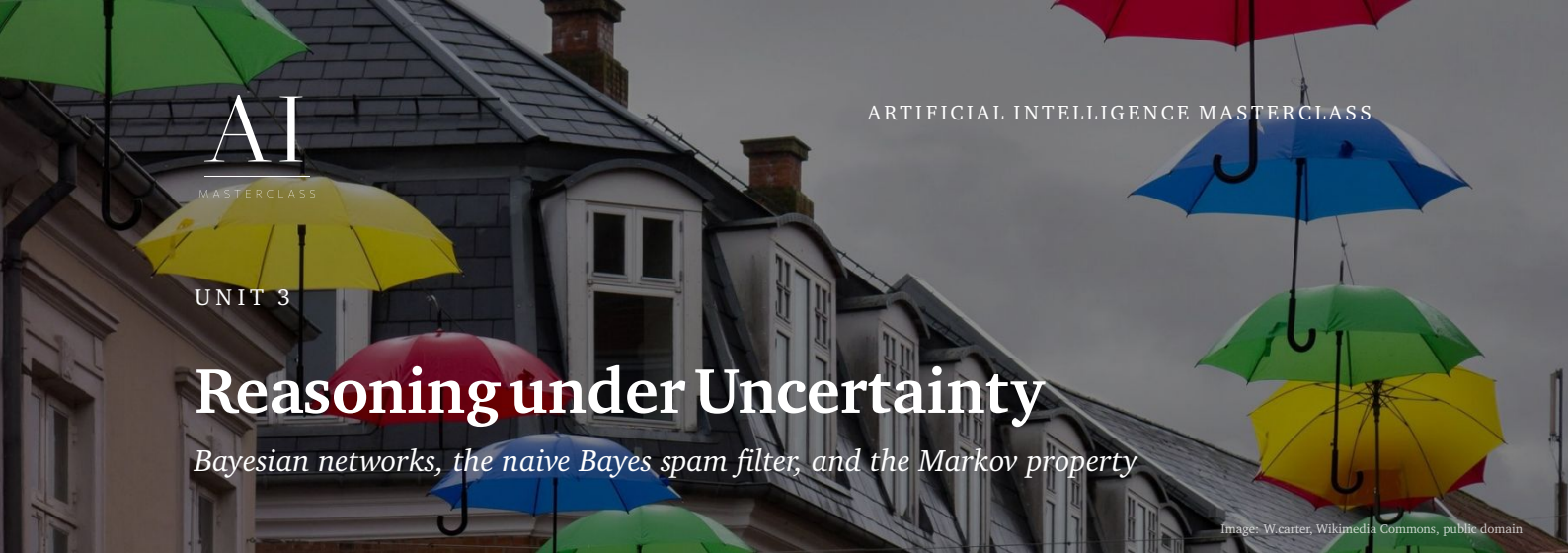


UNIT 3

Reasoning under Uncertainty

Bayesian networks, the naive Bayes spam filter, and the Markov property

Image: W.carter, Wikimedia Commons, public domain

By the end of this unit, I should be able to answer these questions:

- Why can a joint probability table for a few dozen yes-or-no questions not be stored on any computer, and what does conditional independence do about that?
- How does a Bayesian network encode a situation, and how do I compute the probability of a cause given an observed effect from its tables?
- Why does a positive result of a good test for a rare disease still leave the disease unlikely, and what changes when a second independent test is also positive?
- How does a naive Bayes spam filter decide, and why is the assumption in its name wrong and useful at the same time?
- What is the Markov property, and what does a hidden Markov model hide?

A robot's distance sensor is right most of the time and wrong now and then, a cough is usually a cold and occasionally something worse, and the word "free" appears in most spam and in some perfectly ordinary mail. A machine that acts in the world must decide with evidence of this kind, and logic, which knows only true and false, cannot hold a belief that is 80 per cent sure. This unit shows how a machine keeps degrees of belief in many quantities at once without drowning in numbers, how it updates them when new evidence arrives, and how the same idea filters spam and follows the weather from day to day.

1 Degrees of belief and the joint distribution

Unit 1 introduced a probability as a number attached to an event; this unit reads it as a degree of belief. The two rules of Unit 1, the sum rule and the product rule, are all that is needed here, and the rest of this section is about one object that follows from them. The **joint distribution** of a set of random variables assigns a probability to every combination of values they can take together. For two yes-or-no variables that is a table of four numbers, and the numbers add up to one. To illustrate this, Table 1 gives the joint distribution of two variables of a doctor's morning, whether a patient has influenza and whether the patient has a fever. The numbers are set by hand for this example and are reused in Section 2.

	Fever	No Fever
Flu	0.0450	0.0050
No flu	0.0475	0.9025

Table 1. The joint distribution of the two variables flu and fever. Each cell is the probability that a patient has that combination, and the four cells add up to one.

The table is the complete description of the situation, because every question about the two variables can

be answered from it with the two rules of Unit 1. The probability of a fever is the sum of the column, $P(\text{fever}) = 0.0450 + 0.0475 = 0.0925$, so that about 9 patients in 100 have one, and the probability of flu given a fever is the ratio

$$P(\text{flu} \mid \text{fever}) = \frac{P(\text{flu}, \text{fever})}{P(\text{fever})} = \frac{0.0450}{0.0925} = 0.486,$$

so that about half of the patients with a fever have flu. Summing a row or a column is called **marginalization**, after the habit of writing the sums in the margin of the table, and dividing by the probability of what has been observed is **conditioning**. The whole of probabilistic reasoning consists of these two operations applied to a joint distribution, and the difficulty of the subject lies entirely in the size of that distribution.

WORKED EXAMPLE The joint table of 30 yes-or-no questions

A modest medical situation might involve 30 yes-or-no variables, such as ten diseases, fifteen symptoms, and five test results. The joint distribution has one entry for every combination, which is $2^{30} = 1{,}073{,}741{,}824$ entries, a little over a billion. Since the entries add up to one, the last entry follows from the others, so the table has $2^{30} - 1$ free numbers. Stored in the 32-bit floating point format that Unit 1 introduced, with 4 bytes per number, they take 4.29 GB, about half the working memory of a small laptop, for a situation with only 30 variables. With 300 variables the table would have $2^{300} \approx 10^{90}$ entries, which exceeds the number of atoms in the observable universe, about 10^{80}. The memory is the smaller problem. Every one of those numbers would have to be estimated from data, and no hospital has ever seen a billion distinct combinations of symptoms.

