Stochastic Generative Models

The idea of a *stochastic generative model* is a very common framework in which to formulate, or understand, machine learning methods.

The general idea consists of 3 steps:

Modelling: You assume — usually an implausible assumption, taken literally — that both the training data and the new examples (the test data) are samples generated independently by some stochastic (probabilistic) process Q within some space Φ of possible processes. Often the possible processes in Φ are all structurally the same and vary only in numerical parameters, but that is not always the case.

Sometimes the processes in Φ are considered all equally likely; sometimes there is a prior probability distribution $P(Q)$ over Φ.

In principle, modelling is carried out before looking at the specific training data at all — just on the basis of the structure of the data and some general sense of its characteristics. In practice, that is not always or even usually the case.

There are no fixed rules for modelling and certainly no formulas. You have to use your understanding of the problem and the data and, more often than not, you also have to do empirical testing to find something that works.

Training: You find the process $Q \in \Phi$ which is most probable given the training data T. That is, you are trying to find

$Q = \text{argmax}_{Z \in \Phi} P(Z|T)$.

Using Bayes' law, that is equal to

$Q = \text{argmax}_{Z \in \Phi} P(T|Z) \cdot P(Z)/P(T) = \text{argmax}_{Z \in \Phi} P(T|Z) \cdot P(Z)$

since the denominator $P(T)$ does not depend on Z.

If the process Z are all equally probable, this simplifies further to $\text{argmax}_{Z \in \Phi} P(T|Z)$

This is known in statistics as *maximum likelihood estimation*: find the process Z that maximizes the probability that Z would generate the training data.

Inference: Having chosen the generative process Q, you now use it in solving whatever task it is that you have to solve. Specifically, if you are doing supervised classification learning — predicting the value of a classification attribute C from predictive attributes $A_1 \ldots A_k$ — then you choose the value v of C that Q says is most probable given the values of the predictive attributes. That is, you want to find

$\text{argmax}_v P(X.C = v \mid X.A_1 = u_1, X.A_2 = u_2, \ldots X.A_k = u_k)$

Really simple case: Flipping a coin

Suppose that you have a weighted coin. You flip it 10 times and get the sequence "HHTHHHTHHH". What should you predict for the next flip? Obviously, heads. But let's see how this works out in terms of theory of stochastic generative models.

Modelling The class of models *hi* will be the set of all Bernoulli processes with parameter p: $\Phi = \{B_p \mid : p \in 0 \leq p \leq 1\}$.

Training: For any particular model $B_p \in \Phi$, the probability of this specific outcome — HHTHHHTHHHH — is $p \cdot p \cdot (1-p) \cdot p \cdot p \cdot p \cdot (1-p) \cdot p \cdot p \cdot p = p^8 \cdot (1-p)^2$.

We wish to find the value of p that maximizes the above expression. Let us write $f(p) = p^8(1-p)^2$.

We can find the maximum by taking the derivative and setting it equal to 0, giving us the equation

$$0 = f'(p) = 8p^7(1-p)^2 - 2p^8(1-p) = (p^7(1-p)) \cdot [8(1-p) - 2p]$$

So, excluding the values $p - 0$ and $p = 1$ which are clearly not maxima, we have $8 - 10p = 0$ so $p = 0.8$. We therefore adopt a mode in which we are flipping a coin that comes up head with probability 0.8.

Inference: To predict the next coin, we use the chosen model. In the chosen model, the probability of heads is 0.8 and the probability of tails is 0.2, so we predict heads with probability 0.8.

In general suppose that we've flipped a coin and gotten a heads and b tails, in some specific order. If the coin is weighted to come up heads with probability p, then the probability of that outcome is $f(p) = p^a(1-p)^b$. Again, the maximum is attained at a point where the derivative is 0, so we have

$$0 = f'(p) = ap^{a-1}(1-p)^b - bp^a(1-p)^{b-1} = p^{a-1}(1-p)^{b-1} \cdot [a(1-p) - bp]$$

So we have $a - (a+b)p = 0$ so $p = a/(a+b)$. So the model we choose is the one where the weight is the fraction of heads.

