

## Handout 2: Gradient descent

Lecturer &amp; author: Georgios P. Karagiannis

georgios.karagiannis@durham.ac.uk

**Aim.** To introduce gradient descent, its motivation, description, practical tricks, analysis in the convex scenario, and implementation.

### Reading list & references:

- Bishop, C. M. (2006). Pattern recognition and machine learning. New York: Springer.
- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.

### 1. MOTIVATIONS

*Note 1.* Consider a learning problem  $(\mathcal{H}, \mathcal{Z}, \ell)$ . Learning may involve the computation of the minimizer  $h^* \in \mathcal{H}$ , where  $\mathcal{H}$  is a class of hypotheses, of the empirical risk function (ERF)  $\hat{R}(h) = \frac{1}{n} \sum_{i=1}^n \ell(h, z_i)$  given a finite sample  $\{z_i; i = 1, \dots, n\}$  generated from the data generating model  $g(\cdot)$  and using loss  $\ell(\cdot)$ ; that is

$$(1.1) \quad h^* = \arg \min_{h \in \mathcal{H}} (\hat{R}(h)) = \arg \min_{h \in \mathcal{H}} \left( \frac{1}{n} \sum_{i=1}^n \ell(h, z_i) \right)$$

If analytical minimization of (1.1) is impossible or impractical, numerical procedures can be applied; eg Gradient Descent (GD) algorithms. Such approaches introduce numerical errors in the solution.

### 2. DESCRIPTION

*Notation 2.* For the sake of notation simplicity and generalization, we will present Gradient Descent (GD) in the following minimization problem

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

where here  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , and  $w \in \mathcal{H} \subseteq \mathbb{R}^d$ ;  $f(\cdot)$  is the function to be minimized, e.g.,  $f(\cdot)$  can be an empirical risk function  $\hat{R}(\cdot)$ .

**Assumption 3.** Assume (for now) that  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is a differentiable function.

**Definition 4.** Given a pre-specified learning rate  $\eta_t > 0$ , the Gradient Descent (GD) algorithm for the solution of the minimization problem (2.1) is given in Algorithm 1

---

**Algorithm 1** Gradient descent algorithm with learning rate  $\eta_t$ 

---

For  $t = 1, 2, 3, \dots$  iterate:

(1) compute

$$(2.2) \quad w^{(t+1)} = w^{(t)} - \eta_t \nabla f(w^{(t)})$$

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

If  $t \geq T$  then STOP

---

where

$$\nabla f(w) = \left( \frac{\partial}{\partial x_1} f(x), \dots, \frac{\partial}{\partial x_d} f(x) \right)^\top \Big|_{x=w}$$

is the gradient of  $f$  at  $w$ .

*Remark 5.* Interpreting GD, GD produces a chain  $\{w^{(t)}\}$  that drifts towards the minimum  $w^*$ . It evolves directed against the direction of the gradient  $\nabla f(\cdot)$  and at a rate controlled by the learning rate  $\eta_t$ .

*Remark 6.* For more intuitive explanation, consider the (1st order) Taylor polynomial for the approximation of  $f(w)$  in a small area around  $u$  (i.e.  $\|v - u\| = \text{small}$ )

$$f(u) \approx P(u) = f(w) + \langle u - w, \nabla f(w) \rangle$$

Assuming convexity for  $f$ , it is

$$(2.3) \quad f(u) \geq \underbrace{f(w) + \langle u - w, \nabla f(w) \rangle}_{=P(u;w)}$$

