Lecture 11: Graphical Models for Inference

So far we have seen two graphical models that are used for inference - the Bayesian network and the Join tree. These two both represent the same joint probability distribution, but in different ways. Both express local properties which are used to make the inference computations tractable. The Bayesian net has the desirable property that it expresses a causal relationship between the variables which can be used for reasoning about their behaviour. We will now look at some other graphical models that have been used for inference, and show that they are each specific examples of a more general method called belief propagation.

Tanner Graphs

The Tanner graph was originally used to depict constraints imposed for error recovery using parity checking. It is a bi-partite graph, meaning it has two types of nodes, and each node only has neighbours of the opposite type. A simple example is shown in Figure 1. Here the circles represent bits that are being transmitted down a noisy channel, and the squares represent parity constraints. Each square checks for even parity and will evaluate to 1 if the sum of the three bits connected to it is even. We can think of bits 1,2 and 3 representing the data and bits 4, 5 and 6 as parity bits whose values will help us to recover from an error. Given that an error occurs, we can formulate the recovery procedure as a probabilistic inference problem as follows. Suppose that the probability of a bit being flipped during transmission is P_f, and we write X_i to be the value (0 or 1) that was transmitted for bit b_i and Y_i as the value that is received for bit b_i. We have that:

$$\begin{aligned} P(X_i|Y_i) &= 1 - P_f \; if \; X_i = Y_i \\ & \quad P_f \; otherwise \end{aligned}$$

and we can work out the probability that a set of bits $(X_1, X_2, ..X_N)$ is correct by taking the product of the conditional probabilities:

$$P(X_1, X_2, ..X_N) = \prod_{j=1}^{N} P(X_j|Y_j)$$

Our inference problem is to determine the most likely transmitted string. The above equation simply tells us that the most likely transmitted string is the received string, since the joint probability will always be a maximum for the case where $\forall i : X_i = Y_i$. To use it for error correction we need to impose the parity constraints, which is done simply by multiplying by the parity check functions which we will denote Ψ. So for the example graph of Figure 1 we get the joint probability distribution:

$$P(X_1, X_2, ..X_6) = \Psi(X_1, X_5, X_3)\Psi(X_1, X_2, X_4)\Psi(X_2, X_3, X_6)\prod_{j=1}^{N} P(X_j|Y_j)$$

Any bit string that fails the parity check will now have a probability of zero, and, if this is the case, we can select the most probable bit string from the set of possible valid bit strings in order to recover from the error. Notice that the way the parity functions express a relationship (or structure) between the bits is analogous to the way in which the conditional probability matrices $P(A|B)$ express relationships between variables in a Bayesian network.

Factor Graphs

A factor graph is a graphical model representing the factorisation of a function. It is in fact the most general graphical model,and can express all the probabilistic inference procedures we have seen so far. In general if we write a factorisation as:

$$g(X_1, X_2, X_3, ...X_n) = \prod_{j=1}^{m} f_j(S_j)$$

The corresponding factor graph G is a triple (X, F, E) where:

X	is a set of variables $(X_1, X_2, X_3, ...X_n)$
F	is a set of factors $(f_1, f_2, ...f_m)$
E	is a set of edges joining factors and variables

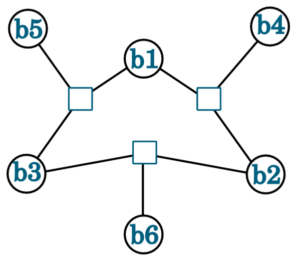

Figure 1: A Tanner graph for parity checking

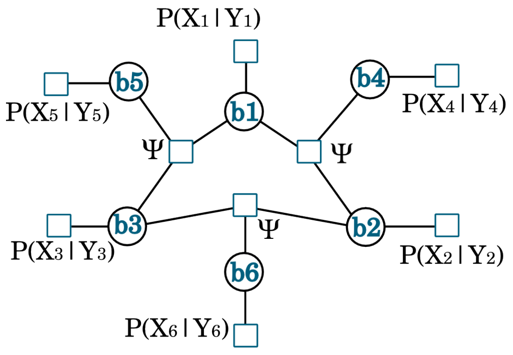

Figure 2: The equivalent factor graph of Figure 1

The factor graph for the six bit parity checker shown in figure 1 is shown in figure 2. Each factor is represented by a square node, and each variable by a round node. We see that the joint probability of the variables is simply the product of all the factors.

