

Machine learning in science and society

From automated science to beneficial artificial intelligence

Christos Dimitrakakis

September 22, 2019

Contents

1	Introduction	7
1.1	Introduction to machine learning	8
1.1.1	Data analysis, learning and planning	8
1.1.2	Experiment design	12
1.1.3	Bayesian inference.	14
1.1.4	Course overview	17
2	Simple decision problems	21
2.1	Nearest neighbours	22
2.2	Reproducibility	27
2.2.1	The human as an algorithm	30
2.2.2	Algorithmic sensitivity	32
2.2.3	Beyond the data you have: simulation and replication	35
2.3	Beliefs and probabilities	38
2.3.1	Probability and Bayesian inference	41
2.4	Hierarchies of decision making problems	47
2.4.1	Simple decision problems	47
2.4.2	Decision rules	50
2.4.3	Statistical testing	51
2.5	Formalising Classification problems	58
2.6	Classification with stochastic gradient descent	62
2.6.1	Neural network models	63
2.7	Naive Bayes classifiers	67
3	Privacy	71
3.1	Database access models	72
3.2	Privacy in databases	74
3.3	k -anonymity	75
3.4	Differential privacy	76
3.4.1	Other differentially private mechanisms	82
3.4.2	Utility of queries	85
3.4.3	Privacy and reproducibility	85
4	Fairness	89
4.1	Graphical models	90
4.1.1	Testing conditional independence	94
4.1.2	Hierarchical Bayesian models	96
4.2	Fairness in machine learning	99

4.3	Concepts of fairness	102
4.3.1	Group fairness and conditional independence	102
4.3.2	Individual fairness and meritocracy	103
4.3.3	A unifying view of fairness	104
4.3.4	Bayesian fairness	105
4.3.5	Further reading	106
4.4	Project: Credit risk for mortgages	108
4.4.1	Deadline 1: September 18	108
4.4.2	Deadline 2: October 2	109
4.4.3	Deadline 3: October 16	109
5	Recommendation systems	111
5.1	Recommendation systems	112
5.1.1	Least squares representation	116
5.1.2	Preferences as a latent variable	117
5.1.3	The recommendation problem	118
5.2	More fun with latent variable models	120
5.3	Social networks	124
5.4	Sequential structures	125
6	Causality	127
6.1	Introduction	128
6.1.1	Decision diagrams	130
6.1.2	Common structural assumptions	131
6.2	Interventions	135
6.3	Policy evaluation and optimisation	136
6.4	Individual effects and counterfactuals	139
6.4.1	Disturbances and structural equation models	140
6.4.2	Example: Learning instrumental variables	141
6.4.3	Discussion	142
6.4.4	Exercises	142
7	Bandit problems and experiment design	145
7.1	Introduction	146
7.2	Bandit problems	146
7.2.1	An example: Bernoulli bandits	148
7.2.2	The stochastic n -armed bandit problem	149
7.2.3	Estimation and Robbins-Monro approximation	150
7.2.4	Decision-theoretic bandit process	152
7.2.5	Heuristic algorithms for the n -armed bandit problem	155
7.3	Contextual Bandits	156
7.4	Case study: experiment design for clinical trials	157
7.4.1	Practical approaches to experiment design	158
8	Markov decision processes	161
8.1	Markov decision processes and reinforcement learning	162
8.1.1	Value functions	164
8.2	Finite horizon, undiscounted problems	165
8.2.1	Policy evaluation	165
8.2.2	Monte-Carlo policy evaluation	166

CONTENTS

5

8.2.3	Backwards induction policy evaluation	167
8.2.4	Backwards induction policy optimisation	168
8.3	Infinite-horizon	169
8.3.1	Examples	169
8.3.2	MDP Algorithms	171
9	Index	175

Chapter 1

Introduction

1.1 Introduction to machine learning

What are the central problems in machine learning?

Problems in machine learning are similar to problems in science. Scientists must plan experiments intelligently and collect data. They 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.

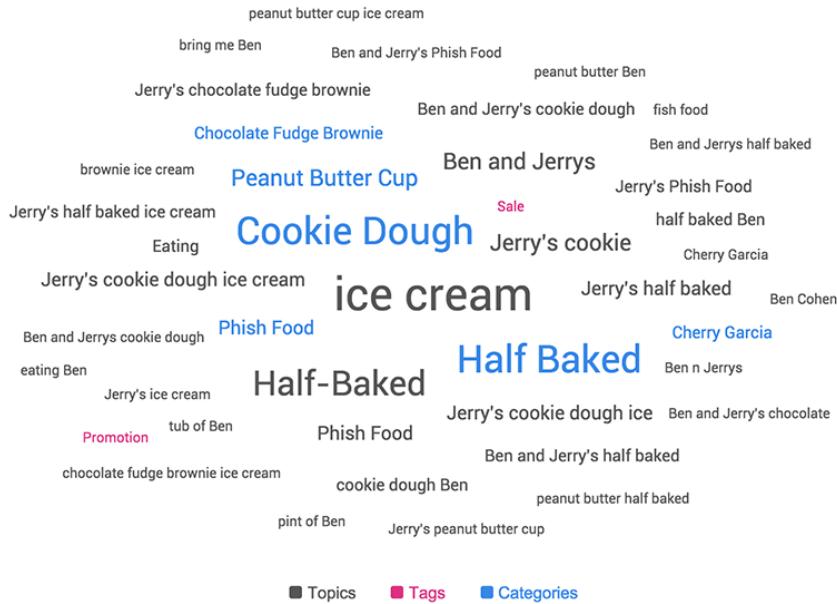
We must be careful, however. In many cases we need to be able to interpret what our model tells us. We also must make sure that the any results we obtain are reproducible. This is something that we shall emphasize in this course.

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 where 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.

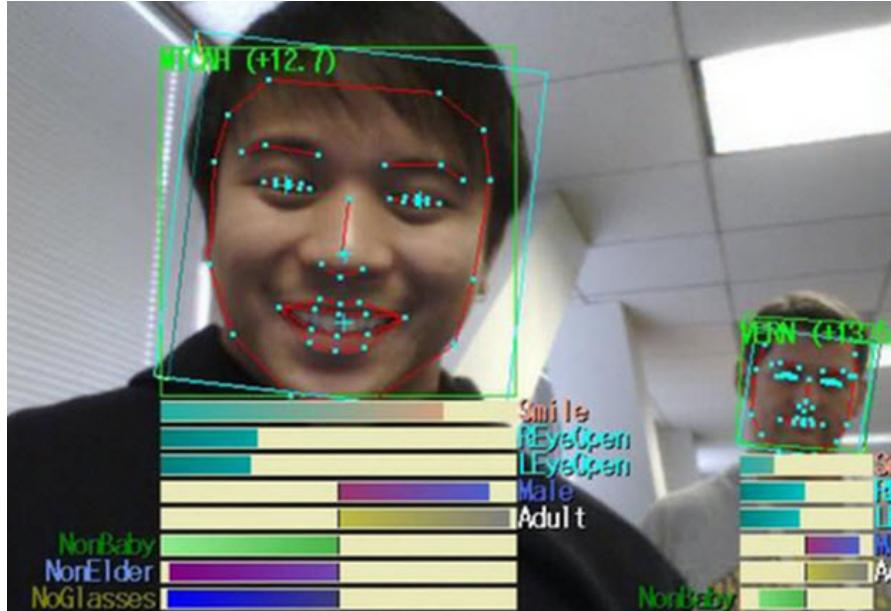
1.1.1 Data analysis, learning and planning

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



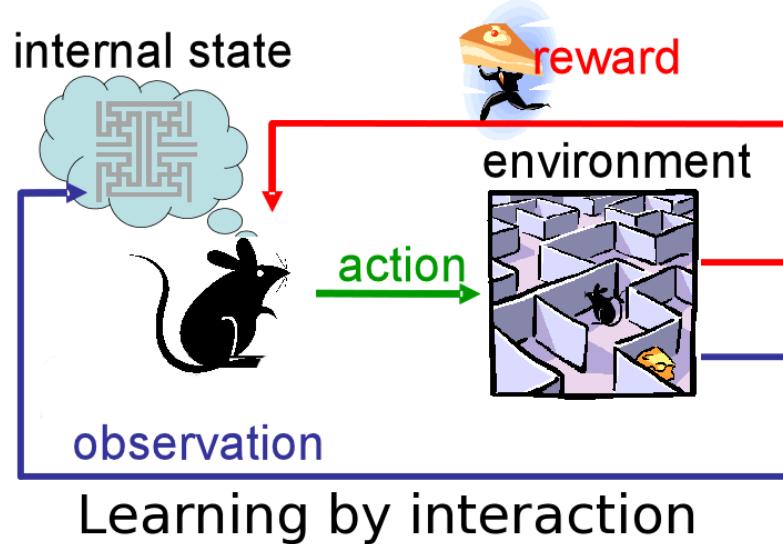
A supervised learning problem: object recognition

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?



Reinforcement learning

Take actions a_1, \dots, a_t , so as to maximise utility $U = \sum_{t=1}^T r_t$

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.

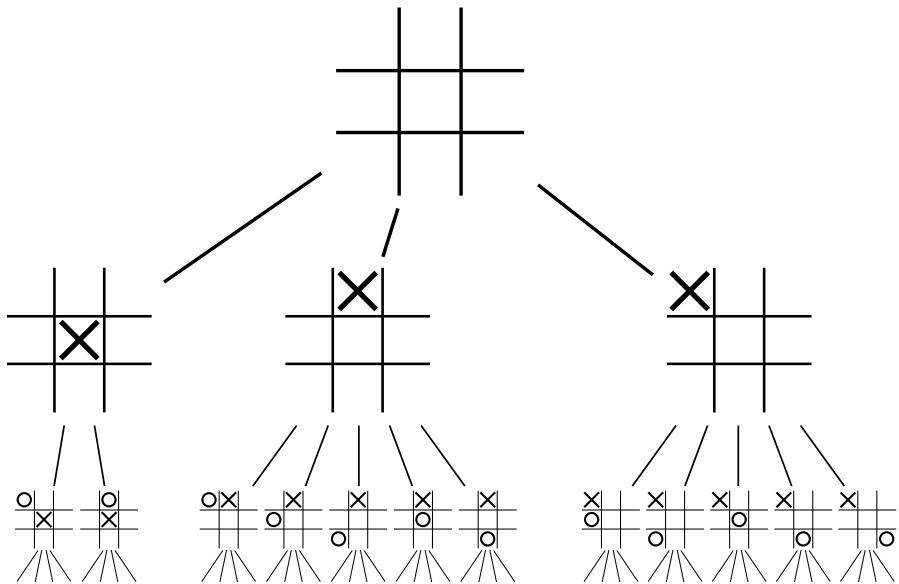
An example in robotics is trying to teach a robot to flip pancakes. One easy thing we can try is to show the robot how to do it, and then let it just copy the demonstrated movement. However, this doesn't work! The robot needs to explore variations of the movement, until it manages to successfully flip 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

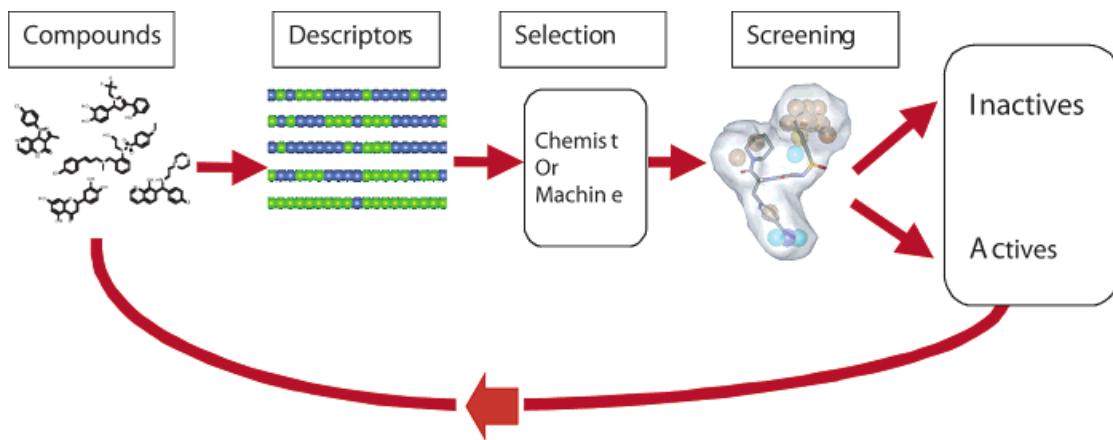
An example that typifies trial and error learning are bandit problems. Imagine that you are in a Casino and you wish to maximise the amount of money you make during the night. There are a lot of machines to play. If you knew which one was the best, then you'd just play it all night long. However, you 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.

Adam, the robot scientist



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??

Drug discovery



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 idea 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

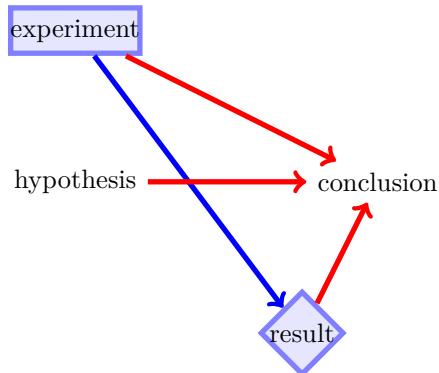


Figure 1.1: Dependence diagram between selection of an experiment, formulation of a hypothesis, and drawing of a conclusion. The result depends only on the experiment. However, the conclusion depends on the experiment, hypothesis and the obtained result. The red lines indicate computational dependencies,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

Tycho Brahe's minute eye measurements

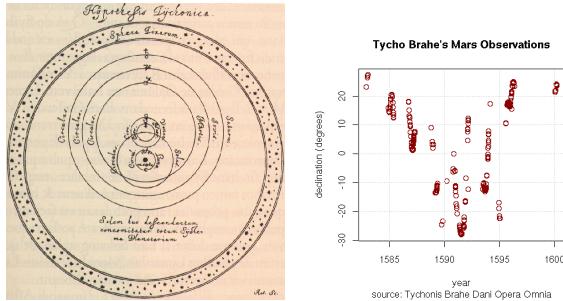
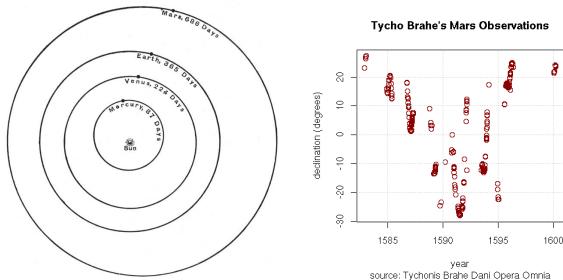


Figure 1.2: Tycho's measurements of the orbit of Mars and the conclusion about the actual orbits, under the assumption of an earth-centric universe with circular orbits.

- Hypothesis: Earth-centric, Circular orbits
- Conclusion: *Specific* circular orbits

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.

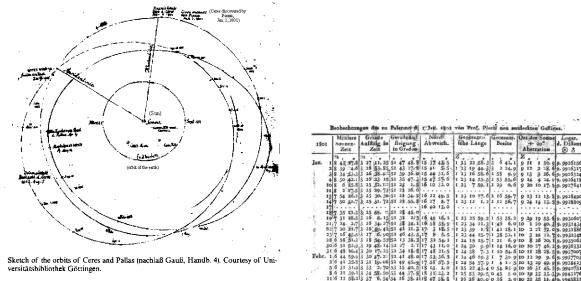
Johannes Kepler's alternative hypothesis



- Hypothesis: Circular *or* elliptic orbits
- Conclusion: Specific *elliptic* orbits

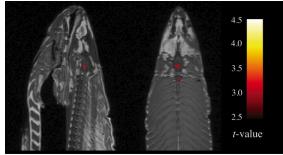
Kepler had a more general hypothesis: that orbits could be circular or elliptic, and he actually accepted that the planets orbited the sun. This led him to the broadly correct model of all planets being in elliptical orbits around the sun. However, the actual verification that all things do not revolve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

200 years later, Gauss formalised this statistically



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).

A warning: The dead salmon mirage



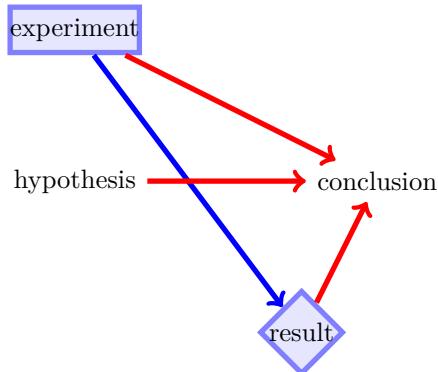
It is quite easy to draw the wrong conclusions from applying machine learning / statistics to your data. For example, it was fashionable to perform fMRI studies in humans to see whether some neurons have a particular functional role. There were even articles saying that "we found the neurons encoding for Angelina Jolie". So some scientists tried to replicate those results. They took a dead salmon, and put it in 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 significantly antly different from what people actually do in the field. [src/reproducibility/mri_analysis.ipynb](#)

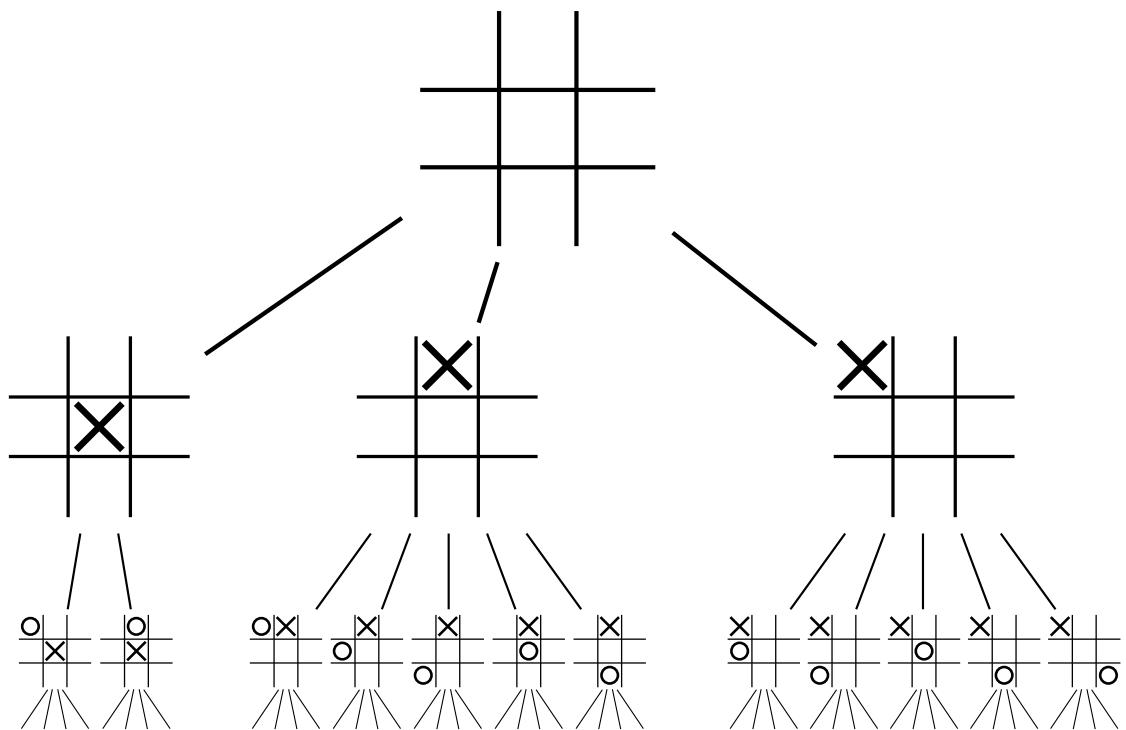
Planning future experiments

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 Course 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.

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

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 $f : \mathcal{X} \rightarrow \mathcal{Y}$ 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

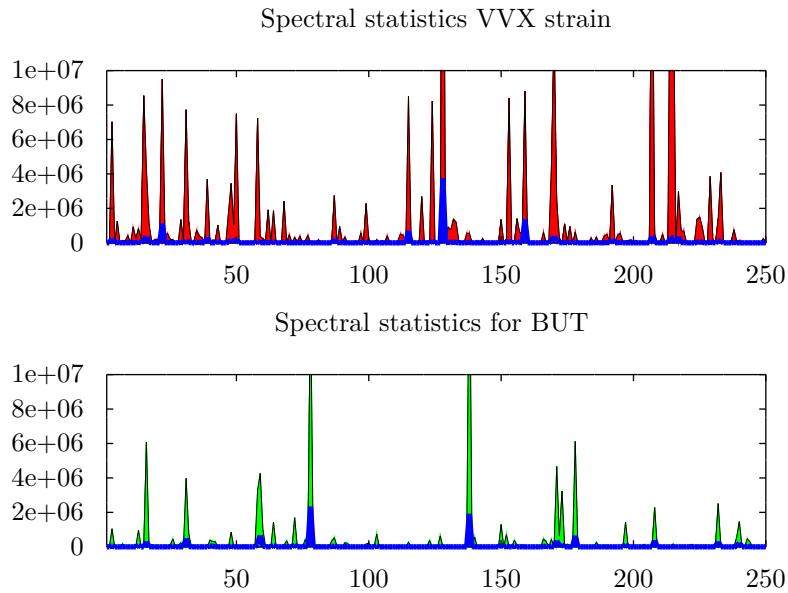
Simple decision problems

This chapter deals with simple decision problems, whereby a decision maker (DM) makes a simple choice among many. In some of these 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?

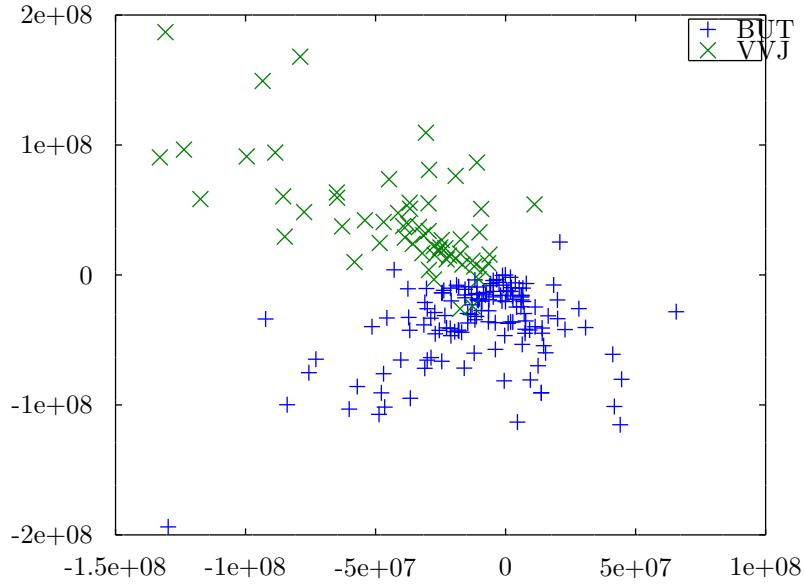
2.1 Nearest neighbours

Discriminating between diseases



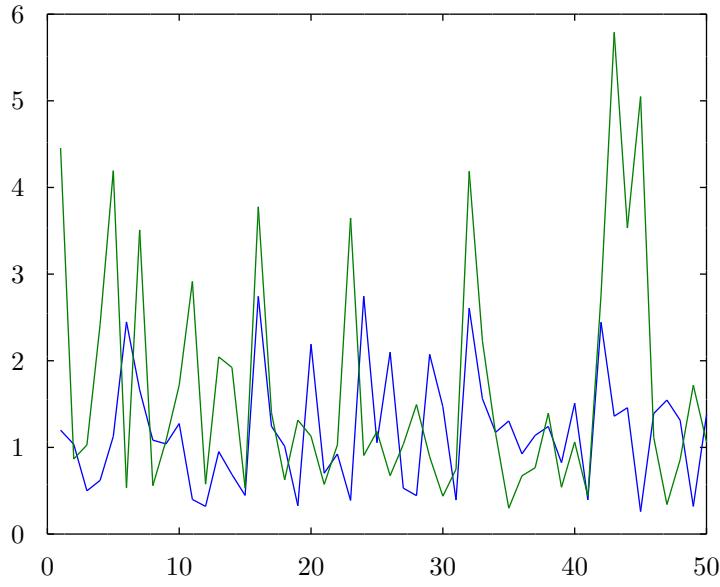
Let's tackle the problem of discriminating between different disease vectors. Ideally, we'd like to have a simple test that tells us what ails us. One kind of test is mass spectrometry. This graph shows spectrometry results for two types of bacteria. There is plenty of variation within each type, both 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 description 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 $D = \{(x_1, y_1), \dots, (x_T, y_T)\}$, $k \geq 1$, $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$, new point $x \in \mathcal{X}$
 - 2: $D = \text{Sort}(D, d)$ % 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** $\mathbf{p} \triangleq (p_1, \dots, p_k)$
-

Algorithm parameters

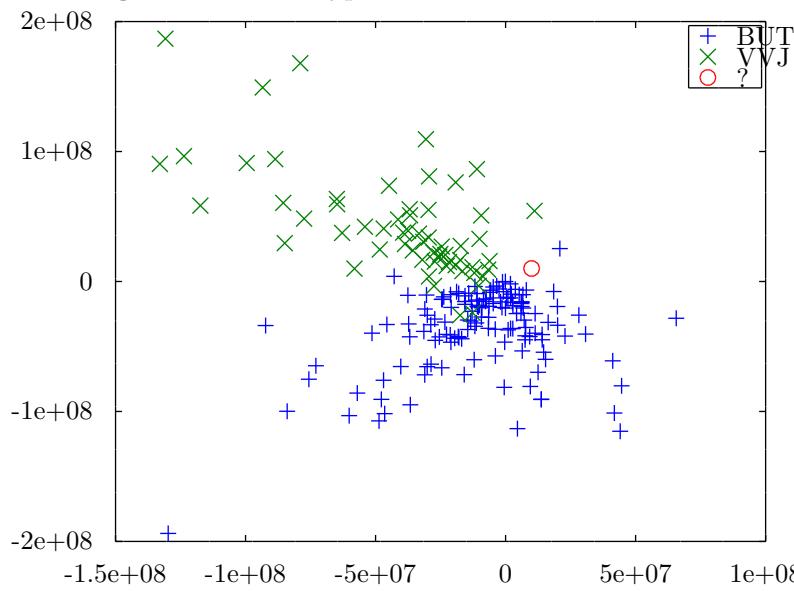
In order to use the algorithm, we must specify some parameters, namely.

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



Figure 2.1: The nearest neighbours algorithm was introduced by Fix and Hodges Jr¹³, who also proved consistency properties.

Nearest neighbour: What type is the new bacterium?



Given that the + points represent the BUT type, and the x 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:

$$\pi(a | x) = \mathbb{I}\{p_a \geq p_y \forall y\}, \quad p = \text{k-NN}(D, k, d, x)$$

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 k-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 hyperparameters 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?

2.2 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.



A simple example is the 2016 election. 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 $\lambda : \mathcal{D} \rightarrow \Pi$, 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.

