

Instructor's Solution Manual

Artificial Intelligence

A Modern Approach

Fourth Edition

Stuart J. Russell and Peter Norvig

with contributions from

Nalin Chhibber, Ernest Davis, Nicholas J. Hay, Jared Moore, Alex Rudnick,
Mehran Sahami, Xiaocheng Mesut Yang, and Albert Yu

This solution manual is intended for the instructor of a class. Students should use the online site for exercises at aimacode.github.io/aima-exercises. That site is open for anyone to use. It offers solutions for some but not all of the exercises; an instructor can check there to see which ones have solutions. The exercises are online rather than in the textbook itself because (a) the textbook is long enough as is, and (b) we wanted to be able to update the exercises frequently.

Copyright © 2022

EXERCISES 1

INTRODUCTION

Note that for many of the questions in this chapter, we give references where answers can be found rather than writing them out—the full answers would be far too long.

1.1 What Is AI?

Exercise 1.1.#DEFA

Define in your own words: (a) intelligence, (b) artificial intelligence, (c) agent, (d) rationality, (e) logical reasoning.

- a. Dictionary definitions of **intelligence** talk about “the capacity to acquire and apply knowledge” or “the faculty of thought and reason” or “the ability to comprehend and profit from experience.” These are all reasonable answers, but if we want something quantifiable we would use something like “the ability to act successfully across a wide range of objectives in complex environments.”
- b. We define **artificial intelligence** as the study and construction of agent programs that perform well in a given class of environments, for a given agent architecture; they *do the right thing*. An important part of that is dealing with the uncertainty of what the current state is, what the outcome of possible actions might be, and what is it that we really desire.
- c. We define an **agent** as an entity that takes action in response to percepts from an environment.
- d. We define **rationality** as the property of a system which does the “right thing” given what it knows. See Section 2.2 for a more complete discussion. The basic concept is *perfect* rationality; Section ?? describes the impossibility of achieving perfect rationality and proposes an alternative definition.
- e. We define **logical reasoning** as the process of deriving new sentences from old, such that the new sentences are necessarily true if the old ones are true. (Notice that does not refer to any specific syntax or formal language, but it does require a well-defined notion of truth.)

Exercise 1.1.#TURI

Read Turing’s original paper on AI (Turing, 1950). In the paper, he discusses several objections to his proposed enterprise and his test for intelligence. Which objections still carry

weight? Are his refutations valid? Can you think of new objections arising from developments since he wrote the paper? In the paper, he predicts that, by the year 2000, a computer will have a 30% chance of passing a five-minute Turing Test with an unskilled interrogator. What chance do you think a computer would have today? In another 25 years?

See the solution for exercise 26.1 for some discussion of potential objections.

The probability of fooling an interrogator depends on just how unskilled the interrogator is. A few entrants in the Loebner prize competitions have fooled judges, although if you look at the transcripts, it looks like the judges were having fun rather than taking their job seriously. There certainly have been examples of a chatbot or other online agent fooling humans. For example, see the description of the Julia chatbot at www.lazytd.com/ltd/julia/. We'd say the chance today is something like 10%, with the variation depending more on the skill of the interrogator rather than the program. In 25 years, we expect that the entertainment industry (movies, video games, commercials) will have made sufficient investments in artificial actors to create very credible impersonators.

Note that governments and international organizations are seriously considering rules that require AI systems to be identified as such. In California, it is already illegal for machines to impersonate humans in certain circumstances.

Exercise 1.1.#REFL

Are reflex actions (such as flinching from a hot stove) rational? Are they intelligent?

Yes, they are rational, because slower, deliberative actions would tend to result in more damage to the hand. If “intelligent” means “applying knowledge” or “using thought and reasoning” then it does not require intelligence to make a reflex action.

Exercise 1.1.#SYAI

To what extent are the following computer systems instances of artificial intelligence:

- Supermarket bar code scanners.
- Web search engines.
- Voice-activated telephone menus.
- Spelling and grammar correction features in word processing programs.
- Internet routing algorithms that respond dynamically to the state of the network.

- Although bar code scanning is in a sense computer vision, these are not AI systems. The problem of reading a bar code is an extremely limited and artificial form of visual interpretation, and it has been carefully designed to be as simple as possible, given the hardware.
- In many respects. The problem of determining the relevance of a web page to a query is a problem in natural language understanding, and the techniques are related to those

Exercises 1 Introduction

we will discuss in Chapters 23 and 24. Search engines also use clustering techniques analogous to those we discuss in Chapter 20. Likewise, other functionalities provided by a search engines use intelligent techniques; for instance, the spelling corrector uses a form of data mining based on observing users' corrections of their own spelling errors. On the other hand, the problem of indexing billions of web pages in a way that allows retrieval in seconds is a problem in database design, not in artificial intelligence.

- To a limited extent. Such menus tends to use vocabularies which are very limited – e.g. the digits, “Yes”, and “No” — and within the designers’ control, which greatly simplifies the problem. On the other hand, the programs must deal with an uncontrolled space of all kinds of voices and accents. Modern digital assistants like Siri and the Google Assistant make more use of artificial intelligence techniques, but still have a limited repertoire.
- Slightly at most. The spelling correction feature here is done by string comparison to a fixed dictionary. The grammar correction is more sophisticated as it need to use a set of rather complex rules reflecting the structure of natural language, but still this is a very limited and fixed task.

The spelling correctors in search engines would be considered much more nearly instances of AI than the Word spelling corrector are, first, because the task is much more dynamic – search engine spelling correctors deal very effectively with proper names, which are detected dynamically from user queries – and, second, because of the technique used – data mining from user queries vs. string matching.

- This is borderline. There is something to be said for viewing these as intelligent agents working in cyberspace. The task is sophisticated, the information available is partial, the techniques are heuristic (not guaranteed optimal), and the state of the world is dynamic. All of these are characteristic of intelligent activities. On the other hand, the task is very far from those normally carried out in human cognition. In recent years there have been suggestions to base more core algorithmic work on machine learning.

Exercise 1.1.#COGN

Many of the computational models of cognitive activities that have been proposed involve quite complex mathematical operations, such as convolving an image with a Gaussian or finding a minimum of the entropy function. Most humans (and certainly all animals) never learn this kind of mathematics at all, almost no one learns it before college, and almost no one can compute the convolution of a function with a Gaussian in their head. What sense does it make to say that the “vision system” is doing this kind of mathematics, whereas the actual person has no idea how to do it?

Presumably the brain has evolved so as to carry out this operations on visual images, but the mechanism is only accessible for one particular purpose in this particular cognitive task of image processing. Until about two centuries ago there was no advantage in people (or animals) being able to compute the convolution of a Gaussian for any other purpose.

The really interesting question here is what we mean by saying that the “actual person” can do something. The person can see, but he cannot compute the convolution of a Gaussian;

but computing that convolution is *part* of seeing. This is beyond the scope of this solution manual.

Exercise 1.1.#EVOR

Why would evolution tend to result in systems that act rationally? What goals are such systems designed to achieve?

The notion of acting rationally *presupposes* an objective, whether explicit or implicit. We understand evolution as a process that operates in the physical world, where there are no inherent objectives. So the question is really asking whether evolution tends to produce systems whose behavior can be interpreted consistently as rational according to some objective.

It is tempting to say that evolution tends to produce organisms that act rationally in the pursuit of reproduction. This is not completely wrong but the true picture is much more complex because of the question of what “system” refers to—it could be organisms (humans, rats, bacteria), superorganisms (ant and termite colonies, human tribes, corals), and even individual genes and groups of genes within the genome. Selection and mutation processes operate at all these levels. By definition, the systems that exist are those whose progenitors have reproduced successfully. If we consider an ant colony, for example, there are many individual organisms (e.g., worker ants) that do not reproduce at all, so it is not completely accurate to say that evolution produces organisms whose objective is to reproduce.

Exercise 1.1.#AISC

Is AI a science, or is it engineering? Or neither or both? Explain.

This question is intended to be about the essential nature of the AI problem and what is required to solve it, but could also be interpreted as a sociological question about the current practice of AI research.

A *science* is a field of study that leads to the acquisition of empirical knowledge by the scientific method, which involves falsifiable hypotheses about what is. A pure *engineering* field can be thought of as taking a fixed base of empirical knowledge and using it to solve problems of interest to society. Of course, engineers do bits of science—e.g., they measure the properties of building materials—and scientists do bits of engineering to create new devices and so on.

The “human” side of AI is clearly an empirical science—called cognitive science these days—because it involves psychological experiments designed out to find out how human cognition actually works. What about the the “rational” side? If we view it as studying the abstract relationship among an arbitrary task environment, a computing device, and the program for that computing device that yields the best performance in the task environment, then the rational side of AI is really mathematics and engineering; it does not require any empirical knowledge about the *actual* world—and the *actual* task environment—that we inhabit; that a given program will do well in a given environment is a *theorem*. (The same is true of pure decision theory.) In practice, however, we are interested in task environments that do approximate the actual world, so even the rational side of AI involves finding out what the actual

Exercises 1 Introduction

world is like. For example, in studying rational agents that communicate, we are interested in task environments that contain humans, so we have to find out what human language is like. In studying perception, we tend to focus on sensors such as cameras that extract useful information from the actual world. (In a world without light, cameras wouldn't be much use.) Moreover, to design vision algorithms that are good at extracting information from camera images, we need to understand the actual world that generates those images. Obtaining the required understanding of scene characteristics, object types, surface markings, and so on is a quite different kind of science from ordinary physics, chemistry, biology, and so on, but it is still science.

In summary, AI is definitely engineering but it would not be especially useful to us if it were not also an empirical science concerned with those aspects of the real world that affect the design of intelligent systems for that world.

Exercise 1.1.#INTA

“Surely computers cannot be intelligent—they can do only what their programmers tell them.” Is the latter statement true, and does it imply the former?

This depends on your definition of “intelligent” and “tell.” In one sense computers only do what the programmers command them to do, but in another sense what the programmers consciously tells the computer to do often has very little to do with what the computer actually does. Anyone who has written a program with an ornery bug knows this, as does anyone who has written a successful machine learning program. So in one sense Samuel “told” the computer “learn to play checkers better than I do, and then play that way,” but in another sense he told the computer “follow this learning algorithm” and it learned to play. So we’re left in the situation where you may or may not consider learning to play checkers to be a sign of intelligence (or you may think that learning to play in the right way requires intelligence, but not in this way), and you may think the intelligence resides in the programmer or in the computer.

Exercise 1.1.#INTB

“Surely animals cannot be intelligent—they can do only what their genes tell them.” Is the latter statement true, and does it imply the former?

The point of this exercise is to notice the parallel with the previous one. Whatever you decided about whether computers could be intelligent in 1.11, you are committed to making the same conclusion about animals (including humans), *unless* your reasons for deciding whether something is intelligent take into account the mechanism (programming via genes versus programming via a human programmer). Note that Searle makes this appeal to mechanism in his Chinese Room argument (see Chapter 27).

Exercise 1.1.#INTC

“Surely animals, humans, and computers cannot be intelligent—they can do only what their constituent atoms are told to do by the laws of physics.” Is the latter statement true, and does it imply the former?

Again, your definition of “intelligent” drives your answer to this question.

1.2 The Foundations of Artificial Intelligence

Exercise 1.2.#NTRC

There are well-known classes of problems that are intractably difficult for computers, and other classes that are provably undecidable. Does this mean that AI is impossible?

No. It means that AI systems should avoid trying to solve intractable problems. Usually, this means they can only approximate optimal behavior. Notice that humans don’t solve NP-complete problems either. Sometimes they are good at solving specific instances with a lot of structure, perhaps with the aid of background knowledge. AI systems should attempt to do the same.

Exercise 1.2.#SLUG

The neural structure of the sea slug *Aplysia* has been widely studied (first by Nobel Laureate Eric Kandel) because it has only about 20,000 neurons, most of them large and easily manipulated. Assuming that the cycle time for an *Aplysia* neuron is roughly the same as for a human neuron, how does the computational power, in terms of memory updates per second, compare with the personal computer described in Figure 1.2?

Depending on what you want to count, the computer has a thousand to a million times more storage, and a thousand times more operations per second.

Exercise 1.2.#INTR

How could introspection—reporting on one’s inner thoughts—be inaccurate? Could I be wrong about what I’m thinking? Discuss.

Just as you are unaware of all the steps that go into making your heart beat, you are also unaware of most of what happens in your thoughts. You do have a conscious awareness of some of your thought processes, but the majority remains opaque to your consciousness. The field of psychoanalysis is based on the idea that one needs trained professional help to analyze one’s own thoughts. Neuroscience has also shown that we are unaware of much of the activity in our brains.

1.3 The History of Artificial Intelligence

Exercise 1.3.#IQEV

Suppose we extend Evans's ANALOGY program (Evans, 1968) so that it can score 200 on a standard IQ test. Would we then have a program more intelligent than a human? Explain.

No. IQ test scores correlate well with certain other measures, such as success in college, ability to make good decisions in complex, real-world situations, ability to learn new skills and subjects quickly, and so on, but *only* if they're measuring fairly normal humans. The IQ test doesn't measure everything. A program that is specialized only for IQ tests (and specialized further only for the analogy part) would very likely perform poorly on other measures of intelligence. Consider the following analogy: if a human runs the 100m in 10 seconds, we might describe him or her as *very athletic* and expect competent performance in other areas such as walking, jumping, hurdling, and perhaps throwing balls; but we would not describe a Boeing 747 as *very athletic* because it can cover 100m in 0.4 seconds, nor would we expect it to be good at hurdling and throwing balls.

Even for humans, IQ tests are controversial because of their theoretical presuppositions about innate ability (distinct from training effects) and the generalizability of results. See *The Mismeasure of Man* (Stephen Jay Gould, 1981) or *Multiple Intelligences: the Theory in Practice* (Howard Gardner, 1993) for more on IQ tests, what they measure, and what other aspects there are to "intelligence."

Exercise 1.3.#PRMO

Some authors have claimed that perception and motor skills are the most important part of intelligence, and that "higher level" capacities are necessarily parasitic—simple add-ons to these underlying facilities. Certainly, most of evolution and a large part of the brain have been devoted to perception and motor skills, whereas AI has found tasks such as game playing and logical inference to be easier, in many ways, than perceiving and acting in the real world. Do you think that AI's traditional focus on higher-level cognitive abilities is misplaced?

Certainly perception and motor skills are important, and it is a good thing that the fields of vision and robotics exist (whether or not you want to consider them part of "core" AI). But given a percept, an agent still has the task of "deciding" (either by deliberation or by reaction) which action to take. This is just as true in the real world as in artificial micro-worlds such as chess-playing. So computing the appropriate action will remain a crucial part of AI, regardless of the perceptual and motor system to which the agent program is "attached." On the other hand, it is true that a concentration on micro-worlds has led AI away from the really interesting environments such as those encountered by self-driving cars.

Exercise 1.3.#WINT

Several “AI winters,” or rapid collapses in levels of economic and academic activity (and media interest) associated with AI, have occurred. Describe the causes of each collapse and of the boom in interest that preceded it.

In addition to the information in the chapter, ? (?), ? (?), and ? (?) provide ample starting material for the aspiring historian of AI. One can identify at least three AI winters (although the phrase was not applied to the first one, because the original phrase **nuclear winter** did not emerge until the early 1980s).

- a. As noted in the chapter, research funding dried up in the early 1970s in both the US and UK. The ostensible reason was failure to make progress on the rather lavish promises of the 1960s, particularly in the areas of neural networks and machine translation. In 1970, the US Congress curtailed most AI funding from ARPA, and in 1973 the Lighthill report in the UK ended funding for all but a few researchers. Lighthill referred particularly to the difficulties of overcoming the combinatorial explosion.
- b. In the late 1980s, the expert systems boom ended, due largely to the difficulty and expense of building and maintaining expert systems for complex applications, the lack of a valid uncertainty calculus in these systems, and the lack of interoperability between AI software and hardware and existing data and computation infrastructure in industry.
- c. In the early 2000s, the end of the dot-com boom also ended an upsurge of interest in the use of AI systems in the burgeoning online ecosystem. AI systems had been used for such tasks as information extraction from web pages to support shopping engines and price comparisons; various kinds of search engines; planning algorithms for achieving complex goals requiring several steps and combining information from multiple web pages; and converting human-readable web pages into machine-readable database tuples to allow global information aggregation, as in citation databases constructed from online pdf files.

It is also interesting to explore the extent to which the winters were due to over-optimistic and exaggerated claims by AI researchers or to over-enthusiasm and over-interpretation of the significance of early results by funders and investors.

Exercise 1.3.#DLAI

The resurgence of interest in AI in the 2010s is often attributed to deep learning. Explain what deep learning is, how it relates to AI as a whole, and where the core technical ideas actually originated.

Deep learning is covered in Section 1.3.8, where it is defined as “machine learning using multiple layers of simple, adjustable computing elements.” Thus, it is a particular branch of machine learning, which is itself a subfield of AI. Since many AI systems do not use learning at all, and there are many effective machine learning techniques that are unrelated to deep learning, the view (often expressed in popular articles on AI) that deep learning has “replaced” AI is wrong for multiple reasons.

Some of the key technical ideas are the following (see also Chapter 21):

- *Networks of simple, adjustable computing elements*: ? (?), drawing on ? (?), ?).
- *Backpropagation*, i.e., a localized way of computing gradients of functions expressed by networks of computing elements, based on the chain rule of calculus: ? (?), ?). For neural network learning specifically, ? (?) preceded by more than a decade the much better-known work on ? (?).
- *Convolutional networks* with many copies of small subnetworks, all sharing the same patterns of weights: This is usually attributed to work in the 1990s on handwritten digit recognition by ? (?) at AT&T Bell Labs. ? (?) acknowledge the influence of the **neocognitron** model (?), which in turn was inspired by the neuroscience work of ? (?), ?).
- *Stochastic gradient descent* to facilitate learning in deep networks: as described in the historical notes section of Chapter 19, ? (?) explored stochastic approximations to gradient methods, including convergence properties; the first application to neural networks was by ? (?) and independently by ? (?) in their ADALINE networks.

1.4 The State of the Art

Exercise 1.4.#SOTA

Examine the AI literature to discover whether the following tasks can currently be solved by computers:

- a. Playing a decent game of table tennis (Ping-Pong).
- b. Driving in the center of Cairo, Egypt.
- c. Driving in Victorville, California.
- d. Buying a week's worth of groceries at the market.
- e. Buying a week's worth of groceries on the Web.
- f. Playing a decent game of bridge at a competitive level.
- g. Discovering and proving new mathematical theorems.
- h. Writing an intentionally funny story.
- i. Giving competent legal advice in a specialized area of law.
- j. Translating spoken English into spoken Swedish in real time.
- k. Performing a complex surgical operation.

For the currently infeasible tasks, try to find out what the difficulties are and predict when, if ever, they will be overcome.

- a. (ping-pong) A reasonable level of proficiency was achieved by Andersson's robot (Andersson, 1988).
- b. (driving in Cairo) No. Although there has been a lot of progress in automated driving, they operate in restricted domains: on the highway, in gated communities, or in well-mapped cities with limited traffic problems. Driving in downtown Cairo is too unpredictable for any of these to work.

- c. (driving in Victorville, California) Yes, to some extent, as demonstrated in DARPA's Urban Challenge. Some of the vehicles managed to negotiate streets, intersections, well-behaved traffic, and well-behaved pedestrians in good visual conditions.
- d. (shopping at the market) No. No robot can currently put together the tasks of moving in a crowded environment, using vision to identify a wide variety of objects, and grasping the objects (including squishable vegetables) without damaging them. The component pieces are nearly able to handle the individual tasks, but it would take a major integration effort to put it all together.
- e. (shopping on the web) Yes. Software robots are capable of handling such tasks, particularly if the design of the web grocery shopping site does not change radically over time.
- f. (bridge) Yes. Programs such as GIB now play at a solid level.
- g. (theorem proving) Yes. For example, the proof of Robbins algebra.
- h. (funny story) No. While some computer-generated prose and poetry is hysterically funny, this is invariably unintentional, except in the case of programs that echo back prose that they have memorized.
- i. (legal advice) Yes, in some cases. AI has a long history of research into applications of automated legal reasoning. Two outstanding examples are the Prolog-based expert systems used in the UK to guide members of the public in dealing with the intricacies of the social security and nationality laws. The social security system is said to have saved the UK government approximately \$150 million in its first year of operation. However, extension into more complex areas such as contract law awaits a satisfactory encoding of the vast web of common-sense knowledge pertaining to commercial transactions and agreement and business practices.
- j. (translation) Yes. Although translation is not perfect, it is serviceable, and is used by travellers every day.
- k. (surgery) Yes. Robots are increasingly being used for surgery, although always under the command of a doctor. Robotic skills demonstrated at superhuman levels include drilling holes in bone to insert artificial joints, suturing, and knot-tying. They are not yet capable of planning and carrying out a complex operation autonomously from start to finish.

Exercise 1.4.#CONT

Various subfields of AI have held contests by defining a standard task and inviting researchers to do their best. Examples include the ImageNet competition for computer vision, the DARPA Grand Challenge for robotic cars, the International Planning Competition, the Robocup robotic soccer league, the TREC information retrieval event, and contests in machine translation, speech recognition, and other fields. Investigate one of these contests, and describe the progress made over the years. To what degree have the contests advanced the state of the art in AI? Do what degree do they hurt the field by drawing energy away from new ideas?

The progress made in these contests is a matter of fact, but the impact of that progress is a matter of opinion. Some examples:

- **DARPA Grand Challenge for Robotic Cars:** In 2004 the Grand Challenge was a 240 km race through the Mojave Desert. It clearly stressed the state of the art of autonomous driving, and in fact no competitor finished the race. The best team, CMU, completed only 12 of the 240 km. In 2005 the race featured a 212km course with fewer curves and wider roads than the 2004 race. Five teams finished, with Stanford finishing first, edging out two CMU entries. This was hailed as a great achievement for robotics and for the Challenge format. In 2007 the Urban Challenge put cars in a city setting, where they had to obey traffic laws and avoid other cars. This time CMU edged out Stanford. The competition appears to have been a good testing ground to put theory into practice, something that the failures of 2004 showed was needed. But it is important that the competition was done at just the right time, when there was theoretical work to consolidate, as demonstrated by the earlier work by Dickmanns (whose VaMP car drove autonomously for 158km in 1995) and by Pomerleau (whose Navlab car drove 5000km across the USA, also in 1995, with the steering controlled autonomously for 98% of the trip, although the brakes and accelerator were controlled by a human driver).
- **International Planning Competition:** In 1998, five planners competed: Blackbox, HSP, IPP, SGP, and STAN. The result page (<ftp://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html>) stated “all of these planners performed very well, compared to the state of the art a few years ago.” Most plans found were 30 or 40 steps, with some over 100 steps. In 2008, the competition had expanded quite a bit: there were more tracks (satisficing vs. optimizing; sequential vs. temporal; static vs. learning). There were about 25 planners, including submissions from the 1998 groups (or their descendants) and new groups. Solutions found were much longer than in 1998. In sum, the field has progressed quite a bit in participation, in breadth, and in power of the planners. In the 1990s it was possible to publish a Planning paper that discussed only a theoretical approach; now it is necessary to show quantitative evidence of the efficacy of an approach. The field is stronger and more mature now, and it seems that the planning competition deserves some of the credit. However, some researchers feel that too much emphasis is placed on the particular classes of problems that appear in the competitions, and not enough on real-world applications.
- **Robocup Robotic Soccer:** This competition has proved extremely popular, attracting 300–500 teams from 40–50 countries since the mid-2000s (up from 38 teams from 11 countries in 1997). The “standard platform” league, originally based on the Sony AIBO four-legged robot, switched to the humanoid Aldebaran Nao robot in 2009. There are also small and mid-size leagues in which teams are free to design their own robots provided they satisfy certain physical constraints. The competition has served to increase interest and participation in robotics and has led to some advances in both robotics and AI (particularly related to learning in team games). Victory on competitions often depends less on AI than on specific tricks for improving ball-handling and shooting. The long-term goal is to defeat a human World-Cup-winning team by 2050, although the precise details of ensuring safety of human participants remain to be worked out.
- **TREC Information Retrieval Conference:** This is one of the oldest competitions,

started in 1992. The competitions have served to bring together a community of researchers, have led to a large literature of publications, and have seen progress in participation and in quality of results over the years. In the early years, TREC served its purpose as a place to do evaluations of retrieval algorithms on text collections that were large for the time. However, starting around 2000 TREC became less relevant as the advent of the World Wide Web created a corpus that was available to anyone and was much larger than anything TREC had created, and the development of commercial search engines surpassed academic research.

- **ImageNet:** The ImageNet database is a hand-labelled collection of over 14 million images in over 20,000 categories (?). The ImageNet competition (more properly, the ImageNet Large Scale Visual Recognition Challenge or ILSVRC) is based on a subset of 1,000 categories, 90 of which are breeds of dog. The availability of such large training sets is often asserted to be a contributing factor in the emergence of deep learning. In the 2012 competition, the decisive win by AlexNet (?) set off a frenzy of research that has reduced error rates on ILSVRC well below the human level of about 5%. Research by (?), however, suggests that for many types of deep learning the progress may be illusory: in many cases, the object is “recognized” through the color and spatial distribution of background pixels such as the grass on which a dog is sitting.

Overall, we see that whatever you measure is bound to increase over time. For most of these competitions, the measurement was a useful one, and the state of the art has progressed. In the case of ICAPS, some planning researchers worry that too much attention has been lavished on the competition itself. In some cases, progress has left the competition behind, as in TREC, where the resources available to commercial search engines outpaced those available to academic researchers. In this case the TREC competition was useful—it helped train many of the people who ended up in commercial search engines—and in no way drew energy away from new ideas. In computer vision, as in planning, improved performance on competition benchmarks has become almost essential for a new idea to be published, which many argue is a serious obstacle to research and may lead to an entire field become stuck in a dead-end approach.

Exercise 1.4.#DROB

Investigate the state of the art for domestic robots: what can be done (with what assumptions and restrictions on the environment) and what problems remain unsolved? Where is research most needed?

Creating a fully general domestic robot that will work “out of the box” in any household remains a distant goal. At the time of writing (mid-2021), robot demonstrations for a number of specific tasks have been given. Some examples:

- Folding laundry (?), <https://youtu.be/gy5g33S0Gzo>. The robot assumes that a pile contains only rectangular pieces of cloth and requires a uniform green background. Subsequent work allowed a PR2 to complete almost the entire laundry cycle with a fairly wide assortment of laundry (?), https://www.nsf.gov/discoveries/disc_summ.jsp?casa_id=4000. However, a demonstration exhibit at the Victoria & Albert Museum in London showed

Exercises 1 Introduction

the fragility of the solution: the commercial Baxter robot broke down too often for the exhibit to function for more than a fraction of the time.

- Loading and unloading a dishwasher: the subject of academic research-lab demos since the early 2000s (CMU’s HERB robot, for example), this one is creeping closer to commercial realization, with companies such as Samsung showing carefully edited demos of robotic products that are not yet available. As with other such demos, the technology is not ready to take on an arbitrary kitchen and dishwasher. Also, many robots are far from waterproof.
- Cooking: (?) describe an integrated system intended to download recipes from the Internet, turn them into executable plans, and carry out those plans in a real kitchen. Only quite preliminary results were achieved: <https://www.csail.mit.edu/node/6480>. As with washing up, cooking is very messy and current general-purpose robots need to be carefully wrapped in plastic protective gear. There are also commercial “robot cooks” that claim to cook dishes in the home. Typically these are special-purpose systems that require specially prepared and sized ingredients and are generally not robust to failure (see, e.g., <https://www.youtube.com/watch?v=M8r0gmQXm1Y> and https://www.youtube.com/watch?v=GyEHRXA_aA4). Again, we are far from having a robot that go into anyone’s kitchen with a bag of ingedients from the supermarket and make dinner.

Successful teleoperation demos (where a human controls a robot to carry out tasks such as opening a beer bottle) suggest that the problems do not lie primarily with robotic hardware (except for the need to design robots that are immune to water, grease, and solid ingredients. Perception has advanced considerably, but it is still difficult for a robot to analyze a pile of ingredients in a bowl and judge where the surface is and how well mixed the ingredients are. Spatial reasoning involving continuous, irregular shapes and liquid, semiliquid, and powdered ingredients or complex objects made from cloth, leather, etc., is quite weak. Also lacking is the ability to plan and manage domestic activities on a continuous basis, with constant interruptions from humans who rearrange the world.

Exercise 1.4.#JRNL

The vast majority of academic publications and media articles report on successes of AI. Examine recent articles describing failures of AI. (?) and (?) are good examples, but feel free to find your own.) How serious are the problems identified? Are they examples of overclaiming of or fundamental problems with the technical approach?

?(?) define *overinterpretation* as occurring when a machine learning algorithm finds predictive regularities in parts of the input data that are semantically irrelevant. Those regularities reflect particular properties of the training and test data, such as when all photos of Norfolk Terriers are taken with the dog sitting in various poses a particular crimson carpet. In that case, simply recognizing the presence of a few pixels of crimson suffices to identify the breed of dog. They show that this problem arises quite frequently with modern deep learning systems applied to standard data sets such as ImageNet and CIFAR-10. For some categories, such as “airplane” in CIFAR-10, a single pixel suffices to classify the object. In

almost all cases, the trained network labels images with high confidence using only a very small subset of fairly randomly distributed pixels that, to a human, is unintelligible. This seems to reveal a fundamental problem both with the technology (the networks appear to have too much capacity and no notion of what they are looking for) and the standard train/test methodology (which fails to detect the non-robustness of the learned classifiers). Put another way, the network does not know that “Norfolk Terrier” is a breed of dog rather than a carpet color (“Crimson Carpet”), so the task solved by the machine learning algorithm is not the one solved by a human.

? (?) describe a similar issue: for any given training set, there are typically vast numbers of learned network configurations that give equally good (or even perfect) performance on held-out data. They call this *underspecification* and note that it leads to poor performance after deployment in a wide range of real-world applications including computer vision, medical imaging, natural language processing, clinical risk prediction based on electronic health records, and medical genomics. They conclude that machine learning pipelines must be constrained by strong semantic priors if learning is to be effective and reliable (?; see also).

For a paper highly critical of the use of machine learning to diagnose COVID-19 from chest X-rays, see ? (?). The authors selected 62 of the most promising and carefully documented studies from a total of 2,212 published in 2020, but found that “none of the models identified are of potential clinical use due to methodological flaws and/or underlying biases.” The problems identified were primarily failings on the part of the studies’ authors, who may not have been aware of the stringent criteria for real-world deployment of high-stakes diagnostic tools. However, even a methodologically perfect study using deep learning might well have failed for reasons given in the preceding two paragraphs.

1.5 Risks and Benefits of AI

Exercise 1.5.#PRIN

Find and analyze at least three sets of proposed principles for the governance of AI. What do the sets of principles have in common? How do they differ? How implementable are these principles?

There are several hundred published sets of principles: some of the best-known are the OECD, Beijing, and Asilomar principles. ? (?), and ? (?) provide useful analytical surveys and summaries.

Exercise 1.5.#EURG

Study the 2021 EU “Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)” (or its final version, if available). Which provisions appear to have the most direct and tangible impact on what kinds of AI systems can be deployed?

If students have never seen legislation before, this will be an eye-opening exercise. Most

Exercises 1 Introduction

of the document stipulates practices in testing, documentation, etc. Only a few of the clauses have tangible impact—most obviously, those concerning facial recognition, impersonation of humans, and deepfakes. In these three areas, the rules are quite strict compared to those in force in most other parts of the world.

Exercise 1.5.#LATM

Summarize the pros and cons of allowing the development, deployment, and use of lethal autonomous weapons.

A good place to start is the Congressional Research Service report “International Discussions Concerning Lethal Autonomous Weapon Systems,” dated October 15, 2020. Unfortunately the report leaves out the principal argument against lethal autonomous weapons: they will become cheap, scalable weapons of mass destruction that will proliferate easily and are likely to be used against civilian populations.

Exercise 1.5.#BIAS

Many researchers have pointed to the possibility that machine learning algorithms will produce classifiers that display racial, gender, or other forms of bias. How does this bias arise? Is it possible to constrain machine learning algorithms to produce rigorously fair predictions?

Bias in ML occurs largely because of two things: first, the bias that exists in data generated by a biased society, and second, specifying an incorrect objective for machine learning algorithms—namely, maximizing agreement with the training data. This is not the real objective: in fact, we care about fairness too.

It’s possible to define various measures of fairness and to design algorithms that respect them. See ? (?) for a survey. Unfortunately, it is not possible to simultaneously satisfy all “reasonable” definitions of fairness, and there may be some sacrifice in accuracy.

Exercise 1.5.#SIFI

Examine at least three science-fiction movies in which AI systems threaten to (or actually do) “take control.” In the movie, does the takeover stem from “spooky emergent consciousness” or from a poorly defined objective? If the latter, what was it?

Here are three examples:

- a. *Colossus—The Forbin Project*: A US defense computer tasked with ensuring peace and security encounters a Soviet computer with the same goal. They exchange information and decide that the goal is best achieved by jointly controlling all nuclear weapons and using threats of destruction to force humans to drop all aggressive war plans. Here, the problem is clearly a poorly defined objective.
- b. *Ex Machina*: Ava, a very human-like android, brilliantly engineers her escape from a remote facility by outwitting her human captors and allies. Although it is never stated

explicitly in the film, her objective seems to be in places populated by large numbers of people; she is portrayed as having this objective as a result of a process of consciousness emerging from the sum total of human interactions recorded in a global social media platform.

- c. *Transcendence*: Will Caster, a Berkeley AI professor, is gunned down by anti-AI terrorists. Before he dies, his brain is uploaded to a quantum supercomputer. The machine becomes conscious and begins to quickly outrun the human race, threatening to take over the world. Here, the risk comes from the possibility that the machine's conscious goals differ from those of its human progenitor.

EXERCISES 2

INTELLIGENT AGENTS

2.1 Agents and Environments

Exercise 2.1.#DFAG

Define in your own words the following terms: agent, environment, sensor, actuator, percept, agent function, agent program.

Mobile agent

The following are just some of the many possible definitions that can be written:

- *Agent*: an entity that perceives and acts; or, one that *can be viewed* as perceiving and acting. Essentially any object qualifies; the key point is the way the object implements an agent function. (Note: some authors restrict the term to *programs* that operate *on behalf of* a human, or to programs that can cause some or all of their code to run on other machines on a network, as in **mobile agents**.)
- *Environment*: the world in which the agent is situated, on which the agent's actuators produce effects and from which the agent's sensors receive percepts.
- *Sensor*: part of the agent whose state is affected by the environment and whose state can enter into the agent's computations directly as a value.
- *Actuator*: part of the agent whose state can be set by the agent's computations and affects the environment.
- *Percept*: the observed value of the sensor state (usually, all the sensors taken together) at a given time.
- *Agent function*: given an agent viewed as a fixed physical object, the agent function that specifies the agent's action in response to every possible percept sequence.
- *Agent program*: that program which, combined with a machine architecture, produces an agent function. In our simple designs, the program takes a new percept on each invocation and returns an action. Note that the agent program is generally not simply "implementing" the agent function exactly, because the program may take more than one time step to return an action.

Exercise 2.1.#AGEX

For each of the following agents, specify the sensors, actuators, and environment: microwave oven, chess program, autonomous supply delivery plane.

The following are just some of the many possible definitions that can be written:

- *Microwave oven*: Here it is important to view the oven as the agent, not the human who is using it; this means the buttons are sensors, not actuators! It might seem that there are no decisions to make but the internal controller logic can be quite complex, particularly if there are additional sensors. For safety, no power should be supplied if the door is open!
 - *Sensors*: Button state, door-open sensor. Some microwaves have humidity sensors, temperature sensors, or acoustic sensors (for popcorn).
 - *Actuators*: Turn on/turn off/adjust microwave power; sound completion alarm; indicate illegal button state.
 - *Environment*: The contents of the oven and the state of the door (open/closed).
- *Chess program*:
 - *Sensors*: A typical program accepts keyboard, mouse, or touchscreen input indicating the opponent's intended move, resignation, hint request, etc.
 - *Actuators*: Display a legal move, or the board resulting from the move.
 - *Environment*: At a minimum, the environment includes a representation of the board state. There are some subtle issues related to whether this is identical to the program's own internal representation or the displayed board that is accessible to the opponent. A more sophisticated program might include the human opponent as part of the environment, and try to build a model of that human's playing style and ability in order to improve its chances of winning or make the game more entertaining/instructive.
- *Autonomous supply delivery plane*: The answers here are quite similar to those for the autonomous taxi in Section 2.3. Details can be found in numerous online sources describing current autonomous delivery systems.
 - *Sensors*: GPS, air speed (e.g., Pitot tube), altimeter (actually an air pressure sensor, so reported altitude depends on meteorological conditions), sensors reporting state of each actuator, 3-axis accelerometer, possibly a camera.
 - *Actuators*: power to propeller, ailerons, elevators, rudder; drop payload.
 - *Environment*: geographical area of flight, atmospheric conditions.

Exercise 2.1.#AGFN

This exercise explores the differences between agent functions and agent programs.

- a. Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.
- b. Are there agent functions that cannot be implemented by any agent program?
- c. Given a fixed machine architecture, does each agent program implement exactly one agent function?
- d. Given an architecture with n bits of storage, how many different possible agent programs are there?
- e. Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?

Although these questions are very simple, they hint at some very fundamental issues. Our answers are for the simple agent designs for *static* environments where nothing happens while the agent is deliberating; the issues get even more interesting for dynamic environments.

- a. Yes; take any agent program and insert null statements that do not affect the output.
- b. Yes; the agent function might specify that the agent print *true* when the percept is a Turing machine program that halts, and *false* otherwise. (Note: in dynamic environments, for machines of less than infinite speed, the rational agent function may not be implementable; e.g., the agent function that always plays a winning move, if any, in a game of chess.)
- c. Yes; the agent's behavior is fixed by the architecture and program.
- d. There are 2^n agent programs, although many of these will not run at all. (Note: Any given program can devote at most n bits to storage, so its internal state can distinguish among only 2^n past histories. Because the agent function specifies actions based on percept histories, there will be many agent functions that cannot be implemented because of lack of memory in the machine.)
- e. It depends on the program and the environment. If the environment is dynamic, speeding up the machine may mean choosing different (perhaps better) actions and/or acting sooner. If the environment is static and the program pays no attention to the passage of elapsed time, the agent function is unchanged.

2.2 Good Behavior: The Concept of Rationality

Exercise 2.2.#PRMT

Suppose that the performance measure is concerned with just the first T time steps of the environment and ignores everything thereafter. Show that a rational agent's action may depend not just on the state of the environment but also on the time step it has reached.

This question tests the student's understanding of environments, rational actions, and performance measures. Any sequential environment in which rewards may take time to arrive will work, because then we can arrange for the reward to be "over the horizon." Suppose that in any state there are two action choices, a and b , and consider two cases: the agent is in state s at time T or at time $T - 1$. In state s , action a reaches state s' with reward 0, while action b reaches state s again with reward 1; in s' either action gains reward 10. At time $T - 1$, it's rational to do a in s , with expected total reward 10 before time is up; but at time T , it's rational to do b with total expected reward 1 because the reward of 10 cannot be obtained before time is up.

Students may also provide common-sense examples from real life: investments whose payoff occurs after the end of life, exams where it doesn't make sense to start the high-value question with too little time left to get the answer, and so on.

The environment state can include a clock, of course; this doesn't change the gist of the answer—now the action will depend on the clock as well as on the non-clock part of the state—but it does mean that the agent can never be in the same state twice.

Exercise 2.2.#VACR

Let us examine the rationality of various vacuum-cleaner agent functions.

- a. Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational under the assumptions listed on page 40.
- b. Describe a rational agent function for the case in which each movement costs one point. Does the corresponding agent program require internal state?
- c. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? If not, why not?

Notice that for our simple environmental assumptions we need not worry about quantitative uncertainty.

- a. It suffices to show that for all possible actual environments (i.e., all dirt distributions and initial locations), this agent cleans the squares at least as fast as any other agent. This is trivially true when there is no dirt. When there is dirt in the initial location and none in the other location, the world is clean after one step; no agent can do better. When there is no dirt in the initial location but dirt in the other, the world is clean after two steps; no agent can do better. When there is dirt in both locations, the world is clean after three steps; no agent can do better. (Note: in general, the condition stated in the first sentence of this answer is much stricter than necessary for an agent to be rational.)
- b. The agent in (a) keeps moving backwards and forwards even after the world is clean. It is better to do *NoOp* once the world is clean (the chapter says this). Now, since the agent's percept doesn't say whether the other square is clean, it would seem that the agent must have some memory to say whether the other square has already been cleaned. To make this argument rigorous is more difficult—for example, could the agent arrange things so that it would only be in a clean left square when the right square was already clean? As a general strategy, an agent *can* use the environment itself as a form of **external memory**—a common technique for humans who use things like appointment calendars and knots in handkerchiefs. In this particular case, however, that is not possible. Consider the reflex actions for $[A, Clean]$ and $[B, Clean]$. If either of these is *NoOp*, then the agent will fail in the case where that is the initial percept but the other square is dirty; hence, neither can be *NoOp* and therefore the simple reflex agent is doomed to keep moving. In general, the problem with reflex agents is that they have to do the same thing in situations that look the same, even when the situations are actually quite different. In the vacuum world this is a big liability, because every interior square (except home) looks either like a square with dirt or a square without dirt.
External memory
- c. If we consider asymptotically long lifetimes, then it is clear that learning a map (in some form) confers an advantage because it means that the agent can avoid bumping into walls. It can also learn where dirt is most likely to accumulate and can devise an optimal inspection strategy. The precise details of the exploration method needed to construct a complete map appear in Chapter 4; methods for deriving an optimal

inspection/cleanup strategy are in Chapter 22.

Exercise 2.2.#KPER

Describe three different task environments in which the performance measure is easy to specify completely and correctly, and three in which it is not.

Obviously there are many possible answers here. For the “easy” case, it makes sense to consider artificially defined task environments: electronic calculators (answers correct to the required number of significant digits, elapsed time), chess programs (win/draw/loss subject to time constraints), spider solitaire (number of suits completed, number of moves, elapsed time). But even in these environments, one must be careful, because such performance measures ignore *all other considerations*. For example, a general-purpose intelligent agent that has the ability to display messages or access the internet might improve its chess performance by acquiring additional hardware. Marvin Minsky pointed out that a machine designed to calculate as many digits of π as possible could take over the world to do so.

For the “hard” case, almost any task involving humans will do: tutor, automated taxi, digital personal assistant, military strategist, and so on. Although Section 2.3 lists the elements of the taxi’s performance measure (reaching destination, fuel consumption, wear and tear, trip time, traffic laws, politeness to other drivers, safety, passenger comfort, profit), the appropriate tradeoffs among these are far from clear. For example, if the taxi is 200 yards from the destination but stuck in a traffic jam that will take 20 minutes to clear, should it suggest to the passengers that they walk the remaining 200 yards? And how does this decision depend on the weather and general safety and legality of this premature dropoff location?

In addition to difficult tradeoffs, there will typically be attributes that are omitted from the performance measure that do in fact matter and can be affected by the agent. Obviously it doesn’t make sense to ask students to list the attributes that are omitted, because they could simply include them! Instead, one might ask which attributes are “obvious” and which are “non-obvious,” i.e., likely to be omitted on a first or second pass.

Exercise 2.2.#EVSA

Write an essay on the relationship between evolution and one or more of autonomy, intelligence, rationality, and learning.

Exercise 2.2.#AGTF

For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

- a. An agent that senses only partial information about the state cannot be perfectly rational.
- b. There exist task environments in which no pure reflex agent can behave rationally.
- c. There exists a task environment in which every agent is rational.
- d. The input to an agent program is the same as the input to the agent function.

- e. Every agent function is implementable by some program/machine combination.
- f. Suppose an agent selects its action uniformly at random from the set of possible actions.
There exists a deterministic task environment in which this agent is rational.
- g. It is possible for a given agent to be perfectly rational in two distinct task environments.
- h. Every agent is rational in an unobservable environment.
 - i. A perfectly rational poker-playing agent never loses.
 - j. For a simple reflex agent in a partially observable environment, a randomized policy can outperform any deterministic policy.
- k. There is a model-based reflex agent that can remember all of its percepts.
- l. Suppose agent A1 is rational and agent A2 is irrational. There exists a task environment where A2's actual score will be greater than A1's actual score.

- a. *An agent that senses only partial information about the state cannot be perfectly rational.*

False. Perfect rationality refers to the ability to make good decisions given the sensor information received. Moreover, in *some* environments, an agent can deliberately ignore part of its sensor information and still be perfectly rational. For example, a physical chess agent can ignore flecks of dust on the chessboard and pretty much anything that isn't on the chessboard or the clock.

- b. *There exist task environments in which no pure reflex agent can behave rationally.*

True. A pure reflex agent ignores previous percepts, so cannot obtain an optimal state estimate in a partially observable environment. For example, correspondence chess is played by sending moves; if the other player's move is the current percept, a reflex agent could not keep track of the board state and would have to respond to, say, "a4" in the same way regardless of the position in which it was played.

- c. *There exists a task environment in which every agent is rational.*

True. For example, in an environment with a single state, such that all actions have the same reward, it doesn't matter which action is taken. More generally, any environment that is reward-invariant under permutation of the actions will satisfy this property.

- d. *The input to an agent program is the same as the input to the agent function.*

False. The agent function, notionally speaking, takes as input the entire percept sequence up to that point, whereas the agent program takes the current percept only.

- e. *Every agent function is implementable by some program/machine combination.*

False. For example, the environment may contain Turing machines and input tapes and the agent's job is to solve the halting problem; there is an agent *function* that specifies the right answers, but no agent program can implement it. Another example would be an agent function that requires solving intractable problem instances of arbitrary size in constant time.

- f. *Suppose an agent selects its action uniformly at random from the set of possible actions.*

There exists a deterministic task environment in which this agent is rational.

True. This is a special case of (c); if it doesn't matter which action you take, selecting

randomly is rational.

- g.** *It is possible for a given agent to be perfectly rational in two distinct task environments.*
 True. For example, we can arbitrarily modify the parts of the environment that are unreachable by any optimal policy as long as they stay unreachable.
- h.** *Every agent is rational in an unobservable environment.*
 False. Some actions are stupid—and the agent may know this if it has a model of the environment—even if one cannot perceive the environment state.
- i.** *A perfectly rational poker-playing agent never loses.*
 False. Unless it draws the perfect hand, the agent can always lose if an opponent has better cards. This can happen for game after game. The correct statement is that the agent’s expected winnings are nonnegative.
- j.** *For a simple reflex agent in a partially observable environment, a randomized policy can outperform any deterministic policy.*
 True. Some vacuum robots use this idea to get themselves “unstuck” from corners. A partially observable environment can have hidden state that makes any particular deterministically chosen action fail when executed in response to a given percept, whereas a randomized policy may eventually chose the right action (if there is one). As an example, consider a reflex agent that needs the PIN to unlock a phone. A randomized agent will eventually emit the correct sequence, whereas a deterministic agent can only emit the same sequence over and over.
- k.** *There is a model-based reflex agent that can remember all of its percepts.*
 True. A model-based agent updates an internal state representation—its “memory”—for each new percept. Memorizing the percepts is possible (assuming unbounded memory, which is a reasonable caveat). This may not be the most perspicuous representation of the current state of the world, but any other such representation of the current state could be computed from it on demand, for example by running any other model-based reflex agent’s state estimation algorithm on the memorized percept sequence.
- l.** *Suppose agent A1 is rational and agent A2 is irrational. There exists a task environment where A2’s actual score will be greater than A1’s actual score.*
 True. Rational decisions are defined by expected outcomes, not actual outcomes. A1 might just be unlucky.

2.3 The Nature of Environments

Exercise 2.3.#PEAS

For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties listed in Section 2.3.2.

- Playing soccer.
- Exploring the subsurface oceans of Titan.
- Shopping for used AI books on the Internet.
- Playing a tennis match.

- Practicing tennis against a wall.
- Performing a high jump.
- Knitting a sweater.
- Bidding on an item at an auction.

Many of these can actually be argued either way, depending on the level of detail and abstraction.

- A. Partially observable, stochastic, sequential, dynamic, continuous, multi-agent.
- B. Partially observable, stochastic, sequential, dynamic, continuous, single agent (unless there are alien life forms that are usefully modeled as agents).
- C. Partially observable, deterministic, sequential, static, discrete, single agent. This can be multi-agent and dynamic if we buy books via auction, or dynamic if we purchase on a long enough scale that book offers change.
- D. Fully observable, stochastic, episodic (every point is separate), dynamic, continuous, multi-agent.
- E. Fully observable, stochastic, episodic, dynamic, continuous, single agent.
- F. Fully observable, stochastic, sequential, static, continuous, single agent.
- G. Fully observable, deterministic, sequential, static, continuous, single agent.
- H. Fully observable, strategic, sequential, static, discrete, multi-agent.

Exercise 2.3.#VCES

Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in Figure 2.2 and specified on page 40. Your implementation should be modular so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily. (*Note:* for some choices of programming language and operating system there are already implementations in the online code repository.)

The code repository implements a vacuum-cleaner environment. Students can easily extend it to generate different shaped rooms, obstacles, dirt generation processes, sensor suites, and so on.

Exercise 2.3.#ENVP

For each of the following task environment properties, rank the example task environments from most to least according to how well the environment satisfies the property. Lay out any assumptions you make to reach your conclusions.

- a. *Fully Observable:* driving; document classification; tutoring a high-school student in calculus; skin cancer diagnosis from images.

- b. *Continuous*: driving; spoken conversation; written conversation; climate engineering by stratospheric aerosol injection.
- c. *Stochastic*: driving; sudoku; poker; soccer.
- d. *Static*: chat room; checkers; tax planning; tennis.

- a. *Fully Observable*: document classification > skin cancer diagnosis from images > driving > tutoring a high-school student in calculus.

Document classification is a fairly canonical example of a (non-sequential) observable problem, because the correct classification depends almost entirely on the visible text of the document itself. There might be a slight influence from “provenance” information (date, authorship, etc.) that may not be directly observable. Skin cancer diagnosis can sometimes be done well from an image of the lesion, but other factors such as patient age, changes in the lesion over time, medical history, and family history can be important. Driving is often considered to be observable because we imagine that we are making decisions based on what we see, but (1) velocity and turn signal status of other vehicles can be judged only from multiple image frames, and (2) assessing the intended actions of other vehicles may require accumulating information over an extended period—e.g., to determine if a vehicle is stopped or broken down, driving slowly or looking for an address or a parking spot, turning left or has forgotten to turn off the turn signal. Other vehicles, hedges, fog, and so on can obscure visual access to important aspects of the driving environment. Tutoring is almost completely unobservable: what matters is the student’s level of understanding, learning style, basic math skills, etc. Clues must be gathered over days, weeks, and months.

- b. *Continuous*: climate engineering > driving > spoken conversation > written conversation.

Climate engineering by aerosol injection is quintessentially continuous: the engineer must decide how much to inject, where, and when, and all of these are continuous quantities. The control actions of driving are mostly continuous (steering, acceleration/braking) but there are discrete elements (turn signal, headlights). More importantly, the problem is usually handled using discrete high-level actions (change lanes left, take exit, etc.) that have implementations as continuous control problems. This kind of discrete/continuous hierarchy is very common; playing chess in the physical world is a perfect example. Spoken conversation is closer to chess than driving: roughly speaking, we choose the discrete words to say and delegate the saying to continuous motor control routines. Prosody (volume, pitch, and speed variation) is, however, an important continuous element in how we speak that is largely absent from written communication.

- c. *Stochastic*: poker > soccer > driving > sudoku.

In poker, nearly everything is determined by the fall of the cards, which is entirely stochastic from the viewpoint of the players. Both soccer and driving contain elements that are fairly deterministic, such as the flight of the ball and the response of the engine, and elements that are stochastic, such as tire punctures and the outcomes of tackles. Yet typically one can make reasonably reliable driving plans over many minutes, whereas it

is essentially impossible to predict the state of a soccer game one minute into the future. Sudoku, of course, is entirely deterministic.

- d. *Static*: tax planning > checkers > chat room > tennis.

While no human activity is completely static, given the finite length of our lifetimes, tax planning comes close—the typical “deadline” to get it done is often weeks or months, and the relevant aspects of the environment (life/death, number of offspring, tax law) may change even more slowly. In checkers, the world state doesn’t change until someone moves, but the clock ticks so the problem is semi-dynamic. In the chat room, long delays in replying are unacceptable, so it is a fairly real-time environment, but not nearly as real-time as tennis, where a delay of a split second often makes the difference between winning and losing a point.

2.4 The Structure of Agents

Exercise 2.4.#SIRA

Implement a simple reflex agent for the vacuum environment in Exercise VACUUM-START-EXERCISE. Run the environment with this agent for all possible initial dirt configurations and agent locations. Record the performance score for each configuration and the overall average score.

Pseudocode for a reflex agent is given in Figure 2.8. For states 1, 3, 5, 7 in Figure 4.9, the performance measures are 1996, 1999, 1998, 2000 respectively. The average is 1998.25.

Exercise 2.4.#VCPE

Consider a modified version of the vacuum environment in Exercise VACUUM-START-EXERCISE, in which the agent is penalized one point for each movement.

- a. Can a simple reflex agent be perfectly rational for this environment? Explain.
 - b. What about a reflex agent with state? Design such an agent.
 - c. How do your answers to a and b change if the agent’s percepts give it the clean/dirty status of every square in the environment?
-
- a. No. Consider the reflex actions for $[A, Clean]$ and $[B, Clean]$. If either of these is *NoOp*, then the agent will fail in the case where that is the initial percept but the other square is dirty; hence, neither can be *NoOp* and therefore the simple reflex agent is doomed to keep moving, which loses points unnecessarily. In general, the problem with reflex agents is that they have to do the same thing in situations that *look the same*, even when the situations are *actually quite different*. In the vacuum world this is a big liability, because every interior square (except home) looks either like a square with dirt or a square without dirt.
 - b. Yes, a reflex agent with state can be perfectly rational. It simply needs to remember whether it has visited the other square. If so, it can do nothing after cleaning the current

square (if needed). If not, it then goes to the other square and cleans it if needed.

- c. In this case, a simple reflex agent can be perfectly rational. The agent can consist of a table with eight entries, indexed by percept, that specifies an action to take for each possible state. After the agent acts, the world is updated and the next percept will tell the agent what to do next. For larger environments, constructing a table is infeasible. Instead, the agent could run one of the optimal search algorithms in Chapter 3 and execute the first step of the solution sequence. Again, no internal state is *required*, but it would help to be able to store the solution sequence instead of recomputing it for each new percept.

Exercise 2.4.#VACU

Consider a modified version of the vacuum environment in Exercise VACUUM-START-EXERCISE, in which the geography of the environment—its extent, boundaries, and obstacles—is unknown, as is the initial dirt configuration. (The agent can go *Up* and *Down* as well as *Left* and *Right*.)

- a. Can a simple reflex agent be perfectly rational for this environment? Explain.
- b. Can a simple reflex agent with a *randomized* agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.
- c. Can you design an environment in which your randomized agent will perform poorly? Show your results.
- d. Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

- a. Because the agent does not know the geography and perceives only location and local dirt, and cannot remember what just happened, it will get stuck forever against a wall when it tries to move in a direction that is blocked—that is, unless it randomizes.
- b. One possible design cleans up dirt and otherwise moves in a randomly chosen direction. This is fairly close to what the early-model Roomba™ vacuum cleaner does, although the Roomba has a bump sensor and randomizes only when it hits an obstacle. It works reasonably well but it can take a long time to cover all the squares at least once. Many modern robot cleaners build a map and derive an efficient cleaning pattern.
- c. An example is shown in Figure S2.1. Students may also wish to measure clean-up time for linear or square environments of different sizes, and compare those to the efficient online search algorithms described in Chapter 4. It's worth noting that in the infinite-time limit on a bidirectional graph, an agent that moves randomly will spend time in each square proportional to its degree in the graph (the number of neighbors). Still, it can take a long time to get from one part of the graph to another. If the agent is in a long corridor, its expected travel after n time steps is $O(\sqrt{n})$ squares.
- d. A reflex agent with state can build a map (see Chapter 4 for details). An online depth-first exploration will reach every state in time linear in the size of the environment; therefore, the agent can do much better than the simple reflex agent.

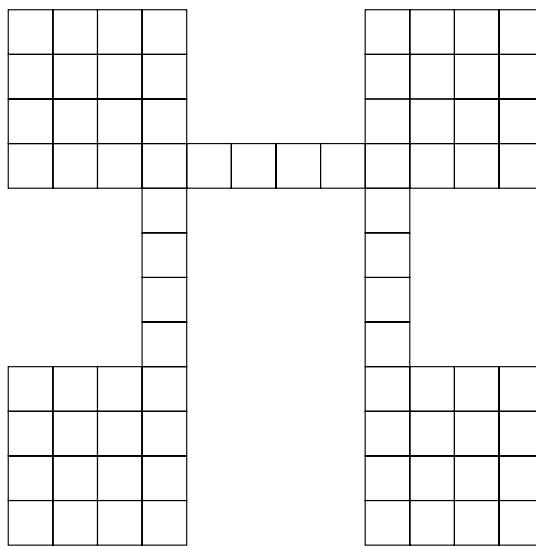


Figure S2.1 An environment in which random motion will take a long time to cover all the squares.

The question of rational behavior in unknown environments is a complex one but it is worth encouraging students to think about it. We need to have some notion of the prior probability distribution over the class of environments; call this the initial **belief state**. Any action yields a new percept that can be used to update this distribution, moving the agent to a new belief state. Once the environment is completely explored, the belief state collapses to a single possible environment. Therefore, the problem of optimal exploration can be viewed as a search for an optimal strategy in the space of possible belief states. This is a well-defined, if horrendously intractable, problem. Chapter 17 discusses some cases (called **bandit problems**) where optimal exploration is possible. Another concrete example of exploration is the Minesweeper computer game (see Exercise 7.22). For very small Minesweeper environments, optimal exploration is feasible although the belief state update is nontrivial to explain.

Exercise 2.4.#VACB

Repeat Exercise VACUUM-UNKNOWN-GEOG-EXERCISE for the case in which the location sensor is replaced with a “bump” sensor that detects the agent’s attempts to move into an obstacle or to cross the boundaries of the environment. Suppose the bump sensor stops working; how should the agent behave?

The problem appears at first to be very similar; the main difference is that instead of using the location percept to build the map, the agent has to “invent” its own locations (which, after all, are just nodes in a data structure representing the state space graph). When a bump is detected, the agent assumes it remains in the same location and can add a wall to its map. For grid environments, the agent can keep track of its (x, y) location and so can tell when it has

returned to an old state. In the general case, however, there is no simple way to tell if a state is new or old.

Exercise 2.4.#VFIN

The vacuum environments in the preceding exercises have all been deterministic. Discuss possible agent programs for each of the following stochastic versions:

- Murphy's law: twenty-five percent of the time, the *Suck* action fails to clean the floor if it is dirty and deposits dirt onto the floor if the floor is clean. How is your agent program affected if the dirt sensor gives the wrong answer 10% of the time?
- Small children: At each time step, each clean square has a 10% chance of becoming dirty. Can you come up with a rational agent design for this case?

- For a reflex agent, this presents no *additional* challenge, because the agent will continue to *Suck* as long as the current location remains dirty. For an agent that constructs a sequential plan, every *Suck* action would need to be replaced by “*Suck until clean*.” If the dirt sensor can be wrong on each step, then the agent might want to wait for a few steps to get a more reliable measurement before deciding whether to *Suck* or move on to a new square. Obviously, there is a trade-off because waiting too long means that dirt remains on the floor (incurring a penalty), but acting immediately risks either dirtying a clean square or ignoring a dirty square (if the sensor is wrong). A rational agent must also continue touring and checking the squares in case it missed one on a previous tour (because of bad sensor readings). It is not immediately obvious how the waiting time at each square should change with each new tour. These issues can be clarified by experimentation, which may suggest a general trend that can be verified mathematically. This problem is a partially observable Markov decision process—see Chapter 17. Such problems are hard in general, but some special cases may yield to careful analysis.
- In this case, the agent must keep touring the squares indefinitely. The probability that a square is dirty increases monotonically with the time since it was last cleaned, so the rational strategy is, roughly speaking, to repeatedly execute the shortest possible tour of all squares. (We say “roughly speaking” because there are complications caused by the fact that the shortest tour may visit some squares twice, depending on the geography.) This problem is also a partially observable Markov decision process.

Exercise 2.4.#DFRA

Define in your own words the following terms: rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.

The following are just some of the many possible definitions that can be written:

- Rationality*: a property of agents that choose actions that maximize their expected utility, given the percepts to date.

- *Autonomy*: a property of agents whose behavior is determined by their own experience rather than solely by their initial programming.
- *Reflex agent*: an agent whose action depends only on the current percept.
- *Model-based agent*: an agent whose action is derived directly from an internal model of the current world state that is updated over time.
- *Goal-based agent*: an agent that selects actions that it believes will achieve explicitly represented goals.
- *Utility-based agent*: an agent that selects actions that it believes will maximize the expected utility of the outcome state.
- *Learning agent*: an agent whose behavior improves over time based on its experience.

Exercise 2.4.#GBUT

Write pseudocode agent programs for the goal-based and utility-based agents.

The design of goal- and utility-based agents depends on the structure of the task environment. The simplest such agents, for example those in chapters 3 and 10, compute the agent's entire future sequence of actions in advance before acting at all. This strategy works for static and deterministic environments which are either fully-known or unobservable.

For fully-observable and fully-known static environments a policy can be computed in advance which gives the action to be taken in any given state.

For partially-observable environments the agent can compute a conditional plan, which specifies the sequence of actions to take as a function of the agent's perception. In the extreme, a conditional plan gives the agent's response to every contingency, and so it is a representation of the entire agent function.

In all cases it may be either intractable or too expensive to compute everything out in advance. Instead of a conditional plan, it may be better to compute a single sequence of actions which is likely to reach the goal, then monitor the environment to check whether the plan is succeeding, repairing or replanning if it is not. It may be even better to compute only the start of this plan before taking the first action, continuing to plan at later time steps.

Pseudocode for simple goal-based agent is given in Figure S2.2. GOAL-ACHIEVED tests to see whether the current state satisfies the goal or not, doing nothing if it does. PLAN computes a sequence of actions to take to achieve the goal. This might return only a prefix of the full plan, the rest will be computed after the prefix is executed. This agent will act to maintain the goal: if at any point the goal is not satisfied it will (eventually) replan to achieve the goal again.

At this level of abstraction the utility-based agent is not much different than the goal-based agent, except that action may be continuously required (there is not necessarily a point where the utility function is "satisfied"). Pseudocode is given in Figure S2.3.

Exercise 2.4.#FURN

Consider a simple thermostat that turns on a furnace when the temperature is at least 3 degrees below the setting, and turns off a furnace when the temperature is at least 3 degrees

```
function GOAL-BASED-AGENT( $\check{p}$ ercept) returns an action
  persistent:  $\check{s}$ tate, the agent's current conception of the world state
     $\check{m}$ odel, a description of how the next state depends on current state and action
     $\check{g}$ oal, a description of the desired goal state
     $\check{p}$ lan, a sequence of actions to take, initially empty
     $\check{a}$ ction, the most recent action, initially none

   $\check{s}$ tate  $\leftarrow$  UPDATE-STATE( $\check{s}$ tate,  $\check{a}$ ction,  $\check{p}$ ercept,  $\check{m}$ odel)
  if GOAL-ACHIEVED( $\check{s}$ tate,  $\check{g}$ oal) then return a null action
  if  $\check{p}$ lan is empty then
     $\check{p}$ lan  $\leftarrow$  PLAN( $\check{s}$ tate,  $\check{g}$ oal,  $\check{m}$ odel)
     $\check{a}$ ction  $\leftarrow$  FIRST( $\check{p}$ lan)
     $\check{p}$ lan  $\leftarrow$  REST( $\check{p}$ lan)
  return  $\check{a}$ ction
```

Figure S2.2 A goal-based agent.

```
function UTILITY-BASED-AGENT( $\check{p}$ ercept) returns an action
  persistent:  $\check{s}$ tate, the agent's current conception of the world state
     $\check{m}$ odel, a description of how the next state depends on current state and action
     $\check{u}$ tility-function, a description of the agent's utility function
     $\check{p}$ lan, a sequence of actions to take, initially empty
     $\check{a}$ ction, the most recent action, initially none

   $\check{s}$ tate  $\leftarrow$  UPDATE-STATE( $\check{s}$ tate,  $\check{a}$ ction,  $\check{p}$ ercept,  $\check{m}$ odel)
  if  $\check{p}$ lan is empty then
     $\check{p}$ lan  $\leftarrow$  PLAN( $\check{s}$ tate,  $\check{u}$ tility-function,  $\check{m}$ odel)
     $\check{a}$ ction  $\leftarrow$  FIRST( $\check{p}$ lan)
     $\check{p}$ lan  $\leftarrow$  REST( $\check{p}$ lan)
  return  $\check{a}$ ction
```

Figure S2.3 A utility-based agent.

above the setting. Is a thermostat an instance of a simple reflex agent, a model-based reflex agent, or a goal-based agent?

The thermostat is best understood as a simple reflex agent. Although the temperature setting might be viewed as a goal, the agent has no notion of how its actions will lead to the goal; so it's better to view the temperature setting as part of the agent's percepts. A more complex control system might well have an internal model of the house, the behavior of its occupants, the capabilities of the heating system, and so on, and would be viewed as a goal-based system in much the same way as a self-driving car that tries to reach a specified

destination quickly and cheaply.

Exercise 2.4.#AGPR

Here is pseudocode for three agent programs A, B, C:

```
function A(percept)
return  $f_A()$ 
```

```
function B(percept)
return  $f_B(\text{percept})$ 
```

```
function C(percept)
persistent: percepts, initially []
percepts  $\leftarrow \text{push}(\text{percept}, \text{percepts})$ 
return  $f_C(\text{percepts})$ 
```

In each of these agents, the function f is some arbitrary, possibly randomized, function of its inputs with no internal state of its own; the agent program runs on a computer with unbounded memory but finite clock speed. We'll assume also that the environment and its performance measure are computable.

- a. Suppose the environment is fully observable, deterministic, discrete, single-agent, and static. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?
 - b. Suppose the environment is *partially* observable, deterministic, discrete, single-agent, and static. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?
 - c. Suppose the environment is partially observable, *stochastic*, discrete, single-agent, and *dynamic*. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?
-
- a. B, C. For a fully observable environment, only the current percept is required for an optimal decision. Because the environment is static, computation is not an issue. Note that Agent A cannot make optimal decisions because it always makes the *same* decision (or samples a decision from the same probability distribution), having no internal state.
 - b. C. Agent B, the reflex agent, cannot always function optimally in a partially observable environment because it ignores previous percepts and therefore fails to take into account relevant information for the current decision.
 - c. None of the agents can be optimal for an arbitrary dynamic environment, because we can make the environment complex enough to render optimal decisions infeasible for any finite-speed machine.

EXERCISES 3

SOLVING PROBLEMS BY SEARCHING

3.1 Problem-Solving Agents

Exercise 3.1.#FORM

Explain why problem formulation must follow goal formulation.

In goal formulation, we decide which aspects of the world we are interested in, and which can be ignored or abstracted away. Then in problem formulation we decide how to manipulate the important aspects (and ignore the others). If we did problem formulation first we would not know what to include and what to leave out. That said, it can happen that there is a cycle of iterations between goal formulation, problem formulation, and problem solving until one arrives at a sufficiently useful and efficient solution.

3.2 Example Problems

Exercise 3.2.#PRFO

Give a complete problem formulation for each of the following problems. Choose a formulation that is precise enough to be implemented.

- a. There are six glass boxes in a row, each with a lock. Each of the first five boxes holds a key unlocking the next box in line; the last box holds a banana. You have the key to the first box, and you want the banana.
- b. You start with the sequence ABABAECCEC, or in general any sequence made from A, B, C, and E. You can transform this sequence using the following equalities: AC = E, AB = BC, BB = E, and Ex = x for any x. For example, ABBC can be transformed into AEC (using BB = E), and then AC (using Ex = x), and then E (using AC = E). Your goal is to produce the sequence E.
- c. There is an $n \times n$ grid of squares, each square initially being either unpainted floor or a bottomless pit. You start standing on an unpainted floor square, and can either paint the square under you or move onto an adjacent unpainted floor square. You want the whole floor painted.
- d. A container ship is in port, loaded high with containers. There 13 rows of containers, each 13 containers wide and 5 containers tall. You control a crane that can move to any

location above the ship, pick up the container under it, and move it onto the dock. You want the ship unloaded.

- a. Initial state: as described in the question.

Goal test: you have banana.

Successor function: open any box you have the key for, get the contents of any open box.

Cost function: number of actions.

- b. Initial state: ABABAECCEC.

Goal test: is the state E

Successor function: apply an equality substituting one subsequence for the other.

Cost function: number of transformations.

- c. Initial state: all floor squares unpainted, you start standing on one square unpainted floor square.

Goal test: all floor squares painted.

Successor function: paint current tile, move to adjacent unpainted floor tile.

Cost function: number of moves.

- d. Initial state: all containers stacked on the ship.

Goal test: all containers unloaded.

Successor function: move crane to a certain location, pick up a container, put down container.

Cost function: time taken to unload ship.

Exercise 3.2.#RMAZ

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

- a. Formulate this problem. How large is the state space?
- b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?
- c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?
- d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

Exercises 3 Solving Problems by Searching

- a. We'll define the coordinate system so that the center of the maze is at $(0, 0)$, and the maze itself is a square from $(-1, -1)$ to $(1, 1)$.

Initial state: robot at coordinate $(0, 0)$, facing North.

Goal test: either $|x| > 1$ or $|y| > 1$ where (x, y) is the current location.

Successor function: move forwards any distance d ; change direction robot it facing.

Cost function: total distance moved.

The state space is infinitely large, since the robot's position is continuous.

- b. The state will record the intersection the robot is currently at, along with the direction it's facing. At the end of each corridor leaving the maze we will have an exit node. We'll assume some node corresponds to the center of the maze.

Initial state: at the center of the maze facing North.

Goal test: at an exit node.

Successor function: move to the next intersection in front of us, if there is one; turn to face a new direction.

Cost function: total distance moved.

There are $4n$ states, where n is the number of intersections.

- c. Initial state: at the center of the maze.

Goal test: at an exit node.

Successor function: move to next intersection to the North, South, East, or West.

Cost function: total distance moved.

We no longer need to keep track of the robot's orientation since it is irrelevant to predicting the outcome of our actions, and not part of the goal test. The motor system that executes this plan will need to keep track of the robot's current orientation, to know when to rotate the robot.

- d. State abstractions:

- (i) Ignoring the height of the robot off the ground, whether it is tilted off the vertical.
- (ii) The robot can face in only four directions.
- (iii) Other parts of the world ignored: possibility of other robots in the maze, the weather in the Caribbean.

Action abstractions:

- (i) We assumed all positions we safely accessible: the robot couldn't get stuck or damaged.
- (ii) The robot can move as far as it wants, without having to recharge its batteries.
- (iii) Simplified movement system: moving forwards a certain distance, rather than controlled each individual motor and watching the sensors to detect collisions.

3.3 Search Algorithms

3.4 Uninformed Search Strategies

3.5 Informed (Heuristic) Search Strategies

3.6 Heuristic Functions

Exercise 3.6.#GRRB

You have a 9×9 grid of squares, each of which can be colored red or blue. The grid is initially colored all blue, but you can change the color of any square any number of times. Imagining the grid divided into nine 3×3 sub-squares, you want each sub-square to be all one color but neighboring sub-squares to be different colors.

- a. Formulate this problem in the straightforward way. Compute the size of the state space.
- b. You need color a square only once. Reformulate, and compute the size of the state space. Would breadth-first graph search perform faster on this problem than on the one in (a)? How about iterative deepening tree search?
- c. Given the goal, we need consider only colorings where each sub-square is uniformly colored. Reformulate the problem and compute the size of the state space.
- d. How many solutions does this problem have?
- e. Parts (b) and (c) successively abstracted the original problem (a). Can you give a translation from solutions in problem (c) into solutions in problem (b), and from solutions in problem (b) into solutions for problem (a)?

- a. Initial state: all squares colored blue.

Goal test: each sub-square colored the same, and adjacent subsquares colored different.

Successor function: color a square red or blue.

Cost function: number of times squares are colored.

There are 2^{81} states.

- b. To avoid recoloring squares we could either record squares that have been colored, and not allow them to be colored again, or color all the squares in a fixed order. We describe the latter:

Initial state: all squares colored blue, the top-left square is next to be colored.

Goal test: each sub-square colored the same, and adjacent subsquares colored different.

Successor function: if we haven't colored all squares yet, color the current square red or blue (this automatically updates the next square to be colored).

Cost function: number of squares colored.

There are $82 \cdot 2^{81}$ states, since we need to record whether there are squares left to be colored, and if so which square is next.

- c. Initial state: all sub-squares colored blue, the top-left sub-square is next to be colored.
- Goal test: adjacent sub-squares colored differently.
- Successor function: if we haven't colored all sub-squares yet, color the current sub-square red or blue.
- Cost function: number of sub-squares colored.

The state space has $10 \cdot 2^9$ nodes.

- d. This problem has 2 solutions: as soon as you choose the color of any square, the color of any other square is determined.
- e. Given a solution for problem (c) first determine final coloring of squares, after all actions are taken. Then color each of the squares the required color in order.

Given a solution for problem (b) first determine final coloring of squares. We know that each sub-square will be colored a single color, so we can color them in order to reach the same goal state.

Exercise 3.6.#ROMF

Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.1. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

- a. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)
- b. Let $D(i, j)$ be the straight-line distance between cities i and j . Which of the following heuristic functions are admissible? (i) $D(i, j)$; (ii) $2 \cdot D(i, j)$; (iii) $D(i, j)/2$.
- c. Are there completely connected maps for which no solution exists?
- d. Are there maps in which all solutions require one friend to visit the same city twice?

- a. State space: States are all possible city pairs (i, j) . The map is *not* the state space.
Successor function: The successors of (i, j) are all pairs (x, y) such that $Adjacent(x, i)$ and $Adjacent(y, j)$.
Goal: Be at (i, i) for some i .
Step cost function: The cost to go from (i, j) to (x, y) is $\max(d(i, x), d(j, y))$.
- b. In the best case, the friends head straight for each other in steps of equal size, reducing their separation by twice the time cost on each step. Hence (iii) is admissible.
- c. Yes: e.g., a map with two nodes connected by one link. The two friends will swap places forever. The same will happen on any chain if they start an odd number of steps apart. (One can see this best on the graph that represents the state space, which has two disjoint sets of nodes.) The same even holds for a grid of any size or shape, because every move changes the Manhattan distance between the two friends by 0 or 2.
- d. Yes: take any of the unsolvable maps from part (c) and add a self-loop to any one of the nodes. If the friends start an odd number of steps apart, a move in which one of the friends takes the self-loop changes the distance by 1, rendering the problem solvable. If the self-loop is not taken, the argument from (c) applies and no solution is possible.

Exercise 3.6.#PART

Show that the 8-puzzle states are divided into two disjoint sets, such that any state is reachable from any other state in the same set, while no state is reachable from any state in the other set. (*Hint:* See Berlekamp *et al.*, (1982).) Devise a procedure to decide which set a given state is in, and explain why this is useful for generating random states.

From <http://www.cut-the-knot.com/pythagoras/fifteen.shtml>, this proof applies to the fifteen puzzle, but the same argument works for the eight puzzle:

Definition: The goal state has the numbers in a certain order, which we will measure as starting at the upper left corner, then proceeding left to right, and when we reach the end of a row, going down to the leftmost square in the row below. For any other configuration besides the goal, whenever a tile with a greater number on it precedes a tile with a smaller number, the two tiles are said to be **inverted**.

Proposition: For a given puzzle configuration, let N denote the sum of the total number of inversions and the row number of the empty square. Then $(N \bmod 2)$ is invariant under any legal move. In other words, after a legal move an odd N remains odd whereas an even N remains even. Therefore the goal state in Figure 3.3, with no inversions and empty square in the first row, has $N = 1$, and can only be reached from starting states with odd N , not from starting states with even N .

Proof: First of all, sliding a tile horizontally changes neither the total number of inversions nor the row number of the empty square. Therefore let us consider sliding a tile vertically.

Let's assume, for example, that the tile A is located directly over the empty square. Sliding it down changes the parity of the row number of the empty square. Now consider the total number of inversions. The move only affects relative positions of tiles A , B , C , and D . If none of the B , C , D caused an inversion relative to A (i.e., all three are larger than A) then after sliding one gets three (an odd number) of additional inversions. If one of the three is smaller than A , then before the move B , C , and D contributed a single inversion (relative to A) whereas after the move they'll be contributing two inversions - a change of 1, also an odd number. Two additional cases obviously lead to the same result. Thus the change in the sum N is always even. This is precisely what we have set out to show.

So before we solve a puzzle, we should compute the N value of the start and goal state and make sure they have the same parity, otherwise no solution is possible.

Exercise 3.6.#NQUE

Consider the n -queens problem using an efficient incremental formulation where a state is represented as a n -element vector of row numbers for queens (one for each column). Explain why the state space has at least $\sqrt[3]{n!}$ states and estimate the largest n for which exhaustive exploration is feasible. (*Hint:* Derive a lower bound on the branching factor by considering the maximum number of squares that a queen can attack in any column.)

The formulation puts one queen per column, with a new queen placed only in a square

Exercises 3 Solving Problems by Searching

that is not attacked by any other queen. To simplify matters, we'll first consider the n -rooks problem. The first rook can be placed in any square in column 1 (n choices), the second in any square in column 2 except the same row that as the rook in column 1 ($n - 1$ choices), and so on. This gives $n!$ elements of the search space.

For n queens, notice that a queen attacks at most three squares in any given column, so in column 2 there are at least $(n - 3)$ choices, in column at least $(n - 6)$ choices, and so on. Thus the state space size $S \geq n \cdot (n - 3) \cdot (n - 6) \dots$. Hence we have

$$\begin{aligned} S^3 &\geq n \cdot n \cdot n \cdot (n - 3) \cdot (n - 3) \cdot (n - 6) \cdot (n - 6) \cdot (n - 6) \dots \\ &\geq n \cdot (n - 1) \cdot (n - 2) \cdot (n - 3) \cdot (n - 4) \cdot (n - 5) \cdot (n - 6) \cdot (n - 7) \cdot (n - 8) \dots \\ &= n! \end{aligned}$$

or $S \geq \sqrt[3]{n!}$.

Exercise 3.6.#COMP

Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.

- a. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.
- b. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. The monkey would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates.
- c. You have a program that outputs the message “illegal input record” when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.
- d. You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

- a. Initial state: No regions colored.

Goal test: All regions colored, and no two adjacent regions have the same color.

Successor function: Assign a color to a region.

Cost function: Number of assignments.

- b. Initial state: As described in the text.

Goal test: Monkey has bananas.

Successor function: Hop on crate; Hop off crate; Push crate from one spot to another; Walk from one spot to another; grab bananas (if standing on crate).

Cost function: Number of actions.

- c. Initial state: considering all input records.

Goal test: considering a single record, and it gives “illegal input” message.

Successor function: run again on the first half of the records; run again on the second half of the records.

Cost function: Number of runs.

Note: This is a **contingency problem**; you need to see whether a run gives an error message or not to decide what to do next.

- d. Initial state: jugs have values $[0, 0, 0]$.

Successor function: given values $[x, y, z]$, generate $[12, y, z]$, $[x, 8, z]$, $[x, y, 3]$ (by filling); $[0, y, z]$, $[x, 0, z]$, $[x, y, 0]$ (by emptying); or for any two jugs with current values x and y , pour y into x ; this changes the jug with x to the minimum of $x + y$ and the capacity of the jug, and decrements the jug with y by the amount gained by the first jug.

Cost function: Number of actions.

Exercise 3.6.#PPPO

Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in Figure ???. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.

- a. Suppose the state space consists of all positions (x, y) in the plane. How many states are there? How many paths are there to the goal?
- b. Explain briefly why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?
- c. Define the necessary functions to implement the search problem, including an ACTIONS function that takes a vertex as input and returns a set of vectors, each of which maps the current vertex to one of the vertices that can be reached in a straight line. (Do not forget the neighbors on the same polygon.) Use the straight-line distance for the heuristic function.
- d. Apply one or more of the algorithms in this chapter to solve a range of problems in the domain, and comment on their performance.

- a. If we consider all (x, y) points, then there are an infinite number of states, and of paths.
- b. (For this problem, we consider the start and goal points to be vertices.) The shortest distance between two points is a straight line, and if it is not possible to travel in a straight line because some obstacle is in the way, then the next shortest distance is a sequence of line segments, end-to-end, that deviate from the straight line by as little as possible. So the first segment of this sequence must go from the start point to a tangent point on an obstacle – any path that gave the obstacle a wider girth would be longer. Because the obstacles are polygonal, the tangent points must be at vertices of the obstacles, and hence the entire path must go from vertex to vertex. So now the state space is the set of vertices, of which there are 35 in Figure ???.
- c. Code not shown.
- d. Implementations and analysis not shown.

Exercise 3.6.#NNEG

We said in the chapter that we would not consider problems with negative path costs. In this exercise, we explore this decision in more depth.

- a. Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.
 - b. Does it help if we insist that step costs must be greater than or equal to some negative constant c ? Consider both trees and graphs.
 - c. Suppose that a set of actions forms a loop in the state space such that executing the set in some order results in no net change to the state. If all of these actions have negative cost, what does this imply about the optimal behavior for an agent in such an environment?
 - d. One can easily imagine actions with high negative cost, even in domains such as route finding. For example, some stretches of road might have such beautiful scenery as to far outweigh the normal costs in terms of time and fuel. Explain, in precise terms, within the context of state-space search, why humans do not drive around scenic loops indefinitely, and explain how to define the state space and actions for route finding so that artificial agents can also avoid looping.
 - e. Can you think of a real domain in which step costs are such as to cause looping?
-
- a. Any path, no matter how bad it appears, might lead to an arbitrarily large reward (negative cost). Therefore, one would need to exhaust all possible paths to be sure of finding the best one.
 - b. Suppose the greatest possible reward is c . Then if we also know the maximum depth of the state space (e.g. when the state space is a tree), then any path with d levels remaining can be improved by at most cd , so any paths worse than cd less than the best path can be pruned. For state spaces with loops, this guarantee doesn't help, because it is possible to go around a loop any number of times, picking up c reward each time.
 - c. The agent should plan to go around this loop forever (unless it can find another loop with even better reward).
 - d. The value of a scenic loop is lessened each time one revisits it; a novel scenic sight is a great reward, but seeing the same one for the tenth time in an hour is tedious, not rewarding. To accommodate this, we would have to expand the state space to include a memory—a state is now represented not just by the current location, but by a current location and a bag of already-visited locations. The reward for visiting a new location is now a (diminishing) function of the number of times it has been seen before.
 - e. Real domains with looping behavior include eating junk food and going to class.

Exercise 3.6.#MICA

The **missionaries and cannibals** problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of mis-

sionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

- a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.
- b. Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?
- c. Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

- a. Here is one possible representation: A state is a six-tuple of integers listing the number of missionaries, cannibals, and boats on the first side, and then the second side of the river. The goal is a state with 3 missionaries and 3 cannibals on the second side. The cost function is one per action, and the successors of a state are all the states that move 1 or 2 people and 1 boat from one side to another.
- b. The search space is small, so any optimal algorithm works. For an example, see the file `aima-lisp/search/domains/cannibals.lisp`. It suffices to eliminate moves that circle back to the state just visited. From all but the first and last states, there is only one other choice.
- c. It is not obvious that almost all moves are either illegal or revert to the previous state. There is a feeling of a large branching factor, and no clear way to proceed.

Exercise 3.6.#DEFS

Define in your own words the following terms: state, state space, search tree, search node, goal, action, transition model, and branching factor.

A **state** is a situation that an agent can find itself in. We distinguish two types of states: world states (the actual concrete situations in the real world) and representational states (the abstract descriptions of the real world that are used by the agent in deliberating about what to do).

A **state space** is a graph whose nodes are the set of all states, and whose links are actions that transform one state into another.

A **search tree** is a tree (a graph with no undirected loops) in which the root node is the start state and the set of children for each node consists of the states reachable by taking any action.

A **search node** is a node in the search tree.

A **goal** is a state that the agent is trying to reach.

An **action** is something that the agent can choose to do.

A **successor function** described the agent's options: given a state, it returns a set of (action, state) pairs, where each state is the state reachable by taking the action.

The **branching factor** in a search tree is the number of actions available to the agent.

Exercise 3.6.#STAT

What's the difference between a world state, a state description, and a search node? Why is this distinction useful?

A world state is how reality is or could be. In one world state we're in Arad, in another we're in Bucharest. The world state also includes which street we're on, what's currently on the radio, and the price of tea in China. A state description is an agent's internal description of a world state. Examples are *In(Arad)* and *In(Bucharest)*. These descriptions are necessarily approximate, recording only some aspect of the state.

We need to distinguish between world states and state descriptions because state description are lossy abstractions of the world state, because the agent could be mistaken about how the world is, because the agent might want to imagine things that aren't true but it could make true, and because the agent cares about the world not its internal representation of it.

Search nodes are generated during search, representing a state the search process knows how to reach. They contain additional information aside from the state description, such as the sequence of actions used to reach this state. This distinction is useful because we may generate different search nodes which have the same state, and because search nodes contain more information than a state representation.

Exercise 3.6.#AABS

An action such as going to Sibiu really consists of a long sequence of finer-grained actions: turn on the car, release the brake, accelerate forward, etc. Having composite actions of this kind reduces the number of steps in a solution sequence, thereby reducing the search time. Suppose we take this to the logical extreme, by making super-composite actions out of every possible sequence of *Go* actions. Then every problem instance is solved by a single super-composite action, such as *Go(Sibiu) Go(Rimnicu Vilcea) Go(Pitesti) Go(Bucharest)*. Explain how search would work in this formulation. Is this a practical approach for speeding up problem solving?

The state space is a tree of depth one, with all states successors of the initial state. There is no distinction between depth-first search and breadth-first search on such a tree. If the sequence length is unbounded the root node will have infinitely many successors, so only algorithms which test for goal nodes as we generate successors can work.

What happens next depends on how the composite actions are sorted. If there is no particular ordering, then a random but systematic search of potential solutions occurs. If they are sorted by dictionary order, then this implements depth-first search. If they are sorted by length first, then dictionary ordering, this implements breadth-first search.

A significant disadvantage of collapsing the search space like this is if we discover that a plan starting with the action "unplug your battery" can't be a solution, there is no easy way to ignore all other composite actions that start with this action. This is a problem in particular for informed search algorithms.

Discarding sequence structure is not a particularly practical approach to search.

Exercise 3.6.#FINS

Does a finite state space always lead to a finite search tree? How about a finite state space that is a tree? Can you be more precise about what types of state spaces always lead to finite search trees? (Adapted from Bender (1996).)

No, a finite state space does not always lead to a finite search tree. Consider a state space with two states, both of which have actions that lead to the other. This yields an infinite search tree, because we can go back and forth any number of times. However, if the state space is a finite tree, or in general, a finite DAG (directed acyclic graph), then there can be no loops, and the search tree is finite.

Exercise 3.6.#GRAS

Prove that GRAPH-SEARCH satisfies the graph separation property illustrated in Figure 3.6. (*Hint:* Begin by showing that the property holds at the start, then show that if it holds before an iteration of the algorithm, it holds afterwards.) Describe a search algorithm that violates the property.

The graph separation property states that “every path from the initial state to an unexplored state has to pass through a state in the frontier.”

At the start of the search, the frontier holds the initial state; hence, trivially, every path from the initial state to an unexplored state includes a node in the frontier (the initial state itself).

Now, we assume that the property holds at the beginning of an arbitrary iteration of the GRAPH-SEARCH algorithm in Figure ???. We assume that the iteration completes, i.e., the frontier is not empty and the selected leaf node n is not a goal state. At the end of the iteration, n has been removed from the frontier and its successors (if not already explored or in the frontier) placed in the frontier. Consider any path from the initial state to an unexplored state; by the induction hypothesis such a path (at the beginning of the iteration) includes at least one frontier node; except when n is the only such node, the separation property automatically holds. Hence, we focus on paths passing through n (and no other frontier node). By definition, the next node n' along the path from n must be a successor of n that (by the preceding sentence) is already not in the frontier. Furthermore, n' cannot be in the explored set, since by assumption there is a path from n' to an unexplored node not passing through the frontier, which would violate the separation property as every explored node is connected to the initial state by explored nodes (see lemma below for proof this is always possible). Hence, n' is not in the explored set, hence it will be added to the frontier; then the path will include a frontier node and the separation property is restored.

The property is violated by algorithms that move nodes from the frontier into the explored set before all of their successors have been generated, as well as by those that fail to add some of the successors to the frontier. Note that it is not necessary to generate *all* successors of a node at once before expanding another node, as long as partially expanded nodes remain in the frontier.

Lemma: Every explored node is connected to the initial state by a path of explored nodes.

Proof: This is true initially, since the initial state is connected to itself. Since we never remove nodes from the explored region, we only need to check new nodes we add to the explored list on an expansion. Let n be such a new explored node. This is previously on the frontier, so it is a neighbor of a node n' previously explored (i.e., its parent). n' is, by hypothesis is connected to the initial state by a path of explored nodes. This path with n appended is a path of explored nodes connecting n' to the initial state.

Exercise 3.6.#TFSA

Which of the following are true and which are false? Explain your answers.

- Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.
 - $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
 - A* is of no use in robotics because percepts, states, and actions are continuous.
 - Breadth-first search is complete even if zero step costs are allowed.
 - Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.
-
- False:* a lucky DFS might expand exactly d nodes to reach the goal. A* largely dominates any graph-search algorithm that is *guaranteed to find optimal solutions*.
 - True:* $h(n) = 0$ is always an admissible heuristic, since costs are nonnegative.
 - True:* A* search is often used in robotics; the space can be discretized or skeletonized.
 - True:* depth of the solution matters for breadth-first search, not cost.
 - False:* a rook can move across the board in move one, although the Manhattan distance from start to finish is 8.

Exercise 3.6.#KTWO

Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k + 1$.

- Draw the portion of the state space for states 1 to 15.
- Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.
- How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?
- Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?
- Call the action going from k to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

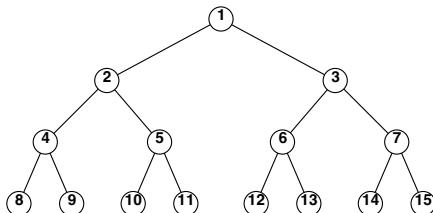


Figure S3.1 The state space for the problem defined in Ex. 3.15.

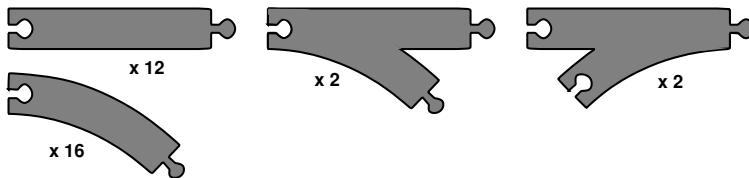


Figure S3.2 Basic wooden railway pieces.

- See Figure S3.1.
- Breadth-first: 1 2 3 4 5 6 7 8 9 10 11
Depth-limited: 1 2 4 8 9 5 10 11
Iterative deepening: 1; 1 2 3; 1 2 4 5 3 6 7; 1 2 4 8 9 5 10 11
- Bidirectional search is very useful, because the only successor of n in the reverse direction is $\lfloor(n/2)\rfloor$. This helps focus the search. The branching factor is 2 in the forward direction; 1 in the reverse direction.
- Yes; start at the goal, and apply the single reverse successor action until you reach 1.
- The solution can be read off the binary numeral for the goal number. Write the goal number in binary. Since we can only reach positive integers, this binary expansion begins with a 1. From most- to least-significant bit, skipping the initial 1, go Left to the node $2n$ if this bit is 0 and go Right to node $2n + 1$ if it is 1. For example, suppose the goal is 11, which is 1011 in binary. The solution is therefore Left, Right, Right.

Exercise 3.6.#BRIW

A basic wooden railway set contains the pieces shown in Figure S3.2. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.

- Suppose that the pieces fit together *exactly* with no slack. Give a precise formulation of the task as a search problem.
- Identify a suitable uninformed search algorithm for this task and explain your choice.

- c. Explain why removing any one of the “fork” pieces makes the problem unsolvable.
- d. Give an upper bound on the total size of the state space defined by your formulation.
(Hint: think about the maximum branching factor for the construction process and the maximum depth, ignoring the problem of overlapping pieces and loose ends. Begin by pretending that every piece is unique.)

- a. **Initial state:** one arbitrarily selected piece (say a straight piece).

Successor function: for any open peg, add any piece type from remaining types. (You can add to open holes as well, but that isn’t necessary as all complete tracks can be made by adding to pegs.) For a curved piece, add *in either orientation*; for a fork, add *in either orientation* and (if there are two holes) connecting *at either hole*. It’s a good idea to disallow any overlapping configuration, as this terminates hopeless configurations early. (Note: there is no need to consider open holes, because in any solution these will be filled by pieces added to open pegs.)

Goal test: all pieces used in a single connected track, no open pegs or holes, no overlapping tracks.

Step cost: one per piece (actually, doesn’t really matter).

- b. All solutions are at the same depth, so depth-first search would be appropriate. (One could also use depth-limited search with limit $n - 1$, but strictly speaking it’s not necessary to do the work of checking the limit because states at depth $n - 1$ have no successors.) The space is very large, so uniform-cost and breadth-first would fail, and iterative deepening simply does unnecessary extra work. There are many repeated states, so it might be good to use a closed list.
- c. A solution has no open pegs or holes, so every peg is in a hole, so there must be equal numbers of pegs and holes. Removing a fork violates this property. There are two other “proofs” that are acceptable: 1) a similar argument to the effect that there must be an even number of “ends”; 2) each fork creates two tracks, and only a fork can rejoin those tracks into one, so if a fork is missing it won’t work. The argument using pegs and holes is actually more general, because it also applies to the case of a three-way fork that has one hole and three pegs or one peg and three holes. The “ends” argument fails here, as does the fork/rejoin argument (which is a bit handwavy anyway).
- d. The maximum possible number of open pegs is 3 (starts at 1, adding a two-peg fork increases it by one). Pretending each piece is unique, any piece can be added to a peg, giving at most $12 + (2 \cdot 16) + (2 \cdot 2) + (2 \cdot 2 \cdot 2) = 56$ choices per peg. The total depth is 32 (there are 32 pieces), so an upper bound is $168^{32} / (12! \cdot 16! \cdot 2! \cdot 2!)$ where the factorials deal with permutations of identical pieces. One could do a more refined analysis to handle the fact that the branching factor shrinks as we go down the tree, but it is not pretty.

Exercise 3.6.#RESE

Implement two versions of the $\text{RESULT}(s, a)$ function for the 8-puzzle: one that copies and edits the data structure for the parent node s and one that modifies the parent state directly (undoing the modifications as needed). Write versions of iterative deepening depth-first search that use these functions and compare their performance.

For the 8 puzzle, there shouldn't be much difference in performance. Indeed, the file "aima-lisp/search/domains/puzzle8.lisp" shows that you can represent an 8 puzzle state as a single 32-bit integer, so the question of modifying or copying data is moot. But for the $n \times n$ puzzle, as n increases, the advantage of modifying rather than copying grows. The disadvantage of a modifying successor function is that it only works with depth-first search (or with a variant such as iterative deepening).

Exercise 3.6.#ITLE

Iterative lengthening search is an iterative analog of uniform cost search. The idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.

- Show that this algorithm is optimal for general path costs.
- Consider a uniform tree with branching factor b , solution depth d , and unit step costs. How many iterations will iterative lengthening require?
- Now consider step costs drawn from the continuous range $[\epsilon, 1]$, where $0 < \epsilon < 1$. How many iterations are required in the worst case?
- Implement the algorithm and apply it to instances of the 8-puzzle and traveling salesperson problems. Compare the algorithm's performance to that of uniform-cost search, and comment on your results.

a. The algorithm expands nodes in order of increasing path cost; therefore the first goal it encounters will be the goal with the cheapest cost.

b. It will be the same as iterative deepening, d iterations, in which $O(b^d)$ nodes are generated.

c. d/ϵ

d. Implementation not shown.

Exercise 3.6.#ITDW

Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs. $O(n)$).

Consider a domain in which every state has a single successor, and there is a single goal at depth n . Then depth-first search will find the goal in n steps, whereas iterative deepening search will take $1 + 2 + 3 + \dots + n = O(n^2)$ steps.

Exercise 3.6.#WWL

Write a program that will take as input two web page URLs and find a path of links from one to the other. What is an appropriate search strategy? Is bidirectional search a good idea? Could a search engine be used to implement a predecessor function?

As an ordinary person (or agent) browsing the web, we can only generate the successors of a page by visiting it. We can then do breadth-first search, or perhaps best-search search where the heuristic is some function of the number of words in common between the start and goal pages; this may help keep the links on target. Search engines keep the complete graph of the web, and may provide the user access to all (or at least some) of the pages that link to a page; this would allow us to do bidirectional search.

Exercise 3.6.#VACG

Consider the vacuum-world problem defined in Figure 2.2.

- a. Which of the algorithms defined in this chapter would be appropriate for this problem? Should the algorithm use tree search or graph search?
- b. Apply your chosen algorithm to compute an optimal sequence of actions for a 3×3 world whose initial state has dirt in the three top squares and the agent in the center.
- c. Construct a search agent for the vacuum world, and evaluate its performance in a set of 3×3 worlds with probability 0.2 of dirt in each square. Include the search cost as well as path cost in the performance measure, using a reasonable exchange rate.
- d. Compare your best search agent with a simple randomized reflex agent that sucks if there is dirt and otherwise moves randomly.
- e. Consider what would happen if the world were enlarged to $n \times n$. How does the performance of the search agent and of the reflex agent vary with n ?

Code not shown, but a good start is in the code repository. Clearly, graph search must be used—this is a classic grid world with many alternate paths to each state. Students will quickly find that computing the optimal solution sequence is prohibitively expensive for moderately large worlds, because the state space for an $n \times n$ world has $n^2 \cdot 2^n$ states. The completion time of the random agent grows less than exponentially in n , so for any reasonable exchange rate between search cost ad path cost the random agent will eventually win.

Exercise 3.6.#SESC

Prove each of the following statements, or give a counterexample:

- a. Breadth-first search is a special case of uniform-cost search.
- b. Depth-first search is a special case of best-first tree search.
- c. Uniform-cost search is a special case of A* search.

- a. When all step costs are equal, $g(n) \propto \text{depth}(n)$, so uniform-cost search reproduces breadth-first search.
- b. Breadth-first search is best-first search with $f(n) = \text{depth}(n)$; depth-first search is best-first search with $f(n) = -\text{depth}(n)$; uniform-cost search is best-first search with $f(n) = g(n)$.
- c. Uniform-cost search is A* search with $h(n) = 0$.

Exercise 3.6.#RBFS

Compare the performance of A* and RBFS on a set of randomly generated problems in the 8-puzzle (with Manhattan distance) and TSP (with MST—see Exercise 3.MSTR) domains. Discuss your results. What happens to the performance of RBFS when a small random number is added to the heuristic values in the 8-puzzle domain?

The student should find that on the 8-puzzle, RBFS expands more nodes (because it does not detect repeated states) but has lower cost per node because it does not need to maintain a queue. The number of RBFS node re-expansions is not too high because the presence of many tied values means that the best path changes seldom. When the heuristic is slightly perturbed, this advantage disappears and RBFS's performance is much worse.

For TSP, the state space is a tree, so repeated states are not an issue. On the other hand, the heuristic is real-valued and there are essentially no tied values, so RBFS incurs a heavy penalty for frequent re-expansions.

Exercise 3.6.#BULU

Trace the operation of A* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f , g , and h score for each node.

The sequence of queues is as follows:

L[0+244=244]
M[70+241=311], T[111+329=440]
L[140+244=384], D[145+242=387], T[111+329=440]
D[145+242=387], T[111+329=440], M[210+241=451], T[251+329=580]
C[265+160=425], T[111+329=440], M[210+241=451], M[220+241=461], T[251+329=580]
T[111+329=440], M[210+241=451], M[220+241=461], P[403+100=503], T[251+329=580], R[411+193=604],
D[385+242=627]
M[210+241=451], M[220+241=461], L[222+244=466], P[403+100=503], T[251+329=580], A[229+366=595],
R[411+193=604], D[385+242=627]
M[220+241=461], L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], T[251+329=580],
A[229+366=595], R[411+193=604], D[385+242=627]
L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], L[290+244=534], D[295+242=537],
T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]
P[403+100=503], L[280+244=524], D[285+242=527], M[292+241=533], L[290+244=534], D[295+242=537],
T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662]

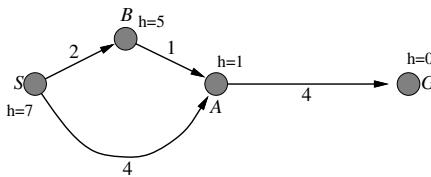


Figure S3.3 A graph with an inconsistent heuristic on which GRAPH-SEARCH fails to return the optimal solution. The successors of S are A with $f = 5$ and B with $f = 7$. A is expanded first, so the path via B will be discarded because A will already be in the closed list.

B[$504+0=504$], L[$280+244=524$], D[$285+242=527$], M[$292+241=533$], L[$290+244=534$], D[$295+242=537$], T[$251+329=580$], A[$229+366=595$], R[$411+193=604$], D[$385+242=627$], T[$333+329=662$], R[$500+193=693$], C[$541+160=701$]

Exercise 3.6.#EVC0

Sometimes there is no good evaluation function for a problem but there is a good comparison method: a way to tell whether one node is better than another without assigning numerical values to either. Show that this is enough to do a best-first search. Is there an analog of A^* for this setting?

If we assume the comparison function is transitive, then we can always sort the nodes using it, and choose the node that is at the top of the sort. Efficient priority queue data structures rely only on comparison operations, so we lose nothing in efficiency—except for the fact that the comparison operation on states may be much more expensive than comparing two numbers, each of which can be computed just once.

A^* relies on the division of the total cost estimate $f(n)$ into the cost-so-far and the cost-to-go. If we have comparison operators for each of these, then we can prefer to expand a node that is better than other nodes on both comparisons. Unfortunately, there will usually be no such node. The tradeoff between $g(n)$ and $h(n)$ cannot be realized without numerical values.

Exercise 3.6.#AFAL

Devise a state space in which A^* using GRAPH-SEARCH returns a suboptimal solution with an $h(n)$ function that is admissible but inconsistent.

See Figure S3.3.

Exercise 3.6.#HEUR

Accurate heuristics don't necessarily reduce search time in the worst case. Given any depth d , define a search problem with a goal node at depth d , and write a heuristic function such that $|h(n) - h^*(n)| \leq O(\log h^*(n))$ but A^* expands all nodes of depth less than d .

Let each state be a binary string, with the initial state the empty string, where the successors of a string x are $\{x0, x1\}$ the string extended by one bit, and where each action has unit cost. Suppose all strings with length greater than d are goals, along with the string of d zeros. Define $h(x) = d - |x|$ if x is of length at most d , and $h(x) = 0$ otherwise. Note that $|h(n) - h^*(n)| \leq 1$. Since A* will expand all nodes with f -value less than d , it will expand all nodes of depth less than d .

Exercise 3.6.#HEPA

The **heuristic path algorithm** (Pohl, 1977) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this complete? For what values is it optimal, assuming that h is admissible? What kind of search does this perform for $w = 0$, $w = 1$, and $w = 2$?

It is complete whenever $0 \leq w < 2$. $w = 0$ gives $f(n) = 2g(n)$. This behaves exactly like uniform-cost search—the factor of two makes no difference in the *ordering* of the nodes. $w = 1$ gives A* search. $w = 2$ gives $f(n) = 2h(n)$, i.e., greedy best-first search. We also have

$$f(n) = (2 - w)[g(n) + \frac{w}{2 - w}h(n)]$$

which behaves exactly like A* search with a heuristic $\frac{w}{2-w}h(n)$. For $w \leq 1$, this is always less than $h(n)$ and hence admissible, provided $h(n)$ is itself admissible.

Exercise 3.6.#UNGR

Consider the unbounded version of the regular 2D grid shown in Figure 3.6. The start state is at the origin, $(0,0)$, and the goal state is at (x, y) .

- a. What is the branching factor b in this state space?
- b. How many distinct states are there at depth k (for $k > 0$)?
- c. What is the maximum number of nodes expanded by breadth-first tree search?
- d. What is the maximum number of nodes expanded by breadth-first graph search?
- e. Is $h = |u - x| + |v - y|$ an admissible heuristic for a state at (u, v) ? Explain.
- f. How many nodes are expanded by A* graph search using h ?
- g. Does h remain admissible if some links are removed?
- h. Does h remain admissible if some links are added between nonadjacent states?

- a. The branching factor is 4 (number of neighbors of each location).
- b. The states at depth k form a square rotated at 45 degrees to the grid. Obviously there are a linear number of states along the boundary of the square, so the answer is $4k$.
- c. Without repeated state checking, BFS expends exponentially many nodes: counting precisely, we get $((4^{x+y+1} - 1)/3) - 1$.

- d. There are quadratically many states within the square for depth $x + y$, so the answer is $2(x + y)(x + y + 1) - 1$.
- e. True; this is the Manhattan distance metric.
- f. False; all nodes in the rectangle defined by $(0, 0)$ and (x, y) are candidates for the optimal path, and there are quadratically many of them, all of which may be expended in the worst case.
- g. True; removing links may induce detours, which require more steps, so h is an underestimate.
- h. False; nonlocal links can reduce the actual path length below the Manhattan distance.

Exercise 3.6.#VEGR

n vehicles occupy squares $(1, 1)$ through $(n, 1)$ (i.e., the bottom row) of an $n \times n$ grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in $(i, 1)$ must end up in $(n - i + 1, n)$. On each time step, every one of the n vehicles can move one square up, down, left, or right, or stay put; but if a vehicle stays put, one other adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square.

- a. Calculate the size of the state space as a function of n .
- b. Calculate the branching factor as a function of n .
- c. Suppose that vehicle i is at (x_i, y_i) ; write a nontrivial admissible heuristic h_i for the number of moves it will require to get to its goal location $(n - i + 1, n)$, assuming no other vehicles are on the grid.
- d. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.
 - (i) $\sum_{i=1}^n h_i$.
 - (ii) $\max\{h_1, \dots, h_n\}$.
 - (iii) $\min\{h_1, \dots, h_n\}$.

- a. n^{2n} . There are n vehicles in n^2 locations, so roughly (ignoring the one-per-square constraint) $(n^2)^n = n^{2n}$ states.
- b. 5^n .
- c. Manhattan distance, i.e., $|(n - i + 1) - x_i| + |n - y_i|$. This is exact for a lone vehicle.
- d. Only (iii) $\min\{h_1, \dots, h_n\}$. The explanation is nontrivial as it requires two observations. First, let the *work* W in a given solution be the total *distance* moved by all vehicles over their joint trajectories; that is, for each vehicle, add the lengths of all the steps taken. We have $W \geq \sum_i h_i \geq n \cdot \min\{h_1, \dots, h_n\}$. Second, the total work we can get done per step is $\leq n$. (Note that for every car that jumps 2, another car has to stay put (move 0), so the total work per step is bounded by n .) Hence, completing all the work requires at least $n \cdot \min\{h_1, \dots, h_n\}/n = \min\{h_1, \dots, h_n\}$ steps.

Exercise 3.6.#KNIG

Consider the problem of moving k knights from k starting squares s_1, \dots, s_k to k goal squares g_1, \dots, g_k , on an unbounded chessboard, subject to the rule that no two knights can land on the same square at the same time. Each action consists of moving *up to* k knights simultaneously. We would like to complete the maneuver in the smallest number of actions.

- a. What is the maximum branching factor in this state space, expressed as a function of k ?
 - b. Suppose h_i is an admissible heuristic for the problem of moving knight i to goal g_i by itself. Which of the following heuristics are admissible for the k -knight problem? Of those, which is the best?
 - (i) $\min\{h_1, \dots, h_k\}$.
 - (ii) $\max\{h_1, \dots, h_k\}$.
 - (iii) $\sum_{i=1}^k h_i$.
 - c. Repeat (b) for the case where you are allowed to move only one knight at a time.
-
- a. 9^k . For each of k knights, we have one of 8 moves in addition to the possibility of not moving at all; each action executes one of these 9 choices for each knight (unless some choices conflict by landing on the same square), so there are 9^k actions.
 - b. (i) and (ii) are admissible. If the h_i were exact, (ii) would be exact for the relaxed problem where knights can land on the same square. The h_i are admissible and hence no larger than the exact values, so (ii) is admissible. (i) is no larger than (ii), so (i) is admissible. (iii) is not admissible. For example, if each g_i is one move from its s_i , then (iii) returns k whereas the optimal solution cost is 1.
 - (ii) dominates (i) so it must be as good or better. (iii) is probably of little value since it isn't admissible and completely ignores the capability for parallel moves.
 - c. In this case all are admissible. (i) and (ii) as the problem in part (b) is a relaxation of the problem in this part, and (i) and (ii) are admissible for the relaxed problem. (iii) is admissible because it exact for the relaxed problem where knights can land on the same square.
 - (iii) is best since it dominates the rest and is now admissible.

Exercise 3.6.#IAFA

We saw on page ?? that the straight-line distance heuristic leads greedy best-first search astray on the problem of going from Iasi to Fagaras. However, the heuristic is perfect on the opposite problem: going from Fagaras to Iasi. Are there problems for which the heuristic is misleading in both directions?

Going between Rimnicu Vilcea and Lugoj is one example. The shortest path is the southern one, through Mehadia, Dobra and Craiova. But a greedy search using the straight-line heuristic starting in Rimnicu Vilcea will start the wrong way, heading to Sibiu. Starting at Lugoj, the heuristic will correctly lead us to Mehadia, but then a greedy search will return to Lugoj, and oscillate forever between these two cities.

Exercise 3.6.#HEUE

Invent a heuristic function for the 8-puzzle that sometimes overestimates, and show how it can lead to a suboptimal solution on a particular problem. (You can use a computer to help if you want.) Prove that if h never overestimates by more than c , A* using h returns a solution whose cost exceeds that of the optimal solution by no more than c .

The heuristic $h = h_1 + h_2$ (adding misplaced tiles and Manhattan distance) sometimes overestimates. Now, suppose $h(n) \leq h^*(n) + c$ (as given) and let G_2 be a goal that is suboptimal by more than c , i.e., $g(G_2) > C^* + c$. Now consider any node n on a path to an optimal goal. We have

$$\begin{aligned} f(n) &= g(n) + h(n) \\ &\leq g(n) + h^*(n) + c \\ &\leq C^* + c \\ &\leq g(G_2) \end{aligned}$$

so G_2 will never be expanded before an optimal goal is expanded.

Exercise 3.6.#CONH

Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

A heuristic is consistent iff, for every node n and every successor n' of n generated by any action a ,

$$h(n) \leq c(n, a, n') + h(n')$$

One simple proof is by induction on the number k of nodes on the shortest path to any goal from n . For $k = 1$, let n' be the goal node; then $h(n) \leq c(n, a, n')$. For the inductive case, assume n' is on the shortest path k steps from the goal and that $h(n')$ is admissible by hypothesis; then

$$h(n) \leq c(n, a, n') + h(n') \leq c(n, a, n') + h^*(n') = h^*(n)$$

so $h(n)$ at $k + 1$ steps from the goal is also admissible.

Exercise 3.6.#MSTR

The traveling salesperson problem (TSP) can be solved with the minimum-spanning-tree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

- a. Show how this heuristic can be derived from a relaxed version of the TSP.

- b. Show that the MST heuristic dominates straight-line distance.
- c. Write a problem generator for instances of the TSP where cities are represented by random points in the unit square.
- d. Find an efficient algorithm in the literature for constructing the MST, and use it with A* graph search to solve instances of the TSP.

This exercise reiterates a small portion of the classic work of Held and Karp (1970).

- a. The TSP problem is to find a minimal (total length) path through the cities that forms a closed loop. MST is a relaxed version of that because it asks for a minimal (total length) graph that need not be a closed loop—it can be any fully-connected graph. As a heuristic, MST is admissible—it is always shorter than or equal to a closed loop.
- b. The straight-line distance back to the start city is a rather weak heuristic—it vastly underestimates when there are many cities. In the later stage of a search when there are only a few cities left it is not so bad. To say that MST dominates straight-line distance is to say that MST always gives a higher value. This is obviously true because a MST that includes the goal node and the current node must either be the straight line between them, or it must include two or more lines that add up to more. (This all assumes the triangle inequality.)
- c. See `aima-list/search/domains/tsp.lisp` for a start at this. The file includes a heuristic based on connecting each unvisited city to its nearest neighbor, a close relative to the MST approach.
- d. See Cormen *et al.*, 1990, page 505 for an algorithm that runs in $O(E \log E)$ time, where E is the number of edges. The code repository currently contains a somewhat less efficient algorithm.

Exercise 3.6.#GASC

On page 100, we defined the relaxation of the 8-puzzle in which a tile can move from square A to square B if B is blank. The exact solution of this problem defines **Gaschnig's heuristic** (Gaschnig, 1979). Explain why Gaschnig's heuristic is at least as accurate as h_1 (misplaced tiles), and show cases where it is more accurate than both h_1 and h_2 (Manhattan distance). Explain how to calculate Gaschnig's heuristic efficiently.

The misplaced-tiles heuristic is exact for the problem where a tile can move from square A to square B. As this is a relaxation of the condition that a tile can move from square A to square B if B is blank, Gaschnig's heuristic cannot be less than the misplaced-tiles heuristic. As it is also admissible (being exact for a relaxation of the original problem), Gaschnig's heuristic is therefore more accurate.

If we permute two adjacent tiles in the goal state, we have a state where misplaced-tiles and Manhattan both return 2, but Gaschnig's heuristic returns 3.

To compute Gaschnig's heuristic, repeat the following until the goal state is reached: let B be the current location of the blank; if B is occupied by tile X (not the blank) in the goal

state, move X to B; otherwise, move any misplaced tile to B. Students could be asked to prove that this is the optimal solution to the relaxed problem.

Exercise 3.6.#MANH

We gave two simple heuristics for the 8-puzzle: Manhattan distance and misplaced tiles. Several heuristics in the literature purport to improve on this—see, for example, Nilsson (1971), Mostow and Prieditis (1989), and Hansson *et al.*(1992). Test these claims by implementing the heuristics and comparing the performance of the resulting algorithms.

Students should provide results in the form of graphs and/or tables showing both runtime and number of nodes generated. (Different heuristics have different computation costs.) Run-times may be very small for 8-puzzles, so you may want to assign the 15-puzzle or 24-puzzle instead. The use of pattern databases is also worth exploring experimentally.

EXERCISES 4

SEARCH IN COMPLEX ENVIRONMENTS

4.1 Local Search and Optimization Problems

Exercise 4.1.#ASTF

For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

- a. A hill-climbing algorithm that never visits states with lower value (or higher cost) is guaranteed to find the optimal solution if given enough time to find a solution.
- b. Suppose the temperature schedule for simulated annealing is set to be constant up to time N and zero thereafter. For any finite problem, we can set N large enough so that the algorithm returns an optimal solution with probability 1.
- c. For any local-search problem, hill-climbing will return a global optimum if the algorithm is run starting at any state that is a neighbor of a neighbor of a globally optimal state.
- d. In a nondeterministic, partially observable problem, improving an agent's transition model (i.e., eliminating spurious outcomes that never actually occur) will never increase the size of the agent's belief states.
- e. Stochastic hill climbing is guaranteed to arrive at a global optimum.
- f. In a continuous space, gradient descent with a fixed step size is guaranteed to converge to a local or global optimum.
- g. Gradient descent finds the global optimum from any starting point if and only if the function is convex.

- a. False. Such an algorithm will reach a local optimum and stop (or wander on a plateau).
- b. False. If the temperature is fixed the algorithm always has a certain amount of randomness, so when it stops there is no guarantee it will be in an optimal state. Cooling slowly towards zero is the key to optimality.
- c. False. The intervening neighbor could be a local minimum and the current state is on another slope leading to a local maximum. Consider, for example, starting at the state valued 5 in the linear sequence 7,6,5,0,9,0,0.
- d. True. For any given sequence of actions and observations, removing spurious action outcomes can only eliminate states from the final belief state. (The formal proof is

straightforward.)

- e. False. Stochastic hill climbing chooses from the uphill moves at random, but, unlike simulated annealing, it stops if there are none, so it still gets trapped in local optima.
- f. False. While convexity implies that gradient descent will find a global optimum, the converse is not true. A function can have no local optima without being convex—e.g., in one dimension, it just needs to be strictly decreasing/increasing to the left/right of the global optimum.

Exercise 4.1.#SPCA

Give the name of the algorithm that results from each of the following special cases:

- a. Local beam search with $k = 1$.
- b. Local beam search with one initial state and no limit on the number of states retained.
- c. Simulated annealing with $T = 0$ at all times (and omitting the termination test).
- d. Simulated annealing with $T = \infty$ at all times.
- e. Genetic algorithm with population size $N = 1$.

- a. Local beam search with $k = 1$ is hill-climbing search.
- b. Local beam search with one initial state and no limit on the number of states retained, resembles breadth-first search in that it adds one complete layer of nodes before adding the next layer. Starting from one state, the algorithm would be essentially identical to breadth-first search except that each layer is generated all at once.
- c. Simulated annealing with $T = 0$ at all times: ignoring the fact that the termination step would be triggered immediately, the search would be identical to first-choice hill climbing because every downward successor would be rejected with probability 1. (Exercise may be modified in future printings.)
- d. Simulated annealing with $T = \infty$ at all times is a random-walk search: it always accepts a new state.
- e. Genetic algorithm with population size $N = 1$: if the population size is 1, then the two selected parents will be the same individual; crossover yields an exact copy of the individual; then there is a small chance of mutation. Thus, the algorithm executes a random walk in the space of individuals.

Exercise 4.1.#BRIQ

Exercise BRIO-EXERCISE considers the problem of building railway tracks under the assumption that pieces fit exactly with no slack. Now consider the real problem, in which pieces don't fit exactly but allow for up to 10 degrees of rotation to either side of the “proper” alignment. Explain how to formulate the problem so it could be solved by simulated annealing.

Despite its humble origins, this question raises many of the same issues as the scientifically important problem of protein design. There is a discrete assembly space in which pieces are chosen to be added to the track and a continuous configuration space determined by the

“joint angles” at every place where two pieces are linked. Thus we can define a state as a set of oriented, linked pieces and the associated joint angles in the range $[-10, 10]$, plus a set of un-linked pieces. The linkage and joint angles exactly determine the physical layout of the track; we can allow for (and penalize) layouts in which tracks lie on top of one another, or we can disallow them. The evaluation function would include terms for how many pieces are used, how many loose ends there are, and (if allowed) the degree of overlap. We might include a penalty for the amount of deviation from 0-degree joint angles. (We could also include terms for “interestingness” and “traversability”—for example, it is nice to be able to drive a train starting from any track segment to any other, ending up in either direction without having to lift up the train.) The tricky part is the set of allowed moves. Obviously we can unlink any piece or link an unlinked piece to an open peg with either orientation at any allowed angle (possibly excluding moves that create overlap). More problematic are moves to join a peg and hole on already-linked pieces and moves to change the angle of a joint. Changing one angle may force changes in others, and the changes will vary depending on whether the other pieces are at their joint-angle limit. In general there will be no unique “minimal” solution for a given angle change in terms of the consequent changes to other angles, and some changes may be impossible.

Exercise 4.1.#TSPL

In this exercise, we explore the use of local search methods to solve TSPs of the type defined in Exercise TSP-MST-EXERCISE.

- a. Implement and test a hill-climbing method to solve TSPs. Compare the results with optimal solutions obtained from the A* algorithm with the MST heuristic (Exercise TSP-MST-EXERCISE).
- b. Repeat part (a) using a genetic algorithm instead of hill climbing. You may want to consult Larrañaga *et al.* (1999) for some suggestions for representations.

Here is one simple hill-climbing algorithm:

- Connect all the cities into an arbitrary path.
- Pick two points along the path at random.
- Split the path at those points, producing three pieces.
- Try all six possible ways to connect the three pieces.
- Keep the best one, and reconnect the path accordingly.
- Iterate the steps above until no improvement is observed for a while.

Exercise 4.1.#HILL

Generate a large number of 8-puzzle and 8-queens instances and solve them (where possible) by hill climbing (steepest-ascent and first-choice variants), hill climbing with random restart, and simulated annealing. Measure the search cost and percentage of solved problems and graph these against the optimal solution cost. Comment on your results.

Code not shown.

4.2 Local Search in Continuous Spaces

Exercise 4.2.#SADD

In a continuous state space, a **saddle point** is a point at which all partial derivatives are zero but the point is neither a minimum nor a maximum; instead, it is a minimum in some directions but a maximum in others. Explain how the various continuous-space local search algorithms in the chapter function if started at such a point.

Continuous state space local search requires either gradient or empirical gradient methods.

Empirical methods involve discretizing the continuous state space using some constant δ such that local search algorithms function as before. Nonetheless, too large a choice of δ will change the state space such that it no longer resembles the continuous space and even for a very small δ a solution may be intractable or otherwise infinitesimally distant from the optimum. Thus, as in the discrete case, empirical gradient methods will get stuck on saddle points depending on their dimensionality.

Line search and other local gradient methods attempt to navigate to the local minimum by stepping according to the product of a local gradient and a step size parameter. These methods are affected by ridges or saddle points given that they cannot exhaustively consider the effect of the step size on every possible partial gradient and thus may "miss" the ridge (or valley) leading to the global maximum (or minimum).

Analytic gradient methods (even for only the local gradient), such as that of Newton-Raphson, should conceivably distinguish saddle points and should therefore be able to follow the appropriate ridge or valley, but in high dimensional spaces such analyses as computing the Hessian matrix become intractable and thus suffer the fate of other local gradient search methods.

4.3 Search with Nondeterministic Actions

Exercise 4.3.#CONP

The AND-OR-GRAPH-SEARCH algorithm in Figure 4.11 checks for repeated states only on the path from the root to the current state. Suppose that, in addition, the algorithm were to store *every* visited state and check against that list. (See BREADTH-FIRST-SEARCH in Figure 3.9 for an example.) Determine the information that should be stored and how the algorithm should use that information when a repeated state is found. (*Hint:* You will need to distinguish at least between states for which a successful subplan was constructed previously and states for which no subplan could be found.) Explain how to use labels, as defined in Section 4.3.3, to avoid having multiple copies of subplans.

See Figure S4.1 for the adapted algorithm. For states that OR-SEARCH finds a solution for it records the solution found. If it later visits that state again it immediately returns that solution.

When OR-SEARCH fails to find a solution it has to be careful. Whether a state can be solved depends on the path taken to that solution, as we do not allow cycles. So on failure

```

function AND-OR-GRAPH-SEARCH(problem) returns a conditional plan, or failure
  OR-SEARCH(problem.INITIAL-STATE, problem, [])

function OR-SEARCH(state, problem, path) returns a conditional plan, or failure
  if problem.GOAL-TEST(state) then return the empty plan
  if state has previously been solved then return RECALL-SUCCESS(state)
  if state has previously failed for a subset of path then return failure
  if state is on path then
    RECORD-FAILURE(state, path)
    return failure
  for each action in problem.ACTIONS(state) do
    plan  $\leftarrow$  AND-SEARCH(RESULTS(state, action), problem, [state | path])
    if plan  $\neq$  failure then
      RECORD-SUCCESS(state, [action | plan])
      return [action | plan]
  return failure

function AND-SEARCH(states, problem, path) returns a conditional plan, or failure
  for each si in states do
    plani  $\leftarrow$  OR-SEARCH(si, problem, path)
    if plani = failure then return failure
  return [if s1 then plan1 else if s2 then plan2 else ... if sn-1 then plann-1 else plann]

```

Figure S4.1 AND-OR search with repeated state checking.

OR-SEARCH records the value of *path*. If a state is which has previously failed when *path* contained any subset of its present value, OR-SEARCH returns failure.

To avoid repeating sub-solutions we can label all new solutions found, record these labels, then return the label if these states are visited again. Post-processing can prune off unused labels. Alternatively, we can output a direct acyclic graph structure rather than a tree.

Exercise 4.3.#CONL

Explain precisely how to modify the AND-OR-GRAPH-SEARCH algorithm to generate a cyclic plan if no acyclic plan exists. You will need to deal with three issues: labeling the plan steps so that a cyclic plan can point back to an earlier part of the plan, modifying OR-SEARCH so that it continues to look for acyclic plans after finding a cyclic plan, and augmenting the plan representation to indicate whether a plan is cyclic. Show how your algorithm works on (a) the slippery vacuum world, and (b) the slippery, erratic vacuum world. You might wish to use a computer implementation to check your results.

The question statement describes the required changes in detail, see Figure S4.2 for the modified algorithm. When OR-SEARCH cycles back to a state on *path* it returns a token *loop* which means to loop back to the most recent time this state was reached along the path to it. Since *path* is implicitly stored in the returned plan, there is sufficient information for later

```

function AND-OR-GRAPH-SEARCH(problem) returns a conditional plan, or failure
  OR-SEARCH(problem.INITIAL-STATE, problem, [])

function OR-SEARCH(state, problem, path) returns a conditional plan, or failure
  if problem.GOAL-TEST(state) then return the empty plan
  if state is on path then return loop
    cyclic – plan  $\leftarrow$  None
  for each action in problem.ACTIONS(state) do
    plan  $\leftarrow$  AND-SEARCH(RESULTS(state, action), problem, [state | path])
    if plan  $\neq$  failure then
      if plan is acyclic then return [action | plan]
      cyclic – plan  $\leftarrow$  [action | plan]
    if cyclic – plan  $\neq$  None then return cyclic – plan
  return failure

function AND-SEARCH(states, problem, path) returns a conditional plan, or failure
  loopy  $\leftarrow$  True
  for each si in states do
    plani  $\leftarrow$  OR-SEARCH(si, problem, path)
    if plani = failure then return failure
    if plani  $\neq$  loop then loopy  $\leftarrow$  False
  if not loopy then
    return [if s1 then plan1 else if s2 then plan2 else ... if sn-1 then plann-1 else plann]
  return failure

```

Figure S4.2 AND-OR search with repeated state checking.

processing, or a modified implementation, to replace these with labels.

The plan representation is implicitly augmented to keep track of whether the plan is cyclic (i.e., contains a *loop*) so that OR-SEARCH can prefer acyclic solutions.

AND-SEARCH returns failure if all branches lead directly to a *loop*, as in this case the plan will always loop forever. This is the only case it needs to check as if all branches in a finite plan loop there must be some And-node whose children all immediately loop.

Algorithm results

Traces of the algorithm running follow. The state is encoded by specifying whether the squares are clean (c) or dirty (d), with the letter in capitals if the vacuum cleaner is at that spot e.g., Cd means the left square is clean, the right is dirty, and the vacuum is on the left square. Indentation follows nesting of function calls. OR and AND lines are printed at the start of OR-SEARCH and AND-SEARCH, respectively. trying and testing lines are printed at the top of the loops within OR-SEARCH and AND-SEARCH, respectively.

Exercise 4.3.#NPPA

We can turn the navigation problem in Exercise PATH-PLANNING-EXERCISE into an environment as follows:

- The percept will be a list of the positions, *relative to the agent*, of the visible vertices. The percept does *not* include the position of the robot! The robot must learn its own position from the map; for now, you can assume that each location has a different “view.”
 - Each action will be a vector describing a straight-line path to follow. If the path is unobstructed, the action succeeds; otherwise, the robot stops at the point where its path first intersects an obstacle. If the agent returns a zero motion vector and is at the goal (which is fixed and known), then the environment teleports the agent to a *random location* (not inside an obstacle).
 - The performance measure charges the agent 1 point for each unit of distance traversed and awards 1000 points each time the goal is reached.
- a. Implement this environment and a problem-solving agent for it. After each teleportation, the agent will need to formulate a new problem, which will involve discovering its current location.
 - b. Document your agent’s performance (by having the agent generate suitable commentary as it moves around) and report its performance over 100 episodes.
 - c. Modify the environment so that 30% of the time the agent ends up at an unintended destination (chosen randomly from the other visible vertices if any; otherwise, no move at all). This is a crude model of the motion errors of a real robot. Modify the agent so that when such an error is detected, it finds out where it is and then constructs a plan to get back to where it was and resume the old plan. Remember that sometimes getting back to where it was might also fail! Show an example of the agent successfully overcoming two successive motion errors and still reaching the goal.
 - d. Now try two different recovery schemes after an error: (1) head for the closest vertex on the original route; and (2) replan a route to the goal from the new location. Compare the performance of the three recovery schemes. Would the inclusion of search costs affect the comparison?
 - e. Now suppose that there are locations from which the view is identical. (For example, suppose the world is a grid with square obstacles.) What kind of problem does the agent now face? What do solutions look like?

The student needs to make several design choices in answering this question. First, how will the vertices of objects be represented? The problem states the percept is a list of vertex positions, but that is not precise enough. Here is one good choice: The agent has an orientation (a heading in degrees). The visible vertexes are listed in clockwise order, starting straight ahead of the agent. Each vertex has a relative angle (0 to 360 degrees) and a distance. We also want to know if a vertex represents the left edge of an obstacle, the right edge, or an interior point. We can use the symbols L, R, or I to indicate this.

The student will need to do some basic computational geometry calculations: intersection of a path and a set of line segments to see if the agent will bump into an obstacle, and visibility calculations to determine the percept. There are efficient algorithms for doing this on a set of line segments, but don’t worry about efficiency; an exhaustive algorithm is ok. If this seems too much, the instructor can provide an environment simulator and ask the student only to

program the agent.

To answer (c), the student will need some exchange rate for trading off search time with movement time. It is probably too complex to make the simulation asynchronous real-time; easier to impose a penalty in points for computation.

For (d), the agent will need to maintain a set of possible positions. Each time the agent moves, it may be able to eliminate some of the possibilities. The agent may consider moves that serve to reduce uncertainty rather than just get to the goal.

4.4 Search in Partially Observable Environments

Exercise 4.4.#CONF

In Section 4.4.1 we introduced belief states to solve sensorless search problems. A sequence of actions solves a sensorless problem if it maps every physical state in the initial belief state b to a goal state. Suppose the agent knows $h^*(s)$, the true optimal cost of solving the physical state s in the fully observable problem, for every state s in b . Find an admissible heuristic $h(b)$ for the sensorless problem in terms of these costs, and prove its admissibility. Comment on the accuracy of this heuristic on the sensorless vacuum problem of Figure 4.14. How well does A* perform?

A sequence of actions is a solution to a belief state problem if it takes every initial physical state to a goal state. We can relax this problem by requiring it take only *some* initial physical state to a goal state. To make this well defined, we'll require that it finds a solution for the physical state with the most costly solution. If $h^*(s)$ is the optimal cost of solution starting from the physical state s , then

$$h(S) = \max_{s \in S} h^*(s)$$

is the heuristic estimate given by this relaxed problem. This heuristic assumes any solution to the most difficult state the agent thinks possible will solve all states.

On the sensorless vacuum cleaner problem in Figure 4.14, h correctly determines the optimal cost for all states except the central three states (those reached by `[suck]`, `[suck, left]` and `[suck, right]`) and the root, for which h estimates to be 1 unit cheaper than they really are. This means A* will expand these three central nodes, before marching towards the solution.

Exercise 4.4.#BELS

This exercise explores subset-superset relations between belief states in sensorless or partially observable environments.

- a. Prove that if an action sequence is a solution for a belief state b , it is also a solution for any subset of b . Can anything be said about supersets of b ?
- b. Explain in detail how to modify graph search for sensorless problems to take advantage of your answers in (a).

- c. Explain in detail how to modify AND-OR search for partially observable problems, beyond the modifications you describe in (b).
- a. An action sequence is a solution for belief state b if performing it starting in any state $s \in b$ reaches a goal state. Since any state in a subset of b is in b , the result is immediate. Any action sequence which is *not* a solution for belief state b is also not a solution for any superset; this is the contrapositive of what we've just proved. One cannot, in general, say anything about arbitrary supersets, as the action sequence need not lead to a goal on the states outside of b . One can say, for example, that if an action sequence solves a belief state b and a belief state b' then it solves the union belief state $b \cup b'$.
- b. On expansion of a node, do not add to the frontier any child belief state which is a superset of a previously explored belief state.
 - c. If you keep a record of previously solved belief states, add a check to the start of OR-search to check whether the belief state passed in is a subset of a previously solved belief state, returning the previous solution in case it is.

Exercise 4.4.#MVLS

On page 128 it was assumed that a given action would have the same cost when executed in any physical state within a given belief state. (This leads to a belief-state search problem with well-defined step costs.) Now consider what happens when the assumption does not hold. Does the notion of optimality still make sense in this context, or does it require modification? Consider also various possible definitions of the “cost” of executing an action in a belief state; for example, we could use the *minimum* of the physical costs; or the *maximum*; or a cost *interval* with the lower bound being the minimum cost and the upper bound being the maximum; or just keep the set of all possible costs for that action. For each of these, explore whether A* (with modifications if necessary) can return optimal solutions.

Consider a very simple example: an initial belief state $\{S_1, S_2\}$, actions a and b both leading to goal state G from either initial state, and

$$\begin{aligned} c(S_1, a, G) &= 3; & c(S_2, a, G) &= 5; \\ c(S_1, b, G) &= 2; & c(S_2, b, G) &= 6. \end{aligned}$$

In this case, the solution $[a]$ costs 3 or 5, the solution $[b]$ costs 2 or 6. Neither is “optimal” in any obvious sense.

In some cases, there *will* be an optimal solution. Let us consider just the deterministic case. For this case, we can think of the cost of a plan as a mapping from each initial physical state to the actual cost of executing the plan. In the example above, the cost for $[a]$ is $\{S_1:3, S_2:5\}$ and the cost for $[b]$ is $\{S_1:2, S_2:6\}$. We can say that plan p_1 *weakly dominates* p_2 if, for each initial state, the cost for p_1 is no higher than the cost for p_2 . (Moreover, p_1 *dominates* p_2 if it weakly dominates it *and* has a lower cost for some state.) If a plan p weakly dominates all others, it is optimal. Notice that this definition reduces to ordinary optimality in

the observable case where every belief state is a singleton. As the preceding example shows, however, a problem may have no optimal solution in this sense. A perhaps acceptable version of A* would be one that returns any solution that is not dominated by another.

To understand whether it is possible to apply A* at all, it helps to understand its dependence on Bellman's (1957) **principle of optimality**: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.* It is important to understand that this is a restriction on performance measures designed to facilitate efficient algorithms, not a general definition of what it means to be optimal.

In particular, if we define the cost of a plan in belief-state space as the minimum cost of any physical realization, we violate Bellman's principle. Modifying and extending the previous example, suppose that a and b reach S_3 from S_1 and S_4 from S_2 , and then reach G from there:

$$\begin{aligned} c(S_1, a, S_3) &= 6; & c(S_2, a, S_4) &= 2; \\ c(S_1, b, S_3) &= 6; & c(S_2, b, S_4) &= 1. \cdot c(S_3, a, G) = 2; & c(S_4, a, G) &= 2; \\ c(S_3, b, G) &= 1; & c(S_4, b, G) &= 9. \end{aligned}$$

In the belief state $\{S_3, S_4\}$, the minimum cost of $[a]$ is $\min\{2, 2\} = 2$ and the minimum cost of $[b]$ is $\min\{1, 9\} = 1$, so the optimal plan is $[b]$. In the initial belief state $\{S_1, S_2\}$, the four possible plans have the following costs:

$$[a, a] : \min\{8, 4\} = 4; [a, b] : \min\{7, 11\} = 7; [b, a] : \min\{8, 3\} = 3; [b, b] : \min\{7, 10\} = 7.$$

Hence, the optimal plan in $\{S_1, S_2\}$ is $[b, a]$, which does *not* choose b in $\{S_3, S_4\}$ even though that is the optimal plan at that point. This counterintuitive behavior is a direct consequence of choosing the minimum of the possible path costs as the performance measure.

This example gives just a small taste of what might happen with nonadditive performance measures. Details of how to modify and analyze A* for general path-dependent cost functions are given by Dechter and Pearl (1985). Many aspects of A* carry over; for example, we can still derive lower bounds on the cost of a path through a given node. For a belief state b , the minimum value of $g(s) + h(s)$ for each state s in b is a lower bound on the minimum cost of a plan that goes through b .

Exercise 4.4.#VACL

Consider the sensorless version of the erratic vacuum world. Draw the belief-state space reachable from the initial belief state $\{1, 2, 3, 4, 5, 6, 7, 8\}$, and explain why the problem is unsolvable.

The belief state space is shown in Figure S4.3. No solution is possible because no path leads to a belief state all of whose elements satisfy the goal. If the problem is fully observable, the agent reaches a goal state by executing a sequence such that *Suck* is performed only in a dirty square. This ensures deterministic behavior and every state is obviously solvable.

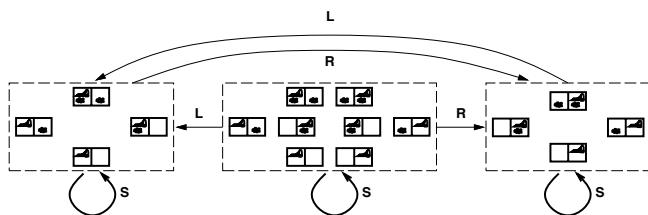


Figure S4.3 The belief state space for the sensorless vacuum world under Murphy's law.

4.5 Online Search Agents and Unknown Environments

Exercise 4.5.#ONOF

Suppose that an agent is in a 3×3 maze environment like the one shown in Figure 4.19. The agent knows that its initial location is (1,1), that the goal is at (3,3), and that the actions *Up*, *Down*, *Left*, *Right* have their usual effects unless blocked by a wall. The agent does *not* know where the internal walls are. In any given state, the agent perceives the set of legal actions; it can also tell whether the state is one it has visited before. Assume that the agent's percept is exactly the set of unblocked directions (i.e., blocked directions are illegal actions).

- Explain how this online search problem can be viewed as an offline search in belief-state space, where the initial belief state includes all possible environment configurations. How large is the initial belief state? How large is the space of belief states?
- How many distinct percepts are possible in the initial state?
- Describe the first few branches of a contingency plan for this problem. How large (roughly) is the complete plan?

Notice that this contingency plan is a solution for *every possible environment* fitting the given description. Therefore, interleaving of search and execution is not strictly necessary even in unknown environments.

There are 12 possible locations for internal walls, so there are $2^{12} = 4096$ possible environment configurations. A belief state designates a *subset* of these as possible configurations; for example, before seeing any percepts all 4096 configurations are possible—this is a single belief state.

- Online search is equivalent to offline search in belief-state space where each action in a belief-state can have multiple successor belief-states: one for each percept the agent could observe after the action. A successor belief-state is constructed by taking the previous belief-state, itself a set of states, replacing each state in this belief-state by the successor state under the action, and removing all successor states which are inconsistent with the percept. This is exactly the construction in Section 4.4.2. AND-OR search can be used to solve this search problem. The initial belief state has $2^{10} = 1024$ states in it, as we know whether two edges have walls or not (the upper and right edges have no walls) but nothing more. There are $2^{2^{12}}$ possible belief states, one for each set

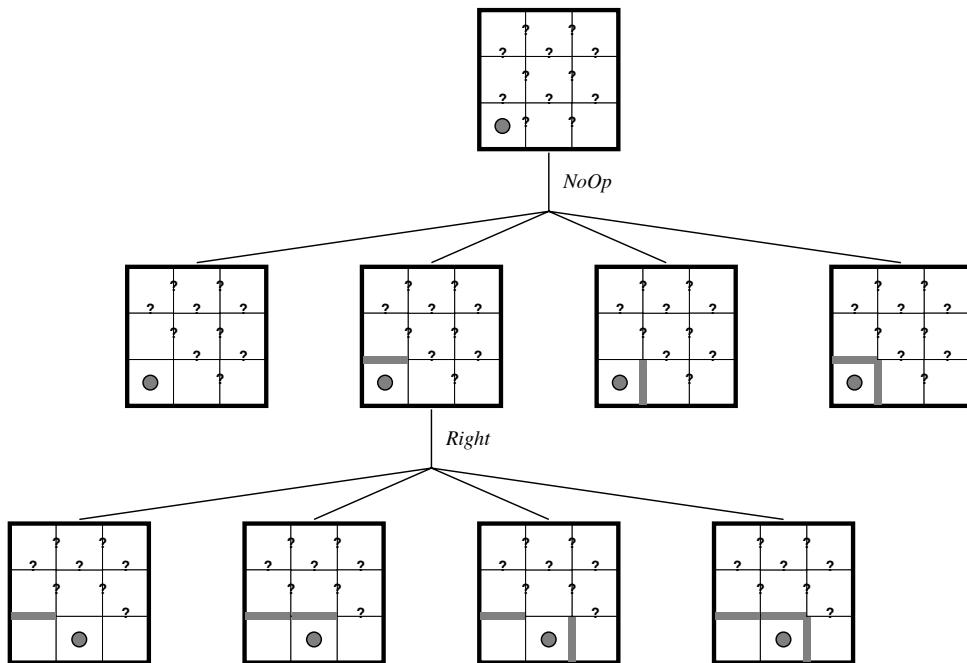


Figure S4.4 The 3×3 maze exploration problem: the initial state, first percept, and one selected action with its perceptual outcomes.

of environment configurations.

We can view this as a contingency problem in belief state space. After each action and percept, the agent learns whether or not an internal wall exists between the current square and each neighboring square. Hence, each reachable belief state can be represented exactly by a list of status values (present, absent, unknown) for each wall separately. That is, the belief state is completely decomposable and there are exactly 3^{12} reachable belief states. The maximum number of possible wall-percepts in each state is 16 (2^4), so each belief state has four actions, each with up to 16 nondeterministic successors.

- b. Assuming the external walls are known, there are two internal walls and hence $2^2 = 4$ possible percepts.
- c. The initial null action leads to four possible belief states, as shown in Figure S4.4. From each belief state, the agent chooses a single action which can lead to up to 8 belief states (on entering the middle square). Given the possibility of having to retrace its steps at a dead end, the agent can explore the entire maze in no more than 18 steps, so the complete plan (expressed as a tree) has no more than 8^{18} nodes. On the other hand, there are just 3^{12} reachable belief states, so the plan could be expressed more concisely as a table of actions indexed by belief state (a **policy** in the terminology of Chapter 17).

Exercise 4.5.#PPHC

In this exercise, we examine hill climbing in the context of robot navigation, using the environment in Figure ?? as an example.

- a. Repeat Exercise PATH-PLANNING-AGENT-EXERCISE using hill climbing. Does your agent ever get stuck in a local minimum? Is it *possible* for it to get stuck with convex obstacles?
- b. Construct a nonconvex polygonal environment in which the agent gets stuck.
- c. Modify the hill-climbing algorithm so that, instead of doing a depth-1 search to decide where to go next, it does a depth- k search. It should find the best k -step path and do one step along it, and then repeat the process.
- d. Is there some k for which the new algorithm is guaranteed to escape from local minima?
- e. Explain how LRTA* enables the agent to escape from local minima in this case.

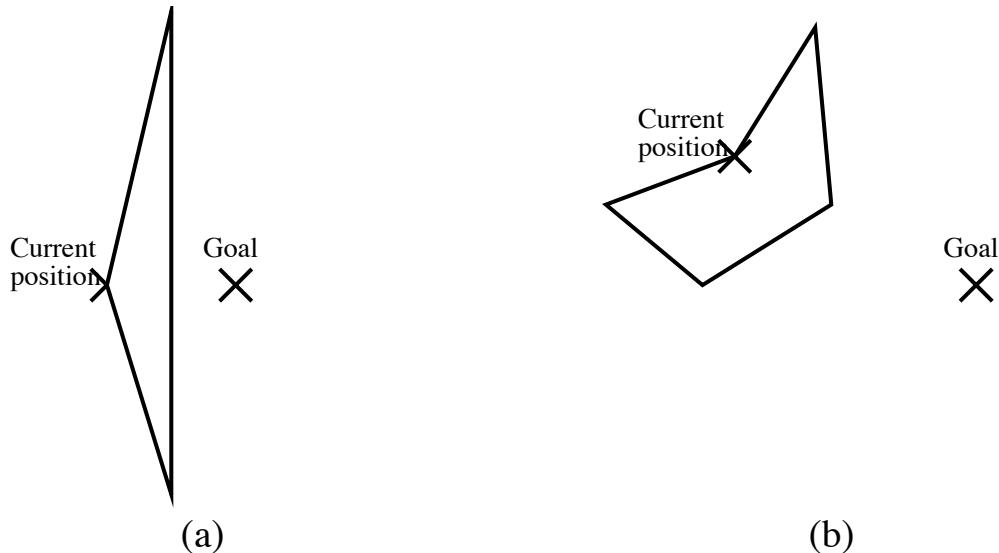


Figure S4.5 (a) Getting stuck with a convex obstacle. (b) Getting stuck with a nonconvex obstacle.

Hillclimbing is surprisingly effective at finding reasonable if not optimal paths for very little computational cost, and seldom fails in two dimensions.

- a. It is possible (see Figure S4.5(a)) but very unlikely—the obstacle has to have an unusual shape and be positioned correctly with respect to the goal.
- b. With nonconvex obstacles, getting stuck is much more likely to be a problem (see Figure S4.5(b)).
- c. Notice that this is just depth-limited search, where you choose a step along the best path even if it is not a solution.

Exercises 4 Search in Complex Environments

- d. Set k to the maximum number of sides of any polygon and you can always escape.
- e. LRTA* always makes a move, but may move back if the old state looks better than the new state. But then the old state is penalized for the cost of the trip, so eventually the local minimum fills up and the agent escapes.

Exercise 4.5.#ODFS

Like DFS, online DFS is incomplete for reversible state spaces with infinite paths. For example, suppose that states are points on the infinite two-dimensional grid and actions are unit vectors $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$, tried in that order. Show that online DFS starting at $(0, 0)$ will not reach $(1, -1)$. Suppose the agent can observe, in addition to its current state, all successor states and the actions that would lead to them. Write an algorithm that is complete even for bidirected state spaces with infinite paths. What states does it visit in reaching $(1, -1)$?

Since we can observe successor states, we always know how to backtrack from to a previous state. This means we can adapt iterative deepening search to solve this problem. The only difference is backtracking must be explicit, following the action which the agent can see leads to the previous state.

The algorithm expands the following nodes:

Depth 1: $(0, 0)$, $(1, 0)$, $(0, 0)$, $(-1, 0)$, $(0, 0)$

Depth 2: $(0, 1)$, $(0, 0)$, $(0, -1)$, $(0, 0)$, $(1, 0)$, $(2, 0)$, $(1, 0)$, $(0, 0)$, $(1, 0)$, $(1, 1)$, $(1, 0)$, $(1, -1)$

Exercise 4.5.#CLRT

Relate the time complexity of LRTA* to its space complexity.

The space complexity of LRTA* is dominated by the space required for $result[a, s]$, i.e., the product of the number of states visited (n) and the number of actions tried per state (m). The time complexity is at least $O(nm^2)$ for a naive implementation because for each action taken we compute an H value, which requires minimizing over actions. A simple optimization can reduce this to $O(nm)$. This expression assumes that each state-action pair is tried at most once, whereas in fact such pairs may be tried many times, as the example in Figure 4.22 shows.

EXERCISES 5

ADVERSARIAL SEARCH AND GAMES

5.1 Game Theory

Exercise 5.1.#ORAC

Suppose you have an oracle, $OM(s)$, that correctly predicts the opponent's move in any state. Using this, formulate the definition of a game as a (single-agent) search problem. Describe an algorithm for finding the optimal move.

The translation uses the model of the opponent $OM(s)$ to fill in the opponent's actions, leaving our actions to be determined by the search algorithm. Let $P(s)$ be the state predicted to occur after the opponent has made all their moves according to OM . Note that the opponent may take multiple moves in a row before we get a move, so we need to define this recursively. We have $P(s) = s$ if PLAYERS is us or TERMINAL-TEST s is true, otherwise $P(s) = P(\text{RESULT}(s, OM(s)))$.

The search problem is then given by:

- a. Initial state: $P(S_0)$ where S_0 is the initial game state. We apply P as the opponent may play first
- b. Actions: defined as in the game by ACTION s .
- c. Successor function: $\text{RESULT}'(s, a) = P(\text{RESULT}(s, a))$
- d. Goal test: goals are terminal states
- e. Step cost: the cost of an action is zero unless the resulting state s' is terminal, in which case its cost is $M - \text{UTILITY}(s')$ where $M = \max_s \text{UTILITY}(s)$. Notice that all costs are non-negative.

Notice that the state space of the search problem consists of game state where we are to play and terminal states. States where the opponent is to play have been compiled out. One might alternatively leave those states in but just have a single possible action.

Any of the search algorithms of Chapter 3 can be applied. For example, depth-first search can be used to solve this problem, if all games eventually end. This is equivalent to using the minimax algorithm on the original game if $OM(s)$ always returns the minimax move in s .

Exercise 5.1.#TEPZ

Consider the problem of solving two 8-puzzles: there are two 8-puzzle boards, you can make a move on either one, and the goal is to solve both.

- a. Give a complete problem formulation.
 - b. How large is the reachable state space? Give an exact numerical expression.
 - c. Suppose we make the problem adversarial as follows: two players take turns moving; a coin is flipped to determine the puzzle on which to make a move in that turn: heads puzzle one; tails puzzle two. The winner is the first to solve either puzzle. Which algorithm can be used to choose a move in this setting?
 - d. In the adversarial problem, will one player eventually win if both play perfectly?
-
- a. Initial state: two arbitrary 8-puzzle states. Successor function: one move on an unsolved puzzle. (You could also have actions that change both puzzles at the same time; this is OK but technically you have to say what happens when one is solved but not the other.) Goal test: both puzzles in goal state. Path cost: 1 per move.
 - b. Each puzzle has $9!/2$ reachable states (remember that half the states are unreachable). The joint state space has $(9!)^2/4$ states.
 - c. This is like backgammon; expectiminimax works.
 - d. No. Consider a state in which the coin flip mandates that you work on a puzzle that is 2 steps from the goal. Should you move one step closer? If you do, your opponent wins if they get to work on that puzzle on the next turn, or on any subsequent turn before you do. So the opponent's probability of winning is *at least* $1/2 + 1/8 + 1/32 + \dots = 2/3$. Therefore you're better off moving *away* from the goal. (There's no way to stay the same distance from the goal.)

Exercise 5.1.#PUEV

Imagine that the problem in Exercise 3.ROMF, in which two friends try to meet up on the map of Romania, is modified so that one of the friends wants to avoid the other. The problem then becomes a two-player **pursuit–evasion** game. We assume now that the players take turns moving. The game ends only when the players are on the same node; the terminal payoff to the pursuer is minus the total time taken. (The evader “wins” by never losing.) An example is shown in Figure ??.

- a. Copy the game tree and mark the values of the terminal nodes.
- b. Next to each internal node, write the strongest fact you can infer about its value (a number, one or more inequalities such as “ ≥ 14 ”, or a “?”).
- c. Beneath each question mark, write the name of the node reached by that branch.
- d. Explain how a bound on the value of the nodes in (c) can be derived from consideration of shortest-path lengths on the map, and derive such bounds for these nodes. Remember the cost to get to each leaf as well as the cost to solve it.
- e. Now suppose that the tree as given, with the leaf bounds from (d), is evaluated from left to right. Circle those “?” nodes that would *not* need to be expanded further, given the bounds from part (d), and cross out those that need not be considered at all.
- f. Can you prove anything in general about who wins the game on a map that is a tree?

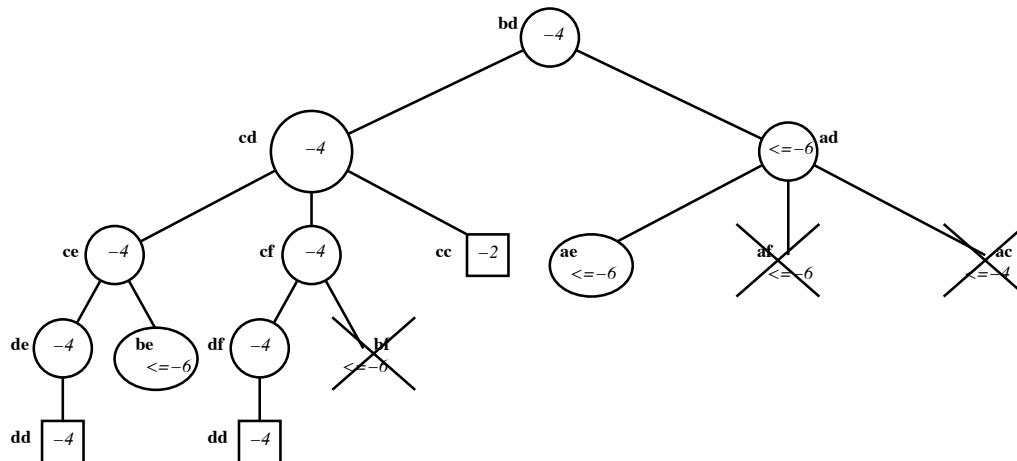


Figure S5.1 Pursuit-evasion solution tree.

- See Figure S5.1; the values are just (minus) the number of steps along the path from the root.
- See Figure S5.1; note that there is both an upper bound and a lower bound for the left child of the root.
- See figure.
- The shortest-path length between the two players is a lower bound on the total capture time (here the players take turns, so no need to divide by two), so the “?” leaves have a capture time greater than or equal to the sum of the cost from the root and the shortest-path length. Notice that this bound is derived when the Evader plays very badly. The true value of a node comes from best play by both players, so we can get better bounds by assuming better play. For example, we can get a better bound from the cost when the Evader simply moves backwards and forwards rather than moving towards the Pursuer.
- See figure (we have used the simple bounds). Notice that once the right child is known to have a value below -6 , the remaining successors need not be considered.
- The pursuer always wins if the tree is finite. To prove this, let the tree be rooted as the pursuer’s current node. (I.e., pick up the tree by that node and dangle all the other branches down.) The evader must either be at the root, in which case the pursuer has won, or in some subtree. The pursuer takes the branch leading to that subtree. This process repeats at most d times, where d is the maximum depth of the original subtree, until the pursuer either catches the evader or reaches a leaf node. Since the leaf has no subtrees, the evader must be at that node.

Exercise 5.1.#GAME

Describe and implement state descriptions, move generators, terminal tests, utility functions, and evaluation functions for one or more of the following stochastic games: Monopoly,

Scrabble, bridge play with a given contract, or Texas hold 'em poker.

The basic physical states of these games are fairly easy to describe. One important thing to remember for Scrabble and bridge is that the physical state is not accessible to all players and so cannot be provided directly to each player by the environment simulator; rather each player should be provided with the part of the state they can observe: the board and their hand. Particularly in bridge, to play well each player needs to maintain some best guess (or multiple hypotheses) as to the actual state of the world.

Exercise 5.1.#RTSG

Describe and implement a *real-time, multiplayer* game-playing environment, where time is part of the environment state and players are given fixed time allocations.

Code not shown.

Exercise 5.1.#PHGM

Discuss how well the standard approach to game playing would apply to games such as tennis, pool, and croquet, which take place in a continuous physical state space.

The most obvious change is that the space of states, and of actions, are now continuous. For example, in pool, the cueing direction, angle of elevation, speed, and point of contact with the cue ball are all continuous quantities.

The simplest solution is just to discretize the action space and then apply standard methods. This might work for tennis (modelled crudely as alternating shots with speed and direction), but for games such as pool and croquet it is likely to fail miserably because small changes in direction have large effects on action outcome. Instead, one must analyze the game to identify a discrete set of meaningful local goals, such as “potting the 4-ball” in pool or “laying up for the next hoop” in croquet. Then, in the current context, a local optimization routine can work out the best way to achieve each local goal, resulting in a discrete set of possible choices. Typically, these games are stochastic, so the backgammon model is appropriate provided that we use sampled outcomes instead of summing over the infinite number of possible outcomes.

Whereas pool and croquet are modelled correctly as turn-taking games, tennis is not. While one player is moving to the ball, the other player is moving to anticipate the opponent’s return. This makes tennis more like the simultaneous-action games studied in Chapter 17. In particular, it may be reasonable to derive *randomized* strategies so that the opponent cannot anticipate where the ball will go.

5.2 Optimal Decisions in Games

Exercise 5.2.#MMXO

Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Can you come up with a game tree in which MAX can do still better using a *suboptimal* strategy against a suboptimal MIN?

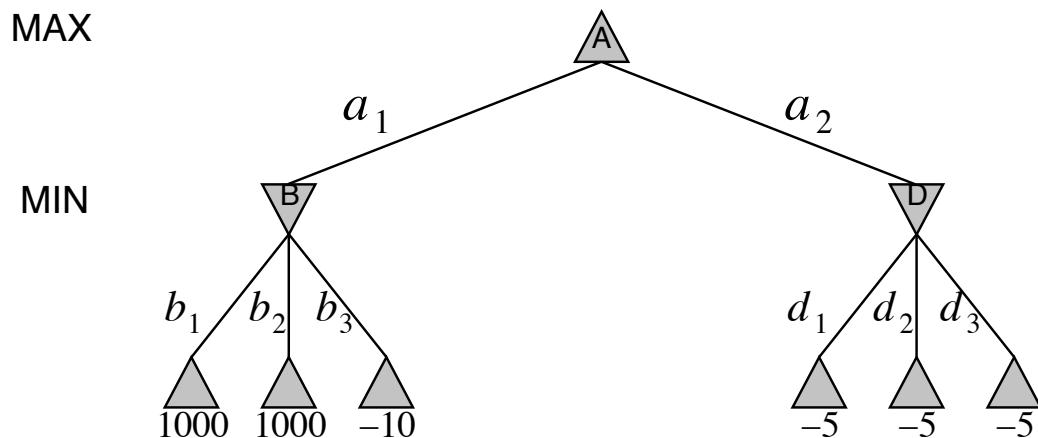


Figure S5.2 A simple game tree showing that setting a trap for MIN by playing a_i is a win if MIN falls for it, but may also be disastrous. The minimax move is of course a_2 , with value -5 .

Consider a MIN node whose children are terminal nodes. If MIN plays suboptimally, then the value of the node is greater than or equal to the value it would have if MIN played optimally. Hence, the value of the MAX node that is the MIN node's parent can only be increased. This argument can be extended by a simple induction all the way to the root. *If the suboptimal play by MIN is predictable*, then one can do better than a minimax strategy. For example, if MIN always falls for a certain kind of trap and loses, then setting the trap guarantees a win even if there is actually a devastating response for MIN. This is shown in Figure S5.2.

Exercise 5.2.#AKQP

Consider the simplified game of poker where there are exactly two players and three cards, ranked in the following order: Ace beats King and King beats Queen. Each player is dealt one of the three cards, which they view independently. Then the players simultaneously declare whether they are going to wager \$1. If both players wager \$1, then the player with the higher card takes all the money. If only one player wagers, then that player takes all the

money regardless of the cards. If neither player bets, no money changes hands.

- Which formalisms would work for this game? (a) expectiminimax search, (b) Greedy search, (c) CSP, (d) Minimax search?
- Under what conditions would α - β pruning work?

- expectiminimax is not appropriate because there are

Exercise 5.2.#GMTR

Consider the two-player game described in Figure ??.

- Draw the complete game tree, using the following conventions:
 - Write each state as (s_A, s_B) , where s_A and s_B denote the token locations.
 - Put each terminal state in a square box and write its game value in a circle.
 - Put *loop states* (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a “?” in a circle.
- Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the “?” values and why.
- Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?
- This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins if n is even and loses if n is odd.

- The game tree, complete with annotations of all minimax values, is shown in Figure S5.3.
- The “?” values are handled by assuming that an agent with a choice between winning the game and entering a “?” state will always choose the win. That is, $\min(-1,?)$ is -1 and $\max(+1,?)$ is $+1$. If all successors are “?”, the backed-up value is “?”.
- Standard minimax is depth-first and would go into an infinite loop. It can be fixed by comparing the current state against the stack; and if the state is repeated, then return a “?” value. Propagation of “?” values is handled as above. Although it works in this case, it does not *always* work because it is not clear how to compare “?” with a drawn position; nor is it clear how to handle the comparison when there are wins of different degrees (as in backgammon). Finally, in games with chance nodes, it is unclear how to compute the average of a number and a “?”. Note that it is *not* correct to treat repeated states automatically as drawn positions; in this example, both $(1,4)$ and $(2,4)$ repeat in the tree but they are won positions.

What is really happening is that each state has a well-defined but initially unknown value. These unknown values are related by the minimax equation at the bottom of 149. If the game tree is acyclic, then the minimax algorithm solves these equations by

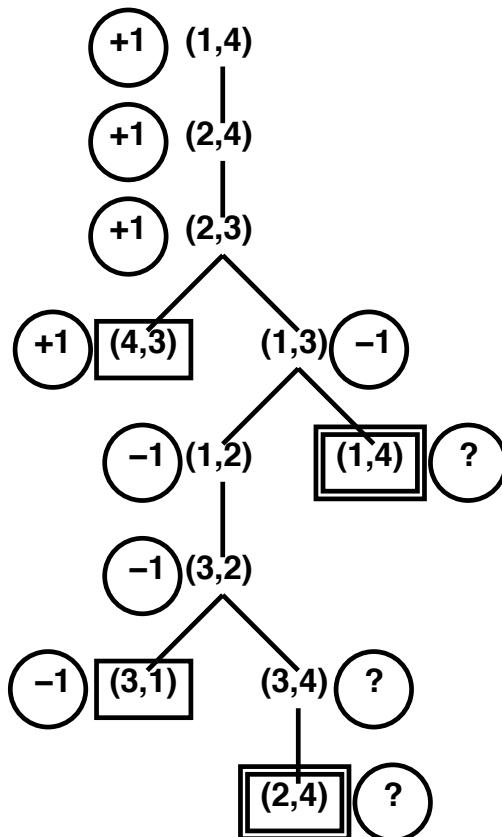


Figure S5.3 The game tree for the four-square game in Exercise 5.8. Terminal states are in single boxes, loop states in double boxes. Each state is annotated with its minimax value in a circle.

propagating from the leaves. If the game tree has cycles, then a dynamic programming method must be used, as explained in Chapter 17. (Exercise 17.7 studies this problem in particular.) These algorithms can determine whether each node has a well-determined value (as in this example) or is really an infinite loop in that both players prefer to stay in the loop (or have no choice). In such a case, the rules of the game will need to define the value (otherwise the game will never end). In chess, for example, a state that occurs 3 times (and hence is assumed to be desirable for both players) is a draw.

- d. This question is a little tricky. One approach is a proof by induction on the size of the game. Clearly, the base case $n = 3$ is a loss for A and the base case $n = 4$ is a win for A. For any $n > 4$, the initial moves are the same: A and B both move one step towards each other. Now, we can see that they are engaged in a subgame of size $n - 2$ on the squares $[2, \dots, n - 1]$, except that there is an extra choice of moves on squares 2 and $n - 1$. Ignoring this for a moment, it is clear that if the " $n - 2$ " is won for A, then A gets to the square $n - 1$ before B gets to square 2 (by the definition of winning) and therefore gets to n before B gets to 1, hence the " n " game is won for A. By the same line of reasoning, if " $n - 2$ " is won for B then " n " is won for B. Now, the presence of

the extra moves complicates the issue, but not too much. First, the player who is slated to win the subgame $[2, \dots, n - 1]$ never moves back to his home square. If the player slated to lose the subgame does so, then it is easy to show that he is bound to lose the game itself—the other player simply moves forward and a subgame of size $n - 2k$ is played one step closer to the loser’s home square.

5.3 Heuristic Alpha–Beta Tree Search

Exercise 5.3.#TTTG

This problem exercises the basic concepts of game playing, using tic-tac-toe (noughts and crosses) as an example. We define X_n as the number of rows, columns, or diagonals with exactly n X ’s and no O ’s. Similarly, O_n is the number of rows, columns, or diagonals with just n O ’s. The utility function assigns +1 to any position with $X_3 = 1$ and -1 to any position with $O_3 = 1$. All other terminal positions have utility 0. For nonterminal positions, we use a linear evaluation function defined as $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$.

- Approximately how many possible games of tic-tac-toe are there?
- Show the whole game tree starting from an empty board down to depth 2 (i.e., one X and one O on the board), taking symmetry into account.
- Mark on your tree the evaluations of all the positions at depth 2.
- Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0, and use those values to choose the best starting move.
- Circle the nodes at depth 2 that would *not* be evaluated if alpha–beta pruning were applied, assuming the nodes are generated in the optimal order for alpha–beta pruning.

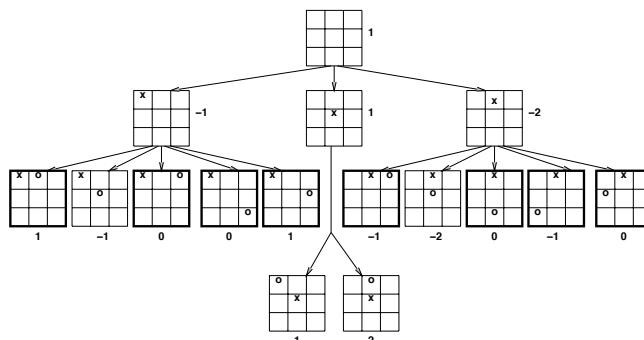


Figure S5.4 Part of the game tree for tic-tac-toe, for Exercise 5.9.

For **a**, there are less than $9!$ games. (This is the number of move sequences that fill up the board, but many wins and losses end before the board is full.) For **b–e**, Figure S5.4 shows the game tree, with the evaluation function values below the terminal nodes and the backed-up

values to the right of the non-terminal nodes. The values imply that the best starting move for X is to take the center. The terminal nodes with a bold outline are the ones that do not need to be evaluated, assuming the optimal ordering.

Exercise 5.3.#GTTT

Consider the family of generalized tic-tac-toe games, defined as follows. Each particular game is specified by a set \mathcal{S} of *squares* and a collection \mathcal{W} of *winning positions*. Each winning position is a subset of \mathcal{S} . For example, in standard tic-tac-toe, \mathcal{S} is a set of 9 squares and \mathcal{W} is a collection of 8 subsets of \mathcal{W} : the three rows, the three columns, and the two diagonals. In other respects, the game is identical to standard tic-tac-toe. Starting from an empty board, players alternate placing their marks on an empty square. A player who marks every square in a winning position wins the game. It is a tie if all squares are marked and neither player has won.

- Let $N = |\mathcal{S}|$, the number of squares. Give an upper bound on the number of nodes in the complete game tree for generalized tic-tac-toe as a function of N .
- Give a lower bound on the size of the game tree for the worst case, where $\mathcal{W} = \{\}$.
- Propose a plausible evaluation function that can be used for any instance of generalized tic-tac-toe. The function may depend on \mathcal{S} and \mathcal{W} .
- Assume that it is possible to generate a new board and check whether it is a winning position in $100N$ machine instructions and assume a 2 gigahertz processor. Ignore memory limitations. Using your estimate in (a), roughly how large a game tree can be completely solved by alpha–beta in a second of CPU time? a minute? an hour?

- An upper bound on the number of terminal nodes is $N!$, one for each ordering of squares, so an upper bound on the total number of nodes is $\sum_{i=1}^N i!$. This is not much bigger than $N!$ itself as the factorial function grows superexponentially. This is an overestimate because some games will end early when a winning position is filled.

This count doesn't take into account transpositions. An upper bound on the number of distinct game states is 3^N , as each square is either empty or filled by one of the two players. Note that we can determine who is to play just from looking at the board.

- In this case no games terminate early, and there are $N!$ different games ending in a draw. So ignoring repeated states, we have exactly $\sum_{i=1}^N i!$ nodes.

At the end of the game the squares are divided between the two players: $\lceil N/2 \rceil$ to the first player and $\lfloor N/2 \rfloor$ to the second. Thus, a good lower bound on the number of distinct states is $\binom{N}{\lceil N/2 \rceil}$, the number of distinct terminal states.

- For a state s , let $X(s)$ be the number of winning positions containing no O's and $O(s)$ the number of winning positions containing no X's. One evaluation function is then $Eval(s) = X(s) - O(s)$. Notice that empty winning positions cancel out in the evaluation function.

Alternatively, we might weight potential winning positions by how close they are to completion.

- Using the upper bound of $N!$ from (a), and observing that it takes $100N N!$ instructions.

Exercises 5 Adversarial Search and Games

At 2GHz we have 2 billion instructions per second (roughly speaking), so solve for the largest N using at most this many instructions. For one second we get $N = 9$, for one minute $N = 11$, and for one hour $N = 12$.

Exercise 5.3.#MMBD

In a full-depth minimax search of a tree with depth D and branching factor B , with $\alpha - \beta$ pruning, what is the minimum number of leaves that must be explored to compute the best move?

There are B^D leaf nodes. In $\alpha - \beta$ pruning we have to look at all the first player's moves but we only have to look at one of the second players moves. Thus we skip every other level and only have to look at $B^{D/2}$.

Exercise 5.3.#MMTD

In a minimax tree with a branching factor of 3 and depth 2 (one max layer, one min layer with 3 nodes, and a leaf layer with 9 total nodes) what is the maximum number of nodes that can be pruned by alpha-beta pruning?

4 (2 each in the rightmost 2 leaf layers).

Exercise 5.3.#ABTF

Which of the following statements about alpha-beta pruning are true or false?

- Alpha-beta pruning may find an approximately optimal strategy, rather than the minimax optimal strategy.
- Alpha-beta prunes the same number of subtrees regardless of the order of child nodes.
- Alpha-beta generally requires more run-time than minimax on the same game tree.

None of these are true. Alpha-beta will always find the optimal strategy against an opponent that plays optimally. If an ordering heuristic is available, we can expand nodes in an order that maximizes pruning. Alpha-beta will require less run-time than minimax except in contrived cases.

Exercise 5.3.#GGAM

Develop a general game-playing program, capable of playing a variety of games.

- Implement move generators and evaluation functions for one or more of the following games: Kalah, Connect Four, Othello (Reversi), checkers, and chess.
- Construct a general alpha-beta game-playing agent.

- c. Compare the effect of increasing search depth, improving move ordering, and improving the evaluation function. How close does your effective branching factor come to the ideal case of perfect move ordering?
- d. Implement a selective search algorithm, such as B* (Berliner, 1979), conspiracy number search (McAllester, 1988), or MGSS* (Russell and Wrefald, 1989) and compare its performance to A*.

See the online code repository for definitions of several games and game-playing agents. Notice that the game-playing environment is essentially a generic environment with the update function defined by the rules of the game. Turn-taking is achieved by having agents do nothing until it is their turn to move. At the time of writing, blog.gamesolver.org has a nice writeup of solving Connect Four.

