Machine learning in science and society

From automated science to beneficial artificial intelligence

Christos Dimitrakakis

November 8, 2022

Preface

This book is a technical introduction to important aspects of machine learning in science and society. It covers the most fundamental concepts in privacy, fairness, causality, reproducibility and experiment design. While I have tried to make the book as rigorous as possible, in the interest of brevity some technical details are omitted.

I whole-heartedly recommend the book "The Ethical Algorithm" as a non-technical companion to this book.

Contents

Chapter 1

Introduction

What this this book about?

1.1 Introduction to machine learning

Problems in machine learning are similar to problems in science. Scientists must plan experiments intelligently and collect data. The must be able to use the data to verify a different hypothesis. More generally, they must be able to make decisions under uncertainty (Without uncertainty, there would be no need to gather more data). Similar problems appear in more mundane tasks, like learning to drive a car.

For that reason, science is a very natural application area for machine learning. We can model the effects of climate change and how to mitigate it; discover structure in social networks; map the existence of dark matter in the universe by intelligently shifting through weak gravitational lens data, and not only study the mechanisms of protein folding, but discover methods to synthesize new drugs.

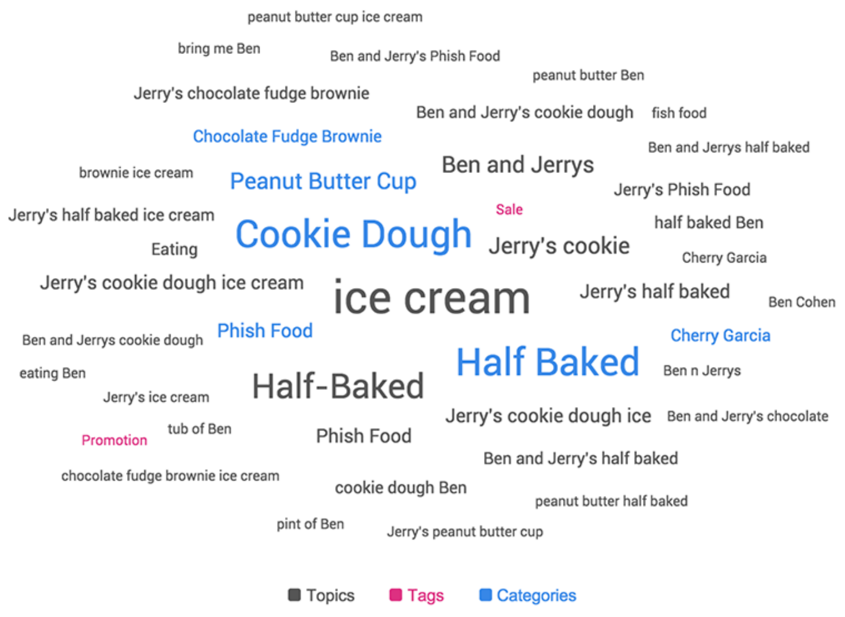
We must be careful, however. In many cases we need to be able to interpret what our model tells us. We also must make sure that the any results we obtain are reproducible.

While machine learning models in science are typically carefully handcrafted by scientists and experts in machine learning and statistics, this is not typically the case in everyday applications. Nevertheless, well-known or home-grown machine learning models are being deployed across the application spectrum. This involve home assistants that try and get you want, web advertising, which tries to find new things for you to want, lending, which tries to optimally lend you money so that you buy what you didn't need before. We also have autonomous vehicles, which take you were you want to go, and ridesharing services, which do the same thing, but use humans instead. Finally, there are many applications in public policy, such as crime prevention, justice, and disease control which use machine learning. In all those cases, we have to worry about a great many things that are outside the scope of the machine learning problems itself. These are (a) privacy: you don't want your data used in ways that you have not consented to (b) fairness: you don't want minorities to be disadvantaged and (c) safety: you don't want your car to crash.

1.1.1 Data analysis, learning and planning

To make the above more concrete, let's have a look at a number of problems in machine learning. These involve learning from and analysing data, including inferring decision rules, and constructing complex plans using the evidence gleaned from the data. Machine learning problems are commonly separated in three different types: supervised, unsupervised and reinforcement learning. Typical supervised learning problems include classification and regression, while unsupervised problems include compression, clustering and topic modelling. Reinforcement learning, on the other hand, is concerned with artificially intelligent agents more generally, with examples including game playing and adaptive control. Their main differences are two. Firstly, the *type* of feedback we have about learning performance. Secondly, and perhaps more importantly, whether or not the problem involves *active data collection*. In this course, we will try and take a global view of these problems in the context of decision theory.

Can machines learn from data?



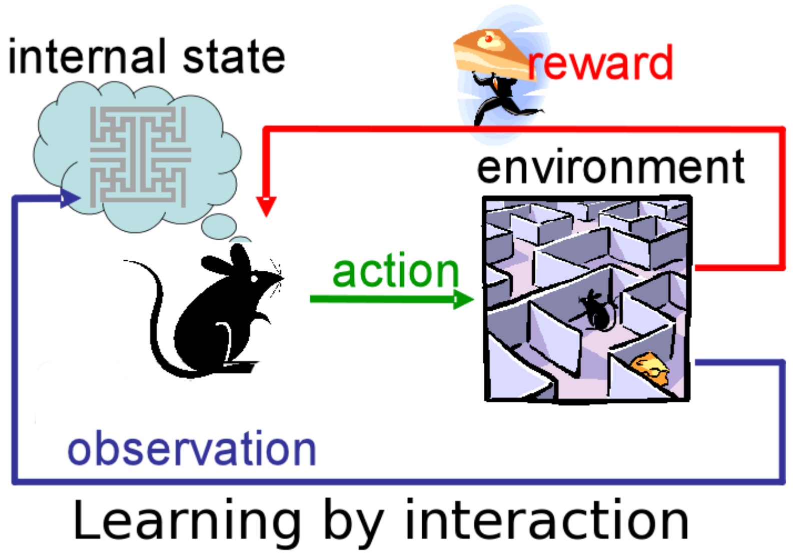
An unsupervised learning problem: topic modelling

A supervised learning problem: object recognition

You can use machine learning just to analyse, or find structure in the data. This is generally called unsupervised learning. One such example is topic modelling, where you let the algorithm find topics from a corpus of text. These days machines are used to learn from in many applications. These include speech recognition, facial authentication, weather prediction, etc. In general, in these problems we are given a *labelled* dataset with, say, example images from each class. Unfortunately this does not scale very well, because obtaining labels is expensive.

This is partially how science works, because what we need to do is to find a general rule of nature from data. Starting from some hypothesis and some data, we reach a conclusion. However, many times we may need to actively experiment to obtain more data, perhaps because we found that our model is wrong.

Can machines learn from their mistakes?



So, what happens when we make a mistake? Can we somehow recognise it? Humans and other animals can actually learn from their mistakes. Consider the proverbial rat in the maze. At some intervals, the experimenter places some cheese in there, and the rat must do a series of actions to obtain it, such as navigating the maze and pulling some levers. It doesn't know how to get to the cheese easily, but it slowly learns the layout of the maze through observation, and in the end, through trial-and-error it is able to get to the cheese very efficiently.

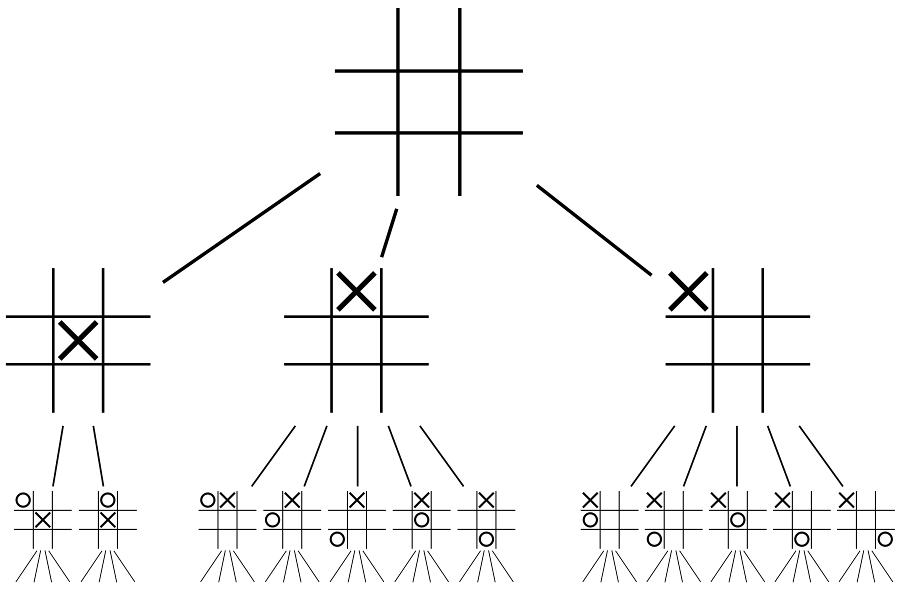
We can formalise this as a reinforcement learning problem, where the rat takes a series of actions; at each step it also obtains a reward, let's say equal to 0 when it has no cheese, and 1 when it eats cheese. Then we can declare that the rat's utility is the sum of all rewards over time, i.e. the total amount of cheese it can eat before it dies. The rat needs to explore the environment in order to be able to get to the cheese.

An example in robotics is trying to teach a robot to flip pancakes. One easy thing we can try is to show the robot how to do it, and then let it just copy the demonstrated movement. However, this doesn't work! The robot needs to explore variations of the movement, until it manages to successfully flip pancakes. Again, we can formulate this as a reinforcement learning problem, with a reward that is high whenever the pancake's position is flipped, and on the pan; and low everywhere else. Then the robot can learn to perform this behaviour through trial and error. It's important to note that in this example, merely demonstration is not enough. Neither is reinforcement learning enough. The same thing is true for the recent success of AlphaGo in beating a master human: apart from planning, they used both demonstration data and self-play, so that it could learn through trial and error.

Can machines make complex plans?

I suppose the first question is whether machines can plan ahead. Indeed, even for large problems, such as Go, machines can now perform at least as well as top-rated humans. How is this achieved?

Machines can make complex plans!



The basic construction is the planning tree. This is an enumeration of all possible future events. If a complete enumeration is impossible, a partial tree is constructed. However this requires evaluating non-terminal game positions. In the old times, this was done with heuristics, but now this is data-driven, both through the use of expert databases, and through self-play and reinforcement learning.

1.1.2 Experiment design

An example that typifies trial and error learning are bandit problems. Imagine that you are in a Casino and you wish to maximise the amount of money you make during the night. There are a lot of machines to play. If you knew which one was the best, then you'd just play it all night long. However, you must also spend time trying out different machines, in order to get an estimate of how much money each one gives out. The trade off between trying out different machines and playing the one you currently think is best is called the exploration-exploitation trade-off and it appears in many problems of experiment design for science.

Let's say we want to build a robot scientist and tell it to discover a cure for cancer. What does the scientist do and how can the robot replicate it??

Simplifying the problem a bit, consider that you have a large number of drug candidates for cancer and you wish to discover those that are active against it. The ideas is that you select some of them, then screen them, to sort them into active and inactive. However, there are too many drugs to screen, so the process is interactive. At each cycle, we select some drugs to screen, classify them, and then use this information to select more drugs to screen. This cycle, consequently has two parts: 1. Selecting some drugs given our current knowledge. 2. Updating our knowledge given new evidence.

Drawing conclusions from results

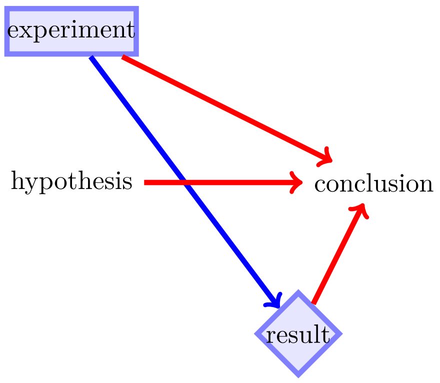
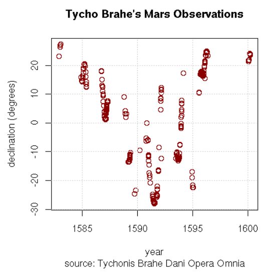


Figure 1.1: Dependence diagram between selection of an experiment, formulation of a hypothesis, and drawing of a conclusion. The result depends only on the experiment. However, the conclusion depends on the experiment, hypothesis and the obtained result. The red lines indicate computational dependencies, while the blue lines indicate physical dependencies.

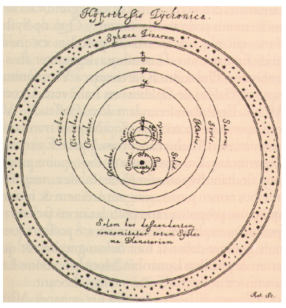
In general, we would like to have some method which can draw conclusions from results. This involves starting with a hypothesis, performing an experiment to verify or refute it, obtain some experimental result; and then concluding for or against the hypothesis. Here the arrows show dependencies between these variables. So what do we mean by "hypothesis" in this case?

1.1.3 Bayesian inference.

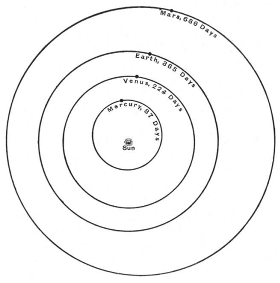
Let's take the example of planetary orbits. Here Tycho famously spent 20 years experimentally measuring the location of Mars. He had a hypothesis: that planetary orbits were circular, but he didn't know which were the right orbits. When he tried to fit his data to this hypothesis, he concluded a specific circular orbit for Mars ...around Earth.



(a) Tycho's Measurements



(b) Tycho's conclusion

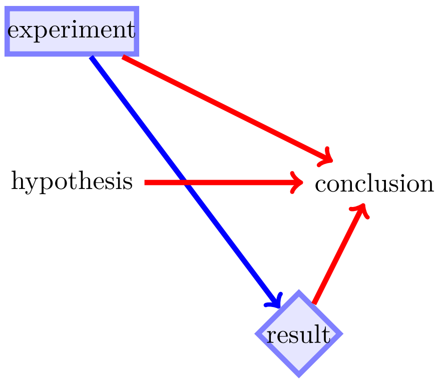


(c) Kepler's conclusion

Figure 1.2: How given the same data, one can reach different conclusion depending on one's modelling assumptions.

Kepler had a more general hypothesis: that orbits could be circular or elliptic, and he actually accepted that the planets orbited the sun. This led him to the broadly correct model of all planets being in elliptical orbits around the sun. However, the actual verification that all things do not revolve around earth, requires different experiments.

Later on, Gauss collected even more experimental data to calculate the orbit of Ceres. He



did this using one of the first formal statistical methods; this allowed him to avoid cheating (like Kepler did, to accentuate his finding that orbits were elliptical).

It is quite easy to draw the wrong conclusions from applying machine learning / statistics to your data. For example, it was fashionable to perform fMRI studies in humans to see whether some neurons have a particular functional role. There were even articles saying that "we found the neurons encoding for Angelina Jolie". So some scientists tried to replicate those results. They took a dead salmon, and put it an fMRI scanner. They checked its brain activity when it was shown images of happy or sad people. Perhaps surprisingly, they found an area of the brain that was correlated with the pictures - so it seemed, as though the dead salmon could distinguish photos of happy people from sad ones. However, this was all due to a misapplication of statistics. In this course, we will try and teach you to avoid such mistakes.

A simple simulation study

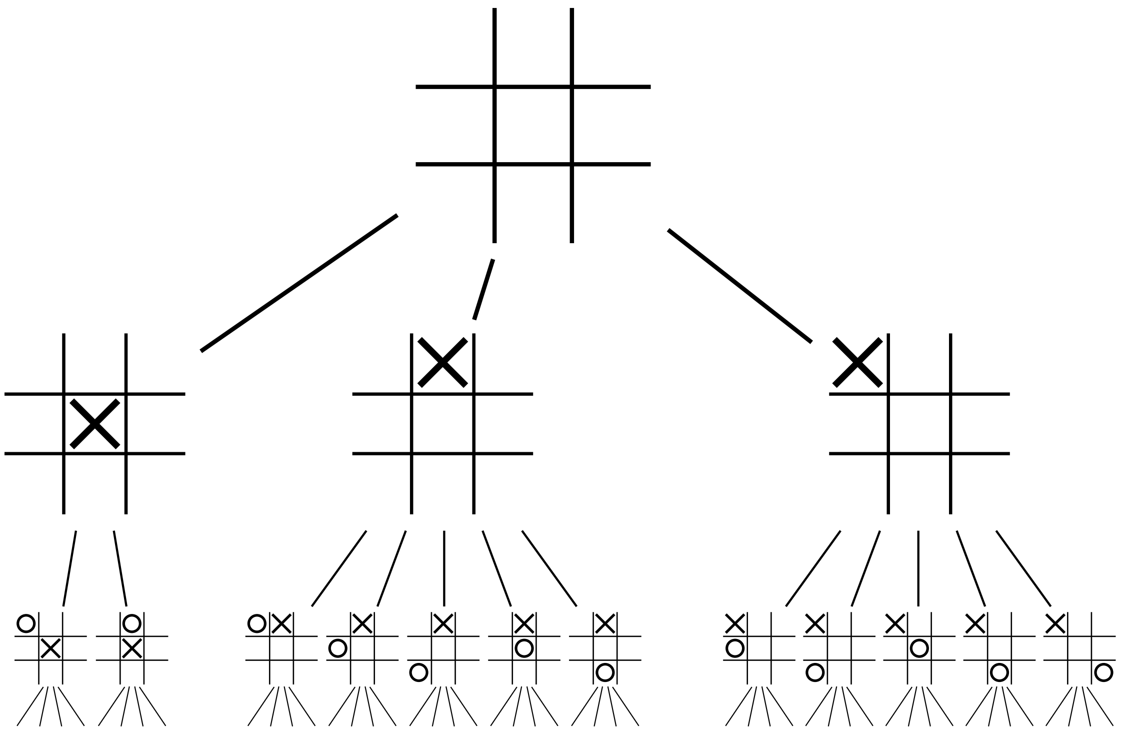
Sometimes we want to use a simple simulation study to understand how well our methods work. The following code is an example of how to do this. Here we are doing a simplified fMRI analysis, but the general idea is not significhey, antly different from what people actually do in the field. `src/reproducibility/mri_analysis.ipynb`

Planning future experiments

So far we have focused only on the problem of data analysis. However, we also need to think about the problem of planning for experiments. This is called *experiment design*. Experiment designs are usually fixed. In that case, we can use assumptions about the data and how much accuracy we need to design the experiment. However, it is also possible to have an adaptive experiment design where our future experiments depend on our current conclusions.

In general, optimal experiment design is indeed difficult, especially in setting such as drug discovery where the number of experiments is huge. However, conceptually, there is a simple and elegant solution to this problem.

Planning experiments is like Tic-Tac-Toe



The basic idea is to think of experiment design as a game between the scientist and Nature. At every step, the scientist plays an X to denote an experiment. Then Nature responds with an Observation. The main difference from a game is that Nature is (probably) not adversarial. We can also generalise this idea to problems in robotics, etc.

These kinds of techniques, coming from the reinforcement learning literature have been successfully used at the university of Manchester to create a robot, called Eve, that recently (re)-discovered a malaria drug.

1.1.4 Book overview

Machine learning in practice

Avoiding pitfalls

- Choosing hypotheses.
- Correctly interpreting conclusions.
- Using a good testing methodology.

Machine learning in society

- Privacy — Credit risk.
- Fairness — Job market.

- Safety — Medicine.

One of the things we want to do in this course is teach you to avoid common pitfalls.

Now I want to get into a different track. So far everything has been about pure research, but now machine learning is pervasive: Our phones, cars, watches, bathrooms, kettles are connected to the internet and send a continuous stream of data to companies. In addition, many companies and government actors use machine learning algorithms to make or support decisions. This creates a number of problems in privacy, fairness and safety.

The view from statistics

While in machine learning people generally discuss mainly supervised, unsupervised or reinforcement learning, these are actually three rather broad, but still limited categories of problems. There are many other problems, such as semi-supervised learning, active learning, imitation/apprenticeship learning, inverse reinforcement learning, preference elicitation, adaptive control, and possibly many others to come. The statistical view is that there basically three types of problems:

Inference

Given what we know, what can we say about how the world works, the current state of the world or events in the past?

Prediction.

Can we predict specific evens in the future? This frequently is done through some type of inference.

Decision making.

Given what we know, and what we want to achieve, what is the best decision we can make? This typically requires some ability to predict the effect of our actions.

We will encounter many specific inference, prediction and decision making tasks during this course.

Course structure

Module structure

- *Activity*-based, hands-on.
- Mini-lectures with short exercises in each class.
- Technical tutorials and labs in alternate week.

Modules

Three mini-projects.

- Simple decision problems: Credit risk.

- Sequential problems: Medical diagnostics and treatment.

Technical topics

The book covers a number of technical topics. Sections marked with an asterisk (*) may be skipped safely upon a first reading without compromising understanding of the remaining material.

Machine learning problems

- Unsupervised learning. Loosely speaking, this is simply the problem of estimating some structure from data. In statistical terms, it is usually the problem of estimating some joint distribution of random variables under some model assumptions. Problems in unsupervised learning include clustering, anomaly detection, compression.
- Supervised learning. In this setting data can be split in two groups of variables. One group that is always available, and another group that must be predicted. A special case of the problem is when we wish to estimate some function $f : \mathcal{X} \to \mathcal{Y}$ from data. Classical problems in this setting are classification and regression.
- Reinforcement learning. This is a very general sequential decision problem, where an agent must learn how to behave optimally in an unknown environment only by limited feedback and reinforcement. The standard setting involves the agent trying to maximise its (expected) cumulative reward over time.

Algorithms and models

- Bayesian inference and graphical models.
- Stochastic optimisation and neural networks.
- Backwards induction and Markov decision processes.

Further reading

- Bennett et al.[3] describe how the usual uncorrected analysis of fMRI data leads to the conclusion that the dead salmon can reason about human images.
- Bennett et al.[2] discuss how to perform analyses of medical images in a principled way. They also introduce the use of simulations in order to test how well a particular method is going to perform.

Resources

- Online QA platform: `https://piazza.com/class/jufgabrw4d57nh`
- Course code and notes: `https://github.com/olethrosdc/ml-society-science`
- Book `https://github.com/olethrosdc/ml-society-science/notes.pdf`

Chapter 2

Privacy

One interpretation of privacy is simply the ability to maintain a personal secret. Is it possible to maintain perfect secrecy? Can the thoughts in our head be perfectly safe, or can they be revealed indirectly through our actions?

While we certainly *can* securely store and transmit data using cryptography, the problems we will discuss in this chapter are *not* solvable solely through cryptography. We are interested in scenarios where we must make a public decision or release some summary statistics that depend on private data, in such a way as to minimise harm to the individuals contributing their data.

If we never have to reveal any of our computations, cryptography is what is needed. For example, through homomorphic computation, an untrusted party can even perform some computations on encrypted data, returning an encrypted result to us, while learning nothing about the original secret. As long as the data, and the results of any computation on it, are kept under lock and key, our personal information cannot be revealed.

However, sometimes we must make public decisions or release public information based on this data. Then it can be revealed indirectly. For example, you can trust your doctor to maintain confidentiality, but when you go to the pharmacy to get the prescribed medicine, somebody can infer the medical condition you suffer from.

It is even possible to learn personal information from aggregate statistics. Let us say your doctor publishes a list of cases of different diseases every week, together with some other information such as the approximate patient age. Even though your own data is mixed with that of all other patients, it is possible to infer your diagnosis, especially with some additional side-information: If somebody knows you were the only person in that age group visiting the doctor that week, they will learn your diagnosis.

From that point of view, it is not the data itself, but the complete process of data collection, treatment and public release that can be characterised as private or non-private. For that reason, we will emphasise an *algorithmic* view of privacy: participants entrusts their data to an algorithm, which produces a useful output in return. The algorithmic process is typically not fully automated, as it also depends on some human input: One example is a medical study examining different treatments for a disease. While humans select and administer the treatments, they will typically rely on a randomised strategy for assigning treatments to individuals, and use a statistical method to report their results. Given this mixture of ad-hoc decisions and formal algorithmic methods, is it possible to guarantee privacy in any sense? What kind of guarantees can we make?

Generally speaking, an algorithm has good privacy properties, if the amount of information that can be revealed through the algorithm's output about any individual contributing data

ALL I WANT IS A SOCIAL NETWORK SMART ENOUGH TO PUT A FOOD ITEM INTO MY MOUTH JUST AS I REALIZE THAT I WANT IT, BUT WHICH ALSO DOESN'T COLLECT ANY PERSONAL DATA! IS THAT TOO MUCH TO ASK?

Just because they're the problem, doesn't mean we aren't.

is bounded. In particular, we are interested in how much an adversary can learn about any individual's input to the algorithm from the algorithm's output.

This does not preclude learning general facts about individuals from the output. For example, a study about the use of steroids in sports may show that 90% of sprinters with times under 10 seconds are using steroids, while only 50% of slower sprinters do so. Any sprinter with a time under 10 seconds is thus suspected of using steroids by association. However, it does not matter if their data has been used in the study. The publication of the result does not impact their privacy, but the amount of harm it does to them does not depend in their participation: the same statistical result would have been obtained with or without them.

In this chapter, we will look at two formal concepts of privacy protection: k-anonymity and *differential privacy*. The first is a simple method for anonymising databases. However, it provides only limited resistance to identification. The latter is a more general concept, which provides full information-theoretic protection to individuals. A major problem with any privacy definition and method, however is correct interpretation of the privacy concept used, and correct implementation of the algorithm used.

2.1 Things we do with data

How do we use data in the first place? Sometimes we simply publish the data, perhaps after some initial processing. Publication of datasets is useful for researchers that want to do perform further analysis on the data. Usually though, we collect data in order to calculate specific statistics. For example the census collects data about the number of people in different households, wages, etc, and then publishes tables detailing the average age and number of people per household in different areas of the country. This demographic information is useful for policy makers, urban planning to organise voting centers and the distribution of police and fire stations, as well as other public services.

Fundamentally, privacy in statistics is an *issue of trust*. The analyst, be it a human, or an automated service, will use your data to make decisions. You must also decide who to trust and how much. Do we trust the data analyst? How much privacy are we willing to sacrifice to the analyst? How much to the public at large? What you want out of the service. Is the service important enough to sacrifice significant amounts of privacy? What is an acceptable trade-off between utility and privacy? These are difficult questions and are hard to quantify, hence we assume that we have already decided how to answer them, and we simply want to find an appropriate methodology for achieving a good result.

Consider a researcher wishing to collect data for a statistical analysis. If the analysis is eventually published,[1] this creates two possible scenarios for that may lead to privacy violations.

- Publication of “anonymised” data. Sometimes we may collect data in order to publish the dataset itself for other researchers to use. This is common practice in machine learning, with image classification datasets being a good example. However, there is always some privacy risk even if identifying information such as names are removed.

- Public data analysis. In this setting, we only publish summary statistics or models about the data. A prime example is a national census analysis, which may provide detailed demographic information for all towns and regions in a nation. Although only aggregate data is published, it is theoretically possible to infer personal information. For that reason, the US Census Bureau conducted its analysis in 2020 using differential private algorithms.

⚠ **Cryptography is not enough.**

Cryptography provides secure communication and computation, authentication and verification. These are used to establish secure channels with somebody that we trust. However, the privacy violations we are concerned with relate to *publicly released* outputs of algorithms. It is not important whether or not all the data and computation are encrypted: as long as the algorithm generates a public output, it is theoretically possible for somebody to learn something about the algorithm’s input.

2.2 Statistical disclosure

“Data is everywhere”, said the statistician, “data, data!”.

In the past, statistical analysis was performed with laboriously collected and annotated datasets. Even as recently as in the early 21st century, databases for machine learning were limited to a few thousand entries at most. At the time of writing, not only has the size of datasets become extremely large, but the sources of data are much more diverse. Data are collected and commercialised whenever we visit a website and even as we walk around with our phone. To a limited extent, there is a tradeoff between what we can get out of a service and what we pay into it. Many free services such as navigation software rely on collecting user data to perform better: If you can tell that there is a traffic jam in Central Avenue, you can after all try and take another route. As long as informed consent exists, use of private data is generally regarded as unproblematic.[2]

However, even apparently benign data collected with appropriate consent can lead to serious and unexpected privacy violations. There are three famous examples of this: Firstly, the identification of people in supposedly anonymous health data in the 1990s in the state of Massachussets, which we will go over in detail in this chapter. Secondly, the identifications of users through anonymised movie ratings in the Netflix dataset. Finally, the ability to discover if any given individual’s data is contained in a pooled genomic study.

Anonymisation

[1] If somebody knows that the analysis is being conducted, however, they could still learn something private from the fact that the analysis has *not* been published.

[2] This is actually underscored by the GDPR legislation, which focuses on consent and data use methods.

Data is collected for many reasons. Any typical service you might want to use will require a minimal amount of data. For example, a dating service will at a minimum require your age and location, as shown in the example below.

Birthday	Name	Height	Weight	Age	Postcode	Profession
06/07	Li Pu	190	80	60-70	1001	Politician
06/14	Sara Lee	185	110	70+	1001	Rentier
01/01	A. B. Student	170	70	40-60	6732	Time Traveller

Example 1 (Typical relational database in a dating website.). If somebody is a user in a dating website, you expect them to give some minimal personal information to be stored in the site's database. This might include their birthday, location and profession, for example.

When you submit your data to a service, you expect it to be used responsibly. For the dating service, you expect it to use the information to find good matches, based on your preferences and location. Whenever somebody uses the service, they obtain some information about you, at least indirectly. For example, if they make a query for similar singles in their neighbourhood, and they see your profile, then they have learned that you live nearby.

If we wish to publish a database, frequently we need to protect the identities of the people involved. A simple idea is to erase directly identifying information. However, this does not really work most of the time, especially since attackers can have side-information that can reveal the identities of individuals in the original data, when combined with the published dataset.

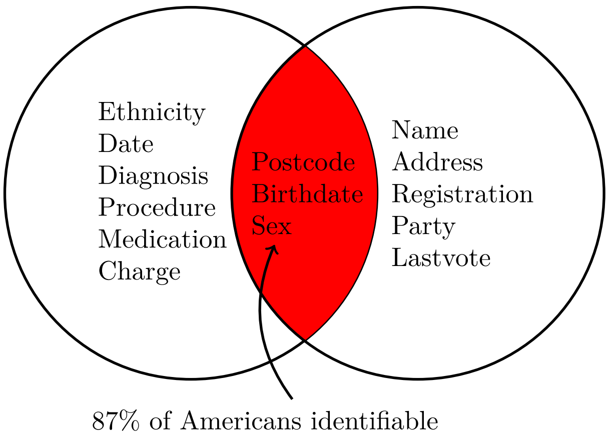
The simple act of hiding or using random identifiers is called anonymisation. However this is generally insufficient as other identifying information may be used to re-identify individuals in the data. In particular, even if somebody is unable to infer the information of any individual from the published, anonymised, dataset, they may be able to do so via some side-information. All that is needed is another dataset with some columns in common with the dataset they want to target.

Record linkage

As an example, in the 1990s a the governor of Massachussets, decided to publish anonymised information about the health records of individual state employs. They were careful to hide all obviously identifying information such as their name, and thus claimed that there are no potential privacy violations. However, they left in some information that they thought would be useful for researchers: the postcode, the birthdate, and the sex of each individual.

This allowed a PhD student, McSweeney, to perform the following linkage attack. She order a floppy disk containing database of voting records in the state. This contained names and addresses, as well as the postcode, birth date and sex of every voter. In that way, she was able to cross-reference this data, and so obtain the identities of individuals in that database. The first record she obtained was that of the governor himself, Bill Weld. Later, she estimated that approximately 87% of Americans are uniquely identifiable through those three attributes.

Clearly, anonymisation is not enough. So, is there a way to formally guarantee privacy? In the next section we will go over the solution of McSweeny, which provides us with an algorithmic definition of anonymity. While this provides some degree of protection against certain linkage attacks, it is sadly insufficient to protect privacy in general. Later, we will introduce the notion of differential privacy, which protects individuals against statistical disclosure in an information-theoretic manner.



(a) Voter registration

(b) Bill Weld

Figure 2.1: Linkage attack on a anonymised patient information database. A misguided attempt to release medical information as public data (left circle in figure (a)) by then-governor Bill Weld resulted in a simple linkage attack. This was done by accessing a party registration database (right circle in figure (b)) which contained three common fields with the medical database.

2.3 Algorithmic privacy.

The above discussion should help emphasise that *a dataset* cannot be characterised as private or non-private. We should actually start from the observation that *any data contributed by an individual* should be considered private, because it could be combined with other datasets to obtain sensitive personal information.

For that reason, we wish to ensure that, whenever individuals contribute data for processing, the result of the processing cannot be used to reveal whatever information they had contributed. Thus, the idea of privacy only applies on *algorithms* that are used on personal data. Generally speaking, we wish for algorithms to both be useful and have good privacy properties. Unfortunately, as we shall see in the sequel, there is a distinct trade-off between privacy and utility. However, let us specify more precisely what we mean by an algorithm.

What is an algorithm?

Any functional process is an algorithm. We typically identify inputs with observations or features $\mathcal{X}$, and outputs with actions $\mathcal{A}$ by the algorithm: i.e. we view the algorithm as taking actions that have an observable effect on its external environment. We generally consider *stochastic* algorithms.

Definition 2.3.1 (Stochastic algorithm π)**.** A stochastic algorithm π with input domain $\mathcal{X}$ and output domain $\mathcal{A}$ is a mapping $\pi : \mathcal{X} \to \boldsymbol{\Delta}(\mathcal{A})$ from observations $\mathcal{X}$ to distributions over outputs $\boldsymbol{\Delta}(\mathcal{A})$. When $\mathcal{A}$ is finite, we will write $\pi(a \mid x)$ to denote the conditional probability that the algorithm outputs a given input x.

The general case.

It is best to think of $\pi(\cdot|x)$ as a probability measure over $\mathcal{A}$. Then we write

$$\pi(A \mid x) \triangleq \mathbb{P}_\pi(a \in A \mid x), \qquad A \subset \mathcal{A},$$

for the conditional probability that algorithm's output is in some set $A \subset \mathcal{A}$ given input x. This allows us to treat the case when $\mathcal{A}$ is continuous or discrete with the same notation.

Algorithmic randomness.

We can construct a random algorithm through access to a random coin ω taking values in Ω. We can then define the output through a deterministic function $a_\pi(\omega, x)$. Since ω is random, the output of the function is also random. If P is the probability distribution of ω, then:

$$\pi(A \mid x) = P(\{\omega \in \Omega \mid a_\pi(\omega, x) \in A\}),$$

i.e. the probability that the algorithm's output is in A is equal to the measure of the values of ω for which the $a_\pi(\omega, x) \in A$.

What is private?

2.4 k-anonymity

k-anonymity

(a) Samarati

(b) Sweeney

The concept of k-anonymity was introduced by Samarati and Sweeney[17] and provides good guarantees against inferring personal information from a single database. This requires the analyst to first determine the variables of interest, and then determine which variables could be potentially used to identify somebody in the database. *It's the analyst's job to define quasi-identifiers.*

Definition 2.4.1 (k-anonymity)**.** A database provides k-anonymity if for every person in the database is indistinguishable from $k-1$ persons with respect to *quasi-identifiers*.

This hope is that, if the database satisfies k-anonymity it can be safely released, without revealing any private information directly. Let us now walk through an extended example.

k-anonymity example
In particular, let us say that the analyst simply wants to calculate some statistics about how different professions correlate with age, weight, height and where people live. Some areas of the country might produce more politicians, for example. And taller people may be more successful in politics. The initial data collected might look like the table below. It was obvious to the analyst, that even if he did remove all the names, somebody knowing where A. B. Student lived and saw the table would have little trouble recognising them.

Birthday	Name	Height	Weight	Age	Postcode	Profession
06/07	Li Pu	190	80	65	1001	Politician
06/14	Sara Lee	185	110	67	1001	Rentier
06/12	Nikos Karavas	180	82	72+	1243	Politician
01/01	A. B. Student	170	70	52	6732	Time Traveller
05/08	Li Yang	175	72	35	6910	Politician

Table 2.1: 1-anonymity.

After thinking about it for a bit, the analyst decides to remove the birthday and name, and broadly categorise people according to their height in increments of 10cm, the weight in increments of 20cm, and keep just the first digit of the postcode. Now that looked much more reasonable. Still, somebody that knows that Li Yang and A. B. Student are in the table, as well as their postcodes, and they also know that Li Yang is a politician, can infer that A. B. Student is a Time Traveller.

Height	Weight	Age	Postcode	Profession
180-190	80+	60+	1*	Politician
180-190	80+	60+	1*	Rentier
180-190	80+	60+	1*	Politician
170-180	60-80	20-60	6*	Time Traveller
170-180	60-80	20-60	6*	Politician

Table 2.2: 2-anonymity: the database can be partitioned in sets of at least 2 records

However, with enough information, somebody may still be able to infer something about the individuals. In the example above, it remains true that if somebody knows that both A. B. Student and Li Yang are in the database, as well as their postcodes, as well as that Li Yang is a politician, they can infer A. B. Student's profession. Fortunately, there is a way to protect individual information from adversaries with arbitrary side-information. This is given by differential privacy.

2.5 Differential privacy

This section introduces one of the main tools for giving formal guarantees about the privacy of any algorithm ran on a dataset, *differential privacy*. This will provide individual-level privacy, in the sense that an algorithm that is differentially private guarantees that no adversary can significantly increase their knowledge about any particular individual by observing the algorithm's output. This is independent of the adversary's existing knowledge, or computational power.

While k-anonymity can protect against specific re-identification attacks when used with care, it says little about what to do when the adversary has a lot of knowledge. For example, if the adversary knows the data of everybody that has participated in the database, it is trivial for them to infer what our own data is. For some particularly sensitive datasets, we may want for the adversary to be unable to tell whether or not your data was part of the base. Differential privacy offers protection against adversaries with unlimited side-information or computational power. Informally, an algorithmic computation is differentially-private if an adversary cannot distinguish two "neighbouring" database based on the result of the computation. Informally, two databases are neighbours when they are identical apart from the data of one person. A differentially private algorithm, because of its randomness, makes it impossible for somebody to tell from the algorithm's output whether any specific individual's data was in the database.

