1

An Introduction to Bayesian Methods for Interaction Design

John H. Williamson

Abstract

Bayesian modelling has much to offer those working in human–computer interaction, but many of the concepts are alien. This chapter introduces Bayesian modelling in interaction design. The chapter outlines the philosophical stance that sets Bayesian approaches apart, as well as a light introduction to the nomenclature and computational and mathematical machinery. We discuss specific models of relevance to interaction, including probabilistic filtering, non-parametric Bayesian inference, approximate Bayesian computation and belief networks. We include a worked example of a Fitts' law modelling task from a Bayesian perspective, applying Bayesian linear regression via a probabilistic program. We identify five distinct facets of Bayesian interaction: probabilistic interaction in the control loop; Bayesian optimisation at design time; analysis of empirical results with Bayesian statistics; visualisation and interaction with Bayesian models; and Bayesian cognitive modelling of users. We conclude with a discussion of the pros and cons of Bayesian approaches, the ethical implications therein and suggestions for further reading.

1.1 Introduction

We assume that most readers will be coming to this text from an interaction design background and are looking to expand their knowledge of Bayesian approaches, and we started from this framing when structuring this chapter. Some readers may be coming the other way, from a Bayesian statistics background to interaction design. These readers will find interesting problems and applications of statistical methods in interaction design.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

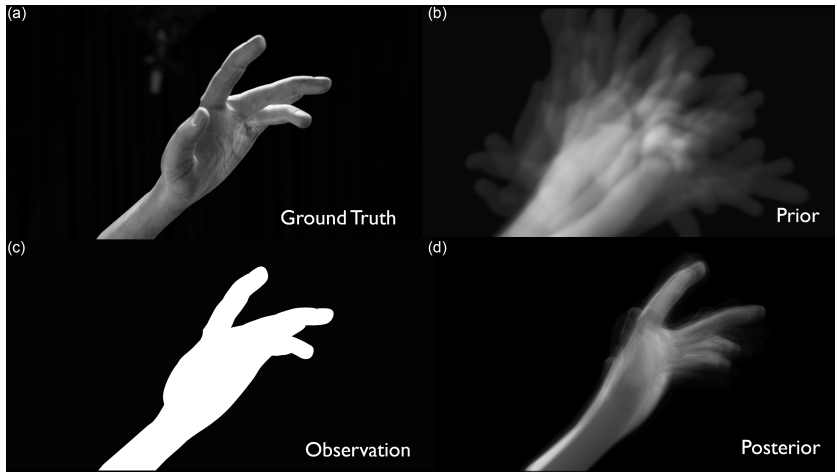

Figure 1.1 Which hand pose, generated the silhouette (c)? We cannot resolve a unique answer from this **observation**. Instead, we can start from a **prior** set (b) of viable hand poses and infer the distribution of *likely* poses given the observed silhouette (the **posterior** belief about poses) (d). **Uncertainty** about the pose is represented in the prior and reduced (but still present) in the posterior, inferred after observing the silhouette.

This book discusses how Bayesian approaches can be used to build models of human interactions with machines. Modelling is the cornerstone of good science, and actionable computational models of interactive systems are the basis of *computational interaction* [42]. A Bayesian approach makes *uncertainty* a first-class element of a model and provides the technical means to reason about uncertain beliefs computationally.

Human–computer interaction is rife with uncertainty. Explicitly modelling uncertainty is a bountiful path to better models and ultimately better interactive systems. The Bayesian world view gives an elegant and compelling basis to reason about the problems we face in interaction, and it comes with a superbly equipped wardrobe of computational tools to apply the theory. Bayesian approaches can be engaged across the whole spectrum of interaction, from the most fine-grained, pixel-level modelling of a pointer to questions about the social impact of always-on augmented reality. Everyone involved in interaction design, at every level, can benefit from these ideas. Thinking about interaction in Bayesian terms can be a refreshing perspective to re-examine old problems.

And, as this book illustrates, it can also be transformational in practically delivering the human–computer interactions of the future.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

A Note on This Chapter
This chapter is intended to be a high-level look at Bayesian approaches *from the point of view of an interaction designer*. Where possible, I have omitted mathematical terminology; the Appendix of the book gives a short introduction to standard terminology and notation. In some places I have provided skeleton code in Python. This is not intended to be executable, but to be a readable way to formalise the concepts for a computer scientist audience, and should be interpretable even if you are not familiar with Python. All data and examples are synthetic.

The chapter is structured as follows:

- A short introduction to Bayesian inference for the unfamiliar.
- A high-level discussion of the distinctive aspects of Bayesian modelling.
- A detailed worked example of Bayesian modelling in an HCI problem.
- A short summary of Bayesian algorithms and techniques particularly relevant to interaction, including approximate Bayesian computation, Bayesian optimisation and probabilistic filtering.
- A discussion of the important facets of Bayesian interaction design.
- Finally, a reflection on the implications of these ideas as well as recommendations for further reading.

1.1.1 What Are Bayesian Methods?

Bayesian methods is a broad term. In this book, the ideas are linked by the fundamental property of representing *uncertain belief* using probability distributions, and updating those beliefs with evidence. The underpinning of probability theory puts this on a firm theoretical basis, but the concept is simple: we represent what we know about the specific aspects of the world with a **distribution** that tells us how likely possible configurations of the world are, and then refine belief about these possibilities with data. We can repeat this process as required, accumulating evidence and reducing our uncertainty. In its

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

simplest form, this boils down to simply counting potential configurations of the world, then adjusting those counts to be compatible with some observed data. This idea has been become vastly more practical as computational power has surged, making the efficient 'counting of possibilities' a feasible task for complex problems.

Why Model at All?

'I am never content until I have constructed a mechanical model of the subject I am studying. If I succeed in making one, I understand, otherwise I do not.' – Lord Kelvin, *Baltimore Lectures on Molecular Dynamics and the Wave Theory of Light.* 1884
For the twenty-first century, replace 'mechanical' with 'computational'.

Modelling creates a simplified version of a problem that we can more easily manipulate, and could be mathematical, computational or physical in nature. Good science depends on good models. Models can be shared, criticised and re-used. Fields of study where there is healthy exchange of models can 'ratchet' constructively, one investigation feeding into the next. In interaction design, modelling has been relatively weak. When models have been used, they have often been descriptive in nature rather than causal. One of the motivations for a Bayesian approach is in the adoption of statistical models that are less about describing or predicting the superficial future state of the world and more about predicting the underlying state of the world. The other motivation is to build and work with models that properly account for uncertainty.

We can consider the relative virtues of models, in terms of their authenticity to the real-world phenomena, their complexity or their mathematical convenience. However, for the purposes of human–computer interaction, several virtues are especially relevant:

- Models that are *generative* and can be executed in a computer simulation to produce synthetic data.
- Models that are *computational* and can be manipulated, transformed and validated algorithmically; for example, written as programs.
- Models that are conveniently *parameterisable* and ideally have parameters that are meaningful and interpretable.
- Models that are *causal* and describe the underlying origins of phenomena rather than predict the manifestations of phenomena.
- Models that preserve and propagate *uncertainty*.
- Models that fit well with software engineering practices to deploy them, whether embedded in an interaction loop, in design tools or in analyses of evaluations.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

1.1.2 What Is Distinctive about *Bayesian* Modelling?

Bayesian modelling has several salient consequences:

- We can often directly use *simulators* of the process we believe to be generating the world that we observe, instead of relying on abstract, fixed models that can be difficult to shoehorn into interaction problems. For example, we might be able to use a detailed, agent-based simulation of pedestrian movement rather than a standard regression model.
- We reason from belief to evidence, not the other way around. This subtle difference means that we have a way to easily fuse information from many sources. This can range from sensor fusion in an inertial measuring unit to meta-reviews of surveys in the literature.
- We have a *universal* approach to solving problems that gives us a simple and consistent way to formulate questions and reason our way to answers.
- We also have a *universal* language with which to exchange and combine information: the probability distribution. Want to plug a language model into a gesture recogniser? No problem – exchange probability distributions.
- That same freedom and flexibility to model, and the need to represent distributions rather than values, implies technical difficulties. The devil is in the details.

1.1.3 How Is This Relevant to Interaction Design?

Everything we do with interactive systems has substantial *uncertainty* inherent in it. We don't know who our users are. We don't know what they want, or how they behave, or even how they tend to move. We don't know where they are, or in what context they are operating. The evidence that we *can* acquire is typically weakly informative and often indirectly related to problems we wish to address. This extends across all levels, from tightly closed control loops to design-time questions or retrospective evaluations. For example:

- Do a user's pointing movements indicate an intention to press button A or B?
- Is now a good moment to pop up a dialog?
- How many touch interaction events will happen in the next 500 ms?
- How tired is the user right now?
- Is it better to allocate a shorter keyboard shortcut to Save or for Refresh?
- Does adding spring-back to a scrolling menu increase or decrease user stress?
- Which volatility visualisation strategy helps users make more rational decisions?
- Is this interactive system more or less likely to polarise society?

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

We typically have at least partial models of how the human world works: from psychology, physiology, sociology or physics. Good interaction design behooves us to take advantage of all the modelling we can derive from the research of others. Being able to slot together models from disparate fields is essential to advance science. The Bayesian approach of formally incorporating knowledge as priors can make this a consistent and reasonable thing to do.

We are in the business of interacting with computers – so computational methods are universally available to us. We care little about methods that are efficient to be hand-solved algebraically. The blossoming field of *computational* Bayesian statistics means that we can realistically embed Bayesian models in interactive systems or use them to design and analyse empirical studies at the push of a button. We have problems where it is important to pool and fuse information, whether in low-level fusion of sensor streams or in combining survey data from multiple studies. We have fast CPUs and GPUs and software libraries that subsume the fiddly details of inference.

1.1.4 What Does This Give Us?

Why might we consider Bayesian approaches?

- Taming uncertainty by representing and manipulating it grants us robustness, whether this is robustness within a control loop or in the interpretation of the evaluation of a system. Represented uncertainty *regularises* predictions and avoids making extreme inferences based on limited data.
- We explicitly and precisely model *prior beliefs*. This allows knowledge to be encoded, inspected and shared, whether among software components or among researchers.
- A focus on generative models leads us to model constructively, to build models that synthesise what we expect to observe. These models can be strikingly more insightful than models that seek to summarise or describe what we have observed.
- Bayesian inference makes it realistic to fuse information from many sources without ad hoc tricks, and a principled way to deal with missing data and imputation.

Most of all, Bayesian approaches give us a new perspective from which to garner insight into problems of interaction, supported by a bedrock of mathematical and computational tools.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

1.1.5 Is This Just for Statistical Analysis?

Bayesian methods are a powerful tool for empirical analysis, and historically Bayesian methods have been used for statistical analyses of the type familiar to HCI researchers in user evaluations. But that is not their only role in interaction design, and arguably not even the most important role they can play. Bayesian methods can be used directly within the control loop as a way of robustly tracking states (for example, using probabilistic filtering). Bayesian optimisation makes it possible to optimise systems that are hard and expensive to measure, such as subjective responses to UI layouts. Bayesian ideas can change the way we think about how users make sense of interfaces, how we should represent uncertainty to them and how we should predict users' cognitive processes.

1.2 A Short Tutorial

Terminology We will use a number of specific terms in the rest of the chapter:

- **model** a simplified representation of a problem that has some parts that can vary. Our models will always be implemented as computer programs.
- **parameter** one variable in a model that partially determines how a model operates.
- **configuration** a collection of parameter values that fully specifies a specific instantiation of a model.
- **observations** values which we observe, i.e. data.
- **probability** a number between 0 and 1 representing how much we believe something.
- **distribution** an assignment of probabilities to possible configurations, defining how likely each is.
- **sample** a specific value, drawn at random according to a probability distribution.
- **prior** a belief about the world before observing data, as a distribution over configurations.
- **posterior** a belief about the world after observing data, as a distribution over configurations.
- **likelihood** a belief about how likely observations are, given a configuration, as a distribution over possible observations.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

We will also use the following notation for probability:

- $P(A)$ the probability that event A occurs.
- $P(A, B)$ the probability that both event A and event B occur together.
- $P(A|B)$ the probability that event A occurs, if we *know* that B occurs.

See the Appendix of this book for a more thorough explanation.

1.2.1 An Example of Bayesian Inference

Imagine we have three app variants deployed to a group of users, A, B and C. App A has two buttons on the splash screen, App B has four, and App C has nine. We get a log event that indicates that '3' in the splash screen was pressed, but not which app generated it. Which app was the user using, given this information (Figure 1.2)?

We have an unobserved **parameter** (which app is being used) and observed **evidence** (button 3 was pressed). Let us further assume there are 10 test users using the app: five using A, two using B and three using C. This gives us a **prior** belief about which app is being used (for example, if we knew nothing about the interaction, we expect it is 50% more likely that App C is being used than App B). We also need to assume a *model* of behaviour. We might assume a very simple model that users are equally likely to press any button – the **likelihood** of choosing any button is equal.

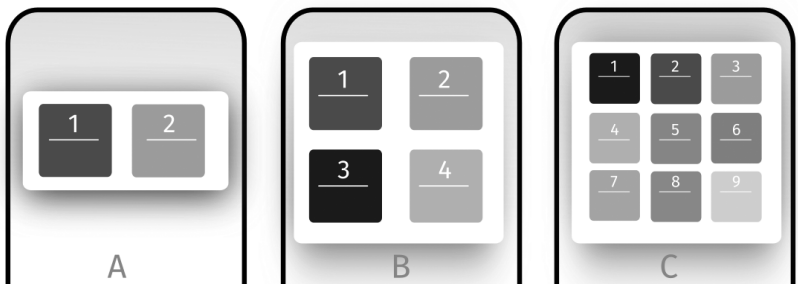

Figure 1.2 Which app was used? We know button 3 was pressed, but not on which app.

This is a problem of Bayesian updating (Figure 1.3); how to move from a prior probability distribution over apps to a posterior distribution over apps, having observed some evidence in the form of a button press.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

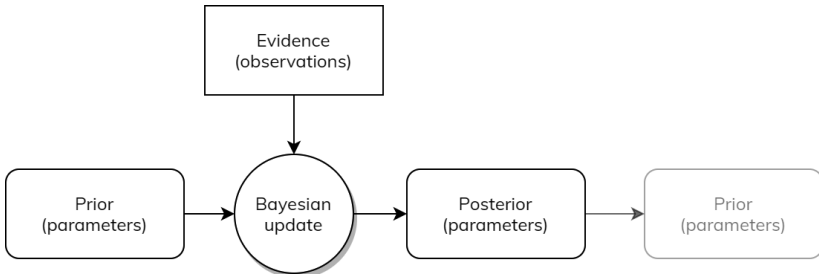

Figure 1.3 Bayesian inference takes a prior probability distribution, which represents beliefs about parameters, incorporates observed evidence and produces a posterior distribution which captures those beliefs that are compatible with the evidence. The posterior from one inference step can form the prior of a subsequent update.

How do we compute this? In this case, we can just count up the possibilities for each app, as shown in Figure 1.4.

We know that button 3 was logged, so:

- There is no possibility that the user was using App A, which has only two buttons.
- If they were using App B, 1/4 of the time they would press 3; and 1/9 of the time if using App C.

These numbers come directly from our assumption that buttons are pressed with equal likelihood, and so the **likelihoods** of seeing button 3 for each app are (A = 0, B = 1/4, C = 1/9). Given our prior knowledge about the number of apps in use, we can multiply these likelihoods by how likely we thought the particular app was *before* observing the '3'. This **prior** was (A = 5/10, B = 2/10, C = 3/10). This gives us: (A = 0 * 5/10, B = 1/4 * 2/10, C = 1/9 * 3/10) = (0, 1/20, 1/30). We can normalise this so it sums to 1 to make it a proper probability distribution: (0, 3/5, 2/5). This is the **posterior** distribution, the probability distribution revised to be compatible with the evidence. We *now* believe there is a 60% chance that App B was used and a 40% chance App C was used.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

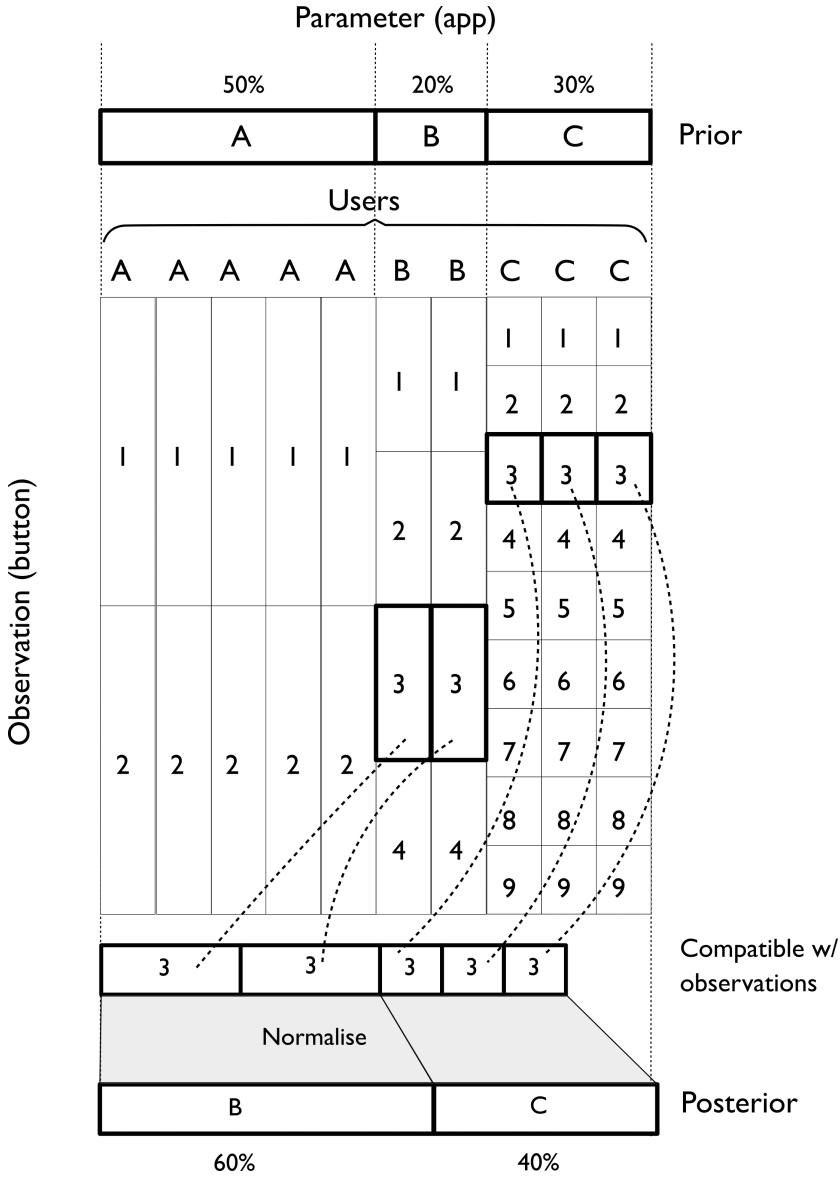

Figure 1.4 A table showing the parameters and the likelihood of each possible option (top). By selecting those compatible with the evidence, we can work out the division of possibilities that gives us the posterior probability distribution (bottom).

