Statistical Techniques in Robotics (16-831, F10) Lecture#8 (Thursday September 26)

Inference in Gibbs Fields

Lecturer: Drew Bagnell *Scribe: Jiaji Zhou*

1 Problems for Inference

Following are the inference questions we would like to have answered while examining a Gibbs' field:

Consider binary vector state first:

- What is the mostly likely state? i.e. compute

$$\arg\max_{\mathbf{x}} p(\mathbf{x}),$$

where $\mathbf{x}$ is a vector of the random variables representing the states.

Some examples where the maximum probability is used are: obtaining likely segmentation of an image, and computing the most probable map in a mapping problem.

- What is $p(\mathbf{x})$ for some $\mathbf{x}$?

This problem is hard because of the normalizer. Let's say we had the Gibbs' field of the type shown in Figure 1. The probability of a particular pattern $\mathbf{x}_0$ is computed as:

$$P(\mathbf{x}_0) = \frac{1}{Z} \exp\left(-\sum_{ij} f_{ij}(x_{0i}, x_{0j}) \right).$$

The normalizer Z is computed as the sum of the numerator over all possible configurations of all the x's.

$$Z = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} \exp\left(-\sum_{ij} f_{ij}(x_i, x_j) \right). \quad (1)$$

If the x's are binary, evaluation of Z takes 2^n amount of work. In general for discrete random variables, the evaluation will take p^n work where p is the number of values that the random variable can take.

- Marginals: $p(x_1)$ Some examples where marginals are used are: in speech reconstruction (to find the probability of a particular word), or depth reconstruction (probability of individual depths at various pixels).

To compute the marginal of x_1, we sum the probability distribution over all variables except x_1. Again, for binary variables, this takes $O(2^n)$ work.

$$p(x_1) = \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} p(\mathbf{x}). \quad (2)$$

[0]Some content (mainly for shortest path analogy and gibbs sampling) adapted from previous scribe: Natasha Kholgade

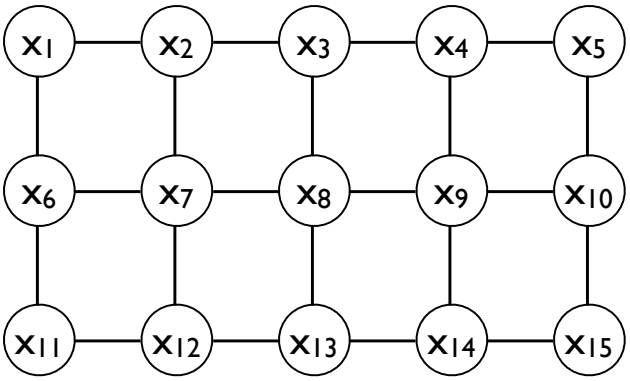

Figure 1: Gibbs' field with maximum clique size 2.

- Expected value of a function g under p, i.e. $\mathbb{E}_p g(\mathbf{x})$, which we could compute by drawing samples from p.

  An instance where this may be used is: g is a function describing a room, what is the probability of the robot being in the room?

The following sections address these questions.

2 Argmax

2.1 Chains

Computing the most probable state is easy for a chain. The computation of maximum probability for all other Gibbs' fields is a generalization of the case for the chain.

Let us consider the chain in Figure 2. For convenience, we shall drop the negative sign, and assume it is incorporated within the f's. We are interested in computing the following maximum:

$$\max_{x_1} \max_{x_2} ... \max_{x_n} \left(\frac{1}{Z} \exp \left(\sum_{ij} f_{ij}(x_i, x_j) \right) \right).$$

The maximizers will be unchanged if we take the log of the probability and drop Z (since it is constant). For the chain, we can rewrite the maximum after taking the log and removing Z as:

$$\max_{x_1} \max_{x_2} \max_{x_3} \max_{x_4} \max_{x_5} \left(f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_3, x_4) + f_4(x_4, x_5) \right). \tag{3}$$

If the x's are binary, computing 3 as it is takes 2^5 work. We would need to enumerate all possible values of the x's, compute the sum on the right, and choose the maximum of 32 sums. This can

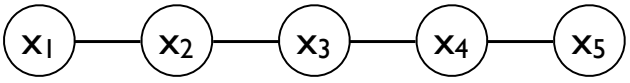

Figure 2: Chain with 5 nodes.