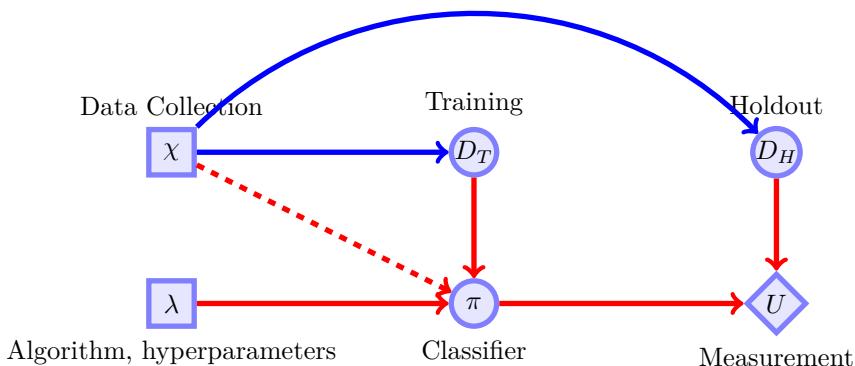


Figure 2.2: The decision process in classification.

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 is going to be biased.

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 $\pi(a | x)$ to each possible label a , and our utility is simply the identity function $U(a, y) \triangleq \mathbb{I}\{a = y\}$.

Classification accuracy

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

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:

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

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.

2.2.1 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 hyperparameters by executing an algorithm herself. This involves trying different algorithms and hyperparameters on the same training data D_T and then measuring their performance in the holdout set D_H .

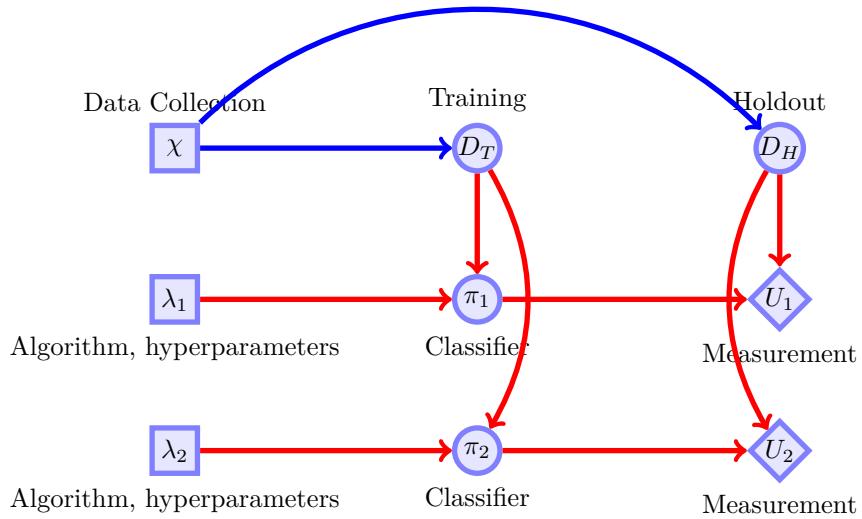


Figure 2.3: Selecting algorithms and hyperparameters through holdouts

Holdout sets

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

- Original data D , e.g. $D = (x_1, \dots, x_T)$.
- Training data $D_T \subset D$, e.g. $D_T = x_1, \dots, x_n$, $n < T$.
- Holdout data $D_H = D \setminus D_T$, used to measure the quality of the result.
- Algorithm λ with hyperparameters 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. $D = (x_1, \dots, x_T)$. We then split this into a training data set $D_T \subset D$, e.g. $D_T = x_1, \dots, x_n$, $n < T$ 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 2.4 illustrates this.

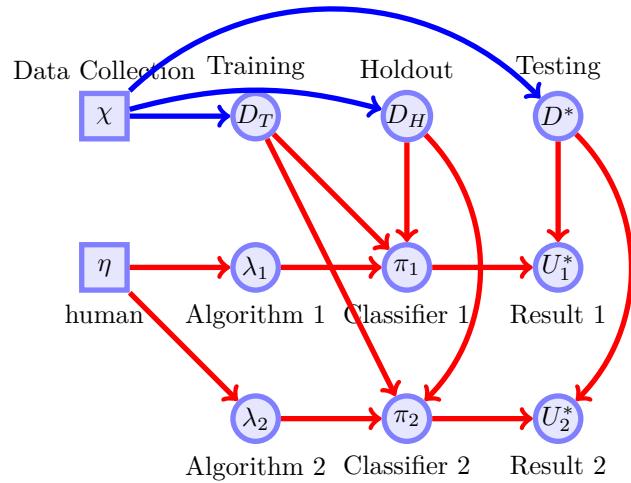


Figure 2.4: Simplified dependency graph for selecting hyperparameters for different algorithms, and comparing them on an independent test set. For the i -th algorithm, the classifier model is

2.2.2 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 might not be easy to collect data in the first place.

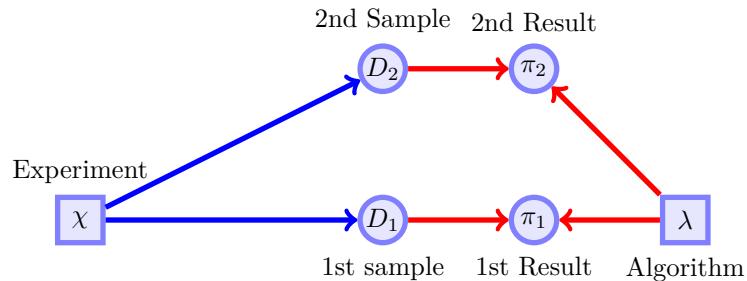


Figure 2.5: Multiple samples

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.

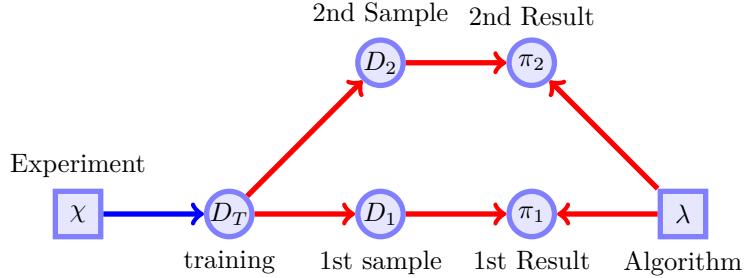


Figure 2.6: Bootstrap replicates of a single sample

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** $i = 1, \dots, k$
3. $D^{(i)} = \text{Bootstrap}(D)$
4. **return** $\{D^{(i)} \mid i = 1, \dots, k\}$.

where $\text{Bootstrap}(D)$ 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*⁵.

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 λ , 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** $\{y_1, \dots, y_k\}$.

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 1. 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 $a_t = x_t$.
 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.

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

2.2.3 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 analysis³, 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. Create a simulation that allows you to collect data similar to the real one.
2. Collect data from the simulation and analyse it according to your protocol.
3. If the results are not as expected, alter the protocol or the simulation. In which cases do you get good results?
4. Finally, use the best-performing method as 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?

EXERCISE 1. Work in teams of 2-3 students.

Select an arbitrary classification dataset from <https://archive.ics.uci.edu/ml/datasets.html?task=cla>.

Select any arbitrary machine learning algorithm for classification from `scikitlearn` that can be used with this dataset, and identify its main hyperparameters.

Varying at least one hyperparameter, use bootstrapping and/or cross-validation to find the optimal value for that hyperparameter, and report its performance. How close to the reported accuracy do you expect its performance to be in reality? What are the factors that might cause it to deviate?

Write a short report summarising both your methodology and your results. Exchange this report with another group of students. See whether you can reproduce exactly what they have done.

2.3 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

For any probability measure² P on (Ω, Σ) ,

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 2.3.1 (Conditional probability). The probability of A happening if we know that B has happened is defined to be:

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

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 the events divided by the second event. We can rewrite this definition as follows, by using the definition for $P(B | A)$

Bayes's theorem

For $P(A_1 \cup A_2) = 1$, $A_1 \cap A_2 = \emptyset$,

$$P(A_i | B) = \frac{P(B | A_i)P(A_i)}{P(B)} = \frac{P(B | A_i)P(A_i)}{P(B | A_1)P(A_1) + P(B | A_2)P(A_2)}$$

EXAMPLE 2 (probability of rain). What is the probability of rain given a forecast x_1 or x_2 ?

$$\begin{array}{l|l} \omega_1: \text{rain} & P(\omega_1) = 80\% \\ \omega_2: \text{dry} & P(\omega_2) = 20\% \end{array}$$

Table 2.1: Prior probability of rain tomorrow

$$\begin{array}{l|l} x_1: \text{rain} & P(x_1 | \omega_1) = 90\% \\ x_2: \text{dry} & P(x_2 | \omega_2) = 50\% \end{array}$$

Table 2.2: Probability the forecast is correct

$$\begin{aligned} P(\omega_1 | x_1) &= 87.8\% \\ P(\omega_1 | x_2) &= 44.4\% \end{aligned}$$

Table 2.3: 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, $P_\mu(y_t = c | x_t)$. If we somehow obtained the distribution of data $P_\mu(x_t | y_t)$ 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:

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

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

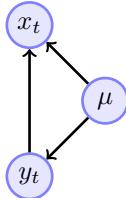


Figure 2.7: A generative classification model. μ identifies the model (parameter). x_t are the features and y_t the class label of the t -th example.

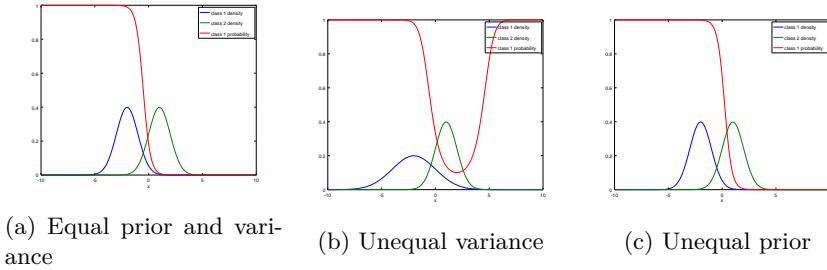


Figure 2.8: The effect of changing variance and prior on the classification decision when we assume a normal distribution.

EXAMPLE 3 (Normal distribution). A simple example is when x_t is normally distributed in a manner that depends on the class. Figure 2.8 shows the distribution of x_t for two different classes, with means of -1 and $+1$ respectively, for three different cases. In the first case, both classes have variance of 1 , and we assume the same prior probability for both

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

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

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, ξ .

Subjective probability measure ξ

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

Bayesian inference illustration

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

- *Prior* belief $\xi(\mu)$ 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:

$$\xi(\mu | h) \propto P_\mu(h)\xi(\mu) \quad (\text{conclusion} = \text{evidence} \times \text{prior})$$



2.3.1 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 4. 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 5. 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 ξ on a set of models \mathcal{M} specifies our subjective belief $\xi(\mu)$ that each model is true.³

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

$$\xi(\mu_1 | \mathbf{x}) = \frac{\mathbb{P}(\mathbf{x} | \mu_1)\xi(\mu_1)}{\mathbb{P}_\xi(\mathbf{x})},$$

where

$$\mathbb{P}_\xi(\mathbf{x}) \triangleq \mathbb{P}(\mathbf{x} | \mu_1)\xi(\mu_1) + \mathbb{P}(\mathbf{x} | \mu_0)\xi(\mu_0).$$

The only thing left to specify is $\xi(\mu_1)$, 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 $\mathcal{M} = \{\mu_1, \dots, \mu_k\}$.
- Each model μ is represented by a specific probabilistic model for any possible data x , that is $P_\mu(x) \equiv \mathbb{P}(x | \mu)$.
- For each model, we have a prior probability $\xi(\mu)$ that it is correct.
- After observing the data, we can calculate a posterior probability that the model is correct:

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

Interpretation

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

It must be emphasized that $P_\mu(x) = \mathbb{P}(x | \mu)$ 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 2 (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.

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

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:

$$\begin{array}{lll} P_{\mu_1}(x_t = 1) = 1, & P_{\mu_1}(x_t = 0) = 0. & \text{(true medium model)} \\ P_{\mu_0}(x_t = 1) = 1/2, & P_{\mu_0}(x_t = 0) = 1/2. & \text{(non-medium model)} \end{array}$$

The only thing left to specify is $\xi(\mu_1)$, 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.

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

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.

$$\begin{aligned} \xi(\mu_1 | x) &= \frac{P_{\mu_1}(x)\xi(\mu_1)}{\mathbb{P}_\xi(x)} && \text{(posterior belief)} \\ \mathbb{P}_\xi(x) &\triangleq P_{\mu_1}(x)\xi(\mu_1) + P_{\mu_0}(x)\xi(\mu_0). && \text{(marginal distribution)} \end{aligned}$$

Throw a coin 4 times, and have a classmate make a prediction. What your belief that your classmate is a medium? Is the prior you used reasonable?

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 $\xi(\mu) = 1/3$ 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 2.4: Predictions by three different entities for the probability of rain on a particular day, along with whether or not it actually rained.

EXERCISE 3. • n meteorological stations $\{\mu_i \mid i = 1, \dots, n\}$

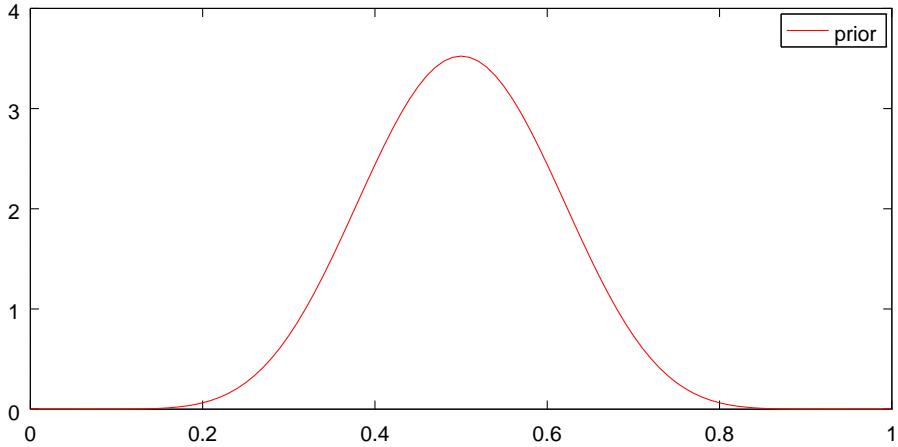
- The i -th station predicts rain $P_{\mu_i}(x_t \mid x_1, \dots, x_{t-1})$.
- Let $\xi_t(\mu)$ be our belief at time t . Derive the next-step belief $\xi_{t+1}(\mu) \triangleq \xi_t(\mu|y_t)$ in terms of the current belief ξ_t .
- Write a python function that computes this posterior

$$\xi_{t+1}(\mu) \triangleq \xi_t(\mu|x_t) = \frac{P_\mu(x_t \mid x_1, \dots, x_{t-1})\xi_t(\mu)}{\sum_{\mu'} P_{\mu'}(x_t \mid x_1, \dots, x_{t-1})\xi_t(\mu')}$$

Bayesian inference for Bernoulli distributions

Estimating a coin's bias

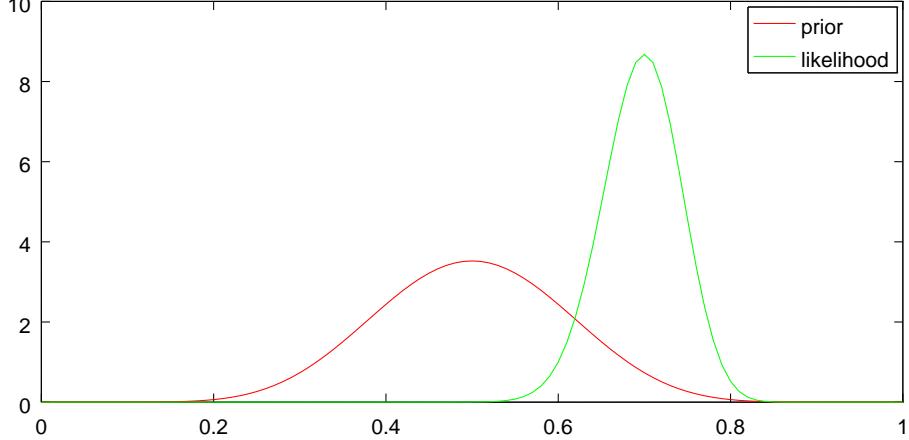
A fair coin comes heads 50% of the time. We want to test an unknown coin, which we think may not be completely fair.

Figure 2.9: Prior belief ξ about the coin bias θ .

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

Figure 2.10: Prior belief ξ about the coin bias θ and likelihood of θ for the data.

different models. From these, we calculate a *posterior* distribution over the correct models. This represents our conclusion given our prior and the data. If the prior distribution is described by the so-called Beta density

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

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

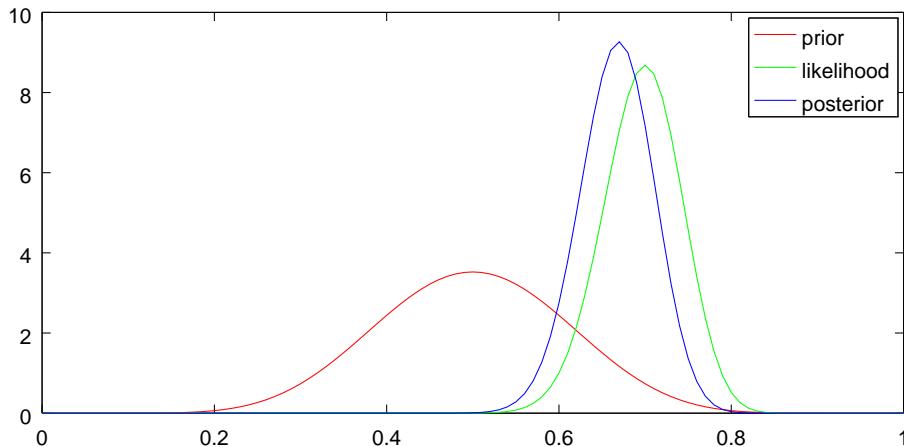


Figure 2.11: Prior belief $\xi(\theta)$ about the coin bias θ , likelihood of θ for the data, and posterior belief $\xi(\theta | x)$

Learning outcomes

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?



2.4 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.

2.4.1 Simple decision problems

Preferences

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

EXAMPLE 6. Food

- A McDonald's cheeseburger
- B Surstromming
- C Oatmeal

Money

- A 10,000,000 SEK
- B 10,000,000 USD
- C 10,000,000 BTC

Entertainment

- A Ticket to Liseberg
- B Ticket to Rebstar
- C Ticket to Nutcracker

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* $U : \mathcal{R} \rightarrow \mathbb{R}$, 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. Even when this assumption is correct, individuals do not have a common utility function.

EXERCISE 4. 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 7. 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.

$\rho(\omega, a)$	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 2.5: Rewards and utilities.

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* $U : \mathcal{R} \rightarrow \mathbb{R}$, assigning values to reward.
- We (weakly) prefer A to B iff $U(A) \geq U(B)$.

EXAMPLE 8. 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
- $\max_a \min_\omega U = 0$
- $\min_\omega \max_a U = 0$

Expected utility

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

EXAMPLE 9. 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!

$\rho(\omega, a)$	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
$\mathbb{E}_P(U a)$	0	-1.2

Table 2.6: Rewards, utilities, expected utility for 20% probability of rain.

2.4.2 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:

$$U(\xi, a) \triangleq \sum_{\mu} U(\mu, a) \xi(\mu)$$

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

Definition 2.4.1 (Bayes utility). The maximising decision for ξ has an expected utility equal to:

$$U^*(\xi) \triangleq \max_{a \in \mathcal{A}} U(\xi, a). \quad (2.4.1)$$

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 5. 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 $\xi(\mu) = 1/3$ for each model.

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?

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

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?

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

2.4.3 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 2.8: 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 10 (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 $\mathbf{x} = x_1, \dots, x_n$,

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

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:

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

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

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

$$\mathbb{P}(\mathbf{x} = 1, \dots, 1 | \mu_1) = 1, \quad \mathbb{P}(\mathbf{x} = 1, \dots, 1 | \mu_0) = 2^{-n}. \quad (2.4.2)$$

Now we can calculate the posterior distribution, which is

$$\xi(\mu_1 | \mathbf{x} = 1, \dots, 1) = \frac{1 \times \xi(\mu_1)}{1 \times \xi(\text{model}_1) + 2^{-n}(1 - \xi(\mu_1))}.$$

Our expected utility for taking action a_0 is actually

$$\mathbb{E}_{\xi}(U | a_0) = 1 \times \xi(\mu_0 | \mathbf{x}) + 0 \times \xi(\mu_1 | \mathbf{x}), \quad \mathbb{E}_{\xi}(U | a_1) = 0 \times \xi(\mu_0 | \mathbf{x}) + 1 \times \xi(\mu_1 | \mathbf{x})$$

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 11. Construction of the test for the medium

- μ_0 is simply the $Bernoulli(1/2)$ model: responses are by chance.
- We need to design a policy $\pi(a | \mathbf{x})$ 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 constructing a threshold test from the inverse-CDF.

Using *p*-values to construct statistical tests

Definition 2.4.2 (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 $f : \mathcal{X} \rightarrow [0, 1]$ is designed to have the property:

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

If our decision rule is:

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

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.

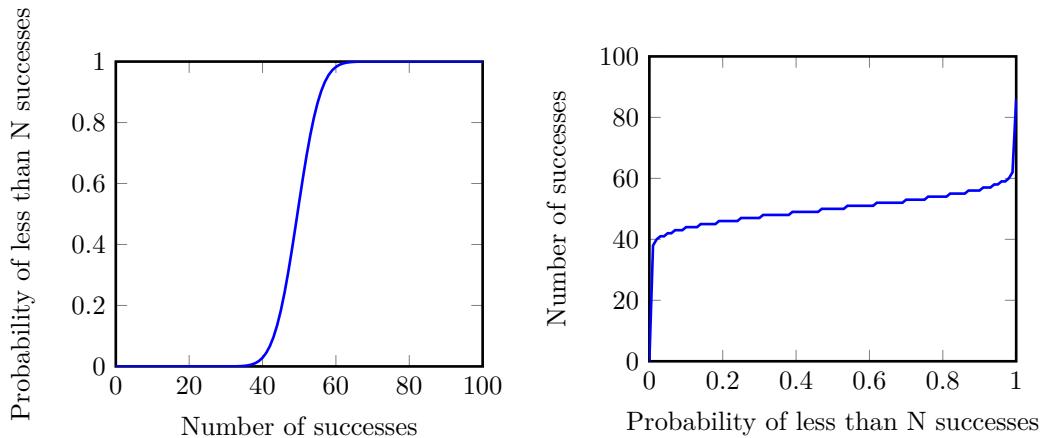
Issues with p -values

- 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. $\delta \neq \mathbb{P}(\neg\mu_0 | x)$.
- The null-rejection error probability is the same irrespective of the amount of data (by design).

p -values for the medium example

Let us consider the example of the medium.

- μ_0 is simply the $Bernoulli(1/2)$ model: responses are by chance.
- CDF: $P_{\mu_0}(N \leq n | K = 100)$ 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 .



Building a test

The test statistic

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

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

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 6. • 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?

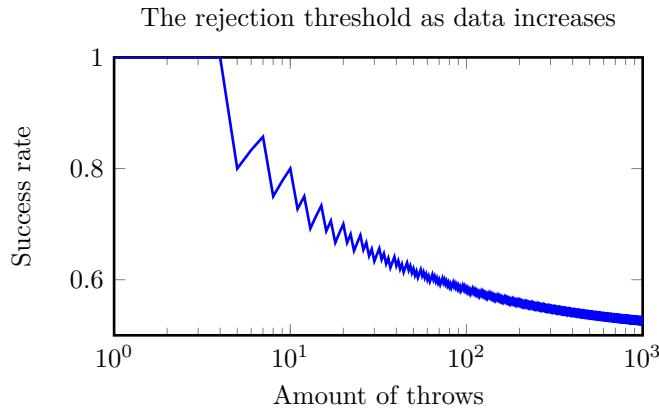


Figure 2.12: Here we see how the rejection threshold, in terms of the success rate, changes with the number of throws to achieve an error rate of $\delta = 0.05$.

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.

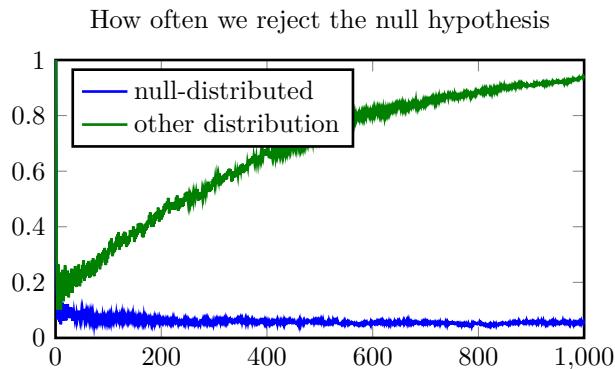


Figure 2.13: Here we see the rejection rate of the null hypothesis (μ_0) for two cases. Firstly, for the case when μ_0 is true. Secondly, when the data is generated from $\text{Bernoulli}(0.55)$.

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 $\text{Bernoulli}(0.55)$ 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 $\text{Bernoulli}(0.55)$ 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 version of the test

EXAMPLE 12. 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 $\xi(\mu_i) = 1/2$. Here we place an equal probability in both models.
3. μ_0 : *Bernoulli*(1/2). This is the same as the null hypothesis test.
4. μ_1 : *Bernoulli*(θ), $\theta \sim \text{Unif}([0, 1])$. 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 $\xi(\mu | x)$.
6. Choose a_i , where $i = \arg \max_j \xi(\mu_j | x)$.

Bayesian model averaging for the alternative model μ_1

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

$$P_{\mu_1}(x) = \int_{\Theta} B_{\theta}(x) d\beta(\theta) \quad (2.4.3)$$

$$\xi(\mu_0 | x) = \frac{P_{\mu_0}(x)\xi(\mu_0)}{P_{\mu_0}(x)\xi(\mu_0) + P_{\mu_1}(x)\xi(\mu_1)} \quad (2.4.4)$$

Posterior probability of null hypothesis

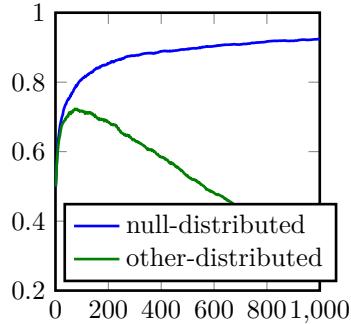


Figure 2.14: Here we see the convergence of the posterior probability.

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.

Rejection of null hypothesis for Bernoulli(0.5)

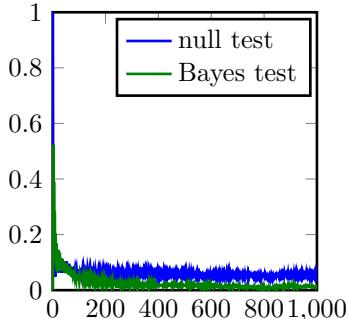


Figure 2.15: Comparison of the rejection probability for the null and the Bayesian test when μ_0 is true.

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.