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Figure 1.5 Simulating the predictive model and highlighting elements where button 3 is pressed.

Figure 1.6 There are 12 Bs and 6 Cs in this random sample; a 66%/33% split close to the expected 60%/40% split.

This is easy to verify if we simulate this model to generate synthetic data.

```
import random

def simulate_app():
    # simulate a random user with a random app
    app = random.choice("AAAAABBCCC")
    if app=='A':
            button = random.choice([1,2])
    if app=='B':
            button = random.choice([1,2,3,4])
    if app=='C':
            button = random.choice([1,2,3,4,5,6,7,8,9])
    return app, button
```

If we run this simulation, and highlight the events where `button=3`, we get output as in Figure 1.5.

Sorting the selected events and shading them shows the clear pattern that B is favoured over C (Figure 1.6).

There are two key insights. First, the result of Bayesian inference is not always intuitively obvious, but if we can consider all possible configurations and count the compatible ones, we will correctly infer a probability distribution. Secondly, having a clear understanding of a model in terms of how it generates observations from unobserved parameters – to be able to simulate the model process – is a useful way to understand models and to verify their behaviour.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Another Observation

A Bayesian update transforms a probability distribution (over apps, in this case) to another probability distribution. What happens if we see *another* observation? For example, we might next observe that the user next pressed the '2' button on the same app. How does this affect our belief? *We use the posterior from the previous step (A = 0, B = 0.6, C = 0.4) as the new prior*, and repeat the exact same process to get a new posterior. We can do this process over and over again, as new observations arrive.

- New prior (old posterior): (A = 0, B = 0.6, C = 0.4)
- Observation: '2'
- Likelihood: (A = 1/2, B = 1/4, C = 1/9)
- Unnormalised posterior: (A = 1/2 * 0 = 0, B = 1/4 * 0.6 = 0.15, C = 1/9 * 0.4 = 0.044...)
- Posterior, after normalising: (A = 0, B = 0.77, C = 0.23)

We are now slightly more confident that the app being used is B, but with reasonable uncertainty between B and C. If the second button observed had instead been '6', the posterior would have assigned all probability to C and zero to all the other apps – because no other app could have generated a button press with label '6'.

Continuous Variables

When we want to deal with continuous variables and cannot exhaustively enumerate values, there are technical snags in extending the idea of counting. But modern inference software makes it easy to extend to a continuous world with little effort. These basic Bayesian concepts put very little restriction on the kinds of problems that we can tackle.

A continuous example When might we encounter continuous variables? Imagine we have an app that can show social media feeds at different font sizes. We might have a hypothesis that reading speed changes with font size. If we measure how long a user spent reading a message, what font size were they using (Figure 1.7)?

We cannot enumerate all possible reading times or font sizes, but we can still apply the same approach by assuming that these have distributions defined by functions which we can manipulate. A single measurement will in this case give us very little information because the inter-subject variability in reading time drowns out the useful signal; but sufficient measurements can be combined to update our belief.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Figure 1.7 Can we work out what font size someone is using from how long they spend reading a message?

1.2.2 A Bayesian Machine

At the heart of a Bayesian inference problem, we can imagine a *probabilistic simulator* as a device like Figure 1.8. This is a simulator that is designed to mimic some aspect of the world. The behaviour of the simulator is adjusted by *parameters*, which specify what the simulator will do. We can imagine these are dials that we can set to change the behaviour simulation. This simulator can (usually) take a real-world observation and pronounce its *likelihood*: how likely this observation was to have been generated by the simulator *given the current settings of the parameters*. It can typically also produce *samples*: example runs from the simulator with those parameter settings. This simulator is *stochastic*. One setting of the parameters will give different samples on different runs, because we simulate the random variation of the world. We assume that we have a probability distribution over possible parameter settings – some are more likely than others.

Here is the basic generative model, sketched in Python:

```
class Simulator:

    def samples(self, parameters, n):
        # return n random observations given parameters
        # (this corresponds to the output on the right)

    def likelihood(self, parameters, observations):
        # return the likelihood of some observation
        # *given the parameters* (i.e. dial settings)
        # (this corresponds to the input on the left)
```

Inference Engine

An inference engine can take a simulator like this and manage the distributions over the parameters. This involves setting prior distributions over the parameters and performing inference to compute posterior distributions using the likelihood given by the simulator. Parameter values drawn from the prior or posterior can be translated into synthetic observations by feeding them into the simulator,

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

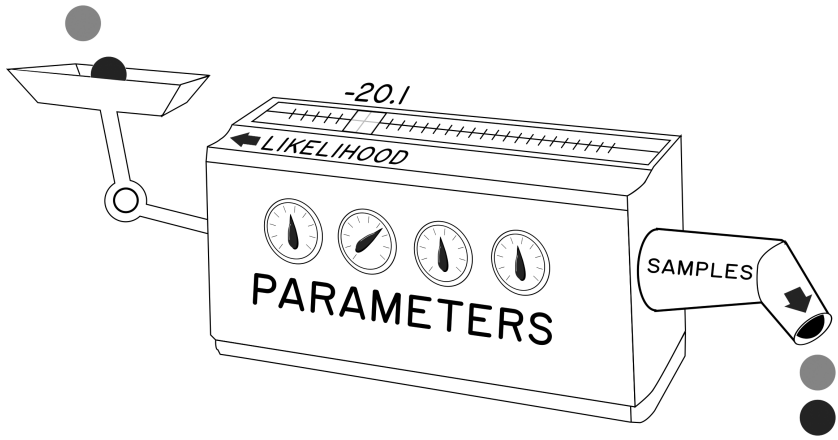

Figure 1.8 A cartoon of a probabilistic simulator, which encodes a model of the world. Parameters (dials, top) change the simulation. Distributions are maintained over possible dial settings. The simulator can synthesise samples (right spigot) or take real-world observations and determine how likely they are *under the current parameter settings* (left arm).

generating samples from the distribution known as the **posterior** (or prior) **predictive**. We can also compute summary results using **expectations**. To compute an expectation, we pass a function, and the inference engine computes the average value of that function evaluated at all possible parameters, weighted by how likely that parameter setting is.

The inference engine *inverts* the simulator. Given observations, it updates the likely settings of parameters.

A Reading Time Simulator

For example, we might model the reading time of the user based on font size, as in the example in the subsection 'Continuous Variables'. The simulator might have three 'dials' to adjust (parameters): the average reading time, the change in reading time per point of font size, and the typical random variation in reading time. This is a very simplistic cartoon of reading time, but enough that we can use it to model something about the world. By tweaking these parameters, we can set up different simulation scenarios.

```
import scipy.stats as st
import numpy as np

class ReadingTimeMachine:
    # store the initial parameters
    def __init__(self, mean_time,
```

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

```
                font_time, std_time):
    self.mean_time = mean_time
    self.font_time = font_time
    self.std_time = std_time


# given a font_size, generate
# n random simulations by drawing from
# a normal distribution
def simulate(self, n, font_size):
    model = st.norm(self.mean_time +
                    self.font_time * font_size,
                    std_time)

    return model.rvs(n)

# given a list of reading times and font sizes
# compute how likely a
# (reading_time, font_size) pair is
# under the current parameters. Return the sum
# of the log likelihood. The log is only used
# to make computations more numerically stable.
def log_likelihood(self, reading_times,
                        font_sizes):
    llik = 0
    for time, size in zip(reading_times, font_sizes):
            model = st.norm(self.mean_time +
                            self.font_time * size,
                            self.std_time)
            llik += model.logpdf(time)
    return llik
```

If we set the dials to 'average time = 500ms, font time = 10ms/pt, variation = +/−100ms' and cranked the sample output with font size set to 12 (i.e. called `simulate(n, font_size=12)`), the machine would spit out times: 600.9 ms, 553.3 ms, 649.2 ms, etc.

If we fed the machine an observation, say 300 ms and font size 8, it would give a (log-)likelihood (e.g. via `log_likelihood([300], [9]) = -9.7` for the settings above). Given another observation, say 1800 ms, it would give a much smaller value (−72.7, in this example), as such an observation is very unlikely given the settings of the machine.

Inference

One traditional, non-Bayesian approach to using this machine would be to feed a bunch of data into the likelihood inlet, and then iteratively adjust the parameters until the data was 'as likely as possible' – *maximum likelihood estimation* (MLE). This optimisation approach would tweak the dials to best

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

approximate the world (a model can never *reproduce* the world, but we can align its behaviour with the world).

> We'd usually not tweak the dials randomly until things got better, but use information about the slope or curvature of the likelihood function to quickly find the best setting, which is often done with automatic differentiation libraries. Traditionally, derivatives of likelihood functions were worked out and used for the optimisation process. This MLE approach is the basis of most machine learning, even if not always stated in these terms.

Bayesian inference instead puts prior distributions on the parameters, describing likely configurations of these parameters. Then, given the data, it computes how likely *every* possible combination of the parameters is by multiplying the prior by the likelihood of each sample. This gives us a new set of distributions for the parameters which is more tightly concentrated around settings that more closely correspond to the evidence.

What's the difference? In the optimisation (MLE) case, imagine we are using the reading time machine, and we only have an observation from one user, of 50 000 ms at font size 12. What is the most likely setting of the machine given this data? It will be a very unrealistic average time, a very large font size time, or an extremely large variation; any combination is possible.

> In practice, no one would use maximum likelihood in such a naive way, and instead would use some process to *regularise* the estimates that would have been used. However, this could be seen as a roundabout way of specifying a prior that implicitly favours certain parameterisations.

A Bayesian model would have specified likely distributions of the parameters in advance. A single data point would move these relatively little, especially one so unlikely under the prior distributions. We'd have to see *lots* of observations to be convinced that we'd really encountered a population of users who took 50 seconds to read a sentence.

Data space and parameter space In Bayesian modelling, it is important to distinguish the **data space** of observations and the **parameter space** of parameters in the model. These are usually quite distinct. In the example above, the observations are in a data space of reading times, which are just single scalar values. Each model configuration is a point in a three-dimensional parameter space `(mean_time, font_time, std_time)`. Both the prior and

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

the posterior are distributions over parameter space. Bayesian inference uses values in observation space to constrain the possible values in parameter space to move from the prior to the posterior. When we discuss the results of inference, we talk about the *posterior* distribution, which is the distribution over the parameters (dial settings on our machine).

The *posterior predictive* is the result of transforming the parameter space back into distribution of simulated observations *in the data space*. We could imagine getting these samples by repeatedly randomly setting the parameters according to the posterior distribution, then cranking the sample generation handle to simulate outputs. The *observations* would be measured reading times. The *prior* and *posterior* of the reaction times would be a distribution over `(mean_time, font_time, std_time)`. The *posterior predictive* would again be a distribution over reading times.

Types of Uncertainty: Aleatoric and Epistemic

This brings up a subtlety in Bayesian modelling. We have uncertainty about values, because the simulator, and the world it simulates, is stochastic; given a set of fixed parameters, it generates plausible values which have random variation. But we *also* have uncertainty about the settings of the parameters, which is what we are updating during the inference process. The inference process is indirect in this sense; it updates the 'hidden' or 'latent' parameters that we postulate are controlling the generative process. For example, even if we fix the parameters of our reading-time simulator, it will emit a range of plausible values. But we don't *know* the value of the parameters (such as font time), and so there is a distribution over these as well.

We can classify uncertainties: *aleatoric* uncertainty, arising from random variations, that cannot be eliminated by better modelling; and *epistemic* uncertainty, arising from our uncertainty about the model itself. In most computational Bayesian inference, we also have a third source of uncertainty: *approximation* uncertainty. This arises because most methods for performing Bayesian inference do not compute exact posterior distributions, and this introduces an independent source of variation. For example, many standard inference algorithms that applied the same model and same data twice would yield two different approximate posteriors, assuming the random seed was not fixed. The approximation error should be slight for well-behaved models but can be important, particularly for large and complex models where few samples can be obtained for computational reasons.

- **aleatoric** random noise in observations (e.g. random reading times given fixed parameters); modelled by the likelihood.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- **epistemic** the unknown state of parameters or model structure, modelled by the parameter distribution.
- **approximation** error in parameter estimates due to computational approximations, e.g. as introduced by Monte Carlo sampling.

1.3 Bayesian Approaches

1.3.1 What Are the Key Ideas in Bayesian Approaches?

There are some distinctive aspects of Bayesian approaches that distinguish them from other ways of solving problems in interaction design. We summarise these briefly, to give a flavour of how thinking and computation change as we move to a Bayesian perspective.

> We will refer to individuals applying Bayesian principles and adopting a Bayesian world view as 'Bayesians'. No one is really ever a 'true Bayesian', but it is a useful shorthand to delineate the Bayesian perspective from the non-Bayesian perspective. There are many subsets of Bayesian thought with slightly different assumptions and approaches; see Weisberg [57] for an in-depth discussion of the philosophical and technical distinctions of these varieties.

Beliefs Are Probabilities

A Bayesian represents *all* beliefs as probabilities. A belief is a measure of how likely a configuration of a system is. A probability is a real number between 0 and 1. Larger numbers indicate a higher degree of certainty. Manipulation of belief comes down to reassigning probabilities in the light of evidence.

For example, we might have a belief that two versions, A and B, of a website have different 'comprehension' scores, but we have no idea which is better, if any. We could represent this as a probability distribution, perhaps a 50/50 split in the absence of any further information: $A_{\text{better}} = 0.5, B_{\text{better}} = 0.5$. We could make observations, by running a user trial and gathering data, and form a new belief $A_{\text{better}} = 0.8, B_{\text{better}} = 0.2$. Whether version A or B is better is not a knowable fact (there isn't any possible route to precisely determine it), nor is it the result of some long sequence of identical experiments. Instead it is just a quantified belief. We used to believe that version A was as likely to be as good as B; we now believe that A is probably better (Figure 1.9).

If we went further, we might quantify *how much* better A was than B on some scale of relative comprehension and represent that as a probability distribution.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

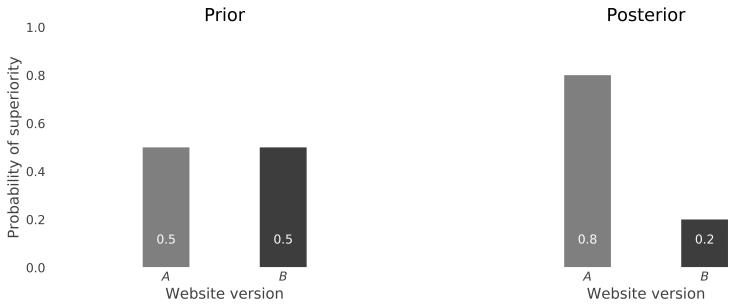

Figure 1.9 Belief that version A or B is the superior website for comprehension, represented as probabilities.

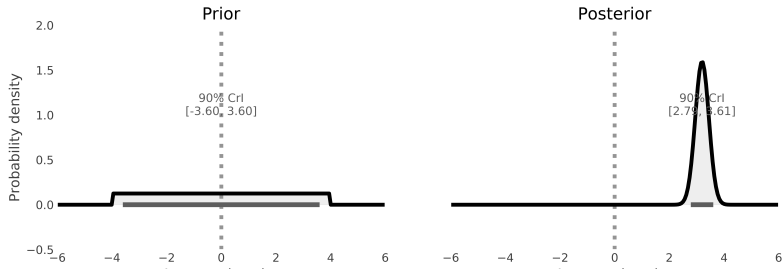

Figure 1.10 Distribution over relative change in reading comprehension contracts from prior to posterior.

Perhaps we'd assume that A could be anywhere from $[-4, 4]$ units of comprehension better than B. After doing an experiment, this might be concentrated with 90% of the probability now in the range $[2.7, 3.6]$ (Figure 1.10).

Representing a full distribution like this can be much more enlightening than a dichotomous approach that only considers the relative superiority of one belief above another. For example, we might know that each unit of increased comprehension is 'worth' 10 extra repeat visits to our website. We can now make concrete statements like 'we expect around 32 additional return visits with version B' directly from the posterior distribution. Representing distributions over many hypotheses can make *decisions* much easier to reason about.

Distributions, Not Points

To be Bayesian, we work not with definite values but with *distributions* over possible values. For example, we would not write code to find the best estimate of how long a user takes to notice a popup, or optimise to find the geometrical

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

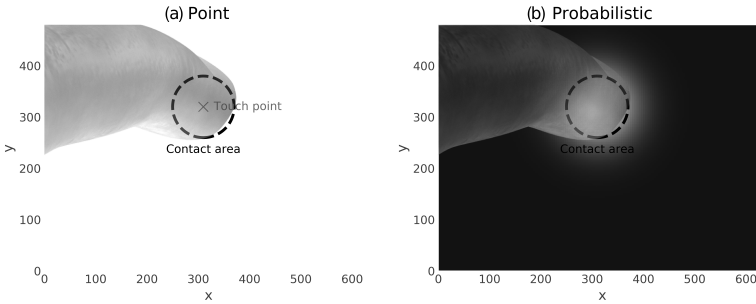

Figure 1.11 Imagine a touchscreen inferring a finger contact point from a contact blob (dashed circle). A standard approach would find the single point that best represents the intended touch (a). A Bayesian approach would form a probability distribution over possible touches, given an input (b).

configuration of the hand pose that is most compatible with a camera image. That is not congruent with a Bayesian world view which deals exclusively with beliefs about configurations. Instead, we would consider a distribution over *all possible* times, or *all possible* poses (Figure 1.11). After we update this with evidence (e.g. by running a user evaluation and showing many popups), we expect our distribution to have contracted around likely configurations. Wherever possible, we *keep* our beliefs as full distributions and avoid at all times reducing them to point estimates, such as the most likely configuration.

The fundamental principle: If we don't know something for sure, we preserve the whole distribution over possible configurations.

This has several consequences:

- We always explicitly have a representation of uncertainty.
- We have a universal language (or data type) with which we reason – the probability distribution.
- The probability distribution is a data type that is hard to work with, and consequently we often have to rely on approximations to do computations.
- We need to be able to summarise and visualise distributions to report them.

It is hard to communicate directly about distributions, so they are often reported using summary statistics, like the mean and standard deviation, or visualised with histograms or Box plots. Bayesian posterior distributions are

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

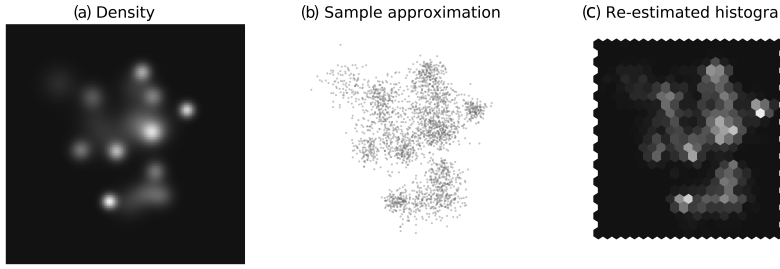

Figure 1.12 A complex density (a) can be approximated with a collection of samples (b) – definite values – which are easy to transform algorithmically. They can be transformed back into an approximate distribution, e.g. with a histogram (c).

often summarised in terms of **credible intervals** (CrI). These are intervals which cover a certain percentage of the probability mass or density. For example, a 90% credible interval defines an interval in which we believe that an unknown parameter lies with 90% certainty (given the priors).

