An Introduction to Bayesian Machine Learning

Daniel Hernández–Lobato[1],

June 29, 2015

slides by

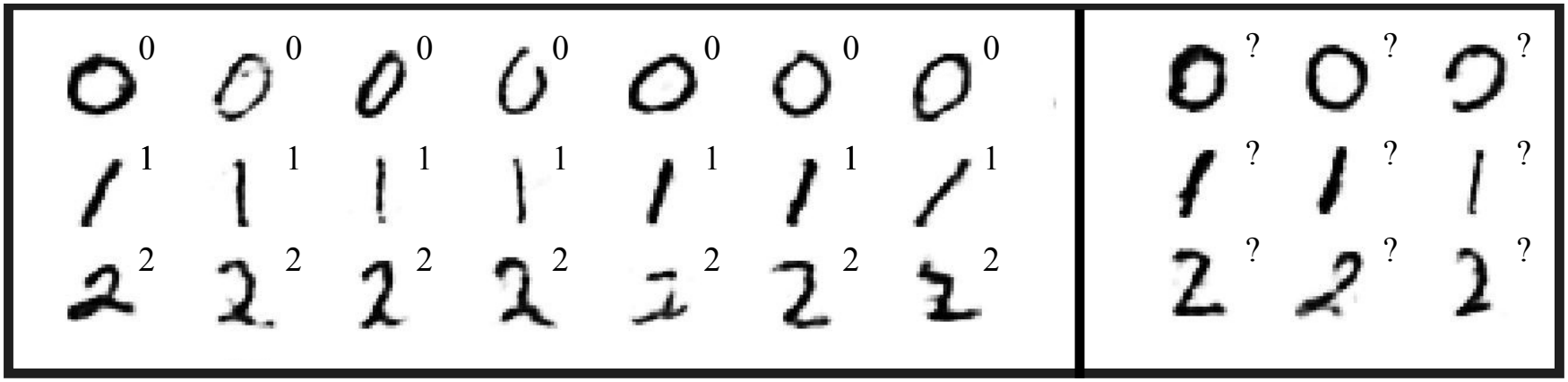
José Miguel Hernández-Lobato.

[1]Universidad Autónoma de Madrid.

What is Machine Learning?

The design of computational systems that discover patterns in a collection of data instances in an automated manner.

The ultimate goal is to use the discovered patterns to make predictions on new data instances not seen before.



Instead of manually encoding patterns in computer programs, we make computers learn these patterns without explicitly programming them.

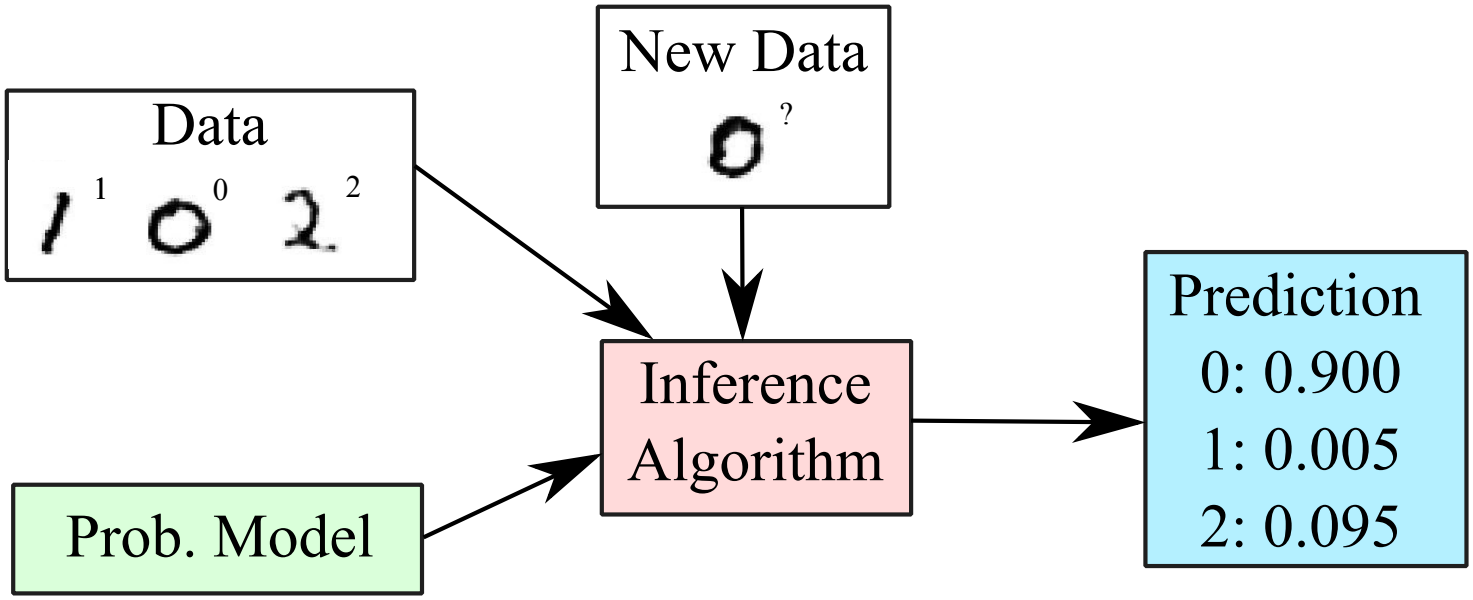
Figure source [Hinton et al. 2006].

Model-based Machine Learning

We design a probabilistic model which explains how the data is generated.

An inference algorithm combines model and data to make predictions.

Probabilities are used to deal with uncertainty in the model or the data.



Probabilistic programming languages such as Infer.Net or Church provide a powerful implementation of MML.

Basics of Probability Theory

Everything needed follows from just two rules:

Sum rule:

$$p(x) = \int p(x, y)\, dy\,.$$

Product rule:

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

They can be combined to obtain **Bayes' rule:**

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(y|x)p(y)}{\int p(x, y)\, dy}\,.$$

Independence of X and Y: $p(x, y) = p(x)p(y)$.

Conditional independence of X and Y given Z: $p(x, y|z) = p(x|z)p(y|z)$.

The Bayesian Framework

The probabilistic model $\mathcal{M}$ with parameters $\boldsymbol{\theta}$ explains how the data $\mathcal{D}$ is generated by specifying the likelihood function $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})$.

Our initial uncertainty on $\boldsymbol{\theta}$ is encoded in the prior distribution $p(\boldsymbol{\theta}|\mathcal{M})$.

Bayes' rule allows us to update our uncertainty on $\boldsymbol{\theta}$ given $\mathcal{D}$:

$$p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}.$$

We can then generate probabilistic predictions for some quantity y_{new} of a new data instance x_{new} given $\mathcal{D}$ and $\mathcal{M}$ using

$$p(y_{\text{new}}|x_{\text{new}}, \mathcal{D}, \mathcal{M}) = \int p(y_{\text{new}}|\boldsymbol{\theta}, x_{\text{new}}, \mathcal{M})p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})d\boldsymbol{\theta}\,.$$

These predictions will be obtained using an inference algorithm.