Rejection of null hypothesis for Bernoulli(0.55)

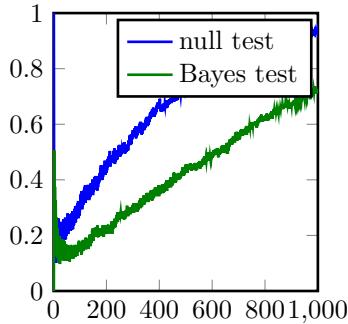


Figure 2.16: Comparison of the rejection probability for the null and the Bayesian test when μ_1 is true.

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

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>

2.5 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 $P_\mu(y_t | x_t)$, we want

to define a decision rule $\pi(a_t | x_t)$ 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 $\pi(a_t | x_t)$ 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

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

EXERCISE 7. If we have a model $P_\mu(y_t | x_t)$, and a suitable U , what is the optimal decision to make?

Deciding the class given a model family

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

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

Posterior over classification models

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

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 $P_\mu(x_1, \dots, x_T) = P_{\mu'}(x_1, \dots, x_T)$. 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 :

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

The Bayes rule for maximising $\mathbb{E}_\xi(U | a, x_t, D_T)$

The decision rule simply chooses the action:

$$a_t \in \arg \max_{a \in \mathcal{A}} \sum_y \sum_{\mu \in \mathcal{M}} P_\mu(y_t = y | x_t) \xi(\mu | D_T) U(a, y) \quad (2.5.1)$$

$$= \arg \max_{a \in \mathcal{A}} \sum_y \mathbb{P}_{\xi|D_T}(y_t | x_t) U(a, y) \quad (2.5.2)$$

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

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

Approximating the model**Full Bayesian approach for infinite \mathcal{M}**

Here ξ can be a probability density function and

$$\xi(\mu | D_T) = P_\mu(D_T) \xi(\mu) / \mathbb{P}_{\xi}(D_T), \quad \mathbb{P}_{\xi}(D_T) = \int_{\mathcal{M}} P_\mu(D_T) \xi(\mu) d\mu,$$

can be hard to calculate.

Maximum a posteriori model

We only choose a single model through the following optimisation:

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

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?

2.6 Classification with stochastic gradient descent

Classification as an optimisation problem.

Finding the optimal policy for our belief ξ is not normally very difficult. However, it requires that we maintain the complete distribution ξ and that we also under some probability distribution P . 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.

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

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

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

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

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

$$\max_{\theta \in \Theta} f(\pi_{\theta}, \mu, U), \quad f(\pi_{\theta}, \mu, U) \triangleq \mathbb{E}_{\mu}^{\pi_{\theta}}(U) \quad (2.6.1)$$

$$f(\pi_{\theta}, \mu, U) = \sum_{x, y, a} U(a, y) \pi_{\theta}(a | x) P_{\mu}(y | x) P_{\mu}(x) \quad (2.6.2)$$

$$\approx \sum_{t=1}^T \sum_{a_t} U(a_t, y_t) \pi_{\theta}(a_t | x_t), \quad (x_t, y_t)_{t=1}^T \sim P_{\mu}. \quad (2.6.3)$$

In practice, this is the empirical expectation on the training set $\{(x_t, y_t) \mid t = 1, \dots, T\}$. 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.

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

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 methodos

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

$$g(\theta) = \int_{\mathcal{M}} f(\theta, \mu) d\xi(\mu)$$

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta f(\theta_i, \mu_i), \quad \mu_i \sim \xi.$$

Stochastic gradient methods are commonly employed in neural networks.

2.6.1 Neural network models

Two views of neural networks

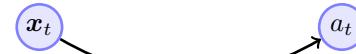
In the simplest sense a neural network is simply as parametrised functions f_θ . In classification, neural networks can be used as probabilistic models, so they describes the probability $P_\theta(y|x)$, or as classification policies so that $f_\theta(x, a)$ describes the probability $\pi_\theta(a |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

Neural network classification model $P_\theta(y | x_t)$



Objective: Find the best model for D_T .

Neural network classification policy $\pi(a_t | x_t)$



Objective: Find the best policy for $U(a, x)$.

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

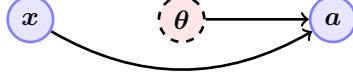


Figure 2.17: Abstract graphical model for a neural network

A neural network as used for modelling classification or regression problems, is simply a parametrised mapping $\mathcal{X} \rightarrow \mathcal{Y}$. If we include the network parameters, then it is instead a mapping $\mathcal{X} \times \Theta \rightarrow \mathcal{Y}$, as seen in Figure 2.19.

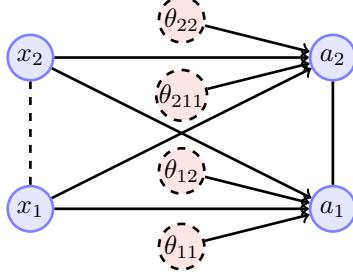


Figure 2.18: Graphical model for a linear neural network.

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.

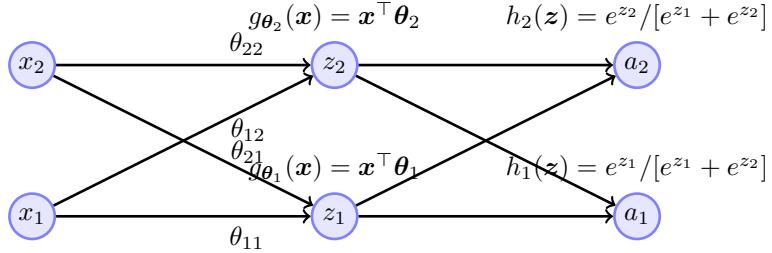


Figure 2.19: Architectural view of a linear neural network.

Definition 2.6.1 (Linear classifier). A linear classifier with N inputs and C outputs is parametrised by

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

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

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 \mathbf{x}_t are features $\mathbf{x}_t = (x_{t,1} \dots, x_{t,n})$ so that $\mathcal{X} \subset \mathbb{R}^N$. It is convenient to consider the network output as a vector on the simplex $\mathbf{y} \in \Delta^A$, 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 \mathbf{x} itself.

Gradient ascent for a matrix U

$$\begin{aligned} & \max_{\theta} \sum_{t=1}^T \sum_{a_t} U(a_t, y_t) \pi_{\theta}(a_t \mid x_t) && \text{(objective)} \\ & \nabla_{\theta} \sum_{t=1}^T \sum_{a_t} U(a_t, y_t) \pi_{\theta}(a_t \mid x_t) && \text{(gradient)} \\ & = \sum_{t=1}^T \sum_{a_t} U(a_t, y_t) \nabla_{\theta} \pi_{\theta}(a_t \mid x_t) && \text{(2.6.4)} \end{aligned}$$

We now need to calculate the gradient of the policy.

Chain Rule of Differentiation

$$\begin{aligned} f(z), z = g(x), \quad & \frac{df}{dx} = \frac{df}{dg} \frac{dg}{dx} && \text{(scalar version)} \\ \nabla_{\theta} \pi = \nabla_g \pi \nabla_{\theta} g & && \text{(vector version)} \end{aligned}$$

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 networks.
- 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?

2.7 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 \mathbf{x} , we would like to calculate the probability of different classes C given the data, $\mathbb{P}(C | \mathbf{x})$. So here, the class is the hypothesis.

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

$$\mathbb{P}(C | \mathbf{x}) = \frac{\mathbb{P}(\mathbf{x} | C) \mathbb{P}(C)}{\sum_i \mathbb{P}(\mathbf{x} | C_i) \mathbb{P}(C_i)}$$

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

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

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.

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

where $N_k(i, C)$ is the number 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 2.7.1. 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 2.7.2. In fact, the Laplace smoothing model corresponds to a so-called Dirichlet 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

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

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 $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ 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 $\xi(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Chapter 3

Privacy

Participating in a study always carries a risk for individuals, namely that of data disclosure. In this chapter, we first explain how simple database query methods, and show even a small number of queries to a database they can compromise the privacy of individuals. We then introduce to formal concepts of privacy protection: k -anonymity and differential privacy. The first is relatively simple to apply and provides some limited resistance to identification of individuals through record linkage attacks. The latter is a more general concept, and can be simple apply in some settings, while it offers 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.

3.1 Database access models

Databases

ID	Name	Salary	Deposits	Age	Postcode	Profession
1959060783	Li Pu	150,000	1e6	60	1001	Politician
1946061408	Sara Lee	300,000	-1e9	72	1001	Rentier
2100010101	A. B. Student	10,000	100,000	40	1001	Time Traveller

EXAMPLE 13 (Typical relational database in a tax office).

Database access

- When owning the database: Direct look-up.
- When accessing a server etc: Query model.

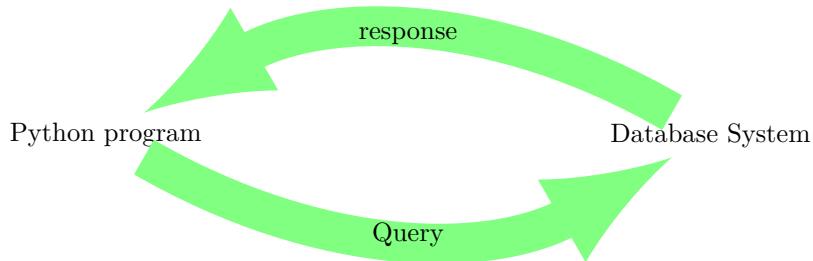


Figure 3.1: Database access model

SQL: A language for database access

Creating and filling tables

- `CREATE TABLE table-name (column1, column2)` Create a new table
- `INSERT INTO table-name VALUES ('value1', 'value2')` Add specific values into a table
- `INSERT INTO table-name VALUES (?, ?, variable)` Fill in values from a variable

EXAMPLE 14. Database creation `src/privacy/database-creation.py` `src/privacy/database-access.py`

Queries in SQL

The SELECT statement

- `SELECT column1, column2 FROM table;` This selects only some columns from the table
- `SELECT * FROM table;` This selects all the columns from the table

Selecting rows

```
SELECT * FROM table WHERE column = value;
```

Arithmetic queries

Here are some example SQL statements

- `SELECT COUNT(column) FROM table WHERE condition;` This allows you to count the number of rows matching condition
- `SELECT AVG(column) FROM table WHERE condition;` This lets you to count the number of rows matching condition
- `SELECT SUM(column) FROM table WHERE condition;` This is used to sum up the values in a column.

3.2 Privacy in databases

Anonymisation

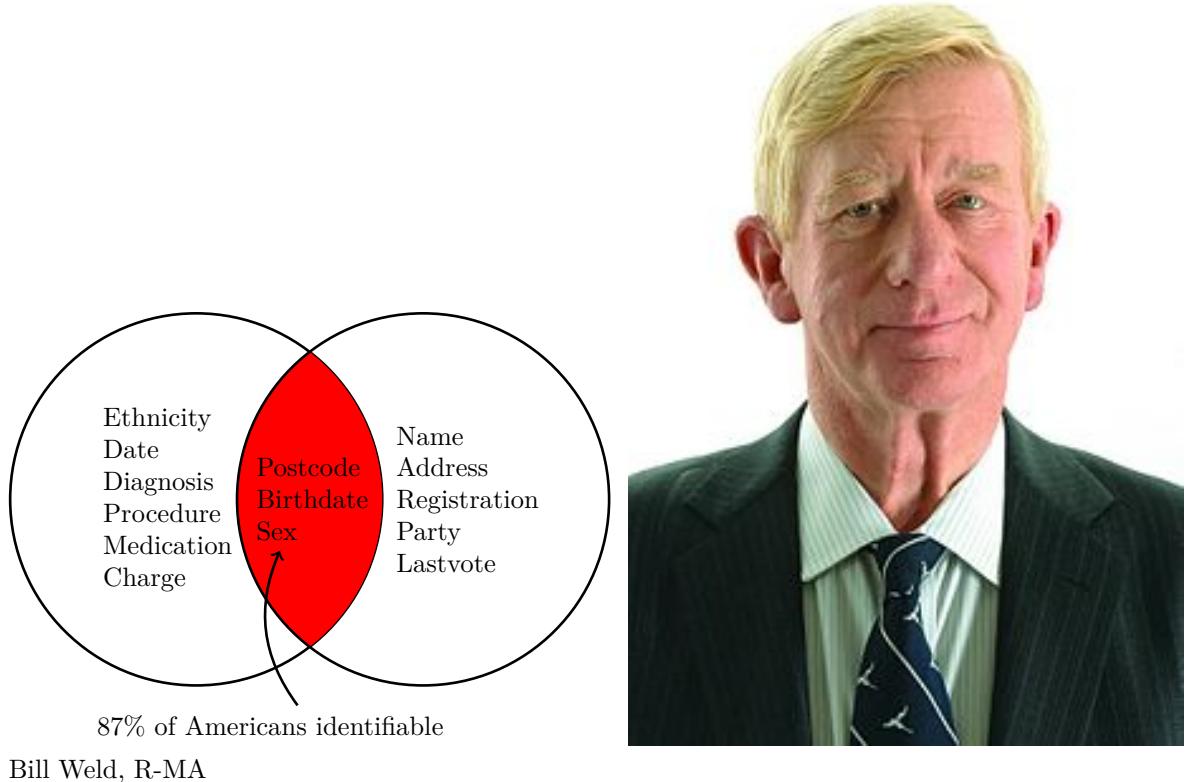
If we wish to publish a database, frequently we need to protect identities of people involved. The simplest method for doing that is simply erasing directly identifying information. However, this does not really work most of the time, especially since attackers can have side-information that can reveal the identities of individuals in the original data.

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 15 (Typical relational database in Tinder).

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.

Record linkage



Bill Weld, R-MA

ID	Name	Salary	Deposits	Age	Postcode	Profession
1959060783	Li Pu	150,000	1e6	60	1001	Politician
1946061408	Sara Lee	300,000	-1e9	72	1001	Rentier
2100010101	A. B. Student	10,000	100,000	40	6732	Time Traveller

EXAMPLE 16 (Typical relational database in a tax office).

Birthday	Name	Height	Weight	Age	Postcode	Profession
06/07		190	80	60-70	1001	Politician
06/14		185	110	70+	1001	Rentier
01/01		170	70	40-60	6732	Time Traveller

EXAMPLE 17 (Typical relational database in a tax office).

3.3 k-anonymity

k-anonymity



(a) Samarati



(b) Sweeney

The concept of k -anonymity was introduced by Samarati and Sweeney²⁰ and provides good guarantees when accessing a single database

Definition 3.3.1 (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*.

It's the analyst's job to define quasi-identifiers

Birthday	Name	Height	Weight	Age	Postcode	Profession
06/07	Li Pu	190	80	60+	1001	Politician
06/14	Sara Lee	185	110	60+	1001	Rentier
06/12	Nikos Papadopoulos	180	82	60+	1243	Politician
01/01	A. B. Student	170	70	40-60	6732	Time Traveller
05/08	Li Yang	175	72	30-40	6910	Time Traveller

Table 3.1: 1-anonymity.

Birthday	Name	Height	Weight	Age	Postcode	Profession
		180-190	80+	60+	1*	
		180-190	80+	60+	1*	
		180-190	80+	60+	1*	
		170-180	60-80	20-60	6*	
		170-180	60-80	20-60	6*	

Table 3.2: 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

3.4 Differential privacy

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 power. 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. 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 similar database based on the result of the computation. While the notion of similarity is for the analyst to define, it is common to say that two databases are similar when they are identical apart from the data of one person.

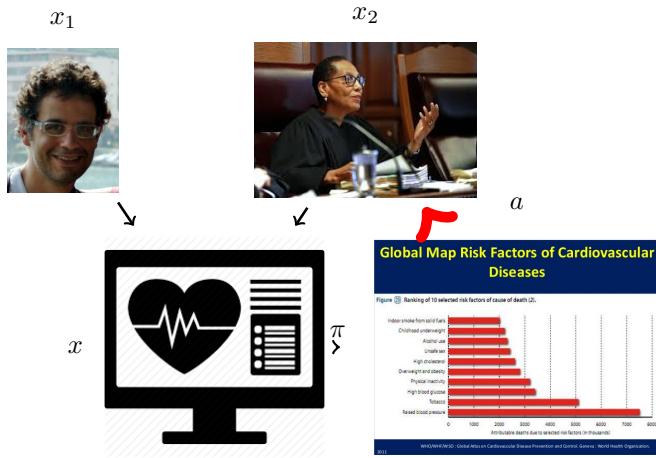


Figure 3.3: If two people contribute their data $x = (x_1, x_2)$ to a medical database, and an algorithm π computes some public output a from x , then it should be hard to infer anything about the data from the public output.

Privacy desiderata

Consider a scenario where n persons give their data x_1, \dots, x_n to an analyst. This analyst then performs some calculation $f(x)$ on the data and published the result. 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.

Example: The prevalence of drug use 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.

EXERCISE 8. 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?

Solution. Since the responses are random, we will deal with expectations first

$$\begin{aligned}\mathbb{E} p &= \frac{1}{2} \times \frac{1}{2} + q \times \frac{1}{2} = \frac{1}{4} + \frac{q}{2} \\ q &= 2\mathbb{E} p - \frac{1}{2}.\end{aligned}$$

□

The problem with this approach, of course, is that we are effectively throwing away half of our data. 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.

The randomised response mechanism

The above idea can be generalised. Consider we have data x_1, \dots, x_n from n users and we transform it randomly to y_1, \dots, y_n using the following mapping.

Definition 3.4.1 (Randomised response). The i -th user, whose data is $x_i \in \{0, 1\}$, responds with $a_i \in \{0, 1\}$ with probability

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

where $j \neq k$.

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

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

This mechanism satisfies so-called ϵ -differential privacy, which we will define later.

EXERCISE 9. Let the adversary have a prior $\xi(x = 0) = 1 - \xi(x = 1)$ over the values of the true response of an individual. we use the randomised response mechanism with p and the adversary observes the randomised data $a = 1$ for that individual, then what is $\xi(x = 1 | a = 1)$?

The local privacy model

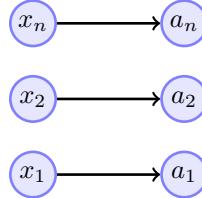


Figure 3.4: The local privacy model

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 no individual will provide their true data with certainty. This model allows us to publish a complete dataset of private responses.

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.

Definition 3.4.2 (ϵ -Differential Privacy). A stochastic algorithm $\pi : \mathcal{X} \rightarrow \mathcal{A}$, where \mathcal{X} is endowed with a neighbourhood relation N , is said to be ϵ -differentially private if

$$\left| \ln \frac{\pi(a | x)}{\pi(a | x')} \right| \leq \epsilon, \quad \forall x N x'. \quad (3.4.1)$$

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

In this context, two datasets are usually called neighbouring if $x = (x_1, \dots, x_{i-1}, x_i, x_{i+1} \dots, x_n)$ and $x' = (x_1, \dots, x_{i-1}, x_{i+1} \dots, x_n)$, i.e. if one dataset is missing an element.

A slightly weaker definition of neighbourhood is to say that $x N x'$ if $x' = (x_1, \dots, x_{i-1}, x'_i, x_{i+1} \dots, x_n)$, i.e. if one dataset has an altered element. We will usually employ this latter definition, especially for the local privacy model.

Defining neighbourhoods

Birthday	Name	Height	Weight
06/07	Li Pu	190	80
06/14	Sara Lee	185	110
06/12	<i>Nikos Papadopoulos</i>	170	82
01/01	A. B. Student	170	70
05/08	Li Yang	175	72

Table 3.3: Data 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 3.4: 1-Neighbour x'

Birthday	Name	Height	Weight
06/07	Li Pu	190	80
06/14	Sara Lee	185	110
06/13	Nikos Papadopoulos	180	80
01/01	A. B. Student	170	70
05/08	Li Yang	175	72

Table 3.5: 2-Neighbour x'

The definition of differential privacy

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

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 confidentiality of responders' information while maintaining data that are suitable for their intended uses.

Open problems

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

Remark 3.4.1. 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.

Remark 3.4.2. The randomised response mechanism with $p \leq 1/2$ is $(\ln \frac{1-p}{p})$ -DP.

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

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

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

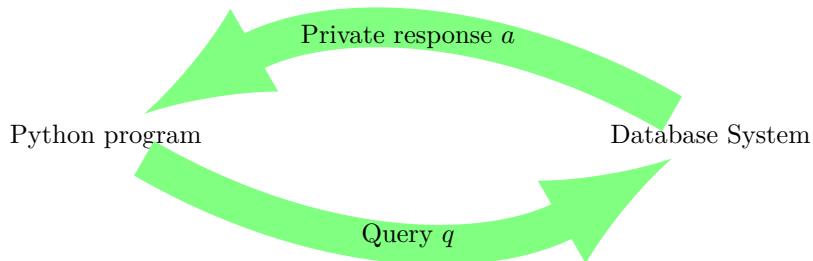


Figure 3.5: Private database access model

Response policy

The policy defines a distribution over responses a given the data x and the query q .

$$\pi(a | x, q)$$

Differentially private queries

There is no actual DP-SELECT statement, but we can imagine it.

The DP-SELECT statement

- DP-SELECT ϵ column1, column2 FROM table; This selects only some columns from the table
- DP-SELECT ϵ * FROM table; This selects all the columns from the table

Selecting rows

DP-SELECT ϵ * FROM table WHERE column = value;

Arithmetic queries

Here are some example SQL statements

- DP-SELECT ϵ COUNT(column) FROM table WHERE condition; This allows you to count the number of rows matching condition
- DP-SELECT ϵ AVG(column) FROM table WHERE condition; This lets you to count the number of rows matching condition
- DP-SELECT ϵ SUM(column) FROM table WHERE condition; This is used to sum up the values in a column.

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 bound the total privacy loss.

Definition 3.4.3 (T -fold adaptive composition). In this privacy model, an adversary is allowed to compose T queries. The composition is *adaptive*, in the sense that the next query is allowed to depend on the previous queries and their results.

Theorem 3.4.1. *For any $\epsilon > 0$, the class of ϵ -differentially private mechanism satisfy $T\epsilon$ -differential privacy under T -fold adaptive composition.*

EXERCISE 10. Adversary knowledge Assume that the adversary knows that the data is either \mathbf{x} or \mathbf{x}' . For concreteness, assume the data is either

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

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

$$\mathbf{x}' = (x_1, \dots, x_j = 1, \dots, x_n).$$

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

$$\xi(\mathbf{x}) = 1 - \xi(\mathbf{x}')$$

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 $\xi(\mathbf{x} | a, \pi)$ after having seen the output, if π is ϵ -DP?

Solution. We can write the adversary posterior as follows.

$$\xi(\mathbf{x} | a, \pi) = \frac{\pi(a | \mathbf{x})\xi(\mathbf{x})}{\pi(a | \mathbf{x})\xi(\mathbf{x}) + \pi(a | \mathbf{x}')\xi(\mathbf{x}')} \quad (3.4.2)$$

$$\geq \frac{\pi(a | \mathbf{x})\xi(\mathbf{x})}{\pi(a | \mathbf{x})\xi(\mathbf{x}) + \pi(a | \mathbf{x})e^\epsilon\xi(\mathbf{x}')} \quad (\text{from DP definition})$$

$$= \frac{\xi(\mathbf{x})}{\xi(\mathbf{x}) + e^\epsilon\xi(\mathbf{x}')} \quad (3.4.3)$$

But this is not very informative. We can also write

$$\frac{\xi(\mathbf{x} | a, \pi)}{\xi(\mathbf{x}' | a, \pi)} = \frac{\pi(a | \mathbf{x})\xi(\mathbf{x})}{\pi(a | \mathbf{x}')\xi(\mathbf{x}')} \geq \frac{\pi(a | \mathbf{x})\xi(\mathbf{x})}{\pi(a | \mathbf{x})e^{-\epsilon}\xi(\mathbf{x}')} = \frac{\xi(\mathbf{x})}{\xi(\mathbf{x}')}e^\epsilon \quad (3.4.4)$$

□

Dealing with 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 \mathbf{x}_t in a database. Then the mechanism can release each $a_{t,i}$ independently.

Independent release of multiple attributes.

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.

3.4.1 Other differentially private mechanisms

The Laplace mechanism.

A simple method to obtain a differentially private algorithm from a deterministic function $f : \mathcal{X} \rightarrow \mathbb{R}$, is to use additive noise, so that the output of the algorithm is simply

$$a = f(x) + \omega, \quad \omega \sim \text{Laplace}.$$

The amount of noise added, together with the smoothness of the function f , determine the amount of privacy we have.

Definition 3.4.4 (The Laplace mechanism). For any function $f : \mathcal{X} \rightarrow \mathbb{R}$,

$$\pi(a | x) = \text{Laplace}(f(x), \lambda), \quad (3.4.5)$$

where the Laplace density is defined as

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

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

Here, $\text{Laplace}(\mu, \lambda)$ is the density $f(x) = \frac{\lambda}{2} \exp(-\lambda|x - \mu|)$.

EXAMPLE 18 (Calculating the average salary). • The i -th person receives salary x_i

- We wish to calculate the average salary in a private manner.

Local privacy model

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

Centralised privacy model

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

How should we add noise in order to guarantee privacy?

The centralised privacy model

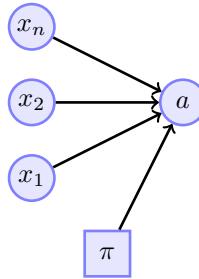


Figure 3.6: The centralised privacy model

Assumption 3.4.1. *The data x is collected and the result a is published by a trusted curator*

DP properties of the Laplace mechanism

Definition 3.4.5 (Sensitivity). The sensitivity of a function f is

$$\mathbb{L}(f) \triangleq \sup_{x \in \mathcal{X}} |f(x) - f(x')|$$

If we define a metric d , so that $d(x, x') = 1$ for $x \neq x'$, 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 19. If $f : \mathcal{X} \rightarrow [0, B]$, e.g. $\mathcal{X} = \mathbb{R}$ and $f(x) = \min\{B, \max\{0, x\}\}$, then $\mathbb{L}(f) = B$.

EXAMPLE 20. If $f : [0, B]^n \rightarrow [0, B]$ 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

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

□

Theorem 3.4.2. *The Laplace mechanism on a function f with sensitivity $\mathbb{L}(f)$, ran with $\text{Laplace}(\lambda)$ is $\mathbb{L}(f)/\lambda$ -DP.*

Proof.

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

□

So we need to use $\lambda = \mathbb{L}(f)/\epsilon$ for ϵ -DP. What is the effect of applying the Laplace mechanism in the local versus centralised model? Here let us assume $x_i \in [0, B]$ for all i and consider the problem of calculating the average.

Laplace 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 $2(M/\epsilon)^2$, so the total variance is $2M^2/\epsilon^2 n$.

Laplace in the centralised privacy model

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

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.

3.4.2 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 .
- User asks query q .
- System responds with a .
- There is a common utility function $U : \mathcal{X}, \mathcal{A}, \mathcal{Q} \rightarrow \mathbb{R}$.

We wish to maximise U with our answers, but are constrained by the fact that we also want to preserve privacy.

The utility $U(x, a, q)$ describes how appropriate each response a given by the system for a query r is given the data x . It can be seen as how useful the response is¹ 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: $U(q, a, x)$. Let $\mathbb{L}(U(q)) \triangleq \sup_{x \neq x'} |U(q, a, x) - U(q, a, x')|$ denote the sensitivity of a query. Then the following mechanism is ϵ -differentially private.

