Efficient Ordering of Stochastic Gradient Descent

End-Term Presentation

April 17, 2023

Team Members

1. EE19B002: Aditya Sharma
2. EE19B022: Gautham Govind A
3. EE19B038: Manikandan Sritharan
4. EE19B142: Akshat Bhandari

Table of Contents

Table of Contents

Up Next...

RWSGD

- In one particular version of SGD, termed Random Walk SGD (RWSGD), we decide the input sequence $\{X_t\}_{t\geq 0}$ through a random walk on a Markov chain.
- The key benefit of this setting is that each sample/node can update the parameters using local gradient and then pass the updated parameters instead of divulging the gradient itself.
- Existing analysis of RWSGD in literature focuses primarily on **Finite-time bounds** which provide upper bounds for either $\mathbb{E}[f(\tilde{\theta}_t) - f(\theta^*)]$ or $\mathbb{E}[||\nabla f(\theta_t)||_2^2]$.

RWSGD: Analysis

- The bounds are of the form:

$$\mathbb{E}[||\nabla f(\theta_t)||_2^2] \leq \mathcal{O}\left(\frac{\max\{M, 1/\log(1/\beta)\}}{t^{1-\alpha}}\right)$$

  where M depends on local gradients at the minimizer and β is the Second Largest Eigenvalue Modulus (SLEM) of the transition matrix.
- The finite time bounds depend on the value of β or SLEM of the transition matrix.
- The general intuition over the years has been that Markov chains with smaller SLEM lead to faster convergence of the SGD iterations.

Generalised SGD Update rule

- Given a general stationary distribution π the objective function is rewritten as follows,

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} F(\theta, i) = \mathbb{E}_{X\sim\pi}[G(\theta, X)],$$

  where function $G(\theta, i) \triangleq \frac{1}{n\pi_i} F(\theta, i)$ for any $\theta \in \Theta, i \in [n]$.
- If π was an uniform distribution, $G(\theta, i) = F(\theta, i)$ for all $\theta \in \Theta$.

Generalised SGD Update rule

- If the input sequence is instead a simple random walk on a connected graph $G(V, E)$ with $V = [n]$, we have $\pi \propto d$ (degree of nodes), and $G(\theta, i) = \frac{1^T d}{n d_i} F(\theta, i)$ for all $\theta \in \Theta, i \in V$.
- The generalized update rule will then be,

$$\theta_{t+1} = Proj_{\Theta}(\theta_t - \gamma_{t+1} \nabla G(\theta_t, X_{t+1}))$$

The Big Result

- Using the aforementioned update rule and the regularity conditions, the authors then show that:

$$\theta_t \xrightarrow[t\to\infty]{a.s.} \theta^*, \text{ and } \frac{\theta_t - \theta^*}{\sqrt{\gamma_t}} \xrightarrow[t\to\infty]{dist} \mathcal{N}(0, \mathbf{V}_X)$$

- The authors then proceed to show that the Loewner ordering of $\mathbf{V}_X$ can be obtained from the ordering of the scalar asymptotic variance $\sigma_X^2(g)$ for some function g.

Up Next...

Asymptotic bounds

- At this point, it is important to examine the importance of these results practically.
- The application of the result to cases where we are interested only in the asymptotic behaviour is rather straightforward.
- Intuitively, a "smaller" covariance matrix (in the Loewner ordering sense) translates to a lower number of samples required to estimate the true parameter up to a required precision.

Asymptotic bounds

- To put it more concretely, it can be seen that

$$\mathbf{w}^{\mathbf{T}} \frac{(\theta_t - \theta^*)}{\sqrt{\gamma_t}} \xrightarrow[t\to\infty]{dist} \mathcal{N}(0, \mathbf{w}^{\mathbf{T}} \mathbf{V}_X \mathbf{w})$$

  where $\mathbf{w}$ is a vector of weights.
- Therefore asymptotically, we can estimate the confidence interval as:

$$P\left(\frac{\mathbf{w}^{\mathbf{T}}(\theta_t - \theta^*)}{\sqrt{\gamma_t \mathbf{w}^{\mathbf{T}} \mathbf{V}_X \mathbf{w}}} > \alpha\right) \approx \frac{1}{\sqrt{2\pi}} \int_\alpha^\infty e^{-x^2/2} dx$$

Asymptotic bounds

- Clearly, for a smaller and more precise confidence interval, we require the value of $\mathbf{w}^{\mathbf{T}}\mathbf{V}_X\mathbf{w}$ to be small.
- This is exactly what Loewner ordering of $\mathbf{V}_X$ gives us; a smaller covariance matrix in the Loewner sense gives a smaller value for the above described quantity.

Finite-time performance

- The objective now is to see if the techniques developed so far can be used to analyze the finite-time performance of the algorithms.
- In general, **the answer is No**.
- There is no clear connection between finite-time MSE and asymptotic variance in the general case; the authors only claim a "potential" smaller MSE depending on the objective function.