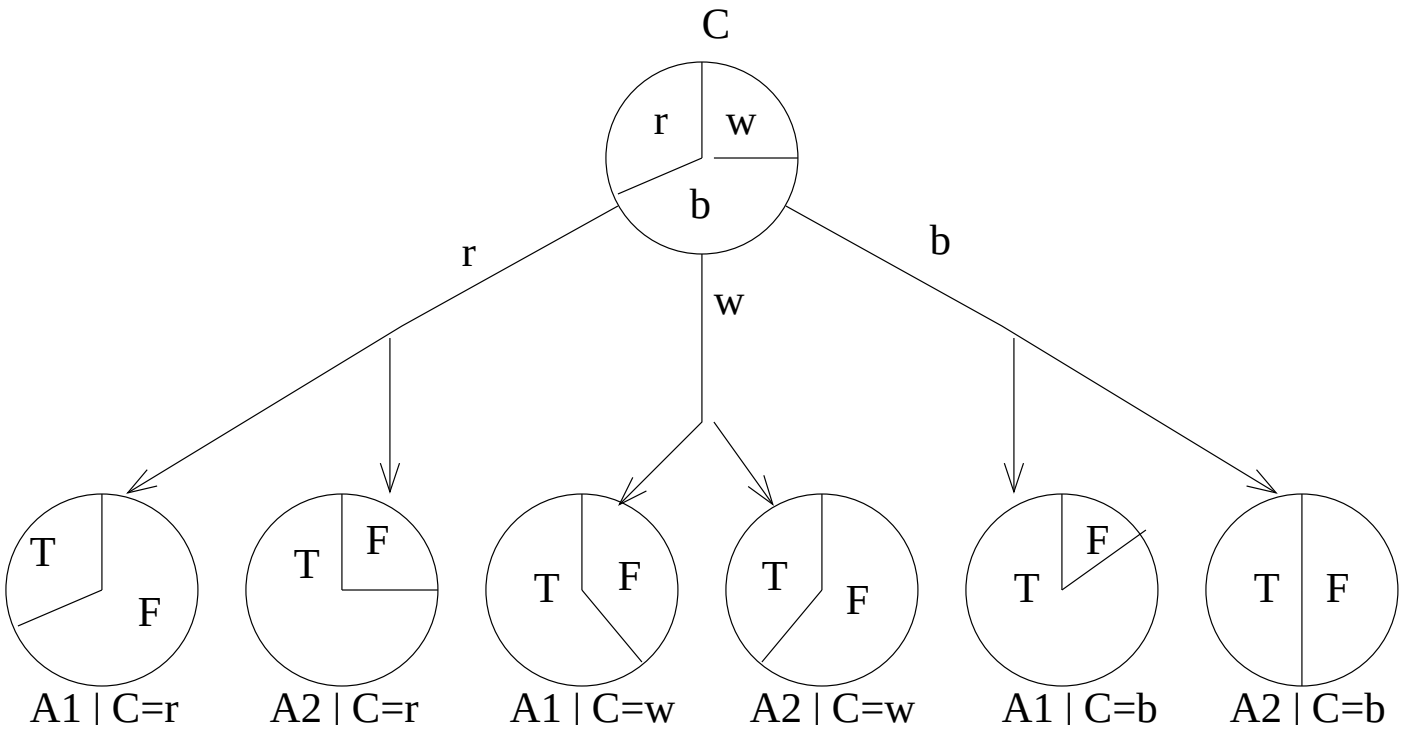
Now, imagine that you've flipped the coin 150 times and it came up heads the first 100 times and tails the last 50 times. You might well suspect that something fishy is going on here; the coin somehow changed its weighting after the 100th flip, or you were originally flipping a two-headed coin and somehow that got replaced by a two-tailed flip. Therefore, it might seem reasonable to predict tails on the next flip, since there's not much reason to suppose that such a swap will occur precisely now. However, that has to be addressed by going back to the *modelling* step and changing the class of models to include the possibility of coins changing their characteristics. Within the class of models that we defined at the start, there is no way even to consider that possibility.

