Purdue University

OPT4DL

HW 2

September 17, 2024

0033678229 John Yeary

Contents

1 Question 1

1.1 Part a

This paper revolves around the central idea that ”not all samples are created equal.” During training, most samples contribute significantly in the early epochs, but further gains are achieved by focusing on harder samples. These are samples where the model struggles the most. The authors identify these samples as samples with large gradient norms.

The authors propose using importance sampling to prioritize such hard samples. By selecting examples with larger gradient norms, they aim to reduce the variance of the gradient estimator, which leads to more efficient training. The authors establish a connection between the variance of the gradient and the speed of convergence, showing that focusing on more informative samples accelerates training.

The authors go on to derive an upper bound to the gradient norm, which is a function of the smoothness L, and the gradient of the loss function with respect to outputs from the last layer in the neural network. This provides a more efficient calculation than re-calculating the gradient of the loss function per sample for every iteration.

The authors also include an estimate for the variance reduction enabled by their importance sampling scheme, allowing their algorithm to switch on when the opportunity arises.

1.2 Part B

One strength in this paper is that they provided a metric for evaluating when to turn on their algorithm. It could be prohibitively computationally expensive to evaluate the gradient per sample for each epoch, but by gating the algorithm by their metric, there is a guaranteed boost. Further, they provided a weaker, but computationally effective bound for the per sample gradient, which again helps with computation.

Something I’m curious of is how Principal Component Analysis (PCA) might help this algorithm. It seems like the idea is that you capture most of the variance within your dataset with a handful of samples. If you were to perform PCA initially or maybe periodically, attempting to maximize the variance of your gradient and project your dataset onto the principal vectors, I’d imagine you can reduce the number of samples required by a fair amount. I’d assume you would want to re-introduce some of the noise to your dataset in some way to prevent over fitting.

2 Question 2

2.1 Part A

This paper is focused on estimating the gradient with a bias towards samples with the highest current loss. This is another importance sampling scheme, similar to Question 1. Instead of attempting to quantify the gradient of the loss, this paper suggests calculating a gradient estimate using a batch consisting of samples with the highest loss values. This gradient estimate then gets used to update the weights.

Intuitively in a classification setting, this suggests that samples at the boundary between classes will be more influential in the final parameter set than samples far from the boundary. By the author's implementation one can tune the sensitive cluster size by growing or shrinking the batch size. At the extremes, this means that the weights will be tuned by considering all of the points, or only a single point.

The authors guarantee a sublinear convergence rate.

2.2 Part B

One thing that struck me about this paper was that by biasing sampling, you no longer guarantee that you generalize well. This paper does specify that it is only for convex or weakly convex cases, so it may not matter much. It would not be able to generalize to non-convex as well.

If you had sufficiently 'hard' data, it may be difficult to generalize the model to unseen data. Effectively this algorithm ignores 'easy' data, but if 'easy' covers 95% of the test data, it may be fine to train against.

You might be able to fix this potential issue by including some amount of 'easy' data per batch. I'd think you could maintain two pools of sample data, one of 'easy' and one 'hard'. You could then sample varying amounts from each pool, depending on where you are in your training process, if you believe you're stuck at a local extrema, etc.. The further you are in your training, the more 'hard' data you introduce.

3 Question 3

3.1 Part A

This paper proposes an adaptive step size that attempts to minimize the distance between the function at the current iteration and the optimal function value for the current iteration. This relies upon knowledge of the optimal function value, and the stochastic gradient at the current iteration to compute.

The authors argue that this optimal value is available depending upon the type of loss function used. This is built on the idea that:

$$f_i{}^* = inf_x f_i(x)$$

So the function itself is not required.

This is distinct from the deterministic Polyak form, which depends upon using the true minimum of the function.

The authors provide convergence analysis, and support the claim that their algorithm will converge for convex, weakly convex and non convex functions.

The step size will require knowledge of the optimal function value at each step, which may be difficult for many learning instances. They provide an example using a logistic loss with and without regularization, which they claim is 0 or has a closed form expression in the regularized case.

The authors argue that overparameterized models allow each individual loss function to drive to its minimum when $x = x^*$.

3.2 Part B

It seems difficult to apply the step size to over parameterized cases. It seems that the step size could change very rapidly from iteration to iteration. With no form of filtering on the step size, any outlier that may no be near its optimum function value will cause a jump in step size. For a noisy/complex enough function,this could mean you jump over the minima many times. The author did mention an upper bound, so this case may be covered by that bound.