Few students will be familiar with kalah, so it is a fair assignment, but the game is boring—depth 6 lookahead and a purely material-based evaluation function are enough to beat most humans. Othello is interesting and about the right level of difficulty for most students. Chess and checkers are sometimes unfair because usually a small subset of the class will be experts while the rest are beginners.

Exercise 5.3.#TPNZ

Describe how the minimax and alpha–beta algorithms change for two-player, non-zero-sum games in which each player has a distinct utility function and both utility functions are known to both players. If there are no constraints on the two terminal utilities, is it possible for any node to be pruned by alpha–beta? What if the player’s utility functions on any state sum to a number between constants $-k$ and k , making the game almost zero-sum?

The minimax algorithm for non-zero-sum games works exactly as for multiplayer games; that is, the evaluation function is a vector of values, one for each player, and the backup step selects whichever vector has the highest value for the player whose turn it is to move. Alpha–beta pruning is not possible in general non-zero-sum games, because an unexamined leaf node might be optimal for both players.

Exercise 5.3.#SUNF

The Sunfish project, github.com/thomasahle/sunfish describes a simple chess engine, a little over 100 lines of Python (plus some data files), that plays at an Expert level. Load the project, experiment with it, see what you can learn, and what you can improve. Suggestions:

- a. Improve the speed with a bitboard representation, parallel search, or a port to C or Rust.
- b. Explain how the MTD search improves on alpha-beta without set bounds.
- c. Add some endgame information.

- d. Add extensions to search when the board is in a nonquiescent state.
- e. Run experiments where you compare two versions of Sunfish, with and without a change, and see which one wins more in a series of games.

Each student will have their own way of approaching this exercise.

Exercise 5.3.#ABPR

Develop a formal proof of correctness for alpha–beta pruning. To do this, consider the situation shown in Figure ???. The question is whether to prune node n_j , which is a max-node and a descendant of node n_1 . The basic idea is to prune it if and only if the minimax value of n_1 can be shown to be independent of the value of n_j .

- a. Node n_1 takes on the minimum value among its children: $n_1 = \min(n_2, n_{21}, \dots, n_{2b_2})$. Find a similar expression for n_2 and hence an expression for n_1 in terms of n_j .
- b. Let l_i be the minimum (or maximum) value of the nodes to the *left* of node n_i at depth i , whose minimax value is already known. Similarly, let r_i be the minimum (or maximum) value of the unexplored nodes to the right of n_i at depth i . Rewrite your expression for n_1 in terms of the l_i and r_i values.
- c. Now reformulate the expression to show that in order to affect n_1 , n_j must not exceed a certain bound derived from the l_i values.
- d. Repeat the process for the case where n_j is a min-node.

This question is not as hard as it looks. The derivation below leads directly to a definition of α and β values. The notation n_i refers to (the value of) the node at depth i on the path from the root to the leaf node n_j . Nodes $n_{i1} \dots n_{ib_i}$ are the siblings of node i .

- a. We can write $n_2 = \max(n_3, n_{31}, \dots, n_{3b_3})$, giving

$$n_1 = \min(\max(n_3, n_{31}, \dots, n_{3b_3}), n_{21}, \dots, n_{2b_2})$$

Then n_3 can be similarly replaced, until we have an expression containing n_j itself.

- b. In terms of the l and r values, we have

$$n_1 = \min(l_2, \max(l_3, n_3, r_3), r_2)$$

Again, n_3 can be expanded out down to n_j . The most deeply nested term will be $\min(l_j, n_j, r_j)$.

- c. If n_j is a max node, then the lower bound on its value only increases as its successors are evaluated. Clearly, if it exceeds l_j it will have no further effect on n_1 . By extension, if it exceeds $\min(l_2, l_4, \dots, l_j)$ it will have no effect. Thus, by keeping track of this value we can decide when to prune n_j . This is exactly what α - β does.
- d. The corresponding bound for min nodes n_k is $\max(l_3, l_5, \dots, l_k)$.

Exercise 5.3.#ABPT

Prove that alpha–beta pruning takes time $O(2^{m/2})$ with optimal move ordering, where m is the maximum depth of the game tree.

The result is given in Section 6 of Knuth and Moore (1975). The exact statement (Corollary 1 of Theorem 1) is that the algorithms examines $b^{\lfloor m/2 \rfloor} + b^{\lceil m/2 \rceil} - 1$ nodes at level m . These are exactly the nodes reached when Min plays only optimal moves and/or Max plays only optimal moves. The proof is by induction on m .

Exercise 5.3.#CHET

Suppose you have a chess program that can evaluate 5 million nodes per second. Decide on a compact representation of a game state for storage in a transposition table. About how many entries can you fit in a 32-gigabyte in-memory table? Will that be enough for the three minutes of search allocated for one move? Can you estimate how many table lookups can you do in the time it would take to do one evaluation?

With 32 pieces, each needing 6 bits to specify its position on one of 64 squares, we need 24 bytes (6 32-bit words) to store a position, and let's say 4 more bytes to store a pointer to it. So we can store a little over a billion positions in the table. This is enough for the 900 million positions generated during a three-minute search.

Generating the hash key directly from an array-based representation of the position might be quite expensive. Modern programs (e.g., Heinz, 2000) carry along the hash key and modify it as each new position is generated. Suppose this takes on the order of 20 operations and an evaluation takes 2000 operations. Then we can do roughly 100 lookups per evaluation.

Exercise 5.3.#PRUP

This question considers pruning in games with chance nodes. Figure ?? shows the complete game tree for a trivial game. Assume that the leaf nodes are to be evaluated in left-to-right order, and that before a leaf node is evaluated, we know nothing about its value—the range of possible values is $-\infty$ to ∞ .

- a. Copy the figure, mark the value of all the internal nodes, and indicate the best move at the root with an arrow.
 - b. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers.
 - c. Suppose the leaf node values are known to lie between -2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?
 - d. Circle all the leaves that need not be evaluated under the assumption in (c).
-
- a. See Figure S5.5.

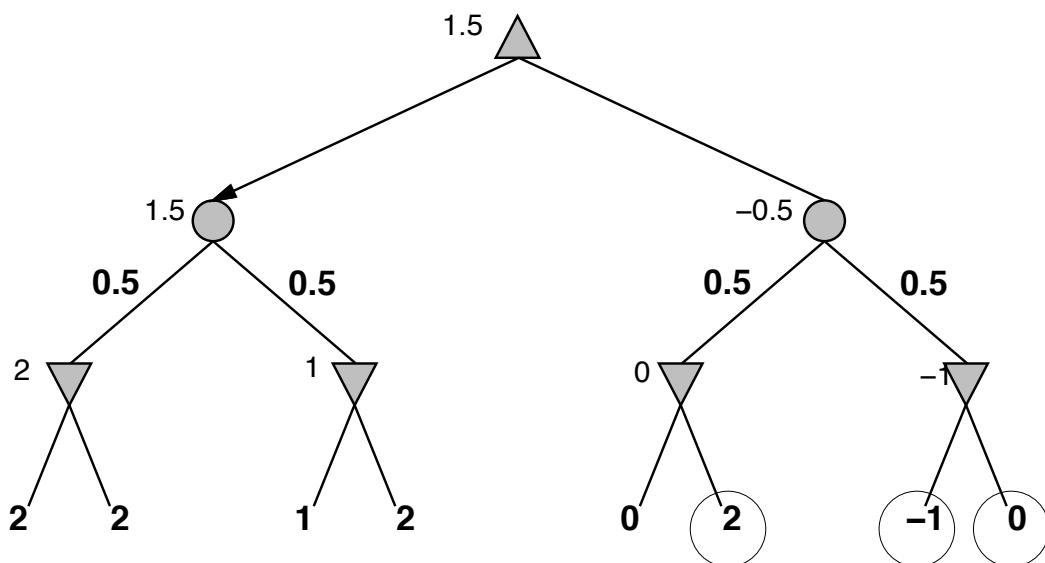


Figure S5.5 Pruning with chance nodes solution.

- b. Given nodes 1–6, we would need to look at 7 and 8: if they were both $+\infty$ then the values of the min node and chance node above would also be $+\infty$ and the best move would change. Given nodes 1–7, we do not need to look at 8. Even if it is $+\infty$, the min node cannot be worth more than -1 , so the chance node above cannot be worth more than -0.5 , so the best move won't change.
- c. The worst case is if either of the third and fourth leaves is -2 , in which case the chance node above is 0. The best case is where they are both 2, then the chance node has value 2. So it must lie between 0 and 2.
- d. See figure.

5.4 Monte Carlo Tree Search

Exercise 5.4.#MCTG

Take one of the games (tic-tac-toe, Connect Four, Kalah, Othello, checkers, etc.) for which you implemented a move generator, and apply Monte Carlo tree search to the move generator. Compare the performance of an MCTS player to an alpha-beta player.

Typically it is easier to apply MCTS; you don't need to be clever about an evaluation function. Performance with MTCS will be good on games that end in about a dozen moves, and will not be as good on games that take a hundred moves (unless students have access to a large computing cluster).

Exercise 5.4.#KRIG

Read the paper “Monte Carlo Tree Search Techniques in the Game of Kriegspiel” by Paolo Ciancarini and Gian Piero Favini, and report on how well Monte Carlo tree search works in a game of imperfect information. Can you replicate the experiment, either for Kriegspiel or for another partially observable game? Can you improve on the techniques used in the paper?

Each student will have their own approach to this exercise.

Exercise 5.4.#SGSM

Read *Monte Carlo Tree Search: A Review of Recent Modifications and Applications*, by Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk, arXiv:2103.04931. Report on one or more of the following ideas for MCTS. Is the idea more help for minimax search or for MCTS? For what kinds of applications is the idea good for?

- a. Transposition tables
- b. History heuristic
- c. Opening books
- d. Action reduction
- e. Early termination

Each student will have their own approach to this exercise.

5.5 Stochastic Games

Exercise 5.5.#EMMX

Implement the expectiminimax algorithm and the *-alpha–beta algorithm, which is described by Ballard (1983), for pruning game trees with chance nodes. Try them on a game such as backgammon and measure the pruning effectiveness of *-alpha–beta.

Exercise 5.5.#EMXC

True or False: In expectiminimax search with two players, one max and the other chance, one can use pruning to reduce the search cost.

Pruning is about exploiting your knowledge of how both players will choose actions to look at less nodes. Since in expectiminimax one player will be picking randomly we have to look at every one of their actions, which will affect the expected value and the value of the node above it.

Exercise 5.5.#EMXP

In each of the cases below, state whether a node can be pruned always, sometimes, or never. Assume that in the expectiminimax game that outcome values are bounded between +1 and -1.

- In a **minimax** game, a leaf node that is the **first** child of its parent.
- In a **expectiminimax** game, a leaf node that is the **first** child of its parent.
- In a **minimax** game, a leaf node that is the **last** child of its parent.
- In a **expectiminimax** game, a leaf node that is the **last** child of its parent.

The first child can never be pruned. Subsequent children can be pruned either in minimax or expectiminimax games. Note that if the game outcomes were not bounded—say in a game of backgammon where the value of the game could be doubled any number of times—then pruning can never be done in expectiminimax games.

Exercise 5.5.#GAML

Prove that with a positive linear transformation of leaf values (i.e., transforming a value x to $ax + b$ where $a > 0$), the choice of move remains unchanged in a game tree, even when there are chance nodes.

The general strategy is to reduce a general game tree to a one-ply tree by induction on the depth of the tree. The inductive step must be done for min, max, and chance nodes, and simply involves showing that the transformation is carried through the node. Suppose that the values of the descendants of a node are $x_1 \dots x_n$, and that the transformation is $ax + b$, where a is positive. We have

$$\begin{aligned}\min(ax_1 + b, ax_2 + b, \dots, ax_n + b) &= a \min(x_1, x_2, \dots, x_n) + b \\ \max(ax_1 + b, ax_2 + b, \dots, ax_n + b) &= a \max(x_1, x_2, \dots, x_n) + b \\ p_1(ax_1 + b) + p_2(ax_2 + b) + \dots + p_n(ax_n + b) &= a(p_1x_1 + p_2x_2 + \dots + p_nx_n) + b\end{aligned}$$

Hence the problem reduces to a one-ply tree where the leaves have the values from the original tree multiplied by the linear transformation. Since $x > y \Rightarrow ax + b > ay + b$ if $a > 0$, the best choice at the root will be the same as the best choice in the original tree.

Exercise 5.5.#MCGM

Consider the following procedure for choosing moves in games with chance nodes:

- Generate some dice-roll sequences (say, 50) down to a suitable depth (say, 8).
- With known dice rolls, the game tree becomes deterministic. For each dice-roll sequence, solve the resulting deterministic game tree using alpha–beta.
- Use the results to estimate the value of each move and to choose the best.

Will this procedure work well? Why (or why not)?

This procedure will give incorrect results. Mathematically, the procedure amounts to assuming that averaging commutes with min and max, which it does not. Intuitively, the choices made by each player in the deterministic trees are based on full knowledge of future dice rolls, and bear no necessary relationship to the moves made without such knowledge. (Notice the connection to the discussion of card games in Section 5.6.2 and to the general problem of fully and partially observable Markov decision problems in Chapter 17.) In practice, the method works reasonably well, and it might be a good exercise to have students compare it to the alternative of using expectiminimax with sampling (rather than summing over) dice rolls.

Exercise 5.5.#SGEP

Consider a game in which three players, A, B, and C, are trying to solve an 8-puzzle. A player receives +1 for making the final move that solves the puzzle, -1 if another player does so. If the same state is repeated 3 times, the game ends with everyone receiving 0. But the players do not just alternate turns; instead before each turn a fair die roll determines whether A, B, or C will move. The die has been cast, it is A's turn to move, and A sees that the puzzle is two steps from being solved. To keep things simple, remember that in an 8-puzzle every move takes you either one step closer or one step further away from the goal (there is no possibility to stay the same distance from the goal), so each player has essentially two choices. Should A move towards the goal or away from it? Justify your answer *quantitatively*.

If A moves towards the goal, whoever goes next will win; A has a $1/3$ chance of going next, so A's expected payoff is $(1/3)(+1) + (2/3)(-1) = -1/3$. If A moves away, the game is now three steps from being solved. If it ever gets to being two steps away, whoever has the next move will do the same analysis as A and also move away, so no one will ever win. Because the state space is finite, some position will repeat three times. This confirms that the expected payoff of moving away is 0, which is better than $-1/3$, so moving away is the best move.

Exercise 5.5.#MAXT

In the following, a “max” tree consists only of max nodes, whereas an “expectiminimax” tree consists of a max node at the root with alternating layers of chance and max nodes. At chance nodes, all outcome probabilities are nonzero. The goal is to *find the value of the root* with a bounded-depth search. For each of (a)–(f), either give an example or explain why this is impossible.

- a. Assuming that leaf values are finite but unbounded, is pruning (as in alpha–beta) ever possible in a max tree?
- b. Is pruning ever possible in an expectiminimax tree under the same conditions?
- c. If leaf values are all nonnegative, is pruning ever possible in a max tree? Give an example, or explain why not.
- d. If leaf values are all nonnegative, is pruning ever possible in an expectiminimax tree? Give an example, or explain why not.

- e. If leaf values are all in the range $[0, 1]$, is pruning ever possible in a max tree? Give an example, or explain why not.
 - f. If leaf values are all in the range $[0, 1]$, is pruning ever possible in an expectiminimax tree?
 - g. Consider the outcomes of a chance node in an expectiminimax tree. Which of the following evaluation orders is most likely to yield pruning opportunities?
 - (i) Lowest probability first
 - (ii) Highest probability first
 - (iii) Doesn't make any difference
-
- a. No pruning. In a max tree, the value of the root is the value of the best leaf. Any unseen leaf might be the best, so we have to see them all.
 - b. No pruning. An unseen leaf might have a value arbitrarily higher or lower than any other leaf, which (assuming non-zero outcome probabilities) means that there is no bound on the value of any incompletely expanded chance or max node.
 - c. No pruning. Same argument as in (a).
 - d. No pruning. Nonnegative values allow *lower* bounds on the values of chance nodes, but a lower bound does not allow any pruning.
 - e. Yes. If the first successor has value 1, the root has value 1 and all remaining successors can be pruned.
 - f. Yes. Suppose the first action at the root has value 0.6, and the first outcome of the second action has probability 0.5 and value 0; then all other outcomes of the second action can be pruned.
 - g. (ii) Highest probability first. This gives the strongest bound on the value of the node, all other things being equal.

Exercise 5.5.#MMXF

Players MAX and MIN are playing a game with a finite depth of possible moves. MAX calculates the minimax value of the root to be M . Assume that each player has at least 2 possible actions at every turn and that every distinct sequence of moves leads to a distinct score. Which of the following are true?

- a. Assume MIN is playing suboptimally, and MAX **does not** know this. The outcome of the game can be better than M (i.e. higher for MAX).
- b. Assume MAX **knows** player MIN is playing randomly. There exists a policy for MAX such that MAX can guarantee a better outcome than M .
- c. Assume MAX **knows** MIN is playing suboptimally on every move and **knows** the policy π_{MIN} that MIN is using (MAX knows exactly how MIN will play). There exists a policy for MAX such that MAX can guarantee a better outcome than M .

- d. Assume MAX **knows** MIN is playing suboptimally at all times but **does not know** the policy π_{MIN} that MIN is using (MAX knows MIN will choose a suboptimal action at each turn, but does not know which suboptimal action). There exists a policy for MAX such that MAX can guarantee a better outcome than M .
- a. True; MAX may take advantage of MIN's errors without having to know about it ahead of time.
- b. False; MIN is playing randomly, but MIN might randomly make the right move each time.
- c. True; Since MIN is playing a different move from the optimal one each time, the game will end up in a different sequence of moves from the optimal sequence, and we assumed that every sequence of moves arrives at a different outcome, so if MAX maximizes each move then the outcome will be $> M$.
- d. True (same reason as previous answer).

5.6 Partially Observable Games

Exercise 5.6.#TFGM

Which of the following are true and which are false? Give brief explanations.

- a. In a fully observable, turn-taking, zero-sum game between two perfectly rational players, it does not help the first player to know what strategy the second player is using—that is, what move the second player will make, given the first player's move.
- b. In a partially observable, turn-taking, zero-sum game between two perfectly rational players, it does not help the first player to know what move the second player will make, given the first player's move.
- c. A perfectly rational backgammon agent never loses.
- a. *In a fully observable, turn-taking, zero-sum game between two perfectly rational players, it does not help the first player to know what strategy the second player is using—that is, what move the second player will make, given the first player's move.*
True. The second player will play optimally, and so is perfectly predictable up to ties. Knowing which of two equally good moves the opponent will make does not change the value of the game to the first player.
- b. *In a partially observable, turn-taking, zero-sum game between two perfectly rational players, it does not help the first player to know what move the second player will make, given the first player's move.*
False. In a partially observable game, knowing the second player's move tells the first player additional information about the game state that would otherwise be available only to the second player. For example, in Kriegspiel, knowing the opponent's future

move tells the first player where one of the opponent's pieces is; in a card game, it tells the first player one of the opponent's cards.

- c. A perfectly rational backgammon agent never loses.

False. Backgammon is a game of chance, and the opponent may consistently roll much better dice. The correct statement is that the *expected* winnings are optimal. It is suspected, but not known, that when playing first the expected winnings are positive even against an optimal opponent.

Exercise 5.6.#CHPI

Consider carefully the interplay of chance events and partial information in each of the games in Exercise 5.GAME.

- a. For which is the standard expectiminimax model appropriate? Implement the algorithm and run it in your game-playing agent, with appropriate modifications to the game-playing environment.
- b. For which would the scheme described in Exercise 5.MCGM be appropriate?
- c. Discuss how you might deal with the fact that in some of the games, the players do not have the same knowledge of the current state.

One can think of chance events during a game, such as dice rolls, in the same way as hidden but preordained information (such as the order of the cards in a deck). The key distinctions are whether the players can influence what information is revealed and whether there is any asymmetry in the information available to each player.

- a. Expectiminimax is appropriate only for backgammon and Monopoly. In bridge and Scrabble, each player knows the cards/tiles he or she possesses but not the opponents'. In Scrabble, the benefits of a fully rational, randomized strategy that includes reasoning about the opponents' state of knowledge is small (but can make the difference in championship play), but in bridge the questions of knowledge and information disclosure are central to good play.
- b. None, for the reasons described earlier.
- c. Key issues include reasoning about the opponent's beliefs, the effect of various actions on those beliefs, and methods for representing them. Since belief states for rational agents are probability distributions over all possible states (including the belief states of others), this is nontrivial.

5.7 Limitations of Game Search Algorithms

Exercise 5.7.#LIMG

- a. Implement a version of search (either alpha-beta or Monte Carlo) that cuts off the search early if it finds a move that is “obviously” best (using Sunfish or some other game

engine, or code of your own). Compare the performance of this version to the standard version.

- b. Do the same for a version that, when two moves are completely symmetric, considers only one of the two.

[

Students should expect to see measurable but small improvements with either change.

Exercise 5.7.#MENA

Re-implement one of the first machine learning game playing system, a version of MENACE, the Machine Educable Noughts and Crosses (Tic-Tac-Toe) Engine (Michie, 1963). The original version was implemented with 304 matchboxes filled with colored beads. You might find it easier to use a computer.

- a. Before it has learned anything, MENACE will play randomly: each possible move in each position has n chances of being selected. On the very first move there are 9 possible moves, so each has a $n/(9n)$ chance of being selected. MENACE makes its moves against a player (it could be an optimal minimax opponent) who makes their move until the game ends.
- b. If MENACE has won, each of the moves it makes is rewarded by having 3 more chances added to its odds (i.e. an increase from n to $n+3$). If the game is a draw, 1 more chance is added, and if a loss, 1 chance is subtracted.
- c. Record MENACE's history of wins and losses over many games. How long does it take to reach equilibrium where it avoids losing?
- d. Experiment with different choices of values of n , and of the 3/1/-1 rewards. Which choices lead to faster winning performance?
- e. When your program has reached equilibrium, compare its policy to the optimal policy from a minimax algorithm. Report on the results.

It should take about 200 moves to reach equilibrium. The rate of training will vary somewhat depending on the opponent. Does the opponent ever make a mistake? When there are multiple moves that all lead to a draw with best play, does the opponent pick one at random, or always pick the same one?

EXERCISES 6

CONSTRAINT SATISFACTION PROBLEMS

6.1 Defining Constraint Satisfaction Problems

Exercise 6.1.#AUSM

How many solutions are there for the three-color map-coloring problem in Figure 6.1?
How many solutions if four colors are allowed? Two colors?

There are 18 solutions for coloring Australia with three colors. Start with SA, which can have any of three colors. Then moving clockwise, WA can have either of the other two colors, and everything else is strictly determined; that makes 6 possibilities for the mainland, times 3 for Tasmania yields 18. There are no solutions with 2 colors, because there are several places where three territories all neighbor each other. There are 768 solutions with four colors, as demonstrated by the following program (which also verifies the other counts with the result: 3: 18, 4: 768, 2: 0).

```
from itertools import product

def valid(WA, NT, SA, Q, NSW, V, T) -> bool:
    """Is this assignment of colors to territories valid?””
    return (SA not in {WA, NT, Q, NSW, V}
            and WA != NT != Q != NSW != V)

{N: sum(valid(*colors) for colors in product(range(N), repeat=7))
 for N in (3, 4, 2)}
```

Exercise 6.1.#MDBS

Mom, Dad, Baby, Student, Teacher, and Guide are lining up next to each other in six linear spots labeled 1 to 6, one to a spots. Baby needs to line up between Mom and Dad. Student and Teacher need to be next to each other. Guide needs to be at one end, in spot 1 or 6. Formulate this problem as a CSP: list the variables, their domains, and the constraints. Encode unary constraints as a constraint rather than pruning the domain. (No need to solve the problem, just provide variables, domains and constraints.)

- Variables: $\{M, D, B, S, T, G\}$
- Domains: $\{1, 2, 3, 4, 5, 6\}$ for all variables
- Constraints: $\text{alldiff}(M, D, B, S, T, G), |B - M| = 1, |B - P| = 1, |S - T| = 1, G \in \{1, 6\}.$

Exercise 6.1.#KKNI

Consider the problem of placing k knights on an $n \times n$ chessboard such that no two knights are attacking each other, where k is given and $k \leq n^2$.

- Choose a CSP formulation. In your formulation, what are the variables?
 - What are the possible values of each variable?
 - What sets of variables are constrained, and how?
 - Now consider the problem of putting *as many knights as possible* on the board without any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.
-
- Solution A: There is a variable corresponding to each of the n^2 positions on the board.
Solution B: There is a variable corresponding to each knight.
 - Solution A: Each variable can take one of two values, {occupied,vacant}
Solution B: Each variable's domain is the set of squares.
 - Solution A: every pair of squares separated by a knight's move is constrained, such that both cannot be occupied. Furthermore, the entire set of squares is constrained, such that the total number of occupied squares should be k .
Solution B: every pair of knights is constrained, such that no two knights can be on the same square or on squares separated by a knight's move. Solution B may be preferable because there is no global constraint, although Solution A has the smaller state space when k is large.
 - Any solution must describe a *complete-state* formulation because we are using a local search algorithm. For simulated annealing, the successor function must completely connect the space; for random-restart, the goal state must be reachable by hillclimbing from some initial state. Two basic classes of solutions are:
Solution C: ensure no attacks at any time. Actions are to remove any knight, add a knight in any unattacked square, or move a knight to any unattacked square.
Solution D: allow attacks but try to get rid of them. Actions are to remove any knight, add a knight in any square, or move a knight to any square.

Exercise 6.1.#XWRD

Consider the problem of constructing (not solving) crossword puzzles: fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided

and that the task is to fill in the blank squares by using any subset of the word list. Formulate this problem precisely in two ways:

- As a general search problem. Choose an appropriate search algorithm and specify a heuristic function. Is it better to fill in blanks one letter at a time or one word at a time?
- As a constraint satisfaction problem. Should the variables be words or letters?

Which formulation do you think will be better? Why?

a. Crossword puzzle construction can be solved many ways. One simple choice is depth-first search. Each successor fills in a word in the puzzle with one of the words in the dictionary. It is better to go one word at a time rather than one letter at a time, to minimize the number of steps.

b. As a CSP, there are even more choices. You could have a variable for each box in the crossword puzzle; in this case the value of each variable is a letter, and the constraints are that the letters must make words. This approach is feasible with a most-constraining value heuristic. Alternately, we could have each string of consecutive horizontal or vertical boxes be a single variable, and the domain of the variables be words in the dictionary of the right length. The constraints would say that two intersecting words must have the same letter in the intersecting box. Solving a problem in this formulation requires fewer steps, but the domains are larger (assuming a big dictionary) and there are fewer constraints. Both formulations are feasible.

Exercise 6.1.#CSPD

Give precise formulations for each of the following as constraint satisfaction problems:

- Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.
- Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that they can teach.
- Hamiltonian tour: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

a. For rectilinear floor-planning, one possibility is to have a variable for each of the small rectangles, with the value of each variable being a 4-tuple consisting of the x and y coordinates of the upper left and lower right corners of the place where the rectangle will be located. The domain of each variable is the set of 4-tuples that are the right size for the corresponding small rectangle and that fit within the large rectangle. Constraints say that no two rectangles can overlap; for example if the value of variable R_1 is $[0, 0, 5, 8]$, then no other variable can take on a value that overlaps with the $0, 0$ to $5, 8$ rectangle.

b. For class scheduling, one possibility is to have three variables for each class, one with times for values (e.g. MWF8:00, TuTh8:00, MWF9:00, ...), one with classrooms for values

(e.g. Wheeler110, Evans330, ...) and one with instructors for values (e.g. Abelson, Bibel, Canny, ...). Constraints say that only one class can be in the same classroom at the same time, and an instructor can only teach one class at a time. There may be other constraints as well (e.g. an instructor should not have two consecutive classes).

c. For Hamiltonian tour, one possibility is to have one variable for each stop on the tour, with binary constraints requiring neighboring cities to be connected by roads, and an AllDiff constraint that all variables have a different value.

Exercise 6.1.#CRYP

Solve the cryptarithmetic problem in Figure 6.2 by hand, using the strategy of backtracking with forward checking and the MRV and least-constraining-value heuristics. Show the trace of each variable choice and assignment.

The exact steps depend on certain choices you are free to make; here are the ones I made:

- a. Choose the X_3 variable. Its domain is $\{0, 1\}$.
- b. Choose the value 1 for X_3 . (We can't choose 0; it wouldn't survive forward checking, because it would force F to be 0, and the leading digit of the sum must be non-zero.)
- c. Choose F , because it has only one remaining value.
- d. Choose the value 1 for F .
- e. Now X_2 and X_1 are tied for minimum remaining values at 2; let's choose X_2 .
- f. Either value survives forward checking, let's choose 0 for X_2 .
- g. Now X_1 has the minimum remaining values.
- h. Again, arbitrarily choose 0 for the value of X_1 .
- i. The variable O must be an even number (because it is the sum of $T + T$ less than 5 (because $O + O = R + 10 \times 0$). That makes it most constrained.
- j. Arbitrarily choose 4 as the value of O .
- k. R now has only 1 remaining value.
- l. Choose the value 8 for R .
- m. T now has only 1 remaining value.
- n. Choose the value 7 for T .
- o. U must be an even number less than 9; choose U .
- p. The only value for U that survives forward checking is 6.
- q. The only variable left is W .
- r. The only value left for W is 3.
- s. This is a solution.

This is a rather easy (under-constrained) puzzle, so it is not surprising that we arrive at a solution with no backtracking (given that we are allowed to use forward checking).

Exercise 6.1.#NARY

Show how a single ternary constraint such as “ $A + B = C$ ” can be turned into three binary constraints by using an auxiliary variable. You may assume finite domains. (*Hint:* Consider a new variable that takes on values that are pairs of other values, and consider constraints such as “ X is the first element of the pair Y .”) Next, show how constraints with more than three variables can be treated similarly. Finally, show how unary constraints can be eliminated by altering the domains of variables. This completes the demonstration that any CSP can be transformed into a CSP with only binary constraints.

The problem statement sets out the solution fairly completely. To express the ternary constraint on A , B and C that $A + B = C$, we first introduce a new variable, AB . If the domain of A and B is the set of numbers N , then the domain of AB is the set of pairs of numbers from N , i.e. $N \times N$. Now there are three binary constraints, one between A and AB saying that the value of A must be equal to the first element of the pair-value of AB ; one between B and AB saying that the value of B must equal the second element of the value of AB ; and finally one that says that the sum of the pair of numbers that is the value of AB must equal the value of C . All other ternary constraints can be handled similarly.

Now that we can reduce a ternary constraint into binary constraints, we can reduce a 4-ary constraint on variables A, B, C, D by first reducing A, B, C to binary constraints as shown above, then adding back D in a ternary constraint with AB and C , and then reducing this ternary constraint to binary by introducing CD .

By induction, we can reduce any n -ary constraint to an $(n - 1)$ -ary constraint. We can stop at binary, because any unary constraint can be dropped, simply by moving the effects of the constraint into the domain of the variable.

Exercise 6.1.#ZEBR

Consider the following logic puzzle: In five houses, each with a different color, live five persons of different nationalities, each of whom prefers a different brand of candy, a different drink, and a different pet. Given the following facts, the questions to answer are “Where does the zebra live, and in which house do they drink water?”

The Englishman lives in the red house.

The Spaniard owns the dog.

The Norwegian lives in the first house on the left.

The green house is immediately to the right of the ivory house.

The man who eats Hershey bars lives in the house next to the man with the fox.

Kit Kats are eaten in the yellow house.

The Norwegian lives next to the blue house.

The Smarties eater owns snails.

The Snickers eater drinks orange juice.

The Ukrainian drinks tea.

The Japanese eats Milky Ways.

Kit Kats are eaten in a house next to the house where the horse is kept.

Coffee is drunk in the green house.

Milk is drunk in the middle house.

Discuss different representations of this problem as a CSP. Why would one prefer one representation over another?

The “Zebra Puzzle” can be represented as a CSP by introducing a variable for each color, pet, drink, country, and cigarette brand (a total of 25 variables). The value of each variable is a number from 1 to 5 indicating the house number. This is a good representation because it is easy to represent all the constraints given in the problem definition this way. (We have done so in the Python implementation of the code.) Besides ease of expressing a problem, the other reason to choose a representation is the efficiency of finding a solution. Here we have mixed results—on some runs, min-conflicts local search finds a solution for this problem in seconds, while on other runs it fails to find a solution after minutes.

Another representation is to have five variables for each house, one with the domain of colors, one with pets, and so on.

6.2 Constraint Propagation: Inference in CSPs

Exercise 6.2.#GRAE

Consider the graph with 8 nodes $A_1, A_2, A_3, A_4, H, T, F_1, F_2$. A_i is connected to A_{i+1} for all i , each A_i is connected to H , H is connected to T , and T is connected to each F_i . Find a 3-coloring of this graph by hand using the following strategy: backtracking with conflict-directed backjumping, the variable order $A_1, H, A_4, F_1, A_2, F_2, A_3, T$, and the value order R, G, B .

- a. $A_1 = R$.
- b. $H = R$ conflicts with A_1 .
- c. $H = G$.
- d. $A_4 = R$.
- e. $F_1 = R$.
- f. $A_2 = R$ conflicts with A_1 , $A_2 = G$ conflicts with H , so $A_2 = B$.
- g. $F_2 = R$.
- h. $A_3 = R$ conflicts with A_4 , $A_3 = G$ conflicts with H , $A_3 = B$ conflicts with A_2 , so backtrack. Conflict set is $\{A_2, H, A_4\}$, so jump to A_2 . Add $\{H, A_4\}$ to A_2 's conflict set.
- i. A_2 has no more values, so backtrack. Conflict set is $\{A_1, H, A_4\}$ so jump back to A_4 . Add $\{A_1, H\}$ to A_4 's conflict set.
- j. $A_4 = G$ conflicts with H , so $A_4 = B$.
- k. $F_1 = R$

- l.** $A_2 = R$ conflicts with A_1 , $A_2 = G$ conflicts with H , so $A_2 = B$.
- m.** $F_2 = R$
- n.** $A_3 = R$.
- o.** $T = R$ conflicts with F_1 and F_2 , $T = G$ conflicts with G , so $T = B$.
- p.** Success.

Exercise 6.2.#XYAC

Consider a CSP with variables X, Y with domains $\{1, 2, 3, 4, 5, 6\}$ for X and $\{2, 4, 6\}$ for Y , and constraints $X < Y$ and $X + Y > 8$. List the values that will remain in the domain of X after enforcing arc consistency for the arc $X \rightarrow Y$ (which prunes the domain of X , not Y).

The resulting domain of X is $\{3, 4, 5\}$.

Exercise 6.2.#CPTF

Are the following statements true or false?

- a.** Running forward checking after the assignment of a variable in backtracking search will ensure that every variable is arc consistent with every other variable.
- b.** In a CSP constraint graph, a link (edge) between any two variables implies that those two variables may not take on the same values.
- c.** For any particular CSP constraint graph, there exists exactly one minimal cutset.
- d.** Two different CSP search algorithms may give different results on the same constraint satisfaction problem.
- e.** A CSP can only have unary and binary constraints.
- f.** If forward checking eliminates an inconsistent value, enforcing arc consistency would eliminate it as well.
- g.** If enforcing arc consistency eliminates an inconsistent value, forward checking would eliminate it as well.
- h.** When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates depends on the order in which arcs are processed from the queue.

- a.** False. Forward checking only checks variables that are connected to the variable being assigned.
- b.** False. A link represents any constraint, including, e.g., equality!
- c.** False. There may be several minimal cutsets of equal size. This is common when the graph has symmetries.
- d.** True. A CSP may have many solutions, and which one is found first depends on the order in which the space of assignments is searched. If there is only one solution, then all algorithms should agree.

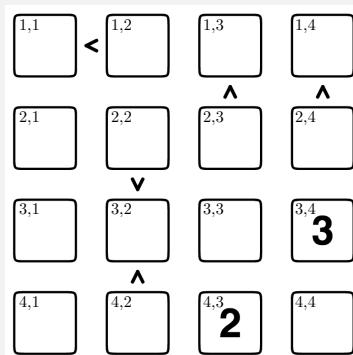
- e. False. There are several examples in the book and elsewhere of constraints with more than two variables, including cryptarithmetic and sudoku problems. (It is possible to reduce any n -ary constraint into a set of binary constraints.)
- f. True.
- g. False.
- h. False. Arc consistency eliminates values which cannot be possible, it does not make choices when there are several possibilities, so any order will end up with the same results.

Exercise 6.2.#FUTO

Futoshiki is a Sudoku-like Japanese logic puzzle that is very simple, but can be quite challenging. You are given an $n \times n$ grid, and must place the numbers $1, \dots, n$ in the grid such that every row and column has exactly one of each. Additionally, the assignment must satisfy the inequalities placed between some adjacent squares. The inequalities apply only to the two adjacent squares, and do not directly constrain other squares in the row or column.

To the right is an instance of this problem,

for size $n = 4$. Some of the squares have known values.



Let's formulate this puzzle as a CSP. We will use 4^2 variables, one for each cell, with X_{ij} as the variable for the cell in the i th row and j th column. The only unary constraints will be those assigning the known initial values to their respective squares (e.g. $X_{34} = 3$).

- a. Complete the formulation of the CSP. Describe the domains of the variables, the two unary constraints, and all binary constraints you think are necessary. You can describe the constraints using concise mathematical notation. Do not use general n -ary constraints (such as *alldiff*).
- b. After enforcing unary constraints, consider the binary constraints relating X_{14} and X_{24} . Enforce arc consistency on just these constraints and state the resulting domains for the two variables.
- c. Suppose we enforced unary constraints and ran arc consistency on this CSP, pruning the domains of all variables as much as possible. After this, what is the maximum possible domain size for any variable? *Hint:* consider the least constrained variable(s); you should *not* have to run every step of arc consistency to answer this.
- d. Suppose we enforced unary constraints and ran arc consistency on the initial CSP in the figure above. What is the maximum possible domain size for a variable adjacent to an inequality?

e. By inspection of column 2, we find it is necessary that $X_{32} = 1$, despite not having found an assignment to any of the other cells in that column. Would running arc consistency find this requirement? Explain why or why not.

a. Domains: $X_{ij} \in \{1, 2, 3, 4\}, \forall i, j$

Unary constraints: $X_{34} = 3, X_{43} = 2$

Inequality binary constraints: $X_{11} < X_{12}, X_{13} < X_{23}, X_{14} < X_{24}, X_{32} < X_{22}, X_{32} < X_{42}$

Row binary constraints: $X_{ij} \neq X_{ik}, \forall i, j, k, j \neq k$

Column binary constraints: $X_{ij} \neq X_{kj}, \forall i, j, k, i \neq k$

b. $X_{14} \in \{1, 2\}, X_{24} \in \{2, 4\}$. Note that both threes are removed from the column constraint with X_{34} .

c. The maximum possible domain size is 4 (ie, no values are removed from the original domain). Consider X_{21} . We will not be able to eliminate any values from its domain through arc consistency.

d. The maximum domain size is 3. An inequality constraint always eliminates either 1 or 4 from a variable's domain.

e. No, arc consistency would not find this requirement. Enforcing the $X_{32} \rightarrow X_{42}$ arc and the $X_{42} \rightarrow X_{43}$ arc leaves X_{42} with a domain of $\{3, 4\}$. Enforcing the $X_{32} < X_{22}$ constraints leaves $X_{32} \in \{1, 2, 3\}$ and $X_{22} \in \{2, 3, 4\}$. Enforcing that they are all different does not remove any values. After this point, every arc in this column is consistent and X_{32} is not required to be 1.

6.3 Backtracking Search for CSPs

Exercise 6.3.#MCON

Explain why it is a good heuristic to choose the variable that is *most* constrained but the value that is *least* constraining in a CSP search.

The most constrained variable makes sense because it chooses a variable that is (all other things being equal) likely to cause a failure, and it is more efficient to fail as early as possible (thereby pruning large parts of the search space). The least constraining value heuristic makes sense because it allows the most chances for future assignments to avoid conflict.

Exercise 6.3.#MAPR

Generate random instances of map-coloring problems as follows: scatter n points on the unit square; select a point X at random, connect X by a straight line to the nearest point Y such that X is not already connected to Y and the line crosses no other line; repeat the previous step until no more connections are possible. The points represent regions on the

map and the lines connect neighbors. Now try to find k -colorings of each map, for both $k = 3$ and $k = 4$, using min-conflicts, backtracking, backtracking with forward checking, and backtracking with MAC. Construct a table of average run times for each algorithm for values of n up to the largest you can manage. Comment on your results.

Exercise 6.3.#ACTA

Use the AC-3 algorithm to show that arc consistency can detect the inconsistency of the partial assignment $\{ WA = \text{green}, V = \text{red} \}$ for the problem shown in Figure 6.1.

We'll trace through each iteration of the **while** loop in AC-3 (for one possible ordering of the arcs):

- a. Remove $SA - WA$, delete G from SA .
- b. Remove $SA - V$, delete R from SA , leaving only B .
- c. Remove $NT - WA$, delete G from NT .
- d. Remove $NT - SA$, delete B from NT , leaving only R .
- e. Remove $NSW - SA$, delete B from NSW .
- f. Remove $NSW - V$, delete R from NSW , leaving only G .
- g. Remove $Q - NT$, delete R from Q .
- h. Remove $Q - SA$, delete B from Q .
- i. remove $Q - NSW$, delete G from Q , leaving no domain for Q .

Exercise 6.3.#ABCT

Consider a CSP with three variables: A , B , and C . Each of the three variables can take on one of two values: either 1 or 2. There are three constraints: $A \neq B$, $B \neq C$, and $A \neq C$. What values for what variables would be eliminated by enforcing *arc-consistency*? Explain your answer.

No values would be eliminated by arc-consistency. There is no solution to this CSP, but arc-consistency does not detect this. For every individual arc either value has a consistent assignment.

Exercise 6.3.#ACWC

What is the worst-case complexity of running AC-3 on a tree-structured CSP?

On a tree-structured graph, no arc will be considered more than once, so the AC-3 algorithm is $O(ED)$, where E is the number of edges and D is the size of the largest domain.

Exercise 6.3.#ACTF

AC-3 puts back on the queue *every* arc (X_k, X_i) whenever *any* value is deleted from the domain of X_i , even if each value of X_k is consistent with several remaining values of X_i . Suppose that, for every arc (X_k, X_i) , we keep track of the number of remaining values of X_i that are consistent with each value of X_k . Explain how to update these numbers efficiently and hence show that arc consistency can be enforced in total time $O(n^2d^2)$.

The basic idea is to preprocess the constraints so that, for each value of X_i , we keep track of those variables X_k for which an arc from X_k to X_i is satisfied by that particular value of X_i . This data structure can be computed in time proportional to the size of the problem representation. Then, when a value of X_i is deleted, we reduce by 1 the count of allowable values for each (X_k, X_i) arc recorded under that value. This is very similar to the forward chaining algorithm in Chapter 7. See Mohr and Henderson (1986) for detailed proofs.

6.4 Local Search for CSPs

Exercise 6.4.#SUDK

We introduced Sudoku as a CSP to be solved by search over partial assignments because that is the way people generally undertake solving Sudoku problems. It is also possible, to attack Sudoku problems with local search over complete assignments. How well would a local solver using the min-conflicts heuristic do on Sudoku problems?

It is certainly possible to solve Sudoku problems in this fashion. However, it is not as effective as the partial-assignment approach, and not as effective as min-conflicts is on the N-queens problem. Perhaps that is because there are two different types of conflicts: a conflict with one of the numbers that defines the initial problem is one that must be corrected, but a conflict between two numbers that were placed elsewhere in the grid can be corrected by replacing either of the two. A version of min-conflicts that recognizes the difference between these two situations might do better than the naive min-conflicts algorithm.

Exercise 6.4.#CSLS

Suppose we are solving a CSP with local search. The problem has N variables, each of which has a domain with d values (for example, in Sudoku, $N = 81$ and $d = 9$). We are interested in the successors of any state in the local search: from a given assignment to variables, what other assignments will we consider, from which we will choose the one with the minimal conflicts. Using $O()$ notation, describe the computational complexity in terms of N and d for each of the following approaches to generating successors:

- Randomly choose a variable, then consider all assignments to that variable.
- For the variable with the most conflicts, consider all assignments to it.

- c. Consider all states that differ by the assignment to exactly one variable.
- d. Consider all states that differ by the assignment to one or two variables. (This can be useful to escape from plateaus).

- a. $O(d)$
- b. $O(d + N)$
- c. $O(Nd)$
- d. $O(N^2d^2)$

Exercise 6.4.#RCSP

Using a CSP solver program and another program to generate random problem instances of CSPs, report on the time to solve the problem as a function of the ratio of the number of constraints to the number of variables.

Details will vary somewhat depending on exactly the structure of the random problems, but solving should be fast except for a small critical range of the constraints-to-variables ratio.

Exercise 6.4.#BCTA

Ali, Bo, Cleo, and Dallas are picking their entrees at a restaurant. The choices are pasta, quesadillas, risotto, and sushi. They have some strict dietary preferences:

- Cleo will not order sushi.
- Ali and Bo want to steal each other's food, so they will order different dishes.
- Bo likes carbs, so he will only order pasta or risotto.
- Cleo likes to be unique in her food orders and will not order the same dish as anybody else, with one exception: Ali and Cleo are actually twins, and always order the same dish as each other.
- Dallas really dislikes quesadillas and will not order them.

Answer the following questions for this situation:

- a. If we formulate this as a CSP with variables $\{A, B, C, D\}$, each with domain $\{P, Q, R, S\}$, what are the constraints?
- b. We will run basic backtracking search to solve this CSP and make sure that every person (variable) is matched with their dream dish (value). We will select unassigned variables in alphabetical order, and we will also iterate over values in alphabetical order. What assignment will backtracking search return?
- c. Now assume that no values have been assigned to the variables and we will run one iteration of forward checking. What value(s) will be eliminated for which variables if we assign "pasta" to Ali?

- d. Now assume we will solve the problem with local search using the min-conflicts algorithm. Assume we start with the initial assignment $\{A = P, B = P, C = P, D = P\}$ and choose B as the variable to change. How many conflicts does the current value of B pose? What value for B would minimize the number of conflicts?
- a. $C \neq S, A \neq B, B \in \{P, R\}, C \notin \{B, D\}, D \neq Q$.
 b. $\{A = P, B = R, C = P, D = R\}$.
 c. Eliminate P from B ; eliminate Q, S, R from C ; eliminate nothing from D .
 d. $B = S$ conflicts with $A \neq B, B \in \{P, R\}$, and $C \notin \{B, D\}$, so 3 conflicts. Setting $B = R$ would resolve all 3 of B 's conflicts; no other value choice would do that.

6.5 The Structure of Problems

Exercise 6.5.#TCSP

The TREE-CSP-SOLVER (Figure 6.10) makes arcs consistent starting at the leaves and working backwards towards the root. Why does it do that? What would happen if it went in the opposite direction?

We establish arc-consistency from the bottom up because we will then (after establishing consistency) solve the problem from the top down. It will always be possible to find a solution (if one exists at all) with no backtracking because of the definition of arc consistency: whatever choice we make for the value of the parent node, there will be a value for the child.

Exercise 6.5.#CSPE

Define in your own words the terms constraint, commutativity, backtracking search, arc consistency, backjumping, min-conflicts, and cycle cutset.

A constraint is a restriction on the possible values of two or more variables. For example, a constraint might say that $A = a$ is not allowed in conjunction with $B = b$.

Two actions are **commutative** if they have the same effect no matter which order they are performed in.

Backtracking search is a form of depth-first search in which there is a single representation of the state that gets updated for each successor, and then must be restored when a dead end is reached.

A directed arc from variable A to variable B in a CSP is **arc consistent** if, for every value in the current domain of A , there is some consistent value of B .

Backjumping is a way of making backtracking search more efficient, by jumping back more than one level when a dead end is reached.

Min-conflicts is a heuristic for use with local search on CSP problems. The heuristic says that, when given a variable to modify, choose the value that conflicts with the fewest

number of other variables.

A **cycle cutset** is a set of variables which when removed from the constraint graph make it acyclic (i.e., a tree). When the variables of a cycle cutset are instantiated the remainder of the CSP can be solved in linear time.

Exercise 6.5.#CGCS

Consider a CSP with a constraint graph consisting of n variables arranged in a circle, where each variable has two constraints, one with each neighbor on either side. Explain how to solve this class of CSPs efficiently, in time $O(n)$.

We can use cutset conditioning to reduce the circle to a tree structure, which can be efficiently solved without backtracking by one backward arc-enforcing pass and one forward value-setting pass. In other words, pick any variable X , set it to some value v from the domain, and execute the efficient tree-solving procedure. Repeat for all values in the domain.

Exercise 6.5.#GCCS

Suppose that a graph is known to have a cycle cutset of no more than k nodes. Describe a simple algorithm for finding a minimal cycle cutset whose run time is not much more than $O(n^k)$ for a CSP with n variables. Search the literature for methods for finding approximately minimal cycle cutsets in time that is polynomial in the size of the cutset. Does the existence of such algorithms make the cycle cutset method practical?

A simple algorithm for finding a cutset of no more than k nodes is to enumerate all subsets of nodes of size $1, 2, \dots, k$, and for each subset check whether the remaining nodes form a tree. This algorithm takes time $(\sum_{i=1}^k n^i)$, which is $O(n^k)$.

Becker and Geiger (1994) give an algorithm called MGA (modified greedy algorithm) that finds a cutset that is no more than twice the size of the minimal cutset, using time $O(E + V \log(V))$, where E is the number of edges and V is the number of variables.

Whether the cycle cutset approach is practical depends more on the graph than on the cutset-finding algorithm. That is because, for a cutset of size c , we still have an exponential (d^c) factor before we can solve the CSP. So any graph with a large cutset will be intractable to solve by this method, even if we could find the cutset with no effort at all.

Exercise 6.5.#TILS

Consider the problem of tiling a surface (completely and exactly covering it) with n dominoes (2×1 rectangles). The surface is an arbitrary edge-connected (i.e., adjacent along an edge, not just a corner) collection of $2n$ 1×1 squares (e.g., a checkerboard, a checkerboard with some squares missing, a 10×1 row of squares, etc.).

- Formulate this problem precisely as a CSP where the dominoes are the variables.
- Formulate this problem precisely as a CSP where the squares are the variables, keeping

the state space as small as possible. (*Hint:* does it matter which particular domino goes on a given pair of squares?)

- c. Construct a surface consisting of 6 squares such that your CSP formulation from part (b) has a *tree-structured* constraint graph.
- d. Describe exactly the set of solvable instances that have a tree-structured constraint graph.

The key aspect of any correct formulation is that a complete assignment satisfying the constraints must be a real solution to the real problem.

- a. Variable domains: all pairs of adjacent squares. You might want to avoid having the same pair of squares appearing twice in different orders.
Constraints: every pair of variables is connected by a constraint stating that their values may not overlap (i.e., they cannot share any square).
- b. Variable domains: the set of (up to four) adjacent squares. The idea is that the domino covering this square can choose exactly one of the adjacent squares to cover too.
Constraints: between every pair of adjacent squares A and B . A can have value B iff B has value A .
- c. The constraints in (b) are binary constraints relating adjacent squares, so they will form a loop whenever the squares form a loop (e.g., any 2 by 2 block). So any problem where the “width” of the shape is 1 is OK, e.g., 6 in a row, and there are no loops.

Exercise 6.5.#BCSP

In a general constraint satisfaction problem with N binary-valued variables, what is the minimum, and the maximum number of times that backtracking search will backtrack, expressed in $O()$ notation (i.e. $O(1)$, $O(n^2)$, etc.).

The minimum is $O(1)$; if we get lucky there may be no backtracking at all. The maximum is $O(2^n)$; we might get unlucky and every assignment except the last one needs backtracking. There are 2^N assignments of N binary variables.

Exercise 6.5.#SPCS

Suppose you have a state-space search problem defined by the usual stuff:

- a set of states s ;
- an initial state s_0 ;
- a set of actions A including the *NoOp* action that has no effect;
- a transition model $Result(s, a)$;
- a set of goal states G .

Unfortunately, you have no search algorithms! All you have is a CSP solver. How could you reformulate this as a CSP? You may assume that you are given the maximum number of

steps, T that any plan can have. Make sure that your formulation makes it easy to see what the plan is.

The straightforward solution is to have variables S_0, \dots, S_T for the state at each time step, with the domain of each being the set of states s ; and variables A_0, \dots, A_{T-1} for the action to be taken at each time step, with the domain of each being A . The constraints are

- $S_0 = s_0$;
- $S_T \in G$;
- For t from 0 to $T - 1$, $S_{t+1} = \text{Result}(S_t, A_t)$.

This would always find a solution of T steps if there is one; if there are shorter solutions it would pad them out with *NoOp* actions. Alternatively, you could specify the goal as $(S_0 \in G) \vee (S_1 \in G) \vee \dots \vee (S_T \in G)$, or you could run the CSP solver multiple times for successive values of T starting from 0.

EXERCISES 7

LOGICAL AGENTS

7.1 Knowledge-Based Agents

7.2 The Wumpus World

7.3 Logic

Exercise 7.3.#WUMT

Suppose the agent has progressed to the point shown in Figure 7.4(a), page 213, having perceived nothing in [1,1], a breeze in [2,1], and a stench in [1,2], and is now concerned with the contents of [1,3], [2,2], and [3,1]. Each of these can contain a pit, and at most one can contain a wumpus. Following the example of Figure 7.5, construct the set of possible worlds. (You should find 32 of them.) Mark the worlds in which the KB is true and those in which each of the following sentences is true:

$$\begin{aligned}\alpha_2 &= \text{"There is no pit in [2,2]."} \\ \alpha_3 &= \text{"There is a wumpus in [1,3]."}\end{aligned}$$

Hence show that $KB \models \alpha_2$ and $KB \models \alpha_3$.

To save space, we'll show the list of models as a table (Figure S7.1) rather than a collection of diagrams. There are eight possible combinations of pits in the three squares, and four possibilities for the wumpus location (including nowhere).

We can see that $KB \models \alpha_2$ because every line where KB is true also has α_2 true. Similarly for α_3 .

Exercise 7.3.#PRVM

Prove, or find a counterexample to, each of the following assertions, where α , β , and γ represent any logical sentence:

- a. If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \wedge \beta) \models \gamma$
- b. If $(\alpha \wedge \beta) \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).
- c. If $\alpha \models (\beta \vee \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).

- a. If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \wedge \beta) \models \gamma$.

Model	KB	α_2	α_3
$P_{1,3}$		<i>true</i>	
$P_{2,2}$		<i>true</i>	
$P_{3,1}$		<i>true</i>	
$P_{1,3}, P_{2,2}$			
$P_{2,2}, P_{3,1}$			
$P_{3,1}, P_{1,3}$		<i>true</i>	
$P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{1,3}$		<i>true</i>	<i>true</i>
$W_{1,3}, P_{1,3}$		<i>true</i>	<i>true</i>
$W_{1,3}, P_{2,2}$			<i>true</i>
$W_{1,3}, P_{3,1}$	<i>true</i>	<i>true</i>	<i>true</i>
$W_{1,3}, P_{1,3}, P_{2,2}$			<i>true</i>
$W_{1,3}, P_{2,2}, P_{3,1}$			<i>true</i>
$W_{1,3}, P_{3,1}, P_{1,3}$		<i>true</i>	<i>true</i>
$W_{1,3}, P_{1,3}, P_{3,1}, P_{2,2}$			<i>true</i>
$W_{3,1},$		<i>true</i>	
$W_{3,1}, P_{1,3}$		<i>true</i>	
$W_{3,1}, P_{2,2}$			
$W_{3,1}, P_{3,1}$		<i>true</i>	
$W_{3,1}, P_{1,3}, P_{2,2}$			
$W_{3,1}, P_{2,2}, P_{3,1}$			
$W_{3,1}, P_{3,1}, P_{1,3}$		<i>true</i>	
$W_{3,1}, P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{2,2}$		<i>true</i>	
$W_{2,2}, P_{1,3}$		<i>true</i>	
$W_{2,2}, P_{2,2}$			
$W_{2,2}, P_{3,1}$		<i>true</i>	
$W_{2,2}, P_{1,3}, P_{2,2}$			
$W_{2,2}, P_{2,2}, P_{3,1}$			
$W_{2,2}, P_{3,1}, P_{1,3}$		<i>true</i>	
$W_{2,2}, P_{1,3}, P_{3,1}, P_{2,2}$			

Figure S7.1 A truth table constructed for Ex. 7.2. Propositions not listed as true on a given line are assumed false, and only *true* entries are shown in the table.

True. This follows from monotonicity.

- b. If $(\alpha \wedge \beta) \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).
- False. Consider $\alpha \equiv A, \beta \equiv B, \gamma \equiv (A \wedge B)$.
- c. If $\alpha \models (\beta \vee \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).
- False. Consider $\beta \equiv A, \gamma \equiv \neg A$.

7.4 Propositional Logic: A Very Simple Logic

Exercise 7.4.#PENG

For each English sentence on the left, there is a corresponding logical sentence on the right, but *not necessarily the one across from it*. Work out which English sentence corresponds to which propositional logic sentence, and hence determine the meaning (in English) of each proposition symbol.

English

There is a Wumpus at (0, 1).

If the agent is at (0, 1) and there is a Wumpus at (0, 1),
then the agent is not alive.

The agent is at (0, 0) and there is no Wumpus at (0, 1).

The agent is at (0, 0) or (0, 1), but not both.

Propositional Logic

$(C \vee B) \wedge (\neg C \vee \neg B)$

$C \wedge \neg D$

$\neg A \vee \neg(B \wedge D)$

D

A = the agent is alive

B = the agent is at (0, 1)

C = the agent is at (0, 0)

D = there is a Wumpus at (0, 1).

Exercise 7.4.#UNIC

(Adapted from Barwise and Etchemendy (1993).) Given the following, can you prove that the unicorn is mythical? How about magical? Horned?

If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned.
The unicorn is magical if it is horned.

As human reasoners, we can see from the first two statements, that if it is mythical, then it is immortal; otherwise it is a mammal. So it must be either immortal or a mammal, and thus horned. That means it is also magical. However, we can't deduce anything about whether it is mythical. To provide a formal answer, we can enumerate the possible worlds ($2^5 = 32$ of them with 5 proposition symbols), mark those in which all the assertions are true, and see which conclusions hold in all of those. Or, we can let the machine do the work—in this case, the Lisp code for propositional reasoning:

```
> (setf kb (make-prop-kb))
#S(PROP-KB SENTENCE (AND))
> (tell kb "Mythical => Immortal")
T
> (tell kb "~Mythical => ~Immortal ^ Mammal")
T
> (tell kb "Immortal | Mammal => Horned")
T
> (tell kb "Horned => Magical")
```

```

function PL-TRUE?( $\check{s}$ ,  $\check{m}$ ) returns true or false
  if  $\check{s} = \text{True}$  then return true
  else if  $\check{s} = \text{False}$  then return false
  else if SYMBOL?( $\check{s}$ ) then return LOOKUP( $\check{s}$ ,  $\check{m}$ )
  else branch on the operator of  $\check{s}$ 
     $\neg$ : return not PL-TRUE?(ARG1( $\check{s}$ ),  $\check{m}$ )
     $\vee$ : return PL-TRUE?(ARG1( $\check{s}$ ),  $\check{m}$ ) or PL-TRUE?(ARG2( $\check{s}$ ),  $\check{m}$ )
     $\wedge$ : return PL-TRUE?(ARG1( $\check{s}$ ),  $\check{m}$ ) and PL-TRUE?(ARG2( $\check{s}$ ),  $\check{m}$ )
     $\Rightarrow$ : (not PL-TRUE?(ARG1( $\check{s}$ ),  $\check{m}$ )) or PL-TRUE?(ARG2( $\check{s}$ ),  $\check{m}$ )
     $\Leftrightarrow$ : PL-TRUE?(ARG1( $\check{s}$ ),  $\check{m}$ ) iff PL-TRUE?(ARG2( $\check{s}$ ),  $\check{m}$ )

```

Figure S7.2 Pseudocode for evaluating the truth of a sentence wrt a model.

```

T
> (ask kb "Mythical")
NIL
> (ask kb "\~{}Mythical")
NIL
> (ask kb "Magical")
T
> (ask kb "Horned")
T

```

Exercise 7.4.#TRUV

Consider the problem of deciding whether a propositional logic sentence is true in a given model.

- Write a recursive algorithm $\text{PL-TRUE?}(s, m)$ that returns true if and only if the sentence s is true in the model m (where m assigns a truth value for every symbol in s). The algorithm should run in time linear in the size of the sentence. (Alternatively, use a version of this function from the online code repository.)
- A *partial* model is one that specifies a truth value for a subset of the symbols. Give three examples of (non-tautologous, non-contradictory) sentences and associated (non-empty) partial models such that the sentence can be determined to be true or false in the partial model.
- Show that the truth value (if any) of a sentence in a partial model cannot be determined efficiently in general.
- Modify your PL-TRUE? algorithm so that it can sometimes judge truth from partial models, while retaining its recursive structure and linear run time. Give three examples of sentences whose truth in a partial model is *not* detected by your algorithm.
- Investigate whether the modified algorithm makes TT-ENTAILS? more efficient.

- See Figure S7.2. We assume the language has built-in Boolean operators **not**, **and**, **or**, **iff**.

b. There are many possible examples; here are three:

- $A \wedge P$ is false in the partial model $\{A = \text{false}\}$.
- $A \vee P$ is true in the partial model $\{A = \text{true}\}$.
- $(A \Rightarrow P) \iff A$ is false in the partial model $\{A = \text{false}\}$.

Note that tautologous and contradictory sentences have fixed truth values *regardless* of the model, so they can be given values in any partial model too.

- c.** A general algorithm for partial models must handle the empty partial model, with no assignments. In that case, the algorithm must determine validity and unsatisfiability, which are co-NP-complete and NP-complete respectively.
- d.** It helps if **and** and **or** evaluate their arguments in sequence, terminating on false or true arguments, respectively. In that case, the algorithm already has the desired properties: in the partial model where P is true and Q is unknown, $P \vee Q$ returns true, and $\neg P \wedge Q$ returns false. But the truth values of $Q \vee \neg Q$, $Q \vee \text{True}$, and $Q \wedge \neg Q$ are not detected.
- e.** Early termination in Boolean operators will provide a very substantial speedup. In most languages, the Boolean operators already have the desired property, so you would have to write special “dumb” versions and observe a slow-down.

Exercise 7.4.#ABCD

Consider a vocabulary with only four propositions, A , B , C , and D . How many models are there for the following sentences?

- a.** $B \vee C$.
- b.** $\neg A \vee \neg B \vee \neg C \vee \neg D$.
- c.** $(A \Rightarrow B) \wedge A \wedge \neg B \wedge C \wedge D$.
- d.** $(A \wedge B) \vee (C \wedge D)$.
- e.** $B \Rightarrow (A \wedge B)$.

These can be computed by counting the rows in a truth table that come out true, but each has some simple property that allows a short-cut:

- a.** Sentence is false only if B and C are false, which occurs in 4 cases for A and D , leaving 12.
- b.** Sentence is false only if A , B , C , and D are false, which occurs in 1 case, leaving 15.
- c.** The last four conjuncts specify a model in which the first conjunct is false, so 0.
- d.** 4 worlds satisfy $A \wedge B$, 4 satisfy $C \wedge D$, minus 1 for double-counting the model that satisfies both, leaving 7.
- e.** The sentence is true when B is false (8) and when B is true and A is true (4), so 12 in all.

Exercise 7.4.#BCON

We have defined four binary logical connectives.

- a. Are there any others that might be useful?
- b. How many binary connectives can there be?
- c. Why are some of them not very useful?

A binary logical connective is defined by a truth table with 4 rows. Each of the four rows may be true or false, so there are $2^4 = 16$ possible truth tables, and thus 16 possible connectives. Six of these are trivial ones that ignore one or both inputs; they correspond to *True*, *False*, P , Q , $\neg P$ and $\neg Q$. Four of them we have already studied: \wedge , \vee , \Rightarrow , \Leftrightarrow . The remaining six are potentially useful. One of them is reverse implication (\Leftarrow instead of \Rightarrow), and the other five are the negations of \wedge , \vee , \Leftrightarrow , \Rightarrow and \Leftarrow . The first three of these are sometimes called *nand*, *nor*, and *xor*.

7.5 Propositional Theorem Proving

Exercise 7.5.#LGEQ

Using a method of your choice, verify each of the equivalences in Figure 7.11 (page 223).

We use the truth table code in Lisp in the directory `logic/prop.lisp` to show each sentence is valid. We substitute P , Q , R for α, β, γ because of the lack of Greek letters in ASCII. To save space in this manual, we only show the first four truth tables:

```
> (truth-table "P ^ Q <=> Q ^ P")
-----
P   Q   P ^ Q   Q ^ P   (P ^ Q) <=> (Q ^ P)
-----
F   F   F       F           \\\(true\\)
T   F   F       F           T
F   T   F       F           T
T   T   T       T           T
-----
NIL

> (truth-table "P | Q <=> Q | P")
-----
P   Q   P | Q   Q | P   (P | Q) <=> (Q | P)
-----
F   F   F       F           T
T   F   T       T           T
F   T   T       T           T
T   T   T       T           T
-----
NIL

> (truth-table "P ^ (Q ^ R) <=> (P ^ Q) ^ R")
-----
P   Q   R   Q ^ R   P ^ (Q ^ R)   P ^ Q ^ R   (P ^ (Q ^ R)) <=> (P ^ Q ^ R)
-----
```

Exercises 7 Logical Agents

F	F	F	F	F	F	T
T	F	F	F	F	F	T
F	T	F	F	F	F	T
T	T	F	F	F	F	T
F	F	T	F	F	F	T
T	F	T	F	F	F	T
F	T	T	T	F	F	T
T	T	T	T	T	T	T

NIL

> (truth-table "P | (Q | R) <=> (P | Q) | R")

P	Q	R	Q R	P (Q R)	P Q R	(P (Q R)) <=> (P Q R)
F	F	F	F	F	F	T
T	F	F	F	T	T	T
F	T	F	T	T	T	T
T	T	F	T	T	T	T
F	F	T	T	T	T	T
T	F	T	T	T	T	T
F	T	T	T	T	T	T
T	T	T	T	T	T	T

NIL

For the remaining sentences, we just show that they are valid according to the validity function:

```
> (validity "~~P <=> P")
VALID
> (validity "P => Q <=> ~Q => ~P")
VALID
> (validity "P => Q <=> ~P | Q")
VALID
> (validity "(P <=> Q) <=> (P => Q) ^ (Q => P)")
VALID
> (validity "~(P ^ Q) <=> ~P | ~Q")
VALID
> (validity "~(P | Q) <=> ~P ^ ~Q")
VALID
> (validity "P ^ (Q | R) <=> (P ^ Q) | (P ^ R)")
VALID
> (validity "P | (Q ^ R) <=> (P | Q) ^ (P | R)")
VALID
```

Exercise 7.5.#VUNN

Decide whether each of the following sentences is valid, unsatisfiable, or neither. Verify your decisions using truth tables or the equivalence rules of Figure 7.11 (page 223).

- a. $\text{Smoke} \Rightarrow \text{Smoke}$
- b. $\text{Smoke} \Rightarrow \text{Fire}$
- c. $(\text{Smoke} \Rightarrow \text{Fire}) \Rightarrow (\neg \text{Smoke} \Rightarrow \neg \text{Fire})$

- d. $\text{Smoke} \vee \text{Fire} \vee \neg \text{Fire}$
- e. $((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire}) \Leftrightarrow ((\text{Smoke} \Rightarrow \text{Fire}) \vee (\text{Heat} \Rightarrow \text{Fire}))$
- f. $(\text{Smoke} \Rightarrow \text{Fire}) \Rightarrow ((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire})$
- g. $\text{Big} \vee \text{Dumb} \vee (\text{Big} \Rightarrow \text{Dumb})$

- a. Valid.
- b. Neither.
- c. Neither.
- d. Valid.
- e. Valid.
- f. Valid.
- g. Valid.

Exercise 7.5.#TFMO

Which of the following are correct?

- a. $\text{False} \models \text{True}$.
- b. $\text{True} \models \text{False}$.
- c. $(A \wedge B) \models (A \Leftrightarrow B)$.
- d. $A \Leftrightarrow B \models A \vee B$.
- e. $A \Leftrightarrow B \models \neg A \vee B$.
- f. $(A \wedge B) \Rightarrow C \models (A \Rightarrow C) \vee (B \Rightarrow C)$.
- g. $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$.
- h. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B)$.
- i. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$.
- j. $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable.
- k. $(A \Leftrightarrow B) \wedge (\neg A \vee B)$ is satisfiable.
- l. $(A \Leftrightarrow B) \Leftrightarrow C$ has the same number of models as $(A \Leftrightarrow B)$ for any fixed set of proposition symbols that includes A, B, C .

In all cases, the question can be resolved easily by referring to the definition of entailment.

- a. $\text{False} \models \text{True}$ is true because False has no models and hence entails every sentence AND because True is true in all models and hence is entailed by every sentence.
- b. $\text{True} \models \text{False}$ is false.
- c. $(A \wedge B) \models (A \Leftrightarrow B)$ is true because the left-hand side has exactly one model that is one of the two models of the right-hand side.
- d. $A \Leftrightarrow B \models A \vee B$ is false because one of the models of $A \Leftrightarrow B$ has both A and B false, which does not satisfy $A \vee B$.

- e. $A \Leftrightarrow B \models \neg A \vee B$ is true because the RHS is $A \Rightarrow B$, one of the conjuncts in the definition of $A \Leftrightarrow B$.
- f. $(A \wedge B) \Rightarrow C \models (A \Rightarrow C) \vee (B \Rightarrow C)$ is true because the RHS is false only when both disjuncts are false, i.e., when A and B are true and C is false, in which case the LHS is also false. This may seem counterintuitive, and would not hold if \Rightarrow is interpreted as “causes.”
- g. $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$ is true; proof by truth table enumeration, or by application of distributivity (Fig 7.11).
- h. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B)$ is true; removing a conjunct only allows more models.
- i. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$ is false; removing a disjunct allows fewer models.
- j. $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable; model has A and $\neg B$.
- k. $(A \Leftrightarrow B) \wedge (\neg A \vee B)$ is satisfiable; RHS is entailed by LHS so models are those of $A \Leftrightarrow B$.
- l. $(A \Leftrightarrow B) \Leftrightarrow C$ does have the same number of models as $(A \Leftrightarrow B)$; half the models of $(A \Leftrightarrow B)$ satisfy $(A \Leftrightarrow B) \Leftrightarrow C$, as do half the non-models, and there are the same numbers of models and non-models.

Exercise 7.5.#DEDU

Prove each of the following assertions:

- a. α is valid if and only if $\text{True} \models \alpha$.
- b. For any α , $\text{False} \models \alpha$.
- c. $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.
- d. $\alpha \equiv \beta$ if and only if the sentence $(\alpha \Leftrightarrow \beta)$ is valid.
- e. $\alpha \models \beta$ if and only if the sentence $(\alpha \wedge \neg \beta)$ is unsatisfiable.

Remember, $\alpha \models \beta$ iff in every model in which α is true, β is also true. Therefore,

- a. α is valid if and only if $\text{True} \models \alpha$.

Forward: If α is valid it is true in all models, hence it is true in all models of True .

Backward: if $\text{True} \models \alpha$ then α must be true in all models of True , i.e., in all models, hence α must be valid.

- b. For any α , $\text{False} \models \alpha$.

False doesn't hold in any model, so α trivially holds in every model of False .

- c. $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.

Both sides are equivalent to the assertion that there is no model in which α is true and β is false, i.e., no model in which $\alpha \Rightarrow \beta$ is false.

- d. $\alpha \equiv \beta$ if and only if the sentence $(\alpha \Leftrightarrow \beta)$ is valid.

Both sides are equivalent to the assertion that α and β have the same truth value in every model.

- e. $\alpha \models \beta$ if and only if the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable.

As in c, both sides are equivalent to the assertion that there is no model in which α is true and β is false.

Exercise 7.5.#LEPW

Any propositional logic sentence is logically equivalent to the assertion that each possible world in which it would be false is not the case. From this observation, prove that any sentence can be written in CNF.

Each possible world can be written as a conjunction of literals, e.g. $(A \wedge B \wedge \neg C)$. Asserting that a possible world is not the case can be written by negating that, e.g. $\neg(A \wedge B \wedge \neg C)$, which can be rewritten as $(\neg A \vee \neg B \vee C)$. This is the form of a clause; a conjunction of these clauses is a CNF sentence, and can list the negations of all the possible worlds that would make the sentence false.

Exercise 7.5.#CLIM

This exercise looks into the relationship between clauses and implication sentences.

- Show that the clause $(\neg P_1 \vee \dots \vee \neg P_m \vee Q)$ is logically equivalent to the implication sentence $(P_1 \wedge \dots \wedge P_m) \Rightarrow Q$.
- Show that every clause (regardless of the number of positive literals) can be written in the form $(P_1 \wedge \dots \wedge P_m) \Rightarrow (Q_1 \vee \dots \vee Q_n)$, where the P s and Q s are proposition symbols. A knowledge base consisting of such sentences is in **implicative normal form** or **Kowalski form** (Kowalski, 1979).
- Write down the full resolution rule for sentences in implicative normal form.

- $P \Rightarrow Q$ is equivalent to $\neg P \vee Q$ by implication elimination (Figure 7.11), and $\neg(P_1 \wedge \dots \wedge P_m)$ is equivalent to $(\neg P_1 \vee \dots \vee \neg P_m)$ by de Morgan's rule, so $(\neg P_1 \vee \dots \vee \neg P_m \vee Q)$ is equivalent to $(P_1 \wedge \dots \wedge P_m) \Rightarrow Q$.
- A clause can have positive and negative literals; let the negative literals have the form $\neg P_1, \dots, \neg P_m$ and let the positive literals have the form Q_1, \dots, Q_n , where the P_i s and Q_j s are symbols. Then the clause can be written as $(\neg P_1 \vee \dots \vee \neg P_m \vee Q_1 \vee \dots \vee Q_n)$. By the previous argument, with $Q = Q_1 \vee \dots \vee Q_n$, it is immediate that the clause is equivalent to

$$(P_1 \wedge \dots \wedge P_m) \Rightarrow Q_1 \vee \dots \vee Q_n .$$

- For atoms p_i, q_i, r_i, s_i where $p_j = q_k$:

$$\frac{\begin{array}{c} p_1 \wedge \dots \wedge p_j \dots \wedge p_{n_1} \Rightarrow r_1 \vee \dots \vee r_{n_2} \\ s_1 \wedge \dots \wedge s_{n_3} \Rightarrow q_1 \vee \dots \wedge q_k \dots \vee q_{n_4} \end{array}}{p_1 \wedge \dots \wedge p_{j-1} \wedge p_{j+1} \wedge p_{n_1} \wedge s_1 \wedge \dots \wedge s_{n_3} \Rightarrow r_1 \vee \dots \wedge r_{n_2} \vee q_1 \vee \dots \wedge q_{k-1} \wedge q_{k+1} \vee \dots \wedge q_{n_4}}$$

Exercise 7.5.#RADI

According to some political pundits, a person who is radical (R) is electable (E) if he/she is conservative (C), but otherwise is not electable.

- a. Which of the following are correct representations of this assertion?

- (i) $(R \wedge E) \iff C$
- (ii) $R \Rightarrow (E \iff C)$
- (iii) $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

- b. Which of the sentences in (a) can be expressed in Horn form?

- a. Correct representations of “a person who is radical is electable if he/she is conservative, but otherwise is not electable”:

(i) $(R \wedge E) \iff C$

No; this sentence asserts, among other things, that all conservatives are radical, which is not what was stated.

(ii) $R \Rightarrow (E \iff C)$

Yes, this says that if a person is a radical then they are electable if and only if they are conservative.

(iii) $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

No, this is equivalent to $\neg R \vee \neg C \vee E \vee \neg E$ which is a tautology, true under any assignment.

- b. Horn form:

- (i) Yes:

$$\begin{aligned} (R \wedge E) \iff C &\equiv ((R \wedge E) \Rightarrow C) \wedge (C \Rightarrow (R \wedge E)) \\ &\equiv ((R \wedge E) \Rightarrow C) \wedge (C \Rightarrow R) \wedge (C \Rightarrow E) \end{aligned}$$

- (ii) Yes:

$$\begin{aligned} R \Rightarrow (E \iff C) &\equiv R \Rightarrow ((E \Rightarrow C) \wedge (C \Rightarrow E)) \\ &\equiv \neg R \vee ((\neg E \vee C) \wedge (\neg C \vee E)) \\ &\equiv (\neg R \vee \neg E \vee C) \wedge (\neg R \vee \neg C \vee E) \end{aligned}$$

- (iii) Yes, e.g., *True* \Rightarrow *True*.

Exercise 7.5.#TCNF

A propositional 2-CNF expression is a conjunction of clauses, each containing *exactly 2* literals, e.g.,

$$(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee D) \wedge (\neg C \vee G) \wedge (\neg D \vee G).$$

- a. Prove using resolution that the above sentence entails G .

- b. Two clauses are *semantically distinct* if they are not logically equivalent. How many semantically distinct 2-CNF clauses can be constructed from n proposition symbols?
 - c. Using your answer to (b), prove that propositional resolution always terminates in time polynomial in n given a 2-CNF sentence containing no more than n distinct symbols.
 - d. Explain why your argument in (c) does not apply to 3-CNF.
-
- a. The negated goal is $\neg G$. Resolve with the last two clauses to produce $\neg C$ and $\neg D$. Resolve with the second and third clauses to produce $\neg A$ and $\neg B$. Resolve these successively against the first clause to produce the empty clause.
 - b. This can be answered with or without *True* and *False* symbols; we'll omit them for simplicity. First, each 2-CNF clause has two places to put literals. There are $2n$ distinct literals, so there are $(2n)^2$ syntactically distinct clauses. Now, many of these clauses are semantically identical. Let us handle them in groups. There are $C(2n, 2) = (2n)(2n - 1)/2 = 2n^2 - n$ clauses with two different literals, if we ignore ordering. All these clauses are semantically distinct except those that are equivalent to *True* (e.g., $(A \vee \neg A)$), of which there are n , so that makes $2n^2 - 2n + 1$ clauses with distinct literals. There are $2n$ clauses with repeated literals, all distinct. So there are $2n^2 + 1$ distinct clauses in all.
 - c. Resolving two 2-CNF clauses cannot increase the clause size; therefore, resolution can generate only $O(n^2)$ distinct clauses before it must terminate.
 - d. First, note that the number of 3-CNF clauses is $O(n^3)$, so we cannot argue for nonpolynomial complexity on the basis of the number of different clauses! The key observation is that resolving two 3-CNF clauses can *increase* the clause size to 4, and so on, so clause size can grow to $O(n)$, giving $O(2^n)$ possible clauses.

Exercise 7.5.#PRPC

Prove each of the following assertions:

- a. Every pair of propositional clauses either has no resolvents, or all their resolvents are logically equivalent.
 - b. There is no clause that, when resolved with itself, yields (after factoring) the clause $(\neg P \vee \neg Q)$.
 - c. If a propositional clause C can be resolved with a copy of itself, it must be logically equivalent to *True*.
-
- a. If the clauses have no complementary literals, they have no resolvents. If they have one pair of complementary literals, they have one resolvent, which is logically equivalent to itself. If they have more than one pair, then one pair resolves away and the other pair appears in the resolvent as $(\dots A \vee \neg A \dots)$ which renders the resolvent logically equivalent to *True*.
 - b. The original clause must include both $\neg P$ and $\neg Q$, for them to appear in the resolvent.

There must be a third literal that was resolved away. It must be either P or Q , since any other literal would not be complementary to any of the literals in the clause. If it were P , then one copy of P would be in the resolvent; the same goes for Q .

- c. Such a clause must contain a literal and its complement, and their disjunction is a tautology.

Exercise 7.5.#FOOD

Consider the following sentence:

$$[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \Rightarrow [(Food \wedge Drinks) \Rightarrow Party].$$

- a. Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.
- b. Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).
- c. Prove your answer to (a) using resolution.

- a. A simple truth table has eight rows, and shows that the sentence is true for all models and hence valid.

- b. For the left-hand side we have:

$$\begin{aligned} &(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party) \\ &(\neg Food \vee Party) \vee (\neg Drinks \vee Party) \\ &(\neg Food \vee Party \vee \neg Drinks \vee Party) \\ &(\neg Food \vee \neg Drinks \vee Party) \end{aligned}$$

and for the right-hand side we have

$$\begin{aligned} &(Food \wedge Drinks) \Rightarrow Party \\ &\neg(Food \wedge Drinks) \vee Party \\ &(\neg Food \vee \neg Drinks) \vee Party \\ &(\neg Food \vee \neg Drinks \vee Party) \end{aligned}$$

The two sides are identical in CNF, and hence the original sentence is of the form $P \Rightarrow P$, which is valid for any P .

- c. To prove that a sentence is valid, prove that its negation is unsatisfiable. I.e., negate it, convert to CNF, use resolution to prove a contradiction. We can use the above CNF result for the LHS.

$$\begin{aligned} &\neg[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \Rightarrow [(Food \wedge Drinks) \Rightarrow Party] \\ &[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \wedge \neg[(Food \wedge Drinks) \Rightarrow Party] \\ &(\neg Food \vee \neg Drinks \vee Party) \wedge Food \wedge Drinks \wedge \neg Party \end{aligned}$$

Each of the three unit clauses resolves in turn against the first clause, leaving an empty clause.

Exercise 7.5.#DNFX

A sentence is in **disjunctive normal form** (DNF) if it is the disjunction of conjunctions of literals. For example, the sentence $(A \wedge B \wedge \neg C) \vee (\neg A \wedge C) \vee (B \wedge \neg C)$ is in DNF.

- Any propositional logic sentence is logically equivalent to the assertion that some possible world in which it would be true is in fact the case. From this observation, prove that any sentence can be written in DNF.
- Construct an algorithm that converts any sentence in propositional logic into DNF. (*Hint:* The algorithm is similar to the algorithm for conversion to CNF given in Section 7.5.2.)
- Construct a simple algorithm that takes as input a sentence in DNF and returns a satisfying assignment if one exists, or reports that no satisfying assignment exists.
- Apply the algorithms in (b) and (c) to the following set of sentences:

$$\begin{aligned} A &\Rightarrow B \\ B &\Rightarrow C \\ C &\Rightarrow \neg A . \end{aligned}$$

- Since the algorithm in (b) is very similar to the algorithm for conversion to CNF, and since the algorithm in (c) is much simpler than any algorithm for solving a set of sentences in CNF, why is this technique not used in automated reasoning?
- Each possible world can be expressed as the conjunction of all the literals that hold in the model. The sentence is then equivalent to the disjunction of all these conjunctions, i.e., a DNF expression.
- A trivial conversion algorithm would enumerate all possible models and include terms corresponding to those in which the sentence is true; but this is necessarily exponential-time. We can convert to DNF using the same algorithm as for CNF except that we distribute \wedge over \vee at the end instead of the other way round.
- A DNF expression is satisfiable if it contains at least one term that has no contradictory literals. This can be checked in linear time, or even during the conversion process. Any completion of that term, filling in missing literals, is a model.
- The first steps give

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg A) .$$

Converting to DNF means taking one literal from each clause, in all possible ways, to generate the terms (8 in all). Choosing each literal corresponds to choosing the truth value of each variable, so the process is very like enumerating all possible models. Here, the first term is $(\neg A \wedge \neg B \wedge \neg C)$, which is clearly satisfiable.

- The problem is that the final step typically results in DNF expressions of exponential size, so we require both exponential time AND exponential space.

Exercise 7.5.#CNFZ

- a.** A certain procedure to convert a sentence to CNF contains four steps (1–4 below); each step is based on a logical equivalence. For each step, which of the stated equivalences are valid?
- (i) Step 1: drop biconditionals
 - a) $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$
 - b) $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \vee (\beta \Rightarrow \alpha))$
 - c) $(\alpha \Leftrightarrow \beta) \equiv (\alpha \wedge \beta)$
 - (ii) Step 2: drop implications
 - a) $(\alpha \Rightarrow \beta) \equiv (\alpha \vee \neg\beta)$
 - b) $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$
 - c) $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \wedge \beta)$
 - (iii) Step 3: move not inwards
 - a) $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$
 - b) $\neg(\alpha \vee \beta) \equiv (\neg\alpha \vee \neg\beta)$
 - c) $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$
 - (iv) Step 4: move “or” inwards and “and” outwards
 - a) $(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta \vee \gamma)$
 - b) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$
 - c) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$
- b.** The Convert-to-CNF-ish procedure applies the *first* equivalence (the one labeled “a”) from each of the four steps in part (a). Show the transformed sentence generated by Convert-to-CNF-ish at each stage, when applied to the input sentence $A \Leftrightarrow (C \vee D)$.
- c.** Is the final output of Convert-to-CNF-ish equivalent to the input sentence in part (b)? If not, give a possible world where the input and output sentences have different values.

- a.** The valid equivalences are as follows:

- (i) Step 1: (a) is valid.
- (ii) Step 2: (b) is valid.
- (iii) Step 3: (a) and (c) are valid.
- (iv) Step 4: (b) is valid.

- b.** The sequence of sentences is as follows: $A \Leftrightarrow (C \vee D)$

$$\begin{aligned} & (A \Rightarrow (C \vee D)) \wedge ((C \vee D) \Rightarrow A) \\ & (A \vee \neg(C \vee D)) \wedge ((C \vee D) \vee \neg A) \\ & (A \vee (\neg C \wedge \neg D)) \wedge ((C \vee D) \vee \neg A) \\ & (A \vee \neg C \vee \neg D) \wedge (C \vee D \vee \neg A) \end{aligned}$$

- c. No. A counterexample is any model where the two sentences have different truth values. The first clause in the final sentence says $(C \wedge D) \Rightarrow A$ rather than $(C \vee D) \Rightarrow A$. So counterexamples are $\{A = \text{false}, C = \text{true}, D = \text{false}\}$ and $\{A = \text{false}, C = \text{false}, D = \text{true}\}$.

7.6 Effective Propositional Model Checking

Exercise 7.6.#SATC

This question considers representing satisfiability (SAT) problems as CSPs.

- a. Draw the constraint graph corresponding to the SAT problem

$$(\neg X_1 \vee X_2) \wedge (\neg X_2 \vee X_3) \wedge \dots \wedge (\neg X_{n-1} \vee X_n)$$

for the particular case $n = 5$.

- b. How many solutions are there for this general SAT problem as a function of n ?
- c. Suppose we apply BACKTRACKING-SEARCH (page 192) to find *all* solutions to a SAT CSP of the type given in (a). (To find *all* solutions to a CSP, we simply modify the basic algorithm so it continues searching after each solution is found.) Assume that variables are ordered X_1, \dots, X_n and *false* is ordered before *true*. How much time will the algorithm take to terminate? (Write an $O(\cdot)$ expression as a function of n .)
- d. We know that SAT problems in Horn form can be solved in linear time by forward chaining (unit propagation). We also know that every tree-structured binary CSP with discrete, finite domains can be solved in time linear in the number of variables (Section 6.5). Are these two facts connected? Discuss.

- a. The graph is simply a connected chain of 5 nodes, one per variable.
- b. $n + 1$ solutions. Once any X_i is true, all subsequent X_j 's must be true. Hence the solutions are i falses followed by $n - i$ trues, for $i = 0, \dots, n$.
- c. The complexity is $O(n^2)$. This is somewhat tricky. Consider what part of the complete binary tree is explored by the search. The algorithm must follow all solution sequences, which themselves cover a quadratic-sized portion of the tree. Failing branches are all those trying a *false* after the preceding variable is assigned *true*. Such conflicts are detected immediately, so they do not change the quadratic cost.
- d. These facts are not obviously connected. Horn-form logical inference problems need not have tree-structured constraint graphs; the linear complexity comes from the nature of the constraint (implication) not the structure of the problem.

Exercise 7.6.#ESAT

Explain why every nonempty propositional clause, by itself, is satisfiable. Prove rigorously that every set of five 3-SAT clauses is satisfiable, provided that each clause mentions exactly three distinct variables. What is the smallest set of such clauses that is unsatisfiable? Construct such a set.

A clause is a disjunction of literals, and its models are the *union* of the sets of models of each literal; and each literal satisfies half the possible models. (Note that *False* is unsatisfiable, but it is really another name for the empty clause.) A 3-SAT clause with three distinct variables rules out exactly 1/8 of all possible models, so five clauses can rule out no more than 5/8 of the models. Eight clauses are needed to rule out all models. Suppose we have variables A, B, C . There are eight models, and we write one clause to rule out each model. For example, the model $\{A = \text{false}, B = \text{false}, C = \text{false}\}$ is ruled out by the clause $(\neg A \vee \neg B \vee \neg C)$.

Exercise 7.6.#UNCL

Explain why the following set of clauses is unsatisfiable *without* using truth tables:

$$\begin{array}{ll} A \vee B \vee C & \neg A \vee B \vee C \\ A \vee B \vee \neg C & \neg A \vee B \vee \neg C \\ A \vee \neg B \vee C & \neg A \vee \neg B \vee C \\ A \vee \neg B \vee \neg C & \neg A \vee \neg B \vee \neg C \end{array}$$

Since each clause is false in exactly one model, and all clauses are different and therefore false in different models, all 8 models are ruled out hence the conjunction is unsatisfiable. There are other elegant proofs: for example, suppose there is a satisfying model $[A = a, B = b, C = c]$; that falsifies the clause $\neg a \vee \neg b \vee \neg c$, which must be a member of the set, hence no such model can exist. Some more elaborate proofs based on splitting on variable values amount to truth tables in disguise. One approach definitely won't work: claiming that, say, $A \vee B \vee C$ and $\neg A \vee \neg B \vee \neg C$ are contradictory; in fact, there are 6 models where both are true!

Exercise 7.6.#CCLX

Convert the following set of sentences to clausal form.

- S1: $A \Leftrightarrow (B \vee E)$.
- S2: $E \Rightarrow D$.
- S3: $C \wedge F \Rightarrow \neg B$.
- S4: $E \Rightarrow B$.
- S5: $B \Rightarrow F$.
- S6: $B \Rightarrow C$

Give a trace of the execution of DPLL on the conjunction of these clauses.

The CNF representations are as follows:

- S1: $(\neg A \vee B \vee E) \wedge (\neg B \vee A) \wedge (\neg E \vee A)$.
- S2: $(\neg E \vee D)$.
- S3: $(\neg C \vee \neg F \vee \neg B)$.
- S4: $(\neg E \vee B)$.
- S5: $(\neg B \vee F)$.
- S6: $(\neg B \vee C)$.

We omit the DPLL trace, which is easy to obtain from the version in the code repository.

Exercise 7.6.#RESO

Use resolution to prove the sentence $\neg A \wedge \neg B$ from the clauses in Exercise 7.CCLX.

To prove the conjunction, it suffices to prove each literal separately. To prove $\neg B$, add the negated goal S7: B .

- Resolve S7 with S5, giving S8: F .
- Resolve S7 with S6, giving S9: C .
- Resolve S8 with S3, giving S10: $(\neg C \vee \neg B)$.
- Resolve S9 with S10, giving S11: $\neg B$.
- Resolve S7 with S11 giving the empty clause.

To prove $\neg A$, add the negated goal S7: A .

- Resolve S7 with the first clause of S1, giving S8: $(B \vee E)$.
- Resolve S8 with S4, giving S9: B .
- Proceed as above to derive the empty clause.

Exercise 7.6.#MINE

Minesweeper, the well-known computer game, is closely related to the wumpus world. A minesweeper world is a rectangular grid of N squares with M invisible mines scattered among them. Any square may be probed by the agent; instant death follows if a mine is probed. Minesweeper indicates the presence of mines by revealing, in each probed square, the *number* of mines that are directly or diagonally adjacent. The goal is to probe every unmined square.

- a. Let $X_{i,j}$ be true iff square $[i, j]$ contains a mine. Write down the assertion that exactly two mines are adjacent to $[1,1]$ as a sentence involving some logical combination of $X_{i,j}$ propositions.
- b. Generalize your assertion from (a) by explaining how to construct a CNF sentence asserting that k of n neighbors contain mines.
- c. Explain precisely how an agent can use DPLL to prove that a given square does (or does not) contain a mine, ignoring the global constraint that there are exactly M mines in all.
- d. Suppose that the global constraint is constructed from your method from part (b). How does the number of clauses depend on M and N ? Suggest a way to modify DPLL so that the global constraint does not need to be represented explicitly.
- e. Are any conclusions derived by the method in part (c) invalidated when the global constraint is taken into account?
- f. Give examples of configurations of probe values that induce *long-range dependencies* such that the contents of a given unprobed square would give information about the contents of a far-distant square. (*Hint:* consider an $N \times 1$ board.)

- a. This is a disjunction with 28 disjuncts, each one saying that two of the neighbors are true and the others are false. The first disjunct is

$$X_{2,2} \wedge X_{1,2} \wedge \neg X_{0,2} \wedge \neg X_{0,1} \wedge \neg X_{2,1} \wedge \neg X_{0,0} \wedge \neg X_{1,0} \wedge \neg X_{2,0}$$

The other 27 disjuncts each select two different $X_{i,j}$ to be true.

- b. There will be $\binom{n}{k}$ disjuncts, each saying that k of the n symbols are true and the others false.
 - c. For each of the cells that have been probed, take the resulting number n revealed by the game and construct a sentence with $\binom{n}{8}$ disjuncts. Conjoin all the sentences together. Then use DPLL to answer the question of whether this sentence entails $X_{i,j}$ for the particular i, j pair you are interested in.
 - d. To encode the global constraint that there are M mines altogether, we can construct a disjunct with $\binom{M}{N}$ disjuncts, each of size N . Remember, $\binom{M}{N=M!/(M-N)!}$. So for a Minesweeper game with 100 cells and 20 mines, this will be more than 10^{39} , and thus cannot be represented in any computer. However, we can represent the global constraint within the DPLL algorithm itself. We add the parameter min and max to the DPLL function; these indicate the minimum and maximum number of unassigned symbols that must be true in the model. For an unconstrained problem the values 0 and N will be used for these parameters. For a mineseeper problem the value M will be used for both min and max . Within DPLL, we fail (return false) immediately if min is less than the number of remaining symbols, or if max is less than 0. For each recursive call to DPLL, we update min and max by subtracting one when we assign a true value to a symbol.
 - e. No conclusions are invalidated by adding this capability to DPLL and encoding the global constraint using it.
 - f. Consider this string of alternating 1's and unprobed cells (indicated by a dash):

| = | 1 | = | 1 | = | 1 | = | 1 | = | 1 | = | 1 | = | 1 | = | 1 | = |

There are two possible models: either there are mines under every even-numbered dash, or under every odd-numbered dash. Making a probe at either end will determine whether cells at the far end are empty or contain mines.

Exercise 7.6.#KNOW

How long does it take to prove $KB \models \alpha$ using DPLL when α is a literal *already contained in KB* ? Explain.

It will take time proportional to the number of pure symbols plus the number of unit clauses. We assume that $KB \Rightarrow \alpha$ is false, and prove a contradiction. $\neg(KB \Rightarrow \alpha)$ is equivalent to $KB \wedge \neg\alpha$. From this sentence the algorithm will first eliminate all the pure symbols, then it will work on unit clauses until it chooses either α or $\neg\alpha$ (both of which are unit clauses); at that point it will immediately recognize that either choice (true or false) for α leads to failure, which means that the original non-negated assertion α is entailed.

Exercise 7.6.#DPLL

Trace the behavior of DPLL on the knowledge base in Figure 7.16 when trying to prove Q , and compare this behavior with that of the forward-chaining algorithm.

We omit the DPLL trace, which is easy to obtain from the version in the code repository. The behavior is very similar: the unit-clause rule in DPLL ensures that all known atoms are propagated to other clauses.

Exercise 7.6.#DCSP

The DPLL satisfiability algorithm is a backtracking search algorithm with 3 heuristic improvements: PURE-SYMBOLS, UNIT-CLAUSES, and EARLY TERMINATION. This question connects these improvements to more general CSP techniques used with backtracking search.

- a. In the following CNF expression, which symbols are pure and what value does the PURE-SYMBOLS heuristic assign to those symbols?

$$(C \vee D) \wedge (C \vee \neg A) \wedge (\neg D \vee A) \wedge (\neg B \vee A)$$

- b. Which of the following CSP techniques are equivalent to the PURE-SYMBOLS and UNIT-CLAUSES heuristics when applied to SAT for CNF sentences?

MINIMUM-REMAINING-VALUES

FORWARD-CHECKING

LEAST-CONSTRAINING-VALUE

BACKTRACKING

No equivalent CSP technique

- c. DPLL performs early termination in two steps: SUCCESS-DETECTION and FAILURE-DETECTION. First, SUCCESS-DETECTION checks to see if *all* clauses evaluate to true. If so, the sentence is satisfiable and any unassigned propositional symbols can be assigned arbitrarily. Next, FAILURE-DETECTION checks if *any* clause evaluates to false. In this case the sentence is unsatisfiable and this branch of the search can be pruned. Which, if any, of these forms of early termination does the general CSP backtracking algorithm use?

- a. The pure symbols are B and C, and the values assigned are false and true, respectively.
b. The PURE-SYMBOLS heuristic is equivalent to LEAST-CONSTRAINING-VALUE. Assigning a value to a pure literal can only cause clauses to become true, so it will impose 0 additional constraints on the remaining literals. Notice that a pure symbol (except in a unit clause) can still take on either value, so this is not an MRV choice.

The UNIT-CLauses heuristic is equivalent to MINIMUM-REMAINING-VALUES. A variable in a unit clause can only be assigned to a single value in a satisfying assignment,

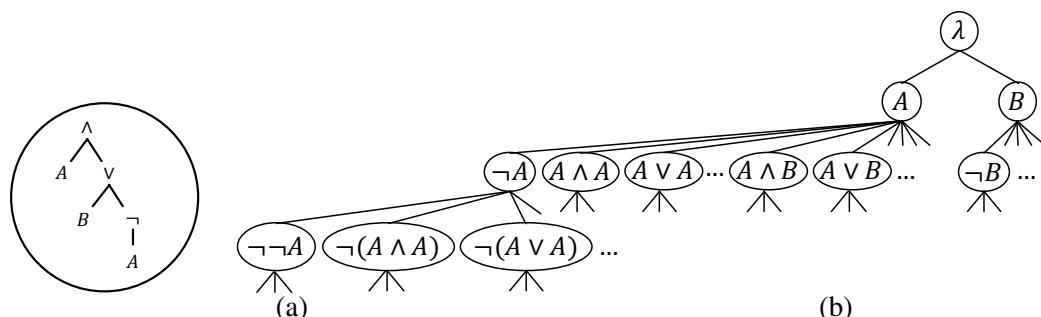
otherwise that clause would be false. Thus, there is only 1 remaining value for such literals.

- c. In the backtracking CSP algorithm in Chapter 6, early termination is triggered whenever the domain for a variable is empty. This is exactly the case when a clause has been reduced to false, so FAILURE-DETECTION is an application of this technique to SAT. Early detection of success is possible in SAT because if *any* literal in a clause is true, the clause is satisfied, and no further assignments to the other variables can unsatisfy it. This is *not* true in general CSPs: a partial assignment may not violate any constraints, but it is never (in general) certain to become a solution because further assignments may violate some constraints. Hence early success detection is not possible in general CSPs.

Exercise 7.6.#FCNF

Is a randomly generated 4-CNF sentence with n symbols and m clauses more or less likely to be solvable than a randomly generated 3-CNF sentence with n symbols and m clauses? Explain.

It is more likely to be solvable: adding literals to disjunctive clauses makes them easier to satisfy.



(a) The sentence $A \wedge (B \vee \neg A)$ drawn as a syntax tree with 5 edges. (b) Part of the PQ search space for sentences with symbols A and B .

Exercise 7.6.#LOGS

In this question we consider the problem of searching for the smallest propositional logic sentence ϕ that satisfies some condition $G(\phi)$. (E.g., “find the smallest unsatisfiable sentence containing two distinct symbols.”) The proposition symbols are given as part of the problem. The size of a sentence is defined as the sum of the sizes of its logical connectives, where \neg has size 1 and the other connectives have size 2. It is helpful to think of the sentence as a syntax tree with proposition symbols at the leaves; then the size is the number of edges in the tree. Figure 7.3(a) shows an example.

- a. Let the RS problem formulation to explore the search space of sentences be as follows:

- There is a dummy start state λ , which is an empty sentence that never satisfies G . The actions applicable in this state simply replace λ by one of the proposition symbols.
- For all other states, the actions take any occurrence of a proposition symbol (call it p) in the tree and replace it with a new subtree: either $\neg q$ for any symbol q , or $(r * s)$ where r and s are any symbols and “ $*$ ” is any binary connective.

Does the RS formulation generate all and only the syntactically valid sentences?

- b.** The PQ formulation replaces the second clause from the RS formulation as follows:

- For all other states, the actions take any occurrence of a proposition symbol (call it p) in the tree and replace it with a new subtree: either $\neg p$, or $p * q$ where q is any symbol and “ $*$ ” is any binary connective.

Part of the search space generated by the PQ formulation is shown in Figure ??(b).

Does this formulation generate all and only the syntactically valid sentences?

- c.** Which of the two formulations is best? Explain.
- d.** Define an appropriate step cost function for the PQ formulation.
- e.** Assuming there are n symbols, give a $O()$ -expression for the branching factor at depth d using PQ.
- f.** Using your $O()$ answer for the branching factor, give a $O()$ -expression for the number of nodes at depth d of the PQ search tree.
- g.** We will say that G is a *semantic* condition if $G(\phi)$ depends only on the *meaning* of ϕ , in the following sense: if two sentences ϕ and ψ are logically equivalent, then $G(\phi) = G(\psi)$. Furthermore, let the reduced-PQ formulation be identical to the PQ formulation, except that it uses only the connectives \neg , \wedge , and \vee rather than the full set. Is it the case that whenever the PQ search space contains a solution for a semantic G then so does the reduced-PQ search space? Explain.
- h.** Suppose we are running a uniform-cost graph search, which maintains the property that $g(n)$ for every node in the frontier is the optimal cost to reach n from the root. In a standard graph search, we discard a newly generated successor state without adding it to the frontier if and only if the *identical* state is already in the frontier set or explored set. Assuming we have a semantic condition G , when is it possible to discard a newly generated successor state?
 - (i) Sentences that are identical to a sentence already in the frontier or explored set
 - (ii) Sentences that logically entail a sentence already in the frontier or explored set
 - (iii) Sentences that are logically equivalent to a sentence already in the frontier or explored set
 - (iv) Sentences that are logically entailed by a sentence already in the frontier or explored set
- i.** Now we will apply this general search machinery for sentences to supervised machine learning: the problem of finding concise hypotheses that are consistent with a set of labeled examples and predict labels for unlabeled examples.

- Let X_1, \dots, X_n be the Boolean input variables and Y be the “output” Boolean property we are trying to predict. A hypothesis H asserts that Y is some function of the inputs, i.e., H is a sentence $Y \Leftrightarrow \phi$, where ϕ is a propositional formula containing only X_1, \dots, X_n . For example, let H_0 be the sentence $Y \Leftrightarrow (X_1 \vee X_2)$; then H_0 asserts that the output is true exactly when either of the inputs X_1, X_2 is true.
- A labeled example E gives the output value for particular values of the inputs; i.e., it is a sentence $\psi \Rightarrow \chi$, where ψ is a conjunction of literals, one for each input symbol, and χ is a literal containing the output symbol. Thus if there are just two inputs, three examples might be

$$\begin{aligned} E_1 : \quad & X_1 \wedge X_2 \Rightarrow Y \\ E_2 : \quad & X_1 \wedge \neg X_2 \Rightarrow Y \\ E_3 : \quad & \neg X_1 \wedge \neg X_2 \Rightarrow \neg Y \end{aligned}$$

Show, in general, that every labeled example sentence is *false in exactly one model* for the symbols X_1, \dots, X_n, Y .

- j. Show, by completing the following truth table and marking the relevant rows, that the hypothesis H_0 given above logically entails all three examples.

X_1	X_2	Y	E_1	E_2	E_3	H_0
F	F	F				
F	F	T				
F	T	F				
F	T	T				
T	F	F				
T	F	T				
T	T	F				
T	T	T				

- k. For supervised machine learning, then, we want to find an H , by searching among formulas ϕ , that entails all the examples. Suppose you have available a solver $SAT(\alpha)$ that returns *true* if α is satisfiable and *false* otherwise. Which of the following expressions correctly implements $G(\phi)$?

- $SAT(H \wedge E_1 \wedge E_2 \wedge E_3)$
- $SAT(H \wedge \neg(E_1 \wedge E_2 \wedge E_3))$
- $\neg SAT(H \wedge \neg(E_1 \wedge E_2 \wedge E_3))$
- $\neg SAT(H \wedge E_1 \wedge E_2 \wedge E_3)$

- l. Now suppose we have an unlabeled test example U_4 described only by the ψ -part:

$$U_4 : \quad \neg X_1 \wedge X_2$$

Now we want to predict the label for U_4 given a hypothesis H . Which of the following does this correctly?

- $SAT(H \wedge U_4 \wedge Y)$

- $\neg SAT(H \vee (U_4 \rightarrow Y))$
- $\neg SAT(H \wedge (U_4 \rightarrow Y))$
- $\neg SAT(H \wedge \neg(U_4 \rightarrow Y))$

- a. Yes. Proof by induction on the size of the sentence. Base case is trivial. For the inductive case, consider the tree representation of a sentence. The subtree expression(s) of this tree are smaller, and hence can be generated by this process; and the root operator can be generated, with two symbols that can be replaced by any expression generated by this process.
- b. Yes, by the same argument.
- c. PQ is more efficient because it has a smaller branching factor and eliminates redundant copies of sentences. For any sentence with an occurrence of p in some location, there is another sentence identical except that the occurrence of p is replaced by an occurrence of q . (Essentially this is the inductive hypothesis in the proof.) Hence there is no need to replace p by other symbols.
- d. We want the step cost to reflect the desirability (i.e., size) of solution expressions; so cost 1 for introducing a negation, 2 for introducing a binary connective.
- e. Each step adds no more than 1 symbol occurrence; so at depth d there are up to d symbols. There is one way to negate a symbol and n ways to add each of four binary connectives, so the branching factor is $O(4nd)$.
- f. The off-the-shelf answer of b^d doesn't work because b is not constant. The number of nodes is the product of the branching factors, so $O((4n)^d \cdot d!)$.
- g. Every propositional logic sentence has an equivalent sentence in CNF, which uses only \neg , \wedge , and \vee .
- h. Options (i) and (iii) are correct.
- i. An implication is false iff the LHS is true and the RHS is false. Those conditions set the values of all the symbols because the LHS is a conjunction of n literals, one per input symbol.
- j. The completed truth table is as follows; the relevant rows are marked by arrows.

X_1	X_2	Y	E_1	E_2	E_3	H_0
F	F	F	T	T	T	T \leftarrow
F	F	T	T	T	F	F
F	T	F	T	T	T	F
F	T	T	T	T	T	T \leftarrow
T	F	F	T	F	T	F
T	F	T	T	T	T	T \leftarrow
T	T	F	F	T	T	F
T	T	T	T	T	T	T \leftarrow

- k. Option (iii) only.
- l. Option (iv) only.

7.7 Agents Based on Propositional Logic

Exercise 7.7.#SSAL

Write a successor-state axiom for the *Locked* predicate, which applies to doors, assuming the only actions available are *Lock* and *Unlock*.

$$\text{Locked}^{t+1} \Leftrightarrow [\text{Lock}^t \vee (\text{Locked}^t \wedge \neg \text{Unlock}^t)].$$

Exercise 7.7.#OPTW

Discuss what is meant by *optimal* behavior in the wumpus world. Show that the HYBRID-WUMPUS-AGENT is not optimal, and suggest ways to improve it.

Optimal behavior for the Wumpus world is defined by the task environment in Section 7.2: optimal behavior maximizes the expected performance measure. We need to take an expectation because we are unsure of the initial state of the world.

The hybrid agent is not optimal as, for example, when it decides to risk a square with a pit it doesn't necessarily pick a square with minimal probability of having a pit: a breezy square that has three unvisited neighbours is safer to explore from than one which has two. One improvement, therefore, would explore from breezy squares with the most unexplored neighbours when there are no non-breezy square to explore from.

Exercise 7.7.#TSWA

Suppose an agent inhabits a world with two states, S and $\neg S$, and can do exactly one of two actions, a and b . Action a does nothing and action b flips from one state to the other. Let S^t be the proposition that the agent is in state S at time t , and let a^t be the proposition that the agent does action a at time t (similarly for b^t).

- a. Write a successor-state axiom for S^{t+1} .
- b. Convert the sentence in (a) into CNF.
- c. Show a resolution refutation proof that if the agent is in $\neg S$ at time t and does a , it will still be in $\neg S$ at time $t + 1$.

a. $S^{t+1} \Leftrightarrow [(S^t \wedge a^t) \vee (\neg S^t \wedge b^t)].$

- b. Because the agent can do exactly one action, we know that $b^t \equiv \neg a^t$ so we replace b^t throughout. We obtain four clauses:

- 1: $(\neg S^{t+1} \vee S^t \vee \neg a^t)$
- 2: $(\neg S^{t+1} \vee \neg S^t \vee a^t)$
- 3: $(S^{t+1} \vee \neg S^t \vee \neg a^t)$
- 4: $(S^{t+1} \vee S^t \vee a^t)$

- c. The goal is $(\neg S^t \wedge a^t) \Rightarrow \neg S^{t+1}$. Negated, this becomes three clauses: 5: $\neg S^t$; 6: a^t ; 7: S^{t+1} . Resolving 5, 6, 7 against 1, we obtain the empty clause.

Exercise 7.7.#SSAX

Section 7.7.1 provides some of the successor-state axioms required for the wumpus world. Write down axioms for all remaining fluent symbols.

The remaining fluents are the orientation fluents (*FacingEast* etc.) and *WumpusAlive*. The successor-state axioms are as follows:

$$\begin{aligned} FacingEast^{t+1} &\Leftrightarrow (FacingEast^t \wedge \neg(TurnLeft^t \vee TurnRight^t)) \\ &\quad \vee (FacingNorth^t \wedge TurnRight^t) \\ &\quad \vee (FacingSouth^t \wedge TurnLeft^t) \end{aligned}$$

$$WumpusAlive^{t+1} \Leftrightarrow WumpusAlive^t \wedge \neg(WumpusAhead^t \wedge HaveArrow^t \wedge Shoot^t).$$

The *WumpusAhead* fluent does not need a successor-state axiom, since it is definable synchronously in terms of the agent location and orientation fluents and the wumpus location. The definition is extraordinarily tedious, illustrating the weakness of proposition logic. Note also that in the second edition we described a successor-state axiom (in the form of a circuit) for *WumpusAlive* that used the *Scream* observation to infer the wumpus's death, with no need for describing the complicated physics of shooting. Such an axiom suffices for state estimation, but nor for planning.

Exercise 7.7.#HYBR

Modify the HYBRID-WUMPUS-AGENT to use the 1-CNF logical state estimation method described on page 243. We noted on that page that such an agent will not be able to acquire, maintain, and use more complex beliefs such as the disjunction $P_{3,1} \vee P_{2,2}$. Suggest a method for overcoming this problem by defining additional proposition symbols, and try it out in the wumpus world. Does it improve the performance of the agent?

The required modifications are to add definitional axioms such as

$$P_{3,1} \text{ or } P_{2,2} \Leftrightarrow P_{3,1} \vee P_{2,2}$$

and to include the new literals on the list of literals whose truth values are to be inferred at each time step.

One natural way to extend the 1-CNF representation is to add test additional non-literal sentences. The sentences we choose to test can depend on inferences from the current KB. This can work if the number of additional sentences we need to test is not too large.

For example, we can query the knowledge base to find out which squares we know have pits, which we know might have pits, and which states are breezy (we need to do this to compute the un-augmented 1-CNF belief state). Then, for each breezy square, test the sentence “one of the neighbours of this square which might have a pit does have a pit.” For example, this would test $P_{3,1} \vee P_{2,2}$ if we had perceived a breeze in square (2,1). Under the Wumpus

physics, this literal will be true iff the breezy square has no known pit around it.

Exercise 7.7.#KBPO

What are the primary differences in the logical inferences required for a logic-based agent operating in a fully observable environment versus a partially observable environment?

In a partially observable environment, information about the current state must be inferred from information about previous states, actions, and percepts. That is, the agent needs to do state-estimation inferences. In a fully observable environment, these inferences are unnecessary because the percepts provide all relevant information about the current state; indeed, successor-state axioms may not be needed at all! In both cases, the agent may need to process the sensor information to infer propositions that are directly useful for action, such as propositions asserting that particular adjacent squares are safe to move into.

EXERCISES

8

FIRST-ORDER LOGIC

8.1 Representation Revisited

Exercise 8.1.#LKNB

A logical knowledge base represents the world using a set of sentences with no explicit structure. An **analogical** representation, on the other hand, has physical structure that corresponds directly to the structure of the thing represented. Consider a road map of your country as an analogical representation of facts about the country—it represents facts with a map language. The two-dimensional structure of the map corresponds to the two-dimensional surface of the area.

- a. Give five examples of *symbols* in the map language.
- b. An *explicit* sentence is a sentence that the creator of the representation actually writes down. An *implicit* sentence is a sentence that results from explicit sentences because of properties of the analogical representation. Give three examples each of *implicit* and *explicit* sentences in the map language.
- c. Give three examples of facts about the physical structure of your country that cannot be represented in the map language.
- d. Give two examples of facts that are much easier to express in the map language than in first-order logic.
- e. Give two other examples of useful analogical representations. What are the advantages and disadvantages of each of these languages?

This question will generate a wide variety of possible solutions. The key distinction between analogical and sentential representations is that the analogical representation automatically generates consequences that can be “read off” whenever suitable premises are encoded. When you get into the details, this distinction turns out to be quite hard to pin down—for example, what does “read off” mean?—but it can be justified by examining the time complexity of various inferences on the “virtual inference machine” provided by the representation system.

- a. Depending on the scale and type of the map, symbols in the map language typically include city and town markers, road symbols (various types), lighthouses, historic monuments, river courses, freeway intersections, etc.
- b. Explicit and implicit sentences: this distinction is a little tricky, but the basic idea is that when the map-drawer plunks a symbol down in a particular place, he says one explicit thing (e.g. that Coit Tower is here), but the analogical structure of the map representa-

tion means that many implicit sentences can now be derived. Explicit sentences: there is a monument called Coit Tower at this location; Lombard Street runs (approximately) east-west; San Francisco Bay exists and has this shape. Implicit sentences: Van Ness is longer than North Willard; Fisherman's Wharf is north of the Mission District; the shortest drivable route from Coit Tower to Twin Peaks is the following . . .

- c. Sentences unrepresentable in the map language: Telegraph Hill is approximately conical and about 430 feet high (assuming the map has no topographical notation); in 1890 there was no bridge connecting San Francisco to Marin County (map does not represent changing information); Interstate 680 runs either east or west of Walnut Creek (no disjunctive information).
- d. Sentences that are easier to express in the map language: any sentence that can be written easily in English is not going to be a good candidate for this question. Any *linguistic* abstraction from the physical structure of San Francisco (e.g. San Francisco is on the end of a peninsula at the mouth of a bay) can probably be expressed equally easily in the predicate calculus, since that's what it was designed for. Facts such as the shape of the coastline, or the path taken by a road, are best expressed in the map language. Even then, one can argue that the coastline drawn on the map actually consists of lots of individual sentences, one for each dot of ink, especially if the map is drawn using a digital plotter. In this case, the advantage of the map is really in the ease of inference combined with suitability for human "visual computing" apparatus.
- e. Examples of other analogical representations:
 - Analog audio tape recording. Advantages: simple circuits can record and reproduce sounds. Disadvantages: subject to errors, noise; hard to process in order to separate sounds or remove noise etc.
 - Traditional clock face. Advantages: easier to read quickly, determination of how much time is available requires no additional computation. Disadvantages: hard to read precisely, cannot represent small units of time (ms) easily.
 - All kinds of graphs, bar charts, pie charts. Advantages: enormous data compression, easy trend analysis, communicate information in a way which we can interpret easily. Disadvantages: imprecise, cannot represent disjunctive or negated information.

8.2 Syntax and Semantics of First-Order Logic

Exercise 8.2.#KNBT

Consider a knowledge base containing just two sentences: $P(a)$ and $P(b)$. Does this knowledge base entail $\forall x P(x)$? Explain your answer in terms of models.

The knowledge base does not entail $\forall x P(x)$. To show this, we must give a model where $P(a)$ and $P(b)$ but $\forall x P(x)$ is false. Consider any model with three domain elements, where a and b refer to the first two elements and the relation referred to by P holds only for those two elements.

Exercise 8.2.#VALID

Is the sentence $\exists x, y \ x = y$ valid? Explain.

The sentence $\exists x, y \ x = y$ is valid. A sentence is valid if it is true in every model. An existentially quantified sentence is true in a model if it holds under any extended interpretation in which its variables are assigned to domain elements. According to the standard semantics of FOL as given in the chapter, every model contains at least one domain element, hence, for any model, there is an extended interpretation in which x and y are assigned to the first domain element. In such an interpretation, $x = y$ is true.

Exercise 8.2.#SENO

Write down a logical sentence such that every world in which it is true contains exactly one object.

$\forall x, y \ x = y$ stipulates that there is exactly one object. If there are two objects, then there is an extended interpretation in which x and y are assigned to different objects, so the sentence would be false. Some students may also notice that any unsatisfiable sentence also meets the criterion, since there are no worlds in which the sentence is true.

Exercise 8.2.#SENT

Write down a logical sentence such that every world in which it is true contains exactly two objects.

$\exists x, y \ x \neq y \wedge \forall z \ x = z \vee y = z$ stipulates that there are exactly two objects.

Exercise 8.2.#MCNT

Consider a symbol vocabulary that contains c constant symbols, p_k predicate symbols of each arity k , and f_k function symbols of each arity k , where $1 \leq k \leq A$. Let the domain size be fixed at D . For any given model, each predicate or function symbol is mapped onto a relation or function, respectively, of the same arity. You may assume that the functions in the model allow some input tuples to have no value for the function (i.e., the value is the invisible object). Derive a formula for the number of possible models for a domain with D elements. Don't worry about eliminating redundant combinations.

We will use the simplest counting method, ignoring redundant combinations. For the constant symbols, there are D^c assignments. Each predicate of arity k is mapped onto a k -ary relation, i.e., a subset of the D^k possible k -element tuples; there are 2^{D^k} such mappings. Each function symbol of arity k is mapped onto a k -ary function, which specifies a value for each of the D^k possible k -element tuples. Including the invisible element, there are $D + 1$ choices for each value, so there are $(D + 1)^{D^k}$ functions. The total number of possible combinations

is therefore

$$D^c \cdot \left(\sum_{k=1}^A 2^{D^k} \right) \cdot \left(\sum_{k=1}^A (D+1)^{D^k} \right).$$

Two things to note: first, the number is finite; second, the maximum arity A is the most crucial complexity parameter.

Exercise 8.2.#VALS

Which of the following are valid (necessarily true) sentences?

- a. $(\exists x x=x) \Rightarrow (\forall y \exists z y=z)$.
- b. $\forall x P(x) \vee \neg P(x)$.
- c. $\forall x Smart(x) \vee (x=x)$.

Validity in first-order logic requires truth in all possible models:

- a. $(\exists x x=x) \Rightarrow (\forall y \exists z y=z)$.

Valid. The LHS is valid by itself—in standard FOL, every model has at least one object; hence, the whole sentence is valid iff the RHS is valid. (Otherwise, we can find a model where the LHS is true and the RHS is false.) The RHS is valid because for every value of y in any given model, there is a z —namely, the value of y itself—that is identical to y .

- b. $\forall x P(x) \vee \neg P(x)$.

Valid. For any relation denoted by P , every object x is either in the relation or not in it.

- c. $\forall x Smart(x) \vee (x=x)$.

Valid. In every model, every object satisfies $x=x$, so the disjunction is satisfied regardless of whether x is smart.

Exercise 8.2.#EMPT

Consider a version of the semantics for first-order logic in which models with empty domains are allowed. Give at least two examples of sentences that are valid according to the standard semantics but not according to the new semantics. Discuss which outcome makes more intuitive sense for your examples.

This version of FOL, first studied in depth by Mostowski (1951), goes under the title of **free logic** (Lambert, 1967). By a natural extension of the truth values for empty conjunctions (true) and empty disjunctions (false), every universally quantified sentence is true in empty models and every existentially quantified sentence is false. The semantics also needs to be adjusted to handle the fact that constant symbols have no referent in an empty model.

Examples of sentences valid in the standard semantics but not in free logic include $\exists x x=x$ and $[\forall x P(x)] \Rightarrow [\exists x P(x)]$. More importantly, perhaps, the equivalence of $\phi \vee \exists x \psi$ and $\exists x \phi \vee \psi$ when x does not occur free in ϕ , which is used for putting sentences into CNF, does not hold.

One could argue that $\exists x \ x = x$, which simply states that the model is nonempty, is not naturally a valid sentence, and that it ought to be possible to contemplate a universe with no objects. However, experience has shown that free logic seems to require extra work to rule out the empty model in many commonly occurring cases of logical representation and reasoning.

Exercise 8.2.#TORF

True or false? Explain.

- $\exists x \ x = Rumpelstiltskin$ is a valid (necessarily true) sentence of first-order logic.
- Every existentially quantified sentence in first-order logic is true in any model that contains exactly one object.
- $\forall x, y \ x = y$ is satisfiable.

- True. In every model, the constant symbol *Rumpelstiltskin* must have a referent; the extended interpretation in which x is assigned to that referent satisfies the existential sentence.
- False. We can easily write an existentially quantified sentence that forces there to be at least two objects: $\exists x, y \ x \neq y$. We can also write unsatisfiable things about one object: $\exists x \ P(x) \wedge \neg P(x)$. Finally, even the simple sentence $\exists x \ P(x)$ is false in the one-object-model where P is the empty relation.
- True. $\forall x, y \ x = y$ is true in any one-object model.

8.3 Using First-Order Logic

Exercise 8.3.#JIMG

Does the fact $\neg Spouse(George, Laura)$ follow from the facts $Jim \neq George$ and $Spouse(Jim, Laura)$? If so, give a proof; if not, supply additional axioms as needed. What happens if we use *Spouse* as a unary function symbol instead of a binary predicate?

The fact $\neg Spouse(George, Laura)$ does not follow. We need to assert that at most one person can be the spouse of any given person:

$$\forall x, y, z \ Spouse(x, z) \wedge Spouse(y, z) \Rightarrow x = y .$$

With this axiom, a resolution proof of $\neg Spouse(George, Laura)$ is straightforward.

If *Spouse* is a unary function symbol, then the question is whether $\neg Spouse(Laura) = George$ follows from $Jim \neq George$ and $Spouse(Laura) = Jim$. The answer is yes, it does follow. They could not both be the value of the function applied to the same argument if they were different objects.

Exercise 8.3.#MAPC

This exercise uses the function *MapColor* and predicates *In(x, y)*, *Borders(x, y)*, and *Country(x)*, whose arguments are geographical regions, along with constant symbols for various regions. In each of the following we give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

- a. Paris and Marseilles are both in France.

- (i) $In(Paris \wedge Marseilles, France)$.
- (ii) $In(Paris, France) \wedge In(Marseilles, France)$.
- (iii) $In(Paris, France) \vee In(Marseilles, France)$.

- b. There is a country that borders both Iraq and Pakistan.

- (i) $\exists c \ Country(c) \wedge Border(c, Iraq) \wedge Border(c, Pakistan)$.
- (ii) $\exists c \ Country(c) \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
- (iii) $[\exists c \ Country(c)] \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
- (iv) $\exists c \ Border(Country(c), Iraq \wedge Pakistan)$.

- c. All countries that border Ecuador are in South America.

- (i) $\forall c \ Country(c) \wedge Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)$.
- (ii) $\forall c \ Country(c) \Rightarrow [Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)]$.
- (iii) $\forall c \ [Country(c) \Rightarrow Border(c, Ecuador)] \Rightarrow In(c, SouthAmerica)$.
- (iv) $\forall c \ Country(c) \wedge Border(c, Ecuador) \wedge In(c, SouthAmerica)$.

- d. No region in South America borders any region in Europe.

- (i) $\neg[\exists c, d \ In(c, SouthAmerica) \wedge In(d, Europe) \wedge Borders(c, d)]$.
- (ii) $\forall c, d \ [In(c, SouthAmerica) \wedge In(d, Europe)] \Rightarrow \neg Borders(c, d)$.
- (iii) $\neg\forall c \ In(c, SouthAmerica) \Rightarrow \exists d \ In(d, Europe) \wedge \neg Borders(c, d)$.
- (iv) $\forall c \ In(c, SouthAmerica) \Rightarrow \forall d \ In(d, Europe) \Rightarrow \neg Borders(c, d)$.

- e. No two adjacent countries have the same map color.

- (i) $\forall x, y \ \neg Country(x) \vee \neg Country(y) \vee \neg Borders(x, y) \vee \neg(MapColor(x) = MapColor(y))$.
- (ii) $\forall x, y \ (Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge \neg(x = y)) \Rightarrow \neg(MapColor(x) = MapColor(y))$.
- (iii) $\forall x, y \ Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge \neg(MapColor(x) = MapColor(y))$.
- (iv) $\forall x, y \ (Country(x) \wedge Country(y) \wedge Borders(x, y)) \Rightarrow MapColor(x \neq y)$.

- a. Paris and Marseilles are both in France.

- (i) $In(Paris \wedge Marseilles, France)$.
- (2) Syntactically invalid. Cannot use conjunction inside a term.
- (ii) $In(Paris, France) \wedge In(Marseilles, France)$.

- (1) Correct.
- (iii) $In(Paris, France) \vee In(Marseilles, France)$.
- (3) Incorrect. Disjunction does not express “both.”
- b.** There is a country that borders both Iraq and Pakistan.
- $\exists c \ Country(c) \wedge Border(c, Iraq) \wedge Border(c, Pakistan)$.
 - (1) Correct.
 - (ii) $\exists c \ Country(c) \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
 - (3) Incorrect. Use of implication in existential.
 - (iii) $[\exists c \ Country(c)] \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
 - (2) Syntactically invalid. Variable c used outside the scope of its quantifier.
 - (iv) $\exists c \ Border(Country(c), Iraq \wedge Pakistan)$.
 - (2) Syntactically invalid. Cannot use conjunction inside a term.
- c.** All countries that border Ecuador are in South America.
- (i) $\forall c \ Country(c) \wedge Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)$.
 - (1) Correct.
 - (ii) $\forall c \ Country(c) \Rightarrow [Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)]$.
 - (1) Correct. Equivalent to (i).
 - (iii) $\forall c \ [Country(c) \Rightarrow Border(c, Ecuador)] \Rightarrow In(c, SouthAmerica)$.
 - (3) Incorrect. The implication in the LHS is effectively an implication in an existential; in particular, it sanctions the RHS for all non-countries.
 - (iv) $\forall c \ Country(c) \wedge Border(c, Ecuador) \wedge In(c, SouthAmerica)$.
 - (3) Incorrect. Uses conjunction as main connective of a universal quantifier.
- d.** No region in South America borders any region in Europe.
- (i) $\neg[\exists c, d \ In(c, SouthAmerica) \wedge In(d, Europe) \wedge Borders(c, d)]$.
 - (1) Correct.
 - (ii) $\forall c, d \ [In(c, SouthAmerica) \wedge In(d, Europe)] \Rightarrow \neg Borders(c, d)$.
 - (1) Correct.
 - (iii) $\neg\forall c \ In(c, SouthAmerica) \Rightarrow \exists d \ In(d, Europe) \wedge \neg Borders(c, d)$.
 - (3) Incorrect. This says there is some country in South America that borders every country in Europe!
 - (iv) $\forall c \ In(c, SouthAmerica) \Rightarrow \forall d \ In(d, Europe) \Rightarrow \neg Borders(c, d)$.
 - (1) Correct.
- e.** No two adjacent countries have the same map color.
- (i) $\forall x, y \ \neg Country(x) \vee \neg Country(y) \vee \neg Borders(x, y) \vee \neg(MapColor(x) = MapColor(y))$.
 - (1) Correct.
 - (ii) $\forall x, y \ (Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge \neg(x = y)) \Rightarrow \neg(MapColor(x) = MapColor(y))$.
 - (1) Correct. The inequality is unnecessary because no country borders itself.
 - (iii) $\forall x, y \ Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge \neg(MapColor(x) = MapColor(y))$.
 - (3) Incorrect. Uses conjunction as main connective of a universal quantifier.

(iv) $\forall x, y \ (Country(x) \wedge Country(y) \wedge Borders(x, y)) \Rightarrow MapColor(x \neq y)$.

(2) Syntactically invalid. Cannot use inequality inside a term.

Exercise 8.3.#BOSS

Consider a vocabulary with the following symbols:

Occupation(p, o): Predicate. Person p has occupation o .

Customer(p1, p2): Predicate. Person $p1$ is a customer of person $p2$.

Boss(p1, p2): Predicate. Person $p1$ is a boss of person $p2$.

Doctor, Surgeon, Lawyer, Actor: Constants denoting occupations.

Emily, Joe: Constants denoting people.

Use these symbols to write the following assertions in first-order logic:

- a. Emily is either a surgeon or a lawyer.
- b. Joe is an actor, but he also holds another job.
- c. All surgeons are doctors.
- d. Joe does not have a lawyer (i.e., is not a customer of any lawyer).
- e. Emily has a boss who is a lawyer.
- f. There exists a lawyer all of whose customers are doctors.
- g. Every surgeon has a lawyer.

- a. $O(E, S) \vee O(E, L)$.
- b. $O(J, A) \wedge \exists p \ p \neq A \wedge O(J, p)$.
- c. $\forall p \ O(p, S) \Rightarrow O(p, D)$.
- d. $\neg \exists p \ C(J, p) \wedge O(p, L)$.
- e. $\exists p \ B(p, E) \wedge O(p, L)$.
- f. $\exists p \ O(p, L) \wedge \forall q \ C(q, p) \Rightarrow O(q, D)$.
- g. $\forall p \ O(p, S) \Rightarrow \exists q \ O(q, L) \wedge C(p, q)$.

Exercise 8.3.#DOGS

In each of the following we give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

- a. Every cat loves its mother or father.

- (i) $\forall x \ Cat(x) \Rightarrow Loves(x, Mother(x)) \vee Father(x)$.
- (ii) $\forall x \ \neg Cat(x) \vee Loves(x, Mother(x)) \vee Loves(x, Father(x))$.
- (iii) $\forall x \ Cat(x) \wedge (Loves(x, Mother(x)) \vee Loves(x, Father(x)))$.

- b. Every dog who loves one of its brothers is happy.

- (i) $\forall x \ Dog(x) \wedge (\exists y \ Brother(y, x) \wedge Loves(x, y)) \Rightarrow Happy(x)$.

- (ii) $\forall x, y \ Dog(x) \wedge Brother(y, x) \wedge Loves(x, y) \Rightarrow Happy(x)$.
 (iii) $\forall x \ Dog(x) \wedge [\forall y \ Brother(y, x) \Leftrightarrow Loves(x, y)] \Rightarrow Happy(x)$.
- c. No dog bites a child of its owner.
- (i) $\forall x \ Dog(x) \Rightarrow \neg Bites(x, Child(Owner(x)))$.
 - (ii) $\neg \exists x, y \ Dog(x) \wedge Child(y, Owner(x)) \wedge Bites(x, y)$.
 - (iii) $\forall x \ Dog(x) \Rightarrow (\forall y \ Child(y, Owner(x)) \Rightarrow \neg Bites(x, y))$.
 - (iv) $\neg \exists x \ Dog(x) \Rightarrow (\exists y \ Child(y, Owner(x)) \wedge Bites(x, y))$.
- d. Everyone's zip code within a state has the same first digit.
- (i) $\forall x, s, z_1 [State(s) \wedge LivesIn(x, s) \wedge Zip(x) = z_1] \Rightarrow [\forall y, z_2 LivesIn(y, s) \wedge Zip(y) = z_2 \Rightarrow Digit(1, z_1) = Digit(1, z_2)]$.
 - (ii) $\forall x, s [State(s) \wedge LivesIn(x, s) \wedge \exists z_1 Zip(x) = z_1] \Rightarrow [\forall y, z_2 LivesIn(y, s) \wedge Zip(y) = z_2 \wedge Digit(1, z_1) = Digit(1, z_2)]$.
 - (iii) $\forall x, y, s State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \Rightarrow Digit(1, Zip(x)) = Zip(y)$.
 - (iv) $\forall x, y, s State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \Rightarrow Digit(1, Zip(x)) = Digit(1, Zip(y))$.

- a. Every cat loves its mother or father.
- (i) $\forall x \ Cat(x) \Rightarrow Loves(x, Mother(x)) \vee Father(x)$.
 (2) Syntactically invalid. Cannot have a disjunction inside a term.
 - (ii) $\forall x \ \neg Cat(x) \vee Loves(x, Mother(x)) \vee Loves(x, Father(x))$.
 (1) Correct. (Rewrite as implication with disjunctive consequence.)
 - (iii) $\forall x \ Cat(x) \wedge (Loves(x, Mother(x)) \vee Loves(x, Father(x)))$.
 (3) Incorrect. Use of \wedge with \forall means that everything is asserted to be a cat.
- b. Every dog who loves one of its brothers is happy.
- (i) $\forall x \ Dog(x) \wedge (\exists y \ Brother(y, x) \wedge Loves(x, y)) \Rightarrow Happy(x)$.
 (1) Correct.
 - (ii) $\forall x, y \ Dog(x) \wedge Brother(y, x) \wedge Loves(x, y) \Rightarrow Happy(x)$.
 (1) Correct. Logically equivalent to (i).
 - (iii) $\forall x \ Dog(x) \wedge [\forall y \ Brother(y, x) \Leftrightarrow Loves(x, y)] \Rightarrow Happy(x)$.
 (3) Incorrect. States that dogs are happy if they love all of, and only, their brothers.
- c. No dog bites a child of its owner.
- (i) $\forall x \ Dog(x) \Rightarrow \neg Bites(x, Child(Owner(x)))$.
 (3) Incorrect. Uses *Child* as a function instead of a relation.
 - (ii) $\neg \exists x, y \ Dog(x) \wedge Child(y, Owner(x)) \wedge Bites(x, y)$.
 (1) Correct.
 - (iii) $\forall x \ Dog(x) \Rightarrow (\forall y \ Child(y, Owner(x)) \Rightarrow \neg Bites(x, y))$.
 (1) Correct. Logically equivalent to (ii).
 - (iv) $\neg \exists x \ Dog(x) \Rightarrow (\exists y \ Child(y, Owner(x)) \wedge Bites(x, y))$.
 (3) Incorrect. Uses \Rightarrow with \exists .

d. Everyone's zip code within a state has the same first digit.

$$(i) \forall x, s, z_1 [State(s) \wedge LivesIn(x, s) \wedge Zip(x) = z_1] \Rightarrow [\forall y, z_2 LivesIn(y, s) \wedge Zip(y) = z_2 \Rightarrow Digit(1, z_1) = Digit(1, z_2)].$$

(1) Correct.

$$(ii) \forall x, s [State(s) \wedge LivesIn(x, s) \wedge \exists z_1 Zip(x) = z_1] \Rightarrow [\forall y, z_2 LivesIn(y, s) \wedge Zip(y) = z_2 \wedge Digit(1, z_1) = Digit(1, z_2)].$$

(2) Syntactically invalid. Uses z_1 outside scope of its quantifier. Also uses \wedge as the main connective in the universally quantified RHS.

$$(iii) \forall x, y, s State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \Rightarrow Digit(1, Zip(x) = Zip(y)).$$

(2) Syntactically invalid. Cannot use equality within a term.

$$(iv) \forall x, y, s State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \Rightarrow Digit(1, Zip(x)) = Digit(1, Zip(y)).$$

(1) Correct. Since Zip is a function, there is no need to define additional variables to name the zip codes.

Exercise 8.3.#LAND

Complete the following exercises about logical sentences:

a. Translate into *good, natural* English (no xs or ys !):

$$\forall x, y, l SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l) \Rightarrow Understands(x, y) \wedge Understands(y, x).$$

b. Explain why this sentence is entailed by the sentence

$$\forall x, y, l SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l) \Rightarrow Understands(x, y).$$

c. Translate into first-order logic the following sentences:

(i) Understanding leads to friendship.

(ii) Friendship is transitive.

Remember to define all predicates, functions, and constants you use.

a. People who speak the same language understand each other.

b. Suppose that an extended interpretation with $x \rightarrow A$ and $y \rightarrow B$ satisfy

$$SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l)$$

for some l . Then from the second sentence we can conclude $Understands(A, B)$. The extended interpretation with $x \rightarrow B$ and $y \rightarrow A$ also must satisfy

$$SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l),$$

allowing us to conclude $Understands(B, A)$. Hence, whenever the second sentence holds, the first holds.

- c. Let $Understands(x, y)$ mean that x understands y , and let $Friend(x, y)$ mean that x is a friend of y .
- (i) It is not completely clear if the English sentence is referring to mutual understanding and mutual friendship, but let us assume that is what is intended:
 $\forall x, y \ Understands(x, y) \wedge Understands(y, x) \Rightarrow (Friend(x, y) \wedge Friend(y, x)).$
 - (ii) $\forall x, y, z \ Friend(x, y) \wedge Friend(y, z) \Rightarrow Friend(x, z).$

Exercise 8.3.#PEAN

Rewrite the first two Peano axioms in Section 8.3.3 as a single axiom that defines $NatNum(x)$ so as to exclude the possibility of natural numbers except for those generated by the successor function.

This exercise requires a rewriting similar to the Clark completion of the two Horn clauses:

$$\forall n \ NatNum(n) \Leftrightarrow [n = 0 \vee \exists m \ NatNum(m) \wedge n = S(m)].$$

Exercise 8.3.#WUMD

Equation (8.4) on page 271 defines the conditions under which a square is breezy. Here we consider two other ways to describe this aspect of the wumpus world.

- a. We can write **diagnostic rules** leading from observed effects to hidden causes. For finding pits, the obvious diagnostic rules say that if a square is breezy, some adjacent square must contain a pit; and if a square is not breezy, then no adjacent square contains a pit. Write these two rules in first-order logic and show that their conjunction is logically equivalent to Equation (8.4).
- b. We can write **causal rules** leading from cause to effect. One obvious causal rule is that a pit causes all adjacent squares to be breezy. Write this rule in first-order logic, explain why it is incomplete compared to Equation (8.4), and supply the missing axiom.

- a. The two implication sentences are

$$\begin{aligned} \forall s \ Breezy(s) &\Rightarrow \exists r \ Adjacent(r, s) \wedge Pit(r) \\ \forall s \ \neg Breezy(s) &\Rightarrow \neg \exists r \ Adjacent(r, s) \wedge Pit(r). \end{aligned}$$

The converse of the second sentence is

$$\forall s \ \exists r \ Adjacent(r, s) \wedge Pit(r) \Rightarrow Breezy(s)$$

which, combined with the first sentence, immediately gives

$$\forall s \ Breezy(s) \Leftrightarrow \exists r \ Adjacent(r, s) \wedge Pit(r).$$

- b. To say that a pit causes all adjacent squares to be breezy:

$$\forall s \ Pit(s) \Rightarrow [\forall r \ Adjacent(r, s) \Rightarrow Breezy(r)] .$$

This axiom allows for breezes to occur spontaneously with no adjacent pits. It would be incorrect to say that a non-pit causes all adjacent squares to be non-breezy, since there might be pits in other squares causing one of the adjacent squares to be breezy. But if *all* adjacent squares have no pits, a square is non-breezy:

$$\forall s \ [\forall r \ Adjacent(r, s) \Rightarrow \neg Pit(r)] \Rightarrow \neg Breezy(s) .$$

Exercise 8.3.#KINS

Write axioms describing the predicates *Grandchild*, *Greatgrandparent*, *Ancestor*, *Brother*, *Sister*, *Daughter*, *Son*, *FirstCousin*, *BrotherInLaw*, *SisterInLaw*, *Aunt*, and *Uncle*. Find out the proper definition of *mth cousin n times removed*, and write the definition in first-order logic. Now write down the basic facts depicted in the family tree in Figure ???. Using a suitable logical reasoning system, TELL it all the sentences you have written down, and ASK it who are Elizabeth's grandchildren, Diana's brothers-in-law, Zara's great-grandparents, and Eugenie's ancestors.

Make sure you write definitions with \Leftrightarrow . If you use \Rightarrow , you are only imposing constraints, not writing a real definition. Note that for aunts and uncles, we include the relations whom the OED says are more strictly defined as aunts-in-law and uncles-in-law, since the latter terms are not in common use.

$$\begin{aligned} Grandchild(c, a) &\Leftrightarrow \exists b \ Child(c, b) \wedge Child(b, a) \\ Greatgrandparent(a, d) &\Leftrightarrow \exists b, c \ Child(d, c) \wedge Child(c, b) \wedge Child(b, a) \\ Ancestor(a, x) &\Leftrightarrow Child(x, a) \vee \exists b \ Child(b, a) \wedge Ancestor(b, x) \\ Brother(x, y) &\Leftrightarrow Male(x) \wedge Sibling(x, y) \\ Sister(x, y) &\Leftrightarrow Female(x) \wedge Sibling(x, y) \\ Daughter(d, p) &\Leftrightarrow Female(d) \wedge Child(d, p) \\ Son(s, p) &\Leftrightarrow Male(s) \wedge Child(s, p) \\ FirstCousin(c, d) &\Leftrightarrow \exists p_1, p_2 \ Child(c, p_1) \wedge Child(d, p_2) \wedge Sibling(p_1, p_2) \\ BrotherInLaw(b, x) &\Leftrightarrow \exists m \ Spouse(x, m) \wedge Brother(b, m) \\ SisterInLaw(s, x) &\Leftrightarrow \exists m \ Spouse(x, m) \wedge Sister(s, m) \\ Aunt(a, c) &\Leftrightarrow \exists p \ Child(c, p) \wedge [Sister(a, p) \vee SisterInLaw(a, p)] \\ Uncle(u, c) &\Leftrightarrow \exists p \ Child(c, p) \wedge [Brother(a, p) \vee BrotherInLaw(a, p)] \end{aligned}$$

There are several equivalent ways to define an *mth cousin n times removed*. One way is to look at the distance of each person to the nearest common ancestor. Define *Distance*(*c, a*) as follows:

$$\begin{aligned} Distance(c, c) &= 0 \\ Child(c, b) \wedge Distance(b, a) = k &\Rightarrow Distance(c, a) = k + 1 . \end{aligned}$$

Thus, the distance to one's grandparent is 2, great-great-grandparent is 4, and so on. Now we have

$$\begin{aligned} MthCousinNTimesRemoved(c, d, m, n) &\Leftrightarrow \\ \exists a \ Distance(c, a) = m + 1 \wedge Distance(d, a) = m + n + 1. \end{aligned}$$

The facts in the family tree are simple: each arrow represents two instances of *Child* (e.g., *Child(William, Diana)* and *Child(William, Charles)*), each name represents a sex proposition (e.g., *Male(William)* or *Female(Diana)*), each “bowtie” symbol indicates a *Spouse* proposition (e.g., *Spouse(Charles, Diana)*). Making the queries of the logical reasoning system is just a way of debugging the definitions.

Exercise 8.3.#COMM

Write down a sentence asserting that $+$ is a commutative function. Does your sentence follow from the Peano axioms? If so, explain why; if not, give a model in which the axioms are true and your sentence is false.

Commutativity of $+$ is asserted by

$$\forall m, n \ NatNum(m) \wedge NatNum(n) \Rightarrow + (m, n) = + (n, m).$$

The sentence follows from the Peano axioms and can be proved by double induction over m and n . For example, the base case for $m = 0$ is proved as follows: Omitting all references to *NatNum* for clarity, we must prove

$$\forall n \ + (0, n) = +(n, 0).$$

- Base case $n = 0$: the assertion reduces to $+(0, 0) = +(0, 0)$, which is trivially true.
- Inductive step: given $\forall n \ + (0, n) = +(n, 0)$, prove $\forall n \ + (0, S(n)) = +(S(n), 0)$.

We have

$$\begin{aligned} +(S(n), 0) &= S(+ (n, 0)) \quad \text{by the second axiom for addition} \\ &= S(+ (0, n)) \quad \text{by the inductive hypothesis} \\ &= S(n) \quad \text{by the first axiom for addition} \\ &= +(0, S(n)) \quad \text{by the first axiom for addition.} \end{aligned}$$

The inductive step for m proceeds similarly.

Exercise 8.3.#SETM

Explain what is wrong with the following proposed definition of the set membership predicate \in :

$$\begin{aligned} \forall x, s \ x \in \{x|s\} \\ \forall x, s \ x \in s \Rightarrow \forall y \ x \in \{y|s\}. \end{aligned}$$

Although these axioms are sufficient to prove set membership when x is in fact a member of a given set, they have nothing to say about cases where x is not a member. For example, it is not possible to prove that x is not a member of the empty set. These axioms may therefore be suitable for a logical system, such as Prolog, that uses negation-as-failure.

Exercise 8.3.#LIST

Using the set axioms as examples, write axioms for the list domain, including all the constants, functions, and predicates mentioned in the chapter.

Here we translate *List?* to mean “proper list” in Lisp terminology, i.e., a cons structure with *Nil* as the “rightmost” atom.

$$\begin{aligned}
 &List?(Nil) \\
 &\forall x, l \ List?(l) \Leftrightarrow List?(Cons(x, l)) \\
 &\forall x, y \ First(Cons(x, y)) = x \\
 &\forall x, y \ Rest(Cons(x, y)) = y \\
 &\forall x \ Append(Nil, x) = x \\
 &\forall v, x, y, z \ List?(x) \Rightarrow (Append(x, y) = z \Leftrightarrow Append(Cons(v, x), y) = Cons(v, z)) \\
 &\forall x \ \neg Find(x, Nil) \\
 &\forall x \ List?(z) \Rightarrow (Find(x, Cons(y, z)) \Leftrightarrow (x = y \vee Find(x, z)))
 \end{aligned}$$

Exercise 8.3.#ADJX

Explain what is wrong with the following proposed definition of adjacent squares in the wumpus world:

$$\forall x, y \ Adjacent([x, y], [x + 1, y]) \wedge Adjacent([x, y], [x, y + 1]).$$

There are several problems with the proposed definition. It allows one to prove, say, $Adjacent([1, 1], [1, 2])$ but not $Adjacent([1, 2], [1, 1])$; so we need an additional symmetry axiom. It does not allow one to prove that $Adjacent([1, 1], [1, 3])$ is false, so it needs to be written as

$$\forall s_1, s_2 \Leftrightarrow \dots$$

Finally, it does not work as the boundaries of the world, so some extra conditions must be added.

Exercise 8.3.#WUML

Write out the axioms required for reasoning about the wumpus's location, using a constant symbol *Wumpus* and a binary predicate *At(Wumpus, Location)*. Remember that there is only one wumpus.

We need the following sentences:

$$\begin{aligned} \forall s_1 \ Smelly(s_1) &\Leftrightarrow \exists s_2 \ Adjacent(s_1, s_2) \wedge In(Wumpus, s_2) \\ \exists s_1 \ In(Wumpus, s_1) \wedge \forall s_2 \ (s_1 \neq s_2) &\Rightarrow \neg In(Wumpus, s_2). \end{aligned}$$

Exercise 8.3.#JOAN

Assuming predicates *Parent(p, q)* and *Female(p)* and constants *Joan* and *Kevin*, with the obvious meanings, express each of the following sentences in first-order logic. (You may use the abbreviation \exists^1 to mean “there exists exactly one.”)

- a. Joan has a daughter (possibly more than one, and possibly sons as well).
 - b. Joan has exactly one daughter (but may have sons as well).
 - c. Joan has exactly one child, a daughter.
 - d. Joan and Kevin have exactly one child together.
 - e. Joan has at least one child with Kevin, and no children with anyone else.
-
- a. $\exists x \ Parent(Joan, x) \wedge Female(x)$.
 - b. $\exists^1 x \ Parent(Joan, x) \wedge Female(x)$.
 - c. $\exists x \ Parent(Joan, x) \wedge Female(x) \wedge [\forall y \ Parent(Joan, y) \Rightarrow y = x]$.
(This is sometimes abbreviated “*Female($\iota(x)Parent(Joan, x)$)*”).
 - d. $\exists^1 c \ Parent(Joan, c) \wedge Parent(Kevin, c)$.
 - e.
- $$\begin{aligned} \exists c \ Parent(Joan, c) \wedge Parent(Kevin, c) \wedge \forall d, p \ [Parent(Joan, d) \wedge Parent(p, d)] \\ \Rightarrow [p = Joan \vee p = Kevin] \end{aligned}$$

Exercise 8.3.#ARTH

Arithmetic assertions can be written in first-order logic with the predicate symbol $<$, the function symbols $+$ and \times , and the constant symbols 0 and 1 . Additional predicates can also be defined with biconditionals.

- a. Represent the property “ x is an even number.”
- b. Represent the property “ x is prime.”
- c. Goldbach’s conjecture is the conjecture (unproven as yet) that every even number is equal to the sum of two primes. Represent this conjecture as a logical sentence.

- a. $\forall x \text{ Even}(x) \Leftrightarrow \exists y \ x = y + y.$
- b. $\forall x \text{ Prime}(x) \Leftrightarrow \forall y, z \ x = y \times z \Rightarrow y = 1 \vee z = 1.$
- c. $\forall x \text{ Even}(x) \Rightarrow \exists y, z \ \text{Prime}(y) \wedge \text{Prime}(z) \wedge x = y + z.$

Exercise 8.3.#EQWA

In Chapter 6, we used equality to indicate the relation between a variable and its value. For instance, we wrote $WA = red$ to mean that Western Australia is colored red. Representing this in first-order logic, we must write more verbosely $\text{ColorOf}(WA) = red$. What incorrect inference could be drawn if we wrote sentences such as $WA = red$ directly as logical assertions?

If we have $WA = red$ and $Q = red$ then we could deduce $WA = Q$, which is undesirable to both Western Australians and Queenslanders.

Exercise 8.3.#KEYS

Write in first-order logic the assertion that every key and at least one of every pair of socks will eventually be lost forever, using only the following vocabulary: $Key(x)$, x is a key; $Sock(x)$, x is a sock; $Pair(x, y)$, x and y are a pair; Now , the current time; $Before(t_1, t_2)$, time t_1 comes before time t_2 ; $Lost(x, t)$, object x is lost at time t .

$$\begin{aligned} \forall k \ Key(k) &\Rightarrow [\exists t_0 \ Before(Now, t_0) \wedge \forall t \ Before(t_0, t) \Rightarrow Lost(k, t)] \\ \forall s_1, s_2 \ Sock(s_1) \wedge Sock(s_2) \wedge Pair(s_1, s_2) &\Rightarrow \\ &[\exists t_1 \ Before(Now, t_1) \wedge \forall t \ Before(t_1, t) \Rightarrow Lost(s_1, t)] \vee \\ &[\exists t_2 \ Before(Now, t_2) \wedge \forall t \ Before(t_2, t) \Rightarrow Lost(s_2, t)]. \end{aligned}$$

Notice that the disjunction allows for both socks to be lost, as the English sentence implies.

Exercise 8.3.#ENGL

For each of the following sentences in English, decide if the accompanying first-order logic sentence is a good translation. If not, explain why not and correct it. (Some sentences may have more than one error!)

- a. No two people have the same social security number.

$$\neg \exists x, y, n \ Person(x) \wedge Person(y) \Rightarrow [HasSS\#(x, n) \wedge HasSS\#(y, n)].$$

- b. John's social security number is the same as Mary's.

$$\exists n \ HasSS\#(John, n) \wedge HasSS\#(Mary, n).$$

- c. Everyone's social security number has nine digits.

$$\forall x, n \ Person(x) \Rightarrow [HasSS\#(x, n) \wedge Digits(n, 9)].$$

- d. Rewrite each of the above (uncorrected) sentences using a function symbol $SS\#$ instead of the predicate $HasSS\#$.

- a. "No two people have the same social security number."

$$\neg \exists x, y, n \ Person(x) \wedge Person(y) \Rightarrow [HasSS\#(x, n) \wedge HasSS\#(y, n)].$$

This uses \Rightarrow with \exists . It also says that no person has a social security number because it doesn't restrict itself to the cases where x and y are not equal. Correct version:

$$\neg \exists x, y, n \ Person(x) \wedge Person(y) \wedge \neg(x = y) \wedge [HasSS\#(x, n) \wedge HasSS\#(y, n)]$$

- b. "John's social security number is the same as Mary's."

$$\exists n \ HasSS\#(John, n) \wedge HasSS\#(Mary, n).$$

This is OK.

- c. "Everyone's social security number has nine digits."

$$\forall x, n \ Person(x) \Rightarrow [HasSS\#(x, n) \wedge Digits(n, 9)].$$

This says that everyone has every number. $HasSS\#(x, n)$ should be in the premise:

$$\forall x, n \ Person(x) \wedge HasSS\#(x, n) \Rightarrow Digits(n, 9)$$

- d. Here $SS\#(x)$ denotes the social security number of x . Using a function enforces the rule that everyone has just one.

$$\neg \exists x, y \ Person(x) \wedge Person(y) \Rightarrow [SS\#(x) = SS\#(y)]$$

$$SS\#(John) = SS\#(Mary)$$

$$\forall x \ Person(x) \Rightarrow Digits(SS\#(x), 9)$$

Exercise 8.3.#ENGT

Translate into first-order logic the sentence "Everyone's DNA is unique and is derived from their parents' DNA." You must specify the precise intended meaning of your vocabulary terms. (Hint: Do not use the predicate $Unique(x)$, since uniqueness is not really a property

of an object in itself!)

Let $DNA(x)$ be a function denoting (a string representation of) a person's DNA and $DerivedFrom(d, d_1, d_2)$ be true if string d is derived from strings d_1 and d_2 . (Technically speaking, this predicate is a bit vague as the amount of allowed recombination and mutation is not specified.) Then we have

$$\begin{aligned} \forall x, y \ Person(x) \wedge Person(y) \wedge x \neq y &\Rightarrow DNA(x) \neq DNA(y) \\ \forall x \ Person(x) &\Rightarrow DerivedFrom(DNA(x), DNA(Mother(x)), DNA(Father(x))) \end{aligned}$$

Exercise 8.3.#ENGG

For each of the following sentences in English, decide if the accompanying first-order logic sentence is a good translation. If not, explain why not and correct it.

- a. Any apartment in London has lower rent than some apartments in Paris.

$$\forall x \ [Apt(x) \wedge In(x, London)] \Rightarrow \exists y \ ([Apt(y) \wedge In(y, Paris)] \Rightarrow (Rent(x) < Rent(y))).$$

- b. There is exactly one apartment in Paris with rent below \$1000.

$$\exists x \ Apt(x) \wedge In(x, Paris) \wedge \forall y \ [Apt(y) \wedge In(y, Paris) \wedge (Rent(y) < Dollars(1000))] \Rightarrow (y = x).$$

- c. If an apartment is more expensive than all apartments in London, it must be in Moscow.

$$\forall x \ Apt(x) \wedge [\forall y \ Apt(y) \wedge In(y, London) \wedge (Rent(x) > Rent(y))] \Rightarrow In(x, Moscow).$$

- a. Any apartment in London has lower rent than some apartments in Paris.

$$\forall x \ [Apt(x) \wedge In(x, London)] \Rightarrow \exists y \ ([Apt(y) \wedge In(y, Paris)] \Rightarrow (Rent(x) < Rent(y))).$$

This uses \Rightarrow with \exists ; replacing it with \wedge makes the sentence correct:

$$\forall x \ [Apt(x) \wedge In(x, London)] \Rightarrow \exists y \ ([Apt(y) \wedge In(y, Paris)] \wedge (Rent(x) < Rent(y))).$$

- b. There is exactly one apartment in Paris with rent below \$1000.

$$\exists x \text{ } Apt(x) \wedge In(x, \text{Paris}) \wedge \\ \forall y \text{ } [Apt(y) \wedge In(y, \text{Paris}) \wedge (Rent(y) < Dollars(1000))] \Rightarrow (y = x).$$

This sentence would hold in a world with no apartments below \$1000. We need to say that x has the required property:

$$\exists x \text{ } Apt(x) \wedge In(x, \text{Paris}) \wedge (Rent(x) < Dollars(1000)) \wedge \\ \forall y \text{ } [Apt(y) \wedge In(y, \text{Paris}) \wedge (Rent(y) < Dollars(1000))] \Rightarrow (y = x).$$

- c. If an apartment is more expensive than all apartments in London, it must be in Moscow.

$$\forall x \text{ } Apt(x) \wedge [\forall y \text{ } Apt(y) \wedge In(y, \text{London}) \wedge (Rent(x) > Rent(y))] \Rightarrow \\ In(x, \text{Moscow}).$$

This uses \wedge with \forall . Replacing it with an implication fixes the problem:

$$\forall x \text{ } Apt(x) \wedge [\forall y \text{ } Apt(y) \wedge In(y, \text{London}) \Rightarrow (Rent(x) > Rent(y))] \Rightarrow \\ In(x, \text{Moscow}).$$

Exercise 8.3.#FOLV

Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define):

- a. Some students took French in spring 2001.
- b. Every student who takes French passes it.
- c. Only one student took Greek in spring 2001.
- d. The best score in Greek is always higher than the best score in French.
- e. Every person who buys a policy is smart.
- f. No person buys an expensive policy.
- g. There is an agent who sells policies only to people who are not insured.
- h. There is a barber who shaves all men in town who do not shave themselves.
- i. A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.
- j. A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.
- k. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.
- l. All Greeks speak the same language. (Use $Speaks(x, l)$ to mean that person x speaks language l .)

In this exercise, it is best not to worry about details of tense and larger concerns with consistent ontologies and so on. The main point is to make sure students understand connectives and quantifiers and the use of predicates, functions, constants, and equality. Let the basic vocabulary be as follows:

Takes(x, c, s): student x takes course c in semester s ;

Passes(x, c, s): student x passes course c in semester s ;

Score(x, c, s): the score obtained by student x in course c in semester s ;

$x > y$: x is greater than y ;

F and G : specific French and Greek courses (one could also interpret these sentences as referring to *any* such course, in which case one could use a predicate *Subject*(c, f) meaning that the subject of course c is field f);

Buys(x, y, z): x buys y from z (using a binary predicate with unspecified seller is OK but less felicitous);

Sells(x, y, z): x sells y to z ;

Shaves(x, y): person x shaves person y

Born(x, c): person x is born in country c ;

Parent(x, y): x is a parent of y ;

Citizen(x, c, r): x is a citizen of country c for reason r ;

Resident(x, c): x is a resident of country c ;

Fools(x, y, t): person x fools person y at time t ;

Student(x), *Person*(x), *Man*(x), *Barber*(x), *Expensive*(x), *Agent*(x), *Insured*(x), *Smart*(x), *Politician*(x): predicates satisfied by members of the corresponding categories.

- a. Some students took French in spring 2001.

$$\exists x \text{ } \textit{Student}(x) \wedge \textit{Takes}(x, F, \text{Spring}2001).$$

- b. Every student who takes French passes it.

$$\forall x, s \text{ } \textit{Student}(x) \wedge \textit{Takes}(x, F, s) \Rightarrow \textit{Passes}(x, F, s).$$

- c. Only one student took Greek in spring 2001.

$$\exists x \text{ } \textit{Student}(x) \wedge \textit{Takes}(x, G, \text{Spring}2001) \wedge \forall y \text{ } y \neq x \Rightarrow \neg \textit{Takes}(y, G, \text{Spring}2001).$$

- d. The best score in Greek is always higher than the best score in French.

$$\forall s \text{ } \exists x \text{ } \forall y \text{ } \textit{Score}(x, G, s) > \textit{Score}(y, F, s).$$

- e. Every person who buys a policy is smart.

$$\forall x \text{ } \textit{Person}(x) \wedge (\exists y, z \text{ } \textit{Policy}(y) \wedge \textit{Buys}(x, y, z)) \Rightarrow \textit{Smart}(x).$$

- f. No person buys an expensive policy.

$$\forall x, y, z \text{ } \textit{Person}(x) \wedge \textit{Policy}(y) \wedge \textit{Expensive}(y) \Rightarrow \neg \textit{Buys}(x, y, z).$$

- g. There is an agent who sells policies only to people who are not insured.

$$\exists x \text{ } \textit{Agent}(x) \wedge \forall y, z \text{ } \textit{Policy}(y) \wedge \textit{Sells}(x, y, z) \Rightarrow (\textit{Person}(z) \wedge \neg \textit{Insured}(z)).$$

- h. There is a barber who shaves all men in town who do not shave themselves.

$$\exists x \text{ } \textit{Barber}(x) \wedge \forall y \text{ } \textit{Man}(y) \wedge \neg \textit{Shaves}(y, y) \Rightarrow \textit{Shaves}(x, y).$$

- i. A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.

$$\forall x \text{ } \textit{Person}(x) \wedge \textit{Born}(x, \text{UK}) \wedge (\forall y \text{ } \textit{Parent}(y, x) \Rightarrow ((\exists r \text{ } \textit{Citizen}(y, \text{UK}, r)) \vee \textit{Resident}(y, \text{UK}))) \Rightarrow \textit{Citizen}(x, \text{UK}, \text{Birth}).$$

- j. A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

$$\begin{aligned} \forall x \ Person(x) \wedge \neg Born(x, UK) \wedge (\exists y \ Parent(y, x) \wedge Citizen(y, UK, Birth)) \\ \Rightarrow Citizen(x, UK, Descent). \end{aligned}$$

- k. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

$$\begin{aligned} \forall x \ Politician(x) \Rightarrow \\ (\exists y \ \forall t \ Person(y) \wedge Fools(x, y, t)) \wedge \\ (\exists t \ \forall y \ Person(y) \Rightarrow Fools(x, y, t)) \wedge \\ \neg(\forall t \ \forall y \ Person(y) \Rightarrow Fools(x, y, t)) \end{aligned}$$

- l. All Greeks speak the same language.

$$\begin{aligned} \forall x, y, l \ Person(x) \wedge [\exists r \ Citizen(x, Greece, r)] \wedge Person(y) \wedge [\exists r \ Citizen(y, Greece, r)] \\ \wedge Speaks(x, l) \Rightarrow Speaks(y, l) \end{aligned}$$

Exercise 8.3.#NAPD

Write a general set of facts and axioms to represent the assertion “Wellington heard about Napoleon’s death” and to correctly answer the question “Did Napoleon hear about Wellington’s death?”

This is a very educational exercise but also highly nontrivial. Once students have learned about resolution, ask them to do the proof too. In most cases, they will discover missing axioms. Our basic predicates are $Heard(x, e, t)$ (x heard about event e at time t); $Occurred(e, t)$ (event e occurred at time t); $Alive(x, t)$ (x is alive at time t).

$$\begin{aligned} \exists t \ Heard(W, DeathOf(N), t) \\ \forall x, e, t \ Heard(x, e, t) \Rightarrow Alive(x, t) \\ \forall x, e, t_2 \ Heard(x, e, t_2) \Rightarrow \exists t_1 \ Occurred(e, t_1) \wedge t_1 < t_2 \\ \forall t_1 \ Occurred(DeathOf(x), t_1) \Rightarrow \forall t_2 \ t_1 < t_2 \Rightarrow \neg Alive(x, t_2) \\ \forall t_1, t_2 \ \neg(t_2 < t_1) \Rightarrow ((t_1 < t_2) \vee (t_1 = t_2)) \\ \forall t_1, t_2, t_3 \ (t_1 < t_2) \wedge ((t_2 < t_3) \vee (t_2 = t_3)) \Rightarrow (t_1 < t_3) \\ \forall t_1, t_2, t_3 \ ((t_1 < t_2) \vee (t_1 = t_2)) \wedge (t_2 < t_3) \Rightarrow (t_1 < t_3) \end{aligned}$$

Exercise 8.3.#BEAT

Consider a first-order logical knowledge base that describes worlds containing people, songs, albums (e.g., “Meet the Beatles”) and disks (i.e., particular physical instances of CDs). The vocabulary contains the following symbols:

- $CopyOf(d, a)$: Predicate. Disk d is a copy of album a .
- $Owns(p, d)$: Predicate. Person p owns disk d .
- $Sings(p, s, a)$: Album a includes a recording of song s sung by person p .
- $Wrote(p, s)$: Person p wrote song s .

McCartney, Gershwin, BHoliday, Joe, EleanorRigby, TheManILove, Revolver: Constants with the obvious meanings.

Express the following statements in first-order logic:

- a. Gershwin wrote “The Man I Love.”
- b. Gershwin did not write “Eleanor Rigby.”
- c. Either Gershwin or McCartney wrote “The Man I Love.”
- d. Joe has written at least one song.
- e. Joe owns a copy of *Revolver*.
- f. Every song that McCartney sings on *Revolver* was written by McCartney.
- g. Gershwin did not write any of the songs on *Revolver*.
- h. Every song that Gershwin wrote has been recorded on some album. (Possibly different songs are recorded on different albums.)
- i. There is a single album that contains every song that Joe has written.
- j. Joe owns a copy of an album that has Billie Holiday singing “The Man I Love.”
- k. Joe owns a copy of every album that has a song sung by McCartney. (Of course, each different album is instantiated in a different physical CD.)
- l. Joe owns a copy of every album on which all the songs are sung by Billie Holiday.

- a. $W(G, T)$.
- b. $\neg W(G, E)$.
- c. $W(G, T) \vee W(M, T)$.
- d. $\exists s \ W(J, s)$.
- e. $\exists x \ C(x, R) \wedge O(J, x)$.
- f. $\forall s \ S(M, s, R) \Rightarrow W(M, s)$.
- g. $\neg [\exists s \ W(G, s) \wedge \exists p \ S(p, s, R)]$.
- h. $\forall s \ W(G, s) \Rightarrow \exists p, a \ S(p, s, a)$.
- i. $\exists a \ \forall s \ W(J, s) \Rightarrow \exists p \ S(p, s, a)$.
- j. $\exists d, a, s \ C(d, a) \wedge O(J, d) \wedge S(B, T, a)$.
- k. $\forall a \ [\exists s \ S(M, s, a)] \Rightarrow \exists d \ C(d, a) \wedge O(J, d)$.
- l. $\forall a \ [\forall s, p \ S(p, s, a) \Rightarrow S(B, s, a)] \Rightarrow \exists d \ C(d, a) \wedge O(J, d)$.

8.4 Knowledge Engineering in First-Order Logic

Exercise 8.4.#ADDR

Extend the vocabulary from Section 8.4 to define addition for n -bit binary numbers. Then encode the description of the four-bit adder in Figure ??, and pose the queries needed to verify that it is in fact correct.

There are three stages to go through. In the first stage, we define the concepts of one-bit and n -bit addition. Then, we specify one-bit and n -bit adder circuits. Finally, we verify that the n -bit adder circuit does n -bit addition.

- One-bit addition is easy. Let Add_1 be a function of three one-bit arguments (the third is the carry bit). The result of the addition is a list of bits representing a 2-bit binary number, least significant digit first:

$$\begin{aligned}Add_1(0, 0, 0) &= [0, 0] \\Add_1(0, 0, 1) &= [0, 1] \\Add_1(0, 1, 0) &= [0, 1] \\Add_1(0, 1, 1) &= [1, 0] \\Add_1(1, 0, 0) &= [0, 1] \\Add_1(1, 0, 1) &= [1, 0] \\Add_1(1, 1, 0) &= [1, 0] \\Add_1(1, 1, 1) &= [1, 1]\end{aligned}$$

- n -bit addition builds on one-bit addition. Let $Add_n(x_1, x_2, b)$ be a function that takes two lists of binary digits of length n (least significant digit first) and a carry bit (initially 0), and constructs a list of length $n + 1$ that represents their sum. (It will always be exactly $n + 1$ bits long, even when the leading bit is 0—the leading bit is the overflow bit.)

$$\begin{aligned}Add_n([], [], b) &= [b] \\Add_1(b_1, b_2, b) &= [b_3, b_4] \Rightarrow Add_n([b_1|x_1], [b_2|x_2], b) = [b_3|Add_n(x_1, x_2, b_4)]\end{aligned}$$

- The next step is to define the structure of a one-bit adder circuit, as given in the text. Let $Add_1Circuit(c)$ be true of any circuit that has the appropriate components and connections:

$$\begin{aligned}\forall c \ Add_1Circuit(c) \Leftrightarrow \\ \exists x_1, x_2, a_1, a_2, o_1 \ Type(x_1) = Type(x_2) = XOR \\ \wedge Type(a_1) = Type(a_2) = AND \wedge Type(o_1) = OR \\ \wedge Connected(Out(1, x_1), In(1, x_2)) \wedge Connected(In(1, c), In(1, x_1)) \\ \wedge Connected(Out(1, x_1), In(2, a_2)) \wedge Connected(In(1, c), In(1, a_1)) \\ \wedge Connected(Out(1, a_2), In(1, o_1)) \wedge Connected(In(2, c), In(2, x_1)) \\ \wedge Connected(Out(1, a_1), In(2, o_1)) \wedge Connected(In(2, c), In(2, a_1)) \\ \wedge Connected(Out(1, x_2), Out(1, c)) \wedge Connected(In(3, c), In(2, x_2)) \\ \wedge Connected(Out(1, o_1), Out(2, c)) \wedge Connected(In(3, c), In(1, a_2))\end{aligned}$$

Notice that this allows the circuit to have additional gates and connections, but they won't stop it from doing addition.

