Lecture 5, Inference 1 - Belief Propagation

DS-GA 1005 Inference and Representation, Fall 2023

Yoav Wald

10/04/2023

Outline

- So far we've seen ways to represent distributions in factorized forms
- Factorizations convey statistical independence properties on underlyig distribution
- But, how do we use them and what for?

Reminder: Gibbs distributions

Definition (Gibbs distribution)

Let $\{\Psi_C\}_{C \in \mathcal{C}}$ be a set of non-negative local factors, where $C \subseteq V$ for any $C \in \mathcal{C}$. P is a Gibbs distribution parameterized by the set if

$$P(x_1, \ldots, x_d) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(\mathbf{x}_C),$$

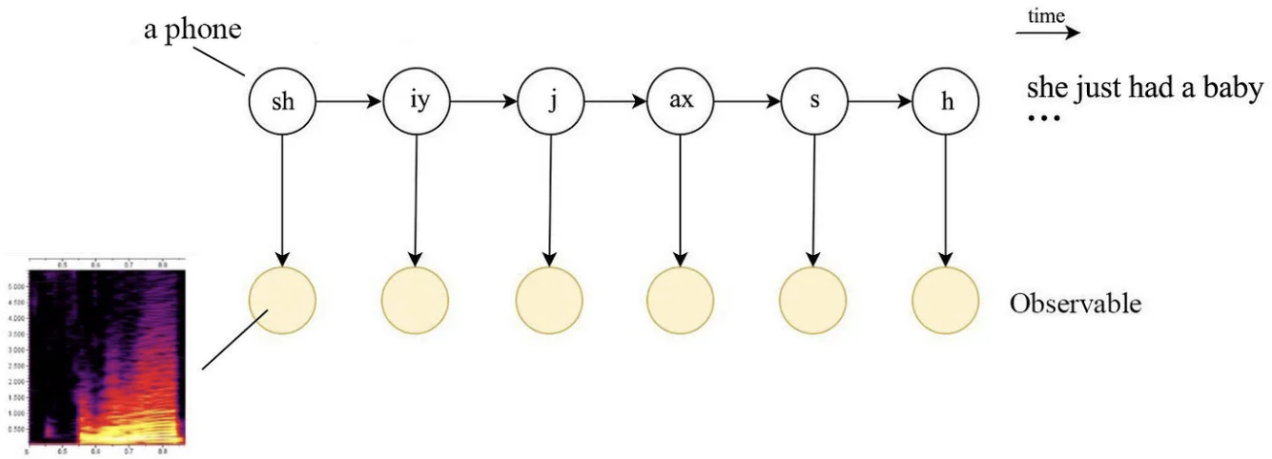
where $Z = \int \prod_{C \in \mathcal{C}} \Psi_C(\mathbf{x}_C) d\mathbf{x}$