See  
Handout 1

meaning that  $P$  lower bounds  $f$ . Hence we could design an updating mechanism producing  $w^{(t+1)}$  which is nearby  $w^{(t)}$  (small steps) and which minimize the linear approximation  $P(w)$  of  $f(w)$  at  $w^{(t)}$

$$(2.4) \quad P(w; w^{(t)}) = f(w^{(t)}) + \langle w - w^{(t)}, \nabla f(w^{(t)}) \rangle.$$

while hoping that this mechanism would push the produced chain  $\{w^{(t)}\}$  towards the minimum because of (2.3). Hence we could recursively minimize the linear approximation (2.4) and the distance between the current state  $w^{(t)}$  and the next  $w$  value to produce  $w^{(t+1)}$ ; namely

$$(2.5) \quad \begin{aligned} w^{(t+1)} &= \arg \min_{\forall w} \left( \frac{1}{2} \|w - w^{(t)}\|^2 + \eta P(w; w^{(t)}) \right) \\ &= \arg \min_{\forall w} \left( \frac{1}{2} \|w - w^{(t)}\|^2 + \eta \left( f(w^{(t)}) + \langle w - w^{(t)}, \nabla f(w^{(t)}) \rangle \right) \right) \\ &= w^{(t)} - \eta \nabla f(w^{(t)}) \end{aligned}$$

where parameter  $\eta > 0$  controls the trade off in (2.5).

*Remark 7.* Given  $T$  GD algorithm iterations, the output of GD can be (but not a exclusively),

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

$$(2.6) \quad w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$$

- (2) or the best value discovered

$$w_{\text{GD}}^{(T)} = \arg \min_{\forall w_t} \left( f \left( w^{(t)} \right) \right)$$

- (3) or the last value discovered

$$w_{\text{GD}}^{(T)} = w^{(T)}$$

*Note 8.* GD output converges to a local minimum,  $w_{\text{GD}}^{(T)} \rightarrow w_*$  (in some sense), under different sets of regularity conditions (some are weaker other stronger). Section 4 has a brief analysis.

*Remark 9.* The parameter  $\eta_t$  is called learning rate, step size, or gain.  $\{\eta_t\}$  is a non-negative sequence and it is chosen by the practitioner. In principle, regularity conditions (Note 8) often imply restrictions on the decay of  $\{\eta_t\}$  which guide the practitioner to parametrize it properly. Some popular choices of learning rate  $\eta_t$  are:

- (1) constant;  $\eta_t = \eta$ , for where  $\eta > 0$  is a small value. The rationale is that GD chain  $\{w_t\}$  performs constant small steps towards the (local) minimum  $w_*$  and then oscillate around it.
- (2) decreasing and converging to zero;  $\eta_t \searrow$  with  $\lim_{t \rightarrow \infty} \eta_t = 0$ . E.g.  $\eta_t = \left(\frac{C}{t}\right)^\varsigma$  where  $\varsigma \in [0.5, 1]$  and  $C > 0$ . The rationale is that GD algorithm starts by performing larger steps (controlled by  $C$ ) at the beginning to explore the area for discovering possible minima. Also it reduces the size of those steps with the iterations (controlled by  $\varsigma$ ) such that eventually when the chain  $\{w_t\}$  is close to a possible minimum  $w_*$  value to converge and do not overshoot.
- (3) decreasing and converging to a tiny value  $\tau_*$ ;  $\eta_t \searrow$  with  $\lim_{t \rightarrow \infty} \eta_t = \tau_*$  E.g.  $\eta_t = \left(\frac{C}{t}\right)^\varsigma + \tau_*$  with  $\varsigma \in (0.5, 1]$ ,  $C > 0$ , and  $\tau_* \approx 0$ . Same as previously, but the algorithm aims at oscillating around the detected local minimum.
- (4) constant until an iteration  $T_0$  and then decreasing; Eg  $\eta_t = \left(\frac{C}{\max(t, T_0)}\right)^\varsigma$  with  $\varsigma \in [0.5, 1]$  and  $C > 0$ , and  $T_0 < T$ . The rationale is that at the first stage of the iterations (when  $t \leq T_0$ ) the algorithm may need a constant large steps for a significant number of iterations  $T_0$  in order to explore the domain; and hence in order for the chain  $\{w_t\}$  to reach the area around the (local) minimum  $w_*$ . In the second stage, hoping that the chain  $\{w_t\}$  may be in close proximity to the (local) minimum  $w_*$  the algorithm progressively performs smaller steps to converge towards the minimum  $w_*$ . The first stage ( $t \leq T_0$ ) is called burn-in; the values  $\{w_t\}$  produced during the burn-in ( $t \leq T_0$ ) are often discarded/ignored from the output of the GD algorithm.

