Markov Random Field (MRF) and Inference Methods: Belief Propagation (BP), Loopy BP

Kashob Roy, Amit Roy and Abdullah Al Thaki

September 11, 2021

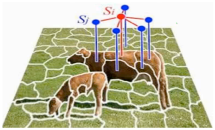
AGenCy Lab
Independent University Bangladesh

Outline

- Real-life Applications
- Markov Random Field
  - A toy example to explain MRF
  - Conditional Independence
  - Factorization: Pairwise
  - Clique Factorization
- Inference in MRF
  - Belief Propagation : Sum-Product Algorithm
  - Loopy BP
  - Linearized BP

Image Segmentation Formulation

- Partition the image pixels into regions corresponding to distinct parts of scene
- binary class segmentation: foreground or background
  - pixel treated as a random variable X_i that has a domain {0,1}
  - 0 = foreground and 1 = background
- A node for each pixel (we can versegment image into superpixels amd classify each superpixels.)
- Edge between neighboring pixels

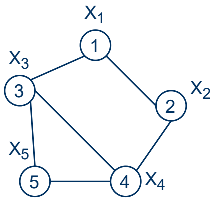


- Two facts:
  1. label of pixel depends on the statistics over color, texture,etc.
  2. two neighboring pixel are likely to have same label.

The join probability distribution, $P(X_1, .., X_n)$ can be formulated through a undirected graph where node and edge constraints take care of two facts respectively.

Node Classification

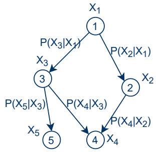
- Graph of nodes $X_1, ..X_n$ related by set of edges E
- Assign each node X_1 a label from $V = v_1, .., v_k$
- Each node, taken in isolation, has its preference among possible labels which be called unary potential or node potentials,
- Also need to impose a soft **smoothness** constraint (aka edge potentials) that neighboring nodes should take similar values



Similar to image segmentation problem, we can formulate the join probability distribution, $P(X_1, .., X_n)$ through a undirected graph

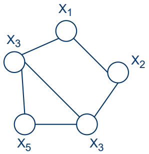
PGM: Bayesian Network & Markov Random Field

- In general, this model is called a Probabilistic Graphical Model (PGM) that defines a **parameterized probability distribution** and holds a set of **conditional independence assumptions**
- Directed Graphical Model aka **BN**



  – $P(X_1, X_2, X_3, X_4, X_5) = P(X_1) \cdot P(X_2|X_1) \cdot P(X_3|X_1) \cdot P(X_4|X_2) \cdot P(X_4|X_3) \cdot P(X_5|X_3)$

- Undirected Graphical Model aka **MRF**



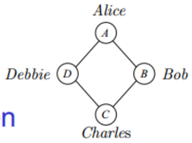
  – A graph over random variables $X_1, .., X_n$ and cycles are permitted
  – **Potential Function** Ψ, also called "factors", are used to define the joint probability $P(X_1, .., X_n)$

A Toy Example of Undirected Graphical Model

1. Four students study in pairs to work on homework

$Alice$ and Bob are friends
Bob and $Charles$ study together
$Charles$ and $Debbie$ argue with each other
$Debbie$ and $Alice$ study together
$Alice$ and $Charles$ can't stand each other
Bob and $Debbie$ had relationship ended badly

A Social Network with
$(A \perp C | \{B, D\})$ and $(B \perp D | A, C)$



2. Professor may have mis-spoken
   e.g., on a machine learning topic

3. Students may have figured out the problem
   e.g., by thinking about issue
   or by studying textbook

4. Students transmit this understanding to his/her study partner

Modeling the Example: Misconception Problem

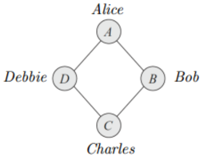
Probability of misconception of one person depends on whether their study partner has a misconception

1. *Conditional Independence I*: **Alice and Charles can't stand each other**
   It implies that $(A \perp C | B, D)$
2. *Conditional Independence II*: **Bob and Debbie had relationship ended badly**
   Also, means that $(B \perp D | A, C)$

We need to model

$$(A \perp C | B, D), (B \perp D | A, C)$$

Note that: There does not exist a **BN** that can capture these two assumptions



Modeling the Misconception Problem

Four binary random variables representing whether or not student has misconception

$X \in \{A, B, C, D\}$

x^1 : student has the misconception

x^0 : Student does not have a misconcpetion

Probabilities assuming four variables are independent

a^0	a^1
0.3	0.7

b^0	b^1
0.2	0.8

etc

a^0 = has misconception
a^1 = no misconception

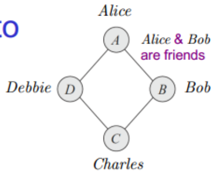
To get contextual interactions between variables we need a MN

Capturing Affinities between Variables

- Let $\boldsymbol{D}$ be a set of random variables
- A factor ϕ is a function from $Val(\boldsymbol{D})$ to R
  - where Val is the set of values that $\boldsymbol{D}$ can take
- A factor is non-negative if all its entries are non-negative
- The set of variables $\boldsymbol{D}$ is called the scope of the factor, denoted $Scope[\Phi]$
- In our example, $\phi_1(A,B): Val(A,B) \rightarrow R^+$
  - Higher the value, more compatible they are

Example of a Factor

- Factor is not normalized to sum to one
- Values need not be in $[0,1]$
- $\phi_1(A,B)$ asserts that A and B
  - are more likely to agree than disagree
  - More weight to when they agree than when they disagree
    - Translates to higher probability for disagreeing values



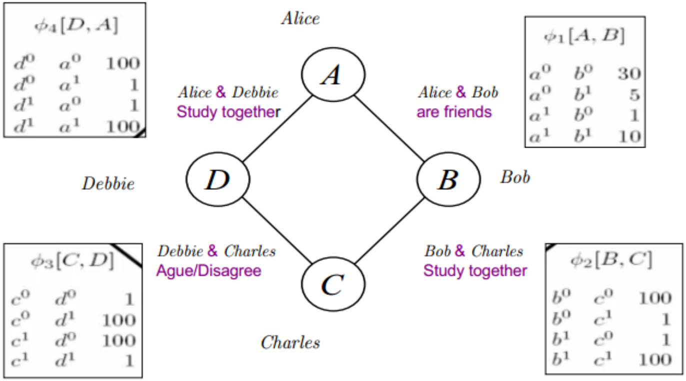
$\phi_1[A,B]$		
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

a^0 = has misconception
a^1 = no misconception

b^0 = has misconception
b^1 = no misconception

12

Factors of the Misconception Example



$\phi_4[D, A]$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1[A, B]$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