Finite-time performance

driven by different stochastic inputs. Since this is achieved via Loewner ordering, it also leads to smaller confidence intervals in the long run as mentioned earlier, as well as potentially smaller MSE[8] depending on the objective function.

Figure: Statement regarding finite-time mse

Finite-time performance

- However, it is possible to obtain such a connection for the specific case of quadratic objective functions.
- Essentially they satisfy the linear stochastic approximation of the form:

$$\theta_{t+1} = \theta_t - \gamma_{t+1}(\mathbf{A}\theta_t - \mathbf{b}(X_t + 1))$$

- For this particular case, it is possible to establish a connection between finite-time MSE and asymptotic covariance matrix; however a general result is not available as of now.

Up Next...

General comments

- Though the theoretical sections of the paper are very well written, the given descriptions of the experiments performed are rather limited.
- The exact implementational details are not provided in many cases; in such cases we have resorted to secondary sources.

Review of results

- The simulations performed serve the purpose of demonstrating the following results presented in the paper:
  - Smaller asymptotic variance indicates a more efficient input sequence for SGD.
  - Non-Backtracking Random Walk (NBRW) is more efficient than Simple Random Walk (SRW) as input to SGD.
  - Shuffling based methods are more efficient than IID sampling as input to SGD.

Generalized SGD update rule

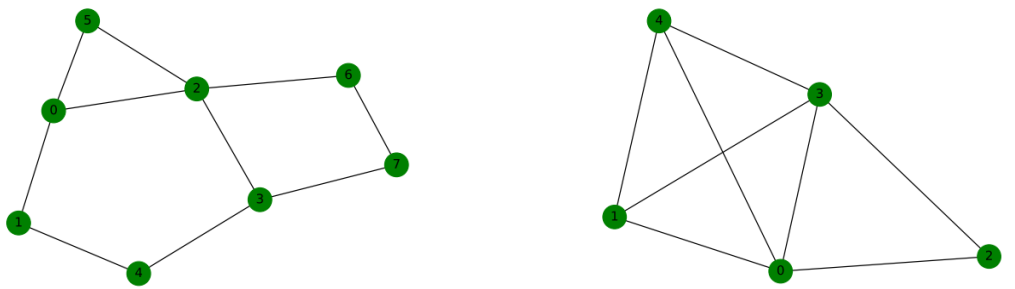
- As we have seen, the authors construct a modified SGD update rule of the following form for obtaining the results proved in the paper:

$$\theta_{t+1} = Proj_{\Theta}(\theta_t - \gamma_{t+1} \nabla G(\theta_t, X_{t+1}))$$

- For the cases under consideration, we can relate G and F as follows:
  - SRW and NBRW: $G(\theta, i) = \frac{\mathbf{1}^T \mathbf{d}}{n d_i} F(\theta, i)$
  - IID sampling and shuffling: $G(\theta, i) = F(\theta, i)$

Experiment

- To understand the significance of Asymptotic Variance, we compare three types of Markov Processes (MHRW, Modified MHRW, FMMC) on the following two graphs



(a) 8-node Graph G1 (b) 5-node Graph G2

Figure: Topology of Graphs

MHRW

- The Metropolis-Hasting Random Walk is characterized by the following transition probability matrix.

$$P(i,j) = \begin{cases} \min\left\{\frac{1}{d_i}, \frac{1}{d_j}\right\}, & j \in N(i) \\ 1 - \sum_{j \in N(i)} P(i,j), & j = i \end{cases}$$

where d_i is the degree of node i and $N(i)$ is the set of node i's neighbors.

Modified MHRW

- **Peskun Ordering**: For two finite, ergodic, reversible Markov chains $\{X_t\}_{t\geq 0}, \{Y_t\}_{t\geq 0}$ on the state space $\mathcal{V}$ with transition matrices $\mathbf{P}_X, \mathbf{P}_Y$ having the same stationary distribution π, it is said that $\mathbf{P}_Y$ dominates $\mathbf{P}_X$ off the diagonal, written as $\mathbf{P}_X \preceq \mathbf{P}_Y$ if $P_X(i,j) \leq P_Y(i,j)$ for all $i,j \in \mathcal{V}$ and $i \neq j$
- **Lemma**: If $\mathbf{P}_X \preceq \mathbf{P}_Y$, then $\sigma^2_X(g) \geq \sigma^2_Y(g)$ for any scalar-valued function g with $\mathbb{E}_{\boldsymbol{\pi}}\left(g^2\right) < \infty$, that is, $\{Y_t\}_{t\geq 0}$ is more efficient than $\{X_t\}_{t\geq 0}$.
- Notice that the chain $\{Y_t\}_{t\geq 0}$ is more efficient than $\{X_t\}_{t\geq 0}$, and has lower self-transition probabilities.