Definition 3.4.6 (The Exponential mechanism). For any utility function $U : \mathcal{Q} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$, define the policy

$$\pi(a | x) \triangleq \frac{e^{\epsilon U(q, a, x) / \mathbb{L}(U(q))}}{\sum_{a'} e^{\epsilon U(q, a', x) / \mathbb{L}(U(q))}} \quad (3.4.6)$$

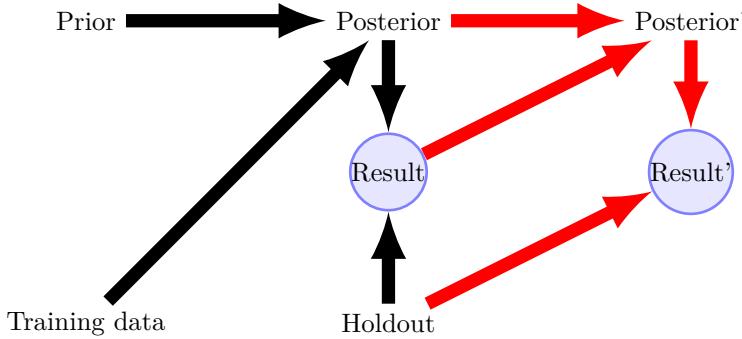
Clearly, when $\epsilon \rightarrow 0$, this mechanism is uniformly random. When $\epsilon \rightarrow \infty$ the action maximising $U(q, a, x)$ is always chosen.

Although the exponential mechanism can be used to describe most known DP mechanisms, its best use is in settings where there is a natural utility function.

3.4.3 Privacy and reproducibility

The unfortunate practice of adaptive analysis

¹This 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.



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 holdout¹¹²

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.

²Also see <https://ai.googleblog.com/2015/08/the-reusable-holdout-preserving.html>

Available privacy toolboxes***k*-anonymity**

- <https://github.com/qiyuangong/Mondrian> Mondrian *k*-anonymity

Differential privacy

- <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

Learning outcomes**Understanding**

- Linkage attacks and *k*-anonymity.
- Inferring data from summary statistics.
- The local versus global 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?

Chapter 4

Fairness

When machine learning algorithms are applied at scale, it can be difficult to imagine what their effects might be. In 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.

4.1 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

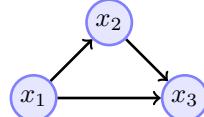


Figure 4.1: Graphical model (directed acyclic graph) for three variables.

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.

Joint probability

Let $\mathbf{x} = (x_1, \dots, x_n)$. Then $\mathbf{x} : \Omega \rightarrow X$, $X = \prod_i X_i$ and:

$$\mathbb{P}(\mathbf{x} \in A) = P(\{\omega \in \Omega \mid \mathbf{x}(\omega) \in A\}).$$

When X_i are finite, we can typically write

$$\mathbb{P}(\mathbf{x} = \mathbf{a}) = P(\{\omega \in \Omega \mid \mathbf{x}(\omega) = \mathbf{a}\}),$$

for the probability that $x_i = a_i$ for all $i \in [n]$.

Factorisation

For any subsets $B \subset [n]$ and its complement C so that $\mathbf{x}_B = (x_i)_{i \in B}$, $\mathbf{x}_C = (x_i)_{i \notin B}$

$$\mathbb{P}(\mathbf{x}) = \mathbb{P}(\mathbf{x}_B \mid \mathbf{x}_C) \mathbb{P}(\mathbf{x}_C)$$

So we can write any joint distribution as

$$\mathbb{P}(x_1) \mathbb{P}(x_2 \mid x_1) \mathbb{P}(x_3 \mid x_1, x_2) \cdots \mathbb{P}(x_n \mid x_1, \dots, x_{n-1}).$$

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

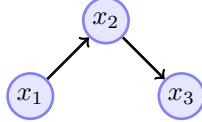


Figure 4.2: Graphical model for the factorisation $\mathbb{P}(x_3 | x_2) \mathbb{P}(x_2 | x_1) \mathbb{P}(x_1)$.

Conditional independence

We say x_i is conditionally independent of \mathbf{x}_B given \mathbf{x}_D and write $x_i | \mathbf{x}_D \perp\!\!\!\perp \mathbf{x}_B$ iff

$$\mathbb{P}(x_i, \mathbf{x}_B | \mathbf{x}_D) = \mathbb{P}(x_i | \mathbf{x}_D) \mathbb{P}(\mathbf{x}_B | \mathbf{x}_D).$$

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 random variables.

Definition 4.1.1 (Directed graphical model). A collection of n random variables $x_i : \Omega \rightarrow X_i$, and let $X \triangleq \prod_i X_i$, with underlying probability measure P on Ω . Let $\mathbf{x} = (x_i)_{i=1}^n$ and for any subset $B \subset [n]$ let

$$\mathbf{x}_B \triangleq (x_i)_{i \in B} \tag{4.1.1}$$

$$\mathbf{x}_{-j} \triangleq (x_i)_{i \neq j} \tag{4.1.2}$$

In a graphical model, conditional independence is represented through directed edges.

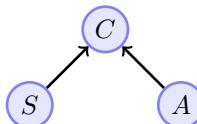


Figure 4.3: Smoking and lung cancer graphical model, where S : Smoking, C : cancer, A : asbestos exposure.

EXAMPLE 21 (Smoking and lung cancer). It has been found by Lee¹⁹ 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 other 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*.

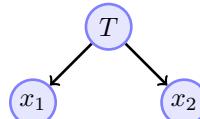


Figure 4.4: Time of arrival at work graphical model where T is a traffic jam and x_1 is the time John arrives at the office and x_2 is the time Jane arrives at the office.

EXAMPLE 22 (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 .

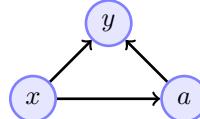


Figure 4.5: Kidney treatment model, where x : severity, y : result, a : treatment applied

	Treatment A	Treatment B	Severity	Treatment A	Treatment B
Small stones	87	270	Small stones)	93%	87%
Large stones	263	80	Large stones	73%	69%
			Average	78%	83%

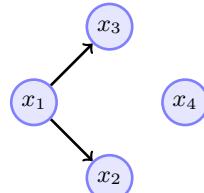
EXAMPLE 23 (Treatment effects). A curious example is that of applying one of two treatments for kidneys. In the data, it is clear that one treatment is best for both large and small stones. However, when the data is aggregated it appears as though treatment B is best. This is because treatment A is chosen 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 4.6: School admission graphical model, where z : gender, s : school applied to, a : whether you were admitted.

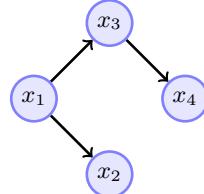
School	Male	Female
A	62%	82%
B	63%	68%
C	37%	34%
D	33%	35%
E	28%	24%
F	6%	7%
<i>Average</i>	45%	38%

EXAMPLE 24 (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 11. Factorise the following graphical model.

$$\mathbb{P}(\mathbf{x}) = \mathbb{P}(x_1) \mathbb{P}(x_2 | x_1) \mathbb{P}(x_3 | x_1) \mathbb{P}(x_4)$$

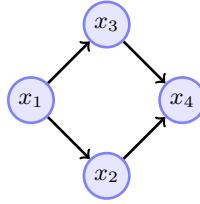


EXERCISE 12. Factorise the following graphical model.

$$Xb \mathbb{P}(\mathbf{x}) = \mathbb{P}(x_1) \mathbb{P}(x_2 | x_1) \mathbb{P}(x_3 | x_1) \mathbb{P}(x_4 | x_3)$$

EXERCISE 13. What dependencies does the following factorisation imply?

$$\mathbb{P}(\mathbf{x}) = \mathbb{P}(x_1) \mathbb{P}(x_2 | x_1) \mathbb{P}(x_3 | x_1) \mathbb{P}(x_4 | x_2, x_3)$$



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.

4.1.1 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 4.1.1. *If $x_i \mid \mathbf{x}_D \perp\!\!\!\perp \mathbf{x}_B$ then*

$$\mathbb{P}(x_i \mid \mathbf{x}_B, \mathbf{x}_D) = \mathbb{P}(x_i \mid \mathbf{x}_D)$$

This implies

$$\mathbb{P}(x_i \mid \mathbf{x}_B = b, \mathbf{x}_D) = \mathbb{P}(x_i \mid \mathbf{x}_B = b', \mathbf{x}_D)$$

so we can measure independence by seeing how the distribution of x_i changes when we vary \mathbf{x}_B , keeping \mathbf{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 25.

$$\|\mathbb{P}(a \mid y, z) - \mathbb{P}(a \mid y)\|_1$$

which for discrete a, y, z is:

$$\max_{i,j} \|\mathbb{P}(a \mid y = i, z = j) - \mathbb{P}(a \mid y = i)\|_1 = \max_{i,j} \left\| \sum_k \mathbb{P}(a = k \mid y = i, z = j) - \mathbb{P}(a = k \mid y = i) \right\|_1.$$

See also `src/fairness/ci_test.py`

Coin tossing, revisited

EXAMPLE 26. The Beta-Bernoulli prior



Figure 4.7: Graphical model for a Beta-Bernoulli prior

$$\theta \sim \text{Beta}(\xi_1, \xi_2), \quad \text{i.e. } \xi \text{ are Beta distribution parameters} \quad (4.1.3)$$

$$x | \theta \sim \text{Bernoulli}(\theta), \quad \text{i.e. } P_\theta(x) \text{ is a Bernoulli} \quad (4.1.4)$$

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, ξ 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 $\xi(\theta | x)$.

EXAMPLE 27. An alternative model for coin-tossing This is an elaboration of Example 12 for hypothesis testing.

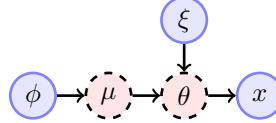


Figure 4.8: Graphical model for a hierarchical prior

- μ_1 : A Beta-Bernoulli model with $\text{Beta}(\xi_1, \xi_2)$
- μ_0 : The coin is fair.

$$\theta | \mu = \mu_0 \sim \mathcal{D}(0.5), \quad \text{i.e. } \theta \text{ is always 0.5} \quad (4.1.5)$$

$$\theta | \mu = \mu_1 \sim \text{Beta}(\xi_1, \xi_2), \quad \text{i.e. } \theta \text{ has a Beta distribution} \quad (4.1.6)$$

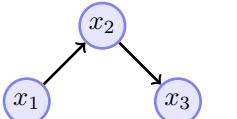
$$x | \theta \sim \text{Bernoulli}(\theta), \quad \text{i.e. } P_\theta(x) \text{ is Bernoulli} \quad (4.1.7)$$

Here the posterior over the two models is simply

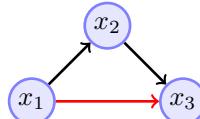
$$\phi(\mu_0 | x) = \frac{P_{0.5}(x)\phi(\mu_0)}{P_{0.5}(x)\phi(\mu_0) + \mathbb{P}_{\mu_1}(x)\phi(\mu_1)}, \quad \mathbb{P}_{\mu_1(x)} = \int_0^1 P_\theta(x) d\xi(\theta).$$

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 4.9: The two alternative models

EXAMPLE 28. Assume data $D = \{x_1^t, x_2^t, x_3^t \mid t = 1, \dots, T\}$ with $x_i^t \in \{0, 1\}$. First consider model Θ_0 where the following conditional independence holds

$$P_\theta(x_3 | x_2, x_1) = P_\theta(x_2 | x_1), \quad \forall \theta \in \Theta_0.$$

In the alternative model Θ_1 there is no independence assumption. So the likelihood for either a model in either set is

$$P_\theta(D) = \prod_t P_\theta(x_3^t | x_2^t) P_\theta(x_2^t | x_1^t) P_\theta(x_1^t), \quad \theta \in \Theta_0 \quad (4.1.8)$$

$$P_\theta(D) = \prod_t P_\theta(x_3^t | x_2^t, x_1^t) P_\theta(x_2^t | x_1^t) P_\theta(x_1^t), \quad \theta \in \Theta_1 \quad (4.1.9)$$

The parameters for this example can be defined as follows

$$\theta_1 \triangleq P_\theta(x_1^t = 1) \quad (\mu_0, \mu_1)$$

$$\theta_{2|1}^i \triangleq P_\theta(x_2^t = 1 | x_1^t = i) \quad (\mu_0, \mu_1)$$

$$\theta_{3|2}^j \triangleq P_\theta(x_3^t = 1 | x_2^t = j) \quad (\mu_0)$$

$$\theta_{3|2,1}^{i,j} \triangleq P_\theta(x_3^t = 1 | x_2^t = j, x_1^t = i) \quad (\mu_1)$$

We model each one of these parameters with a separate Beta-Bernoulli distribution.

4.1.2 Hierarchical Bayesian models

Given some data D , the Bayesian approach would involve specifying a hierarchical prior ξ 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 $\xi_i(\theta)$ over Θ_i , so that we obtain the following hierarchical model



Figure 4.10: 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.

$$\mu_i \sim \phi \quad (4.1.10)$$

$$\theta | \mu = \mu_i \sim \xi_i \quad (4.1.11)$$

Marginal likelihood

This gives the the following marginal likelihood for the combined models and each of the models respectively.

$$\mathbb{P}_\phi(D) = \phi(\mu_0) \mathbb{P}_{\mu_0}(D) + \phi(\mu_1) \mathbb{P}_{\mu_1}(D) \quad (4.1.12)$$

$$\mathbb{P}_{\mu_i}(D) = \int_{\Theta_i} P_\theta(D) d\xi_i(\theta). \quad (4.1.13)$$

Model posterior

$$\phi(\mu | D) = \frac{\mathbb{P}_\mu(D)\phi(\mu)}{\sum_i \mathbb{P}_{\mu_i}(D)\phi(\mu_i)} \quad (4.1.14)$$

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

$$\int_{\Theta} P_{\theta}(D) d\xi(\theta) \approx \sum_{n=1}^N P_{\theta_n}(D) + O(1/\sqrt{N}), \quad \theta_n \sim \xi \quad (4.1.15)$$

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 $\max_{\theta} P_{\theta}(D)$. 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 ξ . Then it holds

Importance sampling

For any two measures ξ, ψ on Θ , we can write:

$$\int_{\Theta} P_{\theta}(D) d\xi(\theta) = \int_{\Theta} P_{\theta}(D) \frac{d\psi(\theta)}{d\xi(\theta)} d\xi(\theta) = \int_{\Theta} P_{\theta}(D) \frac{d\xi(\theta)}{d\psi(\theta)} d\psi(\theta) \approx \sum_{n=1}^N P_{\theta}(D) \frac{d\xi(\theta_n)}{d\psi(\theta_n)}, \quad \theta_n \sim \psi \quad (4.1.16)$$

This allows us to estimate the marginal likelihood with respect to a belief ξ by sampling from an alternative belief ψ .

Sequential updating of the marginal likelihood

$$\mathbb{P}_{\xi}(D) = \mathbb{P}_{\xi}(x_1, \dots, x_T) \quad (4.1.17)$$

$$= \mathbb{P}_{\xi}(x_2, \dots, x_T | x_1) \mathbb{P}_{\xi}(x_1) \quad (4.1.18)$$

$$= \prod_{t=1}^T \mathbb{P}_{\xi}(x_t | x_1, \dots, x_{t-1}) \quad (4.1.19)$$

$$= \prod_{t=1}^T \int_{\Theta} P_{\theta_n}(x_t) d\xi(\theta | x_1, \dots, x_{t-1}) \underbrace{\qquad}_{\text{posterior at time } t} \quad (4.1.20)$$

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 29 (Beta-Bernoulli). The marginal predictive distribution for a Beta-Bernoulli prior is

$$\mathbb{P}_\xi(x_t = 1 \mid x_1, \dots, x_{t-1}) = \frac{\alpha_t}{\alpha_t + \beta_t},$$

with $\alpha_t = \alpha_0 + \sum_{n=1}^{t-1} x_n$, $\beta_t = \beta_0 + \sum_{n=1}^{t-1} (1 - x_n)$

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



Figure 4.11: In some cases, it appears as though automating this procedure might lead to better outcomes. But is that generally true?

4.2 Fairness in machine learning

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 blacks¹

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.

¹Pro-publica, 2016

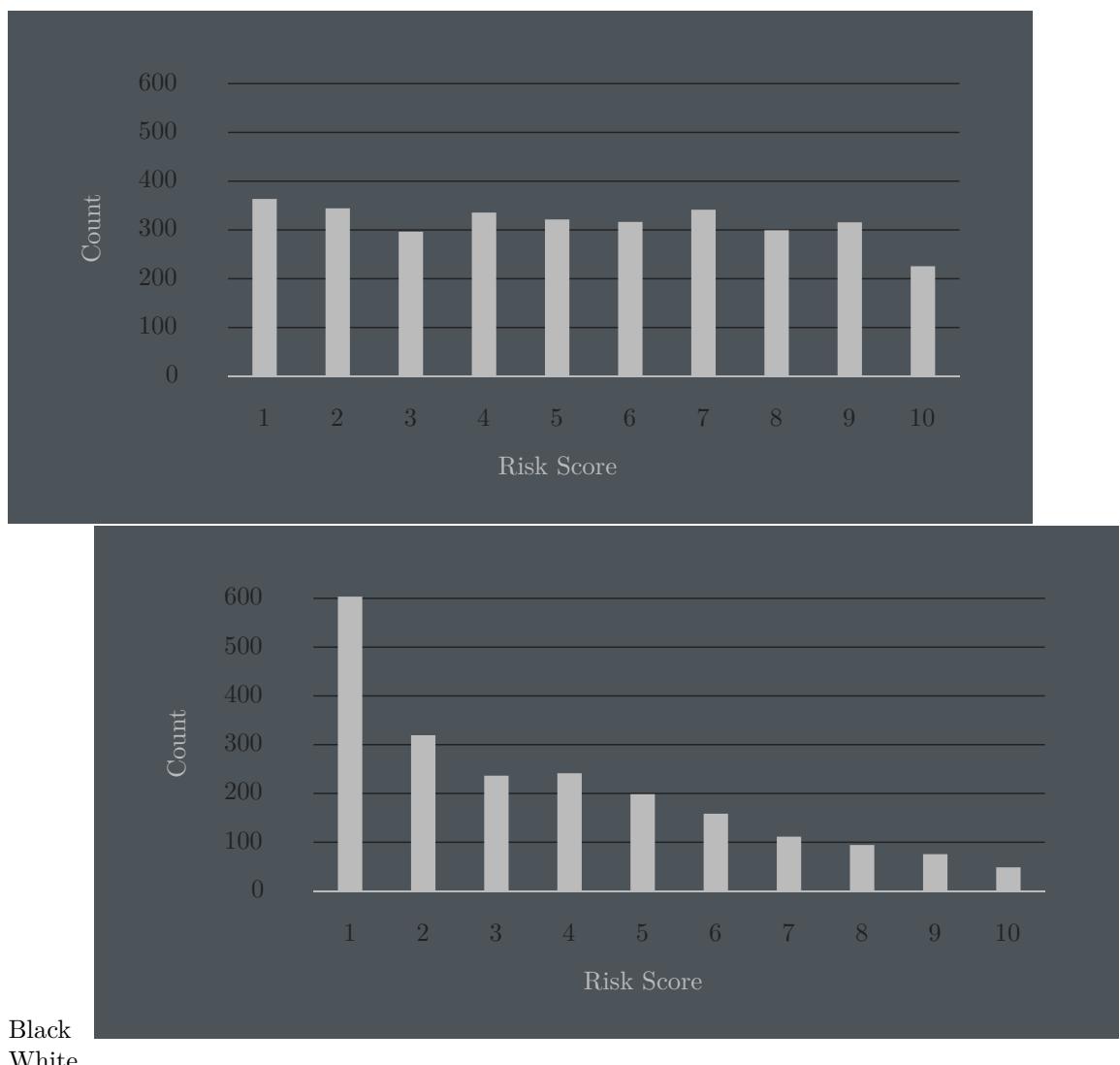


Figure 4.12: Apparent bias in risk scores towards black versus white defendants.

But scores equally accurately predict recidivism²

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.

²Washington Post, 2016

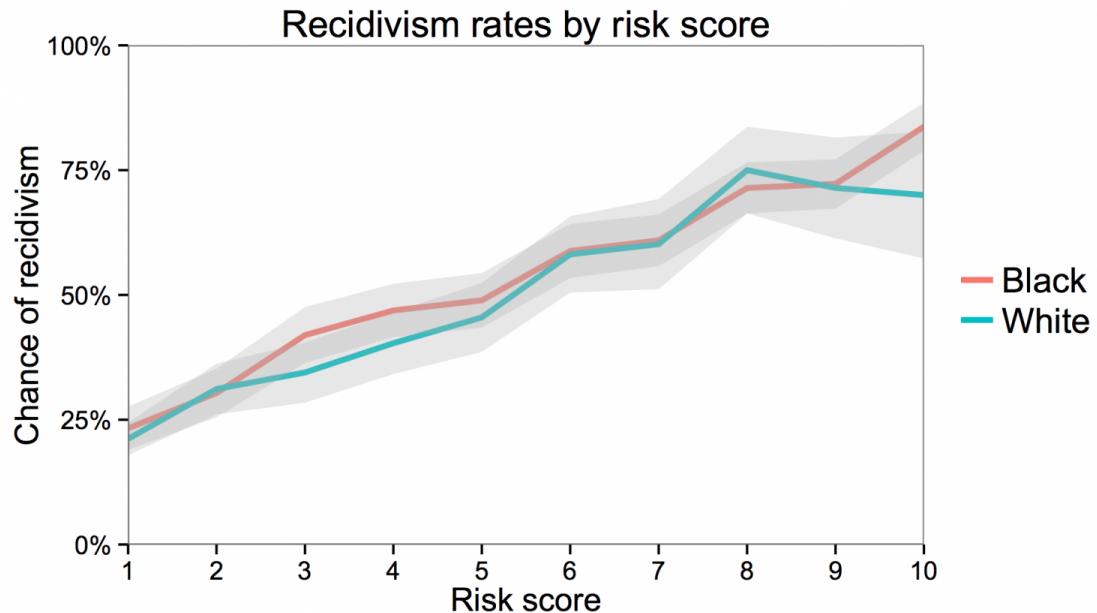


Figure 4.13: Recidivism rates by risk score.

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?

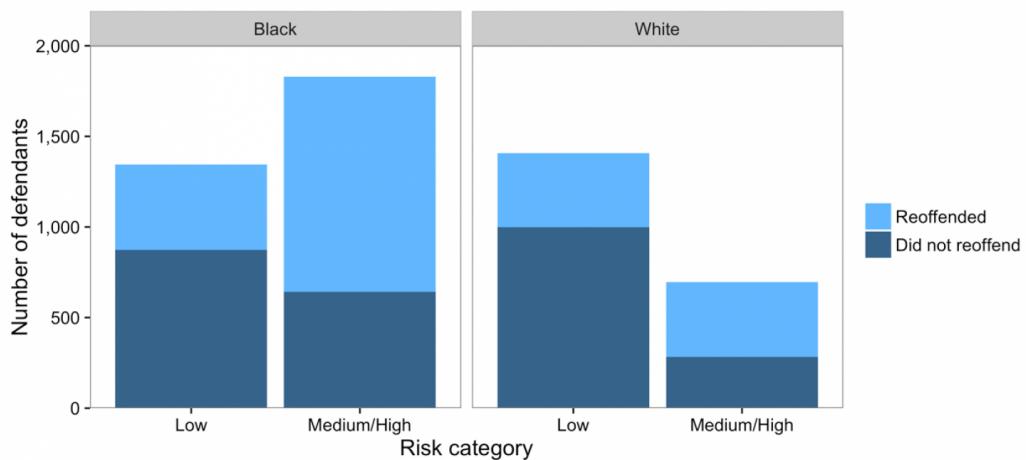


Figure 4.14: Score breakdown based on recidivism rates.

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.

4.3 Concepts of fairness

Bail decisions, revisited

Let us think of this problem in terms of bail decisions made by a judge using some policy π with $\pi(a | x)$ being the probability that the judge decides a when she observes x . Let y be the outcome, which may or may not depend on a . In this particular case, a is either release or jail. And y is appears for trial or not. If we accept the tenets of decision theory, there is also a utility function $U(a, y)$ defined on which the judge bases her decision.

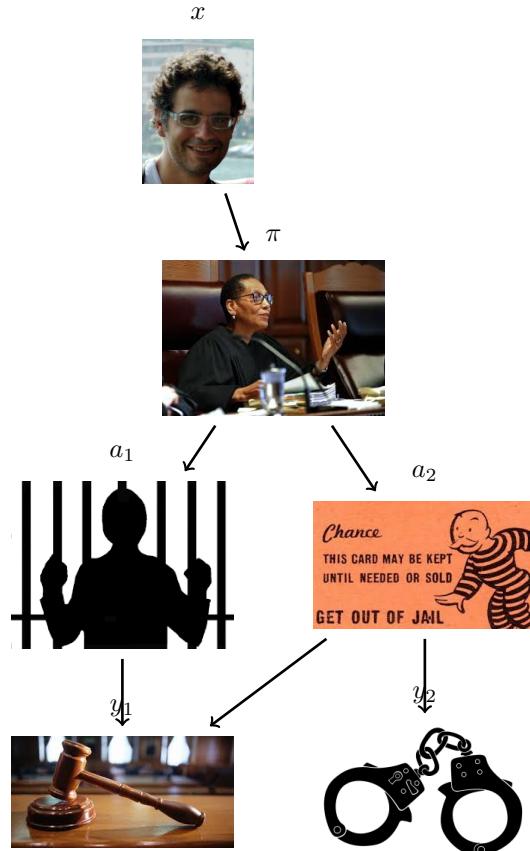


Figure 4.15: The bail decision process, simplified.

4.3.1 Group fairness and conditional independence

So how can we reframe the above fairness notions in a more precise way? Both of them involve conditional independence between y, a and a sensitive attribute z , such as race. The first notion says that the actions of the judge (or equivalently, the scores of the algorithm) are *calibrated* with respect to the outcomes. The second says that they are *balanced*, so that were the outcome known to the judge, she would be making a decision independently of the defendant's race. Both of these conditions were discussed in a more restricted setting by

Definition 4.3.1 (Calibration). A policy π is calibrated for parameter θ with respect to z if

$$\mathbb{P}_\theta^\pi(y | a, z) = \mathbb{P}_\theta^\pi(y | a), \quad \forall a, z. \quad (4.3.1)$$

You will observe that calibration here means that

$$y \perp\!\!\!\perp z | a, \theta, \pi$$

i.e. that y is independent of z given the judge's action a , so the distribution of outcomes is the same for every one of our actions no matter what the value of z is.

EXAMPLE 30. Figure 4.13 shows how actual recidivism (y) relates to risk scores (a) for different races (z). In that case, it is apparent that, at least approximately, $P(y | a, z) = P(y | a, z')$, so the COMPAS method satisfies calibration.

