Machine Learning and Neural Networks III (MATH3431)

Epiphany term

Georgios P. Karagiannis

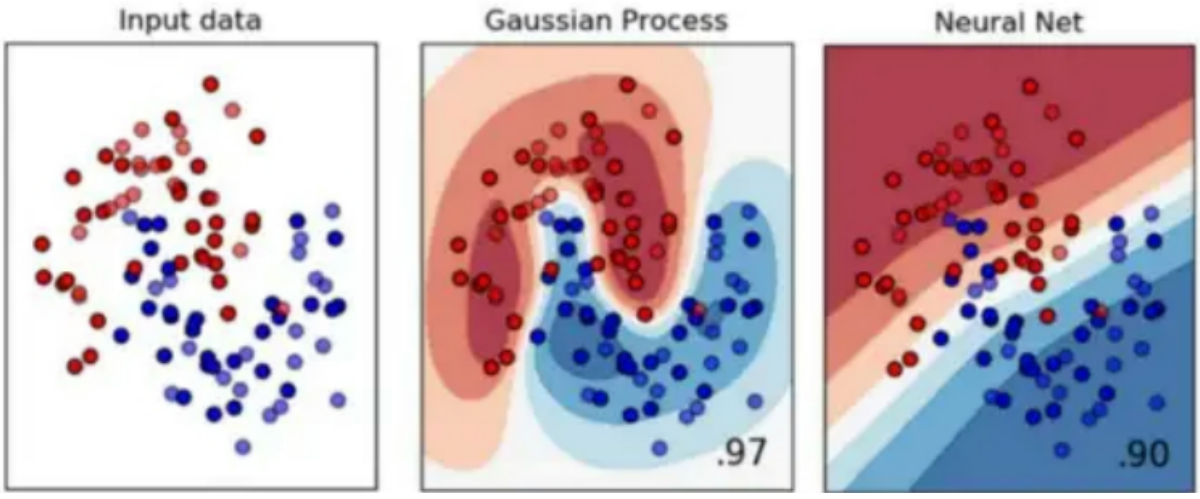
georgios.karagiannis@durham.ac.uk

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

2025/01/29 at 16:15:05

Concepts

- Convex learning problems
- Stochastic learning
- Support vector machines
- Artificial neural networks
- Kernel methods, trick
- Gaussian process regression



Reading list

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

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-recognition-machine-learning/`
- 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 It is very comprehensive (compared to others) and it goes deep enough for the module although it may be a bit outdated.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:

- Vapnik, V. (1999). The nature of statistical learning theory. Springer science & business media.
  - Important textbook in the statistical machine learning theory. To have have an in deep understanding about statistical learning, one has to read it together with the textbook of Shalev-Shwartz, S., & Ben-David, S. (2014)
- Devroye, L., Györfi, L., & Lugosi, G. (2013). A probabilistic theory of pattern recognition (Vol. 31). Springer Science & Business Media.
  - Important textbook in the theoretical aspects about machine learning algorithms. The level on difficulty is advanced as it requires moderate knowledge of probability.
- Theodoridis, S. (2015). Machine learning: a Bayesian and optimization perspective. Academic press.
  - A textbook in general machine learning methods. It includes a large collection of different machine learning methods. It does not go too deep into the theoretical details of the method.
- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
  - A popular textbook in general 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 general 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.
- Bishop, C. M., & Bishop, H. (2024). Deep learning: foundations and concepts. New York: Springer.
  - A textbook in machine learning methods focusing on Neural Networks philosophy and deep learning. Essentially, this is an update of the earlier book "Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press." where: the concepts about Neural networks, stochastic gradient descent have been extended; new deep learning concepts have been added; and the kernel methods are left out.
- 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.

Preparatory textbooks:

- Strang, G. (2019). Linear algebra and learning from data. Wellesley-Cambridge Press.
  - Material regarding linear algebra concepts.
- Boyd, S. P., & Vandenberghe, L. (2004). Convex optimization. Cambridge university press.
  - Material regarding Convex optimization, quadratic optimization, Lagrange duality.

Contents

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

Lecture notes 1: Machine learning -A recap on: definitions, notation, and formalism

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.
  - Ch. 1 Introduction
- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
  - Ch. 1 Introduction

1. General Introductions and 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, fitting) 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 will focus on this here) and the unsupervised learning.

Supervised learning problems involve applications where the training data $z \in \mathcal{Z}$ comprise 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 discover 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 (or help us make decisions about) $y \in \mathcal{Y}$ 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. Notation & definitions in learning

Note 1. The aim is to learn the mapping $x \to y$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with purpose to predict $y \in \mathcal{Y}$ from $x \in \mathcal{X}$. Denote $z = (x, y)^\top$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 2. 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 3. We often denote the set of hypothesis as $\mathcal{H}$; i.e. $h \in \mathcal{H}$.

Example 4. (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 $y \in \mathcal{Y} \subseteq \mathbb{R}$. A hypothesis is a linear function $h : \mathcal{X} \to \mathcal{Y}$ (that learner wishes to learn) with $h(x) = \langle w, x \rangle$ approximating the mapping $x \to y$. The hypothesis set is $\mathcal{H} = \left\{ x \to \langle w, x \rangle : w \in \mathbb{R}^d \right\}$.

Example 5. (Binary Classification) Consider the classification problem where the goal is to learn the mapping $x \to y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} = \{-1, +1\}$. A hypothesis can be a function $h : \mathcal{X} \to \mathcal{Y}$ with $h(x) = \operatorname{sign}(\langle w, x \rangle)$ approximating the mapping $x \to y$. The hypothesis set $\mathcal{H} = \left\{ x \to \operatorname{sign}(\langle w, x \rangle) : w \in \mathbb{R}^d \right\}$.

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 $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$. Loss function $\ell(h, z)$ for $h \in \mathcal{H}$ and $z \in \mathcal{Z}$ is specified according to the purpose the machine learning algorithm. It reflects how the "error" is quantified for a given hypothesis h and a given example z. The rule is "the greater the error the greater the value of the loss".

Example 7. (Cont. Example 4) In regression problems $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y} \subset \mathbb{R}$ is uncountable, a potential loss function is

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

Example 8. (Cont. Example 5) In binary classification problems with hypothesis $h : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y} = \{0, 1\}$ is discrete, a loss function can be

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

Definition 9. 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 10. The standard multiple linear regression problem with regressors $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and response $y \in \mathcal{Y} \subseteq \mathbb{R}$, is a learning problem with examples domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} = \mathbb{R}^{d+1}$, hypothesis class $\mathcal{H} = \left\{ x \to \langle w, x \rangle : w \in \mathbb{R}^d \right\}$, and loss function $\ell_{\text{sq}}(h, (x, y)) = (h(x) - y)^2$.

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

Example 11. The binary classification regression problem with regressors $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and response $y \in \mathcal{Y} = \{-1, +1\}$, is a learning problem with examples domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, hypothesis class $\mathcal{H} = \left\{x \to \text{sign}\left(\langle w, x\rangle\right) : w \in \mathbb{R}^d\right\}$, and loss function $\ell_{0-1}\left(h, (x, y)\right) = 1\left(h(x) \neq y\right) = 1\left(\langle w, x\rangle\, y < 0\right)$.

Definition 12. Data generation model $g\left(\cdot\right)$ is the probability distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, unknown to the learner which describes the probabilistic law that realizations $z = (x, y) \in \mathcal{Z}$ are realized.

Conventions...

Note 13. If the Hypothesis set $\mathcal{H}$ is a known parametric family of functions; i.e. $\mathcal{H} = \{h_w\left(\cdot\right); w \in \mathcal{W}\}$ parameterized by unknown $w \in \mathcal{W}$, then we can equivalently consider the convention $\mathcal{H} = \{w \in \mathcal{W}\} = \mathcal{W}$ keeping in mind that the learner's output is restricted to formula $h_w\left(\cdot\right)$.

Example 14. Because it involves only linear functions as predictors $h_w\left(x\right) = \langle w, x\rangle$, we could consider a hypothesis class $\mathcal{H} = \left\{w \in \mathbb{R}^d\right\} = \mathbb{R}^d$ and loss function loss $\ell\left(w, (x, y)\right) = \left(\langle w, x\rangle - y\right)^2$ for simplicity.

Example 15. Because it involves only linear functions as predictors $h_w\left(x\right) = \text{sign}\left(\langle w, x\rangle\right)$, we could consider a hypothesis class $\mathcal{H} = \left\{w \in \mathbb{R}^d\right\} = \mathbb{R}^d$ and loss function loss $\ell\left(w, (x, y)\right) = 1\left(\langle x, w\rangle\, y < 0\right)$ for simplicity.

2.1. **Learning...**

Definition 16. (Risk function) The risk function $R_g\left(h\right)$ of h is the expected loss of the hypothesis $h \in \mathcal{H}$, w.r.t. the data generation model (which is a probability distribution) g over domain Z; i.e.

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

Definition 17. (Risk minimization learning) The Risk minimization (RM) learning paradigm computes the optimal predictor h^* as the minimizer of the risk $R_g\left(h\right)$ of h; i.e.

$$h^* = \arg\min_h\left(R_g\left(h\right)\right) \tag{2.2}$$

Example 18. (Cont. Ex. 7) The risk function is $R_g\left(h_w\right) = \text{E}_{z\sim g}\left(h_w(x) - y\right)^2 = \text{E}_{z\sim g}\left(\langle w, x\rangle - y\right)^2$, 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.

Example 19. (Cont. Ex.) The risk function is $R_g\left(h_w\right) = \text{E}_{z\sim g}\left(1\left(h_w(x) \neq y\right)\right) = \text{Pr}_{z\sim g}\left(\langle x, w\rangle\, y < 0\right)$.

Note 20. 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, instead of the Risk function in (2.2), one use the Empirical risk function as a Monte Carlo approximation of it.

Definition 21. Training data set $\mathcal{S}$ of size m is any finite sequence of pairs $\left(z_i = (x_i, y_i)\,; i = 1, ..., m\right)$, called examples, in $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$; i.e. $\mathcal{S} = \{(x_i, y_i)\,; i = 1, ..., m\}$. This is the information that the learner has assess.

Definition 22. (Empirical risk function) The Empirical Risk Function (ERF) $\hat{R}_S(h)$ of h is the expectation of loss of h over a given sample $S = (z_1, ..., z_m) \in \mathcal{Z}^m$; i.e.

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

Definition 23. (Empirical risk minimization learning) The Empirical risk minimization (ERM) learning paradigm computes the optimal predictor h^* as the minimizer of the ERF $\hat{R}_S(h)$ of h; i.e.

$$h^* = \arg\min_h \left(\hat{R}_S(h) \right) \tag{2.3}$$

Example 24. (Cont. Example 18) Given given sample $S = \{(x_i, y_i); i = 1, ..., m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 = \frac{1}{m} \sum_{i=1}^{m} \left(x_i^\top w - y_i \right)^2$.