Bayesian Model Comparison

Given a particular $\mathcal{D}$, we can use the model evidence $p(\mathcal{D}|\mathcal{M})$ to reject both overly simple models, and overly complex models.

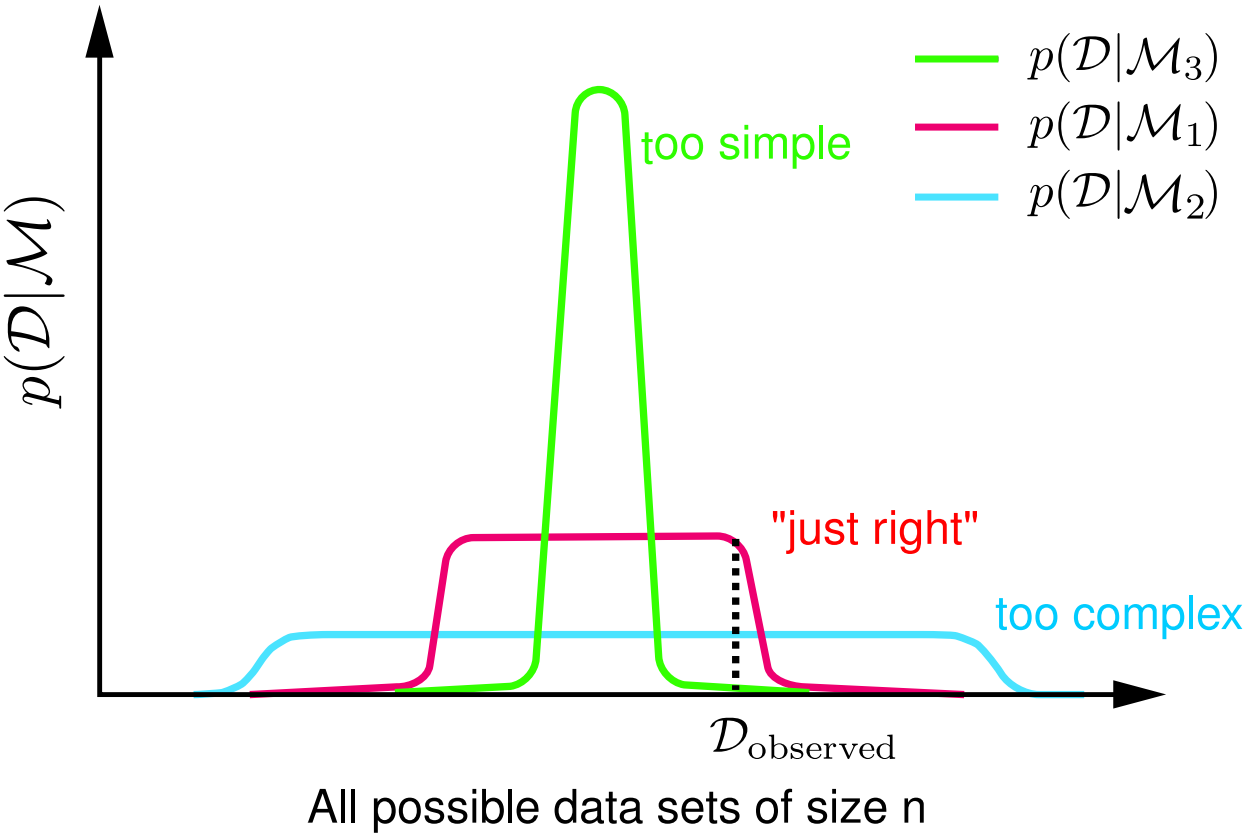


Figure source [Ghahramani 2012].

Probabilistic Graphical Models

The Bayesian framework requires to specify a high-dimensional distribution $p(x_1, \ldots, x_k)$ on the data, model parameters and latent variables.

Working with fully flexible joint distributions is intractable!

We will work with structured distributions, in which the random variables interact directly with only few others. These distributions will have many conditional independencies.

This structure will allow us to:

- Obtain a compact representation of the distribution.
- Use computationally efficient inference algorithms.

The framework of probabilistic graphical models allows us to represent and work with such structured distributions in an efficient manner.

Some Examples of Probabilistic Graphical Models

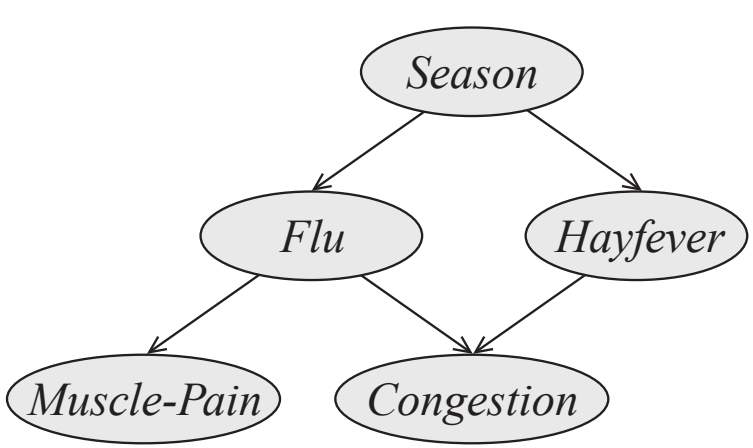
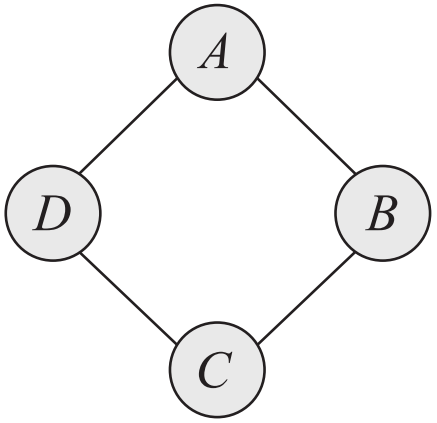
	Bayesian Network	Markov Network
Graphs		
Independencies	$(F \perp H \mid S)$, $(C \perp S \mid F, H)$ $(M \perp H, C \mid F)$, $(M \perp C \mid F)$, ...	$(A \perp C \mid B, D)$, $(B \perp D \mid A, C)$
Factorization	$p(S, F, H, M, C) = p(S)p(F \mid S)$ $p(H \mid S)p(C \mid F, H)p(M \mid F)$	$p(A, B, C, D) = \frac{1}{Z}\phi_1(A, B)$ $\phi_2(B, C)\phi_3(C, D)\phi_4(A, D)$

Figure source [Koller et al. 2009].

Bayesian Networks

A BN $\mathcal{G}$ is a DAG whose nodes are random variables $X_1, \ldots, X_d$.