- Now we define what we mean by an n -bit adder circuit, following the design of Figure 8.6. We will need to be careful, because an n -bit adder is not just an $n - 1$ -bit adder plus a one-bit adder; we have to connect the overflow bit of the $n - 1$ -bit adder to the

carry-bit input of the one-bit adder. We begin with the base case, where $n = 0$:

$$\begin{aligned} \forall c \ Add_nCircuit(c, 0) &\Leftrightarrow \\ Signal(Out(1, c)) &= 0 \end{aligned}$$

Now, for the recursive case we specify that the first connect the “overflow” output of the $n - 1$ -bit circuit as the carry bit for the last bit:

$$\begin{aligned} \forall c, n \ n > 0 \Rightarrow [Add_nCircuit(c, n) &\Leftrightarrow \\ \exists c_2, d \ Add_nCircuit(c_2, n - 1) \wedge Add_1Circuit(d) & \\ \wedge \forall m \ (m > 0) \wedge (m < 2n - 1) \Rightarrow In(m, c) = In(m, c_2) & \\ \wedge \forall m \ (m > 0) \wedge (m < n) \Rightarrow Out(m, c) = Out(m, c_2) & \\ \wedge Connected(Out(n, c_2), In(3, d)) & \\ \wedge Connected(In(2n - 1, c), In(1, d)) \wedge Connected(In(2n, c), In(2, d)) & \\ \wedge Connected(Out(1, d), Out(n, c)) \wedge Connected(Out(2, d), Out(n + 1, c)) & \end{aligned}$$

- Now, to verify that a one-bit adder *circuit* actually adds correctly, we ask whether, given any setting of the inputs, the outputs equal the sum of the inputs:

$$\begin{aligned} \forall c \ Add_1Circuit(c) \Rightarrow \\ \forall i_1, i_2, i_3 \ Signal(In(1, c)) = i_1 \wedge Signal(In(2, c)) = i_2 \wedge Signal(In(3, c)) = i_3 \\ \Rightarrow Add_1(i_1, i_2, i_3) = [Out(1, c), Out(2, c)] \end{aligned}$$

If this sentence is entailed by the KB, then every circuit with the *Add₁Circuit* design is in fact an adder. The query for the n -bit can be written as

$$\begin{aligned} \forall c, n \ Add_nCircuit(c, n) \Rightarrow \\ \forall x_1, x_2, y \ InterleavedInputBits(x_1, x_2, c) \wedge OutputBits(y, c) \\ \Rightarrow Add_n(x_1, x_2, y) \end{aligned}$$

where *InterleavedInputBits* and *OutputBits* are defined appropriately to map bit sequences to the actual terminals of the circuit. [Note: this logical formulation has not been tested in a theorem prover and we hesitate to vouch for its correctness.]

Exercise 8.4.#CIRR

The circuit representation in the chapter is more detailed than necessary if we care only about circuit functionality. A simpler formulation describes any m -input, n -output gate or circuit using a predicate with $m + n$ arguments, such that the predicate is true exactly when the inputs and outputs are consistent. For example, NOT gates are described by the binary predicate *NOT*(i, o), for which *NOT*(0, 1) and *NOT*(1, 0) are known. Compositions of gates are defined by conjunctions of gate predicates in which shared variables indicate direct connections. For example, a NAND circuit can be composed from *ANDs* and *NOTs*:

$$\forall i_1, i_2, o_a, o \ AND(i_1, i_2, o_a) \wedge NOT(o_a, o) \Rightarrow NAND(i_1, i_2, o) .$$

Using this representation, define the one-bit adder in Figure 8.6 and the four-bit adder in Figure ??, and explain what queries you would use to verify the designs. What kinds of queries are *not* supported by this representation that *are* supported by the representation in Section 8.4?

Strictly speaking, the primitive gates must be defined using logical equivalences to exclude those combinations not listed as correct. If we are using a logic programming system, we can simply list the cases. For example,

$$AND(0, 0, 0) \quad AND(0, 1, 0) \quad AND(1, 0, 0) \quad AND(1, 1, 1).$$

For the one-bit adder, we have

$$\begin{aligned} \forall i_1, i_2, i_3, o_1, o_2 \quad &Add_1\ Circuit(i_1, i_2, i_3, o_1, o_2) \Leftrightarrow \\ \exists o_{x1}, o_{a1}, o_{x2} \quad &XOR(i_1, i_2, o_{x1}) \wedge XOR(o_{x1}, i_3, o_1) \\ \wedge AND(i_1, i_2, o_{a1}) \wedge AND(i_3, o_{x1}, o_{a2}) \\ \wedge OR(o_{a2}, o_{a1}, o_2) \end{aligned}$$

To form a verification query, we need some independent definition of one-bit binary addition. Let us assume this is supplied by a predicate *Add*₁. Then we need to prove

$$\forall i_1, i_2, i_3, o_1, o_2 \quad Add_1\ Circuit(i_1, i_2, i_3, o_1, o_2) \Rightarrow Add_1(i_1, i_2, i_3, o_1, o_2)$$

*Add*₁ itself can be defined by a lookup table or through the builtin arithmetic functions of the theorem prover.

The simplified representation cannot support queries about whether particular terminals are connected in a given circuit, since the terminals are not reified (nor is the circuit itself). Thus, it cannot be used to debug a faulty circuit; nor is it easy to extend with timing information, non-Boolean signal values, etc.

Exercise 8.4.#PASS

Obtain a passport application for your country, identify the rules determining eligibility for a passport, and translate them into first-order logic, following the steps outlined in Section 8.4.

The answers here will vary by country. The two key rules for UK passports are given above.

EXERCISES 9

INFERENCE IN FIRST-ORDER LOGIC

9.1 Propositional vs. First-Order Inference

Exercise 9.1.#UNIP

Prove that Universal Instantiation is sound and that Existential Instantiation produces an inferentially equivalent knowledge base.

We want to show that any sentence of the form $\forall v \ \alpha$ entails any universal instantiation of the sentence. The sentence $\forall v \ \alpha$ asserts that α is true in all possible extended interpretations. For any model specifying the referent of ground term g , the truth of $\text{SUBST}(\{v/g\}, \alpha)$ must be identical to the truth value of some extended interpretation in which v is assigned to an object, and in all such interpretations α is true.

EI states: for any sentence α , variable v , and constant symbol k that does not appear elsewhere in the knowledge base,

$$\frac{\exists v \ \alpha}{\text{SUBST}(\{v/k\}, \alpha)} .$$

If the knowledge base with the original existentially quantified sentence is KB and the result of EI is KB' , then we need to prove that KB is satisfiable iff KB' is satisfiable. Forward: if KB is satisfiable, it has a model M for which an extended interpretation assigning v to some object o renders α true. Hence, we can construct a model M' that satisfies KB' by assigning k to refer to o ; since k does not appear elsewhere, the truth values of all other sentences are unaffected. Backward: if KB' is satisfiable, it has a model M' with an assignment for k to some object o . Hence, we can construct a model M that satisfies KB with an extended interpretation assigning v to o ; since k does not appear elsewhere, removing it from the model leaves the truth values of all other sentences are unaffected.

Exercise 9.1.#EXIN

From $\text{Likes}(Jerry, IceCream)$ it seems reasonable to infer $\exists x \ \text{Likes}(x, IceCream)$. Write down a general inference rule, **Existential Introduction**, that sanctions this inference. State carefully the conditions that must be satisfied by the variables and terms involved.

For any sentence α containing a ground term g and for any variable v not occurring in α ,

we have

$$\frac{\alpha}{\exists v \text{ SUBST}^*(\{g/v\}, \alpha)}$$

where SUBST^* is a function that substitutes any or all of the occurrences of g with v . Notice that substituting just one occurrence and applying the rule multiple times is not the same, because it results in a weaker conclusion. For example, $P(a, a)$ should entail $\exists x P(x, x)$ rather than the weaker $\exists x, y P(x, y)$.

Exercise 9.1.#KBOS

Suppose a knowledge base contains just one sentence, $\exists x \text{ AsHighAs}(x, \text{Everest})$. Which of the following are legitimate results of applying Existential Instantiation?

- a. $\text{AsHighAs}(\text{Everest}, \text{Everest})$.
- b. $\text{AsHighAs}(\text{Kilimanjaro}, \text{Everest})$.
- c. $\text{AsHighAs}(\text{Kilimanjaro}, \text{Everest}) \wedge \text{AsHighAs}(\text{BenNevis}, \text{Everest})$
(after two applications).

Both b and c are sound conclusions; a is unsound because it introduces the previously-used symbol *Everest*. Note that c does not imply that there are two mountains as high as Everest, because nowhere is it stated that *BenNevis* is different from *Kilimanjaro* (or *Everest*, for that matter).

9.2 Unification and First-Order Inference

Exercise 9.2.#MGUX

For each pair of atomic sentences, give the most general unifier if it exists:

- a. $P(A, B, B), P(x, y, z)$.
- b. $Q(y, G(A, B)), Q(G(x, x), y)$.
- c. $\text{Older}(\text{Father}(y), y), \text{Older}(\text{Father}(x), \text{John})$.
- d. $\text{Knows}(\text{Father}(y), y), \text{Knows}(x, x)$.

This is an easy exercise to check that the student understands unification.

- a. $\{x/A, y/B, z/B\}$ (or some permutation of this).
- b. No unifier (x cannot bind to both A and B).
- c. $\{y/\text{John}, x/\text{John}\}$.
- d. No unifier (because the occurs-check prevents unification of y with $\text{Father}(y)$).

Exercise 9.2.#SUBL

Consider the subsumption lattices shown in Figure 9.2 (page 286).

- Construct the lattice for the sentence $\text{Employs}(\text{Mother}(\text{John}), \text{Father}(\text{Richard}))$.
- Construct the lattice for the sentence $\text{Employs}(\text{IBM}, y)$ (“Everyone works for IBM”). Remember to include every kind of query that unifies with the sentence.
- Assume that STORE indexes each sentence under every node in its subsumption lattice. Explain how FETCH should work when some of these sentences contain variables; use as examples the sentences in (a) and (b) and the query $\text{Employs}(x, \text{Father}(x))$.

- For the sentence $\text{Employs}(\text{Mother}(\text{John}), \text{Father}(\text{Richard}))$, the page isn’t wide enough to draw the diagram as in Figure 9.2, so we will draw it with indentation denoting children nodes:

```
[1] Employs(x, y)
[2] Employs(x, Father(z))
[3] Employs(x, Father(Richard))
[4] Employs(Mother(w), Father(Richard))
[5] Employs(Mother(John), Father(Richard))
[6] Employs(Mother(w), Father(z))
[4] ...
[7] Employs(Mother(John), Father(z))
[5] ...
[8] Employs(Mother(w), y)
[9] Employs(Mother(John), y)
[10] Employs(Mother(John), Father(z))
[5] ...
[6] ...
```

- For the sentence $\text{Employs}(\text{IBM}, y)$, the lattice contains $\text{Employs}(x, y)$ and $\text{Employs}(y, y)$.
- c.

Exercise 9.2.#FOLH

Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens:

- Horses, cows, and pigs are mammals.
- An offspring of a horse is a horse.
- Bluebeard is a horse.
- Bluebeard is Charlie’s parent.
- Offspring and parent are inverse relations.
- Every mammal has a parent.

We use a very simple ontology to make the examples easier:

- a. $Horse(x) \Rightarrow Mammal(x)$
 $Cow(x) \Rightarrow Mammal(x)$
 $Pig(x) \Rightarrow Mammal(x).$
- b. $Offspring(x, y) \wedge Horse(y) \Rightarrow Horse(x).$
- c. $Horse(Bluebeard).$
- d. $Parent(Bluebeard, Charlie).$
- e. $Offspring(x, y) \Rightarrow Parent(y, x)$
 $Parent(x, y) \Rightarrow Offspring(y, x).$
 (Note we couldn't do $Offspring(x, y) \Leftrightarrow Parent(y, x)$ because that is not in the form expected by Generalized Modus Ponens.)
- f. $Mammal(x) \Rightarrow Parent(G(x), x)$ (here G is a Skolem function).

Exercise 9.2.#CENS

Suppose we put into a logical knowledge base a segment of the U.S. census data listing the age, city of residence, date of birth, and mother of every person, using social security numbers as identifying constants for each person. Thus, George's age is given by $Age(443-65-1282, 56)$. Which of the following indexing schemes S1–S5 enable an efficient solution for which of the queries Q1–Q4 (assuming normal backward chaining)?

- **S1:** an index for each atom in each position.
- **S2:** an index for each first argument.
- **S3:** an index for each predicate atom.
- **S4:** an index for each *combination* of predicate and first argument.
- **S5:** an index for each *combination* of predicate and second argument and an index for each first argument.
- **Q1:** $Age(443-44-4321, x)$
- **Q2:** $ResidesIn(x, Houston)$
- **Q3:** $Mother(x, y)$
- **Q4:** $Age(x, 34) \wedge ResidesIn(x, TinyTownUSA)$

We will give the average-case time complexity for each query/scheme combination in the following table. (An entry of the form “1; n ” means that it is $O(1)$ to find the first solution to the query, but $O(n)$ to find them all.) We make the following assumptions: hash tables give $O(1)$ access; there are n people in the data base; there are $O(n)$ people of any specified age; every person has one mother; there are H people in Houston and T people in Tiny Town; T is much less than n ; in Q4, the second conjunct is evaluated first.

	Q1	Q2	Q3	Q4
S1	1	1; H	1; n	T; T
S2	1	n; n	1; n	n; n
S3	n	n; n	1; n	n ² ; n ²
S4	1	n; n	1; n	n; n
S5	1	1; H	1; n	T; T

Anything that is $O(1)$ can be considered “efficient,” as perhaps can anything $O(T)$. Note that S1 and S5 dominate the other schemes for this set of queries. Also note that indexing on predicates plays no role in this table (except in combination with an argument), because there are only 3 predicates (which is $O(1)$). It would make a difference in terms of the constant factor.

9.3 Forward Chaining

Exercise 9.3.#CCSP

Explain how to write any given 3-SAT problem of arbitrary size using a single first-order definite clause and no more than 30 atomic sentences.

Consider a 3-SAT problem of the form

$$(x_{1,1} \vee x_{2,1} \vee \neg x_{3,1}) \wedge (\neg x_{1,2} \vee x_{2,2} \vee x_{3,2}) \wedge \dots$$

We want to rewrite this as a single definite clause of the form

$$A \wedge B \wedge C \wedge \dots \Rightarrow Z,$$

along with a few ground clauses. We can do that with the definite clause

$$\text{OneOf}(x_{1,1}, x_{2,1}, \text{Not}(x_{3,1})) \wedge \text{OneOf}(\text{Not}(x_{1,2}), x_{2,2}, x_{3,2}) \wedge \dots \Rightarrow \text{Solved}.$$

The key is that any solution to the definite clause has to assign the same value to each occurrence of any given variable, even if the variable is negated in some of the SAT clauses but not others. We also need to define *OneOf*. This can be done concisely as follows:

$$\begin{aligned} &\text{OneOf}(\text{True}, x, y) \\ &\text{OneOf}(x, \text{True}, y) \\ &\text{OneOf}(x, y, \text{True}) \\ &\text{OneOf}(\text{Not}(\text{False}), x, y) \\ &\text{OneOf}(x, \text{Not}(\text{False}), y) \\ &\text{OneOf}(x, y, \text{Not}(\text{False})) \end{aligned}$$

9.4 Backward Chaining

Exercise 9.4.#HRKB

This question considers Horn KBs, such as the following:

$$\begin{aligned} P(F(x)) &\Rightarrow P(x) \\ Q(x) &\Rightarrow P(F(x)) \\ P(A) \\ Q(B) \end{aligned}$$

Let FC be a breadth-first forward-chaining algorithm that repeatedly adds all consequences of currently satisfied rules; let BC be a depth-first left-to-right backward-chaining algorithm that tries clauses in the order given in the KB. Which of the following are true?

- a. FC will infer the literal $Q(A)$.
 - b. FC will infer the literal $P(B)$.
 - c. If FC has failed to infer a given literal, then it is not entailed by the KB.
 - d. BC will return *true* given the query $P(B)$.
 - e. If BC does not return *true* given a query literal, then it is not entailed by the KB.
-
- a. False; $Q(A)$ is not entailed.
 - b. True; via $P(F(B))$.
 - c. True; breadth-first FC is complete for Horn KBs.
 - d. False; infinite loop applying the first rule repeatedly.
 - e. False; $P(b)$ is an example.

Exercise 9.4.#BCAR

Suppose you are given the following axioms:

1. $0 \leq 3$.
2. $7 \leq 9$.
3. $\forall x \quad x \leq x$.
4. $\forall x \quad x \leq x + 0$.
5. $\forall x \quad x + 0 \leq x$.
6. $\forall x, y \quad x + y \leq y + x$.
7. $\forall w, x, y, z \quad w \leq y \wedge x \leq z \Rightarrow w + x \leq y + z$.
8. $\forall x, y, z \quad x \leq y \wedge y \leq z \Rightarrow x \leq z$

- a. Give a backward-chaining proof of the sentence $7 \leq 3 + 9$. (Be sure, of course, to use only the axioms given here, not anything else you may know about arithmetic.) Show only the steps that leads to success, not the irrelevant steps.
- b. Give a forward-chaining proof of the sentence $7 \leq 3 + 9$. Again, show only the steps that lead to success.

This is quite tricky but students should be able to manage if they check each step carefully.

- a. (Note: At each resolution, we rename the variables in the rule).

Goal G0: $7 \leq 3 + 9$	Resolve with (8) $\{x1/7, z1/3 + 9\}$.
Goal G1: $7 \leq y1$	Resolve with (4) $\{x2/7, y1/7 + 0\}$. Succeeds.
Goal G2: $7 + 0 \leq 3 + 9$.	Resolve with (8) $\{x3/7 + 0, z3/3 + 9\}$
Goal G3: $7 + 0 \leq y3$	Resolve with (6) $\{x4/7, y4/0, y3/0 + 7\}$ Succeeds.
Goal G4: $0 + 7 \leq 3 + 9$	Resolve with (7) $\{w5/0, x5/7, y5/3, z5/9\}$.
Goal G5: $0 \leq 3$.	Resolve with (1). Succeeds.
Goal G6: $7 \leq 9$.	Resolve with (2). Succeeds.
G4 succeeds	
G2 succeeds.	
G0 succeeds.	
b. From (1),(2), (7) $\{w/0, x/7, y/3, z/9\}$ infer	
(9) $0 + 7 \leq 3 + 9$.	
From (9), (6), (8) $\{x1/0, y1/7, x2/0 + 7, y2/7 + 0, z2/3 + 9\}$ infer	
(10) $7 + 0 \leq 3 + 9$.	
($x1, y1$ are renamed variables in (6). $x2, y2, z2$ are renamed variables in (8).)	
From (4), (10), (8) $\{x3/7, x4/7, y4/7 + 0, z4/3 + 9\}$ infer	
(11) $7 \leq 3 + 9$.	
($x3$ is a renamed variable in (4). $x4, y4, z4$ are renamed variables in (8).)	

Exercise 9.4.#STDF

One might suppose that we can avoid the problem of variable conflict in unification during backward chaining by standardizing apart all of the sentences in the knowledge base once and for all. Show that, for some sentences, this approach cannot work. (*Hint:* Consider a sentence in which one part unifies with another.)

This would work if there were no recursive rules in the knowledge base. But suppose the knowledge base contains the sentences:

$$\begin{aligned} &Member(x, [x|r]) \\ &Member(x, r) \Rightarrow Member(x, [y|r]) \end{aligned}$$

Now take the query $Member(3, [1, 2, 3])$, with a backward chaining system. We unify the query with the consequent of the implication to get the substitution $\theta = \{x/3, y/1, r/[2, 3]\}$. We then substitute this in to the left-hand side to get $Member(3, [2, 3])$ and try to back chain on that with the substitution θ . When we then try to apply the implication again, we get a failure because y cannot be both 1 and 2. In other words, the failure to standardize apart causes failure in some cases where recursive rules would result in a solution if we did standardize apart.

Exercise 9.4.#HORB

In this exercise, use the sentences you wrote in Exercise 9.FOLH to answer a question by using a backward-chaining algorithm.

- a. Draw the proof tree generated by an exhaustive backward-chaining algorithm for the query $\exists h \ Horse(h)$, where clauses are matched in the order given.
- b. What do you notice about this domain?
- c. How many solutions for h actually follow from your sentences?
- d. Can you think of a way to find all of them? (*Hint:* See Smith *et al.* (1986).)

This questions deals with the subject of looping in backward-chaining proofs. A loop is bound to occur whenever a subgoal arises that is a substitution instance of one of the goals on the stack. Not all loops can be caught this way, of course, otherwise we would have a way to solve the halting problem.

- a. The proof tree is shown in Figure S9.1. The branch with $Offspring(Bluebeard, y)$ and $Parent(y, Bluebeard)$ repeats indefinitely, so the rest of the proof is never reached.
- b. We get an infinite loop because of rule b, $Offspring(x, y) \wedge Horse(y) \Rightarrow Horse(x)$. The specific loop appearing in the figure arises because of the ordering of the clauses—it would be better to order $Horse(Bluebeard)$ before the rule from b. However, a loop will occur no matter which way the rules are ordered if the theorem-prover is asked for all solutions.
- c. One should be able to prove that both Bluebeard and Charlie are horses.
- d. Smith *et al.* 1986 recommend the following method. Whenever a “looping” goal occurs (one that is a substitution instance of a supergoal higher up the stack), suspend the attempt to prove that subgoal. Continue with all other branches of the proof for the supergoal, gathering up the solutions. Then use those solutions (suitably instantiated if necessary) as solutions for the suspended subgoal, continuing that branch of the proof to find additional solutions if any. In the proof shown in the figure, the $Offspring(Bluebeard, y)$ is a repeated goal and would be suspended. Since no other way to prove it exists, that branch will terminate with failure. In this case, Smith’s method is sufficient to allow the theorem-prover to find both solutions.

Exercise 9.4.#BCCR

Trace the execution of the backward-chaining algorithm in Figure 9.6 (page 293) when it is applied to solve the crime problem (page 287). Show the sequence of values taken on by the *goals* variable, and arrange them into a tree.

Here is a goal tree:

```
goals = [Criminal(West)]
goals = [American(West), Weapon(y), Sells(West, y, z), Hostile(z)]
goals = [Weapon(y), Sells(West, y, z), Hostile(z)]
```

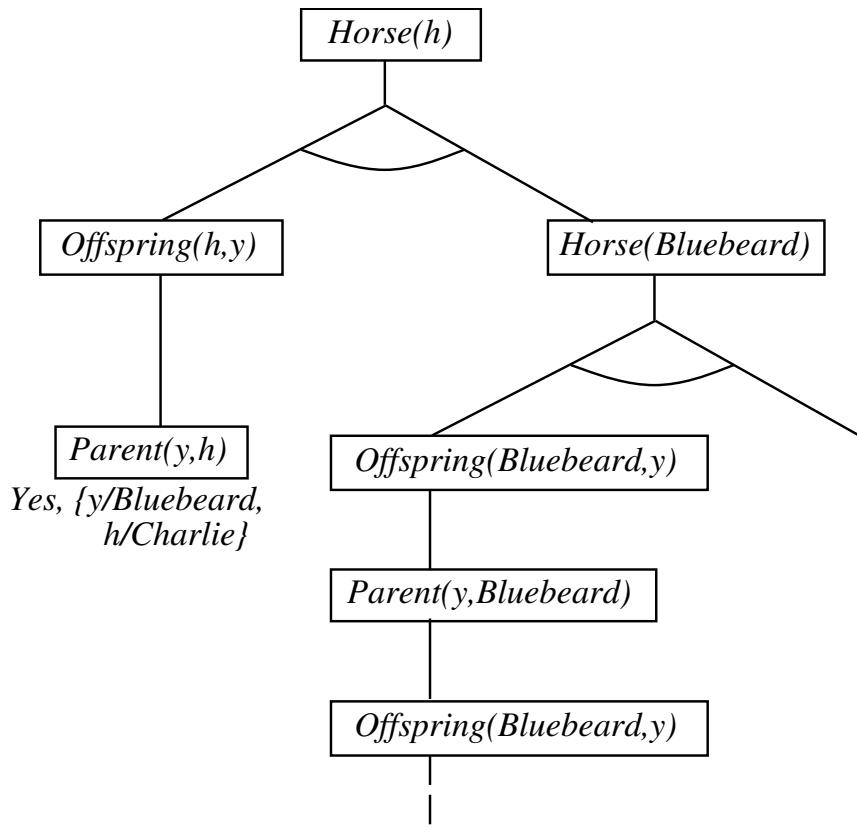


Figure S9.1 Partial proof tree for finding horses.

```

goals = [Missle(y), Sells(West, y, z), Hostile(z)]
goals = [Sells(West, M1, z), Hostile(z)]
    goals = [Missle(M1), Owns(Nono, M1), Hostile(Nono)]
    goals = [Owns(Nono, M1), Hostile(Nono)]
    goals = [Hostile(Nono)]
goals = []
  
```

Exercise 9.4.#PRLC

The following Prolog code defines a predicate P . (Remember that uppercase terms are variables, not constants, in Prolog.)

```

P(X, [X|Y]) .
P(X, [Y|Z]) :- P(X, Z) .
  
```

- Show proof trees and solutions for the queries $P(A, [2, 1, 3])$ and $P(2, [1, A, 3])$.
- What standard list operation does P represent?

- a. In the following, an indented line is a step deeper in the proof tree, while two lines at the same indentation represent two alternative ways to prove the goal that is unindented above it. P1 and P2 refer to the first and second clauses of the definition respectively. We show each goal as it is generated and the result of unifying it with the head of each clause.

```

P(A, [2,1,3])           goal
P(2, [2|[1,3]])         unify with head of P1
=> solution, with A = 2

P(A, [2|[1,3]])         unify with head of P2
P(A, [1,3])              subgoal
P(1, [1,3])              unify with head of P1
=> solution, with A = 1

P(A, [1|[3]])           unify with head of P2
P(A, [3])                subgoal
P(3, [3|[]])            unify with head of P1
=> solution, with A = 3

P(A, [3|[]])            unify with head of P2
P(A, [])                 subgoal (fails)

P(2, [1,A,3])           goal
P(2, [1|[A,3]])         unify with head of P2
P(2, [A,3])              subgoal
P(2, [2,3])              unify with head of P1
=> solution, with A = 2

P(2, [A|[3]])           unify with head of P2
P(2, [3])                subgoal
P(2, [3|[]])            unify with head of P2
P(2, [])                 subgoal

```

- b. P could better be called Member; it succeeds when the first argument is an element of the list that is the second argument.

Exercise 9.4.#PRLS

This exercise looks at sorting in Prolog.

- Write Prolog clauses that define the predicate `sorted(L)`, which is true if and only if list L is sorted in ascending order.
- Write a Prolog definition for the predicate `perm(L,M)`, which is true if and only if L is a permutation of M.
- Define `sort(L,M)` (M is a sorted version of L) using `perm` and `sorted`.
- Run `sort` on longer and longer lists until you lose patience. What is the time complexity of your program?
- Write a faster sorting algorithm, such as insertion sort or quicksort, in Prolog.

The different versions of `sort` illustrate the distinction between logical and procedural semantics in Prolog.

- `sorted([]).`
`sorted([X]).`

```

sorted([X,Y|L]) :- X<Y, sorted([Y|L]).  

b. perm([],[]).  

perm([X|L],M) :-  

    delete(X,M,M1),  

    perm(L,M1).  

delete(X,[X|L],L).           %% deleting an X from [X|L] yields L  

delete(X,[Y|L],[Y|M]) :- delete(X,L,M).  

member(X,[X|L]).  

member(X,[_|L]) :- member(X,L).  

c. sort(L,M) :- perm(L,M), sorted(M).

```

This is about as close to an executable formal specification of sorting as you can get—it says the absolute minimum about what sort means: in order for M to be a sort of L, it must have the same elements as L, and they must be in order.

- d.** Unfortunately, this doesn't fare as well as a program as it does as a specification. It is a generate-and-test sort: perm generates candidate permutations one at a time, and sorted tests them. In the worst case (when there is only one sorted permutation, and it is the last one generated), this will take $O(n!)$ generations. Since each perm is $O(n^2)$ and each sorted is $O(n)$, the whole sort is $O(n!n^2)$ in the worst case.
- e.** Here's a simple insertion sort, which is $O(n^2)$:

```

isort([],[]).  

isort([X|L],M) :- isort(L,M1), insert(X,M1,M).  

insert(X,[],[X]).  

insert(X,[Y|L],[X,Y|L]) :- X=<Y.  

insert(X,[Y|L],[Y|M]) :- Y<X, insert(X,L,M).

```

Exercise 9.4.#DFSM

This exercise looks at the recursive application of rewrite rules, using logic programming. A rewrite rule (or **demodulator** in OTTER terminology) is an equation with a specified direction. For example, the rewrite rule $x + 0 \rightarrow x$ suggests replacing any expression that matches $x + 0$ with the expression x . Rewrite rules are a key component of equational reasoning systems. Use the predicate `rewrite(X, Y)` to represent rewrite rules. For example, the earlier rewrite rule is written as `rewrite(X+0, X)`. Some terms are *primitive* and cannot be further simplified; thus, we write `primitive(0)` to say that 0 is a primitive term.

- Write a definition of a predicate `simplify(X, Y)`, that is true when Y is a simplified version of X—that is, when no further rewrite rules apply to any subexpression of Y.
- Write a collection of rules for the simplification of expressions involving arithmetic operators, and apply your simplification algorithm to some sample expressions.
- Write a collection of rewrite rules for symbolic differentiation, and use them along with your simplification rules to differentiate and simplify expressions involving arithmetic expressions, including exponentiation.

This exercise illustrates the power of pattern-matching, which is built into Prolog.

- a. The code for simplification looks straightforward, but students may have trouble finding the middle way between undersimplifying and looping infinitely.

```
simplify(X, X) :- primitive(X).
simplify(X, Y) :- evaluable(X), Y is X.
simplify(Op(X)) :- simplify(X, X1), simplify_exp(Op(X1)).
simplify(Op(X, Y)) :- simplify(X, X1), simplify(Y, Y1), simplify_exp(Op(X1, Y1)).

simplify_exp(X, Y) :- rewrite(X, X1), simplify(X1, Y).
simplify_exp(X, X).
```

```
primitive(X) :- atom(X).
```

- b. Here are a few representative rewrite rules drawn from the extensive list in Norvig (1992).

```
Rewrite(X+0, X).
Rewrite(0+X, X).
Rewrite(X+X, 2*X).
Rewrite(X*X, X^2).
Rewrite(X^0, 1).
Rewrite(0^X, 0).
Rewrite(X*N, N*X) :- number(N).
Rewrite(ln(e^X), X).
Rewrite(X^Y*X^Z, X^(Y+Z)).
Rewrite(sin(X)^2+cos(X)^2, 1).
```

- c. Here are the rules for differentiation, using $d(Y, X)$ to represent the derivative of expression Y with respect to variable X .

```
Rewrite(d(X, X), 1).
Rewrite(d(U, X), 0) :- atom(U), U /= X.
Rewrite(d(U+V, X), d(U, X)+d(V, X)).
Rewrite(d(U-V, X), d(U, X)-d(V, X)).
Rewrite(d(U*V, X), V*d(U, X)+U*d(V, X)).
Rewrite(d(U/V, X), (V*d(U, X)-U*d(V, X))/(V^2)).
Rewrite(d(U^N, X), N*U^(N-1)*d(U, X)) :- number(N).
Rewrite(d(log(U), X), d(U, X)/U).
Rewrite(d(sin(U), X), cos(U)*d(U, X)).
Rewrite(d(cos(U), X), -sin(U)*d(U, X)).
Rewrite(d(e^U, X), d(U, X)*e^U).
```

Exercise 9.4.#PRSE

This exercise considers the implementation of search algorithms in Prolog. Suppose that `successor(X, Y)` is true when state Y is a successor of state X ; and that `goal(X)` is true when X is a goal state. Write a definition for `solve(X, P)`, which means that P is a path (list of states) beginning with X , ending in a goal state, and consisting of a sequence of legal steps as defined by `successor`. You will find that depth-first search is the easiest way to do this. How easy would it be to add heuristic search control?

Once you understand how Prolog works, the answer is easy:

```
solve(X, [X]) :- goal(X).
solve(X, [X|P]) :- successor(X, Y), solve(Y, P).
```

We could render this in English as “Given a start state, if it is a goal state, then the path consisting of just the start state is a solution. Otherwise, find some successor state such that

there is a path from the successor to the goal; then a solution is the start state followed by that path.”

Notice that `solve` can not only be used to find a path P that is a solution, it can also be used to verify that a given path is a solution.

If you want to add heuristics (or even breadth-first search), you need an explicit queue. The algorithms become quite similar to the versions written in Lisp or Python or Java or in pseudo-code in the book.

9.5 Resolution

Exercise 9.5.#SKOL

These questions concern concern issues with substitution and Skolemization.

- Given the premise $\forall x \exists y P(x, y)$, it is not valid to conclude that $\exists q P(q, q)$. Give an example of a predicate P where the first is true but the second is false.
- Suppose that an inference engine is incorrectly written with the occurs check omitted, so that it allows a literal like $P(x, F(x))$ to be unified with $P(q, q)$. (As mentioned, most standard implementations of Prolog actually do allow this.) Show that such an inference engine will allow the conclusion $\exists y P(q, q)$ to be inferred from the premise $\forall x \exists y P(x, y)$.
- Suppose that a procedure that converts first-order logic to clausal form incorrectly Skolemizes $\forall x \exists y P(x, y)$ to $P(x, Sk0)$ —that is, it replaces y by a Skolem constant rather than by a Skolem function of x . Show that an inference engine that uses such a procedure will likewise allow $\exists q P(q, q)$ to be inferred from the premise $\forall x \exists y P(x, y)$.
- A common error among students is to suppose that, in unification, one is allowed to substitute a term for a Skolem constant instead of for a variable. For instance, they will say that the formulas $P(Sk1)$ and $P(A)$ can be unified under the substitution $\{Sk1/A\}$. Give an example where this leads to an invalid inference.

- Let $P(x, y)$ be the relation “ x is less than y ” over the integers. Then $\forall x \exists y P(x, y)$ is true but $\exists x P(x, x)$ is false.
- Converting the premise to clausal form gives $P(x, Sk0(x))$ and converting the negated goal to clausal form gives $\neg P(q, q)$. If the two formulas can be unified, then these resolve to the null clause.
- If the premise is represented as $P(x, Sk0)$ and the negated goal has been correctly converted to the clause $\neg P(q, q)$ then these can be resolved to the null clause under the substitution $\{q/Sk0, x/Sk0\}$.
- Suppose you are given the premise $\exists x Cat(x)$ and you wish to prove $Cat(Socrates)$. Converting the premise to clausal form gives the clause $Cat(Sk1)$. If this unifies with $Cat(Socrates)$ then you can resolve this with the negated goal $\neg Cat(Socrates)$ to give the null clause.

Exercise 9.5.#BRSI

A popular children's riddle is "Brothers and sisters have I none, but that man's father is my father's son." Use the rules of the family domain (Section 8.3.2 on page 266) to show who that man is. You may apply any of the inference methods described in this chapter. Why do you think that this riddle is difficult?

Surprisingly, the hard part to represent is "who is that man." We want to ask "what relationship does that man have to some known person," but if we represent relations with predicates (e.g., $\text{Parent}(x, y)$) then we cannot make the relationship be a variable in first-order logic. So instead we need to reify relationships. We will use $\text{Rel}(r, x, y)$ to say that the family relationship r holds between people x and y . Let Me denote me and MrX denote "that man." We will also need the Skolem constants FM for the father of Me and FX for the father of MrX . The facts of the case (put into implicative normal form) are:

- (1) $\text{Rel}(\text{Sibling}, Me, x) \Rightarrow \text{False}$
- (2) $\text{Male}(MrX)$
- (3) $\text{Rel}(\text{Father}, FX, MrX)$
- (4) $\text{Rel}(\text{Father}, FM, Me)$
- (5) $\text{Rel}(\text{Son}, FX, FM)$

We want to be able to show that Me is the only son of my father, and therefore that Me is father of MrX , who is male, and therefore that "that man" is my son. The relevant definitions from the family domain are:

- (6) $\text{Rel}(\text{Parent}, x, y) \wedge \text{Male}(x) \Leftrightarrow \text{Rel}(\text{Father}, x, y)$
- (7) $\text{Rel}(\text{Son}, x, y) \Leftrightarrow \text{Rel}(\text{Parent}, y, x) \wedge \text{Male}(x)$
- (8) $\text{Rel}(\text{Sibling}, x, y) \Leftrightarrow x \neq y \wedge \exists p \text{ } \text{Rel}(\text{Parent}, p, x) \wedge \text{Rel}(\text{Parent}, p, y)$
- (9) $\text{Rel}(\text{Father}, x_1, y) \wedge \text{Rel}(\text{Father}, x_2, y) \Rightarrow x_1 = x_2$

and the query we want is:

$$(Q) \text{ } \text{Rel}(r, MrX, y)$$

We want to be able to get back the answer $\{r/Son, y/Me\}$. Translating 1-9 and Q into INF

(and negating Q and including the definition of \neq) we get:

- (6a) $Rel(Parent, x, y) \wedge Male(x) \Rightarrow Rel(Father, x, y)$
- (6b) $Rel(Father, x, y) \Rightarrow Male(x)$
- (6c) $Rel(Father, x, y) \Rightarrow Rel(Parent, x, y)$
- (7a) $Rel(Son, x, y) \Rightarrow Rel(Parent, y, x)$
- (7b) $Rel(Son, x, y) \Rightarrow Male(x)$
- (7c) $Rel(Parent, y, x) \wedge Male(x) \Rightarrow Rel(Son, x, y)$
- (8a) $Rel(Sibling, x, y) \Rightarrow x \neq y$
- (8b) $Rel(Sibling, x, y) \Rightarrow Rel(Parent, P(x, y), x)$
- (8c) $Rel(Sibling, x, y) \Rightarrow Rel(Parent, P(x, y), y)$
- (8d) $Rel(Parent, P(x, y), x) \wedge Rel(Parent, P(x, y), y) \wedge x \neq y \Rightarrow Rel(Sibling, x, y)$
- (9) $Rel(Father, x_1, y) \wedge Rel(Father, x_2, y) \Rightarrow x_1 = x_2$
- (N) $True \Rightarrow x = y \vee x \neq y$
- (N') $x = y \wedge x \neq y \Rightarrow False$
- (Q') $Rel(r, MrX, y) \Rightarrow False$

Note that (1) is non-Horn, so we will need resolution to be sure of getting a solution. It turns out we also need demodulation to deal with equality. The following lists the steps of the proof, with the resolvents of each step in parentheses:

- | | |
|--|-------------------------------|
| (10) $Rel(Parent, FM, Me)$ | (4, 6c) |
| (11) $Rel(Parent, FM, FX)$ | (5, 7a) |
| (12) $Rel(Parent, FM, y) \wedge Me \neq y \Rightarrow Rel(Sibling, Me, y)$ | (10, 8d) |
| (13) $Rel(Parent, FM, y) \wedge Me \neq y \Rightarrow False$ | (12, 1) |
| (14) $Me \neq FX \Rightarrow False$ | (13, 11) |
| (15) $Me = FX$ | (14, N) |
| (16) $Rel(Father, Me, MrX)$ | (15, 3, <i>demodulation</i>) |
| (17) $Rel(Parent, Me, MrX)$ | (16, 6c) |
| (18) $Rel(Son, MrX, Me)$ | (17, 2, 7c) |
| (19) $False \{r/Son, y/Me\}$ | (18, Q') |

Exercise 9.5.#ANCH

Suppose a knowledge base contains just the following first-order Horn clauses:

$$\begin{aligned} & Ancestor(Mother(x), x) \\ & Ancestor(x, y) \wedge Ancestor(y, z) \Rightarrow Ancestor(x, z) \end{aligned}$$

Consider a forward chaining algorithm that, on the j th iteration, terminates if the KB contains a sentence that unifies with the query, else adds to the KB every atomic sentence that can be inferred from the sentences already in the KB after iteration $j - 1$.

- For each of the following queries, say whether the algorithm will (1) give an answer (if so, write down that answer); or (2) terminate with no answer; or (3) never terminate.

- (i) $\text{Ancestor}(\text{Mother}(y), \text{John})$
 - (ii) $\text{Ancestor}(\text{Mother}(\text{Mother}(y)), \text{John})$
 - (iii) $\text{Ancestor}(\text{Mother}(\text{Mother}(\text{Mother}(y))), \text{Mother}(y))$
 - (iv) $\text{Ancestor}(\text{Mother}(\text{John}), \text{Mother}(\text{Mother}(\text{John})))$
 - b.** Can a resolution algorithm prove the sentence $\neg \text{Ancestor}(\text{John}, \text{John})$ from the original knowledge base? Explain how, or why not.
 - c.** Suppose we add the assertion that $\neg(\text{Mother}(x) = x)$ and augment the resolution algorithm with inference rules for equality. Now what is the answer to (b)?
- a.** Results from forward chaining:
- (i) $\text{Ancestor}(\text{Mother}(y), \text{John})$: Yes, $\{y/\text{John}\}$ (immediate).
 - (ii) $\text{Ancestor}(\text{Mother}(\text{Mother}(y)), \text{John})$: Yes, $\{y/\text{John}\}$ (second iteration).
 - (iii) $\text{Ancestor}(\text{Mother}(\text{Mother}(\text{Mother}(y))), \text{Mother}(y))$: Yes, $\{\}$ (second iteration).
 - (iv) $\text{Ancestor}(\text{Mother}(\text{John}), \text{Mother}(\text{Mother}(\text{John})))$: Does not terminate.

- b.** Although resolution is complete, it cannot prove this because it does not follow. Nothing in the axioms rules out the possibility of everything being the ancestor of everything else.
- c.** Same answer.

Exercise 9.5.#SHAV

Let \mathcal{L} be the first-order language with a single predicate $S(p, q)$, meaning “ p shaves q .” Assume a domain of people.

- a.** Consider the sentence “There exists a person P who shaves every one who does not shave themselves, and only people that do not shave themselves.” Express this in \mathcal{L} .
- b.** Convert the sentence in (a) to clausal form.
- c.** Construct a resolution proof to show that the clauses in (b) are inherently inconsistent. (Note: you do not need any additional axioms.)

- a.** $\exists p \forall q S(p, q) \Leftrightarrow \neg S(q, q)$.
- b.** There are two clauses, corresponding to the two directions of the implication.
 C1: $\neg S(Sk1, q) \vee \neg S(q, q)$.
 C2: $S(Sk1, q) \vee S(q, q)$.
- c.** Applying factoring to C1, using the substitution $q/Sk1$ gives:
 C3: $\neg S(Sk1, Sk1)$.
 Applying factoring to C2, using the substitution $q/Sk1$ gives:
 C4: $S(Sk1, Sk1)$.
 Resolving C3 with C4 gives the null clause.

Exercise 9.5.#RESV

How can resolution be used to show that a sentence is valid? Unsatisfiable?

This question tests both the student's understanding of resolution and their ability to think at a high level about relations among sets of sentences. Recall that resolution allows one to show that $KB \models \alpha$ by proving that $KB \wedge \neg\alpha$ is inconsistent. Suppose that in general the resolution system is called using $\text{ASK}(KB, \alpha)$. Now we want to show that a given sentence, say β is valid or unsatisfiable.

A sentence β is valid if it can be shown to be true without additional information. We check this by calling $\text{ASK}(KB_0, \beta)$ where KB_0 is the empty knowledge base.

A sentence β that is unsatisfiable is inconsistent by itself. So if we empty the knowledge base again and call $\text{ASK}(KB_0, \neg\beta)$ the resolution system will attempt to derive a contradiction starting from $\neg\neg\beta$. If it can do so, then it must be that $\neg\neg\beta$, and hence β , is inconsistent.

Exercise 9.5.#REST

Construct an example of two clauses that can be resolved together in two different ways giving two different outcomes.

There are two ways to do this: one literal in one clause that is complementary to two different literals in the other, such as

$$P(x) \quad \neg P(a) \vee \neg P(b)$$

or two complementary pairs of literals, such as

$$P(x) \vee Q(x) \quad \neg P(a) \vee \neg Q(b).$$

Note that this does not work in propositional logic: in the first case, the two literals in the second clause would have to be identical; in the second case, the remaining unresolved complementary pair after resolution would render the result a tautology.

Exercise 9.5.#HORA

From "Horses are animals," it follows that "The head of a horse is the head of an animal." Demonstrate that this inference is valid by carrying out the following steps:

- Translate the premise and the conclusion into the language of first-order logic. Use three predicates: $\text{HeadOf}(h, x)$ (meaning " h is the head of x "), $\text{Horse}(x)$, and $\text{Animal}(x)$.
- Negate the conclusion, and convert the premise and the negated conclusion into conjunctive normal form.
- Use resolution to show that the conclusion follows from the premise.

This is a form of inference used to show that Aristotle's syllogisms could not capture all

sound inferences.

- a. $\forall x \ Horse(x) \Rightarrow Animal(x)$
 $\forall x, h \ Horse(x) \wedge HeadOf(h, x) \Rightarrow \exists y \ Animal(y) \wedge HeadOf(h, y)$

- b. A. $\neg Horse(x) \vee Animal(x)$
B. $Horse(G)$
C. $HeadOf(H, G)$
D. $\neg Animal(y) \vee \neg HeadOf(H, y)$

(Here A. comes from the first sentence in a. while the others come from the second. H and G are Skolem constants.)

- c. Resolve D and C to yield $\neg Animal(G)$. Resolve this with A to give $\neg Horse(G)$. Resolve this with B to obtain a contradiction.

Exercise 9.5.#QUOR

Here are two sentences in the language of first-order logic:

- (A) $\forall x \ \exists y \ (x \geq y)$
- (B) $\exists y \ \forall x \ (x \geq y)$

- a. Assume that the variables range over all the natural numbers $0, 1, 2, \dots, \infty$ and that the “ \geq ” predicate means “is greater than or equal to.” Under this interpretation, translate (A) and (B) into English.
- b. Is (A) true under this interpretation?
- c. Is (B) true under this interpretation?
- d. Does (A) logically entail (B)?
- e. Does (B) logically entail (A)?
- f. Using resolution, try to prove that (A) follows from (B). Do this even if you think that (B) does not logically entail (A); continue until the proof breaks down and you cannot proceed (if it does break down). Show the unifying substitution for each resolution step. If the proof fails, explain exactly where, how, and why it breaks down.
- g. Now try to prove that (B) follows from (A).

This exercise tests the students’ understanding of models and implication.

- a. (A) translates to “For every natural number there is some other natural number that is smaller than or equal to it.” (B) translates to “There is a particular natural number that is smaller than or equal to any natural number.”
- b. Yes, (A) is true under this interpretation. You can always pick the number itself for the “some other” number.
- c. Yes, (B) is true under this interpretation. You can pick 0 for the “particular natural number.”
- d. No, (A) does not logically entail (B).
- e. Yes, (B) logically entails (A).
- f. We want to try to prove via resolution that (A) entails (B). To do this, we set our knowledge base to consist of (A) and the negation of (B), which we will call $(\neg B)$, and try to

derive a contradiction. First we have to convert (A) and (-B) to canonical form. For (-B), this involves moving the \neg in past the two quantifiers. For both sentences, it involves introducing a Skolem function:

- (A) $x \geq F_1(x)$
- (-B) $\neg F_2(y) \geq y$

Now we can try to resolve these two together, but the occurs check rules out the unification. It looks like the substitution should be $\{x/F_2(y), y/F_1(x)\}$, but that is equivalent to $\{x/F_2(y), y/F_1(F_2(y))\}$, which fails because y is bound to an expression containing y . So the resolution fails, there are no other resolution steps to try, and therefore (B) does not follow from (A).

- g. To prove that (B) entails (A), we start with a knowledge base containing (B) and the negation of (A), which we will call (-A):

- (-A) $\neg F_1 \geq y$
- (B) $x \geq F_2$

This time the resolution goes through, with the substitution $\{x/F_1, y/F_2\}$, thereby yielding *False*, and proving that (B) entails (A).

Exercise 9.5.#RSNC

Resolution can produce nonconstructive proofs for queries with variables, so we had to introduce special mechanisms to extract definite answers. Explain why this issue does not arise with knowledge bases containing only definite clauses.

One way of seeing this is that resolution allows reasoning by cases, by which we can prove C by proving that either A or B is true, without knowing which one. If the query contains a variable, we cannot prove that any *particular* instantiation gives a fact that is entailed. With definite clauses, we always have a single chain of inference, for which we can follow the chain and instantiate variables; the solution is always a single MGU.

Exercise 9.5.#ALLC

We said in this chapter that resolution cannot be used to generate all logical consequences of a set of sentences. Can any algorithm do this?

Not exactly. Part of the definition of algorithm is that it must terminate. Since there can be an infinite number of consequences of a set of sentences, no algorithm can generate them all. Another way to see that the answer is no is to remember that entailment for FOL is semidecidable. If there were an algorithm that generates the set of consequences of a set of sentences S , then when given the task of deciding if B is entailed by S , one could just check if B is in the generated set. But we know that this is not possible, therefore generating the set of sentences is impossible.

If we relax the definition of “algorithm” to allow for programs that *enumerate* the consequences, in the same sense that a program can enumerate the natural numbers by printing them out in order, the answer is yes. For example, we can enumerate them in order of the deepest allowable nesting of terms in the proof.

EXERCISES 10

KNOWLEDGE REPRESENTATION

10.1 Ontological Engineering

[[need exercises]]

10.2 Categories and Objects

Exercise 10.2.#OTTT

Define an ontology in first-order logic for tic-tac-toe. The ontology should contain situations, actions, squares, players, marks (X, O, or blank), and the notion of winning, losing, or drawing a game. Also define the notion of a forced win (or draw): a position from which a player can force a win (or draw) with the right sequence of actions. Write axioms for the domain. (Note: The axioms that enumerate the different squares and that characterize the winning positions are rather long. You need not write these out in full, but indicate clearly what they look like.)

Sortal predicates:

$Player(p)$

$Mark(m)$

$Square(q)$

Constants:

Xp, Op : Players.

$X, O, Blank$: Marks.

$Q11, Q12 \dots Q33$: Squares.

$S0$: Situation.

Atemporal:

$MarkOf(p)$: Function mapping player p to his/her mark.

$Winning(q1, q2, q3)$: Predicate. Squares $q1, q2, q3$ constitute a winning position.

$Opponent(p)$: Function mapping player p to his opponent.

Situation Calculus:

$Result(a, s)$.

$Poss(a, s)$.

State:

$TurnAt(s)$: Function mapping situation s to the player whose turn it is.

$Marked(q, s)$: Function mapping square q and situation s to the mark in q at s .

$\text{Wins}(p, s)$. Player p has won in situation s .

Action:

$\text{Play}(p, q)$: Function mapping player p and square q to the action of p marking q .

Atemporal axioms:

A1. $\text{MarkOf}(Xp) = X$.

A2. $\text{MarkOf}(Op) = O$.

A3. $\text{Opponent}(Xp) = Op$.

A4. $\text{Opponent}(Op) = Xp$.

A5. $\forall p \text{ Player}(p) \Leftrightarrow p = Xp \vee p = Op$.

A6. $\forall m \text{ Mark}(m) \Leftrightarrow m = X \vee m = O \vee m = \text{Blank}$.

A7. $\forall q \text{ Square}(q) \Leftrightarrow q = Q11 \vee q = Q12 \vee \dots \vee q = Q33$.

A8. $\forall q1, q2, q3 \text{ WinningPosition}(q1, q2, q3) \Leftrightarrow$

$$[q1 = Q11 \wedge q2 = Q12 \wedge q3 = Q13] \vee$$

$$[q1 = Q21 \wedge q2 = Q22 \wedge q3 = Q23] \vee$$

... (Similarly for the other six winning positions) \vee

$$[q1 = Q31 \wedge q2 = Q22 \wedge q3 = Q13]$$
.

Definition of winning:

A9. $\forall p, s \text{ Wins}(p, s) \Leftrightarrow$

$\exists q1, q2, q3 \text{ WinningPosition}(q1, q2, q3) \wedge$

$\text{MarkAt}(q1, s) = \text{MarkAt}(q2, s) = \text{MarkAt}(q3, s) = \text{MarkOf}(p)$

Causal Axioms:

A10. $\forall p, q \text{ Player}(p) \wedge \text{Square}(q) \Rightarrow$

$\text{MarkAt}(q, \text{Result}(\text{Play}(p, q), s)) = \text{MarkOf}(p)$.

A11. $\forall p, a, s \text{ TurnAt}(p, s) \Rightarrow \text{TurnAt}(\text{Opponent}(p), \text{Result}(a, s))$.

Precondition Axiom:

A12. $\text{Poss}(\text{Play}(p, q), s) \Rightarrow \text{TurnAt}(s) = p \wedge \text{MarkAt}(q, s) = \text{Blank}$.

Frame Axiom: A13. $q1 \neq q2 \Rightarrow \text{MarkAt}(q1, \text{Result}(\text{Play}(p, q2), s)) = \text{MarkAt}(q1, s)$.

Unique names:

A14. $X \neq O \neq \text{Blank}$.

(Note: the unique property on players $Xp \neq Op$ follows from A14, A1, and A2.)

A15-A50. For each i, j, k, m between 1 and 3 such that either $i \neq k$ or $j \neq m$ assert the axiom $Qij \neq Qkm$.

Note: In many theories it is useful to posit unique names axioms between entities of different sorts e.g. $\forall p, q \text{ Player}(p) \wedge \text{Square}(q) \Rightarrow p \neq q$. In this theory these are not actually necessary; if you want to imagine a circumstance in which player Xp is actually the same entity as square $Q23$ or the same as the action $\text{Play}(Xp, Q23)$ there is no harm in it.

Exercise 10.2.#CSUG

You are to create a system for advising computer science undergraduates on what courses to take over an extended period in order to satisfy the program requirements. (Use whatever requirements are appropriate for your institution.) First, decide on a vocabulary for repre-

senting all the information, and then represent it; then formulate a query to the system that will return a legal program of study as a solution. You should allow for some tailoring to individual students, in that your system should ask what courses or equivalents the student has already taken, and not generate programs that repeat those courses.

Suggest ways in which your system could be improved—for example to take into account knowledge about student preferences, the workload, good and bad instructors, and so on. For each kind of knowledge, explain how it could be expressed logically. Could your system easily incorporate this information to find all feasible programs of study for a student? Could it find the *best* program?

Most schools distinguish between required courses and elected courses, and between courses inside the department and outside the department. For each of these, there may be requirements for the number of courses, the number of units (since different courses may carry different numbers of units), and on grade point averages. We show our chosen vocabulary by example:

- Student Jones' complete course of study for the whole college career consists of Math1, CS1, CS2, CS3, CS21, CS33 and CS34, and some other courses outside the major.

$$\begin{aligned} & \text{Take}(Jones, \\ & \quad \{Math1, EE1, Bio24, CS1, CS2, CS3, CS21, CS33, CS34 | others\}) \end{aligned}$$

- Jones meets the requirements for a major in Computer Science

$$\text{Major}(Jones, CS)$$

- Courses Math1, CS1, CS2, and CS3 are required for a Computer Science major.

$$\begin{aligned} & \text{Required}(\{Math1, CS1, CS2, CS3\}, CS) \\ & \forall s, d \text{ Required}(s, d) \Leftrightarrow \\ & \quad (\forall p \exists others \text{ Major}(p, d) \Rightarrow \text{Take}(p, \text{Union}(s, others))) \end{aligned}$$

- A student must take at least 18 units in the CS department to get a degree in CS.

$$\begin{aligned} & \text{Department}(CS1) = CS \wedge \text{Department}(Math1) = Math \wedge \dots \\ & \text{Units}(CS1) = 3 \wedge \text{Units}(CS2) = 4 \wedge \dots \\ & \text{RequiredUnitsIn}(18, CS, CS) \\ & \forall u, d \text{ RequiredUnitsIn}(u, d) \Leftrightarrow \\ & \quad (\forall p \exists s, others \text{ Major}(p, d) \Rightarrow \text{Take}(p, \text{Union}(s, others))) \\ & \quad \wedge \text{AllInDepartment}(s, d) \wedge \text{TotalUnits}(s) \geq u \\ & \forall s, d \text{ AllInDepartment}(s, d) \Leftrightarrow (\forall c \ c \in s \Rightarrow \text{Department}(c) = d) \\ & \forall c \text{ TotalUnits}(\{\}) = 0 \\ & \forall c, s \text{ TotalUnits}(\{c|s\}) = \text{Units}(c) + \text{TotalUnits}(s) \end{aligned}$$

One can easily imagine other kinds of requirements; these just give you a flavor.

In this solution we took “over an extended period” to mean that we should recommend a set of courses to take, without scheduling them on a semester-by-semester basis. If you wanted to do that, you would need additional information such as when courses are taught, what is a reasonable course load in a semester, and what courses are prerequisites for what others. For example:

```
Taught(CS1, Fall)
Prerequisites({CS1, CS2}, CS3)
TakeInSemester(Jones, Fall95, {Math1, CS1, English1, History1})
MaxCoursesPerSemester(5)
```

The problem with finding the *best* program of study is in defining what *best* means to the student. It is easy enough to say that all other things being equal, one prefers a good teacher to a bad one, or an interesting course to a boring one. But how do you decide which is best when one course has a better teacher and is expected to be easier, while an alternative is more interesting and provides one more credit? Chapter 16 uses utility theory to address this. If you can provide a way of weighing these elements against each other, then you can choose a best program of study; otherwise you can only eliminate some programs as being worse than others, but can’t pick an absolute best one. Complexity is a further problem: with a general-purpose theorem-prover it’s hard to do much more than enumerate legal programs and pick the best.

Exercise 10.2.#EVER

Figure 10.1 shows the top levels of a hierarchy for everything. Extend it to include as many real categories as possible. A good way to do this is to cover all the things in your everyday life. This includes objects and events. Start with waking up, and proceed in an orderly fashion noting everything that you see, touch, do, and think about. For example, a random sampling produces music, news, milk, walking, driving, gas, Soda Hall, carpet, talking, Professor Fateman, chicken curry, tongue, \$7, sun, the daily newspaper, and so on.

You should produce both a single hierarchy chart (on a large sheet of paper) and a listing of objects and categories with the relations satisfied by members of each category. Every object should be in a category, and every category should be in the hierarchy.

This exercise might be suitable for a term project. At this point, we want to strongly urge that you do assign some of these exercises (or ones like them) to give your students a feeling of what it is really like to do knowledge representation. In general, students find classification hierarchies easier than other representation tasks. A recent twist is to compare one’s hierarchy with online ones such as [yahoo.com](http://www.yahoo.com).

Exercise 10.2.#WATR

Represent the following seven sentences using and extending the representations developed in the chapter:

- a. Water is a liquid between 0 and 100 degrees.

- b. Water boils at 100 degrees.
- c. The water in John's water bottle is frozen.
- d. Perrier is a kind of water.
- e. John has Perrier in his water bottle.
- f. All liquids have a freezing point.
- g. A liter of water weighs more than a liter of alcohol.

Remember that we defined substances so that *Water* is a category whose elements are all those things of which one might say “it’s water.” One tricky part is that the English language is ambiguous. One sense of the word “water” includes ice (“that’s frozen water”), while another sense excludes it: (“that’s not water—it’s ice”). The sentences here seem to use the first sense, so we will stick with that. It is the sense that is roughly synonymous with H_2O .

The other tricky part is that we are dealing with objects that change (freeze and melt) over time. Thus, it won’t do to say $w \in \text{Liquid}$, because w (a mass of water) might be a liquid at one time and a solid at another. For simplicity, we will use a situation calculus representation, with sentences such as $T(w \in \text{Liquid}, s)$. There are many possible correct answers to each of these. The key thing is to be *consistent* in the way that information is represented. For example, do not use *Liquid* as a predicate on objects if *Water* is used as a substance category.

- a. “Water is a liquid between 0 and 100 degrees.” We will translate this as “For any water and any situation, the water is liquid iff and only if the water’s temperature in the situation is between 0 and 100 centigrade.”

$$\begin{aligned} \forall w, s \ w \in \text{Water} \Rightarrow \\ (\text{Centigrade}(0) < \text{Temperature}(w, s) < \text{Centigrade}(100)) \Leftrightarrow \\ T(w \in \text{Liquid}, s) \end{aligned}$$

- b. “Water boils at 100 degrees.” It is a good idea here to do some tool-building. On page 243 we used *MeltingPoint* as a predicate applying to individual instances of a substance. Here, we will define *SBoilingPoint* to denote the boiling point of all instances of a substance. The basic meaning of boiling is that instances of the substance becomes gaseous above the boiling point:

$$\begin{aligned} S\text{BoilingPoint}(c, bp) \Leftrightarrow \\ \forall x, s \ x \in c \Rightarrow \\ (\forall t \ T(\text{Temperature}(x, t), s) \wedge t > bp \Rightarrow T(x \in \text{Gas}, s)) \end{aligned}$$

Then we need only say $S\text{BoilingPoint}(\text{Water}, \text{Centigrade}(100))$.

- c. “The water in John’s water bottle is frozen.”

We will use the constant *Now* to represent the situation in which this sentence holds. Note that it is easy to make mistakes in which one asserts that only some of the water

in the bottle is frozen.

$$\begin{aligned} \exists b \ \forall w \ w \in Water \wedge b \in WaterBottles \wedge Has(John, b, Now) \\ \wedge Inside(w, b, Now) \Rightarrow (w \in Solid, Now) \end{aligned}$$

- d. “Perrier is a kind of water.”

$$Perrier \subset Water$$

- e. “John has Perrier in his water bottle.”

$$\begin{aligned} \exists b \ \forall w \ w \in Water \wedge b \in WaterBottles \wedge Has(John, b, Now) \\ \wedge Inside(w, b, Now) \Rightarrow w \in Perrier \end{aligned}$$

- f. “All liquids have a freezing point.”

Presumably what this means is that all substances that are liquid at room temperature have a freezing point. If we use *RTLiquidSubstance* to denote this class of substances, then we have

$$\forall c \ RTLiquidSubstance(c) \Rightarrow \exists t \ SFreezingPoint(c, t)$$

where *SFreezingPoint* is defined similarly to *SBoilingPoint*. Note that this statement is false in the real world: we can invent categories such as “blue liquid” which do not have a unique freezing point. An interesting exercise would be to define a “pure” substance as one all of whose instances have the same chemical composition.

- g. “A liter of water weighs more than a liter of alcohol.”

$$\begin{aligned} \forall w, a \ w \in Water \wedge a \in Alcohol \wedge Volume(w) = Liters(1) \\ \wedge Volume(a) = Liters(1) \Rightarrow Mass(w) > Mass(a) \end{aligned}$$

Exercise 10.2.#DECM

Write definitions for the following:

- a. *ExhaustivePartDecomposition*
- b. *PartPartition*
- c. *PartwiseDisjoint*

These should be analogous to the definitions for *ExhaustiveDecomposition*, *Partition*, and *Disjoint*. Is it the case that *PartPartition*(*s*, *BunchOf*(*s*))? If so, prove it; if not, give a counterexample and define sufficient conditions under which it does hold.

This is a fairly straightforward exercise that can be done in direct analogy to the corresponding definitions for sets.

- a. *ExhaustivePartDecomposition* holds between a set of parts and a whole, saying that anything that is a part of the whole must be a part of one of the set of parts.

$$\forall s, w \text{ } \textit{ExhaustivePartDecomposition}(s, w) \Leftrightarrow (\forall p \text{ } \textit{PartOf}(p, w) \Rightarrow \exists p_2 \ p_2 \in s \wedge \textit{PartOf}(p, p_2))$$

- b. *PartPartition* holds between a set of parts and a whole when the set is disjoint and is an exhaustive decomposition.

$$\forall s, w \text{ } \textit{PartPartition}(s, w) \Leftrightarrow \textit{PartwiseDisjoint}(s) \wedge \textit{ExhaustivePartDecomposition}(s, w)$$

- c. A set of parts is *PartwiseDisjoint* if when you take any two parts from the set, there is nothing that is a part of both parts.

$$\forall s \text{ } \textit{PartwiseDisjoint}(s) \Leftrightarrow \forall p_1, p_2 \ p_1 \in s \wedge p_2 \in s \wedge p_1 \neq p_2 \Rightarrow \neg \exists p_3 \text{ } \textit{PartOf}(p_3, p_1) \wedge \textit{PartOf}(p_3, p_2)$$

It is *not* the case that $\textit{PartPartition}(s, \textit{BunchOf}(s))$ for any s . A set s may consist of physically overlapping objects, such as a hand and the fingers of the hand. In that case, $\textit{BunchOf}(s)$ is equal to the hand, but s is not a partition of it. We need to ensure that the elements of s are partwise disjoint:

$$\forall s \text{ } \textit{PartwiseDisjoint}(s) \Rightarrow \textit{PartPartition}(s, \textit{BunchOf}(s)).$$

Exercise 10.2.#ALTM

An alternative scheme for representing measures involves applying the units function to an abstract length object. In such a scheme, one would write $\textit{Inches}(\textit{Length}(L_1)) = 1.5$. How does this scheme compare with the one in the chapter? Issues include conversion axioms, names for abstract quantities (such as “50 dollars”), and comparisons of abstract measures in different units (50 inches is more than 50 centimeters).

In the scheme in the chapter, a conversion axiom looks like this:

$$\forall x \text{ } \textit{Centimeters}(2.54 \times x) = \textit{Inches}(x).$$

“50 dollars” is just $\$(50)$, the name of an abstract monetary quantity. For any measure function such as $\$$, we can extend the use of $>$ as follows:

$$\forall x, y \ x > y \Rightarrow \$(x) > \$(y).$$

Since the conversion axiom for dollars and cents has

$$\forall x \ Cents(100 \times x) = \$\$(x)$$

it follows immediately that $\$(50) > Cents(50)$.

In the new scheme, we must introduce objects whose lengths are converted:

$$\forall x \ Centimeters(Length(x)) = 2.54 \times Inches(Length(x)).$$

There is no obvious way to refer directly to “50 dollars” or its relation to “50 cents”. Again, we must introduce objects whose monetary value is 50 dollars or 50 cents:

$$\forall x, y \ \$Value(x) = 50 \wedge Cents(Value(y)) = 50 \Rightarrow \$Value(x) > \$Value(y)$$

Exercise 10.2.#TOMS

Write a set of sentences that allows one to calculate the price of an individual tomato (or other object), given the price per pound. Extend the theory to allow the price of a bag of tomatoes to be calculated.

For an instance i of a substance s with price per pound c and weight n pounds, the price of i will be $n \times c$, or in other words:

$$\begin{aligned} \forall i, s, n, c \ i \in s \wedge PricePer(s, Pounds(1)) = \$\$(c) \wedge Weight(i) = Pounds(n) \\ \Rightarrow Price(i) = \$\$(n \times c) \end{aligned}$$

If b is the set of tomatoes in a bag, then $BunchOf(b)$ is the composite object consisting of all the tomatoes in the bag. Then we have

$$\begin{aligned} \forall i, s, n, c \ b \subset s \wedge PricePer(s, Pounds(1)) = \$\$(c) \\ \wedge Weight(BunchOf(b)) = Pounds(n) \\ \Rightarrow Price(BunchOf(b)) = \$\$(n \times c) \end{aligned}$$

Exercise 10.2.#NAME

Add sentences to extend the definition of the predicate $Name(s, c)$ so that a string such as “laptop computer” matches the appropriate category names from a variety of stores. Try to make your definition general. Test it by looking at ten online stores, and at the category names they give for three different categories. For example, for the category of laptops, we found the names “Notebooks,” “Laptops,” “Notebook Computers,” “Notebook,” “Laptops and Notebooks,” and “Notebook PCs.” Some of these can be covered by explicit $Name$ facts, while others could be covered by sentences for handling plurals, conjunctions, etc.

Plurals can be handled by a *Plural* relation between strings, e.g.,

$$\text{Plural}(\text{"computer"}, \text{"computers"})$$

plus an assertion that the plural (or singular) of a name is also a name for the same category:

$$\forall c, s_1, s_2 \ Name(s_1, c) \wedge (\text{Plural}(s_1, s_2) \vee \text{Plural}(s_2, s_1)) \Rightarrow Name(s_2, c)$$

Conjunctions can be handled by saying that any conjunction string is a name for a category if one of the conjuncts is a name for the category:

$$\forall c, s, s_2 \ \text{Conjunct}(s_2, s) \wedge Name(s_2, c) \Rightarrow Name(s, c)$$

where *Conjunct* is defined appropriately in terms of concatenation. Probably it would be better to redefine *RelevantCategoryName* instead.

Exercise 10.2.#SMKC

One part of the shopping process that was not covered in this chapter is checking for compatibility between items. For example, if a digital camera is ordered, what accessory batteries, memory cards, and cases are compatible with the camera? Write a knowledge base that can determine the compatibility of a set of items and suggest replacements or additional items if the shopper makes a choice that is not compatible. The knowledge base should work with at least one line of products and extend easily to other lines.

Here is an initial sketch of one approach. (Others are possible.) A given object to be purchased may *require* some additional parts (e.g., batteries) to be functional, and there may also be *optional* extras. We can represent requirements as a relation between an individual object and a class of objects, qualified by the number of objects required:

$$\forall x \ x \in \text{Coolpix995DigitalCamera} \Rightarrow \text{Requires}(x, \text{AABattery}, 4).$$

We also need to know that a particular object is compatible, i.e., fills a given role appropriately. For example,

$$\begin{aligned} \forall x, y \ x \in \text{Coolpix995DigitalCamera} \wedge y \in \text{DuracellAABattery} \\ \Rightarrow \text{Compatible}(y, x, \text{AABattery}) \end{aligned}$$

Then it is relatively easy to test whether the set of ordered objects contains compatible required objects for each object.

Exercise 10.2.#SMKG

A complete solution to the problem of inexact matches to the buyer's description in shopping is very difficult and requires a full array of natural language processing and information retrieval techniques. (See Chapters 23 and ??.) One small step is to allow the user to spec-

ify minimum and maximum values for various attributes. The buyer must use the following grammar for product descriptions:

$$\begin{array}{lcl} \text{Description} & \rightarrow & \text{Category} [\text{Connector Modifier}]^* \\ \text{Connector} & \rightarrow & \text{"with"} \mid \text{"and"} \mid \text{","} \\ \text{Modifier} & \rightarrow & \text{Attribute} \mid \text{Attribute Op Value} \\ \text{Op} & \rightarrow & \text{"}=\\>\text{"} \mid \text{"}<\text{"} \end{array}$$

Here, *Category* names a product category, *Attribute* is some feature such as “CPU” or “price,” and *Value* is the target value for the attribute. So the query “computer with at least a 2.5 GHz CPU for under \$500” must be re-expressed as “computer with CPU $>$ 2.5 GHz and price $<$ \$500.” Implement a shopping agent that accepts descriptions in this language.

Chapter 23 explains how to use logic to parse text strings and extract semantic information. The outcome of this process is a definition of what objects are acceptable to the user for a specific shopping request; this allows the agent to go out and find offers matching the user’s requirements. We omit the full definition of the agent, although a skeleton may appear on the AIMA project web pages.

10.3 Events

Exercise 10.3.#WNDO

Develop a representational system for reasoning about windows in a window-based computer interface. In particular, your representation should be able to describe:

- The state of a window: minimized, displayed, or nonexistent.
- Which window (if any) is the active window.
- The position of every window at a given time.
- The order (front to back) of overlapping windows.
- The actions of creating, destroying, resizing, and moving windows; changing the state of a window; and bringing a window to the front. Treat these actions as atomic; that is, do not deal with the issue of relating them to mouse actions. Give axioms describing the effects of actions on fluents. You may use either event or situation calculus.

Assume an ontology containing *situations*, *actions*, *integers* (for x and y coordinates) and *windows*. Define a language over this ontology; that is, a list of constants, function symbols, and predicates with an English description of each. If you need to add more categories to the ontology (e.g., pixels), you may do so, but be sure to specify these in your write-up. You may (and should) use symbols defined in the text, but be sure to list these explicitly.

A plausible language might contain the following primitives:

Temporal Predicates:

Poss(a, s) – Predicate: Action a is possible in situation s . As in section 10.3

Result(a, s) – Function from action a and situation s to situation. As in section 10.3.

Arithmetic: $x < y, x \leq y, x + y, 0$.

Window State:

Minimized(w, s), Displayed(w, s), Nonexistent(w, s), Active(w, s) – Predicates. In all these w is a window and s is a situation. (“*Displayed(w, s)*” means existent and non-minimized; it includes the case where all of w is actually occluded by other windows.)

Window Position:

RightEdge(w, s), LeftEdge(w, s), TopEdge(w, s), BottomEdge(w, s): Functions from a window w and situation s to a coordinate.

ScreenWidth, ScreenHeight: Constants.

Window Order:

InFront(w1, w2, s): Predicate. Window $w1$ is in front of window $w2$ in situation s .

Actions:

Minimize(w), MakeVisible(w), Destroy(w), BringToFront(w) – Functions from a window w to an action.

Move(w, dx, dy) – Move window w by dx to the left and dy upward. (Quantities dx and dy may be negative.)

Resize(w, dxl, dxr, dyb, dyt) – Resize window w by dxl on the left, dxr on the right, dyb on bottom, and dyt on top.

Exercise 10.3.#WNDP

State the following in the language you developed for the previous exercise:

- a. In situation S_0 , window W_1 is behind W_2 but sticks out on the left and right. Do *not* state exact coordinates for these; describe the *general* situation.
- b. If a window is displayed, then its top edge is higher than its bottom edge.
- c. After you create a window w , it is displayed.
- d. A window can be minimized if it is displayed.

- a $LeftEdge(W_1, S_0) < LeftEdge(W_2, S_0) \wedge RightEdge(W_2, S_0) < RightEdge(W_1, S_0) \wedge TopEdge(W_1, S_0) \leq TopEdge(W_2, S_0) \wedge BottomEdge(W_2, S_0) \leq BottomEdge(W_1, S_0) \wedge InFront(W_2, W_1, S_0)$.
- b $\forall w, s \quad Displayed(w, s) \Rightarrow BottomEdge(w, s) < TopEdge(w, s)$.
- c $\forall w, s \quad Poss(Create(w), s) \Rightarrow Displayed(w, Result(Create(w), s))$.
- d $Displayed(w, s) \Rightarrow Poss(Minimize(w), s)$

Exercise 10.3.#SPMK

(Adapted from an example by Doug Lenat.) Your mission is to capture, in logical form, enough knowledge to answer a series of questions about the following simple scenario:

Yesterday John went to the North Berkeley Safeway supermarket and bought two pounds of tomatoes and a pound of ground beef.

Start by trying to represent the content of the sentence as a series of assertions. You should write sentences that have straightforward logical structure (e.g., statements that objects have certain properties, that objects are related in certain ways, that all objects satisfying one property satisfy another). The following might help you get started:

- Which classes, objects, and relations would you need? What are their parents, siblings and so on? (You will need events and temporal ordering, among other things.)
- Where would they fit in a more general hierarchy?
- What are the constraints and interrelationships among them?
- How detailed must you be about each of the various concepts?

To answer the questions below, your knowledge base must include background knowledge. You'll have to deal with what kind of things are at a supermarket, what is involved with purchasing the things one selects, what the purchases will be used for, and so on. Try to make your representation as general as possible. To give a trivial example: don't say "People buy food from Safeway," because that won't help you with those who shop at another supermarket. Also, don't turn the questions into answers; for example, question (c) asks "Did John buy any meat?"—not "Did John buy a pound of ground beef?"

Sketch the chains of reasoning that would answer the questions. If possible, use a logical reasoning system to demonstrate the sufficiency of your knowledge base. Many of the things you write might be only approximately correct in reality, but don't worry too much; the idea is to extract the common sense that lets you answer these questions at all. A truly complete answer to this question is *extremely* difficult, probably beyond the state of the art of current knowledge representation. But you should be able to put together a consistent set of axioms for the limited questions posed here.

- a. Is John a child or an adult? [Adult]
- b. Does John now have at least two tomatoes? [Yes]
- c. Did John buy any meat? [Yes]
- d. If Mary was buying tomatoes at the same time as John, did he see her? [Yes]
- e. Are the tomatoes made in the supermarket? [No]
- f. What is John going to do with the tomatoes? [Eat them]
- g. Does Safeway sell deodorant? [Yes]
- h. Did John bring some money or a credit card to the supermarket? [Yes]
- i. Does John have less money after going to the supermarket? [Yes]

This is the most involved representation problem. It is suitable for a group project of 2

or 3 students over the course of at least 2 weeks. Solutions should include a taxonomy, a choice of situation calculus, fluent calculus, or event calculus for handling time and change, and enough background knowledge. If a logic programming system or theorem prover is not used, students might want to write out the proofs for at least some of the answers.

Exercise 10.3.#SPML

Make the necessary additions or changes to your knowledge base from the previous exercise so that the questions that follow can be answered. Include in your report a discussion of your changes, explaining why they were needed, whether they were minor or major, and what kinds of questions would necessitate further changes.

- a. Are there other people in Safeway while John is there? [Yes—staff!]
- b. Is John a vegetarian? [No]
- c. Who owns the deodorant in Safeway? [Safeway Corporation]
- d. Did John have an ounce of ground beef? [Yes]
- e. Does the Shell station next door have any gas? [Yes]
- f. Do the tomatoes fit in John's car trunk? [Yes]

Normally one would assign the preceding exercise in one assignment, and then when it is done, add this exercise (possibly varying the questions). That way, the students see whether they have made sufficient generalizations in their initial answer, and get experience with debugging and modifying a knowledge base.

Exercise 10.3.#EVEW

Write event calculus axioms to describe the actions in the wumpus world.

Section 10.3 includes a couple of axioms for the wumpus world:

$$\begin{aligned} \text{Initiates}(e, \text{HaveArrow}(a), t) &\Leftrightarrow e = \text{Start} \\ \text{Terminates}(e, \text{HaveArrow}(a), t) &\Leftrightarrow e \in \text{Shootings}(a) \end{aligned}$$

Here is an axiom for turning; the others are similar albeit more complex. Let the term $\text{TurnRight}(a)$ denote the event category of the agent turning right. We want to say about it that if (say) the agent is facing south up to the beginning of the action, then it is facing west after the action ends, and so on.

$$\begin{aligned} T(\text{TurnRight}(a), i) &\Leftrightarrow \\ &[\exists h \ Meets(h, i) \wedge T(\text{FacingSouth}(a), h) \Rightarrow \\ &\quad \text{Clipped}(\text{FacingSouth}(a), i) \wedge \text{Restored}(\text{FacingWest}(a), i)] \\ &\vee \quad \dots \end{aligned}$$

Exercise 10.3.#INAL

State the interval-algebra relation that holds between every pair of the following real-world events:

- LK*: The life of President Kennedy.
- IK*: The infancy of President Kennedy.
- PK*: The presidency of President Kennedy.
- LJ*: The life of President Johnson.
- PJ*: The presidency of President Johnson.
- LO*: The life of President Obama.

Starts(*IK*, *LK*).

Finishes(*PK*, *LK*).

During(*LK*, *LJ*).

Meets(*LK*, *PJ*).

Overlap(*LK*, *LC*).

Before(*IK*, *PK*).

During(*IK*, *LJ*).

Before(*IK*, *PJ*).

Before(*IK*, *LC*).

During(*PK*, *LJ*).

Meets(*PK*, *PJ*).

During(*PK*, *LC*).

During(*PJ*, *LJ*).

Overlap(*LJ*, *LC*).

During(*PJ*, *LC*).

Exercise 10.3.#RBGO

This exercise concerns the problem of planning a route for a robot to take from one city to another. The basic action taken by the robot is *Go*(*x*, *y*), which takes it from city *x* to city *y* if there is a route between those cities. *Road*(*x*, *y*) is true if and only if there is a road connecting cities *x* and *y*; if there is, then *Distance*(*x*, *y*) gives the length of the road. See the map on page 64 for an example. The robot begins in Arad and must reach Bucharest.

- a. Write a suitable logical description of the initial situation of the robot.
- b. Write a suitable logical query whose solutions provide possible paths to the goal.
- c. Write a sentence describing the *Go* action.
- d. Now suppose that the robot consumes fuel at the rate of .02 gallons per mile. The robot starts with 20 gallons of fuel. Augment your representation to include these considerations.
- e. Now suppose some of the cities have gas stations at which the robot can fill its tank. Extend your representation and write all the rules needed to describe gas stations, in-

cluding the *Fillup* action.

This question takes the student through the initial stages of developing a logical representation for actions that incorporates more and more realism. Implementing the reasoning tasks in a theorem-prover is also a good idea. Although the use of logical reasoning for the initial task—finding a route on a graph—may seem like overkill, the student should be impressed that we can keep making the situation more complicated simply by describing those added complications, with no additions to the reasoning system.

a. $At(Robot, Arad, S_0)$.

b. $\exists s \ At(Robot, Bucharest, s)$.

c. The successor-state axiom should be mechanical by now. $\forall a, x, y, s :$

$$\begin{aligned} At(Robot, y, Result(a, s)) &\Leftrightarrow [(a = Go(x, y) \\ &\quad \wedge Road(x, y) \wedge At(Robot, x, s)) \\ &\quad \vee (At(Robot, y, s) \\ &\quad \wedge \neg(\exists z \ a = Go(y, z) \wedge z \neq y))] \end{aligned}$$

d. To represent the fuel the robot has in a given situation, use the function $FuelLevel(Robot, s)$; this refers not to actual fuel or to a number, but to an abstract amount. Let $Full = Gallons(20)$ be a constant denoting the fuel capacity of the tank. The initial situation is then described by $At(Robot, Arad, S_0) \wedge FuelLevel(Robot, S_0) = Full$. Also, $Distance(x, y)$ will return an abstract length object.

Perhaps the trickiest part is handling the measures properly. To simplify things, we have used a little trick, defining “inverse” unit functions such as $Miles^{-1}$, which returns the number of miles in a given length object:

$$\forall x \ Miles^{-1}(Miles(x)) = x .$$

Now, the successor-state axiom for location is extended as follows (note that we do not

say what happens if the robot runs out of gas):

$$\begin{aligned}
 & At(Robot, y, Result(a, s)) \\
 \Leftrightarrow & [(a = Go(x, y) \\
 & \wedge Road(x, y) \wedge At(Robot, x, s) \\
 & \wedge 0.02 \times Miles^{-1}(Distance(x, y)) \leq Gallons^{-1}(FuelLevel(Robot, s))) \\
 \vee & (At(Robot, y, s) \\
 & \wedge \neg(\exists z \ a = Go(y, z) \wedge z \neq y))] \\
 FuelLevel(Robot, Result(a, s)) = f \\
 \Leftrightarrow & [(a = Go(x, y) \\
 & \wedge Road(x, y) \wedge At(Robot, x, s) \\
 & \wedge 0.02 \times Miles^{-1}(Distance(x, y)) \leq Gallons^{-1}(FuelLevel(Robot, s))) \\
 & \wedge f = FuelLevel(Robot, s) - Gallons(0.02 \times Miles^{-1}(Distance(x, y))) \\
 \vee & (f = Fuel(Robot, s) \\
 & \wedge \neg(\exists v, w \ a = Go(v, w) \wedge v \neq w))]
 \end{aligned}$$

- e. The simplest way to extend the representation is to add the predicate $GasStation(x)$, which is true of cities with gas stations. The *Fillup* action is described by adding another clause to the above axiom for *Fuel*, saying that $f = Full$ when $a = FillUp$.

Exercise 10.3.#EVES

Investigate ways to extend the event calculus to handle *simultaneous* events. Is it possible to avoid a combinatorial explosion of axioms?

The main difficulty with simultaneous (also called concurrent) events and actions is how to account correctly for possible interference. A good starting point is the expository paper by Shanahan (1999). Section 5 of that paper shows how to manage concurrent actions by the introduction of additional generic predicates *Cancels* and *Canceled*, describing circumstances in which actions may interfere with each other. We avoid lots of “non-cancellation” assertions using the same predicate-completion trick as in successor-state axioms, and the meaning of cancellation is defined once and for all through its connection to clipping, restoring, etc.

Exercise 10.3.#EXRT

Construct a representation for exchange rates between currencies that allows for daily fluctuations.

For quantities such as length and time, the conversion axioms such as

$$Centimeters(2.54 \times d) = Inches(d)$$

are absolutes that hold (with a few exceptions) for all time. The same is true for conversion axioms within a given currency; for example, $US\$(1) = US\(100) . When it comes to

conversion *between* currencies, we make the simplifying assumption that at any given time t there is a prevailing exchange rate:

$$T(\text{ExchangeRate}(\text{UK}\mathcal{L}(1), \text{US\$}(1)) = 1.55, t)$$

and the rate is reciprocal:

$$\text{ExchangeRate}(\text{UK}\mathcal{L}(1), \text{US\$}(1)) = 1 / \text{ExchangeRate}(\text{US\$}(1), \text{UK}\mathcal{L}(1)).$$

What we cannot do, however, is write

$$T(\text{UK}\mathcal{L}(1) = \text{US\$}(1.55), t)$$

thereby *equating* abstract amounts of money in different currencies. At any given moment, prevailing exchange rates across the world's currencies need not be consistent, and using equality across currencies would therefore introduce a *logical* inconsistency. Instead, exchange rates should be interpreted as indicating a willingness to exchange, perhaps with some commission; and exchange rate inconsistency is an opportunity for arbitrage. A more sophisticated model would include the entity offering the rate, limits on amounts and forms of payment, etc.

Exercise 10.3.#FIXD

Define the predicate *Fixed*, where $\text{Fixed}(\text{Location}(x))$ means that the location of object x is fixed over time.

Any object x is an event, and $\text{Location}(x)$ is the event that for every subinterval of time, refers to the place where x is. For example, $\text{Location}(\text{Peter})$ is the complex event consisting of his home from midnight to about 9:00 today, then various parts of the road, then his office from 10:00 to 1:30, and so on. To say that an event is fixed is to say that any two moments of the event have the same spatial extent:

$$\begin{aligned} \forall e \text{ } \text{Fixed}(e) &\Leftrightarrow \\ (\forall a, b \text{ } a \in \text{Moments} \wedge b \in \text{Moments} \wedge \text{Subevent}(a, e) \wedge \text{Subevent}(b, e)) \\ &\Rightarrow \text{SpatialExtent}(a) = \text{SpatialExtent}(b) \end{aligned}$$

Exercise 10.3.#TRAD

Describe the event of trading something for something else. Describe buying as a kind of trading in which one of the objects traded is a sum of money.

Let $\text{Trade}(b, x, a, y)$ denote the class of events where person b trades object y to person

a for object *x*:

$$\begin{aligned} T(\text{Trade}(b, x, a, y), i) \Leftrightarrow \\ T(\text{Owns}(b, y), \text{Start}(i)) \wedge T(\text{Owns}(a, x), \text{Start}(i)) \wedge \\ T(\text{Owns}(b, x), \text{End}(i)) \wedge T(\text{Owns}(a, y), \text{End}(i)) \end{aligned}$$

Now the only tricky part about defining buying in terms of trading is in distinguishing a price (a measurement) from an actual collection of money.

$$T(\text{Buy}(b, x, a, p), i) \Leftrightarrow \exists m \text{ Money}(m) \wedge \text{Trade}(b, x, a, m) \wedge \text{Value}(m) = p$$

Exercise 10.3.#OWNR

The two preceding exercises assume a fairly primitive notion of ownership. For example, the buyer starts by *owning* the dollar bills. This picture begins to break down when, for example, one's money is in the bank, because there is no longer any specific collection of dollar bills that one owns. The picture is complicated still further by borrowing, leasing, renting, and bailment. Investigate the various commonsense and legal concepts of ownership, and propose a scheme by which they can be represented formally.

There are many possible approaches to this exercise. The idea is for the students to think about doing knowledge representation for real; to consider a host of complications and find some way to represent the facts about them. Some of the key points are:

- Ownership occurs over time, so we need either a situation-calculus or interval-calculus approach.
- There can be joint ownership and corporate ownership. This suggests the owner is a group of some kind, which in the simple case is a group of one person.
- Ownership provides certain rights: to use, to resell, to give away, etc. Much of this is outside the definition of ownership *per se*, but a good answer would at least consider how much of this to represent.
- Own can own abstract obligations as well as concrete objects. This is the idea behind the futures market, and also behind banks: when you deposit a dollar in a bank, you are giving up ownership of that particular dollar in exchange for ownership of the right to withdraw another dollar later. (Or it could coincidentally turn out to be the exact same dollar.) Leases and the like work this way as well. This is tricky in terms of representation, because it means we have to reify transactions of this kind. That is, *Withdraw(person, money, bank, time)* must be an object, not a predicate.

Exercise 10.3.#SMKI

Our description of Internet shopping omitted the all-important step of actually *buying* the product. Provide a formal logical description of buying, using event calculus. That is,

define the sequence of events that occurs when a buyer submits a credit-card purchase and then eventually gets billed and receives the product.

Here is a simple version of the answer; it can be elaborated *ad infinitum*. Let the term $Buy(b, x, s, p)$ denote the event category of buyer b buying object x from seller s for price p . We want to say about it that b transfers the money to s , and s transfers ownership of x to b .

$$\begin{aligned} T(Buy(b, x, s, p), i) \Leftrightarrow \\ T(Owes(s, x), Start(i)) \wedge \\ \exists m \ Money(m) \wedge p = Value(m) \wedge T(Owes(b, m), Start(i)) \wedge \\ T(Owes(b, x), End(i)) \wedge T(Owes(s, m), End(i)) \end{aligned}$$

10.4 Mental Objects and Modal Logic

Exercise 10.4.#CDFH

Consider a game played with a deck of just 8 cards, 4 aces and 4 kings. The three players, Alice, Bob, and Carlos, are dealt two cards each. Without looking at them, they place the cards on their foreheads so that the other players can see them. Then the players take turns either announcing that they know what cards are on their own forehead, thereby winning the game, or saying “I don’t know.” Everyone knows the players are truthful and are perfect at reasoning about beliefs.

- Game 1. Alice and Bob have both said “I don’t know.” Carlos sees that Alice has two aces (A-A) and Bob has two kings (K-K). What should Carlos say? (*Hint:* consider all three possible cases for Carlos: A-A, K-K, A-K.)
- Describe each step of Game 1 using the notation of modal logic.
- Game 2. Carlos, Alice, and Bob all said “I don’t know” on their first turn. Alice holds K-K and Bob holds A-K. What should Carlos say on his second turn?
- Game 3. Alice, Carlos, and Bob all say “I don’t know” on their first turn, as does Alice on her second turn. Alice and Bob both hold A-K. What should Carlos say?
- Prove that there will always be a winner to this game.

(Adapted from Fagin *et al.* (1995).)

Just to get you started: In Game 1, Alice says “I don’t know.” If Carlos had K-K, and given that Alice can see Bob’s K-K, then she would know that Bob and Carlos had all four kings between them and she would announce A-A. Therefore, Carlos does not have K-K. Then Bob says “I don’t know.” If Carlos had A-A, and given that Bob can see Alice’s A-A, then he would know that Alice and Carlos had all four aces between them and he would announce A-A. Therefore, Carlos does not have A-A. Therefore Carlos should announce A-K.

Exercise 10.4.#LGOM

The assumption of *logical omniscience*, discussed on page 328, is of course not true of any actual reasoners. Rather, it is an *idealization* of the reasoning process that may be more or less acceptable depending on the applications. Discuss the reasonableness of the assumption for each of the following applications of reasoning about knowledge:

- a. Partial knowledge adversary games, such as card games. Here one player wants to reason about what his opponent knows about the state of the game.
 - b. Chess with a clock. Here the player may wish to reason about the limits of his opponent's or his own ability to find the best move in the time available. For instance, if player A has much more time left than player B, then A will sometimes make a move that greatly complicates the situation, in the hopes of gaining an advantage because he has more time to work out the proper strategy.
 - c. A shopping agent in an environment in which there are costs of gathering information.
 - d. Reasoning about public key cryptography, which rests on the intractability of certain computational problems.
-
- A. The logical omniscience assumption is a reasonable idealization. The limiting factor here is generally the information available to the players, not the difficulty of making inferences.
 - B. This kind of reasoning cannot be accommodated in a theory with logical omniscience. If logical omniscience were true, then every player could always figure out the optimal move instantaneously.
 - C. Logical omniscience is a reasonable idealization. The costs of getting the information are almost always much greater than the costs of reasoning with it.
 - D. It depends on the kind of reasoning you want to do. If you want to reason about the relation of cryptography to particular computational problems, then logical omniscience cannot be assumed, because the assumption entails that any computational problem can be solved instantly. On the other hand, if you are willing to idealize the encryption/decryption as a magical process with no computational basis, then it may be reasonable to apply a theory with logical omniscience to other aspects of the theory.

10.5 Reasoning Systems for Categories

Exercise 10.5.#DSCL

Translate the following description logic expression (from page 332) into first-order logic, and comment on the result:

$$\begin{aligned} & \text{And}(\text{Man}, \text{AtLeast}(3, \text{Son}), \text{AtMost}(2, \text{Daughter}), \\ & \quad \text{All}(\text{Son}, \text{And}(\text{Unemployed}, \text{Married}, \text{All}(\text{Spouse}, \text{Doctor}))), \\ & \quad \text{All}(\text{Daughter}, \text{And}(\text{Professor}, \text{Fills}(\text{Department}, \text{Physics}, \text{Math}))) . \end{aligned}$$

This corresponds to the following open formula:

$$\begin{aligned}
 & \text{Man}(x) \wedge \exists s_1, s_2, s_3 \ Son(s_1, x) \wedge Son(s_2, x) \wedge Son(s_3, x) \\
 & \quad \wedge s_1 \neq s_2 \wedge s_1 \neq s_3 \wedge s_2 \neq s_3 \\
 & \quad \wedge \neg \exists d_1, d_2, d_3 \ Daughter(d_1, x) \wedge Daughter(d_2, x) \wedge Daughter(d_3, x) \\
 & \quad \quad \wedge d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \\
 & \quad \wedge \forall s \ Son(s, x) \Rightarrow Unemployed(s) \wedge Married(s) \wedge Doctor(Spouse(s)) \\
 & \quad \wedge \forall d \ Daughter(d, x) \Rightarrow Professor(d) \wedge \\
 & \quad \quad (Department(d) = Physics \vee Department(d) = Math) .
 \end{aligned}$$

Exercise 10.5.#INHX

Recall that inheritance information in semantic networks can be captured logically by suitable implication sentences. This exercise investigates the efficiency of using such sentences for inheritance.

- Consider the information in a used-car catalog such as Kelly's Blue Book—for example, that 1973 Dodge vans are (or perhaps were once) worth \$575. Suppose all this information (for 11,000 models) is encoded as logical sentences, as suggested in the chapter. Write down three such sentences, including that for 1973 Dodge vans. How would you use the sentences to find the value of a *particular* car, given a backward-chaining theorem prover such as Prolog?
- Compare the time efficiency of the backward-chaining method for solving this problem with the inheritance method used in semantic nets.
- Explain how forward chaining allows a logic-based system to solve the same problem efficiently, assuming that the KB contains only the 11,000 sentences about prices.
- Describe a situation in which neither forward nor backward chaining on the sentences will allow the price query for an individual car to be handled efficiently.
- Can you suggest a solution enabling this type of query to be solved efficiently in all cases in logic systems? (*Hint:* Remember that two cars of the same year and model have the same price.)

In many AI and Prolog textbooks, you will find it stated plainly that implications suffice for the implementation of inheritance. This is true in the logical but not the practical sense.

- Here are three rules, written in Prolog. We actually would need many more clauses on the right hand side to distinguish between different models, different options, etc.

```

worth(X, 575) :- year(X, 1973), make(X, dodge), style(X, van).
worth(X, 27000) :- year(X, 1994), make(X, lexus), style(X, sedan).
worth(X, 5000) :- year(X, 1987), make(X, toyota), style(X, sedan).

```

To find the value of JB, given a data base with `year(jb, 1973), make(jb, dodge)`

and `style(jb, van)` we would call the backward chainer with the goal `worth(jb, D)`, and read the value for `D`.

- b. The time efficiency of this query is $O(n)$, where n in this case is the 11,000 entries in the Blue Book. A semantic network with inheritance would allow us to follow a link from `JB` to `1973-dodge-van`, and from there to follow the `worth` slot to find the dollar value in $O(1)$ time.
- c. With forward chaining, as soon as we are told the three facts about `JB`, we add the new fact `worth(jb, 575)`. Then when we get the query `worth(jb, D)`, it is $O(1)$ to find the answer, assuming indexing on the predicate and first argument. This makes logical inference seem just like semantic networks except for two things: the logical inference does a hash table lookup instead of pointer following, and logical inference explicitly stores `worth` statements for each individual car, thus wasting space if there are a lot of individual cars. (For this kind of application, however, we will probably want to consider only a few individual cars, as opposed to the 11,000 different models.)
- d. If each category has many properties—for example, the specifications of all the replacement parts for the vehicle—then forward-chaining on the implications will also be an impractical way to figure out the price of a vehicle.
- e. If we have a rule of the following kind:

```
worth(X, D) :- year-make-style(X, Yr, Mk, St),  
                 year-make-style(Y, Yr, Mk, St), worth(Y, D).
```

together with facts in the database about some other specific vehicle of the same type as `JB`, then the query `worth(jb, D)` will be solved in $O(1)$ time with appropriate indexing, regardless of how many other facts are known about that type of vehicle and regardless of the number of types of vehicle.

Exercise 10.5.#NATS

One might suppose that the syntactic distinction between unboxed links and singly boxed links in semantic networks is unnecessary, because singly boxed links are always attached to categories; an inheritance algorithm could simply assume that an unboxed link attached to a category is intended to apply to all members of that category. Show that this argument is fallacious, giving examples of errors that would arise.

When categories are reified, they can have properties as individual objects (such as *Cardinality* and *Supersets*) that do not apply to their elements. Without the distinction between boxed and unboxed links, the sentence *Cardinality(SingletonSets, 1)* might mean that every singleton set has one element, or that there is only one singleton set.

10.6 Reasoning with Default Information

[[need exercises]]

EXERCISES 11

AUTOMATED PLANNING

11.1 Definition of Classical Planning

Exercise 11.1.#PSPL

Describe the differences and similarities between problem solving and planning.

Both problem solver and planner are concerned with getting from a start state to a goal using a set of defined operations or actions, most commonly in a deterministic, discrete, observable environment (although those constraints can be relaxed). In problem solving we use atomic representations. In planning, however, we open up the representation of states, goals, and plans, using factored or relational representations that allow for a wider variety of algorithms that decompose the search space, search forwards or backwards, and use automated generation of heuristic functions.

Exercise 11.1.#PDDR

Consider a robot whose operation is described by the following PDDL operators:

$Op(\text{ACTION}:Go(x, y), \text{PRECOND}:At(Robot, x), \text{EFFECT}:\neg At(Robot, x) \wedge At(Robot, y))$
 $Op(\text{ACTION}:Pick(o), \text{PRECOND}:At(Robot, x) \wedge At(o, x), \text{EFFECT}:\neg At(o, x) \wedge Holding(o))$
 $Op(\text{ACTION}:Drop(o), \text{PRECOND}:At(Robot, x) \wedge Holding(o), \text{EFFECT}:At(o, x) \wedge \neg Holding(o))$

- The operators allow the robot to hold more than one object. Show how to modify them with an *EmptyHand* predicate for a robot that can hold only one object.
- Assuming that these are the only actions in the world, write a successor-state axiom for *EmptyHand*.

- EmptyHand* is not affected by *Go*, so we modify just the *Pick* and *Drop* operators:

$Op(\text{ACTION}:Pick(o), \text{PRECOND}:EmptyHand() \wedge At(Robot, x) \wedge At(o, x),$
 EFFECT: $\neg EmptyHand() \wedge \neg At(o, x) \wedge Holding(o))$
 $Op(\text{ACTION}:Drop(o), \text{PRECOND}:At(Robot, x) \wedge Holding(o),$
 EFFECT: $EmptyHand() \wedge At(o, x) \wedge \neg Holding(o))$

Notice that STRIPS does not allow negated preconditions, so we could not use $\neg Holding(p)$ as a precondition for *Pick(o)*; this is why we need *EmptyHand*. Also, we cannot use

$\neg \text{EmptyHand}$ as a precondition of $\text{Drop}(o)$; but this is no problem because we already have $\text{Holding}(o)$ as a precondition.

- b. In English, the hand is empty after doing an action if it was empty before and the action was not a successful *Pick*; or if an object was dropped.

$$\begin{aligned} \text{EmptyHand}(\text{Result}(a, s)) \Leftrightarrow \\ [(\text{EmptyHand}(s) \wedge \neg \exists o, x \ (\text{At}(\text{Robot}, x) \wedge \text{At}(o, x) \wedge a = \text{Pick}(o))) \\ \vee (\text{Holding}(o, s) \wedge a = \text{Drop}(o))] \end{aligned}$$

Exercise 11.1.#ARST

Given the action schemas and initial state from Figure 11.1, what are all the applicable concrete instances of $\text{Fly}(p, \text{from}, \text{to})$ in the state described by

$$\begin{aligned} \text{At}(P_1, \text{JFK}) \wedge \text{At}(P_2, \text{SFO}) \wedge \text{Plane}(P_1) \wedge \text{Plane}(P_2) \\ \wedge \text{Airport}(\text{JFK}) \wedge \text{Airport}(\text{SFO}) ? \end{aligned}$$

This is an easy exercise, the point of which is to understand that “applicable” means satisfying the preconditions, and that a concrete action instance is one with the variables replaced by constants. The applicable actions are:

$$\begin{aligned} \text{Fly}(P_1, \text{JFK}, \text{SFO}) \\ \text{Fly}(P_1, \text{JFK}, \text{JFK}) \\ \text{Fly}(P_2, \text{SFO}, \text{JFK}) \\ \text{Fly}(P_2, \text{SFO}, \text{SFO}) \end{aligned}$$

A minor point of this is that the action of flying nowhere—from one airport to itself—is allowable by the definition of *Fly*, and is applicable (if not useful).

Exercise 11.1.#MOBA

The monkey-and-bananas problem is faced by a monkey in a laboratory with some bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at *A*, the bananas at *B*, and the box at *C*. The monkey and box have height *Low*, but if the monkey climbs onto the box he will have height *High*, the same as the bananas. The actions available to the monkey include *Go* from one place to another, *Push* an object from one place to another, *ClimbUp* onto or *ClimbDown* from an object, and *Grasp* or *Ungrasp* an object. The result of a *Grasp* is that the monkey holds the object if the monkey and object are in the same place at the same height.

- a. Write down the initial state description.
- b. Write the six action schemas.
- c. Suppose the monkey wants to fool the scientists, who are off to tea, by grabbing the bananas, but leaving the box in its original place. Write this as a general goal (i.e., not

assuming that the box is necessarily at C) in the language of situation calculus. Can this goal be solved by a classical planning system?

- d. Your schema for pushing is probably incorrect, because if the object is too heavy, its position will remain the same when the *Push* schema is applied. Fix your action schema to account for heavy objects.

This exercise is intended as a fairly easy exercise in describing a domain.

- a. The initial state is:

$$\begin{aligned} & At(Monkey, A) \wedge At(Bananas, B) \wedge At(Box, C) \wedge \\ & Height(Monkey, Low) \wedge Height(Box, Low) \wedge Height(Bananas, High) \wedge \\ & Pushable(Box) \wedge Climbable(Box) \end{aligned}$$

- b. The actions are:

$$\begin{aligned} & Action(ACTION:Go(x, y), PRECOND:At(Monkey, x), \\ & \quad EFFECT:At(Monkey, y) \wedge \neg(At(Monkey, x))) \\ & Action(ACTION:Push(b, x, y), PRECOND:At(Monkey, x) \wedge Pushable(b), \\ & \quad EFFECT:At(b, y) \wedge At(Monkey, y) \wedge \neg At(b, x) \wedge \neg At(Monkey, x)) \\ & Action(ACTION:ClimbUp(b), \\ & \quad PRECOND:At(Monkey, x) \wedge At(b, x) \wedge Climbable(b), \\ & \quad EFFECT:On(Monkey, b) \wedge \neg Height(Monkey, Low) \\ & \quad \wedge Height(Monkey, High)) \\ & Action(ACTION:Grasp(b), \\ & \quad PRECOND:Height(Monkey, h) \wedge Height(b, h) \\ & \quad \wedge At(Monkey, x) \wedge At(b, x), \\ & \quad EFFECT:Have(Monkey, b)) \\ & Action(ACTION:ClimbDown(b), \\ & \quad PRECOND:On(Monkey, b) \wedge Height(Monkey, High), \\ & \quad EFFECT:\neg On(Monkey, b) \wedge \neg Height(Monkey, High) \\ & \quad \wedge Height(Monkey, Low)) \\ & Action(ACTION:UnGrasp(b), PRECOND:Have(Monkey, b), \\ & \quad EFFECT:\neg Have(Monkey, b)) \end{aligned}$$

- c. In situation calculus, the goal is a state s such that:

$$Have(Monkey, Bananas, s) \wedge (\exists x \ At(Box, x, s_0) \wedge At(Box, x, s))$$

In STRIPS, we can only talk about the goal state; there is no way of representing the fact that there must be some relation (such as equality of location of an object) between two states within the plan. So there is no way to represent this goal.

- d. Actually, we did include the *Pushable* precondition in the solution above.

Exercise 11.1.#SHAK

The original STRIPS planner was designed to control Shakey the robot. Figure 11.1 shows a version of Shakey's world consisting of four rooms lined up along a corridor, where each room has a door and a light switch. The actions in Shakey's world include moving from place to place, pushing movable objects (such as boxes), climbing onto and down from rigid objects (such as boxes), and turning light switches on and off. The robot itself could not climb on a box or toggle a switch, but the planner was capable of finding and printing out plans that were beyond the robot's abilities. Shakey's six actions are the following:

- $Go(x, y, r)$, which requires that Shakey be *At* x and that x and y are locations *In* the same room r . By convention a door between two rooms is in both of them.
- Push a box b from location x to location y within the same room: $Push(b, x, y, r)$. You will need the predicate *Box* and constants for the boxes.
- Climb onto a box from position x : $ClimbUp(x, b)$; climb down from a box to position x : $ClimbDown(b, x)$. We will need the predicate *On* and the constant *Floor*.
- Turn a light switch on or off: $TurnOn(s, b)$; $TurnOff(s, b)$. To turn a light on or off, Shakey must be on top of a box at the light switch's location.

Write PDDL sentences for Shakey's six actions and the initial state from Figure 11.1. Construct a plan for Shakey to get *Box*₂ into *Room*₂.

The actions are quite similar to the monkey and bananas problem—you should probably assign only one of these two problems. The actions are:

```
Action(ACTION:Go(x, y), PRECOND:At(Shakey, x) ∧ In(x, r) ∧ In(y, r),
EFFECT:At(Shakey, y) ∧ ¬(At(Shakey, x)))
Action(ACTION:Push(b, x, y), PRECOND:At(Shakey, x) ∧ Pushable(b),
EFFECT:At(b, y) ∧ At(Shakey, y) ∧ ¬At(b, x) ∧ ¬At(Shakey, x))
Action(ACTION:ClimbUp(b), PRECOND:At(Shakey, x) ∧ At(b, x) ∧ Climbable(b),
EFFECT:On(Shakey, b) ∧ ¬On(Shakey, Floor))
Action(ACTION:ClimbDown(b), PRECOND:On(Shakey, b),
EFFECT:On(Shakey, Floor) ∧ ¬On(Shakey, b))
Action(ACTION:TurnOn(l), PRECOND:On(Shakey, b) ∧ At(Shakey, x) ∧ At(l, x),
EFFECT:TurnedOn(l))
Action(ACTION:TurnOff(l), PRECOND:On(Shakey, b) ∧ At(Shakey, x) ∧ At(l, x),
EFFECT:¬TurnedOn(l))
```

The initial state is:

$$\begin{aligned}
 & In(Switch_1, Room_1) \wedge In(Door_1, Room_1) \wedge In(Door_1, Corridor) \\
 & In(Switch_1, Room_2) \wedge In(Door_2, Room_2) \wedge In(Door_2, Corridor) \\
 & In(Switch_1, Room_3) \wedge In(Door_3, Room_3) \wedge In(Door_3, Corridor) \\
 & In(Switch_1, Room_4) \wedge In(Door_4, Room_4) \wedge In(Door_4, Corridor) \\
 & In(Shakey, Room_3) \wedge At(Shakey, X_S) \\
 & In(Box_1, Room_1) \wedge In(Box_2, Room_1) \wedge In(Box_3, Room_1) \wedge In(Box_4, Room_1) \\
 & Climbable(Box_1) \wedge Climbable(Box_2) \wedge Climbable(Box_3) \wedge Climbable(Box_4) \\
 & Pushable(Box_1) \wedge Pushable(Box_2) \wedge Pushable(Box_3) \wedge Pushable(Box_4) \\
 & At(Box_1, X_1) \wedge At(Box_2, X_2) \wedge At(Box_3, X_3) \wedge At(Box_4, X_4) \\
 & TurnwdOn(Switch_1) \wedge TurnedOn(Switch_4)
 \end{aligned}$$

A plan to achieve the goal is:

$$\begin{aligned}
 & Go(X_S, Door_3) \\
 & Go(Door_3, Door_1) \\
 & Go(Door_1, X_2) \\
 & Push(Box_2, X_2, Door_1) \\
 & Push(Box_2, Door_1, Door_2) \\
 & Push(Box_2, Door_2, Switch_2)
 \end{aligned}$$

Exercise 11.1.#FTUM

A finite Turing machine has a finite one-dimensional tape of cells, each cell containing one of a finite number of symbols. One cell has a read and write head above it. There is a finite set of states the machine can be in, one of which is the accept state. At each time step, depending on the symbol on the cell under the head and the machine's current state, there are a set of actions we can choose from. Each action involves writing a symbol to the cell under the head, transitioning the machine to a state, and optionally moving the head left or right. The mapping that determines which actions are allowed is the Turing machine's program. Your goal is to control the machine into the accept state.

Represent the Turing machine acceptance problem as a planning problem. If you can do this, it demonstrates that determining whether a planning problem has a solution is at least as hard as the Turing acceptance problem, which is PSPACE-hard.

One representation is as follows. We have the predicates:

- a. $HeadAt(c)$: tape head at cell location c , true for exactly one cell.
- b. $State(s)$: machine state is s , true for exactly one cell.
- c. $ValueOf(c, v)$: cell c 's value is v .
- d. $LeftOf(c_1, c_2)$: cell c_1 is one step left from cell c_2 .
- e. $TransitionLeft(s_1, v_1, s_2, v_2)$: the machine in state s_1 upon reading a cell with value v_1 may write value v_2 to the cell, change state to s_2 , and transition to the left.

- f. $TransitionRight(s_1, v_1, s_2, v_2)$: the machine in state s_1 upon reading a cell with value v_1 may write value v_2 to the cell, change state to s_2 , and transition to the right.

The predicates $HeadAt$, $State$, and $ValueOf$ are fluents, the rest are constant descriptions of the machine and its tape. Two actions are required:

```
Action(RunLeft(s1, c1, v1, , s2, c2, v2),
  PRECOND: State(s1) ∧ HeadAt(c1) ∧ ValueOf(c1, v1)
    ; ∧ TransitionLeft(s1, v1, s2, v2) ∧ LeftOf(c2, c1)
  EFFECT: ¬State(s1) ∧ State(s2) ∧ ¬HeadAt(c1) ∧ HeadAt(c2)
    ∧ ¬ValueOf(c1, v1) ∧ ValueOf(c1, v2))
```

```
Action(RunRight(s1, c1, v1, , s2, c2, v2),
  PRECOND: State(s1) ∧ HeadAt(c1) ∧ ValueOf(c1, v1)
    ; ∧ TransitionRight(s1, v1, s2, v2) ∧ LeftOf(c1, c2)
  EFFECT: ¬State(s1) ∧ State(s2) ∧ ¬HeadAt(c1) ∧ HeadAt(c2)
    ∧ ¬ValueOf(c1, v1) ∧ ValueOf(c1, v2))
```

The goal will typically be to reach a fixed accept state. A simple example problem is:

```
Init(HeadAt(C0) ∧ State(S1) ∧ ValueOf(C0, 1) ∧ ValueOf(C1, 1)
  ∧ ValueOf(C2, 1) ∧ ValueOf(C3, 0) ∧ LeftOf(C0, C1) ∧ LeftOf(C1, C2)
  ∧ LeftOf(C2, C3) ∧ TransitionLeft(S1, 1, S1, 0) ∧ TransitionLeft(S1, 0, Saccept, 0)
Goal(State(Saccept))
```

Note that the number of literals in a state is linear in the number of cells, which means a polynomial space machine require polynomial state to represent.

Exercise 11.1.#HOLD

The goals we have considered so far all ask the planner to make the world satisfy the goal at just one time step. Not all goals can be expressed this way: you do not achieve the goal of suspending a chandelier above the ground by throwing it in the air. More seriously, you wouldn't want your spacecraft life-support system to supply oxygen one day but not the next. A *maintenance goal* is achieved when the agent's plan causes a condition to hold continuously from a given state onward. Describe how to extend the formalism of this chapter to support maintenance goals.

The simplest extension allows for maintenance goals that hold in the initial state and must remain true throughout the execution of the plan. Safety goals (do no harm) are typically of this form. This extends classical planning problems to allow a maintenance goal. A plan solves the problem if the final state satisfies the regular goals, and all visited states satisfy the maintenance goal.

The life-support example cannot, however, be solved by a finite plan. An extension to infinite plans can capture this, where an infinite plan solves a planning problem if the goal is eventually satisfied by the plan, i.e., there is a point after which the goal is continuously true. Infinite solutions can be described finitely with loops.

For the chandelier example we can allow NoOp actions which do nothing except model the passing of physics. The idea is that a solution will have a finite prefix with an infinite tail (i.e., a loop) of NoOps. This will allow the problem specification to capture the instability of a thrown chandelier, as after a certain number of time steps it would no longer be suspended.

Exercise 11.1.#PLOP

Some of the operations in standard programming languages can be modeled as actions that change the state of the world. For example, the assignment operation changes the contents of a memory location, and the print operation changes the state of the output stream. A program consisting of these operations can also be considered as a plan, whose goal is given by the specification of the program. Therefore, planning algorithms can be used to construct programs that achieve a given specification.

- a. Write an action schema for the assignment operator (assigning the value of one variable to another). Remember that the original value will be overwritten!
- b. Show how object creation can be used by a planner to produce a plan for exchanging the values of two variables by using a temporary variable.

We need one action, *Assign*, which assigns the value in the source register (or variable if you prefer, but the term “register” makes it clearer that we are dealing with a physical location) *sr* to the destination register *dr*:

Action(ACTION:Assign(dr, sr),
PRECOND: Register(dr) \wedge Register(sr) \wedge Value(dr, dv) \wedge Value(sr, sv),
EFFECT: Value(dr, sv) \wedge \neg Value(dr, dv))

Now suppose we start in an initial state with $Register(R_1) \wedge Register(R_2) \wedge Value(R_1, V_1) \wedge Value(R_2, V_2)$ and we have the goal $Value(R_1, V_2) \wedge Value(R_2, V_1)$. Unfortunately, there is no way to solve this as is. We either need to add an explicit $Register(R_3)$ condition to the initial state, or we need a way to create new registers. That could be done with an action for allocating a new register:

Action(ACTION:Allocate(r),
EFFECT: Register(r))

Then the following sequence of steps constitutes a valid plan:

Allocate(R_3)
Assign(R_3, R_1)
Assign(R_1, R_2)
Assign(R_2, R_1)

11.2 Algorithms for Classical Planning

Exercise 11.2.#SUSS

Figure 11.3 (page 347) shows a blocks-world problem that is known as the **Sussman anomaly**. The problem was considered anomalous because the noninterleaved planners of the early 1970s could not solve it. Write a definition of the problem and solve it, either by hand or with a planning program. A noninterleaved planner is a planner that, when given two subgoals G_1 and G_2 , produces either a plan for G_1 concatenated with a plan for G_2 , or vice versa. Explain why a noninterleaved planner cannot solve this problem.

The initial state is:

$$On(B, Table) \wedge On(C, A) \wedge On(A, Table) \wedge Clear(B) \wedge Clear(C)$$

The goal is:

$$On(A, B) \wedge On(B, C)$$

First we'll explain why it is an anomaly for a noninterleaved planner. There are two subgoals; suppose we decide to work on $On(A, B)$ first. We can clear C off of A and then move A on to B . But then there is no way to achieve $On(B, C)$ without undoing the work we have done. Similarly, if we work on the subgoal $On(B, C)$ first we can immediately achieve it in one step, but then we have to undo it to get A on B .

Now we'll show how things work out with an interleaved planner such as POP. Since $On(A, B)$ isn't true in the initial state, there is only one way to achieve it: $Move(A, x, B)$, for some x . Similarly, we also need a $Move(B, x', C)$ step, for some x' . Now let's look at the $Move(A, x, B)$ step. We need to achieve its precondition $Clear(A)$. We could do that either with $Move(b, A, y)$ or with $MoveToTable(b, A)$. Let's assume we choose the latter. Now if we bind b to C , then all of the preconditions for the step $MoveToTable(C, A)$ are true in the initial state, and we can add causal links to them. We then notice that there is a threat: the $Move(B, x', C)$ step threatens the $Clear(C)$ condition that is required by the $MoveToTable$ step. We can resolve the threat by ordering $Move(B, x', C)$ after the $MoveToTable$ step. Finally, notice that all the preconditions for $Move(B, x', C)$ are true in the initial state. Thus, we have a complete plan with all the preconditions satisfied. It turns out there is a well-ordering of the three steps:

$$\begin{aligned} & MoveToTable(C, A) \\ & Move(B, Table, C) \\ & Move(A, Table, B) \end{aligned}$$

Exercise 11.2.#BSPD

Prove that backward search with PDDL problems is complete.

Briefly, the reason is the same as for forward search: in the absence of function symbols, a PDDL state space is finite. Hence any complete search algorithm will be complete for PDDL planning, whether forward or backward.

Exercise 11.2.#BIDP

Examine the definition of **bidirectional search** in Chapter 3.

- Would bidirectional state-space search be a good idea for planning?
- What about bidirectional search in the space of partial-order plans?
- Devise a version of partial-order planning in which an action can be added to a plan if its preconditions can be achieved by the effects of actions already in the plan. Explain how to deal with conflicts and ordering constraints. Is the algorithm essentially identical to forward state-space search?

- It is feasible to use bidirectional search, because it is possible to invert the actions. However, most of those who have tried have concluded that bidirectional search is generally not efficient, because the forward and backward searches tend to miss each other. This is due to the large state space. A few planners, such as PRODIGY (Fink and Blythe, 1998) have used bidirectional search.
- Again, this is feasible but not popular. PRODIGY is in fact (in part) a partial-order planner: in the forward direction it keeps a total-order plan (equivalent to a state-based planner), and in the backward direction it maintains a tree-structured partial-order plan.
- An action A can be added if all the preconditions of A have been achieved by other steps in the plan. When A is added, ordering constraints and causal links are also added to make sure that A appears after all the actions that enabled it and that a precondition is not disestablished before A can be executed. The algorithm does search forward, but it is not the same as forward state-space search because it can explore actions in parallel when they don't conflict. For example, if A has three preconditions that can be satisfied by the non-conflicting actions B , C , and D , then the solution plan can be represented as a single partial-order plan, while a state-space planner would have to consider all $3!$ permutations of B , C , and D .

Exercise 11.2.#FBSS

We contrasted forward and backward state-space searchers with partial-order planners, saying that the latter is a plan-space searcher. Explain how forward and backward state-space search can also be considered plan-space searchers, and say what the plan refinement operators are.

A forward state-space planner maintains a partial plan that is a strict linear sequence of actions; the plan refinement operator is to add an applicable action to the end of the sequence, updating literals according to the action's effects.

A backward state-space planner maintains a partial plan that is a reversed sequence of actions; the refinement operator is to add an action to the beginning of the sequence as long as the action's effects are compatible with the state at the beginning of the sequence.

Exercise 11.2.#SATX

Up to now we have assumed that the plans we create always make sure that an action's preconditions are satisfied. Let us now investigate what propositional successor-state axioms such as $\text{HaveArrow}^{t+1} \Leftrightarrow (\text{HaveArrow}^t \wedge \neg \text{Shoot}^t)$ have to say about actions whose preconditions are not satisfied.

- Show that the axioms predict that nothing will happen when an action is executed in a state where its preconditions are not satisfied.
- Consider a plan p that contains the actions required to achieve a goal but also includes illegal actions. Is it the case that

$$\text{initial state} \wedge \text{successor-state axioms} \wedge p \models \text{goal} ?$$

- With first-order successor-state axioms in situation calculus, is it possible to prove that a plan containing illegal actions will achieve the goal?

- We can illustrate the basic idea using the axiom given. Suppose that Shoot^t is true but HaveArrow^t is false. Then the RHS of the axiom is false, so HaveArrow^{t+1} is false, as we would hope. More generally, if an action precondition is violated, then both ActionCausesF^t and $\text{ActionCausesNotF}^t$ are false, so the generic successor-state axiom reduces to

$$F^{t+1} \Leftrightarrow \text{False} \vee (F^t \wedge \text{True}) .$$

which is the same as saying $F^{t+1} \Leftrightarrow F^t$, i.e., nothing happens.

- Yes, the plan plus the axioms will entail goal satisfaction; the axioms will copy every fluent across an illegal action and the rest of the plan will still work. Note that goal entailment is trivially achieved if we add precondition axioms, because then the plan is logically inconsistent with the axioms and every sentence is entailed by a contradiction. Precondition axioms are a way to *prevent* illegal actions in satisfiability-based planning methods.
- No. As written in Section 10.4.2, the successor-state axioms preclude proving anything about the outcome of a plan with illegal actions. When $\text{Poss}(a, s)$ is false, the axioms say nothing about the situation resulting from the action.

Exercise 11.2.#STTR

Consider how to translate a set of action schemas into the successor-state axioms of situation calculus.

- Consider the schema for $Fly(p, from, to)$. Write a logical definition for the predicate $Poss(Fly(p, from, to), s)$, which is true if the preconditions for $Fly(p, from, to)$ are satisfied in situation s .
- Next, assuming that $Fly(p, from, to)$ is the only action schema available to the agent, write down a successor-state axiom for $At(p, x, s)$ that captures the same information as the action schema.
- Now suppose there is an additional method of travel: $Teleport(p, from, to)$. It has the additional precondition $\neg Warped(p)$ and the additional effect $Warped(p)$. Explain how the situation calculus knowledge base must be modified.
- Finally, develop a general and precisely specified procedure for carrying out the translation from a set of action schemas to a set of successor-state axioms.

The main point here is that writing each successor-state axiom correctly requires knowing *all* the actions that might add or delete a given fluent; writing a STRIPS axiom, on the other hand, requires knowing *all* the fluents that a given action might add or delete.

a.

$$\begin{aligned} Poss(Fly(p, from, to), s) \Leftrightarrow \\ At(p, from, s) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to) . \end{aligned}$$

b.

$$\begin{aligned} Poss(a, s) \Rightarrow \\ (At(p, to, Result(a, s)) \Leftrightarrow \\ (\exists from \ a = Fly(p, from, to)) \vee \\ (At(p, to, s) \wedge \neg \exists new \ new \neq to \wedge a = Fly(p, to, new))) . \end{aligned}$$

c. We must add the possibility axiom for the new action:

$$\begin{aligned} Poss(Teleport(p, from, to), s) \Leftrightarrow \\ At(p, from, s) \wedge \neg Warped(p, s) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to) . \end{aligned}$$

The successor-state axiom for location must be revised:

$$\begin{aligned} Poss(a, s) \Rightarrow \\ (At(p, to, Result(a, s)) \Leftrightarrow \\ (\exists from \ a = Fly(p, from, to)) \vee \\ (\exists from \ a = Teleport(p, from, to)) \vee \\ (At(p, to, s) \wedge \neg \exists new \ new \neq to \wedge \\ (a = Fly(p, to, new) \vee a = Teleport(p, to, new)))) . \end{aligned}$$

Finally, we must add a successor-state axiom for *Warped*:

$$\begin{aligned} \text{Poss}(a, s) \Rightarrow \\ (\text{Warped}(p, \text{Result}(a, s)) \Leftrightarrow \\ (\exists \text{from}, \text{to } a = \text{Teleport}(p, \text{from}, \text{to})) \vee \text{Warped}(p, s)) . \end{aligned}$$

- d. The basic procedure is essentially given in the description of classical planning as Boolean satisfiability in 10.4.1, except that there is no grounding step, the precondition axioms become definitions of *Poss* for each action, and the successor-state axioms use the structure given in 10.4.2 with existential quantifiers for all free variables in the actions, as shown in the examples above.

Exercise 11.2.#SATD

In the SATPLAN algorithm in Figure 7.22 (page 244), each call to the satisfiability algorithm asserts a goal g^T , where T ranges from 0 to T_{\max} . Suppose instead that the satisfiability algorithm is called only once, with the goal $g^0 \vee g^1 \vee \dots \vee g^{T_{\max}}$.

- a. Will this always return a plan if one exists with length less than or equal to T_{\max} ?
 - b. Does this approach introduce any new spurious “solutions”?
 - c. Discuss how one might modify a satisfiability algorithm such as WALKSAT so that it finds short solutions (if they exist) when given a disjunctive goal of this form.
-
- a. Yes, this will find a plan whenever the normal SATPLAN finds a plan no longer than T_{\max} .
 - b. This will not cause SATPLAN to return an incorrect solution, but it might lead to plans that, for example, achieve and unachieve the goal several times.
 - c. There is no simple and clear way to induce WALKSAT to find short solutions, because it has no notion of the length of a plan—the fact that the problem is a planning problem is part of the encoding, not part of WALKSAT. But if we are willing to do some rather brutal surgery on WALKSAT, we can achieve shorter solutions by identifying the variables that represent actions and (1) tending to randomly initialize the action variables (particularly the later ones) to false, and (1) preferring to randomly flip an earlier action variable rather than a later one.

11.3 Heuristics for Planning

Exercise 11.3.#NEGE

Explain why dropping negative effects from every action schema in a planning problem results in a relaxed problem.

Goals and preconditions can only be positive literals. So a negative effect can only make it harder to achieve a goal (or a precondition to an action that achieves the goal). There-

fore, eliminating all negative effects only makes a problem easier. This would *not* be true if negative preconditions and goals were allowed.

11.4 Hierarchical Planning

Exercise 11.4.#FDEX

You have a number of trucks with which to deliver a set of packages. Each package starts at some location on a grid map, and has a destination somewhere else. Each truck is directly controlled by moving forward and turning. Construct a hierarchy of high-level actions for this problem. What knowledge about the solution does your hierarchy encode?

We first need to specify the primitive actions: for movement we have $Forward(t)$, $TurnLeft(t)$, and $TurnRight(t)$ where t is a truck, and for package delivery we have $Load(p, t)$ and $Unload(p, t)$ where p is a package and t is a truck. These can be given PDDL descriptions in the usual way.

The hierarchy can be built in a number of ways, but one is to use the HLA $Navigate(t, [x, y])$ to take a truck t to coordinates $[x, y]$, and $Deliver(t, p)$ to deliver package p to its destination with truck t . We assume the fluent $At(o, [x, y])$ for trucks and packages o records their current position $[x, y]$, the predicate $Destination(p, [x', y'])$ gives the package's destination.

This hierarchy (Figure S??) encodes the knowledge that trucks can only carry one package at a time, that we need only drop packages off at their destinations not intermediate points, and that we can serialize deliveries (in reality, trucks would move in parallel, but we have no representation for parallel actions here). From a higher-level, the hierarchy says that the planner needs only to choose which trucks deliver which packages in what order, and trucks should navigate given their destinations.

Exercise 11.4.#HLAU

Suppose that a high-level action has exactly one implementation as a sequence of primitive actions. Give an algorithm for computing its preconditions and effects, given the complete refinement hierarchy and schemas for the primitive actions.

To simplify the problem, we assume that at most one refinement of a high-level action will be applicable at a given time (not much of a restriction since there is a unique solution).

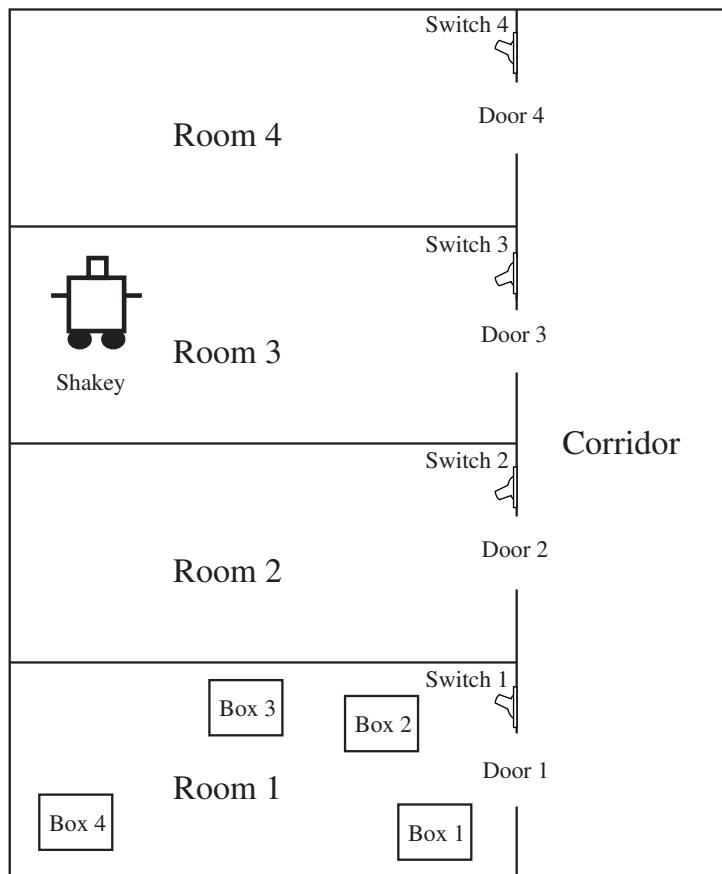
The algorithm shown below maintains at each point the net preconditions and effects of the prefix of h processed so far. This includes both preconditions and effects of primitive actions, and preconditions of refinements. Note that any literal not in effect is untouched by the prefix currently processed.

```

net_preconditions <- {}
net_effects <- {}
remaining <- [h]

while remaining not empty:
    a <- pop remaining

```



Shakey's world. Shakey can move between landmarks within a room, can pass through the door between rooms, can climb climbable objects and push pushable objects, and can flip light switches.

```

if a is primitive:
    add to net_preconditions any precondition of a not in effects
    add to net_effects the effects of action a, first removing any
        complementary literals
else:
    r <- the unique refinement whose preconditions do not include
        literals negated in net_effect or net_preconditions
    add to net_preconditions any preconditions of r not in effect
    prepend to remaining the sequence of actions in r
  
```

Exercise 11.4.#OPTR

Suppose that the optimistic reachable set of a high-level plan is a superset of the goal set; can anything be concluded about whether the plan achieves the goal? What if the pessimistic reachable set doesn't intersect the goal set? Explain.

We cannot draw any conclusions. Just knowing that the optimistic reachable set is a superset of the goal is no more help than knowing only that it intersects the goal: the optimistic reachable set only guarantees that we cannot reach states outside of it, not that we can reach any of the states inside it. Similarly, the pessimistic reachable set only says we can definitely reach state inside of it, not that we cannot reach states outside of it.

Exercise 11.4.#HLAP

Write an algorithm that takes an initial state (specified by a set of propositional literals) and a sequence of HLAs (each defined by preconditions and angelic specifications of optimistic and pessimistic reachable sets) and computes optimistic and pessimistic descriptions of the reachable set of the sequence.

To simplify, we don't model HLA precondition tests. (Comparing the preconditions to the optimistic and pessimistic descriptions can sometimes determine if preconditions are definitely or definitely not satisfied, respectively, but may be inconclusive.)

The operation to propagate 1-CNF descriptions through descriptions is the same for optimistic and pessimistic descriptions, and is as follows:

```

state <- initial state

for each HLA h in order:
    for each literal in the description of h:
        choose case depending on form of literal:
            +1:           state <- state - {-l} + {l}
            -1:           state <- state - {l} + {-l}
            poss add l:   state <- state + {l}
            poss del l:   state <- state + {-l}
            poss add del l: state <- state + {l,-l}

description <- conjunction of all literals which are
                not part of a complementary pair in state

```

11.5 Planning and Acting in Nondeterministic Domains

Exercise 11.5.#NDTE

Consider the following argument: In a framework that allows uncertain initial states, **nondeterministic effects** are just a notational convenience, not a source of additional representational power. For any action schema a with nondeterministic effect $P \vee Q$, we could always replace it with the conditional effects **when** R : $P \wedge$ **when** $\neg R$: Q , which in turn can be reduced to two regular actions. The proposition R stands for a random proposition that is unknown in the initial state and for which there are no sensing actions. Is this argument correct? Consider separately two cases, one in which only one instance of action schema a is in the plan, the other in which more than one instance is.

It is equivalent in the first case, by the argument given above. However, if there are two or more copies of the same schema in a plan it is different: all actions will have the same

nondeterministic effect: if R is true both will result in P , otherwise both result in Q . To allow copies of the same schema to differ we need one random variable per copy.

Exercise 11.5.#FLIP

Suppose the *Flip* action always changes the truth value of variable L . Show how to define its effects by using an action schema with conditional effects. Show that, despite the use of conditional effects, a 1-CNF belief state representation remains in 1-CNF after a *Flip*.

Flip can be described using conditional effects:

Action(Flip,
EFFECT:when $L: \neg L \wedge \text{when } \neg L: L$) .

To see that a 1-CNF belief state representation stays 1-CNF after *Flip*, observe that there are three cases. If L is true in the belief state, then it is false after *Flip*; conversely if it is false. Finally, if L is unknown before, then it is unknown after: either L or $\neg L$ can obtain. All other components of the belief state remain unchanged, since it is 1-CNF.

Exercise 11.5.#BLKA

In the blocks world we were forced to introduce two action schemas, *Move* and *MoveToTable*, in order to maintain the *Clear* predicate properly. Show how conditional effects can be used to represent both of these cases with a single action.

Using the second definition of *Clear* in the chapter—namely, that there is a clear space for a block—the only change is that the destination remains clear if it is the table:

Action(Move(b, x, y),
PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y)$,
EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge (\text{when } y \neq Table: \neg Clear(y))$)

Exercise 11.5.#ALTV

Conditional effects were illustrated for the *Suck* action in the vacuum world—which square becomes clean depends on which square the robot is in. Can you think of a new set of propositional variables to define states of the vacuum world, such that *Suck* has an *unconditional* description? Write out the descriptions of *Suck*, *Left*, and *Right*, using your propositions, and demonstrate that they suffice to describe all possible states of the world.

Let $CleanH$ be true iff the robot's current square is clean and $CleanO$ be true iff the other square is clean. Then *Suck* is characterized by

Action(Suck, PRECOND:, EFFECT: $CleanH$)

Unfortunately, moving affects these new literals! For *Left* we have

$$\begin{aligned} \text{Action}(\text{Left}, \text{PRECOND}: & \text{AtR}, \\ \text{EFFECT}: & \text{AtL} \wedge \neg \text{AtR} \wedge \text{when } \text{CleanH}: \text{CleanO} \wedge \text{when } \neg \text{CleanO}: \text{CleanH} \\ & \wedge \text{when } \neg \text{CleanO}: \neg \text{CleanH} \wedge \text{when } \neg \text{CleanH}: \neg \text{CleanO}) \end{aligned}$$

with the dual for *Right*.

Exercise 11.5.#DIRT

Find a suitably dirty carpet, free of obstacles, and vacuum it. Draw the path taken by the vacuum cleaner as accurately as you can. Explain it, with reference to the forms of planning discussed in this chapter.

The main thing to notice here is that the vacuum cleaner moves repeatedly over dirty areas—presumably, until they are clean. Also, each forward move is typically short, followed by an immediate reversing over the same area. This is explained in terms of a disjunctive outcome: the area may be fully cleaned or not, the reversing enables the agent to check, and the repetition ensures completion (unless the dirt is ingrained). Thus, we have a strong cyclic plan with sensing actions.

Exercise 11.5.#SHAM

The following quotes are from the backs of shampoo bottles. Identify each as an unconditional, conditional, or execution-monitoring plan. (a) “Lather. Rinse. Repeat.” (b) “Apply shampoo to scalp and let it remain for several minutes. Rinse and repeat if necessary.” (c) “See a doctor if problems persist.”

- (a) Literally its an unconditional plan, but generally people would interpret it similarly to (b).
- (b) Conditional: the final action loops to the start only if “necessary”.
- (c) This can be seen as a conditional plan. But it can also be seen as execution monitoring: we continue treatment until the problem resolves itself. If we notice the problem persisting, we complete the plan.

Exercise 11.5.#DOCO

Consider the following problem: A patient arrives at the doctor’s office with symptoms that could have been caused either by dehydration or by disease *D* (but not both). There are two possible actions: *Drink*, which unconditionally cures dehydration, and *Medicate*, which cures disease *D* but has an undesirable side effect if taken when the patient is dehydrated. Write the problem description, and diagram a sensorless plan that solves the problem, enumerating all relevant possible worlds.

The two actions can be represented as:

$$\begin{aligned} \text{Action}(\text{Drink}, & \\ \text{EFFECT:when } \neg D: \text{Cured}) . \\ \text{Action}(\text{Medicate}, & \\ \text{EFFECT:when } D: \text{Cured} \wedge \text{when } D: \text{SideEffect}) . \end{aligned}$$

The goal is $\text{Cured} \wedge \neg \text{SideEffect}$.

The plan $[\text{Drink}, \text{Medicate}]$ solves the problem. To see this, note that the relevant possible worlds before executing the plan are $\{D, \neg D\}$. After executing *Drink* the worlds are $\{D, \neg D \wedge \text{Cured}\}$ and after *Medicate* they are $\{D \wedge \text{Cured}, \neg D \wedge \text{Cured}\}$.

Exercise 11.5.#MEDI

To the medication problem in the previous exercise, add a *Test* action that has the conditional effect *CultureGrowth* when *Disease* is true and in any case has the perceptual effect *Known(CultureGrowth)*. Diagram a conditional plan that solves the problem and minimizes the use of the *Medicate* action.

One solution plan is $[\text{Test, ifCultureGrowth then } [\text{Drink, Medicate}]]$.

11.6 Time, Schedules, and Resources

Exercise 11.6.#CPMJ

In Figure 11.14 we showed how to describe actions in a scheduling problem by using separate fields for DURATION, USE, and CONSUME. Now suppose we wanted to combine scheduling with nondeterministic planning, which requires nondeterministic and conditional effects. Consider each of the three fields and explain if they should remain separate fields, or if they should become effects of the action. Give an example for each of the three.

The natural nondeterministic generalization of DURATION, USE, and CONSUME represents each as an *interval* of possible values rather than a single value. Algorithms that work with quantities can all be modified relatively easily to manage intervals over quantities—for example, by representing them as inequalities for the lower and upper bounds. Thus, if the agent starts with 10 screws and the first action in a plan consumes 2–4 screws, then a second action requiring 5 screws is still executable.

When it comes to conditional effects, however, the fields must be treated differently. The USE field refers to a constraint holding *during* the action, rather than *after* it is done. Thus, it has to remain a separate field, since it is not treated in the same way as an effect. The DURATION and CONSUME fields both describe effects (on the clock and on the quantity of a resource); thus, they can be folded into the conditional effect description for the action.

[[need exercises]]

11.7 Analysis of Planning Approaches

[[need exercises]]

EXERCISES 12

QUANTIFYING UNCERTAINTY

12.1 Acting under Uncertainty

Exercise 12.1.#XXXX

The chapter proposes probability theory as the basis for reasoning under uncertainty and provides one argument based on work by de Finetti. Compare and contrast de Finetti's argument with other arguments by Ramsey, Cox, Savage, Jeffrey, and Jaynes (see the historical and bibliographical notes for references). Also consider any rebuttals you can find.

All offer arguments premised on an individual making bets proportional to their degree of belief.

Ramsey (1931) introduced the terminology of “degrees of belief” by imagining a rational individual making decisions. Such a decision must be based on the utility of an outcome to an agent, the evidence available to the agent, and the agent’s degrees of belief in the outcomes available. His formulation considered “possible worlds” to ensue based on which propositions came out true.

de Finetti (1937) validates the axioms of probability as identified by Kolmogorov (1933) through the betting argument described in this text. He also considers expected utility like Ramsey, but does not go so far as Ramsey, Savage, and later thinkers to consider an agent’s degrees of belief as ascertainable from their decisions.

Cox (1946) with his consistency (or rationality) theorems, proves that any system of reasoning over uncertainty which admits his assumptions is equivalent to probability theory.

Savage (1954) offers an alternative decision theoretic formulation to that of Ramsey, one encompassing states, consequences, acts, events, and preferences (relative to an agent) over acts. A few of his assumptions, regarding the relation between consequences and acts and events and acts, later researchers found to be limiting.

Jeffrey (1983) demonstrates the analytic superiority of the Bayesian, degrees of belief interpretation in light of various failures of a frequentist approach by use of an agent expressing preferences and confidences. His approach suffers (or allows) a non-uniqueness problem in which the decisions of an agent might be shown to adhere to multiple formulations of expected utility.

Jaynes (2003) provides “a deeper logical foundation” of Kolmogorov’s axioms, avoiding the infinite sets which paradoxically plague the formulation of de Finetti. Much of his text allows for diverse interpretation and he contrasts the Bayesian approach as expressing a prior on an agent’s belief state (such as a model of the world) which can alternatively be ignored by assuming maximum entropy over the space of consideration.

12.2 Basic Probability Notation

Exercise 12.2.#PAGB

Show from first principles that $P(a | b \wedge a) = 1$.

The “first principles” needed here are the definition of conditional probability, $P(X|Y) = P(X \wedge Y)/P(Y)$, and the definitions of the logical connectives. It is not enough to say that if $B \wedge A$ is “given” then A must be true! From the definition of conditional probability, and the fact that conjunction is commutative, associative, and idempotent, we have

$$P(A|B \wedge A) = \frac{P(A \wedge (B \wedge A))}{P(B \wedge A)} = \frac{P(B \wedge A)}{P(B \wedge A)} = 1$$

Exercise 12.2.#SUMO

- a. Suppose a sample space is defined by the Cartesian product of the ranges of a set of Boolean variables X_1, \dots, X_n , and let ϕ be any logical proposition expressed in terms of these variables. Prove from first principles (including the basic axioms of probability, Equation (12.1)) that Equation (12.2), i.e., $P(\phi) = \sum_{\omega \in \phi} P(\omega)$.
- b. Now show that $\sum_{x_i} P(X_i = x_i) = 1$ for any variable X_i , i.e., the distribution for each random variable must sum to 1.

- a. Assuming the propositions are independent, that there is a unique proposition which denotes the assignment to the random variables,

$$P(\phi) = P(\bigcup_{\omega \in \phi} \omega) = \sum_{\omega \in \phi} P(\omega)$$

- b. As is the case for a random variable, each assignment is mutually exclusive and exhaustive; therefore each assignment is a possible world, ω , of the total sample space, Ω , of the random variable.

$$\sum_{x_i} P(X = x_i) = \sum_{\omega} P(X = \omega) = \sum_{\omega \in \Omega} P(\omega) = 1$$

Exercise 12.2.#INEX

Prove Equation (12.5) from Equations (12.1) and (12.2).

Equation (12.5) states that $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$. This can be proved directly from Equation (12.2), using obvious abbreviations for the possible-world probabilities:

$$P(a \vee b) = p_{a,b} + p_{a,\neg b} + p_{\neg a,b}$$

$$P(a) = p_{a,b} + p_{a,\neg b}$$

$$P(b) = p_{a,b} + p_{\neg a,b}$$

$$P(a \wedge b) = p_{a,b}.$$

Exercise 12.2.#TFPG

For each of the following statements, either prove it is true or give a counterexample.

- a. If $P(a | b, c) = P(b | a, c)$, then $P(a | c) = P(b | c)$
- b. If $P(a | b, c) = P(a)$, then $P(b | c) = P(b)$
- c. If $P(a | b) = P(a)$, then $P(a | b, c) = P(a | c)$

- a. True. By the product rule we know $P(b, c)P(a|b, c) = P(a, c)P(b|a, c)$, which by assumption reduces to $P(b, c) = P(a, c)$. Dividing through by $P(c)$ gives the result.
- b. False. The statement $P(a|b, c) = P(a)$ merely states that a is independent of b and c , it makes no claim regarding the dependence of b and c . A counter-example: a and b record the results of two independent coin flips, and $c = b$.
- c. False. While the statement $P(a|b) = P(a)$ implies that a is independent of b , it does not imply that a is conditionally independent of b given c . A counter-example: a and b record the results of two independent coin flips, and c equals the xor of a and b .

Exercise 12.2.#RATB

Would it be rational for an agent to hold the three beliefs $P(A) = 0.4$, $P(B) = 0.3$, and $P(A \vee B) = 0.5$? If so, what range of probabilities would be rational for the agent to hold for $A \wedge B$? Make up a table like the one in Figure 12.2, and show how it supports your argument about rationality. Then draw another version of the table where $P(A \vee B) = 0.7$. Explain why it is rational to have this probability, even though the table shows one case that is a loss and three that just break even. (*Hint:* what is Agent 1 committed to about the probability of each of the four cases, especially the case that is a loss?)

Probably the easiest way to keep track of what's going on is to look at the probabilities of the atomic events. A probability assignment to a set of propositions is consistent with the axioms of probability if the probabilities are consistent with an assignment to the atomic events that sums to 1 and has all probabilities between 0 and 1 inclusive. We call the probabilities of the atomic events w , x , y , and z , as follows:

	B	$\neg B$
A	w	x
$\neg A$	y	z

We then have the following equations:

$$\begin{aligned}P(A) &= w + x = 0.4 \\P(B) &= w + y = 0.3 \\P(A \vee B) &= w + x + Y = 0.5 \\P(\text{True}) &= w + x + y + z = 1\end{aligned}$$

From these, it is straightforward to infer that $w = 0.2$, $x = 0.2$, $y = 0.1$, and $z = 0.5$. Therefore, $P(A \wedge B) = w = 0.2$. Thus the probabilities given are consistent with a rational assignment, and the probability $P(A \wedge B)$ is exactly determined.

If $P(A \vee B) = 0.7$, then $P(A \wedge B) = w = 0$. Thus, even though the bet outlined in Figure 12.2 loses if A and B are both true, the agent believes this to be impossible so the bet is still rational.

Exercise 12.2.#EXEX

This question deals with the properties of possible worlds, defined on page 392 as assignments to all random variables. We will work with propositions that correspond to exactly one possible world because they pin down the assignments of all the variables. In probability theory, such propositions are called **atomic events**. For example, with Boolean variables X_1 , X_2 , X_3 , the proposition $x_1 \wedge \neg x_2 \wedge \neg x_3$ is an atomic event because it fixes the assignment of the variables; in the language of propositional logic, we would say it has exactly one model.

- a. Prove, for the case of n Boolean variables, that any two distinct atomic events are mutually exclusive; that is, their conjunction is equivalent to *false*.
- b. Prove that the disjunction of all possible atomic events is logically equivalent to *true*.
- c. Prove that any proposition is logically equivalent to the disjunction of the atomic events that entail its truth.

- a. Each atomic event is a conjunction of n literals, one per variable, with each literal either positive or negative. For the events to be distinct, at least one pair of corresponding literals must be nonidentical; hence, the conjunction of the two events contains the literals X_i and $\neg X_i$ for some i , so the conjunction reduces to *False*.
- b. Proof by induction on n . For $n = 0$, the only event is the empty conjunction *True*, and the disjunction containing only this event is also *True*. Inductive step: assume the claim holds for n variables. The disjunction for $n + 1$ variables consists of pairs of disjuncts of the form $(T_n \wedge X_{n+1}) \vee (T_n \wedge \neg X_{n+1})$ for all possible atomic event conjunctions T_n . Each pair logically reduces to T_n , so the entire disjunction reduces to the disjunction for n variables, which by hypothesis is equivalent to *True*.
- c. Let α be the sentence in question and μ_1, \dots, μ_k be the atomic event sentences that entail its truth. Let M_i be the model corresponding to μ_i (its *only* model). To prove that $\mu_1 \vee \dots \vee \mu_k \equiv \alpha$, simply observe the following:
 - Because $\mu_i \models \alpha$, α is true in all the models of μ_i , so α is true in M_i .
 - The models of $\mu_1 \vee \dots \vee \mu_k$ are exactly M_1, \dots, M_k because any two atomic events are mutually exclusive, so any given model can satisfy at most one disjunct, and

a model that satisfies a disjunct must be the model corresponding to that atomic event.

- If any model M satisfies α , then the corresponding atomic-event sentence μ entails α , so the models of α are exactly M_1, \dots, M_k .

Hence, α and $\mu_1 \vee \dots \vee \mu_k$ have the same models, so are logically equivalent.

Exercise 12.2.#POFK

Consider the set of all possible five-card poker hands dealt fairly from a standard deck of fifty-two cards.

- How many atomic events are there in the joint probability distribution (i.e., how many five-card hands are there)?
- What is the probability of each atomic event?
- What is the probability of being dealt a royal straight flush? Four of a kind?

This is a classic combinatorics question that could appear in a basic text on discrete mathematics. The point here is to refer to the relevant axioms of probability. The question also helps students to grasp the concept of the joint probability distribution as the distribution over all possible states of the world.

- There are $\binom{52}{5} = (52 \times 51 \times 50 \times 49 \times 48)/(1 \times 2 \times 3 \times 4 \times 5) = 2,598,960$ possible five-card hands. Note that the order of the cards does not matter.
- By the fair-dealing assumption, each of these is equally likely. By Equation (12.1), each hand therefore occurs with probability $1/2,598,960$.
- There are four hands that are royal straight flushes (one in each suit). By Equation (12.2), since the events are mutually exclusive, the probability of a royal straight flush is just the sum of the probabilities of the atomic events, i.e., $4/2,598,960 = 1/649,740$. For “four of a kind” events, there are 13 possible “kinds” and for each, the fifth card can be one of 48 possible other cards. The total probability is therefore $(13 \times 48)/2,598,960 = 1/4,165$.

These questions can easily be augmented by more complicated ones, e.g., what is the probability of getting a full house given that you already have two pairs? What is the probability of getting a flush given that you have three cards of the same suit? Or you could assign a project of producing a poker-playing agent, and have a tournament among them. Note that poker play (mostly betting) is complicated by the game-theoretic nature of the problem. For example, a player who bets a large amount may be bluffing. See Chapter 18.

Exercise 12.2.#PASC

In his letter of August 24, 1654, Pascal was trying to show how a pot of money should be allocated when a gambling game must end prematurely. Imagine a game where each turn consists of the roll of a die, player E gets a point when the die is even, and player O gets a point when the die is odd. The first player to get 7 points wins the pot. Suppose the game is

interrupted with E leading 4–2. How should the money be fairly split in this case? What is the general formula? (Fermat and Pascal made several errors before solving the problem, but you should be able to get it right the first time.)

Let e and o be the initial scores, m be the score required to win, and p be the probability that E wins each round. One can easily write down a recursive formula for the probability that E wins from the given initial state:

$$w_E(p, e, o, m) = \begin{cases} 1 & \text{if } e = m \\ 0 & \text{if } o = m \\ p \cdot w_E(p, e + 1, o, m) + (1 - p) \cdot w_E(p, e, o + 1, m) & \text{otherwise} \end{cases}$$

This translates directly into code that can be used to compute the answer,

$$w_E(0.5, 4, 2, 7) = 0.7734375 .$$

With a bit more work, we can derive a nonrecursive formula:

$$w_E(p, e, o, m) = p^{m-e} \sum_{i=0}^{m-o-1} \binom{i+m-e-1}{i} (1-p)^i .$$

Each term in the sum corresponds to the probability of winning by exactly a particular score; e.g., starting from 4–2, one can win by 7–2, 7–3, 7–4, 7–5, or 7–6. Each final score requires E to win exactly $m-e$ rounds while the opponent wins exactly i rounds, where $i = 0, 1, \dots, m-o-1$; and the combinatorial term counts the number of ways this can happen without E winning first by a larger margin. One can check the nonrecursive formula by showing that it satisfies the recursive formula. (It may be helpful to suggest to students that they start by building the lattice of states implied by the above recursive formula and calculating (bottom-up) the symbolic win probabilities in terms of p rather than 0.5, so that they can see the general shape emerging.)

Exercise 12.2.#XXXX

In this question we consider conditional distributions for binary random variables, expressed as tables.

- a. A , B , C , and D are binary random variables. How many entries are in the following conditional probability tables and what is the sum of the values in each table?
 - (i) $\mathbf{P}(A | C)$
 - (ii) $\mathbf{P}(A, D | B = \text{true}, C = \text{true})$
 - (iii) $\mathbf{P}(B | A = \text{true}, C, D)$
- b. Consider the conditional distribution $\mathbf{P}(X_1, \dots, X_\ell | Y_1, \dots, Y_m, Z_1 = z_1, \dots, Z_n = z_n)$, represented as a complete table. Assuming that all variables are binary, derive expressions for the *number* of the probabilities in the table and their *sum*.

- a. (i) $\mathbf{P}(A | C)$: 4 and 2.
 (ii) $\mathbf{P}(A, D | B = \text{true}, C = \text{true})$: 4 and 1.
 (iii) $\mathbf{P}(B | A = \text{true}, C, D)$: 8 and 4.
- b. Because the Z-variables are all instantiated, they play no role. There are 2^m “rows” of the table, each corresponding to an assignment to the Y-variables, and each row has 2^ℓ entries for the possible assignments to the X-variables, so there are $2^{\ell+m}$ entries in all. Each row must sum to one, being a distribution over the X-variables, so the sum is 2^m .

12.3 Inference Using Full Joint Distributions

Exercise 12.3.#XXXX

Given the full joint distribution shown in Figure 12.3, calculate the following:

- a. $\mathbf{P}(\text{toothache})$.
- b. $\mathbf{P}(\text{Cavity})$.
- c. $\mathbf{P}(\text{Toothache} | \text{cavity})$.
- d. $\mathbf{P}(\text{Cavity} | \text{toothache} \vee \text{catch})$.

The main point of this exercise is to understand the various notations of bold versus non-bold P, and uppercase versus lowercase variable names. The rest is easy, involving a small matter of addition.

- a. This asks for the probability that *Toothache* is true.

$$\mathbf{P}(\text{toothache}) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$$

- b. This asks for the vector of probability values for the random variable *Cavity*. It has two values, which we list in the order $\langle \text{true}, \text{false} \rangle$. First add up $0.108 + 0.012 + 0.072 + 0.008 = 0.2$. Then we have

$$\mathbf{P}(\text{Cavity}) = \langle 0.2, 0.8 \rangle.$$

- c. This asks for the vector of probability values for *Toothache*, given that *Cavity* is true.

$$\mathbf{P}(\text{Toothache} | \text{cavity}) = \langle (0.108 + 0.012) / 0.2, (0.072 + 0.008) / 0.2 \rangle = \langle 0.6, 0.4 \rangle$$

- d. This asks for the vector of probability values for *Cavity*, given that either *Toothache* or *Catch* is true. First compute $P(\text{toothache} \vee \text{catch}) = 0.108 + 0.012 + 0.016 + 0.064 + 0.072 + 0.144 = 0.416$. Then

$$\begin{aligned} \mathbf{P}(\text{Cavity} | \text{toothache} \vee \text{catch}) &= \\ &\langle (0.108 + 0.012 + 0.072) / 0.416, (0.016 + 0.064 + 0.144) / 0.416 \rangle = \\ &\langle 0.4615, 0.5384 \rangle \end{aligned}$$

12.4 Independence

Exercise 12.4.#XXXX

Find values for the probabilities a and b in joint probability table below so that the binary variables X and Y are independent.

X	Y	$P(X, Y)$
t	t	$3/5$
t	f	$1/5$
f	t	a
f	f	b

First, we know that the distribution sums to 1, so $a + b = 1/5$. Second, independence requires that $\mathbf{P}(X, Y) = \mathbf{P}(X)\mathbf{P}(Y)$. From the first two rows, where the value of X is the same, we have that $P(Y = t)/P(Y = f) = 3$. In the third and fourth row, the value of X is the same, so $a/b = 3$ also. Hence, $a = 3/20$ and $b = 1/20$.

Exercise 12.4.#XXXX

Suppose X and Y are *independent* random variables over the domain $\{1, 2, 3\}$ with $P(X = 3) = 1/6$. Given the following partially specified joint distribution, what are the remaining values? Write your answers as simplified fractions in the blanks.

$X \setminus Y$	1	2	3
1	1/4	1/16	
2	1/6	1/24	
3			

Since X and Y are independent, we have that $P(X = x, Y = y) = P(X = x)P(Y = y)$ for all x and y , so it suffices to determine the marginal distributions $P(X)$ and $P(Y)$.

We begin with $P(X)$. First observe that

$$\frac{P(X = 1, Y = 1)}{P(X = 2, Y = 1)} = \frac{P(X = 1)P(Y = 1)}{P(X = 2)P(Y = 1)} = \frac{P(X = 1)}{P(X = 2)} = \frac{1/4}{1/6} = \frac{3}{2}.$$

Combining this with the fact that any probability distribution sums to 1, we find that

$$P(X = 1) + P(X = 2) + P(X = 3) = \frac{3}{2} \cdot P(X = 2) + P(X = 2) + \frac{1}{6} = \frac{5}{2} \cdot P(X = 2) + \frac{1}{6} = 1,$$

which implies $P(X = 2) = 1/3$. It follows that $P(X = 1) = (3/2) \cdot P(X = 2) = 1/2$.

To recover the marginal distribution of Y , we note that

$$P(X = 1, Y = 1) = P(X = 1)P(Y = 1) = (1/2) \cdot P(Y = 1) = 1/4,$$

$$P(X = 1, Y = 2) = P(X = 1)P(Y = 2) = (1/2) \cdot P(Y = 2) = 1/16,$$

so $P(Y = 1) = 1/2$ and $P(Y = 2) = 1/8$. It follows that $P(Y = 3) = 1 - P(Y = 1) - P(Y = 2) = 3/8$.

Exercise 12.4.#XXXX

- a. Suppose $A \perp\!\!\!\perp B$. Determine the missing entries x and y of the joint distribution $P(A, B)$, where A and B take values in $\{0, 1\}$.

$$P(A = 0, B = 0) = 0.1$$

$$P(A = 0, B = 1) = 0.3$$

$$P(A = 1, B = 0) = x$$

$$P(A = 1, B = 1) = y$$

- b. Suppose $B \perp\!\!\!\perp C \mid A$. Determine the missing entries x, y, z of the joint distribution $P(A, B, C)$.

$$P(A = 0, B = 0, C = 0) = 0.01$$

$$P(A = 0, B = 0, C = 1) = 0.02$$

$$P(A = 0, B = 1, C = 0) = 0.03$$

$$P(A = 0, B = 1, C = 1) = x$$

$$P(A = 1, B = 0, C = 0) = 0.01$$

$$P(A = 1, B = 0, C = 1) = 0.1$$

$$P(A = 1, B = 1, C = 0) = y$$

$$P(A = 1, B = 1, C = 1) = z$$

- a. To solve this we use the two constraints: A and B are independent, and the distribution sums to 1. From independence, we have

$$y/x = \frac{P(A = 1, B = 1)}{P(A = 1, B = 0)} = \frac{P(A = 0, B = 1)}{P(A = 0, B = 0)} = \frac{P(B = 1)}{P(B = 0) = 3}.$$

So $y = 3x$. From sum-to-1 we have $x + y = 0.6$. Solving, we obtain $x = 0.15$ and $y = 0.45$.

- b. From conditional independence, as in (a), we obtain $x = 0.03 \cdot \frac{0.02}{0.01} = 0.06$ and $z/y = 10$. From sum-to-1, we have $0.01 + 0.02 + 0.03 + 0.06 + 0.01 + 0.1 + y + z = 1$ so $y + z = 0.77$. Solving, we get $y = 0.07, z = 0.7$.

Exercise 12.4.#XXXX

Deciding to put probability theory to good use, we encounter a slot machine with three

independent wheels, each producing one of the four symbols BAR, BELL, LEMON, or CHERRY with equal probability. The slot machine has the following payout scheme for a bet of 1 coin (where “?” denotes that we don’t care what comes up for that wheel):

BAR/BAR/BAR pays 20 coins
 BELL/BELL/BELL pays 15 coins
 LEMON/LEMON/LEMON pays 5 coins
 CHERRY/CHERRY/CHERRY pays 3 coins
 CHERRY/CHERRY/? pays 2 coins
 CHERRY/?/? pays 1 coin

- Compute the expected “payback” percentage of the machine. In other words, for each coin played, what is the expected coin return?
- Compute the probability that playing the slot machine once will result in a win.
- Estimate the mean and median number of plays you can expect to make until you go broke, if you start with 10 coins. You can run a simulation to estimate this, rather than trying to compute an exact answer.

- To compute the expected payback for the machine, we determine the probability for each winning outcome, multiply it by the amount that would be won in that instance, and sum all possible winning combinations. Since each symbol is equally likely, the first four cases have probability $(1/4)^3 = 1/64$.

However, in the case of computing winning probabilities for cherries, we must only consider the highest paying case, so we must subtract the probability for dominating winning cases from each subsequent case (e.g., in the case of two cherries, we subtract off the probability of getting three cherries):

$$\begin{aligned} \text{CHERRY/CHERRY/?} & \quad 3/64 = (1/4)^2 - 1/64 \\ \text{CHERRY/?/?} & \quad 12/64 = (1/4)^1 - 3/64 - 1/64 \end{aligned}$$

The expectation is therefore

$$20 \cdot 1/64 + 15 \cdot 1/64 + 5 \cdot 1/64 + 3 \cdot 1/64 + 2 \cdot 3/64 + 1 \cdot 12/64 = 61/64.$$

Thus, the expected payback percentage is $61/64$ (which is less than 1 as we would expect of a slot machine that was actually generating revenue for its owner).

- We can tally up the probabilities we computed in the previous section, to get

$$1/64 + 1/64 + 1/64 + 1/64 + 3/64 + 12/64 = 19/64.$$

Alternatively, we can observe that we win if either all symbols are the same (denote this event S), or if the first symbol is cherry (denote this event C). Then applying the inclusion-exclusion identity for disjunction:

$$P(S \vee C) = P(S) + P(C) - P(S \wedge C) = (1/4)^2 + 1/4 - 1/64 = 19/64.$$

- c. Using a simple Python simulation, we find a mean of about 210, and a median of 21. This shows the distribution of number of plays is heavy tailed: most of the time you run out of money relatively quickly, but occasionally you last for thousands of plays.

```

import random

def trial():
    funds = 10
    plays = 0
    while funds >= 1:
        funds -= 1
        plays += 1
        slots = [random.choice(
            ["bar", "bell", "lemon", "cherry"])
            for i in range(3)]
        if slots[0] == slots[1]:
            if slots[1] == slots[2]:
                num_equal = 3
            else:
                num_equal = 2
        else:
            num_equal = 1
        if slots[0] == "cherry":
            funds += num_equal
        elif num_equal == 3:
            if slots[0] == "bar":
                funds += 20
            elif slots[0] == "bell":
                funds += 15
            else:
                funds += 5
    return plays

def test(trials):
    results = [trial() for i in xrange(trials)]
    mean = sum(results) / float(trials)
    median = sorted(results)[trials/2]
    print "%s trials: mean=%s, median=%s" % (trials, mean, median)

test(10000)

```

Exercise 12.4.#XXXX

We wish to transmit an n -bit message to a receiving agent. The bits in the message are independently corrupted (flipped) during transmission with ϵ probability each. With an extra parity bit sent along with the original information, a message can be corrected by the receiver if at most one bit in the entire message (including the parity bit) has been corrupted. Suppose we want to ensure that the correct message is received with probability at least $1 - \delta$. What is the maximum feasible value of n ? Calculate this value for the case $\epsilon = 0.001$, $\delta = 0.01$.

The correct message is received if either zero or one of the $n + 1$ bits are corrupted. Since corruption occurs independently with probability ϵ , the probability that zero bits are corrupted is $(1 - \epsilon)^{n+1}$. There are $n + 1$ mutually exclusive ways that exactly one bit can be corrupted, one for each bit in the message. Each has probability $\epsilon(1 - \epsilon)^n$, so the overall probability that exactly one bit is corrupted is $n\epsilon(1 - \epsilon)^n$. Thus, the probability that the correct message is received is $(1 - \epsilon)^{n+1} + n\epsilon(1 - \epsilon)^n$.

The maximum feasible value of n , therefore, is the largest n satisfying the inequality

$$(1 - \epsilon)^{n+1} + n\epsilon(1 - \epsilon)^n \geq 1 - \delta.$$

Numerically solving this for $\epsilon = 0.001$, $\delta = 0.01$, we find $n = 147$.

Exercise 12.4.#INDI

Show that the three forms of independence in Equation (12.11) are equivalent.

Independence is symmetric (that is, a and b are independent iff b and a are independent) so $P(a|b) = P(a)$ is the same as $P(b|a) = P(b)$. So we need only prove that $P(a|b) = P(a)$ is equivalent to $P(a \wedge b) = P(a)P(b)$. The product rule, $P(a \wedge b) = P(a|b)P(b)$, can be used to rewrite $P(a \wedge b) = P(a)P(b)$ as $P(a|b)P(b) = P(a)P(b)$, which simplifies to $P(a|b) = P(a)$.

Exercise 12.4.#XXXX

Consider the following probability distributions:

A	$P(A)$
t	0.8
f	0.2

A	B	$P(B A)$
t	t	0.9
t	f	0.1
f	t	0.6
f	f	0.4

B	C	$P(C B)$
t	t	0.8
t	f	0.2
f	t	0.8
f	f	0.2

C	D	$P(D C)$
t	t	0.25
t	f	0.75
f	t	0.5
f	f	0.5

Given just these tables and no independence assumptions, calculate the following probabilities, showing your working. If it is impossible to calculate without more independence assumptions, specify instead a minimal set of independence assumptions that would allow you to answer the question.

- $P(a, \neg b)$.
- $P(b)$.
- $P(\neg a, \neg b, c)$.
- Now assume C is independent of A given B and D is independent of A and B given C. Calculate $P(a, \neg b, c, d)$.

a. $P(a, \neg b) = P(a)P(\neg b | a) = 0.8 \times 0.1 = 0.08$.

b.

$$\begin{aligned} P(b) &= \sum_a P(b|a)P(a) = P(b|a)P(a) + P(b|\neg a)P(\neg a) \\ &= (0.9 \times 0.8) + (0.6 \times 0.2) = 0.84 . \end{aligned}$$

- c. Not possible. Could do it given that C is independent of A given B.
d. This is a preview of some ideas in Chapter 13, specifically the chain rule. It might be a good idea to provide the chain rule as part of the question, but it can be done using material from this chapter by repeated application of the product rule and use of the given conditional independence assertions:

$$\begin{aligned} P(a, \neg b, c, d) &= P(\neg b, c, d | a)P(a) = P(c, d | a, \neg b)P(\neg b | a)P(a) \\ &= P(d | a, \neg b, c)P(c | a, \neg b)P(\neg b | a)P(a) \\ &= P(d | c)P(c | \neg b)P(\neg b | a)P(a) = 0.25 \times 0.8 \times 0.1 \times 0.8 = 0.016 . \end{aligned}$$

Exercise 12.4.#XXXX

For each of the following assertions, say whether it is true or false and support your answer with arguments or counterexamples where appropriate.

- a. $\mathbf{P}(A, B) = \mathbf{P}(A)\mathbf{P}(B)$.
- b. $\mathbf{P}(A | B) = \mathbf{P}(A)\mathbf{P}(B)$.
- c. $\mathbf{P}(A, B) = \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A | B)$.
- d. $\mathbf{P}(A, B, C) = \mathbf{P}(A | B, C)\mathbf{P}(B | C)\mathbf{P}(C)$.
- e. $\mathbf{P}(A, B, C) = \mathbf{P}(C | A, B)\mathbf{P}(A)\mathbf{P}(B)$.
- f. $\mathbf{P}(A, B, C) = \mathbf{P}(A | B)\mathbf{P}(B | C)\mathbf{P}(C)$.
- g. $\mathbf{P}(A, B, C) = \mathbf{P}(A | B, C)\mathbf{P}(B | A, C)\mathbf{P}(C | A, B)$.
- h. $\mathbf{P}(A, B, C) = \mathbf{P}(C, B | A)\mathbf{P}(A)$.
- i. $\mathbf{P}(A, B, C) = \mathbf{P}(C | A, B)\mathbf{P}(A, B)$.
- j. $\mathbf{P}(A, B) = \sum_c \mathbf{P}(A | B, C = c)\mathbf{P}(B | C = c)\mathbf{P}(C = c)$.
- k. If $\mathbf{P}(X, Y | Z) = \mathbf{P}(X | Z)\mathbf{P}(Y | Z)$ then X is independent of Y given Z.
- l. If $\mathbf{P}(X, Y, Z) = \mathbf{P}(X, Z)\mathbf{P}(Y)$ then X is independent of Y given Z.
- m. If $\mathbf{P}(X, Y, Z) = \mathbf{P}(X)\mathbf{P}(Z)\mathbf{P}(Y)$ then X is independent of Y given Z.

- a. False, $\mathbf{P}(A, B) = \mathbf{P}(A)\mathbf{P}(B)$ iff. $A \perp\!\!\!\perp B$.
- b. False, $\mathbf{P}(A | B) = \frac{\mathbf{P}(A, B)}{\mathbf{P}(B)}$.
- c. False, eg. if $\mathbf{P}(A) \perp\!\!\!\perp \mathbf{P}(B)$ then $\mathbf{P}(A, B) \neq \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A | B) = \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A)$.
- d. True by application of the chain rule .
- e. False, $\mathbf{P}(A, B, C) = \mathbf{P}(C | A, B)\mathbf{P}(A | B)\mathbf{P}(B)$.
- f. False, $\mathbf{P}(A, B, C) = \mathbf{P}(A | B, C)\mathbf{P}(B | C)\mathbf{P}(C)$.
- g. False, $\mathbf{P}(A | B, C)\mathbf{P}(B | A, C)\mathbf{P}(C | A, B) = \frac{3\mathbf{P}(A, B, C)}{\mathbf{P}(A, B)\mathbf{P}(A, C)\mathbf{P}(B, C)}$.

- h. True by application of the product rule where $A' = B, C$ and $B' = A$.
- i. True by application of the product rule where $A' = C$ and $B' = A, C$.
- j. True by definition of a probability distribution, $\mathbf{P}(C)$, over all assignments—the marginalization of a conditioned variable.
- k. True, $X \perp\!\!\!\perp Y | Z$ by the definition of conditional independence.
- l. False, for $X \perp\!\!\!\perp Y | Z$, $\mathbf{P}(X, Y, Z) = \mathbf{P}(X | Z) \mathbf{P}(Y | Z) \mathbf{P}(Z)$.
- m. False, for $X \perp\!\!\!\perp Y | Z$, $\mathbf{P}(X, Y, Z) = \mathbf{P}(X | Z) \mathbf{P}(Y | Z) \mathbf{P}(Z)$.