Corresponds to undirected models, today we will work with these distributions

- Recall that distributions over Bayesian networks are also Gibbs distributions

Examples: Inference in graphical models

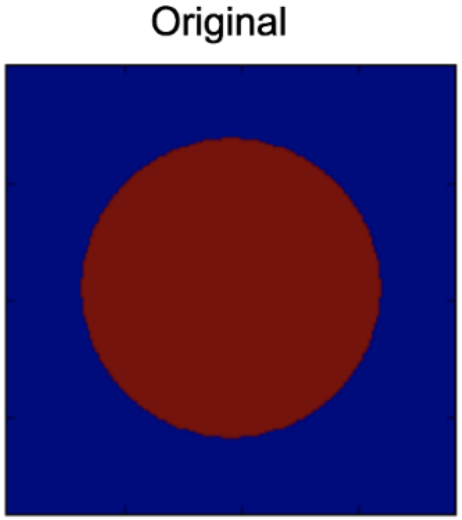
Speech recognition with HMMs

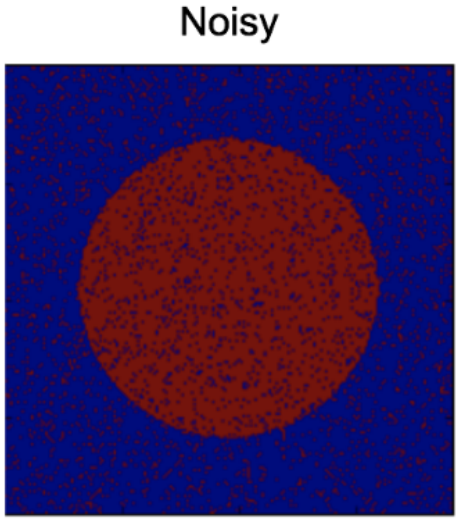


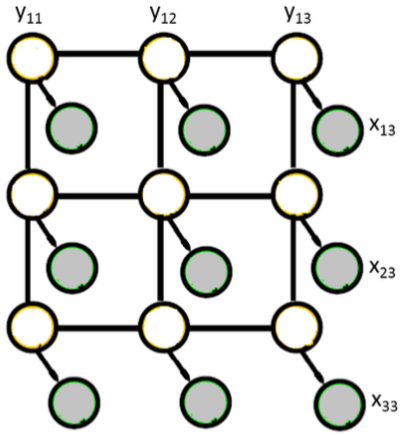
- We are given models $P(W_t \mid W_{t-1})$ (next word predictor), and $P(X_t \mid W_t)$ (sound waveform given word)
- At test time, we observe a recorded signal $\mathbf{X} = X_1, \ldots, X_T$, and wish to recognize the words being uttered
- Reasonable goal: solve $\arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{X})$

Examples: Inference in graphical models

Image denoising with MRFs







X : noisy pixels
Y : "true" pixels

- Variable for each noisy pixel X_i and corresponding clean pixel Y_i; potential $\Psi_i(X_i, Y_i)$ controls pixels' tendency to be similar
- Potentials $\Psi_{ij}(Y_i, Y_j)$ for grid-shape graph, modelling the tendency of nearby pixels to be similar
- As in HMM, wish to infer denoised image $\arg\max_{\mathbf{Y}} P(\mathbf{Y} \mid \mathbf{X})$

Types of Inference Problems

- **Marginal inference**: calculate $P(x_i)$ for some $i \in V$, or $P(\mathbf{x}_C)$ for some $C \subseteq V$
- **Maximium A-Posteriori (MAP) inference:** find $\arg\max_{\mathbf{x}} P(x_1, \ldots, x_d)$
- Tools for marginal inference will also let us handle conditioning since we can divide marginals $P(A \mid B) = \frac{P(A,B)}{P(B)}$

Inference is Hard

- What is the computational complexity of inference (as a function of number of variables d)?

$$P(x_i) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}}, \cdots \sum_{x_d} P(x_1, \ldots, x_d)$$

- Computing marginal is #P-hard, MAP inference is NP-hard
- Are there structures where inference is efficient?

Inference on Pairwise distributions

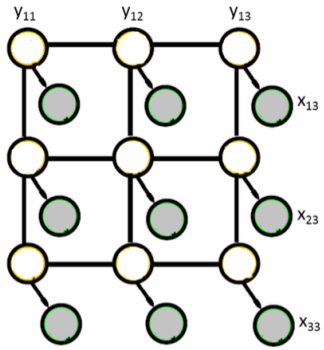
- Gibbs distribution

$$P(x_1, \ldots, x_d) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(\mathbf{x}_C),$$

- For simplicity, let us start with $C = V \cup E$ for an undirected graph $G = (V, E)$

$$P(x_1, \ldots, x_d) = \frac{1}{Z} \prod_{i \in V} \Psi_i(x_i) \prod_{(i,j) \in E} \Psi_{ij}(x_i, x_j)$$

- Inference is still hard, let us start with (arguably) the simplest connectivity structure



X : noisy pixels
Y : "true" pixels

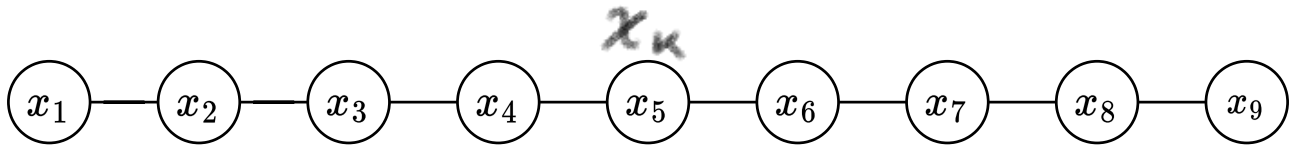
Warmup: Marginal Inference on a Chain

Let us start with the a simple chain graph, and assume variables are binary $\mathbf{x}_i \in \{0,1\} \quad \forall i \in [d]$.

$$P(x_k) = \sum_{\tilde{\mathbf{x}} \in \{0,1\}^d : \tilde{x}_k = x_k} P(\tilde{\mathbf{x}})$$

$$= \frac{1}{Z} \sum_{\tilde{\mathbf{x}} \in \{0,1\}^d : \tilde{x}_k = x_k} \prod_{i \in V} \Psi_i(\tilde{x}_i) \prod_{(i,j) \in E} \Psi_{ij}(\tilde{x}_i, \tilde{x}_j)$$

$$\propto \sum_{\tilde{\mathbf{x}} \in \{0,1\}^d : \tilde{x}_k = x_k} \Psi_1(\tilde{x}_1) \prod_{j=2}^{d} \Psi_j(\tilde{x}_j) \Psi_{j-1,j}(\tilde{x}_{j-1}, \tilde{x}_j)$$