Definition 4.3.2 (Balance). A policy π is balanced for parameter θ with respect to z if:

$$\mathbb{P}_\theta^\pi(a | y, z) = \mathbb{P}_\theta^\pi(a | y), \quad \forall y, z. \quad (4.3.2)$$

On the other hand, balance means that

$$a \perp\!\!\!\perp z | y,$$

i.e. that a is independent of z given the true outcome y .³

EXAMPLE 31. Figure 4.14 shows how risk scores (a) relate to actual recidivism (y) for different races (z). In that case, it is apparent that, at least approximately, $P(a | y, z) \neq P(a | y, z')$, so the COMPAS method does not satisfy balance.

4.3.2 Individual fairness and meritocracy.

A different concept of fairness is meritocracy. For example, if one candidate for a job is better than another candidate, perhaps that candidate should be taken for the job.

Let us consider merit from the point of view of the decision maker, who can either hire ($a_t = 1$) or not hire ($a_t = 0$) the t -th applicant. If the applicant has characteristics x_t and merit y_t , the DM's decision has utility $U(a_t, y_t)$. In order to model meritocracy, we assign an inherent *quality* to y , expressed as an ordering, so that $U(1, y) \geq U(1, y')$ if $y \geq y'$. Assuming $P_\theta(x_t, y_t)$ is known to the DM then clearly she should make the decision by solving the following maximisation problem:

Meritocratic decision

$$a_t(\theta, x_t) \in \arg \max_a \mathbb{E}_\theta(U | a, x_t) = \int_y U(a_t, y) \mathbb{E}_\theta(U | a_t, x_t) \quad (4.3.3)$$

Here, the notion of meritocracy is defined through our utility function. Although it would be better to consider the candidate's utility instead, this is in practice difficult, because we'd have to somehow estimate each individual's utility function. Finally, we are taking the expectation here is because we may not know for certain what the quality attribute of a given person might be.

³This definition only really makes sense when y does not depend on a at all. When this is not the case, it's easy to construct a random variable y' that does not depend on a so that y can be written as a function $y(y', a)$. Then we can achieve balance with respect to y' .

Smooth fairness

It makes sense to combine the idea of meritocracy with that of similarity. That is, similar people should be treated similarly. This means that we should find a policy π that maximises utility U and makes similar decisions for similar people.

Let \mathcal{X} be equipped with a metric ρ , and let D be a divergence between distributions, such as the KL-divergence. We can then formalise the above intuition as follows:

$$D[\pi(a | x), \pi(a | x')] \leq \rho(x, x'). \quad (4.3.4)$$

This is a so-called Lipschitz condition on the policy, and is illustrated in the figure below.

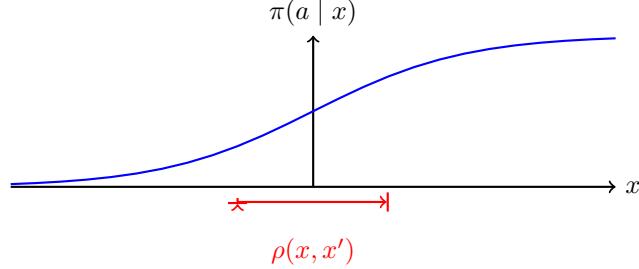


Figure 4.16: A Lipschitz function

If we wish to find the optimal policy that satisfies this constraint, then it is naturally to think about this as a constrained optimisation problem

The constrained maximisation problem

$$\max_{\pi} \{U(\pi) \mid \rho(x, x') \leq \epsilon\} \quad (4.3.5)$$

The practical problem in this framework is how to define the metric ρ in the first place. A natural idea is to simply use the probability of y as a metric. Then it doesn't matter how different x, x' appear: the only important thing is how what is the distribution of y implied by the different attributes.

The metric ρ in terms of the distribution $y | x$

Firstly, assume you somehow have some probability law \mathbb{P} over the variables of interest. To fix ideas, let us consider the total variation norm and assume $y \in \mathcal{Y}$ takes a finite number of values. Then we can define the metric as

$$\rho(x, x') \triangleq \|\mathbb{P}(y | x) - \mathbb{P}(y | x')\|_1 = \sum_{i \in \mathcal{Y}} \|\mathbb{P}(y = i | x) - \mathbb{P}(y = i | x')\|_1 \quad (4.3.6)$$

4.3.3 A unifying view of fairness

In both cases, we defined conditional independence for a fixed probability distribution $P_\theta(x, y, z)$ on the various variables. We also considered meritocratic fairness with respect to θ . First of

all, we can simply define a function F that measures the amount of fairness violation for a given parameter value. As an example, the deviation from calibration of a policy π for a given parameter value θ can be written as:

$$F_{\text{calibration}}(\theta, \pi) \triangleq \max_{a,z} \sum_i |\mathbb{P}_\theta^\pi(y = i | a, z) - \mathbb{P}_\theta^\pi(y = i | a, z')| \quad (4.3.7)$$

Similarly, the deviation from calibration can be written as:

$$F_{\text{balance}}(\theta, \pi) \triangleq \max_{a,z} \sum_i |\mathbb{P}_\theta^\pi(a = i | y, z) - \mathbb{P}_\theta^\pi(a = i | y, z')| \quad (4.3.8)$$

Finally, we can also create a deviation for the constraints in the maximisation problem of (4.3.5), so that

$$F_{\text{merit}}(\theta, \pi) \triangleq \exp(\rho(x, x') - \epsilon). \quad (4.3.9)$$

If we the decision maker did not care about fairness, she could just maximise some utility function U of interest. In order to take fairness into account, we define the value of the policy to be a linear combination between the original utility and the fairness violation.

The value of a policy

Let λ represent the trade-off between utility and fairness. The combined value for a given parameter θ is

$$V(\lambda, \theta, \pi) = \lambda \overbrace{U(\theta, \pi)}^{\text{utility}} - \underbrace{(1 - \lambda)F(\theta, \pi)}_{\text{fairness violation}} \quad (4.3.10)$$

Remark 4.3.1. It is also possible to define the above problem in terms of a constrained optimisation

$$\max \{U(\theta, \pi) \mid F(\theta, \pi) \leq \epsilon\}.$$

The advantage of specifying just a single value function to maximise is that unconstrained problems are usually simpler to handle.

4.3.4 Bayesian fairness

The previous section defined a simple universal framework for trading off utility and fairness in terms of an unconstrained optimisation problem. This involved defining the value of a policy for a given trade-off λ and an underlying distribution with parameter θ . However, in a learning context this parameter cannot be assumed to be known. One solution to this problem is to assume a Bayesian approach and simply specify a subjective probability distribution ξ over parameters. Typically, this would be calculated from a prior distribution and some available data.

The Bayesian decision problem

In the Bayesian setting, we simply need to maximise the expected V over all possible model parameters. Assume we have some probability ξ over the parameters. Typically, ξ would be a posterior distribution calculated from some prior and some data, but for simplicity we just write $\xi(\theta)$ in this section. Then the expected value of a policy π is simply the following.

The Bayesian value of a policy

With some abuse of notation, we also define the value for a subjective belief, modelled as a distribution ξ , over parameters

$$V(\lambda, \xi, \pi) = \int_{\Theta} V(\lambda, \theta, \pi) d\xi(\theta). \quad (4.3.11)$$

This is simply the expected value of our policy over each possible parameter θ , with respect to the belief ξ .

Maximising the value of this policy should be possible with stochastic gradient ascent we can easily obtain samples from ξ .

4.3.5 Further reading

Recently algorithmic fairness has been studied quite extensively in the context of statistical decision making. Dwork et al.¹⁰, Kilbertus et al.¹⁷, Kleinberg et al.¹⁸? ? studied fairness under the one shot statistical decision making framework in this chapter. Jabbari et al.¹⁵, Joseph et al.¹⁶ studied fairness in sequential decision making settings. Fairness has also been studied in other machine learning topics, such as clustering⁷, natural language processing⁴ and recommendation systems⁶.

Satisfying fairness constraints while maximizing expected utility in a more general setting in the conditional independence setting has been considered by ?, under a specific model. Dimitrakakis et al.⁹ consider the Bayesian setting, where the model is unknown.

In the individual fairness-as-meritocracy framework, Dwork et al.¹⁰ and look for decision rules that are smooth in a sense that similar *individuals* are treated similarly.

Finally, ? which considers the problem of uncertainty from the point of view of causal modelling.

Online resources

- COMPAS analysis by propublica <https://github.com/propublica/compas-analysis>
- Open policing database <https://openpolicing.stanford.edu/>

Learning outcomes

Understanding

- Graphical models.
- Conditional independence.
- Fairness as independence.

- Fairness as meritocracy.

Skills

- Be able to specify a graphical model capturing dependencies between variables.
- Be able to verify if a policy satisfies a fairness condition.

Reflection

- When looking at sensitive attributes, how easy is it to determine fairness?
- How should we balance the needs of individuals, the decision maker and society?
- Does having more data available make it easier to achieve fairness?
- How do these fairness concept relate to classical game theory ideas?

4.4 Project: Credit risk for mortgages

Consider a bank that must design a decision rule for giving loans to individuals. In this particular case, some of each individual's characteristics are partially known to the bank. We can assume that the insurer has a linear utility for money and wishes to maximise expected utility. Assume that the t -th individual is associated with relevant information x_t , sensitive information z_t and a potential outcome y_t , which is whether or not they will default on their mortgage. For each individual t , the decision rule chooses $a \in \mathcal{A}$ with probability $\pi(a_t = a | x_t)$.

As an example, take a look at the historical data in `data/credit/german.data-numeric`, described in `data/credit/german.doc`. Here there are some attributes related to financial situation, as well as some attributes related to personal information such as gender and marital status.

A skeleton for the project is available at <https://github.com/olethrosdc/ml-society-science/tree/master/src/project-1>. Start with `random_banker.py` as a template, and create a new module `name_banker.py`. You can test your implementation with the `TestLending.py` program.

For ensuring progress, the project is split into three parts:

4.4.1 Deadline 1: September 18

The first part of the project focuses on a baseline implementation of a banker module.

1. Design a policy for giving or denying credit to individuals, given their probability for being credit-worthy. Assuming that if an individual is credit-worthy, you will obtain a return on investment of $r = 5\%$ per month.⁴ Take into account the length of the loan to calculate the utility through `NameBanker.expected_utility()`. Assume that the loan is either fully repaid at the end of the lending period n , or not at all to make things simple. If an individual is not credit-worthy you will lose your investment of m credits, otherwise you will gain $m[(1 + r)^n - 1]$. Ignore macroeconomic aspects, such as inflation. In this section, simply assume you have a model for predicting creditworthiness as input to your policy, which you can access `NameBanker.get_proba()`.
2. Implement `NameBanker.fit()` to fit a model for calculating the probability of creditworthiness from the german data. Then implement `NameBanker.predict_proba()` to predict the probability of the loan being returned for new data. What are the implicit assumptions about the labelling process in the original data, i.e. what do the labels represent?
3. Combine the model with the first policy to obtain a policy for giving credit, given only the information about the individual and previous data seen. In other words, implement `Namebanker.get_best_action()`.
4. Finally, using `TestLending.py` as a baseline, create a jupyter notebook where you document your model development. Then compare your model against `RandomBanker`.

Here is a brief explanation of the purposes of each function:

⁴Feel free to experiment with different values for this.

<code>fit()</code>	Create a model estimating probabilities for <code>predict_proba()</code>
<code>predict_proba()</code>	Return the probability that a person will return the loan
<code>expected_utility()</code>	Calculate the expected utility of a particular action for a given individual
<code>get_best_action()</code>	Return the action maximising expected utility

Table 4.1: Function explanataion

4.4.2 Deadline 2: October 2

The second part of the project focuses on issues of reproducibility, reliability, privacy and fairness. That is, how desirable would it be to use this model in practice? Here are some sample questions that you can explore, but you should be free to think about other questions.

1. Is it possible to ensure that your policy maximises revenue? How can you take into account the uncertainty due to the limited and/or biased data? What if you have to decide for credit for thousands of individuals and your model is wrong? How should you take that type of risk into account?⁵
2. Does the existence of this database raise any privacy concerns? If the database was secret (and only known by the bank), but the credit decisions were public, how would that affect privacy? (a) Explain how you would protect the data of the people in the training set. (b) Explain how would protect the data of the people that apply for new loans. (c) *Implement* a private decision making mechanism for (b),⁶ and estimate the amount of loss in utility as you change the privacy guarantee.

4.4.3 Deadline 3: October 16

Choose one concept of fairness, e.g. balance of decisions with respect to gender. How can you measure whether your policy is fair? How does the original training data affect the fairness of your policy? To help you in this part of the project, here is a list of guiding questions.

- Identify sensitive variables. Do the original features already imply some bias in data collection?
- Analyse the data or your decision function with simple statistics such as histograms.
- For balance (or calibration), measure the total variation of the action (or outcome) distribution for different outcomes (or actions) when the sensitive variable varies.
- Advanced: What would happen if you were looking at fairness by also taking into account the amount of loan requested?
- Advanced: Using stochastic gradient descent, find a policy that balances out fairness and utility.

Submit a final report about your project, either as a standalone PDF or as a jupyter notebook. For this, you can imagine playing the role of an analyst who submits a possible decision rule to the bank, or the authorities. You'd like to show that your decision rule is quite likely to make a

⁵You do not need to implement anything specific for this to pass the assignment, but you should outline an algorithm in a precise enough manner that it can be implemented. In either case you should explain how your solution mitigates this type of risk.

⁶If you have already implemented (a) as part of the tutorial, feel free to include the results in your report.

profit, that it satisfies *some* standard of privacy and that it does not unduly discriminate between applicants. You should definitely point out any possible *deficiencies* in your analysis due to your assumptions, methodology, or the available data from which you are drawing conclusions.

Chapter 5

Recommendation systems

Structured learning problems involve multiple latent variables with a complex structure. These range from clustering and speech recognition to DNA and biological and social network analysis. Since structured problems include relationships between many variables, they can be analysed using graphical models.

5.1 Recommendation systems



Figure 5.1: The recommendation problem

In many machine learning applications, we are dealing with the problem of proposing one or more alternatives to a human. The human can accept zero or more of these choices. As an example, when using an internet search engine, we typically see two things: (a) A list of webpages matching our search terms (b) A smaller list of advertisements that might be relevant to our search. At a high level,

The recommendation problem

At time t

1. A customer x_t appears. For the internet search problem, x_t would at least involve the search term used.
2. We present a choice a_t . For the matching website, the choice is ranked list of websites. For the advertisements, however, it is typical
3. The customer chooses y_t . This might include selecting one or more of items suggested in a_t . The choice of the customer may not be directly visible.
4. We obtain a reward $r_t = \rho(a_t, y_t) \in \mathbb{R}$. Typically this is a payment either from the customer or an advertiser.

The two problems in recommendation systems

-
- The modelling (or prediction) problem. Given the data, how to e.g. predict what movies a user likes and dislikes.
 - The recommendation problem. What movie(s) to actually recommend to a user.

Although closely linked, those two problems have different evaluation metrics. The recommendation problem is harder, especially because the data do not tell us what our recommendation have been in the past, but only what users watched.

How to predict user preferences?

In order for us to be able to provide good recommendations, we need to be able to predict user preferences. In the case of movies, preferences of a person for a movie can be expressed in terms of the hypothetical rating that a user would give to a movie. As Fig. 5.2 shows, we frequently do have data about individual user ratings for movies, and we would like to somehow use those.

Example: Item-based CF

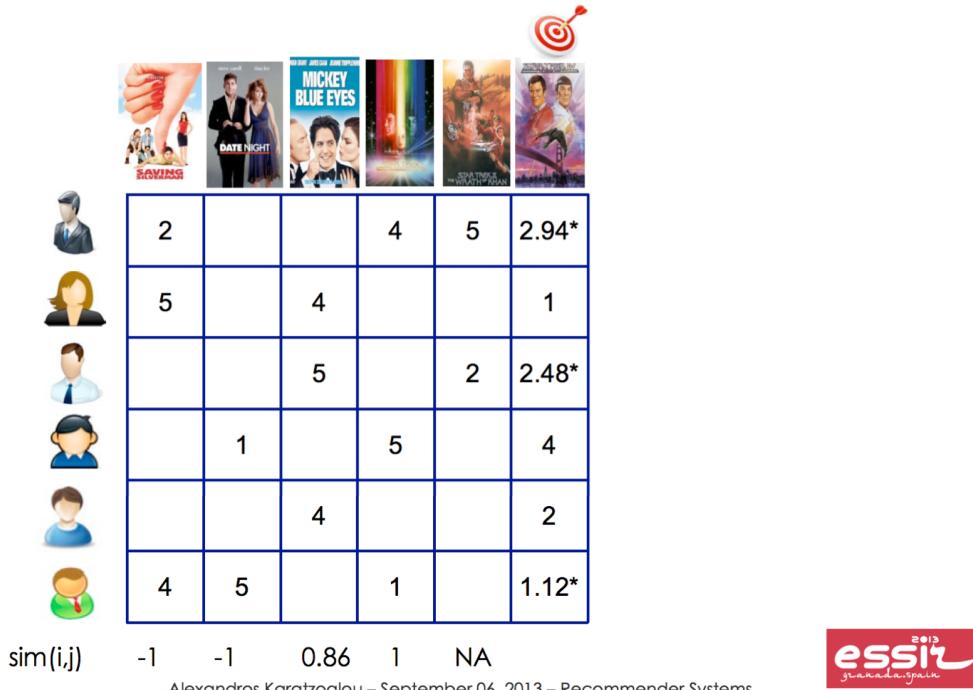


Figure 5.2: User ratings

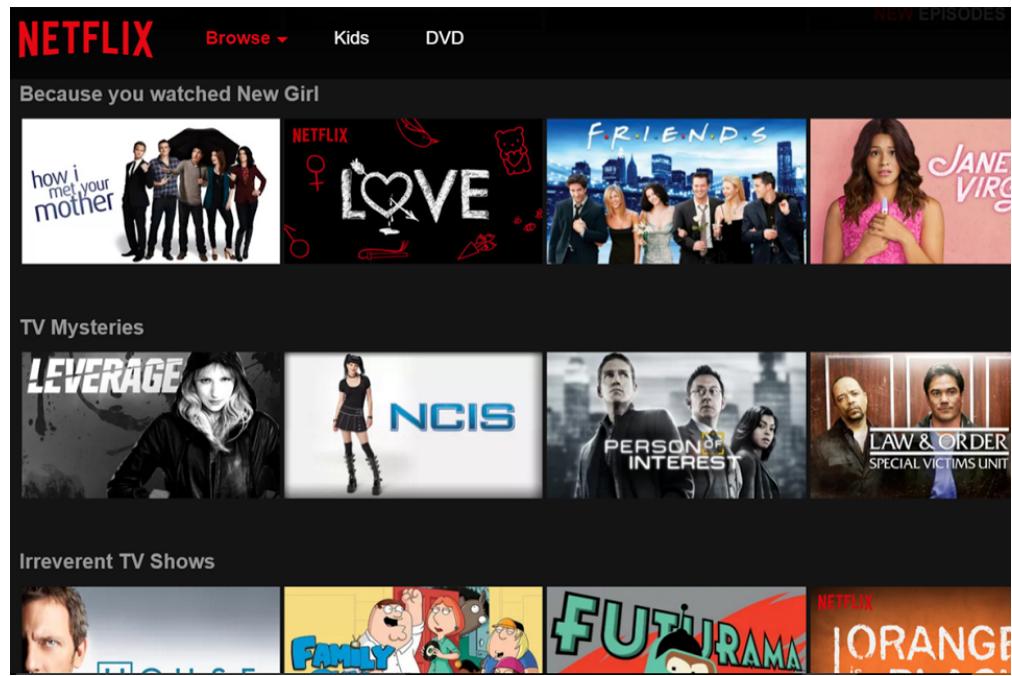


Figure 5.3: What to recommend?

EXAMPLE 32. In the case of Netflix and related services, we would like to suggest movies to users which they are more likely to watch, as shown in Figure 5.3. However, how can we tell which movies those can be? It is probably not useful to just recommend them to rewatch a previously watched movie. We need to somehow take into account information across our user database: if somebody watched mostly the same films as you, then maybe you'd be interested in watching those movies she has that you haven't seen.

In the Netflix catalogue, in particular, users also post reviews of the movies they have watched, as shown in Figure 5.2. This allows us to be able to guess the ratings of users from previous user's ratings.

Predictions based on similarity

Content-based filtering.

- Users typically like similar items. For example, a horror movie fan typically rates horror movies highly.
- That means we can use one user's ratings and *item information* to predict their ratings for other items. In this scenario, we do not need to take into account the ratings of other people.

Collaborative filtering

- *Similar users have similar tastes.* For example, consider two users t, u who have each watched a set of movies \mathcal{M}_t and \mathcal{M}_u respectively, and $\mathcal{M}_{t,u} = \mathcal{M}_t \cap \mathcal{M}_u$ is the set of common movies. If their ratings are the same for those movies, i.e. $x_{t,m} = x_{u,m} \forall m \in \mathcal{M}_{t,u}$, then it's a good guess that they might have the same ratings for movies they have not both watched.
- That means we can use similar user's *ratings* to predict the ratings for other users. The advantage is that ratings are readily available. The disadvantage is that new users have too few data to be matched to other users.

k -NN for similarity

EXERCISE 14. • Define a distance $d : \mathcal{X}^M \times \mathcal{X}^M \rightarrow \mathbb{R}_+$ between user ratings.

- Apply a k -NN-like algorithm to prediction of user ratings from the dataset.

Similarity between users

Let us define a similarity $w_{ij} \geq 0$ between two users so that

$$\sum_{j \neq i} w_{ij} = 1, \quad w_{ij}^m \triangleq w_{ij} \mathbb{I}\{x_{j,m}\} / \sum_k w_{ik} \mathbb{I}\{x_{k,m}\}.$$

There w_{ij}^m only considers those users who have rated movie m , so that $\sum_{j \neq i: x_{j,m} > 0} w_{ij}^m = 1$. The overall similarity w_{ij} itself does not have to sum up to one, but it does need to be non-negative.

EXAMPLE 33 (k -nearest neighbours). $w_{ij} = 1/k$ for the k nearest neighbours with respect to d .

EXAMPLE 34 (Weighted distance).

$$w_{ij} = \frac{\exp[-d(i,j)]}{\sum_{k \neq i} \exp[-d(i,j)]}$$

Inferred ratings

Then we can define the inferred ratings to be the weighted average rating.

$$\hat{x}_{u,m} = \sum_{j \neq u} w_{uj}^m x_{u,m}.$$

But if the value $x_{u,m}$ is missing, we need to only

A naive distance metric

A simple idea is to just look at the difference between the raw ratings in terms of the L1 norm:

$$d(i,j) \triangleq \|\mathbf{x}_i - \mathbf{x}_j\|_1.$$

However this has the problem that this makes users who have watched different amounts of movies look very different.

Ignoring movies which are not shared.

We perhaps would only like to look at similarity for users with respect to which movies they have rated. This would lead to a distance such as

$$d(i, j) \triangleq \sum_m \mathbb{I}\{x_{i,m} \wedge x_{j,m}\} |x_{i,m} - x_{j,m}|$$

Using side-information

In some cases, we have additional information about the products or users. For example, users might be connected in a social network. Then we can tag users as similar if they are strongly connected, even if one of them has few or no movie ratings.

Inferring a latent representation

Rather than going through specific algorithms for calculating similarities, we can think about learning a latent representation from the data. For example we could infer a network of users, and so a distance, from the individual ratings:

$$d(i, j) \triangleq f(\mathbf{x}_i, \mathbf{x}_j, \theta)$$

More generally, we can try and infer some other latent representation.

5.1.1 Least squares representation

Latent representation

The predictive model

- x_{um} rating of user u for movie m .
- $r_{um} = \mathbb{I}\{x_{um} > 0\}$ indicates which movies are rated.
- $\mathbf{z}_m \in \mathbb{R}^n$: an n -dimensional representation of a movie.
- $\mathbf{c}_u \in \mathbb{R}^n$: an n -dimensional representation of a user.

Given \mathbf{C}, \mathbf{Z} , our predicted movie rating can be written as

$$\hat{x}_{u,m} \triangleq \mathbf{c}_u^\top \mathbf{z}_m, \quad \hat{\mathbf{X}} \triangleq \mathbf{C}^\top \mathbf{Z}.$$

On the left side we have individual ratings and on the right side, in matrix form.

We can now try and see how well this prediction fits the data for a movie. One idea is to simply use $|\hat{x}_{m,u} - x_{m,u}|$ to show the difference between the prediction and the actual rating. Now can now define the prediction *error* for a given representation \mathbf{C}, \mathbf{Z} as

$$f(\mathbf{C}, \mathbf{Z}) = \|(\mathbf{R} \circ \hat{\mathbf{X}} - \mathbf{R} \circ \mathbf{X})^\top (\mathbf{R} \circ \hat{\mathbf{X}} - \mathbf{R} \circ \mathbf{X})\|_1$$

Here multiplying with the \mathbf{R} matrix ensures that we ignore pairs with no ratings.

Alternating Least Squares²¹

$$\mathbf{C}_{t+1} = \arg \min_{\mathbf{C}} f(\mathbf{C}, \mathbf{Z}_t) + \lambda g(\mathbf{Z}, \mathbf{Z}_t) \quad (5.1.1)$$

$$\mathbf{Z}_t = \arg \min_{\mathbf{Z}} f(\mathbf{C}_t, \mathbf{Z}) + \lambda g(\mathbf{Z}_t, \mathbf{Z}) \quad (5.1.2)$$

5.1.2 Preferences as a latent variable

A simple preference model

As a simple model, we can assume that each person belongs to a *type*. Every type has the same preferences over films. In the simplest possible model, a user of type c_i that has watched a movie m will rate the film deterministically $x_{c,m}$. More generally, we can assume the following model.

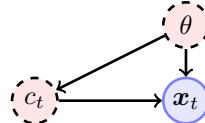


Figure 5.4: Basic preference model

EXAMPLE 35 (Discrete preference model). • User type $c \in \mathcal{C}$. For simplicity, we can think of there being a finite number of types $\mathcal{C} = \{1, \dots, n\}$.

- User ratings \mathbf{x} with $x_m \in \mathcal{X} = \{0, 1\}$ rating for movie m .
- Preference distribution

$$P_\theta(\mathbf{x}|\mathbf{c}) = \prod_{m=1}^M \theta_{m,c}^{x_m} (1 - \theta_{m,c})^{(1-x_m)}.$$

- $P_\theta(c) = \theta_c$, $\sum_c \theta_c = 1$.

A more complex preference model

As a simple model, we can assume that each person belongs to a *type*. Every type has the same preferences over films. In the simplest possible model, a user of type c_i that has watched a movie m will rate the film deterministically $x_{c,m}$. More generally, we can assume the following model.