It seems that it would be beneficial to introduce some simple filtering to the step size, either a moving average or something similar. That way, you prevent the 'noisy' jumps I described above.

Further expansion into ranked choice batching or other methods could help highlight the benefits of this step size.

4 Question 4

4.1 Part A

Problem statement: Given an L-Smooth function $\mathcal{L}(\mathbf{w})$ that is not necessarily convex, demonstrate that:

$$||\nabla\mathcal{L}(\mathbf{w})|| <= 2L[\mathcal{L}(\mathbf{w}) - \mathcal{L}^*] \tag{1}$$

By L-Lipschitz:

$$\mathcal{L}(\mathbf{x}) <= \mathcal{L}(\mathbf{y}) + < \nabla\mathcal{L}(\mathbf{y}), x - y > + \frac{L}{2}||x-y||^2 \tag{2}$$

Rearranging:

$$\mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{y}) - < \nabla\mathcal{L}(\mathbf{y}), x - y > <= \frac{L}{2}||x-y||^2 \tag{3}$$

Let $\Phi(\mathbf{x}) = \mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{y}) - < \nabla\mathcal{L}(\mathbf{y}), x - y >$. Now:

$$\nabla\Phi(\mathbf{x}) = \nabla\mathcal{L}(\mathbf{x}) - \nabla\mathcal{L}(\mathbf{y}) \tag{4}$$

$$||\nabla\Phi(\mathbf{x})|| = ||\nabla\mathcal{L}(\mathbf{x}) - \nabla\mathcal{L}(\mathbf{y})|| <= L||x-y|| \tag{5}$$

So Φ is L-Lipschitz.

Let $\mathbf{z} = \mathbf{x} - \frac{||\nabla\Phi(x)||}{L}v$ Where $\mathbf{z}$ is along the curve: $||v|| = 1; < \nabla\Phi(\mathbf{x}), v >= ||\nabla\Phi(x)||$.

$$\Phi(\mathbf{z}) <= \Phi(\mathbf{x}) + < \nabla\Phi(\mathbf{x}), z - x > + \frac{L}{2}||\mathbf{Z} - \mathbf{x}||^2 \tag{6}$$

$$\Phi(\mathbf{z}) <= (\mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{y}) - < \nabla\mathcal{L}(\mathbf{y}), \mathbf{x} - \mathbf{y} >) + < \nabla\Phi(\mathbf{x}), -\frac{||\nabla\Phi(x)||}{L}v > + \frac{L}{2}||\frac{\nabla\Phi(\mathbf{x})}{L}v||^2 \tag{7}$$

$$\Phi(\mathbf{z}) <= (\mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{y}) - < \nabla\mathcal{L}(\mathbf{y}), \mathbf{x} - \mathbf{y} >) - \frac{||\nabla\Phi(\mathbf{x})||^2}{L} + \frac{1}{2L}||\nabla\Phi(\mathbf{x})||^2 \tag{8}$$

Let $\mathbf{y} = w^*$ and let $\mathbf{x} = w \in \mathbb{R}^d$. Since w^* is a minima, $\nabla\mathcal{L}(w^*) = 0$.

$$\Phi(\mathbf{z}) <= \mathcal{L}(w) - \mathcal{L}^* - \frac{1}{2L}||\nabla\mathcal{L}(w)||^2 \tag{9}$$

$$\Phi(w^*) = \mathcal{L}^* - \mathcal{L}^* - < 0, y - y >= 0 \tag{10}$$

Plugging in $\Phi(\mathbf{z})$:

$$\mathcal{L}(\mathbf{z}) - \mathcal{L}^* <= \mathcal{L}(w) - \mathcal{L}^* - \frac{1}{2L}||\nabla\mathcal{L}(w)||^2 \tag{11}$$

$$\mathcal{L}(z) <= \mathcal{L}(w) - \frac{1}{2L}||\nabla\mathcal{L}(w)||^2 \tag{12}$$

