Machine Learning and Neural Networks (MATH3431) Epiphany term, 2024

Handout 5: Stochastic gradient descent

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce the stochastic gradient descent (motivation, description, practical tricks, analysis in the convex scenario, and implementation).

Reading list & references:

(1) Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
- Ch. 14.3 Stochastic Gradient Descent (SGD), 14.5 Variants, 14.5 Learning with SGD

(2) Bottou, L. (2012). Stochastic gradient descent tricks. In Neural networks: Tricks of the trade (pp. 421-436). Springer, Berlin, Heidelberg.

(3) Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." Advances in neural information processing systems 26 (2013).

(4) Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12, no. 7 (2011).

Code of the Examples can be found in `https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2024/tree/main/Lecture_handouts/code/05.Stochastic_gradient_descent`

1. Motivations for stochastic gradient descent

Problem 1. Consider a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$. Learning may involve the computation of the minimizer $w^* \in \mathcal{H}$, where $\mathcal{H}$ is a class of hypotheses, of the risk function (RF) $R(w) = \mathrm{E}_{z\sim g}(\ell(w,z))$ given an unknown data generating model $g(\cdot)$ and using a known tractable loss $\ell(\cdot,\cdot)$; that is

$$w^* = \arg\min_{\forall w\in\mathcal{H}} (R_g(w)) = \arg\min_{\forall w\in\mathcal{H}} (\mathrm{E}_{z\sim g}(\ell(w,z))) \tag{1.1}$$

Note 2. Gradient descent (GD) cannot be directly utilized to address Problem 1 (i.e., minimize the Risk function) because g is unknown, and because (1.1) involves an integral which may be computationally intractable. Instead it aims to minimize the ERF $\hat{R}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i)$ which ideally is used as a proxy when data size n is big (big-data).

Note 3. The implementation of GD may be computationally impractical even in problems where we need to minimize an ERF $\hat{R}_n(w)$ if we have big data ($n \approx$big). This is because GD requires the recursive computation of the exact gradient $\nabla\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(w, z_i)$ using all the data $\{z_i\}$ at each iteration. That may be too slow.

Note 4. Stochastic gradient descent (SGD) aims at solving (1.1), and overcoming the issues in Notes 2 & 3 by using an unbiased estimator of the actual gradient (or some sub-gradient) based on a sample (a single example or a set of examples) properly drawn from g.

2. Stochastic gradient descent

Problem 5. For the sake of notation simplicity and generalization, we present Stochastic Gradient Descent (SGD) in the following minimization problem

$$w^* = \arg\min_{w\in\mathcal{H}} \left(f\left(w\right)\right) \tag{2.1}$$

where here $f : \mathbb{R}^d \to \mathbb{R}$, and $w \in \mathcal{H} \subseteq \mathbb{R}^d$; $f\left(\cdot\right)$ is the unknown function to be minimized, e.g., $f\left(\cdot\right)$ can be the risk function $R_g\left(w\right) = \mathrm{E}_{z\sim g}\left(\ell\left(w, z\right)\right)$.

Algorithm 6. *Stochastic Gradient Descent (SGD) with learning rate $\eta_t > 0$ for the solution of the minimization problem 5*

For $t = 1, 2, 3, ...$ iterate:

(1) *compute*

$$w^{(t+1)} = w^{(t)} - \eta_t v_t, \tag{2.2}$$

where v_t is a random vector such that $E\left(v_t|w^{(t)}\right) \in \partial f\left(w^{(t)}\right)$

(2) *terminate if a termination criterion is satisfied, e.g.*

$$\textit{If } t \geq T_{\max} \textit{ then STOP}$$

Note 7. If f is differentiable at $w^{(t)}$, it is $\partial f\left(w^{(t)}\right) = \left\{\nabla f\left(w^{(t)}\right)\right\}$. Hence v_t is such as $\mathrm{E}\left(v_t|w^{(t)}\right) = \nabla f\left(w^{(t)}\right)$ in Algorithm 6 step 1.

Note 8. Assume f is differentiable (for simplicity). To compare SGD with GD, we can re-write (2.2) in the SGD Algorithm 6 as

$$w^{(t+1)} = w^{(t)} - \eta_t \left[\nabla f\left(w^{(t)}\right) + \xi_t\right], \tag{2.3}$$

where

$$\xi_t := v_t - \nabla f\left(w^{(t)}\right)$$

represents the (observed) noise introduced in (2.2) by using a random realization of the exact gradient.

Note 9. Given T SGD algorithm iterations, the output of SGD can be (but not a exclusively)

(1) the average (after discarding the first few iterations of $w^{(t)}$ for stability reasons)

$$w_{\mathrm{SGD}}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)} \tag{2.4}$$

(2) or the best value discovered

$$w_{\mathrm{SGD}}^{(T)} = \arg\min_{\{w_t\}} \left(f\left(w^{(t)}\right)\right)$$

(3) or the last value discovered

$$w_{\mathrm{SGD}}^{(T)} = w^{(T)}$$

Note 10. SGD output converges to a local minimum, $w_{\text{SGD}}^{(T)} \to w_*$ (in some sense), under different sets of regularity conditions –Section 3 has a brief analysis. To achieve this, Conditions 11 on the learning rate are rather inevitable and should be satisfied.

