Lecture Notes

MATHEMATICS FOR MACHINE LEARNING

This version: October 30, 2025

Latest version: github.com/felipe-tobar/Maths-for-ML

Felipe Tobar
Department of Mathematics
Imperial College London

f.tobar@imperial.ac.uk
www.ma.ic.ac.uk/~ft410

Contents

1 Introduction

MISSING

2 Optimisation

NB: in this chapter, we follow (Murphy, 2022).

Optimisation is central to ML, since models are *trained* by minimising a loss function (or optimising a reward function). In general, model design involves the definition of a training objective, that is, a function that denotes how good a model is. This training objective is a function of the training data and a model, the latter usually represented by its parameters. The best model is is the chosen by optimising this function.

Example: Linear regression (LR)

In the LR setting, we aim to determine the function

$$\begin{aligned} f &: \mathbb{R}^M \to \mathbb{R} \\ x &\mapsto f(x) = a^\top x + b, \quad a \in \mathbb{R}^M, b \in \mathbb{R} \end{aligned} \tag{2.1}$$

conditional to a set of observations

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^M \times \mathbb{R}. \tag{2.2}$$

Using least squares, the function f is chosen via minimisation of the sum of the square differences between observations $\{y_i\}_{i=1}^N$ and predictions $\{f(x_i)\}_{i=1}^N$. That is, we aim to minimise he loss:

$$J(\mathcal{D}, f) = \sum_{i=1}^N (y_i - f(x_i))^2 = \sum_{i=1}^N (y_i - a^\top x_i - b)^2. \tag{2.3}$$

[TODO: Generate figure: Check fig 1 ML lecture notes]

Example: Logistic regression

Here, we aim to determine the function

$$\begin{aligned} f &: \mathbb{R}^M \to \mathbb{R} \\ x &\mapsto f(x) = \frac{1}{1 + e^{-\theta^\top x + b}}, \quad \theta \in \mathbb{R}^M, b \in \mathbb{R} \end{aligned} \tag{2.4}$$

conditional to the observations

$$\mathcal{D} = \{(x_i, c_i)\}_{i=1}^N \subset \mathbb{R}^M \times \{0, 1\}. \tag{2.5}$$

The standard loss function for the classification problem is the cross entropy, given by:

$$J(\mathcal{D}, f) = -\frac{1}{N} \sum_{i=1}^N \left(c_i \log f(x_i) + (1 - c_i) \log(1 - f(x_i))\right) \tag{2.6}$$

$$= \frac{1}{N} \sum_{i=1}^N \Big(\log(1 + e^{-\theta^\top x + b}) - y_i(-\theta^\top x + b)\Big) \tag{2.7}$$

[TODO: Generate figure]

Example: Clustering (K-means)

Given a set of observations

$$\mathcal{D} = \{x_i\}_{i=1}^{N} \subset \mathbb{R}^M, \tag{2.8}$$

we aim to find cluster centres (or prototypes) $\mu_1, \mu_2, \ldots, \mu_K$ and *assignment variables* $\{r_{ik}\}_{i,k=1}^{N,K}$, to minimise the following loss

$$J(\mathcal{D}, f) = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||x_i - \mu_k||^2 \tag{2.9}$$

$$\tag{2.10}$$

[TODO: Generate figure]

2.1 Terminology

We denote an optimisation problem as follows:

$$\min_{x \in \mathcal{X}} f(x) \quad \text{s.t.} \quad g_i(x) \leq 0, \; h_j(x) = 0, \; i = 1, \ldots, I, \; j = 1, \ldots, J. \tag{2.11}$$

We describe the components of this statement in detail:

- **Objective function:** The function $f : \mathcal{X} \to \mathbb{R}$ is the quantity to be minimised, with respect to x.
- **Optimisation variable:** Minimising f requires fining the value of x such that $f(x)$ is minimum. This is also written as

$$x_\star = \arg\min_{x \in \mathcal{X}} f(x) \quad \text{s.t.} \quad g_i(x) \leq 0, \; h_j(x) = 0. \tag{2.12}$$