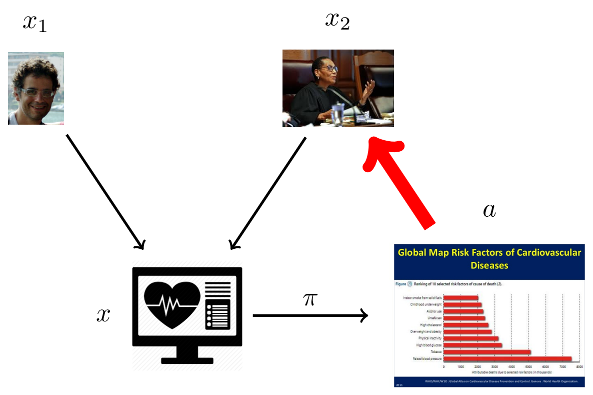


Figure 2.3: If two people contribute their data $x = (x_1, x_2)$ to a medical database, and an algorithm π computes some public output a from x, then it should be hard infer anything about the data from the public output.

Consider the example given in Figure 2.3, where two people contribute their data to a medical database. The i-th individual contributes data x_i, and the complete dataset is $x = (x_1, x_2)$. The algorithm π defines a distribution over the set of possible outputs $a \in \mathcal{A}$, with $\pi(a|x)$ being the probability that the algorithm outputs a if the data is x. If the algorith was deterministic, then it might be possible for an adversary to invert the computation and obtain x from a. But even if that is not possible, maybe they can learn *something* about the data from the output. In the section below, we will formalise this notion.

Privacy desiderata

Consider a scenario where n persons give their data $x_1, \ldots, x_n$ to an analyst. This analyst then performs some calculation on the data and publishes the result through a randomised algorithm π, where for any output a, and any dataset x, $\pi(a|x)$ is the probability that the algorithm generates a. The following properties are desirable from a general standpoint.

Anonymity. Individual participation in the study remains a secret. From the release of the calculations results, nobody can significantly increase their probability of identifying an individual in the database.

Secrecy. The data of individuals is not revealed. The release does not significantly increase the probability of inferring individual's information x_i.

Side-information. Even if an adversary has arbitrary side-information, he cannot use that to amplify the amount of knowledge he would have obtained from the release.

Utility. The released result has, with high probability, only a small error relative to a calculation that does not attempt to safeguard privacy.

The prevalence of drugs in sport

Let's say you need to perform a statistical analysis of the drug-use habits of athletes. Obviously, even if you promise the athlete not to reveal their information, you still might not convince them. Yet, you'd like them to be truthful. The trick is to allow them to randomly change their answers, so that you can't be *sure* if they take drugs, no matter what they answer.

Algorithm for randomising responses about drug use

1. Flip a coin.
2. If it comes heads, respond truthfully.
3. Otherwise, flip another coin and respond `yes` if it comes heads and `no` otherwise.

Calculating the true rate of responses.

Assume that the observed rate of positive responses in a sample is p, that everybody follows the protocol, and the coin is fair. Then, what is the true rate q of drug use in the population?

The problem with this approach, of course, is that we are effectively throwing away half of our data sources. In particular, if we repeated the experiment with a coin that came heads at a rate ϵ, then our error bounds would scale as $O(1/\sqrt{\epsilon n})$ for n data points.

This algorithm is very specific: it assumes binary responses, and it uses a fair coin, which introduces a lot of noise. Since the coin flips make the responses noisy, we may want to have some way of controlling it. In general, we want to consider an algorithm that takes data $x_1, \ldots, x_n$ from n users transforms it randomly to $a_1, \ldots, a_n$ using the following mapping.

The binary randomised response mechanism

Definition 2.5.1 (Randomised response)**.** The i-th user, whose data is $x_i \in \{0, 1\}$, responds with $a_i \in \{0, 1\}$ with probability

$$\pi(a_i = j \mid x_i = k) = p, \qquad \pi(a_i = k \mid x_i = k) = 1 - p,$$

where $j \neq k$.

Given the complete data x, the algorithm's output is $a = (a_1, \ldots, a_n)$. Since the algorithm independently calculates a new value for each data entry, the output probability is

$$\pi(a \mid x) = \prod_i \pi(a_i \mid x_i)$$

This mechanism satisfies the formal notion of ϵ-differential privacy, which will be given in Definition 2.5.2. In a more general setting, we may have multiple possible responses. While the algorithm can be trivially generalised to n-ary outputs, the special case of when the outputs are integers is deferred until later.

The original randomised response mechanism[19]

As first proposed by Warner, the mechanism distributes spinners to people. The spinner has a probability p of landing on A and $1 - p$ of landing on B. This can be easily arranged by having the corresponding areas to have the appropriate proportions. The interesting thing about this mechanism is that the responders must merely say whether or not the spinner landed on the group they identify with. They do *not* have to reveal a group. This makes the mechanism feel like you are revealing less, even if it is just a special case of a general randomised response mechanism.

What can we learn from the output?

In the Bayesian setting, we can think of the adversary as having some prior belief $\beta(x_i)$, expressed as a probability distribution over the secret value of each individual.

After the adversary observes the output, they can form a posterior belief $\beta(x_i \mid a_i)$, representing the information they collected. The following example quantifies the amount of information gained by the adversary.

EXAMPLE 2. For simplicity, consider only one individual, and let the adversary have a prior $\beta(x = 0) = 1 - \beta(x = 1)$ over the values of the true response of an individual. We use the randomised response mechanism with parameter p and the adversary observes the randomised data $a = 1$ for that individual, then what is $\mathbb{P}^{\pi}_{\beta}(x = 1 \mid a = 1)$, assuming the adversary knows the mechanism?

Proof. Bayes's theorem states that[3]

$$\mathbb{P}^{\pi}_{\beta}(x \mid a) = \pi(a \mid x)\beta(x)/\,\mathbb{P}^{\pi}_{\beta}(a),$$

where

$$\mathbb{P}^{\pi}_{\beta}(a) = \sum_{x'} \pi(a \mid x')\beta(x').$$

Let $q = \beta(x = 1)$. Then we have:

$$\mathbb{P}^{\pi}_{\beta}(x = 1 \mid a = 1) = pq/[pq + (1 - p)(1 - q)].$$

It is particularly interesting to consider the case where $q = 1/2$. Then

$$\mathbb{P}^{\pi}_{\beta}(x = 1 \mid a = 1) = p,$$

so we only have limited evidence for whether $x = 1$. Now consider the case where we have some arbitrary prior q, and $p = 1/2$. This means that the output of the algorithm is completely random. Consequently:

$$\beta(x = 1 \mid a = 1) = q = \beta(x = 1).$$

So, in this scenario we learn nothing from the algorithm's output. □

[3]If there are too many symbols for you, you can write Bayes theorem simply with $P(x|a) = P(a|x)P(x)/P(a)$.

Differential privacy.

Now let us take a look at a way to characterise the the inherent privacy properties of algorithms. This is called differential privacy, and it can be seen as a bound on the information an adversary with arbitrary power or side-information could extract from the result of a computation π on the data. For reasons that will be made clear later, this computation has to be stochastic.

☕ ϵ-Differential Privacy

Definition 2.5.2. A stochastic algorithm $\pi : \mathcal{X} \to \boldsymbol{\Delta}(\mathcal{A})$, where $\mathcal{X}$ is endowed with a neighbourhood relation N, is said to be ϵ-differentially private if

$$\left| \ln \frac{\pi(A \mid x)}{\pi(A \mid x')} \right| \leq \epsilon, \qquad \forall x N x', \quad A \subset \mathcal{A}. \tag{2.5.1}$$

This form is related to standard notions of statistical distance such as the KL divergence $\sum_a \ln \frac{\pi(a|x)}{\pi(a|x')} \pi(a \mid x)$. However, the above inequality can be equivalently rewritten as

$$\pi(A \mid x) \leq e^{\epsilon} \pi(A \mid x').$$

Frequently (and particularly when $\mathcal{A}$ is finite) we can also use $\pi(a \mid x)$ without any technical difficulties.

Typically, algorithms are applied to datasets $x = (x_1, \ldots, x_n)$ composed of the data of n individuals. Thus, all privacy guarantees relate to the data contributed by these individuals.

Neighbourhoods

Differential privacy guarantees that it is hard to distinguish neighbouring datasets. Hence, the definition of neighbourhood we use reflects what we want to protect. It makes sense to define neighbourhoods in terms of changes in one individual's data, because then an adversary cannot learn about the values of any particular individual.

In this book we will use two definitions of neighbourhoods. The first is constructed so that the adversary cannot distinguish whether or not any particular individual's information is in the dataset. If not, then they cannot infer the individual's presence from the output of the algorithm.

Definition 2.5.3 (Addition/deletion neighbourhood)**.** If two datasets x, x' are neighbours, then we write $x N x'$, and it holds that

$$x = (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n), \qquad x' = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n),$$

for some i, i.e. if one dataset is missing an element.

This is an important concept if the participation in the dataset is itself sensitive. For example, if somebody is enlisted in study for experimental treatment of a rare disease, then the mere fact that they are part of the study is strong evidence that they have the disease.

The second definition is slightly weaker. It determines whether or not we can distinguish between different *values* of individual data. If not, then they cannot infer whether the data submitted by the individual has a particular value.

Definition 2.5.4 (Edit neighbourhood)**.** We say that two datasets x, x' are neighbours, and write xN_ex' if

$$x = (x_1, \ldots, x_i, \ldots, x_n), \qquad x' = (x_1, \ldots, x'_i, \ldots, x_n), \qquad x_i \neq x'_i.$$

i.e. if one dataset has an altered element.

If x, x' are 1-neighbours under the second definition, then they are 2-neighbours under the first definition. To see this, create a new dataset $\hat{x}$ from x, without the data of person i. Then $xN\hat{x}$ and $\hat{x}Nx'$, since we can change the data of one person by removing the original data x_i and then re-inserting the altered data x'_i. This is illustrated in the example below.

Neihgbourhood example

In this example, we have three datasets, x, x' and $hatx$. In the second dataset, $\hat{x}$, the highlighted row in x is missing. In the third dataset, x', another row with the same name is added, but the height and weight are changed.

Birthday	Name	Height	Weight
06/07	Li Pu	190	80
06/14	Sara Lee	185	110
06/12	*John Smith*	*170*	*82*
01/01	A. B. Student	170	70
05/08	Li Yang	175	72

Table 2.3: Data x

In fact, $\hat{x}$ is obtained through a deletion from x. Of course, the operation can be reversed: x can be obtained from an addition to $\hat{x}$.

Birthday	Name	Height	Weight
06/07	Li Pu	190	80
06/14	Sara Lee	185	110
01/01	A. B. Student	170	70
05/08	Li Yang	175	72

Table 2.4: $\hat{x}$, 1-Neighbour of x.

We can now instead add another row to $\hat{x}$, which will be similar to the row we had removed. This will have the effect of altering the original data in x.

Birthday	Name	Height	Weight
06/07	Li Pu	190	80
06/14	Sara Lee	185	110
06/12	John Smith	*180*	*80*
01/01	A. B. Student	170	70
05/08	Li Yang	175	72

Table 2.5: x', 2-Neighbour of x.

Since x, x' only differ in the contents of a single row, they are edit-neighbours.

As we hinted earlier, the randomised response mechanism satisfies ϵ-DP. The ϵ parameter is dependent on p, with higher values giving a smaller ϵ, and thus better privacy protection.

Remark 2.5.1. The randomised response mechanism with $p \leq 1/2$ is $(\ln \frac{1-p}{p})$-DP with respect to the edit neighbourhood N_e.

Proof. Consider $x = (x_1, \ldots, x_j, \ldots, x_n)$, $x' = (x_1, \ldots, x'_j, \ldots, x_n)$. Then

$$\begin{aligned}
\pi(a \mid x) &= \prod_i \pi(a_i \mid x_i) \\
&= \pi(a_j \mid x_j) \prod_{i \neq j} \pi(a_i \mid x_i) \\
&\leq \frac{1-p}{p} \pi(a_j \mid x'_j) \prod_{i \neq j} \pi(a_i \mid x_i) \\
&= \frac{1-p}{p} \pi(a \mid x')
\end{aligned}$$

$\pi(a_j = k \mid x_j = k) = 1 - p$ so the ratio is $\max\{(1-p)/p, p/(1-p)\} \leq (1-p)/p$ for $p \leq 1/2$. □

Moving to a new neighbourhood

Is the randomised-response mechanism for it to be differentially private with respect to the insertion-neighbourhood definition? If not, is it possible to modify it in order to satisfy that privacy definition? *Hint: The mechanism must be able to hide the participation of a single individual in the database.*

Finally, it may be convenient to the look at neighbourhoods in terms of a distance between datasets. Let $\mathbb{N}^{|\mathcal{X}|}$ be the set of all possible dataset histograms, i.e. counts of different possible rows in each dataset. Then the distance between two datasets is simply the total difference in counts:

Definition 2.5.5 (Hamming distance between datasets)**.**

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \sum_j |\sum_{i=1}^{n} \mathbb{I}\{x_i = j\} - \sum_{i=1}^{n'} \mathbb{I}\{x_i = j\}| = \sum_j |N_j(\boldsymbol{x}) - N_j(\boldsymbol{x}')|, \tag{2.5.2}$$

where $N_j(\boldsymbol{x})$ is the number of elements in $\boldsymbol{x}$ equal to j.

Let us see how the Hamming distance relates to the two neighbourhoods we defined. In particular, $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$ if and only if $\boldsymbol{x} N \boldsymbol{x}'$.

Taking the first definition, if $\boldsymbol{x}'$ has one row less than $\boldsymbol{x}$, but is otherwise identical, then there is exactly one value j for which $N_j(\boldsymbol{x}) = N_j(\boldsymbol{x}') + 1$, with $N_j(\boldsymbol{x}) = N_j(\boldsymbol{x}')$ otherwise. Consequently, $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$.

On the other hand, for the second neighbourhood definition, things are slightly different. There, we assume that $x_i \neq x'_i$ for one i. Without loss of generality, let us say that $x_i = j$ and $x'_i = k$, with $j \neq k$. Then $N_j(\boldsymbol{x}) = N_j(\boldsymbol{x}') + 1$ and $N_k(\boldsymbol{x}') = N_k(\boldsymbol{x}) + 1$. Consequently $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 2$.

The reverse direction follows by contradiction: if the Hamming distance is one, then the two databases must differ in at least one element. If they differ in more, then their distance has to be larger than one.

2.6 The local and central privacy models

So far, we have only seen the concept of differential privacy applied as a method to generate a privatised version of a dataset. In general, however, we want to perform some specific calculations on the data. Is it possible to compute things privately in the first place? Let us assume that you have defined a function $f : \mathcal{X} \to \mathcal{Y}$, which you wish to compute on arbitrary data $x \in \mathcal{X}$. If you wish to preserve privacy, however, you need the computation to satisfy some formal guarantee like differential privacy. The simplest way to do this is by simply generating a dataset a with a DP algorithm and then processing the data with the function we have already specified. This corresponds to the local privacy model.

The local privacy model.

Given a function $f : \mathcal{X} \to \mathcal{Y}$, and a set of individual data $\boldsymbol{x} = (x_i)_{i=1}^n$, use a differentially private algorithm $\pi(\boldsymbol{a}|\boldsymbol{x})$ to generate $\boldsymbol{a} \in \mathcal{X}$. Then output $f(\boldsymbol{a})$.

The advantage of the local model is that the calculation f does not need to be modified. The individuals do not have to trust anybody, as they can modify their data locally. However, they must take care that the DP calculation is performed correctly.

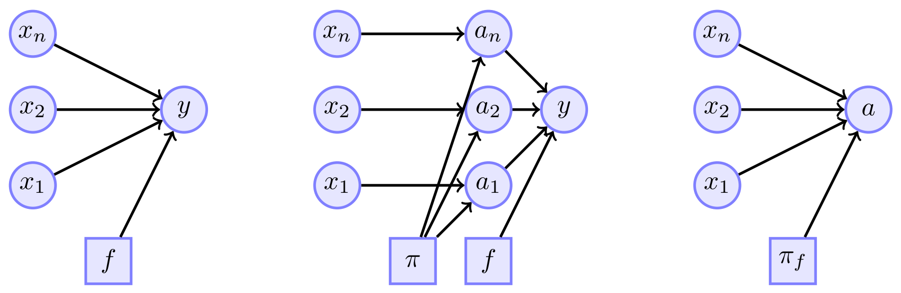
Typically, in the local privacy model, the i-th individual's data x_i is used to generate a private response a_i. This means that individual data is only available to a local private mechanism. This model allows us to publish a complete dataset of private responses. In the central privacy model, the data x is collected and the result a is published by a *trusted curator.*

The central privacy model.

A trusted curator obtains the data $\boldsymbol{x}$ of all individuals and selects a DP policy $\pi(\boldsymbol{a} \mid \boldsymbol{x})$ to generate the output a, so that $\boldsymbol{a} \approx f(\boldsymbol{x})$.

The main advantage of the centralised model is that approximating f with a differentially-private version can be much more accurate than simply using the original function with noisy data. However, obtaining such an approximation is not always easy.

The dependency diagrams in Figures 2.4 shows how the output depends on the individual data, the non-private and private algorithm.



(a) Non-private calculation (b) Local privacy (c) Central privacy

Figure 2.4: Non-privacy, local privacy, and centralised privacy models. In the first case, the value of y depends directly on the function f to be calculated, as well as the data x. In the local privacy model, noise is added to the individual data with a differentially private algorithm π, and the function is calculated on the output. In the central model, a stochastic version π_f of the original function f is calculated on the data.

Trust models.

Google uses a local privacy model to collect data from Android phone users. How does this compare to Google gathering data in the clear and then performing private computations? Who do the users have to trust? What are the privacy risks in either case?

2.6.1 Properties of differential privacy.

Remark 2.6.1. Any differentially private algorithm must be stochastic.

To prove that this is necessary, consider the example of counting how many people take drugs in a competition. If the adversary only doesn't know whether you in particular take drugs, but knows whether everybody else takes drugs, it's trivial to discover your own drug habits by looking at the total. This is because in this case, $f(x) = \sum_i x_i$ and the adversary knows x_i for all $i \neq j$. Then, by observing $f(x)$, he can recover $x_j = f(x) - \sum_{i \neq j} x_i$. Consequently, it is not possible to protect against adversaries with arbitrary side information without stochasticity.

Randomness is also necessary in cryptography. In that setting, Alice wants to communicate a message x to Bob. In a secret key cryptographic scheme, information theoretic security is achieved by making sure that the encrypted message a comes from a uniform distribution $\pi(a|x)$. This is achieved by uniformly selecting a key and selecting an appropriate hash function

Depending on the DP scheme, each query answered may leak privacy. In particular, if we always respond with an ϵ-DP mechanism, after T queries our privacy guarantee is $T\epsilon$. There exist mechanisms that do not respond to each query independently, which can reduce the total privacy loss, but those are outside the scope of this chapter.

Definition 2.6.1 (T-fold composition)**.** In this privacy model, we compose a mechanism out of T differential private mechanisms $\pi_1, \ldots, \pi_T$. This composition is fixed ahead of time.

Theorem 2.6.1. *For any $\epsilon > 0$, the class of ϵ-differentially private mechanism satisfy $T\epsilon$-differential privacy under T-fold composition. More generally, if each mechanism is ϵ_i-DP, the composed mechanism is $\sum_{i=1}^{T} \epsilon_i$-DP.*

Theorem 2.6.2 (Post-processing)**.** *Let mechanism $\pi(a \mid x)$ be ϵ-DP. Applying any transformation $f : A \to Y$ to the output of the mechanism to obtain $y = f(a)$, results in another ϵ-DP mechanism.*

The composition theorem is a very useful tool, as it allows us to create new mechanisms that are composed of simpler parts. As a first example, we can look at how we can use it to create a randomised response algorithm when the respondents provide multiple attributes.

Randomised response for multiple attributes.

Up to now we have been discussing the case where each individual only has one attribute. However, in general each individual t contributes multiple data $x_{t.i}$, which can be considered as a row $\boldsymbol{x}_t$ in a database. Then the mechanism can release each $a_{t,i}$ independently.

For n users and k attributes, if the release of each attribute i is ϵ-DP then the data release is $k\epsilon$-DP. Thus to get ϵ-DP overall, we need ϵ/k-DP per attribute.

The result follows immediately from the composition theorem. We can see each attribute release as the result of an individual query. More generally, if each attribute i is released with an ϵ_i-DP mechanism, the overall mechanism is $\sum_i \epsilon_i$-DP.

Adversary knowledge

Assume that the adversary knows that the data is either $\boldsymbol{x}$ or $\boldsymbol{x}'$. For concreteness, assume the data is either

$$\boldsymbol{x} = (x_1, \dots, x_j = 0, \dots, x_n)$$

where x_i indicates whether or not the i-th person takes drugs, or

$$\boldsymbol{x}' = (x_1, \dots, x'_j = 1, \dots, x_n).$$

In other words, the adversary knows the data of all people apart from one, the j-th person. We can assume that the adversary has some prior belief

$$\beta(\boldsymbol{x}) = 1 - \beta(\boldsymbol{x}')$$

for the two cases. Assume the adversary knows the output a of a mechanism π. What can we say about the posterior distribution of the adversary $\beta(\boldsymbol{x} \mid a, \pi)$ after having seen the output, if π is ϵ-DP? How does it depend on ϵ?

Solution. We can write the adversary posterior as follows.

$$\begin{aligned}
\mathbb{P}^{\pi}_{\beta}(\boldsymbol{x} \mid a) &= \frac{\pi(a \mid \boldsymbol{x})\beta(\boldsymbol{x})}{\pi(a \mid \boldsymbol{x})\beta(\boldsymbol{x}) + \pi(a \mid \boldsymbol{x}')\beta(\boldsymbol{x}')} && (2.6.1)\\
&\geq \frac{\pi(a \mid \boldsymbol{x})\beta(\boldsymbol{x})}{\pi(a \mid \boldsymbol{x})\beta(\boldsymbol{x}) + \pi(a \mid \boldsymbol{x})e^{\epsilon}\beta(\boldsymbol{x}')} && \text{(from DP definition)}\\
&= \frac{\beta(\boldsymbol{x})}{\beta(\boldsymbol{x}) + e^{\epsilon}\beta(\boldsymbol{x}')}. && (2.6.2)
\end{aligned}$$

Note that $\pi(a \mid \boldsymbol{x}) \leq e^{\epsilon}\pi(a \mid \boldsymbol{x}')$ and conversely $\pi(a \mid \boldsymbol{x}') \leq e^{\epsilon}\pi(a \mid \boldsymbol{x})$. We can also then bound the quantity from above:

$$\mathbb{P}^{\pi}_{\beta}(\boldsymbol{x} \mid a) \leq \frac{\beta(\boldsymbol{x})}{\beta(\boldsymbol{x}) + e^{-\epsilon}\beta(\boldsymbol{x}')}.$$

Consequently, $\lim_{\epsilon \to 0} \mathbb{P}^{\pi}_{\beta}(\boldsymbol{x} \mid a) = \beta(\boldsymbol{x})$, hence the information gained by the adversary is bounded by the prior and the information loss ϵ. □

2.7 The Laplace mechanism

Many times we already have a function $f : \mathcal{X} \to \mathbb{R}$ we want to calculate on data, and we would like to make the function preserve privacy. This clearly falls within the *central privacy* model: The analyst has access to the data and function, and wishes for the algorithm generating the final output to be private. One solution, that can satisfy differential privacy, is to add noise to the function's output. We start by first calculating the value of the function for the data we have, $f(x)$, and then we add some random noise ω, hence our calculation is random:

$$a = f(x) + \omega.$$

The amount and type of noise added, together with the smoothness of the function f, determine the amount of privacy we have. One of the simplest noise-adding mechanisms is to add zero-mean Laplace noise. Then we write $\omega \sim \mathcal{Laplace}(\lambda)$.[4] The mechanism is defined below.

[4]When unspecified, the mean parameter is assumed to be zero.

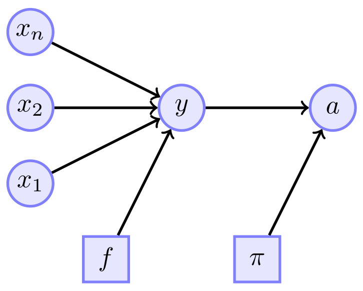


Figure 2.5: Laplace mechanism

Definition 2.7.1 (The Laplace mechanism)**.** For any function $f : \mathcal{X} \to \mathbb{R}$, the output a of the mechanism is sampled from a Laplace distribution with mean $f(x)$ and scaling parameter λ, i.e.

$$\pi(a \mid x) = \mathit{Laplace}(f(x), \lambda). \tag{2.7.1}$$

The probability density function of the Laplace distribution with mean μ and scaling λ is given by:

$$p(\omega \mid \mu, \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|\omega - \mu|}{\lambda}\right).$$

and has mean μ and variance $2\lambda^2$.

The Laplace mechanism for averages

Here we have n individuals for which we wish to calculate the average salary.

- The i-th person receives salary x_i
- We wish to calculate the average salary in a private manner.

We can do this in two ways. By using a DP mechanism on each individual salary and then calculating the average, or by first calculating the average and then applying a DP mechanism to the result. In particular, we can add try adding Laplace noise in both cases.

The local model. In this case, $\pi(a \mid x)$ is obtained by independent Laplace noise ω_i for each individual:

- Obtain $y_i = x_i + \omega_i$, where $\omega_i \sim \mathit{Laplace}(\lambda)$.
- Return $a = n^{-1} \sum_{i=1}^n y_i$.

The centralised model. In this case, $\pi(a \mid x)$ is obtained by averaging first and adding noise later.

- Calcualte $y = n^{-1} \sum_{i=1}^n x_i$.
- Return $a = y + \omega$, where $\omega \sim \mathit{Laplace}(\lambda')$.

We must tune λ, λ' appropriately in to obtain the needed ϵ-DP guarantee.

In the centralised privacy model, the non-private calculation directly outputs $y = f(x)$. We can approximate this with a DP calculation with distribution $\pi(a \mid x)$. The Laplace mechanism does so by first calculating $y = f(x)$ and then generating $\pi(a \mid y)$ so that $\mathbb{E}_\pi[a \mid x] = f(x)$.

Let us now talk about how the Laplace mechanism that can be used both in the centralised and local model when $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} \subset \mathbb{R}$.

DP properties of the Laplace mechanism

To use the Laplace mechanism in the centralised setting, we must relate it to a specific function f that we wish to compute. In particular, the mecnahism works by adding noise to the output of the function f in a carefully calibrated manner so that we achieve exactly ϵ-differential privacy.

Definition 2.7.2 (Sensitivity)**.** The sensitivity of a function f is

$$\mathbb{L}(f) \triangleq \sup_{xNx'} |f(x) - f(x')|$$

If we define a metric d, so that $d(x, x') = 1$ for xNx', then:

$$|f(x) - f(x')| \leq \mathbb{L}(f)\, d(x, x'),$$

i.e. f is $\mathbb{L}(f)$-Lipschitz with respect to d.

Example 3. If $f : \mathcal{X} \to [0, B]$, e.g. $\mathcal{X} = \mathbb{R}$ and $f(x) = \min\{B, \max\{0, x\}\}$, then $\mathbb{L}(f) = B$.

Example 4. If $f : [0, B]^n \to [0, B]$ is $f = \frac{1}{n}\sum_{t=1}^{n} x_t$, then $\mathbb{L}(f) = B/n$.

Proof. Consider two neighbouring datasets x, x' differing in example j. Then

$$f(x) - f(x') = \frac{1}{n}\left[f(x_j) - f(x'_j)\right] \leq \frac{1}{n}[B - 0]$$

□

Theorem 2.7.1. *The Laplace mechanism on a function f with sensitivity $\mathbb{L}(f)$, ran with $\mathcal{Laplace}(\lambda)$ is $\mathbb{L}(f)/\lambda$-DP. Consequently, if the Laplace mechanism is ran with $\lambda = \mathbb{L}(f)/\epsilon$, then it is ϵ-DP.*

Proof.

$$\frac{\pi(a \mid x)}{\pi(a \mid x')} = \frac{e^{|a - f(x')|/\lambda}}{e^{|a - f(x)|/\lambda}} \leq \frac{e^{|a - f(x)|/\lambda + \mathbb{L}(f)/\lambda}}{e^{|a - f(x)|/\lambda}} = e^{\mathbb{L}(f)/\lambda}$$

The first step follows from the definition of the Laplace mechanism. The inequality follows from the fact that for xNx', we have $|f(x) - f(x')| \leq \mathbb{L}(f)$. In particular,

(a) If $f(x') - a \geq 0$ then

$$|f(x') - a| = f(x') - a \leq \mathbb{L}(f) + f(x) - a \leq L + |a - f(x)|,$$

as $f(x') \leq f(x) + \mathbb{L}(f)$ from the Lipschitz property.

(b) If $f(x') - a < 0$ then

$$|f(x') - a| = a - f(x') \leq \mathbb{L}(f) - f(x) + a \leq L + |a - f(x)|,$$

as $-f(x') \leq \mathbb{L}(f) - f(x)$.

Replacing into the exponential gives us the required inequality. The final result is obtained with elementary algebra. □

DP properties of the average in the local and central model.

What is the effect of applying the Laplace mechanism in the local versus centralised model? Let us continue the average example. Here let us assume $x_i \in [0, B]$ for all i.

The Laplace mechanism in the local privacy model

The sensitivity of the individual data is B, so to obtain ϵ-DP we need to use $\lambda = B/\epsilon$. The variance of each component is $2(B/\epsilon)^2$, so the total variance is $2B^2/\epsilon^2 n$.

The Laplace mechanism in the centralised privacy model

The sensitivity of f is B/n, so we only need to use $\lambda = \frac{B}{n\epsilon}$. The variance of a is $2(B/\epsilon n)^2$.

Thus the two models have a significant difference in the variance of the estimates obtained, for the same amount of privacy. While the central mechanism has variance $O(n^{-2})$, the local one is $O(n^{-1})$ and so our estimates will need much more data to be accurate under this mechanism. In particular, we need square the amount of data in the local model as we need in the central model. Nevertheless, the local model may be the only possible route if we have no specific use for the data.

2.8 Interactive data access.

In a lot of applications, we cannot pre-define the computation that we want to perform. After we have collected the data, we need to be perform an adaptive data analysis. This requires performing a sequence of computations on the data, where the result of one computation determines what computation we will do next.

We can think of this as performing queries to the database. You may be familiar with database access languages such as SQL, where you ask a question such as "what is the sum of the attribute `age` in table `students`?"and obtain the exact sum back. It is possible to answer such queries through a differentially private mechanism. However, the more queries you answer, the more you reveal about the original data. In addition, since an adversary may be cleverly adjusting the queries to more efficiently discover a secret, we need the concept of adaptive composition.

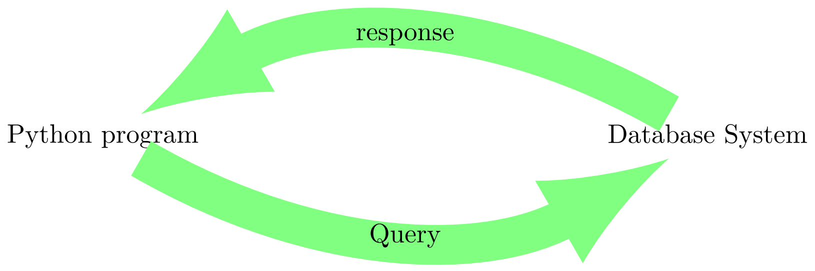


Figure 2.6: Database access model

2.8.1 Utility of queries

Rather than saying that we wish to calculate a private version of some specific function f, sometimes it is more useful to consider the problem from the perspective of the utility of different answers to queries. More precisely, imagine the interaction between a database system and a user:

Interactive queries

- System has data x.
- At time t, user asks query $q_t = q$.
- System responds with $a_t = a$.
- There is a common utility function $U : \mathcal{X}, \mathcal{A}, \mathcal{Q} \to \mathbb{R}$.

We wish to give the most useful answer, by return a that maximises U, i.e. $a = \arg\max_{a'} U(x, a', q)$, but are constrained by the fact that we also want to preserve privacy.

The utility $U(x, a, q)$ describes how appropriate each answer a given by the system for a query q is given the data x. It can be seen as how useful the response is [5] It allows us to quantify exactly how much we would gain by replying correctly. The exponential mechanism, described below is a simple differentially private mechanism for responding to queries while trying to maximise utility for *any possible* utility function.

The Exponential Mechanism.

Here we assume that we can answer queries q, whereby each possible answer a to the query has a different utility to the DM: $U(q, a, x)$. The idea is that the best answer to the query should have the highest utility. The further away from the correct answer the response is, the lower the utility should be. This allows us to quantify how much we value correct answers to a query.

As an example, if the optimal response to a query is given by a real-valued function $f(q, x)$, then one possible utility function is $U(q, a, x) = -[f(q, x) - a]^2$. This results in the correct response having a utility of zero, and responses whose value is further away will have negative values.

The exponential mechanism allows us to answer queries in a "soft" manner, by using this utility function. It assigns lower probability to answers with lower utility, hence the better responses have a higher likelihood of being selected.

Definition 2.8.1 (The Exponential mechanism)**.** For any utility function $U : \mathcal{Q} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$, define the policy, which implicitly depends on U and ϵ, as:

$$\pi(a \mid x, q) \triangleq \frac{e^{\epsilon U(q,a,x)/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}}, \tag{2.8.1}$$

where $\mathbb{L}\left(U(q)\right) \triangleq \max_a \sup_{xNx'} |U(q, a, x) - U(q, a, x')|$ denotes the sensitivity of a query.

Clearly, when $\epsilon \to 0$, this mechanism is uniformly random. When $\epsilon \to \infty$ the action maximising $U(q, a, x)$ is always chosen (see Exercise 4). Although the exponential mechanism can be used to describe some known DP mechanisms (see Exercise 5), its best use is in settings where there is a natural utility function.

[5] This is essentially the utility to the user that asks the query, but it could be the utility to the person that answers. In either case, the motivation does not matter the action should maximise it, but is constrained by privacy.

Theorem 2.8.1. *The exponential mechanism is ϵ-differentially private.*

Proof. We only need to look at the ratio between two different distributions of answers to queries. Then we have:

$$\begin{aligned}
\frac{\pi(a \mid x, q)}{\pi(a \mid x', q)} &= \frac{e^{\epsilon U(q,a,x)/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}} \times \frac{\sum_{a'} e^{\epsilon U(q,a',x')/\mathbb{L}(U(q))}}{e^{\epsilon U(q,a,x')/2\mathbb{L}(U(q))}} \\
&= e^{\epsilon U(q,a,x)/2\mathbb{L}(U(q)) - \epsilon U(q,a,x')/2\mathbb{L}(U(q))} \times \frac{\sum_{a'} e^{\epsilon U(q,a',x')/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}} \\
&= e^{\epsilon [U(q,a,x) - U(q,a,x')]/2\mathbb{L}(U(q))} \times \frac{\sum_{a'} e^{\epsilon U(q,a',x')/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}} \\
&\le e^{\epsilon/2} \times \frac{\sum_{a'} e^{\epsilon U(q,a',x')/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}} \\
&\le e^{\epsilon/2} \times \frac{\sum_{a'} e^{\epsilon [(U(q,a',x) + \mathbb{L}(U(q)))/2\mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q,a',x)/2\mathbb{L}(U(q))}} \le e^{\epsilon}
\end{aligned}$$

□

Interactive differential privacy.

So far we only defined differential privacy in terms of a fixed mechanism. However, in general the mechanism must respond to sequential queries $q_1, \ldots, q_t$, with answers $a_1, \ldots, a_t$. The answer can depend arbitrarily on the sequence of responses and queries. So, privacy must hold no matter what the previous sequence of questions and answers. This gives rise to the following definition.

Definition 2.8.2 (Interactive DP)**.** An algorithm π satisfies interactive $\{\epsilon_1, \ldots, \epsilon_t\}$ differential privacy with respect to N if

$$\ln \frac{\pi(a_t | x, q_1, \ldots, q_t, a_1, \ldots, a_{t-1})}{\pi(a_t | x', q_1, \ldots, q_t, a_1, \ldots, a_{t-1})} \le \epsilon_t, \qquad \forall x N x'. \tag{2.8.2}$$

Since the queries are not defined in advance, this means that q_t may also depend on all the previous answers $a_1, \ldots, a_{t-1}$.

Answering queries independently in the interactive setting is sufficient for achieving differential privacy ϵ_t at each step t. The total privacy loss is at most $\sum_{i=1}^{t} \epsilon_i$. However, it is possible to bound the privacy loss more tightly with advanced composition theorems.

2.8.2 Differential privacy as a hypothesis testing game.

2.9 Advanced topics.

In this section, we give a brief overview of some more advanced topics on privacy. The interested reader is urged to look into the given references.

2.9.1 Relaxations of differential privacy.

In practice, satisfying differential privacy with a small ϵ results in mechanisms with low utility. It is possible to relax the definition of differential privacy to include an additive term, as shown below, which greatly improves the privacy-utility trade-off we can achieve.

Definition 2.9.1. Approximate differential privacy. A stochastic algorithm $\pi : \mathcal{X} \to \boldsymbol{\Delta}(\mathcal{A})$, where $\mathcal{X}$ is endowed with a neighbourhood relation N, is said to be (ϵ, δ) differentially private if

$$\pi(A \mid x) \leq \pi(A \mid x')\epsilon + \delta, \qquad \forall x N x', \quad A \subset \mathcal{A}. \tag{2.9.1}$$

One simple interpretation of this definition is that the mechanism is $(\epsilon, 0)$-DP with probability $1 - \delta$. Conversely, this implies that with a very small probability δ, the complete dataset might be revealed. For that reason, the parameter δ must be as small as possible.

Approximate differential privacy is satisfied by the *Gaussian* mechanism. This is similar to the Laplace mechanism, but scales noise to the ℓ_2 sensitivity of the function.

The Gaussian mechanism

Given a function $f : \mathcal{X} \to \mathbb{R}^k$, the Gaussian mechanism with parameter σ outputs $a \sim \mathcal{N}(f(x), \sigma)$.

Let ℓ_2 sensitivity of f be

$$\mathbb{L}\left(f\right)_2 = \max\left\{\|f(x) - f(y)\|_2 \mid xNy\right\}.$$

For $\sigma \geq 2\ln(1.25/\delta)\frac{\Delta_2(f)}{\epsilon}$, then the Gaussian mechanism is (ϵ, δ)-differentially private.

Renyi differential privacy.

The careful reader will have noticed that differential privacy is essentially a bound on the distributional distance of mechanism outputs for similar inputs. In particular, the distance used for ϵ-DP is $D_\infty(P\|Q) \triangleq \sup_A |\ln P(A)/Q(A)|$. We can use any other divergence between distributions, such as the KL divergence, as long as this becomes unbounded for distributions with unequal support.