Section 2 represents the same 30 variables by a network in which each variable depends directly on at most two others. Each variable then needs a table with at most $2^2 = 4$ rows, one for each combination of values of the two variables it depends on (its parents, as Section 2 will call them), and one number per row, so the whole network needs at most $30 \times 4 = 120$ numbers. The joint table needs about 10^9 numbers, the network at most 120, and the network can be filled in from a few thousand patient records.

The saving in the example comes from one idea, which is called conditional independence. Two variables A and B are **conditionally independent** given a third variable C when, once the value of C is known, learning B tells nothing more about A. In the doctor's morning, for example, once a cough is added to the flu and the fever of Table 1, a patient's fever and cough are related, since both are more common with flu, but once the doctor knows that the patient has flu, the fever says nothing further about whether to expect a cough. The relation between fever and cough runs entirely through the flu.

KEY EQUATION Conditional independence

$$P(A \mid B, C) = P(A \mid C) \qquad \text{or equivalently} \qquad P(A, B \mid C) = P(A \mid C)\, P(B \mid C)$$

The variables A and B are conditionally independent given C when the probability of A no longer changes with B once C is known; the comma in $P(A \mid B, C)$ means that both B and C are known. Plain independence, $P(A, B) = P(A)\, P(B)$, is the special case with no conditioning variable, and the two are different properties. Two variables can be dependent and still conditionally independent given a third, and the other way round.

The last sentence of the box deserves an example, because it is the point at which beginners most often go wrong. Over the summer, ice cream sales and the number of drownings at a beach rise and fall together, so the two are dependent, and someone who knew the day's ice cream sales could guess the number of drownings better than someone who did not. Once the day's temperature is known, however, the sales add nothing, because both numbers depend on the heat and on nothing else that links them. The two are conditionally independent given the temperature.

CHECK YOUR UNDERSTANDING Question 1

At a beach, the daily ice cream sales and the daily number of drownings rise and fall together over the year. Which statement describes the situation correctly?

(A) The two are independent, because neither one causes the other

(B) The two are dependent, and conditionally independent given the day's temperature

(C) The two are conditionally dependent given the temperature, because both rise with the heat

(D) The two are independent, but become dependent once the temperature is known

The solution is at the end of the unit.

2 Bayesian networks

A **Bayesian network** writes the conditional independences of a situation as a drawing and the remaining dependences as small tables. It has one node for each random variable and an arrow from a node to every variable that depends directly on it. The nodes from which arrows arrive at a node are its **parents**, and the arrows may not form a cycle, so that one can never follow arrows from a node back to itself. Each node carries a **conditional probability table**, usually abbreviated to CPT, with one row for every combination of values of its parents and, in that row, the probability of each value of the node. A node without parents carries a plain probability. Figure 1 shows a network for the doctor's morning with four variables, where influenza and a common cold are the two possible causes of the symptoms. The name of the structure goes back to a paper of 1985 by Judea Pearl of the University of California, Los Angeles, and his book *Probabilistic Reasoning in Intelligent Systems*, published in 1988, made the networks the standard tool for uncertain reasoning in artificial intelligence; he received the Turing Award for 2011 largely for this work.

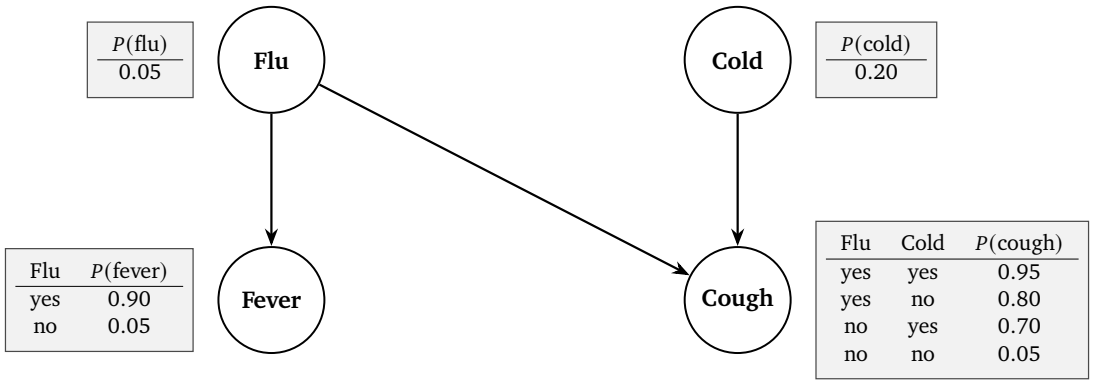


Figure 1. A Bayesian network with four yes-or-no variables and its conditional probability tables. Flu and a cold have no parents and carry plain probabilities; fever depends on flu alone; cough depends on both. Each table lists the probability of "yes" for every combination of the parents' values, and the probability of "no" is one minus that. The numbers are set by hand for this unit.

The tables in the figure hold $1 + 1 + 2 + 4 = 8$ numbers. A joint table of four yes-or-no variables has $2^4 = 16$ cells and 15 free numbers, so even this small network nearly halves the count, and the saving grows with every variable that is added, as Section 1 showed for 30 of them. The absence of an arrow from cold to fever is the claim that a cold does not change the chance of a fever once flu is accounted for, and the absence of an arrow between flu and cold is the claim that catching one does not change the chance of catching the other. Arrows usually point from cause to effect, because then the tables are the ones a doctor can estimate, such as the probability of a fever given flu, but the mathematics does not require it; a network is a statement about which variables are conditionally independent of which, and nothing more.

The joint distribution of all the variables follows from the tables by a rule that Unit 1 used in its two-variable form. Applying the product rule repeatedly, one variable at a time, gives the chain rule of probability, and in a Bayesian network each factor of the chain simplifies, because a variable is conditionally independent of everything that is not its descendant once its parents are known. The descendants of a node are the nodes that can be reached from it by following arrows forwards.

KEY EQUATION The chain rule of probability and the factorization of a Bayesian network

$$P(x_1, x_2, \ldots, x_n) = P(x_1)\, P(x_2 \mid x_1)\, P(x_3 \mid x_1, x_2) \cdots P(x_n \mid x_1, \ldots, x_{n-1})$$
$$= \prod_{i=1}^{n} P\big(x_i \mid \text{parents}(x_i)\big)$$

The first equality is the chain rule and holds for any variables in any order. The second holds in a Bayesian network, where $\text{parents}(x_i)$ are the variables with arrows into x_i, and it says that the joint probability of any complete assignment, that is, of one chosen value for every variable, is the product of one entry from each conditional probability table.

For the network of Figure 1, the probability that a patient has flu, no cold, a fever, and a cough is $P(\text{flu})\, P(\text{no cold})\, P(\text{fever} \mid \text{flu})\, P(\text{cough} \mid \text{flu}, \text{no cold}) = 0.05 \times 0.8 \times 0.9 \times 0.8 = 0.0288$, which is one entry of the joint table, computed on demand from four small numbers. Any other entry is found the same way.

