Lecture Notes

MATHEMATICS FOR MACHINE LEARNING

This version: November 11, 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) and (?, ?). For a deeper presentation of the topics covered here, also see (?, ?) and (?, ?).

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 or **loss**, that is, a function that denotes **how well a model fits the data**. This training objective is a function of the training data and a chosen model, the latter usually represented by its parameters. The best model is thus chosen by optimising the loss function.

Example: Linear regression (LR)

In the LR setting, we aim to determine the function

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

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 the loss:

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

We show an example of a linear model learnt from data in Figure 1.

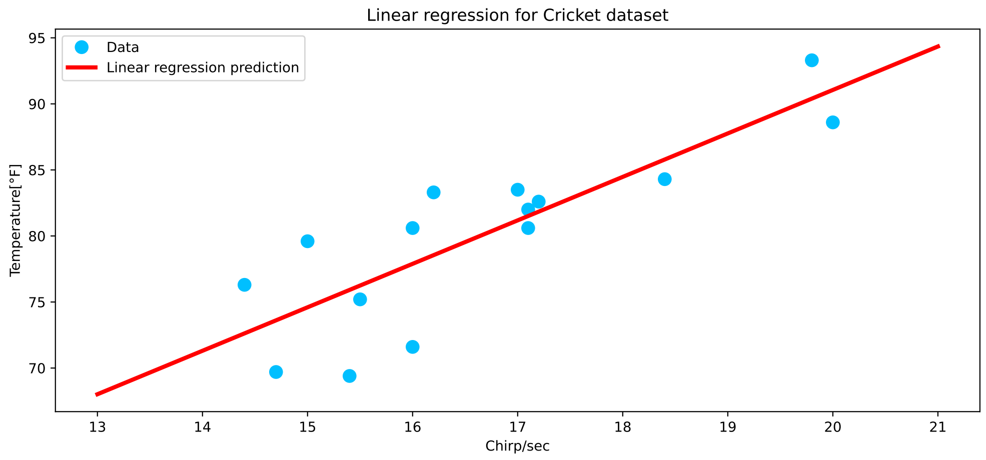

Fig. 1. Example of a linear regression model minimising the least squares loss.

Example: Logistic regression

Here, we aim to determine the function

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

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:

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

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

Figure 2 shows an example of a binary classification task minimising the cross entropy to separate data generated from two Gaussian distributions.

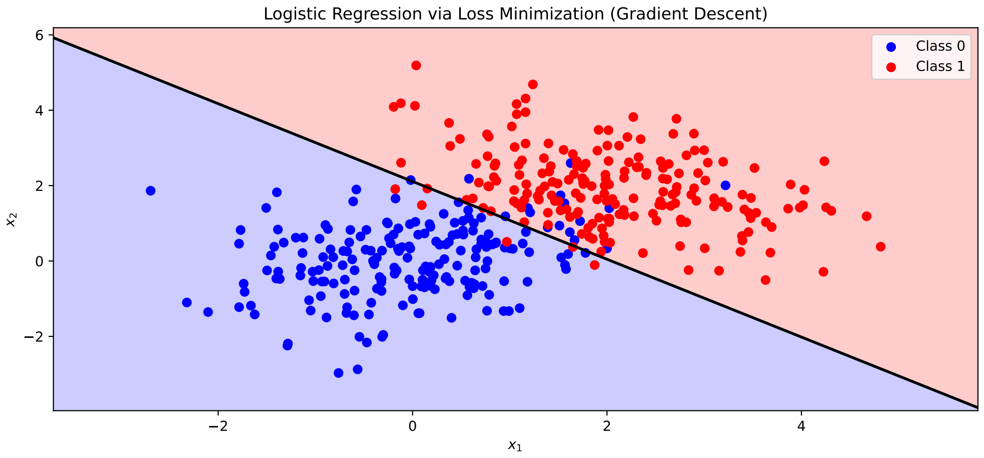

Fig. 2. Example of a logistic regression model to classify data from two Gaussians.

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

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

An example of a K-means model after the loss minimisation is shown on Figure 3

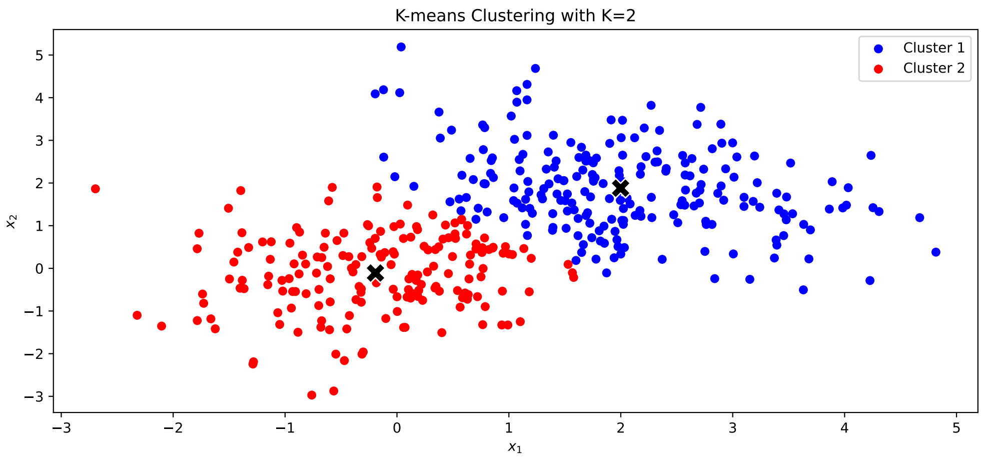

Fig. 3. Implementation of the K-means method to clustering data from two Gaussians. The figure shows the learnt centroids (black cross) and the colours correspond to the cluster assignments.

2.1 Terminology

We denote an optimisation problem as follows:

$$\min_{x \in \mathcal{X}} f(x), \quad \text{s.t.,} \quad g_i(x) \leq 0, \; h_j(x) = 0, \; i = 1, \ldots, I, \; j = 1, \ldots, J. \tag{2.10}$$

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 finding the value of x, such that $f(x)$ is minimum. This is also written as

$$x_\star = \arg\min_{x \in \mathcal{X}} f(x), \quad \text{s.t.,} \quad g_i(x) \leq 0, \; h_j(x) = 0. \tag{2.11}$$

- **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

$$C = \{x \in \mathcal{X}, \quad \text{s.t.,} \quad g_i(x) \leq 0, \; h_j(x) = 0, \; i = 1, \ldots, I, \; j = 1, \ldots, J\}. \tag{2.12}$$

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

$$x_\star \text{ is a local optima} \iff \exists \lambda > 0 \text{ s.t. } x_\star = \underset{x\in\mathcal{X} \text{ s.t. } ||x-x_\star||\leq\lambda}{\arg\min} f(x). \tag{2.13}$$

$$x_\star \text{ is a global optima} \iff x_\star = \underset{x\in\mathcal{X}}{\arg\min} f(x). \tag{2.14}$$

Example: Unique and non-unique closed form minima

In unconstrained optimisation, we have functions that have a unique global minimum and others that have more than one. For instance, Figure 4 shows $(x-1)^2+(y+2)^2$ on the left, which has a unique minimiser on $(x,y) = (1,-2)$ and $(x^2-1)^2+(y^2-1)^2$ on the right, where every global minimiser belongs to the set $\{(x,y) : x = +-1 \wedge y = +-1\}$.