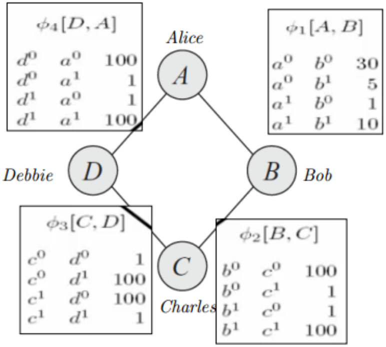
$\phi_3[C, D]$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2[B, C]$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Pairwise Compatibility Factors



$\phi_4[D, A]$		
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1[A, B]$		
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

$\phi_3[C, D]$		
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2[B, C]$		
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

a^0 = has misconception
a^1 = no misconception

b^0 = has misconception
b^1 = no misconception

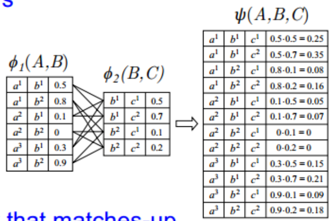
Alice and Debbie
are compitable

Charles and *Debbie*
are incompatible

Most likely instantiations are
when they disagree

Definition of the Factor Product

Let X, Y and Z be three disjoint sets of variables
$\phi_1(X,Y)$ and $\phi_2(Y,Z)$ are two factors
Factor product $\phi_1 \mathbf{X}\, \phi_2$ is a factor
ψ: $Val(X,Y,Z) \rightarrow R$
as follows:
$\psi\ (X,Y,Z) = \phi_1(X,Y).\phi_2(Y,Z)$



$\phi_1(A,B)$

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

$\phi_2(B,C)$

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2

$\psi(A,B,C)$

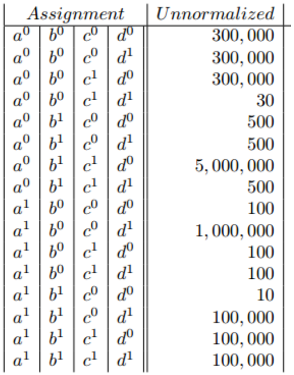
a^1	b^1	c^1	$0.5 \cdot 0.5 = 0.25$
a^1	b^1	c^2	$0.5 \cdot 0.7 = 0.35$
a^1	b^2	c^1	$0.8 \cdot 0.1 = 0.08$
a^1	b^2	c^2	$0.8 \cdot 0.2 = 0.16$
a^2	b^1	c^1	$0.1 \cdot 0.5 = 0.05$
a^2	b^1	c^2	$0.1 \cdot 0.7 = 0.07$
a^2	b^2	c^1	$0 \cdot 0.1 = 0$
a^2	b^2	c^2	$0 \cdot 0.2 = 0$
a^3	b^1	c^1	$0.3 \cdot 0.5 = 0.15$
a^3	b^1	c^2	$0.3 \cdot 0.7 = 0.21$
a^3	b^2	c^1	$0.9 \cdot 0.1 = 0.09$
a^3	b^2	c^2	$0.9 \cdot 0.2 = 0.18$

Two factors are multiplied in a way that matches-up the common part Y

In the example, factors do not correspond to either probabilities or conditional probabilities

Combining Local Models into Global

- We combine local interactions (models) by multiplying them $\phi_1(A, B) \cdot \phi_2(B, C) \cdot \phi_3(C, D) \cdot \phi_4(D, A)$
  Alice, *Bob* and *Debbie* have **misconception** but *Charles* **does not**
  $\phi_1(a^1, b^1) \cdot \phi_2(b^1, c^0) \cdot \phi_3(c^0, d^1) \cdot \phi_4(d^1, a^1) = 10 \cdot 1 \cdot 100 \cdot 100 = 100000$
- Covert to a legal distribution by performing a normalization.



Assignment				*Unnormalized*
a^0	b^0	c^0	d^0	$300,000$
a^0	b^0	c^0	d^1	$300,000$
a^0	b^0	c^1	d^0	$300,000$
a^0	b^0	c^1	d^1	30
a^0	b^1	c^0	d^0	500
a^0	b^1	c^0	d^1	500
a^0	b^1	c^1	d^0	$5,000,000$
a^0	b^1	c^1	d^1	500
a^1	b^0	c^0	d^0	100
a^1	b^0	c^0	d^1	$1,000,000$
a^1	b^0	c^1	d^0	100
a^1	b^0	c^1	d^1	100
a^1	b^1	c^0	d^0	10
a^1	b^1	c^0	d^1	$100,000$
a^1	b^1	c^1	d^0	$100,000$
a^1	b^1	c^1	d^1	$100,000$

Normalized Joint Distribution

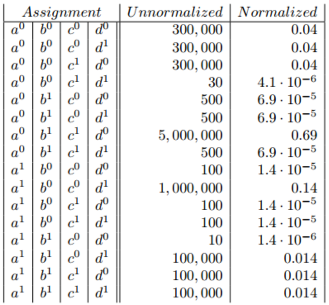
$$P(a,b,c,d) = \frac{1}{Z}\phi_1(a,b)\cdot\phi_2(b,c)\cdot\phi_3(c,d)\cdot\phi_4(d,$$

where,

$$Z = \sum_{a,b,c,d} \phi_1(a,b)\cdot\phi_2(b,c)\cdot\phi_3(c,d)\cdot\phi_4(d,a)$$

Z is a normalizing constant called the **Partition Function**

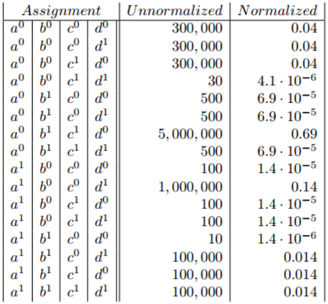
The value of Z here is 7201840



Assignment				*Unnormalized*	*Normalized*
a^0	b^0	c^0	d^0	$300,000$	0.04
a^0	b^0	c^0	d^1	$300,000$	0.04
a^0	b^0	c^1	d^0	$300,000$	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	$5,000,000$	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	$1,000,000$	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	$100,000$	0.014
a^1	b^1	c^1	d^0	$100,000$	0.014
a^1	b^1	c^1	d^1	$100,000$	0.014

