CS 6347

Lectures 6 & 7

Approximate MAP Inference
Belief Propagation

• Efficient method for inference on a tree

• Represent the variable elimination process as a collection of messages passed between nodes in the tree

  • The messages keep track of the potential functions produced throughout the elimination process
Belief Propagation (for pairwise MRFs)

- \( p(x_1, ..., x_n) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \)

\[
m_{i \rightarrow j}(x_j) = \sum_{x_i} \phi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i)
\]

where \( N(i) \) is the set of neighbors of node \( i \) in the graph

- Messages are passed in two phases: from the leaves up to the root and then from the root down to the leaves
To construct the marginal distributions, we look at the beliefs produced by the algorithm

\[ b_i(x_i) = \frac{1}{Z} \phi_i(x_i) \prod_{k \in N(i)} m_{k \rightarrow i}(x_i) \]

\[ b_{ij}(x_i, x_j) = \frac{1}{Z} \phi_i(x_i) \phi_j(x_j) \psi_{ij}(x_i, x_j) \left( \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i) \right) \left( \prod_{k \in N(j) \setminus i} m_{k \rightarrow j}(x_j) \right) \]

Last time, we argued that, on a tree,

\[ b_i(x_i) = \sum_{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n} p(x_1, \ldots, x_n) \]
MAP Inference

- Compute the most likely assignment under the (conditional) joint distribution

\[ x^* = \arg\max_x p(x) \]

- Can encode 3-SAT, maximum independent set problem, etc. as a MAP inference problem
Max-Product (for pairwise MRFs)

- \( p(x_1, ..., x_n) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \)

\[
m_{i \to j}(x_j) = \max_{x_i} \left[ \phi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{k \to i}(x_i) \right]
\]

- Guaranteed to produced the correct answer on a tree
- Typical applications do not require computing \( Z \)
To construct the maximizing assignment, we look at the max-marginal produced by the algorithm

\[ \mu_i(x_i) = \frac{1}{Z} \phi_i(x_i) \prod_{k \in N(i)} m_{k \rightarrow i}(x_i) \]

\[ \mu_{ij}(x_i, x_j) = \frac{1}{Z} \phi_i(x_i) \phi_j(x_j) \psi_{ij}(x_i, x_j) \left( \prod_{k \in N(i) \setminus j} m_{k \rightarrow i}(x_i) \right) \left( \prod_{k \in N(j) \setminus i} m_{k \rightarrow j}(x_j) \right) \]

Again, on a tree,

\[ \mu_i(x_i) = \max_{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n} p(x_1, \ldots, x_n) \]
Reparameterization

- The messages passed in max-product and sum-product can be used to construct a reparameterization of the joint distribution

\[
p(x_1, ..., x_n) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j)
\]

and

\[
p(x_1, ..., x_n) = \frac{1}{Z} \prod_{i \in V} \left[ \phi_i(x_i) \prod_{k \in N(i)} m_{k \to i}(x_i) \right] \prod_{(i,j) \in E} \frac{\psi_{ij}(x_i, x_j)}{m_{i \to j}(x_j)m_{j \to i}(x_i)}
\]
Reparameterization

\[
p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{i \in V} \left[ \phi_i(x_i) \prod_{k \in N(i)} m_{k \rightarrow i}(x_i) \right] \prod_{(i,j) \in E} \frac{\psi_{ij}(x_i, x_j)}{m_{i \rightarrow j}(x_j)m_{j \rightarrow i}(x_i)}
\]

• Reparameterizations do not change the partition function, the MAP solution, or the factorization of the joint distribution

• They push "weight" around between the different factors

• Other reparameterizations are possible/useful
Max-Product Tree Reparameterization

• On a tree, the joint distribution has a special form

\[ p(x_1, \ldots, x_n) = \frac{1}{Z'} \prod_{i \in V} \mu_i(x_i) \prod_{(i,j) \in E} \frac{\mu_{ij}(x_i, x_j)}{\mu_i(x_i)\mu_j(x_j)} \]

• \( \mu_i \) is the max-marginal distribution of the \( i^{th} \) variable and \( \mu_{ij} \) is the max-marginal distribution for the edge \((i,j) \in E\)

• How to express \( \mu_{ij} \) as a function of the messages and the potential functions?
While max-product solves the MAP problem on trees, the MAP problem in MRFs is, in general, intractable (could use it to find a maximal independent set!)