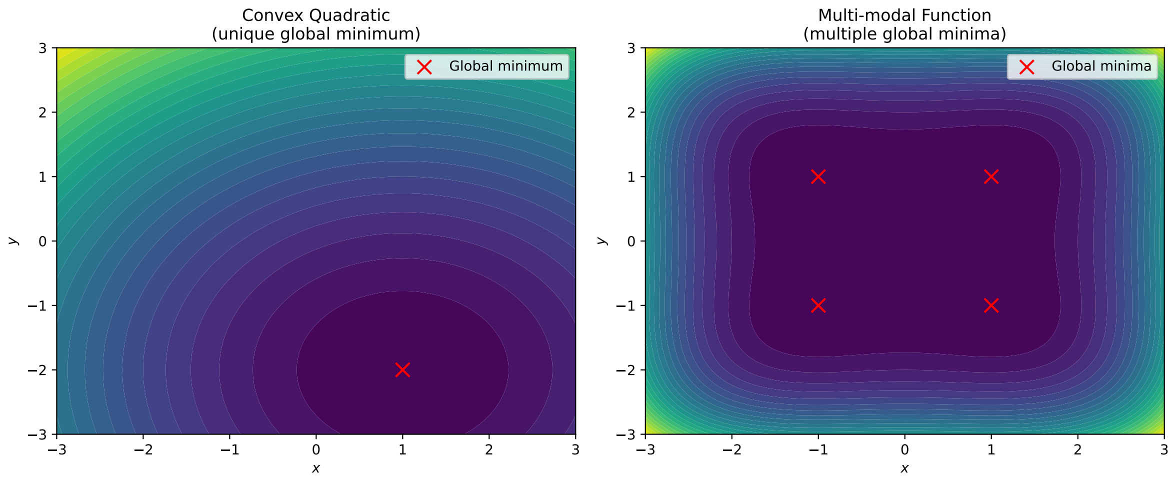

Fig. 4. Functions with a unique (left) and non-unique (right) global minima.

Interplay between constraints and local/global optima

On the other hand, we may be presented with functions such as $f(x,y) = \sin(x)*\sin(y)+0.1(x^2+y^2)$, which has a global minimum but also local minima. Figure 5 shows how different restrictions change the number and type of optima. In this case, when restricting the minimisation problem to the circle centred in $(1.5, 1)$ with radius 2, we observe a minimiser that is not the global minimum of the unconstrained problem (an unfeasible point) nor one of the local minima.

2.2 Continuous unconstrained optimisation

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

$$\theta \in \underset{\theta\in\Theta}{\arg\min} L(\theta). \tag{2.15}$$

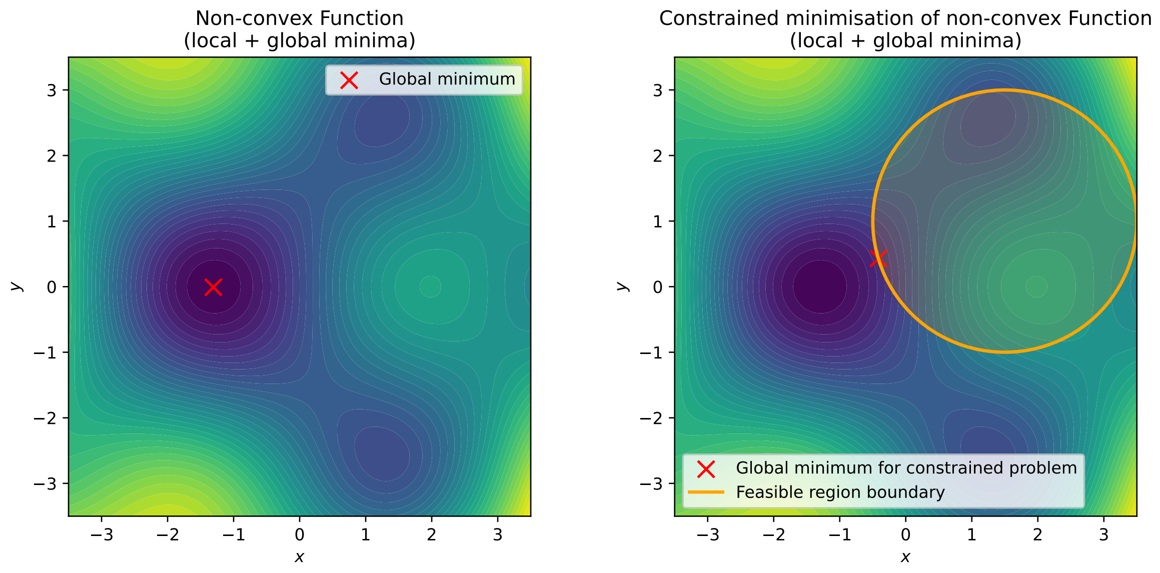

Fig. 5. Minima for constrained (left) and unconstrained (right) problems.

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

$$\forall \theta \in \Theta,\ L(\theta_\star) \leq L(\theta), \tag{2.16}$$

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

$$\exists\, \delta > 0 \text{ s.t. } \forall\, \theta \in \Theta\ \ \|\theta - \theta_\star\| < \delta \ \Rightarrow\ L(\theta_\star) \leq L(\theta). \tag{2.17}$$

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

$$\begin{aligned} f \colon \mathbb{R}^2 &\to \mathbb{R} \\ x &\mapsto f(x) = (p-1)x^2 + (p+1)y^2, \quad p \in \mathbb{R} \end{aligned} \tag{2.18}$$

Observe that

$$\nabla f = \begin{bmatrix} 2(p-1)x \\ 2(p+1)y \end{bmatrix}, \tag{2.19}$$

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

$$\nabla^2 f = \begin{bmatrix} 2(p-1) & 0 \\ 0 & 2(p+1) \end{bmatrix}, \tag{2.20}$$

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

Figure 6 shows the function behaviour for different p. What happens when $|p| = 1$?

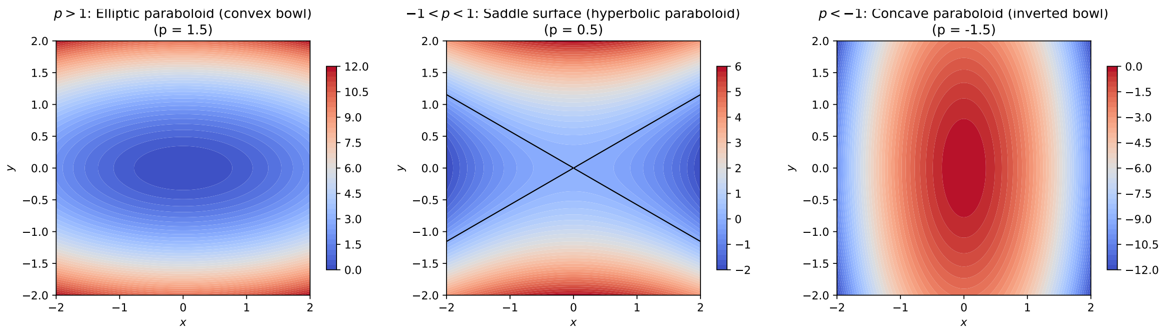

Fig. 6. Different types of critical points for $f(x, y) = (p-1)x^2 + (p+1)y^2$.