By definition of L^*, $L^* <= L(z)$:

$$\mathcal{L}^* <= \mathcal{L}(z) <= \mathcal{L}(w) - \frac{1}{2L}||\nabla\mathcal{L}(w)||^2 \tag{13}$$

$$\mathcal{L}^* - \mathcal{L}(w) <= -\frac{1}{2L}||\nabla\mathcal{L}(w)||^2 \tag{14}$$

$$2L[\mathcal{L}(w) - \mathcal{L}^*] >= ||\nabla\mathcal{L}(w)||^2 \tag{15}$$

4.2 Part B

Problem Statement: Consider a convex and L-smooth function $\mathcal{L}(w), w \in \mathbb{R}^d$. Show that GD, i.e. $w_{t+1} = w_t - \eta \nabla \mathcal{L}(w_t)$ applied to minimizing$\mathcal{L}(w)$ satisfies:

$$\mathcal{L}(\tilde{w}) - \mathcal{L}^* = \mathcal{O}(\frac{L||w_1 - w^*||^2}{T}) \tag{16}$$

With $\tilde{w}$ as an output, and $\eta = \frac{1}{2L}$.

Proof: Using L-Lipschitz:

$$\begin{aligned}
\mathcal{L}(w_{t+1}) &\leq \mathcal{L}(w_t) + \nabla \mathcal{L}(w_t)^\top (w_{t+1} - w_t) + \frac{L}{2}\|w_t - w_{t+1}\|^2 \\
\mathcal{L}(w_{t+1}) &\leq \mathcal{L}(w_t) + \eta||\nabla \mathcal{L}(w_t)||^2 - \frac{L\eta^2}{2}||\nabla \mathcal{L}(w_t)||^2 \leftarrow \text{After substituting update rule for } w_{t+1} \\
\mathcal{L}(w_{t+1}) &\leq \mathcal{L}(w_t) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla \mathcal{L}(w_t)\|^2 \\
\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) &\leq \mathcal{L}(w_t) - \mathcal{L}(w^*) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla \mathcal{L}(w_t)\|^2 \leftarrow \text{subtract } \mathcal{L}(w^*) \text{ from both sides}
\end{aligned}$$

By convexity:

$$\mathcal{L}(w_t) - \mathcal{L}(w^*) \leq \nabla \mathcal{L}(w_t)^\top (w_t - w^*)$$

Which by Cauchy-Schwartz, results in:

$$\mathcal{L}(w_t) - \mathcal{L}(w^*) \leq \|\nabla \mathcal{L}(w_t)\| \cdot \|w_t - w^*\|$$

Solving for $||\nabla \mathcal{L} w_t||$:

$$\|\nabla \mathcal{L}(w_t)\| \geq \frac{\mathcal{L}(w_t) - \mathcal{L}(w^*)}{\|w_t - w^*\|}$$

Substitution back into the result above:

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) \leq \mathcal{L}(w_t) - \mathcal{L}(w^*) - \eta\left(1 - \frac{L\eta}{2}\right)\frac{(\mathcal{L}(w_t) - \mathcal{L}(w^*))^2}{||w_t - w^*||^2}$$

Note that $||w_t - w^*||$ decreases with t; and is bounded by $||w_1 - w^*||$. Finally:

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) \leq \mathcal{L}(w_t) - \mathcal{L}(w^*) - \eta\left(1 - \frac{L\eta}{2}\right)\frac{(\mathcal{L}(w_t) - \mathcal{L}(w^*))^2}{||w_1 - w^*||^2}$$

This implies that the stated $\mathcal{O}$ bound holds. To demonstrate, apply induction.

4.2.1 Recursive Update/ Induction

Assume

$$\mathcal{L}(w_t) – \mathcal{L}(w^*) \text{ is bounded by} \frac{L||w_1 - w^*||^2}{T} \tag{17}$$

Base case: $t = 1$

$$\mathcal{L}(w_t) - \mathcal{L}(w^*) \leq L||w_1 - w^*||^2 \tag{18}$$

Applying the assumption:

$$\begin{aligned}
\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) &\leq \mathcal{L}(w_t) - \mathcal{L}(w^*) - \eta\left(1 - \frac{L\eta}{2}\right)\frac{(\mathcal{L}(w_t) - \mathcal{L}(w^*))^2}{||w_1 - w^*||^2} \\
\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) &\leq \frac{L||w_1 - w^*||^2}{t}(1 - \frac{\eta}{t}\left(1 - \frac{L\eta}{2}\right))
\end{aligned}$$

Substituting: $\eta = \frac{1}{2L}$

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w^*) \leq \frac{L||w_1 - w^*||^2}{t}(1 - \frac{3}{8Lt})$$

As $t \to T$:

$$\mathcal{L}(w_t) - \mathcal{L}(w^*) \leq \frac{L||w_1 - w^*||^2}{T} = \mathcal{O}(\frac{L||w_1 - w^*||^2}{T})$$

Assuming T is sufficiently large to allow the $\frac{3}{8LT}$ term to drop out. This holds for any t, meaning that however $\tilde{w}$ is selected, it will still have this bound.