Naïve Bayes as a stochastic generative model

A comparatively simple and typical example of a machine learning that can be viewed a stochastic generative model is Naïve Bayes.

For concreteness, suppose that we have a supervised classification problem in which the classification attributed C has three values r, w, b and there are two Boolean predictive attributes A_1 and A_2.

The class Φ includes stochastic processes of the following structure. To generate an instance, you first spin an uneven roulette wheel (or throw a random dart) to choose the value of C. Then, depending on the value you get for C, you go to two additional roulette wheels, one to determine the value of A_1 and the other to determine the value of A_2. There are thus six roulette wheels for the second stage: One for each value of C and for each of the attributes.



Let us write p_r, p_w, p_b for the probability of the three outcomes on the C wheel; and for $v \in \{r, b, w\}$; $i \in \{1, 2\}$; $x \in \{t, f\}$ let $p_{v,i,x}$ be the probability that wheel $A_i|C = v$ has outcome x. For example $p_{r,2,t}$ is the probability that wheel $A_2|C = r$ comes up t. The learning task, then, is to find the values of the probabilities p_v and $p_{v,i,x}$ that maximize the overall probability of the training set; and the inference task is, given those probabilities and the values of the predictive attribute, to predict the value of the classification attibute.

Inference in Naïve Bayes

We'll start with inference, which is simpler. Suppose that we've fixed on a particular process Q, that is, we've selected values for p_c and $p_{c,a,x}$. Then clearly
$P(A_1 = x_1, A_2 = x_2, C = v) = P(A_1 = x_1 \mid C = v) \cdot P(A_2 = x_2 \mid C = v) \cdot P(C = v)$
since the second level of roulette wheels depends on the color, and the two roulette wheels for a particular color are independent of each other.

So $P(C = v \mid A_1 = x_1, A_2 = x_2) =$
$P(A_1 = x_1 \mid C = v) \cdot P(A_2 = x_2 \mid C = v) \cdot P(C = v) / \ P(A_1 = x_1, A_2 = x_2) \ \propto$
(ignoring the denominator, which doesn't depend on v)
$P(A_1 = x_1 \mid C = v) \cdot P(A_2 = x_2 \mid C = v) \cdot P(C = v)$
and you choose the value of v that maximizes that expression. Which is exactly what Naive Bayes says to do. This is hardly surprising: we've constructed a stochastic process that satisfies the same independence assumptions (the predictive attributes are conditionally independent given the classification attribute.)

Training in Naïve Bayes

In the generative stochastic process framework, the problem of training is formulated as follows. Suppose that you have a training set T that looks something like this:

ID #	A_1	A_2	C
1	T	F	r
2	F	F	b
3	T	F	w
...			
999	T	T	w
1000	F	F	r

We assume that this data was generated by some process Q in Φ. The question is, which process Q is most likely, given T? That is, what is $\text{argmax}_{Q\in\Phi}P(Q|T)$? As we showed above, assuming that all processes in Q are equally likely, this is the same as finding $\text{argmax}_Q P(T|Q)$. So we have to calculate the probability of a particular process $Q \in \Phi$ generating this specific training set, if you run Q 1000 times. Of course, that probability is tiny beyond words, but it's much larger for some Q's than others.

To start: If you take one particular row — say row 3, "T, F, w". The probability that a process Q with parameters $p_v, p_{v,i,x}$ generates that row on its third trial is $p_w \cdot p_{w,1,t} \cdot p_{w,2,f}$. In general, if you have a row $A_1 = x, A_2 = y, C = v$, then the probability that Q will generate that row in a particular trial is $p_v \cdot p_{v,1,x} \cdot p_{v,2,y}$.