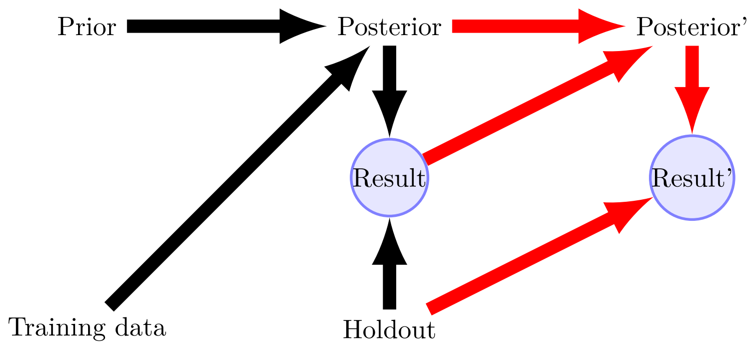
Reproducibility

Training and testing, overfitting on the test set.

The unfortunate practice of adaptive analysis

In the ideal data analysis, we start from some prior hypothesis, then obtain some data, which we split into training and holdout. We then examine the training data and obtain a posterior that corresponds to our conclusions. We can then measure the quality of these conclusions in the independent holdout set.

However, this is not what happens in general. Analysts typically use the same holdout repeatedly, in order to improve the performance of their algorithms. This can be seen as indirectly using the holdout data to obtain a new posterior, and so it is possible that you can overfit on the holdout data, even if you never directly see it. It turns out we can solve this problem if we use differential privacy, so that the analyst only sees a differentially private version of queries.



The reusable holdout[106]

One idea to solve this problem is to only allow the analyst to see a private version of the result. In particular, the analyst will only see whether or not the holdout result is τ-close to the training result.

Algorithm parameters

- Performance measure f.
- Threshold τ. How close do we want f to be on the training versus holdout set?
- Noise σ. How much noise should we add?
- Budget B. How much are we allowed to learn about the holdout set?

Algorithm idea

Run algorithm λ on data D_T and get e.g. classifier parameters θ.
Run a DP version of the function $f(\theta, D_H) = \mathbb{I}\{U(\theta, D_T) \geq \tau U(\theta, D_H)\}$.

So instead of reporting the holdout performance at all, you just see if you are much worse than the training performance, i.e. if you're overfitting. The fact that the mechanism is DP also makes it difficult to learn the holdout set. See the thresholdout link for more details.

2.10 Discussion

The definition of differential privacy

- First rigorous mathematical definition of privacy.
- Relaxations and generalisations possible.
- Connection to learning theory and reproducibility.

[6]Also see `https://ai.googleblog.com/2015/08/the-reusable-holdout-preserving.html`

Current uses

- Apple. DP is used internally in the company to "protect user privacy". It is not clear exactly what they are doing but their efforts seem to be going in the right direction.
- Google. The company has a DP API available based on randomised response, RAPPOR.
- Uber. Elastic sensitivity for SQL queries, which is available as open source. This is a good thing, because it is easy to get things wrong with privacy.
- US 2020 Census. It uses differential privacy to protect the condidentiality of responders' information while maintaining data that are suitable for their intended uses.

Open problems

- Complexity of differential privacy.
- Verification of implementations and queries.

Available privacy toolboxes

Differential privacy

- Open DP <https://opendp.org/>
- `https://github.com/bmcmenamin/thresholdOut-explorations`Threshold out
- `https://github.com/steven7woo/Accuracy-First-Differential-Privacy`Accuracy-constrained DP
- `https://github.com/menisadi/pydp`Various DP algorithms
- `https://github.com/haiphanNJIT/PrivateDeepLearning` Deep learning and DP

k-anonymity

- `https://github.com/qiyuangong/Mondrian` Mondrian k-anonymity

Learning outcomes

Understanding

- Linkage attacks and k-anonymity.
- Inferring data from summary statistics.
- The local versus centralised differential privacy model.
- False discovery rates.

Skills

- Make a dataset satisfy k-anonymity with respect to identifying attributes.
- Apply the randomised response and Laplace mechanism to data.
- Apply the exponential mechanism to simple decision problems.
- Use differential privacy to improve reproducibility.

Reflection

- How can potentially identifying attributes be chosen to achieve k-anonymity?
- How should the parameters of the two ideas, ϵ-DP and k-anonymity be chosen?
- Does having more data available make it easier to achieve privacy?

Further reading

- k-anonymity[17]
- Randomness, privacy, and the US 2020 census[13]
- The paper introducing differential privacy[9]
- Differential privacy book[8]
- Bayesian inference and privacy[6]
- Local differential privacy and statistics[7]
- Local differential privacy and applications[20]
- The exponential mechanism[16]

2.11 Exercises

Exercise 1. Show that the randomised response mechanism, as originally defined, is not differentially private with respect to the addition/deletion neighbourhood.

Proof. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ be three datasets with $\boldsymbol{x}_1 N \boldsymbol{x}_2 N \boldsymbol{x}_3$ and $\boldsymbol{x}_1 N_e \boldsymbol{x}_3$. First of all, an algorithm that is ϵ-DP with respect to the N, is 2ϵ-DP with respect to N_e. This follows from the fact that $\pi(a|\boldsymbol{x}_1) \leq e^{\epsilon}\pi(a|\boldsymbol{x}_2) \leq e^{2\epsilon}\pi(a|\boldsymbol{x}_3)$. What about the converse? Consider the case where $\boldsymbol{x}_1$ has n entries and $\boldsymbol{x}_2$ has $n+1$ entries. Let A_n be the set of all responses with n entries. Then clearly $\pi(A_n|\boldsymbol{x}_1) = 1$ and $\pi(A_n|\boldsymbol{x}_2) = 0$. Consequently $|\ln \frac{\pi(A_n|\boldsymbol{x}_1)}{\pi(A_n|\boldsymbol{x}_2)}|$ is not bounded by any constant, and differential privacy is not satisfied with respect to insertions and deletions. $\square$

Exercise 2. Define a variant of the binary randomised response mechanism that satisfies differential privacy with respect to the insertion and deletion neighbourhood. More precisely, it should hold that for any dataset pair $\boldsymbol{x} = (x_1, \ldots, x_n)$, and $\boldsymbol{x}' = (x_1, \ldots, x_n)$, $\boldsymbol{x} = (x_1, \ldots, x_n, x_{n+1})$, the mechanism satisfies

$$\pi(A|\boldsymbol{x}) \leq e^{\epsilon}\pi(A|\boldsymbol{x}'), \qquad \forall A \subset \mathcal{A}.$$

Proof. First of all, for the mechanism to satisfy ϵ-DP, it must be the case that an output size must have a non-zero probability: If with an n-size input we can only generate $\leq n$-size outputs, then we cannot generate a $n+1$-size output. Consequently, the randomised response mechanism defined for N-neighbourhoods makes sense only when we know the maximum possible value of n. This is normally the case when we are surveying a population whose size is assumed to be common knowledge.

In particular, consider a mechanism which samples person i with probability q so that $s_i = 1$ if a person is selected. If a person is selected, then a_i is drawn from a standard randomised response mechanism (which flips x_i with probability p), otherwise $a_i = \perp$. Then, we can factorise the probability of a specific output as follows:

$$\pi(a_i = k | x_i = k) = \pi(a_i = k | s_i = 1, x_i = k)$$

□

Exercise 3. The mean estimator $\hat{\theta}$ for the randomised response mechanism with flipping probability p, where the true data generation process is Bernoulli with parameter θ, is defined as

$$\hat{\theta}(a) = \frac{\bar{a} - p}{1 - 2p},$$

where $\bar{a} = \frac{1}{T}\sum_{t=1}^{T} a_t$ is the mean of the observed answers. Prove that this estimator is unbiased, i.e. that

$$\mathbb{E}[\hat{\theta}] = \theta,$$

where the expectation is taken over the unobserved data randomness and the randomness of the mechanism.

Exercise 4. Prove that the exponential mechanism is uniform when $\epsilon \to 0$ and deterministically returns the maximising action when $\epsilon \to \infty$.

Exercise 5. Prove that the exponential mechanism, when we used to calculate a noisy version of the function $q(x)$ with $q : \mathcal{X} \to \mathbb{R}$ and utility $U(a, q, x) = -|q(x) - a|^p$, results in the Laplace mechanism for $p = 1$ and the Gaussian mechanism for $p = 2$.

Exercise 6. Consider the following relaxation of differential privacy to KL divergences. A mechanism is ϵ-KL-private if

$$\sum_a \ln \frac{\pi(a|x)}{\pi(a|x')} \pi(a|x) \leq \epsilon$$

Prove that two-folded composition of such a mechanism is 2ϵ-KL-private.

Proof. For simplicity, we compose the same mechanism twice over a finite alphabet. Then $\pi(a, a'|x) = \pi(a|x)\pi(a'|x)$ because the mechanism is not interactive.

$$\begin{aligned}
\sum_{a,a'} \ln \frac{\pi(a, a'|x)}{\pi(a, a'|x')} \pi(a, a'|x) &= \sum_{a,a'} \ln \frac{\pi(a|x)\pi(a'|x)}{\pi(a|x')\pi(a'|x')} \pi(a|x)\pi(a'|x) \\
&= \sum_{a,a'} \ln \frac{\pi(a|x)}{\pi(a|x')} \pi(a|x)\pi(a'|x) + \ln \frac{\pi(a'|x)}{\pi(a'|x')} \pi(a|x)\pi(a'|x) \\
&= \sum_{a'} \pi(a'|x) \sum_a \ln \frac{\pi(a|x)}{\pi(a|x')} \pi(a|x) + \sum_a \pi(a|x) \sum_{a'} \ln \frac{\pi(a'|x)}{\pi(a'|x')} \pi(a'|x) \\
&\leq \sum_{a'} \pi(a'|x)\epsilon + \sum_a \pi(a|x)\epsilon = 2\epsilon.
\end{aligned}$$

$\square$

Chapter 3

Fairness

When machine learning algorithms are applied at scale, it can be difficult to imagine what their effects might be. Ir this part of the course, we consider notions of fairness as seen through the prism of conditional independence and meritocracy. The first notion requires that we look deeper into directed graphical models.

3.1 Fairness in machine learning

3.2 Concepts of fairness

Fairness

What is it?

- *Meritocracy.*
- Proportionality and representation.
- Equal treatment.
- *Non-discrimination.*

Meritocracy

Example 5 (College admissions).
- Student A has a grade 4/5 from Gota Highschool.
- Student B has a grade 5/5 from Vasa Highschool.

Example 6 (Additional information).
- 70% of admitted Gota graduates with 4+ get their degree.
- 50% of admitted Vasa graduates with 5 get their degree.

We still don't know how a *specific* student will do!

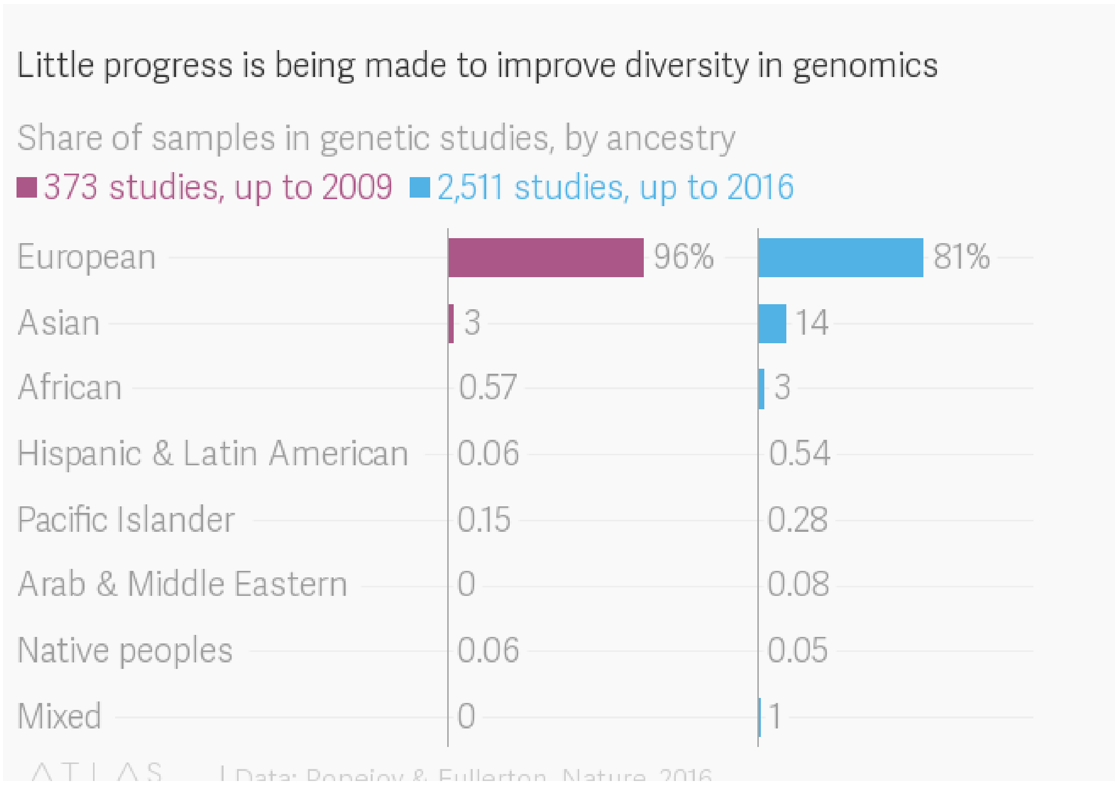
<

4->Solutions

- Admit *everybody*?

- Admit *randomly*?
- Use *prediction* of individual academic performance?

Proportional representation



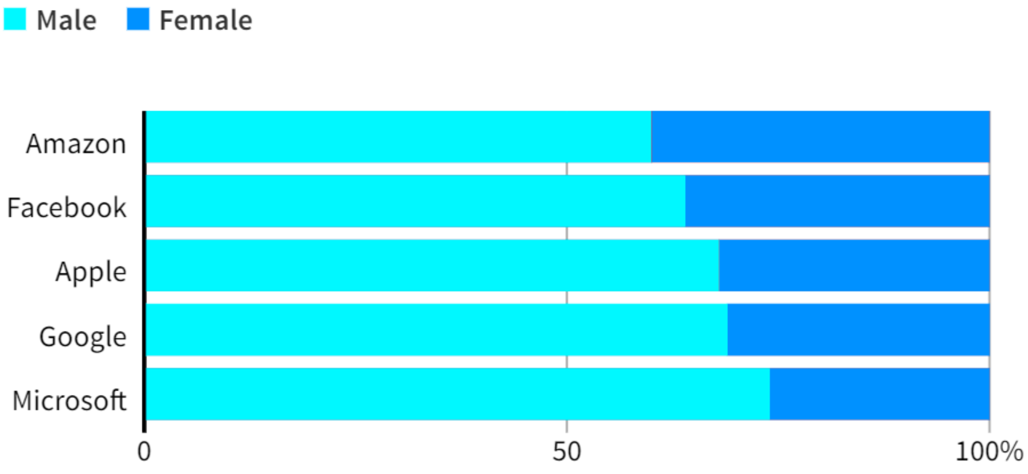
https://qz.com/1367177/

Hiring decisions

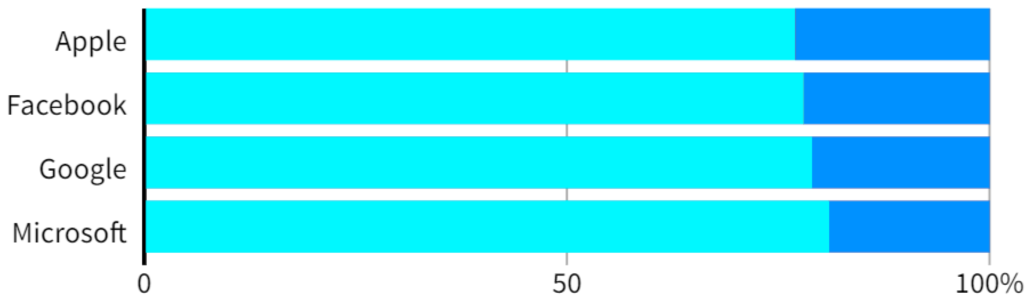
Dominated by men

Top U.S. tech companies have yet to close the gender gap in hiring, a disparity most pronounced among technical staff such as software developers where men far outnumber women. Amazon's experimental recruiting engine followed the same pattern, learning to penalize resumes including the word "women's" until the company discovered the problem.

GLOBAL HEADCOUNT



EMPLOYEES IN TECHNICAL ROLES



Note: Amazon does not disclose the gender breakdown of its technical workforce.
Source: Latest data available from the companies, since 2017.

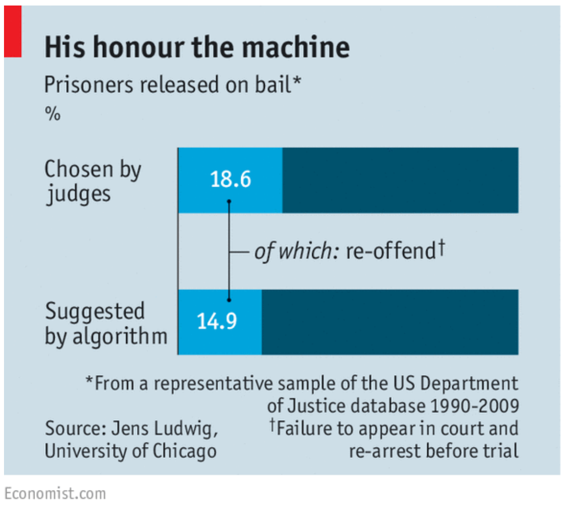


Figure 3.1: In some cases, it appears as though automating this procedure might lead to better outcomes. But is that generally true?

The problem of fairness in machine learning and artificial intelligence has only recently been widely recognised. When any algorithm is implemented at scale, no matter the original objective and whether it is satisfied, it has significant societal effects. In particular, even when considering the narrow objective of the algorithm, even if it improves it overall, it may increase inequality.

In this course we will look at two aspects of fairness. The first has to do with disadvantaged populations that form distinct social classes due to a shared income stratum, race or gender. The second has to do with meritocratic notions of fairness.

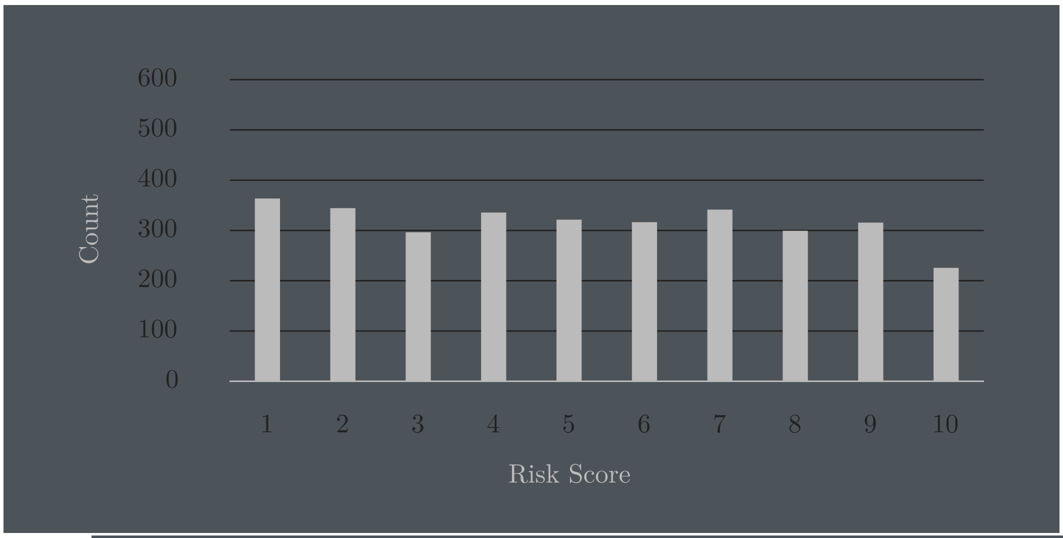
Bail decisions

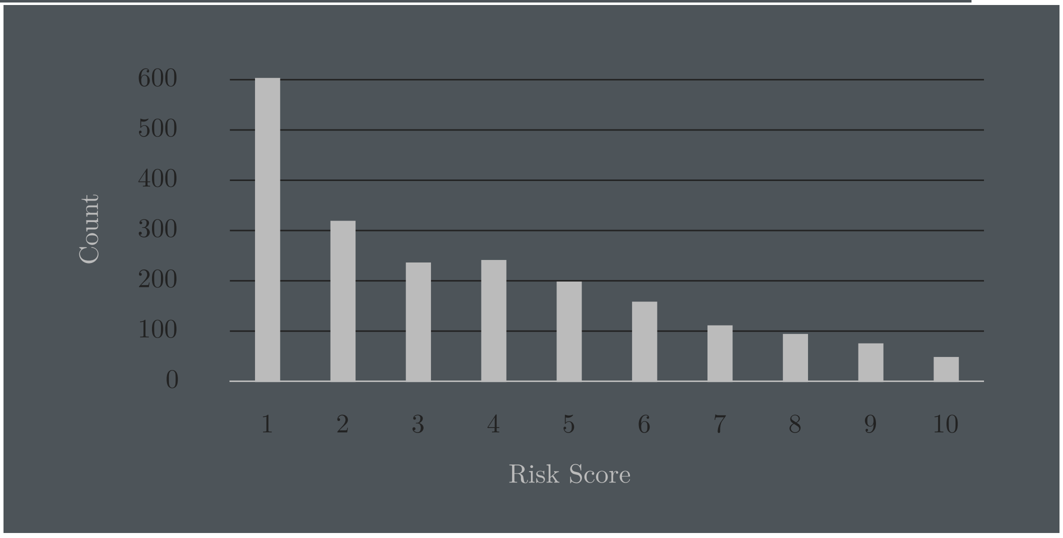
For our example regarding disadvantaged populations, consider the example of bail decisions in the US court system. When a defendant is charged, the judge has the option to either place them in jail pending trial, or set them free, under the condition that the defendant pays some amount of bail. The amount of bail (if any) is set to an amount that would be expected to deter flight or a relapse.

Whites get lower scores than blacks[1]

In a different study, it was shown that a commonly used software tool for determining 'risk scores' in the US was biased towards white defendants, who seemed to be always getting lower scores than blacks.

[1] Pro-publica, 2016





Black
White

Figure 3.2: Apparent bias in risk scores towards black versus white defendants.

But scores equally accurately predict recidivsm[2]

On the other hand, the scores generated by the software seemed to be very predictive on whether or not defendants would re-offend, independently of their race.

[2]Washington Post, 2016

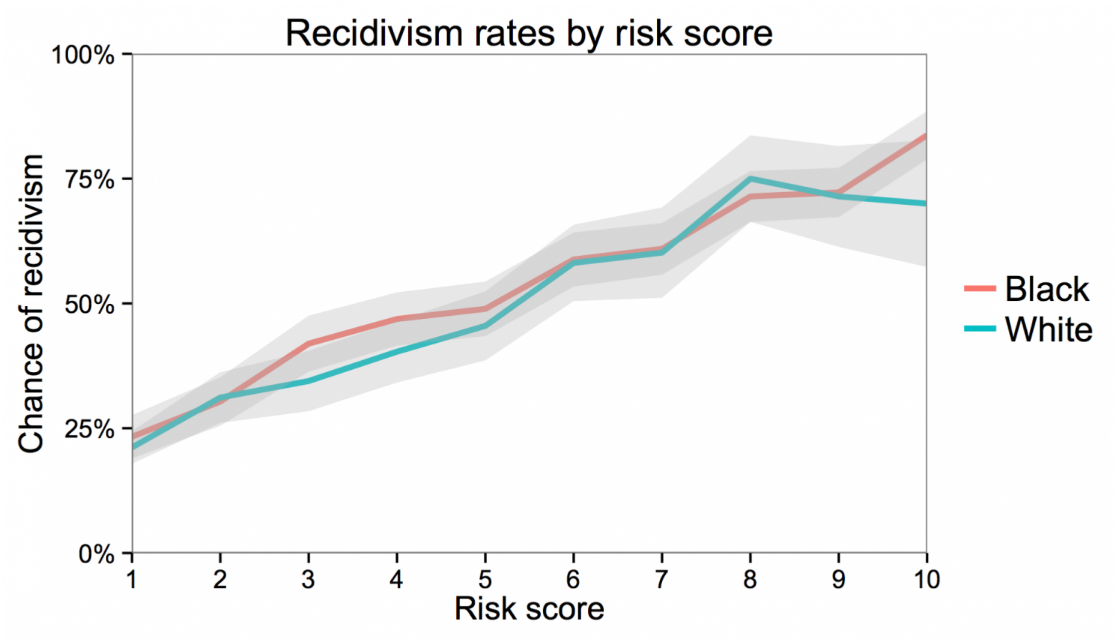


Figure 3.3: Recidivism rates by risk score.

But non-offending blacks get higher scores
On the third hand, we see that the system seemed to give higher risk scores to non-offending blacks. So, is there a way to fix that or not?

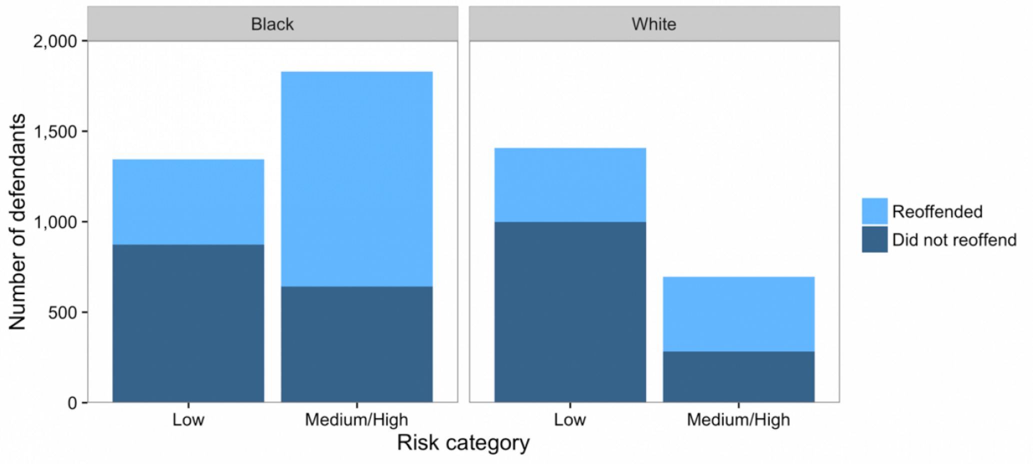


Figure 3.4: Score breakdown based on recidivism rates.

How can we explain this discrepancy? We can show that in fact, each one of these different measures of bias in our decision rules can be seen as a notion of conditional independence.

Chapter 4

Decision and Learning Theory

This chapter deals with simple decision problems, whereby a decision maker (DM) makes a simple choice among many. In some of this problems the DM has to make a decision after first observing some side-information. Then the DM uses a *decision rule* to assign a probability to each possible decision for each possible side-information. However, designing the decision rule is not trivial, as it relies on previously collected data. A higher-level decision includes choosing the decision rule itself. The problems of classification and regression fall within this framework. While most steps in the process can be automated and formalised, a lot of decisions are actual design choices made by humans. This creates scope for errors and misinterpretation of results.

In this chapter, we shall formalise all these simple decision problems from the point of view of statistical decision theory. The first question is, given a real world application, what type of decision problem does it map to? Then, what kind of machine learning algorithms can we use to solve it? What are the underlying assumptions and how valid are our conclusions?

4.1 Hierarchies of decision making problems

All machine learning problems are essentially decision problems. This essentially means replacing some human decisions with machine decisions. One of the simplest decision problems is classification, where you want an algorithm to decide the correct class of some data, but even within this simple framework there is a multitude of decisions to be made. The first is how to frame the classification problem the first place. The second is how to collect, process and annotate the data. The third is choosing the type of classification model to use. The fourth is how to use the collected data to find an optimal classifier within the selected type. After all this has been done, there is the problem of classifying new data. In this course, we will take a holistic view of the problem, and consider each problem in turn, starting from the lowest level and working our way up.

4.1.1 Simple decision problems

Preferences

The simplest decision problem involves selecting one item from a set of choices, such as in the following examples

Food

EXAMPLE 7. A McDonald's cheeseburger

B Surstromming

C Oatmeal

Money

A 10,000,000 CHF

B 10,000,000 USD

C 1,000 BTC

Entertainment

A 1 Week in Venice

B 1 Week Hoch-Tour in the Alps

C 1 Week Transatlantic Cruise

Rewards and utilities

In the decision theoretic framework, the things we receive are called rewards, and we assign a utility value to each one of them, showing which one we prefer.

- Each choice is called a *reward* $r \in \mathcal{R}$.
- There is a *utility function* $U : \mathcal{R} \to \mathbb{R}$, assigning values to reward.
- We (weakly) prefer A to B iff $U(A) \geq U(B)$.

In each case, given U the choice between each reward is trivial. We just select the reward:

$$r^* \in \arg\max_r U(r)$$

The main difficult is actually selecting the appropriate utility function. In a behavioural context, we simply assume that humans act with respect to a specific utility function. However, figuring out this function from behavioural data is non trivial. ven when this assumption is correct, individuals do not have a common utility function.

EXERCISE 7. From your individual preferences, derive a *common utility function* that reflects everybody's preferences in the class for each of the three examples. Is there a simple algorithm for deciding this? Would you consider the outcome fair?

Preferences among random outcomes

EXAMPLE 8. Would you rather ...

A Have 100 EUR now?

B Flip a coin, and get 200 EUR if it comes heads?

The expected utility hypothesis

Rational decision makers prefer choice A to B if

$$\mathbb{E}(U|A) \geq \mathbb{E}(U|B),$$

where the expected utility is

$$\mathbb{E}(U|A) = \sum_r U(r)\,\mathbb{P}(r|A).$$

In the above example, $r \in \{0, 100, 200\}$ and $U(r)$ is increasing, and the coin is fair.

Risk and monetary rewards

When $r \in \mathbb{R}$, as in the case of monetary rewards, we can use the shape of the utility function determines the amount of risk aversion. In particular:

- If U is convex, we are risk-seeking. In the example above, we would prefer B to A, as the expected utility of B would be higher than A, even though they give the same amount of money on average.
- If U is linear, we are risk neutral. In the example above, we would be indifferent between A and B, as the expected amount of money is the same as the amount of money we get.
- If U is concave, we are risk-averse. Hence, in the example above, we prefer A.

This idea of risk can be used with any other utility function. We can simply replace the original utility function U with a monotonic function $f(U)$ to achieve risk-sensitive behaviour. However, this is not the only risk-sensitive approach possible.

Uncertain rewards

However, in real life, there are many cases where we can only choose between uncertain outcomes. The simplest example are lottery tickets, where rewards are essentially random. However, in many cases the rewards are not really random, but simply uncertain. In those cases it is useful to represent our uncertainty with probabilities as well, even though there is nothing really random.

- Decisions $a \in \mathcal{A}$
- Each choice is called a *reward* $r \in \mathcal{R}$.
- There is a *utility function* $U : \mathcal{R} \to \mathbb{R}$, assigning values to reward.
- We (weakly) prefer A to B iff $U(A) \geq U(B)$.

Example 9. You are going to work, and it might rain. What do you do?

- a_1: Take the umbrella.
- a_2: Risk it!
- ω_1: rain
- ω_2: dry

$\rho(\omega, a)$	a_1	a_2
ω_1	dry, carrying umbrella	wet
ω_2	dry, carrying umbrella	dry
$U[\rho(\omega, a)]$	a_1	a_2
ω_1	0	-10
ω_2	0	1

Table 4.1: Rewards and utilities.

$\rho(\omega, a)$	a_1	a_2
ω_1	dry, carrying umbrella	wet
ω_2	dry, carrying umbrella	dry
$U[\rho(\omega, a)]$	a_1	a_2
ω_1	0	-10
ω_2	0	1
$\mathbb{E}(U \mid a)$	0	-1.2

Table 4.2: Rewards, utilities, expected utility for 20% probability of rain.

- $\max_a \min_\omega U = 0$
- $\min_\omega \max_a U = 0$

Expected utility

$$\mathbb{E}(U \mid a) = \sum_r U[\rho(\omega, a)] \, \mathbb{P}(\omega \mid a)$$

Example 10. You are going to work, and it might rain. The forecast said that the probability of rain (ω_1) was 20%. What do you do?

- a_1: Take the umbrella.
- a_2: Risk it!

4.1.2 Decision rules

We now move from simple decisions to decisions that depend on some observation. We shall start with a simple problem in applied meteorology. Then we will discuss hypothesis testing as a decision making problem. Finally, we will go through an exercise in Bayesian methods for classification.

Bayes decision rules

Consider the case where outcomes are independent of decisions:

$$U(\beta, a) \triangleq \sum_\mu U(\mu, a)\beta(\mu)$$

This corresponds e.g. to the case where $\beta(\mu)$ is the belief about an unknown world.

Definition 4.1.1 (Bayes utility)**.** The maximising decision for β has an expected utility equal to:

$$U^*(\beta) \triangleq \max_{a \in \mathcal{A}} U(\beta, a). \tag{4.1.1}$$

The n-meteorologists problem

Of course, we may not always just be interested in classification performance in terms of predicting the most likely class. It strongly depends on the problem we are actually wanting to solve. In biometric authentication, for example, we want to guard against the unlikely event that an impostor will successfully be authenticated. Even if the decision rule that always says 'OK' has the lowest classification error in practice, the expected cost of impostors means that the optimal decision rule must sometimes say 'Failed' even if this leads to false rejections sometimes.

EXERCISE 8. Assume you have n meteorologists. At each day t, each meteorologist i gives a probability $p_{t,\mu_i} \triangleq P_{\mu_i}(x_t = \text{rain})$ for rain. Consider the case of there being three meteorologists, and each one making the following prediction for the coming week. Start with a uniform prior $\beta(\mu) = 1/3$ for each model.

	M	T	W	T	F	S	S
CNN	0.5	0.6	0.7	0.9	0.5	0.3	0.1
SMHI	0.3	0.7	0.8	0.9	0.5	0.2	0.1
YR	0.6	0.9	0.8	0.5	0.4	0.1	0.1
Rain?	Y	Y	Y	N	Y	N	N

Table 4.3: Predictions by three different entities for the probability of rain on a particular day, along with whether or not it actually rained.

1. What is your belief about the quality of each meteorologist after each day?
2. What is your belief about the probability of rain each day?

$$P_\beta(x_t = \text{rain} \mid x_1, x_2, \dots x_{t-1}) = \sum_{\mu \in \mathcal{M}} P_\mu(x_t = \text{rain} \mid x_1, x_2, \dots x_{t-1}) \beta(\mu \mid x_1, x_2, \dots x_{t-1})$$

3. Assume you can decide whether or not to go running each day. If you go running and it does not rain, your utility is 1. If it rains, it's -10. If you don't go running, your utility is 0. What is the decision maximising utility in expectation (with respect to the posterior) each day?

4.1.3 Statistical testing*

A common type of decision problem is a statistical test. This arises when we have a set of possible candidate models $\mathcal{M}$ and we need to be able to decide which model to select after we see the evidence. Many times, there is only one model under consideration, μ_0, the so-called *null hypothesis*. Then, our only decision is whether or not to accept or reject this hypothesis.

Simple hypothesis testing

Let us start with the simple case of needing to compare two models.

The simple hypothesis test as a decision problem

- $\mathcal{M} = \{\mu_0, \mu_1\}$
- a_0: Accept model μ_0
- a_1: Accept model μ_1

U	μ_0	μ_1
a_0	1	0
a_1	0	1

Table 4.4: Example utility function for simple hypothesis tests.

There is no reason for us to be restricted to this utility function. As it is diagonal, it effectively treats both types of errors in the same way.

Example 11 (Continuation of the medium example).

- μ_1: that John is a medium.
- μ_0: that John is not a medium.

Let x_t be 0 if John makes an incorrect prediction at time t and $x_t = 1$ if he makes a correct prediction. Let us once more assume a Bernoulli model, so that John's claim that he can predict our tosses perfectly means that for a sequence of tosses $\boldsymbol{x} = x_1, \ldots, x_n$,

$$P_{\mu_1}(\boldsymbol{x}) = \begin{cases} 1, & x_t = 1 \forall t \in [n] \\ 0, & \exists t \in [n] : x_t = 0. \end{cases}$$

That is, the probability of perfectly correct predictions is 1, and that of one or more incorrect prediction is 0. For the other model, we can assume that all draws are independently and identically distributed from a fair coin. Consequently, no matter what John's predictions are, we have that:

$$P_{\mu_0}(\boldsymbol{x} = 1 \ldots 1) = 2^{-n}.$$

So, for the given example, as stated, we have the following facts:

- If John makes one or more mistakes, then $\mathbb{P}(\boldsymbol{x} \mid \mu_1) = 0$ and $\mathbb{P}(\boldsymbol{x} \mid \mu_0) = 2^{-n}$. Thus, we should perhaps say that then John is not a medium
- If John makes no mistakes at all, then

$$\mathbb{P}(\boldsymbol{x} = 1, \ldots, 1 \mid \mu_1) = 1, \qquad \mathbb{P}(\boldsymbol{x} = 1, \ldots, 1 \mid \mu_0) = 2^{-n}. \tag{4.1.2}$$

Now we can calculate the posterior distribution, which is

$$\beta(\mu_1 \mid \boldsymbol{x} = 1, \ldots, 1) = \frac{1 \times \beta(\mu_1)}{1 \times \beta(\mu_1) + 2^{-n}(1 - \beta(\mu_1))}.$$

Our expected utility for taking action a_0 is actually

$$\mathbb{E}_\beta(U \mid a_0) = 1 \times \beta(\mu_0 \mid \boldsymbol{x}) + 0 \times \beta(\mu_1 \mid \boldsymbol{x}), \qquad \mathbb{E}_\beta(U \mid a_1) = 0 \times \beta(\mu_0 \mid \boldsymbol{x}) + 1 \times \beta(\mu_1 \mid \boldsymbol{x})$$

Null hypothesis test

Many times, there is only one model under consideration, μ_0, the so-called *null hypothesis*. This happens when, for example, we have no simple way of defining an appropriate alternative. Consider the example of the medium: How should we expect a medium to predict? Then, our only decision is whether or not to accept or reject this hypothesis.

The null hypothesis test as a decision problem

- a_0: Accept model μ_0
- a_1: Reject model μ_0

Example 12. Construction of the test for the medium

- μ_0 is simply the $\mathcal{Bernoulli}(1/2)$ model: responses are by chance.
- We need to design a policy $\pi(a \mid \boldsymbol{x})$ that accepts or rejects depending on the data.
- Since there is no alternative model, we can only construct this policy according to its properties when μ_0 is true.
- In particular, we can fix a policy that only chooses a_1 when μ_0 is true a proportion δ of the time.
- This can be done by construcing a threshold test from the inverse-CDF.