2.1 Inference by enumeration

A question put to a network has the form "what is the probability of this variable, given that I have observed those?" The observed variables are the **evidence**, the variable asked about is the **query**, and the remaining ones, which are neither, are **hidden**. The method that answers it directly from the factorization is called **inference by enumeration** and has four steps.

1. Write the joint probability of all variables as the product of the tables.
2. Put the observed values of the evidence into every factor.
3. For each value of the query variable, add up the product over every combination of values of the hidden variables, which is the marginalization of Section 1.
4. Divide each sum by the total of the sums, so that the results add up to one, which is the conditioning of Section 1.

WORKED EXAMPLE The probability of flu given a fever and a cough

A patient of the network of Figure 1 arrives with a fever and a cough. The query is flu, the evidence is fever and cough, and the hidden variable is cold, which has to be summed over. With the evidence fixed, the product of the tables for a patient with flu is

$$P(\text{flu}, \text{fever}, \text{cough}) = P(\text{flu})\, P(\text{fever} \mid \text{flu}) \sum_{\text{cold}} P(\text{cold})\, P(\text{cough} \mid \text{flu}, \text{cold}),$$

and the sum runs over the two values of cold. Only the last two factors stand inside the sum, because they are the only ones whose value changes with cold; the first two do not depend on it and are taken out in front. With the numbers of the figure, where 0.2×0.95 is the term for a cold and 0.8×0.8 the term for no cold,

$$0.05 \times 0.9 \times (0.2 \times 0.95 + 0.8 \times 0.8) = 0.045 \times 0.83 = 0.03735.$$

For a patient without flu the same product reads

$$0.95 \times 0.05 \times (0.2 \times 0.7 + 0.8 \times 0.05) = 0.0475 \times 0.18 = 0.00855.$$

The two numbers are the probabilities of the two complete cases, flu with these symptoms and no flu with these symptoms, and the last step divides by their total of 0.0459:

$$P(\text{flu} \mid \text{fever}, \text{cough}) = \frac{0.03735}{0.03735 + 0.00855} = 0.814.$$

With both symptoms the patient has flu with a probability of about 81 per cent. The same calculation with the fever alone gives 49 per cent, which is the value read from Table 1, and with the cough alone only 20 per cent, because a cough is well explained by the far more common cold.

Enumeration is exact, and for a network of a dozen variables it is quick. It fails for large networks for the same reason the joint table failed, since the sum in step 3 runs over every combination of the hidden variables, and twenty hidden yes-or-no variables mean 2^{20} terms, about a million, for each value of the query. Cleverer exact methods reuse partial sums, but Gregory Cooper proved in 1990 that exact inference in a general network is NP-hard, which means that no known method is fast for every network, so large networks are answered approximately, by methods that draw random samples from the network and count; those methods lie beyond this unit.

RESEARCH SPOTLIGHT

Zoubin Ghahramani *University of Cambridge*

A machine that knows what it does not know

Zoubin Ghahramani wrote a review in *Nature* in 2015 that asks how a machine should learn from experience, and answers that it should represent everything it is unsure about as a probability distribution and update that distribution with the rules of this unit. A learning machine in this view carries a model of the world, a belief about the model's parameters before any data have been seen, and a rule for how probable the data are under each setting of the parameters; learning is the updating of that belief with the data, which Section 3 will call computing the posterior, and a prediction averages over all parameter values weighted by how plausible each one still is. The review places under this one recipe the Bayesian networks of this section, the classifier of Section 4, and the sequence models of Section 5, and it presents four lines of research built on it, among them probabilistic programming languages, in which the programmer writes down the model and the computer works out the inference, and Bayesian optimization, which chooses the next experiment where the expected gain is largest. Its argument is that a system which reports its uncertainty can decide when to gather more data, when to act, and when to hand a decision to a person, and that a system which reports a single answer cannot. This unit is the first step of the programme the review describes.

Read the paper: Ghahramani (2015), Probabilistic machine learning and artificial intelligence, Nature 521, 452

3 Bayes' theorem and the base rate

The commonest question put to a network has one cause and one observed effect, and for that case the enumeration of the last section collapses into a single formula, which Unit 1 introduced as Bayes' rule and which is stated here in its general form with the names that every later unit uses.

PRINCIPLE Bayes' theorem

For a hypothesis H and evidence E with $P(E) > 0$, and with $\neg H$ standing for "not H",

$$P(H \mid E) = \frac{P(E \mid H)\,P(H)}{P(E)}, \qquad P(E) = P(E \mid H)\,P(H) + P(E \mid \neg H)\,P(\neg H).$$

$P(H)$ is the **prior**, the degree of belief in H before the evidence; $P(E \mid H)$ is the **likelihood**, the probability of the evidence if H is true; $P(H \mid E)$ is the **posterior**, the belief after the evidence; and $P(E)$ is the probability of the evidence, which adds the two ways in which it can arise. The theorem follows from the product rule written in both orders, $P(H, E) = P(E \mid H)\,P(H) = P(H \mid E)\,P(E)$, and holds for any two events.

Unit 1 applied the theorem to a disease with a prevalence of 1 per cent, which means that one person in 100 has it, and to a test that is 95 per cent sensitive and 95 per cent specific. Sensitive means that the test is positive in 95 per cent of the ill, and specific means that it is negative in 95 per cent of the healthy. Unit 1 found that a positive result leaves the patient ill with a probability of only about 16 per cent. The result surprises almost everyone the first time, and the mistake behind the surprise has a name. The **base rate fallacy** is the habit of judging the evidence, a positive result from a good test, while forgetting the prior, which here is the prevalence, the base rate of the disease, and says that the disease is rare. Figure 2 shows the cure for the fallacy, which is to think in whole people instead of probabilities. Among 1,000 people, 10 have the disease and about 10 of them test positive, while 990 do not have it and about 50 of them test positive all the same. A positive result therefore comes from one of about 60 people, of whom 10 are ill, and 10/60 is the 16 per cent. Doctors and patients who are given the numbers in this form, as counts in a population, get the answer right far more often than those who are given percentages, a finding that Gerd Gigerenzer and colleagues established in a series of studies from the 1990s on.