Exercise 12.4.#XXXX

- a. Assuming that A is independent of B given C and that A and C are absolutely independent, what is the most factored representation of $\mathbf{P}(A, B, C)$?
- b. Assuming that A is independent of B given C , what is the most factored representation of $\mathbf{P}(A, B | C)$?
- c. Given no independence assumptions, what is the most factored representation of $\mathbf{P}(A | B, C)$?
- d. Assuming that A is independent of B given C , what is the most factored representation of $\mathbf{P}(A | B, C)$?
- e. Assuming that A is absolutely independent of B , what is the most factored representation of $\mathbf{P}(A | B, C)$?
- f. Assuming that A is independent of B given C , write an expression equivalent to $\mathbf{P}(A | B)$ using A, B , and C .
- g. Which of the following expressions are equal to 1, given no independence assumptions?
 - (i) $\sum_a \mathbf{P}(A = a | B)$.
 - (ii) $\sum_b \mathbf{P}(A | B = b)$.
 - (iii) $\sum_a \sum_b P(A = a, B = b)$.
 - (iv) $\sum_a \sum_b P(A = a | B = b)$.
 - (v) $\sum_a \sum_b P(A = a) P(B = b)$.
- h. Which of the following expressions hold for any distribution over four random variables A, B, C and D ?
 - (i) $\mathbf{P}(A, B | C, D) = \mathbf{P}(A | C, D) \mathbf{P}(B | A, C, D)$.
 - (ii) $\mathbf{P}(A, B) = \mathbf{P}(A, B | C, D) \mathbf{P}(C, D)$.
 - (iii) $\mathbf{P}(A, B | C, D) = \mathbf{P}(A, B) \mathbf{P}(C, D) \mathbf{P}(C, D | A, B)$.
 - (iv) $\mathbf{P}(A, B | C, D) = \mathbf{P}(A, B) \mathbf{P}(D) \mathbf{P}(C, D | A, B)$.

- a. $\mathbf{P}(A, B | C) \mathbf{P}(C) = \mathbf{P}(A | C) \mathbf{P}(B | C) \mathbf{P}(C) = \mathbf{P}(A) \mathbf{P}(B | C) \mathbf{P}(C)$.
- b. $\mathbf{P}(A | B, C) = \mathbf{P}(A | C) \mathbf{P}(B | C)$.
- c. The same or $\frac{\mathbf{P}(B, C | A) \mathbf{P}(A)}{\mathbf{P}(B, C)}$.
- d. $\mathbf{P}(A | B, C) = \frac{\mathbf{P}(A, B, C)}{\mathbf{P}(B, C)} = \frac{\mathbf{P}(A, B | C) \mathbf{P}(C)}{\mathbf{P}(B | C) \mathbf{P}(C)} = \frac{\mathbf{P}(A | C) \mathbf{P}(B | C) \mathbf{P}(C)}{\mathbf{P}(B | C) \mathbf{P}(C)} = \mathbf{P}(A | C)$.
- e. The same, or: $\mathbf{P}(A | B, C) = \frac{\mathbf{P}(A, B, C)}{\mathbf{P}(B, C)} = \frac{\mathbf{P}(C | A, B) \mathbf{P}(A) \mathbf{P}(B)}{\mathbf{P}(B, C)}$.

f. $\mathbf{P}(A | B) = \frac{\mathbf{P}(A, B)}{\mathbf{P}(B)} = \frac{\sum_c P(A, B | C=c) P(C=c)}{\mathbf{P}(B)} = \frac{\sum_c P(A | C=c) P(B | C=c) P(C=c)}{\mathbf{P}(B)}.$

g. Most steps follow the rules for marginalization or conditioning.

(i) $\sum_a \mathbf{P}(A = a | B) = 1.$

(ii) $\sum_b \mathbf{P}(A | B = b) = \alpha \mathbf{P}(A).$

(iii) $\sum_a \sum_b P(A = a, B = b) = \sum_a P(A = a) = 1.$

(iv) $\sum_a \sum_b P(A = a | B = b) = \sum_a \alpha P(A = a) = \alpha.$

(v) $\sum_a \sum_b P(A = a) P(B = b) = \sum_a P(A = a) \sum_b P(B = b) = \sum_a P(A = a) = 1.$

h. (i) True by the product rule; $\mathbf{P}(A, B | C, D) = \mathbf{P}(B | A, C, D) \mathbf{P}(A | C, D).$

(ii) False; $\mathbf{P}(A, B) = \sum_c \sum_d P(A, B | C = c, D = d) P(C = c, D = d).$

(iii) False; $\mathbf{P}(A, B | C, D) = \mathbf{P}(A, B) \frac{1}{\mathbf{P}(C, D)} \mathbf{P}(C, D | A, B).$

(iv) False; see above.

Exercise 12.4.#XXXX

For each of the following expression, derive an equivalent expression in terms of *only* the given conditional distributions, using the given conditional independence assumptions. If it is not possible, say so.

a. Express $\mathbf{P}(A, B | C)$ in terms of $\mathbf{P}(A), \mathbf{P}(A | C), \mathbf{P}(B | C), \mathbf{P}(C | A, B)$ given no conditional independence assumptions.

b. Express $\mathbf{P}(B | A, C)$ in terms of $\mathbf{P}(A), \mathbf{P}(A | C), \mathbf{P}(B | A), \mathbf{P}(C | A, B)$ and no conditional independence assumptions.

c. Express $\mathbf{P}(C)$ in terms of $\mathbf{P}(A | B), \mathbf{P}(B), \mathbf{P}(B | A, C), \mathbf{P}(C | A)$ given that A and B are absolutely independent.

d. Express $\mathbf{P}(A, B, C)$ in terms of $\mathbf{P}(A | B, C), \mathbf{P}(B), \mathbf{P}(B | A, C), \mathbf{P}(C | B, A)$ given that A is conditionally independent of B given C .

a. Not possible.

b. $\frac{\mathbf{P}(A) \mathbf{P}(B | A) \mathbf{P}(C | A, B)}{\sum_b \mathbf{P}(A) \mathbf{P}(B | A) \mathbf{P}(C | A, B)}$

c. $\sum_a \mathbf{P}(A | B) \mathbf{P}(C | A)$

d. Not possible.

Exercise 12.4.#XXXX

a. Consider the following two assertions, where $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{X}, \mathbf{Y}$, and \mathbf{Z} are sets of random variables:

(i) \mathbf{U} is independent of \mathbf{V} given \mathbf{W} .

(ii) \mathbf{X} is independent of \mathbf{Y} given \mathbf{Z} .

Under what conditions, in terms of subset/superset relationships among $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{X}, \mathbf{Y}$, and \mathbf{Z} , can one say that (i) entails (ii)?

b. We will say that a conditional independence assertion C_1 is **strictly weaker** than an assertion C_2 if C_2 entails C_1 and C_1 does not entail C_2 . For each of the following equations, give the weakest conditional independence assertion required to make it true (if any).

- (i) $\mathbf{P}(A, C) = \mathbf{P}(A | B) \mathbf{P}(C)$.
- (ii) $\mathbf{P}(A | B, C) = \frac{\mathbf{P}(A) \mathbf{P}(B | A) \mathbf{P}(C | A)}{\mathbf{P}(B | C) \mathbf{P}(C)}$
- (iii) $\mathbf{P}(A, B) = \sum_c \mathbf{P}(A | B, c) \mathbf{P}(B | c) \mathbf{P}(c)$
- (iv) $\mathbf{P}(A, B | C, D) = \mathbf{P}(A | C, D) \mathbf{P}(B | A, C, D)$
- (v) $\mathbf{P}(A, B, C, D) = \mathbf{P}(A | B, C) \mathbf{P}(D | B, C) \mathbf{P}(B) \mathbf{P}(C | B)$
- (vi) $\mathbf{P}(A, B, C, D) = \mathbf{P}(B) \mathbf{P}(C) \mathbf{P}(A | B, C) \mathbf{P}(D | A, B, C)$

a. $\mathbf{X} \subseteq \mathbf{U}$, $\mathbf{Y} \subseteq \mathbf{V}$, and $\mathbf{Z} \supseteq \mathbf{W}$.

- b.

 - (i) $A \perp\!\!\!\perp C$ and $A \perp\!\!\!\perp B$.
 - (ii) $B \perp\!\!\!\perp C | A$.
 - (iii) None required by conditioning and the product rule.
 - (iv) None required by the product rule.
 - (v) $A \perp\!\!\!\perp D | B, C$.
 - (vi) $B \perp\!\!\!\perp C$.

Exercise 12.4.#XXXX

Simplify each of the following expressions down a single probability term, without making any independence assumptions:

- a. $\sum_{a'} \mathbf{P}(a' | D) \mathbf{P}(b | a', D)$.
- b. $\sum_{b', c', d'} \mathbf{P}(A) \mathbf{P}(b' | A) \mathbf{P}(c' | A, b') \mathbf{P}(d' | A, b', c')$.
- c. $\frac{\sum_{b'} \mathbf{P}(A | b', D) \mathbf{P}(b' | D) \mathbf{P}(D)}{\sum_{a', b'} \mathbf{P}(a' | b', D) \mathbf{P}(b' | D) \mathbf{P}(D)}$.

a. $\mathbf{P}(b | D)$

b. $\mathbf{P}(A)$

c. $\mathbf{P}(A | D)$

12.5 Bayes' Rule and Its Use

Exercise 12.5.#XXXX

Formulate each of the following as probability models, stating all your assumptions, and use the probability model to answer the questions.

- a. Aliens can be friendly or not; 75% are friendly. Friendly aliens arrive during the day 90% of the time, while unfriendly ones always arrive at night. If an alien arrives at night, how likely is it to be friendly?
- b. Half of all monsters live in attics, while the rest live in basements. While 80 % of all monsters are fuzzy, *all* attic-living monsters are fuzzy. What is the probability that a basement-living monsters is fuzzy?

- a. Let $\langle f, \neg f \rangle$ represent whether or not an alien is friendly and $\langle d, \neg d \rangle$ whether an alien arrives during the day. Then, $P(f) = .75$, $P(\neg f) = .25$, $P(d | f) = .9$, $P(\neg d | f) = .1$, $P(d | \neg f) = 0$, and $P(\neg d | \neg f) = 1$. Thus,

$$\begin{aligned} P(f | \neg d) &= \frac{P(\neg d | f) P(f)}{P(\neg d)} \\ &= \frac{P(\neg d | f) P(f)}{P(\neg d | f) P(f) + P(\neg d | \neg f) P(\neg f)} \\ &= \frac{.1 \times .75}{.1 \times .75 + 1 \times .25} \\ &\approx .23. \end{aligned}$$

- b. Let $\langle a, b \rangle$ represent whether a monster lives in the attic or basement and $\langle f, \neg f \rangle$ whether it is fuzzy. Then we have $P(a) = .5$, $P(b) = .5$, $P(f) = .8$, and $P(f | a) = 1$, $P(\neg f | a) = 0$. Thus,

$$\begin{aligned} P(f) &= P(f | b) P(b) + P(f | a) P(a) \\ P(f | b) &= \frac{P(f) - P(f | a) P(a)}{P(b)} \\ &= \frac{.8 - 1 \times .5}{.5} \\ &= .6. \end{aligned}$$

Exercise 12.5.#XXXX

Let A and B be Boolean random variables. You are given the following quantities:

$$\begin{aligned} P(A = Jtrue) &= \frac{1}{2} \\ P(B = true | A = true) &= 1 \\ P(B = true) &= \frac{3}{4} \end{aligned}$$

What is $P(B = true | A = false)$?

The simplest way to solve this is to realize that

$$P(B = true) = P(B = true | A = true)P(A = true) + P(B = true | A = false)P(A = false).$$

Using this fact, you can solve for $\mathbf{P}(B = \text{true} | A = \text{false})$:

$$(1) \left(\frac{1}{2} \right) + \mathbf{P}(B = \text{true} | A = \text{false}) \left(\frac{1}{2} \right) = \frac{3}{4}$$

$$\Rightarrow \mathbf{P}(B = \text{true} | A = \text{false}) = \frac{1}{2}$$

Exercise 12.5.#MDAB

Consider two medical tests, A and B, for a virus. Test A is 95% effective at recognizing the virus when it is present, but has a 10% false positive rate (indicating that the virus is present, when it is not). Test B is 90% effective at recognizing the virus, but has a 5% false positive rate. The two tests use independent methods of identifying the virus. The virus is carried by 1% of all people. Say that a person is tested for the virus using only one of the tests, and that test comes back positive for carrying the virus. Which test returning positive is more indicative of someone really carrying the virus? Justify your answer mathematically.

Let V be the statement that the patient has the virus, and A and B the statements that the medical tests A and B returned positive, respectively. The problem statement gives:

$$\begin{aligned} P(V) &= 0.01 \\ P(A|V) &= 0.95 \\ P(A|\neg V) &= 0.10 \\ P(B|V) &= 0.90 \\ P(B|\neg V) &= 0.05 \end{aligned}$$

The test whose positive result is more indicative of the virus being present is the one whose posterior probability, $P(V|A)$ or $P(V|B)$ is largest. One can compute these probabilities directly from the information given, finding that $P(V|A) = 0.0876$ and $P(V|B) = 0.1538$, so B is more indicative.

Equivalently, the question is asking which test has the highest posterior odds ratio $P(V|A)/P(\neg V|A)$. From the odd form of Bayes theorem:

$$\frac{P(V|A)}{P(\neg V|A)} = \frac{P(A|V)}{P(A|\neg V)} \frac{P(V)}{P(\neg V)}$$

we see that the ordering is independent of the probability of V , and that we just need to compare the likelihood ratios $P(A|V)/P(A|\neg V) = 9.5$ and $P(B|V)/P(B|\neg V) = 18$ to find the answer.

Exercise 12.5.#HEDP

Suppose you are given a coin that lands *heads* with probability x and *tails* with probability $1 - x$. Are the outcomes of successive flips of the coin independent of each other given

that you know the value of x ? Are the outcomes of successive flips of the coin independent of each other if you do *not* know the value of x ? Justify your answer.

If the probability x is known, then successive flips of the coin are independent of each other, since we know that each flip of the coin will land *heads* with probability x . Formally, if F_1 and F_2 represent the results of two successive flips, we have

$$P(F_1 = \text{heads}, F_2 = \text{heads}|x) = x * x = P(F_1 = \text{heads}|x)P(F_2 = \text{heads}|x)$$

Thus, the events $F_1 = \text{heads}$ and $F_2 = \text{heads}$ are independent.

If we do not know the value of x , however, the probability of each successive flip is dependent on the result of all previous flips. The reason for this is that each successive flip gives us information to better estimate the probability x (i.e., determining the posterior estimate for x given our prior probability and the evidence we see in the most recent coin flip). This new estimate of x would then be used as our “best guess” of the probability of the coin coming up *heads* on the next flip. Since this estimate for x is based on all the previous flips we have seen, the probability of the next flip coming up *heads* depends on how many *heads* we saw in all previous flips, making them dependent.

For example, if we had a uniform prior over the probability x , then one can show that after n flips if m of them come up heads then the probability that the next one comes up heads is $(m + 1)/(n + 2)$, showing dependence on previous flips.

Exercise 12.5.#XXXX

After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease and that the test is 99% accurate (i.e., the probability of testing positive when you do have the disease is 0.99, as is the probability of testing negative when you don't have the disease). The good news is that this is a rare disease, striking only 1 in 10,000 people of your age. Why is it good news that the disease is rare? What are the chances that you actually have the disease?

We are given the following information:

$$P(\text{test}|\text{disease}) = 0.99$$

$$P(\neg\text{test}|\neg\text{disease}) = 0.99$$

$$P(\text{disease}) = 0.0001$$

and the observation *test*. What the patient is concerned about is $P(\text{disease}|\text{test})$. Roughly speaking, the reason it is a good thing that the disease is rare is that $P(\text{disease}|\text{test})$ is proportional to $P(\text{disease})$, so a lower prior for *disease* will mean a lower value for $P(\text{disease}|\text{test})$. Roughly speaking, if 10,000 people take the test, we expect 1 to actually have the disease, and most likely test positive, while the rest do not have the disease, but 1% of them (about 100 people) will test positive anyway, so $P(\text{disease}|\text{test})$ will be about 1 in 100. More precisely,

using the normalization equation from page 480:

$$\begin{aligned}
 P(\text{disease} | \text{test}) &= \frac{P(\text{test} | \text{disease})P(\text{disease})}{P(\text{test} | \text{disease})P(\text{disease}) + P(\text{test} | \neg \text{disease})P(\neg \text{disease})} \\
 &= \frac{0.99 \times 0.0001}{0.99 \times 0.0001 + 0.01 \times 0.9999} \\
 &= .009804
 \end{aligned}$$

The moral is that when the disease is much rarer than the test accuracy, a positive test result does not mean the disease is likely. A false positive reading remains much more likely.

Here is an alternative exercise along the same lines: A doctor says that an infant who predominantly turns the head to the right while lying on the back will be right-handed, and one who turns to the left will be left-handed. Isabella predominantly turned her head to the left. Given that 90% of the population is right-handed, what is Isabella's probability of being right-handed if the test is 90% accurate? If it is 80% accurate?

The reasoning is the same, and the answer is 50% right-handed if the test is 90% accurate, 69% right-handed if the test is 80% accurate.

Exercise 12.5.#CONB

It is quite often useful to consider the effect of some specific propositions in the context of some general background evidence that remains fixed, rather than in the complete absence of information. The following questions ask you to prove more general versions of the product rule and Bayes' rule, with respect to some background evidence \mathbf{e} :

- a. Prove the conditionalized version of the general product rule:

$$\mathbf{P}(X, Y | \mathbf{e}) = \mathbf{P}(X | Y, \mathbf{e})\mathbf{P}(Y | \mathbf{e}) .$$

- b. Prove the conditionalized version of Bayes' rule in Equation (12.13).

The basic axiom to use here is the definition of conditional probability:

- a. We have

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

and

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

hence

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

- b. The derivation here is the same as the derivation of the simple version of Bayes' Rule on page 426. First we write down the dual form of the conditionalized product rule,

simply by switching A and B in the above derivation:

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

Therefore the two right-hand sides are equal:

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

Dividing through by $\mathbf{P}(B|E)$ we get

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

Exercise 12.5.#PXYZ

Show that the statement of conditional independence

$$\mathbf{P}(X, Y | Z) = \mathbf{P}(X | Z)\mathbf{P}(Y | Z)$$

is equivalent to each of the statements

$$\mathbf{P}(X | Y, Z) = \mathbf{P}(X | Z) \quad \text{and} \quad \mathbf{P}(B | X, Z) = \mathbf{P}(Y | Z).$$

The key to this exercise is rigorous and frequent application of the definition of conditional probability, $\mathbf{P}(X|Y) = \mathbf{P}(X, Y)/\mathbf{P}(Y)$. The original statement that we are given is:

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

We start by applying the definition of conditional probability to two of the terms in this statement:

$$\mathbf{P}(A, B|C) = \frac{\mathbf{P}(A, B, C)}{\mathbf{P}(C)} \quad \text{and} \quad \mathbf{P}(B|C) = \frac{\mathbf{P}(B, C)}{\mathbf{P}(C)}$$

Now we substitute the right hand side of these definitions for the left hand sides in the original statement to get:

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

Now we need the definition once more:

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

We substitute this right hand side for $\mathbf{P}(A, B, C)$ to get:

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

Finally, we cancel the $\mathbf{P}(B, C)$ and $\mathbf{P}(C)$ s to get:

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

The second part of the exercise follows from by a similar derivation, or by noticing that A and B are interchangeable in the original statement (because multiplication is commutative and A, B means the same as B, A).

In Chapter 13, we will see that in terms of Bayesian networks, the original statement means that C is the lone parent of A and also the lone parent of B . The conclusion is that knowing the values of B and C is the same as knowing just the value of C in terms of telling you something about the value of A .

Exercise 12.5.#XXXX

You have three coins in your pocket:

- Coin 1 is a fair coin that comes up heads with probability $1/2$.
- Coin 2 is a biased coin that comes up heads with probability $1/4$.
- Coin 3 is a biased coin that comes up heads with probability $3/4$.

Suppose you pick one of the coins uniformly at random and flip it three times. If you observe the sequence HHT (where H stands for heads and T stands for tails), what is the probability that you chose Coin 3?

Let C_i denote the event that coin i was chosen. Using Bayes' rule, we have that

$$P(C_3 | HHT) = \frac{P(HHT | C_3)P(C_3)}{P(HHT)} = \frac{P(HHT | C_3)P(C_3)}{\sum_{i=1}^3 P(HHT | C_i)P(C_i)} = \frac{P(HHT | C_3)}{\sum_{i=1}^3 P(HHT | C_i)},$$

where the final equality follows from the fact that $P(C_i) = 1/3$ for each i . Substituting in the values from the problem, we obtain

$$P(C_3 | HHT) = \frac{\frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4}}{\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} + \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4}} = \frac{\frac{9}{64}}{\frac{1}{8} + \frac{3}{64} + \frac{9}{64}} = \frac{9}{20}.$$

Exercise 12.5.#XXXX

Suppose you are given a bag containing n unbiased coins. You are told that $n - 1$ of these coins are normal, with heads on one side and tails on the other, whereas one coin is a fake, with heads on both sides.

- Suppose you reach into the bag, pick out a coin at random, flip it, and get a head. What is the (conditional) probability that the coin you chose is the fake coin?
- Suppose you continue flipping the coin for a total of k times after picking it and see k heads. Now what is the conditional probability that you picked the fake coin?
- Suppose you wanted to decide whether the chosen coin was fake by flipping it k times. The decision procedure returns *fake* if all k flips come up heads; otherwise it returns *normal*. What is the (unconditional) probability that this procedure makes an error?

- a. A typical “counting” argument goes like this: There are n ways to pick a coin, and 2 outcomes for each flip (although with the fake coin, the results of the flip are indistinguishable), so there are $2n$ total atomic events, each equally likely. Of those, only 2 pick the fake coin, and $2 + (n - 1)$ result in heads. So the probability of a fake coin given heads, $P(\text{fake}|\text{heads})$, is $2/(2 + n - 1) = 2/(n + 1)$.

Often such counting arguments go astray when the situation gets complex. It may be better to do it more formally:

$$\begin{aligned}\mathbf{P}(\text{Fake}|\text{heads}) &= \alpha \mathbf{P}(\text{heads}|\text{Fake}) \mathbf{P}(\text{Fake}) \\ &= \alpha \langle 1.0, 0.5 \rangle \langle 1/n, (n-1)/n \rangle \\ &= \alpha \langle 1/n, (n-1)/2n \rangle \\ &= \langle 2/(n+1), (n-1)/(n+1) \rangle\end{aligned}$$

- b. Now there are $2^k n$ atomic events, of which 2^k pick the fake coin, and $2^k + (n - 1)$ result in heads. So the probability of a fake coin given a run of k heads, $P(\text{fake}|\text{heads}^k)$, is $2^k/(2^k + (n - 1))$. Note this approaches 1 as k increases, as expected. If $k = n = 12$, for example, than $P(\text{fake}|\text{heads}^{10}) = 0.9973$.

Doing it the formal way:

$$\begin{aligned}\mathbf{P}(\text{Fake}|\text{heads}^k) &= \alpha \mathbf{P}(\text{heads}^k|\text{Fake}) \mathbf{P}(\text{Fake}) \\ &= \alpha \langle 1.0, 0.5^k \rangle \langle 1/n, (n-1)/n \rangle \\ &= \alpha \langle 1/n, (n-1)/2^k n \rangle \\ &= \langle 2^k/(2^k + n - 1), (n-1)/(2^k + n - 1) \rangle\end{aligned}$$

- c. The procedure makes an error if and only if a fair coin is chosen and turns up heads k times in a row. The probability of this

$$P(\text{heads}^k | \neg \text{fake}) P(\neg \text{fake}) = (n-1)/2^k n .$$

Exercise 12.5.#NOEM

In this exercise, you will complete the normalization calculation for the meningitis example. First, make up a suitable value for $P(s | \neg m)$, and use it to calculate unnormalized values for $P(m | s)$ and $P(\neg m | s)$ (i.e., ignoring the $P(s)$ term in the Bayes' rule expression, Equation (12.14)). Now normalize these values so that they add to 1.

The important point here is that although there are often many possible routes by which answers can be calculated in such problems, it is usually better to stick to systematic “standard” routes such as Bayes’ Rule plus normalization. Chapter 14 describes general-purpose, systematic algorithms that make heavy use of normalization. We could guess that $P(S|\neg M) \approx 0.05$, or we could calculate it from the information already given (although the idea here is to assume that $P(S)$ is *not* known):

$$P(S|\neg M) = \frac{P(\neg M|S)P(S)}{P(\neg M)} = \frac{(1 - P(M|S))P(S)}{1 - P(\neg M)} = \frac{0.9998 \times 0.05}{0.99998} = 0.049991$$

Normalization proceeds as follows:

$$\begin{aligned} P(M|S) &\propto P(S|M)P(M) = 0.5/50,000 = 0.00001 \\ P(\neg M|S) &\propto P(S|\neg M)P(\neg M) = 0.049991 \times 0.99998 = 0.04999 \\ P(M|S) &= \frac{0.00001}{0.00001+0.04999} = 0.0002 \\ P(\neg M|S) &= \frac{0.00001}{0.00001+0.04999} = 0.9998 \end{aligned}$$

Exercise 12.5.#XXXX

This exercise investigates the way in which conditional independence relationships affect the amount of information needed for probabilistic calculations.

- a. Suppose we wish to calculate $P(h | e_1, e_2)$ and we have no conditional independence information. Which of the following sets of numbers are sufficient for the calculation?
 - (i) $\mathbf{P}(E_1, E_2), \mathbf{P}(H), \mathbf{P}(E_1 | H), \mathbf{P}(E_2 | H)$
 - (ii) $\mathbf{P}(E_1, E_2), \mathbf{P}(H), \mathbf{P}(E_1, E_2 | H)$
 - (iii) $\mathbf{P}(H), \mathbf{P}(E_1 | H), \mathbf{P}(E_2 | H)$
- b. Suppose we know that $\mathbf{P}(E_1 | H, E_2) = \mathbf{P}(E_1 | H)$ for all values of H, E_1, E_2 . Now which of the three sets are sufficient?

The question would have been slightly more consistent if we had asked about the calculation of $\mathbf{P}(H|E_1, E_2)$ instead of $P(H|E_1, E_2)$. Showing that a given set of information is *sufficient* is relatively easy: find an expression for $\mathbf{P}(H|E_1, E_2)$ in terms of the given information. Showing *insufficiency* can be done by showing that the information provided does not contain enough independent numbers.

- a. Bayes’ Rule gives

$$\mathbf{P}(H|E_1, E_2) = \frac{\mathbf{P}(E_1, E_2 | H)\mathbf{P}(H)}{\mathbf{P}(E_1, E_2)}$$

Hence the information in (ii) is sufficient—in fact, we don’t need $\mathbf{P}(E_1, E_2)$ because we can use normalization. Intuitively, the information in (iii) is insufficient because $\mathbf{P}(E_1 | H)$ and $\mathbf{P}(E_2 | H)$ provide no information about correlations between E_1 and E_2 that might be induced by H . Mathematically, suppose H has m possible values and E_1 and E_2 have n_1 and n_2 possible respectively. $\mathbf{P}(H|E_1, E_2)$ contains $(m - 1)n_1n_2$

independent numbers, whereas the information in (iii) contains $(m - 1) + m(n_1 - 1) + m(n_2 - 1)$ numbers—clearly insufficient for large m , n_1 , and n_2 . Similarly, the information in (i) contains $(n_1 n_2 - 1) + m + m(n_1 - 1) + m(n_2 - 1)$ numbers—again insufficient.

- b. If E_1 and E_2 are conditionally independent given H , then

$$\mathbf{P}(E_1, E_2 | H) = \mathbf{P}(E_1 | H)\mathbf{P}(E_2 | H).$$

Using normalization, (i), (ii), and (iii) are each sufficient for the calculation.

Exercise 12.5.#BLRV

Let X, Y, Z be Boolean random variables. Label the eight entries in the joint distribution $\mathbf{P}(X, Y, Z)$ as a through h . Express the statement that X and Y are conditionally independent given Z , as a set of equations relating a through h . How many *nonredundant* equations are there?

Let the probabilities be as follows:

x	y	z	$P(x, y, z)$
F	F	F	a
F	F	T	b
F	T	F	c
F	T	T	d
T	F	F	e
T	F	T	f
T	T	F	g
T	T	T	h

Conditional independence asserts that

$$\mathbf{P}(X, Y | Z) = \mathbf{P}(X | Z)\mathbf{P}(Y | Z)$$

which we can rewrite in terms of the joint distribution using the definition of conditional probability and marginals:

$$\begin{aligned} \frac{\mathbf{P}(X, Y, Z)}{\mathbf{P}(Z)} &= \frac{\mathbf{P}(X, Z)}{\mathbf{P}(Z)} \cdot \frac{\mathbf{P}(Y, Z)}{\mathbf{P}(Z)} \\ \mathbf{P}(X, Y, Z) &= \frac{\mathbf{P}(X, Z)\mathbf{P}(Y, Z)}{\mathbf{P}(Z)} = \frac{\left(\sum_y \mathbf{P}(X, y, Z)\right) \left(\sum_x \mathbf{P}(x, Y, Z)\right)}{\sum_{x,y} \mathbf{P}(x, y, Z)}. \end{aligned}$$

Exercises 12 Quantifying Uncertainty

Now we instantiate X, Y, Z in all 8 ways to obtain the following 8 equations:

$$\begin{aligned} a &= (a+c)(a+e)/(a+c+e+g) \text{ or } ag = ce \\ b &= (b+d)(b+f)/(b+d+f+h) \text{ or } bh = df \\ c &= (a+c)(c+g)/(a+c+e+g) \text{ or } ce = ag \\ d &= (b+d)(d+h)/(b+d+f+h) \text{ or } df = bh \\ e &= (e+g)(a+e)/(a+c+e+g) \text{ or } ce = ag \\ f &= (f+h)(b+f)/(b+d+f+h) \text{ or } df = bh \\ g &= (e+g)(c+g)/(a+c+e+g) \text{ or } ag = ce \\ h &= (f+h)(d+h)/(b+d+f+h) \text{ or } bh = df . \end{aligned}$$

Thus, there are only 2 nonredundant equations, $ag = ce$ and $bh = df$. This is what we would expect: the general distribution requires $8 - 1 = 7$ parameters, whereas the Bayes net with Z as root and X and Y as conditionally independent children requires 1 parameter for Z and 2 each for X and Y , or 5 in all. Hence the conditional independence assertion removes two degrees of freedom.

Exercise 12.5.#XXXX

Suppose you are a witness to a nighttime hit-and-run accident involving a taxi in Athens. All taxis in Athens are blue or green. You swear, under oath, that the taxi was blue. Extensive testing shows that, under the dim lighting conditions, discrimination between blue and green is 75% reliable. (Adapted from Pearl (1988).)

- a. Is it possible to calculate the most likely color for the taxi? (*Hint:* distinguish carefully between the proposition that the taxi *is* blue and the proposition that it *appears* blue.)
- b. What if you know that 9 out of 10 Athenian taxis are green?

The relevant aspect of the world can be described by two random variables: B means the taxi *was* blue, and LB means the taxi *looked* blue. The information on the reliability of color identification can be written as

$$P(LB|B) = 0.75 \quad P(\neg LB|\neg B) = 0.75$$

We need to know the probability that the taxi was blue, given that it looked blue:

$$\begin{aligned} P(B|LB) &\propto P(LB|B)P(B) \propto 0.75P(B) \\ P(\neg B|LB) &\propto P(LB|\neg B)P(\neg B) \propto 0.25(1 - P(B)) \end{aligned}$$

Thus we cannot decide the probability without some information about the prior probability of blue taxis, $P(B)$. For example, if we knew that all taxis were blue, i.e., $P(B) = 1$, then obviously $P(B|LB) = 1$. On the other hand, if we adopt Laplace's *Principle of Indifference*, which states that propositions can be deemed equally likely in the absence of any differentiating information, then we have $P(B) = 0.5$ and $P(B|LB) = 0.75$. Usually we will have *some* differentiating information, so this principle does not apply.

Given that 9 out of 10 taxis are green, and *assuming the taxi in question is drawn ran-*

domly from the taxi population, we have $P(B) = 0.1$. Hence

$$\begin{aligned}P(B|LB) &\propto 0.75 \times 0.1 \propto 0.075 \\P(\neg B|LB) &\propto 0.25 \times 0.9 \propto 0.225 \\P(B|LB) &= \frac{0.075}{0.075+0.225} = 0.25 \\P(\neg B|LB) &= \frac{0.225}{0.075+0.225} = 0.75\end{aligned}$$

12.6 Naive Bayes Models

Exercise 12.6.#XXXX

Consider a model with one binary class variable Y and n k -ary features F_1, F_2, \dots, F_n .

- a. How many parameters are required to represent the full joint distribution $\mathbf{P}(Y, F_1, F_2, \dots, F_n)$ in tabular form, given no conditional independence properties?
 - b. How many are needed when the naive Bayes model is applicable?
-
- a. The table size is $|Y| \cdot |F_i|^n = 2k^n$, minus 1 for the sum-to-1 constraint, so $2k^n - 1$ parameters are required.
 - b. In the naive Bayes model, we need 1 for the prior on Y and, for each F_i , $k-1$ parameters for each of the 2 values of Y , so $2n(k-1) + 1$ parameters are needed.

Exercise 12.6.#XXXX

Write out a general algorithm for answering queries of the form $\mathbf{P}(Cause | \mathbf{e})$, using a naive Bayes distribution. Assume that the evidence \mathbf{e} may assign values to *any subset* of the effect variables.

We can apply the definition of conditional independence as follows:

$$\mathbf{P}(Cause | \mathbf{e}) = \mathbf{P}(\mathbf{e}, Cause) / \mathbf{P}(\mathbf{e}) = \alpha \mathbf{P}(\mathbf{e}, Cause).$$

Now, divide the effect variables into those with evidence, \mathbf{E} , and those without evidence, \mathbf{Y} .

We have

$$\begin{aligned}
 \mathbf{P}(Cause|\mathbf{e}) &= \alpha \sum_{\mathbf{y}} \mathbf{P}(\mathbf{y}, \mathbf{e}, Cause) \\
 &= \alpha \sum_{\mathbf{y}} \mathbf{P}(Cause) \mathbf{P}(\mathbf{y}|Cause) \left(\prod_j \mathbf{P}(e_j|Cause) \right) \\
 &= \alpha \mathbf{P}(Cause) \left(\prod_j \mathbf{P}(e_j|Cause) \right) \sum_{\mathbf{y}} \mathbf{P}(Cause) \mathbf{P}(\mathbf{y}|Cause) \\
 &= \alpha \mathbf{P}(Cause) \left(\prod_j \mathbf{P}(e_j|Cause) \right)
 \end{aligned}$$

where the last line follows because the summation over \mathbf{y} is 1. Therefore, the algorithm computes the product of the conditional probabilities of the evidence variables given each value of the cause, multiplies each by the prior probability of the cause, and normalizes the result.

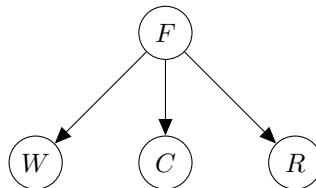


Figure S12.1 A naive Bayes model for fishing.

Exercise 12.6.#FISH

A non-angler, curious to know what counts as a good day of fishing (F) at the lake and puzzled by the phenomenon that it can sometimes be a good day even when no fish are caught, decides to create a naive Bayes model with F as the Boolean class variable and three features: whether it rained (R), how many fish were caught (C) with values $\{\text{none}, \text{some}, \text{lots}\}$, and whether it was windy (W). A naive Bayes net is shown in Figure S12.1.

- a. Here are some plausible probability tables estimated from (entirely made up) data:

$P(F = \text{false})$	$P(F = \text{true})$
0.9	0.1

F	$P(W = \text{false} F)$	$P(W = \text{true} F)$
false	0.5	0.5
true	0.8	0.2

F	$P(C = \text{none} F)$	$P(C = \text{some} F)$	$P(C = \text{lots} F)$
false	0.7	0.2	0.1
true	0.3	0.4	0.3

F	$P(R = \text{false} F)$	$P(R = \text{true} F)$
<i>false</i>	0.6	0.4
<i>true</i>	0.9	0.1

Given this model, calculate the following probabilities:

- (i) $P(F = \text{true} | R = \text{true}, C = \text{none}, W = \text{true})$.
 - (ii) $P(F = \text{true} | R = \text{false}, C = \text{lots}, W = \text{false})$.
 - (iii) $P(F = \text{true} | R = \text{true}, C = \text{some}, W = \text{false})$.
- b. Derive symbolic expressions for the following probabilities in terms of $\mathbf{P}(R | C)$, $\mathbf{P}(C | F)$, $\mathbf{P}(W | F)$, and $\mathbf{P}(F)$:
- (i) $\mathbf{P}(R)$.
 - (ii) $\mathbf{P}(R | C, W)$.
 - (iii) $\mathbf{P}(R, C, W | F)$.
- c. Comment on the plausibility of the conditional independence assumptions made by the naive Bayes model.

a. (i)

$$\begin{aligned}
 & P(F = \text{true} | R = \text{true}, C = \text{none}, W = \text{true}) \\
 &= \frac{P(R = \text{true}, C = \text{none}, W = \text{true} | F = \text{true}) P(F = \text{True})}{P(R = \text{true}, C = \text{none}, W = \text{true})} \\
 &= \frac{P(R = \text{true} | F = \text{true}) P(C = \text{none} | F = \text{true}) P(W = \text{true} | F = \text{true}) P(F = \text{True})}{P(R = \text{true}) P(C = \text{none}) P(W = \text{true})} \\
 &= \frac{.1 \times .3 \times .2 \times 1}{.1 \times .3 \times .2 \times .1 + .4 \times .7 \times .5 \times .9} \\
 &\approx .005.
 \end{aligned}$$

(ii)

$$\begin{aligned}
 & P(F = \text{true} | R = \text{false}, C = \text{lots}, W = \text{false}) \\
 &= \frac{P(R = \text{false} | F = \text{true}) P(C = \text{lots} | F = \text{true}) P(W = \text{false} | F = \text{true}) P(F = \text{True})}{P(R = \text{false}) P(C = \text{lots}) P(W = \text{false})} \\
 &= \frac{.9 \times .3 \times .8 \times 1}{.9 \times .3 \times .8 \times .1 + .6 \times .1 \times .5 \times .9} \\
 &\approx .4.
 \end{aligned}$$

(iii)

$$\begin{aligned}
 & P(F = \text{true} | R = \text{true}, C = \text{some}, W = \text{false}) \\
 &= \frac{P(R = \text{true} | F = \text{true}) P(C = \text{some} | F = \text{true}) P(W = \text{false} | F = \text{true}) P(F = \text{True})}{P(R = \text{true}) P(C = \text{some}) P(W = \text{false})} \\
 &= \frac{.1 \times .4 \times .8 \times 1}{.1 \times .4 \times .8 \times .1 + .9 \times .2 \times .5 \times .9} \\
 &\approx .04.
 \end{aligned}$$

- b. (i) $\mathbf{P}(R) = \sum_f \mathbf{P}(R | f)P(f)$.
(ii) $\frac{\sum_f P(f)P(R | f)P(C | f)P(W | f)}{\sum_f P(f)P(W | f)P(C | f)}$.
(iii) $\mathbf{P}(R | F)\mathbf{P}(C | F)\mathbf{P}(W | F)$.
- c. The assumptions of the model, that W, C, R are independent given F , are unrealistic; wind and rain are not independent and might each reduce the likelihood of catching fish (for only the seasoned anglers would be likely to stay out in such conditions). Also, having a good day fishing is an effect and not a cause of whether it is windy, one catches any fish, and whether it is rainy.

Exercise 12.6.#XXXX

You are given points from 2 classes, shown as rectangles and dots in Figure S12.2. For each of the following sets of points, decide whether the set satisfies or fails to satisfy all the Naïve Bayes modelling assumptions. Note that in (c), 4 rectangles overlap with 4 dots.

The conditional independence assumptions made by the Naïve Bayes model are that features are conditionally independent when given the class. Features being independent once the class label is known means that for a fixed class the distribution for f_1 cannot depend on f_2 , and the other way around. Concretely, for discrete-valued features as shown below, this means each class needs to have a distribution that corresponds to an axis-aligned rectangle. No other assumption is made by the Naïve Bayes model. Note that linear separability is not an assumption of the Naïve Bayes model—what is true is that for a Naïve Bayes model with all binary variables the decision boundary between the two classes is a hyperplane (i.e., it's a linear classifier). That, however, wasn't relevant to the question as the question examined which probability distribution a Naïve Bayes model can represent, not which decision boundaries.

A note about feature independence: The Naïve Bayes model assumes features are conditionally independent given the class. Why does this result in axis-aligned rectangles for discrete feature distributions? Intuitively, this is because fixing one value is uninformative about the other: within a class, the values of one feature are constant across the other. For instance, the dark square class in (b) has $f_1 \in [-0.5, 0.5]$ and $f_2 \in [-1, 0]$ and fixing one has no impact on the domain of the other. However, when the features of a class are not axis-aligned then fixing one limits the domain of the other, inducing dependence. In (e), fixing $f_2 = 1.5$ restricts f_1 to the two points at the top, whereas fixing $f_2 = 0$ gives a larger domain.

Exercise 12.6.#BAYS

Text categorization is the task of assigning a given document to one of a fixed set of categories on the basis of the text it contains. Naive Bayes models are often used for this task. In these models, the query variable is the document category, and the “effect” variables are the presence or absence of each word in the language; the assumption is that words occur independently in documents, with frequencies determined by the document category.

- a. Explain precisely how such a model can be constructed, given as “training data” a set

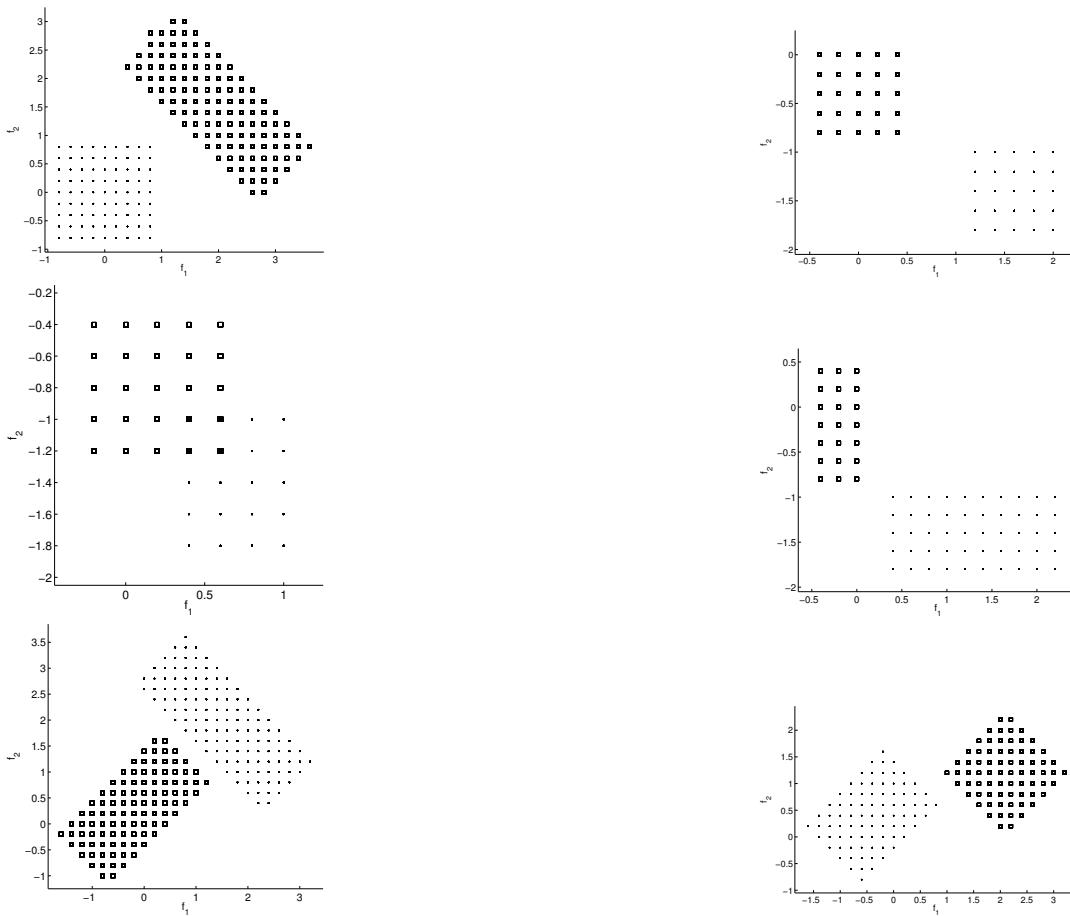


Figure S12.2 Six data sets that may or may not satisfy the naive Bayes assumption.

of documents that have been assigned to categories.

- Explain precisely how to categorize a new document.
- Is the conditional independence assumption reasonable? Discuss.

This question is essentially previewing material in Chapter 23 (page 842), but students should have little difficulty in figuring out how to estimate a conditional probability from complete data.

- The model consists of the prior probability $\mathbf{P}(Category)$ and the conditional probabilities $\mathbf{P}(Word_i|Category)$. For each category c , $\mathbf{P}(Category = c)$ is estimated as the fraction of all documents that are of category c . Similarly, $\mathbf{P}(Word_i = true|Category = c)$ is estimated as the fraction of documents of category c that contain word i .
- See the answer for 13.17. Here, every evidence variable is observed, since we can tell

if any given word appears in a given document or not.

- c. The independence assumption is clearly violated in practice. For example, the word pair “artificial intelligence” occurs more frequently in any given document category than would be suggested by multiplying the probabilities of “artificial” and “intelligence”.

12.7 The Wumpus World Revisited

Exercise 12.7.#PITP

In our analysis of the wumpus world, we used the fact that each square contains a pit with probability 0.2, independently of the contents of the other squares. Suppose instead that exactly $N/5$ pits are scattered at random among the N squares other than [1,1]. Are the variables $P_{i,j}$ and $P_{k,l}$ still independent? What is the joint distribution $\mathbf{P}(P_{1,1}, \dots, P_{4,4})$ now? Redo the calculation for the probabilities of pits in [1,3] and [2,2].

This probability model is also appropriate for Minesweeper (Ex. 7.11). If the total number of pits is fixed, then the variables $P_{i,j}$ and $P_{k,l}$ are no longer independent. In general,

$$P(P_{i,j} = \text{true} | P_{k,l} = \text{true}) < P(P_{i,j} = \text{true} | P_{k,l} = \text{false})$$

because learning that $P_{k,l} = \text{true}$ makes it less likely that there is a mine at $[i, j]$ (as there are now fewer to spread around). The joint distribution places equal probability on all assignments to $P_{1,2} \dots P_{4,4}$ that have exactly 3 pits, and zero on all other assignments. Since there are 15 squares, the probability of each 3-pit assignment is $1/\binom{15}{3} = 1/455$.

To calculate the probabilities of pits in [1, 3] and [2, 2], we start from Figure 13.7. We have to consider the probabilities of complete assignments, since the probability of the “other” region assignment does not cancel out. We can count the total number of 3-pit assignments that are consistent with each partial assignment in 13.7(a) and 13.7(b).

In 13.7(a), there are three partial assignments with $P_{1,3} = \text{true}$:

- The first fixes all three pits, so corresponds to 1 complete assignment.
- The second leaves 1 pit in the remaining 10 squares, so corresponds to 10 complete assignments.
- The third also corresponds to 10 complete assignments.

Hence, there are 21 complete assignments with $P_{1,3} = \text{true}$.

In 13.7(b), there are two partial assignments with $P_{1,3} = \text{false}$:

- The first leaves 1 pit in the remaining 10 squares, so corresponds to 10 complete assignments.
- The second leaves 2 pits in the remaining 10 squares, so corresponds to $\binom{10}{2} = 45$ complete assignments.

Hence, there are 55 complete assignments with $P_{1,3} = \text{false}$. Normalizing, we obtain

$$\mathbf{P}(P_{1,3}) = \alpha/21, 55 = \langle 0.276, 0.724 \rangle .$$

With $P_{2,2} = \text{true}$, there are four partial assignments with a total of $\binom{10}{2} + 2 \cdot \binom{10}{1} + \binom{10}{0} = 66$ complete assignments. With $P_{2,2} = \text{false}$, there is only one partial assignment with $\binom{10}{1} = 10$ complete assignments. Hence

$$\mathbf{P}(P_{2,2}) = \alpha \langle 66, 10 \rangle = \langle 0.868, 0.132 \rangle .$$

Exercise 12.7.#PITQ

Redo the probability calculation for pits in [1,3] and [2,2], assuming that each square contains a pit with probability 0.01, independent of the other squares. What can you say about the relative performance of a logical versus a probabilistic agent in this case?

First we redo the calculations of $P(\text{frontier})$ for each model in Figure 12.6. The three models with $P_{1,3} = \text{true}$ have probabilities 0.0001, 0.0099, 0.0099; the two models with $P_{1,3} = \text{false}$ have probabilities 0.0001, 0.0099. Then

$$\begin{aligned}\mathbf{P}(P_{1,3} | \text{known}, b) &= \alpha' \langle 0.01(0.0001 + 0.0099 + 0.0099), 0.99(0.0001 + 0.0099) \rangle \\ &\approx \langle 0.1674, 0.8326 \rangle .\end{aligned}$$

The four models with $P_{2,2} = \text{true}$ have probabilities 0.0001, 0.0099, 0.0099, 0.9801; the one model with $P_{2,2} = \text{false}$ has probability 0.0001. Then

$$\begin{aligned}\mathbf{P}(P_{2,2} | \text{known}, b) &= \alpha' \langle 0.01(0.0001 + 0.0099 + 0.0099 + 0.9801), 0.99 \times 0.0001 \rangle \\ &\approx \langle 0.9902, 0.0098 \rangle .\end{aligned}$$

This means that [2,2] is almost certain death; a probabilistic agent can figure this out and choose [1,3] or [3,1] instead. Its chance of death at this stage will be 0.1674, while a logical agent choosing at random among the three squares will die with probability $(0.1674 + 0.9902 + 0.1674)/3 = 0.4416$. The reason that [2,2] is so much more likely to be a pit in this case is that, for it *not* to be a pit, *both* of [1,3] and [3,1] must contain pits, which is very unlikely. Indeed, as the prior probability of pits tends to 0, the posterior probability of [2,2] tends to 1.

Exercise 12.7.#WUMA

Implement a hybrid probabilistic agent for the wumpus world, based on the hybrid agent in Figure 7.20 and the probabilistic inference procedure outlined in this chapter.

The solution for this exercise is omitted. The main modification to the agent in Figure 7.20 is to calculate, after each move, the safety probability for each square that is not provably safe or fatal, and choose the safest if there is no unvisited safe square.

EXERCISES 13

PROBABILISTIC REASONING

13.1 Representing Knowledge in an Uncertain Domain

Exercise 13.1.#MRFS

In this chapter we present Bayesian networks, originally called belief networks, for the representation of uncertainty but other types of graphical models may also be used. Consider, **Markov networks**, also called Markov random fields, cited in the bibliographical and historical notes. What differentiates Markov and Bayesian networks? What are advantages and disadvantages of each? What is a domain in which each network might be more suitable than the other?

Bayesian networks are directed acyclic graphs; Markov networks are undirected cyclic graphs. These formulations necessarily admit different independence assumptions; d-separation is necessary to determine the conditional independence of nodes in a Bayesian network while only separation between two nodes (through any path) is necessary in a Markov network. Additionally, values in Markov networks do not represent probabilities but arbitrary-valued potentials which, in Bayesian networks, must be conditional probabilities. Because of these differences finding the joint probability of a Markov networks is NP-hard while it is considerably easier in a Bayesian network (one must compute the product of all nodes in a Markov network as opposed to successive products of directed subsets in a Bayes net). Medical diagnosis in which nodes (such as symptoms and diseases) represent causal relationships would better suit Bayesian networks while image processing in which nodes (such as pixels in an image) do not represent causality would better suit Markov networks.

13.2 The Semantics of Bayesian Networks

Exercise 13.2.#DAGS

Recall that any directed acyclic graph G has an associated family of probability distributions, which consists of all probability distributions that can be represented by a Bayes net with structure G . Consider the six directed acyclic graphs in Figure S13.1.

- a. Assume all we know about the joint distribution $P(A, B, C)$ is that it can be represented by the product $P(A | B, C)P(B | C)P(C)$. Which of the six graphs are guaranteed to be able to represent $P(A, B, C)$?

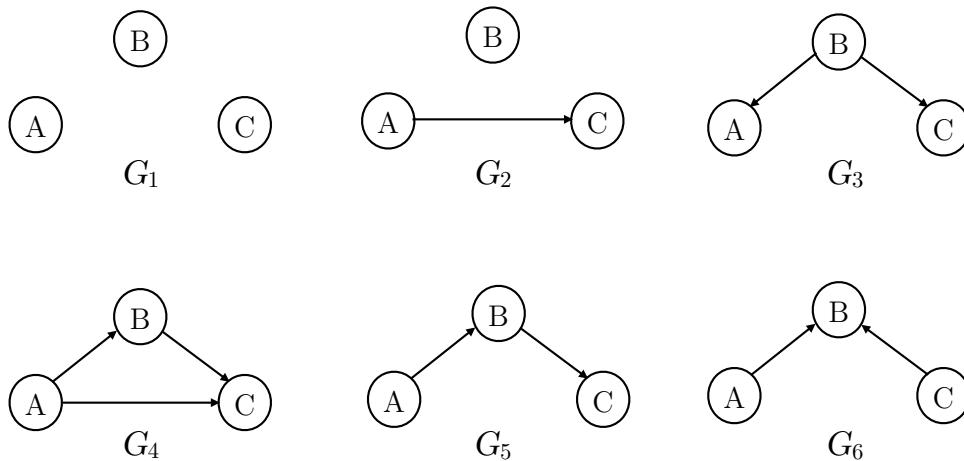


Figure S13.1 Six directed acyclic graphs.

- b. Now assume all we know about the joint distribution $P(A, B, C)$ is that it can be represented by the product $P(C | B)P(B | A)P(A)$. Which of the six graphs are guaranteed to be able to represent $P(A, B, C)$?
- a. In this case, the decomposition of $P(A, B, C)$ is just that given by the chain rule, so no conditional independence is implied. G_4 is fully connected, and is therefore able to represent any joint distribution. The others assert conditional independences and so may not be able to represent $P(A, B, C)$.
- b. G_3 , G_4 , and G_5 can represent the distribution. G_1 assumes all variables are independent; G_2 asserts that B is independent of the others; and G_6 asserts $A \perp\!\!\!\perp C$.

Exercise 13.2.#ABDE

Consider a Bayes net over the random variables A, B, C, D, E with the structure shown in Figure S13.2, with full joint distribution $P(A, B, C, D, E)$.

- a. Consider the marginal distribution $P(A, B, D, E) = \sum_c P(A, B, c, D, E)$, where C was eliminated. Draw the minimal Bayes net that is guaranteed to be able to represent this distribution.
- b. Assume we are given an observation: $A = a$. Draw the minimal Bayes net that is guaranteed to be able to represent the distribution $P'(B, C, D, E) = P(B, C, D, E | A = a)$.

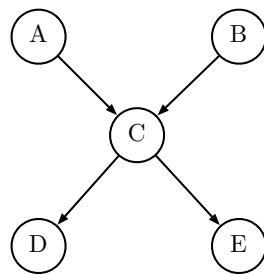
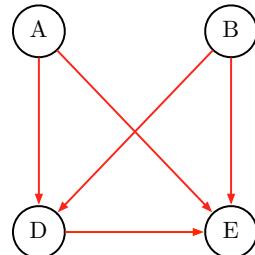


Figure S13.2 A Bayes net over the random variables A, B, C, D, E .

- c. Assume we are given two observations: $D = d, E = e$. Draw the minimal Bayes net that is guaranteed to be able to represent the distribution $P''(A, B, C) = P(A, B, C | D = d, E = e)$.

- a. For $P(A, B, D, E)$ we have



The high level overview for these types of problems is that the resultant graph must be able to encode the same conditional independence assumptions from the initial Bayes net. In the BN above, we have the following independence assumptions:

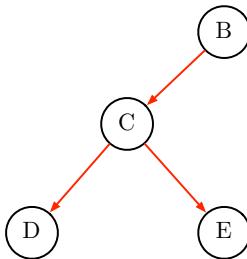
- $A \perp\!\!\!\perp B$
- $A \perp\!\!\!\perp D | C$
- $B \perp\!\!\!\perp D | C$
- $A \perp\!\!\!\perp E | C$
- $B \perp\!\!\!\perp E | C$
- $D \perp\!\!\!\perp E | C$

When we marginalize out C , we remove C from the graph. The conditional independence assumptions involving C no longer matter, so we just need to preserve:

$$A \perp\!\!\!\perp B .$$

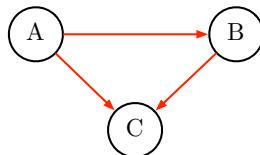
The graph shown guarantees this. The arrow between D and E could go either way.

- b. For $P'(B, C, D, E) = P(B, C, D, E | A = a)$, we have



Observing A fixes its value and removes it from the Bayes net. By d-separation no further dependence is introduced.

- c. For $P''(A, B, C) = P(A, B, C | D = d, E = e)$ we have



Observing D and E makes an active path to their parent C , which in turn activates the common-effect triple ABC and renders A and B dependent.

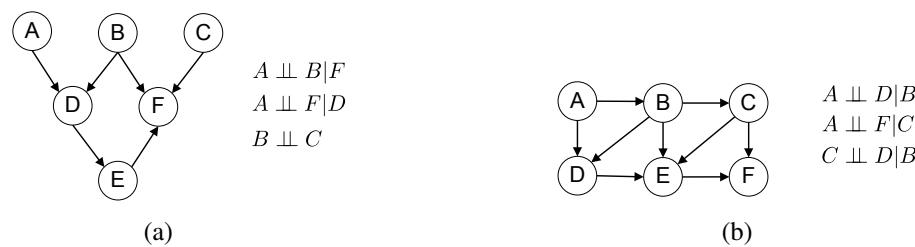


Figure S13.3 Figure for Exercise 13.RMVE.

Exercise 13.2.#RMVE

For the graphs in Figure S13.3, what is the minimal set of edges that must be removed such that the corresponding independence relations are guaranteed to be true?

- a. Remove AD .

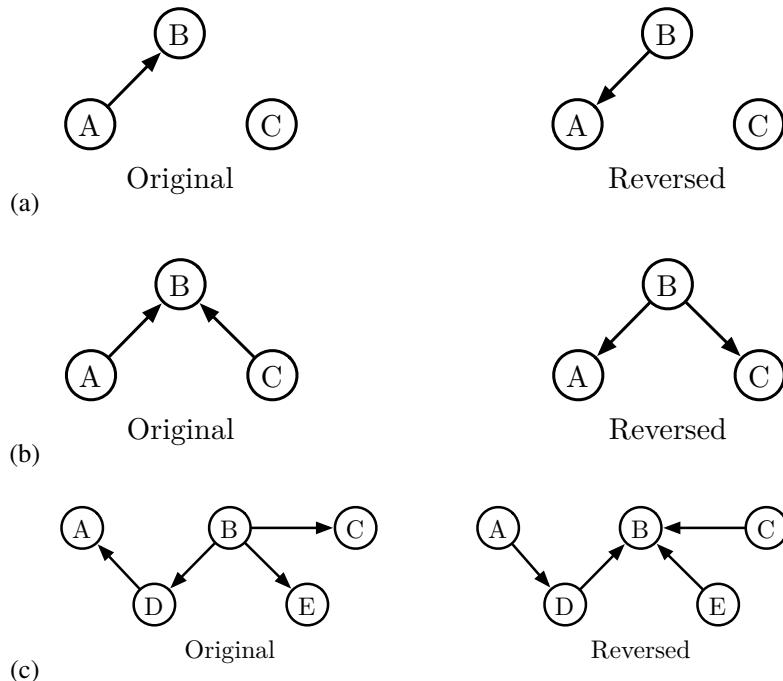


Figure S13.4 Figure for Exercise 13.REVB.

- b. Remove AD and either EF or AB .

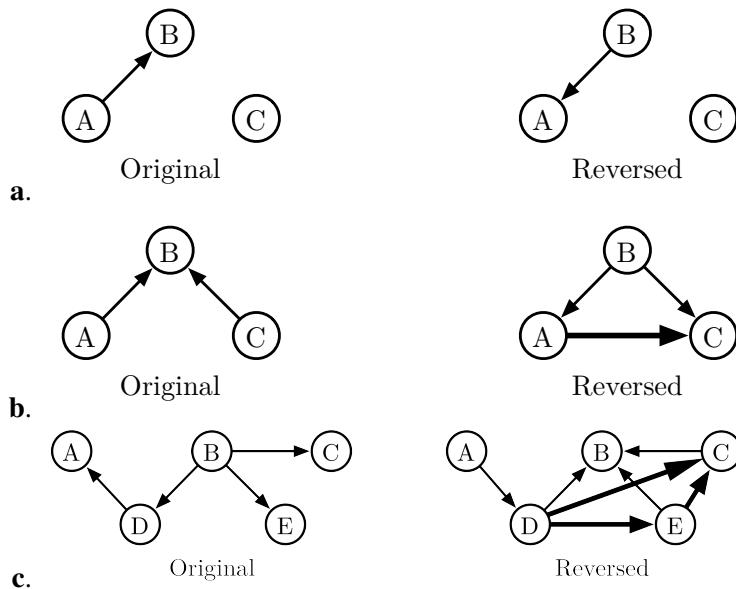
Exercise 13.2.#REVB

Figure S13.3 shows pairs of Bayes nets. In each, the **original** network is shown on the left. The **reversed** network, shown on the right, has all the arrows reversed. Therefore, the reversed network may not be able to represent all of the distributions that the original network is able to represent.

For each pair, add the *minimal* number of arrows to the **reversed** network such that it is guaranteed to be able to represent all the distributions that the **original** network is able to represent.

The reversal of common effects and common causes requires more arrows so that the dependence of children (with parents unobserved) and the dependence of parents (with a common child observed) can be captured.

To guarantee that a Bayes net B' is able to represent all of the distributions of an original Bayes net B , B' must only make a subset of the independence assumptions in B . If independences are in B' that are not in B , then the distributions of B' have constraints that could prevent it from representing a distribution of B .



(a)



(b)

Figure S13.5 Figure for Exercise 13.MINB.**Exercise 13.2.#MINB**

For the following Bayes nets, add the *minimal* number of arrows such that the resulting structure is able to represent all distributions that satisfy the stated independence and non-independence constraints (note these are constraints on the Bayes net *structure*, so each non-independence constraint should be interpreted as disallowing all structures that make the given independence assumption). If no arrows are needed, write “none needed.” If no such Bayes net is possible, write “not possible.”

a. See Figure S13.5 (a). **Constraints:**

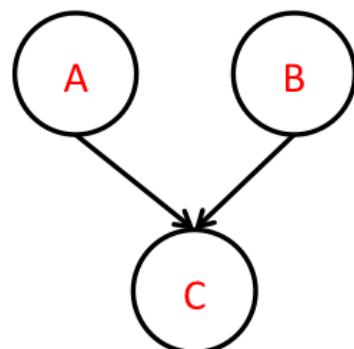
- $A \perp\!\!\!\perp B$
- $\text{not } A \perp\!\!\!\perp B | \{C\}$

b. See Figure S13.5 (b). **Constraints:**

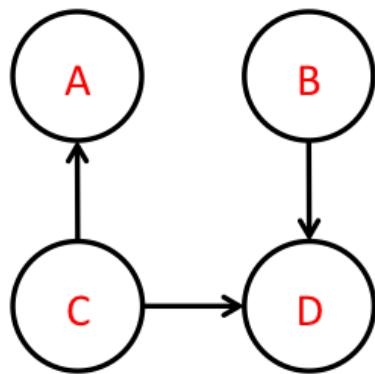
- $A \perp\!\!\!\perp D | \{C\}$
- $A \perp\!\!\!\perp B$
- $B \perp\!\!\!\perp C$
- $\text{not } A \perp\!\!\!\perp B | \{D\}$

c. See Figure S13.5 (b). **Constraints:**

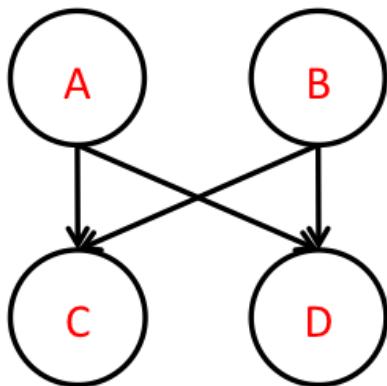
- $A \perp\!\!\!\perp B$
- $C \perp\!\!\!\perp D | \{A, B\}$
- $\text{not } C \perp\!\!\!\perp D | \{A\}$
- $\text{not } C \perp\!\!\!\perp D | \{B\}$



a.



b.



c.

Exercise 13.2.#CPTE

Equation (13.1) on page 415 defines the joint distribution represented by a Bayesian network in terms of the parameters $\theta(X_i | \text{Parents}(X_i))$. This exercise asks you to derive the equivalence between the parameters and the conditional probabilities $\mathbf{P}(X_i | \text{Parents}(X_i))$ from this definition.

- Consider a simple network $X \rightarrow Y \rightarrow Z$ with three Boolean variables. Use Equations (12.3) and (12.7) (pages 390 and 396) to express the conditional probability $P(z | y)$ as the ratio of two sums, each over entries in the joint distribution $\mathbf{P}(X, Y, Z)$.
- Now use Equation (13.1) to write this expression in terms of the network parameters $\theta(X)$, $\theta(Y | X)$, and $\theta(Z | Y)$.
- Next, expand out the summations in your expression from part (b), writing out explicitly the terms for the true and false values of each summed variable. Assuming that all network parameters satisfy the constraint $\sum_{x_i} \theta(x_i | \text{parents}(X_i)) = 1$, show that the resulting expression reduces to $\theta(z | y)$.
- Generalize this derivation to show that $\theta(X_i | \text{Parents}(X_i)) = \mathbf{P}(X_i | \text{Parents}(X_i))$ for any Bayesian network.

This question is quite tricky and students may require additional guidance, particularly on the last part. It does, however, help them become comfortable with operating on complex sum-of-product expressions, which are at the heart of graphical models.

- By Equations (12.3) and (12.7), we have

$$P(z | y) = \frac{P(y, z)}{P(y)} = \frac{\sum_x P(x, y, z)}{\sum_{x, z'} P(x, y, z')} .$$

- By Equation (13.1), this can be written as

$$P(z | y) = \frac{\sum_x \theta(x) \theta(y | x) \theta(z | y)}{\sum_{x, z'} \theta(x) \theta(y | x) \theta(z' | y)} .$$

- For students who are not familiar with direct manipulation of summation expressions,

the expanding-out step makes it a bit easier to see how to simplify the expressions. Expanding out the sums, collecting terms, using the sum-to-1 property of the parameters, and finally cancelling, we have

$$\begin{aligned} P(z|y) &= \frac{\theta(x)\theta(y|x)\theta(z|y) + \theta(\neg x)\theta(y|\neg x)\theta(z|y)}{\theta(x)\theta(y|x)\theta(z|y) + \theta(x)\theta(y|x)\theta(\neg z|y) + \theta(\neg x)\theta(y|\neg x)\theta(z|y) + \theta(\neg x)\theta(y|\neg x)\theta(\neg z|y)} \\ &= \frac{\theta(z|y) [\theta(x)\theta(y|x) + \theta(\neg x)\theta(y|\neg x)]}{[\theta(x)\theta(y|x) + \theta(\neg x)\theta(y|\neg x)] [\theta(z|y) + \theta(\neg z|y)]} \\ &= \frac{\theta(z|y) [\theta(x)\theta(y|x) + \theta(\neg x)\theta(y|\neg x)]}{[\theta(x)\theta(y|x) + \theta(\neg x)\theta(y|\neg x)]} \\ &= \theta(z|y). \end{aligned}$$

If, instead, students are prepared to work on the summations directly, the key step is moving the sum over z' inwards::

$$\begin{aligned} P(z|y) &= \frac{\theta(z|y) \sum_x \theta(x)\theta(y|x)}{\sum_x \theta(x)\theta(y|x) \sum_{z'} \theta(z'|y)} \\ &= \frac{\theta(z|y) \sum_x \theta(x)\theta(y|x)}{\sum_x \theta(x)\theta(y|x)} \\ &= \theta(z|y). \end{aligned}$$

- d. The general case is a bit more difficult—the key to a simple proof is figuring out how to split up all the variables. First, however, we need a little lemma: for any set of variables \mathbf{V} , we have

$$\sum_{\mathbf{v}} \prod_i \theta(v_i | pa(V_i)) = 1.$$

This generalizes the sum-to-1 rule for a single variable, and is easily proved by induction given any topological ordering for the variables in \mathbf{V} .

One of the principal rules for manipulating nested summations is that a particular summation can be pushed to the right as long as all occurrences of that variable remain to the right of the summation. For this reason, the *descendants* of Z , which we will call \mathbf{U} , are a very useful subset of the variables in the network. In particular, they have the property that they cannot be parents of any other variable in the network. (If there was such a variable, it would be a descendant of Z by definition!) We will divide the variables into Z , \mathbf{Y} (the parents of Z), \mathbf{U} (the descendants of Z), and \mathbf{X} (all other

variables). We know that variables in \mathbf{X} and \mathbf{Y} have no parents in Z and \mathbf{U} . So we have

$$\begin{aligned}
 P(z | \mathbf{y}) &= \frac{\sum_{\mathbf{x}, \mathbf{u}} P(\mathbf{x}, \mathbf{y}, z, \mathbf{u})}{\sum_{\mathbf{x}, z', \mathbf{u}} P(\mathbf{x}, \mathbf{y}, z', \mathbf{u})} \\
 &= \frac{\sum_{\mathbf{x}, \mathbf{u}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \theta(z | \mathbf{y}) \prod_k \theta(u_k | pa(U_k))}{\sum_{\mathbf{x}, z', \mathbf{u}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \theta(z' | \mathbf{y}) \prod_k \theta(u_k | pa(U_k))} \\
 &= \frac{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \theta(z | \mathbf{y}) \sum_{\mathbf{u}} \prod_k \theta(u_k | pa(U_k))}{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \sum_{z'} \theta(z' | \mathbf{y}) \sum_{\mathbf{u}} \prod_k \theta(u_k | pa(U_k))} \\
 &\quad \text{(moving the sums in as far as possible)} \\
 &= \frac{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \theta(z | \mathbf{y})}{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \sum_{z'} \theta(z' | \mathbf{y})} \\
 &\quad \text{(using the generalized sum-to-1 rule for } \mathbf{u} \text{)} \\
 &= \frac{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j)) \theta(z | \mathbf{y})}{\sum_{\mathbf{x}} \prod_i \theta(x_i | pa(X_i)) \prod_j \theta(y_j | pa(Y_j))} \\
 &\quad \text{(using the sum-to-1 rule for } z' \text{)} \\
 &= \theta(z | \mathbf{y}).
 \end{aligned}$$

Exercise 13.2.#ARCR

The operation of **arc reversal** in a Bayesian network allows us to change the direction of an arc $X \rightarrow Y$ while preserving the joint probability distribution that the network represents (Shachter, 1986). Arc reversal may require introducing new arcs: all the parents of X also become parents of Y , and all parents of Y also become parents of X .

- a. Assume that X and Y start with m and n parents, respectively, and that all variables have k values. By calculating the change in size for the CPTs of X and Y , show that the total number of parameters in the network cannot decrease during arc reversal. (*Hint:* the parents of X and Y need not be disjoint.)
- b. Under what circumstances can the total number remain constant?
- c. Let the parents of X be $\mathbf{U} \cup \mathbf{V}$ and the parents of Y be $\mathbf{V} \cup \mathbf{W}$, where \mathbf{U} and \mathbf{W} are disjoint. The formulas for the new CPTs after arc reversal are as follows:

$$\begin{aligned}
 \mathbf{P}(Y | \mathbf{U}, \mathbf{V}, \mathbf{W}) &= \sum_x \mathbf{P}(Y | \mathbf{V}, \mathbf{W}, x) \mathbf{P}(x | \mathbf{U}, \mathbf{V}) \\
 \mathbf{P}(X | \mathbf{U}, \mathbf{V}, \mathbf{W}, Y) &= \mathbf{P}(Y | X, \mathbf{V}, \mathbf{W}) \mathbf{P}(X | \mathbf{U}, \mathbf{V}) / \mathbf{P}(Y | \mathbf{U}, \mathbf{V}, \mathbf{W}).
 \end{aligned}$$

Prove that the new network expresses the same joint distribution over all variables as the original network.

- a. Suppose that X and Y share l parents. After the reversal Y will gain $m - l$ new parents, the $m - l$ original parents of X that it does not share with Y , and loses one parent: X . After the reversal X will gain $n - l$ new parents, the $n - l - 1$ original parents of Y that it does not share with X and isn't X itself, and plus Y . So, after the reversal Y will have $n + (m - l - 1) = m + (n - l - 1)$ parents, and X will have $m + (n - l) = n + (m - l)$

parents.

Observe that $m - l \geq 0$, since this is the number of original parents of X not shared with Y , and that $n - l - 1 \geq 0$, since this is the number of original parents of Y not shared with X and not equal to X . This shows the number of parameters can only increase: before we had $k^m + k^n$, after we have $k^{m+(n-l-1)} + k^{n+(m-l)}$.

(As a sanity check on our counting above, if we are reversing a single arc without any extra parents, we have $l = 0$, $m = 0$, and $n = 1$; the previous formulas say we will have $m' = 0$ and $n' = 1$ afterwards, which is correct.)

- b. For the number of parameters to remain constant, assuming that $k > 1$, requires by our previous calculation that $m - l = 0$ and $n - l - 1 = 0$. This holds exactly when X and Y share all their parents originally (except Y also has X as a parent).

- c. For clarity, denote by $P'(Y|U, V, W)$ and $P'(X|U, V, W, Y)$ the new CPTs, and note that the set of variables $V \cup W$ does not include X . It suffices to show that

$$P'(X, Y|U, V, W) = P(X, Y|U, V, W)$$

To see this, let D denote the variables, outside of $\{X, Y\} \cup U \cup V \cup W$, which have either X or Y as ancestor in the original network, and \bar{D} those which don't. Since the arc reversed graph only adds or removes arcs incoming to X or Y , it cannot change which variables lie in D or \bar{D} . We then have

$$\begin{aligned} P(D, \bar{D}, X, Y, U, V, W) &= P(\bar{D}, U, V, W)P(X, Y|U, V, W)P(D|X, Y, U, V, W) \\ &= P'(\bar{D}, U, V, W)P(X, Y|U, V, W)P(D|X, Y, U, V, W) \\ &= P'(\bar{D}, U, V, W)P(X, Y|U, V, W)P'(D|X, Y, U, V, W) \\ &= P'(\bar{D}, U, V, W)P'(X, Y|U, V, W)P'(D|X, Y, U, V, W) \\ &= P'(D, \bar{D}, X, Y, U, V, W) \end{aligned}$$

the second as arc reversal does not change the CPTs of variables in \bar{D}, U, V, W contains all its parents, the third as if we condition on X, Y, U, V, W the original and arc-reversed Bayesian networks are the same, and the fourth by hypothesis.

Then, calculating:

$$\begin{aligned}
 P'(X, Y|U, V, W) &= P'(Y|U, V, W)P'(X|U, V, W, Y) \\
 &= \left(\sum_x P(Y|V, W, x)P(x|U, V) \right) P(Y|X, V, W)P(X|U, V) \\
 &= \left(\sum_x P(Y|U, V, W, x)P(x|U, V)/P(Y|U, V, W) \right) P(Y|X, V, W)P(X|U, V) \\
 &= \left(\sum_x \frac{P(Y, U, V, W, x)P(x, U, V)P(U, V, W)}{P(U, V, W, x)P(U, V)P(Y, U, V, W)} \right) P(Y|X, V, W)P(X|U, V) \\
 &= \left(\sum_x P(x|Y, U, V, W)P(x|U, V)/P(x|U, V, W) \right) P(Y|X, V, W)P(X|U, V) \\
 &= \left(\sum_x P(x|Y, U, V, W) \right) P(Y|X, V, W)P(X|U, V) \\
 &= P(Y|X, V, W)P(X|U, V)
 \end{aligned}$$

where the third step follows as V, W, x is the parent set of Y it's conditionally independent of U , and the second to last step follows as U, V is the parent set of X it's conditionally independent of x .

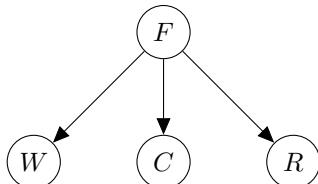


Figure S13.6 A naive Bayes model for fishing.

Exercise 13.2.#BFSH

Exercise Exercise 12.FISH introduced the naive Bayes model in Figure S13.6. (F) is true iff today was a good day of fishing. There are three features: whether it rained (R), how many fish were caught (C) with values $\{\text{none}, \text{some}, \text{lots}\}$, and whether it was windy (W).

For each of the true distributions in Figure S13.7, determine whether the Naive Bayes modeling assumption holds and whether the naive Bayes model is guaranteed to be able to represent the true conditional probability, $\mathbf{P}(F|W, R, C)$.

- a. The naive Bayes assumption does not hold as W is not independent of R given the class F . However, based on the true distribution model, we can conclude that $P(F|W, R, C) =$

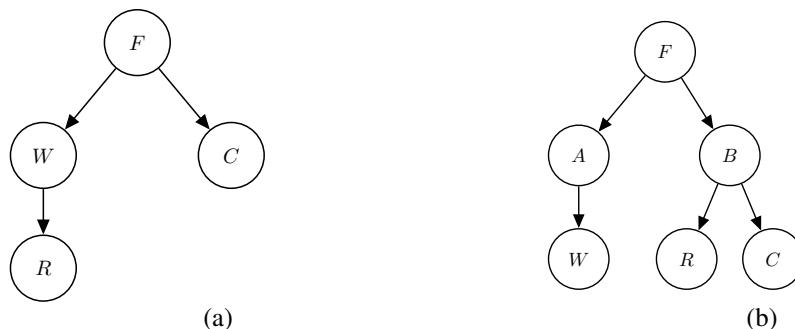


Figure S13.7 Two true distributions for Exercise 13.BFSH.

$P(F|W, C)$. In other words, R is not required to model the conditional distribution. If we set $P(R|F) = \text{constant}$ (same for both values of F), the naive Bayes model will also have $P(F|W, R, C) = P(F|W, C)$ and can exactly model the conditional distribution for F implied by this Bayes net.

- b. The naive Bayes assumption does not hold as R is not independent of C given the class F and it cannot represent the true distribution.

Exercise 13.2.#COIN

We have a bag of three biased coins a , b , and c with probabilities of coming up heads of 30%, 60%, and 75%, respectively. One coin is drawn randomly from the bag (with equal likelihood of drawing each of the three coins), and then the coin is flipped three times to generate the outcomes X_1 , X_2 , and X_3 .

- a. Draw the Bayesian network corresponding to this setup and define the necessary CPTs.
- b. Calculate which coin was most likely to have been drawn from the bag if the observed flips come out heads twice and tails once.

- a. With the random variable C denoting which coin $\{a, b, c\}$ we drew, the network has C at the root and X_1 , X_2 , and X_3 as children.

The CPT for C is:

C	$P(C)$
a	$1/3$
b	$1/3$
c	$1/3$

The CPT for X_i given C are the same, and equal to:

C	X_1	$P(C)$
a	<i>heads</i>	0.3
b	<i>heads</i>	0.6
c	<i>heads</i>	0.75

- b. The coin most likely to have been drawn from the bag given this sequence is the value of C with greatest posterior probability $P(C|2 \text{ heads}, 1 \text{ tails})$. Now,

$$\begin{aligned} P(C|2 \text{ heads}, 1 \text{ tails}) &= P(2 \text{ heads}, 1 \text{ tails}|C)P(C)/P(2 \text{ heads}, 1 \text{ tails}) \\ &\propto P(2 \text{ heads}, 1 \text{ tails}|C)P(C) \\ &\propto P(2 \text{ heads}, 1 \text{ tails}|C) \end{aligned}$$

where in the second line we observe that the constant of proportionality $1/P(2 \text{ heads}, 1 \text{ tails})$ is independent of C , and in the last we observe that $P(C)$ is also independent of the value of C since it is, by hypothesis, equal to $1/3$.

From the Bayesian network we can see that X_1 , X_2 , and X_3 are conditionally independent given C , so for example

$$\begin{aligned} P(X_1 = \text{tails}, X_2 = \text{heads}, X_3 = \text{heads}|C = a) &= P(X_1 = \text{tails}|C = a)P(X_2 = \text{heads}|C = a)P(X_3 = \text{heads}|C = a) \\ &= 0.7 \times 0.3 \times 0.3 = 0.063 \end{aligned}$$

Note that since the CPTs for each coin are the same, we would get the same probability above for any ordering of 2 heads and 1 tails. Since there are three such orderings, we have

$$P(2\text{heads}, 1\text{tails}|C = a) = 3 \times 0.063 = 0.189.$$

Similar calculations to the above find that

$$\begin{aligned} P(2\text{heads}, 1\text{tails}|C = b) &= 0.432 \\ P(2\text{heads}, 1\text{tails}|C = c) &= 0.422 \end{aligned}$$

showing that coin b is most likely to have been drawn.

Alternatively, one could directly compute the value of $P(C|2 \text{ heads}, 1 \text{ tails})$.

Exercise 13.2.#BURG

Consider the Bayesian network in Figure 13.2.

- If no evidence is observed, are *Burglary* and *Earthquake* independent? Prove this from the numerical semantics and from the topological semantics.
- If we observe *Alarm = true*, are *Burglary* and *Earthquake* independent? Justify your answer by calculating whether the probabilities involved satisfy the definition of conditional independence.

- a. Yes. From the numerical semantics, we have

$$\begin{aligned}
 P(B, E) &= \sum_{a,j,m} P(B, E, a, j, m) = \sum_{a,j,m} P(B)P(E)P(a|B, E)P(j|a)P(m|a) \\
 &= P(B)P(E) \sum_{a,j,m} P(a|B, E)P(j|a)P(m|a) \\
 &= P(B)P(E) \sum_a P(a|B, E) \left(\sum_j P(j|a) \right) \left(\sum_m P(m|a) \right) \\
 &= P(B)P(E)
 \end{aligned}$$

using the sum-to-1 constraint for the conditional distributions. Topologically, E is independent of its non-descendants (i.e., B) given its parents (i.e., the empty set), so B and E are absolutely independent.

- b. We check whether $P(B, E|a) = P(B|a)P(E|a)$. First computing $P(B, E|a)$

$$\begin{aligned}
 P(B, E|a) &= \alpha P(a|B, E)P(B, E) \\
 &= \alpha \begin{cases} .95 \times 0.001 \times 0.002 & \text{if } B = b \text{ and } E = e \\ .94 \times 0.001 \times 0.998 & \text{if } B = b \text{ and } E = \neg e \\ .29 \times 0.999 \times 0.002 & \text{if } B = \neg b \text{ and } E = e \\ .001 \times 0.999 \times 0.998 & \text{if } B = \neg b \text{ and } E = \neg e \end{cases} \\
 &= \alpha \begin{cases} 0.0008 & \text{if } B = b \text{ and } E = e \\ 0.3728 & \text{if } B = b \text{ and } E = \neg e \\ 0.2303 & \text{if } B = \neg b \text{ and } E = e \\ 0.3962 & \text{if } B = \neg b \text{ and } E = \neg e \end{cases}
 \end{aligned}$$

where α is a normalization constant. Checking whether $P(b, e|a) = P(b|a)P(e|a)$ we find

$$P(b, e|a) = 0.0008 \neq 0.0863 = 0.3736 \times 0.2311 = P(b|a)P(e|a)$$

showing that B and E are not conditionally independent given A .

Exercise 13.2.#MRBL

Suppose that in a Bayesian network containing an unobserved variable Y , all the variables in the Markov blanket $MB(Y)$ have been observed. Suppose we now remove the node Y from the network. Y 's parents lose a child, and Y 's children lose a parent. The CPTs for Y 's children given their now-reduced sets of parents are reset to *arbitrary* values. Prove that the posterior distribution $Q(X_i|mb(Y))$ for any other unobserved variable X_i in the new network is identical to $P(X_i|mb(Y))$ in the original network.

Let \mathbf{X} be the set of all variables in the original Bayesian network except for Y and $MB(Y)$, with $X_i \in \mathbf{X}$, and let $MB(Y)$ be comprised of $\mathbf{W} = \text{children}(Y)$ and the remaining variables \mathbf{U} . The key point here is that in the new network defined by Q , the parents of variables in \mathbf{W} are all variables in $MB(W)$, so they are all *observed* variables; and the only differences in the CPTs for variables in Q compared to P are for the variables in \mathbf{W} .

Hence, we have

$$\begin{aligned}
 Q(\mathbf{X}|mb(Y)) &= \alpha Q(\mathbf{X}, mb(Y)) \\
 &= \alpha \prod_i Q(X_i|Pa(X_i)) \prod_j Q(U_j|Pa(U_j)) \prod_k Q(W_k|pa(W_k)) \\
 &= \alpha' \prod_i Q(X_i|Pa(X_i)) \prod_j Q(U_j|Pa(U_j)) \\
 &= \alpha' \prod_i P(X_i|Pa(X_i)) \prod_j P(U_j|Pa(U_j)) = \alpha' P(\mathbf{X}, mb(Y)) = P(\mathbf{X}|mb(Y)).
 \end{aligned}$$

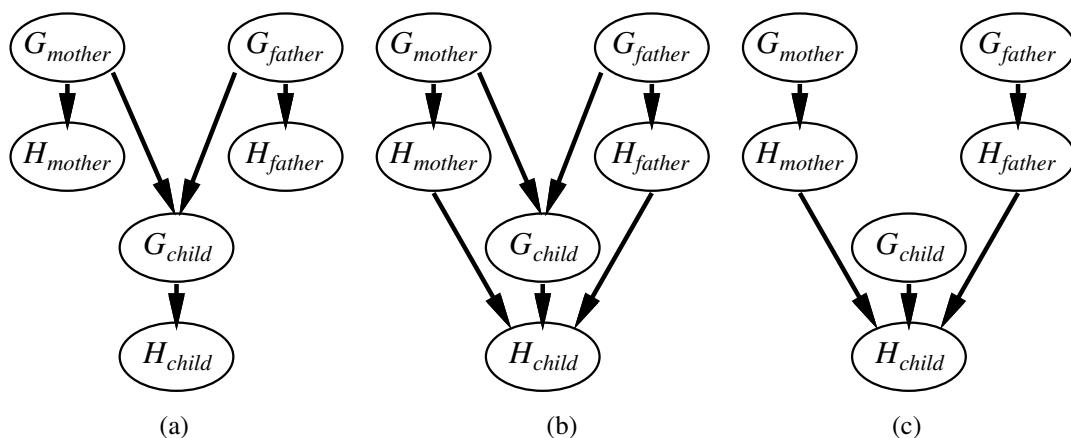


Figure S13.8 Figure for Exercise 13.HAND.

Exercise 13.2.#HAND

Let H_x be a random variable denoting the handedness of an individual x , with possible values l or r . A common hypothesis is that left- or right-handedness is inherited by a simple mechanism; that is, perhaps there is a gene G_x , also with values l or r , and perhaps actual handedness turns out mostly the same (with some probability s) as the gene an individual possesses. Furthermore, perhaps the gene itself is equally likely to be inherited from either of an individual's parents, with a small nonzero probability m of a random mutation flipping the handedness.

- Which of the three networks in Figure S13.8 claim that $\mathbf{P}(G_{father}, G_{mother}, G_{child}) = \mathbf{P}(G_{father})\mathbf{P}(G_{mother})\mathbf{P}(G_{child})$?
- Which of the three networks make independence claims that are consistent with the hypothesis about the inheritance of handedness?
- Which of the three networks is the best description of the hypothesis?
- Write down the CPT for the G_{child} node in network (a), in terms of s and m .

- e. Suppose that $P(G_{\text{father}} = l) = P(G_{\text{mother}} = l) = q$. In network (a), derive an expression for $P(G_{\text{child}} = l)$ in terms of m and q only, by conditioning on its parent nodes.
- f. Under conditions of genetic equilibrium, we expect the distribution of genes to be the same across generations. Use this to calculate the value of q , and, given what you know about handedness in humans, explain why the hypothesis described at the beginning of this question must be wrong.

- a. (c) matches the equation. The equation describes absolute independence of the three genes, which requires no links among them.
- b. (a) and (b). The *assertions* are the *absent* links; the extra links in (b) may be unnecessary but they do not assert an actual dependence. (c) asserts independence of genes which contradicts the inheritance scenario.
- c. (a) is best. (b) has spurious links among the H variables, which are not directly causally connected in the scenario described. (In reality, handedness may also be passed down by example/training.)
- d. Notice that the $l \rightarrow r$ and $r \rightarrow l$ mutations cancel when the parents have different genes, so we still get 0.5.

G_{mother}	G_{father}	$P(G_{\text{child}} = l \dots)$	$P(G_{\text{child}} = r \dots)$
l	l	$1 - m$	m
l	r	0.5	0.5
r	l	0.5	0.5
r	r	m	$1 - m$

- e. This is a straightforward application of conditioning:

$$\begin{aligned}
 P(G_{\text{child}} = l) &= \sum_{g_m, g_f} P(G_{\text{child}} = l | g_m, g_f) P(g_m, g_f) \\
 &= \sum_{g_m, g_f} P(G_{\text{child}} = l | g_m, g_f) P(g_m) P(g_f) \\
 &= (1 - m)q^2 + 0.5q(1 - q) + 0.5(1 - q)q + m(1 - q)^2 \\
 &= q^2 - mq^2 + q - q^2 + m - 2mq + mq^2 \\
 &= q + m - 2mq
 \end{aligned}$$

- f. Equilibrium means that $P(G_{\text{child}} = l)$ (the prior, with no parent information) must equal $P(G_{\text{mother}} = l)$ and $P(G_{\text{father}} = l)$, i.e.,

$$q + m - 2mq = q, \text{ hence } q = 0.5.$$

But few humans are left-handed ($x \approx 0.08$ in fact), so something is wrong with the symmetric model of inheritance and/or manifestation. The “high-school” explanation is that the “right-hand gene is dominant,” i.e., preferentially inherited, but current studies suggest also that handedness is not the result of a single gene and may also involve cultural factors. An entire journal (*Laterality*) is devoted to this topic.

Exercise 13.2.#MARB

The **Markov blanket** of a variable is defined on page 419. Prove that a variable is independent of all other variables in the network, given its Markov blanket and derive Equation (13.10) (page 443).

These proofs are tricky for those not accustomed to manipulating probability expressions, and students may require some hints.

- a. There are several ways to prove this. Probably the simplest is to work directly from the global semantics. First, we rewrite the required probability in terms of the full joint:

$$\begin{aligned} P(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) &= \frac{P(x_1, \dots, x_n)}{P(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)} \\ &= \frac{P(x_1, \dots, x_n)}{\sum_{x_i} P(x_1, \dots, x_n)} \\ &= \frac{\prod_{j=1}^n P(x_j|parents X_j)}{\sum_{x_i} \prod_{j=1}^n P(x_j|parents X_j)} \end{aligned}$$

Now, all terms in the product in the denominator that do not contain x_i can be moved outside the summation, and then cancel with the corresponding terms in the numerator. This just leaves us with the terms that do mention x_i , i.e., those in which X_i is a child or a parent. Hence, $P(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ is equal to

$$\frac{P(x_i|parents X_i) \prod_{Y_j \in Children(X_i)} P(y_j|parents(Y_j))}{\sum_{x_i} P(x_i|parents X_i) \prod_{Y_j \in Children(X_i)} P(y_j|parents(Y_j))}$$

Now, by reversing the argument in part (b), we obtain the desired result.

- b. This is a relatively straightforward application of Bayes' rule. Let $\mathbf{Y} = Y_1, \dots, y_\ell$ be the children of X_i and let \mathbf{Z}_j be the parents of Y_j other than X_i . Then we have

$$\begin{aligned} \mathbf{P}(X_i|MB(X_i)) &= \mathbf{P}(X_i|Parents(X_i), \mathbf{Y}, \mathbf{Z}_1, \dots, \mathbf{Z}_\ell) \\ &= \alpha \mathbf{P}(X_i|Parents(X_i), \mathbf{Z}_1, \dots, \mathbf{Z}_\ell) \mathbf{P}(\mathbf{Y}|Parents(X_i), X_i, \mathbf{Z}_1, \dots, \mathbf{Z}_\ell) \\ &= \alpha \mathbf{P}(X_i|Parents(X_i)) \mathbf{P}(\mathbf{Y}|X_i, \mathbf{Z}_1, \dots, \mathbf{Z}_\ell) \\ &= \alpha \mathbf{P}(X_i|Parents(X_i)) \prod_{Y_j \in Children(X_i)} P(Y_j|Parents(Y_j)) \end{aligned}$$

where the derivation of the third line from the second relies on the fact that a node is independent of its nondescendants given its parents.

Exercise 13.2.#DCAR

Consider the network for car diagnosis shown in Figure S13.9.

- a. Extend the network with the Boolean variables *IcyWeather* and *StarterMotor*.

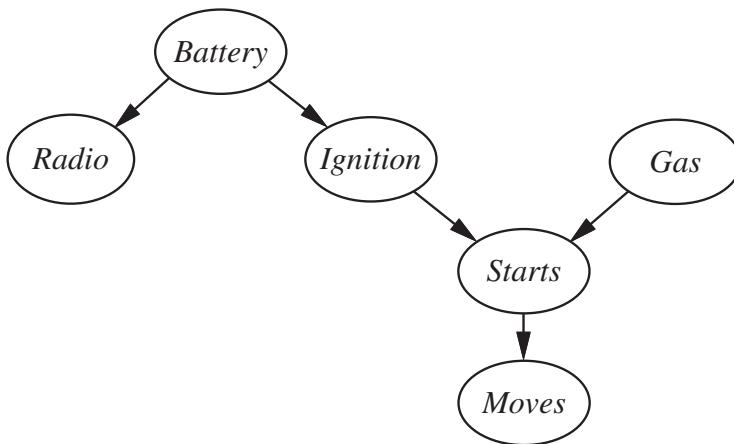


Figure S13.9 Car network for Exercise 13.DCAR.

- b. Give reasonable conditional probability tables for all the nodes.
- c. How many independent values are contained in the joint probability distribution for eight Boolean nodes, assuming that no conditional independence relations are known to hold among them?
- d. How many independent probability values do your network tables contain?
- e. The conditional distribution for *Starts* could be described as a **noisy-AND** distribution. Define this family in general and relate it to the noisy-OR distribution.

Adding variables to an existing net can be done in two ways. Formally speaking, one should insert the variables into the variable ordering and rerun the network construction process from the point where the first new variable appears. Informally speaking, one never really builds a network by a strict ordering. Instead, one asks what variables are direct causes or influences on what other ones, and builds local parent/child graphs that way. It is usually easy to identify where in such a structure the new variable goes, but one must be very careful to check for possible induced dependencies downstream.

- a. *IcyWeather* is not caused by any of the car-related variables, so needs no parents. It directly affects the battery and the starter motor. *StarterMotor* is an additional precondition for *Starts*. The new network is shown in Figure S13.10.
- b. Reasonable probabilities may vary a lot depending on the kind of car and perhaps the personal experience of the assessor. The following values indicate the general order of magnitude and relative values that make sense:
 - A reasonable prior for *IcyWeather* might be 0.05 (perhaps depending on location and season).
 - $P(\text{Battery}|\text{IcyWeather}) = 0.95, P(\text{Battery}|\neg\text{IcyWeather}) = 0.997$.
 - $P(\text{StarterMotor}|\text{IcyWeather}) = 0.98, P(\text{StarterMotor}|\neg\text{IcyWeather}) = 0.999$.

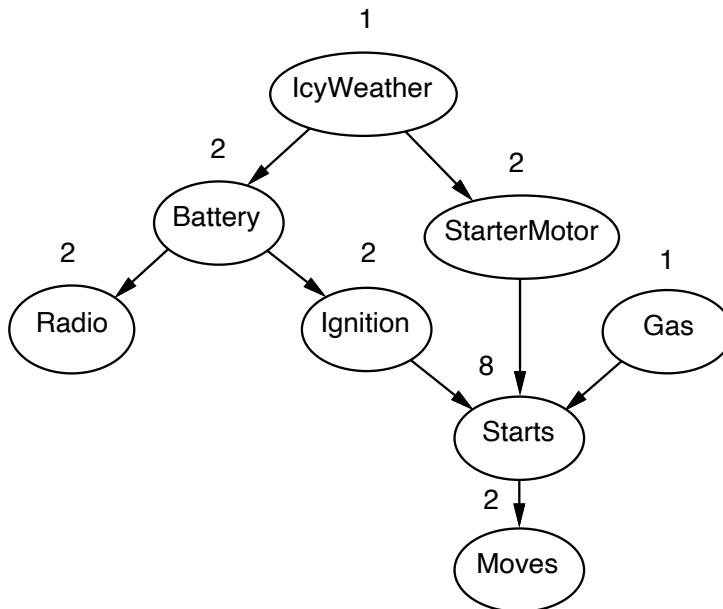


Figure S13.10 Car network amended to include *IcyWeather* and *StarterMotorWorking* (*SMW*).

- $P(\text{Radio}|\text{Battery}) = 0.9999, P(\text{Radio}|\neg\text{Battery}) = 0.05.$
 - $P(\text{Ignition}|\text{Battery}) = 0.998, P(\text{Ignition}|\neg\text{Battery}) = 0.01.$
 - $P(\text{Gas}) = 0.995.$
 - $P(\text{Starts}|\text{Ignition}, \text{StarterMotor}, \text{Gas}) = 0.9999, \text{ other entries } 0.0.$
 - $P(\text{Moves}|\text{Starts}) = 0.998.$
- c. With 8 Boolean variables, the joint has $2^8 - 1 = 255$ independent entries.
- d. Given the topology shown in Figure S13.10, the total number of independent CPT entries is $1+2+2+2+2+1+8+2= 20.$
- e. The CPT for *Starts* describes a set of nearly necessary conditions that are together almost sufficient. That is, all the entries are nearly zero except for the entry where all the conditions are true. That entry will be not quite 1 (because there is always some other possible fault that we didn't think of), but as we add more conditions it gets closer to 1. If we add a *Leak* node as an extra parent, then the probability is exactly 1 when all parents are true. We can relate noisy-AND to noisy-OR using de Morgan's rule: $A \wedge B \equiv \neg(\neg A \vee \neg B).$ That is, noisy-AND is the same as noisy-OR except that the polarities of the parent and child variables are reversed. In the noisy-OR case, we have

$$P(Y = \text{true}|x_1, \dots, x_k) = 1 - \prod_{\{i:x_i = \text{true}\}} q_i$$

where q_i is the probability that the *presence* of the i th parent *fails* to cause the child to

be *true*. In the noisy-AND case, we can write

$$P(Y = \text{true} | x_1, \dots, x_k) = \prod_{\{i: x_i = \text{false}\}} r_i$$

where r_i is the probability that the *absence* of the i th parent *fails* to cause the child to be *false* (e.g., it is magically bypassed by some other mechanism).

Exercise 13.2.#FLUN

Consider a simple Bayesian network with root variables *Cold*, *Flu*, and *Malaria* and child variable *Fever*, with a noisy-OR conditional distribution for *Fever* as described in Section 13.2.2. By adding appropriate auxiliary variables for inhibition events and fever-inducing events, construct an equivalent Bayesian network whose CPTs (except for root variables) are deterministic. Define the CPTs and prove equivalence.

Recall that the noisy-OR CPT for a Boolean variable X with Boolean parents U_1, \dots, U_k is given by

$$P(X = \text{true} | u_1, \dots, u_k) = 1 - \prod_{\{j: u_j = \text{true}\}} q_j .$$

We would like to construct a deterministic CPT for X with additional parent variables \mathbf{Z} , such that the original conditional distribution is recovered when those variables are summed out:

$$\sum_{\mathbf{z}} P(X = \text{true} | u_1, \dots, u_k, \mathbf{z}) = 1 - \prod_{\{j: u_j = \text{true}\}} q_j .$$

Let Z_j be a Boolean inhibitory variable for parent U_j ; it will be a root variable with prior probability q_j . The basic idea is that the effect occurs if any cause occurs and the cause is not inhibited; equivalently, the effect does not occur if all the occurring causes are inhibited. So the required CPT implements the following deterministic relationship:

$$X = (U_1 \wedge \neg Z_1) \vee \dots \vee (U_k \wedge \neg Z_k) .$$

Since the inhibition variables are independent, the probability of all occurring causes being inhibited is exactly $\prod_{\{j: u_j = \text{true}\}} q_j$, as required.

Exercise 13.2.#LGEX

Consider the family of linear Gaussian networks, as defined on page 422.

- a. In a two-variable network, let X_1 be the parent of X_2 , let X_1 have a Gaussian prior, and let $\mathbf{P}(X_2 | X_1)$ be a linear Gaussian distribution. Show that the joint distribution $P(X_1, X_2)$ is a multivariate Gaussian, and calculate its covariance matrix.

- b.** Prove by induction that the joint distribution for a general linear Gaussian network on X_1, \dots, X_n is also a multivariate Gaussian.

This exercise is a little tricky and will appeal to more mathematically oriented students.

- a.** The basic idea is to multiply the two densities, match the result to the standard form for a multivariate Gaussian, and hence identify the entries in the inverse covariance matrix. Let's begin by looking at the multivariate Gaussian. From page 982 in Appendix A we have

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}((\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu))},$$

where μ is the mean vector and Σ is the covariance matrix. In our case, \mathbf{x} is $(x_1 \ x_2)^\top$ and let the (as yet) unknown μ be $(m_1 \ m_2)^\top$. Suppose the inverse covariance matrix is

$$\Sigma^{-1} = \begin{pmatrix} c & d \\ d & e \end{pmatrix}$$

Then, if we multiply out the exponent, we obtain

$$\begin{aligned} -\frac{1}{2} ((\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)) &= \\ -\frac{1}{2} \cdot c(x_1 - m_1)^2 + 2d(x_1 - m_1)(x_2 - m_2) + f(x_2 - m_2)^2 & \end{aligned}$$

Looking at the distributions themselves, we have

$$P(x_1) = \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-(x_1 - \mu_1)^2 / (2\sigma_1^2)}$$

and

$$P(x_2|x_1) = \frac{1}{\sigma_2 \sqrt{2\pi}} e^{-(x_2 - (ax_1 + b))^2 / (2\sigma_2^2)}$$

hence

$$P(x_1, x_2) = \frac{1}{\sigma_1 \sigma_2 (2\pi)} e^{-(\sigma_2^2 (x_2 - (ax_1 + b))^2 + \sigma_1^2 (x_2 - (ax_1 + b))^2) / (2\sigma_1^2 \sigma_2^2)}$$

We can obtain equations for c , d , and e by picking out the coefficients of x_1^2 , $x_1 x_2$, and x_2^2 :

$$c = (\sigma_2^2 + a^2 \sigma_1^2) / \sigma_1^2 \sigma_2^2$$

$$2d = -2a / \sigma_2^2$$

$$e = 1 / \sigma_2^2$$

We can check these by comparing the normalizing constants.

$$\frac{1}{\sigma_1 \sigma_2 (2\pi)} = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} = \frac{1}{(2\pi) \sqrt{1/|\Sigma^{-1}|}} = \frac{1}{(2\pi) \sqrt{1/(ce - d^2)}}$$

from which we obtain the constraint

$$ce - d^2 = 1/\sigma_1^2 \sigma_2^2$$

which is easily confirmed. Similar calculations yield m_1 and m_2 , and plugging the results back shows that $P(x_1, x_2)$ is indeed multivariate Gaussian. The covariance matrix is

$$\Sigma = \begin{pmatrix} c & d \\ d & e \end{pmatrix}^{-1} = \frac{1}{ce - d^2} \begin{pmatrix} e & -d \\ -d & c \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & a\sigma_1^2 \\ a\sigma_1^2 & \sigma_2^2 + a^2\sigma_1^2 \end{pmatrix}$$

- b.** The induction is on n , the number of variables. The base case for $n=1$ is trivial. The inductive step asks us to show that if any $P(x_1, \dots, x_n)$ constructed with linear-Gaussian conditional densities is multivariate Gaussian, then any $P(x_1, \dots, x_n, x_{n+1})$ constructed with linear-Gaussian conditional densities is also multivariate Gaussian. Without loss of generality, we can assume that X_{n+1} is a leaf variable added to a network defined in the first n variables. By the product rule we have

$$\begin{aligned} P(x_1, \dots, x_n, x_{n+1}) &= P(x_{n+1}|x_1, \dots, x_n)P(x_1, \dots, x_n) \\ &= P(x_{n+1}|\text{parents}(X_{n+1}))P(x_1, \dots, x_n) \end{aligned}$$

which, by the inductive hypothesis, is the product of a linear Gaussian with a multivariate Gaussian. Extending the argument of part (a), this is in turn a multivariate Gaussian of one higher dimension.

Exercise 13.2.#PROD

The probit distribution defined on page 424 describes the probability distribution for a Boolean child, given a single continuous parent.

- a.** How might the definition be extended to cover multiple continuous parents?
 - b.** How might it be extended to handle a *multivalued* child variable? Consider both cases where the child's values are ordered (as in selecting a gear while driving, depending on speed, slope, desired acceleration, etc.) and cases where they are unordered (as in selecting bus, train, or car to get to work). (*Hint:* Consider ways to divide the possible values into two sets, to mimic a Boolean variable.)
-
- a.** With multiple continuous parents, we must find a way to map the parent value vector to a single threshold value. The simplest way to do this is to take a linear combination of the parent values.
 - b.** For ordered values $y_1 < y_2 < \dots < y_d$, we assume some unobserved continuous dependent variable y^* that is normally distributed conditioned on the parent variables,

and define cutpoints c_j such that $Y = y_j$ iff $c_{j-1} \leq y^* \leq c_j$. The probability of this event is given by subtracting the cumulative distributions at the adjacent cutpoints.

The unordered case is not obviously meaningful if we insist that the relationship between parents and child be mediated by a single, real-valued, normally distributed variable.

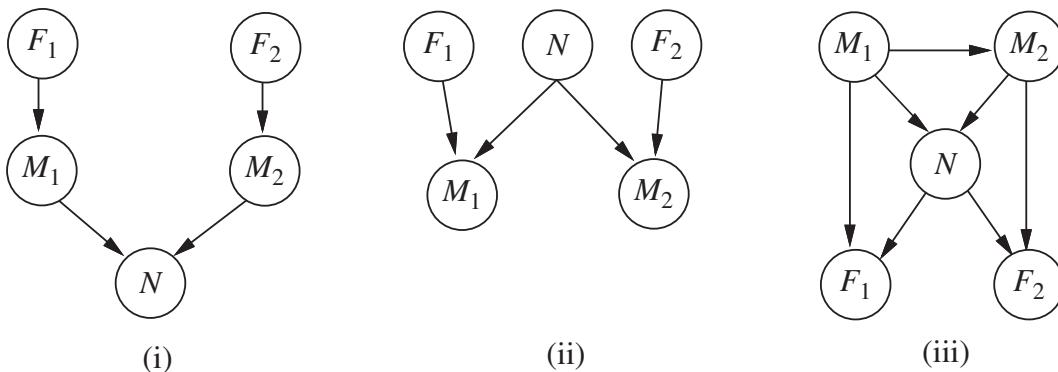


Figure S13.11 A Bayesian network for Exercise 13.TELC.

Exercise 13.2.#TELE

Two astronomers in different parts of the world make measurements M_1 and M_2 of the number of stars N in some small region of the sky, using their telescopes. Normally, there is a small possibility e of error by up to one star in each direction. Each telescope can also (with a much smaller probability f) be badly out of focus (events F_1 and F_2), in which case the scientist will undercount by three or more stars (or if N is less than 3, fail to detect any stars at all). Consider the three networks shown in Figure S13.11.

- Which of these Bayesian networks are correct (but not necessarily efficient) representations of the preceding information?
- Which is the best network? Explain.
- Write out a conditional distribution for $\mathbf{P}(M_1 | N)$, for the case where $N \in \{1, 2, 3\}$ and $M_1 \in \{0, 1, 2, 3, 4\}$. Each entry in the conditional distribution should be expressed as a function of the parameters e and/or f .
- Suppose $M_1 = 1$ and $M_2 = 3$. What are the *possible* numbers of stars if you assume no prior constraint on the values of N ?
- What is the *most likely* number of stars, given these observations? Explain how to compute this, or if it is not possible to compute, explain what additional information is needed and how it would affect the result.
- Although (i) in some sense depicts the “flow of information” during calculation, it is clearly incorrect as a network, since it says that given the measurements M_1 and M_2 ,

the number of stars is independent of the focus. (ii) correctly represents the causal structure: each measurement is influenced by the actual number of stars and the focus, and the two telescopes are independent of each other. (iii) shows a correct but more complicated network—the one obtained by ordering the nodes M_1, M_2, N, F_1, F_2 . If you order M_2 before M_1 you would get the same network except with the arrow from M_1 to M_2 reversed.

- b.** (ii) requires fewer parameters and is therefore better than (iii).
- c.** To compute $\mathbf{P}(M_1|N)$, we will need to condition on F_1 (that is, consider both possible cases for F_1 , weighted by their probabilities).

$$\begin{aligned}\mathbf{P}(M_1|N) &= \mathbf{P}(M_1|N, F_1)\mathbf{P}(F_1|N) + \mathbf{P}(M_1|N, \neg F_1)\mathbf{P}(\neg F_1|N) \\ &= \mathbf{P}(M_1|N, F_1)\mathbf{P}(F_1) + \mathbf{P}(M_1|N, \neg F_1)\mathbf{P}(\neg F_1)\end{aligned}$$

Let f be the probability that the telescope is out of focus. The exercise states that this will cause an “undercount of three or more stars,” but if $N = 3$ or less the count will be 0 if the telescope is out of focus. If it is in focus, then we will assume there is a probability of e of counting one too few, and e of counting one too many. The rest of the time ($1 - 2e$), the count will be accurate. Then the table is as follows:

	$N = 1$	$N = 2$	$N = 3$
$M_1 = 0$	$f + e(1-f)$	f	f
$M_1 = 1$	$(1-2e)(1-f)$	$e(1-f)$	0.0
$M_1 = 2$	$e(1-f)$	$(1-2e)(1-f)$	$e(1-f)$
$M_1 = 3$	0.0	$e(1-f)$	$(1-2e)(1-f)$
$M_1 = 4$	0.0	0.0	$e(1-f)$

Notice that each column has to add up to 1. Reasonable values for e and f might be 0.05 and 0.002.

- d.** This question causes a surprising amount of difficulty, so it is important to make sure students understand the reasoning behind an answer. One approach uses the fact that it is easy to reason in the forward direction, that is, try each possible number of stars N and see whether measurements $M_1 = 1$ and $M_2 = 3$ are possible. (This is a sort of mental simulation of the physical process.) An alternative approach is to enumerate the possible focus states and deduce the value of N for each. Either way, the solutions are $N = 2, 4$, or ≥ 6 .
- e.** We cannot calculate the most likely number of stars without knowing the prior distribution $P(N)$. Let the priors be p_2, p_4 , and $p_{\geq 6}$. The posterior for $N = 2$ is $p_2 e^2 (1 - f)^2$; for $N = 4$ it is at most $p_4 e f$ (at most, because with $N = 4$ the out-of-focus telescope could measure 0 instead of 1); for $N \geq 6$ it is at most $p_{\geq 6} f^2$. If we assume that the priors are roughly comparable, then $N = 2$ is most likely because we are told that f is much smaller than e .

For follow-up or alternate questions, it is easy to come up with endless variations on the same theme involving sensors, failure nodes, hidden state. One can also add in complex mechanisms, as for the *Starts* variable in Exercise 13.DCAR.

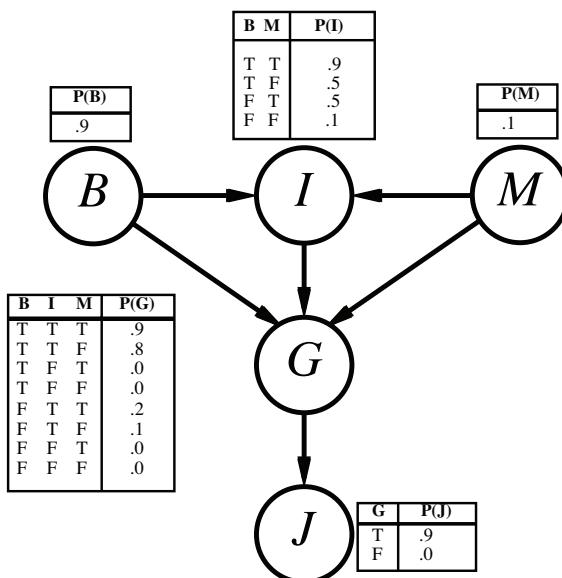


Figure S13.12 A Bayesian network for Exercise 13.BATF.

Exercise 13.2.#BATF

Consider the Bayes net shown in Figure S13.12.

- Which of the following are asserted by the network *structure*?
 - $\mathbf{P}(B, I, M) = \mathbf{P}(B)\mathbf{P}(I)\mathbf{P}(M)$.
 - $\mathbf{P}(J|G) = \mathbf{P}(J|G, I)$.
 - $\mathbf{P}(M|G, B, I) = \mathbf{P}(M|G, B, I, J)$.
- Calculate the value of $P(b, i, m, \neg g, j)$.
- Calculate the value of $P(b, i, \neg m, g, j)$.
- Calculate the probability that someone goes to jail given that they broke the law, have been indicted, and face a politically motivated prosecutor.
- A **context-specific independence** (see page 420) allows a variable to be independent of some of its parents given certain values of others. In addition to the usual conditional independences given by the graph structure, what context-specific independences exist in the Bayes net in Figure S13.12?
- Suppose we want to add the variable $P = \text{PresidentialPardon}$ to the network; draw the new network and briefly explain any links you add.

- The network asserts (ii) and (iii). (For (iii), consider the Markov blanket of M .)

$$\begin{aligned} \mathbf{P}(b, i, m, \neg g, j) &= P(b)P(m)P(i|b, m)P(\neg g|b, i, m)P(j|\neg g) \\ &= .9 \times .1 \times .9 \times .1 \times 0 = 0. \end{aligned}$$

It would be reasonable to avoid the unnecessary work and point out that the model

precludes going to jail if found not guilty so the event must have probability 0.

- c. $P(b, i, \neg m, g, j) = P(b)P(\neg m)P(i|b, \neg m)P(g|b, i, \neg m)P(j|g)$
 $= .9 \times .9 \times .5 \times .8 \times .9 = .2916$
- d. Since B, I, M are fixed true in the evidence, we can treat G as having a prior of 0.9 and just look at the submodel with G, J :

$$\begin{aligned}\mathbf{P}(J|b, i, m) &= \alpha \sum_g \mathbf{P}(J, g) = \alpha [\mathbf{P}(J, g) + \mathbf{P}(J, \neg g)] \\ &= \alpha [\langle P(j, g), P(\neg j, g) \rangle + \langle P(j, \neg g), P(\neg j, \neg g) \rangle] \\ &= \alpha [(.81, .09) + \langle 0, 0.1 \rangle] = (.81, .19)\end{aligned}$$

That is, the probability of going to jail is 0.81.
- e. Intuitively, a person cannot be found guilty if not indicted, regardless of whether they broke the law and regardless of the prosecutor. This is what the CPT for G says; so G is context-specifically independent of B and M given $I = \text{false}$.
- f. A pardon is unnecessary if the person is not indicted or not found guilty; so I and G are parents of P . One could also add B and M as parents of P , since a pardon is more likely if the person is actually innocent and if the prosecutor is politically motivated. (There are other causes of *Pardon*, such as *LargeDonationToPresidentsParty*, but such variables are not currently in the model.) The pardon (presumably) is a get-out-of-jail-free card, so P is a parent of J .

Exercise 13.2.#BNTF

Which of the following are true, and which are false?

- a. Bayes nets are organized into layers with connections only between adjacent layers.
- b. The topology of a Bayes net can assert that some variables are *not* conditionally independent.
- c. Some Bayes nets require as many parameters as the explicitly tabulated full joint distribution.
- d. In any Bayes net, the parents of a single child are always conditionally independent of each other given the child.

- a. False. Any connections are allowed as long as the network remains acyclic.
- b. False. The topology asserts conditional *independence* relationships, but cannot assert *dependence*. Even if a link exists, the associated CPT can still deny that there is any dependence.
- c. True. This is the case when every variable has all of its predecessors as parents.
- d. False. The parents will usually be conditionally *dependent* given the child, as one cause explains away another.

Exercise 13.2.#BNCI

In the Bayes net in Figure S13.13, state whether each of the following assertions is necessarily true, necessarily false, or undetermined.

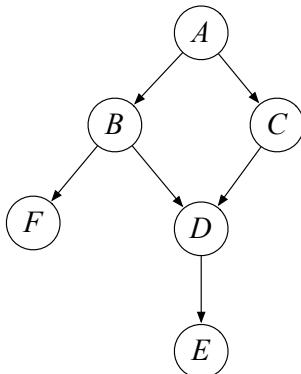


Figure S13.13 A Bayesian network for Exercise 13.BNCI.

- a. A is absolutely independent of E .
- b. B is conditionally independent of C given A .
- c. F is conditionally independent of C given A .
- d. B is conditionally independent of C given A and E .

Remember that the topology of a Bayes net cannot assert that a given independence does not hold.

- a. Undetermined.
- b. True.
- c. True.
- d. Undetermined.

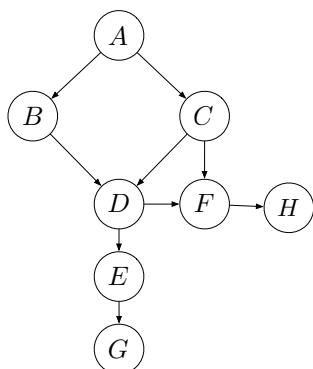


Figure S13.14 A Bayesian network for Exercise 13.CIBN.

Exercise 13.2.#CIBN

In the Bayes net in Figure S13.14, state whether each of the following assertions is necessarily true, necessarily false, or undetermined.

- a. B is absolutely independent of C .
- b. B is conditionally independent of C given G .
- c. B is conditionally independent of C given H .
- d. A is conditionally independent of D given G .
- e. A is conditionally independent of D given H .
- f. B is conditionally independent of C given A, F .
- g. F is conditionally independent of B given D, A .
- h. F is conditionally independent of B given D, C .

Remember that the topology of a Bayes net cannot assert that a given independence does not hold.

- a. Undetermined.
- b. Undetermined.
- c. Undetermined.
- d. Undetermined.
- e. Undetermined.
- f. Undetermined.
- g. Undetermined.
- h. True.

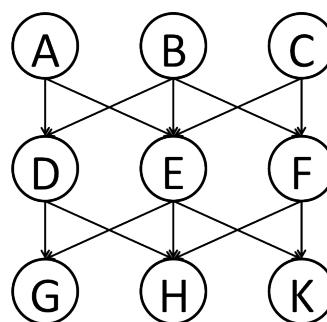


Figure S13.15 A Bayesian network for Exercise 13.TFBN.

Exercise 13.2.#TFBN

In the Bayes net in Figure S13.15, state whether each of the following assertions is necessarily true, necessarily false, or undetermined.

- a. A is absolutely independent of C .
- b. A is conditionally independent of C given E .
- c. A is conditionally independent of C given G .
- d. A is absolutely independent of K .
- e. A is conditionally independent of G given D, E, F .
- f. A is conditionally independent of B given D, E, F .
- g. A is conditionally independent of C given D, F, K .
- h. A is conditionally independent of G given D .

Remember that the topology of a Bayes net cannot assert that a given independence does not hold.

- a. True.
- b. Undetermined.
- c. Undetermined.
- d. Undetermined.
- e. True.
- f. Undetermined.
- g. Undetermined.
- h. Undetermined.

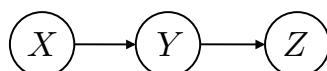


Figure S13.16 A Bayesian network for Exercise 13.BNNT.

Exercise 13.2.#BNNT

In the Bayes net in Figure S13.16, which of the following are *necessarily* true?

- a. $P(X, Y, Z) = P(X)P(Y|X)P(Z|X, Y)$.
- b. $P(X, Y, Z) = P(X)P(Y|X)P(Z|Y)$.
- c. $P(X, Y, Z) = P(X)P(Y|Z)P(Z|X)$.
- d. $P(X, Y, Z) = P(Z)P(Y|Z)P(X|Y, Z)$.
- e. $P(X, Y, Z) = P(Z)P(Y|Z)P(X|Y)$.

- a. True.
- b. True.
- c. False.
- d. True.
- e. True.

Exercise 13.2.#PRRV

Let P be a probability distribution over random variables A, B, C . Let Q be another probability distribution over the same variables, defined by a Bayes net in which B and C are conditionally independent given A . We are given that $Q(A) = P(A)$, $Q(B|A) = P(B|A)$, and $Q(C|A) = P(C|A)$. Which if the following can be deduced? Explain.

- a. $Q(B) = P(B)$.
- b. $Q(C) = P(C)$.
- c. $Q(A, B, C) = P(A, B, C)$.

- a. True. $Q(B) = \sum_a Q(B|a)Q(a) = \sum_a P(B|a)P(a) = P(B)$.
- b. True, by the same argument.
- c. False. In P , it is not necessarily the case that B and C are conditionally independent given A . We have

$$P(A, B, C) = P(A)P(B|A)P(C|A, B) = Q(A)Q(B|A)P(C|A, B)$$

and

$$Q(A, B, C) = Q(A)Q(B|A)Q(C|A) = Q(A)Q(B|A)P(C|A)$$

but we cannot establish that $P(C|A, B) = P(C|A)$.

Exercise 13.2.#CDBV

You are given the following conditional distributions that connect the binary variables W, X, Y, Z :

X	W	$P(W X)$	X	Y	$P(Y X)$	Z	W	$P(W Z)$
0	0	0.4	0	0	0.3	0	0	0.2
0	1	0.6	0	1	0.7	0	1	0.8
1	0	0.4	1	0	0.1	1	0	0.8
1	1	0.6	1	1	0.9	1	1	0.2

Which of the Bayes nets in Figure S13.17 can represent a joint distribution that is consistent with these conditional distributions? Of those, which are minimal, in the sense that no edge can be removed while retaining the property?

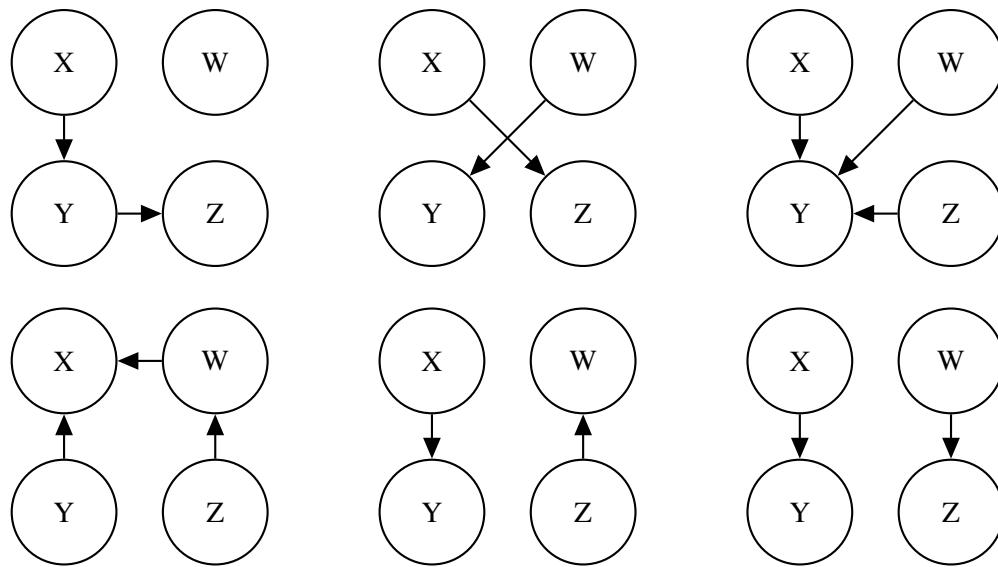


Figure S13.17 Bayes nets for Exercise 13.CDBV

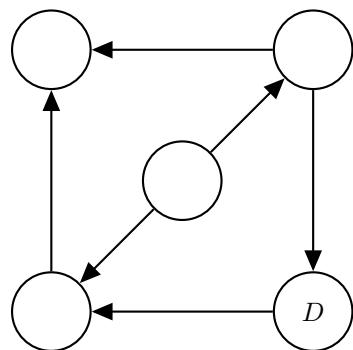


Figure S13.18 Bayes net for Exercise 13.ABCE.

Remember that the topology of a Bayes net makes claims of conditional independence through the absence of arrows. The first three networks assert that W and Z are absolutely independent, which is not the case. Examining the distribution $P(W|X)$, we see that the probability of W does not actually depend on X , so W and X are absolutely independent. The only dependencies that are required are between X and Y and between W and Z . The three networks in the second row capture these dependencies. (The direction of the arrow doesn't matter.) The fourth network has a superfluous arrow from W to X , so the fifth and sixth networks are minimal.

Exercise 13.2.#ABCE

Label the blank nodes in the Bayes net below with the variables $\{A, B, C, E\}$ such that the following independence assertions are true:

- A is conditionally independent of B given D, E ;
- E is conditionally independent of D given B ;
- E is conditionally independent of C given A, B ;
- C is conditionally independent of D given A, B ;

There is only one solution, as follows:

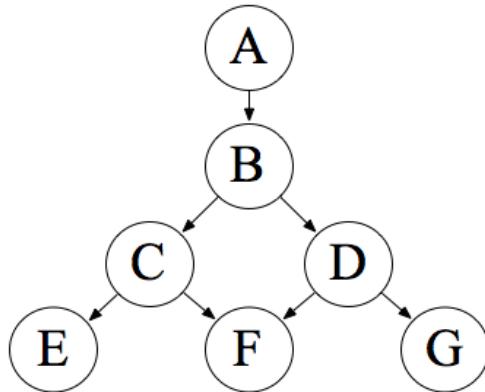
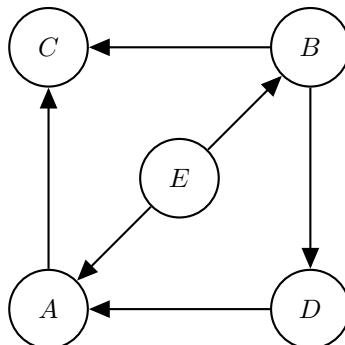


Figure S13.19 Figure for Exercise 13.DSEP.

Exercise 13.2.#DSEP

- Consider the Bayes net in Figure S13.19.
 - Given B , what variable(s) is E guaranteed to be independent of?
 - Given B and F , what variable(s) is G guaranteed to be independent of?

- b. Now we'd like to formulate d-separation as a search problem. Specifically, you're given a variable X and a variable Y, a Bayes net G, and a set of observed variables E. You're also given E^+ , which is the set of variables that are the parents or ancestors of evidence variables. Given this information, define a search problem that finds Y if X and Y are not d-separated, and does not find a goal otherwise. You may find the notation $W \rightarrow U \in G$ meaning "an arc from W to U is in the Bayes net" helpful. A full credit solution will have a minimal state space.
- c. Give a non-trivial consistent heuristic for this problem.
- a.
- (i) Given B, E is guaranteed to be independent of A, D, G.
 - (ii) Given B and F, G is guaranteed to be independent of A.
- b.
- **State Space:** (n, d) where n is a node and $d \in \{\text{in}, \text{out}, \text{either}\}$.
 - **Initial State:** (X, either) .
 - **Successor Function:** The successor function extends the current state using active triples (from the Bayes' ball algorithm). Every node and edge direction that can be appended to the current state to yield a valid active triple in the graph is returned. The direction 'either' can be interpreted as either 'in' or 'out' for extension using an active triple.
 - **Goal Test:** Is equal to (Y, in) or (Y, out) .
- c. The length of the shortest path from the node n in the current state to Y, where edge direction is ignored.

I	$P(I)$	R	$P(R)$	M	$P(M I,R)$
+i	0.8	+r	0.4	+m	0
-i	0.2	-r	0.6	-m	1.0
				+m	0
				-m	1.0
				+r	0.7
				-r	0.3
				+i	0.2
				-i	0.8

```

graph TD
    I((I)) --> M((M))
    R((R)) --> M
  
```

Figure S13.20 Figure for Exercise 13.MUMP.

Exercise 13.2.#MUMP

There has been an outbreak of mumps in your college. You feel fine, but you're worried that you might already be infected. You decide to use Bayes nets to analyze the probability that you've contracted the mumps.

You first think about the following two factors:

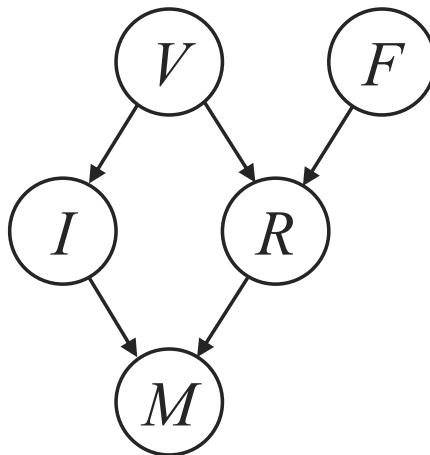


Figure S13.21 Figure for Exercise 13.MUMP.

- You think you have immunity from the mumps ($+i$) due to being vaccinated recently, but the vaccine is not completely effective, so you might not be immune ($-i$).
- Your roommate didn't feel well yesterday, and though you aren't sure yet, you suspect they might have the mumps ($+r$).

Denote these random variables by I and R . Let the random variable M take the value $+m$ if you have the mumps, and $-m$ if you do not. You write down the Bayes net in Figure S13.20 to describe your chances of being sick:

- Fill in the following table with the joint distribution over I , M , and R , $P(I, M, R)$.

I	R	M	$P(I, R, M)$
$+i$	$+r$	$+m$	0
$+i$	$+r$	$-m$	
$+i$	$-r$	$+m$	0
$+i$	$-r$	$-m$	
$-i$	$+r$	$+m$	0.056
$-i$	$+r$	$-m$	0.024
$-i$	$-r$	$+m$	0.024
$-i$	$-r$	$-m$	0.096

- What is the marginal probability $P(+m)$ that you have the mumps?
- Assuming you do have the mumps, you're concerned that your roommate may have the disease as well. What is the probability $P(+r | +m)$ that your roommate has the mumps given that you have the mumps? Note that you still don't know whether or not you have immunity.

You're still not sure if you have enough information about your chances of having the

mumps, so you decide to include two new variables in the Bayes net. Your roommate went to a party over the weekend, and there's some chance another person at the party had the mumps ($+f$). Furthermore, both you and your roommate were vaccinated at a clinic that reported a vaccine mix-up. Whether or not you got the right vaccine ($+v$ or $-v$) has ramifications for both your immunity (I) and the probability that your roommate has since contracted the disease (R). Accounting for these, you draw the modified Bayes net shown in Figure S13.21:

- d. Which of the following statements are *guaranteed* to be true for this Bayes net?

- (i) $V \perp\!\!\!\perp M | I, R$
- (ii) $V \perp\!\!\!\perp M | R$
- (iii) $M \perp\!\!\!\perp F | R$
- (iv) $V \perp\!\!\!\perp F$
- (v) $V \perp\!\!\!\perp F | M$
- (vi) $V \perp\!\!\!\perp F | I$

I	R	M	$P(I, R, M)$
$+i$	$+r$	$+m$	0
$+i$	$+r$	$-m$	0.32
$+i$	$-r$	$+m$	0
a. $+i$	$-r$	$-m$	0.48
$-i$	$+r$	$+m$	0.056
$-i$	$+r$	$-m$	0.024
$-i$	$-r$	$+m$	0.024
$-i$	$-r$	$-m$	0.096

b.

$$\begin{aligned}
 P(+m) &= \sum_{i,r} P(i, r, +m) \\
 &= P(+i, +r, +m) + P(+i, -r, +m) + P(-i, +r, +m) + P(-i, -r, +m) \\
 &= 0 + 0 + 0.056 + 0.024 = 0.08 = \frac{8}{100}
 \end{aligned}$$

c.

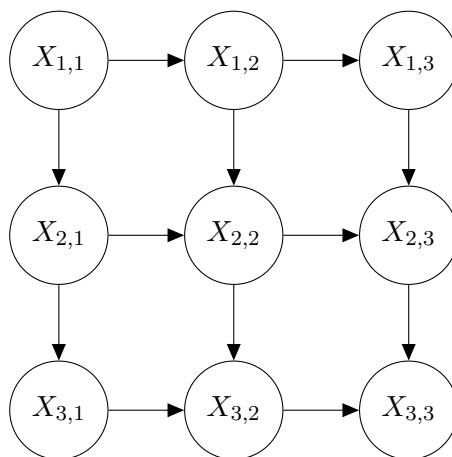
$$P(+r | +m) = \frac{P(+r, +m)}{P(+m)} = \frac{\sum_i P(i, +r, +m)}{P(+m)} = \frac{0 + 0.056}{0.080} = 0.143 = \frac{7}{10}$$

- d. Assertions (i), (iv), and (vi) are guaranteed to be true.

Exercise 13.2.#NINE

Consider the Bayes net below in Figure S13.22 with 9 variables:

- a. Which random variables are independent of $X_{3,1}$?

**Figure S13.22** Caption

- b. Which random variables are conditionally independent of $X_{3,1}$ given $X_{1,1}$?
 c. Which random variables are conditionally independent of $X_{3,1}$ given $X_{1,1}$ and $X_{3,3}$?

- a. None—there is at least one active path between $X_{3,1}$ and every other node.
 b. $X_{1,2}$ and $X_{1,3}$. $X_{1,1}$ blocks the only active paths to both $X_{1,2}$ and $X_{1,3}$, so both of those become independent of $X_{3,1}$ given $X_{1,1}$.
 c. None. The path from a node down to $X_{3,3}$ and up to another node is an active path.
 [[check]]

Exercise 13.2.#EDGD

For the Bayes net structures in Figure S?? and Figure S?? that are missing a direction on their edges, assign a direction to each edge such that the Bayes net structure implies the stated conditional independences and does not imply the conditional independences stated not to hold.

a. **Constraints:**

- $D \perp\!\!\!\perp G$
- not $D \perp\!\!\!\perp A$
- $D \perp\!\!\!\perp E$
- $H \perp\!\!\!\perp F$

b. **Constraints:**

- $D \perp\!\!\!\perp F$
- not $D \perp\!\!\!\perp G$

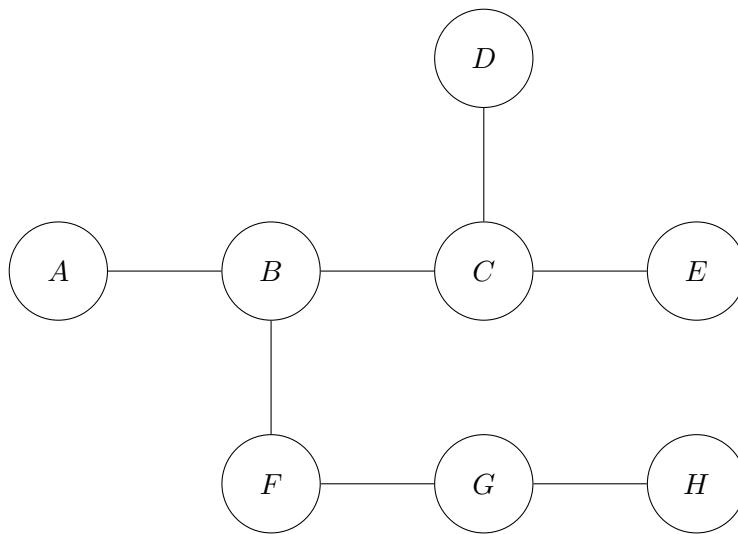


Figure S13.23 Figure (a) for Exercise 13.EDGD.

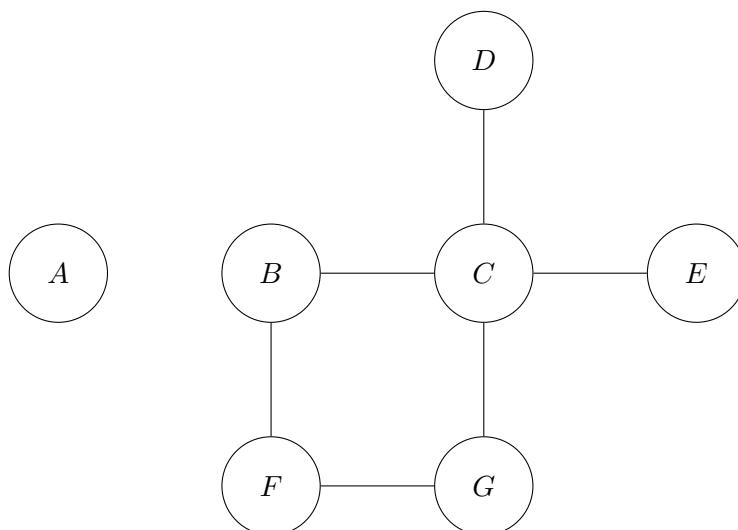


Figure S13.24 Figure (b) for Exercise 13.EDGD.

- $D \perp\!\!\!\perp E$
- Bayes net has no directed cycles

- a. The following edge directions are required:

$$B \rightarrow A$$

$$C \rightarrow B$$

$$D \rightarrow C$$

$$E \rightarrow C$$

$$F \rightarrow B$$

$$F \rightarrow G$$

$$H \rightarrow G$$

- b. The following edge directions are required:

$$C \rightarrow B$$

$$F \rightarrow B$$

$$F \rightarrow G$$

$$C \rightarrow G$$

$$D \rightarrow C$$

$$E \rightarrow C$$

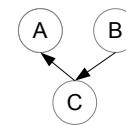
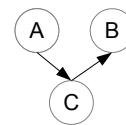
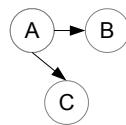
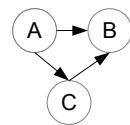
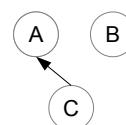
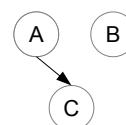
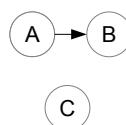
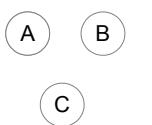


Figure S13.25 A Bayesian network for Exercise 13.PABC.

Exercise 13.2.#PABC

In which of the Bayes nets in Figure S13.25 does the equation $P(A, B)P(C) = P(A)P(B, C)$ necessarily hold?

It holds in the first, third, and fourth networks. We can rewrite the equation as follows:

$$P(A, B)P(C) = P(A)P(B, C)$$

$$\Rightarrow P(A, B)/P(A) = P(B, C)/P(C)$$

$$\Rightarrow P(B | A) = P(B | C).$$

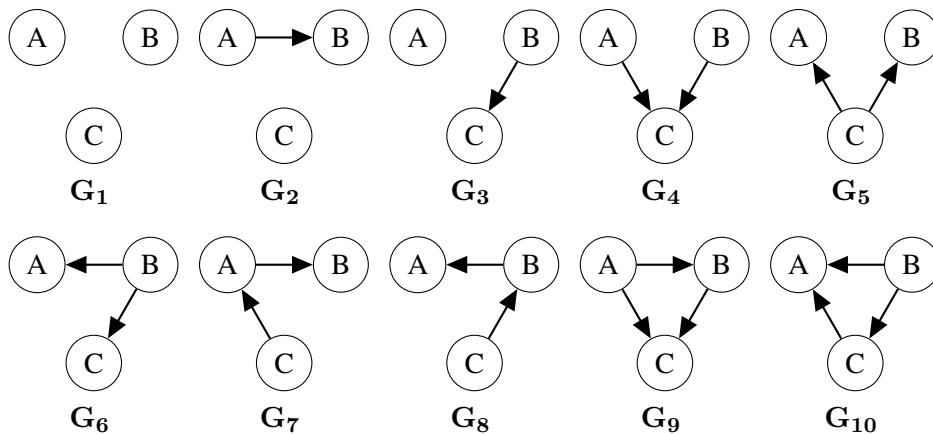


Figure S13.26 Ten Bayes nets for Exercise 13.GTEN.

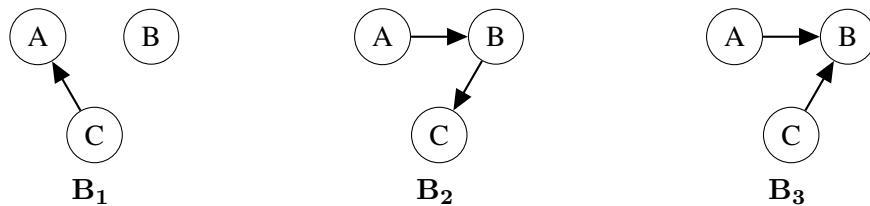


Figure S13.27 Three Bayes nets for Exercise 13.GTEN.

The only way this can *necessarily* hold (i.e., without numerical coincidences) is if B is absolutely independent of A and C .

Exercise 13.2.#GTEN

Assume we are given the ten Bayes nets in Figure S13.26, labeled G_1 to G_{10} .

Assume we are also given the three Bayes nets in Figure S13.27, labeled B_1 to B_3 .

- Assume we know that a joint distribution d_1 (over A, B, C) can be represented by Bayes net B_1 . Which of G_1 through G_{10} are guaranteed to be able to represent d_1 ?
- Assume we know that a joint distribution d_2 (over A, B, C) can be represented by Bayes net B_2 . Which of G_1 through G_{10} are guaranteed to be able to represent d_2 ?
- Assume we know that a joint distribution d_3 (over A, B, C) *cannot* be represented by Bayes net B_3 . Which of G_1 through G_{10} are guaranteed to be able to represent d_3 ?
- Assume we know that a joint distribution d_4 (over A, B, C) can be represented by Bayes nets B_1 , B_2 , and B_3 . Which of G_1 through G_{10} are guaranteed to be able to represent d_4 ?

It helps to enumerate all of the (conditional) independence assumptions encoded in the Bayes nets. They are:

- $\mathbf{G}_1: A \perp\!\!\!\perp B; A \perp\!\!\!\perp B | C; A \perp\!\!\!\perp C; A \perp\!\!\!\perp C | B; B \perp\!\!\!\perp C; B \perp\!\!\!\perp C | A$
- $\mathbf{G}_2: A \perp\!\!\!\perp C; A \perp\!\!\!\perp C | B; B \perp\!\!\!\perp C; B \perp\!\!\!\perp C | A$
- $\mathbf{G}_3: A \perp\!\!\!\perp B; A \perp\!\!\!\perp B | C; A \perp\!\!\!\perp C; A \perp\!\!\!\perp C | B$
- $\mathbf{G}_4: A \perp\!\!\!\perp B$
- $\mathbf{G}_5: A \perp\!\!\!\perp B | C$
- $\mathbf{G}_6: A \perp\!\!\!\perp C | B$
- $\mathbf{G}_7: B \perp\!\!\!\perp C | A$
- $\mathbf{G}_8: A \perp\!\!\!\perp C | B$
- $\mathbf{G}_9: \{\}$
- $\mathbf{G}_{10}: \{\}$
- $\mathbf{B}_1: A \perp\!\!\!\perp B; A \perp\!\!\!\perp B | C; B \perp\!\!\!\perp C; B \perp\!\!\!\perp C | A$
- $\mathbf{B}_2: A \perp\!\!\!\perp C | B$
- $\mathbf{B}_3: A \perp\!\!\!\perp C$

a. $\mathbf{G}_4, \mathbf{G}_5, \mathbf{G}_7, \mathbf{G}_9, \mathbf{G}_{10}$.

Since \mathbf{B}_1 can represent \mathbf{d}_1 , we know that \mathbf{d}_1 must satisfy the assumptions that \mathbf{B}_1 follows, which are:

$A \perp\!\!\!\perp B; A \perp\!\!\!\perp B | C; B \perp\!\!\!\perp C; B \perp\!\!\!\perp C | A$. We cannot assume that \mathbf{d}_1 satisfies the other two assumptions, which are $A \perp\!\!\!\perp C$ and $A \perp\!\!\!\perp C | B$, and so a Bayes net that makes at least one of these two extra assumptions will not be guaranteed to be able to represent \mathbf{d}_1 . This eliminates the choices $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_6, \mathbf{G}_8$. The other choices $\mathbf{G}_4, \mathbf{G}_5, \mathbf{G}_7, \mathbf{G}_9, \mathbf{G}_{10}$ are guaranteed to be able to represent \mathbf{d}_1 because they do not make any additional independence assumptions that \mathbf{B}_1 makes.

b. $\mathbf{G}_6, \mathbf{G}_8, \mathbf{G}_9, \mathbf{G}_{10}$.

Since \mathbf{B}_2 can represent \mathbf{d}_2 , we know that \mathbf{d}_2 must satisfy the assumptions that \mathbf{B}_2 follows, which is just: $A \perp\!\!\!\perp C | B$. We cannot assume that \mathbf{d}_2 satisfies any other assumptions, and so a Bayes net that makes at least one other extra assumptions will not be guaranteed to be able to represent \mathbf{d}_2 . This eliminates the choices $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4, \mathbf{G}_5, \mathbf{G}_7$. The other choices $\mathbf{G}_6, \mathbf{G}_8, \mathbf{G}_9, \mathbf{G}_{10}$ are guaranteed to be able to represent \mathbf{d}_2 because they do not make any additional independence assumptions that \mathbf{B}_2 makes.

c. $\mathbf{G}_9, \mathbf{G}_{10}$.

Since \mathbf{B}_3 cannot represent \mathbf{d}_3 , we know that \mathbf{d}_3 is unable to satisfy at least one of the assumptions that \mathbf{B}_3 follows. Since \mathbf{B}_3 only makes one independence assumption, which is $A \perp\!\!\!\perp C$, we know that \mathbf{d}_3 does not satisfy $A \perp\!\!\!\perp C$. However, we can't claim anything about whether or not \mathbf{d}_3 makes any of the other independence assumptions. \mathbf{d}_3 might not make any (conditional) independence assumptions at all, and so only the Bayes nets that don't make any assumptions will be guaranteed to be able to represent \mathbf{d}_3 . Hence, the answers are the fully connected Bayes nets, which are $\mathbf{G}_9, \mathbf{G}_{10}$.

d. All ten networks can represent d_4 .

Since $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ can represent \mathbf{d}_4 , we know that \mathbf{d}_4 must satisfy the assumptions that $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ make. The union of assumptions made by these Bayes nets are: $A \perp\!\!\!\perp$

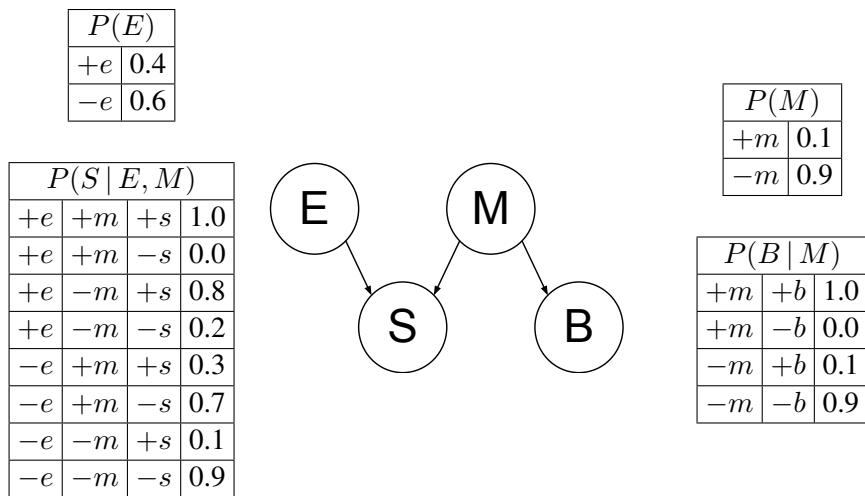


Figure S13.28 A Bayes net for the end of the world.

$\perp B; A \perp\!\!\!\perp B | C; B \perp\!\!\!\perp C | A, A \perp\!\!\!\perp C, A \perp\!\!\!\perp C | B$. Note that this set of assumptions encompasses all the possible assumptions that you can make with 3 random variables, so any Bayes net over A, B, C will be able to represent d_4 .

13.3 Exact Inference in Bayesian Networks

Exercise 13.3.#DOOM

A smell of sulphur (S) can be caused either by rotten eggs (E) or as a sign of the doom brought by the Mayan Apocalypse (M). The Mayan Apocalypse also causes the oceans to boil (B). The Bayesian network and corresponding conditional probability tables for this situation are shown in Figure S13.28.

- Compute the joint probability $P(-e, -s, -m, -b)$.
 - What is the probability that the oceans boil?
 - What is the probability that the Mayan Apocalypse is occurring, given that the oceans are boiling?
 - What is the probability that the Mayan Apocalypse is occurring, given that there is a smell of sulphur, the oceans are boiling, and there are rotten eggs?
 - What is the probability that rotten eggs are present, given that the Mayan Apocalypse is occurring?
-
- $P(-e, -s, -m, -b) = P(-e)P(-m)P(-s | -e, -m)P(-b | -m) = (0.6)(0.9)(0.9)(0.9) = 0.4374$.
 - $P(+b) = P(+b | +m)P(+m) + P(+b | -m)P(-m) = (1.0)(0.1) + (0.1)(0.9) = 0.19$.

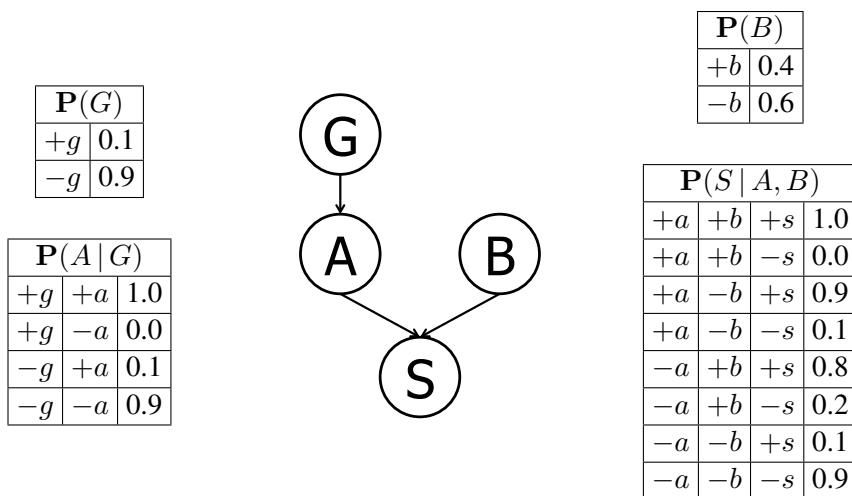


Figure S13.29 A Bayes net for genetic disease.

c. $P(+m | +b) = \frac{P(+b | +m)P(+m)}{P(+b)} = \frac{(1.0)(0.1)}{0.19} \approx .5263.$

d.

$$\begin{aligned} P(+m | +s, +b, +e) &= \frac{P(+m, +s, +b, +e)}{\sum_m P(m, +s, +b, +e)} \\ &= \frac{P(+e)P(+m)P(+s | +e, +m)P(+b | +m)}{\sum_m P(+e)P(m)P(+s | +e, m)P(+b | m)} \\ &= \frac{(0.4)(0.1)(1.0)(1.0)}{(0.4)(0.1)(1.0)(1.0) + (0.4)(0.9)(0.8)(0.1)} \\ &= \frac{0.04}{0.04 + 0.0288} \approx .5814 \end{aligned}$$

e. $P(+e | +m) = P(+e) = 0.4$ since E is independent of M .

Exercise 13.3.#BGEN

Suppose that a patient can have a symptom (S) that can be caused by two different diseases (A and B). It is known that the variation of gene G plays a big role in the occurrence of disease A . The Bayes net and corresponding conditional probability tables for this situation are in Figure S13.29.

- Compute the joint probability $\mathbf{P}(+g, +a, +b, +s)$.
- What is the probability that a patient has disease A ?
- What is the probability that a patient has disease A given that they have disease B ?
- What is the probability that a patient has disease A given that they have symptom S and disease B ?

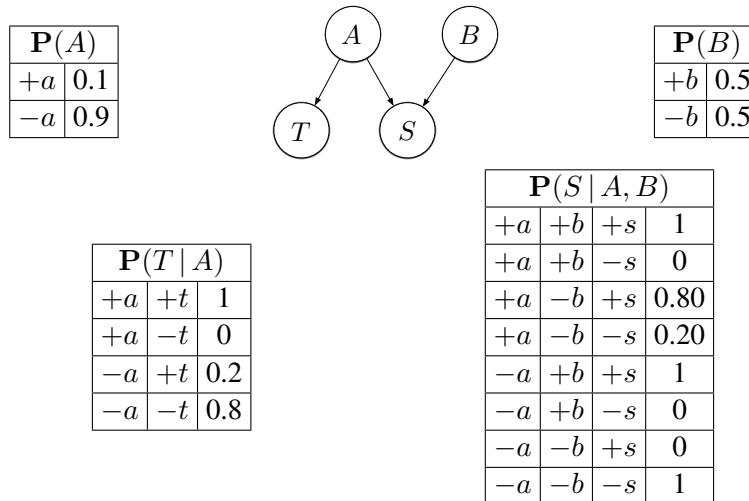


Figure S13.30 A Bayes net with a test for disease.

- e. What is the probability that a patient has the disease-carrying gene variation G given that they have disease A ?
- f. What is the probability that a patient has the disease-carrying gene variation G given that they have disease B ?

- a. $P(+g, +a, +b, +s) = P(+g)P(+a | +g)P(+b)P(+s | +b, +a) = (0.1)(1.0)(0.4)(1.0) = 0.04$.
- b. $P(+a) = P(+a | +g)P(+g) + P(+a | -g)P(-g) = (1.0)(0.1) + (0.1)(0.9) = 0.19$.
- c. $P(+a | +b) = P(+a) = 0.19$ because A and B are absolutely independent.
- d.
$$P(+a | +s, +b) = \frac{P(+a, +b, +s)}{P(+a, +b, +s) + P(-a, +b, +s)} = \frac{P(+a)P(+b)P(+s | +a, +b)}{P(+a)P(+b)P(+s | +a, +b) + P(-a)P(+b)P(+s | -a, +b)} \\ = \frac{(0.19)(0.4)(1.0)}{(0.19)(0.4)(1.0) + (0.81)(0.4)(0.8)} = \frac{0.076}{0.076 + 0.2592} \approx 0.2267.$$
- e.
$$P(+g | +a) = \frac{P(+g)P(+a | +g)}{P(+g)P(+a | +g) + P(-g)P(+a | -g)} = \frac{(0.1)(1.0)}{(0.1)(1.0) + (0.9)(0.1)} = \frac{0.1}{0.1 + 0.09} = 0.5263.$$
- f. $P(+g | +b) = P(+g) = 0.1$, since $G \perp\!\!\!\perp B$.

Exercise 13.3.#BTST

Suppose that a patient can have a symptom (S) that can be caused by two different diseases (A and B). Disease A is much rarer, but there is a test T that tests for the presence of A . The Bayes net and corresponding conditional probability tables for this situation are shown in Figure S13.30.

- a. Compute the entry $\mathbf{P}(-a, -t, +b, +s)$ from the joint distribution.

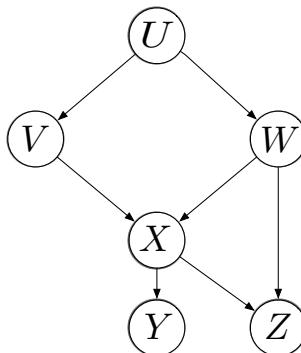


Figure S13.31 A simple Bayes net.

- b. What is the probability that a patient has disease A given that they have disease B ?
- c. What is the probability that a patient has disease A given that they have symptom S , disease B , and test T returns positive?
- d. What is the probability that a patient has disease A given that they have symptom S and test T returns positive?
- e. Suppose that both diseases A and B become more likely as a person ages. Add any necessary variables and/or arcs to the Bayes net to represent this change. For any variables you add, *briefly* (one sentence or less) state what they represent. Also, state one independence or conditional independence assertion that is *removed* due to your changes.
- f. Based only on the structure of the (new) Bayes net given in Figure S13.31, decide whether the following conditional independence assertions are guaranteed to be true, guaranteed to be false, or cannot be determined by the structure alone.

- (i) $V \perp\!\!\!\perp W$
- (ii) $V \perp\!\!\!\perp W \mid U$
- (iii) $V \perp\!\!\!\perp W \mid U, Y$
- (iv) $V \perp\!\!\!\perp Z \mid U, X$
- (v) $X \perp\!\!\!\perp Z \mid W$

- a. $P(-a, -t, +b, +s) = P(-t \mid -a)P(-a)P(+s \mid +b, -a)P(+b) = (0.8)(0.9)(1)(0.5) = 0.36$.
- b. The network asserts absolute independence of A and B , so $P(+a \mid +b) = P(+a) = 0.1$.

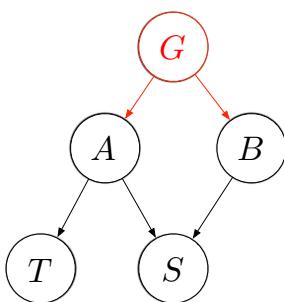
c.

$$\begin{aligned} P(+a | +t, +s, +b) &= \frac{\mathbf{P}(+a, +t, +s, +b)}{\mathbf{P}(+t, +s, +b)} = \frac{\mathbf{P}(+a)\mathbf{P}(+t | +a)\mathbf{P}(+s | +a, +b)\mathbf{P}(+b)}{\sum_{a \in \{+a, -a\}} \mathbf{P}(a, +t, +s, +b)} \\ &= \frac{(0.1)(1)(1)(0.5)}{(0.1)(1)(1)(0.5) + (0.9)(0.2)(1)(0.5)} = \frac{.05}{.05 + .09} = \frac{5}{14} \approx 0.35. \end{aligned}$$

d. $\mathbf{P}(+a | +t, +s) = \frac{\mathbf{P}(+a, +t, +s)}{\mathbf{P}(+t, +s)} = \frac{\sum_b \mathbf{P}(+a)\mathbf{P}(+t | +a)\mathbf{P}(+s | +a, b)\mathbf{P}(b)}{\sum_a \sum_b \mathbf{P}(a)\mathbf{P}(+t | a)\mathbf{P}(+s | a, b)\mathbf{P}(b)} = 0.5.$

e. **New variable(s) (and brief definition):** G: Age in years, or as a Boolean for old or not, etc.

Removed conditional independence assertion: There are a few. The simplest is $A \perp\!\!\!\perp B$ is no longer guaranteed. Another is $B \perp\!\!\!\perp T$.



- f. (i) $V \perp\!\!\!\perp W$ Cannot be determined.
 (ii) $V \perp\!\!\!\perp W | U$ Guaranteed true.
 (iii) $V \perp\!\!\!\perp W | U, Y$ Cannot be determined.
 (iv) $V \perp\!\!\!\perp Z | U, X$ Cannot be determined.
 (v) $X \perp\!\!\!\perp Z | W$ Cannot be determined.

Exercise 13.3.#BJNT

- a. Express the joint probability distribution $\mathbf{P}(A, B, C)$ induced by the Bayes net in Figure S13.32 in terms of the associated conditional probability distributions.

Compute the values of the following probabilities:

- b. $P(C = \text{true})$.
 c. $P(A = \text{true}, B = \text{true})$.
 d. $(A = \text{true}, B = \text{true} | C = \text{true})$.

- a. The joint probability is the product of the conditional probability if each variable given its parents. A and B have no parents, so we have $\mathbf{P}(A, B, C) = P(A)P(B)P(C | A, B)$.
 b. $P(C = \text{true}) = \sum_{a,b} P(a)P(b)P(C = \text{true} | a, b) = \frac{1}{4}\frac{3}{4}1 + \frac{1}{4}\frac{1}{4}0 + \frac{3}{4}\frac{1}{4}\frac{1}{2} + \frac{3}{4}\frac{1}{4}0 = \frac{15}{32}$.
 c. $P(A = \text{true}, B = \text{true}) = P(A = \text{true})P(B = \text{true}) = \frac{3}{16}$, by absolute independence of A and B.

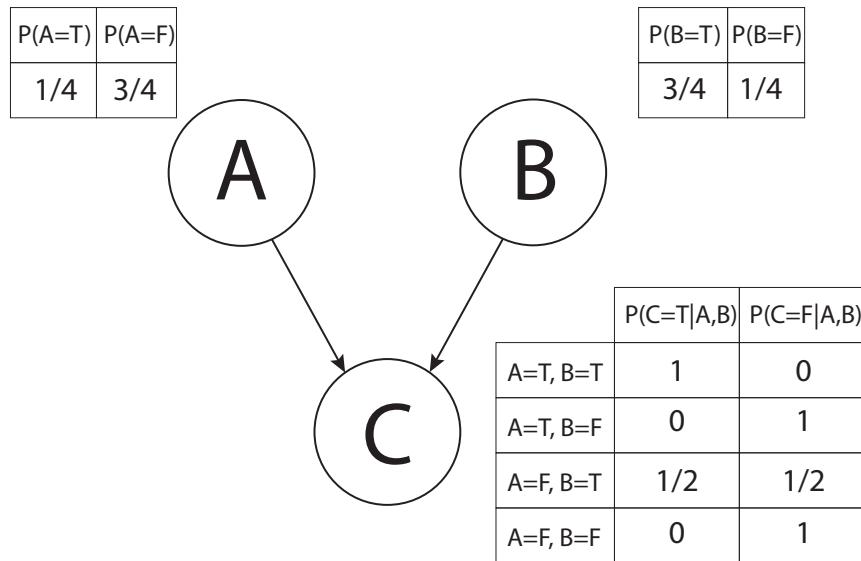


Figure S13.32 A Bayesian network for Exercise 13.BJNT.

d.

$$\begin{aligned}
 (A = \text{true}, B = \text{true} | C = \text{true}) &= \frac{P(A = \text{true}, B = \text{true}, C = \text{true})}{P(C = \text{true})} \\
 &= \frac{P(A = \text{true})P(B = \text{true})P(C = \text{true}|A = \text{true}, B = \text{true})}{P(C = \text{true})} \\
 &= (\frac{1}{4} \frac{3}{4} 1) / \frac{15}{32} = \frac{2}{5}
 \end{aligned}$$

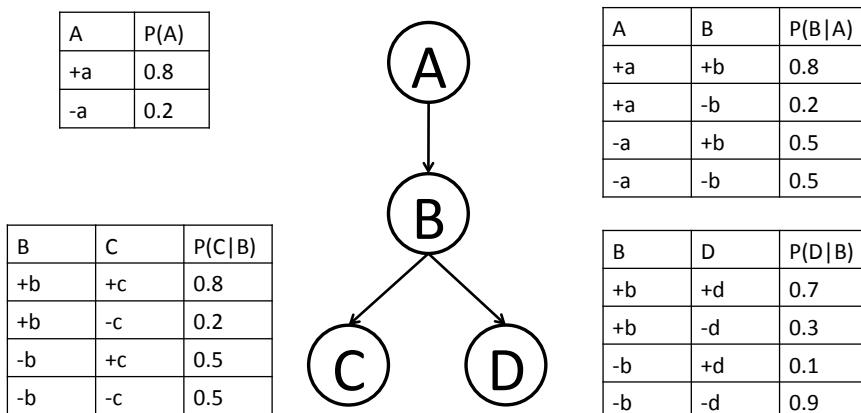


Figure S13.33 A Bayesian network for Exercise 13.BJTW.

Exercise 13.3.#BJTW

Consider the joint distribution $P(A, B, C, D)$ defined by the Bayes net in Figure S13.33. Compute the values of the following quantities:

- $P(A = \text{true})$.
- $P(A = \text{true}, B = \text{false}, C = \text{false}, D = \text{true})$.
- $P(A = \text{true}, B = \text{false}, C = \text{false}, D = \text{true})$.

- 0.8.
- $0.8 \times 0.2 \times 0.1 \times 0.5 = 0.008$.
- $\frac{0.8 \times 0.2 \times 0.1 \times 0.5}{0.8 \times 0.2 \times 0.1 \times 0.5 + 0.2 \times 0.5 \times 0.1 \times 0.5} = 0.615$

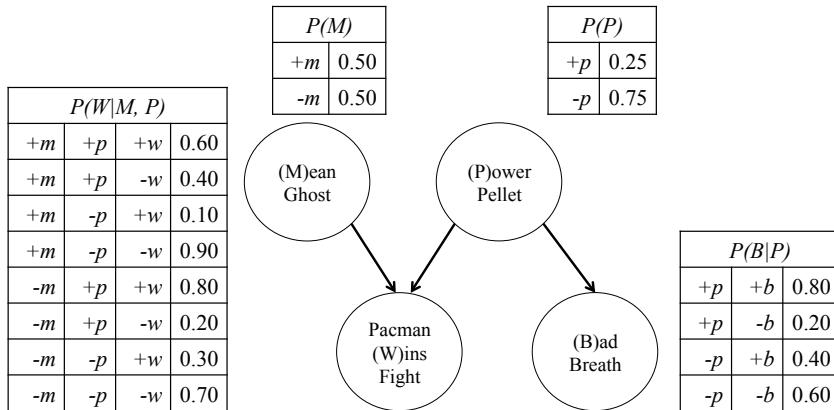


Figure S13.34 A Bayesian network for Exercise 13.PACL.

$P(M, P, W, B)$				
$+m$	$+p$	$+w$	$+b$	0.0800
$+m$	$+p$	$+w$	$-b$	0.0150
$+m$	$+p$	$-w$	$+b$	0.0400
$+m$	$+p$	$-w$	$-b$	0.0100
$+m$	$-p$	$+w$	$+b$	0.0150
$+m$	$-p$	$+w$	$-b$	0.0225
$+m$	$-p$	$-w$	$+b$	0.1350
$+m$	$-p$	$-w$	$-b$	0.2025

Figure S13.35 A table for Exercise 13.PACL.

Exercise 13.3.#PACL

PacLabs has just created a new type of mini power pellet that is small enough for Pacman to carry around with him when he's running around mazes. Unfortunately, these mini-pellets don't guarantee that Pacman will win all his fights with ghosts, and they look just like the regular dots Pacman carries around to snack on.

Pacman just ate a snack (P), which was either a mini-pellet ($+p$), or a regular dot ($-p$), and is about to get into a fight (W), which he can win ($+w$) or lose ($-w$). Both these variables are unknown, but fortunately, Pacman is a master of probability. He knows that his bag of snacks has 5 mini-pellets and 15 regular dots. He also knows that if he ate a mini-pellet, he has a 70% chance of winning, but if he ate a regular dot, he only has a 20% chance.

- a. What is $P(+w)$, the marginal probability that Pacman will win?
- b. Pacman won! Hooray! What is the conditional probability $P(+p \mid +w)$ that the food he ate was a mini-pellet, given that he won?

Pacman can make better probability estimates if he takes more information into account. First, Pacman's breath, B , can be bad ($+b$) or fresh ($-b$). Second, there are two types of ghost (M): mean ($+m$) and nice ($-m$). Pacman has encoded his knowledge about the situation in the Bayes net in Figure S13.34.

- c. What is the probability of the atomic event $(-m, +p, +w, -b)$, where Pacman eats a mini-pellet and has fresh breath before winning a fight against a nice ghost?
- d. Which of the following conditional independence statements are guaranteed to be true by the Bayes net graph structure?

- (i) $W \perp\!\!\!\perp B$
- (ii) $W \perp\!\!\!\perp B \mid P$
- (iii) $M \perp\!\!\!\perp P$
- (iv) $M \perp\!\!\!\perp P \mid W$
- (v) $M \perp\!\!\!\perp B$
- (vi) $M \perp\!\!\!\perp B \mid P$
- (vii) $M \perp\!\!\!\perp B \mid W$

For the remainder of this question, use the half of the joint probability table that has been computed for you in Figure S13.35.

- e. What is the marginal probability, $P(+m, +b)$ that Pacman encounters a mean ghost and has bad breath?
- f. Pacman observes that he has bad breath and that the ghost he's facing is mean. What is the conditional probability, $P(+w \mid +m, +b)$, that he will win the fight, given his observations?
- g. Pacman's utility is +10 for winning a fight, -5 for losing a fight, and -1 for running away from a fight. Pacman wants to maximize his expected utility. Given that he has bad breath and is facing a mean ghost, should he stay and fight, or run away? Justify your answer numerically!

a.

$$\begin{aligned} P(+w) &= P(+w, +p) + P(+w, -p) = P(+w| +p)P(+p) + P(+w| -p)P(-p) \\ &= \frac{7}{10} \times \frac{1}{4} + \frac{2}{10} \times \frac{3}{4} = \frac{13}{40} = 0.325 \end{aligned}$$

b.

$$\begin{aligned} P(+p| +w) &= \frac{P(+w, +p)}{P(+w)} = \frac{P(+w| +p)P(+p)}{P(+w)} \\ &= \frac{\frac{7}{10} \times \frac{1}{4}}{\frac{13}{40}} = \frac{7}{13} \approx 0.538 \end{aligned}$$

c. $P(-m, +p, +w, -b) = P(-m)P(+p)P(+w|-m, +p)P(-b|+p) = \frac{1}{2} \times \frac{1}{4} \times \frac{4}{5} \times \frac{1}{5} = \frac{1}{50} = 0.02.$

d. Conditional independence assertions ii, iii, v, and vi are correct.

e. $P(+m, +b) = 0.08 + 0.04 + 0.015 + 0.135 = 0.27.$

f. $P(+w| +m, +b) = \frac{P(+w, +m, +b)}{P(+m, +b)} = \frac{0.08+0.015}{0.27} = \frac{19}{54} \approx 0.352.$

g. This question is a simple preview of material from Chapter 16, but we have already seen the basic idea in the context of backgammon and other game of chance in Chapter ??.

Let U_f be the utility of fighting and U_r be the utility of running.

$$\begin{aligned} E(U_f| +m, +b) &= 10 \times P(+w| +m, +b) + (-5) \times P(-w| +m, +b) \\ &\approx 10 \times 0.352 - 5 \times 0.648 \\ &= 0.28 > -1 = U_r \end{aligned}$$

Since $E(U_f| +m, +b) > E(U_r| +m, +b)$, Pacman should stay and fight.

Exercise 13.3.#VETF

Which of the following are true assertions about the variable elimination algorithm, and which are false?

- a.** When changing a Bayes net by removing a parent from a variable, the maximum factor size (where size is the number of non-fixed variables involved in the factor) generated during the optimally ordered variable elimination is reduced by at most 1.
- b.** The ordering of variables in variable elimination affects the maximum factor size generated by at most a factor of two.
- c.** The size of factors generated during variable elimination is upper-bounded by twice the size of the largest conditional probability table in the original Bayes net.
- d.** The size of factors generated during variable elimination is the same if we exactly reverse the elimination ordering.
- e.** During variable elimination, the ordering of elimination does not affect the final answer.

a. False.

b. False.

c. False.