Using p-values to construct statistical tests

Definition 4.1.2 (Null statistical test)**.** A statistical test π is a decision rule for accepting or rejecting a hypothesis on the basis of evidence. A p-value test rejects a hypothesis whenever the value of the statistic $f(x)$ is smaller than a threshold. The statistic $f : \mathcal{X} \to [0, 1]$ is designed to have the property:

$$P_{\mu_0}(\{x \mid f(x) \leq \delta\}) = \delta.$$

If our decision rule is:

$$\pi(a \mid x) = \begin{cases} a_0, & f(x) \geq \delta \\ a_1, & f(x) < \delta, \end{cases}$$

the probability of rejecting the null hypothesis when it is true is exactly δ.

This is because, by definition, $f(x)$ has a uniform distribution under μ_0. Hence the value of $f(x)$ itself is uninformative: high and low values are equally likely. In theory we should simply choose δ before seeing the data and just accept or reject based on whether $f(x) \leq \delta$. However nobody does that in practice, meaning that p-values are used incorrectly. Better not to use them at all, if uncertain about their meaning.

Issues with p-values

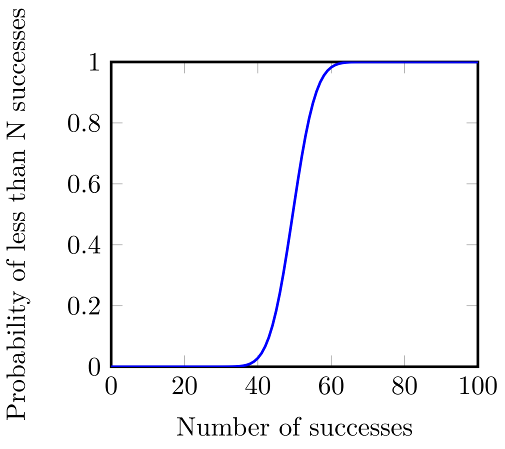
- They only measure quality of fit *on the data.*
- Not robust to model misspecification. For example, zero-mean testing using the χ^2-test has a normality assumption.
- They ignore effect sizes. For example, a linear analysis may determine that there is a significant deviation from zero-mean, but with only a small effect size of 0.01. Thus, reporting only the p-value is misleading
- They do not consider prior information.
- They do not represent the probability of having made an error. In particular, a p-value of δ does not mean that the probability that the null hypothesis is false given the data x, is δ, i.e. $\delta \neq \mathbb{P}(\neg\mu_0 \mid x)$.
- The null-rejection error probability is the same irrespective of the amount of data (by design).

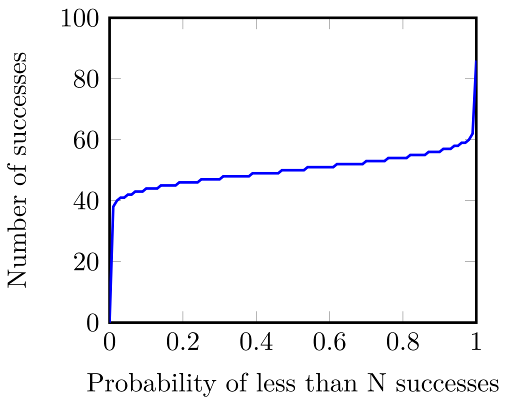
Let us consider the example of the medium.

p-values for the medium example

- μ_0 is simply the $\mathcal{Bernoulli}(1/2)$ model: responses are by chance.

- CDF: $P_{\mu_0}(N \leq n \mid K = 100)$ is the probability of at most N successes if we throw the coin 100 times. This is in fact the cumulative probability function of the binomial distribution. Recall that the binomial represents the distribution for the number of successes of independent experiments, each following a Bernoulli distribution.
- ICDF: the number of successes that will happen with probability at least δ
- e.g. we'll get at most 50 successes a proportion $\delta = 1/2$ of the time.
- Using the (inverse) CDF we can construct a policy π that selects a_1 when μ_0 is true only a δ portion of the time, for any choice of δ.





Constructing a Null-Hypothesis test with frequentist properties

The test statistic

We want the test to reflect that we don't have a significant number of failures.

$$f(x) = 1 - \text{binocdf}(\sum_{t=1}^{n} x_t, n, 0.5)$$

⚠ **What $f(x)$ is and is not**

- It is a **statistic** which is $\leq \delta$ a δ portion of the time when μ_0 is true.
- It is **not** the probability of observing x under μ_0.
- It is **not** the probability of μ_0 given x.

Exercise 9.
- Let us throw a coin 8 times, and try and predict the outcome.
- Select a p-value threshold so that $\delta = 0.05$. For 8 throws, this corresponds to > 6 successes or $\geq 87.5\%$ success rate.
- Let's calculate the p-value for each one of you
- What is the rejection performance of the test?

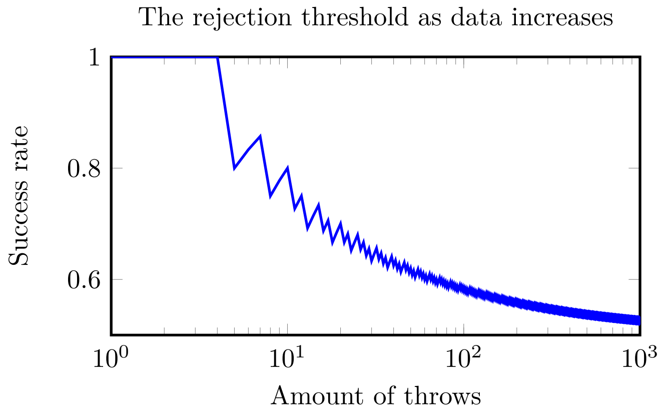


Figure 4.1: Here we see how the rejection threshold, in terms of the success rate, changes with the number of throws to achieve an error rate of $\delta = 0.05$.

As the amount of throws goes to infinity, the threshold converges to 0.5. This means that a statistically significant difference from the null hypothesis can be obtained, even when the actual model from which the data is drawn is only slightly different from 0.5.

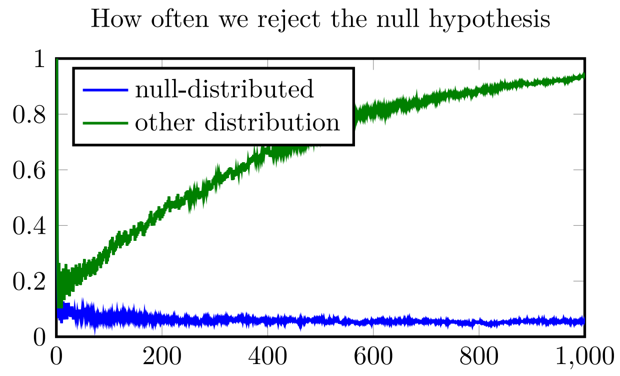


Figure 4.2: Here we see the rejection rate of the null hypothesis (μ_0) for two cases. Firstly, for the case when μ_0 is true. Secondly, when the data is generated from $\mathcal{Bernoulli}(0.55)$.

As we see, this method keeps its promise: the null is only rejected 0.05 of the time when it's true. We can also examine how often the null is rejected when it is false... but what should we compare against? Here we are generating data from a $\mathcal{Bernoulli}(0.55)$ model, and we can see the rejection of the null increases with the amount of data. This is called the *power* of the test with respect to the $\mathcal{Bernoulli}(0.55)$ distribution.

⚠ **Statistical power and false discovery.**

Beyond not rejecting the null when it's true, we also want:

- High power: Rejecting the null when it is false.

- Low false discovery rate: Accepting the null when it is true.

Power

The power depends on what hypothesis we use as an alternative. This implies that we cannot simply consider a plain null hypothesis test, but must formulate a specific alternative hypothesis.

False discovery rate

False discovery depends on how likely it is *a priori* that the null is false. This implies that we need to consider a prior probability for the null hypothesis being true.

Both of these problems suggest that a Bayesian approach might be more suitable. Firstly, it allows us to consider an infinite number of possible alternative models as the alternative hypothesis, through Bayesian model averaging. Secondly, it allows us to specify prior probabilities for each alternative. This is especially important when we consider some effects unlikely.

The Bayesian Null-Hypothesis test

Example 13.

1. Set $U(a_i, \mu_j) = \mathbb{I}\{i = j\}$. This choice makes sense if we care equally about either type of error.
2. Set $\beta(\mu_i) = 1/2$. Here we place an equal probability in both models.
3. μ_0: $\mathcal{Bernoulli}(1/2)$. This is the same as the null hypothesis test.
4. μ_1: $\mathcal{Bernoulli}(\theta)$, $\theta \sim \mathcal{Unif}([0,1])$. This is an extension of the simple hypothesis test, with an alternative hypothesis that says "the data comes from an arbitrary Bernoulli model".
5. Calculate $\beta(\mu \mid x)$.
6. Choose a_i, where $i = \arg\max_j \beta(\mu_j \mid x)$.

Bayesian model averaging for the alternative model μ_1

In this scenario, μ_0 is a simple point model, e.g. corresponding to a $\mathcal{Bernoulli}(1/2)$. However μ_1 is a marginal distribution integrated over many models, e.g. a *Beta* distribution over Bernoulli parameters.

$$P_{\mu_1}(x) = \int_\Theta B_\theta(x) \, \mathrm{d}\beta(\theta) \tag{4.1.3}$$

$$\beta(\mu_0 \mid x) = \frac{P_{\mu_0}(x)\beta(\mu_0)}{P_{\mu_0}(x)\beta(\mu_0) + P_{\mu_1}(x)\beta(\mu_1)} \tag{4.1.4}$$

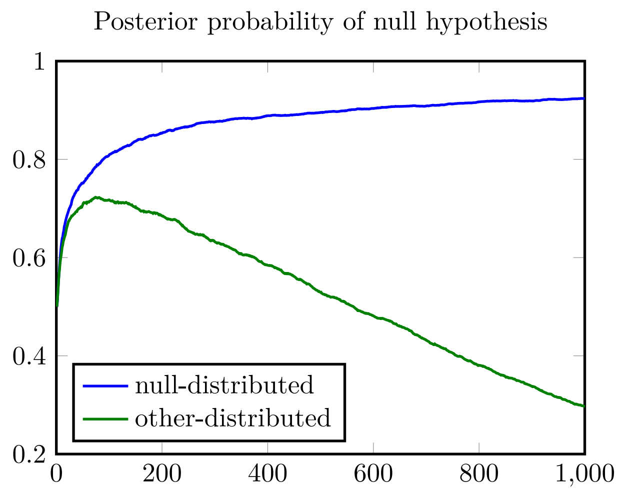


Figure 4.3: Here we see the convergence of the posterior probability.

As can be seen in the figure above, in both cases, the posterior converges to the correct value, so it can be used to indicate our confidence that the null is true.

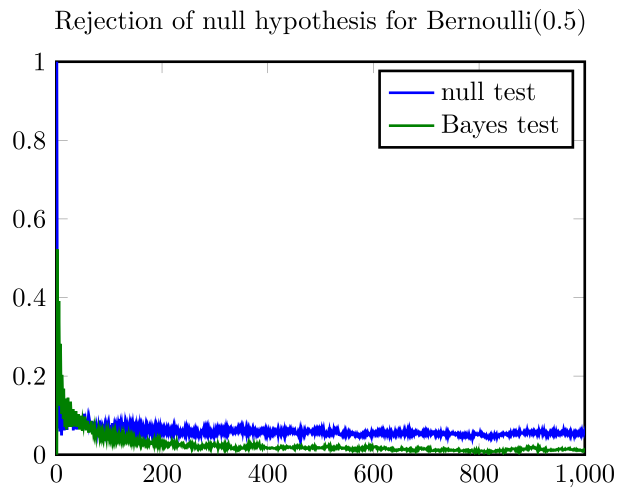


Figure 4.4: Comparison of the rejection probability for the null and the Bayesian test when μ_0 is true.

Now we can use this Bayesian test, with uniform prior, to see how well it performs. While the plain null hypothesis test has a fixed rejection rate of 0.05, the Bayesian test's rejection rate converges to 0 as we collect more data.

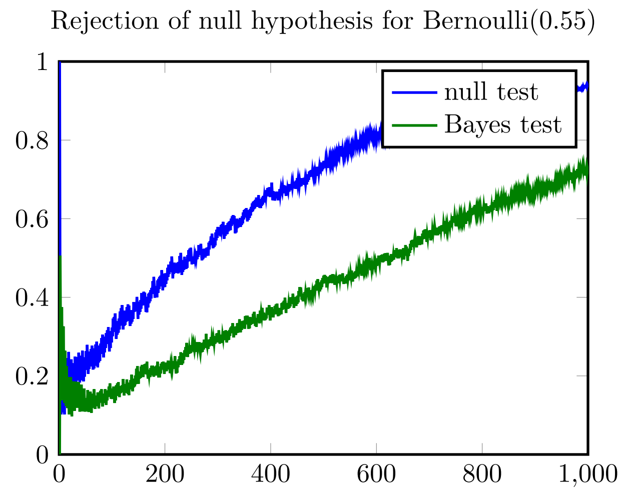


Figure 4.5: Comparison of the rejection probability for the null and the Bayesian test when μ_1 is true.

However, both methods are able to reject the null hypothesis more often when it is false, as long as we have more data.

Concentration inequalities and confidence intervals

There are a number of inequalities from probability theory that allow us to construct high-probability confidence intervals. The most well-known of those is Hoeffding's inequality , the simplest version of which is the following:

Lemma 4.1.1 (Hoeffding's inequality)**.** *Let $x_1, \ldots, x_n$ be a series of random variables, $x_i \in [0, 1]$. Then it holds that, for the empirical mean:*

$$\mu_n \triangleq \frac{1}{n} sum_{t=1}^n x_t$$

$$\mathbb{P}(\mu_n \geq \mathbb{E}\,\mu_n + \epsilon) \leq e^{-2n\epsilon^2}. \tag{4.1.5}$$

When x_t are i.i.d, $\mathbb{E}\,\mu_n = \mathbb{E}\,x_t$. This allows us to construct an interval of size ϵ around the true mean. This can generalise to a two-sided interval:

$$\mathbb{P}(|\mu_n - \mathbb{E}\,\mu_n| \geq \epsilon) \leq 2e^{-2n\epsilon^2}.$$

We can also rewrite the equation to say that, with probability at least $1 - \delta$

$$|\mu_n - \mathbb{E}\,\mu_n| \leq \sqrt{\frac{\ln 2/\delta}{2n}}$$

Further reading

Points of significance (Nature Methods)

- Importance of being uncertain `https://www.nature.com/articles/nmeth.2613`
- Error bars `https://www.nature.com/articles/nmeth.2659`
- P values and the search for significance `https://www.nature.com/articles/nmeth.4120`
- Bayes' theorem `https://www.nature.com/articles/nmeth.3335`
- Sampling distributions and the bootstrap `https://www.nature.com/articles/nmeth.3414`

4.2 Formalising classification problems

One of the simplest decision problems is classification. At the simplest level, this is the problem of observing some data point $x_t \in \mathcal{X}$ and making a decision about what class $\mathcal{Y}$ it belongs to. Typically, a fixed classifier is defined as a decision rule $\pi(a|x)$ making decisions $a \in \mathcal{A}$, where the decision space includes the class labels, so that if we observe some point x_t and choose $a_t = 1$, we essentially declare that $y_t = 1$.

Typically, we wish to have a classification policy that minimises classification error.

Deciding a class given a model

In the simplest classification problem, we observe some features x_t and want to make a guess a_t about the true class label y_t. Assuming we have some probabilistic model $P_\mu(y_t \mid x_t)$, we want to define a decision rule $\pi(a_t \mid x_t)$ that is optimal, in the sense that it maximises expected utility for P_μ.

- Features $x_t \in \mathcal{X}$.
- Label $y_t \in \mathcal{Y}$.
- Decisions $a_t \in \mathcal{A}$.
- Decision rule $\pi(a_t \mid x_t)$ assigns probabilities to actions.

Standard classification problem

In the simplest case, the set of decisions we make are the same as the set of classes

$$\mathcal{A} = \mathcal{Y}, \qquad U(a, y) = \mathbb{I}\{a = y\}$$

Exercise 10. If we have a model $P_\mu(y_t \mid x_t)$, and a suitable U, what is the optimal decision to make?

Deciding the class given a model family

- Training data $D_T = \{(x_i, y_i) \mid i = 1, \ldots, T\}$
- Models $\{P_\mu \mid \mu \in \mathcal{M}\}$.
- Prior β on $\mathcal{M}$.

Similarly to our example with the meteorological stations, we can define a posterior distribution over models.

Posterior over classification models

$$\beta(\mu \mid D_T) = \frac{P_\mu(y_1, \ldots, y_T \mid x_1, \ldots, x_T)\beta(\mu)}{\sum_{\mu' \in \mathcal{M}} P_{\mu'}(y_1, \ldots, y_T \mid x_1, \ldots, x_T)\beta(\mu')}$$

This posterior form can be seen as weighing each model according to how well they can predict the class labels. It is a correct form as long as, for every pair of models μ, μ' we have that $P_\mu(x_1, \ldots, x_T) = P_{\mu'}(x_1, \ldots, x_T)$. This assumption can be easily satisfied without specifying a particular model for the x. If not dealing with time-series data, we assume independence between x_t:

$$P_\mu(y_1, \ldots, y_T \mid x_1, \ldots, x_T) = \prod_{i=1}^{T} P_\mu(y_i \mid x_i)$$

The *Bayes rule* for maximising $\mathbb{E}_\beta(U \mid a, x_t, D_T)$

The decision rule simply chooses the action:

$$a_t \in \arg\max_{a \in \mathcal{A}} \sum_y \sum_{\mu \in \mathcal{M}} P_\mu(y_t = y \mid x_t)\beta(\mu \mid D_T)U(a, y) \tag{4.2.1}$$

$$= \arg\max_{a \in \mathcal{A}} \sum_y \mathbb{P}_{\beta \mid D_T}(y_t = y \mid x_t)U(a, y) \tag{4.2.2}$$

We can rewrite this by calculating the posterior marginal marginal label probability

$$\mathbb{P}_{\beta \mid D_T}(y_t \mid x_t) \triangleq \mathbb{P}_\beta(y_t \mid x_t, D_T) = \sum_{\mu \in \mathcal{M}} P_\mu(y_t \mid x_t)\beta(\mu \mid D_T).$$

Approximating the model

Full Bayesian approach for infinite $\mathcal{M}$

Here β can be a probability density function and

$$\beta(\mu \mid D_T) = P_\mu(D_T)\beta(\mu) / \mathbb{P}_\beta(D_T), \qquad \mathbb{P}_\beta(D_T) = \int_{\mathcal{M}} P_\mu(D_T)\beta(\mu)\,\mathrm{d},$$

can be hard to calculate.

Maximum a posteriori model

We only choose a single model through the following optimisation:

$$\mu_{\mathrm{MAP}}(\beta, D_T) = \arg\max_{\mu \in \mathcal{M}} P_\mu(D_T)\beta(\mu) = \arg\max_{\mu \in \mathcal{M}} \overbrace{\ln P_\mu(D_T)}^{\text{goodness of fit}} + \underbrace{\ln \beta(\mu)}_{\text{regulariser}}.$$

You can think of the goodness of fit as how well the model fits the training data, while the regulariser term simply weighs models according to some criterion. Typically, lower weights are used for more complex models.

Learning outcomes

Understanding

- Preferences, utilities and the expected utility principle.
- Hypothesis testing and classification as decision problems.
- How to interpret p-values Bayesian tests.
- The MAP approximation to full Bayesian inference.

Skills

- Being able to implement an optimal decision rule for a given utility and probability.
- Being able to construct a simple null hypothesis test.

Reflection

- When would expected utility maximisation not be a good idea?
- What does a p value represent when you see it in a paper?
- Can we prevent high false discovery rates when using p values?
- When is the MAP approximation good?

4.3 Beliefs and probabilities

Probability can be used to describe purely chance events, as in for example quantum physics. However, it is mostly used to describe uncertain events, such as the outcome of a dice roll or a coin flip, which only appear random. In fact, one can take it even further than that, and use it to model subjective uncertainty about any arbitrary event. Although probabilities are not the only way in which we can quantify uncertainty, it is a simple enough model, and with a rich enough history in mathematics, statistics, computer science and engineering that it is the most useful.

Uncertainty

Axioms of probability

Let Ω be the certain event, and Σ is an appropriate σ-algebra on Ω. A probability measure P on (Ω, Σ) has the following properties:

1. The probability of the certain event is $P(\Omega) = 1$
2. The probability of the impossible event is $P(\emptyset) = 0$

3. The probability of any event $A \in \Sigma$ is $0 \leq P(A) \leq 1$.
4. If A, B are disjoint, i.e. $A \cap B = \emptyset$, meaning that they cannot happen at the same time, then
$$P(A \cup B) = P(A) + P(B)$$

Sometimes we would like to calculate the probability of some event A happening given that we know that some other event B has happened. For this we need to first define the idea of conditional probability.

Definition 4.3.1 (Conditional probability)**.** The probability of A happening if we know that B has happened is defined to be:
$$P(A \mid B) \triangleq \frac{P(A \cap B)}{P(B)}.$$

Conditional probabilities obey the same rules as probabilities. Here, the probability measure of any event A given B is defined to be the probability of the intersection of of the events divided by the second event. We can rewrite this definition as follows, by using the definition for $P(B \mid A)$

Bayes's theorem

For $P(A_1 \cup A_2) = 1$, $A_1 \cap A_2 = \emptyset$,
$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{P(B)} = \frac{P(B \mid A_i)P(A_i)}{P(B \mid A_1)P(A_1) + P(B \mid A_2)P(A_2)}$$

Example 14 (probability of rain). What is the probability of rain given a forecast x_1 or x_2?

ω_1: rain	$P(\omega_1) = 80\%$
ω_2: dry	$P(\omega_2) = 20\%$

Table 4.5: Prior probability of rain tomorrow

x_1: rain	$P(x_1 \mid \omega_1) = 90\%$
x_2: dry	$P(x_2 \mid \omega_2) = 50\%$

Table 4.6: Probability the forecast is correct

$$P(\omega_1 \mid x_1) = 87.8\%$$
$$P(\omega_1 \mid x_2) = 44.4\%$$

Table 4.7: Probability that it will rain given the forecast

Classification in terms of conditional probabilities

Conditional probability naturally appears in classification problems. Given a new example vector of data $x_t \in \mathcal{X}$, we would like to calculate the probability of different classes $c \in \mathcal{Y}$ given the data, $P_\mu(y_t = c \mid x_t)$. If we somehow obtained the distribution of data $P_\mu(x_t \mid y_t)$ for each possible class, as well as the prior class probability $P_\mu(y_t = c)$, from Bayes's theorem, we see that we can obtain the probability of the class:

$$P_\mu(y_t = c \mid x_t) = \frac{P_\mu(x_t \mid y_t = c) P_\mu(y_t = c)}{\sum_{c' \in \mathcal{Y}} P_\mu(x_t \mid y_t = c') P_\mu(y_t = c')}$$

for any class c. This directly gives us a method for classifying new data, as long as we have a way to obtain $P_\mu(x_t \mid y_t)$ and $P_\mu(y_t)$.

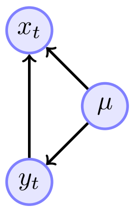
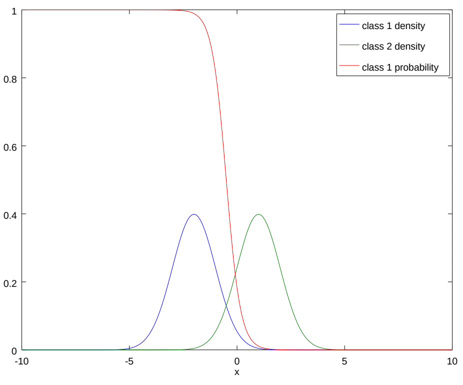
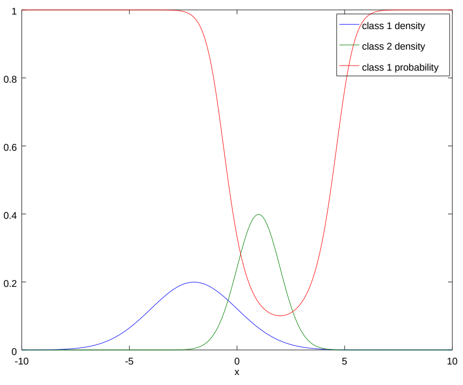


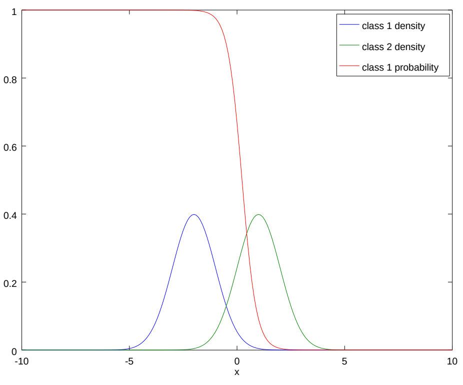
Figure 4.6: A generative classification model. μ identifies the model (paramter). x_t are the features and y_t the class label of the t-th example.



(a) Equal prior and variance



(b) Unequal variance



(c) Unequal prior

Figure 4.7: The effect of changing variance and prior on the classification decision when we assume a normal distribution.

Example 15 (Normal distribution). A simple example is when x_t is normally distributed in a matter that depends on the class. Figure 4.7 shows the distribution of x_t for two different classes, with means of -1 and $+1$ respectively, for three different case. In the first case, both classes have variance of 1, and we assume the same prior probability for both

$$x_t \mid y_t = 0 \sim \mathcal{N}(-1, 1), \qquad x_t \mid y_t = 1 \sim \mathcal{N}(1, 1)$$

$$x_t \mid y_t = 0 \sim \mathcal{N}(-1, 1), \qquad x_t \mid y_t = 1 \sim \mathcal{N}(1, 1)$$

But how can we get a probability model in the first place?

Subjective probability

While probabilities apply to truly random events, they are also useful for representing subjective uncertainty. In this course, we will use a special symbol for subjective probability, β.

Subjective probability measure β

- If we think event A is more likely than B, then $\beta(A) > \beta(B)$.
- Usual rules of probability apply:
  1. $\beta(A) \in [0, 1]$.
  2. $\beta(\emptyset) = 0$.
  3. If $A \cap B = \emptyset$, then $\beta(A \cup B) = \beta(A) + \beta(B)$.

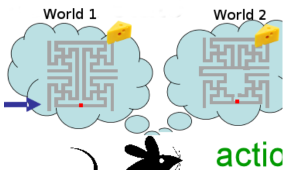
Bayesian inference illustration

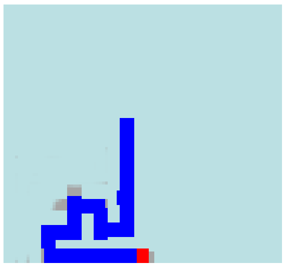
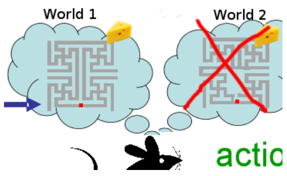
Here is a simple example to illustrate the idea of Bayesian inference. Imagine you are a rat in the maze and you need to get to the cheese. In the example below, you believe that the maze lay out is only one of two possibilities μ_1, μ_2. Initially you might believe that the first model is more likely than the second, so you decide upon some function β so that $\beta(\mu_1) > \beta(\mu_2)$, for example $\beta(\mu_1) = 2/3$, $\beta(\mu_2) = 1/3$. Now you start acting in the maze, collecting data h consisting of your history of observation in this maze. The data gives a different amount of *evidence* for each model of the world, which we can write as $P_\mu(h)$. In the end, we want to combine this evidence with our initial belief to arrive at a conclusion. In the Bayesian setting, all of these quantities are probabilities, and the conclusion is simply the conditional probability of the model given the data.

Use a subjective belief $\beta(\mu)$ on $\mathcal{M}$

- *Prior* belief $\beta(\mu)$ represents our initial uncertainty.
- We *observe history h*.
- Each possible μ assigns a *probability* $P_\mu(h)$ to h.
- We can use this to *update* our belief via Bayes' theorem to obtain the *posterior* belief:

$$\beta(\mu \mid h) \propto P_\mu(h)\beta(\mu) \qquad \text{(conclusion = evidence} \times \text{prior)}$$



prior

evidence


conclusion

4.3.1 Probability and Bayesian inference

One of the most important methods in machine learning and statistics is that of Bayesian inference. This is the most fundamental method of drawing conclusions from data and explicit prior assumptions. In Bayesian inference, prior assumptions are represented as a probabilities on a space of hypotheses. Each hypothesis is seen as a probabilistic model of all possible data that we can see.

Frequently, we want to draw conclusions from data. However, the conclusions are never solely inferred from data, but also depend on prior assumptions about reality.

Some examples

Example 16. John claims to be a medium. He throws a coin n times and predicts its value always correctly. Should we believe that he is a medium?

- μ_1: John is a medium.
- μ_0: John is not a medium.

The answer depends on what we *expect* a medium to be able to do, and how likely we thought he'd be a medium in the first place.

Example 17. Traces of DNA are found at a murder scene. We perform a DNA test against a database of 10^4 citizens registered to be living in the area. We know that the probability of a false positive (that is, the test finding a match by mistake) is 10^{-6}. If there is a match in the database, does that mean that the citizen was at the scene of the crime?

Bayesian inference

Now let us apply this idea to our specific problem. We already have the probability of the observation for each model, but we just need to define a *prior probability* for each model. Since this is usually completely subjective, we give it another symbol.

Prior probability

The prior probability β on a set of models $\mathcal{M}$ specifies our subjective belief $\beta(\mu)$ that each model is true.[a]

[a]More generally β is a probability measure.

This allows us to calculate the probability of John being a medium, given the data:

$$\beta(\mu_1 \mid \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{x} \mid \mu_1)\beta(\mu_1)}{\mathbb{P}_\beta(\boldsymbol{x})},$$

where

$$\mathbb{P}_\beta(\boldsymbol{x}) \triangleq \mathbb{P}(\boldsymbol{x} \mid \mu_1)\beta(\mu_1) + \mathbb{P}(\boldsymbol{x} \mid \mu_0)\beta(\mu_0).$$

The only thing left to specify is $\beta(\mu_1)$, the probability that John is a medium before seeing the data. This is our subjective prior belief that mediums exist and that John is one of them. More generally, we can think of Bayesian inference as follows:

- We start with a set of mutually exclusive models $\mathcal{M} = \{\mu_1, \ldots, \mu_k\}$.
- Each model μ is represented by a specific probabilistic model for any possible data x, that is $P_\mu(x) \equiv \mathbb{P}(x \mid \mu)$.
- For each model, we have a prior probability $\beta(\mu)$ that it is correct.
- After observing the data, we can calculate a posterior probability that the model is correct:

$$\beta(\mu \mid x) = \frac{\mathbb{P}(x \mid \mu)\beta(\mu)}{\sum_{\mu' \in \mathcal{M}} \mathbb{P}(x \mid \mu')\beta(\mu')} = \frac{P_\mu(x)\beta(\mu)}{\sum_{\mu' \in \mathcal{M}} P_{\mu'}(x)\beta(\mu')}.$$

Interpretation

- $\mathcal{M}$: Set of all possible models that could describe the data.
- $P_\mu(x)$: Probability of x under model μ.
- Alternative notation $\mathbb{P}(x \mid \mu)$: Probability of x given that model μ is correct.
- $\beta(\mu)$: Our belief, before seeing the data, that μ is correct.
- $\beta(\mu \mid x)$: Our belief, aftering seeing the data, that μ is correct.

It must be emphasized that $P_\mu(x) = \mathbb{P}(x \mid \mu)$ as they are simply two different notations for the same thing. In words the first can be seen as the probability that model μ assigns to data x, while the second as the probability of x if μ is the true model. Combining the prior belief with evidence is key in this procedure. Our posterior belief can then be used as a new prior belief when we get more evidence.

EXERCISE 11 (Continued example for medium). Now let us apply this idea to our specific problem. We first make an independence assumption. In particular, we can assume that success and failure comes from a Bernoulli distribution with a parameter depending on the model.

$$P_\mu(x) = \prod_{t=1}^{n} P_\mu(x_t). \qquad \text{(independence property)}$$

We first need to specify how well a medium could predict. Let's assume that a true medium would be able to predict perfectly, and that a non-medium would only predict randomly. This leads to the following models:

$$P_{\mu_1}(x_t = 1) = 1, \qquad P_{\mu_1}(x_t = 0) = 0. \qquad \text{(true medium model)}$$
$$P_{\mu_0}(x_t = 1) = 1/2, \qquad P_{\mu_0}(x_t = 0) = 1/2. \qquad \text{(non-medium model)}$$

The only thing left to specify is $\beta(\mu_1)$, the probability that John is a medium before seeing the data. This is our subjective prior belief that mediums exist and that John is one of them.

$$\beta(\mu_0) = 1/2, \qquad \beta(\mu_1) = 1/2. \qquad \text{(prior belief)}$$

Combining the prior belief with evidence is key in this procedure. Our posterior belief can then be used as a new prior belief when we get more evidence.

$$\beta(\mu_1 \mid x) = \frac{P_{\mu_1}(x)\beta(\mu_1)}{\mathbb{P}_\beta(x)} \qquad \text{(posterior belief)}$$

$$\mathbb{P}_\beta(x) \triangleq P_{\mu_1}(x)\beta(\mu_1) + P_{\mu_0}(x)\beta(\mu_0). \qquad \text{(marginal distribution)}$$

If a classmate correctly predicts 4 coin tosses, what is your belief they are a medium?

Sequential update of beliefs

Assume you have n meteorologists. At each day t, each meteorologist i gives a probability $p_{t,\mu_i} \triangleq P_{\mu_i}(x_t = \text{rain})$ for rain. Consider the case of there being three meteorologists, and each one making the following prediction for the coming week. Start with a uniform prior $\beta(\mu) = 1/3$ for each model.

	M	T	W	T	F	S	S
CNN	0.5	0.6	0.7	0.9	0.5	0.3	0.1
SMHI	0.3	0.7	0.8	0.9	0.5	0.2	0.1
YR	0.6	0.9	0.8	0.5	0.4	0.1	0.1
Rain?	Y	Y	Y	N	Y	N	N

Table 4.8: Predictions by three different entities for the probability of rain on a particular day, along with whether or not it actually rained.

Exercise 12.
- n meteorological stations $\{\mu_i \mid i = 1, \ldots, n\}$
- The i-th station predicts rain $P_{\mu_i}(x_t \mid x_1, \ldots, x_{t-1})$.
- Let $\beta_t(\mu)$ be our belief at time t. Derive the next-step belief $\beta_{t+1}(\mu) \triangleq \beta_t(\mu|x_t)$ in terms of the current belief β_t.
- Write a python function that computes this posterior

$$\beta_{t+1}(\mu) \triangleq \beta_t(\mu|x_t) = \frac{P_\mu(x_t \mid x_1, \ldots, x_{t-1})\beta_t(\mu)}{\sum_{\mu'} P_{\mu'}(x_t \mid x_1, \ldots, x_{t-1})\beta_t(\mu')}$$

Bayesian inference for Bernoulli distributions

Estimating a coin's bias

A fair coin comes heads 50% of the time. We want to test an unknown coin, which we think may not be completely fair.

For a sequence of throws $x_t \in \{0, 1\}$,

$$P_\theta(x) \propto \prod_t \theta^{x_t}(1-\theta)^{1-x_t} = \theta^{\#\text{Heads}}(1-\theta)^{\#\text{Tails}}$$

Say we throw the coin 100 times and obtain 70 heads. Then we plot the *likelihood* $P_\theta(x)$ of different models. From these, we calculate a *posterior* distribution over the correct models.

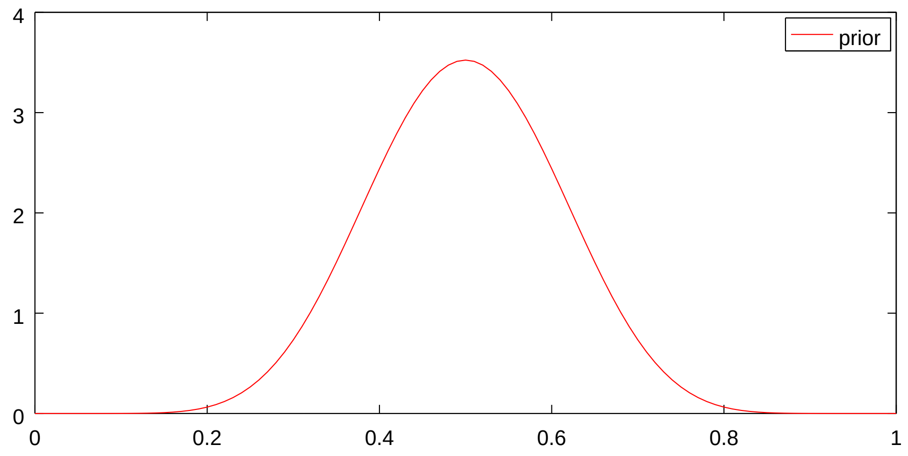


Figure 4.8: Prior belief β about the coin bias θ.

This represents our conclusion given our prior and the data. If the prior distribution is described by the so-called Beta density

$$f(\theta \mid \alpha, \beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

where α, β describe the shape of the Beta distribution.

Riemann and Lebesgue integrals

Since β is a probability measure over models Θ, it is always simple to write the posterior probability given some data x in the following terms when Θ is discrete (finite or countable):

$$\beta(\theta \mid x) = \frac{P_\theta(x)\beta(\theta)}{\sum_{\theta'} P_{\theta'(x)\beta(\theta')}}.$$

However, in many situations Θ is uncountable, i.e. $\Theta \subset \mathbb{R}^k$. Then, as both the prior β and the posterior $\beta(\cdot \mid x)$ have to be probability measures on Θ, they are defined over subsets of Θ. This means that we can write the posterior in terms of Lebesgue integrals:

$$\beta(B \mid x) = \frac{\int_B P_\theta(x) \, \mathrm{d}\beta(\theta)}{\int_\Theta P_\theta(x) \, \mathrm{d}\beta(\theta)}.$$

Alternatively, we can abuse notation and use $\beta(\theta)$ to describe a density, so that the *posterior density* is written in terms of a Riemann integral: *posterior density*

$$\beta(\theta \mid x) \triangleq \frac{P_\theta(x)\beta(\theta) \, \mathrm{d}\theta}{\int_\Theta P_{\theta'}(x)\beta(\theta')\theta'}.$$

However, the advantage of the Lebesgue integral notation is that it works the same no matter whether Θ is discrete or continuous.

