose the sequence $(\hat{f}'_k)_{k\in\mathbb{N}}$ is independent and identically distributed. For all $k \in \mathbb{N}$ and $x \in \mathcal{A}$, $f'(x) = \mathbb{E}_{k-1}\{\hat{f}'_k(x)\}$. Moreover, there are assumptions on convexity and strong smoothness to guarantee the convergence and robustness of the stochastic gradient estimates. Specifically, the strong smoothness assumption guarantees the gradient provides valuable information in the region we constrain it to. This is important because essentially, the gradient of a function measures how the function changes when the steps the algorithm takes move in one direction from the point. When the gradient moves arbitrarily, the algorithm does not guarantee convergence and potentially requires more computational power to reach the region where it can provide useful information. In other words, smoothness ensures that when we move in the direction opposite of the gradient, we can still minimize the value of the function to potentially reach the minimizer, $\boldsymbol{x}_0$. Even though we have strong smoothness, it is not enough to guarantee us the optimal $\boldsymbol{x}_0$ because the gradient can be 0, meaning the algorithm gets stuck at local minima. Hence, in theory, convexity is provided to ensure the existence of a global optimum: There exists $\mu > 0$ such that the function f is μ-strongly convex. Having established strong smoothness and convexity conditions, a function f is both L-strongly smooth and convex, there exists $L > 0$ such that for all $k \in \mathbb{N}$, $\hat{f}_k^{'}$ is L-co-coercive. Besides the requirements of co-coerciveness, we need to establish the bounds on the noise of the algorithm around the optimum. In theory, it is required to be uniformly bounded while in practice, this requirement can be hard to meet. With that, we assume a weaker version of boundedness on the optimization error such that there exists $\sigma > 0$ such that for all $k \in \mathbb{N}$,

$$\mathbb{E}_{k-1}||\hat{f}_k^{'}(x_*)||_2^2 \leq \sigma^2.$$

With all of the assumptions we have, we can analyze the behavior of SGD with constant step size $\eta_{k+1} = \eta_k$. We can use induction to derive the error bounds on each individual iterates such that if all the aforementioned assumptions hold and $\eta_k = \eta \in (0, \frac{1}{2L})$, then for $\beta := 1 - 2\eta\mu(1 - \eta L)$,

$$\mathbb{E}(E_k) \leq \beta^k ||x_0 - x_*||_2^2 + \frac{2\eta}{\mu}\sigma^2, \tag{2}$$

and therefore,

$$\mathbb{E}(||x_k - x_0||_2^2) \leq \beta^{k/2}||x_0 - x_*||_2 + \frac{2^{\frac{1}{2}}\eta^{\frac{1}{2}}\sigma}{\mu^{\frac{1}{2}}} \tag{3}$$

This theorem implies that the dependence of the expected errors on the initialization is weak due to the exponential decay of the first term. Moreover, the second term indicates an irreducible expected squared error of order η, scaled by the term σ^2 and the inverse of μ, which leads to less accurate fitted values. Note that the smaller μ is, the bigger the expected squared error becomes. Thus, it is natural for us to want to improve the dependence on the initialization term x_0 while the error term is deterministic, we can average after the $k/2$ iterations, denoted as $\bar{x}_{k/2:k}$. If all the assumptions hold, the error bound on this iterate-average is

$$\begin{aligned}
&\frac{1}{||f''(x_*)^{-1}||_2}\mathbb{E}\{||\bar{x}_{k/2:k} - x_*||_2\} \\
&\leq \varrho\eta\sigma^2 + \frac{\sigma}{k^{1/2}}(1 + 2^{3/2}\rho^{1/2}) + \frac{2\varrho\sigma^2}{\mu k} + \frac{3\rho^{1/2}\sigma}{\eta k}(2 + \rho^{1/2}) \\
&+ \beta^{k/2}\frac{||x_0 - x_*||_2}{\eta k}\{2(1 + \rho^{1/2}) + \frac{\varrho}{2}\beta^{k/2}||x_0 - x_*||_2
\end{aligned} \tag{4}$$

with $\eta_k = \eta \in (0, 1/L)$, then for k even, $\varrho := M/\mu$ and $\beta := 1 - 2\eta\mu(1 - \eta L)$.

Now, consider error analysis for SGD with decreasing step size as $\eta_k \to 0$ for $k \to \infty$. For consistency, the algorithm uses a commonly-used step size schedule $\eta_k = \eta/k^\alpha$ for $\alpha \in (1/2, 1)$ and $\eta \in (0, 1/2L)$, then the error bound of the iterates is:

$$\mathbb{E}(E_k) \leq \exp\left\{-\frac{\mu\eta}{2}(k^{1-\alpha} - 1)\right\}\left(||x_0 - x||_2^2 + \frac{4\alpha\sigma^2\eta^2}{2\alpha - 1}\right) + \frac{2\sigma^2\eta}{\mu k^\alpha} \tag{5}$$

The term $\exp(-\frac{\mu\eta}{2}k^{1-\alpha} - 1)||x_0 - x||_2^2$ decays sub-exponentially but still depends on the initialization while the second term does not depend on it. For the second term, since α is in the denominator, if we increase $\alpha \to 1$, we obtain a faster convergence rate since it minimizes the value of the second term.