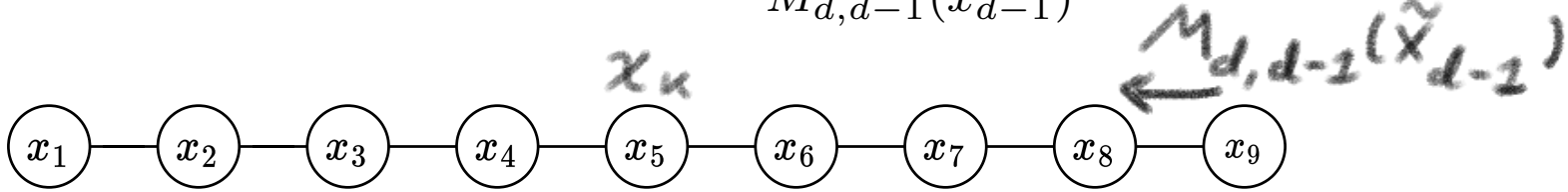


Warmup: Marginal Inference on a Chain

$$P(x_k) \propto \sum_{\tilde{\mathbf{x}} \in \{0,1\}^d : \tilde{x}_k = x_k} \Psi_1(\tilde{x}_1) \prod_{j=2}^{d} \Psi_j(\tilde{x}_j) \Psi_{j-1,j}(\tilde{x}_{j-1}, \tilde{x}_j)$$

$$= \sum_{\tilde{x}_1, \ldots, \tilde{x}_{d-1} : \tilde{x}_k = x_k} \left(\Psi_1(\tilde{x}_1) \prod_{j=2}^{d-1} \Psi_j(\tilde{x}_j) \Psi_{j-1,j}(\tilde{x}_{j-1}, \tilde{x}_j) \right.$$

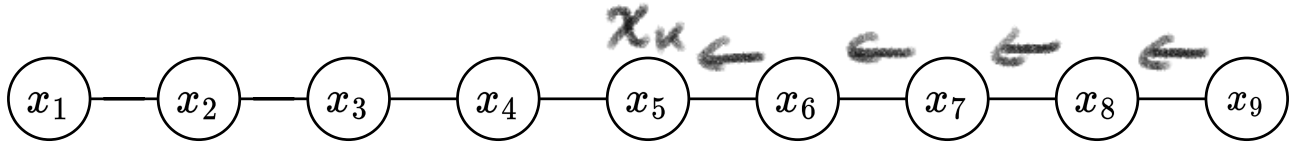
$$\left. \cdot \underbrace{\sum_{\tilde{x}_d} \Psi_d(\tilde{x}_d) \Psi_{d-1,d}(\tilde{x}_{d-1}, \tilde{x}_d)}_{M_{d,d-1}(\tilde{x}_{d-1})} \right)$$



Warmup: Marginal Inference on a Chain

$$P(x_k) \propto \sum_{\tilde{x}_1,\ldots,\tilde{x}_{d-1}:\tilde{x}_k=x_k} \Psi_1(\tilde{x}_1) \prod_{j=2}^{d-1} \Psi_j(\tilde{x}_j)\Psi_{j-1,j}(\tilde{x}_{j-1},\tilde{x}_j) M_{d,d-1}(\tilde{x}_{d-1})$$

- Computational complexity to calculate $M_{d,d-1}(\tilde{x}_{d-1})$ is $O(1)$!
- Define $M_{d-1,d-2}(\tilde{x}_{d-2}) =$
  $\kappa \sum_{\tilde{x}_{d-1}} \left\{ \Psi_{d-1}(\tilde{x}_{d-1})\Psi_{d-2,d-1}(\tilde{x}_{d-2},\tilde{x}_{d-1}) M_{d,d-1}(\tilde{x}_{d-1}) \right\}$
  - κ - normalization constant
- Continue pushing summations inside until we get $M_{i+1,i}(x_i)\ldots$

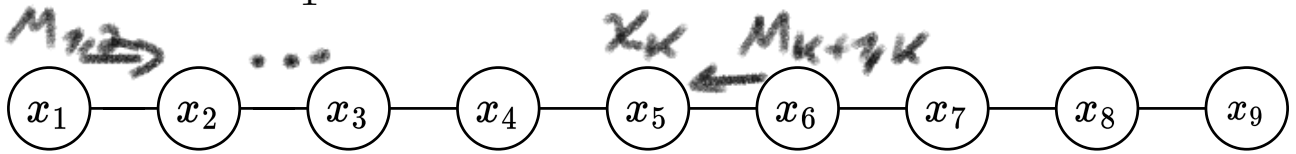


Warmup: Marginal Inference on a Chain

$$P(x_k) \propto \sum_{\tilde{x}_1,\ldots,\tilde{x}_{d-1}:\tilde{x}_k=x_k} \Psi_1(\tilde{x}_1) \prod_{j=2}^{d-1} \Psi_j(\tilde{x}_j)\Psi_{j-1,j}(\tilde{x}_{j-1},\tilde{x}_j) M_{d,d-1}(\tilde{x}_{d-1})$$