Modified MHRW

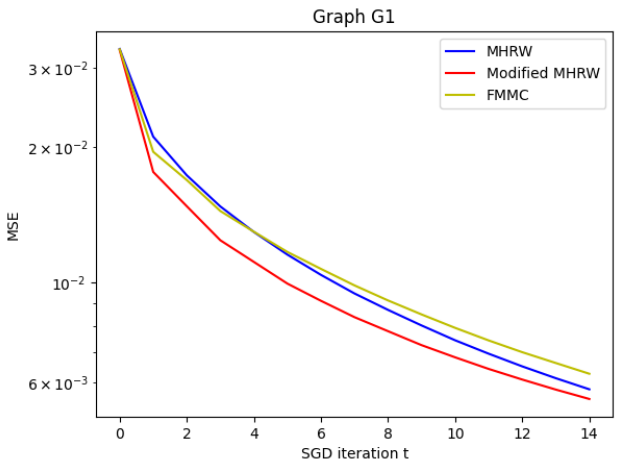
- We can construct a more efficient chain by reducing the self transition probabilities of the MHRW and redistributing it to off-diagonal entries, in a way that the resulting matrix is doubly-stochastic.
- The above problem can be modelled as the following Convex Optimization Problem.
- Minimize $\|(P + D) \cdot I\|_1$ such that
  - $(P + D)\mathbf{1} = \mathbf{1}$
  - $1^T(P + D) = 1^T$

  where P is the initial transition probability matrix (constant) and D is the additional change (variable).

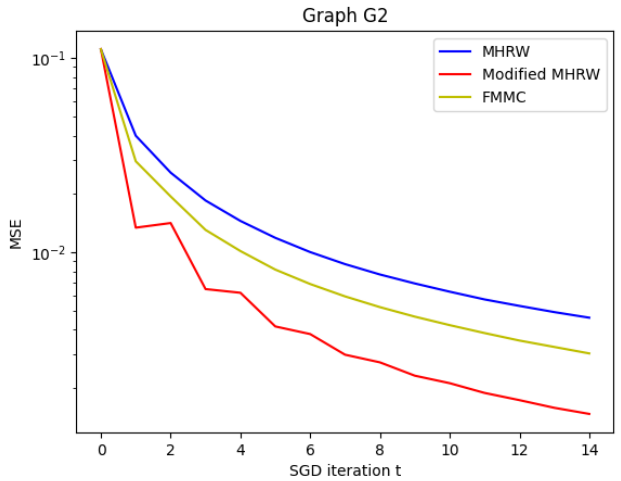
FMMC

- The Fastest Mixing Markov Chain is the Markov Chain with the smallest SLEM.
- We can compute the FMMC transition matrix by solving the following Convex Optimization Problem
- Minimize $\left\| P - (1/n)\mathbf{1}\mathbf{1}^T \right\|_2$ such that
  - $P \geq 0$
  - $P\mathbf{1} = \mathbf{1}$
  - $P = P^T$

Results



(a) Graph G1



(b) Graph G2

Figure: Plots for Graphs

Logistic loss

- We consider the strictly-convex L_2-regularized logistic regression loss function:

$$\tilde{f}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{x}_i^T \theta)) + \frac{1}{2}||\theta||_2^2$$

- We choose $(\mathbf{x}_i, y_i)$ from the CIFAR-10 dataset.
- We use the graph "Dolphins" (62 nodes) for performing RWSGD.

Setup

- Note that since the graph has only 62 nodes, we can only train 62 datapoints as each sample has to be assigned a node. So we have $n = 62$.
- Further, the authors use a 6x6x3 cropped version of CIFAR-10 images. Since the exact mechanism of cropping isn't specified, we simple resize the 32x32x3 image to 6x6x3.
- Instead of using all 10 classes, we select two classes randomly, assign the label of $+1$ and -1 to these classes so that $\tilde{f}(\theta)$ obtains the input in the desired format.

Setup

- To plot the error $\mathbb{E}||\theta_t - \theta^*||_2^2$, it is necessary to obtain θ^*, the true optimizer.
- Since $\tilde{f}(\theta)$ cannot be optimized analytically, we rely on numerical techniques provided by the *scipy* library to find θ^*.
- We approximate the expectation by averaging the error over 50 runs.