Learning outcomes

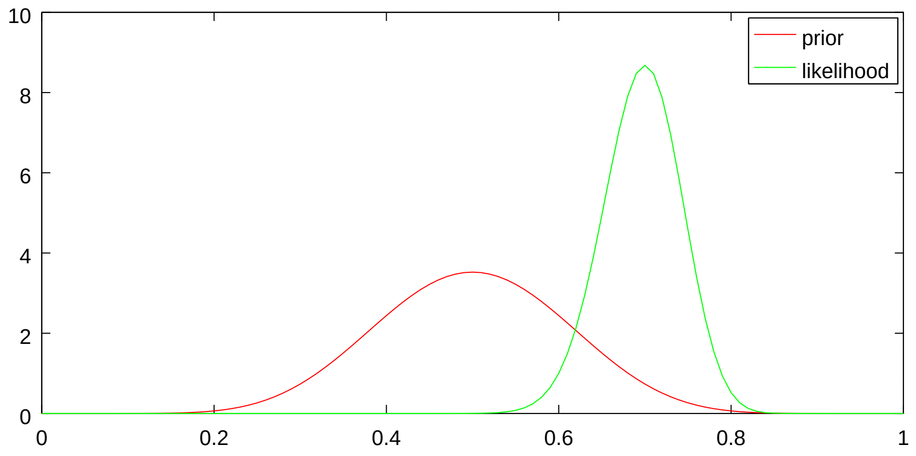


Figure 4.9: Prior belief β about the coin bias θ and likelihood of θ for the data.

Understanding

- The axioms of probability, marginals and conditional distributions.
- The philosophical underpinnings of Bayesianism.
- The simple conjugate model for Bernoulli distributions.

Skills

- Be able to calculate with probabilities using the marginal and conditional definitions and Bayes rule.
- Being able to implement a simple Bayesian inference algorithm in Python.

Reflection

- How useful is the Bayesian representation of uncertainty?
- How restrictive is the need to select a prior distribution?
- Can you think of another way to explicitly represent uncertainty in a way that can incorporate new evidence?

4.4 Classification with stochastic gradient descent*

Classification as an optimisation problem.

Finding the optimal policy for our belief β is not normally very difficult. However, it requires that we maintain the complete distribution β and the set of models $P_\mu(y \mid x)$. In simple decision problems, e.g. where the set of actions $\mathcal{A}$ is finite, it is possible to do this calculation on-the-fly. However, in some cases we might not have a model.

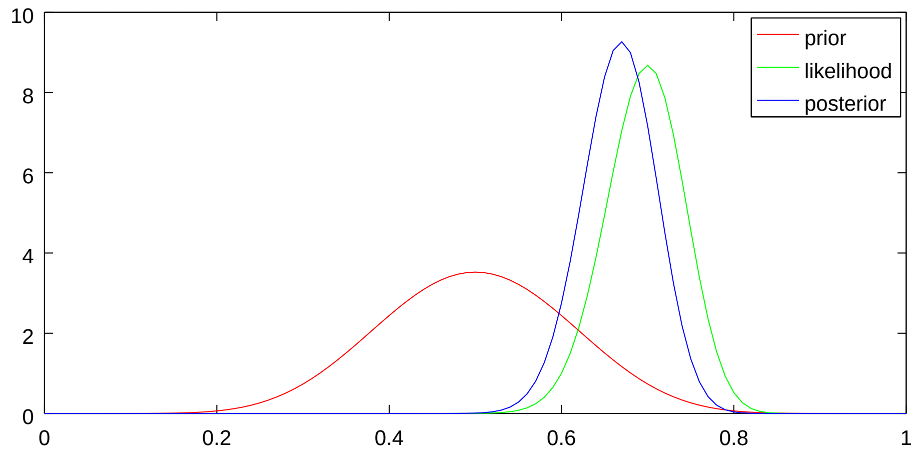


Figure 4.10: Prior belief $\beta(\theta)$ about the coin bias θ, likelihood of θ for the data, and posterior belief $\beta(\theta \mid x)$

Recall that we wish to maximise expected utility for some policy under some distribution. In general, this has the form

$$\max_{\pi} \mathbb{E}^{\pi}_{\mu}(U).$$

We also know that any expectation can be approximated by sampling. Let $P_\mu(\omega)$ be the distribution on outcomes defined by our model. Then

$$\mathbb{E}^{\pi}_{\mu}(U) = \sum_{\omega} U(a, \omega) P_\mu(\omega) \approx T^{-1} \sum_{t=1}^{T} U(a, \omega_t), \qquad \omega_t \sim P_\mu(\omega),$$

i.e. when we can replace the explicit summation over all possible outcomes, weighed by their probability through averaging over T outcomes sampled from the correct distribution. In fact this approximation is *unbiased*, as its expectation is equal to the expected utility.

The μ-optimal classifier

Since the performance measure is simply an expectation, it is intuitive to directly optimise the decision rule with respect to an approximation of the expectation

$$\max_{\theta \in \Theta} f(\pi_\theta, \mu, U), \qquad f(\pi_\theta, \mu, U) \triangleq \mathbb{E}^{\pi_\theta}_{\mu}(U) \tag{4.4.1}$$

$$f(\pi_\theta, \mu, U) = \sum_{x,y,a} U(a, y) \pi_\theta(a \mid x) P_\mu(y \mid x) P_\mu(x) \tag{4.4.2}$$

$$\approx \sum_{t=1}^{T} \sum_{a_t} U(a_t, y_t) \pi_\theta(a_t \mid x_t), \qquad (x_t, y_t)_{t=1}^{T} \sim P_\mu. \tag{4.4.3}$$

In practice, this is the empirical expectation on the training set $\{(x_t, y_t) \mid t = 1, \ldots, T\}$. However, when the amount of data is insufficient, this expectation may be far from reality, and so our classification rule might be far from optimal.

The Bayes-optimal classifier

An alternative idea is to use our uncertainty to create a distribution over models, and then use this distribution to obtain a single classifier that does take the uncertainty into account.

$$\max_{\theta \in \Theta} f(\pi_\theta \beta) \approx \max_{\theta \in \Theta} N^{-1} \sum_{n=1}^{N} \pi_\theta(a_t = y_n \mid x_t = x_n), \qquad (x_n, y_n) \sim P_{\mu_n}, \mu_n \sim \beta.$$

In this case, the integrals are replaced by sampling models μ_n from the belief, and then sampling (x_n, y_n) pairs from P_{μ_n}.

Stochastic gradient methods

To find the maximum of a differentiable function g, we can use gradient descent

Gradient ascent

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta g(\theta_i).$$

When f is an expectation, we don't need to calculate the full gradient. In fact, we only need to take one sample from the related distribution.

Stochastic gradient ascent

$$g(\theta) = \int_{\mathcal{M}} f(\theta, \mu) \, \mathrm{d}\beta(\mu)$$
$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta f(\theta_i, \mu_i), \qquad \mu_i \sim \beta.$$

Stochastic gradient methods are commonly employed in neural networks.

4.4.1 Neural network models

Two views of neural networks

In the simplest sense a neural network is simply as parametrised functions f_θ. In classification, neural networks can be used as probabilistic models, so they describes the probability $P_\theta(y|\boldsymbol{x})$, or as classification policies so that $f_\theta(x, a)$ describes the probability $\pi_\theta(a \mid x)$ of selecting class label a. Let us begin by describing the simplest type of neural network model, the perceptron.

Neural network classification model $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}_t)$

$$\begin{tikzcd} \boldsymbol{x}_t \arrow[r, bend right] & \boldsymbol{y}_t \end{tikzcd}$$

Objective: Find the best model for D_T.

In this case, we consider a utility function that minimises the divergence between the network's probability predictions and the labels, such as $U(\theta, y_t, x_t) = \ln P_\theta(y_t|x_t)$.

Neural network classification policy $\pi(a_t \mid \boldsymbol{x}_t)$

$\boldsymbol{x}_t$
a_t

Objective: Find the best policy for $U(a, \boldsymbol{x})$.

In this case, we consider a utility function that considers the actual classification decisions, so that if $u(a_t, y_t)$ is how much we gain by choosing a_t when the label is y_t, then the expected utility for a decision a is $U(a, \boldsymbol{x}_t) = \sum_y \mathbb{P}(y_t = y|\boldsymbol{x}_t)u(a, y)$.

⚠ Difference between the two views

- We can use standard probabilistic methods for P.
- Finding the optimal π is an optimisation problem. However, estimating P can also be formulated as an optimisation.

Linear networks and the perceptron algorithm

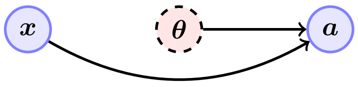


Figure 4.11: Abstract graphical model for a neural network

A neural network as used for modelling classification or regression problems, is simply a parametrised mapping $\mathcal{X} \to \mathcal{Y}$. If we include the network parameters, then it is instead a mapping $\mathcal{X} \times \Theta \to \mathcal{Y}$, as seen in Figure 4.13.

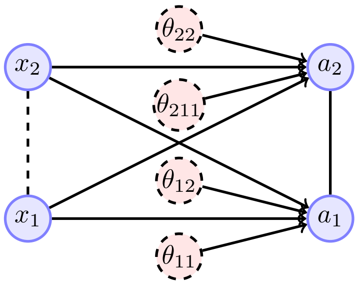


Figure 4.12: Graphical model for a linear neural network.

If we see each possible output as a different random variable, this creates a dependence. After all, we are really splitting one variable into many. In particular, if the network's output is the probability of each action, then we must make sure these sum to 1.

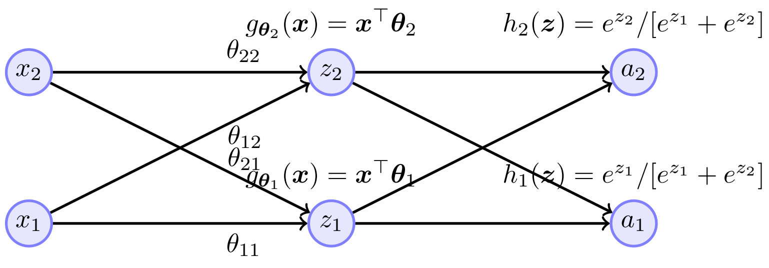

$g_{\boldsymbol{\theta}_2}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}_2$
$h_2(\boldsymbol{z}) = e^{z_2}/[e^{z_1} + e^{z_2}]$
θ_{22}
x_2
z_2
a_2
θ_{12}
θ_{21}
$g_{\boldsymbol{\theta}_1}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}_1$
$h_1(\boldsymbol{z}) = e^{z_1}/[e^{z_1} + e^{z_2}]$
x_1
z_1
a_1
θ_{11}

Figure 4.13: Architectural view of a linear neural network.

Definition 4.4.1 (Linear classifier)**.** A linear classifier with N inputs and C outputs is parametrised by

$$\boldsymbol{\Theta} = \begin{bmatrix}\boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_C\end{bmatrix} = \begin{bmatrix}\theta_{1,1} & \cdots & \theta_{1,C} \\ \vdots & \ddots & \vdots \\ \theta_N & \cdots & \theta_{N,C}\end{bmatrix}$$

$$\pi_\Theta(a \mid \boldsymbol{x}) = \exp\left(\boldsymbol{\theta}_a^\top \boldsymbol{x}\right) / \sum_{a'} \exp\left(\boldsymbol{\theta}_{a'}^\top \boldsymbol{x}\right)$$

Even though the classifier has a linear structure, the final non-linearity at the end is there to ensure that it defines a proper probability distribution over decisions. For classification problems, the observations $\boldsymbol{x}_t$ are features $\boldsymbol{x}_t = (x_{t,1} \ldots, x_{t,n})$ so that $\mathcal{X} \subset \mathbb{R}^N$. It is convenient to consider the network output as a vector on the simplex $\boldsymbol{y} \in \boldsymbol{\Delta}^A$, i.e. $\sum_{i=1}^C y_{t,i} = 1$, $y_{t,i} \geq 0$. In the neural network model for classification, we typically ignore dependencies between the $x_{t,i}$ features, as we are not very interested in the distribution of $\boldsymbol{x}$ itself.

Gradient ascent for a matrix U

$$\max_\theta \sum_{t=1}^T \sum_{a_t} U(a_t, y_t)\pi_\theta(a_t \mid x_t) \qquad \text{(objective)}$$

$$\nabla_\theta \sum_{t=1}^T \sum_{a_t} U(a_t, y_t)\pi_\theta(a_t \mid x_t) \qquad \text{(gradient)}$$

$$= \sum_{t=1}^T \sum_{a_t} U(a_t, y_t)\nabla_\theta \pi_\theta(a_t \mid x_t) \tag{4.4.4}$$

We now need to calculate the gradient of the policy.

Chain Rule of Differentiation

$$f(z), z = g(x), \qquad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx} \qquad \text{(scalar version)}$$

$$\nabla_\theta \pi = \nabla_g \pi \nabla_\theta g \qquad \text{(vector version)}$$

Learning outcomes

Understanding

- Classification as an optimisation problem.
- (Stochastic) gradient methods and the chain rule.
- Neural networks as probability models or classification policies.
- Linear neural netwoks.
- Nonlinear network architectures.

Skills

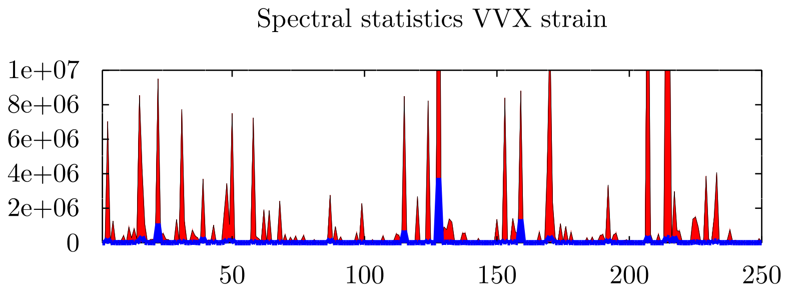
- Using a standard NN class in python.

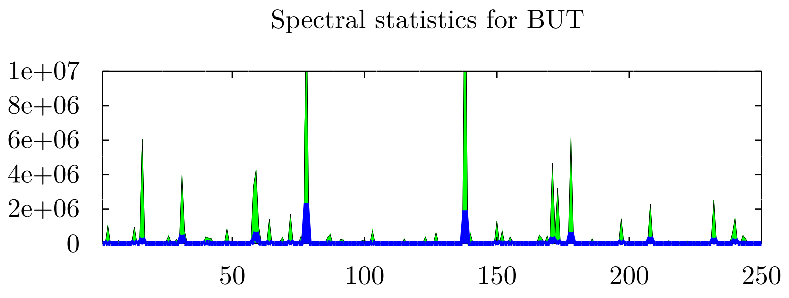
Reflection

- How useful is the ability to have multiple non-linear layers in a neural network.
- How rich is the representational power of neural networks?
- Is there anything special about neural networks other than their allusions to biology?

4.5 Nearest neighbours

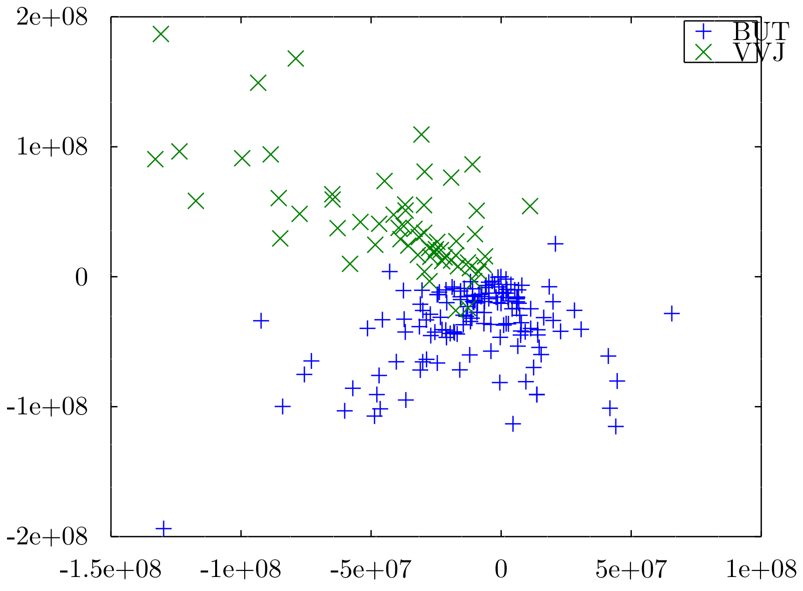
Discriminating between diseases





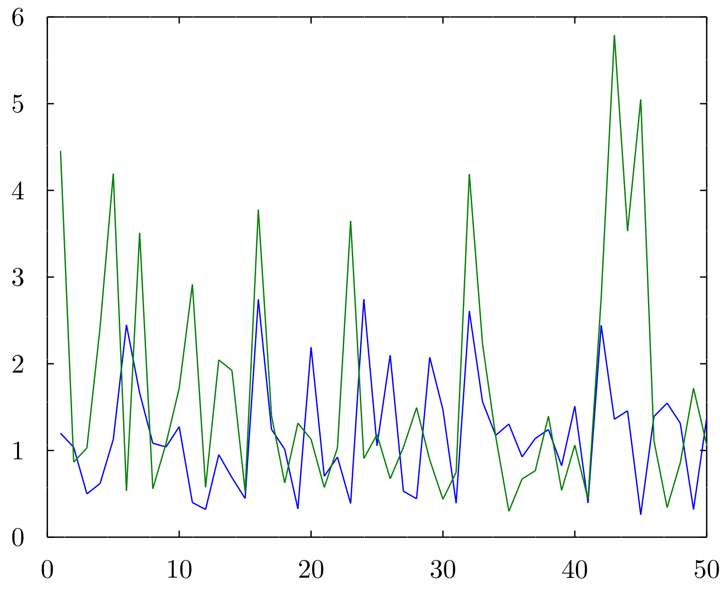
Let's tackle the problem of discriminating between different disease vectors. Ideally, we'd like to have a simple test that tells us what ails us. One kind of test is mass spectrometry. This graph shows spectrometry results for two types of bacteria. There is plenty of variation within each type, both due to measurement error and due to changes in the bacterial strains. Here, we plot the average and maximum energies measured for about 100 different examples from each strain.

Nearest neighbour: the hidden secret of machine learning



Now, is it possible to identify an unknown strain based on this data? Actually, this is possible. Sometimes, very simple algorithms work very well. One of the simplest one involves just measuring the distance between the decsription of a new unknown strain and known ones. In this visualisation, I projected the 1300-dimensional data into a 2-dimensional space. Here you can clearly see that it is possible to separate the two strains. We can use the distance to examples VVT and BUT in order to decide the type of an unknown strain.

Comparing spectral data



The choice of distance in this kind of algorithm is important, particularly for very high dimensions. For something like a spectrogram, one idea is look at the total area of the difference between two spectral lines.

The nearest neighbour algorithm

The nearest neighbour algorithm for classification (Alg. 1) does not include any complicated learning. Given a training dataset D, it returns a classification decision for any new point x by simply comparing it to its closest k neighbours in the dataset. It then estimates the probability p_y of each class y by calculating the average number of times the neighbours take the class y.

Algorithm 1 K-NN Classify

```
1: Input Data $D = \{(x_1, y_1), \ldots, (x_T, y_T)\}$, $k \geq 1$, $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, new point $x \in \mathcal{X}$
2: $D = \texttt{Sort}(D, d)$ % Sort $D$ so that $d(x, x_i) \leq d(x, x_{i+1})$.
3: $p_y = \sum_{i=1}^{k} \mathbb{I}\{y_i = y\} / k$ for $y \in \mathcal{Y}$.
4: Return $\boldsymbol{p} \triangleq (p_1, \ldots, p_k)$
```

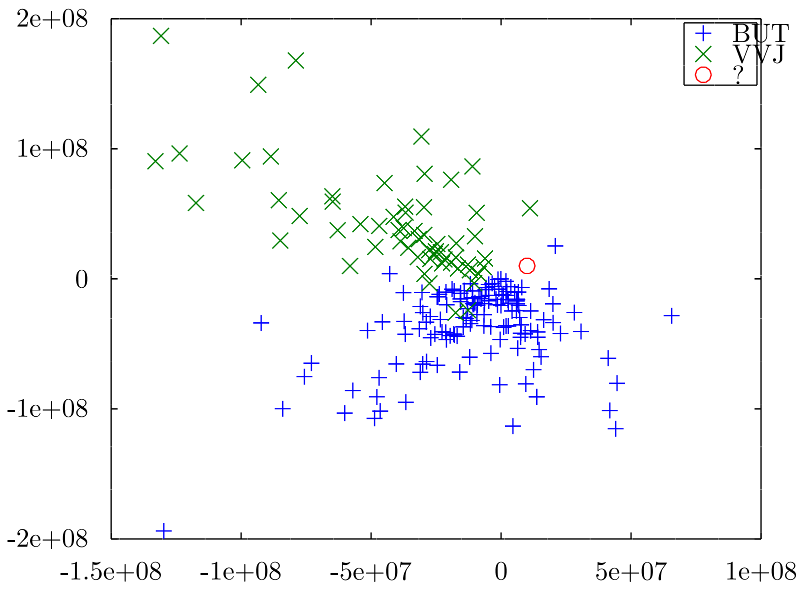
⚠ **Algorithm parameters**

In order to use the algorithm, we must specify some parameters, namely.

- Neighbourhood $k \geq 1$. The number of neighbours to consider.
- Distance $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$. The function we use to determine what is a neighbour.

Figure 4.14: The nearest neighbours algorithm was introduced by Fix and Hodges Jr [12], who also proved consistency properties.

Nearest neighbour: What type is the new bacterium?



Given that the $+$ points represent the BUT type, and the $\times$ points the VVJ type, what type of bacterium could the circle point be?

Separating the model from the classification policy

- The K-NN algorithm returns a model giving class probabilities for new data points.
- It is up to us to decide how to use this model to decide upon a given class. A typical decision making rule can be in the form of a policy π that depends on what the model says. However, the simplest decision rule is to take the most likely class:

$$\pi(a \mid x) = \mathbb{I}\left\{p_a \geq p_y \forall y\right\}, \qquad \boldsymbol{p} = \textsc{k}\text{-NN}(D, k, d, x)$$

Discussion: Shortcomings of k-nearest neighbour

- Choice of k The larger k is, the more data you take into account when making your decision. This means that you expect your classes to be more spread out.
- Choice of metric d. The metric d encodes prior information you have about the structure of the data.
- Representation of uncertainty. The predictions of kNN models are simply based on distances and counting. This might not be a very good way to represent uncertainty about class label. In particular, label probabilities should be more uncertain when we have less data.
- Scaling with large amounts of data. A naive implementation of kNN requires the algorithm to shift through all the training data to find the k nearest neighbours, suggesting a super-linear computation time. However, advanced data structures such as Cover Trees (or even KD-trees in low dimensional spaces) can be used to find the neighbours in polylog time.

- Meaning of label probabilities. It is best to think of κ-NN as a *model* for predicting the class of a new example from a finite set of existing classes. The model itself might be incorrect, but this should nevertheless be OK for our purposes. In particular, we might later use the model in order to derive classification rules.

Learning outcomes

Understanding

- How kNN works
- The effect of hyperameters k, d for nearest neighbour.
- The use of kNN to classify new data.

Skills

- Use a standard kNN class in python
- Optimise kNN hyperparameters in an unbiased manner.
- Calculate probabilities of class labels using kNN.

Reflection

- When is kNN a good model?
- How can we deal with large amounts of data?
- How can we best represent uncertainty?

4.6 Naive Bayes classifiers*

One special case of this idea is in classification, when each hypothesis corresponds to a specific class. Then, given a new example vector of data $\boldsymbol{x}$, we would like to calculate the probability of different classes C given the data, $\mathbb{P}(C \mid \boldsymbol{x})$. So here, the class is the hypothesis.

From Bayes's theorem, we see that we can write this as

$$\mathbb{P}(C \mid \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{x} \mid C)\,\mathbb{P}(C)}{\sum_i \mathbb{P}(\boldsymbol{x} \mid C_i)\,\mathbb{P}(C_i)}$$

for any class C. This directly gives us a method for classifying new data, as long as we have a way to obtain $\mathbb{P}(\boldsymbol{x} \mid C)$ and $\mathbb{P}(C)$.

But should we use for the probability model $\mathbb{P}$?

Naive Bayes classifier

Naive Bayes classifiers are one of the simplest classification methods. They can have a full Bayesian interpretation under some assumptions, but otherwise they are too simplistic to be useful.

Calculating the prior probability of classes

A simple method is to simply count the number of times each class appears in the training data $D_T = ((x_t, y_t))_{t=1}^T$. Then we can set

$$\mathbb{P}(C) = 1/T \sum_{t=1}^{T} \mathbb{I}\{y_t = C\}$$

The Naive Bayes classifier uses the following model for observations, where observations are independent of each other given the class. Thus, for example the result of three different tests for lung cancer (stethoscope, radiography and biopsy) only depend on whether you have cancer, and not on each other.

Probability model for observations

$$\mathbb{P}(\boldsymbol{x} \mid C) = \mathbb{P}(x(1), \dots, x(n) \mid C) = \prod_{k=1}^{n} \mathbb{P}(x(k) \mid C).$$

There are two different types of models we can have, one of which is mostly useful for continuous attributes and the other for discrete attributes. In the first, we just need to count the number of times each feature takes different values in different classes.

Discrete attribute model.

Here we simply count the average number of times that the attribute k had the value i when the label was C. This is in fact analogous to the conditional probability definition.

$$\mathbb{P}(x(k) = i \mid C) = \frac{\sum_{t=1}^{T} \mathbb{I}\{x_t(k) = i \wedge y_t = C\}}{\sum_{t=1}^{T} \mathbb{I}\{y_t = C\}} = \frac{N_k(i, C)}{N(C)},$$

where $N_k(i, C)$ is the numb l l .er of examples in class C whose k-th attribute has the value i, and $N(C)$ is the number of examples in class C.

⚠ **Full Bayesian approach versus maximum likelihood**

This estimation is simple maximum likelihood, as it does not maintain a distribution over the parameters.

Sometimes we need to be able to deal with cases where there are no examples at all of one class. In that case, that class would have probability zero. To get around this problem, we add "fake observations" to our data. This is called *Laplace smoothing.*

Remark 4.6.1. In Laplace smoothing with constant λ, our probability model is

$$\mathbb{P}(x(k) = i \mid C) = \frac{\sum_{t=1}^{T} \mathbb{I}\{x_t(k) = i \wedge y_t = C\} + \lambda}{\sum_{t=1}^{T} \mathbb{I}\{y_t = C\} + n_k\lambda} = \frac{N_k(i, C) + \lambda}{N(C) + n_k\lambda}.$$

where n_k is the number of values that the k-th attribute can take. This is necessary, because we want $\sum_{i=1}^{n_k} \mathbb{P}(x(k) = i \mid C) = 1$. (You can check that this is indeed the case as a simple exercise).

Remark 4.6.2. In fact, the Laplace smoothing model corresponds to a so-called Dirichelt prior over polynomial parameters with a marginal probability of observation equal to the Laplace smoothing. This is an extension of Beta-Bernoulli example from binary outcomes to multiple outcomes.

Continuous attribute model.

Here we can use a Gaussian model for each continuous dimension.

$$\mathbb{P}(x(k) = v \mid C) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(v-\mu)^2}{\sigma^2}},$$

where μ and σ are the mean and variance of the Gaussian, typically calculated from the training data as:

$$\mu = \frac{\sum_{t=1}^{T} x_t(k)\, \mathbb{I}\{y_t = C\}}{\sum_{t=1}^{T} \mathbb{I}\{y_t = C\}},$$

i.e. μ is the mean of the k-th attribute when the label is C and

$$\sigma = \frac{\sum_{t=1}^{T} [x_t(k) - \mu]^2\, \mathbb{I}\{y_t = C\}}{\sum_{t=1}^{T} \mathbb{I}\{y_t = C\}},$$

i.e. σ is the variance of the k-th attribute when the label is C. Sometimes we can just fix σ to a constant value, i.e. $\sigma = 1$.

⚠ Full Bayesian approach

This estimation is simple maximum likelihood, as it selects a single parameter pair $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ for every class and does not maintain a distribution over the parameters. It also assumes independence between the features. The full Bayesian approach considers an arbitrary covariance matrix $\boldsymbol{\Sigma}$ and maintains a distribution $\beta(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Chapter 5

Reproducibility

5.1 Reproducibility

One of the main problems in science is reproducibility: when we are trying to draw conclusions from one specific data set, it is easy to make a mistake. For that reason, the scientific process requires us to use our conclusions to make testable predictions, and then test those predictions with new experiments. These new experiments should bear out the results of the previous experiments. In more detail, reproducibility can be thought of as two different aspects of answering the question “can this research be replicated?”

Computational reproducibility: Can the study be repeated?

Can we, from the available information and data, exactly reproduce the reported methods and results?

This is something that is useful to be able to even to the original authors of a study. The standard method for achieving this is using version control tools so that the exact version of algorithms, text and data used to write up the study is appropriately labelled. Ideally, any other researcher should be able to run a single script to reproduce all of the study and its computations. The following tools are going to be used in this course:

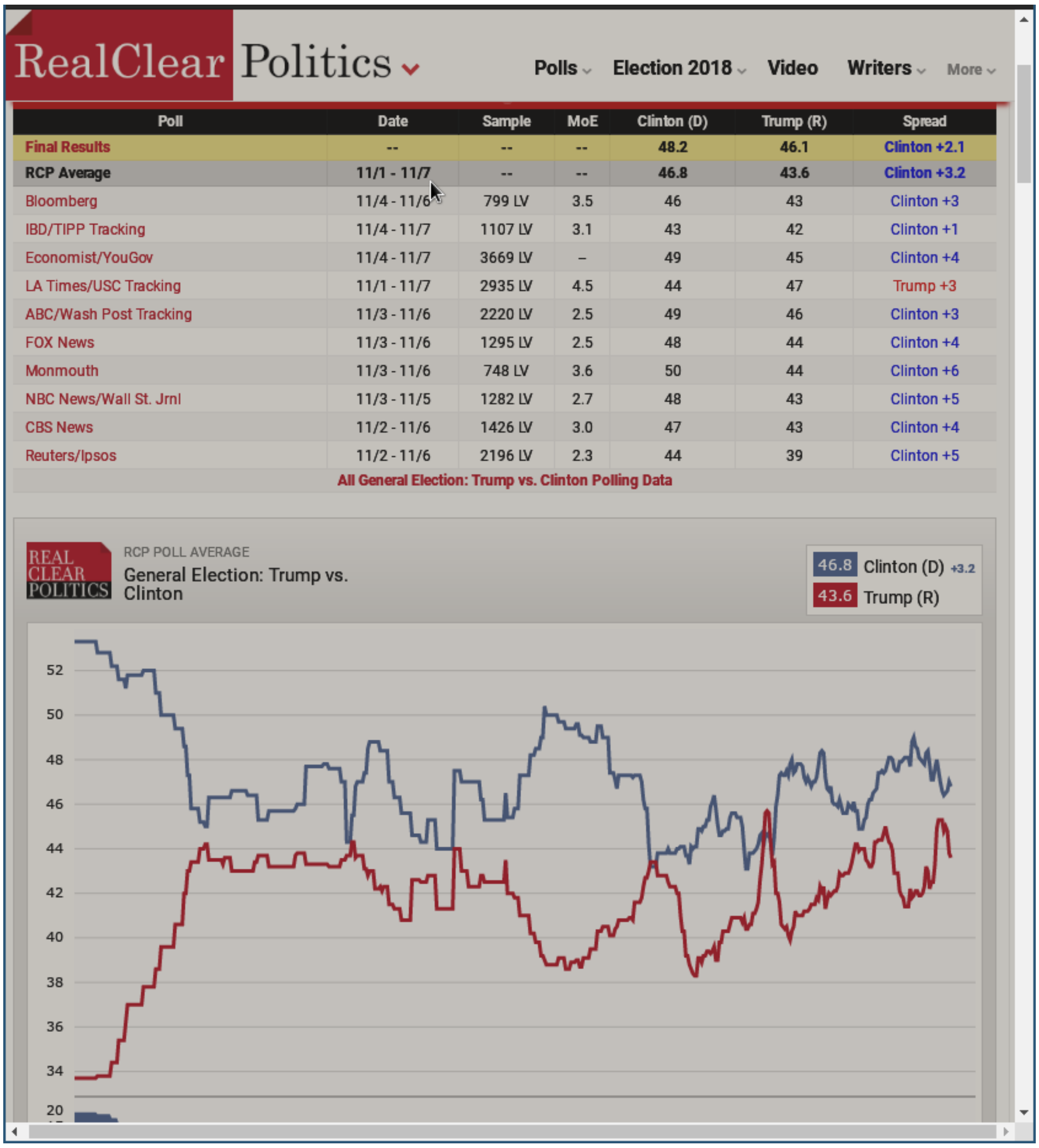
- `jupyter` notebooks for interactive note taking.
- `svn`, `git` or `mercurial` version control systems for tracking versions, changes and collaborating with others.

Scientific reproducibility: Is the conclusion correct?

Can we, from the available information and a *new* set of data, reproduce the conclusions of the original study?

Here followup research may involve using exactly the same methods. In AI research would mean for example testing whether an algorithm is really performing as well as it is claimed, by testing it in new problems. This can involve a re-implementation. In more general scientific research, it might be the case that the methodology proposed by an original study is flawed, and so a better method should be used to draw better conclusions. Or it might simply be that more data is needed.

When publishing results about a *new method*, computational reproducibility is essential for scientific reproducibility.



Poll	Date	Sample	MoE	Clinton (D)	Trump (R)	Spread
Final Results	--	--	--	48.2	46.1	Clinton +2.1
RCP Average	11/1 - 11/7	--	--	46.8	43.6	Clinton +3.2
Bloomberg	11/4 - 11/6	799 LV	3.5	46	43	Clinton +3
IBD/TIPP Tracking	11/4 - 11/7	1107 LV	3.1	43	42	Clinton +1
Economist/YouGov	11/4 - 11/7	3669 LV	--	49	45	Clinton +4
LA Times/USC Tracking	11/1 - 11/7	2935 LV	4.5	44	47	Trump +3
ABC/Wash Post Tracking	11/3 - 11/6	2220 LV	2.5	49	46	Clinton +3
FOX News	11/3 - 11/6	1295 LV	2.5	48	44	Clinton +4
Monmouth	11/3 - 11/6	748 LV	3.6	50	44	Clinton +6
NBC News/Wall St. Jrnl	11/3 - 11/5	1282 LV	2.7	48	43	Clinton +5
CBS News	11/2 - 11/6	1426 LV	3.0	47	43	Clinton +4
Reuters/Ipsos	11/2 - 11/6	2196 LV	2.3	44	39	Clinton +5

All General Election: Trump vs. Clinton Polling Data

A simple example is the 2016 presidential election in the United States, where polling forecasts were not able to predict the outcome. In general, while we can make models about people's opinions regarding candidates in order to predict voting totals, the test of these models comes in the actual election. Unfortunately the only way we have of tuning our models is on previous elections, which are not that frequent, and on the results of previous polls. In addition, predicting the winner of an election is slightly different from predicting how many people are prepared to vote for them across the country. This, together with other factors such as shifting opinions, motivation and how close the sampling distribution is to the voting distribution have a significant effect on accuracy.

The same thing can be done in when dealing purely with data, by making sure we use some of the data as input to the algorithm, and other data to measure the quality of the algorithm itself. In the following, we assume we have some algorithm $\lambda : \mathcal{D} \to \Pi$, where $\mathcal{D}$ is the universe of possible input data and Π the possible outputs, e.g. all possible classification policies. We also assume the existence of some quality measure U. How should we measure the quality of our algorithmic choices?

Take classification as an example. For a given training set, simply memorising all the labels of each example gives us perfect performance on the training set. Intuitively, this is not a good measure of performance, as we'd probably get poor results on a freshly sampled set. We can think of the training data as input to an algorithm, and the resulting classifier as the algorithm output. The evaluation function also requires some data in order to measure the performance of the policy. This can be expressed into the following principle.

⚠ **The principle of independent evaluation**

Data used for estimation cannot be used for evaluation.

This applies both to computer-implemented and human-implemented algorithms.

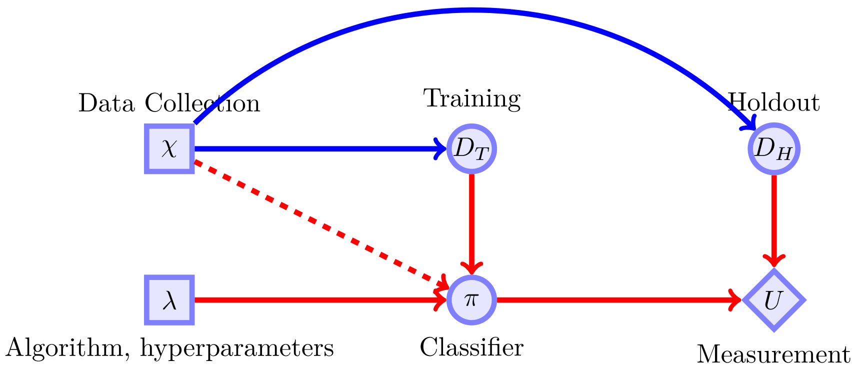


Figure 5.1: The decision process in classification.

One can think of the decision process in classification as follows. First, we decide to collect some data according to some experimental protocol χ. We also decide to use some algorithm (with associated hyperparameters) λ together with data D_T we will obtain from our data collection in order to obtain a classification policy π. Typically, we need to measure the quality of a policy according to how well it classifies on unknown data. This is because our policy has been generated using D_T, and so any measurement of its quality using the same data D_T, is going to be biased.

Classification accuracy. For classification problems, there is a natural metric U to measure: The classification accuracy of the classifier. If the classification decisions are stochastic, then the classifier assigns probability $\pi(a \mid x)$ to each possible label a, and our utility is simply the identity function $U(a, y) \triangleq \mathbb{I}\{a = y\}$.

Classification accuracy

For any fixed classifier π, the classification accuracy under the data distribution χ is:

$$\mathbb{E}_\chi[U(\pi)] = \sum_{x,y} \underbrace{\mathbb{P}_\chi(x,y)}_{\text{Data probability}} \overbrace{\pi(a = y \mid x)}^{\text{Decision probability}}$$

The classification accuracy of policy π under χ is the expected number of times the policy decides π chooses the correct class. However, when approximating χ with a sample D_H, we instead obtain the empirical estimate:

$$\mathbb{E}_{D_H} U(\pi) = \sum_{(x,y)\in D_H} \pi(a = y \mid x)/|D_H|.$$

Of course, there is no reason to limit ourselves to the identity function. The utility could very well be such that some errors are penalised more than other errors. Consider for example an intrusion detection scenario: it is probably more important to correctly classify intrusions. This can be taken into account by weighing correct decisions about intrusions more relative to correct decisions about normal data.

5.1.1 The human as an algorithm

The human as an algorithm.

The same way with which an algorithm creates a model from some prior assumptions and data, so can a human select an algorithm and associated hyperparamters by executing an algorithm herself. This involves trying different algorithms and hyperparametrs on the same training data D_T and then measuring their performance in the holdout set D_H.

