Introduction to Machine Learning

by Khallil Ebrahim Benyattou

すこ
少しずつ

January 20, 2026

This page intentionally left almost blank.

Contents

CHAPTER 1

Introduction

[The] human perception system is rife with all ways of getting it wrong. [...] All it takes is a few sketches that are cleverly done and your brain can't figure it out! So, we are poor data-taking devices. That's why we have such a thing as science. Because we have machines that don't care what side of the bed they woke up (on) in the morning. [...] They'll get the data right.

Neil deGrasse Tyson, 2009
@ **Cosmic Quandaries**

An **algorithm** is an unambiguous sequence of instructions designed to complete a task. For simple tasks, it's relatively easy to manufacture an appropriate (and perhaps even optimal) algorithm. For more complicated tasks, it's often more practical to help the computer "learn" how to develop its own algorithm. I define **learning** as the process of turning information into knowledge and understanding.

Machine learning (ML) is the name given to the study of computer algorithms that improve *automatically* through experience (data).

An early definition of ML can be attributed to Arthur Samuel (1901–1990) — a pioneer of artificial intelligence research:

[Machine Learning is] a field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

One of the first examples of a computer learning a task without being explicitly programmed was Samuel's own work on simulating the game of checkers. The computer even ended up surpassing Samuel himself. Nowadays, it isn't so surprising that computers are able to surpass human-level performance (on narrow tasks) but it was a real doozie back in the day.

A formal definition of the types of algorithms studied in machine learning is attributed to computer scientist Tom Mitchell:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.

Tom M. Mitchell, 1997

e.g. For the task T of playing checkers, one could define the performance measure P as the percentage of games won against opponents, and the training experience E could be measured by the number of games simulated by the computer against itself. More concretely:

Experience E presents itself in the form of data. Data are observations. Thus, machine learning is a subfield of computer science that focuses on the study of algorithms that can learn from data and make predictions.

So far we've been very general and it helps to establish some form of taxonomy in the field of machine learning.

1.1 Paradigms

At a high level, a **paradigm** in machine learning is the broader philosophy or theoretical framework under which an algorithm is trained to solve a particular task. The paradigm one works in is informed by the type (structure and availability) of (**training**) **data** that has been collected and the nature of the feedback the algorithm receives i.e. how, if, or when the algorithm is informed about its performance. The broadest paradigms[1] are:

- **Supervised learning**
  - Type of data: A static (fixed) collection of labelled[2] data.
  - Feedback: Explicit and correct labels to compare our algorithm's predictions against.
- **Unsupervised learning**
  - Type of data: Static, unlabelled data.
  - Feedback: No explicit feedback and the algorithm must come to its own conclusions about the pattern or structure of the input data.
- Semi-supervised learning
  - Type of data: Static — a combination of labelled and unlabelled data.
  - Feedback: Explicit labels for a portion of the data (as in supervised learning) but the algorithm is designed to (often) infer patterns about the unlabelled data from the labelled data.
- **Reinforcement learning** is based on an agent-environment interaction framework driven by reward maximisation (or penalty minimisation).
  - Type of data: Dynamic — a set of observations (states, actions and transition probabilities) and rewards (and discounts for future reward value) that evolve over time.
  - Feedback: Rewards or penalties over time (immediate or delayed) based on the agent's sequential actions.

1.2 Tasks and Models

Suppose like before, that we are given a set of data and a task to accomplish within a particular paradigm of machine learning. The task is to investigate some real-life/natural phenomenon e.g. a *binary* classification task on e-mail data organises e-mails into *two* distinct groups — spam and not spam.

Oftentimes, such problems are particularly complex. To this end, it's beneficial to make some assumptions (influenced by either knowledge a priori or through exploratory data analysis) about which features/components are relevant, and how they relate to each other:

[1]There are more emerging paradigms but these are the broadest ones.
[2]The data presents itself as a set of (feature vector, label) pairs.

In general parlance, a **model** is a mathematical, logical or physical representation of a process, concept or item. We use models to idealise complex phenomena and extract important information. The value of a model is that it approximates reality in a way that's useful.

All models are wrong, but some are useful.

George Box (probably)

Simpler and less flexible models tend to be easier to interpret. The tradeoff of a simple model is that it's less likely to capture enough of a phenomenon's inherent complexity — the resulting errors in calculation are called **bias**. There's an important balance to strike between flexibility and interpretability.

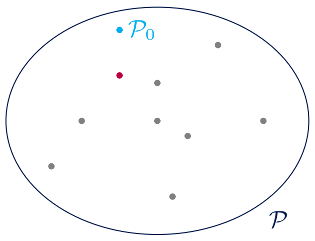
The term **machine learning model** is perhaps the most ambiguous term available to man. I'll try to be specific:

- A **probabilistic model** is the name given to the assumptions made about a process that generates training data.
- A **training/learning algorithm** is the procedure that uses the training data to learn a suitable function.
- I reserve the term **(machine learning) model** to refer to the **learned function** that has been **output** by a training algorithm.

Mathematics and statistics are natural languages to formalise a model.

A *mathematical model* is a description of a system using mathematical concepts and language. A *statistical model* is a type of mathematical model which embodies a set of statistical assumptions[3] concerning the generation of sample data. Statistical models differ from other mathematical models by being non-deterministic. Thus, in a statistical model, some of the variables don't have specific values and instead have probability distributions — these variables are *stochastic*.

Formally, a statistical model can be thought of as a pair $(\Omega, \mathcal{P})$ where Ω is a set of possible observations (the outcome/sample space) and $\mathcal{P}$ is a set of probability distributions on Ω. The thinking behind this definition is that there exists a true probability distribution $\mathcal{P}_0$ that dictates the generation of the sample data that's been collected. $\mathcal{P}$ is typically (always?) parameterised in some way so we can write $\mathcal{P} = \{\mathcal{P}_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$. This collection contains a distribution that adequately approximates $\mathcal{P}_0$.



A statistical model $(\Omega, \mathcal{P} = \{\mathcal{P}_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\})$ is called *parametric* if Θ has finite dimension i.e. $\Theta \subseteq \mathbb{R}^k$ where $k \in \mathbb{Z}^+$. k is called the dimension of the model. A model is called *non-parametric* if Θ is infinite-dimensional.

[3]Usually specified as a mathematical relationship between random variables and non-random variables.

Most of the recent economic value generated by machine learning is through supervised learning but there are important use-cases for unsupervised learning.

Andrew Ng (2018)

We begin with supervised learning $\downarrow$

CHAPTER 2

Supervised Learning

Supervised learning is the paradigm of machine learning under which we use labelled training data to employ an algorithm that outputs a machine learning model which can correctly classify an unseen input, or predict an outcome based on said input.

The labelled training data is a set of $n \in \mathbb{Z}$ (object, label) pairs

$$\mathcal{T} := \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$$

We call this set the **training set**. Each pair $(x^{(i)}, y^{(i)})$ is called a **training example**:

- Each $x^{(i)}$ is an **input vector** of dimension p which contains information about each object:

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix}.$$

  - In many applications, there is typically some form of feature-extraction that is performed on the original data variables. This results in a **feature vector**:

$$\boldsymbol{\phi}(x^{(i)}) = \begin{bmatrix} \phi_1(x^{(i)}) \\ \vdots \\ \phi_M(x^{(i)}) \end{bmatrix},$$

    where each feature $\phi_j(x^{(i)})$ is the result of evaluating the j^{th} basis function ϕ_j at $x^{(i)}$. Each ϕ_j can be non-linear so this allows for greater flexibility in modelling.

- $y^{(i)}$ is called the **label** or **response** corresponding to $x^{(i)}$.

Let's say the input vectors $x^{(i)}$ live in a space $\mathcal{X}$ and the labels $y^{(i)}$ live in $\mathcal{Y}$. Then, $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{Y}$.

The goal of a supervised learning algorithm is to use a set of training data $\mathcal{T}$ to learn a function $h\colon \mathcal{X} \to \mathcal{Y}$ that predicts the value of $y \in \mathcal{Y}$ corresponding to a given input feature vector $x \in \mathcal{X}$. Such a function h has historically been called a **hypothesis**.

A successful (and if we're lucky, optimal) h will exhibit the following two properties:

- For each example $(x^{(i)}, y^{(i)}) \in \mathcal{T}$, the prediction $h(x^{(i)})$ is a "good" estimate for the true label $y^{(i)}$.
- The estimate h also generalises well in the sense that its output $h(x)$ for some unseen vector $x \in \mathcal{X}$ is a "good" predictor for its corresponding expected value of $y \in \mathcal{Y}$ given x.

A supervised learning problem is called

- a **regression problem** if $\mathcal{Y} = \mathbb{R}$, or
- a **classification problem** if the output label y can only take on a discrete number of values i.e. $\mathcal{Y} = \{c_1, \dots, c_K\}$ for $K < \infty$. In this case, h is typically called a **classifier**.

2.1 Probability to Model Uncertainty

Supervised learning is interesting because the response may not be a deterministic function of the inputs. In such a case, one can employ a statistical model that makes assumptions about the nature of the data-generating phenomenon under investigation.

A good first step is always to plot the data $\mathcal{T}$ to see if there's some obvious relationship between the inputs and their respective responses. From this, we can build up some picture of what a good hypothesis h relating them could look like.

Example 2.1.1

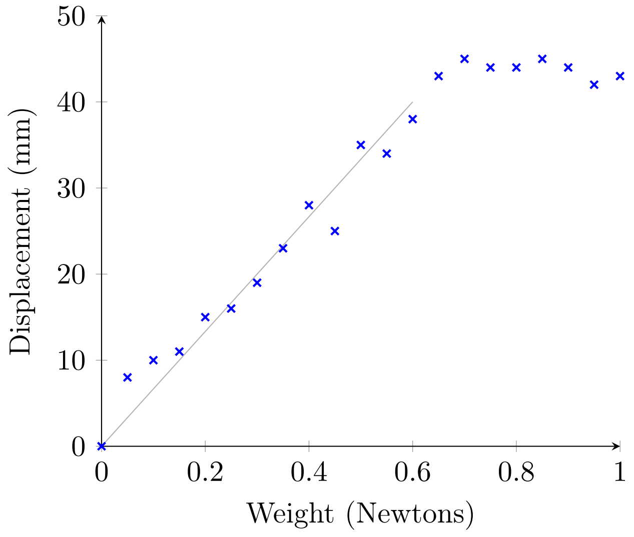


Figure 2.1: Plotting the relationship between the displacement of a spring and the weight placed on it to investigate Hooke's Law. The weight values live in $\mathcal{X}$, and displacement in $\mathcal{Y}$.

From the above data, we could conceivably use some form of positive square root (or even some linear transformation of a logarithmic) function for h. There are many such candidate functions h and we may call their collective $\mathcal{H}$ a **hypothesis class**.

Let's say we chose to use the collection of linear transformations of positive square root functions $\mathcal{H} = \{a + b\sqrt{x+c} \text{ where } x \geqslant c \colon a, b, c \in \mathbb{R}\}$. We may write each hypothesis as $h_{(a,b,c)}(x)$ and so the triple (a, b, c) parameterises $\mathcal{H}$.

Definition 2.1.2 In general, we write

$$\mathcal{H} := \{h_{\boldsymbol{\theta}} \colon \boldsymbol{\theta} \in \Theta\}$$

for the **parametric hypothesis class** depending on a parameter $\boldsymbol{\theta} \in \Theta$.

The choice of $\mathcal{H}$ limits the scope of functions that we can search through (via the parameters) in order to find the best hypothesis that approximates the true relationship between inputs and responses. It's clear to see that this choice has far-reaching consequences for all subsequent analysis.

Going back to the plotted data, we also observe that the data-collection method is almost never perfect, so there will be some random error in the observed values. This can be seen in the figure by the data being distributed somewhat randomly about a positive linear trend (represented by the straight line) which plateaus at around 0.6 Newtons. Incorporating such uncertainty is another motivator for the use of a statistical model, written in the language of probability.

2.1.1 Probabilistic Model

We require a few definitions to properly state the assumptions being made.

Definition 2.1.3

- Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. This will model the randomness governing the phenomenon from which we wish to sample our data.
- Let $(E, \mathcal{E})$ be another measurable space. A **random element** is an $(\mathcal{F}, \mathcal{E})$-measurable map $X \colon \Omega \to E$ i.e. $\forall B \in \mathcal{E},\ X^{-1}(B) \in \mathcal{F}$. Such maps mathematically formalise the process of drawing a single observable (number, vector etc. e.g. height) corresponding to an outcome (e.g. the drawing of a student from a class).
    - If $(E, \mathcal{E}) = (\mathbb{R}, \mathcal{B}_{\mathbb{R}})$, then X is a real-valued **random variable**.
    - If $(E, \mathcal{E}) = (\mathbb{R}^n, \mathcal{B}_{\mathbb{R}^n})$, then X is a real-valued **random vector**.
- A **random sample** of size n is a collection of random variables $X_i \colon (\Omega, \mathcal{F}) \to (E, \mathcal{E})$ defined on the same probability space $(\Omega, \mathcal{F}, \tilde{\mathbb{P}})$ that are mutually $\tilde{\mathbb{P}}$-independent[1], and identically distributed with distribution $\mathbb{P}$ s.t. $\tilde{\mathbb{P}} = \otimes_n \mathbb{P}$. We denote this by
$$X_i \overset{\text{i.i.d.}}{\sim} \mathbb{P}.$$

Let $(\mathbf{X}, Y) \colon \Omega \to \mathcal{X} \times \mathcal{Y}$ be the random element representing the process of observing one input-output pair (x, y) from an outcome realised by the underlying phenomenon (which is governed by Ω). Henceforth, I am going to let $\mathbb{P}$ denote the distribution governing the underlying phenomenon, and so the push-forward measure $(\mathbf{X}, Y)_\sharp \mathbb{P}$ (which I will denote by $\mathbb{P}_{(\mathbf{X},Y)}$) is the true distribution governing the observation of input-output pairs (x, y).

The assumption we will repeatedly make throughout is that our training set $\mathcal{T}$ is a realisation of a random sample
$$(\mathbf{X}^{(1)}, Y^{(1)}), \ldots, (\mathbf{X}^{(n)}, Y^{(n)}) \overset{\text{i.i.d.}}{\sim} \mathbb{P}_{(\mathbf{X},Y)}.$$
This is called the **random design setup**.

In particular, each $(\mathbf{X}^{(i)}, Y^{(i)}) \colon \Omega \to \mathcal{X} \times \mathcal{Y}$ is a random element and its realisation in our training set is $(x^{(i)}, y^{(i)})$.

[1]The reason for using $\tilde{\mathbb{P}}$ is a technical one which involves a suppression of notation. We wish for our underlying probability measure $\tilde{\mathbb{P}}$ to be able to support statements about several random variables so it's defined on a σ-algebra on Ω, and Ω can be thought of as the n-fold product of the outcome space Ω of each single run of the experiment (which we represent by X_i). Accordingly, we redefine X_i to be defined on $\tilde{\Omega}$ as the indicator of the i^{th} component. Thus, when we write Ω in our definition, we are writing it as a placeholder for $\tilde{\Omega}$. For more on this, see **Chapter 13** from [1].

2.2 Loss and Risk

Notation 2.2.1 Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and $X \colon \Omega \to \mathbb{R}$ be a random variable i.e. X is $\mathcal{F}$-measurable.

- I use ordinary parentheses for the expectation of X
$$\mathbb{E}(X) = \int_\Omega X(\omega)\,\mathrm{d}\mathbb{P}(\omega),$$
- and square parentheses for random variables e.g.
  - The conditional expectation $\mathbb{E}[X \,|\, \mathcal{G}]$ of X given a sub-σ-algebra $\mathcal{G} \subseteq \mathcal{F}$. Any random variable Z is called a conditional expectation of X given $\mathcal{G}$ if Z is $\mathcal{G}$-measurable, and satisfies the averaging property i.e. for every $B \in \mathcal{G}$:
  $$\mathbb{E}(\mathbb{1}_B Z) = \mathbb{E}(\mathbb{1}_B \mathbb{E}[X \,|\, \mathcal{G}]).$$
  - The conditional probability of $A \in \mathcal{F}$ given $\mathcal{G}$ is denoted by $\mathbb{P}[A \,|\, \mathcal{G}]$
  - I write $\mathrm{Var}[X \,|\, \mathcal{G}]$ to represent the conditional variance (as seen later).

By virtue of the randomness in our assumptions so far, making (almost) any choice of hypothesis will yield predictions $h(x^{(i)})$ that fall short from their respective observed labels $y^{(i)}$ for $x^{(i)}$ i.e. we incur a loss.

Definition 2.2.2 The loss[2] of a particular hypothesis $h \in \mathcal{H}$ associated with a particular training example $(x^{(i)}, y^{(i)})$ is made precise as a non-negative real-valued function $L_h \colon \mathcal{X} \times \mathcal{Y} \to [0, +\infty)$

$$(x^{(i)}, y^{(i)}) \longmapsto L_h(x^{(i)}, y^{(i)}) := L(h(x^{(i)}), y^{(i)})$$

which quantifies how good our hypothesis is at predicting $y^{(i)}$ from $x^{(i)}$. The function L that maps from $\mathcal{Y} \times \mathcal{Y} \to [0, +\infty)$ is sometimes also called a loss function.

In statistics, it's common practice to average the losses over an entire population:

Definition 2.2.3 The deterministic number $R(h)$, called the statistical risk, defined by the following expected value

$$\begin{aligned} R(h) := \mathbb{E}(L(h(\mathbf{X}), Y)) &:= \int_\Omega L(h(\mathbf{X}(\omega)), Y(\omega))\,\mathrm{d}\mathbb{P}(\omega) \\ &= \int_{\mathcal{X}\times\mathcal{Y}} L(h(\boldsymbol{x}), y)\,\mathrm{d}\mathbb{P}_{(\mathbf{X},Y)}(\boldsymbol{x}, y) \end{aligned}$$

captures how bad our hypothesis is on average in predicting the label of an unseen input.

Remarks 2.2.4

- The expectation is calculated with respect to the true distribution of the data i.e. under the assumption that we know $\mathbb{P}_{(\mathbf{X},Y)}$.
- We think of the statistical risk as the composition $R(h) = \mathbb{E}(L_h \circ (\mathbf{X}, Y))$. Note that the domains and codomains match up properly.
  - $(\mathbf{X}, Y) \colon \Omega \to \mathcal{X} \times \mathcal{Y}$
  - $L_h \colon \mathcal{X} \times \mathcal{Y} \to [0, +\infty)$

  So $L_h \circ (\mathbf{X}, Y) \colon \Omega \to [0, +\infty)$ is a measurable function (assuming that h and L are appropriately measurable) that we can take the expectation of.

[2]Sometimes a problem requires one to maximise $-L$, in which case $-L$ is called a reward, or profit function. Each subfield has its own terminology but the same idea persists.

- A natural follow-up question to ask is: Given this knowledge of the joint distribution of $(\mathbf{X}, Y)$, what is the minimal achievable expected loss, and at which hypothesis is it attained?

A function that minimises the statistical risk would be the best hypothesis h available with respect to the given loss L. We denote this minimiser by h^* and give it a name:

Definition 2.2.5 The Bayes predictor h^* is the optimal solution to $\min_h R(h)$.

By the tower law of conditional expectation,

$$R(h) = \mathbb{E}(\mathbb{E}[L(h(\mathbf{X}), Y) \mid \sigma(\mathbf{X})])$$

and the inner expression (the conditional expectation) is a $\sigma(\mathbf{X})$-measurable random variable. By the Doob-Dynkin[3] Representation Theorem, there exists some measurable[4] $r_h \colon \mathcal{X} \to \mathbb{R}$ s.t.

$$\mathbb{E}[L(h(\mathbf{X}), Y) \mid \sigma(\mathbf{X})] = r_h(\mathbf{X}).$$

The map r_h is called the conditional expectation function, and we define it pointwise by

$$r_h(x) = \mathbb{E}[L_h(h(\mathbf{X}), Y) \mid \mathbf{X} = x]$$

where the output is understood via the disintegration theorem associated with the conditional law of $L_h \circ (\mathbf{X}, Y)$ given $\mathbf{X}$. Slightly more detail about the reasons for this are below:

Wildly ambitious attempt to summarise conditional probability

- σ-algebras can be thought of as information pertaining to random variables. A random variable X being $\mathcal{F}$-measurable means that the information in $\mathcal{F}$ is enough to completely characterise X.
- The conditional expectation of X given a sub-σ-algebra $\mathcal{G} \subseteq \mathcal{F}$ is "the" best $\mathcal{G}$-measurable approximation of X. (We remark that it's only unique $\mathbb{P}$-a.s. because an integral-based equation is its defining property.)
- We define the conditional probability of $A \in \mathcal{F}$ given $\mathcal{G}$ by the conditional expectation of $\mathbb{1}_A$ given $\mathcal{G}$. The random variable $\mathbb{1}_A$ encodes whether or not A occurs. This r.v. may or may not be something we can determine from the information $\mathcal{G}$ alone. If we can't, then we turn to $\mathbb{P}[A \mid \mathcal{G}]$ as the best $\mathcal{G}$-measurable approximation to A.
  - $\mathbb{P}[A \mid G]$ can be thought of as encoding the updated belief about A given the partial information $\mathcal{G}$, from which we can reproduce (via the defining property of conditional expectation) the probabilities $\mathbb{P}(A \cap B)$ for $B \in \mathcal{G}$ i.e.

$$\begin{aligned}\mathbb{P}(A \cap B) &= \mathbb{E}(\mathbb{1}_B \mathbb{1}_A) \\ &= \mathbb{E}(\mathbb{1}_B \mathbb{E}[\mathbb{1}_A \mid \mathcal{G}]) \\ &= \mathbb{E}(\mathbb{1}_B \mathbb{P}[A \mid \mathcal{G}]) \\ &= \int_B \mathbb{P}[A \mid \mathcal{G}](\omega) \, \mathrm{d}\mathbb{P}(\omega).\end{aligned}$$

- We can extend this concept to every $A \in \mathcal{F}$, and define the conditional probability on $\mathcal{F}$ given $\mathcal{G}$ as the map $\kappa \colon \mathcal{F} \times \Omega \to [0,1]$ defined for every $A \in \mathcal{F}$ by

$$\omega \longmapsto \kappa(A, \omega) := \mathbb{P}[A \mid \mathcal{G}](\omega).$$

- If we look to the other half of this map, fixing $\omega \in \Omega$, we would like to have a probability measure for each ω. The historical reasoning for this seems to be that we can use κ to model a stochastic process where each ω can be considered a starting "state" and so $\kappa(\cdot, \omega)$ would be a probability measure that we can use to model how we arrive at the next state.

[3]A more detailed version of this theorem is Theorem A.42 from A.42 [2, p. 587].

[4]Indeed, suppose that $(\mathcal{X}, \mathcal{E})$ is the measurable space that $\mathbf{X} \colon \Omega \to \mathcal{X}$ maps into. Then the function here is more precisely defined as $r_h \colon \operatorname{Im}(\mathbf{X}) \to \mathbb{R}$ and is $\mathcal{E}|_{\mathbf{X}(\Omega)}$-$\mathcal{B}_{\mathbb{R}}$-measurable, where $\mathcal{E}|_{\mathbf{X}(\Omega)}$ is the trace σ-algebra of $\mathcal{E}$ on $\mathbf{X}(\Omega)$.

- A technical problem arises. For every $\kappa(\cdot,\omega)$ to be a probability measure, we demand countable additivity of each map. If we demand countable additivity from $\kappa(\cdot,\omega)=\mathbb{P}[\cdot\,|\,\mathcal{G}](\omega)$ **for every** ω, then we have potentially uncountably many null sets to exclude for all of these equalities to hold. Such a union may be non-null. To rectify this, we define what is known as a regular conditional probability on $\mathcal{F}$ given $\mathcal{G}$ as a map for which:
  - **For every** $\omega\in\Omega$, the map $\kappa(\cdot,\omega)$ is a probability measure on $(\Omega,\mathcal{F})$.
  - For every $A\in\mathcal{F}$, the map $\kappa(A,\cdot)$ is a version of $\mathbb{P}[A\,|\,\mathcal{G}](\omega)$.
- Given that an r.c.p. on $\mathcal{F}$ given $\mathcal{G}$ exists, we may re-write the defining property of $\kappa(A,\cdot)$ in the suggestive form

$$\begin{aligned}\mathbb{E}(\mathbb{1}_B\mathbb{1}_A)&=\mathbb{E}(\mathbb{1}_B\kappa(A,\cdot))\\&=\int_B\kappa(A,\omega)\,\mathrm{d}\mathbb{P}(\omega)\\&=\int_B\int_\Omega\mathbb{1}_A(\omega')\,\mathrm{d}\kappa(\cdot,\omega)(\omega')\,\mathrm{d}\mathbb{P}(\omega)\end{aligned}$$

  which expresses that the conditional expectation $\kappa(A,\cdot)=\mathbb{P}[A\,|\,\mathcal{G}](\cdot)=\mathbb{E}[\mathbb{1}_A\,|\,\mathcal{G}](\cdot)$ is equal $\mathbb{P}$-a.s. to the Lebesgue integral of $\mathbb{1}_A$ with respect to the probability measure $\kappa(\cdot,\omega)$.
- This can be extended from indicators $\mathbb{1}_A$ to any $X\in L^1(\Omega,\mathcal{F},\mathbb{P})$ and so we have

$$\mathbb{E}[X\,|\,\mathcal{G}]=\int_\Omega X(\omega')\,\mathrm{d}\kappa(\cdot,\omega)(\omega')\qquad\text{for }\mathbb{P}\text{-a.e. }\omega\in\Omega.$$

- Now we focus on the case where $\mathcal{G}$ is generated by a random variable $Y\colon(\Omega,\mathcal{F})\to(E,\mathcal{E})$ i.e. $\mathcal{G}=\sigma(Y)$. By the Doob-Dynkin representation theorem, since $\kappa(A,\cdot)=\mathbb{P}[A\,|\,\sigma(Y)](\cdot)=\mathbb{E}[\mathbb{1}_A\,|\,\sigma(Y)](\cdot)$ is $\sigma(Y)$-measurable, it factors through Y to give a re-parameterisation of $\kappa\colon\mathcal{F}\times\Omega$ to $\kappa_Y:\mathcal{F}\times E\to[0,1]$.
- This regular conditional probability on $\mathcal{F}$ generated by Y (i.e. given $\mathcal{G}=\sigma(Y)$) is a map $\kappa_Y:\mathcal{F}\times E\to[0,1]$ satisfying the following:
  - For every $y\in E$, $\kappa_Y(\cdot,y)$ is a probability measure on $\mathcal{F}$
  - For each $A\in\mathcal{F}$, the mapping $y\longmapsto\kappa_Y(A,y)$ is
    - an $\mathcal{E}$-measurable function satisfying for $\mathbb{P}$-a.e. $\omega\in\Omega$:

$$\kappa_Y(A,Y(\omega))=\mathbb{P}[A\,|\,\sigma(Y)](\omega),$$

    - and is $\mathbb{P}_Y$-integrable, satisfying the following disintegration formula for all $A\in\mathcal{F}$ and $D\in\mathcal{E}$:

$$\mathbb{P}(A\cap Y^{-1}(D))=\int_D\kappa_Y(A,y)\,\mathrm{d}\mathbb{P}_Y(y).$$

- This disintegration formula can be re-written in a way that generalises handily for any integrable random variable X and $D\in\mathcal{E}$:

$$\int_{Y^{-1}(D)}X(\omega)\,\mathrm{d}\mathbb{P}(\omega)=\int_D\left(\int_\Omega X(\omega')\,\mathrm{d}\mathbb{P}^y(\omega')\right)\mathrm{d}\mathbb{P}_Y(y).$$

- From this, there's a bit more re-writing (noticing that the LHS is an expectation of X against the indicator $\mathbb{1}_{Y^{-1}(D)}$, and so we may use the defining property of conditional expectation to write

$$\int_{Y^{-1}(D)}\mathbb{E}[X\,|\,\sigma(Y)]\,\mathrm{d}\mathbb{P}(\omega)=\int_D\left(\int_\Omega X(\omega')\,\mathrm{d}\mathbb{P}^y(\omega')\right)\mathrm{d}\mathbb{P}_Y(y).$$

- By the Doob-Dynkin theorem, the integrand on the LHS can be written as a function h_Y of Y. Indeed, this h_Y is the pointwise conditional expectation function and that reveals itself by a change of variables to get the above equality in terms of integrals with respect to $\mathbb{P}_Y$!

Definition 2.2.6 We call the assignment h_Y the **pointwise conditional expectation function** of X given $Y=y$, and we also denote it by

$$y\longmapsto h_Y(y):=\int_\Omega X(\omega')\,\mathrm{d}\mathbb{P}^y(\omega')=:\mathbb{E}[X\,|\,Y=y].$$

The rest of the development of theory about Bayes predictors and Bayes risk.

Need to re-write this bit:

Of course in practice we don't know the joint distribution $\mathbb{P}_{(\mathbf{X},Y)}$ of our population so we can't compute $R(h)$ directly, but we do have access to our training data $\mathbb{T}$ drawn i.i.d. from a random sample assumed to have the true distribution.
Instead, we use our aforementioned random sample to define an estimator called the **empirical risk**:

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(\mathbf{X}^{(i)}), Y^{(i)}).$$

The $(\mathbf{X}^{(i)}, Y^{(i)})$ are still i.i.d. with distribution $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ but now we have the possibility of using the realisations of the random sample (i.e. the training data $\mathcal{T}$) to compute the empirical risk

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$$

which is just the average loss on the training set.

CHAPTER 3
Regression

Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \mathbb{R} = (-\infty, +\infty)$.

Regression is a term used as a shorthand for regression analysis — a series of statistical techniques developed to investigate the relationship between a collection of (one or more) independent/predictor variables (the regressors) $X_1, \dots, X_p$, and a[1] real-valued dependent/response variable (the regressand) Y.

