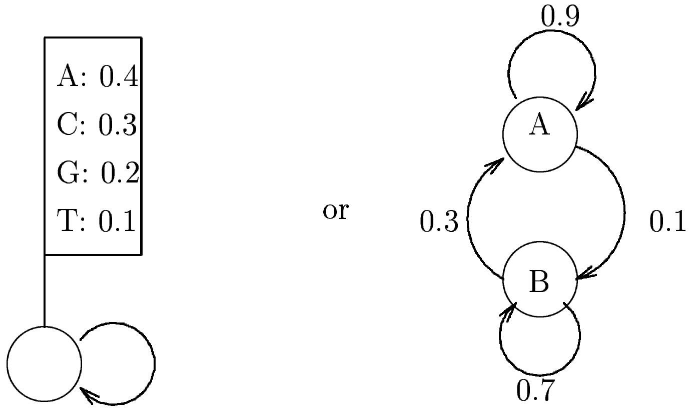
Probabilistic Sequence Models

A *probabilistic sequence model* is a set of instructions for generating sequences (over some specified alphabet of permissible characters), which can use probabilistic operations, such as:

- print a letter that is `A` with probability 0.4, `C` with probability 0.3, `G` with probability 0.2, or `T` with probability 0.1

- go to statement 1 with probability 0.4, to statement 5 with probability 0.3, to statement 17 with probability 0.2, or to statement 9 with probability 0.1

We will specify a model with a kind of "flowchart", such as the following.



On the left is pictured a model that repeatedly prints a letter according to fixed *emission probabilities.* Such a model (perhaps with different letters and/or probabilities) is called an *i.i.d.* (independent identically distributed) model. With the model pictured on the right, each *state* (the conventional name for nodes or vertices in this context) emits a fixed letter, and the probabilistic nature comes from the *transition probabilities*, which control the frequency with which the next state is chosen at each step. Here, for instance, the model emits runs of `A` and of `B`, with the former generally longer than the latter. A model of this sort, i.e., where each state emits a fixed letter *and where no two states have the same letter*, is called a *Markov chain.* It is straightforward to see that Markov models are more general than i.i.d. models, i.e., for any i.i.d. model there is an equivalent Markov model, but not necessarily vice versa. There are a number of additional kinds of probabilistic sequence models, as we will soon discover.

Scoring a given sequence. For a probabilistic sequence model to be really useful, we will need to have a second set of instructions that, when given a sequence, will score the sequence according to the model. If there is only one set of instructions (i.e., path through the flowchart) that generates the sequence, then the score is simply the product of all emission and/or transition probabilities along that path. For instance, the score of `AAC` for the above i.i.d. model is $(0.4)^2(0.3)$. It is not difficult to see that for any i.i.d. model, the sum of the scores for all sequences *of a fixed length* is 1, i.e., the model provides a probability distribution for all sequences of a given length. (A sequence containing a letter that is not emitted by the model has probability 0.) A Markov model, e.g., above on the right, also provides a probability distribution for sequences of a fixed length, so long as we decide how the first state is picked. One approach is to add an unlabeled "start state", with transitions (having probabilities) to the other states.

For a probabilistic model where several paths through the flow chart can generate the same output sequence, scoring a given sequence is not so straightforward. This is why we required that no two states emit the same letter. There are cases where we can allow duplicate letters and yet preserve the property of uniqueness of the path. For example, see "inhomogeneous Markov models", below. Later, when we discuss "hidden Markov models", we will see what happens when an output sequence can be generated in more than one way.

Learning a model's parameters. We can think of a model's probabilities (emission and/or transition) as parameters that can be varied. Given a "training set" of sequences, we may want to fit the parameters to the training set. For instance, in the above i.i.d. model for DNA sequences, suppose we try to adjust the emission probabilities to fit the sequence:

