

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

2024/01/25 at 14:09:38

Concepts

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



Reading list

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

Main texts:

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 It is very comprehensive (compared to others) and it goes deep enough for the module although it may be a bit outdated.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

Supplementary textbooks:

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

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

Contents

1. Handout 1: Machine learning –A recap on: definitions, notation, and formalism
2. Handout 2: Elements of convex learning problems
3. Handout 3: Learnability, and stability
4. Handout 4: Gradient descent
5. Handout 5: Stochastic gradient descent

Handout 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 LOOSE DEFINITIONS

Pattern recognition is the automated discovery of patterns and regularities in data $z \in \mathcal{Z}$. **Machine learning (ML)** are statistical procedures for building and understanding probabilistic methods that 'learn'. **ML algorithms** \mathfrak{A} build a (probabilistic/deterministic) model able to make predictions or decisions with minimum human interference and can be used for pattern recognition. **Learning** (or training, estimation, 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. $x \mapsto y$. 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. (LOOSE) NOTATION & DEFINITIONS IN LEARNING

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

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

Example 3. (Linear Regression)¹ Consider the regression problem where the goal is to learn the mapping $x \rightarrow y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \subseteq \mathbb{R}$. A hypothesis is a linear function $h : \mathcal{X} \rightarrow \mathcal{Y}$ (that learner wishes to learn) with $h(x) = \langle w, x \rangle$ approximating the mapping $x \rightarrow y$. The hypothesis set $\mathcal{H} = \{x \rightarrow \langle w, x \rangle : w \in \mathbb{R}^d\}$.

Example 4. (Binary Classification) Consider the classification problem where the goal is to learn the mapping $x \rightarrow y$ where $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} \{-1, +1\}$. A hypothesis can be a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ with $h(x) = \text{sign}(\langle w, x \rangle)$ approximating the mapping $x \rightarrow y$. The hypothesis set $\mathcal{H} = \{x \rightarrow \text{sign}(\langle w, x \rangle) : w \in \mathbb{R}^d\}$.

Definition 5. Training data set \mathcal{S} of size m is any finite sequence of pairs $(z_i = (x_i, y_i) ; i = 1, \dots, m)$ in $\mathcal{X} \times \mathcal{Y}$; i.e. $\mathcal{S} = \{(x_i, y_i) ; i = 1, \dots, m\}$. This is the information that the learner has assess.

Definition 6. Data generation model $g(\cdot)$ is the probability distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, unknown to the learner that has generated the data. E.g. $z \sim g$.

Definition 7. We denote as $\mathfrak{A}(\mathcal{S})$ the hypothesis (outcome) that a learning algorithm \mathfrak{A} returns given training sample \mathcal{S} .

Definition 8. (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 $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}_+$. 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 9. (Cont. Example 3) 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 10. (Cont. Example 4) In binary classification problems with hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{Y} = \{0, 1\}$ is discrete, a loss function can be

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

¹ $\langle w, x \rangle = w^\top x$

Definition 11. (Risk function) The risk function $R_g(h)$ 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.

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

Remark 12. In learning, an ideal way to obtain an optimal predictor h^* is to compute the minimizer of the risk; i.e.

$$(2.2) \quad h^* = \arg \min_{\forall h} (R_g(h))$$

Example 13. (Cont. Ex. 9) The risk function is $R_g(h) = \mathbb{E}_{z \sim g} (h(x) - y)^2$, and it measures the quality of the hypothesis function $h : \mathcal{X} \rightarrow \mathcal{Y}$, (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 from h and the true target values y at every x .

Note 14. Computing the risk minimizer may be practically challenging due to the integration w.r.t. the unknown data generation model g involved in the expectation (2.1). Sub-optimally, one may use the Empirical risk function instead of the Risk function in (2.2).

Definition 15. (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 $S = (z_1, \dots, z_m) \in \mathcal{Z}^m$; i.e.

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

Remark 16. Given Empirical Risk Function (ERF) $\hat{R}_S(h)$ of h the optimal predictor h^* is the minimizer of the ERF; i.e.

$$(2.3) \quad h^* = \arg \min_{\forall h} (\hat{R}_S(h))$$

Example 17. (Cont. Example 13) Given given sample $S = \{(x_i, y_i); i = 1, \dots, m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2$.

Example 18. (Cont. Example 10) Given given sample $S = \{(x_i, y_i); i = 1, \dots, m\}$ the empirical risk function is $\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^m 1(h(x_i) \neq y_i)$.

Remark 19. If the Hypothesis set \mathcal{H} is a known parametric family of functions; i.e. $\mathcal{H} = \{h_w(\cdot); w \in \mathcal{W}\}$ parameterized by unknown $w \in \mathcal{W}$, then we can equivalently consider $\mathcal{H} = \{w \in \mathcal{W}\} = \mathcal{W}$ keeping in mind that the learner's output is restricted to $h_w(\cdot)$.

Example 20. Consider the multiple linear regression problem with regressors $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and response $y \in \mathcal{Y} \subseteq \mathbb{R}$. Because it involves only linear functions as predictors $h_w(x) = \langle w, x \rangle$, we could consider a hypothesis class $\mathcal{H} = \{w \in \mathbb{R}^d\} = \mathbb{R}^d$ and loss function $\ell(w, (x, y)) = (\langle w, x \rangle - y)^2$ for computational simplicity. The latter will be mainly used.

Example 21. 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

$$(2.4) \quad R_g(\theta) = \mathbb{E}_{z \sim g} \left(\log \left(\frac{g(z)}{f_\theta(z)} \right) \right) = \mathbb{E}_{z \sim g} (\log(g(z))) - \mathbb{E}_{z \sim g} (\log(f_\theta(z)))$$

whose minimizer is

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

as the first term in (2.4) is constant. Note that in the Maximum Likelihood Estimation technique the MLE θ_{MLE} is the minimizer

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

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

Definition 22. 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 23. 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 $\mathcal{H} = \{x \rightarrow \langle w, x \rangle : w \in \mathbb{R}^d\}$, and loss function $\ell_{\text{sq}}(h, (x, y)) = (h(x) - y)^2$.

APPENDIX A. USEFUL THINGS

Below are some standard notation used as default in the notes except in cases that is defined otherwise.

- q -norm: When $x \in \mathbb{R}^d$ $\|x\|_q := \left(\sum_{j=1}^d x_j^q \right)^{1/q}$
- Manhattan norm: When $x \in \mathbb{R}^d$ $\|x\|_1 := \sum_{j=1}^d |x_j|$
- Euclidean norm: When $x \in \mathbb{R}^d$ $\|x\|_2 := \sqrt{\sum_{j=1}^d x_j^2}$. When $\|\cdot\|$ we will assume the Euclidean norm.
- Infinity norm or maximum norm: $\|x\|_\infty := \max_{\forall j} |x_j|$
- Inner product of x, y : If $x, y \in \mathbb{R}^d$ then $\langle x, y \rangle = x^\top y$. So $\langle x, x \rangle = \|x\|^2$

Also some standard formulas.