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:

$$\lambda x + (1-\lambda)x' \in \mathcal{S}, \quad \forall \lambda \in [0, 1]. \tag{2.21}$$

For illustration, Figure 7 shows two convex and two 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,

$$\text{epi}(f) = \{ (x, t) \in \mathcal{X} \times \mathbb{R} \mid f(x) \le t \}. \tag{2.22}$$

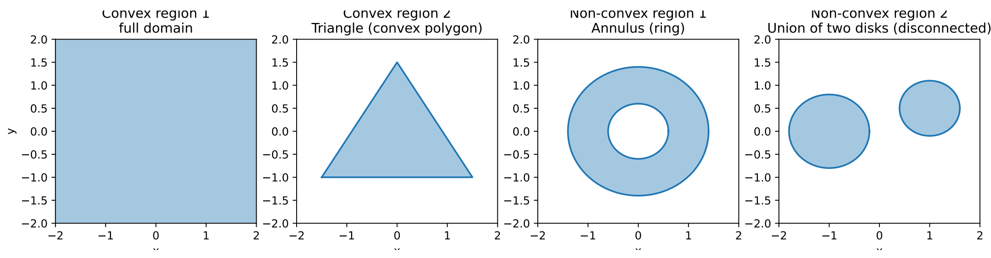

Fig. 7. Examples of convex and non-convex sets, highlighted in blue.

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 $\forall x, x' \in \mathcal{X}$

$$f\big(\lambda x + (1-\lambda)x'\big) \;\leq\; \lambda f(x) + (1-\lambda)f(x'), \quad \forall\, \lambda \in [0,1]. \tag{2.23}$$

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 $\mathbb{R}$:

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

Figure 8 shows the epigraphs for these 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 $a, b \in \mathbb{R}, a < b$,

$$f(b) - f(a) = \int_a^b f'(x)dx, \tag{2.24}$$

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

$$f(x') = f(x_0) + \nabla f(x_0)\rho u + \mathcal{O}(\rho^2), \tag{2.25}$$

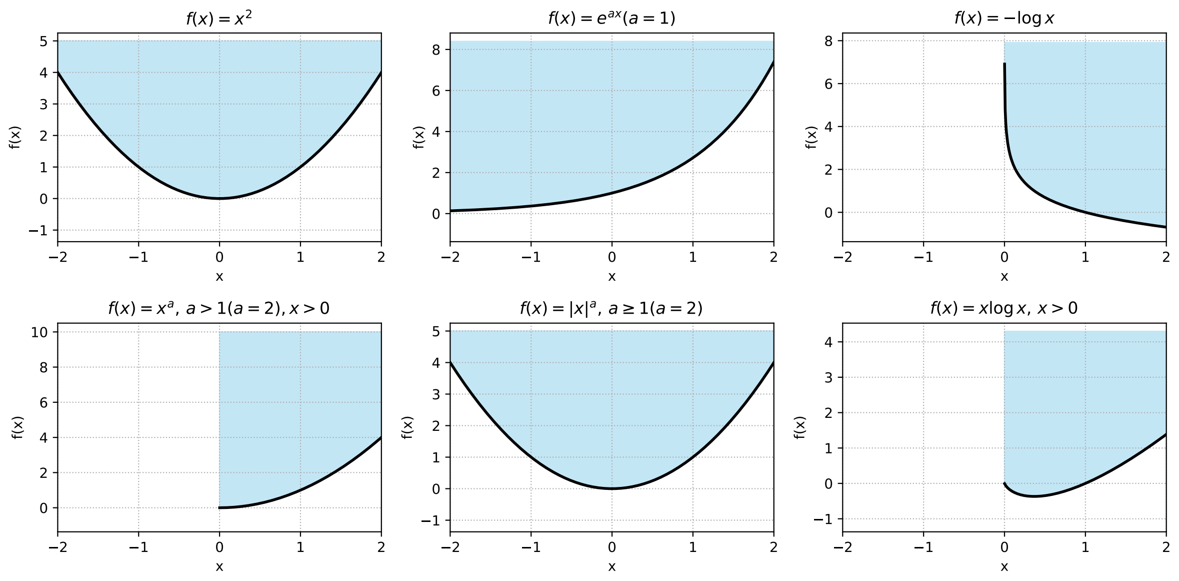

Fig. 8. Convex 1D functions with their epigraph in light blue.

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,

$$f'(b) - f'(a) = \int_a^b f''(x)dx \geq 0, \tag{2.26}$$

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

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

equivalently,

$$f(b) \geq f(a) + (b - a)f'(a), \tag{2.28}$$

meaning that the function f *is always above its tangent.* Evaluating (2.28) for (a, z) and (b, z), where $z = (1 - t)a + tb$, we have

$$f(z) \geq f(a) + (z - a)f'(a) \tag{2.29}$$

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

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

$$f(z) \geq (1 - t)f(a) + tf(b) + (1 - t)(tb - ta)f'(a) + t[(1 - t)a - (1 - t)b]f'(b) \tag{2.31}$$

$$= (1 - t)f(a) + tf(b) + (1 - t)t(b - a)[f'(a) - f'(b)] \tag{2.32}$$

$$\geq (1 - t)f(a) + tf(b). \tag{2.33}$$

This concludes the proof. Discuss in class how to extend this to arbitrary dimension $d > 1$. ■

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

$$x_{t+1} = x_t + \eta_t d_t, \tag{2.34}$$

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

$$L(x_t + \eta_t d_t) \leq L(x_t). \tag{2.35}$$

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 it: 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

Let us consider the function

$$J = (\theta - 3)^2. \tag{2.36}$$

In Figure 9 we show how the steepest descent converges/diverges for different learning rates.

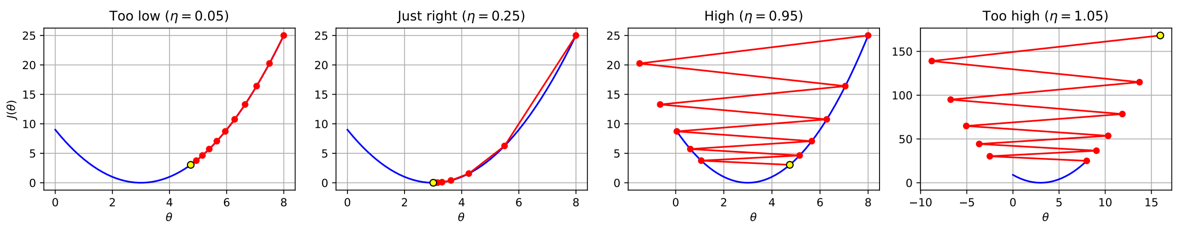

Fig. 9. Gradient based optimisation with different learning rates.