4.3 Part C

Now consider using SGD for minimizing the convex and L-smooth function $\mathcal{L}(w) = \mathbb{E}_z[\ell(w, z)]$ under the SFO oracle introduced in class. Show that SGD, i.e. $w_{t+1} = w_t - \eta \nabla \ell(w_t, z_t)$, where z_t are i.i.d. samples from P_z, applied to minimizing $\mathcal{L}(w)$ satisfies

$$\mathbb{E}_{z_1,\ldots,z_T}\left[\mathcal{L}(\hat{w}) - \mathcal{L}^*\right] = O\left(\frac{L\|w_1 - w^*\|^2}{T} + \frac{\|w_1 - w^*\|\sigma}{\sqrt{T}}\right)$$

for some output $\hat{w}$ and

$$\eta = \min\left\{\frac{1}{2L}, \frac{\|w_1 - w^*\|}{\sqrt{T}\sigma^2}\right\}.$$

4.3.1 Lemma

Starting with the update rule:

$$\|w_{t+1} - w^*\|^2 = \|w_t - \eta_t g_t - w^*\|^2 = \|w_t - w^*\|^2 + \eta_t^2\|g_t\|^2 - 2\eta_t\langle g_t, w_t - w^*\rangle \quad (19)$$

$$\langle g_t, w_t - w^*\rangle = \frac{\|w_t - \eta_t g_t - w^*\|^2 - \|w_{t+1} - w^*\|^2}{2\eta_t} + \frac{\eta_t\|g_t\|^2}{2} \quad (20)$$

Where $\|w_{t+1} - w^*\|^2 = \|w_t - w^*\|^2 + \| - \eta_t g_t\|^2 - 2\eta_t\langle g_t, w_t - w^*\rangle$ by the quadratic inequality for the inner product.

Summing over t:

$$\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle = \sum_{t=1}^{T}\left(\frac{\|w_t - \eta_t g_t - w^*\|^2 - \|w_{t+1} - w^*\|^2}{2\eta_t} + \frac{\eta_t\|g_t\|^2}{2}\right) \quad (21)$$

$$= \frac{\|w_1 - \eta g_1 - w^*\|^2 - \|w_{T+1} - w^*\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2 \quad (22)$$

Where $\|w_{T+1} - w^*\|^2$ is dropped due to negativity. By telescoping, the following is obtained:

$$\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle \leq \frac{\|w_1 - \eta_1 g_1 - w^*\|^2}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2$$

$$\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle \leq \frac{\|w_1 - w^*\|^2}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2 \quad (23)$$

For SGD, we need to take expected values to 'account for' the randomness. Let z be i.i.d:

4.3.2 Right Side

$$\begin{aligned}\mathbb{E}_{z_{1:T}}[\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle] &\leq \mathbb{E}_{z_{1:T}}[\frac{\|w_1 - w^*\|^2}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2] \\ &\leq \frac{||w_1 - w^*||^2}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}[||g_t||^2]\end{aligned}$$

Now, $g_t = \nabla\ell_t(w, z_t)$. From the above, looking at $\mathbb{E}_{z_{1:T}}[||g_t||^2]$:

$$\begin{aligned}\mathbb{E}_{z_{1:t}}\left[\|g_t\|^2\right] &= \mathbb{E}_{z_{1:t-1}}\left[\mathbb{E}_{z_t}\left[\|g_t\|^2 \mid z_{1:t-1}\right]\right] \\ \mathbb{E}_{z_t}\left[\|g_t - \nabla\mathcal{L}_t\|^2 \mid z_{1:t-1}\right] &= \mathbb{E}_{z_t}\left[\|g_t - \nabla\mathcal{L}_t\|^2\right] \leq \sigma^2 \leftarrow \text{From SFO/ bounded variance} \\ \mathbb{E}_{z_t}\left[\|g_t\|^2\right] &= \|\nabla\mathcal{L}_t\|^2 + \mathbb{E}_{z_t}\left[\|g_t - \nabla\mathcal{L}_t\|^2\right] \leftarrow \text{from second momentum equality} \\ \mathbb{E}_{z_{1:t}}\left[\|g_t\|^2\right] &= \mathbb{E}_{z_{1:t-1}}\left[\|\nabla\mathcal{L}_t\|^2 + \mathbb{E}_{z_t}\left[\|g_t - \nabla\mathcal{L}_t\|^2\right] z_{1:t-1}\right] \leq \mathbb{E}_{z1:t}\left[\|\nabla\mathcal{L}_t\|^2\right] + \sigma^2\end{aligned}$$