Condition 11. Regarding the learning rate (or gain) $\{\eta_t\}$ should satisfy conditions

(1) $\eta_t \geq 0$,
(2) $\sum_{t=1}^{\infty} \eta_t = \infty$
(3) $\sum_{t=1}^{\infty} \eta_t^2 < \infty$

Note 12. The popular learning rates $\{\eta_t\}$ in Note 8 in Handout 4 "Gradient descent" satisfy Condition 11 and hence can be used in SGD too. Once parameterized, η_t can be tuned based on pilot runs using a reasonably small fraction of the training data set.

Note 13. (Intuition on Condition 11). Assume that v_t is bounded. Condition 11(3) aims at reducing the effect of the randomness in v_t (introduced noise ξ_t) because it implies $\eta_t \searrow 0$ as $t \to \infty$; if this was not the case then

$$w^{(t+1)} - w^{(t)} = -\eta_t v_t \to 0$$

may not be satisfied and the chain $\left\{w^{(t)}\right\}$ may not converge. Condition 11(2) prevents η_t from reducing too fast and allows the generated chain $\left\{w^{(t)}\right\}$ to be able to converge. E.g., after t iterations

$$\begin{aligned}\left\|w^{(t)} - w^*\right\| &= \left\|w^{(t)} \pm w^{(0)} - w^*\right\| \geq \left\|w^{(0)} - w^*\right\| - \left\|w^{(t)} - w^{(0)}\right\| \\ &\geq \left\|w^{(0)} - w^*\right\| - \sum_{t=0}^{\infty} \left\|w^{(t+1)} - w^{(t)}\right\| = \left\|w^{(0)} - w^*\right\| - \sum_{t=0}^{T-1} \|\eta_t v_t\|\end{aligned}$$

However if it was $\sum_{t=1}^{\infty} \eta_t < \infty$ it would be $\sum_{t=0}^{\infty} \|\eta_t v_t\| < \infty$ and hence $w^{(t)}$ would never converge to w^* if the seed $w^{(0)}$ is far enough from w^*.

3. Analysis of SGD (Algorithm 6)

Note 14. Recall that the stochasticity of SGD comes from the stochastic sub-gradients $\{v_t\}$ in (2.2); hence the expectations below are under these random vectors' distributions.

Proposition 15. *Let $f(\cdot)$ be a convex function. If we run SGD algorithm of f with learning rate $\eta_t > 0$ for T steps, the output $w_{\text{GD}}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$* satisfies

$$\mathrm{E}\left(f\left(w_{\text{SGD}}^{(T)}\right)\right) - f(w^*) \leq \frac{\|w^*\|^2}{2\eta T} + \frac{\eta}{2}\frac{1}{T}\sum_{t=1}^{T} \mathrm{E}\|v_t\|^2 \tag{3.1}$$

Proof. Let $v_{1:t} = (v_1, ..., v_t)$. By Jensens' inequality (or see 4.3 in Handout 4)

$$\mathrm{E}\left(f\left(w_{\text{SGD}}^{(T)}\right) - f(w^*)\right) \leq \mathrm{E}\left(\frac{1}{T}\sum_{t=1}^{T}\left(f\left(w^{(t)}\right) - f(w^*)\right)\right) = \frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left(f\left(w^{(t)}\right) - f(w^*)\right) \tag{3.2}$$

I will try to use Lemma 22 from Handout 4, hence I need to show

$$\mathrm{E}\left(f\left(w^{(t)}\right) - f(w^*)\right) \leq \mathrm{E}\left(\langle w^{(t)} - w^*, v_t\rangle\right) \tag{3.3}$$

where the expectation is under $v_{1:T}$. It is

$$\begin{aligned}\mathrm{E}_{v_{1:T}}\left(\langle w^{(t)}-w^{*},v_{t}\rangle\right) =&\mathrm{E}_{v_{1:t}}\left(\langle w^{(t)}-w^{*},v_{t}\rangle\right)\\ =&\mathrm{E}_{v_{1:t-1}}\left(\mathrm{E}_{v_{1:t}}\left(\langle w^{(t)}-w^{*},v_{t}\rangle|v_{1:t-1}\right)\right) \quad (\text{ law of total expectation })\end{aligned}$$

But $w^{(t)}$ is fully determined by $v_{1:t-1}$, (see (2.2)) so

$$\mathrm{E}_{v_{1:t-1}}\left(\mathrm{E}_{v_{1:t}}\left(\langle w^{(t)}-w^{*},v_{t}\rangle|v_{1:t-1}\right)\right) =\mathrm{E}_{v_{1:t-1}}\left(\langle w^{(t)}-w^{*},\mathrm{E}_{v_{1:t}}\left(v_{t}|v_{1:t-1}\right)\rangle\right)$$

As $w^{(t)}$ is fully determined by $v_{1:t-1}$ then $\mathrm{E}_{v_{1:t}}\left(v_{t}|v_{1:t-1}\right)=\mathrm{E}_{v_{1:t}}\left(v_{t}|w^{(t)}\right)\in\partial f\left(w^{(t)}\right)$, hence $\mathrm{E}_{v_{1:t}}\left(v_{t}|v_{1:t-1}\right)$ is a sub-gradient. By sub-gradient definition

$$\begin{aligned}\mathrm{E}_{v_{1:t-1}}\left(\langle w^{(t)}-w^{*},\mathrm{E}_{v_{1:t}}\left(v_{t}|v_{1:t-1}\right)\rangle\right) \geq&\mathrm{E}_{v_{1:t-1}}\left(f\left(w^{(t)}\right)-f\left(w^{*}\right)\right)\\ =&\mathrm{E}_{v_{1:T}}\left(f\left(w^{(t)}\right)-f\left(w^{*}\right)\right)\end{aligned} \tag{3.4}$$