The learning rate is usually tuned based on heuristics. When implementing the rule

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

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

2.4.2 Momentum

In higher dimensions, we want to move faster in some directions and slower in others, 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 receives updates of a higher magnitude. This is particularly useful when the evaluation of the gradient is noisy. Figure 10 shows how learning changes using momentum, with the same learning rates as the previous example.

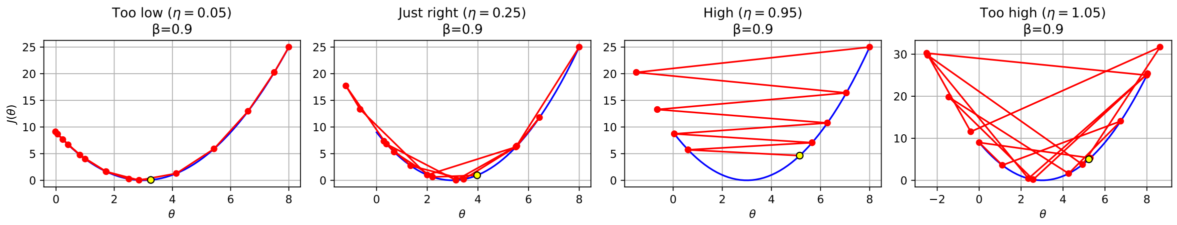

Fig. 10. Gradient based optimisation with momentum.

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 around 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 to accelerate (or de-accelerate) convergence.

Example: convergence for a parabola (2)

Let us consider the same parabolic function as before. This time we will use the Newton method presented above.
As shown in Figure 11, this method converges in one step (for a quadratic function). That is, the update corresponds to the closed form solution of the minimisation problem.

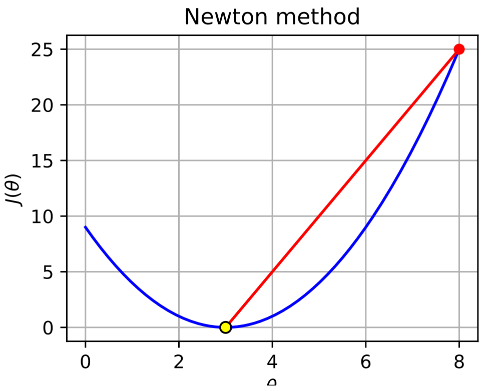

Fig. 11. One-step parabola minimisation with the Newton method.

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 the law $q(z)$ is unknown. However, since we usually have samples of q, we can do a sample approximation of 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

Consider a toy example of linear regression where $q(y|x) = \mathcal{N}(y; x, \sigma^2)$ for a given $\sigma > 0$. Consequently, we can compare the true expectation (equal to the variance in this case) with the empirical approximation for increasing values of N.

Figure 12 shows how the empirical loss changes as we consider more and more samples.

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 magnitudes of the coordinates of the gradient g_t, as this ensures convergence (or prevents divergence). We next explore some different choices of this estimate.

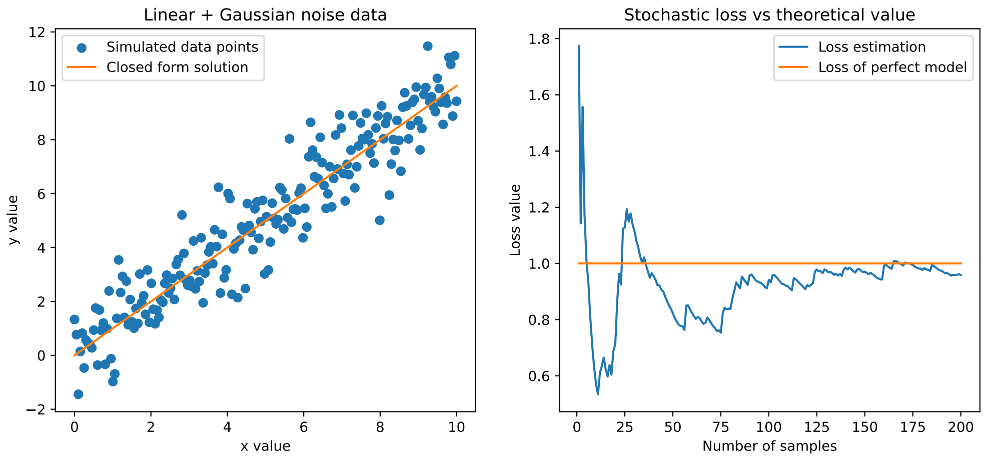

Fig. 12. Estimation of the true loss via the empirical value from data.

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 Adagrad is to 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 close 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) proposed by (?, ?), 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

We will consider the function

$$f(x, y) = 0.1x^2 + y^2 . \tag{2.54}$$

Notice that this function is convex, hence we would expect a reasonable gradient based method to converge. However, the speed and way in which they converge might be different in every case. This text function will illustrate this, since its curvature is much more pronounced along the y axis.

We visualise the effect of the different optimisers presented above in Figure 13. Notice that SGD converges more slowly than RMSprop and Adam. Adagrad, on the other hand, slows down too early.

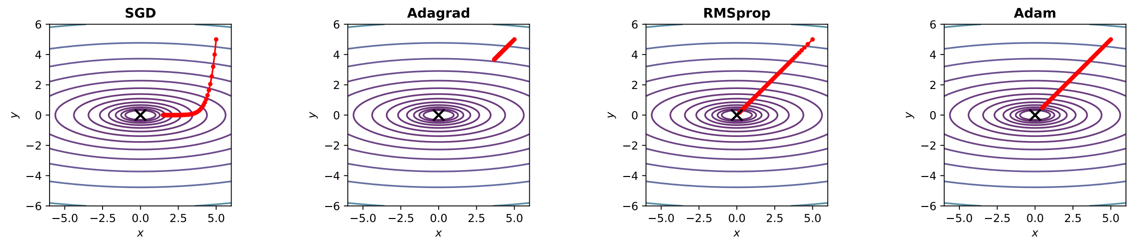

Fig. 13. Convergence comparison for different optimisers.

2.6 Training and Regularisation of Neural Networks

2.6.1 Network Architecture

As discussed previously, achieving good generalisation does not simply require arbitrarily wide networks. The **architecture** of a neural network refers to its overall structure: number of layers, neurons per layer, connectivity, and activation functions.

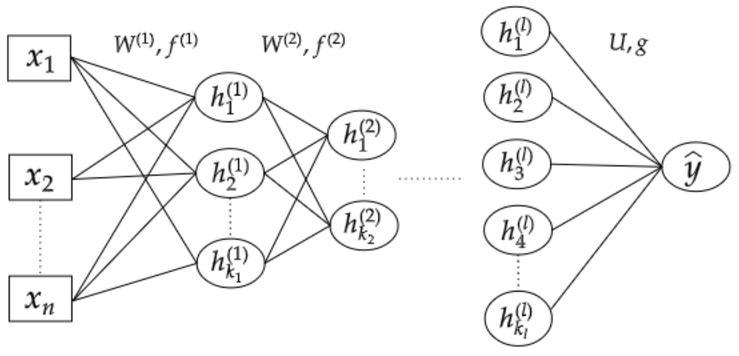

Fig. 14. A neural network of depth l.