Approximation Is King

Probability distributions are hard to work with. As a consequence, almost all practical Bayesian inference relies on approximations. Much of the traditional complexity of Bayesian methods comes from the contortions required to do manipulate distributions. This has become much less tricky now that there are software tools that can apply approximations to almost any model at the press of a button.

There are several important approximations used in practice, and they can largely be separated into two major classes: variational inference, where we represent a complex distribution with the 'best-fitting' distribution of a simpler one; and sample-based methods, where we represent distributions as collections of samples; definite points in parameter space (Figure 1.12).

Approximations can sometimes confound the use of Bayesian models. The obvious, pure way to solve a problem may not mesh well with the approximations available, and this may motivate changes in the model to satisfy the limitations of computational engines. This problem is lessening as 'plug and play' inference engines become more flexible, but it is often an unfortunate necessity to think about how a model will be approximated.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Integrate, Don't Optimise

As a consequence of the choice to represent belief as distribution, our techniques for solving problems are typically to *integrate* over possible configurations, rather than to optimise to find a precisely identified solution.

> *Integrate* is used here in the mathematical sense of 'summing over all possible values'.

For example, imagine a virtual keyboard that was interpreting a touch point as a keypress. We might model each key as being likely to be generated by some spatial distribution of touch points *given the size of the user's finger*, or more precisely the screen contact area of the finger pad. Depending on how big we believe the user's finger to be, our estimate of which key might have been intended will be different: a fat finger will be less precise.

How do we identify the key pressed, as Bayesians? We do *not* identify the most likely (or even worse, a fixed default) finger size and use that to infer the distribution over possible key presses. Instead, we would integrate over *all possible* finger sizes from a finger size distribution, and consider all likely possibilities. If we become more informed about the user's finger pad size, perhaps from a pressure sensor or from some calibration process, we can use that information immediately, by refining this finger size distribution (Figure 1.13).

This comes at a cost. As we increase the number of dimensions – the number of parameters we have – the volume of the parameter space to be considered increases exponentially. If we had to integrate over all possible finger sizes, all possible finger orientations, all possible skin textures and so on, this 'true' space of possibilities becomes enormous. This makes exhaustive integration computationally infeasible as models become more complex. The reason Bayesian methods work in practice is that approximations allow us to efficiently integrate 'where it matters' and ignore the rest of the parameter volume.

Expectations

The focus on integrating means that we often work with **expectations**, the expected *value* averaging over all possible configurations weighted by how likely they are. This assumes we attach some *value* to configurations. This could be a simple number, such as a dollar cost or a time penalty, or it could be a vector or matrix or any value which we can weight and add together.

In a touch keyboard example, we might have a correction penalty for a mistyped key, the cost of correcting a mistake; say in units of time. What is the expected correction penalty once we observe a touch point? We can compute this given a posterior distribution over keys, and a per-key correction cost.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

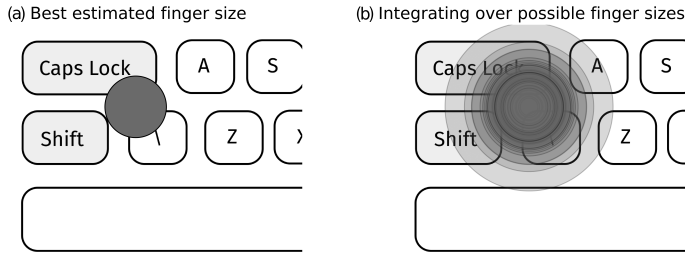

Figure 1.13 What key was intended to be pressed? This varies depending on what finger size we think the user has. We need to integrate over the possibilities, weighted by how likely they are, to get a correct distribution over possibilities. (a) A naive distribution using a single finger size splits the probability evenly between `|` and `Caps Lock` and zero elsewhere; (b) integrating over possible finger sizes indicates there is some small probability of `Shift`, and even `A` or `Z`.

Assume the key actuated is the key with highest probability, k_1. For each of the *other* keys, $k_2, \ldots, k_n$, we can multiply the probability that key k_i was intended by the penalty time it would take to correct k_1 to k_i. It is then trivial to consider the expected correction cost if some keys are more expensive than others (perhaps `backspace` has a high penalty, but `shift` has no penalty).

In an empirical analysis, we might model how an interactive exhibit in a museum affects reported subjective engagement versus a static exhibit. We might also have a model that predicts increase in time spent looking at an exhibit given a reported engagement. Following a survey, we could form a posterior distribution over subjective engagement, pass it through the time prediction model and compute the expected increase in dwell time that interaction brings (e.g. 49.2s additional dwell time).

Bayes' Rule

Bayesian inference updates beliefs using Bayes' rule. Bayes' rule is stated, mathematically, as follows:

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

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

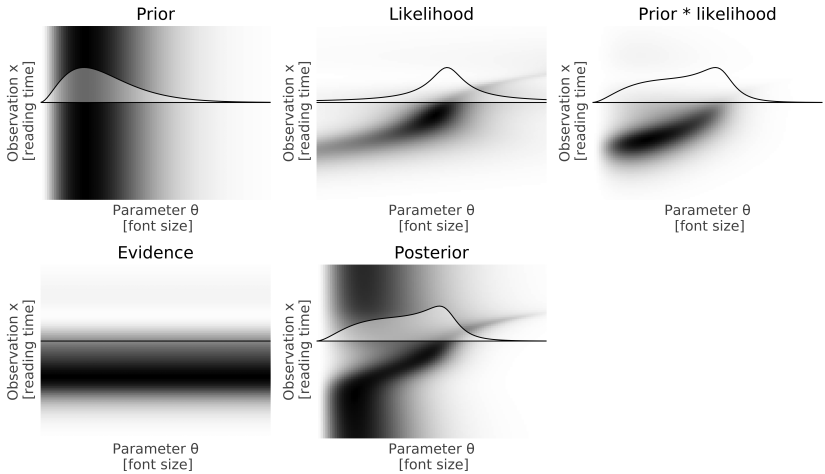

Figure 1.14 Bayesian inference. Consider a simple model with one parameter θ (say, font size) and one observed variable x (say, reading time). Assume we want to infer font size used given an observed reading time. The **prior** distribution weights possible values of θ in advance of seeing data, and does not depend on x. A **likelihood** function is defined by the model so that it maps every possible input x to a distribution over θ. If we observe a specific x (black horizontal line), one likelihood is 'selected' (light-shaded regions). This is multiplied by the prior, and then normalised by the **evidence** (which depends on x but not θ) to produce a proper **posterior** distribution over θ. The posterior can be used as the prior in a future inference of the same type (as in probabilistic filtering) or fed into another inference process.

or in words,

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}},$$

or often simplified to

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

This means that reasoning moves from a **prior** belief (what we believed before), to a **posterior** belief (what we now believe), by computing the **likelihood** of the data we have for every possible configuration of the model. We combine simply by multiplying the probability from the prior and the likelihood **over every possible configuration** (Figure 1.14).

To simplify our representations, and limit what we mean by 'every possible configuration', we assume that our models have some 'moving parts' – parameters, traditionally collected into a single vector θ – that describe specific configurations, in some space of possible configurations. All Bayes' rule tells

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

us is that the probability of each possible θ can be updated from some initial prior belief (quantified by a real number assigned to each configuration of θ, a probability) via a likelihood (giving us another number) and then normalising the result (the evidence) so that the probabilities of each configuration still add up to 1.

Priors

As a consequence of Bayes' rule, it is necessary for all Bayesian inference to have **priors**. That is, we must quantify, precisely, what we believe about every parameter before we observe data. This is analogous to traditional logic; we require **axioms**, which we can manipulate with logic to reach conclusions. It is not possible to reason logically without axioms, nor is it possible to perform Bayesian inference without priors. This is a powerful and flexible way of explicitly encoding beliefs. It has been curiously controversial in statistics, where it is has been criticised as *subjective*. We will leave the gory details of this debate to others.

Priors are defined by assigning probability distributions to the parameters. Priors can be chosen to enforce hard constraints (e.g. a negative reaction time to a visual stimulus is impossible unless we believe in precognition, so a prior on that parameter could reasonably have probability zero assigned to all negative times), but typically they are chosen so as to be *weakly informative* – they represent a reasonable bound on what we expect but do not rigidly constrain the possible posterior beliefs. Priors are an explicit way of encoding inductive bias, and the ability to specify a prior that captures domain knowledge grants Bayesian methods its great strength in small data regimes. When we have few data points, we can still make reasonable predictions if supported by an informative prior. Eliciting appropriate priors requires thought and engagement with domain experts.

Latent Variables

Bayesian approaches involve inference; the process of determining what is hidden from what is seen. We *assume* that there are some parameters that explain what we observe but whose value is not known. These are **hidden** or **latent** parameters.

For example, if we are building a computer vision-based finger pose tracker, we might *presume* a set of latent variables (parameters) that describe joint angles of the hand, and describe the images that we observe as states generated from the (unknown) true joint angle parameters. Inference refines our estimates of these joint angles following the observation of an image and allows us to establish what hand poses are compatible with the observed imagery – not

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

which hand pose, but what poses are likely. We never identify latent variables; we only refine our belief about plausible values.

Latent variables sometimes have to be accounted for in an inference, even though they are not what we are directly interested in. These are *nuisance variables*. For example, in the hand tracker, the *useful* parameters are the joint angles of the hand. But in a practical hand tracker, we might have to account for the lighting of the scene, or the camera lens parameters, or the skin tone. We are not interested in inferring these nuisance variables, but we may have to estimate them to reliably estimate the joint angles.

In a simpler scenario, we might predict how much time a user spends reading a news article on a mobile device as a function of the font size used, and *assume* that this follows some linear trend, characterised by a slope β_1 and a constant offset β_0:

$$\text{read time s} = \beta_1 * \text{font size pt} + \beta_0 + \text{noise}.$$

β_1, β_0 are latent variables that describe all possible forms of this line. By observing pairs of `read_time` and `font_size` we can narrow down our distribution over the latent variables β_0, β_1. Obviously, this simplistic model is not a true description of how reading works; but it is still a useful approximation.

In many scientific models there are many more latent variables than observed variables. Imagine inferring the complex genetic pathways in a biological system from a few sparse measurements of metabolite masses – there are many latent parameters and low-dimensional observations. In interaction, we sometimes have this problem: for example, modelling social dynamics with many unknown parameters from very sparse measurements. Often, however, we have the opposite problem, particularly when dealing with inference in the control loop: we have a large number of observed variables (e.g. pixels from an image from camera) which are generated from a much smaller small set of latent parameters (e.g. which menu option the user wants).

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Simulations and Generative Models

Bayesian methods were historically called the 'method of inverse probability'. This is because we build Bayesian models by writing down *what we expect to observe given some unobserved parameters*, and **not** *what unobserved parameters we have given some observation*. In other words, we write *forward models* that have some assumed underlying mechanics, governed by values we do not know. These are often not particularly realistic ways of representing the way the world works, but they are useful approximations that can lead to insight. We can run these models forward to produce synthetic observations and update our belief about the unobserved (latent) parameters. This is a **generative** approach to modelling; we build models that ought to generate what we observe.

Forward and inverse We can characterise Bayesian approaches as using generative, **forward** models, from parameters to observations. Approaches common in machine learning, like training a classifier to label images, are **inverse** models. These map from observations to hidden variables. A Bayesian image modelling approach would map labels $\rightarrow$ images; an inverse model would map images $\rightarrow$ labels. One major advantage of the generative approach is that it is easy to fuse other sources of information. For example, we might augment a vision system with an audio input. A Bayesian model would now map labels $\rightarrow$ images, sounds – two distinct manifestations of some common phenomena, with evidence from either being easily combined. An inverse model can be trained to learn sounds $\rightarrow$ labels, but it is harder to combine this with an existing images $\rightarrow$ labels model.

In practice, we usually use a combination of forward models and inverse models to make inference computationally efficient. For example, imagine we are tracking a cursor using an eye tracker. We want to know what on-screen spatial target a user is fixating on, given a camera feed from the eyes. A 'pure' Bayesian approach would generate *eye images* given targets; synthesise actual pixel images of eyes and compare them with observations to update a belief over targets (or be able to compute the likelihood of an image given a parameter setting).

This is theoretically possible but practically difficult, both for computational reasons (images are high-dimensional) and because of the need to integrate over a vast range of irrelevant variables (size of the user's pupils, colour of the viscera, etc.). A typical compromise solution would be to use an inverse model, such as traditional signal processing and machine learning, to extract a pupil contour from the image. Then, the Bayesian side could infer a distribution over parameterised contours given a target, and use that to identify targets.

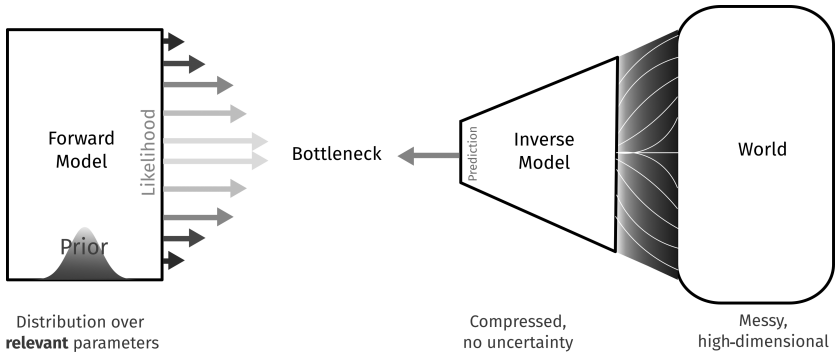

Figure 1.15 An inverse model which does not represent uncertainty but is efficient can be used to compress observations so that Bayesian inference can be used to maintain uncertainty over states that matter.

This is a common pattern: the **inverse model bottleneck**, where some early parts of the model are implemented in forward mode and inferred in a Bayesian fashion; but these are compared against results from an inverse model that has compressed the messy and high-dimensional raw observations into a form where Bayesian inference is practical (Figure 1.15). Combinations of modern non-Bayesian machine learning methods with Bayesian models can be extremely powerful. A deep network, for example, can be used to compress images into a lower-dimensional space to be fed into a Bayesian model. This can turn theoretically correct but computationally impractical pure Bayesian models into workable solutions.

Decision Rules and Utilities

Bayesian methods in their narrowest sense are concerned only with updating probabilities of different possible configurations. This, on its own, is insufficient to make **decisions**. In an HCI context, we often have to make irreversible state changes.

For example, in a probabilistic user interface, at some point, we have to perform actions, that is, make a decision about which action to perform. Similar issues come up when deciding whether interface A is more usable than interface B; we might well have *both* a probability of superiority of A over B, and a value gained by choosing A over B. Whenever we have to go from a distribution to a state change, we need a decision rule, and this usually implies that we also have a *utility function* $U(x)$ that ascribes values to outcomes.

In a probabilistic user interface, we might have just updated the distribution over the possible options based on a voice command, $P(\text{option}|\text{voice})$ from a

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Action	P(A|speech)	P(A|language)	P(A|speech,language)	U(Right)	U(Wrong)	Exp. U
reply all	0.28	0.38	0.65	1	−10	−2.90
reply Paul	0.03	0.38	0.06	1	−1	−0.87
delete Paul	0.56	0.04	0.13	1	−5	−4.23
delete all	0.14	0.19	0.16	1	−100	−83.71

Figure 1.16 An example of utility in decision making. This table shows possible voice commands that could be compatible with an utterance recorded. The speech recogniser gives some probability to different actions A according to the acoustics $P(A|\text{speech})$. This is combined with a prior from a language model that assigns probability to different commands based on prior usage $P(A|\text{language})$. These are combined into a posterior probability $P(A|\text{speech, language})$. For each action, there is also a possible benefit to the user $U(Right)$ (in this case they are all equal) and a possible cost $U(Wrong)$, which might capture the work required to undo the action if it were triggered in error. Given this table, the *least likely* option is 'reply Paul' but it is the option with highest *expected utility* **(Exp. U)** – the rational choice.

speech recogniser. Which option should be actuated? The probabilities don't tell us. We also need a **decision rule**, which will typically involve attribution of **utility** (goodness, danger, etc.) to those options.

The decision rule will combine the probability and the utility to identify which (if any) option should be actuated. A simple model is maximum expected utility: choose the action that maximises the average product of the probability and utility. This is a rational way to make decisions: choose the decision that is most likely to maximise the 'return' (or minimise the 'loss') in the long run. But many decision rules are possible which will be appropriate in different situations, such as:

- The option with the highest posterior probability (sometimes called the *maximum a posteriori* (MAP) estimate), if no utility function is known or appropriate. However, this ignores the fact that the outcomes might have different values ('reply all' is potentially more destructive than 'reply Paul').
- The option with the highest expected value, which is the product of the probability and the utility. In a probabilistic interface, this automatically makes useful actions easier to select and dangerous ones hard to select; we need to be more certain that an action is intended if it is less desirable.
- The option which minimises regret, which would also capture the lost utility of not selecting particularly valuable actions. This might be particularly relevant in time-constrained interactions, where actions may not be freely available in the future.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Any time we have to take action based on a Bayesian model, we need to define a decision rule to turn probabilities into choices. This almost always requires some form of utility function. Utility functions can be hard to define and may require careful thought and justification.

1.3.2 What about Machine Learning? Is It Just the Same Thing?

Modern machine learning uses a wide range of methods, but the dominant approach at the time of writing is distinctly optimisation focused, as opposed to Bayesian. A neural network, for example, is trained by adjusting parameters to find the **best** parameter setting that minimises a prediction error and so makes the best predictions (or some other loss function). A Bayesian approach to do the same task would find the distribution over parameters (network weights) most compatible with the observations, and not a single best estimate. There are extensive Bayesian machine learning models, from simple Bayesian logistic regression to sophisticated multi-layer Gaussian processes and Bayesian neural networks, but these are currently less widespread.

Most ML systems also try to map from some observation (like an image of a hand) to a hidden state (which hand pose is this?), learning the inverse problem directly, from outputs to inputs. This can be very powerful and is computationally efficient, but it is hard to fuse with other information. Bayesian models map from hidden states to observations, and adding new 'channels' of inputs to fuse together is straightforward – just combine the probability distributions.

Bayesian methods are most obviously applicable when uncertainty is relevant, and where the parameters that are being inferred are interpretable elements of a generative model. Bayesian neural networks, for example, provide some measure of uncertainty, but because the parameters of a neural network are relatively inscrutable, some of the potential benefit of a Bayesian approach is lost. Distributions over parameters are less directly useful in a black box context. All Bayesian machine learning methods retain the advantage of robustness in prediction that comes from representing uncertainty. They are less vulnerable to the specific details of the selection of the optimal parameter setting and may be able to degrade more gracefully when predictions are required from inputs far from the training data.

Bayesian machine learning methods have sometimes been seen as more computationally demanding, though this is perhaps less relevant in the era of billion-parameter-deep learning models. The ideal Bayesian method integrates over all possibilities, and so the problem complexity grows exponentially with dimension unless clever shortcuts can be used. Machine learning approaches like deep networks rely on (automatic) differentiation rather than exhaustive

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

integration, and more easily scale to large numbers of parameters. This is why we see deep networks with a billion parameters, but rarely Bayesian models with more than tens of thousands of parameters. However, many human–computer interaction problems have only a handful of parameters and are much more constrained by limited data than the flexibility of modelling. Bayesian methods are powerful in this domain.