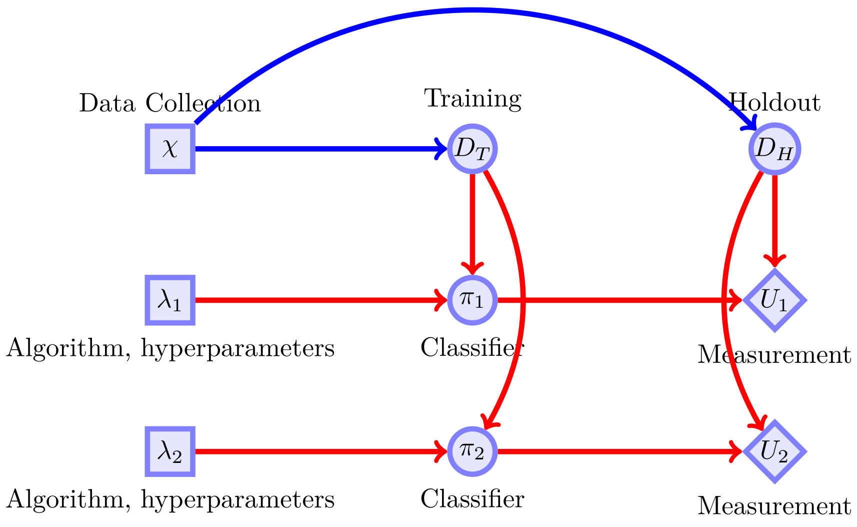


Figure 5.2: Selecting algorithms and hyperparameters through holdouts

Holdout sets

To summarise, holdout sets are used in order to be able to evaluate the performance of specific algorithms, or hyparameter selection.

- Original data D, e.g. $D = (x_1, \ldots, x_T)$.
- Training data $D_T \subset D$, e.g. $D_T = x_1, \ldots, x_n$, $n < T$.
- Holdout data $D_H = D \setminus D_T$, used to measure the quality of the result.
- Algorithm λ with hyperparametrs ϕ.
- Get algorithm output $\pi = \lambda(D_T, \phi)$.
- Calculate quality of output $U(\pi, D_H)$

We start with some original data D, e.g. $D = (x_1, \ldots, x_T)$. We then split this into a training data set $D_T \subset D$, e.g. $D_T = x_1, \ldots, x_n$, $n < T$ and holdout dataset $D_H = D \setminus D_T$. This is used to measure the quality of selected algorithms λ and hyperparameters ϕ. We run an algorithm/hyperparameter combination on the training data and obtain a result $\pi = \lambda(D_T, \phi)$. [1] We then calculate the quality of the output $U(\pi, D_H)$ on the holdout set. Unfortunately, the combination that appears the best due to the holdout result may look inferior in a fresh sample. Following the principle of "data used for evaluation cannot be used for estimation", we must measure performance on another sample. This ensures that we are not biased in our decision about what is the best algorithm.

Holdout and test sets for unbiased algorithm comparison

Consider the problem of comparing a number of different algorithms in Λ. Each algorithm λ has a different set of hyperparameters Φ_λ. The problem is to choose the best parameters for each algorithm, and then to test them independently. A simple meta-algorithm for doing this is based on the use of a *holdout* set for choosing hyperparameters for each algorithm, and a *test* set to measure algorithmic performance.

Algorithm 2 Unbiased adaptive evaluation through data partitioning

```
Partition data into $D_T, D_H, D^*$.
for $\lambda \in \Lambda$ do
    for $\phi \in \Phi_\lambda$ do
        $\pi_{\phi,\lambda} = \lambda(D_T, \phi)$.
    end for
    Get $\pi^*_\lambda$ maximising $U(\pi_{\phi,\lambda}, D_H)$.
    $u_\lambda = U(\pi^*_\lambda, D^*)$.
end for
$\lambda^* = \arg\max_\lambda u_\lambda$.
```

[1]As typically algorithms are maximising the quality metric on the training data,

$$\lambda(D_T) = \arg\max_y U(y, D_T)$$

we typically obtain a biased estimate, which depends both on the algorithm itself and the training data. For k-NN in particular, when we measure accuracy on the training data, we can nearly always obtain near-perfect accuracy, but not always perfect. Can you explain why?

Final performance measurement

When comparing many algorithms, where we must select a hyperparameter for each one, then we can use one dataset as input to the algorithms, and another for selecting hyperparameters. That means that we must use another dataset to measure performance. This is called the testing set. Figure 5.3 illustrates this.

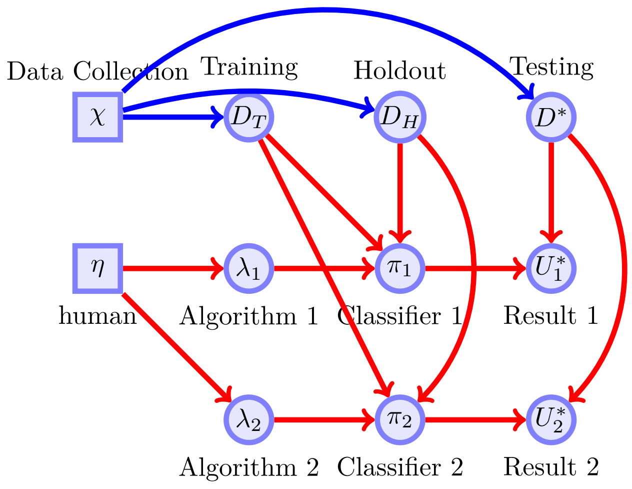


Figure 5.3: Simplified dependency graph for selecting hyperparameters for different algorithms, and comparing them on an independent test set. For the i-th algorithm, the classifier model is

5.1.2 Algorithmic sensitivity

The algorithm's output does have a dependence on its input, obviously. So, how sensitive is the algorithm to the input?

Independent data sets

One simple idea is to just collect independent datasets and see how the output of the algorithm changes when the data changes. However, this is quite expensive, as it not might be easy to collect data in the first place.

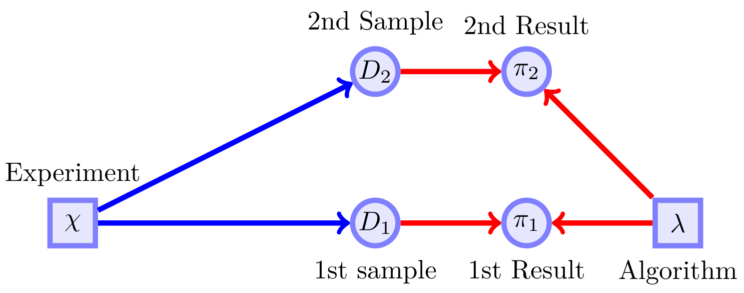


Figure 5.4: Multiple samples

Bootstrap samples

A more efficient idea is to only collect one dataset, but then use it to generate more datasets. The simplest way to do that is by sampling with replacement from the original dataset, new datasets of the same size as the original. Then the original dataset is sufficiently large, this is approximately the same as sampling independent datasets. As usual, we can evaluate our algorithm on an independent data set.

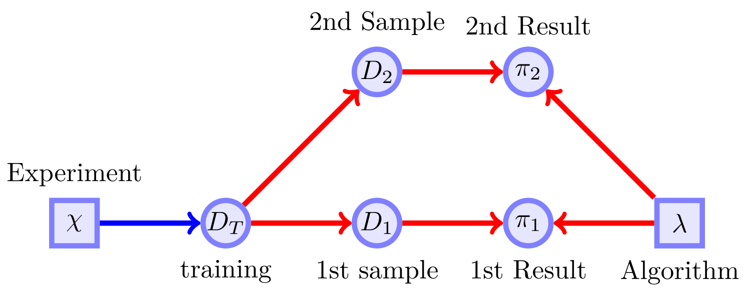


Figure 5.5: Bootstrap replicates of a single sample

Bootstrapping

Bootstrapping is a general technique that can be used to:

- Estimate the sensitivity of λ to the data x.
- Obtain a distribution of estimates π from λ and the data x.
- When estimating the performance of an algorithm on a small dataset D^*, use bootstrap samples of D^*. This allows us to take into account the inherent uncertainty in measured performance. It is very useful to use bootstrapping with pairwise comparisons.

Bootstrapping

```
1. **Input** Training data $D$, number of samples $k$.
2. **For** $i = 1, \ldots, k$
3.     $D^{(i)} = \text{Bootstrap}(D)$
4. **return** $\left\{D^{(i)} \;\middle|\; i = 1, \ldots, k\right\}$.
```

where Bootstrap(D) samples with replacement $|D|$ points from D_T.

In more detail, remember that even though the test score is an *independent* measurement of an algorithm's performance, it is *not* the actual expected performance. At best, it's an unbiased estimate of performance. Hence, we'd like to have some way to calculate a likely performance range from the test data. Bootstrapping can help: by taking multiple samples of the test set and calculating performance on each one, we obtain an empirical distribution of scores.

Secondly, we can use it to tell us something about the sensitivity of our algorithm. In particular, by taking multiple samples from the training data, we can end up with multiple models. If the models are only slightly different, then the algorithm is more stable and we can be more confident in its predictions.

Finally, bagging also allows us to generate probabilistic predictions from deterministic classification algorithms, by simply averaging predictions from multiple bootstrapped predictors. This is called *bagging predictors*[4].

Cross-validation

While we typically use a single training, hold-out and test set, it might be useful to do this multiple times in order to obtain more robust performance estimates. In the simplest case, cross-validation can be used to obtain multiple training and hold-out sets from a single dataset. This works by simply partitioning the data in *k folds* and then using one of the folds as a holdout and the remaining $k - 1$ as training data. This is repeated k times. When k is the same size as the original training data, then the method is called *leave-one-out cross-validation.*

k-fold Cross-Validation

```
1. **Input** Training data $D_T$, number of folds $k$, algorithm $\lambda$, measurement function $U$
2. Create the partition $D^{(1)} \dots, D^{(k)}$ so that $\bigcup_{i=1}^k D^{(k)} = D$.
3. Define $D_T^{(i)} = D \setminus D^{(i)}$
4.     $\pi_i = \lambda(D_T^{(i)})$
5. **For** $i = 1, \dots, k$:
6.     $\pi_i = \lambda(D^{(i)})$
7.     $u_i = U(\pi_i)$
8. **return** $\{y_1, \dots, y_i\}$.
```

Online evaluation

We can get around this problem if we consider online evaluation of learning algorithms. This means that the learning algorithm is always evaluated on previously unseen data. However, when new data is seen, it can be used by the algorithm to learn.

Example 18. Online prediction accuracy

- Adaptive decision rule π
- At time t
  1. π predicts a_t
  2. The true data x_t is observed and we see whether $a_t = y_t$.
  3. π adapts to the new data x_t

For this example, you can consider the decision rule π as being conditioned on the previous data, i.e.

$$\pi(a_t \mid x_1, \dots, x_t)$$

5.1.3 Beyond the data you have: simulation and replication

In the end, however, you are always limited by the data you actually have. The more you tweak your models and algorithms to improve performance with your current dataset, the more you are simply engineering a solution that is adapted to the specific data. This may not generalise well, even if you are using cross-validation or bootstrapping every step of the way. How can you then make sure that your methodology is robust?

The first method is simply to simulate data from an artificial process, where the ground truth (e.g. the labels) is known, and where the dataset size and dimensionality is similar to the one you have. You can use this to see whether the overall process is robust, and this can give you confidence that you will get reasonable results when using real data. The second method requires actually collecting new data, and repeating the study. If the results can be replicated, the original study was not a fluke.

Simulation

Simulation can be extremely useful. It allows you to examine the performance of various methods as you change aspect of the data-generating process without ever having to look at the data in detail. Since the data is synthetically generated, you always know the ground truth, so you know precisely how good your methods are going to be. This is useful in particular when you want to perform a null hypothesis test, and want to see under which conditions you actually accept or reject a null hypothesis.

A good example of the use of simulation to validate a method is in the article by Bennett et al.[2] where they discuss the use of corrections for multiple comparison tests. This followed their study of uncorrected methods for fMRI analysis[3], where they found that commonly used such methods would detect meaningful brain activity in a dead salmon. They use simulation to select a correction method that would be neither too conservative (i.e. not detecting any significant brain activity) nor too sensitive (i.e. detecting activity where there is none).

Steps for a simulation pre-study

1. **Criteria**: Define your quality criteria, e.g. a utility function U.
2. **Algorithms**: Define the class of algorithms, models or to consider.
3. **Simulation**: Create a simulation that allows you to collect data similar to the real one.
4. **Protocol**: This defines the processing and model pipeline, as you would do it with the actual data.
5. **Pre-study**: Collect data from the simulation and analyse it with the created according to your protocl.
6. **Adjustment:** If the results are not as expected, alter the simulation, protocol etc as needed. The aim is to find a *protocol* for the pre-study that would lead to (a) reproducible results (b) allows you to analyse the behaviour of different algorithms under different assumptions.
7. **Use:** Apply the protocol to actual data, whether it has been already collected, or will be collected according to the protocol.

Independent replication

The gold standard for reproducibility is independent replication. Simply have another team try and reproduce the results you obtained, using completely new data. If the replication is successful, then you can be pretty sure there was no flaw in your original analysis. In typical scientific publishing, the replication study is done by a different set of authors than those of the original study.

Replication study

1. Reinterpret the original hypothesis and experiment.
2. Collect data according to the original protocol, *unless flawed.* It is possible that the original experimental protocol had flaws. Then the new study should try and address this through an improved data collection process. For example, the original study might not have been double-blind. The new study can replicate the results in a double-blind regime.
3. Run the analysis again, *unless flawed.* It is possible that the original analysis had flaws. For example, possible correlations may not have been taken into account.
4. See if the conclusions are in agreement.

Learning outcomes

Understanding

- What is a hold-out set, cross-validation and bootstrapping.
- The idea of not reusing data input to an algorithm to evaluate it.
- The fact that algorithms can be implemented by both humans and machines.

Skills

- Use git and notebooks to document your work.
- Use hold-out sets or cross-validation to compare parameters/algorithms in Python.
- Use bootstrapping to get estimates of uncertainty in Python.

Reflection

- What is a good use case for cross-validation over hold-out sets?
- When is it a good idea to use bootstrapping?
- How can we use the above techniques to avoid the false discovery problem?
- Can these techniques fully replace independent replication?

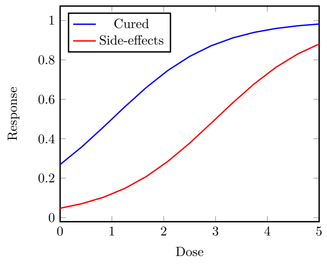
Chapter 6

Causality

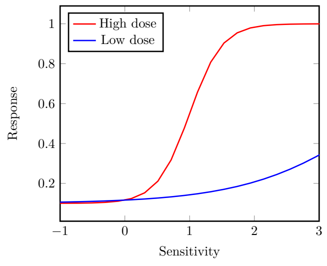
6.1 Introduction

Headaches and aspirins

Causal questions do not just deal with statistical relationships. The meaning of these questions is slightly different depending on whether we are talking about the population at large, or a specific individual. For populations, the main question is whether or not our actions have a causal effect. In observational data, we also need to consider the *direction of causation*.



(a) Dose-response curve.



(b) Response sensitivity

Figure 6.1: Investigating the response of the population to various doses of the drug.

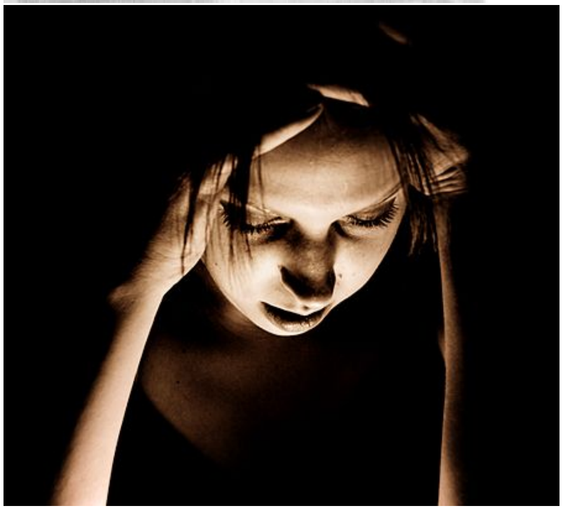
Example 19 (Population effects). We can ask ourselves two different questions about the effect of population effect aspirin on headaches.

- Is aspirin an effective cure for headaches?

- Does having a headache lead to aspirin-taking?

To examine the effect of aspirin on the population, we can look at the dose-response curve in Figure 6.1a. We first have to define what it means to be 'cured'. We also need to define possible side-effects. We can then measure how increased doses of aspirin lead to different outcomes. If the experiment has been properly conducted, this dose-response curve suggests that aspirin does have an effect in curing headaches, though it also has some side-effects. However, not all individuals are the same. Let us assume each person has a different *sensitivity* to aspirin, so that some individuals responds better to treatment, especially at high doses, and some do not respond very much at all, as shown in Figure 6.1b

For individuals, the first question is, what is the possible effect of our actions? This is called the *effect of causes.* The second question is, what was the reason for something happening? That is called the *cause of effects?*



Example 20 (Individual effects). We can ask ourselves two different questions about the individual effect of aspirin on headaches.

- Effects of *Causes*: Will *my* headache pass *if I take* an aspirin?
- *Causes* of Effects: Would *my* headache have passed if I had *not taken* an aspirin?

In order to be able to meaningfully talk about effects and causes we must also introduce decisions. Formally, there is nothing different in the decisions in this section and those introduced in Section 4.1. However, in this case we will try and use decisions to model outside interventions in a "natural" system, whereby a *null* decision means that we do not intervene.

Overview

Inferring causal models

We can distinguish different *models* from observational or experimental data.

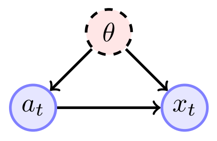
Inferring individual effects

The effect of possible intervention on an individual is not generally determinable. We usually require strong assumptions.

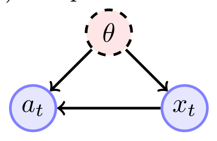
Decision-theoretic view

There are many competing approaches to causality. We will remain within the decision-theoretic framework, which allows us to crisply define both our knowledge and assumptions.

What causes what?



(a) Independence of a_t.



(b) Independence of x_t.

Example 21. Suppose we have data x_t, a_t where

- x_t: lung cancer
- a_t: smoking

Does smoking cause lung cancer or does lung cancer make people smoke? Can we compare the two models above to determine it?

The answer is no. Let us consider two different parametrisations of the distribution. One in which a_t generates x_t, and the converse, for any given parameter value θ, as given below:

$$P_\theta(D) = \prod_t P_\theta(x_t, a_t) = \prod_t P_{\theta'}(x_t \mid a_t) P_{\theta'}(a_t) = \prod_t P_{\theta''}(a_t \mid x_t) P_{\theta''}(x_t).$$

In particular, for any parametrisation θ of the joint distribution, there exists a θ' and θ'' giving the same probabilities for all x_t, a_t. For the example above, we can look at Bernoulli distributions for both variables so that $P_\theta(x_t = x, a_t = a) = \theta_{x,a}$. Then

$$\theta'_a = \sum_x \theta_{x,a}, \qquad \theta'_{x|a} = \theta_{x,a}/\theta'_a$$
$$\theta''_x = \sum_a \theta_{x,a}, \qquad \theta'_{a|x} = \theta_{x,a}/\theta''_x.$$

This means we can define prior distributions β, β', β'' in these three parameter spaces that give exactly the same results, e.g. by modelling each parameter as an independent Beta distribution. So, clearly simply looking at a simple graphical model does not let us answer this question.

6.1.1 Decision diagrams

However, graphical models *can* be extended to model causal relations. In particular, we can use *decision diagrams*[1], which include not only random variables, but also *decision* variables, denoted with squares, as well as utility variables, denoted via diamonds. In the following examples, we assume there are some underlying distributions specified by parameters θ, which we include in the diagrams for clarity. Even though it may seem intuitively sensible to suppose it, the arrow directions in the diagrams *do not* indicate direct causes. The only important thing for determining whether some variable influences another is whether or not there is independence between the corresponding decision and random variables.

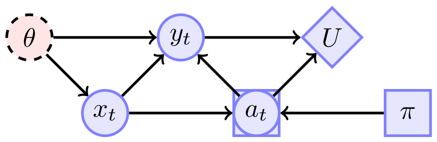


Figure 6.4: A typical decision diagram where x_t: individual information, y_t: individual result, a_t: action, π: policy

Example 22 (Taking an aspirin). The diagram in Figure 6.4 does not completely specify the decision problem. For aspirin taking, we can define the following variables:

- Individual t
- Individual information x_t
- $a_t = 1$ if t takes an aspirin, and 0 otherwise.
- $y_t = 1$ if the headache is cured in 30 minutes, 0 otherwise.
- π: intervention policy.

Example 23 (A recommendation system). Consider the example of a recommendation system, where we have data of the form (x_t, a_t, y_t). The performance of the recommendation system depends not only on the parameter θ, but also on the chosen policy π.

[1]Otherwise called influence diagrams

- x_t: User information (random variable)
- a_t: System action (random variable)
- y_t: Click (random varaible)
- π: recommendation policy (decision variable).

In both cases, there are some questions we can ask using the underlying model. The dependency structure is not enough to know *a priori* whether we can obtain meaningful answers. This depends on the specific assumptions we make about the model.

Conditional distributions and decision variables.

We begin with a parenthesis on conditional distributions. We normally define the conditional distribution of A given B under a probability measure P as:

$$P(A \mid B) \triangleq \frac{P(A \cap B)}{P(B)}.$$

However, decision variables are outside the scope of this probability measure, and yet we need to define conditional distributions using them.

The conditional distribution of decisions

If $\pi \in \Pi$ is a decision variable, we represent the conditional distribution of any random variable a given π simply as a collection of probability measures $\{\pi(a) \mid \pi \in \Pi\}$, one for each possible value π. The following notations will be equivalent:

$$\pi(a) \equiv \mathbb{P}^{\pi}(a) \equiv \mathbb{P}(a \mid \pi).$$

The reader should note that the standard definition of a conditional distribution also $P(A \mid B)$ creates a collection of distributions on A, with elements $P_B(A)$. However, it also specifies a rule for doing so from the complete distribution P.

If the random variables a also depends on some probability law P_θ, then it will be convenient to use the notation

$$\mathbb{P}^{\pi}_{\theta}(a) \equiv \mathbb{P}(a \mid \theta, \pi).$$

6.1.2 Common structural assumptions

In order to be clear about what constitutes an observation by the experimenter and what is a decision, we must clearly separate random variables from decision variables. The individual actions may be random variables, but they will depend on decisions taken. As we will see later, this is useful for modelling interventions.

Basic causal structures

Directed graphical models are not sufficient to determine causality by themselves, as they only determine correlations between random variables. If we have decision variables, however, we can always determine whether or not our decisions influence outcomes.

Non-cause

$$\begin{tikzcd} & \theta \arrow[dr] & \\ \pi \arrow[r] & a_t & y_t \arrow[l] \end{tikzcd}$$

Figure 6.5: π does not cause y

In the diagram above, we see that $y_t \perp\!\!\!\perp \pi$.

Example 24. Consider the model

$$y_t \sim \mathcal{N}(0, 1)$$
$$a_t \mid y_t, \pi \sim \mathcal{N}(y_t + \pi, 1)$$

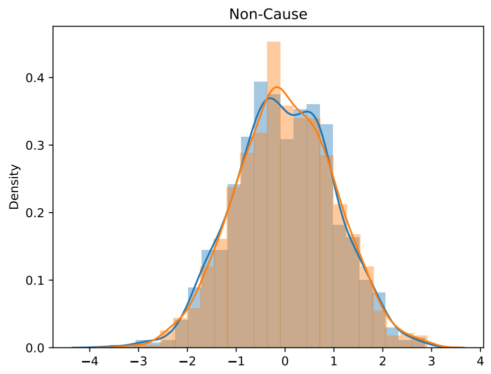


Figure 6.6: $\mathbb{P}^{\pi}(y_t)$ for $\pi \in \{-1, 1\}$ when π is not a cause for y_t

In this example, we see tht y_t is independent of the policy π. However, y_t is not independent of the action taken, as the action depends on y_t directly. The correlation between y, a is shown in Figure 6.9a.

No confounding

Confounding is a term that indicates the existence of latent variables that create dependencies between y_t, π, a_t. We are sure that there is no confounding whenever $y_t \perp\!\!\!\perp \pi \mid a_t$, as captured by the diagram in Figure 6.7. In this case π is a direct cause for y_t through a_t.

$$\begin{tikzcd} & \theta \arrow[dr] & \\ \pi \arrow[r] & a_t \arrow[r] & y_t \end{tikzcd}$$

Figure 6.7: No confounding: π causes y_t

Example 25. Consider the model

$$a_t \sim \mathcal{N}(\pi, 1)$$
$$y_t \mid a_t, \pi \sim \mathcal{N}(a_t, 1)$$

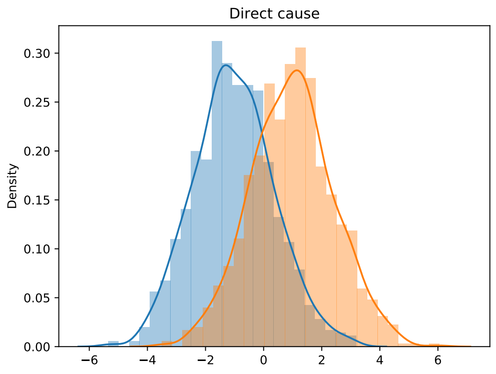
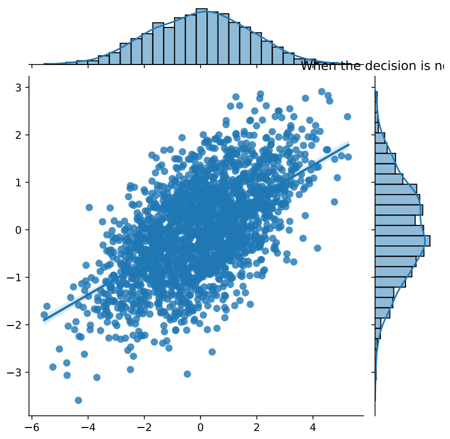
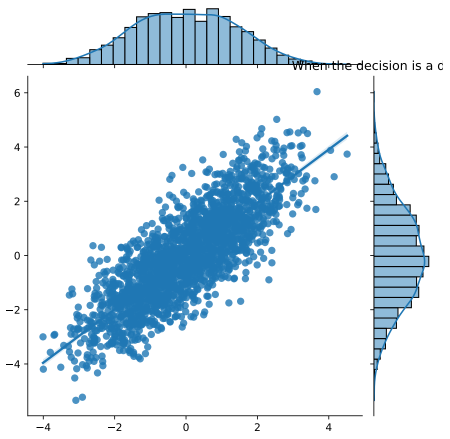


Figure 6.8: $\mathbb{P}^\pi(y_t)$ for $\pi \in \{-1, 1\}$ when π is a direct cause for y_t

We can see how the distribution of y_t changes when π changes in Figure 6.8. In this case there is also a correlation between a_t, y_t as seen in Figure 6.9.



(a) Non-cause



(b) Cause

Figure 6.9: Correlation between a_t and y_t

Covariates

Sufficient covariate

Sometimes the variable of interest is not conditionally independent of the treatment, unless there exists a *sufficient covariate* x_t such that $y_t \perp\!\!\!\perp \pi \mid a_t, x_t$. If x_t is not observed, then it is sometimes called a confounder.

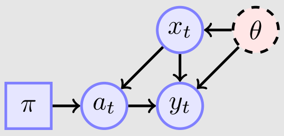

$$\begin{tikzcd} & & x_t \arrow[dl] \arrow[d] & \theta \arrow[l] \arrow[dl] \\ \pi \arrow[r] & a_t \arrow[r] & y_t & \end{tikzcd}$$

Figure 6.10: Sufficient covariate x_t

Example 26. Consider the model

$$\begin{aligned} x_t &\sim \mathcal{N}(0,1) \\ a_t &\sim \mathcal{N}(x_t + \pi, 1) \\ y_t \mid a_t, \pi &\sim \mathcal{N}(x_t + a_t, 1), \end{aligned}$$

Here x_t influences the outcome y_t, but also directly influences a_t through the policy π. As we can see in Figure 6.8, the policy then has an influence on y_t

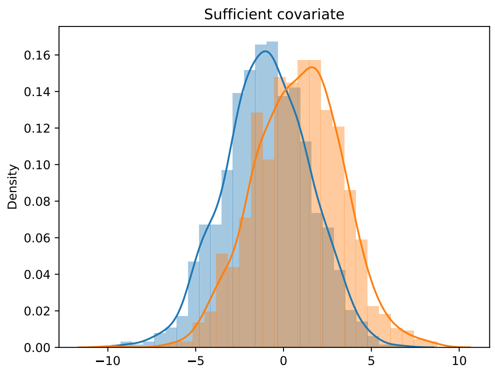


Figure 6.11: $\mathbb{P}^\pi(y_t)$ for $\pi \in \{-1, 1\}$ when π is a direct cause for y_t

Definition 6.1.1 (non-confounder)**.** A latent variable x_t is not a confounder if either $x_t \perp\!\!\!\perp a_t \mid \pi$ or $x_t \perp\!\!\!\perp y_t \mid a_t$

Instrumental variables and confounders

If the sufficient covariate x_t is not observed, we may still have another variable available, z_t, on the basis of which we make our decisions. This is called an *instrumental variable.* . More formally, z_t is an instrumental variable with respect to y_t since the latent variables on which y_t depends satisfy $x_t, \theta \perp\!\!\!\perp \pi$ and the outcome variable is independent of the policy given the action and the two latent variables: $y \perp\!\!\!\perp pol \mid a_t, x_t, \theta$.

In this case z_t and x_t are dependent, but the effect of the treatment depends on x_t directly. As x_t is a latent covariate, it can be called a *confounder.*

$$\begin{tikzcd}
 & z_t \arrow[d] & x_t \arrow[l] \arrow[d] & \theta \arrow[l] \arrow[ll, bend right] \arrow[dl] \\
\pi \arrow[r] & a_t \arrow[r] & y_t &
\end{tikzcd}$$

Figure 6.12: Instrumental variable z_t, confounder x_t

EXAMPLE 27. Consider the model

$$\begin{aligned}
x_t &\sim \mathcal{N}(0, 1) \\
z_t &\sim \mathcal{N}(x_t, 1) \\
a_t &\sim \mathcal{N}(z_t + \pi, 1) \\
y_t \mid a_t, \pi &\sim \mathcal{N}(x_t + a_t, 1)
\end{aligned}$$

In this scenario, x_t directly influences the outcome y_t, but is not observed.[2]

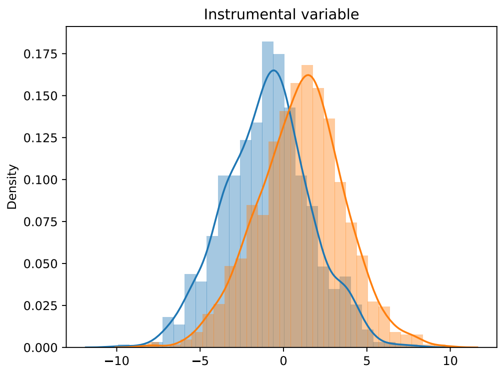


Figure 6.13: $\mathbb{P}^{\pi}(y_t)$ for $\pi \in \{-1, 1\}$ when π is a direct cause for y_t

Finally, variables which are neither confounders, nor instrumental, are called nuisance variables. Typically these include the latent variables of the problem, and any other variable that is marginalised out.

6.2 Interventions

Interventions are of primary interest when we have a set of observational data, collected under a *null* or *default* policy π_0. We then wish to intervene with some policy π in order to maximise our utility function, or to simply try and estimate the exact relationships between variables.

Modelling interventions

- Observational data D. This represents data we have collected from some previous regime. In order to be able to model the problem precisely, we must posit the existence of some default policy π_0 under which the data was collected.

[2]Hence, it can be called a confounder.

- Policy space Π. This must include π_0, as well as any other policies that the decision maker may be able to choose in the future.

Default policy

The space of policies Π includes a *default policy* π_0, under which the data was collected. The policy π_0 might already be known, if for example the data was collected with a specific algorithm. However, frequently π_0 is not given, and must also be inferred from the data. In that case, it can be seen as an additional parameter, complementary to θ.

Intervention policies

Except π_0, policies $\pi \in \Pi$ represent different interventions specifying a distribution $\pi(a_t \mid x_t)$.

- Direct interventions. The simplest scenario is when we are able to choose a π for which we know $\pi(a_t \mid x_t)$. This counts as a direct intervention, as we can specify any distribution of actions allowed in Π. If Π includes all conditional distributions, we can select an arbitrary action for every individual. This assumption is plausible for algorithmic decision making such as recommendation systems, but implausible when the actions are taken by another agent, such as a human.

- Indirect interventions and non-compliance. In this scenario we assume that, while we are free to choose policies from Π, we do not know what distribution $\pi(a_t \mid x_t)$ each policy specifies. In algorithmic decision making, this occurs whenever π represents hyperparameters and algorithms for which we have insufficient information. Then policies must be evaluated through some type of black-box (e.g. A/B) testing. When the actions are taken by (human) agents, the policy implies making a recommendation, which may not be followed by the agent. If we denote the recommendation by v_t, then we can write $\pi(a_t \mid x_t) = \sum_{z_t} \pi(a_t \mid z_t, x_t)\pi(z_t \mid x_t)$. In this scenario we can freely specify $\pi(z_t \mid x_t)$, but $\pi(a_t \mid z_t, x_t)$ must be estimated. For that reason, it is usually simpler to simply marginalise out a_t. But perhaps the simplest approach is to consider non-compliance as a confounder x_t, and z_t as an instrumental variable.

Example 28 (Weight loss). Consider weight loss. We can collect observational data from a population of overweight adults over a year. We can imagine that x represents the weight and vital statistics of an individual and y their change in weight after a year. We may also observe their individual actions a, such as whether or not they are following a particular diet or exercise regime. Under the default policy π_0, their actions are determined only the individuals. Consider an alternative policy π, which prescribes diet and exercise regimes. Due to non-compliance, actual actions taken by individuals may differ from prescribed actions. In addition, actions might not be observed.

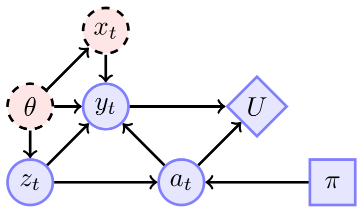


Figure 6.14: Model of non-compliance as a confounder.

6.3 Policy evaluation and optimisation

The value of an observed policy

In this section, we will focus on the general model of Figure 6.4. If we have data $D = \{(x_t, a_t, y_t) \mid t \in [T]\}$ generated from some policy π_0, we can always infer the average quality of each action a under that policy.

$$\hat{\mathbb{E}}_D(U \mid a) \triangleq \frac{1}{|\{t \mid a_t = a\}|} \sum_{t : a_t = a} U(a_t, y_t) \tag{6.3.1}$$

$$\approx \mathbb{E}^{\pi_0}_\theta(U \mid a) \qquad (a_t, y_t) \sim \mathbb{P}^{\pi_0}_\theta . \tag{6.3.2}$$

Can we calculate the value of another policy? As we have seen from Simpson's paradox, it is folly to simply select

$$\hat{a}^*_D \in \arg\max_a \hat{\mathbb{E}}_D(U \mid a),$$

as the action also depends on the observations x through the policy. To clarify this, let us look again at the model shown in Figure 6.4.

$$\begin{aligned} x_t \mid \theta &\sim P_\theta(x) \\ y_t \mid \theta, x_t, a_t &\sim P_\theta(y \mid x_t, a_t) \\ a_t \mid x_t, \pi &\sim \pi(a \mid x_t). \end{aligned}$$

Assume that $x \in \mathcal{X}$, a continuous space, but $y \in \mathcal{Y}$ is discrete. In this scenario, then the value of an action under a policy π is nonsensical, as it does not really depend on the policy itself:

$$\mathbb{E}^\pi_\theta(U \mid a) = \int_{\mathcal{X}} \mathrm{d}P_\theta(x) \sum_{y \in \mathcal{Y}} P_\theta(y \mid x, a) U(a, y).$$

We see that there is a clear dependence on the distribution of x, and there is no dependence on the policy any more. In fact, equation above only tells us the expected utility we'd get if we always chose the same action a. But what is the optimal policy? First, we have to define the value of a policy.

The value of a policy

$$\mathbb{E}^\pi_\theta(U) = \int_{\mathcal{X}} \mathrm{d}P_\theta(x) \sum_{a \in \mathcal{A}} \pi(a \mid x) \sum_{y \in \mathcal{Y}} P_\theta(y \mid x, a) U(a, y)$$

The optimal policy under a known parameter θ is given simply by

$$\max_{\pi \in \Pi} \mathbb{E}^\pi_\theta(U),$$

where Π is the set of allowed policies.

Monte-Carlo estimation

The simplest method to estimate the value of an alternative policy is to use Monte-Carlo estimation and importance sampling. However, this estimate can have a very high variance if the alternative policy is very different from the original policy.

Importance sampling[a]

We can obtain an unbiased estimate of the utility in a model-free manner through importance sampling:

$$
\begin{aligned}
\mathbb{E}^{\pi}_{\theta}(U) &= \int_{\mathcal{X}} \mathrm{d}P_\theta(x) \sum_a \mathbb{E}_\theta(U \mid a, x)\pi(a \mid x) \\
&\approx \frac{1}{T} \sum_a \sum_t \mathbb{E}_\theta(U \mid a, x_t)\pi(a \mid x_t), && x_t \sim P_\theta(x) \\
&= \frac{1}{T} \sum_t \sum_a \mathbb{E}_\theta(U \mid a, x_t) \frac{\pi(a \mid x_t)}{\pi_0(a \mid x_t)} \pi_0(a \mid x_t) \\
&\approx \frac{1}{T} \sum_{t=1}^T U_t \frac{\pi(a_t \mid x_t)}{\pi_0(a_t \mid x_t)}, && a_t \mid x_t \sim \pi_0, \quad U_t \mid x_t, a_t \sim P_\theta(U_t \mid x_t, a_t)
\end{aligned}
$$

Bayesian estimation

Unforunately this method has high variance. If we π_0 is given, we can calculate the utility of any policy to whatever degree of accuracy we wish. We begin with a prior β on Θ and obtain the following, assuming the policy π_0 is stationary.

$$
\begin{aligned}
\beta(\theta \mid D, \pi_0) &\propto \prod_t \mathbb{P}^{\pi_0}_\theta(x_t, y_t, a_t) \\
\mathbb{E}^{\pi}_{\beta}(U \mid D) &= \int_\Theta \mathbb{E}^{\pi}_{\theta}(U) \, \mathrm{d}\beta(\theta \mid D) \\
&= \int_\Theta \int_{\mathcal{X}} \mathrm{d}P_\theta(x) \sum_{t=1}^T \sum_a \mathbb{E}_\theta(U \mid a, x)\pi(a \mid x) \, \mathrm{d}\beta(\theta \mid D).
\end{aligned}
$$

Causal inference and policy optimisation

Causal inference requires building a complete model for the effect of both the model parameter θ, representing nature, and the policy π, representing the decision maker. This means that we have to be explicit about the dependencies of random variables on the model and the policy.

