Markov chains and Markov Random Fields (MRFs)

1 Why Markov Models

We discuss Markov models now. This is the simplest statistical model in which we don't assume that all variables are independent; we assume that there are important dependencies, but also conditional independencies that we can take advantage of. Markov models show up in a lot of ways in computer vision. For example:

- In modeling texture, we can represent a patch of an image by output of a large number of filters. If we assume that some of these filter outputs are dependent on each other (for example, filters at adjacent scales), but that there is conditional independence between different filter outputs, we can get a tractable model for use in texture synthesis.
- The shape of contours can be modeled as a Markov chain. In this approach, the future course of a contour that passes through point x might depend on the tangent of the contour at x, but not on the shape of the contour prior to reaching x. This oversimplifies, but is implicitly used in many approaches to perceptual grouping and contour segmentation.
- In action recognition, we often model the movements that constitute an action as a Markov chain. This assumption means, for example, that if I want to predict what is about to happen when someone is in the middle of leaving work and getting in their car, that this will depend on their current state (where is the person and car right now, and what is their current trajectory) but not on previous states (once you are standing in front of the car, how you will get in doesn't really depend on how you left the building and arrived in front of the car).
- Again, in modeling texture, we might assume that the appearance of a pixel in a textured image patch will depend on some of the surrounding pixels, but given these pixels, it will be independent of the rest of the image.

2 Markov Chains

We will start by discussing the most simple Markov model, a Markov chain. We have already talked about these a little, since diffusion of a single particle can be thought of as a Markov chain. We can also use Markov chains to model contours, and they are used, explicitly or implicitly, in many contour-based segmentation algorithms. One of the key advantages of 1D Markov models is that they lend themselves to dynamic programming solutions.

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, x_n. What makes them Markov is a conditional independence property that says that each state only depends on the previous state. That is:

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

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 central limit theorem.

Markov chains have some nice properties. One is that their steady state can be found by solving an eigenvalue problem. 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_j^t be the probability of being in state j and time t. Now let's put these in a vector, so that: $p^t = (p_1^t, , p_n^t)$ 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 $\sum_j p_j^t = 1$. 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. (In principle, these probabilities might also depend on t, that is, vary over time, but I'll assume for now that they don't.) 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: $p^t = Ap^{t-1}$. 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:

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

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: $A = [.75.5; .25.5]$; 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 $p^t = [2/31/3]$. Then the chances that we will wind up in state 1 at time t+1 is $2/3 * 3/4 + 1/3 * 1/2 = +1/6 = 2/3$.

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:

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

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_2 is a smaller than λ_1, it will become insignificant, and the larger eigenvalue dominates. 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 $(\frac{1}{2}, \frac{1}{2})$ 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.

Markov Random Fields

Markov chains provided us with a way to model 1D objects such as contours probabilistically, in a way that led to nice, tractable computations. We now consider 2D Markov models. These are more powerful, but not as easy to compute with. In addition we will consider two additional issues. First, we will consider adding *observations* to our models. These observations are conditioned on the value of the set of random variables we are modeling. (If we had considered observations with Markov chains, we would have arrived at Hidden Markov Models (HMMS), which are widely used in vision and other fields).

In MRFs, we also consider a set of random variables which have some conditional independence properties. We introduce some terminology which is slightly different from the 1D case. We say that the MRF contains a set of *sites*, which we may index with two values, i and j, when we wish to emphasize the 2D structure of the sites. So we might talk about site $S_{i,j}$ and it's horizontal neighbor, $S_{i+1,j}$. It may also be convenient to use a single index, referring to sites as S_i. In this case, the order of the sites is arbitrary.

We also suppose that we have a set of labels, L_d. Labels might take on continuous values, but we'll assume that we have a discrete set of labels. Every site will have a label. So the sites may be thought of as random variables that can take a discrete set of values. A *labeling* assigns a label to every site. We denote this as $f = \{f_1, ..., f_m\}$, so that f_i is the label of site S_i.

Next, we assume that our sites have a neighborhood structure. N_i denotes all the sites that are neighbors to S_i. That is, $S_j \in N_i$ means S_j and S_i are neighbors. We can define any neighborhood structure that we want, with the constraint that being neighbors is symmetric, that is, that $S_j \in N_i \Leftrightarrow S_i \in N_j$. Also, a site is not its own neighbor. We denote the set of all neighborhoods as N.