We adopt the following notation:

$$h^{(k)} = f^{(k)}(h^{(k-1)}W^{(k)} + b^{(k)}), \quad k \in \{1, \ldots, l\}, \quad h^{(0)} = x, \tag{2.55}$$

$$\hat{y} = g(h^{(l)}U + c), \tag{2.56}$$

where g denotes the output activation defining the **output unit**.

2.6.2 Cost Function and Output Units

Unlike linear models, the use of nonlinear activations renders the cost function generally non-convex, so gradient descent does not guarantee convergence to a global optimum. Although least squares may be used, in practice one minimises the negative log-likelihood of a probabilistic model $p(y|\boldsymbol{x};\boldsymbol{\theta})$:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(\boldsymbol{y}|\boldsymbol{x})]. \tag{2.57}$$

The choice of output unit determines the appropriate likelihood and loss:

1. **Linear:** $\hat{\boldsymbol{y}} = h^{(l)}U + c$

   Models $\mathcal{N}(\boldsymbol{y}|\hat{\boldsymbol{y}}, \boldsymbol{I})$ and is equivalent to minimising MSE; suited for regression.

2. **Sigmoid:** $\hat{y} = \text{sig}(h^{(l)}U + c)$

   Used for binary classification with $p(y|\boldsymbol{x}) = \text{Bernoulli}(y|p)$ and $\hat{y} = p(y = 1|\boldsymbol{x})$.

3. **Softmax:** $\hat{\boldsymbol{y}} = \text{softmax}(h^{(l)}U + c)$

   For multiclass problems, with
   $$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

2.6.3 Forward Propagation

In a feedforward network, information flows from input $\boldsymbol{x}$ to output $\hat{\boldsymbol{y}}$. During training, this process computes the cost $J(\boldsymbol{X}, \boldsymbol{\theta})$.

Algoritmo 1 Forward Propagation

Require: Depth l, weights $\boldsymbol{W}^{(k)}$, biases $\boldsymbol{b}^{(k)}$, input $\boldsymbol{x}$, output $\boldsymbol{y}$, and output parameters U, c, g.

```
1: $\boldsymbol{h}^{(0)} \leftarrow \boldsymbol{x}$
2: for $k = 1, \dots, l$ do
3:     $\boldsymbol{u}^{(k)} \leftarrow \boldsymbol{h}^{(k-1)}\boldsymbol{W}^{(k)} + \boldsymbol{b}^{(k)}$
4:     $\boldsymbol{h}^{(k)} \leftarrow f^{(k)}(\boldsymbol{u}^{(k)})$
5: $\hat{\boldsymbol{y}} \leftarrow g(\boldsymbol{h}^{(l)}U + c)$
6: $J \leftarrow L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
```

2.6.4 Backward Propagation – Preliminaries

The **backpropagation** algorithm propagates cost information backwards to compute gradients efficiently. While analytic gradients can be derived, automatic differentiation provides faster and more scalable computation, enabling modern deep learning.

Assume training is performed in **mini-batches** $(x^d)_{d=1}^N$, enabling efficient matrix operations on GPUs. For an MSE loss in regression:

$$J(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2N}\sum_{d=1}^{N}(\hat{y}_d - y_d)^2.$$

Our goal is to update all weights $w_{ij}^{(k)}$. Using the chain rule:

$$\frac{\partial J_d}{\partial w_{ij}^{(k)}} = \frac{\partial J_d}{\partial u_{dj}^{(k)}} \frac{\partial u_{dj}^{(k)}}{\partial w_{ij}^{(k)}},$$

where we define the **error term**

$$\delta_{dj}^{(k)} \equiv \frac{\partial J_d}{\partial u_{dj}^{(k)}},$$

and note that

$$\frac{\partial u_{dj}^{(k)}}{\partial w_{ij}^{(k)}} = h_{di}^{(k-1)}.$$

Hence,

$$\frac{\partial J_d}{\partial w_{ij}^{(k)}} = \delta_{dj}^{(k)} h_{di}^{(k-1)},$$

and the total gradient (ignoring the constant $1/N$ for now) is

$$\frac{\partial J}{\partial W^{(k)}} = (h^{(k-1)})^T @ \delta^{(k)}. \tag{2.58}$$

2.6.5 Backward Propagation – Hidden Layers

From the chain rule:

$$\delta_{dj}^{(k)} = \frac{\partial J_d}{\partial u_{dj}^{(k)}} = \sum_{a=1}^{k_{k+1}} \frac{\partial J_d}{\partial u_{da}^{(k+1)}} \frac{\partial u_{da}^{(k+1)}}{\partial u_{dj}^{(k)}} = \sum_{a=1}^{k_{k+1}} \delta_{da}^{(k+1)} \frac{\partial u_{da}^{(k+1)}}{\partial u_{dj}^{(k)}},$$

where, since $\frac{\partial u_{da}^{(k+1)}}{\partial u_{dj}^{(k)}} = w_{ja}^{(k+1)} f'(u_{dj}^{(k)})$, we have

$$\delta_{dj}^{(k)} = f'(u_{dj}^{(k)}) \sum_{a=1}^{k_{k+1}} w_{ja}^{(k+1)} \delta_{da}^{(k+1)}.$$

In matrix form:

$$\delta^{(k)} = f'(u^{(k)}) * (\delta^{(k+1)} @ (W^{(k+1)})^T). \tag{2.59}$$

2.6.6 Backward Propagation – Output Layer

For a regression problem with linear output and MSE loss:

$$\delta_{d1}^{(l)} = (\hat{y}_d - y_d)(\hat{y}_d)' = (\hat{y}_d - y_d),$$

and with the normalisation term:

$$\delta^{(l)} = \frac{1}{N}(\hat{y} - y). \tag{2.60}$$

Furthermore,

$$\frac{\partial J}{\partial b^{(k)}} = \mathrm{Sum}_1(\delta^{(k)}), \quad \frac{\partial J}{\partial U} = (h^{(l)})^T @ \delta^{(l)}, \quad \frac{\partial J}{\partial c} = \mathrm{Sum}_1(\delta^{(l)}), \tag{2.61}$$

where $\mathrm{Sum}_1(A)$ denotes summation over the first axis.