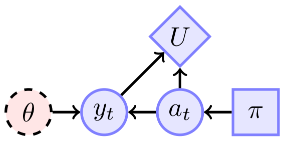


Figure 6.15: Simple decision problem.

Example 29. Let $a_t, y_t \in \{0, 1\}$, $\theta \in [0, 1]^2$ and

$$y_t \mid a_t = a \sim \mathcal{Bernoulli}(\theta_a)$$

Then, by estimating θ, we can predict the effect of any action. How can we estimate θ from historical data? We simply have to select the right parameter value. Simply put, each choice of a corresponds to one part of the parameter vector. This means that the maximum likelihood estimate

$$\hat{\theta}_a \triangleq \frac{1}{|\{t \mid a_t = a\}|} \sum_{\{t \mid a_t = a\}} y_t$$

is valid. We can also consider a product-Beta prior $\mathcal{Beta}(\alpha_a^0, \beta_b^0)$ for each one of the Bernoulli parameters, so that the posterior after t observations is

$$\alpha_a^t = \alpha_a^0 + \sum_{\{t \mid a_t = a\}} y_t, \qquad \beta_a^t = \beta_a^0 + \sum_{\{t \mid a_t = a\}} (1 - y_t).$$

How can we optimise the policy? Let us parametrise our policy with $\pi(a_t = 1) = w$.

For a fixed θ, the value of the policy is

$$\mathbb{E}_\theta^\pi U = \theta_1 w + \theta_0 (1 - w)$$

The gradient with respect to w is

$$\nabla \mathbb{E}_\theta^\pi U = \theta_1 - \theta_0,$$

so we can use the update

$$w^{(n+1)} = w^{(n)} + \delta^{(n)} \theta_1 - \theta_0.$$

However, $w \in [0, 1]$, which means our optimisation must be constrained. Then we obtain that $w = 1$ if $\theta_1 > \theta_0$ and 0 otherwise.

When θ is not known, we can use stochastic gradient descent.

$$\nabla \mathbb{E}_\beta^\pi U = \int_\Theta [\nabla \mathbb{E}_\theta^\pi U] \, \mathrm{d}\beta(\theta)$$

to obtain:

$$w^{(n+1)} = w^{(n)} + \delta^{(n)} \theta_1^{(n)} - \theta_0^{(n)}.$$

where $\theta^{(n)} \sim \beta$.

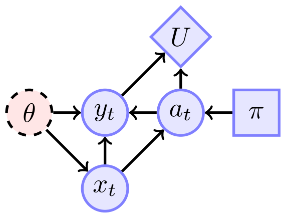


Figure 6.16: Decision problem with covariates.

EXAMPLE 30. Let $a_t, x_t = \{0, 1\}$, $y_t \in \mathbb{R}$, $\theta \in \mathbb{R}^4$ and

$$y_t \mid a_t = a, x_t = x \sim \mathcal{Bernoulli}(\theta_{a,x})$$

Then, by estimating θ, we can predict the effect of any action.

6.4 Individual effects and counterfactuals

Counterfactual analysis is mainly about questions relative to individuals, and specifically about what the effects of alternative actions would have been in specific instances in the past. We will assume a decision-theoretic viewpoint throughout, in order to be as clear as possible and avoiding imprecise language.

6.4.1 Disturbances and structural equation models

A structural equation model describes the random variables as deterministic functions of the decisions variables and the random exogenous disturbances. This allows us to separate the unobserved randomness from the known functional relationship between the other variables. Structurally, the model is essentially a variant of decision diagrams, as shown in Figure 6.17.

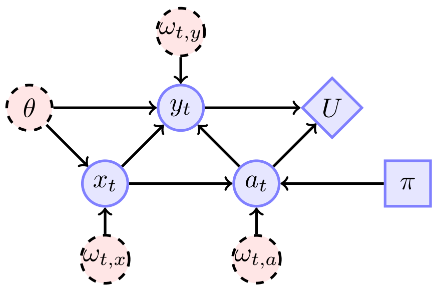


Figure 6.17: Decision diagram with exogenous disturbances ω.

We still need to specify particular functional relationships between the variables. Generally speaking, a random variable taking values in $\mathcal{X}$, is simply a function $\Omega \times \Theta \to \mathcal{X}$. For example, in Figure 6.17 $y_t = f_y(\omega, \theta)$. Taking into account the dependencies, this can be rewritten in terms of a function of the other random variables, and the local disturbance: $y_t = f_{y|a,x}(a, x, \omega_{t,y}, \theta)$. The choice of the function, together with the distribution of the parameter θ and the disturbances ω, fully determines our model.

Example 31 (Structural equation model for Figure 6.17). In structural equation models, the only random variables are the exogenous disturbances. In a fully Bayesian framework, θ is also a latent random variable. The remaining variables are deterministic functions.

$$\begin{aligned}
\theta &\sim \mathcal{N}(\mathbf{0}_4, \boldsymbol{I}_4), \\
x_t &= \theta_0 \omega_{t,x}, & \omega_{t,x} &\sim \mathcal{B}ernoulli(0.5) \\
y_t &= \theta_1 + \theta_2 x_t + \theta_3 a_t + \omega_{t,y}, & \omega_{t,y} &\sim \mathcal{N}(0,1) \\
a_t &= \pi(x_t) + \omega_{t,a} \mod |\mathcal{A}| & \omega_{t,a} &\sim 0.1\, \mathcal{D}(0) + 0.9\, \mathcal{U}nif(\mathcal{A}),
\end{aligned}$$

Structural equation models are particularly interesting in applications such as economics, where there are postulated relations between various economic quantities. If relationships between variables satisfy nice properties, then we can perform counterfactual inferences of the type : “What if I had *not* taken an aspirin?” In the example above, if we can infer the noise variables ω, we can change the value of some choice variables, i.e. a_t and see the effect on other variables like y_t directly.

Treatment-unit additivity

An example for a particular assumption for structural equation models is treatment-unit additivity. This states that the outcome depends on the action through a deterministic transformation and some additive noise that is only applied to the specific individual. More specifically, if individual t obtains treatment a_t, then the outcome only depends on a_t and individual-specific noise $\omega_{t,y}$. The assumption is given more formally below.

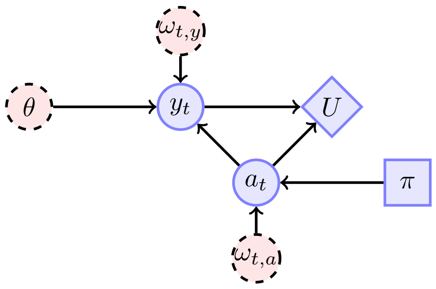


Figure 6.18: Decision diagram for treatment-unit additivity

Assumption 6.4.1 (TUA)**.** *For any given treatment $a \in \mathcal{A}$, the response variable satisfies*

$$y_t = g(a_t) + \omega_{t,y}$$

This implies that $\mathbb{E}[y_t \mid a_t = a] = g(a_t) + \mathbb{E}(\omega_{t,y})$. This implies that we might be able to easily estimate g, for example if the noise is zero-mean. The assumption makes sense for a lot of cases where we do not expect the outcomes of different individuals to be correlated through some confounding variable.

6.4.2 Example: Learning instrumental variables

This example is adapted from Hartford et al. [14], who use a deep learning to infer causal effects in the presence of instrumental variables. They break down the problem in two prediction tasks: the first is treatment prediction, and the second conditional treatment distribution. Unfortunately they do not use a decision-theoretic framework and so the difference between actual prices and policy changes is unclear.

Example 32 (Pricing model). In the following pricing model, we wish to understand how sales are affected by different pricing policies. In this example, there is a variable z_t which is an instrument, as it varies for reasons that are independent of demand and only affects sales through ticket prices.

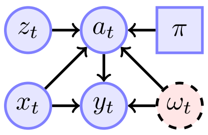


Figure 6.19: Graph of structural equation model for airport pricing policy π: a_t is the actual price, z_t are fuel costs, x_t is the customer type, y_t is the amount of sales, ω_t is whether there is a conference. The dependency on θ is omitted for clarity.

There are a number of assumptions we can make on the instrument z_t.

Assumption 6.4.2 (Relevance)**.** *a_t depends on z_t.*

In our example, it also depends on x_t.

Assumption 6.4.3 (Exclusion)**.** $z_t \perp\!\!\!\perp y_t \mid x_t, a_t, \omega_t$.

In other words, the outcome does not depend on the instrument directly. This was also satisfied in our first example of an instrumental variable.

Assumption 6.4.4 (Unconfounded instrument)**.** $z_t \perp\!\!\!\perp \omega_t \mid x_t$.

Prediction tasks

We can use the following structural equation model

$$y_t = g_\theta(a_t, x_t) + \omega_t, \qquad \mathbb{E}_\theta\, \omega_t = 0, \qquad \forall \theta \in \Theta \tag{6.4.1}$$

There are two slightly different prediction tasks we can think of in this model.

Standard prediction

In standard prediction tasks, we just want to estimate the distribution of sales given the characteristics and price. Since the actions are correlated with the outcome through the confounder, this estimate is biased.

$$\mathbb{P}^\pi_\theta(y_t \mid x_t, a_t), \qquad \mathbb{E}^\pi_\theta(y_t \mid x_t, a_t) = g_\theta(x_t, a_t) + \mathbb{E}^\pi_\theta(\omega_t \mid x_t, a_t).$$

[a]Also known as Propensity Scoring

Counterfactual prediction

$$\mathbb{E}^\pi_\theta(y_t \mid x_t, z_t) = \int_{\mathcal{A}} \underbrace{[g(a_t \mid x_t, z_t) + \mathbb{E}_\theta(\omega \mid x_t)]}_{h(a_t, x_t)} \,\mathrm{d}\pi(a_t \mid x_t)$$

6.4.3 Discussion

Further reading

- Pearl, *Causality.*
- Dawid[5]'s decision-theoretic causality
- Halpern's book `https://www.cs.cornell.edu/home/halpern/papers/causalitybook-ch1-3.html`
- Halpern's video lecture: `https://www.cs.cornell.edu/home/halpern/papers/causalitybook-ch1-3 html`

6.4.4 Exercises

In the following exercises, we are taking actions a_t and obtaining outcomes y_t. Our utility function is simply $U = y_t$.

EXERCISE 13. Let us have some data generated from a null treatment policy π_0 of the form (a_t, y_t). There is a simple model that explains the data of the form

$$y_t \mid a_t = a, \theta \sim \mathcal{N}(a + \theta, 1),$$

where the actions are distribution according to $\pi(a_t)$.

- Assume that $\pi_0 \in [0, 1]$ is given and it is $a_t \mid \pi = \pi_0 \sim \mathcal{Bernoulli}(\pi_0)$. First, estimate θ. Then, calculate the distribution of $y_t \mid \pi_0, \theta$ for any other policy and plot the resulting mean and variance as π changes. You can do this first in a maximum-likelihood manner. Advanced: estimate the posterior distribution of θ for a normal prior on θ.

- Now assume that π_0 is not given. This means that you must also estimate π_0 itself before estimating the effect of any other policy π on the data.
- In this exercise, can you learn about other actions when you are not taking them? Why?

Exercise 14. Let us have some data generated from a null treatment policy π_0 of the form (a_t, y_t). Let us now consider a slightly model where $\theta \in \mathbb{R}^2$.

$$y_t \mid a_t = a, \theta \sim \mathcal{N}(\theta_a, 1),$$

where the actions are distribution according to $\pi(a_t)$.

- Assume that $\pi_0 \in [0, 1]$ is given and it is $a_t \mid \pi = \pi_0 \sim \mathcal{Bernoulli}(\pi_0)$. First, estimate θ. Then, calculate the distribution of $y_t \mid \pi_0, \theta$ for any other policy and plot the resulting mean and variance as π changes. You can do this first in a maximum-likelihood manner. Advanced: estimate the posterior distribution of θ for a normal prior on θ.
- Now assume that π_0 is not given. This means that you must also estimate π_0 itself before estimating the effect of any other policy π on the data.
- In this exercise, can you learn about other actions when you are not taking them? Why?

Exercise 15. Given your estimates, find the optimal policy for each one of those cases. Measure the quality of this policy on

- The actual data you have already (e.g. using importance sampling)
- On new simulations (using the testing framework).

Advanced: The optimal policy when θ is known is to always take the same action. Does that still hold when θ is not known and you are estimating it all the time from new data?

Exercise 16 (Advanced). Let us have some data generated from a null treatment policy π_0 of the form (x_t, a_t, y_t), with $a_t, x_t \in \{0, 1\}$.

$$y_t \mid a_t = a, x_t = x, \theta \sim \mathcal{N}(\theta_{a,x}, 1),$$

where the actions are distribution according to $\pi_0(a_t \mid x_t)$.

- Assume that π_0 is given and it is $a_t \mid x_t = x, \pi = \pi_0 \sim \mathcal{Bernoulli}(w_x)$. First, estimate θ. Repeat your analysis.
- Now assume that π_0 is not given. Again, repeat your analysis.
- Is there now globally better action a_t? Should it depend on x_t, like in the observed policy? Can you estimate the optimal policy?

Chapter 7

Experiment design

7.1 Introduction

This unit describes the very general formalism of Markov decision processes (MDPs) for formalising problems in sequential decision making. Thus a *Markov decision process* can be used to model stochastic path problems, stopping problems, reinforcement learning problems, experiment design problems, and control problems.

Markov decision proces

We being by taking a look at the problem of *experimental design.* One instance of this problem occurs when considering how to best allocate treatments with unknown efficacy to patients in an adaptive manner, so that the best treatment is found, or so as to maximise the number of patients that are treated successfully. The problem, originally considered by**? ?** , informally can be stated as follows.

experimental design

We have a number of treatments of unknown efficacy, i.e. some of them work better than the others. We observe patients one at a time. When a new patient arrives, we must choose which treatment to administer. Afterwards, we observe whether the patient improves or not. Given that the treatment effects are initially unknown, how can we maximise the number of cured patients? Alternatively, how can we discover the best treatment? The two different problems are formalised below.

Adaptive treatment allo tion

Example 33. Consider k treatments to be administered to T volunteers. To each volunteer only a single treatment can be assigned. At the t-th trial, we treat one volunteer with some treatment $a_t \in \{1, \ldots, k\}$. We then obtain obtain a reward $r_t = 1$ if the patient is treated and 0 otherwise. We wish to choose actions maximising the utility $U = \sum_t r_t$. This would correspond to maximising the number of patients that get treated over time.

Adaptive hypothesis t ing

Example 34. An alternative goal would be to do a *clinical trail*, in order to find the best possible treatment. For simplicity, consider the problem of trying to find out whether a particular treatment is better or not than a placebo. We are given a hypothesis set Ω, with each $\omega \in \Omega$ corresponding to different models for the effect of the treatment and the placebo. Since we don't know what is the right model, we place a prior β_0 on Ω. We can perform T experiments, after which we must make decide whether or not the treatment is significantly better than the placebo. To model this, we define a decision set $\mathcal{A} = \{a_0, a_1\}$ and a utility function $U : \mathcal{A} \times \Omega \to \mathbb{R}$, which models the effect of each decision a given different versions of reality ω. One hypothesis $\omega \in \Omega$ is true. To distinguish them, we can choose from a set of k possible experiments to be performed over T trials. At the t-th trial, we choose experiment a_t and observe outcome $x_t \in \mathcal{X}$, with $x_t \sim P_\omega$ drawn from the true hypothesis. Our posterior is

$$\beta_t(\omega) \triangleq \beta_0(\omega \mid a_1, \ldots, a_t, x_1, \ldots, x_t).$$

The reward is $r_t = 0$ for $t < T$ and

$$r_T = \max_{a \in \mathcal{A}} \mathbb{E}_{\beta_T}(U \mid a).$$

Our utility in this can again be expressed as a sum over individual rewards, $U = \sum_{t=1}^{T} r_t$.

Both formalizations correspond to so-called *bandit problems* which we take a closer look at in the following section.

7.2 Bandit problems

The simplest bandit problem is the stochastic n-armed bandit. We are faced with n different one-armed bandit machines, such as those found in casinos. In this problem, at time t, you have to choose one *action* (i.e. a machine) $a_t \in \mathcal{A} = \{1, \ldots, n\}$. In this setting, each time t you play a machine, you receive a reward r_t, with fixed expected value $\omega_i = \mathbb{E}(r_t \mid a_t = i)$. Unfortunately, you do not know ω_i, and consequently the best arm is also unknown. How do you then choose arms so as to maximise the total expected reward?

Definition 7.2.1 (The stochastic n-armed bandit problem.)**.** This is the problem of selecting a sequence of actions $a_t \in \mathcal{A}$, with $\mathcal{A} = \{1, \ldots, n\}$, so as to maximise expected utility, where the utility is

$$U = \sum_{t=0}^{T-1} r_t,$$

where $T \in (0, \infty]$ is the horizon. The reward r_t is stochastic, and only depends on the current action, with expectation $\mathbb{E}(r_t \mid a_t = i) = \omega_i$.

icy

In order to select the actions, we must specify some *policy* or decision rule. This can only depend on the sequence of previously taken actions and observed rewards. Usually, the policy $\pi : \mathcal{A}^* \times \mathbb{R}^* \to \mathcal{A}$ is a deterministic mapping from the space of all sequences of actions and rewarsd to actions. That is, for every observation and action history $a_1, r_1, \ldots, a_{t-1}, r_{t-1}$ it suggests a single action a_t. However, it could also be a stochastic policy, that specifies a mapping to action distributions. We use the following notation for stochastic history-dependent bandit policies,

$$\pi(a_t \mid a^{t-1}, r^{t-1}) \tag{7.2.1}$$

to mean the probability of actions a_t given the history until time t.

How can we solve bandit problems? One idea is to apply the Bayesian decision-theoretic framework we have developed earlier to maximise utility in expectation. More specifically, given the horizon $T \in (0, \infty]$, we define define our utility from time t to be:

$$U_t = \sum_{k=1}^{T-t} r_{t+k}. \tag{7.2.2}$$

To apply the decision theoretic framework, we need to define a suitable family of probability measures $\mathcal{F}$, indexed by parameter $\omega \in \Omega$ describing the reward distribution of each bandit, together with a prior distribution β on Ω. Since ω is unknown, we cannot maximise the expected utility with respect to it. However, we can always maximise expected utility with respect to our belief β. That is, we replace the ill-defined problem of maximising utility in an unknown model with that of maximising expected utility given a distribution over possible models. The problem can be written in a simple form:

$$\max_\pi \mathbb{E}^\pi_\beta U_t = \max_\pi \int_\Omega \mathbb{E}^\pi_\omega U_t \, \mathrm{d}\beta\omega. \tag{7.2.3}$$

The difficulty lies not in formalising the problem, but in the fact that the set of learning policies is quite large, rendering the optimisation infeasible.

The following figure summarises the statement of the bandit problem in the Bayesian setting.

Decision-theoretic statement of the bandit problem

- Let $\mathcal{A}$ be the set of arms.
- Define a family of distributions $\mathcal{F} = \{P_{\omega,i} \mid \omega \in \Omega, i \in \mathcal{A}\}$ on $\mathbb{R}$.
- Assume the i.i.d model $r_t \mid \omega, a_t = i \sim P_{\omega,i}$.
- Define prior β on Ω.
- Select a policy $\pi : \mathcal{A}^* \times \mathbb{R}^* \to \mathcal{A}$ maximising

$$\mathbb{E}^\pi_\beta U = \mathbb{E}^\pi_\beta \sum_{t=0}^{T-1} r_t$$

There are two main difficulties with this approach. The first is specifying the family and the prior distribution: this is effectively part of the problem formulation and can severely influence the solution. The second is calculating the policy that maximises expected utility given a prior and family. The first problem can be resolved by either specifying a subjective prior distribution, or by selecting a prior distribution that has good worst-case guarantees. The second problem is hard to solve, because in general, such policies are history dependent and the set of all possible histories is exponential in the horizon T.

7.2.1 An example: Bernoulli bandits

As a simple illustration, consider the case when the reward for choosing one of the n actions is either 0 or 1, with some fixed, yet unknown probability depending on the chosen action. This can be modelled in the standard Bayesian framework using the Beta-Bernoulli conjugate prior. More specifically, we can formalise the problem as follows.

Consider n Bernoulli distributions with unknown parameters ω_i $(i = 1, \ldots, n)$ such that

$$r_t \mid a_t = i \sim \mathcal{B}ernoulli(\omega_i), \qquad \mathbb{E}(r_t \mid a_t = i) = \omega_i. \tag{7.2.4}$$

Each Bernoulli distribution thus corresponds to the distribution of rewards obtained from each bandit that we can play. In order to apply the statistical decision theoretic framework, we have to quantify our uncertainty about the parameters ω in terms of a probability distribution.

We model our belief for each bandit's parameter ω_i as a Beta distribution $\mathcal{B}eta(\alpha_i, \beta_i)$, with density $f(\omega \mid \alpha_i, \beta_i)$ so that

$$\beta(\omega_1, \ldots, \omega_n) = \prod_{i=1}^n f(\omega_i \mid \alpha_i, \beta_i).$$

Recall that the posterior of a Beta prior is also a Beta. Let

$$N_{t,i} \triangleq \sum_{k=1}^t \mathbb{I}\{a_k = i\}$$

be the number of times we played arm i and

$$\hat{r}_{t,i} \triangleq \frac{1}{N_{t,i}} \sum_{k=1}^{t} r_t \, \mathbb{I}\{a_k = i\}$$

be the *empirical reward* of arm i at time t. We can let this equal 0 when $N_{t,i} = 0$. Then, the posterior distribution for the parameter of arm i is

$$\beta_t = \mathcal{B}eta(\alpha_i + N_{t,i}\hat{r}_{t,i} \ , \ \beta_i + N_{t,i}(1 - \hat{r}_{t,i})).$$

Since $r_t \in \{0, 1\}$ the possible states of our belief given some prior are $\mathbb{N}^{2n}$.

In order for us to be able to evaluate a policy, we need to be able to predict the expected utility we obtain. This only depends on our current belief, and the state of our belief corresponds to the state of the bandit problem. This means that everything we know about the problem at time t can be summarised by β_t. For Bernoulli bandits, sufficient statistic for our belief is the number of times we played each bandit and the total reward from each bandit. Thus, our state at time t is entirely described by our priors α, β (the initial state) and the vectors

ef state

$$N_t = (N_{t,1}, \ldots, N_{t,i}) \tag{7.2.5}$$

$$\hat{r}_t = (\hat{r}_{t,1}, \ldots, \hat{r}_{t,i}). \tag{7.2.6}$$

At any time t, we can calculate the probability of observing $r_t = 1$ or $r_t = 0$ if we pull arm i as:

$$\beta_t(r_t = 1 \mid a_t = i) = \frac{\alpha_i + N_{t,i}\hat{r}_{t,i}}{\alpha_i + \beta_i + N_{t,i}}$$

So, not only we can predict the immediate reward based on our current belief, but we can also predict all next possible beliefs: the next state is well-defined and depends only on the current state. As we shall see later, this type of decision problem is more generally called a Markov decision process (Definition **??**). For now, we shall more generally (and precisely) define the bandit process itself.

7.2.2 The stochastic n-armed bandit problem

The stochastic n-armed bandit problem

Let us return to the example of bandit problems. As before, we have n actions corresponding to probability distributions P_i on the real numbers.

$$\mathcal{F} = \{P_i \mid i = 1, \ldots, n\}.$$

At each time-step t we select an action a_t, obtaining a random reward distributed according to:

$$r_t \mid a_t = i \sim P_i.$$

Our objective is to find a policy π maximising the expected total reward.

$$\mathbb{E}_\pi U_t = \mathbb{E}_\pi \sum_{k=t}^{T} r_k, \qquad a_t^* \triangleq \max\{\mathbb{E}(r_t \mid a_t = i) \mid i = 1, \ldots, n\}.$$

Had we known the distribution, we could simply always the maximising action, as the expected reward of the i-th action can be easily calculated from P_i and the reward only depends on our current action. The situation is similar when $\mathcal{F}$ is a parametric family unknown parameter ω^*, outlined below.

$$\mathcal{F} = \{P_i(\cdot \mid \omega) \mid \omega \in \Omega\}, \qquad r_t \mid a_t = i, \omega^* = \omega \sim P_i(r \mid \omega^*). \tag{7.2.7}$$

If in addition we have a subjective belief β over Ω, we could (as explained in Sec. 7.2) in principle calculate the policy maximising the β-expected utility:

$$\mathbb{E}^\pi_\beta U_t = \mathbb{E}^\pi_\beta \sum_{k=t}^T r_k. \tag{7.2.8}$$

This of course will now have to be a history-dependent policy. In the remainder of this section, we shall examine algorithms algorithms which eventually convergence to the optimal action, but for which we cannot always guarantee a good initial behaviour.

7.2.3 Estimation and Robbins-Monro approximation

The basic idea of the Robbins-Monro stochastic approximation algorithm**?** is to maintain a set of *point estimates* of a parameter we want to approximate and perform *random* steps that on average move towards the solution, in a way to be made more precise later. It can in fact be seen as a generalisation of stochastic gradient descent.

Algorithm 3 Robbins-Monro bandit algorithm

```
1: input Step-sizes $(\eta_t)_t$, initial estimates $(\mu_{i,0})_i$, policy $\pi$.
2: for $t = 1, \ldots, T$ do
3:     Take action $a_t = i$ with probability $\pi(i \mid a_1, \ldots, a_{t-1}, r_1, \ldots, r_{t-1})$.
4:     Observe reward $r_t$.
5:     $\mu_{t,i} = \eta_{i,t} r_t + (1 - \eta_{i,t}) \mu_{i,t-1}$        // estimation step
6:     $\mu_{t,i} = \mu_{j,t-1}$ for $j \neq i$.
7: end for
8: return $\boldsymbol{\mu_T}$
```

A simple Robbins-Monro algorithm for the n-armed bandit problem is given in Algorithm 3. The input is a particular policy π, that gives us a probability over the next actions given the observed history, a set of initial estates $\mu_{i,0}$ for the bandit means, and a sequence of step sizes η.

If you examine the updates carefully, you will be able to find what the cost function you are trying to minimise is. This simple update rule can be seen as trying to minimise the expected squared error between your estimated reward, and the random reward obtained by each bandit. Consequently, the variance of the reward of each bandit plays an important role.

The step-sizes η must obey certain constraints in order for the algorithm to work, in particular it must decay neither too slowly, nor too fast. There is one particular choice, for which our estimates are in fact the mean estimate of the expected value of the reward for each action i, which is a natural choice if the bandits are stationary.

The other question is what policy to use to take actions. We must take all actions often enough, so that we have good estimates for the expected reward of every bandit. One simple way to do it is to play the apparently best bandit most of the time, but to sometimes select bandits randomly. This is called ϵ-greedy action selection. This ensures that all actions are tried a sufficient number of times.

Definition 7.2.2. ϵ-greedy action selection

$$\hat{\pi}^*_\epsilon \triangleq (1 - \epsilon_t)\hat{\pi}^*_t + \epsilon_t \, \mathcal{U}\!nif(\mathcal{A}), \tag{7.2.9}$$

$$\hat{\pi}^*_t(i) = \mathbb{I}\left\{i \in \hat{\mathcal{A}}^*_t\right\} / |\hat{\mathcal{A}}^*_t|, \qquad \hat{\mathcal{A}}^*_t = \arg\max_{i \in \mathcal{A}} \mu_{t,i} \tag{7.2.10}$$

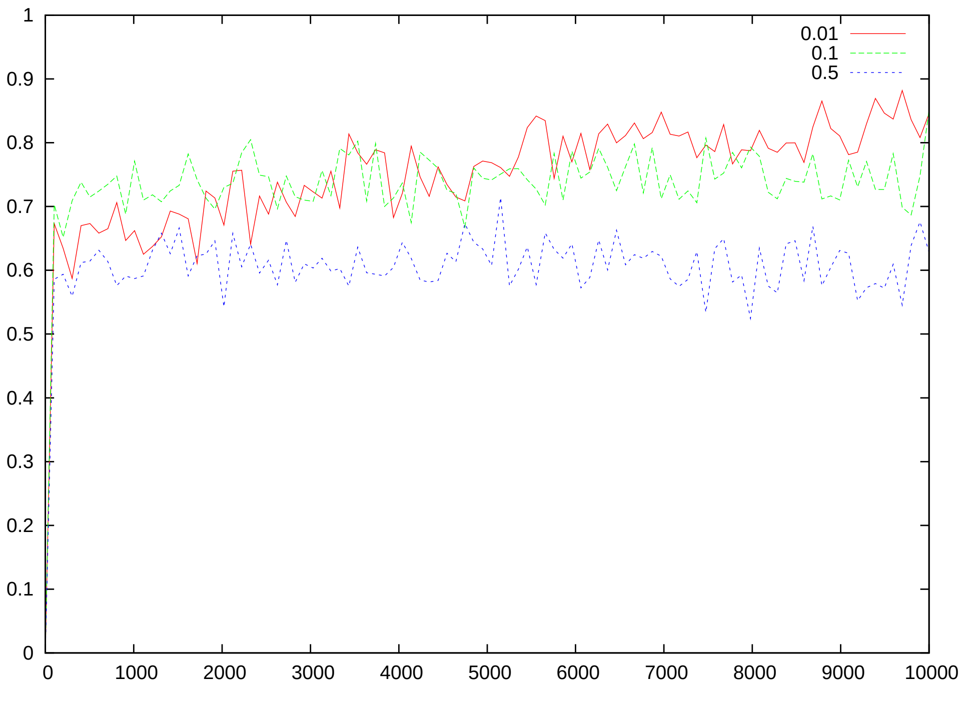


Figure 7.1: $\epsilon_t = 0.1$, $\eta \in \{0.01, 0.1, 0.5\}$.

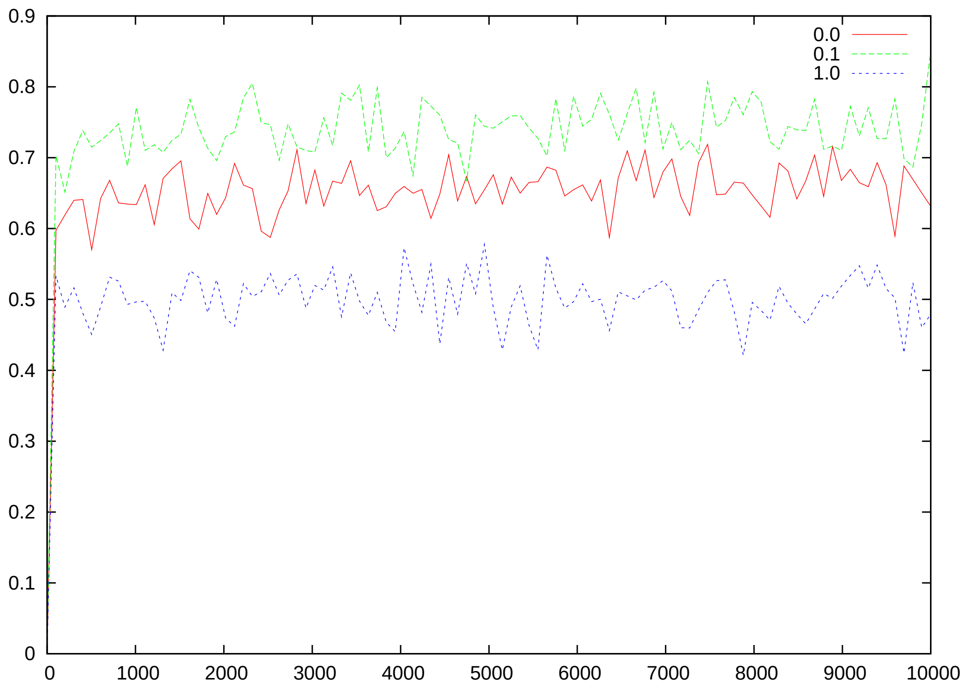


Figure 7.2: $\epsilon_t = \epsilon$, $\eta = 0.1$.

The main two parameters of the algorithm are randomness ϵ-greedy action selection and the step-size. Figures 7.1 and 7.2 show the average reward obtained, if we keep the step size α or the randomness ϵ fixed, respectively. We see that there the choice of values really affects convergence.

For a fixed ϵ, we find that larger values of α tend to give a better result eventually, while smaller values have a better initial performance. This is a natural trade-off, since large α appears to "learn" fast, but it also "forgets" quickly. That is, for a large α, our estimates mostly depend upon the last few rewards observed.

Things are not so clear-cut for the choice of ϵ. We see that the choice of $\epsilon = 0$, is significantly worse than $\epsilon = 0.1$. So, that appears to suggest that there is an optimal level of exploration. How should that be determined? Ideally, we should be able to to use the decision-theoretic solution seen earlier, but perhaps a good heuristic way of choosing ϵ may be good enough.

7.2.4 Decision-theoretic bandit process

The basic bandit process can be seen in Figure **??**. We can now define the general decision-theoretic bandit process, not restricted to independent Bernoulli bandits.

Definition 7.2.3. Let $\mathcal{A}$ be a set of actions, not necessarily finite. Let Ω be a set of possible parameter values, indexing a family of probability measures $\mathcal{F} = \{P_{\omega,a} \mid \omega \in \Omega, a \in \mathcal{A}\}$. There is some $\omega \in \Omega$ such that, whenever we take action $a_t = a$, we observe reward $r_t \in \mathcal{R} \subset \mathbb{R}$ with probability measure:

$$P_{\omega,a}(R) \triangleq \mathbb{P}_\omega(r_t \in R \mid a_t = a), \qquad R \subseteq \mathbb{R}. \tag{7.2.11}$$

Let β_1 be a prior distribution on Ω and let the posterior distributions be defined as:

$$\beta_{t+1}(B) \propto \int_B P_{\omega,a_t}(r_t)\, \mathrm{d}\beta_t(\omega). \tag{7.2.12}$$

The next belief is random, since it depends on the random quantity r_t. In fact, the probability of the next reward lying in R if $a_t = a$ is given by the following marginal distribution:

$$P_{\beta_t,a}(R) \triangleq \int_\Omega P_{\omega,a}(R) \,\mathrm{d}\beta_t(\omega). \tag{7.2.13}$$

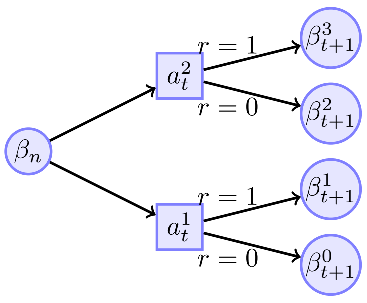


Figure 7.3: A partial view of the multi-stage process. Here, the probability that we obtain $r = 1$ if we take action $a_t = i$ is simply $P_{\beta_t,i}(\{1\})$.

Finally, as β_{t+1} deterministically depends on β_t, a_t, r_t, the probability of obtaining a particular next belief is the same as the probability of obtaining the corresponding rewards leading to the next belief. In more detail, we can write:

$$\mathbb{P}(\beta_{t+1} = \beta \mid \beta_t, a_t) = \int_{\mathcal{R}} \mathbb{I}\left\{\beta_t(\cdot \mid a_t, r_t = r) = \beta\right\} \mathrm{d}P_{\beta_t,a}(r). \tag{7.2.14}$$

In practice, although multiple reward sequences may lead to the same beliefs, we frequently ignore that possibility for simplicity. Then the process becomes a tree. A solution to the problem of what action to select is given by a backwards induction algorithm.

The backwards induction algorithm

$$U^*(\beta_t) = \max_{a_t} \mathbb{E}(r_t \mid \beta_t, a_t) + \sum_{\beta_{t+1}} \mathbb{P}(\beta_{t+1} \mid \beta_t, a_t) U^*(\beta_{t+1}). \tag{7.2.15}$$

kwards induction

The above equation is the *backwards induction* algorithm for bandits. If you look at this structure, you can see that next belief only depends on the current belief, action and reward, i.e. it satisfies the Markov property, as seen in Figure 7.3.[1] The intuition behind the algorithm lies in the following observation

$$\begin{aligned}
\mathbb{E}^\pi_{\beta_t}(U_t) &= \mathbb{E}^\pi_{\beta_t}\left(\sum_{k=t}^T r_k\right) && (7.2.16)\\
&= \mathbb{E}^\pi_{\beta_t}\left(r_t + \sum_{k=t+1}^T r_k\right) && (7.2.17)\\
&= \mathbb{E}^\pi_{\beta_t}(r_t) + \mathbb{E}^\pi_{\beta_t}(U_{t+1}) && (7.2.18)\\
&= \mathbb{E}^\pi_{\beta_t}(r_t) + \int_{\mathcal{B}} \mathbb{E}^\pi_{\beta_{t+1}}(U_{t+1}) \,\mathrm{d}\,\mathbb{P}(\beta_{t+1} \mid \beta_t, \pi) && (7.2.19)
\end{aligned}$$

[1] In fact, a decision-theoretic bandit process is a specific case of a Markov decision process

Optimising over all possible policies is possible since the remaining utility from time $t+1$ does not depend upon our previous decisions. This is due to the additive utility function, and allows us to do:

$$\max \mathbb{E}^{\pi}_{\beta_t}(U_t) = \max_{a_t} \mathbb{E}_{\beta_t}(r_t \mid a_t) + \int_{\mathcal{B}} \max_{\pi'} \mathbb{E}^{\pi'}_{\beta_{t+1}}(U_{t+1}) \, \mathrm{d}\,\mathbb{P}(\beta_{t+1} \mid \beta_t, a_t) \tag{7.2.20}$$

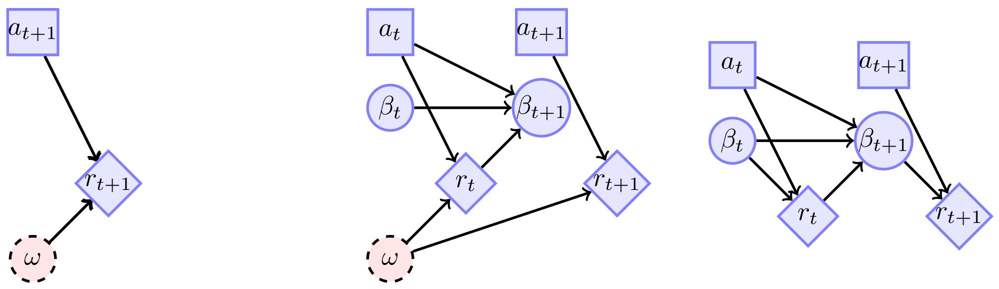


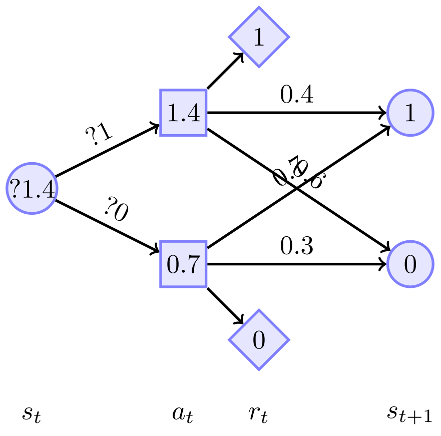
Figure 7.4: Three views of the bandit process. Figure **??** shows the basic bandit process, from the view of an external observer. The decision maker selects a_t, while the parameter ω of the process is hidden. It then obtains reward r_t. The process repeats for $t = 1, \ldots, T$. The decision-theoretic bandit process is shown in Figures **??** and **??**. While ω is not known, at each time step t we maintain a belief β_t on Ω. The reward distribution is then defined through our belief. In Figure **??**, we can see that complete process, where the dependency on ω is clear. In Figure **??**, we marginalise out ω and obtain a model where the transitions only depend on the current belief and action.

