

Markov models

We discuss Markov models now. These have relevance in a few ways:

1. Markov models are a good way to model local, overlapping sets of information, which reflect cues to our understanding of regions.
2. Markov chains provide a stochastic model of diffusion that applies to individual particles, which gives us the foundation for diffusion we've studied.
3. This stochastic diffusion also provides a useful model of the spread of information throughout an image.

We will start by discussing the most simple Markov model, a Markov chain, which can serve as a model of diffusion processes. In a Markov chain, we have a sequence of random variables, which we can think of as describing the state of a system, x_1, x_2, \dots, x_n . What makes them Markov is a conditional independence property that says that each state only depends on the previous state. That is:

$$P(x_n | x_1, x_2, \dots, x_{n-1}) = P(x_n | x_{n-1})$$

Diffusion of a single particle offers us a simple example of this. Let x_i be the position of a particle at time i . Suppose at each time step, the particle jumps either one unit to the left or right, or stays in the same place. We can see that x_i depends on x_{i-1} , but that if we know x_{i-1} the previous values of x are irrelevant. We can also see that this is a stochastic version of the deterministic diffusion we've studied. If there are many particles, we can use the law of large numbers to assume that a fixed fraction of them jump to the left and right. Remember that we saw that the position x_i will have a Gaussian distribution for large i , because of the law of large numbers.

Many physical processes can be modeled as Markov chains, because this just means that the new state only depends on the previous state. We can also use diffusion as a model of contour shape. We can view this as a prior on the shape of contours, so that if we want to reconstruct an image in which contours have been fragmented, we can do this by combining image information with a prior. We can model the smoothness of contours by a shape model in which the direction of contours tends to be smooth, but diffuses according to a Gaussian.

With this prior, the probability of a contour is the product of the Gaussian distribution. The log of this is the integral of the curvature squared, which gives an intuitive notion of smoothness.

If a Markov chain involves transitions between a discrete set of states, it's very useful to describe these transitions from state to state using vectors and matrices. Let p^t_j be the probability of being in state j and time t . Now let's put these in a vector, so that: $p^t = (p^t_1, \dots, p^t_n)$ gives the probability distribution over all states we can be in. These are probabilities because even if the state at time 0 is deterministic, after that it will be stochastic. Notice that $\sum_j p^t_j = 1$. Now, the Markov model is completely specified by the probability that if I'm in state s_j at time $t-1$ that I'll be in state s_i at time t , for all

i and j . We build a matrix with these probabilities, called A , with entries A_{ij} . Notice that every column of A has to sum to 1 because if I am in state j at time $t-1$ I have to move to exactly one state at time t .

This is convenient, because we have:

$$p^t = A p^{t-1}.$$

This just says that the probability that I wind up in, eg., state 1 at time t is the sum of the probability that I'm in state i at time $t-1$ and move to state 1, for all possible i 's.

Notice that this is more general than the diffusion processes we solved with convolution. This matrix multiplication is only equivalent to a convolution if the matrix is a band around the diagonal, with every row the same, but shifted.

This formulation makes it easy to study the asymptotic behavior of a Markov chain. It is just the limit of:

$$A(A(\dots(A(p^0)\dots))) = A^n(p^0)$$

Notice that because A is stochastic, the elements of p^t always sum to 1.

As an example, let's consider the simple Markov chain given by:

$$A = \begin{bmatrix} .75 & .5 \\ .25 & .5 \end{bmatrix};$$

Using Matlab, we can see that if we pick p randomly and apply A to it many times we get $[2/3, 1/3]$ as our answer. We can work out that this is a steady state of the system.

Suppose we have $p^t = [2/3, 1/3]$. Then the chances that we will wind up in state 1 at time $t+1$ is $2/3 * 3/4 + 1/3 * 1/2 = 1/2 + 1/6 = 2/3$.

We can also see, with Matlab, that this asymptotic state is an eigenvector of A . This is because this repeated multiplication is just the power method of computing the eigenvector associated with the largest eigenvalue of A . That is, if we keep multiplying p by A , the result converges to the leading eigenvector. This is true regardless of the initial condition of p .

Proof by example: Suppose $p = av_1 + bv_2$, where v_1 and v_2 are eigenvectors of A , with associated eigenvalues of λ_1 and λ_2 . Then:

$$A(av_1 + bv_2) = A(av_1) + A(bv_2) = a\lambda_1 v_1 + b\lambda_2 v_2. \text{ And similarly:}$$

$$A^n(av_1 + bv_2) = a(\lambda_1)^n v_1 + b(\lambda_2)^n v_2.$$

As n goes to infinity, if λ_1 or λ_2 is smaller, it will become insignificant, and the larger eigenvalue dominates. If this is λ_1 , the result converges to a vector in the direction of v_1 . This also means that the leading eigenvalue of A must be 1; this happens because A is a stochastic matrix.

I've skipped some conditions needed to prove this. First, the stochastic process must be ergodic. This means that we can get to any state from any other state. Otherwise, we might start in one state and never be able to get to the leading eigenvector from it.

Second, the Markov chain must have a unique leading eigenvalue. Otherwise, the result could vary among linear combinations of the leading eigenvectors. Or the result might not converge, but could be periodic. For example, if

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

then $(1/2 \ 1/2)$ is an eigenvector, but the result can also oscillate. Third, the initial condition must be generic. If it has no component of the leading eigenvector in it, it can't converge to that.

So the leading eigenvector of the transition matrix gives us a probability distribution over the asymptotic state. This can also be interpreted as the expected fraction of time the system stays in each state, over time. This makes sense, since the probability that the system is in a state at time n is the expected amount of time it spends there at time n .

Linear Relaxation Labeling

Now we're going to look at relaxation labeling, which is a classic approach to solving various image interpretation problems. In particular, we'll look at linear relaxation labeling, which is closely related to Markov chains. By the way, R.L. was developed at the University of Maryland by Hummel, Zucker and Rosenfeld. To some extent it has been superseded now by Belief propagation, but it's still interesting, it introduces some ideas that are used in belief propagation, and it also tends to get reinvented, so it's good to know.

Relaxation labeling is a way to assign labels to different portions of an image. It can be used when there are multiple possible labels, but we'll keep things simple by considering the case where there is just one, binary label. As an example, suppose we have a bunch of edgelets, and we want to assign each one a label as either figure or background. We use relaxation labeling when the correct label for one image object provides information about other image objects. We iterate, and at each iteration we update the label of each object based on input from other objects. For example, if we have nearby, smoothly connecting edgelets, then if one is figure, the other is also likely to be figure.

In L.R.L. we assign each edgelet a measure of figureness that ranges from 0 to 1. 0 means it seems most like background, and 1 means most like foreground. Then we assign a weight from one edgelet to another. This weight is high if when one is figure, this makes us think the other is figure. To use this in updating the edgelet, we multiply the figure label of the other edgelet by this weight. We do this for all edgelets, and add up the results. So we can list the labels in a vector, v , and stick the *affinities* in a matrix. Then, to update the labels, we just have to multiply the label vectors by the affinity matrix.

Of course, this looks just like finding the asymptotic state of a Markov model. So one way to interpret Linear Relaxation Labeling is that a particle is jumping around from one edgelet to another. We start with particles distributed around the edgelets, and let them

jump from one edgelet to another. They will tend to collect on the figure, where there is good continuation, and die out at isolated edgelets.