Inference and Representation: Belief Propagation

Rahul G. Krishnan

New York University

Lab 4, September 28, 2016

Outline

Recap

- Consider a graph G where we run the variable elimination algorithm
- Eliminating a variable X from a graph creates a new factor connecting all neighbors of X (denoted $N(X)$) with each other
- This removes edges that were incident on X but creates new **fill edges** in the graph
- For every variable that is eliminated, these fill edges complete a clique on the set $N(X)$

Recap

- If we collect all these fill edges and super-impose them onto the original graph, we obtain $\mathcal{I}_{\Phi,\prec}$
- $\mathcal{I}_{\Phi,\prec}$ is the induced graph and is a function of a set of factors Φ and an elimination ordering $\prec$.

Recap

- Intuitively, when $\mathcal{I}_{\Phi,\prec}$ is a lot denser than G: bad news.
- Why?
- It means, somewhere in our variable elimination algorithm, we eliminated a variable with a large set $N(x)$
- This created a new clique comprised primarily of fill edges that are responsible for the increased density of the graph
- The larger the clique, the more storage we would need while running variable elimination

Treewidth

- Define N_{max} as the size of the largest clique in $\mathcal{I}_{\Phi,\prec}$
- Then the induced width: $w_{\mathcal{G},\prec} = N_{max} - 1$
- This is the width of the graph $\mathcal{I}_{\Phi,\prec}$ induced by applying VE to G with ordering $\prec$
- $w^*_{\mathcal{G}} = \min_{\prec} w_{\mathcal{G},\prec}$
- The treewidth is a *number* associated with every graph G
- It does *not* depend on the ordering we use in variable elimination
- The best running time achievable by VE is $O(mk^{w^*_{\mathcal{G}}+1})$ (m is the initial number of factors, $k = |Val(X)|$)

Example 1: A tree structured MRF

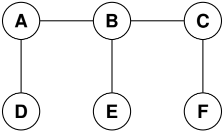


Figure: A tree structured MRF: HMM

Example 2: A grid structured MRF

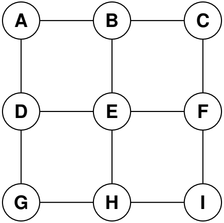


Figure: A grid structured MRF

Example 3: A fully connected MRF

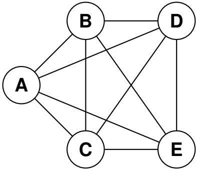


Figure: A fully connected graph on 5 variables

Example 4: A cycle

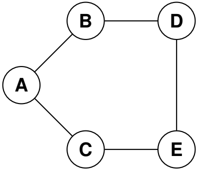


Figure: A cycle on 5 variables

Chordal Graphs

- Graphs of the form $\mathcal{I}_{\Phi,\prec}$ are called chordal or triangulated
- Defn: A chordal graph $\hat{G}$ is one in which every induced cycle has at most three vertices
- If you begin with a chordal graph, you can show that there exists an elimination ordering which yields no additional fill edges (See Koller & Friedman)

Example 5: A chordal cycle

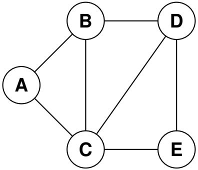
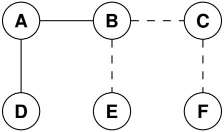


Figure: A chordal cycle $A - B - C - D - E$ on 5 variables

Belief Propagation

- What if we are interested in marginal inference over a larger subset of variables? Say all of them?
- Running VE repeatedly for each one is costly
- Can we do better?

Re-Use of Computation



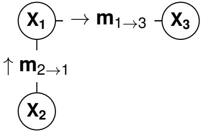
- Let the dashed arrows denote the factors whose variables we have marginalized out.
- What happens if we are interested in the marginal probability of A and D?
- With variable elimination, we would have to re-do computations corresponding to the dotted arrows.
- Could we cache the result we use to estimate the marginal probability of A and re-use it when estimating the marginal probability of D

Sum Product Belief Propagation

- We're going to talk about an algorithm that, on tree structured MRFs, computes all marginals with two passes through the graph
- The space requirement is linear in the number of variables
- Key Idea: Local computation for global inference
- Based on the idea of passing messages (partial summations) between neighboring vertices of the graph

Message

- $\mathbf{m}_{j\to i}(x_i) = \sum_{x_j}(\phi_j(x_j)\phi_{ij}(x_i, x_j)\prod_{k\in N(X_j)\setminus i}\mathbf{m}_{k\to j}(x_j))$
- Lets look at a specific example where all variables X_1, X_2, X_3 are binary:



- Lets consider the (vector) message from X_1 to X_3:

$$\mathbf{m}_{1\to 3}(X_3 = 0) = \sum_{v\in\{0,1\}}(\phi_1(X_1 = v)\phi_{1,3}(X_1 = v, X_3 = 0)\mathbf{m}_{2\to 1}(X_1 = v))$$

- Here we've assumed $\mathbf{m}_{2\to 1}$ exists. There must be an implicit order to how these messages are sent.