Answering Queries

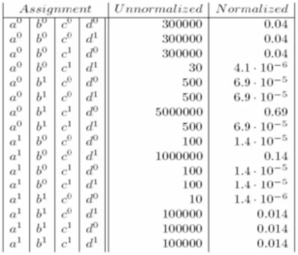
- We can obtain any desired probability from the joint distribution as usual, $P(b^0) = 0.268$: *Bob* is 26% likely to have a misconception
  $P(b^1|c^0) = 0.06$: if *Charles* does not have the misconception, *Bob* is only 6% likely to have misconception.
- Most probable joint probability: $P(a^0, b^1, c^1, d^0) = 0.69$: *Alice*, *Debby* **have no misconception**, *Bob*, *Charles* **have misconception**



Assignment				*Unnormalized*	*Normalized*
a^0	b^0	c^0	d^0	$300,000$	0.04
a^0	b^0	c^0	d^1	$300,000$	0.04
a^0	b^0	c^1	d^0	$300,000$	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	$5,000,000$	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	$1,000,000$	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	$100,000$	0.014
a^1	b^1	c^1	d^0	$100,000$	0.014
a^1	b^1	c^1	d^1	$100,000$	0.014

Factors $\neq$ Marginal Probabilities

Joint Distribution

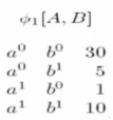


Assignment				Unnormalized	Normalized
a^0	b^0	c^0	d^0	300000	0.04
a^0	b^0	c^0	d^1	300000	0.04
a^0	b^0	c^1	d^0	300000	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	5000000	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	1000000	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	100000	0.014
a^1	b^1	c^1	d^0	100000	0.014
a^1	b^1	c^1	d^1	100000	0.014

Marginal distribution over Alice,Bob

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

Factor over Alice,Bob

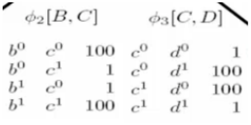


$\phi_1[A, B]$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

Most likely: $a^0, b^1 (disagree)$ $a^0, b^0 (agree)$

Because probability takes into account influence of other factors

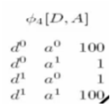


$\phi_2[B, C]$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

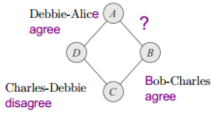
$\phi_3[C, D]$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1



$\phi_4[D, A]$

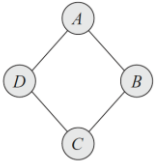
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100



Preserving Independence in Misconception Problem

- Tight connection between factorization and independencies
  - P supports $(X \perp Y | Z)$ iff we can write distribution as $P(X) = \phi_1(X, Z)\phi_2(Y, Z)$
    - In our example, we can write

$$P(A,B,C,D) = \underbrace{\left[\frac{1}{Z}\phi_1(A,B)\cdot\phi_2(B,C)\cdot\right]}_{\text{Factor with } \{B,\{A,C\}\}} \underbrace{\phi_3(C,D)\cdot\phi_4(D,A)}_{\text{Factor with } \{D,\{A,C\}\}}$$



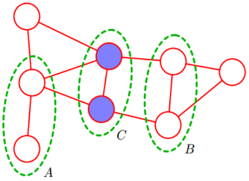
- Therefore $(B \perp D \mid A, C)$
- By grouping factors with $\{A,\{B,D\}\}$ and $\{C,\{B,D\}\}$
- We get $(A \perp C \mid B, D)$

Are Pairwise Potentials are sufficient?

- **One idea: associate a factor with each edge**
  but it is insufficient to specify arbitrary joint distribution.
  - Consider a fully connected graph
    - There are no independence assumptions
  - If all variables are binary
    - Each factor over an edge has 4 parameters
    - Total number of parameters is $4 \times {}^nC_2$
  - An arbitrary distribution needs $2^n - 1$ parameters
- Pairwise factors have insufficient parameters
- For more general representation, factors should be defined over arbitrary subsets of variables.

Conditional Independence in MRF

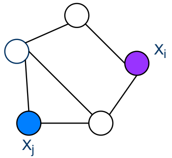
- Conditional Independence is determined by simple graph separation
  - Consider three sets of nodes A, B, and C
  - Cond. Independence property: $A \perp B | C$
  - This propery holds only if all possible paths that connects nodes in A to nodes in B are "**blocked**":
    - all such paths pass through one or more nodes in C.
- If there at least one unblocked path between A and B:
  - there exists at least some distributions corresponding to the graph that do not satisfy $A \perp B | C$ property.



Factorization Properties

- Factorization rule corresponds to the conditional independence test
- The joint distribution p(x) expressed as a product of functions defined over sets of variables that are local to the graph

- Consider two nodes x_i and x_j not connect by a link
  - They are conditionally independent given all other nodes in graph
    - Because there is no direct path between them
    - all other paths pass through nodes that are observed and hence those paths are "**blocked**"
  - Expressed as: $p(x_i, x_j | X_{\setminus\{i,j\}} = p(x_i | X_{\setminus\{i,j\}} \cdot p(x_j | X_{\setminus\{i,j\}}$
    - Where $X_{\setminus\{i,j\}}$: set of X of all variables with x_i and x_j removed
- For the conditional independence to hold
  - factorization is such that x_i and x_j do not appear in the same factor.
    - No path between them other than going through others
  - leads to **graph concept of clique**



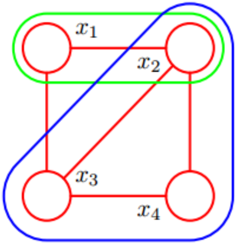
Factors as Cliques

- Set of variables in clique $\mathcal{C}$ is denoted $X_\mathcal{C}$
- Joint distribution as a product of clique potential functions,

$$p(X) = \frac{1}{Z}\prod_{\mathcal{C}} \Psi_\mathcal{C}(X_\mathcal{C})$$

- Where Z , the partition function, is a normalization constant

$$Z = \sum_{X}\prod_{\mathcal{C}} \Psi_\mathcal{C}(X_\mathcal{C})$$



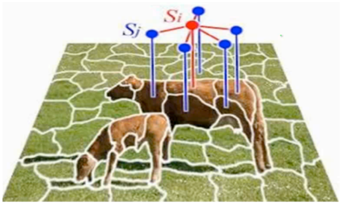
- Clique potential can be derived by multiplying all factors assigned to each clique
  - we can write $\Psi_i(x_2, x_3, x_4) = \psi_j(x_2, x_3) \cdot \psi_k(x_2, x_4) \cdot \psi_k(x_3, x_4)$
  - Limits the expressibility of model to some extent.
    - because we can not define the clique potential function by an arbitrary function that is limited by pairwise potentials.
    - In practice, pairwise potentials are more popular.