- d. False.
 - e. True. The algorithm is always sound.
-

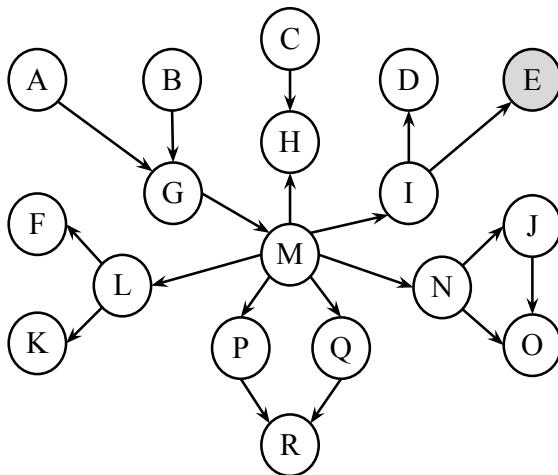


Figure S13.36 A Bayesian network for Exercise 13.ELIM.

Exercise 13.3.#ELIM

Consider the Bayes net in Figure S13.36 with the query $P(Q | +e)$.

- a. Which variables can be ignored when computing the answer to the query?
- b. Prove a general result concerning the irrelevance of variables in computing the posterior distribution of set of query variables given a set of evidence variables.

As stated in the chapter, variables that are not in the union of the ancestors of the query and ancestors of the evidence can be pruned.

- a. In computing $P(Q | +e)$, we can ignore C, D, F, H, J, K, L, N, O, P, R.
- b. The proof of irrelevance not shown.

Exercise 13.3.#ELMU

- a. For the Bayes net in Figure S13.37, we are given the query $P(Z | +y)$. All variables are Boolean. Assume we run variable elimination to compute the answer to this query, with the following variable elimination ordering: U, V, W, T, X .

After inserting evidence, we have the following factors to start out with:

$$P(U), P(V), P(W | U, V), P(X | V), P(T | V), P(+y | W, X), P(Z | T) .$$

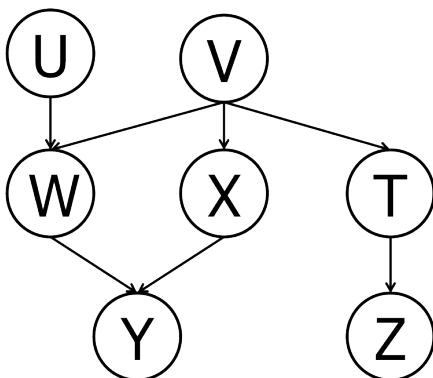


Figure S13.37 A Bayesian network for Exercise 13.ELMU.

When eliminating U we generate a new factor f_1 as follows:

$$f_1(V, W) = \sum_u P(u)P(W | u, V).$$

This leaves us with the factors:

$$P(V), P(X | V), P(T | V), P(+y | W, X), P(Z | T), f_1(V, W).$$

Now complete the remaining steps:

- (i) When eliminating V we generate a new factor f_2 as follows:
 - (ii) This leaves us with the factors:
 - (iii) When eliminating W we generate a new factor f_3 as follows:
 - (iv) This leaves us with the factors:
 - (v) When eliminating T we generate a new factor f_4 as follows:
 - (vi) This leaves us with the factor:
 - (vii) When eliminating X we generate a new factor f_5 as follows:
 - (viii) This leaves us with the factor:
- b. Briefly explain how $P(Z | +y)$ can be computed from f_5 .
- c. Amongst f_1, f_2, \dots, f_5 , which is the largest factor generated? How large is this factor?
- d. Find a variable elimination ordering for the same query, i.e., for $P(Z | y)$, for which the maximum size factor generated along the way is smallest. Hint: the maximum size factor generated in your solution should have only 2 variables, for a size of $2^2 = 4$ entries.

- a. (i) When eliminating V we generate a new factor f_2 as follows:

$$f_2(T, W, X) = \sum_v P(v)P(X | v)P(T | v)f_1(v, W).$$

- (ii) This leaves us with the factors:

$$P(+y | W, X), P(Z | T), f_2(T, W, X).$$

- (iii) When eliminating W we generate a new factor f_3 as follows:

$$f_3(T, X, +y) = \sum_w P(+y | w, X)f_2(T, w, X).$$

- (iv) This leaves us with the factors:

$$P(Z | T), f_3(T, X, +y).$$

- (v) When eliminating T we generate a new factor f_4 as follows:

$$f_4(X, +y, Z) = \sum_t P(Z | t)f_3(t, X, +y).$$

- (vi) This leaves us with the factor:

$$f_4(X, +y, Z).$$

- (vii) When eliminating X we generate a new factor f_5 as follows:

$$f_5(+y, Z) = \sum_x f_4(x, +y, Z).$$

- (viii) This leaves us with the factor:

$$f_5(+y, Z)$$

- b. Simply renormalize f_5 to obtain $P(Z | +y)$. Concretely, $P(z | +y) = \frac{f_5(z, +y)}{\sum_{z'} f_5(z', +y)}$.
- c. $f_2(T, W, X)$ is the largest factor generated. It has 3 variables, hence $2^3 = 8$ entries.
- d. One possible ordering is T, X, W, U, V .

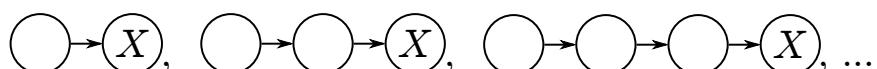


Figure S13.38 A Bayesian network for Exercise 13.SEQE.

Exercise 13.3.#SEQE

Consider the sequence of graphs in Figure S13.38 and the application of variable elimination to each in order to compute $P(X)$. For each, regardless of the elimination ordering, the largest factor produced in finding will have a table with 2^2 entries, assuming that all variables are Boolean.

Now draw a sequence of graphs such that, if you used the best elimination ordering for each graph, the largest factor table produced in variable elimination would have a constant number of entries, but if you used the worst elimination ordering for each graph, the number of entries in the largest factor table would grow exponentially as you move down the sequence. Provide (i) the sequence of graphs, (ii) the sequence of queries for which variable elimination is done, (iii) the best ordering, (iv) the worst ordering.

There are many solutions, here is one family of Bayes net graphs that will do, with query $P(Y_n | Z_1, \dots, Z_n)$, a best ordering: $Y_1, Y_2, \dots, Y_{n-1}, X$, a worst ordering: $X, Y_1, Y_2, \dots, Y_{n-1}$.

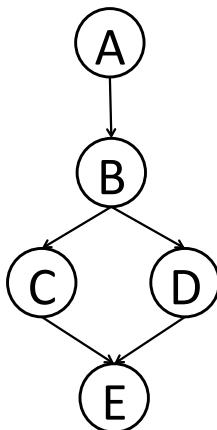
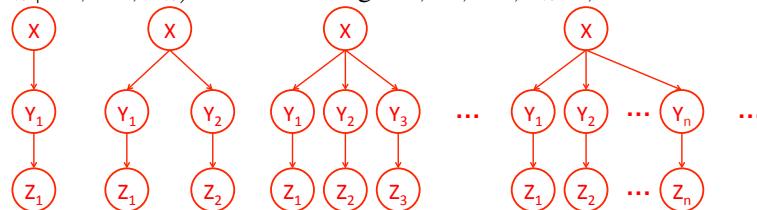


Figure S13.39 A Bayesian network for Exercise 13.ELMO.

Exercise 13.3.#ELMO

In this question, we consider the efficiency of variable elimination as a function of the elimination ordering. For any variable, X , let $|X|$ denote the size of X 's range (the number of values it can take). At any stage in the variable elimination process, there will be a set of

factors $\mathcal{F} = \{F_1, \dots, F_n\}$. Each factor F_i can be written as $P(L_i | R_i)$, where L_i and R_i are both sets of variables (for simplicity, assume that there is no evidence). For any variable X , we define $I(X)$ to be the set of indices of factors that include X : $I(X) = \{i \mid X \in (L_i \cup R_i)\}$.

When eliminating a specific variable, Y , we start by joining the appropriate factors from \mathcal{F} to create a single joined factor, called $join(Y, \mathcal{F})$. Note that $join(Y, \mathcal{F})$ is created *before* performing the elimination step, so it still includes Y .

- For a given factor, F , we use $size(F)$ to denote the size of the table needed to represent F . Give a precise general expression for $size(join(Y, \mathcal{F}))$, using the notation above.
- Consider a generic Bayes net with variables $X \in V$ that encodes a set of initial factors \mathcal{F}_0 , where the query is to compute the marginal $\mathbf{P}(Q)$. Formulate a standard search problem (as in Chapter 3) that determines the optimal variable elimination ordering, where cost is measured by the sum of the sizes of the tables created during elimination. Note that for this problem we are only concerned with the sum of the sizes of the tables created by the *join* step, not the smaller tables that are subsequently created by the *eliminate* step. (You may also find it helpful to use the notation $eliminate(X, F)$ to denote the result of summing over values of a variable X to eliminate X from factor F .)
- Let W be the set of variables remaining to be eliminated and let $parents(X)$ and $children(X)$ denote the sets containing all of X 's parents/children in the Bayes net. Which of the following are admissible heuristics for the search problem you defined in part (b)?
 - $|W|$
 - $|\mathcal{F}|$
 - $\sum_{X \in W} |X|$
 - $\prod_{X \in W} |X|$
 - $\sum_{X \in W} \left(|X| \prod_{Y \in parents(X)} |Y| \right)$
 - $\sum_{X \in W} \left(|X| \prod_{Y \in children(X)} |Y| \right)$
- Consider the query $P(D | +e, +c)$ on the Bayes net given in Figure S13.39. Draw the complete search tree for finding the optimal elimination order for this problem. Annotate nodes with states, and annotate costs and actions on the edges. Hint: the start state is $(\{P(A), P(B | A), P(+c | B), P(D | B), P(+e | +c, D)\}, \{A, B\})$, and your tree should have five nodes in all. What is the optimal plan and what is its cost?

- a. The size is $\prod_{I(X_i) \in \{Y, \mathcal{F}\}} |X_i|$

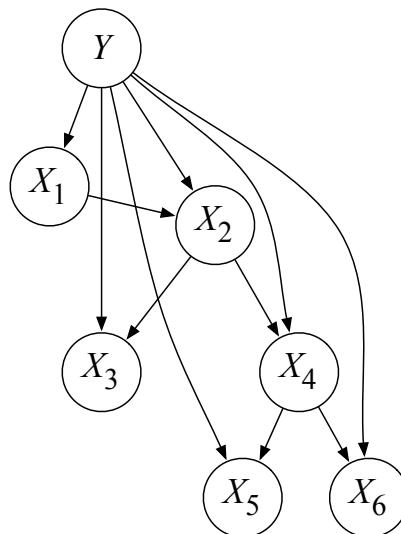
- b. The problem formulation is as follows:

- State space: a state consists of the current set of factors and a set of variables still to be eliminated.
- Initial state: the set of conditional probability tables from the Bayes net and the set of non-evidence, non-query variables.
- Goal states: any state where the set of variables still to be eliminated is empty.

- Actions: eliminate any variable in the set of variables still to be eliminated.
 - Transition model: the new state contains the new factor description that would result from summing out the variable to be eliminated (without actually doing the computation!); and the eliminated variable is removed from the set of variables still to be eliminated.
 - Action cost function: the number of entries in the table representation of the new factor.
- c. The admissible heuristics are (i), (ii), and (iii). The rest might overestimate the size of the table in networks that can efficiently eliminate variable.
- d. The tree is as follows:
-
- ```

graph TD
 Root["({P(A)}, P(B|A), P(+c|B),
P(D|B), P(+e|+c, D)), {A, B}]" -- "A, 2" --> Node1["({f1(B)},
P(+c|B),
P(D|B), P(+e|+c, D)), {B}"]
 Root -- "B, 4" --> Node2["({P(A)}, f2(A, +c, D),
P(+e|+c, D)), {A}"]
 Node1 -- "B, 2" --> Node3["({f3(+c, D, +e}), { })"]
 Node2 -- "A, 2" --> Node4["({f4(+c, D, +e}), { })"]

```
- There is one optimal plan: the left branch, which first eliminates  $A$  and then  $B$ . Its cost is  $2 + 2 = 4$ .



**Figure S13.40** A TANBnetwork for Exercise 13.TANB.

### Exercise 13.3.#TANB

A tree-augmented Naive Bayes model (TANB) is identical to a Naive Bayes model, except the features are no longer assumed conditionally independent given the class  $Y$ . Specif-

ically, if  $(X_1, X_2, \dots, X_n)$  are the observable features, a TANB allows  $X_1, \dots, X_n$  to be in a tree-structured Bayes net in addition to having  $Y$  as a parent. An example is given in Figure S13.40:

- a. Suppose we observe no variables as evidence in the TANB above. What is the classification rule for the TANB? Write the formula in terms of the conditional distributions in the TANB.
- b. Assume we observe all the variables  $X_1 = x_1, X_2 = x_2, \dots, X_6 = x_6$  in the TANB above. Write the classification rule for the TANB in terms of the conditional distributions.
- c. Specify an elimination order that is efficient for variable elimination applied to the query  $P(Y | X_5 = x_5)$  in the TANB above (the query variable  $Y$  should not be included in your ordering). How many variables are in the biggest factor (there may be more than one; if so, list only one of the largest) induced by variable elimination with your ordering? Which variables are they?
- d. Specify an elimination order that is efficient for the query  $P(X_3 | X_5 = x_5)$  in the TANB above (including  $X_3$  in your ordering). How many variables are in the biggest factor (there may be more than one; if so, list only one of the largest) induced by variable elimination with your ordering? Which variables are they?
- e. Does it make sense to run Gibbs sampling to do inference in a TANB? In two or fewer sentences, justify your answer.

- a. The solution is simply the maximal  $y$  according to the prior probabilities of  $y$ :  $\operatorname{argmax}_y P(y)$ .

- b. We want the most probable  $y$  given the variables  $X_1, \dots, X_6$ , which is (using the same reasoning as for Naive Bayes)

$$\begin{aligned}
 & \operatorname{argmax}_y P(y | x_1, \dots, x_6) \\
 &= \operatorname{argmax}_y P(y, x_1, \dots, x_6) \\
 &= \operatorname{argmax}_y P(y)P(x_1 | y)P(x_2 | x_1, y)P(x_3 | x_2, y)P(x_4 | x_2, y) \\
 &\quad P(x_5 | x_4, y)P(x_6 | x_4, y).
 \end{aligned}$$

- c. We can ignore the variables  $X_3$  and  $X_6$ , since when we marginalize them in the elimination order, they will sum to 1. Thus any elimination order including  $X_3$  or  $X_6$  is

incorrect. Otherwise, we essentially just walk up the tree:

$$\begin{aligned}
 P(Y | x_5) &\propto P(Y, x_5) = \sum_{x_1, x_2, x_3, x_4, x_6} P(Y, x_1, \dots, x_6) \\
 &= P(Y) \sum_{x_1, \dots, x_4, x_6} P(x_1 | Y) P(x_2 | x_1, Y) P(x_3 | x_2, Y) P(x_4 | x_2, Y) \\
 &\quad P(x_5 | x_4, Y) P(x_6 | x_4, Y) \\
 &= P(Y) \sum_{x_1, x_2, x_4} P(x_1 | Y) P(x_2 | x_1, Y) P(x_4 | x_2, Y) P(x_5 | x_4, Y) \\
 &\quad \underbrace{\sum_{x_3} P(x_3 | x_1, Y) \sum_{x_6} P(x_6 | x_4, Y)}_{=1} \\
 &= P(Y) \sum_{x_1} P(x_1 | Y) \sum_{x_2} P(x_2 | x_1, Y) \sum_{x_4} P(x_4 | x_2, Y) P(x_5 | x_4, Y)
 \end{aligned}$$

So possible orders include  $X_4 \prec X_1 \prec X_2$ ,  $X_4 \prec X_2 \prec X_1$ ,  $X_1 \prec X_2 \prec X_4$ , and  $X_1 \prec X_4 \prec X_2$ . Any order must start with one of  $X_4$  or  $X_1$ , then so long as  $X_2$  follows one of them it is correct.

- d. Given that  $X_6$  is a child node of  $X_4$ , it marginalizes to 1 and has no effect on inference. So any elimination ordering including  $X_6$  is incorrect. Other than that, we can explicitly compute the elimination by marginalizing, and we get

$$\begin{aligned}
 P(X_3 | x_5) &\propto P(X_3, x_5) \\
 &= \sum_{x_1, x_2, x_4, y} P(y) P(x_1 | y) P(x_2 | x_1, y) P(X_3 | x_2, y) P(x_5 | x_4, y) P(x_4 | x_2, y) \\
 &= \sum_y P(y) \sum_{x_2} P(X_3 | x_2, y) \sum_{x_1} P(x_1 | y) P(x_2 | x_1, y) \sum_{x_4} P(x_5 | x_4, y) P(x_4 | x_2, y).
 \end{aligned}$$

The maximum factor size is 3; one such example above is  $(X_4, X_2, Y)$ . So one possible ordering is  $X_4 \prec X_1 \prec X_2 \prec Y \prec X_3$ . The possible orders must have one of  $X_4$  and  $X_1$  first, which yields a factor over  $(X_2, Y)$  as well as factors  $(X_2, Y, X_3)$  and  $(Y)$ . Then any ordering of  $X_2, Y$  will be fine.  $X_3$  must clearly be last. Note that eliminating  $Y$  early or  $X_2$  before eliminating one of  $X_4$  and  $X_1$  will yield factors of size 4, so that is incorrect.

- e. No, it does not really make sense to perform Gibbs sampling. Inference is always efficient—factors are of size at most 3—because everything is a tree given the node  $Y$ .

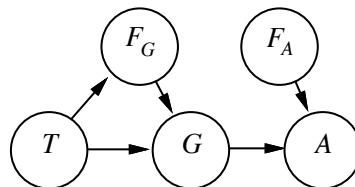
### Exercise 13.3.#NUKE

In your local nuclear power station, there is an alarm that senses when a temperature gauge exceeds a given threshold. The gauge measures the temperature of the core. Consider the Boolean variables  $A$  (alarm sounds),  $F_A$  (alarm is faulty), and  $F_G$  (gauge is faulty) and the multivalued nodes  $G$  (gauge reading) and  $T$  (actual core temperature).

- Draw a Bayesian network for this domain, given that the gauge is more likely to fail when the core temperature gets too high.
- Is your network a polytree? Why or why not?
- Suppose there are just two possible actual and measured temperatures, normal and high; the probability that the gauge gives the correct temperature is  $x$  when it is working, but  $y$  when it is faulty. Give the conditional probability table associated with  $G$ .
- Suppose the alarm works correctly unless it is faulty, in which case it never sounds. Give the conditional probability table associated with  $A$ .
- Suppose the alarm and gauge are working and the alarm sounds. Calculate an expression for the probability that the temperature of the core is too high, in terms of the various conditional probabilities in the network.

This question exercises many aspects of the student's understanding of Bayesian networks and uncertainty.

- A suitable network is shown in Figure S13.41. The key aspects are: the failure nodes are parents of the sensor nodes, and the temperature node is a parent of both the gauge and the gauge failure node. It is exactly this kind of correlation that makes it difficult for humans to understand what is happening in complex systems with unreliable sensors.



**Figure S13.41** A Bayesian network for the nuclear alarm problem.

- No matter which way the student draws the network, it should not be a polytree because of the fact that the temperature influences the gauge in two ways.
- The CPT for  $G$  is shown below. Students should pay careful attention to the semantics of  $F_G$ , which is true when the gauge is *faulty*, i.e., *not* working.

|                     | $T = \text{Normal}$ |            | $T = \text{High}$ |            |
|---------------------|---------------------|------------|-------------------|------------|
|                     | $F_G$               | $\neg F_G$ | $F_G$             | $\neg F_G$ |
| $G = \text{Normal}$ | $y$                 | $x$        | $1 - y$           | $1 - x$    |
| $G = \text{High}$   | $1 - y$             | $1 - x$    | $y$               | $x$        |

- The CPT for  $A$  is as follows:

|          | $G = \text{Normal}$ | $G = \text{High}$ |       |            |
|----------|---------------------|-------------------|-------|------------|
|          | $F_A$               | $\neg F_A$        | $F_A$ | $\neg F_A$ |
| $A$      | 0                   | 0                 | 0     | 1          |
| $\neg A$ | 1                   | 1                 | 1     | 0          |

- e. This part actually asks the student to do something usually done by Bayesian network algorithms. The great thing is that doing the calculation without a software package makes it easy to see the nature of the calculations that the algorithms are systematizing. It illustrates the magnitude of the achievement involved in creating complete and correct algorithms.

Abbreviating  $T = \text{High}$  and  $G = \text{High}$  by  $T$  and  $G$ , the probability of interest here is  $P(T|A, \neg F_G, \neg F_A)$ . Because the alarm's behavior is deterministic, we can reason that if the alarm is working and sounds,  $G$  must be  $\text{High}$ . Because  $F_A$  and  $A$  are d-separated from  $T$ , we need only calculate  $P(T|\neg F_G, G)$ .

There are several ways to go about doing this. The “opportunistic” way is to notice that the CPT entries give us  $P(G|T, \neg F_G)$ , which suggests using the generalized Bayes’ Rule to switch  $G$  and  $T$  with  $\neg F_G$  as background:

$$P(T|\neg F_G, G) \propto P(G|T, \neg F_G)P(T|\neg F_G)$$

We then use Bayes’ Rule again on the last term:

$$P(T|\neg F_G, G) \propto P(G|T, \neg F_G)P(\neg F_G|T)P(T)$$

A similar relationship holds for  $\neg T$ :

$$P(\neg T|\neg F_G, G) \propto P(G|\neg T, \neg F_G)P(\neg F_G|\neg T)P(\neg T)$$

Normalizing, we obtain

$$P(T|\neg F_G, G) = \frac{P(G|T, \neg F_G)P(\neg F_G|T)P(T)}{P(G|T, \neg F_G)P(\neg F_G|T)P(T) + P(G|\neg T, \neg F_G)P(\neg F_G|\neg T)P(\neg T)}$$

The “systematic” way to do it is to revert to joint entries (noticing that the subgraph of  $T$ ,  $G$ , and  $F_G$  is completely connected so no loss of efficiency is entailed). We have

$$P(T|\neg F_G, G) = \frac{P(T, \neg F_G, G)}{P(G, \neg F_G)} = \frac{P(T, \neg F_G, G)}{P(T, G, \neg F_G) + P(T, G, F_G)}$$

Now we use the chain rule to rewrite the joint entries as CPT entries:

$$P(T|\neg F_G, G) = \frac{P(T)P(\neg F_G|T)P(G|T, \neg F_G)}{P(T)P(\neg F_G|T)P(G|T, \neg F_G) + P(\neg T)P(\neg F_G|\neg T)P(G|\neg T, \neg F_G)}$$

which of course is the same as the expression arrived at above. Letting  $P(T) = p$ ,

$P(F_G|T) = g$ , and  $P(F_G|\neg T) = h$ , we get

$$P(T|\neg F_G, G) = \frac{p(1-g)(1-x)}{p(1-g)(1-x) + (1-p)(1-h)x}$$

### Exercise 13.3.#CHEA

Cheating dealers have become a serious problem at the mini-Blackjack tables. A mini-Blackjack deck has 3 card types (5,10,11) and an honest dealer is equally likely to deal each of the 3 cards. When a player holds 11, cheating dealers deal a 5 with probability  $\frac{1}{4}$ , 10 with probability  $\frac{1}{2}$ , and 11 with probability  $\frac{1}{4}$ . You estimate that  $\frac{4}{5}$  of the dealers in your casino are honest ( $H = \text{true}$ ) while  $\frac{1}{5}$  are cheating ( $H = \text{false}$ ).

- a. You see a dealer deal an 11 to a player holding 11. What is the probability that the dealer is cheating?

The casino has decided to install a camera to observe its dealers. Cheating dealers are observed doing suspicious things on camera ( $C = \text{true}$ )  $\frac{4}{5}$  of the time, while honest dealers are observed doing suspicious things  $\frac{1}{4}$  of the time.

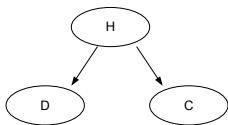
- b. Draw a Bayes net with the variables  $H$  (honest dealer),  $D$  (card dealt to a player holding 11), and  $C$  (suspicious behavior on camera). Write the conditional probability tables.
- c. List all conditional independence assertions made by your Bayes net.
- d. What is the probability that a dealer is honest given that he deals a 10 to a player holding 11 and is observed doing something suspicious?

You can either arrest dealers or let them continue working. If you arrest a dealer and he turns out to be cheating, you will earn a \$4 bonus. However, if you arrest the dealer and he turns out to be innocent, he will sue you for -\$10. Allowing the cheater to continue working will cost you -\$2, while allowing an honest dealer to continue working will get you \$1. Assume a linear utility function  $U(x) = x$ .

- e. You observe a dealer doing something suspicious ( $C$ ) and also observe that he deals a 10 to a player holding 11. Should you arrest the dealer?
- f. A private investigator approaches you and offers to investigate the dealer from the previous part. If you hire him, he will tell you with 100% certainty whether the dealer is cheating or honest, and you can then make a decision about whether to arrest him or not. How much would you be willing to pay for this information?

a.

$$\begin{aligned} P(\neg H | D = 11) &= \frac{P(\neg H, D = 11)}{P(D = 11)} \\ &= \frac{P(D = 11 | \neg H)P(\neg H)}{P(D = 11 | \neg H)P(\neg H) + P(D = 11 | H)P(H)} \\ &= \frac{(1/4)(2/10)}{(1/4)(2/10) + (1/3)(8/10)} = 3/19 \end{aligned}$$



b.

|  | $H$   | $P(H)$ |
|--|-------|--------|
|  | true  | 0.8    |
|  | false | 0.2    |

| $H$   | $D$ | $P(D   H)$ |
|-------|-----|------------|
| true  | 5   | 1/3        |
| true  | 10  | 1/3        |
| true  | 11  | 1/3        |
| false | 5   | 1/4        |
| false | 10  | 1/2        |
| false | 11  | 1/4        |

| $H$   | $C$   | $P(C   H)$ |
|-------|-------|------------|
| true  | true  | 1/4        |
| true  | false | 3/4        |
| false | true  | 4/5        |
| false | false | 1/5        |

c. The Bayes net asserts only  $D \perp\!\!\!\perp C | H$ .

d.

$$\begin{aligned}
 P(h | D = 10, c) &= \frac{P(h, D = 10, c)}{P(D = 10, c)} \\
 &= \frac{P(h)P(D = 10 | h)p(C | H)}{P(h)P(D = 10 | h)P(c | h) + P(\neg h)P(D = 10 | \neg h)P(c | \neg h)} \\
 &= \frac{(4/5)(1/3)(1/4)}{(4/5)(1/3)(1/4) + (1/5)(1/2)(4/5)} = \frac{5}{11}
 \end{aligned}$$

e. This question is a simple preview of material from Chapter 16, but we have already seen the basic idea in the context of backgammon and other game of chance in Chapter ???. Arresting the dealer yields an expected payoff of

$$4 * P(\neg H | D = 10, C) + (-10) * P(H | D = 10, C) = 4(6/11) + (-10)(5/11) = -26/11 /$$

Letting him continue working yields a payoff of

$$(-2) * P(\neg H | D = 10, C) + 1 * P(H | D = 10, C) = (-2)(6/11) + (1)(5/11) = -7/11 .$$

Therefore, you should let the dealer continue working.

f. This question addresses the theory of information value from Chapter 16. If used prior to students studying that material, it might be a good idea to provide a hint.

If you hire the private investigator, if the dealer is a cheater you can arrest him for a payoff of \$4. If he is an honest dealer you can let him continue working for a payoff of

\$1. The benefit from hiring the investigator is therefore

$$(4) * P(\neg h \mid D = 10, c) + 1 * P(h \mid D = 10, c) = 4(6/11) + (1)(5/11) = 29/11.$$

If you do not hire the investigator, your best course of action is to let the dealer continue working for an expected payoff of  $-7/11$ . Therefore, you are willing to pay up to  $29/11 - (-7/11) = 36/11$  to hire the investigator.

### Exercise 13.3.#TELC

Consider the network shown in Figure S13.11(ii), and assume that the two telescopes work identically.  $N \in \{1, 2, 3\}$  and  $M_1, M_2 \in \{0, 1, 2, 3, 4\}$ , with the symbolic CPTs as described in Exercise TELESCOPE-EXERCISE. Using the enumeration algorithm (Figure 13.11 on page 429), calculate the probability distribution  $\mathbf{P}(N \mid M_1 = 2, M_2 = 2)$ .

The symbolic expression evaluated by the enumeration algorithm is

$$\begin{aligned}\mathbf{P}(N \mid M_1 = 2, M_2 = 2) &= \alpha \sum_{f_1, f_2} \mathbf{P}(f_1, f_2, N, M_1 = 2, M_2 = 2) \\ &= \alpha \sum_{f_1, f_2} P(f_1)P(f_2)\mathbf{P}(N)P(M_1 = 2 \mid f_1, N)P(M_2 = 2 \mid f_2, N).\end{aligned}$$

Because an out-of-focus telescope cannot report 2 stars in the given circumstances, the only non-zero term in the summation is for  $F_1 = F_2 = \text{false}$ , so the answer is

$$\begin{aligned}\mathbf{P}(N \mid M_1 = 2, M_2 = 2) &= \alpha(1 - f)(1 - f)\langle p_1, p_2, p_3 \rangle \langle e, (1 - 2e), e \rangle \langle e, (1 - 2e), e \rangle \\ &= \alpha' \langle p_1 e^2, p_2 (1 - 2e)^2, p_3 e^2 \rangle.\end{aligned}$$

### Exercise 13.3.#VEEX

Consider the variable elimination algorithm in Figure 13.13 (page 432).

- a. Section 13.3 applies variable elimination to the query

$$\mathbf{P}(\text{Burglary} \mid \text{JohnCalls} = \text{true}, \text{MaryCalls} = \text{true}).$$

Perform the calculations indicated and check that the answer is correct.

- b. Count the number of arithmetic operations performed, and compare it with the number performed by the enumeration algorithm.
- c. Suppose a network has the form of a *chain*: a sequence of Boolean variables  $X_1, \dots, X_n$  where  $\text{Parents}(X_i) = \{X_{i-1}\}$  for  $i = 2, \dots, n$ . What is the complexity of computing  $\mathbf{P}(X_1 \mid X_n = \text{true})$  using enumeration? Using variable elimination?
- d. Prove that the complexity of running variable elimination on a polytree network is linear in the size of the tree for any variable ordering consistent with the network structure.

This question definitely helps students get a solid feel for variable elimination. Students

may need some help with the last part if they are to do it properly.

a.

$$\begin{aligned}
 P(B|j, m) &= \alpha P(B) \sum_e P(e) \sum_a P(a|b, e) P(j|a) P(m|a) \\
 &= \alpha P(B) \sum_e P(e) \left[ .9 \times .7 \times \begin{pmatrix} .95 & .29 \\ .94 & .001 \end{pmatrix} + .05 \times .01 \times \begin{pmatrix} .05 & .71 \\ .06 & .999 \end{pmatrix} \right] \\
 &= \alpha P(B) \sum_e P(e) \begin{pmatrix} .598525 & .183055 \\ .59223 & .0011295 \end{pmatrix} \\
 &= \alpha P(B) \left[ .002 \times \begin{pmatrix} .598525 \\ .183055 \end{pmatrix} + .998 \times \begin{pmatrix} .59223 \\ .0011295 \end{pmatrix} \right] \\
 &= \alpha \begin{pmatrix} .001 \\ .999 \end{pmatrix} \times \begin{pmatrix} .59224259 \\ .001493351 \end{pmatrix} \\
 &= \alpha \begin{pmatrix} .00059224259 \\ .0014918576 \end{pmatrix} \\
 &\approx \langle .284, .716 \rangle
 \end{aligned}$$

- b. Including the normalization step, there are 7 additions, 16 multiplications, and 2 divisions. The enumeration algorithm has two extra multiplications.
- c. To compute  $\mathbf{P}(X_1|X_n = \text{true})$  using enumeration, we have to evaluate two complete binary trees (one for each value of  $X_1$ ), each of depth  $n - 2$ , so the total work is  $O(2^n)$ . Using variable elimination, the factors never grow beyond two variables. For example, the first step is

$$\begin{aligned}
 \mathbf{P}(X_1|X_n = \text{true}) &= \alpha \mathbf{P}(X_1) \dots \sum_{x_{n-2}} P(x_{n-2}|x_{n-3}) \sum_{x_{n-1}} P(x_{n-1}|x_{n-2}) P(X_n = \text{true}|x_{n-1}) \\
 &= \alpha \mathbf{P}(X_1) \dots \sum_{x_{n-2}} P(x_{n-2}|x_{n-3}) \sum_{x_{n-1}} \mathbf{f}_{X_{n-1}}(x_{n-1}, x_{n-2}) \mathbf{f}_{X_n}(x_{n-1}) \\
 &= \alpha \mathbf{P}(X_1) \dots \sum_{x_{n-2}} P(x_{n-2}|x_{n-3}) \mathbf{f}_{\overline{X_{n-1} X_n}}(x_{n-2})
 \end{aligned}$$

The last line is isomorphic to the problem with  $n - 1$  variables instead of  $n$ ; the work done on the first step is a constant independent of  $n$ , hence (by induction on  $n$ , if you want to be formal) the total work is  $O(n)$ .

- d. Here we can perform an induction on the number of nodes in the polytree. The base case is trivial. For the inductive hypothesis, assume that any polytree with  $n$  nodes can be evaluated in time proportional to the size of the polytree (i.e., the sum of the CPT sizes). Now, consider a polytree with  $n + 1$  nodes. Any node ordering consistent with the topology will eliminate first some leaf node from this polytree. To eliminate any leaf node, we have to do work proportional to the size of its CPT. Then, *because the network is a polytree*, we are left with *independent* subproblems, one for each parent. Each subproblem takes total work proportional to the sum of its CPT sizes, so the total

work for  $n + 1$  nodes is proportional to the sum of CPT sizes.

### Exercise 13.3.#VELT

Assume we are running variable elimination, and we currently have the following three factors:

| $A$          | $B$          | $f_1(A, B)$ | $A$          | $C$          | $D$          | $f_2(A, C, D)$ | $B$          | $D$          | $f_3(B, D)$ |
|--------------|--------------|-------------|--------------|--------------|--------------|----------------|--------------|--------------|-------------|
| <i>true</i>  | <i>true</i>  | 0.1         | <i>true</i>  | <i>true</i>  | <i>true</i>  | 0.2            | <i>true</i>  | <i>true</i>  | 0.2         |
| <i>true</i>  | <i>false</i> | 0.5         | <i>true</i>  | <i>false</i> | <i>false</i> | 0.1            | <i>true</i>  | <i>false</i> | 0.2         |
| <i>false</i> | <i>true</i>  | 0.2         | <i>false</i> | <i>true</i>  | <i>true</i>  | 0.5            | <i>false</i> | <i>true</i>  | 0.5         |
| <i>false</i> | <i>false</i> | 0.5         | <i>false</i> | <i>true</i>  | <i>false</i> | 0.2            | <i>false</i> | <i>false</i> | 0.1         |
|              |              |             | <i>false</i> | <i>false</i> | <i>true</i>  | 0.5            |              |              |             |
|              |              |             | <i>false</i> | <i>false</i> | <i>false</i> | 0.2            |              |              |             |

The next step in the variable elimination is to eliminate  $B$ .

- Which factors will participate in the elimination process of  $B$ ?
- Perform the join over the factors that participate in the elimination of  $B$ . Your answer should be a table similar to the tables above, it is your job to figure out which variables participate and what the numerical entries are.
- Now perform the summation over  $B$  for the factor you obtained from the join and show the factor that results.

- a.  $f_1, f_3$

| $A$          | $B$          | $D$          | $f'_4(A, B, D)$         |
|--------------|--------------|--------------|-------------------------|
| <i>true</i>  | <i>true</i>  | <i>true</i>  | $0.1 \times 0.2 = 0.02$ |
| <i>true</i>  | <i>true</i>  | <i>false</i> | $0.1 \times 0.2 = 0.02$ |
| <i>true</i>  | <i>false</i> | <i>true</i>  | $0.5 \times 0.5 = 0.25$ |
| <i>true</i>  | <i>false</i> | <i>false</i> | $0.5 \times 0.1 = 0.05$ |
| <i>false</i> | <i>true</i>  | <i>true</i>  | $0.2 \times 0.2 = 0.04$ |
| <i>false</i> | <i>true</i>  | <i>false</i> | $0.2 \times 0.2 = 0.04$ |
| <i>false</i> | <i>false</i> | <i>true</i>  | $0.5 \times 0.5 = 0.25$ |
| <i>false</i> | <i>false</i> | <i>false</i> | $0.5 \times 0.1 = 0.05$ |

- b.

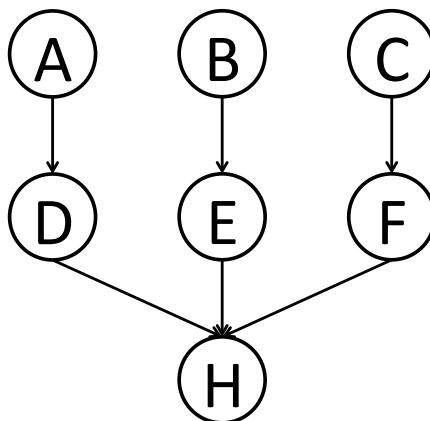
| $A$          | $B$          | $D$          | $f'_4(A, B, D)$         |
|--------------|--------------|--------------|-------------------------|
| <i>true</i>  | <i>true</i>  | <i>true</i>  | $0.1 \times 0.2 = 0.02$ |
| <i>true</i>  | <i>true</i>  | <i>false</i> | $0.1 \times 0.2 = 0.02$ |
| <i>true</i>  | <i>false</i> | <i>true</i>  | $0.5 \times 0.5 = 0.25$ |
| <i>true</i>  | <i>false</i> | <i>false</i> | $0.5 \times 0.1 = 0.05$ |
| <i>false</i> | <i>true</i>  | <i>true</i>  | $0.2 \times 0.2 = 0.04$ |
| <i>false</i> | <i>true</i>  | <i>false</i> | $0.2 \times 0.2 = 0.04$ |
| <i>false</i> | <i>false</i> | <i>true</i>  | $0.5 \times 0.5 = 0.25$ |
| <i>false</i> | <i>false</i> | <i>false</i> | $0.5 \times 0.1 = 0.05$ |

- c.

| $A$          | $D$          | $f_4(A, D)$          |
|--------------|--------------|----------------------|
| <i>true</i>  | <i>true</i>  | $0.02 + 0.25 = 0.27$ |
| <i>true</i>  | <i>false</i> | $0.02 + 0.05 = 0.07$ |
| <i>false</i> | <i>true</i>  | $0.04 + 0.25 = 0.29$ |
| <i>false</i> | <i>false</i> | $0.04 + 0.05 = 0.09$ |

### Exercise 13.3.#BPAH

For the Bayes net shown in Figure S13.42, consider the query  $P(A|h)$ , and the variable elimination ordering  $B, E, C, F, D$ .



**Figure S13.42** A Bayes network for Exercise 13.BPAH.

- a. In the table below fill in the factor generated at each step—we did the first row for you.

| Variable Eliminated | Factor Generated | Current Factors                                          |
|---------------------|------------------|----------------------------------------------------------|
| (none)              | (none)           | $P(A), P(B), P(C), P(D A), P(E B), P(F C), P(h D, E, F)$ |
| $B$                 | $f_1(E)$         | $P(A), P(C), P(D A), P(F C), P(h D, E, F), f_1(E)$       |
| $E$                 |                  |                                                          |
| $C$                 |                  |                                                          |
| $F$                 |                  |                                                          |
| $D$                 |                  |                                                          |

- b. Which is the largest factor generated? Assuming all variables are binary, how many entries does the corresponding table have?

| Variable Eliminated | Factor Generated | Current Factors                                          |
|---------------------|------------------|----------------------------------------------------------|
| (none)              | (none)           | $P(A), P(B), P(C), P(D A), P(E B), P(F C), P(h D, E, F)$ |
| $B$                 | $f_1(E)$         | $P(A), P(C), P(D A), P(F C), P(h D, E, F), f_1(E)$       |
| $E$                 | $f_2(h, D, F)$   | $P(A), P(C), P(D A), P(F C), f_2(h, D, F)$               |
| $C$                 | $f_3(F)$         | $P(A), P(D A), f_2(h, D, F), f_3(F)$                     |
| $F$                 | $f_4(h, D)$      | $P(A), P(D A), f_4(h, D)$                                |
| $D$                 | $f_5(h, A)$      | $P(A), f_5(h, A)$                                        |

a.

- b. The largest factor is  $f_2(h, D, F)$ , which has  $2^2 = 4$  entries.

**Exercise 13.3.#BNCM**

Investigate the complexity of exact inference in general Bayesian networks:

- Prove that any 3-SAT problem can be reduced to exact inference in a Bayesian network constructed to represent the particular problem and hence that exact inference is NP-hard. (*Hint:* Consider a network with one variable for each proposition symbol, one for each clause, and one for the conjunction of clauses.)
- The problem of counting the number of satisfying assignments for a 3-SAT problem is #P-complete. Show that exact inference is at least as hard as this.

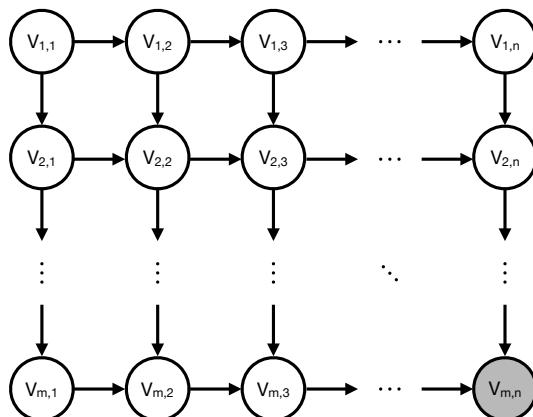
- a. Consider a 3-CNF formula  $C_1 \wedge \dots \wedge C_n$  with  $n$  clauses where each clause is a disjunct  $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$  of literals i.e., each  $\ell_{ij}$  is either  $P_k$  or  $\neg P_k$  for some atomic proposition  $P_1, \dots, P_m$ .

Construct a Bayesian network with a (boolean) variable  $S$  for the whole formula,  $C_i$  for each clause, and  $P_k$  for each atomic proposition. We will define parents and CPTs such that for any assignment to the atomic propositions,  $S$  is true if and only if the 3-CNF formula is true.

Atomic propositions have no parents, and are true with probability 0.5. Each clause  $C_i$  has as its parents the atomic propositions corresponding to the literals  $\ell_{i1}, \ell_{i2}$ , and  $\ell_{i3}$ ). The clause variable is true iff one of its literals is true. Note that this is a deterministic CPT. Finally,  $S$  has all the clause variables  $C_i$  as its parents, and if true if any only if all clause variables are true.

Notice that  $P(S = \text{True}) > 0$  if and only if the formula is satisfiable, and exact inference will answer this question.

- b. Using the same network as in part (a), notice that  $P(S = \text{True}) = s2^{-m}$  where  $s$  is the number of satisfying assignments to the atomic propositions  $P_1, \dots, P_m$ .



**Figure S13.43** A Bayes network for Exercise 13.LATT.

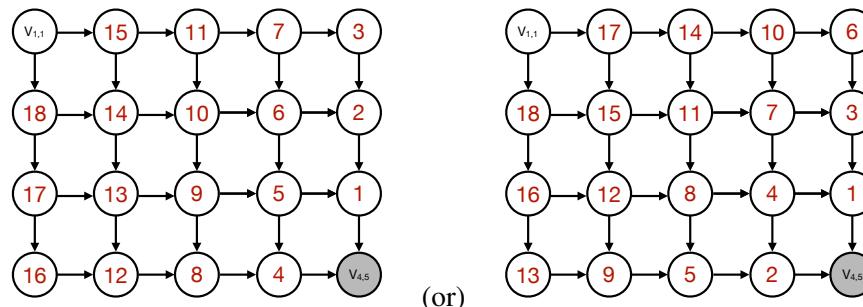
**Exercise 13.3.#LATT**

Consider doing inference in an  $m \times n$  lattice Bayes net, as shown in Figure S13.43. The network consists of  $mn$  binary variables  $V_{i,j}$ , and you have observed that  $V_{m,n} = +v_{m,n}$ .

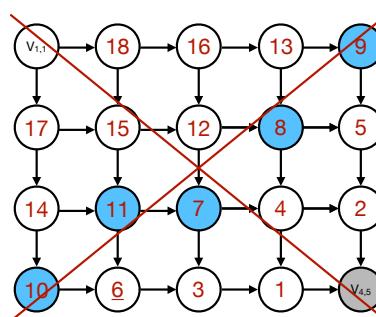
You wish to calculate  $P(V_{1,1} | +v_{m,n})$  using variable elimination. To maximize computational efficiency, you wish to use a variable elimination ordering for which the size of the largest generated factor is as small as possible.

- First consider the special case where  $m = 4$  and  $n = 5$ . What is the optimal elimination order?
- Now consider the general case (assume  $m > 2$  and  $n > 2$ ). What is the size of the largest factor generated under the most efficient elimination ordering?

- a. There are several possible orderings, here are two:



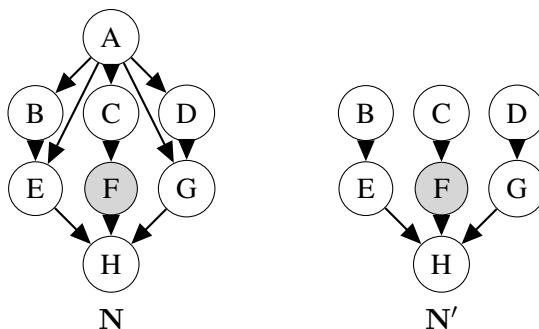
Minor variations on the orderings given above are possible. However, it's important to start near the same corner as the evidence variable and to never create a factor that involves more than 4 non-evidence variables. For example, the ordering shown below is suboptimal (eliminating node 6 will create a size  $2^5$  factor involving the five nodes highlighted in blue):



- b. The largest factor should have  $\min(m, n)$  variables and  $2^{\min(m,n)}$  table entries.

**Exercise 13.3.#CUTS**

- a. Consider answering  $P(H | +f)$  by variable elimination in the Bayes nets  $N$  and  $N'$  shown in Figure S13.44, where the elimination order is alphabetical and all variables



**Figure S13.44** Bayes nets for Exercise 13.CUTS.

are binary. How large are the largest factors made during variable elimination for  $N$  and  $N'$ ?

- b. Borrowing an idea from **cutset conditioning** in Chapter 6, we may be able to simplify variable elimination in  $N$  by instantiating some variables. Let's pick an **instantiation set** to pretend to observe, and then do variable elimination with these additional instantiations.

Consider the original query  $P(H \mid +f)$ , but let  $A$  be the instantiation set so  $A = a$  is observed. Now the query is  $H$  with observations  $F = +f, A = a$ .

- (i) What is the size of the largest factor made during variable elimination with the  $A = a$  instantiation?
- (ii) Given a Bayes net over  $n$  binary variables with  $k$  variables chosen for the instantiation set, how many instantiations of the set are there?
- c. Let's answer  $P(H \mid +f)$  by variable elimination with the instantiations of  $A$ .
  - (i) What quantity does variable elimination for  $P(H \mid +f)$  with the  $A = +a$  instantiation compute *without normalization*? That is, what is the last factor made by elimination?
  - (ii) Let  $I_+(H) = F(H, +a, +f)$  and  $I_-(H) = F(H, -a, +f)$  be the last factors made by variable elimination with instantiations  $A = +a$  and  $A = -a$ . How do we calculate the desired answer  $P(+h \mid +f)$ ?
- d. What is the time complexity of instantiated elimination, expressed in terms of  $n$  (the number of variables),  $k$  (the instantiation set size),  $f$  (the dimension of the largest factor made by elimination without instantiation), and  $i$  (the dimension of the largest factor made by elimination with instantiation)?

- a. (i) The largest factor made during variable elimination for  $N$  is  $f(B, C, D, E, G)$  after eliminating  $A$ . It has  $2^5 = 32$  entries.
- (ii) The largest factor made during variable elimination for  $N'$  is  $f(G, H, +f)$  after eliminating  $E$ . It has  $2^2 = 4$  entries.
- b. (i) The largest factor made during variable elimination with the  $A = a$  instantiation is

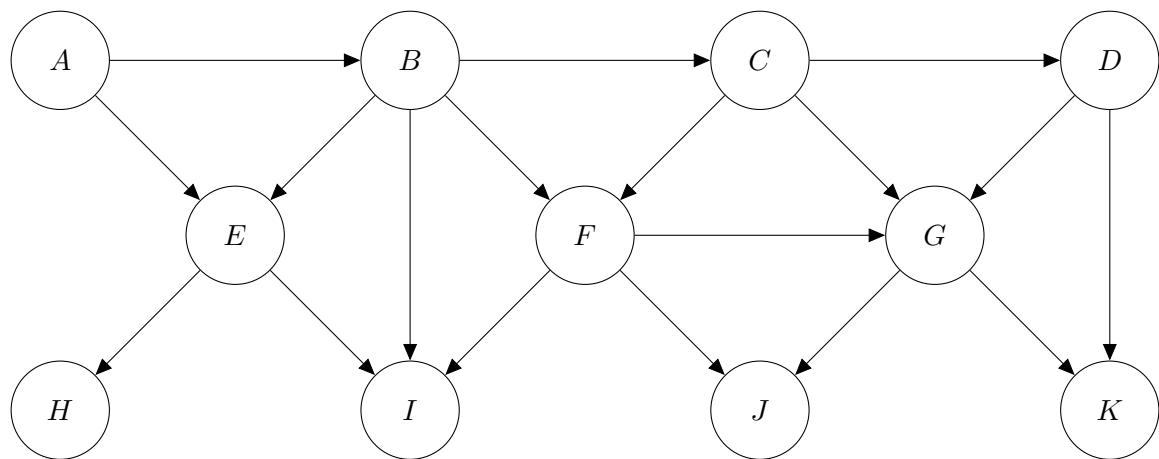
- $f(G, H, +f)$  as in  $N'$  in the previous question.
- (ii) For a selected instantiation set of  $k$  binary variables, there are  $2^k$  total instantiations of these variables.
  - c. (i) The last factor from variable elimination for  $P(H | +f)$  with  $A = +a$  is  $P(H, +a, +f)$ . At the end of variable elimination, the last factor is equal to the equivalent entries of the joint distribution with the eliminated variables summed out and the selected values of the evidence variables.
  - (ii)  $P(+h | +f) = \frac{I_+(+h) + I_-(+h)}{\sum_h I_+(h) + I_-(h)}$ . The last factors are the entries from the corresponding joint  $I_+(+h) = p(+h, +a, +f)$  and  $I_-(+h) = p(+h, -a, +f)$  and so on. By the law of total probability  $P(+h, +f) = P(+h, +a, +f) + P(+h, -a, +f)$ , so the joint of the original query and evidence can be computed from the instantiated elimination factors. For the conditional  $p(+h | +f) = P(+h, +f) / P(+f)$ , normalize by the sum over the query  $\sum_h p(h, +f) = \sum_h \sum_a p(h, a, +f) = f(+h, +a, +f) + f(+h, -a, +f) + f(-h, +a, +f) + f(-h, -a, +f)$  where the joint over the query and evidence is again computed from the law of total probability over  $A$ .
  - d. The time complexity of instantiated elimination is  $O(n \exp(i+k))$ . To carry out instantiated elimination, we have to do variable elimination  $\exp(k)$  times for all the settings of the instantiation set. Each of these eliminations takes time bounded by  $n \exp(i)$  as the largest factor is the most expensive to eliminate and there are  $n$  variables to eliminate. If the instantiation set is not too large and the size of the factors made by instantiation elimination are small enough this method can be exponentially faster than regular elimination. The catch is how to select the instantiation set.

### Exercise 13.3.#WMCX

The idea behind weighted model counting (WMC) is to use the machinery of SAT-solvers to perform probabilistic inference. As noted in Equation (12.9), the answer to a probabilistic query  $\mathbf{P}(X | \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e})$  is obtained by normalizing a sum of joint probabilities of the form  $P(x, \mathbf{e}, \mathbf{y})$  where  $\mathbf{y}$  denotes values of the hidden variables. Because  $X$ ,  $\mathbf{E}$ , and  $\mathbf{Y}$  together constitute all the variables in the Bayes net, the probabilities  $P(x, \mathbf{e}, \mathbf{y})$  for the possible worlds  $x, \mathbf{e}, \mathbf{y}$  are just products of conditional probabilities from the network.

A WMC algorithm is a particular kind of SAT-solver that, given a CNF sentence  $S$  with weights  $W(l)$  associated with each literal  $l$ , computes the sum of the weights of all satisfying assignments of  $S$ , where the weight of an assignment is the product of the weights of its constituent literals.

- a. Suppose  $S = [(A \vee B) \wedge (\neg B \vee \neg C) \wedge (\neg B \vee C)]$ ,  $W(A) = W(B) = W(C) = 0.7$ , and  $W(\neg A) = W(\neg B) = W(\neg C) = 0.3$ . Show that a WMC solver, given  $S$  and  $W$  as inputs, returns 0.21.
- b. In order to use the WMC algorithm for inference, you will need to develop a translation from a Bayes net model to a weighted CNF representation. For the purposes of this exercise, you may assume that all variables in the Bayes net are Boolean, so they can be mapped directly into propositional variables in the CNF sentence. In addition, for every variable  $V$  with parents  $\mathbf{U}$ , introduce a Boolean variable  $R_{v|\mathbf{u}}$  for each possible



**Figure S13.45** Bayes net for Exercise 13.BLNK.

value  $v$  of  $V$  and every combination of values  $\mathbf{u}$  for  $\mathbf{U}$ . Now, given a query variable  $X$  and evidence  $\mathbf{e}$ , show how to define a CNF sentence  $S$  and weights  $W(l)$  such that the WMC solver, given  $S$  and  $W$ , returns a number equal to  $P(X = \text{true}, \mathbf{e})$ . Prove that your definition yields the correct answer.

c. Implementation project:

- Identify a suitable Bayes net input format from one of the standard Bayes net packages, and find an efficient WMC solver.
- Implement the translation algorithm to convert any given Boolean-variable Bayes net, evidence set, and query into weighted CNF.
- Compare the computation time and memory requirements of exact inference in the Bayes net package versus running the WMC solver, for a range of networks, evidence sets, and queries. Comment on your results.

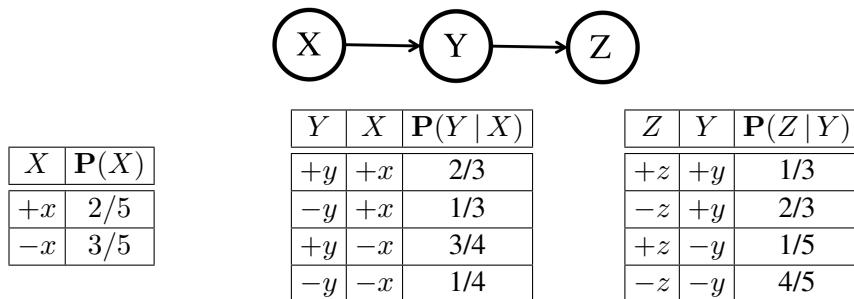
The discovery of a suitable implementation is left for interested readers.

## 13.4 Approximate Inference for Bayesian Networks

### Exercise 13.4.#BLNK

In the network in Figure S13.45, identify the Markov blanket of each variable.

**A** :  $B, E$ ; **B** :  $A, E, F, C$ ; **C** :  $B, F, G, D$ ; **D** :  $C, G, K$ ; **E** :  $A, B, H, I$ ; **F** :  $B, C, G, I, J$ ; **G** :  $C, D, F, J, K$ ; **H** :  $E$ ; **I** :  $B, E, F$ ; **J** :  $F, G$ ; **K** :  $D, G$

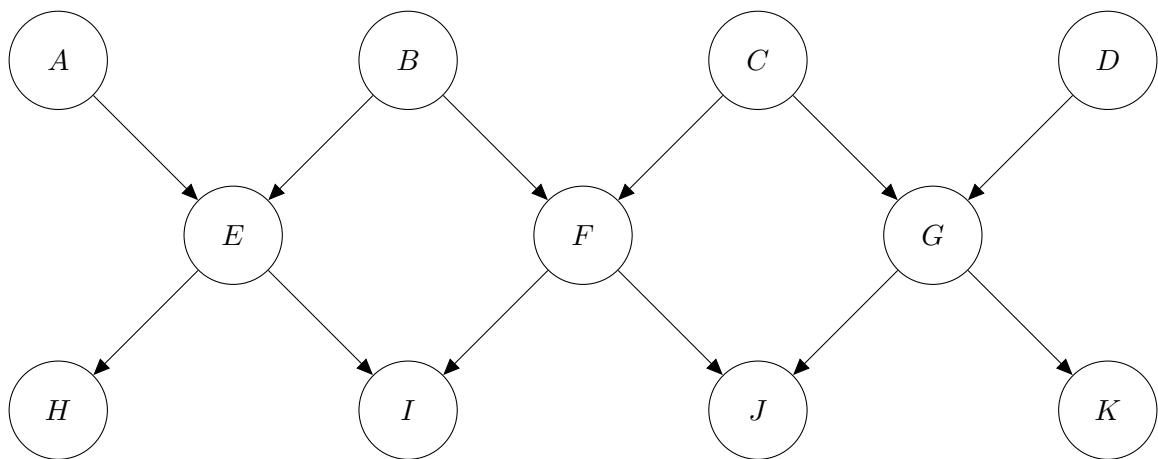


**Figure S13.46** Bayes net and distributions for Exercise 13.XYZS.

### Exercise 13.4.#XYZS

Assume the Bayes net, and the corresponding distributions over the variables in the Bayes net from Figure S13.46.

- Your task is now to estimate  $\mathbf{P}(+y | +x, +z)$  using rejection sampling. Below are some samples that have been produced by prior sampling (that is, the rejection stage in rejection sampling hasn't happened yet). Which of the following samples would be rejected by rejection sampling?
  - $+x, +y, +z$
  - $-x, +y, +z$
  - $-x, -y, +z$
  - $+x, -y, -z$
  - $+x, -y, +z$
- Using rejection sampling, give an estimate of  $\mathbf{P}(+y | +x, +z)$  from the five prior samples, or state why it cannot be computed.
- Using the following samples (which were generated using likelihood weighting), estimate  $\mathbf{P}(+y | +x, +z)$  using likelihood weighting, or state why it cannot be computed.
  - $+x, +y, +z$
  - $+x, -y, +z$
  - $+x, +y, +z$
- Given a sequence of samples, each specifying a value for all the variables in the network, how can one tell if the sequence *could* have been generated by Gibbs sampling?
  - Samples ii, iii, iv would be rejected.
  - Of the two remaining samples, one has  $y+$ , so the estimate is  $1/2$ .
  - The three weights are as follows:  $w_1 = w_3 = P(+x) * P(+z | +y) = 2/5 * 1/3 = 2/15$  and  $w_2 = P(+x) * P(+z | -y) = 2/5 * 1/5 = 2/25$ . Hence  $\mathbf{P}(+y | +x, +z) \approx (2w_1) / (2w_1 + w_2) = 10/13$ .



**Figure S13.47** Bayes net for Exercise 13.HIJK

- d.  $P(Z | X)$  is better, because evidence influences the choice of downstream variables, but not upstream variables, and  $X$  is a (grand)parent of  $Z$ .
- e. First, each sample can differ from the preceding sample in the value of only one variable. Second, each transition has to correspond to a non-zero value for the Gibbs distribution. This can be checked by seeing which variable changed (if any) and computing the Gibbs distribution for that variable given its Markov blanket.

#### Exercise 13.4.#HIJK

We are running Gibbs sampling in the Bayes net shown in Figure S13.47 for the query  $P(B, C | +a, +b, +c, +d, +e, +f, +g, +h, +i, +j, +k)$ . The current state is  $+a, +b, +c, +d, +e, +f, +g, +h, +i, +j, +k$ . Write out an expression for the Gibbs sampling distribution for each of  $A$ ,  $F$ , and  $K$  in terms of conditional probabilities available in the network.

This question requires identifying the Markov blanket and then a straightforward application of Equation (13.10). For  $A$ , we have

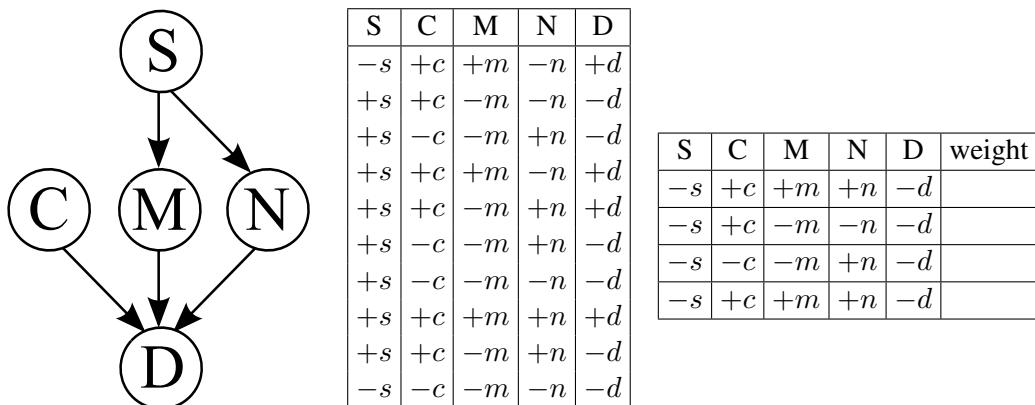
$$P(A | +b, +e) \propto P(+e | A, +b)P(+b).$$

For  $F$ , we have

$$P(F | +b, +c, +e, +g, +i, +j) \propto P(F | +b, +c)P(+i | +e, F)P(+j | F, +g).$$

For  $K$  we have just  $P(K | +g)$ .

#### Exercise 13.4.#DANC



**Figure S13.48** A Bayes net, CPT, and samples for Exercise 13.DANC.

You are an exobiologist, studying the wide range of life in the universe. You are also an avid dancer and have an excellent model of the way species invent dancing. The key variables are:

- Sound sensing (S): Whether or not a species has the ability to sense sound
- Cold climate (C): Whether or not the native planet of the species has cold weather
- Music (M): Whether or not the species invented music
- Non-verbal communication (N): Whether or not the species has any form of non-verbal communication

You model the relationships between these variables and **dancing (D)** using the Bayes net specified in Figure S13.48.

You want to know how likely it is for a dancing, sound-sensing species to invent music, according to this Bayes net. You decide to do inference via sampling. You use prior sampling to draw the samples in Figure S13.48.

- Based on rejection sampling using the samples above, what is the answer to your query  $P(-m \mid +d, +s)$ ?

While your sampling method has worked fairly well in many cases, for rare cases (like species that can't sense sound) your results are less accurate as rejection sampling rejects almost all of the data. You decide to use likelihood weighting instead. The conditional probabilities of the Bayes net are listed in Figure S13.49.

You now wish to compute the probability that a species that has no sound-sensing ( $-s$ ) or dancing ( $-d$ ) nonetheless has music ( $+m$ ), using likelihood weighting. I.e., you want  $P(+m \mid -s, -d)$ .

- You draw the samples in Figure S13.48, using likelihood weighting. For each of these samples, indicate its weight.

| C  | M  | N  | D  | $P(D   C, M, N)$ | S  | M  | $P(M   S)$ | S  | $P(S)$ |
|----|----|----|----|------------------|----|----|------------|----|--------|
| +c | +m | +n | +d | 0.9              | +s | +m | 0.8        | +s | 0.9    |
| +c | +m | +n | -d | 0.1              | +s | -m | 0.2        | -s | 0.1    |
| +c | +m | -n | +d | 0.8              | -s | +m | 0.1        |    |        |
| +c | +m | -n | -d | 0.2              | -s | -m | 0.9        |    |        |
| +c | -m | +n | +d | 0.8              |    |    |            |    |        |
| +c | -m | +n | -d | 0.2              |    |    |            |    |        |
| +c | -m | -n | +d | 0.2              |    |    |            |    |        |
| +c | -m | -n | -d | 0.8              |    |    |            |    |        |
| -c | +m | +n | +d | 0.8              |    |    |            |    |        |
| -c | +m | +n | -d | 0.2              |    |    |            |    |        |
| -c | +m | -n | +d | 0.5              |    |    |            |    |        |
| -c | +m | -n | -d | 0.5              |    |    |            |    |        |
| -c | -m | +n | +d | 0.6              |    |    |            |    |        |
| -c | -m | +n | -d | 0.4              |    |    |            |    |        |
| -c | -m | -n | +d | 0.1              |    |    |            |    |        |
| -c | -m | -n | -d | 0.9              |    |    |            |    |        |

Figure S13.49 CPTs for dancing aliens.

c. Compute the answer to your query,  $P(+m | -s, -d)$ , using likelihood weighting with these samples.

a. The probability estimate is 1/3. Simply find all the rows in the table above with  $+d$  and  $+s$ , and count the number of times  $-m$  occurs divided by the total number of such rows.

b.

| S  | C  | M  | N  | D  | weight      |
|----|----|----|----|----|-------------|
| -s | +c | +m | +n | -d | <b>0.01</b> |
| -s | +c | -m | -n | -d | <b>0.08</b> |
| -s | -c | -m | +n | -d | <b>0.04</b> |
| -s | +c | +m | +n | -d | <b>0.01</b> |

c.  $P(+m | -s, -d) \approx 1/7$ ; divide the sum of the weights corresponding to the  $+m$  rows by the total sum of the weights.

### Exercise 13.4.#MONY

Alice, Bob, Carol, and Dave are being given some money, but they have to share it in a very particular way:

- First, Alice will be given an integer number of dollars  $A$ , chosen uniformly at random between 1 and 100 (inclusive).

- Then Bob receives from Alice an integer number of dollars  $B$ , chosen uniformly at random between 1 and  $A$  (inclusive).
  - Then Carol receives from Bob an integer number of dollars  $C$ , chosen uniformly at random between 1 and  $B$  (inclusive).
  - Then Dave receives from Carol an integer number of dollars  $D$ , chosen uniformly at random between 1 and  $C$  (inclusive).
- Draw the Bayes net with variables  $A, B, C, D$  that corresponds to this process.
  - Write down an exact formula for  $P(B = b)$ , the probability that Bob receives  $b$  dollars. (Your answer may contain a summation.)
  - Show that  $P(A = a | B = b)$  is proportional to  $1/a$ .
  - Suppose Dave receives \$5 from Carol. You would like to estimate the probability  $P(A > 50 | D = 5)$ . Your first attempt uses rejection prior sampling, generating the following samples from the prior:

$$\begin{aligned} &(A = 52, B = 21, C = 10, D = 5) \\ &(A = 34, B = 21, C = 6, D = 3) \\ &(A = 96, B = 48, C = 12, D = 2) \\ &(A = 13, B = 12, C = 10, D = 1) \\ &(A = 54, B = 12, C = 11, D = 6) \\ &(A = 91, B = 32, C = 31, D = 29) \end{aligned}$$

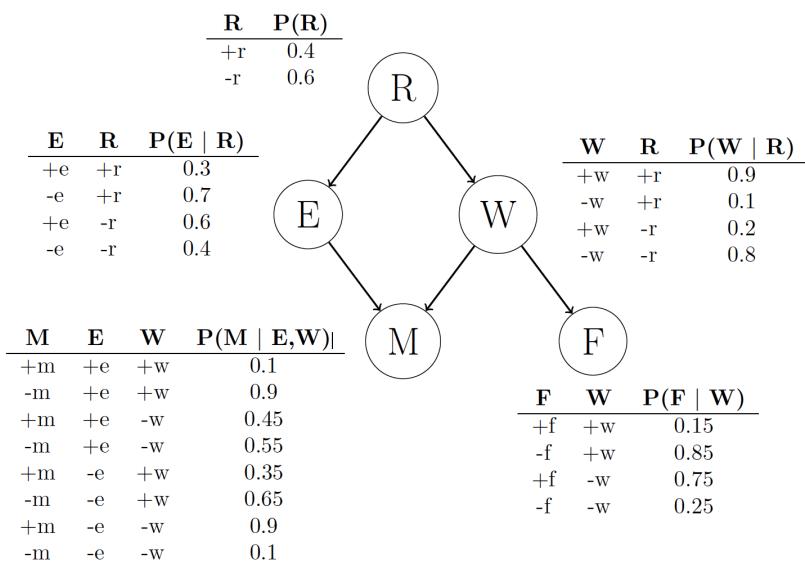
What is the estimated probability for the query based on these samples?

- Your next attempt uses likelihood weighting, generating the following samples:

$$\begin{aligned} &(S = 52, J = 21, B = 10, X = 5) \\ &(S = 34, J = 21, B = 4, X = 5) \\ &(S = 87, J = 12, B = 10, X = 5) \\ &(S = 41, J = 12, B = 5, X = 5) \\ &(S = 91, J = 32, B = 4, X = 5) \end{aligned}$$

Calculate the weights for each sample and find the estimated value for the query.

- The Bayes net is  $A \rightarrow B \rightarrow C \rightarrow D$ .
- $P(B = b) = \sum_{a=1}^{100} P(b | a)P(a) = \frac{1}{100} \sum_{a=b}^{100} P(b | a) = \frac{1}{100} \sum_{a=b}^{100} \frac{1}{a}$ . For the mathematically inclined, this summation has a closed-form solution, giving  $\frac{1}{100}(\psi(101) - \psi(b))$  where  $\psi$  is the digamma function.
- Using Bayes' rule,  $P(a | b) = \alpha P(b | a)P(a) = \alpha \frac{1}{100a}$  for  $b \leq a \leq 100$  and 0 otherwise.
- Only the first sample has  $D = 5$ . All other samples are rejected. In that sample,  $A > 50$ , so the estimated probability is 1.
- The weights are 0.1, 0.0, 0.1, 0.2, 0.0. The total weight of samples with  $A > 50$  is 0.2, out of a total weight for all samples of 0.4, so the estimate is 0.5.



**Figure S13.50** A Bayes net and CPTs for sampling.

### Exercise 13.4.#SMPL

Consider the Bayes net and corresponding probability tables shown in Figure S13.50.

Fill in the following table with the probabilities of drawing each respective sample given that we are using each of the following sampling techniques. For rejection sampling, we say that a sample has been drawn only if it is not rejected. You may leave your answer in the form of an expression such as  $.8 \cdot .4$  without multiplying it out. (Hint:  $P(f, m) = .181$ )

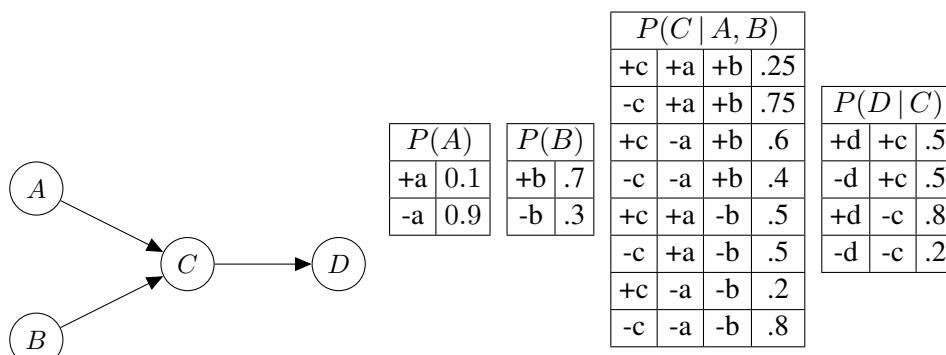
| $P(\text{sample} \text{method})$ | $(+r, +e, -w, +m, +f)$ | $(+r, -e, +w, -m, +f)$ |
|----------------------------------|------------------------|------------------------|
| prior sampling                   |                        |                        |
| rejection sampling               |                        |                        |
| likelihood weighting             |                        |                        |

| $P(\text{sample} \text{method})$ | $(+r, +e, -w, +m, +f)$                                                  | $(+r, -e, +w, -m, +f)$                                   |
|----------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------|
| prior sampling                   | $.4 \times .3 \times .1 \times .45 \times .75 = .00405$                 | $.4 \times .7 \times .9 \times .65 \times .15 = 0.02457$ |
| rejection sampling               | $\frac{P(+r, +e, -w, +m, +f)}{P(+m, +f)} = \frac{.00405}{.181} = .0224$ | 0                                                        |
| likelihood weighting             | $P(+r)P(+e r)P(-w r) = .4 \times .3 \times .1 = .012$                   | 0                                                        |

**Exercise 13.4.#RMVY**

Exercise 13.MRBL asks you to prove that removing an observed variable  $Y$  from a Bayes net has no effect on the posterior distribution of any variable  $X$  that is outside  $Y$ 's Markov blanket, provided that every variable in  $Y$ 's markov blanket is observed. In this question, we consider the effect of removing  $Y$  on inference algorithms that estimate the posterior for  $X$  from samples. What is the effect of removing  $Y$  on (i) rejection sampling and (ii) likelihood weighting? Consider both correctness and efficiency.

Because the posterior is unchanged, and both algorithms are consistent, they will still return the correct answers in the limit of infinitely many samples. Because  $Y$  is unobserved, it has no effect on which samples are rejected by rejection sampling, so for any finite sample size the answers will be identical. For importance sampling, we have to consider whether sampling  $Y$  affects the answer with a finite sample size. The answer is yes: suppose, for example, that we observe  $W = \text{true}$ , where  $W$  is a child of  $Y$ , and we know  $P(W = \text{true} | Y = \text{false}) = 0$ . This means that samples with  $Y = \text{false}$  get zero weight. It's possible (albeit unlikely) that all  $N$  samples happen to choose  $Y = \text{false}$ , so we'd get no useful information about the query at all. Thus, if  $Y$  is provably irrelevant to the query, as it is here, importance sampling should *avoid* sampling it because it can only increase the variance of the estimate.



**Figure S13.51** A Bayes net for sampling

**Exercise 13.4.#SMP2**

Assume you are given the Bayes net and the corresponding CPTs shown in Figure S13.51.

- Assume we receive evidence that  $A = +a$ . If we were to draw samples using rejection sampling, what is the expected fraction of samples that will be **rejected**?
- Next, assume we observed both  $A = +a$  and  $D = +d$ . What are the weights for the following samples under likelihood weighting sampling?

- (i)  $(+a, -b, +c, +d)$   
 (ii)  $(+a, -b, -c, +d)$   
 (iii)  $(+a, +b, -c, +d)$
- c. Given the samples in the previous question, estimate  $P(-b | +a, +d)$ .
- d. Assume we need to (approximately) answer two different inference queries for this graph:  $P(C | +a)$  and  $P(C | +d)$ . You are required to answer one query using likelihood weighting and one query using Gibbs sampling. Which query would you answer with which algorithm? Justify your answer.
- a. Since  $P(+a) = \frac{1}{10}$ , we would expect that only 10% of the samples could be saved. Therefore, expected 90% of the samples will be rejected.
- b. The weights are as follows:
- (i)  $(+a, -b, +c, +d)$ :  $P(+a) \cdot P(+d | +c) = 0.1 * 0.5 = 0.05$
  - (ii)  $(+a, -b, -c, +d)$ :  $P(+a) \cdot P(+d | -c) = 0.1 * 0.8 = 0.08$
  - (iii)  $(+a, +b, -c, +d)$ :  $P(+a) \cdot P(+d | -c) = 0.1 * 0.8 = 0.08$
- c.  $P(-b | +a, +d) = \frac{P(+a) \cdot P(+d | +c) + P(+a) \cdot P(+d | -c)}{P(+a) \cdot P(+d | +c) + 2 \cdot P(+a) \cdot P(+d | -c)} = \frac{0.05 + 0.08}{0.05 + 2 \cdot 0.08} = \frac{13}{21}.$
- d. Use likelihood weighting for  $P(C | +a)$  and Gibbs sampling for  $P(C | +d)$ . This is because likelihood weighting only takes upstream evidence into account when sampling. Therefore, Gibbs, which utilizes both upstream and downstream evidence, is more suited to the query  $P(C | +d)$  which has downstream evidence.

### Exercise 13.4.#PRSA

Consider the problem of generating a random sample from a specified distribution on a single variable. Assume you have a random number generator that returns a random number uniformly distributed between 0 and 1.

- a. Let  $X$  be a discrete variable with  $P(X = x_i) = p_i$  for  $i \in \{1, \dots, k\}$ . The **cumulative distribution** of  $X$  gives the probability that  $X \in \{x_1, \dots, x_j\}$  for each possible  $j$ . (See also Appendix A.1.) Explain how to calculate the cumulative distribution in  $O(k)$  time and how to generate a single sample of  $X$  from it. Can the latter be done in less than  $O(k)$  time?
- b. Now suppose we want to generate  $N$  samples of  $X$ , where  $N \gg k$ . Explain how to do this with an expected run time per sample that is *constant* (i.e., independent of  $k$ ).
- c. Now consider a continuous-valued variable with a parameterized distribution (e.g., Gaussian). How can samples be generated from such a distribution?
- d. Suppose you want to query a continuous-valued variable and you are using a sampling algorithm such as LIKELIHOODWEIGHTING to do the inference. How would you have to modify the query-answering process?

- a. To calculate the cumulative distribution of a discrete variable, we start from a vector representation  $p$  of the original distribution and a vector  $P$  of the same dimension. Then, we loop through  $i$ , adding up the  $p_i$  values as we go along and setting  $P_i$  to the running sum,  $\sum_{j=i}^i p_j$ . To sample from the distribution, we generate a random number  $r$  uniformly in  $[0, 1]$ , and then return  $x_i$  for the smallest  $i$  such that  $P_i \geq r$ . A naive way to find this is to loop through  $i$  starting at 1 until  $P_i \geq r$ . This takes  $O(k)$  time. A more efficient solution is binary search: start with the full range  $[1, k]$ , choose  $i$  at the midpoint of the range. If  $P_i < r$ , set the range from  $i$  to the upper bound, otherwise set the range from the lower bound to  $i$ . After  $O(\log k)$  iterations, we terminate when the bounds are identical or differ by 1.
- b. If we are generating  $N \gg k$  samples, we can afford to preprocess the cumulative distribution. The basic insight required is that if the original distribution were uniform, it would be possible to sample in  $O(1)$  time by returning  $\lceil kr \rceil$ . That is, we can index directly into the correct part of the range (analog random access, one might say) instead of searching for it. Now, suppose we divide the range  $[0, 1]$  into  $k$  equal parts and construct a  $k$ -element vector, each of whose entries is a list of all those  $i$  for which  $P_i$  is in the corresponding part of the range. The  $i$  we want is in the list with index  $\lceil kr \rceil$ . We retrieve this list in  $O(1)$  time and search through it in order (as in the naive implementation). Let  $n_j$  be the number of elements in list  $j$ . Then the expected runtime is given by

$$\sum_{j=1}^k n_j \cdot 1/k = 1/k \cdot \sum_{j=1}^k n_j = 1/k \cdot O(k) = O(1)$$

The variance of the runtime can be reduced by further subdividing any part of the range whose list contains more than some small constant number of elements.

- c. One way to generate a sample from a univariate Gaussian is to compute the discretized cumulative distribution (e.g., integrating by Taylor's rule) and use the algorithm described above. We can compute the table once and for all for the standard Gaussian (mean 0, variance 1) and then scale each sampled value  $z$  to  $\sigma z + \mu$ . If we had a closed-form, invertible expression for the cumulative distribution  $F(x)$ , we could sample exactly, simply by returning  $F^{-1}(r)$ . Unfortunately the Gaussian density is not exactly integrable. Now, the density  $\alpha x e^{-x^2/2}$  is exactly integrable, and there are cute schemes for using two samples and this density to obtain an exact Gaussian sample. We leave the details to the interested instructor.
- d. When querying a continuous variable using Monte carlo inference, an exact closed-form posterior cannot be obtained. Instead, one typically defines discrete ranges, returning a histogram distribution simply by counting the (weighted) number of samples in each range.

### Exercise 13.4.#RAIN

Consider the query  $\mathbf{P}(\text{Rain} \mid \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$  in Figure 13.15(a) (page 435) and how Gibbs sampling can answer it.

- a. How many states does the Markov chain have?
- b. Calculate the **transition matrix**  $\mathbf{Q}$  containing  $k(\mathbf{y} \rightarrow \mathbf{y}')$  for all  $\mathbf{y}, \mathbf{y}'$ .
- c. What does  $\mathbf{Q}^2$ , the square of the transition matrix, represent?
- d. What about  $\mathbf{Q}^n$  as  $n \rightarrow \infty$ ?
- e. Explain how to do probabilistic inference in Bayesian networks, assuming that  $\mathbf{Q}^n$  is available. Is this a practical way to do inference?

- a. There are two uninstantiated Boolean variables (*Cloudy* and *Rain*) and therefore four possible states.
- b. First, we compute the sampling distribution for each variable, conditioned on its Markov blanket.

$$\begin{aligned}\mathbf{P}(C|r, s) &= \alpha \mathbf{P}(C)\mathbf{P}(s|C)\mathbf{P}(r|C) \\ &= \alpha \langle 0.5, 0.5 \rangle \langle 0.1, 0.5 \rangle \langle 0.8, 0.2 \rangle = \alpha \langle 0.04, 0.05 \rangle = \langle 4/9, 5/9 \rangle\end{aligned}$$

$$\begin{aligned}\mathbf{P}(C|\neg r, s) &= \alpha \mathbf{P}(C)\mathbf{P}(s|C)\mathbf{P}(\neg r|C) \\ &= \alpha \langle 0.5, 0.5 \rangle \langle 0.1, 0.5 \rangle \langle 0.2, 0.8 \rangle = \alpha \langle 0.01, 0.20 \rangle = \langle 1/21, 20/21 \rangle\end{aligned}$$

$$\begin{aligned}\mathbf{P}(R|c, s, w) &= \alpha \mathbf{P}(R|c)\mathbf{P}(w|s, R) \\ &= \alpha \langle 0.8, 0.2 \rangle \langle 0.99, 0.90 \rangle = \alpha \langle 0.792, 0.180 \rangle = \langle 22/27, 5/27 \rangle\end{aligned}$$

$$\begin{aligned}\mathbf{P}(R|\neg c, s, w) &= \alpha \mathbf{P}(R|\neg c)\mathbf{P}(w|s, R) \\ &= \alpha \langle 0.2, 0.8 \rangle \langle 0.99, 0.90 \rangle = \alpha \langle 0.198, 0.720 \rangle = \langle 11/51, 40/51 \rangle\end{aligned}$$

Strictly speaking, the transition matrix is only well-defined for the variant of MCMC in which the variable to be sampled is chosen randomly. (In the variant where the variables are chosen in a fixed order, the transition probabilities depend on where we are in the ordering.) Now consider the transition matrix.

- Entries on the diagonal correspond to self-loops. Such transitions can occur by sampling *either* variable. For example,

$$k((c, r) \rightarrow (c, r)) = 0.5P(c|r, s) + 0.5P(r|c, s, w) = 17/27$$

- Entries where one variable is changed must sample that variable. For example,

$$k((c, r) \rightarrow (c, \neg r)) = 0.5P(\neg r|c, s, w) = 5/54$$

- Entries where both variables change cannot occur. For example,

$$k((c, r) \rightarrow (\neg c, \neg r)) = 0$$

This gives us the following transition matrix, where the transition is from the state given

by the row label to the state given by the column label:

$$\begin{array}{c|cccc} & (c, r) & (c, \neg r) & (\neg c, r) & (\neg c, \neg r) \\ \hline (c, r) & 17/27 & 5/54 & 5/18 & 0 \\ (c, \neg r) & 11/27 & 22/189 & 0 & 10/21 \\ (\neg c, r) & 2/9 & 0 & 59/153 & 20/51 \\ (\neg c, \neg r) & 0 & 1/42 & 11/102 & 310/357 \end{array}$$

- c.  $\mathbf{Q}^2$  represents the probability of going from each state to each state in two steps.
- d.  $\mathbf{Q}^n$  (as  $n \rightarrow \infty$ ) represents the long-term probability of being in each state starting in each state; for ergodic  $\mathbf{Q}$  these probabilities are independent of the starting state, so every row of  $\mathbf{Q}$  is the same and represents the posterior distribution over states given the evidence.
- e. We can produce very large powers of  $\mathbf{Q}$  with very few matrix multiplications. For example, we can get  $\mathbf{Q}^2$  with one multiplication,  $\mathbf{Q}^4$  with two, and  $\mathbf{Q}^{2^k}$  with  $k$ . Unfortunately, in a network with  $n$  Boolean variables, the matrix is of size  $2^n \times 2^n$ , so each multiplication takes  $O(2^{3n})$  operations.

### Exercise 13.4.#GIBP

This exercise explores the stationary distribution for Gibbs sampling methods.

- a. The convex composition  $[\alpha, q_1; 1 - \alpha, q_2]$  of  $q_1$  and  $q_2$  is a transition probability distribution that first chooses one of  $q_1$  and  $q_2$  with probabilities  $\alpha$  and  $1 - \alpha$ , respectively, and then applies whichever is chosen. Prove that if  $q_1$  and  $q_2$  are in detailed balance with  $\pi$ , then their convex composition is also in detailed balance with  $\pi$ . (Note: this result justifies a variant of GIBBS-ASK in which variables are chosen at random rather than sampled in a fixed sequence.)
- b. Prove that if each of  $q_1$  and  $q_2$  has  $\pi$  as its stationary distribution, then the sequential composition  $q = q_1 \circ q_2$  also has  $\pi$  as its stationary distribution.

- a. Supposing that  $q_1$  and  $q_2$  are in detailed balance we have:

$$\begin{aligned} \pi(\mathbf{x})(\alpha q_1(\mathbf{x} \rightarrow \mathbf{x}') + (1 - \alpha)q_2(\mathbf{x} \rightarrow \mathbf{x}')) \\ = \alpha \pi(\mathbf{x})q_1(\mathbf{x} \rightarrow \mathbf{x}') + (1 - \alpha)\pi(\mathbf{x})q_2(\mathbf{x} \rightarrow \mathbf{x}') \\ = \alpha \pi(\mathbf{x})q_1(\mathbf{x}' \rightarrow \mathbf{x}) + (1 - \alpha)\pi(\mathbf{x})q_2(\mathbf{x}' \rightarrow \mathbf{x}) \\ = \pi(\mathbf{x})(\alpha q_1(\mathbf{x}' \rightarrow \mathbf{x}) + (1 - \alpha)q_2(\mathbf{x}' \rightarrow \mathbf{x})) \end{aligned}$$

- b. The sequential composition is defined by

$$(q_1 \circ q_2)(\mathbf{x} \rightarrow \mathbf{x}') = \sum_{\mathbf{x}''} q_1(\mathbf{x} \rightarrow \mathbf{x}'')q_2(\mathbf{x}'' \rightarrow \mathbf{x}').$$

If  $q_1$  and  $q_2$  both have  $\pi$  as their stationary distribution, then:

$$\begin{aligned}\sum_{\mathbf{x}} \pi(\mathbf{x})(q_1 \circ q_2)(\mathbf{x} \rightarrow \mathbf{x}') &= \sum_{\mathbf{x}} \pi(\mathbf{x}) \sum_{\mathbf{x}''} q_1(\mathbf{x} \rightarrow \mathbf{x}'') q_2(\mathbf{x}'' \rightarrow \mathbf{x}') \\&= \sum_{\mathbf{x}''} q_2(\mathbf{x}'' \rightarrow \mathbf{x}') \sum_{\mathbf{x}} \pi(\mathbf{x}) q_1(\mathbf{x} \rightarrow \mathbf{x}'') \\&= \sum_{\mathbf{x}''} q_2(\mathbf{x}'' \rightarrow \mathbf{x}') \pi(\mathbf{x}'') \\&= \pi(\mathbf{x}')\end{aligned}$$

### Exercise 13.4.#MEHA

The **Metropolis–Hastings** algorithm is a member of the MCMC family; as such, it is designed to generate samples  $\mathbf{x}$  (eventually) according to target probabilities  $\pi(\mathbf{x})$ . (Typically we are interested in sampling from  $\pi(\mathbf{x}) = P(\mathbf{x} | \mathbf{e})$ .) Like simulated annealing, Metropolis–Hastings operates in two stages. First, it samples a new state  $\mathbf{x}'$  from a **proposal distribution**  $q(\mathbf{x}' | \mathbf{x})$ , given the current state  $\mathbf{x}$ . Then, it probabilistically accepts or rejects  $\mathbf{x}'$  according to the **acceptance probability**

$$\alpha(\mathbf{x}' | \mathbf{x}) = \min \left( 1, \frac{\pi(\mathbf{x}') q(\mathbf{x} | \mathbf{x}')}{\pi(\mathbf{x}) q(\mathbf{x}' | \mathbf{x})} \right).$$

If the proposal is rejected, the state remains at  $\mathbf{x}$ .

- a. Consider an ordinary Gibbs sampling step for a specific variable  $X_i$ . Show that this step, considered as a proposal, is guaranteed to be accepted by Metropolis–Hastings. (Hence, Gibbs sampling is a special case of Metropolis–Hastings.)
- b. Show that the two-step process above, viewed as a transition probability distribution, is in detailed balance with  $\pi$ .

- a. Because a Gibbs transition step is in detailed balance with  $\pi$ , we have that the acceptance probability is one:

$$\begin{aligned}\alpha(\mathbf{x}' | \mathbf{x}) &= \min \left( 1, \frac{\pi(\mathbf{x}') q(\mathbf{x} | \mathbf{x}')}{\pi(\mathbf{x}) q(\mathbf{x}' | \mathbf{x})} \right) \\&= 1\end{aligned}$$

since by definition of detailed balance we have

$$\pi(\mathbf{x}') q(\mathbf{x} | \mathbf{x}') = \pi(\mathbf{x}) q(\mathbf{x}' | \mathbf{x}).$$

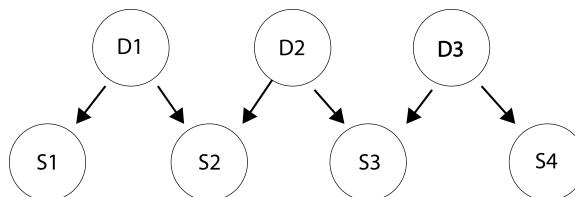
- b. To prove this in two stages. For  $\mathbf{x} \neq \mathbf{x}'$  the transition probability distribution is

$q(x' | x)\alpha(\mathbf{x}' | \mathbf{x})$  and we have:

$$\begin{aligned}\pi(\mathbf{x})q(x' | x)\alpha(\mathbf{x}' | \mathbf{x}) &= \pi(\mathbf{x})q(x' | x) \min\left(1, \frac{\pi(\mathbf{x}')q(\mathbf{x} | \mathbf{x}')}{\pi(\mathbf{x})q(\mathbf{x}' | \mathbf{x})}\right) \\ &= \min(\pi(\mathbf{x})q(x' | x), \pi(\mathbf{x}')q(\mathbf{x} | \mathbf{x}')) \\ &= \pi(\mathbf{x}')q(x | x') \min\left(\frac{\pi(\mathbf{x})q(\mathbf{x}' | \mathbf{x})}{\pi(\mathbf{x}')q(\mathbf{x} | \mathbf{x}')}, 1\right) \\ &\quad \pi(\mathbf{x}')q(x | x')\alpha(\mathbf{x} | \mathbf{x}')\end{aligned}$$

For  $\mathbf{x} = \mathbf{x}'$  the transition probability is some  $q'(x | x)$  which always satisfies the equation for detailed balance:

$$\pi(\mathbf{x})q'(x | x) = \pi(\mathbf{x}')q'(x | x).$$



**Figure S13.52** A 3-zigzag Bayes net.

### Exercise 13.4.#ZGZG

A  $k$ -zigzag network has  $k$  Boolean root variables and  $k + 1$  Boolean leaf variables, where root  $i$  is connected to leaves  $i$  and  $i + 1$ . Figure S13.52 shows an example for  $k = 3$ , where each  $D_i$  represents a Boolean disease variable and each  $S_j$  is a Boolean symptom variable.

- Does having symptom  $S_4$  affect the probability of disease  $D_1$ ? Why or why not?
- Using only conditional probabilities from the model, express the probability of having symptom  $S_1$  but not symptom  $S_2$ , given disease  $D_1$ .
- Can exact inference in a  $k$ -zigzag net can be done in time  $O(k)$ ? Explain.
- Suppose the values of all the symptom variables have been observed, and you would like to do Gibbs sampling on the disease variables (i.e., sample each variable given its Markov blanket). What is the largest number of non-evidence variables that have to be considered when sampling any particular disease variable? Explain your answer.
- Suppose  $k = 50$ . You would like to run Gibbs sampling for 1 million samples. Is it a good idea to precompute all the sampling distributions, so that when generating each individual sample no arithmetic operations are needed? Explain.
- A  $k$ -zigzag++ network is a  $k$ -zigzag network with two extra variables: one is a root connected to all the leaves and one is a leaf to which all the roots are connected. You would like to run Gibbs sampling for 1 million samples in a 50-zigzag++ network. Is it a good idea to precompute all the sampling distributions? Explain.

g. Returning to the  $k$ -zigzag network, let us assume that in every case the values of all symptom and disease variables are observed. This means that we can consider training a neural net to predict diseases from symptoms directly (rather than using a Bayes net and probabilistic inference), where the symptoms are inputs to the neural net and the diseases are the outputs. The issue is the internal connectivity of the neural net: to which diseases should the symptoms be connected? A neural-net expert opines that we can simply reverse the arrows in the figure, so that  $D_1$  is predicted only as a function  $f_1(S_1, S_2)$  and so on (The functions  $f_i$  will be learned from data, but the representation of  $f_i$  and the learning algorithm is not relevant here.) The reasoning is that argues that because  $D_1$  doesn't affect  $S_3$  and  $S_4$ , they are not needed for predicting  $D_1$ . Is the expert right? Explain.

- No.  $D_1$  is independent of its nondescendants (including  $S_4$ ) given its parents (empty set), i.e.,  $D_1$  is absolutely independent of  $S_4$ .
- $P(s_1 \wedge \neg s_2 | D_1) = P(s_1 | D_1)P(\neg s_2 | D_1) = P(s_1 | D_1) \sum_{d_2} P(d_2)P(\neg s_2 | D_1, d_2)$ .
- This is correct: the simplest explanation is that the net is a polytree and the CPT sizes are independent of  $k$ .
- The Markov blanket of a disease variable includes its parents, children, children's other parents; for a disease variable, the non-evidence variables in the Markov blanket are just the two neighboring disease variables.
- Yes. Precomputation requires space for 200 numbers and saves tens of millions of computations.
- No. The Markov blanket of every disease now includes all other diseases, and a table of  $2^{50}$  numbers is impossible to compute or store.
- No. While  $D_1$  is absolutely independent of  $S_3$ , it is not independent given  $S_1$  and  $S_2$ . The reason is that the value of  $S_3$  affects the probability of  $D_2$ ;  $D_2$  in turn can “explain away”  $S_2$ , thereby changing the probability of  $D_1$ . The same argument applies, by extension, to all symptoms. Thus, the neural network would need to be completely connected, leading to the need to train  $O(k)$   $k$ -ary functions.

## 13.5 Causal Networks

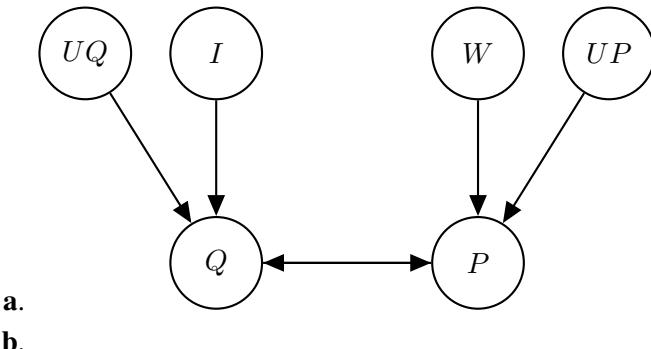
### Exercise 13.5.#DMND

In this problem, we will play economist. Consider four variables, price ( $P$ ), demand ( $Q$ ), income ( $I$ ), and wages ( $W$ ). More specifically, where  $Q$  is the quantity of household demand for a product  $A$ ,  $P$  is the unit price of product  $A$ ,  $I$  is household income,  $W$  is the wage rate for producing product  $A$ . Assume that income and wages are modeled separately (are independent). This example comes from Pearl (2000).

- Using your background knowledge of which variables influence each other when

changed, express this problem as a causal network. Include terms for unmodeled variables.

- Express this problem as a system of structural equations.
- Suppose we now change the price of the item, we  $do(p = x)$ . What is the new joint distribution,  $P(I, Q, W | do(p = x))$ ?



- 
- 

$$\begin{aligned} I &= f_I(UI) \\ W &= f_W(UW) \\ Q &= f_Q(I, P, UQ) \\ P &= f_P(W, Q, UP) \end{aligned}$$

- 
- 
- $P(I, Q, W | do(p = x)) = P(I)P(Q | I, = x)P(W)$

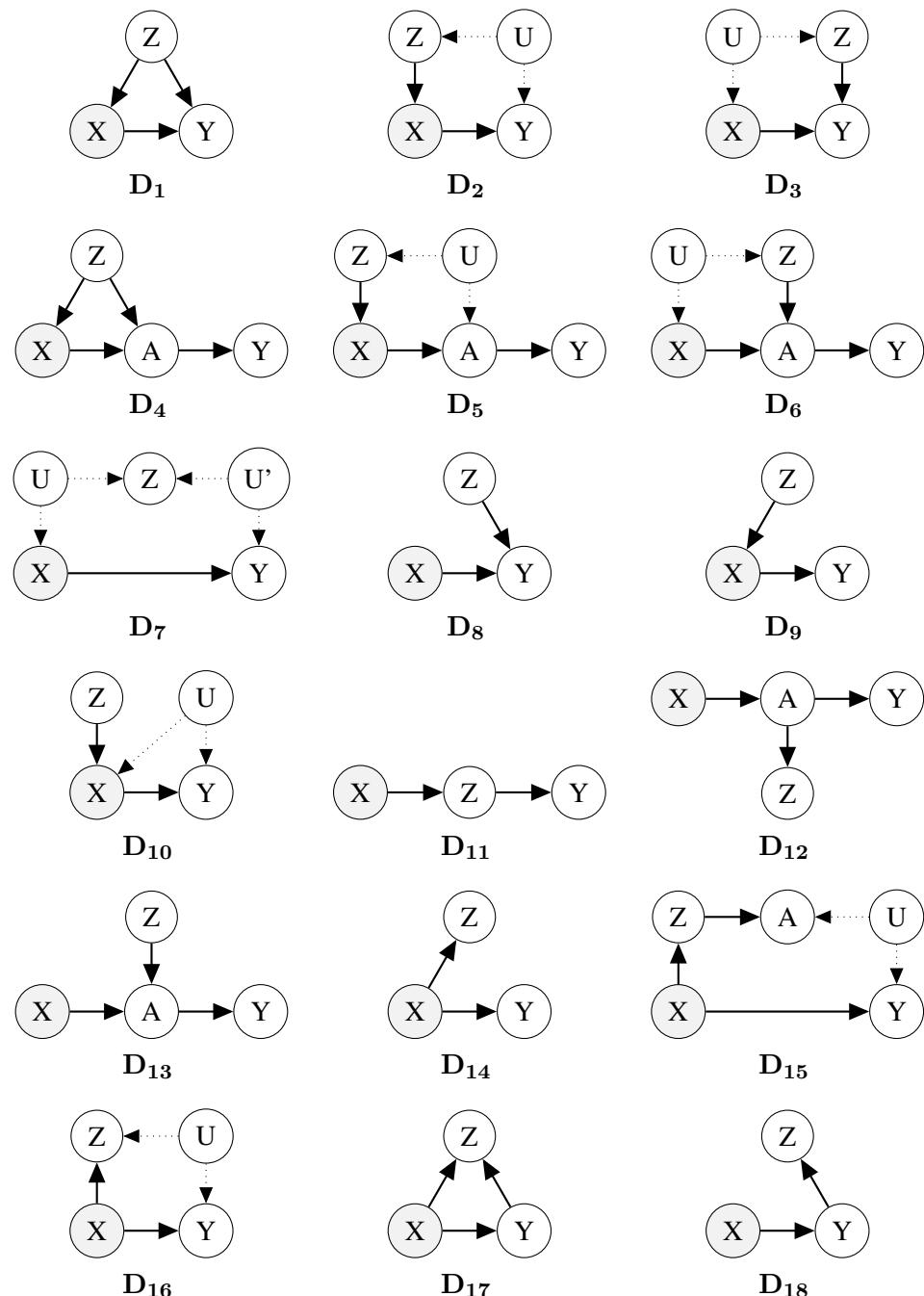
### Exercise 13.5.#EFCT

Prove that if there is no direct path between two variables,  $X$  and  $Y$  in a causal network, that there is no causal effect between them.

The proof follows from the Markov assumption of causal networks. For example in the sprinkler example in Figure 13.23 the action  $do(Sprinkler = true)$  has no effect on the variable  $Cloudy$ . Equation (13.20) shows that any effect that the sprinkler might have had is removed by the act of intervening—the variables are d-separated. That is,  $X \perp\!\!\!\perp Y | do(X)$ .

### Exercise 13.5.#DOOR

Scientific inquiry investigates the causal relation of variables; scientists study the effects of interventions. Causal networks provide a framework for such analysis by exactly specifying variables' putative relations. These networks, drawing on the work of Cinelli et. al (2021), can show us which treatments (the variable,  $X$ ) we can expect to measure the effect of (on the variable,  $Y$ ) when controlling for others. In particular, we're interested in whether a variable (here,  $Z$ ) would serve as a good, bad, or neutral control over the influence of  $X$  on  $Y$  in an observational setting (we have not intervened, as in a randomized control trial, to



**Figure S13.53** Causal networks for Exercise 13.DOOR. The grey circled nodes are observed treatments. Unmodeled variables,  $U$ , are shown to influence others with dotted lines.

$do(X)$ ); whether its inclusion in a regression analysis, for example, would reduce bias (close a confounding path; a good control), increase bias (open a confounding path; a bad control), or keep bias the same (neither open nor close a path; a neutral control).

That is, we want to measure the effect of  $X$  on  $Y$ . Thus, in which cases ( $D_1$  through  $D_{18}$ ) would keeping track of  $Z$  as well tell us more (good), less (bad), or nothing (neutral) about their relation? One way to measure this is the back-door criterion: does  $Z$  d-separate  $X$  from  $Y$ ? (Pearl (2000) goes more in depth on the specifics of the back-door criterion.)

- a. For each of the networks in Figure S13.53, state whether  $Z$  would serve as a good, bad, or neutral control. If a good or bad control state the reason (which confounding path was opened or closed or otherwise).
  - b. Now consider the networks in Figure S13.53 in an experimental setting, in which we  $do(X)$ . Is  $Z$  a good, bad, or neutral control? Why?
- a.  $D_1$  : Good; closes  $X \leftarrow Z \rightarrow Y$ .  
 $D_2$  : Good; closes  $X \leftarrow Z \leftarrow U \rightarrow Y$ .  
 $D_3$  : Good; closes  $X \leftarrow U \leftarrow Z \rightarrow Y$ .  
 $D_4$  : Good; closes  $X \leftarrow Z \leftarrow A \rightarrow Y$ .  
 $D_5$  : Good; closes  $X \leftarrow Z \leftarrow U \rightarrow A \rightarrow Y$ .  
 $D_6$  : Good; closes  $X \leftarrow U \rightarrow Z \rightarrow A \rightarrow Y$ .  
 $D_7$  : Bad; opens  $X \leftarrow U \rightarrow Z \leftarrow U' \rightarrow Y$ .  
 $D_8$  : Neutral.  
 $D_9$  : Neutral.  
 $D_{10}$  : Bad; opened  $Z \rightarrow X \rightarrow Y$ .  
 $D_{11}$  : Bad; closes  $X \rightarrow Z \rightarrow Y$ , what we are trying to measure.  
 $D_{12}$  : Bad; controls  $X \rightarrow A \rightarrow Y$ , what we are trying to measure.  
 $D_{13}$  : Neutral.  
 $D_{14}$  : Neutral.  
 $D_{15}$  : Neutral.  
 $D_{16}$  : Bad; opens  $X \rightarrow Z \leftarrow U \rightarrow Y$ .  
 $D_{17}$  : Bad; opens  $X \rightarrow Z \leftarrow Y$ .  
 $D_{18}$  : Bad; opens a (virtual path) from the unmodeled variable which influences  $Y$ ,  $X \rightarrow Y \leftarrow U_y$ .
- b. In all of the networks except  $D_{16,17,18}$  and  $D_{11,12}$  (in which  $Z$  is an effect of  $X$  and follow the same reason as in (a)),  $Z$  becomes a neutral control because by Equation (13.20) the back-door paths are removed by intervening.

# EXERCISES 14

## PROBABILISTIC REASONING OVER TIME

### 14.1 Time and Uncertainty

#### Exercise 14.1.#AUGM

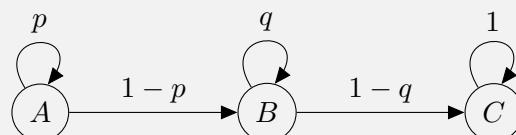
Show that any second-order Markov process can be rewritten as a first-order Markov process with an augmented set of state variables. Can this always be done *parsimoniously*, i.e., without increasing the number of parameters needed to specify the transition model?

For each variable  $U_t$  that appears as a parent of a variable  $X_{t+2}$ , define an auxiliary variable  $U_{t+1}^{old}$ , such that  $U_t$  is parent of  $U_{t+1}^{old}$  and  $U_{t+1}^{old}$  is a parent of  $X_{t+2}$ . This gives us a first-order Markov model. To ensure that the joint distribution over the original variables is unchanged, we keep the CPT for  $X_{t+2}$  unchanged except for the new parent name, and we require that  $\mathbf{P}(U_{t+1}^{old}|U_t)$  is an identity mapping, i.e., the child has the same value as the parent with probability 1. Since the parameters in this model are fixed and known, there is no effective increase in the number of free parameters in the model.

### 14.2 Inference in Temporal Models

#### Exercise 14.2.#OBJT

Suppose that an object is moving according to the following transition model:



Here,  $0 < p < 1$  and  $0 < q < 1$  are arbitrary probabilities. At time 0, the object is known to be in state  $A$ .

- What is the probability that the object is in  $A$  at time  $n \geq 0$ ?
- What is the probability that the object first reaches  $B$  at time  $n \geq 1$ ?
- What is the probability that the object is in  $B$  at time  $n \geq 1$ ?
- What is the probability that the object first reaches  $C$  at time  $n \geq 2$ ?

- e. What is the probability that the object is in  $C$  at time  $n \geq 2$ ?

We define  $\beta(n)$  and  $\gamma(n)$  as the probability the object first reaches  $B$  and  $C$ , respectively at time  $n$ . We define  $a(n), b(n), c(n)$  as the probability that the object is in  $A, B, C$ , respectively at time  $n$ .

- a. For the object to be in  $A$  at time  $n$ , it must have stayed in  $A$  for  $n$  steps, which occurs with probability  $a(n) \equiv p^n$ .
- b. For the object to first reach  $B$  at time  $n$ , it must have stayed in  $A$  for  $n - 1$  steps, then transitioned to  $B$ . This occurs with probability  $\beta(n) \equiv a(n-1) \cdot (1-p) = p^{n-1}(1-p)$ .
- c. For the object to be in  $B$  at time  $n$ , it must have first reached  $B$  at time  $i$  for some  $1 \leq i \leq n$ , then stayed there for  $n - i$  steps. Summing over all values of  $i$  gives

$$\begin{aligned} b(n) &\equiv \sum_{i=1}^n \beta(i) \cdot q^{n-i} \\ &= \sum_{i=1}^n p^{i-1}(1-p)q^{n-i} \\ &= \frac{(1-p)q^n}{p} \sum_{i=1}^n \left(\frac{p}{q}\right)^i \\ &= \frac{(1-p)q^n}{p} \cdot \frac{p}{q} \cdot \frac{1 - \left(\frac{p}{q}\right)^n}{1 - \frac{p}{q}} \\ &= (1-p) \frac{q^n - p^n}{q - p} \end{aligned}$$

where the simplifications in the last two lines assume  $p \neq q$ .

- d. For the object to first reach  $C$  at time  $n$ , it must have been in  $B$  at time  $n - 1$ , then transitioned to  $C$ . This occurs with probability

$$\gamma(n) \equiv b(n-1) \cdot (1-q) = (1-p) \frac{q^{n-1} - p^{n-1}}{q - p} (1-q).$$

- e. For the object to be in  $C$  at time  $n$ , it must not be in  $A$  or  $B$  at time  $n$ . This occurs with probability

$$1 - a(n) - b(n) = 1 - p^n - (1-p) \frac{q^n - p^n}{q - p}.$$

Alternatively, for the object to be in  $C$  at time  $n$ , it must have first reached  $C$  at time  $i$  for some  $2 \leq i \leq n$ , then stayed there for  $n - i$  steps. Note that we can equivalently range over  $1 \leq i \leq n$  for computational convenience, since  $\gamma(1) = 0$ . Summing over

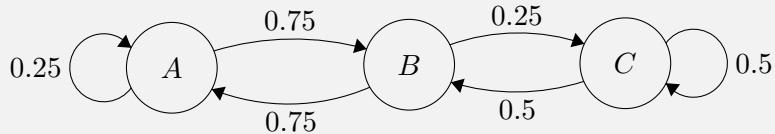
all values of  $i$  gives

$$\begin{aligned}
 c(n) &= \sum_{i=2}^n \gamma(i) \cdot 1^{n-i} \\
 &= \sum_{i=1}^n \gamma(i) \cdot 1^{n-i} \\
 &= \sum_{i=1}^n (1-p) \frac{q^{i-1} - p^{i-1}}{q-p} (1-q) \\
 &= \frac{(1-p)(1-q)}{q-p} \left( \frac{1-q^n}{1-q} - \frac{1-p^n}{1-p} \right) \\
 &= \frac{(1-p)(1-q^n) - (1-q)(1-p^n)}{q-p},
 \end{aligned}$$

which is equivalent to the previous expression.

### Exercise 14.2.#STAD

Consider a Markov chain with 3 states and transition probabilities as shown below:



Compute the stationary distribution. That is, compute  $P_\infty(A), P_\infty(B), P_\infty(C)$ .

$$P_\infty(A) = 0.4, P_\infty(B) = 0.4, P_\infty(C) = 0.2$$

### Exercise 14.2.#CONV

In this exercise, we examine what happens to the probabilities in the umbrella world in the limit of long time sequences.

- a. Suppose we observe an unending sequence of days on which the umbrella appears. Show that, as the days go by, the probability of rain on the current day increases monotonically toward a fixed point. Calculate this fixed point.
- b. Now consider *forecasting* further and further into the future, given just the first two umbrella observations. First, compute the probability  $P(r_{2+k}|u_1, u_2)$  for  $k = 1 \dots 20$  and plot the results. You should see that the probability converges towards a fixed point. Prove that the exact value of this fixed point is 0.5.

- a. For all  $t$ , we have the filtering formula

$$\mathbf{P}(R_t|u_{1:t}) = \alpha \mathbf{P}(u_t|R_t) \sum_{R_{t-1}} \mathbf{P}(R_t|R_{t-1}) P(R_{t-1}|u_{1:t-1}).$$

At the fixed point, we additionally expect that  $\mathbf{P}(R_t|u_{1:t}) = \mathbf{P}(R_{t-1}|u_{1:t-1})$ . Let the fixed-point probabilities be  $\langle \rho, 1 - \rho \rangle$ . This provides us with a system of equations:

$$\begin{aligned}\langle \rho, 1 - \rho \rangle &= \alpha \langle 0.9, 0.2 \rangle \langle 0.7, 0.3 \rangle \rho + \langle 0.3, 0.7 \rangle (1 - \rho) \\ &= \alpha \langle 0.9, 0.2 \rangle (\langle 0.4\rho, -0.4\rho \rangle + \langle 0.3, 0.7 \rangle) \\ &= \frac{1}{0.9(0.4\rho + 0.3) + 0.2(-0.4\rho + 0.7)} \langle 0.9, 0.2 \rangle (\langle 0.4\rho, -0.4\rho \rangle + \langle 0.3, 0.7 \rangle)\end{aligned}$$

Solving this system, we find that  $\rho \approx 0.8933$ .

- b. The probability converges to  $\langle 0.5, 0.5 \rangle$  as illustrated in Figure S14.1. This convergence makes sense if we consider a fixed-point equation for  $\mathbf{P}(R_{2+k}|U_1, U_2)$ :

$$\begin{aligned}\mathbf{P}(R_{2+k}|U_1, U_2) &= \langle 0.7, 0.3 \rangle P(r_{2+k-1}|U_1, U_2) + \langle 0.3, 0.7 \rangle P(\neg r_{2+k-1}|U_1, U_2) \\ \mathbf{P}(r_{2+k}|U_1, U_2) &= 0.7P(r_{2+k-1}|U_1, U_2) + 0.3(1 - P(r_{2+k-1}|U_1, U_2)) \\ &= 0.4P(r_{2+k-1}|U_1, U_2) + 0.3\end{aligned}$$

That is,  $P(r_{2+k}|U_1, U_2) = 0.5$ .

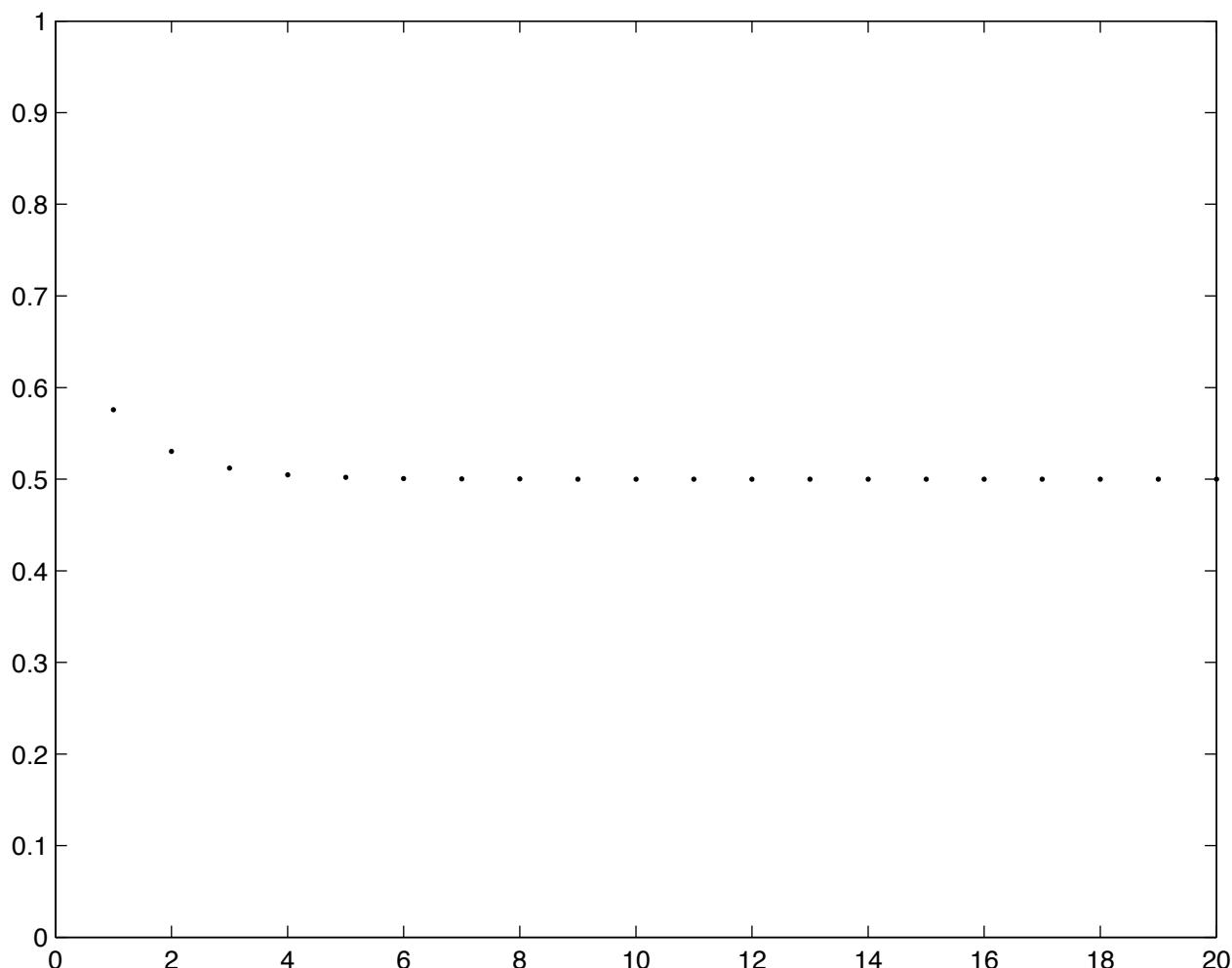
Notice that the fixed point does not depend on the initial evidence.

### Exercise 14.2.#LIKELIHOOD

Computing the evidence likelihood  $L_{1:t} = P(\mathbf{e}_{1:t})$  in a temporal sequence can be done using a recursive computation similar to the filtering algorithm. Show that the **likelihood message**  $\ell_{1:t}(\mathbf{X}_t) = \mathbf{P}(\mathbf{X}_t, \mathbf{e}_{1:t})$  satisfies the same recursive relationship as the filtering message; that is,

$$\ell_{1:t+1} = \text{FORWARD}(\ell_{1:t}, \mathbf{e}_{t+1}).$$

$$\begin{aligned}\ell_{1:t+1} &= \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t+1}) \\ &= \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{e}_{1:t}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{x_t} \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}, \mathbf{x}_t) \mathbf{P}(\mathbf{x}_t | \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{e}_{1:t}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{x_t} \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \mathbf{P}(\mathbf{x}_t, \mathbf{e}_{1:t}) \\ &= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{x_t} \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \ell_{1:t}\end{aligned}$$



**Figure S14.1** A graph of the probability of rain as a function of time, forecast into the future.

### Exercise 14.2.#ISLE

This exercise develops a space-efficient variant of the forward–backward algorithm described in Figure 14.4 (page 470). We wish to compute  $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$  for  $k = 1, \dots, t$ . This will be done with a divide-and-conquer approach.

- a. Suppose, for simplicity, that  $t$  is odd, and let the halfway point be  $h = (t + 1)/2$ . Show that  $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$  can be computed for  $k = 1, \dots, h$  given just the initial forward message  $\mathbf{f}_{1:0}$ , the backward message  $\mathbf{b}_{h+1:t}$ , and the evidence  $\mathbf{e}_{1:h}$ .
- b. Show a similar result for the second half of the sequence.
- c. Given the results of (a) and (b), a recursive divide-and-conquer algorithm can be constructed by first running forward along the sequence and then backward from the end,

storing just the required messages at the middle and the ends. Then the algorithm is called on each half. Write out the algorithm in detail.

- d. Compute the time and space complexity of the algorithm as a function of  $t$ , the length of the sequence. How does this change if we divide the input into more than two pieces?

This exercise develops the Island algorithm for smoothing in DBNs (Binder *et al.*, 1997).

- a. The chapter shows that  $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$  can be computed as

$$\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t}) = \alpha \mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t} | \mathbf{X}_k) = \alpha \mathbf{f}_{1:k} \mathbf{b}_{k+1:t}$$

The forward recursion (Equation 15.3) shows that  $\mathbf{f}_{1:k}$  can be computed from  $\mathbf{f}_{1:k-1}$  and  $\mathbf{e}_k$ , which can in turn be computed from  $\mathbf{f}_{1:k-2}$  and  $\mathbf{e}_{k-1}$ , and so on down to  $\mathbf{f}_{1:0}$  and  $\mathbf{e}_1$ . Hence,  $\mathbf{f}_{1:k}$  can be computed from  $\mathbf{f}_{1:0}$  and  $\mathbf{e}_{1:k}$ . The backward recursion (Equation 15.7) shows that  $\mathbf{b}_{k+1:t}$  can be computed from  $\mathbf{b}_{k+2:t}$  and  $\mathbf{e}_{k+1}$ , which in turn can be computed from  $\mathbf{b}_{k+3:t}$  and  $\mathbf{e}_{k+2}$ , and so on up to  $\mathbf{b}_{h+1:t}$  and  $\mathbf{e}_h$ . Hence,  $\mathbf{b}_{k+1:t}$  can be computed from  $\mathbf{b}_{h+1:t}$  and  $\mathbf{e}_{k+1:h}$ . Combining these two, we find that  $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$  can be computed from  $\mathbf{f}_{1:0}$ ,  $\mathbf{b}_{h+1:t}$ , and  $\mathbf{e}_{1:h}$ .

- b. The reasoning for the second half is essentially identical: for  $k$  between  $h$  and  $t$ ,  $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$  can be computed from  $\mathbf{f}_{1:h}$ ,  $\mathbf{b}_{t+1:t}$ , and  $\mathbf{e}_{h+1:t}$ .
- c. The algorithm takes 3 arguments: an evidence sequence, an initial forward message, and a final backward message. The forward message is propagated to the halfway point and the backward message is propagated backward. The algorithm then calls itself recursively on the two halves of the evidence sequence with the appropriate forward and backward messages. The base case is a sequence of length 1 or 2.
- d. At each level of the recursion the algorithm traverses the entire sequence, doing  $O(t)$  work. There are  $O(\log_2 t)$  levels, so the total time is  $O(t \log_2 t)$ . The algorithm does a depth-first recursion, so the total space is proportional to the depth of the stack, i.e.,  $O(\log_2 t)$ . With  $n$  islands, the recursion depth is  $O(\log_n t)$ , so the total time is  $O(t \log_n t)$  but the space is  $O(n \log_n t)$ .

### Exercise 14.2.#VITE

On page 471, we outlined a flawed procedure for finding the most likely state sequence, given an observation sequence. The procedure involves finding the most likely state at each time step, using smoothing, and returning the sequence composed of these states. Show that, for some temporal probability models and observation sequences, this procedure returns an impossible state sequence (i.e., the posterior probability of the sequence is zero).

This is a very good exercise for deepening intuitions about temporal probabilistic reasoning. First, notice that the impossibility of the sequence of most likely states cannot come from an impossible observation because the smoothed probability at each time step includes the evidence likelihood at that time step as a factor. Hence, the impossibility of a sequence must

arise from an impossible transition. Now consider such a transition from  $X_k = i$  to  $X_{k+1} = j$  for some  $i, j, k$ . For  $X_{k+1} = j$  to be the most likely state at time  $k + 1$ , even though it cannot be reached from the most likely state at time  $k$ , we can simply have an  $n$ -state system where, say, the smoothed probability of  $X_k = i$  is  $(1 + (n - 1)\epsilon)/n$  and the remaining states have probability  $(1 - \epsilon)/n$ . The remaining states all transition deterministically to  $X_{k+1} = j$ . From here, it is a simple matter to work out a specific model that behaves as desired.

## 14.3 Hidden Markov Models

### Exercise 14.3.#HMM

Equation (14.12) describes the filtering process for the matrix formulation of HMMs. Give a similar equation for the calculation of likelihoods, which was described generically in Equation (14.7).

The propagation of the  $\ell$  message is identical to that for filtering:

$$\ell_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \ell_{1:t}$$

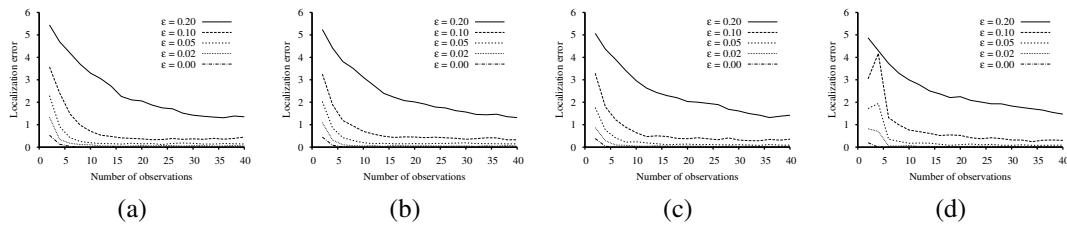
Since  $\ell$  is a column vector, each entry  $\ell_i$  of which gives  $P(X_t = i, \mathbf{e}_{1:t})$ , the likelihood is obtained simply by summing the entries:

$$L_{1:t} = P(\mathbf{e}_{1:t}) = \sum_i \ell_i .$$

### Exercise 14.3.#VACS

Consider the vacuum worlds of Figure 4.18 (perfect sensing) and Figure 14.7 (noisy sensing). Suppose that the robot receives an observation sequence such that, with perfect sensing, there is exactly one possible location it could be in. Is this location necessarily the most probable location under noisy sensing for sufficiently small noise probability  $\epsilon$ ? Prove your claim or find a counterexample.

Let  $\ell$  be the single possible location under deterministic sensing. Certainly, as  $\epsilon \rightarrow 0$ , we expect intuitively that  $P(X_t = \ell | e_{1:t}) \rightarrow 1$ . If we assume that all reachable locations are equally likely to be reached under the uniform motion model, then the claim that  $\ell$  is the most likely location under noisy sensing follows immediately: any other location must entail at least one sensor discrepancy—and hence a probability penalty factor of  $\epsilon$ —on every path reaching it in  $t - 1$  steps, otherwise it would be logically possible in the deterministic setting. The assumption is incorrect, however: if the neighborhood graph has outdegree  $k$ , the probability of reaching any two locations could differ by a factor of  $O(k^t)$ . If we set  $\epsilon$  smaller than this, the claim still holds. But for any fixed  $\epsilon$ , there are neighborhood graphs and observation sequences such that the claim may be false for sufficiently large  $t$ . Essentially, if  $t - 1$  steps of random movement are much more likely to reach  $m$  than  $\ell$ —e.g., if  $\ell$  is at



**Figure S14.2** Graphs showing the expected localization error as a function of time, for bias values of 1.0 (unbiased), 2.0, 5.0, 10.0.

the end of a long tunnel of length exactly  $t - 1$ —then that can outweigh the cost of a sensor error or two. Notice that this argument requires an environment of unbounded size; for any bounded environment, we can bound the reachability ratio and set  $\epsilon$  accordingly.

### Exercise 14.3.#HMMR

In Section 14.3.2, the prior distribution over locations is uniform and the transition model assumes an equal probability of moving to any neighboring square. What if those assumptions are wrong? Suppose that the initial location is actually chosen uniformly from the northwest quadrant of the room and the *Move* action actually tends to move southeast. Keeping the HMM model fixed, explore the effect on localization and path accuracy as the southeasterly tendency increases, for different values of  $\epsilon$ .

This exercise is an important one: it makes very clear the difference between the actual environment and the agent’s model of it. To generate the required data, the student will need to run a world simulator (movement and percepts) using the true model (northwest prior, southeast movement tendency), while running the agent’s state estimator using the assumed model (uniform prior, uniformly random movement). The student will also begin to appreciate the inexpressiveness of HMMs after constructing the  $64 \times 64$  transition matrix from the more natural representation in terms of coordinates.

Perhaps surprisingly, the data for expected localization error (expected Manhattan distance between true location and the posterior state estimate) show that having an incorrect model is not too problematic. A “southeast” bias of  $b$  was implemented by multiplying the probability of any south or east move by  $b$  and then renormalizing the distribution before sampling. Graphs for four different values of the bias are shown in Figure S14.2. The results suggest that the sensor data sequence overwhelms any error introduced by the incorrect motion model.

### Exercise 14.3.#ROOM

Consider a version of the vacuum robot (page 477) that has the policy of going straight for as long as it can; only when it encounters an obstacle does it change to a new (randomly selected) heading. To model this robot, each state in the model consists of a (*location*, *head-*

*ing) pair. Implement this model and see how well the Viterbi algorithm can track a robot with this model. The robot's policy is more constrained than the random-walk robot; does that mean that predictions of the most likely path are more accurate?*

The code for this exercise is very similar to that for Exercise 14.HMMR. The main difference is the state space: instead of 64 locations, the state space has 256 location-heading pairs, and the transition matrix is  $256 \times 256$ —starting to be a little painful when running hundreds of trials. We also need to add a “bump” bit to the percept vector, and we assume this is perfectly observed (else the robot could not always decide to pick a new heading). Generally we expect localization to be more accurate, since the sensor sequence need only disambiguate among a small number of possible headings rather than an exponentially growing set of random-walk paths. Also the exact bump sensor will eliminate many possible states completely.

### Exercise 14.3.#FILL

This exercise is concerned with filtering in an environment with no landmarks. Consider a vacuum robot in an empty room, represented by an  $n \times m$  rectangular grid. The robot's location is hidden; the only evidence available to the observer is a noisy location sensor that gives an approximation to the robot's location. If the robot is at location  $(x, y)$  then with probability .1 the sensor gives the correct location, with probability .05 each it reports one of the 8 locations immediately surrounding  $(x, y)$ , with probability .025 each it reports one of the 16 locations that surround those 8, and with the remaining probability of .1 it reports “no reading.” The robot's policy is to pick a direction and follow it with probability .8 on each step; the robot switches to a randomly selected new heading with probability .2 (or with probability 1 if it encounters a wall). Implement this as an HMM and do filtering to track the robot. How accurately can we track the robot's path?

The code for this exercise is very similar to that for Exercise 14.ROOM. The state is again a location/heading pair with a  $256 \times 256$  transition matrix. The observation model is different: instead of a 4-bit percept (16 possible percepts), the percept is a location (or null), for  $n \times m + 1$  possible percepts. Generally speaking, tracking works well near the walls because any bump (or even the lack thereof) helps to pin down the location. Away from the walls, the location uncertainty will be slightly worse but still generally accurate because the location errors are independent and unbiased and the policy is fairly predictable. It is reasonable to expect students to provide snapshots or movies of true location and posterior estimate, particularly if they are given suitable graphics infrastructure to make this easy.

### Exercise 14.3.#HMME

Consider an HMM with state variables  $\{X_i\}$  and emission variables  $\{Y_i\}$ .

- (i)** [true or false]  $X_i$  is always conditionally independent of  $Y_{i+1}$  given  $X_{i+1}$ .

- (ii) [true or false] There exists an HMM where  $X_i$  is conditionally independent of  $Y_i$  given  $X_{i+1}$ .
- (iii) [true or false] If  $Y_i = X_i$  with probability 1, and the state space is of size  $k$ , then the most efficient algorithm for computing  $p(X_t|y_1 \dots, y_t)$  takes  $O(k)$  or less time.

True, True, True

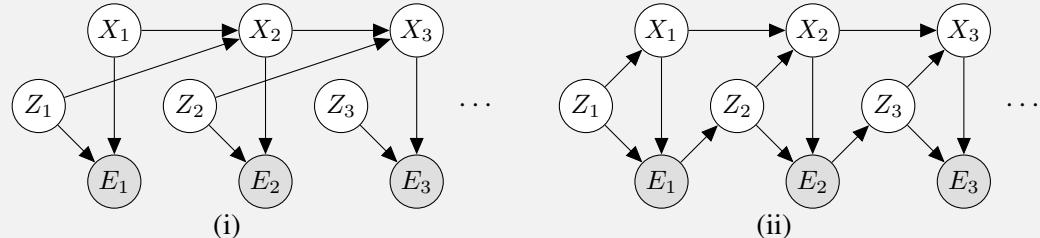
### Exercise 14.3.#ELPS

Recall that for a standard HMM the Elapse Time update and the Observation update are of the respective forms:

$$P(X_t | e_{1:t-1}) = \sum_{x_{t-1}} P(X_t | x_{t-1})P(x_{t-1} | e_{1:t-1})$$

$$P(X_t | e_{1:t}) \propto P(X_t | e_{1:t-1})P(e_t | x_t)$$

We now consider the following two HMM-like models:



Derive the modified Elapse Time update and the modified Observation update that correctly compute the beliefs from the quantities that are available in the Bayes' Net.

#### (i) First Bayes Net

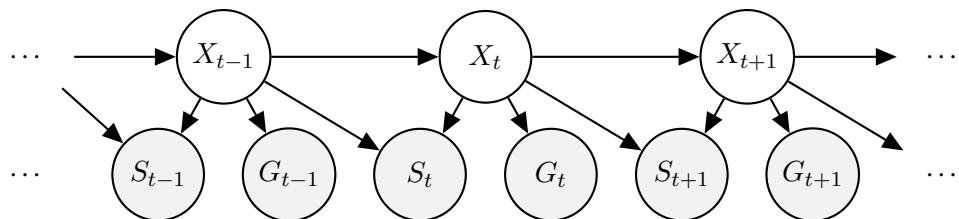
(a) In the elapse time update, we want to get from  $P(X_{t-1}, Z_{t-1}|e_{1:t-1})$  to  $P(X_t, Z_t|e_{1:t-1})$ .

$$\begin{aligned} P(X_t, Z_t | e_{1:t-1}) &= \sum_{x_{t-1}, z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1} | e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} | e_{1:t-1}) P(X_t | x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &\quad P(Z_t | X_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} | e_{1:t-1}) P(X_t | x_{t-1}, z_{t-1}) P(Z_t) \end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

(b) In the observation update, we want to get from  $P(X_t, Z_t|e_{1:t-1})$  to  $P(X_t, Z_t|e_{1:t})$ .

$$\begin{aligned} P(X_t, Z_t | e_{1:t}) &\propto P(X_t, Z_t, e_t | e_{1:t-1}) \\ &\propto P(X_t, Z_t | e_{1:t-1}) P(e_t | X_t, Z_t, e_{1:t-1}) \\ &\propto P(X_t, Z_t | e_{1:t-1}) P(e_t | X_t, Z_t) \end{aligned}$$

**Figure S14.3** HMM for Exercise 14.TRFC.

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

(ii) Second Bayes Net

- (a) In the elapse time update, we want to get from  $P(X_{t-1}, Z_{t-1}|e_{1:t-1})$  to  $P(X_t, Z_t|e_{1:t-1})$ .

$$\begin{aligned}
 & P(X_t, Z_t|e_{1:t-1}) \\
 &= \sum_{x_{t-1}, z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1}|e_{1:t-1}) \\
 &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1})P(Z_t|x_{t-1}, z_{t-1}, e_{1:t-1})P(X_t|Z_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\
 &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1})P(Z_t|e_{t-1})P(X_t|x_{t-1}, Z_t)
 \end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

- (b) In the observation update, we want to get from  $P(X_t, Z_t|e_{1:t-1})$  to  $P(X_t, Z_t|e_{1:t})$ .

$$\begin{aligned}
 P(X_t, Z_t|e_{1:t}) &\propto P(X_t, Z_t, e_t|e_{1:t-1}) \\
 &\propto P(X_t, Z_t|e_{1:t-1})P(e_t|X_t, Z_t, e_{1:t-1}) \\
 &\propto P(X_t, Z_t|e_{1:t-1})P(e_t|X_t, Z_t)
 \end{aligned}$$

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

**Exercise 14.3.#TRFC**

Transportation researchers are trying to improve traffic in the city but, in order to do that, they first need to estimate the location of each of the cars in the city. They need our help to model this problem as an inference problem of an HMM. For this question, assume that only *one* car is being modeled.

- a. We define the variables as follows:  $X$ , the location of the car;  $S$ , the noisy location of the car from the signal strength at a nearby cell phone tower; and  $G$ , the noisy location

of the car from GPS. In our HMM model Figure S14.3, the signal-dependent location  $S_t$  not only depends on the current state  $X_t$  but also depends on the previous state  $X_{t-1}$ .

We want to compute the belief  $P(x_t|s_{1:t}, g_{1:t})$ . In this part we consider an update that combines the dynamics and observation update in a *single* update. Complete the **forward update** expression by filling in the expression of the form:

$$P(x_t|s_{1:t}, g_{1:t}) = \underline{\hspace{10em}} P(x_{t-1}|s_{1:t-1}, g_{1:t-1}).$$

- b. We consider extending the Viterbi algorithm for the HMM from the previous part. We want to find the most likely sequence of states  $X_{1:T}$  given the sequence of observations  $s_{1:T}$  and  $g_{1:T}$ . Find the dynamic programming update for  $t > 1$  for the modified HMM of the following form:

$$m_t[x_t] = \underline{\hspace{10em}} m_{t-1}[x_{t-1}].$$

- a. For this modified HMM, we have the dynamics and observation update in a single update because one of the previous independence assumptions does not longer holds.

$$\begin{aligned} P(x_t|s_{1:t}, g_{1:t}) &= \sum_{x_{t-1}} P(x_{t-1}, x_t|s_t, g_t, s_{1:t-1}, g_{1:t-1}) \\ &= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(x_{t-1}, x_t, s_t, g_t|s_{1:t-1}, g_{1:t-1}) \\ &= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t, g_t|x_{t-1}, x_t, s_{1:t-1}, g_{1:t-1}) \\ &\quad P(x_{t-1}, x_t|s_{1:t-1}, g_{1:t-1}) \\ &= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t, g_t|x_{t-1}, x_t) P(x_t|x_{t-1}, s_{1:t-1}, g_{1:t-1}) \\ &\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}) \\ &= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t|x_{t-1}, x_t) P(g_t|x_{t-1}, x_t) P(x_t|x_{t-1}) \\ &\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}) \\ &= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t|x_{t-1}, x_t) P(g_t|x_t) P(x_t|x_{t-1}) \\ &\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}) \end{aligned}$$

In the third to last step, we use the independence assumption  $S_t, G_t \perp\!\!\!\perp S_{1:t-1}, G_{1:t-1} | X_{t-1}, X_t$ ; in the second to last step, we use the independence assumption  $S_t \perp\!\!\!\perp G_t | X_{t-1}, X_t$  and  $X_t \perp\!\!\!\perp S_{1:t-1}, G_{1:t-1} | X_{t-1}$ ; and in the last step, we use the independence assumption

$$G_t \perp\!\!\!\perp X_{t-1} | X_t.$$

- b. If we remove the summation from the forward update equation of part (b), we get a joint probability of the states,

$$P(x_{1:t}|s_{1:t}, g_{1:t}) = \frac{P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}).$$

We can define  $m_t[x_t]$  to be the maximum joint probability of the states (for a particular  $x_t$ ) given all past and current observations, times some constant, and then we can find a recursive relationship for  $m_t[x_t]$ ,

$$\begin{aligned} m_t[x_t] &= P(s_{1:t}, g_{1:t}) \max_{x_{1:t-1}} P(x_{1:t}|s_{1:t}, g_{1:t}) \\ &= P(s_{1:t}, g_{1:t}) \max_{x_{1:t-1}} \frac{P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\ &= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1}) \frac{P(s_{1:t}, g_{1:t})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \\ &\quad \max_{x_{1:t-2}} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\ &= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})P(s_{1:t-1}, g_{1:t-1}) \\ &\quad \max_{x_{1:t-2}} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\ &= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]. \end{aligned}$$

Notice that the maximum joint probability of states up to time  $t = T$  given all past and current observations is given by

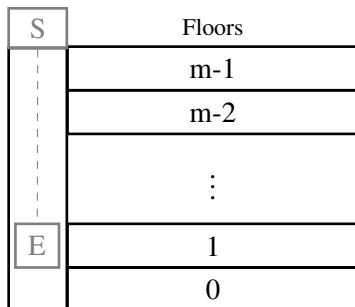
$$\max_{x_{1:T}} P(x_{1:T}|s_{1:T}, g_{1:T}) = \frac{\max_{x_T} m_T[x_T]}{P(s_{1:T}, g_{1:T})}.$$

We can recover the actual most likely sequence of states by bookkeeping back pointers of the states the maximized the Viterbi update equations.

### Exercise 14.3.#ELEV

Assume the elevator of the Disney Tower of Terror  $E$  follows a Markovian process and has  $m$  floors at which it can stop. In the dead of night, you install a sensor  $S$  at the top of the shaft that gives approximate distance measurements, quantized into  $n$  different distance bins. Assume that the elevator stops at  $T$  floors as part of the ride and the initial distribution of the elevator is uniform over the  $m$  floors as shown in Figure S14.4.

You want to know the most probable sequence of (hidden) states  $X_{1:T}$  given your observations  $y_{1:T}$  from the sensor, so you turn to the Viterbi algorithm, which performs the following update at each step:



**Figure S14.4** Disney Tower for Exercise 14.ELEV.

$$m_t[x_t] = P(y_t|x_t) \max_{x_{t-1}} [P(x_t|x_{t-1})m_{t-1}[x_{t-1}]]$$

$$a_t[x_t] = \arg \max_{x_{t-1}} m_{t-1}[x_{t-1}]$$

- a. What is the runtime and space complexity of the Viterbi algorithm to determine all previous states for this scenario?
- b. If we only want to determine the previous  $K$  states, what is the runtime and space complexity of the Viterbi algorithm to determine the previous  $K$  states?
- c. Suppose you instead only wish to determine the current distribution (at time  $T$ ) for the elevator, given your  $T$  observations, so you use the forward algorithm, with update step shown here:

$$P(X_t|y_t) \propto P(y_t|x_t) \sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}, y_{0:t-1})$$

Additionally, from your previous analysis, you note that there are some states which are unreachable from others (e.g., the elevator cannot travel from the top floor to the bottom in a single timestep). Specifically, from each state, there are between  $G/2$  and  $G$  states which can be reached in the next timestep, where  $G < m$ . What is the runtime and space complexity for the forward algorithm to estimate the current state at time  $T$ , assuming we ignore states that cannot be reached in each update?

- d. Suppose you decide to use a particle filter instead of the forward algorithm. What is the runtime of a particle filter with  $P$  particles?

- a. **Runtime.** For each timestep that we want to decode, we loop over all states and consider the transition probability to that state from all previous states. Thus, the runtime is  $Tm^2$ , since we repeat this process for each timestep ( $T$ ), and at each timestep we have a double for loop for states (one to compute m-value for all states, one to compute arg max of previous m-values and transition probabilities, so  $m^2$ ).

**Space Complexity.** We store only our a-value arrays (which keep track of the best transition along the path to the current state). These are essentially back-pointers so we can reconstruct the best path. For a sequence of length  $T$ , we then have a matrix of these pointers that is  $T \times m$ , so we take up  $Tm$  space.

- b. **Runtime.** The runtime is the same as part a since the forward pass part of the Viterbi algorithm is unchanged (e.g., we still need to consider all transition probabilities at all timesteps to determine the most likely path, regardless of how far back we want to go).

**Space Complexity.** If we only wish to determine the previous  $K$  states, we only need to store a-values corresponding to the previous  $K$  timesteps (since when we reconstruct the path, we only need to go  $K$  steps back). Thus, the space complexity decreases to  $Km$  from part a.

- c. **Runtime.** The normal runtime for the forward algorithm is  $Tm^2$  (very similar to Viterbi - we consider transition probabilities from each state to each other state during our forward pass). However, if we can ignore those states with zero transition probability, then we only need to consider  $G$  states when calculating our sum for each state. Thus, the runtime is reduced to  $TmG$ .

**Space Complexity.** The space complexity for the forward algorithm is unchanged by how many states we can reach from each state, so this is the same as the normal space complexity for the forward algorithm. It is  $m$  because we store  $P(X_t|y_t)$  for each state and we only store one timestep (since we only care about the current state probabilities).

- d. Assuming the time for resampling a single particle is constant, then the runtime is  $TP$ . At each timestep (total of  $T$ ), we perform the time elapse and observation updates in constant time for each of the  $P$  particles.

## 14.4 Particle Filtering

### Exercise 14.4.#PFTW

Consider two particle filtering implementations:

**Implementation 1:** Initialize particles by sampling from initial state distribution and assigning uniform weights.

1. Propagate particles, retaining weights
2. Resample according to weights
3. Weight according to observations

**Implementation 2:** Initialize particles by sampling from initial state distribution.

1. Propagate unweighted particles
2. Weight according to observations
3. Resample according to weights

(i) [true or false] Implementation 2 will typically provide a better approximation of the estimated distribution than implementation 1.

(ii) [true or false] If the transition model is deterministic then both implementations provide equally good estimates of the distribution.

- (iii) [true or false] If the observation model is deterministic then both implementations provide equally good estimates of the distribution.

True, True, False

#### Exercise 14.4.#PFCD

- (iv) [true or false] With a deterministic transition model and a stochastic observation model, as time goes to infinity, when running a particle filter we will end up with all identical particles.
- (v) [true or false] With a deterministic observation model, all particles might end up having zero weight.
- (vi) [true or false] It is possible to use particle filtering when the state space is continuous.
- (vii) [true or false] It is possible to use particle filtering when the state space is discrete.
- (viii) [true or false] As the number of particles goes to infinity, particle filtering will represent the same probability distribution that you'd get by using exact inference.
- (ix) [true or false] Particle filtering can represent a flat distribution (i.e. uniform) with fewer particles than it would need for a more concentrated distribution (i.e. Gaussian).

True, True, True, True, True, False

#### Exercise 14.4.#PFHM

In which settings is particle filtering better than exact HMM inference?

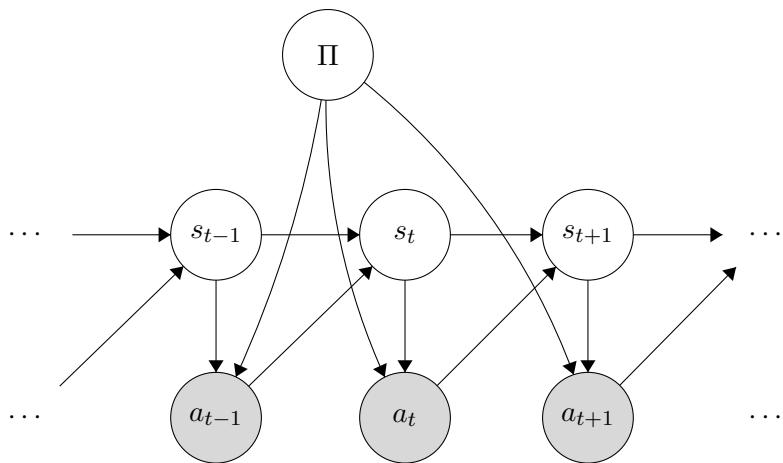
- Large vs. Small state spaces.
- Prioritizing runtime vs. accuracy.

Particle Filtering works better in large state space settings where we wish to prioritize runtime over accuracy.

Exact HMM inference scales with the state space. For particle filtering, the number of particles determines the time complexity and quality of the approximation. Large (even continuous) state spaces can be approximated by a much smaller number of particles.

#### Exercise 14.4.#PFNM

Consider an HMM with  $T$  timesteps, hidden state variables  $X_1, \dots, X_T$ , and observed variables  $E_1, \dots, E_T$ . Let  $S$  be the number of possible states for each hidden state variable  $X$ . We want to compute (with the forward algorithm) or estimate (with particle filtering)  $P(X_T | E_1 = e_1, \dots, E_T = e_T)$ . How many particles, in terms of  $S$  and  $T$ , would it take for particle filtering to have the same time complexity as the forward algorithm? You can assume that, in particle filtering, each sampling step can be done in constant time for a single particle (though this is not necessarily the case in reality).



**Figure S14.5** Bayes net for Exercise 14.PFDP.

$S^2$

#### Exercise 14.4.#PFDP

Consider a modified version of the apprenticeship problem. We are observing an agent's actions in an MDP and are trying to determine which out of a set  $\{\pi_1, \dots, \pi_n\}$  the agent is following. Let the random variable  $\Pi$  take values in that set and represent the policy that the agent is acting under. We consider only *stochastic* policies, so that  $A_t$  is a random variable with a distribution conditioned on  $S_t$  and  $\Pi$ . As in a typical MDP,  $S_t$  is a random variable with a distribution conditioned on  $S_{t-1}$  and  $A_{t-1}$ . The full Bayes net is shown below.

The agent acting in the environment knows what state it is currently in (as is typical in the MDP setting). Unfortunately, however, we, the observer, cannot see the states  $S_t$ . Thus we are forced to use an adapted particle filtering algorithm to solve this problem. Concretely, we will develop an efficient algorithm to estimate  $P(\Pi | a_{1:t})$ .

The Bayes net for this problem is in Figure S14.5.

- For  $t > 10$ , what sets of variables need to be given for  $S_t \perp\!\!\!\perp S_{t-2}$ ?
- We will compute our estimate for  $P(\Pi | a_{1:t})$  by coming up with a recursive algorithm for computing  $P(\Pi, S_t | a_{1:t})$ . (We can then sum out  $S_t$  to get the desired distribution; in this problem we ignore that step.) Write a recursive expression for  $P(\Pi, S_t | a_{1:t})$  in terms of the CPTs in the Bayes net above. Hint: Think about the forward algorithm.
- We now try to adapt particle filtering to approximate this value. Each particle will contain a single state  $s_t$  and a potential policy  $\pi_i$ . Write pseudocode for the body of the loop in our adapted particle filtering algorithm. Specifically, write the Elapse Time and Incorporate evidence steps. Remember, we want to approximate  $P(\Pi, S_t | a_{1:t})$ .

- Using d-separation, we can recover the independence relations as shown.

- $S_t \perp\!\!\!\perp S_{t-2} | S_{t-1}, A_{1:t-1}$
  - $S_t \perp\!\!\!\perp S_{t-2} | \Pi, S_{t-1}$
  - $S_t \perp\!\!\!\perp S_{t-2} | \Pi, S_{t-1}, A_{1:t-1}$
- b.  $P(\Pi, S_t | a_{1:t}) \propto \sum_{s_{t-1}} P(\Pi, s_{t-1} | a_{1:t-1})P(a_t | S_t, \Pi)P(S_t | s_{t-1}, a_{t-1})$
- c. The pseudocode is as follows.
1. Elapse time: for each particle  $(s_t, \pi_i)$ , sample a successor  $s_{t+1}$  from  $P(S_{t+1} | s_t, a_t)$ . The policy  $\pi'$  in the new particle is  $\pi_i$ .
  2. Incorporate evidence: To each new particle  $(s_{t+1}, \pi')$ , assign weight  $P(a_{t+1} | s_{t+1}, \pi')$ .
  3. Resample particles from the weighted particle distribution.

## 14.5 Kalman Filters

---

### Exercise 14.5.#KFSW

Often, we wish to monitor a continuous-state system whose behavior switches unpredictably among a set of  $k$  distinct “modes.” For example, an aircraft trying to evade a missile can execute a series of distinct maneuvers that the missile may attempt to track. A Bayesian network representation of such a **switching Kalman filter** model is shown in Figure ??.

- a. Suppose that the discrete state  $S_t$  has  $k$  possible values and that the prior continuous state estimate  $\mathbf{P}(\mathbf{X}_0)$  is a multivariate Gaussian distribution. Show that the prediction  $\mathbf{P}(\mathbf{X}_1)$  is a **mixture of Gaussians**—that is, a weighted sum of Gaussians such that the weights sum to 1.
- b. Show that if the current continuous state estimate  $\mathbf{P}(\mathbf{X}_t | \mathbf{e}_{1:t})$  is a mixture of  $m$  Gaussians, then in the general case the updated state estimate  $\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1})$  will be a mixture of  $km$  Gaussians.
- c. What aspect of the temporal process do the weights in the Gaussian mixture represent?

The results in (a) and (b) show that the representation of the posterior grows without limit even for switching Kalman filters, which are among the simplest hybrid dynamic models.

- a. Looking at the fragment of the model containing just  $S_0$ ,  $\mathbf{X}_0$ , and  $\mathbf{X}_1$ , we have

$$\mathbf{P}(\mathbf{X}_1) = \sum_{s_0=1}^k P(s_0) \int_{\mathbf{x}_0} P(\mathbf{x}_0) \mathbf{P}(X_1 | \mathbf{x}_0, s_0)$$

From the properties of the Kalman filter, we know that the integral gives a Gaussian for each different value of  $s_0$ . Hence, the prediction distribution is a mixture of  $k$  Gaussians, each weighted by  $P(s_0)$ .

- b.** The update equation for the switching Kalman filter is

$$\begin{aligned} \mathbf{P}(\mathbf{X}_{t+1}, S_{t+1} | \mathbf{e}_{1:t+1}) \\ = \alpha \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}, S_{t+1}) \sum_{s_t=1}^k \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{x}_t, s_t | \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1}, S_{t+1} | \mathbf{x}_t, s_t) \\ = \alpha \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{s_t=1}^k P(s_t | \mathbf{e}_{1:t}) \mathbf{P}(S_{t+1} | s_t) \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{x}_t | \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t, s_t) \end{aligned}$$

We are given that  $\mathbf{P}(\mathbf{x}_t | \mathbf{e}_{1:t})$  is a mixture of  $m$  Gaussians. Each Gaussian is subject to  $k$  different linear-Gaussian projections and then updated by a linear-Gaussian observation, so we obtain a sum of  $km$  Gaussians. Thus, after  $t$  steps we have  $k^t$  Gaussians.

- c.** Each weight represents the probability of one of the  $k^t$  sequences of values for the switching variable.

### Exercise 14.5.#KALM

Complete the missing step in the derivation of Equation (14.19) on page 481, the first update step for the one-dimensional Kalman filter.

This is a simple exercise in algebra. We have

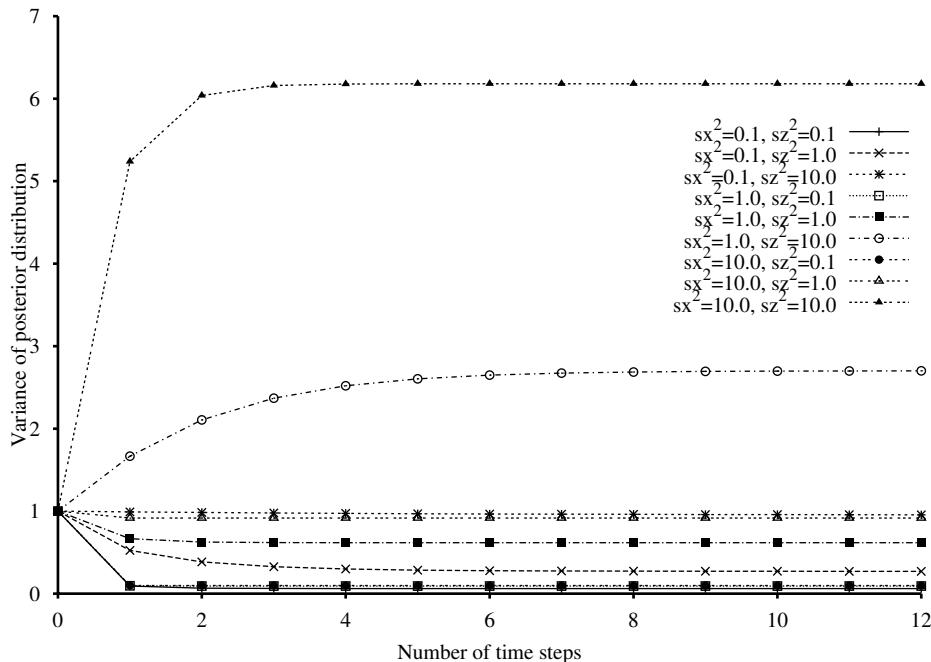
$$\begin{aligned} P(x_1 | z_1) &= \alpha e^{-\frac{1}{2} \left( \frac{(z_1 - x_1)^2}{\sigma_z^2} \right)} e^{-\frac{1}{2} \left( \frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2} \right)} \\ &= \alpha e^{-\frac{1}{2} \left( \frac{(\sigma_0^2 + \sigma_x^2)(z_1 - x_1)^2 + \sigma_z^2(x_1 - \mu_0)^2}{\sigma_z^2(\sigma_0^2 + \sigma_x^2)} \right)} \\ &= \alpha e^{-\frac{1}{2} \left( \frac{(\sigma_0^2 + \sigma_x^2)(z_1^2 - 2z_1x_1 + x_1^2) + \sigma_z^2(x_1^2 - 2\mu_0x_1 + \mu_0^2)}{\sigma_z^2(\sigma_0^2 + \sigma_x^2)} \right)} \\ &= \alpha e^{-\frac{1}{2} \left( \frac{(\sigma_0^2 + \sigma_x^2 + \sigma_z^2)x_1^2 - 2((\sigma_0^2 + \sigma_x^2)z_1 + \sigma_z^2\mu_0)x_1 + c}{\sigma_z^2(\sigma_0^2 + \sigma_x^2)} \right)} \\ &= \alpha' e^{-\frac{1}{2} \left( \frac{(x_1 - \frac{(\sigma_0^2 + \sigma_x^2)z_1 + \sigma_z^2\mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_z^2})^2}{\frac{(\sigma_0^2 + \sigma_x^2)\sigma_z^2}{(\sigma_0^2 + \sigma_x^2 + \sigma_z^2)}} \right)}. \end{aligned}$$

### Exercise 14.5.#VARI

Let us examine the behavior of the variance update in Equation (14.20) (page 481).

- Plot the value of  $\sigma_t^2$  as a function of  $t$ , given various values for  $\sigma_x^2$  and  $\sigma_z^2$ .
- Show that the update has a fixed point  $\sigma^2$  such that  $\sigma_t^2 \rightarrow \sigma^2$  as  $t \rightarrow \infty$ , and calculate the value of  $\sigma^2$ .
- Give a qualitative explanation for what happens as  $\sigma_x^2 \rightarrow 0$  and as  $\sigma_z^2 \rightarrow 0$ .

- a.** See Figure S14.6.



**Figure S14.6** Graph for Ex. 15.7, showing the posterior variance  $\sigma_t^2$  as a function of  $t$  for various values of  $\sigma_x^2$  and  $\sigma_z^2$ .

b. We can find a fixed point by solving

$$\sigma^2 = \frac{(\sigma^2 + \sigma_x^2)\sigma_z^2}{\sigma^2 + \sigma_x^2 + \sigma_z^2}$$

for  $\sigma^2$ . Using the quadratic formula and requiring  $\sigma^2 \geq 0$ , we obtain

$$\sigma^2 = \frac{-\sigma_x^2 + \sqrt{\sigma_x^4 + 4\sigma_x^2\sigma_z^2}}{\sigma_z^2}$$

We omit the proof of convergence, which, presumably, can be done by showing that the update is a contraction (i.e., after updating, two different starting points for  $\sigma_t$  become closer).

c. As  $\sigma_x^2 \rightarrow 0$ , we see that the fixed point  $\sigma^2 \rightarrow 0$  also. This is because  $\sigma_x^2 = 0$  implies a deterministic path for the object. Each observation supplies more information about this path, until its parameters are known completely.

As  $\sigma_z^2 \rightarrow 0$ , the variance update gives  $\sigma^{t+1} \rightarrow 0$  immediately. That is, if we have an exact observation of the object's state, then the posterior is a delta function about that observed value regardless of the transition variance.

## 14.6 Dynamic Bayesian Networks

### Exercise 14.6.#VACP

We have described three policies for the vacuum robot: (1) a uniform random walk, (2) a bias for wandering southeast, as described in Exercise 14.HMMR, and (3) the policy described in Exercise 14.ROOM. Suppose an observer is given the observation sequence from a vacuum robot, but is not sure which of the three policies the robot is following. What approach should the observer use to find the most likely path, given the observations? Implement the approach and test it. How much does the localization accuracy suffer, compared to the case in which the observer knows which policy the robot is following?

The correct model is a DBN with location and heading variables along with a single atemporal variable for the policy  $\Phi$  which has all the state variables as children. We can reduce the inference problem to computations on the three original models by summing out the policy variable as follows:

$$\begin{aligned} P(X_t | e_{1:t}) &= \sum_{\phi} P(X_t | e_{1:t}, \phi)P(\phi | e_{1:t}) \\ &= \alpha \sum_{\phi} P(X_t | e_{1:t}, \phi)P(e_{1:t} | \phi)P(\phi) \end{aligned}$$

Thus, we need to run both the filtering calculation and the likelihood calculation (as described in the text) for each of the three policies. The likelihood times the prior gives the posterior for the policy, and the state estimate is the posterior-weighted combination of the three state estimates.

Generally we expect the policy to be quickly recognizable in cases where it has a significant impact on the percepts; if the impact is small (as, for example, in the case of a small southeasterly bias) the inability to identify the policy will not have a large effect on accuracy.

### Exercise 14.6.#SLEP

A professor wants to know if students are getting enough sleep. Each day, the professor observes whether the students sleep in class, and whether they have red eyes. The professor has the following domain theory:

- The prior probability of getting enough sleep, with no observations, is 0.7.
- The probability of getting enough sleep on night  $t$  is 0.8 given that the student got enough sleep the previous night, and 0.3 if not.
- The probability of having red eyes is 0.2 if the student got enough sleep, and 0.7 if not.
- The probability of sleeping in class is 0.1 if the student got enough sleep, and 0.3 if not.

Formulate this information as a dynamic Bayesian network that the professor could use to filter or predict from a sequence of observations. Then reformulate it as a hidden Markov model that has only a single observation variable. Give the complete probability tables for the model.

The DBN has three variables:  $S_t$ , whether the student gets enough sleep;  $R_t$ , whether they have red eyes in class;  $C_t$ , whether the student sleeps in class.  $S_t$  is a parent of  $S_{t+1}$ ,  $R_t$ , and  $C_t$ . The CPTs are given by

$$\begin{aligned} P(s_0) &= 0.7 \\ P(s_{t+1}|s_t) &= 0.8 \\ P(s_{t+1}|\neg s_t) &= 0.3 \\ P(r_t|s_t) &= 0.2 \\ P(r_t|\neg s_t) &= 0.7 \\ P(c_t|s_t) &= 0.1 \\ P(c_t|\neg s_t) &= 0.3 \end{aligned}$$

To reformulate as an HMM with a single observation node, simply combine the 2-valued variables “having red eyes” and “sleeping in class” into a single 4-valued variable, multiplying together the emission probabilities. (Probability tables omitted.)

### Exercise 14.6.#SLER

For the DBN specified in Exercise 14.SLE and for the evidence values

- $\mathbf{e}_1$  = not red eyes, not sleeping in class
- $\mathbf{e}_2$  = red eyes, not sleeping in class
- $\mathbf{e}_3$  = red eyes, sleeping in class

perform the following computations:

- a. State estimation: Compute  $P(EnoughSleep_t|\mathbf{e}_{1:t})$  for each of  $t = 1, 2, 3$ .
- b. Smoothing: Compute  $P(EnoughSleep_t|\mathbf{e}_{1:3})$  for each of  $t = 1, 2, 3$ .
- c. Compare the filtered and smoothed probabilities for  $t = 1$  and  $t = 2$ .

a. We apply the forward algorithm to compute these probabilities.

$$\begin{aligned}
 P(S_0) &= \langle 0.7, 0.3 \rangle \\
 P(S_1) &= \sum_{s_0} P(S_1|s_0)P(s_0) \\
 &= (\langle 0.8, 0.2 \rangle 0.7 + \langle 0.3, 0.7 \rangle 0.3) \\
 &= \langle 0.65, 0.35 \rangle \\
 P(S_1|e_1) &= \alpha P(e_1|S_1)P(S_1) \\
 &= \alpha \langle 0.8 \times 0.9, 0.3 \times 0.7 \rangle \langle 0.65, 0.35 \rangle \\
 &= \alpha \langle 0.72, 0.21 \rangle \langle 0.65, 0.35 \rangle \\
 &= \langle 0.8643, 0.1357 \rangle \\
 P(S_2|e_1) &= \sum_{s_1} P(S_2|s_1)P(s_1|e_1) \\
 &= \langle 0.7321, 0.2679 \rangle \\
 P(S_2|e_{1:2}) &= \alpha P(e_2|S_2)P(S_2|e_1) \\
 &= \langle 0.5010, 0.4990 \rangle \\
 P(S_3|e_{1:2}) &= \sum_{s_2} P(S_3|s_2)P(s_2|e_{1:2}) \\
 &= \langle 0.5505, 0.4495 \rangle \\
 P(S_3|e_{1:3}) &= \alpha P(e_3|S_3)P(S_3|e_{1:2}) \\
 &= \langle 0.1045, 0.8955 \rangle
 \end{aligned}$$

Similar to many students during the course of the school term, the student observed here seems to have a higher likelihood of being sleep deprived as time goes on!

b. First we compute the backwards messages:

$$\begin{aligned}
 P(e_3|S_3) &= \langle 0.2 \times 0.1, 0.7 \times 0.3 \rangle \\
 &= \langle 0.02, 0.21 \rangle \\
 P(e_3|S_2) &= \sum_{s_3} P(e_3|s_3)P(s_3|S_2) \\
 &= \langle 0.02 \times 0.8 + 0.21 \times 0.2, 0.02 \times 0.3 + 0.21 \times 0.7 \rangle \\
 &= \langle 0.0588, 0.153 \rangle \\
 P(e_{2:3}|S_1) &= \sum_{s_2} P(e_2|s_2)P(e_3|s_2)P(s_2|S_1) \\
 &= \langle 0.0233, 0.0556 \rangle
 \end{aligned}$$

Then we combine these with the forwards messages computed previously and normal-

ize:

$$\begin{aligned} P(S_1|e_{1:3}) &= \alpha P(S_1|e_1)P(e_{2:3}|S_1) \\ &= \langle 0.7277, 0.2723 \rangle \end{aligned}$$

$$\begin{aligned} P(S_2|e_{1:3}) &= \alpha P(S_2|e_{1:2})P(e_3|S_1) \\ &= \langle 0.2757, 0.7243 \rangle \end{aligned}$$

$$P(S_3|e_{1:3}) = \langle 0.1045, 0.8955 \rangle$$

- c. The smoothed analysis places the time the student started sleeping poorly one step earlier than than filtered analysis, integrating future observations indicating lack of sleep at the last step.

### Exercise 14.6.#SLES

Suppose that a particular student shows up with red eyes and sleeps in class every day. Given the model described in Exercise 14.SLEP, explain why the probability that the student had enough sleep the previous night converges to a fixed point rather than continuing to go down as we gather more days of evidence. What is the fixed point? Answer this both numerically (by computation) and analytically.

The probability reaches a fixed point because there is always some chance of spontaneously starting to sleep well again, and students who sleep well sometimes have red eyes and sleep in class. Even if we knew for sure that the student didn't sleep well on day  $t$ , and that they slept in class with red eyes on day  $t + 1$ , there would still be a chance that they slept well on day  $t + 1$ .

Numerically one can repeatedly apply the forward equations to find equilibrium probabilities of  $\langle 0.0432, 0.9568 \rangle$ .

Analytically, we are trying to find the vector  $(p_0, p_1)^T$  which is the fixed point to the forward equation, which one can pose in matrix form as

$$(p_0, p_1)^T = \alpha \begin{pmatrix} 0.016 & 0.006 \\ 0.042 & 0.147 \end{pmatrix} (p_0, p_1)^T$$

where  $\alpha$  is a normalization constant. That is,  $(p_0, p_1)^T$  is an eigenvector of the given matrix. Computing, we find that the only positive eigenvalue is 0.1487, which has eigenvector (normalized to sum to one)  $(0.0432, 0.9568)^T$ , just as we numerically computed.

### Exercise 14.6.#BATS

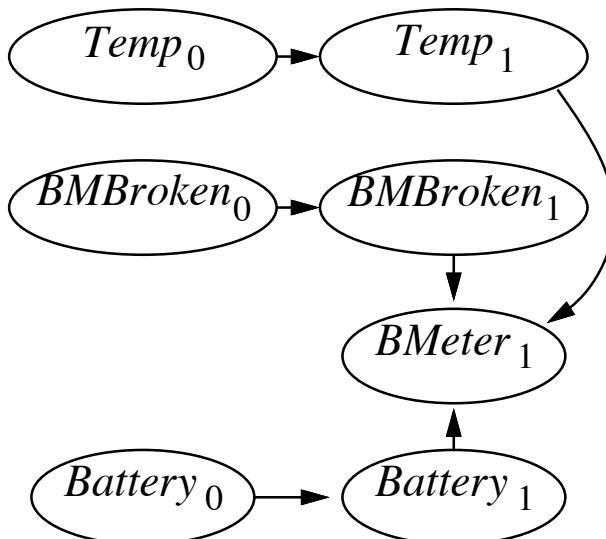
This exercise analyzes in more detail the persistent-failure model for the battery sensor in Figure 14.15(a) (page 489).

- Figure 14.15(b) stops at  $t = 32$ . Describe qualitatively what should happen as  $t \rightarrow \infty$  if the sensor continues to read 0.
- Suppose that the external temperature affects the battery sensor in such a way that transient failures become more likely as temperature increases. Show how to augment the

DBN structure in Figure 14.15(a), and explain any required changes to the CPTs.

- c. Given the new network structure, can battery readings be used by the robot to infer the current temperature?

- a. The curve of interest is the one for  $E(Battery_t | \dots 5555000000 \dots)$ . In the absence of any useful sensor information from the battery meter, the posterior distribution for the battery level is the same as the projection without evidence. The transition model for the battery includes a small probability for downward transitions in the battery level at each time step, but zero probability for upward transitions (there are no recharging actions in the model). Thus, the stationary distribution towards which the battery level tends has value 0 with probability 1. The curve for  $E(Battery_t | \dots 5555000000 \dots)$  will asymptote to 0.
- b. See Figure S14.7. The CPT for  $BMeter_1$  has a probability of transient failure (i.e., reporting 0) that increases with temperature.
- c. The agent can obviously calculate the posterior distribution over  $Temp_t$  by filtering the observation sequence in the usual way. This posterior can be informative if the effect of temperature on transient failure is non-negligible and transient failures occur more frequently than do major changes in temperature. Essentially, the temperature is estimated from the frequency of “blips” in the sequence of battery meter readings.



**Figure S14.7** Modification of Figure 15.13(a) to include the effect of external temperature on the battery meter.

**Exercise 14.6.#BATT**

Consider the DBN in Figure 14.13(b). In the chapter, the battery level  $Battery_t$  and the battery meter reading  $BMeter_t$  are assumed to be integer-valued with a range of 0 to 5. In this exercise, we will look at a more realistic model where they are continuous variables; for simplicity we will assume a range  $[0, 1]$ .

- Give exact expressions for the sensor model  $P(BMeter_t | Battery_t)$ , for both beta distributions and truncated normal distributions, where the mode is at the true value and the standard deviation is 0.1. (Details of these distributions are available in many online sources.) Plot the conditional distributions for  $Battery_t = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ .
- Describe in detail a suitable distribution for the transition model  $P(Battery_{t+1} | Battery_t, \dot{\mathbf{X}}_t)$ , where  $\dot{\mathbf{X}}_t$  is a two-dimensional velocity vector. You may assume that the battery drains at a small constant rate  $r$  plus an amount proportional to the absolute velocity of the robot, with a standard deviation that is also proportional to the absolute velocity. Remember that the battery charge cannot go below zero and cannot increase.
- Explain how to integrate a  $Charging_t$  variable into the DBN, which is true just when the robot is plugged into the charging station.
- What might a reasonable prior  $P(Battery_0)$  look like?

[[TBC]]

**Exercise 14.6.#DBNE**

Consider applying the variable elimination algorithm to the umbrella DBN unrolled for three slices, where the query is  $\mathbf{P}(R_3|u_1, u_2, u_3)$ . Show that the space complexity of the algorithm—the size of the largest factor—is the same, regardless of whether the rain variables are eliminated in forward or backward order.