We have seen that a Bayesian network can be looked on as factorisation of the joint probability distribution which is made on the basis of known (or assumed) conditional dependencies between the variables. In general we write:

$$P(X_1, X_2, X_3, ...X_n) = \prod_{i=1}^{n} P(X_i|Parents(X_i))$$

So for the metastatic cancer network (Figure 3) we can write:

$$P(A, B, C, D, E) = P(A)P(B|A)P(C|A)P(D|B\&C)P(E|C)$$

and we see that the factor graph variables are the same as the network variables and the factors are simply the prior and conditional probability matrices. The Bayesian network and its equivalent factor graph are shown below.

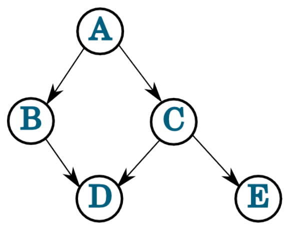

Figure 3: Bayesian Net for Metastatic Cancer

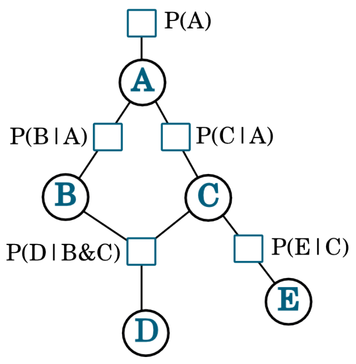

Figure 4: The equivalent factor graph

We have also seen how the join tree method allows us to find a different factorisation of the same probability distribution based on finding a particular set of cliques and for each clique defining a potential function. This representation is called a Markov random field. We can write a general Markov random field factorisation as:

$$P(X_1, X_2, X_3, ...X_n) = \frac{1}{Z} \prod_{j=1}^{m} \Psi_j(V_j)$$

where V_j is a subset of variables (belonging to clique j in the case of a join tree) and Z is a normalisation factor

$$Z = \sum_{X} \prod_{j=1}^{m} \Psi_j(V_j)$$

The sum is take over all the possible states of the complete set of variables X.

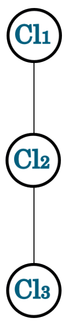

Figure 5: Join Tree for Metastatic Cancer

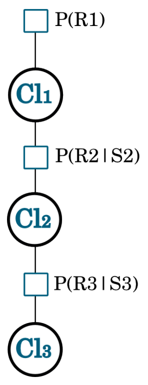

Figure 6: Factor Graph of the Join Tree (Figure 5)

For the metastatic cancer network (Figure 5) we can write (following the notation in the lecture on join trees):

$$P(A, B, C, D, E) = \Psi(C_1)\Psi(C_2)\Psi(C_3) = \Psi(A, B, C)\Psi(B, C, D)\Psi(C, E)$$

The normalisation factor Z is 1 in this case since we initialised the potential functions with exactly the prior and conditional probability matrices, and ensured that any message that divided one potential function multiplied its parent. We see that the join tree is a different factorisation of the joint probability distribution that is also represented by a factor graph (Figure 6). However, in this case the factor graph representation is not particularly useful.

Pairwise Markov Random Fields

Pairwise Markov random fields are commonly used as a graphical structure for making inferences. Their simplicity has meant that they are often used in theoretical studies as a fundamental building block from which other inference algorithms can be derived. In a pairwise field the variables are always joined in cliques of size two, which means that factors join at most two variables. The classic example of the use of pairwise Markov random fields is in the area of image processing. In Figure 7 the image pixels are depicted by filled circles, and are labelled Y_i. Note that the two dimensional image pixels are indexed by a single index i. The inferred model of the image is depicted by the unfilled circles and the pixels are labelled X_i. The definition of the inferred model depends on the application. For example, if we are interested in image segmentation, then the X_i variables may be discrete and used to label segments of the image. In a brain image the labels might be "white matter", "grey matter", "cerebro-spinal fluid" and "lesion", and the inference problem is to decide which label each pixel should have. The Y_i pixels in this case are observed (or evidence) variables, equivalent to the observed bits in the Tanner graph, and the X_i are hidden variables whose values are unknown and cannot be measured, equivalent to the true bit string that the Tanner graph is designed to recover.