Prediction and Explanation

Much of machine learning is focused on solving the *prediction* problem, learning to make predictions from data. Bayesian methods address predictions but can be especially powerful in solving the *explanation* problem; identifying what is generating or causing some observations. In an interaction context, perhaps we wish to *predict* the reading speed of a user looking at tweets as a function of the font size used; we could build a Bayesian model to do this. But we might alternatively wish to determine which changes in font size and changes in typeface choice (e.g. serif and sans-serif) might best *explain* changes in reading speed observed from a large in-the-wild study. Modelling uncertainty in this task is critical, as is the ability to incorporate established models of written language comprehension. Bayesian methods excel at this.

1.3.3 How Would I Do These Computations?

We have so far spoken in very high-level terms about Bayesian models. How are these executed in practice? This is typically via some form of approximation (Figure 1.17).

Exact Methods

In some very special cases, we can directly compute posterior distributions in closed form. This typically restricts us to represent our models with very specific distribution types. Much of traditional Bayesian statistics is concerned with these methods, but except for the few cases where they can be exceptionally computationally efficient, they are too limiting for most interaction problems.

An Exact Example: Beta-Binomial Models

A classic example where exact inference is possible is a beta-binomial model, where we observe counts of binary outcomes (0 or 1) and want to estimate the distribution of the parameter that biases the outcomes to be zeros rather than 1s. If we *assume* that we can represent the distribution over this parameter using a *beta distribution* (a fairly flexible way of representing distributions over values bounded in the range $[0, 1]$), then we can write a prior as beta distribution,

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

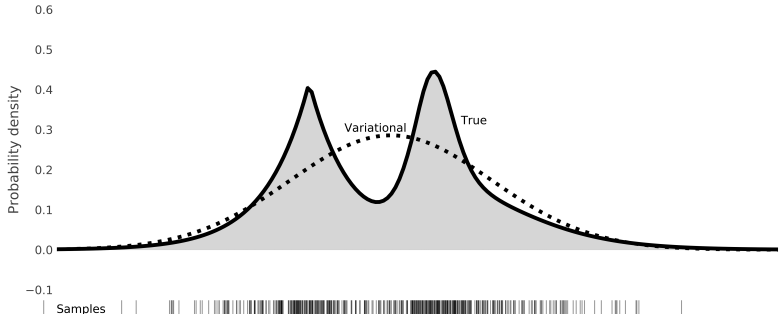

Figure 1.17 Most approximation methods represent complex distributions (solid line) using either Monte Carlo approaches which use random samples to approximate distributions (samples illustrated as vertical ticks in the lower strip), or variational methods which represent complex distributions with simple and easily paramaterised distributions optimised to best fit the true distribution (dashed curve).

observe some 0s and 1s and write a new posterior beta distribution down exactly, following some simple computations.

For example, we might model whether or not a user opens an app on their phone each morning. What can we say about the distribution of the *tendency* to open the app? For example, it is not reasonable to believe that a user will *never* open an app if they don't open it the first day, so we need a prior to regularise our computations. We can then make observations and compute posteriors exactly, as long as we are happy that a beta distribution is flexible enough to capture our belief. Because Bayesian updating moves just from one distribution to another, we can update these distributions in any order, in batches or singly, *assuming* that the observations are independent of each other.

Monte Carlo Approximation

The most promising and most general approach to Bayesian inference is **sample-based** approaches that sidestep manipulation of distributions by approximating them as collections of samples. To perform computations, we draw *random* samples from distributions, apply operations to the *samples* and then re-estimate statistics we are interested in. In Bayesian applications, we draw random samples from the posterior distribution to perform inference. This makes operations computationally trivial. Instead of working with tricky analytical solutions, we can select, summarise or transform samples just as we would ordinary tables of data. These methods normally operate by randomly sampling realisations from a distribution and are known as **Monte Carlo** approximations.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

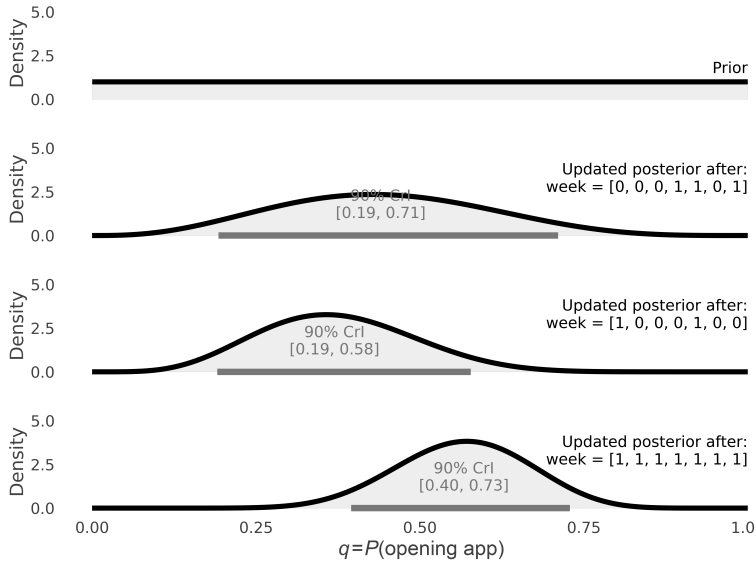

Figure 1.18 Beta-binomial exact inference. We want to model the propensity for a user to open an app on a given day. We can see the user activity as a process that has a bias q to produce a 0 (no open) over a 1 (open). If we think a beta distribution captures our uncertainty about this parameter q, we can exactly update the posterior distribution over q following batches of binary observations x. In each row, the distribution over q is shown after one new week of observations (right on each panel) is observed. Each distribution becomes the prior for the successive one beneath. '90% CrI', the 90% credible interval, (grey horizontal line) indicates a range of parameters where the propensity lies with 90% probability, given the priors and model we have chosen.

Markov chain Monte Carlo (MCMC) is a specific class of algorithms which can be used to obtain Monte Carlo approximations to distributions from which it is hard to sample. In particular, MCMC makes it easy to sample from the product of a prior and likelihood, and thus draw samples from the posterior distribution. MCMC sets up a 'process' that walks through the space of the distribution, making local steps to find new samples. There are many ways of implementing this, but under relatively weak assumptions this can be shown to *eventually* draw samples from any posterior.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Interested readers are invited to view the interactive gallery of MCMC algorithms by Feng [17] to get a deeper understanding of how MCMC sampling works in practice.

MCMC is very powerful and general, but there are a number of MCMC algorithms available, and each has its own parameters to tweak that affect the inference results. This is undesirable: our posterior distributions should depend only on the prior and the evidence we have, not on settings like the 'step size' in an MCMC algorithm. In practice, MCMC is often a bit like running a hot rod car: there's a lot of tuning to get smooth performance, and if you don't know what you are doing, it might blow up. There is an art to tuning a MCMC algorithm to make them tick over smoothly, and many diagnostics to verify that the sampling process is behaving itself.

Monte Carlo approaches generate samples from posterior distributions, but we often want to represent and report results in terms of distributions. This requires a conversion step back from samples into summaries of the approximated distributions. Common approaches to do this include histograms or kernel density estimates (e.g. for visualisation). Alternatively, summary statistics, such as means, medians or credible intervals, can be computed directly from the samples themselves. All MCMC methods have approximation error. This error reduces as the number of samples increases, but slowly (the Monte Carlo error decreases as $O(\sqrt{N})$, *assuming* the sampling is working correctly).

Variational Approximation

Variational methods approximate posteriors with distributions of a simple, constrained form which are easy to manipulate. The approximating distributions are optimised to best fit the posterior. One common approach, for example, is to represent the posterior with a normal distribution, which can be completely represented by a mean vector (location) and covariance matrix (scale/spread).

Variational approximations have benefits and drawbacks:

- They are typically *extremely* efficient. When they are applicable, they can be orders-of-magnitude faster than Monte Carlo approximations for the same level of accuracy.
- They often have relatively few parameters of their own to tweak, so less tuning and tweaking is required to get good results than is common in Monte Carlo approaches.
- They are relatively rigid in the posterior forms that can be represented. This depends on the approximation used, but, for example, a variational approximation with a normal distribution cannot represent multiple modes (peaks in the probability density), which may be important.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- Variational methods typically need to be specifically derived for a particular class of models. Most variational methods cannot simply be slotted into a probabilistic program and instead need expert skills to construct.

Some modern methods, such as automatic differentiation variational inference (**ADVI**), *can* be used without custom derivations and can be plugged into virtually any Bayesian inference models with continuous parameters. ADVI can be used in a wide range of modelling problems, but it has a limited ability to represent complex posteriors.

In interaction problems, variational methods are an excellent choice *if* an existing variational method is a good fit to the problem at hand *and* the form of posterior expected is compatible with the approximating distribution. They can be particularly valuable when low-latency response is required, for example, when embedded in the interaction loop.

Probabilistic Programming

A rapidly developing field for Bayesian inference is **probabilistic programming**, where we write down probabilistic models in an augmented programming language. This transforms modelling from a mysterious art of statisticians to an ordinary programming problem. In probabilistic programming languages, *random variables* are first class values that represent distributions rather than definite values. We can set priors for these variables and then 'expose' the program to observed data to infer distributions over variables. Inference becomes a matter of selecting an algorithm to run. This is often a MCMC based approach (e.g. in Stan [12]), but other tools allow variational methods to be plugged in as well (as in pymc3 [48]). Probabilistic programs can encode complex simulators and are easy and familiar for computer scientists to use. Probabilistic programming languages still need tuning and diagnostics of their underlying inference engines, but otherwise are plug-and-play inference machines. As an example, the following pymc3 code implements the model of reading time as a linear relationship.

This code implements the simple linear reading time example as a probabilistic program in pymc3

```
with pm.Model():
    # prior on slopes;
    # probably around 0, not much more
    # than 10-20 in magnitude
    b1 = pm.Normal(0, 10)

    # prior on constant reading time;
    # positive and probably
```

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

```
# less than 30-60 seconds
b0 = pm.HalfNormal(30.0)

# prior on measurement noise;
# positive and not likely to
# be much more than 10-20
measurement_noise = pm.HalfNormal(10.0)

# font_size is observed.
# We set ba uniform prior here to
# allow simulation without data
font_size = pm.Uniform("font_size", 2,
                       32, observed=font_size)

# estimated average reading
# time is a linear function
mean_read_time = b1 * font_size + b0

# and the reading time is observed
read_time = pm.Normal("read_time",
                      mu=mean_read_time,
                      sigma=measurement_noise,
                      observed=read_time)
```

Observing a table of pairs of the observed variables `read_time` and `font_size` would let us infer distributions over `b0` and `b1` and `measurement_noise` – more precisely, an MCMC sampler would draw a sequence of random samples approximately from the posterior distribution, and return a table of posterior parameter distributions. This sample sequence from an MCMC process is known as a **trace**. Traces can be visualised directly or represented via summary statistics.

1.4 Can You Give Me an Example?

Let's work through a worked example of Bayesian analysis. We'll examine a problem familiar to many interaction designers: Fitts' law [19]. This 'law' is an approximate model of pointing behaviour that predicts time to acquire a target as a function of how wide a target is and how far away it is (Figure 1.19). It is a well-established model in the HCI literature [39].

1.4.1 Model

The Fitts' law model is often stated in the form:

$$MT = a + b \log_2 \left(\frac{d}{w} + 1 \right).$$

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

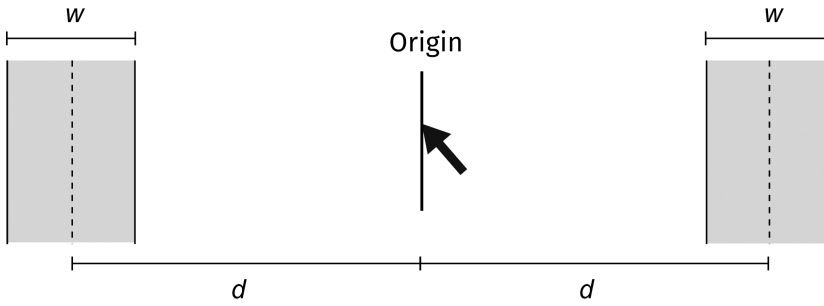

Figure 1.19 Fitts' law pointing task. The distance d and width w determine time to acquire the target. However, there are unknown parameters a and b that parameterise this relationship for different input devices.

This tells us that we predict that MT (the movement time to acquire a target) will be determined by the logarithm of the ratio of the target distance d and target size w. This is a crude but surprisingly robust predictive model. The two parameters a and b are constants that vary according to the input device used. In statistical terminology, this is a generalised linear model with a log link function. It can be easier to see the linear nature of the model by writing $ID = \log_2(\frac{d}{w} + 1)$ and the model is then just $MT = a + bID$ – i.e. a straight-line relationship between MT and ID defined by a, b. The term ID is often given in units of *bits*; the justification for doing so comes from information theory. A higher ID indicates a larger space of distinguishable targets, and thus more information communicated by a pointing action.

How might we approach modelling a new pointing device in a Bayesian manner? Let's assume we run an experiment with various settings of ID (by asking users to select targets with some preset distances and sizes). This fixes ID; it is an independent variable. We measure MT, the dependent variable. We are therefore interested in modelling the latent parameters a and b, which we cannot observe directly. We *know* that our measurements are noisy. Running the same trial with the same ID will not give the exact same MT. So we must model the expected noise, which we will notate as ϵ. Perhaps we expect it to be normally distributed, and we can write our model down:

$$MT = a + bID + \epsilon,$$

$$MT = a + bID + \mathcal{N}(0, \sigma^2).$$

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

The notation

$$\mathcal{N}(0, \sigma^2)$$

indicates normally distributed random noise with a standard deviation of σ. Its presence indicates that even if we knew a and b and ID, there would be random variation in MT – and we are assuming that this is normally distributed with some scale σ.

We don't know what σ is, so it becomes another latent parameter to infer. Unlike in, say, least square regression, we don't *have* to assume that our noise is normally distributed, but it is a reasonable and simple assumption for this problem. For a justification, see Section A.3.1 in the Appendix, subsection on the normal distribution.

In code, our generative model is something like:

```
class FittsSimulator:

    def __init__(self, a, b):
        self.a, self.b = a, b


    def simulate(self, n, d, w):
        # compute ID
        ID = np.log2(d / w + 1)

        # generate random samples
        mu = a + b * ID
        return scipy.stats.norm(mu, sigma).rvs(n)

    def log_likelihood(self, ds, ws, mts):
        # compute IDs
        IDs = np.log2(ds / ws + 1)
        mu = a + b * IDs

        # compute how likely these movement times
        # given a collection of matching d, w pairs
        return np.sum(scipy.stats.norm(mu, sigma).logpdf(mts))
```

Priors

To do Bayesian inference, we must set priors on our latent parameters. These represent what we believe about the world. Let's measure MT in seconds, and ID in bits, to give us units to work with. Now we can assume some priors on a, b and σ. Reviewing our variables:

- ***MT*** is the dependent variable and is observed.
- ***ID*** is the independent variable and is also observed.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- **a** is the 'offset' and is unobserved. We might assume it has a normal prior distribution, perhaps mean 0, standard deviation 1.
- **b** is the 'slope' and is unobserved. We might again assume a normal prior distribution, perhaps mean 0, standard deviation 2.
- σ is the noise level and is also unobserved. Here, we need a positive value. We might choose a 'half-normal' with standard deviation 1.

These priors are *weakly informative*. These are our conservative rough guesses as plausible values (it is not likely that we have a 3-second constant offset a, but it's not *impossible*). There is nothing special about this choice of normal distributions. It is simply a convenient way to encode our rough initial belief.

A common question:

- Q: Did we not just **choose** the answer? Couldn't we set the prior to whatever we want to get the answer we want to see?
- A: **No, we did not**, and this argument is ill-founded. We *could* write down a prior that specified the answer. For example, we could set a prior that puts all probability density on the possibility b = 0.0. This is possible, but obviously the evidence will never change this belief. It is equivalent to logical reasoning that started with the *axiom* 'all apples are red', then followed a process of reasoning. The final result would be 'apples are red', because we *assumed* that to start the reasoning process! Likewise, a prior specifies our assumptions explicitly. This is both a reasonable thing to do, and a valuable one – it requires us to be explicit in stating our assumptions, and in a form that we can then *test and inspect*, with ideas like prior predictive checks.

Prior Predictive Checks

What do these priors imply? One major advantage of a Bayesian model is that we can draw samples from the prior and see if they look plausible. It's most useful to see these as the lines MT, ID space that a, b imply, even though we are sampling from a, b, σ. Transforming from the prior distribution over parameters to the observed variables gives us *prior predictive* samples. We can see that the prior chosen can represent many lines, a much more diverse set than what we are likely to encounter (Figure 1.20), and can conclude that our priors are not unreasonably restrictive. Here we are just eyeballing the visualisations as a basic check; in other situations, we might compute summary statistics and

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

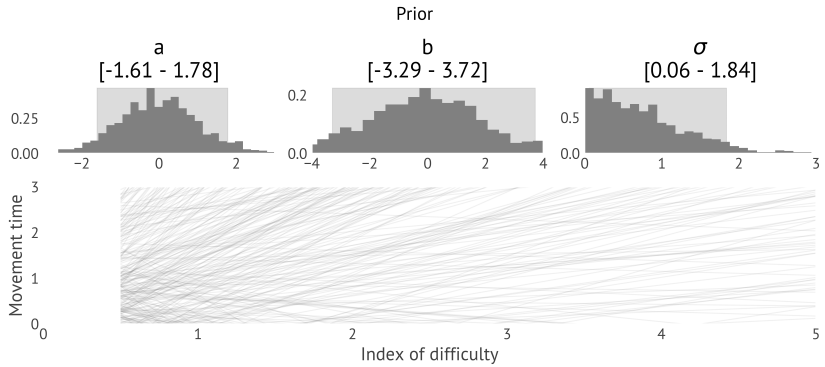

Figure 1.20 Prior predictive visualisation for the priors we set above. At the top, histograms showing the distribution of the hidden parameters; numbers indicate the (centred) 90% credible interval – a region in which we believe the true parameter is 90% likely to lie within. Below are those parameters used to draw possible lines compatible with the prior model. We see that there are very many linear models compatible with our model. Our prior distributions are at least flexible enough to be compatible with our genuine prior beliefs.

validate them numerically (e.g. testing that known positive values are positive in the prior predictive).

Inference

Now imagine we run a pointing experiment with users and capture MT, ID pairs and that are plotted in Figure 1.21.

Our model outputs the likelihood of seeing a set of MT, ID pairs for any possible a, b, σ. **Note: our model does not predict a, b, σ given MT, ID, but tells us how likely an MT, ID pair is under a setting of a, b, σ!** An inference engine can approximate the posterior distribution following the observations. Figure 1.22 shows how the posterior and posterior predictive change as more observations are made (typically, we'd only visualise the posterior after observing all the data, in this case $6 \times 18 = 108$ data points). The posterior distribution contracts as additional data points constrain the possible hypotheses.