$$= \sum_{\tilde{x}_1,\ldots,\tilde{x}_{k-1}} \Psi_1(\tilde{x}_1) \prod_{j=2}^{k} \Psi_j(\tilde{x}_j)\Psi_{j-1,j}(\tilde{x}_{j-1},\tilde{x}_j) M_{k+1,k}(\tilde{x}_k)$$

- So far computational complexity scales as $d - k$
- Now we can do the same from the other side of the chain
- Push summation on x_1 inside and define
  $M_{1,2}(\tilde{x}_2) = \sum_{\tilde{x}_1} \Psi_1(\tilde{x}_1)\Psi_{1,2}(\tilde{x}_1,\tilde{x}_2)$



Warmup: Marginal Inference on a Chain

$$P(x_k) \propto \sum_{\tilde{x}_1,\ldots,\tilde{x}_{k-1}} \Psi_1(\tilde{x}_1) \prod_{j=2}^{k} \Psi_j(\tilde{x}_j)\Psi_{j-1,j}(\tilde{x}_{j-1},\tilde{x}_j) M_{k+1,k}(x_k)$$

$$= \sum_{\tilde{x}_2,\ldots,\tilde{x}_{k-1}} M_{1,2}(\tilde{x}_2)\Psi_2(\tilde{x}_1) \prod_{j=3}^{k} \Psi_j(\tilde{x}_j)\Psi_{j-1,j}(\tilde{x}_{j-1},\tilde{x}_j) M_{k+1,k}(x_k)$$

$$\ldots = M_{k-1,k}(x_k)\Psi_k(x_k)M_{k+1,k}(x_k)$$

- We conclude that $P(x_k) = \kappa M_{k-1,k}(x_k)\Psi_k(x_k)M_{k+1,k}(x_k)$
- Computational complexity is $O(d)$!

Marginal Inference on Trees

Let us generalize the algorithm

- *Observation 1*: Trivial to generalize beyond binary variables, simply sum/integrate over all $Val(X_j)$ instead of $\{0, 1\}$
- *Observation 2*: Switching order of sums and products will work as long as there is no cycle in the graph

Marginal Inference on Trees

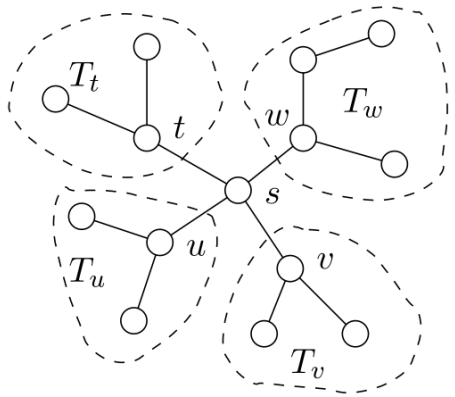
Switching order of sums and products works as long as G is a tree

- If G is a tree and we want $P(x_s)$, set s as the *root*
- For each $i \in V$ that is leaf in this tree calculate

$$M_{i,pa(i)}(x_{pa(i)}) = \sum_{\tilde{x}_i} \Psi_i(\tilde{x}_i)\Psi_{i,pa(i)}(\tilde{x}_i, x_{pa(i)})$$

- Next, let v be a leaf's parent, then send a messages to its own parent s and continue until we're done ...

$$M_{v,s}(x_s) = \sum_{\tilde{x}_v} \Psi_v(\tilde{x}_v)\Psi_{v,s}(\tilde{x}_v, x_s) \prod_{i \in N(v)\setminus s} M_{i,v}(\tilde{x}_v)$$



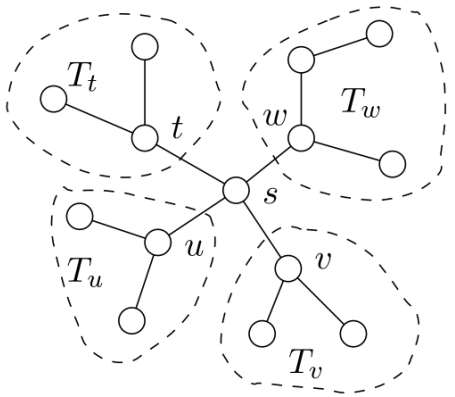
Marginal Inference on Trees

Switching order of sums and products works as long as G is a tree

- General form of calculating messages:

$$M_{v,s}(x_s) = \sum_{\tilde{x}_v} \Psi_v(\tilde{x}_v)\Psi_{v,s}(\tilde{x}_v, x_s) \prod_{i \in N(v)\setminus s} M_{i,v}(\tilde{x}_v)$$