Example 25. (Cont. Example 8) Given given sample $S = \{(x_i, y_i); i = 1, ..., m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1(h(x_i) \neq y_i) = \frac{1}{m} \sum_{i=1}^{m} 1(\langle x_i, w \rangle y_i < 0)$.

Definition 26. We denote as $\mathfrak{A}(\mathcal{S})$ the hypothesis (outcome) that a learning algorithm $\mathfrak{A}$ returns given training sample S.

3. Regularized loss minimization (RLM) learning

Note 27. Consider a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with training data set $\mathcal{S}$.

Definition 28. (Regularized loss minimization learning) Regularized loss minimization learning paradigm computes the optimal rule $h^* \in \mathcal{H}$ by jointly minimizing the Risk function $R_g(h)$ (or Empirial Risk Function $\hat{R}_{\mathcal{S}}(h)$) and a regularization function $J : \mathcal{H} \to \mathbb{R}$ where $J(h)$ increases in value with the complexity of the hypothesis $h \in \mathcal{H}$. Formally, the Regularized loss minimization rule h^* is

$$h^* = \arg\min_{h \in \mathcal{H}} \left(R_g(h) + J(h) \right), \text{ given the Risk function}$$

$$h^* = \arg\min_{h \in \mathcal{H}} \left(\hat{R}_{\mathcal{S}}(w) + J(h) \right), \text{ given the Empirical risk function}$$

3.1. ℓ_0, ℓ_1,and ℓ_2 norm regularization problems.

Note 29. Consider a parameterized hypothesis $h_w(\cdot)$ (e.g. $h_w(x) = \eta(\langle x, w \rangle)$) with $\eta(\cdot)$ a function and $w \in \mathbb{R}^d$. Consider the hypothesis class in the simplified form $\mathcal{H} = \mathcal{W} = \left\{ w \in \mathbb{R}^d \right\} = \mathbb{R}^d$ of Note 13. Assume we want the vector w to be sparse, in the sense that "sparser" means more zero elements.

3.1.1. ℓ_0 *norm regularization.*

Note 30. The problem of minimizing the empirical risk subject to a budget of k features can be written as

$$\underset{w \in \mathcal{H}}{\text{minimize}} R(w)$$

$$\text{subject to } \|w\|_0 \leq k_0$$

where $\|w\|_0 = \#\{j : w_j \neq 0\}$. By Lagrange multiplies we can re-write it as

$$w^* = \arg\min_{w \in \mathcal{H}} \left(R(w) + \lambda_0 \|w\|_0 \right)$$

for some $\lambda_0 \geq 0$ (k_0 dependent). Hence, $h^*(x) = h_{w^*}(x)$.

Example 31. (Cont. Example 14) The ℓ_0-RLM rule in Example 14 becomes

$$w^* = \arg\min_{w} \left(\mathrm{E}_{z \sim g} \left(\langle w, x_i \rangle - y \right)^2 + \lambda_0 \|w\|_0 \right)$$

using the Risk function, and

$$w^* = \arg\min_{w} \left(\frac{1}{m} \sum_{i=1}^{m} \left(\langle w, x_i \rangle - y_i \right)^2 + \lambda_0 \|w\|_0 \right)$$

using the Empirical risk function.

3.1.2. *ℓ_1 norm regularization (LASSO).*

Note 32. Solving ℓ_0 norm optimization problem is computationally hard. To simplify computations, we can replace $J(\cdot) = \|\cdot\|_0$ with $J(\cdot) = \|\cdot\|_1$ i.e.

$$\begin{aligned} &\underset{w \in \mathcal{H}}{\text{minimize}} R(w) \\ &\text{subject to } \|w\|_1 \leq k_1 \end{aligned}$$

where $\|w\|_1 = \sum_j |w_j|$. By Lagrange multiplies we can re-write it as

$$w^* = \arg\min_{w} \left(R(w) + \lambda_1 \|w\|_1 \right)$$

for some $\lambda_1 \geq 0$ (k_1 dependent). Hence, $h^*(x) = h_{w^*}(x)$.

Example 33. (Cont. Example 14) The ℓ_1-RLM rule in Example 14 becomes

$$w^* = \arg\min_{w} \left(\mathrm{E}_{z \sim g} \left(\langle w, x \rangle - y \right)^2 + \lambda_1 \sum_{j=1}^{d} |w_j| \right)$$

using the Risk function , and

$$w^* = \arg\min_{w} \left(\frac{1}{m} \sum_{i=1}^{m} \left(\langle w, x_i \rangle - y_i \right)^2 + \lambda_1 \sum_{j=1}^{d} |w_j| \right)$$

using the Empirical risk function.

3.1.3. *ℓ_2 norm regularization (Ridge).*

Note 34. To further simplify computations, we can consider $J(\cdot) = \|\cdot\|_2$ i.e.

$$\begin{aligned} &\underset{w}{\text{minimize}} R(w) \\ &\text{subject to } \|w\|_2^2 \leq k_2 \end{aligned}$$

where $\|w\|_2^2 = \sum_j w_j^2$. By Lagrange multiplies we can re-write it as

$$w^* = \arg\min_w \left(R(w) + \lambda_2 \|w\|_2^2 \right)$$

for some $\lambda_1 \geq 0$ (k_1 dependent). Hence, $h^*(x) = h_{w^*}(x)$.

Example 35. (Cont. Example 14) The ℓ_2-RLM rule in Example 14 becomes

$$w^* = \arg\min_w \left(\mathrm{E}_{z\sim g} \left(\langle w, x \rangle - y \right)^2 + \lambda_2 \sum_{j=1}^{d} w_j^2 \right)$$

using the Risk function, and

$$w^* = \arg\min_w \left(\frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2 + \lambda_2 \sum_{j=1}^{d} w_j^2 \right)$$

using the Empirical risk function.

Example 36. (Cont. Example 15) The ℓ_2-RLM rule in Example 15 becomes

$$w^* = \arg\min_w \left(\Pr_{z\sim g} \left(\langle x, w \rangle \, y < 0 \right) + \lambda_2 \sum_{j=1}^{d} w_j^2 \right)$$

using the Risk function, and

$$w^* = \arg\min_w \left(\frac{1}{m} \sum_{i=1}^{m} 1 \left(\langle x_i, w \rangle \, y_i < 0 \right) + \lambda_2 \sum_{j=1}^{d} w_j^2 \right)$$

using the Empirical risk function.

Appendix A. Further examples

Example 37. 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

$$R_g(\theta) = \mathrm{E}_{z\sim g}\left(\log\left(\frac{g(z)}{f_\theta(z)}\right)\right) = \mathrm{E}_{z\sim g}(\log(g(z))) - \mathrm{E}_{z\sim g}(\log(f_\theta(z))) \tag{A.1}$$

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 (A.1) is constant. Note that in the Maximum Likelihood Estimation technique the MLE θ_{MLE} is the minimizer

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

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 $R_S(\theta) = \frac{1}{m}\sum_{i=1}^{m}(-\log(f_\theta(z_i)))$.

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

Lecture notes 2: Elements of convex learning problems

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

Aim. To introduce convex learning problems that can be used as a framework for the analysis of stochastic gradient related learning algorithms.

Reading list & references:

- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
  - Ch. 12 Convex Learning Problems

Further reading

- Bishop, C. M. (2006). Pattern recognition and machine learning. New York: Springer.

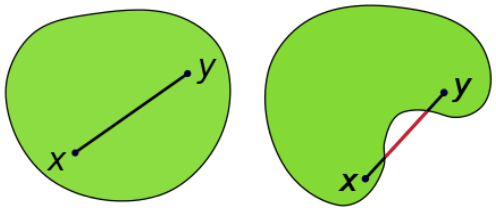
1. Motivations

Note 1. We introduce learning problems associated with convexity, Lipschitzness and smoothness to be able to analyze and understand the machine learning (ML) tools we will discuss later on (eg stochastic gradient descent, SVM). We discuss ways allowing to address non-convex ML problems (eg, Artificial neural networks, Gaussian process regression) in the convex setting.

2. Convexity

Note 2. Convexity is a central concept in learning, e.g. least squares, as it often considers a Euclidean distance as a Risk function to be minimized.

Definition 3. A set C is convex if for any $u, v \in C$ and for any $\alpha \in [0,1]$ we have that $\alpha u + (1-\alpha)\, v \in C$.



(A) (B)

FIGURE 2.1. (2.1a) is a Convex set ; (2.1b) is a non-convex set

Example 4. For instance $\mathbb{R}^d$ for $d \geq 1$ is a convex set.

Definition 5. 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 $\alpha \in [0, 1]$

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

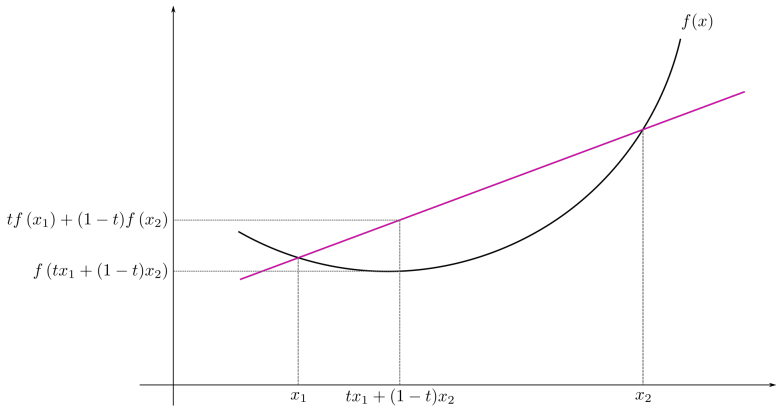


FIGURE 2.2. A convex function

Example 6. The function $f : \mathbb{R}^d \to \mathbb{R}_+$ with $f(x) = \|x\|_2^2$ is convex function. For any $u, v \in C$ and for any $\alpha \in [0, 1]$ it is

$$\begin{aligned}
&\|\alpha u + (1-\alpha) v\|_2^2 - \alpha \|u\|_2^2 - (1-\alpha) \|v\|_2^2 \\
&= (\alpha u + (1-\alpha) v)^\top (\alpha u + (1-\alpha) v) - \alpha \|u\|_2^2 - (1-\alpha) \|v\|_2^2 \\
&= \ldots = -\alpha (1-\alpha) \|u - v\|_2^2 \leq 0
\end{aligned}$$

Note 7. Every local minimum of a convex function is the global minimum.