AACCGTACCAGGATGCCACC

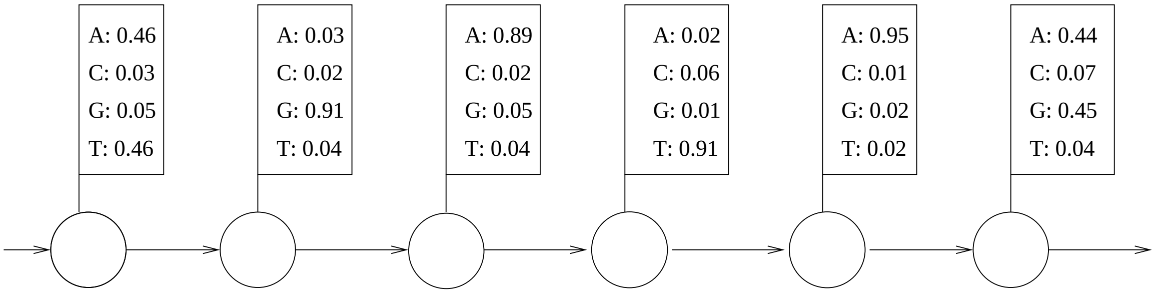
which has 6 A's, 8 C, 4 G's and 2 T's. It is natural to estimate the emission probabilities as simply the fraction of that letter in the training set, which in this case exchanges the probabilities of A and C in the above picture. But why is that better than their original settings? The probability of AACCGTACCAGGATGCCACC given the original parameter values is $p_1 = (0.4)^6(0.3)^8(0.2)^4(0.1)^2$, whereas with the new values it is $p_2 = (0.4)^8(0.3)^6(0.2)^4(0.1)^2$, so the ratio p_2/p_1 is $(0.4)^2/(0.3)^2 > 1$. In general, it can be show that estimating parameters of an i.i.d. model simply as the frequencies in the training set gives the *maximum likelihood estimate*, i.e., it maximizes the score (i.e., probability) of the training set over all possible parameter settings.

Let us verify this in a simple case. Consider an i.i.d. model that emits X with probability p and Y with probability $1-p$. Suppose we want to "learn" p from a training set with 8 X's and 5 Y's, so the relative frequency of X is 8/13. The probability of the training set is $P(p) = p^8(1-p)^5$, which equals 0 if p is 0 or 1, but iis positive if $0 < p < 1$. Thus the maximum value is attained when the derivative, $P'(p)$, equals zero. Some rudimentary calculus shows that the maximum is, indeed, attained at $p = 8/13$.

Similarly, estimating the transition probabilities in a Markov model to be the relative frequency that the edge is taken, over all paths that are traversed as the training set is generated, gives the maximum likelihood estimator. For instance, suppose we want to model CpG islands, which are regions of DNA sequence that have a relatively high proportion of CG dinucleotides. Since we're modeling a set of DNA sequences, the Markov chain will have four nodes. Suppose that in a training set of verified CpG islands, there are 1000 C nucleotides, neglecting those that lie at the very end of a training sequence. If, of these, 267 are immediately followed by a G, then the C-to-G transition probability is estimated as 0.267.

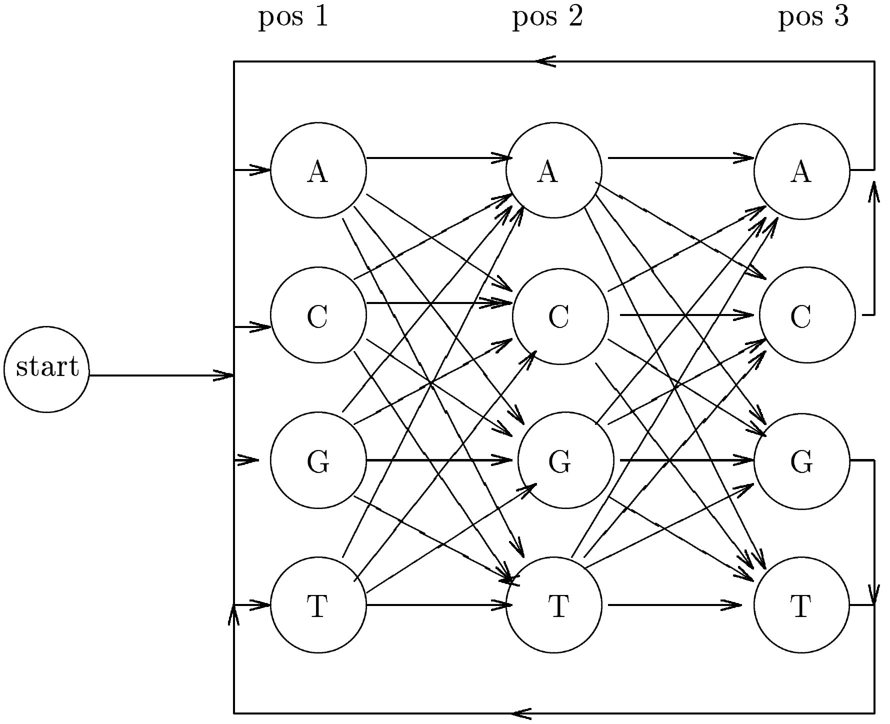
In summary, for any useful probabilistic sequence model we ask for three things: (1) instructions to generate sequences according to the model, (2) instructions to score any given sequence according to the model and (3) a way to adjust the model's parameters according to any given training set of sequences. Actually, to a certain extent, (2) implies (3). Suppose that for any setting of the model parameters and any given sequence we can determine the probability that the model generates the sequence. Then, we can use a numerical optimization procedure to adjust the parameters so as to maximize the sequence's probability (i.e., perform maximum likelihood estimation), although this approach faces all of the normal hurdles of numerical optimization, such as avoiding local maxima.

