Machine Learning and Neural Networks III (MATH3431)

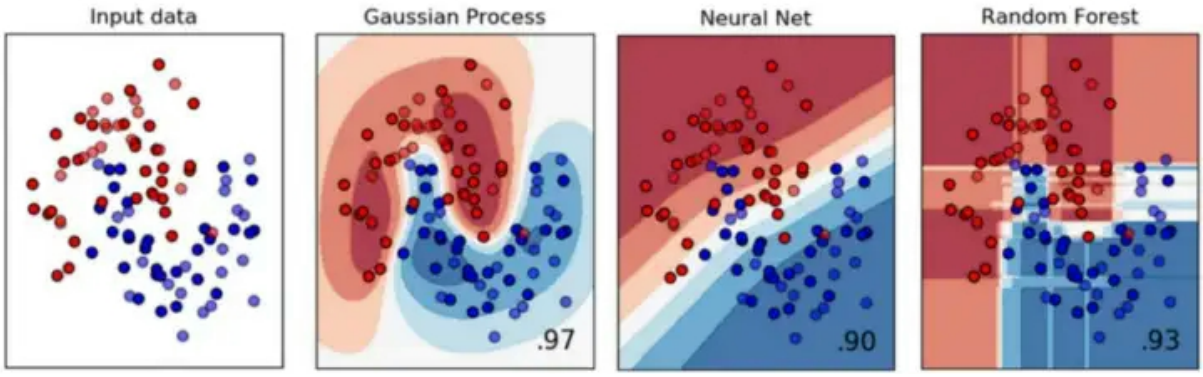
Epiphany term

Georgios P. Karagiannis

georgios.karagiannis@durham.ac.uk

Department of Mathematical Sciences (Office MCS3088)
Durham University
Stockton Road Durham DH1 3LE UK

2023/01/17 at 15:30:18



Reading list

These lecture Handouts have been derived based on the above reading list.

Main texts:

- Bishop, C. M. (2006). Pattern recognition and machine learning. New York: Springer.
  - It is a classical textbook in machine learning (ML) methods. It discusses all the concepts introduced in the course (not necessarily in the same depth). It is one of the main textbooks in the module. The level on difficulty is easy.
  - Students who wish to have a textbook covering traditional concepts in machine learning are suggested to get a copy of this textbook. It is available online from the Microsoft's website `https://www.microsoft.com/en-us/research/publication/pattern-recogniti`
- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
  - It has several elements of theory about machine learning algorithms. It is one of the main textbooks in the module. The level on difficulty is advanced as it requires moderate knowledge of maths.
- Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press.
  - It is a classical textbook about 'traditional' artificial neural networks (ANN). It is very comprehensive (compared to others) and it goes deep enough for the module although it may be a bit outdated. It is one of the main textbooks in the module for ANN. The level on difficulty is moderate.

Supplementary textbooks:

- Ripley, B. D. (2007). Pattern recognition and neural networks. Cambridge university press.
  - A classical textbook in artificial neural networks (ANN) that also covers other machine learning concepts. It contains interesting theory about ANN.
  - It is suggested to be used as a supplementary reading for neural networks as it contains a few interesting theoretical results. The level on difficulty is moderate.
- Williams, C. K., & Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.
  - A classic book in Gaussian process regression (GPR) that covers the material we will discuss in the course about GPR. It can be used as a companion textbook with that of (Bishop, C. M., 2006). The level on difficulty is easy.

- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
  - A popular textbook in machine learning methods. It discusses all the concepts introduced in the module. It focuses more on the probabilistic/Bayesian framework but not with great detail. It can be used as a comparison textbook for brief reading about ML methods just to see another perspective than that in (Bishop, C. M., 2006). The level on difficulty is easy.
- Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.
  - A textbook in machine learning methods. It covers a smaller number of ML concepts than (Murphy, K. P., 2012) but it contains more fancy/popular topics such as deep learning ideas. It is suggested to be used in the same manner as (Murphy, K. P., 2012). The level on difficulty is easy.
- Barber, D. (2012). Bayesian reasoning and machine learning. Cambridge University Press.
  - A textbook in machine learning methods from a Bayesian point of view. It discusses all the concepts introduced apart from ANN and stochastic gradient algorithms. It aims to be more 'statistical' than those of Murphy and Bishop. The level on difficulty is easy.
- Devroye, L., Györfi, L., & Lugosi, G. (2013). A probabilistic theory of pattern recognition (Vol. 31). Springer Science & Business Media.
  - Theoretical aspects about machine learning algorithms. The level on difficulty is advanced as it requires moderate knowledge of probability.

Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 0: Learning problem: Definitions, notation, and formulation –A recap

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

Aim. To get some definitions and set-up about the learning procedure; essentially to formalize what introduced in term 1.

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. Introductions and loose definitions

Pattern recognition is the automated discovery of patterns and regularities in data $z \in \mathcal{Z}$. **Machine learning (ML)** are statistical procedures for building and understanding probabilistic methods that 'learn'. **ML algorithms** $\mathfrak{A}$ build a (probabilistic/deterministic) model able to make predictions or decisions with minimum human interference and can be used for pattern recognition. **Learning** (or training, estimation) is called the procedure where the ML model is tuned. **Training data** (or observations, sample data set, examples) is a set of observables $\{z_i \in \mathcal{Z}\}$ used to tune the parameters of the ML model. By $\mathcal{Z}$ we denote the examples (or observables) domain. **Test set** is a set of available examples/observables $\{z_i'\}$ (different than the training data) used to verify the performance of the ML model for a given a measure of success. **Measure of success** (or performance) is a quantity that indicates how bad the corresponding ML model or Algorithm performs (eg quantifies the failure/error), and can also be used for comparisons among different ML models; eg, **Risk function** or **Empirical Risk Function**. Two main problems in ML are the supervised learning (we focus here) and the unsupervised learning.