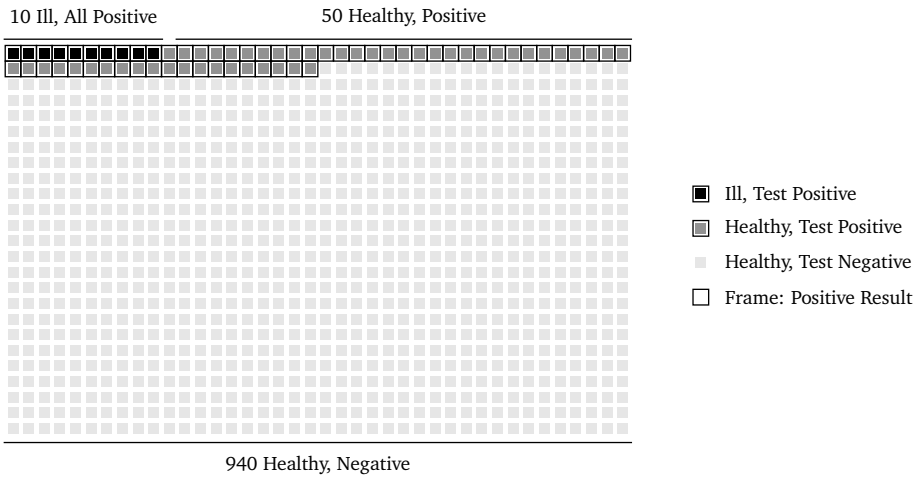


Figure 2. One thousand people, of whom 10 have the disease (dark) and 990 do not (light). The test marks about 10 of the ill and about 50 of the healthy as positive, drawn with a frame around them. Of the roughly 60 people with a positive result, 10 are ill, so a positive result means the disease with a probability of about 16 per cent. The counts of 9.5 and 49.5 are rounded to whole people.

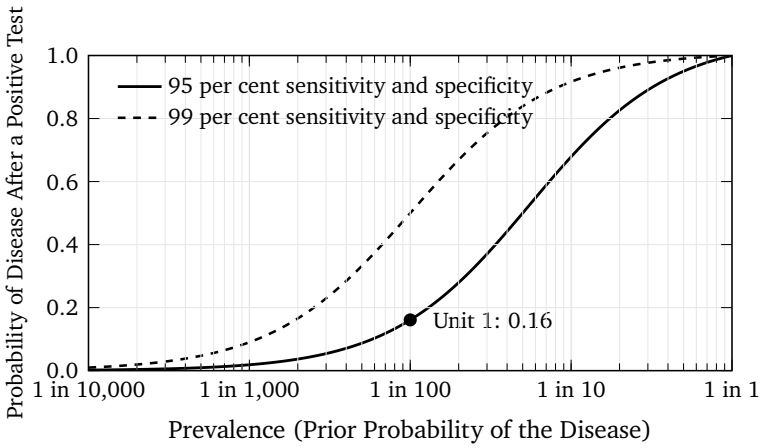
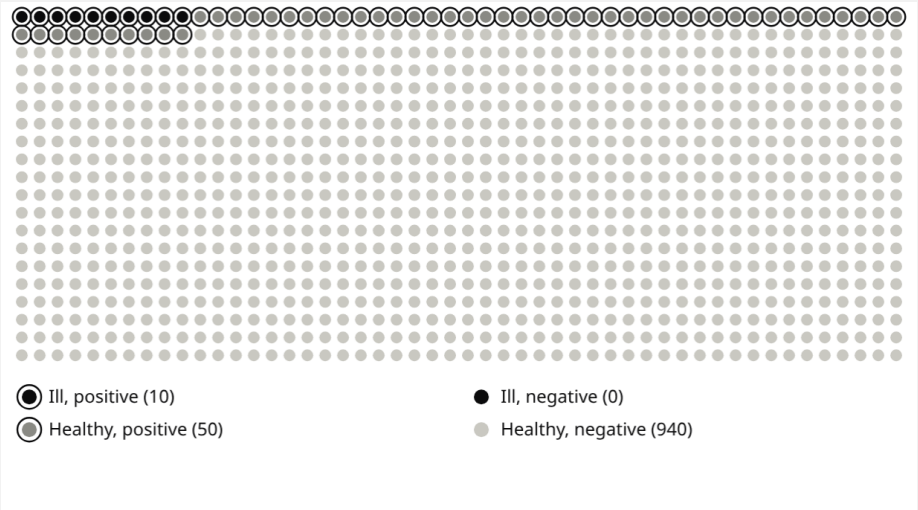


Figure 3. The probability of the disease after one positive result, plotted against the prevalence on a logarithmic axis, computed from Bayes' theorem for a test with 95 per cent sensitivity and specificity (solid) and for one with 99 per cent of each (dashed). The dot marks the case of Unit 1.

The posterior depends on the prior as strongly as on the test. Figure 3 plots the probability of the disease after a positive result against the prevalence for the test above and for a much better one. At a prevalence of 1 in 10,000, as in screening a whole population for a rare condition, even the better test leaves a positive person healthy with a probability of 99 per cent. At a prevalence of 50 per cent, which is roughly the situation of a doctor who tests only patients whose symptoms already point to the disease, the same tests give posteriors of 95 and 99 per cent. The test has not changed; the prior has.

INTERACTIVE A positive test in a population of 1,000



Move the prevalence, the sensitivity, and the specificity, or pick one of the three situations, and watch the 1,000 people regroup: the ill who test positive, the healthy who test positive, and the rest. The readout gives the probability of the disease after a positive and after a negative result, computed from Bayes' theorem. Notice that at a prevalence of 1 in 1,000 even a test with 99 per cent sensitivity and specificity leaves most positive people healthy.

Click the picture or this link to try it in the online version of this unit.

3.1 Two tests

A second test is the natural next step, and the cleanest way to see what it does is to write Bayes' theorem in terms of odds. The **odds** of a hypothesis are the ratio $P(H)/P(\neg H)$, so that a probability of 0.2 is odds of $0.2/0.8 = 1/4$, or 1 to 4, and the **likelihood ratio** of a piece of evidence is $P(E \mid H)/P(E \mid \neg H)$, the factor by which the evidence is more probable if the hypothesis is true than if it is false.

DERIVATION Bayes' theorem in odds form

Write the theorem once for H and once for $\neg H$ and divide the two:

$$\frac{P(H \mid E)}{P(\neg H \mid E)} = \frac{P(E \mid H)\,P(H)}{P(E \mid \neg H)\,P(\neg H)}.$$

The probability of the evidence $P(E)$ cancels, and the result reads, in words, that the posterior odds are the prior odds multiplied by the likelihood ratio. If a second piece of evidence E_2 is conditionally independent of the first given the hypothesis, so that $P(E_2 \mid E_1, H) = P(E_2 \mid H)$ and the same for $\neg H$, then the posterior after E_1 serves as the prior for E_2, and the odds are multiplied by the second likelihood ratio in turn. Independent pieces of evidence multiply the odds; they do not add the probabilities.