- **Restrictions:** These are denoted by the functions g_i and h_i above, which describe the requirements for the optimiser in the form of equalities and inequalities, respectively.
- **Feasible region:** This is the subset of the domain that complies with the restrictions, that is

$$C = \{x \in \mathcal{X}, \quad \text{s.t.} \quad g_i(x) \leq 0, \; h_j(x) = 0, \; i = 1, \ldots, I, \; j = 1, \ldots, J\} \tag{2.13}$$

- **Local / global optima.** Values for the optimisation variable that solve the optimisation problem wither locally or globally. More formally:

$$x_\star \text{ is a local optima} \iff \exists \lambda > 0 \quad \text{s.t.} \quad x_\star = \arg\min_{\substack{x \in \mathcal{X} \\ \text{s.t.} \; ||x - x_\star|| \leq \lambda}} f(x). \tag{2.14}$$

$$x_\star \text{ is a global optima} \iff x_\star = \arg\min_{x \in \mathcal{X}} f(x). \tag{2.15}$$

Interplay between constrains and local/global optima

[TODO: generate figure, how different restrictions change the number and type of optima]

Example: XXX

[TODO: Show a few parametric functions and indicate their (closed-form) minima]

2.2 Continuous unconstrained optimisation

We will ignore constrains in this section, and we will focus on problems of the form

$$\theta \in \arg\min_{\theta \in \Theta} L(\theta). \tag{2.16}$$

We emphasise that if $\theta_\star$ satisfies the above, then

$$\forall \theta \in \Theta,\ L(\theta_\star) \leq L(\theta), \tag{2.17}$$

meaning that it is a **global** optimum. However, as this might be very hard to find, we are also interested in local optima, that is, $\theta_\star$ such that

$$\exists\, \delta > 0\ \ \forall \theta \in \Theta\ \text{ s.t. }\ \|\theta - \theta_\star\| < \delta\ \Rightarrow\ L(\theta_\star) \leq L(\theta). \tag{2.18}$$

We now review the optimality conditions.

Assumption 2.1. The loss function L is twice differentiable.

Denoting $g(\theta) = \nabla_\theta L(\theta)$ and $H(\theta) = \nabla^2_\theta L(\theta)$, we can state the following optimality conditions.

- **First order necessary condition:** If $\theta_\star$ is a local minimum, then
  - $\nabla_\theta L(\theta_\star) = 0$
- **Second order necessary condition:** If $\theta_\star$ is a local minimum, then
  - $\nabla_\theta L(\theta_\star) = 0$
  - $\nabla^2_\theta L(\theta_\star)$ is positive semidefinite
- **Second order sufficient condition:** If $\theta_\star$ is a local minimum if and only if
  - $\nabla_\theta L(\theta_\star) = 0$
  - $\nabla^2_\theta L(\theta_\star)$ is positive definite

Example: different stationary points

Let us consider the function

$$\begin{aligned} f &: \mathbb{R}^2 \to \mathbb{R} \\ & x \mapsto f(x) = (p-1)x^2 + (p+1)y^2, \quad p \in \mathbb{R} \end{aligned} \tag{2.19}$$

Observe that

$$\nabla f = \begin{bmatrix} 2(p-1)x \\ 2(p+1)y \end{bmatrix}, \tag{2.20}$$

meaning that the only stationary points is $(x, y) = (0, 0)$. Furthermore,

$$\nabla^2 f = \begin{bmatrix} 2(p-1) & 0 \\ 0 & 2(p+1) \end{bmatrix}, \tag{2.21}$$

where we have 3 possible cases:

- $p > 1$: The stationary point is a minimum
- $-1 < p < 1$: The stationary point is a *saddle point*
- $p < -1$: The stationary point is a maximum

[TODO: generate figure for all three cases, discuss case $|p| = 1$]

2.3 Convex optimisation

This setting is defined by having a convex objective function and a convex feasible region. Critically, in the setting of convex optimisation a local minimum (according to the first/second order conditions presented above) is a global minimum. We next formally provide the relevant definitions.