- As with the chain, we aggregate messages from v's children, who are exactly $N(v) \setminus s$
- Eventually $P(X_s) \propto \Psi_s(x_s) \prod_{i \in N(s)} M_{i,s}(x_s)$



Belief Propagation on Trees

- Belief Propagation (BP) is an instance of *dynamic programming*: using solutions of subproblems to solve the entire task
  - Large task - Calculate messages $M_{ij}(x_i)$ for all $(i, j) \in E$, where $G = (V, E)$ is a tree
  - Subtasks - Let $r \in V$ be an arbitrary root, calculate messages $M_{ij}(x_i)$ in $G_c = (V_c, E_c)$ for subtree where $c \in N(r)$
- What happens if just update the messages iteratively with no particular order?

Sum-Product Belief Propagation

Sum-Product Belief Propagation

Input : Potentials $\{\Psi_i\}_{i \in V}$ and $\{\Psi_{ij}\}_{ij \in E}$

Output: Estimates $b_i(x_i)$ of $P(x_i)$ for all $X_i \in V$

while *not converged* **do**

Update for all $(i, j) \in E$,

$$M_{ji}^{(t)}(x_i) \leftarrow \kappa \sum_{\tilde{x}_j} \left\{ \Psi_{ij}(x_i, \tilde{x}_j) \psi_j(\tilde{x}_j) \prod_{k \in N(j) \setminus i} M_{k,j}(\tilde{x}_j) \right\},$$

and similarly for $M_{ij}^{(t)}(x_j)$.

end

return $b_i(x_i) \propto \Psi_i(x_i) \prod_{j \in N(i)} M_{j,i}(x_i)$ *for all* $X_i \in V$

Convergence of BP on Trees

Define: diameter of a graph is $\mathrm{diam}(G) = \max_{i,j \in V} \mathrm{dist}(i,j)$. Here, $\mathrm{dist}(i,j)$ is the length if the shortest path between the nodes

Theorem (BP is Exact on Trees)

Consider a Gibbs distribution P that factorizes over a tree G with $\mathrm{diam}(G) = t^*$, then

1. *The BP updates converge to a fixed point after at most t^* iterations, irrespective of initial messages. That is, for any $t > t^*$*

$$M_{ij}^t(x_j) = M_{ij}^{t^*}(x_j)$$

   for all $(i,j) \in E$ (and also for $M_{ji}(x_i)$)

2. *The beliefs $b_i(x_i)$ returned by BP are the marginals of P*

Proof Idea: induction on diameter of subtrees

Belief Propagation: Important Facts

- How about MAP inference? to calculate $\max_{\mathbf{x}} P(\mathbf{x})$ we can use a similar algorithm, where $\sum$ is replaced by $\max$
  - Note that $\max$ and $\prod$ commute, same as $\sum$ and $\prod$ do (i.e. $\max$ can be "pushed inside")
  - The resulting algorithm is called Max-Product Belief Propagation

Belief Propagation: Important Facts

- How about MAP inference? to calculate $\max_{\mathbf{x}} P(\mathbf{x})$ we can use a similar algorithm, where $\sum$ is replaced by $\max$
- What marginals can be calculated after messages converged? Notice that once we have $M_{ij}(x_i), M_{ji}(x_j)$ for all $(i,j) \in E$, we can calculate the marginals $P(x_k)$ for *all* $X_k \in V$.
- Straightforward generalization to factor graph handles non-pairwise factors (runtime exponential in size of largest factor)

Loopy Belief Propagation

What about non-tree graphs? Easy to observe that we *can* run this algorithm on non-tree graphs

Sum-Product Belief Propagation

Input : Potentials $\{\Psi_i\}_{i \in V}$ and $\{\Psi_{ij}\}_{ij \in E}$
Output: Estimates $b_i(x_i)$ of $P(x_i)$ for all $X_i \in V$
while *not converged* **do**
Update for all $(i, j) \in E$,

$$M_{ji}^{(t)}(x_i) \leftarrow \kappa \sum_{\tilde{x}_j} \left\{ \Psi_{ij}(x_i, \tilde{x}_j) \psi_j(\tilde{x}_j) \prod_{k \in N(j) \setminus i} M_{k,j}(\tilde{x}_j) \right\},$$

and similarly for $M_{ij}^{(t)}(x_j)$.
end
return $b_i(x_i) \propto \Psi_i(x_i) \prod_{j \in N(i)} M_{j,i}(x_i)$ *for all* $X_i \in V$

