Machine Learning and Neural Networks (MATH3431) Epiphany term, 2025

Lecture notes 6: Variations of stochastic gradient descent

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

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

Reading list & references:

(1) Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." Advances in neural information processing systems 26 (2013).
(2) 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).

Problem 1. Consider the following minimization problem

$$w^* = \arg\min_w \left(f\left(w\right)\right) \tag{0.1}$$

where here $f : \mathbb{R}^d \to \mathbb{R}$, and $w \in \mathcal{W} \subseteq \mathbb{R}^d$; $f\left(\cdot\right)$ is the unknown function to be minimized.

Note 2. For instance, $f\left(\cdot\right)$ can be the risk function $R_g\left(w\right) = \mathrm{E}_{z\sim g}\left(\ell\left(h_w, z\right)\right)$ associated to a learning problem $\left(\mathcal{H}, \mathcal{Z}, \ell\right)$ with hypothesis $h_w \in \mathcal{H}$ parameterized by $w \in \mathcal{W}$ and training dataset $\mathcal{S}$ drawn from an unknown data generating process $g\left(\cdot\right)$.

1. Stochastic gradient descent with projection

Note 3. Consider the scenario in Problem 1 where the optimization problem requires to discover w^* in a restricted/bounded set. Assume the function to be minimized is convex in the restricted set $\mathcal{W}$, e.g. $\mathcal{W} = \{w : \|w\| \leq B\}$, but non-convex in $\mathbb{R}^d$. Direct implementation of SGD (Algorithm 6) may produce a chain stepping out $\mathcal{W}$ and hence an output $w_{\mathrm{SGD}} \notin \mathcal{W}$. SGD can be modified to address this issue by including a projection step guaranteeing $w \in \mathcal{W}$ as in Algorithm 4.

Algorithm 4. *Stochastic Gradient Descent with learning rate $\eta_t > 0$ and with projection in $\mathcal{W}$ for Problem 5*

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

(1) *compute*

$$w^{\left(t+\frac{1}{2}\right)} = w^{(t)} - \eta_t v_t, \tag{1.1}$$

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*

$$w^{(t+1)} = \arg\min_{w \in \mathcal{W}} \left(\left\| w - w^{\left(t+\frac{1}{2}\right)} \right\| \right) \tag{1.2}$$

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

Note 5. 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.

Example 6. [1] Consider a (rather naive) loss function $\ell\left(w, z = (x, y)\right) = -\cos\left(0.5\left(y - w_1 - w_2 x\right)\right)$, a hypothesis class $\mathcal{H} = \left\{h_w\left(x\right) = w^\top x : w \in \mathcal{W}\right\}$ and $\mathcal{W} = \left\{w \in \mathbb{R}^2 : \|w\| \leq 1.5\right\}$, and assume that inputs x in dataset S are such that $x \in [-1, 1]$. Note that $-\cos\left(\cdot\right)$ 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 ($m = 1$) 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

$$\begin{aligned} w^{(t+1/2)} =& w^{(t)} - \frac{25}{t} \begin{pmatrix} \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) \\ \sin\left(0.5\left(y_{j^*} - w_1^{(t)} - w_2^{(t)} x_{j^*}\right)\right) x_{j^*} \end{pmatrix} \\ w^{(t+1/2)} =& \arg\min_{\|w\| \leq 1.5} \left(\left\| w - w^{(t+1/2)} \right\| \right) \end{aligned}$$

  (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.

[1] `https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2025/tree/main/Lecture_notes/code/example_projSGD.R`

Note 7. 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.

Note 8. To address this, SGD (Algorithm 6) can be modified in the update step as in Algorithm 9 by using a preconditioner P_t that accounts for the curvature (or geometry in general of f).

Algorithm 9. *Preconditioned Stochastic Gradient Descent with learning rate $\eta_t > 0$, and preconditioner P_t for the solution of the minimization problem (3.1)*

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

(1) *compute*

$$w^{(t+1)} = w^{(t)} - \eta_t P_t v_t, \tag{2.1}$$

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.*

$$\textit{If } t \geq T \textit{ then STOP}$$

Note 10. A natural choice of P_t can be $P_t := [H_t + \epsilon I_d]^{-1}$, where H_t is the Hessian matrix $[H_t]_{i,j} = \frac{\partial^2}{\partial w_i \partial w_j} f(w)\Big|_{w=w^{(t)}}$ (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.

Note 11. 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 (2.1) with the full Hessian, and such operations can be too unstable/inaccurate due to the random error induced by the stochasticity of the gradient.

2.1. **Adaptive Stochastic Gradient Decent (AdaGrad).**

Note 12. 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.

Note 13. 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).