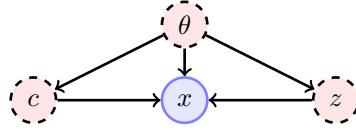


Figure 5.5: Preference model

Preference model

- User type $\mathbf{c} \in \mathcal{C}$. For simplicity, we can think of there being a finite number of types $\mathcal{C} = \{1, \dots, n\}$.
- Movie type $\mathbf{z} \in \mathcal{Z}$.
- Preference distribution

$$P_\theta(x|\mathbf{c}, \mathbf{z}) = \mathcal{N}(\mathbf{c}^\top \mathbf{z}, \sigma_\theta)$$
- Feature prior

$$P_\theta(\mathbf{c}) = \mathcal{N}(0, \lambda_\theta)$$

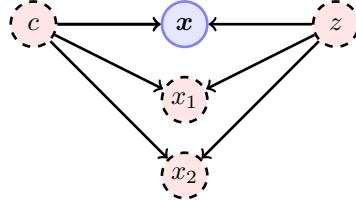
5.1.3 The recommendation problem**What to recommend**

Figure 5.6: Preference model

The recommendation problem for a given θ

$$\max_{\pi} \mathbb{E}_{\theta}^{\pi}(U | \mathbf{x}) = \max_a \sum_{c,z} U(a, y) \mathbb{P}(y | a, c, z) P_\theta(c, z | \mathbf{x}) \quad (5.1.3)$$

$$= \max_a \sum_{c,z} U(a, y) \sum_{x_a} \mathbb{P}(y | a, x_a) P_\theta(x_a | c, z) P_\theta(c, z | \mathbf{x}) \quad (5.1.4)$$

Two ways to model the effect of actions

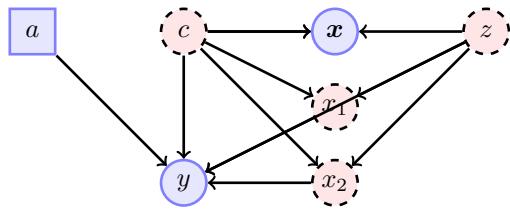


Figure 5.7: Preference model

What is the right model for the effects of our actions? In the most general case, the model could depend on the complete set of latent ratings of all the movies. However, it is hard to interpret this, as the user is also probably not aware of what these ratings are themselves. So it seems simpler and more appropriate to predict the outcome based on our action and the latent representation, especially since we will be marginalising over the individual ratings anyway.

5.2 More fun with latent variable models

Clustering is the problem of automatically segregating data of different types into clusters. When the goal is *anomaly detection*, then there are typically two clusters. When the goal is *compression* or *auto-encoding* then there are typically as many clusters as needed for sufficiently good accuracy.

Clusters as latent variables

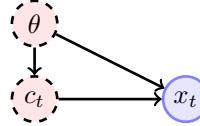


Figure 5.8: Graphical model for independent data from a cluster distribution.

The clustering distribution

The learning problem is to estimate the parameter θ describing the distribution of observations x_t and clusters c_t .

$$x_t \mid c_t = c, \theta \sim P_\theta(x|c), \quad c_t \mid \theta \sim P_\theta(c), \quad \theta \sim \xi(\theta)$$

Given a parameter θ , the clustering problem is to estimate the probability of each cluster for each new observation.

$$P_\theta(c_t \mid x_t) = \frac{P_\theta(x_t \mid c_t)P_\theta(c_t)}{\sum_{c'} P_\theta(x_t \mid c_t = c')P_\theta(c_t = c')}$$

Bayesian formulation of the clustering problem

- Prior ξ on parameter space Θ .
- Data $x^T = x_1, \dots, x_T$. Cluster assignments c^T unknown.
- Posterior $\xi(\cdot \mid x^T)$.

Posterior distribution

The data we obtain do not include the cluster assignments, but we can still formulate the posterior distribution of parameters given the data.

$$\xi(\theta \mid x^T) = \frac{P_\theta(x^T)\xi(\theta)}{\int_\Theta P_{\theta'}(x^T) d\xi(\theta')}, \quad P_\theta(x^T) = \sum_{c^T \in \mathcal{C}^T} \underbrace{P_\theta(x^T \mid c^T)}_{\text{Cluster Density}} \underbrace{P_\theta(c^T)}_{\text{Cluster prior}} \quad (5.2.1)$$

We simply need to expand the data-dependent term to include all possible cluster assignments. This is of course not trivial, since the number of assignments is exponential in T . However, algorithms such as Markov Chain Monte Carlo can be used instead.

Marginal posterior prediction

$$P_\xi(c_t | x_t, x^T) = \int_{\Theta} P_\theta(c_t | x_t) d\xi(\theta | x^T)$$

EXAMPLE 36 (Preference clustering). The learning problem is to estimate the parameter θ describing the distribution of observations x_t and clusters c_t . In this example, we can assume

$$\mathcal{C} = \{1, \dots, C\}, \quad x_{t,m} \in \{0, 1\}.$$

This means that all movies are either watched or not, and we'd simply want to predict which movie somebody is likely to watch. This allows us to use the following simple priors, splitting the parameters in two parts $\theta = (\theta_1, \theta_2)$.

Model family

$$P_{\theta_1}(c_t = c) = \theta_{1,c}, \quad c_t \sim \text{Multinomial}(\theta_1) \quad (5.2.2)$$

$$P_{\theta_2}(x_{t,m} = 1 | c_t = c) = \theta_{2,m,c} \quad x_{t,m} | c_t = c \sim \text{Bernoulli}(\theta_{2,m,c}) \quad (5.2.3)$$

Since everything is discrete, it makes sense that we can use a Multinomial model for the cluster distribution and a Bernoulli model for whether or not a movie was watched. Now we only need to specify a useful prior for each one of those. The standard priors to use, are a Beta prior for the Bernoulli and the Dirichlet for the Multinomial, as they are conjugate.

Prior

$$\theta_1 \sim \text{Dirichlet}(\gamma), \quad \theta_2 \sim \text{Beta}(\alpha, \beta) \quad (5.2.4)$$

Typically $\gamma = (1/2, \dots, 1/2)$ and $\alpha = \beta = 1/2$ to allow for the possibility of nearly deterministic behaviour.

See `src/pymc/beta_bernoulli_clustering.py`

Supervised learning

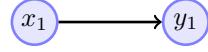
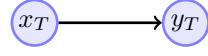
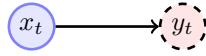


Figure 5.9: Graphical model for a classical supervised learning problem.

Semi-supervised learning

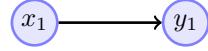
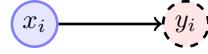
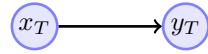
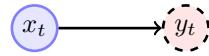


Figure 5.10: Graphical model for a classical semi-supervised learning problem.

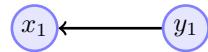
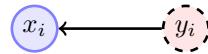
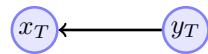
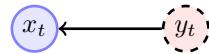


Figure 5.11: Generative version of the semi-supervised model

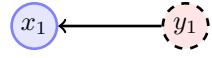
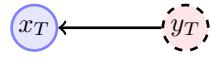
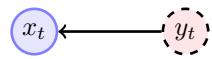


Figure 5.12: Basic unsupervised learning model

Applications

- Clustering
- Compression

5.3 Social networks

Social networks afford us another opportunity to take a look at data. We can use connections between users to infer their similarity: if two users are connected, then they are more likely to have similar preferences.

Network model

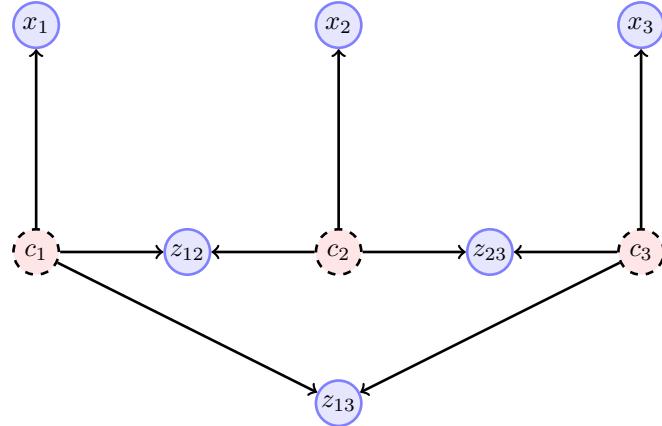


Figure 5.13: Graphical model for data from a social network.

In the model seen in Figure 5.13, each user t is characterised by their cluster membership c_t and emits data x_t . Users t, u are connected when $z_{t,u} = 1$.

5.4 Sequential structures

The simplest type of structure in data is sequences. Examples include speech, text and DNA sequences, as well as data acquired in any sequential decision making problem such as recommendation systems or robotics. Sequential data is always thought to arise from some Markovian processes, defined below.

Markov process

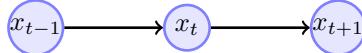


Figure 5.14: Graphical model for a Markov process.

Definition 5.4.1 (Markov process). A Markov process is a sequence of variables $x_t : \Omega \rightarrow \mathcal{X}$ such that $x_{t+1} \mid x_t \perp\!\!\!\perp x_{t-k} \forall k \leq 1$.

Application

- Sequence compression (especially with variable order Models).
- Web-search (Page-Rank)
- Hidden Markov Models.
- MCMC.

Hidden Markov model

Frequently the sequential dependency is not in the data itself, but in some hidden underlying markov process. In that case, the hidden variable x_t is the *state* of the process. The observed variable y_t is simply an observation.

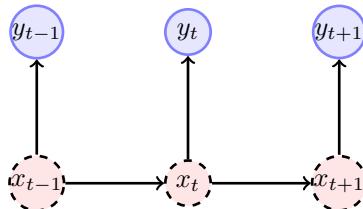


Figure 5.15: Graphical model for a hidden Markov model.

$$\begin{array}{ll} P_\theta(x_{t+1} \mid x_t) & \text{(transition distribution)} \\ P_\theta(y_t \mid x_t) & \text{(emission distribution)} \end{array}$$

For any given parater value θ , it is easy to estimate the probability distribution over states given the observations $P_\theta(x^T \mid y^T)$. As an example, if y^T is raw speech data and x^T is a sequence of

words, and θ are the parameters of our speech model, then we can obtain probabilities for every possible sequence of words that was uttered. Frequently, though, in speech recognition we are only interested in the most likely sequence of words. This makes the problem simple enough to be solved instantaneously by modern cellphones.

Application

- Speech recognition.
- Filtering (Kalman Filter).
- DNA analysis.

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*.

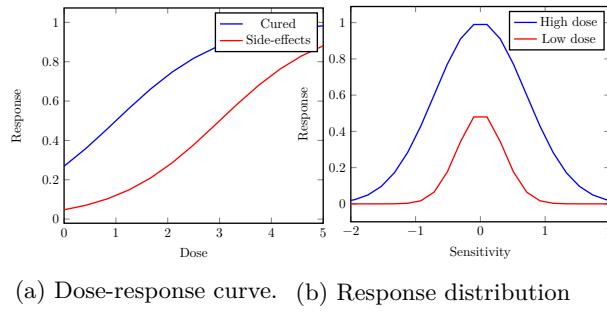


Figure 6.1: Investigation the response of the population to various doses of the drug.

EXAMPLE 37 (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?

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 38 (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 2.4. 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.

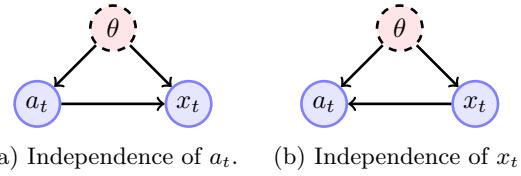
Inferring individual effects

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?



EXAMPLE 39. 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:

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

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

$$\begin{aligned} \theta'_a &= \sum_x \theta_{x,a}, & \theta'_{x|a} &= \theta_{x,a}/\theta'_a \\ \theta''_x &= \sum_a \theta_{x,a}, & \theta'_{a|x} &= \theta_{x,a}/\theta''_x. \end{aligned}$$

This means we can define prior distributions ξ, ξ', ξ'' 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.

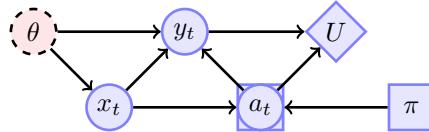


Figure 6.4: A typical decision diagram where x_t : individual information, y_t : individual result, a_t : action, π : policy

EXAMPLE 40 (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 41 (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 .

- x_t : User information (random variable)
- a_t : System action (random variable)
- y_t : Click (random variable)
- 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:

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

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

¹Otherwise called influence diagrams

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 $\{\pi(a) | \pi \in \Pi\}$, one for each possible value π . The following notations will be equivalent:

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

The reader should note that the standard definition of a conditional distribution also $P(A | B)$ 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

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

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

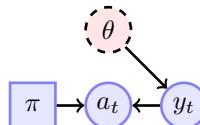


Figure 6.5: π does not cause y

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

EXAMPLE 42. Consider the model

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

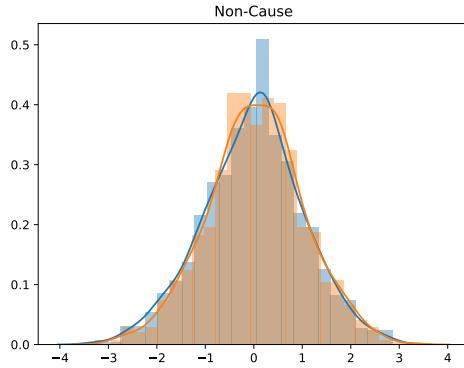


Figure 6.6: $\mathbb{P}^\pi(y_t)$ for $\pi \in \{-1, 1\}$ when π is not a cause for y_t

In this example, we see that 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 $y_t \perp\!\!\!\perp \pi | a_t$, as captured by the diagram in Figure 6.7. In this case π is a direct cause for y_t through a_t .

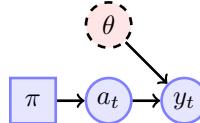


Figure 6.7: No confounding: π causes y_t

EXAMPLE 43. Consider the model

$$\begin{aligned} a_t &\sim \mathcal{N}(\pi, 1) \\ y_t | a_t, \pi &\sim \mathcal{N}(a_t, 1) \end{aligned}$$

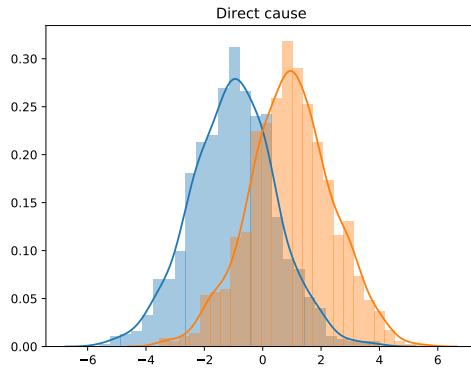


Figure 6.8: $\mathbb{P}^\pi(y_t)$ for $\pi \in \{-1, 1\}$ when π is a direct cause for y_t

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.

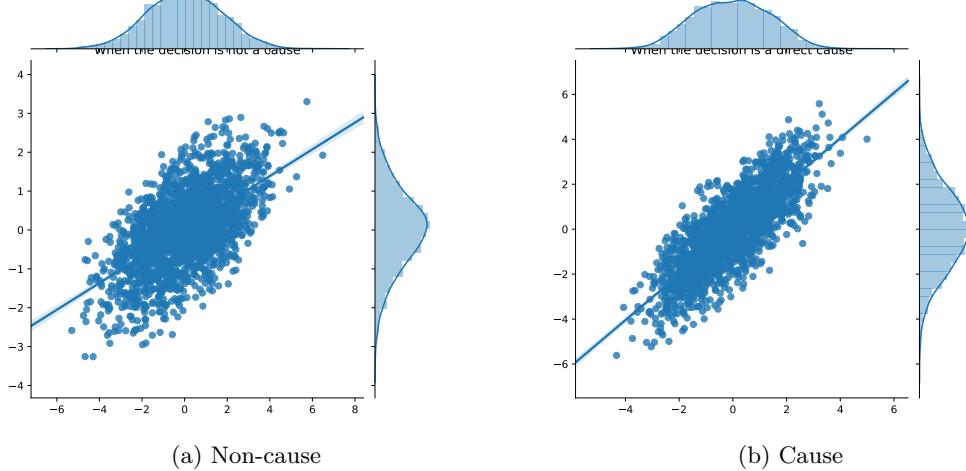
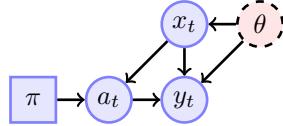


Figure 6.9: Correlation between a_t and y_t

Covariates

Sufficient covariate

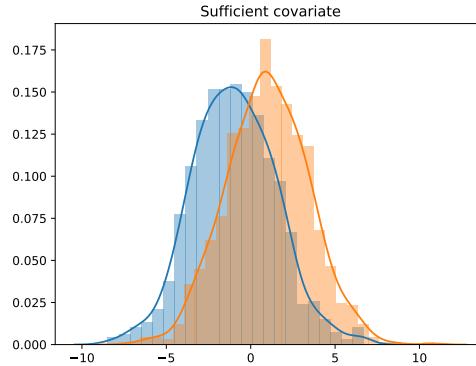
Sometimes the variable of interest is not conditionally independent of the treatment, unless there exists a *sufficient covariate* x_t such that $y_t \perp\!\!\!\perp \pi | a_t, x_t$. If x_t is not observed, then it is sometimes called a confounder.

Figure 6.10: Sufficient covariate x_t

EXAMPLE 44. Consider the model

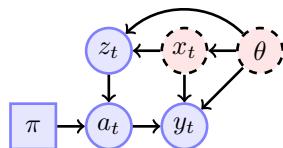
$$\begin{aligned} x_t &\sim \mathcal{N}(0, 1) \\ a_t &\sim \mathcal{N}(x_t + \pi, 1) \\ y_t | a_t, \pi &\sim \mathcal{N}(x_t + a_t, 1), \end{aligned}$$

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

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

Figure 6.12: Instrumental variable z_t

EXAMPLE 45. 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

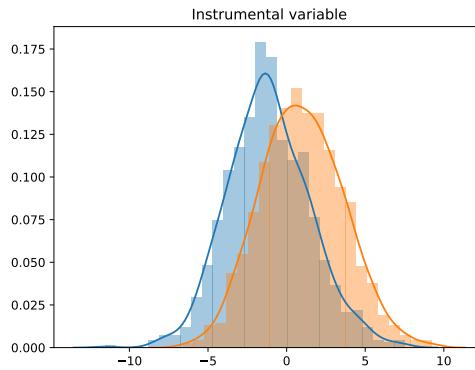


Figure 6.13: $\mathbb{P}^\pi(y_t)$ for $\pi \in \{-1, 1\}$ when π is a direct cause for y_t

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.
- 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 .

²Hence, it can be called a confounder.

Intervention policies

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

- Direct interventions. The simplest scenario is when we are able to choose a π for which we know $\pi(a_t | x_t)$. 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 $\pi(a_t | x_t)$ 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 $\pi(a_t | x_t) = \sum_{z_t} \pi(a_t | z_t, x_t) \pi(z_t | x_t)$. In this scenario we can freely specify $\pi(z_t | x_t)$, but $\pi(a_t | z_t, x_t)$ 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 46 (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 by 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.

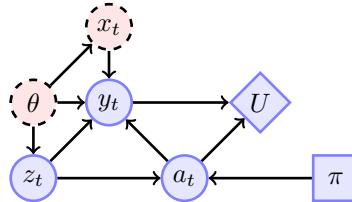


Figure 6.14: Model of non-compliance as a confounder.

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.

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

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

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:

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

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

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

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

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

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 sampling³

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

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

Bayesian estimation

Unfortunately 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 ξ on Θ and obtain the following, assuming the policy π_0 is stationary.

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

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.

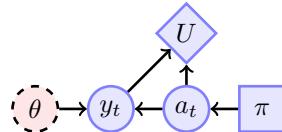


Figure 6.15: Simple decision problem.

EXAMPLE 47. Let $a_t, y_t \in \{0, 1\}$, $\theta \in [0, 1]^2$ and

$$y_t | a_t = a \sim \text{Bernoulli}(\theta_a)$$

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 $Beta(\alpha_a^0, \beta_b^0)$ for each one of the Bernoulli parameters, so that the posterior after t observations is

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

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.

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

to obtain:

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

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

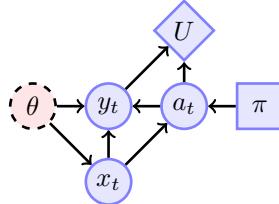


Figure 6.16: Decision problem with covariates.

EXAMPLE 48. Let $a_t, x_t \in \{0, 1\}$, $y_t \in \mathbb{R}$, $\theta \in \mathbb{R}^4$ and

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

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.

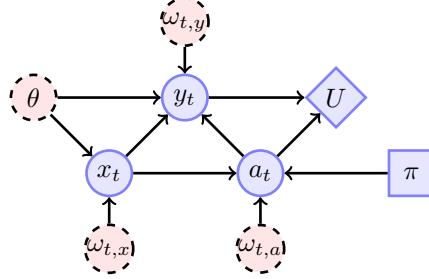


Figure 6.17: Decision diagram with exogenous disturbances ω .

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 $\Omega \times \Theta \rightarrow \mathcal{X}$. 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 49 (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.

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

Structural equation models are particularly interesting in applications such as economics, where there are postulated relations between various economic quantities.

Treatment-unit additivity

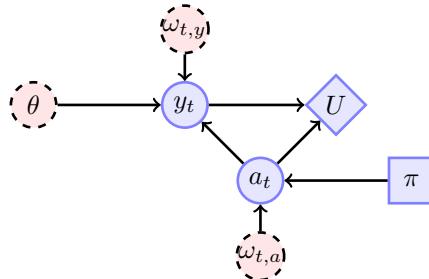


Figure 6.18: Decision diagram for treatment-unit additivity

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

$$\mathbb{E}[y_t | a_t = a] = g(a_t) + \mathbb{E}(\omega_{t,y})$$

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 50 (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.

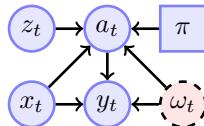


Figure 6.19: Graph of structural equation model for airport pricing policy π : a_t is the actual price, z_t are fuel costs, x_t is the customer type, y_t is the amount of sales, ω_t is whether there is a conference. The dependency on θ is omitted for clarity.

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). $z_t \perp\!\!\!\perp y_t | x_t, a_t, \omega_t$.

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). $z_t \perp\!\!\!\perp \omega_t | x_t$.

Prediction tasks

We can use the following structural equation model

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

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.

$$\mathbb{P}_\theta^\pi(y_t | x_t, a_t), \quad \mathbb{E}_\theta^\pi(y_t | x_t, a_t) = g_\theta(x_t, a_t) + \mathbb{E}_\theta^\pi(\omega_t | x_t, a_t).$$

Counterfactual prediction

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

6.4.3 Discussion**Further reading**

- Pearl, *Causality*.
- Dawid⁸

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 15. 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

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

where the actions are distribution according to $\pi(a_t)$.

- Assume that $\pi_0 \in [0, 1]$ is given and it is $a_t | \pi = \pi_0 \sim \text{Bernoulli}(\pi_0)$. First, estimate θ . Then, calculate the distribution of $y_t | \pi_0, \theta$ 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 16. 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 $\theta \in \mathbb{R}^2$.

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

where the actions are distribution according to $\pi(a_t)$.

- Assume that $\pi_0 \in [0, 1]$ is given and it is $a_t | \pi = \pi_0 \sim \text{Bernoulli}(\pi_0)$. First, estimate θ . Then, calculate the distribution of $y_t | \pi_0, \theta$ 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 17. 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 18 (Advanced). Let us have some data generated from a null treatment policy π_0 of the form (x_t, a_t, y_t) , with $a_t, x_t \in \{0, 1\}$.

$$y_t | a_t = a, x_t = x, \theta \sim \mathcal{N}(\theta_{a,x}, 1),$$

where the actions are distribution according to $\pi_0(a_t | x_t)$.

- Assume that π_0 is given and it is $a_t | x_t = x, \pi = \pi_0 \sim \text{Bernoulli}(w_x)$. 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

Bandit problems and 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.

We begin 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 [1], informally can be stated as follows.

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.

EXAMPLE 51. 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 $a_t \in \{1, \dots, k\}$. We then 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.

EXAMPLE 52. An alternative goal would be to do a *clinical trial*, 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 ξ_0 on Ω . We can perform T experiments, after which we must make a decision 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 $U : \mathcal{A} \times \Omega \rightarrow \mathbb{R}$, which models the effect of each decision d 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 $a_t \in \{1, \dots, k\}$ and observe outcome $x_t \in \mathcal{X}$, with $x_t \sim P_\omega$ drawn from the true hypothesis. Our posterior is

$$\xi_t(\omega) \triangleq \xi_0(\omega | a_1, \dots, a_t, x_1, \dots, x_t).$$

The reward is $r_t = 0$ for $t < T$ and

$$r_T = \max_{a \in \mathcal{A}} \mathbb{E}_{\xi_T}(U | a).$$

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) $a_t \in \mathcal{A} = \{1, \dots, n\}$. In this setting, each time t you play a machine, you receive a reward r_t , with fixed expected value $\omega_i = \mathbb{E}(r_t | a_t = i)$. 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 $\mathcal{A} = \{1, \dots, n\}$, 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 $\mathbb{E}(r_t | a_t = i) = \omega_i$.

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 $\pi : \mathcal{A}^* \times \mathbb{R}^* \rightarrow \mathcal{A}$ is a deterministic mapping from the space of all sequences of actions and rewards to actions. That is, for every observation and action history $a_1, r_1, \dots, a_{t-1}, r_{t-1}$ 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,

$$\pi(a_t | a^{t-1}, r^{t-1}) \quad (7.2.1)$$

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 our utility from time t to be:

$$U_t = \sum_{k=1}^{T-t} r_{t+k}. \quad (7.2.2)$$

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 ξ 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 ξ . 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:

$$\max_{\pi} \mathbb{E}_{\xi}^{\pi} U_t = \max_{\pi} \int_{\Omega} \mathbb{E}_{\omega}^{\pi} U_t d\xi \omega. \quad (7.2.3)$$

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 $\mathcal{F} = \{P_{\omega,i} | \omega \in \Omega, i \in \mathcal{A}\}$ on \mathbb{R} .