- Jensens' inequality: If $x \in \mathbb{R}^d$ and $f : \mathbb{R}^d \rightarrow \mathbb{R}$ then

$$\begin{cases} f(\mathbb{E}(x)) \leq \mathbb{E}(f(x)) & \text{if } f \text{ is convex} \\ f(\mathbb{E}(x)) \geq \mathbb{E}(f(x)) & \text{if } f \text{ is concave} \end{cases}$$

- Cauchy-Schwarz inequality: If $x, y \in \mathbb{R}^d$ then $|\langle x, y \rangle|^2 \leq \langle x, x \rangle \langle y, y \rangle$ equiv. $|\langle x, y \rangle| \leq \|x\| \|y\|$.

Handout 2: Elements of convex learning problems

Lecturer & author: Georgios P. Karagiannis

georgios.karagiannis@durham.ac.uk

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

Reading list & references:

- 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 concepts of convexity and smoothness that facilitate the analysis and understanding of the learning problems and their solutions that we will discuss (eg stochastic gradient descent, SVM) later on. Also learning problems with such characteristics can be learned more efficiently.

Note 2. Some of the ML problems discussed in the course (eg, Artificial neural networks, Gaussian process regression) are non-convex. To overcome this problem, we will introduce the concept of surrogate loss function that allows a non-convex problem to be handled with the tools introduced in the convex setting.

2. CONVEXITY

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

Note 4. Namely, a set C is convex if for any $u, v \in C$, the line segment between u and v is contained in C .

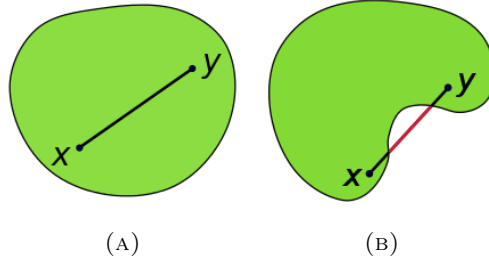


FIGURE 2.1. (2.1a) is a Convex set ; (2.1b) is a non-convex set

Example 5. For instance \mathbb{R}^d for $d \geq 1$ is a convex set.

Definition 6. Let C be a convex set. A function $f : C \rightarrow \mathbb{R}$ 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 7. The function $f : \mathbb{R} \rightarrow \mathbb{R}_+$ with $f(x) = x^2$ is convex function. For any $u, v \in C$ and for any $\alpha \in [0, 1]$ it is

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

Proposition 8. Every local minimum of a convex function is the global minimum.

Proposition 9. Let $f : C \rightarrow \mathbb{R}$ 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 10. Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 11. 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 10.

Proposition 12. Let $f_j : \mathbb{R}^d \rightarrow \mathbb{R}$ convex functions for $j = 1, \dots, r$. Then:

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

Solution.

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

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

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

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

Example 13. $g(x) = |x|$ is convex according to Example 12, as $g(x) = |x| = \max(-x, x)$.

3. LIPSCHITZNESS

Definition 14. Let $C \in \mathbb{R}^d$. Function $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ is ρ -Lipschitz over C if for every $w_1, w_2 \in C$ we have that

$$(3.1) \quad \|f(w_1) - f(w_2)\| \leq \rho \|w_1 - w_2\|. \quad \text{Lipschitz condition}$$

Conclusion 15. That means: a Lipschitz function $f(x)$ cannot change too drastically wrt x .

Example 16. Consider the function $f : \mathbb{R} \rightarrow \mathbb{R}_+$ with $f(x) = x^2$.

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

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

Solution.

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

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

(2) It is

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

Theorem 17. Let functions g_1 be ρ_1 -Lipschitz and g_2 be ρ_2 -Lipschitz. Then f with $f(x) = g_1(g_2(x))$ is $\rho_1\rho_2$ -Lipschitz.

Solution. See Exercise 2 from the exercise sheet

Example 18. Let functions g be ρ -Lipschitz. Then f with $f(x) = g(\langle v, x \rangle + b)$ is $(\rho|v|)$ -Lipschitz.

Solution. It is

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

Note 19. So, given Examples 16 and 18, in the linear regression setting using loss $\ell(w, z = (x, y)) = (w^\top x - y)^2$, the loss function is ρ -Lipschitz for a given $z = (x, y)$ and bounded $\|w\| < \rho$.

4. SMOOTHNESS

Definition 20. A differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is β -smooth if its gradient is β -Lipschitz; namely for all $v, w \in \mathbb{R}^d$

$$(4.1) \quad \|\nabla f(w_1) - \nabla f(w_2)\| \leq \beta \|w_1 - w_2\|.$$

Theorem 21. Function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is β -smooth iff

$$(4.2) \quad f(v) \leq f(w) + \langle \nabla f(w), v - w \rangle + \frac{\beta}{2} \|v - w\|^2$$

Theorem 22. Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ with $f(w) = g(\langle w, x \rangle + y)$ $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a β -smooth function. Then f is a $(\beta\|x\|^2)$ -smooth.

Proof. See Exercise 3 from the Exercise sheet □

Example 23. Let $f(w) = (\langle w, x \rangle + y)^2$ for $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Then f is $(2\|x\|^2)$ -smooth.

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

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

Hence from Theorem 22, f is $(2\|x\|^2)$ -smooth.

Example 24. 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 $(2\|x\|^2)$ -smooth.

Solution. Follows from Example 23.

5. CONVEX LEARNING PROBLEMS

Definition 25. Convex learning problem is a learning problem $(\mathcal{H}, \mathcal{Z}, \ell)$ that the hypothesis class \mathcal{H} is a convex set, and the loss function ℓ is a convex function for each example $z \in \mathcal{Z}$.

Example 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 $y \in \mathbb{R}$. This imposes a convex learning problem due to Examples 5 and 12.

Definition 27. 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 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-Lipschitz-Bounded Learning Problem if $\mathcal{H} = \{w \in \mathbb{R}^d : \|w\| \leq B\}$ due to Examples 12, and 16(2).

Definition 29. 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 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-Smooth-Bounded Learning Problem if $\mathcal{H} = \{w \in \mathbb{R}^d : \|w\| \leq B\}$ due to Examples 12, and 24.

Proposition 31. *If ℓ is a convex loss function and the class \mathcal{H} is convex, then the $ERM_{\mathcal{H}}$ problem, of minimizing the empirical risk $\hat{R}_{\mathcal{S}}(w)$ over \mathcal{H} , is a convex optimization problem (that is, a problem of minimizing a convex function over a convex set).*

Proof. The $ERM_{\mathcal{H}}$ problem is

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

given a sample $\mathcal{S} = \{z_1, \dots, z_m\}$ 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 Proposition (12). Hence ERM rule is a problem of minimizing a convex function subject to the constraint that the solution should be in a convex set. \square

Example 32. 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

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

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 Proposition 31.

Note 33. Problems like that in Proposition 31 can be efficiently solved with algorithms such as Stochastic Gradients Descent to be introduced later.