The limitations of the mentioned theorems and corollary are that 1) they assume convexity everywhere, 2) strong smoothness is assumed, and 3) the theorems are about the expectations (averages) of individual paths so the error bounds are not guaranteed for the entire SGD paths in both cases, constant and decreasing step sizes. To address the limitations of the assumptions made in these theorems and corollary, in section 2, we consider cases where convexity is not assumed everywhere and is dependent on variables used in the distribution. Then, in section 3, we explore the behaviors of all three SGD algorithms, constant and decreasing step sizes with iterated averages when changing the configurations of different parameters in the distribution.

3 Methods:

The observed data is $\mathcal{D} = \{(y_n, \boldsymbol{\beta}_n)\}_{n=1}^N$ with $y_n \in \mathbb{R}$ is the outcome and $\boldsymbol{\beta}_n \in \mathbb{R}^D$ are the covariates (features). For $\boldsymbol{\beta} \in \mathbb{R}^D$, the linear regression formula becomes $y_n = \boldsymbol{z}_n^T\boldsymbol{\beta} + \varepsilon_n$ where $\varepsilon_1, ..., \varepsilon_N$ are i.i.d. mean-zero random variables and are t-distributed with scale e^{ψ} and ν degrees of freedom. Thus, the conditional distribution of y_n is

$$y_n|\boldsymbol{z}_n, \boldsymbol{x} \sim \mathcal{T}(\boldsymbol{z}_n^T\boldsymbol{\beta}, e^{\psi}, \nu) \tag{6}$$

where $\mathcal{T}(\hat{y}, s, \nu)$ denotes the t distribution with the location parameter $\hat{y}$, scale parameter s, and ν is the degrees of freedom. The density of this t distribution is

$$p(y|\hat{y}, s, \nu) = \frac{c(\nu)}{s}\left\{1 + \frac{1}{\nu}\left(\frac{y-\hat{y}}{s}\right)^2\right\}^{-(\nu+1)/2} \tag{7}$$

with $c(\nu)$ is a normalization term that depends on ν. Consider the log loss equation of the nth observation, we have

$$\ell_n(\boldsymbol{x}) = -\log p(y_n|\boldsymbol{z}_n^T\boldsymbol{\beta}, e^{\psi}, \nu) \tag{8}$$

$$= \begin{cases} \frac{\nu+1}{2}\log\left\{1 + \frac{e^{-2\psi}}{\nu}(y - \boldsymbol{z}_i^T\boldsymbol{\beta})^2\right\} + \psi & \text{if} \quad \nu < \infty \\ \frac{e^{-2\nu}}{2}(y - \boldsymbol{z}_i^T\boldsymbol{\beta})^2 + \psi & \text{if} \quad \nu < \infty \end{cases} \tag{9}$$

In this experiment, we simulated a 10-dimensional dataset of $N = 10000$ observations. Assume a multivariate t-distributed noise, which is the error term in the regression formula, with 10 degrees of freedom and non-diagonal variance. Using the noise, we generated the responses using regression coefficients $\beta = (1, 2, ..., 10)$. Since t-distribution is a heavy-tailed distribution, it can generate many outliers considering that $N = 10000$ observations. Hence, for simplicity, we center our responses around the mean of the distribution, which is computed by averaging the responses generated.

As the SGD update algorithm depends on the loss function, the convexity and smoothness of the loss function determine the behaviors of the algorithm. Thus, it is of interest to investigate the convexity and smoothness of the loss function in Eq.8. Specifically, we are interested in the term $p(y_n|\boldsymbol{z}_n^T\boldsymbol{\beta}, e^{\psi}, \nu)$. Define

$$g(\hat{y}, \psi, y) := \frac{\nu+1}{2}\log\left\{1 + \frac{e^{-2\psi}}{\nu}(y-\hat{y})^2\right\} + \psi$$

and the loss function then becomes $\ell_{(n)}(\boldsymbol{x}) = g(\boldsymbol{z}_n^T\boldsymbol{\beta}, \psi, \nu)$.

