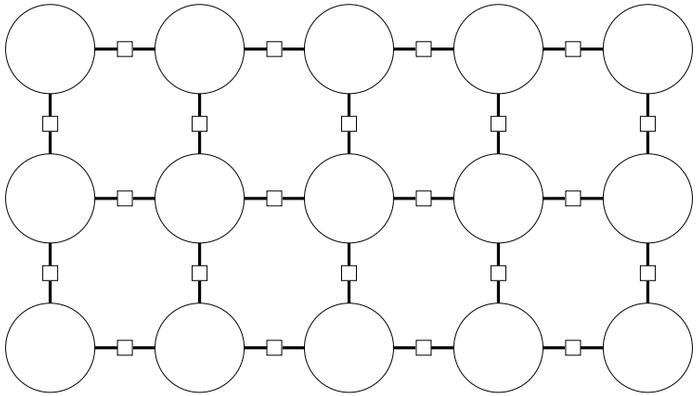
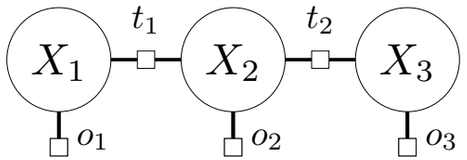


Markov networks: Gibbs sampling

- In this module, I will present Gibbs sampling, a simple algorithm for approximately computing marginal probabilities.

Review: Markov networks



Definition: Markov network

A Markov network is a factor graph which defines a joint distribution over random variables $X = (X_1, \ldots, X_n)$:

$$\mathbb{P}(X = x) = \frac{\text{Weight}(x)}{Z}$$

where $Z = \sum_{x'} \text{Weight}(x')$ is the normalization constant.

Objective: compute marginal probabilities $\mathbb{P}(X_i = v) = \sum_{x: x_i = v} \mathbb{P}(X = x)$

x_1	x_2	x_3	Weight(x)	$\mathbb{P}(X = x)$
0	1	1	4	0.15
0	1	2	4	0.15
1	1	1	4	0.15
1	1	2	4	0.15
1	2	1	2	0.08
1	2	2	8	0.31

$$Z = 4 + 4 + 4 + 4 + 2 + 8 = 26$$

$$\mathbb{P}(X_2 = 1) = 0.15 + 0.15 + 0.15 + 0.15 = 0.62$$
$$\mathbb{P}(X_2 = 2) = 0.08 + 0.31 = 0.38$$

- Recall that a Markov network is defined by a factor graph, which provides a non-negative weight to each assignment x.
- If we compute the normalization constant Z and divide, then we get a probability distribution over joint assignments.
- For the object tracking example, we can compute the normalization factor to get joint probabilities. Note that this gives us a notion of uncertainty over the possible assignments.
- The main objective after defining a joint distribution is to compute marginal probabilities, which allow us to ask pointed questions about variables (e.g., X_2). Marginal probabilities are computed by summing the probabilities over all assignments satisfying the given condition.

Gibbs sampling

Algorithm: Gibbs sampling

Initialize x to a random complete assignment

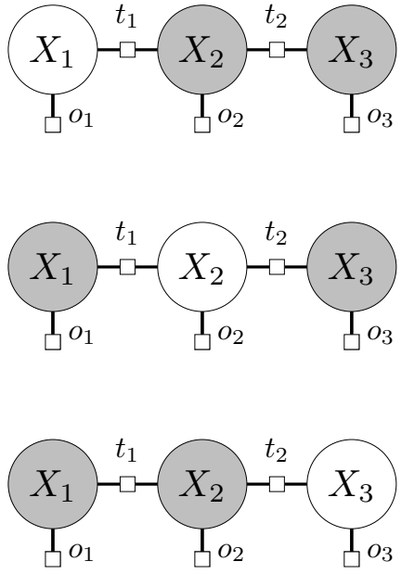
Loop through $i = 1, \ldots, n$ until convergence:

Set $x_i = v$ with prob. $\mathbb{P}(X_i = v \mid X_{-i} = x_{-i})$

(X_{-i} denotes all variables except X_i)

Increment $\text{count}_i(x_i)$

Estimate $\hat{\mathbb{P}}(X_i = x_i) = \frac{\text{count}_i(x_i)}{\sum_v \text{count}_i(v)}$



