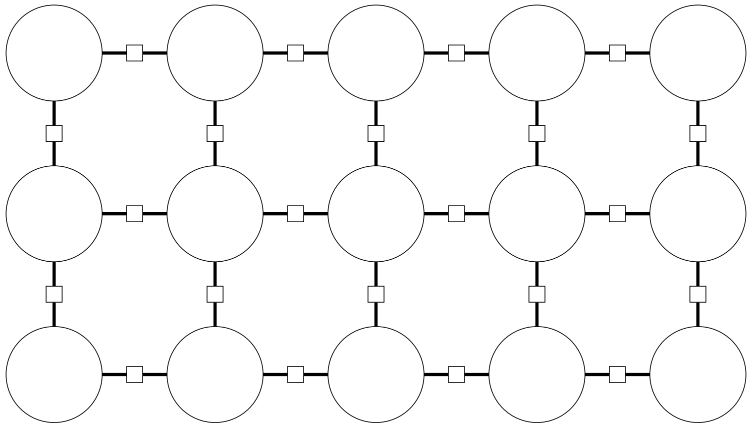
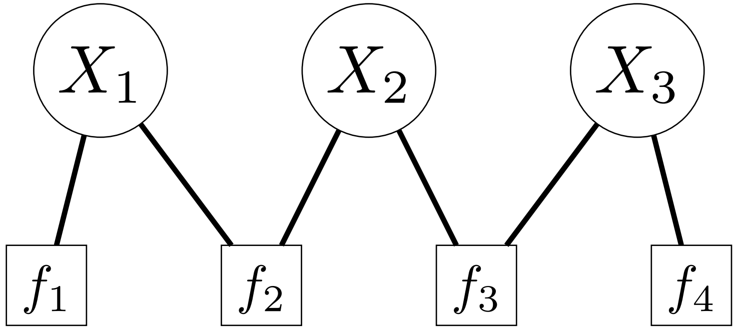
Markov networks: overview



☆ connects factor graphs w/ probability

Review: factor graphs



Definition: factor graph

Variables:

$X = (X_1, \dots, X_n)$, where $X_i \in \text{Domain}_i$

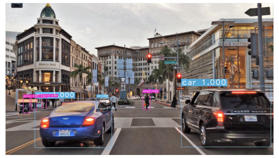
Factors:

$f_1, \dots, f_m$, with each $f_j(X) \geq 0$

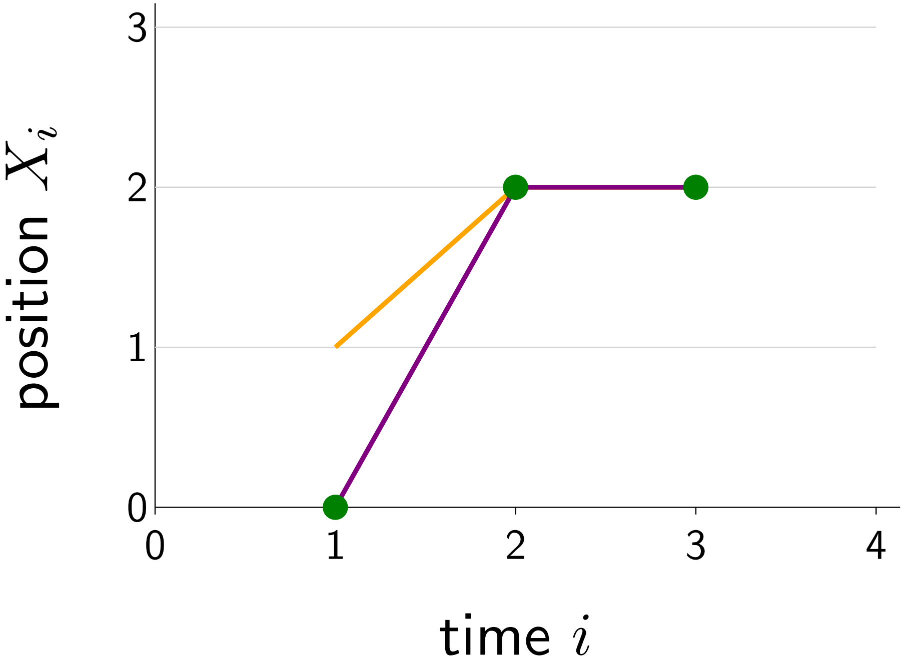
Definition: assignment weight

Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

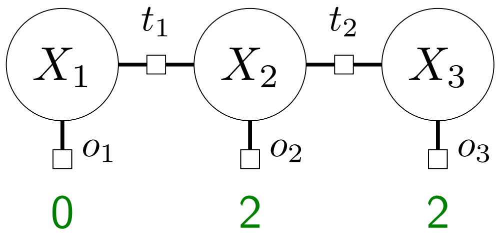
$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$