6. NON-CONVEX LEARNING PROBLEMS (SURROGATE TREATMENT)

Remark 34. A learning problem may involve non-convex loss function $\ell(w, z)$ which implies a non-convex risk function $R_g(w)$. However, our learning algorithm will be analyzed in the convex setting. A suitable treatment to overcome this difficulty would be to upper bound the non-convex loss function $\ell(w, z)$ by a convex surrogate loss function $\tilde{\ell}(w, z)$ for all w , and use $\tilde{\ell}(w, z)$ instead of $\ell(w, z)$.

Example 35. Consider the binary classification problem with inputs $x \in \mathcal{X}$, outputs $y \in \{-1, +1\}$; we need to learn $w \in \mathcal{H}$ from hypothesis class $\mathcal{H} \subset \mathbb{R}^d$ with respect to the loss

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

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

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

which is convex (Example 12) wrt w . Note that:

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

Then we can compute

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

instead of

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

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

Remark 36. (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}_{alg} is the output of the learning algorithm under $\tilde{R}_g(\cdot)$ then we have the upper bound

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

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.

7. STRONG CONVEXITY

Note 37. Strong convexity is a central concept in regularization, e.g. Ridge, as it makes a convex loss function strongly convex by adding a shrinkage term.

Definition 38. (Strongly convex functions) A function f is λ -strongly convex function is for all w , u , and $\alpha \in (0, 1)$ we have

$$(7.1) \quad f(\alpha w + (1 - \alpha)u) \leq \alpha f(w) + (1 - \alpha)f(u) - \frac{\lambda}{2}\alpha(1 - \alpha)\|w - u\|^2$$



FIGURE 7.1. Strongly convex function

Proposition 39.

- (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

Proof. Both can be checked from the definition by substitution. □

Lemma 40. If f is λ -strongly convex and $w^* = \arg \min_w f(w)$ is a minimizer of f then for any w

$$f(w) - f(w^*) \geq \frac{\lambda}{2} \|w - w^*\|^2$$

Proof. Exercise 7 in the Exercise sheet. □

Proposition 41. If ℓ is a convex loss function and the class \mathcal{H} is convex, then the Ridge $ERM_{\mathcal{H}}$ problem, with learning rule

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

is a 2λ -strongly convex learning problem.

Proposition 42. (ERM with Ridge regularization) If ℓ is a convex loss function, 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 the $ERM_{\mathcal{H}}$ 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 Proposition 31, $\lambda \|\cdot\|_2^2$ is 2λ -strongly convex, hence $\hat{R}_{\mathcal{S}}(w) + \lambda \|w\|_2^2$ is a 2λ -strongly convex function. Hence the above $ERM_{\mathcal{H}}$ problem is a strongly convex optimization problem. □

Handout 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. (too advanced)

1. LEARNABLE PROBABLY APPROXIMATELY CORRECT (PAC) LEARNING

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

Note 2. Learning algorithms \mathcal{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 \mathcal{A} will find a predictor $\mathcal{A}(\mathcal{S})$ 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 $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$ and a learning algorithm \mathcal{A} with the following property:

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

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

Note 5. It may be easier to work with expectations: by using Markov inequality (1.1) becomes

$$(1.2) \quad \Pr_{\mathcal{S} \sim g} \left(R_g(\mathcal{A}(\mathcal{S}_m)) - \min_{h' \in \mathcal{H}} (R_g(h')) \leq \epsilon \right) \geq 1 - \frac{1}{\epsilon} \mathbb{E}_{\mathcal{S} \sim g} \left(R_g(\mathcal{A}(\mathcal{S}_m)) - \min_{h' \in \mathcal{H}} (R_g(h')) \right)$$

and hence we need to work with expectations and bounded above.

Hint: Markov inequality $\Pr(X \geq a) \leq \frac{1}{a} \mathbb{E}(X)$ for $X \geq 0$.

Remark 6. About 1.2: The accuracy parameter ϵ determines how far the output rule $\mathfrak{A}(\mathcal{S}_m)$ of \mathfrak{A} can be from the optimal one (this corresponds to the ‘approximately correct’). The confidence parameter δ indicates how likely the classifier is to meet that accuracy requirement (corresponds to the “probably” part of “PAC”).

2. ANALYSIS OF THE RISK BASED ON THE TRADE-OFF FITTING VS STABILITY

Note 7. Let $R^* = \min_{h \in \mathcal{H}} (R(h))$ 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

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

Note 8. Over-fitting can be represented by $R_g(\mathfrak{A}(\mathcal{S})) - \hat{R}_{\mathcal{S}}(\mathfrak{A}(\mathcal{S}))$. However, $\hat{R}_{\mathcal{S}}(\cdot)$ 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 9. For a learning algorithm \mathfrak{A} , as a measure of over-fitting we consider the expected difference between true Risk and empirical Risk

$$(2.2) \quad \mathbb{E}_{\mathcal{S} \sim g} (R_g(\mathfrak{A}(\mathcal{S})) - \hat{R}_{\mathcal{S}}(\mathfrak{A}(\mathcal{S})))$$

Definition 10. We say that learning algorithm \mathfrak{A} suffers from over-fitting when (2.2) is ‘too’ large.

Note 11. The expected Risk of a learning algorithm $\mathfrak{A}(\mathcal{S})$ can be decomposed as

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

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

Proposition 13. 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

$$\mathbb{E}_{\mathcal{S} \sim g} (R_g(\mathfrak{A}(\mathcal{S}_m))) \leq \min_{h' \in \mathcal{H}} (R_g(h')) + \epsilon$$

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

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

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

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 $\mathcal{S} = \{z_1, \dots, z_m\}$ 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 $\mathcal{S}^{(i)} = \{z_1, \dots, z_{i-1}, z', z_{i+1}, \dots, z_m\}$ is another training dataset equal to \mathcal{S} but the i th element being replaced by another independent example $z' \sim g$, then a good learning algorithm \mathfrak{A} would produce a small value of

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

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

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

Note 18. The following result demonstrates the direct association of stability and over-fitting as defined in Definitions 10 & 17.

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

$$(3.2) \quad \mathbb{E}_{\mathcal{S} \sim g} \left(R_g(\mathfrak{A}(\mathcal{S})) - \hat{R}_{\mathcal{D}}(\mathfrak{A}(\mathcal{S})) \right) = \mathbb{E}_{\substack{\mathcal{S} \sim g, z' \sim g \\ i \sim U\{1, \dots, m\}}} \left(\ell(\mathfrak{A}(\mathcal{S}^{(i)}), z_i) - \ell(\mathfrak{A}(\mathcal{S}), z_i) \right)$$

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

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

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

and

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

□

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 $J : \mathcal{H} \rightarrow \mathbb{R}$ aims at a RLM learning rule that is the minimizer

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

where $\hat{R}_{\mathcal{S}}(\cdot) = \frac{1}{m} \sum_{i=1}^m \ell(\cdot, z_i)$, and $\mathcal{S} = \{z_1, \dots, z_m\} \subset$ 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 $J(w; \lambda) = \lambda \|w\|_2^2$ with $\lambda > 0$, $w \in \mathcal{W}$ and produces learning rule

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

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 $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots$ with $\mathcal{H}_i = \{w \in \mathbb{R}^d : \|w\|_2 < i\}$ 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

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

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

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

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

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. Lipschitzness.