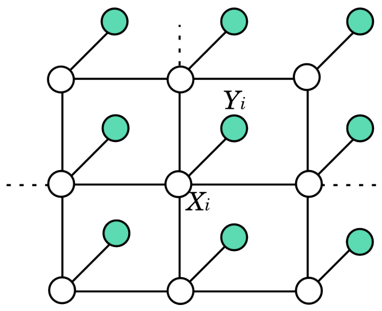

Figure 7: A pairwise Markov random field model of an image

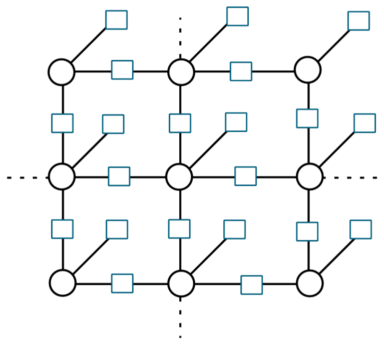

Figure 8: The equivalent factor graph of the pairwise markov random field

The Markov assumption in the model expresses the local property that a pixel is very likely to be similar to its neighbours. In the simplest case, where an image is corrupted by random noise in the measurement process, this assumption can be used, analogously to the parity checking example, to restore the true underlying image from the measured pixels. In segmentation applications the underlying assumption is that the segments that we want to label are large regions with some common property, in which case the Markov assumption is reasonable. For images with rapidly changing textures it does not hold, and preprocessing becomes necessary. Overall we can think of the pairwise Markov random field as fusing both the measured and the structural information to make a decision about the image.

An image inference problem may be expressed by writing a joint probability distribution over the variables. To do this we need to define two "compatibility" functions: $\Phi(X_i, Y_i)$ and $\Psi(X_i, X_j)$. The first relates the observed and hidden variables, so we could think of it as a conditional probability $P(X_i|Y_i)$, for example the probability of a region label given the pixel value. The second expresses the relationship between the pixels, so in the case of pixels that are not adjacent it will be one (expressing no information), and for pixels connected in the grid it can be again thought of as a conditional probability, for example the probability of one region label given the adjacent region label. To get the joint probability we simply take the product (as usual) to get:

$$P(X,Y) = \frac{1}{Z}\prod_{i,j}\Psi(X_i,X_j)\prod_i\Phi(X_i,Y_i)$$

where $X = (X_1, X_2, ..X_n)$ is the set of hidden variables and $Y = (Y_1, Y_2, ..Y_n)$ is the set of observed variables. Note that this is a scalar equation equivalent to the joint probability equations we used for Bayesian networks. It is another factorisation of a joint probability distribution, and its factor graph is shown in Figure 8. As before the value of Z is chosen to normalise the joint probability distribution, and is found by summing the products over the entire set of values that the variables can take. Cleary this will create a massive computational demand in the case of an image, and for that reason iterative estimates of the optimal X_i values are the only possible solution option. Here we see the advantage of the Markov assumption, since the variable X_i depends only on its four neighbours and on its corresponding evidence node.

Belief Propagation

We have already discussed belief propagation in Bayesian networks extensively in terms of the passing of λ and π messages between connected nodes, and we have observed that in a singly connceted network the propagation is very fast in reaching the correct posterior probability distributions of the unknown variables. The process can be defined for other graphical models, and to illustrate this we will look at the pairwise Markov random field. We will first define a message from variable X_i to variable X_j which will have a dimension equal to the states of X_j and will express the evidence that X_i holds for the states of X_j. This message is exactly the same as the λ message that we used extensively in Pearl's propagation algorithm.

The belief in a variable is the same as the posterior probability over the states of a variable in a Bayesian network. For any one state of variable X_i we can write it using the scalar equation:

$$b(X_i(s)) = \frac{1}{Z}\Phi(X_i(s),Y_i)\prod_{k\epsilon N(i)} m_k(X_i(s))$$

where $X_i(s)$ means state s of variable X_i, $m_k(X_i(s))$ is the message (or evidence) from variable X_k for state s of variable X_i, $N(i)$ is the set of variables adjacent to variable X_i, and Z is the normalisation constant that makes the belief a probability distribution over the states of the variable.

