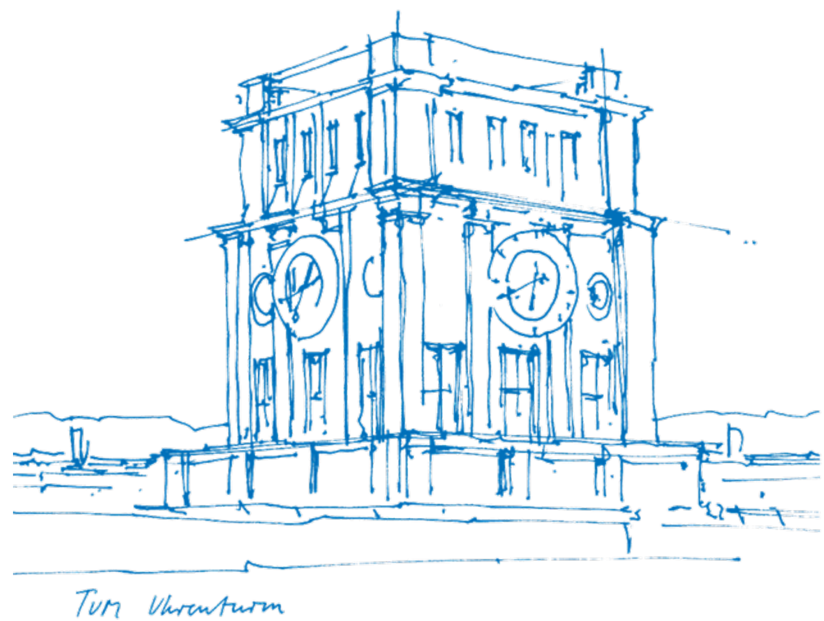
III : Inference on Graphical Models

Tao Wu, Yuesong Shen, Zhenzhang Ye

Computer Vision & Artificial Intelligence
Technical University of Munich



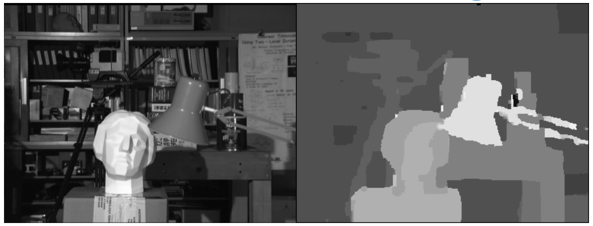
Motivation

– Many computer vision tasks boil down to inference on graphical models.

Denoising

Optical flow

Stereo matching



Inpainting

Super-resolution

1. **Probabilistic inference**: compute marginal distribution

$$p(y) = \sum_x p(y, x).$$

2. **MAP inference**: compute maximum of posterior distribution

$$\arg\max_y p(y|x).$$

Exact Inference

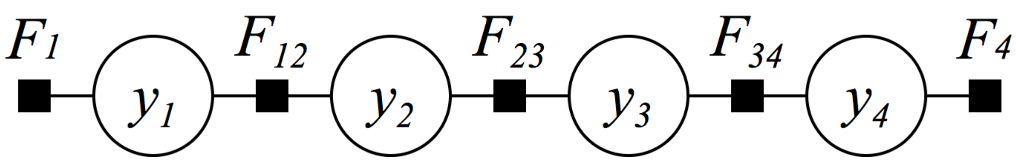
Outline of the Section

- Basic idea: Variable elimination.
- Junction tree algorithm on arbitrary MRFs.
- Belief propagation on tree factor graphs.

Example: Marginal Query on a "Chain" MRF

Joint distribution represented by MRF:

$$p(y_1, y_2, y_3, y_4) = \frac{1}{Z}\phi_1(y_1) \cdot \phi_{12}(y_1, y_2) \cdot \phi_{23}(y_2, y_3) \cdot \phi_{34}(y_3, y_4) \cdot \phi_4(y_4).$$



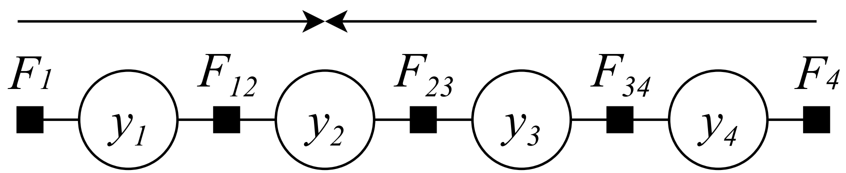
Query about marginal distribution $p(y_2) = ?$

Variable Elimination

Apply **variable elimination** (VE) to the marginal query:

$$
\begin{aligned}
p(y_2) &= \sum_{y_1,y_3,y_4} p(y_1, y_2, y_3, y_4) \\
&= \sum_{y_1,y_3,y_4} \frac{1}{Z}\phi_1(y_1)\phi_{12}(y_1, y_2)\phi_{23}(y_2, y_3)\phi_{34}(y_3, y_4)\phi_4(y_4) \\
&= \frac{1}{Z} \underbrace{\sum_{y_1} \Big(\phi_1(y_1)\phi_{12}(y_1, y_2)\Big)}_{=:\, m_{1\to 2}(y_2)} \sum_{y_3} \Big(\phi_{23}(y_2, y_3) \underbrace{\sum_{y_4} \big(\phi_{34}(y_3, y_4)\phi_4(y_4)\big)}_{=:\, m_{4\to 3}(y_3)} \Big) \\
&= \frac{1}{Z} m_{1\to 2}(y_2) \underbrace{\sum_{y_3} \Big(\phi_{23}(y_2, y_3) m_{4\to 3}(y_3)\Big)}_{=:\, m_{3\to 2}(y_2)} \\
&= \frac{1}{Z} m_{1\to 2}(y_2) m_{3\to 2}(y_2), \\
Z &= \sum_{y_2} m_{1\to 2}(y_2) m_{3\to 2}(y_2).
\end{aligned}
$$

Variable Elimination and Beyond



- This algorithm is called **sum-product** VE.
- Sum-product VE yields *exact* inference (of one node marginal) on any *tree-structured factor graph*.
- Observed nodes (a.k.a. *evidence*) can be introduced as reduced factors.
- A similar algorithm can be derived for MAP inference – simply switch all "sum" to "max". The resulting algorithm is called **max-product** VE.
- We shall consider two different extensions beyond VE:

1. Inference on arbitrary MRFs? $\rightsquigarrow$ **Junction tree algorithm**.
2. Compute all node/factor marginals at one shot? $\rightsquigarrow$ **Belief propagation**.

Junction Tree

- For an undirected graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the **junction tree** of $\mathcal{H}$ is a tree $\mathcal{T}$ s.t.
  1. The nodes of $\mathcal{T}$ consist of the *maximal cliques* of $\mathcal{H}$.
  2. The edge S_{ij} between two nodes C_i, C_j of $\mathcal{T}$ (i.e. two maximal cliques of $\mathcal{H}$) is given by $S_{ij} = C_i \cap C_j$ (known as the *running intersection property*).
- $\mathcal{H}$ is **triangulated** if every cycle of length ≥ 4 has a *chord*. (A chord is an edge that is not part of the cycle but connects two vertices of the cycle.)
- Theorem [Lauritzen '96]: A graph has a junction tree iff it is triangulated.

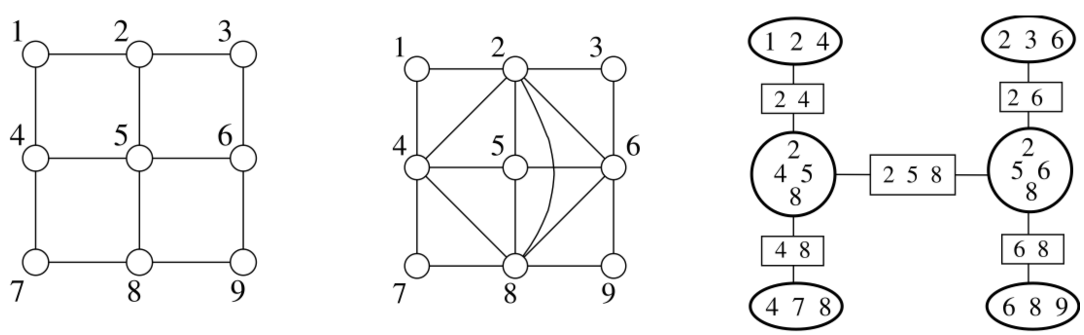
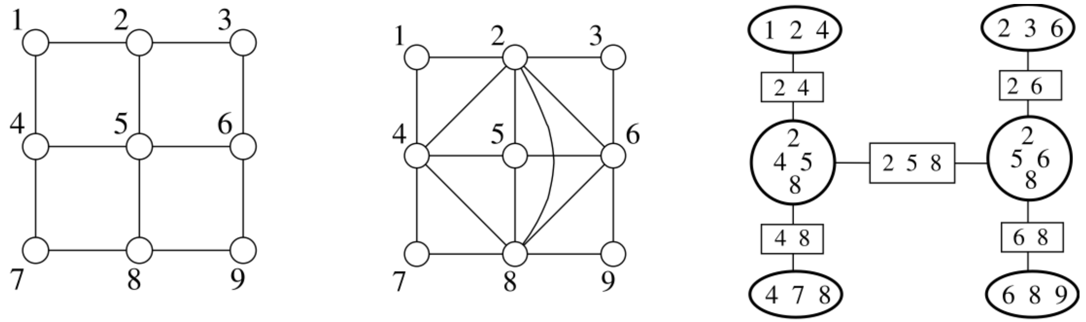


Figure:[1] (a) Original graph; (b) Triangulation of (a); (c) Junction tree for (b).