- Parameters  $C, \varsigma, \tau_*, T_0$  may be chosen based on pilot runs.

*Remark 10.* There are several practical termination criteria that can be used in GD Algorithm 1(step 2). They aim to terminate the recursion in practice. Some popular termination criteria are

- (1) terminate when the gradient is sufficiently close to zero; i.e. if  $\|\nabla f(w^{(t)})\| \leq \epsilon$  for some pre-specified tiny  $\epsilon > 0$  then STOP

(2) terminate when the chain  $w^{(t)}$  does not change; i.e. if  $\|w^{(t+1)} - w^{(t)}\| \leq \epsilon \|w^{(t)}\|$  for some pre-specified tiny  $\epsilon > 0$  then STOP

(3) terminate when a pre-specified number of iterations  $T$  is performed; i.e. if  $t \geq T$  then STOP

Here (1) may be deceive if the chain is in a flat area, (2) may be deceived if the learning rate become too small, (3) is obviously a last resort.

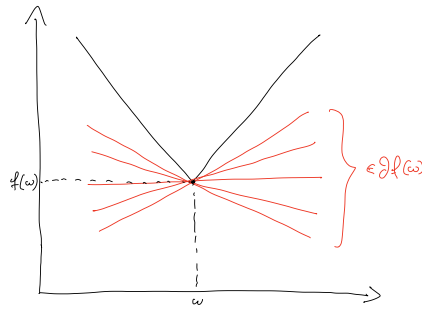
### 3. GD FOR NON-DIFFERENTIABLE FUNCTIONS (USING SUB-GRADIENTS)

*Note 11.* In several learning problems the function to be minimized is not differentiable. GD can be extended to address such problems with the use of subgradients.

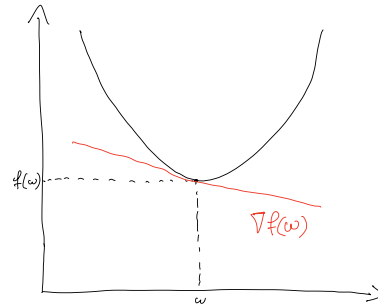
**Definition 12.** Vector  $v$  is called subgradient of a function  $f : S \rightarrow \mathbb{R}$  at  $w \in S$  if

$$(3.1) \quad \forall u \in S, \quad f(u) \geq f(w) + \langle u - w, v \rangle$$

*Note 13.* Essentially there may be more than one subgradients of the function at a specific point. As seen by (3.1), subgradients are the slopes of all the lines passing through the point  $(w, f(w))$  and been under the function  $f(\cdot)$ .



(A) subgradients satisfying (3.1) in the non-differentiable case



(B) gradient satisfying the equality in (3.1) in the differentiable case

*Notation 14.* The set of subgradients of function  $f : S \rightarrow \mathbb{R}$  at  $w \in S$  is denoted by  $\partial f(w)$ .

**Definition 15.** The Gradient Descent algorithm using subgradients in non-differentiable cases, results by replacing the gradient  $\nabla f(w^{(t)})$  in (2.2) with any of subgradient  $v_t$  from the set of subgradients  $\partial f(w^{(t)})$  at  $w^{(t)}$ ; namely

$$(3.2) \quad w^{(t+1)} = w^{(t)} - \eta_t v_t; \quad \text{where } v_t \in \partial f(w^{(t)})$$

#### 3.1. Construction of subgradient.

*Note 16.* We discuss how to construct subgradients in practice.