Analysis

What is the value of a and b for this input device? A Bayesian analysis gives us distributions, not numbers; we can summarise these distributions with statistics. After the $N = 108$ observations, the 90% credible intervals are $a = [-0.05, 0.13]$ seconds and $b = [0.5, 0.57]$ bits/second. What about σ? The 90% CrI is [0.25, 0.31] seconds. This gives us a sense of how noisy our predictions are;

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

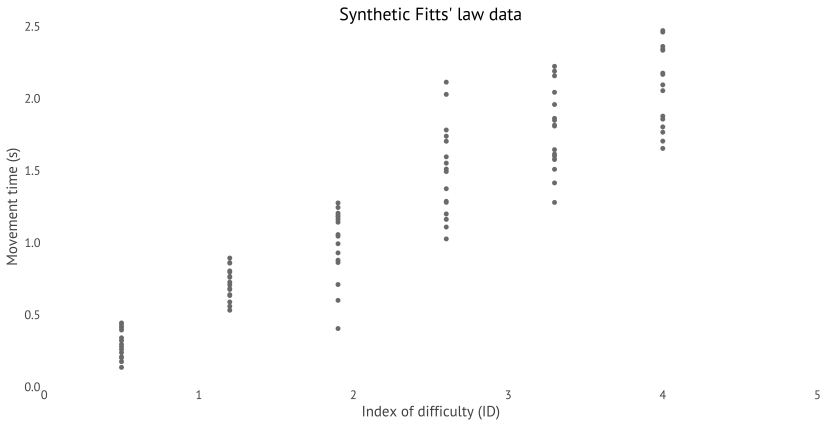

Figure 1.21 The raw data for the Fitts' law problem – 18 replicates of 6 different values for ID and corresponding movement times. All of these data are synthetic.

small σ indicates a clear relationship; big σ indicates weak relationship. What we *cannot* do from this is separate aleatoric measurement noise (e.g. human variability) from epistemic modelling noise (e.g. perhaps Fitts' law is too crude to model the motions we see).

Alternative priors What if we had chosen weaker priors? The inference is essentially unchanged even if we use very broad priors, as in Figure 1.23. If we had reason to choose tighter priors, perhaps being informed by other studies, we'd also get very similar results, as shown in Figure 1.24. Note the effect on the predictions when we use only 5 data points – we have much more realistic fits with the stronger priors in the small data case. It's important to note that these are alternative hypotheses we might have made *before* we observed the data. If we adjust priors after seeing the results of inference, the inference may be polluted by this 'unnatural foresight'. P-hacking-like approaches where priors are iteratively adjusted to falsely construct a posterior are just as possible in Bayesian inference as in frequentist approaches, although perhaps easier to detect. Alternative priors could be postulated *if* they arose from external independent knowledge; e.g. another expert in Fitts' law suggests some more realistic bounds.

A New Dataset

Perhaps we observe another dataset. In this case we have 40 MT, ID measurements from an in-the-wild, unstructured capture from an unknown

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

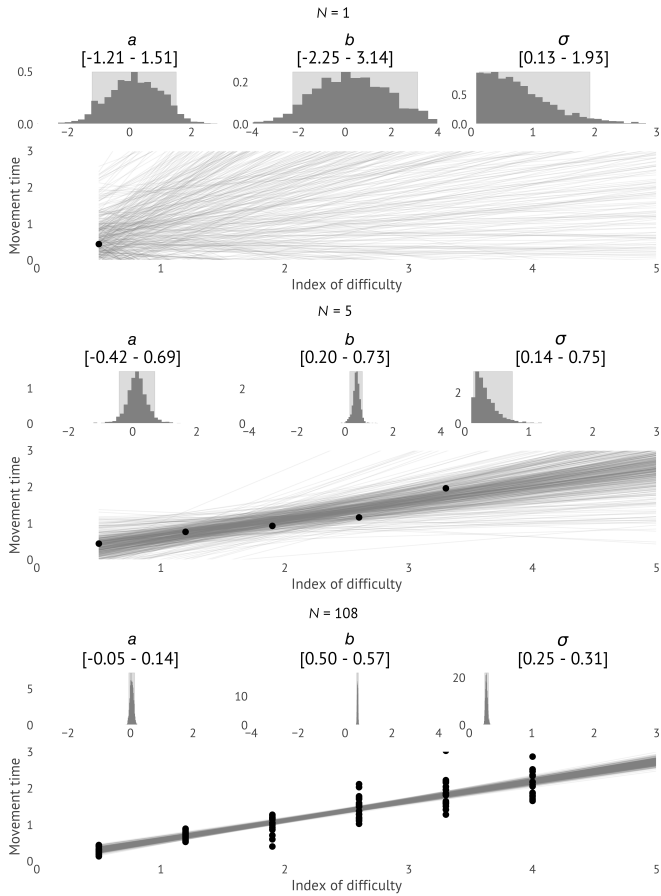

Figure 1.22 Bayesian inference for the Fitts' law task as data are acquired; N indicates the number of data points included. The space of plausible models contracts as more data points are included. Each block shows histograms for a, b, σ (the posterior) as well as the posterior predictive (the lines in the MT, ID space.) Shaded areas of the histogram, and the numbers $[a, b]$ above indicate centred 90% credible intervals.

pointing device (Figure 1.25). **How likely is it that the *b* parameter is different in this dataset?**

This question might be a suitable proxy for whether these 40 measurements are from the same pointing device or distinct pointing device. We can fit our model to these data (independently of the first model) and then compute the distribution of $b_1 - b_2$, the change in b across the two datasets (b_1 being the

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

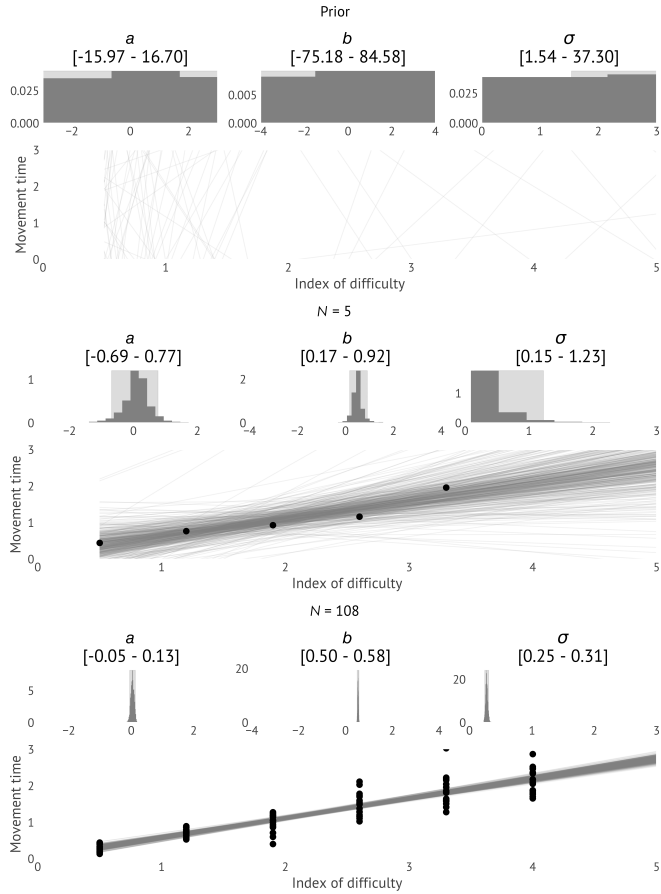

Figure 1.23 Much broader priors have essentially no effect on the inference with the full dataset ($N = 108$, lower), though have higher uncertainty if we only observe a few data points ($N = 5$, upper).

original and b_2 the new, in-the-wild dataset). This gives us a distribution (Figure 1.26), from which we can be *relatively* confident that the b value is different, and we are probably dealing with data collected from another device.

Since we have a predictive model, we can easily compute derived values. For example, we could ask the concrete question: how much longer would it take to select a width 2 distance 5 target using this second device than the first device. We can push this through our model and obtain the distribution shown in Figure 1.27.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

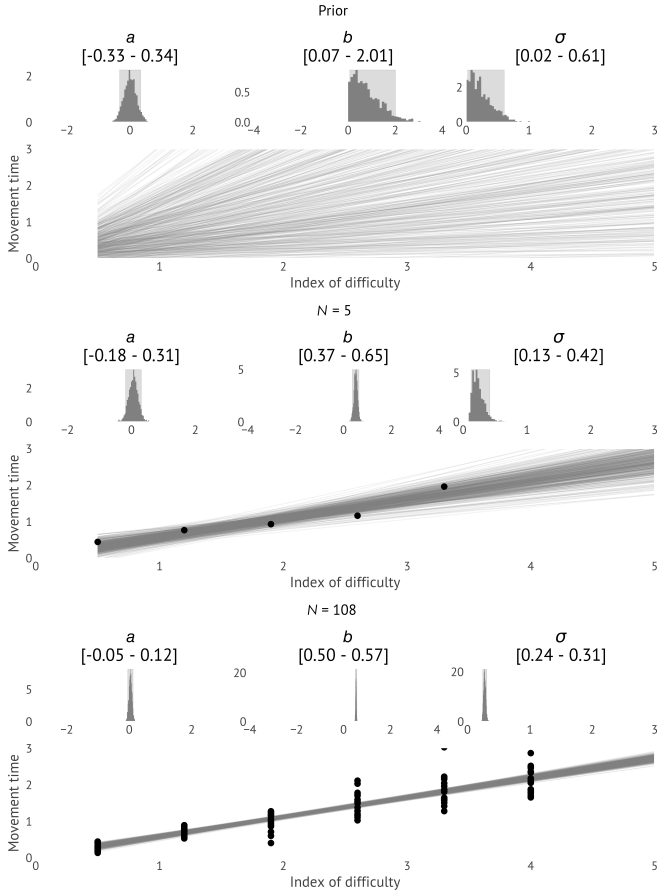

Figure 1.24 Tighter priors also have little effect with the full dataset ($N = 108$, lower), but the informed priors constrains the belief more effectively when there are only a few data points ($N = 5$, upper).

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

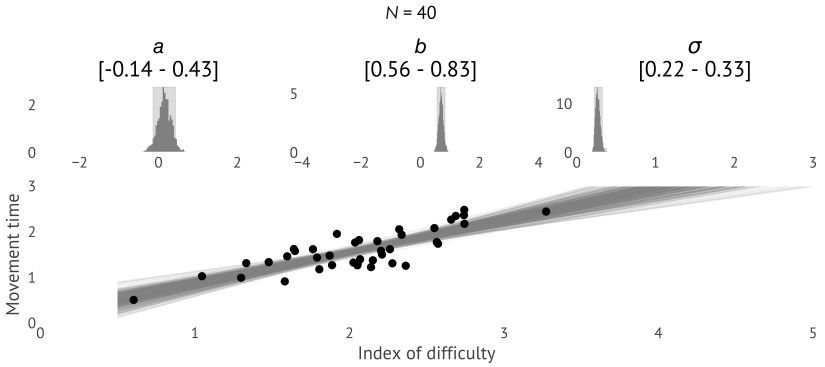

Figure 1.25 A new dataset, from uncontrolled observational studies of an unknown pointing device. We can refit the Bayesian model and estimate the parameters as before.

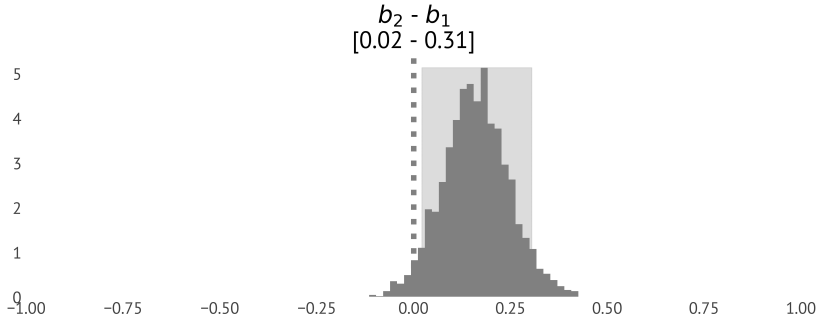

Figure 1.26 The change in distribution of b going from the posterior fitted on the original data to the posterior on the new data. The 90% CrI does not overlap 0, but it is close. So it is likely that there is a real difference in b, but the evidence is relatively weak.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

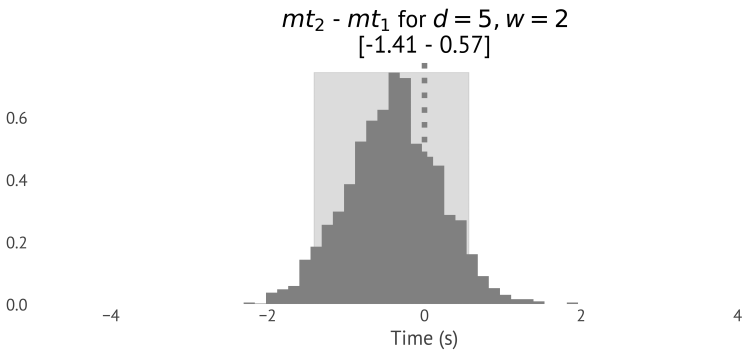

Figure 1.27 The predicted change in movement time to acquire a width 2 target 5 units away when switching from the first pointing device to the second. We expect the second device to take about 1 second longer to acquire this target, but there is reasonably large uncertainty.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

What's the Point?

Why did we do this? What benefits did a Bayesian approach give us?

- **Uncertainty:** For one, we have realistic and useful uncertainty in our parameters. The credible intervals for our parameters are easy to interpret (there *is* a 90% chance that a lies in $[-0.05, 0.13]$, given our model, priors and observations).
- **Priors:** It is very easy for us to incorporate domain knowledge. If we had done studies before, we might have justified tighter priors. Even without other Fitts' law studies, we could have formed more informative priors than we did based on basic scientific knowledge; we'd expect that b has to be positive, and we'd also expect that the human motor system cannot generate more than 30 bits/second. We can *directly* use this information in the inference. Or perhaps another researcher is sceptical of these priors and prefers to be less cautious. We can re-run the inference and get new results; we'll find in this case that the priors have very little effect on the results with this much data.
- **Flexibility of modelling:** One advantage is that it is simple to write alternative models. When writing such models in a probabilistic programming language, the following modifications are one or two lines of extra code:
  - We might assume that noise is actually Gamma distributed (or any other noise model) so $MT = a + bID + \Gamma(\alpha, \beta)$. We'd just put priors on α and β and run the inference again.
  - We might assume there is some small quadratic term, perhaps $MT = a + bID + cID^2 + \mathcal{N}(0, \sigma)$. We'd put a prior on c that would suppose it to be small, because we know the relationship is *roughly* linear, and re-run the inference.
  - Perhaps we assume that Fitts' law holds for between some range of IDs, but becomes increasingly linear in distance after targets get a certain distance away. We can write this as a model and infer the unknown crossover point between Fitts' and linear behaviour: $MT = a + bID + \max(k(w - m), 0) + \mathcal{N}(0, \sigma)$, adding two new parameters k and m
  - We might instead assume that the parameters vary *per participant*, so $MT = a_i + b_i ID + \sigma_i$, and infer the parameter vectors $a_1, a_2, \ldots$. In this case, the user ID i is observed.
  - As a more sophisticated approach, we might assume that parameters vary per participant, but that the participants' parameters come from some common distribution (that generates a_i, b_i, σ_i.). This is a 'partially pooled' model and encodes our belief that humans vary but have similar characteristics, and can be a powerful way to efficiently model populations.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- **Relevant hypotheses:** There is a distinct absence of statements about null hypotheses. These *might* be relevant, and we could analyse the data with frequentist methods to answer them. But they probably aren't what really interests us in this specific problem. Instead, we have relative likelihoods of hypotheses that *do* matter. For example, if we are interested in comparing our two input devices, we can make statements like 'acquiring a $(w = 2, d = 5)$ target has a 90% chance of taking [0.57, 1.41] seconds longer when using the second device'. Compare this to the statement 'if we repeatedly ran this exact experiment, there is a less than 5% chance that we'd see differences equal to or bigger than this if the differences were purely random'.

Is This Generative Modelling?

Fitts' law isn't a particularly *generative* way to think about pointing motions. Fitts' law describes the data, but it is not a strong explanation of underlying causes. A more sophisticated model might, for example, simulate the pointer trajectories observed during pointing. We could, for example, infer the parameters of a controller we suppose is approximating how humans acquire targets, generating spatial trajectories. Bayesian inference could be applied, but now we would be able to make richer predictions about pointing (for example, predicting error rates instead of just time to acquire, or properly accounting for very close or very distant targets).

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

What is the difference between generative and descriptive modelling? These distinctions lie on a spectrum between 'what happens' and 'why does this happen', and there is no shining line that divides them. Consider an example:

- Observation: The cat meows around 22:30 each night.
- To build a **descriptive model**, we could measure the time of each meow on a sequence of nights, and build a model by estimating a distribution giving probability of a meow given clock time. This would give us some ability to predict meowing episodes in the future. It describes the observations statistically and can be used for prediction, but it is a weak *explanation*.
- A more **generative** model would be built using expert knowledge to extract causal factors. The cat meows because it is hungry and it anticipates treats. Treats are administered when the humans go to bed. The humans go to bed around 22:30. We can now build a more detailed causal model that links $clock \Longrightarrow bedtime \Longrightarrow anticipation \Longrightarrow meow \Longleftarrow hunger$. This might not make better predictions of the next night than a descriptive model, but it does give us insight in counterfactual scenarios. If the hour changes due to daylight saving time, we'd expect the meowing to follow, because humans use clock time to schedule their lives. If the cat is well fed before bedtime, the meowing will be suppressed. If the cat is alone, it won't meow.

1.4.2 Bayesian Workflows

This worked example outlined the main steps in Bayesian modelling for this example. In general, how should we go about building Bayesian models in an interactive systems context? What do we need to define? How do we know if we have been successful? How do we communicate results? *Workflows* for Bayesian modelling are an active area of research [25, 49]. A high-level summary of the general process is as follows:

- **Define a model.** Ideally, we want a *generative* model that describes how we believe the world works, governed by a set of *parameters* that can vary. For example, say we were modelling the effect of frame rate variation on VR sickness. We would want to create a model that when fed a sequence of frame timings would output a probability of induced nausea. It would be possible but less desirable to use a general model like logistic regression. A better model

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

would be constructed based on psychological and physiological models with parameters in meaningful units that represents our best knowledge of how the process truly works.

- **Construct priors.** We need to specify priors for every parameter in the model. We seek to find priors that are as informative as possible (and thus give us the most precise inferences with the least data) without introducing undue bias. Priors should be elicited from expert knowledge; from previous studies or published literature; or, if nothing else, informed guesses with clear justifications. Priors should **not** be established by looking at the specific data under consideration! 'Fitting' priors to data and then running inference with the same data inevitably biases the results.
- **Test priors.** Bayesian models are executable and we can sample possible simulated observations before doing any inference – sampling from the *prior predictive*. We can and must check that the priors look sensible. This can range from a quick eyeballing of histograms to more in-depth analysis of summary statistics of the prior simulations. Verifying prior predictives with domain experts can save a great deal of pain later. These simulations are most valuable if the model is easy to interpret generatively. Simulating from the prior also lets us test the rest of the analysis pipeline *in advance of observing any data* and verify that visualisations and summary statistics will be computed as we want.
- **Fit model.** With our model and priors set, and our data acquired, we can perform inference. This will eventually produce an estimate of the posterior distribution, often in the form of a **trace** from an MCMC process – a sequence of samples approximately drawn from the posterior.
- **Tune and diagnose.** The inference process can go wrong. Poorly specified priors or models with awkward parameterisations can make inference engines go haywire. This may result in bad approximations or excessive computation time. Inference engines themselves have many settings that can be adjusted that affect the approximation process. It is essential to perform at least some diagnostics to verify that posteriors are being approximated accurately. Many standard packages will compute basic diagnostics automatically. Any problematic results will need to be corrected and re-run until diagnostics 'run clean'.
- **Test the posterior predictive.** Posterior distributions should be run through the generative model to generate synthetic observations from the *posterior predictive*. These should indicate a close match between the true observations and the generated samples from the fitted model; certainly better than samples from the prior predictive. This might be established by comparing expectations (like the mean and standard deviation) of the posterior predictive

against the observations, or verifying that the observations would lie in sensible quantiles of the posterior predictive (e.g. not way out in the tails). More thorough testing might involve re-fitting the model, using the posterior predictive as synthetic observations, and verifying that the posterior distributions are not substantially altered.