Proposition 1 *Treating ψ and y as fixed constants, the Hessian of $g(\hat{y}, \psi, \nu)$ is*

$$\frac{\partial^2 g}{\partial \hat{y}^2} = \frac{e^{-2\psi}\nu(\nu+1) - e^{-4\psi}(\nu+1)(y-\hat{y})^2}{(\nu + e^{-2\psi}(y-\hat{y})^2)^2}.$$

Then, $g(\hat{y}, \psi, \nu)$ is convex on $|y - \hat{y}| < \nu^{1/2}e^{\psi}$, which is equivalent to $\ell_{(n)}(\boldsymbol{x}) = g(\boldsymbol{z}_n^T\boldsymbol{\beta}, \psi, \nu)$ is convex on $|y - \hat{y}| < \nu^{1/2}e^{\psi}$.

Proposition 2 *Treating y and $\hat{y}$ as fixed constants, the Hessian of $g(\hat{y}, \psi, \nu)$ is*

$$\frac{\partial^2 g}{\partial \psi^2} = \frac{2\nu e^{-2\psi}(\nu+1)(y-\hat{y})^2}{(\nu + e^{-2\psi(y-\hat{y})^2})^2}$$

which is positive for all $\psi > 0$, leading to convexity of $\ell_{(n)}(\boldsymbol{x}) = g(\boldsymbol{z}_n^T\boldsymbol{\beta}, \psi, \nu)$ everywhere on ψ.

The function g is convex in a certain region of $\hat{y}$ and non-convex outside of that region. Since the loss function is a negative log function of g, the convexity of $\ell_{(n)}(\boldsymbol{x})$ follows the same convexity rule, specifically only convex in the region where $|y - \hat{y}| < \nu^{1/2}e^{\psi}$. Meanwhile, following the same logic, we see that $\ell_{(n)}(\boldsymbol{x})$ is convex everywhere on ψ. Note that if a function is non-convex in one variable, it is non-convex. This means that if $\hat{y}$ is in the region of $|y - \hat{y}| < \nu^{1/2}e^{\psi}$, then the loss function is convex. If $\hat{y}$ lies outside of that region, the loss function is non-convex. The proof of convexity can be found in Appendix A and B.

Another important property of SGD to consider is the strong smoothness assumption: When the Hessian result is unbounded, the function is not strongly smooth. From Prop.(1), when $|y - \hat{y}| \to \infty$, then the second term in the numerator of the Hessian is $e^{-4\psi}(\nu+1)(y-\hat{y})^2 \to \infty$, meaning it is $\nabla^2 g(\hat{y}) < c$ where c is a constant and $c < \infty$. On the other hand, when we consider $|y - \hat{y}| \to 0$, we have the second term of the numerator goes to 0, meaning the $\nabla^2 g(\hat{y}) < d$ with d a random constant. In this case, the Hessian of the function g over $\hat{y}$ is bounded so the function g is strongly smooth in $\hat{y}$.

On the other hand, consider the Hessian of the function g over ψ. When $\psi \to \infty$, the term $e^{-2\psi} \to 0$, so the Hessian also goes to 0. On the other hand, when $\psi \to -\infty$, the term $e^{-2\psi} \to \infty$, so the Hessian also goes to ∞. Hence, the Hessian result is unbounded, so no strong smoothness is concluded here. The proof of strong smoothness can be found in Appendix C.

In summary, in the case of linear regression in which we assume the t-distribution for the error terms, the loss function has been proven to be convex over a certain region of $\hat{y}$ and convex while being neither convex nor strongly smooth anywhere over ψ. These conclusions have violated the assumptions the theory of SGD required convexity and strong smoothness everywhere. In theory, it is important to know that the convexity property of the loss function influences the convergence properties of SGD which ensures a single global minimum of the investigated function f. Meanwhile, the smoothness of the loss function relates to the behavior of its gradient and enables a more efficient optimization process toward to true unknown parameter without too much noise and oscillations. However, in practice, it is hard to guarantee the existence of both properties for the function of interest, and the function we demonstrated above is an example of performing SGD when the loss function is not convex and strongly smooth. In the next section, different configurations of parameters were considered to investigate the behavior of this scenario.

4 Experiments

The accuracy of the algorithm is demonstrated as the norm of the differences between the true optimum and the fitted values. SGD algorithm with constant step size η and decreasing step size $\frac{\eta_0}{k^\alpha}$, along with the iterate-average SGD with a constant step size that uses the most recent half of the iterates. In the experiment, we set $\nu = 5, \eta = 0.2, \eta_0 = 5, \alpha = 0.51$. The batch size is $B = 10$ and the initialization of $\boldsymbol{x}_0 = \mathbf{0}$. The number of iterations, denoted as K, per epoch, is calculated as $\left\lfloor \frac{N}{B} \right\rfloor$. The following subsections correspond to changing configurations of the parameters in the t-distribution of the error terms in the regression model.

4.1 The effect of iterations:

The number of iterations is essential because it affects the convergence rate and computational power requirements, among many other factors such as performance accuracy and generalization. Thus, when the theoretical assumptions do not hold, it is natural to ask a question on the effect of iterations and if changing the iterations can offset the difficulties of model assessment. Hence, we experiment with two different numbers of epochs, 10 and 100, which are directly related to the number of iterations as explained above.