Let $\mathrm{PA}_{X_i}^{\mathcal{G}}$ be the parents of X_i in $\mathcal{G}$.
The network is annotated with the conditional distributions $p(X_i|\mathrm{PA}_{X_i}^{\mathcal{G}})$.

Conditional Independencies:

Let $\mathrm{ND}_{X_i}^{\mathcal{G}}$ be the variables in $\mathcal{G}$ which are non-descendants of X_i in $\mathcal{G}$.

$\mathcal{G}$ encodes the conditional independencies $(X_i \perp \mathrm{ND}_{X_i}^{\mathcal{G}} | \mathrm{PA}_{X_i}^{\mathcal{G}}),\ i = 1, \ldots, d$.

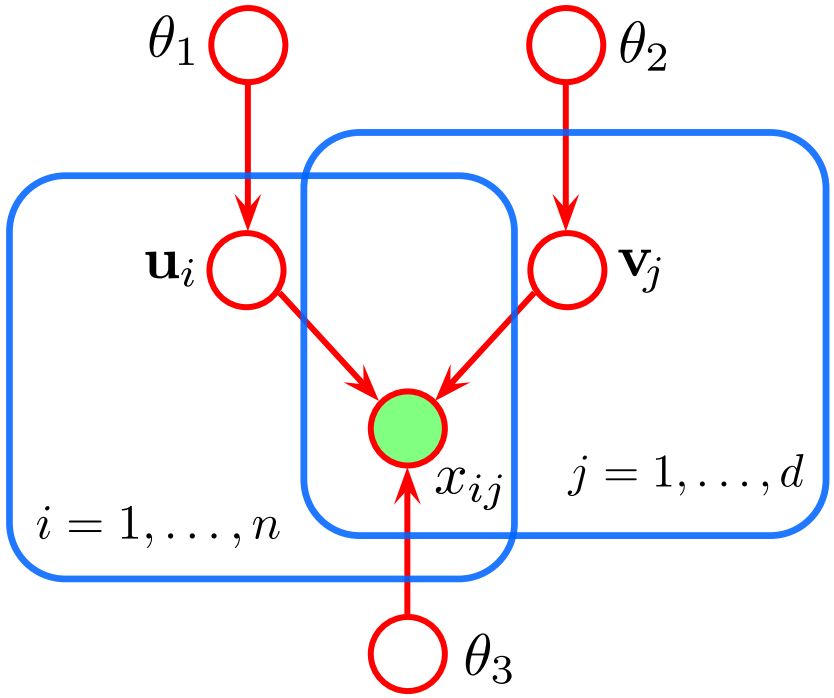
Factorization:

$\mathcal{G}$ encodes the factorization $p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i|\mathrm{pa}_{x_i}^{\mathcal{G}})$.

BN Examples: Matrix Factorization Model

We have observations $x_{i,j}$ from an $n \times d$ matrix $\mathbf{X}$.

The entries of $\mathbf{X}$ are generated as a function of the entries of a low rank matrix $\mathbf{U}\mathbf{V}^\mathsf{T}$, $\mathbf{U}$ is $n \times k$ and $\mathbf{V}$ is $d \times k$ and $k \ll \min(n, d)$.



$$p(\mathbf{X},\mathbf{U},\mathbf{V},\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\theta}_3) =$$
$$\left[\prod_{i=1}^{n}\prod_{j=1}^{d} p(x_{i,j}|\mathbf{u}_i,\mathbf{v}_j,\theta_3)\right]$$
$$\left[\prod_{i=1}^{n} p(\mathbf{u}_i|\theta_1)\right]\left[\prod_{i=j}^{d} p(\mathbf{v}_j|\theta_2)\right]$$
$$p(\theta_1)p(\theta_2)p(\theta_3)\,.$$

D-separation

Conditional independence properties can be read directly from the graph.

We say that the sets of nodes A, B and C satisfy $(A \perp B | C)$ when all of the possible paths from any node in A to any node in B are blocked.

A path will be blocked if it contains a node x with arrows meeting at x

1 - i) head-to-tail or ii) tail-to-tail and x is C.

2 - head-to-head and neither x, nor any of its descendants, is in C.

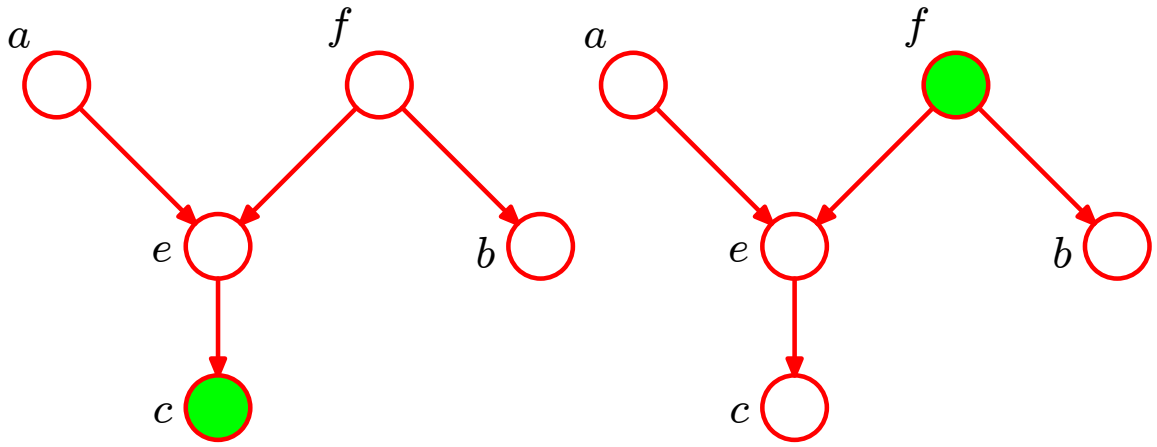


Figure source [Bishop 2006].

$(a \perp b | c)$ does not follow from the graph.

$(a \perp b | f)$ is implied by the graph.

Bayesian Networks as Filters

$p(x_1, \ldots, x_k)$ must satisfy the CIs implied by d-separation.

$p(x_1, \ldots, x_k)$ must factorize as $p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i | \text{pa}_{x_i}^{\mathcal{G}})$.

$p(x_1, \ldots, x_k)$ must satisfy the CIs $(X_i \perp \text{ND}_{X_i}^{\mathcal{G}} | \text{PA}_{X_i}^{\mathcal{G}})$, $i = 1, \ldots, d$.

The three filters are the same!