Example: object tracking



factor graph



x_1	$o_1(x_1)$
0	2
1	1
2	0

x_2	$o_2(x_2)$
0	0
1	1
2	2

x_3	$o_3(x_3)$
0	0
1	1
2	2

$\lvert x_i - x_{i+1}\rvert$	$t_i(x_i, x_{i+1})$
0	2
1	1
2	0

[demo]

Maximum weight assignment

CSP objective: find the maximum weight assignment

$$\max_x \mathsf{Weight}(x)$$

x_1	x_2	x_3	Weight(x)
0	1	1	4
0	1	2	4
1	1	1	4
1	1	2	4
1	2	1	2
1	2	2	8

Maximum weight assignment: $\{x_1 : 1, x_2 : 2, x_3 : 2\}$ (weight 8)

But this doesn't represent all the other possible assignments...

no uncertainty in assignments

Definition

Definition: Markov network

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

prob of assignment x

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

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

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

only 31% certain, even tho it's the max assignment

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

Represents uncertainty!

Marginal probabilities

Example question: where was the object at time step 2 (X_2)?

Definition: Marginal probability

The marginal probability of $X_i = v$ is given by:

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

sum over possible complete assignments s.t. $x_i = v$

Object tracking example:

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

$$\mathbb{P}(X_2 = 1) = 0.15 + 0.15 + 0.15 + 0.15 = 0.62$$

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

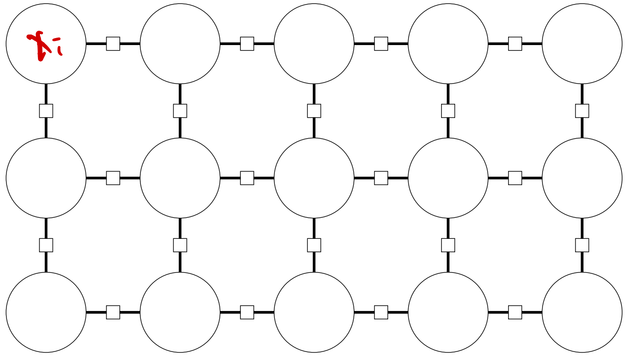
Note: different than max weight assignment!

at step 2, most likely to be at position 1

Application: Ising model

Ising model: classic model from statistical physics to model ferromagnetism

markov network



$X_i \in \{-1, +1\}$: atomic spin of site i

$f_{ij}(x_i, x_j) = \exp(\beta x_i x_j)$ wants same spin

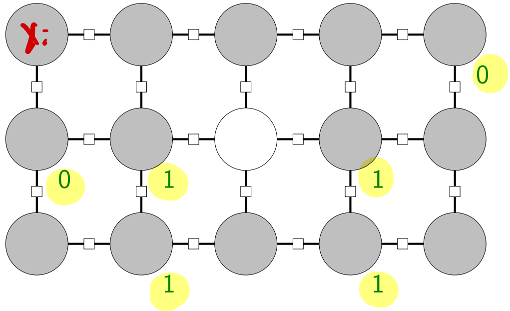
strength of affinity between x_i & x_j

Samples as β increases:

Figure 2 from Perez (1998)