[1] Wainwright and Jordan, "Graphical Models, Exponential Families, and Variational Inference".

Junction Tree Algorithm (Sketch)



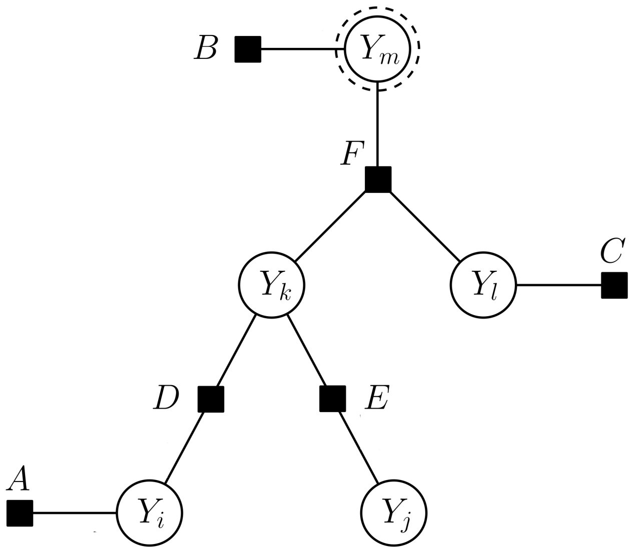
Sum-product message passing on a junction tree $\mathcal{T}$ appears like:

$$m_{C_i \to C_j}(y_{C_j \cap C_i}) = \sum_{y_{C_i \setminus C_j}} \phi_{C_i}(y_{C_i}) \prod_{C_k \in \mathrm{nbr}_{\mathcal{T}}(C_i) \setminus \{C_j\}} m_{C_k \to C_i}(y_{C_i \cap C_k}).$$

Overall **junction tree algorithm** for exact inference on an arbitrary MRF:

1. Given an MRF with cycles, triangulate it by adding edges as necessary.
2. Form a junction tree $\mathcal{T}$ for the triangulated MRF.
3. Run VE on the junction tree $\mathcal{T}$.

Belief Propagation on Tree Factor Graphs[2]



- Factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$: assumed to be a tree.
- Neighbors of a variable or factor node:

$$\text{nbr}_{\mathcal{G}}(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\},$$
$$\text{nbr}_{\mathcal{G}}(F) = \{i \in \mathcal{V} : (i, F) \in \mathcal{E}\}.$$

- (Log-domain) energies: $E_F(y_F) = -\log \phi_F(y_F)$.

[2] Illustrations for BP are extracted from Nowozin & Lampert, 2011.

BP: Leaf-to-Root Stage

0. Pick $Y_r \in \mathcal{V}$ as the tree root (e.g. Y_m in the figure).

1a. Schedule the leaf-to-root messages.

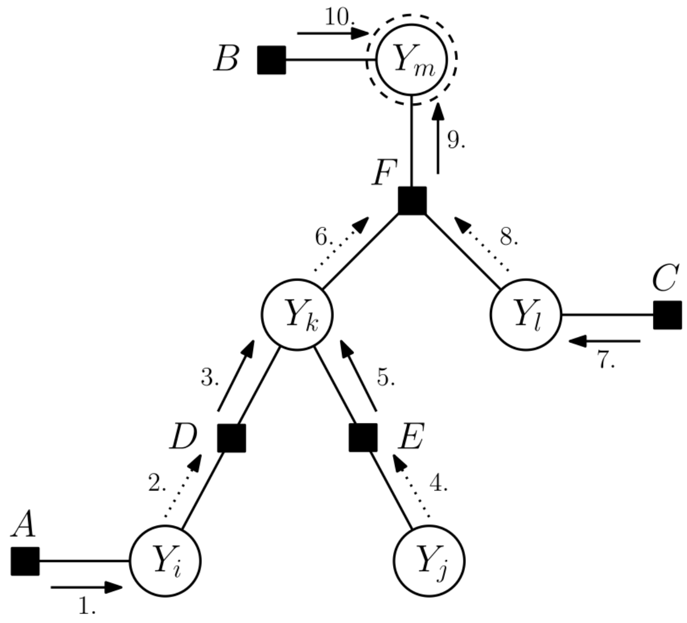


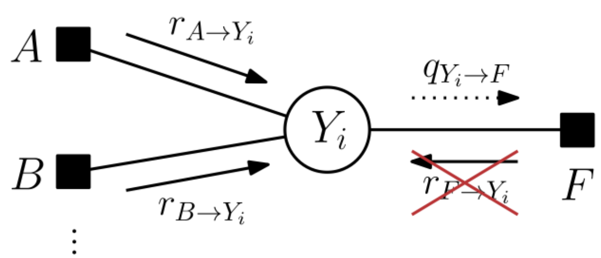
Figure: Belief propagation: leaf-to-root stage.

1b. Compute all leaf-to-root messages (detailed in the next slide).

BP: Compute Messages

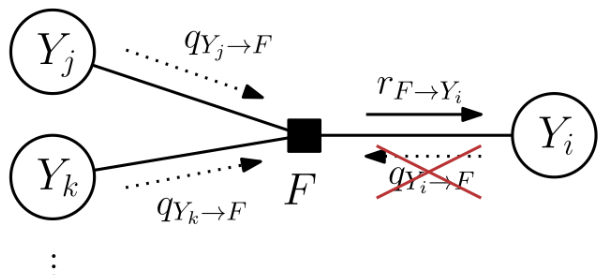
- Compute variable-to-factor message:

$$q_{i \to F}(y_i) = \sum_{F' \in \mathrm{nbr}_{\mathcal{G}}(i) \setminus \{F\}} r_{F' \to i}(y_i).$$



- Compute factor-to-variable message:

$$r_{F \to i}(y_i) = \log \sum_{y_{F \setminus \{i\}}} \exp \left(- E_F(y_F) + \sum_{i' \in \mathrm{nbr}_{\mathcal{G}}(F) \setminus \{i\}} q_{i' \to F}(y_{i'}) \right).$$



BP: Compute the Partition Function

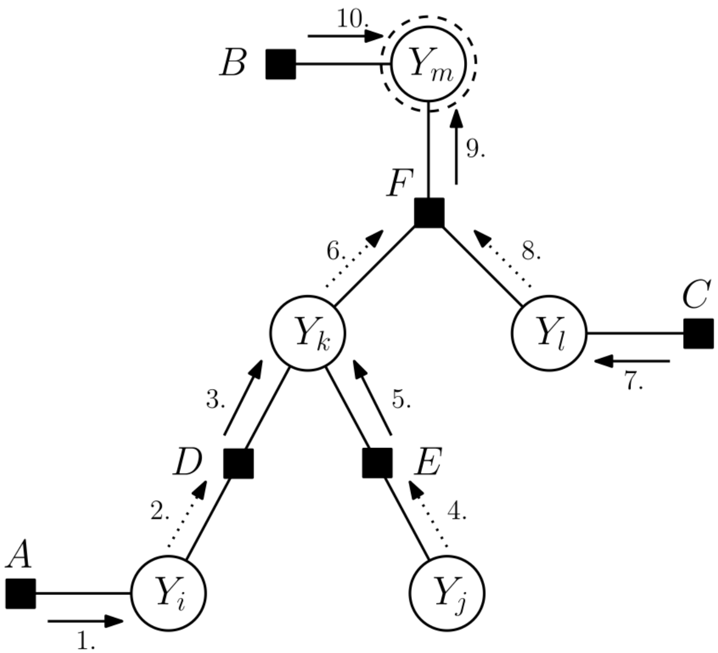


Figure: Belief propagation: leaf-to-root stage.

1c. Compute the log partition function:

$$\log Z = \log \sum_{y_r} \exp \left(\sum_{F \in \mathrm{nbr}_{\mathcal{G}}(r)} r_{F \to r}(y_r) \right).$$

BP: Root-to-Leaf Stage

2a. Schedule the root-to-leaf messages.