3.1 Setup

As established in the previous section, we assume that there is some true joint distribution between predictors and response, represented by $(\mathbf{X}, Y) \sim \mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$. Since our goal is to optimally predict the response Y given the predictors $\mathbf{X}$, we let $f\colon \mathcal{X} \to \mathcal{Y}$ denote a deterministic function s.t.

$$f(\mathbf{X}) = \mathbb{E}[Y \mid \mathbf{X}].$$

Since $\mathbf{X}$ and $\mathbf{Y}$ are random variables[2], there will naturally be some error between the prediction $f(\mathbf{X})$ and Y. We denote this error by the random variable[3]

$$\varepsilon := Y - f(\mathbf{X}) = Y - \mathbb{E}[Y \mid \mathbf{X}].$$

It immediately follows that we may re-write this equality in the familiar form

$$Y = f(\mathbf{X}) + \varepsilon,$$

but more importantly it follows that

$$\begin{aligned}
\mathbb{E}[\varepsilon \mid \mathbf{X}] &= \mathbb{E}[Y - f(\mathbf{X}) \mid \mathbf{X}] \\
&= \mathbb{E}[Y \mid \mathbf{X}] - \mathbb{E}[f(\mathbf{X}) \mid \mathbf{X}] \\
&= \mathbb{E}[Y \mid \mathbf{X}] - f(\mathbf{X}) \\
&= \mathbb{E}[Y \mid \mathbf{X}] - \mathbb{E}[Y \mid \mathbf{X}] \\
&= 0.
\end{aligned}$$

This property $\mathbb{E}[\varepsilon \mid \mathbf{X}] = 0$ is called strict exogeneity. It means that our errors have zero mean conditional on the regressors $\mathbf{X}$.

3.2 Estimating the Regression Function

Since we don't know $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$, we can't calculate $\mathbb{E}[Y \mid \mathbf{X}]$ directly. Instead, we must estimate f by choosing a hypothesis[4] h from a (suitable) class of candidate functions $\mathcal{H}$ called the **hypothesis class**. Then we shall use our sampled data $\mathcal{T}$ to optimise h in a way that best approximates $\mathbb{E}[Y \mid \mathbf{X}]$.

[1]Strictly speaking, this would be a multivariable univariate regression — variable refers to predictors, and variate refers to responses.
[2]Does Y need to be L^1 for the conditional expectation to exist?
[3]Can this error be thought of as the randomness in Y, on average, that isn't captured by $\mathbf{X}$?
[4]We will use this function to make predictions for unseen examples $x \in \mathcal{X}$.

For regression, a common choice[5] of loss function is the **squared/quadratic loss** $L(h(\mathbf{X}), Y) = (h(\mathbf{X}) - Y)^2$. For quadratic loss, the statistical risk of h is $R(h) := \mathbb{E}\big((Y - h(\mathbf{X}))^2\big)$ and is called the **mean squared error** (**MSE** for short). The assumptions that Y and $h(\mathbf{X}) \in L^2$ are required for this minimisation problem to be meaningful.

A function that minimises the MSE would be the best hypothesis available under quadratic loss. This minimiser is denoted by

$$h^* := \operatorname*{argmin}_h \mathbb{E}\Big((Y - h(\mathbf{X}))^2\Big).$$

Does such a function exist, and if so, which form does it take? By conditioning on $\mathbf{X}$, we can rewrite the MSE using the law of total expectation

$$\mathbb{E}\Big((Y - h(\mathbf{X}))^2\Big) = \mathbb{E}\Big(\mathbb{E}\Big[(Y - h(\mathbf{X}))^2 \,\Big|\, \mathbf{X} = x\Big]\Big).$$

It suffices to minimise the argument of the expectation on the RHS pointwise because given $\mathbf{X} = x$, $h(\mathbf{X})$ is equal to some constant $h(x) =: c$ so we concern ourselves with:

$$h^*(x) = \operatorname*{argmin}_c \mathbb{E}\Big[(Y - c)^2 \,\Big|\, \mathbf{X} = x\Big]$$

This argmin is actually equal to the conditional expectation $\mathbb{E}[Y \mid \mathbf{X} = x]$ of Y given $\mathbf{X} = x$.

There are three ways to verify this:

1. Add and subtract $\mathbb{E}(Y \mid \mathbf{X})$ from the statistical risk expression and expand.
2. Observe that
$$L^2(\Omega, \mathbb{P}) := \{\text{random variables } Y : \Omega \to \mathbb{R}$$
such that $\mathbb{E}\big(|Y|^2\big) < \infty\}$ is a vector space and when equipped with the inner product $\langle X, Y \rangle = \mathbb{E}(XY)$ becomes a Hilbert space. Then we can use the theory of orthogonal projections to prove that $\mathbb{E}(Y \mid \mathbf{X}) := \mathbb{E}(Y \mid \sigma(\mathbf{X}))$ (where $\sigma(X)$ is the sigma-algebra generated by $\mathbf{X}$) is equal to $\pi_{\sigma(\mathbf{X})}$ ($\mathbb{P}$-almost surely) where
$$\pi_{\sigma(\mathbf{X})} : L^2(\Omega, \mathcal{A}, \mathbb{P}) \to L^2(\Omega, \sigma(\mathbf{X}), \mathbb{P})$$
is the projection map defined by
$$\|Y - \pi_{\sigma(\mathbf{X})}(Y)\| = \inf\{\|Y - Z\| : Z \in L^2(\Omega, \sigma(\mathbf{X}), \mathbb{P})\}$$
where $\sigma(\mathbf{X})$ is a sub-algebra of $\mathcal{A}$. In summary, this is a fancy way to say that the geometry of Hilbert space tells us that minimisation in said space becomes a problem of orthogonal projection.
3. A differentiation argument.

This minimiser $h^*(x) = \mathbb{E}[Y \mid \mathbf{X} = x]$ coincides with f, and so the terms **regression function** and **conditional expectation function (CEF)** are synonymous.

As stated at the top of this section, we don't know the (joint and hence) conditional distribution of Y given $\mathbf{X}$. We can get around this by modelling the regression function as some simple functional form h and then use the training data to minimise the empirical risk instead. In the case that the regression function h is parametric i.e. $h = h_{\boldsymbol{\theta}}$, we can search for

$$\operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \Big(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})\Big)^2.$$

This is an example of **parametric learning**. Below we discuss a particular choice of $\mathcal{H}$ consisting of "linear" hypotheses.

[5]Why quadratic loss in the first place? It feels a bit like the choice was justified in hindsight (which I don't take issue with but I can't really figure it out).

- Develop the method
- Realise that convergence issues happen in the algorithms you devise and then
- go back to alter the loss to have properties that make things more convenient e.g. convexity.

3.3 Linear Regression

So far there's been no restriction on how to represent the hypothesis h i.e. $\mathcal{H}$ is a very broad class of functions. We'd like to, in some meaningful way, limit the family of hypotheses over which we'll learn. Making assumptions about the functional form of h will restrict $\mathcal{H}$. This section will assume that h is a linear hypothesis — the simplest form.

A **linear hypothesis** takes the form of an affine function

$$h_{\boldsymbol{\theta}}(\mathbf{X}) = \theta_0 + \theta_1 X_1 + \cdots + \theta_p X_p,$$

where we collect the unknown quantities θ_j, called the **parameters** (or **weights**) of our model, into a vector

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_p \end{bmatrix} \in \mathbb{R}^{p+1}.$$

The linearity of our hypothesis refers to **linearity in the coefficients** θ_j.

Example 3.3.1

- $h_{\boldsymbol{\theta}}(X) = \theta_0 + \theta_1 X^3$ is linear in $\boldsymbol{\theta} = \begin{bmatrix}\theta_0 & \theta_1\end{bmatrix}^\top \in \mathbb{R}^2$,
- but $h_{\boldsymbol{\theta}}(X) = \theta_0 + \sin(\theta_1) X$ is not linear in $\boldsymbol{\theta}$.

Such a function $h_{\boldsymbol{\theta}}$ is also called a **linear predictor**, or a **discriminant function**. For notational convenience, one can define an intercept term $X_0 = 1$ so that h may be re-written as

$$h_{\boldsymbol{\theta}}(\boldsymbol{X}) = \langle \boldsymbol{\theta}, \mathbf{X} \rangle := \sum_{i=0}^{p} \theta_i X_i = \mathbf{X}^\top \boldsymbol{\theta},$$

where $\boldsymbol{X} = (X_0, X_1, \ldots, X_p)$ is defined by $\mathbf{X}(\omega) = \begin{bmatrix} 1 \\ X_1(\omega) \\ \vdots \\ X_p(\omega) \end{bmatrix} \in \mathbb{R}^{p+1}$, and we call θ_0 a **bias** term.

A nice way to interpret a linear hypothesis is by the expression $\partial_j h(\boldsymbol{X}) = \theta_j$ i.e. holding all other predictors fixed, θ_j represents the effect on h of a one unit increase in X_j. This behaviour characterises a linear-response model.

A **linear response model** is suitable for situations where the response variable can vary, to a good approximation, indefinitely in either direction. More generally, a linear-response model works for any quantity that only varies by a relatively[6] small amount compared to the variation in the predictive values e.g. human height.

These assumptions aren't suitable for some types of response variables and this will be discussed later under GLMs as a flexible generalisation of linear regression.

Here I collect the above assumptions to define a linear regression model for the conditional expectation of Y given $\mathbf{X}$.

[6]The same as linear approx in analysis: $\Delta x \approx 0 \implies \Delta y = f(x + \Delta x) - f(x) \approx 0$.

Let $(\mathbf{X}, Y) \sim \mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$. A (population-level) **linear regression model** is a regression model

$$Y = f(\mathbf{X}) + \varepsilon = \mathbb{E}[Y \mid \mathbf{X}] + \varepsilon \quad \text{s.t.} \quad \mathbb{E}[\varepsilon \mid \mathbf{X}] = 0$$

that assumes linearity of the conditional mean i.e. that the conditional expectation of Y given $\mathbf{X}$ is a linear hypothesis h i.e. that there exists a vector of parameters $\boldsymbol{\theta} = \begin{bmatrix}\theta_0 & \cdots & \theta_p\end{bmatrix}^\top$ s.t.

$$\mathbb{E}[Y \mid \mathbf{X}] = h_{\boldsymbol{\theta}}(\mathbf{X}) = \mathbf{X}^\top \boldsymbol{\theta}.$$

On the sample-level, we introduce more assumptions that allow us to relate how our data

$$\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_1^n$$

is related to the population. As usual, we assume that our training set is a realisation of a random sample

$$(\mathbf{X}^{(1)}, Y^{(1)}), \dots, (\mathbf{X}^{(n)}, Y^{(n)}) \overset{\text{i.i.d.}}{\sim} \mathbb{P}_{\mathcal{X}\times\mathcal{Y}}.$$

We can think of these as identical copies of $(\mathbf{X}, Y)$ so the population assumptions hold for each $(\mathbf{X}^{(i)}, Y^{(i)})$. The linearity assumption can be written in matrix-form to summarise the n equations into one.

For the sake of simplifying notation, we'll collect the predictor random vectors into a single random vector (of random vectors)

$$\mathsf{X} = (\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}) \colon (\Omega, \mathcal{F}) \to ((\mathbb{R}^{p+1})^n, \mathcal{B}_{(\mathbb{R}^{p+1})^n}).$$

A realisation of X is given by

$$\mathsf{X}(\omega) = \begin{bmatrix}\mathbf{X}^{(1)}(\omega) \\ \vdots \\ \mathbf{X}^{(n)}(\omega)\end{bmatrix} = \begin{bmatrix}x^{(1)} \\ \vdots \\ x^{(n)}\end{bmatrix}$$

Statistics textbooks present $\mathsf{X}(\omega)$ in a more presentable form by identifying this vector of column vectors $(\mathbb{R}^{p+1})^n$ with a matrix in $\mathbb{R}^{n\times(p+1)}$. Doing so allows one to succinctly represent linearity over all the training examples (a system of n equations) as a matrix equality. I'll denote the map that makes such an identification[7] by Φ so a realisation of $\mathsf{X}(\omega)$ is identified with the matrix

$$\mathfrak{X} := \Phi(\mathsf{X}(\omega)) = \begin{bmatrix}(x^{(1)})^\top \\ \vdots \\ (x^{(n)})^\top\end{bmatrix} = \begin{bmatrix}1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_p^{(n)}\end{bmatrix}.$$

This matrix $\mathfrak{X}$ whose rows are the input features of each example in $\mathcal{T}$ is called the **design matrix** of our model. When I need to speak about conditioning on a random variable I will refer to X, and when I wish to speak of systems of equations (like I will immediately below) I will refer to $\mathfrak{X}$. On the level of random variables, we have that

$$\Phi(\mathsf{X}) = \Phi(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}) = \begin{bmatrix}(\mathbf{X}^{(1)})^\top \\ \vdots \\ (\mathbf{X}^{(n)})^\top\end{bmatrix}.$$

The rest of the notation is straight-forward. Simply collect the responses and errors into their own respective random vectors $\mathbf{Y}$ and $\boldsymbol{\varepsilon}$, and denote their realisations by labels $\boldsymbol{y}$ and residuals $\boldsymbol{\epsilon}$:

$$\mathbf{Y} = \begin{bmatrix}Y^{(1)} \\ \vdots \\ Y^{(n)}\end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix}y^{(1)} \\ \vdots \\ y^{(n)}\end{bmatrix}, \ \boldsymbol{\varepsilon} = \begin{bmatrix}\varepsilon^{(1)} \\ \vdots \\ \varepsilon^{(n)}\end{bmatrix}, \text{ and } \boldsymbol{\epsilon} = \begin{bmatrix}\epsilon^{(1)} \\ \vdots \\ \epsilon^{(n)}\end{bmatrix}.$$

[7] I believe no information is lost by this identification because the σ-algebras generated by both are the same, just re-labelled, and so the ensuing conditional expectation calculations would be the same given either $\sigma(\mathsf{X})$ or $\sigma(\Phi(\mathsf{X}))$.

It follows that

$$\Phi(\mathsf{X})\boldsymbol{\theta} = \begin{bmatrix} (\mathbf{X}^{(1)})^\top \boldsymbol{\theta} \\ \vdots \\ (\mathbf{X}^{(n)})^\top \boldsymbol{\theta} \end{bmatrix}.$$

Finally, we may write all n equations as[8]

$$\mathbf{Y} = \Phi(\mathsf{X})\boldsymbol{\theta} + \boldsymbol{\varepsilon},$$

and so a realisation $\mathcal{T}$ of the experiment that these random variables formalise can be represented by the system:

$$\boldsymbol{y} = \mathcal{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}.$$

3.3.1 Extra Assumptions

2. Strict exogeneity: The errors have 0 mean conditional on the features i.e. for $i = 1, \ldots, n$:

$$0 = \mathbb{E}\left[\varepsilon^{(i)} \,\middle|\, \sigma(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)})\right].$$

3. The rank of the $n \times (p+1)$ regression matrix, X, is $p+1$ with probability 1.

4. Constant conditional second moment of errors given predictors:
An assumption that the variances of all the error terms are equal (or similar):

$$\mathbb{E}\left[(\varepsilon^{(i)})^2 \,\middle|\, \sigma(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)})\right] = \sigma^2 > 0.$$

5. No correlation between observations:

$$\mathbb{E}\left[\varepsilon^{(i)}\varepsilon^{(j)} \,\middle|\, \sigma(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)})\right] = 0 \qquad \text{for } i, j = 1, \ldots, n \text{ where } i \neq j$$

Assumption 3 is left stated in the form Hayashi does (with respect to fixed design i.e. in the situation where one has the design matrix X after observing the random sample). I'm not yet sure what the random regressors version of the statement is.

Definition 3.3.2 Let A be an $m \times n$ matrix with real entries.

- The **column space** of A, denoted by colspace(A), is the subspace of $\mathbb{R}^m$ spanned by the columns of A.
- The **column rank** of A is the dimension of the column space of A. Equivalently, the column rank is equal to the size of the largest linearly independent subset of its columns.

Analogous definitions hold if one replaces 'column' with 'row'. It doesn't really matter which we use because:

Theorem 3.3.3 The row rank and column rank of a matrix A are equal. Thus, we simply refer to the **rank** of a matrix.

Furthermore, $\text{rank}(A) \leqslant \min(m, n)$.

[8]Not only is this succinct but by *vectorising* the data in a programming language, we also benefit from linear algebra being well-optimised in built-in language packages/modules like `numpy` in Python. Such packages, in general, yield faster results than basic `for`-loops.

Remarks 3.3.4

- Since we are assuming that our sample is i.i.d., the strict exogeneity assumption is sometimes written as conditioning on only one of the $\mathbf{X}^{(i)}$ instead of on the whole of $\sigma(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)})$. It turns out that these are equivalent.

  Proof. Let $\mathcal{G}_n := \sigma(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)})$. Then $\sigma(\mathbf{X}^{(i)}) \subseteq \mathcal{G}_n$, and by the tower property:

  $$\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \sigma(\mathbf{X}^{(i)})\Big] \,\Big|\, \mathcal{G}_n\Big] = \mathbb{E}(0 \,|\, \mathcal{G}_n) = 0$$

  For the reverse implication, assume that $\mathbb{E}\big[\varepsilon^{(i)} \,|\, \mathcal{G}_n\big] = 0$ and run the above argument backwards to conclude that $\mathbb{E}\big[\varepsilon^{(i)} \,|\, \sigma(\mathbf{X}^{(i)})\big] = 0$. ■

  Economists prefer to write "$|\, \mathcal{G}_n$" because it inspires notation like $\mathbb{E}(\varepsilon \,|\, \mathcal{X})$ (found in Hayashi [3]) which is read as "Fix the realised design matrix. Then the errors have mean zero."

- When we combine assumptions 2 and 4, we get conditional homoskedasticity of our error terms:

  Definition 3.3.5 The assumption of constant conditional variance

  $$\mathrm{Var}\Big[\varepsilon^{(i)} \,\Big|\, \sigma(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)})\Big] = \sigma^2 > 0$$

  of our errors given the predictors $\mathbf{X}$ is called **conditional homoskedasticity**.

  Proof. For short, let $\sigma(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}) =: \mathcal{G}_n$. Then,

  $$\begin{aligned}
  \mathrm{Var}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big] &= \mathbb{E}\Big[\Big(\varepsilon^{(i)} - \mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big]\Big)^2 \,\Big|\, \mathcal{G}_n\Big] \\
  &= \mathbb{E}\Big[(\varepsilon^{(i)})^2 - \varepsilon^{(i)}\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big] + \mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big]^2 \,\Big|\, \mathcal{G}_n\Big] \\
  &= \mathbb{E}\Big[(\varepsilon^{(i)})^2 \,\Big|\, \mathcal{G}_n\Big] - \underbrace{\mathbb{E}\Big[\varepsilon^{(i)}\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big] \,\Big|\, \mathcal{G}_n\Big]}_{\mathbb{E}[g(\mathbf{X}) \,|\, \mathcal{G}_n] = g(\mathbf{X})} + \mathbb{E}\Big[\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big]^2 \,\Big|\, \mathcal{G}_n\Big] \\
  &= \mathbb{E}\Big[(\varepsilon^{(i)})^2 \,\Big|\, \mathcal{G}_n\Big] - 2(\mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big])^2 + \mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big]^2 \\
  &= \mathbb{E}\Big[(\varepsilon^{(i)})^2 \,\Big|\, \mathcal{G}_n\Big] - \mathbb{E}\Big[\varepsilon^{(i)} \,\Big|\, \mathcal{G}_n\Big]^2 \\
  &= \sigma^2 - 0
  \end{aligned}$$

  ■

- Strict exogeneity implies $\mathbb{E}\big(\varepsilon^{(i)}\big) = 0$ for all $i = 1, \dots, n$.

  Proof. By the law of total expectation, $\mathbb{E}\big(\varepsilon^{(i)}\big) = \mathbb{E}\big(\mathbb{E}\big[\varepsilon^{(i)} \,|\, \sigma(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)})\big]\big)$. ■

3.4 Ordinary Least-Squares (OLS)

Least-squares fitting is the name given to the mathematical procedure for finding the best-fitting curve to a collection of data by way of minimising the sum of squared residuals of the examples from the curve's respective predictions.

Ordinary[9] **least-squares** is a type of **linear** least-squares fitting method that estimates the unknown parameters $\boldsymbol{\theta}$ in a linear regression model by using a dataset $\mathcal{T}$. We denote the sum of squared residuals by $J(\boldsymbol{\theta})$ and define it by any constant multiple of

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\epsilon^{(i)})^2 := \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right)^2,$$

and in matrix form:

$$J(\boldsymbol{\theta}) = \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon} = (\mathcal{X}\boldsymbol{\theta} - \boldsymbol{y})^\top (\mathcal{X}\boldsymbol{\theta} - \boldsymbol{y}).$$

Goal: We wish to find $\boldsymbol{\theta}$ that minimises our loss $J(\boldsymbol{\theta})$ over $\boldsymbol{\theta} \in \mathbb{R}^{p+1}$.

The approach followed in CS229 opts to minimise $J(\boldsymbol{\theta})$ by calculating its gradient $\nabla_{\boldsymbol{\theta}}$ w.r.t. $\boldsymbol{\theta}$ by using some "known facts" about the matrix derivative. The calculation of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ has been side-barred below, and I've collected the matrix derivative facts in **Example A.2.2**.

The calculation is done by finding the gradient with respect to $\boldsymbol{\theta}$ of the cost function $J(\boldsymbol{\theta})$, setting that gradient to zero to get what are known as the normal equations, and then finding stationary points. The relevant definitions for the following calculations can be found in **Chapter A**.

We begin by setting the constant multiple of $J(\boldsymbol{\theta})$ to $1/2$ which simplifies future calculations.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left(\frac{1}{2} (\mathcal{X}\boldsymbol{\theta} - \boldsymbol{y})^\top (\mathcal{X}\boldsymbol{\theta} - \boldsymbol{y}) \right) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \left(\boldsymbol{\theta}^\top \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta} - \boldsymbol{\theta}^\top \mathcal{X}^\top \boldsymbol{y} - \boldsymbol{y}^\top \mathcal{X}\boldsymbol{\theta} + \boldsymbol{y}^\top \boldsymbol{y} \right) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \mathrm{tr} \left(\boldsymbol{\theta}^\top \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta} - \boldsymbol{\theta}^\top \mathcal{X}^\top \boldsymbol{y} - \boldsymbol{y}^\top \mathcal{X}\boldsymbol{\theta} + \boldsymbol{y}^\top \boldsymbol{y} \right) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \left(\mathrm{tr}(\boldsymbol{\theta}^\top \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta}) - \mathrm{tr}(\boldsymbol{y}^\top \mathcal{X}\boldsymbol{\theta}) - \mathrm{tr}(\boldsymbol{y}^\top \mathcal{X}\boldsymbol{\theta}) + \mathrm{tr}(\boldsymbol{y}^\top \boldsymbol{y}) \right) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \mathrm{tr}(\boldsymbol{\theta}^\top \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \mathrm{tr}(\boldsymbol{y}^\top \mathcal{X}\boldsymbol{\theta}) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \mathrm{tr}(\boldsymbol{\theta}^\top \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta}) - \mathcal{X}^\top \boldsymbol{y} \\
&\overset{(5)}{=} \frac{1}{2} ((\mathcal{X}^\top \mathcal{X})^\top \boldsymbol{\theta} + \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta}) - \mathcal{X}^\top \boldsymbol{y} = \mathcal{X}^\top \mathcal{X}\boldsymbol{\theta} - \mathcal{X}^\top \boldsymbol{y}
\end{aligned}$$

In the third equality, note that $\mathrm{tr}(a) = a$ for $a \in \mathbb{R}$. We introduced the trace for convenience of computation using 5 by setting $A^\top = \boldsymbol{\theta}$, $B = \mathcal{X}^\top \mathcal{X}$, and $C = I$.

Instead, I decided to follow the fantastic **lecture notes** (lectures 2–4, and 7) from the Spring 2016 Convex and Conic Optimisation course by Amir Ali Ahmadi. These concise notes gave me a solid understanding of basic optimisation, and the least-squares example was an exercise for the reader so all errors herein are my own.

[9] There's some contention over what 'ordinary' actually means. The most sensible explanation I've found is from **this Stack Exchange comment** suggesting that it's there to contrast with the weight parameters in weighted least-squares regression — a technique that can be found in the following subsection. Namely, if the weights in WLS are all equal to 1, you get OLS.

3.4.1 Convex Optimisation

Theorem 3.4.1 Consider the minimisation problem

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|^2$$

where A is an $m\times n$ matrix, and $b\in\mathbb{R}^n$. Under the assumption that the columns of A are linearly independent,

$$x^* = (A^\top A)^{-1}A^\top b$$

is the unique global solution.

Proof. We can re-write the objective function in matrix form $f(x) = (Ax-b)^\top(Ax-b)$. According to **Corollary A.1.4**, calculating the gradient of f at x can be done by first finding the Fréchet derivative of f at x, and then transposing the resulting bounded linear map.

Definition 3.4.2 Let $U \overset{\text{open}}{\subseteq} \mathbb{R}^n$ with $x\in U$ and $f\colon U\to\mathbb{R}$. If there exists a bounded linear[10] map $L_x\colon \mathbb{R}^n\to\mathbb{R}$ that satisfies

$$\lim_{v\to\mathbf{0}\text{ in }\mathbb{R}^n} \frac{|f(x+v)-f(x)-L_xv|}{\|v\|} = 0$$

then f is said to be Fréchet differentiable at x. The linear map L_x is called the Fréchet derivative of f at x and is denoted by $\mathrm{d}f(x)$.

We first calculate the numerator of the difference quotient:

$$\begin{aligned}
f(x+v)-f(x) &= (A(x+v)-b)^\top(A(x+v)-b)-f(x)\\
&= (Ax-b+Av)^\top(Ax-b+Av)-f(x)\\
&= f(x)+(Ax-b)^\top(Av)+(Av)^\top(Ax-b)+(Av)^\top(Av)-f(x)\\
&= \underbrace{(Ax-b)^\top(Av)}_{\in\mathbb{R}}+(Av)^\top(Ax-b)+(Av)^\top(Av)\\
&= ((Ax-b)^\top(Av))^\top+(Av)^\top(Ax-b)+(Av)^\top(Av)\\
&= 2(Av)^\top(Ax-b)+(Av)^\top(Av)
\end{aligned}$$

Now we conjecture that the linear part of the remainder $L_xv = 2(Ax-b)^\top Av$ is our derivative candidate. The matrix representation of the map L_x is $2(Ax-b)^\top A$. It's clearly linear because $L_x(v+w) = L_xv + L_xw$, and $L_x(\alpha v) = \alpha L_xv$. Every linear map between normed vector spaces whose domain is finite dimensional is automatically bounded. Now we verify the limit in the definition of the Fréchet derivative. The difference quotient is given by

$$\begin{aligned}
\frac{|f(x+v)-f(x)-L_xv|}{\|v\|} = \frac{|(Av)^\top(Av)|}{\|v\|} &= \frac{\|Av\|^2}{\|v\|}\\
&\leqslant \frac{\|A\|_{\mathrm{op}}\|v\|^2}{\|v\|}\\
&= \|A\|_{\mathrm{op}}\|v\| \longrightarrow 0 \text{ as } v\to\mathbf{0}\text{ in }\mathbb{R}^n.
\end{aligned}$$

Therefore, $\mathrm{d}f(x) = L_x$ as linear maps, $Jf(x) = 2(Ax-b)^\top A$ is the coordinate representation of $\mathrm{d}f(x)$ with respect to the standard bases of $\mathbb{R}^n$ and $\mathbb{R}$, from which it follows that $\nabla f(x) = 2A^\top(Ax-b) = 2A^\top Ax - 2A^\top b$.

[10]We also denote this by $L_x \in \mathcal{L}(\mathbb{R}^n;\mathbb{R})$.

Relating this back to our regression problem, we call the system of equations

$$\mathcal{X}^\top \mathcal{X}\boldsymbol{\theta} = \mathcal{X}^\top \boldsymbol{y}$$

the **normal equations**.

Theorem 3.4.3 (First-Order Necessary Condition for Optimality) If x^* is an unconstrained[11] local minimum of a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$, then $\nabla(x^*) = 0$.

Note that f is differentiable. By the FONC, x^* being an unconstrained local minimum solution $\implies \nabla f(x^*) = 0$. Note that $\nabla f(x) = 0 \iff A^\top Ax = A^\top b$. We wish to invert $A^\top A$ to get a solution x^*. Since A is assumed to have linearly independent columns, it follows that $A^\top A$ is invertible.

Proof. A matrix A is invertible iff its nullspace is trivial $\{0\}$. Observe that

$$\begin{aligned} (A^\top A)x = 0 &\implies x^\top (A^\top A)x = 0 \\ &\iff (Ax)^\top (Ax) = 0 \\ &\iff \|Ax\|^2 = 0 \\ &\iff Ax = \mathbf{0} \end{aligned}$$

i.e. if x is in the nullspace of $A^\top A$, then $Ax = \mathbf{0}$. If we can demonstrate further that $x = 0$, then the invertibility of $A^\top A$ follows.

Notice that

$$Ax = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^n a_{1j}x_j \\ \sum_{j=1}^n a_{2j}x_j \\ \vdots \\ \sum_{j=1}^n a_{mj}x_j \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \cdots + \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n$$

i.e. Ax is a linear combination of the columns of A with weight x_j corresponding to column j. By the linear independence of the columns, this forces $x_j = 0$ for every j from 1 to n i.e $x = \mathbf{0}$. In summary, we've shown that $x \in \text{nullspace}(A^\top A) \implies x = \mathbf{0}$ which is equivalent to $A^\top A$ being invertible. †

Thus, an unconstrained optimal solution to our optimisation problem is

$$x^* = (A^\top A)^{-1} A^\top b.$$

Now we appeal to the second-order sufficient condition for optimality.

Theorem 3.4.4 (Second-Order Sufficient Condition) Suppose that $f\colon \mathbb{R}^n$ is twice continuously differentiable, that there exists a point x^* such that $\nabla f(x^*) = 0$, and that $\nabla^2 f(x^*) \succ 0$. Then x^* is a strict local minimum of f.