Application: image denoising

Example: image denoising



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

$o_i(x_i) = [x_i = \text{observed value at } i]$

- Neighboring pixels more likely to be same than different

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

Summary

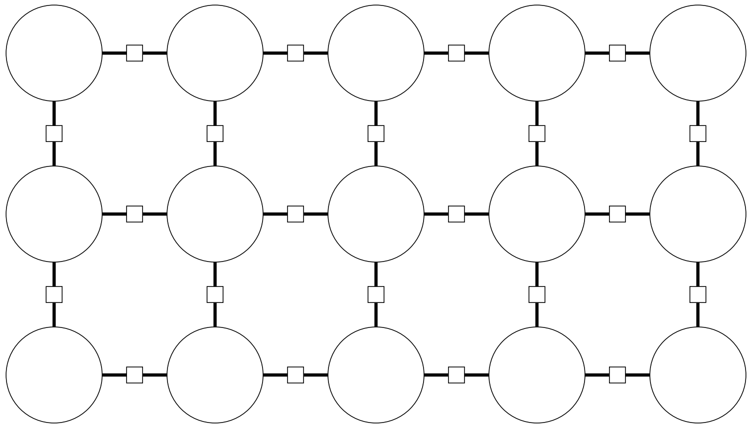
Markov networks = factor graphs + probability

- Normalize weights to get probablity distribution
- Can compute marginal probabilities to focus on variables

	CSPs	Markov networks
Unknown:	variables	random variables
	weights	probabilities
Objective:	maximum weight assignment	marginal probabilities

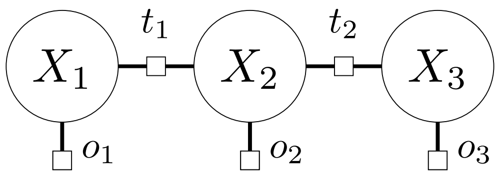
finding

Markov networks: Gibbs sampling



☆ approximately computing marginal probs

Review: Markov networks



Definition: Markov network

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

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

→ convert weight to prob

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

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

$\mathbb{P}(X_2 = 1) = 0.15 + 0.15 + 0.15 + 0.15 = 0.62$

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

Gibbs sampling

→ similar to local search

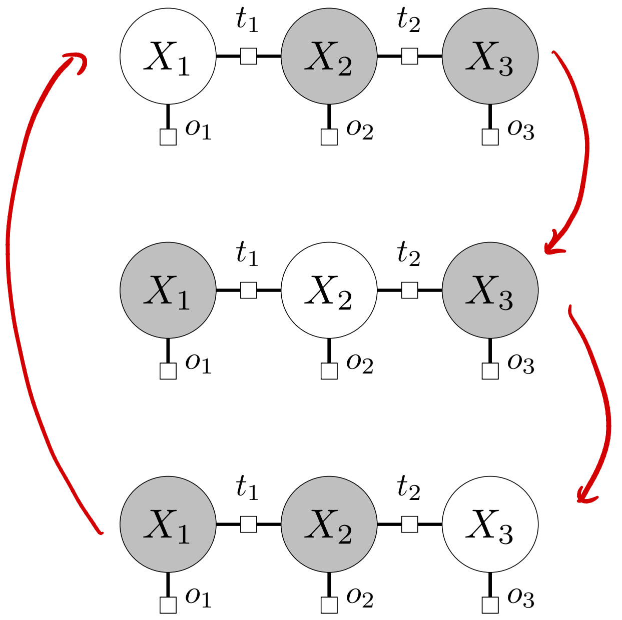
Algorithm: Gibbs sampling

Initialize x to a random complete assignment

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

- Set $x_i = v$ with prob. $\mathbb{P}(X_i = v \mid X_{-i} = x_{-i})$ → use weights to form distribution, sample v according to that
  - (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

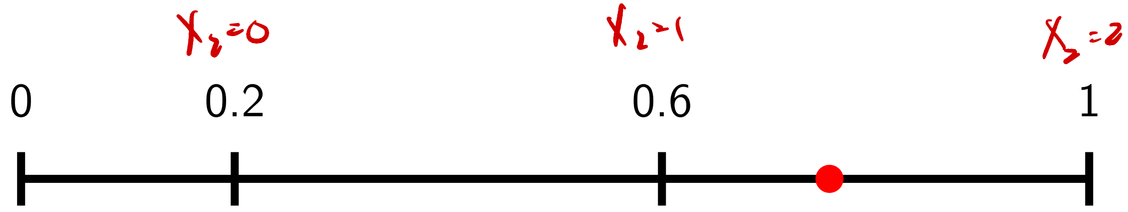
try assigning $X_i = v$

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

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

Weight$(x \cup \{X_2 : 2\}) = 2$ → prob. 0.4

normalize

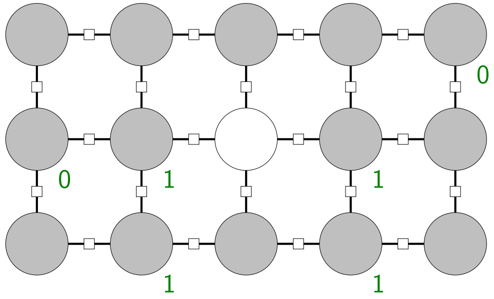


[demo]

