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 stochastic gradient descent variations for restricted sets, adaptive step functions, and reduced errors.

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).

1. Stochastic gradient descent with projection

Problem 1. Consider the minimization problem

$$w^* = \arg\min_w \left(f\left(w \right) \right) \tag{1.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$, $w \in \mathcal{W}$, and assume $\mathcal{W} \subset \mathbb{R}^d$ is a restricted domain.

Note 2. Consider Problem 1 requiring to discover w^* in a restricted/bounded set $\mathcal{W} \subset \mathbb{R}^d$, e.g. $\mathcal{W} = \left\{ w \in \mathbb{R}^d : \|w\| \leq B \right\}$ for $B > 0$. If f is convex over $\mathcal{W}$, but non-convex in $\mathbb{R}^d / \mathcal{W}$, direct implementation of (standard) SGD (Algorithm 6) may produce a chain stepping out $\mathcal{W}$ and hence an output $w_{\text{SGD}} \notin \mathcal{W}$.

Note 3. Stochastic gradient descent with projection (prjSGD) is a modified SGD suitable for applications with restricted sets and relies on the consideration of a projection step guaranteeing $w \in \mathcal{W}$.

Algorithm 4. *Stochastic Gradient Descent with learning rate $\eta_t > 0$ and with projection in $\mathcal{W}$ for Problem1*

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

(1) *compute*

$$w^{\left(t+\frac{1}{2}\right)} = w^{(t)} - \eta_t v_t, \tag{1.2}$$

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

(2) *compute*

$$w^{(t+1)} = \arg\min_{w \in \mathcal{W}} \left(\left\| w - w^{\left(t+\frac{1}{2}\right)} \right\| \right) \tag{1.3}$$

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

Note 5. The analysis of SGD with projection is given as a Exercise 13 from the Exercise sheet.

Example 6. [1] Consider a (rather naive) loss function $\ell(w, z = (x, y)) = -\cos(0.5(y - w_1 - w_2 x))$, a hypothesis class $\mathcal{H} = \left\{h_w(x) = w^\top x : w \in \mathcal{W}\right\}$ 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 $-\cos(\cdot)$ 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 ($m = 1$) 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

2. Preconditioned SGD abd the AdaGrad algorithm

Problem 7. Consider the minimization problem

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

where $f : \mathcal{W} \to \mathbb{R}$, and assume $\mathcal{W} \equiv \mathbb{R}^d$.

Note 8. All SGD (introduced earlier) are first order methods, in the sense they consider only the gradient, and hence they ignore the curvature of the space. Consequently they may present slow converge in cases the curvature changes, eg. among dimensions of w.

Note 9. Preconditioned Stochastic Gradient Descent (Algorithm 10) is a modified SGD aiming to expedite convergence of SGD by using a preconditioner P_t that accounts for the curvature (or geometry in general) of f.

Algorithm 10. *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*

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

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

Note 11. 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 $\left[H_t\right]_{i,j} = \frac{\partial^2}{\partial w_i \partial w_j} f\left(w\right)\Big|_{w=w^{(t)}}$ (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.

Note 12. Using the inverse of the full Hessian as preconditioner P_t may be too expensive to perform matrix operations in (2.2) and 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 13. AdaGrad (Algorithm 15) is a modified SGD algorithm that aims to adaptively adjust the learning rate of SGD, perform more informative gradient-based learning, and by improve convergence performance over standard SGD by dynamically incorporating knowledge of the geometry of function $f\left(\cdot\right)$ in earlier iterations.

Note 14. It is particularly suitable in cases where the data are sparse and sparse features w's are more informative. It 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).

Algorithm 15. *Adaptive Stochastic Gradient Decent (AdaGrad) is a preconditioned SGD (Algorithm 10) with preconditioner* $P_t = [diag(G_t) + \epsilon I_d]^{-1/2}$ *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. I.e.*

$$w^{(t+1)} = w^{(t)} - \eta_t \left[diag(G_t) + \epsilon I_d\right]^{-1/2} v_t. \tag{2.3}$$

Note 16. AdaGrad 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

$$[G_t]_{j,j} = \sum_{\tau=1}^{t} (v_{\tau,j})^2 \tag{2.4}$$

where j denotes the j-th dimension of w. Hence (2.3) and (2.3)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 (2.4) makes the sum keep growing during training and causes the learning rate to shrink and become 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 (2.3) 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 $j^{(t)}$ from $\left\{1, ..., n = 10^6\right\}$
  (2) compute

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

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

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

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

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

because

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

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.