Substituting back in:

$$\begin{aligned}\mathbb{E}_{z_{1:T}}[\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle] &\leq \frac{||w_1 - w^*||^2}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}[||\nabla\mathcal{L}||^2] + \sigma^2 \\ &\leq \frac{||w_1 - w^*||^2}{\eta} + \frac{\eta\sigma^2 T}{2} + \sum_{t=1}^{T}\mathbb{E}_{z_{1:T}}[||\nabla\mathcal{L}||^2]\end{aligned}$$

By Lipschitzness:

$$\mathbb{E}_{z_{1:T}}[||\nabla\mathcal{L}||^2] \leq \mathbb{E}_{z_{1:T}}[||L||^2] = L^2$$

So:

$$\mathbb{E}_{z_{1:T}}[\sum_{t=1}^{T}\langle g_t, w_t - w^*\rangle] \leq \frac{\|w_1 - w^*\|^2}{\eta} + \frac{\eta}{2}T(\sigma^2 + L^2) \tag{24}$$

4.3.3 Left Side

To simplify the left side: By i.i.d:

$$E_{z_1,\ldots,z_T}\left[\sum_{t=1}^{T}\langle\nabla\ell(w_t, z_t), w_t - w^*\rangle\right] = \sum_{t=1}^{T}E_{z_1:t}\left[\sum_{t=1}^{T}\langle\nabla\ell(w_t, z_t), w_t - w^*\rangle\right]$$

Then, using $E[X] = E_y[E_x[X|y]]$, we can get

$$\sum_{t=1}^{T}E_{z_1:t}\left[\sum_{t=1}^{T}\langle\nabla\ell(w_t, z_t), w_t - w^*\rangle\right] = \sum_{t=1}^{T}E_{z_1:t-1}\left[E_{z_t}\left[\langle\nabla\ell(w_t, z_t)\underbrace{\text{random}}_{\text{random}}, w_t - w^*\underbrace{\text{deterministic}}_{\text{deterministic}}\rangle|z_1,\ldots,z_{t-1}\right]\right]$$

Because w_t is not a function of z_t (i.i.d), we can get

$$\sum_{t=1}^{T}E_{z_1:t-1}\left[E_{z_t}\left[\langle\nabla\ell(w_t, z_t)\underbrace{\text{random}}_{\text{random}}, w_t - w^*\underbrace{\text{deterministic}}_{\text{deterministic}}\rangle|z_1,\ldots,z_{t-1}\right]\right] = \sum_{t=1}^{T}E_{z_1:t-1}\left[\langle E_{z_t}\left[\nabla\ell(w_t, z_t)|z_1,\ldots,z_{t-1}\right]\rangle, w_t -$$

The part above follows because we assumed SGD is unbiased. So,

$$\sum_{t=1}^{T}E_{z_1:t-1}\left[\langle E_{z_t}\left[\nabla\ell(w_t, z_t)|z_1,\ldots,z_{t-1}\right], w_t - w^*\rangle\right] = \sum_{t=1}^{T}E_{z_1:t-1}\left[\langle\nabla\mathcal{L}(w_t), w_t - w^*\rangle\right]$$

Using convexity, we have:

$$\sum_{t=1}^{T} E_{z_1:t-1}\left[\langle \nabla \mathcal{L}(w_t), w_t - w^* \rangle\right] \geq \sum_{t=1}^{T} E_{z_1:t-1}\left[\mathcal{L}(w_t) - \mathcal{L}(w^*)\right]$$

By telescoping:

$$\sum_{t=1}^{T} \mathbb{E}_{z_1:t-1}\left[\mathcal{L}(w_t) - \mathcal{L}(w^*)\right] = \mathbb{E}_{z_{1:T}}[\mathcal{L}(w_T) - \mathcal{L}^*]$$

4.3.4 Final Evaluation

Putting the two sides together:

$$\mathbb{E}_{z_{1:T}}[\mathcal{L}(w_T) - \mathcal{L}^*] \leq \frac{\|w_1 - w^*\|^2}{\eta} + \frac{\eta}{2} T(\sigma^2 + L^2)$$