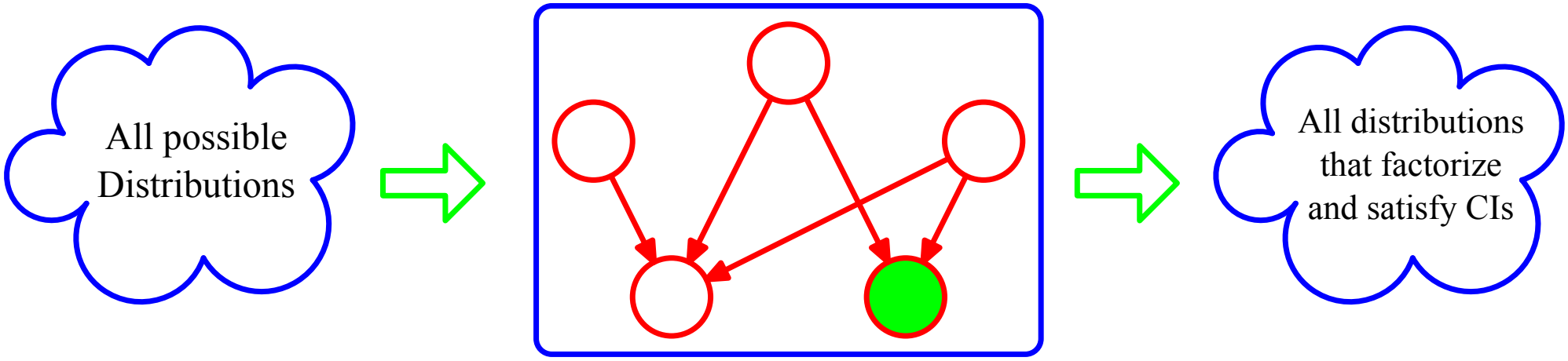


Figure source [Bishop 2006].

Markov Networks

A MN is an undirected graph $\mathcal{G}$ whose nodes are the r.v. $X_1, \ldots, X_d$.

It is annotated with the potential functions $\phi_1(\mathbf{D}_1), \ldots, \phi_k(\mathbf{D}_k)$, where $\mathbf{D}_1, \ldots, \mathbf{D}_k$ are sets of variables, each forming a maximal clique of $\mathcal{G}$, and $\phi_i, \ldots, \phi_k$ are positive functions.

Conditional Independencies:

$\mathcal{G}$ encodes the conditional independencies $(A \perp B | C)$ for any sets of nodes A, B and C such that C separates A from B in $\mathcal{G}$.

Factorization:

$\mathcal{G}$ encodes the factorization $p(X_1, \ldots, X_d) = Z^{-1} \prod_{i=1}^{k} \phi_i(\mathbf{D}_i)$, where Z is a normalization constant.

Clique and Maximal Clique

Clique: Fully connected subset of nodes.

Maximal Clique: A clique in which we cannot include any more nodes without it ceasing to be a clique.

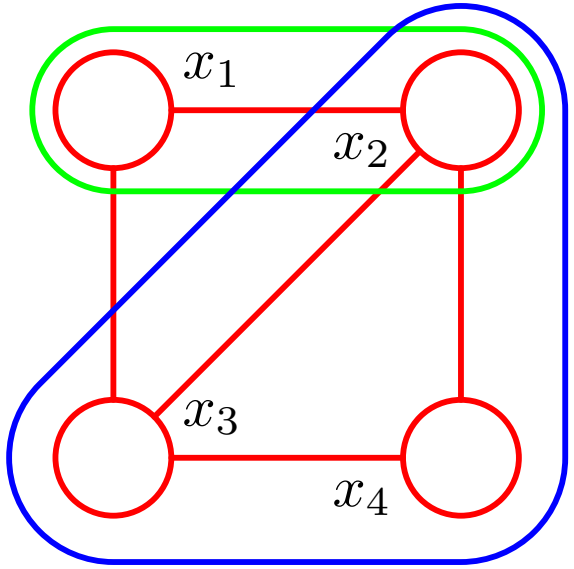


Figure source [Bishop 2006].

(x_1, x_2) is a clique but not a maximal clique.

(x_2, x_3, x_4) is a maximal clique.

$$p(x_1, \ldots, x_4) = \frac{1}{Z}\phi_1(x_1, x_2, x_3)\phi_2(x_2, x_3, x_4)\,.$$

MN Examples: Potts Model

Let $x_1, \ldots, x_n \in \{1, \ldots, C\}$,

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{i \sim j} \phi_{ij}(x_i, x_j)\,,$$

where

$$\log \phi_{ij}(x_i, x_j) = \begin{cases} \beta > 0 & \text{if} \quad x_i = x_j \\ 0 & \text{otherwise} \end{cases},$$

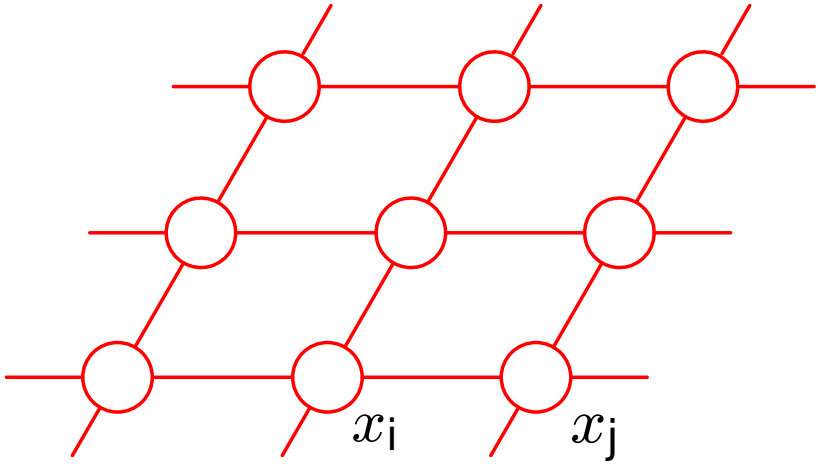


Figure source [Bishop 2006].

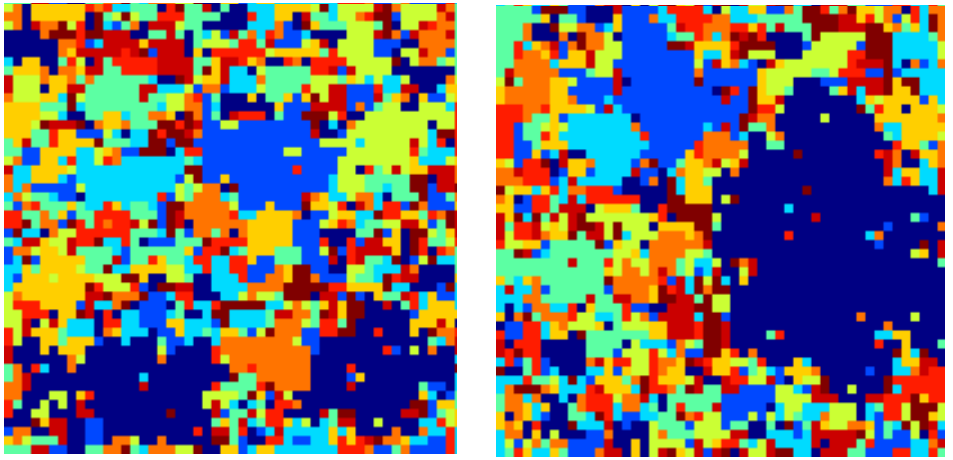


Figure source Erik Sudderth.