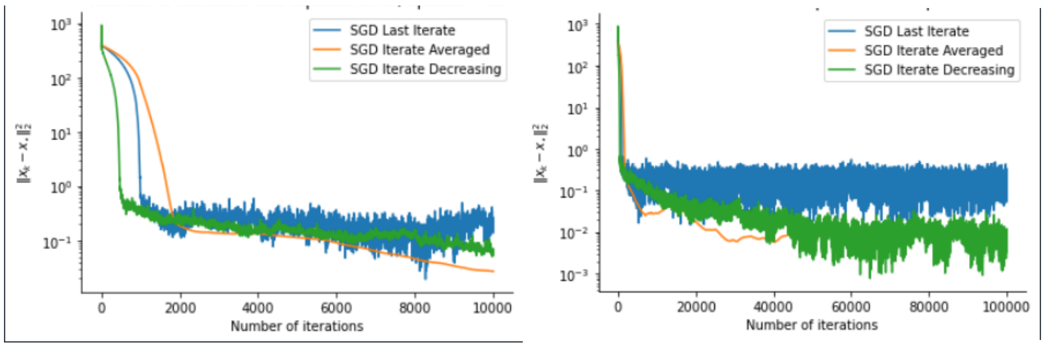

Figure 1: Accuracy of each algorithm for when the number of epochs is 10 (left) and is 100 (right)

In both cases, all three algorithms converge fast at the number of iterations less than 1000. However, the bigger epochs cause less noise in all of the algorithms. The reason for this behavior comes from the relationship among epochs, the number of iterations K, and batch size B. In this setup, we choose the number of epochs to be 10 and $B = 10$, while $N = 10000$. The higher number of iterations is equivalent to the number of times the algorithms update the parameters, and with the number of epochs and batch size are inversely proportional. Hence, to reduce the noise level in the optimization process, the tradeoff between those two quantities should be considered. In this case, if the batch size was increased to 100, then the number of iterations required to achieve the same convergence rate remains the same, potentially reducing the noise level in the algorithms.

4.2 The effect of initializations:

As explained in section 3, error analysis on the performance of SGD algorithms is dependent on the initializations, or in other words, the starting point of the algorithm. In this experiment, we initialize the starting point increasingly further away from the optimal point in the positive direction and reset other parameters to the default.

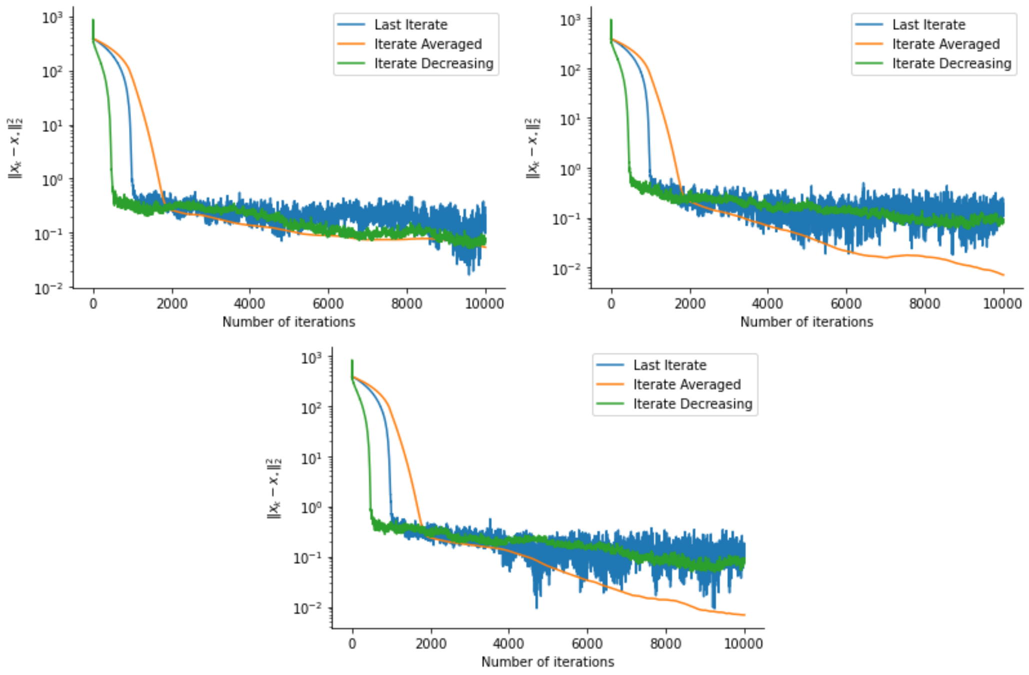

Figure 2: Accuracy of each algorithm with initializations increasingly further from the optimal point by 100 (upper left), 500 (upper right), and 1000 (bottom) units.