Gradient

- We compute the gradient of the loss considering one sample as follows:

$$\nabla_\theta F(\theta, \mathbf{x_i}) = \nabla \left(\log(1 + \exp(-y_i \mathbf{x}_i^T \theta)) + \frac{1}{2} \theta^T \theta \right)$$
$$= \frac{-\exp(-y_i \mathbf{x}_i^T \theta)}{1 + \exp(-y_i \mathbf{x}_i^T \theta)} y_i \mathbf{x}_i + \frac{1}{2} \theta$$

Results

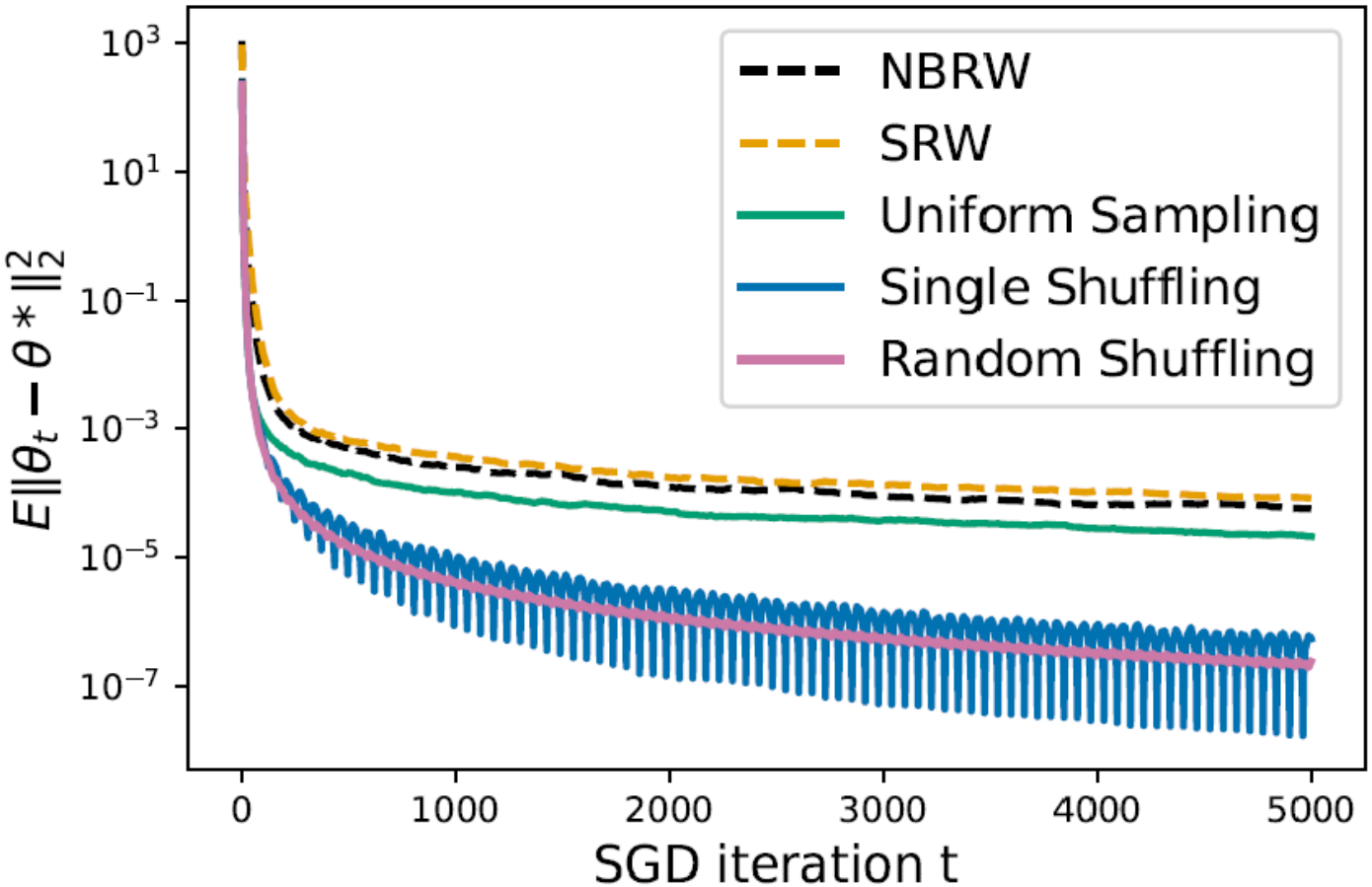


Figure: Error plot (given)