**Fact 17.** Some properties of subgradient sets that help for their construction

- (1) If function  $f : S \rightarrow \mathbb{R}$  is differentiable at  $w$  then the only subgradient of  $f$  at  $w$  is the gradient  $\nabla f(w)$ , and (3.1) is equality; i.e.  $\partial f(w) = \{\nabla f(w)\}$ .  
(2) for constants  $\alpha, \beta$  and convex function  $f(\cdot)$ , it is

$$\partial(\alpha f(w) + \beta) = \alpha(\partial f(w)) = \{\alpha v : v \in \partial f(w)\},$$

- (3) for convex functions  $f(\cdot)$  and  $g(\cdot)$ , it is

$$\partial(f(w) + g(w)) = \partial f(w) + \partial g(w) = \{v + u : v \in \partial f(w), \text{ and } u \in \partial g(w)\}$$

**Example 18.** Consider the function  $f : \mathbb{R} \rightarrow \mathbb{R}_+$  with  $f(w) = |w| = \begin{cases} w & w \geq 0 \\ -w & w < 0 \end{cases}$ . Find the set of subgradients  $\partial f(w)$  for each  $w \in \mathbb{R}$ .

**Solution.** Using Fact 17, it is  $\partial f(w) = 1$  for  $w > 0$  and  $\partial f(w) = -1$  for  $w < 0$  as  $f$  is differentiable for  $x \neq 0$ . At  $x = 0$ ,  $f$  is not differentiable; hence from condition (3.1) it is

$$\forall u \in \mathbb{R}, |u| \geq |0| + (u - 0)v$$

which is satisfied for  $v \in [-1, 1]$ . Hence,

$$\partial f(w) = \begin{cases} \{-1\} & , w < 0 \\ [-1, 1] & , w = 0 \\ \{1\} & , w > 0 \end{cases}$$

#### 4. ANALYSIS OF GRADIENT DESCENT

**Assumption 19.** For the sake of the analysis of the GD, let us consider:

- (1) constant learning rate  $\eta_t = \eta$ ,  
(2) GD output  $w_{GD}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$

Also as we will see, function  $f(\cdot)$  should be convex and Lipschitz fin order for the following results to hold

*Notation 20.*  $w^*$  is the minimizer in (2.1).

*Note 21.* We consider Lemma 22 as given.

**Lemma 22.** Let  $\{v_t; t = 1, \dots, T\}$  be a sequence of vectors. Any algorithm with  $w^{(1)} = 0$  and  $w^{(t+1)} = w^{(t)} - \eta v_t$  for  $t = 1, \dots, T$  satisfies

No need to  
memorize

$$(4.1) \quad \sum_{t=1}^T \langle w^{(t)} - w^*, v_t \rangle \leq \frac{\|w^*\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|v_t\|^2$$

*Proof.* Omitted; see the 2nd reference. □

*Note 23.* To find an upper bound of the GD error, we try to bound the error  $f(w_{GD}^{(T)}) - f(w^*)$  with purpose to use Lemma 22.

**Proposition 24.** Consider the minimization problem (2.1). Given Assumptions 19, the error can be bounded as

$$(4.2) \quad f(w_{GD}^{(T)}) - f(w^*) \leq \frac{1}{T} \sum_{t=1}^T \langle w^{(t)} - w^*, v_t \rangle$$

where  $v_t \in \partial f(w^{(t)})$ . If  $f(\cdot)$  is differentiable then  $v_t = \nabla f(w^{(t)})$

*Proof.* It is<sup>1</sup>

$$(4.3) \quad \begin{aligned} f(w_{GD}^{(T)}) - f(w^*) &= f\left(\frac{1}{T} \sum_{t=1}^T w_t\right) - f(w^*) \\ &\leq \frac{1}{T} \sum_{t=1}^T (f(w_t) - f(w^*)) && \text{(by Jensen's inequality)} \\ &\leq \frac{1}{T} \sum_{t=1}^T \langle w^{(t)} - w^*, \nabla f(w^{(t)}) \rangle && \text{(by convexity of } f(\cdot) \text{)} \end{aligned}$$

□

*Note 25.* The following provides conditions under which the gradient (or sub-gradient) is bounded. This is necessary in order to bound (4.2) with (4.1) in a meaningful manner.