From Directed Graphs to Undirected Graphs: Moralization

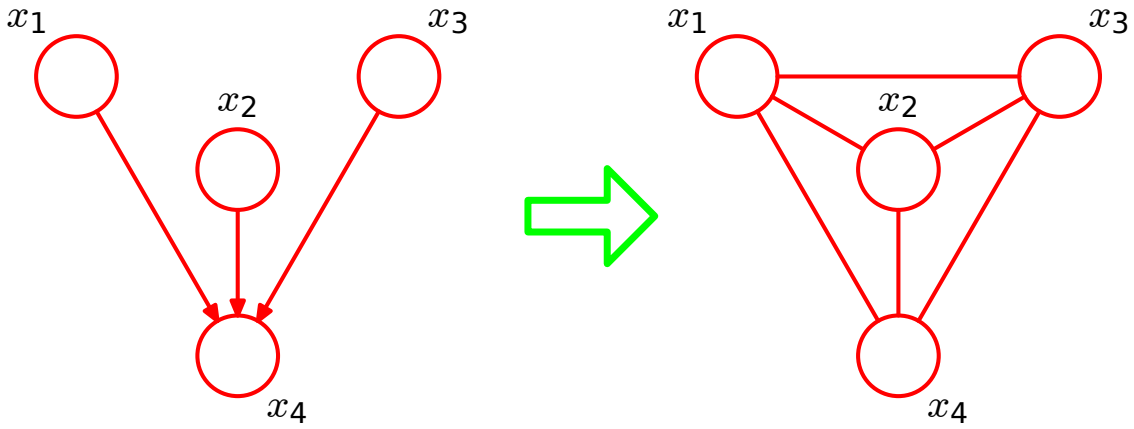
Let $p(x_1, \ldots, x_4) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)$.
How do we obtain the corresponding undirected model?

$p(x_4|x_1, x_2, x_3)$ implies that $x_1, \ldots, x_4$ must be in a maximal clique.

General method:

1- Fully connect all the parents of any node.

2- Eliminate edge directions.



Moralization adds the fewest extra links and so retains the maximum number of CIs.

Figure source [Bishop 2006].

CIs in Directed and Undirected Models

If all the CIs of $p(x_1, \ldots, x_n)$ are reflected in $\mathcal{G}$, and vice versa, then $\mathcal{G}$ is said to be a perfect map.

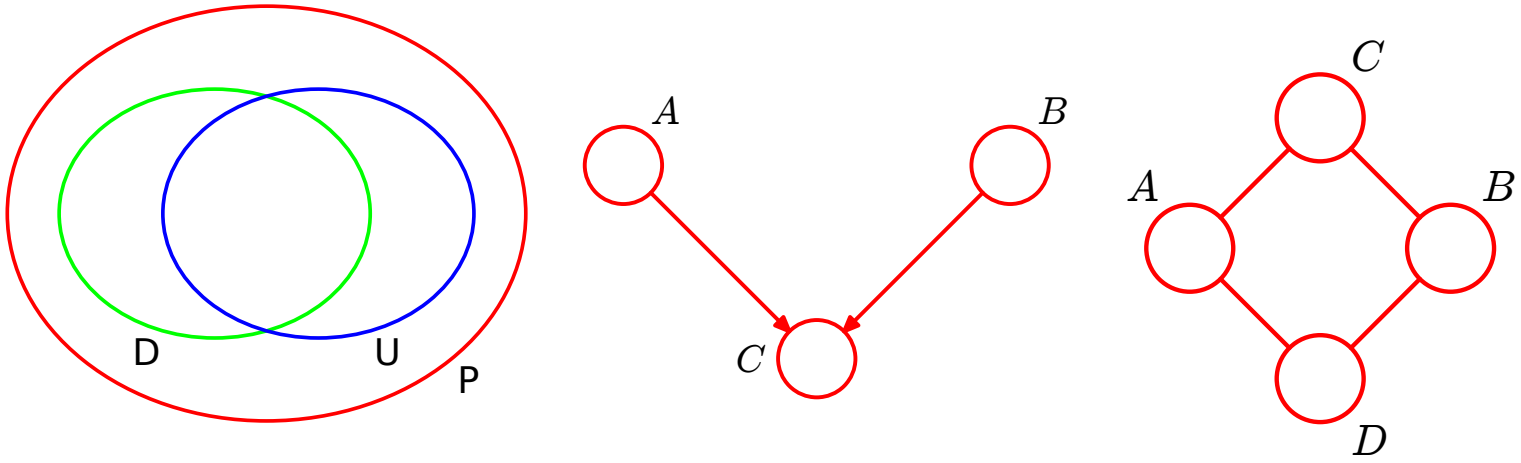


Figure source [Bishop 2006].

Summary

With ML computers learn patterns and then use them to make predictions.
With ML we avoid to manually encode patterns in computer programs.

Model-based ML separates knowledge about the data generation process (model) from reasoning and prediction (inference algorithm).

The Bayesian framework allows us to do model-based ML using probability distributions which must be structured for tractability.

Probabilistic graphical models encode such structured distributions by specifying several CIs (factorizations) that they must satisfy.

Bayesian Networks and Markov Networks are two different types of graphical models which can express different types of CIs.

References

Detailed references:

- Hinton, G. E., Osindero, S. and Teh, Y. A fast learning algorithm for deep belief nets. Neural Computation 18, 2006, 1527-1554.
- Koller, D. and Friedman, N. Probabilistic Graphical Models: Principles and Techniques Mit Press, 2009
- Bishop, C. M. Model-based machine learning Philosophical Transactions of the Royal Society A: Mathematical,Physical and Engineering Sciences, 2013, 371
- Ghahramani Z. Bayesian nonparametrics and the probabilistic approach to modelling. Philosophical Transactions of the Royal Society A, 2012.
- Murphy, K. Machine Learning: A Probabilistic Perspective Mit Press, 2012

Inference in Graphical Models with Discrete Variables

Daniel Hernández–Lobato[1],

June 29, 2015

slides by

José Miguel Hernández-Lobato and Daniel Hernández-Lobato.

[1]Universidad Autónoma de Madrid.

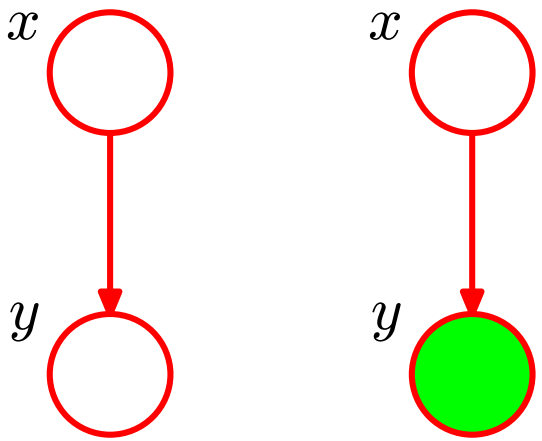
What is inference?

