Machine Learning and Neural Networks III (MATH3431)

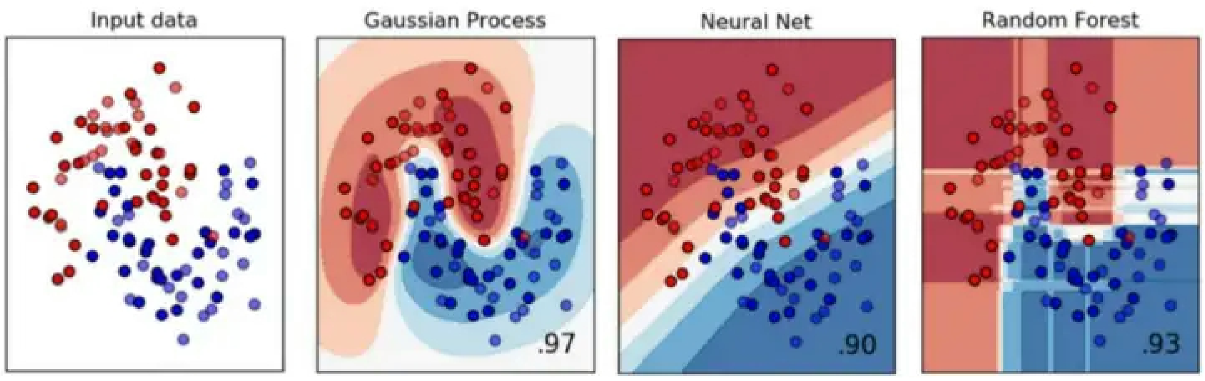
Epiphany term

Georgios P. Karagiannis

georgios.karagiannis@durham.ac.uk

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

2023/01/12 at 10:02:29



Reading list

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

Main texts:

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 It is very comprehensive (compared to others) and it goes deep enough for the module although it may be a bit outdated.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

Supplementary textbooks:

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

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

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

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

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

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

Reading list & references:

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

1. Introductions and loose definitions

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

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

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

2. (loose) Notation in learning

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

then the risk is

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

whose minimizer is

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

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

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

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

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

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

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

Handout 1: Elements of convex learning problems

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

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

Reading list & references:

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

1. Motivations

Note 1. Introducing convexity and smoothness in the learning problems makes easier the (theoretical) analysis of the problem and its solution.

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

Note 3. Extensions of handle non-convex problems as below can be done via surrogates –to be discussed.

2. Convex learning problem

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

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

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

or

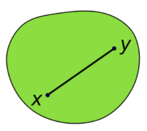
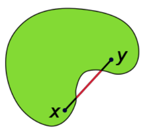
$$w^{**}=\arg\min_{\forall w}\frac{1}{m}\sum_{i=1}^{m}\left(<w,x_{i}>-y\right)^{2}$$

is a convex learning problem for arguments discussed below.

3. Convexity

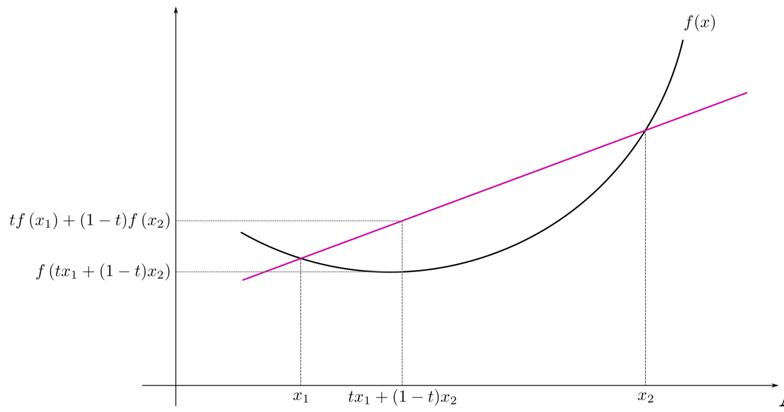
Definition 6. A set C is convex if for any $u,v\in C$, the line segment between u and v is contained in C. Namely,

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


A convex set

A non-convex set

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

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



A convex function

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

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

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

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

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

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

Proof. See Exercise 1 in the Exercise sheet. □

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

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

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

Solution.

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

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

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

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

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

4. Lipschitzbness

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

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

Note 16. So a Lipschitz function $f(x)$ cannot change too drastically wrt x.

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

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

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

Solution.

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

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

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

Solution. See Exercise 2 from the exercise

Example 19. Let functions g be ρ-Lipschitz and g_2 has $g_2(x) =< v, x > +b$. Then f with $f(x) = g(< v, x > +b)$ is ρ-Lipschitz.

$$|f(w_1) - f(w_2)| = |g_1(< v, x > +b) - g_1(< v, x > +b)| \leq \rho_1 |< v, w_1 > +b- < v, w_2 > -b|$$
$$\leq \rho_1 \left| v^\top w_1 - v^\top w_2 \right| = \rho_1 |v| |w_1 - w_2|$$

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

5. Smoothness

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

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

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

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

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

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

holds. Hence if both conditions imply upper and lower bounds

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

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

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

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

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

which provide assumptions to bound the gradient.