Results

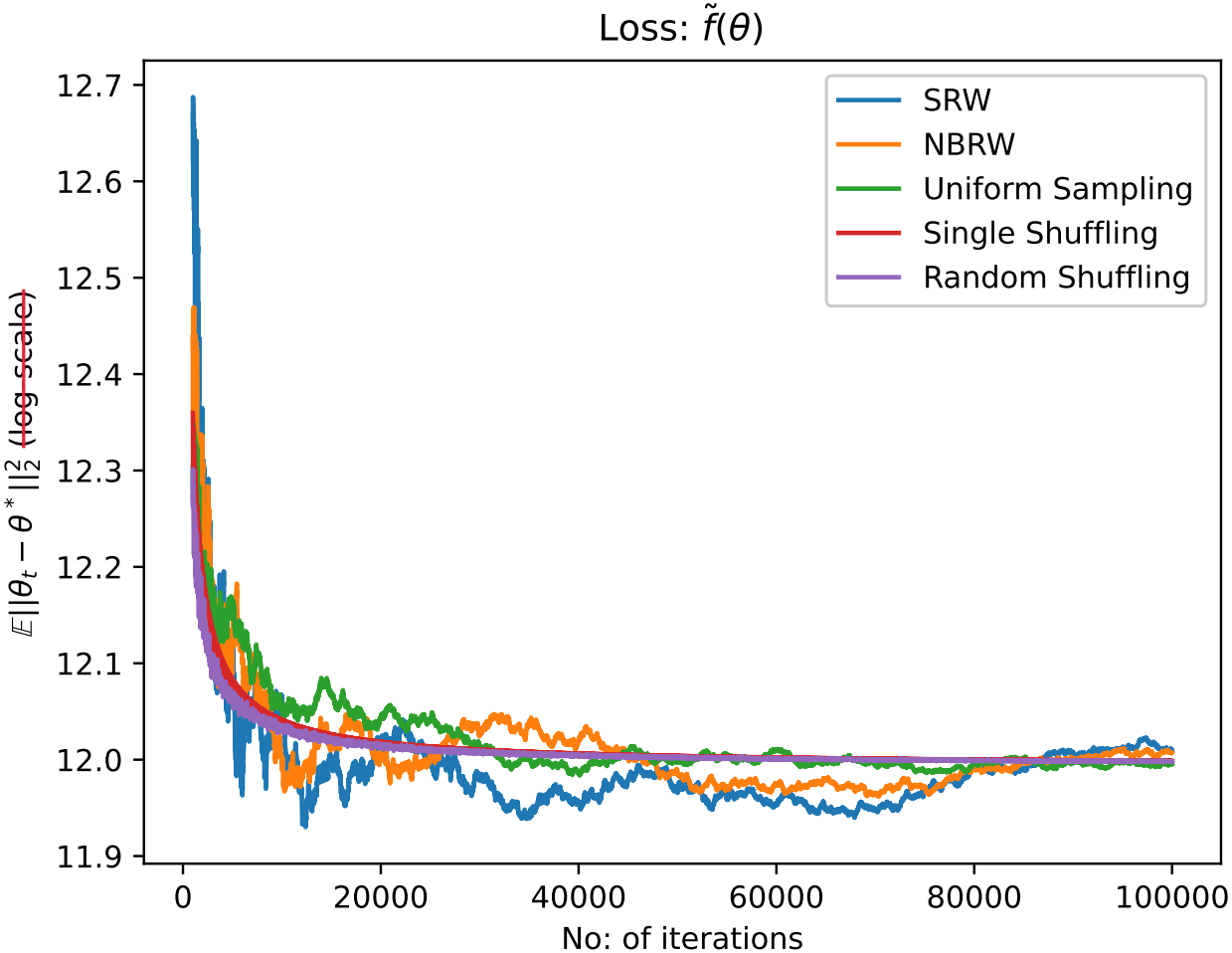


Figure: Error plot (ours)

Observations

- The asymptotic structure of the plots look similar; we observe that as the iteration count becomes high, the ordering of the errors are similar for both plots.
- We observe that the finite-time performance of the variants are **not similar** in the two plots. This is to be expected because:
  1. We are estimating the expected error by sampling. This is bound to introduce some noise. It might be possible to fix this by increasing the number of samples taken.
  2. Even if we were to obtain the exact value for expected error, since the loss function is not quadratic, there is no theoretical reason to expect any correlation between asymptotic behaviour and finite-time behaviour as we have seen before.

Sum of non-convex functions

- We consider the loss function:

$$\hat{f}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \theta^T(\mathbf{a}_i\mathbf{a}_i^T + \mathbf{D}_i)\theta + \mathbf{b}^T\theta$$

- We generate $\mathbf{a}_i$s as random vectors in $\mathbb{R}^{10}$ such that $\sum_{i=1}^{n} \mathbf{a}_i\mathbf{a}_i^T$ is invertible and $\mathbf{b}$ as a random vector in $\mathbb{R}^{10}$.
- $\mathbf{D}_i$s are diagonal matrices in $\mathbb{R}^{10\text{x}10}$; half of them (chosen randomly) are equal to $1.1\mathbb{I}$ whereas the other half take the value $-1.1\mathbb{I}$, where $\mathbb{I}$ is the 10x10 identity matrix.