- Don’t expect to be able to solve the problem exactly
- Will settle for “good” approximations
- Can use max-product messages as a starting point
- This is an active area of research
Upper Bounds

$$\max_{x_1, \ldots, x_n} p(x_1, \ldots, x_n) \leq \frac{1}{Z} \prod_{i \in V} \max_{x_i} \phi_i(x_i) \prod_{(i,j) \in E} \max_{x_i,x_j} \psi_{ij}(x_i, x_j)$$

• This provides an upper bound on the optimization problem

• Do other reparameterizations provide better bounds?
\[ L(m) = \frac{1}{Z} \prod_{i \in V} \max_{x_i} \left[ \phi_i(x_i) \prod_{k \in N(i)} m_{k \to i}(x_i) \right] \prod_{(i,j) \in E} \max_{x_i, x_j} \left[ \frac{\psi_{ij}(x_i, x_j)}{m_{i \to j}(x_j)m_{j \to i}(x_i)} \right] \]

- We construct a dual optimization problem

\[ \min_{m \geq 0} L(m) \geq \max_x p(x) \]

- Equivalently, we can minimize the convex function \( U \)

\[ U(\log m) = -\log Z + \sum_{i \in V} \max_{x_i} \left[ \log \phi_i(x_i) + \sum_{\{k \in N(i)\}} \log m_{k \to i}(x_i) \right] \]

\[ + \sum_{(i,j) \in E} \max_{x_i, x_j} \left[ \log \psi_{ij}(x_i, x_j) - \log m_{i \to j}(x_j) - \log m_{j \to i}(x_i) \right] \]
Convex and Concave Functions

Concave

Convex

Neither
Optimizing the Dual

• Minimizing $U(\log m)$

  • Block coordinate descent: improve the bound by changing only a small subset of the messages at a time (usually look like message-passing algorithms)

  • Subgradient descent: variant of gradient descent for non-differentiable functions

• Many more optimization methods...

• Note that $\min_{m \geq 0} L(m)$ is not necessarily equal to $\max_x p(x)$, so this procedure only yields an approximation to the maximal value
Gradient Descent

- Iterative method to minimize a differentiable convex function $f$ (for non-differentiable use subgradients)

- Intuition: step along a direction in which the function is decreasing

- Pick an initial point $x_0$

- Iterate until convergence

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t)$$

where $\gamma_t = \frac{2}{2+t}$ is the $t^{th}$ step size
Subgradients

For a convex function \( g(x) \), a subgradient at a point \( x^0 \) is any tangent line/plane through the point \( x^0 \) that underestimates the function everywhere.
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.

If $\vec{0}$ is a subgradient at $x^0$, then $x^0$ is a global minimum.
We can also express the MAP problem as a 0,1 integer programming problem

- Convert a maximum of a product into a maximum of a sum by taking logs
- Introduce indicator variables, \( \tau \), to represent the chosen assignment
• Introduce indicator variables for a specific assignment