The message sent from one variable to another depends on the (unnormalised) belief that the sending variable has received from all variables except the recipient (equivalent to the π messages in Pearl's propagation algorithm). We can write this in scalar form as:

$$b_{\backslash j}(X_i(s)) = \Phi(X_i(s),Y_i)\prod_{k\epsilon N(i)\backslash j} m_k(X_i(s))$$

where the notation $\backslash j$ means "excluding variable j". If we use bold face $(\mathbf{b}_{\backslash\mathbf{j}}(X_i))$ to indicate the vector of beliefs for the states of a variable we can then define the message, in vector form, from variable X_i to X_j as:

$$\mathbf{m_i}(X_j) = \mathbf{b}_{\backslash\mathbf{j}}(X_i)\ \Psi(X_i,X_j)$$

Belief propagation is a generalisation of Pearl's propagation algorithm in Bayesian networks. The notion of causality has been dropped, so we have only one type of message which is computed equivalently to the π message. The compatibility matrix $\Psi(X_i, X_j)$ can be thought of as a joint probability matrix. In vision applications the X_i variables all have the same states, and so the matrix is symmetric and messages can be passed in either direction using the same matrix equation.

Loopy Propagation

When we discussed Pearl's propagation algorithm we noted that it worked for singly connected networks, and for networks with sufficient instantiated nodes to block any loop. Moreover in this case the propagation was very fast. For networks where the singly connected criterion was not met we could solve the problem by converting the network into a join tree, which is a singly connected Markov random field. Now, looking at the pairwise Markov random field we may wonder how the propagation can work in a vision application since there are multiple loops. The answer is that it is not guarenteed to work, but will do so in many cases. It has been shown that belief propagation with loops is equivalent to an optimisation problem in statistical Physics. There are three possible outcomes of loopy propagation. Firstly that the process converges to the optimal result, second that it converges but to a secondary maximum and thirdly that is does not converge but cycles. Computer vision algorithms based on pairwise Markov random fields are strongly constrained by the evidence variables Y_i (pixel intensities) which are constants while the message passing takes place. They therefore tend to converge in a stable manner. Bayesian networks are not necessarily so strongly constrained. Therefore their behaviour in loopy propagation is data dependent and harder to determine.

Summary

Graphical models in probabilistic inference express local properties that allow us to calculate posterior probability distributions in a reasonable time frame. For any set of variables the probability distribution over an unknown can always be calculated by multiplication and marginalisation of the joint probability distribution. Thus if we want to compute a distribution of one variable, say X_n out of a set $(X_1, X_2, ..X_n)$ we can do so by first multiplying the joint probability distribution by the values of any measured variable, and then marginalising using the equation:

$$P(X_n) = \sum_{X_1} \sum_{X_2} .. \sum_{X_{n-1}} P(X_1, X_2, X_3 ... X_N)$$

However, for any problem other than very small ones this is computationally intractible as the size of the state space explodes. Graphical models allow us to decompose large problems into smaller ones by invoking local properties. In the case of Bayesian networks these local properties are the conditional independencies among variables which allows us to express the joint probabilility distribution as a product of several conditional probability distributions. Similarly the join tree algorithm discovers a different factorisation of the joint probability distribution that always results in a singly connected network. However this is at the expense of having fewer factors and therefore greater computational demands to find an individual variable's posterior distribution. The pairwise Markov random field for making inferences about images exploits the local coherence implicit in images. As each of these methods is a different form of factorisation, the factor graph can represent them all, and is (arguably) a simpler and clearer model than the Bayesian network, though its simplifications come at the cost of loosing an explicit representation of cause.

Graphical models are used extensively in other fields such as bio-informatics where it is possible to describe the behaviour of, for example, large numbers of genes involved in cell regulation as a collection of random variables with sparse dependencies. Another place where graphical models are used is neural circuitry, and inference using aggreagates of spiking neurones is currently the subject of extensive research.