3. Stochastic Variance Reduced Gradient (SVRG)

Note 19. Recall, in Note 16 of Lecture Notes 5, the upper bound of the error ((3.1); in L.N. 5) in SGD depends on the variance of the stochastic sub-gradient i.e. $\mathrm{E}\left\|v_t - \mathrm{E}\left(v_t\right)\right\|^2$.

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

$$\mathrm{E}\left\|v_t\right\|^2 = \underbrace{\mathrm{E}\left\|v_t - \mathrm{E}\left(v_t\right)\right\|^2}_{=\mathrm{tr}(\mathrm{Var}(v_t))} + \underbrace{\left\|\mathrm{E}\left(v_t\right)\right\|^2}_{\text{const.}} \tag{3.2}$$

A way to improve the SGD may be by reducing / controlling the variances at each dimension of v_t.

3.1. Batch Stochastic gradient descent.

Note 20. A way to reduce variance in (3.2) is to consider a larger batch size $m = \left|\mathcal{J}^{(t)}\right|$. Note that

$$\mathrm{E}_{z\sim g}\left\|v_t - \mathrm{E}\left\|v_t\right\|\right\|^2 = \mathrm{E}_{z\sim g}\left\|\nabla_w \frac{1}{\left|\mathcal{J}^{(t)}\right|}\sum_{j\in\mathcal{J}^{(t)}} \ell\left(w, z_j\right) - \nabla_w R_g\left(w\right)\right\|^2 = \frac{\mathrm{E}_{z_j\sim g}\left\|\nabla_w \ell\left(w, z_j\right) - \nabla_w R_g\left(w\right)\right\|^2}{\left|\mathcal{J}^{(t)}\right|}$$

where to top part of the fraction is the gradient variance for SGD with $m = 1$.

3.2. Stochastic Variance Reduced Gradient (SVRG).

Note 21. Control variate is a general way to perform variance reduction. Let $x \in \mathbb{R}^d$ be an unbiased estimator of θ. The goal is to compute $v = x + c\left(y - \mathrm{E}\left(y\right)\right)$ by finding a suitable control variable $y \in \mathbb{R}^d$ and constant $c \in \mathbb{R}$ such that $\mathrm{tr}\left(\mathrm{Var}_c\left(v\right)\right) < \mathrm{tr}\left(\mathrm{Var}\left(x\right)\right)$ with purpose to use it as an unbiased estimator of θ instead of x $\mathrm{E}_c\left(v\right) = \mathrm{E}\left(x\right) = \theta$.

Example 22. Let random variables $x \in \mathbb{R}^d$, and $y \in \mathbb{R}^d$. Let $v = x + c\left(y - \mathrm{E}\left(y\right)\right)$ for some constant $c \in \mathbb{R}$. It is $\mathrm{E}_c\left(v\right) = \mathrm{E}\left(x\right)$. Also

$$\mathrm{E}_c\left\|v - \mathrm{E}\left(v\right)\right\|^2 = \mathrm{E}\left\|x - \mathrm{E}\left(x\right)\right\|^2 + c^2\mathrm{E}\left\|y - \mathrm{E}\left(y\right)\right\|^2 + 2c\mathrm{E}\left\langle x, y\right\rangle$$

which is minimized for

$$0 = \frac{\mathrm{d}}{\mathrm{d}c}\mathrm{E}_c\left\|v - \mathrm{E}\left(v\right)\right\|^2\Big|_{c=c^*} \implies c^* = -\frac{\mathrm{E}\left\langle x, y\right\rangle}{\mathrm{E}\left\|y - \mathrm{E}\left(y\right)\right\|^2}$$

with

$$\mathrm{E}_{c^*}\left\|v - \mathrm{E}\left(v\right)\right\|^2 = \mathrm{E}\left\|x - \mathrm{E}\left(x\right)\right\|^2 - \frac{\mathrm{E}\left\langle x, y\right\rangle}{\mathrm{E}\left\|y - \mathrm{E}\left(y\right)\right\|^2} \tag{3.3}$$

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 $z_{j^{(t)}}$ indexed by $j^{(t)} \in \{1, ..., n\}$.*

(2) *compute*

$$w^{(t+1)} = w^{(t)} - \eta_t v_t \tag{3.4}$$

where

$$v_t = \underbrace{\nabla \ell \left(w^{(t)}, z_{j^{(t)}} \right)}_{=x} \underbrace{-}_{=c} \left(\underbrace{\nabla \ell \left(\tilde{w}, z_{j^{(t)}} \right)}_{=y} - \frac{1}{n} \sum_{i=1}^{n} \nabla \ell \left(\tilde{w}, z_i \right) \right) \tag{3.5}$$

.