Note 14. 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 15. *Adaptive Stochastic Gradient Decent (AdaGrad) can be presented in terms of preconditioned SGD (Algorithm 9) with preconditioner $P_t = [diag(G_t) + \epsilon I_d]^{-1/2}$ in (2.1)*

$$w^{(t+1)} = w^{(t)} - \eta_t \left[diag(G_t) + \epsilon I_d\right]^{-1/2} v_t, \tag{2.2}$$

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, $G_t = \sum_{\tau=1}^{t} v_\tau v_\tau^\top$ 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.

Note 16. 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

$$[G_t]_{j,j} = \sum_{\tau=1}^{t} (v_{\tau,j})^2 \tag{2.3}$$

where j denotes the j-th dimension of w. Hence (2.2) and (2.2)imply that the j-th dimension of w is updated as

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

Note 17. The accumulation of positive terms in (2.3) 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 (2.2) a (user specified rate) $\eta_t \geq 0$ and tuning it properly via pilot runs.

Example 18. [2] 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 j^* from $\{1, ..., n = 10^6\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$
  (2) compute

$$\begin{aligned} v_t =& \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} x_{j^*} - 2y_{j^*} \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} x_{j^*}^2 - 2y_{j^*} x_{j^*} \end{pmatrix} \\ G_t =& G_{t-1} + \begin{pmatrix} v_{t,1}^2 \\ v_{t,2}^2 \end{pmatrix} \\ w^{(t+1)} =& \begin{pmatrix} w_1^{(t)} - \frac{\eta_t}{\sqrt{G_{t,1}+\epsilon}} v_{t,1} \\ w_2^{(t)} - \frac{\eta_t}{\sqrt{G_{t,2}+\epsilon}} v_{t,2} \end{pmatrix}, \end{aligned}$$

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

because

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

[2] https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2025/tree/main/Lecture_notes/code/example_AdaGrad.R

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.

3. Stochastic Variance Reduced Gradient (SVRG) Ref [1]

Note 19. Recall the upper bound of the error ((3.1) ; in Lect. Notes5) in SGD depends on the variance of the stochastic gradient, as shown in (6) of Note 16. Hence the algorithm may be improved by reducing the variance of each element of v_t.

Note 20. 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 21. For simplicity, SVRG is presented in the case where $c = 1$ and the loss function $\ell\,(h_w, z)$ is differentiable wrt w at every z hence its gradient exists. However it is applicable to the more general cases introduced.

Note 22. Every κ iterations, SVRG keeps a snapshot $\tilde{w}$, and computes the gradient using all data i.e. $\frac{1}{n}\sum_{i=1}^{n} \ell\,(\tilde{w}, z_i)$. At each iteration t, the update is

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

given that a random $\tilde{z}^{(t)}$ has been collected from the sample. The symbols below the brackets are given with reference to Remark 20.

Algorithm 23. *Stochastic Variance Reduced Gradient with learning rate $\eta_t > 0$ for Problem 1.*

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

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

(2) *compute*

$$w^{(t+1)} = w^{(t)} - \eta_t \left[\nabla \ell \left(w^{(t)}, \tilde{z}^{(t)} \right) - \nabla \ell \left(\tilde{w}, \tilde{z}^{(t)} \right) + \frac{1}{n} \sum_{i=1}^{n} \nabla \ell \left(\tilde{w}, z_i \right) \right] \tag{3.1}$$

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

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

Note 24. Iterations of SVRG are computationally faster than those of full GD, but SVRG can still match the theoretical convergence rate of GD.

Note 25. The frequency of the snapshots κ 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 26. [3] 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

$$w^{(t+1)} = \begin{bmatrix} w_1^{(t)} \\ w_2^{(t)} \end{bmatrix} - \eta_t \left(\begin{bmatrix} 2\left(w_1^{(t)} - \tilde{w}_1^{(t)} \right) + 2\left(w_2^{(t)} - \tilde{w}_2^{(t)} \right) x_{j^*} \\ 2\left(w_2^{(t)} - \tilde{w}_2^{(t)} \right) x_{j^*} + 2\left(w_2^{(t)} - \tilde{w}_2^{(t)} \right) (x_{j^*})^2 \end{bmatrix} + \nabla \hat{R}_{\mathcal{D}} \left(\tilde{w} \right) \right) \tag{3.2}$$

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

$$\frac{1}{n} \sum_{i=1}^{n} \ell \left(\tilde{w}, z_i \right) = \begin{pmatrix} 2\tilde{w}_1^{(t)} + 2\tilde{w}_2^{(t)} \bar{x} - 2\bar{y} \\ 2\tilde{w}_1^{(t)} \bar{x} + 2\tilde{w}_2^{(t)} \overline{x^2} - 2y^\top x \end{pmatrix} = \nabla \hat{R}_{\mathcal{D}} \left(\tilde{w} \right) \tag{3.3}$$