- **Visualise and form summary statistics for the posteriors (and posterior predictive).** The posterior distributions need to be visualised in some way to communicate an overall picture of the results. This is often challenging, because a model with many parameters can have many joint interactions that cannot all be seen. Reporting and visualisation should represent all uncertainty. This could range from showing a blurred contour onscreen (in an online probabilistic tracking scenario) to rendering histograms, Box plots, cumulative distribution functions, dot quantile plots [18] or violin plots (in a paper reporting on an empirical study). If necessary, summary statistics like means and standard deviations or credible intervals can be used to compress results into a few numbers. If decisions need to be made, then utility functions need to be defined and then combined with posterior distributions.

This workflow is presented from the perspective of performing an empirical analysis. The principles transfer to other uses of Bayesian models in interaction. For example, if we were building a probabilistic filter to track a user's head orientation, we would:

- build a model that predicted head dynamics (e.g. based on biophysics);
- define priors over likely poses;
- test to confirm that these did not look silly (e.g. by rendering them as head poses of a computer graphics character);
- estimate the model online from sensor inputs (e.g. using a particle filter);
- use diagnostics to make sure the particle filter is tracking (e.g. effective sample size);
- render a point cloud visualisation of the posterior distribution for the user.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

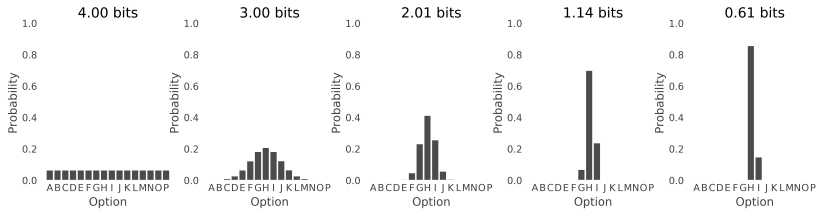

Figure 1.28 A sequence of distributions being updated with evidence, each distribution becoming the prior in the next round. There are 16 possible choices, and initially all are equally likely (an entropy of 4 bits). As information is acquired, the entropy drops towards zero.

1.5 Topics of Special Relevance to Interaction Design

1.5.1 The Relation to Information Theory

Interaction can be seen as the *flow of information* between entities. In human–computer interaction, for example, information flows from users to systems to indicate intent, and information flows back via the display. **Information theory**, as pioneered by Shannon [52], is closely linked to probability theory and integrates cleanly with Bayesian approaches. In particular, we can measure, mathematically, the information required to change one distribution into another. This corresponds directly to how much information we need to pass through a communication channel, such as a human–computer interface, to specify a new distribution.

The key concept is that of *entropy*, the measure of uncertainty in a distribution, sometimes characterised as a measure of the 'surprise' samples from a distribution would have. Entropy is a single number that quantifies how uncertain a distribution is. It is often measured in units of *bits* and tells us how much additional information must be provided to uniquely determine the outcome from a distribution. For example, a distribution with an entropy of 4.3 bits requires knowledge of a little more than four definite yes-or-no questions to completely identify its value. A distribution with zero bits of entropy concentrates all probability mass on a single outcome, so there is no surprise and no additional information needed to resolve the value.

Entropy is less straightforward when dealing with distributions over continuous values. Instead of an absolute measure of entropy, we talk about the *relative entropy*: the information required to move from one distribution to another, which is also called the *Kullback–Leibler* (KL) divergence.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

Entropy is essential in determining how much information must be communicated through a channel to identify (select, in an interaction context) a specific outcome. When we perform a Bayesian update, we will move from a prior to a posterior in light of evidence. If the evidence has constrained the space of possibilities – that is, we have learned something from it – then we will have a precisely quantifiable reduction in entropy as a consequence. Interaction can be seen as a sequential update of probabilities to reduce a system's entropy about intended actions, as in Figure 1.28.

For example, in a pointing task, like operating a calculator app, we might have space divided into a 4×4 grid of buttons. Pressing one of the calculator's buttons selects one of 16 options. If we wish do so without error, this necessarily communicates 4 bits of information, as $\log_2(16) = 4$. Whatever input we use, we need 4 bits of information to unambiguously choose an option. But this information does not have to come from the same source. If we *know* that the + key is pushed much more often than the $\sqrt{}$ key, then we have pre-existing information. This prior belief would *reduce* the information required to operate the calculator by pointing, because there are effectively fewer options – less information is required because the selection is in a sense already part-way completed. We can represent more commonly used keys with fewer bits and less frequently used keys with more bits. We could, for example, permit sloppier pointing without increasing the error rate by interpreting pointing actions differently (e.g. by varying the effective size of the buttons). This is a process of decoding intent from uncertain input.

In the scenario of a user-system interaction, we can view the user as 'bit-store' of state, which encodes an intention with respect to the system (for example, 'please cancel this calendar appointment'). The user has to squeeze this intention through the communication channel of the interface to contract the distribution the system has over possible actions so that specific state change happens. Questions about how much information has to flow, and how quickly a decision is being made, are most naturally framed in terms of entropy (Figure 1.29).

As a concrete example, a pointing device which follows Fitts' law [19] might generate a maximum k bits/second for a given user, pointing device and interface layout. If there are n options, with equal prior probability, it will take at least $\lceil (\log_2 n)/k \rceil$ pointing actions to reliably acquire a target. If there is a prior distribution over targets with entropy h, then it takes at least $\lceil h/k \rceil$ pointing actions if we somehow interpret pointing actions more efficiently.

One of the earliest foundational papers incorporating Bayesian methods into an interaction loop is Dasher [56], a text entry system that directly links probabilistic language models to a dynamic target layout. Dasher implements

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

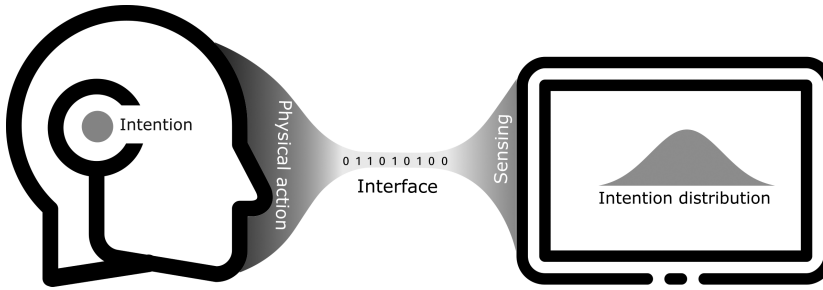

Figure 1.29 A human–computer interface is limited in terms of how quickly information can flow from a user to a system to reduce the entropy of the system's belief distribution. Modelling entropy is essential in understanding the limitations of an interaction method.

an elegant link between information–theoretic approaches and the problem of optimal selection via 1D pointing.

1.5.2 What Is 'Approximate Bayesian Computation'?

Approximate Bayesian Computation (ABC) is a **likelihood-free** way of performing inference. It is useful in the case where we have a simulator, but there is no likelihood 'inlet' – no way of directly computing the likelihood of an observation given a parameter setting (imagine the `Simulator` class from earlier with the `likelihood` method deleted).

Instead, ABC approaches synthesise samples under different parameter configurations and compare these synthetic samples with observations to update distributions over parameters; in the simplest case, just rejecting parameter settings that result in simulation runs too different from the real observations. This approximation comes at a significant cost in terms of computational resources (large numbers of synthetic samples are needed) and inferential power (it is harder to infer parameters reliably). The huge advantage is that if we *only* have a simulator that can generate samples, even if it is not or could not be written in a probabilistic manner, then we can still perform Bayesian inference with it. This means that we can, for example, retro-fit 'legacy' simulators that know nothing of likelihood. Alternatively, we can build Bayesian models for problems when it is conceptually challenging to even define what a likelihood function would look like.

For example, we might have a simulator that can generate likely arm trajectories in a target acquisition task, based on a biomechanical simulation of muscle activations. Given some arm trajectories from a motion tracker, and some

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

priors over muscle activation patterns, how can we get a posterior distribution over muscle activations? The ABC approach would involve simulating many synthetic arm trajectories given the prior over muscle activations, selecting or weighting those samples that are close to the observed trajectories, and updating the distribution using the corresponding, *known* muscle activations that go with each synthetic trajectory. By averaging over many examples this can be used to infer an approximate posterior.

1.5.3 How Do 'Bayesian Networks' Relate?

Bayesian networks, **Bayes nets**, or **belief networks** are ways of representing relationships between variables in a probabilistic model. They are a compact way of representing and managing uncertainty and have many applications in user interfaces. In most interaction contexts, Bayes nets are used to model relationships between discrete variables, as in the example below with binary outcomes. Models are represented as directed acyclic graphs (DAG) which specifies dependencies between variables. Variables are represented as nodes, and dependencies as edges. Variables may be observed or unobserved (latent). This representation makes it easy to factor the model into independent elements, and the directionality of edges captures the causal relation between variables. The relationship between variables is captured by *conditional probability tables* (CPTs) that specify distributions for outcomes of child variables given all possible states of their parents. There are various implementation strategies to efficiently encode conditional probability tables to avoid exhaustive specification of every possible combination.

Inference is a process of updating the distributions on unknown variables when some variables are known. In small networks, with discrete nodes, this can often be done exactly. Approximations such as Monte Carlo methods can be applied for more complex models.

The example in Figure 1.30 shows a simple Bayes net with Boolean-valued variables (binary outcomes) that models focus change in a desktop user interface, its effect on user frustration and the effect of this frustration on heart rate and the probability of a user making an immediate error in typing. Focus changes can be induced either by the Alt-Tab hotkey or by a dialog stealing focus. Changes in focus affect frustration depending on their source. Changes in frustration can increase heart rate and/or make typing errors more likely. Given observations of *some* of these variables, and the DAG and conditional probability tables, we can answer questions like:

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- What is the probability the user is annoyed, given that focus changed and heart rate increased? $P(UA|FC, HR) \approx 0.56$.
- What is the probability of a heart rate increase, given Alt-Tab was pressed and focus did *not* change? $P(HR|AT, \overline{FC}) \approx 0.18$.
- What is the probability that Alt-Tab was pressed, given that we did not observe a heart rate increase? $P(AT|\overline{HR}) \approx 0.38$.
- What is the probability that there was a typing error, given that we observed a heart rate increase? $P(TE|HR) \approx 0.16$.
- What is the probability that there was a typing error but no change in heart rate, given that a dialog did not steal focus? $P(TE, \overline{HR}|\overline{DF}) \approx 0.046$.

It is important to realise that the directions of the arrows specify causal relations. The model describes the probability distribution of variables as consequences of the states of their parents. Inference about the state of variables can progress in *either* direction.

Bayesian networks have a long history in human–computer interaction. As well as inferring specific probabilities in a network, it is also possible to *learn* conditional probability tables from observations and thus 'fit' a belief network with a given graph structure to observations. For example, in the scenario above we might log focus change events, heart rate and other factors in a set of user trials, and then use the event co-occurrences to update the conditional probability tables. This can be done in a Bayesian manner by setting priors on the CPTs and conducting ordinary Bayesian inference. In certain cases it is further possible, though computationally challenging, to infer the structure of the networks themselves from observations, i.e. to learn the graph structure as well as the conditional probability tables.

Note: Somewhat confusingly, Bayes nets are probabilistic models, but *not* necessarily Bayesian in the sense we are using in this chapter. It *is* possible to do Bayesian inference on Bayes nets, but many applications of Bayes nets do not do so and use standard frequentist estimation. However, as probabilistic models with wide application in interaction design, it makes sense to include them here.

Belief networks can be extended to model sequences of observations over time, **dynamic belief networks** (DBNs). This includes models like the Hidden Markov Model (HMM), traditionally used in speech and gesture recognition. These models have dependency graphs that include state at the previous time step as parents and are powerful in modelling sequential processes. Hidden Markov Models, for example, are used to infer unobserved sequences of discrete

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

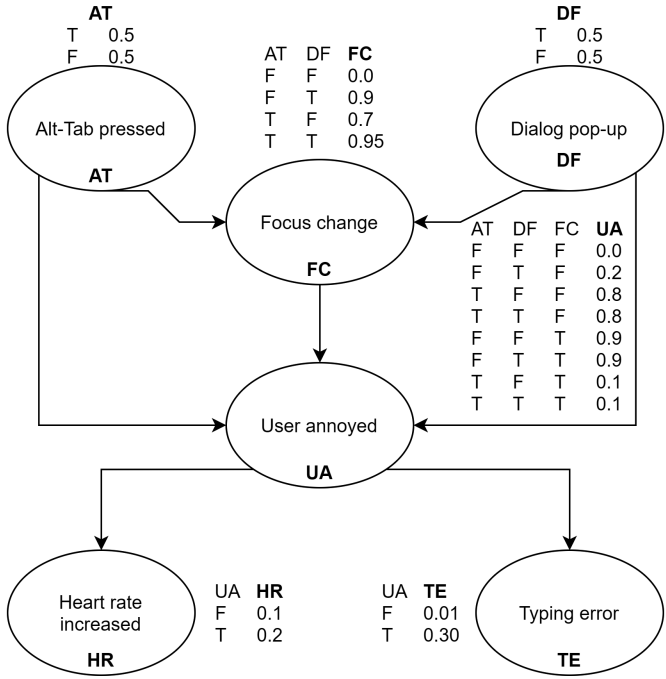

Figure 1.30 An example Bayes net in an interaction context. In this case, all variables (ellipses) are Boolean and have possible outcomes True (T) or False (F). Arrows indicate causal relations between variables. Conditional probability tables (text next to ellipses) show the probability of a variable taking on the True state (right column) given all possible configurations of its immediate parents (left columns). This simple Bayes net models the relationship between focus changes in a window manager and user frustration.

states that are believed to be 'causing' observations. An HMM for speech recognition might be used to infer a sequence of phonemes (unobserved states) from a sequence of acoustic features (observations), where the underlying model is that an phoneme sequence (i.e. spoken language) is being generated by a human speaker and 'causing' the acoustic observations. The HMM can then be used to decode a probability distribution over possible phoneme sequences; this can be combined with a probabilistic language model to further refine the recognition process. Dynamic belief nets are closely related to **probabilistic filtering**, the online (i.e. inference during a process) estimation of states. Probabilistic filters encompass DBNs, but the probabilistic filtering approach is typically identified with problems with continuous multi-dimensional

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

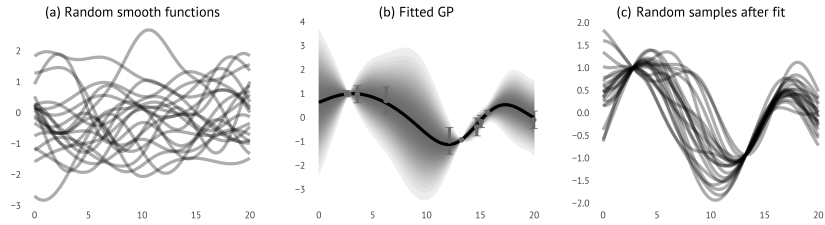

Figure 1.31 Gaussian process models form probability distributions over functions themselves. (a) Random functions drawn from a distribution over functions with a particular smoothness. (b) Observations have constrained the distribution, but note that the uncertainty is preserved (shaded area) and measurement uncertainty on each point (error bars) is taken into account. (c) Random samples from this distribution over functions compatible with the observations are shown.

unobserved states; whereas DBN approaches like Hidden Markov Models are typically applied in problems with discrete unobserved states.

1.5.4 What about 'Bayesian Non-Parametrics'?

We have presumed, so far, that our models have a fixed set of parameters that define a configuration – a few moving parts that can be adjusted. Bayesian non-parametric methods do not assume a parametric form, and instead form distributions over *possible functions* that could have generated data. These models are constructed by defining a class of functions, such as a particular space of functions of variable smoothness, and then forming a prior distribution over all possible functions of this class. This prior is updated with observations to produce a new distribution over functions which are compatible with the data. In the simplest case, this might be distribution over all functions which pass through some data points, a distribution over interpolating functions of a specific smoothness.

The most important of these approaches is the Gaussian process (GP), an exceptionally flexible modelling tool. The details of the GP are beyond this book, but it allows the definition of a space of functions via *kernels* that define how nearby observations co-vary; this becomes a restriction on the smoothness of functions. GPs are a powerful way to interpolate and extrapolate functions from observed samples (Figure 1.31), with appropriate uncertainty, and have a huge range of uses in interaction design.

In the simplest cases, non-parametric models like GPs can be used as smooth interpolators which maintain uncertainty, for example, to predict expected

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

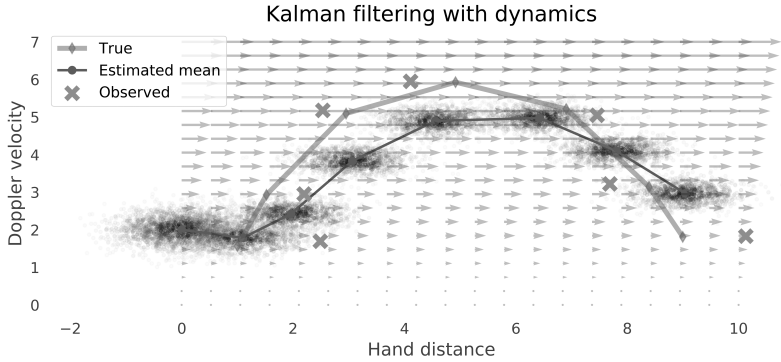

Figure 1.32 Probabilistic filtering in a hand tracking problem. We estimate a distribution over the distance and velocity of a hand, which are measured separately by a sensor. Noisy observations update the inference. Dynamics (arrows showing vector field) transform posterior distributions (shown as point clouds) to form the priors at the next step. Even with heavy noise in the position estimate, the dynamic model can make reliable predictions (posterior mean shown as a solid line).

offsets between actual touch and intended touch [11]. One important use in interaction design is as **proxy objective functions** in Bayesian optimisation. GPs are often used to represent an unknown function mapping properties of an interface to some quantitative measure, such as reported satisfaction or response time. By sequentially updating the distribution over functions, optimisation can be performed *at the same time* as learning about the function. This can be an efficient way to optimise interface designs with humans in the loop.