Supervised learning problems involve applications where the training data $z \in \mathcal{Z}$ comprises examples of the input vectors $x \in \mathcal{X}$ along with their corresponding target vectors $y \in \mathcal{Y}$; i.e. $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. By $\mathcal{X}$ we denote the inputs (or instances) domain, and by $\mathcal{Y}$we denote the target domain. **Classification problems** are those which aim to assign each input vector x to one of a finite number of discrete categories of y. **Regression problems** are those where the output y consists of one or more continuous variables. All in all, the learner wishes to recover an unknown pattern (i.e. functional relationship) between components $x \in \mathcal{X}$ that serves as inputs and components $y \in \mathcal{Y}$ that act as outputs; i.e. $x \longmapsto y$. Hence, $\mathcal{X}$ is the input domain, and $\mathcal{Y}$ is the output (or target) domain. The goal of learning is to discover a function which predicts $y \in \mathcal{Y}$ from $x \in \mathcal{X}$.

Unsupervised learning problems involve applications where the training data $z \in \mathcal{Z}$ consist of a set of input vectors $x \in \mathcal{X}$ without any corresponding target values ; i.e. $\mathcal{Z} = \mathcal{X}$. In clustering the

goal is to discover groups of similar examples within the data of it is to discover groups of similar examples within the data.

2. (loose) Notation in learning

Definition 1. The learner's output is a function, $h : \mathcal{X} \to \mathcal{Y}$ which predicts $y \in \mathcal{Y}$ from $x \in \mathcal{X}$. It is also called hypothesis, prediction rule, predictor, or classifier.

Notation 2. We often denote the set of hypothesis as $\mathcal{H}$; i.e. $h \in \mathcal{H}$.

Example 3. (Linear Regression)[1] Consider the regression problem where the goal is to learn the mapping $x \to y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \subseteq \mathbb{R}$. Hypothesis is a linear function $h : \mathcal{X} \to \mathcal{Y}$ (that learner wishes to learn) to approximate mapping $x \to y$. The hypothesis set $\mathcal{H} = \left\{x \to \langle w, x\rangle : w \in \mathbb{R}^d\right\}$. We can use the loss $\ell(h, (x, y)) = (h(x) - y)^2$.

Definition 4. Training data set $\mathcal{S}$ of size m is any finite sequence of pairs $((x_i, y_i); i = 1, ..., m)$ in $\mathcal{X} \times \mathcal{Y}$. This is the information that the learner has assess.

Definition 5. Data generation model $g(\cdot)$ is the probability distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, unknown to the learner that has generated the data.

Definition 6. We denote as $\mathfrak{A}(\mathcal{S})$ the hypothesis (outcome) that a learning algorithm $\mathfrak{A}$ returns given training sample S.

Definition 7. (Loss function) Given any set of hypothesis $\mathcal{H}$ and some domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a loss function $\ell(\cdot)$ is any function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$. The purpose of loss function $\ell(h, z)$ is to quantify the "error" for a given hypothesis h and example z –the greater the error the greater its value of the loss.

Example 8. (Cont. Example 3) In regression problems $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y} \subset \mathbb{R}$ is uncountable, a loss function can be

$$\ell_{\text{sq}}(h, (x, y)) = (h(x) - y)^2$$

Example 9. In binary classification problems with $h : \mathcal{X} \to \mathcal{Y}$ a learner where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y} = \{0, 1\}$ is discrete, a loss function can be

$$\ell_{0-1}(h, (x, y)) = 1(h(x) \neq y),$$

Definition 10. (Risk function) The risk function $R_g(h)$ of h is the expected loss of the hypothesis $h \in \mathcal{H}$, w.r.t. probability distribution g over domain Z; i.e.

$$R_g(h) = \mathrm{E}_{z\sim g}(\ell(h, z)) \tag{2.1}$$

Remark 11. In learning, an ideal way to obtain an optimal predictor h^* is to compute the risk minimizer

$$h^* = \arg\min_{\forall h}(R_g(h))$$

[1] $\langle w, x\rangle = w^\top x$

Example 12. (Cont. Ex. 8) The risk function is $R_g(h) = \mathrm{E}_{z\sim g}(h(x)-y)^2$, and it measures the quality of the hypothesis function $h : \mathcal{X} \to \mathcal{Y}$, (or equiv. the validity of the class of hypotheses $\mathcal{H}$) against the data generating model g, as the expected square difference between the predicted values form h and the true target values y at every x.

Note 13. Computing the risk minimizer may be practically challenging due to the integration w.r.t. the unknown data generation model g involved in the expectation (2.1). Sub-optimally, one may resort to the Empirical risk function.

Definition 14. (Empirical risk function) The empirical risk function $\hat{R}_S(h)$ of h is the expectation of loss of h over a given sample $S = (z_1, ..., z_m) \in \mathcal{Z}^m$; i.e.

$$\hat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i).$$

Example 15. (Cont. Example 12) Given given sample $S = \{(x_i, y_i); i = 1, ..., m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(x_i)-y_i)^2$.

Example 16. Consider a learning problem where the true data generation distribution (unknown to the learner) is $g(z)$, the statistical model (known to the learner) is given by a sampling distribution $f_\theta(y) := f(y|\theta)$ labeled by an unknown parameter θ. The goal is to learn θ. If we assume loss function

$$\ell(\theta, z) = \log\left(\frac{g(z)}{f_\theta(z)}\right)$$

then the risk is

$$R_g(\theta) = \mathrm{E}_{z\sim g}\left(\log\left(\frac{g(z)}{f_\theta(z)}\right)\right) = \mathrm{E}_{z\sim g}(\log(g(z))) - \mathrm{E}_{z\sim g}(\log(f_\theta(z))) \tag{2.2}$$

whose minimizer is

$$\theta^* = \arg\min_{\forall\theta}(R_g(\theta)) = \arg\min_{\forall\theta}(\mathrm{E}_{z\sim g}(-\log(f_\theta(z))))$$

as the first term in (2.2) is constant. Note that in the Maximum Likelihood Estimation technique the MLE θ_{MLE} is the minimizer of

$$\theta_{\mathrm{MLE}} = \arg\min_{\forall\theta}\left(\frac{1}{m}\sum_{i=1}^{m}(-\log(f_\theta(z_i)))\right)$$

where $S = \{z_1, ..., z_m\}$ is an IID sample from g. Hence, MLE θ_{MLE} can be considered as the minimizer of the empirical risk $R_S(\theta) = \frac{1}{m}\sum_{i=1}^{m}(-\log(f_\theta(z_i)))$.