- 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



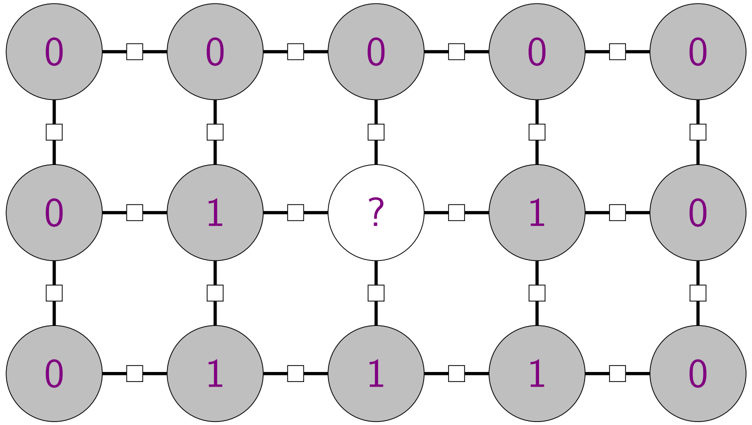
- $X_i \in \{0, 1\}$ is pixel value in location i
- Subset of pixels are observed

$o_i(x_i) = [x_i = \text{observed value at } i]$

- Neighboring pixels more likely to be same than different

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

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:

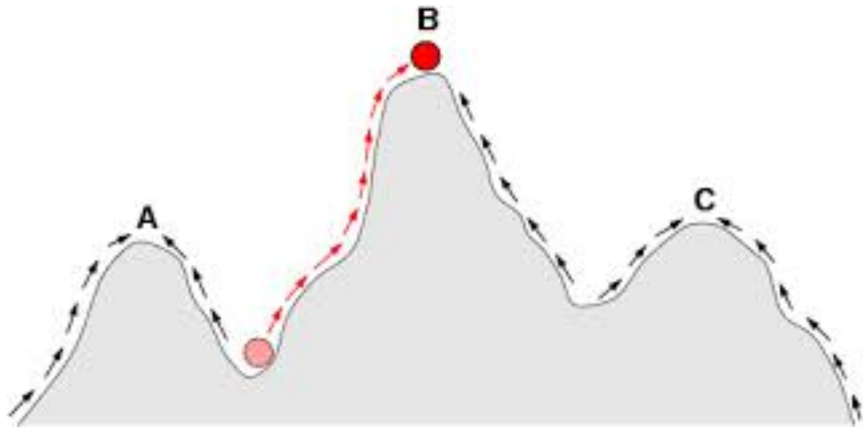
v	weight	$\mathbb{P}(X_i = v \mid X_{-i} = x_{-i})$
0	$2 \cdot 1 \cdot 1 \cdot 1$	0.2
1	$1 \cdot 2 \cdot 2 \cdot 2$	0.8

- 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.

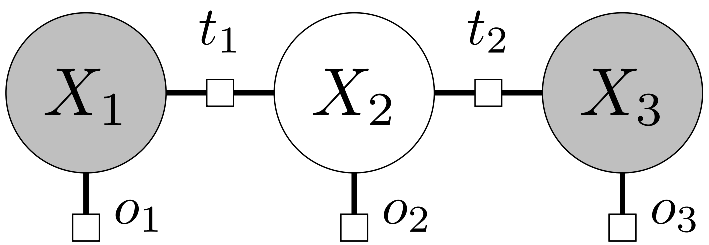
Search versus sampling

	Iterated Conditional Modes	Gibbs sampling	
	maximum weight assignment	marginal probabilities	
at each step:	choose best value	sample a value	samples don't converge
	converges to local optimum	marginals converge to correct answer*	

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



Summary



- Objective: compute marginal probabilities $\mathbb{P}(X_i = x_i)$

- Gibbs sampling: sample one variable at a time, count visitations to values

- More generally: Markov chain Monte Carlo (MCMC) powerful toolkit of randomized procedures