Note 8. 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 9. *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 10. Consider the regression problem with regressor $x \in \mathbb{R}^d$, and response $y \in \mathbb{R}$ and predictor rule $h(x) = \langle w, x\rangle$. The loss function $\ell(w, (x, y)) = (\langle w, x\rangle - y)^2$ is convex because $g(a) = (a)^2$ is convex and Proposition 9.

Note 11. 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 $w_j > 0$

Proof.

(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 12. $g(x) = \|x\|_1$ is convex according to Note 11, since $|x_j| = \max(-x_j, x_j)$ and $\|x\|_1 = \sum_j |x_j|$.

3. Strong convexity

Note 13. Strong convexity is a central concept in Ridge regularization, as it makes a convex loss function strongly convex by adding a shrinkage term.

Definition 14. (Strongly convex functions) A function f is λ-strongly convex function is for all w, u, and $\alpha \in (0,1)$ we have

$$f(\alpha w + (1-\alpha)u) \leq \alpha f(w) + (1-\alpha) f(u) - \frac{\lambda}{2}\alpha(1-\alpha)\|w-u\|^2 \tag{3.1}$$

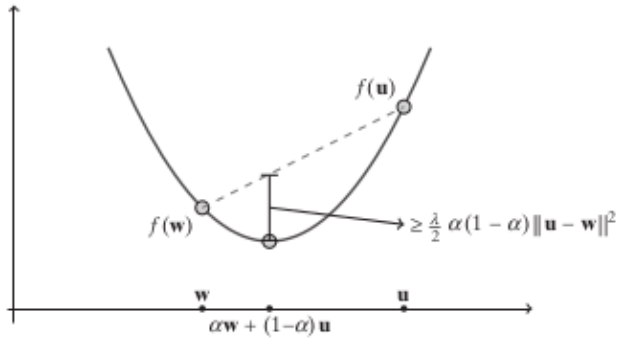


FIGURE 3.1. Strongly convex function

Note 15. The following can be checked from the definition by substitution

(1) The function $f(w) = \lambda\|w\|^2$ is 2λ-strongly convex
(2) If $f$is λ-strongly convex and g is convex then $f+g$ is λ-strongly convex

4. Lipschitzness

Definition 16. Let $C \subseteq \mathbb{R}^d$. Function $f : C \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $w_1, w_2 \in C$ we have that

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

Note 17. That means: a Lipschitz function $f(x)$ cannot change too drastically wrt x.

Example 18. 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 $C = \{x \in \mathbb{R} : |x| < \rho/2\}$.

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

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

Note 19. 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.

[See Exercise 2 from the exercise sheet]

Example 20. Let functions g be ρ-Lipschitz. Then f with $f(x) = g(< v, x > +b)$ is $(\rho|v|)$-Lipschitz.

Solution. It is

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

Note 21. So, given Examples 18 and 20, in the linear regression setting using loss $\ell(w, z = (x, y)) = \left(w^\top x - y\right)^2$, the loss function is -Lipschitz for a given $z = (x, y)$ and and bounded $\|w\| < \rho$.

5. Smoothness

Definition 22. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz; namely for all $v, w \in \mathbb{R}^d$

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

Note 23. Function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth iff

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

Note 24. 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.

[See Exercise 3 from the Exercise sheet]

Example 25. 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.

To show that we set $f(w) = g(< w, x > +y)$ where $g(a) = a^2$. g is 2-smooth since

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

Hence from Theorem 24, f is $\left(2\|x\|^2\right)$-smooth.

Example 26. Consider the regression problem with predictor rule $h(x) = \langle w, x \rangle$, loss function $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$, feature $x \in \mathbb{R}^d$, and target $y \in \mathbb{R}$. Then $\ell(w, \cdot)$ is $\left(2\|x\|^2\right)$-smooth.

[Follows from Example 25.]

6. Convex learning problems

Definition 27. 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 $\ell(\cdot, z)$ is a convex function for each example $z \in \mathcal{Z}$.

Example 28. Consider the regression problem with predictor rule $h(x) = \langle w, x \rangle$, loss function $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$, feature $x \in \mathbb{R}^d$, and target $y \in \mathbb{R}$. This imposes a convex learning problem due to Examples 4 and 11.

Definition 29. Convex-Lipschitz-Bounded Learning Problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with parameters ρ, and B, is called the learning problem whose the hypothesis class $\mathcal{H}$ is a convex set, for all $w \in \mathcal{H}$ it is $\|w\| \leq B$, and the loss function $\ell(\cdot, z)$ is convex and ρ-Lipschitz function for all $z \in \mathcal{Z}$.

Example 30. Consider the regression problem with predictor rule $h(x) = \langle w, x \rangle$, loss function $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$, feature $x \in \mathbb{R}^d$, and target $y \in \mathbb{R}$. This imposes a Convex-Lipschitz-Bounded Learning Problem if $\mathcal{H} = \left\{w \in \mathbb{R}^d : \|w\| \leq B\right\}$ due to Examples 11, and 18(2).

Definition 31. Convex-Smooth-Bounded Learning Problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with parameters β, and B, is called the learning problem whose the hypothesis class $\mathcal{H}$ is a convex set, for all $w \in \mathcal{H}$ it is $\|w\| \leq B$, and the loss function $\ell(\cdot, z)$ is convex, nonnegative, and β-smooth function for all $z \in \mathcal{Z}$.

Example 32. Consider the regression problem with predictor rule $h(x) = \langle w, x \rangle$, loss function $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$, feature $x \in \mathbb{R}^d$, and target $y \in \mathbb{R}$. This imposes a Convex-Smooth-Bounded Learning Problem if $\mathcal{H} = \left\{w \in \mathbb{R}^d : \|w\| \leq B\right\}$ due to Examples 11, and 26.

Note 33. (Empirical risk minimization learning problem) If ℓ is a convex loss function and the class $\mathcal{H}$ is convex, then the $\mathrm{ERM}_{\mathcal{H}}$ problem, of minimizing the empirical risk $\hat{R}_{\mathcal{S}}(w)$ over $\mathcal{H}$, is a convex optimization problem

Proof. The Empirical risk minimization (ERM) problem $(\mathcal{H}, \mathcal{Z}, \ell)$ is

$$w^* = \arg\min_{w \in \mathcal{H}} \left\{\hat{R}_{\mathcal{S}}(w)\right\}$$

given a sample $\mathcal{S} = \{z_1, ..., z_m\}$ for $\hat{R}_{\mathcal{S}}(w) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, z_i)$. $\hat{R}_{\mathcal{S}}(w)$ is a convex function from Note (11). Hence ERM rule is a problem of minimizing a convex function subject to the constraint that the solution should be in a convex set. □

Example 34. Multiple linear regression with predictor rule $h(x) = \langle w, x\rangle$, loss function $\ell(w,(x,y)) = (\langle w, x\rangle - y)^2$, feature $x \in \mathbb{R}^d$, and target $y \in \mathbb{R}$ where

$$w^* = \arg\min_w \mathrm{E}\,(\langle w, x\rangle - y)^2$$

or

$$w^{**} = \arg\min_w \frac{1}{m}\sum_{i=1}^{m}(\langle w, x_i\rangle - y_i)^2$$

is a convex learning problem –from Note 33.

Note 35. (Ridge / ℓ_2-RLM problem) If ℓ is a convex loss function w.r.t. w, the class $\mathcal{H}$ is convex, and $J(\cdot;\lambda) = \lambda\|\cdot\|_2^2$ with $\lambda > 0$ then $\hat{R}_{\mathcal{S}}(w) + \lambda\|w\|_2^2$ is a 2λ-strongly convex function, and hence the ℓ_2-RLM problem

$$w^* = \arg\min_{w\in\mathcal{H}} \left\{\hat{R}_{\mathcal{S}}(w) + \lambda\|w\|_2^2\right\}$$

is a strongly convex optimization problem (i.e. the learning rule is the minimizer of a strongly convex function over a convex set).

Proof. $\hat{R}_{\mathcal{S}}(\cdot)$ is a convex function from Note 33, $\lambda\|\cdot\|_2^2$ is 2λ-strongly convex, hence $\hat{R}_{\mathcal{S}}(w) + \lambda\|w\|_2^2$ is a 2λ-strongly convex function. Hence the above Ridge RLM problem is a strongly convex optimization problem. □

7. Non-convex learning problems (surrogate treatment)

Note 36. A learning problem may involve non-convex loss function $\ell(w,z)$ w.r.t. w which implies a non-convex risk function $R_g(w)$. A suitable treatment to analyze the associated learning tools within the convex setting 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 37. 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

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

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 11) wrt w. Note that:

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

Then we can compute

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

instead of

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

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

Note 38. (Intuitions...) Using a convex surrogate loss function instead the convex one, facilitates 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, 2025

Lecture notes 3: Learnability and stability in learning problems

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

Aim. To introduce concepts PAC, fitting vs stability trade off, stability, and their implementation in regularization problems and convex problems.

Reading list & references:

- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
  - Ch. 2, 3, 13
- Vapnik, V. (2000). The nature of statistical learning theory. Springer science & business media.

1. Learnable Probably Approximately Correct (PAC) Learning

Note 1. We formally define the broad learning problem we will work on.

Note 2. Learning algorithms $\mathfrak{A}$ use training data sets $\mathcal{S}$ which may miss characteristics of the unknown data-generating process g. Essentially, approximations (aka errors) are inevitable; some characteristics of data generating process g will be missed even if we use a very representative training set $\mathcal{S}$. We cannot hope that the learning algorithm will find a hypothesis whose error is smaller than the minimal possible error.

Note 3. The PAC learning problem requires no prior assumptions about the data-generating process g. It requires that the learning algorithm $\mathfrak{A}$ will find a predictor $\mathfrak{A}(\mathcal{S})$ whose error is not much larger than the best possible error of a predictor in some given benchmark hypothesis class. So essentially, in practice, the researcher's effort falls on the hypothesis class $\mathcal{H}$.

Definition 4. (Agnostic PAC Learnability for General Loss Functions) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a domain $\mathcal{Z}$ and a loss function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm $\mathfrak{A}$ with the following property:

- for every $\epsilon \in (0,1)$, $\delta \in (0,1)$, and distribution g over $\mathcal{Z}$, when running algorithm $\mathfrak{A}$ given training set $\mathcal{S}_m = \{z_1, ..., z_m\}$ with $z_i \overset{\text{IID}}{\sim} g$ for $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ then $\mathfrak{A}$ returns

$\mathfrak{A}(\mathcal{S}) \in \mathcal{H}$ such as

$$\Pr_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}_m\right)\right) - \min_{h'\in\mathcal{H}}\left(R_g\left(h'\right)\right) \leq \epsilon\right) \geq 1-\delta \tag{1.1}$$

Note 5. About (1.1): The accuracy parameter ϵ determines how far the output rule $\mathfrak{A}(\mathcal{S}_m)$ of $\mathfrak{A}$ can be from the optimal rule (this corresponds to the 'approximately correct' part of "PAC"). The confidence parameter δ indicates how likely the classifier is to meet that accuracy requirement (corresponds to the "probably" part of "PAC").