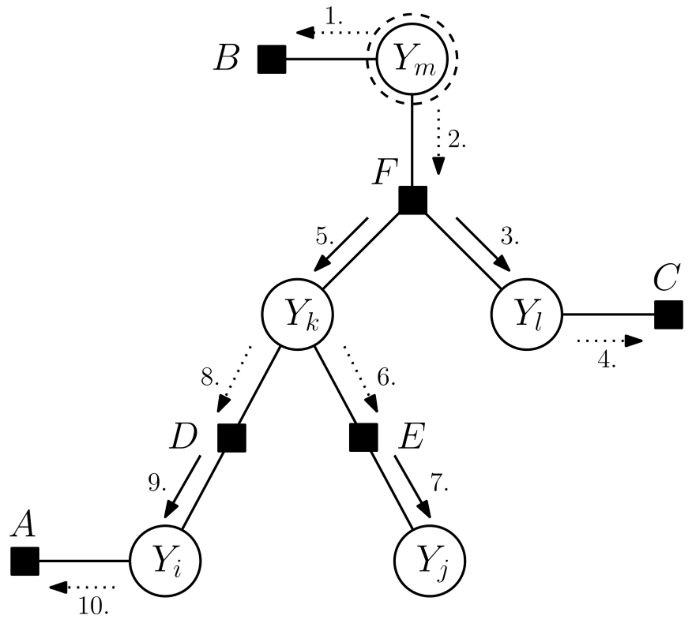


Figure: Belief propagation: root-to-leaf stage.

2b. Compute the root-to-leaf messages using the same formulas on page 12.

BP: Compute Factor / Variable Marginals

2c. Alongside Step 2b, combine messages and compute factor marginals:

$$\mu_F(y_F) := p(y_F) = \exp\Big(-E_F(y_F) + \sum_{i \in \text{nbr}_\mathcal{G}(F)} q_{i \to F}(y_i) - \log Z \Big),$$

as well as variable marginals:

$$\mu_i(y_i) := p(y_i) = \exp\Big(\sum_{F \in \text{nbr}_\mathcal{G}(i)} r_{F \to i}(y_i) - \log Z \Big).$$

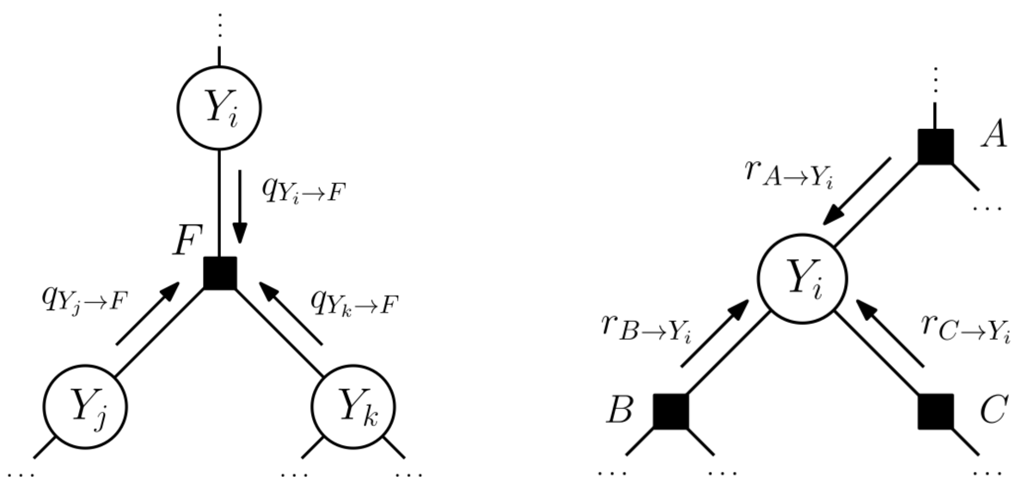


Figure: (left) Factor marginal; (right) Variable marginal.

BP on Pairwise MRFs (as exercise)

For a pairwise MRF $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the joint distribution is factorized by

$$p(y) = \exp\Big(-\sum_{i\in\mathcal{V}} E_i(y_i) - \sum_{(i,j)\in\mathcal{E}} E_{ij}(y_i, y_j) - \log Z\Big).$$

BP on such pairwise MRF can be simplified:

- Variable-to-variable message is computed by

$$m_{i\to j}(y_j) = \log \sum_{y_i} \exp\Big(-E_i(y_i) - E_{ij}(y_i, y_j) + \sum_{k\in \mathrm{nbr}_{\mathcal{H}}(i)\setminus\{j\}} m_{k\to i}(y_i)\Big).$$

- Variable marginal is computed by

$$\mu_i(y_i) = \exp\Big(-E_i(y_i) + \sum_{k\in \mathrm{nbr}_{\mathcal{H}}(i)} m_{k\to i}(y_i) - \log Z\Big).$$

Further Reading

- Koller & Friedman, Chapters 9, 10.
- Murphy, Chapter 20.
- Nowozin & Lampert, Section 3.1.

Variational Inference

Outline of this Section

- Basic idea: Variational inference.
- Mean field (MF) method.
- Loopy belief propagation (LBP).

Approximation by Tractable Distributions

- Goal: probabilistic inference on joint distribution $p(y)$ represented by *general* MRF (i.e. possibly with loops).

- Instead of tackling the inference on p directly, we first seek for an approximation q within a family $\mathcal{Q}$ consisting of "tractable" distributions:

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\,(q \,|\, p)\,.$$

- The **Kullback-Leibler (KL) divergence** (a.k.a. *relative entropy*) between two distributions q, p (assuming the "absolute continuity" $q \ll p$) is defined by

$$\mathrm{KL}\,(q \,|\, p) = \sum_{y} q(y) \log \frac{q(y)}{p(y)}.$$

- Basic properties of KL:
  1. $\mathrm{KL}\,(q \,|\, p) = 0$ iff $p = q$.
  2. $\mathrm{KL}\,(q \,|\, p) \geq 0 \; \forall q, p$.
  3. $\mathrm{KL}\,(\cdot \,|\, \cdot)$ is not symmetric. Nor does it satisfy the triangle inequality.

Preliminaries to Variational Inference

- Represented by a factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, p takes the form

$$p(y) = \exp\Big(-\sum_{F \in \mathcal{F}} E_F(y_F) - \log Z\Big).$$

- Plug p into KL divergence $\rightsquigarrow$

$$\begin{aligned}\mathrm{KL}\,(q \,|\, p) &= \sum_y q(y) \log \frac{q(y)}{p(y)} = \sum_y q(y) \log q(y) - \sum_y q(y) \log p(y) \\ &= -H(q) + \sum_{F \in \mathcal{F}} \sum_{y_F} \mu_F[q](y_F) E_F(y_F) + \log Z.\end{aligned}$$

- $H(q)$ is the **entropy** of distribution q.
- $\mu_F[q]$ is the marginal distribution of q over variables Y_F.
- $F_{\text{Gibbs}}(q;p) := \mathrm{KL}\,(q \,|\, p) - \log Z = -H(q) + \sum_{F \in \mathcal{F}} \sum_{y_F} \mu_F[q](y_F) E_F(y_F)$ is called the **Gibbs free energy**.
- $\mathrm{KL}\,(q \,|\, p) \geq 0 \;\Rightarrow\;$ $\log Z$ is lower bounded by $-F_{\text{Gibbs}}(q;p)$.

Mean Field Approximation

In (naive) **mean field** method, $\mathcal{Q}$ consists of q factorized by only unaries:

$$q(y) = \prod_{i \in \mathcal{V}} q_i(y_i).$$

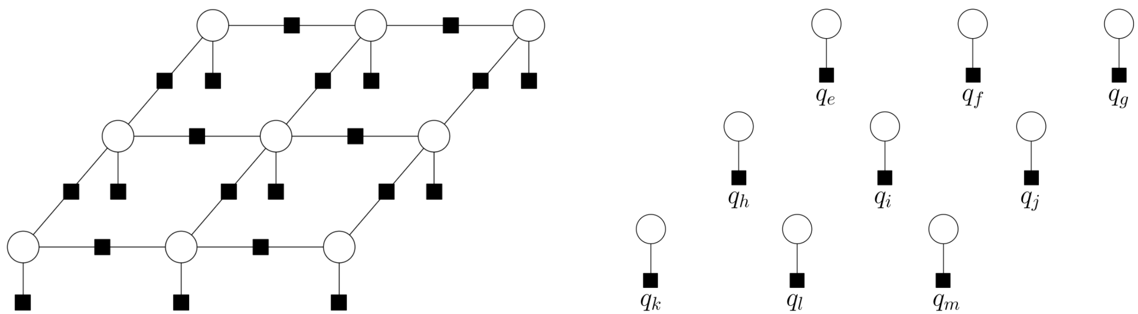


Figure: (left) Original factor graph; (right) (Naive) mean field approximation.