Definition 17. A learning problem with hypothesis class $\mathcal{H}$, examples domain $\mathcal{Z}$, and loss function ℓ may be denoted with a triplet $(\mathcal{H}, \mathcal{Z}, \ell)$.

Example 18. Consider the multiple linear regression problem $x \longmapsto y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \subseteq \mathbb{R}$. Till now, we set the learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ in the linear regression with hypothesis class $\mathcal{H} = \left\{x \to \langle w, x\rangle : w \in \mathbb{R}^d\right\}$, loss $\ell(h, (x, y)) = (h(x)-y)^2$, and examples domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{Y} \in \mathbb{R}$. Because learning problem involves only linear functions as predictors

$h(x) = \langle w, x \rangle$, this learning problem could be defined equivalently with a hypothesis class $\mathcal{H} = \left\{ w \in \mathbb{R}^d \right\}$ and loss function loss $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$. The latter will be mainly used.

Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 1: Elements of convex learning problems

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

Aim. To introduce elements of convexity, Lipschitzness, and smoothness that can be used for the analysis of stochastic gradient related learning algorithms.

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. We introduce concepts of convexity and smoothness that facilitate the (theoretical) analysis of the learning problems and their solution that we will discuss (eg stochastic gradient descent) later on. Also learning problems with such characteristics can be learned more efficiently.

Note 2. Most of the ML problems discussed in the course (eg, Artificial neural networks, Gaussian process regression) are usually non-convex.

Note 3. To overcome this problem, we will introduce the concept of surrogate loss function that allows a non-convex problem to be handled with the tools introduced int he convex setting.

2. Convex learning problems

Definition 4. Convex learning problem is a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ that the hypothesis class $\mathcal{H}$ is a convex set, and the loss function ℓ is a convex function for each example $z \in \mathcal{Z}$.

Example 5. Multiple linear regression $< w, x > \to y$ with $y \in \mathbb{R}$, hypothesis class $\mathcal{H} = \left\{w \in \mathbb{R}^d\right\}$ and loss $\ell\left(w, (x, y)\right) = \left(< w, x > -y\right)^2$ with

$$w^* = \arg\min_{\forall w} \mathrm{E}\left(< w, x > -y\right)^2$$

or

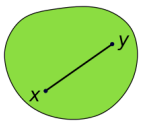
$$w^{**} = \arg\min_{\forall w} \frac{1}{m} \sum_{i=1}^{m} \left(< w, x_i > -y\right)^2$$

is a convex learning problem for reasons that will be discussed below.

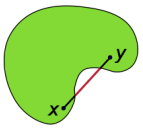
3. Convexity

Definition 6. A set C is convex if for any $u, v \in C$, the line segment between u and v is contained in C. Namely,

- for any $u, v \in C$ and for any $\alpha \in [0, 1]$ we have that $\alpha u + (1 - \alpha) v \in C$.



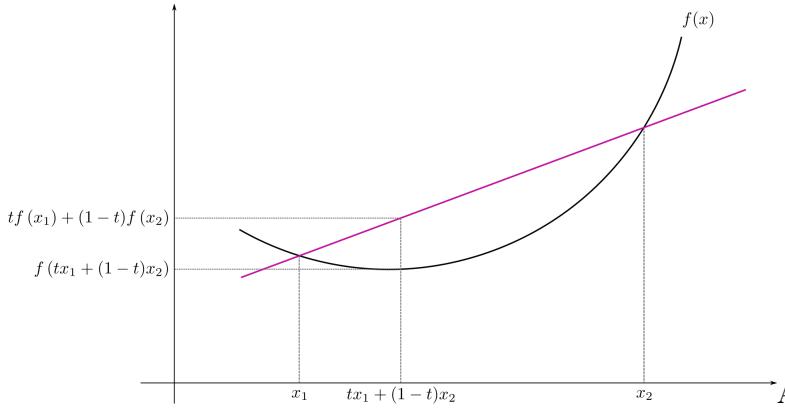
A convex set



A non-convex set

Definition 7. Let C be a convex set. A function $f : C \to R$ is convex function if for any $u, v \in C$ and for any $\alpha \in [0, 1]$

$$f(\alpha u + (1-\alpha) v) \leq \alpha f(u) + (1-\alpha) f(v)$$



A convex function

Example 8. The function $f : \mathbb{R} \to \mathbb{R}_+$ with $f(x) = x^2$ is convex function. For any $u, v \in C$ and for any $\alpha \in [0, 1]$ it is

$$(\alpha u + (1-\alpha) v)^2 - \alpha (u)^2 + (1-\alpha) (v)^2 = -\alpha (1-\alpha) (u-v)^2 \leq 0$$

Proposition 9. *Every local minimum of a convex function is the global minimum.*

Proposition 10. *Let $f : C \to \mathbb{R}$ be convex function. The tangent of f at $w \in C$ is below f, namely*

$$\forall u \in C \quad f(u) \geq f(w) + \langle \nabla f(w), u - w \rangle$$

Proposition 11. *Let $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(w) = g(\langle w, x \rangle + y)$ for some $x \in \mathbb{R}^d$, $y \in \mathbb{R}$. If g is convex function then f is convex function.*

Proof. See Exercise 1 in the Exercise sheet. □

Example 12. Consider the regression problem $x \mapsto y$ with $x \in \mathbb{R}^d$, $y \in \mathbb{R}$ and predictor $h(x) = \langle w, x \rangle$. The loss function $\ell(w, (x, y)) = (\langle w, x \rangle + y)^2$ is convex because $g(a) = (a)^2$ is convex and Proposition 11.

Example 13. Let $f_j : \mathbb{R}^d \to \mathbb{R}$ convex functions for $j = 1, ..., r$. Then:

(1) $g(x) = \max_{\forall j} (f_j(x))$ is a convex function
(2) $g(x) = \sum_{j=1}^{r} w_j f_j(x)$ is a convex function where $w_j > 0$

Solution.