Some other kinds of probabilistic sequence models. Suppose we want to model DNA binding sites for the GATA transcription factor, whose consensus binding site is WGATAR, where W means A or T, and R means A or G. Many actual GATA binding sites deviate from this consensus in at least one position, and a more accurate representation than the consensus sequence might be derived from the frequencies observed in a sample of experimentally confirmed GATA binding sites, as follows.



Another example comes from modeling introns. Frequently, the percentages of the various nucleotides is about the same in introns and exons, but that there are significant differences in relative frequencies of hexamers (six consecutive nucleotides). For instance, the hexamer `ACACAC` might be common in introns but rare in exons, whereas appearance of certain other hexamers might provide evidence that the sequence is from an exon. We might try modeling intron sequences as a *Markov model of order 5*, i.e., for each possible combination of five consecutive nucleotides, the next nucleotide occurs with a certain probability distribution. For example, given `ACACA`, the next nucleotide might be `C` with probability 0.31. Such a model has $4 * 4^5 = 4096$ parameters. Actually, such a fifth-order Markov chain for DNA sequences is equivalent to a first-order (i.e., regular) Markov model with 1024 states — one for each 5-mer. Each state has transitions to only 4 states, e.g., from `ACACA` to `CACAC`, but not to `TTTTT`.

For modeling exons, let's suppose that dinucleotide frequencies are useful. (Actually, one wants to use hexamer frequencies, but that would complicate the following discussion.) That is, we are going to assume that each nucleotide can be predicted depending on the immediately preceding nucleotide. The point here is that with coding regions, it is well known that nucleotide content depends strongly on position in the codon. For instance, `C` might follow `A` with probability 0.31 if the `C` is at the start of a codon, but only probability 0.23 if it is in the middle position. Thus, it is natural to use an *inhomogeneous Markov model*, pictured like this:



In this picture, transition probabilities are not shown. Each state, including the start state, has 4 transition probabilities, for a total of 52 parameters in the model. Although it violates the ban on duplicate labels, no problems arise since a given nucleotide sequence is generated by a unique path, and so scoring is straightforward. It is left to the reader's imagination to see how this model can be generalized to an inhomogeneous Markov model of order 6, which is quite useful in gene-prediction programs for modeling coding regions.

A typical use of probabilistic sequence models. We've already seen a use that illustrates a general theme in bioinformatics, namely we used an i.i.d. model of "correct alignments" to determine good ways of scoring pairwise alignments. There, the underlying alphabet of symbols was just the possible alignment columns, so that a sequence of these symbols was a pairwise alignment. For instance, to derived BLOSUM62 scores, we started with a training set of protein alignments (where the two sequence could be at most 62% identical). An i.i.d. model of these sequences was training according to maximum likelihood (i.e., simply using relative frequencies of each of the 400 possible symbols). We then determined an i.i.d. null model for "random alignments" (i.e., of unrelated sequences. The next step was to score a given alignment as the ratio of the probabilities under these two models (the "odds ratio"). The logarithm of this ratio was used to permit computation using sums instead of products, and to keep the size of the resulting numbers within a reasonable range.

This illustrates the general theme of applying probabilistic sequence models to identify a class of biological sequences, such as GATA binding sites. We pick an appropriate type of model, select the maximum likelihood parameter values using a training set, and score a potential member of the class using the logarithm of the ratio of the probability that the sequence is generated by the model versus the probability that it is generated by an appropriate null model. For instance, with some assumed backgound nucleotide composition, this might give the following *weight matrix.* (Values given here are purely fictitious.)

pos	1 (W)	2 (G)	3 (A)	4 (T)	5 (A)	6 (R)
A	2.2	–3.3	5.1	–2.9	4.8	1.9
C	–4.0	–3.8	–2.9	–4.1	–3.7	–4.1
G	–3.4	5.1	–4.1	–3.2	–2.9	2.2
T	3.1	–4.4	–3.9	4.4	–3.8	–3.7

Be sure to understand the difference between such a weight matrix and the probabilities (or frequencies) in the underlying probabilistic model. For instance, entries of the weight matrix don't need to fall between 0 and 1, and we score a sequence by adding the entries, not by multiplying them.