1 Probabilities

Probability is a tricky word—usually meaning the likelyhood of something occuring—or how frequent something is. Obviously, if something happens frequently, then its probability of happening is high.

1.1 Basics

Probabilities always involve three things: a random variable X, an alphabet $\boldsymbol{\mathcal{A}_x}$, and the corresponding probabilities $\boldsymbol{\mathcal{P}_x}$. In this setup, X takes on values $x \in \boldsymbol{\mathcal{A}_x}$ with probability $\boldsymbol{\mathcal{P}_x}$. Probabilities of subsets are just sums of the individual elements of the subsets; if $\boldsymbol{T} \subseteq \boldsymbol{\mathcal{A}_x}$, then

$$P(\boldsymbol{T}) = P(x \in \boldsymbol{T}) = \sum_{a_i} P(x = a_i)$$

When more than one variable are involved, we have a *joint probability*. For two variables, we may write $P(x, y)$. For five, we may write $P(a, b, c, d, e)$.

For example, for a single die[1], the alphabet is $\{1, 2, 3, 4, 5, 6\}$, since any single throw can land on some number 1 through 6. Consider throwing *two* die, the outcomes may be:

$$\begin{aligned}
2 &= \{1,1\} \\
3 &= \{1,2\} \text{ or } \{2,1\} \\
4 &= \{1,3\} \text{ or } \{2,2\} \text{ or } \{3,1\} \\
5 &= \{1,4\} \text{ or } \{2,3\} \text{ or } \{3,2\} \text{ or } \{4,1\} \\
6 &= \{1,5\} \text{ or } \{2,4\} \text{ or } \{3,3\} \text{ or } \{4,2\} \text{ or } \{5,1\} \\
7 &= \{1,6\} \text{ or } \{2,5\} \text{ or } \{3,4\} \text{ or } \{4,3\} \text{ or } \{5,2\} \text{ or } \{6,1\} \\
8 &= \{2,6\} \text{ or } \{3,5\} \text{ or } \{4,4\} \text{ or } \{5,3\} \text{ or } \{6,2\} \\
9 &= \{3,6\} \text{ or } \{4,5\} \text{ or } \{5,4\} \text{ or } \{6,3\} \\
10 &= \{4,6\} \text{ or } \{5,5\} \text{ or } \{6,4\} \\
11 &= \{5,6\} \text{ or } \{6,5\} \\
12 &= \{6,6\}
\end{aligned}$$

That's 36 outcomes, each having 1 in 36 chance of occuring. For example, if you throw two dice, your chances of getting a "2", or $P(2)$ are $1/36$. Your chances of getting "11", or $P(11)$ are $2/36$ (since there are two subsets that add up to 11, namely, $\{5, 6\}$ and $\{6, 5\}$). What about $P(7)$? We can get that any number of ways:

$$\{1,6\} \text{ or } \{2,5\} \text{ or } \{3,4\} \text{ or } \{4,3\} \text{ or } \{5,2\} \text{ or } \{6,1\}$$

There are six ways of getting a "7". Each one of those has a $1/36$ chance of coming up, thus $P(7) = 6/36$.

What are the chances of us throwing a "7" where one of the die comes up as "1"? Here are the outcomes when at least one die is a 1:

$$\{1,1\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{1,6\}, \{2,1\}, \{3,1\}, \{4,1\}, \{5,1\}, \{6,1\}$$

[1]Small cube with a number on each side.

That makes 11/36. We already know that chance of getting a "7" is 6/36. To add the probabilities gets us:

$$P(\text{at least one die is 1}) + P(7) = 11/36 + 6/36 = 17/36$$

But we counted some of them twice! $\{1, 6\}$ and $\{6, 1\}$ show up for both $P(\text{at least one die is 1})$ and $P(7)$, so we must subtract them... So the end result is:

$$P(\text{at least one die is 1}) + P(7) - P(\{1, 6\} \text{ or } \{6, 1\}) = 11/36 + 6/36 - 2/36 = 15/36$$

To put that into set nation:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

We obtain a *marginal probability* $P(x)$ from a joint probability $P(x, y)$ via summation:

$$P(x) = \sum_{y \in \boldsymbol{A_y}} P(x, y)$$

This is often called *marginalization*, or *summing out*. For example, we can find the probabilities for a single die by summing out the 2nd die from example above.