(1) For any $u, v \in \mathbb{R}^d$ and for any $\alpha \in [0,1]$

$$\begin{aligned} g(\alpha u + (1-\alpha) v) =& \max_{\forall j} (f_j(\alpha u + (1-\alpha) v)) \\ \leq& \max_{\forall j} (\alpha f_j(u) + (1-\alpha) f_j(v)) && (f_j \text{ is convex}) \\ \leq& \alpha \max_{\forall j} (f_j(u)) + (1-\alpha) \max_{\forall j} (f_j(v)) && (\max(\cdot) \text{ is convex}) \\ \leq& \alpha g(u) + (1-\alpha) g(v) \end{aligned}$$

(2) For any $u, v \in \mathbb{R}^d$ and for any $\alpha \in [0,1]$

$$\begin{aligned} g(\alpha u + (1-\alpha) v) =& \sum_{j=1}^{r} w_j f_j(\alpha u + (1-\alpha) v) \\ \leq& \alpha \sum_{j=1}^{r} w_j f_j(u) + (1-\alpha) \sum_{j=1}^{r} w_j f_j(v) && (f_j \text{ is convex}) \\ \leq& \alpha g(u) + (1-\alpha) g(v) \end{aligned}$$

Example 14. $g(x) = |x|$ is convex according to Example 13, as $g(x) = |x| = \max(-x, x)$.

4. Lipschitzbness

Definition 15. Let $C \in \mathbb{R}^d$. Function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $w_1, w_2 \in C$ we have that

$$\|f(w_1) - f(w_2)\| \leq \rho \|w_1 - w_2\|. \qquad \text{Lipschitz condition} \tag{4.1}$$

Conclusion 16. That means: a Lipschitz function $f(x)$ cannot change too drastically wrt x.

Example 17. Consider the function $f : \mathbb{R} \to \mathbb{R}_+$ with $f(x) = x^2$.

(1) f is not a ρ-Lipschitz in $\mathbb{R}$.
(2) f is a ρ-Lipschitz in $C = \{x \in \mathbb{R} : |x| < \rho/2\}$.

$$|f(x_2) - f(x_1)| = \left|x_2^2 - x_1^2\right| = |(x_2 + x_1)(x_2 - x_1)| \leq 2\rho/2 (x_2 - x_1) = \rho |x_2 - x_1|$$

Solution.

(1) For $x_1 = 0$ and $x_2 = 1 + \rho$, it is

$$|f(x_2) - f(x_1)| = (1+\rho)^2 > \rho(1+\rho) = \rho |x_2 - x_1|$$

(2) It is

$$|f(x_2) - f(x_1)| = \left|x_2^2 - x_1^2\right| = |(x_2 + x_1)(x_2 - x_1)| \leq 2\rho/2 (x_2 - x_1) = \rho |x_2 - x_1|$$

Theorem 18. *Let functions g_1 be ρ_1-Lipschitz and g_2 be ρ_2-Lipschitz. Then f with $f(x) = g_1(g_2(x))$ is $\rho_1\rho_2$-Lipschitz.*

Solution. See Exercise 2 from the exercise sheet

Example 19. Let functions g be ρ-Lipschitz. Then f with $f(x) = g(< v, x > +b)$ is ρ-Lipschitz.

$$\begin{aligned}|f(w_1) - f(w_2)| = &|g(< v, w_1 > +b) - g(< v, w_2 > +b)| \leq \rho |< v, w_1 > +b - < v, w_2 > -b| \\ \leq &\rho_1 \left| v^\top w_1 - v^\top w_2 \right| \leq \rho_1 |v| |w_1 - w_2|\end{aligned}$$

Note 20. So, given Examples 17 and 19, in the linear regression setting using loss $\ell(w, z) = \left(w^\top x - z\right)^2$, the loss unction is -Lipschitz for a given z and and bounded $\|w\| < \rho$.

5. Smoothness

Definition 21. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz; namely for all $v, w \in \mathbb{R}^d$

$$\|\nabla f(w_1) - \nabla f(w_2)\| \leq \beta \|w_1 - w_2\|. \tag{5.1}$$

Theorem 22. *Function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth iff*

$$f(v) \leq f(w) + < \nabla f(w), v - w > + \frac{\beta}{2} \|v - w\|^2 \tag{5.2}$$

Remark 23. If $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth then (5.2) holds, and if it is convex as well then

$$f(v) \geq f(w) + < \nabla f(w), v - w >$$

holds. Hence if both conditions imply upper and lower bounds

$$f(v) - f(w) \in \left(< \nabla f(w), v - w >, < \nabla f(w), v - w > + \frac{\beta}{2} \|v - w\|^2 \right)$$

Remark 24. If $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth then for $v, w \in \mathbb{R}^d$ such that $v = w - \frac{1}{\beta} \nabla f(w)$ then by (5.2), it is

$$\frac{1}{2\beta} \|\nabla f(w)\|^2 \leq f(w) - f(v)$$

If additionally $f(x) > 0$ for all $x \in \mathbb{R}^d$ then

$$\|\nabla f(w)\|^2 \leq 2\beta f(w)$$

which provides assumptions to bound the gradient.

Theorem 25. *Let $f : \mathbb{R}^d \to \mathbb{R}$ with $f(w) = g(\langle w, x \rangle + y)$ $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Let $g : \mathbb{R} \to \mathbb{R}$ be a β-smooth function. Then f is a $\left(\beta \|x\|^2\right)$-smooth.*

Solution. See Exercise 3 from the Exercise sheet

Example 26. Let $f(w) = (< w, x > +y)^2$ for $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Then f is $\left(2 \|x\|^2\right)$-smooth.

Solution. It is $f(w) = g(< w, x > +y)$ for $g(a) = a^2$. g is 2-smooth since

$$\left\|g'(w_1) - g'(w_2)\right\| = \|2w_1 - 2w_2\| \leq 2 \|w_1 - w_2\|.$$

Hence from (25), f is $\left(2 \|x\|^2\right)$-smooth.

Remark 27. A a learning problem may involve non-convex loss function $\ell(w,z)$ which implies a non-convex risk function $R_g(w)$. However, our learning algorithm will be analyzed in the convex setting. A suitable treatment to overcome this difficulty would be to upper bound the non-convex loss function $\ell(w,z)$ by a convex surrogate loss function $\tilde{\ell}(w,z)$ for all w, and use $\tilde{\ell}(w,z)$ instead of $\ell(w,z)$.