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

Note 31. Let $\tilde{R}_S(w) = \hat{R}_S(w) + \lambda \|w\|_2^2$. $\tilde{R}_S(\cdot)$ is 2λ -strongly convex as the sum of a convex function $\hat{R}_S(\cdot)$ (Assumption 30) and a 2λ -strongly convex function $J(\cdot; \lambda) = \lambda \|\cdot\|_2^2$ (results directly from Definition 38 in Handout 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*

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

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

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

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

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

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

$$(4.5) \quad \tilde{R}_S(\mathfrak{A}(S^{(i)})) - \tilde{R}_S(\mathfrak{A}(S)) \leq \frac{\ell(\mathfrak{A}(S^{(i)}), z_i) - \ell(\mathfrak{A}(S), z_i)}{m} + \frac{\ell(\mathfrak{A}(\textcolor{red}{S}), z') - \ell(\mathfrak{A}(\textcolor{red}{S}^{(i)}), z')}{m}$$

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

$$(4.6) \quad \lambda \|\mathfrak{A}(S^{(i)}) - \mathfrak{A}(S)\| \leq \frac{\ell(\mathfrak{A}(S^{(i)}), z_i) - \ell(\mathfrak{A}(S), z_i)}{m} + \frac{\ell(\mathfrak{A}(\textcolor{red}{S}), z') - \ell(\mathfrak{A}(\textcolor{red}{S}^{(i)}), z')}{m}$$

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 Lipschitzness.

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

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

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

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

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

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

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

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

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

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-average-replace-one-stable with rate $\epsilon(m) = 2 \frac{\rho^2}{\lambda m}$; i.e.

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

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

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

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. $\mathcal{H} = \left\{ w \in \mathbb{R}^d : \|w\|_2^2 \leq B \right\}$.

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

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

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

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

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

$$(4.14) \quad \mathbb{E}_{\mathcal{S} \sim g} (R_g (\mathfrak{A} (\mathcal{S}))) \leq \min_{w \in \mathcal{H}} R_g (w) + \epsilon$$

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

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

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

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

APPENDIX A. RECALL STRONG CONVEX FUNCTIONS FROM HANDOUT 2.

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

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

- The function $f(w) = \lambda\|w\|_2^2$ is 2λ -strongly convex
- If f is λ -strongly convex and g is convex then $f + g$ is λ -strongly convex
- If f is λ -strongly convex and u is a minimizer of f then for any w

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

Handout 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

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

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

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

2. DESCRIPTION

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

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

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

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

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

Algorithm 1 Gradient descent algorithm with learning rate η_t

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

(1) compute

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

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

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

where

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

is the gradient of f at w .

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

Remark 6. (More intuition) Consider the (1st order) Taylor polynomial for the approximation of $f(w)$ in a small area around u (i.e. $\|v - u\| = \text{small}$)

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

Assuming convexity for f , it is

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

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

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

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

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

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

Remark 7. Given T 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)

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

(2) or the best value discovered

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

(3) or the last value discovered

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

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

Remark 9. The parameter 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 8) often imply restrictions on the decay of $\{\eta_t\}$ which guide the practitioner to parametrize it properly. Some popular choices of learning rate η_t are:

- (1) constant; $\eta_t = \eta$, for where $\eta > 0$ is a small value. The rationale is that GD chain $\{w_t\}$ performs constant small steps towards the (local) minimum w_* and then oscillate around it.
- (2) decreasing and converging to zero; $\eta_t \searrow$ with $\lim_{t \rightarrow \infty} \eta_t = 0$. E.g. $\eta_t = \left(\frac{C}{t}\right)^\varsigma$ where $\varsigma \in [0.5, 1]$ and $C > 0$. The rationale is that GD algorithm starts by performing larger steps (controlled by C) at the beginning to explore the area for discovering possible minima. Also it reduces the size of those steps with the iterations (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 $\lim_{t \rightarrow \infty} \eta_t = \tau_*$ E.g. $\eta_t = \left(\frac{C}{t}\right)^\varsigma + \tau_*$ with $\varsigma \in (0.5, 1]$, $C > 0$, and $\tau_* \approx 0$. Same as previously, but the algorithm aims at oscillating around the detected local minimum.
- (4) constant until an iteration T_0 and then decreasing; Eg $\eta_t = \left(\frac{C}{\max(t, T_0)}\right)^\varsigma$ with $\varsigma \in [0.5, 1]$ and $C > 0$, and $T_0 < T$. The rationale is that at the first stage of the iterations (when $t \leq T_0$) the algorithm may need a constant large steps for a significant number of iterations T_0 in order to explore the domain; and hence in order for the chain $\{w_t\}$ to reach the area around the (local) minimum w_* . In the second stage, hoping that the chain $\{w_t\}$ may be in close proximity to the (local) minimum w_* the algorithm progressively performs smaller steps to converge towards the minimum w_* . The first stage ($t \leq T_0$) is called burn-in; the values $\{w_t\}$ produced during the burn-in ($t \leq T_0$) are often discarded/ignored from the output of the GD algorithm.

- Parameters $C, \varsigma, \tau_*, T_0$ may be chosen based on pilot runs against a small fraction of the training data set.

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

- (1) terminate when the gradient is sufficiently close to zero; i.e. if $\|\nabla f(w^{(t)})\| \leq \epsilon$ for some pre-specified tiny $\epsilon > 0$ then STOP
- (2) terminate when the chain $w^{(t)}$ does not change; i.e. if $\|w^{(t+1)} - w^{(t)}\| \leq \epsilon \|w^{(t)}\|$ for some pre-specified tiny $\epsilon > 0$ then STOP
- (3) terminate when a pre-specified number of iterations T is performed; i.e. if $t \geq T$ then STOP

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

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

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

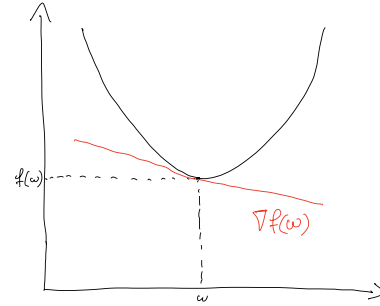
Definition 12. Vector v is called subgradient of a function $f : S \rightarrow \mathbb{R}$ at $w \in S$ if

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

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



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



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

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

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

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

3.1. Construction of subgradient.

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

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

- (1) *If function $f : S \rightarrow \mathbb{R}$ is differentiable at w then the only subgradient of f at w is the gradient $\nabla f(w)$, and (3.1) is equality; i.e. $\partial f(w) = \{\nabla f(w)\}$.*
- (2) *for constants 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 18. Consider the function $f : \mathbb{R} \rightarrow \mathbb{R}_+$ with $f(w) = |w| = \begin{cases} w & w \geq 0 \\ -w & w < 0 \end{cases}$. Find the set of subgradients $\partial f(w)$ for each $w \in \mathbb{R}$.

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

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

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

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

4. ANALYSIS OF GRADIENT DESCENT

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

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

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

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

Note 21. We consider Lemma 22 as a given Fact.