- Such q is "tractable" because $\{q_i(y_i)\}$ provide variable marginals.
- Quick facts:

$$H(q) = \sum_{i \in \mathcal{V}} H(q_i) = -\sum_{i \in \mathcal{V}} \sum_{y_i} q_i(y_i) \log q_i(y_i),$$

$$\mu_F[q](y_F) = \prod_{i \in \mathrm{nbr}_{\mathcal{G}}(F)} q_i(y_i).$$

Mean Field (MF) Approximation

Derivation of MF approximation:

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\,(q \,|\, p) = \arg\min_{q \in \mathcal{Q}} F(q; p)$$

$$= \arg\min_{q \in \mathcal{Q}} -H(q) + \sum_{F \in \mathcal{F}} \sum_{y_F} \mu_F[q](y_F) E_F(y_F)$$

$$= \arg\min_{\{q_i\}_{i \in \mathcal{V}}} \sum_{i \in \mathcal{V}} \sum_{y_i} q_i(y_i) \log q_i(y_i) + \sum_{F \in \mathcal{F}} \sum_{y_F} \Bigg(\prod_{i \in \mathrm{nbr}_{\mathcal{G}}(F)} q_i(y_i) \Bigg) E_F(y_F).$$

Each q_i lies in the probability simplex Δ_i, i.e.

$$q_i(y_i) \geq 0 \quad \forall y_i,$$

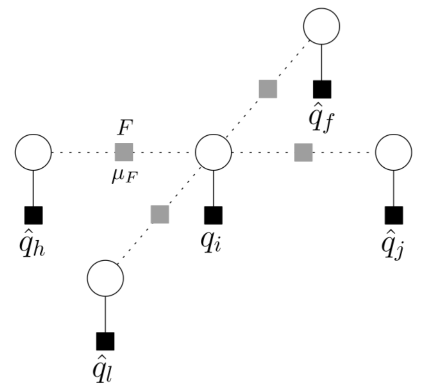
$$\sum_{y_i} q_i(y_i) = 1.$$

The optimization can be resolved by *coordinate descent* (next slide).

MF Update Formula

For each block q_i, fix $\widehat{q}_{i'}(y_{i'}) = q_{i'}(y_{i'})\ \forall i' \neq i$ and solve:

$$q_i^* = \arg\min_{q_i \in \Delta_i} \sum_{y_i} q_i(y_i) \log q_i(y_i) + \sum_{F \in \text{nbr}_{\mathcal{G}}(i)} \sum_{y_F} \left(\prod_{i' \in \text{nbr}_{\mathcal{G}}(F) \setminus \{i\}} \widehat{q}_{i'}(y_{i'}) \right) q_i(y_i) E_F(y_F).$$



We obtain an analytical solution via Lagrange multiplier λ for $\sum_{y_i} q_i^*(y_i) = 1$:

$$q_i^*(y_i) = \exp\left(-1 - \sum_{F \in \text{nbr}_{\mathcal{G}}(i)} \sum_{y_{F \setminus \{i\}}} \left(\prod_{i' \in \text{nbr}_{\mathcal{G}}(F) \setminus \{i\}} \widehat{q}_{i'}(y_{i'}) \right) E_F(y_F) + \lambda \right)$$

$$\propto \exp\left(- \sum_{F \in \text{nbr}_{\mathcal{G}}(i)} \sum_{y_{F \setminus \{i\}}} \left(\prod_{i' \in \text{nbr}_{\mathcal{G}}(F) \setminus \{i\}} \widehat{q}_{i'}(y_{i'}) \right) E_F(y_F) \right).$$

Some Remarks on MF

- The term $\prod_{i' \in \text{nbr}_{\mathcal{G}}(F) \backslash \{i\}} \widehat{q}_{i'}(y_{i'})$ is taken to be 1 if $\text{nbr}_{\mathcal{G}}(F) \backslash \{i\} = \emptyset$.
- For a pairwise MRF $\mathcal{H}$, the MF update rule can be simplified as

$$q_i^*(y_i) \propto \exp\left(- E_i(y_i) - \sum_{j \in \text{nbr}_{\mathcal{H}}(i)} \sum_{y_j} \widehat{q}_j(y_j) E_{ij}(y_i, y_j) \right).$$

- MF is an iterative procedure which converges to a *locally optimal* solution q^*.
- Upon convergence, $\{q_i^*\}$ directly provide (approximate) variable marginals.
- The tractable family $\mathcal{Q}$ can be more sophisticated than factorizations of unaries in naive mean field. $\rightsquigarrow$ *Structured mean field* approximation.

From Belief Propagation to Loopy Belief Propagation

- Previously we have seen how belief propagation works on tree factor graphs.
- We can use similar update rules to derive **loopy belief propagation** (LBP).
- Although LBP does not guarantee the convergence (if at all) to the true marginal, it often performs well and is widely used in practice[3].
- In the following, we first present the LBP algorithm and then interpret it from perspective of variational inference.

[3] Murphy et al., "Loopy Belief Propagation for Approximate Inference: An Empirical Study".

Loopy Belief Propagation

On a factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, (sum-product) LBP proceeds as follows.

0. Initialize all variable-to-factor messages: $q_{i\to F}(y_i) = 0$. Then iterate:
1. Update all factor-to-variable messages:

$$r_{F\to i}(y_i) = \log \sum_{y_{F\setminus\{i\}}} \exp\Big(-E_F(y_F) + \sum_{i'\in \mathrm{nbr}_{\mathcal{G}}(F)\setminus\{i\}} q_{i'\to F}(y_{i'})\Big).$$

2. Update all (normalized) variable-to-factor messages:

$$\bar{q}_{i\to F}(y_i) = \sum_{F'\in \mathrm{nbr}_{\mathcal{G}}(i)\setminus\{F\}} r_{F'\to i}(y_i),$$

$$\delta_{i\to F} = \log \sum_{y_i} \exp\Big(\bar{q}_{i\to F}(y_i)\Big),$$

$$q_{i\to F}(y_i) = \bar{q}_{i\to F}(y_i) - \delta_{i\to F}.$$

Loopy Belief Propagation (cont'd)

3. Update all factor marginals (beliefs):

$$\mu_F(y_F) \propto \exp\Big(-E_F(y_F) + \sum_{i \in \text{nbr}_\mathcal{G}(F)} q_{i \to F}(y_i) \Big).$$

4. Update all variable marginals (beliefs):

$$\mu_i(y_i) \propto \exp\Big(\sum_{F \in \text{nbr}_\mathcal{G}(i)} r_{F \to i}(y_i) \Big).$$

Differences compared to BP:

- The normalization constants in the computation of marginals differ at each factor/variable.
- The log partition function is not directly available, but it can be approximated by the **Bethe free energy**:

$$-\log Z \approx F_{\text{Bethe}}(\mu; p) := \sum_{i \in \mathcal{V}} (1 - |\text{nbr}_\mathcal{G}(i)|) \sum_{y_i} \mu_i(y_i) \log \mu_i(y_i) + \sum_{F \in \mathcal{F}} \sum_{y_F} \mu_F(y_F) \Big(E_F(y_F) + \log \mu_F(y_F) \Big).$$

Interpretation of LBP

On a pairwise MRF $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, LBP can be interpreted as an attempt to solve:

$$\underset{\{\mu_i\}_{i\in\mathcal{V}},\, \{\mu_{ij}\}_{(i,j)\in\mathcal{E}}}{\text{minimize}} \sum_{i\in\mathcal{V}} (1 - |\,\text{nbr}_{\mathcal{H}}(i)|) \sum_{y_i} \mu_i(y_i) \log \mu_i(y_i)$$
$$+ \sum_{(i,j)\in\mathcal{E}} \sum_{y_i, y_j} \mu_{ij}(y_i, y_j)\Big(E_{ij}(y_i, y_j) + \log \mu_{ij}(y_i, y_j)\Big)$$

$$\text{subject to } \mu_i(y_i) \geq 0,\ \ \mu_{ij}(y_i, y_j) \geq 0,\ \sum_{y_i} \mu_i(y_i) = 1,\ \sum_{y_i} \mu_{ij}(y_i, y_j) = \mu_j(y_j).$$

- The constraints impose *local consistency* between node marginals $\{\mu_i\}$ and edge marginals $\{\mu_{ij}\}$.
- However, $\{\mu_i\}, \{\mu_{ij}\}$ under these constraints are may not be marginals of any joint distribution on $\mathcal{H}$ (i.e. outer approximation of *marginal polytope*).
- LBP updates can be derived from an iterative algorithm for the above constrained optimization.
- An amazing theory on variational inference arise in this context — we point those interested to the "monster" paper [Jordan & Wainwright, 2008].

LBP vs. MF

(+) (Naive) MF optimizes over only variable marginals; LBP optimizes over variable and factor marginals under local consistency constraints.

(+) LBP does exact inference on factor graphs without loops; MF is exact on a strict subclass of factor graphs, on which all true factor marginals are factorized by $\mu_F(y_F) = \prod_{i \in \text{nbr}_{\mathcal{G}}(F)} \mu_i(y_i)$ (hence an inner approximation of marginal polytope).

(+) While both being approximate inference techniques, LBP tends to be more accurate than MF in practice.

(−) MF provides a lower bound of the log partition function (given by negative Gibbs free energy), while LBP does not.