Example 28. Consider the binary classification problem with inputs $x \in \mathcal{X}$, outputs $y \in \{-1,+1\}$; we need to learn $w \in \mathcal{H}$ from hypothesis class $\mathcal{H} \subset \mathbb{R}^d$ with respect to the loss

$$\ell(w,(x,y)) = 1_{(y\langle w,x\rangle \leq 0)}$$

with $y \in \mathbb{R}$, and $x \in \mathbb{R}^d$. Here $\ell(\cdot)$ is non-convex. A convex surrogate loss function can be

$$\tilde{\ell}(w,(x,y)) = \max(0, 1 - y\langle w,x\rangle)$$

which is convex (Example 14) wrt w. Note that:

- $\tilde{\ell}(w,(x,y))$ is convex wrt w ; because $\max(\cdot)$ is convex
- $\ell(w,(x,y)) \leq \tilde{\ell}(w,(x,y))$ for all $w \in \mathcal{H}$

Then we can compute

$$\tilde{w}_* = \arg\min_{\forall x}\left(\tilde{R}_g(w)\right) = \arg\min_{\forall x}\left(\mathrm{E}_{(x,y)\sim g}\left(\max(0,1-y\langle w,x\rangle)\right)\right)$$

instead of

$$w_* = \arg\min_{\forall x}\left(R_g(w)\right) = \arg\min_{\forall x}\left(\mathrm{E}_{(x,y)\sim g}\left(1_{(y\langle w,x\rangle\leq 0)}\right)\right)$$

Of course by using the surrogate loss instead of the actuall one, we introduce some approximation error in the produced output $\tilde{w}_* \neq w_*$.

Remark 29. (Intuitions...) Using a convex surrogate loss function instead the convex one, fascilitates computations but introduces extra error to the solution. If $R_g(\cdot)$ is the risk under the non-convex loss, $\tilde{R}_g(\cdot)$ is the risk under the convex surrogate loss, and $\tilde{w}_{\text{alg}}$ is the output of the learning algorithm under $\tilde{R}_g(\cdot)$ then we have the upper bound

$$R_g(\tilde{w}_{\text{alg}}) \leq \underbrace{\min_{w\in\mathcal{H}}\left(R_g(w)\right)}_{\text{I}} + \underbrace{\left(\min_{w\in\mathcal{H}}\left(\tilde{R}_g(w)\right) - \min_{w\in\mathcal{H}}\left(R_g(w)\right)\right)}_{\text{II}} + \underbrace{\epsilon}_{\text{III}}$$

where term I is the approximation error measuring how well the hypothesis class performs on the generating model, term II is the optimization error due to the use of surrogate loss instead of the actual non-convex one, and term III is the estimation error due to the use of a training set and not the whole generation model.

Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 0: Learning problem: Definitions, notation, and formulation –A recap

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

Aim. To get some definitions and set-up about the learning procedure; essentially to formalize what introduced in term 1.

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. Introductions and loose definitions

Pattern recognition is the automated discovery of patterns and regularities in data $z \in \mathcal{Z}$. **Machine learning (ML)** are statistical procedures for building and understanding probabilistic methods that 'learn'. **ML algorithms** $\mathfrak{A}$ build a (probabilistic/deterministic) model able to make predictions or decisions with minimum human interference and can be used for pattern recognition. **Learning** (or training, estimation) is called the procedure where the ML model is tuned. **Training data** (or observations, sample data set, exemplars) is a set of observables $\{z_i \in \mathcal{Z}\}$ used to tune the parameters of the ML model. **Test set** is a set of available examples/observables $\{z_i'\}$ (different than the training data) used to verify the performance of the ML model for a given a measure of success. **Measure of success** (or performance) is a quantity that indicates how bad the corresponding ML model or Algorithm performs (eg quantifies the failure/error), and can also be used for comparisons among different ML models; eg, **Risk function** or **Empirical Risk Function**. Two main problems in ML are the supervised learning (we focus here) and the unsupervised learning.

Supervised learning problems involve applications where the training data $z \in \mathcal{Z}$ comprises examples of the input vectors $x \in \mathcal{X}$ along with their corresponding target vectors $y \in \mathcal{Y}$; i.e. $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. **Classification problems** are those which aim to assign each input vector x to one of a finite number of discrete categories of y. **Regression problems** are those where the output y consists of one or more continuous variables. All in all, the learner wishes to recover an unknown pattern (i.e. functional relationship) between components $x \in \mathcal{X}$ that serves as inputs and components $y \in \mathcal{Y}$ that act as outputs; i.e. $x \longmapsto y$. Hence, $\mathcal{X}$ is the input domain, and $\mathcal{Y}$ is the output domain. The goal of learning is to discover a function which predicts $y \in \mathcal{Y}$ from $x \in \mathcal{X}$.

Unsupervised learning problems involve applications where the training data $z \in \mathcal{Z}$ consists of a set of input vectors $x \in \mathcal{X}$ without any corresponding target values ; i.e. $\mathcal{Z} = \mathcal{X}$. In clustering the goal is to discover groups of similar examples within the data of it is to discover groups of similar examples within the data.

2. (loose) Notation in learning

Definition 1. The learner's output is a function, $h : \mathcal{X} \to \mathcal{Y}$ which predicts $y \in \mathcal{Y}$ from $x \in \mathcal{X}$. It is also called hypothesis, prediction rule, predictor, or classifier.

Notation 2. We often denote the set of hypothesis as $\mathcal{H}$; i.e. $h \in \mathcal{H}$.

Definition 3. Training data set $\mathcal{S}$ of size m is any finite sequence of pairs $((x_i, y_i) ; i = 1, ..., m)$ in $\mathcal{X} \times \mathcal{Y}$. This is the information that the learner has assess.

Definition 4. Data generation model $g(\cdot)$ is the probability distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, unknown to the learner that has generated the data.

Definition 5. We denote as $\mathfrak{A}(\mathcal{S})$ the hypothesis (outcome) that a learning algorithm $\mathfrak{A}$ returns given training sample S.