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

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

Proof. Omitted; see the Reference 1. □

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

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

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

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

Proof. It is¹

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

□

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

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

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

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

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

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

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

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

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

¹Jensen's inequality for convex $f(\cdot)$ is $f(\mathbb{E}(x)) \leq \mathbb{E}(f(x))$

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

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

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

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

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

where $\|\nabla f(\cdot)\| \leq \rho$.

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

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

Proposition 30. *(Cont Prop. 28) 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*

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

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

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

Proof. Part 1 is a simple substitution from Proposition 28, and part 2 is implied from part 1. \square

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

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

5. EXAMPLES ³

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

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

³Code is available in https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2024/tree/main/Lecture_handouts/code/04.Gradient_descent/example_1.R

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

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

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

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

The GD Algorithm 1 with learning rate η is

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

(1) compute

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

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

If $t \geq T$ then STOP

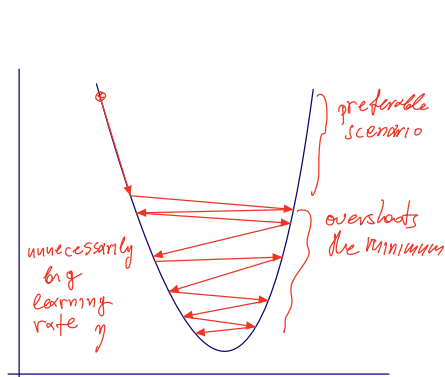
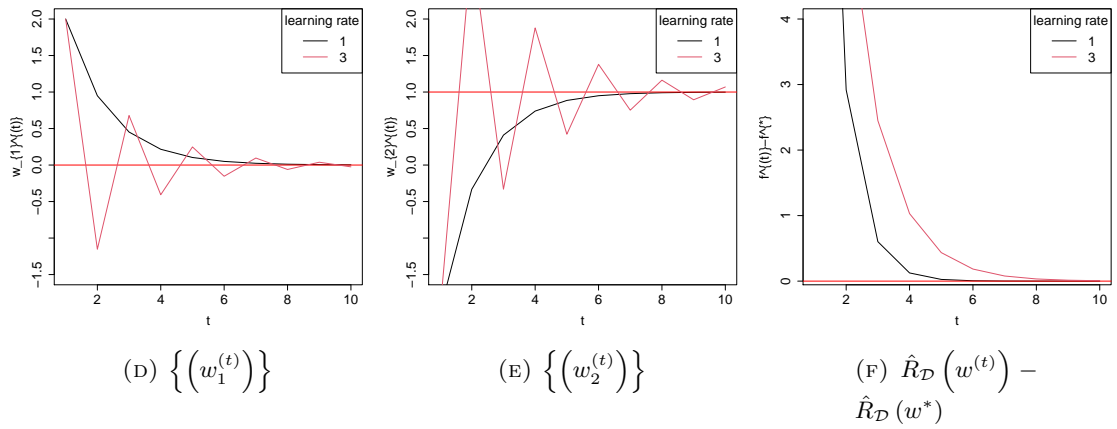
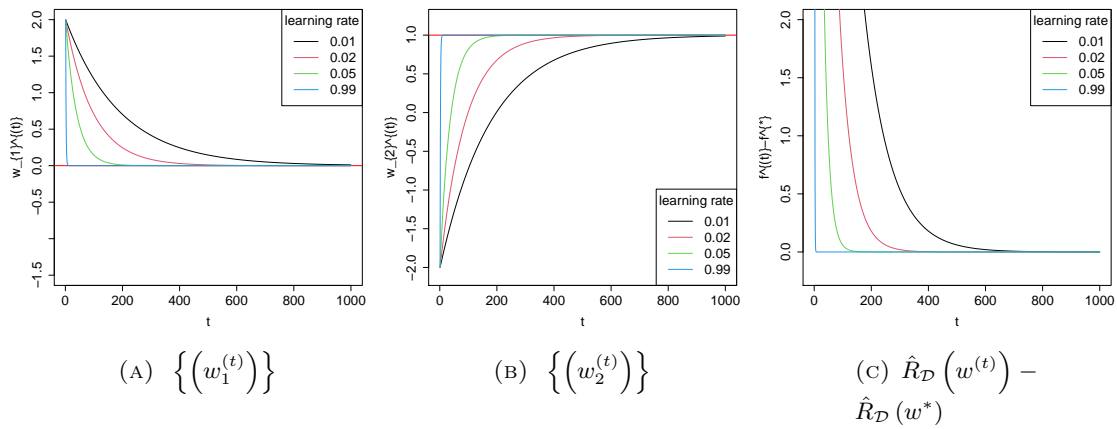
This is because $\hat{R}_{\mathcal{D}}(w)$ is differentiable in \mathbb{R}^2 so $\partial \hat{R}_{\mathcal{D}}(w) = \left\{ \nabla \hat{R}_{\mathcal{D}}(w) \right\}$ and because

$$\nabla \hat{R}_{\mathcal{D}}(w) = \begin{pmatrix} \frac{d}{dw_1} \hat{R}_{\mathcal{D}}(w) \\ \frac{d}{dw_2} \hat{R}_{\mathcal{D}}(w) \end{pmatrix} = \dots = \begin{pmatrix} 2w_1^{(t)} + 2w_2^{(t)} \bar{x} - 2\bar{y} \\ 2w_1^{(t)} \bar{x} + 2w_2^{(t)} \overline{x^2} - 2\bar{y}^\top x \end{pmatrix}$$

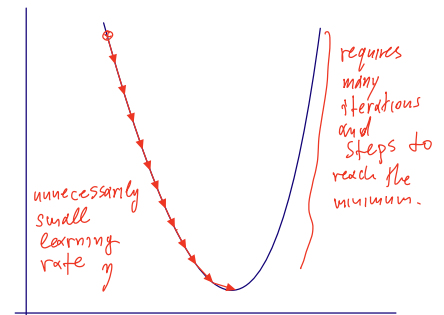
Consider data size $n = 100$, and parameters $\rho = 0.2$, $\sigma_y^2 = 1$ and $\sigma_x^2 = 1$. Then the real value (5.2) that I need to learn equals to $w^* = (0, 1)^\top$. Consider a GD seed $w_0 = (2, -2)$, and total number of iterations $T = 1000$.

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

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



(G) Unnecessarily large learning rate



(H) Unnecessarily small learning rate

FIGURE 5.1

Handout 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.
 - Further reading

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 $w^* \in \mathcal{H}$, where \mathcal{H} is a class of hypotheses, of the risk function (RF) $R(w) = \mathbb{E}_{z \sim g}(\ell(w, z))$ given an unknown data generating model $g(\cdot)$ and using a known tractable loss $\ell(\cdot, \cdot)$; that is

$$(1.1) \quad w^* = \arg \min_{w \in \mathcal{H}} (R_g(w)) = \arg \min_{w \in \mathcal{H}} (\mathbb{E}_{z \sim g}(\ell(w, z)))$$

Remark 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) involves an integral which may be computationally intractable. Instead it aims to minimize the ERF $\hat{R}(w) = \frac{1}{n} \sum_{i=1}^n \ell(w, z_i)$ which ideally is used as a proxy when data size n is big (big-data).