Relationship btw Factorization & Con. Independence

- ***UI***: set of such distributions that are consistent with the set of conditional independence corresponding to the graph
- ***UF***: set of such distributions that are expressed as factorization w.r.t the maximal cliques.
- *Hammersley-Clifford* theorem states that UI and UF are identical:
  - if potential functions are restricted to be strictly positive
- In practice, $\Psi_C(x_C) = exp\left\{ -E(x_C) \right\}$ is commonly used
  - $E(x_C)$ is an energy functiom
  - the joint distribution is called **Gibbs Distribution**

MRF into Image Segmentation

- A node for each pixel or superpixel, X_i
- Edge between nodes if regions are adjacent
- Joint probability distribution over an image defined using Log-linear model as,



$$P(X_1, .., X_b; \theta) = \frac{1}{Z(\theta)} exp\left\{ -\sum_{i=1}^{k} E_i(D_i, \theta_i) \right\}$$

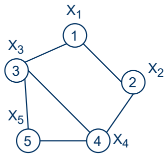
where, $\Psi(D_i) = exp(-E_i(D_i, \theta_i))$

$$Z(\theta) = \sum_{\xi} exp\left\{ -\sum_{i=1}^{k} E_i(\xi, \theta_i) \right\}$$

- E_i is the energy function
  - D_i is the domain of X_i
- k is the number of factors

MRF into Node classification

- For each node, node potentials defined as $\mathcal{E}_i(X_i)$
- To impose a soft **smoothness** constraint, edge potentials defined as $\mathcal{E}_{ij}(X_i, X_j)$
- From the perspective of physics, **Higher energy states have lower probability**



- The Goal of node classification is to minize the energy function:

$$arg \min_{x_1,..x_n} E(x_1, ..x_n) = \sum_i \mathcal{E}_i(x_i) + \sum_{\{i,j\}} \mathcal{E}_{ij}(x_i, x_j)$$

- Returns the setting of random variables/nodes that have the highest joint probability

Inference

Inference tasks of PGM is to :

1. Find marginals of random variables [**Sum-Product Algorithm**]
2. Find the setting of the variables that has the highest joint probability and the value of that probability [**Max-Sum Algorithm**]

Sum-Product Algorithm: Exact Inference

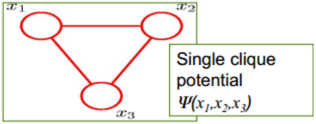
- Also known as **Belief Propagation**
- Basically, a message passing algorithm on a factor graph
  - A efficient, exact inference algorithm for finding marginals

$$p(x) = \sum_{\mathbf{X} \setminus x} p(\mathbf{X})$$

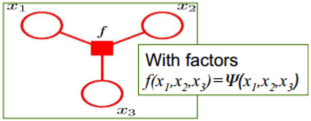
    where $\mathbf{X} \setminus x$ denotes the set of variables in $\mathbf{X}$ with variable x omitted
  - Several marginal computation can be shared efficiently
- Factor graph must have a tree structure.

Conversion of Undirected Graph to Factor Graph

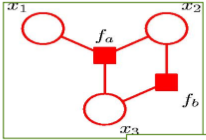
- Steps in converting distribution expressed as undirected graph:
  1. Create variable nodes corresponding to nodes in original
  2. Create factor nodes for maximal cliques x_s
  3. Factors $f_s(x_s)$ set equal to clique potentials
- Several different factor graphs possible from same distribution



Single clique potential $\Psi(x_1,x_2,x_3)$



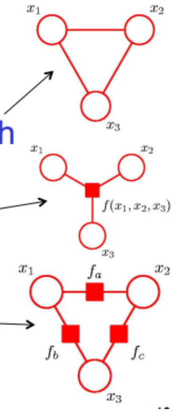
With factors $f(x_1,x_2,x_3)=\Psi(x_1,x_2,x_3)$



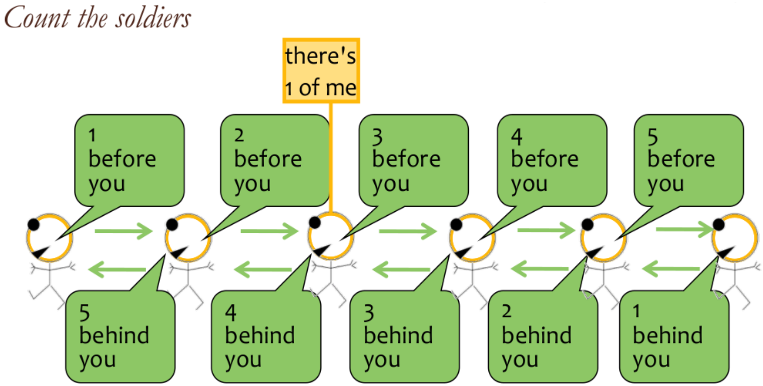
With factors $f_a(x_1,x_2,x_3)f_b(x_1,x_2)=\Psi(x_1,x_2,x_3)$

Multiple Factor Graphs for same Graph

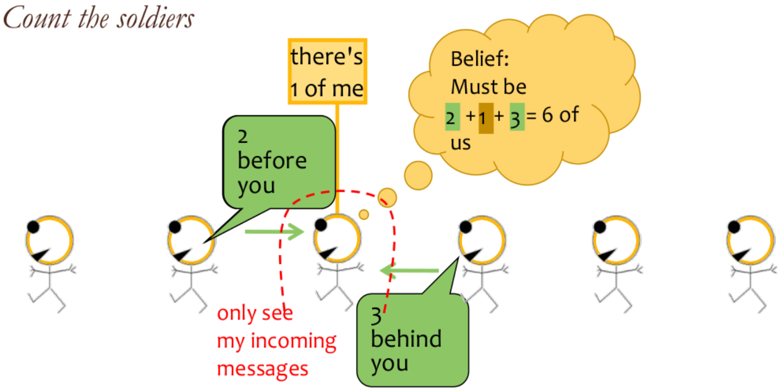
- Factor graphs are specific about factorization
- A fully connected undirected graph
- Joint distribution in two forms
  - In general form
    $p(x)=f(x_1, x_2, x_3)$
  - As a specific factorization
    $p(x)=f_a(x_1, x_2) f_b(x_1, x_3) f_c(x_2, x_3)$