Hence combining (3.4), (3.3), and (3.2)

$$\mathrm{E}\left(f\left(w_{\mathrm{SGD}}^{(T)}\right)-f\left(w^{*}\right)\right)\leq\frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left(\langle w^{(t)}-w^{*},v_{t}\rangle\right)$$

Then Lemma 22 in Handout 4 "Gradient descent" implies

$$\mathrm{E}\left(f\left(w_{\mathrm{SGD}}^{(T)}\right)-f\left(w^{*}\right)\right)\leq\mathrm{E}\left(\frac{1}{T}\frac{\left\|w^{*}\right\|^{2}}{2\eta}+\frac{\eta}{2}\frac{1}{T}\sum_{t=1}^{T}\left\|v_{t}\right\|^{2}\right)=\frac{\mathrm{E}\left\|w^{*}\right\|^{2}}{2\eta T}+\frac{\eta}{2}\frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left\|v_{t}\right\|^{2}$$

□

Note 16. The upper bound in (3.1) depends on the variation of v_t as

$$\mathrm{E}\left\|v_{t}\right\|^{2}=\sum_{j=1}^{d}\mathrm{Var}\left(v_{t,j}\right)+\sum_{j=1}^{d}\left(\mathrm{E}\left(v_{t,j}\right)\right)^{2} \tag{3.5}$$

where d is the dimension of $v_t=\left(v_{t,1},...,v_{t,d}\right)^{\top}$. The second term on the right hand side of (3.5) is constant as $v_{t,j}$ is the unbiased estimator of the sub-gradient by construction.

Proposition 17. *(Cont. Proposition 15) Let $f(\cdot)$ be a convex function, and let $\mathcal{H}=\{w\in\mathbb{R}:\|w\|\leq B\}$. Let $E\|v_t\|^2\leq\rho^2$. Assume we run SGD algorithm of $f(\cdot)$ with learning rate $\eta_t=\sqrt{\frac{B^2}{\rho^2T}}$ for T steps, and output $w_{\mathrm{SGD}}^{(T)}=\frac{1}{T}\sum_{t=1}^{T}w^{(t)}$. Then*

(1) *upper bound on the sub-optimality is*

$$\mathrm{E}\left(f\left(w_{\mathrm{SGD}}^{(T)}\right)\right)-f\left(w^{*}\right)\leq\frac{B\rho}{\sqrt{T}} \tag{3.6}$$

(2) *a given level of accuracy ε such that $E\left(f\left(w_{SGD}^{(T)}\right)\right)-f\left(w^{*}\right)\leq\varepsilon$ can be achieved after T iterations*

$$T\geq\frac{B^{2}\rho^{2}}{\varepsilon^{2}}.$$

Proof. It follows from Proposition 15. Condition $\mathrm{E}\left\|v_t\right\|^2 \leq \rho^2$ can be achieved, for instance by assuming Lipschitz (See Lemma 26 from "Handout 4: Gradient descent"). □

4. Stochastic gradient descent with projection

Note 18. Consider the scenario in Problem 1 where the learning problem requires to discover w^* in the restricted/bounded set $\mathcal{H}$. Assume the function to be minimized is convex in the restricted hypothesis set $\mathcal{H}$, e.g. $\mathcal{H} = \{w : \|w\| \leq B\}$, but non-convex in $\mathbb{R}^d$. Direct implementation of vanilla SGD (Algorithm 6) may produce a chain stepping out $\mathcal{H}$ and hence an output $w_{\mathrm{SGD}} \notin \mathcal{H}$. SGD can be modified to address this issue, as in Algorithm 19, by including a projection step guaranteeing $w \in \mathcal{H}$.

Algorithm 19. *Stochastic Gradient Descent with learning rate $\eta_t > 0$ and with projection in $\mathcal{H}$ for Problem 5*

For $t = 1, 2, 3, ...$ iterate:

(1) *compute*

$$w^{\left(t+\frac{1}{2}\right)} = w^{(t)} - \eta_t v_t, \tag{4.1}$$

where v_t is a random vector such that $E\left(v_t | w^{(t)}\right) \in \partial f\left(w^{(t)}\right)$

(2) *compute*

$$w^{(t+1)} = \arg\min_{w \in \mathcal{H}} \left(\left\| w - w^{\left(t+\frac{1}{2}\right)} \right\| \right) \tag{4.2}$$

(3) *terminate if a termination criterion is satisfied*

Note 20. The analysis in Section 3 holds for the SGD with projection after a slight modification in the math proofs; see Exercise 13 from the Exercise sheet.

5. Implementation of SGD in the learning Problem 1

Note 21. Proposition 22 allows a convenient implementation of SGD to the learning problem (Problem 1).