Algoritmo 2 Backward Propagation

```
Require: Learning rate $\lambda$.

1: Perform forward propagation, store $(\hat{y}), (u^{(k)}), (h^{(k)})$.
2: Compute output error $\delta^{(l)}$ according to the output unit.
3: Update $U \leftarrow U - \lambda\frac{\partial J}{\partial U}, \quad c \leftarrow c - \lambda\frac{\partial J}{\partial c}$.
4: for $k = l, \dots, 1$ do
5:     Compute $\delta^{(k)}$ using the hidden-layer recursion.
6:     Evaluate $\frac{\partial J}{\partial W^{(k)}}, \frac{\partial J}{\partial b^{(k)}}$.
7:     Update $W^{(k)} \leftarrow W^{(k)} - \lambda\frac{\partial J}{\partial W^{(k)}}$.
8:     Update $b^{(k)} \leftarrow b^{(k)} - \lambda\frac{\partial J}{\partial b^{(k)}}$.
```

Observación 2.1. The algorithm is applied to each mini-batch for several **epochs**. Increasing epochs generally decreases training error, but excessive training may degrade generalisation due to **overfitting**.

2.6.7 Regularisation

Neural networks are often large models; controlling their complexity is crucial. **Regularisation** techniques aim to reduce the generalisation error while retaining sufficient model capacity.

L2 Regularisation. Adds a quadratic penalty on weights:

$$\tilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\alpha}{2}\|\boldsymbol{\theta}\|_2^2.$$

The hyperparameter α controls the regularisation strength. Usually, only weights are penalised, not biases. In practice this corresponds to *weight decay*:

$$W^{(k)} \leftarrow (1-\beta)W^{(k)} - \lambda\frac{\partial J}{\partial W^{(k)}}, \quad \beta = \lambda\alpha.$$

3 Probability

3.1 Introduction

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

The field of probability studies, from a quantitative perspective, how *likely* an event is. Conceptually, and perhaps historically, there are two main interpretations of probability. The first one is **frequentist probability**, which relates to frequency of occurrence, and then applies only to event that can be repeated an infinite number of times, such as throwing a dice or flopping a coin. As a consequence, this standpoint fails to assign a probability to events that are impossible to repeat, such as the average temperature of the Earth's surface reaching an all-time maximum in the year 2025. A second interpretation is that of **Bayesian probability**, which represents uncertainty about the occurrence of an event. This uncertainty might come from different sources, such unknown features in the experiments (epistemological uncertainty) or random components (aleatoric uncertainty). In this case, events need not be repeatable be be assigned with a probability.

A basic knowledge of probability theory, definitions and results in fundamental in ML. This is because in ML we design, train and deploy mathematical models that i) aim to capture/quantify uncertainty, and ii) deal with noise-corrupted training data. Therefore, a rigorous account of uncertainty is central to real-world ML applications.

3.1.1 Definitions

To start studying probability, we will focus on the outcome ω of a hypothetical experiment, e.g., throwing a dice, where ω can take values $\{1, 2, 3, 4, 5, 6\}$. In this context, we can define:

Definition 3.1 (Sample space)**.** The set containing all the possible outcomes ω of an experiment is called sample space and is denoted by Ω.

Definition 3.2 (Event space)**.** The set $\mathcal{A}$, referred to as event space, contains all possible subset of the sample space Ω. Therefore, each element $A \in \mathcal{A}$ represents a possible results of the experiment.

Definition 3.3 (Probability)**.** The function

$$\mathbb{P} : \mathcal{A} \to [0, 1] \tag{3.1}$$
$$x \mapsto \mathbb{P}(x) \tag{3.2}$$

denotes the probability of the result of the experiment falling inside the elements of $\mathcal{A}$, that is, $\mathbb{P}(A) = \mathbb{P}(\omega \in A)$. The function $\mathbb{P}$ needs to fulfil some standard properties such as $\mathbb{P}(\Omega) = 1, \mathbb{P}(\emptyset) = 0$, and $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$.

$\mathcal{A}$ vs Ω

Why is the probability defined over $\mathcal{A}$ and not over Ω? Discuss via some examples

Examples

[TODO: Consider basic examples (dice, coin, uniform, rain), and present the sample space, event space, and probability]

We refer to the triplet $(\Omega, \mathcal{A}, \mathbb{P})$ as **probability space**.

3.1.2 Basic properties

The joint probability of events A and B is denoted by

$$\mathbb{P}(A \wedge B) = \mathbb{P}(A \cap B) = \mathbb{P}(A, B), \tag{3.3}$$

note that if $\omega \in A$ and $\omega \in B$, then, $\omega \in A \cap B$.

The conditional probability of the event A occurring, given that the event B occurred is denoted by

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A, B)}{\mathbb{P}(B)}, \tag{3.4}$$

which is only valid when $\mathbb{P}(B) > 0$.

Additionally, we say that events A and B are **independent** iff $\mathbb{P}(A, B) = \mathbb{P}(A)\mathbb{P}(B)$. Observe that this implies that

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A, B)}{\mathbb{P}(B)} = \frac{\mathbb{P}(A)\mathbb{P}(B)}{\mathbb{P}(B)} = \mathbb{P}(A), \tag{3.5}$$

meaning that when A and B are independent, the latter provides no **information** for A.

Lastly, the probability of intersection, i.e., the probability of the events $\omega \in A$ or $\omega \in B$ is given by

$$\mathbb{P}(A \vee B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \wedge B). \tag{3.6}$$

3.2 Random variables

In general, other than basic toy examples, we do not refer to the underlying experiment and its sample/event spaces. We instead consider *quantities of interest* that result from the outcome of the experiment. Therefore, let us consider a map from the sample space to a target space $\mathcal{T}$, e.g., $\mathcal{T} = \mathbb{R}$, $\mathcal{T} = \mathbb{N}$ or $\mathcal{T} = \{1, 2, 3, \ldots, N\}$.

Definition 3.4 (Random variable)**.** A function

$$X : \Omega \to \mathcal{T} \tag{3.7}$$
$$\omega \mapsto X(\omega) \tag{3.8}$$

is referred to as random variable. In general, we will denote the function as X and its value as x, ignoring the explicit dependence on ω.

Observación 3.1. From now on, we will consider the outcome of the experiment (living in the sample space) as the value of the RV, this aims to have a more streamlined setup avoiding the need of a map from Ω to $\mathcal{T}$. Accordingly, the event space $\mathcal{A}$ and the probability $\mathbb{P}$ correspond to the outcomes (values) of X, we will usually denote the sample space by $\mathcal{X}$.

3.2.1 Discrete RVs

If the sample space is finite or countably infinite, we will say that the RVis discrete. In this case, we denote the probability of the event $X = x$ by $\mathbb{P}(X = x)$. We define de following.

Definition 3.5 (Probability mass function (pmf))**.** For a discrete RV $X \in \mathcal{X}$ the function

$$p_X : \mathcal{X} \to [0, 1] \tag{3.9}$$
$$x \mapsto p_X(x) = \mathbb{P}(X = x) \tag{3.10}$$

denotes the probability of the event in which the RV X takes the value $x \in \mathcal{X}$. Evidently,

$$\sum_{x \in \mathcal{X}} p_X(x) = 1. \tag{3.11}$$

Discrete RV

[TODO: Give an example, e.g., a dice, picking items and/or a infinite state space one]

3.2.2 Continuous RVs

If $\mathcal{X} = \mathbb{R}^d$, with $d > 1$, or any other infinitely uncountable space, we say that the RV is continuous. Observe that in this case we cannot defined a pmf as in Def. 3.5. This is because since there is an uncountable number of symbols in the sample space $\mathcal{X}$, we cannot assign each of them with a probability strictly greater than zero and still a total probability mass equal to one. Therefore, we make use of the event space and assign probabilities to intervals rather tan particular values.