The problem of inference: Given that some of the variables in a GM are clamped to observed values, we want to compute the conditional distribution of a subset of other variables.

Simplest example: a Bayesian network (BN) for $p(x, y)$.

The BN represents the joint distribution for x and y as $p(x, y) = p(y|x)p(x)$.

Given y, we use Bayes' theorem to obtain $p(x|y) = p(y, x)/p(y)$, where $p(y) = \sum_x p(y, x)$.



Difficult problem with probably exponential cost in the worst case.

However, for some GM exact inference is tractable.

Variable Elimination

Given the BN $A \to B \to C \to D$ we want to compute $p(d)$.

The chain rule of BNs and the sum rule of probability theory lead to
$p(d) = \sum_a \sum_b \sum_c p(d|c)p(c|b)p(b|a)p(a)$.

This operation has cost $\mathcal{O}(n^4)$, when the variables can take n values each.
However, we can reorder the computations to obtain
$p(d) = \sum_c p(d|c) \sum_b p(c|b) \sum_a p(b|a)p(a)$.

In this case, the computation of $p(d)$ has cost $\mathcal{O}(n^2)$.
Selecting a specific order of computations can produce large savings.

Main Idea

The GM expresses the joint distribution as a product of factors which depend only on a small number of variables.

Only need to compute some expressions once. By caching intermediate results, we avoid generating very large factors (probability tables).

Sum-product Variable Elimination Algorithm

Procedure Sum-Product-VE

Input

Set of factors $\mathbf{\Phi}$.

Set of variables to be eliminated $\mathbf{Z}$.

Ordering of $\mathbf{Z}$: $Z_1, \ldots, Z_k$.

1: **for** $i = 1, \ldots, k$

2: $\quad \mathbf{\Phi} \leftarrow$ Eliminate-Var$(\mathbf{\Phi}, Z_i)$

3: **return** $\prod_{\phi \in \mathbf{\Phi}} \phi$

Procedure Eliminate-Var

Input

Set of factors $\mathbf{\Phi}$.

Variable to be summed out Z.

1: $\mathbf{\Phi}' \leftarrow \{\phi \in \mathbf{\Phi} : Z \in \text{Scope}[\phi]\}$

2: $\mathbf{\Phi}'' \leftarrow \mathbf{\Phi} - \mathbf{\Phi}'$

3: $\psi \leftarrow \prod_{\phi \in \mathbf{\Phi}'} \phi$

4: $\tau \leftarrow \sum_Z \psi$

5: **return** $\mathbf{\Phi}'' \cup \{\tau\}$

where scope(ϕ) is the set of variables on which ϕ is evaluated.

When a set of variables $\mathbf{Z}'$ are clamped to observed values, we

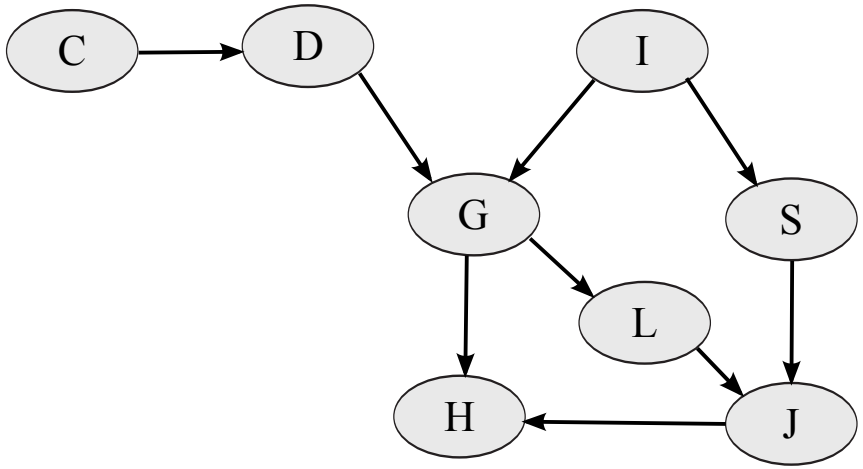
1 - Reduce the factors in $\mathbf{\Phi}$ which depend on any variable in $\mathbf{Z}'$.

2 - Eliminate only the variables in $\mathbf{Z} - \mathbf{Z}'$.

Example of the Execution of Variable Elimination

We ask for $p(J)$ for the joint distribution $p(J, C, D, I, H, G, S, L)$ given by the BN in the figure.

Slide source [Koller et al. 2009]



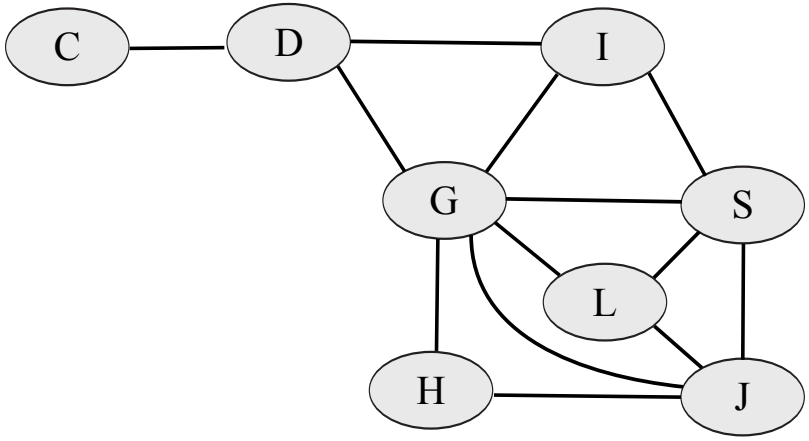
Step	Variable Eliminated	Factors used to compute ψ_i	Variables Involved	New Factor
1	C	$\phi_C(C)$, $\phi_D(D, C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G, I, D)$, $\tau_1(D)$	G I, D	$\tau_2(G, I)$
3	I	$\phi_I(I)$, $\phi_S(S, I)$, $\tau_2(G, I)$	G, S, I	$\tau_3(G, S)$
4	H	$\phi_H(H, G, J)$	H G, J	$\tau_4(G, J)$
5	G	$\tau_4(G, J)$, $\tau_3(G, S)$, $\phi_L(L, G)$	G, J, L, S	$\tau_5(J, L, S)$
6	S	$\tau_5(J, L, S)$, $\phi_J(J, L, S)$	J, L, S	$\tau_6(J, L)$
7	L	$\tau_6(J, L)$	J, L	$\tau_7(J)$