Loopy Belief Propagation

What about non-tree graphs? Easy to observe that we *can* run this algorithm on non-tree graphs

- Is it guaranteed to retrieve a correct solution? No
- Is it guaranteed to converge? No
- But, it works surprisingly well in practice and used in many applications in the past and also today
- Next: understanding BP from a different point-of-view

Inference as Optimization

Consider another approach to finding the marginals of P

Idea: if it difficult to do inference on P, use a surrogate distribution

- Define a family $\mathcal{P}$ of distributions, where marginals for each $Q \in \mathcal{P}$ can be calculated efficiently
- Given the distribution P for which we would like to do inference, approximate it with the marginals of the *"closest"* candidate $Q \in \mathcal{P}$
- This type of approximation scheme is called Variational Inference

Boltzmann Distributions

Our treatment of inference as optimization makes a mild assumption that $P(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x}))$ for some "energy function" $E : \mathcal{X} \to \mathbb{R}$

- Distributions of this form are called Boltzmann distributions
- When P is factorizes over an undirected graph G, we have $E(\mathbf{x}) = -\sum_{ij \in E} \log \Psi_{ij}(x_i, x_j) - \sum_{i \in V} \log \Psi_i(x_i)$
- Where have we seen this form of distribution before?

Inference as Optimization

- Idea: doing inference on P is difficult? Use a surrogate distribution!
  - Define a family $\mathcal{P}$ of distributions, where marginals for each $Q \in \mathcal{P}$ can be calculated efficiently
  - Given the distribution P for which we would like to do inference, approximate it with the marginals of the *"closest"* candidate $Q \in \mathcal{P}$

Naïve Mean Field Variational Inference

- Idea: doing inference on P is difficult? Use a surrogate distribution!
  - Define a family $\mathcal{P}$ of distributions, where marginals for each $Q \in \mathcal{P}$ can be calculated efficiently

    Simplest option: fully factorized distribution
  - Given the distribution P for which we would like to do inference, approximate it with the marginals of the *"closest"* candidate $Q \in \mathcal{P}$

    Measure of distance: KL-divergence $D_{KL}(Q\|P) = \mathbb{E}_Q\left[\log \frac{P}{Q}\right]$

Kullback-Leibler Divergence

Definition

For a Gibbs distribution $P(\mathbf{x})$ and a positive distribution Q the Kullback-Leibler divergence (also relative entropy) is

$$D_{KL}(Q\|P) = \sum_{\mathbf{x}} Q(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}$$

$\rightarrow \frac{1}{Z}\exp\{-E(x)\}$

Identities:

Jensen

- $D_{KL}(Q\|P) = -\mathbb{E}_Q \log \frac{Q}{P} \geq -\log \mathbb{E}_Q \frac{Q}{P} \geq 0$, equal iff $P = Q$
- The KL-divergence when P is a Boltzmann distribution is

$$D_{KL}(Q\|P) = \underbrace{\sum_{\mathbf{x}} Q(\mathbf{x})E(\mathbf{x})}_{\text{“Internal energy”: } U(Q;\Psi)} + \underbrace{\sum_{\mathbf{x}} Q(\mathbf{x}) \log Q(\mathbf{x})}_{\text{Neg. Entropy: } -H(Q)} + \log Z$$

Naïve Mean Field

Let us write down the optimization problem for Naïve Mean Field

$$\min_{Q \in \mathcal{P}} D_{KL}(Q \| P) = U(Q; \Psi) - H(Q) + \log Z$$

Note: $\log Z$ does not depend on Q, we can minimize $U(Q; \Psi) - H(Q)$

Naïve Mean Field

Let us write down the optimization problem for Naïve Mean Field

$$\min_{Q \in \mathcal{P}} U(Q; \Psi) - H(Q)$$

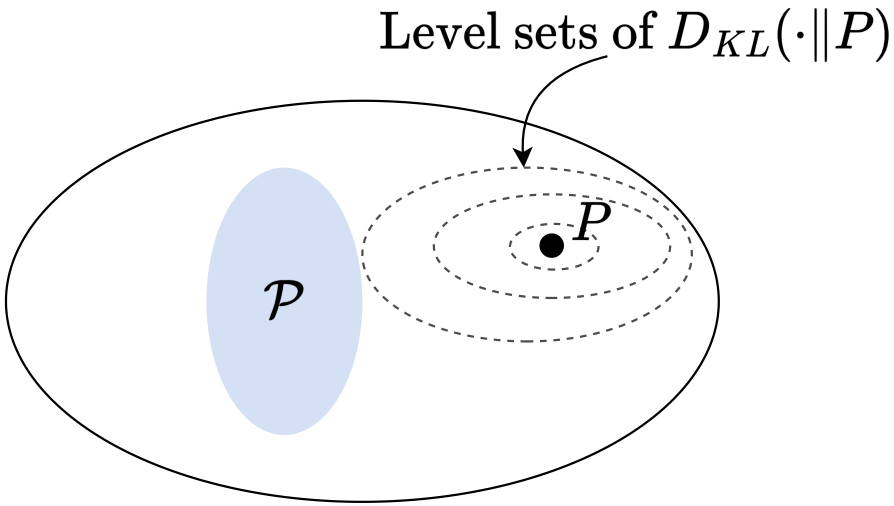
- With fully factorized distributions,
  $\mathcal{P} = \{\prod_i Q_i(x_i) : Q_i(x_i) > 0, \sum Q_i(x_i) = 1\}$
- Write down $U(Q; \Psi)$ and $H(Q)$