WORKED EXAMPLE A second positive test

For the test of Unit 1, a positive result is 95 per cent probable if the patient is ill and 5 per cent probable if not, so its likelihood ratio is $0.95/0.05 = 19$. The prior odds of the disease are $0.01/0.99 = 1/99$. After one positive result the odds are $19/99$, that is, 19 ill for every 99 healthy among patients with this result, which is the probability $19/(19+99) = 0.161$, the 16 per cent of Unit 1 again. A second test of the same quality, carried out so that its errors have nothing to do with those of the first, multiplies the odds by 19 once more, to $361/99$, and the probability becomes

$$P(\text{ill} \mid +,+) = \frac{361}{361+99} = 0.785.$$

After two independent positive results the patient is ill with a probability of about 78 per cent. Figure 4 draws the same calculation as a tree. If the second test had come out negative instead, its likelihood ratio would have been $0.05/0.95 = 1/19$, the two factors would have cancelled, and the patient would be back at the prior of 1 per cent, which is what two contradictory results of equal quality should do.

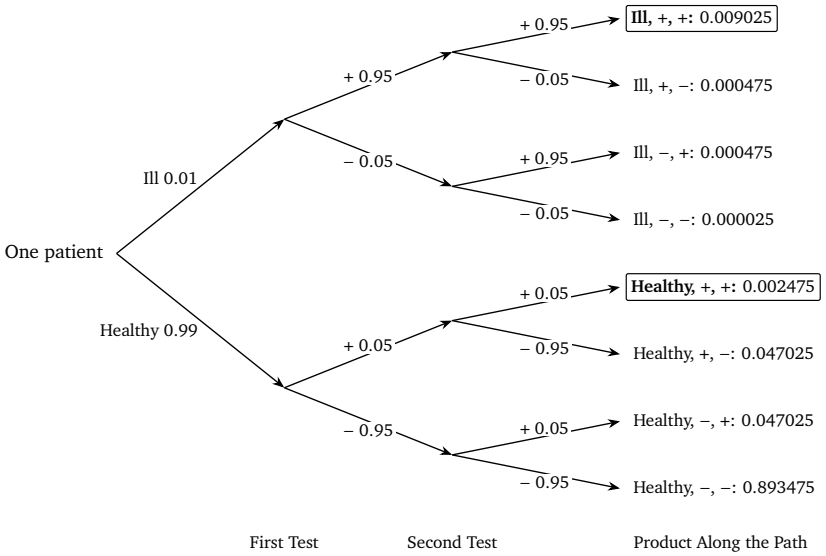


Figure 4. The two-test case as a tree. Each branch carries its probability, each leaf the product along its path, and the posterior after two positive results is the share of the "ill" leaf among the two leaves with two positives, $0.009025/(0.009025 + 0.002475) = 0.785$.

The independence in the example is a real condition, and in medicine it is often violated. If the two tests fail for the same reason, for instance because both detect the same substance and the patient's body produces it for another reason, then the second positive adds much less than a factor of 19, and the honest posterior lies between 16 and 78 per cent. A Bayesian network with the two tests as children of the disease node states the independence explicitly, and a network with an extra arrow between the tests, or an extra common cause, states its violation.

CHECK YOUR UNDERSTANDING Question 2

A doctor applies the test of this section, with 95 per cent sensitivity and 95 per cent specificity, only to patients whose symptoms already give a prior probability of 10 per cent. What is the probability of the disease after one positive result?

(A) About 16 per cent
(B) About 95 per cent
(C) About 68 per cent
(D) About 50 per cent

The solution is at the end of the unit.

EXCURSION *An essay published after its author's death*

The theorem carries the name of Thomas Bayes (c. 1701-1761), a Presbyterian minister in Tunbridge Wells in England, who never published it. After his death his friend Richard Price (1723-1791) found the manuscript among his papers, added an introduction, and read it to the Royal Society of London on 23 December 1763; it appeared in the *Philosophical Transactions* under the title "An Essay towards solving a Problem in the Doctrine of Chances". Bayes asked a question in the opposite direction from the gamblers' questions of his time. They asked how probable an outcome is given the chances; he asked what the chances are, given the outcomes observed, and his answer contains the theorem in a special case. The essay was little read. Pierre-Simon Laplace (1749-1827), who apparently did not know it, found the general rule on his own and published it in 1774 in a memoir

> on the probability of causes given events, and it was Laplace who applied it for decades, to the masses of the planets, to the births of boys and girls in Paris, and to the credibility of witnesses. The rule that adds one to every count before dividing, which Section 4 uses, is also his, which is why it is called Laplace smoothing.



Figure 5. Pierre-Simon Laplace as chancellor of the Senate, painted by Jean-Baptiste Paulin Guérin in 1838, after Laplace's death. Credit: Jean-Baptiste Paulin Guérin, Wikimedia Commons, public domain.

4 The naive Bayes spam filter

A spam filter has to decide, for each incoming email, between two values of one variable, which is its **class**, spam or ordinary mail. Its evidence is the email itself, represented in the way Unit 1 represented a sentence, as a list that says how often each word of a fixed vocabulary occurs in it. A Bayesian network for this has the class as the one parent and one child node for each word of the vocabulary, with an arrow from the class to every word and no arrows between the words, as Figure 6 shows for a vocabulary of six. The network is called the **naive Bayes classifier**, and the word "naive" refers to the missing arrows, which claim that the words are conditionally independent of each other given its class.

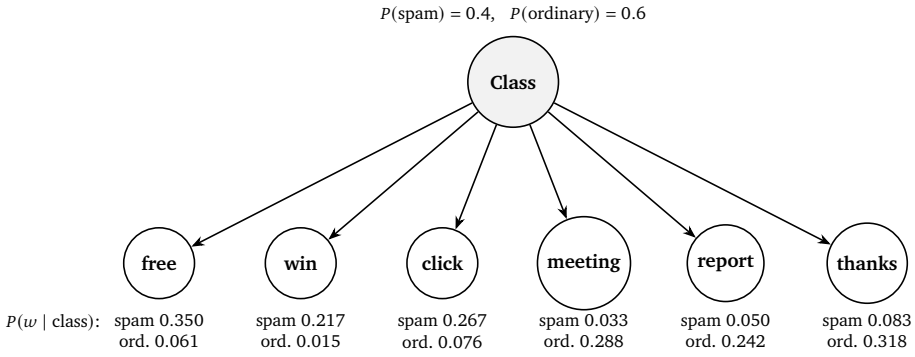


Figure 6. The naive Bayes classifier as a Bayesian network. The class is the only parent; each word of the vocabulary is a child with no other parents, which is the claim that the words are conditionally independent given the class. The numbers are the smoothed probabilities of Table 2, as computed in the worked example.