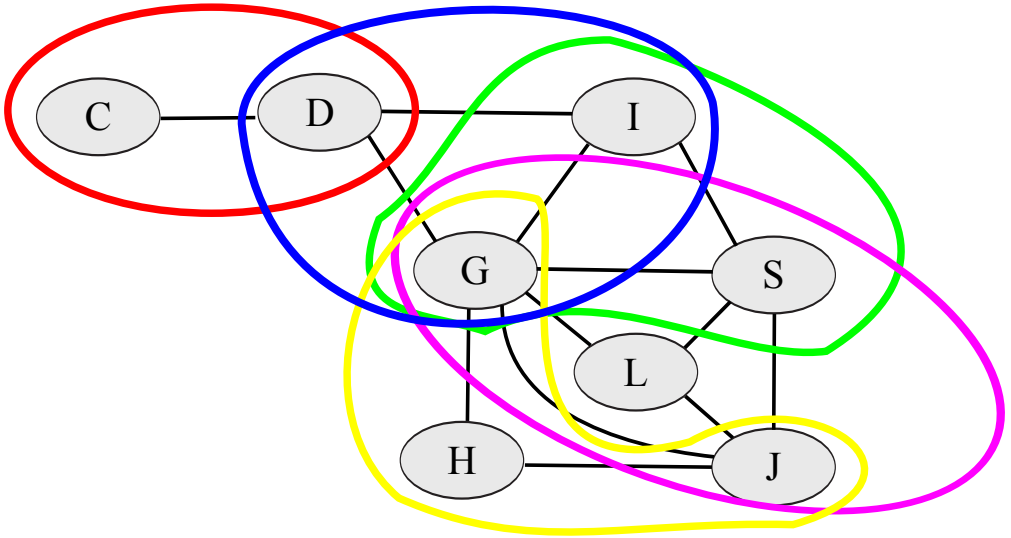
Induced Graph (IG)

Undirected graph where two random variables are connected if the two of them appear in some factor during the execution of variable elimination.

The scope of every factor produced during VE is a clique in the IG.

Every maximal clique is the scope of one of the ψ produced during VE.





The cost of VE is exponential in the number of nodes in the largest clique of the IG. Depends on the elimination ordering ! Finding the best ordering is NP-hard . Alternative: use a heuristic method .

Clique Tree

A run of VE defines a clique tree such that:

1 - We have a node for each factor ψ_i produced by VE.

2 - Each node is associated with the set of variables $\mathbf{C}_i = \text{Scope}[\psi_i]$.

3 - We connect $\mathbf{C}_i$ and $\mathbf{C}_j$ if τ_i is used to compute τ_j.

4 - For the link between $\mathbf{C}_i$ and $\mathbf{C}_j$ we have the sepset $\mathbf{S}_{ij} = \mathbf{C}_i \cap \mathbf{C}_j$.

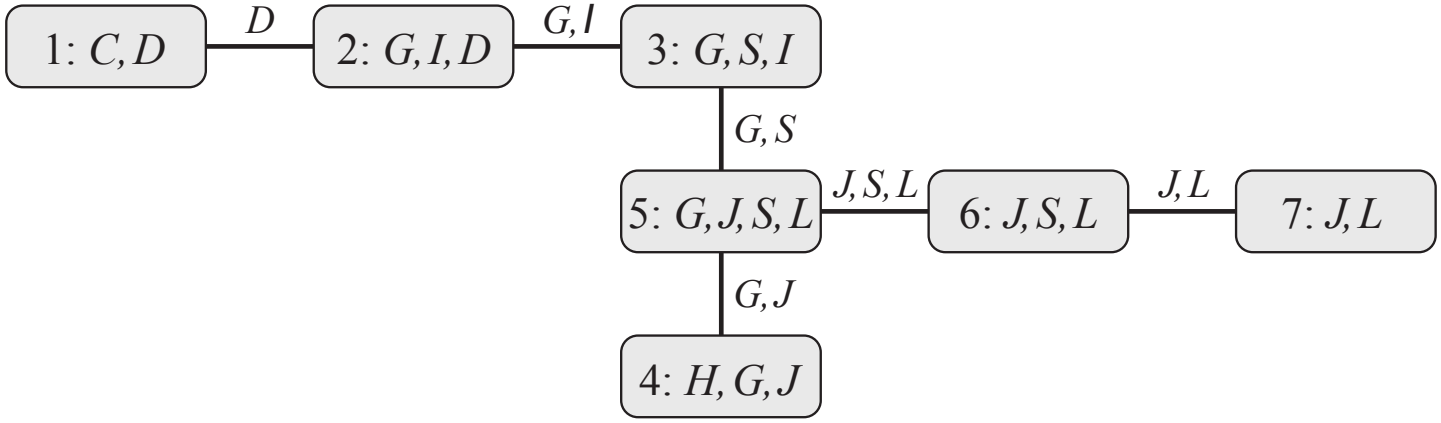


Figure [Koller et al. 2009].

The clique tree satisfies the following properties:

Running intersection property (RIP) : for any variable X such that $X \in \mathbf{C}_i$ and $X \in \mathbf{C}_j$, then X is also in every node in the path between $\mathbf{C}_i$ and $\mathbf{C}_j$.

Family preservation property (FPP) : each factor $\phi \in \mathbf{\Phi}$ is associated with a node $\mathbf{C}_i$ such that $\text{Scope}[\phi] \subseteq \mathbf{C}_i$.

Clique Tree Message Passing

VE can be implemented via message passing in a given clique tree.

By the FPP, any $\phi \in \mathbf{\Phi}$ is assigned to a node which we denote by $\alpha(\phi)$.

The initial potential for $\mathbf{C}_j$ is $\psi_j(\mathbf{C}_j) = \prod_{\phi:\alpha(\phi)=j} \phi\,.$

The message from $\mathbf{C}_i$ to its neighbor $\mathbf{C}_j$ is $\delta_{i\to j} = \sum_{\mathbf{C}_i-\mathbf{S}_{ij}} \psi_i \prod_{k\in \mathrm{Nb}(i)-\{j\}} \delta_{k\to i}\,.$