The process works exactly as on page 507. We start with the full expression:

$$\mathbf{P}(R_3|u_1, u_2, u_3) = \alpha \sum_{r_1} \sum_{r_2} P(r_1)P(u_1|r_1)P(r_2|r_1)P(u_2|r_2)\mathbf{P}(R_3|r_2)\mathbf{P}(u_3|R_3)$$

Whichever order we push in the summations, the variable elimination process never creates factors containing more than two variables, which is the same size as the CPTs in the original network. In fact, given an HMM sequence of arbitrary length, we can eliminate the state variables in any order.

**Exercise 14.6.#RBPF**

Consider the Bayes net obtained by unrolling the DBN in Figure 14.20 to time step  $t$ .

Use the conditional independence properties of this network to show that

$$\begin{aligned} & \mathbf{P}(Dirt_{1,0:t}, \dots, Dirt_{42,0:t} | DirtSensor_{1:t}, WallSensor_{1:t}, Location_{1:t}) \\ &= \prod_i \mathbf{P}(Dirt_{i,0:t} | DirtSensor_{1:t}, Location_{1:t}). \end{aligned}$$

$$\begin{aligned} & \mathbf{P}(Dirt_{1,0:t}, \dots, Dirt_{42,0:t} | DirtSensor_{1:t}, WallSensor_{1:t}, Location_{1:t}) \\ &= \mathbf{P}(Dirt_{1,0:t}, \dots, Dirt_{42,0:t} | DirtSensor_{1:t}, Location_{1:t}) \\ &= \mathbf{P}(Dirt_{1,0:t} | DirtSensor_{1:t}, Location_{1:t}) \\ &\quad \mathbf{P}(\dots | DirtSensor_{1:t}, Location_{1:t}) \\ &\quad \mathbf{P}(Dirt_{42,0:t} | DirtSensor_{1:t}, Location_{1:t}) \\ &= \prod_i \mathbf{P}(Dirt_{i,0:t} | DirtSensor_{1:t}, Location_{1:t}). \end{aligned}$$

### Exercise 14.6.#RAOB

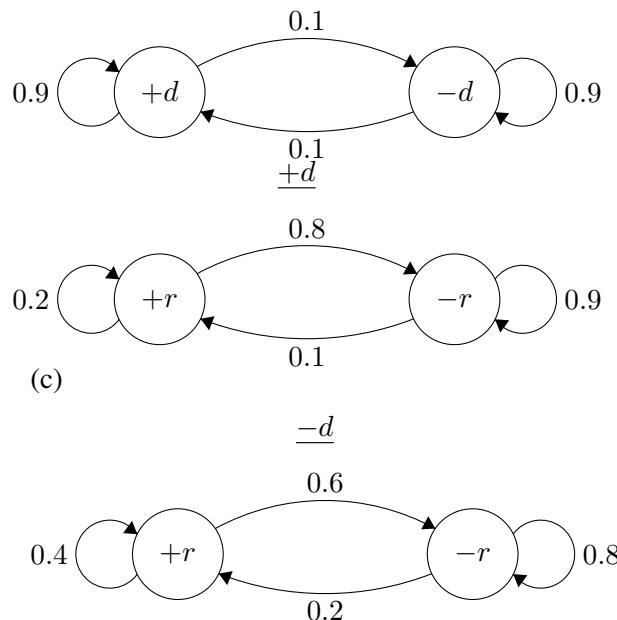
Consider a probability model  $\mathbf{P}(\mathbf{X}, \mathbf{Y}, Z, \mathbf{E})$ , where  $Z$  is a single query variable and evidence  $\mathbf{E} = \mathbf{e}$  is given. A basic Monte Carlo algorithm generates  $N$  samples (ideally) from  $\mathbf{P}(\mathbf{X}, \mathbf{Y}, Z | \mathbf{E} = \mathbf{e})$  and estimates the query probability  $P(Z = z | \mathbf{E} = \mathbf{e})$  from those samples. This gives an unbiased estimate but the variance may be quite large. The basic idea of **Rao-Blackwellization** in this context is to generate  $N$  samples of, say,  $(\mathbf{X}, Z)$  and, for each sample  $\mathbf{x}_j, z_j$ , to perform exact inference for  $\mathbf{P}(\mathbf{Y} | \mathbf{x}_j, z_j, \mathbf{e})$ . Explain how this yields an estimate for the query  $P(Z = z | \mathbf{E} = \mathbf{e})$  and show that the variance of the estimate is no larger than that from the original non-Rao-Blackwellized procedure.

TBC

### Exercise 14.6.#DRGT

In California, whether it rains or not from each day to the next forms a Markov chain (note: this is a terrible model for real weather). However, sometimes California is in a drought and sometimes it is not. Whether California is in a drought from each day to the next itself forms a Markov chain, and the state of this Markov chain affects the transition probabilities in the rain-or-shine Markov chain. The state diagram for droughts, rain given drought, and rain given no drought appear in Figure S14.8.

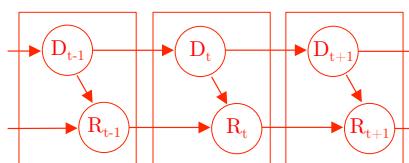
- a. Draw a dynamic Bayes net which encodes this behavior. Use variables  $D_{t-1}, D_t, D_{t+1}, R_{t-1}, R_t$ , and  $R_{t+1}$ . Assume that on a given day, it is determined whether or not there is a drought before it is determined whether or not it rains that day.
- b. Draw the CPT for  $D_t$  in the above DBN. Fill in the actual numerical probabilities.



**Figure S14.8** State diagrams for drought, ( $+d$ ) rain given drought, and ( $-d$ ) rain given no drought.

- Draw the CPT for  $R_t$  in the above DBN. Fill in the actual numerical probabilities.  
Suppose we are observing the weather on a day-to-day basis, but we cannot directly observe whether California is in a drought or not. We want to predict whether or not it will rain on day  $t+1$  given observations of whether or not it rained on days 1 through  $t$ .
- First, we need to determine whether California will be in a drought on day  $t+1$ . Derive a formula for  $P(D_{t+1}|r_{1:t})$  in terms of the given probabilities (the transition probabilities on the above state diagrams) and  $P(D_t|r_{1:t})$  (that is, you can assume we've already computed the probability there is a drought today given the weather over time).
- Now derive a formula for  $P(R_{t+1}|r_{1:t})$  in terms of  $P(D_{t+1}|r_{1:t})$  and the given probabilities.

- The DBN can be constructed as follows.



- Reading off the probabilities of the markov chain into a CPT, we get:

## Exercises 14 Probabilistic Reasoning over Time

| $P(D_t D_{t-1})$ |        |     |
|------------------|--------|-----|
| $+d_{t-1}$       | $+d_t$ | 0.9 |
| $+d_{t-1}$       | $-d_t$ | 0.1 |
| $-d_{t-1}$       | $+d_t$ | 0.1 |
| $-d_{t-1}$       | $-d_t$ | 0.9 |

c. Reading off the probabilities of the markov chain into a CPT, we get:

| $P(R_t R_{t-1}, D_t)$ |            |        |     |
|-----------------------|------------|--------|-----|
| $+d_t$                | $+r_{t-1}$ | $+r_t$ | 0.2 |
| $+d_t$                | $+r_{t-1}$ | $-r_t$ | 0.8 |
| $+d_t$                | $-r_{t-1}$ | $+r_t$ | 0.1 |
| $+d_t$                | $-r_{t-1}$ | $-r_t$ | 0.9 |
| $-d_t$                | $+r_{t-1}$ | $+r_t$ | 0.4 |
| $-d_t$                | $+r_{t-1}$ | $-r_t$ | 0.6 |
| $-d_t$                | $-r_{t-1}$ | $+r_t$ | 0.2 |
| $-d_t$                | $-r_{t-1}$ | $-r_t$ | 0.8 |

d.  $P(D_{t+1}|r_{1:t}) = \sum_{d_t} P(D_{t+1}|d_t)P(d_t|r_{1:t})$

e.  $P(R_{t+1}|r_{1:t}) = \sum_{d_{t+1}} P(d_{t+1}|r_{1:t})P(R_{t+1}|r_t, d_{t+1})$

# EXERCISES 15

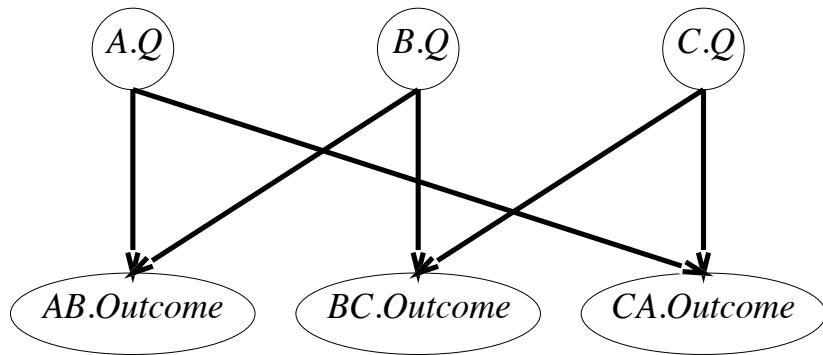
## PROBABILISTIC PROGRAMMING

### 15.1 Relational Probability Models

#### Exercise 15.1.#SOCC

Three soccer teams  $A$ ,  $B$ , and  $C$ , play each other once. Each match is between two teams, and can be won, drawn, or lost. Each team has a fixed, unknown degree of quality—an integer ranging from 0 to 3—and the outcome of a match depends probabilistically on the difference in quality between the two teams.

- a. Construct a relational probability model to describe this domain, and suggest numerical values for all the necessary probability distributions.
  - b. Construct the equivalent Bayesian network for the three matches.
  - c. Suppose that in the first two matches  $A$  beats  $B$  and draws with  $C$ . Using an exact inference algorithm of your choice, compute the posterior distribution for the outcome of the third match.
  - d. Suppose there are  $n$  teams in the league and we have the results for all but the last match. How does the complexity of predicting the last game vary with  $n$ ?
  - e. Investigate the application of MCMC to this problem. How quickly does it converge in practice and how well does it scale?
- 
- a. The classes are  $Team$ , with instances  $A$ ,  $B$ , and  $C$ , and  $Match$ , with instances  $AB$ ,  $BC$ , and  $CA$ . Each team has a quality  $Q$  and each match has a  $Team_1$  and  $Team_2$  and an  $Outcome$ . The team names for each match are of course fixed in advance. The prior over quality could be uniform and the probability of a win for team 1 should increase with  $Q(Team_1) - Q(Team_2)$ .
  - b. The random variables are  $A.Q$ ,  $B.Q$ ,  $C.Q$ ,  $AB.Outcome$ ,  $BC.Outcome$ , and  $CA.Outcome$ . The network is shown in Figure S15.1.
  - c. The exact result will depend on the probabilities used in the model. With any prior on quality that is the same across all teams, we expect that the posterior over  $BC.Outcome$  will show that  $C$  is more likely to win than  $B$ .
  - d. The inference cost in such a model will be  $O(2^n)$  because all the team qualities become coupled.
  - e. MCMC appears to do well on this problem, provided the probabilities are not too skewed. Our results show scaling behavior that is roughly linear in the number of teams, although we did not investigate very large  $n$ .



**Figure S15.1** Bayes net showing the dependency structure for the team quality and game outcome variables in the soccer model.

---

[[need exercises]]

## 15.2 Open-Universe Probability Models

---

[[need exercises]]

## 15.3 Keeping Track of a Complex World

---

[[need exercises]]

## 15.4 Programs as Probability Models

---

[[need exercises]]

# EXERCISES 16

## MAKING SIMPLE DECISIONS

### 16.1 Combining Beliefs and Desires under Uncertainty

#### Exercise 16.1.#ALMG

(Adapted from David Heckerman.) This exercise concerns the **Almanac Game**, which is used by decision analysts to calibrate numeric estimation. For each of the questions that follow, give your best guess of the answer, that is, a number that you think is as likely to be too high as it is to be too low. Also give your guess at a 25th percentile estimate, that is, a number that you think has a 25% chance of being too high, and a 75% chance of being too low. Do the same for the 75th percentile. (Thus, you should give three estimates in all—low, median, and high—for each question.)

- a. Number of passengers who flew between New York and Los Angeles in 1989.
- b. Population of Warsaw in 1992.
- c. Year in which Coronado discovered the Mississippi River.
- d. Number of votes received by Jimmy Carter in the 1976 presidential election.
- e. Age of the oldest living tree, as of 2002.
- f. Height of the Hoover Dam in feet.
- g. Number of eggs produced in Oregon in 1985.
- h. Number of Buddhists in the world in 1992.
- i. Number of deaths due to AIDS in the United States in 1981.
- j. Number of U.S. patents granted in 1901.

The correct answers appear after the last exercise of this chapter. From the point of view of decision analysis, the interesting thing is not how close your median guesses came to the real answers, but rather how often the real answer came within your 25% and 75% bounds. If it was about half the time, then your bounds are accurate. But if you’re like most people, you will be more sure of yourself than you should be, and fewer than half the answers will fall within the bounds. With practice, you can calibrate yourself to give realistic bounds, and thus be more useful in supplying information for decision making. Try this second set of questions and see if there is any improvement:

- a. Year of birth of Zsa Zsa Gabor.
- b. Maximum distance from Mars to the sun in miles.
- c. Value in dollars of exports of wheat from the United States in 1992.
- d. Tons handled by the port of Honolulu in 1991.

- e. Annual salary in dollars of the governor of California in 1993.
- f. Population of San Diego in 1990.
- g. Year in which Roger Williams founded Providence, Rhode Island.
- h. Height of Mt. Kilimanjaro in feet.
- i. Length of the Brooklyn Bridge in feet.
- j. Number of deaths due to automobile accidents in the United States in 1992.

It is interesting to create a histogram of accuracy on this task for the students in the class. It is also interesting to record how many times each student comes within, say, 10% of the right answer. Then you get a profile of each student: this one is an accurate guesser but overly cautious about bounds, etc.

### Exercise 16.1.#STPT

In 1713, Nicolas Bernoulli stated a puzzle, now called the St. Petersburg paradox, which works as follows. You have the opportunity to play a game in which a fair coin is tossed repeatedly until it comes up heads. If the first heads appears on the  $n$ th toss, you win  $2^n$  dollars.

- a. Show that the expected monetary value of this game is infinite.
  - b. How much would you, personally, pay to play the game?
  - c. Nicolas's cousin Daniel Bernoulli resolved the apparent paradox in 1738 by suggesting that the utility of money is measured on a logarithmic scale (i.e.,  $U(S_n) = a \log_2 n + b$ , where  $S_n$  is the state of having \$ $n$ ). What is the expected utility of the game under this assumption?
  - d. What is the maximum amount that it would be rational to pay to play the game, assuming that one's initial wealth is \$ $k$ ?
- a. The probability that the first heads appears on the  $n$ th toss is  $2^{-n}$ , so

$$EMV(L) = \sum_{n=1}^{\infty} 2^{-n} \cdot 2^n = \sum_{n=1}^{\infty} 1 = \infty$$

- b. Typical answers range between \$4 and \$100.
- c. Assume initial wealth (after paying  $c$  to play the game) of \$( $k - c$ ); then

$$U(L) = \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

Assume  $k - c = \$0$  for simplicity. Then

$$\begin{aligned} U(L) &= \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(2^n) + b) \\ &= \sum_{n=1}^{\infty} 2^{-n} \cdot an + b \\ &= 2a + b \end{aligned}$$

- d. The maximum amount  $c$  is given by the solution of

$$a \log_2 k + b = \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

For our simple case, we have

$$a \log_2 c + b = 2a + b$$

or  $c = \$4$ .

## 16.2 Utility Functions

---

### Exercise 16.2.#USED

Chris considers four used cars before buying the one with maximum expected utility. Pat considers ten cars and does the same. All other things being equal, which one is more likely to have the better car? Which is more likely to be disappointed with their car's quality? By how much (in terms of standard deviations of expected quality)?

Pat is more likely to have a better car than Chris because she has more information with which to choose. She is more likely to be disappointed, however, if she takes the expected utility of the best car at face value. Using the results of exercise 16.11, we can compute the expected disappointment to be about 1.54 times the standard deviation by numerical integration.

### Exercise 16.2.#ASSU

Assess your own utility for different incremental amounts of money by running a series of preference tests between some definite amount  $M_1$  and a lottery  $[p, M_2; (1-p), 0]$ . Choose different values of  $M_1$  and  $M_2$ , and vary  $p$  until you are indifferent between the two choices. Plot the resulting utility function.

This is an interesting exercise to do in class. Choose  $M_1 = \$100$ ,  $M_2 = \$100, \$1000, \$10000, \$1000000$ . Ask for a show of hands of those preferring the lottery at different values of  $p$ . Students will almost always display risk aversion, but there may be a wide spread in its onset. A curve can be plotted for the class by finding the smallest  $p$  yielding a majority vote for the lottery.

**Exercise 16.2.#ASSE**

Write a computer program to automate the process in Exercise 16.ASSU. Try your program out on several people of different net worth and political outlook. Comment on the consistency of your results, both for an individual and across individuals.

The program itself is pretty trivial. But note that there are some studies showing you get better answers if you ask subjects to move a slider to indicate a proportion, rather than asking for a probability number. So having a graphical user interface is an advantage. The main point of the exercise is to examine the data, expose inconsistent behavior on the part of the subjects, and see how people vary in their choices.

**Exercise 16.2.#SURC**

The Surprise Candy Company makes candy in two flavors: 75% are strawberry flavor and 25% are anchovy flavor. Each new piece of candy starts out with a round shape; as it moves along the production line, a machine randomly selects a certain percentage to be trimmed into a square; then, each piece is wrapped in a wrapper whose color is chosen randomly to be red or brown. 70% of the strawberry candies are round and 70% have a red wrapper, while 90% of the anchovy candies are square and 90% have a brown wrapper. All candies are sold individually in sealed, identical, black boxes.

Now you, the customer, have just bought a Surprise candy at the store but have not yet opened the box. Consider the three Bayes nets in Figure 16.1.

- a. Which network(s) can correctly represent  $\mathbf{P}(Flavor, Wrapper, Shape)$ ?
- b. Which network is the best representation for this problem?
- c. Does network (i) assert that  $\mathbf{P}(Wrapper|Shape) = \mathbf{P}(Wrapper)$ ?
- d. What is the probability that your candy has a red wrapper?
- e. In the box is a round candy with a red wrapper. What is the probability that its flavor is strawberry?
- f. A unwrapped strawberry candy is worth  $s$  on the open market and an unwrapped anchovy candy is worth  $a$ . Write an expression for the value of an unopened candy box.
- g. A new law prohibits trading of unwrapped candies, but it is still legal to trade wrapped candies (out of the box). Is an unopened candy box now worth more than less than, or the same as before?

- a. Networks (ii) and (iii) can represent this network but not (i).
  - (ii) is fully connected, so it can represent any joint distribution.
  - (iii) follows the generative story given in the problem: the flavor is determined (presumably) by which machine the candy is made by, then the shape is randomly cut, and the wrapper randomly chosen, the latter choice independently of the former.
  - (i) cannot represent this, as this network implies that the wrapper color and shape are marginally independent, which is not so: a round candy is likely to be strawberry, which is in turn likely to be wrapped in red, whilst conversely a square candy is likely

- to be anchovy which is likely to be wrapped in brown.
- Unlike (ii), (iii) is a polytree, which simplifies inference. Its edges also follow the causal direction, so probabilities will be easier to elicit. Indeed, the problem statement has already given them.
  - Yes, because Wrapper and Shape are d-separated.
  - Once we know the Flavor we know the probability its wrapper will be red or brown. So we marginalize Flavor out:

$$\begin{aligned}
 \mathbf{P}(Wrapper = red) &= \sum_f \mathbf{P}(Wrapper = red, Flavor = f) \\
 &= \sum_f \mathbf{P}(Flavor = f) \mathbf{P}(Wrapper = red | Flavor = f) \\
 &= 0.75 \times 0.7 + 0.25 \times 0.1 \\
 &= 0.55
 \end{aligned}$$

- We apply Bayes theorem, by first computing the joint probabilities

$$\begin{aligned}
 \mathbf{P}(Flavor = strawberry, Shape = round, Wrapper = red) &= \mathbf{P}(Flavor = strawberry) \times \mathbf{P}(Shape = round | Flavor = strawberry) \\
 &\quad \times \mathbf{P}(Wrapper = red | Flavor = strawberry) \\
 &= 0.75 \times 0.7 \times 0.7 \\
 &= 0.3675
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{P}(Flavor = anchovy, Shape = round, Wrapper = red) &= \mathbf{P}(Flavor = anchovy) \times \mathbf{P}(Shape = round | Flavor = anchovy) \\
 &\quad \times \mathbf{P}(Wrapper = red | Flavor = anchovy) \\
 &= 0.25 \times 0.1 \times 0.1 \\
 &= 0.0025
 \end{aligned}$$

Normalizing these probabilities yields that it is strawberry with probability  $0.3675 / (0.3675 + 0.0025) \approx 0.9932$ .

- Its value is the probability that you have a strawberry upon unwrapping times the value of a strawberry, plus the probability that you have a anchovy upon unwrapping times the value of an anchovy or

$$0.75s + 0.25a .$$

- The value is the same, by the axiom of decomposability.

### Exercise 16.2.#ALLP

Prove that the judgments  $B \succ A$  and  $C \succ D$  in the Allais paradox (page 538) violate the axiom of substitutability.

First observe that  $C \sim [0.25, A; 0.75, \$0]$  and  $D \sim [0.25, B; 0.75\$0]$ . This follows from the axiom of decomposability. But by substitutability this means that the preference ordering

between the lotteries  $A$  and  $B$  must be the same as that between  $C$  and  $D$ .

### Exercise 16.2.#ALLQ

Consider the Allais paradox described on page 538: an agent who prefers  $B$  over  $A$  (taking the sure thing), and  $C$  over  $D$  (taking the higher EMV) is not acting rationally, according to utility theory. Do you think this indicates a problem for the agent, a problem for the theory, or no problem at all? Explain.

As mentioned in the text, agents whose preferences violate expected utility theory demonstrate irrational behavior, that is they can be made either to accept a bet that is a guaranteed loss for them (the case of violating transitivity is given in the text), or reject a bet that is a guaranteed win for them. This indicates a problem for the agent.

### Exercise 16.2.#LOTT

Tickets to a lottery cost \$1. There are two possible prizes: a \$10 payoff with probability 1/50, and a \$1,000,000 payoff with probability 1/2,000,000. What is the expected monetary value of a lottery ticket? When (if ever) is it rational to buy a ticket? Be precise—show an equation involving utilities. You may assume current wealth of  $\$k$  and that  $U(S_k) = 0$ . You may also assume that  $U(S_{k+10}) = 10 \times U(S_{k+1})$ , but you may not make any assumptions about  $U(S_{k+1,000,000})$ . Sociological studies show that people with lower income buy a disproportionate number of lottery tickets. Do you think this is because they are worse decision makers or because they have a different utility function? Consider the value of contemplating the possibility of winning the lottery versus the value of contemplating becoming an action hero while watching an adventure movie.

The expected monetary value of the lottery  $L$  is

$$EMV(L) = \frac{1}{50} \times \$10 + \frac{1}{2000000} \times \$1000000 = \$0.70$$

Although  $\$0.70 < \$1$ , it is not *necessarily* irrational to buy the ticket. First we will consider just the utilities of the monetary outcomes, ignoring the utility of actually playing the lottery game. Using  $U(S_{k+n})$  to represent the utility to the agent of having  $n$  dollars more than the current state, and assuming that utility is linear for small values of money (i.e.,  $U(S_{k+n}) \approx n(U(S_{k+1}) - U(S_k))$  for  $-10 \leq n \leq 10$ ), the utility of the lottery is:

$$\begin{aligned} U(L) &= \frac{1}{50}U(S_{k+10}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \\ &\approx \frac{1}{5}U(S_{k+1}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \end{aligned}$$

This is more than  $U(S_{k+1})$  when  $U(S_{k+1,000,000}) > 1,600,000U(\$1)$ . Thus, for a purchase to be rational (when only money is considered), the agent must be quite risk-seeking. This would be unusual for low-income individuals, for whom the price of a ticket is non-trivial. It

is possible that some buyers do not internalize the magnitude of the very low probability of winning—to imagine an event is to assign it a “non-trivial” probability, in effect. Apparently, these buyers are better at internalizing the large magnitude of the prize. Such buyers are clearly acting irrationally.

Some people may feel their current situation is intolerable, that is,  $U(S_k) \approx U(S_{k\pm 1}) \approx u_\perp$ . Therefore the situation of having one dollar more or less would be equally intolerable, and it would be rational to gamble on a high payoff, even if one that has low probability.

Gamblers also derive pleasure from the excitement of the lottery and the temporary possession of at least a non-zero chance of wealth. So we should add to the utility of playing the lottery the term  $t$  to represent the thrill of participation. Seen this way, the lottery is just another form of entertainment, and buying a lottery ticket is no more irrational than buying a movie ticket. Either way, you pay your money, you get a small thrill  $t$ , and (most likely) you walk away empty-handed. (Note that it could be argued that doing this kind of decision-theoretic computation decreases the value of  $t$ . It is not clear if this is a good thing or a bad thing.)

### Exercise 16.2.#MICM

How much is a micromort worth to you? Devise a protocol to determine this. Ask questions based both on paying to avoid risk and being paid to accept risk.

The protocol would be to ask a series of questions of the form “which would you prefer” involving a monetary gain (or loss) versus an increase (or decrease) in a risk of death. For example, “would you pay \$100 for a helmet that would eliminate completely the one-in-a-million chance of death from a bicycle accident.”

### Exercise 16.2.#KMAX

Let continuous variables  $X_1, \dots, X_k$  be independently distributed according to the same probability density function  $f(x)$ . Prove that the density function for  $\max\{X_1, \dots, X_k\}$  is given by  $k f(x)(F(x))^{k-1}$ , where  $F$  is the cumulative distribution for  $f$ .

First observe that the cumulative distribution function for  $\max\{X_1, \dots, X_k\}$  is  $(F(x))^k$  since

$$\begin{aligned} P(\max\{X_1, \dots, X_k\} \leq x) &= P(X_1 \leq x, \dots, X_k \leq x) \\ &= P(X_1 \leq x) \dots P(X_k \leq x) \\ &= F(x)^k \end{aligned}$$

where the second to last step follows by independence. The result follows as the probability density function is the derivative of the cumulative distribution function.

### Exercise 16.2.#EXPU

Economists often make use of an exponential utility function for money:  $U(x) = -e^{-x/R}$ , where  $R$  is a positive constant representing an individual's risk tolerance. Risk tolerance reflects how likely an individual is to accept a lottery with a particular expected monetary value (EMV) versus some certain payoff. As  $R$  (which is measured in the same units as  $x$ ) becomes larger, the individual becomes less risk-averse.

- Assume Mary has an exponential utility function with  $R = \$500$ . Mary is given the choice between receiving \$500 with certainty (probability 1) or participating in a lottery which has a 60% probability of winning \$5000 and a 40% probability of winning nothing. Assuming Mary acts rationally, which option would she choose? Show how you derived your answer.
- Consider the choice between receiving \$100 with certainty (probability 1) or participating in a lottery which has a 50% probability of winning \$500 and a 50% probability of winning nothing. Approximate the value of  $R$  (to 3 significant digits) in an exponential utility function that would cause an individual to be indifferent to these two alternatives. (You might find it helpful to write a short program to help you solve this problem.)

- a. Getting \$400 for sure has expected utility

$$-e^{-400/400} = -1/e \approx -0.3679$$

while the getting \$5000 with probability 0.6 and \$0 otherwise has expected utility

$$0.6 - e^{-5000/400} + 0.5 - e^{-0/400} = -(0.6e^{-12.5} + 0.5) \approx -0.5000$$

so one would prefer the sure bet.

- b. We want to find  $R$  such that

$$e^{-100/R} = 0.5e^{-500/R} + 0.5$$

Solving this numerically, we find  $R = 152$  up to 3sf.

### Exercise 16.2.#FGAM

Alex is given the choice between two games. In Game 1, a fair coin is flipped and if it comes up heads, Alex receives \$100. If the coin comes up tails, Alex receives nothing. In Game 2, a fair coin is flipped twice. Each time the coin comes up heads, Alex receives \$50, and Alex receives nothing for each coin flip that comes up tails. Assuming that Alex has a monotonically increasing utility function for money in the range  $[\$0, \$100]$ , show mathematically that if Alex prefers Game 2 to Game 1, then Alex is risk averse (at least with respect to this range of monetary amounts).

Since Alex prefers Game 2 to Game 1, we have

$$0.5U(\$0) + 0.5U(\$100) < U(\$50).$$

Since  $\$50 = 0.5\$0 + 0.5\$100$  is the expected monetary value of the Game 1 lottery, Alex is risk averse.

### Exercise 16.2.#188-UTIL

Consider the following lotteries:

- $L_1 = [1, 1]$ .
- $L_2 = [0.5, 2; 0.5, 0]$ .
- $L_3 = [1, 2]$ .

Four students have expressed their preferences over these lotteries as follows:

- Adam is indifferent between lottery  $L_2$  and lottery  $L_1$ .
- Becky prefers lottery  $L_1$  to lottery  $L_2$ .
- Charles is indifferent between lottery  $L_3$  and lottery  $L_2$ .
- Diana prefers lottery  $L_2$  to lottery  $L_1$ .

Match each student with a utility function that is consistent with their stated preferences. Each student has a different utility function.

- a. Which student has the utility function of  $U(x) = x^2$
  - b. Which student has the utility function of  $U(x) = x$
  - c. Which student has the utility function of  $U(x) = \sqrt{x}$
  - d. Which student has the utility function of none of the ones above
- 
- a. Diana, since  $0.5 \times 0^2 + 0.5 \times 2^2 > 1^2$
  - b. Adam, since  $0.5 \times 0 + 0.5 \times 2 = 1$
  - c. Becky, since  $0.5 \times \sqrt{0} + 0.5 \times \sqrt{2} < \sqrt{1}$
  - d. Charles, since none of the three utility functions satisfies  $0.5 \times U(0) + 0.5 \times U(2) = U(2)$

### Exercise 16.2.#188-UTPR

True or False: Assume Agent 1 has a utility function  $U_1$  and Agent 2 has a utility function  $U_2$ . If  $U_1 = k_1 U_2 + k_2$  with  $k_1 > 0, k_2 > 0$  then Agent 1 and Agent 2 have the same preferences.

True. For any  $a, b : U_2(a) > U_2(b)$  equivalent to  $k_1 U_2(a) > k_1 U_2(b)$  since  $k_1 > 0$ . Then we have  $k_1 U_2(a) > k_1 U_2(b)$  equivalent to  $k_1 U_2(a) + k_2 > k_1 U_2(b) + k_2$  for any  $k_2$

### Exercise 16.2.#188-PELL

Origin:

sp16\_midterm\_utilities

PacLad and PacLass are arguing about the value of eating certain numbers of pellets. Neither knows their exact utility functions, but it is known that they are both rational and that PacLad prefers eating more pellets to eating fewer pellets. For any  $n$ , let  $E_n$  be the event of eating  $n$  pellets. So for PacLad, if  $m \geq n$ , then  $E_m \succeq E_n$ . For any  $n$  and any  $k < n$ , let  $L_{n\pm k}$  refer to a lottery between  $E_{n-k}$  and  $E_{n+k}$ , each with probability  $\frac{1}{2}$ .

*Reminder:* For events  $A$  and  $B$ ,  $A \sim B$  denotes that the agent is indifferent between  $A$  and  $B$ , while  $A \succ B$  denotes that  $A$  is preferred to  $B$ .

a. Are the following statements guaranteed to be true?

- (i) Under PacLad's preferences, for any  $n, k$ ,  $L_{n\pm k} \sim E_n$ .
- (ii) Under PacLad's preferences, for any  $k$ , if  $m \geq n$ , then  $L_{m\pm k} \succeq L_{n\pm k}$
- (iii) Under PacLad's preferences, for any  $k, l$ , if  $m \geq n$ , then  $L_{m\pm k} \succeq L_{n\pm l}$ .

b. To decouple from the previous part, suppose we are given now that under PacLad's preferences, for any  $n, k$ ,  $L_{n\pm k} \sim E_n$ . Suppose PacLad's utility function in terms of the number of pellets eaten is  $U_1$ . For each of the following, suppose PacLass's utility function,  $U_2$ , is defined as given in terms of  $U_1$ . Choose **all** statements which are guaranteed to be true of PacLass's preferences under each definition. You should assume that all utilities are positive (greater than 0).

- (i)  $U_2(n) = aU_1(n) + b$  for some positive integers  $a, b$ 
  - (A)  $E_4 \succeq E_3$
  - (B)  $L_{4\pm 1} \sim L_{4\pm 2}$
  - (C)  $L_{4\pm 1} \succ E_4$
- (ii)  $U_2(n) = \frac{1}{U_1(n)}$ 
  - (A)  $E_4 \succeq E_3$
  - (B)  $L_{4\pm 1} \sim L_{4\pm 2}$
  - (C)  $L_{4\pm 1} \succ E_4$

a. (i) False

All we know is that PacLad's utility is an increasing function of the number of pellets. One utility function consistent with this is  $U(E_n) = 2^n$ . Then the expected utility of  $L_{2\pm 1}$  is  $\frac{1}{2}U(E_1) + \frac{1}{2}U(E_3) = \frac{1}{2}(2 + 8) = 5$ . Since  $U(E_2) = 2^2 = 4$ ,  $L_{2\pm 1} \succ E_2$ . The only class of utility functions that give the guarantee that this claim is true is linear utility functions. This is a mathematical way of writing the PacLad is risk-neutral; but this is not given as an assumption in the problem.  $2^n$  is a good counterexample because it is a risk-seeking utility function. A risk-avoiding utility function would have worked just as well.

(ii) True

The expected utility of  $L_{m\pm k}$  is  $\frac{1}{2}U(E_{m-k}) + \frac{1}{2}U(E_{m+k})$ , and that of  $L_{n\pm k}$  is  $\frac{1}{2}U(E_{n-k}) + \frac{1}{2}U(E_{n+k})$ . Since  $m - k \geq n - k$ ,  $E_{m-k} \succeq E_{n-k}$ , so  $U(E_{m-k}) \geq U(E_{n-k})$ . Similarly, since  $m + k \geq n + k$ ,  $E_{m+k} \succeq E_{n+k}$ , so  $U(E_{m+k}) \geq U(E_{n+k})$ . Thus  $\frac{1}{2}U(E_{m-k}) + \frac{1}{2}U(E_{m+k}) \geq \frac{1}{2}U(E_{n-k}) + \frac{1}{2}U(E_{n+k})$  and there-

fore  $L_{m\pm k} \succeq L_{n\pm k}$ .

(iii) False

Consider again the utility function  $U(E_n) = 2^n$ . It is a risk-seeking utility function as mentioned in part (i), so we should expect that if this were PacLad's utility function, he would prefer a lottery with higher variance (i.e. a higher  $k$  value). So for a counterexample, we look to  $L_{3\pm 1}$  and  $L_{3\pm 2}$  (i.e.  $m = n = 3$ ,  $k = 1$ ,  $l = 2$ ). The expected utility of  $L_{3\pm 1}$  is  $\frac{1}{2}U(E_2) + \frac{1}{2}U(E_4) = \frac{1}{2}(4 + 16) = 10$ . The expected utility of  $L_{3\pm 2}$  is  $\frac{1}{2}U(E_1) + \frac{1}{2}U(E_5) = \frac{1}{2}(2 + 32) = 17 > 10$ . Thus  $L_{n\pm l} \succ L_{m\pm k}$ . Once again, this is a statement that would only be true for a risk-neutral utility function. A risk-avoiding utility function could also have been used for a counterexample.

b. (i) A, B

The guarantee that under PacLad's preferences for any  $n, k$ ,  $L_{n\pm k} \sim E_n$  means that PacLad is risk-neutral and therefore his utility function is linear. An affine transformation, as this  $aU_1(n) + b$  is called, of a linear function is still a linear function, so we have that PacLass's utility function is also linear and thus she is also risk-neutral. Therefore she is indifferent to the variance of lotteries with the same expectation (first option) and she does *not* prefer a lottery to deterministically being given the expectation of that lottery (**not** third option). Since  $a$  is positive,  $U_2$  is also an increasing function (second option).

(ii) C

Since  $U_1$  is an increasing function,  $U_2$  is decreasing, and thus the preferences over deterministic outcomes are flipped (**not** second option).

The expected utility of  $L_{4\pm 1}$  is  $\frac{1}{2}(U_2(3) + U_2(5)) = \frac{1}{2}\left(\frac{1}{U_1(3)} + \frac{1}{U_1(5)}\right)$ . We know that  $U_1$  is linear, so write  $U_1(n) = an + b$  for some  $a, b$ . Then substituting this into this expression for  $\mathbb{E}[U_2(L_{4\pm 1})]$  and simplifying algebraically yields  $\frac{1}{2}\left(\frac{8a+2b}{15a^2+8ab+b^2}\right) = \frac{4a+b}{15a^2+8ab+b^2}$ . By the same computation for  $L_{4\pm 2}$ , we get  $\mathbb{E}[U_2(L_{4\pm 2})] = \frac{4a+b}{12a^2+8ab+b^2}$ . Since we only know that  $U_1$  is increasing and linear, the only constraint on  $a$  and  $b$  is that  $a$  is positive. So let  $a = 1, b = 0$ . Then  $\mathbb{E}[U_2(L_{4\pm 2})] = \frac{1}{3} > \frac{4}{15} = \mathbb{E}[U_2(L_{4\pm 1})]$  and thus  $L_{4\pm 2} \succ L_{4\pm 1}$  (**not** first option). Similarly, for this  $U_1$ ,  $U_2(4) = \frac{1}{U_1(4)} = \frac{1}{4} < \frac{1}{3} = \mathbb{E}[U_2(L_{4\pm 2})]$  and thus  $L_{4\pm 1} \succ E_4$  (third option).

What follows is a more general argument that could have been used to answer this question if particular numbers were not specified.

In order to determine PacLass's attitude toward risk, we take the second derivative of  $U_2$  with respect to  $n$ . By the chain rule,  $\frac{dU_2(n)}{dn} = \frac{dU_2(n)}{dU_1(n)} \cdot \frac{dU_1(n)}{dn}$ . Since  $U_1$  is an increasing linear function of  $n$ ,  $\frac{dU_1(n)}{dn}$  is some positive constant  $a$ , so  $\frac{dU_2(n)}{dn} = a \frac{dU_2(n)}{dU_1(n)} = -a \frac{1}{(U_1(n))^2}$ . Taking the derivative with respect to  $n$  again and using the chain rule yields  $\frac{d^2U_2(n)}{dn^2} = \frac{d}{dU_1(n)} \left( -a \frac{1}{(U_1(n))^2} \right) \cdot \frac{dU_1(n)}{dn} = \frac{1}{2}a^2 \frac{1}{(U_1(n))^3}$ .  $U_1$  is always positive, so this is a positive number and thus the second derivative of PacLass's utility function is everywhere positive. This means the utility function is strictly convex (equivalently "concave up"), and thus all secant lines on the plot of the curve lie above the curve itself.

In general, strictly convex utility functions are risk-seeking. To see this, consider  $L_{n\pm k}$  and  $E_n$ . The expected utility of  $L_{n\pm k}$  is  $\frac{1}{2}U_2(n - k) + \frac{1}{2}U_2(n + k)$ , which corresponds to the midpoint of the secant line drawn between the points  $(n - k, U_2(n - k))$  and  $(n + k, U_2(n + k))$ , which both lie on the curve. That point is  $(n, \mathbb{E}[U(L_{n\pm k})]) = (n, \frac{1}{2}U_2(n - k) + \frac{1}{2}U_2(n + k))$ . The utility of  $E_n$  is  $U(n)$ , which lies on the curve at the point  $(n, U_2(n))$ . Since  $U_2$  is strictly convex, the secant line lies above the curve, so we must have  $\mathbb{E}[U_2(L_{n\pm k})] > U(n)$ .

With that proof that PacLass is risk-seeking, we can address the remaining two options: she is not indifferent to the variance of a lottery (**not** the first option), and she prefers the lottery over the deterministic outcome (the third option).

### Exercise 16.2.#188-INSU

PacBaby just found a \$100 bill—it is the only thing she owns. Ghosts are nice enough not to kill PacBaby, but when they find PacBaby they will steal all her money. The probability of the ghosts finding PacBaby is 20%. PacBaby's utility function is  $U(x) = \log(1 + x)$  (this is the natural logarithm, i.e.,  $\log e^x = x$ ), where  $x$  is the total monetary value she owns. When PacBaby gets to keep the \$100 (ghosts don't find her) her utility is  $U(\$100) = \log(101)$ . When PacBaby loses the \$100 (per the ghosts taking it from her) her utility is  $U(\$0) = \log(1 + 0) = 0$ .

- What is the expected utility for PacBaby?
  - Pacgressive offers theft insurance: if PacBaby pays an insurance premium of \$30, then they will reimburse PacBaby \$70 if the ghosts steal all her money (after paying \$30 in insurance, she would only have \$70 left). What is the expected utility for PacBaby if she takes insurance? For PacBaby to maximize her expected utility should she take this insurance?
  - In the above scenario, what is the expected *monetary* value of selling the insurance from Pacgressive's point of view?
- 
- $0.8 \times \log(101) + 0.2 \times \log(1) = 0.8 \times \log(101) + 0.2 \times 0 \approx 3.6921$
  - When taking insurance, PacBaby's expected utility equals  $0.8 \log(1 + 70) + 0.2 \log(1 + 70) = \log(71) \approx 4.2627$ . Yes, PacBaby should take the insurance.
  - The expected monetary value equals  $0.8 \times 30 + 0.2 \times (-40) = 16$ .

## 16.3 Multiattribute Utility Functions

### Exercise 16.3.#PREI

Show that if  $X_1$  and  $X_2$  are preferentially independent of  $X_3$ , and  $X_2$  and  $X_3$  are preferentially independent of  $X_1$ , then  $X_3$  and  $X_1$  are preferentially independent of  $X_2$ .

The complete proof is given by Keeney and Raiffa (1976).

## 16.4 Decision Networks

---

### Exercise 16.4.#APID

This exercise completes the analysis of the airport-siting problem in Figure 16.6.

- a. Provide reasonable variable domains, probabilities, and utilities for the network, assuming that there are three possible sites.
- b. Solve the decision problem.
- c. What happens if changes in technology mean that each aircraft generates half the noise?
- d. What if noise avoidance becomes three times more important?
- e. Calculate the VPI for *AirTraffic*, *Litigation*, and *Construction* in your model.

This exercise can be solved using an influence diagram package such as IDEAL. The specific values are not especially important. Notice how the tedium of encoding all the entries in the utility table cries out for a system that allows the additive, multiplicative, and other forms sanctioned by MAUT.

One of the key aspects of the fully explicit representation in Figure 16.5 is its amenability to change. By doing this exercise as well as Exercise 16.9, students will augment their appreciation of the flexibility afforded by declarative representations, which can otherwise seem tedious.

- a. For this part, one could use symbolic values (high, medium, low) for all the variables and not worry too much about the exact probability values, or one could use actual numerical ranges and try to assess the probabilities based on some knowledge of the domain. Even with three-valued variables, the cost CPT has 54 entries.
- b. This part almost certainly should be done using a software package.
- c. If each aircraft generates half as much noise, we need to adjust the entries in the *Noise* CPT.
- d. If the noise attribute becomes three times more important, the utility table entries must all be altered. If an appropriate (e.g., additive) representation is available, then one would only need to adjust the appropriate constants to reflect the change.
- e. This part should be done using a software package. Some packages may offer VPI calculation already. Alternatively, one can invoke the decision-making package repeatedly to do all the what-if calculations of best actions and their utilities, as required in the VPI formula. Finally, one can write general-purpose VPI code as an add-on to a decision-making package.

### Exercise 16.4.#ARPT

Repeat Exercise 16.APID, using the action-utility representation shown in Figure 16.7.

The information associated with the utility node in Figure 16.6 is an action-value table, and can be constructed simply by averaging out the *Deaths*, *Noise*, and *Cost* nodes in Figure 16.5. As explained in the text, modifications to aircraft noise levels or to the importance

of noise do not result in simple changes to the action-value table. Probably the easiest way to do it is to go back to the original table in Figure 16.5. The exercise therefore illustrates the tradeoffs involved in using compiled representations.

#### Exercise 16.4.#ARPS

For either of the airport-siting diagrams from Exercises ?? and ??, to which conditional probability table entry is the utility most sensitive, given the available evidence?

The answer to this exercise depends on the probability values chosen by the student.

#### Exercise 16.4.#BACO

Modify and extend the Bayesian network code in the code repository to provide for creation and evaluation of decision networks and the calculation of information value.

This is relatively straightforward in the AIMA2e code release. We need to add node types for action nodes and utility nodes; we need to be able to run standard Bayes net inference on the network given fixed actions, in order to compute the posterior expected utility; and we need to write an “outer loop” that can try all possible actions to find the best. Given this, adding VPI calculation is straightforward, as described in the answer to Exercise 16.8.

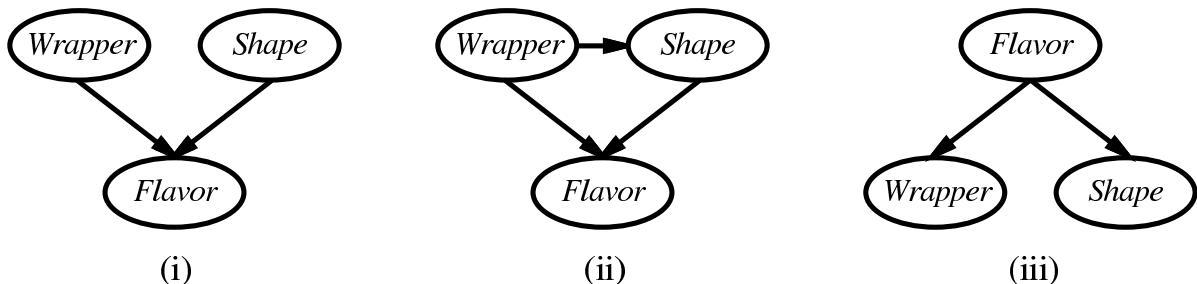
#### Exercise 16.4.#TXBK

Consider a student who has the choice to buy or not buy a textbook for a course. We’ll model this as a decision problem with one Boolean decision node,  $B$ , indicating whether the agent chooses to buy the book, and two Boolean chance nodes,  $M$ , indicating whether the student has mastered the material in the book, and  $P$ , indicating whether the student passes the course. Of course, there is also a utility node,  $U$ . A certain student, Sam, has an additive utility function: 0 for not buying the book and -\$100 for buying it; and \$2000 for passing the course and 0 for not passing. Sam’s conditional probability estimates are as follows:

$$\begin{aligned}P(p|b, m) &= 0.9 & P(m|b) &= 0.9 \\P(p|b, \neg m) &= 0.5 & P(m|\neg b) &= 0.7 \\P(p|\neg b, m) &= 0.8 \\P(p|\neg b, \neg m) &= 0.3\end{aligned}$$

You might think that  $P$  would be independent of  $B$  given  $M$ , But this course has an open-book final—so having the book helps.

- a. Draw the decision network for this problem.
- b. Compute the expected utility of buying the book and of not buying it.
- c. What should Sam do?
  - a. See Figure S16.2.



Three proposed Bayes nets for the Surprise Candy problem, Exercise 16.SURC.

- b. For each of  $B = b$  and  $B = \neg b$ , we compute  $P(p|B)$  and thus  $P(\neg p|B)$  by marginalizing out  $M$ , then use this to compute the expected utility.

$$\begin{aligned}
 P(p|b) &= \sum_m P(p|b, m)P(m|b) \\
 &= 0.9 \times 0.9 + 0.5 \times 0.1 \\
 &\equiv 0.86 \\
 P(p|\neg b) &= \sum_m P(p|\neg b, m)P(m|\neg b) \\
 &= 0.8 \times 0.7 + 0.3 \times 0.3 \\
 &\equiv 0.65
 \end{aligned}$$

The expected utilities are thus:

$$\begin{aligned}
 EU[b] &= \sum_p P(p|b)U(p, b) \\
 &= 0.86(2000 - 100) + 0.14(-100) \\
 &= 1620 \\
 EU[\neg b] &= \sum_p P(p|\neg b)U(p, \neg b) \\
 &= 0.65 \times 2000 + 0.14 \times 0 \\
 &= 1300
 \end{aligned}$$

- c. Buy the book, Sam.

### **Exercise 16.4.#188-VALE**

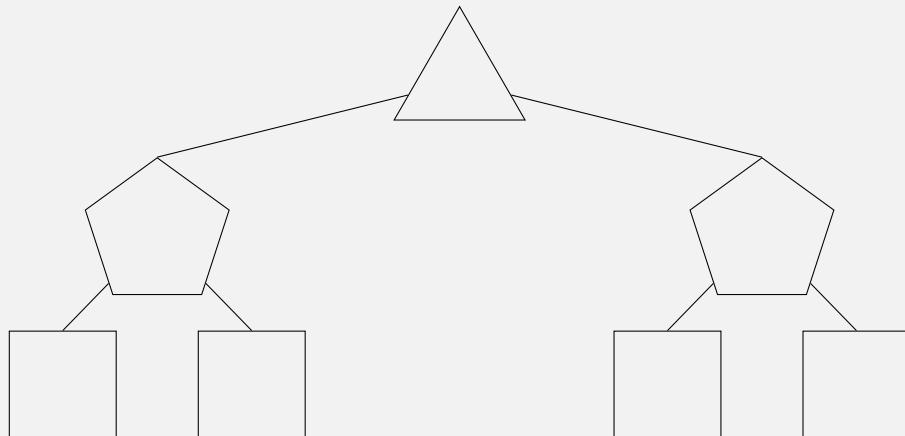
Origin:

su19 midterm2 decision networks

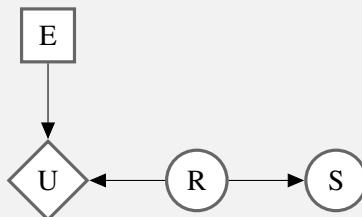
Valerie has just found a cookie on the ground. She is concerned that the cookie contains raisins, which she really dislikes but she still wants to eat the cookie. If she eats the cookie and it contains raisins she will receive a utility of  $-100$  and if the cookie doesn't contain raisins she will receive a utility of  $10$ . If she doesn't eat the cookie she will get  $0$  utility. The

cookie contains raisins with probability 0.1.

- a. We want to represent this decision network as an expectimax game tree. Fill in the nodes of the tree below, with the top node representing her maximizing choice.

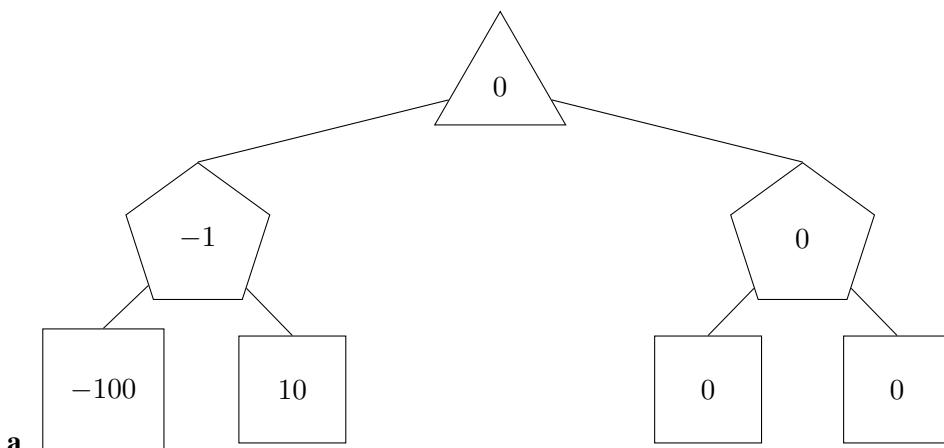


- b. Should Valerie eat the cookie?
- c. Valerie can now smell the cookie to judge whether it has raisins before she eats it. However, since she dislikes raisins she does not have much experience with them and cannot recognize their smell well. As a result she will incorrectly identify raisins when there are no raisins with probability 0.2 and will incorrectly identify no raisins when there are raisins with probability 0.3. This decision network can be represented by the diagram below where E is her choice to eat, U is her utility earned, R is whether the cookie contains raisins, and S is her attempt at smelling.



Valerie has just smelled the cookie and she thinks it doesn't have raisins. Write the probability, X, that the cookie has raisins given that she smelled no raisins as a simplest form fraction or decimal.

- d. What is her maximum expected utility, Y given that she smelled no raisins? You can answer in terms of X or as a simplest form fraction or decimal.
- e. What is the Value of Perfect Information (VPI) of smelling the cookie? You can answer in terms of X and Y or as a simplest form fraction or decimal.



b. No

c. 0.04

$$P(+r| -s) = \frac{P(-s|r)P(+r)}{P(-s)} = \frac{P(-s|r)P(+r)}{P(-s|r)P(+r) + P(-s|-r)P(-r)} = \frac{.3*.1}{.3*.1 + .8*.9} = \frac{.03}{.75} = .04$$

d.  $-100X + 10(1 - X)$  or 5.6

$$\begin{aligned} MEU(-s) &= \max(MEU(eating| -s), MEU(noteating| -s)) = \\ &= \max(P(+r| -s)*EU(eating, +r) + P(-r| -s)*EU(eating, -r), MEU(noteating)) = \\ &= \max(X * (-100) + (1 - X) * 10, 0) = \\ &= X * 100 + (1 - X) * 10 \end{aligned}$$

e.  $0.75 \times Y$  or 4.2

$$VPI(S) = MEU(S) - MEU(\{\})$$

$$MEU(S) = P(-s)MEU(-s) + P(+s)MEU(+s)$$

$$P(-s) = .75 \text{ from part (c), } MEU(-s) = Y$$

$MEU(+s) = 0$  because it was better for her to not eat the raisin without knowing anything, smelling raisins will only make it more likely for the cookie to have raisins and it will still be best for her to not eat and earn a utility of 0. Note this means we do not have to calculate  $P(+s)$ .

$$MEU(\{\}) = 0$$

$$VPI(S) = .75 * Y + 0 - 0 = .75 * Y$$

## 16.5 The Value of Information

### Exercise 16.5.#VPIX

(Adapted from Pearl (1989).) A used-car buyer can decide to carry out various tests with various costs (e.g., kick the tires, take the car to a qualified mechanic) and then, depending on the outcome of the tests, decide which car to buy. We will assume that the buyer is deciding whether to buy car  $c_1$ , that there is time to carry out at most one test, and that  $t_1$  is the test of  $c_1$  and costs \$50.

A car can be in good shape (quality  $q^+$ ) or bad shape (quality  $q^-$ ), and the tests might help indicate what shape the car is in. Car  $c_1$  costs \$1,500, and its market value is \$2,000 if it

is in good shape; if not, \$700 in repairs will be needed to make it in good shape. The buyer's estimate is that  $c_1$  has a 70% chance of being in good shape.

- Draw the decision network that represents this problem.
  - Calculate the expected net gain from buying  $c_1$ , given no test.
  - Tests can be described by the probability that the car will pass or fail the test given that the car is in good or bad shape. We have the following information:  
 $P(\text{pass}(c_1, t_1) | q^+(c_1)) = 0.8$   
 $P(\text{pass}(c_1, t_1) | q^-(c_1)) = 0.35$
- Use Bayes' theorem to calculate the probability that the car will pass (or fail) its test and hence the probability that it is in good (or bad) shape given each possible test outcome.
- Calculate the optimal decisions given either a pass or a fail, and their expected utilities.
  - Calculate the value of information of the test, and derive an optimal conditional plan for the buyer.

This question is a simple exercise in sequential decision making, and helps in making the transition to Chapter 17. It also emphasizes the point that the value of information is computed by examining the *conditional* plan formed by determining the best action for each possible outcome of the test. It may be useful to introduce "decision trees" (as the term is used in the decision analysis literature) to organize the information in this question. (See Pearl (1988), Chapter 6.) Each part of the question analyzes some aspect of the tree. Incidentally, the question assumes that utility and monetary value coincide, and ignores the transaction costs involved in buying and selling.

- The decision network is shown in Figure S??.
- The expected net gain in dollars is

$$P(q^+)(2000 - 1500) + P(q^-)(2000 - 2200) = 0.7 \times 500 + 0.3 \times -200 = 290$$

- The question could probably have been stated better: Bayes' theorem is used to compute  $P(q^+|Pass)$ , etc., whereas conditionalization is sufficient to compute  $P(Pass)$ .

$$\begin{aligned} P(Pass) &= P(Pass|q^+)P(q^+) + P(Pass|q^-)P(q^-) \\ &= 0.8 \times 0.7 + 0.35 \times 0.3 = 0.665 \end{aligned}$$

Using Bayes' theorem:

$$P(q^+|Pass) = \frac{P(Pass|q^+)P(q^+)}{P(Pass)} = \frac{0.8 \times 0.7}{0.665} \approx 0.8421$$

$$P(q^-|Pass) \approx 1 - 0.8421 = 0.1579$$

$$P(q^+|\neg Pass) = \frac{P(\neg Pass|q^+)P(q^+)}{P(\neg Pass)} = \frac{0.2 \times 0.7}{0.335} \approx 0.4179$$

$$P(q^-|\neg Pass) \approx 1 - 0.4179 = 0.5821$$

- d. If the car passes the test, the expected value of buying is

$$\begin{aligned} P(q^+|Pass)(2000 - 1500) + P(q^-|Pass)(2000 - 2200) \\ = 0.8421 \times 500 + 0.1579 \times -200 = 378.92 \end{aligned}$$

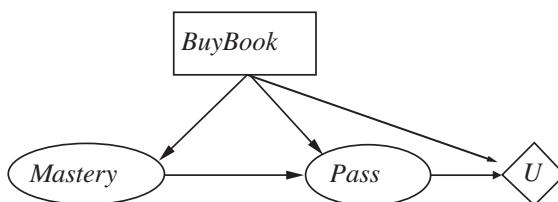
Thus buying is the best decision given a pass. If the car fails the test, the expected value of buying is

$$\begin{aligned} P(q^+|\neg Pass)(2000 - 1500) + P(q^-|\neg Pass)(2000 - 2200) \\ = 0.4179 \times 500 + 0.5821 \times -200 = 92.53 \end{aligned}$$

Buying is again the best decision.

- e. Since the action is the same for both outcomes of the test, the test itself is worthless (if it is the only possible test) and the optimal plan is simply to buy the car without the test. (This is a trivial conditional plan.) For the test to be worthwhile, it would need to be more discriminating in order to reduce the probability  $P(q^+|\neg Pass)$ . The test would also be worthwhile if the market value of the car were less, or if the cost of repairs were more.

An interesting additional exercise is to prove the general proposition that if  $\alpha$  is the best action for all the outcomes of a test then it must be the best action in the absence of the test outcome.



**Figure S16.2** A decision network for the book-buying problem.

### Exercise 16.5.#NNVP

Recall the definition of *value of information* in Section 16.6.

- a. Prove that the value of information is nonnegative and order independent.
- b. Explain why it is that some people would prefer not to get some information—for example, not wanting to know the sex of their baby when an ultrasound is done.
- c. A function  $f$  on sets is **submodular** if, for any element  $x$  and any sets  $A$  and  $B$  such that  $A \subseteq B$ , adding  $x$  to  $A$  gives a greater increase in  $f$  than adding  $x$  to  $B$ :

$$A \subseteq B \Rightarrow (f(A \cup \{x\}) - f(A)) \geq (f(B \cup \{x\}) - f(B)).$$

Submodularity captures the intuitive notion of *diminishing returns*. Is the value of information, viewed as a function  $f$  on sets of possible observations, submodular? Prove this or find a counterexample.

- a. Intuitively, the value of information is nonnegative because in the worst case one could simply ignore the information and act as if it was not available. A formal proof therefore begins by showing that this policy results in the same expected utility. The formula for the value of information is

$$VPI_E(E_j) = \left( \sum_k P(E_j = e_{jk}|E) EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) \right) - EU(\alpha|E)$$

If the agent does  $\alpha$  given the information  $E_j$ , its expected utility is

$$\sum_k P(E_j = e_{jk}|E) EU(\alpha|E, E_j = e_{jk}) = EU(\alpha|E)$$

where the equality holds because the LHS is just the conditionalization of the RHS with respect to  $E_j$ . By definition,

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) \geq EU(\alpha|E, E_j = e_{jk})$$

hence  $VPI_E(E_j) \geq 0$ .

- b. One explanation is that people are aware of their own irrationality and may want to avoid making a decision on the basis of the extra information. Another might be that the value of information is small compared to the value of surprise—for example, many people prefer not to know in advance what their birthday present is going to be.
- c. Value of information is not submodular in general. Suppose that  $A$  is the empty set and  $B$  is the set  $Y = 1$ ; and suppose that the optimal decision remains unchanged unless both  $X = 1$  and  $Y = 1$  are observed.

### Exercise 16.5.#HUNT

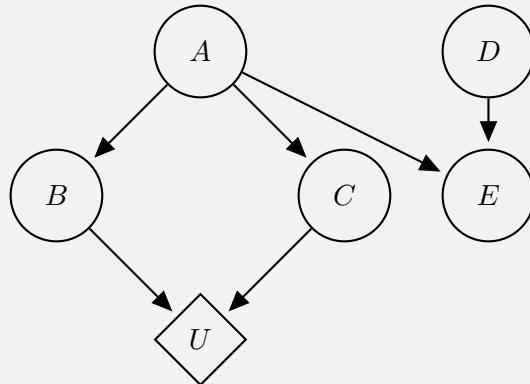
Figure ?? shows a myopic **function** INFORMATION-GATHERING-AGENT( $t$ ) **returns** hat chooses the next evidence variable to observe according to the ratio  $VPI(E_j)/C(E_j)$ , where  $C(E_j)$  is the cost of observing  $E_j$ . The optimal algorithm for the treasure-hunt problem ranks the prospects according to  $P(E_j)/C(E_j)$ , where  $P(E_j)$  is the probability that the treasure is in location  $j$ .

- Calculate  $VPI(E_j)$ , assuming the treasure is valued at  $v$ .
- Determine if the two algorithms give the same behavior for the treasure-hunt problem.
- Does the value  $v$  matter? Explain how to incorporate it into the treasure-hunt algorithm and give a proof of optimality for the new algorithm.

[[TBC]]

**Exercise 16.5.#188-DSEP**

Consider a decision network with the following structure, where node  $U$  is the utility:



- In the graph above, how do we know if a node is guaranteed to have  $VPI = 0$ ?
- Can any node be guaranteed to have  $VPI > 0$ ? Why or why not?

- Any node which is d-separated from the parents of the utility node is guaranteed to have 0 VPI. In the graph above,  $VPI(D) = 0$ ,  $VPI(E|A) = 0$ ,  $VPI(A|B, C) = 0$ ,
- No, we can never guarantee any node in the graph to have  $VPI > 0$ , since here we have no assumptions about the utility function. We could have  $U(B, C) = 0$ , in which case  $MEU$  will always be 0 regardless of the information we have and thus  $VPI$  is 0 for all nodes.

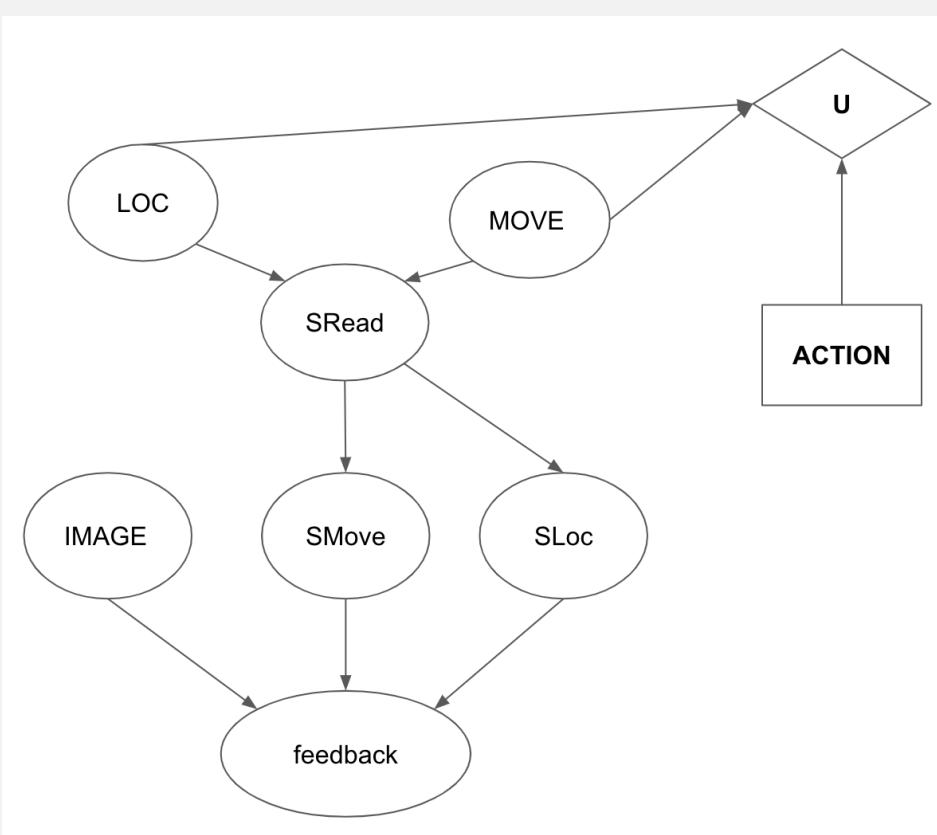
**Exercise 16.5.#188-VEHI**

Origin:

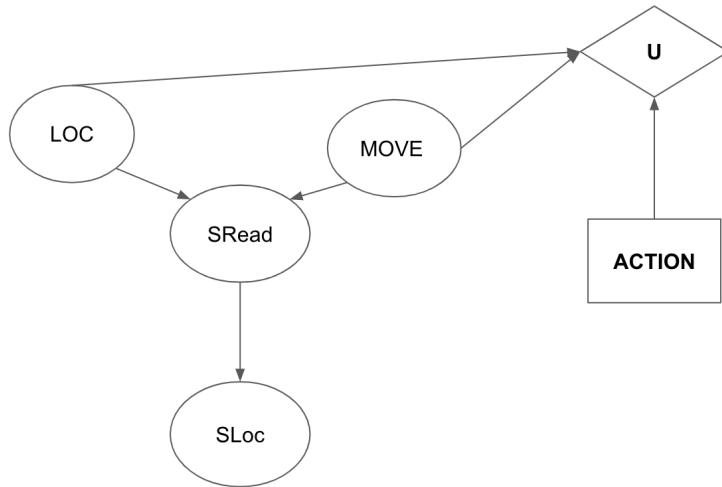
`fa19_final_lindsay_vpi`

A vehicle is trying to identify the situation of the world around it using a set of sensors located around the vehicle. Each sensor reading is based off of an object's location (LOC) and an object's movement (MOVE). The sensor reading will then produce various values for its predicted location, predicted movement, and image rendered, which is then sent back to the user. The vehicle takes an action, and we assign some utility to the action based on the object's location and movement. Possible actions are MOVE TOWARDS, MOVE AWAY, and STOP.

- For part (a) only, suppose the decision network faced by the vehicle is the following.



- (i) Based on the diagram above, is it **possibly** true that  $VPI(SMove, SRead) > VPI(SRead)$ ?
- (ii) Based on the diagram above, is it **necessarily** true that  $VPI(SRead) = 0$ ?
- b. For the remainder of this problem, let's assume that your startup has less money, so we use a simpler sensor network. One possible sensor network can be represented as follows.



You have distributions of  $P(LOC)$ ,  $P(MOVE)$ ,  $P(SRead|LOC, MOVE)$ ,  $P(SLoc|SRead)$  and utility values  $U(ACTION, LOC, MOVE)$ . What is equation for determining the expected utility for some ACTION  $a$ ?

- c. Your colleague Bob invented a new sensor to observe values of  $SLoc$ .
  - (i) Suppose that your company had no sensors till this point. What is  $VPI(SLoc)$  in terms of the given distributions?
  - (ii) Gaagle, an established company, wants to sell your startup a device that gives you  $SRead$ . Given that you already have Bob's device (that gives you  $SLoc$ ), what is the maximum amount of money you should pay for Gaagle's device? Suppose you value \$1 at 1 utility. Please simplify your answer as much as possible.

- a. (i) No

$VPI(SMove, SRead) = VPI(SMove|SRead) + VPI(SRead)$ , therefore we can cancel  $VPI(SRead)$  from both side, which becomes asking if  $VPI(SMove|SRead) > 0$ .

And we can see that cannot be true, because after shading in  $SRead$ , there is no active path connecting  $SMove$  and  $U$ .

- (ii) No

$VPI(SRead)$  could be  $> 0$  because  $SRead - MOVE - U$  is an active path between  $SRead$  and  $U$

b.  $EU(action) = \sum_{loc} P(loc) \sum_{move} P(move) U(action, loc, move)$

We can eliminate  $SRead$  and  $SLoc$  via marginalization, so they don't need to be included the expression

- c. (i) Substituting in the definitions

$$\begin{aligned} VPI(SLoc) &= MEU(SLoc) - MEU(\{ \}) \\ &= \left( \sum_{sloc} P(sloc) MEU(SLoc = sloc) \right) - \max_a EU(a) \end{aligned}$$

- (ii) We should pay

$$\begin{aligned} VPI(SRead|SLoc) &= VPI(SRead, SLoc) - VPI(SLoc) \\ &= VPI(SRead) + VPI(SLoc|SRead) - VPI(SLoc) \\ &= VPI(SRead) + 0 - VPI(SLoc) \\ &= VPI(SRead) - VPI(SLoc) \end{aligned}$$

### Exercise 16.5.#188-MONT

Origin:

su16\_midterm2\_VPI

You are the latest contestant on Monty Hall's game show, which has undergone a few changes over the years.

In the game, there are  $n$  closed doors: behind one door is a car ( $U(car) = 1000$ ), while the other  $n - 1$  doors each have a goat behind them ( $U(goat) = 10$ ). You are permitted to open exactly one door and claim the prize behind it.

You begin by choosing a door uniformly at random.

- a. What is your expected utility?
- b. After you choose a door but before you open it, Monty offers to open  $k$  other doors, each of which are guaranteed to have a goat behind it.  
If you accept this offer, should you keep your original choice of a door, or switch to a new door?
- c. What is the value of the information that Monty is offering you?
- d. Monty is changing his offer!

After you choose your initial door, you are given the offer to choose any other door and open this second door. If you do, after you see what is inside the other door, you may switch your initial choice (to the newly opened door) or keep your initial choice.

What is the value of this new offer?

- e. Monty is generalizing his offer: you can pay  $\$d^3$  to open  $d$  doors as in the previous part. (Assume that  $U(\$x) = x$ .) You may now switch your choice to any of the open doors (or keep your initial choice). What is the largest value of  $d$  for which it would be rational to accept the offer?

- a.  $(1000 * \frac{1}{n} + 10 * \frac{n-1}{n})$  or  $(10 + 990 * \frac{1}{n})$

We can calculate the expected utility via the usual formula of expectation, or we can note that there is a guaranteed utility of 10, with a small probability of a bonus utility.

## Exercises 16 Making Simple Decisions

The latter is a bit simpler, so the answers to the following parts use this form.

b.  $EU(keep) = 10 + 990 * \frac{1}{n}$

$$EU(switch) = 10 + 990 * \frac{(n-1)}{n*(n-k-1)}$$

Action that achieves  $MEU$ : switch

The expected utility if we keep must be the same as the answer from the previous part: the probability that we have a winning door has not changed at all, since we have gotten no meaningful information.

In order to win a car by switching, we must have chosen a goat door previously (probability  $\frac{n-1}{n}$ ) and then switch to the car door (probability  $\frac{1}{n-k-1}$ ).

Since  $n - 1 > n - k - 1$  for positive  $k$ , switching gets a larger expected utility.

c.  $990 * \frac{1}{n} * \frac{k}{n-k-1}$

The formula for VPI is  $MEU(e) - MEU(\{\})$ . Thus, we want the difference between  $EU(switch)$  (the optimal action if Monty opens the doors) and our expected utility from part (a).

(It is true that  $EU(keep)$  happens to have the same numeric expression as in part (a), but this fact is not meaningful in answering this part.)

d.  $\frac{990}{n}$

Intuitively, if we take this offer, it is as if we just chose two doors in the beginning, and we win if either door has the car behind it. Unlike in the previous parts, if the new door has a goat behind it, it is not more optimal to switch doors.

Mathematically, letting  $D_i$  be the event that door  $i$  has the car, we can calculate this as  $P(D_2 \cup D_1) = P(D_1) + P(D_2) = 1/n + 1/n = 2/n$ , to see that  $MEU(\text{offer}) = 10 + 990 * \frac{2}{n}$ . Subtracting the expected utility without taking the offer, we are left with  $990 * \frac{1}{n}$ .

e.  $d = \sqrt{\frac{990}{n}}$

It is a key insight (whether intuitive or determined mathematically) that the answer to the previous part is constant for each successive door we open. Thus, the value of opening  $d$  doors is just  $d * 990 * \frac{1}{n}$ . Setting this equal to  $d^3$ , we can solve for  $d$ .

## 16.6 Unknown Preferences

### Exercise 16.6.#OFFS

In the off-switch problem (Section 16.7.2), we have assumed that Harriet acts rationally. Suppose instead that she is **Boltzmann-rational**, i.e., she follows a randomized policy that chooses action  $x$  with a softmax probability:

$$\pi(x) = \frac{e^{\beta U_x}}{\sum_y e^{\beta U_y}}.$$

- a. Derive the general condition for Robbie to defer to Harriet, assuming that Robbie's prior for Harriet's utility for the immediate action  $a$  is  $P(u)$ .

- b. Determine the minimum value of  $\beta$  such that Robbie defers to Harriet in the example of Figure 16.11.

[[TBC]]

# EXERCISES 17

## MAKING COMPLEX DECISIONS

### 17.1 Sequential Decision Problems

#### Exercise 17.1.#MDPX

For the  $4 \times 3$  world shown in Figure 17.1, calculate which squares can be reached from (1,1) by the action sequence [Right, Right, Right, Up, Up] and with what probabilities. Explain how this computation is related to the prediction task (see Section 14.2.1) for a hidden Markov model.

The best way to calculate this is NOT by thinking of all ways to get to any given square and how likely all those ways are, but to compute the occupancy probabilities at each time step. These are as follows:

|       |   | Right | Right | Right | Up    | Up     |
|-------|---|-------|-------|-------|-------|--------|
| (1,1) | 1 | .1    | .02   | .011  | .0075 | .00854 |
| (1,2) |   | .1    | .09   | .075  | .0238 | .01076 |
| (1,3) |   |       | .01   | .010  | .0698 | .08260 |
| (2,1) |   | .8    | .24   | .064  | .0779 | .06883 |
| (2,3) |   |       |       | .008  | .0074 | .01810 |
| (3,1) |   |       | .64   | .256  | .0576 | .01547 |
| (3,2) |   |       |       | .064  | .2112 | .06720 |
| (3,3) |   |       |       |       | .0520 | .21130 |
| (4,1) |   |       |       | .512  | .0768 | .01344 |
| (4,2) |   |       |       |       | .4160 | .49856 |
| (4,3) |   |       |       |       |       | .00520 |

Projection in an HMM involves multiplying the vector of occupancy probabilities by the transition matrix. Here, the only difference is that there is a different transition matrix for each action.

#### Exercise 17.1.#SEQP

Select a specific member of the set of policies that are optimal for  $R(s) > 0$  as shown in Figure 17.2(b), and calculate the fraction of time the agent spends in each state, in the limit, if the policy is executed forever. (*Hint:* Construct the state-to-state transition probability matrix corresponding to the policy and see Exercise MARKOV-CONVERGENCE-EXERCISE.)

If we pick the policy that goes *Right* in all the optional states, and construct the corresponding transition matrix  $\mathbf{T}$ , we find that the equilibrium distribution—the solution to  $\mathbf{T}\mathbf{x} = \mathbf{x}$ —has occupancy probabilities of  $1/12$  for  $(2,3)$ ,  $(3,1)$ ,  $(3,2)$ ,  $(3,3)$  and  $2/3$  for  $(4,1)$ . These can be found most simply by computing  $\mathbf{T}^n\mathbf{x}$  for any initial occupancy vector  $\mathbf{x}$ , for  $n$  large enough to achieve convergence.

### Exercise 17.1.#NONS

Suppose that we define the utility of a state sequence to be the *maximum* reward obtained in any state in the sequence. Show that this utility function does not result in stationary preferences between state sequences. Is it still possible to define a utility function on states such that MEU decision making gives optimal behavior?

Stationarity requires the agent to have identical preferences between the sequence pair  $[s_0, s_1, s_2, \dots]$ ,  $[s_0, s'_1, s'_2, \dots]$  and between the sequence pair  $[s_1, s_2, \dots]$ ,  $[s'_1, s'_2, \dots]$ . If the utility of a sequence is its maximum reward, we can easily violate stationarity. For example,

$$[4, 3, 0, 0, 0, \dots] \sim [4, 0, 0, 0, 0, \dots]$$

but

$$[3, 0, 0, 0, \dots] \succ [0, 0, 0, 0, \dots].$$

We can still define  $U^\pi(s)$  as the expected maximum reward obtained by executing  $\pi$  starting in  $s$ . The agent's preferences seem peculiar, nonetheless. For example, if the current state  $s$  has reward  $R_{\max}$ , the agent will be indifferent among all actions, but once the action is executed and the agent is no longer in  $s$ , it will suddenly start to care about what happens next.

### Exercise 17.1.#FMDP

Can any finite search problem be translated exactly into a Markov decision problem such that an optimal solution of the latter is also an optimal solution of the former? If so, explain *precisely* how to translate the problem and how to translate the solution back; if not, explain *precisely* why not (i.e., give a counterexample).

A finite search problem (see Chapter 3) is defined by an initial state  $s_0$ , a successor function  $S(s)$  that returns a set of action-state pairs, a step cost function  $c(s, a, s')$ , and a goal test. An optimal solution is a least-cost path from  $s_0$  to any goal state.

To construct the corresponding MDP, define  $R(s, a, s') = -c(s, a, s')$  unless  $s$  is a goal state, in which case  $R(s, a, s') = 0$  (see Ex. 17.5 for how to obtain the effect of a three-argument reward function); define  $T(s, a, s') = 1$  if  $(a, s') \in S(s)$ ; and  $\gamma = 1$ . An optimal solution to this MDP is a policy that follows the least-cost path from each state to its nearest goal state.

**Exercise 17.1.#REWQ**

Sometimes MDPs are formulated with a reward function  $R(s, a)$  that depends on the action taken or with a reward function  $R(s, a, s')$  that also depends on the outcome state.

- Write the Bellman equations for these formulations.
- Show how an MDP with reward function  $R(s, a, s')$  can be transformed into a different MDP with reward function  $R(s, a)$ , such that optimal policies in the new MDP correspond exactly to optimal policies in the original MDP.
- Now do the same to convert MDPs with  $R(s, a)$  into MDPs with  $R(s)$ .

This is a deceptively simple exercise that tests the student's understanding of the formal definition of MDPs. Some students may need a hint or an example to get started.

- The key here is to get the max and summation in the right place. For  $R(s, a)$  we have

$$U(s) = \max_a [R(s, a) + \gamma \sum_{s'} T(s, a, s') U(s')]$$

and for  $R(s, a, s')$  we have

$$U(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma U(s')] .$$

- There are a variety of solutions here. One is to create a “pre-state”  $pre(s, a, s')$  for every  $s, a, s'$ , such that executing  $a$  in  $s$  leads not to  $s'$  but to  $pre(s, a, s')$ . In this state is encoded the fact that the agent came from  $s$  and did  $a$  to get here. From the pre-state, there is just one action  $b$  that always leads to  $s'$ . Let the new MDP have transition  $T'$ , reward  $R'$ , and discount  $\gamma'$ . Then

$$\begin{aligned} T'(s, a, pre(s, a, s')) &= T(s, a, s') \\ T'(pre(s, a, s'), b, s') &= 1 \\ R'(s, a) &= 0 \\ R'(pre(s, a, s'), b) &= \gamma^{-\frac{1}{2}} R(s, a, s') \\ \gamma' &= \gamma^{\frac{1}{2}} \end{aligned}$$

- In keeping with the idea of part (b), we can create states  $post(s, a)$  for every  $s, a$ , such that

$$\begin{aligned} T'(s, a, post(s, a, s')) &= 1 \\ T'(post(s, a, s'), b, s') &= T(s, a, s') \\ R'(s) &= 0 \\ R'(post(s, a, s')) &= \gamma^{-\frac{1}{2}} R(s, a) \\ \gamma' &= \gamma^{\frac{1}{2}} \end{aligned}$$

### Exercise 17.1.#THRC

For the environment shown in Figure 17.1, find all the threshold values for  $R(s)$  such that the optimal policy changes when the threshold is crossed. You will need a way to calculate the optimal policy and its value for fixed  $R(s)$ . (*Hint:* Prove that the value of any fixed policy varies linearly with  $R(s)$ .)

This can be done fairly simply by:

- Call `policy-iteration` (from "uncertainty/algorithms/dp.lisp") on the Markov Decision Processes representing the 4x3 grid, with values for the step cost ranging from, say, 0.0 to 1.0 in increments of 0.02.
- For any two adjacent policies that differ, run binary search on the step cost to pinpoint the threshold value.
- Convince yourself that you haven't missed any policies, either by using too coarse an increment in step size (0.02), or by stopping too soon (1.0).

One useful observation in this context is that the expected total reward of any fixed policy is linear in  $r$ , the per-step reward for the empty states. Imagine drawing the total reward of a policy as a function of  $r$ —a straight line. Now draw all the straight lines corresponding to all possible policies. The reward of the *optimal* policy as a function of  $r$  is just the max of all these straight lines. Therefore it is a piecewise linear, convex function of  $r$ . Hence there is a very efficient way to find *all* the optimal policy regions:

- For any two consecutive values of  $r$  that have different optimal policies, find the optimal policy for the midpoint. Once two consecutive values of  $r$  give the same policy, then the interval between the two points must be covered by that policy.
- Repeat this until two points are known for each distinct optimal policy.
- Suppose  $(r_{a1}, v_{a1})$  and  $(r_{a2}, v_{a2})$  are points for policy  $a$ , and  $(r_{b1}, v_{b1})$  and  $(r_{b2}, v_{b2})$  are the next two points, for policy  $b$ . Clearly, we can draw straight lines through these pairs of points and find their intersection. This does not mean, however, that there is no other optimal policy for the intervening region. We can determine this by calculating the optimal policy for the intersection point. If we get a different policy, we continue the process.

The policies and boundaries derived from this procedure are shown in Figure S17.1. The figure shows that there are *nine* distinct optimal policies! Notice that as  $r$  becomes more negative, the agent becomes more willing to dive straight into the  $-1$  terminal state rather than face the cost of the detour to the  $+1$  state.

The somewhat ugly code is as follows. Notice that because the lines for neighboring policies are very nearly parallel, numerical instability is a serious problem.

```
(defun policy-surface (mdp r1 r2 &aux prev (unchanged nil))
 "returns points on the piecewise-linear surface
 defined by the value of the optimal policy of mdp as a
 function of r"
 (setq rvplist
 (list (cons r1 (r-policy mdp r1)) (cons r2 (r-policy mdp r2)))))
```

## Exercises 17 Making Complex Decisions

```

(do ()
 (unchanged rvplist)
 (setq unchanged t)
 (setq prev nil)
 (dolist (rvp rvplist)
 (let* ((rest (cdr (member rvp rvplist :test #'eq)))
 (next (first rest))
 (next-but-one (second rest)))
 (dprint (list (first prev) (first rvp)
 ' * (first next) (first next-but-one)))
 (when next
 (unless (or (= (first rvp) (first next))
 (policy-equal (third rvp) (third next) mdp))
 (dprint "Adding new point(s)")
 (setq unchanged nil)
 (if (and prev next-but-one
 (policy-equal (third prev) (third rvp) mdp)
 (policy-equal (third next) (third next-but-one) mdp))
 (let* ((intxy (policy-vertex prev rvp next next-but-one))
 (int (cons (xy-x intxy) (r-policy mdp (xy-x intxy)))))
 (dprint (list "Found intersection" intxy))
 (cond ((or (< (first int) (first rvp))
 (> (first int) (first next)))
 (dprint "Intersection out of range!")
 (let ((int-r (/ (+ (first rvp) (first next)) 2)))
 (setq int (cons int-r (r-policy mdp int-r))))
 (push int (cdr (member rvp rvplist :test #'eq)))))
 (or (policy-equal (third rvp) (third int) mdp)
 (policy-equal (third next) (third int) mdp)
 (dprint "Found policy boundary")
 (push (list (first int) (second int) (third next))
 (cdr (member rvp rvplist :test #'eq))))
 (push (list (first int) (second int) (third rvp))
 (cdr (member rvp rvplist :test #'eq))))
 (t (dprint "Found new policy!")
 (push int (cdr (member rvp rvplist :test #'eq)))))))
 (let* ((int-r (/ (+ (first rvp) (first next)) 2))
 (int (cons int-r (r-policy mdp int-r))))
 (dprint (list "Adding split point" (list int-r (second int)))))
 (push int (cdr (member rvp rvplist :test #'eq)))))))
 (setq prev rvp)))
 (defun r-policy (mdp r &aux U)
 (set-rewards mdp r)
 (setq U (value-iteration mdp
 (copy-hash-table (mdp-rewards mdp) #'identity)
 :epsilon 0.0000000001))
 (list (gethash '(1 1) U) (optimal-policy U (mdp-model mdp) (mdp-rewards mdp)))))

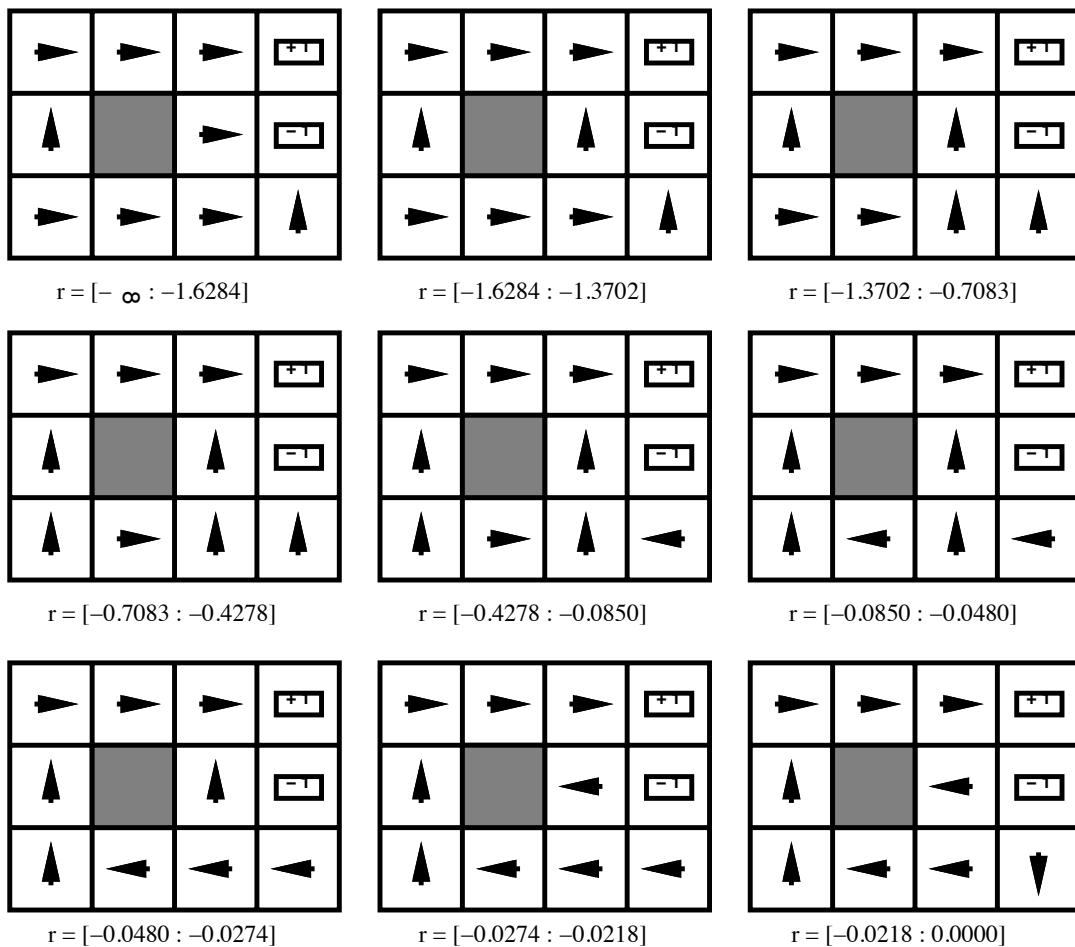
 (defun set-rewards (mdp r &aux (rewards (mdp-rewards mdp))
 (terminals (mdp-terminal-states mdp)))
 (maphash #'(lambda (state reward)
 (unless (member state terminals :test #'equal)
 (setf (gethash state rewards) r)))
 rewards)))

```

```
(defun policy-equal (p1 p2 mdp &aux (match t)
 (terminals (mdp-terminal-states mdp)))
 (maphash #'(lambda (state action)
 (unless (member state terminals :test #'equal)
 (unless (eq (caar (gethash state p1)) (caar (gethash state p2)))
 (setq match nil))))
 p1)
 match)

(defun policy-vertex (rvp1 rvp2 rvp3 rvp4)
 (let ((a (make-xy :x (first rvp1) :y (second rvp1)))
 (b (make-xy :x (first rvp2) :y (second rvp2)))
 (c (make-xy :x (first rvp3) :y (second rvp3)))
 (d (make-xy :x (first rvp4) :y (second rvp4))))
 (intersection-point (make-line :xy1 a :xy2 b)
 (make-line :xy1 c :xy2 d)))

(defun intersection-point (l1 l2)
 ;;; l1 is line ab; l2 is line cd
 ;;; assume the lines cross at alpha a + (1-alpha) b,
 ;;; also known as beta c + (1-beta) d
 ;;; returns the intersection point unless they're parallel
 (let* ((a (line-xy1 l1))
 (b (line-xy2 l1))
 (c (line-xy1 l2))
 (d (line-xy2 l2))
 (xa (xy-x a)) (ya (xy-y a))
 (xb (xy-x b)) (yb (xy-y b))
 (xc (xy-x c)) (yc (xy-y c))
 (xd (xy-x d)) (yd (xy-y d))
 (q (- (* (- xa xb) (- yc yd))
 (* (- ya yb) (- xc xd)))))
 (unless (zerop q)
 (let ((alpha (/ (- (* (- xd xb) (- yc yd))
 (* (- yd yb) (- xc xd)))
 q)))
 (make-xy :x (float (+ (* alpha xa) (* (- 1 alpha) xb)))
 :y (float (+ (* alpha ya) (* (- 1 alpha) yb))))))))
```



**Figure S17.1** Optimal policies for different values of the cost of a step in the  $4 \times 3$  environment, and the boundaries of the regions with constant optimal policy.

## 17.2 Algorithms for MDPs

### Exercise 17.2.#VICT

Equation (17.11) on page 575 states that the Bellman operator is a contraction.

- a. Show that, for any functions  $f$  and  $g$ ,

$$|\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)| .$$

- b. Write out an expression for  $|(B U_i - B U'_i)(s)|$  and then apply the result from (a) to complete the proof that the Bellman operator is a contraction.

- a. To find the proof, it may help first to draw a picture of two arbitrary functions  $f$  and  $g$  and mark the maxima; then it is easy to find a point where the difference between the functions is bigger than the difference between the maxima. Assume, w.l.o.g., that  $\max_a f(a) \geq \max_a g(a)$ , and let  $f$  have its maximum value at  $a^*$ . Then we have

$$\begin{aligned} |\max_a f(a) - \max_a g(a)| &= \max_a f(a) - \max_a g(a) \quad (\text{by assumption}) \\ &= f(a^*) - \max_a g(a) \\ &\leq f(a^*) - g(a^*) \\ &\leq \max_a |f(a) - g(a)| \quad (\text{by definition of max}) \end{aligned}$$

- b. From the definition of the  $B$  operator (Equation (17.10)) we have

$$\begin{aligned} |(B U_i - B U'_i)(s)| &= |R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U_i(s') \\ &\quad - R(s) - \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U'_i(s')| \\ &= \gamma \left| \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U_i(s') - \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U'_i(s') \right| \\ &\leq \gamma \max_{a \in A(s)} \left| \sum_{s'} P(s' | s, a) U_i(s') - \sum_{s'} P(s' | s, a) U'_i(s') \right| \\ &= \gamma \left| \sum_{s'} P(s' | s, a^*(s)) U_i(s') - \sum_{s'} P(s' | s, a^*(s)) U'_i(s') \right| \\ &= \gamma \left| \sum_{s'} P(s' | s, a^*(s)) (U_i(s') - U'_i(s')) \right| \end{aligned}$$

Inserting this into the expression for the max norm, we have

$$\begin{aligned} \|B U_i - B U'_i\| &= \max_s |(B U_i - B U'_i)(s)| \\ &\leq \gamma \max_s \left| \sum_{s'} P(s' | s, a^*(s)) (U_i(s') - U'_i(s')) \right| \\ &\leq \gamma \max_s |U_i(s) - U'_i(s)| = \gamma \|U_i - U'_i\| \end{aligned}$$

### Exercise 17.2.#MDPZ

This exercise considers two-player MDPs that correspond to zero-sum, turn-taking games like those in Chapter 5. Let the players be  $A$  and  $B$ , and let  $R(s)$  be the reward for player  $A$  in state  $s$ . (The reward for  $B$  is always equal and opposite.)

- Let  $U_A(s)$  be the utility of state  $s$  when it is  $A$ 's turn to move in  $s$ , and let  $U_B(s)$  be the utility of state  $s$  when it is  $B$ 's turn to move in  $s$ . All rewards and utilities are calculated from  $A$ 's point of view (just as in a minimax game tree). Write down Bellman equations defining  $U_A(s)$  and  $U_B(s)$ .
- Explain how to do two-player value iteration with these equations, and define a suitable termination criterion.
- Consider the game described in Figure ?? on page ?? . Draw the state space (rather than

the game tree), showing the moves by  $A$  as solid lines and moves by  $B$  as dashed lines. Mark each state with  $R(s)$ . You will find it helpful to arrange the states  $(s_A, s_B)$  on a two-dimensional grid, using  $s_A$  and  $s_B$  as “coordinates.”

- d. Now apply two-player value iteration to solve this game, and derive the optimal policy.

- a. For  $U_A$  we have

$$U_A(s) = R(s) + \max_a \sum_{s'} P(s'|a, s) U_B(s')$$

and for  $U_B$  we have

$$U_B(s) = R(s) + \min_a \sum_{s'} P(s'|a, s) U_A(s') .$$

- b. To do value iteration, we simply turn each equation from part (a) into a Bellman update and apply them in alternation, applying each to all states simultaneously. The process terminates when the utility vector for one player is the same as the previous utility vector *for the same player* (i.e., two steps earlier). (Note that typically  $U_A$  and  $U_B$  are not the same in equilibrium.)
- c. The state space is shown in Figure S17.2.
- d. We mark the terminal state values in bold and initialize other values to 0. Value iteration proceeds as follows:

|       | (1,4) | (2,4) | (3,4) | (1,3) | (2,3) | (4,3)     | (1,2) | (3,2) | (4,2)     | (2,1)     | (3,1)     |
|-------|-------|-------|-------|-------|-------|-----------|-------|-------|-----------|-----------|-----------|
| $U_A$ | 0     | 0     | 0     | 0     | 0     | <b>+1</b> | 0     | 0     | <b>+1</b> | <b>-1</b> | <b>-1</b> |
| $U_B$ | 0     | 0     | 0     | 0     | -1    | <b>+1</b> | 0     | -1    | <b>+1</b> | <b>-1</b> | <b>-1</b> |
| $U_A$ | 0     | 0     | 0     | -1    | +1    | <b>+1</b> | -1    | +1    | <b>+1</b> | <b>-1</b> | <b>-1</b> |
| $U_B$ | -1    | +1    | +1    | -1    | -1    | <b>+1</b> | -1    | -1    | <b>+1</b> | <b>-1</b> | <b>-1</b> |
| $U_A$ | +1    | +1    | +1    | -1    | +1    | <b>+1</b> | -1    | +1    | <b>+1</b> | <b>-1</b> | <b>-1</b> |
| $U_B$ | -1    | +1    | +1    | -1    | -1    | <b>+1</b> | -1    | -1    | <b>+1</b> | <b>-1</b> | <b>-1</b> |

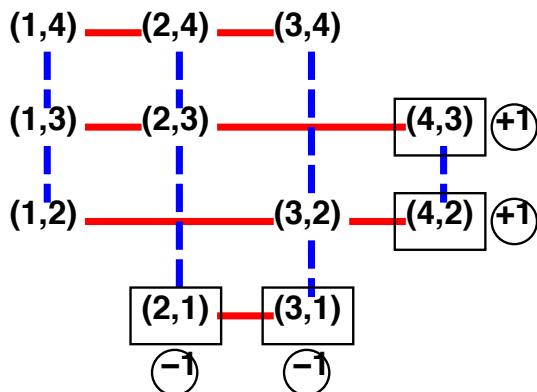
and the optimal policy for each player is as follows:

|           | (1,4) | (2,4) | (3,4) | (1,3) | (2,3) | (4,3) | (1,2) | (3,2) | (4,2) | (2,1) | (3,1) |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| $\pi_A^*$ | (2,4) | (3,4) | (2,4) | (2,3) | (4,3) |       | (3,2) | (4,2) |       |       |       |
| $\pi_B^*$ | (1,3) | (2,3) | (3,2) | (1,2) | (2,1) |       | (1,3) | (3,1) |       |       |       |

### Exercise 17.2.#TMDP

Consider the  $3 \times 3$  world shown in Figure ??(a). The transition model is the same as in the  $4 \times 3$  Figure 17.1: 80% of the time the agent goes in the direction it selects; the rest of the time it moves at right angles to the intended direction.

Implement value iteration for this world for each value of  $r$  below. Use discounted rewards with a discount factor of 0.99. Show the policy obtained in each case. Explain intu-



**Figure S17.2** State-space graph for the game in Figure ??.

itively why the value of  $r$  leads to each policy.

- a.  $r = 100$
- b.  $r = -3$
- c.  $r = 0$
- d.  $r = +3$

a.  $r = 100.$

|   |   |   |
|---|---|---|
| u | 1 | . |
| u | 1 | d |
| u | 1 | l |

See the comments for part d. This should have been  $r = -100$  to illustrate an alternative behavior:

|   |   |   |
|---|---|---|
| r | r | . |
| d | r | u |
| r | r | u |

Here, the agent tries to reach the goal quickly, subject to attempting to avoid the square  $(1, 3)$  as much as possible. Note that the agent will choose to move Down in square  $(1, 2)$  in order to actively avoid the possibility of “accidentally” moving into the square  $(1, 3)$  if it tried to move Right instead, since the penalty for moving into square  $(1, 3)$  is so great.

b.  $r = -3.$

|   |   |   |
|---|---|---|
| r | r | . |
| r | r | u |
| r | r | u |

Here, the agent again tries to reach the goal as fast as possible while attempting to avoid the square  $(1, 3)$ , but the penalty for square  $(1, 3)$  is not so great that the agent will try to actively avoid it at all costs. Thus, the agent will choose to move Right in

## Exercises 17 Making Complex Decisions

square  $(1, 2)$  in order to try to get closer to the goal even if it occasionally will result in a transition to square  $(1, 3)$ .

c.  $r = 0.$

|   |   |   |
|---|---|---|
| r | r | . |
| u | u | u |
| u | u | u |

Here, the agent again tries to reach the goal as fast as possible, but will try to do so via a path that includes square  $(1, 3)$  if possible. This results from the fact that square  $(1, 3)$  does not incur the reward of  $-1$  in all other non-goal states, so it reaching the goal via a path through that square can potentially have slightly greater reward than another path of equal length that does not pass through  $(1, 3)$ .

d.  $r = 3.$

|   |   |   |
|---|---|---|
| u | 1 | . |
| u | 1 | d |
| u | 1 | 1 |

### Exercise 17.2.#MDPO

Consider the  $101 \times 3$  world shown in Figure ??(b). In the start state the agent has a choice of two deterministic actions, *Up* or *Down*, but in the other states the agent has one deterministic action, *Right*. Assuming a discounted reward function, for what values of the discount  $\gamma$  should the agent choose *Up* and for which *Down*? Compute the utility of each action as a function of  $\gamma$ . (Note that this simple example actually reflects many real-world situations in which one must weigh the value of an immediate action versus the potential continual long-term consequences, such as choosing to dump pollutants into a lake.)

The utility of Up is

$$50\gamma - \sum_{t=2}^{101} \gamma^t = 50\gamma - \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma},$$

while the utility of Down is

$$-50\gamma + \sum_{t=2}^{101} \gamma^t = -50\gamma + \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma}.$$

Solving numerically we find the indifference point to be  $\gamma \approx 0.9844$ : larger than this and we want to go Down to avoid the expensive long-term consequences, smaller than this and we want to go Up to get the immediate benefit.

### Exercise 17.2.#MDPU

Consider an undiscounted MDP having three states,  $(1, 2, 3)$ , with rewards  $-1, -2, 0$ , respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: *a* and *b*. The transition model is as follows:

- In state 1, action  $a$  moves the agent to state 2 with probability 0.8 and makes the agent stay put with probability 0.2.
- In state 2, action  $a$  moves the agent to state 1 with probability 0.8 and makes the agent stay put with probability 0.2.
- In either state 1 or state 2, action  $b$  moves the agent to state 3 with probability 0.1 and makes the agent stay put with probability 0.9.

Answer the following questions:

- What can be determined *qualitatively* about the optimal policy in states 1 and 2?
- Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action  $b$  in both states.
- What happens to policy iteration if the initial policy has action  $a$  in both states? Does discounting help? Does the optimal policy depend on the discount factor?

- Intuitively, the agent wants to get to state 3 as soon as possible, because it will pay a cost for each time step it spends in states 1 and 2. However, the only action that reaches state 3 (action  $b$ ) succeeds with low probability, so the agent should minimize the cost it incurs while trying to reach the terminal state. This suggests that the agent should definitely try action  $b$  in state 1; in state 2, it might be better to try action  $a$  to get to state 1 (which is the better place to wait for admission to state 3), rather than aiming directly for state 3. The decision in state 2 involves a numerical tradeoff.
- The application of policy iteration proceeds in alternating steps of value determination and policy update.

- *Initialization:*  $U \leftarrow \langle -1, -2, 0 \rangle$ ,  $P \leftarrow \langle b, b \rangle$ .

- *Value determination:*

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.1u_3 + 0.9u_2 \\ u_3 &= 0 \end{aligned}$$

That is,  $u_1 = -10$  and  $u_2 = -20$ .

*Policy update:* In state 1,

$$\sum_j T(1, a, j)u_j = 0.8 \times -20 + 0.2 \times -10 = -18$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

so action  $b$  is still preferred for state 1.

## Exercises 17 Making Complex Decisions

In state 2,

$$\sum_j T(1, a, j)u_j = 0.8 \times -10 + 0.2 \times -20 = -12$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 \times 0.9 \times -20 = -18$$

so action  $a$  is preferred for state 1. We set  $\text{unchanged?} \leftarrow \text{false}$  and proceed.

- *Value determination:*

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

Once more  $u_1 = -10$ ; now,  $u_2 = -15$ . *Policy update:* In state 1,

$$\sum_j T(1, a, j)u_j = 0.8 \times -15 + 0.2 \times -10 = -14$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

so action  $b$  is still preferred for state 1.

In state 2,

$$\sum_j T(1, a, j)u_j = 0.8 \times -10 + 0.2 \times -15 = -11$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 \times 0.9 \times -15 = -13.5$$

so action  $a$  is still preferred for state 1.  $\text{unchanged?}$  remains true, and we terminate.

Note that the resulting policy matches our intuition: when in state 2, try to move to state 1, and when in state 1, try to move to state 3.

- c. An initial policy with action  $a$  in both states leads to an unsolvable problem. The initial value determination problem has the form

$$\begin{aligned} u_1 &= -1 + 0.2u_1 + 0.8u_2 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

and the first two equations are inconsistent. If we were to try to solve them iteratively,

we would find the values tending to  $-\infty$ .

Discounting leads to well-defined solutions by bounding the penalty (expected discounted cost) an agent can incur at either state. However, the choice of discount factor will affect the policy that results. For  $\gamma$  small, the cost incurred in the distant future plays a negligible role in the value computation, because  $\gamma^n$  is near 0. As a result, an agent could choose action  $b$  in state 2 because the discounted short-term cost of remaining in the non-terminal states (states 1 and 2) outweighs the discounted long-term cost of action  $b$  failing repeatedly and leaving the agent in state 2. An additional exercise could ask the student to determine the value of  $\gamma$  at which the agent is indifferent between the two choices.

### Exercise 17.2.#MDPF

Consider the  $4 \times 3$  world shown in Figure 17.1.

- Implement an environment simulator for this environment, such that the specific geography of the environment is easily altered. Some code for doing this is already in the online code repository.
- Create an agent that uses policy iteration, and measure its performance in the environment simulator from various starting states. Perform several experiments from each starting state, and compare the average total reward received per run with the utility of the state, as determined by your algorithm.
- Experiment with increasing the size of the environment. How does the run time for policy iteration vary with the size of the environment?

The framework for this problem is in `uncertainty/domains/4x3-mdp.lisp`. There is still some synthesis for the student to do for answer b. For c. some experimental design is necessary.

### Exercise 17.2.#POLL

How can the policy evaluation algorithm be used to calculate the expected loss experienced by an agent using a given set of utility estimates  $U$  and an estimated model  $P$ , compared with an agent using correct values?

The policy evaluation algorithm calculates  $U^\pi(s)$  for a given policy  $\pi$ . The policy for an agent that thinks  $U$  is the true utility and  $P$  is the true model would be based on Equation (17.4):

$$\pi(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s') .$$

Given this policy, the policy loss compared to the true optimal policy, starting in state  $s$ , is just  $U^{\pi^*}(s) - U^\pi(s)$ .

**Exercise 17.2.#UCTT**

In this exercise we explore the application of UCT to Tetris.

- a. Create an implementation the Tetris MDP as described in Figure 17.5. Each action simply places the current piece in any reachable location and orientation.
- b. Estimate the reward for a purely random policy by running rollouts.
- c. Implement a version of UCT (Section 5.4) suitable for MDPs.
- d. Apply your algorithm to Tetris and measure its performance as a function of the number of rollouts per move, assuming a purely random policy for rollouts and a value  $C = \sqrt{2}$  for the parameter that controls the exploration/exploitation tradeoff.
- e. Come up with a better rollout policy and measure its performance as a function of the number of rollouts and CPU time.

The implementation of this exercise left up to interested readers.

**Exercise 17.2.#VAIT**

Value iteration:

- (i) Is a model-free method for finding optimal policies.
- (ii) Is sensitive to local optima.
- (iii) Is tedious to do by hand.
- (iv) Is guaranteed to converge when the discount factor satisfies  $0 < \gamma < 1$ .

(iii) and (iv). Value iteration requires a model (an specified MDP), and is not sensitive to getting stuck in local optima.

**Exercise 17.2.#MDPT**

Please indicate whether the following statements are true or false

- a. If the only difference between two MDPs is the value of the discount factor then they must have the same optimal policy.
- b. When using features to represent the Q-function (rather than having a tabular representation) it is possible that Q-learning does not find the optimal Q-function  $Q^*$ .
- c. For an infinite horizon MDP with a finite number of states and actions and with a discount factor  $\gamma$ , with  $0 < \gamma < 1$ , value iteration is guaranteed to converge.
- d. When getting to act only for a finite number of steps in an MDP, the optimal policy is stationary. (A stationary policy is a policy that takes the same action in a given state, independent of at what time the agent is in that state.)

- a. False. A counterexample suffices to show the statement is false. Consider an MDP with two sink states. Transitioning into sink state  $A$  gives a reward of 1, transitioning into

sink state  $B$  gives a reward of 10. All other transitions have zero rewards. Let  $A$  be one step North from the start state. Let  $B$  be two steps South from the start state. Assume actions always succeed. Then if the discount factor  $\gamma < 0.1$  the optimal policy takes the agent one step North from the start state into  $A$ , if the discount factor  $\gamma > 0.1$  the optimal policy takes the agent two steps South from the start state into  $B$ .

- b.** True. Whenever the optimal  $Q$ -function,  $Q^*$ , cannot be represented as a weighted combination of features, then the feature-based representation would not even have the expressiveness to find the optimal  $Q$ -function,  $Q^*$ .
- c.** True.  $V_k$  and  $V_{k+1}$  are at most  $\gamma^k \max |R|$  different so value iteration converges for any  $0 \leq \gamma < 1$ . See lecture slides for more details.
- d.** False. A simple counter-example suffices. Assume that the agent can reach a reward of \$10 within 2 time steps or a reward of \$1 within one time step. If the horizon is less than two then the agent will aim for the latter. See lecture slides for more details.

### Exercise 17.2.#CUGD

Consider an  $(N + 1) \times (N + 1) \times (N + 1)$  cubic gridworld. Luckily, all the cells are empty – there are no walls within the cube. For each cell, there is an action for each adjacent facing open cell (no corner movement), as well as an action *stay*. The actions all move into the corresponding cell with probability  $p$  but stay with probability  $1 - p$ . *Stay* always stays. The reward is always zero except when you enter the goal cell at  $(N, N, N)$ , in which case it is 1 and the game then ends. The discount is  $0 < \gamma < 1$ .

- a.** How many iterations  $k$  of value iteration will there be before  $V_k(0, 0, 0)$  becomes non-zero? If this will never happen, write *never*.
  - b.** If and when  $V_k(0, 0, 0)$  first becomes non-zero, what will it become? If this will never happen, write *never*.
  - c.** What is  $V^*(0, 0, 0)$ ? If it is undefined, write *undefined*.
- 
- a.**  $3N$ . At  $V_0$  the value of the goal is correct. At  $V_1$  all cells next to the goal are non-zero, at  $V_2$  all cells next to those are nonzero, and so on.
  - b.**  $(\gamma p)^{3N}/\gamma$ . The value update of a cell  $c$  in this problem is  $V'(c) = p(r_{c'} + \gamma V(c')) + (1 - p)V(c)$ . The first time the value of a state becomes non-zero, the value is  $V'(c) = p(r_{c'} + \gamma V(c'))$ .

$$V'(g) = p(1 + \gamma V(goal)) = p \quad \text{for a cell } g \text{ adjacent to the goal.}$$

$$V'(c) = p\gamma V(c') \quad \text{for other cells since the reward is 0.}$$

Carrying out the value recursion, the goal reward  $+1$  is multiplied by  $p$  for the step to the goal and  $p\gamma$  for each further step. The number of steps from the start to the goal is  $3N$ , the Manhattan distance in the cube. The first non-zero  $V(0, 0, 0)$  is  $(\gamma p)^{3N}/\gamma$  since every step multiplies in  $p\gamma$  except the last step to the goal, which multiplies in  $p$ . Equivalently, the first non-zero value is  $p(\gamma p)^{3N-1}$  with  $(\gamma p)^{3N-1}$  for all the steps from the start to a cell adjacent to the goal and  $p$  for the transition to the goal.

c.  $\frac{1}{\gamma} \left( \frac{\gamma p}{1-\gamma+\gamma p} \right)^{3N}$

To see why, let  $V^*(d)$  be the value function of states whose Manhattan distance from the goal is  $d$ . By symmetry, all states with the same Manhattan distance from the goal will have the same value. Write the Bellman equations:

$$V^*(d) = \gamma(1-p)V^*(d) + \gamma p V^*(d-1) \quad \text{for all } d > 1$$

$$V^*(1) = p + \gamma(1-p)V^*(1) + \gamma p V^*(0)$$

$$V^*(0) = \gamma V^*(0)$$

and solve starting with  $V^*(0) = 0$  to get the answer.

### Exercise 17.2.#VALG

Suppose we run value iteration in an MDP with only non-negative rewards (that is,  $R(s, a, s') \geq 0$  for any  $(s, a, s')$ ). Let the values on the  $k$ th iteration be  $V_k(s)$  and the optimal values be  $V^*(s)$ . Initially, the values are 0 (that is,  $V_0(s) = 0$  for any  $s$ ).

a. Mark *all* of the options that are *guaranteed* to be true.

- (i) For any  $s, a, s'$ ,  $V_1(s) = R(s, a, s')$
- (ii) For any  $s, a, s'$ ,  $V_1(s) \leq R(s, a, s')$
- (iii) For any  $s, a, s'$ ,  $V_1(s) \geq R(s, a, s')$
- (iv) None of the above are guaranteed to be true.

b. Mark *all* of the options that are *guaranteed* to be true.

- (i) For any  $k, s$ ,  $V_k(s) = V^*(s)$
- (ii) For any  $k, s$ ,  $V_k(s) \leq V^*(s)$
- (iii) For any  $k, s$ ,  $V_k(s) \geq V^*(s)$
- (iv) None of the above are guaranteed to be true.

a. (iv) only.

$$V_1(s) = \max_a \sum_{s'} T(s, a, s')R(s, a, s') \text{ (using the Bellman equation and setting } V_0(s') = 0\text{).}$$

Now consider an MDP where the best action in state X is clockwise, which goes to state Y with a reward of 6 with probability 0.5 and goes to state Z a reward of 4 with probability 0.5. Then  $V_1(X) = 0.5(6) + 0.5(4) = 5$ .

Notice that setting  $(s, a, s') = (X, \text{clockwise}, Z)$  gives a counterexample for the second option and  $(s, a, s') = (X, \text{clockwise}, Y)$  gives a counterexample for the third option.

b. (ii) only.

Intuition: Values can never decrease in an iteration. In the first iteration, since all rewards are positive, the values increase. In any other iteration, the components that contribute to  $V_{k+1}(s)$  are  $R(s, a, s')$  and  $V(s')$ .  $R(s, a, s')$  is the same across all iterations, and  $V(s')$  increased in the previous iteration, so we expect  $V_{k+1}(s)$  to increase as well.

More formally, we can prove  $V_k(s) \leq V_{k+1}(s)$  by induction.

Base Case:  $V_1(s) = \max_a \sum_{s'} T(s, a, s') R(s, a, s')$ .

Since  $R(s, a, s') \geq 0$ ,  $T(s, a, s') \geq 0$ , we have  $V_1(s) \geq 0$ , and so  $V_0(s) \leq V_1(s)$ .

Induction:  $V_{k+1}(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$

$\geq \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_{k-1}(s')]$  (using  $V_k(s') \geq V_{k-1}(s')$  from the inductive hypothesis)

$$= V_k(s).$$

This immediately leads to  $V_k(s) \leq V^*(s)$  (since we can think of  $V^*(s)$  as  $V_\infty(s)$ ).

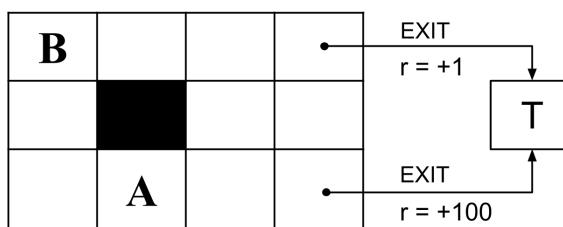


Figure S17.3 MDP for Exercise 17.GMDP.

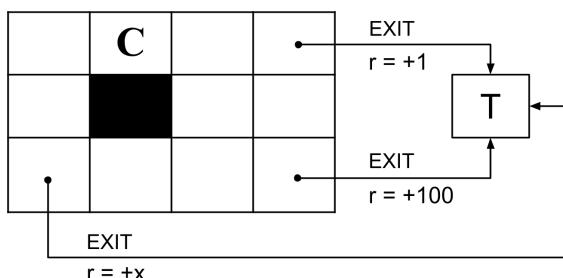


Figure S17.4 MDP for Exercise 17.GMDP.

### Exercise 17.2.#GMDP

- a. Please indicate if the following statements are true or false.

- (i) Let  $A$  be the set of all actions and  $S$  the set of states for some MDP. Assuming that  $|A| \ll |S|$ , one iteration of value iteration is generally faster than one iteration of policy iteration that solves a linear system during policy evaluation.
- (ii) For any MDP, changing the discount factor does not affect the optimal policy for the MDP.

The following problem will take place in various instances of a grid world MDP. Shaded cells represent walls. In all states, the agent has available actions  $\uparrow, \downarrow, \leftarrow, \rightarrow$ .

Performing an action that would transition to an invalid state (outside the grid or into a wall) results in the agent remaining in its original state. In states with an arrow coming out, the agent has an *additional* action *EXIT*. In the event that the *EXIT* action is taken, the agent receives the labeled reward and ends the game in the terminal state  $T$ . Unless otherwise stated, all other transitions receive no reward, and all transitions are deterministic.

For all parts of the problem, assume that value iteration begins with all states initialized to zero, i.e.,  $V_0(s) = 0 \forall s$ . **Let the discount factor be  $\gamma = \frac{1}{2}$  for all following parts.**

- b.** Suppose that we are performing value iteration on the grid world MDP in Figure S17.3.
  - (i) What's the optimal values for A and B?
  - (ii) After how many iterations  $k$  will we have  $V_k(s) = V^*(s)$  for all states  $s$ ? If it never occurs, write “never”.
  - (iii) Suppose that we wanted to re-design the reward function. For which of the following new reward functions would the optimal policy **remain unchanged**? Let  $R(s, a, s')$  be the original reward function.
    - (I)  $R_1(s, a, s') = 10R(s, a, s')$
    - (II)  $R_2(s, a, s') = 1 + R(s, a, s')$
    - (III)  $R_3(s, a, s') = R(s, a, s')^2$
    - (IV)  $R_4(s, a, s') = -1$
    - (V) None
- c.** For the problem in Figure S17.4, we add a new state in which we can take the *EXIT* action with a reward of  $+x$ .
  - (i) For what values of  $x$  is it *guaranteed* that our optimal policy  $\pi^*$  has  $\pi^*(C) = \leftarrow$ ?
  - (ii) For what values of  $x$  does value iteration take the **minimum** number of iterations  $k$  to converge to  $V^*$  for all states? Write  $\infty$  and  $-\infty$  if there is no upper or lower bound, respectively.
  - (iii) Fill the box with value  $k$ , the **minimum** number of iterations until  $V_k$  has converged to  $V^*$  for all states.

- a.** Fill out the following True/False questions.

- (i) True. One iteration of value iteration is  $O(|S|^2|A|)$ , whereas one iteration of policy iteration is  $O(|S|^3)$ , so value iteration is generally faster when  $|A| \ll |S|$
- (ii) False. Consider an infinite horizon setting where we have 2 states  $A, B$ , where we can alternate between  $A$  and  $B$  forever, gaining a reward of 1 each transition, or exit from  $B$  with a reward of 100. In the case that  $\gamma = 1$ , the optimal policy is to forever oscillate between  $A$  and  $B$ . If  $\gamma = \frac{1}{2}$ , then it is optimal to exit.
- b.** (i)  $V^*(A) = 25, V^*(B) = \frac{25}{8}$

- (ii) 6
- (iii) (I), (II), (III)

$R_1$ : Scaling the reward function does not affect the optimal policy, as it scales all Q-values by 10, which retains ordering

$R_2$ : Since reward is discounted, the agent would get more reward exiting than infinitely cycling between states

$R_3$ : The only positive reward remains to be from exiting state +100 and +1, so the optimal policy doesn't change

$R_4$ : With negative reward at every step, the agent would want to exit as soon as possible, which means the agent would not always exit at the bottom-right square.

- c. (i)  $50 < x < \infty$ .

We go left if  $Q(C, \leftarrow) > Q(C, \rightarrow)$ .  $Q(C, \leftarrow) = \frac{1}{8}x$ , and  $Q(C, \rightarrow) = \frac{100}{16}$ .

Solving for  $x$ , we get  $x > 50$ .

- (ii)  $50 \leq x \leq 200$ . The two states that will take the longest for value iteration to become non-zero from either  $+x$  or  $+100$ , are states  $C$ , and  $D$ , where  $D$  is defined as the state to the right of  $C$ .  $C$  will become nonzero at iteration 4 from  $+x$ , and  $D$  will become nonzero at iteration 4 from  $+100$ . We must bound  $x$  so that the optimal policy at  $D$  does not choose to go to  $+x$ , or else value iteration will take 5 iterations. Similar reasoning for  $D$  and  $+x$ . Then our inequalities are  $\frac{1}{8}x \geq \frac{100}{16}$  and  $\frac{1}{16}x \leq \frac{100}{8}$ . Simplifying, we get the following bound on  $x$ :  $50 \leq x \leq 200$

- (iii) 4. See the explanation for the part above

### Exercise 17.2.#LQRX

Consider the following deterministic MDP with 1-dimensional continuous states and actions and a finite task horizon:

**State Space  $\mathcal{S}$ :**  $\mathbb{R}$

**Action Space  $\mathcal{A}$ :**  $\mathbb{R}$

**Reward Function:**  $R(s, a, s') = -qs^2 - ra^2$  where  $r > 0$  and  $q \geq 0$

**Deterministic Dynamics/Transition Function:**  $s' = cs + da$  (i.e., the next state  $s'$  is a deterministic function of the action  $a$  and current state  $s$ )

**Task Horizon:**  $T \in \mathbb{N}$

**Discount Factor:**  $\gamma = 1$  (no discount factor)

Hence, we would like to maximize a quadratic reward function that rewards small actions and staying close to the origin. In this problem, we will design an optimal agent  $\pi_t^*$  and also solve for the optimal agent's value function  $V_t^*$  for all timesteps.

By induction, we will show that  $V_t^*$  is quadratic. Observe that the base case  $t = 0$  trivially holds because  $V_0^*(s) = 0$ . For all parts below, assume that  $V_t^*(s) = -p_t s^2$  (Inductive Hypothesis).

- a. (i) Write the equation for  $V_{t+1}^*(s)$  as a function of  $s$ ,  $q$ ,  $r$ ,  $a$ ,  $c$ ,  $d$ , and  $p_t$ . If your expression contains  $\max$ , you do not need to simplify the  $\max$ .
  - (ii) Now, solve for  $\pi_{t+1}^*(s)$ . Recall that you can find local maxima of functions by computing the first derivative and setting it to 0.
- b. Assume  $\pi_{t+1}^* = k_{t+1}s$  for some  $k_{t+1} \in \mathbb{R}$ . Solve for  $p_{t+1}$  in  $V_{t+1}^*(s) = -p_{t+1}s^2$ .

a. (i)  $V_{t+1}^*(s) = \max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(cs + da)^2$

$$\begin{aligned} V_{t+1}^*(s) &= \max_{a \in \mathbb{R}} R(s, a, s') + V_t^*(s') \\ &= \max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(s')^2 \\ &= \max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(cs + da)^2 \end{aligned}$$

(ii)  $\pi_{t+1}^*(s) = -\frac{cdp_t}{r+p_td^2}s$

We want to solve for the  $a$  that maximizes the value of  $V_{t+1}^*(s)$ . This is equivalent to maximizing  $-ra^2 - p_{t+1}(cs + da)^2$ . Differentiate this function, set it to 0 and solve for  $a$ .

$$\frac{d}{da}[-ra^2 - p_t(cs + da)^2] = -2ra - 2(cs + da)p_td = 0$$

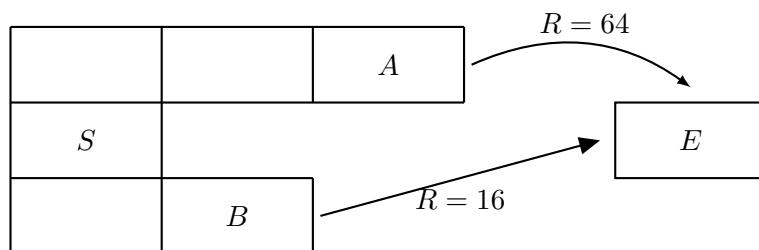
Solving for  $a$ , we find that:

$$\pi_{t+1}^*(s) = -\frac{cdp_t}{r + p_td^2}s = k_{t+1}s$$

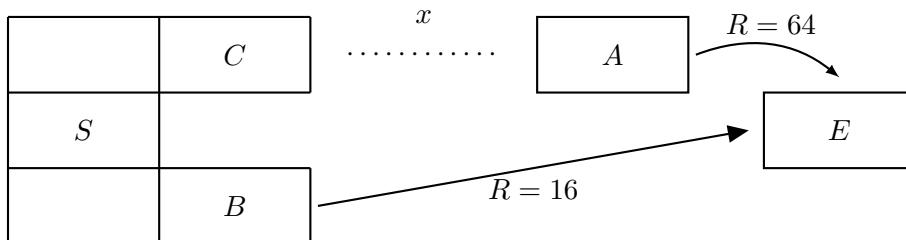
Observe that the optimal policy is a linear function of the state.

b.  $p_{t+1} = q + rk_{t+1}^2 + p_t(c + dk_{t+1})^2$

$$\begin{aligned} V_{t+1}^*(s) &= R(s, \pi_{t+1}^*(s), s') + V_t^*(s') \\ &= -qs^2 - r(k_{t+1}s)^2 - p_ts'^2 \\ &= -qs^2 - r(k_{t+1}s)^2 - p_t(cs + d(k_{t+1}s))^2 \\ &= -(q + rk_{t+1}^2 + p_t(c + dk_{t+1})^2)s^2 \\ &= -p_{t+1}s^2 \end{aligned}$$



**Figure S17.5** MDP for Exercise 17.DLR.



**Figure S17.6** Second MDP for Exercise 17.DLR.

### Exercise 17.2.#DLR

Pacman finds himself inside the grid world MDP depicted in Figure S17.5. Each rectangle represents a possible state. At each state, Pacman can take actions up, down, left or right. If an action moves him into a wall, he will stay in the same state. At states A and B, Pacman can take the exit action to receive the indicated reward and enter the terminal state, E. Note  $R(s, a, s') = 0$  otherwise. Once in the terminal state the game is over and no actions can be taken. Let the discount factor  $\gamma = \frac{1}{2}$  for this problem, unless otherwise specified.

- a. (i) What is the optimal action at state S?  
 (ii) How many iterations  $k$  will it take before  $V_k(S) = V^*(S)$ ?  
 (iii) What are all the values that  $V_k(S)$  will take on during the entire process of value iteration.
- b. Now Ghost wants to mess with Pacman. She wants to change some of the rules of this grid world so that Pacman does not exit from state A. **All subquestions are independent** of each other so consider each change on its own.
  - (i) First, Ghost wants to change the discount factor. Write a bound on the discount factor that guarantees Pacman exits from B instead of A. Any valid value of  $\gamma \in (0, 1]$  that satisfies your inequality should cause Pacman to exit from B instead of A.
  - (ii) Next, Ghost thinks she can change the reward function to accomplish this. Write a bound on the reward from A,  $R(A, \text{exit}, E)$ , that guarantees Pacman exits from B instead of A? Any value of  $R(A, \text{exit}, E)$  that satisfies your inequality should cause Pacman to exit from B instead of A.
  - (iii) Ghost came up with a bunch of reward functions,  $R'(s, a, s')$ . Select the new reward functions that cause Pacman not to exit from state A. Note  $R(s, a, s')$  is the original reward function from the problem description so the reward from every state is going to be affected.
    - (A)  $R'(s, a, s') = 1 + R(s, a, s')$
    - (B)  $R'(s, a, s') = 100 + R(s, a, s')$
    - (C)  $R'(s, a, s') = -1 + R(s, a, s')$
    - (D)  $R'(s, a, s') = -100 + R(s, a, s')$
    - (E)  $R'(s, a, s') = -R(s, a, s')$

- (F)  $R'(s, a, s') = 2R(s, a, s')$
- (iv) Ghost realizes she can stop Pacman from exiting from A by adding a certain amount of grids,  $x$ , in between C and A as depicted in Figure S17.6. Give a lower bound for  $x$  that **guarantees** Pacman does not exit from A.
- c. Another way that Ghost can mess with Pacman is by choosing parameters such that Pacman's value iteration never converges. Select which reward function and discount factor pairs cause value iteration to never converge.
- (A)  $R'(s, a, s') = 100 + R(s, a, s')$ ,  $\gamma = 0.9$   
 (B)  $R'(s, a, s') = -100 + R(s, a, s')$ ,  $\gamma = 0.9$   
 (C)  $R'(s, a, s') = -1 + R(s, a, s')$ ,  $\gamma = 1.0$   
 (D)  $R'(s, a, s') = 1 + R(s, a, s')$ ,  $\gamma = 1.0$

- a. (i) Up.

Pacman can get  $64 * .5^3 = 8$  from exiting from A and he can get  $16 * .5^2 = 4$  from exiting from B so he should move up to go towards A.

- (ii)  $k = 4$

It takes one iteration for A to learn its value. One more iteration for the node to the left of A. Another iteration for the node to the left of that (and above S). And one last iteration for S to learn its value of 8 from the node above it. That is 4 iterations in total.

- (iii) 0, 4, 8

All values start at 0. Then  $V_k(S)$  will equal 4 when it learns it can get 4 from exiting from B after 3 iterations (since B is closer than A). Finally it will update to 8 when it learns it can get 8 from exiting from A.

- b. (i)  $\gamma < \frac{1}{4}$

Since A is farther away from S than B, the rewards from A have the discount factor applied more than the rewards from B. Thus if we can find a discount factor where the rewards are equal any discount factor less than ( $\gamma$ ) that will cause Pacman to exit from B. If we choose a discount factor of .25 then the node to the left of A will have a value of  $64 * .25 = 16$  which is the same as B (and both of those nodes are the same distance from S). So .25 causes the rewards to be the same and is our answer.

- (ii)  $R(A, \text{exit}, E) < 32$

Similarly to part bi) we just need to choose a reward value that causes the rewards from A to equal the rewards from B. If we choose 32 the total rewards Pacman can get from A is  $32 * .5^3 = 4$  which is the same as the rewards from B. Thus if the rewards from A are any amount less than ( $\gamma$ ) that Pacman will exit from B.

- (iii) (B), (D), (E)

For this problem we need to analyze which reward functions will cause Pacman to either oscillate infinitely or exit from B. The rewards that add 1 and -1 do not change the rewards enough to accomplish this. Also scaling the rewards by 2

changes nothing comparatively so it also doesn't work. Adding 100 to every reward causes Pacman to change states infinitely instead of exiting because he can get  $\frac{100}{1-0.5} = 200$  from going back and forth. Thus he won't exit from A in this case. The reward function that adds -100 to every reward and the function that is just the negative of the old reward both cause Pacman to want to exit the game as fast as possible which is through B.

- (iv)  $x \geq 2$

By adding grids we can apply the discount factor to the rewards from A as many times as we want. If we add one grid the reward from C  $64 * .5^2 = 16$  which is the same as from B (and C and B are the same distance from S). Since the problem said guaranteed, we need to add one more grid to make sure the reward from A is strictly less than the reward from C so the answer is 2.

- c. D.

Note any discount factor less than 1 will always converge by a proof from lecture. The function that added -1 to every reward will also converge because Pacman will want to exit the game instead of accumulating an infinite negative reward. The function that added 1 to every reward will never converge because Pacman can accumulate infinite rewards by going back and forth.

## 17.3 Bandit Problems

---

### Exercise 17.3.#KATV

Figure 17.13 shows two MDPs: one,  $M$ , represents a two-armed bandit where one has the choice to continue with the first arm or to switch permanently to a second arm with fixed reward  $\lambda$ ; the other, a **restart MDP**  $M^s$ , gives one a choice to continue with the first arm or restart the sequence. The figure illustrates the construction of  $M^s$  for the case where  $M$  has a deterministic reward sequence and just two arms (including the  $\lambda$ -arm). Explain how to construct  $M^s$  when  $M$  has  $k + 1$  arms (including the  $\lambda$ -arm) and each arm is a general MRP. Show that the value of  $M^s$  equals the minimum value of  $\lambda$  such that one would be indifferent in  $M$  between pulling the best arm and switching to the  $\lambda$ -arm forever.

See Katehakis and Veinott (1987) for the original proof and Cowan and Katehakis (2015) for a more up-to-date formulation of the problem.

### Exercise 17.3.#SELC

At each step in a **selection problem** (Section 17.3.4), an agent samples from one of  $k$  arms and obtains some information about the true utility of that arm. Selection problems differ from bandit problems in two ways: (1) the cost  $c$  of each sample is fixed, independent of the value of the arm, and (2) at some point the agent must stop sampling and commit to one of the arms. Let  $\mu_i$  be the true utility of arm  $i$  and let  $e_1, \dots, e_N$  be the evidence obtained in the first  $N$  samples, after which the agent stops and commits. Then the agent's overall

expected utility, assuming it commits to what appears to be the best arm, is given by

$$E[-cN + \max_i E[\mu_i | e_1, \dots, e_N]].$$

For parts (a–d) we will assume that the arms are Bernoulli arms, i.e., each sample returns 0 or 1 and the true utility  $\mu_i$  is the probability that the arm returns a 1 on any given sample. The agent is assumed to have a prior probability  $P_i(\mu_i)$  for each arm.

- Give a precise formulation of the Bernoulli selection problem as an undiscounted MDP  $M$  with countably many states.
- Because the state space of  $M$  is unbounded, some policies can keep sampling forever. A policy  $\pi$  that samples forever with finite probability (bounded away from zero) incurs infinite expected cost  $cN$ . Explain why an optimal policy for  $M$  stops with probability 1.
- Give an explicit upper bound on the expected number of samples taken by an optimal policy.
- Consider the two-armed selection problem where the agent knows that  $\mu_1 = 1/2$  and that  $\mu_2 = 1/3$  or  $2/3$  with equal probability. Give a qualitative description of the optimal policy for this problem and show that it is *possible* for the optimal policy to keep sampling forever without ever committing to one arm or the other. Explain how this result is consistent with your answers in (b) and (c).
- Now consider a selection problem with three *deterministic* arms that always return a fixed amount. (Such arms need be sampled at most once!) The agent knows that  $\mu_1 = -1.5$  or  $+1.5$  with equal probability,  $\mu_2 = 0.25$  or  $+1.75$  with equal probability, and  $\mu_3 = \lambda$ . There are three possible actions: sample arm 1, sample arm 2, and stop. Formulate and solve the corresponding MDP and plot the  $Q$ -value of each action as a function of  $\lambda$  for  $\lambda \in [-2, +2]$ . Show that the optimal choice between arms 1 and 2 can change depending on the value of  $\lambda$ , and hence that there is no index function for this problem.

[[TBC]]  
[[need exercises]]

## 17.4 Partially Observable MDPs

### Exercise 17.4.#POMD

Let the initial belief state  $b_0$  for the  $4 \times 3$  POMDP on page 588 be the uniform distribution over the nonterminal states, i.e.,  $\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$ . Calculate the exact belief state  $b_1$  after the agent moves *Left* and its sensor reports 1 adjacent wall. Also calculate  $b_2$  assuming that the same thing happens again.

The belief state update is given by Equation (17.16), i.e.,

$$b'(s') = \alpha P(e | s') \sum_s P(s' | s, a) b(s) .$$

It may be helpful to compute this in two stages: update for the action, then update for the observation. The observation probabilities  $P(e | s')$  are all either 0.9 (for squares that actually have one wall) or 0.1 (for squares with two walls). The following table shows the results. Note in particular how the probability mass concentrates on (3,2).

|       |        | <i>Left</i> | 1 wall | <i>Left</i> | 1 wall |
|-------|--------|-------------|--------|-------------|--------|
| (1,1) | .11111 | .20000      | .06569 | .09197      | .02090 |
| (1,2) | .11111 | .11111      | .03650 | .04234      | .00962 |
| (1,3) | .11111 | .20000      | .06569 | .09197      | .02090 |
| (2,1) | .11111 | .11111      | .03650 | .27007      | .06136 |
| (2,3) | .11111 | .11111      | .03650 | .05985      | .01360 |
| (3,1) | .11111 | .11111      | .32847 | .06861      | .14030 |
| (3,2) | .11111 | .11111      | .32847 | .30219      | .61791 |
| (3,3) | .11111 | .02222      | .06569 | .03942      | .08060 |
| (4,1) | .11111 | .01111      | .00365 | .00036      | .00008 |
| (4,2) | 0      | .01111      | .03285 | .03321      | .06791 |
| (4,3) | 0      | 0           | 0      | 0           | 0      |

#### Exercise 17.4.#PDPT

Consider a version of the two-state POMDP on page 591 in which the sensor is 90% reliable in state 0 but provides no information in state 1 (that is, it reports 0 or 1 with equal probability). Analyze, either qualitatively or quantitatively, the utility function and the optimal policy for this problem.

Policies for the 2-state MDP all have a threshold belief  $p$ , such that if  $b(0) > p$  then the optimal action is *Go*, otherwise it is *Stay*. The question is, what does this change do to the threshold? By making sensing more informative in state 0 and less informative in state 1, the change has made state 0 more desirable, hence the threshold value  $p$  increases.

[[need exercices]]

## 17.5 Algorithms for Solving POMDPs

#### Exercise 17.5.#TPDP

What is the time complexity of  $d$  steps of POMDP value iteration for a sensorless environment?

In a sensorless environment, POMDP value iteration is essentially the same as ordinary state-space search—the branching occurs only on action choices, not observations. Hence the time

complexity is  $O(|A|^d)$ .

### Exercise 17.5.#POMC

Silver and Veness (2011) describe the POMCP algorithm for approximate online decision-making in POMDPs. It is essentially a combination of the UCT algorithm and a particle filter to represent and update belief states.

- a. Implement the POMCP algorithm as described. The algorithm itself should make no assumptions about how the POMDP model is implemented; that is, view it as an abstract data type.
- b. For this part, use simple, atomic representations for states and percepts, with matrices for transition and sensor models. (See, for example, Section 14.3.) Create a model for the 4x3 POMDP and apply the algorithm to solve it. Measure the POMCP agent's performance as a function of the number of particles and the number of rollouts.
- c. Now consider models represented as dynamic decision networks. Apply POMCP to solve vacuum worlds with stochastic dirt (see, for example, Figure 14.20), with a reward function based on the number of clean squares at the current step.

[[TBC]]

[[need exercises]]

# EXERCISES 18

## MULTIAGENT DECISION MAKING

### 18.1 Properties of Multiagent Environments

#### **Exercise 18.1.#GTDF**

Define the following in your own words:

- a. Multiagent system
  - b. Multibody planning
  - c. Coordination problem
  - d. Agent design
  - e. Mechanism design
  - f. Cooperative game
  - g. Non-cooperative game
- 
- a. Multiagent system: any environment with more than one actor.
  - b. Multibody planning: when a single central agent creates a joint plan for multiple physical units (such as a fleet of delivery robots).
  - c. Agent design: designing an agent that can act optimally in an environment (or at least approximate optimality).
  - d. Mechanism design: defining the rules of a game for collective good.
  - e. Cooperative game: a game in which it is possible for agents to enter into a binding agreement, without the possibility of renegeing on the agreement.
  - f. Non-cooperative game: a game in which no binding agreement is possible, and agents must consider the other agent's strategies.

#### **Exercise 18.1.#CCGT**

Give some examples, from movies or literature, of bad guys with a formidable army (robotic or otherwise) that inexplicably is under centralized control rather than more robust multiagent control, so that all the good guys have to do is defeat the one master controller in order to win.

Some examples include:

Movies: Independence Day, Edge of Tomorrow, Wizard of Oz, Tron, Logan's Run, I, Robot.

Literature: The Machine Stops, The Moon Is a Harsh Mistress.

### Exercise 18.1.#GLDB

The British TV game show called “GoldenBalls” had a game element in which two players independently choose whether to “split” or “steal” the jackpot. Read a description of the rules at [en.wikipedia.org/wiki/Golden\\_Balls#Split\\_or\\_Steal?](https://en.wikipedia.org/wiki/Golden_Balls#Split_or_Steal?).

- Formulate the game as a payoff matrix, assuming utility is equal to monetary reward, and analyze it using solution concepts and social welfare concepts.
- What other factors might play a role in the formulation of utility? How does this change the game?
- Now watch the video at [www.youtube.com/watch?v=S0qjK3TWZE8](https://www.youtube.com/watch?v=S0qjK3TWZE8) or [tinyurl.com/ofruebv](http://tinyurl.com/ofruebv) and explain what is happening using game theory concepts.

- The payoff matrix is:

|              | <i>split</i> | <i>steal</i> |
|--------------|--------------|--------------|
| <i>split</i> | 50, 50       | 0, 100       |
| <i>steal</i> | 100, 0       | 0, 0         |

(steal) is a dominant strategy, so (steal, steal) is a dominant strategy equilibrium. The only outcome that is *not* a Nash equilibrium is (split, split). The only outcome that does not maximise utilitarian social welfare is (steal, steal). The outcomes (steal, steal) and (split, split) maximize egalitarian social welfare.

- As always, the value of money might not be linear with the amount of money. In this game, since the TV show is seen by many viewers, players may derive utility from viewers collective perception of them: positive utility from being seen as being fair (splitting), or honest (keeping their word), or clever (influencing the opponent to arrive at a good outcome); or negative utility from being seen as greedy (stealing) or naive (letting their opponent steal from them). It is not clear how to add the value of viewer perception to the value of money.
- Both players want to avoid the (steal, steal) dominant strategy equilibrium. In the video, Nick realizes that declaring “trust me, I’m going to play split” is not a credible threat. So Nick instead declares (1) “trust me, I’m going to play steal” and (2) “I will give you half the money,” thereby *changing the game*. The first part is a credible threat because steal is a dominant strategy. The second part is only partially credible. But if Abraham believes the first part is credible, and believes the second part with anything more than probability 0, then Abraham is better off playing split than steal. Interestingly, Nick then plays split, despite his promise/threat, because Nick realizes that there is a chance Abraham will play steal. (Perhaps Nick could then convince Abraham to share some of the money, on the grounds that he demonstrated good faith.)

## 18.2 Non-Cooperative Game Theory

### Exercise 18.2.#SWG

Either prove or disprove each of the following statements in the context of  $2 \times 2$  games (you may find it helpful to do proofs by providing examples or counter examples).

- If a player  $i$  has a dominant strategy in a game, then in every Nash equilibrium of that game player  $i$  will choose a dominant strategy.
- Every dominant strategy equilibrium of a game is a Nash equilibrium.
- Every Nash equilibrium of a game is a dominant strategy equilibrium.
- If a game outcome maximises utilitarian social welfare, then is Pareto efficient.
- If a game outcome is Pareto efficient, then it maximises utilitarian social welfare.
- If all utilities in a game are positive, then any outcome that maximises the product of utilities of players is Pareto efficient.
- If all utilities in a game are positive, then any Pareto efficient outcome of the game will maximise the product of utilities of players.

- True.
- True.
- False.
- True.
- False.
- True.
- False.

### Exercise 18.2.#DOMQ

Show that a dominant strategy equilibrium is a Nash equilibrium, but not vice versa.

This question is simple a matter of examining the definitions. In a dominant strategy equilibrium  $[s_1, \dots, s_n]$ , it is the case that for every player  $i$ ,  $s_i$  is optimal for every combination  $t_{-i}$  by the other players:

$$\forall i \ \forall t_{-i} \ \forall s'_i \ [s_i, t_{-i}] \succsim [s'_i, t_{-i}] .$$

In a Nash equilibrium, we simply require that  $s_i$  is optimal for the particular current combination  $s_{-i}$  by the other players:

$$\forall i \ \forall s'_i \ [s_i, s_{-i}] \succsim [s'_i, s_{-i}] .$$

Therefore, dominant strategy equilibrium is a special case of Nash equilibrium. The converse does not hold, as we can show simply by pointing to the CD/DVD game, where neither of the Nash equilibria is a dominant strategy equilibrium.

**Exercise 18.2.#RPSG**

In the children's game of rock–paper–scissors each player reveals at the same time a choice of rock, paper, or scissors. Paper wraps rock, rock blunts scissors, and scissors cut paper. In the extended version rock–paper–scissors–fire–water, fire beats rock, paper, and scissors; rock, paper, and scissors beat water; and water beats fire. Write out the payoff matrix and find a mixed-strategy solution to this game.

In the following table, the rows are labelled by A's move and the columns by B's move, and the table entries list the payoffs to A and B respectively.

|   | R    | P    | S    | F    | W    |
|---|------|------|------|------|------|
| R | 0,0  | -1,1 | 1,-1 | -1,1 | 1,-1 |
| P | 1,-1 | 0,0  | -1,1 | -1,1 | 1,-1 |
| S | -1,1 | 1,-1 | 0,0  | -1,1 | 1,-1 |
| F | 1,-1 | 1,-1 | 1,-1 | 0,0  | -1,1 |
| W | -1,1 | -1,1 | -1,1 | 1,-1 | 0,0  |

Suppose A chooses a mixed strategy  $[r : R; p : P; s : S; f : F; w : W]$ , where  $r + p + s + f + w = 1$ . The payoff to A of B's possible pure responses are as follows:

$$R : +p - s + f - w$$

$$P : -r + s + f - w$$

$$S : +r - p + f - w$$

$$F : -r - p - s + w$$

$$W : +r + p + s - f$$

It is easy to see that no option is dominated over the whole region. Solving for the intersection of the hyperplanes, we find  $r = p = s = 1/9$  and  $f = w = 1/3$ . By symmetry, we will find the same solution when B chooses a mixed strategy first.

**Exercise 18.2.#PIRT**

Consider the following scenario:

*Five pirates wish to divide the loot of a 100 gold pieces. They are democratic pirates, in their own way, and it is their custom to make such divisions in the following manner: The fiercest pirate makes a proposal about the division and everybody (including the proposer) votes on it. If 50 percent or more are in favor, the proposal is implemented. Otherwise the proposer is thrown overboard, and the procedure is repeated with the next fiercest pirate.*

*All the pirates enjoy throwing their fellows overboard, but given a choice they prefer more gold. Of course, they intensely dislike being thrown overboard themselves. (Specifically, we say that each pirate assigns a utility of 1 to each gold piece, a utility of 1/100 to throwing another pirate overboard, and a utility of -1000 to being thrown overboard). All pirates are rational and know that the*

*other pirates are also rational. Moreover, no two pirates are equally fierce, so there is a precise order, known to all, of making proposals. The gold pieces are indivisible and arrangements to share pieces are not permitted, because no pirate trusts his mates.*

What proposal should the fiercest pirate make?

We proceed by backwards induction in order to derive the proposal made by the fiercest pirate; we call the first proposer 1, the second 2 and so on.

pirate 5 remains: Pirate 5 receives all the gold, plus  $\frac{1}{100}$  for each pirate tossed overboard, for a total utility of  $100 + \frac{4}{100}$ , while the rest get at most  $-999$  (they get some utility from throwing other pirates overboard, before being thrown overboard themselves).

4 proposes: Even if pirate 5 rejects a division, pirate 4's vote is 50% of the total vote, so any division proposed by pirate 4 is accepted. Thus, the best choice for player 4 is to keep all the money while player 5 gets nothing.

3 proposes: In order not to get tossed overboard, pirate 3 must propose a division that will guarantee either 4 or 5 a better payoff than what they are receiving should only the two of them remain. He cannot sway pirate 4, as he can get  $100 + \frac{3}{100}$  by rejecting the offer, and the most pirate 3 can offer him is 100, for a total utility of  $100 + \frac{2}{100}$  (i.e., pirate 4 prefers to get 100 by throwing pirate 3 overboard rather than accepting his offer). However, pirate 3 can offer 1 to pirate 5 (and nothing to pirate 4)—pirate 5 will accept as this guarantees him at least some gold. This is obviously the best that pirate 3 can do, so he can guarantee 99 coins for himself.

2 proposes: In order to have his proposal accepted, pirate 2 needs the approval of just one other pirate. The pirate who is easiest to bribe (i.e. the one who guarantees the highest payoff to pirate 2) is pirate 4; by giving pirate 4 one gold coin (and nothing to pirates 3 and 5), pirate 2 makes pirate 4 accept the division.

1 proposes: In order to have his proposal accepted, pirate 1 needs two more votes. By giving 3 and 5 one coin each, he ensures that they will get strictly higher utility than what they can get if 2 proposes. Thus, in the unique subgame perfect Nash equilibrium of this game pirate 1 keeps 98 coins, offers 1 coin to pirates 3 and 5 and nothing to pirates 2 and 4, and this proposal is accepted since pirates 1, 3, and 5 vote for it. Note that no one gets thrown overboard!

### Exercise 18.2.#TFMG

Solve the game of *three-finger Morra*.

The payoff matrix for three-finger Morra is as follows:

|                 | <i>O: one</i>   | <i>O: two</i>   | <i>O: three</i> |
|-----------------|-----------------|-----------------|-----------------|
| <i>E: one</i>   | $E = 2, O = -2$ | $E = -3, O = 3$ | $E = 4, O = -4$ |
| <i>E: two</i>   | $E = -3, O = 3$ | $E = 4, O = -4$ | $E = -5, O = 5$ |
| <i>E: three</i> | $E = 4, O = -4$ | $E = -5, O = 5$ | $E = 6, O = -6$ |

Suppose  $E$  chooses a mixed strategy  $[p_1 : \text{one}; p_2 : \text{two}; p_3 : \text{three}]$ , where  $p_1 + p_2 + p_3 = 1$ . The payoff to  $E$  of  $O$ 's possible pure responses are as follows:

$$\begin{aligned}\text{one} &: 2p_1 - 3p_2 + 4p_3 \\ \text{two} &: -3p_1 + 4p_2 - 5p_3 \\ \text{three} &: 4p_1 - 5p_2 + 6p_3\end{aligned}$$

It is easy to see that no option is dominated over the whole region. Solving for the intersection of the hyperplanes, we find  $p_1 = 1/4$ ,  $p_2 = 1/2$ ,  $p_3 = 1/4$ . The expected value is 0.

### Exercise 18.2.#FBPK

In the game of football (“soccer” in the US), a player who is awarded a penalty kick scores about 3/4 of the time. Suppose we model a penalty kick as a game between two players, the shooter,  $S$ , and the goalkeeper,  $G$ . The shooter has 4 possible actions:

- LC: Aim for left corner of the goal.
- LM: Aim for left middle of the goal.
- RM: Aim for right middle of the goal.
- RC: Aim for right corner of the goal.

Aiming for a corner risks missing the net completely, but if the shot is on target it is difficult for the goalkeeper to make a save. Aiming more in the middle gives less risk of missing the net, but more chance for a save. The goalkeeper has 3 possible actions:

- L: Lean to the shooter's left.
- M: Stay in the middle.
- R: Lean to the shooter's right.

Leaning to one side as the shooter strikes the ball makes it easier to reach a shot to that side, but harder to reach a shot to the other side. The expected percent of goals scored, is as follows:

|    | L  | M  | R  |
|----|----|----|----|
| LC | 65 | 80 | 85 |
| LM | 50 | 55 | 90 |
| RM | 90 | 55 | 50 |
| RC | 85 | 80 | 65 |

Football is a zero-sum game, so these are the values for  $S$ , and the negative of these are the values for  $G$ . What are equilibrium strategies for  $S$  and  $G$  and what is the outcome of the game?

Also, can you come up with a slightly different payoff matrix that results in equilibrium strategies where both players use at least three of their available actions at least some of the time?

The equilibrium strategy for  $S$  is a mixed strategy: [0.5: LC, 0.5: RC]. The equilibrium strategy for  $G$  is also a mixed strategy: [0.5: L, 0.5: R]. The outcome is 75 for  $S$  (and thus -75 for  $G$ ).

|    | L  | M  | R  |
|----|----|----|----|
| LC | 67 | 80 | 85 |
| LM | 60 | 60 | 94 |
| RM | 94 | 60 | 60 |
| RC | 85 | 80 | 67 |

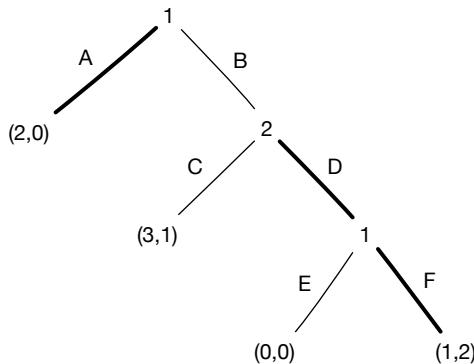
There are many possible answers to the second part of the question. Below is a payoff matrix with outcome 76.19 for  $S$  where there are two symmetric equilibrium strategies, one of which is [0.58: **LC**, 0.19: **RM**, 0.23: **RC**] for  $S$  and [0.48: **L**, 0.04: **M**, 0.48: **R**] for  $G$ .

### Exercise 18.2.#XFPS

Consider a 2 player game in which player 1 can choose  $A$  or  $B$ . The game ends if she chooses  $A$ , while it continues to player 2 if he chooses  $B$ . Player 2 can then choose  $C$  or  $D$  with the game ending if  $C$  is chosen, and continuing again to player 1 if  $D$  is chosen. Player 1 can then choose  $E$  or  $F$ , with the game ending either choice.

- a. Model this as an extensive form game.
- b. How many pure strategies does each player have?
- c. Identify the subgames of this game.
- d. Suppose that choice  $A$  gives utilities  $(2, 0)$  (i.e., 2 to player  $A$ , 0 to player  $E$ ), choice  $C$  gives  $(3, 1)$ , choice  $E$  gives  $(0, 0)$ , and  $F$  gives  $(1, 2)$ . Then what are the pure Nash equilibria of the game? What SPNE outcome(s) do you obtain through backwards induction?

- a. The extensive form:



- b. The players' pure strategies are given by:

$$\begin{aligned}\Sigma_1 &= \{AE, AF, BE, BF\} \\ \Sigma_2 &= \{C, D\}\end{aligned}$$

Hence, player 1 has four pure strategies and player 2 two.

- c. Three: one for each decision node.

- d. Modelling the game in normal form:

|    | C           | D           |
|----|-------------|-------------|
| AE | 2, 0        | <u>2, 0</u> |
| AF | 2, 0        | <u>2, 0</u> |
| BE | <u>3, 1</u> | 0, 0        |
| BF | 3, 1        | 1, 2        |

The pure Nash equilibria are:

$$(AE, D), \quad (AF, D), \quad (BE, C).$$

By backwards induction,  $(AF, D)$  can be seen to be the only the subgame perfect Nash equilibrium of this game.

### Exercise 18.2.#XFTP

Consider the following scenario:

Two players ( $N = \{1, 2\}$ ) must choose between three outcomes  $\Omega = \{a, b, c\}$ . The rule they use is the following: Player 1 goes first, and vetoes one of the outcomes. Player two then chooses one of the remaining two outcomes.

Suppose that player preferences are given by:

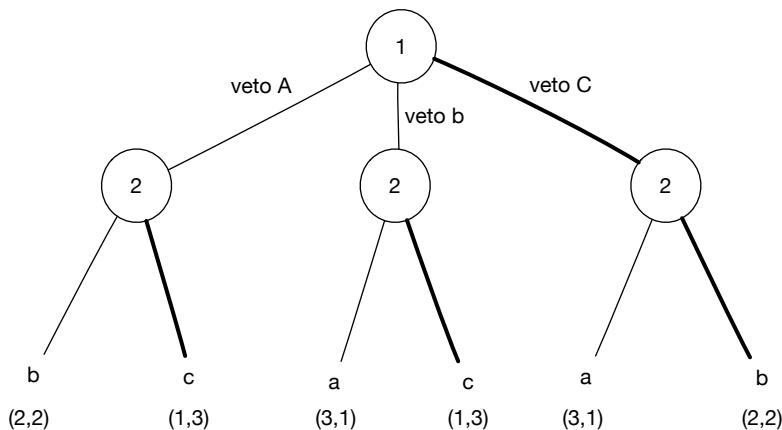
$$\begin{aligned} a &\succ_1 b \succ_1 c \\ c &\succ_2 b \succ_2 a \end{aligned}$$

- i) Express this scenario as an extensive form game.
- ii) Identify any Subgame Perfect Nash Equilibria.

First let's fix utility functions to make things clear:

$$\begin{aligned} u_1(a) &= 3 & u_1(b) &= 2 & u_1(c) &= 1 \\ u_2(a) &= 1 & u_2(b) &= 2 & u_2(c) &= 3 \end{aligned}$$

The following figure then illustrates the extensive form game (numbers inside decision nodes indicate the decision maker for that node).



Darker lines coming out from each node indicate subgame perfect Nash equilibria choices for that decision node.

Via backwards induction, the unique subgame perfect equilibrium is thus where player 1 vetoes *c*, and player 2 then chooses *b*. With our utility functions, this results in both players getting a payoff of 2.

### Exercise 18.2.#PRDG

In the *Prisoner's Dilemma*, consider the case where after each round, Ali and Bo have probability  $X$  meeting again. Suppose both players choose the perpetual punishment strategy (where each will choose *refuse* unless the other player has ever played *testify*). Assume neither player has played *testify* thus far. What is the expected future total payoff for choosing to *testify* versus *refuse* when  $X = .2$ ? How about when  $X = .05$ ? For what value of  $X$  is the expected future total payoff the same whether one chooses to *testify* or *refuse* in the current round?

For  $X = 0.2$ , the payoffs for *testify* and *refuse* are

$$0 + \sum_{t=1}^{\infty} 0.2^t \cdot (-5) = -1.25; \quad \sum_{t=0}^{\infty} 0.2^t \cdot (-1) = -1.25.$$

and for  $X = 0.05$  they are

$$0 + \sum_{t=1}^{\infty} 0.05^t \cdot (-5) \approx -0.263; \quad \sum_{t=0}^{\infty} 0.05^t \cdot (-1) = -1.053.$$

The payoffs are identical for  $X = 0.2$ .

### Exercise 18.2.#FEDG

The following payoff matrix, from Bernstein (1996), shows a game between politicians and the Federal Reserve.

|                 | Fed: contract  | Fed: do nothing | Fed: expand    |
|-----------------|----------------|-----------------|----------------|
| Pol: contract   | $F = 7, P = 1$ | $F = 9, P = 4$  | $F = 6, P = 6$ |
| Pol: do nothing | $F = 8, P = 2$ | $F = 5, P = 5$  | $F = 4, P = 9$ |
| Pol: expand     | $F = 3, P = 3$ | $F = 2, P = 7$  | $F = 1, P = 8$ |

Politicians can expand or contract fiscal policy, while the Fed can expand or contract monetary policy. (And of course either side can choose to do nothing.) Each side also has preferences for who should do what—neither side wants to look like the bad guys. The payoffs shown are simply the rank orderings: 9 for first choice through 1 for last choice. Find the Nash equilibrium of the game in pure strategies. Is this a Pareto-optimal solution? You might wish to analyze the policies of recent administrations in this light.

We apply iterated strict dominance to find the pure strategy. First, *Pol: do nothing* dominates *Pol: contract*, so we drop the *Pol: contract* row. Next, *Fed: contract* dominates *Fed: do nothing* and *Fed: expand* on the remaining rows, so we drop those columns. Finally, *Pol: expand* dominates *Pol: do nothing* on the one remaining column. Hence the only Nash equilibrium is a dominant strategy equilibrium with *Pol: expand* and *Fed: contract*. This is not Pareto optimal: it is worse for both players than the four strategy profiles in the top right quadrant.

### Exercise 18.2.#PIRI

Consider the following scenario:

*There are two pirates operating among three islands A, B, and C. On each island, two treasures are buried: a large one worth 2 and another smaller one worth 1. The prevailing winds in the area are such that from island A you can only reach island B, from island B only island C, and from island C only island A. Once on an island, the pirates only have time to excavate one treasure, before heading to the next island. The pirates are not on good terms, and if they are at the same island at the same time they will fight, and as a result, they will find no treasures (but will not suffer any additional damage). If they do not meet, both pirates will visit all three islands and each pirate will find three treasures. Each of the pirates has to decide where to start their treasure hunt.*

- Model this setting as a normal-form game.
- Compute a mixed Nash equilibrium of this game and argue that it is unique.
- Compute players' expected utilities in this equilibrium.

Note that, as soon as we know where each pirate starts, we can compute their payoffs. Thus, for each player the pure strategy space is  $\{A, B, C\}$  and the payoff matrix is given by

|   | A    | B    | C    |
|---|------|------|------|
| A | 0, 0 | 4, 5 | 5, 4 |
| B | 5, 4 | 0, 0 | 4, 5 |

|   |      |      |      |
|---|------|------|------|
| C | 4, 5 | 5, 4 | 0, 0 |
|---|------|------|------|

Next, we will compute the strategy of the first pirate. Suppose the second pirate chooses  $A$  with probability  $p$  and  $B$  with probability  $q$ . The first pirate's payoff from choosing  $A$  is  $0 \cdot p + 4 \cdot q + 5(1 - p - q)$ ; his payoff from choosing  $B$  is  $5 \cdot p + 0 \cdot q + 4(1 - p - q)$ ; his payoff from choosing  $C$  is  $4 \cdot p + 5 \cdot q + 0(1 - p - q)$ . All these payoffs must be equal. We obtain  $5 - 5p - q = 4 + p - 4q$  and hence  $6p - 3q = 1$ , and  $5 - 5p - q = 4p + 5q$  and hence  $9p + 6q = 5$ . From this we obtain  $21p = 7$  and hence  $p = 1/3$ ; substituting this into first equation we get  $q = 1/3$ . Thus, in the unique fully mixed Nash equilibrium each player chooses each strategy with probability  $1/3$ .

To compute the expected payoff of player 1 in the fully mixed Nash equilibrium, we can assume that this player chooses  $A$  (as he gets the same payoff from each pure strategy). Then his expected payoff is  $(0 + 5 + 4)/3 = 3$ .

### Exercise 18.2.#TUR

Avi and Bailey are friends, and enjoy a night out together in the pub. They each will independently decide to go either to the Turf or the Rose.

Avi mildly prefers the Rose over the Turf and would get a utility of 2 from going to the Rose with Bailey, but only 1 from the Turf.

Bailey has a very strong preference for the Turf and would get a utility of 5 for going to the Turf with Avi, and only 1 for the Rose.

If they go to different pubs, they both receive a utility of 0.

- Formally express this scenario as a strategic form game.
- Compute all Nash equilibria (pure and mixed) for the game.
- Compute the expected utility of each player in each equilibrium.

|     |   | Bailey |   |
|-----|---|--------|---|
|     |   | T      | B |
| Avi | T | 5      | 0 |
|     | B | 1      | 0 |

|        |   | Avi |   |
|--------|---|-----|---|
|        |   | T   | B |
| Bailey | T | 1   | 0 |
|        | B | 0   | 2 |

There are two pure Nash Equilibria:  $(T, T)$ , which pays off Avi with 1 and Bailey with 5, and  $(R, R)$ , which pays off Avi with 2 and Bailey with 1.

There is one mixed strategy equilibrium in which Avi plays  $T$  with probability  $1/6$  and Bailey plays  $T$  with probability  $2/3$ .

Expected utilities:

$$EU_A(p, q) = \frac{2}{3} * 1 + \frac{1}{3} * 0 = \frac{2}{3}$$

$$EU_B(p, q) = \frac{1}{6} * 5 + \frac{5}{6} * 0 = \frac{5}{6}$$

## 18.3 Cooperative Game Theory

### Exercise 18.3.#CGNE

Define the core of a cooperative game  $G = (N, v)$ , and show by way of example that the core of a cooperative game may be empty.

There are many games with an empty core. A standard one is as a game with three players and characteristic function  $v(C) = 1$  iff  $|C| \geq 2$  and  $v(C) = 0$  otherwise.

The core is the set of all payoff vectors  $(p_1, \dots, p_n)$  such that  $p_i \geq 0$  for all  $i \in [n]$ , and for all  $C \subseteq [n]$  we have  $p_C \geq v(C)$ , where  $p_C = \sum_{i \in C} p_i$ .

### Exercise 18.3.#WVCG

Recall that a weighted voting game is a cooperative game defined by a structure  $G = [q; w_1, \dots, w_n]$  where  $q$  is the quota, the players are  $N = \{1, \dots, n\}$ , the value  $w_i$  is the weight of player  $i$ , and the characteristic function of the game is defined as follows:

$$v(C) = \begin{cases} 1 & \text{if } \sum_{i \in C} w_i \geq q \\ 0 & \text{otherwise.} \end{cases}$$

Compute the players' Shapley values and describe the outcomes in the core in the following weighted voting games:

- $G = [13; 7, 7, 3, 3]$
- $G = [10; 3, 3, 3, 3, 1, 1]$

- First, consider a player  $i$  with weight 7; he is pivotal whenever he is preceded by any two players, or when he is preceded by the other player of weight 7. There are  $3! = 6$  permutations where  $i$  is third, and  $2! = 2$  more where he is preceded by the other player of weight 7. Thus, the Shapley value of the players of weight 7 is  $\frac{6+2}{4!} = \frac{1}{3}$ . By symmetry, both players of weight 7 have a Shapley value of  $\frac{1}{3}$ ; by efficiency, the Shapley value of the two players of weight 3 totals  $1 - 2 \times \frac{1}{3} = \frac{1}{3}$ . Since they are symmetric as well, the Shapley value of each of them is  $\frac{1}{6}$ .
- Consider a player of weight 1. he is pivotal if and only if he appears in position 4, after three players of weight 3. There are 4 ways to choose the player of weight 3 who appears after him,  $3! = 6$  ways to order the three players of weight 3 who appear before him, and 2 ways to order the two players (of weight 3 and 1) who appear after him, i.e.,  $4 \times 6 \times 2 = 48$  permutations. Thus, the Shapley value of each player of weight 1 is  $\frac{48}{6!} = \frac{1}{15}$ . By efficiency, the sum of Shapley values of the players of weight 3 is  $1 - \frac{2}{15} = \frac{13}{15}$ , so by symmetry the Shapley value of each of these players is  $\frac{13}{60}$ .

**Exercise 18.3.#CGOT**

Suppose we are given a cooperative game  $G = (\{1, 2\}, v)$  with characteristic function  $v$  defined by:

$$v(C) = \begin{cases} 1 & \text{if } C \text{ contains exactly one player} \\ 0 & \text{otherwise.} \end{cases}$$

Show that weighted voting games cannot capture this “singleton” game: we will not be able to find a quota  $q$  and weights  $w_i$  such that for all coalitions  $C$ ,  $\sum_{i \in C} w_i \geq q$  iff  $C$  contains exactly one player.

*Hint:* You can give a counterexample involving a game with just two players.

Suppose for sake of contradiction that there does exist a weighted voting game to represent the singleton game, and let  $q, w_1, w_2$  be the quota and respective weights. We must have:

$$\begin{aligned} 0 &< q \\ w_1 &\geq q \\ w_2 &\geq q \\ w_1 + w_2 &< q \end{aligned}$$

It immediately follows that  $w_1$  and  $w_2$  must be  $> 0$ .

Since  $w_1 \geq q$ , then  $w_1 + w_2 \geq q$ . Contradiction: no values  $w_1, w_2, q$  can satisfy these properties. (This is, incidentally, the same as the proof that perceptrons cannot capture XOR.)

**Exercise 18.3.#LOWG**

In the Landowner and Workers game there is a landowner  $\ell$  and  $n$  workers  $w_1, \dots, w_n$ . A group of workers may lease the land from the landowner and grow vegetables on it. Their productivity depends on the group size: a group of  $k$  workers can grow  $f(k)$  tons of vegetables, where  $f$  is an increasing function of  $k$ ,  $f(0) = 0$ . The characteristic function of this game is then given by  $v(C) = f(|C| - 1)$  if  $\ell \in C$  and  $v(C)$  otherwise. Compute the Shapley values of all players in this game.

When the landowner is in position  $k$ ,  $k = 1, \dots, n + 1$ , his predecessors have a value of 0 without him, and with him they have a value of  $f(k - 1)$ . This means that the landowner contributes  $f(k - 1)$  to all permutations in which he is in position  $k$ . There are exactly  $n!$  such permutations, so the total contribution to the Shapley value of the landowner from the permutations in which he is in position  $k$  is  $\frac{n!}{(n+1)!} f(k - 1) = \frac{f(k-1)}{n+1}$ . Thus, in total, the Shapley value of the owner is simply  $\frac{1}{n+1} \sum_{k=1}^{n+1} f(k - 1)$ , or the average value of  $f$  over the points  $0, \dots, n$ . Let us denote by  $\varphi_w$  the Shapley value of a worker and by  $\varphi_L$  the value of the landowner; all workers are symmetric so they all must have the same value. By efficiency,  $n\varphi_w + \varphi_L = f(n)$ , so  $\varphi_w = \frac{f(n) - \varphi_L}{n}$ , which equals  $\frac{f(n) - \frac{1}{n+1} \sum_{k=0}^n f(k)}{n}$ .

## 18.4 Making Collective Decisions

### Exercise 18.4.#DAUC

A Dutch auction is similar in an English auction, but rather than starting the bidding at a low price and increasing, in a Dutch auction the seller starts at a high price and gradually lowers the price until some buyer is willing to accept that price. (If multiple bidders accept the price, one is arbitrarily chosen as the winner.) More formally, the seller begins with a price  $p$  and gradually lowers  $p$  by increments of  $d$  until at least one buyer accepts the price. Assuming all bidders act rationally, is it true that for arbitrarily small  $d$ , a Dutch auction will always result in the bidder with the highest value for the item obtaining the item? If so, show mathematically why. If not, explain how it may be possible for the bidder with highest value for the item not to obtain it.

Independent private values

This question really has two answers, depending on what assumption is made about the probability distribution over bidder's private valuations  $v_i$  for the item.

In a Dutch auction, just as in a first-price sealed-bid auction, bidders must estimate the likely private values of the other bidders. When the price is higher than  $v_i$ , agent  $i$  will not bid, but as soon as the price reaches  $v_i$ , he faces a dilemma: bid now and win the item at a higher price than necessary, or wait and risk losing the item to another bidder. In the standard models of auction theory, each bidder, in addition to a private value  $v_i$ , has a probability density  $p_i(v_1, \dots, v_n)$  over the private values of all  $n$  bidders for the item. In particular, we consider **independent private values**, so that the distribution over the other bidders' values is independent of  $v_i$ . Each bidder will choose a bid—i.e., the first price at which they will bid if that price is reached—through a bidding function  $b_i(v_i)$ .