(−) Compared to LBP, it is easier to extend MF to distributions other than discrete and Gaussian, due to the simplicity of working with only variable marginals.

Further Reading

- Murphy, Chapters 21, 22.
- Nowozin & Lampert, Sections 3.2, 3.3.
- Koller & Friedman, Chapter 11.
- Jordan & Wainwright, Chapters 4, 5.

Sampling-based Inference

Outline of the Section

- Monte Carlo (MC) method.
- Markov chain Monte Carlo (MCMC) method.
- Sampling of Bayesian network and Markov random field.

Basic Principle of Sampling

Given a distribution p, we can approximate p using a finite sequence of **samples** $\{x_n\}_{n=1}^N$ in the sense that:

$$\mathbb{E}_{x \sim p}[f(x)] = \sum_x f(x)p(x) \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n) \quad \text{for any function } f.$$

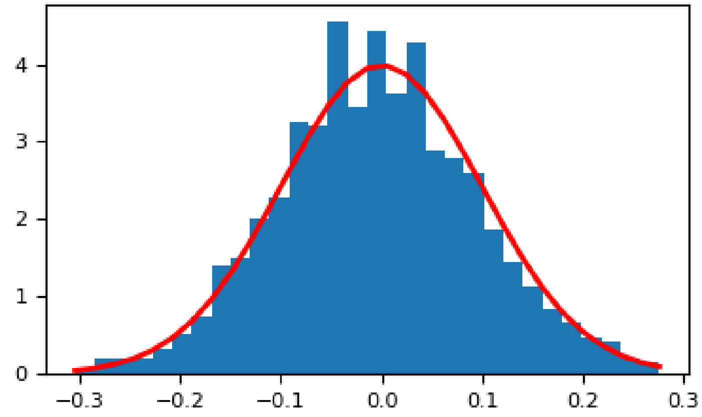


Figure: Sampling of a Gaussian[4].

[4]https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.normal.html

Pseudo-Random Number Generator

Linear congruential generator for sampling Unif(0, 1):

$$x_{n+1} = (a \cdot x_n + c) \mod m.$$

- Most fundamental sampler above all.
- The generated samples are *pseudo-random* — $\{x_n\}$ are "deterministic" if the generator (i.e. parameters a, c, m) and the *seed* x_0 are fixed.

Source	modulus m	multiplier a	increment c	output bits of seed in *rand()* or *Random(L)*
Numerical Recipes	2^{32}	1664525	1013904223	
Borland C/C++	2^{32}	22695477	1	bits 30..16 in *rand()*, 30..0 in *lrand()*
glibc (used by GCC)[9]	2^{31}	1103515245	12345	bits 30..0
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++ [10] C90, C99, C11: Suggestion in the ISO/IEC 9899 [11], C18	2^{31}	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2^{32}	134775813	1	bits 63..32 of *(seed * L)*
Turbo Pascal	2^{32}	134775813 ($0x8088405_{16}$)	1	
Microsoft Visual/Quick C/C++	2^{32}	214013 ($343FD_{16}$)	2531011 ($269EC3_{16}$)	bits 30..16
Microsoft Visual Basic (6 and earlier)[12]	2^{24}	1140671485 ($43FD43FD_{16}$)	12820163 ($C39EC3_{16}$)	
RtlUniform from Native API[13]	$2^{31} - 1$	2147483629 ($7FFFFFED_{16}$)	2147483587 ($7FFFFFC3_{16}$)	
Apple CarbonLib, C++11's `minstd_rand0` [14]	$2^{31} - 1$	16807	0	see MINSTD
C++11's `minstd_rand` [14]	$2^{31} - 1$	48271	0	see MINSTD
MMIX by Donald Knuth	2^{64}	6364136223846793005	1442695040888963407	
Newlib, Musl	2^{64}	6364136223846793005	1	bits 63...32
VMS's **MTH$RANDOM**,[15] old versions of glibc	2^{32}	69069 ($10DCD_{16}$)	1	
Java's java.util.Random, POSIX [ln]rand48, glibc [ln]rand48[_r]	2^{48}	25214903917 ($5DEECE66D_{16}$)	11	bits 47...16

Figure: Commonly used linear congruential generators[5].

[5]https://en.wikipedia.org/wiki/Linear_congruential_generator

Sampling Gaussians

- Sample univariate Gaussian distribution by **Box-Muller method**:
  1. Sample $(z_1, z_2) \sim p_z(z_1, z_2) = \frac{1}{\pi}\mathbf{1}\{z_1^2 + z_2^2 \leq 1\}$ (i.e. uniform distribution supported on the unit 2D circle).
  2. Perform the Box-Muller transformation and output x_1, x_2:

$$x_i = z_i \sqrt{\frac{-2\log(z_1^2 + z_2^2)}{z_1^2 + z_2^2}}, \quad i \in \{1, 2\}.$$

  Fact: x_1, x_2 are two i.i.d. samples under Normal(0, 1):

$$p_x(x_1, x_2) = p_z(z_1, z_2) \left| \frac{\partial(z_1, z_2)}{\partial(x_1, x_2)} \right| = \frac{1}{\sqrt{2\pi}} \exp(-x_1^2/2) \cdot \frac{1}{\sqrt{2\pi}} \exp(-x_2^2/2).$$

- Sample multivariate Gaussian distribution, $y \sim \text{Normal}(\mu, \Sigma)$, by:
  1. Perform Cholesky decomposition $\Sigma = LL^\top$.
  2. Sample $x \sim \text{Normal}(0, I)$, and output $y := Lx + \mu$.

  Fact: $\mathbb{E}[y] = \mu$, and $\text{Var}[y] = L\,\text{Var}[x]L^\top = LIL^\top = \Sigma$.

Sampling by Inverse CDF

Sample by **inverse Cumulative Distribution Function**:

- Let $u \sim \text{Unif}(0,1)$ and F_p be the CDF for (univariate) distribution p, i.e.

$$F_p(y) := \int_{-\infty}^{y} p(x)dx = \int_{-\infty}^{\infty} \mathbf{1}\{x \leq y\} p(x)dx.$$

- Note that $x \sim p \Leftrightarrow P(x \leq y) = F_p(y)$.

- We assert $F_p^{-1}(u) \sim p$, since

$$\begin{aligned} P(F_p^{-1}(u) \leq y) &= P(u \leq F_p(y)) && \text{(since } F_p \text{ is monotone)} \\ &= F_p(y). && \text{(since } P(u \leq v) = v \;\; \forall v \in [0,1]\text{)} \end{aligned}$$

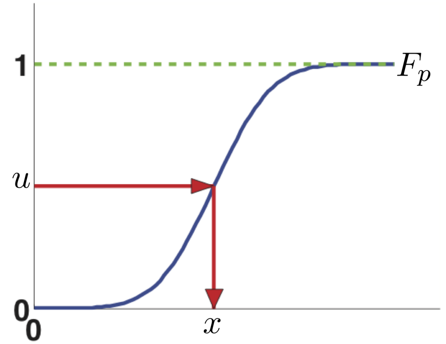


Figure: Sampling using inverse CDF [Murphy, Figure 23.1].

Rejection Sampling

- Inverse CDF sampling requires explicit knowledge of F_p^{-1}.
- **Rejection Sampling**:
  Require: *unnormalized* target distribution $\widetilde{p}$ (i.e. $\widetilde{p}(x)/Z_p = p(x)$ for target distribution p), *proposal distribution* q and constant $M > 0$ s.t. $Mq(x) \geq \widetilde{p}(x)\ \forall x$ ($\Rightarrow p \ll q$).

1. Sample $x \sim q$, and $u \sim \text{Unif}(0, 1)$.
2. If $u > \frac{\widetilde{p}(x)}{Mq(x)}$, reject the proposed sample x; otherwise, accept x.

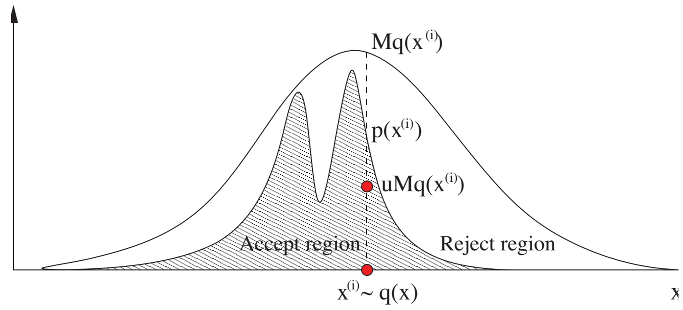


Figure: Rejection sampling [Murphy, Figure 23.2].