Sum of non-convex functions

- Note that for the cases where $\mathbf{D}_i$ is $-1.1\mathbb{I}$, the function:

$$\theta^T(\mathbf{a}_i\mathbf{a}_i^T + \mathbf{D}_i)\theta + \mathbf{b}^T\theta$$

  could possibly be non-convex.
- However, since $\sum_{i=1}^{n} \mathbf{D}_i = 0$, the function $\hat{f}(\theta)$ itself is always convex as the hessian can be written as $\sum_{i=1}^{n}(\mathbf{a}_i\mathbf{a}_i^T)/n$ which is a PSD matrix.

Computing the gradient

- We compute the gradient of the loss considering one sample as follows:

$$\begin{aligned}\nabla_\theta F(\theta, \mathbf{x_i}) &= \nabla(\theta^T(\mathbf{a}_i\mathbf{a}_i^T + \mathbf{D}_i)\theta + \mathbf{b}^T\theta) \\ &= 2(\mathbf{a}_i\mathbf{a}_i^T + \mathbf{D}_i)\theta + \mathbf{b}\end{aligned}$$

- It is also possible to compute θ^* analytically in this case by setting the gradient to zero. This yields:

$$\theta^* = \frac{-1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{a}_i\mathbf{a}_i^T\right)^{-1}\mathbf{b}$$

as $\sum_{i=1}^{n}\mathbf{D}_i = 0$.

Setup

- We begin by simulating this case on the same graph Dolphins (62 nodes).
- The averaging is done over 100 runs.

Results

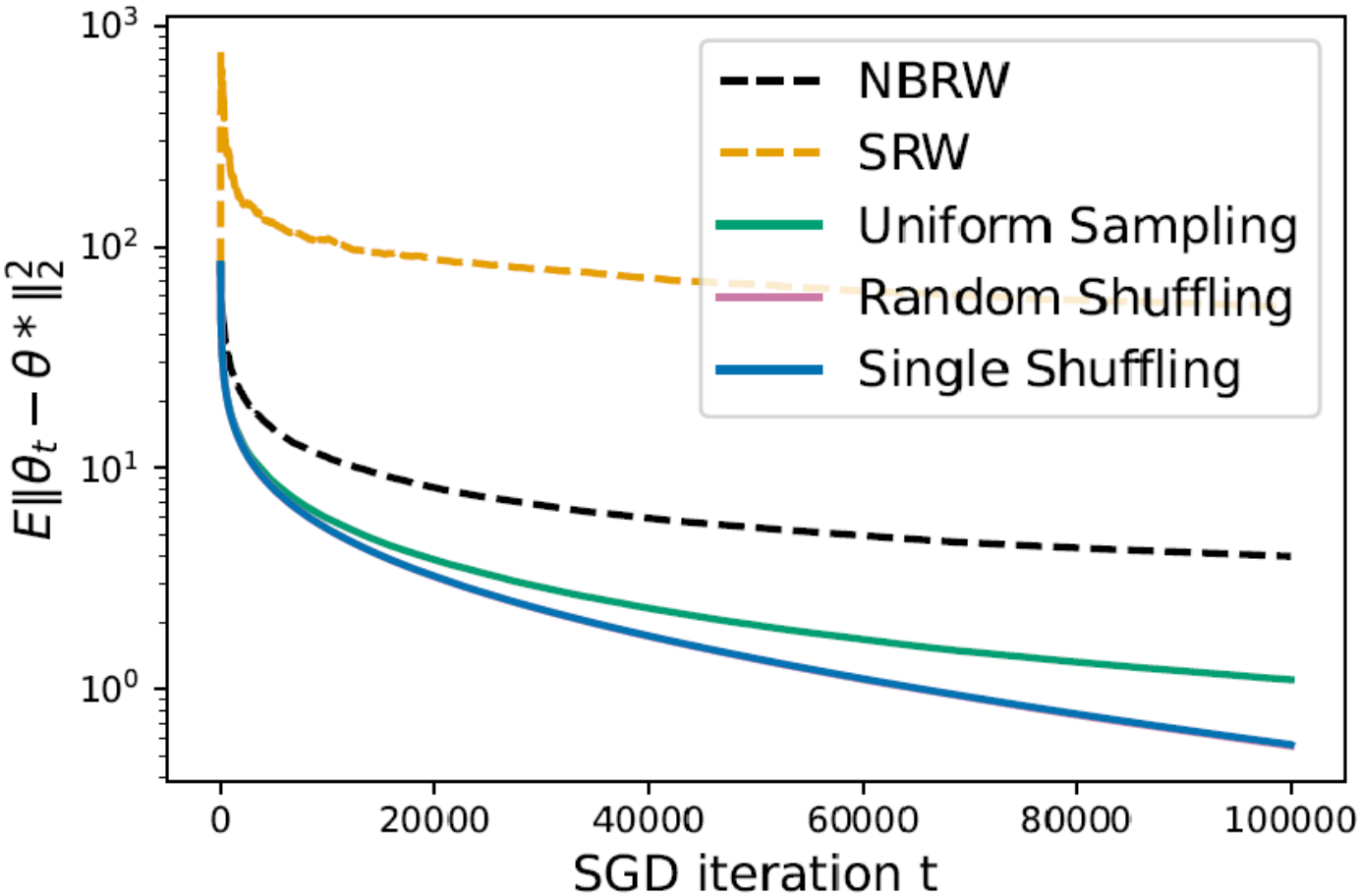


Figure: Error plot (given)