Definition 6. (Loss function) Given any set of hypothesis $\mathcal{H}$ and some domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a loss function $\ell(\cdot)$ is any function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$. The purpose of loss function $\ell(h, z)$ is to quantify the "error" for a given hypothesis h and example z –the greater the error the greater its value of the loss.

Example 7. In binary classification problems where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y} = \{0, 1\}$ is discrete, a loss function can be

$$\ell_{0-1}(h, (x, y)) = 1(h(x) = y),$$

Example 8. In regression problems $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y}$ is uncountable, a loss function can be

$$\ell_{\text{sq}}(h, (x, y)) = (h(x) - y)^2$$

Definition 9. (Risk function) The risk function $R_g(h)$ of h is the expected loss of the hypothesis $h \in \mathcal{H}$, w.r.t. probability distribution g over domain Z; i.e.

$$R_g(h) = \mathrm{E}_{z \sim g}(\ell(h, z)) \tag{2.1}$$

Remark 10. In learning, an ideal way to obtain an optimal predictor h^* is to compute the risk minimizer

$$h^* = \arg\min_{\forall}(R_g(h))$$

Example 11. (Cont. Ex. 8)The risk function is $R_g(h) = \mathrm{E}_{z \sim g}(h(x) - y)^2$, and it measures the quality of the hypothesis function $h : \mathcal{X} \to \mathcal{Y}$ as the expected square difference between the predicted values h and the true target values y at every x.

Remark 12. Computing the risk minimizer may be practically challenging due to the integration w.r.t. the unknown data generation model g involved in the expectation (2.1). Suboptimally, one may resort to the Empirical risk function.

Definition 13. (Empirical risk function) The empirical risk function $\hat{R}_S(h)$ of h is the expectation of loss of h over a given sample $S = (z_1, ..., z_m) \in \mathcal{Z}^m$; i.e.

$$\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i).$$

Example 14. (Cont. Example 11) Given given sample $S = \{(x_i, y_i); i = 1, ..., m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$.

Example 15. Consider a learning problem where the true data generation distribution (unknown to the learner) is $g(z)$, the statistical model (known to the learner) is given by a sampling distribution $f(y|\theta)$ where the parameter θ is unknown. The goal is to learn θ. If we assume loss function

$$\ell(\theta, z) = \log\left(\frac{g(z)}{f_\theta(z)}\right)$$

then the risk is

$$R_g(\theta) = \mathrm{E}_{z\sim g}\left(\log\left(\frac{g(z)}{f_\theta(z)}\right)\right) = \mathrm{E}_{z\sim g}(\log(g(z))) - \mathrm{E}_{z\sim g}(\log(f_\theta(z))) \tag{2.2}$$

whose minimizer is

$$\theta^* = \arg\min_{\forall\theta}(R_g(\theta)) = \arg\min_{\forall\theta}(\mathrm{E}_{z\sim g}(-\log(f_\theta(z))))$$

as the first term in (2.2) is constant. Note that in the Maximum Likelihood Estimation technique the MLE θ_{MLE} is the minimizer of

$$\theta_{\mathrm{MLE}} = \arg\min_{\forall\theta}\left(\frac{1}{m}\sum_{i=1}^{m}(-\log(f_\theta(z_i)))\right)$$

where $S = \{y_1, ..., y_m\}$ is an IID sample from g. Hence, MLE θ_{MLE} can be considered as the minimizer of the empirical risk $R_S(\theta) = \frac{1}{m}\sum_{i=1}^{m}(-\log(f_\theta(z_i)))$.

Example 16. (Linear Regression) Consider the multiple linear regression problem $x \longmapsto y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \subseteq \mathbb{R}^d$.

- The hypothesis is a linear function $h : \mathcal{X} \to \mathcal{Y}$ (that learner wishes to learn) to approximate mapping $x \longmapsto y$. The hypothesis set $\mathcal{H} = \left\{< w, x > \longmapsto y : w \in \mathbb{R}^d\right\}$. We can use the loss $\ell(h, (x, y)) = (h(x) - y)^2$.
- Equivalently, learning problem can be set differently because the predictor (linear function) is parametrized by $w \in \mathbb{R}^d$ as $< w, x > \longmapsto y$. Set $\mathcal{H} = \left\{w \in \mathbb{R}^d\right\}$. The set of examples is $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The loss is $\ell(w, (x, y)) = (< w, x > -y)^2$.

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`

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

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

for $i = 1, ..., n$.

Consider a hypothesis space $\mathcal{H}$ of linear functions $h : \mathbb{R}^2 \to \mathbb{R}$ with

$$h(w) = w_1 + w_2 x$$

The exact solution (which we pretend we do not know) is given as

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

To learn the optimal $w^* = (w_1^*, w_2^*)$, we consider a loss

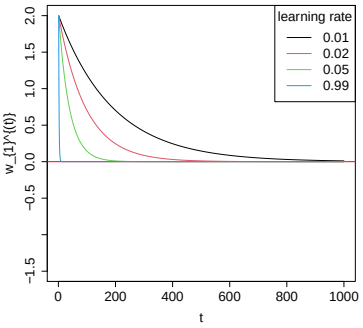
$$\ell(w) = (y_i - [w_1 + w_2 x_i])^2$$

which leads to the minimization problem

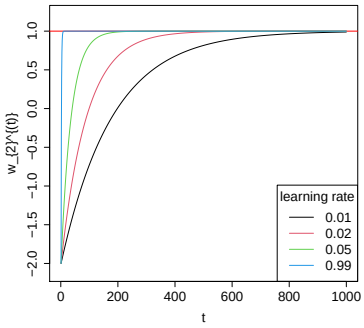
$$w^* = \arg\min_{\forall w} \left(\hat{R}_{\mathcal{D}}(w) \right) = \arg\min_{\forall w} \left(\frac{1}{n} \sum_{i=1}^n (y_i - w_1 - w_2 x_i)^2 \right)$$

Consider data size $n = 100$, and parameters $\rho = 0.2$, $\sigma_y^2 = 1$ and $\sigma_x^2 = 1$.