In reality, the reward depends only on the action and the unknown ω, as can be seen in Figure **??**. This is the point of view of an external observer. However, from the point of view of the decision maker, the distribution of ω only depends on his current belief. Consequently, the distribution of rewards also only depends on the current belief, as we can marginalise over ω. This gives rise to the decision-theoretic bandit process shown in Figure **??**. In the following section, we shall consider Markov decision processes more generally.

Illustration of backwards induction

Backwards induction (Dynamic programming)

```
for $t = 1, 2, \ldots$ and $\beta_t \in \mathcal{B}_t$ do
    $U^*_t(\beta_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}(r_t \mid \beta_t, a_t) + \sum_{\beta_{t+1} \in \mathcal{B}_{t+1}} \mathbb{P}(\beta_{t+1} \mid \beta_t, a_t) U^*_{t+1}(\beta_{t+1})$
end for
```



Exercise 17. What is the value $v_t(s_t)$ of the first state?

A 1.4

B 1.05

C 1.0

D 0.7

E 0

7.2.5 Heuristic algorithms for the n-armed bandit problem

Here we introduce two algorithms, UCB (upper confidence bound) and Thompson sampling, which are nearly optimal heuristics. Although the following algorithms are not optimal in the sense that the maximise Bayes-expected utility, they perform nearly as well as an oracle that knows the model parameters θ. In particular, if $\pi^*(\theta)$ is the policy that knows the true parameter,and π the UCB or Thompson sampling policy, then the expected difference in utility relative to the oracle[2] is

$$L(\pi, \theta) \triangleq \mathbb{E}_\theta^{\pi^*} \sum_{t=1}^T (r_t) - \mathbb{E}_\theta^{\pi} \sum_{t=1}^T (r_t) \leq O(\sqrt{T}) \forall \theta \in \Theta.$$

This general bound is parameter-independent, but there are also $O(\ln(T))$ bounds that depend on θ.

The UCB algorithm

For rewards in $[0, 1]$ we can apply the UCB algorithm introduced by Auer et al. [1]. This applies Hoeffding's inequality to construct a confidence interval around estimates of the mean reward of each arm, and simply picks the arm with the highest upper confidence interval, as shown in Algorithm 4.

[2]Also called the regret of the algorithm

Algorithm 4 UCB1

```
  **Input** $\mathcal{A}$
  $\hat{\theta}_{0,i} = 1, \forall i$
  **for** $t = 1, \ldots$ **do**
      $a_t = \arg\max_{i \in \mathcal{A}} \left\{ \hat{\theta}_{t-1,i} + \sqrt{\frac{2 \ln t}{N_{t-1,i}}} \right\}$
      $r_t \sim P_\theta(r \mid a_t)$ // play action and get reward
      /* update model */
      $N_{t,a_t} = N_{t-1,a_t} + 1$
      $\hat{\theta}_{t,a_t} = [N_{t-1,a_t} \theta_{t-1,a_t} + r_t] / N_{t,a_t}$
      $\forall i \neq a_t,\ N_{t,i} = N_{t-1,i},\ \hat{\theta}_{t,i} = \hat{\theta}_{t-1,i}$
  **end for**
```

The Thompson sampling algorithm

In the Bayesian setting, whenever we can define some prior belief β_0 over parameters Θ, we can use Thompson sampling, first introduced by Thompson[18]. The idea of this algorithm is to simply sample a parameter value $\hat{\theta}$ from the posterior, and then select the action that seems optimal with respect to the sample, as shown in Algorithm 5.

Algorithm 5 Thompson sampling

```
  **Input** $\mathcal{A}, \beta_0$
  **for** $t = 1, \ldots$ **do**
      $\hat{\theta} \sim \beta_{t-1}(\theta)$
      $a_t \in \arg\max_a \mathbb{E}_{\hat{\theta}}[r_t \mid a_t = a]$.
      $r_t \sim P_\theta(r \mid a_t)$ // play action and get reward
      $\beta_t(\theta) = \beta_{t-1}(\theta \mid a_t, r_t)$. // update model
  **end for**
```

7.3 Contextual Bandits

In the simplest bandit setting, our only information when selecting an arm is the sequence of previous plays and rewards obtained. However, in many cases we have more information whenever we draw an arm.

Example 35 (Clinical trials). Consider an example where we have some information x_t about an individual patient t, and we wish to administer a treatment a_t. For whichever treatment we administer, we can observe an outcome y_t. Our goal is to maximise expected utility.

Definition 7.3.1 (The contextual bandit problem.)**.** At time t,

- We observe $x_t \in \mathcal{X}$.
- We play $a_t \in \mathcal{A}$.
- We obtain $r_t \in \mathbb{R}$ with $r_t \mid a_t = a, x_t = x \sim P_\theta(r \mid a, x)$.

Example 36 (The linear bandit problem).

- $\mathcal{A} = [n]$, $\mathcal{X} = \mathbb{R}^k$, $\theta = (\theta_1, \ldots, \theta_n)$, $\theta_i \in \mathbb{R}^k$, $r \in \mathbb{R}$.
- $r \sim \mathcal{N}(\theta_a^\top x), 1)$

Example 37 (A clinical trial example). In this scenario, each individual is described by a real vector x_t, and the outcome is described by a logistic model. The reward is simply a known function of the action and outcome.

- $\mathcal{A} = [n]$, $\mathcal{X} = \mathbb{R}^k$, $\theta = (\theta_1, \ldots, \theta_n)$, $\theta_i \in \mathbb{R}^k$, $y \in \{0, 1\}$.
- $y \sim \mathcal{B}ernoulli(1/(1 + exp[-(\theta_a^\top x)^2])$.
- $r = U(a, y)$.

Algorithms for the contextual bandit problem

The simplest algorithm we can use is Thompson sampling, shown in Algorithm 6

Algorithm 6 Thompson sampling for contextual bandits

```
Input $\mathcal{A}, \beta_0$
for $t = 1, \ldots$ do
    $\hat{\theta} \sim \beta_{t-1}(\theta)$
    Observe $x_t$.
    $a_t \in \arg\max_a \mathbb{E}_{\hat{\theta}}[r_t \mid x_t, a_t = a]$.
    $r_t \sim P_\theta(r \mid a_t)$ // play action and get reward // update model
    $\beta_t(\theta) = \beta_{t-1}(\theta \mid a_t, r_t, x_t)$.
end for
```

We can also consider the full decision theoretic solution to Thompson sampling

Backwards induction in the contextual setting

```
for $n = 1, 2, \ldots$ and $s \in \mathcal{S}$ do
    $$\mathbb{E}(U_t \mid x_t, \beta_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}(r_t \mid x_t, \beta_t, a_t) + \sum_{\beta_{t+1}} \mathbb{P}(\beta_{t+1} \mid \beta_t, x_t, a_t)\, \mathbb{E}(U_{t+1} \mid \beta_{t+1})$$
end for
```

As β_{t+1} is a deterministic function of β_t, x_t, a_t, we can simply replace the sum in the right hand side as follows:

$$\mathbb{E}(U_{t+1} \mid x_t, a_t) = \sum_{\beta_{t+1}} \mathbb{P}(\beta_{t+1} \mid \beta_t, x_t, a_t)\, \mathbb{E}(U_{t+1} \mid \beta_{t+1}) \tag{7.3.1}$$

$$= \sum_{r_t} \mathbb{P}(r_t \mid \beta_t, x_t, a_t)\, \mathbb{E}[U_{t+1} \mid \beta_t(\cdot \mid r_t, x_t, a_t)] \tag{7.3.2}$$

$$\tag{7.3.3}$$

where $\mathbb{P}(r_t \mid \beta_t, x_t, a_t) = \int_\Theta P_\theta(r_t \mid x_t, a_t)\, d\beta_t(\theta)$ is the marginal reward distribution.

7.4 Case study: experiment design for clinical trials

While standard bandit problems are inherently interesting for computer-mediated tasks, they are not typical of experiment design problems that involve humans in the loop. In particular, you would expect humans to only be able to select and implement a small number of intervention policies.

EXAMPLE 38 (One-stage problems). In a classical one-stage experiment design problem, the experimenter only has a single observation

- Initial belief β_0
- Side information $\boldsymbol{x}$
- Simultaneously takes actions $\boldsymbol{a}$.
- Observes outcomes $\boldsymbol{y}$.

You can see this as a standard one-stage contextual bandit problem, but it is better to take advantage of the fact that all the variables are vectors i.e $.\boldsymbol{x} = x_{1:k}$. You can also see this as a parallelisation of the sequential problem. The goal here is typically to maximise expected utility after the data has been collected. Thus, the question is how to optimise the data collection process itself.

$$\mathbb{E}^{\pi}_{\beta_0}(U \mid \boldsymbol{x}) = \sum_{\boldsymbol{a},\boldsymbol{y}} \mathbb{P}_{\beta_0}(\boldsymbol{y} \mid \boldsymbol{a}, \boldsymbol{x}) \pi(\boldsymbol{a} \mid \boldsymbol{x}) \underbrace{\mathbb{E}^{\pi}_{\beta_0}(U \mid \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})}_{\text{post-hoc value}} \tag{7.4.1}$$

There are a few different typical objectives one could have for this type of design. The first might be, how to maximise expected information gain

Definition 7.4.1 (Expected information gain)**.**

$$\mathbb{E}^{\pi}_{\beta_0}(\mathbb{D}(\beta_1 \| \beta_0) \mid \boldsymbol{x}) = \sum_{\boldsymbol{a},\boldsymbol{y}} \mathbb{P}_{\beta_0}(\boldsymbol{y} \mid \boldsymbol{a}, \boldsymbol{x}) \pi(\boldsymbol{a} \mid \boldsymbol{x}) \mathbb{D}(\beta_0(\cdot \mid \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}) \| \beta_0) \tag{7.4.2}$$

As you can see, here there is no dependence on the policy. We just care about getting the maximal amount of information from our experiment

An alternative is to be able to maximise expected utility for the optimal policy after the observations have been seen.

Definition 7.4.2 (Expected utility of final policy)**.** For some simple reward function $\rho(x_t, y_t)$, maximise:

$$\mathbb{E}^{\pi}_{\beta_0}\left(\max_{\pi_1} \mathbb{E}^{\pi_1}_{\beta_1} \rho \middle| \boldsymbol{x}\right) = \sum_{\boldsymbol{a},\boldsymbol{y}} \mathbb{P}_{\beta_0}(\boldsymbol{y} \mid \boldsymbol{a}, \boldsymbol{x}) \pi(\boldsymbol{a} \mid \boldsymbol{x}) \max_{\pi_1} \mathbb{E}^{\pi_1}_{\beta_0}(\rho \mid \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{y}) \tag{7.4.3}$$

$$\mathbb{E}^{\pi_1}_{\beta_0}(\rho \mid \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{y}) = \sum_{a,x,y} \rho(a, y) \, \mathbb{P}_{\beta_1}(y \mid x, a) \pi_1(a \mid x) \, \mathbb{P}_{\beta_1}(x) \tag{7.4.4}$$

7.4.1 Practical approaches to experiment design

Unfortunately, a lot of the time it is not possible to simply select an appropriate prior distribution, select a model, etc, However, the same process can be used in practical scenarios. The procedure can be seen as follows.

Experiment design for a one-stage problem

- Select some model $\mathbb{P}$ for generating data. This can be based on historical data. For example, you can try and fit a neural network, a Gaussian process, or any other model on historical data.

- Select an inference and/or decision making algorithm λ for the task. For example, you may simply want to create a classifier, or you may want to do null-hypothesis testing. In all cases, your algorithm and decision making procedure should be fully defined at this point.

- Select a performance measure U.
- Generate data D from $\mathbb{P}$ and measure the performance of λ on D.

Experiment design for a multi-stage problem

Here we would like to distinguish two cases. The first, where we care about performance during the experiment (e.g. maximising the number of patients cured), or whether we care about the best policy we can find after the experiment has been concluded (e.g. finding the best treatment). There are many heuristic approaches we can follow.

- Use myopic (n-step) experiment design. When your utility is of the form $U = \sum_{k=t}^{T} r_t$, do not solve the complete problem. Instead, at each step t, find the policy π_t maximising $\mathbb{E}^{\pi_t}_{\beta_t} \sum_{k=t}^{t+n} r_t$. For $n = 1$, you obtain a simply myopic strategy.
- Use a simple heuristic like UCB (or Linear UCB), or Thompson sampling.
- If computing posteriors is hard, use bootstrapping-based Thompson sampling[11] or MCMC.

Index

.1 Graphical models

Graphical models are a very useful tool for modelling the relationship between multiple variables. The simplest such models, probabilistic graphical models (otherwise known as Bayesian networks) involve directed acyclic graphs between random variables. There are two other types of probabilistic models, factor graph and undirected graphical models, which are equivalent to each other, though not to directed models.

Graphical models

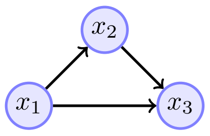


Figure 5: Graphical model (directed acyclic graph) for three variables.

Consider for example the model in Figure **??**. It involves three variables, x_1, x_2, x_3 and there are three arrows, which show how one variable depends on another. Simply put, if you think of each x_k as a stochastic function, then x_k's value only depends on the values of its parents, i.e. the nodes that are point to it. In this example, x_1 does not depend on any other variable, but the value of x_2 depends on the value of x_1. Such models are useful when we want to describe the joint probability distribution of all the variables in the collection.

The graphical model allows us to factorise the joint probability distribution of these random variables in a simplified manner. First, we define what we mean by a joint probability with respect to some probability measure P.

Joint probability

Let P is a probability measure on (Ω, Σ). Then let the random variable $\boldsymbol{x} = (x_1, \ldots, x_n)$ so that $\boldsymbol{x} : \Omega \to X$, $X = \prod_i X_i$. The joint probability of $\boldsymbol{x}$ can be written in terms of the underlying probability measure P:

$$\mathbb{P}(\boldsymbol{x} \in A) = P(\{\omega \in \Omega \mid \boldsymbol{x}(\omega) \in A\}).$$

When X_i are finite, we can typically write

$$\mathbb{P}(\boldsymbol{x} = \boldsymbol{a}) = P(\{\omega \in \Omega \mid \boldsymbol{x}(\omega) = \boldsymbol{a}\}),$$

for the probability that $x_i = a_i$ for all $i \in [n]$. Through the definition of conditional probability, we can always factorise the joint distribution of random variables as follows:

Factorisation

For any subsets $B \subset [n]$ and its complement C so that $\boldsymbol{x}_B = (x_i)_{i \in B}$, $\boldsymbol{x}_C = (x_i)_{i \notin B}$

$$\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x}_B \mid \boldsymbol{x}_C)\, \mathbb{P}(\boldsymbol{x}_C)$$

So we can write any joint distribution as

$$\mathbb{P}(x_1)\,\mathbb{P}(x_2 \mid x_1)\,\mathbb{P}(x_3 \mid x_1, x_2)\cdots\mathbb{P}(x_n \mid x_1, \ldots, x_{n-1}).$$

Although the above factorisation is always possible to do, sometimes our graphical model has a structure that makes the factors much simpler. In fact, the main reason for introducing graphical models is to represent dependencies between variables. For a given model, we can infer whether some variables are in fact dependent, independent, or conditionally independent.

Directed graphical models and conditional independence

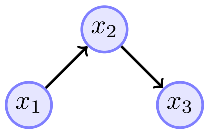


Figure 6: Graphical model for the factorisation $\mathbb{P}(x_3 \mid x_2)\,\mathbb{P}(x_2 \mid x_1)\,\mathbb{P}(x_1)$.

Conditional independence

We say x_i is conditionally independent of $\boldsymbol{x}_B$ given $\boldsymbol{x}_D$ and write $x_i \mid \boldsymbol{x}_D \perp\!\!\!\perp \boldsymbol{x}_B$ iff

$$\mathbb{P}(x_i, \boldsymbol{x}_B \mid \boldsymbol{x}_D) = \mathbb{P}(x_i \mid \boldsymbol{x}_D)\,\mathbb{P}(\boldsymbol{x}_B \mid \boldsymbol{x}_D).$$

Directed graphical models

A graphical model is a convenient way to represent conditional independence between variables. There are many variants of graphical models, whose name is context dependent. Other names used in the literature are probabilistic graphical models, Bayesian networks, causal graphs, or decision diagrams. In this set of notes we focus on directed graphical models that depict dependencies between ranom variables.

Definition .1.1 (Directed graphical model)**.** A collection of n random variables $x_i : \Omega \to X_i$, and let $X \triangleq \prod_i X_i$, with underlying probability measure P on Ω. Let $\boldsymbol{x} = (x_i)_{i=1}^n$ and for any subset $B \subset [n]$ let

$$\boldsymbol{x}_B \triangleq (x_i)_{i \in B} \tag{.1.1}$$

$$\boldsymbol{x}_{-j} \triangleq (x_i)_{i \neq i} \tag{.1.2}$$

In a graphical model, conditional independence is represented through directed edges.

Example 39 (Specifying a probability model). A graphical model does not specify the complete distribution, only an allowed factorisation. If we want to specify a complete distribution for the above graphical model of x_1, x_2, x_3, we can use the following notation:

$$x_1 \sim f \tag{.1.3}$$

$$x_2 \mid x_1 = a \sim g(a) \tag{.1.4}$$

$$x_3 \mid x_2 = b \sim h(b), \tag{.1.5}$$

where f, g, h are three different distributions, with g and h being specified through a single parameter.

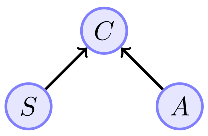


Figure 7: Smoking and lung cancer graphical model, where S: Smoking, C: cancer, A: asbestos exposure.

Example 40 (Smoking and lung cancer). It has been found by Lee[15] that lung incidence not only increases with both asbestos exposure and smoking. This is in agreement with the graphical model shown. The study actually found that there is an amplification effect, whereby smoking and asbestos exposure increases cancer risk by 28 times compared to non-smokers. This implies that the risk is not simply additive. The graphical model only tells us that there is a dependency, and does not describe the nature of this dependency precisely.

Explaining away

Even though S, A are independent, they become dependent once you know C. For example, let us say we know that you have cancer and that our model says that it's very unlikely to have cancer unless you either smoke or are exposed to asbestos. When we also learn that you do not have asbestos exposure, smoking becomes more likely. In either words, if cancer is caused by either smoking or asbestos, and we rule out asbestos, the only remaining explanation is smoking. This is what is generally called *explaining away.*

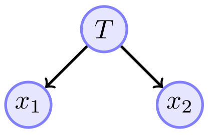


Figure 8: Time of arrival at work graphical model where T is a traffic jam and x_1 is the time John arrives at the office and x_2 is the time Jane arrives at the office.

Example 41 (Time of arrival at work). In this model, the arrival times of John and Jane may seem correlated. However, there is a common cause: The existence of a traffic jam. Whenever there is a traffic jam, both John and Jane are usually late. Whenever there is not a traffic jam, they are both mostly on time.

Conditional independence

Even though x_1, x_2 are correlated, they become independent once you know T.

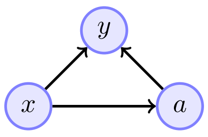


Figure 9: Kidney treatment model, where x: severity, y: result, a: treatment applied

	Treatment A	Treatment B
Small stones	87	270
Large stones	263	80

Severity	Treatment A	Treatment B
Small stones)	93%	87%
Large stones	73%	69%
Average	78%	83%

EXAMPLE 42 (Treatment effects). A curious example is that of applying one of two treatments for kidneys. In the data, it is clear that one treatment is best for both large and small stones. However, when the data is aggregated it appears as though treatment B is best. This is because treatment A is chosen much more frequently when the stones are large, and that's when both treatments perform worse. This apparent discrepancy is called *Simpson's paradox*

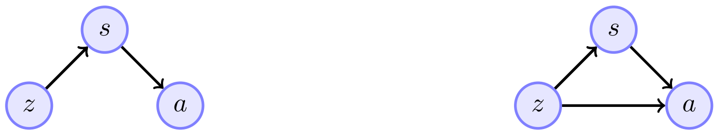
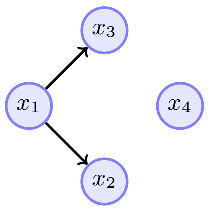


Figure 10: School admission graphical model, where z: gender, s: school applied to, a: whether you were admitted.

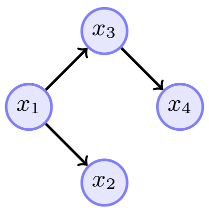
School	Male	Female
A	62%	82%
B	63%	68%
C	37%	34%
D	33%	35%
E	28%	24%
F	6%	7%
Average	*45%*	*38%*

EXAMPLE 43 (School admission). In this example, it appears as though female candidates have a lower acceptance rate than males. However what is missing is the fact that many more males are applying to easier schools. Thus, it is possible that the data is explainable by the fact that admission only reflects the difficulty of each school, and the overall gender imbalance is due to the choices made by the applicants. However, an alternative model is that the admissions process also explicitly takes gender into account. However, both of these models may be inadequate, as we do not have data about each individual applicant, such as their grades. We shall discuss this issue further when we talk about causality, confounding variables and counterfactuals.



EXERCISE 18. Factorise the following graphical model.

$$\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(x_1)\,\mathbb{P}(x_2 \mid x_1)\,\mathbb{P}(x_3 \mid x_1)\,\mathbb{P}(x_4)$$

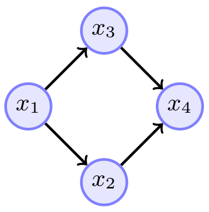


Exercise 19. Factorise the following graphical model.

$$\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(x_1)\,\mathbb{P}(x_2 \mid x_1)\,\mathbb{P}(x_3 \mid x_1)\,\mathbb{P}(x_4 \mid x_3)$$

Exercise 20. What dependencies does the following factorisation imply?

$$\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(x_1)\,\mathbb{P}(x_2 \mid x_1)\,\mathbb{P}(x_3 \mid x_1)\,\mathbb{P}(x_4 \mid x_2, x_3)$$



⚠ **Deciding conditional independence**

There is an algorithm for deciding conditional independence of any two variables in a graphical model. However, this is beyond the scope of these notes. Here, we shall just use these models as a way to encode dependencies that we assume exist.

.1.1 Inference and prediction in graphical models

Inference and prediction in graphical models

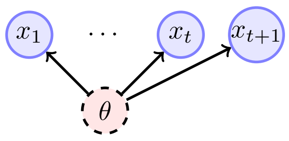


Figure 11: Inference and prediction in a graphical model

.

In this example, $x_1, \ldots, x_t$ are all i.i.d, drawn from the distribution $P_\theta(x_t) = \mathbb{P}(x_t \mid \theta)$.

Inference of latent variables

$$\mathbb{P}(\theta \mid x_1, \ldots, x_t)$$

Inference in graphical models typically refers to the problem of estimating the distribution of some *hiddeen* or *latent* variable from data. These variables are generally thought of as two types:

- Model parameters. These generally do not change over time. One example is estimating the mean of a Gaussian distribution from data.

- System states. These typically are time-indexed. We will see further examples of such variables when we discuss latent variable models. On example is inferring location from GPS measurements.

Prediction

$$\mathbb{P}(x_{t+1} \mid x_1, \ldots, x_t) = \int_\Theta \mathbb{P}(x_{t+1} \mid \theta) \, \mathrm{d}\, \mathbb{P}(\theta \mid x_1, \ldots, x_t)$$

Prediction is a type of inference, but differs in that the variable whose distribution we wish to estimate is only temporarily not observed: we can actually obtain its value in the future. Thus, a prediction is always testable!

Coin tossing, revisited

Example 44. The Beta-Bernoulli prior

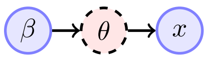


Figure 12: Graphical model for a Beta-Bernoulli prior

$$\theta \sim \mathcal{B}eta(\beta_1, \beta_2), \qquad \text{i.e. } \beta \text{ are Beta distribution parameters} \tag{.1.6}$$
$$x \mid \theta \sim \mathcal{B}ernoulli(\theta), \qquad \text{i.e. } P_\theta(x) \text{ is a Bernoulli} \tag{.1.7}$$

In this example, it is obvious why we use the notation above for describing hierarchical models. We simply state what is the distribution on one variable conditioned on the other variables. Here, β is fixed, and it is something we can choose arbitrarily. The data x is observed, while the parameter θ remains *latent*. Using Bayes theorem, we can derive the distribution for $\beta(\theta \mid x)$. In this particular case, we elide referring to a sample $x_1. \ldots, x_t$ as they are all i.i.d. *latent*

Example 45. The n-meteorologists problem (continuation of Exercise 8)

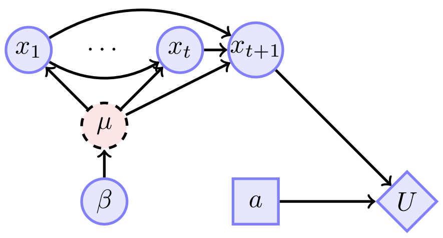


Figure 13: Inference, prediction and decisions in a graphical model.

In this problem, we allow each model's predictions to have arbitrary dependences with the past i.e.

$$P_\mu(x_{t+1} \mid x_t, x_{t-1}, \ldots, x_1).$$

For inference, the details of how each model works are not important: just the probability of the next observation given the previous ones.

.1.2 Testing conditional independence

Measuring independence

The simplest way to measure independence is by looking at whether or not the distribution of the possibly dependent variable changes when we change the value of the other variables.

Theorem .1.1. *If* $x_i \mid \boldsymbol{x}_D \perp\!\!\!\perp \boldsymbol{x}_B$ *then*

$$\mathbb{P}(x_i \mid \boldsymbol{x}_B, \boldsymbol{x}_D) = \mathbb{P}(x_i \mid \boldsymbol{x}_D)$$

This implies

$$\mathbb{P}(x_i \mid \boldsymbol{x}_B = b, \boldsymbol{x}_D) = \mathbb{P}(x_i \mid \boldsymbol{x}_B = b', \boldsymbol{x}_D)$$

so we can measure independence by seeing how the distribution of x_i changes when we vary $\boldsymbol{x}_B$, keeping $\boldsymbol{x}_D$ fixed. For any given model, there is either a dependence or there is not. However, sometimes we might be able to tolerate some amount of dependence. Thus, we can simply measure the deviation from independence through a metric or divergence on distributions.

Example 46.

$$\| \mathbb{P}(a \mid y, z) - \mathbb{P}(a \mid y)\|_1$$

which for discrete a, y, z is:

$$\max_{i,j} \| \mathbb{P}(a \mid y = i, z = j) - \mathbb{P}(a \mid y = i)\|_1 = \max_{i,j} \| \sum_k \mathbb{P}(a = k \mid y = i, z = j) - \mathbb{P}(a = k \mid y = i)\|_1.$$

See also `src/fairness/ci_test.py`

Example 47. An alternative model for coin-tossing This is an elaboration of Example 13 for hypothesis testing.

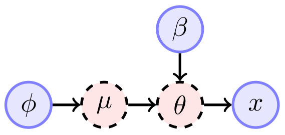


Figure 14: Graphical model for a hierarchical prior

- μ_1: A Beta-Bernoulli model with $\mathcal{B}eta(\beta_1, \beta_2)$
- μ_0: The coin is fair.

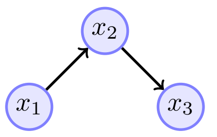
$$\theta \mid \mu = \mu_0 \sim \mathcal{D}(0.5), \qquad \text{i.e. } \theta \text{ is always } 0.5 \tag{.1.8}$$
$$\theta \mid \mu = \mu_1 \sim \mathcal{B}eta(\beta_1, \beta_2), \qquad \text{i.e. } \theta \text{ has a Beta distribution} \tag{.1.9}$$
$$x \mid \theta \sim \mathcal{B}ernoulli(\theta), \qquad \text{i.e. } P_\theta(x) \text{ is Bernoulli} \tag{.1.10}$$

Here the posterior over the two models is simply

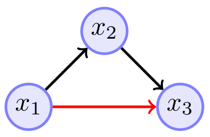
$$\phi(\mu_0 \mid x) = \frac{P_{0.5}(x)\phi(\mu_0)}{P_{0.5}(x)\phi(\mu_0) + \mathbb{P}_{\mu_1}(x)\phi(\mu_1)}, \qquad \mathbb{P}_{\mu_1(x)} = \int_0^1 P_\theta(x) \, \mathrm{d}\beta(\theta).$$

Bayesian testing of independence

For a given distributional model P_θ, conditional independence either holds or does not. Consider the set of model parameters Θ_0 where, for each parameter $\theta \in \Theta_0$, we have a conditional independence condition, while Θ_1 may corresponds to models where there may not be independence. To make this more concrete, let's give an example.



(a) Θ_0 assumes independence



(b) Θ_1 does *not* assume independence

Figure 15: The two alternative models

Example 48. Assume data $D = \left\{x_1^t, x_2^t, x_3^t \mid t = 1, \ldots, T\right\}$ with $x_i^t \in \{0, 1\}$. First consider model Θ_0 where the following conditional independence holds

$$P_\theta(x_3 \mid x_2, x_1) = P_\theta(x_2 \mid x_1), \qquad \forall \theta \in \Theta_0.$$

In the alternative model Θ_1 there is no independence assumption. So the likelihood for either a model in either set is

$$P_\theta(D) = \prod_t P_\theta(x_3^t \mid x_2^t) P_\theta(x_2^t \mid x_1^t) P_\theta(x_1^t), \qquad \theta \in \Theta_0 \tag{.1.11}$$

$$P_\theta(D) = \prod_t P_\theta(x_3^t \mid x_2^t, x_1^t) P_\theta(x_2^t \mid x_1^t) P_\theta(x_1^t), \qquad \theta \in \Theta_1 \tag{.1.12}$$

The parameters for this example can be defined as follows

$$\theta_1 \triangleq P_\theta(x_1^t = 1) \tag{μ_0, μ_1}$$

$$\theta_{2|1}^i \triangleq P_\theta(x_2^t = 1 \mid x_1^t = i) \tag{μ_0, μ_1}$$

$$\theta_{3|2}^j \triangleq P_\theta(x_3^t = 1 \mid x_2^t = j) \tag{μ_0}$$

$$\theta_{3|2,1}^{i,j} \triangleq P_\theta(x_3^t = 1 \mid x_2^t = j, x_1^t = i) \tag{μ_1}$$

We model each one of these parameters with a separate Beta-Bernoulli distribution.

.1.3 Hierarchical Bayesian models

Given some data D, the Bayesian approach would involve specifying a hierarchical prior β so that $\phi(\mu_1) = 1 - \phi(\mu_0)$ specifies a probability on the two model structures, while for the i-th model we define a prior $\beta_i(\theta)$ over Θ_i, so that we obtain the following hierarchical model

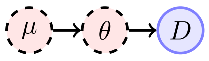


Figure 16: Hierarchical model.

Here the specific model μ is unobserved, as well as its parameters θ. Only the data D is observed. Our prior distribution is omitted from the graph.

$$\mu_i \sim \phi \tag{.1.13}$$

$$\theta \mid \mu = \mu_i \sim \beta_i \tag{.1.14}$$

Marginal likelihood

This gives the the following marginal likelihood for the combined models and each of the models respectively.

$$\mathbb{P}_\phi(D) = \phi(\mu_0)\,\mathbb{P}_{\mu_0}(D) + \phi(\mu_1)\,\mathbb{P}_{\mu_1}(D) \tag{.1.15}$$

$$\mathbb{P}_{\mu_i}(D) = \int_{\Theta_i} P_\theta(D)\,\mathrm{d}\beta_i(\theta). \tag{.1.16}$$

Model posterior

$$\phi(\mu \mid D) = \frac{\mathbb{P}_\mu(D)\phi(\mu)}{\sum_i \mathbb{P}_{\mu_i}(D)\phi(\mu_i)} \tag{.1.17}$$

Calculating the marginal likelihood
Generally speaking, calculating the marginal likelihood for a model with an uncountable parameter set is hard. However, conjugate models admit closed form solutions and efficient calculations. Firstly, let's rewrite the marginal likelihood in terms of an integral Monte-Carlo approximation.

Monte-Carlo approximation

$$\int_\Theta P_\theta(D)\,\mathrm{d}\beta(\theta) \approx \frac{1}{N}\sum_{n=1}^N P_{\theta_n}(D) + O(1/\sqrt{N}), \qquad \theta_n \sim \beta \tag{.1.18}$$

Even though this approximation is reasonable at first glance, the problem is that the leading constant of the error scales approximately proportionally to the maximum likelihood $\max_\theta P_\theta(D)$. This is a consequence of Hoeffding's inequality (**??**). Thus, the more data we have the more samples we need to get a good approximation with this simple Monte Carlo approach. For that reason, one typically uses a sample from a proposal distribution ψ which is different from β. Then it holds

Importance sampling

For any two measures β, ψ on Θ, we can write:

$$\int_\Theta P_\theta(D)\,\mathrm{d}\beta(\theta) = \int_\Theta P_\theta(D)\frac{\mathrm{d}\psi(\theta)}{\mathrm{d}\psi(\theta)}\,\mathrm{d}\beta(\theta) = \int_\Theta P_\theta(D)\frac{\mathrm{d}\beta(\theta)}{\mathrm{d}\psi(\theta)}\,\mathrm{d}\psi(\theta) \approx \sum_{n=1}^N P_\theta(D)\frac{\mathrm{d}\beta(\theta_n)}{\mathrm{d}\psi(\theta_n)}, \quad \theta_n \sim \psi \tag{.1.19}$$

This allows us to estimate the marginal likelihood with respect to a belief β by sampling from an alternative belief ψ.

Sequential updating of the marginal likelihood

$$
\begin{aligned}
\mathbb{P}_\beta(D) &= \mathbb{P}_\beta(x_1, \ldots, x_T) && (.1.20)\\
&= \mathbb{P}_\beta(x_2, \ldots, x_T \mid x_1)\,\mathbb{P}_\beta(x_1) && (.1.21)\\
&= \prod_{t=1}^{T} \mathbb{P}_\beta(x_t \mid x_1, \ldots, x_{t-1}) && (.1.22)\\
&= \prod_{t=1}^{T} \int_\Theta P_{\theta_n}(x_t)\,\mathrm{d}\,\underbrace{\beta(\theta \mid x_1, \ldots, x_{t-1})}_{\text{posterior at time } t} && (.1.23)
\end{aligned}
$$

The nice thing about this break down is that for a simple model such as Beta-Bernoulli, the individual datapoint marginal likelihoods are easy to compute

EXAMPLE 49 (Beta-Bernoulli). The marginal predictive distribution for a Beta-Bernoulli prior is

$$\mathbb{P}_\beta(x_t = 1 \mid x_1, \ldots, x_{t-1}) = \frac{\alpha_t}{\alpha_t + \beta_t},$$

with $\alpha_t = \alpha_0 + \sum_{n=1}^{t-1} x_n, \quad \beta_t = \beta_0 + \sum_{n=1}^{t-1}(1 - x_n)$

Further reading

Python sources

- A simple python measure of conditional independence `src/fairness/ci_test.py`
- A simple test for discrete Bayesian network `src/fairness/DirichletTest.py`
- Using the PyMC package `https://docs.pymc.io/notebooks/Bayes_factor.html`

Bibliography

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256, 2002.

[2] Craig M. Bennett, George L. Wolford, and Michael B. Miller. The principled control of false positives in neuroimaging. *Social cognitive and affective neuroscience*, 4 4:417–22, 2009. URL `https://pdfs.semanticscholar.org/19c3/d8b67564d0e287a43b1e7e0f496eb1e8a945.pdf`.

[3] Craig M Bennett, Abigail A Baird, Michael B Miller, and George L Wolford. Journal of serendipitous and unexpected results. *Journal of Serendipitous and Unexpected Results (jsur. org)-Vol*, 1(1):1–5, 2012. URL `https://teenspecies.github.io/pdfs/NeuralCorrelates.pdf`.

[4] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. URL `citeseer.nj.nec.com/breiman96bagging.html`.

[5] Philip Dawid. The decision-theoretic approach to causal inference. *Causality: Statistical perspectives and applications*, pages 25–42, 2012. URL `https://arxiv.org/pdf/1405.2292.pdf`.

[6] Christos Dimitrakakis, Blaine Nelson, Zuhe Zhang, Aikaterini Mitrokotsa, and Benjamin I. P. Rubinstein. Differential privacy for bayesian inference through posterior sampling. *Journal of Machine Learning Research*, 18(1):343–381, 2017.

[7] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.

[8] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[10] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349 (6248):636–638, 2015.

[11] Dean Eckles and Maurits Kaptein. Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009*, 2014.

[12] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951.

[13] Simson L. Garfinkel and Philip Leclerc. Randomness concerns when deploying differential privacy, 2020.

[14] Jason Hartford, Greg Lewis, Kevin Leyton-Brown, and Matt Taddy. Counterfactual prediction with deep instrumental variables networks. Technical Report 1612.09596, arXiv, 2016.

[15] PN Lee. Relation between exposure to asbestos and smoking jointly and the risk of lung cancer. *Occupational and environmental medicine*, 58(3):145–153, 2001.

[16] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[17] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International, 1998.

[18] W.R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of two Samples. *Biometrika*, 25(3-4):285–294, 1933.

[19] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. ISSN 01621459. URL `http://www.jstor.org/stable/2283137`.

[20] Mengmeng Yang, Lingjuan Lyu, Jun Zhao, Tianqing Zhu, and Kwok-Yan Lam. Local differential privacy and its applications: A comprehensive survey. *arXiv preprint arXiv:2008.03686*, 2020.