Definition 2.1 (Convex set)**.** $\mathcal{S}$ is a convex set if $\forall x, x' \in \mathcal{S}$, we have:

$$\lambda x + (1-\lambda)x' \in \mathcal{S}, \quad \forall\, \lambda \in [0, 1]. \tag{2.22}$$

[TODO: Generate figures for convex and non-convex sets]

Definition 2.2 (Epigraph of a function)**.** The epigraph of a function $f : \mathcal{X} \to \mathbb{R}$ is the set defined by the region above the graph of the function, that is,

$$\text{epi}(f) = \{\, (x, t) \in \mathcal{X} \times \mathbb{R} \mid f(x) \leq t \,\}. \tag{2.23}$$

Definition 2.3 (Convex function)**.** f is a convex function if its epigraph is convex. Equivalently, f is convex is it is supported on a convex set and $\forall x, x' \in \mathcal{X}$

$$f\big(\lambda x + (1-\lambda)x'\big) \;\leq\; \lambda f(x) + (1-\lambda) f(x'), \quad \forall\, \lambda \in [0, 1]. \tag{2.24}$$

Furthermore, is the inequality is strict, we say that the function is **strictly convex**.

Example: Convex functions (in 1D)

The following are convex function from $\mathbb{R}$ to $\mathbb{R}$:

- $f(x) = x^2$
- $f(x) = e^{ax}$, $a \in \mathbb{R}$
- $f(x) = -\log x$
- $f(x) = x^a$, $a > 1$, $x > 0$
- $f(x) = |x|^a$, $a \geq 1$
- $f(x) = x \log x$, $x > 0$

[TODO: Generate figures, indicate epigraph (for convex and non-convex functions]

We now review some important results in convex optimisation

Proposition 2.1. Consider $f : \mathcal{X} \subset \mathbb{R} \to \mathbb{R}$ differentiable. We have that if $f'(x) \geq 0 \forall x \in \mathbb{R}$, f is non-decreasing

Proof. By the fundamental theorem of calculus, we have that for $a, b \in \mathbb{R}, a < b$,

$$f(b) - f(a) = \int_a^b f'(x)dx, \tag{2.25}$$

since $f'(x) \geq 0, \forall x \in [a, b]$, we have $\int_a^b f'(x)dx \geq 0$, therefore $f(b) \geq f(a)$, which means that f is non-decreasing. ■

Proposition 2.2. Consider $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ differentiable. The direction of maximum growth of f at x_0 is along its gradient $\nabla f(x_0)$

Proof. Let us consider $x' = x_0 + \rho u$, where $u \in \mathcal{X}, ||u|| = 1$, and $\rho > 0$ is a small constant. We find the maximum growth direction by maximising $f(x') - f(x_0)$ with respect to u. We consider the Taylor expansion

$$f(x') = f(x_0) + \nabla f(x_0)\rho u + \mathcal{O}(\rho^2), \tag{2.26}$$

and thus conclude that $f(x') - f(x_0) \simeq \nabla f(x_0)\rho u$, meaning that the maximum growth can be achieved by choosing u parallel to $\nabla f(x_0)$. That is, $\nabla f(x_0)$ is the direction of maximum growth for f at x_0. ■

Teorema 2.1. *Suppose $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ twice differentiable, then f is convex if and only if ∇^2 is positive semi definite.*