Note 6. It may be easier to work with expectations by using Markov inequality in (1.1).

Proposition 7. *Let $\mathfrak{A}$ be a learning algorithm that guarantees the following:*

- *If $m \geq m_{\mathcal{H}}(\epsilon)$ then for every distribution g, it is*

$$E_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}_m\right)\right)\right) \leq \min_{h'\in\mathcal{H}}\left(R_g\left(h'\right)\right) + \epsilon$$

Then $\mathfrak{A}$ satisfies the PAC guarantee in Definition 4:

- *for every $\delta \in (0,1)$, if $m \geq m_{\mathcal{H}}(\epsilon\delta)$* then

$$\Pr_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}_m\right)\right) - \min_{h'\in\mathcal{H}}\left(R_g\left(h'\right)\right) \leq \epsilon\right) \geq 1-\delta$$

Proof. Let $\xi = R_g\left(\mathfrak{A}\left(\mathcal{S}_m\right)\right) - \min_{h'\in\mathcal{H}}\left(R_g\left(h'\right)\right)$. From Markov's inequality $\Pr\left(\xi \geq \mathrm{E}\left(\xi\right)/\delta\right) \leq \frac{1}{\mathrm{E}(\xi)/\delta}\mathrm{E}\left(\xi\right) = \delta$. Namely, $\Pr\left(\xi \leq \mathrm{E}\left(\xi\right)/\delta\right) = 1-\delta$. But it is given that if $m \geq m_{\mathcal{H}}\left((\epsilon\delta)\right)$ for every distribution g it is $\mathrm{E}\left(\xi\right) \leq (\epsilon\delta)$. So by substitution $\Pr\left(\xi \leq (\epsilon\delta)/\delta\right) \geq 1-\delta$ implies $\Pr\left(\xi \leq \epsilon\right) \geq 1-\delta$. Now substitute back ξ and we conclude the proof. □

2. Analysis of the risk based on the trade-off fitting vs stability

Note 8. Let $R^* = \min_{\forall h}\left(R\left(h\right)\right)$ be an ideal/optimal (hence minimum) Risk, and $\mathfrak{A}(\mathcal{S})$ the learning rule from a learning algorithm $\mathfrak{A}$ trained against dataset $\mathcal{S}$. The Risk of a learning algorithm $\mathfrak{A}$ can be decomposed as

$$R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right) - R^* = \hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right) - R^* + \underbrace{R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right) - \hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)}_{\text{over-fitting}} \tag{2.1}$$

Note 9. Over-fitting can be represented by $R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right) - \hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)$. However, $\hat{R}_{\mathcal{S}}\left(\cdot\right)$ is a random variable and, here, for our computational convenience, we focus on its expectation w.r.t. $\mathcal{S} \sim g$. Hence, we provide the following (arguable) definition for over-fitting on which we base our analysis.

Definition 10. For a learning algorithm $\mathfrak{A}$, as a measure of over-fitting we consider the expected difference between true Risk and empirical Risk

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right) - \hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right) \tag{2.2}$$

Definition 11. We say that learning algorithm $\mathfrak{A}$ suffers from over-fitting when (2.2) is 'too' large.

Note 12. The expected Risk of a learning algorithm $\mathfrak{A}(\mathcal{S})$ can be decomposed as

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}(\mathcal{S})\right)\right) = \underbrace{\mathrm{E}_{\mathcal{S}\sim g}\left(\hat{R}_{\mathcal{S}}\left(\mathfrak{A}(\mathcal{S})\right)\right)}_{\text{(I)}} + \underbrace{\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}(\mathcal{S})\right) - \hat{R}_{\mathcal{S}}\left(\mathfrak{A}(\mathcal{S})\right)\right)}_{\text{(II)}} \tag{2.3}$$

by applying expectations in (2.1) and ignoring R^*. (I) indicates how well $\mathfrak{A}(\mathcal{S})$ fits the training set $\mathcal{S}$, and (II) indicates the discrepancy between the true and empirical risks of $\mathfrak{A}(\mathcal{S})$. In Section 3, we argue that the over-fitting term (II) is directly related to a certain type of stability of $\mathfrak{A}(\mathcal{S})$.

Note 13. The following result connects the need for upper bounding (and minimizing this bound) the Expected Risk in (2.3) and PAC learning (Definition 4).

Note 14. Hence, we aim to design a learning algorithm $\mathfrak{A}(\mathcal{S})$ that both fits the training set and is stable; i.e, keep both (I) and (II) in (2.3) small. As seen later, in certain learning problems, there may be a trade-off between empirical risk term (I) and (II) in (2.3). Consequently, we aim to upper bound (2.3) by upper bounding (I) and (II) individually and decide which term we should benefit against the other to achieve a certain total bound.

3. Stability and its association with over-fitting

Notation 15. Consider a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$. Let $\mathcal{S} = \{z_1, ..., z_m\}$ be a training sample, and $\mathfrak{A}$ be a learning algorithm with output $\mathfrak{A}(\mathcal{S})$.

Note 16. It is reasonable to consider that a learning algorithm $\mathfrak{A}$ can be stable if a small change of the algorithm input does not change the algorithm output much. Mathematically formalizing this, we can say that if $\mathcal{S}^{(i)} = \{z_1, ..., z_{i-1}, z', z_{i+1}, ..., z_m\}$ is another training dataset equal to $\mathcal{S}$ but the ith element being replaced by another independent example $z' \sim g$, then a good learning algorithm $\mathfrak{A}$ would produce a small value of

$$\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell\left(\mathfrak{A}(\mathcal{S}), z_i\right) \geq 0$$

Definition 17. A learning algorithm $\mathfrak{A}$ is **on-average-replace-one-stable** with rate a decreasing function $\epsilon : \mathbb{N} \to \mathbb{R}$ if for every distribution g

$$\mathrm{E}_{\substack{\mathcal{S}\sim g \\ z'\sim g, i\sim \mathrm{U}\{1,...,m\}}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell\left(\mathfrak{A}(\mathcal{S}), z_i\right)\right) \leq \epsilon(m) \tag{3.1}$$

Note 18. The following result demonstrates the direct association of stability and over-fitting as defined in Definitions 11 & 17.

Theorem 19. *For any learning algorithm* $\mathfrak{A}$ *it is*

(3.2)

$$E_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right)-\hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right)=E_{\substack{\mathcal{S}\sim g,\ z'\sim g\\ i\sim U\{1,...,m\}}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right),z_i\right)-\ell\left(\mathfrak{A}\left(\mathcal{S}\right),z_i\right)\right)$$

where g *is a distribution,* $\mathcal{S}=\{z_1,...,z_m\}$ *and* $\mathcal{S}^{(i)}=\{z_1,...,z_{i-1},z',z_{i+1},...,z_m\}$ are *training datasets with* $z',z_1,...,z_m\overset{\text{iid}}{\sim}g$.

Proof. As $z',z_1,...,z_m\overset{\text{iid}}{\sim}g$, then for every i

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right)=\mathrm{E}_{\substack{\mathcal{S}\sim g\\ z'\sim g}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}\right),z'\right)\right)=\mathrm{E}_{\substack{\mathcal{S}\sim g\\ z'\sim g}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right),z_i\right)\right)$$

and

$$\mathrm{E}_{\mathcal{S}\sim g}\left(\hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right)=\mathrm{E}_{\substack{\mathcal{S}\sim g\\ i\sim U\{1,...,m\}}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right),z_i\right)\right)$$

□

Note 20. Hence, we desire to design learning algorithms corresponding to as small as possible (3.2).

4. Implementation in regularized loss learning problems

Definition 21. A Regularized Loss Minimization (RLM) learning problem $(\mathcal{H},\mathcal{Z},\ell)$ with a regularization function $J:\mathcal{H}\to\mathbb{R}$ aims at a RLM learning rule that is the minimizer

$$h^*=\arg\min_{h\in\mathcal{H}}\left(\hat{R}_{\mathcal{S}}\left(h\right)+J\left(h\right)\right) \tag{4.1}$$

where $\hat{R}_{\mathcal{S}}\left(\cdot\right)=\frac{1}{m}\sum_{i=1}^{m}\ell\left(\cdot,z_i\right)$, and $\mathcal{S}=\{z_1,...,z_m\}\subset$ an IID training sample.

Remark 22. The motivation for considering the regularization function J in (4.1) is to: (1.) control complexity and (2.) improve stability; as we will see later.

Note 23. Here, we make our example more specific and narrow it to the Ridge RLM learning problem (could have been LASSO etc.).

Definition 24. The Ridge RLM learning problem $(\mathcal{H},\mathcal{Z},\ell)$, with $\mathcal{H}=\mathcal{W}\subset\mathbb{R}^d$, uses regularization function $J\left(w;\lambda\right)=\lambda\left\|w\right\|_2^2$ with $\lambda>0$, $w\in\mathcal{W}$ and produces learning rule

$$\mathfrak{A}\left(\mathcal{S}\right)=\arg\min_{w\in\mathcal{W}}\left(\hat{R}_{\mathcal{S}}\left(w\right)+\lambda\left\|w\right\|_2^2\right) \tag{4.2}$$

Note 25. Recall (Term 1) how the regularization function in Ridge RLM learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ penalizes complexity. Essentially, it implies a sequence of hypothesis $\mathcal{H}_1 \subset \mathcal{H}_2 \subset ...$ with $\mathcal{H}_i = \left\{w \in \mathbb{R}^d : \|w\|_2 < i\right\}$ due to duality of the corresponding minimization problem.

Note 26. Below, we will try to analyze the behavior of Ridge RLM learning rule (4.2) w.r.t. the Risk decomposition (2.3). In particular, to upper bounded w.r.t. the shrinkage term λ, training sample size m, and other characteristics.

4.1. **Bounding the empirical risk (I) in (2.3).**

Note 27. From (4.2), we have

$$\begin{aligned}\hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right) &\leq \hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right) + \lambda \left\|\mathfrak{A}\left(\mathcal{S}\right)\right\|_2^2 \\ &\leq \hat{R}_{\mathcal{S}}\left(w\right) + \lambda \left\|w\right\|_2^2; \quad \forall w \in \mathcal{W}\end{aligned}$$

and by taking expectations w.r.t. $\mathcal{S}$, it is

$$\mathrm{E}_{\mathcal{S}\sim g}\left(\hat{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right) \leq R_g\left(w\right) + \lambda \left\|w\right\|_2^2; \quad \forall w \in \mathcal{W} \tag{4.3}$$

because $\mathrm{E}_{\mathcal{S}\sim g}\left(\hat{R}_{\mathcal{S}}\left(\cdot\right)\right) = \frac{1}{m}\sum_{i=1}^m \mathrm{E}_{\mathcal{S}\sim g}\left(\ell\left(\cdot, z_i\right)\right) = R_g\left(\cdot\right)$.