Example: sampling one variable

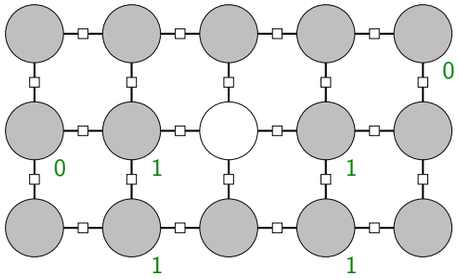
$\text{Weight}(x \cup \{X_2 : 0\}) = 1$	prob. 0.2
$\text{Weight}(x \cup \{X_2 : 1\}) = 2$	prob. 0.4
$\text{Weight}(x \cup \{X_2 : 2\}) = 2$	prob. 0.4

[demo]

- Now we present Gibbs sampling, a simple algorithm for approximately computing marginal probabilities. The algorithm follows the template of local search, where we change one variable at a time, but unlike Iterated Conditional Modes (ICM), Gibbs sampling is a randomized algorithm.
- Gibbs sampling proceeds by going through each variable X_i, considering all the possible assignments of X_i with some $v \in \text{Domain}_i$, and setting $X_i = v$ with probability equal to the conditional probability of $X_i = v$ given everything else.
- To perform this step, we can rewrite this expression using laws of probability: $\mathbb{P}(X_i = v \mid X_{-i} = x_{-i}) = \frac{\text{Weight}(x \cup \{X_i : v\})}{Z \mathbb{P}(X_{-i} = x_{-i})}$, where the denominator is a new normalization constant. We don't need to compute it directly. Instead, we first compute the weight of $x \cup \{X_i : v\}$ for each v, and then normalize to get a distribution. Finally we sample a v according to that distribution.
- Along the way, for each variable X_i that we're interested in tracking, we keep a counter $\text{count}_i(v)$ of how many times we've seen $X_i = v$. These counts can be normalized at any time to produce an estimate $\hat{\mathbb{P}}(X_i = x_i)$ of the marginal probability.

Application: image denoising

Example: image denoising



- $X_i \in \{0, 1\}$ is pixel value in location i
- Subset of pixels are observed

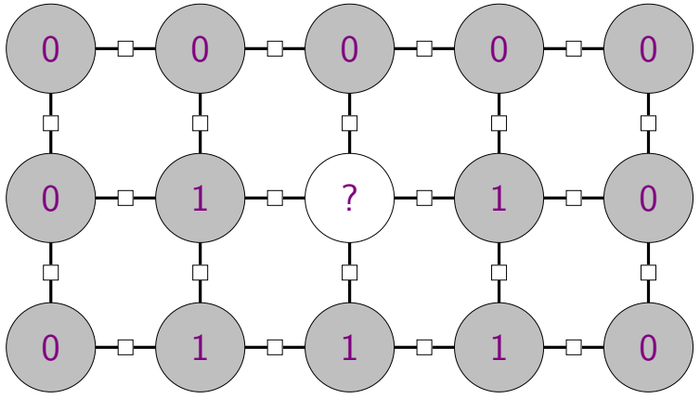
$o_i(x_i) = [x_i =$ observed value at $i]$

- Neighboring pixels more likely to be same than different

$t_{ij}(x_i, x_j) = [x_i = x_j] + 1$

- Let's apply Gibbs sampling to the image denoising application.
- Recall that we have a grid of pixels, a subset of which are observed, and we wish to fill in the remaining pixels.
- The unknown pixels are represented by a variable X_i for each pixel i. We have observation factors can constrain the observed pixels, and transition factors that encourage neighboring pixels to agree.

Gibbs sampling for image denoising



$$t_{ij}(x_i, x_j) = [x_i = x_j] + 1$$

Scan through image and update each pixel given rest:

v	weight	$\mathbb{P}(X_i = v \mid X_{-i} = x_{-i})$
0	$2 \cdot 1 \cdot 1 \cdot 1$	0.2
1	$1 \cdot 2 \cdot 2 \cdot 2$	0.8

- Let us compute the Gibbs sampling update. We go through each pixel X_i and try to update its value.
- For the given example, we consider both values 0 and 1, and multiply exactly the transition factors that depend on that value. Assume there are no observation factors here.
- The factor returns 2 if the pixel values agree and 1 if they disagree.
- We then normalize the weights to form a distribution and then sample v.
- Intuitively, the neighbors are all trying to pull $X_{(3,2)}$ towards their values, and 0.8 reflects the fact that the pull towards 1 is stronger.