Remark 3. The implementation of GD may be computationally impractical even in problems where we need to minimize an ERF $\hat{R}_n(w)$ if we have big data ($n \approx \text{big}$). This is because GD requires the recursive computation of the exact gradient $\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla \ell(w, z_i)$ using all the data $\{z_i\}$ at each iteration. That may be too slow.

Remark 4. Stochastic gradient descent (SGD) aims at solving (1.1), and overcoming the issues in Remarks 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

2.1. Description.

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

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

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

Algorithm 6. *Stochastic Gradient Descent (SGD) with learning rate $\eta_t > 0$ for the solution of the minimization problem (2.1)*

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

(1) compute

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

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

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

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

Remark 7. If f is differentiable at $w^{(t)}$, it is $\partial f(w^{(t)}) = \{\nabla f(w^{(t)})\}$. Hence v_t is such as $\mathbb{E}(v_t | w^{(t)}) = \nabla f(w^{(t)})$ 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

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

where

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

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

Remark 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)

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

(2) or the best value discovered

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

(3) or the last value discovered

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

Note 10. SGD output converges to a local minimum, $w_{\text{SGD}}^{(T)} \rightarrow w_*$ (in some sense), under different sets of regularity conditions. Section 4 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) $\sum_{t=1}^{\infty} \eta_t^2 < \infty$

Remark 12. The popular learning rates $\{\eta_t\}$ in Remark 9 in Handout 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.

Remark 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 $t \rightarrow \infty$; if this was not the case then

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

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

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

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. STOCHASTIC GRADIENT DESCENT WITH PROJECTION

Remark 14. Consider the scenario in Problem 1 where the learning problem requires to discover w^* in the restricted/bounded set \mathcal{H} . Assume the function to be minimized is convex in the restricted hypothesis set \mathcal{H} , e.g. $\mathcal{H} = \{w : \|w\| \leq B\}$, but non-convex in \mathbb{R}^d . Direct implementation of vanilla SGD (Algorithm 6) may produce a chain stepping out \mathcal{H} and hence an output $w_{\text{SGD}} \notin \mathcal{H}$. SGD can be modified to address this issue, as in Algorithm 15, by including a projection step guaranteeing $w \in \mathcal{H}$.

Algorithm 15. *Stochastic Gradient Descent with learning rate $\eta_t > 0$ and with projection in \mathcal{H} for problem in (2.1)*

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

(1) compute

$$(3.1) \quad w^{(t+\frac{1}{2})} = w^{(t)} - \eta_t v_t,$$

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

(2) compute

$$(3.2) \quad w^{(t+1)} = \arg \min_{w \in \mathcal{H}} \left(\|w - w^{(t+\frac{1}{2})}\| \right)$$

(3) terminate if a termination criterion is satisfied

4. ANALYSIS OF SGD (ALGORITHM 6)

Note 16. Recall 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.

Theorem 17. *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 $w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$ satisfies*

$$(4.1) \quad E \left(f \left(w_{\text{GD}}^{(T)} \right) \right) - f(w^*) \leq \frac{\|w^*\|^2}{2\eta T} + \frac{\eta}{2} \frac{1}{T} \sum_{t=1}^T E \|v_t\|^2$$

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

$$(4.2) \quad E \left(f \left(w_{\text{GD}}^{(T)} \right) - f(w^*) \right) \leq E \left(\frac{1}{T} \sum_{t=1}^T \left(f(w^{(t)}) - f(w^*) \right) \right) = \frac{1}{T} \sum_{t=1}^T E \left(f(w^{(t)}) - f(w^*) \right)$$

I will try to use Lemma 22 from Handout 4, hence I need to show

$$(4.3) \quad E \left(f(w^{(t)}) - f(w^*) \right) \leq E \left(\langle w^{(t)} - w^*, v_t \rangle \right)$$

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

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

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

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

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

$$(4.4) \quad \begin{aligned} E_{v_{1:t-1}} \left(\langle w^{(t)} - w^*, E_{v_{1:t}}(v_t|v_{1:t-1}) \rangle \right) &\geq E_{v_{1:t-1}} \left(f(w^{(t)}) - f(w^*) \right) \\ &= E_{v_{1:T}} \left(f(w^{(t)}) - f(w^*) \right) \end{aligned}$$

Hence combining (4.4), (4.3), and (4.2)

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

Then Lemma 22 in Handout 4 “Gradient descent” implies

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

□

Remark 18. The upper bound in (4.1) depends on the variation of v_t as

$$(4.5) \quad E\|v_t\|^2 = \sum_{j=1}^d \text{Var}(v_{t,j}) + \sum_{j=1}^d (E(v_{t,j}))^2$$

where d is the dimension of $v_t = (v_{t,1}, \dots, v_{t,d})^\top$. The second term on the right hand side of (4.5) is constant as $v_{t,j}$ is the unbiased estimator of the sub-gradient by construction.

Proposition 19. (Cont. Theorem 17) Let $f(\cdot)$ be a convex function, and let $\mathcal{H} = \{w \in \mathbb{R} : \|w\| \leq B\}$. 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 $w_{\text{SGD}}^{(T)} = \frac{1}{T} \sum_{t=1}^T w^{(t)}$. Then

(1) upper bound on the sub-optimality is

$$(4.6) \quad E \left(f(w_{\text{SGD}}^{(T)}) \right) - f(w^*) \leq \frac{B\rho}{\sqrt{T}}$$

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

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

Proof. It follows from Proposition 17. □

Remark 20. Lemma 22 in Handout 4 holds even when projection steps are added in the vanilla SGD algorithm, hence the above analysis holds for the SGD with projections (Algorithm 15) too.

5. IMPLEMENTATION OF SGD IN THE LEARNING PROBLEM 1

Note 21. Proposition 22 allows a convenient implementation of SGD to the learning problem (Problem 1).

Proposition 22. For a randomly drawn example $z \sim g(\cdot)$, the sub-gradient v of $\ell(w, z)$ at point w is an unbiased estimator of the sub-gradient of the risk $R_g(w)$ at point w .

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

$$(5.1) \quad \ell(u, z) - \ell(w, z) \geq \langle u - w, v \rangle$$

It is

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

Hence, by definition, v is such that $\mathbb{E}_{z \sim g} (v | w)$ is a sub-gradient of $R_g(w)$. \square

Example 23. Consider that loss ℓ is differentiable wrt w . If $v = \nabla_w \ell(w, z)$, then

$$\mathbb{E}_{z \sim g} (v | w) = \mathbb{E}_{z \sim g} (\nabla_w \ell(w, z) | w) = \nabla_w \mathbb{E}_{z \sim g} (\ell(w, z) | w) = \nabla_w R_g(w)$$

Note 24. Below we show how SGD can be implement in the learning Problem 1.

Note 25. Assume there is available a finite dataset $\mathcal{S}_n = \{z_i; i = 1, \dots, n\}$ of size n which consists of independent realizations z_i from the data generating distribution g ; $z_i \stackrel{\text{ind}}{\sim} g$. Batch SGD (Algorithm 26) is an implementation of the SGD (Algorithm 6) in the learning Problem 1.