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

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

Note 24. Prepend, we do not know c in (3.3). The variance of gradient v_t^{SVRG} is

$$\begin{aligned} \mathrm{E}_{z\sim g} \left\| v_t^{\text{SVRG}} - \mathrm{E} \left(v_t^{\text{SVRG}} \right) \right\|^2 &= \mathrm{E}_{z\sim g} \left\| \overbrace{\nabla_w \ell \left(w^{(t)}, z_{j^{(t)}} \right) + c \left(\nabla_w \ell \left(\tilde{w}, z_{j^{(t)}} \right) - \frac{1}{n} \sum_{i=1}^{n} \nabla \ell \left(\tilde{w}, z_i \right) \right)}^{v_t^{\text{SVRG}} =} - \nabla_w R_g \left(w^{(t)} \right) \right\|^2 \\ &\leq \mathrm{E}_{z\sim g} \left\| \nabla_w \ell \left(w^{(t)}, z_{j^{(t)}} \right) + c \nabla_w \ell \left(\tilde{w}, z_{j^{(t)}} \right) \right\|^2 \end{aligned}$$

As optimal c is often intractable, we sub-optimally set $c = -1$ and get

$$\mathrm{E}_{z\sim g} \left\| v_t^{\text{SVRG}} - \mathrm{E} \left(v_t^{\text{SVRG}} \right) \right\|^2 \leq \mathrm{E}_{z\sim g} \left\| \nabla_w \ell \left(w^{(t)}, z_{j^{(t)}} \right) - \nabla_w \ell \left(\tilde{w}, z_{j^{(t)}} \right) \right\|^2$$

If $\ell \left(\cdot, z_j \right)$ is β_j-smooth, then

$$\mathrm{E}_{z\sim g} \left\| v_t^{\text{SVRG}} - \mathrm{E} \left(v_t^{\text{SVRG}} \right) \right\|^2 \leq \max_j \left(\{ \beta_j \} \right) \left\| w^{(t)} - \tilde{w} \right\|^2$$

Hence to bound/control variance, point $\tilde{w}$ should be chosen to be as close as possible to $w^{(t)}$. Perhaps by taking a snapshot $\tilde{w} \leftarrow w^{(t)}$ every κ-th iteration.

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 in (3.5) and hence the bigger the performance improvement (See (3.3)); however the iterations will be slower.

Note 26. Iterations of SVRG are computationally faster than those of full GD but SVRG can still match the theoretical convergence rate of GD.

Example 27. [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 $\mathcal{J}^{(t)} = \left\{ j^{(t)} \right\}$ by drawing one number $j^{(t)}$ from $\left\{ 1, ..., n = 10^6 \right\}$.

(2) compute

$$w^{(t+1)} = \begin{bmatrix} w_1^{(t)} \\ w_2^{(t)} \end{bmatrix} - \eta_t \left(\begin{bmatrix} 2\left(w_1^{(t)} - \tilde{w}_1 \right) + 2\left(w_2^{(t)} - \tilde{w}_2 \right) x_{j^{(t)}} \\ 2\left(w_2^{(t)} - \tilde{w}_2 \right) x_{j^{(t)}} + 2\left(w_2^{(t)} - \tilde{w}_2 \right) \left(x_{j^{(t)}} \right)^2 \end{bmatrix} + \nabla \hat{R}_{\mathcal{S}} \left(\tilde{w} \right) \right) \tag{3.6}$$

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

(4) compute

$$\frac{1}{n} \sum_{i=1}^{n} \ell \left(\tilde{w}, z_i \right) = \begin{pmatrix} 2\tilde{w}_1 + 2\tilde{w}_2 \bar{x} - 2\bar{y} \\ 2\tilde{w}_1 \bar{x} + 2\tilde{w}_2 \overline{x^2} - 2y^\top x \end{pmatrix} = \nabla \hat{R}_{\mathcal{S}} \left(\tilde{w} \right) \tag{3.7}$$