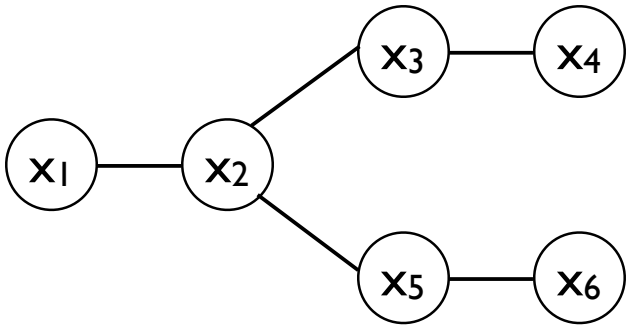

Figure 3: Gibbs' field with tree structure.

get prohibitively expensive for large chains. We can reduce the work to linear by 'pushing the max to the right' using the **generalized distributive law**. This particular version of the generalized distributive law is called the max-sum version (though it may also be max-product if the original probability distribution is retained without taking logs).

Using the generalized distributive law, we rewrite 3 as:

$$\max_{x_1} \max_{x_2} \left(f_1(x_1, x_2) + \max_{x_3} \left(f_2(x_2, x_3) + \max_{x_4} \left(f_3(x_3, x_4) + \max_{x_5} \left(f_4(x_4, x_5) \right) \right) \right) \right). \tag{4}$$

We can represent the last term as a function of x_4:

$$q_4(x_4) = \max_{x_5} \left(f_4(x_4, x_5) \right).$$

The actual computation for $q(x_4)$ is done as follows: for each value of x_4 (i.e. for $x_4 = 0$ and $x_4 = 1$), obtain the value of x_5 that maximizes $f_4(x_4, x_5)$, and store the maximizing values of x_5 for both values of x_4 together with the corresponding value of $q_4(x_4)$.

Next, we can represent the terms in x_3 and x_4 as a function of x_3:

$$q_3(x_3) = \max_{x_4} \left(f_3(x_3, x_4) + q(x_4) \right).$$

Just like before, for every value of x_3, compute the value of x_4 that maximizes the right side of the above expression. This time, while testing different values of x_4 to check for the maximum, perform a lookup for $q_4(x_4)$ cached from the previous step.

Repeat the above step all the way back to x_1. Now move in the forward direction: given the value of x_1 that corresponds to $q_1(x_1)$ (which will in fact be the maximum we are interested in), figure

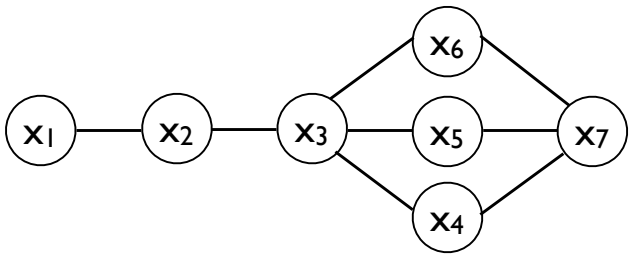

Figure 4: Non-tree Gibbs' field.

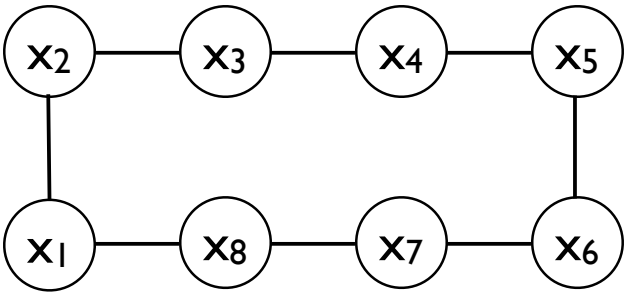

Figure 5: Cyclical Gibbs' field.

out what is the corresponding cached value of x_2 and work forward till x_5. This is an instance of dynamic programming: it is used a lot in graphical models to reduce the computation time of inference-related quantities based on the generalized distributive law.