Note 28. From (4.3), we observe that part (I) in the expected risk decomposition (2.3), (aka the upper bound of the expected empirical risk) increases with the regularization term $\lambda > 0$. (!!!) –Although anticipated, it did not start well.

4.2. **Bounding the empirical risk (II) in (2.3).**

Note 29. As we will see to bound (II) in (2.3) we will have to impose an additional condition on the behavior of loss function apart from convexity; e.g. Lipstchitzness.

Assumption 30. *The loss function $\ell\left(\cdot, z\right)$ in (4.2) is convex for any $z \in \mathcal{Z}$.*

Note 31. Let $\tilde{R}_{\mathcal{S}}\left(w\right) = \hat{R}_{\mathcal{S}}\left(w\right) + \lambda\left\|w\right\|_2^2$. $\tilde{R}_{\mathcal{S}}\left(\cdot\right)$ is 2λ-strongly convex as the sum of a convex function $\hat{R}_{\mathcal{S}}\left(\cdot\right)$ (Assumption 30) and a 2λ-strongly convex function $J\left(\cdot;\lambda\right) = \lambda\left\|\cdot\right\|_2^2$ (results directly from Definition 14 in Lecture notes 2).

Fact 32. *(To be used in Note 33) If f is* λ-strongly convex and u is a minimizer of f then for any w

$$f\left(w\right) - f\left(u\right) \geq \frac{\lambda}{2}\left\|w - u\right\|^2$$

Note 33. Let $\mathfrak{A}\left(\mathcal{S}\right)$ be the Ridge learning algorithm output minimizing (4.2). Because $\tilde{R}_{\mathcal{S}}\left(\cdot\right)$ is 2λ-strongly convex and $\mathfrak{A}\left(\mathcal{S}\right)$ is its minimizer, according to Fact 32, for $\mathfrak{A}\left(\mathcal{S}\right)$ and any $w \in \mathcal{W}$, it is

$$\tilde{R}_{\mathcal{S}}\left(w\right) - \tilde{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}\right)\right) \geq \lambda\left\|w - \mathfrak{A}\left(\mathcal{S}\right)\right\|^2, \quad \forall w \in \mathcal{W} \tag{4.4}$$

Note 34. Also, for any $w, u \in \mathcal{W}$, it is

$$\begin{aligned}\tilde{R}_{\mathcal{S}}(w) - \tilde{R}_{\mathcal{S}}(u) &= \left(\hat{R}_{\mathcal{S}}(w) + \lambda \|w\|_2^2\right) - \left(\hat{R}_{\mathcal{S}}(u) + \lambda \|u\|_2^2\right) \\ &= \left(\hat{R}_{\mathcal{S}^{(i)}}(w) + \lambda \|w\|_2^2\right) - \left(\hat{R}_{\mathcal{S}^{(i)}}(u) + \lambda \|u\|_2^2\right) \\ &\quad + \frac{\ell(w, z_i) - \ell(u, z_i)}{m} + \frac{\ell(u, z') - \ell(w, z')}{m} \\ &= \tilde{R}_{\mathcal{S}^{(i)}}(w) - \tilde{R}_{\mathcal{S}^{(i)}}(u) + \frac{\ell(w, z_i) - \ell(u, z_i)}{m} + \frac{\ell(u, z') - \ell(w, z')}{m}\end{aligned}$$

Choosing $w = \mathfrak{A}\left(\mathcal{S}^{(i)}\right)$ and $u = \mathfrak{A}(\mathcal{S})$, and the fact that $\tilde{R}_{\mathcal{S}^{(i)}}\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right)\right) \leq \tilde{R}_{\mathcal{S}^{(i)}}(\mathfrak{A}(\mathcal{S}))$, it is

(4.5)

$$\tilde{R}_{\mathcal{S}}\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right)\right) - \tilde{R}_{\mathcal{S}}(\mathfrak{A}(\mathcal{S})) \leq \frac{\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell(\mathfrak{A}(\mathcal{S}), z_i)}{m} + \frac{\ell(\mathfrak{A}(\mathcal{S}), z') - \ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z'\right)}{m}$$

Note 35. Then (4.4) and (4.5) imply

(4.6)

$$\lambda \left\|\mathfrak{A}\left(\mathcal{S}^{(i)}\right) - \mathfrak{A}(\mathcal{S})\right\| \leq \frac{\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell(\mathfrak{A}(\mathcal{S}), z_i)}{m} + \frac{\ell(\mathfrak{A}(\mathcal{S}), z') - \ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z'\right)}{m}$$

Note 36. Now that we brought it in form (4.6), we can impose/use an additional assumption on the loss to bound it. In this example we use Lipstchitzness.

Assumption 37. *The loss function $\ell(\cdot, z)$ in (4.2) is convex and ρ-Lipschitz for any $z \in \mathcal{Z}$.*

Note 38. Given ρ-Lipschitzness in Assumption 37, it is

$$\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell(\mathfrak{A}(\mathcal{S}), z_i) \leq \rho \left\|\mathfrak{A}\left(\mathcal{S}^{(i)}\right) - \mathfrak{A}(\mathcal{S})\right\| \tag{4.7}$$

$$\ell(\mathfrak{A}(\mathcal{S}), z') - \ell\left(\mathcal{S}^{(i)}, z'\right) \leq \rho \left\|\mathfrak{A}\left(\mathcal{S}^{(i)}\right) - \mathfrak{A}(\mathcal{S})\right\| \tag{4.8}$$

and hence plugging 4.7 and (4.8) in (4.6) yields

$$\left\|\mathfrak{A}\left(\mathcal{S}^{(i)}\right) - \mathfrak{A}(\mathcal{S})\right\| \leq 2\frac{\rho}{\lambda m} \tag{4.9}$$

Note 39. Plugging (4.9) in (4.7) yields

$$\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell(\mathfrak{A}(\mathcal{S}), z_i) \leq 2\frac{\rho^2}{\lambda m}$$

Note 40. Using Theorem 19, we get an upper bound for the stability / over-fitting

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g(\mathfrak{A}(\mathcal{S})) - \hat{R}_{\mathcal{D}}(\mathfrak{A}(\mathcal{S}))\right) = \mathrm{E}_{\substack{\mathcal{S}\sim g,\, z'\sim g \\ i\sim \mathrm{U}\{1,\ldots,m\}}}\left(\ell\left(\mathfrak{A}\left(\mathcal{S}^{(i)}\right), z_i\right) - \ell(\mathfrak{A}(\mathcal{S}), z_i)\right) \leq 2\frac{\rho^2}{\lambda m}$$

Note 41. After this saga, the researcher could come to the conclusion that: A Ridge RLM learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with the loss function which is convex and ρ-Lipschitz, regularizer $J(\cdot;\lambda) = \lambda \|\cdot\|^2$, $\lambda > 0$, and learning rule trained against an iid sample $\mathcal{S} = \{z_i\}_{i=1}^m$ from g is on-averave-replace-one-stable with rate $\epsilon(m) = 2\frac{\rho^2}{\lambda m}$; i.e.

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g(\mathfrak{A}(\mathcal{S})) - \hat{R}_{\mathcal{S}}(\mathfrak{A}(\mathcal{S}))\right) \leq 2\frac{\rho^2}{\lambda m} \tag{4.10}$$

Note 42. From (4.10), we see that stability improves (and over-fitting decreases) as the shrinkage parameter λ increases.

4.3. **Bounding the Risk (2.3).**

Note 43. Given the bounds (4.3) and (4.10), the decomposition of the expected Risk in (2.3) yields that: A Ridge RLM learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with the loss function which is convex and ρ-Lipschitz, regularizer $J(\cdot;\lambda) = \lambda \|\cdot\|^2$, $\lambda > 0$, and learning rule trained against an iid sample $\mathcal{S} = \{z_i\}_{i=1}^m$ from g has Expected Risk bound

$$\mathrm{E}_{\mathcal{S}\sim g}(R_g(\mathfrak{A}(\mathcal{S}))) \leq \underbrace{R_g(w) + \lambda \|w\|_2^2}_{\text{(I)}} + \underbrace{2\frac{\rho^2}{\lambda m}}_{\text{(II)}}; \quad \forall w \in \mathcal{W} \tag{4.11}$$

Note 44. From (4.11), we see that there is a trade-off between Empirical Risk (I) and stability/overfitting (II) with regards the regularization parameter λ. It is desirable to use the optimal $\lambda > 0$ corresponding to the smallest bound in (4.11); it has to both fit the training data well (but perhaps not too well) and be very stable to different training data from the same g (but perhaps not too stable)!

Assumption 45. *Assume that the learning problem is additionally B-bounded by $B > 0$; i.e. $\mathcal{H} = \left\{w \in \mathbb{R}^d : \|w\|_2 \leq B\right\}$.*

Note 46. If additionally the learning problem is B-bounded then the upper bound in (4.11) becomes

$$\mathrm{E}_{\mathcal{S}\sim g}(R_g(\mathfrak{A}(\mathcal{S}))) \leq \underbrace{\min_{w\in\mathcal{H}} R_g(w) + \lambda B^2}_{\text{(I)}} + \underbrace{2\frac{\rho^2}{\lambda m}}_{\text{(II)}} \tag{4.12}$$

Note 47. Furthermore, if we choose a shrinkage parameter $\lambda_m = \sqrt{\frac{2}{m}}\frac{\rho}{B}$ in (4.2) the upper bound in (4.11) becomes

$$\mathrm{E}_{\mathcal{S}\sim g}(R_g(\mathfrak{A}(\mathcal{S}))) \leq \min_{w\in\mathcal{H}} R_g(w) + \lambda_m B\sqrt{\frac{8}{m}} \tag{4.13}$$

4.4. **Deriving PAC guarantees.**

Note 48. In particular, if the size m of the training sample $\mathcal{S}$ is

$$m \geq \frac{\lambda_m^2 B^2 8}{\epsilon^2} = \frac{8\rho^2 B^2}{\epsilon^2}$$

for some $\epsilon > 0$ then for every distribution g, the upper bound in (4.11) becomes

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right) \leq \min_{w\in\mathcal{H}} R_g\left(w\right) + \epsilon \tag{4.14}$$

and hence we can have a PAC guarantee that is summarized in the following.