1.5.5 What Are Probabilistic Filters?

Probabilistic filtering is sequential Bayesian inference and is used to estimate parameters that vary over time. This is particularly salient in interaction problems where we often have an ongoing interaction *process* and want to infer states as they are happening.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

This means we move from a prior to a posterior on a series of time steps, at each step having an estimate of some unknown state. Probabilistic filters are of wide use in the interaction loop, particularly in problems like estimating a stable cursor from noisy sensing, or fusing together multiple sensors, perhaps running at different sampling rates. For example, we might be tracking the distance and movement of a user's hand to a mobile device screen, based on a Doppler return from an ultrasonic sensor. This sensor might give us *both* crude and noisy estimates of distance but reasonably accurate velocity estimates. How can we fuse this information to obtain reliable estimates of hand distance? This involves a predictive model over time.

We can treat the true position of the hand as an unobserved parameter and estimate it from sensor data using Bayesian inference. A simple probabilistic filter uses posteriors from the previous time step to form the prior in the following step. To account for the passage of time, *dynamics* are applied to the posterior before it becomes the next prior (Figure 1.32). These dynamics are a predictive model that moves the distribution forward in time. The dynamics can often be very simple and can involve parameters that are also simultaneously inferred. For example, we might assume that hand position changes by the current estimated hand velocity over a fixed time interval. We can update both position and velocity using ordinary Bayesian inference, then apply the velocity to the posterior distribution of positions and feed this forward to the next time step.

Techniques like (unscented [55]) Kalman filters [1] and particle filters (also called sequential Monte Carlo filtering) make implementing probabilistic filters in interaction problems straightforward – once the modelling is done – and reasonably computationally efficient.

1.6 Facets of Bayesian Methods for Interaction Design

Bayesian approaches intersect with interaction problems in several ways. Five of these facets are outlined in the sections below:

1. Bayesian inference at **interaction time**, inferring the intention of a user in the control loop.
2. Bayesian optimisation at **design time**, efficiently optimising designs with humans in the design process.
3. Bayesian analysis at **evaluation time**, analysing the outcomes of an empirical interaction work.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

4. Interaction to support Bayesian modelling, through visualisation, workflow support and interactive model construction and exploration.
5. Bayesian models as an approximation for human cognition, to guide the design of interactive systems with well-founded predictions of user behaviour.

1.6.1 Optimal Mind-Reading: How Can Bayesian Methods Work Out What a User Wants to Do?

Bayesian methods can be used to represent the problem of interaction itself – how does information flow from human to computer? This can be used to derive robust models based around inference of intention. Strong prior models of what we expect users to do allow us to extract maximum value from input and preserve uncertainty about user intent. If we already know what intentions are *likely* to be expressed, we do not need as much information to reliably determine the true intention. This is a model of interaction founded in the idea of the interface as a concentrator of belief, whose mechanics are driven by the logic of probability. Such an interface represents, manipulates and displays uncertainty as a first-class value [50, 51]. This can extend throughout the interaction loop, from low-level inference about user state from sensors [47], interpretation of pointing actions [27], probabilistic GUIs [10], text entry [56], error-tolerant interfaces [58], motion correlation [54] and 2D selection [37].

We can conceive of an interface as a system that tries to *infer* what the user wants. We formulate a distribution over possible outcomes (e.g. over items on a menu), and an associated prior (e.g. from historical frequencies of interaction). We then update this probability distribution using observed inputs (e.g. the sequence of motion events from a pointing device).

> Note that this involves building a model that simulates the sequence of motion events *given* the menu item: a forward model that predicts for all possible menu items what the observed pointer movements would be! This is the opposite of the typical way of thinking about this problem.

Bayesian probabilistic interfaces let us formulate the intent inference problem in this way. This has some interesting effects:

- We can consistently introduce priors to imbue our interfaces with 'intelligence', without special ad hoc hacks;
  - these priors can include static, historical models,
  - or they can come from other input streams ('sensor fusion'),

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

  - or they can come from preceding states to fuse together information over time ('probabilistic filtering').
- We can simulate into the future and predict likely future actions (and, e.g., pre-cache likely responses).
- We can incorporate models of how interaction will unfold (e.g. what does a pointing movement look like) directly into our interface.
- We can provide feedback and display uncertainty to the user and make the behaviour of an interface more interpretable and predictable.
- We have a distribution over outcomes, and we can choose appropriate decision rules to actuate events. This can incorporate utility functions to account for different values that events might have.

This view on interaction sees user intentions as **unknown values** which are partially observed through inputs. The time series of inputs from the user give a partial, noisy, incomplete view of intention inside the user's head, along with a great deal of superfluous information. This is equivalent to the information–theoretic viewpoint of an interface as a noisy channel through which intention flows.

We try to infer a *generative model*, which is a simplified representation of intention and how it is mediated and transformed by the world. The stronger the model we have available, the more effectively we can infer intention. In this view, improving interaction (or at least *input*) comes down to more efficiently concentrating probability density where a user wants it to go. A better pointing device reduces uncertainty faster; a better display helps a user understand how best to target future actions to concentrate belief as desired; a better model of the user intentions concentrates belief with less explicit effort on the part of a user.

1.6.2 Fast Tuning in a Noisy World: How Can Bayesian Approaches Be Used to Optimise User Interfaces?

In an optimisation problem we have one or more *objective functions* (each a single numerical measure of goodness or badness), which depend upon some *parameters*, which we are interested in adjusting to minimise the objective function, usually bounded by some *constraints*. Many design problems can be framed in these terms. As an example, we might want to improve an aspect of a user interface, such as the scrolling speed of a photo viewer. This has an adjustable parameter (speed), bounded by some maximum and minimum speed (constraints), and we could derive a measure of performance, like the reported subjective satisfaction, as our objective function.

Numerical optimisation is an extremely powerful tool to solve these types of problems, but it developed in engineering contexts like the design of aeroplane wings, where strong mathematical models were well established and precise measurements were practical. Objective functions, however, are not always known, as is very often the case in interaction design. We would *not* expect to have any good model of satisfaction as a function of scroll speed, and it would be impractical to imagine deriving one from first principles. We may instead have a situation where we can only measure the value of the objective function at a few *definite* parameter settings, perhaps with significant measurement noise. In the scrolling example, we are free to sample different scroll speeds and ask users how they like it. Acquiring measurement points like this is expensive if humans are in the loop, so it is important to be parsimonious in sampling possible parameter settings. Humans are expensive to measure, noisy in their actions and not governed by simple mathematical formulae.

These issues motivate the use of a *proxy objective function*, a model that we learn from data that approximates the functional relationship between the parameters and the response (Figure 1.33). We now have to deal with two problems: Which specific example parameters (speeds, in the example) should we test with users? And how should we deal with the fact that the measurements we make may be noisy? We certainly don't expect to be able to repeat an experiment with a fixed scrolling speed and get the same satisfaction level. These are problems well solved by **Bayesian optimisation**, where we form a distribution over possible proxy objective functions, update these from measurements and can sequentially optimally select the most *informative* next test to make, taking into account the (epistemic) uncertainty of our model and the (aleatoric) uncertainty of our measurements.

This can range from simple Bayesian A/B testing to sophisticated modelling of user behaviour at a fine level of granularity. Bayesian optimisation can be applied to a huge range of problems with expensive or noisy functions, from inferring subjective preferences [9] and effective interface design [16] to optimising touch sensor configurations. Proxy objective function models like Gaussian processes [46] are well-supported by software packages and can often be slotted straight into a HCI problem. We can still combine these proxy objective functions with strong priors. As we incrementally improve our priors, our optimisation automatically becomes more informed and the sampling of measurements becomes more efficient.

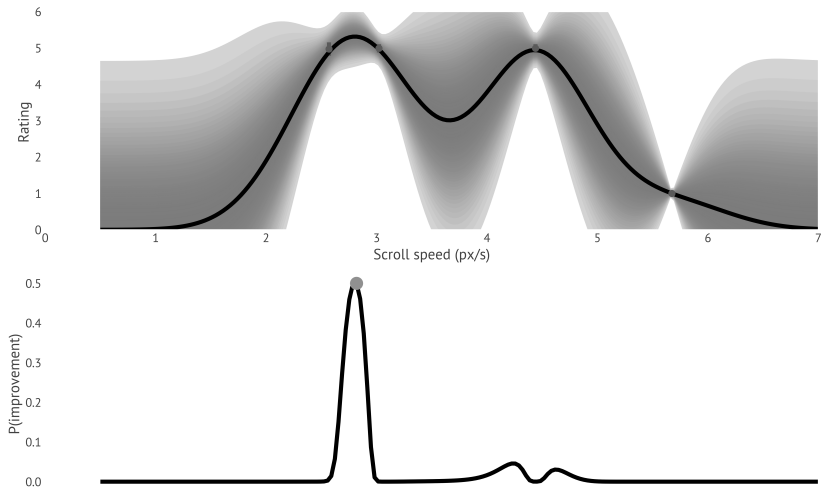

Figure 1.33 A mock example of Bayesian optimisation, using a proxy function to estimate user satisfaction (each point representing averaged scores from many participants on a 1- to 5-point scale) as the speed of a scrolling photo viewer is adjusted. Which speed should we test next to most quickly tune the photo browser? The uncertainty in the proxy function gives informed strategies to do so. The upper pane shows a Gaussian process proxy function fitted to four noisy satisfaction score measurements. The lower panel shows the probability of improvement across the space of scroll speeds. By selecting the point with maximal probability of improvement (marked with a circle), we define a strategy to find the next scroll speed to try out with users.

1.6.3 Evaluating with Uncertainty and Limited Data: How Can Bayesian Methods Be Used to Analyse the Results of Human–Computer Experiments?

Human–computer interaction, by its very nature, depends heavily on the evaluation of interactive systems with users. Empirical evaluations are a basic and near universal aspect of HCI research and practice.

Bayesian approaches offer a potentially superior way of analysing some kinds of quantitative experimental work that arise in HCI. Experiments of all types result in numbers, but we know that the interpretation of these are subject to uncertainty; this is why we have statistics. Statistics is divided into two schools of thought: *frequentist* statistics, which encompasses classical approaches such as null hypothesis testing widely used in HCI; and *Bayesian* statistics, which involves quite distinct principles of inference, reasoning from a prior to a posterior. These two schools of thought remain bitterly divided over the correct

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

way to interpret data. Kay et al. [31] motivate the use of Bayesian statistics in the evaluation of HCI experiments and put forward the case that Bayesian statistics are a better fit to the research practices in HCI, particularly in the re-usability of analyses from previous work and dealing with small sample sizes and weak effects. The awkward fit of 'dichotomous inference' (does this effect exist or not?) that is the focus of frequentist methods to interaction design has also come under criticism [3].

It is important to note that both frequentist and Bayesian methods are valid ways of interpreting data, but they answer different questions and have distinct trade-offs. For historical reasons, empirical research in psychology-adjacent fields such as HCI has focused almost exclusively on frequentist methods, particularly null hypothesis testing (NHST). While powerful and well-understood, they are not always well suited to answer the questions that we wish to investigate in HCI, and in the worst cases degrade into cargo cult statistics ('just slap a t-test on it and hope for $p < 0.05$'). Bayesian methods are no less susceptible to poor research practices, but they do require a more explicit consideration of the problem.

From an interaction design perspective, Bayesian approaches can directly answer questions of interest and can incorporate first-principles models from domain experts. They are well suited to problems where there is small data, where large controlled studies may not be practical or desirable. Bayesian statistical models make it easy to incorporate complex models, such as hierarchical regression. There are opportunities for novel and efficient experimental designs (e.g. online Bayesian experimental design) and practical meta-analyses. The advance of easy-to-use packages for Bayesian inference (e.g. stan, pymc3, brms, pyro) makes powerful Bayesian inference models reachable for non-specialist researchers. *Statistical Rethinking* [40] is recommended reading as a non-HCI introduction to Bayesian data analysis. Phelan et al. [44] give some guidance for the application of Bayesian models specifcally in human–computer interaction.

We cannot adequately describe the distinction between the two statistical schools of thought in this chapter. We refer interested readers to the recommended reading at the end of the chapter. A brief summary of the distinctive aspects of Bayesian approaches to designing and interpreting empirical work for interaction designers is summarised below:

- Bayesian methods require *priors* to be defined. Frequentist methods do not. This has several consequences:
  - we *have* to be able to come up with justifiable priors, which may be hard;
  - but if we do have them, we can use them (and frequentist methods cannot);

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

  - our priors force us to 'lay our cards on the table' and make our assumptions explicit;
  - priors are actionable and scrutable, in that we can visualise or otherwise explore their consequences.

- Bayesian methods answer questions about the specific evidence in this instance, given the priors. Frequentist methods answer questions about long-run behaviour, without regard to priors. In HCI, we often only *can* ever have one experiment and may be less concerned about objective statements about the long-run behaviour.
- Frequentist methods are ideal to establish if an effect exists (as opposed to a random occurence), in the context of randomised controlled trial. This is, for example, vitally important in medical contexts. In HCI, such problems do occur, but there are other research investigations of interest that do not fit into this dichotomous mode of thinking, or for which randomised controlled trials are not feasible or meaningful. In these cases, a Bayesian approach may be more flexible.
  - Bayesian methods *can* be used for dichotomous analyses, using **Bayes factors** to compare the relative likelihood of two competing hypotheses with the same evidence. However, this is sensitive to the choice of priors and therefore requires particular caution.
- Bayesian methods are flexible regarding combining evidence acquired at any time, from any source. It is possible to run a Bayesian design where experiments are run until a result is established with a certain degree of confidence. This is harder to do correctly with a frequentist approach. Likewise, it is easier to combine multiple studies in a Bayesian framework than it is to conduct a meta-analysis of studies in a frequentist framework. A Bayesian framework can feed the results of previous studies in *directly* as priors.
- Many frequentist methods have to be applied very carefully to preserve their validity (for example, it is essential to apply corrections to preserve false positive resistance in the presence of multiple tests on the same data). Bayesian methods still require care in application but often break in more obvious ways.
- All modelling involves a bias–variance trade-off. A more complex model with more parameters will require more evidence to be constrained than a simpler model with fewer parameters. In a Bayesian framework, this is usually easy to inspect; the posterior distribution will tell us how informed the parameters of our model are. This can be harder to reason about in frequentist models.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- Bayesian methods can use models of essentially any form that can be computed, including complex simulators. Frequentist methods require a more limited class of models.
  - Bayesian models require the statistical model to be defined in detail (e.g. as a program), which may require more work, but this model is a free choice. Frequentist methods can devolve into picking an approach from a zoo of tests and procedures, which may be an awkward fit.
  - It is easy in Bayesian approaches to build *hierarchical models*, where the parameters at one 'layer' depend on another 'layer'. For example, we might model reaction time as coming from a distribution that is distinct for each user; and the distribution of user distributions might be governed by a common population model. This is a flexible way of gradually introducing more sophisticated models.
- Frequentist methods can often be computed in a closed-form directly, exactly and efficiently. This is rarer in Bayesian methods, where approximate and computationally intensive methods are used (such as MCMC).
  - Frequentist methods are rarely troubled by computational problems. Because Bayesian methods depend on computational approximations, it is often necessary to tweak models so that the computation works efficiently (e.g. reparameterisation to make MCMC sample well). This is a clear case of the tail wagging the dog! Approximations can also go wrong and produce dubious results. This requires vigilance in monitoring diagnostics and can involve tricky 'debugging'.
- Bayesian methods give results in terms of the likelihood of different hypotheses. This is often the question we want to answer, and means we can use the statistics to work the argument we want to make, rather than having the statistics define the argument we are able to make:
  - **Bayesian:** 'The 95% credible interval for increase in reading speed from version A to version B is [1.7, 2.1]', meaning that we are 95% sure that the reading speed improvement is in that range, given our priors.
  - **Frequentist:** 'The 95% confidence interval for increase in reading speed is [1.7, 2.1]', meaning we expect that if we ran this experiment repeatedly, the interval we computed would capture the true reading speed 95% of the time (but this specific interval might not!).
  - **Bayesian:** 'Given our priors, the expected increase in tweets read per day is +2.9, and a decrease in tweets read has a probability of less than 5%', meaning exactly what it says.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

- **Frequentist:** 'We find a mean increase of 2.9 tweets/day, and [following an appropriate test] we reject the null hypothesis with $p < 0.05$', meaning that if we ran this experiment repeatedly, and the null hypothesis was true and variation in tweets was purely random, then we would expect to see results as extreme or more extreme than this less than 5% of the time. Is this a question we are interested in knowing the answer to?
- In all cases, these statements depend on the assumptions made. In the Bayesian case, these are the models and the priors. In the frequentist case, we have no priors to worry about, but we have a restricted class of models and distributions available, and we must make sure that these are appropriate for the problem. Many common frequentist models are *normal linear models* and are powerful if these assumptions hold but problematic if not.

1.6.4 Bayesian Interaction: How Can We Visualise and Interact with Bayesian Models?

One notable problem with Bayesian models is that they can be hard to understand, particularly for non-experts and particularly when couched in the traditional technical language. The results of Bayesian inference can be rich – and therefore hard to summarise and superficially non-intuitive. Building Bayesian models and verifying they are doing what is expected is a task that can be error-prone. This is a problem that human–computer interaction is well placed to solve.

Bayesian interaction is the problem of how to display, explain, explore, construct and criticise probabilistic Bayesian models: how and where to put users in the loop in Bayesian modelling. This involves supporting users in making rational and informed decisions, like assessment of risk or of expected value, in comprehending the structure of Bayesian models and the interpretation of their parameters, and in aiding the development of new models and debugging and criticising them. User interaction with Bayesian models has a few important aspects:

- **Visualisation of uncertainty:** Distributions are the basis of Bayesian modelling, and so the representation of distributions to users is important, particularly the visualisation of uncertainty [28, 43]. People have a hard time understanding distributions and their implications, and this is exacerbated when distributions are over a high-dimensional parameter space. Visualising and summarising distributions in ways that preserve uncertainty are significant challenges in HCI. Even in simpler problems there can be user-facing uncertainty that should not be elided. Interacting with noisy sensors, like a

brain–computer interface, results in an uncertain interface state. Reflecting this uncertainty to users could improve interaction, but requires new interface techniques such as uncertain cursors.

- **Interactive exploration of Bayesian models:** Early work in Bayesian visualisation has focused on static summaries of posteriors. There are many open research questions in user interfaces that support dynamic interaction. Interactive systems to explore possible configurations is the most promising way to communicate large Bayesian models. The ability to slice, brush or tour posterior distributions can reveal structured correlations in the results of inference [53].
- **Prior elicitation and Bayesian workflow support:** Bayesian models depend upon suitable priors. There are open research questions around the best interfaces to elicit priors from experts, and to test and verify their validity (e.g. via interactive prior predictive checks). This ties into the problem of **Bayesian workflow** research: how to support users in building, running and verifying Bayesian models, such as visualisation in Bayesian workflows [22] and interactive prior elicitation [30]. This might range from debugging tools to diagnose poor parameterisations or misbehaving MCMC samplers, to defining wider processes to construct and revise models within a scientific team.

1.6.5 A Higher Perspective: How Can Bayesian Ideas Help Us Understand Human Cognition?

There is a school of thought that interprets the thought processes of humans and other living beings as an approximate form of Bayesian inference. This 'Bayesian brain' hypothesis [20, 21] implies that we are all engaged in some form of approximate Bayesian inference, from low-level sensory perception through to higher-level cognition. It posits a model of cognition where organisms form predictive models of the world [26] and revise them in light of sensory evidence in a manner compatible with Bayesian belief updates. This framework gives a structure by which causal origins of perceptual stimuli can be inferred by organisms, and by links to information-theoretic models of behaviour and perception [29].