Message Passing Intution

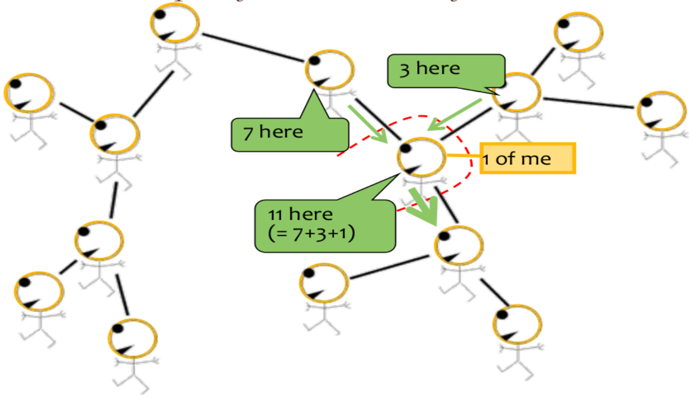


Message Passing Intution


Count the soldiers
there's 1 of me
2 before you
Belief: Must be 2 + 1 + 3 = 6 of us
only see my incoming messages
3 behind you

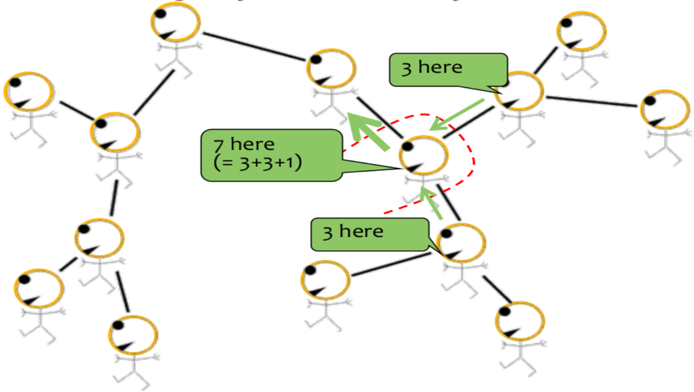
Message Passing Intution

Each soldier receives reports from all branches of tree

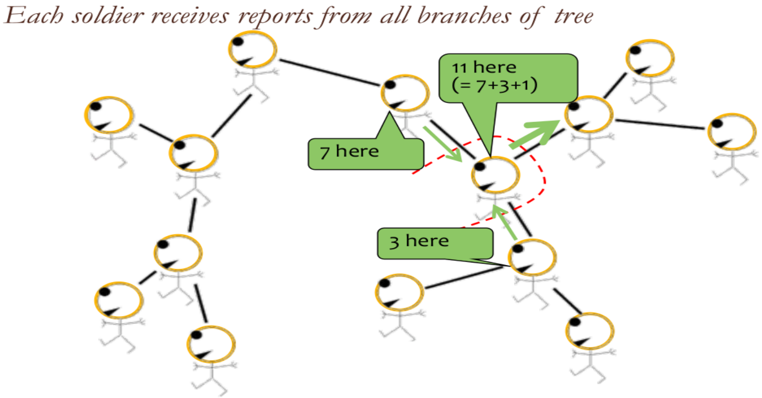


Message Passing Intution

Each soldier receives reports from all branches of tree

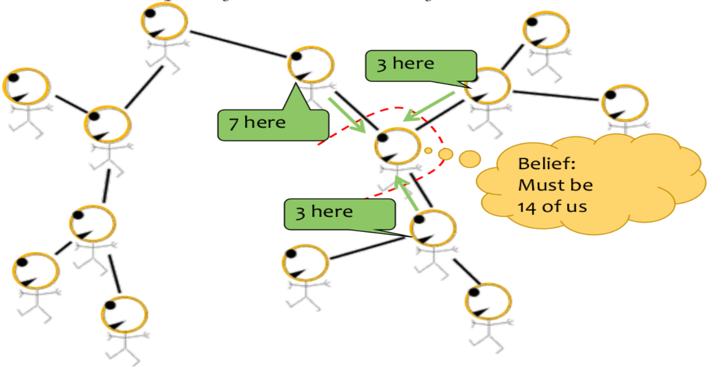


Message Passing Intution

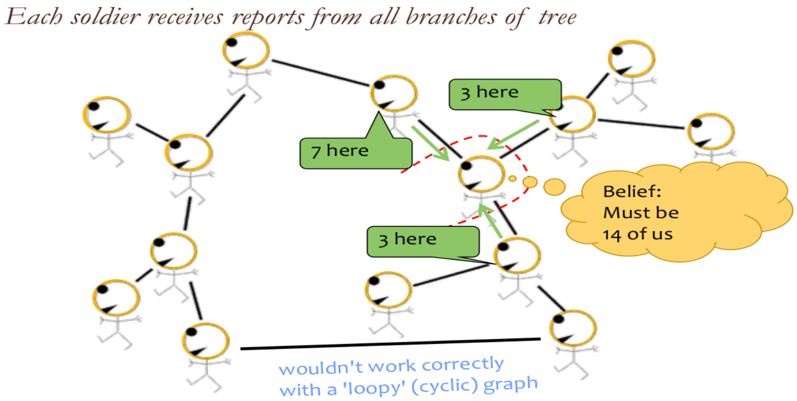


Message Passing Intution

Each soldier receives reports from all branches of tree



Message Passing Intution

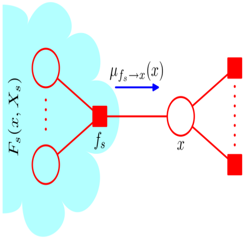

Each soldier receives reports from all branches of tree
7 here
3 here
3 here
Belief: Must be 14 of us
wouldn't work correctly with a 'loopy' (cyclic) graph

Belief Propagation

- The joint distribution can be written as a product of the form

$$p(\mathbf{X}) = \prod_{s \in ne(x)} F_s(x, X_s) \tag{1}$$

- $ne(x)$ - set of neighbor factor nodes of x
- X_s - set of all variables in the subtree connected to the variable node x via the factor node f_s
- $F_s(x, X_s)$ - product of all the factors in the group associated with factor f_s



A fragment of a factor graph illustrating the evaluation of the marginal p(x)