Summary 49. Let $(\mathcal{H}, \mathcal{Z}, \ell)$ be a learning problem convex-Lipschitz-bounded with parameters ρ and B, and let m be the training data size. The associated Ridge Regularized Loss Minimization rule with regularization function $\lambda_m = \sqrt{\frac{2}{m}}\frac{\rho}{B}$ satisfies

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right) \leq \min_{w\in\mathcal{H}} R_g\left(w\right) + \lambda_m B\sqrt{\frac{8}{m}}$$

Moreover, if $m \geq \frac{8\rho^2 B^2}{\epsilon^2}$, $\epsilon > 0$ then for every distribution g we can get a PAC -guarantee

$$\mathrm{E}_{\mathcal{S}\sim g}\left(R_g\left(\mathfrak{A}\left(\mathcal{S}\right)\right)\right) \leq \min_{w\in\mathcal{H}} R_g\left(w\right) + \epsilon$$

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

Lecture notes 4: 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:

(1) Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
- Ch. 14.1 Gradient Descent

1. Motivations

Problem 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

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

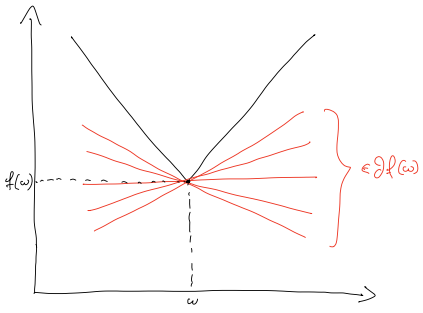
or its corresponding regularized version. If analytical minimization of (1.1) is impossible or impractical, numerical procedures can be applied; eg Gradient Descent (GD) algorithms. Such approaches introduce numerical errors in the solution.

2. subgradient

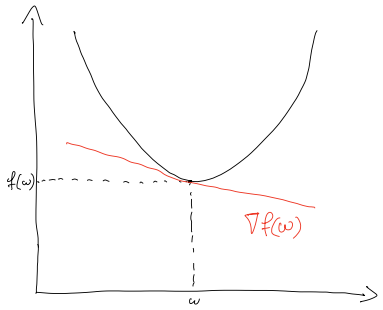
Definition 2. Vector v is called subgradient of a function $f : S \to \mathbb{R}$ at $w \in S$ if

$$\forall u \in S, \;\; f(u) \geq f(w) + \langle u - w, v \rangle \tag{2.1}$$

Note 3. There may be more than one subgradients of a function at a specific point. As seen by (2.1), 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 (2.1) in the non-differentiable case



(B) gradient satisfying the equality in (2.1) in the differentiable case

Definition 4. The set of subgradients of function $f : S \to \mathbb{R}$ at $w \in S$ is denoted by $\partial f(w)$.

Fact 5. *Properties of subgradient sets useful for the subgradient 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 (2.1) 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*

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

Example 6. 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 5, 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 (2.1) it is

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

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

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

3. Description of Gradient Descent

Problem 7. For the sake of notation simplicity and generalization, we will present Gradient Descent (GD) in the following minimization problem

$$w^* = \arg\min_{w \in \mathcal{H}} \left(f\left(w\right) \right) \tag{3.1}$$

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

Algorithm 8. *Gradient Descent (GD) algorithm with learning rate $\eta_t > 0$ for the solution of the minimization problem (3.1)*

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

(1) *compute*

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

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

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

Note 9. (Intuition) Assume f is differentiable. GD produces a chain $\left\{w^{(t)}\right\}$ that drifts towards a minimum w^*. The chain is directed towards the opposite direction of that of the gradient $\nabla f\left(\cdot\right)$ and at a rate controlled by the learning rate η_t.

Note 10. (More intuition) Assume f is differentiable. Consider the (1st order) Taylor polynomial for the approximation of $f\left(w\right)$ in a small area around u (i.e. $\|v - u\| = \text{small}$)

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

Assuming convexity for f, it is

$$f\left(u\right) \geq \underbrace{f\left(w\right) + \langle u - w, \nabla f\left(w\right) \rangle}_{= P(u;w)} \tag{3.3}$$

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 $P\left(w\right)$ of $f\left(w\right)$ at $w^{(t)}$

$$P\left(w; w^{(t)}\right) = f\left(w^{(t)}\right) + \langle w - w^{(t)}, \nabla f\left(w^{(t)}\right) \rangle. \tag{3.4}$$

while hoping that this mechanism would push the produced chain $\left\{w^{(t)}\right\}$ towards the minimum because of (3.3). Hence we could recursively minimize the linear approximation (3.4) and the

distance between the current state $w^{(t)}$ and the next w value to produce $w^{(t+1)}$; namely

$$
\begin{aligned}
w^{(t+1)} =& \arg\min_{\forall w} \left(\frac{1}{2} \left\| w - w^{(t)} \right\|^2 + \eta P\left(w; w^{(t)} \right) \right) \\
=& \arg\min_{\forall w} \left(\frac{1}{2} \left\| w - w^{(t)} \right\|^2 + \eta \left(f\left(w^{(t)} \right) + \langle w - w^{(t)}, \nabla f\left(w^{(t+1)} \right) \rangle \right) \right) \\
=& w^{(t)} - \eta \nabla f\left(w^{(t)} \right)
\end{aligned} \tag{3.5}
$$

where parameter $\eta > 0$ controls the trade off in (3.5).

Note 11. GD output (e.g. in Note 14) converges to a local minimum, $w_{\mathrm{GD}}^{(T)} \to w_*$ (in some sense), under different sets of regularity conditions (some are weaker other stronger). Section 4 has a brief analysis.

Note 12. The parameter η_t is called learning rate (or step size, gain). It determines the size of the steps GD takes to reach a (local) minimum. $\{\eta_t\}$ is a non-negative sequence and it is chosen by the practitioner. In principle, regularity conditions (Note 11) 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 beginning 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 against a small fraction of the training data set.

Note 13. There are several practical termination criteria that can be used in GD Algorithm 8(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 $\left\|\nabla f\left(w^{(t)}\right)\right\| \leq \epsilon$, 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.

Note 14. Given T iterations of GD algorithm, 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)

$$w_{\mathrm{GD}}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)} \tag{3.6}$$

(2) or the best value discovered

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

(3) or the last value discovered

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

4. Analysis of Gradient descent (Algorithm 8)

Note 15. We analyze the behavior of GD addressing the minimization Problem 7 under Assumptions 16.

Assumption 16. *For the sake of the analysis of the GD, let us consider:*

(1) *The function $f(\cdot)$ is convex and Lipschitz*
(2) *GD has a constant learning rate $\eta_t = \eta$,*
(3) *GD output $w_{GD}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$*

No need to memorize

Lemma 17. *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. Omitted; see the Exercise 12 in the Exercise sheet 1. □

Note 18. 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 17.

Note 19. Consider the minimization problem (3.1). Given Assumptions 16, the error can be bounded as

$$f\left(w_{\mathrm{GD}}^{(T)}\right) - f(w^*) \leq \frac{\|w^*\|^2}{2\eta T} + \frac{\eta}{2}\frac{1}{T}\sum_{t=1}^{T}\|v_t\|^2 \tag{4.2}$$

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

Proof. It is[1]

$$
\begin{aligned}
f\left(w_{\mathrm{GD}}^{(T)}\right) - f\left(w^*\right) =& f\left(\frac{1}{T}\sum_{t=1}^{T} w_t\right) - f\left(w^*\right) \\
\leq& \frac{1}{T}\sum_{t=1}^{T}\left(f\left(w_t\right) - f\left(w^*\right)\right) && \text{(by Jensen's inequality)} \qquad (4.3) \\
\leq& \frac{1}{T}\sum_{t=1}^{T}\left\langle w^{(t)} - w^*, v_t\right\rangle && \text{(by convexity of } f\left(\cdot\right)) \\
\leq& \frac{\left\|w^*\right\|^2}{2\eta T} + \frac{\eta}{2}\frac{1}{T}\sum_{t=1}^{T}\left\|v_t\right\|^2 && \text{(by roceeding Lemma)} \qquad (4.4)
\end{aligned}
$$

□

Note 20. Note 19 shows that it is important to bound the (sub-)gradient in (4.2) in a meaningful manner. Lemma 21 shows that if we assume Lipschitzness as well, the (sub-)gradient has the so-called self-bounded behavior.

Lemma 21. [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 $v \in \partial f\left(w\right)$ it is $\left\|v\right\| \leq \rho$.*

Proof. $\Longrightarrow$ Let $f : S \to \mathbb{R}$ be ρ-Lipschitz over convex set S, $w \in S$ and $v \in \partial f\left(w\right)$.

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

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

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

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

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

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

Note 22. The following summaries Note 19 and Lemma 21 with respect to the GD algorithm under Assumption 16.

Note 23. Let $f\left(\cdot\right)$ be a convex and ρ-Lipschitz function. If we run GD algorithm of f with learning rate $\eta > 0$ for T steps the output $w_{\mathrm{GD}}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$ satisfies

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

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

[2] If this was a Homework there would be a Hint:

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

where $\|v_t\| \leq \rho$, $v_t \in \partial f\left(w^{(t)}\right)$. If $f(\cdot)$ is differentiable then $v_t = \nabla f\left(w^{(t)}\right)$.

Proof. Straightforward from Lemma 17 and Note 19. □

Note 24. The following shows that given a 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.

Note 25. (Cont Prop. 23) Let $f(\cdot)$ be a convex and ρ-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

$$f\left(w_{\text{GD}}^{(T)}\right) - f(w^*) \leq \frac{B\rho}{\sqrt{T}} \tag{4.5}$$

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

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

Proof. Part 1 is a simple substitution from Proposition 23, and part 2 is implied from part 1. □

The result on Note 25 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).