**Proposition 26.** <sup>2</sup> $f : S \rightarrow \mathbb{R}$  is  $\rho$ -Lipschitz over an open convex set  $S$  if and only if for all  $w \in S$  and  $v \in \partial f(w)$  it is  $\|v\| \leq \rho$ .

*Proof.*  $\implies$  Let  $f : S \rightarrow \mathbb{R}$  be  $\rho$ -Lipschitz over convex set  $S$ ,  $w \in S$  and  $v \in \partial f(w)$ .

- Since  $S$  is open we get that there exist  $\epsilon > 0$  such as  $u := w + \epsilon \frac{v}{\|v\|}$  where  $u \in S$ . So  $\langle u - w, v \rangle = \epsilon \|v\|$  and  $\|u - w\| = \epsilon$ .
- From the subgradient definition we get

$$f(u) - f(w) \geq \langle u - w, v \rangle = \epsilon \|v\|$$

- From the Lipschitzness of  $f(\cdot)$  we get

$$f(u) - f(w) \leq \rho \|u - w\| = \rho \epsilon$$

Therefore  $\|v\| \leq \rho$ .

For  $\Leftarrow$  see Exercise 4 in the Exercise sheet. □

*Note 27.* The following summarizes Lemma 22, and Propositions 24, 26 with respect to the GD algorithm.

<sup>1</sup>Jensen's inequality for convex  $f(\cdot)$  is  $E(f(x)) \leq f(E(x))$

<sup>2</sup>If this was a Homework there would be a Hint:

- If  $S$  is open there exist  $\epsilon > 0$  such as  $u = w + \epsilon \frac{v}{\|v\|}$  such as  $u \in S$

**Proposition 28.** Let  $f(\cdot)$  be a convex and Lipschitz function. If we run GD algorithm of  $f$  with learning rate  $\eta > 0$  for  $T$  steps the output  $w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$  satisfies

$$f(w_{\text{GD}}^{(T)}) - f(w^*) \leq \frac{\|w^*\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^T \|\nabla f(w^{(t)})\|^2$$

**Solution.** Straightforward from Lemma 22, and Propositions 24, 26.

*Note 29.* The following shows that a given learning rate depending on the iteration  $t$ , we can reduce the upper bound of the error as well as find the number of required iterations to achieve convergence.

**Proposition 30.** Let  $f(\cdot)$  be a convex and Lipschitz function, and let  $\mathcal{H} = \{w \in \mathbb{R} : \|w\| \leq B\}$ . Assume we run GD algorithm of  $f(\cdot)$  with learning rate  $\eta_t = \sqrt{\frac{B^2}{\rho^2 T}}$  for  $T$  steps, and output  $w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$ . Then

(1) upper bound on the sub-optimality is

$$(4.4) \quad f(w_{\text{GD}}^{(T)}) - f(w^*) \leq \frac{B\rho}{\sqrt{T}}$$

(2) a given level off accuracy  $\varepsilon$  such that  $f(w_{\text{GD}}^{(T)}) - f(w^*) \leq \varepsilon$  can be achieved after  $T$  iterations

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

**Solution.** Part 1 is a simple substitution from Proposition 28, and part 2 is implied from part 1.

*Note 31.* The result on Proposition 30 heavily relies on setting suitable values for  $B$  and  $\rho$  which is rather a difficult task to be done in very complicated learning problems (e.g., learning a neural network).

*Remark 32.* The above results from the analysis of the GD also hold for the GD with subgradients; just replace  $\nabla f(\cdot)$  with any  $v_t \in \partial f(\cdot)$ .