Since, for any particular Q, all the rows are independent, the probability of the entire table is the product of the probabilities of the rows. Let n_v be the total number of rows $X_c = v$; and let $n_{v,i,x}$ be the total number of rows in which $X.C = v$ and $X.A_i = x$. Now if you multiply the expression $p_v \cdot p_{v,1,x} \cdot p_{v,2,y}$ across all the rows, then the term p_r will appear n_r times in the product, and likewise p_w and p_b and the term $p_{r,1,w}$ will appear $n_{r,1,w}$ times in the product, and likewise all the other conditional probabilities. So we can write down the probability of the table as a whole:

$$P(T|Q) = p_r^{n_r} \cdot p_w^{n_w} \cdot p_b^{n_b} \cdot p_{r,1,t}^{n_{r,1,t}} \cdot p_{r,1,f}^{n_{r,1,f}} \cdot p_{r,2,t}^{n_{r,2,t}} \cdot p_{r,2,f}^{n_{r,2,f}} \cdot p_{w,1,t}^{n_{w,1,t}} \cdot p_{w,1,f}^{n_{w,1,f}} \cdot p_{w,2,t}^{n_{w,2,t}} \cdot p_{w,2,f}^{n_{w,2,f}} \cdot p_{b,1,t}^{n_{b,1,t}} \cdot p_{b,1,f}^{n_{b,1,f}} \cdot p_{b,2,t}^{n_{b,2,t}} \cdot p_{b,2,f}^{n_{b,2,f}}$$

or more compactly

$$P(T|Q) = \prod_{v\in\{r,w,b\}} p_v^{n_v} \cdot \prod_{v\in\{r,w,b\},i\in\{1,2\},x\in\{t,f\}} p_{v,i,x}^{n_{v,i,x}}$$

The n values are given by the table; our task is to find the p values that make this expression as large as possible. That may not look like much fun, but it's actually much easier than it appears.

Your first thought might be, "Let's just choose all of the p's to be 1," but that's not allowed. The probabilities of the three colors have to add up to 1; that is $p_r + p_w + p_b = 1$. Likewise, for any particular color v and attribute i, the probabilities of True and False must add up to 1; that is, $p_{v,i,t} + p_{v,i,f} = 1$. So we have to find the largest value of this product subject to these constraints and the constraint that all the p lie between 0 and 1. We can enforce the constraints by replacing p_b by $1 - p_r - p_w$ and by replacing $p_{v,i.f}$ by $1 - p_{v,i,t}$. That gives us the formula

$$P(T|Q) = p_r^{n_r} \cdot p_w^{n_w} \cdot (1 - p_r - p_w)^{n_b} \cdot \prod_{v \in \{r,w,b\}, i \in \{1,2\}} p_{v,i,t}^{n_{v,i,t}} \cdot (1 - p_{v,i,t})^{n_{v,i,f}}$$

and now find the values of p_r, p_w and $p_{v,i,t}$ that maximizes the product.

Let's start with a simple case. Suppose that we have a weighted coin that comes up heads with some unknown probability p. We flip it 100 times, and it comes up heads 90 times and tails 10 times, in some specific sequence. What is the value of p that makes this outcome most likely? Well, the probability of this outcome is $f(p) = p^{90} \cdot (1-p)^{10}$. To find the value of p that maximizes f, we differentiate f with respect to p, set it equal to 0, and solve.

$$0 = df/dp = 90p^{89}(1-p)^{10} - 10p^{90}(1-p)^9 = p^{89}(1-p)^9[90(1-p) - 10p] = p^{89}(1-p)^9[90 - 100p]$$

So $p = 0.9 = 90/100$, exactly the frequency of heads in the outcome. Not surprising, but gratifying.

Generalizing, suppose that the coin comes up head h times and tails t times. The probability of a specific sequence with that outcome is $f(p) = p^h(1-p)^t$. Setting the derivative of f equal to 0 gives

$$0 = df/dp = hp^{h-1}(1-p)^t - 10p^h(1-p)^{t-1} = p^{h-1}(1-p)^{t-1}[h(1-p) - tp] = p^{h-1}(1-p)^{t-1}[h - (h+t)p]$$

So $p = h/(h+t)$, again, exactly the frequency of heads in the outcome.