We see that in all three scenarios, the SGD algorithm with decreasing step size outperforms the SGd algorithm with constant step size with less noise and has the fastest convergence rate among all three. Under the first initialization experiment in which the distance is 100 units further from the optimal parameters, the performance of the decreasing step size algorithm and iterated averaged performance is almost the same as the lines are overlapping with each other, while the constant step size algorithm has more noise and oscillates more as the number of iterations increases. However, for the middle and last plots with initializations 500 and 1000 units further from the optimal points, the iterated averages have the lowest error, meaning higher accuracy.

From the observations, with the same logic as in the previous part, we can explain why the constant size algorithm has more noise than the decreasing step size SGD algorithm as the number of iterations increases, but the effect of initializations on the algorithms can be seen clearly, specifically for the iterate-averaged SGD algorithms. The reason for its better performance is that we consider the averages on the most recent 50% of the iterations with constant step size, leading to better predictions of the fitted values. Refer to Eq.(4), the dependence on the initializations is more profound than that of Eq.(3) and Eq.(5) and this explains how the error rate reduces in iterate-averaging SGD algorithm with our choice of initializations in one direction. In other words, increasing number of iterations reduces the error rate given that $||\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_*|| < \varepsilon$, then ε signifies the irreducible noise in the algorithms. Specifically, ε stands for the noise terms in Eq.(4), Eq.(3), and Eq.(5) that do not depend on the initialization. The noise calculation is computed empirically by taking the averages of the error rate over a subset of iterations of the constant step size algorithm, which results in $\varepsilon \sim 0.07$.

4.3 The effect of step size:

As for error analysis bounds, the effect of step size on the irreducible noise influences the performance and convergence of an SGD algorithm, both with constant and decreasing step size. For constant step sizes, we are interested in the behavior of the constant step size SGD and iterate-averaged SGD algorithms.

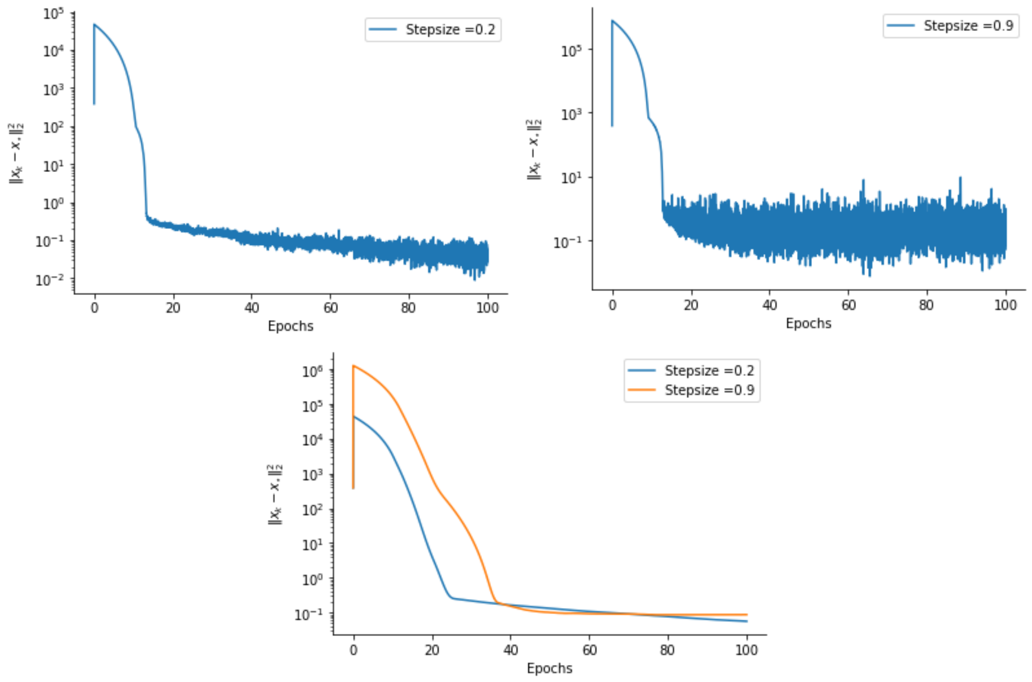

Figure 3: Accuracy of constant step size of 0.2 (top left) and 0.9 (top right) and iterate averaged of the iterations (bottom).

From the plots, it is clear to see the effect of different step sizes on the noise level of the algorithm. For the SGD algorithm with constant step sizes, when $\eta = 0.2$, the noise in the algorithm is smaller, and less oscillations around the optimal parameter compared to that when $\eta = 0.9$. From Eq.(2), the noise level can be approximated by the step size η, scaled by σ^2 and flattened by μ. With the direct proportional relation with η, the bigger the step size is, the more noise the algorithm produces.