Proposition 22. *For a randomly drawn example $z \sim g(\cdot)$, the sub-gradient v of $\ell(w, z)$ at point w is an unbiased estimator of the sub-gradient of the risk $R_g(w)$ at point w.*

Proof. Let v be a sub-gradient of $\ell(w, z)$ at point w, then

$$\ell(u, z) - \ell(w, z) \geq \langle u - w, v \rangle \tag{5.1}$$

It is

$$\begin{aligned} R_g(u) - R_g(w) &= \mathrm{E}_{z \sim g}\left(\ell(u, z) - \ell(w, z) | w\right) \geq \mathrm{E}_{z \sim g}\left(\langle u - w, v \rangle | w\right) \\ &= \langle u - w, \mathrm{E}_{z \sim g}(v | w) \rangle \end{aligned}$$

Hence, by definition, v is such that $\mathrm{E}_{z \sim g}(v | w)$ is a sub-gradient of $R_g(w)$. □

Example 23. Consider that loss ℓ is differentiable wrt w. If $v = \nabla_w \ell\,(w, z)$, then

Note 24. Assume there is available a finite dataset $\mathcal{S}_n = \{z_i; i = 1, ..., n\}$ of size n which consists of independent realizations z_i from the data generating distribution g; $z_i \overset{\text{ind}}{\sim} g$. Batch SGD (Algorithm 25) is an implementation of the SGD (Algorithm 6) in the learning Problem 1.

Algorithm 25. *Batch Stochastic Gradient Descent with learning rate $\eta_t > 0$, and batch size m, for Problem 1.*

For $t = 1, 2, 3, ...$ iterate:

(1) *get a random sub-sample $\left\{\tilde{z}_j^{(t)}; j = 1, ..., m\right\}$ of size m with or without replacement from the complete data-set $\mathcal{S}_n$.*

(2) *compute*

$$w^{(t+1)} = w^{(t)} - \eta_t v_t, \tag{5.2}$$

where $v_t = \frac{1}{m}\sum_{j=1}^{m} \tilde{v}_{t,j}$ and $\tilde{v}_{t,j} \in \partial_w \ell\left(w^{(t)}, \tilde{z}_j^{(t)}\right)$.

(3) *terminate if a termination criterion is satisfied*

Note 26. Step 1 can be described equivalently as

(1) Randomly generate a set $\mathcal{J}^{(t)} \subseteq \{1, ..., n\}^m$ of m indices from 1 to n with or without replacement, and set a $\tilde{\mathcal{S}}_m = \left\{z_i; i \in \mathcal{J}^{(t)}\right\}$.

Hence $v_t = \frac{1}{m}\sum_{j \in \mathcal{J}^{(t)}} v_{t,j}$ where $\tilde{v}_{t,j} \in \partial_w \ell\left(w^{(t)}, z_j\right)$.

Note 27. If it is possible to sample anytime fresh examples z_i directly from the data generation model g instead of just having access to only a given finite dataset of examples $\mathcal{S}_n$, then step 1 in Algorithm 25 can become

(1) sample $\tilde{z}_j^{(t)} \sim g(\cdot)$ for $j = 1, ..., m$.

Definition 28. Online Stochastic gradient descent is the special case of Algorithm 25 using sub-samples of size one ($m = 1$), namely, only one example is randomly chosen in Step 1 of Algorithm 25.

Note 29. In theory, using larger batch size m has the benefit that reduces the variance of v_t at iteration t due to averaging effect, stabilizes the SGD algorithm, and reduces the error bound (3.1); see Remark 16.

Note 30. In practice, for a given fixed computational time, using smaller batch size m has the benefit that the algorithm iterates faster as each iteration processes less number of examples. E.g., consider the extreme cases GD vs online SGD utilized in a scenario of big-data (large training data set): if the dataset consists of several replications of the same values, GD (using all the data) has to process the same information multiple times, while the online SGD (using only one example at a time) would avoid this issue.

Note 31. In practice, for a given fixed computational time, it is possible for a batch SGD with smaller batch size m to present better generalization properties (wrt the theoretical assumptions) than those with larger m (GD is included). It is observed for the former to be often less prone to getting stuck in shallow local minima because of the additional amount of "noise" E.g., consider the extreme cases GD vs online SGD in a scenario with non-convex risk function (e.g. our theoretical assumptions as violated): if the Risk function presents local minima, considering less examples randomly chosen each time may cause fluctuations in the gradient that allow the chain to accidentally jump/escape to an area with a lower minimum.

Proposition 32. *(Implementing Proposition 17) Consider a convex-Lipschitz-bounded problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with parameters ρ and B. Then for every $\epsilon > 0$, if we run SGD Algorithm 6 to minimize Problem 1 for $T \geq \frac{B^2}{\epsilon^2}\rho^2$ iterations then it produces output $w_{SGD}^{(T)}$ satisfying*

$$E\left(R_g\left(w_{SGD}^{(T)}\right)\right) \leq \min_{\forall w \in \mathcal{H}} \left(R_g\left(w\right)\right) + \epsilon$$

and hence we can achieve PAC guarantees.