$$U(Q; \Psi) = -\sum_{ij} \sum_{x_i, x_j} Q_i(x_i) Q_j(x_j) \log \Psi_{ij}(x_i, x_j)$$
$$- \sum_i \sum_{x_i} Q_i(x_i) \log \Psi_i(x_i)$$
$$H(Q) = -\sum_i \sum_{x_i} Q_i(x_i) \log Q_i(x_i) = \sum_i H(Q_i)$$

Naïve Mean Field

To summarize so far:

- Our idea is to find the closest distribution to P out of a set $\mathcal{P}$



- For $\mathcal{P} = \{\prod_i Q_i(x_i)\}$, optimization is often tractable. Generally, solution is *not* $\prod_i P(X_i)$
- Can we go beyond factorized distributions? Tree structures? Connection to BP?

how to write well?? $I(x) = \sum_i$ 4

Factorization of Tree-Structures with Marginals

Proposition

For a Gibbs distribution $Q(\mathbf{x})$ that factorizes over a tree $G = (V, E)$ it holds that

$$Q(\mathbf{x}) = \prod_{ij \in E} Q(x_i, x_j) \prod_{i \in V} Q(x_i)^{1-d(i)},$$

where $d(i)$ is the degree of i in G.

Proof.

By induction on size of the tree. We choose some x_k for which $d(k) = 1$,[a] and denote $l = N(k)$. Writing $\mathbf{x}_{-k}$ as the random vector with all variables but x_k, we write

$$P(\mathbf{x}) = P(\mathbf{x}_{-k})P(x_k \mid \mathbf{x}_{-k}) = P(\mathbf{x}_{-k})P(x_k \mid x_l).$$

...

Factorization of Tree-Structures with Marginals

Proof.

...

$$P(\mathbf{x}) = P(\mathbf{x}_{-k})P(x_k \mid \mathbf{x}_{-k}) = P(\mathbf{x}_{-k})P(x_k \mid x_l).$$

The first equality is due the Bayes rule. The second holds because l separates k from other nodes in G, and as we learned last class, for Markov networks this means $x_k \perp\!\!\!\perp x_{-k} \setminus x_l \mid x_l$. Now rewrite $P(x_k \mid x_l) = P(x_k, x_l)/P(x_l)$, and also use the inductive hypothesis since $P(\mathbf{x}_{-k})$ factorizes on the tree $(V \setminus \{k\}, E \setminus (k,l))$. The degree of x_l in this subgraph is $d(l) - 1$, hence

$$P(\mathbf{x}) = \underbrace{\prod_{(i,j)\in E\setminus(k,l)} P(x_i, x_j) \prod_{i\neq k,l} P(x_i)^{1-d(i)} \cdot P(x_l)^{2-d(l)}}_{P(\mathbf{x}_{-k})}$$

$$\cdot P(x_k, x_l)P(x_l)^{-1} \underbrace{P(x_k)^{1-d(k)}}_{=1,\text{ as } d(k)=0} \ldots$$

Factorization of Tree-Structures with Marginals

Proof.

...

$$P(\mathbf{x}) = \underbrace{\prod_{(i,j)\in E\setminus(k,l)} P(x_i,x_j) \prod_{i\neq k,l} P(x_i)^{1-d(i)} \cdot P(x_l)^{2-d(l)}}_{P(\mathbf{x}_{-k})}$$
$$\cdot P(x_k,x_l)P(x_l)^{-1} \underbrace{P(x_k)^{1-d(k)}}_{=1,\text{ as } d(k)=0}$$

Putting this together we get the desired result

$$P(\mathbf{x}) = \prod_{(i,j)\in E} P(x_i,x_j) \prod_i P(x_i)^{1-d(i)}$$

□

Factorization of Tree-Structures with Marginals

Proposition

For a Gibbs distribution $Q(\mathbf{x})$ that factorizes over a tree $G = (V, E)$ it holds that

$$Q(\mathbf{x}) = \prod_{ij \in E} Q(x_i, x_j) \prod_{i \in V} Q(x_i)^{1-d(i)},$$

where $d(i)$ is the degree of i in G.