For decreasing step size, we are interested in changing the η_0 and α to see the behavior of the algorithm.

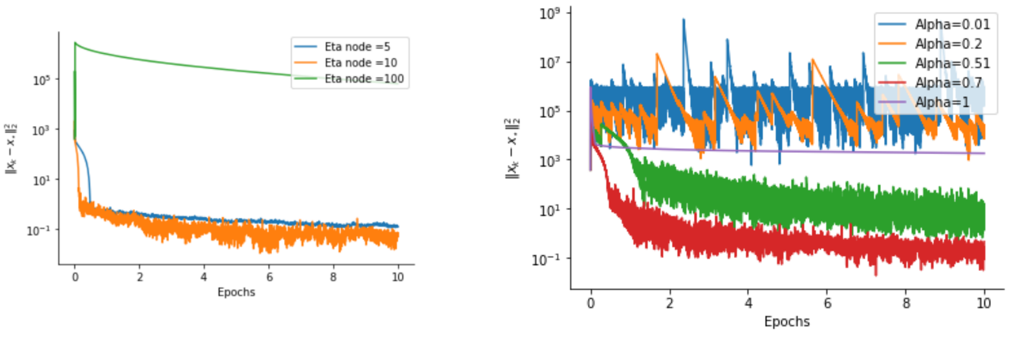

Figure 4: Accuracy of decreasing step size algorithm with changing η_0 (left) and α (right).

Using the first plot in Figure 4, we see that the SGD algorithm with decreasing step sizes shows vastly different performances with changing $\eta_0 = 5, 10, 100$ and observe that $\eta = 10$ performs the best. The intuition for this observation is that from Eq.(5), notice that the second term denotes the irreducible noise and η_0 has a direct proportional relation with the expected error rate, hence the worse performance following when η_0 increases drastically. On the other hand, using the second term of the bound, as α increases, we observe that $\alpha = 0.7$ performs well. As α increases, the second term of the bound minimizes, leading to a faster convergence rate but it requires a tradeoff as $k \to \infty$. The first term of the bound still depends on α, so finding the right α on a case-by-case basis is critical to achieving a good convergence rate with efficiency and accuracy.

4.4 The effect of batch size:

Batch size, along with our choice of the number of epochs, determines the number of iterations. As mentioned in the experiment with different number of epochs, the choice of batch size can give us information on the accuracy, convergence rate, and noise level in the optimization process. Next, we assess the differences in the performance of all three algorithms with different batch sizes, 5, 10, and 100.

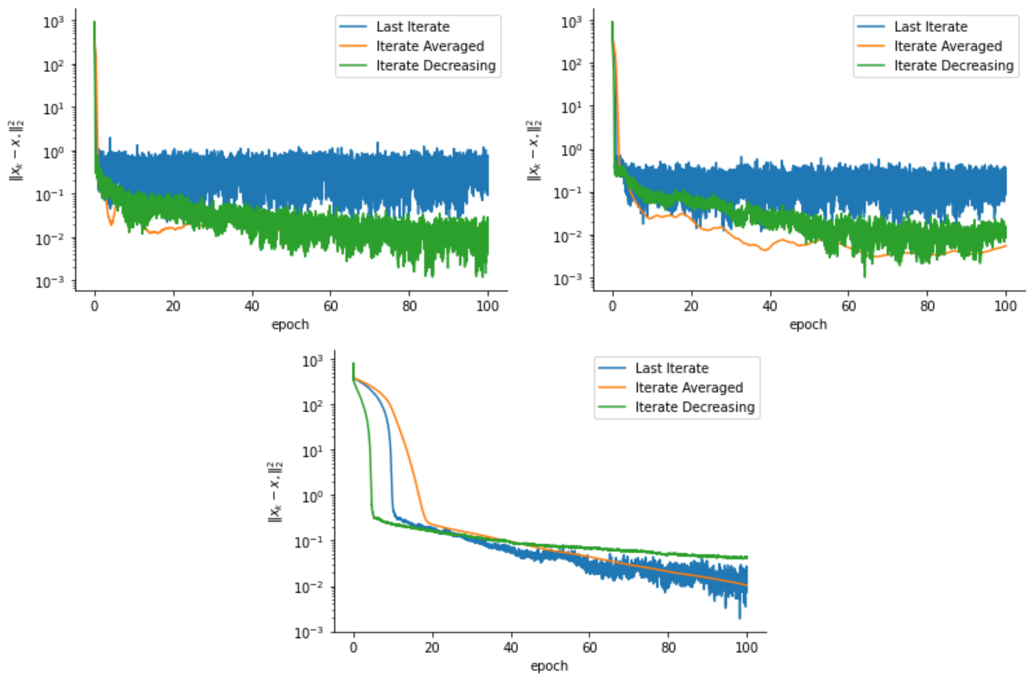

Figure 5: Accuracy of each algorithm with $B = 5$ (upper left), $B = 10$ (upper right), and $B = 100$ (bottom).