- Proof: (univariate case) $P(x \leq y | x \text{ accepted}) = \frac{P(x \leq y,\ x \text{ accepted})}{P(x \text{ accepted})} =$

$$\frac{\iint \mathbf{1}\{u \leq \widetilde{p}(x)/(Mq(x)),\ x \leq y\} q(x) du\, dx}{\iint \mathbf{1}\{u \leq \widetilde{p}(x)/(Mq(x))\} q(x) du\, dx} = \frac{\frac{1}{M}\int_{-\infty}^{y} \widetilde{p}(x) dx}{\frac{1}{M}\int_{-\infty}^{\infty} \widetilde{p}(x) dx} = F_p(y).$$

Importance Sampling

- In rejection sampling, $P(x \text{ accepted}) = \frac{1}{M}\int_{-\infty}^{\infty} \widetilde{p}(x)dx$, i.e., many proposed samples are potentially wasted.
- In contrast, **importance sampling** uses all samples by weighting them:

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} w_n f(x_n),$$

with $x_n \sim q$ i.i.d. and $w_n = \frac{p(x_n)}{q(x_n)}$.

- Extend importance sampling to *unnormalized* distributions $\widetilde{p}$, $\widetilde{q}$:

$$\mathbb{E}_{x\sim p}[f(x)] = \frac{Z_q}{Z_p}\int f(x)\frac{\widetilde{p}(x)}{\widetilde{q}(x)}q(x)dx \approx \frac{Z_q}{Z_p}\frac{1}{N}\sum_{n=1}^{N}\frac{\widetilde{p}(x_n)}{\widetilde{q}(x_n)}f(x_n), \quad x_n \sim q \text{ i.i.d.}$$

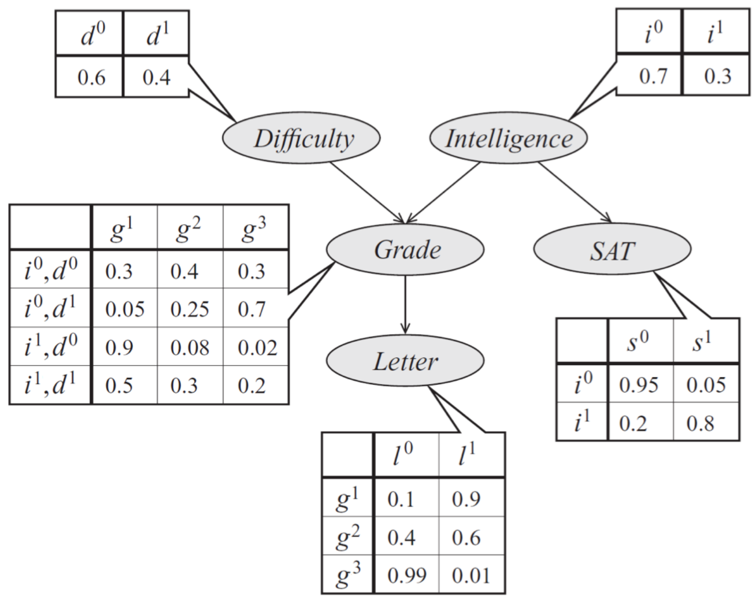
$$\frac{Z_p}{Z_q} = \int \frac{1}{Z_q}\widetilde{p}(x)dx = \int \frac{\widetilde{p}(x)}{\widetilde{q}(x)}q(x)dx \approx \frac{1}{N}\sum_{n=1}^{N}\frac{\widetilde{p}(x'_n)}{\widetilde{q}(x'_n)}, \quad x'_n \sim q \text{ i.i.d.}$$

We often take $x'_n = x_n$. For finite N, this yields a *biased estimator* of p.

Sampling of Bayesian Network

Recall that the distribution represented by BN is given by

$$p(x) = \prod_{i \in \mathcal{V}} p(x_i | (x_j)_{j \in \mathrm{Pa}_{\mathcal{G}}(i)}).$$



Ancestral sampling: Given that no variables are observed, we can follow the topological order of the BN and sample each individual conditional distribution.

Sampling of BN with Evidence

In case the BN $\mathcal{G}$ contains observed nodes (called **evidence**), we can modify ancestral sampling (AS) as follows:

- **Logic sampling**: Perform AS. Whenever a sampled node takes different value from the evidence, reject the whole sample and start again.
- LS is closely related to rejection sampling. Unsurprisingly, it is inefficient for wasting samples.
- **Likelihood weighting**: Perform AS. Whenever node i is observed (written $i \in \mathcal{O}$), we *clamp* the observed value, i.e. $x_i := \bar{x}_i$, and *weight* the whole sample x by the probability of the clamped node $p(\bar{x}_i | x_{\mathrm{Pa}_{\mathcal{G}}(i)})$.
- LW can be interpreted as importance sampling with weights given by:

$$w(x) = \frac{\widetilde{p}(x)}{q(x)} = \frac{\mathbf{1}\{x_{\mathcal{O}} = \bar{x}_{\mathcal{O}}\} \prod_{i \in \mathcal{V}} p(x_i | x_{\mathrm{Pa}_{\mathcal{G}}(i)})}{\prod_{i \in \mathcal{V} \setminus \mathcal{O}} p(x_i | x_{\mathrm{Pa}_{\mathcal{G}}(i)}) \prod_{i \in \mathcal{O}} \delta_{\bar{x}_i}(x_i)} = \prod_{i \in \mathcal{O}} p(\bar{x}_i | x_{\mathrm{Pa}_{\mathcal{G}}(i)}).$$

$\delta_{\bar{x}}$ denotes the **Dirac distribution** defined by $\delta_{\bar{x}}(x) = \begin{cases} 1 & \text{if } x = \bar{x}, \\ 0 & \text{otherwise.} \end{cases}$

Towards Markov Chain Monte Carlo

- Monte Carlo sampling requires exact or rough knowledge of the partition function (of an MRF), hence impractical for high dimensional distributions.
- Instead of generating i.i.d. samples, **Markov Chain Monte Carlo** (MCMC) constructs a *Markov chain* using *"adaptive" proposal distributions*, in a way that the Markov chain converges to a *stationary distribution* identical to the target distribution.

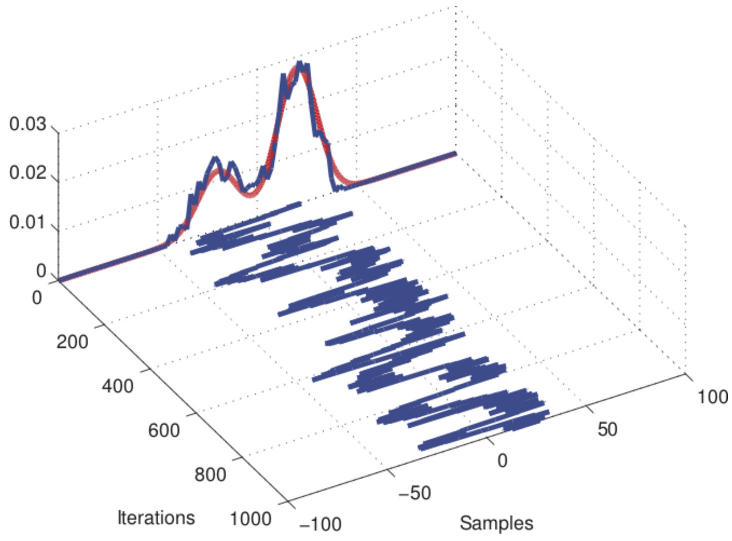


Figure: Sampling by MCMC [Murphy, Figure 24.7].

Markov Chain

- The (discrete-time) **Markov chain** (MC) is a sequence of RVs $(X_n)_{n=1}^{\infty}$ satisfying the Markov property:

$$P(X_{n+1} = x | X_1, ..., X_n \text{ given}) = P(X_{n+1} = x | X_n \text{ given}).$$

"The future depends on the past only through the present."

- Further assume:
  1. All X_n has a *finite state space* $\mathcal{X}$.
  2. The MC is *time-homogeneous*, i.e., the transition probability is time-independent

$$P(X_{n+1} = x' | X_n = x) =: \pi(x'|x) \quad \forall n,$$

with $\pi(x'|x) \geq 0,\ \sum_{x'} \pi(x'|x) = 1$. π is the **transition kernel** of the MC.

- Denote by p_n the distribution at time step n:

$$p_n(x) = P(X_n = x) \quad \Rightarrow \ p_{n+1}(x') = \sum_x p_n(x)\pi(x'|x).$$

Relevant Notions on Markov Chain

- p_* is a **stationary distribution** for the MC if

$$p_*(x') = \sum_x p_*(x)\pi(x'|x) \;\; \forall x' \in \mathcal{X}.$$

- The MC is **irreducible** if

$$\forall x, x' \in \mathcal{X} \;\; \exists n(x, x') \;\; \text{s.t. } P(X_n = x'|X_0 = x) > 0,$$

i.e., it is possible to get to any state from any state in finite steps.

- A state $x \in \mathcal{X}$ has *period* T_x if

$$T_x = \gcd\{n > 0 : P(X_n = x|X_0 = x) > 0\}, \quad \text{\# "greatest common divisor"}$$

i.e., any loop over state x must occur in a multiple of T_x steps.
We say the MC is **aperiodic** if $T_x = 1 \; \forall x \in \mathcal{X}$.