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

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

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

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

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

6. Non-convex learning problems (treatments)

Remark 27. The loss function of a learning problem may be non-convex. A proper treatment would be to upper bound the non-convex loss function by a convex surrogate loss function.

Example 28. consider the problem of learning $w \in \mathcal{H}$ from hypothesis set $\mathcal{H}$ with respect to the $0-1$ loss

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

which is non-convex. A convex surrogate loss function can be

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

which is convex (Example 14) wrt w. Note that $\max(\cdot)$ is convex as $\max(1, \alpha u + (1-\alpha)v) \leq \alpha \max(1,u) + (1-\alpha)\max(1,v)$.

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

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

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.

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

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

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

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

Reading list & references:

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

1. Introductions and loose definitions

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

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

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

2. (loose) Notation in learning

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

then the risk is

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

whose minimizer is

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

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

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

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

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

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

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

Handout 1: 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.

**This handout is still Under construction

1. Motivations

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

$$w^* = \arg\min_{\forall h \in \mathcal{H}} \left(\hat{R}(h)\right) \tag{1.1}$$

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

2. Description

Notation 2. For the sake of simplicity and generalization, we will present Gradient descent in the following minimization problem

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

where here $f : \mathbb{R}^d \to \mathbb{R}$, and $w \in \mathcal{H} \subseteq \mathbb{R}^d$. Eg, f can be an ERF.

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

Definition 4. The Gradient descent algorithm for the solution of the minimization problem 2.1 is in Algorithm 1

Algorithm 1 Gradient descent algorithm

```
• For $t = 1, 2, 3, ...T$, iterate:
      $w^{(t+1)} = w^{(t)} + \eta_t \nabla f\left(w^{(t)}\right)$
```

where

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

is the gradient of f at .

Remark 5. As a motivation, consider the 2nd-Taylor polinomial for the approximation of $f(w)$ around for u in a small area around w

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

Assuming convexity for f, it is

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

meaning that P lower bounds f. Hence we could design an update mechanism producing $w^{(t+1)}$ that is nearby $w^{(t)}$ (small steps) and

$$f(w) \approx f\left(w^{(t)}\right) + \langle w - w^{(t)}, \nabla f\left(w^{(t+1)}\right) \rangle. \tag{2.2}$$

Hence we could recursively minimize the approximation (2.2) and the distance between the current state $w^{(t)}$ and the next w value to produce $w^{(t+1)}$; namely

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

where parameter $\eta > 0$ controls the trade off.

Remark 6. The output of GD can be (but not a exclusively), the average

$$w_{\text{GD}}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} f\left(w^{(t)}\right)$$

after discarding the first few iterations of $w^{(t)}$ for stability reasons, or the best value discovered

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

or the last value discovered

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

Note 7. GD algorithm converges to a local minimum, $w_{\text{GD}}^{(T)} \to w_*$, under different sets of regularity conditions (some weaker other stronger). In Section 4 perform an analysis.

Remark 8. The parameter η_t is called learning rate, step size, or gain. $\{\eta_t\}$ is a non-negative sequence and it is chosen by the practitioner. Regularity conditions (Note 7) often imply restrictions on the decay of $\{\eta_t\}$ which guide the practitioner to parametrise it properly. Some popular choices of learning rate $\eta_t$are:

(1) constant; $\eta_t = \eta$, where where η is a small value. The rational 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; e.g. $\eta_t = \left(\frac{C}{t}\right)^{\varsigma}$ where $\varsigma \in (0.5, 1]$ and $C > 0$. The rational is that GD algorithm at the begining starts by performing larger step to explore the area

for discovering possible minima while reducing the size of the steps with the iterations to converge to a possible w_* value.

(3) decreasing and converging to a tiny value τ_*; e.g. $\eta_t = \left(\frac{C}{t}\right)^{\varsigma} + \tau_*$ where $\varsigma \in (0.5, 1]$, $C > 0$, 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 $\eta_t = \left(\frac{C}{\max(t, T_0)}\right)^{\varsigma}$, where $\varsigma \in (0.5, 1]$ and $C > 0$. The rational is that in the first stage of the iterations the algorithm may need constant larger stems for a significant number of iterations in order to explore the domain and hence the chain $\{w_t\}$ to reach the area around the (local) minimum w_*. In the second stage the chain $\{w_t\}$ may be in a close proximity to the (local) minimum w_* and hence the algorithm needs to perform smaller steps to exploit (converge to w_*). The first stage is called burn-in and it is discarded from the output of the GD algorithm.

Parameters C, ς, τ_* may be chosen based on pilot runs. A set of conditions for $\{\eta_t\}$ are those from Robbins–Monro algorithm

$$\sum_{t=1}^{\infty} \eta_t = \infty \text{ and } \sum_{t=1}^{\infty} \eta_t^2 < \infty$$

(to be seen and discussed later), satisfied by items 2 and 4.

3. Examples

4. Analysis

5. Subgradients