Results

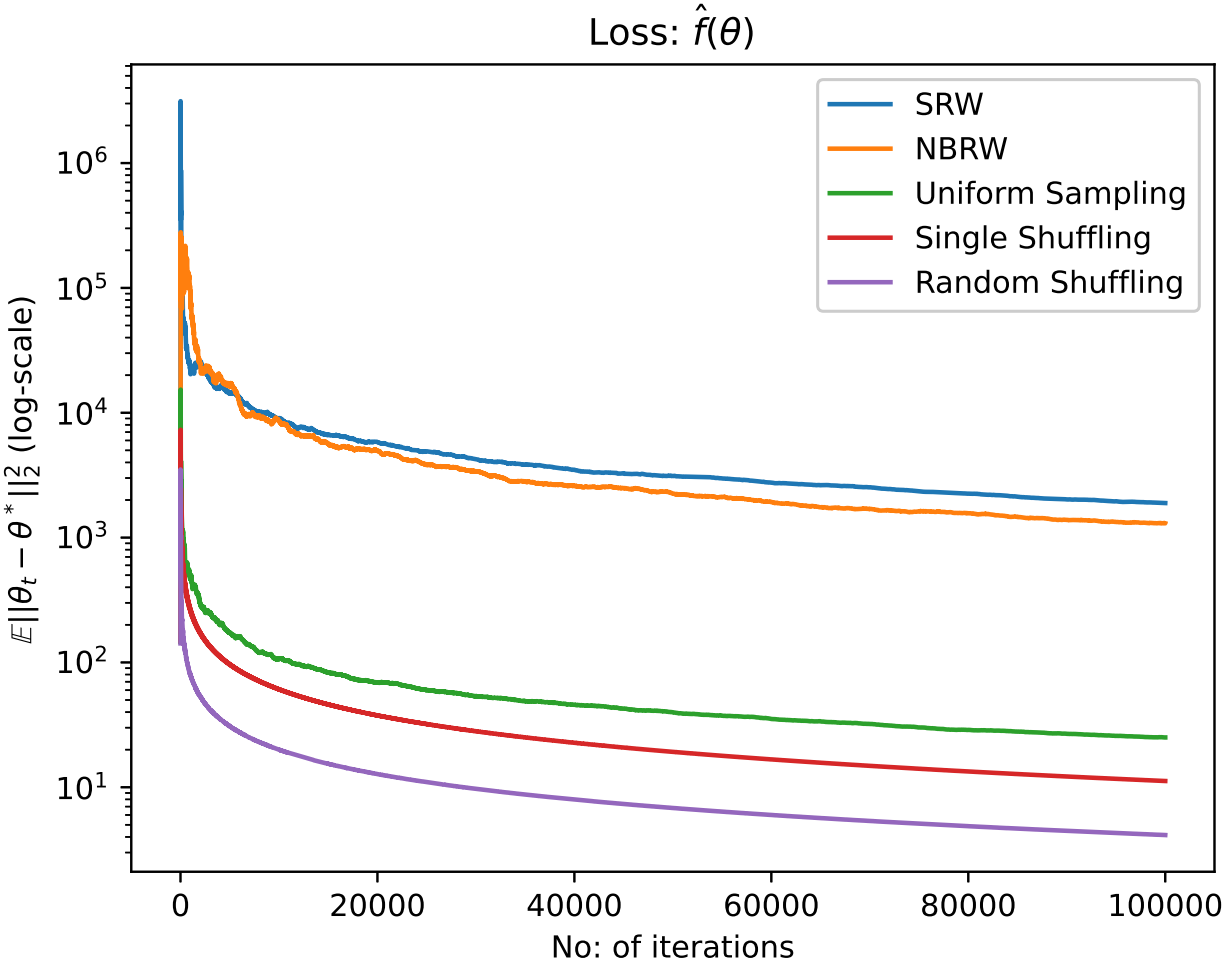


Figure: Error plot (ours)

Observations

- It can be observed that the curves in both plots obey the same order across all timesteps.
- It is also worthwhile to note that the objective function here is quadratic and hence we expect better performance in finite-time as well, which is what we observe in this case.

Macaque-Rhesus graph

- We now try out the same loss function $\hat{f}(\theta)$ on the graph Macaque-Rhesus (242 nodes). Note that this case has not been analyzed in the paper.
- Everything else is same, except for the fact that we now generate 242 samples as we have 242 nodes.