2.1.1 Shortest Path Analogy

Now consider split each x_i into two nodes ($x_i = 0$ and $x_i = 1$), the negative of $f_{ij}(x_i, x_j)$ as weight for directed edges between nodes. Also consider adding a source s connecting to $x_1 = 0$ and $x_1 = 1$ and t connecting to $x_5 = 0$ and $x_5 = 1$, now solving equation (3) equals finding the shortest path from s to t. And since this is a DAG graph, we can perform dynamic programming as described before.

2.2 Generalization of Chains to Trees and Non-trees

The analysis for chains is directly generalizable to trees. If we have a tree as in Figure 3, we can rewrite the maximum as:

$$\max_{x_1}\max_{x_2}\left(f_1(x_1,x_2)+\max_{x_3}\left(f_{23}(x_2,x_3)+\max_{x_4}\left(f_3(x_3,x_4)\right)\right)+\max_{x_5}\left(f_{25}(x_2,x_5)+\max_{x_6}f_5(x_5,x_6)\right)\right).$$

Here, $q(x_6)$ and $q(x_4)$ will contain one term each, $q(x_5)$ and $q(x_3)$ will contain two terms each, and $q(x_2)$ will have three terms. The number of terms of a node is equal to its in-degree.

Non-trees are more complicated. For the example in Figure 4, we can write the maximum as:

$$\max_{x_1}\max_{x_2}\left(f_1(x_1,x_2)+\max_{x_3}\left(f_2(x_2,x_3)+\max_{x_4}\max_{x_5}\max_{x_6}\left(f_{34}(x_3,x_4)+f_{36}(x_3,x_6)+f_{35}(x_3,x_5)\right.\right.\right.$$
$$\left.\left.\left.+\max_{x_7}\left(f_6(x_6,x_7)+f_4(x_4,x_7)+f_5(x_5,x_7)\right)\right)\right)\right).$$

While taking the maximum over x_4, x_5, and x_6, we still have to do $O(2^3)$ work. This is akin to collapsing x_4, x_5, and x_6 into one giant node which can take on 2^3 values; we now need to test each of the 2^3 values to obtain the maximum. It would be nice to know the upper bound work done to compute the maximum; however, computing this upper bound is in itself an $NP-$hard problem! Thus, people use several tricks for non-tree graphs to reduce computation time.

Cut-set conditioning: For cycles like the one in Figure 5, we can perform the computation by breaking the link at a particular node (say, x_1). For each value of x_1 (i.e. for $x_1 = 0$ and $x_1 = 1$), we use the chain tricks to figure out the maximum, and then find the maximum of the two cases.

3 Marginals

The same generalized distributive law trick applies to computing marginals and Z. In this case it takes the sum-product form. For the chain in Figure 2, we can compute Z by 'pushing over the sum'. Equation 1 can be rewritten as:

$$Z = \sum_{x_1} \sum_{x_2} e^{-f_1(x_1,x_2)} \sum_{x_3} e^{-f_2(x_2,x_3)} \sum_{x_4} e^{-f_3(x_3,x_4)} \sum_{x_5} e^{-f_4(x_4,x_5)}. \tag{5}$$

For each pair of variables, x_i and x_j, we cache the sums $\sum_{x_i} e^{-f_i(x_i,x_j)} + q_j(x_j)$ for each value of x_i, pulling the appropriate value of $q_j(x_j)$ cached in the previous step. Marginals can be computed in the same way.

$$p(x_1) = \sum_{x_2} e^{-f_1(x_1,x_2)} \sum_{x_3} e^{-f_2(x_2,x_3)} \sum_{x_4} e^{-f_3(x_3,x_4)} \sum_{x_5} e^{-f_4(x_4,x_5)}. \tag{6}$$