Let's verify the final assumption of the theorem by calculating the Hessian matrix of f at $x^* = (A^\top A)^{-1} A^\top b$. According to **Remarks A.1.6**, we can calculate the Fréchet derivative of $\mathrm{d}f(x)$ i.e. the second-order Fréchet derivative of f, and then its coordinate representation with respect to the standard basis of $\mathbb{R}^n$ is the Hessian of f at x. As before, we calculate the difference.

$$\begin{aligned} \mathrm{d}f(x+v)h - \mathrm{d}f(x)h &= 2(A(x+v) - b)^\top Ah - 2(Ax - b)^\top Ah \\ &= 2(Ax - b)^\top Ah + 2(Av)^\top Ah - 2(Ax - b)^\top Ah \\ &= 2(Av)^\top Ah \\ &= 2v^\top A^\top Ah \end{aligned}$$

[11] Unconstrained means we minimise over all $x \in \mathbb{R}^n$.

Let $(L_x v)h = L_x(v,h) = 2v^\top A^\top Ah$. It's bilinear and automatically bounded. We don't need to verify the defining limit because $\mathrm{d}f(x+v)h - \mathrm{d}f(x)h - L_x(v,h) = 0$. Thus, the Hessian of f is

$$\nabla^2 f(x) = 2A^\top A.$$

We've already shown that $x^\top A^\top Ax = \|Av\| \geqslant 0$ and that $Ax = 0 \iff x = 0$, so $A^\top A \succ 0$. By the second-order sufficient condition for strict convexity

$$\nabla^2 f(x) \succ 0,\ \forall x \in \Omega \implies f \text{ is strictly convex on } \Omega$$

we conclude that our f is strictly convex on $\mathbb{R}^n$. Finally, we remark that the conditions of the following theorem are satisfied for $\Omega = \mathbb{R}^n$ so x^* is the global unique solution on $\mathbb{R}^n$.

Theorem 3.4.5 Consider an optimisation problem $\min f(x)$ subject to $x \in \Omega$, where $f\colon \mathbb{R}^n \to \mathbb{R}$ is strictly convex on Ω, and Ω is a convex set. Then the optimal solution (assuming it exists) must be unique.

■

The assumption that the columns of $\mathcal{X}$ are linearly independent means that there are no redundant features.

In summary, if the columns of $\mathcal{X}$ are linearly independent, then $J(\boldsymbol{\theta})$ is strictly convex on $\mathbb{R}^{p+1}$, and *if an optimal solution exists*, then it's the unique global minimum

$$\boldsymbol{\theta}^* = (\mathcal{X}^\top\mathcal{X})^{-1}\mathcal{X}^\top\boldsymbol{y}$$

that solves the normal equations $\mathcal{X}^\top\mathcal{X}\boldsymbol{\theta} = \mathcal{X}^\top\boldsymbol{y}$.

This conclusion came with the proviso that an optimal solution exists. We haven't (yet) shown this is the case.

3.4.2 Is there always a solution to the normal equations?

Let's look more closely at the normal equations.

$$\mathcal{X}^\top\mathcal{X}\boldsymbol{\theta} = \mathcal{X}^\top\boldsymbol{y}$$

When does a solution exist? On the left, we have $\mathcal{X}^\top\mathcal{X}$ acting on $\boldsymbol{\theta}$, and outputting $\mathcal{X}^\top\boldsymbol{y}$. We can view this as the classical linear algebra problem of solving $Ax = b$ for x. Equivalently[12], solving this system of equations means writing b as a linear combination of the columns of A. Thus, a solution to our normal equations exists iff $\mathcal{X}^\top\boldsymbol{y}$ is in the column space of $\mathcal{X}^\top\mathcal{X}$. This is indeed the case because:

Lemma 3.4.6 The column spaces of $\mathcal{X}^\top\mathcal{X}$ and $\mathcal{X}^\top$ are equal.

Proof. One inclusion is very simple. Let $b \in \operatorname{colspace}(\mathcal{X}^\top\mathcal{X})$ i.e. there exists some $v \in \mathbb{R}^n$ s.t. $\mathcal{X}^\top\mathcal{X}v = b$. Now let $w = \mathcal{X}v \in \mathbb{R}^n$, and so we've shown that there exists some $w \in \mathbb{R}^n$ s.t. $\mathcal{X}^\top w = b$ i.e. $b \in \operatorname{colspace}(\mathcal{X}^\top)$. Thus, $\operatorname{colspace}(\mathcal{X}^\top\mathcal{X}) \subseteq \operatorname{colspace}(\mathcal{X}^\top)$.

[12] As evidenced by our earlier calculation that

$$Ax = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \cdots + \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n.$$

For the reverse inclusion, let $b \in \operatorname{colspace}(\mathcal{X}^\top)$. We wish to show that $b \in \operatorname{colspace}(\mathcal{X}^\top\mathcal{X})$.

■

Thus, there always exists a solution to the normal equations. In the case that $\mathcal{X}$ is of full-rank, there is a unique global solution to the optimisation problem. Otherwise, $\mathcal{X}$ is not of full-rank i.e. $\operatorname{rank}(\mathcal{X}) < \min(n, p+1)$. Note that n is typically much larger than $p+1$ so $\operatorname{rank}(\mathcal{X}) < p+1$. By the rank-nullity theorem,

$$\operatorname{rank}(\mathcal{X}) + \dim \operatorname{nullspace}(\mathcal{X}) = p+1$$

so the dimension of the nullspace of $\mathcal{X}$ is strictly greater than 0 i.e. the nullspace of $\mathcal{X}$ is non-trivial. Let $\boldsymbol{\theta}^*$ be any optimal solution of the normal equations. Then we may take any non-zero vector v in the nullspace of $\mathcal{X}$ and observe that $\boldsymbol{\theta}^* + v$ also satisfies the normal equations

$$\mathcal{X}^\top\mathcal{X}(\boldsymbol{\theta}^* + v) = \mathcal{X}^\top\mathcal{X}\boldsymbol{\theta}^* + \mathcal{X}^\top(\mathcal{X}v) = \mathcal{X}^\top\boldsymbol{y} + \mathcal{X}^\top 0 = \mathcal{X}^\top\boldsymbol{y}.$$

Thus, there are infinitely many solutions to the normal equations.

Which solution does one typically choose out of the infinitely many? I have no idea right now so I'm leaving a blank space for it — feels important.

There are other methods to approximate $\widehat{\boldsymbol{\theta}}$. The following two algorithms are *iterative*:

3.5 Iterative Methods

3.5.1 Gradient Descent

Gradient descent is an iterative searching algorithm that searches for an optimal solution (minimiser) $\widehat{\boldsymbol{\theta}}$ that minimises J. It begins by initialising some $\boldsymbol{\theta}$ and repeatedly performs the *simultaneous* update

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \qquad \text{for } j = 0, \ldots, p.$$

Remarks 3.5.1

- The symbol $:=$ here is for the assignment operator when programming.
- Each iteration of this algorithm moves every θ_j in the direction of steepest decrease of J. This is because the partial derivative ∂_{θ_j} is defined as the direction of greatest increase of a function in the direction of θ_j.
- The constant α is called the **learning rate**[13] and determines how "large" the step is in each iteration.

More explicitly,

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{1}{2} \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right)^2 \\ &= \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) \\ &= \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}\end{aligned}$$

Thus, the update rule looks at every example at each iteration and is called the **batch gradient descent update rule**:

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \qquad \text{for } j = 0, \ldots, p \qquad \text{simultaneously.}$$

The batch gradient descent algorithm is as follows:

Algorithm 1 Batch gradient descent

```
1: procedure BATCH GRADIENT DESCENT
2:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$                                  ▷ Initialise a guess for $\boldsymbol{\theta}$
3:     while convergence criterion do
4:         $\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{n} (h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}) x_j^{(i)}$      ▷ for every $j$
5:     end while
6:     return $\boldsymbol{\theta}_0$
7: end procedure
```

The magnitude of the update of θ_j is proportional to the error term $h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}$. If the algorithm encounters an example for which the prediction nearly matches the actual value of $y^{(i)}$, then $(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}) \approx 0$ and it finds little need to change the parameters.

[13] If all the features are normalised to $[-1, 1]$, then $\alpha = 0.01$ is a sensible value to start with.

In general, gradient descent can be susceptible to local minima. This is because the algorithm makes locally optimal choices at each step/iteration. However, the way this particular OLS optimisation problem was set up guarantees convergence to a unique global minimiser (assuming α isn't too large). J has a unique global minimum.

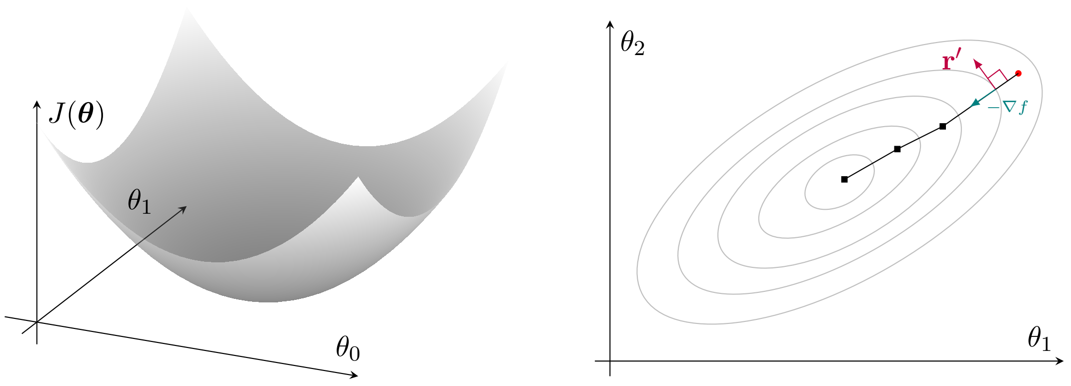


Figure 3.1: A paraboloid-shaped surface (left) with height $J(\boldsymbol{\theta})$ parameterised by $\boldsymbol{\theta} = \left[\theta_0, \theta_1\right]^\top$ and its level sets/contours (right). The contours for constant values of $J(\boldsymbol{\theta})$ are ellipses.

The direction of steepest ascent (of ∇f) is always orthogonal to the level set $f(\theta_1, \theta_2) = k$ of a surface i.e. for a parameterisation $\mathbf{r}$ of the level set, $\langle \nabla f, \mathbf{r}' \rangle = 0$. In the diagram, we follow $-\nabla f$ when updating our parameters. If α is too small, the algorithm will require far too many iterations. If the value of $J(\boldsymbol{\theta})$ increases then there's a very strong chance that α is too large and the minimiser has been overshot. A disadvantage of batch gradient descent is that a single iteration can become very slow if the number n of training examples is very large.

3.5.2 Stochastic Gradient Descent

Instead of scanning through all examples in a single iteration, a less computationally expensive option is to repeatedly loop over the training set $i = 1, \ldots, n$, at each step using only the current example $(x^{(i)}, y^{(i)})$ to update the parameters.

The update rule for a single example $(x^{(i)}, y^{(i)})$ is called the Widrof-Hoff learning rule:

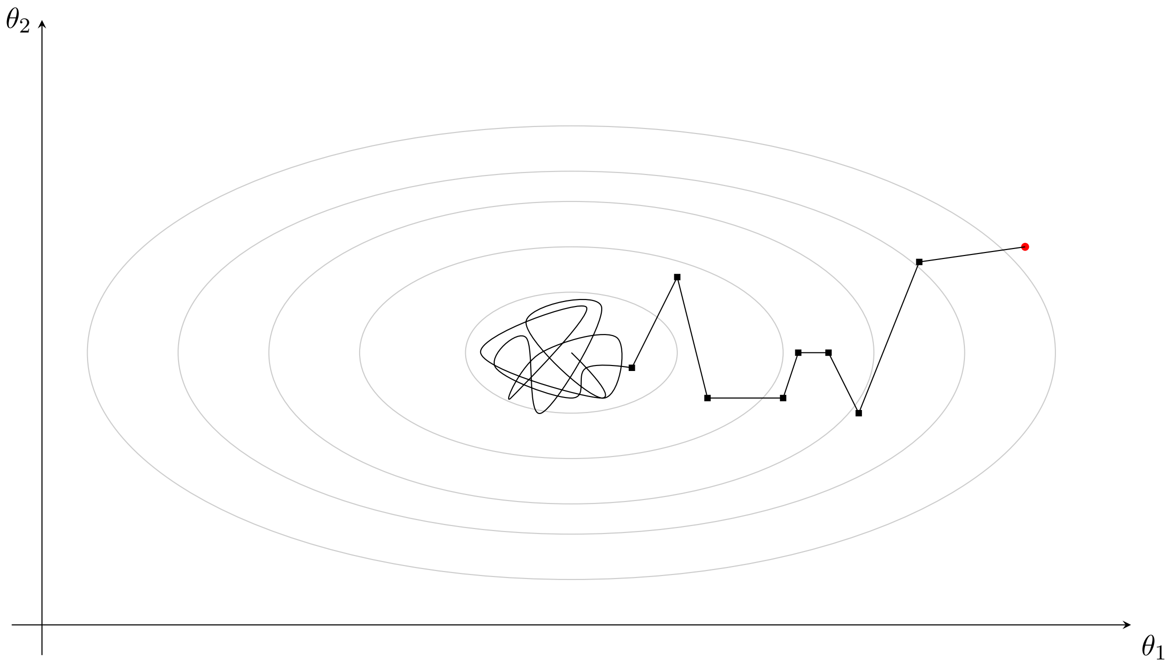
$$\theta_j := \theta_j - \alpha \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

The stochastic gradient descent algorithm is as follows:

Algorithm 2 Stochastic Gradient Descent

```
1: procedure STOCHASTIC GRADIENT DESCENT
2:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$                          ▷ Initialise a guess for $\boldsymbol{\theta}$
3:     for $i \leftarrow 1, n$ do
4:         $\theta_j \leftarrow \theta_j - \alpha \left( h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$          ▷ for every $j$
5:     end for
6:     return $\boldsymbol{\theta}_0$
7: end procedure
```

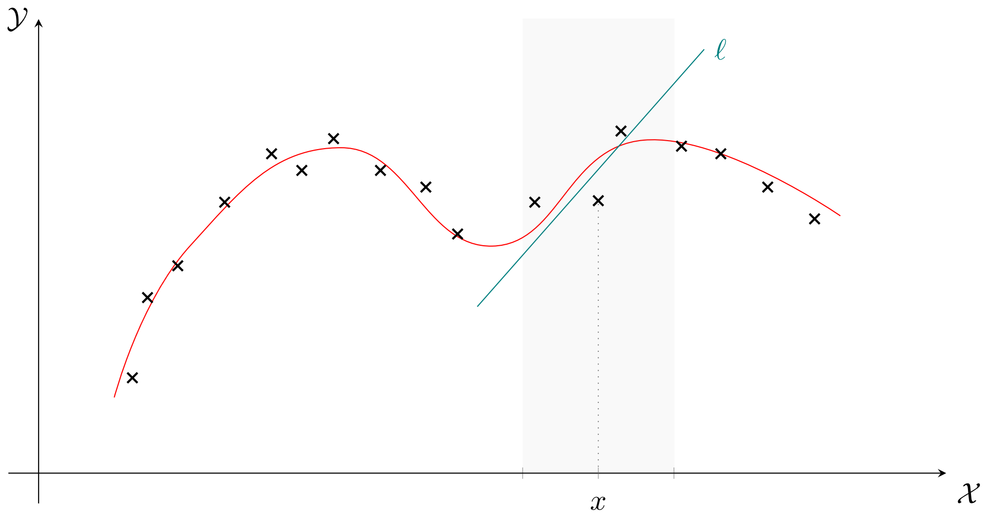
Since the algorithm uses the derivative for one example in each iteration, the parameters $\boldsymbol{\theta}$ are updated and improved slightly but may not be in the most direct direction downhill for J.



To this end, stochastic gradient descent takes a noisy route but on average gets $\boldsymbol{\theta}$ closer to a minimum. Convergence is not guaranteed for this algorithm and $\boldsymbol{\theta}$ will often keep oscillating around a minimum of $J(\boldsymbol{\theta})$. Decreasing the learning rate α in the later stages of the algorithm could yield good results.

3.6 Locally Weighted Linear Regression

Suppose that we have a set of data plotted below.



It's pretty clear what the shape of the data is. How does one fit a curve to the data that looks like the red curve? It's not so straightforward to find features (e.g. $\sqrt{x}$, $\log(x)$, $x^{2/3}$) to fit the data. Luckily, it's possible to sidestep this difficulty using a method called **locally weighted linear regression**:

The goal of ordinary least-squares linear regression was to find optimal parameters $\widehat{\boldsymbol{\theta}}$ that minimise a cost function $J(\boldsymbol{\theta})$ (which quantifies how our hypothesis $h_{\boldsymbol{\theta}}$ predictions differ from the given data) and then output a prediction model $h_{\widehat{\boldsymbol{\theta}}} = \widehat{\boldsymbol{\theta}}^\top x$.

Locally weighted linear regression looks at a small neighbourhood around a query point x, fits a straight line ℓ to the values in this neighbourhood and uses ℓ to make a prediction. To fit said straight line to the data around such a point x, one can use a *modified cost function* $J_{\text{loc}}(\boldsymbol{\theta})$ which places greater emphasis on points close by to x e.g.

$$J_{\text{loc}}(\boldsymbol{\theta}) := \sum_{i=1}^{n} \omega^{(i)} \left(\boldsymbol{\theta}^\top x^{(i)} - y^{(i)}\right)^2.$$

$\omega^{(i)}$ is known as a **weighting function** and for an example $(x^{(i)}, y^{(i)})$ exhibits the following:

- if $|x^{(i)} - x|$ is small, then $\omega^{(i)} \approx 1$
- else if $|x^{(i)} - x|$ is large, then $\omega^{(i)} \approx 0$

Thus, $\omega^{(i)}$ tells us how much attention must be paid to the values of $(x^{(i)}, y^{(i)})$ when fitting the straight line.

A common choice of $\omega^{(i)}$ is $\omega^{(i)} := \exp\left(-\dfrac{(x^{(i)} - x)^2}{2\blacksquare^2}\right)$.

What is the quantity 🟧 in the weighting function?

We need some way to describe how quickly the importance (or weight) of a training example falls off from the query point x. We do so by incorporating a quantity 🟧 $= \tau$ called the **bandwidth**. To illustrate, we graph $\omega^{(i)}$ against training examples:

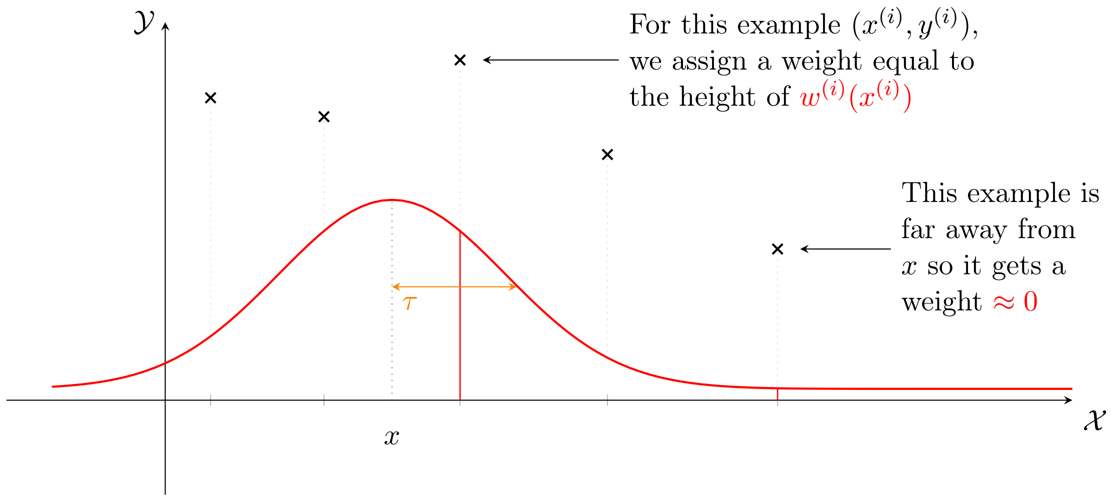

$\mathcal{Y}$
For this example $(x^{(i)}, y^{(i)})$, we assign a weight equal to the height of $w^{(i)}(x^{(i)})$
This example is far away from x so it gets a weight ≈ 0
τ
x
$\mathcal{X}$

Figure 3.2: A visualisation of the weighting function (in red) for a neighbourhood of the query point x.

Parametric and Non-Parametric Algorithms

Unweighted linear regression is an example of a **parametric algorithm** because it has a fixed, finite number of parameters (the θ_i's) which are fit to the data and once $\boldsymbol{\theta}$ is fit and stored, the training data is no longer needed to make future predictions.

On the other hand, making predictions with locally weighted linear regression requires us to keep the entire training set around. It's an example of a *non-parametric* algorithm because the amount of data/parameters that must be kept in order to represent the hypothesis grows linearly with the size of the training data.

Non-parametric performs badly if one has a particularly large training set but a benefit is that the data can be fit quite well without having to manually fiddle with features.

3.7 Normality Assumptions (Ordinary Linear Regression)

A probabilistic assumption reveals that the ordinary least-squares cost function $J(\boldsymbol{\theta})$ is the one we should be minimising because it naturally arises through the method of maximum likelihood estimation. We add this assumption to the linear regression model:

5. Normality of errors:
   The distribution of $\boldsymbol{\varepsilon} = (\varepsilon^{(1)}, \dots, \varepsilon^{(n)})$ conditional on $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}$ is jointly normal $\mathcal{N}(\mathbf{0}, \sigma^2 I_n)$.

With the inclusion of joint normality, a linear regression model is termed a ***Gaussian* linear regression model**.

We always assume that our random variables map into nice enough spaces (e.g. Borel spaces) s.t. there exists a system of regular conditional probabilities of $\boldsymbol{\varepsilon} \in \mathrm{Meas}_{\mathcal{F}, \mathcal{B}_{\mathbb{R}^n}}(\Omega\,; \mathbb{R}^n)$ given $\mathsf{X} := (\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}) \in \mathrm{Meas}_{\mathcal{F}, \mathcal{B}_{(\mathbb{R}^{(p+1)})^n}}(\Omega\,; (\mathbb{R}^{(p+1)})^n)$ i.e. a map $\kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}} \colon \mathcal{B}_{\mathbb{R}^n} \times (\mathbb{R}^{(p+1)})^n \to [0,1]$ s.t.

1. For every $\boldsymbol{x} = (x^{(1)}, \dots, x^{(n)}) \in (\mathbb{R}^{(p+1)})^n$, $\kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}(\cdot, \boldsymbol{x})$ is a probability measure on the codomain $(\mathbb{R}^n, \mathcal{B}_{\mathbb{R}^n})$ of $\boldsymbol{\varepsilon}$.
2. For each $B \in \mathcal{B}_{\mathbb{R}^n}$, the mapping $\boldsymbol{x} \longmapsto \kappa^X_Y(B, \boldsymbol{x})$ is:
   - an $\mathcal{B}_{(\mathbb{R}^{(p+1)})^n}$-measurable function satisfying for $\mathbb{P}$-a.e. $\omega \in \Omega$:
   $$\mathbb{P}\big[\boldsymbol{\varepsilon}^{-1}(B) \,\big|\, \sigma(\mathsf{X})\big](\omega) = \kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}(B, \mathsf{X}(\omega)),$$
   - and is $\mathbb{P}_{\mathsf{X}}$-integrable, satisfying for all $B \in \mathcal{B}_{\mathbb{R}^n}$ and $D \in \mathcal{B}_{(\mathbb{R}^{(p+1)})^n}$:
   $$\mathbb{P}\big(\boldsymbol{\varepsilon}^{-1}(B) \cap \mathsf{X}^{-1}(D)\big) = \int_D \kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}(B, \boldsymbol{x})\, \mathrm{d}\mathbb{P}_{\mathsf{X}}(\boldsymbol{x}).$$

The conditional normality means that $\kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}$ satisfies for every $B \in \mathcal{B}_{\mathbb{R}^n}$:

$$\kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}(B, \boldsymbol{x}) = \gamma_{n,\sigma^2}(B) \quad \text{for } \mathbb{P}_{\mathsf{X}}\text{-a.e. } \boldsymbol{x} \in (\mathbb{R}^{(p+1)})^n,$$

where φ_{n,σ^2} is a multivariate normal probability distribution. In other words, the conditional law of $\boldsymbol{\varepsilon}$ given X has a multivariate Gaussian density with mean $\mathbf{0}$ and covariance matrix $\sigma^2 I_n$. Also, since there's no dependence of φ_{n,σ^2} on $\boldsymbol{x}$, our disintegration formula becomes

$$\mathbb{P}\big(\boldsymbol{\varepsilon}^{-1}(B) \cap \mathsf{X}^{-1}(D)\big) = \int_D \gamma_{n,\sigma^2}(B)\, \mathrm{d}\mathbb{P}_{\mathsf{X}}(\boldsymbol{x}) = \gamma_{n,\sigma^2}(B)\mathbb{P}_{\mathsf{X}}(D)$$

and this factorisation demonstrates that $\boldsymbol{\varepsilon}$ and X are independent.

3.7.1 Consequences of joint conditional normality of our errors

The assumption of joint conditional normality of our errors implies several statements:

Corollary 3.7.1 Each $\varepsilon^{(i)}$ is conditionally normally distributed given $\mathbf{X}^{(i)}$ with the same mean 0 and variance σ^2.

Proof. The idea is to push the conditional law of $\boldsymbol{\varepsilon}$ given X forward via the natural coordinate projection map π_i. Indeed, for each $\boldsymbol{x} \in (\mathbb{R}^{(p+1)})^n$:

$$\kappa^{\varepsilon^{(i)}}_{\mathsf{X}}(\cdot\,, \boldsymbol{x}) := (\pi_i)_\sharp \kappa^{\boldsymbol{\varepsilon}}_{\mathsf{X}}(\cdot\,, \boldsymbol{x}) = (\pi_i)_\sharp \gamma_{n,\sigma^2}(\cdot) = \gamma_{1,\sigma^2}(\cdot)$$

because the marginal distribution of a multivariate normal is univariate normal with variance the $(i,i)^{\text{th}}$ entry in the covariance matrix. ■

Corollary 3.7.2 The $\varepsilon^{(i)}$ are conditionally independent given X.

Proof. This follows from the previous corollary by noting that for every x the joint conditional law of $\boldsymbol{\varepsilon}$ given $\boldsymbol{X}$ factorises into the following product measure of every marginal conditional distribution of $\varepsilon^{(i)}$ given X i.e.

$$\kappa_{\mathsf{X}}^{\boldsymbol{\varepsilon}}(\cdot\,,\boldsymbol{x}) = \gamma_{n,\sigma^2} = \bigotimes_{i=1}^{n} \gamma_{1,\sigma^2}(\cdot) = \bigotimes_{i=1}^{n} \kappa_{\mathsf{X}}^{\varepsilon^{(i)}}(\cdot\,,\boldsymbol{x}).$$

■

Corollary 3.7.3 There are no conditional cross-variance terms.

Proof. I imagine this comes from the covariance matrix entries outside of the main-diagonal being 0. ■

Corollary 3.7.4 The distribution of $\mathbf{Y} = \Phi(\mathsf{X})\boldsymbol{\theta} + \boldsymbol{\varepsilon}$ conditional on X is normal with mean $\Phi(\mathsf{X})\boldsymbol{\theta}$ and covariance matrix $\sigma^2 I_n$.

Proof. According to Section 18.3.2 from KEB103 [1, p. 205], we can in a sense "push-forward" the regular conditional probability of $\boldsymbol{\varepsilon}$ given X by a jointly-measurable map in order to obtain the conditional law of $\mathbf{Y}$.

Definition 3.7.5 Let $(E_X, \mathcal{E}_X)$ and $(E_Y, \mathcal{E}_Y)$ be measurable spaces. A **Markov (or transition) kernel from E_X to E_Y** is a map $\kappa\colon \mathcal{E}_Y \times E_X \to \mathbb{R}$ s.t.

- for every $x \in E_X$, $\kappa(\cdot, x)$ is a probability measure on $(E_Y, \mathcal{E}_Y)$, and
- for every $B \in \mathcal{E}_Y$, $\kappa(B, \cdot)$ is $\mathcal{E}_X$-measurable.

Theorem 3.7.6 Given a jointly measurable map $Z\colon E_Y \times E_X \to E_Z$, the function $\kappa^Z\colon \mathcal{E}_Z \times E_X \to [0,1]$ defined for $B \in \mathcal{E}_Z$ and $x \in E_X$ by

$$\kappa^Z(B, x) := \kappa(\{y \in E_Y : Z(y, x) \in B\}, x)$$

is a Markov kernel from E_X to E_Z.