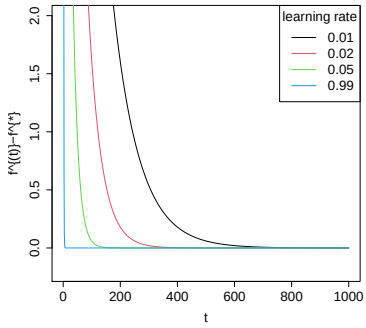
Consider a GD seed $w_0 = (2, -2)$.



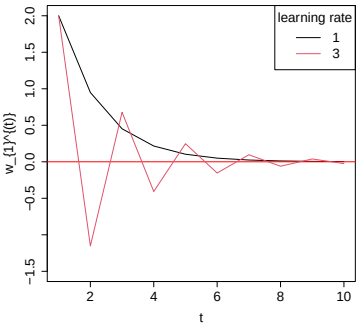
(A) $\left\{\left(w_1^{(t)}\right)\right\}$



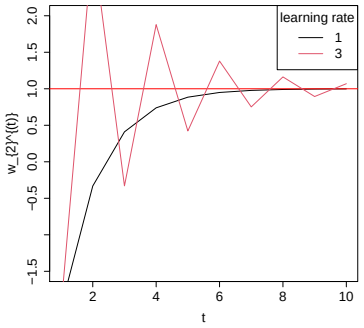
(B) $\left\{\left(w_2^{(t)}\right)\right\}$



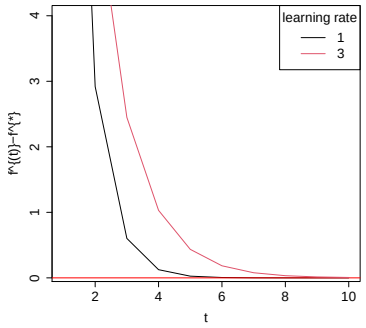
(C) $\hat{R}_{\mathcal{D}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{D}}\left(w^*\right)$



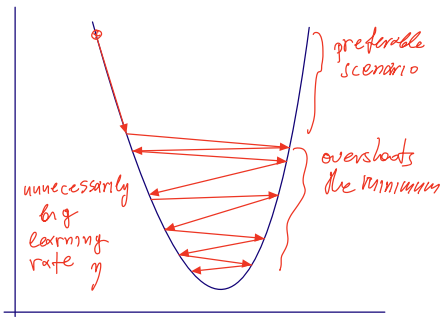
(D) $\left\{\left(w_1^{(t)}\right)\right\}$



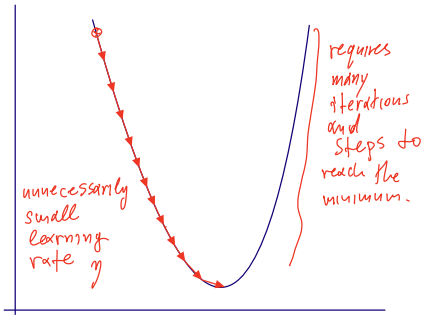
(E) $\left\{\left(w_2^{(t)}\right)\right\}$



(F) $\hat{R}_{\mathcal{D}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{D}}\left(w^*\right)$



(A) Unnecessarily large learning rate



(B) Unnecessarily small learning rate

Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 2: Gradient descent

Lecturer & 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.

**Subject to adjustments

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 $\{z_i; i = 1, ..., n\}$ generated from the data generating model $g(\cdot)$ and using loss $\ell(\cdot)$; that is

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

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

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

where here $f : \mathbb{R}^d \to \mathbb{R}$, 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 $f : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function.*

Definition 4. Given a per-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 η_t

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

(1) compute

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

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

If $t \geq T$ then STOP

where

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

is the gradient of f at w.

Remark 5. Intuitively, GD produces a chain $\left\{w^{(t)}\right\}$ 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 η_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

See Handout 1

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

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 $w^{(t)}$

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

while hoping that this mechanism would push the produced chain $\left\{w^{(t)}\right\}$ 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

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

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)

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

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

Remark 9. The parameter η_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 η_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 $\lim_{t\to\infty}\eta_t = 0$. 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 begining to explore the area for discovering possible minima. Also it reduces the size of those steps with the iterations (controled by ς) 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 τ_*; $\eta_t \searrow$ with $\lim_{t\to\infty}\eta_t = \tau_*$ 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 $t \le T_0$) 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 ($t \le T_0$) is called burn-in; the values $\{w_t\}$ produced during the burn-in ($t \le T_0$) are 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 $\left\|\nabla f\left(w^{(t)}\right)\right\| \le \epsilon$ for some pre-specified tiny $\epsilon > 0$ then STOP

(2) terminate when the chain $w^{(t)}$ does not change; i.e. if $\left\|w^{(t+1)}-w^{(t)}\right\| \leq \epsilon \left\|w^{(t)}\right\|$ 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.

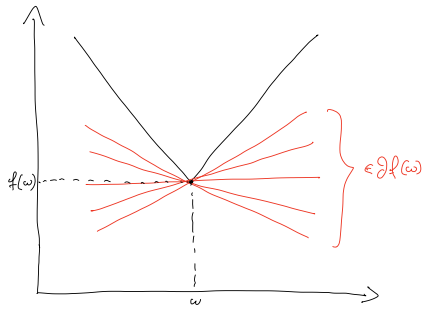
3. GD for non-differentiable functions (using sub-gradients)

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.

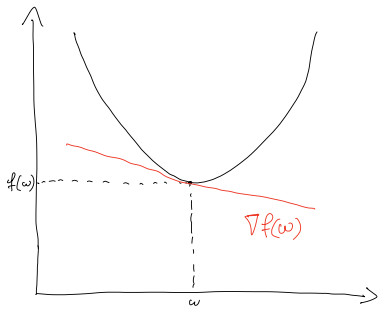
Definition 12. Vector v is called subgradient of a function $f : S \to \mathbb{R}$ at $w \in S$ if

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

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 $f : S \to \mathbb{R}$ 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 $\nabla f\left(w^{(t)}\right)$ in (2.2) with any of subgradient v_t from the set of subgradients $\partial f\left(w^{(t)}\right)$ at $w^{(t)}$; namely

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

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 $f : S \to \mathbb{R}$ 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 α, β 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*

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

Example 18. Consider the function $f : \mathbb{R} \to \mathbb{R}_+$ 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.