The claim is false, since in real mail the word "free" makes "offer" far more likely in the same message, whatever the class. The classifier is nevertheless one of the most used in the history of the field, for two reasons. The first is that it needs so few numbers, two per word plus the prior, whereas a table of all the words together would need one per combination, so it can be estimated from a few hundred emails. The second is that it only has to rank the two classes, and the ranking is often right even when the probabilities behind it are badly off, because the errors of the independence assumption tend to inflate both scores by

similar amounts.

4.1 Estimating the tables by counting

The tables of the classifier are not given by a doctor; they are estimated from a set of emails that someone has already sorted, which is the **training set**. The estimate is the one anyone would make. The prior $P(\text{spam})$ is the fraction of training emails that are spam, and the probability of a word given a class, $P(w \mid c)$, is the number of times w occurs in the emails of class c divided by the total number of word occurrences in that class. Counting in this way is the same as choosing the numbers that make the training set as probable as possible, which statisticians call maximum likelihood estimation; Unit 4 turns the idea into the general recipe of learning, in which the parameters of any model are chosen to make a measure of error, called the loss, as small as possible.

Counting has one defect. A word that never occurred in the ordinary mail of the training set gets the probability zero, and the product rule then assigns probability zero to any ordinary email that contains it, whatever the other words say. One such word would override all the other evidence. The remedy is **Laplace smoothing**, which adds one to every count before dividing, as if every word had been seen once in each class, so that no probability is ever zero. In formulas, with V words in the vocabulary, the estimate of the probability of the word w in class c becomes $(n_{w,c}+1)/(n_c+V)$, where $n_{w,c}$ is the count of w in class c and n_c the total count of words in that class, and the V in the denominator keeps the probabilities of all words adding up to one, because the added ones, one for each of the V words, add V to the total count.

The decision then follows from Bayes' theorem. The classifier computes, for each class, the prior multiplied by the probability of the words given the class, and chooses the larger. The probability of the evidence $P(\text{email})$ is the same for both classes and need not be computed, which is why the rule below carries arg max, the value of c for which the expression is largest, and no denominator.

KEY EQUATION The naive Bayes decision rule with Laplace smoothing

$$\hat{c} = \arg\max_c \; P(c) \prod_{j=1}^{V} P(w_j \mid c)^{x_j} = \arg\max_c \; \Big[\log P(c) + \sum_{j=1}^{V} x_j \log P(w_j \mid c) \Big], \qquad P(w_j \mid c) = \frac{n_{j,c}+1}{n_c+V}$$

An input is the vector of counts $\mathbf{x} = (x_1, \ldots, x_V)$ of the V vocabulary words; $n_{j,c}$ is how often word j occurred in the training emails of class c and n_c the total number of word occurrences in that class. A word that occurs x_j times contributes its probability x_j times, which is what the exponent says, and a word that does not occur has $x_j = 0$ and contributes a factor of one. The rule chooses the class with the largest product, or, which gives the same answer, with the largest sum of logarithms.

The logarithms are a practical necessity rather than a decoration, and Unit 1 gave the reason. An email of 200 words contributes 200 factors, each around 0.01 or smaller, and a product of 200 factors of 0.01 is 10^{-400}, which a 32-bit or even a 64-bit floating point number cannot represent and rounds to zero. The logarithm of the same product is a sum of 200 ordinary negative numbers, about −900, which any computer handles. Since the logarithm increases with the number it is applied to, the class with the larger product is also the class with the larger logarithm, and nothing is lost. To illustrate the whole method, Table 2 gives the counts of a training set of 20 emails made up for this unit, and the worked example classifies a new one.

Word	Count in 8 Spam Emails	Count in 12 Ordinary Emails
free	20	3
win	12	0
click	15	4
meeting	1	18
report	2	15
thanks	4	20
Total	54	60

Table 2. Word counts of a vocabulary of six words in a training set of 20 emails, 8 of them spam. Words outside the vocabulary are ignored. The counts were made up for this example.

WORKED EXAMPLE Classifying an email

A new email contains the words "free", "click", "meeting", and "thanks", each once, besides words that are not in the vocabulary and are ignored. The priors are $P(\text{spam}) = 8/20 = 0.4$ and $P(\text{ordinary}) = 0.6$. With Laplace smoothing and $V = 6$, the spam probabilities divide by $54+6 = 60$ and the ordinary ones by $60 + 6 = 66$:

$$P(\text{free} \mid \text{spam}) = \tfrac{21}{60} = 0.350, \qquad P(\text{free} \mid \text{ord}) = \tfrac{4}{66} = 0.0606,$$
$$P(\text{click} \mid \text{spam}) = \tfrac{16}{60} = 0.267, \qquad P(\text{click} \mid \text{ord}) = \tfrac{5}{66} = 0.0758,$$
$$P(\text{meeting} \mid \text{spam}) = \tfrac{2}{60} = 0.0333, \qquad P(\text{meeting} \mid \text{ord}) = \tfrac{19}{66} = 0.288,$$
$$P(\text{thanks} \mid \text{spam}) = \tfrac{5}{60} = 0.0833, \qquad P(\text{thanks} \mid \text{ord}) = \tfrac{21}{66} = 0.318.$$

The scores are sums of natural logarithms:

$$\begin{aligned} \text{spam:} \quad & \log 0.4 + \log 0.350 + \log 0.267 + \log 0.0333 + \log 0.0833 \\ & = -0.916 - 1.050 - 1.322 - 3.401 - 2.485 = -9.174, \\ \text{ordinary:} \quad & \log 0.6 + \log 0.0606 + \log 0.0758 + \log 0.288 + \log 0.318 \\ & = -0.511 - 2.803 - 2.580 - 1.245 - 1.145 = -8.284. \end{aligned}$$

The ordinary score is the larger, so the email is filed as ordinary mail. The difference of the two scores is the logarithm of the ratio of the two products, and that ratio is the posterior odds of spam against ordinary mail, so taking the exponential of the difference gives the odds, $e^{-9.174+8.284} = e^{-0.890} = 0.41$, which is a probability of spam of $0.41/1.41 = 0.29$, and the filter is far from certain. Without smoothing, an email with the word "win" could never be ordinary, since the count of "win" in ordinary mail is zero; with smoothing its probability given ordinary mail is $1/66$, small but not fatal.

CHECK YOUR UNDERSTANDING Question 3

In the training set of Table 2 the word "win" never occurs in ordinary mail. What would a filter without Laplace smoothing do with a new email that contains "win" once and twenty words typical of ordinary mail?