Belief Propagation

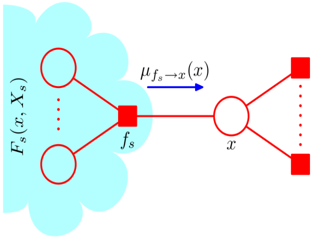
- we get the marginal $p(x)$ -

$$p(x) = \sum_{\mathbf{X}\backslash x}\Big[\prod_{s\in ne(x)} F_s(x, X_s)\Big]$$
$$p(x) = \prod_{s\in ne(x)}\Big[\sum_{X_s} F_s(x, X_s)\Big] \quad (2)$$

- We introduce a set of functions denoting the messages from factor node f_s to variable node x denoted by,

$$\mu_{f_s\to x}(x) \equiv \sum_{X_s} F_s(x, X_s) \quad (3)$$

- Required marginal $p(x)$ can be written as the product of all the incoming messages arriving at node x



A fragment of a factor graph illustrating the evaluation of the marginal p(x)

Belief Propagation

- To evaluate the messages $\mu_{f_s \to x}(x)$, each factor $F_s(x, X_s)$ can be factorized using the factor graph of Figure 1.

$$F_s(x, X_s) = f_s(x, x_1, \dots, x_M) G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM}) \tag{4}$$

- Substituting $F_s(x, X_s)$ in Equation 3 with Equation 4 we obtain

$$\mu_{f_s \to x} = \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in ne(f_s) \setminus x} [\sum_{X_{sm}} G_m(x_m, X_{sm})] \tag{5}$$

$$= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \tag{6}$$

messages from variable nodes x_m to factor node f_s, $\mu_{x_m \to f_s}$ are defined as

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{xm}} G_m(x_m, X_{sm}) \tag{7}$$

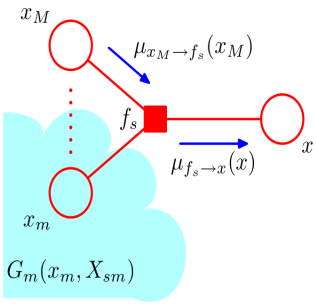


Figure 1: Illustration of the factorization of the subgraph associated with factor node f_s

Belief Propagation

- $G_m(x_m, X_{sm})$ associated with the node x_m defined as the product of terms $F_l(x_m, X_{ml})$

$$G_m(x_m, X_{sm}) = \prod_{l \in ne(x_m) \setminus f_s} F_l(x_m, X_{ml}) \qquad (8)$$

- Hence, Messages from variable node x to factor node f_s,

$$\begin{aligned}\mu_{x_m \to f_s}(x) &\equiv \sum_{X_{xm}} G_m(x_m, X_{sm}) \\ &= \prod_{l \in ne(x_m) \setminus f_s} [\sum_{X_{xm}} F_l(x_m, X_{ml})] \\ &= \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)\end{aligned}$$

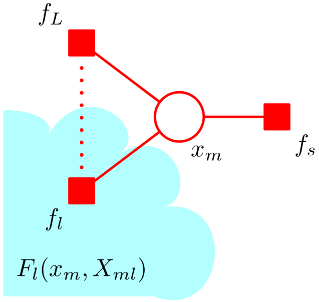


Figure 2: Illustration of the evaluation of the message sent by a variable node to an adjacent factor node

Belief Propagation

- Two kinds of messages are defined over the discussion
  - Messages from factor node f_s to variable node x, $\mu_{f_s \to x}(x)$

$$\mu_{f_s \to x}(x) \equiv \sum_{x_1} \ldots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \tag{9}$$

    To evaluate the message sent by a factor node to a variable node along the link connecting them, take the product of the incoming messages along all other links coming into the factor node, multiply by the factor associated with that node, and then marginalize over all of the variables associated with the incoming messages

  - Messages from variable node x to variable node f_s, $\mu_{x \to f_s}(x)$

$$\mu_{x \to f_s}(x) = \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x}(x) \tag{10}$$

    To evaluate the message sent by a variable node to an adjacent factor node along the link connecting them, take the product of the incoming messages along all of the other links coming into the variable node.

Message Passed from Leaf Nodes

The sum-product algorithm begins with messages sent by the leaf nodes, which depend on whether the leaf node is (a) a variable node, or (b) a factor node.

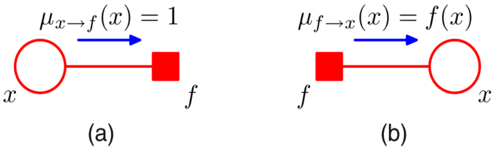


Figure 3: Messaging Passing From Leaf Nodes in Sum-Product Algorithm

Belief Propagation

Input: a factor graph with no cycles
Output: exact marginals for each variable and factor

Algorithm:

1. Initialize the messages to the uniform distribution.

$$\boxed{\mu_{i\to\alpha}(x_i) = 1} \quad \boxed{\mu_{\alpha\to i}(x_i) = 1}$$

1. Choose a root node.
2. Send messages from the **leaves** to the **root**.
   Send messages from the **root** to the **leaves**.

$$\boxed{\mu_{i\to\alpha}(x_i) = \prod_{\alpha\in\mathcal{N}(i)\setminus\alpha} \mu_{\alpha\to i}(x_i)} \quad \boxed{\mu_{\alpha\to i}(x_i) = \sum_{\boldsymbol{x}_{\boldsymbol{\alpha}}:\boldsymbol{x}_{\boldsymbol{\alpha}}[i]=x_i} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{j\in\mathcal{N}(\alpha)\setminus i} \mu_{j\to\alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

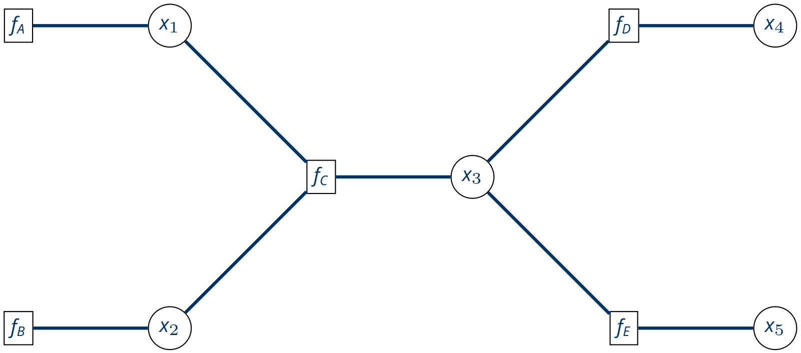
1. Compute the beliefs (unnormalized marginals).