  (4) if a termination criterion is satisfied STOP

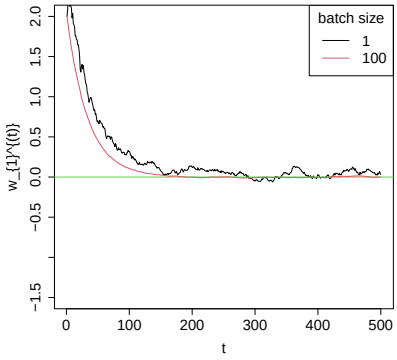
Because (3.3) is actually the gradient of the Risk function at $\tilde{w}$ and

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

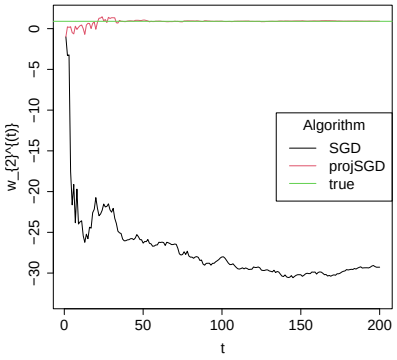
In Figure A.1c we observe the frequency of the snapshots has improved the convergence; however this is not a panacea as seen in Figure A.1f.

[3] `https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2025/tree/main/Lecture_notes/code/example_SVRG.R`

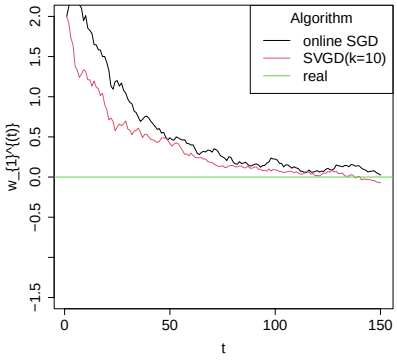
Appendix A. Plots



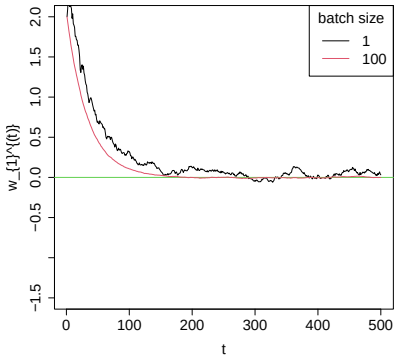
(A) batch SGD Example 30 w_1



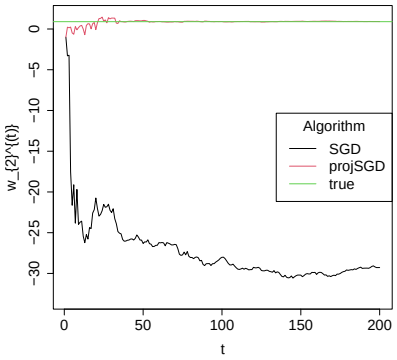
(B) online SGD with projection Example 6w_1



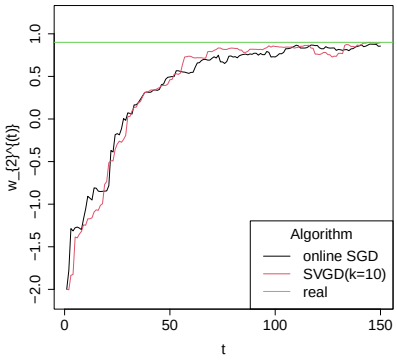
(C) SVRG Example 26 w_1



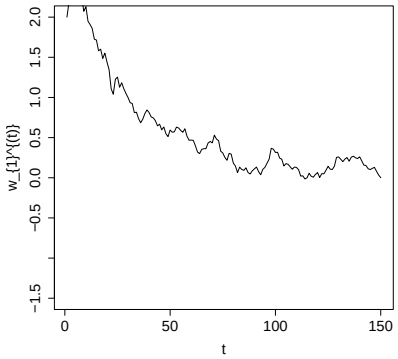
(D) batch SGD Example 30 w_2



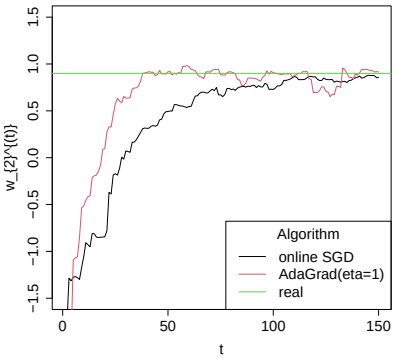
(E) online SGD with projection Example 6 w_2



(F) SVRG Example 26 w_2



(G) online AdaGrad Example 18w_1



(H) online AdaGrad Example 18w_2

FIGURE A.1. Simulations of the Examples