  • $\tau_i(x_i) \in \{0,1\}$ for each $i \in V$ and $x_i$

  • $\tau_{ij}(x_i, x_j) \in \{0,1\}$ for each $(i, j) \in E$ and $x_i, x_j$

• The linear objective function is then

\[
\max_{\tau} \sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \phi_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \psi_{ij}(x_i, x_j)
\]

where the $\tau$'s are required to satisfy certain marginalization conditions
Integer Programming

\[
\max_{\tau} \sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \phi_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \psi_{ij}(x_i, x_j)
\]

such that

\[
\sum_{x_i} \tau_i(x_i) = 1 \quad \text{For all } i \in V
\]

\[
\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i) \quad \text{For all } (i,j) \in E, x_i
\]

\[
\tau_i(x_i) \in \{0,1\} \quad \text{For all } i \in V, x_i
\]

\[
\tau_{ij}(x_i, x_j) \in \{0,1\} \quad \text{For all } (i,j) \in E, x_i, x_j
\]
Integer Programming

\[
\max_{\tau} \sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \phi_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \psi_{ij}(x_i, x_j)
\]

such that

\[
\begin{align*}
\sum_{x_i} \tau_i(x_i) &= 1 & \text{For all } i \in V \\
\sum_{x_j} \tau_{ij}(x_i, x_j) &= \tau_i(x_i) & \text{For all } (i, j) \in E, x_i \\
\tau_i(x_i) &\in \{0,1\} & \text{For all } i \in V, x_i \\
\tau_{ij}(x_i, x_j) &\in \{0,1\} & \text{For all } (i, j) \in E, x_i, x_j
\end{align*}
\]

These constraints define the vertices of the **marginal polytope** (set of all valid marginal distributions).
Marginal Polytope

- Given an assignment to all of the random variables, $x^*$, can construct $\tau$ in the marginal polytope so that the value of the objective function is $\log p(x^*)$
  
  - Set $\tau_i(x_i^*) = 1$, and zero otherwise
  
  - Set $\tau_{ij}(x_i^*, x_j^*) = 1$, and zero otherwise

- Given a $\tau$ in the marginal polytope, can construct an $x^*$ such that the value of the objective function at $\tau$ is equal to $\log p(x^*)$
  
  - Set $x_i^* = \arg\max_{x_i} \tau_i(x_i)$
An Example: Independent Sets

• What is the integer programming problem corresponding to the uniform distribution over independent sets of a graph $G = (V, E)$?

$$p(x_V) = \frac{1}{Z} \prod_{(i,j) \in E} 1_{x_i + x_j \leq 1}$$

(worked out on the board)
Linear Relaxation

• The integer program can be relaxed into a linear program by replacing the 0,1 integrality constraints with linear constraints

  • This relaxed set of constraints forms the local marginal polytope

    • The $\tau$’s no longer correspond to an achievable marginal distribution, so we call them pseudo-marginals

  • We call it a relaxation because the constraints have been relaxed: all solutions to the IP are contained as solutions of the LP

• Linear programming problems can be solved in polynomial time!
Linear Relaxation

\[
\max_{\tau} \sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \phi_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i,x_j} \tau_{ij}(x_i,x_j) \log \psi_{ij}(x_i,x_j)
\]

such that

\[
\sum_{x_i} \tau_i(x_i) = 1 \quad \text{For all } i \in V
\]

\[
\sum_{x_j} \tau_{ij}(x_i,x_j) = \tau_i(x_i) \quad \text{For all } (i,j) \in E, x_i
\]

\[
\tau_i(x_i) \in [0,1] \quad \text{For all } i \in V, x_i
\]

\[
\tau_{ij}(x_i,x_j) \in [0,1] \quad \text{For all } (i,j) \in E, x_i, x_j
\]
An Example: Independent Sets

• What is the **linear** programming problem corresponding to the uniform distribution over independent sets of a graph $G = (V, E)$?

\[
p(x_V) = \frac{1}{Z} \prod_{(i, j) \in E} 1_{x_i + x_j \leq 1}
\]

• The MAP LP is a relaxation of the integer programming problem

• MAP LP could have a better solution... (example in class)
Tightness of the MAP LP

• When is it that solving the MAP LP (or equivalently, the dual optimization) is the same as solving the integer programming problem?

• We say that there is no gap when this is the case

• The answer can be expressed as a structural property of the graph (beyond the scope of this course)