It suffices to check that the map which takes $\boldsymbol{\varepsilon}$ and X to $\mathbf{Y}$ is jointly-measurable. Indeed, let[14]

$$Z\colon \underbrace{\mathbb{R}^n}_{\boldsymbol{\varepsilon}} \times \underbrace{(\mathbb{R}^{(p+1)})^n}_{\mathsf{X}} \to \underbrace{\mathbb{R}^n}_{\mathbf{Y}}$$

be the affine transformation

$$y = Z(\boldsymbol{\epsilon}, \boldsymbol{x}) = \Phi(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{\epsilon}.$$

This is clearly jointly measurable. Thus, the conditional law of $\mathbf{Y}$ given X is the regular conditional probability $\kappa_{\mathsf{X}}^{\mathbf{Y}}\colon \mathcal{B}_{\mathbb{R}^n} \times (\mathbb{R}^{(p+1)})^n \to [0,1]$ defined for any $B \in \mathcal{B}_{\mathbb{R}^n}$ and $\boldsymbol{x} \in (\mathbb{R}^{(p+1)})^n$ by

[14]The underbraces denote which random elements these inputs and outputs correspond to.

$$\begin{aligned}
\kappa_{\mathsf{X}}^{\mathbf{Y}}(B,\boldsymbol{x}) &:= \kappa_{\mathsf{X}}^{\boldsymbol{\varepsilon}}((Z\circ\iota_{\boldsymbol{x}})^{-1}(B),\boldsymbol{x}) \\
&= \kappa_{\mathsf{X}}^{\boldsymbol{\varepsilon}}(\{\boldsymbol{\epsilon}\in\mathbb{R}^n\colon (Z\circ\iota_{\boldsymbol{x}})(\boldsymbol{\epsilon})\in B\},\boldsymbol{x}) \\
&= \kappa_{\mathsf{X}}^{\boldsymbol{\varepsilon}}(\{\boldsymbol{\epsilon}\in\mathbb{R}^n\colon Z(\boldsymbol{\epsilon},\boldsymbol{x})\in B\},\boldsymbol{x}) \\
&= \kappa_{\mathsf{X}}^{\boldsymbol{\varepsilon}}(\{\boldsymbol{\epsilon}\in\mathbb{R}^n\colon \Phi(\boldsymbol{x})\boldsymbol{\theta}+\boldsymbol{\epsilon}\in B\},\boldsymbol{x}) \\
&= \gamma_{n,\sigma^2}(\{\boldsymbol{\epsilon}\in\mathbb{R}^n\colon \Phi(\boldsymbol{x})\boldsymbol{\theta}+\boldsymbol{\epsilon}\in B\}) \\
&= \int_{\{\boldsymbol{\epsilon}\in\mathbb{R}^n\colon \Phi(\boldsymbol{x})\boldsymbol{\theta}+\boldsymbol{\epsilon}\in B\}} \varphi_{n,\sigma^2}(\boldsymbol{\epsilon})\,\mathrm{d}\lambda_{\mathbb{R}^n}(\boldsymbol{\epsilon}) \\
&= \int_{h^{-1}(B)} \varphi_{n,\sigma^2}(\boldsymbol{\epsilon})\,\mathrm{d}\lambda_{\mathbb{R}^n}(\boldsymbol{\epsilon}) \qquad \text{where } \boldsymbol{u}=h(\boldsymbol{\epsilon}):=\Phi(\boldsymbol{x})\boldsymbol{\theta}+\boldsymbol{\epsilon} \\
&= \int_B \varphi_{n,\sigma^2}(\boldsymbol{u}-\Phi(\boldsymbol{x})\boldsymbol{\theta})|\det(J_{h^{-1}})(\boldsymbol{u})|\,\mathrm{d}\lambda_{\mathbb{R}^n}(\boldsymbol{u}) \\
&= \int_B \varphi_{n,\sigma^2}(\boldsymbol{u}-\Phi(\boldsymbol{x})\boldsymbol{\theta})\,\mathrm{d}\lambda_{\mathbb{R}^n}(\boldsymbol{u}) \\
&= \mathcal{N}(\Phi(\boldsymbol{x})\boldsymbol{\theta},\sigma^2 I_n)(B)
\end{aligned}$$

The penultimate line's Jacobian determinant of $h^{-1}(\boldsymbol{u})=\boldsymbol{u}-\Phi(\boldsymbol{x})\boldsymbol{\theta}$ is simply 1 because it's a translation. ■

The final piece of the puzzle follows.

Corollary 3.7.7 The $Y^{(i)}$ are conditionally independent given X, and each $Y^{(i)}$ is conditionally normally distributed given X with mean $\Phi(x^{(i)})\boldsymbol{\theta}$ and variance σ^2

Proof. The proof follows in an entirely analogous fashion to the proof of the independence of the $\varepsilon^{(i)}$ conditional on X. This observation is crucial for the following section. ■

By some leap of faith, I believe the conditional law of $Y^{(i)}$ given $\mathbf{X}^{(i)}$ is the same as that of $Y^{(i)}$ given X. If this is true, then one has the disintegration formula for any $B\in\mathcal{B}_{\mathbb{R}}$ and $D\in\mathcal{B}_{\mathbb{R}^{(p+1)}}$:

$$\begin{aligned}
\mathbb{P}\Big((Y^{(i)})^{-1}(B)\cap(\mathbf{X}^{(i)})^{-1}(D)\Big) &= \int_D \kappa_{\mathbf{X}^{(i)}}^{Y^{(i)}}(B,x)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x) \\
&= \int_D \mathcal{N}(\Phi(x^{(i)})\boldsymbol{\theta},\sigma^2)(B)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x) \\
&= \int_D\int_B \varphi_{1,\Phi(x^{(i)})\boldsymbol{\theta},\sigma^2}(y)\,\mathrm{d}\lambda(y)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x).
\end{aligned}$$

3.7.2 Maximum Likelihood Estimation

We begin with the assumption that our data $\mathcal{T}$ has already been observed from a random sample

$$(\mathbf{X}^{(1)},Y^{(1)}),\dots,(\mathbf{X}^{(n)},Y^{(n)}) \overset{\text{i.i.d.}}{\sim} \mathbb{P}_{\boldsymbol{\theta}}$$

whose distribution is a particular parametric distribution $\mathbb{P}_{\boldsymbol{\theta}}$ from a family $\{\mathbb{P}_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\Theta}$, where Θ is some indexing set. For the purposes of what follows, we assume that this family of probability measures is dominated by a σ-finite measure ν. In the discrete case, ν will be the counting measure; and in the absolutely continuous case, ν is the Lebesgue measure. This domination assumption allows us to speak of the[15] density of $\mathbb{P}_{\boldsymbol{\theta}}$, and therefore speak of the central object of this section.

[15]Technically this is defined up to ν-null sets but it's not a massive abuse of language to say 'the' with the tacit assumption that we're referring to a representative of an equivalence class of functions that differ ν-a.e.!

The method of maximum likelihood estimation is a search for optimal parameters $\boldsymbol{\theta}' \in \Theta$ for which it was most likely that our data had been sampled from $\mathbb{P}_{\boldsymbol{\theta}'}$. We do this by working with a **surrogate** quantity called the joint likelihood L of our data. We view it as a function of $\boldsymbol{\theta}$ in order to carry out the aforementioned optimisation.

Definition 3.7.8 The **joint likelihood function for $\boldsymbol{\theta}$ given our data $\mathcal{T}$** is the density of the joint distribution of our sample (with respect to a dominating measure ν) evaluated at the observed value $\mathcal{T}$.

$$L = \frac{\mathrm{d}\mathbb{P}_{(\mathbf{X}^{(1)},Y^{(1)}),\ldots,(\mathbf{X}^{(n)},Y^{(n)})}}{\mathrm{d}\nu}\bigg|_{\mathcal{T}}.$$

In our case, our random sample is i.i.d. and made up of (absolutely) continuous random vectors (with respect to λ), so we may take $\nu = \otimes^n \lambda$ and obtain[16] the following expression:

$$L(\boldsymbol{\theta}) = \frac{\mathrm{d}\mathbb{P}_{(\mathbf{X}^{(1)},Y^{(1)}),\ldots,(\mathbf{X}^{(n)},Y^{(n)})}}{\mathrm{d}\otimes^n \lambda}\Bigg|_{\mathcal{T}} \overset{\text{i.i.d.}}{=} \left(\prod_{1}^{n} \frac{\mathrm{d}\mathbb{P}_{\boldsymbol{\theta}}}{\mathrm{d}\lambda}\right)\Bigg|_{\mathcal{T}} = \prod_{i=1}^{n} f_{\theta}((x^{(i)}, y^{(i)}))$$

I think it's important to emphasise that the likelihood is not a product of probabilities. Instead, it's a Radon-Nikodym derivative of the (joint) density of the random variable of interest with respect to a (product) dominating measure. In our case with a random sample, the independence implies that our derivative becomes a product of densities. In the case that the original random variable of interest is discrete, the density is indeed the probability mass function (and hence the probability itself). This means that the likelihood is the probability but simply viewed as a function of the parameters. In the general case, there is no such correspondence.

From the disintegration formula earlier, if we assume that $\mathbf{X}^{(i)}$ has density $f_{\mathbf{X}^{(i)},\boldsymbol{\theta}}$, then we have the following equality for any $B \in \mathcal{B}_{\mathbb{R}}$ and $D \in \mathcal{B}_{\mathbb{R}^{(p+1)}}$:

$$\begin{aligned}
\mathbb{P}\Big((Y^{(i)})^{-1}(B) \cap (\mathbf{X}^{(i)})^{-1}(D)\Big) &= \int_D \kappa_{\mathbf{X}^{(i)}}^{Y^{(i)}}(B, x)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x) \\
&= \int_D \mathcal{N}(\Phi(x^{(i)})\boldsymbol{\theta}, \sigma^2)(B)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x) \\
&= \int_D \int_B \varphi_{1,\Phi(x^{(i)})\boldsymbol{\theta},\sigma^2}(y)\,\mathrm{d}\lambda(y)\,\mathrm{d}\mathbb{P}_{\mathbf{X}^{(i)}}(x) \\
&= \int_D \int_B \varphi_{1,\Phi(x^{(i)})\boldsymbol{\theta},\sigma^2}(y)\,\mathrm{d}\lambda(y) f_{\mathbf{X}^{(i)},\boldsymbol{\theta}}\,\mathrm{d}\lambda(x) \\
&= \int_D \int_B \varphi_{1,\Phi(x^{(i)})\boldsymbol{\theta},\sigma^2}(y) f_{\mathbf{X}^{(i)},\boldsymbol{\theta}}(x)\,\mathrm{d}\lambda(y)\,\mathrm{d}\lambda(x)
\end{aligned}$$

which demonstrates the **joint density** f_θ of $(\mathbf{X}^{(i)}, Y^{(i)})$ factorises into the product of the conditional density of $\kappa_{\mathbf{X}^{(i)}}^{Y^{(i)}}$ (which is the normal) and the marginal density of $\mathbf{X}^{(i)}$.

[16] I'm leaving out some details that I expect are true but not quite sure e.g. if $\mu_1, \ldots, \mu_n$ are measures, and $\nu_1, \ldots, \nu_n$ is a collection of σ-finite measures s.t for every i we have $\mu_i \ll \nu_i$, then

$$\frac{\mathrm{d}(\mu_1 \otimes \ldots \otimes \mu_n)}{\mathrm{d}(\nu_1 \otimes \ldots \otimes \nu_n)} = \prod_{i=1}^{n} \frac{\mathrm{d}\mu_i}{\mathrm{d}\nu_i}.$$

Therefore, the likelihood takes the form:

$$\begin{aligned} L(\boldsymbol{\theta}) = \; \dots \; &= \prod_{i=1}^{n} f_\theta((x^{(i)}, y^{(i)})) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \Phi(x^{(i)})\boldsymbol{\theta})^2}{2\sigma^2}\right) f_{\mathbf{X}^{(i)},\boldsymbol{\theta}}(x^{(i)}) \end{aligned}$$

This factorisation is useful for two reasons:

1. We've only made assumptions about the conditional distribution of our errors from which it follows that each $Y^{(i)}$ is conditionally normally distributed given $\mathbf{X}^{(i)}$.

2. If we assume the marginal density is independent of $\boldsymbol{\theta}$, then optimising $L(\boldsymbol{\theta})$ is an equivalent problem to optimising the **conditional likelihood** L_{cond} defined by

$$L_{\text{cond}}(\boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \Phi(x^{(i)})\boldsymbol{\theta})^2}{2\sigma^2}\right).$$

Densities can often take small values, and the conditional likelihood above can be a product of many such small numbers. Multiplying many small numbers together in an algorithm could underflow[17] the computer. We'd prefer to work with a sum instead of a product so need a suitable transformation T. By suitable, we'd like that if $\mathcal{L}(\boldsymbol{\theta})$ is maximised at $\widehat{\boldsymbol{\theta}}$, then $T(\mathcal{L}(\boldsymbol{\theta}))$ is also maximised at $\widehat{\boldsymbol{\theta}}$. A candidate for T which ticks all the boxes is the natural logarithm $T = \log$. It's monotonically increasing and transforms products into sums.

Thus, we typically prefer to use the conditional log likelihood $\ell(\boldsymbol{\theta}) := \log(L_{\text{cond}}(\boldsymbol{\theta}))$:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \Phi(x^{(i)})\boldsymbol{\theta})^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \Phi(x^{(i)})\boldsymbol{\theta})^2}{2\sigma^2}\right)\right) \\ &= n\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2\sigma^2} \underbrace{\sum_{i=1}^{n} (y^{(i)} - \Phi(x^{(i)})\boldsymbol{\theta})^2}_{=J(\boldsymbol{\theta})} \end{aligned}$$

This means that the problem of maximising $L_{\text{cond}}(\boldsymbol{\theta})$:

- is equivalent to maximising $\ell(\boldsymbol{\theta})$
- which is equivalent to minimising $J(\boldsymbol{\theta})$

which was precisely the ordinary least-squares linear regression problem.

In fact, the overall framework defined by the two points below

1. Make a set of probabilistic assumptions about the conditional response.

2. Using the method of maximum likelihood estimation, derive a new type of supervised learning algorithm.

[17] When the result of a calculation is more precise than a computer can represent in memory.

is something we'll re-use for fitting parameters for different types of problems. In particular, when the target variable $y \in \mathcal{Y}$ is now either 0 or 1 i.e. a classification problem.

When building learning algorithms, often we make assumptions about the world that we just know aren't 100% true because it leads to algorithms that are computationally efficient.

Andrew Ng

CHAPTER 4

Classification

Let $\mathcal{X} = \mathbb{R}^p$, and $\mathcal{Y} = \{c_1, \ldots, c_K\}$ for a finite integer $K \in \mathbb{N}$.

4.1 Linear Classifiers

A **classifier** is a function $h \colon \mathcal{X} \to \{c_1, \ldots, c_K\}$, where K is the number of classes under consideration. Such functions partition the input feature space $\mathcal{X}$ into K cells[1] whose boundaries are called decision boundaries. If these decision boundaries are linear, we call h a **linear classifier**.[2] A linear classifier makes decisions by thresholding across linear decision boundaries so it can be thought of as the composition of first performing a linear transformation $T \colon \mathcal{X} \to \mathbb{R}^K$ of an input feature vector $x \in \mathcal{X}$ (which is a way of scoring an input feature), followed by some (activation) thresholding function f to make the decision, i.e. $h = f \circ T$.

4.1.1 AGGREGATION/SCORING T

Since $T \colon \mathcal{X} \to \mathbb{R}^K$ may be represented by a $K \times p$ matrix W, we can denote h by $h_W(x) = f(Wx)$. The entries $w_{i,j}$ (for $i = 1, \ldots, K$ and $j = 1, \ldots, n$) of W are called weights. Each weight places an "importance" on the corresponding feature, and serves as a parameter that can be fine-tuned by an algorithm.

Focusing only on the linear transformation for now, if we think of each row W_i of W as the prototypical object of the class c_i, the i^{th} component of $T(x)$ is given by

$$\sum_{j=1}^{n} w_{i,j} x_j = W_i^\top x = \langle W_i, x \rangle.$$

Each inner product $\langle W_i, x \rangle$ quantifies how "similar" our example's feature representation x is to W_i. Thus, $T(x)$ is a vector of "scores" representing how well x fits into each and every class.

Some books add a bias term $b \in \mathbb{R}^K$ to the linear transformation $T(x) = Wx$ to make an affine transformation $A(x) = Wx + b$. Written explicitly:

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K,1} & w_{K,2} & \cdots & w_{K,n} \end{bmatrix} \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = \begin{bmatrix} \\ \\ \vdots \\ \\ \end{bmatrix}$$

Thankfully this can be viewed as a special case of $T(x) = Wx$ by embedding the input feature vectors into $\mathbb{R}^{p+1}$ with $x_0 = 1$ and

[1] I use this term in the mathematical sense. A **partition** of a set $\mathcal{X}$ is a family of non-empty subsets of $\mathcal{X}$ such that each $x \in \mathcal{X}$ belongs to a unique subset. The subsets in a partition are called **cells**. We say that $\mathcal{X}$ is partitioned into $B_1, \ldots, B_K$ if the B_i are such that for $i \neq j$, $B_i \cap B_j = \emptyset$ and

$$\mathcal{X} = \bigsqcup_{i=1}^{K} B_i.$$

[2] There are different names for linear classifiers; In biology and neuroscience, linear classifiers are often called perceptrons or single units of a neural net. In statistics and machine learning, they are called logistic regression and support vector machines (SVMs) respectively.

appending the bias vector to the left of W:

$$\begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,0} & w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{K,0} & w_{K,1} & w_{K,2} & \dots & w_{K,n} \end{bmatrix} \begin{bmatrix} 1 \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} = \begin{bmatrix} \text{🟥} \\ \text{🟦} \\ \vdots \\ \text{🟩} \end{bmatrix}$$

where $b_i = w_{i,0}$ for $i = 1, \dots, K$.

4.1.2 Threshold f

We can take $f = \operatorname{argmax}_{i \in \{1,\dots,K\}} T_i(x)$ as the subscript i of the class label c_i for our prediction. This follows the previous logic that such an i will correspond to the class c_i for which x is most similar to W_i.

There are certainly other choices of f to define our linear classifier h. Probably something to do with how the decision boundaries are formed; one class vs the rest, pairwise (class vs class) comparisons etc.

The simplest case of classification is between $K = 2$ classes — **binary classification**. Our hypothesis $h\colon \mathcal{X} \to \{c_1, c_2\} := \{-1, 1\}$ is[3] called a **linear binary classifier** and takes the form

$$h(x) = f(T(x)) = f(Wx) = f\left(\begin{bmatrix} W_1^\top x \\ W_2^\top x \end{bmatrix}\right).$$

The class labels inspire the following nomenclature:

- If $y^{(i)} = +1$, then $(x^{(i)}, y^{(i)})$ is called a **positive example**.
- If $y^{(i)} = -1$, then $(x^{(i)}, y^{(i)})$ is called a **negative example**.

Let f return the class label c_i for which $W_i^\top x$ is maximised. Then our hypothesis is

$$h(x) = f(T(x)) = \begin{cases} +1, & \text{if } W_1^\top x \geqslant W_2^\top x \\ -1, & \text{if } W_1^\top x < W_2^\top x. \end{cases}$$

The comparison between $W_1^\top x$ and $W_2^\top x$ can be written in terms of a single vector

$$W_1^\top x - W_2^\top x = (\underbrace{W_1 - W_2}_{=\boldsymbol{\theta}})^\top x.$$

This use of $\boldsymbol{\theta}$ in the linear (or discriminant) function $T(x) = \boldsymbol{\theta} x$ matches with other literature and is consistent with the more general $T(x) = Wx$ at the start of this chapter.

4.2 The Single-Layer Perceptron

Let $K = 2$. Our hypothesis $h\colon \mathcal{X} \to \{-1, 1\}$ takes the form

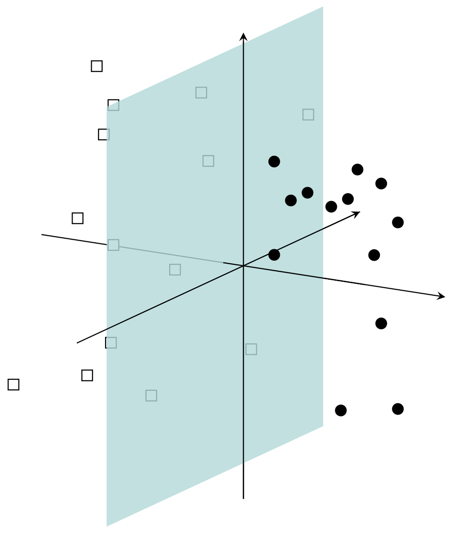
$$h(x) = f(T(x)) = f(\boldsymbol{\theta}^\top x) =: h_{\boldsymbol{\theta}}(x),$$

where we adopt the convention that $x_0 = 1$, and the bias term θ_0 has been incorporated into $\boldsymbol{\theta} \in \mathbb{R}^{p+1}$. Our training set is a collection of examples

$$\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n \subseteq \mathbb{R}^p \times \{-1, +1\}.$$

[3] Up to re-labelling — we could equally have chosen $\{0, 1\}$ or any other two-element set for our labels $\mathcal{Y}$. This choice of course influences the form that f takes.

We can plot the input feature vectors $x^{(i)}$ as points in $\mathbb{R}^p$, and represent each point with a shape corresponding to its label $y^{(i)}$. Geometrically, we wish to learn a linear decision boundary that separates $\mathcal{X}$ into two subsets; one containing only inputs $x^{(i)}$ with a positive label, and its complement containing only inputs $x^{(i)}$ with a negative label. This linear decision boundary is a **hyperplane**[4] of $\mathbb{R}^p$ i.e. a subset of $\mathbb{R}^p$ (with codimension 1) whose points satisfy the equation $\langle \boldsymbol{\theta}, x \rangle := \boldsymbol{\theta}^\top x = 0$ for $\mathbf{0} \neq \boldsymbol{\theta} \in \mathbb{R}^p$.



● Positive examples $y^{(i)} = +1$

□ Negative examples $y^{(i)} = -1$

Figure 4.1: An example of a perfectly linearly separable dataset in $\mathbb{R}^3$ with a separating hyperplane decision boundary (teal).

The above key for denoting positive and negative example feature vectors by shapes will be kept the same for the remainder of this section.

This means that our decision hyperplane (which passes through the origin) is the collection of points orthogonal to $\boldsymbol{\theta}$. This divides $\mathbb{R}^p$ into two **half-spaces**.

- If an example $(x^{(i)}, y^{(i)})$ satisfies $\boldsymbol{\theta}^\top x^{(i)} \geqslant 0$, then $x^{(i)}$ is in the half-space in which our choice of $\boldsymbol{\theta}$ directs into. This is because the inner product $\left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle = \boldsymbol{\theta}^\top x^{(i)}$ is a measure of similarity between the two vectors. Assign the label $+1$ to such an $x^{(i)}$.
- Otherwise, $\left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle < 0$ and we shall assign it the value -1.

These considerations define our hypothesis as

$$\begin{aligned} h_{\boldsymbol{\theta}}(x) &= \begin{cases} +1, & \text{if } \boldsymbol{\theta}^\top x \geqslant 0 \\ -1, & \text{if } \boldsymbol{\theta}^\top x < 0 \end{cases} \\ &=: \operatorname{sgn}(\boldsymbol{\theta}^\top x). \end{aligned}$$

This is the link between linear binary classifiers and decision hyperplanes.

4.2.1 Setup

Given the last section, we know that finding an optimal decision boundary that minimises misclassifications over our training set is tantamount to finding a vector of optimal parameters $\boldsymbol{\theta}$. Searching for an appropriate $\boldsymbol{\theta}$ will follow the same general structure as the previous chapters.

[4] In general, a hyperplane has exactly two unit normals $\pm\hat{\mathbf{n}}$ and an **affine hyperplane** of $\mathbb{R}^p$ with normal vector $+\hat{\mathbf{n}}$ and origin translation $\mathbf{b} \in \mathbb{R}^n$ is defined to be the set of all $\mathbf{x} \in \mathbb{R}^p$ s.t. $\hat{\mathbf{n}} \cdot (\mathbf{x} - \mathbf{b}) = 0$.

Algorithm 3 A single epoch of the training step in the single-layer perceptron algorithm

```
1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$                                   ▷ Initialise $\boldsymbol{\theta}$
2: for $i = 1, \dots, n$ do
3:     Update $\boldsymbol{\theta} \leftarrow \widetilde{\boldsymbol{\theta}}$
4: end for
5: return $\widetilde{\boldsymbol{\theta}}$
```

Consider a particular training example $(x^{(i)}, y^{(i)}) \in \mathcal{T}$.

- If $(x^{(i)}, y^{(i)})$ is correctly classified by $h_{\boldsymbol{\theta}}$, either:
  - $y^{(i)} = +1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = +1$, or
  - $y^{(i)} = -1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = -1$.

  i.e. $y^{(i)} h_{\boldsymbol{\theta}}(x^{(i)}) = +1$.
- If $(x^{(i)}, y^{(i)})$ is misclassified by $h_{\boldsymbol{\theta}}$, either:
  - $y^{(i)} = +1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = -1$, or
  - $y^{(i)} = -1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = +1$.

  i.e. $y^{(i)} h_{\boldsymbol{\theta}}(x^{(i)}) = -1$

It follows that our loss function is very straight-forward to define. For each example, we define the "hard" loss of our hypothesis $h_{\boldsymbol{\theta}}$ by

$$L_{\text{hard}}(h_{\boldsymbol{\theta}}(x^{(i)}), y^{(i)}) = \mathbb{1}\{h_{\boldsymbol{\theta}}(x^{(i)}) \neq y^{(i)}\}.$$

The associated empirical risk (which averages the hypothesis' loss over $\mathcal{T}$) is given by

$$R_{\text{emp.}}(h_{\boldsymbol{\theta}}) := \frac{1}{n} \sum_{i \in \mathcal{M}} L_{\text{hard}}(h_{\boldsymbol{\theta}}(x^{(i)}), y^{(i)}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h_{\boldsymbol{\theta}}(x^{(i)}) \neq y^{(i)}\}.$$

That final term is precisely the misclassification rate of our hypothesis with respect to L_{hard}. Naturally, we'd like to penalise misclassifications so everything checks out with the framework in Chapter 2 — the empirical risk is the quantity we wish to minimise by adjusting $\boldsymbol{\theta}$ in search of an optimal hypothesis $h_{\boldsymbol{\theta}}$ i.e. we want to find

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \mathbb{1}\{h_{\boldsymbol{\theta}}(x^{(i)}) \neq y^{(i)}\}.$$

Equivalently, finding such parameters $\boldsymbol{\theta}$ means that every training example is correctly classified i.e.

$$y^{(i)} h_{\boldsymbol{\theta}}(x^{(i)}) > 0.$$

How do we search for such parameters? The first thought is to spam gradient descent on line 3 in the vague general structure of **Algorithm 3**. The indicator function isn't differentiable in any neighbourhood of its discontinuity. Too bad. No gradient descent for us!

Let's conduct a little thought experiment to side-step this difficulty.

4.2.2 Surrogate perceptron problem

Suppose that we chose a differentiable function f instead of $\operatorname{sgn}(x)$ in our hypothesis, namely $f = \mathrm{id}_{\mathbb{R}}$. Then $h_{\boldsymbol{\theta}}(x^{(i)}) = \boldsymbol{\theta}^\top x^{(i)}$ would've been differentiable. We call this particular hypothesis the **unbounded linear unit**. Support further that instead of L_{hard}, we chose a "softer" loss

$$L_{\text{soft}}(h_{\boldsymbol{\theta}}(x^{(i)}), y^{(i)}) = \max(0, -y^{(i)}\boldsymbol{\theta}^\top x^{(i)}).$$

This is known as the **perceptron loss**.[5]

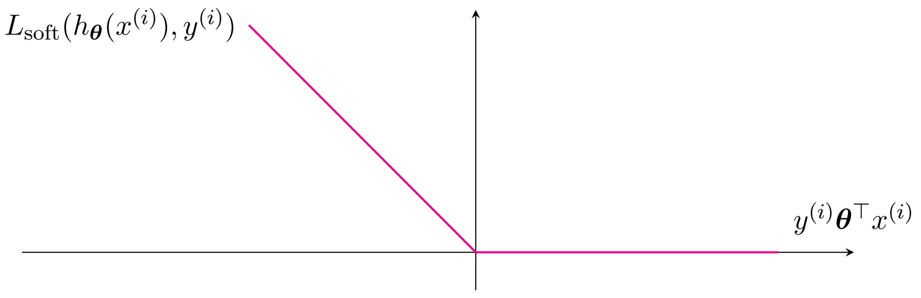


Figure 4.2: The perceptron loss.

We get a bit more information from this than L_{hard} because a larger value of $-y^{(i)}\boldsymbol{\theta}^\top x^{(i)}$ quantifies how off our prediction is (as opposed to the flat value of 1 for a misclassified example). Keep in mind that this loss function behaves the same as L_{hard} for correctly classified examples — they all get assigned a flat 0.

As before, the quantity we wish to minimise is the corresponding empirical risk function

$$^{\text{soft}}R_{\text{emp.}}(h_{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i=1}^{n} L_{\text{soft}}(h_{\boldsymbol{\theta}}(x^{(i)}), y^{(i)}).$$

Let $\mathcal{M} \subseteq \{1, \ldots, n\}$ denote the set of indices corresponding to the examples misclassified by $h_{\boldsymbol{\theta}}$

$$\mathcal{M} := \{i \colon y^{(i)}h_{\boldsymbol{\theta}}(x^{(i)}) < 0\}.$$

It turns out that our soft empirical risk is equal to

$$^{\text{soft}}R_{\text{emp.}}(h_{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i\in\mathcal{M}} -y^{(i)}\boldsymbol{\theta}^\top x^{(i)}.$$

This sum is certainly a quantity we can differentiate and so the method of gradient descent is a viable approach for finding an optimal $\boldsymbol{\theta}$. Let's give that quantity a name

$$J(\boldsymbol{\theta}) := \sum_{i\in\mathcal{M}} -y^{(i)}\boldsymbol{\theta}^\top x^{(i)}.$$

$J(\boldsymbol{\theta})$ is certainly positive. It's also convex:
Proof. We'll use two facts:

- $\max(0, -s)$ is convex.

Proof. $\max(x, y) = \dfrac{x+y}{2} + \dfrac{|x-y|}{2}$ and $|\cdot|$ is certainly convex (by the triangle inequality). ■

[5] A variant $\max(1-s, 0)$ is called the Hinge Loss.

- If $f(\cdot)$ is convex, then $f(A(\cdot)+b)$ is also convex i.e. composition of a function with an affine transformation preserves convexity.