- The MC is **regular** if

$$\exists n \;\; \text{s.t. } P(X_n = x'|X_0 = x) > 0 \;\; \forall x, x' \in \mathcal{X}.$$

Fact: MC is regular $\Rightarrow$ MC is irreducible and aperiodic.

Convergence to Stationary Distribution

Theorem 1: If the transition kernel π of a Markov chain satisfies the *detailed balance condition* for some distribution p_*:

$$p_*(x)\pi(x'|x) = p_*(x')\pi(x|x') \quad \forall x, x' \in \mathcal{X},$$

then p_* is a stationary distribution for the Markov chain.

Proof: $\sum_x p_*(x)\pi(x'|x) = \sum_x p_*(x')\pi(x|x') = p_*(x')\sum_x \pi(x|x') = p_*(x')$.

Theorem 2[6]: Every irreducible, aperiodic, finite-state Markov chain has a limiting distribution

$$p_*(x') = \lim_{n\to\infty} \sum_x P(X_n = x'|X_0 = x)p_0(x),$$

regardless of the initial distribution p_0. Indeed, p_* is equal to the unique stationary distribution of the MC.

[6] [Murphy, Theorem 17.2.1]

Metropolis-Hastings Algorithm

Metropolis-Hastings (MH) algorithm:

Input: unnormalized target distribution $\widetilde{p}$ (i.e. $p_*(x) = \widetilde{p}(x)/Z_p$), proposal distribution $q(\cdot|\cdot)$, initial sample x_0. Loop $n = 0, 1, 2, \ldots$ as follows:

1. Set $x = x_n$. Sample $x' \sim q(x'|x)$.
2. Compute acceptance probability $\alpha = \dfrac{\widetilde{p}(x')q(x|x')}{\widetilde{p}(x)q(x'|x)}$.
3. Compute $r = \min(1, \alpha)$. Sample $u \sim \text{Unif}(0, 1)$.
4. Set new sample to: $x_{n+1} = \begin{cases} x' & \text{if } u < r, \\ x_n & \text{if } u \geq r. \end{cases}$

Some remarks:

- For a given target distribution p_*, a proposal distribution q is valid if $\text{supp}(p_*) \subset \cup_x \text{supp}(q(\cdot|x))$, i.e. $\forall x'$ with $p_*(x') > 0 \; \exists x$ s.t. $q(x'|x) > 0$.
- If q is symmetric, i.e. $q(x'|x) = q(x|x')$, then MH simplifies to the Metropolis algorithm with $\alpha = \frac{\widetilde{p}(x')}{\widetilde{p}(x)}$. Hastings made the correction for asymmetric q.

Analysis of MH Algorithm

We analyze with convergence of the MH algorithm:

1. MH generates a Markov chain with the transition kernel:

$$\pi(x'|x) = \begin{cases} q(x'|x)r(x'|x) & \text{if } x' \neq x, \\ q(x|x) + \sum_{x' \neq x} q(x'|x)(1 - r(x'|x)) & \text{if } x' = x. \end{cases}$$

   $r(x'|x)$ is the conditional probability that x' is accepted after being proposed. We will show that the Markov chain satisfies the detailed balance condition:

$$p_*(x)\pi(x'|x) = p_*(x')\pi(x|x').$$

2. Let two states x and x' ($x \neq x'$) be arbitrarily fixed. Either

$$p_*(x)\pi(x'|x) \leq p_*(x')\pi(x|x'), \qquad (\dagger)$$

   or the reversed inequality holds. Without loss of generality, we proceed with inequality ($\dagger$).

Computer Vision & Artificial Intelligence
Department of Informatics
Technical University of Munich

Analysis of MH Algorithm (cont'd)

$$p_*(x)\pi(x'|x) \leq p_*(x')\pi(x|x'). \qquad (\dagger)$$

3. $(\dagger) \;\Rightarrow\; \alpha(x'|x) = \dfrac{p_*(x')q(x|x')}{p_*(x)q(x'|x)} \leq 1 \;\Rightarrow\; r(x'|x) = \alpha(x'|x)$

   $\Rightarrow\; \pi(x'|x) = q(x'|x)r(x'|x) = q(x'|x)\dfrac{p_*(x')q(x|x')}{p_*(x)q(x'|x)} = \dfrac{p_*(x')}{p_*(x)}q(x|x').$

4. $(\dagger) \;\Rightarrow\; \alpha(x|x') = \dfrac{p_*(x)q(x'|x)}{p_*(x')q(x|x')} \geq 1 \;\Rightarrow\; r(x|x') = 1$

   $\Rightarrow\; \pi(x|x') = q(x|x')r(x|x') = q(x|x').$

5. Combining (3) and (4), we conclude that $p_*(x)\pi(x'|x) = p_*(x')\pi(x|x')$. Hence, by Theorem 1, p_* is a stationary distribution for the Markov chain.

6. If in addition the Markov chain generated by the MH algorithm is irreducible and aperiodic, then by Theorem 2 the Markov chain converges to the unique stationary distribution p_*.

Gibbs Sampling

Gibbs sampling:

Input: unnormalized target distribution $\widetilde{p}((x_i)_{i=1}^{|\mathcal{V}|})$, initial sample x^0.

Loop $n \in \{0, 1, 2, ...\}$:

Loop $i \in \{1, 2, ..., |\mathcal{V}|\}$:

Sample $x_i^{n+1} \sim p(x_i | x_{\{0,...,i-1\}}^{n+1}, x_{\{i+1,...,|\mathcal{V}|\}}^{n})$.

Some remarks:

- If p (or $\widetilde{p}$) is represented by a graphical model (either BN or MRF), then sampling of x_i^{n+1} only involves the Markov blanket of i.
- Gibbs sampling can be interpreted as the MH algorithm with the proposal:

$$q(x'|x) = p(x'_i | x'_{\mathcal{V}\setminus\{i\}}) \delta_{x_{\mathcal{V}\setminus\{i\}}}(x'_{\mathcal{V}\setminus\{i\}}),$$

and 100% acceptance rate:

$$\alpha = \frac{p(x')q(x|x')}{p(x)q(x'|x)} = \frac{p(x'_i|x'_{\mathcal{V}\setminus\{i\}})p(x'_{\mathcal{V}\setminus\{i\}})p(x_i|x_{\mathcal{V}\setminus\{i\}})\delta_{x'_{\mathcal{V}\setminus\{i\}}}(x_{\mathcal{V}\setminus\{i\}})}{p(x_i|x_{\mathcal{V}\setminus\{i\}})p(x_{\mathcal{V}\setminus\{i\}})p(x'_i|x'_{\mathcal{V}\setminus\{i\}})\delta_{x_{\mathcal{V}\setminus\{i\}}}(x'_{\mathcal{V}\setminus\{i\}})} = 1.$$

Example: Gibbs Sampling for Pairwise CRF

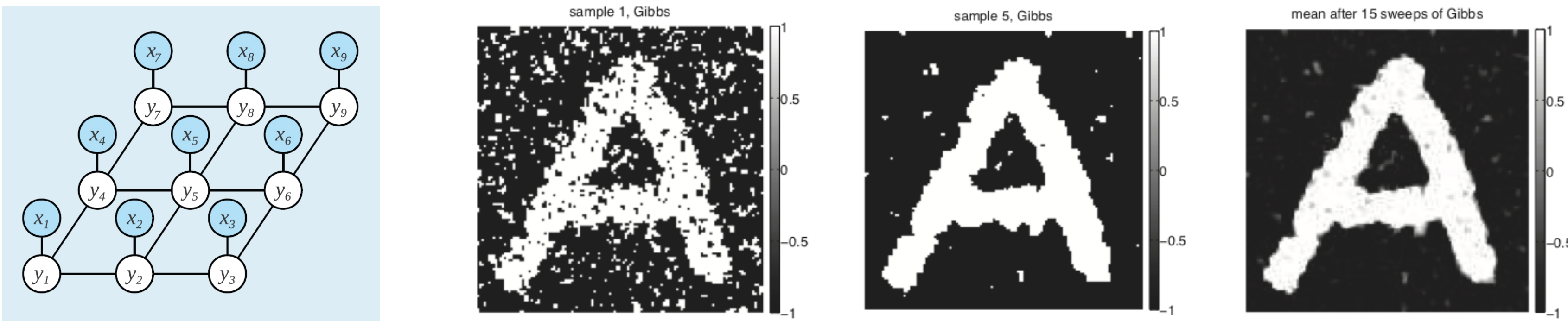


Figure: Gibbs Sampling for Pairwise CRF[7].

We can apply Gibbs sampling to find

$$y \sim p(y|x) \propto \exp\left(-\sum_{i \in \mathcal{V}} E_i(y_i; x_i) - \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j)\right).$$

For each $i \in \mathcal{V}$, sample (e.g. by inverse CDF method):

$$y_i^{n+1} \sim p(y_i|x_i, y^n_{\mathrm{nbr}(i)}) \propto \exp\left(-E_i(y_i; x_i) - \sum_{j \in \mathrm{nbr}(i)} E_{ij}(y_i, y_j^n)\right).$$

[7]Source of images: [Murphy, Figure 24.1].