Results

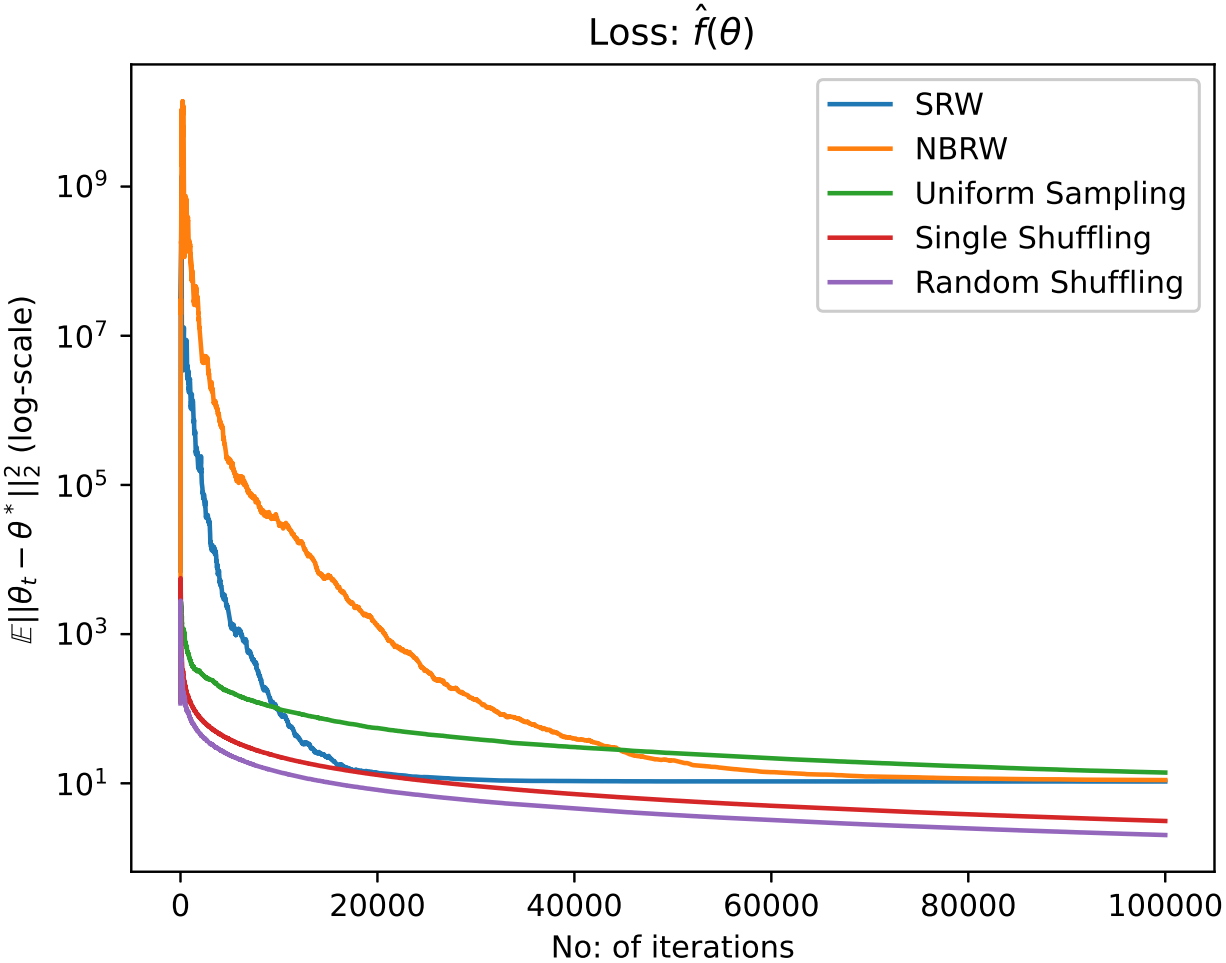


Figure: Error plot (ours)

Observations

- As the curves enter the asymptotic regime, we observe the predicted orders of NBRW $<$ SRW and Shuffling $<$ IID Sampling.
- However, note that unlike the previous cases, in this case NBRW and SRW seem to perform better than IID sampling. But this occurrence does not break the theory as there are no results in the paper comparing the asymptotic variance of SRW/NBRW and IID sampling.
- We can also see that the finite-time performance depends on factors like the structure of the graph as we obtain different plots for the same loss function but with different underlying graph structures.

Up Next...

Concluding remarks

- Through the duration of this semester, we have managed to understand the theoretical results and their implications presented in the paper 'Efficient Ordering of Stochastic Gradient Descent'.
- We have also modelled the simulations by writing the code from scratch and have recreated the results as best as we could. The code is available in this github repo.

THANK YOU!