5. Examples [3]

Example 26. Consider the simple Normal linear regression problem where the dataset $\{z_i = (y_i, x_i)\}_{i=1}^n \in \mathcal{S}$ is generated from a Normal data generating model

$$\begin{pmatrix} y_i \\ x_i \end{pmatrix} \overset{\text{iid}}{\sim} \text{N}\left(\begin{pmatrix} \mu_y \\ \mu_x \end{pmatrix}, \begin{bmatrix} \sigma_y^2 & \rho\sqrt{\sigma_y^2\sigma_x^2} \\ \rho\sqrt{\sigma_y^2\sigma_x^2} & \sigma_x^2 \end{bmatrix}\right) \tag{5.1}$$

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

$$\begin{pmatrix} w_1^* \\ w_2^* \end{pmatrix} = \begin{pmatrix} \mu_y - \rho\frac{\sigma_y}{\sigma_x}\mu_x \\ \rho\frac{\sigma_y}{\sigma_x} \end{pmatrix}. \tag{5.2}$$

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

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

The GD Algorithm 8 with learning rate η is

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

[3] Code is available in `https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2025/tree/main/Lecture_notes/code/example_GD.R`

(1) compute

$$w^{(t+1)} = w^{(t)} - \eta v_t, \qquad \text{where } v_t = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)}\bar{x} - 2\bar{y} \\ 2w_1^{(t)}\bar{x} + 2w_2^{(t)}\overline{x^2} - 2y^\top x \end{pmatrix} \tag{5.3}$$

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

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

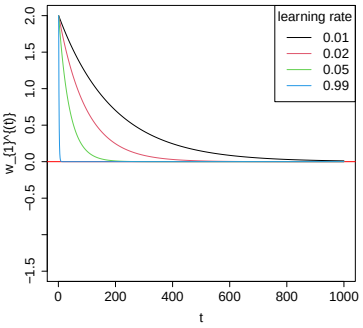
This is because $\hat{R}_{\mathcal{D}}(w)$ is differentiable in $\mathbb{R}^2$ so $\partial \hat{R}_{\mathcal{D}}(w) = \left\{\nabla \hat{R}_{\mathcal{D}}(w)\right\}$ and because

$$\nabla \hat{R}_{\mathcal{D}}(w) = \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}w_1}\hat{R}_{\mathcal{S}}(w) \\ \frac{\mathrm{d}}{\mathrm{d}w_2}\hat{R}_{\mathcal{S}}(w) \end{pmatrix} = ... = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)}\bar{x} - 2\bar{y} \\ 2w_1^{(t)}\bar{x} + 2w_2^{(t)}\overline{x^2} - 2y^\top x \end{pmatrix}$$

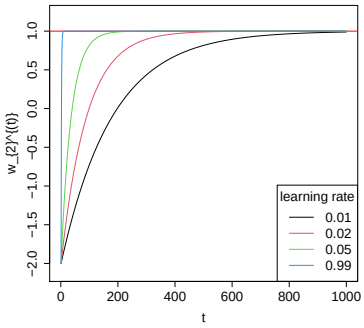
Consider data size $n = 100$, and parameters $\rho = 0.2$, $\sigma_y^2 = 1$ and $\sigma_x^2 = 1$. Then the real value (5.2) that I need to learn equals to $w^* = (0, 1)^\top$. Consider a GD seed $w_0 = (2, -2)$, and total number of iterations $T = 1000$.

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

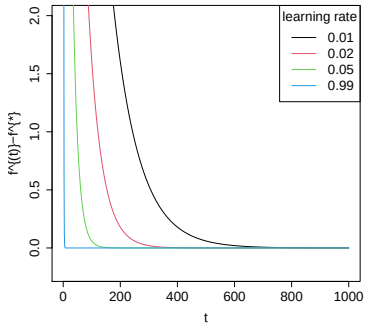
Figures 5.1d, 5.1e, and 5.1f present trace plots of the chain $\left\{\left(w^{(t)}\right)\right\}$, and of the error $\hat{R}_{\mathcal{S}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{S}}(w^*)$ produced by running GD for $T = 1000$ total iterations and for learning rate $\eta = 1.0$ (previously considered) 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 (3.2) that the produced chain overshoots the global minimum; see the cartoon in Figures 5.1g and 5.1h.



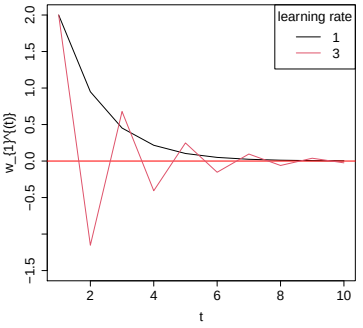
(A) $\left\{\left(w_1^{(t)}\right)\right\}$



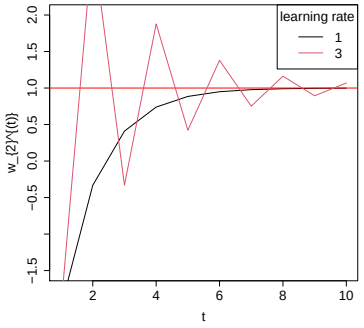
(B) $\left\{\left(w_2^{(t)}\right)\right\}$



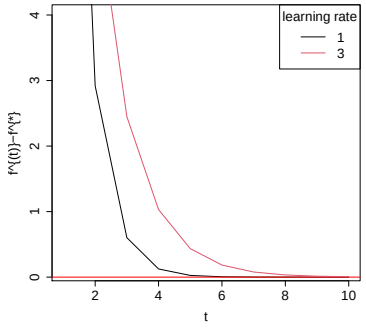
(C) $\hat{R}_{\mathcal{S}}\left(w^{(t)}\right) - \hat{R}_{\mathcal{S}}\left(w^*\right)$



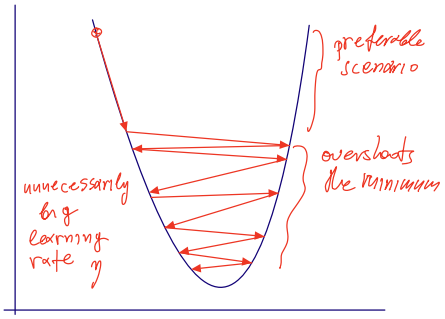
(D) $\left\{\left(w_1^{(t)}\right)\right\}$



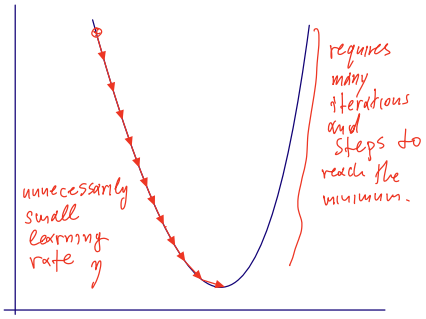
(E) $\left\{\left(w_2^{(t)}\right)\right\}$



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



(G) Unnecessarily large learning rate



(H) Unnecessarily small learning rate

FIGURE 5.1

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

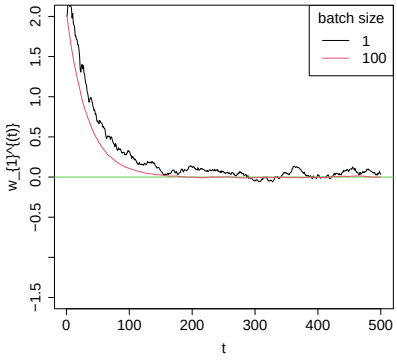
Because (3.7) is actually the gradient of the Risk function at $\tilde{w}$ and

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

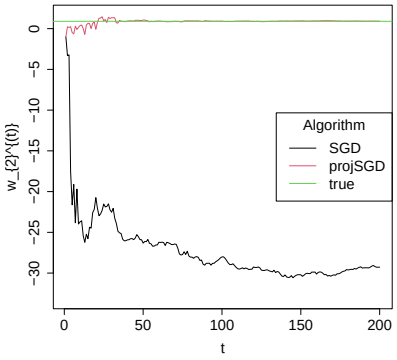
In Figure A.1c we observe the frequency of the snapshots has improved the convergence; however this is not a panacea as seen in Figure A.1f.

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

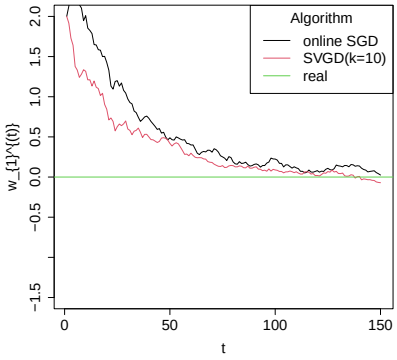
Appendix A. Plots



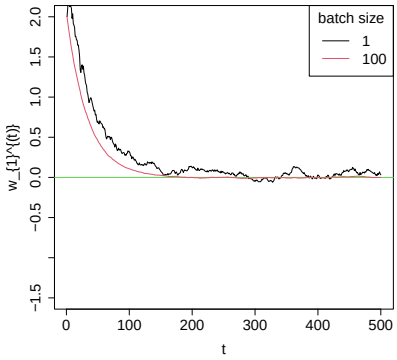
(A) batch SGD Example 30 w_1



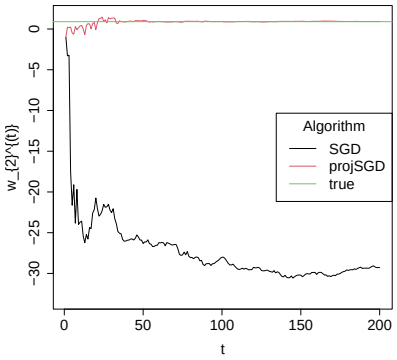
(B) online SGD with projection Example 6w_1



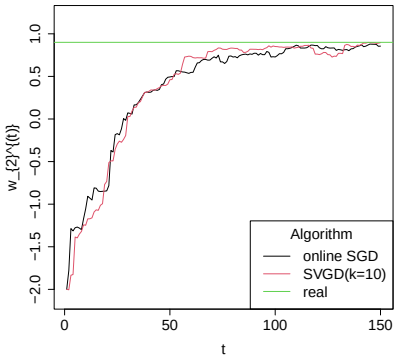
(C) SVRG Example 27 w_1



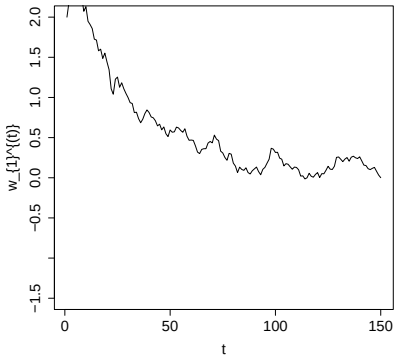
(D) batch SGD Example 30 w_2



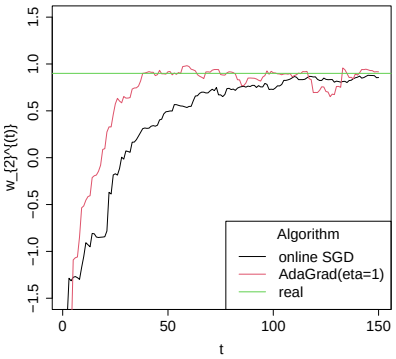
(E) online SGD with projection Example 6 w_2



(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

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

Lecture notes 5: Stochastic gradient descent

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

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

Reading list & references:

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

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

1. Motivations for stochastic gradient descent

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

$$w^* = \arg\min_w (R_g(w)) = \arg\min_w (\mathrm{E}_{z\sim g}(\ell(h_w, z))) \tag{1.1}$$

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

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

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

2. Stochastic gradient descent

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

$$w^* = \arg\min_w (f(w)) \tag{2.1}$$

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

Algorithm 6. *Stochastic Gradient Descent (SGD) with learning rate $\eta_t > 0$ for Problem 5*

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

(1) *compute*

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

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

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

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

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

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

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

where

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

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

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

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

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

(2) or the best value discovered

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

(3) or the last value discovered

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

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

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

(1) $\eta_t \geq 0$,

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

Note 12. The popular learning rates $\{\eta_t\}$ in Note 14 in Lect. notes 4 "Gradient descent" satisfy Condition 11 and hence can be used in SGD too. Once parameterized, η_t can be tuned based on pilot runs using a reasonably small fraction of the training data set.