This neighborhood structure now allows us to define the Markov structure of our distribution. We say that F is an MRF on S with respect to N if and only if:

$$P(f) > 0, \forall f$$

and

$$P(f_i | f_{S-\{i\}}) = P(f_i | f_{N_i})$$

The first condition is needed for some technical reasons. It means that in modeling we can't make any labeling have 0 probability, but since the probability can be very small this isn't much of a restriction. The second condition provides the Markov property.

As an example of this, let's consider the problem of segmenting an image into foreground and background. We can assign a site to every pixel in the image. Our label set is binary, indicating foreground or background. Suppose we wish to encode the constraint that foreground regions tend to be compact, by stating that if a pixel is foreground, its neighboring pixels are also likely to be foreground. We can define a simple neighborhood structure based on 4-connected neighbors. That is, $N_{i,j} = \{S_{i-1,j}, S_{i+1,j}, S_{i,j-1}, S_{i,j+1}\}$ (notice how we switch from using one to two subscripts). Then with the conditional probabilities available to us, we can encode constraints such as that if all of a pixels neighbors are foreground, it is probably foreground, if all its neighbors are background, it is probably background, and in other cases, it is fairly likely to be either. (You may also notice that this MRF has no connection yet to the intensities in the image. This will be handled below.)

MRFs and Gibbs Distributions

Given any set of random variables, there are a number of natural problems to answer. First, given an MRF it is straightforward to determine the probability of any particular labeling. However, figuring out what set of conditional probabilities to use in an MRF is not so simple. For one thing, an arbitrary set of conditional probabilities for different sites and neighborhoods may not be mutually consistent, and it is not obvious how to determine this. Finally, a key problem will be to find the most likely labeling of an MRF.

These problems are made easier by the use of Gibbs distributions, which turn out to be equivalent to MRFs, but in some ways are much easier to work with. In a Gibbs distribution, the cliques capture dependencies between neighborhoods. A set of sites, $\{i_1, i_2, ..., i_n\}$ form a clique if for all k, j, $i_k \in N_j$. Given a probability distribution defined for a set of sites and labels, we say that it is a Gibbs distribution if the distribution is of the following form:

$$P(f) = \frac{1}{Z} e^{-\frac{U(f)}{T}} \tag{1}$$

in which the *energy function*, $U(f)$, is of the form:

$$U(f) = \Sigma_{c \in C} V_c(f)$$

where C is the set of all cliques, and $V_c(f)$ is the *clique potential*, defined for every clique. That is, $P(f)$ is an exponential function over the sum of potentials that can be defined independently for each clique. This is analogous to the independence structure given by neighborhoods in MRFs. In the above terms, T is a scalar called the temperature. In the above equation, Z is a normalizing value needed to make the probabilities sum to 1:

$$Z = \Sigma_{f \in F} e^{-\frac{U(f)}{T}}$$

T is a scalar that determines how sharply peaked the distribution is; note that as T becomes very small, the distribution is dominated by its most likely element.

The main reason Gibbs distributions are important to us is that they turn out to be equivalent to MRFs. That means that for any MRF, we can write it's probability distribution in the form of Equation 1. That means that in learning or designing an MRF, we can focus on finding the clique potentials. Note that to fully specify this distribution is still hard, since we must determine the value of Z. A straightforward way of computing this involves computing a sum with an exponential number of terms. There is a lot of work on approximating this value. However, in many cases we don't need it, because to find the MAP distribution for an MRF we just have the problem of finding the labeling that minimizes the energy function $U(f)$, since Z is a constant factor that applies to all labeling. Finding the labeling that minimizes $U(f)$ is still an NP-hard combinatorial optimization problem in most cases, but there are many algorithms that attack this problem.