(A) File it as spam, because the ordinary product contains a factor of zero

(B) File it as ordinary, because twenty ordinary words outweigh one spam word

(C) Refuse to decide, because the two scores can no longer be compared

(D) File it as ordinary with a probability of exactly 50 per cent

The solution is at the end of the unit.

Unit 5 returns to spam with logistic regression, which models $P(c \mid \mathbf{x})$ directly instead of modelling the words given the class, and compares the two approaches; logistic regression usually wins once the training set is large, and naive Bayes usually wins while it is small.

EXCURSION *Why word counts beat hand-written rules*

Unwanted bulk email took its name from a 1970 sketch of the comedy group Monty Python in which a café serves the tinned meat Spam with everything, and by the late 1990s it was a large fraction of all mail. The first filters were lists of rules written by hand, such as "the subject line is in capitals", and senders learned each rule within weeks. In 1998 Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz at Microsoft Research showed at a workshop of the AAAI conference that a naive Bayes classifier trained on about 1,800 sorted messages caught most spam with very few false alarms, and did better still once a few non-word features, such as the time of sending, were added. The method became widespread after August 2002, when the programmer Paul Graham published an essay titled "A Plan for Spam". He reported that the single word "click" caught 79.7 per cent of his spam with 1.2 per cent of ordinary mail wrongly flagged, and that a filter combining the probabilities of the fifteen most telling words of each message missed fewer than 5 spam messages in 1,000 and flagged no ordinary mail at all. Within a year the method was in many mail programs and server-side filters. Its advantage over rules is the one this unit keeps meeting. A rule is a fact someone must write down and maintain, whereas a table of counts is estimated from the data and updated every time a message is sorted. Modern filters add many other learners to the counts, which lies beyond this unit.

5 Sequences and the Markov property

Many questions in artificial intelligence are about sequences, such as the weather on successive days, the words of a sentence, the positions of a robot, or the sounds of speech, and a network over a sequence of T days would need a node for every day and tables that grow with the length of the sequence. The way out is an assumption about time that Andrey Markov (1856-1922) introduced in 1906.

PRINCIPLE **The Markov property**

A sequence of random variables $X_1, X_2, X_3, \ldots$ has the Markov property when the future is conditionally independent of the past given the present:

$$P(X_{t+1} \mid X_t, X_{t-1}, \ldots, X_1) = P(X_{t+1} \mid X_t).$$

A sequence with this property is a **Markov chain**; its possible values are the **states**, and when the probabilities on the right do not change with t the chain is called homogeneous, which is the only kind this unit uses. The property is a modelling choice, not a law of nature, and whether it holds depends on what one puts into the state.

As a Bayesian network, a Markov chain is a row of nodes with one arrow from each day to the next and nothing else. Every node after the first has one parent and the same conditional probability table, which for k states is a k by k matrix called the **transition matrix** $\mathbf{T}$, with the entry T_{ij} giving the probability of moving from state i to state j in one step. Each row of $\mathbf{T}$ adds up to one, because the chain must go somewhere. Figure 7 draws a two-state chain for the weather, with states sunny and rainy, as a graph whose arrows carry the transition probabilities.

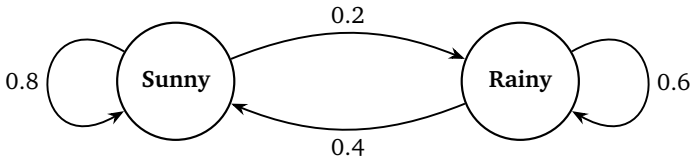


Figure 7. A two-state Markov chain for the weather. Each arrow carries the probability of the transition it shows, and the arrows leaving a state add up to one. The probabilities are set by hand for this unit.

The chain is run forward with a single multiplication. If the probabilities of the states on day t are written as a row vector $\mathbf{p}_t = (P(\text{sunny}), P(\text{rainy}))$, then by the sum and product rules the probabilities on the next day are

$$\mathbf{p}_{t+1} = \mathbf{p}_t\,\mathbf{T}, \qquad \text{that is} \qquad P(X_{t+1} = j) = \sum_i P(X_t = i)\,T_{ij}, \tag{1}$$

which is the matrix-vector product of Unit 1 with the vector written as a row on the left, because the rows of $\mathbf{T}$ are the starting states.

KEY EQUATION Running a Markov chain

$$\mathbf{p}_{t+1} = \mathbf{p}_t\,\mathbf{T}, \qquad \mathbf{p}_t = \mathbf{p}_0\,\mathbf{T}^t$$

$\mathbf{p}_t$ is the row vector of state probabilities after t steps and $\mathbf{T}$ the transition matrix, whose entry T_{ij} is the probability of a step from state i to state j; $\mathbf{T}^t$ is the matrix multiplied by itself t times, one factor for each step. A distribution $\boldsymbol{\pi}$ with $\boldsymbol{\pi}\,\mathbf{T} = \boldsymbol{\pi}$ is a stationary distribution of the chain; for a chain with finitely many states in which every state can reach every other and no cycle forces a fixed rhythm, there is exactly one, and $\mathbf{p}_t$ approaches it from any start.

WORKED EXAMPLE Three days of weather and the long run

The chain of Figure 7 has the transition matrix

$$\mathbf{T} = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix},$$

with the rows and columns in the order sunny, rainy. Today, day 0, is sunny for certain, so $\mathbf{p}_0 = (1, 0)$. The first row of $\mathbf{T}$ gives tomorrow, $\mathbf{p}_1 = (0.8, 0.2)$. The day after,

$$\mathbf{p}_2 = (0.8 \times 0.8 + 0.2 \times 0.4,\ 0.8 \times 0.2 + 0.2 \times 0.6) = (0.72, 0.28),$$

and the third day gives $\mathbf{p}_3 = (0.72 \times 0.8 + 0.28 \times 0.4,\ 0.72 \times 0.2 + 0.28 \times 0.6) = (0.688, 0.312)$. The probability of rain three days after a sunny day is about 31 per cent. Continuing, day 5 gives $(0.670, 0.330)$ and day 7 gives $(0.667, 0.333)$. The derivation below shows that the chain settles at $\boldsymbol{\pi} = (2/3, 1/3)$, so that in the long run two days in three are sunny, whatever the weather was on day 0.