- Assume the i.i.d model $r_t | \omega, a_t = i \sim P_{\omega,i}$.
- Define prior ξ on Ω .
- Select a policy $\pi : \mathcal{A}^* \times \mathbb{R}^* \rightarrow \mathcal{A}$ maximising

$$\mathbb{E}_\xi^\pi U = \mathbb{E}_\xi^\pi \sum_{t=0}^{T-1} r_t$$

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 ($i = 1, \dots, n$) such that

$$r_t | a_t = i \sim \text{Bernoulli}(\omega_i), \quad \mathbb{E}(r_t | a_t = i) = \omega_i. \quad (7.2.4)$$

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 $\text{Beta}(\alpha_i, \beta_i)$, with density $f(\omega | \alpha_i, \beta_i)$ so that

$$\xi(\omega_1, \dots, \omega_n) = \prod_{i=1}^n f(\omega_i | \alpha_i, \beta_i).$$

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

$$\hat{r}_{t,i} \triangleq \frac{1}{N_{t,i}} \sum_{k=1}^t r_k \mathbb{I}\{a_k = i\}$$

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

$$\xi_t = \text{Beta}(\alpha_i + N_{t,i} \hat{r}_{t,i}, \beta_i + N_{t,i} (1 - \hat{r}_{t,i})).$$

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 ξ_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

$$N_t = (N_{t,1}, \dots, N_{t,i}) \quad (7.2.5)$$

$$\hat{r}_t = (\hat{r}_{t,1}, \dots, \hat{r}_{t,i}). \quad (7.2.6)$$

At any time t , we can calculate the probability of observing $r_t = 1$ or $r_t = 0$ if we pull arm i as:

$$\xi_t(r_t = 1 | a_t = i) = \frac{\alpha_i + N_{t,i}\hat{r}_{t,i}}{\alpha_i + \beta_i + N_{t,i}}$$

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 8.1.1). 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.

$$\mathcal{F} = \{P_i | i = 1, \dots, n\}.$$

At each time-step t we select an action a_t , obtaining a random reward distributed according to:

$$r_t | a_t = i \sim P_i.$$

Our objective is to find a policy π maximising the expected total reward.

$$\mathbb{E}_\pi U_t = \mathbb{E}_\pi \sum_{k=t}^T r_k, \quad a_t^* \triangleq \max \{\mathbb{E}(r_t | a_t = i) | i = 1, \dots, n\}.$$

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.

$$\mathcal{F} = \{P_i(\cdot | \omega) | \omega \in \Omega\}, \quad r_t | a_t = i, \omega^* = \omega \sim P_i(r | \omega^*). \quad (7.2.7)$$

If in addition we have a subjective belief ξ over Ω , we could (as explained in Sec. 7.2) in principle calculate the policy maximising the ξ -expected utility:

$$\mathbb{E}_\xi^\pi U_t = \mathbb{E}_\xi^\pi \sum_{k=t}^T r_k. \quad (7.2.8)$$

This of course will now have to be a history-dependent policy. In the remainder of this section, we shall examine algorithms which eventually converge 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 algorithm⁷ 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  $t = 1, \dots, T$  do
3:   Take action  $a_t = i$  with probability  $\pi(i | a_1, \dots, a_{t-1}, r_1, \dots, r_{t-1})$ .
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  $\mu_T$ 

```

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

$$\hat{\pi}_\epsilon^* \triangleq (1 - \epsilon_t)\hat{\pi}_t^* + \epsilon_t \text{Unif}(\mathcal{A}), \quad (7.2.9)$$

$$\hat{\pi}_t^*(i) = \mathbb{I}\left\{i \in \hat{\mathcal{A}}_t^*\right\} / |\hat{\mathcal{A}}_t^*|, \quad \hat{\mathcal{A}}_t^* = \arg \max_{i \in \mathcal{A}} \mu_{t,i} \quad (7.2.10)$$

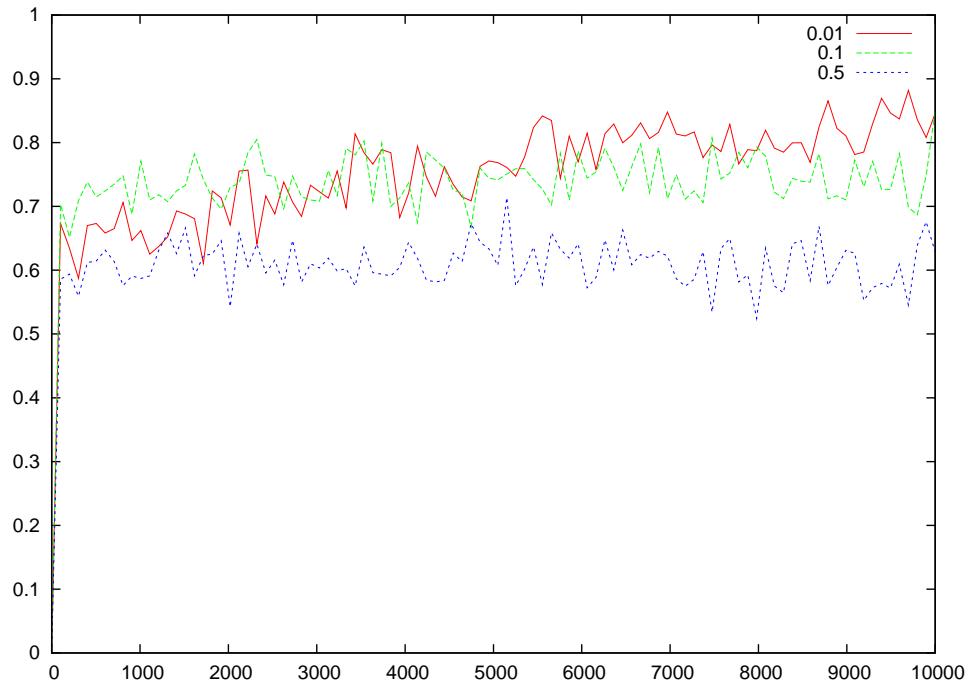
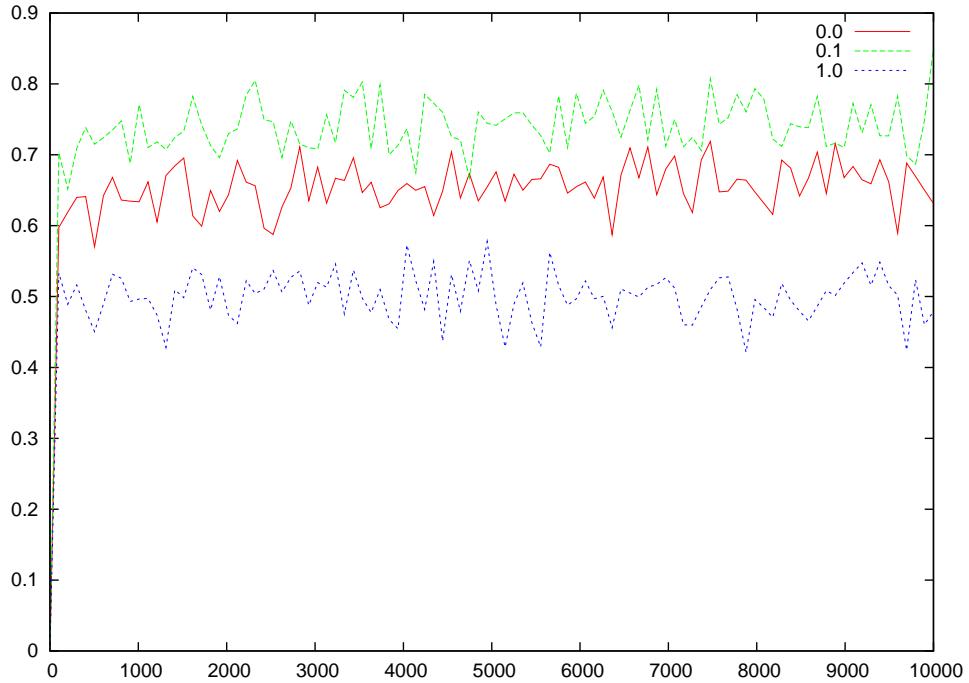


Figure 7.1: $\epsilon_t = 0.1$, $\eta \in \{0.01, 0.1, 0.5\}$.

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 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:

$$P_{\omega,a}(R) \triangleq \mathbb{P}_{\omega}(r_t \in R \mid a_t = a), \quad R \subseteq \mathbb{R}. \quad (7.2.11)$$

Let ξ_1 be a prior distribution on Ω and let the posterior distributions be defined as:

$$\xi_{t+1}(B) \propto \int_B P_{\omega,a_t}(r_t) d\xi_t(\omega). \quad (7.2.12)$$

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:

$$P_{\xi_t, a}(R) \triangleq \int_{\Omega} P_{\omega, a}(R) d\xi_t(\omega). \quad (7.2.13)$$

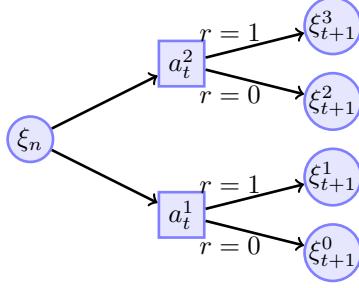


Figure 7.3: A partial view of the multi-stage process. Here, the probability that we obtain $r = 1$ if we take action $a_t = i$ is simply $P_{\xi_t, i}(\{1\})$.

Finally, as ξ_{t+1} deterministically depends on ξ_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:

$$\mathbb{P}(\xi_{t+1} = \xi \mid \xi_t, a_t) = \int_{\mathcal{R}} \mathbb{I}\{\xi_t(\cdot \mid a_t, r_t = r) = \xi\} dP_{\xi_t, a}(r). \quad (7.2.14)$$

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

$$U^*(\xi_t) = \max_{a_t} \mathbb{E}(r_t \mid \xi_t, a_t) + \sum_{\xi_{t+1}} \mathbb{P}(\xi_{t+1} \mid \xi_t, a_t) U^*(\xi_{t+1}). \quad (7.2.15)$$

The above equation is the *backwards induction* algorithm for bandits. If you look at this *backwards induction* 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

¹In fact, a decision-theoretic bandit process is a specific case of a Markov decision process

in the following observation

$$\mathbb{E}_{\xi_t}^{\pi}(U_t) = \mathbb{E}_{\xi_t}^{\pi} \left(\sum_{k=t}^T r_k \right) \quad (7.2.16)$$

$$= \mathbb{E}_{\xi_t}^{\pi} \left(r_t + \sum_{k=t+1}^T r_k \right) \quad (7.2.17)$$

$$= \mathbb{E}_{\xi_t}^{\pi}(r_t) + \mathbb{E}_{\xi_t}^{\pi}(U_{t+1}) \quad (7.2.18)$$

$$= \mathbb{E}_{\xi_t}^{\pi}(r_t) + \int_{\Xi} \mathbb{E}_{\xi_{t+1}}^{\pi}(U_{t+1}) d\mathbb{P}(\xi_{t+1} | \xi_t, \pi) \quad (7.2.19)$$

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:

$$\max \mathbb{E}_{\xi_t}^{\pi}(U_t) = \max_{a_t} \mathbb{E}_{\xi_t}(r_t | a_t) + \int_{\Xi} \max_{\pi'} \mathbb{E}_{\xi_{t+1}}^{\pi'}(U_{t+1}) d\mathbb{P}(\xi_{t+1} | \xi_t, a_t) \quad (7.2.20)$$

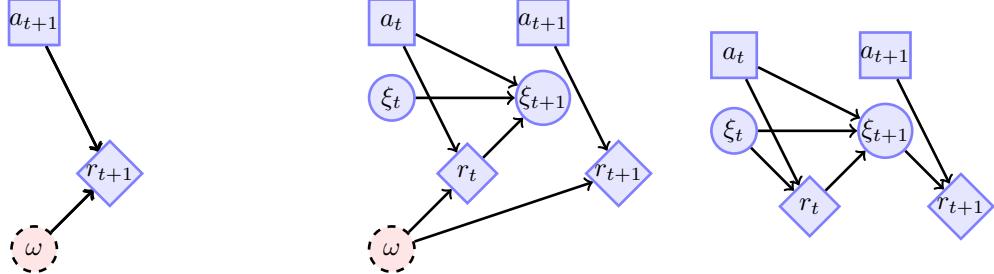


Figure 7.4: Three views of the bandit process. Figure ?? shows the basic bandit process, from the view of an external observer. The decision maker selects a_t , while the parameter ω of the process is hidden. It then obtains reward r_t . The process repeats for $t = 1, \dots, T$. The decision-theoretic bandit process is shown in Figures ?? and ???. While ω is not known, at each time step t we maintain a belief ξ_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.

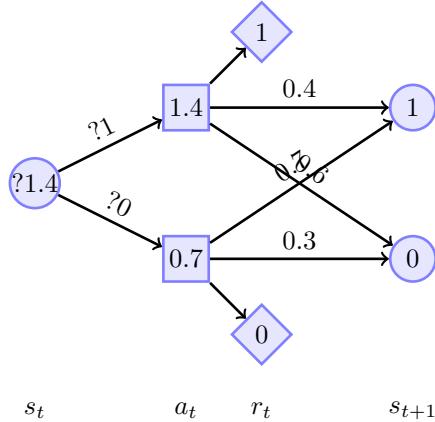
Illustration of backwards induction

Backwards induction (Dynamic programming)

```
for  $t = 1, 2, \dots$  and  $\xi_t \in \Xi_t$  do
```

$$U_t^*(\xi_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}(r_t | \xi_t, a_t) + \sum_{\xi_{t+1} \in \Xi_{t+1}} \mathbb{P}(\xi_{t+1} | \xi_t, a_t) U_{t+1}^*(\xi_{t+1})$$

```
end for
```



EXERCISE 19. 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 they 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 oracle² 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

²Also called the regret of the algorithm

of each arm, and simply picks the arm with the highest upper confidence interval, as shown in Algorithm 4.

Algorithm 4 UCB1

```

Input  $\mathcal{A}$ 
 $\hat{\theta}_{0,i} = 1, \forall i$ 
for  $t = 1, \dots$  do
     $a_t = \arg \max_{i \in \mathcal{A}} \left\{ \hat{\theta}_{t-1,i} + \sqrt{\frac{2 \ln t}{N_{t-1,i}}} \right\}$ 
     $r_t \sim P_\theta(r | a_t)$  // play action and get reward
    /* update model */
     $N_{t,a_t} = N_{t-1,a_t} + 1$ 
     $\hat{\theta}_{t,a_t} = [N_{t-1,a_t} \hat{\theta}_{t-1,a_t} + r_t] / N_{t,a_t}$ 
     $\forall i \neq a_t, N_{t,i} = N_{t-1,i}, \hat{\theta}_{t,i} = \hat{\theta}_{t-1,i}$ 
end for

```

The Thompson sampling algorithm

In the Bayesian setting, whenever we can define some prior belief ξ_0 over parameters Θ , we can use Thompson sampling, first introduced by Thompson²². 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  $\mathcal{A}, \xi_0$ 
for  $t = 1, \dots$  do
     $\hat{\theta} \sim \xi_{t-1}(\theta)$ 
     $a_t \in \arg \max_a \mathbb{E}_{\hat{\theta}}[r_t | a_t = a]$ .
     $r_t \sim P_\theta(r | a_t)$  // play action and get reward
     $\xi_t(\theta) = \xi_{t-1}(\theta | a_t, r_t)$ . // 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 53 (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 $r_t | a_t = a, x_t = x \sim P_\theta(r | a, x)$.

EXAMPLE 54 (The linear bandit problem). • $\mathcal{A} = [n], \mathcal{X} = \mathbb{R}^k, \theta = (\theta_1, \dots, \theta_n), \theta_i \in \mathbb{R}^k, r \in \mathbb{R}$.

- $r \sim \mathcal{N}(\theta_a^\top x), 1)$

EXAMPLE 55 (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.

- $\mathcal{A} = [n]$, $\mathcal{X} = \mathbb{R}^k$, $\theta = (\theta_1, \dots, \theta_n)$, $\theta_i \in \mathbb{R}^k$, $y \in \{0, 1\}$.
- $y \sim \text{Bernoulli}(1/(1 + \exp[-(\theta_a^\top x)^2]))$.
- $r = U(a, y)$.

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  $\mathcal{A}, \xi_0$ 
for  $t = 1, \dots$  do
     $\hat{\theta} \sim \xi_{t-1}(\theta)$ 
    Observe  $x_t$ .
     $a_t \in \arg \max_a \mathbb{E}_{\hat{\theta}}[r_t | x_t, a_t = a]$ .
     $r_t \sim P_\theta(r | a_t)$  // play action and get reward // update model
     $\xi_t(\theta) = \xi_{t-1}(\theta | a_t, r_t, x_t)$ .
end for

```

We can also consider the full decision theoretic solution to Thompson sampling

Backwards induction in the contextual setting

for $n = 1, 2, \dots$ and $s \in \mathcal{S}$ **do**

$$\mathbb{E}(U_t | x_t, \xi_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}(r_t | x_t, \xi_t, a_t) + \sum_{\xi_{t+1}} \mathbb{P}(\xi_{t+1} | \xi_t, x_t, a_t) \mathbb{E}(U_{t+1} | \xi_{t+1})$$

end for

As ξ_{t+1} is a deterministic function of ξ_t, x_t, a_t , we can simply replace the sum in the right hand side as follows:

$$\mathbb{E}(U_{t+1} | x_t, a_t) = \sum_{\xi_{t+1}} \mathbb{P}(\xi_{t+1} | \xi_t, x_t, a_t) \mathbb{E}(U_{t+1} | \xi_{t+1}) \quad (7.3.1)$$

$$= \sum_{r_t} \mathbb{P}(r_t | \xi_t, x_t, a_t) \mathbb{E}[U_{t+1} | \xi_t(\cdot | r_t, x_t, a_t)] \quad (7.3.2)$$

$$(7.3.3)$$

where $\mathbb{P}(r_t | \xi_t, x_t, a_t) = \int_{\Theta} P_\theta(r_t | x_t, a_t) d\xi_t(\theta)$ 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 56 (One-stage problems). In a classical one-stage experiment design problem, the experimenter only has a single observation

- Initial belief ξ_0
- Side information \mathbf{x}
- Simultaneously takes actions \mathbf{a} .
- Observes outcomes \mathbf{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. $\mathbf{x} = \mathbf{x}_{1:k}$. 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.

$$\mathbb{E}_{\xi_0}^{\pi}(U | \mathbf{x}) = \sum_{\mathbf{a}, \mathbf{y}} \mathbb{P}_{\xi_0}(\mathbf{y} | \mathbf{a}, \mathbf{x}) \pi(\mathbf{a} | \mathbf{x}) \underbrace{\mathbb{E}_{\xi_0}^{\pi}(U | \mathbf{x}, \mathbf{a}, \mathbf{y})}_{\text{post-hoc value}} \quad (7.4.1)$$

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).

$$\mathbb{E}_{\xi_0}^{\pi}(\mathbb{D}(\xi_1 \| \xi_0) | \mathbf{x}) = \sum_{\mathbf{a}, \mathbf{y}} \mathbb{P}_{\xi_0}(\mathbf{y} | \mathbf{a}, \mathbf{x}) \pi(\mathbf{a} | \mathbf{x}) \mathbb{D}(\xi_0(\cdot | \mathbf{x}, \mathbf{a}, \mathbf{y}) \| \xi_0) \quad (7.4.2)$$

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:

$$\mathbb{E}_{\xi_0}^{\pi} \left(\max_{\pi_1} \mathbb{E}_{\xi_1}^{\pi_1} \rho | \mathbf{x} \right) = \sum_{\mathbf{a}, \mathbf{y}} \mathbb{P}_{\xi_0}(\mathbf{y} | \mathbf{a}, \mathbf{x}) \pi(\mathbf{a} | \mathbf{x}) \max_{\pi_1} \mathbb{E}_{\xi_0}^{\pi_1}(\rho | \mathbf{a}, \mathbf{x}, \mathbf{y}) \quad (7.4.3)$$

$$\mathbb{E}_{\xi_0}^{\pi_1}(\rho | \mathbf{a}, \mathbf{x}, \mathbf{y}) = \sum_{a, x, y} \rho(a, y) \mathbb{P}_{\xi_1}(y | x, a) \pi_1(a | x) \mathbb{P}_{\xi_1}(x) \quad (7.4.4)$$

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 $\mathbb{E}_{\xi_t}^{\pi_t} \sum_{k=t}^{t+n} r_t$. 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 sampling¹² or MCMC.

Chapter 8

Markov decision processes

8.1 Markov decision processes and reinforcement learning

Bandit problems are one of the simplest instances of reinforcement learning problems. Informally, speaking, these are problems of learning how to act in an unknown environment, only through interaction with the environment and limited reinforcement signals. The learning agent interacts with the environment through actions and observations, and simultaneously obtains rewards. For example, we can consider a rat running through a maze designed by an experimenter, which from time to time finds a piece of cheese, the reward. The goal of the agent is usually to maximise some measure of the total reward. In summary, we can state the problem as follows.

The reinforcement learning problem.

The reinforcement learning problem is the problem of *learning* how to act in an *unknown* environment, only by **interaction** and **reinforcement**.

Generally, we assume that the environment μ that we are acting in has an underlying state $s_t \in \mathcal{S}$, which changes with in discrete time steps t . At each step, the agent obtains an observation $x_t \in \mathcal{X}$ and chooses actions $a_t \in \mathcal{A}$. We usually assume that the environment is such that its next state s_{t+1} only depends on its current state s_t and the last action taken by the agent, a_t . In addition, the agent observes a reward signal r_t , and its goal is to maximise the total reward during its lifetime.

Doing so when the environment μ is unknown, is hard even in seemingly simple settings, like n -armed bandits, where the underlying state never changes. In many real-world applications, the problem is even harder, as the state is not directly observed. Instead, we may simply have some measurements x_t , which give only partial information about the true underlying state s_t .

Reinforcement learning problems typically fall into one of the following three groups: (1) Markov decision processes (MDPs), where the state s_t is observed directly, i.e. $x_t = s_t$; (2) Partially observable MDPs (POMDPs), where the state is hidden, i.e. x_t is only probabilistically dependent on the state; and (3) stochastic Markov games, where the next state also depends on the move of other agents. While all of these problem *descriptions* are different, in the Bayesian setting, they all can be reformulated as MDPs, by constructing an appropriate belief state, similarly to how we did it for the decision theoretic formulation of the bandit problem.

In this chapter, we shall restrict our attention to Markov decision processes. Hence, we shall not discuss the existence of other agents, or the case where we cannot observe the state directly.

Definition 8.1.1 (Markov Decision Process). A Markov decision process μ is a tuple $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where \mathcal{S} is the *state space* and \mathcal{A} is the *action space*. The *transition distribution* being $\mathcal{P} = \{P(\cdot | s, a) | s \in \mathcal{S}, a \in \mathcal{A}\}$ is a collection of probability measures on \mathcal{S} , indexed in $\mathcal{S} \times \mathcal{A}$ and the *reward distribution* $\mathcal{R} = \{\rho(\cdot | s, a) | s \in \mathcal{S}, a \in \mathcal{A}\}$ is a collection of probability measures on \mathbb{R} , such that:

$$P(S | s, a) = \mathbb{P}_\mu(s_{t+1} \in S | s_t = s, a_t = a) \quad (8.1.1)$$

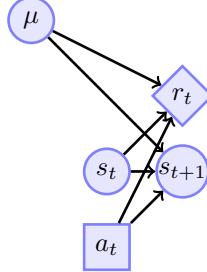
$$\rho(R | s, a) = \mathbb{P}_\mu(r_t \in R | s_t = s, a_t = a). \quad (8.1.2)$$

For simplicity, we shall also use

$$r_\mu(s, a) = \mathbb{E}_\mu(r_{t+1} | s_t = s, a_t = a), \quad (8.1.3)$$

for the expected reward.

Of course, the transition and reward distributions are different for different environments μ . For that reason, we shall usually subscript the relevant probabilities and expectations with μ , unless the MDP is clear from the context.



Markov property of the reward and state distribution

$$\mathbb{P}_\mu(s_{t+1} \in S \mid s_1, a_1, \dots, s_t, a_t) = \mathbb{P}_\mu(s_{t+1} \in S \mid s_t, a_t) \quad (8.1.4)$$

$$\mathbb{P}_\mu(r_t \in R \mid s_1, a_1, \dots, s_t, a_t) = \mathbb{P}_\mu(r_t \in R \mid s_t, a_t) \quad (8.1.5)$$

where $S \subset \mathcal{S}$ and $R \subset \mathcal{R}$ are reward and state subsets respectively.

Figure 8.1: The structure of a Markov decision process.

Dependencies of rewards. Sometimes it is more convenient to have rewards that depend on the next state as well, i.e.

$$r_\mu(s, a, s') = \mathbb{E}_\mu(r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'), \quad (8.1.6)$$

though this is complicates the notation considerably since now the reward is obtained on the next time step. However, we can always replace this with the expected reward for a given state-action pair:

$$r_\mu(s, a) = \mathbb{E}_\mu(r_{t+1} \mid s_t = s, a_t = s) = \sum_{s' \in \mathcal{S}} P_\mu(s' \mid s, a) r_\mu(s, a, s') \quad (8.1.7)$$

In fact, it is notationally more convenient to have rewards that only depend on the current state:

$$r_\mu(s) = \mathbb{E}_\mu(r_t \mid s_t = s). \quad (8.1.8)$$

For simplicity, we shall mainly consider the latter case.

The agent. The environment does not exist in isolation. The actions are taken by an agent, who is interested in obtaining high rewards. Instead of defining an algorithm for choosing actions directly, we define an algorithm for computing policies, which define distributions on actions.

The agent's policy π

$$\begin{array}{ll} \mathbb{P}^\pi(a_t | s_t, \dots, s_1, a_{t-1}, \dots, a_1) & \text{(history-dependent policy)} \\ \mathbb{P}^\pi(a_t | s_t) & \text{(Markov policy)} \end{array}$$

In some sense, the agent is defined by its *policy* π , which is a conditional distribution on actions given the history. The *policy* π is otherwise known as a *decision function*. In general, the policy can be history-dependent. In certain cases, however, there are optimal policies that are Markov. This is for example the case with additive utility functions. In particular, the utility function maps from the sequence of all possible rewards to a real number $U : \mathcal{R}^* \rightarrow \mathbb{R}$, given below:

Definition 8.1.2 (Utility). Given a horizon T and a discount factor $\gamma \in (0, 1]$, the utility function $U : \mathcal{R}^* \rightarrow \mathbb{R}$ is defined as

$$U(r_0, r_1, \dots, r_T) = \sum_{k=0}^T \gamma^k r_k. \quad (8.1.9)$$

It is convenient to give a special name to the utility starting from time t , i.e. the sum of rewards from that time on:

$$U_t \triangleq \sum_{k=0}^{T-t} \gamma^k r_{t+k}. \quad (8.1.10)$$

At any time t , the agent wants to find a policy π *maximising* the *expected total future reward*

$$\mathbb{E}_\mu^\pi U_t = \mathbb{E}_\mu^\pi \sum_{k=0}^{T-t} \gamma^k r_{t+k}. \quad (\text{expected utility})$$

This is so far identical to the expected utility framework we had seen so far, with the only difference that now the reward space is a sequence of numerical rewards and that we are acting within a dynamical system with state space \mathcal{S} . In fact, it is a good idea to think about the *value* of different states of the system under certain policies, in the same way that one thinks about how good different positions are in chess.

8.1.1 Value functions

A value function represents the expected utility of a given state, or state-and-action pair for a specific policy. They are really useful as shorthand notation and as the basis of algorithm development. The most basic of those is the state value function.

State value function

$$V_{\mu,t}^\pi(s) \triangleq \mathbb{E}_\mu^\pi(U_t | s_t = s) \quad (8.1.11)$$

The state value function for a particular policy π can be interpreted as how much utility you should expect if you follow the policy starting from state s at time t , for the particular MDP μ .

State-action value function

$$Q_{\mu,t}^{\pi}(s, a) \triangleq \mathbb{E}_{\mu}^{\pi}(U_t \mid s_t = s, a_t = a) \quad (8.1.12)$$

The state-action value function for a particular policy π can be interpreted as how much utility you should expect if you play action a , at state s at time t , and then follow the policy π , for the particular MDP μ .

It is also useful to define the optimal policy and optimal value functions for a given MDP. In the following, a star indicates optimal quantities. The *optimal policy* π^*

$$\pi^*(\mu) : V_{t,\mu}^{\pi^*(\mu)}(s) \geq V_{t,\mu}^{\pi}(s) \quad \forall \pi, t, s \quad (8.1.13)$$

dominates all other policies π everywhere in \mathcal{S} .