MRFs with Observations

So far, we have only considered distributions over labels. This amounts to a means for specifying a prior distribution over labeled images. But we also want to connect this with the information in a specific image. To do this, we'll use examples in which every site is a pixel, so that there is one piece of image information at each site. We'll call the image information X, with the information at each pixel given by X_i or $X_{i,j}$. We are then interested in solving problems like:

$$argmax_f P(f|X)$$

To make this concrete, let's consider an example of image denoising. We consider an MRF in which each pixel is a site, and two pixels are neighbors if they are 4-connected. Suppose every pixel has an intrinsic intensity from 0 to 255, given by its label. X_i is this intensity, with noise added, so that

$$X_i = f_i + e_i$$

where the e_i are iid and drawn from a zero mean, Gaussian distribution with variance σ^2. Using Bayes law we have:

$$P(f|X) = \frac{P(X|f)P(f)}{P(X)}$$

To compute this we have:

$$P(X|f) = \Pi P(X_i|f_i)$$

Note that each X_i is conditionally independent of all labels, given f_i, ie., that

$$P(X_i|f) = P(X_i|f_i)$$

and also that the X_i are independent of each other given the labels, ie., that:

$$P(X|f) = \Pi P(X_i|f)$$

Our noise model states that the $P(X_i|f_i)$ will be a Gaussian distribution with mean f_i, so that:

$$P(X_i|f_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(f_i - X_i)^2}{2\sigma^2}}$$

Therefore, we can define the clique potential:

$$V_c(f_i) = \frac{(f_i - X_i)^2}{2\sigma^2}$$

(Note we can ignore constant factors, since we normalize anyway with Z). These are the unary cliques, which capture the data (pixels). If we want to, we can add to this a prior on different labels.

We can also define a pairwise clique potential to encode the prior that neighboring pixels should have similar intensities. The simplest way to do this is to define:

$$\begin{aligned} For \quad & c = \{i, j\} \\ V_c = 0 \quad & f_i = f_j \\ V_c = k \quad & f_i \neq f_j \end{aligned}$$

This biases us to have piecewise constant regions in the restored image. On the other hand, if we give:

$$V_c = \|f_i - f_j\|$$

we penalize according to the total variation in the image.

Computing MAP estimates of MRFs and CRFs

There are several kinds of computations that we might want to perform with MRFs and CRFs. These include learning the clique potentials and other parameters, finding a MAP estimate of the labels, given an MRF/CRF and image information, sampling from the conditional distribution (instead of just getting a MAP estimate). We will focus mainly on the MAP estimate problem, since this is very useful.

Initialization: All these algorithms are iterative, and the results can depend on how the solution (labeling) is initialized. The simplest initialization method is to compute the MAP estimate using only the unary clique potentials. So, for denoising, for example, we would initialize the labels to be the corresponding pixel intensity. Of course, there are many other standard heuristics, such as using domain knowledge (ie., a prior on the labels), or trying many random initializations and picking the one that leads to the best solution.

Iterated Conditional Modes: This is the simplest, greedy algorithm. Visit the sites in some order, and for a given site, S_i, choose the label f_i that maximizes $P(f)$ given that all other labels are fixed. Notice that this only requires computing the clique potentials that include S_i for all possible labels, so this requires computation that is linear in the number of labels. ICM is efficient, but can quickly converge to a poor local minimum. It is likely to work well when the unary clique potentials are very strong (note that in the limit, as the unary clique potentials dominate, ICM produces the global optimum.)

Simulated Annealing: This method uses stochastic optimization to avoid some of the local minima that occur with ICM. The idea is to randomly change a label, and then accept this change with a probability that depends on the extent to which this change increases or decreases the overall probability of the labeling. For example, given a set of nodes with labels, f, we randomly select a site, S_i and consider changing the label to f_i'. We let f' denote this new label set containing f_i'. Let $p = \min(1, P(f')/P(f))$. Then we replace f with f' with probability p. This results in our

always accepting a change that improves the labeling, but also in our accepting changes that reduce the probability of the labeling. This can allow us to escape from labelings that are locally, but not globally optimal.

Note that the distribution $P(f)$ becomes very flat for large values of T, and more peaked as T decreases. To take advantage of this, we use an annealing schedule in which T begins with a high value, and gradually decreases. This means that at first we make changes to the labeling almost at random, often moving to less probable labelings, and then gradually move more deterministically only to more probable labelings. It can be proven that if the annealing schedule is chosen properly this will converge to the globally optimal solution, but of course since the problem is NP-hard, this must require an annealing schedule that uses an exponential amount of time.

Belief Propagation: Belief propagation allows for exact inference in graphical models that do not contain loops, such as Markov chain models or models with a tree structure. It has been shown that this can also lead to effective inference in models with loops, such as MRFs, but we won't discuss this algorithm.

Graph Cuts:

We'll talk more about this in the next class.