Algorithm 26. Batch Stochastic Gradient Descent with learning rate $\eta_t > 0$, and batch size m , for Problem 1.

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

(1) get a random sub-sample $\{\tilde{z}_j^{(t)}; j = 1, \dots, m\}$ of size m with or without replacement from the complete data-set \mathcal{S}_n .

(2) compute

$$(5.2) \quad w^{(t+1)} = w^{(t)} - \eta_t v_t,$$

where $v_t = \frac{1}{m} \sum_{j=1}^m \tilde{v}_{t,j}$ and $\tilde{v}_{t,j} \in \partial_w \ell(w^{(t)}, \tilde{z}_j^{(t)})$.

(3) terminate if a termination criterion is satisfied

Remark 27. Step 1 can be presented equivalently as

(1) Randomly generate a set $\mathcal{J}^{(t)} \subseteq \{1, \dots, n\}^m$ of m indices from 1 to n with or without replacement, and set a $\tilde{\mathcal{S}}_m = \{z_i; i \in \mathcal{J}^{(t)}\}$.

Hence $v_t = \frac{1}{m} \sum_{j \in \mathcal{J}^{(t)}} v_{t,j}$ where $\tilde{v}_{t,j} \in \partial_w \ell(w^{(t)}, z_j)$.

Remark 28. A projection step, such as (3.2) can be added right after step 2 in Algorithm 26, if needed.

Remark 29. 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 26 can become

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

Definition 30. Online Stochastic gradient descent is the special case of Algorithm 26 using sub-samples of size one ($m = 1$), namely, only one example is randomly chosen in Step 1 of Algorithm 26.

Remark 31. 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 (4.1); see Remark 18.

Remark 32. 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.

Remark 33. In practice, for a given fixed computational time, it is possible for a batch 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 vs online SGD 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.

6. STOCHASTIC VARIANCE REDUCED GRADIENT (SVRG)

Remark 34. Recall the upper bound of the error (4.1) in SGD depends on the variance of the stochastic gradient, as shown in (4.5) of Remark 18. Hence the algorithm may be improved by reducing the variance of each element of v_t .

Remark 35. Control variate is a general way to perform variance reduction. Let random variables $v \in \mathbb{R}$, and $y \in \mathbb{R}$. Let $z = v + c(y - E(y))$ for some constant $c \in \mathbb{R}$. It is $E_c(z) = E(v)$ and

$$\text{Var}_c(z) = \text{Var}(v) + c^2 \text{Var}(y) + 2c \text{Cov}(v, y)$$

which is minimized for

$$c^* = -\frac{\text{Cov}(v, y)}{\text{Var}(y)}$$

hence

$$\text{Var}_{c^*}(z) = \text{Var}(v) - \frac{(\text{Cov}(v, y))^2}{\text{Var}(y)}$$

Note 36. For simplicity, SVRG is presented in the case where $c = 1$ and the loss function $\ell(w, z)$ is differentiable wrt w at every z hence its gradient exists. However it is applicable to the more general cases introduced.

Remark 37. Every κ iterations, SVRG keeps a snapshot \tilde{w} , and computes the gradient using all data i.e. $\frac{1}{n} \sum_{i=1}^n \ell(\tilde{w}, z_i)$. At each iteration t , the update is

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

given that a random $\tilde{z}^{(t)}$ has been collected from the sample. The symbols below the brackets are given with reference to Remark 35.

Algorithm 38. *Stochastic Variance Reduced Gradient with learning rate $\eta_t > 0$ for Problem 1.*

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

- (1) randomly get an example $\tilde{z}^{(t)}$ from S_n .
- (2) compute

$$(6.1) \quad w^{(t+1)} = w^{(t)} - \eta_t \left[\nabla \ell(w^{(t)}, \tilde{z}^{(t)}) - \nabla \ell(\tilde{w}, \tilde{z}^{(t)}) + \frac{1}{n} \sum_{i=1}^n \nabla \ell(\tilde{w}, z_i) \right]$$

- (3) if modulo $(t, \kappa) = 0$, set $\tilde{w} = w^{(t)}$
- (4) if a termination criterion is satisfied STOP

Remark 39. Iterations of SVRG are computationally faster than those of full GD, but SVRG can still match the theoretical convergence rate of GD.

Remark 40. How often we get snapshots, aka κ , in Algorithm 38 is specified by the researcher. The smaller the κ , the more frequent snapshots, and the more correlated the baseline y will be with the objective x and hence the bigger the performance improvement; however the iterations will be slower.

7. PRECONDITIONED SGD; THE ADAGRAD ALGORITHM

Remark 41. All SGD (introduced earlier) are first order methods in the sense they consider only the gradient. Their advantage is each iteration is fast. The disadvantage is that they ignore the curvature of the space and hence can be slower to converge in cases the curvature changes eg. among dimensions of w .

Remark 42. To address this, SGD (Algorithm 6) can be modified in the update step as in Algorithm 43 by using a preconditioner P_t that accounts for the curvature (or geometry in general of f).

Algorithm 43. *Preconditioned Stochastic Gradient Descent with learning rate $\eta_t > 0$, and preconditioner P_t for the solution of the minimization problem (2.1)*

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

(1) compute

$$(7.1) \quad w^{(t+1)} = w^{(t)} - \eta_t P_t v_t,$$

where v_t is a random vector such that $E(v_t | w^{(t)}) \in \partial f(w^{(t)})$, and P_t is a preconditioner

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

If $t \geq T$ then STOP

Remark 44. A natural choice of P_t can be $P_t := [H_t + \epsilon I_d]^{-1}$, where H_t is the Hessian matrix $[H_t]_{i,j} = \frac{\partial^2}{\partial w_i \partial w_j} f(w) \Big|_{w=w^{(t)}}$ (ie. the gradient of the gradient's elements), and ϵ is a tiny $\epsilon > 0$ to mitigate machine error when Hessian elements are close to zero.

Remark 45. If the preconditioner P_t is set to be the inverse of the full Hessian, it may be too expensive to perform matrix operations in (7.1) with the full Hessian, and such operations can be too unstable/inaccurate due to the random error induced by the stochasticity of the gradient.

7.1. Adaptive Stochastic Gradient Decent (AdaGrad).

Remark 46. AdaGrad aims to dynamically incorporate knowledge of the geometry of function $f(\cdot)$ (to be minimized) in earlier iterations to perform more informative gradient-based learning.

Remark 47. AdaGrad aims to perform larger updates (i.e. high learning rates) for those dimensions of w that are related to infrequent features (largest partial derivative) and smaller updates (i.e. low learning rates) for frequent ones (smaller partial derivative).

Remark 48. Hence, this strategy often improves convergence performance over standard stochastic gradient descent in settings where the data are sparse and sparse features w 's are more informative.

Definition 49. Adaptive Stochastic Gradient Decent (AdaGrad) can be presented in terms of preconditioned SGD (Algorithm 43) with preconditioner $P_t = [\text{diag}(G_t) + \epsilon I_d]^{-1/2}$ in (7.1)

$$(7.2) \quad w^{(t+1)} = w^{(t)} - \eta_t [\text{diag}(G_t) + \epsilon I_d]^{-1/2} v_t,$$

where notation $\text{diag}(A)$ denotes a $d \times d$ matrix whose diagonal is the d dimensional diagonal vector $(A_{1,1}, A_{2,2}, \dots, A_{d,d})$ of $d \times d$ matrix A and whose off-diagonal elements are zero, $G_t = \sum_{\tau=1}^t v_\tau^\top v_\tau$ 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 in case the gradient becomes too close to zero.