We are interested in finding a Nash equilibrium (technically a **Bayes–Nash equilibrium**) in which each bidder's bidding function is optimal given the bidding functions of the other agents. Under risk-neutrality, optimality of a bid  $b$  is determined by the expected payoff, i.e., the probability of winning the auction with bid  $b$  times the profit when paying that amount for the item. Now, agent  $i$  wins the auction with bid  $b$  if all the other bids are less than  $b$ ; let the probability of this happening be  $W_i(b)$  for whatever fixed bidding functions the other bidders use. ( $W_i(b)$  is thus a cumulative probability distribution and nondecreasing in  $b$ ; under independent private values, it does not depend on  $v_i$ .) Then we can write the expected payoff for agent  $i$  as

$$Q_i(v_i, b) = W_i(b)(v_i - b)$$

and the optimality condition in equilibrium is therefore

$$\forall i, b \quad W_i(b_i(v_i))(v_i - b_i(v_i)) \geq W_i(b)(v_i - b). \quad (18.1)$$

We now prove that the bidding functions  $b_i(v_i)$  must be **monotonic**, i.e., *nondecreasing* in the private valuation  $v_i$ . Let  $v$  and  $v'$  be two different valuations, with  $b = b_i(v)$  and  $b' = b_i(v')$ . Applying Equation (18.1) twice, first to say that  $(v, b)$  is better than  $(v, b')$  and then to say

that  $(v', b')$  is better than  $(v', b)$ , we obtain

$$\begin{aligned} W_i(b)(v - b) &\geq W_i(b')(v - b') \\ W_i(b')(v' - b') &\geq W_i(b)(v' - b) \end{aligned}$$

Rearranging, these become

$$\begin{aligned} v(W_i(b) - W_i(b')) &\geq W_i(b)b - W_i(b')b' \\ v'(W_i(b') - W_i(b)) &\geq W_i(b')b' - W_i(b)b \end{aligned}$$

Adding these equations, we have

$$(v' - v)(W_i(b') - W_i(b)) \geq 0$$

from which it follows that if  $v' > v$ , then  $W_i(b') \geq W_i(b)$ . Monotonicity does not follow immediately, however; we have to handle two cases:

- If  $W_i(b') > W_i(b)$ , or if  $W_i$  is strictly increasing, then  $b' \geq b$  and  $b_i(\cdot)$  is monotonic.
- Otherwise,  $W_i(b') = W_i(b)$  and  $W_i$  is flat between  $b$  and  $b'$ . Now if  $W_i$  is flat in any interval  $[x, y]$ , then an optimal bidding function will prefer  $x$  over any other bid in the interval since that maximizes the profit on winning without affecting the probability of winning; hence, we must have  $b' = b$  and again  $b_i(\cdot)$  is monotonic.

Intuitively, the proof amounts to the following: if a higher valuation could result in a lower bid, then by swapping the two bids the agent could increase the *sum* of the payoffs for the two bids, which means that *at least one* of the two original bids is suboptimal.

Returning to the question of efficiency—the property that the item goes to the bidder with the highest valuation—we see that it follows immediately from monotonicity in the case where the bidders’ prior distributions over valuations are **symmetric** or identically distributed.\*

Vickrey (1961) proves that the auction is *not* efficient in the asymmetric case where one player’s distribution is uniform over  $[0, 1]$  and the other’s is uniform over  $[a, b]$  for  $a > 0$ . Milgrom (1989) provides another, more transparent example of inefficiency: Suppose Alice has a known, fixed value of \$101 for an item, while Bob’s value is \$50 with probability 0.8 and \$75 with probability 0.2. Given that Bob will never bid higher than his valuation, Alice can see that a bid of \$51 will win *at least* 80% of the time, giving an expected profit of *at least*  $0.8 \times (\$101 - \$51) = \$40$ . On the other hand, any bid of \$62 or more cannot yield an expected profit at most \$39, regardless of Bob’s bid, and so is dominated by the bid of \$51. Hence, in any equilibrium, Alice’s bid at most \$61. Knowing this, Bob can bid \$62 whenever his valuation is \$75 and be sure of winning. Thus, with 20% probability, the item goes to Bob, whose valuation for it is lower than Alice’s. This violates efficiency.

Besides efficiency in the symmetric case, monotonicity has another important consequence for the analysis of the Dutch (and first-price) auction : it makes it possible to derive

---

\* Vickrey (1961) proved that under this assumption, the Dutch auction is efficient. Vickrey’s argument in Appendix III for the monotonicity of the bidding function is similar to the argument above but, as written, seems to apply only to the uniform-distribution case he was considering. Indeed, much of his analysis beginning with Appendix II is based on an inverse bidding function, which implicitly *assumes* monotonicity of the bidding function. Many other authors also begin by assuming monotonicity, then derive the form of the optimal bidding function, and then show it is monotonic. This proves the existence of an equilibrium with monotonic bidding functions, but not that all equilibria have this property.

the exact form of the bidding function. As it stands, Equation (18.1) is difficult or impossible to solve because the cumulative distribution of the other bidders' bids,  $W_i(b)$ , depends on their bidding functions, so all the bidding functions are coupled together. (Note the similarity to the Bellman equations for an MDP.) With monotonicity, however, we can define  $W_i$  in terms of the known valuation distributions. Assuming independence and symmetry, and writing  $b_i^{-1}(b)$  for the inverse of the (monotonic) bidding function, we have

$$Q_i(v_i, b) = (P(b_i^{-1}(b)))^{n-1}(v_i - b)$$

where  $P(v)$  is the probability that an individual valuation is less than  $v$ . At equilibrium, where  $b$  maximizes  $Q_i$ , the first derivative must be zero:

$$\frac{\partial Q}{\partial b} = 0 = \frac{(n-1)(P(b_i^{-1}(b)))^{n-2}p(b_i^{-1}(b))(v_i - b)}{b'_i(b_i^{-1}(b))} - (P(b_i^{-1}(b)))^{n-1}$$

where we have used the fact that  $df^{-1}(x)/dx = 1/f'(f^{-1}(x))$ .

For an equilibrium bidding function, of course,  $b_i^{-1}(b) = v_i$ ; substituting this and simplifying, we find the following differential equation for  $b_i$ :

$$b'_i(v_i) = (v_i - b_i(v_i)) \cdot (n-1)p(v_i)/P(v_i).$$

To find concrete solutions we also need to establish a boundary condition. Suppose  $v_0$  is the lowest possible valuation for the item; then we must have  $b_i(v_0) = v_0$  (Milgrom and Weber, 1982). Then the solution, as shown by McAfee and McMillan (1987), is

$$b_i(v_i) = v_i - \frac{\int_{v_0}^{v_i} (P(v))^{n-1} dv}{(P(v_i))^{n-1}}.$$

For example, suppose  $p$  is uniform in  $[0, 1]$ ; then  $P(v) = v$  and  $b_i(v_i) = v_i \cdot (n-1)/n$ , which is the classical result obtained by Vickrey (1961).

### Exercise 18.4.#AAUC

Imagine an auction mechanism that is just like an ascending-bid auction, except that at the end, the winning bidder, the one who bid  $b_{max}$ , pays only  $b_{max}/2$  rather than  $b_{max}$ . Assuming all agents are rational, what is the expected revenue to the auctioneer for this mechanism, compared with a standard ascending-bid auction?

In such an auction it is rational to continue bidding as long as winning the item would yield a profit, i.e., one is willing to bid up to  $2v_i$ . The auction will end at  $2v_o + d$ , so the winner will pay  $v_o + d/2$ , slightly less than in the regular version.

### Exercise 18.4.#NHLT

Teams in the National Hockey League historically received 2 points for winning a game and 0 for losing. If the game is tied, an overtime period is played; if nobody wins in overtime,

the game is a tie and each team gets 1 point. But league officials felt that teams were playing too conservatively in overtime (to avoid a loss), and it would be more exciting if overtime produced a winner. So in 1999 the officials experimented in mechanism design: the rules were changed, giving a team that loses in overtime 1 point, not 0. It is still 2 points for a win and 1 for a tie.

- a. Was hockey a zero-sum game before the rule change? After?
- b. Suppose that at a certain time  $t$  in a game, the home team has probability  $p$  of winning in regulation time, probability  $0.78 - p$  of losing, and probability 0.22 of going into overtime, where they have probability  $q$  of winning,  $.9 - q$  of losing, and .1 of tying. Give equations for the expected value for the home and visiting teams.
- c. Imagine that it were legal and ethical for the two teams to enter into a pact where they agree that they will skate to a tie in regulation time, and then both try in earnest to win in overtime. Under what conditions, in terms of  $p$  and  $q$ , would it be rational for both teams to agree to this pact?
- d. Longley and Sankaran (2005) report that since the rule change, the percentage of games with a winner in overtime went up 18.2%, as desired, but the percentage of overtime games also went up 3.6%. What does that suggest about possible collusion or conservative play after the rule change?

Every game is either a win for one side (and a loss for the other) or a tie. With 2 for a win, 1 for a tie, and 0 for a loss, 2 points are awarded for every game, so this is a constant-sum game.

If 1 point is awarded for a loss in overtime, then for some games 3 points are awarded in all. Therefore, the game is no longer constant-sum.

Suppose we assume that team A has probability  $r$  of winning in regular time and team B has probability  $s$  of winning in regular time (assuming normal play). Furthermore, assume team B has a probability  $q$  of winning in overtime (which occurs if there is a tie after regular time). Once overtime is reached (by any means), the expected utilities are as follows:

$$U_A^O = 1 + p$$

$$U_B^O = 1 + q$$

In normal play, the expected utilities are derived from the probability of winning plus the probability of tying times the expected utility of overtime play:

$$U_A = 2r + (1 - r - s)(1 + p)$$

$$U_B = 2s + (1 - r - s)(1 + q)$$

Hence A has an incentive to agree if  $U_A^O > U_A$ , or

$$1 + p > 2r + (1 - r - s)(1 + p) \quad \text{or} \quad rp - r + sp + s > 0 \quad \text{or} \quad p > \frac{r - s}{r + s}$$

and B has an incentive to agree if  $U_B^O > U_B$ , or

$$1 + q > 2s + (1 - r - s)(1 + q) \quad \text{or} \quad sq - s + rq + r > 0 \quad \text{or} \quad q > \frac{s - r}{r + s}$$

When both of these inequalities hold, there is an incentive to tie in regulation play. For any values of  $r$  and  $s$ , there will be values of  $p$  and  $q$  such that both inequalities hold.

For an in-depth statistical analysis of the actual effects of the rule change and a more sophisticated treatment of the utility functions, see “Overtime! Rules and Incentives in the National Hockey League” by Stephen T. Easton and Duane W. Rockerbie, available at [people.uleth.ca/~rockerbie/OVERTIME.PDF](http://people.uleth.ca/~rockerbie/OVERTIME.PDF).

# EXERCISES 19

## LEARNING FROM EXAMPLES

### 19.1 Forms of Learning

#### Exercise 19.1.#LLAN

Consider the problem faced by an infant learning to speak and understand a language. Explain how this process fits into the general learning model. Describe the percepts and actions of the infant, and the types of learning the infant must do. Describe the subfunctions the infant is trying to learn in terms of inputs and outputs, and available example data.

The aim here is to couch language learning in the framework of the chapter, not to solve the problem! This is a very interesting topic for class discussion, raising issues of nature vs. nurture, the indeterminacy of meaning and reference, and so on.

The first step is to appreciate the variety of knowledge that goes under the heading “language.” The infant must learn to recognize and produce speech, learn vocabulary, learn grammar, learn the semantic and pragmatic interpretation of a speech act, and learn strategies for disambiguation, among other things. The performance elements for this (in humans) and their associated learning mechanisms are obviously very complex and as yet little is known about them.

A naive model of the learning environment considers just the exchange of speech sounds. In reality, the physical context of each utterance is crucial: a child must see the context in which “watermelon” is uttered in order to learn to associate “watermelon” with watermelons. Thus, the environment consists not just of other humans but also the physical objects and events about which discourse takes place. Auditory sensors detect speech sounds, while other senses (primarily visual) provide information on the physical context. The relevant effectors are the speech organs and the motor capacities that allow the infant to respond to speech or that elicit verbal feedback.

The performance standard could simply be the infant’s general utility function, however that is realized, so that the infant performs reinforcement learning to perform and respond to speech acts so as to improve its well-being—for example, by obtaining food and attention. However, humans’ built-in capacity for mimicry suggests that the production of sounds similar to those produced by other humans is a goal in itself. The child (once he or she learns to differentiate sounds and learn about pointing or other means of indicating salient objects) is also exposed to examples of supervised learning: an adult says “shoe” or “belly button” while indicating the appropriate object. So sentences produced by adults provide labelled positive examples, and the response of adults to the infant’s speech acts provides further classification feedback.

Mostly, it seems that adults do not correct the child's speech, so there are very few negative classifications of the child's attempted sentences. This is significant because early work on language learning, such as the work of Gold (1967) concentrated just on identifying the set of strings that are grammatical, assuming a particular grammatical formalism. If there are only positive examples, then there is nothing to rule out the grammar  $S \rightarrow Word^*$ . Some theorists (notably Chomsky and Fodor) used what they call the "poverty of the stimulus" argument to say that the basic universal grammar of languages must be innate, because otherwise (given the lack of negative examples) there would be no way that a child could learn a language (under the assumptions of language learning as learning a set of grammatical strings). Critics have called this the "poverty of the imagination" argument—I can't think of a learning mechanism that would work, so it must be innate. Indeed, if we go to probabilistic context free grammars, then it is possible to learn a language without negative examples.

### Exercise 19.1.#LTEN

Repeat Exercise LLAN for the case of learning to play tennis (or some other sport with which you are familiar). Is this supervised learning or reinforcement learning?

Learning tennis is much simpler than learning to speak. The requisite skills can be divided into movement, playing strokes, and strategy. The environment consists of the court, ball, opponent, and one's own body. The relevant sensors are the visual system and proprioception (the sense of forces on and position of one's own body parts). The effectors are the muscles involved in moving to the ball and hitting the stroke. The learning process involves both supervised learning and reinforcement learning. Supervised learning occurs in acquiring the predictive transition models, e.g., where the opponent will hit the ball, where the ball will land, and what trajectory the ball will have after one's own stroke (e.g., if I hit a half-volley *this* way, it goes into the net, but if I hit it *that* way, it clears the net). Reinforcement learning occurs when points are won and lost—this is particularly important for strategic aspects of play such as shot placement and positioning (e.g., in 60% of the points where I hit a lob in response to a cross-court shot, I end up losing the point). In the early stages, reinforcement also occurs when a shot succeeds in clearing the net and landing in the opponent's court. Achieving this small success is itself a sequential process involving many motor control commands, and there is no teacher available to tell the learner's motor cortex which motor control commands to issue.

## 19.2 Supervised Learning

### Exercise 19.2.#SLDF

Define the following machine-learning terms in your own words

- a. Training set
- b. Hypothesis
- c. Bias

d. Variance

- a. A set of input–output pair examples, used as input to a machine learning program to create a hypothesis.
- b. In machine learning, a hypothesis is a function, learned from the training data and a member of the hypothesis space, that maps inputs to outputs.
- c. The amount by which the output of a hypothesis consistently varies from the true answer in a particular direction, regardless of the exact training data.
- d. The amount by which the output of a hypothesis randomly varies from the true answer, when trained on slightly different data sets.

#### Exercise 19.2.#SURL

Describe the differences between supervised, unsupervised, and reinforcement learning.

In supervised learning, the training data consists of input–output pairs, where the labeled outputs are what we are trying to learn. In unsupervised learning, there is no labeled output, and the goal is to find patterns or clusters in the input. In reinforcement learning, the learning is given positive or negative rewards at certain points, and the goal is to maximize rewards.

## 19.3 Learning Decision Trees

#### Exercise 19.3.#DECT

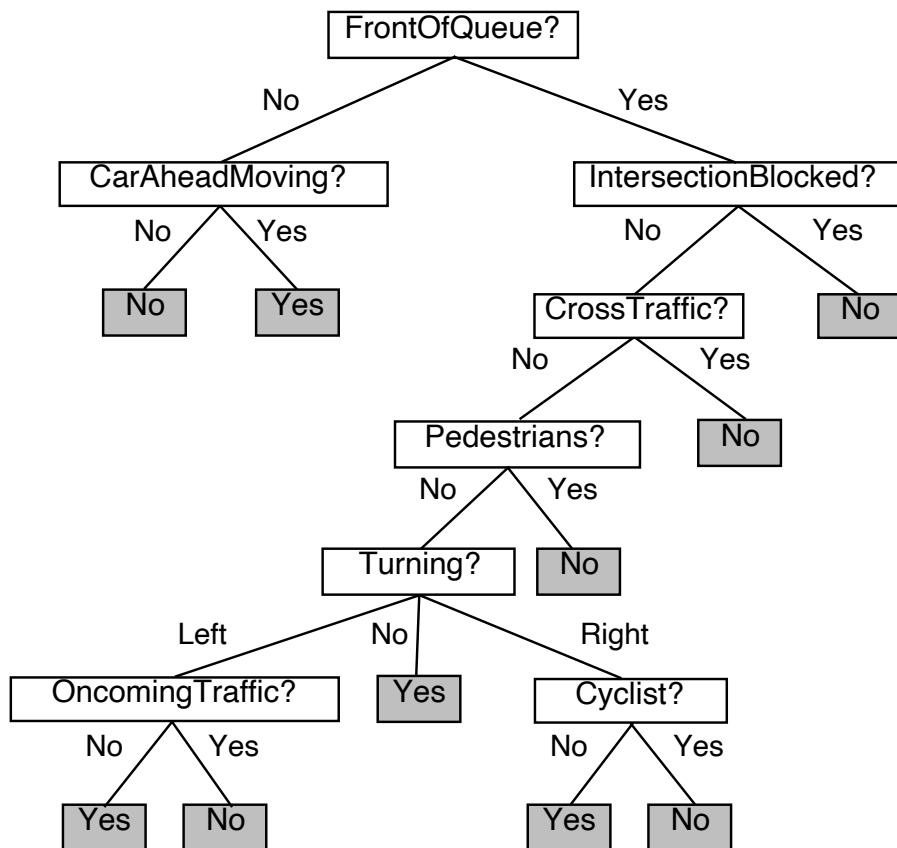
Draw a decision tree for the problem of deciding whether to move forward at a road intersection, given that the light has just turned green.

This is a deceptively simple question, designed to point out the plethora of “exceptions” in real-world situations and the way in which decision trees capture a hierarchy of exceptions. One possible tree is shown in Figure S19.1. One can, of course, imagine many more exceptions. The qualification problem, defined originally for action models, applies *a fortiori* to condition–action rules.

#### Exercise 19.3.#PATH

We never test the same attribute twice along one path in a decision tree. Why not?

In standard decision trees, attribute tests divide examples according to the attribute value. Therefore any example reaching the second test already has a known value for the attribute and the second test is redundant. In some decision tree systems, however, all tests are Boolean even if the attributes are multivalued or continuous. In this case, additional tests of the at-



**Figure S19.1** A decision tree for deciding whether to move forward at a traffic intersection, given a green light.

tribute can be used to check different values or subdivide the range further (e.g., first check if  $X > 0$ , and then if it is, check if  $x > 10$ ).

### Exercise 19.3.#TDEC

Suppose we generate a training set from a decision tree and then apply decision-tree learning to that training set. Is it the case that the learning algorithm will eventually return the correct tree as the training-set size goes to infinity? Why or why not?

The algorithm may not return the “correct” tree, but it will return a tree that is logically equivalent, assuming that the method for generating examples eventually generates all possible combinations of input attributes. This is true because any two decision tree defined on the same set of attributes that agree on all possible examples are, by definition, logically equivalent. The actually form of the tree may differ because there are many different ways to represent the same function. (For example, with two attributes  $A$  and  $B$  we can have one tree with  $A$  at the root and another with  $B$  at the root.) The root attribute of the original tree may

not in fact be the one that will be chosen by the information gain heuristic when applied to the training examples.

### Exercise 19.3.#LEAF

In the recursive construction of decision trees, it sometimes happens that a mixed set of positive and negative examples remains at a leaf node, even after all the attributes have been used. Suppose that we have  $p$  positive examples and  $n$  negative examples.

- Show that the solution used by DECISION-TREE-LEARNING, which picks the majority classification, minimizes the absolute error over the set of examples at the leaf.
- Show that the **class probability**  $p/(p + n)$  minimizes the sum of squared errors.

This question brings a little bit of mathematics to bear on the analysis of the learning problem, preparing the ground for Chapter 20. Error minimization is a basic technique in both statistics and neural nets. The main thing is to see that the error on a given training set can be written as a mathematical expression and viewed as a function of the hypothesis chosen. Here, the hypothesis in question is a single number  $\alpha \in [0, 1]$  returned at the leaf.

- If  $\alpha$  is returned, the absolute error is

$$\begin{aligned} E = p(1 - \alpha) + n\alpha &= \alpha(n - p) + p = n \text{ when } \alpha = 1 \\ &= p \text{ when } \alpha = 0 \end{aligned}$$

This is minimized by setting

$$\begin{aligned} \alpha &= 1 \text{ if } p > n \\ \alpha &= 0 \text{ if } p < n \end{aligned}$$

That is,  $\alpha$  is the majority value.

- First calculate the sum of squared errors, and its derivative:

$$\begin{aligned} E &= p(1 - \alpha)^2 + n\alpha^2 \\ \frac{dE}{d\alpha} &= 2\alpha n - 2p(1 - \alpha) = 2\alpha(p + n) - 2p \end{aligned}$$

The fact that the second derivative,  $\frac{d^2E}{d\alpha^2} = 2(p + n)$ , is greater than zero means that  $E$  is minimized (not maximized) where  $\frac{dE}{d\alpha} = 0$ , i.e., when  $\alpha = \frac{p}{p+n}$ .

### Exercise 19.3.#NNGA

Suppose that an attribute splits the set of examples  $E$  into subsets  $E_k$  and that each subset has  $p_k$  positive examples and  $n_k$  negative examples. Show that the attribute has strictly positive information gain unless the ratio  $p_k/(p_k + n_k)$  is the same for all  $k$ .

This result emphasizes the fact that any statistical fluctuations caused by the random sampling process will result in an apparent information gain.

The easy part is showing that the gain is zero when each subset has the same ratio of positive examples. The gain is defined as

$$B\left(\frac{p}{p+n}\right) - \sum_{k=1}^d \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

Since  $p = \sum p_k$  and  $n = \sum n_k$ , if  $p_k/(p_k + n_k)$  is the same for all  $k$  we must have  $p_k/(p_k + n_k) = p/(p+n)$  for all  $k$ . From this, we obtain

$$\begin{aligned} Gain &= B\left(\frac{p}{p+n}\right) - B\left(\frac{p}{p+n}\right) \frac{1}{p+n} \sum_{k=1}^d p_k + n_k \\ &= B\left(\frac{p}{p+n}\right) - B\left(\frac{p}{p+n}\right) \frac{1}{p+n}(p+n) = 0 \end{aligned}$$

Note that this holds for all values of  $p_k + n_k$ . To prove that the value is positive elsewhere, we can apply the method of Lagrange multipliers to show that this is the only stationary point; the gain is clearly positive at the extreme values, so it is positive everywhere but the stationary point. In detail, we have constraints  $\sum_k p_k = p$  and  $\sum_k n_k = n$ , and the Lagrange function is

$$\Lambda = B\left(\frac{p}{p+n}\right) - \sum_k \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right) + \lambda_1 \left(p - \sum_k p_k\right) + \lambda_2 \left(n - \sum_k n_k\right).$$

Setting its derivatives to zero, we obtain, for each  $k$ ,

$$\begin{aligned} \frac{\partial \Lambda}{\partial p_k} &= -\frac{1}{p+n} B\left(\frac{p_k}{p_k + n_k}\right) - \frac{p_k + n_k}{p+n} \log \frac{p_k}{n_k} \left( \frac{1}{p_k + n_k} - \frac{p_k}{(p_k + n_k)^2} \right) - \lambda_1 = 0 \\ \frac{\partial \Lambda}{\partial n_k} &= -\frac{1}{p+n} B\left(\frac{p_k}{p_k + n_k}\right) - \frac{p_k + n_k}{p+n} \log \frac{p_k}{n_k} \left( \frac{-p_k}{(p_k + n_k)^2} \right) - \lambda_2 = 0. \end{aligned}$$

Subtracting these two, we obtain  $\log(p_k/n_k) = (p+n)(\lambda_2 - \lambda_1)$  for all  $k$ , implying that at any stationary point the ratios  $p_k/n_k$  must be the same for all  $k$ . Given the two summation constraints, the only solution is the one given in the question.

### Exercise 19.3.#TBIN

Consider the following data set comprised of three binary input attributes ( $A_1$ ,  $A_2$ , and  $A_3$ ) and one binary output:

| Example | $A_1$ | $A_2$ | $A_3$ | Output $y$ |
|---------|-------|-------|-------|------------|
| $x_1$   | 1     | 0     | 0     | 0          |
| $x_2$   | 1     | 0     | 1     | 0          |
| $x_3$   | 0     | 1     | 0     | 0          |
| $x_4$   | 1     | 1     | 1     | 1          |
| $x_5$   | 1     | 1     | 0     | 1          |

Use the algorithm in Figure 19.5 (page 660) to learn a decision tree for these data. Show the computations made to determine the attribute to split at each node.

Note that to compute each split, we need to compute  $\text{Remainder}(A_i)$  for each attribute  $A_i$ , and select the attribute that provides the minimal remaining information, since the existing information prior to the split is the same for all attributes we may choose to split on.

Computations for first split: remainders for  $A_1$ ,  $A_2$ , and  $A_3$  are

$$(4/5)(-2/4 \log(2/4) - 2/4 \log(2/4)) + (1/5)(-0 - 1/1 \log(1/1)) = 0.800$$

$$(3/5)(-2/3 \log(2/3) - 1/3 \log(1/3)) + (2/5)(-0 - 2/2 \log(2/2)) \approx 0.551$$

$$(2/5)(-1/2 \log(1/2) - 1/2 \log(1/2)) + (3/5)(-1/3 \log(1/3) - 2/3 \log(2/3)) \approx 0.951$$

Choose  $A_2$  for first split since it minimizes the remaining information needed to classify all examples. Note that all examples with  $A_2 = 0$ , are correctly classified as  $B = 0$ . So we only need to consider the three remaining examples  $(x_3, x_4, x_5)$  for which  $A_2 = 1$ .

After splitting on  $A_2$ , we compute the remaining information for the other two attributes on the three remaining examples  $(x_3, x_4, x_5)$  that have  $A_2 = 1$ . The remainders for  $A_1$  and  $A_3$  are

$$(2/3)(-2/2 \log(2/2) - 0) + (1/3)(-0 - 1/1 \log(1/1)) = 0$$

$$(1/3)(-1/1 \log(1/1) - 0) + (2/3)(-1/2 \log(1/2) - 1/2 \log(1/2)) \approx 0.667.$$

So, we select attribute  $A_1$  to split on, which correctly classifies all remaining examples.

### Exercise 19.3.#NMST

Construct a data set (set of examples with attributes and classifications) that would cause the decision-tree learning algorithm to find a non-minimal-sized tree. Show the tree constructed by the algorithm and the minimal-sized tree that you can generate by hand.

The decision-tree algorithm is greedy: it selects the single attribute which give the most information about the correct classification. If there are two attributes which individually give little information, but jointly give more, the algorithm will not notice this.

| $x_1$ | $x_2$ | $x_3$ | $y$ |
|-------|-------|-------|-----|
| A     | 0     | 0     | -   |
| A     | 1     | 0     | +   |
| A     | 0     | 1     | +   |

As an example problem, consider the following training examples:

|   |   |   |   |
|---|---|---|---|
| A | 1 | 1 | - |
| B | 0 | 0 | - |
| C | 1 | 0 | + |
| C | 0 | 1 | + |
| B | 1 | 1 | - |

A three node tree observing the value of  $x_2$  then the value of  $x_3$  suffices to classify all examples correctly.

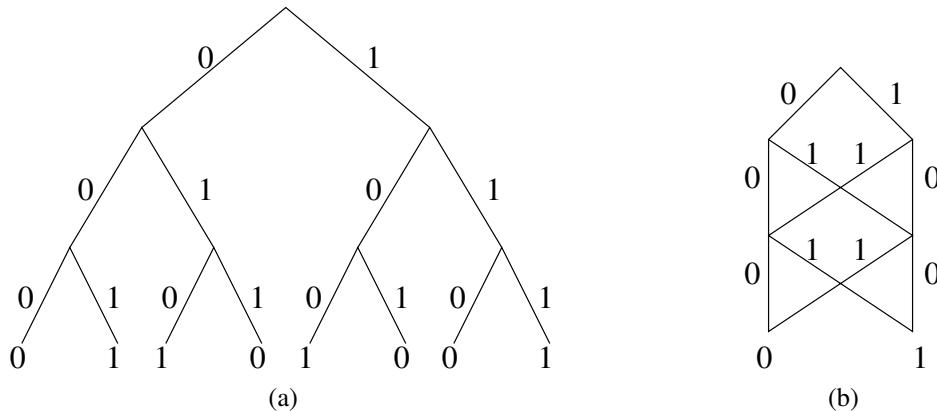
Individually, attributes  $x_2$  and  $x_3$  divide the examples equally between positive and negative classifications: their gain is exactly zero. Attribute  $x_1$  divides the examples into three groups, two of which have only a single classification: its gain is  $1/2$  bit. The decision-tree learning algorithm will therefore pick  $x_1$  and build a four-node tree, and if  $x_1 = A$  we need to observe both  $x_2$  and  $x_3$  to correctly classify the examples.

**Exercise 19.3.#DGRA**

A decision *graph* is a generalization of a decision tree that allows nodes (i.e., attributes used for splits) to have multiple parents, rather than just a single parent. The resulting graph must still be acyclic. Now, consider the XOR function of *three* binary input attributes, which produces the value 1 if and only if an odd number of the three input attributes has value 1.

- Draw a minimal-sized decision *tree* for the three-input XOR function.
- Draw a minimal-sized decision *graph* for the three-input XOR function.

See Figure S19.2, where nodes on successive rows measure attributes  $A_1$ ,  $A_2$ , and  $A_3$ . (Any fixed ordering works.)



**Figure S19.2** XOR function representations: (a) decision tree, and (b) decision graph.

**Exercise 19.3.#PRUN**

This exercise considers  $\chi^2$  pruning of decision trees (Section 19.3.4).

- Create a data set with two input attributes, such that the information gain at the root of the tree for both attributes is zero, but there is a decision tree of depth 2 that is consistent with all the data. What would  $\chi^2$  pruning do on this data set if applied bottom up? If applied top down?
- Modify DECISION-TREE-LEARNING to include  $\chi^2$ -pruning. You might wish to consult Quinlan (1986) for details.

This is a fairly small, straightforward programming exercise. The only hard part is the actual  $\chi^2$  computation; you might want to provide your students with a library function to do this.

**Exercise 19.3.#MISS**

The standard DECISION-TREE-LEARNING algorithm described in the chapter does not handle cases in which some examples have missing attribute values.

- a. First, we need to find a way to classify such examples, given a decision tree that includes tests on the attributes for which values can be missing. Suppose that an example  $\mathbf{x}$  has a missing value for attribute  $A$  and that the decision tree tests for  $A$  at a node that  $\mathbf{x}$  reaches. One way to handle this case is to pretend that the example has *all* possible values for the attribute, but to weight each value according to its frequency among all of the examples that reach that node in the decision tree. The classification algorithm should follow all branches at any node for which a value is missing and should multiply the weights along each path. Write a modified classification algorithm for decision trees that has this behavior.
- b. Now modify the information-gain calculation so that in any given collection of examples  $C$  at a given node in the tree during the construction process, the examples with missing values for any of the remaining attributes are given “as-if” values according to the frequencies of those values in the set  $C$ .

This is another straightforward programming exercise. The follow-up exercise is to run tests to see if the modified algorithm actually does better.

**Exercise 19.3.#GAIN**

In Section 19.3.5, we noted that attributes with many different possible values can cause problems with the gain measure. Such attributes tend to split the examples into numerous small classes or even singleton classes, thereby appearing to be highly relevant according to the gain measure. The **gain-ratio** criterion selects attributes according to the ratio between their gain and their intrinsic information content—that is, the amount of information contained in the answer to the question, “What is the value of this attribute?” The gain-ratio criterion therefore tries to measure how efficiently an attribute provides information on the correct classification of an example. Write a mathematical expression for the information content of an attribute, and implement the gain ratio criterion in DECISION-TREE-LEARNING.

Let the prior probabilities of each attribute value be  $P(v_1), \dots, P(v_n)$ . (These probabilities are estimated by the empirical fractions among the examples at the current node.) From page 540, the intrinsic information content of the attribute is

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log v_i$$

Given this formula and the empirical estimates of  $P(v_i)$ , the modification to the code is straightforward.

**Exercise 19.3.#DLCL**

Construct a *decision list* to classify the data below. Select tests to be as small as possible (in terms of attributes), breaking ties among tests with the same number of attributes by selecting the one that classifies the greatest number of examples correctly. If multiple tests have the same number of attributes and classify the same number of examples, then break the tie using attributes with lower index numbers (e.g., select  $A_1$  over  $A_2$ ).

| Example | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $y$ |
|---------|-------|-------|-------|-------|-----|
| $x_1$   | 1     | 0     | 0     | 0     | 1   |
| $x_2$   | 1     | 0     | 1     | 1     | 1   |
| $x_3$   | 0     | 1     | 0     | 0     | 1   |
| $x_4$   | 0     | 1     | 1     | 0     | 0   |
| $x_5$   | 1     | 1     | 0     | 1     | 1   |
| $x_6$   | 0     | 1     | 0     | 1     | 0   |
| $x_7$   | 0     | 0     | 1     | 1     | 1   |
| $x_8$   | 0     | 0     | 1     | 0     | 0   |

| Test                     | If yes | If no     |
|--------------------------|--------|-----------|
| $A_1 = 1$                | 1      | next test |
| $A_3 = 1 \wedge A_4 = 0$ | 0      | next test |
| $A_2 = 0$                | 0      | 1         |

## 19.4 Model Selection and Optimization

**Exercise 19.4.#BCLA**

Suppose you are running a learning experiment on a new algorithm for Boolean classification. You have a data set consisting of 100 positive and 100 negative examples. You plan to use leave-one-out cross-validation and compare your algorithm to a baseline function, a simple majority classifier. (A majority classifier is given a set of training data and then always outputs the class that is in the majority in the training set, regardless of the input.) You expect the majority classifier to score about 50% on leave-one-out cross-validation, but to your surprise, it scores zero every time. Can you explain why?

If we leave out an example of one class, then the majority of the remaining examples are of the other class, so the majority classifier will always predict the wrong answer.

**Exercise 19.4.#CLAR**

Suppose that a learning algorithm is trying to find a consistent hypothesis when the classifications of examples are actually random. There are  $n$  Boolean attributes, and examples are drawn uniformly from the set of  $2^n$  possible examples. Calculate the number of examples

required before the probability of finding a contradiction in the data reaches 0.5.

Suppose that we draw  $m$  examples. Each example has  $n$  input features plus its classification, so there are  $2^{n+1}$  distinct input/output examples to choose from. For each example, there is exactly one contradictory example, namely the example with the same input features but the opposite classification. Thus, the probability of finding *no* contradiction is

$$\begin{aligned}\frac{\text{number of sequences of } m \text{ non-contradictory examples}}{\text{number of sequences of } m \text{ examples}} &= \frac{2^{n+1} \cdot (2^{n+1} - 1) \dots (2^{n+1} - m + 1)}{2^{m(n+1)}} \\ &= \frac{2^{n+1}!}{(2^{n+1} - m)! 2^{m(n+1)}}\end{aligned}$$

For  $n = 10$ , with 2048 possible examples, a contradiction becomes likely with probability  $> 0.5$  after 54 examples have been drawn.

### Exercise 19.4.#HOVS

Suppose you train a classifier and test it on a held-out validation set. It gets 30% classification accuracy on the training set and 30% classification accuracy on the validation set.

- From what problem is your model most likely suffering: underfitting or overfitting?
  - What could reasonably be expected to improve your classifier's performance on the validation set: adding new features or removing some features? Justify your answer.
  - What could reasonably be expected to improve your classifier's performance on the validation set: collecting more training data or throwing out some training data? Justify your answer.
- 
- Underfitting.
  - Adding new features. Under the current feature representation, we are unable to accurately model the training data for the purpose of the classification task we're interested in. The classifier may be able to deduce more information about the connections between data points and their classes from additional features, allowing it to better model the data for the classification task. For example, a linear perceptron could not accurately model two classes separated by a circle in a 2-dimensional feature space, but by using quadratic features in a kernel perceptron, we can find a perfect separating hyperplane.
  - Collecting more training data. More training data can't hurt. However, given that training and hold-out validation data sets already achieve the same performance, it may be that the underlying problem is not a lack of training data.

## 19.5 The Theory of Learning

### Exercise 19.5.#EMBE

Consider the problem of separating  $N$  data points into positive and negative examples using a linear separator. Clearly, this can always be done for  $N = 2$  points on a line of dimension  $d = 1$ , regardless of how the points are labeled or where they are located (unless the points are in the same place).

- Show that it can always be done for  $N = 3$  points on a plane of dimension  $d = 2$ , unless they are collinear.
- Show that it cannot always be done for  $N = 4$  points on a plane of dimension  $d = 2$ .
- Show that it can always be done for  $N = 4$  points in a space of dimension  $d = 3$ , unless they are coplanar.
- Show that it cannot always be done for  $N = 5$  points in a space of dimension  $d = 3$ .
- The ambitious student may wish to prove that  $N$  points in general position (but not  $N + 1$ ) are linearly separable in a space of dimension  $N - 1$ .

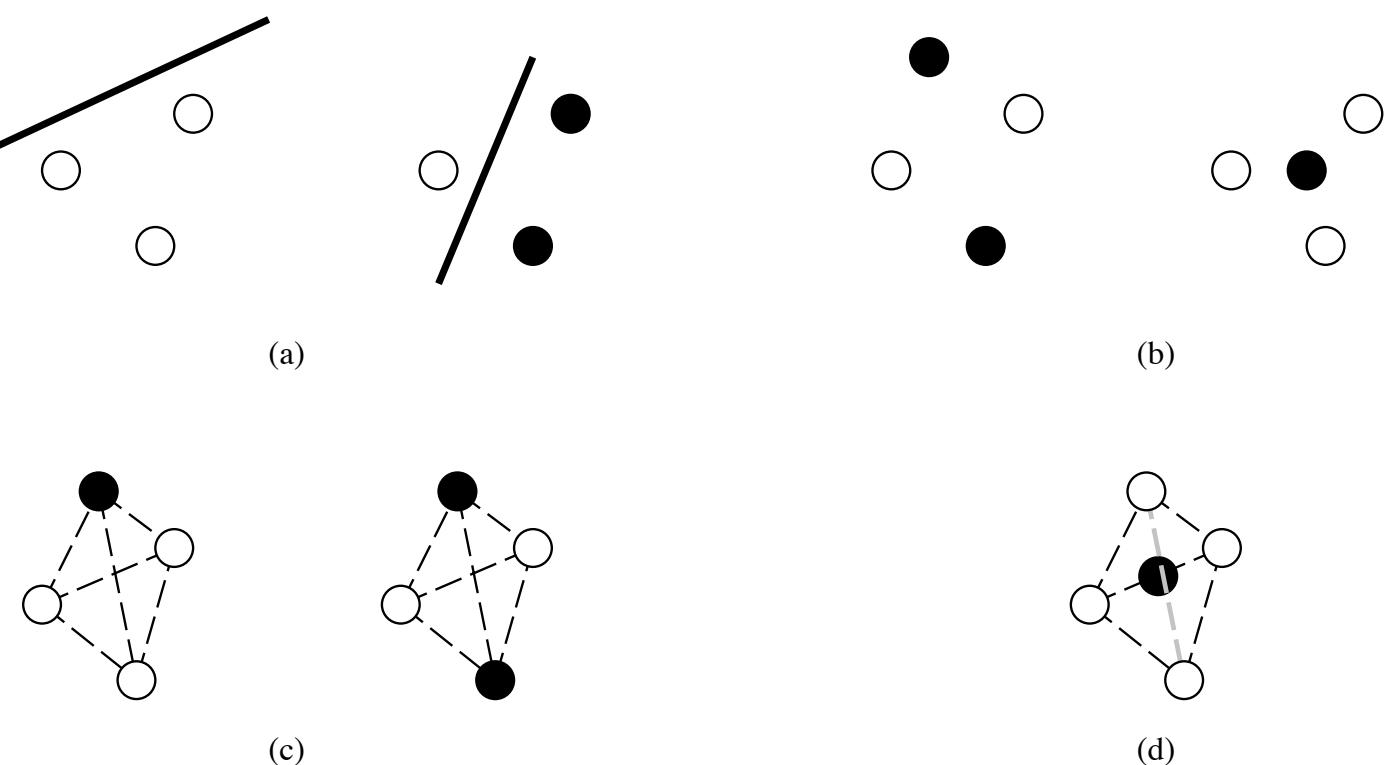
The main purpose of this exercise is to make concrete the notion of the *capacity* of a function class (in this case, linear halfspaces). It can be hard to internalize this concept, but the examples really help.

- Three points in general position on a plane form a triangle. Any subset of the points can be separated from the rest by a line, as can be seen from the two examples in Figure S19.3(a).
- Figure S19.3(b) shows two cases where the positive and negative examples cannot be separated by a line.
- Four points in general position on a plane form a tetrahedron. Any subset of the points can be separated from the rest by a plane, as can be seen from the two examples in Figure S19.3(c).
- Figure S19.3(d) shows a case where a negative point is inside the tetrahedron formed by four positive points; clearly no plane can separate the two sets.
- Proof omitted.

### Exercise 19.5.#DLPR

Prove that a decision list can represent the same function as a decision tree while using at most as many rules as there are leaves in the decision tree for that function. Give an example of a function represented by a decision list using strictly fewer rules than the number of leaves in a minimal-sized decision tree for that same function.

Proof (sketch): Each path from the root to a leaf in a decision tree represents a logical conjunction that results in a classification at the leaf node. We can simply create a decision list by producing one rule to correspond to each such path through the decision tree where the



**Figure S19.3** Illustrative examples for VC dimensions.

rule in the decision list has the test given by the logical conjunction in the path and the output for the rule is the corresponding classification at the leaf of the path. Thus we produce one rule for each leaf in the decision tree (since each leaf determines a unique path), constructing a decision list that captures the same function represented in the decision tree.

A simple example of a function that can be represented with strictly fewer rules in a decision list than the number of leaves in a minimal sized decision tree is the logical conjunction of two boolean attributes:  $A_1 \wedge A_2 \Rightarrow T$ .

The decision list has the form:

| Test                     | If yes | If no |
|--------------------------|--------|-------|
| $A_1 = T \wedge A_2 = T$ | T      | F     |

Note: one could consider this either one rule, or at most two rules if we were to represent it as follows:

| Test                     | If yes | If no     |
|--------------------------|--------|-----------|
| $A_1 = T \wedge A_2 = T$ | T      | next test |
| $T$                      | F      |           |

In either case, the corresponding decision tree has three leaves.

### Exercise 19 5 #DLEX

This exercise concerns the expressiveness of decision lists (Section 19.5).

- a. Show that decision lists can represent any Boolean function, if the size of the tests is

not limited.

- b.** Show that if the tests can contain at most  $k$  literals each, then decision lists can represent any function that can be represented by a decision tree of depth  $k$ .

Note: this is the only exercise to cover the material in section 18.6. Although the basic ideas of computational learning theory are both important and elegant, it is not easy to find good exercises that are suitable for an AI class as opposed to a theory class. If you are teaching a graduate class, or an undergraduate class with a strong emphasis on learning, it might be a good idea to use some of the exercises from Kearns and Vazirani (1994).

- a.** If each test is an arbitrary conjunction of literals, then a decision list can represent an arbitrary DNF (disjunctive normal form) formula directly. The DNF expression  $C_1 \vee C_2 \vee \dots \vee C_n$ , where  $C_i$  is a conjunction of literals, can be represented by a decision list in which  $C_i$  is the  $i$ th test and returns *True* if successful. That is:

$$\begin{aligned} C_1 &\rightarrow \text{True;} \\ C_2 &\rightarrow \text{True;} \\ \dots \\ C_n &\rightarrow \text{True;} \\ \text{True} &\rightarrow \text{False} \end{aligned}$$

Since any Boolean function can be written as a DNF formula, then any Boolean function can be represented by a decision list.

- b.** A decision tree of depth  $k$  can be translated into a decision list whose tests have at most  $k$  literals simply by encoding each path as a test. The test returns the corresponding leaf value if it succeeds. Since the decision tree has depth  $k$ , no path contains more than  $k$  literals.

## 19.6 Linear Regression and Classification

### Exercise 19.6.#PMLG

Section 19.6.5 (page 684) noted that the output of the logistic function could be interpreted as a *probability*  $p$  assigned by the model to the proposition that  $f(\mathbf{x}) = 1$ ; the probability that  $f(\mathbf{x}) = 0$  is therefore  $1 - p$ . Write down the probability  $p$  as a function of  $\mathbf{x}$  and calculate the derivative of  $\log p$  with respect to each weight  $w_i$ . Repeat the process for  $\log(1 - p)$ . These calculations give a learning rule for minimizing the negative-log-likelihood loss function for a probabilistic hypothesis. Comment on any resemblance to other learning rules in the chapter.

This question introduces some of the concepts that are studied in depth in Chapter 20; it could be used as an exercise for that chapter too, but is interesting to see at this stage also.

The logistic output is

$$p = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}} = \frac{1}{1 + e^{-\sum_j w_j x_j}}.$$

Taking the log and differentiating, we have

$$\begin{aligned}\log p &= -\log(1 + e^{-\mathbf{w} \cdot \mathbf{x}}) \\ \frac{\partial \log p}{\partial w_i} &= -\left(\frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}} \frac{\partial}{\partial w_i} (1 + e^{-\mathbf{w} \cdot \mathbf{x}})\right) \\ &= -p \cdot (-x_i) \cdot e^{-\mathbf{w} \cdot \mathbf{x}} = (1 - p)x_i.\end{aligned}$$

For a negative example, we have

$$\begin{aligned}\log(1 - p) &= -\log 1/(1 - p) = -\log(1 + e^{\mathbf{w} \cdot \mathbf{x}}) \\ \frac{\partial \log p}{\partial w_i} &= -\left(\frac{1}{1 + e^{\mathbf{w} \cdot \mathbf{x}}} \frac{\partial}{\partial w_i} (1 + e^{\mathbf{w} \cdot \mathbf{x}})\right) \\ &= -(1 - p) \cdot x_i \cdot e^{\mathbf{w} \cdot \mathbf{x}} = -(1 - p) \cdot x_i \cdot p/(1 - p) = -px_i.\end{aligned}$$

The loss function is  $L = -\log p$  for a positive example ( $y = 1$ ) and  $L = -\log(1 - p)$  for a negative example ( $y = 0$ ). We can write this as a single rule:

$$L = -\log p^y(1 - p)^{(1-y)} = -y \log p - (1 - y) \log(1 - p).$$

Using the above results, we obtain

$$\frac{\partial L}{\partial w_i} = -y(1 - p)x_i + (1 - y)px_i = -x_i(y - p) = -x_i(y - h_{\mathbf{w}}(\mathbf{x}))$$

which has the same form as the linear regression and perceptron learning rules.

### Exercise 19.6.#NBLC

Which of the following are true or false. Briefly justify your answer.

- a. In the case of a binary class and all binary features, Naive Bayes is a linear classifier.
  - b. Naive Bayes trained using maximum-likelihood parameter estimation is guaranteed not to perform worse if more features are added.
  - c. Naive Bayes trained using maximum-likelihood parameter estimation generally performs better on the training set if add- $k$  smoothing is used.
- 
- a. True.  $\arg \max_c \Pr(C = c | X_{1:n} = x_{1:n}) = \arg \max_c \log \Pr(C = c, X_{1:n} = x_{1:n})$ , and  $\log \Pr(C = c, X_{1:n} = x_{1:n})$  is linear in  $x_{1:n}$
  - b. False. If additional dependent features are added, accuracy may be worse. Independent features should help.
  - c. False. Add- $k$  smoothing typically hurts accuracy on the training set, if anything.

## 19.7 Nonparametric Models

### Exercise 19.7.#KNNM

Suppose a 7-nearest-neighbors regression search returns  $\{7, 6, 8, 4, 7, 11, 100\}$  as the 7 nearest  $y$  values for a given  $x$  value. What is the value of  $\hat{y}$  that minimizes the  $L_1$  loss function on this data? There is a common name in statistics for this value as a function of the  $y$  values; what is it? Answer the same two questions for the  $L_2$  loss function.

The  $L_1$  loss is minimized by the median, in this case 7, and the  $L_2$  loss by the mean, in this case 143/7.

For the first, suppose we have an odd number  $2n + 1$  of elements  $y_{-n} < \dots < y_0 < \dots < y_n$ . For  $n = 0$ ,  $\hat{y} = y_0$  is the median and minimizes the loss. Then, observe that the  $L_1$  loss for  $n + 1$  is

$$\frac{1}{2n+3} \sum_{i=-(n+1)}^{n+1} |\hat{y} - y_i| = \frac{1}{2n+3} (|\hat{y} - y_{n+1}| + |\hat{y} - y_{-(n+1)}|) + \frac{1}{2n+3} \sum_{i=-n}^n |\hat{y} - y_i|$$

The first term is equal to  $|y_{n+1} - y_{-(n+1)}|$  whenever  $y_{n+1} \leq \hat{y} \leq y_{-(n+1)}$ , e.g. for  $\hat{y} = y_0$ , and is strictly larger otherwise. But by inductive hypothesis the second term also is minimized by  $\hat{y} = y_0$ , the median.

For the second, notice that as the  $L_2$  loss of  $\hat{y}$  given data  $y_1, \dots, y_n$

$$\frac{1}{n} \sum_i (\hat{y} - y_i)^2$$

is differentiable we can find critical points:

$$0 = \frac{2}{n} \sum_i (\hat{y} - y_i)$$

or  $\hat{y} = (1/n) \sum_i y_i$ . Taking the second derivative we see this is the unique local minimum, and thus the global minimum as the loss is infinite when  $\hat{y}$  tends to either infinity.

### Exercise 19.7.#SVME

Figure 19.22 showed how a circle at the origin can be linearly separated by mapping from the features  $(x_1, x_2)$  to the two dimensions  $(x_1^2, x_2^2)$ . But what if the circle is not located at the origin? What if it is an ellipse, not a circle? The general equation for a circle (and hence the decision boundary) is  $(x_1 - a)^2 + (x_2 - b)^2 - r^2 = 0$ , and the general equation for an ellipse is  $c(x_1 - a)^2 + d(x_2 - b)^2 - 1 = 0$ .

- a. Expand out the equation for the circle and show what the weights  $w_i$  would be for the decision boundary in the four-dimensional feature space  $(x_1, x_2, x_1^2, x_2^2)$ . Explain why this means that any circle is linearly separable in this space.

**b.** Do the same for ellipses in the five-dimensional feature space  $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$ .

a. The circle equation expands into five terms

$$0 = x_1^2 + x_2^2 - 2ax_1 - 2bx_2 + (a^2 + b^2 - r^2)$$

corresponding to weights  $w = (2a, 2b, 1, 1)$  and intercept  $a^2 + b^2 - r^2$ . This shows that a circular boundary is linear in this feature space, allowing linear separability.

In fact, the three features  $x_1, x_2, x_1^2 + x_2^2$  suffice.

b. The (axis-aligned) ellipse equation expands into six terms

$$0 = cx_1^2 + dx_2^2 - 2acx_1 - 2bdx_2 + (a^2c + b^2d - 1)$$

corresponding to weights  $w = (2ac, 2bd, c, d, 0)$  and intercept  $a^2 + b^2 - r^2$ . This shows that an elliptical boundary is linear in this feature space, allowing linear separability.

In fact, the four features  $x_1, x_2, x_1^2, x_2^2$  suffice for any axis-aligned ellipse.

### Exercise 19.7.#SVMX

Construct a support vector machine that computes the XOR function. Use values of +1 and -1 (instead of 1 and 0) for both inputs and outputs, so that an example looks like  $([-1, 1], 1)$  or  $([-1, -1], -1)$ . Map the input  $[x_1, x_2]$  into a space consisting of  $x_1$  and  $x_1 x_2$ . Draw the four input points in this space, and the maximal margin separator. What is the margin? Now draw the separating line back in the original Euclidean input space.

The examples map from  $[x_1, x_2]$  to  $[x_1, x_1 x_2]$  coordinates as follows:

$[-1, -1]$  (negative) maps to  $[-1, +1]$

$[-1, +1]$  (positive) maps to  $[-1, -1]$

$[+1, -1]$  (positive) maps to  $[+1, -1]$

$[+1, +1]$  (negative) maps to  $[+1, +1]$

Thus, the positive examples have  $x_1 x_2 = -1$  and the negative examples have  $x_1 x_2 = +1$ . The maximum margin separator is the line  $x_1 x_2 = 0$ , with a margin of 1. The separator corresponds to the  $x_1 = 0$  and  $x_2 = 0$  axes in the original space—this can be thought of as the limit of a hyperbolic separator with two branches.

## 19.8 Ensemble Learning

### Exercise 19.8.#EERE

Consider a binary classification problem with an ensemble learning algorithm that uses simple majority voting among  $K$  learned hypotheses. Suppose that each hypothesis has error  $\epsilon$  and that the errors made by each hypothesis are independent of the others'. Calculate a

formula for the error of the ensemble algorithm in terms of  $K$  and  $\epsilon$ , and evaluate it for the cases where  $K = 5, 11$ , and  $21$  and  $\epsilon = 0.1, 0.2$ , and  $0.4$ . If the independence assumption is removed, is it possible for the ensemble error to be *worse* than  $\epsilon$ ?

The ensemble makes an error if more than half the hypotheses are wrong. So, for example when  $K = 5$ , that would be when 3, 4, or 5 hypotheses are wrong. There is only 1 way for all 5 to be wrong, but there are 5 ways for 4 to be wrong, and  $\binom{5}{3=10}$  ways for 3 to be wrong. So the overall error rate for the ensemble is:

$$\sum_{i=\lceil K/2 \rceil}^K \binom{K}{i} \epsilon^i (1-\epsilon)^{K-i}$$

Running a program to compute these values gives us:

```
K = 5, e = 0.1, err = 0.008560
K = 5, e = 0.2, err = 0.057920
K = 5, e = 0.4, err = 0.317440
K = 11, e = 0.1, err = 0.000296
K = 11, e = 0.2, err = 0.011654
K = 11, e = 0.4, err = 0.246502
K = 21, e = 0.1, err = 0.000001
K = 21, e = 0.2, err = 0.000970
K = 21, e = 0.4, err = 0.174378
```

If the hypotheses are not independent, we are not guaranteed such good results. In fact, the overall error rate might get worse than with a single hypothesis. The total expected number of correct “votes” over  $N$  problem instances will be  $NK(1 - \epsilon)$ , but it might be that these correct votes are arranged ineffectively—the votes can be “gerrymandered.” Consider what happens when, on some of the problem instances, every hypothesis is correct, and on the other problem instances, the incorrect prediction wins by one vote. Then the error rate will be greater than  $\epsilon$ .

### Exercise 19.8.#BBSG

Define the following terms in your own words.

- a. Bagging
- b. Boosting
- c. Stacking
- d. Random forest

- a. Bagging: Select examples from the training set  $K$  times, and run a machine learning algorithm on each set, to get  $K$  individual hypotheses. The resulting ensemble hypothesis is a function that polls each of the  $K$  individuals and averages their result.

- b. Boosting: An ensemble method where models vote, as in bagging, but hypotheses that do better get more votes, and the training sets are chosen to concentrate on fixing the errors that the ensemble is making.
- c. Stacking: An ensemble method that combines individual hypotheses from different machine learning model classes (as contrasted with bagging, which combines  $K$  hypotheses from the same model class).
- d. Random forest: a way of bagging decision trees but introducing randomness so that the individual trees are less similar to each other.

## 19.9 Developing Machine Learning Systems

### Exercise 19.9.#LRTF

Indicate which of answer(s) in parentheses are correct for each question:

- a. For binary class classification, does logistic regression always produce a linear decision boundary? (Yes; No)
- b. You train a linear classifier on 1,000 training points and discover that the training accuracy is only 50%. Which of the following, if done in isolation, has a good chance of improving your training accuracy? (Add new features; Train on more data; Train on less data)
- c. Does there exist data that is separable with a Gaussian RBF kernel but not with a quadratic kernel? (Yes; No)
- d. Does there exist data that is separable with a linear kernel but not with a quadratic kernel? (Yes; No)
- e. You are training a logistic regression model and you find that your training loss is near 0 but test loss is very high. Which of the following is expected to help to reduce test loss? (Increase the training data size; Decrease the training data size; Increase model complexity; Decrease model complexity; Train on a combination of your training data and your test data but test only on your test data; Conclude that Machine Learning does not work)

- a. Yes
- b. Add new features; Train on less data
- c. Yes
- d. No
- e. Increase the training data size; Decrease model complexity; Train on a combination of your training data and your test data but test only on your test data. *Note:* Since we are facing overfitting we can increase the training data size or decrease the model complexity to combat this. Also, if we train on our test data our test loss will definitely improve dramatically, but you should *never* do this in practice because it defeats the purpose of testing, and will make performance worse when the model is deployed and used on new data.

**Exercise 19.9.#MLHF**

Consider the problem of detecting human faces in images of larger scenes that may contain one or more faces, or no faces at all. The specific goal is to identify the *location and size* of each face in an image, rather than merely determining *whether* a face is present in the image. Suppose you are given as training data a set of images in which all of the faces have been labeled (specified as a list of pixel positions for each face). Describe one approach to solving this problem using *binary classification* machine learning. Specify what will serve as training examples for the binary classification algorithm, and how the binary classifier can be used to detect faces in new images. You don't need to specify the specific features or classification algorithm to use.

A binary classifier can be trained to classify fixed-size rectangular regions of an images as being of a face or not. A particular size of rectangle is chosen. Positive training examples can be obtained by picking, for each identified face in the training images, a minimum-size bounding box of the correct aspect ratio that contains all of the pixels in the face. Negative training examples can be obtained by randomly picking bounding boxes of the correct aspect ratio from the training images that do not contain any face pixels. These examples are then rescaled to be of the correct size.

To process a new image, every possible bounding box on the image of the correct aspect ratio at one of several scales is classified by rescaling as necessary and then invoking the classifier.

**Exercise 19.9.#CLRL**

Your boss provides you with an image dataset in which some of the images contain your company's logo, and others contain competitors' logos. You are tasked to code up a classifier to distinguish your company's logos from competitors' logos. You complete the assignment quickly and even send your boss your code for training the classifier, but your boss says there is a problem: when running your code on a data set where the images are assigned random labels, the classifier achieved perfect accuracy on the training set. Do you agree that this is a problem? Justify your answer.

Yes, this is a problem. The classifier is overfitting the training set. The fact that it had perfect accuracy with random labels suggests that it does not learn any real underlying structure in the data; it most likely just memorized each of the training cases.

**Exercise 19.9.#ROCV**

Use a machine learning package to solve a problem of your choosing using data of your choosing. Plot a t-SNE map or other visualization of your data, and a ROC curve of your results. Experiment with other data exploration tools. What tools were most helpful? What did you learn about the data and the results?

Answers will vary from student to student.

# EXERCISES 20

## LEARNING PROBABILISTIC MODELS

### 20.1 Statistical Learning

#### Exercise 20.1.#BAYC

The data used for Figure 20.1 on page 723 can be viewed as being generated by  $h_5$ . For each of the other four hypotheses, generate a data set of length 100 and plot the corresponding graphs for  $P(h_i | d_1, \dots, d_N)$  and  $P(D_{N+1} = \text{lime} | d_1, \dots, d_N)$ . Comment on your results.

The code for this exercise is a straightforward implementation of Equations 20.1 and 20.2. Figure S20.1 shows the results for data sequences generated from  $h_3$  and  $h_4$ . (Plots for  $h_1$  and  $h_2$  are essentially identical to those for  $h_5$  and  $h_4$ .) Results obtained by students may vary because the data sequences are generated randomly from the specified candy distribution. In (a), the samples very closely reflect the true probabilities and the hypotheses other than  $h_3$  are effectively ruled out very quickly. In (c), the early sample proportions are somewhere between 50/50 and 25/75; furthermore,  $h_3$  has a higher prior than  $h_4$ . As a result,  $h_3$  and  $h_4$  vie for supremacy. Between 50 and 60 samples, a preponderance of limes ensures the defeat of  $h_3$  and the prediction quickly converges to 0.75.

#### Exercise 20.1.#BAYD

Repeat Exercise BAYC, this time plotting the values of  $P(D_{N+1} = \text{lime} | h_{\text{MAP}})$  and  $P(D_{N+1} = \text{lime} | h_{\text{ML}})$ .

(Plots not shown.) plots are shown in Figure S20.1. Because both MAP and ML choose exactly one hypothesis for predictions, the prediction probabilities are all 0.0, 0.25, 0.5, 0.75, or 1.0. For small data sets the ML prediction in particular shows very large variance.

#### Exercise 20.1.#BAYE

Suppose that Ann's utilities for cherry and lime candies are  $c_A$  and  $\ell_A$ , whereas Bob's utilities are  $c_B$  and  $\ell_B$ . (But once Ann has unwrapped a piece of candy, Bob won't buy it.) Presumably, if Bob likes lime candies much more than Ann, it would be wise for Ann to sell her bag of candies once she is sufficiently sure of its lime content. On the other hand, if Ann unwraps too many candies in the process, the bag will be worth less. Discuss the problem of determining the optimal point at which to sell the bag. Determine the expected

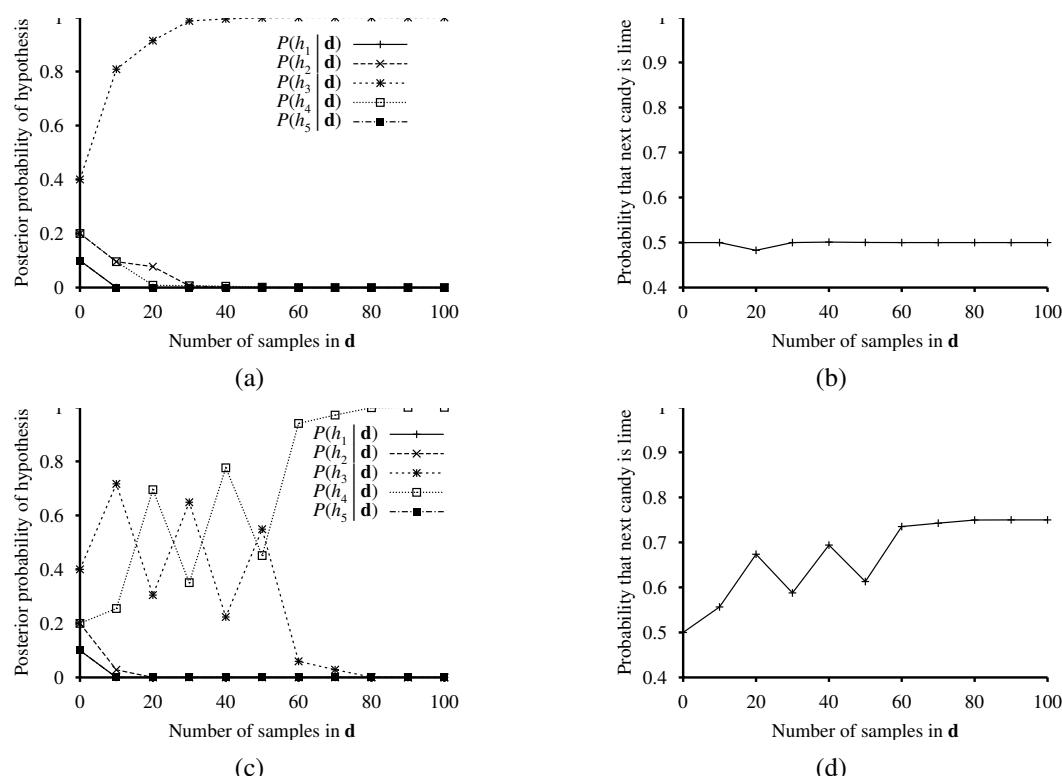
utility of the optimal procedure, given the prior distribution from Section 20.1.

This is a nontrivial sequential decision problem, but can be solved using the tools developed in the book. It leads into general issues of statistical decision theory, stopping rules, etc. Here, we sketch the “straightforward” solution.

We can think of this problem as a simplified form of POMDP (see Chapter 17). The “belief states” are defined by the numbers of cherry and lime candies observed so far in the sampling process. Let these be  $C$  and  $L$ , and let  $U(C, L)$  be the utility of the corresponding belief state. In any given state, there are two possible decisions: *sell* and *sample*. There is a simple Bellman equation relating  $Q$  and  $U$  for the sampling case:

$$Q(C, L, \text{sample}) = P(\text{cherry}|C, L)U(C + 1, L) + P(\text{lime}|C, L)U(C, L + 1)$$

Let the posterior probability of each  $h_i$  be  $P(h_i|C, L)$ , the size of the bag be  $N$ , and the fraction of cherries in a bag of type  $i$  be  $f_i$ . Then the value obtained by selling is given by



**Figure S20.1** Graphs for Ex. 20.1. (a) Posterior probabilities  $P(h_i|d_1, \dots, d_N)$  over a sample sequence of length 100 generated from  $h_3$  (50% cherry + 50% lime). (b) Bayesian prediction  $P(d_{N+1} = \text{lime}|d_1, \dots, d_N)$  given the data in (a). (c) Posterior probabilities  $P(h_i|d_1, \dots, d_N)$  over a sample sequence of length 100 generated from  $h_4$  (25% cherry + 75% lime). (d) Bayesian prediction  $P(d_{N+1} = \text{lime}|d_1, \dots, d_N)$  given the data in (c).

the value of the sampled candies (which Ann gets to keep) plus the price paid by Bob (which equals the expected utility of the remaining candies for Bob):

$$Q(C, L, \text{sell}) = Cc_A + L\ell_A + \sum_i P(h_i|C, L)[(f_iN - C)c_B + ((1 - f_i)N - L)\ell_B]$$

and of course we have

$$U(C, L) = \max\{Q(C, L, \text{sell}), Q(C, L, \text{sample})\}.$$

Thus we can set up a dynamic program to compute  $Q$  given the obvious boundary conditions for the case where  $C + L = N$ . The solution of this dynamic program gives the optimal policy for Ann. It will have the property that if she should sell at  $(C, L)$ , then she should also sell at  $(C, L + k)$  for all positive  $k$ . Thus, the problem is to determine, for each  $C$ , the threshold value of  $L$  at or above which she should sell. A minor complication is that the formula for  $P(h_i|C, L)$  should take into account the non-replacement of candies and the finiteness of  $N$ , otherwise odd things will happen when  $C + L$  is close to  $N$ .

### Exercise 20.1.#DRST

Two statisticians go to the doctor and are both given the same prognosis: A 40% chance that the problem is the deadly disease  $A$ , and a 60% chance of the fatal disease  $B$ . Fortunately, there are anti- $A$  and anti- $B$  drugs that are inexpensive, 100% effective, and free of side-effects. The statisticians have the choice of taking one drug, both, or neither. What will the first statistician (an avid Bayesian) do? How about the second statistician, who always uses the maximum likelihood hypothesis?

The doctor does some research and discovers that disease  $B$  actually comes in two versions, dextro- $B$  and levo- $B$ , which are equally likely and equally treatable by the anti- $B$  drug. Now that there are three hypotheses, what will the two statisticians do?

The Bayesian approach would be to take both drugs. The maximum likelihood approach would be to take the anti- $B$  drug. In the case where there are two versions of  $B$ , the Bayesian still recommends taking both drugs, while the maximum likelihood approach is now to take the anti- $A$  drug, since it has a 40% chance of being correct, versus 30% for each of the  $B$  cases. This is of course a caricature, and you would be hard-pressed to find a doctor, even a rabid maximum-likelihood advocate who would prescribe like this. But you can find ones who do research like this.

## 20.2 Learning with Complete Data

### Exercise 20.2.#MISC.A

You have classification data with classes  $Y \in \{+1, -1\}$  and features  $F_i \in \{+1, -1\}$  for  $i \in \{1, \dots, K\}$ . Say you duplicate each feature, so now each example has  $2K$  features, with  $F_{K+i} = F_i$  for  $i \in \{1, \dots, K\}$ . Compare the *original* feature set with the *doubled* one and

indicate whether the below statements are true or false for Naïve Bayes. You may assume that in the case of ties, class +1 is always chosen. Assume that there are equal numbers of training examples in each class.

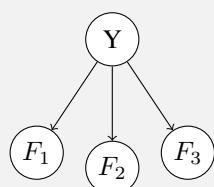
- (i) The test accuracy could be higher with the original features.
- (ii) The test accuracy could be higher with the doubled features.
- (iii) The test accuracy will be the same with either feature set.
- (iv) On a given training instance, the conditional probability  $P(Y|F_1, \dots)$  on a training instance could be more extreme (i.e. closer to 0 or 1) with the original features.
- (v) On a given training instance, the conditional probability  $P(Y|F_1, \dots)$  on a training instance could be more extreme (i.e. closer to 0 or 1) with the doubled features.
- (vi) On a given training instance, the conditional probability  $P(Y|F_1, \dots)$  on a training instance will be the same with either feature set.

- (i) True.
- (ii) False.
- (iii) False.
- (iv) False.
- (v) True.
- (vi) False.

Naïve Bayes makes the conditional independence assumption that all features are independent given the class label. Redundant features lead to “overconfidence” that may result in errors.

### Exercise 20.2.#MISC.F

Consider training the Naive Bayes model shown on the left with the training data provided in the table on the right.



|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| $F_1$ | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| $F_2$ | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| $F_3$ | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| $Y$   | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

Calculate the maximum likelihood estimate of  $P(F_1 = 1 | Y = 0)$ .

$\frac{3}{5}$ . This is found by counting the samples. There are 5 samples where  $Y = 0$ , and  $F_1 = 1$  in 3 of them.

**Exercise 20.2.#XXXX**

Your friend claims that he can write an effective Naive Bayes spam detector with only three features: the hour of the day that the email was received ( $H \in \{1, 2, \dots, 24\}$ ), whether it contains the words “free money” ( $W \in \{\text{yes}, \text{no}\}$ ), and whether the email address of the sender is Known in his address book, Seen before in his inbox, or Unseen before ( $E \in \{\text{K}, \text{S}, \text{U}\}$ ).

- a. Flesh out the following information about this Bayes net:

- (i) Graph structure.
- (ii) Parameters.
- (iii) Size of the set of parameters.

Suppose now that you labeled three of the emails in your mailbox to test this idea:

| spam or ham? | $H$ | $W$ | $E$ |
|--------------|-----|-----|-----|
| spam         | 3   | yes | S   |
| ham          | 14  | no  | K   |
| ham          | 15  | no  | K   |

- b. Use the three instances to estimate the maximum likelihood estimate of the parameters.  
 c. Using the maximum likelihood parameters, find the predicted class of a new datapoint with  $H = 3$ ,  $W = \text{no}$ ,  $E = U$ .  
 d. You observe that you tend to receive spam emails in batches. In particular, if you receive one spam message, the next message is more likely to be a spam message as well. Explain a new graphical model which most naturally captures this phenomena.
- (i) Graph structure.
  - (ii) Parameters.
  - (iii) Size of the set of parameters.

- a. (i) Graph structure: Naive Bayes net with three leaves ( $\text{K}, \text{S}, \text{U}$ ).

(ii) Letting  $i \in \{1, \dots, 23\}$ ,  $j \in \{\text{K}, \text{S}\}$ ,  $c \in \{\text{spam}, \text{ham}\}$ , we have the following parameters:

- $\theta_{\text{spam}}$
- $\theta_{H,i,c}$
- $\theta_{W,c}$
- $\theta_{E,j,c}$

Note that to parametrize a distribution over 24 hours, we only need 23 parameters; and to parametrize a distribution over 3 sender categories, we only need 2 parameters (since the last class in both distributions is forced to take on the value that ensures the entire CPT sums to 1).

- (iii) Size of the set of parameters:  $1 + 23 \cdot 2 + 2 + 2 \cdot 2$ .

- b. •  $\theta_{\text{spam}} = \frac{1}{3}$   
 •  $\theta_{H,3,\text{spam}} = 1$

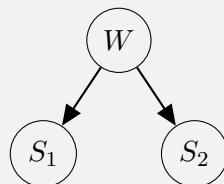
- $\theta_{H,14,ham} = \frac{1}{2}$
  - $\theta_{H,15,ham} = \frac{1}{2}$
  - $\theta_{W,spam} = 1$
  - $\theta_{E,S,spam} = 1$
  - $\theta_{E,K,ham} = 1$
- c. Both assign a likelihood of zero, so no prediction is specified by the maximum likelihood parameters.
- d. (i) Graph structure: An HMM, except that there are three observations at each node.  
(ii) Parameters: Add 2 parameters: transition to spam from spam and from ham.  
(iii) Size of the set of parameters: Add 2 to the expression in the first question.

### Exercise 20.2.#XXXX

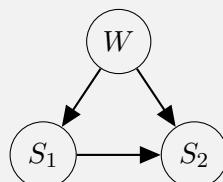
Stoplights  $S_1$  and  $S_2$  can each be in one of two states: green ( $g$ ) or red ( $r$ ). Additionally, the machinery behind both stoplights ( $W$ ) can be in one of two states: working ( $w$ ) or broken ( $b$ ). We collect data by observing the stoplights and the state of their machinery on seven different days. Here is a Naïve Bayes graphical model for the stoplights:

**Data:**

| Day | $S_1$ | $S_2$ | $W$ |
|-----|-------|-------|-----|
| 1   | $g$   | $r$   | $w$ |
| 2   | $g$   | $r$   | $w$ |
| 3   | $g$   | $r$   | $w$ |
| 4   | $r$   | $g$   | $w$ |
| 5   | $r$   | $g$   | $w$ |
| 6   | $r$   | $g$   | $w$ |
| 7   | $r$   | $r$   | $b$ |

**Model:**

- a. Write the probability tables for  $P(W)$ ,  $P(S_1|W)$ ,  $P(S_2|W)$  with the naive Bayes joint distribution that assigns highest probability to the data we observed.
- b. What's the posterior probability  $P(W = b|S_1 = r, S_2 = r)$ ?
- c. Instead of Naïve Bayes, we use the following graphical model and fill in probability tables with estimates that assign highest probability to the data we observed:



- (i) What's the posterior probability  $P(W = b|S_1 = r, S_2 = r)$ ? (Hint: you should not have to do a lot of work.)
- (ii) What is it about the problem that makes the second graphical model more suitable than the first?

|   | $W$ | $\mathbf{P}(W)$ |
|---|-----|-----------------|
| w | 6/7 |                 |
| b | 1/7 |                 |

a.

| $S_1$ | $W$ | $\mathbf{P}(S_1 W)$ |
|-------|-----|---------------------|
| g     | w   | 1/2                 |
| r     | w   | 1/2                 |
| g     | b   | 0                   |
| r     | b   | 1                   |

| $S_2$ | $W$ | $\mathbf{P}(S_2 W)$ |
|-------|-----|---------------------|
| g     | w   | 1/2                 |
| r     | w   | 1/2                 |
| g     | b   | 0                   |
| r     | b   | 1                   |

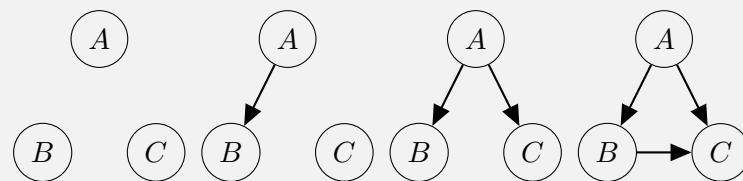
b.

$$\begin{aligned}
 P(W = b|S_1 = r, S_2 = r) &= \frac{P(W = b, S_1 = r, S_2 = r)}{\sum_w P(W = w, S_1 = r, S_2 = r)} \\
 &= \frac{P(W = b)P(S_1 = r|W = b)P(S_2 = r|W = b)}{\sum_w P(W = w)P(S_1 = r|W = w)P(S_2 = r|W = w)} \\
 &= \frac{\frac{1}{7} \cdot 1 \cdot 1}{\frac{1}{7} \cdot 1 \cdot 1 + \frac{6}{7} \cdot \frac{1}{2} \cdot \frac{1}{2}} \\
 &= \frac{2}{5}
 \end{aligned}$$

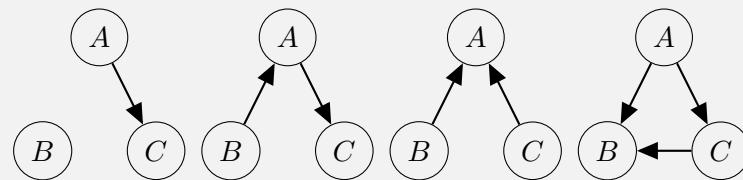
- c. (i) Since there is an edge directly connecting each pair of variables, there are no conditional independence assumptions. The factors are  $P(W)$ ,  $P(S_1|W)$ ,  $P(S_2|S_1, W)$ . So  $P(W = b|S_1 = r, S_2 = r) = \frac{\text{Count}(W=b, S_1=r, S_2=r)}{\text{Count}(S_1=r, S_2=r)} = \frac{1}{1} = 1$
- (ii)  $S_1$  and  $S_2$  are not conditionally independent given  $W$  in the data (and for real stop lights), since when  $W = w$ , they are always anti-correlated. The second graphical model does not make this assumption, but the first one does. Thus, the second model is more suitable.

### Exercise 20.2.#XXXX

You want to learn a Bayes' net over the random variables  $A, B, C$ . You decide you want to learn not only the Bayes' net parameters, but also the structure from the data. You are willing to consider the 8 structures shown below. First you use your training data to perform maximum likelihood estimation of the parameters of each of the Bayes' nets. Then for each of the learned Bayes' nets, you evaluate the likelihood of the training data ( $l^{\text{train}}$ ), and the likelihood of your cross-validation data ( $l^{\text{cross}}$ ). Both likelihoods are shown below each structure.



|                    |        |        |        |        |
|--------------------|--------|--------|--------|--------|
| $l^{\text{train}}$ | 0.0001 | 0.0005 | 0.0015 | 0.0100 |
| $l^{\text{cross}}$ | 0.0001 | 0.0004 | 0.0011 | 0.0009 |
|                    | (a)    | (b)    | (c)    | (d)    |



|                    |        |        |        |        |
|--------------------|--------|--------|--------|--------|
| $l^{\text{train}}$ | 0.0008 | 0.0015 | 0.0020 | 0.0100 |
| $l^{\text{cross}}$ | 0.0006 | 0.0011 | 0.0010 | 0.0009 |
|                    | (e)    | (f)    | (g)    | (h)    |

- a. Which Bayes' net structure will (on expectation) perform best on test-data? (If there is a tie, list all Bayes' nets that are tied for the top spot.) Justify your answer.
- b. Two pairs of the learned Bayes' nets have identical likelihoods. Explain why this is the case.
- c. For every two structures  $S_1$  and  $S_2$ , where  $S_2$  can be obtained from  $S_1$  by adding one or more edges,  $l^{\text{train}}$  is higher for  $S_2$  than for  $S_1$ . Explain why this is the case.

- a. Bayes' nets (c) and (f), as they have the highest cross validation data likelihood.
- b. (c) and (f) have the same likelihoods, and (d) and (h) have the same likelihoods. When learning a Bayes' net with maximum likelihood, we end up selecting the distribution that maximizes the likelihood of the training data from the set of all distributions that can be represented by the Bayes' net structure. (c) and (f) have the same set of conditional independence assumptions, and hence can represent the same set of distributions. This means that they end up with the same distribution as the one that maximizes the training data likelihood, and therefore have identical training and cross validation likelihoods. Same holds true for (d) and (h).
- c. When learning a Bayes' net with maximum likelihood, we end up selecting the distribution that maximizes the likelihood of the training data from the set of all distributions that can be represented by the Bayes' net structure. Adding an edge grows the set of distributions that can be represented by the Bayes' net, and can hence only increase the training data likelihood under the best distribution in this set.

**Exercise 20.2.#XXXX**

The Naïve Bayes model has been famously used for classifying spam. We will use it in the ‘‘bag-of-words’’ model:

- Each email has binary label  $Y$  which takes values in {spam, ham}.
- Each word  $w$  of an email, no matter where in the email it occurs, is assumed to have probability  $P(W = w | Y)$ , where  $W$  takes on words in a pre-determined dictionary. Punctuation is ignored.
- Take an email with  $K$  words  $w_1, \dots, w_K$ . For instance: email ‘‘hi hi you’’ has  $w_1 = \text{hi}$ ,  $w_2 = \text{hi}$ ,  $w_3 = \text{you}$ . Its label is given by:  

$$\arg \max_y P(Y = y | w_1, \dots, w_K) = \arg \max_y P(Y = y) \prod_{i=1}^K P(W = w_i | Y = y).$$

- a.** You are in possession of a bag of words spam classifier trained on a large corpus of emails. Below is a table of some estimated word probabilities.

| $W$                      | note | to  | self | become | perfect |
|--------------------------|------|-----|------|--------|---------|
| $P(W   Y = \text{spam})$ | 1/6  | 1/8 | 1/4  | 1/4    | 1/8     |
| $P(W   Y = \text{ham})$  | 1/8  | 1/3 | 1/4  | 1/12   | 1/12    |

You are given a new email to classify, with only two words:

perfect note

For what threshold value,  $c$ , in the expression  $P(Y = \text{spam}) > c$ , will the bag of words model predict the label ‘‘spam’’ as the most likely label?

- b.** You are given only three emails as a training set:

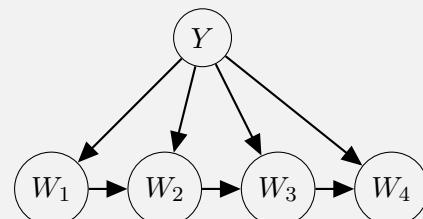
(Spam) dear sir, I write to you in hope of recovering my gold watch.  
 (Ham) hey, lunch at 12?  
 (Ham) fine, watch it tomorrow night.

Given the training set, what are the probability estimates for:

- $P(W = \text{sir} | Y = \text{spam})$
- $P(W = \text{watch} | Y = \text{ham})$
- $P(W = \text{gauntlet} | Y = \text{ham})$
- $P(Y = \text{ham})$

**c. Becoming less naïve**

We are now going to improve the representational capacity of the model. Presence of word  $w_i$  will be modeled not by  $P(W = w_i | Y)$ , where it is only dependent on the label, but by  $P(W = w_i | Y, W_{i-1})$ , where it is also dependent on the previous word. The corresponding model for an email of only four words is given on the right.



- (i) With a vocabulary consisting of  $V$  words, what is the *minimal* number of conditional word probabilities that need to be estimated for this model?

- (ii) Which of the following are expected effects of using the new model instead of the old one, if both are trained with a very large set of emails (equal number of spam and ham examples)?
- The entropy of the posterior  $P(Y|W)$  should on average be lower with the new model. (In other words, the model will tend to be more confident in its answers.)
  - The accuracy on the **training** data should be higher with the new model.
  - The accuracy on the **held-out** data should be higher with the new model.

**a.**

$$\begin{aligned}
 P(Y = \text{spam} \mid w_1 = \text{perfect}, w_2 = \text{note}) &> P(Y = \text{ham} \mid w_1 = \text{perfect}, w_2 = \text{note}) \\
 \left( \frac{P(w_1 = \text{perfect} \mid Y = s) \times}{P(w_2 = \text{note} \mid Y = s) \times P(Y = s)} \right) &> \left( \frac{P(w_1 = \text{perfect} \mid Y = h) \times}{P(w_2 = \text{note} \mid Y = h) \times P(Y = h)} \right) \\
 1/8 \times 1/6 \times P(Y = \text{spam}) &> 1/12 \times 1/8 \times (1 - P(Y = \text{spam})) \\
 2/96 \times P(Y = \text{spam}) &> 1/96 - 1/96 \times P(Y = \text{spam}) \\
 3/96 \times P(Y = \text{spam}) &> 1/96 \\
 P(Y = \text{spam}) &> 1/3
 \end{aligned}$$

So the threshold is  $c = 1/3$ 

- b.** (i) Intuitively, the conditional probability is  $P(\text{word} \mid \text{it's a word in an email of type } Y)$ . We estimate this probability with word counts:

$$\begin{aligned}
 P(W = \text{sir} \mid Y = \text{spam}) &= \frac{c_w(W = \text{sir}, Y = \text{spam}) / c_w(\text{total})}{c_w(Y = \text{spam}) / c_w(\text{total})} \\
 &= \frac{c_w(W = \text{sir}, Y = \text{spam})}{c_w(Y = \text{spam})} = \frac{1}{13}
 \end{aligned}$$

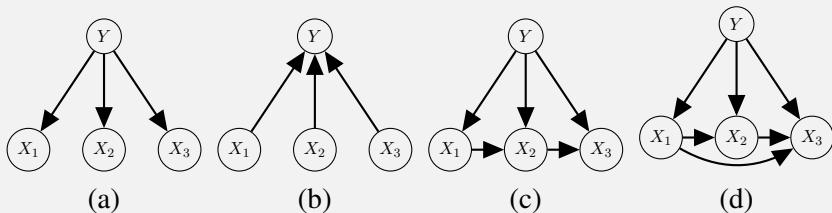
- (ii) Similarly,  $P(W = \text{watch} \mid Y = \text{ham}) = \frac{1}{9}$ .
- (iii) The word “gauntlet” does not occur in our training emails, and since we’re not smoothing, we estimate its conditional probability to be 0.
- (iv) Estimating the prior probability  $P(Y = \text{ham})$  only requires counting emails:  
 $\frac{c_e(Y=\text{ham})}{c_e(\text{total})} = \frac{2}{3}$ .
- c.** (i) We need to consider  $V$  possible words for each one of  $V$  possible previous words and 2 possible labels:  $2V^2$ .
- (ii) A. False  
B. True  
C. True

The new model is closer to an actual model of language, so it should better model emails and thus filter spam from non-spam on both the training and held-out datasets. Remember that Naïve Bayes is an *overconfident* model: it gives

unwarrantedly low-entropy posteriors. This model is less “naïve” and is therefore less overconfident – its posterior can be expected to be higher entropy.

### Exercise 20.2.#XXXX

Abhishek has been getting a lot of spam recently and is not satisfied with his email client’s Naive Bayes spam classifier. Thankfully, he knows about Bayes Nets and has decided to implement his own spam classifier. It is your job to help him model his Bayes Nets and train them. The following are 4 Bayes Nets he is considering. Each variable can take on the values  $\{0, 1\}$ .



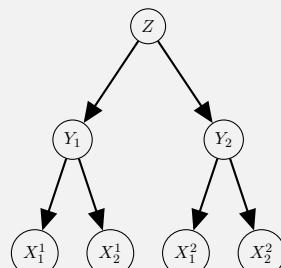
- Abhishek wants to know how much memory he needs to store each of these Bayes Nets on disk. The amount of memory depends on the number of values he would need to store in the CPTs for that Bayes Net. For each of the nets above give the **least** number of parameters he would need to store to completely specify the Bayes Net.
- It’s now time to train the Bayes Nets. Abhishek has training datasets  $D_l$ ,  $D_m$ ,  $D_s$  and a test dataset  $D_t$ . The number of training examples in each set vary as  $|D_l| > |D_m| > |D_s|$ .  $e_{tr}$  and  $e_{te}$  represent train and test errors and their arguments represent which dataset a model was trained on. For example,  $e_{tr}(A, D_l)$  refers to the training error of model A on dataset  $D_l$ .
  - Abhishek tries a bunch of experiments using model  $D$ . In a typical scenario (where train and test data are sampled from the same underlying distribution), order the following errors by their expected value.
    - Order  $e_{tr}(D, D_l), e_{tr}(D, D_m), e_{tr}(D, D_s)$
    - Order  $e_{te}(D, D_l), e_{te}(D, D_m), e_{te}(D, D_s)$
    - Order  $e_{tr}(D, D_l), e_{te}(D, D_l)$
    - Order  $e_{tr}(D, D_s), e_{te}(D, D_s)$
  - Abhishek is now trying to compare performance across different models. Order the following errors by their expected value:
    - $e_{tr}(A, D_l), e_{tr}(B, D_l), e_{tr}(D, D_l)$
    - $e_{te}(A, D_l), e_{te}(B, D_l), e_{te}(D, D_l)$

- The number of parameters in the Bayes Net is the total number of probability values you need in the CPTs of the Bayes Net. You also need to remember that if  $A$  takes  $k$  values,  $P(A)$  can be represented by  $k - 1$  values as the last value can be chosen so that the probabilities sum to 1. For example in (A),  $P(X_1|Y = 0)$  has 1 parameter and so does  $P(X_1|Y = 1)$ .
  - (a): 7

- $P(Y)$  needs 1 parameter,  $P(X_i|Y)$  needs 2 parameters, one parameter for each possible value of Y. Hence we have  $1 + 3 \cdot 2 = 7$ .
  - (b): 11
    - $P(X_i)$  needs 1 parameter,  $P(Y|X_1, X_2, X_3)$  needs 8 parameters, one parameter for each possible value of  $(X_1, X_2, X_3)$ . Hence we have  $3 \cdot 1 + 8 = 11$ .
  - (c): 11
    - $P(Y)$  needs 1 parameter,  $P(X_1|Y)$  needs 2 parameters,  $P(X_2|X_1, Y)$  needs 4 parameters, and  $P(X_3|X_2, Y)$  needs 4 parameters. Hence we have  $1 + 2 + 4 + 4 = 11$ .
  - (d): 15
    - $P(Y)$  needs 1 parameter,  $P(X_1|Y)$  needs 2 parameters,  $P(X_2|X_1, Y)$  needs 4 parameters, and  $P(X_3|X_1, X_2, Y)$  needs 8 parameters. Hence we have  $1 + 2 + 4 + 8 = 15$ .
- b. The rationale is to test knowledge about “power” of a model and overfitting/underfitting. Bayes Nets with less number of parameters have low capacity/power as they can model a more restricted class of distributions due to greater independence assumptions. A more powerful model would overfit on less data and a weak model would underfit. In this question, we are assuming that  $D_l$  is *very* large. Rationale for correct answers (column major):
- (i) A.  $e_{tr}(D, D_l) \geq e_{tr}(D, D_m) \geq e_{tr}(D, D_s)$ . A larger dataset would be tougher to model than a smaller. Thus training error would be greater for a larger dataset.
  - B.  $e_{te}(D, D_s) \geq e_{te}(D, D_m) \geq e_{te}(D, D_l)$ . A smaller dataset would overfit for a complex model like  $D$  and thus testing errors would be more.
  - C.  $e_{tr}(D, D_l) \leq e_{te}(D, D_l)$  Training error less than test error in general because we explicitly model the training data and need to generalize to test data.
  - D.  $e_{tr}(D, D_s) \leq e_{te}(D, D_s)$  Overfitting for smaller data and complex model
- (ii) A.  $e_{tr}(A, D_l) \geq e_{tr}(B, D_l) \geq e_{tr}(D, D_l)$ . Weaker model underfits on large dataset
- B.  $e_{te}(A, D_l) \geq e_{te}(B, D_l) \geq e_{te}(D, D_l)$ . Same as above. Weaker model cant model the distribution well enough if we have a lot of data.

**Exercise 20.2.#XXXX**

You are given a model with two distinct label variables  $Y_1, Y_2$ , and there is a super label  $Z$  which conditions all of these labels, thus giving us this hierarchical naïve Bayes model. The conditional probabilities for the model are parametrized by  $p_1, p_2, q_0, q_1$  and  $r$ . **Note that some of the parameters are shared as in the previous part.**



| $X_1^i$ | $Y_i$ | $P(X_1^i   Y_i)$ |
|---------|-------|------------------|
| 0       | 0     | $p_1$            |
| 1       | 0     | $1 - p_1$        |
| 0       | 1     | $1 - p_1$        |
| 1       | 1     | $p_1$            |

| $X_2^i$ | $Y_i$ | $P(X_2^i   Y_i)$ |
|---------|-------|------------------|
| 0       | 0     | $p_2$            |
| 1       | 0     | $1 - p_2$        |
| 0       | 1     | $1 - p_2$        |
| 1       | 1     | $p_2$            |

| $Y_i$ | $Z$ | $P(Y_i   Z)$ |
|-------|-----|--------------|
| 0     | 0   | $1 - q_0$    |
| 1     | 0   | $q_0$        |
| 0     | 1   | $1 - q_1$    |
| 1     | 1   | $q_1$        |

| $Z$ | $P(Z)$  |
|-----|---------|
| 0   | $1 - r$ |
| 1   | $r$     |

The data for training the model is the following.

| sample number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---------------|---|---|---|---|---|---|---|---|---|----|
| $X_1^1$       | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0  |
| $X_1^2$       | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0  |
| $X_2^1$       | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0  |
| $X_2^2$       | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0  |
| $Y_1$         | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0  |
| $Y_2$         | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0  |
| $Z$           | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0  |

- a. Compute the maximum likelihood estimate of  $p_1, p_2, q_0, q_1$  and  $r$ .
- b. Now we are given a partial data point with  $X_1^2 = 1, X_2^2 = 1, Y_1 = 1$ . Find the probability that  $Y_2 = 1$  in terms of the parameters  $p_1, p_2, q_0, q_1$  and  $r$  (you might not need all of them).

- a. The maximum likelihood estimate of  $p_1$  is the fraction of counts of samples in which  $X_1 = Y$ . In the given training data, samples 1, 2, 4 and 6 have  $X_1 = Y = 0$  and samples 9 and 10 have  $X_1 = Y = 1$ , so 6 out of the 10 samples have  $X_1 = Y$  and thus  $p_1 = \frac{6}{10} = \frac{3}{5}$ . Analogously, 8 out of the 10 samples have  $X_2 = Y$  and thus  $p_2 = \frac{8}{10} = \frac{4}{5}$ . Similarly, we can find:

$$q_0 = \frac{1}{6}$$

$$q_1 = 1$$

$$r = \frac{2}{5}$$

$$\text{b. } P(Y_2 = 1 | X_1^2 = 1, X_2^2 = 1, Y_1 = 1) = \frac{rq_1^2 + (1-r)q_0^2}{rq_1 + (1-r)q_0}$$

**Exercise 20.2.#BNBX**

Explain how to apply the boosting method of Chapter 19 to naive Bayes learning. Test the performance of the resulting algorithm on the restaurant learning problem.

Boosted naive Bayes learning is discussed by Elkan (1997). The application of boosting to naive Bayes is straightforward. The naive Bayes learner uses maximum-likelihood parameter estimation based on counts, so using a weighted training set simply means adding weights rather than counting. Each naive Bayes model is treated as a deterministic classifier that picks the most likely class for each example.

**Exercise 20.2.#LINR**

Consider  $N$  data points  $(x_j, y_j)$ , where the  $y_j$ s are generated from the  $x_j$ s according to the linear Gaussian model in Equation (20.5). Find the values of  $\theta_1$ ,  $\theta_2$ , and  $\sigma$  that maximize the conditional log likelihood of the data.

We have

$$L = -m(\log \sigma + \log \sqrt{2\pi}) - \sum_j \frac{(y_j - (\theta_1 x_j + \theta_2))^2}{2\sigma^2}$$

hence the equations for the derivatives at the optimum are

$$\begin{aligned}\frac{\partial L}{\partial \theta_1} &= -\sum_j \frac{x_j(y_j - (\theta_1 x_j + \theta_2))}{\sigma^2} = 0 \\ \frac{\partial L}{\partial \theta_2} &= -\sum_j \frac{(y_j - (\theta_1 x_j + \theta_2))}{\sigma^2} = 0 \\ \frac{\partial L}{\partial \sigma} &= -\frac{m}{\sigma} + \sum_j \frac{(y_j - (\theta_1 x_j + \theta_2))^2}{\sigma^3} = 0\end{aligned}$$

and the solutions can be computed as

$$\begin{aligned}\theta_1 &= \frac{m \left( \sum_j x_j y_j \right) - \left( \sum_j y_j \right) \left( \sum_j x_j \right)}{m \left( \sum_j x_j^2 \right) - \left( \sum_j x_j \right)^2} \\ \theta_2 &= \frac{1}{m} \sum_j (y_j - \theta_1 x_j) \\ \sigma^2 &= \frac{1}{m} \sum_j (y_j - (\theta_1 x_j + \theta_2))^2\end{aligned}$$

**Exercise 20.2.#XXXX**

Answer the following True/False questions.

- In the case of a binary class and all binary features, Naive Bayes is a linear classifier. Justify your answer.
- Naive Bayes trained using maximum-likelihood parameter estimation is guaranteed not to perform worse if more features are added.

- $\arg \max_c \Pr(C = c | X_{1:n} = x_{1:n}) = \arg \max_c \log \Pr(C = c, X_{1:n} = x_{1:n})$   
We also know  $\log \Pr(C = c, X_{1:n} = x_{1:n})$  is linear in  $x_{1:n}$ , so this is true.
- False. If additional dependent features are added, accuracy may be worse.

**Exercise 20.2.#MISC.B**

Consider the geometric distribution, which has  $P(X = k) = (1 - \theta)^{k-1}\theta$ . Assume in our training data  $X$  took on the values 4, 2, 7, and 9.

- Write an expression for the log-likelihood of the data as a function of the parameter  $\theta$ .
  - What is the value of  $\theta$  that maximizes the log-likelihood, i.e., what is the maximum likelihood estimate for  $\theta$ ?
- $$\begin{aligned} L(\theta) &= P(X = 4)P(X = 2)P(X = 7)P(X = 9) \\ &= (1 - \theta)^3\theta(1 - \theta)^1\theta(1 - \theta)^6\theta(1 - \theta)^8\theta \\ &= (1 - \theta)^{18}\theta^4 \\ \log L(\theta) &= 18 \log(1 - \theta) + 4 \log \theta \end{aligned}$$
  - We take a derivative of the log likelihood equation to find the MLE for theta,  $\theta^{ML}$ :
    - At the maximum we have:  $\frac{\partial \log L(\theta)}{\partial \theta} = 18\left(\frac{-1}{1-\theta}\right) + 4\frac{1}{\theta} = 0$
    - After multiplying both sides by  $(1 - \theta)\theta$ ,  $-18\theta + 4(1 - \theta) = 0$  and hence we have an extremum at  $\theta = \frac{4}{22}$
    - Also:  $\frac{\partial^2 \log L(\theta)}{\partial \theta^2} = \frac{-18}{(1-\theta)^2} - \frac{4}{\theta^2} < 0$  hence extremum is indeed a maximum, and hence  $\theta^{ML} = \frac{4}{22}$ .

**Exercise 20.2.#XXXX**

Identical twins are rare, but just how *unlikely* are they? With the help of the sociology department, you have a representative sample of twins to help you answer the question. The twins data gives the following observations (a twin refers to one pair of two people):

- $m_i$  = number of identical male twins and  $f_i$  = number of identical female twins
- $m_f$  = number of fraternal male twins and  $f_f$  = number of fraternal female twins
- $b$  = number of fraternal opposite gender twins

To model this data, we choose these distributions and parameters:

- Twins are identical with probability  $\theta$ .
  - Given identical twins, the twins are male with probability  $p$ .
  - Given fraternal twins, the probability of male twins is  $q^2$ , probability of female twins is  $(1 - q)^2$  and probability of opposite gender twins is  $2q(1 - q)$ .
- a. Write expressions for the likelihood of the data as a function of the parameters  $\theta, p$ , and  $q$  for the observations  $m_i, f_i, m_f, f_f, b$ .
- b. What are the maximum likelihood estimates for  $\theta, p$  and  $q$ ?

- a. We gather the following probabilities from the definitions in the problem:

- identical male twins:  $\theta p$
- identical female twins:  $\theta(1 - p)$
- fraternal male twin:  $(1 - \theta)q^2$
- fraternal female twins:  $(1 - \theta)(1 - q)^2$
- fraternal opposite gender twins:  $(1 - \theta) \cdot 2q(1 - q)$ .

Therefore, the likelihood is

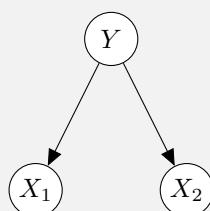
$$\begin{aligned} L(\theta, p, q) &= (\theta p)^{m_i} (\theta(1 - p))^{f_i} ((1 - \theta)q^2)^{m_f} ((1 - \theta)(1 - q)^2)^{f_f} ((1 - \theta)2q(1 - q))^b \\ &= \theta^{(m_i + f_i)} (1 - \theta)^{(m_f + f_f + b)} p^{m_i} (1 - p)^{f_i} q^{2m_f} (1 - q)^{2f_f} (2q(1 - q))^b \end{aligned}$$

- b. To get the maximum likelihood estimate, we have to maximize the log likelihood by taking partial derivatives:

$$\begin{aligned} \frac{\partial l}{\partial \theta} &= \frac{m_i + f_i}{\theta} - \frac{m_f + f_f + b}{1 - \theta} = 0 & \theta_{\text{ML}} &= \frac{m_i + f_i}{m_i + f_i + m_f + f_f + b} \\ \frac{\partial l}{\partial p} &= \frac{m_i}{p} - \frac{m_i + f_i}{1 - p} = 0 & p_{\text{ML}} &= \frac{m_i}{m_i + f_i} \\ \frac{\partial l}{\partial q} &= \frac{2m_f + b}{q} - \frac{2f_f + b}{1 - q} = 0 & q_{\text{ML}} &= \frac{2m_f + b}{2m_f + 2f_f + 2b} \end{aligned}$$

### Exercise 20.2.#XXXX

Consider a naive Bayes classifier with two features, shown below. We have prior information that the probability model can be parameterized by  $\lambda$  and  $p$ , as shown below: Note that  $P(X_1 = 0|Y = 0) = P(X_1 = 1|Y = 1) = p$  and  $P(X_1|Y) = P(X_2|Y)$  (they share the parameter  $p$ ). Call this model M1.



| $Y$ | $P(Y)$        |
|-----|---------------|
| 0   | $\lambda$     |
| 1   | $1 - \lambda$ |

| $X_1$ | $Y$ | $P(X_1 Y)$ |
|-------|-----|------------|
| 0     | 0   | $p$        |
| 1     | 0   | $1 - p$    |
| 0     | 1   | $1 - p$    |
| 1     | 1   | $p$        |

| $X_2$ | $Y$ | $P(X_2 Y)$ |
|-------|-----|------------|
| 0     | 0   | $p$        |
| 1     | 0   | $1 - p$    |
| 0     | 1   | $1 - p$    |
| 1     | 1   | $p$        |

We have a training set that contains all of the following:

- $n_{000}$  examples with  $X_1 = 0, X_2 = 0, Y = 0$
- $n_{010}$  examples with  $X_1 = 0, X_2 = 1, Y = 0$
- $n_{100}$  examples with  $X_1 = 1, X_2 = 0, Y = 0$
- $n_{110}$  examples with  $X_1 = 1, X_2 = 1, Y = 0$
- $n_{001}$  examples with  $X_1 = 0, X_2 = 0, Y = 1$
- $n_{011}$  examples with  $X_1 = 0, X_2 = 1, Y = 1$
- $n_{101}$  examples with  $X_1 = 1, X_2 = 0, Y = 1$
- $n_{111}$  examples with  $X_1 = 1, X_2 = 1, Y = 1$

- Solve for the maximum likelihood estimate (MLE) of the parameter  $p$  with respect to  $n_{000}, n_{100}, n_{010}, n_{110}, n_{001}, n_{101}, n_{011}$ , and  $n_{111}$ .
- For each of the following values of  $\lambda, p, X_1$ , and  $X_2$ , classify the value of  $Y$ 
  - $(\lambda = \frac{3}{4}, p = \frac{5}{8}, X_1 = 0, X_2 = 0)$
  - $(\lambda = \frac{3}{5}, p = \frac{3}{7}, X_1 = 0, X_2 = 0)$
- Now let's consider a new model M2, which has the same Bayes' Net structure as M1, but where we have a  $p_1$  value for  $P(X_1 = 0 | Y = 0) = P(X_1 = 1 | Y = 1) = p_1$  and a separate  $p_2$  value for  $P(X_2 = 0 | Y = 0) = P(X_2 = 1 | Y = 1) = p_2$ , and we don't constrain  $p_1 = p_2$ . Let  $L_{M1}$  be the likelihood of the training data under model M1 with the maximum likelihood parameters for M1. Let  $L_{M2}$  be the likelihood of the training data under model M2 with the maximum likelihood parameters for M2. Are we guaranteed to have  $L_{M1} \leq L_{M2}$ ?

- We first write down the likelihood of the training data,  $T = (Y, X_1, X_2)$ .

$$\begin{aligned}
 L &= \prod_{(y_i, x_{1i}, x_{2i})} P(Y = y_i, X_1 = x_{1i}, X_2 = x_{2i} | p, \lambda) \\
 &= \prod_{(y_i, x_{1i}, x_{2i})} P(Y = y_i | p, \lambda) P(X_1 = x_{1i} | p, \lambda) P(X_2 = x_{2i} | p, \lambda) \\
 &= \left( \prod_1^{n_{000}} \lambda pp \right) \left( \prod_1^{n_{001}} (1 - \lambda)(1 - p)(1 - p) \right) \left( \prod_1^{n_{010}} \lambda p(1 - p) \right) \left( \prod_1^{n_{011}} (1 - \lambda)(1 - p)p \right) \\
 &\quad * \left( \prod_1^{n_{100}} \lambda(1 - p)p \right) \left( \prod_1^{n_{101}} (1 - \lambda)p(1 - p) \right) \left( \prod_1^{n_{110}} \lambda(1 - p)(1 - p) \right) \left( \prod_1^{n_{111}} (1 - \lambda)pp \right) \\
 &= (\lambda^{n_{000}+n_{010}+n_{100}+n_{110}})((1 - \lambda)^{n_{001}+n_{011}+n_{101}+n_{111}})(p^{n_{000}+n_{010}+n_{101}+n_{111}}) \\
 &\quad * (p^{n_{000}+n_{011}+n_{100}+n_{111}})((1 - p)^{n_{001}+n_{011}+n_{100}+n_{110}})((1 - p)^{n_{001}+n_{010}+n_{101}+n_{110}}) \\
 &= (\lambda^{n_{000}+n_{010}+n_{100}+n_{110}})((1 - \lambda)^{n_{001}+n_{011}+n_{101}+n_{111}}) \\
 &\quad * (p^{2n_{000}+n_{010}+n_{011}+n_{100}+n_{101}+2n_{111}})((1 - p)^{2n_{001}+n_{010}+n_{011}+n_{100}+n_{101}+2n_{110}})
 \end{aligned}$$

We then take the logarithm of the likelihood.

$$\begin{aligned}\log(L) &= (n_{000} + n_{010} + n_{100} + n_{110}) \log(\lambda) \\ &\quad + (n_{001} + n_{011} + n_{101} + n_{111}) \log(1 - \lambda)(2n_{000} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{111}) \log(p) \\ &\quad + (2n_{001} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{110}) \log(1 - p)\end{aligned}$$

We want to take the partial derivative with respect to  $p$  and solve for when it is 0 to find the MLE estimate of  $p$ . When we do this, the first two terms only depend on  $\lambda$  and not  $p$ , so their partial derivative is 0.

$$\begin{aligned}0 &= \frac{\partial}{\partial p}(\log(L)) \\ &= (2n_{000} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{111}) \frac{1}{p} \\ &\quad - (2n_{001} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{110}) \frac{1}{1-p}\end{aligned}$$

Multiplying both sides by  $(p)(1 - p)$ , we have:

$$\begin{aligned}0 &= (2n_{000} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{111})(1 - p) \\ &\quad - (2n_{001} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{110})p\end{aligned}$$

and simplifying:

$$\begin{aligned}&(2n_{000} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{111}) \\ &= 2(n_{000} + n_{001} + n_{010} + n_{011} + n_{100} + n_{101} + n_{110} + n_{111})p\end{aligned}$$

so

$$p = \frac{2n_{000} + n_{010} + n_{011} + n_{100} + n_{101} + 2n_{111}}{2(n_{000} + n_{001} + n_{010} + n_{011} + n_{100} + n_{101} + n_{110} + n_{111})}$$

- b. (i) For the first case,  $P(Y = 0, X_1 = 0, X_2 = 0) = \lambda pp$  and  $P(Y = 1, X_1 = 0, X_2 = 0) = (1 - \lambda)(1 - p)(1 - p)$ . Since  $\lambda > (1 - \lambda)$  and  $p > (1 - p)$ , we must have  $P(Y = 0, X_1 = 0, X_2 = 0) > P(Y = 1, X_1 = 0, X_2 = 0)$ .

For the second case, we have  $P(Y = 0, X_1 = 1, X_2 = 0) = \lambda(1 - p)p$  and  $P(Y = 1, X_1 = 1, X_2 = 0) = (1 - \lambda)p(1 - p)$ . Since both expressions have a  $p(1 - p)$  term, the question is reduced to  $\lambda < (1 - \lambda)$  so  $P(Y = 1, X_1 = 1, X_2 = 0) > P(Y = 0, X_1 = 1, X_2 = 0)$ .

Thus the predicted label is  $Y = 1$ .

(ii)  $P(Y = 0, X_1 = 0, X_2 = 0) = \lambda pp = \frac{3}{5} \cdot \frac{3}{7} \cdot \frac{3}{7} = \frac{27}{5 \cdot 7 \cdot 7}$

We also know  $P(Y = 1, X_1 = 0, X_2 = 0) = (1 - \lambda)(1 - p)(1 - p) = \frac{2}{5} \cdot \frac{4}{7} \cdot \frac{4}{7} = \frac{32}{5 \cdot 7 \cdot 7}$ .

## Exercises 20 Learning Probabilistic Models

So  $\frac{27}{5*7*7} = P(Y = 0, X_1 = 0, X_2 = 0) < P(Y = 1, X_1 = 0, X_2 = 0) = \frac{32}{5*7*7}$ .

Thus the predicted label is  $Y = 1$ .

- c.  $M_2$  can represent all of the same probability distributions that  $M_1$  can, but also some more (when  $p_1 \neq p_2$ ). So in general  $M_2$  allows for more fitting of the training data, which results in a higher likelihood.

### Exercise 20.2.#NORX

Consider the noisy-OR model for fever described in Section 13.2.2. Explain how to apply maximum-likelihood learning to fit the parameters of such a model to a set of complete data. (*Hint:* use the chain rule for partial derivatives.)

There are a couple of ways to solve this problem. Here, we show the indicator variable method described on page 743. Assume we have a child variable  $Y$  with parents  $X_1, \dots, X_k$  and let the range of each variable be  $\{0, 1\}$ . Let the noisy-OR parameters be  $q_i = P(Y = 0 | X_i = 1, X_{-i} = 0)$ . The noisy-OR model then asserts that

$$P(Y = 1 | x_1, \dots, x_k) = 1 - \prod_{i=1}^k q_i^{x_i}.$$

Assume we have  $m$  complete-data samples with values  $y_j$  for  $Y$  and  $x_{ij}$  for each  $X_i$ . The conditional log likelihood for  $P(Y | X_1, \dots, X_k)$  is given by

$$\begin{aligned} L &= \sum_j \log \left( 1 - \prod_i q_i^{x_{ij}} \right)^{y_j} \left( \prod_i q_i^{x_{ij}} \right)^{1-y_j} \\ &= \sum_j y_j \log \left( 1 - \prod_i q_i^{x_{ij}} \right) + (1 - y_j) \sum_i x_{ij} \log q_i \end{aligned}$$

The gradient with respect to each noisy-OR parameter is

$$\begin{aligned} \frac{\partial L}{\partial q_i} &= \sum_j -\frac{y_j x_{ij} \prod_i q_i^{x_{ij}}}{q_i \left( 1 - \prod_i q_i^{x_{ij}} \right)} + \frac{(1 - y_j) x_{ij}}{q_i} \\ &= \sum_j \frac{x_{ij} \left( 1 - y_j - \prod_i q_i^{x_{ij}} \right)}{q_i \left( 1 - \prod_i q_i^{x_{ij}} \right)} \end{aligned}$$

### Exercise 20.2.#BETI

This exercise investigates properties of the Beta distribution defined in Equation (20.6).

- a. By integrating over the range  $[0, 1]$ , show that the normalization constant for the distribution  $Beta(\theta; a, b)$  is given by  $\alpha = \Gamma(a + b)/\Gamma(a)\Gamma(b)$  where  $\Gamma(x)$  is the **Gamma function**, defined by  $\Gamma(x + 1) = x \cdot \Gamma(x)$  and  $\Gamma(1) = 1$ . (For integer  $x$ ,  $\Gamma(x + 1) = x!$ .)

- b. Show that the mean is  $a/(a + b)$ .
- c. Find the mode(s) (the most likely value(s) of  $\theta$ ).
- d. Describe the distribution  $Beta(\cdot; \epsilon, \epsilon)$  for very small  $\epsilon$ . What happens as such a distribution is updated?

- a. We will solve this for positive integer  $a$  and  $b$  by induction over  $a$ . Let  $\alpha(a, b)$  be the normalization constant. For the base cases, we have

$$\alpha(1, b) = 1 / \int_0^1 \theta^0 (1 - \theta)^{b-1} d\theta = -1 / [\frac{1}{b} (1 - \theta)^b]_0^1 = b$$

and

$$\frac{\Gamma(1+b)}{\Gamma(1)\Gamma(b)} = \frac{b \cdot \Gamma(b)}{1 \cdot \Gamma(b)} = b.$$

For the inductive step, we assume for all  $b$  that

$$\alpha(a-1, b+1) = \frac{\Gamma(a+b)}{\Gamma(a-1)\Gamma(b+1)} = \frac{a-1}{b} \cdot \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$

Now we evaluate  $\alpha(a, b)$  using integration by parts. We have

$$\begin{aligned} 1/\alpha(a, b) &= \int_0^1 \theta^{a-1} (1 - \theta)^{b-1} d\theta \\ &= [\theta^{a-1} \cdot \frac{1}{b} (1 - \theta)^b]_0^1 + \frac{a-1}{b} \int_0^1 \theta^{a-2} (1 - \theta)^b d\theta \\ &= 0 + \frac{a-1}{b} \frac{1}{\alpha(a-1, b+1)} \end{aligned}$$

Hence

$$\alpha(a, b) = \frac{b}{a-1} \alpha(a-1, b+1) = \frac{b}{a-1} \frac{a-1}{b} \cdot \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$

as required.

- b. The mean is given by the following integral:

$$\begin{aligned} \mu(a, b) &= \alpha(a, b) \int_0^1 \theta \cdot \theta^{a-1} (1 - \theta)^{b-1} d\theta \\ &= \alpha(a, b) \int_0^1 \theta^a (1 - \theta)^{b-1} d\theta \\ &= \alpha(a, b) / \alpha(a+1, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \cdot \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(a+b+1)} \\ &= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \cdot \frac{a\Gamma(a)\Gamma(b)}{(a+b)\Gamma(a+b+1)} = \frac{a}{a+b}. \end{aligned}$$

- c. The mode is found by solving for  $d\text{Beta}(\theta; a, b)/d\theta = 0$ :

$$\begin{aligned} \frac{d}{d\theta}(\alpha(a, b)\theta^{a-1}(1-\theta)^{b-1}) \\ = \alpha(a, b)[(a-1)\theta^{a-2}(1-\theta)^{b-1} - (b-1)\theta^{a-1}(1-\theta)^{b-2}] = 0 \\ \Rightarrow (a-1)(1-\theta) = (b-1)\theta \\ \Rightarrow \theta = \frac{a-1}{a+b-2} \end{aligned}$$

- d.  $\text{Beta}(\theta; \epsilon, \epsilon) = \alpha(\epsilon, \epsilon)\theta^{\epsilon-1}(1-\theta)^{\epsilon-1}$  tends to very large values close to  $\theta = 0$  and  $\theta = 1$ , i.e., it expresses the prior belief that the distribution characterized by  $\theta$  is nearly deterministic (either positively or negatively). After updating with a positive example we obtain the distribution  $\text{Beta}(\cdot; 1 + \epsilon, \epsilon)$ , which has nearly all its mass near  $\theta = 1$  (and the converse for a negative example), i.e., we have learned that the distribution characterized by  $\theta$  is deterministic in the positive sense. If we see a “counterexample”, e.g., a positive and a negative example, we obtain  $\text{Beta}(\theta; 1 + \epsilon, 1 + \epsilon)$ , which is close to uniform, i.e., the hypothesis of near-determinism is abandoned.

### Exercise 20.2.#MLPA

Consider an arbitrary Bayesian network, a complete data set for that network, and the likelihood for the data set according to the network. Give a simple proof that the likelihood of the data cannot decrease if we add a new link to the network and recompute the maximum-likelihood parameter values.

Consider the maximum-likelihood parameter values for the CPT of node  $Y$  in the original network, where an extra parent  $X_{k+1}$  will be added to  $Y$ . If we set the parameters for  $P(y|x_1, \dots, x_k, x_{k+1})$  in the new network to be identical to  $P(y|x_1, \dots, x_k)$  in the original network, regardless of the value  $x_{k+1}$ , then the likelihood of the data is unchanged. Maximizing the likelihood by altering the parameters can then only *increase* the likelihood.

### Exercise 20.2.#LIKIR

Consider a single Boolean random variable  $Y$  (the “classification”). Let the prior probability  $P(Y = \text{true})$  be  $\pi$ . Let’s try to find  $\pi$ , given a training set  $D = (y_1, \dots, y_N)$  with  $N$  independent samples of  $Y$ . Furthermore, suppose  $p$  of the  $N$  are positive and  $n$  of the  $N$  are negative.

- Write down an expression for the likelihood of  $D$  (i.e., the probability of seeing this particular sequence of examples, given a fixed value of  $\pi$ ) in terms of  $\pi$ ,  $p$ , and  $n$ .
- By differentiating the log likelihood  $L$ , find the value of  $\pi$  that maximizes the likelihood.
- Now suppose we add in  $k$  Boolean random variables  $X_1, X_2, \dots, X_k$  (the “attributes”) that describe each sample, and suppose we assume that the attributes are conditionally independent of each other given the goal  $Y$ . Draw the Bayes net corresponding to this assumption.

- d. Write down the likelihood for the data including the attributes, using the following additional notation:

- $\alpha_i$  is  $P(X_i = \text{true}|Y = \text{true})$ .
- $\beta_i$  is  $P(X_i = \text{true}|Y = \text{false})$ .
- $p_i^+$  is the count of samples for which  $X_i = \text{true}$  and  $Y = \text{true}$ .
- $n_i^+$  is the count of samples for which  $X_i = \text{false}$  and  $Y = \text{true}$ .
- $p_i^-$  is the count of samples for which  $X_i = \text{true}$  and  $Y = \text{false}$ .
- $n_i^-$  is the count of samples for which  $X_i = \text{false}$  and  $Y = \text{false}$ .

[Hint: consider first the probability of seeing a single example with specified values for  $X_1, X_2, \dots, X_k$  and  $Y$ .]

- e. By differentiating the log likelihood  $L$ , find the values of  $\alpha_i$  and  $\beta_i$  (in terms of the various counts) that maximize the likelihood and say in words what these values represent.
- f. Let  $k = 2$ , and consider a data set with 4 all four possible examples of the XOR function. Compute the maximum likelihood estimates of  $\pi, \alpha_1, \alpha_2, \beta_1$ , and  $\beta_2$ .
- g. Given these estimates of  $\pi, \alpha_1, \alpha_2, \beta_1$ , and  $\beta_2$ , what are the posterior probabilities  $P(Y = \text{true}|x_1, x_2)$  for each example?

- a. The probability of a positive example is  $\pi$  and of a negative example is  $(1 - \pi)$ , and the data are independent, so the probability of the data is  $\pi^p(1 - \pi)^n$
- b. We have  $L = p \log \pi + n \log(1 - \pi)$ ; if the derivative is zero, we have

$$\frac{\partial L}{\partial \pi} = \frac{p}{\pi} - \frac{n}{1 - \pi} = 0$$

so the ML value is  $\pi = p/(p + n)$ , i.e., the proportion of positive examples in the data.

- c. This is the “naive Bayes” probability model.

- d. The likelihood of a single instance is a product of terms. For a positive example,  $\pi$  times  $\alpha_i$  for each true attribute and  $(1 - \alpha_i)$  for each negative attribute; for a negative example,  $(1 - \pi)$  times  $\beta_i$  for each true attribute and  $(1 - \beta_i)$  for each negative attribute. Over the whole data set, the likelihood is  $\pi^p(1 - \pi)^n \prod_i \alpha_i^{p_i^+} (1 - \alpha_i)^{n_i^+} \beta_i^{p_i^-} (1 - \beta_i)^{n_i^-}$ .

- e. The log likelihood is

$$L = p \log \pi + n \log(1 - \pi) + \sum_i p_i^+ \log \alpha_i + n_i^+ \log(1 - \alpha_i) + p_i^- \log \beta_i + n_i^- \log(1 - \beta_i).$$

Setting the derivatives w.r.t.  $\alpha_i$  and  $\beta_i$  to zero, we have

$$\frac{\partial L}{\partial \alpha_i} = \frac{p_i^+}{\alpha_i} - \frac{n_i^+}{1 - \alpha_i} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \beta_i} = \frac{p_i^-}{\beta_i} - \frac{n_i^-}{1 - \beta_i} = 0$$

giving  $\alpha_i = p_i^+/(p_i^+ + n_i^+)$ , i.e., the fraction of cases where  $X_i$  is true given  $Y$  is true, and  $\beta_i = p_i^-/(p_i^- + n_i^-)$ , i.e., the fraction of cases where  $X_i$  is true given  $Y$  is false.

- f. In the data set we have  $p=2$ ,  $n=2$ ,  $p_i^+=1$ ,  $n_i^+=1$ ,  $p_i^-=1$ ,  $n_i^-=1$ . From our formulæ, we obtain  $\pi=\alpha_1=\alpha_2=\beta_1=\beta_2=0.5$ .
- g. Each example is predicted to be positive with probability 0.5.

## 20.3 Learning with Hidden Variables: The EM Algorithm

### Exercise 20.3.#EMLE

Consider the application of EM to learn the parameters for the network in Figure 20.14(a), given the true parameters in Equation (20.9).

- a. Explain why the EM algorithm would not work if there were just two attributes in the model rather than three.
- b. Show the calculations for the first iteration of EM starting from Equation (20.10).
- c. What happens if we start with all the parameters set to the same value  $p$ ? (*Hint:* you may find it helpful to investigate this empirically before deriving the general result.)
- d. Write out an expression for the log likelihood of the tabulated candy data on page 742 in terms of the parameters, calculate the partial derivatives with respect to each parameter, and investigate the nature of the fixed point reached in part (c).

- a. Consider the ideal case in which the bags were infinitely large so there is no statistical fluctuation in the sample. With two attributes (say, *Flavor* and *Wrapper*), we have five unknowns:  $\theta$  gives the relative sizes of the bags,  $\theta_{F1}$  and  $\theta_{F2}$  give the proportion of cherry candies in each bag, and  $\theta_{W1}$  and  $\theta_{W2}$  give the proportion of red wrappers in each bag. In the data, we observe just the flavor and wrapper for each candy; there are four combinations, so three independent numbers can be obtained. This is not enough to recover five unknowns. With three attributes, there are eight combinations and seven numbers can be obtained, enough to recover the seven parameters.
- b. The computation for  $\theta^{(1)}$  has eight nearly identical expressions and calculations, one of which is shown. The symbolic expression for  $\theta_{F1}^{(1)}$  is shown, but not its evaluation; it would be reasonable to ask students to write out the expression in terms of the parameters, as was done for  $\theta^{(1)}$ , and calculate the value. The final answers are given in the chapter.
- c. Consider the contribution to the update for  $\theta$  from the 273 red-wrapped cherry candies with holes:

$$\frac{273}{1000} \cdot \frac{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)}}{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)} + \theta_{F2}^{(0)}\theta_{W2}^{(0)}\theta_{H2}^{(0)}(1 - \theta^{(0)})}$$

If all of the seven named parameters have value  $p$ , this reduces to

$$\frac{273}{1000} \cdot \frac{p^4}{p^4 + p^3(1-p)} = \frac{273p}{1000}$$

with similar results for the other candy categories. Thus, the new value for  $\theta^{(1)}$  just ends up being  $1000p/1000 = p$ .

We can check the expression for  $\theta_{F1}$  too; for example, the 273 red-wrapped cherry candies with holes contribute an expected count of

$$273P(Bag = 1 | Flavor_j = cherry, Wrapper = red, Holes = 1) \\ = 273 \frac{\theta_{F1}\theta_{W1}\theta_{H1}\theta}{\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1-\theta)} = 273p$$

and the 90 green-wrapped cherry candies with no holes contribute an expected count of

$$90P(Bag = 1 | Flavor_j = cherry, Wrapper = green, Holes = 0) \\ = 90 \frac{\theta_{F1}(1 - \theta_{W1})(1 - \theta_{H1})\theta}{\theta_{F1}(1 - \theta_{W1})(1 - \theta_{H1})\theta + \theta_{F2}(1 - \theta_{W2})(1 - \theta_{H2})(1 - \theta)} \\ = 90p^2(1 - p)^2/p(1 - p)^2 = 90p.$$

Continuing, we find that the new value for  $\theta_{F1}$  is  $560p/1000p = 0.56$ , the proportion of cherry candies in the entire sample.

For  $\theta_{F2}$ , the 273 red-wrapped cherry candies with holes contribute an expected count of

$$273P(Bag = 2 | Flavor_j = cherry, Wrapper = red, Holes = 1) \\ = 273 \frac{\theta_{F2}\theta_{W2}\theta_{H2}(1 - \theta)}{\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1 - \theta)} = 273(1 - p)$$

with similar contributions from the other cherry categories, so the new value is  $560(1 - p)/1000(1 - p) = 0.56$ , as for  $\theta_{F1}$ . Similarly,  $\theta_{W1}^{(1)} = \theta_{W2}^{(1)} = 0.545$ , the proportion of red wrappers in the sample, and  $\theta_{H1}^{(1)} = \theta_{H2}^{(1)} = 0.550$ , the proportion of candies with holes in the sample.

Intuitively, this makes sense: because the bag label is invisible, labels 1 and 2 are *a priori* indistinguishable; initializing all the conditional parameters to the same value (regardless of the bag) provides no means of breaking the symmetry. Thus, the symmetry remains.

On the next iteration, we no longer have all the parameters set to  $p$ , but we do know that, for example,

$$\theta_{F1}\theta_{W1}\theta_{H1} = \theta_{F2}\theta_{W2}\theta_{H2}$$

so those terms cancel top and bottom in the expression for the contribution of the 273 candies to  $\theta_{F1}$ , and once again the contribution is 273p.

To cut a long story short, all the parameters remain fixed after the first iteration, with  $\theta$  at its initial value  $p$  and the other parameters at the corresponding empirical frequencies as indicated above.

- d. This part takes some time but makes the abstract mathematical expressions in the chap-

ter very concrete! The one concession to abstraction will be the use of symbols for the empirical counts, e.g.,

$$N_{cr1} = N(Flavor = cherry, Wrapper = red, Holes = 1) = 273 .$$

with marginal counts  $N_c$ ,  $N_{r1}$ , etc. Thus we have  $\theta_{F1}^{(1)} = N_c/N = 560/1000$ .

The log likelihood is given by

$$\begin{aligned} L(\mathbf{d}) &= \log P(\mathbf{d}) = \log \prod_j P(d_j) = \sum_j \log P(d_j) \\ &= N_{cr1} \log P(F = cherry, W = red, H = 1) + \\ &\quad N_{lr1} \log P(F = lime, W = red, H = 1) + \\ &\quad N_{cr0} \log P(F = cherry, W = red, H = 0) + \\ &\quad N_{lr0} \log P(F = lime, W = red, H = 0) + \\ &\quad N_{cg1} \log P(F = cherry, W = green, H = 1) + \\ &\quad N_{lg1} \log P(F = lime, W = green, H = 1) + \\ &\quad N_{cg0} \log P(F = cherry, W = green, H = 0) + \\ &\quad N_{lg0} \log P(F = lime, W = green, H = 0) \end{aligned}$$

Each of these probabilities can be expressed in terms of the network parameters, giving the following expression for  $L(\mathbf{d})$ :

$$\begin{aligned} &N_{cr1} \log(\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1 - \theta)) + \\ &N_{lr1} \log((1 - \theta_{F1})\theta_{W1}\theta_{H1}\theta + (1 - \theta_{F2})\theta_{W2}\theta_{H2}(1 - \theta)) + \\ &N_{cr0} \log(\theta_{F1}\theta_{W1}(1 - \theta_{H1})\theta + \theta_{F2}\theta_{W2}(1 - \theta_{H2})(1 - \theta)) + \\ &N_{lr0} \log((1 - \theta_{F1})\theta_{W1}(1 - \theta_{H1})\theta + (1 - \theta_{F2})\theta_{W2}(1 - \theta_{H2})(1 - \theta)) + \\ &N_{cg1} \log(\theta_{F1}(1 - \theta_{W1})\theta_{H1}\theta + \theta_{F2}(1 - \theta_{W2})\theta_{H2}(1 - \theta)) + \\ &N_{lg1} \log((1 - \theta_{F1})(1 - \theta_{W1})\theta_{H1}\theta + (1 - \theta_{F2})(1 - \theta_{W2})\theta_{H2}(1 - \theta)) + \\ &N_{cg0} \log(\theta_{F1}(1 - \theta_{W1})(1 - \theta_{H1})\theta + \theta_{F2}(1 - \theta_{W2})(1 - \theta_{H2})(1 - \theta)) + \\ &N_{lg0} \log((1 - \theta_{F1})(1 - \theta_{W1})(1 - \theta_{H1})\theta + (1 - \theta_{F2})(1 - \theta_{W2})(1 - \theta_{H2})(1 - \theta)) \end{aligned}$$

Hence  $\partial L/\partial\theta$  is given by

$$\begin{aligned}
 & N_{cr1} \frac{\theta_{F1}\theta_{W1}\theta_{H1} - \theta_{F2}\theta_{W2}\theta_{H2}}{\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1-\theta)} \\
 & - N_{lr1} \frac{(1-\theta_{F1})\theta_{W1}\theta_{H1} - (1-\theta_{F2})\theta_{W2}\theta_{H2}}{(1-\theta_{F1})\theta_{W1}\theta_{H1}\theta + (1-\theta_{F2})\theta_{W2}\theta_{H2}(1-\theta)} \\
 & + N_{cr0} \frac{\theta_{F1}\theta_{W1}(1-\theta_{H1}) - \theta_{F2}\theta_{W2}(1-\theta_{H2})}{\theta_{F1}\theta_{W1}(1-\theta_{H1})\theta + \theta_{F2}\theta_{W2}(1-\theta_{H2})(1-\theta)} \\
 & - N_{lr0} \frac{(1-\theta_{F1})\theta_{W1}(1-\theta_{H1}) - (1-\theta_{F2})\theta_{W2}(1-\theta_{H2})}{(1-\theta_{F1})\theta_{W1}(1-\theta_{H1})\theta + (1-\theta_{F2})\theta_{W2}(1-\theta_{H2})(1-\theta)} \\
 & + N_{cg1} \frac{\theta_{F1}(1-\theta_{W1})\theta_{H1} - \theta_{F2}(1-\theta_{W2})\theta_{H2}}{\theta_{F1}(1-\theta_{W1})\theta_{H1}\theta + \theta_{F2}(1-\theta_{W2})\theta_{H2}(1-\theta)} \\
 & - N_{lg1} \frac{(1-\theta_{F1})(1-\theta_{W1})\theta_{H1} - (1-\theta_{F2})(1-\theta_{W2})\theta_{H2}}{(1-\theta_{F1})(1-\theta_{W1})\theta_{H1}\theta + (1-\theta_{F2})(1-\theta_{W2})\theta_{H2}(1-\theta)} \\
 & + N_{cg0} \frac{\theta_{F1}(1-\theta_{W1})(1-\theta_{H1}) - \theta_{F2}(1-\theta_{W2})(1-\theta_{H2})}{\theta_{F1}(1-\theta_{W1})(1-\theta_{H1})\theta + \theta_{F2}(1-\theta_{W2})(1-\theta_{H2})(1-\theta)} \\
 & - N_{lg0} \frac{(1-\theta_{F1})(1-\theta_{W1})(1-\theta_{H1}) - (1-\theta_{F2})(1-\theta_{W2})(1-\theta_{H2})}{(1-\theta_{F1})(1-\theta_{W1})(1-\theta_{H1})\theta + (1-\theta_{F2})(1-\theta_{W2})(1-\theta_{H2})(1-\theta)}
 \end{aligned}$$

By inspection, we can see that whenever  $\theta_{F1} = \theta_{F2}$ ,  $\theta_{W1} = \theta_{W2}$ , and  $\theta_{H1} = \theta_{H2}$ , the derivative is identically zero. Moreover, each term in the above expression has the form  $k/f(\theta)$  where  $k$  does not contain  $\theta$  and  $f'(\theta)$  evaluates to zero under these conditions. Thus the second derivative  $\partial^2 L/\partial\theta^2$  is a collection of terms of the form  $-kf'(\theta)/(f(\theta))^2$ , all of which evaluate to zero. In fact, all derivatives evaluate to zero under these conditions, so the likelihood is completely flat with respect to  $\theta$  in the subspace defined by  $\theta_{F1} = \theta_{F2}$ ,  $\theta_{W1} = \theta_{W2}$ , and  $\theta_{H1} = \theta_{H2}$ . Another way to see this is to note that, in this subspace, the terms within the logs in the expression for  $L(\mathbf{d})$  simplify to terms of the form  $\phi_F\phi_W\phi_H\theta + \phi_F\phi_W\phi_H(1-\theta) = \phi_F\phi_W\phi_H$ , so that the likelihood is in fact independent of  $\theta$ !

## Exercises 20 Learning Probabilistic Models

A representative partial derivative  $\partial L / \partial \theta_{F1}$  is given by

$$\begin{aligned}
& N_{cr1} \frac{\theta_{W1}\theta_{H1}\theta}{\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1-\theta)} \\
& - N_{lr1} \frac{\theta_{W1}\theta_{H1}\theta}{(1-\theta_{F1})\theta_{W1}\theta_{H1}\theta + (1-\theta_{F2})\theta_{W2}\theta_{H2}(1-\theta)} \\
& + N_{cr0} \frac{\theta_{W1}(1-\theta_{H1})\theta}{\theta_{F1}\theta_{W1}(1-\theta_{H1})\theta + \theta_{F2}\theta_{W2}(1-\theta_{H2})(1-\theta)} \\
& - N_{lr0} \frac{\theta_{W1}(1-\theta_{H1})\theta}{(1-\theta_{F1})\theta_{W1}(1-\theta_{H1})\theta + (1-\theta_{F2})\theta_{W2}(1-\theta_{H2})(1-\theta)} \\
& + N_{cg1} \frac{(1-\theta_{W1})\theta_{H1}\theta}{\theta_{F1}(1-\theta_{W1})\theta_{H1}\theta + \theta_{F2}(1-\theta_{W2})\theta_{H2}(1-\theta)} \\
& - N_{lg1} \frac{(1-\theta_{W1})\theta_{H1}\theta}{(1-\theta_{F1})(1-\theta_{W1})\theta_{H1}\theta + (1-\theta_{F2})(1-\theta_{W2})\theta_{H2}(1-\theta)} \\
& + N_{cg0} \frac{(1-\theta_{W1})(1-\theta_{H1})\theta}{\theta_{F1}(1-\theta_{W1})(1-\theta_{H1})\theta + \theta_{F2}(1-\theta_{W2})(1-\theta_{H2})(1-\theta)} \\
& - N_{lg0} \frac{(1-\theta_{W1})(1-\theta_{H1})\theta}{(1-\theta_{F1})(1-\theta_{W1})(1-\theta_{H1})\theta + (1-\theta_{F2})(1-\theta_{W2})(1-\theta_{H2})(1-\theta)}
\end{aligned}$$

Unlike the previous case, here the individual terms do not evaluate to zero. Writing  $\theta_{F1} = \theta_{F2} = N_c/N$ , etc., the expression for  $\partial L / \partial \theta_{F1}$  becomes

$$\begin{aligned}
& N_{cr1} \frac{N N_r N_1 \theta}{N_c N_r N_1 \theta + N_c N_r N_1 (1-\theta)} \\
& - N_{lr1} \frac{N N_r N_1 \theta}{(N - N_c) N_r N_1 \theta + (N - N_c) N_r N_1 (1-\theta)} \\
& + N_{cr0} \frac{N N_r (N - N_1) \theta}{N_c N_r (N - N_1) \theta + N_c N_r (N - N_1) (1-\theta)} \\
& - N_{lr0} \frac{N N_r (N - N_1) \theta}{(N - N_c) N_r (N - N_1) \theta + (N - N_c) N_r (N - N_1) (1-\theta)} \\
& + N_{cg1} \frac{N (N - N_r) N_1 \theta}{N_c (N - N_r) N_1 \theta + N_c (N - N_r) N_1 (1-\theta)} \\
& - N_{lg1} \frac{N (N - N_r) N_1 \theta}{(N - N_c) (N - N_r) N_1 \theta + (N - N_c) (N - N_r) N_1 (1-\theta)} \\
& + N_{cg0} \frac{N (N - N_r) (N - N_1) \theta}{N_c (N - N_r) (N - N_1) \theta + N_c (N - N_r) (N - N_1) (1-\theta)} \\
& - N_{lg0} \frac{N (N - N_r) (N - N_1) \theta}{(N - N_c) (N - N_r) (N - N_1) \theta + (N - N_c) (N - N_r) (N - N_1) (1-\theta)}
\end{aligned}$$

This in turn simplifies to

$$\begin{aligned}
\frac{\partial L}{\partial \theta_{F1}} &= \frac{(N_{cr1} + N_{cr0} + N_{cg1} + N_{cg0})N\theta}{N_c} - \frac{(N_{lr1} + N_{lr0} + N_{lg1} + N_{lg0})N\theta}{N - N_c} \\
&= \frac{N_c N \theta}{N_c} - \frac{(N - N_c) N \theta}{N - N_c} = 0.
\end{aligned}$$

Thus, we have a stationary point as expected.

To identify the nature of the stationary point, we need to examine the second derivatives. We will not do this exhaustively, but will note that

$$\begin{aligned}\partial^2 L / \partial \theta_{F1}^2 &= -N_{cr1} \frac{(\theta_{W1}\theta_{H1}\theta)^2}{(\theta_{F1}\theta_{W1}\theta_{H1}\theta + \theta_{F2}\theta_{W2}\theta_{H2}(1-\theta))^2} \\ &\quad - N_{lr1} \frac{(\theta_{W1}\theta_{H1}\theta)^2}{((1-\theta_{F1})\theta_{W1}\theta_{H1}\theta + (1-\theta_{F2})\theta_{W2}\theta_{H2}(1-\theta))^2} \cdots\end{aligned}$$

with all terms negative, suggesting (possibly) a local maximum in the likelihood surface. A full analysis requires evaluating the Hessian matrix of second derivatives and calculating its eigenvalues.

# EXERCISES 21

## DEEP LEARNING

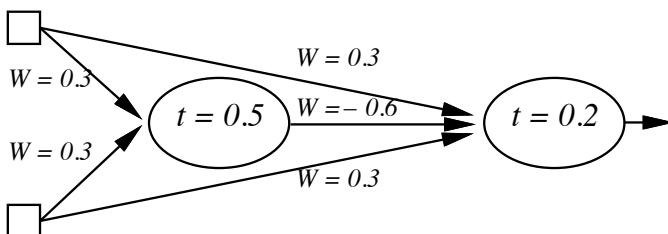
### 21.1 Simple Feedforward Networks

---

#### Exercise 21.1.#NXOR

Construct by hand a neural network that computes the XOR function of two inputs. Make sure to specify what sort of units you are using.

XOR (in fact any Boolean function) is easiest to construct using step-function units. Because XOR is not linearly separable, we will need a hidden layer. It turns out that just one hidden node suffices. To design the network, we can think of the XOR function as OR with the AND case (both inputs on) ruled out. Thus the hidden layer computes AND, while the output layer computes OR but weights the output of the hidden node negatively. The network shown in Figure S21.1 does the trick.



**Figure S21.1** A network of step-function neurons that computes the XOR function.

---

#### Exercise 21.1.#LNNE

Suppose you had a neural network with linear activation functions. That is, for each unit the output is some constant  $c$  times the weighted sum of the inputs.

- Assume that the network has one hidden layer. For a given assignment to the weights  $\mathbf{w}$ , write down equations for the value of the units in the output layer as a function of  $\mathbf{w}$  and the input layer  $\mathbf{x}$ , without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.

- b. Repeat the calculation in part (a), but this time do it for a network with any number of hidden layers.
- c. Suppose a network with one hidden layer and linear activation functions has  $n$  input and output nodes and  $h$  hidden nodes. What effect does the transformation in part (a) to a network with no hidden layers have on the total number of weights? Discuss in particular the case  $h \ll n$ .

This exercise reinforces the student's understanding of neural networks as mathematical functions that can be analyzed at a level of abstraction above their implementation as a network of computing elements. For simplicity, we will assume that the activation function is the same linear function at each node:  $g(x) = cx + d$ . (The argument is the same (only messier) if we allow different  $c_i$  and  $d_i$  for each node.)

- a. The outputs of the hidden layer are

$$H_j = g\left(\sum_k w_{k,j} I_k\right) = c \sum_k w_{k,j} I_k + d$$

The final outputs are

$$O_i = g\left(\sum_j w_{j,i} H_j\right) = c \left( \sum_j w_{j,i} \left( c \sum_k w_{k,j} I_k + d \right) \right) + d$$

Now we just have to see that this is linear in the inputs:

$$O_i = c^2 \sum_k I_k \sum_j w_{k,j} w_{j,i} + d \left( 1 + c \sum_j w_{j,i} \right)$$

Thus we can compute the same function as the two-layer network using just a one-layer perceptron that has weights  $w_{k,i} = \sum_j w_{k,j} w_{j,i}$  and an activation function  $g(x) = c^2 x + d \left( 1 + c \sum_j w_{j,i} \right)$ .

- b. The above reduction can be used straightforwardly to reduce an  $n$ -layer network to an  $(n - 1)$ -layer network. By induction, the  $n$ -layer network can be reduced to a single-layer network. Thus, linear activation functions restrict neural networks to represent only linear functions.
- c. The original network with  $n$  input and output nodes and  $h$  hidden nodes has  $2hn$  weights, whereas the “reduced” network has  $n^2$  weights. When  $h \ll n$ , the original network has far fewer weights and thus represents the i/o mapping more concisely. Such networks are known to learn much faster than the reduced network; so the idea of using linear activation functions is not without merit.

[[need exercises]]

**Exercise 21.1.#188-LOFU**

You would like to train a neural network to classify digits. Your network takes as input an image and outputs probabilities for each of the 10 classes, 0-9. The network's prediction is the class that it assigns the highest probability to. From the following functions, select all that would be suitable loss functions to minimize using gradient descent:

- (A) The square of the difference between the correct digit and the digit predicted by your network
- (B) The probability of the correct digit under your network
- (C) The negative log-probability of the correct digit under your network

C only.

- (A) is incorrect because it is non-differentiable. The correct digit and your model's predicted digit are both integers, and the square of their difference takes on values from the set  $\{0^2, 1^2, \dots, 9^2\}$ . Losses that can be used with gradient descent must take on values from a continuous range and have well-defined gradients.
- (B) is not a loss because you would like to *maximize* the probability of the correct digit under your model, not minimize it.
- (C) is a common loss used for classification tasks. When the probabilities produced by a neural network come from a softmax layer, this loss is often combined with the softmax computation into a single entity known as the “softmax loss” or “softmax cross-entropy loss”.

## 21.2 Computation Graphs for Deep Learning

**Exercise 21.2.#SOGF**

A softmax layer in a neural network takes an input vector  $\mathbf{x}$  and produces an output vector  $\mathbf{y}$ , where

$$y_j = \frac{e^{x_j}}{\sum_{k=1}^d e^{x_k}}.$$

- a. Obtain the derivatives  $\partial y_j / \partial x_i$  in terms of  $x$ -values for the cases  $i = j$  and  $i \neq j$ .
- b. Re-express the derivatives in terms of  $y$ -values.
- c. What can you say about the signs of the derivatives?
- d. Using the indicator function  $\mathbf{1}(i=j)$ , which has value 1 if  $i = j$  and 0 otherwise, combine your two expressions from part (b) into a single expression for the derivative  $\partial y_j / \partial x_i$ .

[[TBC]]

**Exercise 21.2.#SMSG**

A softmax layer in a neural network takes an input vector  $\mathbf{x}$  and produces an output vector  $\mathbf{y}$ , where

$$y_j = \frac{e^{x_j}}{\sum_{k=1^d} e^{x_k}}.$$

Show that the sigmoid function is equivalent to a softmax with  $d = 2$ .

[[TBC]]

**Exercise 21.2.#NNDS**

This exercise asks you to implement the beginnings of a simple deep learning package.

- a. Implement a data structure for general computation graphs, as described in Section 21.1, and define the node types required to support feed-forward neural networks with a variety of activation functions.
- b. Write a function that computes the outputs of the computation graph given the inputs.
- c. Now write a function that computes the error for a labelled example.
- d. Finally, implement the local back-propagation algorithm based on Equations (21.11) and (21.12) and use it to create a stochastic gradient descent learning algorithm for a set of examples.

[[TBC]] The implementation of neural networks can be approached in several different ways. The simplest is probably to store the weights in an  $n \times n$  array. Everything can be calculated as if all the nodes were in each layer, with the zero weights ensuring that only appropriate changes are made as each layer is processed.

Particularly for sparse networks, it can be more efficient to use a pointer-based implementation, with each node represented by a data structure that contains pointers to its successors (for evaluation) and its predecessors (for backpropagation). Weights  $w_{j,i}$  are attached to node  $i$ . In both types of implementation, it is convenient to store the summed input  $in_i = \sum_j w_{j,i} a_j$  and the value  $g'(in_i)$ . The code repository contains an implementation of the pointer-based variety. See the file `learning/algorithms/nn.lisp`, and the function `nn-learning` in that file.

[[need exercises]]

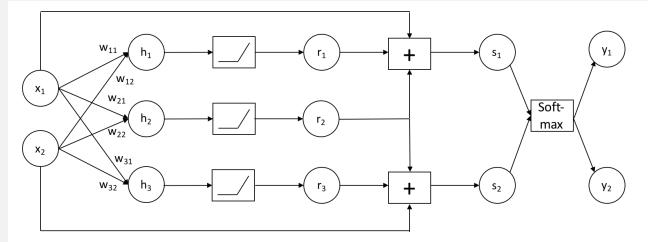
**Exercise 21.2.#188-BAPR**

Origin:

`su19_final_mesut_Backprop_SU19`

The network below is a neural network with inputs  $x_1$  and  $x_2$ , and outputs  $y_1$  and  $y_2$ . The

internal nodes are computed below. All variables are scalar values. Note that  $\text{ReLU}(x) = \max(0, x)$ .



Neural Network

The expressions for the internal nodes in the network are given here for convenience:

$$h_1 = w_{11}x_1 + w_{12}x_2 \quad h_2 = w_{21}x_1 + w_{22}x_2 \quad h_3 = w_{31}x_1 + w_{32}x_2$$

$$r_1 = \text{ReLU}(h_1) \quad r_2 = \text{ReLU}(h_2) \quad r_3 = \text{ReLU}(h_3) \quad s_1 = x_1 + r_1 + r_2$$

$$s_2 = x_2 + r_2 + r_3$$

$$y_1 = \frac{\exp(s_1)}{\exp(s_1) + \exp(s_2)} \quad y_2 = \frac{\exp(s_2)}{\exp(s_1) + \exp(s_2)}$$

### a. Forward Propagation

Suppose for this part only,  $x_1 = 3, x_2 = 5, w_{11} = -10, w_{12} = 7, w_{21} = 2, w_{22} = 5, w_{31} = 4, w_{32} = -4$ . What are the values of the internal nodes? Please simplify any fractions.

(i)  $h_1$

(ii)  $s_1$

(iii)  $y_2$

### b. Back Propagation

Compute the following gradients analytically. The answer should be an expression of any of the nodes in the network ( $x_1, x_2, h_1, h_2, h_3, r_1, r_2, r_3, s_1, s_2, y_1, y_2$ ) or weights  $w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}$ . In the case where the gradient depend on the value of nodes in the network, please list all possible analytical expressions, caused by active/inactive ReLU, separated by comma.

Hint 1: If  $z$  is a function of  $y$ , and  $y$  is a function of  $x$ , the chain rule of taking derivative is:  $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} * \frac{\partial y}{\partial x}$

Hint 2: Hint: Recall that for functions of the form  $g(x) = \frac{1}{1+\exp(a-x)}$ ,  $\frac{\partial g}{\partial x} = g(x)(1-g(x))$

(i)  $\frac{\partial h_2}{\partial x_1}$

(ii)  $\frac{\partial h_1}{\partial w_{21}}$

(iii)  $\frac{\partial r_3}{\partial w_{31}}$

(iv)  $\frac{\partial s_1}{\partial r_1}$

(v)  $\frac{\partial s_1}{\partial x_1}$

(vi)  $\frac{\partial y_2}{\partial s_2}$

(vii)  $\frac{\partial y_1}{\partial x_1}$

- c. In roughly 15 words, what role could the non-negative values in node  $r_2$  play according to its location in the network architecture?

a. (i) 5

$$(-10) * 3 + 7 * 5 = 5$$

(ii) 39

$$3 + \text{ReLU}((-10) * 3 + 7 * 5) + \text{ReLU}(2 * 3 + 5 * 5) = 3 + 5 + 31 = 39$$

(iii)  $\frac{1}{1+\exp(3)}$

$$s_2 = 5 + \text{ReLU}(2 * 3 + 5 * 5) = 5 + 31 = 36$$

$$y_2 = \frac{\exp(36)}{\exp(36)+\exp(39)} = \frac{1}{1+\exp(3)}$$

b. (i)  $w_{21}$

(ii) 0

$h_1$  is independent of weight  $w_{21}$

(iii) 0,  $x_1$  (two possibilities)

If the ReLU is active, the result is straightforward, and thus  $x_1$ . If the ReLU is inactive, the gradient is 0

(iv) 1

(v)  $1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

The ReLU with output  $r_1$  and the ReLU with output  $r_2$  can be active or inactive independently

(vi)  $y_1 y_2$ , or  $y_2(1 - y_2)$

$$y_2 = \frac{\exp(s_2)}{\exp(s_1)+\exp(s_2)} = \frac{1}{1+\exp(s_1-s_2)}$$

$$\frac{\partial y_2}{\partial s_2} = y_2(1 - y_2) = y_1 y_2 = \frac{\exp(s_1-s_2)}{(1+\exp(s_1-s_2))^2} = \frac{\exp(s_1+s_2)}{(\exp(s_1)+\exp(s_2))^2}$$

Please note that, due to symmetry between  $y_1$  and  $y_2$ ,  $\frac{\exp(s_2-s_1)}{(1+\exp(s_2-s_1))^2}$  is also equivalent to the correct answer. If you don't believe it, try simplifying it!

(vii)  $y_1 y_2(1 + w_{11} - w_{31}), y_1 y_2(1 + w_{11}), y_1 y_2(1 - w_{31}), y_1 y_2$  (four possibilities)

Explanation:

$$y_1 = \frac{\exp(s_1)}{\exp(s_1)+\exp(s_2)} = \frac{1}{1+\exp(s_2-s_1)}$$

When calculating  $\frac{\partial y_1}{\partial s_1}$ , this is the correct format for  $g(x) = \frac{1}{1+\exp(a-x)}$ , because there is a negative sign in front of  $s_1$ . We can apply the rule to simply get  $\frac{\partial y_1}{\partial s_1} = y_1(1 - y_1) = y_1 y_2$

But when calculating  $\frac{\partial y_1}{\partial s_2}$ , this is NOT in the correct format, because now there is no negative sign in front of  $s_2$ . One way to resolve this is to create a fake variable  $t_2 = -s_2$ , and  $\frac{\partial t_2}{\partial s_2} = -1$ .

Now  $y_1 = \frac{1}{1+\exp(s_2-s_1)} = \frac{1}{1+\exp(-s_1+s_2)} = \frac{1}{1+\exp(-s_1-t_2)}$ , which is the correct format again. At this point,  $\frac{\partial y_1}{\partial s_2} = \frac{\partial y_1}{\partial t_2} \frac{\partial t_2}{\partial s_2} = [y_1(1 - y_1)](-1) = -y_1 y_2$

As seen in v,  $\frac{\partial s_1}{\partial x_1} = 1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

Similarly,  $\frac{\partial s_2}{\partial x_1} = w_{21} + w_{31}, w_{21}, w_{31}$

In the end,

$$\begin{aligned} \frac{\partial y_1}{\partial x_1} &= \frac{\partial y_1}{\partial s_1} \frac{\partial s_1}{\partial x_1} + \frac{\partial y_1}{\partial s_2} \frac{\partial s_2}{\partial x_1} \\ &= y_1 y_2 * \{1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1\} \\ &\quad + (-y_1 y_2) * \{w_{21} + w_{31}, w_{21}, w_{31}\} \end{aligned}$$

Which can be expanded to this list (note that things can cancel out):  $y_1y_2(1 + w_{11} - w_{31})$ ,  $y_1y_2(1 + w_{11})$ ,  $y_1y_2(1 + w_{11} + w_{21} - w_{31})$ ,  $y_1y_2(1 - w_{31})$ ,  $y_1y_2$ ,  $y_1y_2(1 + w_{21} - w_{31})$ ,  $y_1y_2(1 + w_{11} - w_{21} - w_{31})$ ,  $y_1y_2(1 + w_{11} - w_{21})$ ,  $y_1y_2(1 + w_{11} - w_{31})$ ,  $y_1y_2(1 + -w_{21} - w_{31})$ ,  $y_1y_2(1 - w_{21})$ ,  $y_1y_2(1 - w_{31})$

But, note that the activity of  $r_2$  actually affect both  $\frac{\partial s_1}{\partial x_1}$  and  $\frac{\partial s_2}{\partial x_1}$ ! So we need to remove 6 contradicting terms, including  $y_1y_2(1 + w_{11} + w_{21} - w_{31})$ ,  $y_1y_2(1 + w_{21} - w_{31})$ ,  $y_1y_2(1 + w_{11} - w_{21} - w_{31})$ ,  $y_1y_2(1 + w_{11} - w_{21})$ ,  $y_1y_2(1 - w_{21} - w_{31})$ ,  $y_1y_2(1 - w_{21})$ .

The easier way to think about this is, that if  $r_2$  is activated, then the gradient will cancel out from the two sides. If  $r_2$  is deactivated, it won't appear at all.

We are left with 6 terms:  $y_1y_2(1 + w_{11} - w_{31})$ ,  $y_1y_2(1 + w_{11})$ ,  $y_1y_2(1 - w_{31})$ ,  $y_1y_2$ ,  $y_1y_2(1 + w_{11} - w_{31})$ ,  $y_1y_2(1 - w_{31})$

Removing the duplicates, we are left with the final 4 terms:  $y_1y_2(1 + w_{11} - w_{31})$ ,  $y_1y_2(1 + w_{11})$ ,  $y_1y_2(1 - w_{31})$ ,  $y_1y_2$

- c. Smoothing the inputs into the soft-max functions. (anything reasonable is acceptable)

### Exercise 21.2.#188-MTXB

Origin:

`fa17_final_MLQuestionEasier`

In this question we will perform the backward pass algorithm on the formula

$$f = \frac{1}{2} \|\mathbf{Ax}\|^2$$

Here,  $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ ,  $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ,  $\mathbf{b} = \mathbf{Ax} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ , and  $f = \frac{1}{2} \|\mathbf{b}\|^2 = \frac{1}{2} (b_1^2 + b_2^2)$  is a scalar.

a. Calculate the following partial derivatives of  $f$ .

(i) Find  $\frac{\partial f}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \end{bmatrix}$ .

b. Calculate the following partial derivatives of  $b_1$ .

(i)  $\left( \frac{\partial b_1}{\partial A_{11}}, \frac{\partial b_1}{\partial A_{12}} \right)$

(ii)  $\left( \frac{\partial b_1}{\partial A_{21}}, \frac{\partial b_1}{\partial A_{22}} \right)$

(iii)  $\left( \frac{\partial b_1}{\partial x_1}, \frac{\partial b_1}{\partial x_2} \right)$

c. Calculate the following partial derivatives of  $f$ .

(i)  $\left( \frac{\partial f}{\partial A_{11}}, \frac{\partial f}{\partial A_{12}} \right)$

(ii)  $\left( \frac{\partial f}{\partial A_{21}}, \frac{\partial f}{\partial A_{22}} \right)$

(iii)  $\left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$

d. Now we consider the general case where  $\mathbf{A}$  is an  $n \times d$  matrix, and  $\mathbf{x}$  is a  $d \times 1$  vector.

As before,  $f = \frac{1}{2} \|\mathbf{Ax}\|^2$ .

(i) Find  $\frac{\partial f}{\partial \mathbf{A}}$  in terms of  $\mathbf{A}$  and  $\mathbf{x}$  only.

(ii) Find  $\frac{\partial f}{\partial \mathbf{x}}$  in terms of  $\mathbf{A}$  and  $\mathbf{x}$  only.

a. (i)  $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

b. (i)  $(x_1, x_2)$

(ii)  $(0, 0)$

(iii)  $(A_{11}, A_{12})$

c. Calculate the following partial derivatives of  $f$ .

(i)  $(x_1 b_1, x_2 b_1)$

(ii)  $(x_1 b_2, x_2 b_2)$

(iii)  $(A_{11} b_1 + A_{21} b_2, A_{12} b_1 + A_{22} b_2)$

d. (i)  $\mathbf{Axx}^\top$

(ii)  $\mathbf{A}^\top \mathbf{Ax}$

## 21.3 Convolutional Networks

[[need exercises]]

## 21.4 Learning Algorithms

### Exercise 21.4.#TSOH

Suppose that a training set contains only a single example, repeated 100 times. In 80 of the 100 cases, the single output value is 1; in the other 20, it is 0. What will a back-propagation network predict for this example, assuming that it has been trained and reaches a global optimum? (*Hint:* to find the global optimum, differentiate the error function and set it to zero.)

This question is especially important for students who are not expected to implement or use a neural network system. Together with 20.15 and 20.17, it gives the student a concrete (if slender) grasp of what the network actually does. Many other similar questions can be devised.

Intuitively, the data suggest that a probabilistic prediction  $P(\text{Output} = 1) = 0.8$  is appropriate. The network will adjust its weights to minimize the error function. The error is

$$E = \frac{1}{2} \sum_i (y_i - a_i)^2 = \frac{1}{2} [80(1 - a_1)^2 + 20(0 - a_1)^2] = 50O_1^2 - 80O_1 + 50$$

The derivative of the error with respect to the single output  $a_1$  is

$$\frac{\partial E}{\partial a_1} = 100a_1 - 80$$

Setting the derivative to zero, we find that indeed  $a_1 = 0.8$ . The student should spot the connection to Ex. 18.8.

### Exercise 21.4.#NNRE

The neural network whose learning performance is measured in Figure ?? has four hidden nodes. This number was chosen somewhat arbitrarily. Use a cross-validation method to find the best number of hidden nodes.

This is just a simple example of the general cross-validation model-selection method described in the chapter. For each possible size of hidden layer up to some reasonable bound, the  $k$ -fold cross-validation score is obtained given the existing training data and the best hidden layer size is chosen. This can be done using the AIMA code or with any of several public-domain machine learning toolboxes such as WEKA.

### Exercise 21.4.#BPRE

Section 21.4.1 gives a generic formula for propagating gradients in computation graphs, embodied in Equations (21.11) and (21.12). Apply this process to the simple network in Figure 21.3 in order to rederive the gradient expressions in Equations (21.4) and (21.5).

[[need exercises]]

## 21.5 Generalization

### Exercise 21.5.#188-GENA

Origin:

sp13\_final\_ml-generalization

We consider the following different classifiers for classification of samples in a 2-dimensional feature space.

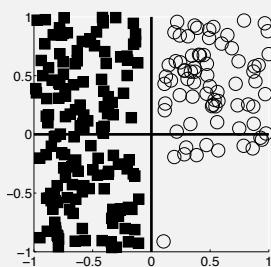
|         |                                                                                    |         |                                                                                                                                                      |
|---------|------------------------------------------------------------------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| PNoBias | Linear perceptron <i>without</i> a bias term<br>(features $[x_1 \ x_2]^\top$ )     | PCutoff | Kernel perceptron with the kernel function $K(x, z) = \max\{0, 0.01 - \ x - z\ _2\}$ ( $\ a - b\ _2$ is the Euclidean distance between $a$ and $b$ ) |
| PBias   | Linear perceptron with a bias term (features $[1 \ x_1 \ x_2]^\top$ )              |         |                                                                                                                                                      |
| PQuad   | Kernel perceptron with the quadratic kernel function $K(x, z) = (1 + x \cdot z)^2$ | 1NN     | 1-nearest neighbor classifier                                                                                                                        |

3NN 3-nearest neighbor classifier

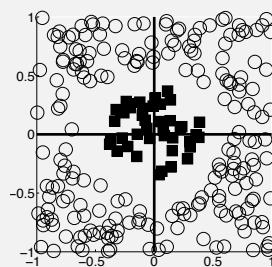
In each of the plots below you are given points from two classes, shown as filled rectangles and open circles. For each plot, list all each classifier that will be able to perfectly classify all of the training data (or, if none, mark “None of these will classify the data perfectly”).

Note that when computing the nearest neighbors for a training data point, the training data point will be its own nearest neighbor.

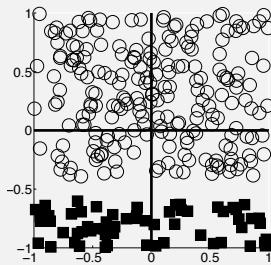
a.



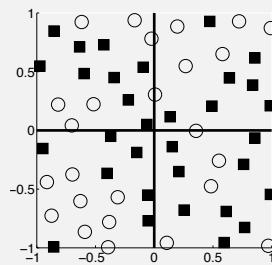
c.



b.



d.



- a. PNoBias, PBias, PQuad, PCutoff, 1NN  
3NN misclassifies the bottom-most circle.

- b. PBias,PQuad,PCutoff,1NN,3NN  
PNoBias is restricted to separators through the origin.
- c. PQuad,PCutoff,1NN,3NN  
The data are not linearly separable.
- d. PCutoff,1NN,3NN  
The decision boundary is complicated and in particular neither linear, nor quadratic.  
1NN and PCutoff classify locally.

**Exercise 21.5.#188-OFUF**

- a. Suppose you train a classifier and test it on a held-out validation set. It gets 80% classification accuracy on the training set and 20% classification accuracy on the validation set.
- (i) From what problem is your model most likely suffering? Underfitting or Overfitting?
  - (ii) To improve your classifier's performance on the validation set, should you add extra feature or remove some features? Please justify.
  - (iii) To improve your classifier's performance on the validation set, should you collect more training data or throw out some training data? Please justify.
- b. Suppose you train a classifier and test it on a held-out validation set. It gets 30% classification accuracy on the training set and 30% classification accuracy on the validation set.
- (i) From what problem is your model most likely suffering? Underfitting or Overfitting?
  - (ii) To improve your classifier's performance on the validation set, should you add extra feature or remove some features? Please justify.
  - (iii) To improve your classifier's performance on the validation set, should you collect more training data or throw out some training data? Please justify.
- c. Your boss provides you with an image dataset in which some of the images contain your company's logo, and others contain competitors' logos. You are tasked to code up a classifier to distinguish your company's logos from competitors' logos. You complete the assignment quickly and even send your boss your code for training the classifier, but your boss is furious. Your boss says that when running your code with images and a random label for each of the images as input, the classifier achieved perfect accuracy on the training set. And this happens for all of the many random labelings that were generated.

Do you agree that this is a problem? Justify your answer.

a. (i) Overfitting

(ii) Either answer was accepted with justification.

Add extra features – adding some really good features could better capture the structure in the data. Remove some features – the model may be using the noise in the abundant feature set to overfit to the training data rather than learning any meaningful underlying structure.

(iii) Collect more data.

More data should yield a more representative sample of the true distribution of the data. Less data is more susceptible to overfitting.

b. (i) Underfitting

(ii) Add extra features.

Under the current feature representation, we are unable to accurately model the training data for the purpose of the classification task we're interested in. The classifier may be able to deduce more information about the connections between data points and their classes from additional features, allowing it to better model the data for the classification task. For example, a linear perceptron could not accurately model two classes separated by a circle in a 2-dimensional feature space, but by using quadratic features in a kernel perceptron, we can find a perfect separating hyperplane.

(iii) Collect more training data or neither.

More training data can only be a good thing. Neither was accepted, too, as given that train and hold-out validation already achieve the same performance, likely the underlying problem is not a lack of training data.

c. Yes, this is a problem.

The classifier is overfitting the training set. The fact that it had perfect accuracy with random labels suggests that it does not learn any real underlying structure in the data; it most likely essentially memorized each of the training cases.

## 21.6 Recurrent Neural Networks

---

[[need exercises]]

## 21.7 Unsupervised Learning and Transfer Learning

---

[[need exercises]]

## 21.8 Applications

---

[[need exercises]]

# EXERCISES 22

## REINFORCEMENT LEARNING

### 22.1 Learning from Rewards

#### **Exercise 22.1.#RLAN**

Investigate the application of reinforcement learning ideas to the modeling of human and animal behavior.

Reinforcement learning as a general “setting” can be applied to almost any agent in any environment. The only requirement is that there be a distinguished reward signal. Thus, given the signals of pain, pleasure, hunger, and so on, we can map human learning directly into reinforcement learning—although this says nothing about how the “program” is implemented. This view misses out several important issues. First, we do not know what the human “reward function” is and how it relates to perceptual experience. Second, there are other forms of learning that occur in humans. These include “speedup learning” (Chapter 21); supervised learning from other humans, where the teacher’s feedback is taken as a distinguished input; and the process of learning the world model, which is “supervised” by the environment.

#### **Exercise 22.1.#RLEV**

Is reinforcement learning an appropriate abstract model for evolution? What connection exists, if any, between hardwired reward signals and evolutionary fitness?

To map evolutionary processes onto the formal model of reinforcement learning, one must find evolutionary analogs for the reward signal, learning process, and the learned policy. Let us start with a simple animal that does not learn during its own lifetime. This animal’s genotype, to the extent that it determines animal’s behavior over its lifetime, can be thought of as the parameters  $\theta$  of a policy  $p_{i\theta}$ . Mutations, crossover, and related processes are the part of the learning algorithm—like an empirical gradient neighborhood generator in policy search—that creates new values of  $\theta$ . One can also imagine a reinforcement learning process that works on many different copies of  $\pi$  simultaneously, as evolution does; evolution adds the complication that each copy of  $\pi$  modifies the environment for other copies of  $\pi$ , whereas in RL the environment dynamics are assumed fixed, independent of the policy chosen by the agent. The most difficult issue, as the question indicates, is the reward function and the underlying objective function of the learning process. In RL, the objective function is to find policies that maximize the expected sum of rewards over time. Biologists usually talk about evolution as maximizing “reproductive fitness,” i.e., the ability of individuals of

a given genotype to reproduce and thereby propagate the genotype to the next generation. In this simple view, evolution’s “objective function” is to find the  $\pi$  that generates the most copies of itself over infinite time. Thus, the “reward signal” is positive for creation of new individuals; death, *per se*, seems to be irrelevant.

Of course, the real story is much more complex. Natural selection operates not just at the genotype level but also at the level of individual genes and groups of genes; the environment is certainly multiagent rather than single-agent; and, as noted in the case of Baldwinian evolution in Chapter 4, evolution may result in organisms that have hardwired reward signals that are related to the fitness reward and may use those signals to learn during their lifetimes.

As far as we know there has been no careful and philosophically valid attempt to map evolution onto the formal model of reinforcement learning; any such attempt must be careful not to *assume* that such a mapping is possible or to *ascrIBE* a goal to evolution; at best, one may be able to interpret what evolution tends to do *as if* it were the result of some maximizing process, and ask what it is that is being maximized.

## 22.2 Passive Reinforcement Learning

### Exercise 22.2.#PASL

Implement a passive learning agent in a simple environment, such as the  $4 \times 3$  world. For the case of an initially unknown environment model, compare the learning performance of the direct utility estimation, TD, and ADP algorithms. Do the comparison for the optimal policy and for several random policies. For which do the utility estimates converge faster? What happens when the size of the environment is increased? (Try environments with and without obstacles.)

The code repository shows an example of this, implemented in the passive  $4 \times 3$  environment. The agents are found under `lisp/learning/agents/passive*.lisp` and the environment is in `lisp/learning/domains/4x3-passive-mdp.lisp`. (The MDP is converted to a full-blown environment using the function `mdp->environment` which can be found in `lisp/uncertainty/environments/mdp.lisp`.)

### Exercise 22.2.#PRPO

Chapter 17 defined a **proper policy** for an MDP as one that is guaranteed to reach a terminal state. Show that it is possible for a passive ADP agent to learn a transition model for which its policy  $\pi$  is improper even if  $\pi$  is proper for the true MDP; with such models, the POLICY-EVALUATION step may fail if  $\gamma = 1$ . Show that this problem cannot arise if POLICY-EVALUATION is applied to the learned model only at the end of a trial.

Consider a world with two states,  $S_0$  and  $S_1$ , with two actions in each state: stay still or move to the other state. Assume the move action is non-deterministic—it sometimes fails, leaving the agent in the same state. Furthermore, assume the agent starts in  $S_0$  and that  $S_1$  is a terminal state. If the agent tries several move actions and they all fail, the agent may conclude

that  $T(S_0, Move, S_1)$  is 0, and thus may choose a policy with  $\pi(S_0) = Stay$ , which is an improper policy. If we wait until the agent reaches  $S_1$  before updating, we won't fall victim to this problem.

### Exercise 22.2.#PRSW

Starting with the passive ADP agent, modify it to use an approximate ADP algorithm as discussed in the text. Do this in two steps:

- a. Implement a priority queue for adjustments to the utility estimates. Whenever a state is adjusted, all of its predecessors also become candidates for adjustment and should be added to the queue. The queue is initialized with the state from which the most recent transition took place. Allow only a fixed number of adjustments.
- b. Experiment with various heuristics for ordering the priority queue, examining their effect on learning rates and computation time.

This question essentially asks for a reimplementation of a general scheme for asynchronous dynamic programming of which the prioritized sweeping algorithm is an example (Moore and Atkeson, 1993). For **a.**, there is code for a priority queue in both the Lisp and Python code repositories. So most of the work is the experimentation called for in **b.**

### Exercise 22.2.#DIRU

The direct utility estimation method in Section 22.2 uses distinguished terminal states to indicate the end of a trial. How could it be modified for environments with discounted rewards and no terminal states?