Proof. It is just re-writing the formula in part 2 of Proposition 17. Notice that here, condition $\mathrm{E}\left\|v_t\right\|^2 \leq \rho^2$ is satisfied by the self-bounding property from Lipschitzness of the loss (see Lemma 26 from "Handout 4: Gradient descent"); i.e. if ℓ is ρ-Lipschitz, then $\|v_t\| \leq \rho$ where $v_t \in \partial_w \ell\left(w^{(t)}, \tilde{z}_j^{(t)}\right)$. □

Example 33. We continue Example 33 in Section 4 of Handout 4. Recall, we considered a hypothesis space $\mathcal{H}$ of linear functions $h : \mathbb{R}^2 \to \mathbb{R}$ with $h\left(w\right) = w_1 + w_2 x$, $w = \left(w_1, w_2\right)^\top$, and $\ell\left(w, z = \left(x, y\right)^\top\right) = \left(y_i - w_1 - w_2 x\right)^2$. Here we consider a big dataset $\mathcal{S}_n = \{z_1, ..., z_n\}$ with $n = 10^6$ examples.

The batch SGD algorithm (Algorithm 25) with learning rate η_t and batch size $m = 10$ is

- For $t = 1, 2, 3, ...$ iterate:
  (1) Randomly generate a set $\mathcal{J}^{(t)}$ by drawing $m = 10$ numbers from $\left\{1, ..., n = 10^6\right\}$, and set $\tilde{\mathcal{S}}_m = \left\{z_i; i \in \mathcal{J}^{(t)}\right\}$
  (2) compute
  $$w^{(t+1)} = w^{(t)} - \eta_t v_t,$$
  where
  $$v_t = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} x_j - 2\frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} y_j \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} x_j^2 - 2 \sum_{j \in \mathcal{J}^{(t)}} y_j x_j \end{pmatrix}$$
  ,
  (3) if $t \geq T = 1000$ STOP

because

$$v_t = \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} \nabla \ell\left(w^{(t)}, \tilde{z}^{(t)}\right) = ... = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} x_j - 2\frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} y_j \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} x_j^2 - 2 \sum_{j \in \mathcal{J}^{(t)}} y_j x_j \end{pmatrix}$$

In Figures A.1a & A.1d, we observe that increasing the batch size has improved the convergence however this is not a panacea. Also it had reduced the oscillations of chain $\left\{w^{(t)}\right\}$.

Example 34. Consider a (rather naive) loss function $\ell(w, z=(x,y)) = -\cos(0.5(y - w_1 - w_2 x))$, a hypothesis class $\mathcal{H} = \left\{w \in \mathbb{R}^2 : \|w\| \leq 1.5\right\}$, and assume that inputs x in dataset $\mathcal{D}$ are such that $x \in [-1, 1]$. Note that $-\cos(\cdot)$ is convex in $[-1.5, 1.5]$ and non-convex in $\mathbb{R}$. Consider learning rate $\eta_t = 50/t$ reducing to zero, and seed $w^{(0)} = (1.5, 1.5)$. An unconstrained SGD may produce a minimizer/solution outside $\mathcal{H}$ because η_t is too large at the first few iterations. We can design the online SGD (batch size $m = 1$) with projection to $\mathcal{H}$ as

- For $t = 1, 2, 3, ...$ iterate:
  (1) randomly generate a set $\mathcal{J}^{(t)} = \{j^*\}$ by drawing one number from $\left\{1, ..., n = 10^6\right\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$
  (2) compute

$$w^{(t+1/2)} = w^{(t)} - \frac{25}{t}\begin{pmatrix} \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) \\ \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) x_{j^*} \end{pmatrix}$$

$$w^{(t+1/2)} = \underset{\|w\| \leq 1.5}{\arg\min}\left(\left\|w - w^{(t+1/2)}\right\|\right)$$

  (3) if $t \geq T = 1000$ STOP

  because

$$v_t = \nabla \ell\left(w^{(t)}, z_{j^*}\right) = \begin{pmatrix} \frac{\partial}{\partial w_1^{(t)}} \ell\left(w^{(t)}, z_{j^*}\right) \\ \frac{\partial}{\partial w_2^{(t)}} \ell\left(w^{(t)}, z_{j^*}\right) \end{pmatrix} = \begin{pmatrix} 0.5 \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) \\ 0.5 \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) x_{j^*} \end{pmatrix}$$

  In Figures A.1b & A.1e, we observe that the SGD got trapped outside $\mathcal{H}$ due to the unreasonably large learning rate at the beginning of the iterations, while the SGD with projection step managed to stay in $\mathcal{H}$ and converge.

6. Stochastic Variance Reduced Gradient (SVRG) Ref [3]

Remark 35. Recall the upper bound of the error (3.1) in SGD depends on the variance of the stochastic gradient, as shown in (3.5) of Remark 16. Hence the algorithm may be improved by reducing the variance of each element of v_t.

Remark 36. Control variate is a general way to perform variance reduction. Let random variables $v \in \mathbb{R}$, and $y \in \mathbb{R}$. Let $z = v + c(y - \mathrm{E}(y))$ for some constant $c \in \mathbb{R}$. It is $\mathrm{E}_c(z) = \mathrm{E}(v)$ and

$$\mathrm{Var}_c(z) = \mathrm{Var}(v) + c^2 \mathrm{Var}(y) + 2c\mathrm{Cov}(x, y)$$

which is minimized for

$$c^* = -\frac{\mathrm{Cov}(v, y)}{\mathrm{Var}(y)}$$

hence

$$\mathrm{Var}_{c^*}(z) = \mathrm{Var}(v) - \frac{(\mathrm{Cov}(v, y))^2}{\mathrm{Var}(y)}$$

Note 37. For simplicity, SVRG is presented in the case where $c = 1$ and the loss function $\ell(w, z)$ is differentiable wrt w at every z hence its gradient exists. However it is applicable to the more general cases introduced.