$$\boxed{b_i(x_i) = \prod_{\alpha\in\mathcal{N}(i)} \mu_{\alpha\to i}(x_i)} \quad \boxed{b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) = \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{i\in\mathcal{N}(\alpha)} \mu_{i\to\alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

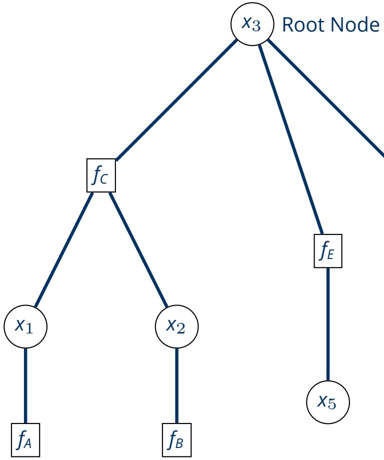
2. Normalize beliefs and return the **exact** marginals.

$$\boxed{p_i(x_i) \propto b_i(x_i)} \quad \boxed{p_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \propto b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})}$$

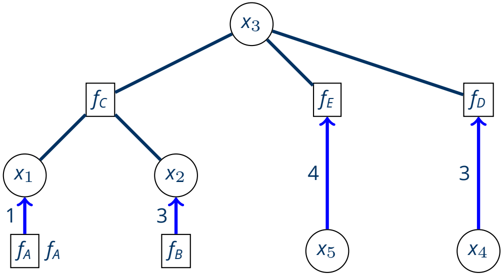
Belief Propagation Simulation



Belief Propagation Simulation: Step 1



Belief Propagation Simulation: Step 2



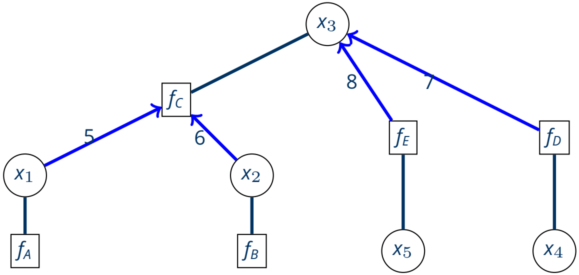
$$1 : \mu_{f_A \to x_1}(x_1) = f_A(x_1)$$

$$2 : \mu_{f_B \to x_2}(x_2) = f_B(x_1 2)$$

$$3 : \mu_{x_4 \to f_D}(x_4) = 1$$

$$4 : \mu_{x_5 \to f_E}(x_5) = 1$$

Belief Propagation Simulation: Step 3



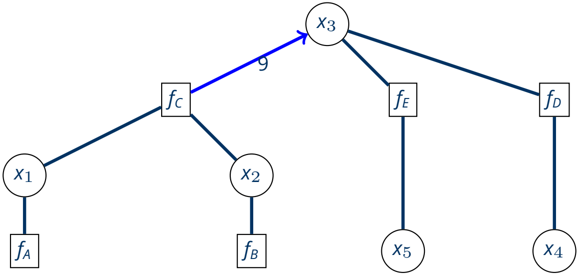
$$5: \quad \mu_{x_1 \to f_C}(x_1) = \mu_{f_A \to x_1}(x_1)$$

$$6: \quad \mu_{x_2 \to f_C}(x_2) = \mu_{f_B \to x_2}(x_2)$$

$$7: \quad \mu_{f_D \to x_3}(x_3) = \sum_{\sim x_3} f_D(x_3, x_4)\mu_{f_4 \to f_D}(x_4)$$

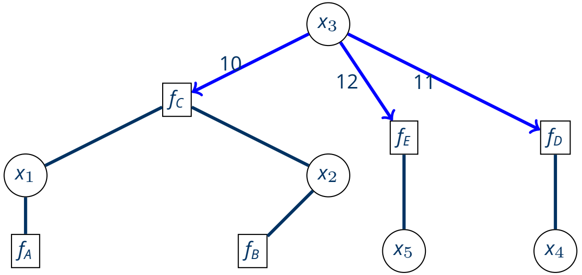
$$8: \quad \mu_{f_E \to x_3}(x_3) = \sum_{\sim x_3} f_D(x_3, x_5)\mu_{f_5 \to f_C}(x_5)$$

Belief Propagation Simulation: Step 4



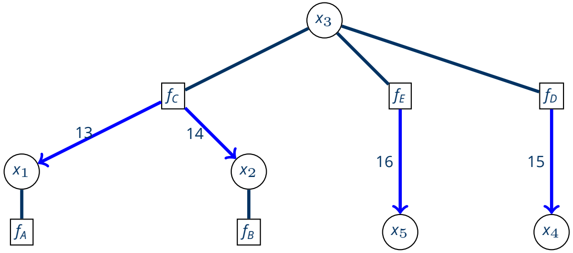
$$9: \ \mu_{f_C \to x_3}(x_3) = \sum_{\sim x_3} f_C(x_1, x_2, x_3)\mu_{x_1 \to f_C}(x_1)\mu_{x_2 \to f_c}(x_2)$$

Belief Propagation Simulation: Step 5



$$10: \quad \mu_{x_3 \to f_D}(x_3) = \mu_{f_D \to x_3}(x_3)\mu_{f_E \to x_3}(x_3)$$
$$11: \quad \mu_{x_3 \to f_D}(x_3) = \mu_{f_C \to x_3}(x_3)\mu_{f_E \to x_3}(x_3)$$
$$12: \quad \mu_{x_3 \to f_E}(x_3) = \mu_{f_C \to x_3}(x_3)\mu_{f_D \to x_3}(x_3)$$

Belief Propagation Simulation: Step 6



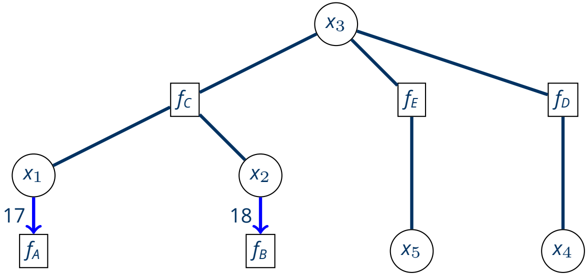
$$13: \quad \mu_{f_C \to x_1}(x_1) = \sum_{\sim x_1} f_C(x_1, x_2, x_3)\mu_{x_2 \to f_C}(x_2)\mu_{x_3 \to f_C}(x_3)$$

$$14: \quad \mu_{f_C \to x_2}(x_2) = \sum_{\sim x_2} f_C(x_1, x_2, x_3)\mu_{x_1 \to f_C}(x_1)\mu_{x_3 \to f_C}(x_3)$$

$$15: \quad \mu_{f_D \to x_4}(x_4) = \sum_{\sim x_4} f_D(x_3, x_4)\mu_{x_3 \to f_D}(x_3)$$