4 Ratios

In general it is not easy to compute $p(\mathbf{x})$ because of the number of terms involved, and the normalizer Z. Instead, what is easy is to compute is ratios, especially for graphical models where there is some structure. For instance, let's say we have two configurations for the Gibbs' field in Figure 1: $\mathbf{x}_0 = [0, 0, 0, ...]$ and $\mathbf{x}_1 = [1, 0, 0, ...]$. In the ratio,

$$\frac{p(\mathbf{x}_0)}{p(\mathbf{x}_1)},$$

a number of things cancel out:

- Z cancels out as it is constant for all $p(\mathbf{x})$.
- Any terms in the Gibbs' field which do not involve x_1 cancel out, as their values will be unchanged. Generally, only terms involving 'changing' nodes remain.

These ratios can be used to compute $\max_{\mathbf{x}} p(\mathbf{x})$ in the following way:

- Start with an initial guess for $\mathbf{x}$.
- Switch a variable at random
- Test the ratio of the current probability with the previous one, and if it is higher, set $\mathbf{x}$ to the new configuration.
- Ascend upwards in this manner till you hit a maximum.

This approach is called coordinate ascent: the variables can be considered coordinates of an N-dimensional space, and we move toward the maximum by changing values along coordinates. The problem with this approach is that you can often hit a local maximum, nevertheless it is a popular approach (often combined with multiple initializations, or some clever strategy to select the initial configuration).

5 Sampling

Samples are valuable because they can be used to compute expected values of functions over a probability distribution. They can also be used to compute marginals by counting. Most sampling techniques use the ratios listed in Section 4 to minimize computation time. This section discusses two techniques: importance sampling and Gibbs' sampling. The latter (or more commonly some variant) is the more frequently used method.

5.1 Importance Sampling

This technique has been discussed earlier in class: We want $\mathbb{E}_p f(\mathbf{x})$, but we can't sample p, so we instead sample $\mathbf{x}_i$'s from a different distribution q and weight each $f(\mathbf{x}_i)$ by $\frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)}$. As far as possible, q should be similar to p.

What distribution should we draw from? We can draw from $q = \Pi q_i(\mathbf{x}_i)$, but this usually does not work since in most cases, it is too far from the actual distribution p. It is easier to sample from chains and trees, so often if a non-tree graph is almost a tree, we can convert it to a tree by breaking some links, sample from the new tree-based distribution, and use importance sampling to compute the expectations for the non-tree graph. Importance sampling usually only works for graphs that are nearly trees: in most examples that are of interest, this is never the case.

5.2 Gibbs' Sampling

Consider the Gibbs' field in Figure 1. Gibbs' sampling works as follows:

1. Start with some initial configuration (say $[0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1]$ in the order x_1 to x_{15}).
2. Pick a node at random (say x_7).
3. Resample the node's probability conditioned on its Markov blanket: generate $x_7^+ \sim p(x_7|N(x_7))$, i.e. flip a coin in proportion to the likelihood of getting a 0 or a 1 given the neighbors.
4. Repeat steps 3 and 4 for T time steps.

For a large enough T, $\mathbf{x}_t$ converges to a sample from $p(\mathbf{x})$. To get the next sample, it is necessary to wait another T time steps, as updates too close to each other may be correlated. If you are computing the expectation of a function under p, it is more advantageous to do that directly over the updates, than to wait for samples: for large enough T, $\frac{1}{T}\sum_t f(\mathbf{x}_t)$ converges to $\mathbb{E}_p f(\mathbf{x})$. Note that in generally x does not converge, it is the expection that converges

5.3 Doubts and Answers

You may have the following doubts as I doubted before:

- How could the expection converge? Think of the problem in 2D case ($x = [x_1, x_2]$). Pick out all those steps that we perform resampling on x_0 while fixing x_1, look at all these x_0 and you will find out they will converge to as if sampled from the marginal probablity $p(x_0)$, and so as x_1.
- It is not iid sampled, why does it work? For adjacent steps it is not iid sampled, but if you look at every K steps (generally K cannot be too small), it is approximately iid sampled.