Further Reading

- Murphy, Chapters 23, 24.
- Nowozin & Lampert, Section 3.4.
- Koller & Friedman, Chapter 12.

MAP Inference

More about MAP Inference

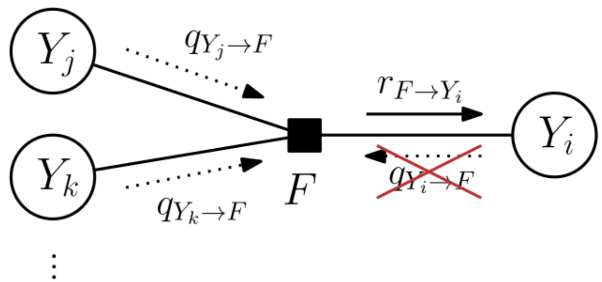
- So far this chapter has been focusing on probabilistic inference.
- MAP inference is about finding $\arg\max_y p(y)$ or $\arg\max_y p(y|x)$.
- To some extent, MAP inference is easier than probabilistic inference for the reason that the partition function Z (in the context of MRF) can be ignored in MAP inference.
- Probabilistic inference algorithms (e.g. variable elimination, (loopy) belief propagation) have analogs for MAP inference: sum-product $\rightarrow$ max-product.
- There also exist fast specialized MAP inference algorithms. We will show one such example: *graph-cut algorithm*.

Max-Product Loopy Belief Propagation

On a factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, the max-product LBP proceeds as follows.

0. Initialize all variable-to-factor messages: $q_{i \to F}(y_i) = 0$. Then iterate:
1. Update all factor-to-variable messages:

$$r_{F \to i}(y_i) = \max_{y_{F \setminus \{i\}}} \Big(-E_F(y_F) + \sum_{i' \in \mathrm{nbr}_{\mathcal{G}}(F) \setminus \{i\}} q_{i' \to F}(y_{i'}) \Big).$$



2. Update the max-beliefs:

$$\mu_i(y_i) = \sum_{F \in \mathrm{nbr}_{\mathcal{G}}(i)} r_{F \to i}(y_i),$$

and their maximizers $y_i^* = \arg\max_{y_i} \mu_i(y_i)$.

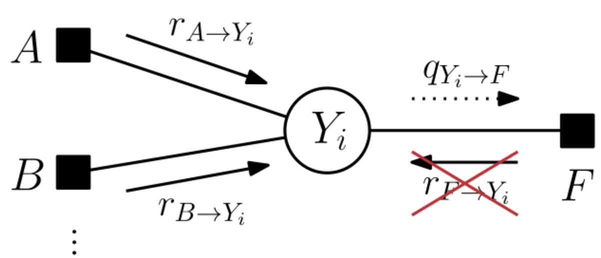
Max-Product Loopy Belief Propagation (cont'd)

3. Update all (normalized) variable-to-factor messages:

$$\bar{q}_{i\to F}(y_i) = \sum_{F'\in \text{nbr}_{\mathcal{G}}(i)\setminus\{F\}} r_{F'\to i}(y_i),$$

$$\delta_{i\to F} = \overline{\sum}_{y_i} \bar{q}_{i\to F}(y_i),$$ # $\overline{\sum}$ stands for averaged sum.

$$q_{i\to F}(y_i) = \bar{q}_{i\to F}(y_i) - \delta_{i\to F}.$$ # Normalization $\Rightarrow \sum_{y_i} q_{i\to F}(y_i) = 0.$



Some comments:

- Due to computation in log-domain, the above algorithm is sometimes called the *max-sum* loopy belief propagation.
- For tree factor graphs, max-product BP is exact upon completion of one leaf-to-root and one root-to-leaf message updates.

Graph-Cut Algorithm

- **Graph cut** algorithms can solve "certain" MAP inference tasks on MRFs in polynomial time. They are widely used in computer vision applications[8].
- Next we demonstrate graph cut on *binary-valued* pairwise MRF $(\mathcal{V}, \mathcal{E})$:

$$p(x) = \frac{1}{Z} \exp\Big(- \sum_{i \in \mathcal{V}} E_i(x_i) - \sum_{(i,j) \in \mathcal{E}} E_{ij}(x_i, x_j) \Big), \quad x \in \{0, 1\}^{\mathcal{V}}.$$

- Assume that all pairwise energies take the special form

$$E_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ \lambda_{ij} & \text{if } x_i \neq x_j, \end{cases}$$

  with $\lambda_{ij} \geq 0 \ \forall (i,j) \in \mathcal{E}$. This encourages neighboring nodes to have the same value. The overall model is called the "generalized Ising model".

- Also assume that $\forall i \in \mathcal{V}$: either $E_i(0) = 0,\ E_i(1) \geq 0$ or $E_i(1) = 0,\ E_i(0) \geq 0$.

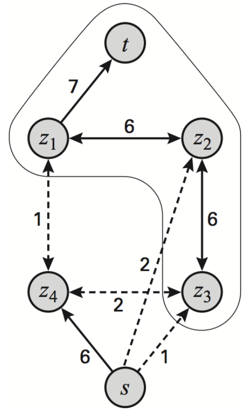
[8] Boykov and Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision".

Construction of Max-Flow/Min-Cut Problem

- Construct a graph such that:
  - The nodes are $\mathcal{V} \cup \{s, t\}$, where s is the *source* and t is the *sink*.
  - If $E_i(1) = 0$, introduce an edge $i \to t$ with cost $E_i(0)$.
  - If $E_i(0) = 0$, introduce an edge $s \to i$ with cost $E_i(1)$.
  - If $(i, j) \in \mathcal{E}$, introduce both edges $i \to j$ and $j \to i$ with cost λ_{ij}.
- The *st*-cut cost on the constructed graph is equal to the MRF energy:

$$\sum_{\substack{x,x' \in \mathcal{V} \cup \{s,t\} \\ x=0,\ x'=1}} \text{cost}(x, x') = \sum_{i \in \mathcal{V}} E_i(x_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(x_i, x_j).$$

- Compute a minimal *st*-cut, e.g. by Ford-Fulkerson algorithm or its variants.



Example (graph cut applied to MRF with 4 nodes):

$E_1(0) = 7,\ E_2(1) = 2,\ E_3(1) = 1,\ E_4(1) = 6,$
$\lambda_{12} = 6,\ \lambda_{23} = 6,\ \lambda_{34} = 2,\ \lambda_{14} = 1.$

Source: [Koller & Friedman, Figure 13.5].

Extension of Graph Cut to Submodular Energies

- We now extend graph cut to *binary-valued* pairwise MRF $(\mathcal{V}, \mathcal{E})$ with *submodular* energies.
- A pairwise energy $E_{ij}(x_i, x_j)$ is said to be **submodular** if
$$E_{ij}(1,1) + E_{ij}(0,0) \leq E_{ij}(0,1) + E_{ij}(1,0).$$
- Construct new energies as follows:
  Initialize $\widetilde{E}_i(\cdot) := E_i(\cdot)\ \forall i \in \mathcal{V},\ \widetilde{E}_{i,j}(\cdot,\cdot) := 0\ \forall (i,j) \in \mathcal{E}$.
  Loop $(i,j) \in \mathcal{E}$:
  $\widetilde{E}_i(1) := \widetilde{E}_i(1) + E_{ij}(1,0) - E_{ij}(0,0)$.
  $\widetilde{E}_j(1) := \widetilde{E}_j(1) + E_{ij}(1,1) - E_{ij}(1,0)$.
  $\widetilde{E}_{ij}(0,1) := E_{ij}(1,0) + E_{ij}(0,1) - E_{ij}(0,0) - E_{ij}(1,1)$.
- Construct a graph such that:
  - The nodes are $\mathcal{V} \cup \{s, t\}$, where s is the *source* and t is the *sink*.
  - If $\widetilde{E}_i(1) \geq \widetilde{E}_i(0)$, introduce an edge $s \to i$ with cost $\widetilde{E}_i(1) - \widetilde{E}_i(0)$.
  - If $\widetilde{E}_i(1) \leq \widetilde{E}_i(0)$, introduce an edge $i \to t$ with cost $\widetilde{E}_i(0) - \widetilde{E}_i(1)$.
  - If $(i,j) \in \mathcal{E}$ and $\widetilde{E}_{ij}(0,1) > 0$, introduce an edge $i \to j$ with cost $\widetilde{E}_{ij}(0,1)$.

Further Reading

Further reading:

- Murphy, Section 22.6.
- Koller & Friedman, Chapter 13.

Interesting topics that are not covered in the lecture:

- Extension of graph cut to non-binary-valued MRFs: alpha-expansion, alpha-beta swap.
- Linear programming relaxation, and its connection to max-product (loopy) belief propagation.
- Dual decomposition.