No need to memorize

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

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

Proof. See Exercise [TBD], from the Exercise sheet. □

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

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

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

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

Proof. It is[1]

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

□

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. [2]*$f : S \to \mathbb{R}$ is ρ-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. $\Longrightarrow$ Let $f : S \to \mathbb{R}$ be ρ-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

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

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

For $\Longleftarrow$ see Exercise [TBD] in the Exercise sheet. □

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

[1] Jensen's inequality for convex $f(.)$ is $\mathrm{E}(f(x)) \leq f(\mathrm{E}(x))$

[2] 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

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

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*

$$f\left(w_{\text{GD}}^{(T)}\right) - f(w^*) \leq \frac{B\rho}{\sqrt{T}} \tag{4.3}$$

(2) *a given level off accuracy ε such that $f\left(w_{GD}^{(T)}\right) - f(w^*) \leq \varepsilon$ 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 ρ 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[3]

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

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

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

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

[3]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`

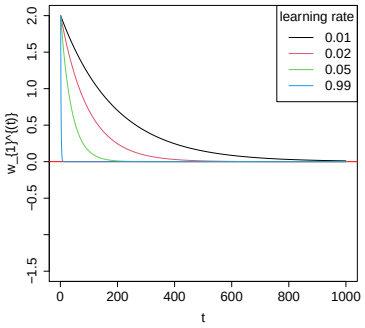
To learn the optimal $w^* = (w_1^*, w_2^*)$, we consider a loss $\ell(w) = (y_i - [w_1 + w_2 x_i])^2$, which leads to the minimization problem

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

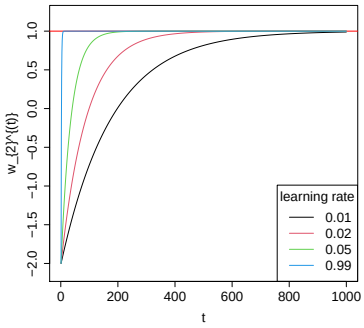
Consider data size $n = 100$, and parameters $\rho = 0.2$, $\sigma_y^2 = 1$ and $\sigma_x^2 = 1$. Consider a GD seed $w_0 = (2, -2)$.

Figures 5.1a, 5.1b, and 5.1c present trace plots of the chain $\left\{ \left(w^{(t)} \right) \right\}$ and error $\hat{R}_{\mathcal{D}} \left(w^{(t)} \right) - \hat{R}_{\mathcal{D}} (w^*)$ 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.

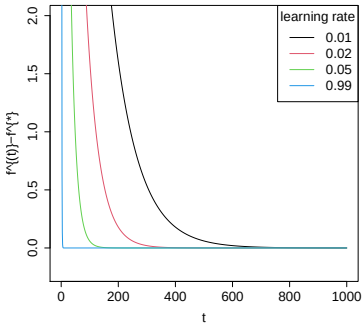
Figures 5.1d, 5.1e, and 5.1f present trace plots of the chain $\left\{ \left(w^{(t)} \right) \right\}$, and of the error $\hat{R}_{\mathcal{D}} \left(w^{(t)} \right) - \hat{R}_{\mathcal{D}} (w^*)$ produced by running GD for $T = 1000$ total iterations and for learning rate $\eta = 1.0$ (previously considered) 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.



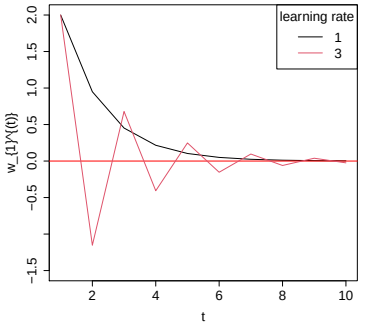
(A) $\left\{\left(w_1^{(t)}\right)\right\}$



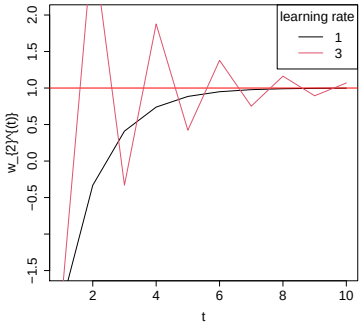
(B) $\left\{\left(w_2^{(t)}\right)\right\}$



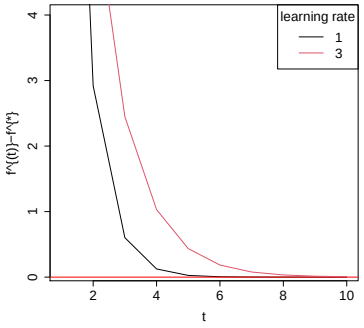
(C) $\hat{R}_{\mathcal{D}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{D}}\left(w^*\right)$



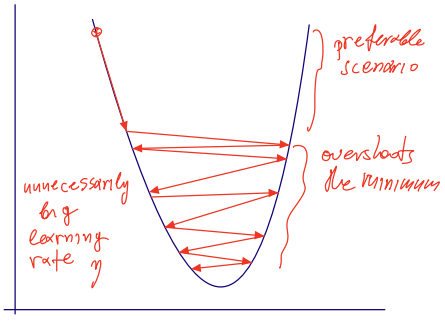
(D) $\left\{\left(w_1^{(t)}\right)\right\}$



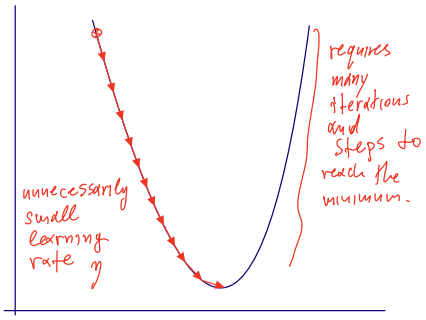
(E) $\left\{\left(w_2^{(t)}\right)\right\}$



(F) $\hat{R}_{\mathcal{D}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{D}}\left(w^*\right)$



(A) Unnecessarily large learning rate



(B) Unnecessarily small learning rate