Now we simply note that $\max(0, -y^{(i)}\boldsymbol{\theta}^\top x^{(i)})$ is the composition $(s \longmapsto \max(0,-s)) \circ (x \longmapsto y^{(i)}\boldsymbol{\theta}^\top x^{(i)})$. ■

Also note that $J(\boldsymbol{\theta})$ is convex in $\boldsymbol{\theta}$ with domain $\mathbb{R}^n$ and is bounded below by 0. The convexity implies that if there's a local minimiser, then it's **a**[6] global minimiser at which J attains its global minimum.
The algorithm outline for the gradient descent goes as follows:

Algorithm 4 Gradient descent on the soft empirical risk

```
1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$                                  ▷ Initialise $\boldsymbol{\theta}$
2: while loop condition do
3:     update $\boldsymbol{\theta} \leftarrow \widetilde{\boldsymbol{\theta}}$ via the gradient descent update rule
4:     update $\mathcal{M} \leftarrow \{i : -y^{(i)}(\widetilde{\boldsymbol{\theta}})^\top x^{(i)} > 0\}$
5: end while
6: return $\widetilde{\boldsymbol{\theta}}$
```

If the data are perfectly linearly separable into two classes, our loop condition can simply be (**while** $\mathcal{M} \neq \emptyset$) and the algorithm terminates when we find optimal parameters $\boldsymbol{\theta}^*$ such that all examples have been correctly classified by $h_{\boldsymbol{\theta}^*}$ i.e. the empirical risk is zero. Note that $\boldsymbol{\theta}^*$ is not necessarily unique. There may be many values of the parameters for which the empirical risk is zero. This is illustrated in the following figure:

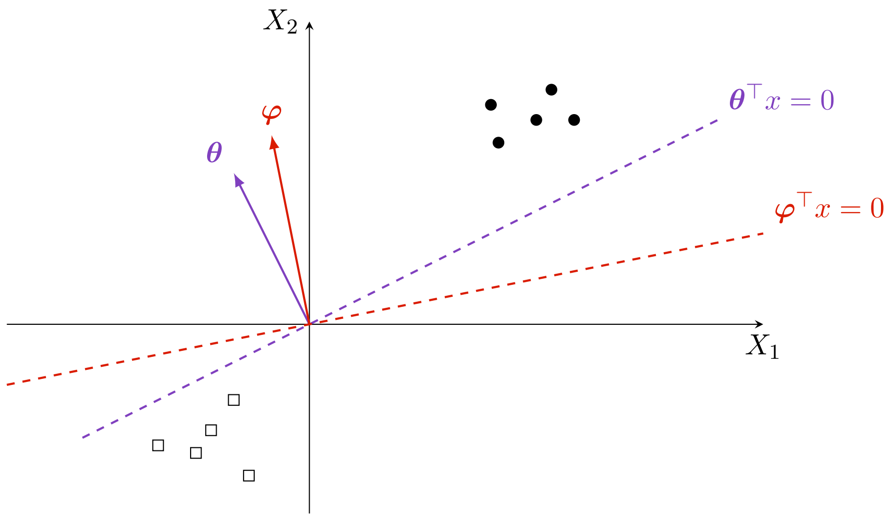


Figure 4.3: Even with perfectly linearly separable data, in $\mathbb{R}^2$ for illustrative convenience, there is no unique decision boundary for perfect classification.

If the data aren't perfectly linearly separable into two classes, a solution $h_{\boldsymbol{\theta}}$ can be still be learned with $R_{\text{emp.}}(h_{\boldsymbol{\theta}}) > 0$ but we'd have to implement a reasonable loop condition. End of thought experiment.

End of thought experiment. Let's return to the non-differentiable hypothesis with $\operatorname{sgn}(\cdot)$ having a jump discontinuity:

[6]Since J is not strictly convex (because max has flat parts), we can't conclude that such a global minimiser is unique.

4.2.3 Classical perceptron

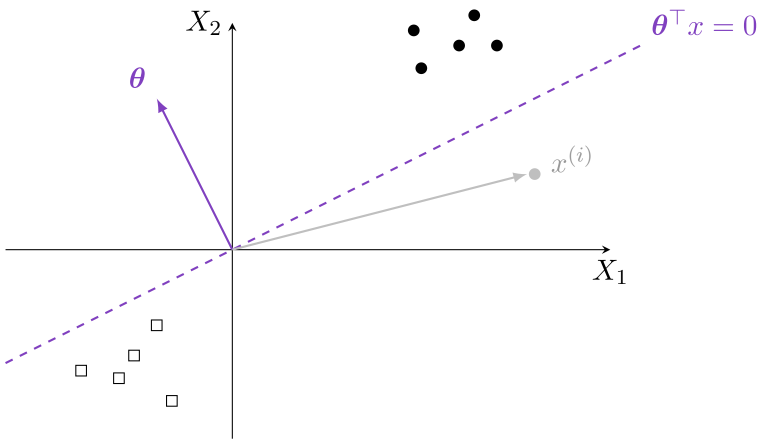
We shall make some geometric considerations to define an appropriate update rule for $\boldsymbol{\theta}$ and *indirectly* minimise the misclassification rate $R_{\text{emp.}}(h_{\boldsymbol{\theta}})$.

Suppose that we are working in $\mathbb{R}^2$ for visual convenience i.e. our feature vectors are of the form

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} \in \mathcal{X} = \mathbb{R}^2.$$

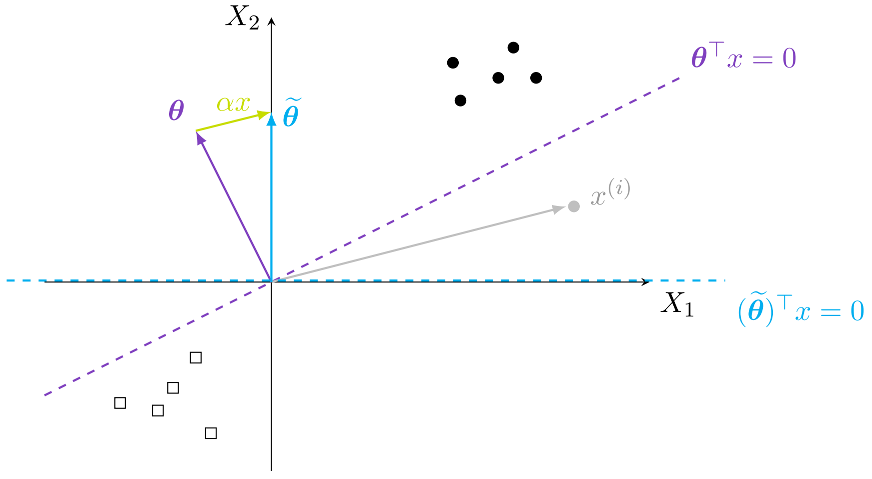
Suppose further that we have initialised $\boldsymbol{\theta}$ and the corresponding hyperplane ~~miraculously~~ correctly classifies all but one training example.

- Suppose that the currently misclassified example $(x^{(i)}, y^{(i)})$'s true label is positive. Highlight this by making it grey ● i.e. $y^{(i)} = +1$ but $h_{\boldsymbol{\theta}}(x^{(i)}) = -1$.



To compensate for this example having been misclassified, a reasonable update of the parameters $\boldsymbol{\theta} \longmapsto \widetilde{\boldsymbol{\theta}}$ would move (and in particular, for our hyperplane *passing through the origin*, rotate) the decision boundary in such a way that $x^{(i)}$ would point more in the direction of $\widetilde{\boldsymbol{\theta}}$ i.e. to increase the value from $\langle \boldsymbol{\theta}, x^{(i)} \rangle < 0$ to $\langle \widetilde{\boldsymbol{\theta}}, x^{(i)} \rangle \geqslant 0$. Since $x^{(i)}$ is of course fixed (as part of the data $\mathcal{T}$ from which we're learning), we could add some non-negative scalar multiple α of $x^{(i)}$ to $\boldsymbol{\theta}$ i.e. $\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} + \alpha x^{(i)} =: \widetilde{\boldsymbol{\theta}}$ because

$$\left\langle \widetilde{\boldsymbol{\theta}}, x^{(i)} \right\rangle := \left\langle \boldsymbol{\theta} + \alpha x^{(i)}, x^{(i)} \right\rangle = \left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle + \alpha \underbrace{\left\langle x^{(i)}, x^{(i)} \right\rangle}_{\geqslant 0} \geqslant \left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle.$$



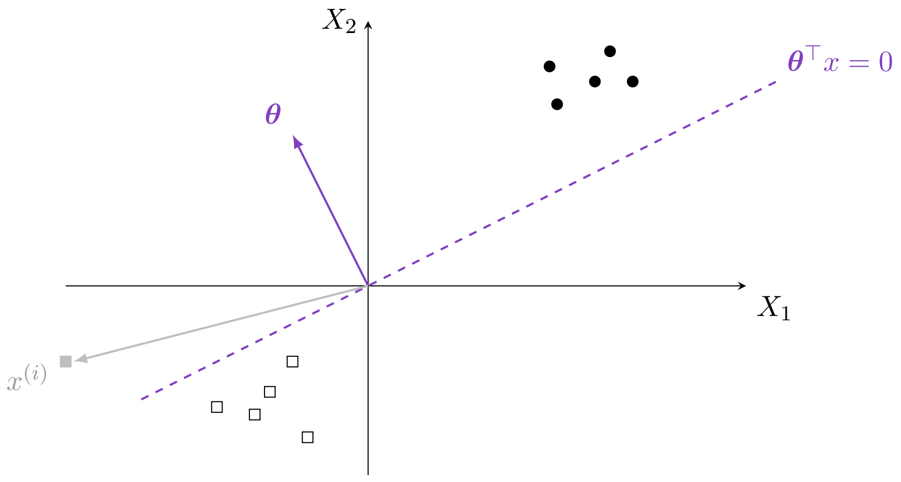
X_2
$\boldsymbol{\theta}$
αx
$\widetilde{\boldsymbol{\theta}}$
$\boldsymbol{\theta}^\top x = 0$
$x^{(i)}$
X_1
$(\widetilde{\boldsymbol{\theta}})^\top x = 0$

Therefore, our update of $\boldsymbol{\theta}$ can be written as

$$\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} + \alpha x^{(i)} =: \boldsymbol{\theta} + y^{(i)} \alpha x^{(i)}.$$

That inclusion of $y^{(i)} = +1$ is non-obvious but the next example will highlight why we've done so.

- Now suppose instead that the misclassification is of a negative example ■ i.e. $y^{(i)} = -1$ but our hypothesis has assigned it the label $h_{\boldsymbol{\theta}}(x^{(i)}) = +1$.

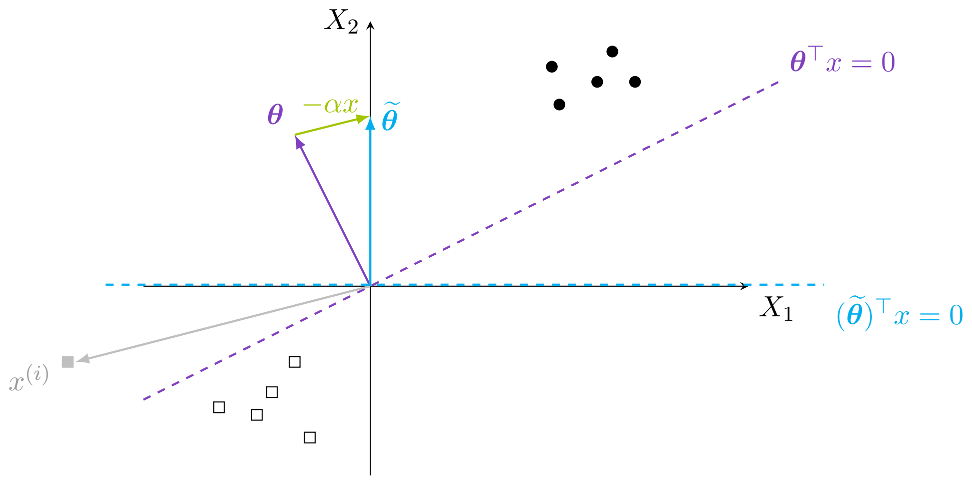


To compensate for this example having been misclassified, we want to achieve the opposite effect to before. Our decision boundary assigns our example a positive value $\langle \boldsymbol{\theta}, x^{(i)} \rangle \geqslant 0$ but we wish to update $\boldsymbol{\theta} \longmapsto \widetilde{\boldsymbol{\theta}}$ so that our example points in the opposite direction to $\widetilde{\boldsymbol{\theta}}$ i.e. $\langle \widetilde{\boldsymbol{\theta}}, x^{(i)} \rangle < 0$. We can do this by subtracting some non-negative scalar multiple α of $x^{(i)}$ to $\boldsymbol{\theta}$ i.e. $\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} - \alpha x^{(i)} =: \widetilde{\boldsymbol{\theta}}$ because

$$\left\langle \widetilde{\boldsymbol{\theta}}, x^{(i)} \right\rangle := \left\langle \boldsymbol{\theta} - \alpha x^{(i)}, x^{(i)} \right\rangle = \left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle - \alpha \underbrace{\left\langle x^{(i)}, x^{(i)} \right\rangle}_{\geqslant 0} \leqslant \left\langle \boldsymbol{\theta}, x^{(i)} \right\rangle.$$

Note that

$$\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} - \alpha x^{(i)} =: \boldsymbol{\theta} + y^{(i)} \alpha x^{(i)}.$$



X_2
$\boldsymbol{\theta}^\top x = 0$
$\boldsymbol{\theta}$
$-\alpha x$
$\widetilde{\boldsymbol{\theta}}$
X_1
$(\widetilde{\boldsymbol{\theta}})^\top x = 0$
$x^{(i)}$

To summarise, both misclassifications reduce to the update of parameters

$$\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} + y^{(i)} \alpha x^{(i)}.$$

We didn't mention it before but if the example is correctly classified, we of course do not update $\boldsymbol{\theta}$ in the current iteration. Our total update rule is

$$\begin{cases} \text{Do nothing} & \text{if } (x^{(i)}, y^{(i)}) \text{ is correctly classified,} \\ \boldsymbol{\theta} \longmapsto \boldsymbol{\theta} + y^{(i)} \alpha x^{(i)} & \text{if } (x^{(i)}, y^{(i)}) \text{ is misclassified.} \end{cases}$$

Equivalence to Rosenblatt's Perceptron (1962)

In the interest of historical preservation, the update rule derived above can be re-written in a way that serves as a special case of Rosenblatt's original 'error correction procedure' formulation introduced in [4, p. 110]. The actual update rule from [4, pp. 292–293] is written as the increment we update the weights by

$$\Delta \nu = a^* \cdot (\operatorname{sgn}(R^* - r^*)) \cdot \epsilon.$$

In our notation:

- An asterisk superscript e.g. a^* denotes the current example being used, and any subsequent objects computed from it.
  - The current input feature vector is $a^* := x^{(i)}$.
- The weights are $\nu := \boldsymbol{\theta}$.
- The *required response* is $R^* := y^{(i)}$.
- The *obtained response* is $r^* := h_{\boldsymbol{\theta}}(x^{(i)})$.
- The learning rate is $\epsilon := \alpha$.

Thus, the update rule is simply given by

$$\boldsymbol{\theta} \longmapsto \boldsymbol{\theta} + \alpha \operatorname{sgn}(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})) x^{(i)}.$$

How this is equivalent to our geometrically-derived update is simple. If our example has been correctly classified, Rosenblatt's rule also doesn't update the parameters for $\operatorname{sgn}(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})) = \operatorname{sgn}(0) = 0$. For the misclassified examples:

- If $y^{(i)} = +1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = -1$, then $\operatorname{sgn}(y^{(i)} - h_\theta(x^{(i)})) = \operatorname{sgn}(2) = 1 =: y^{(i)}$.
- Else, $y^{(i)} = -1$ and $h_{\boldsymbol{\theta}}(x^{(i)}) = +1$, and so $\operatorname{sgn}(y^{(i)} - h_\theta(x^{(i)})) = \operatorname{sgn}(-2) = -1 =: y^{(i)}$.

In both cases, $\operatorname{sgn}(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})) = y^{(i)}$ and so we recover our update rule.

The Algorithm and Comments

At last, we can state a naïve implementation of the **classical perceptron algorithm**.

Algorithm 5 A single epoch of the training step in the perceptron algorithm

```
1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$                                   ▷ Initialise $\boldsymbol{\theta}$
2: while $\mathcal{M} \neq \emptyset$ do
3:     for $i = 1, \dots, n$ do
4:         if $y^{(i)} = h_{\boldsymbol{\theta}}(x^{(i)})$ then
5:             pass
6:         else
7:             for $i = 1, \dots, p$ do
8:                 $\theta_j \leftarrow \theta_j + y^{(i)} \alpha x_j^{(i)}$
9:             end for
10:        end if
11:    end for
12: end while
13: return $\boldsymbol{\theta}$
```

The above algorithm assumes that the data is perfectly linearly separable i.e. once all examples have been correctly classified, the `while` loop terminates. However, for training data that isn't perfectly linearly separable, we have a few options:

- We can fine-tune α as we go along, throttling the size of the updates until a threshold has been crossed (at which point we decide further updates aren't meaningful).
- We can impose a hard cap on the number of iterations.
- We can change our hypothesis from the non-differentiable $\text{sgn}(\cdot)$ function to $h_{\boldsymbol{\theta}} = f \circ T$ for a differentiable f and follow the argument from earlier to minimise the corresponding "soft" loss function instead of the "hard" loss associated with $\text{sgn}(\cdot)$.

The classical perceptron learning algorithm isn't used much in practice; it doesn't have a probabilistic interpretation (as far as I can tell) but it's useful to have a geometric feel for what's happening with hyperplanes. It used to be pretty famous in the 50's and 60's when people thought it was a good model for the brain and how it works. However, a famous paper by Marvin Minsky demonstrated that the perceptron is limited because for some datasets, there does not exist a linear separating boundary.

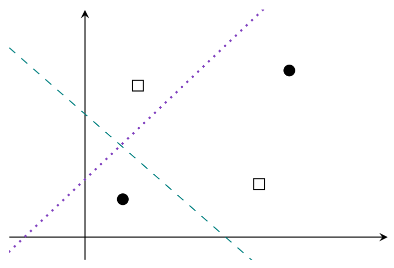
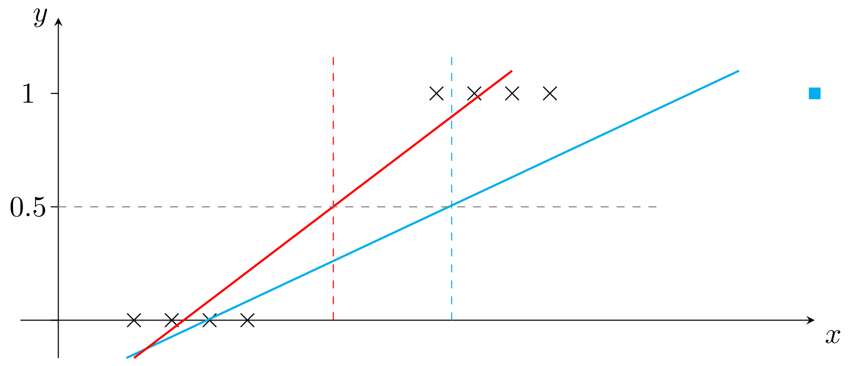


Figure 4.4: A dataset in $\mathbb{R}^2$ for which there does not exist a linear separating decision boundary. Two linear decision boundaries that fail to perfectly classify the dataset are depicted.

In the interest of inference, it would be great to have a model that quantifies how likely each example is to be a member of each class. In comes the modelling technique of logistic regression that uses a softer hypothesis to model the (conditional) probability of an unseen example's label (given its feature vector) — our hypothesis will be a so-called sigmoid function, and we shall see that the perceptron loss can be viewed as the limiting case of such a sigmoid.

4.3 Logistic Regression

Consider the following dataset:



The data is clearly separated into two classes. Suppose that a linear regression hypothesis $h_{\boldsymbol{\theta}}$ is fit to the data. This produces a line of best fit $y = h_{\boldsymbol{\theta}}(x)$. We can use this line to classify examples into two distinct classes. The separating boundary can be set to $h_{\boldsymbol{\theta}}(x) = 0.5$:

$$\text{label of } x = \begin{cases} 0, & h_{\boldsymbol{\theta}}(x) \geqslant 0.5 \\ 1, & h_{\boldsymbol{\theta}}(x) < 0.5 \end{cases}$$

Now say that we add a single example ■ to the dataset. With this extra example, which is clearly classified as positive and doesn't change the structure of the dataset very much, the new straight line fit $h_{\boldsymbol{\theta}'}$ is vastly different and thresholding at 0.5 gives a very different decision boundary. This suggests that the linear regression model doesn't generalise well to even similar training sets of this kind. **Model variance** refers to the amount by which the hypothesis $h_{\boldsymbol{\theta}}$ would change if we estimated it using a different training data set. Thus, linear regression has high variance for this type of binary classification task and it's probably best to find another method that's not so sensitive to perturbing the input $\mathcal{T}$ in order to classify datasets of this nature.

4.3.1 The Model

Without complete data on the population, there's often no perfect rule for classification. Instead, we resort to a stochastic approach that aims to model the conditional distribution of the response given the predictors. We do this, as described in **Section 1.2**:

- Assume that the data $\mathcal{T}$ has been generated by a true probability distribution $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ that lives in a family $\{\mathbb{P}_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\Theta}$ of distributions parameterised by some $\boldsymbol{\theta} \in \Theta$.
- Search through this parameter space Θ by iterating over the training data in order to find an optimal parameter $\boldsymbol{\theta}^*$ that makes $\mathbb{P}_{\boldsymbol{\theta}^*}$ a good approximation for $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$.

Probabilistic Assumptions

As always, we assume that the training examples are realisations of a collection of random vectors

$$(\mathbf{X}^{(i)}, Y^{(i)}) \overset{\text{i.i.d.}}{\sim} \mathbb{P}_{\mathcal{X}\times\mathcal{Y}}.$$

Since Y only assumes values in $\{0, 1\}$, it seems appropriate to model the response by a conditional Bernoulli distribution given $\mathbf{X}$. Since we always assume $\mathcal{X}$ to be a nice enough (Borel) space, there exists a system[7] of regular conditional probabilities $\{\mathbb{P}^{\boldsymbol{x}}(\cdot) := \kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})\}_{\boldsymbol{x}\in\mathcal{X}}$ on

[7]This is somewhat of an abuse of notation because we refer to a system of regular conditional probabilities on $\mathcal{F} = \sigma(\mathbf{X}, Y) = \sigma(\mathbf{X}) \vee \sigma(Y)$ generated by $\mathbf{X} \in \text{Meas}_{\mathcal{F},\mathcal{E}_{\mathcal{X}}}(\Omega\,;\mathcal{X})$ both by the transition kernel $\kappa_{\mathbf{X}}^{Y} : \sigma(\mathbf{X}, Y) \times \mathcal{X} \to \mathbb{R}$ and and the collection $\{\mathbb{P}^{\boldsymbol{x}}\}_{\boldsymbol{x}\in\mathcal{X}}$ where $\mathbb{P}^{\boldsymbol{x}}(\cdot)$ is the name we give every $\kappa_{\mathbf{X}}^{Y}(\cdot, \boldsymbol{x})$.

$\mathcal{F} = \sigma(\mathbf{X}, Y) = \sigma(\mathbf{X}) \vee \sigma(Y)$ given[8] the sub-σ-algebra $\mathcal{G} = \sigma(\mathbf{X})$. The values of each conditional probability measure $\kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})$ is completely determined[9] by its values on $\{Y = 0\}$ and $\{Y = 1\}$, and these values must sum to unity

$$\kappa_{\mathbf{X}}(\{Y = 0\}, \boldsymbol{x}) + \kappa_{\mathbf{X}}(\{Y = 1\}, \boldsymbol{x}).$$

The goal is to model the conditional expectation of the response with a suitable hypothesis h so we can make predictions beyond the training data. In the case of our conditionally distributed Bernoulli random variable Y given $\mathbf{X}$, our conditional expectation is the function

$$\boldsymbol{x} \longmapsto \int_\Omega Y(\omega')\, \mathrm{d}\kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})(\omega') =: \mathbb{E}[Y \,|\, \mathbf{X} = \boldsymbol{x}].$$

Since $Y^{-1}(\{0\}) \sqcup Y^{-1}(\{1\}) = \Omega$, our expression for $\mathbb{E}[Y \,|\, \mathbf{X} = \boldsymbol{x}]$ becomes

$$\begin{aligned}\mathbb{E}[Y \,|\, \mathbf{X} = \boldsymbol{x}] &= \int_{Y^{-1}(\{0\})} Y(\omega')\, \mathrm{d}\kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})(\omega') + \int_{Y^{-1}(\{1\})} Y(\omega')\, \mathrm{d}\kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})(\omega') \\ &= \int_{Y^{-1}(\{1\})} \mathrm{d}\kappa_{\mathbf{X}}(\cdot, \boldsymbol{x})(\omega') \\ &= \kappa_{\mathbf{X}}(Y^{-1}(\{1\}), \boldsymbol{x})\end{aligned}$$

which is the very formal way of writing the familiar expression

$$\text{“}\mathbb{E}[Y = 1 \,|\, \mathbf{X} = x] = \mathbb{P}_{Y|\mathbf{X}=x}(Y = 1 \,|\, \mathbf{X} = x).\text{”}$$

We first incorporate the predictors into the model as a linear combination $\boldsymbol{\theta}^\top \mathbf{X}$ like in linear regression. As before, $\boldsymbol{\theta}^\top \mathbf{X}$ can take any real value. Since Y assumes values in $\{0, 1\}$, and we're assuming that the conditional response is Bernoulli, we need to restrict the output of our hypothesis $[0, 1]$ in some meaningful way so as to interpret it as a probability. One such commonly used function is called the sigmoid/logistic function $g\colon \mathbb{R} \to [0, 1]$ defined by $z \mapsto (1 + e^{-z})^{-1}$. Thus, a candidate hypothesis to model the conditional expectation is $h_{\boldsymbol{\theta}}\colon \mathcal{X} \to [0, 1]$ defined by

$$h_{\boldsymbol{\theta}}(x) = g(\boldsymbol{\theta}^\top x) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top x}}.$$

There are plenty of other functions g that could've been chosen to massage our prediction within $[0, 1]$. The logistic function was chosen in particular because it arises as a natural example of a broader class of algorithms derived from a broader set of probabilistic principles.[10]

All things above considered, the conditional distribution is completely specified by the condition

$$1 = \kappa_{\mathbf{X}}(\{Y = 0\}, \boldsymbol{x}) + \underbrace{\kappa_{\mathbf{X}}(\{Y = 1\}, \boldsymbol{x})}_{=:\, h_{\boldsymbol{\theta}}(\boldsymbol{x})}$$

which can be re-written for $y \in \{0, 1\}$ as

$$\kappa_{\mathbf{X}}(\{Y = y\}, \boldsymbol{x}) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}))^y (1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})^{1-y}.$$

As usual, we don't have access to the actual conditional distribution of Y given $\mathbf{X}$ but we do have a training set $\mathcal{T}$ to analyse with the goal of learning “good” parameters $\boldsymbol{\theta}$ so that our model $h_{\boldsymbol{\theta}}$ best fits the data.

In exactly the same spirit as **Section 3.7.2**, we are interested in searching for parameters $\boldsymbol{\theta}$ that maximise the conditional likelihood given by

$$L_{\text{cond.}}(\boldsymbol{\theta}) = \prod_{i=1}^{n} (h_{\boldsymbol{\theta}}(x^{(i)}))^{y^{(i)}} (1 - h_{\boldsymbol{\theta}}(x^{(i)}))^{1-y^{(i)}}$$

[8]Also said to be ‘generated by $\mathbf{X}$.’

[9]By a standard extension argument, first determining the values of every such measure on the generators of $\sigma(\mathbf{X}, Y)$.

[10]These are called generalised linear models.

The conditional log-likelihood is simpler to maximise:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} \log(h_{\boldsymbol{\theta}}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(x^{(i)})) \right).$$

An example of an algorithm to find an optimal value $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ which maximises $\ell(\boldsymbol{\theta})$ is batch gradient ascent. The update rule for $j = 0, \dots, p$ simultaneously is:

$$\theta_j := \theta_j + \alpha \frac{\partial}{\partial \theta_j} \ell(\boldsymbol{\theta}).$$

We use gradient ascent instead of descent because the log function is strictly concave. We could, equivalently, have used gradient descent on $-\ell(\boldsymbol{\theta})$.

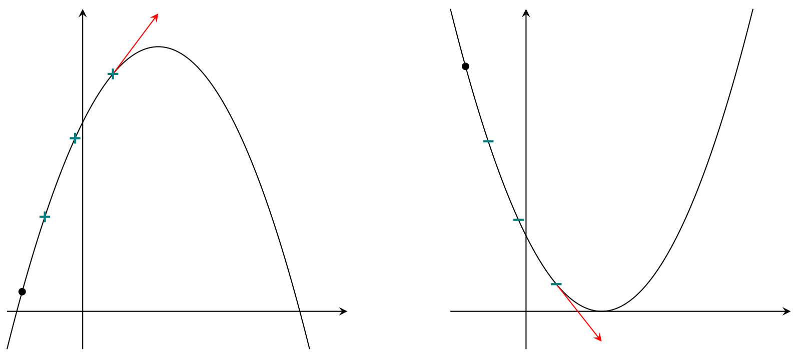


Figure 4.5: A visualisation of gradient ascent for ℓ-concave (left) and gradient descent for ℓ-convex (right).

Plugging the definition of $\ell(\boldsymbol{\theta})$ into the update rule requires some algebra. The derivative of the logistic function is:

$$g'(z) = (g(z))(1 - g(z)) \tag{dσ}$$

For a single example $(x^{(i)}, y^{(i)})$, the partial derivative of $h_{\boldsymbol{\theta}}(x^{(i)})$ w.r.t. θ_j is:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} h_{\boldsymbol{\theta}}(x^{(i)}) &= \frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^\top x^{(i)}) \\
&= g'(\boldsymbol{\theta}^\top x^{(i)}) \frac{\partial}{\partial \theta_j} \left(\boldsymbol{\theta}^\top x^{(i)} \right) \\
&= g'(\boldsymbol{\theta}^\top x^{(i)}) x_j^{(i)} \\
&\overset{(\mathrm{d}\sigma)}{=} g(\boldsymbol{\theta}^\top x^{(i)}) \left(1 - g(\boldsymbol{\theta}^\top x^{(i)}) \right) x_j^{(i)} \\
&= h_{\boldsymbol{\theta}}(x^{(i)}) \left(1 - h_{\boldsymbol{\theta}}(x^{(i)}) \right) x_j^{(i)}
\end{aligned}$$

Thus, the expression for the partial derivative of the log-likelihood is:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} \left(y^{(i)} \log(h_{\boldsymbol{\theta}}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(x^{(i)})) \right) \\
&= \sum_{i=1}^{n} \left(\frac{y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})}{h_{\boldsymbol{\theta}}(x^{(i)}) \left(1 - h_{\boldsymbol{\theta}}(x^{(i)}) \right)} \right) \frac{\partial}{\partial \theta_j} h_{\boldsymbol{\theta}}(x^{(i)}) \\
&= \sum_{i=1}^{n} (y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})) x_j^{(i)}
\end{aligned}$$

and the update rule is given by the following formula

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)}) \right) x_j^{(i)} \qquad \text{for } j = 0, \dots, p \text{ simultaneously.}$$

This update rule looks identical (in form) to that of linear regression but the hypothesis function $h_{\boldsymbol{\theta}}$ is of course different ($g(\boldsymbol{\theta}^\top x)$ instead of $\Phi(x^{(i)})\boldsymbol{\theta} = \boldsymbol{\theta}^\top x$ for linear regression). This is no coincidence and will be re-visited later when GLMs are covered.