The *optimal value function* V^*

$$V_{t,\mu}^*(s) \triangleq V_{t,\mu}^{\pi^*(\mu)}(s), \quad Q_{t,\mu}^*(s) \triangleq Q_{t,\mu}^{\pi^*(\mu)}(s, a). \quad (8.1.14)$$

is the value function of the optimal policy π^* .

Finding the optimal policy when μ is known

When the MDP μ is known, the expected utility of any policy can be calculated. Therefore, one could find the optimal policy by brute force, i.e. by calculating the utility of every possible policy. This might be a reasonable strategy if the number of policies is small. However, there are many better approaches. First, there are iterative/offline methods where an optimal policy is found for all states of the MDP. These either try to estimate the optimal value function directly, or try to iteratively improve a policy until it is optimal. The second type of methods tries to find an optimal policy online. That is, the optimal actions are estimated only for states which can be visited in the future starting from the current state. However, the same main ideas are used in all of these algorithms.

8.2 Finite horizon, undiscounted problems

The conceptually simplest type of problems are finite horizon problems where $T < \infty$ and $\gamma = 1$. The first thing we shall try to do is to evaluate a given policy for a given MDP. There are a number of algorithms that can achieve this.

8.2.1 Policy evaluation

Here we are interested in the problem of determining the value function of a policy π (for $\gamma = 1, T < \infty$). All the algorithms we shall consider can be recovered from the following

recursion. Noting that $U_{t+1} = \sum_{k=1}^{T-t} r_{t+k}$ we have:

$$V_{\mu,t}^{\pi}(s) \triangleq \mathbb{E}_{\mu}^{\pi}(U_t | s_t = s) \quad (8.2.1)$$

$$= \sum_{k=0}^{T-t} \mathbb{E}_{\mu}^{\pi}(r_{t+k} | s_t = s) \quad (8.2.2)$$

$$= \mathbb{E}_{\mu}^{\pi}(r_t | s_t = s) + \mathbb{E}_{\mu}^{\pi}(U_{t+1} | s_t = s) \quad (8.2.3)$$

$$= \mathbb{E}_{\mu}^{\pi}(r_t | s_t = s) + \sum_{i \in \mathcal{S}} V_{\mu,t+1}^{\pi}(i) \mathbb{P}_{\mu}^{\pi}(s_{t+1} = i | s_t = s). \quad (8.2.4)$$

Note that the last term can be calculated easily through marginalisation.

$$\mathbb{P}_{\mu}^{\pi}(s_{t+1} = i | s_t = s) = \sum_{a \in \mathcal{A}} \mathbb{P}_{\mu}(s_{t+1} = i | s_t = s, a_t = a) \mathbb{P}^{\pi}(a_t = a | s_t = s).$$

This derivation directly gives a number of *policy evaluation algorithms*.

Direct policy evaluation Direct policy evaluation is based on (8.2.2), which can be implemented by Algorithm 7. One needs to *marginalise out* all possible state sequences to obtain the expected reward given the state at time $t+k$ giving the following:

$$\mathbb{E}_{\mu}^{\pi}(r_{t+k} | s_t = s) = \sum_{s_{t+1}, \dots, s_{t+k} \in \mathcal{S}^k} \mathbb{E}_{\mu}^{\pi}(r_{t+k} | s_{t+k}) \mathbb{P}_{\mu}^{\pi}(s_{t+1}, \dots, s_{t+k} | s_t).$$

By using the Markov property, we calculate the probability of reaching any state from any other state at different times, and then add up the expected reward we would get in that state under our policy. Then $\hat{V}_t(s) = V_{\mu,t}^{\pi}(s)$ by definition.

Unfortunately it is not a very good idea to use direct policy evaluation. The most efficient implementation involves calculating $P(s_t | s_0)$ recursively for every state. This would result in a total of $|\mathcal{S}|^3 T$ operations. Monte-Carlo evaluations should be considerably cheaper, especially when the transition structure is sparse.

Algorithm 7 Direct policy evaluation

```

1: for  $s \in \mathcal{S}$  do
2:   for  $t = 0, \dots, T$  do
3:      $\hat{V}_t(s) = \sum_{k=t}^T \sum_{j \in \mathcal{S}} \mathbb{P}_{\mu}^{\pi}(s_k = j | s_t = s) \mathbb{E}_{\mu}^{\pi}(r_k | s_k = j).$ 
4:   end for
5: end for

```

8.2.2 Monte-Carlo policy evaluation

Another conceptually simple algorithm is Monte-Carlo policy evaluation shown as Algorithm 8. The idea is that instead of summing over all possible states to be visited, we just draw states from the Markov chain defined jointly by the policy and the Markov decision process. Unlike direct policy evaluation the algorithm needs a parameter K , the number of trajectories to generate. Nevertheless, this is a very useful method, employed within a number of more complex algorithms.

Algorithm 8 Monte-Carlo policy evaluation

```

for  $s \in \mathcal{S}$  do
  for  $k = 0, \dots, K$  do
    Choose initial state  $s_1$ .
    for  $t = 1, \dots, T$  do
       $a_t \sim \pi(a_t | s_t)$  // Take action
      Observe reward  $r_t$  and next state  $s_{t+1}$ .
      Set  $r_{t,k} = r_t$ .
    end for
    Save total reward:
    
$$\hat{V}_k(s) = \sum_{t=1}^T r_{t,k}.$$

  end for
  Calculate estimate:
  
$$\hat{V}(s) = \frac{1}{K} \sum_{k=1}^K \hat{V}_k(s).$$

end for

```

Remark 8.2.1. The estimate \hat{V} of the Monte Carlo evaluation algorithm satisfies

$$\|V - \hat{V}\|_\infty \leq \sqrt{\frac{\ln(2|\mathcal{S}|/\delta)}{2K}} \quad \text{with probability } 1 - \delta$$

Proof. From Hoeffding's inequality (??) we have for any state s that

$$\mathbb{P}\left(|\hat{V}(s) - V(s)| \geq \sqrt{\frac{\ln(2|\mathcal{S}|/\delta)}{2K}}\right) \leq \delta/|\mathcal{S}|.$$

Consequently, using a union bound of the form $P(A_1 \cup A_2 \cup \dots \cup A_n) \leq \sum_i P(A_i)$ gives the required result. \square

The main advantage of Monte-Carlo policy evaluation is that it can be used in very general settings. It can be used not only in Markovian environments such as MDPs, but also in partially observable and multi-agent settings.

8.2.3 Backwards induction policy evaluation

Finally, the backwards induction algorithm shown as Algorithm 9 is similar to the backwards induction algorithm we saw for sequential sampling and bandit problems. However, here we are only evaluating a policy rather than finding the optimal one. This algorithm is slightly less generally applicable than the Monte-Carlo method because it makes Markovian assumptions. The Monte-Carlo algorithm, can be used for environments that with a non-Markovian variable s_t .

Algorithm 9 Backwards induction policy evaluation

For each state $s \in S$, for $t = 1, \dots, T - 1$:

$$\hat{V}_t(s) = r_\mu^\pi(s) + \sum_{j \in S} \mathbb{P}_\mu^\pi(s_{t+1} = j \mid s_t = s) \hat{V}_{t+1}(j), \quad (8.2.5)$$

with $\hat{V}_T(s) = r_\mu^\pi(s)$.

Theorem 8.2.1. *The backwards induction algorithm gives estimates $\hat{V}_t(s)$ satisfying*

$$\hat{V}_t(s) = V_{\mu,t}^\pi(s) \quad (8.2.6)$$

Proof. For $t = T - 1$, the result is obvious. We can prove the remainder by induction. Let (8.2.6) hold for all $t \geq n + 1$. Now we prove that it holds for n . Note that from the recursion (8.2.5) we have:

$$\begin{aligned} \hat{V}_t(s) &= r_\mu(s) + \sum_{j \in S} \mathbb{P}_{\mu,\pi}(s_{t+1} = j \mid s_t = s) \hat{V}_{t+1}(j) \\ &= r(s) + \sum_{j \in S} \mathbb{P}_{\mu,\pi}(s_{t+1} = j \mid s_t = s) V_{\mu,t+1}^\pi(j) \\ &= r(s) + \mathbb{E}_{\mu,\pi}(U_{t+1} \mid s_t = s) \\ &= \mathbb{E}_{\mu,\pi}(U_t \mid s_t = s) = V_{\mu,t}^\pi(s), \end{aligned}$$

where the second equality is by the induction hypothesis, the third and fourth equalities are by the definition of the utility, and the last by definition of $V_{\mu,t}^\pi$. \square

8.2.4 Backwards induction policy optimisation

Backwards induction as given in Alg 10 is the first non-naive algorithm for finding an optimal policy for the sequential problems with T stages. It is basically identical to the backwards induction algorithm we saw in Chapter ??, which was for the very simple sequential sampling problem, as well as the backwards induction algorithm for the decision-theoretic bandit problem.

Algorithm 10 Finite-horizon backwards induction

Input μ , set \mathcal{S}_T of states reachable within T steps.
 Initialise $V_T(s) := \max_a r(s, a)$, for all $s \in \mathcal{S}_T$.
for $n = T - 1, T - 2, \dots, 1$ **do**
 for $s \in \mathcal{S}_n$ **do**
 $\pi_n(s) = \arg \max_a r(s, a) + \sum_{s' \in \mathcal{S}_{n+1}} P_\mu(s' \mid s, a) V_{n+1}(s')$
 $V_n(s) = r(s, a) + \sum_{s' \in \mathcal{S}_{n+1}} P_\mu(s' \mid s, \pi_n(s)) V_{n+1}(s')$
 end for
end for
 Return $\pi = (\pi_n)_{n=1}^T$.

Theorem 8.2.2. *For T -horizon problems, backwards induction is optimal, i.e.*

$$V_n(s) = V_{\mu,n}^*(s) \quad (8.2.7)$$

Proof. Note that the proof below also holds for $r(s, a) = r(s)$. First we show that $V_t \geq V_t^*$. For $n = T$ we evidently have $V_T(s) = \max_a r(s, a) = V_{\mu, T}^*(s)$. Now assume that for $n \geq t + 1$, (8.2.7) holds. Then it also holds for $n = t$, since for any policy π'

$$\begin{aligned} V_t(s) &= \max_a \left\{ r(s, a) + \sum_{j \in \mathcal{S}} P_\mu(j | s, a) V_{t+1}(j) \right\} \\ &\geq \max_a \left\{ r(s, a) + \sum_{j \in \mathcal{S}} P_\mu(j | s, a) V_{\mu, t+1}^*(j) \right\} \quad (\text{by induction assumption}) \\ &\geq \max_a \left\{ r(s, a) + \sum_{j \in \mathcal{S}} P_\mu(j | s, a) V_{\mu, t+1}^{\pi'}(j) \right\} \\ &\geq V_t^{\pi'}(s). \end{aligned}$$

This holds for any policy π' , including $\pi' = \pi$, the policy returned by backwards induction. Then:

$$V_{\mu, t}^*(s) \geq V_{\mu, t}^{\pi}(s) = V_t(s) \geq V_{\mu, t}^*(s).$$

□

Remark 8.2.2. A similar theorem can be proven for arbitrary \mathcal{S} . This requires using \sup instead of \max and proving the existence of a π' that is arbitrary-close in value to V^* . For details, see[?].

8.3 Infinite-horizon

When problems have no fixed horizon, they usually can be modelled as infinite horizon problems, sometimes with help of a *terminating state*, whose visit terminates the problem, or discounted rewards, which indicate that we care less about rewards further in the future. When reward discounting is exponential, these problems can be seen as undiscounted problems with random and geometrically distributed horizon. For problems with no discounting and no termination states there are some complications in the definition of optimal policy. However, we defer discussion of such problems to Chapter ??.

8.3.1 Examples

We begin with some examples, which will help elucidate the concept of terminating states and infinite horizon. The first is shortest path problems, where the aim is to find the shortest path to a particular goal. Although the process terminates when the goal is reached, not all policies may be able to reach the goal, and so the process may never terminate.

Shortest-path problems

We shall consider two types of shortest path problems, deterministic and stochastic. Although conceptually very different, both problems have essentially the same complexity.

Consider an agent moving in a maze, aiming to get to some terminating goal state X . That is, when reaching this state, the agent cannot move anymore, and receives a reward of 0. In general, the agent can move deterministically in the four cardinal directions, and receives a

negative reward at each time step. Consequently, the optimal policy is to move to X as quickly as possible.

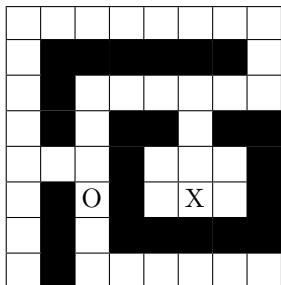
14	13	12	11	10	9	8	7
15		13				6	
16	15	14	4	3	4	5	
17				2			
18	19	20	2	1	2		
19		21	1	0	1		
20		22					
21		23	24	25	26	27	28

Properties

- $\gamma = 1, T \rightarrow \infty$.
- $r_t = -1$ unless $s_t = X$, in which case $r_t = 0$.
- $\mathbb{P}_\mu(s_{t+1} = X|s_t = X) = 1$.
- $\mathcal{A} = \{\text{North, South, East, West}\}$
- Transitions are deterministic and walls block.

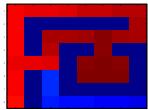
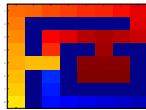
Solving the shortest path problem can be done simply by looking at the distance of any point to X . Then the reward obtained by the optimal policy starting from any point, is simply the negative distance. The optimal policy simply moves to the state with the smallest distance to X .

Stochastic shortest path problem with a pit Now assume the shortest path problem with stochastic dynamics. That is, at each time-step there is a small probability ω that move to a random direction. In addition, there is a pit O , that is a terminating state with a reward of -100 .



Properties

- $\gamma = 1, T \rightarrow \infty$.
- $r_t = -1$, but $r_t = 0$ at X and -100 at O and episode ends.
- $\mathbb{P}_\mu(s_{t+1} = X|s_t = X) = 1$.
- $\mathcal{A} = \{\text{North, South, East, West}\}$
- Moves to a random direction with probability ω . Walls block.

(a) $\omega = 0.1$ (b) $\omega = 0.5$

(c) value

Figure 8.2: Pit maze solutions for two values of ω .

Randomness changes the solution significantly in this environment. When ω is relatively small, it is worthwhile (in expectation) for the agent to pass past the pit, even though there is a risk of falling in and getting a reward of -100 . In the example given, even starting from the third row, the agent prefers taking the short-cut. For high enough ω , the optimal policy

avoids approaching the pit. Still, the agent prefers jumping in the pit, than being trapped at the bottom of the maze forever.

Continuing problems

Finally, many problems have no natural terminating state, but are continuing *ad infinitum*. Frequently, we model those problems using a utility that discounts future rewards exponentially. This way, we can guarantee that the utility is bounded. In addition, exponential discounting also has some economical sense. This is partially because of the effects of inflation, and partially because money now may be more useful than money in the future. Both these effects diminish the value of money over time. As an example, consider the following inventory management problem.

EXAMPLE 57 (Inventory management). There are K storage locations, and each location i can store n_i items. At each time-step there is a probability ϕ_i that a client tries to buy an item from location i , where $\sum_i \phi_i \leq 1$. If there is an item available, when this happens, you gain reward 1. There are two types of actions, one for ordering a certain number u units of stock, paying $c(u)$. Further one may move u units of stock from one location i to another location j , paying $\psi_{ij}(u)$.

An easy special case is when $K = 1$, and we assume that deliveries happen once every m timesteps, and each time-step a client arrives with probability ϕ . Then the state set $\mathcal{S} = \{0, 1, \dots, n\}$ corresponds to the number of items we have, the action set $\mathcal{A} = \{0, 1, \dots, n\}$ to the number of items we may order. The transition probabilities are given by $P(s'|s, a) = \binom{m}{d} \phi^d (1 - \phi)^{m-d}$, where $d = s + a - s'$, for $s + a \leq n$.

8.3.2 MDP Algorithms

Let us now look at three basic algorithms for solving a known Markov decision process. The first, *value iteration*, is a simple extension of the backwards induction algorithm to the infinite horizon case.

Value iteration

In this version of the algorithm, we assume that rewards are dependent only on the state. An algorithm for the case where reward only depends on the state can be obtained by replacing $r(s, a)$ with $r(s)$.

Algorithm 11 Value iteration

```

Input  $\mu, \mathcal{S}$ .
Initialise  $v_0 \in \mathcal{V}$ .
for  $n = 1, 2, \dots$  do
    for  $s \in \mathcal{S}_n$  do
         $\pi_n(s) = \arg \max_{a \in \mathcal{A}} \{r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_\mu(s' | s, a) v_{n-1}(s')\}$ 
         $v_n(s) = r(s, \pi_n(s)) + \gamma \sum_{s' \in \mathcal{S}} P_\mu(s' | s, \pi_n(s)) v_{n-1}(s')$ 
    end for
    break if termination-condition is met
end for
Return  $\pi_n, V_n$ .

```

The value iteration algorithm is a direct extension of the backwards induction algorithm for an infinite horizon. However, since we know that stationary policies are optimal, we do not need to maintain the values and actions for all time steps. At each step, we can merely keep the previous value \mathbf{v}_{n-1} . However, since there is an infinite number of steps, we need to know whether the algorithm converges to the optimal value, and what is the error we make at a particular iteration.

Theorem 8.3.1. *The value iteration algorithm satisfies*

- $\lim_{n \rightarrow \infty} \|\mathbf{v}_n - \mathbf{v}^*\| = 0$.
- For each $\epsilon > 0$ there exists $N_\epsilon < \infty$ such that for all $n \geq N_\epsilon$

$$\|\mathbf{v}_{n+1} - \mathbf{v}_n\| \leq \epsilon(1 - \gamma)/2\gamma. \quad (8.3.1)$$

- For $n \geq N_\epsilon$ the policy π_ϵ that takes action

$$\arg \max_a r(s, a) + \gamma \sum_j p(j|s, a) \mathbf{v}_n(s')$$

is ϵ -optimal, i.e. $V_\mu^{\pi_\epsilon}(s) \geq V_\mu^*(s) - \epsilon$ for all states s .

- $\|\mathbf{v}_{n+1} - \mathbf{v}^*\| < \epsilon/2$ for $n \geq N_\epsilon$.

Proof. The first two statements follow from the fixed-point Theorem ???. Now note that

$$\|V_\mu^{\pi_\epsilon} - \mathbf{v}^*\| \leq \|V_\mu^{\pi_\epsilon} - \mathbf{v}_n\| + \|\mathbf{v}_n - \mathbf{v}^*\|$$

We can bound these two terms easily:

$$\begin{aligned} \|V_\mu^{\pi_\epsilon} - \mathbf{v}_{n+1}\| &= \|\mathcal{L}_{\pi_\epsilon} V_\mu^{\pi_\epsilon} - \mathbf{v}_{n+1}\| && \text{(by definition of } \mathcal{L}_{\pi_\epsilon} \text{)} \\ &\leq \|\mathcal{L}_{\pi_\epsilon} V_\mu^{\pi_\epsilon} - \mathcal{L} \mathbf{v}_{n+1}\| + \|\mathcal{L} \mathbf{v}_{n+1} - \mathbf{v}_{n+1}\| && \text{(triangle)} \\ &= \|\mathcal{L}_{\pi_\epsilon} V_\mu^{\pi_\epsilon} - \mathcal{L}_{\pi_\epsilon} \mathbf{v}_{n+1}\| + \|\mathcal{L} \mathbf{v}_{n+1} - \mathcal{L} \mathbf{v}_n\| && \text{(by definition)} \\ &\leq \gamma \|V_\mu^{\pi_\epsilon} - \mathbf{v}_{n+1}\| + \gamma \|\mathbf{v}_{n+1} - \mathbf{v}_n\|. && \text{(by contraction)} \end{aligned}$$

An analogous argument gives the same bound for the second term $\|\mathbf{v}_n - \mathbf{v}^*\|$. Then, rearranging we obtain

$$\|V_\mu^{\pi_\epsilon} - \mathbf{v}_{n+1}\| \leq \frac{\gamma}{1 - \gamma} \|\mathbf{v}_{n+1} - \mathbf{v}_n\|, \quad \|\mathbf{v}_{n+1} - \mathbf{v}^*\| \leq \frac{\gamma}{1 - \gamma} \|\mathbf{v}_{n+1} - \mathbf{v}_n\|,$$

and the third and fourth statements follow from the second statement. \square

The *termination condition* of value iteration has been left unspecified. However, the theorem above shows that if we terminate when (8.3.1) is true, then our error will be bounded by ϵ . However, better termination conditions can be obtained.

Now let us prove how fast value iteration converges.

Theorem 8.3.2 (Value iteration monotonicity). *Let \mathcal{V} be the set of value vectors with Bellman operator \mathcal{L} . Then:*

1. Let $\mathbf{v}, \mathbf{v}' \in \mathcal{V}$ with $\mathbf{v}' \geq \mathbf{v}$. Then $\mathcal{L}\mathbf{v}' \geq \mathcal{L}\mathbf{v}$.
2. Let $\mathbf{v}_{n+1} = \mathcal{L}\mathbf{v}_n$. If there is an N s.t. $\mathcal{L}\mathbf{v}_N \leq \mathbf{v}_N$, then $\mathcal{L}\mathbf{v}_{N+k} \leq \mathbf{v}_{N+k}$ for all $k \geq 0$ and similarly for \geq .

Proof. Let $\pi \in \arg \max_{\pi} \mathbf{r} + \gamma \mathbf{P}_{\mu, \pi} \mathbf{v}$. Then

$$\mathcal{L}\mathbf{v} = \mathbf{r} + \gamma \mathbf{P}_{\mu, \pi} \mathbf{v} \leq \mathbf{r} + \gamma \mathbf{P}_{\mu, \pi} \mathbf{v}' \leq \max_{\pi'} \mathbf{r} + \gamma \mathbf{P}_{\mu, \pi'} \mathbf{v}',$$

where the first inequality is due to the fact that $\mathbf{P}\mathbf{v} \geq \mathbf{P}\mathbf{v}'$ for any \mathbf{P} . For the second part,

$$\mathcal{L}\mathbf{v}_{N+k} = \mathbf{v}_{N+k+1} = \mathcal{L}^k \mathcal{L}\mathbf{v}_N \leq \mathcal{L}^k \mathbf{v}_N = \mathbf{v}_{N+k}.$$

since $\mathcal{L}\mathbf{v}_N \leq \mathbf{v}_N$ by assumption and consequently $\mathcal{L}^k \mathcal{L}\mathbf{v}_N \leq \mathcal{L}^k \mathbf{v}_N$ by part one of the theorem. \square

Thus, value iteration converges monotonically to V_{μ}^* if the initial value $\mathbf{v}_0 \leq \mathbf{v}'$ for all \mathbf{v}' . If $r \geq 0$, it is sufficient to set $\mathbf{v}_0 = \mathbf{0}$. Then \mathbf{v}_n is always a lower bound on the optimal value function.

Theorem 8.3.3. *Value iteration converges with error in $O(\gamma^n)$. More specifically, for $r \in [0, 1]$ and $\mathbf{v}_0 = \mathbf{0}$,*

$$\|\mathbf{v}_n - V_{\mu}^*\| \leq \frac{\gamma^n}{1 - \gamma}, \quad \|V_{\mu}^{\pi_n} - V_{\mu}^*\| \leq \frac{2\gamma^n}{1 - \gamma}.$$

Proof. The first part follows from the contraction property (Theorem ??):

$$\|\mathbf{v}_{n+1} - \mathbf{v}^*\| = \|\mathcal{L}\mathbf{v}_n - \mathcal{L}\mathbf{v}^*\| \leq \gamma \|\mathbf{v}_n - \mathbf{v}^*\|. \quad (8.3.2)$$

Now divide by γ^n to obtain the final result. \square

Although value iteration converges exponentially fast, the convergence is dominated by the discount factor γ . When γ is very close to one, convergence can be extremely slow. In fact, ? showed that the number of iterations are on the order of $1/(1 - \gamma)$, for bounded accuracy of the input data. The overall complexity is $\tilde{O}(|\mathcal{S}|^2 |\mathcal{A}| L(1 - \gamma)^{-1})$, omitting logarithmic factors, where L is the total number of bits used to represent the input.¹

¹Thus the result is *weakly* polynomial complexity, due to the dependence on the input size description.

Chapter 9

Index

Index

- Adaptive hypothesis testing, 146
- Adaptive treatment allocation, 146
- backwards induction, 153
- bandit problem
 - contextual, 156
- bandit problems, 146
 - stochastic, 146
- belief state, 149
- clinical trial, 146
- confounder, 134
- experimental design, 146
- gradient ascent, 63
 - stochastic, 63, 106, 139
- hierarchical Bayesian model, 95, 96
- hierarchical Bayesian model, Beta-Bernoulli, 98
- Hoeffding inequality, 167
- importance sampling, 97
- see variable
 - instrumental, 134
- latent, 95
- Markov decision process, 146, **162**, 166
- policy, 147, 164
 - bandits, 147
 - classification, 25, 29
 - classification accuracy, 30
 - database response, 81
 - default, 135
 - evaluation, 136
 - exponential mechanism, 85
 - for statistical testing, 52
 - history-dependent, 164
 - intervention, 130, 136
 - Markov, 164
- neural network classification, 63
- optimal, 165
- optimisation, 138
- recommendation, 118, 130
- policy evaluation, 165
 - backwards induction, 167
 - Monte Carlo, 166
- policy optimisation
 - backwards induction, 168
- regret, 155
- reward distribution, 162
- risk, 48
- Simpson's paradox, 92, 137
- termination condition, 172
- transition distribution, 162
- utility, 164
- value function
 - optimal, 165
 - state, 164
 - state-action, 165
- value iteration, 171
- variable
 - instrumental, 134

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] Su Lin Blodgett and Brendan O’Connor. Racial disparity in natural language processing: A case study of social media african-american english. *CoRR*, abs/1707.00061, 2017. URL <http://arxiv.org/abs/1707.00061>.
- [5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. URL citeseer.nj.nec.com/breiman96bagging.html.
- [6] L. Elisa Celis and Nisheeth K. Vishnoi. Fair personalization. *CoRR*, abs/1707.02260, 2017. URL <http://arxiv.org/abs/1707.02260>.
- [7] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair learning in markovian environments. *FATML*, 2017.
- [8] 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>.
- [9] Christos Dimitrakakis, Yang Liu, David Parkes, and Goran Radanovic. Bayesian fairness. Technical Report 1706.00119, arXiv, 2017. URL <https://arxiv.org/abs/1706.00119>.
- [10] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM, 2012.
- [11] 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.

- [12] Dean Eckles and Maurits Kaptein. Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009*, 2014.
- [13] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951.
- [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] Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, and Aaron Roth. Fair learning in markovian environments. *arXiv preprint arXiv:1611.03071*, 2016.
- [16] Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. Rawlsian fairness for machine learning. *arXiv preprint arXiv:1610.09559*, 2016.
- [17] Niki Kilbertus, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. Technical Report 1706.02744, arXiv, 2017.
- [18] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. Technical Report 1609.05807, arXiv, 2016.
- [19] 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.
- [20] 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.
- [21] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 83–90. ACM, 2012.
- [22] 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.