Events tend to occur one after the other. Probability of x given y is called *conditional probability*, and is written $P(x|y)$. This is just a ratio:

$$P(x|y) = \frac{P(x, y)}{P(y)}$$

Rewriting conditional probability gives us the *product rule*:

$$P(x, y) = P(x|y)P(y) \qquad \text{or} \qquad P(x, y) = P(y|x)P(x)$$

If x and y are independent (have no influence on each other's occurrence), the product rule becomes:

$$P(x, y) = P(x)P(y)$$

A practical note on the product rule is that often we don't need to compute actual products of probabilities, but can work with sums of logarithms.

A variation on the product rule and marginalization gives us *conditioning*:

$$P(x) = \sum_{y \in \boldsymbol{A_y}} P(x|y)P(y)$$

Simiarly, we can get a *joint probability* from conditional probabilities via the *chain rule*:

$$P(x) = \prod_{i=1}^{n} P(x_i|x_1, ..., x_{i-1})$$

In other words (writing out the above $\prod$ loop), we get:

$$\begin{aligned} P(a, b) &= P(a|b)P(b) \\ P(a, b, c) &= P(a|b, c)P(b|c)P(c) \\ P(a, b, c, d) &= P(a|b, c, d)P(b|c, d)P(c|d)P(d) \end{aligned}$$

and so on.

1.2 Bayes' theorem

Thomas Bayes (1702-1761) gave rise to a new form of statistical reasoning—the inversion of probabilities. We can view it as

$$Posterior = Prior \times Likelihood$$

where *Posterior* is the probability that the *hypothesis* is true given the evidence. *Prior* is the probability that the hypothesis was true *before* the evidense (ie: an assumption). *Likelihood* is the probability of obtaining the observed evidense given that the hypothesis is true.

Bayes' rule is derived from the *product rule*, by noting:

$$P(x|y)P(y) = P(y|x)P(x) \qquad \text{which leads to:} \qquad P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

The key thing to notice is that Bayes theorem is recursive, which is precisely what allows us to use this rule for learning. For example, let's say the task is to calculate probability of event A every day. On day one, we take a wild guess of what the probability might be (since we have no history). On day 2 (and all subsequent days) we use the prior day's values—allowing the probability of event A to get more accurate with time. In other words, we learn the probability of A!

2 Naive Bayes Classifier

We can use the Bayes rule to do document classification—commonly used to classify emails into spam/nospam categories. For this to work, we need (either assumed, or calculated) prior probabilities of certain word occuring in a certain document category, ie:

$$P(w_i|C)$$

where w_i is some word, and C is some document category (ie: spam, nospam, etc.). The probability of a given document D given a certain document category is:

$$P(D|C) = \Pi_i P(w_i|C)$$

note that none of the probabilities can be zero, otherwise the whole thing is zero. In practice, this is usually accomplished by specifying a very small number as the minimum probability (even if the word doesn't exist in a particular category).

Now we do the Bayes thing:

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

Both $P(D|C)$ and $P(C)$ can be easily estimated from the training data (just count words by category). We never have to estimate $P(D)$ as it's only normalizing the results (making probabilities sum to 1)—which we don't need to determine which category is more likely.

2.1 Markov Processes & Chains

A *Markov process* $M = (\mathbf{\Omega}, \mathbf{\Gamma}, k)$ is a random variable, where $\mathbf{\Omega}$ is the set of all possible values, or state space, $\mathbf{\Gamma}$ is a transition matrix, Γ_{ij} is the probability of going from state i to state j, and k is the starting state.