Recall,

$$\eta = \min\left\{\frac{1}{2L}, \frac{\|w_1 - w^*\|}{\sqrt{T}\sigma^2}\right\}.$$

For $\eta = \frac{1}{2L}$:

$$\mathbb{E}_{z_{1:T}}[\mathcal{L}(w_T) - \mathcal{L}^*] \leq 2L\|w_1 - w^*\| + \frac{T}{4}(L + \frac{\sigma^2}{L})$$

For $\eta = \frac{1}{2L}$:

$$\frac{1}{2L} \leq \frac{\|w_1 - w^*\|}{\sqrt{T}\sigma^2}$$

$$\frac{\sqrt{T}\sigma^2}{2L} \leq ||w_1 - w^*||$$

Therefore, $L||w_1 - w^*||$ must be dominant term.

For $\eta = \frac{\|w_1 - w^*\|}{\sqrt{T}\sigma^2}$:

$$\mathbb{E}_{z_{1:T}}[\mathcal{L}(w_T) - \mathcal{L}^*] \leq \|w_1 - w^*\|\sqrt{T}\sigma(1 + \frac{1}{2} + \frac{L}{2})$$

For the final $\mathcal{O}$ notation, combine the two terms for each choice in η:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{z_1:t-1}\left[\mathcal{L}(w_t) - \mathcal{L}(w^*)\right] \leq (\frac{L\|w_1 - w^*\|}{T} + \frac{\|w_1 - w^\cdot\|\sigma}{\sqrt{T}})$$

$$\mathbb{E}_{z_{1:T}}[\mathcal{L}(w_T) - \mathcal{L}^*] = \mathcal{O}(\frac{L\|w_1 - w^*\|}{T} + \frac{\|w_1 - w^\cdot\|\sigma}{\sqrt{T}})$$

5 Question 5

5.1 Code

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#sigma
def sigma(x):
  return np.maximum(x,0)

# derivative of sigma
def der_sigma(x):
  return np.where(x>0,1,0)

#loss fn
def L(y_true, y_pred):
    return np.mean(np.square(y_true-y_pred))

# gradient
def grad_L(y, w, x):
  pred = np.matmul(x,w)
  return np.array(-2*np.dot(x.T,((y-sigma(pred))*der_sigma(pred))))

# sample with replacement
def samp_with_rep(N,x,y):
  idx = np.random.randint(0,N)
  return x[idx,:],y[idx]

# sample without replacement
def samp_without_rep(N,x,y):
  idx = np.random.permutation(N)
  return (x[idx],y[idx])

# list out constants
d = 200
N = 1000
rng = np.random.default_rng()
e = rng.normal(0.0,0.05,size=(N,1))
w_init = rng.normal(0,1.5,size=(d,1))
w_act = rng.normal(0,1,size=(d,1))
x = rng.uniform(0,1,size=(N,d))
eta = 0.001 #learning rate
T = N #max epoch for sample w/out replace
q = 10 #q values for c and d
outlier_factor = 1.2  # Control the magnitude of outliers
outlier_indices = np.random.choice(N, size=int(0.01 * N), replace=False) #1% outliers
y = np.array(sigma(np.matmul(x,w_act)) + e)
y[outlier_indices] = outlier_factor * (y[outlier_indices])
errtol = 1e-6 #tolerance for stopping

def SGD(w,x,y,samp_idx,eta,errtol):
  x_jj = np.atleast_2d(x[samp_idx,:])
  y_jj = np.atleast_2d(y[samp_idx]).T

  w = w - eta*grad_L(y_jj,w,x_jj)
  loss = L(y,sigma(np.matmul(x,w)))
  return loss,np.abs(loss)<errtol,w

#parts a,b,c,d initialized
w_a = w_init
w_b = w_init
w_c = w_init
w_d = w_init
a_conv = False
printedA = False
```

```
b_conv = False
printedB = False
c_conv = False
printedC = False
d_conv = False
printedD = False
Loss_a = []
Loss_b = []
Loss_c = []
Loss_d = []
Loss_init = L(y,sigma(np.matmul(x,w_a)))
#initialize dataframes for sorting
df_c = pd.DataFrame(x,columns = [f'feature{ii}' for ii in range(d)])
df_c['target'] = y.flatten()
df_d = df_c.copy(deep = True) #deep copy
#initial losses
Loss_a.append(Loss_init)
Loss_b.append(Loss_init)
Loss_c.append(Loss_init)
Loss_d.append(Loss_init)