Remark 38. Every κ iterations, SVRG keeps a snapshot $\tilde{w}$, and computes the gradient using all data i.e. $\frac{1}{n}\sum_{i=1}^{n}\ell\left(\tilde{w},z_i\right)$. At each iteration t, the update is

$$w^{(t+1)}=w^{(t)}-\eta_t\underbrace{\left[\underbrace{\nabla\ell\left(w^{(t)},\tilde{z}^{(t)}\right)}_{=v}-\underbrace{1}_{=c}\underbrace{\left(\nabla\ell\left(\tilde{w},\tilde{z}^{(t)}\right)-\frac{1}{n}\sum_{i=1}^{n}\nabla\ell\left(\tilde{w},z_i\right)\right)}_{y}\right]}_{z}$$

given that a random $\tilde{z}^{(t)}$ has been collected from the sample. The symbols below the brackets are given with reference to Remark 36.

Algorithm 39. *Stochastic Variance Reduced Gradient with learning rate $\eta_t>0$ for Problem 1.*

For $t=1,2,3,...$ iterate:

(1) *randomly get an example $\tilde{z}^{(t)}$ from S_n.*

(2) *compute*

$$w^{(t+1)}=w^{(t)}-\eta_t\left[\nabla\ell\left(w^{(t)},\tilde{z}^{(t)}\right)-\nabla\ell\left(\tilde{w},\tilde{z}^{(t)}\right)+\frac{1}{n}\sum_{i=1}^{n}\nabla\ell\left(\tilde{w},z_i\right)\right] \tag{6.1}$$

(3) *if modulo $(t,\kappa)=0$, set $\tilde{w}=w^{(t)}$*

(4) *if a termination criterion is satisfied STOP*

Remark 40. Iterations of SVRG are computationally faster than those of full GD, but SVRG can still match the theoretical convergence rate of GD.

Remark 41. How often we get snapshots, aka κ, in Algorithm 39 is specified by the researcher. The smaller the κ, the more frequent snapshots, and the more correlated the baseline y will be with the objective x and hence the bigger the performance improvement; however the iterations will be slower.

Example 42. The SVRG with learning rate $\eta_t>0$ and batch size $m=1$ is

- For $t=1,2,3,...$ iterate:
  (1) randomly generate a set $\mathcal{J}^{(t)}=\{j^*\}$ by drawing one number j^* from $\left\{1,...,n=10^6\right\}$, and set $\tilde{\mathcal{S}}_1=\{z_{j^*}\}$
  (2) compute

$$w^{(t+1)}=\begin{bmatrix}w_1^{(t)}\\w_2^{(t)}\end{bmatrix}-\eta_t\left(\begin{bmatrix}2\left(w_1^{(t)}-\tilde{w}_1^{(t)}\right)+2\left(w_2^{(t)}-\tilde{w}_2^{(t)}\right)x_{j^*}\\2\left(w_2^{(t)}-\tilde{w}_2^{(t)}\right)x_{j^*}+2\left(w_2^{(t)}-\tilde{w}_2^{(t)}\right)\left(x_{j^*}\right)^2\end{bmatrix}+\nabla\hat{R}_{\mathcal{D}}\left(\tilde{w}\right)\right) \tag{6.2}$$

  (3) if modulo $(t,\kappa)=0$,
    (a) set $\tilde{w}=w^{(t)}$

(b) compute

$$\frac{1}{n}\sum_{i=1}^{n}\ell\left(\tilde{w},z_i\right)=\begin{pmatrix}2\tilde{w}_1^{(t)}+2\tilde{w}_2^{(t)}\bar{x}-2\bar{y}\\ 2\tilde{w}_1^{(t)}\bar{x}+2\tilde{w}_2^{(t)}\overline{x^2}-2y^\top x\end{pmatrix}=\nabla\hat{R}_{\mathcal{D}}\left(\tilde{w}\right) \tag{6.3}$$

(4) if a termination criterion is satisfied STOP

Because (6.3) is actually the gradient of the Risk function at $\tilde{w}$ and

$$\nabla\ell\left(w^{(t)},z_{j^*}\right)-\nabla\ell\left(\tilde{w},z_{j^*}\right)=\begin{pmatrix}2\left(w_1^{(t)}-\tilde{w}_1^{(t)}\right)+2\left(w_2^{(t)}-\tilde{w}_2^{(t)}\right)x_{j^*}\\ 2\left(w_2^{(t)}-\tilde{w}_2^{(t)}\right)x_{j^*}+2\left(w_2^{(t)}\left(x_{j^*}\right)^2-\tilde{w}_2^{(t)}\left(x_{j^*}\right)^2\right)\end{pmatrix}$$

In Figure A.1c we observe the frequency of the snapshots has improved the convergence; however this is not a panacea as seen in Figure A.1f.

7. Preconditioned SGD; The AdaGrad algorithm

Ref [4]

Remark 43. All SGD (introduced earlier) are first order methods in the sense they consider only the gradient. Their advantage is each iteration is fast. The disadvantage is that they ignore the curvature of the space and hence can be slower to converge in cases the curvature changes eg. among dimensions of w.