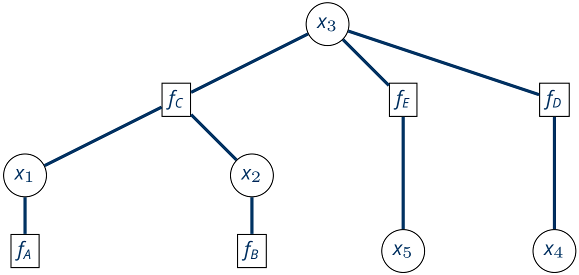
$$16: \quad \mu_{f_E \to x_5}(x_5) = \sum_{\sim x_5} f_E(x_3, x_5)\mu_{x_3 \to f_E}(x_3)$$

Belief Propagation Simulation: Step 7



$$17: \ \mu_{x_1 \to f_A}(x_1) = \mu_{f_C \to x_1}(x_1)$$
$$18: \ \mu_{x_2 \to f_B}(x_2) = \mu_{f_C \to x_2}(x_2)$$

Belief Propagation Simulation: Termination



$$g_1(x_1) = \mu_{f_A \to x_1}(x_1) \cdot \mu_{f_C \to x_1}(x_1)$$
$$g_2(x_2) = \mu_{f_B \to x_2}(x_2) \cdot \mu_{f_C \to x_2}(x_2)$$
$$g_3(x_3) = \mu_{f_C \to x_3}(x_3) \cdot \mu_{f_E \to x_3}(x_3) \cdot \mu_{f_D \to x_3}(x_3)$$
$$g_4(x_4) = \mu_{f_D \to x_4}(x_4)$$
$$g_5(x_5) = \mu_{f_E \to x_4}(x_5)$$

Max Sum Algorithm

Goals:

1. Find the maximum of the joint distribution by propagating messages from the leaves to an arbitrarily chosen root node
2. Finding the configuration of the variables for which the joint distribution attains this maximum value

- We designate a particular variable node as the 'root' of the graph. Then we start a set of messages propagating inwards from the leaves of the tree towards the root, with each node sending its message towards the root once it has received all incoming messages from its other neighbours. The final maximization is performed over the product of all messages arriving at the root node, and gives the maximum value for $p(x)$. This is be called the max-product algorithm and is identical to the sum-product algorithm except that summations are replaced by maximizations.

Max Sum Algorithm

$$\mu_{f_s \to x} = \max_{x_1 \ldots x_M} \left[Inf(x, x_1, \ldots, x_M) + \sum_{m \in ne(f) \setminus x} \mu_{x_m \to f}(x_m) \right] \quad (11)$$

$$\mu_{x \to f}(x) = \sum_{l \in ne(x) \setminus f} \mu_{f_l \to x}(x)$$

(12)

$$\mu_{x \to f}(x) = 0 \quad (13)$$

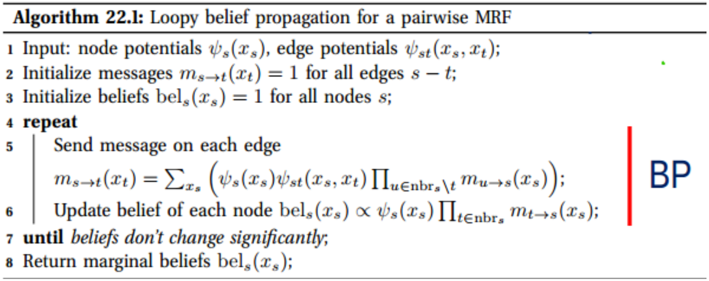
$$\mu_{f \to x}(x) = Inf(x) \quad (14)$$

Max Sum Algorithm

- Finding the configuration of the variables with maximum joint probability
- If a message is sent from a factor node *f* to a variable node *x*, a maximization is performed over all other variable nodes x_1 , $\cdot$,x_M that are neighbours of that factor node, using Equation 11.
- When we perform this maximization, we keep a record of which values of the variables x_1 , $\cdot$,x_M gave rise to the maximum.
- Then, in the back-tracking step, having found x^{max} , we can then use these stored values to assign consistent maximizing states x_1^{max}, $\cdot$,x_M^{max} .
- The max-sum algorithm, with back-tracking, gives an exact maximizing configuration for the variables provided the factor graph is a tree.

Loopy Belief Propagation

- The general idea is to run BP on a graph containing loops
- The fact that the presence of loops does not guarantee convergence



Algorithm 22.1: Loopy belief propagation for a pairwise MRF

```
Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
Initialize messages $m_{s \to t}(x_t) = 1$ for all edges $s - t$;
Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes $s$;
repeat
    Send message on each edge
    $m_{s \to t}(x_t) = \sum_{x_s} \left( \psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \to s}(x_s) \right)$;
    Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \to s}(x_s)$;
until beliefs don't change significantly;
Return marginal beliefs $\text{bel}_s(x_s)$;
```

- When the solution converges, it is usually a good approximation.

Linearized Belief Propagation

- Gatterbauer et al. [1] has derived a matrix formulation of BP like as following,

$$B = E + ABH$$

  where, B - beliefs, E - prior belief or unary potential or prior belief, A - adjacent matrix, and H - pairwise compatibility or pairwise factor matrix
- it iteratively compute the beliefs (like gradients)
- it ensures the exact convergence guarantees (even on graphs with loops)
- it showed promising performances in node classification

[1] Linearized and Single-Pass Belief Propagation, https://arxiv.org/abs/1406.7288

References

1. Pattern Recognition and Machine Learning by Chris Bishop
2. Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman
3. Machine Learning: A Probabilistic Perspective by Kevin P. Murphy
4. https://ermongroup.github.io/cs228-notes/
5. http://www.cs.cmu.edu/ mgormley/courses/10708/index.html
6. https://cedar.buffalo.edu/ srihari/CSE674/index.html