Proof. We consider $d = 1$. Using the FTC,

$$f'(b) - f'(a) = \int_a^b f''(x)dx \geq 0, \tag{2.27}$$

which implies that f' is non-decreasing. Therefore (using FTC again),

$$f(b) - f(a) = \int_a^b f'(x)dx \geq (b - a)f'(a), \tag{2.28}$$

equivalently,

$$f(b) \geq f(a)'(b - a)f'(a), \tag{2.29}$$

meaning that the function f *is always above its tangent.* Evaluating (2.29) for (a, z) and (b, z), where $z = (1 - t)a + tb$, we have

$$f(z) \geq f(a) + (z - a)f'(a) \tag{2.30}$$
$$f(z) \geq f(b) + (z - b)f'(b). \tag{2.31}$$

Then, multiplying the above equations by $(1 - t)$ and t respectively and summing them, we obtain:

$$f(z) \geq (1 - t)f(a) + tf(b) + (1 - t)(tb - ta)f'(a) + t[(1 - t)a - (1 - t)b]f'(b) \tag{2.32}$$
$$= (1 - t)f(a) + tf(b) + (1 - t)t(b - a)[f'(a) - f'(b)] \tag{2.33}$$
$$\geq (1 - t)f(a) + tf(b) \tag{2.34}$$

■

Example: Explore some functions

[TODO: Choose some functions, compute the derivative and Hessian, analyse them]

2.4 First order methods

In general, finding a minimum by setting $\nabla f(x) = 0$ and solving for x is not possible. For that reason, we will consider iterative methods based on gradients.

The idea here is to go *downhill* following the gradient towards the minimum (ignoring the curvature information for now).

We will specify a starting point x_0 and calculate

$$x_{t+1} = x_t + \eta_t d_t, \tag{2.35}$$

where η_t is a *step size* and d_t is a *descent direction*, such as $-\nabla f$. Here, the subindex $\cdot_t$ represents the iteration number (starting from iteration $t = 0$). We iterate until convergence, that is, until the elements in the sequence $x_t, x_{t+1}, x_{t+2}, \ldots$ become constant (or very similar). If convergence is achieved, we will assume the minimum has been found.

Note that there are several *descent directions*, that is, directions d_t such that

$$L(x_t + \eta_t d_t) \leq L(x_t). \tag{2.36}$$

In fact, as long as $d_t^\top \nabla f \leq 0$, d_t is a descent direction. Clearly, choosing $d_t = -\nabla f(x_t)$ is the *steepest descent direction*.

2.4.1 Role of the step size

The step size η_t is also known as *learning rate*. Furthermore, we refer to the set $\{\eta_1, \eta_2, \ldots\}$ as the learning rate schedule.

We will usually consider a constant learning rate, that is, $\eta_t = \eta$, $\forall t \in \mathbb{N}$. Though this is the simplest choice, there are some concerns to this choice: if η is too large, the iteration may fail to converge; whereas if it is too small, it may not converge at all.

Example: convergence for a parabola

[TODO: Plot the level sets of a parabola (or another convex function). Show how the steepest descent converges/diverges for different learning rates]

The learning rate is usually tuned with heuristics. Since we will usually implement

$$x_{t+1} = x_t + \eta \nabla f(x_t), \tag{2.37}$$

we will usually set $\eta < ||\nabla f||^{-1}$, as this will result in a stable autoregressive system for the sequence x_t.

2.4.2 Momentum

In higher dimensions, we want to move faster in some directions and slow in other directions, depending on the value of the gradient in each coordinate. This can be achieved by:

$$m_t = \beta m_{t-1} + \nabla f(x_{t-1}) \tag{2.38}$$

$$x_t = x_{t-1} - \nabla_t m_t, \tag{2.39}$$

where m_t is a smoothed version of the gradient, and $\beta \in [0, 1]$ is a design (memory) parameter. This way, previous values of the gradient have effect on future updates: if a particular coordinate of the gradient is consistently large, then that coordinate will receive updated of a higher magnitude. This is particularly useful when the evaluation of the gradient is noisy.