Definition 3.6 (Cumulative distribution function (cdf))**.** For a continuous RV $X \in \mathcal{X}$, we can define the probability of X to be less or equal that a given value $x \in \mathcal{X}$ by

$$P_X : \mathcal{X} \to [0, 1] \tag{3.12}$$
$$x \mapsto P_X(x) = \mathbb{P}(X \leq x). \tag{3.13}$$

Observe that this also allows to denote the probability of X lying in a bounded interval, that is,

$$\mathbb{P}(a \leq X \leq b) = P_X(b) - P_X(a). \tag{3.14}$$

The cdf is monotonically non-decreasing by construction, and, when $\mathcal{X} = \mathbb{R}$, we have

$$\lim_{x \to -\inf} P_X(x) = 0 \tag{3.15}$$
$$\lim_{x \to \inf} P_X(x) = 1. \tag{3.16}$$

The idea to assign probabilities to intervals, suggest that there exists a density of probability, that is, an amount of probability mass per unit of length. This is formalised via the following definition.

Definition 3.7 (Probability density function (pdf))**.** For a continuous RV $X \in \mathcal{X}$, the probability density function of X is given by the function

$$p_X : \mathcal{X} \to \mathbb{R} \tag{3.17}$$
$$x \mapsto p_X(x) = \frac{d}{dx} P_X(x). \tag{3.18}$$

This is possible when P_X is differentiable.

Knowing the pdf, we can express:

$$\mathbb{P}(a \leq X \leq b) = \int_a^b p_X(x) dx = P_X(b) - P_X(a), \tag{3.19}$$

which is a direct particular case of the fundamental theorem of Calculus.

Also, for Δx small, we have

$$\mathbb{P}(x \leq X \leq x + \Delta x) = \int_x^{x+\Delta x} p_X(x) dx \simeq p(x) \Delta x. \tag{3.20}$$

Lastly, if the cdf is **strictly monotonically increasing**, its inverse exists and it is known as the **quantile function**. The use quantiles is useful when assessing the value of the RV wrt the range of possible values.

Examples

[TODO: Show the Gaussian distribution and present cdf, pdf and quantile]

3.2.3 Properties and identities

Probability can be thought of as an extension of logical reasoning. The following properties are consistent with such extension. Let us first consider, in the same was as we did when we defined event probabilities, the following.

Both for discrete (pmf) and discrete (pdf) RVs, we can denote $p(x, y)$ the joint pdf/pmf for RVs $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. This can be seen as the probability of the *intersection* event, where $X = x$ and $Y = y$.

Furthermore, we can write

$$p_X(x) = \sum_{y \in \mathcal{Y}} p(x, y) \quad \text{(discrete)} \tag{3.21}$$

$$p_X(x) = \int_{\mathcal{Y}} p(x, y) dy \quad \text{(continuous)}. \tag{3.22}$$

In the ML community, this is known as **sum rule**, since it allows us to express the density $p(y)$ from a joint density by summing over all possible values.

A related expression is the so called **law of total probability**, which reads

$$p_X(x) = \sum_{y \in \mathcal{Y}} p(x|y)p(y) \quad \text{(discrete)} \tag{3.23}$$

$$p_X(x) = \int_{\mathcal{Y}} p(x|y)p(y) dy \quad \text{(continuous)}. \tag{3.24}$$

This operation is usually called **marginalisation**, or **disintegration** (of y).

The so called **product rule** is the decomposition of the joint law as follows

$$p(x, y) = p(y|x)p(x), \tag{3.25}$$

which allows to decompose the joint pmf/pdf using the law of the conditional event $X = x$ given that $Y = y$, and the marginal law of Y. A key consequence of the above expression is that, since the factorisation works in both ways, we have

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \tag{3.26}$$

which is known as the **Bayes theorem**. This is a fundamental result in probabilistic ML, since we usually observe measurements y and want to infer parameters or latent variables x.

Examples

[TODO: Let's go over the Month Python example or something about drawing cards]

3.2.4 Moments

Definition 3.8 (Mean)**.** The mean, or expected value, of an RV $X \in \mathcal{X}$, often denoted by $\mu = \mu_X$, is defined as

$$\mathbb{E}[X] = \int_{\mathcal{X}} x p_X(x) dx \quad \text{(continuous)} \tag{3.27}$$

$$\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x p_X(x) \quad \text{(discrete)} \tag{3.28}$$

$$\tag{3.29}$$

where in the continuous case we have assumed the existence of a pdf.

Observe that $\mathbb{E}[\cdot]$ is a linear operator, meaning that

$$\mathbb{E}[aX + b] = a\,\mathbb{E}[X] + b \quad a, b \in \mathbb{R} \tag{3.30}$$

$$\mathbb{E}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathbb{E}[X_i], \tag{3.31}$$

$$\tag{3.32}$$

where $X_i, X_2, \ldots, X_n$ is a collection of arbitrary RVs with finite mean.

Definition 3.9 (Variance)**.** The variance measures the **spread** of a distribution (with respect to its mean) and it given by

$$\mathbb{V}[X] = \mathbb{E}[(X - \mu_X)^2] \tag{3.33}$$

$$= \int_{\mathcal{X}} (x - \mu_X)^2 p_X(x) dx \tag{3.34}$$

$$= \int_{\mathcal{X}} (x^2 - 2x\mu_X + \mu_X^2)^2 p_X(x) dx \tag{3.35}$$

$$= \mathbb{E}[X^2] - \mu_X^2, \tag{3.36}$$

where in the discrete case is obtained similarly.

A related quantity is the **standard deviation**, given by

$$\text{std}[X] = \sqrt{\mathbb{V}[X]}. \tag{3.37}$$

Let us consider an arbitrary sequence of **independent** RVs $X_i, X_2, \ldots, X_n$ and $a, b \in \mathbb{R}$. Two key properties of the variance are

$$\mathbb{V}[aX_1 + b] = a^2\,\mathbb{V}[X_1] \tag{3.38}$$

$$\mathbb{V}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathbb{V}[X_i] \tag{3.39}$$

MISSING: SOME INEQUALITIES AND CORRELATION

3.3 Some particular RVs

3.3.1 Bernoulli and binomial

Consider tossing a coin, where $\mathbb{P}(\text{heads}) = \theta$, where $0 \leq \theta \leq 1$. Note that, as a consequence, $\mathbb{P}(\text{tails}) = 1 - \theta$. This is called the Bernoulli distribution, and it is written as

$$Y \sim \text{Ber}(\theta). \tag{3.40}$$

Bernoulli's pmf is given by

$$p_{\text{Ber}}(y) = \begin{cases} \theta, & y = 1, \\ 1-\theta, & y = 0, \\ 0, & \text{otherwise.} \end{cases}$$

More concisely, we can write the pmf as

$$p_{\text{Ber}}(y) = \theta^y (1-\theta)^{1-y}. \tag{3.41}$$

Let us now consider N Bernoulli trials, denoted $Y_i \sim \text{Ber}(\cdot|\theta), i = 1, \ldots, N$ – this can be thought of as tossing a coin N times. Define the RV S as the total number of heads, that is

$$S = \sum_{i=1}^{N} I(y_i = 1). \tag{3.42}$$

The distribution of S is

$$\text{Bin}(s|N, \theta) = \binom{N}{s} \theta^s (1-\theta)^{N-s}, \tag{3.43}$$

which is known as the Binomial distribution, where $\binom{N}{s} = \frac{N!}{(N-s)!s!}$.