Remark 44. To address this, SGD (Algorithm 6) can be modified in the update step as in Algorithm 45 by using a preconditioner P_t that accounts for the curvature (or geometry in general of f).

Algorithm 45. *Preconditioned Stochastic Gradient Descent with learning rate $\eta_t > 0$, and preconditioner P_t for the solution of the minimization problem (2.1)*

For $t=1,2,3,...$ iterate:

(1) *compute*

$$w^{(t+1)}=w^{(t)}-\eta_t P_t v_t, \tag{7.1}$$

where v_t is a random vector such that $E\left(v_t|w^{(t)}\right)\in\partial f\left(w^{(t)}\right)$, and P_t is a preconditioner

(2) *terminate if a termination criterion is satisfied, e.g.*

If $t\geq T$ then STOP

Remark 46. A natural choice of P_t can be $P_t:=\left[H_t+\epsilon I_d\right]^{-1}$, where H_t is the Hessian matrix $[H_t]_{i,j}=\frac{\partial^2}{\partial w_i\partial w_j}f\left(w\right)\Big|_{w=w^{(t)}}$ (ie. the gradient of the gradient's elements), and ϵ is a tiny $\epsilon>0$ to mitigate machine error when Hessian elements are close to zero.

Remark 47. If the preconditioner P_t is set to be the inverse of the full Hessian, it may be too expensive to perform matrix operations in (7.1) with the full Hessian, and such operations can be too unstable/inaccurate due to the random error induced by the stochasticity of the gradient.

7.1. Adaptive Stochastic Gradient Decent (AdaGrad).

Remark 48. AdaGrad aims to dynamically incorporate knowledge of the geometry of function $f(\cdot)$ (to be minimized) in earlier iterations to perform more informative gradient-based learning.

Remark 49. AdaGrad aims to perform larger updates (i.e. high learning rates) for those dimensions of w that are related to infrequent features (largest partial derivative) and smaller updates (i.e. low learning rates) for frequent ones (smaller partial derivative).

Remark 50. Hence, this strategy often improves convergence performance over standard stochastic gradient descent in settings where the data are sparse and sparse features w's are more informative.

Algorithm 51. *Adaptive Stochastic Gradient Decent (AdaGrad) can be presented in terms of preconditioned SGD (Algorithm 45) with preconditioner* $P_t = [diag(G_t) + \epsilon I_d]^{-1/2}$ *in (7.1)*

$$w^{(t+1)} = w^{(t)} - \eta_t \left[diag(G_t) + \epsilon I_d\right]^{-1/2} v_t, \tag{7.2}$$

where notation $diag(A)$ *denotes a* $d \times d$ *matrix whose diagonal is the* d *dimensional diagonal vector* $(A_{1,1}, A_{2,2}..., A_{d,d})$ *of* $d \times d$ *matrix* A *and whose off-diagonal elements are zero,* $G_t = \sum_{\tau=1}^{t} v_\tau^\top v_\tau$ *is the sum of the outer products of the gradients* $\{v_\tau; \tau \leq t\}$ *up to the state* t*, and* $\epsilon > 0$ *is a tiny value (eg,* 10^{-6}*) set for computational stability is case the gradient becomes too close to zero.*

Remark 52. AdaGrad algorithm individually adapts the learning rate of each dimension of w_t by scaling them inversely proportional to the square root of the sum of all the past squared values of the gradient $\{v_\tau; \tau \leq t\}$. This is because

$$[G_t]_{j,j} = \sum_{\tau=1}^{t} (v_{\tau,j})^2 \tag{7.3}$$

where j denotes the j-th dimension of w. Hence (7.2) and (7.2)imply that the j-th dimension of w is updated as

$$w_j^{(t+1)} = w_j^{(t)} - \eta_t \frac{1}{\sqrt{[G_t]_{j,j} + \epsilon}} v_{t,j}.$$

Remark 53. The accumulation of positive terms in (7.3) makes the sum keep growing during training and causes the learning rate to shrink and becoming infinitesimally small. This offers an automatic way to choose a decreasing learning rate simplifying setting the learning rate; however it may result in a premature and excessive decrease in the effective learning rate. This can be mitigated by still considering in (7.2) a (user specified rate) $\eta_t \geq 0$ and tuning it properly via pilot runs.

Example 54. The AdaGrad with $\eta_t = 1$, $\epsilon = 10^{-6}$, and batch size $m = 1$ is

- Set $G_0 = (0, 0)^\top$.
- For $t = 1, 2, 3, ...$ iterate:
  (1) randomly generate a set $\mathcal{J}^{(t)} = \{j^*\}$ by drawing one number from $\{1, ..., n = 10^6\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$

(2) compute

$$v_t = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} x_{j^*} - 2y_{j^*} \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} x_{j^*}^2 - 2y_{j^*} x_{j^*} \end{pmatrix}$$

$$G_t = G_{t-1} + \begin{pmatrix} v_{t,1}^2 \\ v_{t,2}^2 \end{pmatrix}$$

$$w^{(t+1)} = \begin{pmatrix} w_1^{(t)} - \frac{\eta_t}{\sqrt{G_{t,1}+\epsilon}} v_{t,1} \\ w_2^{(t)} - \frac{\eta_t}{\sqrt{G_{t,2}+\epsilon}} v_{t,2} \end{pmatrix},$$