#part a, with replacement
samp_idx_a = np.random.choice(N,size=T,replace=True)
#part b, without replacement
samp_idx_b = np.random.choice(N,size=T,replace=False)
for ii in range(T):
  print(f'Epoch: {ii}')
  #check if each has converged; then check if I need to print anything
  # append the loss from this iteration to the Loss list for this epoch
  if not a_conv:
    loss_a,a_conv,w_a = SGD(w_a,x,y,samp_idx_a[ii],eta,errtol)
    Loss_a.append(loss_a)
    print(f"A Loss: {loss_a}")
  elif not printedA:
    print(f"A Converged after {ii} epochs")
    printedA = True
  #part b, apply sample w/out replacement
  if not b_conv:
    loss_b,b_conv,w_b = SGD(w_b,x,y,samp_idx_b[ii],eta,errtol)
    Loss_b.append(loss_b)
    print(f'B Loss: {loss_b}')
  elif not printedB:
    print(f"B Converged after {ii} epochs")
    printedB = True

  #part c, sorting by losses
  if not c_conv:
    df_c['pred'] = sigma(np.matmul(df_c.iloc[:,:d].values,w_c))
    df_c['loss'] = np.square(df_c['target']-df_c['pred']) #find loss for each sample
    max_loss_idx = df_c['loss'].nlargest(q).index #sort by loss
    loss_c,c_conv,w_c = SGD(w_c,df_c.iloc[:,:d].values,df_c['target'].values,max_loss_idx,
        eta,errtol)
    Loss_c.append(loss_c)
    print(f"C Loss: {loss_c}")
  elif not printedC:
    print(f"C Converged after {ii} epochs")
    printedC = True

  #part d, sorting by norm of sample gradients
  if not d_conv:
    df_d['grad_norm'] = df_d.apply(lambda row: np.linalg.norm(grad_L(np.atleast_2d(row['
        target']),w_d,np.atleast_2d(row.iloc[:d].values))),axis=1) #apply grad_L to each row
    max_grad_idx = df_d['grad_norm'].nlargest(q).index #sort by gradient norm
    loss_d,d_conv,w_d = SGD(w_d,df_d.iloc[:,:d].values,df_d['target'].values,max_grad_idx,
        eta,errtol)
    Loss_d.append(loss_d)
    print(f"D Loss: {loss_d}")
  elif not printedD:
    print(f"D Converged after {ii} epochs")
    printedD = True
#end T

epochs = np.arange(1,T+1)
plt.plot(epochs,Loss_a[1:],label='a')
plt.plot(epochs,Loss_b[1:],label='b')
plt.plot(epochs,Loss_c[1:],label='c')
plt.plot(epochs,Loss_d[1:],label='d')
plt.title(f'Loss vs. Epochs-Learning Rate ={eta}')
plt.xlabel('Epochs')
plt.ylabel('Loss-MSE')
plt.legend()
plt.show()
#plt.semilogx(epochs,Loss_a[1:],label='a')
#plt.semilogx(epochs,Loss_b[1:],label='b')
#plt.semilogx(epochs,Loss_c[1:],label='c')
#plt.semilogx(epochs,Loss_d[1:],label='d')
#plt.title(f'Loss vs. Epochs-Learning Rate ={eta}')
#plt.xlabel('Epochs (log scaled)')
#plt.ylabel('Loss-MSE')
#plt.legend()
#plt.show()