Logistic regression

Within ML, the Bernoulli distribution is largely used for (probabilistic) classification. In such case we want to predict a binary variable $y \in \{0, 1\}$, which might represent the fact that a sample is of class 1 (and not class 0), conditional to some observed features $x \in \mathbb{R}^d$. We can then model $p(y = 1)$ as

$$p(y|x, \theta) = \text{Ber}(y|f(x, \theta)). \tag{3.44}$$

Since $f(x, \theta)$ represents the parameter of $\text{Ber}(y)$, we need $0 \leq f(x, \theta) \leq 1$. However, to avoid that requirement, we can leave $f(x, \theta)$ unconstrained and do

$$p(y|\theta, x) = \text{Ber}(y|\sigma(f(x, \theta))), \tag{3.45}$$

where

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{3.46}$$

is the logistic function. When we choose $f(x, \theta) = \theta_1^\top x + \theta_2$, we have the logistic regression model.

[TODO: Show plots of Log Reg and sigmoid function.]

The extension of the Bernoulli (resp. Binomial) distribution to the multivariate case, i.e., when the output of the experiments take values on categories $\{1, 2, \ldots, K\}$ with $K > 2$ is known as the categorical (resp. Multinomial). These distributions will be left for personal study.

3.3.2 Uniform distribution

MISSING

3.3.3 Gaussian distribution

The pdf the (scalar) Gaussian RV X is given by

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}, \tag{3.47}$$

when $\mu = 0$ and $\sigma = 1$, we say that X is a standard normal.

Why Gaussian

The Gaussian distribution is widely used for modelling continuous RVs, this is, in part, because:

- Central limit theorem: The scaled sum of multiple independent RVs, with different distributions, converges to a Gaussian RVs
- The Gaussian distribution is the distribution with the maximum entropy (fewer structural assumptions) for a fixed variance
- Its parameters are easy to interpret
- The maths are simple when using it in applications

The Gaussian distribution is central to model **observation noise**, and therefore is key in regression models. That is, one can consider a linear regression models where

$$Y|x \sim \mathcal{N}(y|a^\top x + b, \sigma^2), \tag{3.48}$$

which is equivalent to

$$y = a^\top x + b + \epsilon, \tag{3.49}$$

where $\epsilon \sim \mathcal{N}(0,\sigma^2)$.

MISSING: MVN

3.3.4 Other distributions

There is a number of relevant distributions that we will not cover due to time constraints. These are, among others,

- Student's t
- Cauchy
- Chi-square
- Laplace
- Beta
- inv-Gamma
- Gamma
- Exponential
- half Gaussian

3.4 Transformations of RVs distributions

In ML, we are interested in the construction of expressive **generative models**, i.e., probability distributions. Despite having a large collection of known distributions, sometimes we need to design purpose-specific models; this is achieved by transforming a given RV.

Consider $X \sim p_X(\cdot)$ and $Y = f(X)$ a deterministic transformation. We are interested in computing $p_Y(y)$.

The discrete case is straightforward, as we just need to sum over the possible (discrete) values. That is,

$$p_Y(y) = \sum_{x:y=f(x)} p_X(x). \tag{3.50}$$

Uniform into binary

Consider

$$p_X = Uniform(0, 1, \dots, 9), \tag{3.51}$$

and

$$y = f(x) = \begin{cases} 1, & \text{if } x \text{ is odd,} \\ 0, & \text{if } x \text{ is even.} \end{cases}$$

[TODO: Plot the pmfs]

The continuous case is a bit more involved, since we cannot directly sum over all possible values of the RV. However, we can work with the cdf to show that

$$P_Y(y) = \mathbb{P}(Y \leq y) = \mathbb{P}(f(x) \leq y) = \mathbb{P}(x \in \{x | f(x) \leq y\}). \tag{3.52}$$

Let us notice that if f is invertible, we can rely on the **change of variable theorem**, which states that

$$p_Y(y) = p_X(x) \left| \frac{dx}{dy} \right|. \tag{3.53}$$

This identity can be proven using the cdf. When f is invertible, we can denote $g = f^{-1}$ and note that (assuming that f is non-decreasing)

$$P_Y(y) = \mathbb{P}(Y \leq y) = \mathbb{P}(f(x) \leq y) = \mathbb{P}(x \leq g(y)) = P_X(g(y)). \tag{3.54}$$

We can now take the derivative, to give:

$$p_Y(y) = \frac{d}{dy} P_X(g(y)) = \frac{dP_X(g(y))}{dx} \frac{dx}{dy} = p_X(g(y)) \frac{dx}{dy}. \tag{3.55}$$

If f had been non-increasing, we would have obtained the same expression with the reversed sign. Therefore, we conclude eq. (3.53).

In the multivariate case, the CVT reads

$$p_Y(y) = p_X(x) \left| \det J_q(y) \right|, \tag{3.56}$$

where $J_q(y) = \frac{dg(y)}{dy}$ is the Jacobian of $g = f^{-1}$.

Examples

[TODO: Give 1 or two examples with plots. They could be χ^2, lognormal, uniform, half normal, etc.]

3.5 Monte Carlo sampling

Consider a set $\{x_1, x_2, \dots, x_N\}$ of i.i.d. samples of an RV $X \sim p_X$. We are interested in approximating the law p_X, and also computing expectation wrt it. An empirical estimate of the law of X is the so called **histogram**

$$P_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}(x), \tag{3.57}$$

where $\delta_{x_i}(\cdot)$ denotes de delta-Dirac mass in x_i.

Real approximation?

Does $P_N(x)$ becomes more and more similar to the true law of X as N grows?

Notice that P_N allows to compute (approximate) expectations. Let us consider a function $f : x \mapsto f(x)$, and

$$I_f \overset{\text{def}}{=} \int f(x)p(x)dx \simeq f(x)P_N(x) = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \overset{\text{def}}{=} I_N. \tag{3.58}$$

Due to the Strong Law of Large Numbers (SLLN), we know that I_N get arbitrarily close to I_f as $N \to \infty$.

A relevant question is how to generate the samples needed to approximate P_N, this is known as **sampling**. There are three main approaches to sample from distributions, namely:

- Rejection sampling
- Importance sampling
- Markov chain Monte Carlo.

We will briefly revise them in class.

References

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