Logistic regression is really a method that, given an example x, outputs a probability $h_{\boldsymbol{\theta}}(x)$. It's up to the user to choose some kind of a threshold (e.g. 0.5) for these probabilities in order to define a classifier.

At the end of the perceptron algorithm, we can think of $\boldsymbol{\theta}$ as some linear combination of the misclassified examples during the training phase. With logistic regression, $\boldsymbol{\theta}$ is some linear combination of all examples with the weights being how correct or wrong our hypothesis was during training.

There are other methods we can use to search for a "good" $\boldsymbol{\theta}$:

4.4 Newton's Method

Gradient descent maximises a function using knowledge of its derivative.

An alternative method that can be used for maximising a function is known as Newton's method. Newton's method is a root-finding algorithm that constructs a sequence of real numbers that converge towards a root by leveraging second order Taylor approximations about the iterates. When applied to the derivative f' of a function f, Newton's method finds solutions to the equation $f' = 0$ i.e. the critical points (maxima, minima or saddle/inflexion). In this case, Newton's method on f' is a stronger version of gradient descent as it requires knowledge of f's second derivative.

The method

Let $f\colon \mathbb{R} \to \mathbb{R}$ and suppose that we wish to find a value θ^* of θ so that $f(\theta^*) = 0$. The goal is to pick an initial guess θ_0 and construct a sequence of iterates $\{\theta_k\}$ that converge towards a minimiser of f. The next iterate θ_{k+1} is found by minimising the second-order Taylor approximation about the prior iterate θ_k:

$$f(\theta_k + \varepsilon) \approx f(\theta_k) + f'(\theta_k)\varepsilon + \frac{f''(\theta_k)}{2}\varepsilon^2$$

by finding the critical point of the quadratic in ε

$$0 = \frac{\partial}{\partial \varepsilon}\left(f(\theta_k) + f'(\theta_k)\varepsilon + \frac{f''(\theta_k)}{2}\varepsilon^2\right) \implies \varepsilon = \frac{-f'(\theta_k)}{f''(\theta_k)}$$

and setting $\theta_{k+1} = \theta_k + \varepsilon$. Thus, the update rule is

$$\theta_{k+1} := \theta_k - \frac{f'(\theta_k)}{f''(\theta_k)}.$$

If $\boldsymbol{\theta}$ is vector-valued, Newton's method can be generalised and the update rule becomes $\boldsymbol{\theta} := \boldsymbol{\theta} - H^{-1}\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$, where H is a $(p+1) \times (p+1)$ matrix of all mixed second-order partials called the Hessian and $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ is the usual vector of partial derivatives.

Newton's method enjoys a property called quadratic convergence and is typically faster than batch-gradient descent and requires fewer iterations to get a good approximation for a minimiser. However, each iteration is potentially more expensive because it requires finding and inverting a Hessian matrix (of mixed second order partials).

It definitely has its uses despite not being very popular for requiring more information (in the form of second order derivatives).

4.4.1 Newton-Raphson

The Newton-Raphson method is another method for root-finding and uses tangent lines to approximate a root. Initialise a guess $\theta^{(0)}$ and update the value of $\theta^{(i)}$ by choosing the point where the tangent line has a root

$$\theta^{(i+1)} := \theta^{(i)} - \frac{f'(\theta^{(i)})}{f''(\theta^{(i)})}$$

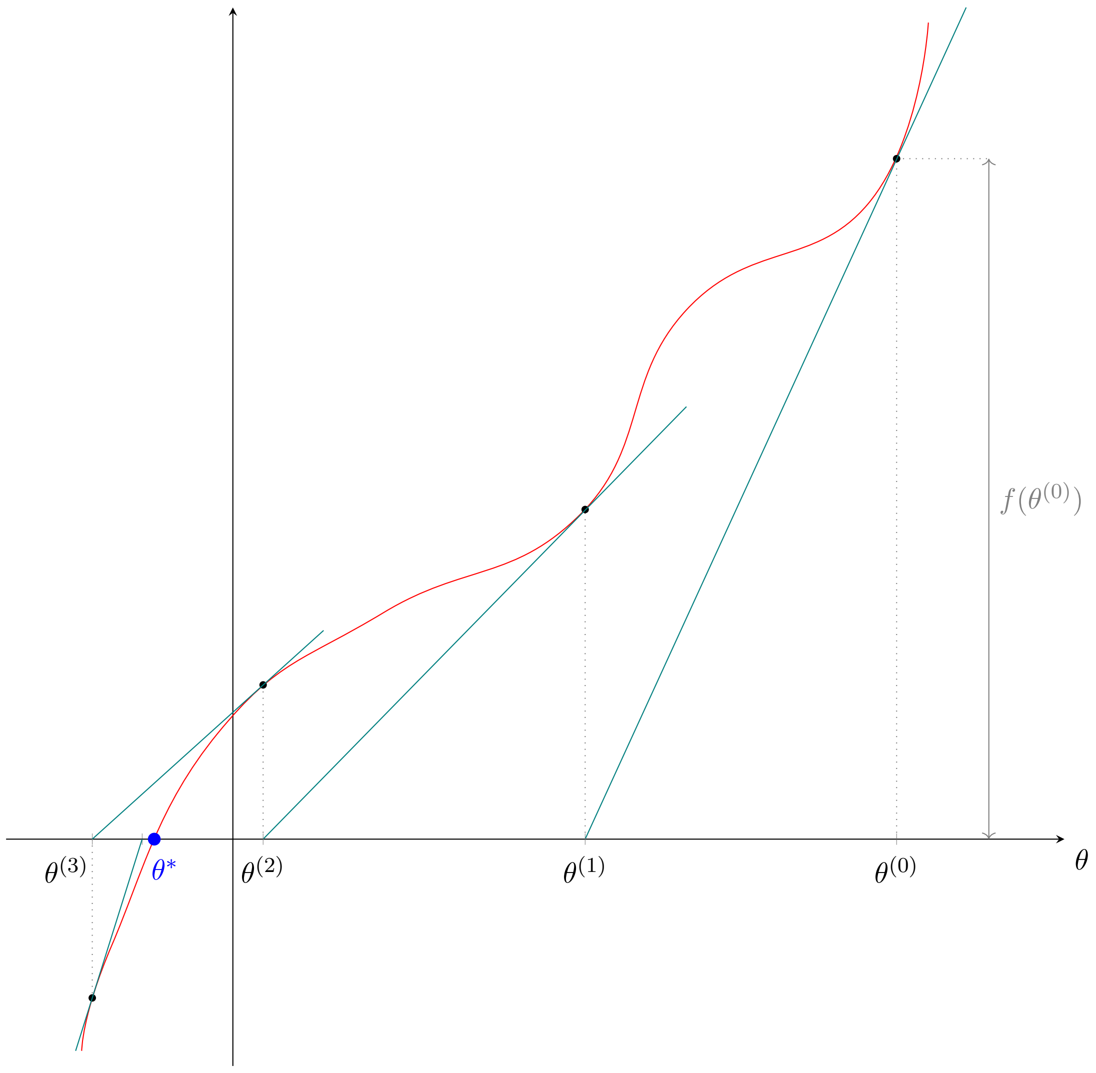


Figure 4.6: A visualisation of using the Newton-Raphson method to approximate the root θ^{*}.

Everything above this point has been fixed.

Any chapter below this point (minus the appendices) might need to be fixed.

CHAPTER 5

Generalised Linear Models (GLMs)

Thus far, we've seen two supervised learning models that arise from natural probabilistic assumptions about data:

Ordinary linear regression	$\{Y\|\mathbf{X}=x;\boldsymbol{\theta}\} \sim \mathcal{N}(\boldsymbol{\theta}^\top x, \sigma^2)$
Logistic regression	$\{Y\|\mathbf{X}=x;\boldsymbol{\theta}\} \sim \text{Bernoulli}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^\top x}}\right)$

These two have a **linear** (in $\boldsymbol{\theta}$) component $\boldsymbol{\theta}^\top x$ in common. This is not a coincidence and turns out to be part of a more general unifying theory - the theory of G**L**Ms. GLMs are just a general way to model data.

The ordinary linear regression model assumes that the conditional distribution of the continuous random variable $Y|\mathbf{X}=x;\boldsymbol{\theta}$ is Gaussian

$$Y|\mathbf{X}=x;\boldsymbol{\theta} \sim \mathcal{N}(\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta}),\sigma^2)$$

and that the conditional mean (cond. expectation) $\mu(x) := \mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta})$ of this distribution is modelled by a hypothesis $h_{\boldsymbol{\theta}}$ that incorporates the predictors in a linear fashion i.e.

$$\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta}) \text{ is modelled by } h_{\boldsymbol{\theta}}(x)=\boldsymbol{\theta}^\top x.$$

Generalised Linear Model

1. The first step to constructing a GLM incorporates the predictors $\mathbf{X}$ as a linear combination denoted $\eta := \boldsymbol{\theta}^\top\mathbf{X}$. This is the L part of a GLM.

2. The generalisation part comes with the choice of conditional distribution for the output:

$$Y|\mathbf{X}=x;\boldsymbol{\theta} \sim \text{ExponentialFamily}(\eta).$$

Given such a choice of distribution, if we were to model the conditional expectation (the value you expect to sample from such a distribution) with $\eta=\boldsymbol{\theta}^\top\mathbf{X}$ as before, we may run into issues:

e.g. Take it as fact (for now) that the Bernoulli distribution is an example of an Exponential-Family distribution. Suppose that $Y|\mathbf{X}=x;\boldsymbol{\theta}\sim\text{Bernoulli}(\varphi)$. Then $\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta})\in[0,1]$. However, modelling $\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta})=\boldsymbol{\theta}^\top x$ would suggest that varying x means that $\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta})$ can take values all over the real line $\mathbb{R}$. This is clearly false.

3. To rectify this, we define what's known as a link function g to make our model for the conditional expectation compatible with the chosen distribution for our output $Y|\mathbf{X}=x;\boldsymbol{\theta}$ as follows:

$$\begin{aligned}\mathbb{E}(Y\mid\mathbf{X}=x;\boldsymbol{\theta}) &= g^{-1}(\boldsymbol{\theta}^\top x)\\ &=: g^{-1}(\eta).\end{aligned}$$

There are three different parameterisations of a GLM:

$$\begin{tikzcd}
\text{model parameter} & \text{natural parameter} & \text{canonical parameters} \\
\boldsymbol{\theta} \arrow[r, mapsto, "\boldsymbol{\theta}^\top x"'] & \eta \arrow[r, shift left=2, "g"] & \begin{array}{c} \varphi \text{ - Bernoulli} \\ \mu, \sigma^2 \text{ - Gaussian} \\ \lambda \text{ - Poisson} \end{array} \arrow[l, shift left=2, "g^{-1}"]
\end{tikzcd}$$

$\boldsymbol{\theta}$ parameterises the exponential family through η. Whenever we learn a GLM, we only learn $\boldsymbol{\theta}$.

5.1 The Exponential Family

The exponential family is a class of families of distributions. In a sense, it's one of the simplest ways to relate $\mathbf{X}$ and $\boldsymbol{\theta}$. Think of $\mathbf{X}$ and $\boldsymbol{\theta}$ as one-dimensional to start with (X, θ). The exponential family restricts the way they can interact with each other and its density has a form symmetric in $\boldsymbol{\theta}$ and x:

$$p_X(x; \theta) = \exp(\theta x) f(x) g(\theta).$$

This naturally generalises to vectors $\mathbf{X}$ and $\boldsymbol{\theta}$. If they are of different dimensions, we can instead consider the inner product of a function η of $\boldsymbol{\theta}$ (that embeds it into a space of dimension matching with $\mathbf{X}$) with a function T of $\mathbf{X}$ e.g. $\langle T(\mathbf{X}), \eta(\boldsymbol{\theta}) \rangle$. Then our density looks like

$$p_{\mathbf{X}}(x; \boldsymbol{\theta}) = \exp(\langle \eta(\boldsymbol{\theta}), T(x) \rangle) f(x) g(\boldsymbol{\theta}).$$

A family of distributions $\mathcal{F} = \{\mathbb{P}_\eta\}$ is said to be an exponential family parameterised by η i.e. $\mathcal{F} \in \text{ExponentialFamily}(\eta)$ if its probability density function can be written in the form

$$p(y; \eta) = b(y) \exp(\eta^\top T(y) - a(\eta)).$$

- y is the output
- η is called the natural parameter
- $T(y)$ is called the sufficient statistic

  (It will often be the case that the distributions have $T(y) = y$)
- $b(y)$ is called the base measure

  It allows us to consider both discrete and continuous variables within the framework of exponential families. It can be thought of as a "change of measure."
- $a(\eta)$ is called the log-partition function

  The quantity $\exp(-a(\eta))$ plays the role of a normalisation constant, ensuring that the density $p(y; \eta)$ sums/integrates to 1.

For a fixed choice of (T, a, b), this defines a family of distributions indexed by η i.e. $\{p(y; \eta)\}_\eta$.

5.1.1 Examples of Exponential Family Distributions

- A Bernoulli distributed discrete random variable $X \sim \text{Bernoulli}(\varphi)$ is in the exponential family. Its mass function can be written as:

$$\begin{aligned} p_X(y;\varphi) &= \varphi^y(1-\varphi)^{1-y} \\ &= \exp(\log(\varphi^y(1-\varphi)^{1-y})) \\ &= \exp(y\log(\varphi^y) + (1-y)\log(1-\varphi)) \\ &= \exp\left(y\log\left(\frac{\varphi^y}{1-\varphi}\right) + \log(1-\varphi)\right) \end{aligned}$$

$b(y) = 1, T(y) = y$. Also $\eta^\top = \eta = \log\left(\frac{\varphi^y}{1-\varphi}\right)$ so we can rearrange for[1] $\varphi = \frac{e^\eta}{1+e^\eta} = \frac{1}{1+e^{-\eta}}$ which implies that $a(\eta) = -\log(1-\varphi) = \log(1+e^\eta)$.

Thus, the class of Bernoulli distributions $\{p(y;\varphi)\}_\varphi$ parameterised by φ is in the exponential family.

- Let X be a continuous, normally distributed random variable with variance $\sigma^2 = 1$ and mean μ. We claim this distribution is in the exponential family. Its density function is:

$$\begin{aligned} f(y;\mu) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y-\mu)^2}{2}\right) \\ &= \frac{1}{\sqrt{2\pi}}e^{-y^2/2}\exp\left(\mu y - \frac{1}{2}\mu^2\right) \end{aligned}$$

So $b(y) = e^{-y^2/2}$, $T(y) = y$, $\eta = \mu$ and $a(\eta) = \frac{1}{2}\mu^2 = \frac{1}{2}\eta^2$ and the claim follows.

The exponential family has some nice properties.

- Maximum likelihood estimation with respect to η is concave i.e. $\log(\mathbb{P}(Y;\eta))$ is concave in η. Since MLE is concave, NLL is convex in η.
- $\mathbb{E}(Y;\eta) = \frac{\partial}{\partial\eta}a(\eta)$
- $\text{Var}(Y;\eta) = \frac{\partial^2}{\partial\eta^2}a(\eta)$

There are various other members of the Exponential Family that can be used to model random variables that take on different types of values:

Data Type	Exponential Family Distribution	Model Name
$\mathbb{R}$	Gaussian/Laplace	Regression
$\{0,1\}$	Bernoulli	Classification
$\mathbb{Z}^+$	Poisson	Count/Poisson Regression
$\mathbb{R}^+$ e.g. time	Gamma/Exponential	Survival Analysis
$\{1,\dots,K\}$	Multinomial	Softmax Regression
Bernoulli Distributions	Beta	?
Categorical Distributions	Dirichlet	?

Back to supervised learning.

The general framework is the same as always. We've been given some training set $\mathcal{T} = \{(x^{(i)}, y^{(i)})\colon i = 1,\dots,n\}$ and we wish to learn a function that predicts a label given a vector of features.

- First make a choice of distribution for the response according to its data type.
- Express this distribution as a member of the ExponentialFamily. This involves finding $a(\eta)$, $b(y)$ and $T(y)$. Most importantly, we find an appropriate link function g so we can model the mean parameter of the distribution (μ, φ etc.) i.e. the conditional expectation with $g(\eta)$.

  At this point, we've supposed that for each $i = 1,\dots,n$:

$$Y|\mathbf{X} = x^{(i)};\boldsymbol{\theta} \sim \text{ExponentialFamily}(\boldsymbol{\theta}^\top x^{(i)}).$$

[1]It's no coincidence that this is the sigmoid function. It relates to the logistic regression algorithm.

- Our hypothesis is $h_{\boldsymbol{\theta}}(x) = g(\boldsymbol{\theta}^\top x)$.
- We intend to learn suitable parameters $\boldsymbol{\theta}$ by doing MLE.

What do we actually gain from developing a unifying framework like GLMs apart from some elegant mathematics?

The most useful benefit of a GLM is that when it comes to maximum likelihood estimation w.r.t $\boldsymbol{\theta}$ on the log likelihood, the learning update rule for gradient descent is always the same regardless of which exponential family distribution is chosen:

$$\theta_j = \theta_j + \alpha \left(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)}) \right) x_j^{(i)} \quad \text{for } j = 1, \dots, n \text{ simultaneously.}$$

[Generalised Linear Models] share a number of properties, such as linearity, that can be exploited to good effect and there is a common method for computing parameter estimates.

McCullagh and Nelder - GLMs (1989)

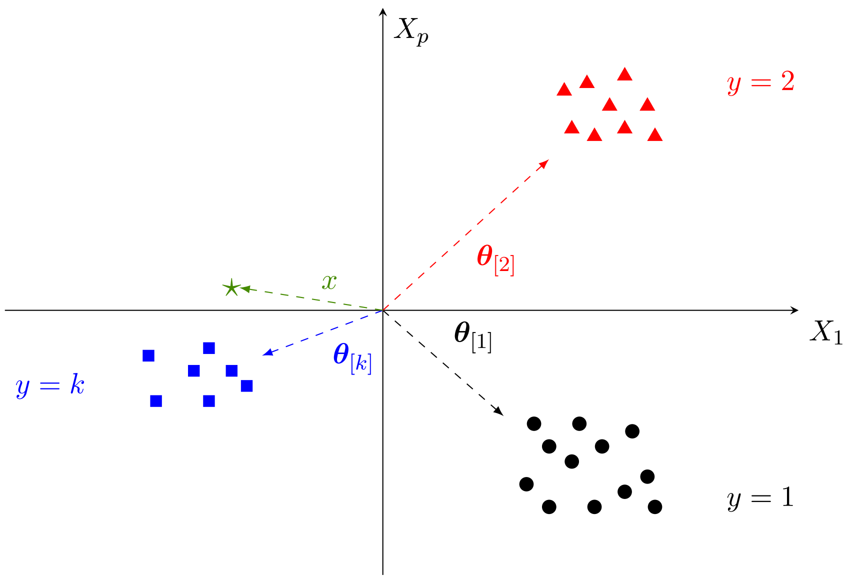
Might help to have a visualisation of GLM assumptions.

5.2 Multi-class Classification (Softmax Regression)

Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0,1\}^k$.

The following is a non-GLM approach which doesn't feature in the lecture notes. Suppose we have data $\mathcal{T}$ that looks like it fits into k distinct classes. Let the label of each example $x^{(i)} \in \mathbb{R}^p$ be a tuple $y^{(i)} \in \mathcal{Y} = \{0,1\}^k$. For example, if $y^{(i)} = (0, \dots, 0, 1)$, then that means the object $(x^{(i)}, y^{(i)})$ belongs to class k.

We can visualise the input space $\mathcal{X} \subseteq \mathbb{R}^p$ as two dimensional for ease of interpretation:



X_p
$y = 2$
$\boldsymbol{\theta}_{[2]}$
x
X_1
$\boldsymbol{\theta}_{[1]}$
$\boldsymbol{\theta}_{[k]}$
$y = k$
$y = 1$

The goal of multi-class classification is to learn a model that can predict the class which best fits a new unseen example ⋆ whose feature vector is x. The main idea is to take the idea of logistic regression where we outputted a single value for the positive class $\mathbb{P}(Y = 1 \,|\, \mathbf{X} = x; \boldsymbol{\theta})$ and extend it to output a vector of probabilities for each class:

$$\mathbf{h}_{\boldsymbol{\theta}}(x) = \begin{bmatrix} \mathbb{P}(Y = 1 \mid \mathbf{X} = x; \boldsymbol{\theta}) \\ \mathbb{P}(Y = 2 \mid \mathbf{X} = x; \boldsymbol{\theta}) \\ \vdots \\ \mathbb{P}(Y = k \mid \mathbf{X} = x; \boldsymbol{\theta}) \end{bmatrix}.$$

In softmax regression, we associate to each class C a vector of parameters $\boldsymbol{\theta}_{[C]}$. This can be seen in the figure above. The intuition behind this is that each $\boldsymbol{\theta}_{[C]}$ should point in the direction of the members of class C so that only those members will give a dot product against $\boldsymbol{\theta}_{[C]}$ that is much larger than examples belonging to other classes. This is a criterion for separating classes from one another. Each $\boldsymbol{\theta}_{[C]}^{\top} x$ can be thought of as a score for how well x matches the class C. We can write these class parameters in a matrix called the parameter matrix:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_{[1]}^{\top} \\ \boldsymbol{\theta}_{[2]}^{\top} \\ \vdots \\ \boldsymbol{\theta}_{[k]}^{\top} \end{bmatrix}.$$

Given an example $x^{(i)}$, we can plot $\boldsymbol{\theta}_{[C]}^{\top} x^{(i)}$ against the classes C. This is called the logit space.

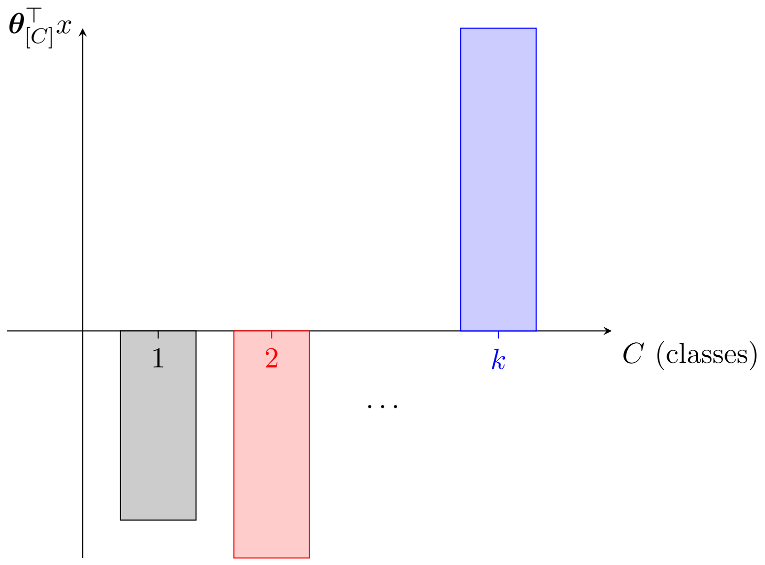


Figure 5.1: A visualisation of the logit space for a test feature vector x. The values of $\boldsymbol{\theta}_{[C]}^{\top} x$ can be negative and greater than 1 which we can't interpret as probabilities.

Probabilities need to be positive and sum up to 1. To define the posterior probability of a class C given an input feature vector, we:

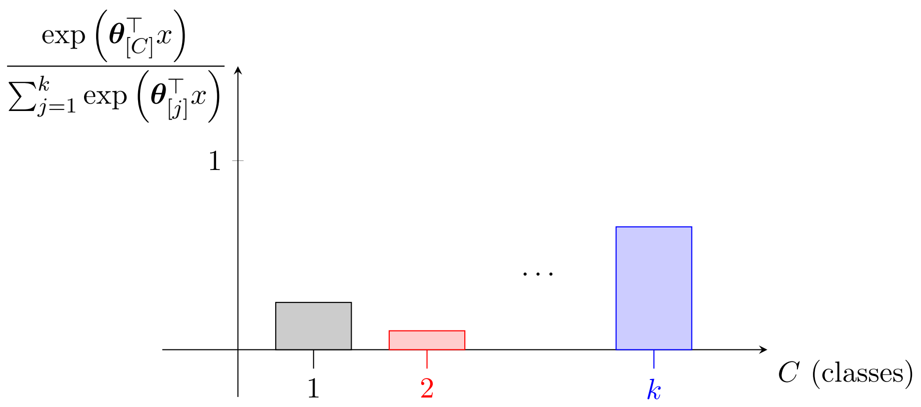
- exponentiate to preserve the "order" of the $\boldsymbol{\theta}^{\top} x$ and make everything positive

$$\exp\left(\boldsymbol{\theta}_{[C]}^{\top} x\right)$$

- and then normalise to get everything in $[0, 1]$ i.e. $\forall C = 1, \dots, k$:

$$\mathbb{P}(Y = C \mid \mathbf{X} = x; \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta}_{[C]}^{\top} x\right)}{\sum_{j=1}^{k} \exp\left(\boldsymbol{\theta}_{[j]}^{\top} x\right)}.$$

These equations define the softmax/multi-class classification model.



A new example x is classified by picking the C for which $\mathbb{P}(Y = C \mid \mathbf{X} = x; \boldsymbol{\theta})$ is largest.

5.2.1 GLM Approach

Unwieldy notation that I'm yet to address.

CHAPTER 6

Generative Learning Algorithms

Statistical classification models (used for regression/classification) fall under three distinct categories, computing classifiers by different approaches:

1. A *discriminative*[1] algorithm learns the conditional distribution of $Y|\mathbf{X} = x$ (for any given datapoint $\mathbf{X} = x$).
2. A *generative* algorithm learns the joint probability distribution of $(\mathbf{X}, Y)$ (over the entire data) and then uses Bayes' theorem to (calculate the "posterior" and thus) define a classifier.
3. Classifiers computed without a probability model are also referred to (loosely) as "discriminative."

All of the classification algorithms discussed thus far have been discriminative. For discriminative algorithms, we maximise the conditional likelihood. For generative algorithms, we instead maximise the joint likelihood to fine-tune the relevant parameters.

Having modelled the joint distribution, at classification time we can predict the posterior distribution $\mathbb{P}(Y = y \,|\, \mathbf{X} = x)$ using $\mathbb{P}(\mathbf{X} \,|\, Y = y)$ and $\mathbb{P}(Y = y)$ (the class prior) in Bayes' rule:

$$\mathbb{P}(Y = y \,|\, \mathbf{X} = x) = \frac{\mathbb{P}(\mathbf{X} = x \,|\, Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(\mathbf{X} = x)}.$$

In the case that Y is binary, we can write the posterior as:

$$\frac{\mathbb{P}(\mathbf{X} = x \,|\, Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(\mathbf{X} = x \,|\, Y = 0)\mathbb{P}(Y = 0) + \mathbb{P}(\mathbf{X} = x \,|\, Y = 1)\mathbb{P}(Y = 1)}. \tag{6.1}$$

6.1 GDA for Binary Classification

Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$.

The setting is binary classification. A discriminative approach is to learn a classifier that serves as a linear separating boundary between the positive and negative classes as seen with logistic regression. An alternative approach is generative and involves building a model for each class i.e. learning the typical features of each class separately. At classification time for an unseen example x, the class that resembles x most closely is the one assigned to it.

Gaussian discriminant analysis (GDA) is one such example of a generative learning algorithm. The GDA model for binary classification asserts that:

- $Y \sim \text{Bernoulli}(\varphi)$,
- $\mathbf{X}|Y = 0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma)$,
- $\mathbf{X}|Y = 1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma)$.

The parameters of this model are $\Sigma \in \mathbb{R}^{p\times p}$, $\varphi \in \mathbb{R}$ and $\boldsymbol{\mu}_0 \in \mathbb{R}^p \ni \boldsymbol{\mu}_1$. We can train these parameters using maximum likelihood estimation as we've done several times before. There is one key difference. Since we're approximating the joint distribution of the training data, we use the **joint** distribution instead of the conditional one we used for discriminative models. As before, we

[1] Also known as a conditional model.

suppose the examples in $\mathcal{T}$ were sampled independently and identically. The joint likelihood $\mathcal{L}$ of $\mathcal{T}$ can then be factored into

$$\begin{aligned}\mathcal{L}(\varphi, \mu_0, \mu_1, \Sigma) &= \prod_{i=1}^{n} \mathbb{P}_{\mathbf{X},Y}\left(x^{(i)}, y^{(i)}; \varphi, \mu_0, \mu_1, \Sigma\right) \\ &= \prod_{i=1}^{n} \mathbb{P}_{\mathbf{X}|Y}\left(x^{(i)} \,\middle|\, y^{(i)}; \mu_0, \mu_1, \Sigma\right) \mathbb{P}_Y\left(y^{(i)}; \varphi\right)\end{aligned}$$

By finding stationary points of the log-likelihood, we arrive at the following maximum likelihood estimates. The working out for all of these can be found in (C.1).