plt.loglog(epochs,Loss_a[1:],label='a')
plt.loglog(epochs,Loss_b[1:],label='b')
plt.loglog(epochs,Loss_c[1:],label='c')
plt.loglog(epochs,Loss_d[1:],label='d')
plt.title(f'Loss vs. Epochs-Learning Rate ={eta}')
plt.xlabel('log(Epochs)')
plt.ylabel('log(Loss-MSE)')
plt.legend()
plt.show()
```

5.2 Results

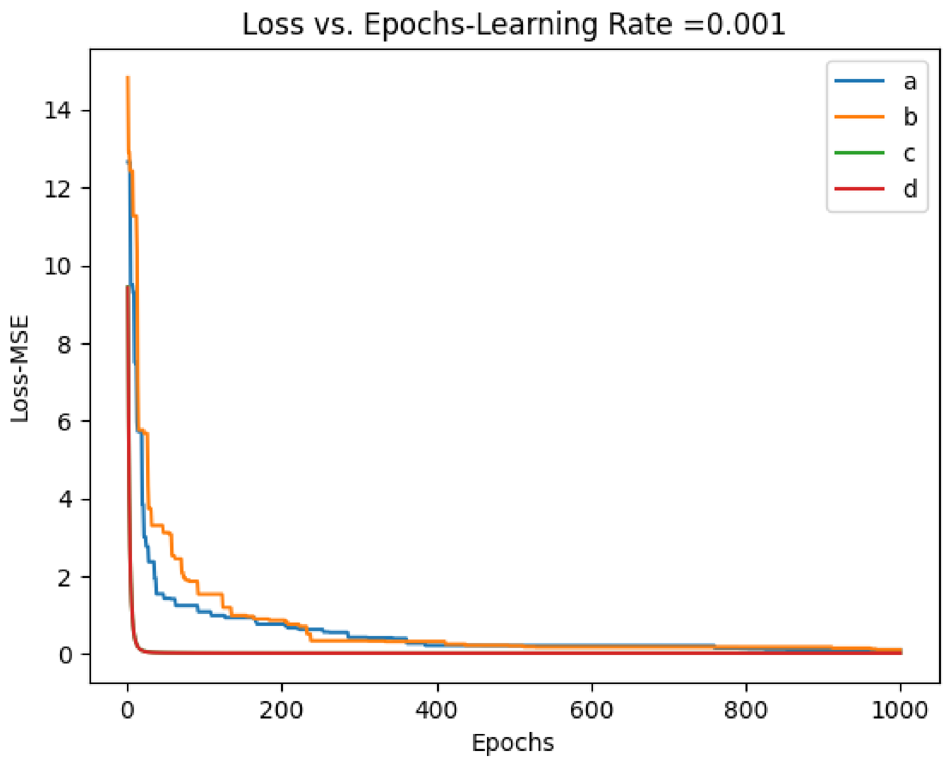

Figure 1: Linear Scale

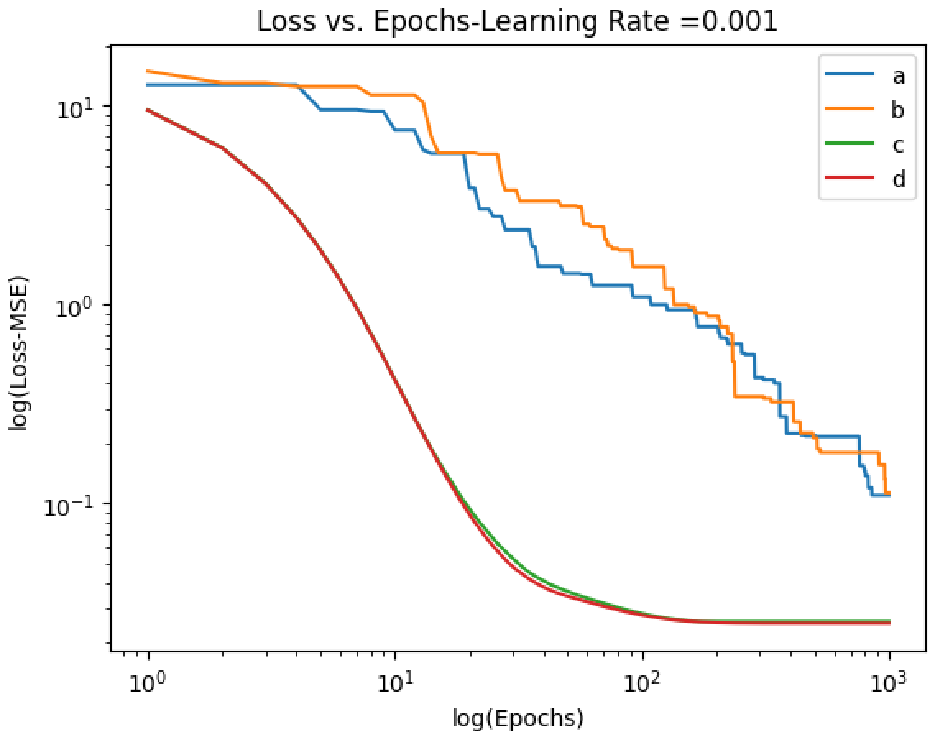

Figure 2: Log-Log Scale