Remark 50. AdaGrad algorithm individually adapts the learning rate of each dimension of w_t by scaling them inversely proportional to the square root of the sum of all the past squared values of the gradient $\{v_\tau; \tau \leq t\}$. This is because

$$(7.3) \quad [G_t]_{j,j} = \sum_{\tau=1}^t (v_{\tau,j})^2$$

where j denotes the j -th dimension of w . Hence (7.2) and (7.2) imply that the j -th dimension of w is updated as

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

Remark 51. The accumulation of positive terms in (7.3) makes the sum keep growing during training and causes the learning rate to shrink and becoming infinitesimally small. This offers an automatic way to choose a decreasing learning rate simplifying setting the learning rate; however it may result in a premature and excessive decrease in the effective learning rate. This can be mitigated by still considering in (7.2) a (user specified rate) $\eta_t \geq 0$ and tuning it properly via pilot runs.

8. EXAMPLE ¹

Note 52. We continue Example in Section 5 in Handout 4. Recall, we considered a hypothesis space \mathcal{H} of linear functions $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ with $h(w) = w_1 + w_2 x$, $w = (w_1, w_2)^\top$, and $\ell(w, z = (x, y)^\top) = (y_i - w_1 - w_2 x)^2$. Here we consider a big dataset $\mathcal{S}_n = \{z_1, \dots, z_n\}$ with $n = 10^6$ examples.

Example 53. The batch SGD algorithm (Algorithm 26) with learning rate η_t and batch size $m = 10$ is

- For $t = 1, 2, 3, \dots$ iterate:
 - (1) Randomly generate a set $\mathcal{J}^{(t)}$ by drawing $m = 10$ numbers from $\{1, \dots, n = 10^6\}$, and set $\tilde{\mathcal{S}}_m = \{z_i; i \in \mathcal{J}^{(t)}\}$
 - (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

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

In Figures 8.1a & 8.1d, 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 $\{w^{(t)}\}$.

Example 54. The SVRG with learning rate $\eta_t > 0$ and batch size $m = 1$ is

- For $t = 1, 2, 3, \dots$ iterate:
 - (1) randomly generate a set $\mathcal{J}^{(t)} = \{j^*\}$ by drawing one number j^* from $\{1, \dots, n = 10^6\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$

¹Code can be found in https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2023/tree/main/Lecture_handouts/code/05.Stochastic_gradient_descent

(2) compute

$$(8.1) \quad w^{(t+1)} = \begin{bmatrix} w_1^{(t)} \\ w_2^{(t)} \end{bmatrix} - \eta_t \left(\begin{bmatrix} 2(w_1^{(t)} - \tilde{w}_1^{(t)}) + 2(w_2^{(t)} - \tilde{w}_2^{(t)}) x_{j^*} \\ 2(w_2^{(t)} - \tilde{w}_2^{(t)}) x_{j^*} + 2(w_2^{(t)} - \tilde{w}_2^{(t)}) (x_{j^*})^2 \end{bmatrix} + \nabla \hat{R}_{\mathcal{D}}(\tilde{w}) \right)$$

(3) if modulo $(t, \kappa) = 0$,

(a) set $\tilde{w} = w^{(t)}$

(b) compute

$$(8.2) \quad \frac{1}{n} \sum_{i=1}^n \ell(\tilde{w}, z_i) = \begin{pmatrix} 2\tilde{w}_1^{(t)} + 2\tilde{w}_2^{(t)} \bar{x} - 2\bar{y} \\ 2\tilde{w}_1^{(t)} \bar{x} + 2\tilde{w}_2^{(t)} \bar{x}^2 - 2\bar{y}^\top x \end{pmatrix} = \nabla \hat{R}_{\mathcal{D}}(\tilde{w})$$

(4) if a termination criterion is satisfied STOP

Because (8.2) is actually the gradient of the Risk function at \tilde{w} and

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

In Figure 8.1c we observe the frequency of the snapshots has improved the convergence; however this is not a panacea as seen in Figure 8.1f.

Example 55. The AdaGrad with $\eta_t = 1$, $\epsilon = 10^{-6}$, and batch size $m = 1$ is

- Set $G_0 = (0, 0)^\top$.
- For $t = 1, 2, 3, \dots$ iterate:
 - (1) randomly generate a set $\mathcal{J}^{(t)} = \{j^*\}$ by drawing one number from $\{1, \dots, n = 10^6\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$
 - (2) compute

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

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

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

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

because

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

In Figures 8.1g & 8.1h, 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.

Example 56. Consider a (rather naive) loss function $\ell(w, z = (x, y)) = -\cos(0.5(y - w_1 - w_2x))$, a hypothesis class $\mathcal{H} = \{w \in \mathbb{R}^2 : \|w\| \leq 1.5\}$, and assume that inputs x in dataset \mathcal{D} 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 (batch size $m = 1$) with projection to \mathcal{H} as

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

- (1) randomly generate a set $\mathcal{J}^{(t)} = \{j^*\}$ by drawing one number from $\{1, \dots, n = 10^6\}$, and set $\tilde{\mathcal{S}}_1 = \{z_{j^*}\}$
- (2) compute

$$w^{(t+1/2)} = w^{(t)} - \frac{50}{t} \begin{pmatrix} \sin(y_{j^*} - w_1^{(t)} - w_2^{(t)}x_{j^*}) \\ \sin(y_{j^*} - w_1^{(t)} - w_2^{(t)}x_{j^*})x_{j^*} \end{pmatrix}$$

$$w^{(t+1/2)} = \arg \min_{\|w\| \leq 1.5} (\|w - w^{(t+1/2)}\|)$$

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

because

$$v_t = \nabla \ell(w^{(t)}, z_{j^*}) = \begin{pmatrix} \frac{\partial}{\partial w_1^{(t)}} \ell(w^{(t)}, z_{j^*}) \\ \frac{\partial}{\partial w_2^{(t)}} \ell(w^{(t)}, z_{j^*}) \end{pmatrix} = \begin{pmatrix} 0.5 \sin(0.5(y_{j^*} - w_1^{(t)} - w_2^{(t)}x_{j^*})) \\ 0.5 \sin(0.5(y_{j^*} - w_1^{(t)} - w_2^{(t)}x_{j^*}))x_{j^*} \end{pmatrix}$$

In Figures 8.1b & 8.1e, 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.



(A) batch SGD Example 53 w_1



(B) online SGD with projection Example 56 w_1



(C) SVRG Example 54 w_1



(D) batch SGD Example 53 w_2



(E) online SGD with projection Example 56 w_2



(F) SVRG Example 54 w_2



(G) online AdaGrad Example 55 w_1



(H) online AdaGrad Example 55 w_2

FIGURE 8.1. Simulations of the Examples