A *Markov chain* is a sequence of states, such as: $X_1, X_2, X_3, \ldots$, generated by a Markov process. Markov chains have the obvious *Markov property* that the future and past states are independent given the current state (the next state only depends on the current state), that is:

$$P(X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x | X_n = x_n)$$

We can find the probability of being in any given state after N steps simply by iterating N times. This turns out to be relatively easy: $P = k\mathbf{\Gamma}^N$ where $\mathbf{\Gamma}$ is raised to Nth power, and is multiplied by k, the starting state probability distribution (distribution with all states set to zero except one, the starting state).

Depending on some properties[2] of the chain, it may reach a *stationary distribution* (after iterating for a while), that is independent of starting values.

2.2 Hidden Markov Model

A Hidden Markov Model (HMM) is similar to a Markov Chain, except the states are not observable (i.e.: they are *hidden*). What is observable are 'output symbols', which are generated (via a certain probability) by the hidden states. A sequence of hidden state transitions generates an observed output sequence.

Imagine you're a guard in an underground facility, where you don't know what the weather is, but you can see folks carrying umbrellas. However, just because you observe someone carrying an umbrella does not mean it is raining outside.

The general problem is to find the most likely state sequence (which cannot be observed) given the observed output sequence. This problem is solved via the Viterbi algorithm.

2.3 Bayesian Networks

Given a probability distribution, say $P(a, b, c, d, e, f, g)$ we can calculate probability of anything we feel is useful. For example, if we wanted to know what is the probability of $P(a, c, g)$ we can just sum over the other variables. Similarly if some variables have definite values, e.g.: $P(a = true, c, f = false, g)$.

Note that the a, b, c, d etc above can be all the everyday things. For example, a may be "fire alarm goes off", b may be "someone calls the fire department", c may be "all phones are dead", d may be "alarm clock goes off" and d may be "alarm clock gets confused with fire alarm", and so on. Useful stuff!

[2]Irreducible: one can go from any state to any other state, even if it takes more than one step. Aperiodic: the chain is not forced into cycles.

In other words, having a probability distribution and the ability to extract information from it is incredibly useful. The major problem is that marginalization (summing out) is terribly expensive to do: it is an exponential operation. To go from $P(a, b, c)$ to $P(a, b)$ we need to sum over *all* the possible values of c—and this adds up to impractical very quickly.

Recall joint probability distributions rewritten as conditional probabilities:

$$\begin{aligned} P(a, b) &= P(a|b)P(b) \\ P(a, b, c) &= P(a|b, c)P(b|c)P(c) \end{aligned}$$

Now imagine that we knew that a was not depenent on b, c. For example, "fire alarm goes off" is independent of "alarm clock goes off". The above then could be rewritten as:

$$\begin{aligned} P(a, b) &= P(a)P(b) \\ P(a, b, c) &= P(a)P(b|c)P(c) \end{aligned}$$

This is suddenly much simpler to deal with—in terms of maginalization.

Bayesian Networks then is a method to write a probability distribution that explicitly specifies conditional *independence*. In a joint probability, everything is assumed to be dependent on everything else—in Bayesian networks we are explicitly saying that some things have nothing to do with each other.

Practically speaking, in the worst case, dealing with Bayesian networks is still exponential. We can remove some dependencies, but the ones that are left are still going to cause exponential calculations.

Folks have came up with various sampling algorithms to poke at the exponential search space and make predictions—without actually going through the entire search space. If curious, search for message passing algorithms.

The above just deals with *evaluating* a bayesian network, but not with actually coming up with one. The simplest way of coming up with one is to ask an expert to draw it (and specify probabilities). This is similar to having an expert specify rules of an expert system.

As a fallback, the expert can specify the structure, and the probabilities can be learned from data (perhaps using bayes rule for learning, or maybe just finding aggregates in the dataset).