It is clear from what we observe is that as the number of batch sizes increases, the level of noise in each algorithm reduces. The convergence rate of the algorithms seems to remain the same, while their accuracy improves as the batch size increases. One possible reason to explain these phenomena is the tradeoff between epochs and batch sizes we used in the previous sections. We set the number of epochs to be 10 while increase the number of batch sizes, making the number of iterations lower than section 4.1 so the algorithms will not either overfit or pick up too much noise that can lead to slow convergence or divergence.

4.5 The effect of ν:

The calculation of the loss function is determined by ν, the degree of freedom, chosen for the t-distribution. When $\nu \to \infty$, the t-distributed error terms become standard normal distribution so we are interested in understanding the performances of all three algorithms with different distributions of the error term in the regression problem.

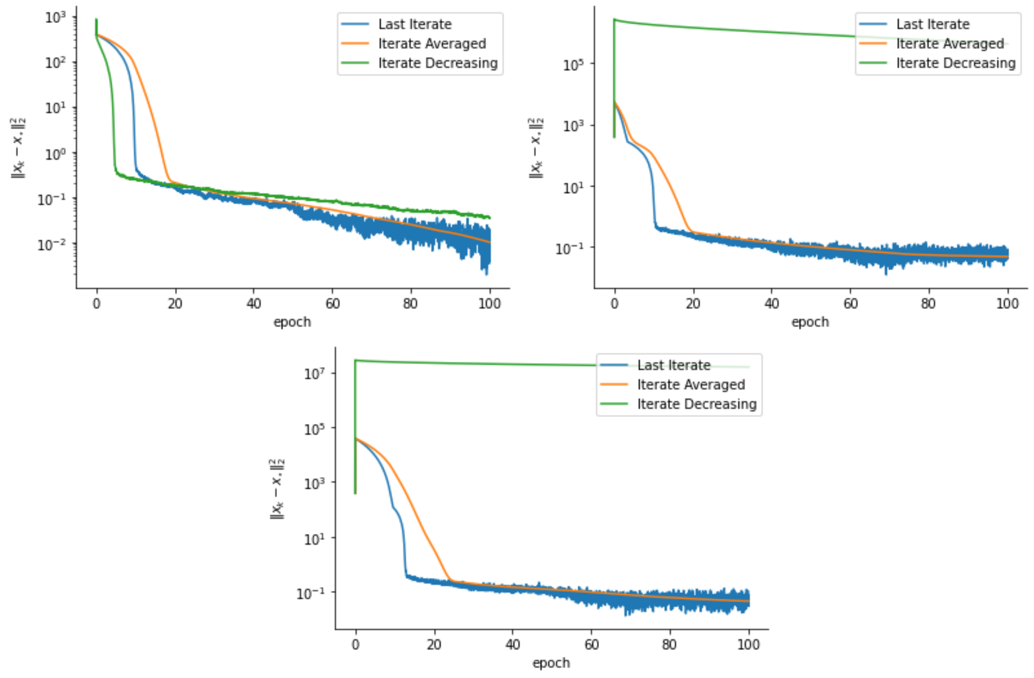

Figure 6: Accuracy of each algorithm with $\nu = 5$ (upper left), $\nu = 1000$ (upper right), and $\nu = \infty$ (bottom).

The performances of the SGD algorithms with constant step size are consistent and remain approximately the same with changing degrees of freedom. However, the error norm of the SGD algorithm with decreasing step size shows us the algorithm is not able to predict well with no sign of convergence within 10 epochs when $\nu = 1000$ and $\nu = \infty$. One potential cause of this bad performance is that the data is generated with outliers due to the error term being t-distributed, but with $\nu \to \infty$, the SGD algorithm is optimizing a regression with standard normal errors that could not capture the outliers well.

5 Conclusion:

The first major takeaway from these experiments is that there is a tradeoff between epochs and batch sizes to achieve the optimal convergence rate and reduce noise in the optimization process. In practice, it is not uncommon to have a large dataset of thousands or even millions of observations, so if the number of epochs is too big while the batch size is too small, then the number of iterations becomes significantly large. If the algorithm iterates and updates its parameters too many times, it can lead to high accuracy on the training set but low accuracy on the test set, which essentially means the inability to generalize the learned results that can lead to inaccurate predictions. Thus, depending on the number of observations, we can set an appropriate number of epochs and batch size to achieve optimality.

A second major takeaway is that even though the convexity and strong smoothness assumptions don't hold, the behaviors of SGD algorithms, with constant step sizes and decreasing step sizes, can still be justified with the error analysis bounds we obtain theoretically when all the assumptions hold. When those assumptions are not met, there are apparent difficulties in obtaining a fast convergence rate and accuracy, but those can be offset with appropriate configurations of parameters of the function of interest.