Now, consider a roulette wheel with three possible outcomes: r, w, b which come up with probabilities p, q, and $1 - p - q$. You spin the wheel a number of times, and it comes up red k times, white m times, and blue n times. The probability of this outcome is $f(p,q) = p^k q^m (1-p-q)^n$. To find the maximum of f, take the partial derivatives with respect to p and q and set both of them equal to 0:

$$0 = \partial f/\partial p = kp^{k-1}q^m(1-p-q)^n - np^k q^m(1-p-q)^{n-1} = p^{k-1}q^m(1-p-q)^{n-1}[k(1-p-q) - np]$$

$$0 = \partial f/\partial q = mp^k q^{m-1}(1-p-q)^n - np^k q^m(1-p-q)^{n-1} = p^k q^{m-1}(1-p-q)^{n-1}[m(1-p-q) - nq]$$

So p and q satisfy the pair of simultaneous linear equations
$k - (k+n)p - kq = 0$
$m - mp - (m+n)q = 0$

which has the solution $p = k/(k+m+n), q = m/(k+m+n)$ so again the probability is exactly equal to the frequency.

Now let go back to the formula in our Naïve Bayes calculation:

$$P(T|Q) = p_r^{n_r} \cdot p_w^{n_w} \cdot (1 - p_r - p_w)^{n_b} \cdot \prod_{v \in \{r,w,b\}, i \in \{1,2\}} p_{v,i,t}^{n_{v,i,t}} \cdot (1 - p_{v,i,t})^{n_{v,i,f}}$$

To maximize that, take the partial derivative with respect to each of the p's and set all of them equal to 0.

That sounds horrendous. But in fact, as far as p_r and p_w are concerned, the whole product involving the $p_{v,i,t}$'s is just one complicated constant; and as far as any of the $p_{v,i,t}$ everything in the product except for $p_{v,i,t}^{n_{v,i,t}}(1 - p_{v,i,t})^{n_{v,i,f}}$ is a complicated constants. So each of these boils down to the two simple cases above, and, finally, finally, we have the solution:

$$p_r = n_r/(n_r + n_w + n_b)$$
$$p_w = n_w/(n_r + n_w + n_b)$$

$p_b = n_b/(n_r + n_w + n_b)$
$p_{v,i,t} = n_{v,i,t}/(n_{v,i,t} + n_{v,i,f})$
$p_{v,i,f} = n_{v,i,f}/(n_{v,i,t} + n_{v,i,f})$

That is, we are choosing the probabilities to be the corresponding frequencies, just like Naïve Bayes prescribes.

And now if you want, you can go onto consider the case of arbitrarily many predictive attributes, each of which has an arbitrary number of values. It's all pretty much the same as this; just somewhat harder to keep track of the subscripts.

Other stochastic generative models

Many other machine learning techniques can be viewed as stochastic generative models, as shown in the following table:

ML technique	**Stochastic model**
Least squares method	Exact line/curve/surface with Gaussian noise in the dependent variable
Principal component analysis	k-dimensional surface with Gaussian noise in all dimensions
k-means (Objective function)	Multiple Gaussian processes of equal variance, with a fixed assignment of points to processes.
EM maximization (Objective function)	Multiple Gaussian processes, randomly chosen with differing variances.
Learned Markov models (e.g. k-gram)	Hidden Markov model.
Probabilistic context-free grammar	PCFG.

The above notation "Objective function" means that the k-means and the EM algorithms are "trying" to find the maximum probability solution, but are not guaranteed to succeed.

Is this two-step process what probability theory actually says you should do?

Finally, you might ask: In stochastic generative models, you first find the most probable process Q from T and then use Q to predict the most likely value of $X.C$ from $X.A_1 \ldots X.A_k$. But why use two steps? Can we do it in one step: Predict the most likely value of $X.C$ from $X.A_1 \ldots X.A_k$, together with the training data? Does that give us the same answer?