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

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

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

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

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

3. Analysis of SGD (Algorithm 6)

Note 14. Recall (Note 8) that the stochasticity of SGD comes from the stochastic sub-gradients $\{v_t\}$ in (2.2); hence the expectations below are under these random vectors' distributions.

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

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

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

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

I will try to use Lemma 17 from Lect. notes 4, hence I need to show

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

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

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

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

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

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

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

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

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

Then Lemma 17 in Lect. notes 4 "Gradient descent" implies

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

□

Note 16. The upper bound in (3.1) depends on $\mathrm{E}\left\|v_{t}\right\|^{2}$ which has to be bounded and hence the variation of v_t because

$$\mathrm{E}\left\|v_{t}\right\|^{2}=\mathrm{E}\left(\left\|v_{t}-\mathrm{E}\left(v_{t}\right)\right\|^{2}\right)+\left\|\mathrm{E}\left(v_{t}\right)\right\|^{2}=\mathrm{tr}\left(\mathrm{Var}\left(v_{t}\right)\right)+\left\|\mathrm{E}\left(v_{t}\right)\right\|^{2}\tag{3.5}$$

where $\mathrm{E}\left(\left\|v_{t}-\mathrm{E}\left(v_{t}\right)\right\|^{2}\right)=\mathrm{tr}\left(\mathrm{Var}\left(v_{t}\right)\right)$ is the sum of all the variances at each dimension of v_t and $\left\|\mathrm{E}\left(v_{t}\right)\right\|^{2}$ is a finite constant as $\mathrm{E}\left(v_{t}\right)\in\partial_{w}f\left(w_{t}\right)$ is the unbiased estimator of the sub-gradient by construction.

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

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

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

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

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

Proof. It follows from Proposition 15. Condition $\mathrm{E}\left\|v_{t}\right\|^{2}\leq\rho^{2}$ can be achieved, for instance by assuming Lipschitz (See Lemma 21 from "Lect. notes 4: Gradient descent"). □

4. Implementation of SGD in the learning Problem 1

Note 18. Note 19 implements SGD to the learning problem (Problem 1)

$$w^{*}=\arg\min_{w}\left(R_{g}\left(w\right)\right)=\arg\min_{w}\left(\mathrm{E}_{z\sim g}\left(\ell\left(h_{w},z\right)\right)\right)$$

Note 19. For a randomly drawn example $z\sim g(\cdot)$, the sub-gradient v of $\ell\left(w,z\right)$ at point w is an unbiased estimator of the sub-gradient of the risk $R_{g}\left(w\right)$ at point w. I.e. if $v\in\partial_{w}\ell\left(w,z\right)$ where $z\sim g(\cdot)$ then $\mathrm{E}_{z\sim g}\left(v|w\right)\in\partial_{w}R_{g}\left(w\right)$.

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

$$\ell\left(u,z\right)-\ell\left(w,z\right)\geq\left\langle u-w,v\right\rangle \tag{4.1}$$

It is

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

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

Note 20. Similarly, the average $\bar{v}=\frac{1}{m}\sum_{i=1}^{m}v_{i}$ of sub-gradients $v_{i}\in\partial_{w}\ell\left(w,\tilde{z}_{i}\right)$ at point w given a randomly drawn set of m examples $\left\{\tilde{z}_{i}\sim g(\cdot);\,i=1,...,m\right\}$ is an unbiased estimator of the risk function sub-gradient; i.e. $\mathrm{E}_{z\sim g}\left(\bar{v}|w\right)\in\partial_{w}R_{g}\left(w\right)$.

Example 21. Assume a differentiable loss function $\ell\left(\cdot,z\right)$ at each z and set sub-gradient $v=\nabla_{w}\ell\left(w,z\right)$, then

$$\mathrm{E}_{z\sim g}\left(v|w\right)=\mathrm{E}_{z\sim g}\left(\nabla_{w}\ell\left(h_{w},z\right)|w\right)=\nabla_{w}\mathrm{E}_{z\sim g}\left(\ell\left(h_{w},z\right)|w\right)=\nabla_{w}R_{g}\left(w\right)$$

Note 22. Assume there is available a finite dataset $\mathcal{S}_{n}=\left\{z_{i};i=1,...,n\right\}$ of size n which consists of independent realizations z_{i} from the data generating distribution g; $z_{i}\overset{\text{ind}}{\sim}g$. SGD (Algorithm 23) is an implementation of the SGD (Algorithm 6) in the learning Problem 1.

Algorithm 23. *Stochastic Gradient Descent with learning rate $\eta_{t}>0$, and batch size m, for Problem 1.*

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

1. *randomly generate a set $\mathcal{J}^{(t)}\subseteq\left\{1,...,n\right\}^{m}$ of m indices from 1 to n with or without replacement.*
2. *compute*

$$w^{(t+1)}=w^{(t)}-\eta_{t}v_{t}, \tag{4.2}$$

where $v_{t}=\frac{1}{\left|\mathcal{J}^{(t)}\right|}\sum_{j\in\mathcal{J}^{(t)}}\nu_{t,j}$ and $\nu_{t,j}\in\partial_{w}\ell\left(w^{(t)},z_{j}\right)$ for any $j\in\mathcal{J}^{(t)}$.

3. *terminate if a termination criterion is satisfied*

Note 24. If it is possible to sample anytime fresh examples z_i directly from the data generation model g instead of just having access to only a given finite dataset of examples $\mathcal{S}_n$, then step 1 in Algorithm 23 can become

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

Note 25. Algorithm 23 may be called online SGD when using sub-samples of size one ($m = 1$) and batch SGD when using sub-samples of larger sizes ($m > 1$) .

Note 26. In theory, using larger batch size m has the benefit that reduces the variance of v_t at iteration t due to averaging effect, stabilizes the SGD algorithm, and reduces the error bound (3.1); see Remark 16.

Note 27. In practice, for a given fixed computational time, using smaller batch size m has the benefit that the algorithm iterates faster as each iteration processes less number of examples. E.g., consider the extreme cases GD vs online SGD utilized in a scenario of big-data (large training data set): if the dataset consists of several replications of the same values, GD (using all the data) has to process the same information multiple times, while the online SGD (using only one example at a time) would avoid this issue.

Note 28. In practice, for a given fixed computational time, it is possible for a SGD with smaller batch size m to present better generalization properties (wrt the theoretical assumptions) than those with larger m (GD is included). It is observed for the former to be often less prone to getting stuck in shallow local minima because of the additional amount of "noise" E.g., consider the extreme cases GD ($\mathcal{J}^{(t)} \equiv \{1, ..., n\}$) vs online SGD ($m = 1$) in a scenario with non-convex risk function (e.g. our theoretical assumptions as violated): if the Risk function presents local minima, considering less examples randomly chosen each time may cause fluctuations in the gradient that allow the chain to accidentally jump/escape to an area with a lower minimum.

Proposition 29. *(Implementing Proposition 17) Consider a convex-Lipschitz-bounded problem $(\mathcal{H}, \mathcal{Z}, \ell)$ with parameters ρ and B. Then for every $\epsilon > 0$, if we run SGD Algorithm 6 to minimize Problem 1 for $T \geq \frac{B^2}{\epsilon^2}\rho^2$ iterations then it produces output $w_{SGD}^{(T)}$ satisfying*

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

and hence we can achieve PAC guarantees.

Proof. It is just re-writing the formula in part 2 of Proposition 17. Notice that here, condition $\mathrm{E}\left\|v_t\right\|^2 \leq \rho^2$ is satisfied by the self-bounding property from Lipschitzness of the loss (see Lemma 21 from "Lect. notes 4: Gradient descent"); i.e. if ℓ is ρ-Lipschitz, then $\|v_t\| \leq \rho$ where $\nu_t \in \partial_w \ell\left(w^{(t)}, z_j\right)$ at $j \in \mathcal{J}^{(t)}$. □

Example 30. [1] We continue Example 26 in Section 4 of Lect. notes 4. Recall, we considered a hypothesis space $\mathcal{H}$ of linear functions $h : \mathbb{R}^2 \to \mathbb{R}$ with $h\left(w\right) = w_1 + w_2 x$, $w = \left(w_1, w_2\right)^\top$,

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

and $\ell\left(w, z=(x,y)^\top\right) = (y_i - w_1 - w_2 x)^2$. Here we consider a big dataset $\mathcal{S}_n = \{z_1, ..., z_n\}$ with $n = 10^6$ examples.

The batch SGD algorithm (Algorithm 23) with learning rate η_t and batch size $m = 10$ is
For $t = 1, 2, 3, ...$ iterate:

(1) Randomly generate a set $\mathcal{J}^{(t)}$ by drawing $m = 10$ numbers from $\left\{1, ..., n = 10^6\right\}$

(2) compute

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

where

$$v_t = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} \frac{1}{m} \sum_{j\in\mathcal{J}^{(t)}} x_j - 2\frac{1}{m}\sum_{j\in\mathcal{J}^{(t)}} y_j \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} \frac{1}{m} \sum_{j\in\mathcal{J}^{(t)}} x_j^2 - 2\sum_{j\in\mathcal{J}^{(t)}} y_j x_j \end{pmatrix}$$

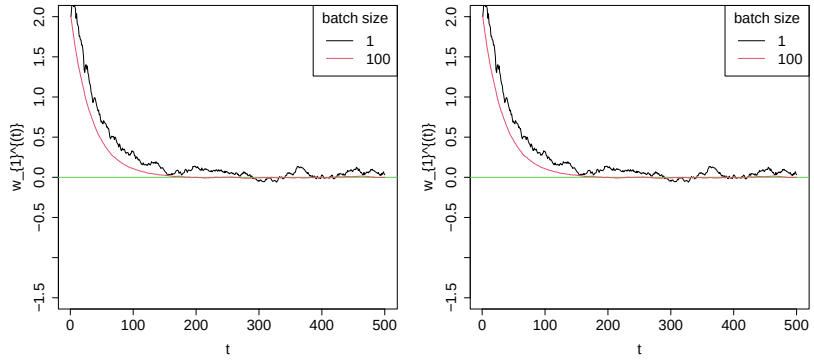
(3) if $t \geq T = 1000$ STOP

because

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

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

Appendix A. Plots



(A) SGD for w_1 (B) SGD for w_2

Figure A.1. Simulations of the Example 30