SGD has been used in the statistics and machine learning community for a long time, yet there are research questions that need to be explored further. One of those is the theoretical guarantee of convergence of momentum-based SGD algorithms on non-convex functions has not been vastly explored. One research topic that could be of interest is how to establish conditions on momentum-based SGD algorithms to ensure that the algorithm will eventually converge to the global optimal point rather than the local optimum, given a scenario in which the local and global optimal points have almost identical domains of attraction and the non-convexity property of the function of interest. Recent studies on momentum-based SGD are still using convexity property. Even though it is arguable that the theoretical grounds for non-convex functions with momentum SGD are specific to each distribution being used, an extensive theoretical study on error analysis will be beneficial to understand the accuracy of the algorithm in this context and this can be potentially extended into the studies on momentum SGD algorithms with constant step sizes and decreasing step sizes. Another future research question is how to improve the SGD algorithms in a way that we can quantify the error bounds on the performance when the data is not independent and identically distributed (i.i.d). Non-i.i.d data can give arbitrary optimization errors, especially when the data gets infinitely large or when other assumptions do not hold. A research question in this scenario is to improve the SGD algorithm to increase the predictability of the parameters or functions of interest and extend it further with theoretical grounds such as error analysis, convergence rates, robustness to misspecification of data, or outliers detection.

Appendix A

Treating ψ and y as fixed constants, we can determine whether $g(\hat{y}, \psi, y)$ is convex by taking the second derivative and see what range of $\hat{y}$ is positive.

$$\begin{aligned}\frac{\partial g}{\partial \hat{y}} &= \left(\frac{\nu+1}{2}\right)\frac{\frac{e^{-2\psi}}{\nu}(-2(y-\hat{y}))}{1+\frac{e^{-2\psi}}{\nu}(y-\hat{y})^2}\\ &= \left(\frac{\nu+1}{2}\right)\left(\frac{-2e^{-2\psi}}{\nu+e^{-2\psi}(y-\hat{y})^2}\right)\\ &= \frac{-e^{-2\psi}(\nu+1)}{\nu+e^{-2\psi}(y-\hat{y})^2}\end{aligned}$$

From there, take the second derivative to obtain the interval of $|y-\hat{y}|$ by applying the quotient rule on the first derivative, we then obtain

$$\frac{\partial^2 g}{\partial \hat{y}^2} = \frac{e^{-2\psi}\nu(\nu+1) - e^{-4\psi}(\nu+1)(y-\hat{y})^2}{(\nu+e^{-2\psi}(y-\hat{y})^2)^2}$$

We consider the positive region only, meaning

$$\begin{aligned}&e^{-2\psi}\nu(\nu+1) - e^{-4\psi}(\nu+1)(y-\hat{y})^2 > 0\\ &\Leftrightarrow e^{-2\psi}\nu(\nu+1) > e^{-4\psi}(\nu+1)(y-\hat{y})^2\\ &\Leftrightarrow (y-\hat{y})^2 < \nu e^{2\psi}\\ &\Leftrightarrow |y-\hat{y}| < \nu^{1/2}e^{\psi}\end{aligned}$$

Appendix B

Suppose $h(\boldsymbol{\beta}) = \boldsymbol{z}_n^T\boldsymbol{\beta}$ and it is clear that $h(\boldsymbol{\beta})$ is a linear function. Then, $g(\boldsymbol{z}_n^T\boldsymbol{\beta}) = g(h(\boldsymbol{\beta}))$ a convex function when $|y-\hat{y}| < \nu^{1/2}e^{\psi}$, so this means $g(\boldsymbol{z}_n^T\boldsymbol{\beta})$ is convex when $|y-\hat{y}| < \nu^{1/2}e^{\psi}$.

Appendix C

Treating $\hat{y}$ and y as constants, we take the first derivative of g and move on to find the second derivative.

$$\begin{aligned}\frac{\partial g}{\partial \hat{\psi}} &= \left(\frac{\nu+1}{2}\right)\frac{e^{-2\psi}(-2)\frac{(y-\hat{y})^2}{\nu}}{1+\frac{e^{-2\psi}}{\nu}(y-\hat{y})^2}+1\\ &= \frac{-(\nu+1)(y-\hat{y})^2e^{-2\psi}}{\nu+e^{-2\psi(y-\hat{y})^2}}+1\end{aligned}$$

Then we can take the second derivative based on the first derivative and obtain

$$\frac{\partial^2 g}{\partial \psi^2} = \frac{2\nu e^{-2\psi}(\nu+1)(y-\hat{y})^2}{(\nu+e^{-2\psi(y-\hat{y})^2})^2}$$

Notice that because the function is convex in the area where the Heissan is positive and the Heissan we obtained is positive on both the numerator and denominator, so the function is convex everywhere in ψ.