DERIVATION The stationary distribution of a two-state chain

Write the stationary distribution as $\boldsymbol{\pi} = (\pi_1, \pi_2)$ with $\pi_1 + \pi_2 = 1$. For a chain that leaves state 1 with probability a and leaves state 2 with probability b, the condition $\boldsymbol{\pi}\,\mathbf{T} = \boldsymbol{\pi}$ for the first component reads $\pi_1(1-a) + \pi_2\,b = \pi_1$, which simplifies to $\pi_1 a = \pi_2 b$. In words, in the long run the probability flowing out of state 1 equals the probability flowing in. With $\pi_2 = 1 - \pi_1$ this gives

$$\pi_1 = \frac{b}{a+b}, \qquad \pi_2 = \frac{a}{a+b}.$$

For the weather chain $a = 0.2$ and $b = 0.4$, so $\pi_1 = 0.4/0.6 = 2/3$ and $\pi_2 = 1/3$. The chain forgets its start at a rate set by $1 - a - b = 0.4$. Subtracting the stationary condition $\pi_1(1-a) + \pi_2 b = \pi_1$ from the step $p_{t+1}(1) = p_t(1)\,(1-a) + p_t(2)\,b$ of Equation (1) shows that the difference $p_t(1) - \pi_1$ is multiplied by $1 - a - b$ at every step, so each day the distance to $\boldsymbol{\pi}$ shrinks by that factor, which is why $\mathbf{p}_3$ in the example is already within about 0.02 of the limit.

Markov first applied his chains to letters rather than weather. In a lecture to the Imperial Academy of Sciences in St Petersburg on 23 January 1913, he reported that he had gone through the first 20,000 letters of Pushkin's verse novel *Eugene Onegin*, classified each as a vowel or a consonant, and counted the pairs. About 43 per cent of the letters were vowels, but a vowel followed a vowel only about 13 per cent of the time and followed a consonant about 66 per cent of the time, so the letters were not independent and a two-state chain described them well. A chain whose states are words rather than letters was for decades the standard model of language in speech recognition, and the probability that a modern language model assigns to the next word is a descendant of that idea, which lies beyond this unit.

CHECK YOUR UNDERSTANDING Question 4

A forecaster predicts tomorrow's weather from the weather of the last three days. Can this forecaster be written as a Markov chain?

(A) No, because the Markov property lets only the previous day influence the next one

(B) No, because a chain whose state covers several days can no longer be homogeneous

(C) Yes, but only if the weather of the three days is conditionally independent given today's

(D) Yes, by taking the last three days together as one state, which gives 8 states in all

The solution is at the end of the unit.

5.1 Hidden Markov models

In the weather chain the state is observed every day. More often the state of interest cannot be seen directly, and what is seen is some noisy consequence of it. A speech recognizer, for example, does not hear the words, only the sound. A **hidden Markov model** is a Markov chain whose states X_t are hidden and which produces on each day an **observation** E_t with a probability $P(E_t \mid X_t)$ that depends on the state of that day alone. As a Bayesian network it is the chain of the last section with one child hanging from each state node, and its tables are the transition matrix, the observation probabilities, and the distribution of the first state. The theory of these models was worked out in the 1960s by Leonard Baum (1931-2017) and colleagues at the Institute for Defense Analyses in Princeton, and from the 1970s to the 2010s they were the core of most speech recognizers.

The simplest question put to such a model is the probability of the hidden state today given all observations so far, and the answer is computed day by day by the **forward algorithm**, which repeats two steps. The first predicts the state of the next day from the current belief with Equation (1), and the second updates the prediction with the new observation by Bayes' theorem, which multiplies each state's probability by the probability of the observation given that state and then divides by the total.

WORKED EXAMPLE Rain inferred from umbrellas

A friend lives in the town of the weather chain and messages every morning whether they are taking an umbrella, which they do on 90 per cent of rainy days and on 10 per cent of sunny days. The day before the messages began was sunny, so on the first morning the prediction is $(0.8, 0.2)$, and the friend takes an umbrella. The update multiplies each state by the probability of an umbrella in that state, 0.1 on a sunny day and 0.9 on a rainy day, $(0.8 \times 0.1,\ 0.2 \times 0.9) = (0.08, 0.18)$, and divides by the total of 0.26, which gives $(0.308, 0.692)$. On the second morning the prediction from this belief

is $(0.308 \times 0.8 + 0.692 \times 0.4,\ 0.308 \times 0.2 + 0.692 \times 0.6) = (0.523, 0.477)$, and the friend takes an umbrella again. The update gives $(0.523 \times 0.1,\ 0.477 \times 0.9) = (0.052, 0.429)$, and after dividing by 0.481 the belief is $(0.109, 0.891)$. After two umbrellas the probability of rain on the second day is about 89 per cent. The computation needs only the belief of the day before and the new observation, however long the sequence has run.

Finding the single most probable sequence of hidden states for a whole series of observations, which a speech recognizer needs, is done by the Viterbi algorithm, and estimating the tables from observations alone by the Baum-Welch algorithm; both lie beyond this unit, as do sequence models whose state is a vector learned from data rather than a label chosen by hand.

6 Probabilistic reasoning today

The three ingredients of this unit, a factorized joint distribution, evidence entered into it, and Bayes' theorem to turn the result round, are at work in systems in daily use. Bayesian networks have supported medical diagnosis since the late 1980s, when networks of hundreds and later thousands of nodes were built for pathology and internal medicine, and spam filters of the kind of Section 4 still run inside every large mail service. What has changed since the 1990s is where the tables come from. Here they were set by hand or counted from a small training set; Unit 4 replaces the counting by a general procedure that chooses the parameters of any model to make a loss as small as possible, and from Unit 6 on the models learned this way are neural networks, whose output probabilities are still read with the rules of this unit.

KEY POINTS

- The joint distribution answers every question by marginalization and conditioning, but for n yes-or-no variables it has $2^n - 1$ free numbers, too many beyond a few dozen variables.
- A Bayesian network writes the joint distribution as $P(x_1, \ldots, x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$, so that each variable needs only a table over its parents; the missing arrows are claims of conditional independence.
- Inference by enumeration fixes the evidence, sums the product of the tables over the hidden variables, and divides by the total; it is exact but grows exponentially with the number of hidden variables.
- Bayes' theorem, $P(H \mid E) = P(E \mid H)\,P(H)/P(E)$, turns a likelihood into a posterior, and the prior matters as much as the evidence, as the 16 per cent of the medical test shows; independent evidence multiplies the odds.
- The naive Bayes classifier is a network with the class as the only parent; its tables are counts from a training set with one added to each, and it decides by comparing sums of logarithms.
- A Markov chain assumes that the future depends on the past only through the present and runs forward by $\mathbf{p}_{t+1} = \mathbf{p}_t\,\mathbf{T}$; a hidden Markov model adds a noisy observation of each state.

Solutions to the questions in the text: 1-B, 2-C, 3-A, 4-D. The online version of this unit explains each answer.