2.4.3 Newton method

Newton's method is

$$x_{t+1} = x_t - \eta_t H_t^{-1} \nabla f(x_t), \tag{2.40}$$

where recall that $H_t = \nabla^2 f(x_t)$ denotes the Hessian of f at x_t. This update follows from considering the second order approximation of the loss function arond the current point, that is:

$$L(x) \simeq L(x_t) + (\nabla f(x_t))^\top (x - x_t) + \frac{1}{2}(x - x_t)^\top H_t (x - x_t), \tag{2.41}$$

the minimum of which is given by

$$x_\star = x_t - H_t^{-1} \nabla L(x_t), \tag{2.42}$$

where the learning rate can also be used.

Example: convergence for a parabola (2)

[TODO: Same a previous example, but with momentum and/or Newton]

2.5 Stochastic gradient descent

We now consider stochastic optimisation, where

$$L(x) = \mathbb{E}_{q(z)} L(x, z), \tag{2.43}$$

that is, when the loss function is random and we aim to minimise its expected value.

For instance, in linear regression we have $L(\theta) = \mathbb{E}_{q(y|x)}(y - \theta^\top x)$.

In practice we are unable to compute this expectation since we don't know the law $q(z)$. However, since we usually have samples of q, we can do a sample approximation og the expectation. In fact, we will consider

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{n} L(x, z_i). \tag{2.44}$$

In the linear regression example, this would be $L(\theta) = \frac{1}{N} \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$.

The gradient is then also approximated using a batch of, say, B samples. That is,

$$\nabla L(\theta) \simeq \frac{1}{N} \sum_{i=1}^{B} \nabla L(x, z_i) \tag{2.45}$$

Example: random loss function for linear regression

[TODO: Plot a q function: its theoretical value and sample approximations using a finite set of datapoints]

Recall that in general we will consider parameter updates of the form

$$\theta_{t+1} = \theta_t - M_t^{-1} g_t \tag{2.46}$$

where M_t is an estimate of the norm of the gradient g_t, as this ensures convergence (or prevents divergence). We next explore some different choices of this estimate.

2.5.1 Adagrad

This considers the following iterative rule (d denotes the coordinate):

$$\theta_{t+1,d} = \theta_{t,d} - \eta_t \frac{1}{\sqrt{s_{t,d} + \epsilon}} g_{t,d}, \tag{2.47}$$

where $s_{t,d} = \sum_{i=1}^{t} g_{i,d}^2$. This can also be expressed in vector format as

$$\Delta\theta_t = -\eta_t \frac{1}{s_t} g_t. \tag{2.48}$$

The aim of this update rule is that control the magnitude of the step size for each coordinate of the parameter independently, based on a rolling estimate using previous gradient evaluations. Notice that the computation of $s_{t,d}$ assigns the same importance to all gradient evaluations in time.

2.5.2 RMSProp

Adagrad might fail to represent the current gradient magnitude by not emphasising current evaluations of the gradient. This can be solved by replacing the sum in the computation of $s_{t,d}$ by a moving average. That is,

$$s_{t+1,d} = \beta s_{t,d} + (1 - \beta) g_{t,d}^2, \tag{2.49}$$

where the *forgetting factor* β is usually chosen closer to 0.9. Then, the update rule also follows

$$\Delta\theta_t = -\eta_t \frac{1}{s_t} g_t. \tag{2.50}$$

2.5.3 Adam

The de facto optimiser in deep learning is Adam (Adaptive Moment Estimation), which combines RMSProp and momentum. Adam's update follows:

$$m_t = \beta_m m_{t-1} + (1 - \beta_m) g_t \tag{2.51}$$

$$s_t = \beta_s s_{t-1} + (1 - \beta_s) g_t^2. \tag{2.52}$$

Then,

$$\theta_t = \theta_{t-1} - \eta_t \frac{m_t}{\sqrt{s_t} + \epsilon} \tag{2.53}$$

Example: different optimisers

[TODO: If possible, show different optimisers in a toy example]

References

Murphy, K. P. (2022). *Probabilistic machine learning: An introduction.* MIT Press. Retrieved from `probml.ai`