## 5. EXAMPLES<sup>3</sup>

**Example 33.** Consider the simple Normal linear regression problem where the dataset  $\{z_i = (y_i, x_i)\}_{i=1}^n \in \mathcal{D}$  is generated from a Normal data generating model

$$\begin{pmatrix} y_i \\ x_i \end{pmatrix} \stackrel{\text{iid}}{\sim} \text{N} \left( \begin{pmatrix} \mu_y \\ \mu_x \end{pmatrix}, \begin{bmatrix} \sigma_y^2 & \rho \sqrt{\sigma_y^2 \sigma_x^2} \\ \rho \sqrt{\sigma_y^2 \sigma_x^2} & \sigma_x^2 \end{bmatrix} \right)$$

for  $i = 1, \dots, n$ . Consider a hypothesis space  $\mathcal{H}$  of linear functions  $h : \mathbb{R}^2 \rightarrow \mathbb{R}$  with  $h(w) = w_1 + w_2 x$ . The exact solution (which we pretend we do not know) is given as

$$\begin{pmatrix} w_1^* \\ w_2^* \end{pmatrix} = \begin{pmatrix} \mu_y - \rho \frac{\sigma_x}{\sigma_y} \mu_x \\ \rho \frac{\sigma_x}{\sigma_y} \end{pmatrix}.$$

<sup>3</sup>Code is available in [https://github.com/georgios-stats/Machine\\_Learning\\_and\\_Neural\\_Networks\\_III\\_Epiphany\\_2023/tree/main/Lecture\\_handouts/code/02.Gradient\\_descent/example\\_1.R](https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2023/tree/main/Lecture_handouts/code/02.Gradient_descent/example_1.R)

To learn the optimal  $w^* = (w_1^*, w_2^*)$ , we consider a loss  $\ell(w) = (y_i - [w_1 + w_2 x_i])^2$ , which leads to the minimization problem

$$w^* = \arg \min_{\forall w} \left( \hat{R}_{\mathcal{D}}(w) \right) = \arg \min_{\forall w} \left( \frac{1}{n} \sum_{i=1}^n (y_i - w_1 - w_2 x_i)^2 \right)$$

Consider data size  $n = 100$ , and parameters  $\rho = 0.2$ ,  $\sigma_y^2 = 1$  and  $\sigma_x^2 = 1$ . Consider a GD seed  $w_0 = (2, -2)$ .

Figures 5.1a, 5.1b, and 5.1c present trace plots of the chain  $\{(w^{(t)})\}$  and error  $\hat{R}_{\mathcal{D}}(w^{(t)}) - \hat{R}_{\mathcal{D}}(w^*)$  produced by running GD for  $T = 1000$  total iterations and for different (each time) constant learning rates  $\eta \in \{0.01, 0.02, 0.05, 0.99\}$ . We observe that the larger learning rates under consideration were able to converge faster to the minimum  $w^*$ . This is because they perform larger steps and can learn faster -this is not a panacea.

Figures 5.1d, 5.1e, and 5.1f present trace plots of the chain  $\{(w^{(t)})\}$ , and of the error  $\hat{R}_{\mathcal{D}}(w^{(t)}) - \hat{R}_{\mathcal{D}}(w^*)$  produced by running GD for  $T = 1000$  total iterations and for learning rate  $\eta = 1.0$  (previously considered) and a very big learning rate  $\eta = 3.0$ . We observe that the very big learning rate  $\eta = 3.0$  presents slower convergence to the minimum  $w^*$ . This is because it creates unreasonably big steps in (2.2) that the produced chain overshoots the global minimum. See the cartoon in Figures 5.1a and 5.1b.





(A)  $\{(w_1^{(t)})\}$



(B)  $\{(w_2^{(t)})\}$



(C)  $\hat{R}_D(w^{(t)}) - \hat{R}_D(w^*)$



(D)  $\{(w_1^{(t)})\}$



(E)  $\{(w_2^{(t)})\}$



(F)  $\hat{R}_D(w^{(t)}) - \hat{R}_D(w^*)$



(A) Unnecessarily large learning rate



(B) Unnecessarily small learning rate