- This means that the entropy $H(Q)$ is

$$H(Q) = \sum_{ij \in E} H\left(Q(x_i, x_j)\right) + \sum_{i \in V} (1 - d_i) \cdot H\left(Q(x_i)\right)$$

- We will consider this type of decomposition for a *non-tree* graph

The Bethe Approximation

Consider a **cyclic** G. The variational principle we study for tree-approximations has the following components

- *pseudo*-marginals $\{b_i(x_i)\}_{i\in V}$ and $\{b_{ij}(x_i,x_j)\}_{ij\in E}$ where

$$\mathbb{L}_G = \left\{ b \geq 0 \ : \ \begin{array}{ll} \sum_{x_i} b_i(x_i) = 1 & \forall i \in V, \\ \sum_{x_i} b_{ij}(x_i,x_j) = b_j(x_j), & \forall ij \in E \\ \sum_{x_j} b_{ij}(x_i,x_j) = b_i(x_i) & \end{array} \right\}$$

- The "entropy" corresponding to these pseudo-marginals:

$$H_{\text{Bethe}}(b) = \sum_{ij\in E} H\left(b_{ij}(x_i,x_j)\right) + \sum_{i\in V}(1-d_i)\cdot H\left(b_i(x_i)\right)$$

The Bethe Approximation

Pseudo-marginals b and $H_{\text{Bethe}}(b)$ must be interpreted with care

- $H_{\text{Bethe}}(b)$ is an entropy-like expression of a function

$$\tilde{Q}(\mathbf{x}) = \prod_{ij \in E} b_{ij}(x_i, x_j) \prod_{i \in V} b_i(x_i)^{1-d(i)}$$

- But since G has cycles, $\tilde{Q}$ is *not a distribution* and H_{Bethe} is not an entropy. It's called the Bethe entropy approximation.
- For $b \in \mathbb{L}_G$, there might not even exist a distribution P whose marginals are $\{b_i\}_{i \in V}$ and $\{b_{i,j}\}_{ij \in E}$!

The Bethe Variational Principle

Finally, we consider the variational principle analogously to the mean-field case

$$\min_{Q \in \mathcal{P}} U(Q; \Psi) - H(Q) \leftrightarrow \min_{b \in \mathbb{L}_G} U(b; \Psi) - H_{\text{Bethe}}(b)$$

- The internal energy $U(b; \Psi)$ takes a similar form as before,

$$\begin{aligned} U(b; \Psi) = &- \sum_{ij} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \Psi_{ij}(x_i, x_j) \\ &- \sum_{i} \sum_{x_i} b_i(x_i) \log \Psi_i(x_i) \end{aligned}$$

- What can we say about this principle? It does not even directly correspond to $D_{KL}(\cdot \| P)$ minimization!

The Bethe Variational Principle

The Bethe variational principle is

$$\min_{b \in \mathbb{L}_G} U(b; \Psi) - H_{\text{Bethe}}(b).$$

We can write down the Lagrangian for the problem,

$$\begin{aligned}\mathcal{L}(b^*, \lambda^*; \Psi) =& U(b; \Psi) - H_{\text{Bethe}}(b) + \sum_{i \in V} \lambda_i C_i(b) \\ &+ \sum_{ij \in E} \left[\sum_{x_i} \lambda_{ji}(x_i) C_{ji}(x_i; b) + \sum_{x_j} \lambda_{ij}(x_j) C_{ij}(x_j; b) \right].\end{aligned}$$

Here we used a shortened notation for the constraints in $\mathcal{L}_G$,

$$C_i(b) := 1 - \sum_{x_i} b_i(x_i) = 0, \;\; C_{ij}(x_j; b) := \sum_{x_i} b_{ij}(x_i, x_j) - b_j(x_j) = 0$$

The Bethe Variational Principle and Belief Propagation

Theorem

For any graph G and $P(\mathbf{x})$ a Boltzmann distribution that factorizes over G, it holds that

1. *Any fixed point of Sum-Product BP specifies a pair (b^*, λ^*) such that*

$$\nabla_b \mathcal{L}(b^*, \lambda^*; \Psi) = 0, \text{ and } \nabla_\lambda \mathcal{L}(b^*, \lambda^*; \Psi) = 0 \qquad (1)$$

2. *When G is a tree there is only one pair (b^*, λ^*) that satisfies equation 1 and b^* corresponds to the marginals of P*

Conclusion

- Inference is (provably) hard!
- Belief Propagation is a practical and useful approach, works best on "tree-like" graphs
- The variational inference view of BP provides some insight and also motivated development of different algorithms
- Next week: more variational inference, this time with *learning* hidden variable models