This is a controversial hypothesis, and one that is hard to gather definitive evidence for or against. However, it can be a powerful lens through which to examine how we will react and behave with interactive systems. Regardless of its biological 'truth', Bayesian cognitive models are amenable to computation and can provoke new thoughts on how to engineer interactions. As a concrete example, Rao [45] models visual cue integration as a Kalman filter, a recursive Bayesian update process. This form of model postulates that living beings

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

combine predictive models of how the world is expected to evolve and evaluate this against evidence from sensory channels. In human–computer interaction, research on understanding how users interpret data visualisations [32, 59] can be modelled by representing users with a Bayesian cognition and the consequent belief updates they would perform under this model.

1.7 Bayesian Pasts and Futures

1.7.1 How Did These Ideas Come About?

Bayesian ideas of probability were first stated in a limited form by Thomas Bayes, in the eighteenth century, in notes that were unpublished until after his death [2]. The ideas were extended by Pierre Simon Laplace in France in the early nineteenth century [35].

Bayesian interpretations fell out of favour, and for many decades these approaches were ignored, either because they could not practically be used for lack of computational power, or on philosophical grounds. Vigorous and bitter debates about validity of Bayesian ideas in the first half of the twentieth century left Bayesian modelling as a niche subject until the end of the twentieth century. We leave the details to others; McGrayne [41] is an accessible history of this conflict.

1.7.2 Why Is This Suddenly Relevant Now?

From the 1980s onwards, computational power became available that made Bayesian statistics suddenly practical. The development of tools like BUGS in the early 1990s, and the subsequent development of efficient Markov chain Monte Carlo samplers interfaced to probabilistic programming languages, brought these modelling tools to specialists who did not have to implement the micro-details of numerical inference. These two factors make large Bayesian models tractable, and reduce the need for clever algebraic manipulations. An increasing number of accessible texts on Bayesian modelling has ignited interest among new audiences.

There is also an increasing realisation that traditional statistical methods are not always well suited to the problems that are encountered in interaction design, and alternative methodologies can be more insightful. Some Bayesian methods, such as Kalman filtering, have long been known in HCI, but as a kind of ‘magical’ special-case algorithm, rather than what is a fairly ordinary use of Bayesian modelling.

1.7.3 Does Uncertainty = Bayesian?

Our primary motivation for applying Bayesian modelling is to properly account for uncertainty. Uncertainty in Bayesian models is represented with probability. Probability is *not* the only way to represent uncertainty, but it is arguably the *right* way to represent it [36]. Probability has a simple, rigorous axiomatic basis [13]. It can further be shown that any non-probabilistic representation in a situation with uncertain outcomes of different values, as in a betting game, is inferior to a probabilistic representation, in terms of expected return [14]. However, there are other models of uncertainty which may be computationally or philosophically more convenient to apply; a review of alternatives is given by Zio and Pedroni [60].

We can also use probability without applying Bayesian ideas, as in frequentist models. Frequentism strictly limits the elements about which we may be uncertain, limiting probability to represent the uncertain outcomes of repeatable experiments (or draws from some distribution). At the same time, this avoids the troubles of subjective probability, and the well-developed mathematical theory for frequentist models means that many quantities of interest can be computed quickly and without resort to approximation. Many useful probabilistic models in human–computer interaction, such as Hidden Markov Models for sequence recognition, are often implemented from a frequentist perspective.

1.7.4 Ethics of Bayesian Interaction

Modelling choices are *not* ethically neutral, even at this highest of abstraction levels. Placed as we are at the junction between computer science and the human world, interaction designers and HCI researchers have a particular role in evaluating the ethical implications of our modelling choices and advancing ethical research practices.

- **Uncertainty:** Above all else, failure to represent uncertainty can be an ethical failing. Bayesian methods are not the only way to work with and represent with uncertainty, but they put uncertainty at the heart of all computation. Discarding or eliding uncertainty can be deception (for example, in reporting of empirical work), or can present direct risks of harm (for example, in safety critical systems). Uncertainty must be accounted for, preserved and represented. Bayesian methods are the most straightforward way to do so; some would argue, the *only* correct way to do so.
- **Priors:** Bayesian methods need priors. On one hand, the requirement that assumptions are always laid out in the open as fully defined priors invites inspection and transparency. This is important if the outcome of inference

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

affects people's lives; these explicit priors can be reviewed, challenged or modified. On the other hand, malevolent manipulation of priors can be used to produce any result at all. This implies a duty and responsibility to document and justify priors, and to publish models that allow inference to be conducted under alternative priors. Reproducibility is important for all science, but especially for Bayesian models.

- **Small data** Bayesian models can often make useful inferences with very small datasets, by using informative priors. This is typical, for example, in astronomical tasks like identifying exoplanets – there is simply very little data to work with. In interaction, human responses are *valuable*. It costs money and time to run experiments with users. Wasteful experimental designs or analyses are unethical. Bayesian methods have two advantages: they can work well in the small data domain; and *active learning* approaches can be used to adapt experiments online to precisely control the uncertainty remaining.

1.7.5 Disadvantages, Cons and Caveats

Why isn't *everything* Bayesian? We've seen how much Bayesian approaches offer, and yet it currently has a tiny foothold in human–computer interaction. Even in other disciplines like astronomy, where it is better established, it is still a minority approach. Part of the reason Bayesian approaches in interaction can appear so appealing is because of the vacuum of general ideas in interaction [33] and the in-rush of enlightenment that these approaches bring lays bare much low-hanging fruit.

A great deal of the slow uptake is historical, stemming from the rancorous debates over the validity of Bayesian ideas in statistics and the absence of workable solutions to perform inference. However, there are real issues with Bayesian approaches that need to be understood.

- **Computational resources:** Bayesian methods can require extensive computational resources, certainly compared with common frequentist experimental analyses. In real-time settings with a human-in-the-loop, Bayesian inference may be difficult to compute quickly enough to maintain responsiveness. In large, complex offline analyses with hundreds of parameters, inference can be extremely time consuming. In cases where uncertainty is not especially relevant, or good priors are not available, other approaches may be much more practical. Non-Bayesian machine learning or statistical approaches will likely outperform Bayesian methods in such cases.
- **Approximations:** Computational approaches to Bayesian inference almost invariably rely on approximations. Approximations can be hard to configure.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

They can go wrong and produce meaningless results. They can be hard to debug. Diagnostics are available, but they are not perfect indicators of flaws. Care must be taken in formulating problems to get a form that is both conceptually correct and practical to approximate. Users of Bayesian methods need to be diligent in verifying the quality of approximations, and reporting carefully exactly how inference was performed and any relevant diagnostic measures.

- **Communication and understanding:** Bayesian methods may be harder to communicate to audiences unfamiliar with them. This presents a barrier to dissemination. There are venues where reviewers expect to see an ANOVA and a p-value and will be uncomfortable if these are absent. On the other hand, Bayesian models and results are often easier to correctly interpret once understood. Credible intervals and posterior histograms coincide more closely with human intuitions than confidence intervals or other frequentist summaries. There is not always a standard way to represent the results of inference. Bespoke visualisations or summary statistics may be needed.
- **Formulation of problems:** Bayesian approaches are extremely general. This also implies that there is a lot of work in correctly specifying a Bayesian approach. The real difficulty is in formulating problems in terms of probabilistic models. This is a difficult skill to acquire and takes a change in perspective that is unfamiliar to most researchers and practitioners. Bayesian models can be much more sophisticated than standard statistical methods. This offers power, but it can often be hard to explain something like a hierarchical Bayesian model, and particularly to give meaningful names and interpretations to high-level parameters ('the mean of the variance of the distribution of variances over variances'). Bayesian models will be specific to the problem under consideration, and not an 'off-the-shelf' statistical model like a t-test or ANOVA. This means that these models *have* to be communicated clearly and rigorously.
- **Subjectivity:** Bayesian methods are typically applied in a *subjective* fashion (there is such a thing as objective Bayesian statistics, but this is perhaps best left alone). It only answers questions that relate to how belief changes from a prior in the light of evidence. Most of the time, this is what we want, and subjectivity is a desirable quality. Subjective beliefs, however, require justification and explanation. Documenting how results are arrived requires care and thought.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

1.8 Where Do I Go from Here?

1.8.1 Introductory Texts on Bayesian Statistics

- For those looking for a first introduction to Bayesian statistics, we particularly recommend McElreath's *Statistical Rethinking* [40], which provides a thorough and highly readable grounding in Bayesian modelling without assuming any background.
- *Think Bayes* (second edition) by Allen Downey [15] is a slender and accessible text on Bayesian statistics with extensive worked examples in Python.
- We also recommend Lambert's *A Student's Guide to Bayesian Statistics* [34], which has excellent supporting video material.
- McGrayne's *The Theory that Would Not Die* [41] is a very accessible popular science account of the history of Bayesian reasoning and is enlightening in establishing the context in which Bayesian methods are discussed and used in the statistical world.
- For a more mathematically rigorous discussion, the series of online articles by Michael Betancourt are self-contained, rigorous and beautifully illustrated. In particular, *Foundations of Probability Theory* [5], *Conditional Probability Theory for Scientists and Engineers* [4], *Probabilistic Computation* [6], *Modeling and Inference* [7] and *Towards a Principled Bayesian Workflow* [8] are approachable for computer scientists with some background in mathematics.

More Advanced Texts

- *Bayesian Data Analysis* by Gelman et al. [23] is the standard book on Bayesian approaches to data analysis. It requires significantly more mathematical background than the introductory texts listed above, but it is a comprehensive resource.
- *Regression and Other Stories* by Gelman, Hill and Vehtari [24] is a thorough introduction to Bayesian regression modelling and encompasses both the how and the why of regression modelling.
- A more adventurous text that links together information theory, machine learning and Bayesian approaches is *Information Theory, Inference and Learning Algorithms* [38], by MacKay (who, in the HCI world, introduced the *Dasher* probabilistic text entry system [56]). This is a more mathematically challenging text, but the perspective that it provides is particularly useful for human–computer interaction, where flows of information interact with inference engines.

https://doi.org/10.1017/9781108874830.003 Published online by Cambridge University Press

References

[1] T. Babb. 2015. *How a Kalman Filter Works, in Pictures.* Bzarg. www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/.

[2] T. Bayes, Thomas. 1763. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical Transactions of the Royal Society of London*, 370–418.

[3] L. Besançon and P. Dragicevic, Pierre. 2019. The continued prevalence of dichotomous inferences at CHI. Pages 1–11 of: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI EA '19. Association for Computing Machinery.

[4] M. Betancourt. 2018 (Oct.). *Conditional Probability Theory (For Scientists and Engineers)*. https://betanalpha.github.io/assets/case_studies/conditional_probability_theory.html.

[5] M. Betancourt. 2018 (Oct.). *Probability Theory (for Scientists and Engineers)*. https://betanalpha.github.io/assets/case_studies/probability_theory.html.

[6] M. Betancourt. 2019 (June). *Probabilistic Computation*. https://betanalpha.github.io/assets/case_studies/probabilistic_computation.html.

[7] M. Betancourt. 2019 (Mar.). *Probabilistic Modeling and Statistical Inference*. https://betanalpha.github.io/assets/case_studies/modeling_and_inference.html.

[8] M. Betancourt. 2020 (Apr.). *Towards a Principled Bayesian Workflow*. https://betanalpha.github.io/assets/case_studies/principled_bayesian_workflow.html.

[9] E. Brochu, T. Brochu and N. de Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. Pages 103–112 of: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '10. Eurographics Association.

[10] D. Buschek, Daniel, and F. Alt. 2017. ProbUI: generalising touch target representations to enable declarative gesture definition for probabilistic GUIs. Pages 4640–4653 of: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Association for Computing Machinery.

[11] D. Buschek, S. Rogers and R. Murray-Smith. 2013. User-specific touch models in a cross-device context. Pages 382–391 of: *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services*. Association for Computing Machinery.

[12] B. Carpenter, A. Gelman, M.D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li and A. Riddell. 2017. Stan: a probabilistic programming language. *Grantee Submission*, **76**(1), 1–32.

[13] R. T. Cox. 1946. Probability, frequency and reasonable expectation. *American journal of physics*, **14**(1), 1–13.

[14] B. De Finetti. 1975. *Theory of Probability: A Critical Introductory Treatment*, Vol. 6. John Wiley & Sons.

[15] A. B. Downey. 2021. *Think Bayes*. O'Reilly Media, Inc.

[16] J. J. Dudley, J. T. Jacques and P. O. Kristensson. 2019. Crowdsourcing interface feature design with Bayesian optimization. Pages 1–12 of: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.

[17] C. Feng. *MCMC Interactive Gallery*. https://chi-feng.github.io/mcmc-demo/app.html.
[18] M. Fernandes, L. Walls, S. Munson, J. Hullman and M. Kay. 2018. Uncertainty displays using quantile dotplots or CDFs improve transit decision-making. Pages 1–12 of: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.
[19] P. M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, **47**(6), 381.
[20] K. Friston. 2010. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, **11**(2), 127–138.
[21] K. Friston. 2012. The history of the future of the Bayesian brain. *Neuroimage*, **62-248**(2), 1230–1233.
[22] J. Gabry, D. Simpson, A. Vehtari, M. Betancourt and A. Gelman, Andrew. 2019. Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **182**(2), 389–402.
[23] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari and D. B. Rubin. 2013. *Bayesian Data Analysis*. CRC Press.
[24] A. Gelman, J. Hill, and A. Vehtari. 2020. *Regression and Other Stories*. Cambridge University Press.
[25] A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner and M. Modrák. 2020. Bayesian Workflow. *arXiv preprint arXiv:2011.01808*.
[26] T. L. Griffiths, and J. B. Tenenbaum. 2006. Optimal predictions in everyday cognition. *Psychological Science*, **17**(9), 767–773.
[27] T. Grossman, and R. Balakrishnan. 2005. A probabilistic approach to modeling two-dimensional pointing. *ACM Transactions on Computer–Human Interaction*, **12**(3), 435–459.
[28] J. Hullman, X. Qiao, M. Correll, A. Kale and M. Kay. 2018. In pursuit of error: a survey of uncertainty visualization evaluation. *IEEE Transactions on Visualization and Computer Graphics*, **25**(1), 903–913.
[29] G. Jensen, R. D. Ward and P. D. Balsam. 2013. Information: theory, brain, and behavior. *Journal of the Experimental Analysis of Behavior*, **100**(3), 408–431.
[30] G. Jones and W. O. Johnson, Wesley O. 2014. Prior elicitation: interactive spreadsheet graphics with sliders can be fun, and informative. *The American Statistician*, **68**(1), 42–51.
[31] M. Kay, G. L. Nelson and E. B. Hekler. 2016. Researcher-centered design of statistics: why Bayesian statistics better fit the culture and incentives of HCI. Pages 4521–4532 of: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.
[32] Y.-S. Kim, L. A. Walls, P. Krafft and J. Hullman. 2019. A Bayesian cognition approach to improve data visualization. Pages 1–14 of: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Association for Computing Machinery.
[33] V. Kostakos. 2015. The big hole in HCI research. *Interactions*, **22**(2), 48–51.
[34] B. Lambert. 2018. *A Student's Guide to Bayesian Statistics*. Sage.
[35] P.-S.Laplace. 1812. *Théorie Analytique Des Probabilités*. Courcier.

[36] D. V. Lindley. 1987. The probability approach to the treatment of uncertainty in artificial intelligence and expert systems. *Statistical Science*, **2**(1), 17–24.

[37] W. Liu, d'Oliveira, M. Beaudouin-Lafon and O. Rioul, Olivier. 2017. Bignav: Bayesian information gain for guiding multiscale navigation. Pages 5869–5880 of: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.

[38] D. J. C. MacKay. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

[39] I. S. MacKenzie. 1992. Fitts' law as a research and design tool in human–computer interaction. *Human-Computer Interaction*, **7**(1), 91–139.

[40] R. McElreath. 2018. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Chapman and Hall/CRC.

[41] S. B. McGrayne. 2011. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of C*. Yale University Press.

[42] A. Oulasvirta, X. Bi and A. Howes. 2018. *Computational Interaction*. Oxford University Press.

[43] L. Padilla, M. Kay, and J. Hullman. 2020. *Uncertainty Visualization*. https://psyarxiv.com/ebd6r

[44] C. Phelan, J. Hullman, M. Kay and P. Resnick. 2019. Some prior(s) experience necessary: templates for getting started with Bayesian analysis. Pages 1–12 of: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.

[45] R. P. N. Rao and D. H. Ballard. 1997. Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, **9**(4), 721–763.

[46] C. E. Rasmussen. 2003. Gaussian processes in machine learning. Pages 63–71 of: O. Bousquet, U. von Luxberg and G. Rätsch, eds., *Advanced Lectures on Machine Learning*. Springer.

[47] S. Rogers, J. Williamson, C. Stewart and R. Murray-Smith. 2011. AnglePose: robust, precise capacitive touch tracking via 3D orientation estimation. Page 2575 of: *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems – CHI '11*. ACM Press.

[48] J. Salvatier, T. V. Wiecki and C. Fonnesbeck. 2016. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, **2**(Apr.), e55.

[49] D. J. Schad, M. Betancourt and S. Vasishth. 2020. Toward a principled Bayesian workflow in cognitive science. *Psychological Methods*, **26**(1), 103–126.

[50] J. Schwarz, S. Hudson, J. Mankoff and A. D. Wilson. 2010. A framework for robust and flexible handling of inputs with uncertainty. Pages 47–56 of: *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery.

[51] J. Schwarz, J. Mankoff and S. Hudson. 2011. Monte Carlo methods for managing interactive state, action and feedback under uncertainty. Pages 235–244 of: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery.

[52] C. E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, **27**(3), 379–423.

[53] E. Taka, S. Stein and J. H. Williamson. 2020. Increasing interpretability of Bayesian probabilistic programming models through interactive visualizations. *Frontiers in Computer Science*, **2**. https://doi.org/10.3389/fcomp.2020.567344.

[54] E. Velloso and C. H. Morimoto. 2021. A probabilistic interpretation of motion correlation selection techniques. Pages 1–13 of: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.

[55] E. A. Wan and R. Van Der Merwe. 2000. The unscented Kalman filter for nonlinear estimation. Pages 153–158 of: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE.

[56] D. J. Ward, A. F. Blackwell and D. J. C. MacKay. 2000. Dasher—a data entry interface using continuous gestures and language models. Pages 129–137 of: *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery.

[57] J. Weisberg. 2011. Varieties of Bayesianism. *Inductive Logic*, **10**, 477–551.

[58] J. H. Williamson, M. Quek, I. Popescu, A. Ramsay and R. Murray-Smith. 2020. Efficient human-machine control with asymmetric marginal reliability input devices. *PLOS One*, **15**(6), e0233603.

[59] W. Wu, L. Xu, R. Chang and E. Wu. 2017. Towards a Bayesian model of data visualization cognition. In: *IEEE Visualization Workshop on Dealing with Cognitive Biases in Visualisations (DECISIVe)*. IEEE.

[60] E. Zio and N. Pedroni. 2013. Literature review of methods for representing uncertainty. *FonCSI*, **2013-03**(ISSN 2100-3874).