When there are no terminal states there are no sequences, so we need to define sequences. We can do that in several ways. First, if rewards are sparse, we can treat any state with a reward as the end of a sequence. We can use equation (21.2); the only problem is that we don't know the exact total reward of the state at the end of the sequence, because it is not a terminal state. We can estimate it using the current  $U(s)$  estimate. Another option is to arbitrarily limit sequences to  $n$  states, and then consider the next  $n$  states, etc. A variation on this is to use a sliding window of states, so that the first sequence is states  $1 \dots n$ , the second sequence is  $2 \dots n + 1$ , etc.

## 22.3 Active Reinforcement Learning

[[need exercises]]

## 22.4 Generalization in Reinforcement Learning

### Exercise 22.4.#TDLQ

Write out the parameter update equations for TD learning with

$$\hat{U}(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g)^2 + (y - y_g)^2}.$$

This utility estimation function is similar to equation (21.9), but adds a term to represent Euclidean distance on a grid. Using equation (21.10), the update equations are the same for  $\theta_0$  through  $\theta_2$ , and the new parameter  $\theta_3$  can be calculated by taking the derivative with respect to  $\theta_3$ :

$$\begin{aligned}\theta_0 &\leftarrow \theta_0 + \alpha (u_j(s) - \hat{U}_\theta(s)) , \\ \theta_1 &\leftarrow \theta_1 + \alpha (u_j(s) - \hat{U}_\theta(s))x , \\ \theta_2 &\leftarrow \theta_2 + \alpha (u_j(s) - \hat{U}_\theta(s))y , \\ \theta_3 &\leftarrow \theta_3 + \alpha (u_j(s) - \hat{U}_\theta(s))\sqrt{(x - x_g)^2 + (y - y_g)^2} .\end{aligned}$$

### Exercise 22.4.#VCRL

Adapt the vacuum world (Chapter 2) for reinforcement learning by including rewards for squares being clean. Make the world observable by providing suitable percepts. Now experiment with different reinforcement learning agents. Is function approximation necessary for success? What sort of approximator works for this application?

The conversion of the vacuum world problem specification into an environment suitable for reinforcement learning can be done by merging elements of `mdp->environment` from `lisp/uncertainty/environments/mdp.lisp` with elements of the corresponding function `problem->environment` from `lisp/search/agents.lisp`. The point here is twofold: first, that the reinforcement learning algorithms in Chapter 20 are limited to accessible environments; second, that the dirt makes a huge difference to the size of the state space. Without dirt, the state space is  $O(n)$  with  $n$  locations. With dirt, the state space is  $O(2^n)$  because each location can be clean or dirty. For this reason, input generalization is clearly necessary for  $n$  above about 10. This illustrates the misleading nature of “navigation” domains in which the state space is proportional to the “physical size” of the environment. “Real-world” environments typically have some combinatorial structure that results in exponential growth.

### Exercise 22.4.#APLM

Implement an exploring reinforcement learning agent that uses direct utility estimation. Make two versions—one with a tabular representation and one using the function approxi-

mator in Equation (22.9). Compare their performance in three environments:

- a. The  $4 \times 3$  world described in the chapter.
- b. A  $10 \times 10$  world with no obstacles and a +1 reward at (10,10).
- c. A  $10 \times 10$  world with no obstacles and a +1 reward at (5,5).

Code not shown. Several reinforcement learning agents are given in the directory `lisp/learning/`

#### Exercise 22.4.#RLGR

Devise suitable features for reinforcement learning in stochastic grid worlds (generalizations of the  $4 \times 3$  world) that contain multiple obstacles and multiple terminal states with rewards of +1 or -1.

Possible features include:

- Distance to the nearest +1 terminal state.
- Distance to the nearest -1 terminal state.
- Number of adjacent +1 terminal states.
- Number of adjacent -1 terminal states.
- Number of adjacent obstacles.
- Number of obstacles that intersect with a path to the nearest +1 terminal state.

#### Exercise 22.4.#RLGM

Extend the standard game-playing environment (Chapter 5) to incorporate a reward signal. Put two reinforcement learning agents into the environment (they may, of course, share the agent program) and have them play against each other. Apply the generalized TD update rule (Equation (22.11)) to update the evaluation function. You might wish to start with a simple linear weighted evaluation function and a simple game, such as tic-tac-toe.

The modification involves combining elements of the environment converter for games (`game->environment` in `lisp/search/games.lisp`) with elements of the function `mdp->environment`. The reward signal is just the utility of winning/drawing/losing and occurs only at the end of the game. The evaluation function used by each agent is the utility function it learns through the TD process. It is important to keep the TD learning process (which is entirely independent of the fact that a game is being played) distinct from the game-playing algorithm. Using the evaluation function with a deep search is probably better because it will help the agents to focus on relevant portions of the search space by improving the quality of play. There is, however, a tradeoff: the deeper the search, the more computer time is used in playing each training game.

**Exercise 22.4.#TENX**

Compute the true utility function and the best linear approximation in  $x$  and  $y$  (as in Equation (22.9)) for the following environments:

- A  $10 \times 10$  world with a single +1 terminal state at (10,10).
- As in (a), but add a -1 terminal state at (10,1).
- As in (b), but add obstacles in 10 randomly selected squares.
- As in (b), but place a wall stretching from (5,2) to (5,9).
- As in (a), but with the terminal state at (5,5).

The actions are deterministic moves in the four directions. In each case, compare the results using three-dimensional plots. For each environment, propose additional features (besides  $x$  and  $y$ ) that would improve the approximation and show the results.

This is a relatively time-consuming exercise. Code not shown to compute three-dimensional plots. The utility functions are:

- $U(x, y) = 1 - \gamma((10 - x) + (10 - y))$  is the true utility, and is linear.
- Same as in a, except that  $U(10, 1) = -1$ .
- The exact utility depends on the exact placement of the obstacles. The best approximation is the same as in a. The features in exercise 21.9 might improve the approximation.
- The optimal policy is to head straight for the goal from any point on the right side of the wall, and to head for (5, 10) first (and then for the goal) from any point on the left of the wall. Thus, the exact utility function is:

$$\begin{aligned} U(x, y) &= 1 - \gamma((10 - x) + (10 - y)) && (\text{if } x \geq 5) \\ &= 1 - \gamma((5 - x) + (10 - y)) - 5\gamma && (\text{if } x < 5) \end{aligned}$$

Unfortunately, this is not linear in  $x$  and  $y$ , as stated. Fortunately, we can restate the optimal policy as “head straight up to row 10 first, then head right until column 10.” This gives us the same exact utility as in a, and the same linear approximation.

- $U(x, y) = 1 - \gamma(|5 - x| + |5 - y|)$  is the true utility. This is also not linear in  $x$  and  $y$ , because of the absolute value signs. All can be fixed by introducing the features  $|5 - x|$  and  $|5 - y|$ .

## 22.5 Policy Search

---

[[need exercises]]

## 22.6 Apprenticeship and Inverse Reinforcement Learning

---

[[need exercises]]

## 22.7 Applications of Reinforcement Learning

---

[[need exercises]]

# EXERCISES 23

## NATURAL LANGUAGE PROCESSING

### 23.1 Language Models

#### **Exercise 23.1.#TXUN**

Read the following text once for understanding, and remember as much of it as you can. There will be a test later.

The procedure is actually quite simple. First you arrange things into different groups. Of course, one pile may be sufficient depending on how much there is to do. If you have to go somewhere else due to lack of facilities that is the next step, otherwise you are pretty well set. It is important not to overdo things. That is, it is better to do too few things at once than too many. In the short run this may not seem important but complications can easily arise. A mistake is expensive as well. At first the whole procedure will seem complicated. Soon, however, it will become just another facet of life. It is difficult to foresee any end to the necessity for this task in the immediate future, but then one can never tell. After the procedure is completed one arranges the material into different groups again. Then they can be put into their appropriate places. Eventually they will be used once more and the whole cycle will have to be repeated. However, this is part of life.

No answer required; just read the passage.

#### **Exercise 23.1.#NGRM**

This exercise explores the quality of the  $n$ -gram model of language. Find or create a monolingual corpus of 100,000 words or more. Segment it into words, and compute the frequency of each word. How many distinct words are there? Also count frequencies of bigrams (two consecutive words) and trigrams (three consecutive words). Now use those frequencies to generate language: from the unigram, bigram, and trigram models, in turn, generate a 100-word text by making random choices according to the frequency counts. Compare the three generated texts with actual language. Finally, calculate the perplexity of each model.

Code not shown. The distribution of words should fall along a Zipfian distribution: a straight line on a log-log scale. The generated language should be similar to the examples in the chapter.

### Exercise 23.1.#SEGM

Write a program to do **segmentation** of words without spaces. Given a string, such as the URL “[thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com](http://thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com),” return a list of component words: [“the,” “longest,” “list,” . . .]. This task is useful for parsing URLs, for spelling correction when words run together, and for languages such as Chinese that do not have spaces between words. It can be solved with a unigram or bigram word model and a dynamic programming algorithm similar to the Viterbi algorithm.

Using a unigram language model, the probability of a segmentation of a string  $s_{1:N}$  into  $k$  nonempty words  $s = w_1 \dots w_k$  is  $\prod_{i=1}^k P_{lm}(w_i)$  where  $P_{lm}$  is the unigram language model. This is not normalized without a distribution over the number of words  $k$ , but let’s ignore this for now.

To see that we can find the most probable segmentation of a string by dynamic programming, let  $p(i)$  be the maximum probability of any segmentation of  $s_{i:N}$  into words. Then  $p(N+1) = 1$  and

$$p(i) = \max_{j=i,\dots,N} P_{lm}(s_{i:j}) p(j+1)$$

because any segmentation of  $s_{i:N}$  starts with a single word spanning  $s_{i:j}$  and a segmentation of the rest of the string  $s_{j+1:N}$ . Because we are using a unigram model, the optimal segmentation of  $s_{j+1:N}$  does not depend on the earlier parts of the string.

Using the techniques of this chapter to form a unigram model accessed by the function `prob_word(word)`, the following Python code solves the above dynamic program to output an optimal segmentation:

```
def segment(text):
 length = len(text)
 max_prob = [0] * (length+1)
 max_prob[length] = 1
 split_idx = [-1] * (length+1)
 for start in range(length,-1,-1):
 for split in range(start+1,length+1):
 p = max_prob[split] * prob_word(text[start:split])
 if p > max_prob[start]:
 max_prob[start] = p
 split_idx[start] = split
 i = 0
 words = []
 while i < length:
 words.append(text[i:split_idx[i]])
 i = split_idx[i]
 if i == -1:
 return None # for text with zero probability
 return words
```

One difficulty is dealing with unknown words. Should a word have probability zero just because it was not seen in the training corpus? If not, what probability should it have? Simple Laplace smoothing would not work well for this application, because the word consisting of the entire input would get the same probability as all other unknown words. The language

model should assign probabilities to unknown words based on their length. One natural option is to fit an exponential distribution to the words lengths of a corpus. Alternatively, one could learn a distribution over the number of words in a string based on its length, add a  $P(k)$  term to the probability of a segmentation, and modify the dynamic program to handle this (i.e., to compute  $p(i, k)$  the maximum probability of segmenting  $s_{i:N}$  into  $k$  words).

Another implementation is discussed at [norvig.com/ngrams/ch14.pdf](http://norvig.com/ngrams/ch14.pdf).

### Exercise 23.1.#LPCS

Consider a text corpus consisting of  $N$  tokens of  $d$  distinct words and the number of times each distinct word  $w$  appears is given by  $x_w$ . We want to apply a version of Laplace smoothing that estimates a word's probability as:

$$\frac{x_w + \alpha}{N + \alpha d}$$

for some constant  $\alpha$  (Laplace recommended  $\alpha = 1$ , but other values are possible.) In the following problems, assume  $N$  is 100,000,  $d$  is 10,000 and  $\alpha$  is 2.

a. Give both the unsmoothed maximum likelihood probability estimate and the Laplace smoothed estimate of a word that appears 1,000 times in the corpus.

b. Do the same for a word that does not appear at all.

c. You are running a Naive Bayes text classifier with Laplace Smoothing, and you suspect that you are overfitting the data. How would you increase or decrease the parameter  $\alpha$ ?

d. Could increasing  $\alpha$  increase or decrease training set error? Increase or decrease validation set error?

a. Unsmoothed:  $1,000/100,000 = 0.01$ ; Smoothed:

b. Unsmoothed: 0; Smoothed:  $2/(100,000 + 20,000)$

c. Increase  $\alpha$ , so that the presence of a word that appears very few times in the corpus has less effect.

d. Increasing  $\alpha$  tends to increase training error, because it trusts the training data less. It may or may not increase validation set error; you need to find the value of  $\alpha$  that balances overfitting and underfitting, and the given information isn't enough to tell.

### Exercise 23.1.#ZIPF

*Zipf's law* of word distribution states the following: Take a large corpus of text, count the frequency of every word in the corpus, and then rank these frequencies in decreasing order. Let  $f_I$  be the  $I$ th largest frequency in this list; that is,  $f_1$  is the frequency of the most common word (usually “the”),  $f_2$  is the frequency of the second most common word, and so on. Zipf's law states that  $f_I$  is approximately equal to  $\alpha/I$  for some constant  $\alpha$ . The law tends to be highly accurate except for very small and very large values of  $I$ .

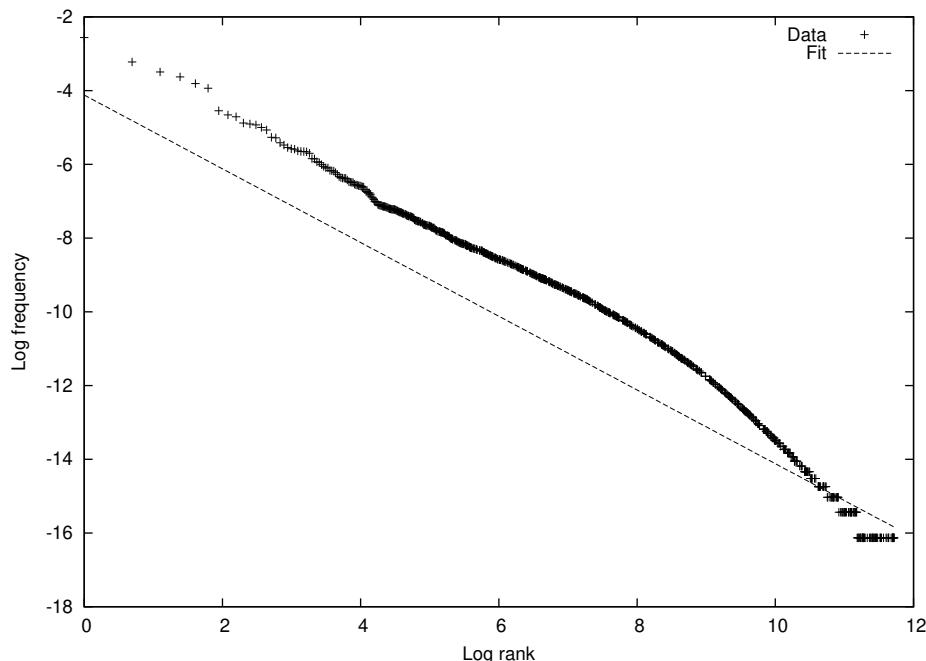
Choose a corpus of at least 20,000 words of online text, and verify Zipf's law experimentally. Define an error measure and find the value of  $\alpha$  where Zipf's law best matches your

experimental data. Create a log–log graph plotting  $f_I$  vs.  $I$  and  $\alpha/I$  vs.  $I$ . (On a log–log graph, the function  $\alpha/I$  is a straight line.) In carrying out the experiment, be sure to eliminate any formatting tokens (e.g., HTML tags) and normalize upper and lower case.

Using a corpus of 123,612 words extracted from wikipedia with unique words removed, we selected  $\alpha$  to optimize two different measures:

- a. Least-squares:  $\sum_{I=1}^N (\log(\alpha/I) - \log f_I)^2$
- b. Weighted least-squares:  $\sum_{I=1}^N (1/I)(\log(\alpha/I) - \log f_I)^2$

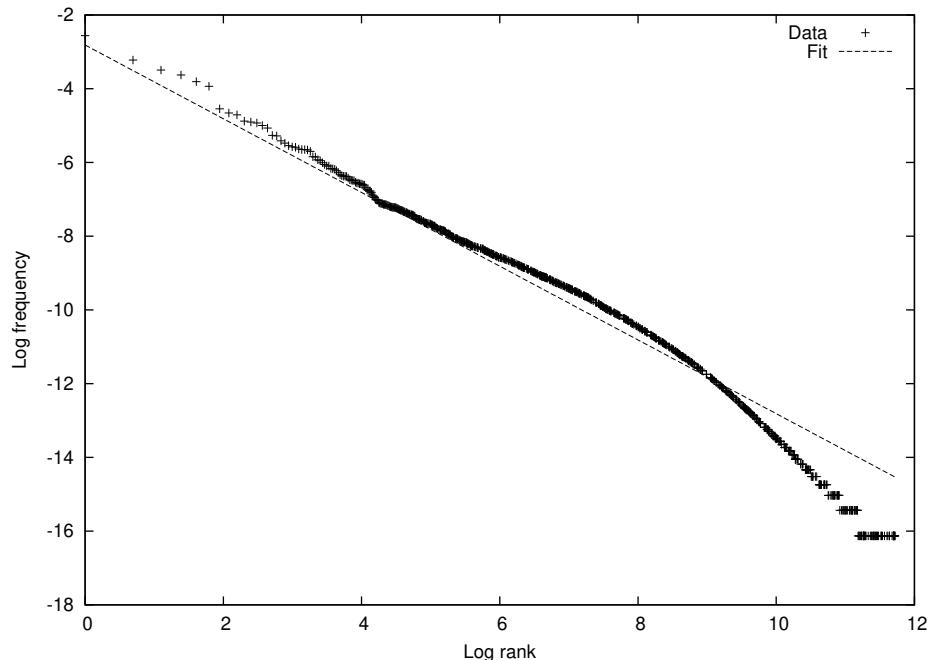
The former measure is the obvious fit of a line to the curve in the log–log domain, resulting in  $\alpha = 0.0162$  illustrated in Figure S23.1. This line is distorted to handle the numerous large values of  $I$  which don't fit the law. The latter measure assigns lower weight to large values of  $I$  to correct for this, resulting in  $\alpha = 0.0600$  illustrated in Figure S23.2



**Figure S23.1** Data and least squares fit of  $\alpha$ .

### Exercise 23.1.#AUTH

In this exercise you will develop a classifier for authorship: given a text, the classifier predicts which of two candidate authors wrote the text. Obtain samples of text from two different authors. Separate them into training and test sets. Now train a language model



**Figure S23.2** Data and weighted least squares fit of  $\alpha$ .

on the training set. You can choose what features to use;  $n$ -grams of words or letters are the easiest, but you can add additional features that you think may help. Then compute the probability of the text under each language model and chose the most probable model. Assess the accuracy of this technique. How does accuracy change as you alter the set of features? This subfield of linguistics is called **stylometry**; its successes include the identification of the author of the disputed *Federalist Papers* and some disputed works of Shakespeare. (Adapted from Jurafsky and Martin (2000).)

Code not shown. The approach suggested here will work in some cases, for authors with distinct vocabularies. For more similar authors, other features such as bigrams, average word and sentence length, parts of speech, and punctuation might help. Accuracy will also depend on how many authors are being distinguished. One interesting way to make the task easier is to group authors into male and female, and try to distinguish the sex of an author not previously seen. This was suggested by the work of Shlomo Argamon.

### Exercise 23.1.#SPAM

This exercise concerns the classification of spam email. Create a corpus of spam email and one of non-spam mail. Examine each corpus and decide what features appear to be useful

for classification: unigram words? bigrams? message length, sender, time of arrival? Then train a classification algorithm (decision tree, naive Bayes, SVM, logistic regression, or some other algorithm of your choosing) on a training set and report its accuracy on a test set.

Code not shown. There are now several open-source projects to do Bayesian spam filtering, so beware if you assign this exercise.

### Exercise 23.1.#POSX

Some linguists have argued as follows:

Children learning a language hear only *positive examples* of the language and no *negative examples*. Therefore, the hypothesis that “every possible sentence is in the language” is consistent with all the observed examples. Moreover, this is the simplest consistent hypothesis. Furthermore, all grammars for languages that are supersets of the true language are also consistent with the observed data. Yet children do induce (more or less) the right grammar. It follows that they begin with very strong innate grammatical constraints that rule out all of these more general hypotheses *a priori*.

Comment briefly on the weak point(s) in this argument, given what you know about statistical learning.

If we consider the simplest kind of maximum likelihood learning, for grammars viewed as generative models, the argument fails. The grammar that accepts every sentence must put a very small probability on each, so the likelihood for such a grammar will be much lower than for the correct grammar. Another way to say this is that if the grammar that accepts every sentence were the true grammar, we would see many more sentences than we actually do!

## 23.2 Grammar

### Exercise 23.2.#GMSL

This exercise concerns grammars for very simple languages.

- a. Write a context-free grammar for the language  $a^n b^n$ .
- b. Write a context-free grammar for the palindrome language: the set of all strings whose second half is the reverse of the first half.
- c. Write a context-sensitive grammar for the duplicate language: the set of all strings whose second half is the same as the first half.

The purpose of this exercise is to get some experience with simple grammars, and to see how context-sensitive grammars are more complicated than context-free. One approach to writing grammars is to write down the strings of the language in an orderly fashion, and then see how

a progression from one string to the next could be created by recursive application of rules. For example:

- a.** The language  $a^n b^n$ : The strings are  $\epsilon, ab, aabb, \dots$  (where  $\epsilon$  indicates the null string). Each member of this sequence can be derived from the previous by wrapping an  $a$  at the start and a  $b$  at the end. Therefore a grammar is:

$$\begin{aligned} S &\rightarrow \epsilon \\ S &\rightarrow a S b \end{aligned}$$

- b.** The palindrome language: Let's assume the alphabet is just  $a, b$  and  $c$ . (In general, the size of the grammar will be proportional to the size of the alphabet. There is no way to write a context-free grammar without specifying the alphabet/lexicon.) The strings of the language include  $\epsilon, a, b, c, aa, bb, cc, aaa, aba, aca, bab, bbb, bcb, \dots$ . In general, a string can be formed by bracketing any previous string with two copies of any member of the alphabet. So a grammar is:

$$S \rightarrow \epsilon \mid a \mid b \mid c \mid a S a \mid b S b \mid c S c$$

- c.** The duplicate language: For the moment, assume that the alphabet is just  $ab$ . (It is straightforward to extend to a larger alphabet.) The duplicate language consists of the strings:  $\epsilon, aa, bb, aaaa, abab, bbbb, baba, \dots$ . Note that all strings are of even length.

One strategy for creating strings in this language is this:

- Start with markers for the front and middle of the string: we can use the non-terminal  $F$  for the front and  $M$  for the middle. So at this point we have the string  $FM$ .
- Generate items at the front of the string: generate an  $a$  followed by an  $A$ , or a  $b$  followed by a  $B$ . Eventually we get, say,  $FaAaAbBM$ . Then we no longer need the  $F$  marker and can delete it, leaving  $aAaAbBM$ .
- Move the non-terminals  $A$  and  $B$  down the line until just before the  $M$ . We end up with  $aabAABM$ .
- Hop the  $A$ s and  $B$ s over the  $M$ , converting each to a terminal ( $a$  or  $b$ ) as we go. Then we delete the  $M$ , and are left with the end result:  $aabaab$ .

Here is a grammar to implement this strategy:

$$\begin{aligned} S &\rightarrow F M && \text{(starting markers)} \\ F &\rightarrow F a A && \text{(introduce symbols)} \\ F &\rightarrow F b B \\ F &\rightarrow \epsilon && \text{(delete the } F \text{ marker)} \\ A a &\rightarrow a A && \text{(move non-terminals down to the } M\text{)} \\ A b &\rightarrow b A \\ B a &\rightarrow a B \\ B b &\rightarrow b B \\ A M &\rightarrow M a && \text{(hop over } M \text{ and convert to terminal)} \\ B M &\rightarrow M b \\ M &\rightarrow \epsilon && \text{(delete the } M \text{ marker)} \end{aligned}$$

Here is a trace of the grammar deriving *aabaab*:

*S*  
*FM*  
*FbBM*  
*FaAbBM*  
*FaAaAbBM*  
*aAaAbBM*  
*aaAAbBM*  
*aaAbABM*  
*aabAABM*  
*aabAAMb*  
*aabAMab*  
*aabMaab*  
*aabaab*

### Exercise 23.2.#CNFX

In this exercise you will transform  $\mathcal{E}_0$  into Chomsky Normal Form (CNF). There are five steps: (a) Add a new start symbol, (b) Eliminate  $\epsilon$  rules, (c) Eliminate multiple words on right-hand sides, (d) Eliminate rules of the form  $(X \rightarrow Y)$ , (e) Convert long right-hand sides into binary rules.

- The start symbol,  $S$ , can occur only on the left-hand side in CNF. Add a new rule of the form  $S' \rightarrow S$ , using a new symbol  $S'$ .
- The empty string,  $\epsilon$  cannot appear on the right-hand side in CNF.  $\mathcal{E}_0$  does not have any rules with  $\epsilon$ , so this is not an issue.
- A word can appear on the right-hand side in a rule only of the form  $(X \rightarrow \text{word})$ . Replace each rule of the form  $(X \rightarrow \dots \text{word} \dots)$  with  $(X \rightarrow \dots W' \dots)$  and  $(W' \rightarrow \text{word})$ , using a new symbol  $W'$ .
- A rule  $(X \rightarrow Y)$  is not allowed in CNF; it must be  $(X \rightarrow Y Z)$  or  $(X \rightarrow \text{word})$ . Replace each rule of the form  $(X \rightarrow Y)$  with a set of rules of the form  $(X \rightarrow \dots)$ , one for each rule  $(Y \rightarrow \dots)$ , where  $(\dots)$  indicates one or more symbols.
- Replace each rule of the form  $(X \rightarrow Y Z \dots)$  with two rules,  $(X \rightarrow Y Z')$  and  $(Z' \rightarrow Z \dots)$ , where  $Z'$  is a new symbol.

Show each step of the process and the final set of rules.

The final grammar is shown below. In step **d**, students may be tempted to drop the rules  $(Y \rightarrow \dots)$ , which fails immediately.

$$\begin{aligned}
 S &\rightarrow NP\ VP \\
 &\quad | \quad S' \text{ } Conj \text{ } S' \\
 S' &\rightarrow NP\ VP \\
 &\quad | \quad SConj \text{ } S' \\
 SConj &\rightarrow S' \text{ } Conj \\
 \\ 
 NP &\rightarrow \mathbf{me} \mid \mathbf{you} \mid \mathbf{I} \mid \mathbf{it} \mid \dots \\
 &\quad | \quad \mathbf{John} \mid \mathbf{Mary} \mid \mathbf{Boston} \mid \dots \\
 &\quad | \quad \mathbf{stench} \mid \mathbf{breeze} \mid \mathbf{wumpus} \mid \mathbf{pits} \mid \dots \\
 &\quad | \quad \text{Article Noun} \\
 &\quad | \quad \text{ArticleAdjs Noun} \\
 &\quad | \quad \text{Digit Digit} \\
 &\quad | \quad NP\ PP \\
 &\quad | \quad NP\ RelClause \\
 ArticleAdjs &\rightarrow \text{Article Adjs} \\
 \\ 
 VP &\rightarrow \mathbf{is} \mid \mathbf{feel} \mid \mathbf{smells} \mid \mathbf{stinks} \mid \dots \\
 &\quad | \quad VP\ NP \\
 &\quad | \quad VP\ Adjective \\
 &\quad | \quad VP\ PP \\
 &\quad | \quad VP\ Adverb \\
 \\ 
 Adjs &\rightarrow \mathbf{right} \mid \mathbf{dead} \mid \mathbf{smelly} \mid \mathbf{breezy} \dots \\
 &\quad | \quad \text{Adjective Adjs} \\
 PP &\rightarrow \text{Prep } NP \\
 RelClause &\rightarrow \text{RelPro } VP
 \end{aligned}$$

### Exercise 23.2.#HMMG

An *HMM grammar* is essentially a standard HMM whose state variable is  $N$  (nonterminal, with values such as *Det*, *Adjective*, *Noun* and so on) and whose evidence variable is  $W$  (word, with values such as *is*, *duck*, and so on). The HMM model includes a prior  $\mathbf{P}(N_0)$ , a transition model  $\mathbf{P}(N_{t+1}|N_t)$ , and a sensor model  $\mathbf{P}(W_t|N_t)$ . Show that every HMM grammar can be written as a PCFG. [Hint: start by thinking about how the HMM prior can be represented by PCFG rules for the sentence symbol. You may find it helpful to illustrate for the particular HMM with values  $A, B$  for  $N$  and values  $x, y$  for  $W$ .]

The prior is represented by rules such as

$$P(N_0 = A) : \quad S \rightarrow A \text{ } S_A$$

where  $S_A$  means “rest of sentence after an  $A$ .” Transitions are represented as, for example,

$$P(N_{t+1} = B \mid N_t = A) : \quad S_A \rightarrow B \ S_B$$

and the sensor model is just the lexical rules such as

$$P(W_t = \text{is} \mid N_t = A) : \quad A \rightarrow \text{is} .$$

### Exercise 23.2.#VPGR

Consider the following PCFG for simple verb phrases:

$$\begin{aligned} 0.1 &: VP \rightarrow Verb \\ 0.2 &: VP \rightarrow Copula \ Adjective \\ 0.5 &: VP \rightarrow Verb \ the \ Noun \\ 0.2 &: VP \rightarrow VP \ Adverb \\ 0.5 &: Verb \rightarrow is \\ 0.5 &: Verb \rightarrow shoots \\ 0.8 &: Copula \rightarrow is \\ 0.2 &: Copula \rightarrow seems \\ 0.5 &: Adjective \rightarrow unwell \\ 0.5 &: Adjective \rightarrow well \\ 0.5 &: Adverb \rightarrow well \\ 0.5 &: Adverb \rightarrow badly \\ 0.6 &: Noun \rightarrow duck \\ 0.4 &: Noun \rightarrow well \end{aligned}$$

- a. Which of the following have a nonzero probability as a VP? (i) shoots the duck well well well (ii) seems the well well (iii) shoots the unwell well badly
- b. What is the probability of generating “is well well”?
- c. What types of ambiguity are exhibited by the phrase in (b)?
- d. Given any PCFG, is it possible to calculate the probability that the PCFG generates a string of exactly 10 words?

- a. (i).
- b. This has two parses. The first uses  $VP \rightarrow VP \ Adverb$ ,  $VP \rightarrow Copula \ Adjective$ ,  $Copula \rightarrow is$ ,  $Adjective \rightarrow well$ ,  $Adverb \rightarrow well$ . Its probability is

$$0.2 \times 0.2 \times 0.8 \times 0.5 \times 0.5 = 0.008 .$$

The second uses  $VP \rightarrow VP \ Adverb$  twice,  $VP \rightarrow Verb$ ,  $Verb \rightarrow is$ , and  $Adverb \rightarrow well$  twice. Its probability is

$$0.2 \times 0.2 \times 0.1 \times 0.5 \times 0.5 \times 0.5 = 0.0005 .$$

The total probability is 0.0085.

- c. It exhibits both lexical and syntactic ambiguity.
- d. True. There can only be finitely many ways to generate the finitely many strings of 10 words.

### Exercise 23.2.#NPGR

Consider the following simple PCFG for noun phrases:

$$\begin{aligned}
 0.6 : NP &\rightarrow Det\ AdjString\ Noun \\
 0.4 : NP &\rightarrow Det\ NounNounCompound \\
 0.5 : AdjString &\rightarrow Adj\ AdjString \\
 0.5 : AdjString &\rightarrow \Lambda \\
 1.0 : NounNounCompound &\rightarrow Noun\ Noun \\
 0.8 : Det &\rightarrow \text{the} \\
 0.2 : Det &\rightarrow \text{a} \\
 0.5 : Adj &\rightarrow \text{small} \\
 0.5 : Adj &\rightarrow \text{green} \\
 0.6 : Noun &\rightarrow \text{village} \\
 0.4 : Noun &\rightarrow \text{green}
 \end{aligned}$$

where  $\Lambda$  denotes the empty string.

- a. What is the longest NP that can be generated by this grammar? (i) three words (ii) four words (iii) infinitely many words
- b. Which of the following have a nonzero probability of being generated as complete NPs?  
 (i) a small green village (ii) a green green green (iii) a small village green
- c. What is the probability of generating “the green green”?
- d. What types of ambiguity are exhibited by the phrase in (c)?
- e. Given any PCFG and any finite word sequence, is it possible to calculate the probability that the sequence was generated by the PCFG?

- a. (iii): we can have infinitely long *AdjStrings*, although the probability decreases exponentially with length.
- b. (i) and (ii).
- c. “The green green” can be generated two ways, so we add the probabilities. If the first green is an adjective, the probability is  $.6 \times .5 \times .5 \times .5 \times .5 \times .4 = 0.015$ . If the first green is a noun, the probability is  $.4 \times .5 \times 1.0 \times .5 \times .5 = 0.05$ . So the total probability is 0.065.
- d. The phrase “the green green” exhibits lexical ambiguity (“green” has two meanings) and syntactic ambiguity (there are two distinct parses).
- e. This simply requires summing over all possible parses, and there are finitely many of them. CFGs are considerably less powerful than Turing machines.

## 23.3 Parsing

### Exercise 23.3.#SOWS

Consider the sentence “Someone walked slowly to the supermarket” and a lexicon consisting of the following words:

$$\begin{array}{ll} \text{Pronoun} \rightarrow \text{someone} & \text{Verb} \rightarrow \text{walked} \\ \text{Adv} \rightarrow \text{slowly} & \text{Prep} \rightarrow \text{to} \\ \text{Article} \rightarrow \text{the} & \text{Noun} \rightarrow \text{supermarket} \end{array}$$

Which of the following three grammars, combined with the lexicon, generates the given sentence? Show the corresponding parse tree(s).

(A):

$$\begin{array}{l} S \rightarrow NP \ VP \\ NP \rightarrow Pronoun \\ NP \rightarrow Article \ Noun \\ VP \rightarrow VP \ PP \\ VP \rightarrow VP \ Adv \ Adv \\ VP \rightarrow Verb \\ PP \rightarrow Prep \ NP \\ NP \rightarrow Noun \end{array}$$

(B):

$$\begin{array}{l} S \rightarrow NP \ VP \\ NP \rightarrow Pronoun \\ NP \rightarrow Noun \\ NP \rightarrow Article \ NP \\ VP \rightarrow Verb \ Vmod \\ VP \rightarrow Adv \ Vmod \\ Vmod \rightarrow Adv \ Vmod \\ Vmod \rightarrow Adv \\ Adv \rightarrow PP \\ PP \rightarrow Prep \ NP \\ PP \rightarrow Prep \ NP \end{array}$$

(C):

$$\begin{array}{l} S \rightarrow NP \ VP \\ NP \rightarrow Pronoun \\ NP \rightarrow Article \ NP \\ VP \rightarrow Verb \ Adv \\ Adv \rightarrow Adv \ Adv \\ Adv \rightarrow PP \\ PP \rightarrow Prep \ NP \\ NP \rightarrow Noun \end{array}$$

For each of the preceding three grammars, write down three sentences of English and three sentences of non-English generated by the grammar. Each sentence should be significantly different, should be at least six words long, and should include some new lexical entries (which you should define). Suggest ways to improve each grammar to avoid generating the non-English sentences.

Grammar (A) does not work, because there is no way for the verb “walked” followed by the adverb “slowly” and the prepositional phrase “to the supermarket” to be parsed as a verb phrase. A verb phrase in (A) must have either two adverbs or be just a verb. Here is the parse under grammar (B):

```
S---NP---Pro---Someone
 |
 |---VP---V---walked
 |
 |---Vmod---Adv---slowly
 |
 |---Vmod---Adv---PP---Prep---to
 |
 |---NP---Det---the
 |
 |---NP---Noun---supermarket
```

Here is the parse under grammar (C):

```
S---NP---Pro---Someone
 |
```

```

| -VP---V---walked
|
| -Adv---Adv---slowly
|
| -Adv---PP---Prep---to
|
| -NP---Det---the
|
| -NP---Noun---supermarket

```

### Exercise 23.3.#TOYG

Consider the following toy grammar:

$$\begin{aligned}
 S &\rightarrow NP\ VP \\
 NP &\rightarrow Noun \\
 NP &\rightarrow NP\ \textbf{and}\ NP \\
 NP &\rightarrow NP\ PP \\
 VP &\rightarrow Verb \\
 VP &\rightarrow VP\ \textbf{and}\ VP \\
 VP &\rightarrow VP\ PP \\
 PP &\rightarrow Prep\ NP
 \end{aligned}$$

$$Noun \rightarrow \textbf{Sally} \mid \textbf{pools} \mid \textbf{streams} \mid \textbf{swims}$$

$$Prep \rightarrow \textbf{in}$$

$$Verb \rightarrow \textbf{pools} \mid \textbf{streams} \mid \textbf{swims}$$

- Show all the parse trees in this grammar for the sentence “Sally swims in streams and pools.”
- Show all the table entries that would be made by a (non-probabalistic) CYK parser on this sentence.

- a. There is a unique parse. Using the grammar in part two of the answer to have at most binary branching, this is:

```

S---NP---Noun---Sally
|
| -VP---VP---V---swims
|
| -PP---Prep---in
|
| -NP---NP---Noun---streams
|
| -ANP---A---and
|
| ---NP---Noun---pools

```

- b. To run CYK on this sentence we first need to convert the grammar into Chomsky Normal Form. For simplicity we'll allow unary rules  $X \rightarrow Y$  which CYK can be adapted to handle. This gives us the grammar:

$$\begin{aligned}
 S &\rightarrow NP\ VP \\
 NP &\rightarrow N \\
 NP &\rightarrow NP\ \textbf{ANP}
 \end{aligned}$$

$$ANP \rightarrow \mathbf{A} \ NP$$

$$NP \rightarrow NP \ PP$$

$$VP \rightarrow V$$

$$VP \rightarrow VP \ \mathbf{AVP}$$

$$AVP \rightarrow A \ VP$$

$$VP \rightarrow VP \ PP$$

$$PP \rightarrow P \ NP$$

$$N \rightarrow \mathbf{Sally} \mid \mathbf{pools} \mid \mathbf{streams} \mid \mathbf{swims}$$

$$P \rightarrow \mathbf{in}$$

$$V \rightarrow \mathbf{pools} \mid \mathbf{streams} \mid \mathbf{swims}$$

$$A \rightarrow \mathbf{and}$$

Running the algorithm results in the table in Figure S23.3.

|       |       |    |       |    |      |
|-------|-------|----|-------|----|------|
| S     |       |    |       |    |      |
|       | VP    |    |       |    |      |
| S     |       | PP |       |    |      |
|       | VP    |    |       | NP |      |
| S     |       | PP |       |    | ANP  |
| N, NP | V, VP | P  | N, NP | A  | N,NP |

Sally      swims      in      streams      and      pools

Figure S23.3 CYK parse table.

## 23.4 Augmented Grammars

### Exercise 23.4.#ACFG

An augmented context-free grammar can represent languages that a regular context-free grammar cannot. Show an augmented context-free grammar for the language  $a^n b^n c^n$ . The allowable values for augmentation variables are 1 and  $\text{SUCCESSOR}(n)$ , where  $n$  is a value.

The rule for a sentence in this language is

$$S(n) \rightarrow A(n) B(n) C(n).$$

Show the rule(s) for each of  $A$ ,  $B$ , and  $C$ .

The rule for  $A$  is

$$\begin{aligned} A(n') &\rightarrow aA(n) \{n' = \text{SUCCESSOR}(n)\} \\ A(1) &\rightarrow a \end{aligned}$$

The rules for  $B$  and  $C$  are similar.

### Exercise 23.4.#ENGO

Augment the  $\mathcal{E}_1$  grammar so that it handles article–noun agreement. That is, make sure that “agents” and “an agent” are  $NPs$ , but “agent” and “an agents” are not.

$$\begin{aligned} NP(case, number, Third) &\rightarrow Name(number) \\ NP(case, Plural, Third) &\rightarrow Noun(Plural) \\ NP(case, number, Third) &\rightarrow Article(number)Noun(number) \\ Article(Singular) &\rightarrow \mathbf{a} \mid \mathbf{an} \mid \mathbf{the} \\ Article(Plural) &\rightarrow \mathbf{the} \mid \mathbf{some} \mid \mathbf{many} \end{aligned}$$

### Exercise 23.4.#DCGN

Using DCG notation, write a grammar for a language that is just like  $\mathcal{E}_1$ , except that it enforces agreement between the subject and verb of a sentence and thus does not generate ungrammatical sentences such as “I smells the wumpus.”

Here is a partial DCG. We include both person and number annotation although English really only differentiates the third person singular for verb agreement (except for the verb *be*).

$$\begin{aligned}
 S &\rightarrow NP(\text{Subjective}, \text{number}, \text{person}) VP(\text{number}, \text{person}) \mid \dots \\
 NP(\text{case}, \text{number}, \text{person}) &\rightarrow \text{Pronoun}(\text{case}, \text{number}, \text{person}) \\
 NP(\text{case}, \text{number}, \text{Third}) &\rightarrow \text{Name}(\text{number}) \mid \text{Noun}(\text{number}) \mid \dots \\
 VP(\text{number}, \text{person}) &\rightarrow VP(\text{number}, \text{person}) NP(\text{Objective}, \text{-, -}) \mid \dots \\
 PP &\rightarrow \text{Preposition } NP(\text{Objective}, \text{-, -}) \\
 \text{Pronoun}(\text{Subjective}, \text{Singular}, \text{First}) &\rightarrow \mathbf{I} \\
 \text{Pronoun}(\text{Subjective}, \text{Singular}, \text{Second}) &\rightarrow \mathbf{you} \\
 \text{Pronoun}(\text{Subjective}, \text{Singular}, \text{Third}) &\rightarrow \mathbf{he} \mid \mathbf{she} \mid \mathbf{it} \\
 \text{Pronoun}(\text{Subjective}, \text{Plural}, \text{First}) &\rightarrow \mathbf{we} \\
 \text{Pronoun}(\text{Subjective}, \text{Plural}, \text{Second}) &\rightarrow \mathbf{you} \\
 \text{Pronoun}(\text{Subjective}, \text{Plural}, \text{Third}) &\rightarrow \mathbf{they} \\
 \text{Pronoun}(\text{Objective}, \text{Singular}, \text{First}) &\rightarrow \mathbf{me} \\
 \text{Pronoun}(\text{Objective}, \text{Singular}, \text{Second}) &\rightarrow \mathbf{you} \\
 \text{Pronoun}(\text{Objective}, \text{Singular}, \text{Third}) &\rightarrow \mathbf{him} \mid \mathbf{her} \mid \mathbf{it} \\
 \text{Pronoun}(\text{Objective}, \text{Plural}, \text{First}) &\rightarrow \mathbf{us} \\
 \text{Pronoun}(\text{Objective}, \text{Plural}, \text{Second}) &\rightarrow \mathbf{you} \\
 \text{Pronoun}(\text{Objective}, \text{Plural}, \text{Third}) &\rightarrow \mathbf{them} \\
 \text{Verb}(\text{Singular}, \text{First}) &\rightarrow \mathbf{smell} \\
 \text{Verb}(\text{Singular}, \text{Second}) &\rightarrow \mathbf{smell} \\
 \text{Verb}(\text{Singular}, \text{Third}) &\rightarrow \mathbf{smells} \\
 \text{Verb}(\text{Plural}, \text{-}) &\rightarrow \mathbf{smell}
 \end{aligned}$$

### Exercise 23.4.#PSFG

Consider the following PCFG:

$$\begin{aligned}
 S &\rightarrow NP \ VP [1.0] \\
 NP &\rightarrow \text{Noun} [0.6] \mid \text{Pronoun} [0.4] \\
 VP &\rightarrow \text{Verb} \ NP [0.8] \mid \text{Modal Verb} [0.2] \\
 \\ 
 \text{Noun} &\rightarrow \mathbf{can} [0.1] \mid \mathbf{fish} [0.3] \mid \dots \\
 \text{Pronoun} &\rightarrow \mathbf{I} [0.4] \mid \dots \\
 \text{Verb} &\rightarrow \mathbf{can} [0.01] \mid \mathbf{fish} [0.1] \mid \dots \\
 \text{Modal} &\rightarrow \mathbf{can} [0.3] \mid \dots
 \end{aligned}$$

The sentence “I can fish” has two parse trees with this grammar. Explain the semantic difference between the two. Show the two trees, their prior probabilities, and their conditional probabilities, given the sentence.

One parse captures the meaning “I am able to fish” and the other “I put fish in cans.” Both have the left branch  $NP \rightarrow \text{Pronoun} \rightarrow \mathbf{I}$ , which has probability 0.16.

- The first has the right branch  $VP \rightarrow \text{Modal Verb} (0.2)$  with  $\text{Modal} \rightarrow \mathbf{can} (0.3)$  and  $\text{Verb} \rightarrow \mathbf{fish} (0.1)$ , so its prior probability is

$$0.16 \times 0.2 \times 0.3 \times 0.1 = 0.00096.$$

- The second has the right branch  $VP \rightarrow VerbNP$  (0.8) with  $Verb \rightarrow \text{can}$  (0.1) and  $NP \rightarrow Noun \rightarrow \text{fish}$  ( $0.6 \times 0.3$ ), so its prior probability is

$$0.16 \times 0.8 \times 0.1 \times 0.6 \times 0.3 = 0.002304.$$

As these are the only two parses, and the conditional probability of the string given the parse is 1, their conditional probabilities given the string are proportional to their priors and sum to 1: 0.294 and 0.706.

## 23.5 Complications of Real Natural Language

---

### Exercise 23.5.#TMGM

Collect some examples of time expressions, such as “two o’clock,” “midnight,” and “12:46.” Also think up some examples that are ungrammatical, such as “thirteen o’clock” or “half past two fifteen.” Write a grammar for the time language.

Here is a start of a grammar:

```
Time => DigitHour ":" DigitMinute
| "midnight" | "noon" | "12 midnight" | "12 noon"
| ClockHour "o'clock"
| Difference BeforeAfter ExtendedHour

DigitHour => 0 | 1 | ... | 23
DigitMinute => 1 | 2 | ... | 60
HalfDigitMinute => 1 | 2 | ... | 29
ClockHour => ClockDigitHour | ClockWordHour
ClockDigitHour => 1 | 2 | ... | 12
ClockWordHour => "one" | ... | "twelve"
BeforeAfter => "to" | "past" | "before" | "after"
Difference => HalfDigitMinute "minutes" | ShortDifference
ShortDifference => "five" | "ten" | "twenty" | "twenty-five" | "quarter" | "half"
ExtendedHour => ClockHour | "midnight" | "noon"
```

The grammar is not perfect; for example, it allows “ten before six” and “quarter past noon,” which are a little odd-sounding, and “half before six,” which is not really OK.

### Exercise 23.5.#JAVA

Outline the major differences between a programming language such as Java and a natural language such as English, commenting on the “understanding” problem in each case. Think about such things as grammar, syntax, semantics, pragmatics, compositionality, context-dependence, lexical ambiguity, syntactic ambiguity, reference finding (including pronouns), background knowledge, and what it means to “understand” in the first place.

The purpose of this exercise is to get the student thinking about the properties of natural language. There is a wide variety of acceptable answers. Here are ours:

- **Grammar and Syntax** Java: formally defined in a reference book. Grammaticality is crucial; ungrammatical programs are not accepted. English: unknown, never formally defined, constantly changing. Most communication is made with “ungrammatical” utterances. There is a notion of graded acceptability: some utterances are judged slightly ungrammatical or a little odd, while others are clearly right or wrong.
- **Semantics** Java: the semantics of a program is formally defined by the language specification. More pragmatically, one can say that the meaning of a particular program is the JVM code emitted by the compiler. English: no formal semantics, meaning is context dependent.
- **Pragmatics and Context-Dependence** Java: some small parts of a program are left undefined in the language specification, and are dependent on the computer on which the program is run. English: almost everything about an utterance is dependent on the situation of use.
- **Compositionality** Java: almost all compositional. The meaning of “A + B” is clearly derived from the meaning of “A” and the meaning of “B” in isolation. English: some compositional parts, but many non-compositional dependencies.
- **Lexical Ambiguity** Java: a symbol such as “Avg” can be locally ambiguous as it might refer to a variable, a class, or a function. The ambiguity can be resolved simply by checking the declaration; declarations therefore fulfill in a very exact way the role played by background knowledge and grammatical context in English. English: much lexical ambiguity.
- **Syntactic Ambiguity** Java: the syntax of the language resolves ambiguity. For example, in “if (X) if (Y) A; else B;” one might think it is ambiguous whether the “else” belongs to the first or second “if,” but the language is specified so that it always belongs to the second. English: much syntactic ambiguity.
- **Reference** Java: there is a pronoun “this” to refer to the object on which a method was invoked. Other than that, there are no pronouns or other means of indexical reference; no “it,” no “that.” (Compare this to stack-based languages such as Forth, where the stack pointer operates as a sort of implicit “it.”) There is reference by name, however. Note that ambiguities are determined by scope—if there are two or more declarations of the variable “X”, then a use of X refers to the one in the innermost scope surrounding the use. English: many techniques for reference.
- **Background Knowledge** Java: none needed to interpret a program, although a local “context” is built up as declarations are processed. English: much needed to do disambiguation.
- **Understanding** Java: understanding a program means translating it to JVM byte code. English: understanding an utterance means (among other things) responding to it appropriately; participating in a dialog (or choosing not to participate, but having the potential ability to do so).

As a follow-up question, you might want to compare different languages, for example: English, Java, Morse code, the SQL database query language, the Postscript document description language, mathematics, etc.

**Exercise 23.5.#NYTS**

Consider the following sentence (from *The New York Times*, July 28, 2008):

Banks struggling to recover from multibillion-dollar loans on real estate are curtailing loans to American businesses, depriving even healthy companies of money for expansion and hiring.

- a. Which of the words in this sentence are lexically ambiguous?
- b. Find two cases of syntactic ambiguity in this sentence (there are more than two.)
- c. Give an instance of metaphor in this sentence.
- d. Can you find semantic ambiguity?

- a. Webster's New Collegiate Dictionary (9th edn.) lists multiple meaning for all these words except "multibillion" and "curtailing".
- b. The attachment of all the propositional phrases is ambiguous, e.g. does "from ... loans" attach to "struggling" or "recover"? Does "of money" attach to "depriving" or "companies"? The coordination of "and hiring" is also ambiguous; is it coordinated with "expansion" or with "curtailing" and "depriving" (using British punctuation).
- c. The most clear-cut case is "healthy companies" as an example of HEALTH for IN A GOOD FINANCIAL STATE. Other possible metaphors include "Banks ... recover" (same metaphor as "healthy"), "banks struggling" (PHYSICAL EFFORT for WORK), and "expansion" (SPATIAL VOLUME for AMOUNT OF ACTIVITY); in these cases, the line between metaphor and polysemy is vague.

## **23.6 Natural Language Tasks**

---

**Exercise 23.6.#SEEN**

Create a test set of ten queries, and pose them to two different Web search engines. Evaluate each one for precision at the top 1, 3, and 10 documents. Can you explain the differences between engines?

Doing the evaluation is easy, if a bit tedious (requiring 200 evaluations for the complete 10 documents  $\times$  2 engines  $\times$  10 queries). Explaining the differences is more difficult. Some things to check are whether the good results in one engine are even in the other engines at all (by searching for unique phrases on the page); check whether the results are commercially sponsored, are produced by human editors, or are algorithmically determined by a search ranking algorithm; check whether each engine implements features such as spelling correction and synonyms.

**Exercise 23.6.#WBCO**

Estimate how much storage space is necessary for the index to a 100 billion-page corpus of Web pages. Show the assumptions you made.

Computations like this are given in the book *Managing Gigabytes* (Witten *et al.*, 1999). Here's one way of doing the computation: Assume an average page is about 10KB (giving us a 10TB corpus), and that index size is linear in the size of the corpus. Bahle *et al.* (2002) show an index size of about 2GB for a 22GB corpus; so our billion page corpus would have an index of about 1TB.

**Exercise 23.6.#REGX**

Write a regular expression or a short program to extract company names. Test it on a corpus of business news articles. Report your recall and precision.

Code not shown. The simplest approach is to look for a string of capitalized words, followed by “Inc” or “Co.” or “Ltd.” or similar markers. A more complex approach is to get a list of company names (e.g. from an online stock service), look for those names as exact matches, and also extract patterns from them. Reporting recall and precision requires a clearly-defined corpus.

**Exercise 23.6.#WESE**

Consider the problem of trying to evaluate the quality of an Information Retrieval system that returns a ranked list of answers (like most Web search engines). The appropriate measure of quality depends on the presumed model of what the searcher is trying to achieve, and what strategy she employs. For each of the following models, propose a corresponding numeric measure.

- a. The searcher will look at the first twenty answers returned, with the objective of getting as much relevant information as possible.
- b. The searcher needs only one relevant document, and will go down the list until she finds the first one.
- c. The searcher has a fairly narrow query and is able to examine all the answers retrieved. She wants to be sure that she has seen everything in the document collection that is relevant to her query. (E.g., a lawyer wants to be sure that she has found *all* relevant precedents, and is willing to spend considerable resources on that.)
- d. The searcher needs just one document relevant to the query, and can afford to pay a research assistant for an hour's work looking through the results. The assistant can look through 100 retrieved documents in an hour. The assistant will charge the searcher for the full hour regardless of whether he finds it immediately or at the end of the hour.
- e. The searcher will look through all the answers. Examining a document has cost \$A; finding a relevant document has value \$B; failing to find a relevant document has cost

$\$C$  for each relevant document not found.

- f. The searcher wants to collect as many relevant documents as possible, but needs steady encouragement. She looks through the documents in order. If the documents she has looked at so far are mostly good, she will continue; otherwise, she will stop.

- Use the precision on the first 20 documents returned.
- Use the reciprocal rank of the first relevant document. Or just the rank, considered as a cost function (large is bad).
- Use the recall.
- Score this as 1 if the first 100 documents retrieved contain at least one relevant to the query and 0 otherwise.
- Score this as  $(A(R + I) + BR - NC)$  where  $R$  is the number of relevant documents retrieved,  $I$  is the number of irrelevant documents retrieved, and  $C$  is the number of relevant documents not retrieved.
- One model would be a probabilistic one, in which, if the user has seen  $R$  relevant documents and  $I$  irrelevant ones, she will continue searching with probability  $p(R, I)$  for some function  $p$ , to be specified. The measure of quality is then the expected number of relevant documents examined.

### Exercise 23.6.#TRAN

Select five sentences and submit them to an online translation service. Translate them from English to another language and back to English. Rate the resulting sentences for grammaticality and preservation of meaning. Repeat the process; does the second round of iteration give worse results or the same results? Does the choice of intermediate language make a difference to the quality of the results? If you know a foreign language, look at the translation of one paragraph into that language. Count and describe the errors made, and conjecture why these errors were made.

The main point of this exercise is to show that current translation software is useful but not perfect. The mistakes made are often amusing for students.

### Exercise 23.6.#TXUT

Without looking back at Exercise 23.TXUN, answer the following questions:

- What are the four steps that are mentioned?
- What step is left out?
- What is “the material” that is mentioned in the text?
- What kind of mistake would be expensive?
- Is it better to do too few things or too many? Why?

This is a very difficult exercise—most readers have no idea how to answer the questions (except perhaps to remember that “too few” is better than “too many”). This is the whole point of the exercise, as we will see in Exercise 23.TXUZ.

### Exercise 23.6.#TXUZ

We forgot to mention that the text in Exercise 23.TXUN is entitled “Washing Clothes.” Reread the text and answer the questions again:

- a. What are the four steps that are mentioned?
- b. What step is left out?
- c. What is “the material” that is mentioned in the text?
- d. What kind of mistake would be expensive?
- e. Is it better to do too few things or too many? Why?

Did you do better this time? Bransford and Johnson (1973) used this text in a controlled experiment and found that the title helped significantly. What does this tell you about how language and memory works?

Now we can answer the difficult questions of 22.7:

- The steps are sorting the clothes into piles (e.g., white vs. colored); going to the washing machine (optional); taking the clothes out and sorting into piles (e.g., socks versus shirts); putting the piles away in the closet or bureau.
- The actual running of the washing machine is never explicitly mentioned, so that is one possible answer. One could also say that drying the clothes is a missing step.
- The material is clothes and perhaps other washables.
- Putting too many clothes together can cause some colors to run onto other clothes and ruin them.
- It is better to do too few: so they won’t run; so they get thoroughly cleaned; so they don’t cause the machine to become unbalanced.

# EXERCISES 24

## DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

### 24.1 Word Embeddings

---

#### Exercise 24.1.#WEVZ

Run a word embedding visualization tool such as [projector.tensorflow.org/](https://projector.tensorflow.org/) and try to get a feel for how the embeddings work: what words are near each other? Are there surprises for unrelated words that are near or related words that are far apart? Report on your findings. Consider:

- a. Common concrete nouns like “people” or “year” or “dog.”
- b. The most common words, which carry little meaning: “the,” “be,” “of,” “to,” etc.
- c. Abstract concepts like “configuration” or “topological.”
- d. Words that are part of sequential series such as numbers, months, presidents, days of the week (is “Monday” closer to “Sunday” or “Tuesday” or “Friday”?).
- e. Words that are part of unordered groups such as “France” or “Maine” or “lion.”
- f. Ambiguous words (is “Turkey” grouped with countries or birds?).
- g. UMAP versus T-SNE versus PCA visualizations.
- h. What words are at the periphery of the embedding? Near the center? Is that significant? Try changing the principal components that are mapped to each of the XYZ axes in PCA and see what difference that makes.
- i. What else can you explore?

Each student will take their own path in exploring this question.

#### Exercise 24.1.#WEMD

Run a notebook to generate a word embedding model such as [www.tensorflow.org/text/guide/word\\_embeddings](https://www.tensorflow.org/text/guide/word_embeddings), which trains an embedding model based on a corpus of IMDB movie reviews. Create the embedding model and visualize the embeddings. Then create another model based on a different text corpus and compare. What similarities and differences do you notice?

If you are not able to run a notebook, you can compare five different pre-built word

embeddings at [vectors.nlpl.eu/explore/embeddings/en/associates/](http://vectors.nlpl.eu/explore/embeddings/en/associates/).

The specific vocabulary of movies, especially the names of actors and directors, will not be present in another corpus. However for two large corpora, many of the most common words will look similar in the embedding model.

### Exercise 24.1.#WEAB

Examine how well word embedding models can answer analogy questions of the form “**A** is to **B** as **C** is to [what]?” (e.g. “Athens is to Greece as Oslo is to Norway”) using vector arithmetic. Create your own embeddings or use an online site such as [lamyiowce.github.io/word2viz/](https://lamyiowce.github.io/word2viz/) or [bionlp-www.utu.fi/wv\\_demo/](https://bionlp-www.utu.fi/wv_demo/).

- a. What analogies work well, and what ones fail?
- b. In handling Country:Capital analogies, is there a problem with ambiguous country names, like “Turkey”?
- c. Do analogies start to fail for rarer words? For example, “one is to 1 as two is to [what]?” will reliably retrieve “2,” but how well does that hold for “ninety” or “thousand”?
- d. What research papers can help you understand word embedding analogies?
- e. What else can you explore?

Each student will take their own path in exploring this question. In general, it seems like analogies work best when there is a word affix (such “great is to greatest as tall is to tallest”) or when the words have very specific references in the real world (like “Oslo”) that are likely to have been mentioned in the training corpus, or when words naturally occur in an order (such as “one, two, three”). However, counting or arithmetic analogies only hold for the first few integers. Word vectors can’t represent everything there is to know about a word. Good papers include [aclanthology.org/S17-1001/](https://aclanthology.org/S17-1001/) and [arxiv.org/abs/1901.09813](https://arxiv.org/abs/1901.09813).

## 24.2 Recurrent Neural Networks for NLP

### Exercise 24.2.#RNCH

So far we’ve concentrated on word embeddings to represent the fundamental units of text. But it is also possible to have a **character-level model**. Read Andrej Karpathy’s 2015 article *The Unreasonable Effectiveness of Recurrent Neural Networks* and download his `char-rnn` code (the PyTorch version at [github.com/jcjohnson/torch-rnn](https://github.com/jcjohnson/torch-rnn) is easier to use than the original version at [github.com/karpathy/char-rnn](https://github.com/karpathy/char-rnn)). Train the model on text of your choice.

- a. Show how the randomly generated text improves with more iterations of training.
- b. Find any other interesting properties of the model on your text.

- c. Replicate some of Karpathy's findings, such as the fact that some cells learn to count nesting level of parentheses.
- d. Compare the character-level RNN model to a character-level  $n$ -gram model as described in Yoav Goldberg's *The unreasonable effectiveness of Character-level Language Models (and why RNNs are still cool)* at [nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139](http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139).
- e. What do you see as the differences between RNN and  $n$ -gram models?

Each student will take their own path in exploring this question. The main difference between models is that the  $n$ -gram model by definition has no memory of context that precedes the previous  $n$  characters, while the RNN can learn to selectively remember parts of the context (such as indentation and nesting).

### Exercise 24.2.#RNTF

Which of the following are true or false?

- a. A RNN is designed specifically for processing word sequences.
  - b. A RNN is designed to process any time-series data.
  - c. An RNN has a limit on how far into the past it can remember.
  - d. An  $n$ -gram model has a limit on how far into the past it can remember.
  - e. An RNN can look into the future.
  - f. An RNN must use LSTM units in order to copy information from one time step to the next.
- 
- a. False; RNN's work on character sequences as well as word sequences, and on any kind of time-series data.
  - b. True.
  - c. False. Information about any input can be passed on from one hidden layer to the next for any number of time steps.
  - d. True. An  $n$ -gram model can only look back  $n$  steps.
  - e. False. A single RNN cannot look at future inputs until their time arrives. But a bidirectional RNN, which consists of two RNNs put together, can work in both directions.
  - f. False. A regular RNN can copy information forward. The advantage of an LSTM is that it can learn what is important to copy, and what is not.

### Exercise 24.2.#RNPS

Apply an RNN to the task of **part of speech tagging**. Run a notebook (such as [www.kaggle.com/tanyadayanand/pos-tagging-using-rnn](http://www.kaggle.com/tanyadayanand/pos-tagging-using-rnn) or [github.com/roemmele/keras-rnn-notebooks/tree/master/pos\\_tagging](https://github.com/roemmele/keras-rnn-notebooks/tree/master/pos_tagging)), train

the model on some tagged sentences, and evaluate it on some test data. Compare the results to a non-recurrent POS tagger, such as [librarycarpentry.org/lc-tdm/13-part-of-speech-tagging-text/index.html](https://librarycarpentry.org/lc-tdm/13-part-of-speech-tagging-text/index.html).)

Students should achieve roughly 99% accuracy.

### Exercise 24.2.#RNTC

Apply an RNN to the task of **text classification**, in particular **binary sentiment analysis**: classifying movie reviews on IMDB as either positive or negative. Run a notebook such as [www.tensorflow.org/text/tutorials/text\\_classification\\_rnn](https://www.tensorflow.org/text/tutorials/text_classification_rnn) and report on your results.

- a. What level of accuracy can you achieve?
- b. Gather some reviews from another site, not IMDB, and see if the trained model performs as well on them.
- c. How does accuracy vary with the length of the review?
- d. Which words are most associated with a positive review? A negative review?

Each student will have their own approach to this exercise. To determine the most predictive (positive/negative) words, students can either examine the weights in their model, or they can vary the input: replace a word in the input with [UNK] and see how the predicted score changes. Repeat over all words.

## 24.3 Sequence-to-Sequence Models

### Exercise 24.3.#YAMS

This exercise is about the difficulty of translation, but does not use a large corpus, nor any complex algorithms—just your own ingenuity. A rare language spoken by only about a hundred people has an unusual number system that is not base ten. Below are the translations of the first ten cube numbers ( $1^3$  to  $10^3$ ) in this language, in some random order. Can you identify which phrase is which number, and how the number system works?

- a. eser tarumpao yuow ptae eser traowo eser
- b. eser traowo yuow
- c. naempr
- d. naempr ptae eser traowo eser
- e. naempr tarumpao yuow ptae yuow traowo naempr
- f. naempr traowo yempoka
- g. tarumpao
- h. yempoka tarumpao yempoka ptae naempr traowo yempoka

- i. yuow ptae yempoka traowo tampui
- j. yuow tarumpao yempoka ptae naempr traowo yuow

The number phrases are from the Ngkolmpu language in New Guinea, which uses base six. Base six makes sense because their staple diet is the yam, and the yam's elongated shape means that six of them fit together neatly in a circle. The numbers in order are:

- a. 1000, 27, 1, 64, 343, 8, 216, 512, 125, 729.

The vocabulary is:

- a. 1 = naempr
- b. 2 = yempoka
- c. 3 = yuow
- d. 4 = eser
- e. 5 = tampui
- f. 6 = traowo
- g.  $6^2 = 36 = \text{ptae}$
- h.  $6^3 = 216 = \text{tarumpao}$

This puzzle is due to Alex Bellos from his Monday math puzzle series at *The Guardian*.

- a.

### Exercise 24.3.#SSMT

Experiment with online sequence-to-sequence neural machine translation model, such as [translate.google.com/](https://translate.google.com/) or [www.bing.com/translator](https://www.bing.com/translator) or [www.deepl.com/translator](https://www.deepl.com/translator). If you know two languages well, look at translations between them. If you don't translate into a language and then back to the language you do know.

- a. How well does the system handle very rare words?
- b. Does the system properly rearrange the order of words when necessary?
- c. Do the translations encode biases?
- d. How does the system do on very long sentences (over 60 words)?

The main point of this exercise is to gain experience with MT systems, not specifically to answer the four parts, but here are answers:

- a. Rare words can be a problem. Systems that are character-based do better than strictly word-based systems at encoding things like verb endings.
- b. The attention mechanism does consistently well on local word reordering (e.g. translating “red pencil” as “crayon rouge”), but can have difficulties, say, when the subject of a 30-word sentence appears first in the source language and last in the target language.

- c. Some systems will encode the biases of their training corpora, for example assuming that the neutral phrase “the doctor” is translated to the masculine “el doctor” in Spanish. Other systems avoid this bias by offering two choices: “la doctora” and “el doctor.”
- d. Neural machine translation systems historically perform poorly on very long sentences. But perhaps there have been improvements by the time you do this exercise.

#### Exercise 24.3.#SSOP

Besides machine translation, describe some other tasks that can be solved by sequence-to-sequence models.

Here are some possibilities:

- a. Summarization: compressing a sentence, paragraph, or document into a shorter summary.
- b. Conversation: Chatbots such as Google’s Meena are built with seq2seq models.
- c. Interpretation: Mathematical problems such as integrals and differential equations can be mapped to their solutions with seq2seq models.
- d. Part of speech tagging: Seq2seq models may represent parts of speech latently as they go about their tasks, but they can be trained to output the POS labels.
- e. Parsing: As with parts of speech, seq2seq models may be trained to output a dependency structure (e.g. the current word has a dependency link to the word 4 words ago) that represents a parse tree.
- f. Morphology: A seq2seq model can be trained to invent the morphological form of a word with a specific purpose. For example, given the sequence “employ, NOUN, AGENT” it can output “employer,” and given “emply, ADJ” it can output “employable.”
- g. Drug discovery: descriptions of molecules can be mapped to a feature vector in multidimensional space; the space of molecules can then be examined to see which are nearby and which have certain relations that might indicate they will be useful as drugs for some purpose.

## 24.4 The Transformer Architecture

#### Exercise 24.4.#PRCV

Since the publication of the textbook, a new architecture called **Perceiver** was introduced by Jaegle *et al.* in their article *Perceiver: General Perception with Iterative Attention* arxiv.org/abs/2103.03206. Read the article and say which of the following are true or false.

- a. Perceiver is a kind of RNN model.
- b. Perceiver is a kind of CNN model.
- c. Perceiver is a kind of seq2seq model.
- d. Perceiver is a kind of Transformer model.
- e. Perceiver was specifically designed for large data sets.

- f. Perceiver was specifically designed for audio/video inputs.
  - g. Perceiver performs worse on computer vision tasks than a CNN model specifically designed for images, but Perceiver is more general.
  - h. Perceiver avoids overfitting.
- 
- a. Either True or False is an acceptable answer. A Perceiver can be interpreted as an RNN, with each layer in the network serving as a step. But it can also be seen as a version of a Transformer, which processes the input all at once, with positional encodings.
  - b. False. Perceiver does not use convolutional units; instead it directly attends to pixels.
  - c. True. Perceiver is a seq2seq model.
  - d. True. Perceiver is a variant of the Transformer model.
  - e. True. Perceiver has minimal inductive biases and wants the data to speak for itself; it is designed to be efficient with billions of examples, each with 100,000 input features.
  - f. False. Perceiver handles those modalities, but the architecture does not bias towards any particular type of input.
  - g. False. Perceiver's performance on image tasks is comparable to a CNN model, not worse.
  - h. False. Perceiver does indeed perform well on a variety of tasks, but its lack of inductive bias means it is susceptible to overfitting.

#### Exercise 24.4.#TRMT

Run a tutorial notebook such as [www.tensorflow.org/text/tutorials/transformer](http://www.tensorflow.org/text/tutorials/transformer) to train a Transformer model on a bilingual corpus to do machine translation. Test it to see how it performs. How does it handle unknown words?

The given tutorial does Portuguese/English translation, but other language pairs are possible. Following through the notebook should introduce students to all parts of the Transformer architecture. The model should mostly transliterate unknown words, but sometimes will change the spelling of an unknown word to match the letter frequencies of the target language, and will sometimes latch on to parts of more familiar words, as in translating “triceratops” to “trijacopters.”

#### Exercise 24.4.#TRSWS

We have considered recurrent models that work a word at a time, and models that work a character at a time. It is also possible to use a **subword representation**, in which, say, the word “searchability” is represented as two tokens, the word “search” followed by the suffix “-ability.”

Run a notebook such as [www.tensorflow.org/text/guide/subwords\\_tokenizer](http://www.tensorflow.org/text/guide/subwords_tokenizer) that allows you to build a subword tokenizer. Train it on some text. What

are the tokens that get represented? What words and what nonwords? How can you visualize the results?

It should be straightforward to run the given notebook. Common words get their own representation. Uncommon words or long words with common prefixes or suffixes are broken into subwords. Punctuation gets its own tokens. Helpful visualizations include plots that consider whether a token is a word or non-word, how frequently it occurs, and how many letters it contains.

## 24.5 Pretraining and Transfer Learning

### Exercise 24.5.#TFTR

Run a notebook such as [www.tensorflow.org/hub/tutorials/tf2\\_text\\_classification](http://www.tensorflow.org/hub/tutorials/tf2_text_classification) that loads a pre-trained text embedding as the first layer and does **transfer learning** for the domain, which in this case is text classification of movie reviews. How well does the transfer learning work?

The model should achieve roughly 85% accuracy on binary classification. The important point of the exercise is gaining experience running the model.

### Exercise 24.5.#TFWV

Run a notebook such as [www.tensorflow.org/tutorials/text/word2vec](http://www.tensorflow.org/tutorials/text/word2vec) that learns word embeddings from a corpus using the skip-gram approach. Train the model on a corpus of Shakespeare, and separately on a corpus of contemporary language. Then test the resulting word embeddings on a downstream task such as classification of movie reviews. Does one word embedding model outperform the other?

Students should be able to show that the contemporary corpus does better than the Shakespeare corpus for downstream tasks involving contemporary language.

## 24.6 State of the art

### Exercise 24.6.#SOQA

Choose a dataset from [paperswithcode.com/task/question-answering](http://paperswithcode.com/task/question-answering) and report on the NLP Question-Answering model that performs best on that dataset. It will be easier if you choose a dataset for which both a paper and code are available. What made the highest-scoring model successful and how does it compare to competing models?

The answer depends on the student's choice.

**Exercise 24.6.#SOGP**

Experiment with a large-scale NLP text generation system. Pretrained online versions come and go; you could try [6b.eleuther.ai/](https://6b.eleuther.ai/) or [transformer.huggingface.co/doc/distil-gpt2](https://transformer.huggingface.co/doc/distil-gpt2) or search for another one. Give the system some prompts and evaluate the text it generates.

- a. Is the generated text grammatical?
- b. Is it relevant to the prompt's topic?
- c. Is it internally consistent from sentence-to-sentence and paragraph-to-paragraph?

- a. Usually the text is grammatical with only a few errors.
- b. Usually it starts out relevant, but may wonder.
- c. Consistency is not guaranteed. Character names can change; points may be repeated; actions may occur out of order.

**Exercise 24.6.#GPMM**

Read [medium.com/@melaniemitchell.me/can-gpt-3-make-analogies-16436](https://medium.com/@melaniemitchell.me/can-gpt-3-make-analogies-16436) Melanie Mitchell's account of trying to replicate her 1980s work on analogy-making with a standard GPT-3 model. Mitchell's 1980s program used symbolic AI to answer questions like "If a b c changes to a b d, what does i j k change to" with "i j l." In some cases GPT-3 can replicate this behavior, using only its pretrained model plus about four lines of prompt showing how the pattern of question and answer work. Experiment with a large-scale NLP text generation system to see how well it does with analogies using letters and numbers like this.

Students should be able to replicate some successes, and also note many failures. With numbers instead of letters, small numbers work well, but language models will do worse than humans at recognizing patterns such as squares and cubes of numbers.

# EXERCISES 25

## COMPUTER VISION

### 25.1 Introduction

---

#### Exercise 25.1.#CVDT

Define the following terms in your own words.

- a. Active and passive sensing
- b. Image feature
- c. Object model
- d. Rendering model

a. Active and passive sensing: Passive sensing is what we normally do with our eyes—we receive stimulus from the world. Active sensing is what bats do with their ultrasound—send out a signal and detect what effect it has on the world. We can also do this with a flashlight.

- b. Image feature
- c. Object model
- d. Rendering model

#### Exercise 25.1.#OTCV

Name as many perceptual channels as you can that are used by robots today.

Vision, sound, touch (with capacitive or resistive sensors), force (with strain gauges, piezoelectric sensors, or other types), range finding (with ultrasonic or laser range finders), proximity sensing (with electromagnetic detector), temperature, acceleration, and global positioning have all been used by various robots.

### 25.2 Image Formation

---

#### Exercise 25.2.#PINH

In the shadow of a tree with a dense, leafy canopy, one sees a number of light spots. Surprisingly, they all appear to be circular. Why? After all, the gaps between the leaves

through which the sun shines are not likely to be circular.

The small spaces between leaves act as pinhole cameras. That means that the circular light spots you see are actually images of the circular sun. You can test this theory next time there is a solar eclipse: the circular light spots will have a crescent bite taken out of them as the eclipse progresses. (Eclipse or not, the light spots are easier to see on a sheet of paper than on the rough forest floor.)

### Exercise 25.2.#SPHR

Consider a picture of a white sphere floating in front of a black backdrop. The image curve separating white pixels from black pixels is sometimes called the “outline” of the sphere. Show that the outline of a sphere, viewed in a perspective camera, can be an ellipse. Why do spheres not look like ellipses to you?

Consider the set of light rays passing through the center of projection (the pinhole or the lens center), and tangent to the surface of the sphere. These define a double cone whose apex is the center of projection. Note that the outline of the sphere on the image plane is just the cross section corresponding to the intersection of this cone with the image plane of the camera. We know from geometry that such a conic section will typically be an ellipse. It is a circle in the special case that the sphere is directly in front of the camera (its center lies on the optical axis).

While on a planar retina, the image of an off-axis sphere would indeed be an ellipse, the human visual system tries to infer what is in the three-dimensional scene, and here the most likely solution is that one is looking at a sphere.

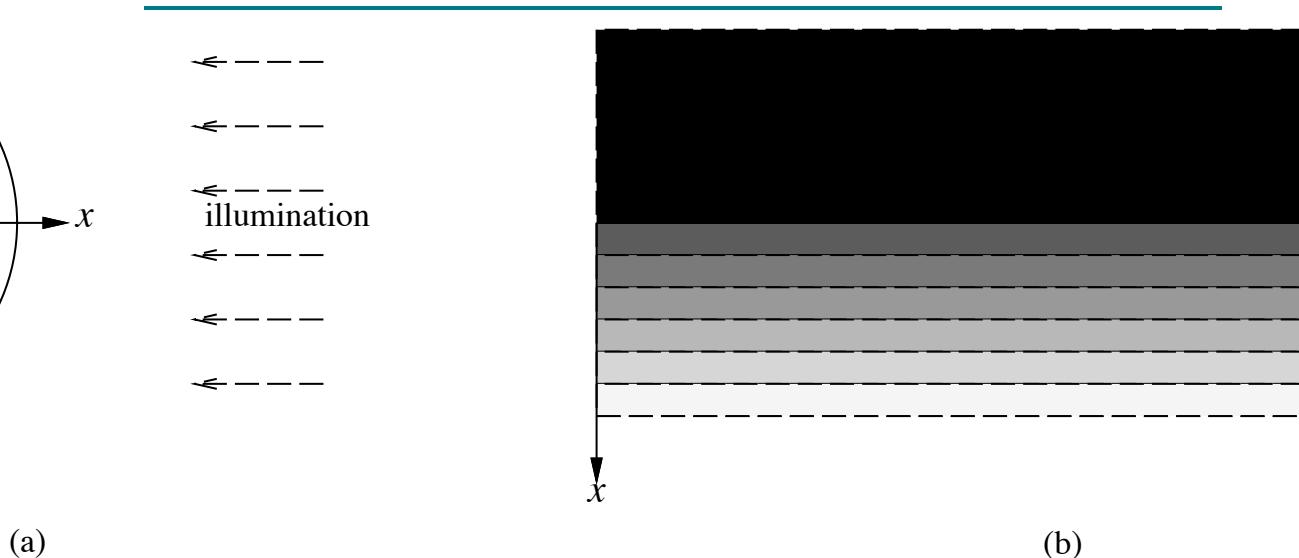
Some students might note that the eye’s retina is not planar but closer to spherical. On a perfectly spherical retina the image of a sphere will be circular. The point of the question remains valid, however.

### Exercise 25.2.#CORR

Consider an infinitely long cylinder of radius  $r$  oriented with its axis along the  $y$ -axis. The cylinder has a Lambertian surface and is viewed by a camera along the positive  $z$ -axis. What will you expect to see in the image if the cylinder is illuminated by a point source at infinity located on the positive  $x$ -axis? Draw the contours of constant brightness in the projected image. Are the contours of equal brightness uniformly spaced?

Recall that the image brightness of a Lambertian surface (page 743) is given by  $I(x, y) = k\mathbf{n}(x, y) \cdot \mathbf{s}$ . Here the light source direction  $\mathbf{s}$  is along the  $x$ -axis. It is sufficient to consider a horizontal cross-section (in the  $x$ - $z$  plane) of the cylinder as shown in Figure S25.1(a). Then, the brightness  $I(x) = k \cos \theta(x)$  for all the points on the right half of the cylinder. The left half is in shadow. As  $x = r \cos \theta$ , we can rewrite the brightness function as  $I(x) = \frac{kx}{r}$  which

reveals that the isobrightness contours in the lit part of the cylinder must be equally spaced. The view from the  $z$ -axis is shown in Figure S25.1(b).



**Figure S25.1** (a) Geometry of the scene as viewed from along the  $y$ -axis. (b) The scene from the  $z$ -axis, showing the evenly spaced isobrightness contours.

### Exercise 25.2.#DOFC

Normally we think of more depth of field as a good thing: we want more of the image in focus. But some photographers prize shallow depth of field, because it makes the subject of the photograph stand out from a blurred background. It is known that if two cameras with the same aperture number (say, f/2) take a picture of the same scene with the same angle of view, then the cameras with a small sensor (such as a cell phone) will have a deeper depth of field and less background blurring, while a large-sensor camera will have shallower depth of field and more background blurring. Why is that?

Depth of field is proportional to  $1/f^2$  where  $f$  is the focal length. So images with the same focal length (and a few other factors being equal) will have the same depth of field regardless of sensor size. But to achieve the same angle of view—the same scene—as a full-frame camera with a 28mm focal length, a cell phone camera would have a focal length of about 4mm. This makes sense: think of the diagram with the triangle tapering down from the image to the lens and then a similar but smaller triangle going to the sensor. To get the same angle of view a small sensor has to be close and a large sensor farther away. Another way to look at it is that an aperture number of f/2 means the actual lens aperture diameter is 28mm/2 or 14mm for the full-frame camera and 4mm/2 = 2mm for the cell phone. If the diameter is larger then there is more opportunity for light from slightly different image patches to be farther apart on the lens and thus farther apart when projected on the sensor.

**Exercise 25.2.#FTCL**

Does a small cell phone camera lens with an aperture of f/2 gather as much light from each patch in the scene as an f/2 lens on a large SLR camera that is taking an image of the same scene?

For each scene patch, a smaller lens gathers less light. But the smaller lens has a smaller focal length to achieve the same scene, so the light is concentrated into a smaller area on the sensor. Thus the amount of light per unit area is the same.

## **25.3 Simple Image Features**

---

**Exercise 25.3.#EDGE**

Edges in an image can correspond to a variety of events in a scene. Consider Figure 25.4 (page 887), and assume that it is a picture of a real three-dimensional scene. Identify ten different brightness edges in the image, and for each, state whether it corresponds to a discontinuity in (a) depth, (b) surface orientation, (c) reflectance, or (d) illumination.

We list the four classes and give two or three examples of each:

- a. *depth*: Between the top of the computer monitor and the wall behind it. Between the side of the clock tower and the sky behind it. Between the white sheets of paper in the foreground and the book and keyboard behind them.
- b. *surface normal*: At the near corner of the pages of the book on the desk. At the sides of the keys on the keyboard.
- c. *reflectance*: Between the white paper and the black lines on it. Between the “golden” bridge in the picture and the blue sky behind it.
- d. *illumination*: On the windowsill, the shadow from the center glass pane divider. On the paper with Greek text, the shadow along the left from the paper on top of it. On the computer monitor, the edge between the white window and the blue window is caused by different illumination by the CRT.

**Exercise 25.3.#WALZ**

The Waltz algorithm takes as input a list of the vertices in a sketch diagram of one or polyhedrons. Each vertex is described by the number of lines that meet and whether the lines are colinear. From that, the algorithm outputs a description of the 3D scene, by propagating constraints about the vertices through the connecting lines. Implement a version and report on how well it works.

An example implementation is in Chapter 17 of <https://github.com/norvig/paip-lisp>.

## 25.4 Classifying Images

### Exercise 25.4.#CLIM

Build an image classification model, by following a pre-built recipe, and report on what worked well and what you had problems with.

(One example: <https://www.kaggle.com/rohandeysarkar/ultimate-image-classification-guide-2020>)

Answers will vary by student.

### Exercise 25.4.#CVML

Compare three architectures for computer vision image classification: (1) convolutional neural networks (CNN), (2) vision transformers (ViT), and (3) multi-layer perceptron mixers (MLP-Mixer). Find relevant research and report on your findings.

Convolutional neural networks are emphasized in the book. Vision transformers are covered in <https://arxiv.org/abs/2010.11929>. Multi-layer perceptrons are covered in the key paper <https://arxiv.org/abs/2105.01601>.

CNNs have an inductive bias that is well-suited to understanding images: the idea that nearby pixels in the image are often nearby elements in the world, and that elements may appear at any location in the image. Vision transformers lack that inductive bias and thus do worse when trained on a small or medium number of images, but start to do well with hundreds of millions of images and can exceed CNN performance. The MLP-Mixer model emphasizes speed of training, which is important for very large models. It is said to achieve similar accuracy as other models with 1/3 the training time.

Students should go into more detail about the various differences.

## 25.5 Detecting Objects

### Exercise 25.5.#YOLO

Research the YOLO (You Only Look Once) approach to object detection, and compare it to the RCNN region proposal approach described in the book.

The key paper is <https://arxiv.org/abs/1506.02640v5> YOLO's big advantage is speed: it can operate in real time (45 frames per second). It is a nice advantage that the code is open source. It achieves this impressive speed by training a single end-to-end network that predicts bounding boxes and category labels all at once. YOLO is not quite as accurate as RCNN methods at locating boxes with great accuracy, but it is resistant to false positives coming from background noise.

**Exercise 25.5.#MAPV**

Object detectors are commonly evaluated with a metric known as mean average precision (mAP). Research this method and explain how it works.

We explained in the book how the “area under the curve” (AUC) metric works: it graphs recall on the x-axis and precision on the y-axis and measures the area under this curve. Average precision (AP) does this, but instead of computing the exact area by integration, it samples the precision at a number of recall points and sums the areas of rectangles whose height is the precision and whose width is the difference from one sample point to the next. The mean average precision (mAP) is then the average of the AP scores over all object classes.

**Exercise 25.5.#CNTN**

A new object detection approach called CenterNet models an object as a single point, the center of its bounding box. Research the approach and report on your findings.

The key paper is <https://arxiv.org/abs/1904.07850> and the github repository is <https://github.com/xingyizhou/CenterNet>. CenterNet gives strictly less information than other approaches: it doesn’t provide the size or shape of the bounding box, or the pose. But its simplicity allows it to be faster, running in real time.

**Exercise 25.5.#I**

Image classification relies on labeled image data. Where does that come from? How accurate are the labels? How hard is it to create the labels?

Read <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/> to see how one machine learning researcher, Andrej Karpathy, spent time labeling ImageNet data and measured his own accuracy at 94%. Try the task yourself and see what accuracy you achieve. Can you think of better ways of collecting labels for images?

Each student will have their own reactions to this task.

## 25.6 The 3D World

**Exercise 25.6.#STRO**

A stereoscopic system is being contemplated for terrain mapping. It will consist of two CCD cameras, each having  $512 \times 512$  pixels on a  $10\text{ cm} \times 10\text{ cm}$  square sensor. The lenses to be used have a focal length of 16 cm, with the focus fixed at infinity. For corresponding points  $(u_1, v_1)$  in the left image and  $(u_2, v_2)$  in the right image,  $v_1 = v_2$  because the  $x$ -axes in the two image planes are parallel to the epipolar lines—the lines from the object to the

camera. The optical axes of the two cameras are parallel. The baseline between the cameras is 1 meter.

- If the nearest distance to be measured is 16 meters, what is the largest disparity that will occur (in pixels)?
- What is the distance resolution at 16 meters, due to the pixel spacing?
- What distance corresponds to a disparity of one pixel?

Before answering this exercise, we draw a diagram of the apparatus (top view), shown in Figure S25.2. Notice that we make the approximation that the focal length is the distance from the lens to the image plane; this is valid for objects that are far away. Notice that this question asks nothing about the  $y$  coordinates of points; we might as well have a single line of 512 pixels in each camera.

- Solve this by constructing similar triangles: whose hypotenuse is the dotted line from object to lens, and whose height is 0.5 meters and width 16 meters. This is similar to a triangle of width 16cm whose hypotenuse projects onto the image plane; we can compute that its height must be 0.5cm; this is the offset from the center of the image plane. The other camera will have an offset of 0.5cm in the opposite direction. Thus the total disparity is 1.0cm, or, at 512 pixels/10cm, a disparity of 51.2 pixels, or 51, since there are no fractional pixels. Objects that are farther away will have smaller disparity. Writing this as an equation, where  $d$  is the disparity in pixels and  $Z$  is the distance to the object, we have:

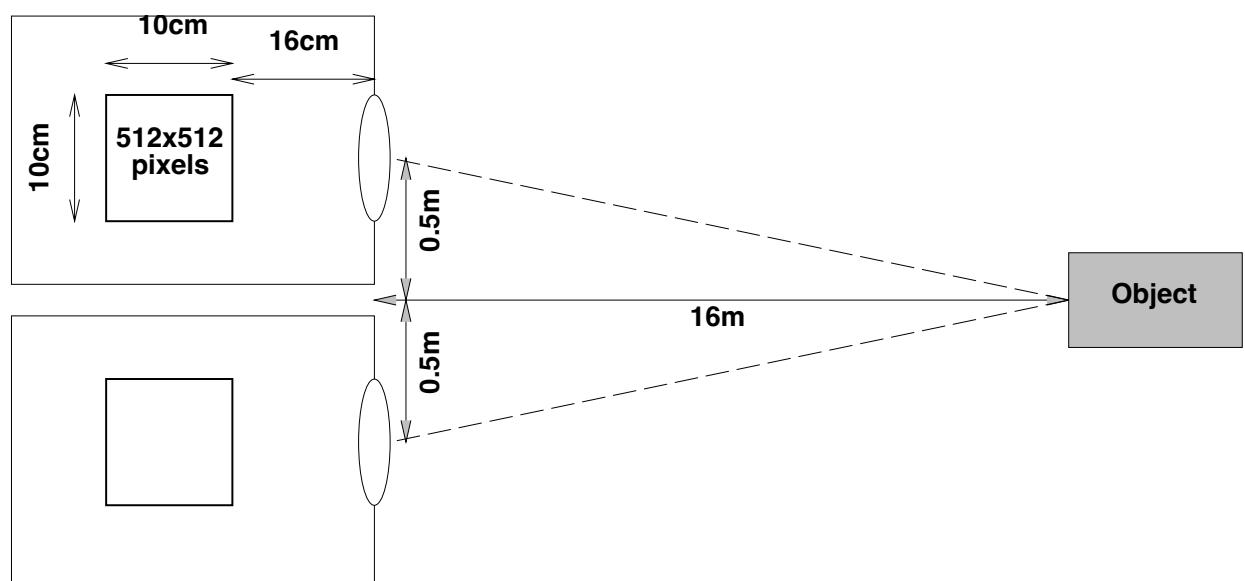
$$d = 2 \times \frac{512 \text{ pixels}}{10 \text{ cm}} \times 16 \text{ cm} \times \frac{0.5 \text{ m}}{Z}$$

- In other words, this question is asking how much further than 16m could an object be, and still occupy the same pixels in the image plane? Rearranging the formula above by swapping  $d$  and  $Z$ , and plugging in values of 51 and 52 pixels for  $d$ , we get values of  $Z$  of 16.06 and 15.75 meters, for a difference of 31cm (a little over a foot). This is the range resolution at 16 meters.
- In other words, this question is asking how far away would an object be to generate a disparity of one pixel? Objects farther than this are in effect out of range; we can't say where they are located. Rearranging the formula above by swapping  $d$  and  $Z$  we get 51.2 meters.

### Exercise 25.6.#TFCV

Which of the following are true, and which are false?

- Finding corresponding points in stereo images is the easiest phase of the stereo depth-finding process.
- Shape-from-texture can be done by projecting a grid of light-stripes onto the scene.



**Figure S25.2** Top view of the setup for stereo viewing (Exercise 24.6).

- c. Lines with equal lengths in the scene always project to equal lengths in the image.
- d. Straight lines in the image necessarily correspond to straight lines in the scene.

- a. False. This can be quite difficult, particularly when some points are occluded from one eye but not the other.
- b. True. The grid creates an apparent texture whose distortion gives good information as to surface orientation.
- c. False.
- d. False. A disk viewed edge-on appears as a straight line.

#### Exercise 25.6.#BOTT

(Courtesy of Pietro Perona.) Figure ?? shows two cameras at X and Y observing a scene. Draw the image seen at each camera, assuming that all named points are in the same horizontal plane. What can be concluded from these two images about the relative distances of points A, B, C, D, and E from the camera baseline, and on what basis?

A, B, C can be viewed in stereo and hence their depths can be measured, allowing the viewer to determine that B is nearest, A and C are equidistant and slightly further away. Neither D nor E can be seen by both cameras, so stereo cannot be used. Looking at the figure, it appears that the bottle *occludes* D from Y and E from X, so D and E must be further away than A, B, C, but their relative depths cannot be determined. There is, however, another

possibility (noticed by Alex Fabrikant). Remember that each camera sees the camera's-eye view not the bird's-eye view. X sees DABC and Y sees ABCE. It is possible that D is very close to camera X, so close that it falls outside the field of view of camera Y; similarly, E might be very close to Y and be outside the field of view of X. Hence, unless the cameras have a 180-degree field of view—probably impossible—there is no way to determine whether D and E are in front of or behind the bottle.

## 25.7 Using Computer Vision

### Exercise 25.7.#ODED

Suppose you have a CNN object detection algorithm that runs very well in a data center on a large cluster of computers. Now you want to deploy it onto cell phones (without draining users' batteries). What can you do?

A good survey is <https://arxiv.org/abs/1704.04861>, which describes MobileNets, which were designed for this purpose. Some things to try:

- a. Use a smaller network with fewer convolutional blocks and fewer parameters.
- b. Experiment with hyperparameters for the size of the model, such as a width multiplier, and understand the tradeoffs of accuracy versus model size, so that you can pick an appropriate size based on the available device.
- c. Quantize your network: Instead of using 32 bit or 64 bit floating point numbers, scale that down to 8 bit integers and reduce memory and CPU demand, at the cost of some loss in accuracy. Quantization is directly supported by platforms such as TensorFlow Mobile and Caffe2Go.
- d. You can reduce the resolution of images you process (and thus the size the network). You may only need 100 pixels on a side, not thousands of pixels.

### Exercise 25.7.#STYT

Examine one of the available code repositories for **style transfer**, such as [https://colab.research.google.com/github/tensorflow/models/blob/master/research/nst\\_blogpost/4\\_Neural\\_Style\\_Transfer\\_with\\_Eager\\_Execution.ipynb](https://colab.research.google.com/github/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb) or [https://www.tensorflow.org/tutorials/generative/style\\_transfer](https://www.tensorflow.org/tutorials/generative/style_transfer) or <https://github.com/hnarayanan/artistic-style-transfer> or <https://www.digitalocean.com/community/tutorials/how-to-perform-neural-style-transfer>. Try it on some combinations of content image and style reference images. Experiment with changing some aspects of the model, such as the number of content layers and style layers, or the loss functions. What works well and what doesn't work? What does "works well" even mean in this context?

The main point of this exercise is to get experience running networks in TensorFlow/Keras/PyTorch, and to think about the resulting output. Students will have different conclusions based on the images they work on and their interests. In general, styles that have distinctive profiles in

terms of low-level frequencies (as seen in Van Gogh, Kandinsky, and Hokusai) work best. “Works well” means that the viewer sees the result as interesting, and that the style and content are both recognizable.

### Exercise 25.7.#CRWD

Crowd counting is the task of estimating the number of people in an image (or images). Whenever there is a big event, organizers of the event tend to exaggerate the crowd size, and competitors try to minimize it. It would be helpful to have an accurate estimate.

Experiment with a crowd counting computer vision implementation such as one found at <https://github.com/topics/crowd-counting>. How well does it work?

Contrast the computer vision approach with an approach described in the article “Quantifying crowd size with mobile phone and Twitter data” (<https://royalsocietypublishing.org/doi/10.1098/rsos.150162>). Crowd counting has societal benefits in traffic control and public safety, but it also raises some ethical issues. Discuss.

The implementations give reasonable numbers, but it is difficult to establish ground truth. It is hard work to look at images and count them by hand, and we don’t know how accurate humans are at this task. Also, an image is just one view of a crowd at one instance in time, and hence is an incomplete record.

One ethical issue is that people are not very good at understanding large numbers, so promoting more use of numbers (as contrasted with other ways of covering an event) may not be helpful. The main issue is one of privacy: people expect to be able to attend an event without revealing their identity. Crowd counting does not detect identities, but it is a step towards more general surveillance.

The approach based on mobile phone data seems to be useful, particularly for events where it is difficult to obtain images (perhaps because it is dark or the event is indoors and aerial photos are not possible). Combining both methods may lead to better estimates.

# EXERCISES 26

## ROBOTICS

### 26.1 Robots

---

[[need exercises]]

### 26.2 Robot Hardware

---

[[need exercises]]

### 26.3 What kind of problem is robotics solving?

---

[[need exercises]]

### 26.4 Robotic Perception

---

#### Exercise 26.4.#MCLB

Monte Carlo localization is *biased* for any finite sample size—i.e., the expected value of the location computed by the algorithm differs from the true expected value—because of the way particle filtering works. In this question, you are asked to quantify this bias.

To simplify, consider a world with four possible robot locations:  $X = \{x_1, x_2, x_3, x_4\}$ . Initially, we draw  $N \geq 1$  samples uniformly from among those locations. As usual, it is perfectly acceptable if more than one sample is generated for any of the locations  $X$ . Let  $Z$  be a Boolean sensor variable characterized by the following conditional probabilities:

$$\begin{array}{ll} P(z | x_1) = 0.8 & P(\neg z | x_1) = 0.2 \\ P(z | x_2) = 0.4 & P(\neg z | x_2) = 0.6 \\ P(z | x_3) = 0.1 & P(\neg z | x_3) = 0.9 \\ P(z | x_4) = 0.1 & P(\neg z | x_4) = 0.9 . \end{array}$$

MCL uses these probabilities to generate particle weights, which are subsequently normalized and used in the resampling process. For simplicity, let us assume we generate only one new sample in the resampling process, regardless of  $N$ . This sample might correspond to any of the four locations in  $X$ . Thus, the sampling process defines a probability distribution over  $X$ .

- What is the resulting probability distribution over  $X$  for this new sample? Answer this question separately for  $N = 1, \dots, 10$ , and for  $N = \infty$ .

- b.** The difference between two probability distributions  $P$  and  $Q$  can be measured by the KL divergence, which is defined as

$$KL(P, Q) = \sum_i P(x_i) \log \frac{P(x_i)}{Q(x_i)}.$$

What are the KL divergences between the distributions in (a) and the true posterior?

- c.** What modification of the problem formulation (not the algorithm!) would guarantee that the specific estimator above is unbiased even for finite values of  $N$ ? Provide at least two such modifications (each of which should be sufficient).

To answer this question, consider all possibilities for the initial samples before and after resampling. This can be done because there are only finitely many states. The result for  $N = \infty$  is simply the posterior, calculated using Bayes rule.

```

int
main(int argc, char *argv[])
{
 // parse command line argument
 if (argc != 3){
 cerr << "Usage: " << argv[0] << " <number of samples>"
 << " <number of states>" << endl;
 exit(0);
 }

 int numSamples = atoi(argv[1]);
 int numStates = atoi(argv[2]);
 cerr << "number of samples: " << numSamples << endl;
 << "number of states: " << numStates << endl;
 assert(numSamples >= 1);
 assert(numStates >= 1);

 // generate counter
 int samples[numSamples];
 for (int i = 0; i < numSamples; i++)
 samples[i] = 0;

 // set up probability tables
 assert(numStates == 4); // presently defined for 4 states
 double condProbOff[4] = {0.8, 0.4, 0.1, 0.1};
 double posteriorProb[numStates];
 for (int i = 0; i < numStates; i++)
 posteriorProb[i] = 0.0;
 double eventProb = 1.0 / pow(numStates, numSamples);

 // loop through all possibilities
 for (int done = 0; !done;){

 // compute importance weights (is probability distribution)
 double weight(numSamples), totalWeight = 0.0;
 for (int i = 0; i < numSamples; i++)
 totalWeight += weight[i] = condProbOff[samples[i]];
 // normalize them
 for (int i = 0; i < numSamples; i++)
 weight[i] /= totalWeight;

 // calculate contribution to posterior probability
 for (int i = 0; i < numSamples; i++)
 posteriorProb[samples[i]] += eventProb * weight[i];
 }
}

// increment counter
for (int i = 0; i < numSamples && i != -1;){
 samples[i]++;
 if (samples[i] >= numStates)
 samples[i+1] = 0;
 else
 i = -1;
 if (i == numSamples)
 done = 1;
}
}

// print result
cout << "Result: ";
for (int i = 0; i < numStates; i++)
 cout << " " << posteriorProb[i];
cout << endl;

// calculate asymptotic expectation
double totalWeight = 0.0;
for (int i = 0; i < numStates; i++)
 totalWeight += condProbOff[i];

cout << "Unbiased: ";
for (int i = 0; i < numStates; i++)
 cout << " " << condProbOff[i] / totalWeight;
cout << endl;

// calculate KL divergence
double kl = 0.0;
for (int i = 0; i < numStates; i++)
 kl += posteriorProb[i] * (log(posteriorProb[i]) -
 log(condProbOff[i] / totalWeight));
cout << "KL divergence: " << kl << endl;
}

```

(a)

(b)

**Figure S26.1** Code to calculate answer to exercise 25.1.

- a.** The program (correctly) calculates the following posterior distributions for the four states, as a function of the number of samples  $N$ . Note that for  $N = 1$ , the measurement is ignored entirely! The correct posterior for  $N = \infty$  is calculated using Bayes rule.

| $N$          | $p(\text{sample at } s_1)$ | $p(\text{sample at } s_2)$ | $p(\text{sample at } s_3)$ | $p(\text{sample at } s_4)$ |
|--------------|----------------------------|----------------------------|----------------------------|----------------------------|
| $N = 1$      | 0.25                       | 0.25                       | 0.25                       | 0.25                       |
| $N = 2$      | 0.368056                   | 0.304167                   | 0.163889                   | 0.163889                   |
| $N = 3$      | 0.430182                   | 0.314463                   | 0.127677                   | 0.127677                   |
| $N = 4$      | 0.466106                   | 0.314147                   | 0.109874                   | 0.109874                   |
| $N = 5$      | 0.488602                   | 0.311471                   | 0.0999636                  | 0.0999636                  |
| $N = 6$      | 0.503652                   | 0.308591                   | 0.0938788                  | 0.0938788                  |
| $N = 7$      | 0.514279                   | 0.306032                   | 0.0898447                  | 0.0898447                  |
| $N = 8$      | 0.522118                   | 0.303872                   | 0.0870047                  | 0.0870047                  |
| $N = 9$      | 0.528112                   | 0.30207                    | 0.0849091                  | 0.0849091                  |
| $N = 10$     | 0.532829                   | 0.300562                   | 0.0833042                  | 0.0833042                  |
| $N = \infty$ | 0.571429                   | 0.285714                   | 0.0714286                  | 0.0714286                  |

b. Plugging the posterior for  $N = \infty$  into the definition of the KL Divergence gives us:

| $N$     | $KL(\hat{p}, p)$ | $N$          | $KL(\hat{p}, p)$ |
|---------|------------------|--------------|------------------|
| $N = 1$ | 0.386329         | $N = 7$      | 0.00804982       |
| $N = 2$ | 0.129343         | $N = 8$      | 0.00593024       |
| $N = 3$ | 0.056319         | $N = 9$      | 0.00454205       |
| $N = 4$ | 0.029475         | $N = 10$     | 0.00358663       |
| $N = 5$ | 0.0175705        | $N = \infty$ | 0                |

c. The proof for  $N = 1$  is trivial, since the re-weighting ignores the measurement probability entirely. Therefore, the probability for generating a sample in any of the locations in  $S$  is given by the initial distribution, which is uniform.

For  $N = 2$ , a proof is easily obtained by considering all  $2^4 = 16$  ways in which initial samples are generated:

| number                   | samples | probability of sample set | $p(z s)$ for each sample      | weights for each sample     | probability of resampling for each location in $S$                  |
|--------------------------|---------|---------------------------|-------------------------------|-----------------------------|---------------------------------------------------------------------|
| 1                        | 0 0     | $\frac{1}{16}$            | $\frac{4}{5}$ $\frac{4}{5}$   | $\frac{1}{2}$ $\frac{1}{2}$ | $\frac{1}{16}$ 0 0 0                                                |
| 2                        | 0 1     | $\frac{1}{16}$            | $\frac{2}{5}$ $\frac{4}{5}$   | $\frac{3}{9}$ $\frac{6}{9}$ | $\frac{1}{24}$ $\frac{48}{48}$ 0 0                                  |
| 3                        | 0 2     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{4}{5}$  | $\frac{1}{9}$ $\frac{8}{9}$ | $\frac{1}{18}$ 0 $\frac{1}{144}$ 0                                  |
| 4                        | 0 3     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{4}{5}$  | $\frac{1}{9}$ $\frac{8}{9}$ | $\frac{1}{18}$ 0 0 $\frac{1}{144}$                                  |
| 5                        | 1 0     | $\frac{1}{16}$            | $\frac{4}{5}$ $\frac{2}{5}$   | $\frac{6}{9}$ $\frac{3}{9}$ | $\frac{1}{24}$ $\frac{48}{48}$ 0 0                                  |
| 6                        | 1 1     | $\frac{1}{16}$            | $\frac{2}{5}$ $\frac{2}{5}$   | $\frac{1}{2}$ $\frac{1}{2}$ | 0 $\frac{1}{16}$ 0 0                                                |
| 7                        | 1 2     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{2}{5}$  | $\frac{1}{5}$ $\frac{4}{5}$ | 0 $\frac{1}{20}$ $\frac{1}{80}$ 0                                   |
| 8                        | 1 3     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{2}{5}$  | $\frac{1}{5}$ $\frac{4}{5}$ | 0 $\frac{1}{20}$ 0 $\frac{1}{80}$                                   |
| 9                        | 2 0     | $\frac{1}{16}$            | $\frac{4}{5}$ $\frac{1}{10}$  | $\frac{8}{9}$ $\frac{1}{9}$ | $\frac{1}{18}$ 0 $\frac{1}{144}$ 0                                  |
| 10                       | 2 1     | $\frac{1}{16}$            | $\frac{2}{5}$ $\frac{1}{10}$  | $\frac{4}{5}$ $\frac{1}{5}$ | 0 $\frac{1}{20}$ $\frac{1}{80}$ 0                                   |
| 11                       | 2 2     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{1}{10}$ | $\frac{1}{2}$ $\frac{1}{2}$ | 0 0 $\frac{1}{16}$ 0                                                |
| 12                       | 2 3     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{1}{10}$ | $\frac{1}{2}$ $\frac{1}{2}$ | 0 0 0 $\frac{1}{32}$                                                |
| 13                       | 3 0     | $\frac{1}{16}$            | $\frac{4}{5}$ $\frac{1}{10}$  | $\frac{8}{9}$ $\frac{1}{9}$ | $\frac{1}{18}$ 0 0 $\frac{1}{144}$                                  |
| 14                       | 3 1     | $\frac{1}{16}$            | $\frac{2}{5}$ $\frac{1}{10}$  | $\frac{4}{5}$ $\frac{1}{5}$ | 0 $\frac{1}{20}$ 0 $\frac{1}{80}$                                   |
| 15                       | 3 2     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{1}{10}$ | $\frac{1}{2}$ $\frac{1}{2}$ | 0 0 0 $\frac{1}{32}$                                                |
| 16                       | 3 3     | $\frac{1}{16}$            | $\frac{1}{10}$ $\frac{1}{10}$ | $\frac{1}{2}$ $\frac{1}{2}$ | 0 0 0 $\frac{1}{16}$                                                |
| sum of all probabilities |         |                           |                               |                             | $\frac{53}{144}$ $\frac{73}{240}$ $\frac{59}{360}$ $\frac{59}{360}$ |

A quick check should convince you that these numbers are the same as above. Placing this into the definition of the KL divergence with the correct posterior distribution, gives us 0.129343.

For  $N = \infty$  we know that the sampler is unbiased. Hence, the probability of generating a sample is the same as the posterior distribution calculated by Bayes filters. Those are given above as well.

- d. Here are two possible modifications. First, if the initial robot location is known with absolute certainty, the sampler above will always be unbiased. Second, if the sensor measurement  $z$  is equally likely for all states, that is  $p(z|s_1) = p(z|s_2) = p(z|s_3) = p(z|s_4)$ , it will also be unbiased. An *invalid* answer, which we frequently encountered in class, pertains to the algorithm (instead of the problem formulation). For example, replacing particle filters by the exact discrete Bayes filer remedies the problem but is not a legitimate answer to this question. Neither is the use of infinitely many particles.

### Exercise 26.4.#MCLI

Implement Monte Carlo localization for a simulated robot with range sensors. A grid map and range data are available from the code repository at [aima.cs.berkeley.edu](https://aima.cs.berkeley.edu). You should demonstrate successful global localization of the robot.

Implementing Monte Carlo localization requires a lot of work but is a premiere way to gain insights into the basic workings of probabilistic algorithms in robotics, and the intricacies inherent in real data. We have used this exercise in many courses, and students consistently expressed having learned a lot. We strongly recommend this exercise!

The implementation is not as straightforward as it may appear at first glance. Common problems include:

- The sensor model models too little noise, or the wrong type of noise. For example, a simple Gaussian will not work here.
- The motion model assumes too little or too much noise, or the wrong type of noise. Here a Gaussian will work fine though.
- The implementation may introduce unnecessarily high variance in the resulting sampling set, by sampling too often, or by sampling in the wrong way. This problem manifests itself by diversity disappearing prematurely, often with the wrong samples surviving. While the basic MCL algorithm, as stated in the book, suggests that sampling should occur after each motion update, implementations that sample less frequently tend to yield superior results. Further, drawing samples independently of each other is inferior to so-called low variance samplers. Here is a version of low variance sampling, in which  $\mathcal{X}$  denotes the particles and  $W$  their importance weights. The resulting resampled particles reside in the set  $S'$ .

```
function LOW-VARIANCE-WEIGHTED-SAMPLE-WITH-REPLACEMENT(S, W):
 $S' = \{\}$
 $b = \sum_{i=1}^N W[i]$
 $r = \text{rand}(0; b)$
 for $n = 1$ to N do
 $i = \text{argmin}_j \sum_{m=1}^j W[m] \geq r$
 add $S[i]$ to S'
 $r = (r + \text{rand}(0; c)) \text{ modulo } b$
```

**return**  $S'$

The parameter  $c$  determines the speed at which we cycle through the sample set. While each sample's probability remains the same as if it were sampled independently, the resulting samples are dependent, and the variance of the sample set  $S'$  is lower (assuming  $c < b$ ). As a pleasant side effect, the low-variance samples is also easily implemented in  $O(N)$  time, which is more difficult for the independent sampler.

- Samples are started in the occupied or unknown parts of the map, or are allowed into those parts during the forward sampling (motion prediction) step of the MCL algorithm.
- Too few samples are used. A few thousand should do the job, a few hundred will probably not.

The algorithm can be sped up by pre-caching all noise-free measurements, for all  $x$ - $y$ - $\theta$  poses that the robot might assume. For that, it is convenient to define a grid over the space of all poses, with 10 centimeters spatial and 2 degrees angular resolution. One might then compute the noise-free measurements for the centers of those grid cells. The sensor model is clearly just a function of those correct measurements; and computing those takes the bulk of time in MCL.

## 26.5 Planning and Control

### Exercise 26.5.#ABMA

Consider a robot with two simple manipulators, as shown in figure ???. Manipulator A is a square block of side 2 which can slide back and on a rod that runs along the x-axis from  $x = -10$  to  $x = 10$ . Manipulator B is a square block of side 2 which can slide back and on a rod that runs along the y-axis from  $y = -10$  to  $y = 10$ . The rods lie outside the plane of manipulation, so the rods do not interfere with the movement of the blocks. A configuration is then a pair  $\langle x, y \rangle$  where  $x$  is the x-coordinate of the center of manipulator A and where  $y$  is the y-coordinate of the center of manipulator B. Draw the configuration space for this robot, indicating the permitted and excluded zones.

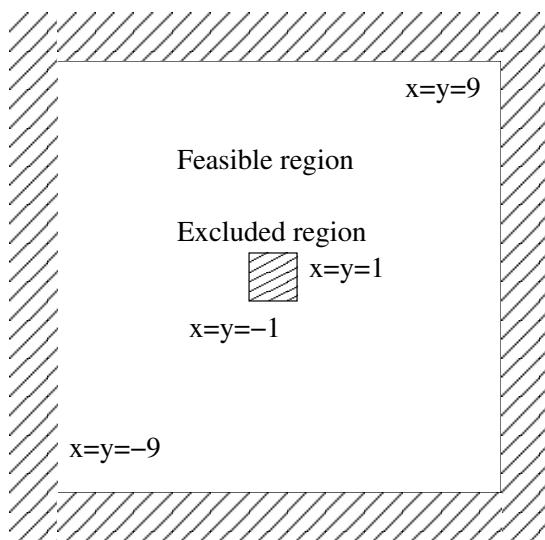
See Figure S26.2.

### Exercise 26.5.#ABMB

Suppose that you are working with the robot in Exercise 26.ABMA above and you are given the problem of finding a path from the starting configuration of figure ?? to the ending configuration. Consider a potential function

$$D(A, Goal)^2 + D(B, Goal)^2 + \frac{1}{D(A, B)^2}$$

where  $D(A, B)$  is the distance between the closest points of A and B.



**Figure S26.2** Robot configuration.

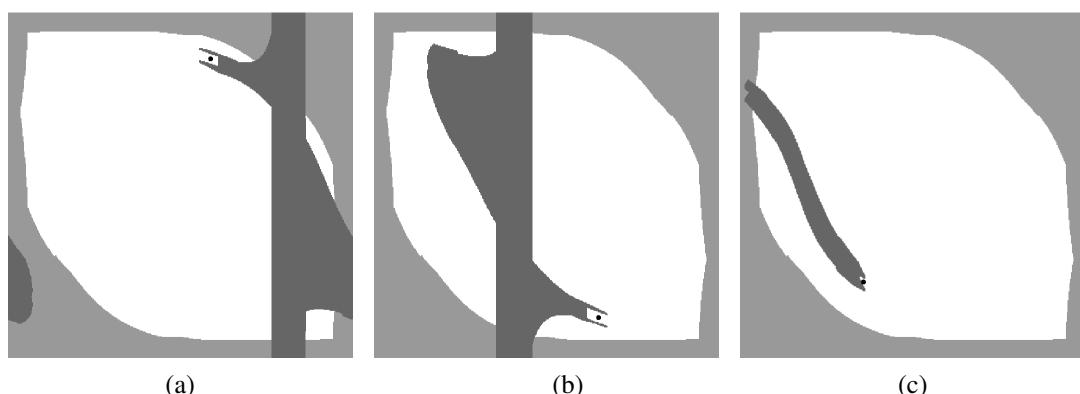
- a. Show that hill climbing in this potential field will get stuck in a local minimum.
  - b. Describe a potential field where hill climbing will solve this particular problem. You need not work out the exact numerical coefficients needed, just the general form of the solution. (Hint: Add a term that “rewards” the hill climber for moving A out of B’s way, even in a case like this where this does not reduce the distance from A to B in the above sense.)
- 
- A. Hill climbing down the potential moves manipulator B down the rod to the point where the derivative of the term “square of distance from current position of B to goal position” is exactly the negative of the derivative of the term “ $1/\text{square of distance from A to B}$ ”. This is a local minimum of the potential function, because it is a minimum of the sum of those two terms, with A held fixed, and small movements of A do not change the value of the term “ $1/\text{square of distance from A to B}$ ”, and only increase the value of the term “square of distance from current position of A to goal position”
  - B. Add a term of the form “ $1/\text{square of distance between the center of A and the center of B}$ .” Now the stopping configuration of part A is no longer a local minimum because moving A to the left decreases this term. (Moving A to the left does also increase the value of the term “square of distance from current position of A to goal position”, but that term is at a local minimum, so its derivative is zero, so the gain outweighs the loss, at least for a while.) For the right combination of linear coefficient, hill climbing will find its way to a correct solution.

**Exercise 26.5.#WSCS**

This exercise explores the relationship between workspace and configuration space using the examples shown in Figure ??.

- a. Consider the robot configurations shown in Figure ??(a) through (c), ignoring the obstacle shown in each of the diagrams. Draw the corresponding arm configurations in configuration space. (*Hint:* Each arm configuration maps to a single point in configuration space, as illustrated in Figure 26.12(b).)
- b. Draw the configuration space for each of the workspace diagrams in Figure ??(a)–(c). (*Hint:* The configuration spaces share with the one shown in Figure ??(a) the region that corresponds to self-collision, but differences arise from the lack of enclosing obstacles and the different locations of the obstacles in these individual figures.)
- c. For each of the black dots in Figure ??(e)–(f), draw the corresponding configurations of the robot arm in workspace. Please ignore the shaded regions in this exercise.
- d. The configuration spaces shown in Figure ??(e)–(f) have all been generated by a single workspace obstacle (dark shading), plus the constraints arising from the self-collision constraint (light shading). Draw, for each diagram, the workspace obstacle that corresponds to the darkly shaded area.
- e. Figure ??(d) illustrates that a single planar obstacle can decompose the workspace into two disconnected regions. What is the maximum number of disconnected regions that can be created by inserting a planar obstacle into an obstacle-free, connected workspace, for a 2DOF robot? Give an example, and argue why no larger number of disconnected regions can be created. How about a non-planar obstacle?

- a. The configurations of the robots are shown by the black dots in Figure S26.3.

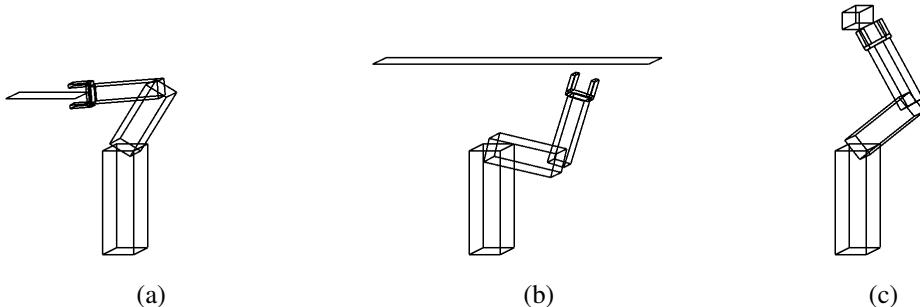


**Figure S26.3** Configuration of the robots.

- b. Figure S26.3 also answers the second part of this exercise: it shows the configuration space of the robot arm constrained by the self-collision constraint and the constraint

imposed by the obstacle.

- c. The three workspace obstacles are shown in Figure S26.4.



**Figure S26.4** Workspace obstacles.

- d. This question is a great mind teaser that illustrates the difficulty of robot motion planning! Unfortunately, for an arbitrary robot, a planar obstacle can decompose the workspace into *any* number of disconnected subspaces. To see, imagine a 1-DOF rigid robot that moves on a horizontal rod, and possesses  $N$  upward-pointing fingers, like a giant fork. A single planar obstacle protruding vertically into one of the free-spaces between the fingers could effectively separate the configuration space into  $N + 1$  disjoint subspaces. A second DOF will not change this.

More interesting is the robot arm used as an example throughout this book. By slightly extending the vertical obstacles protruding into the robot's workspace we can decompose the configuration space into five disjoint regions. The following figures show the configuration space along with representative configurations for each of the five regions.

Is five the maximum for any planar object that protrudes into the workspace of this particular robot arm? We honestly do not know; but we offer a \$1 reward for the first person who presents to us a solution that decomposes the configuration space into six, seven, eight, nine, or ten disjoint regions. For the reward to be claimed, all these regions must be clearly disjoint, and they must be a two-dimensional manifold in the robot's configuration space.

For non-planar objects, the configuration space is easily decomposed into any number of regions. A circular object may force the elbow to be just about maximally bent; the resulting workspace would then be a very narrow pipe that leave the shoulder largely unconstrained, but confines the elbow to a narrow range. This pipe is then easily chopped into pieces by small dents in the circular object; the number of such dents can be increased without bounds.

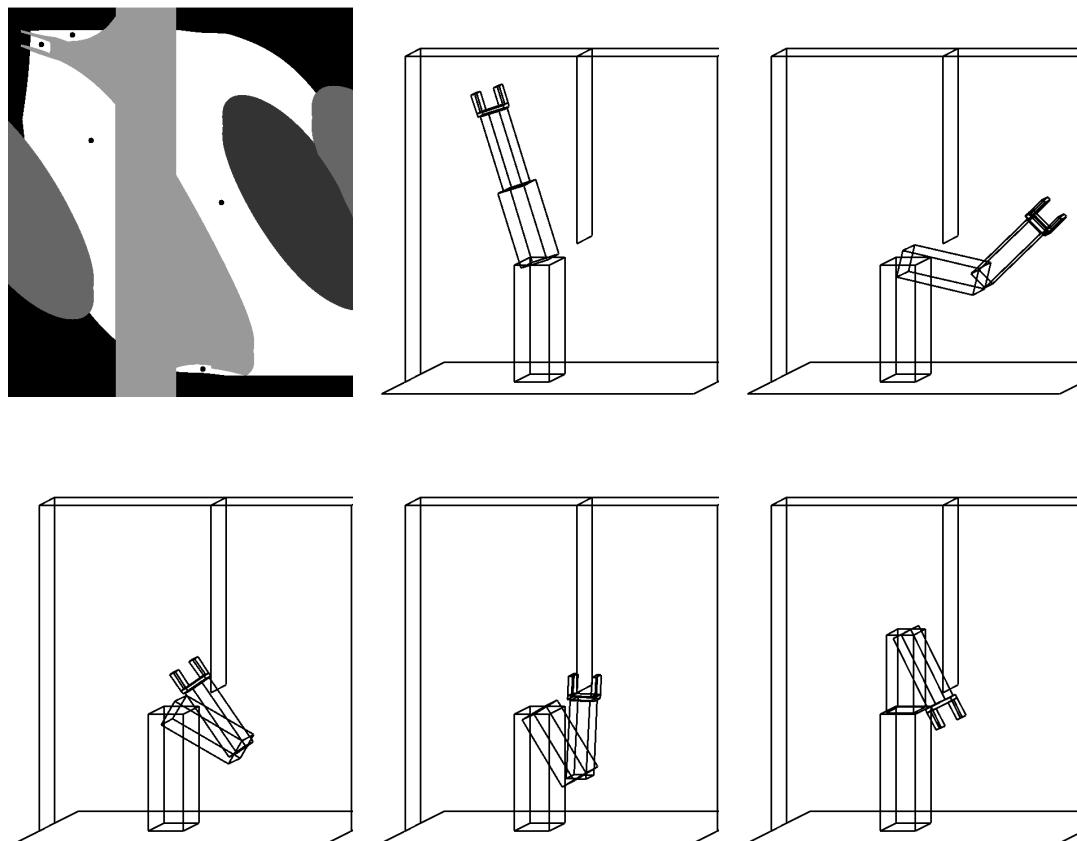
**Exercise 26.5.#INVK**

Consider the robot arm shown in Figure 26.12. Assume that the robot's base element is 70cm long and that its upper arm and forearm are each 50cm long. As argued on page ??, the inverse kinematics of a robot is often not unique. State an explicit closed-form solution of the inverse kinematics for this arm. Under what exact conditions is the solution unique?

Let  $\alpha$  be the shoulder and  $\beta$  be the elbow angle. The coordinates of the end effector are then given by the following expression. Here  $z$  is the height and  $x$  the horizontal displacement between the end effector and the robot's base (origin of the coordinate system):

$$\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} 0\text{cm} \\ 70\text{cm} \end{pmatrix} + \begin{pmatrix} \sin \alpha \\ \cos \alpha \end{pmatrix} \cdot 50\text{cm} + \begin{pmatrix} \sin(\alpha + \beta) \\ \cos(\alpha + \beta) \end{pmatrix} \cdot 50\text{cm}$$

Notice that this is only one way to define the kinematics. The zero-positions of the angles  $\alpha$  and  $\beta$  can be anywhere, and the motors may turn clockwise or counterclockwise. Here we chose define these angles in a way that the arm points straight up at  $\alpha = \beta = 0$ ; furthermore,



**Figure S26.5** Configuration space for each of the five regions.

increasing  $\alpha$  and  $\beta$  makes the corresponding joint rotate counterclockwise.

*Inverse kinematics* is the problem of computing  $\alpha$  and  $\beta$  from the end effector coordinates  $x$  and  $z$ . For that, we observe that the elbow angle  $\beta$  is uniquely determined by the Euclidean distance between the shoulder joint and the end effector. Let us call this distance  $d$ . The shoulder joint is located 70cm above the origin of the coordinate system; hence, the distance  $d$  is given by  $d = \sqrt{x^2 + (z - 70\text{cm})^2}$ . An alternative way to calculate  $d$  is by recovering it from the elbow angle  $\beta$  and the two connected joints (each of which is 50cm long):  $d = 2 \cdot 50\text{cm} \cdot \cos \frac{\beta}{2}$ . The reader can easily derive this from basic trigonometry, exploiting the fact that both the elbow and the shoulder are of equal length. Equating these two different derivations of  $d$  with each other gives us

$$\sqrt{x^2 + (z - 70\text{cm})^2} = 100\text{cm} \cdot \cos \frac{\beta}{2} \quad (26.1)$$

or

$$\beta = \pm 2 \cdot \arccos \frac{\sqrt{x^2 + (z - 70\text{cm})^2}}{100\text{cm}} \quad (26.2)$$

In most cases,  $\beta$  can assume two symmetric configurations, one pointing down and one pointing up. We will discuss exceptions below.

To recover the angle  $\alpha$ , we note that the angle between the shoulder (the base) and the end effector is given by  $\arctan 2(x, z - 70\text{cm})$ . Here  $\arctan 2$  is the common generalization of the arcus tangens to all four quadrants (check it out—it is a function in C). The angle  $\alpha$  is now obtained by adding  $\frac{\beta}{2}$ , again exploiting that the shoulder and the elbow are of equal length:

$$\alpha = \arctan 2(x, z - 70\text{cm}) - \frac{\beta}{2} \quad (26.3)$$

Of course, the actual value of  $\alpha$  depends on the actual choice of the value of  $\beta$ . With the exception of singularities,  $\beta$  can take on exactly two values.

The inverse kinematics is *unique* if  $\beta$  assumes a single value; as a consequence, so does alpha. For this to be the case, we need that

$$\arccos \frac{\sqrt{x^2 + (z - 70\text{cm})^2}}{100\text{cm}} = 0 \quad (26.4)$$

This is the case exactly when the argument of the arccos is 1, that is, when the distance  $d = 100\text{cm}$  and the arm is fully stretched. The end points  $x, z$  then lie on a circle defined by  $\sqrt{x^2 + (z - 70\text{cm})^2} = 100\text{cm}$ . If the distance  $d > 100\text{cm}$ , there is no solution to the inverse kinematic problem: the point is simply too far away to be reachable by the robot arm.

Unfortunately, configurations like these are numerically unstable, as the quotient may be slightly larger than one (due to truncation errors). Such points are commonly called *singularities*, and they can cause major problems for robot motion planning algorithms. A second singularity occurs when the robot is “folded up,” that is,  $\beta = 180^\circ$ . Here the end effector’s position is identical with that of the robot elbow, regardless of the angle  $\alpha$ :  $x = 0\text{cm}$  and  $z = 70\text{cm}$ . This is an important singularity, as there are *infinitely* many solutions to the inverse kinematics. As long as  $\beta = 180^\circ$ , the value of  $\alpha$  can be arbitrary. Thus, this simple robot arm gives us an example where the inverse kinematics can yield zero, one, two, or infinitely many solutions.

**Exercise 26.5.#VORO**

Implement an algorithm for calculating the Voronoi diagram of an arbitrary 2D environment, described by an  $n \times n$  Boolean array. Illustrate your algorithm by plotting the Voronoi diagram for 10 interesting maps. What is the complexity of your algorithm?

Code not shown.

## 26.6 Planning Uncertain Movements

**Exercise 26.6.#MOBH**

Consider a mobile robot moving on a horizontal surface. Suppose that the robot can execute two kinds of motions:

- Rolling forward a specified distance.
- Rotating in place through a specified angle.

The state of such a robot can be characterized in terms of three parameters  $\langle x, y, \phi \rangle$ , the x-coordinate and y-coordinate of the robot (more precisely, of its center of rotation) and the robot's orientation expressed as the angle from the positive x direction. The action "*Roll(D)*" has the effect of changing state  $\langle x, y, \phi \rangle$  to  $\langle x + D \cos(\phi), y + D \sin(\phi), \phi \rangle$ , and the action *Rotate( $\theta$ )* has the effect of changing state  $\langle x, y, \phi \rangle$  to  $\langle x, y, \phi + \theta \rangle$ .

- Suppose that the robot is initially at  $\langle 0, 0, 0 \rangle$  and then executes the actions *Rotate(60°)*, *Roll(1)*, *Rotate(25°)*, *Roll(2)*. What is the final state of the robot?
- Now suppose that the robot has imperfect control of its own rotation, and that, if it attempts to rotate by  $\theta$ , it may actually rotate by any angle between  $\theta - 10^\circ$  and  $\theta + 10^\circ$ . In that case, if the robot attempts to carry out the sequence of actions in (A), there is a range of possible ending states. What are the minimal and maximal values of the x-coordinate, the y-coordinate and the orientation in the final state?
- Let us modify the model in (B) to a probabilistic model in which, when the robot attempts to rotate by  $\theta$ , its actual angle of rotation follows a Gaussian distribution with mean  $\theta$  and standard deviation  $10^\circ$ . Suppose that the robot executes the actions *Rotate(90°)*, *Roll(1)*. Give a simple argument that (a) the expected value of the location at the end is not equal to the result of rotating exactly  $90^\circ$  and then rolling forward 1 unit, and (b) that the distribution of locations at the end does not follow a Gaussian. (Do not attempt to calculate the true mean or the true distribution.)

The point of this exercise is that rotational uncertainty quickly gives rise to a lot of positional uncertainty and that dealing with rotational uncertainty is painful, whether uncertainty is treated in terms of hard intervals or probabilistically, due to the fact that the relation between orientation and position is both non-linear and non-monotonic.

A.  $x = 1 \cdot \cos(60^\circ) + 2 \cdot \cos(85^\circ) =$   
 $y = 1 \cdot \sin(60^\circ) + 2 \cdot \sin(85^\circ) =$

$$\phi = 90^\circ$$

- B. The minimal value of  $x$  is  $1 \cdot \cos(70^\circ) + 2 \cdot \cos(105^\circ) = -0.176$  achieved when the first rotation is actually  $70^\circ$  and the second is actually  $35^\circ$ . The maximal value of  $x$  is  $1 \cdot \cos(50^\circ) + 2 \cdot \cos(65^\circ) = 1.488$  achieved when the first rotation is actually  $50^\circ$  and the second is actually  $15^\circ$ . The minimal value of  $y$  is  $1 \cdot \sin(50^\circ) + 2 \cdot \sin(65^\circ) = 2.579$  achieved when the first rotation is actually  $50^\circ$  and the second is actually  $15^\circ$ . The maximal value of  $y$  is  $1 \cdot \sin(70^\circ) + 2 \cdot \sin(90^\circ) = 2.94$  achieved when the first rotation is actually  $70^\circ$  and the second is actually  $20^\circ$ . The minimal value of  $\phi$  is  $65^\circ$  achieved when the first rotation is actually  $50^\circ$  and the second is actually  $15^\circ$ . The maximal value of  $\phi$  is  $105^\circ$  achieved when the first rotation is actually  $70^\circ$  and the second is actually  $35^\circ$ .
- C. The maximal possible  $y$ -coordinate (1.0) is achieved when the rotation is executed at exactly  $90^\circ$ . Since it is the maximal possible value, it cannot be the mean value. Since there is a maximal possible value, the distribution cannot be a Gaussian, which has non-zero (though small) probabilities for all values.

[\[\[need exercises\]\]](#)

## 26.7 Reinforcement Learning in Robotics

---

[\[\[need exercises\]\]](#)

## 26.8 Humans and Robots

---

[\[\[need exercises\]\]](#)

## 26.9 Alternative Robotic Frameworks

---

### Exercise 26.9.#ROBE

Consider the simplified robot shown in Figure ???. Suppose the robot's Cartesian coordinates are known at all times, as are those of its goal location. However, the locations of the obstacles are unknown. The robot can sense obstacles in its immediate proximity, as illustrated in this figure. For simplicity, let us assume the robot's motion is noise-free, and the state space is discrete. Figure ?? is only one example; in this exercise you are required to address all possible grid worlds with a valid path from the start to the goal location.

- a. Design a deliberate controller that guarantees that the robot always reaches its goal location if at all possible. The deliberate controller can memorize measurements in the form of a map that is being acquired as the robot moves. Between individual moves, it may spend arbitrary time deliberating.

- b. Now design a *reactive* controller for the same task. This controller may not memorize past sensor measurements. (It may not build a map!) Instead, it has to make all decisions based on the current measurement, which includes knowledge of its own location and that of the goal. The time to make a decision must be independent of the environment size or the number of past time steps. What is the maximum number of steps that it may take for your robot to arrive at the goal?
- c. How will your controllers from (a) and (b) perform if any of the following six conditions apply: continuous state space, noise in perception, noise in motion, noise in both perception and motion, unknown location of the goal (the goal can be detected only when within sensor range), or moving obstacles. For each condition and each controller, give an example of a situation where the robot fails (or explain why it cannot fail).

A simple deliberate controller might work as follows: Initialize the robot's map with an empty map, in which all states are assumed to be navigable, or free. Then iterate the following loop: Find the shortest path from the current position to the goal position in the map using A\*; execute the first step of this path; sense; and modify the map in accordance with the sensed obstacles. If the robot reaches the goal, declare success. The robot declares failure when A\* fails to find a path to the goal. It is easy to see that this approach is both complete and correct. The robot always finds a path to a goal if one exists. If no such path exists, the approach detects this through failure of the path planner. When it declares failure, it is indeed correct in that no path exists.

A common reactive algorithm, which has the same correctness and completeness property as the deliberate approach, is known as the BUG algorithm. The BUG algorithm distinguishes two modes, the boundary-following and the go-to-goal mode. The robot starts in go-to-goal mode. In this mode, the robot always advances to the adjacent grid cell closest to the goal. If this is impossible because the cell is blocked by an obstacle, the robot switches to the boundary-following mode. In this mode, the robot follows the boundary of the obstacle until it reaches a point on the boundary that is a local minimum to the straight-line distance to the goal. If such a point is reached, the robot returns to the go-to-goal mode. If the robot reaches the goal, it declares success. It declares failure when the same point is reached twice, which can only occur in the boundary-following mode. It is easy to see that the BUG algorithm is correct and complete. If a path to the goal exists, the robot will find it. When the robot declares failure, no path to the goal may exist. If no such path exists, the robot will ultimately reach the same location twice and detect its failure.

Both algorithms can cope with continuous state spaces provided that they can accurately perceive obstacles, plan paths around them (deliberative algorithm) or follow their boundary (reactive algorithm). Noise in motion can cause failures for both algorithms, especially if the robot has to move through a narrow opening to reach the goal. Similarly, noise in perception destroys both completeness and correctness: In both cases the robot may erroneously conclude a goal cannot be reached, just because its perception was noise. However, a deliberate algorithm might build a probabilistic map, accommodating the uncertainty that arises from the noisy sensors. Neither algorithm as stated can cope with unknown goal locations; however, the deliberate algorithm is easily converted into an *exploration* algorithm by which the

robot always moves to the nearest unexplored location. Such an algorithm would be complete and correct (in the noise-free case). In particular, it would be guaranteed to find and reach the goal when reachable. The BUG algorithm, however, would not be applicable. A common reactive technique for finding a goal whose location is unknown is random motion; this algorithm will with probability one find a goal if it is reachable; however, it is unable to determine when to give up, and it may be highly inefficient. Moving obstacles will cause problems for both the deliberate and the reactive approach; in fact, it is easy to design an adversarial case where the obstacle always moves into the robot's way. For slow-moving obstacles, a common deliberate technique is to attach a timer to obstacles in the grid, and erase them after a certain number of time steps. Such an approach often has a good chance of succeeding.

### Exercise 26.9.#SUBA

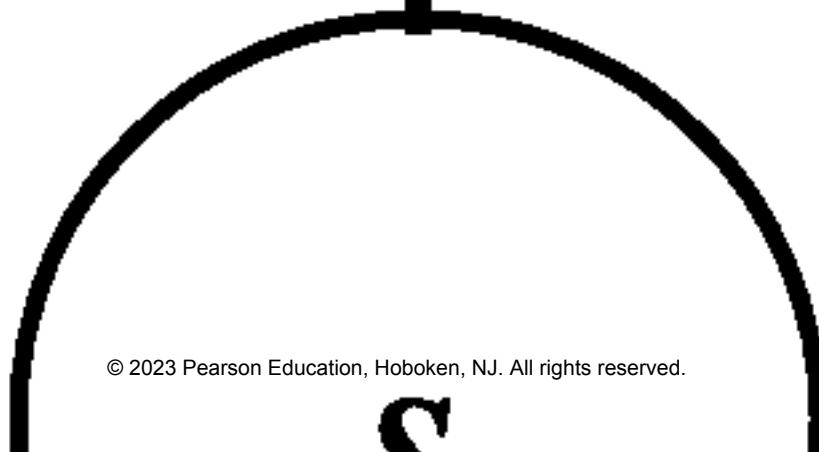
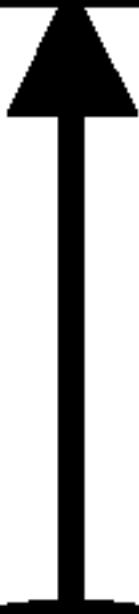
In Figure 26.32(b) on page 969, we encountered an augmented finite state machine for the control of a single leg of a hexapod robot. In this exercise, the aim is to design an AFSM that, when combined with six copies of the individual leg controllers, results in efficient, stable locomotion. For this purpose, you have to augment the individual leg controller to pass messages to your new AFSM and to wait until other messages arrive. Argue why your controller is efficient, in that it does not unnecessarily waste energy (e.g., by sliding legs), and in that it propels the robot at reasonably high speeds. Prove that your controller satisfies the dynamic stability condition given on page ??.

There are a number of ways to extend the single-leg AFSM in Figure 25.22(b) into a set of AFSMs for controlling a hexapod. A straightforward extension—though not necessarily the most efficient one—is shown in the following diagram. Here the set of legs is divided into two, named A and B, and legs are assigned to these sets in alternating sequence. The top level controller, shown on the left, goes through six stages. Each stage lifts a set of legs, pushes the ones still on the ground backwards, and then lowers the legs that have previously been lifted. The same sequence is then repeated for the other set of legs. The corresponding single-leg controller is essentially the same as in Figure 25.22(b), but with added wait-steps for synchronization with the coordinating AFSM. The low-level AFSM is replicated six times, once for each leg.

For showing that this controller is stable, we show that at least one leg group is on the ground at all times. If this condition is fulfilled, the robot's center of gravity will always be above the imaginary triangle defined by the three legs on the ground. The condition is easily proven by analyzing the top level AFSM. When one group of legs is  $s_4$  (or on the way to  $s_4$  from  $s_3$ ), the other is either in  $s_2$  or  $s_1$ , both of which are on the ground. However, this proof only establishes that the robot does not fall over when on flat ground; it makes no assertions about the robot's performance on non-flat terrain. Our result is also restricted to *static stability*, that is, it ignores all dynamic effects such as inertia. For a fast-moving hexapod, asking that its center of gravity be enclosed in the triangle of support may be insufficient.

Wait  
until U  
received

F



### Exercise 26.9.#HUMR

(This exercise was first devised by Michael Genesereth and Nils Nilsson. It works for first graders through graduate students.) Humans are so adept at basic household tasks that they often forget how complex these tasks are. In this exercise you will discover the complexity and recapitulate the last 30 years of developments in robotics. Consider the task of building an arch out of three blocks. Simulate a robot with four humans as follows:

**Brain.** The Brain direct the hands in the execution of a plan to achieve the goal. The Brain receives input from the Eyes, but *cannot see the scene directly*. The brain is the only one who knows what the goal is.

**Eyes.** The Eyes report a brief description of the scene to the Brain: “There is a red box standing on top of a green box, which is on its side” Eyes can also answer questions from the Brain such as, “Is there a gap between the Left Hand and the red box?” If you have a video camera, point it at the scene and allow the eyes to look at the viewfinder of the video camera, but not directly at the scene.

**Left hand and right hand.** One person plays each Hand. The two Hands stand next to each other, each wearing an oven mitt on one hand, Hands execute only simple commands from the Brain—for example, “Left Hand, move two inches forward.” They cannot execute commands other than motions; for example, they cannot be commanded to “Pick up the box.” The Hands must be *blindfolded*. The only sensory capability they have is the ability to tell when their path is blocked by an immovable obstacle such as a table or the other Hand. In such cases, they can beep to inform the Brain of the difficulty.

We have used this exercise in class to great effect. The students get a clearer picture of why it is hard to do robotics. The only drawback is that it is a lot of fun to play, and thus the students want to spend a lot of time on it, and the ones who are just observing feel like they are missing out. If you have laboratory or TA sections, you can do the exercise there.

Bear in mind that being the Brain is a very stressful job. It can take an hour just to stack three boxes. Choose someone who is not likely to panic or be crushed by student derision. Help the Brain out by suggesting useful strategies such as defining a mutually agreed Hand-centric coordinate system so that commands are unambiguous. Almost certainly, the Brain will start by issuing absolute commands such as “Move the Left Hand 12 inches positive y direction” or “Move the Left Hand to (24,36).” Such actions will never work. The most useful “invention” that students will suggest is the guarded motion discussed in Section 25.5—that is, macro-operators such as “Move the Left Hand in the positive y direction until the eyes say the red and green boxes are level.” This gets the Brain out of the loop, so to speak, and speeds things up enormously.

We have also used a related exercise to show why robotics in particular and algorithm design in general is difficult. The instructor uses as props a doll, a table, a diaper and some safety pins, and asks the class to come up with an algorithm for putting the diaper on the baby. The instructor then follows the algorithm, but interpreting it in the least cooperative way possible: putting the diaper on the doll’s head unless told otherwise, dropping the doll on the floor if possible, and so on.

[[need exercises]]

## 26.10 Application Domains

---

[[need exercises]]

# EXERCISES 27

## PHILOSOPHY, ETHICS, AND SAFETY OF AI

### 27.1 The Limits of AI

---

#### Exercise 27.1.#DISA

Go through Turing's list of alleged "disabilities" of machines, identifying which have been achieved, which are achievable in principle by a program, and which are still problematic because they require conscious mental states.

We will take the disabilities (see page 949) one at a time. Note that this exercise might be better as a class discussion rather than written work.

- a. *be kind*: Certainly there are programs that are polite and helpful, but to be kind requires an intentional state, so this one is problematic.
- b. *resourceful*: Resourceful means "clever at finding ways of doing things." Many programs meet this criteria to some degree: a compiler can be clever making an optimization that the programmer might not ever have thought of; a database program might cleverly create an index to make retrievals faster; a checkers or backgammon program learns to play as well as any human. One could argue whether the machines are "really" clever or just seem to be, but most people would agree this requirement has been achieved.
- c. *beautiful*: Its not clear if Turing meant to be beautiful or to create beauty, nor is it clear whether he meant physical or inner beauty. Certainly the many industrial artifacts in the New York Museum of Modern Art, for example, are evidence that a machine can be beautiful. There are also programs that have created art. One of the best known of these is chronicled in *Aaron's code: Meta-art, artificial intelligence, and the work of Harold Cohen* (McCorduck, 1991). There have been many instances of computers making art since, such as with the music of David Cope (Cope, 2008) or the images of Google's Deep Dream (Mordvinstev, 2015).
- d. *friendly* This appears to fall under the same category as *kind*.
- e. *have initiative* Interestingly, there is now a serious debate whether software should take initiative. The whole field of software agents says that it should; critics such as Ben Schneiderman say that to achieve predictability, software should only be an assistant, not an autonomous agent. Notice that the debate over whether software *should* have initiative presupposes that it *has* initiative.

- f. *have a sense of humor* We know of no major effort to produce humorous works. However, this seems to be achievable in principle. All it would take is someone like Harold Cohen who is willing to spend a long time tuning a humor-producing machine. We note that humorous text is probably easier to produce than other media. For a recent treatment of humor, its mechanisms, and biological relevancy, see Hurley, 2011.
- g. *tell right from wrong* There is considerable research in applying AI to legal reasoning, and there are now tools that assist the lawyer in deciding a case and doing research. One could argue whether following legal precedents is the same as telling right from wrong, and in any case this has a problematic conscious aspect to it.
- h. *make mistakes* At this stage, every computer user is familiar with software that makes mistakes! It is interesting to think back to what the world was like in Turing's day, when some people thought it would be difficult or impossible for a machine to make mistakes.
- i. *fall in love* This is one of the cases that clearly requires consciousness. Note that while some people claim that their pets love them, and some claim that pets are not conscious, we don't know of anybody who makes both claims.
- j. *enjoy strawberries and cream* There are two parts to this. First, there has been little to no work on taste perception in AI, so we're nowhere near a breakthrough on this. Second, the "enjoy" part clearly requires consciousness.
- k. *make someone fall in love with it* This criteria is actually not too hard to achieve; machines such as dolls and teddy bears have been doing it to children for centuries. Machines that talk and have more sophisticated behaviors just have a larger advantage in achieving this.
- l. *learn from experience* Part VI shows that this has been achieved many times in AI.
- m. *use words properly* No program uses words perfectly, but there have been many natural language programs that use words properly and effectively within a limited domain (see Chapters 22-23).
- n. *be the subject of its own thought* The problematic word here is "thought." Many programs can process themselves, as when a compiler compiles itself. Perhaps closer to human self-examination is the case where a program has an imperfect representation of itself. One anecdote of this involves Doug Lenat's Eurisko program. It used to run for long periods of time, and periodically needed to gather information from outside sources. It "knew" that if a person were available, it could type out a question at the console, and wait for a reply. Late one night it saw that no person was logged on, so it couldn't ask the question it needed to know. But it knew that Eurisko itself was up and running, and decided it would modify the representation of Eurisko so that it inherits from "Person," and then proceeded to ask itself the question!
- o. *have as much diversity of behavior as man* Clearly, no machine has achieved this, although there is no principled reason why one could not.
- p. *do something really new* This seems to be just an extension of the idea of learning from experience: if you learn enough, you can do something really new. "Really" is subjective, and some would say that no machine has achieved this yet. On the other hand, professional Go players believe that AlphaZero has revolutionized the theory of the game of Go.

**Exercise 27.1.#POPM**

Find and analyze an account in the popular media of one or more of the arguments to the effect that AI is impossible.

This exercise depends on what happens to have been published lately. Here is an excerpt from a review of Roger Penrose's book *Shadows of the Mind*, by Adam Schulman (1995):

Roger Penrose, the distinguished mathematical physicist, has again entered the lists to rid the world of a terrible dragon. The name of this dragon is "strong artificial intelligence."

Strong AI, as its defenders call it, is both a widely held scientific thesis and an ongoing technological program. The thesis holds that the human mind is nothing but a fancy calculating machine—"a computer made of meat"—and that all thinking is merely computation; the program is to build faster and more powerful computers that will eventually be able to do everything the human mind can do and more. Penrose believes that the thesis is false and the program unrealizable, and he is confident that he can prove these assertions. . . .

In Part I of *Shadows of the Mind* Penrose makes his rigorous case that human consciousness cannot be fully understood in computational terms. . . . How does Penrose prove that there is more to consciousness than mere computation? Most people will already find it inherently implausible that the diverse faculties of human consciousness—self-awareness, understanding, willing, imagining, feeling—differ only in complexity from the workings of, say, an IBM PC.

Students should have no problem finding things in this and other articles with which to disagree.

Dubious claims also emerge from the interaction between journalists' desire to write entertaining and controversial articles and academics' desire to achieve prominence and to be viewed as ahead of the curve. Here's one typical result—*Is Nature's Way The Best Way?*, *Omni*, February 1995, p. 62:

Artificial intelligence has been one of the least successful research areas in computer science. That's because in the past, researchers tried to apply conventional computer programming to abstract human problems, such as recognizing shapes or speaking in sentences. But researchers at MIT's Media Lab and Boston University's Center for Adaptive Systems focus on applying paradigms of intelligence closer to what nature designed for humans, which include evolution, feedback, and adaptation, are used to produce computer programs that communicate among themselves and in turn learn from their mistakes. *Profiles In Artificial Intelligence*, David Freedman.

This is not an argument that AI is impossible, just that it has been unsuccessful. The full text of the article is not given, but it is implied that the argument is that evolution worked for humans, therefore it is a better approach for programs than is "conventional computer programming." This is a common argument, but one that ignores the fact that (a) there are many possible solutions to a problem; one that has worked in the past may not be the best in the present (b) we don't have a good theory of evolution, so we may not be able to duplicate human evolution, (c) natural evolution takes millions of years and for almost all animals does not result in intelligence; there is no guarantee that artificial evolution will do better (d) artificial evolution (or genetic algorithms, ALife, neural nets, etc.) is not the only approach that involves feedback, adaptation and learning. "Conventional" AI does this as well.

## 27.2 Can Machines Really Think?

### Exercise 27.2.#DEFI

Attempt to write definitions of the terms “intelligence,” “thinking,” and “consciousness.” Suggest some possible objections to your definitions.

This also might make a good class discussion topic. A consultation of the literature will divulge many recent attempts, such as that of Chollet, 2019 on intelligence. Here are our attempts:

**intelligence:** a measure of the ability of an agent to make the right decisions, given the available evidence. Given the same circumstances, a more intelligent agent will make better decisions on average.

**thinking:** creating internal representations in service of the goal of coming to a conclusion, making a decision, or weighing evidence.

**consciousness:** being aware of one’s own existence, and of one’s current internal state.

Here are some objections (with replies):

For **intelligence**, too much emphasis is put on decision-making. Haven’t you ever known a highly intelligent person who made bad decisions? Also no mention is made of learning. You can’t be intelligent by using brute-force look-up, for example, could you? [The emphasis on decision-making is only a liability when you are working at too coarse a granularity (e.g., “What should I do with my life?”) Once you look at smaller-grain decisions (e.g., “Should I answer a, b, c or none of the above?”), you get at the kinds of things tested by current IQ tests, while maintaining the advantages of the action-oriented approach covered in Chapter 1. As to the brute-force problem, think of intelligence in terms of an ecological niche: an agent only needs to be as intelligent as is necessary to be successful. If this can be accomplished through some simple mechanism, fine. For the complex environments that we humans are faced with, more complex mechanisms are needed.]

For **thinking**, we have the same objections about decision-making, but in general, thinking is the least controversial of the three terms.

For **consciousness**, the weakness is the definition of “aware.” How does one demonstrate awareness? Also, it is not one’s true internal state that is important, but some kind of abstraction or representation of some of the features of it. Deacon, 2011 offers a novel physical argument for biological naturalism with implications that machines cannot have consciousness. Those particularly interested in this subject might consider attempting to support or refute his argument using the formalism of AI.

### Exercise 27.2.#CHRM

Does a refutation of the Chinese room argument necessarily prove that appropriately programmed computers have mental states? Does an acceptance of the argument necessarily mean that computers cannot have mental states?

No. Searle's Chinese room thesis says that there are some cases where running a program that generates the right output for the Chinese room does not cause true understanding/consciousness. The negation of this thesis is therefore that all programs with the right output do cause true understanding/consciousness. So if you were to disprove Searle's *thesis*, then you would have a proof of machine consciousness. However, what this question is getting at is the *argument* behind the thesis. If you show that the argument is faulty, then you may have proved nothing more: it might be that the thesis is true (by some other argument), or it might be false.

### Exercise 27.2.#GODI

Alan Perlis (1982) wrote, "A year spent in artificial intelligence is enough to make one believe in God". He also wrote, in a letter to Philip Davis, that one of the central dreams of computer science is that "through the performance of computers and their programs we will remove all doubt that there is only a chemical distinction between the living and nonliving world." To what extent does the progress made so far in artificial intelligence shed light on these issues? Suppose that at some future date, the AI endeavor has been completely successful; that is, we have build intelligent agents capable of carrying out any human cognitive task at human levels of ability. To what extent would that shed light on these issues?

The progress that has been made so far — a limited class of restricted cognitive activities can be carried out on a computer, some much better than humans, most much worse than humans — is very little evidence. If all cognitive activities can be explained in computational terms, then that would at least establish that cognition does not *require* the involvement of anything beyond physical processes. Of course, it would still be possible that something of the kind is *actually* involved in human cognition, but this would certainly increase the burden of proof on those who claim that it is.

### Exercise 27.2.#GOOD

I. J. Good claims that intelligence is the most important quality, and that building ultraintelligent machines will change everything. A sentient cheetah counters that "Actually speed is more important; if we could build ultrafast machines, that would change everything," and a sentient elephant claims "You're both wrong; what we need is ultrastrong machines." What do you think of these arguments?

This question asks whether our obsession with intelligence merely reflects our view of ourselves as distinct due to our intelligence. One may respond in two ways. First, note that we already have ultrafast and ultrastrong machines (for example, aircraft and cranes) but they have not changed everything—only those aspects of life for which raw speed and strength are important. Good's argument is based on the view that intelligence is important in all aspects of life, since all aspects involve choosing how to act. Second, note that ultraintelligent machines have the special property that they can easily create ultrafast and ultrastrong machines as needed, whereas the converse is not true.

**Exercise 27.2.#IMPO**

Some critics object that AI is impossible, while others object that it is *too* possible and that ultraintelligent machines pose a threat. Which of these objections do you think is more likely? Would it be a contradiction for someone to hold both positions?

To decide if AI is impossible, we must first define it. In this book, we've chosen a definition that makes it easy to show it is possible in theory—for a given architecture, we just enumerate all programs and choose the best. In practice, this might still be infeasible, but recent history shows steady progress at a wide variety of tasks. Now if we define AI as the production of agents that act indistinguishably from (or at least as intelligently as) human beings on any task, then one would have to say that little progress has been made, and some, such as Marvin Minsky, bemoan the fact that few attempts are even being made. Others think it is quite appropriate to address component tasks rather than the “whole agent” problem. Our feeling is that AI is neither impossible nor a looming threat. But it would be perfectly consistent for someone to feel that AI is most likely doomed to failure, but still that the risks of possible success are so great that it should not be pursued for fear of success.

## 27.3 The Ethics of AI

---

**Exercise 27.3.#WEAP**

Why might state actors support or oppose autonomous weapons, as discussed in Section ??? What role can technologists play in this debate?

Arguments **in support** might include: a desire to adhere to international law, such as the Martens Clause to the Geneva convention which bans weapons that violate the “principles of humanity and the dictates of public conscience” (Strelbel, 1982); a desire to establish treaties regulating arms in lieu of an arms race and the security threat such a race might impose, as has occurred with nuclear weapons and with biological agents under the Nixon administration in the United States (as, for example, argued in Russell, 2015 and discussed in Section ???). Of course there will be some parties who do not adhere to such treaties (such as non state entities), but these will likely be of negligible impact compared to that of large nations, although recall the **dual use** problem described in the text.

This presages arguments **in opposition**: dictators, terrorist groups, and repressive regimes may want access to cheaper and deadlier weapons for obvious reasons; even established nations may want to reduce their own casualties (and training costs) by reducing battlefield exposure of soldiers; and autonomous weapons may even be able to reduce casualties overall under the assumption that because the threat to an autonomous weapon is small compared to the threat to a human (presumably we value a lost robot much less than a lost life), the weapon can wait to see how a dangerous situation plays out in a risky or otherwise unknown situation. Robots don't have to shoot first.

As evident in previous arms control movements, technologists, or those with an intimate

knowledge of the weapon technologies, can play quite significant roles. For example, physicist Robert Oppenheimer, who was instrumental in the production of the first atomic bomb, famously became an outspoken opponent of nuclear proliferation after the conclusion of the Second World War. Computer scientists have done the same, such as Norbert Wiener, one of the founders of control theory and famous for his work on anti-aircraft guns, who spoke out against the militarization of computing technology after the conclusion of WWII. The now-defunct Computing Professionals for Social Responsibility and many other organizations have continued in Wiener's legacy. This legacy is notably similar to the expert recommendations with regard to the appropriate use of technology as advocated by the authors of this book. Similar trends have cropped up in the late 2010s movement at major computing companies in which employees began to debate the utility of furnishing such military contracts for government agencies (such as the U.S. DoD). The effects of these actions are not yet apparent and will be borne out in time.

### Exercise 27.3.#NANO

How do the potential threats from AI technology compare with those from other computer science technologies, and to bio-, nano-, and nuclear technologies?

Biological and nuclear technologies provide much more immediate threats of weapons, yielded either by states or by small groups. Nanotechnology threatens to produce rapidly reproducing threats, either as weapons or accidentally, but the feasibility of this technology is still quite hypothetical. As discussed in the text and in the previous exercise, computer technology such as centralized databases, network-attached cameras, and GPS-guided weapons seem to pose a more serious portfolio of threats than AI technology, at least as of today.

### Exercise 27.3.#FAIR

Your company is working on a new model to sort people into a predicted class,  $\hat{y} = \langle 0, 1 \rangle$ . In the table shown, you can see the results of the model over a population of 20 samples of the true classes,  $y = \langle 0, 1 \rangle$ , for people across two groups,  $a$  and  $b$ . Notice the mislabelled points in each entry.

|           | $\hat{y} = 0$         | $\hat{y} = 1$   |
|-----------|-----------------------|-----------------|
| Group $a$ | [0, 0, 0, 0, 0, 0, 1] | [1, 1, 0]       |
| Group $b$ | [0, 0, 0, 0, 1]       | [1, 1, 1, 1, 0] |

- Is the model **well calibrated**? If not, how might that be achieved with these data?
- Does the model achieve **equal outcome**? If not, how might that be achieved with these data?
- Does the model achieve **equal opportunity**? If not, how might that be achieved with these data?
- Does the model have **equal impact**? If not, how might that be achieved with these data?
- What constitutes a sensitive group? What if the groups,  $\langle a, b \rangle$ , were a person's race, such as, in the United States, black and white? What if the groups were a person's height, tall and short?

- f.** What constitutes a sensitive classification? What if this model was used to decide whether to give a longer sentence given predicted recidivism? What about if the model was used to decide whether to display an ad for shoes given predicted efficacy of the ad?
- g.** What other concerns, besides those mentioned, might arise in such attempts at classification?
- h.** What recommendations might you make to this company with regard to algorithmic fairness? Choose one recommendation and expand to describe how it might be implemented.

- a.** Yes, the model is well calibrated; the proportion for each estimated label is equal to proportion of the true label:  $P(\hat{y} = 0 | a) = P(y = 0 | a) = \frac{7}{10}$ ,  $P(\hat{y} = 0 | b) = P(y = 0 | b) = \frac{5}{10}$ ,  $P(\hat{y} = 1 | a) = P(y = 1 | a) = \frac{3}{10}$ , and  $P(\hat{y} = 1 | b) = P(y = 1 | b) = \frac{5}{10}$ .
- b.** The model does not achieve equal outcome because  $P(\hat{y} | a) \neq P(\hat{y} | b)$ . These data might be classified by an alternative model to meet this condition as in the following table. Nonetheless, notice how this condition compromises whether the model is well balanced and does not necessarily entail equal opportunity, although it can.

|                | $\hat{y} = 0$         | $\hat{y} = 1$ |
|----------------|-----------------------|---------------|
| Group <i>a</i> | [0, 0, 0, 0, 0, 0, 1] | [1, 1]        |
| Group <i>b</i> | [0, 0, 0, 0, 0, 1, 1] | [1, 1]        |

- c.** The model does not achieve equal opportunity because the proportion of false positives is not equal between the groups:  $P(\hat{y} = 1 | a, y = 0) = \frac{1}{3} \neq P(\hat{y} = 1 | b, y = 0) = \frac{1}{5}$ .

Given that the base rates between the classes are different,  $P(y = 0 | a) \neq P(y = 0 | b)$ , and we assume some degree of mislabeling on the part of the model,  $P(\hat{y} = 0) \neq P(y = 0)$ , equal opportunity is not possible to achieve unless we compromise the calibration of the model (Kleinberg, 2016). If we were willing to make such a compromise, an example classification output of these data using a new model might be as follows. Notice that we ensuingly compromise the false negative rate as well, while in the first model  $P(\hat{y} = 0 | b, y = 1) = \frac{1}{5}$ , now  $P(\hat{y} = 0 | b, y = 1) = \frac{3}{7}$ .

|                | $\hat{y} = 0$         | $\hat{y} = 1$ |
|----------------|-----------------------|---------------|
| Group <i>a</i> | [0, 0, 0, 0, 0, 0, 1] | [1, 1, 0]     |
| Group <i>b</i> | [0, 0, 0, 0, 1, 1, 1] | [1, 1, 0]     |

- d.** In this case given the lack of information about the classification itself, it is impossible to say whether the model achieves equal impact, that being an extension of equal opportunity with respect to the expected utility of ensuing events caused by a classification. A sufficient answer might explain that or even formulate the costs and benefits of the classifications,  $\hat{y}$ , into expected utilities as Chouldechova (2017) does for the disparate impact of criminal risk sentencing.
- e.** Being fairly open-ended and value based, this question and the next might befit a classroom discussion. What constitutes a sensitive group is of great social, legal, and political importance. Depending on the country such groups will often have some degree of legal protection, such as race, color, national origin, sex, or religion in United States

discrimination law. This is not always the case as is apparent with current and quite recent discussions of sexual orientation, gender, ableness, and more.

- f. Nonetheless, some applications of classification will demand much greater attention than others, perhaps as related to the expected impact of a classification. Clearly criminal risk sentencing demands greater attention than predictive advertising, but, as mentioned in the text, the scale of the latter as well as particular uses (such as for housing or employment) will necessitate attention as well.
- g. It is a basic principle of statistics that data as well as the models based on them are biased; the observer effect tells us that as soon as we measure (decide upon some paradigmatic measure into which we project the world) we introduce bias into our data. Furthermore, see the discussion in the text with regard to whether a defendant has committed a crime as opposed to whether they have been convicted.
- h. A suitable answer might expand on any of the bullet points identified in fairness and bias section. For example, using the above exercise on a real data set and model might constitute, "Examine your data for prejudice..." Alternatively to "Understand how any human annotation of data is done" one might need to systematically (by hand) go through the annotation system and classifications used, looking for those that stand out such as Crawford (2019) do with regard racist and sexist classifications in ImageNet.

### Exercise 27.3.#3REV

Concerns regarding the effect of technology on the future of work are not new. In a 1964 memo to President Lyndon B. Johnson, the Ad Hoc Committee of the Triple Revolution (1964), one comprised of notable social activists, scientists and technologists, warned of revolutions in social justice, automation, and weapons development. Like Karl Marx, John Maynard Keynes, and countless thinkers today, the committee realized that the cybernation revolution, "one brought about by the combination of the computer and the automated self-regulating machine," could free people from work, but that it faced many challenges in doing so.

There is no question that cybernation does increase the potential for the provision of funds to neglected public sectors. Nor is there any question that cybernation would make possible the abolition of poverty at home and abroad. But the industrial system does not possess any adequate mechanisms to permit these potentials to become realities. The industrial system was designed to produce an ever-increasing quantity of goods as efficiently as possible, and it was assumed that the distribution of the power to purchase these goods would occur almost automatically. The continuance of the income-through-jobs link as the only major mechanism for distributing effective demand—for granting the right to consume—now acts as the main brake on the almost unlimited capacity of a cybernated productive system.

- a. Using evidence from the text, do the quoted arguments ring true today?
- b. Relate this passage to the question raised in the text about what to do with regard to automation, "do we want to focus on *cutting cost*, and thus see job loss as a positive; or do we want to focus on *improving quality*, making life better for the worker and the customer?" How does the answer to the previous question change based on which

stakeholder (e.g. a business owner, a worker, a consumer, a policy maker, etc.) is answering it? In the end, who does make those decisions around automation?

- a. A suitable response here would include an historical discussion of **technological unemployment**, the magnification of **income inequality**, and relevant cautions regarding general ignorance of the **pace of change** of those effects. Certainly one can find these arguments repeated today, but what we do not know is whether the arguments are mainly accurate and have not been addressed or whether the arguments are mainly inaccurate and nevertheless have much cachet. Students will necessarily differ in their attributions of the causes: the degree to which a given technology or set of technologies is responsible for an ensuing job loss or inequality. A more nuanced answer will address the degree to which current (and preceding) technologies on the whole do not strictly displace workers (although some do), but rather change the nature of work (one's agency over it, latency of communication, location, and other trends related to globalization). Consider: What do we gain in such an accelerating economy? More happiness? More free time? More toys? These, and other, questions are raised by Wajcman (2014).
- b. and c. Clearly the position of a stakeholder will change the degree to which they are interested in cutting costs or improving quality; in a simple consideration, a business owner wishes to reduce cost while an employee wishes to increase the quality of their life. Given the state of the world, we might then reasonably presume that it is business owners and their ilk who are the ones making decisions around automation, and workplaces in general. But, then, do workers and consumers have no say? Are they not those who elect the policy makers (in democratic countries) who create the rules for businesses? Such complications, while well warranted, expand the scope of this question to political science—there have certainly been many movements, some with spottier records than others, to give workers control over the means of production. Nonetheless, it appears reasonable to assume that most people, regardless of their position, if asked this question, would assert that they are more interested in improving the quality of their life rather than cutting cost (which could be construed as an instrumental goal for the latter). As much psychology research shows us: we do not usually act in our best interests as utility theory purports we ought. It is evident that many people do not make decisions that would be seen to increase their quality of life.

### Exercise 27.3.#NRGY

The text describes some of the **unintended side effects** which might result from artificial intelligence technologies, perhaps from **externalities** in the objective function of a system. Here consider one potential externality: energy use (and, by proxy, greenhouse gas emissions) associated with training deep learning models.

Strubell et al. (2019) make the case that modelers ought to pay more attention to the demands of training, citing Amodei (2018) as indicating how popular deep learning models increased the amount of compute time required by some 300,000 fold in between 2012 and

2018, a trend that appears to continue at the time of this writing. Indeed, Strubell et al. estimate that training a single big transformer model, the state of the art for many natural language processing tasks, would use 192 lbs  $CO_2$ , about a tenth of the emissions released by a passenger flying from New York to San Francisco and back. They estimate that training the well-known BERT model would take 1438 lbs  $CO_2$ , a bit under that round trip cost for a passenger on a cross-country flight. Both of these figures grow thousands of times larger when neural architecture search, tuning, and other hyperparameter experiments are run.

- a. Why might deep learning models be growing by such a scale? (Namely, training operations are doubling every 3.4 months as opposed to the every 2 years which used to describe the rate of change of Moore's law Amodei (2018).)
- b. The cited calculations are rife with uncertainty. Where do you think that uncertainty arises?
- c. What might be done to change the objective function of these models (or of the community of modelers) to incorporate the costs of energy?

- a. The period which Amodei (2018) use to begin the increase in the scale of deep learning models is the emergence of AlexNet, the first major demonstration of the efficacy of deep learning approaches to the broader community. Now, as has previously been demonstrated for high dimensional data, the accuracy of such neural network approaches tends to increase as their capacity increases and as training time increases exponentially as noted in Chapter 21. While the costs of training such models is certainly immense, one presumes that the expected utility of such models is even greater (think of the competition between and the resources of today's technology giants). Thus, because of the increased accuracy of such models and the awareness of such accuracy, it is not surprising that more compute time is being invested into training them. Also see the discussion of the emergence of deep learning in the historical notes of that chapter.
- b. Uncertainty arises because of a number of factors. Most modelers do not report any sort of proxy which could be used to compare compute times and those proxies which are reported are quite vague and are machine dependent. For example, computing a single floating point operation on one machine versus another may result in quite different uses of electricity and the grid system in which a machine sits largely determines the degree to which the electricity used actually results in more greenhouse gases. Lacoste (2019) breaks down the average green house gas emissions for different server regions, finding that the average carbon output of servers in Australia is more than twice that of North America.
- c. There have been a few recent attempts to raise awareness of the exponentially increasing costs of training through appeals to the research community, such as those already cited as well as Bender (2021). Others, such as Henderson (2020) and Lacoste (2019) advocate for new training benchmarks, software packages to more easily measure compute time, and the like, measures which could feasibly be specified directly in the objective function of some an algorithm as a negative cost or constraint on model capacity or training epochs. Others, like Hamerly (2019), propose alternative and less energy-

intensive computational substrates, such as light.

#### Exercise 27.3.#THRT

Analyze the potential threats from AI technology to society. What threats are most serious, and how might they be combated? How do they compare to the potential benefits?

It is hard to give a definitive answer to this question, but it can provoke some interesting essays. Many of the threats are actually problems of computer technology or industrial society in general, with some components that can be magnified by AI—examples include loss of privacy to surveillance, the concentration of power and wealth in the hands of the most powerful, and the magnification of particular societal biases. As discussed in the text, the prospect of robots taking over the world does not appear to be a serious threat in the foreseeable future.

#### Exercise 27.3.#SOCI

Compare the social impact of artificial intelligence in the last fifty years with the social impact of the introduction of electric appliances and the internal combustion engine in the fifty years between 1890 and 1940.

The impact of AI has thus far been extremely small, by comparison. In fact, the social impact of *all* technological advances between 1970 and 2020 has been considerably smaller than the technological advances between 1890 and 1940. The common idea that we live in a world where technological change advances ever more rapidly is out-dated. It is nonetheless the case that, relative to contemporaneous innovations, artificial intelligence is having quite a large impact.

#### Exercise 27.3.#CYBG

Recall the discussion of **transhumanism** from the text. Similar considerations have been made with regard to cyborgs, cybernetic organisms. Either individually or in a discussion group, consider:

- a. What is a cyborg? (Perhaps look up cybernetic, too.)
  - b. Are you a cyborg? Do you know any cyborgs? (For example, consider existing smartphones, prosthetic devices, SCUBA gear, social media, and technologies both new and ancient.)
  - c. Now, do you think your great-great-grandparents would agree with your answer to the previous question? What about your great-great-grandchildren?
- 
- a. Definitions of cyborg (or human vs. transhuman for that matter) are far ranging. The aim here is to challenge students on their assumptions. Does a piece of technology have to be fully integrated with a human for them to be designated a cyborg? Answers may

include a discussion of necessity (can the modified organism survive without the modification? What about current medical technology, then? Does having a single life-saving surgery make an organism a cyborg? Over what time scale must a modification apply?), prevalence (what proportion of organisms—staying aware of what population we purport to sample from—have any given modification?), composition (does the cybernetic system have to be deterministic, command and control-like as in early considerations of AI? What about inferential systems? Must its parts be inorganic? Is a genetically-modified organism a cyborg? Must the modification be controlled using neurological signals? Why not a remote?), origin (does the origin of a modification affect the designation of cyborg? What if the part is made by evolution? What about if it is made by a person? What about evolutionary processes and people, like evolutionary algorithms?), effect (does the modification enhance or detract from an aspect of the organism, such as fitness?), and more.

- b.** This question and question (c) recapitulate the previous question meant to encourage students to ground their answers in examples. One might extend this exercise by challenging students' definitions with further examples—perhaps adversarially by looking for edge cases. If a student does not think that they are a cyborg from their current frame of reference, might someone in a dramatically different frame of reference disagree? To what degree is our definition of cyborg (and human!) a construction of what is not yet the case (must a cyborg always be something not yet actualized)?

# EXERCISES 28

## THE FUTURE OF AI

[[need exercises]]

### 28.1 AI Components

---

#### Exercise 28.1.#FUTP

Compare the price of various component technologies used in AI, and create charts for how those prices have changed over the last twenty years.

The answers are changing quickly, and will depend on exactly what month this assignment is given. Components to compare include CPUs, GPUs, other specialized processors, cloud computing services, light sensors such as digital cameras, active sensors such as lidar and radar, and actuators such as robot arms, motors, and chassis.

[[need exercises]]

### 28.2 AI Architectures

---

[[need exercises]]