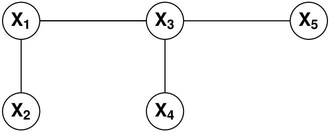
Algorithm

- Now that you know the semantics of a local message, lets look at the global algorithm
- First, pick a variable to act as the *root*
- Pass 1: Send messages from the *leaves* towards the root
- Pass 2: Send messages from the *root* back to the leaves
- For each node X_j, $P(X_j = x_j) \propto \phi_j(x_j) \prod_{i \in N(X_j)} \mathbf{m}_{i \to j}(x_j)$

Nomenclature

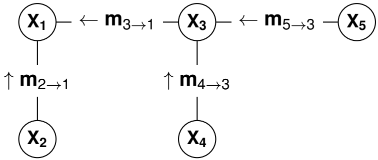
- The algorithm is named Sum-Product Belief Propagation
- $\mathbf{m}_{j\to i}(x_i) = \underbrace{\sum_{x_j}}_{\text{Sum}} (\phi_j(x_j)\phi_{ij}(x_i, x_j) \underbrace{\prod_{k\in N(X_j)\setminus i}}_{\text{Product}} \underbrace{\mathbf{m}_{k\to j}(x_j)}_{\text{Belief}})$
- The interpretation is that $\mathbf{m}_{i\to j}(k)$ corresponds to X_i's belief about $X_j = k$

Example : Belief Propagation



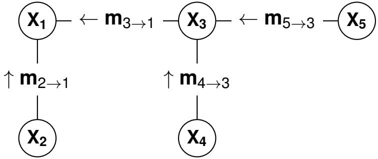
- Lets consider the above binary MRF and pick X_1 as the root
- The algorithm comprises the following messages from leaves to the root: $\mathbf{m}_{5\to3}(k), \mathbf{m}_{4\to3}(k), \mathbf{m}_{3\to1}(k), \mathbf{m}_{2\to1}(k)$ where $k \in \{0, 1\}$
- Then, from the root back to the leaves: $\mathbf{m}_{1\to2}(k), \mathbf{m}_{1\to3}(k), \mathbf{m}_{3\to4}(k), \mathbf{m}_{3\to5}(k)$ where $k \in \{0, 1\}$

Leaves to Root



- $\mathbf{m}_{4\to3}(x_3) = \sum_{x_4} \phi_4(x_4)\phi_{43}(x_4, x_3)$
- $\mathbf{m}_{5\to3}(x_3) = \sum_{x_5} \phi_5(x_5)\phi_{53}(x_5, x_3)$
- $\mathbf{m}_{2\to1}(x_1) = \sum_{x_2} \phi_2(x_2)\phi_{21}(x_2, x_1)$
- $\mathbf{m}_{3\to1}(x_1) = \sum_{x_3} \phi_3(x_3)\phi_{31}(x_3, x_1)\mathbf{m}_{4\to3}(x_3)\mathbf{m}_{5\to3}(x_3)$
- Notice anything? Substitute in $\mathbf{m}_{5\to3}(x_3)$ in the last formula and you'll see a pattern emerge.
- Belief Propagation is exactly implementing variable elimination for an ordering given to you by first eliminating leaves and moving towards the root

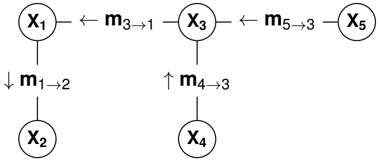
Leaves to Root



- $\mathbf{m}_{4\to3}(x_3) = \sum_{x_4} \phi_4(x_4)\phi_{43}(x_4, x_3)$
- $\mathbf{m}_{5\to3}(x_3) = \sum_{x_5} \phi_5(x_5)\phi_{53}(x_5, x_3)$
- $\mathbf{m}_{2\to1}(x_1) = \sum_{x_2} \phi_2(x_2)\phi_{21}(x_2, x_1)$
- $\mathbf{m}_{3\to1}(x_1) = \sum_{x_3} \phi_3(x_3)\phi_{31}(x_3, x_1)\mathbf{m}_{4\to3}(x_3)\mathbf{m}_{5\to3}(x_3)$
- $p(x_1) \propto \phi_1(x_1)\mathbf{m}_{2\to1}(x_1)\mathbf{m}_{3\to1}(x_1)$

Root to Leaves

- What if we're interested in $p(x_2)$?
- No problem, we just need $\mathbf{m}_{1\to 2}(x_2)$.
- Then: $p(x_2) \propto \phi(x_2)\mathbf{m}_{1\to 2}(x_2) = \phi(x_2)\sum_{x_1}\phi_1(x_1)\phi_{12}(x_1, x_2)\mathbf{m}_{3\to 1}(x_1)$
- Key Point: If we cache $\mathbf{m}_{3\to 1}(x_1)$, this message is trivial to compute. No need to go through the graph again!
- We can repeat this process propagating beliefs out from the root to all the other nodes to estimate their marginals too.



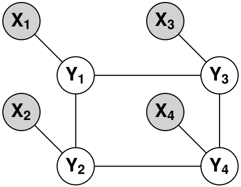
What if your graph is loopy?

- You no longer have a *root* or *leaves* in your graph
- You can still run the aforementioned algorithm anyways. This is called Loopy Belief Propagation.
- Algorithm:
  - Initialize all messages randomly
  - Repeat until messages have converged:
  - Send messages from a node to all its neighbors

Code Example

- We're going to run through an example where we use Loopy BP to perform segmentation
- Our image is $\mathbf{X} = M \times N$ dimensional. Each random variable $X_{ij} \in \{1, \ldots, 32\}$ corresponds to a pixel and represents greyscale intensity.
- We're interested in segmentation so we'll use $\mathbf{L} = M \times N$ to denote the label for every pixel.
- Random variable $L_{ij} \in \{0, 1\}$ denotes whether the pixel belongs to the foreground or background

Problem Setup



- In presence of evidence, the message from X_1 to Y_1 will not involve a summation over states of X_1 but rather a fixed choice of X_1 clamped to the evidence
- The structure captures our prior assumptions of how the labelling of foreground/background should behave