(5) if a termination criterion is satisfied STOP

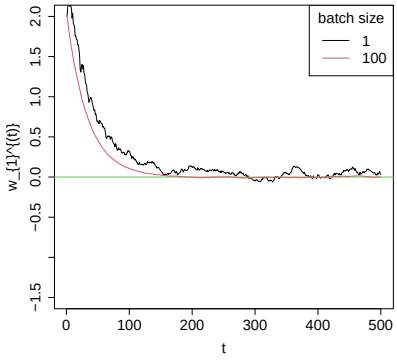
Because (3.7) is actually the gradient of the Risk function at $\tilde{w}$ and

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

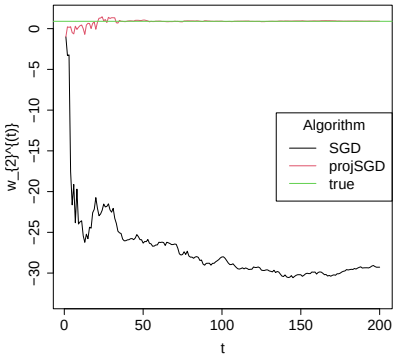
In Figure A.1c we observe the frequency of the snapshots has improved the convergence; however this is not a panacea as seen in Figure A.1f.

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

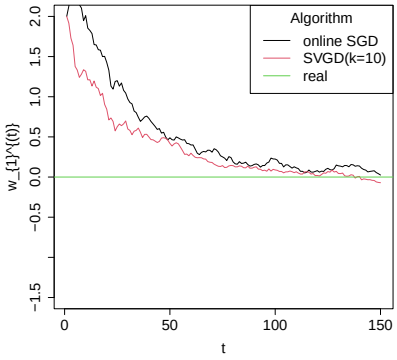
Appendix A. Plots



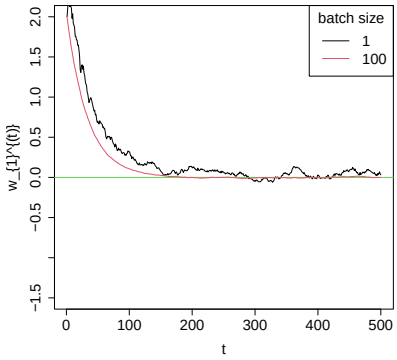
(A) batch SGD Example 30 w_1



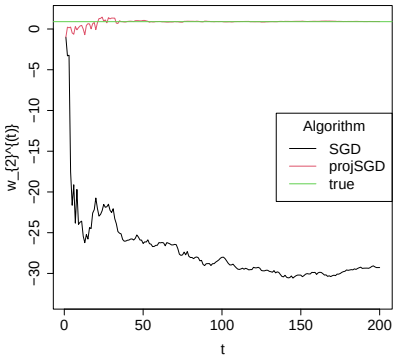
(B) online SGD with projection Example 6w_1



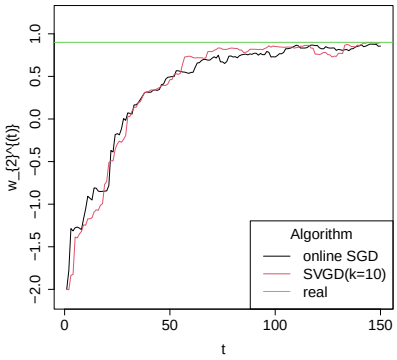
(C) SVRG Example 27 w_1



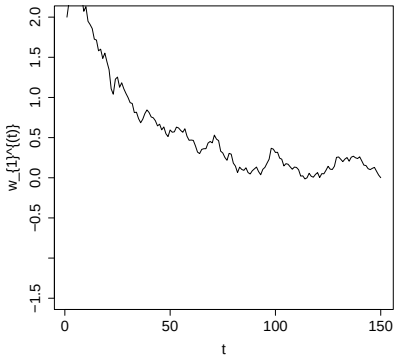
(D) batch SGD Example 30 w_2



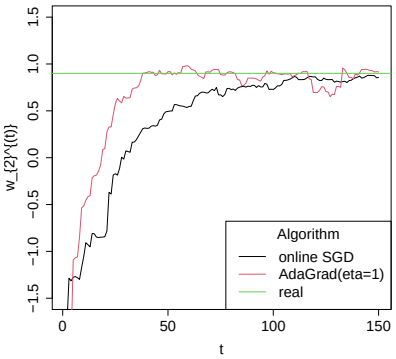
(E) online SGD with projection Example 6 w_2



(F) SVRG Example 27 w_2



(G) online AdaGrad Example 18w_1



(H) online AdaGrad Example 18w_2

FIGURE A.1. Simulations of the Examples