If we don't have an expert to specify a bayesian network, and we need to learn both the structure and probabilities from data, the task becomes... hard. There are many folks doing research into this, but so far, for any interesting size problem, the task is impractical. The main problem is that it is very hard to claim independence with only a finite number of input samples.

There does appear to be a stupidly silly workaround: if the training dataset is big, say 100 gigabytes, with say 1000 variables (this is currently *very* impractical by the best of the best bayesian network learning approaches), why not... instead of learning the bayesian network and then calculate probabilities from it... just run through the dataset? We can answer ANY statistics question on this dataset by scanning through it once and maintaining aggregates. If we need to answer another question, just scan through the dataset again—sure beats spending an exponential amount of time learning a bayesian network, and then

spending an equally exponential amount of time evaluating (even via sampling) the learned network.

2.4 Monte Carlo Methods

Monte Carlo integration is an algorithm that relies on random sampling to compute an integral, possibly of a very complicated multidimensional function. This is accomplished by generating samples within the integral and then averaging them together (weighted average, via probability). The major problem, of course, is how to sample an arbitrary probability distribution. Given any integral:

$$\int_a^b h(x)dx$$

If we can decompose $h(x)$ into function $f(x)$ and a probability distribution $p(x)$ defined over an interval (a, b), then we have:

$$\int_a^b h(x)dx = \int_a^b f(x)p(x)dx = E_{p(x)}\left[f(x)\right] \simeq \frac{1}{n}\sum_{i=1}^{N} f(x_i)$$

As a simple example, we can use this technique to estimate area under a quarter circle, and with that, get an estimate for value of π:

$$\pi \approx \frac{4}{N}\sum_{i=1}^{N}\begin{cases} 1 & \text{if } x_{random}^2 + y_{random}^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

where x_{random} and y_{random} are random numbers from 0 to 1, generated on every loop. The bigger N is (more samples in the integral), the more accurate the estimate for π. Now, consider marginalization:

$$P(x) = \sum_{y \in \mathcal{A}_y} P(x|y)P(y)$$

What if instead of going through all the values of $\mathcal{A}_y$, we randomly sample them? This is the gist of a Monte Carlo method.

Often, the distribution we wish to sample is very complex and has many dimensions. The *Markov Chain Monte Carlo* method [**?**, **?**] constructs a Markov Chain (Section **??**) whose stationary distribution is the distribution we wish to sample. The sampling process then just has to walk the chain for a long enough time to produce accurate samples.

A popular MCMC algorithm is a special case of Metropolis-Hastings, called *Gibbs sampling*. If we have random variables X, and Y, and wish to sample $f(x)$, but only have $f(x|y)$ and $f(y|x)$, we can generate a *Gibbs sequence* via:

$$\begin{array}{lll} X_j & \Leftarrow_{sample} & f(x|Y_j = y_j) \\ Y_{j+1} & \Leftarrow_{sample} & f(y|X_j = x_j) \end{array}$$

where y_0 is specified (or guessed); remainder of the sequence:

$$X_0, Y_1, X_1, Y_2, X_2, \ldots$$

is generated by iteratively applying the above rules. Such a rule can easily be generalized to more than two random variables:

$$\begin{array}{lll} X_j & \Leftarrow_{sample} & f(x|Y_j = y_j, Z_j = z_j) \\ Y_{j+1} & \Leftarrow_{sample} & f(y|X_j = x_j, Z_j = z_j) \\ Z_{j+1} & \Leftarrow_{sample} & f(z|X_j = x_j, Y_{j+1} = y_{j+1}) \end{array}$$

In the above case, the initial values for y_0 and z_0 are specified, etc.

Gibbs sampling sits at the core of most efficient (practical) methods involving Bayesian inference. Applications for the above are mostly simulations to compute some value that doesn't have a closed form, such as valuation of equity indexed annuities [**?**], or computing the credit score [**?**].