The answers to those questions are, yes, you can do that, at least in principle, and, no, it does not necessarily give you the same answer.

You proceed as follows. Again, we have to begin with the modeling step of choosing the space of processes Φ. For simplicity, I'll assume that Φ is actually finite; if you are dealing with an infinite space (you generally are) then all the summation signs below have to replaced with integral signs, but otherwise the analysis is the same.

We are trying, then to calculate the most likely value of the classification attribute of an instance X, given the predictive attributes of X and the training set T, on the assumption that both T and

X were created by independent samplings of some process $Q \in \Phi$.

$\text{argmax}_v\ P(X.C = v \mid X.A_1 = u_1 \ldots X.A_k = u_k, T) =$
$\text{argmax}_v\ \sum_{Q \in \Phi}\ P(X.C = v, Q \mid X.A_1 = u_1 \ldots X.A_k = u_k, T) =$
$\text{argmax}_v\ \sum_{Q \in \Phi}\ P(X.C = v \mid X.A_1 = u_1 \ldots X.A_k = u_k, Q) \cdot P(Q|A_1 \ldots A_k, T) =$
$\text{argmax}_v\ \sum_{Q \in \Phi}\ P(X.C = v \mid X.A_1 = u_1 \ldots X.A_k = u_k, Q) \cdot P(T|A_1 \ldots A_k, Q) \cdot P(Q)/P(T) =$
$\text{argmax}_v\ \sum_{Q \in \Phi}\ P(X.C = v \mid X.A_1 = u_1 \ldots X.A_k = u_k, Q) \cdot P(T|Q) \cdot P(Q)$

That is very similar to the prediction from the two-step approach, except that rather than choosing the best Q and sticking with it, we are adding up over all $Q \in \Phi$. So suppose that there is one process $Q_0 \in \Phi$ that is the most likely given the training set T, which thinks that $X.C$ should be red, based on the predictive attributes, but there are lots of other processes, each not much less likely than Q_0 which all think that $X.C$ should be blue. In the two-step approach, where we choose the best Q, the classifier will predict red; in the one-step approach where we add up over all Q, the classifier will predict blue. This is not a purely theoretical possibility; there are many actual cases where this comes up.

However the one-step approach is rarely used in practice. In training, the two step approach just requires that the system find the Q that maximizes $P(Q|T)$. The one step approach requires that the system compute, and somehow record, $P(Q|T)$ for all processes Q. Worse, at inference time, the two-step approach involves using only one stochastic process to predict $X.C$, namely Q_0. The one-step approach would require computing the results from all possible processes and adding them up, or integrating them. That is rarely feasible in the desired time for inference.

Still, the one-step approach is not entirely ignored. For instance, one can (at least sometimes) justify the Laplacian correction to Naïve Bayes, and some other kinds of smoothing, in terms of a one-step approach.

To see how this works, let's go back to the simplest case, of flipping a coin and predicting the next flip. Suppose that flipped the coin twice and it came up heads both times. According to the calculation we did before, we should now predict that on the next flip it will come up heads with probability 1. That seems rather extreme.

We can avoid that by doing the full Bayesian calculation. Let us suppose that the weighting of a coin w is uniformly distributed between 0 and 1. Then we can calculate:

$P(H_3 \,|H1, H2) = P(H_3, H_1, H_2)/P(H_1, H_2)$

For the numerator:

$$P(H_1, H_2, H3) = \int_{w=0}^{1} P(H1, H2, H3 \mid w) \cdot \tilde{P}(w) dw = \int_{w=0}^{1} w^3 \cdot 1 \, dw = 1/4.$$

Similarly,

$$P(H_1, H_2) = \int_{w=0}^{1} w^2 \, dw = 1/3.$$

So $P(H_3 \,|H1, H2) = P(H_3, H_1, H_2)/P(H_1, H_2) = 3/4$, which is the Laplacian correction.