- $\widehat{\varphi}$ is an estimate for $\mathbb{P}_Y\left(y^{(i)} = 1\right)$ and we can simply interpret this estimate as the fraction of positive examples:
$$\widehat{\varphi} = \frac{\sum_{i=1}^{n} y^{(i)}}{n} = \frac{\sum_{i=1}^{n} \mathbb{1}\left(y^{(i)} = 1\right)}{n}$$
- The estimates of the means of both classes $\widehat{\mu_0}$ and $\widehat{\mu_1}$ can be interpreted as the averages of the example feature vectors of their respective classes. The formulae are:
$$\widehat{\mu_0} = \frac{\sum_{i=1}^{n} \mathbb{1}\left(y^{(i)} = 0\right) x^{(i)}}{\sum_{i=1}^{n} \mathbb{1}\left(y^{(i)} = 0\right)} \qquad \widehat{\mu_1} = \frac{\sum_{i=1}^{n} \mathbb{1}\left(y^{(i)} = 1\right) x^{(i)}}{\sum_{i=1}^{n} \mathbb{1}\left(y^{(i)} = 1\right)}.$$
- The covariance matrix parameter is shared between both distributions. Its estimate is:
$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^\top$$

6.1.1 Link Between GDA and Logistic Regression

Having found these parameters through MLE, to make a prediction for the most likely label $\widehat{y}$ of a test input feature vector x, one would need to compute

$$\begin{aligned}\widehat{y} = \operatorname*{argmax}_y \mathbb{P}_{Y|\mathbf{X}=x}(y \,|\, x) &= \operatorname*{argmax}_y \frac{\mathbb{P}_{\mathbf{X}|Y=y}(x \,|\, y)\mathbb{P}_Y(y)}{\mathbb{P}_X(x)} \\ &= \operatorname*{argmax}_y \mathbb{P}_{\mathbf{X}|Y=y}(x \,|\, y)\mathbb{P}_Y(y).\end{aligned}$$

The denominator need not be calculated as it's a constant in the argmax over y. The posterior distribution $\mathbb{P}(Y = 1 \,|\, \mathbf{X} = x)$ can be computed explicitly using (6.1):

$$\begin{aligned}\mathbb{P}(Y = 1 \,|\, \mathbf{X} = x) &= \frac{\mathbb{P}(\mathbf{X} = x \,|\, Y = 1)\mathbb{P}(Y = 1)}{\mathbb{P}(\mathbf{X} = x \,|\, Y = 0)\mathbb{P}(Y = 0) + \mathbb{P}(\mathbf{X} = x \,|\, Y = 1)\mathbb{P}(Y = 1)} \\ &= \frac{1}{1 + \frac{1-\varphi}{\varphi} \exp\left(\frac{-1}{2}(x - \mu_0)^\top \Sigma^{-1}(x - \mu_0) + \frac{1}{2}(x - \mu_1)^\top \Sigma^{-1}(x - \mu_1)\right)} \\ &= \frac{1}{1 + \exp\left(-\left[(\Sigma(\mu_1 - \mu_0))^\top \Sigma^{-1} x + (\mu_0 + \mu_1)^\top \Sigma^{-1}(\mu_0 - \mu_1) - \log((1-\varphi)/\varphi)\right]\right)} \\ &= \frac{1}{1 + \exp(-(\boldsymbol{\theta}^\top x + \theta_0))}\end{aligned}$$

where $\boldsymbol{\theta} = \Sigma(\mu_0 - \mu_1)^\top$ and $\theta_0 = (\mu_0 + \mu_1)^\top \Sigma^{-1}(\mu_0 - \mu_1) - \log((1 - \varphi)/\varphi)$.

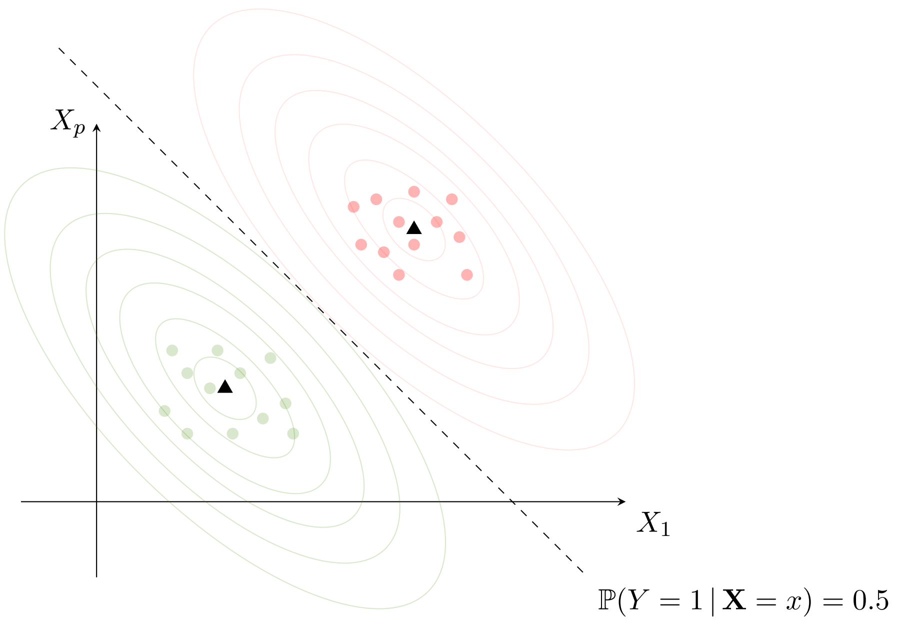


Figure 6.1: The contour plots of both conditional Gaussian distributions (negative in green, positive in red) and a dashed line to represent the decision boundary representing input features that are equally likely to be classified as positive and negative i.e. the set of x s.t. $\mathbb{P}(Y = 1 \mid \mathbf{X} = x) = 0.5$.

Thus, a GDA model uniquely determines a classifier that has a linear[2] decision boundary that arises from a logistic regression.

The converse, that $Y|\mathbf{X} = x$ following a Bernoulli distribution implies $\mathbf{X}|Y = y$ is multivariate Gaussian, is not true. logistic regression model is necessarily the posterior of a GDA model (or any model at that), is not true.

Thus, GDA makes more restrictive assumptions than logistic regression. If the training data exhibits these assumptions, GDA will be more sample-efficient (require less samples to make accurate predictions). Logistic regression on the other hand is pretty robust (i.e. tends to work pretty well even if assumptions aren't made) and so should generally be your first choice of algorithm in practice.

6.1.2 Extra Note on GDA

In GDA, are there any other/extra assumptions on the prior that are needed so that the posterior takes the form of a logistic function?

In general, if $\mathbf{X}|Y = y \sim \text{ExponentialFamily}$ then the posterior takes the form of:

- the logistic function if Y is binary-valued i.e. $\mathcal{Y} = \{0, 1\}$, or
- the softmax function if Y is categorical e.g. $\mathcal{Y} = \{1, \ldots, k\}$.

[2]Consider the model for which $\mathbf{X}|Y = 0$ and $\mathbf{X}|Y = 1$ don't share a covariance matrix. Instead, they have different covariance matrices Σ_0 and Σ_1, respectively. The dashed line in the contour figure above representing the linear decision boundary will no longer be linear - it will curve around the class that is more concentrated. **The posterior distribution in that case can be thought of as being modelled by logistic regression with polynomial features.** $\longleftarrow$ Seems reasonable and will verify later.

CHAPTER 7

Support Vector Machines

All the methods so far have generated linear classifiers. Suppose that the training data one has looks like a non-linear decision boundary would be appropriate:

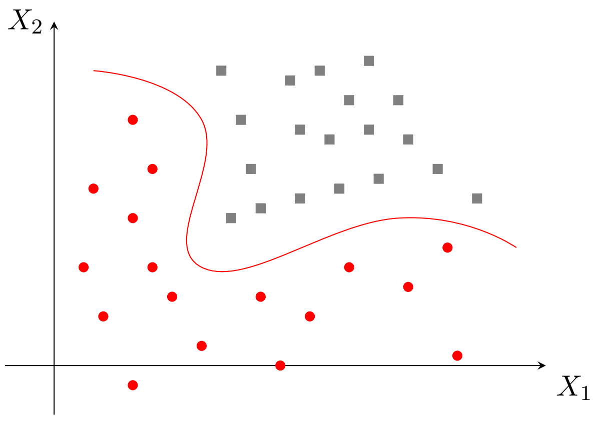


Figure 7.1: A visualisation of two-dimensional feature vectors $(x_1^{(i)}, x_2^{(i)}) \in \mathbb{X}_1 \times \mathbb{X}_2$ that fall into two classes and don't admit a linear decision boundary.

It turns out that one can use the methods we've developed so far like logistic regression and GDA to achieve a non-linear decision boundary. This is done by transforming, via a function ϕ, the feature vectors and embedding them into a higher dimensional space. An example of such a ϕ could take in a feature vector $\mathbf{X} = [X_1, X_2]^\top \in \mathbb{R}^2$ and output

$$\phi \colon \mathbf{X} \longmapsto \phi(\mathbf{X}) := \begin{bmatrix} X_1 \\ X_2 \\ X_1^2 \\ X_2^2 \\ X_1 X_2 \end{bmatrix} \in \mathbb{R}^5.$$

Performing logistic regression on these transformed features generates a non-linear decision boundary. It's quite a complicated procedure to manually pick a certain ϕ for the purposes of most problems, especially highly non-linear decision boundaries. SVMs (support vector machines) address this issue.

A *support vector machine* is a supervised learning model that is used for classification. Given a set of training data whose examples are labelled as belonging to one of two classes, an SVM training algorithm builds a non-probabilistic linear classifier. There are many decision boundaries that could work as a linear classifier. Which one should we choose? SVMs use the particular criterion that an optimal decision boundary is one that puts the widest "separation" between the two classes of training examples.

SVMs are also used to perform non-linear classification by using a function ϕ to map feature vectors in $\mathcal{X}$ into a higher-dimensional space $\phi(\mathcal{X})$ where the data is more likely (but certainly not guaranteed) to be linearly separable. Higher dimensional spaces carry the expense of greater computational cost when it comes to manipulating vectors $\phi(x) \in \mathbb{R}^d$ (e.g. $\phi(x)^\top \phi(z)$) compared to vectors $x \in \mathbb{R}^p$ with $p < d$. A key observation is that during the optimisation process for finding a "suitable" separating boundary, the feature vectors present themselves in a very specific form, namely an inner product $\langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$. This form can be represented by a simpler and more computationally efficient map K known as a kernel. In ML, the kernel trick has its historical roots in isolated digit recognition through work by Vladimir Vapnik.

Some pros and cons of SVMs are:

+ They are useful for potentially highly non-linear decision boundaries.

+ The algorithmic implementation of the SVM is of turn-key type i.e. there aren't many parameters to fiddle with (like the learning rate α in gradient descent).

- They aren't as useful as neural networks nowadays.

The build-up of the theory will be as follows:

- Addressing the linearly separable case with a primitive version of an SVM known as an optimal margin classifier.
- The method of Lagrange multipliers and duality to discuss optimisation methods to find an optimal margin classifier.
- The idea of kernels.
- The inseparable case.

The media talks a lot about machine learning as just neural networks. The set of algorithms in practice is much wider than neural networks and deep learning.

Andrew Ng

The setting for all of this theory will be binary[1] classification. For mathematical convenience, a change of notation is in order. Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{-1, +1\}$ so that -1 represents a negative example and $+1$ a positive one. The parameters $\boldsymbol{\theta} \in \mathbb{R}^{p+1}$ are now separated into $w \in \mathbb{R}^p$ and an intercept term $b \in \mathbb{R}$. The linear classifier will be denoted as $h_{w,b}$ and defined for $x \in \mathcal{X}$ by $h_{w,b}(x) = g(w^\top x + b)$ where

$$g(z) = \begin{cases} +1, & \text{if } z \geqslant 0 \\ -1, & \text{if } z < 0. \end{cases}$$

7.1 Separable Case - Optimal Margin Classifier

Suppose that a training set $\mathcal{T}$ is linearly separable i.e. there exists a hyperplane of the form $w^\top x + b = 0$ for which all positive examples (labelled $y^{(i)} = +1$) satisfy $w^\top x^{(i)} + b > 0$ and all negative examples satisfy $w^\top x^{(i)} + b < 0$. These two conditions can be summarised as $y^{(i)}(w^\top x^{(i)} + b) > 0$ for all training examples.

Support vector machines attempt to minimise the generalisation error through the concept of a margin - the smallest distance between the decision boundary and any of the examples. In SVMs, the decision boundary chosen is the one that maximises margin.

The perpendicular distance of a point x_0 from a hyperplane $w^\top x + b = 0$ is $|w^\top x_0 + b|/\|w\|$. We're only interested in solutions for which the entire training set has been classified correctly. Thus, the distance of each training example to the decision boundary is

$$\frac{y^{(i)}(w^\top x^{(i)} + b)}{\|w\|} =: \gamma^{(i)}$$

[1]In the setting of a multi-class classification problem, one can combine binary SVMs. For example, if N is the number classes, one can train N *one-versus-rest* (say "one" positive, "rest" negative) binary classifiers. To then classify a test point, we pick the class corresponding to the greatest positive distance from its respective boundary to the test point.

and we call $\gamma^{(i)}$ the geometric margin of (w, b) with respect to the example $(x^{(i)}, y^{(i)})$.

The margin is defined to be the closest example to the hyperplane and our goal is to search for parameters (w, b) that maximise this margin i.e. we desire

$$\operatorname*{argmax}_{w,b}\left(\frac{1}{\|w\|}\min_i(y^{(i)}(w^\top x^{(i)}+b))\right).$$

Directly solving this is tough without extra constraints.

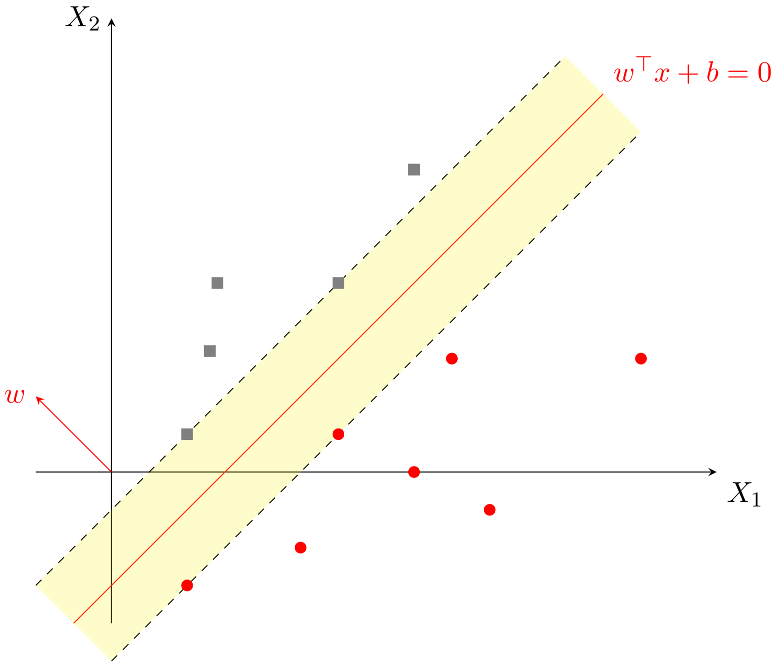


Figure 7.2: A visualisation of the maximum-margin hyperplane separating two classes.

An obvious restriction to make is that our training examples don't fall into the "margin" which is the region shaded in yellow. For this reason, we impose that $y^{(i)}(w^\top x^{(i)}+b) \geqslant \delta$ for some $\delta > 0$. Note that the geometric margin of (w, b) w.r.t an example $(x^{(i)}, y^{(i)})$ is invariant under re-scaling the parameters $(w, b) \mapsto (cw, bw)$. This is because:

$$\begin{aligned}\gamma^{(i)} &= y^{(i)}\left(\frac{w^\top}{\|w\|}x^{(i)}+\frac{b}{\|w\|}\right)\\ &\mapsto y^{(i)}\left(\frac{cw^\top}{|c|\|w\|}x^{(i)}+\frac{cb}{|c|\|w\|}\right) = \operatorname{sign}(c)\gamma^{(i)} = \gamma^{(i)}.\end{aligned}$$

This gives us the freedom to rescale $(w, b) \mapsto (w/\delta, b/\delta)$ so that $y^{(i)}(w^\top x^{(i)}+b) = 1$ for the example closest to the decision boundary. The re-scaling to 1 is purely because 1 is the simplest positive real number. In this case, all the examples will satisfy

$$y^{(i)}(w^\top x^{(i)}+b) \geqslant 1 \qquad \text{for } i = 1, \dots, n.$$

These constraints are known as the canonical representation of the decision boundary. For examples where equality holds, an example's constraint is called active (and inactive for the remainder of example constraints). By definition of there always being at least one closest point, there will always be one active constraint.

The optimisation problem then becomes $\operatorname{argmax}_{w,b}(1/\|w\|)$ subject to the constraints. This is equivalent to

$$\begin{cases}\operatorname*{argmin}_{w,b}(\|w\|^2/2)\\ y^{(i)}(w^\top x^{(i)}+b)-1 \geqslant 0 \qquad \text{for } i = 1, \dots, n\end{cases}$$

7.2 The Dual Representation via Lagrange Multipliers

To solve this constrained optimisation problem, we can use the method of Lagrange multipliers to incorporate the constraints as a weighted sum (whose weights are called Lagrange multipliers) into a function that involves the term we'd like to minimise. For each constraint $c_i \geqslant 0$ for $i = 1, \dots, n$, we multiply c_i by a constant $\alpha_i \geqslant 0$ called a Lagrange multiplier and subtract it from the objective function. This defines the Lagrangian:

$$\begin{aligned}\mathcal{L}(w, b, \alpha) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y^{(i)}(w^\top x^{(i)} + b) - 1) \\ &= \sum_{i=1}^{n} \alpha_i y^{(i)} + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y^{(i)}(w^\top x^{(i)} + b)\end{aligned}$$

Then we minimise this with respect to w and b.

$$\nabla_w \mathcal{L} = \mathbf{0} \implies w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

Substituting these back into the Lagrangian gives what's known as the Wolfe dual representation of the optimisation problem in which we maximise

$$\widetilde{\mathcal{L}}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y^{(i)} y^{(i')} (x^{(i)})^\top x^{(i')}$$

subject to $\alpha_i \geqslant 0$ for $i = 1, \dots, n$ and $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$.

This is a quadratic optimisation problem in α subject to a set of inequality constraints. The $y^{(i)} y^{(i')}$ term compares the output labels of each observation and $x^{(i)} \cdot x^{(i')}$ is a representation of how similar the two vectors are. The $\alpha_i \alpha_{i'}$ term determines whether or not this pair of observations is relevant in defining the decision boundary.

A new input feature vector u is classified according to the sign of

$$y_u = \sum_{i=1}^{n} \alpha_i y^{(i)} (x^{(i)})^\top u + b.$$

Interestingly, in both the expressions for $\widetilde{\mathcal{L}}(\alpha)$ and the decision rule for classifying a new u, the training examples present themselves only in the form of a scalar product $(x^{(i)})^\top x^{(i')}$ and $(x^{(i)})^\top u$ respectively. This is a fundamental observation and the reason why the optimal margin classifier is amenable to generalisation to the case where the training examples (as they are currently) aren't linearly separable. In this case, one idea is that perhaps embedding (via a function ϕ) into a higher dimensional space may separate the points through some hyperplane.

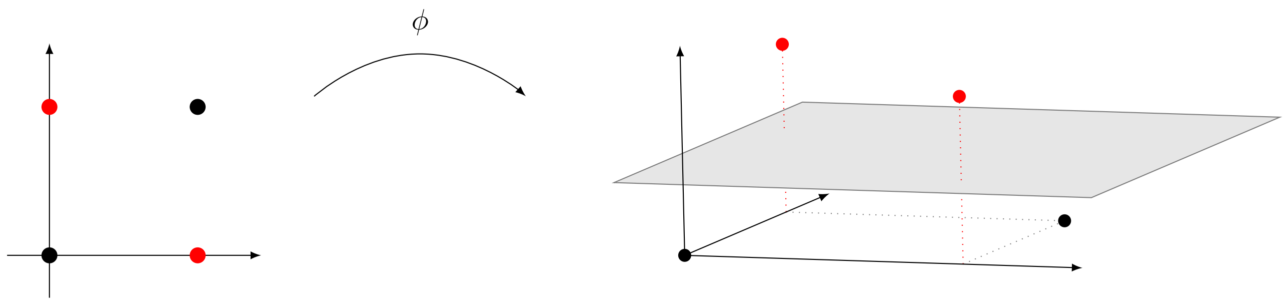


Figure 7.3: A linearly inseparable training set that becomes linearly separable (with a hyperplane, in grey) once embedded into a higher dimensional space via ϕ.

Using this embedding, all instances of $(x^{(i)})^\top x^{(i')}$ in the model can be replaced with $\langle \phi(x^{(i)}), \phi(x^{(i')}) \rangle$. This form can be represented by a simpler and more computationally efficient map K known as a kernel and this defines a support vector machine.

Through the deeper and more general theory of Lagrange multipliers, a constrained optimisation problem of the Wolfe dual form satisfies the Karush-Kuhn-Tucker conditions. In our specific case, there are only 3 conditions:

$$\begin{cases} \alpha_i \geqslant 0 \\ y^{(i)}(w^\top x^{(i)} + b) - 1 \geqslant 0 \\ \alpha_i(y^{(i)}(w^\top x^{(i)} + b) - 1) = 0 \end{cases}$$

This means that each example $(x^{(i)}, y^{(i)})$ satisfies either $\alpha_i = 0$ or $y^{(i)}(w^\top x^{(i)} + b) = 1$.

- For the former, any example for which $\alpha_i = 0$ plays no role in prediction for an unseen vector input u as the term involving α_i doesn't appear in the sum for y_u.
- The remaining examples are called support vectors and lie on the maximum margin hyperplanes.

Once a model is trained, most examples that aren't support vectors can be discarded. This makes SVMs quite practical.

7.3 Kernels

Kernels avoid the need for explicit computation using the embedding ϕ directly. Certain functions $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be represented as an inner product in another space $\phi(\mathcal{X})$.

eg Let $u, v \in \mathbb{R}^n$ and $r, \gamma, \sigma \in \mathbb{R}$. Some common kernels include:

1. Linear: $\langle u, v \rangle$
2. Polynomial: $(\gamma\langle u, v \rangle + r)^d$
3. Sigmoid: $\tanh(\gamma\langle u, v \rangle + r)$
4. Radial Basis Function "rbf"/Gaussian: $e^{-\|u-v\|/\sigma}$ where $\|u - v\| = \sqrt{\langle u - v, u - v \rangle}$.

Chapter 8

Regularisation and Bayesian Statistics

8.1 Bias and Variance

Given a model, it's possible to make the values of $h_{\boldsymbol{\theta}}(x^{(i)})$ as close as we like to the $y^{(i)}$ by including a sufficiently large number of parameters. More complex models are called *highly flexible*. If a model is too flexible, it's not as interpretable as a simple model. It also doesn't help that highly flexible models can suffer from *overfitting* i.e. when the model works too hard to pick up on patterns in the training data that may just be due to random chance rather than by properties of the underlying phenomenon that generated the training data. Such a model doesn't translate well to other sets of data for the same phenomenon. Thus, an overfitted model has narrow scope in its predictive powers.

Two keywords to describe a model are:

- *Variance* refers to the amount by which h would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different h. But ideally the estimate for f should not vary too much between training sets. However, if a method has high variance, then small changes in the training data can result in large changes in h.
- *Bias* is error introduced by modelling i.e. the error introduced by reducing a complex phenomenon to a simple trend/model.

As before, we can measure the quality of fit of a model to $\mathcal{T}$ by looking at the statistical risk with quadratic loss (i.e. the mean squared error MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - h(y^{(i)}))^2.$$

Training MSE is a useful measure for how well the model we've learned fits to the given training data. However, we're often more interested in how our model performs on unseen data. Thus, we also aim to minimise what is known as the *test* MSE.

A *fundamental rule* of statistical learning is that while model flexibility increases, the training MSE decreases but the test MSE may not necessarily also decrease. A small training MSE and large test MSE indicate *overfitting*.

eg Consider the problem of fitting a polynomial $h_{\boldsymbol{\theta}}$ to training data $\mathcal{T}$ via linear regression. The optimisation objective was to find a value of $\boldsymbol{\theta}$ that minimised the residual sum of squares:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\frac{1}{2}\sum_{i=1}^{n}\|y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})\|^2\right]$$

As $\deg(h_{\boldsymbol{\theta}})$ increases, the training error decreases but the trade-off is that the parameters tend to grow very large in magnitude. The figure below shows polynomials of degrees 1, 2 and 16 fit to a set of data generated by a quadratic polynomial with noise:

The line p_1 clearly accounts for variation above and below the line but it's a very poor fit for the data. The quadratic model p_2 looks a lot better by eye and follows the data reasonably well. The 16$^{\text{th}}$ degree polynomial p_{16} overfits to the data on the right-hand-side and clearly has high variance. A different set of data from the same phenomenon would generate a very different hypothesis.

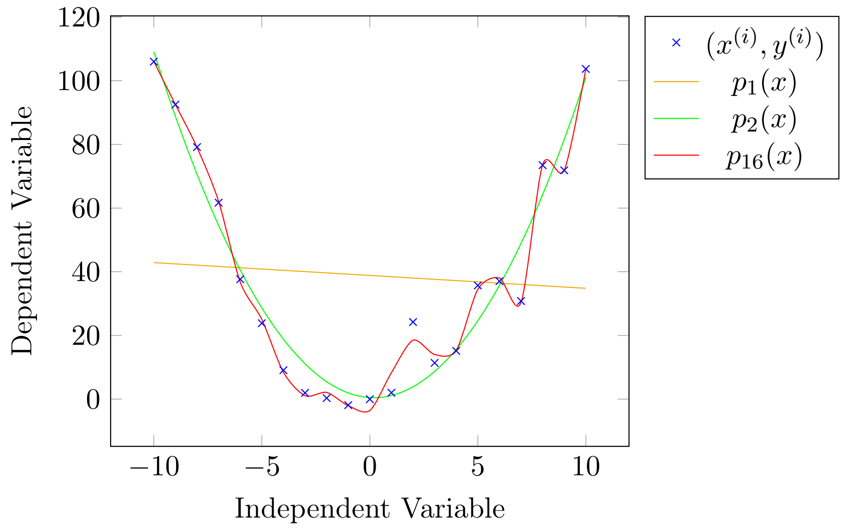


Figure 8.1: Data generated by $\mathbb{Z} \cap [-10, 10] \ni x \longmapsto (x^2 + \omega)$ where $\omega \sim N(0, 8)$ along with best-fit plots of orders 1, 2 and 16.

8.2 Regularisation

One of the most effective ways to prevent overfitting is regularisation. The idea of regularisation is to incorporate a term involving the parameters to the cost function that incentivises the chosen searching algorithm (for $\boldsymbol{\theta}$) to make the values of θ_j smaller. For linear regression, regularisation changes[1] the optimisation objective to

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{i=1}^{n} \|y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)})\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right].$$

The parameter λ controls how much focus there is on minimising the parameters θ_j relative to the sum of squares. $\lambda = 0$ means there is no regularisation. In the case of polynomial fitting, $\lambda \to \infty \implies |\theta_j| \to 0 \implies h_{\boldsymbol{\theta}}(x) = 0$. It's important to pick a value for λ that isn't too large or small.

Having just covered support vector machines that deal with potentially infinitely dimensional spaces, why don't they overfit? **It turns out that the optimisation problem of minimising $\|w\|^2/2$ has a similar effect to regularisation (and limits the space of potential decision boundaries in a special way).**

8.2.1 Bayesian Statistics (Maximum a-Posteriori Estimation)

In an analogous way to how the OLS cost function arose from performing MLE on a probabilistic model $Y | \mathbf{X} = x \sim \mathcal{N}(\boldsymbol{\theta}^\top x, \sigma^2)$ and finding

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{n} \mathbb{P}\Big(Y = y^{(i)} \,\Big|\, \mathbf{X} = x^{(i)}; \boldsymbol{\theta}\Big),$$

regularisation arises from a Bayesian statistics POV. We viewed $\boldsymbol{\theta}$ as an unknown but constant-valued parameter. This view of $\boldsymbol{\theta}$ is taken in a frequentist perspective of statistics. $\boldsymbol{\theta}$ is not random

[1]In the case where one wants to instead maximise an objective function instead, one subtracts the regularisation term from the cost function. Logistic regression is one such example:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\left(\sum_{i=1}^{n} \log(\mathbb{P}\Big(Y = y^{(i)} \,\Big|\, \mathbf{X} = x^{(i)}; \boldsymbol{\theta}\Big) \right) - \lambda \|\boldsymbol{\theta}\|^2 \right]$$

and it's the job of a frequentist to come up with some statistical procedure (like MLE) to try and estimate $\boldsymbol{\theta}$.

A Bayesian persepctive views $\boldsymbol{\theta}$ as a random variable whose value is unknown but before having seen any data, we have some prior beliefs i.e. we express our prior beliefs by specifying a prior distribution $\mathbb{P}(\boldsymbol{\theta})$ on $\boldsymbol{\theta}$. The goal is to find a value of $\boldsymbol{\theta}$ that is most likely after having observed the data i.e. to compute the posterior distribution of the parameters:

$$\begin{aligned}\mathbb{P}(\boldsymbol{\theta} \mid \mathcal{T}) &= \frac{\mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\mathcal{T})} \\ &\propto \text{likelihood} \times \text{prior}\end{aligned}$$

The denominator of the fraction is a normalisation constant to ensure the posterior is a probability distribution.

To make a prediction for a new test example u, we can compute the posterior distribution of the target using the posterior distribution on $\boldsymbol{\theta}$. Taking the equation above and integrating with respect to $\boldsymbol{\theta}$ gives

$$\underbrace{\int_{\boldsymbol{\theta}} \mathbb{P}(\boldsymbol{\theta} \mid \mathcal{T}) \mathrm{d}\boldsymbol{\theta}}_{=1} = \frac{1}{\mathbb{P}(\mathcal{T})} \int_{\boldsymbol{\theta}} \mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \implies \mathbb{P}(\mathcal{T}) = \int_{\boldsymbol{\theta}} \mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$$

This means that for the example u:

$$\mathbb{P}(Y \mid \mathbf{X} = u, \mathcal{T}) = \int_{\boldsymbol{\theta}} \mathbb{P}(Y \mid \mathbf{X} = u, \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}.$$

It's tough to compute such high-dimensional integrals for the posterior of $\boldsymbol{\theta}$ given $\mathcal{T}$, especially in closed form. Instead, we approximate the posterior distribution for $\boldsymbol{\theta}$ with a single point estimate called the maximum a posteriori (or MAP) estimate for $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta}_{\text{MAP}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\frac{\mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\mathcal{T})}\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} \mathbb{P}\Big(Y = y^{(i)} \,\Big|\, \mathbf{X} = x^{(i)}, \boldsymbol{\theta}\Big)\mathbb{P}(\boldsymbol{\theta})\right]\end{aligned}$$

This is the same formula as with maximum likelihood estimation but with an extra factor of the prior. Note that $\boldsymbol{\theta}$ in this case are being conditioned on since it's a random variable.