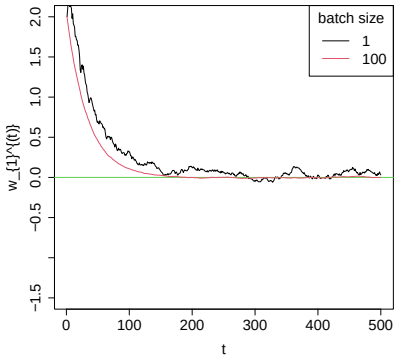
(3) if $t \geq T = 1000$ STOP

because

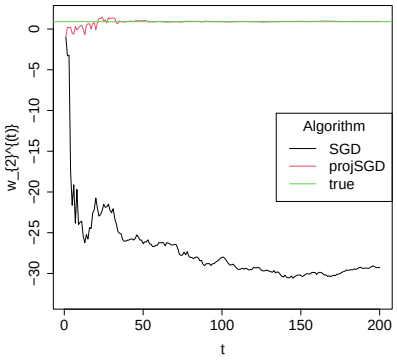
$$v_t = \nabla \ell \left(w^{(t)}, z_{j^*} \right) = \begin{pmatrix} \frac{\partial}{\partial w_1^{(t)}} \ell \left(w^{(t)}, z_{j^*} \right) \\ \frac{\partial}{\partial w_2^{(t)}} \ell \left(w^{(t)}, z_{j^*} \right) \end{pmatrix} = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} x_{j^*} - 2y_{j^*} \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} x_{j^*}^2 - 2y_{j^*} x_{j^*} \end{pmatrix}$$

In Figures A.1g & A.1h, we see that AdaGrad with $\eta = 1$ works (I did not try to tune it), however to make vanilla SGD to work I have to tune $\eta = 0.03$ otherwise for $\eta = 1.0$ it did not work.

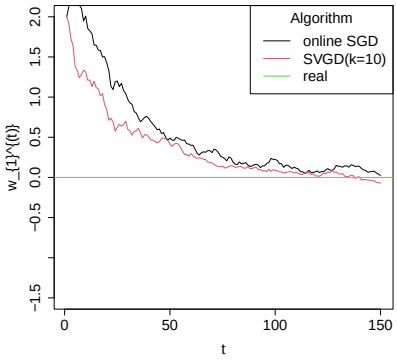
Appendix A. Plots



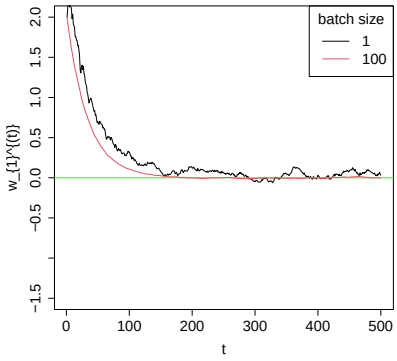
(A) batch SGD Example 33 w_1



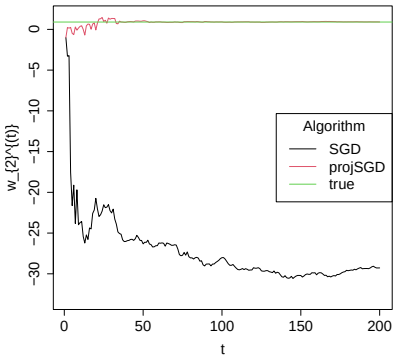
(B) online SGD with projection Example 34w_1



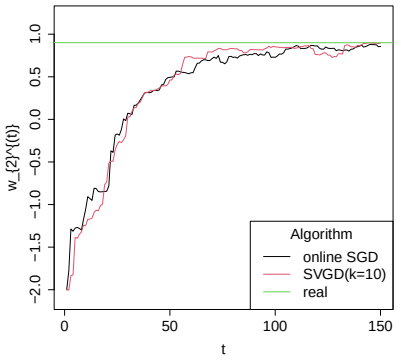
(C) SVRG Example 42 w_1



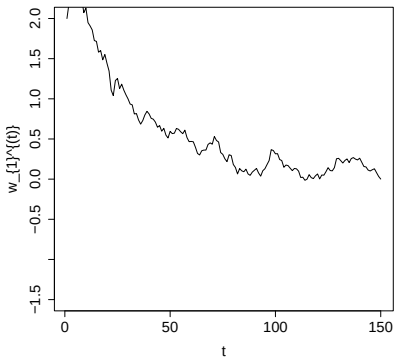
(D) batch SGD Example 33 w_2



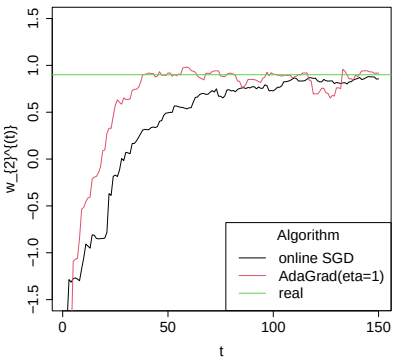
(E) online SGD with projection Example 34 w_2



(F) SVRG Example 42 w_2



(G) online AdaGrad Example 54w_1



(H) online AdaGrad Example 54w_2

FIGURE A.1. Simulations of the Examples
Created on 2024/02/06 at 14:49:44