Image denoising demo

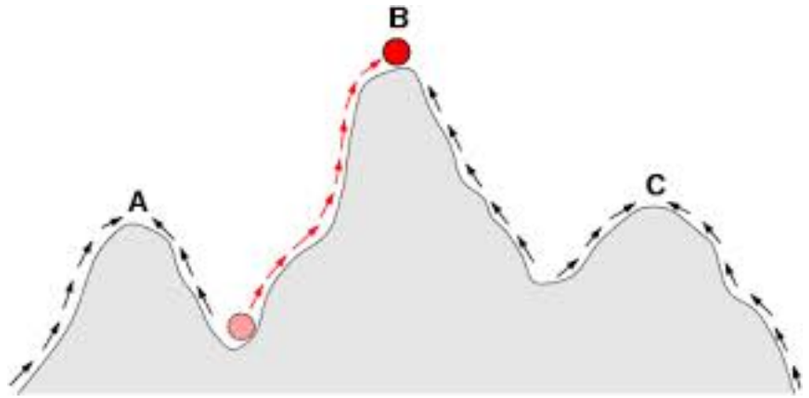
[see web version]

- Let's actually play around with Gibbs sampling for image denoising in the browser.
- Try playing with the demo by modifying the settings to get a feeling for what Gibbs sampling is doing. Each iteration corresponds to resampling each pixel (variable).
- When you hit ctrl-enter for the first time, red and black correspond to 1 and 0, and white corresponds to unobserved.
- `showMarginals` allows you to either view the assignments produced or the marginals estimated from the particles (this gives you a smoother probability estimate of what the pixel values are).
- If you decrease `missingFrac` to 0.3, the problem becomes easier, and the reconstruction looks pretty good.
- If you set `coherenceFactor` to 10, then there will be coupling between neighboring variables, and you'll see sharper lines, although the reconstruction is not perfect.
- If you set `icm` to true, we will use local search rather than Gibbs sampling, which produces very bad solutions.

Search versus sampling

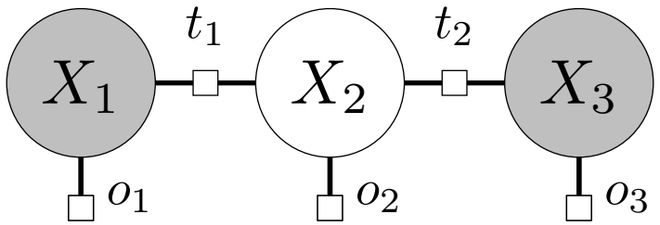
Iterated Conditional Modes	Gibbs sampling
maximum weight assignment	marginal probabilities
choose best value	sample a value
converges to local optimum	marginals converge to correct answer*

*under technical conditions (sufficient condition: all weights positive), but could take exponential time



- It is instructive to compare Gibbs sampling with its cousin, Iterated Conditional Modes (ICM). Both iteratively go through the variables and tries to update each one of them holding the others fixed.
- Recall that the goals are different: ICM tries to find the maximum weight assignment while Gibbs sampling is trying to compute marginal probabilities.
- Accordingly, ICM will choose the value for a variable X_i with the highest weight, whereas Gibbs sampling will use the weights to form a distribution to sample from.
- ICM converges to local optimum, an assignment that can't be improved on. Note that Gibbs sampling is stochastic so in some sense never converges. However, the estimates of hte marginal probabilities do in fact converge under some technical assumptions. The simplest sufficient condition if all weights are positive, but it also suffices that the probability of Gibbs sampling going between any two assignments is positive. A major caveat is that the time it takes to converge can be exponential in the number of variables.
- Advanced: Gibbs sampling is an instance of a Markov Chain Monte Carlo (MCMC) algorithm which generates a sequence of particles $X^{(1)}, X^{(2)}, X^{(3)}, \ldots$. A Markov chain is irreducible if there is positive probability of getting from any assignment to any other assignment (now the probabilities are over the random choices of the sampler). When the Gibbs sampler is irreducible, then in the limit as $t \to \infty$, the distribution of $X^{(t)}$ converges to the true distribution $\mathbb{P}(X)$. MCMC is a very rich topic which we will not talk about very much here.

Summary



- Objective: compute marginal probabilities $\mathbb{P}(X_i = x_i)$
- Gibbs sampling: sample one variable at a time, count visitations
- More generally: Markov chain Monte Carlo (MCMC) powerful toolkit of randomized procedures

- In summary, we are trying to compute the marginal probabilities of a Markov network.
- Gibbs sampling allows us to do this by exploring all the assignments randomly, but very carefully controlled probabilities, so that the visitation frequencies of various values converge to the right answer.
- Gibbs sampling is part of a beautiful and rich set of tools for using randomness to do inference on Markov networks, which I encourage you to check out.