Here's where it links to regularisation. Instead of optimising the posterior directly, we often choose to optimise the negative logarithm of the posterior. This means that

$$\boldsymbol{\theta}_{\text{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \left[\log(\mathbb{P}(\mathcal{T} \mid \boldsymbol{\theta})) + \log(\mathbb{P}(\boldsymbol{\theta}))\right]$$

Assuming that the prior is normally distributed around 0 i.e. $\boldsymbol{\theta} \sim \mathcal{N}(0, \tau^2 I)$, you get

$$\begin{aligned}-\log(\mathbb{P}(\boldsymbol{\theta})) &= -\log\left(\frac{1}{(2\pi)^{p/2}(\det(\tau^2 I))^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\top (\tau^2 I)^{-1}\boldsymbol{\theta}\right)\right) \\ &= \text{constant} + \frac{1}{2\tau^2}\|\boldsymbol{\theta}\|_2^2.\end{aligned}$$

We can drop the constant term in the minimisation problem so $-\log(\mathbb{P}(\boldsymbol{\theta}))$ is proportional to $\|\boldsymbol{\theta}\|_2^2$. That's the regularisation term defined earlier from the frequentist point of view. Thus, MAP can be thought of as a regularisation of ML estimation. Some important points are:

- The parameters $\boldsymbol{\theta}_{\text{MAP}}$ fitted will then have smaller norm than those $\widehat{\boldsymbol{\theta}}$ selected by ML estimation and so Bayesian MAP is less susceptible to overfitting.

- If the training data set is very large (n large) then the regularisation term becomes negligible when compared to the likelihood term.
- Choosing a stronger prior limits the flexibility of the model in favour of preferring plausible solutions.

Chapter 9
Extra Notes

Underdetermined models have too few examples to estimate the parameters. (Similar to mathematical systems of equations where there are less equations than unknowns.)

Lasso = L1 regularisation, Ridge = L2 regularisation

Probability quantifies anticipation (of outcome) and likelihood quantifies trust (in model)

9.1 Statistics Server

The Boston housing dataset is a classic "getting started with regression" dataset. It's a good one and saves you from having to download and clean a lot of data yourself.

Table 2 fallacy (not taught in regression classes)

For a DS master's course, the bare minimum to be expected in terms of mathematical statistics background is MLE, sufficient/complete statistics and hypothesis testing.

CHAPTER A

Differentiation

A.1 Fréchet Derivative

The most general (and incredibly useful!!) definition of derivative that I'm aware of is that of the Fréchet derivative of a function between normed vector spaces.

Definition A.1.1 Let $(\mathbb{X}, \|\cdot\|_{\mathbb{X}})$ and $(\mathbb{Y}, \|\cdot\|_{\mathbb{Y}})$ be normed spaces, $U \subseteq \mathbb{X}$ be open, $x \in U$, and $f \colon U \to \mathbb{Y}$. If there is a bounded linear map $A_x \colon \mathbb{X} \to \mathbb{Y}$ that satisfies

$$\lim_{v \to 0 \text{ in } \mathbb{X}} \frac{\|f(x+v) - f(x) - A_x v\|_{\mathbb{Y}}}{\|v\|_{\mathbb{X}}} = 0 \tag{A.1}$$

then f is said to be **Fréchet differentiable at x**. The linear map A_x is then called the **Fréchet derivative of f at x** and is denoted $\mathrm{d}f(x)$.

Corollary A.1.2 Let $\mathbb{X} = \mathbb{R}^n$ with the standard Euclidean basis $\{e_1, \ldots, e_n\}$, $\mathbb{Y} = \mathbb{R}^m$ with similar remarks. Let $U \subseteq \mathbb{R}^n$ be open, and let $f \colon U \to \mathbb{R}^m$ be Fréchet differentiable at $x \in U$. Then the Jacobian matrix $Jf(x)$ of partial derivatives of f at x exists and is the matrix of the Fréchet derivative $\mathrm{d}f(x)$ with respect to the standard Euclidean bases of $\mathbb{R}^n$ and $\mathbb{R}^m$ i.e

$$\mathrm{d}f(x)v = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \frac{\partial f_m}{\partial x_2}(x) & \cdots & \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

for all $v = \sum_{i=1}^{n} v_i e_i \in \mathbb{R}^n$.

For the following, let $(\mathbb{X}, \|\cdot\|_{\mathbb{X}}) = (\mathbb{R}^n, \|\cdot\|)$, and $(\mathbb{Y}, \|\cdot\|_{\mathbb{X}}) = (\mathbb{R}, |\cdot|)$.

Definition A.1.3 The **gradient of $f \colon \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$** is defined as the vector

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^n$$

of partial derivatives of f.

Corollary A.1.4 The coordinate representation $Jf(x)$ of the Fréchet derivative $\mathrm{d}f(x)$ of $f \colon \mathbb{R}^n \to \mathbb{R}$, and the gradient $\nabla f(x)$ of f at x are related by the transpose operation

$$Jf(x) = (\nabla f(x))^\top.$$

Proof. The coordinate representation of the Fréchet derivative of $f \colon \mathbb{R}^n \to \mathbb{R}$ with respect to the standard basis $\{e_1, \ldots, e_n\}$ of $\mathbb{R}^n$ is a $1 \times n$ matrix i.e. a linear functional on $\mathbb{R}^n$, taking vectors to scalars i.e. a <u>row</u> of partial derivatives

$$Jf(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) & \cdots & \frac{\partial f}{\partial x_n}(x) \end{bmatrix}.$$

The claim follows. ■

Definition A.1.5 Let $(\mathbb{X}, \|\cdot\|_{\mathbb{X}})$ and $(\mathbb{Y}, \|\cdot\|_{\mathbb{Y}})$ be normed spaces, $U \subseteq \mathbb{X}$ be open, $x \in U$, and $f\colon U \to \mathbb{Y}$. If there exists a bounded-linear-map-valued bounded linear map $A_x\colon \mathbb{X} \to \mathcal{L}(\mathbb{X};\mathbb{Y})$ that satisfies

$$\lim_{v\to 0 \text{ in } \mathbb{X}} \frac{\|\mathrm{d}f(x+v) - \mathrm{d}f(x) - A_x v\|_{\mathbb{X}\to\mathbb{Y}}}{\|v\|_{\mathbb{X}}} = 0$$

then f is said to be twice **Fréchet differentiable at x**. The linear map A_x is then called the second-order **Fréchet derivative of f at x** and is denoted $\mathrm{d}^2 f(x)$.

Remarks A.1.6

- We can view an element of $\mathcal{L}(\mathbb{X}, \mathcal{L}(\mathbb{X};\mathbb{Y}))$ as a bounded bilinear map $A_x\colon \mathbb{X}\times\mathbb{X} \to \mathbb{Y}$, treating $A_x(v)w$ as being the same as $A_x(v,w)$.
- The norm $\|\cdot\|_{\mathbb{X}\to\mathbb{Y}}$ is the operator norm. We say that a bilinear map $A\colon \mathbb{X}_1 \times \mathbb{X}_2 \to \mathbb{Y}$ is bounded if
$$\|A\|_{\mathbb{X}\to\mathbb{Y}} := \sup\{\|A(v_1,v_2)\|_{\mathbb{Y}}\colon \|v_1\|_{\mathbb{Y}} \leqslant 1, \|v_2\|_{\mathbb{Y}} \leqslant 1\} < \infty.$$
- If we let $A\colon \mathbb{X} \to \mathbb{Y}$ be a linear map between normed spaces, there are equivalent ways to express the operator norm:
$$\begin{aligned}\|A\|_{\mathbb{X}\to\mathbb{Y}} &= \sup_{0\neq v\in\mathbb{X}} \frac{\|Av\|_{\mathbb{Y}}}{\|v\|_{\mathbb{X}}} \\ &= \sup_{\|v\|_{\mathbb{X}}=1} \frac{\|Av\|_{\mathbb{Y}}}{\|v\|_{\mathbb{X}}} = \sup_{\|v\|_{\mathbb{X}}=1} \|Av\|_{\mathbb{Y}}.\end{aligned}$$
- The limit in the definition of the second-order Fréchet derivative can be written more explicitly by expanding out the operator norm:
$$\begin{aligned}\lim_{v\to 0 \text{ in } \mathbb{X}} \frac{\|\mathrm{d}f(x+v) - \mathrm{d}f(x) - A_x v\|_{\mathbb{X}\to\mathbb{Y}}}{\|v\|_{\mathbb{X}}} &= \lim_{\substack{v\to 0\\ \text{in } \mathbb{X}}} \sup_{h\neq 0} \frac{\|\mathrm{d}f(x+v)h - \mathrm{d}f(x)h - (A_x v)h\|_{\mathbb{Y}}}{\|h\|_{\mathbb{X}}\|v\|_{\mathbb{X}}} \\ &= \lim_{\substack{v\to 0\\ \text{in } \mathbb{X}}} \sup_{\|h\|_{\mathbb{X}}=1} \frac{\|\mathrm{d}f(x+v)h - \mathrm{d}f(x)h - (A_x v)h\|_{\mathbb{Y}}}{\|v\|_{\mathbb{X}}}.\end{aligned}$$

Corollary A.1.7 Let $\mathbb{X} = \mathbb{R}^n$ with the standard Euclidean basis $\{e_1, \ldots, e_n\}$, $\mathbb{Y} = \mathbb{R}$, $U \subseteq \mathbb{R}^n$ be open, and $f\colon U \to \mathbb{R}$ be twice Fréchet differentiable at $x \in U$. Then the matrix representation of $\mathrm{d}^2 f(x)$ with respect to the standard basis of $\mathbb{R}^n$ is called the **Hessian matrix** $Hf(x)$. Its entries are the second-order partial derivatives of f at x i.e.

$$Hf(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1\,\partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_n\,\partial x_1}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1\,\partial x_n}(x) & \cdots & \frac{\partial^2 f}{\partial x_n\,\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^{n\times n}$$

in the sense that for all

$$v = \sum_{i=1}^n v_i e_i \quad \text{and} \quad w = \sum_{i=1}^n w_i e_i,$$

we have that

$$\mathrm{d}^2 f(x)(v,w) = v^\top Hf(x) w.$$

A.2 Matrix Calculus

Definition A.2.1 For a function $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$, we define the derivative of f with respect to $A \in \mathbb{R}^{m\times n}$ by:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}.$$

As a special case for a vector $\boldsymbol{\theta} = [\theta_1, \dots, \theta_n]^\top \in \mathbb{R}^n$ and $J\colon \mathbb{R}^n \to \mathbb{R}$, we define

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}.$$

Example A.2.2

- Some properties about the trace of a square matrix $\mathrm{tr}(A) = \sum_i A_{ii}$.
  - For any two matrices A, B s.t. AB is square, $\mathrm{tr}(AB) = \mathrm{tr}(BA)$. As a corollary, there's a cyclic nature to the trace $\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB)$.
  - $\mathrm{tr}(A) = \mathrm{tr}(A^\top)$
  - $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$
  - $\mathrm{tr}(aA) = a\mathrm{tr}(A)$ for a scalar a.
- Some properties of the matrix derivative:
  1. $\nabla_A \mathrm{tr}(AB) = B^\top$
  2. $\nabla_{A^\top} f(A) = (\nabla_A f(A))^\top$
  3. $\nabla_A \mathrm{tr}(ABA^\top C) = CAB + C^\top AB^\top$
  4. $\nabla_A \det(A) = \det(A) \cdot (A^{-1})^\top$

The following is used in the derivation of the normal equation:
Combine (2) and (3) to make (5):

$$\begin{aligned} \nabla_{A^\top} \mathrm{tr}(ABA^\top C) &\overset{(2)}{=} \Big(\nabla_A \mathrm{tr}(ABA^\top C)\Big)^\top \\ &\overset{(3)}{=} \Big(CAB + C^\top AB^\top\Big)^\top \\ &= B^\top A^\top C^\top + BA^\top C \end{aligned} \tag{5}$$

From regular calculus, we know that $(\mathrm{d}/\mathrm{d}x)(x^2) = 2x$.
The matrix derivative trace analogue is $\nabla_A \mathrm{tr}(AA^\top C) = CA + C^\top A$ which is similar to

$$\frac{\mathrm{d}(a^2 c)}{\mathrm{d}a} = 2ac$$

CHAPTER B

CS229 Problem Sheets

B.1 Problem Sheet 0

QUESTION 1A

Let $f(x) = \frac{1}{2}x^T Ax + b^T x$, where A is a symmetric matrix and $b \in \mathbb{R}^n$ is a vector. What is $\nabla f(x)$?

SOLUTION 1A

$$\begin{aligned} f(x+v) - f(x) &= \frac{1}{2}(x+v)^T A(x+v) + b^T(x+v) - \frac{1}{2}x^T Ax - b^T x \\ &= \frac{1}{2}x^T Av + \frac{1}{2}v^T Ax + \frac{1}{2}v^T Av + b^T v \end{aligned}$$

Conjecture that $A_x v = \frac{1}{2}\left(x^T Av + v^T Ax\right) + b^T v$. Thus, the difference quotient limit (A.1) becomes:

$$\begin{aligned} \frac{1}{2}\lim_{v\to 0 \text{ in } \mathbb{X}} \frac{|v^T Av|}{\|v\|_2} &\leqslant \frac{1}{2}\lim_{v\to 0 \text{ in } \mathbb{X}} \frac{\|v^T\|_2\|A\|_{\text{op}}\|v\|_2}{\|v\|_2} \\ &= \frac{1}{2}\lim_{v\to 0 \text{ in } \mathbb{X}} \|v^T\|_2\|A\|_{\text{op}} \\ &= 0 \end{aligned}$$

Our expression for the Fréchet derivative of f at x is:

$$\begin{aligned} \mathrm{d}f(x)v &= \frac{1}{2}\left(x^T Av + v^T Ax\right) + b^T v \\ &= \frac{1}{2}\left(x^T Av + x^T Av\right) b^T v \qquad \text{since } \mathbb{R} \ni v^T Ax = (v^T Ax)^T = x^T Av \\ &= \left(x^T A + b^T\right) v \end{aligned}$$

Therefore, $\nabla f(x) = \left(x^T A + b^T\right)^T = Ax + b$.

SOLUTION 1A (ALTERNATIVE)

Alternatively, one can resort to an explicit computation. Let $x = [x_1 \ldots x_n]^T$ and A be the matrix whose $(i,j)^{\text{th}}$ entry is a_{ij}.

$$\frac{1}{2}x^T Ax = \frac{1}{2}\sum_{i=1}^{n} x_i \sum_{j=1}^{n} a_{ij}x_j$$

Let A_k denote the k^{th} row of A.

$$\begin{aligned} \frac{\partial(\frac{1}{2}x^T Ax)}{\partial x_k} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\frac{\partial}{\partial x_k}(x_i x_j) &= \frac{1}{2}\left(\sum_{i=1}^{n} a_{ik}x_i + \sum_{j=1}^{n} a_{kj}x_j\right) \\ &= \frac{1}{2}\left(A_k x + (k^{\text{th}} \text{ column of } A)\cdot x\right) \\ &= \frac{1}{2}\left(A_k x + (A^T)_k x\right) \\ &= \frac{1}{2}(2A_k x) \\ &= A_k x \end{aligned}$$

The second equality stems from observing that $\dfrac{\partial}{\partial x_k} x_i x_j = \begin{cases} x_j, & k = i \\ x_i, & k = j \\ 0, & \text{otherwise.} \end{cases}$

Therefore, $\nabla(x^T Ax) = \begin{bmatrix} \vdots \\ A_k x \\ \vdots \end{bmatrix} = Ax.$

$$\therefore \nabla f(x) = \nabla(x^T Ax) + \nabla(b^T x) = Ax + \begin{bmatrix} \vdots \\ \frac{\partial}{\partial x_k} \sum_{i=1}^n b_i x_i \\ \vdots \end{bmatrix} = Ax + \begin{bmatrix} \vdots \\ b_k \\ \vdots \end{bmatrix} = Ax + b$$

Question 1b

Let $f(x) = g(h(x))$ where $g\colon \mathbb{R} \to \mathbb{R}$ and $h\colon \mathbb{R}^n \to \mathbb{R}$ are differentiable. What is $\nabla f(x)$?

Solution 1b

$$\nabla f(x) = \begin{bmatrix} \vdots \\ \frac{\partial (g \circ h)}{\partial x_i}(x) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ g'(h(x)) \frac{\partial h}{\partial x_i} \\ \vdots \end{bmatrix} = g'(h(x)) \begin{bmatrix} \vdots \\ \frac{\partial h}{\partial x_i} \\ \vdots \end{bmatrix} = g'(h(x)) \nabla h(x)$$

Question 1c

The Hessian of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is the $n \times n$ symmetric matrix of second order partial derivatives whose (i,j) entry is $(\nabla^2 f(x))_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x)$. Let $f(x) = x^T Ax + b^T x$. What is $\nabla^2 f(x)$?

Solution 1c

Using (**1a**), our expression for the (θ, k) entry of the Hessian is:

$$\begin{aligned} \frac{\partial^2}{\partial x_\theta \, \partial x_k} f(x) &= \frac{\partial}{\partial x_\theta} \left(\left(\sum_{i=1}^n a_{ik} x_i + \sum_{j=1}^n a_{kj} x_j \right) + b_k \right) \\ &= \frac{1}{2}(a_{\theta k} + a_{k\theta}) \\ &= a_{\theta k} \quad \text{by the symmetry of } A \end{aligned}$$

Therefore, $\nabla^2 f(x) = A$.

Question 1d

Let $f(x) = g(a^T x)$ where $g\colon \mathbb{R} \to \mathbb{R}$ is continuously differentiable and $a \in \mathbb{R}^n$. What are the gradient and Hessian of f at x?

Solution 1d

Define $h\colon R^n \to \mathbb{R}$ by $h(x) = a^T x$. From (**1a**), $\nabla h(x) = a$. Using (**1b**), we have that:

$$\begin{aligned}\nabla f(x) &= \nabla(g \circ h)(x) \\ &= g'(h(x))\nabla h(x) \\ &= g'(a^T x)a\end{aligned}$$

The (k, j) entry of the Hessian of f at x is given by:

$$\begin{aligned}\frac{\partial^2}{\partial x_k\,\partial x_j} f(x) &= \frac{\partial}{\partial x_k}\left(g'(a^T x)\frac{\partial}{\partial x_j}(a^T x)\right) \\ &= \frac{\partial}{\partial x_k}\left(g'(a^T x)\frac{\partial}{\partial x_j}\left(\sum_{i=1}^{n} a_i x_i\right)\right) \\ &= \frac{\partial}{\partial x_k}\left(g'(a^T x)a_j\right) \\ &= g''(a^T x)\frac{\partial}{\partial x_k}(a^T x)a_j \\ &= g''(a^T x)a_k a_j\end{aligned}$$

Finally, the expression for the Hessian is:

$$\nabla^2 f(x) = g''(a^T x)\begin{bmatrix} & \vdots & \\ \cdots & a_k a_j & \cdots \\ & \vdots & \end{bmatrix} = g''(a^T x)aa^T.$$

Question 2

A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is called:

- positive semidefinite, denoted $A \succeq 0$, if for all $x \in \mathbb{R}^n$, $x^T Ax \geqslant 0$,
- positive definite, denoted $A \succ 0$, if for all non-zero $x \in \mathbb{R}^n$, $x^T Ax > 0$.

Question 2a

Let $z \in \mathbb{R}^n$. Show that $A = zz^T$ is positive semidefinite.

Solution 2a

Since $A^T = (zz^T)^T = (z^T)^T z^T = zz^T = A$, A is symmetric. Now let $x \in \mathbb{R}^n$. Then, $x^T Ax = x^T zz^T x = (x^T z)(x^T z)^T = |x^T z|^2 \geqslant 0$.

Question 2b

Let $\vec{0} \neq z \in \mathbb{R}^n$ and $A = zz^T$. What is the nullspace of A and what is the rank of A?

Solution 2b

Question 2c

$m, n \in \mathbb{N}$. Let $A \in \mathbb{R}^{n\times n}$ be positive semidefinite and $B \in \mathbb{R}^{m\times n}$ be arbitrary. Is BAB^T also positive semidefinite? If so, prove it. Otherwise, give a counterexample with explicit A and B.

Solution 2c

$(BAB^T)^T = (B^T)^T A^T B^T = BA^T B^T = BAB^T$ since A is symmetric. Thus, BAB^T is symmetric. Now let $x \in \mathbb{R}^n$ and consider $x^T(BAB^T)x = (x^T B)A(B^T x) = (x^T B)A(x^T B)^T \geqslant 0$ because A is positive semidefinite and $y = x^T B \in \mathbb{R}^n$.

Question 3

The eigenvalues of a matrix $A \in \mathbb{R}^{n\times n}$ are the roots of the characteristic polynomial $p_A(\lambda) = \det(\lambda I - A)$, which may (in general) be complex. They are also defined as the values $\lambda \in \mathbb{C}$ for which there exists a vector $x \in \mathbb{C}^n$ such that $Ax = \lambda x$. We call such a pair (λ, x) an eigenvalue-eigenvector pair. For notational convenience, we define the diagonal matrix with values $\lambda_1, \dots, \lambda_n$ along the main diagonal and 0's elsewhere as:

$$\operatorname{diag}(\lambda_1, \dots, \lambda_n) = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

Question 3a

Suppose that the matrix $A \in \mathbb{R}^{n\times n}$ is diagonalisable i.e. there exists an invertible matrix $T \in \mathbb{R}^{n\times n}$ such that $A = T\Lambda T^{-1}$ where $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$. Use the notation $t^{(i)}$ for the columns of T so that $T = [t^{(1)} \cdots t^{(n)}]$ where $t^{(i)} \in \mathbb{R}^n$. Show that $At^{(i)} = \lambda_i t^{(i)}$ so that the eigenvalue-eigenvector pairs of A are $(t^{(i)}, \lambda_i)$.

Solution 3a

$$\begin{aligned} A = T\Lambda T^{-1} &\implies AT = T\Lambda \\ &\iff A[t^{(1)} \cdots t^{(n)}] = [t^{(1)} \cdots t^{(n)}]\operatorname{diag}(\lambda_1, \dots, \lambda_n) \\ &\iff [At^{(1)} \cdots At^{(n)}] = [\lambda_1 t^{(1)} \cdots \lambda_n t^{(n)}] \\ &\iff At^{(i)} = \lambda_i t^{(i)} \quad \forall i = 1, \dots, n \end{aligned}$$

Question 3b

A matrix U is orthogonal if $U^T U = I$. The spectral theorem, perhaps the most important theorem in linear algebra, states that if $A \in \mathbb{R}^{n\times n}$ is symmetric, then A is diagonalisable by a real orthogonal matrix. That is, there are a diagonal matrix $\Lambda \in \mathbb{R}^{n\times n}$ and an orthogonal matrix $U \in \mathbb{R}^{n\times n}$ such that $A = U\Lambda U^T$. Let $\lambda_i = \lambda_i(A)$ denote the i^{th} eigenvalue of A.

Let A be symmetric. Show that if $U = [u^{(1)} \cdots u^{(n)}]$ is orthogonal, where $u^{(i)} \in \mathbb{R}^n$ and $A = U\Lambda U^T$, then $u^{(i)}$ is an eigenvector of A and $Au^{(i)} = \lambda_i u^{(i)}$, where $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$.

Solution 3b

Right-multiply $A = U\Lambda U^T$ by U to get $AU = U\Lambda(U^T U)$. Since $U^T U = I$, we have that $AU = U\Lambda$ and the calculation that follows is identical to (**3a**) by replacing T with U.

Question 3c

Show that if A is positive semidefinite, then $\lambda_i(A) \geqslant 0$ for each i.

Solution 3c

A being positive semidefinite means, in particular, that A is symmetric. Since A is symmetric, it is diagonalisable by a real orthogonal matrix $U = [u^{(1)} \cdots u^{(n)}]$ and the eigenvalue-eigenvector pairs of A are of the form $(\lambda_i(A), u^{(i)})$. Since U is real orthogonal, each $u^{(i)}$ has unit length i.e. $\|u^{(i)}\|_2 = 1$.

The other property of A being positive semidefinite is that for all $x \in \mathbb{R}^n$, $x^T Ax \geqslant 0$. Let $i \in \{1, \ldots, n\}$ and $x = u^{(i)}$ in particular. Thus, for each $i = 1, \ldots, n$

$$0 \leqslant (u^{(i)})^T(Au^{(i)}) = u^{(i)}\lambda_i u^{(i)} = \lambda_i(\|u^{(i)}\|_2)^2 = \lambda_i.$$

B.2 Problem Sheet 1

Question 1a

Find the hessian H of J:

$$J(\boldsymbol{\theta}) = \frac{-1}{n}\sum_{i=1}^{n}\Big(y^{(i)}\log(h_{\boldsymbol{\theta}}(x^{(i)})) + (1-y^{(i)})\log(1-h_{\boldsymbol{\theta}}(x^{(i)}))\Big)$$

and show that for any vector z, it holds true that $z^\top H z \geqslant 0$.

Solution 1a

This identity has already been derived in the notes:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}h_{\boldsymbol{\theta}}(x^{(i)}) &= \frac{\partial}{\partial\theta_j}g(\boldsymbol{\theta}^\top x^{(i)})\\ &= h_{\boldsymbol{\theta}}(x^{(i)})\left(1-h_{\boldsymbol{\theta}}(x^{(i)})\right)x_j^{(i)}\end{aligned}$$

Using this identity, we can find an expression for the i^{th} entry of $\nabla J(\boldsymbol{\theta})$:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta}) &= \frac{-1}{n}\sum_{i=1}^{n}\left(\frac{y^{(i)}}{h_{\boldsymbol{\theta}}(x^{(i)})}\frac{\partial}{\partial\theta_j}h_{\boldsymbol{\theta}}(x^{(i)}) + \frac{1-y^{(i)}}{1-h_{\boldsymbol{\theta}}(x^{(i)})}\frac{\partial}{\partial\theta_j}(1-h_{\boldsymbol{\theta}}(x^{(i)}))\right)\\ &= \frac{-1}{n}\sum_{i=1}^{n}(y^{(i)}-h_{\boldsymbol{\theta}}(x^{(i)}))x_j^{(i)}\end{aligned}$$

Therefore, the $(j,k)^{\text{th}}$ entry of $H = \nabla^2 J$ is:

$$\frac{\partial^2}{\partial\theta_j\,\partial\theta_k}J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}h_{\boldsymbol{\theta}}(x^{(i)})(1-h_{\boldsymbol{\theta}}(x^{(i)}))x_j^{(i)}x_k^{(i)}$$

The expression for $z^\top H z$ is:

$$\begin{aligned}z^\top H z &= \sum_{j=0}^{p}\sum_{k=0}^{p} z_j H_{jk} z_k\\ &= \sum_{j=0}^{p}\sum_{k=0}^{p} z_j \frac{1}{n}\sum_{i=1}^{n}h_{\boldsymbol{\theta}}(x^{(i)})(1-h_{\boldsymbol{\theta}}(x^{(i)}))x_j^{(i)}x_k^{(i)}z_k\\ &= \frac{1}{n}\sum_{i=1}^{n}\underbrace{\left(\sum_{j=0}^{p}\sum_{k=0}^{p}z_j x_j^{(i)}x_k^{(i)}z_k\right)}_{(x^\top z)^2\geqslant 0}\underbrace{\Big(h_{\boldsymbol{\theta}}(x^{(i)})(1-h_{\boldsymbol{\theta}}(x^{(i)}))\Big)}_{\geqslant 0 \text{ since } g(\dots)\in[0,1]} \geqslant 0\end{aligned}$$

Therefore, $H \succeq 0$ i.e. the Hessian of J is positive semi-definite. Therefore, J is convex i.e. the local minimum is the global minimum.

CHAPTER C

Calculations

C.1 GDA Maximum Likelihood Estimates

Consolidate both parameters μ_0 and μ_1 into a single parameter that takes on either value depending on the value of $y^{(i)}$:

$$\mu_{y^{(i)}} = \mathbb{1}\left(y^{(i)} = 0\right)\mu_0 + \mathbb{1}\left(y^{(i)} = 1\right)\mu_1.$$

$$\begin{aligned}
\ell(\varphi, \mu_{y^{(i)}}, \Sigma) &= \log\left(\prod_{i=1}^{n} \mathbb{P}_{\mathbf{X}|Y}\left(x^{(i)}|y^{(i)}; \mu_{y^{(i)}}, \Sigma\right) \cdot \mathbb{P}(Y)y^{(i)}; \varphi\right) \\
&= n\log\left(\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\right) - \frac{1}{2}\sum_{i=1}^{n}(x - \mu_{y^{(i)}})^{\top}\Sigma^{-1}(x - \mu_{y^{(i)}}) \\
&\quad + \sum_{i=1}^{n}\left(y^{(i)}\log(\varphi) + (1 - y^{(i)})\log(1 - \varphi)\right)
\end{aligned}$$

- $\widehat{\varphi}$

$$\begin{aligned}
\frac{\partial}{\partial\varphi}\ell(\varphi, \mu_{y^{(i)}}, \Sigma) &= \frac{1}{\varphi}\sum_{i=1}^{n} y^{(i)} + \frac{-1}{1-\varphi}\sum_{i=1}^{n}(1 - y^{(i)}) \\
&= \frac{1}{1-\varphi}\left(n - \frac{1}{\varphi}\sum_{i=1}^{n} y^{(i)}\right)
\end{aligned}$$

$$\frac{\partial\ell}{\partial\varphi} = 0 \implies \widehat{\varphi} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} = \overline{y}$$

- $\widehat{\mu_i}$

  Do I take gradients or...

$$\begin{aligned}
\frac{\partial}{\partial\mu_i}\ell(\varphi, \mu_{y^{(i)}}, \Sigma) &= \ldots \\
&= \ldots
\end{aligned}$$

Bibliography

[1] Khallil Ebrahim Benyattou. *Conditional Exkebtations.* 2026.

[2] Mark John Schervish. *Theory of Statistics.* Springer Science & Business Media, 2012.

[3] Fumio Hayashi. *Econometrics.* Princeton University Press, 2000.

[4] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, volume 55. Spartan Books Washinton, DC, 1962.