At any node i, we call *beliefs* the factor given by

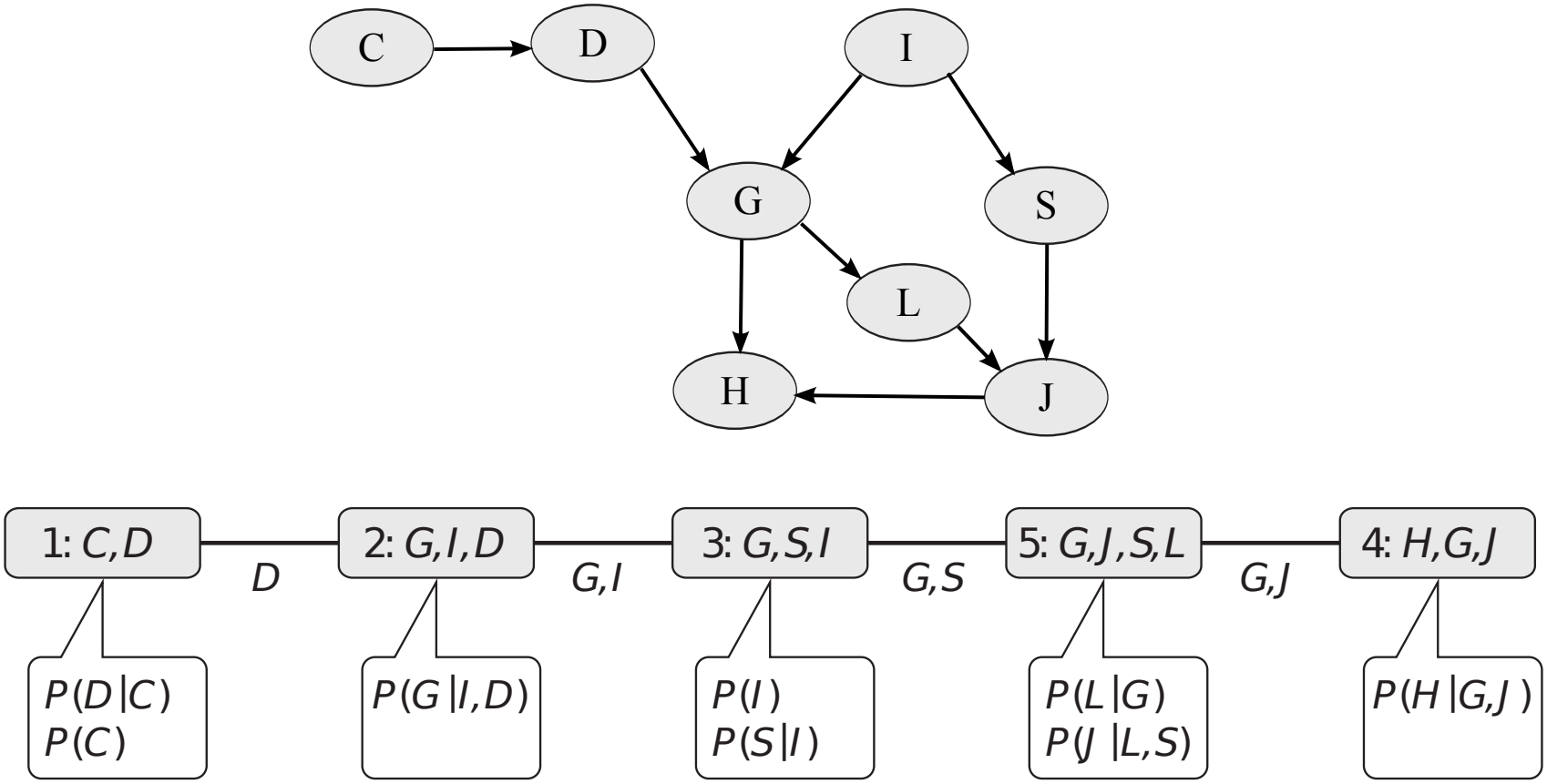
$$\beta_i(\mathbf{C}_i) = \psi_i(\mathbf{C}_i) \prod_{j\in \mathrm{Nb}(i)} \delta_{j\to i} = \sum_{\cup_j \mathbf{C}_j - \mathbf{C}_i} \prod_{\phi\in\mathbf{\Phi}} \phi\,.$$

To compute $p(Z_j)$ we follow the steps

- Select any $\mathbf{C}_i$ s.t. $Z_j \in \mathbf{C}_i$ and compute $\beta_i(\mathbf{C}_i)$ by message passing.
- Sum out in $\beta_i(\mathbf{C}_i)$ the variables $\mathbf{C}_i - \{Z_j\}$.

Example of Message Passing I

Given a GM we obtain a clique tree that satisfies the RIP and the FPP.



To obtain $p(J)$ we select $\mathbf{C}_5$ and do message passing to get $\beta_5(G, J, S, L)$.

Figure source [Koller et al. 2009].

Example of Message Passing II

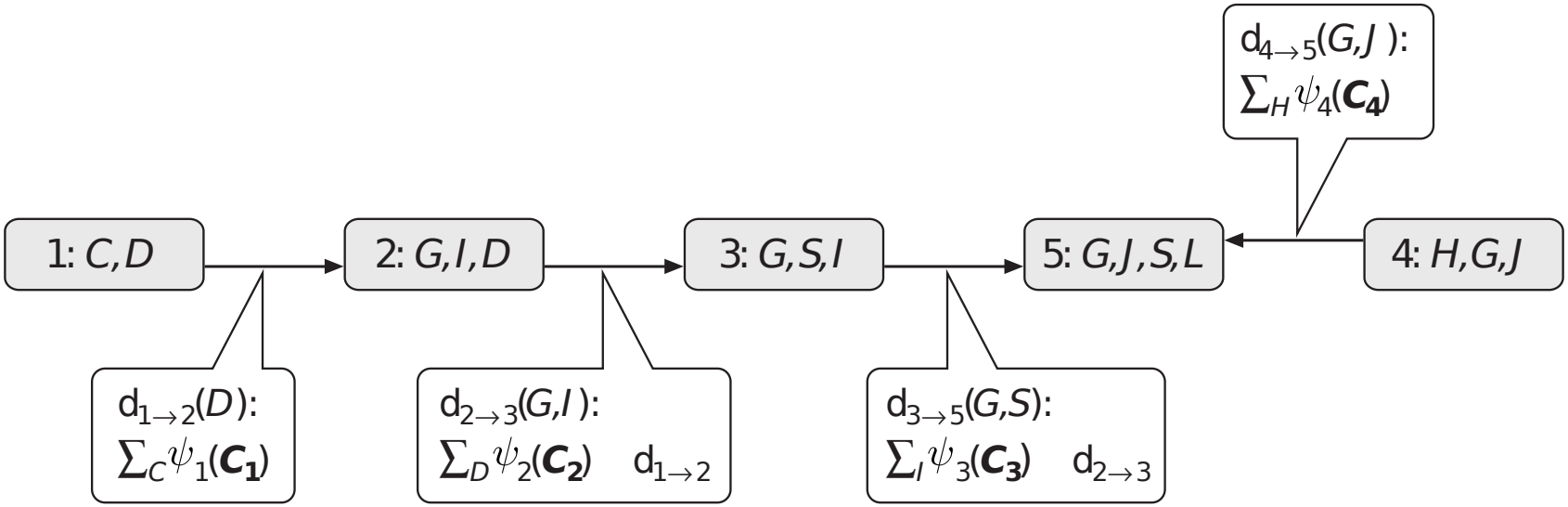


Figure source [Koller et al. 2009].

Calibration of a Clique Tree

Any edge $i \sim j$ in a CT has two messages associated $\delta_{i \to j}$ and $\delta_{j \to i}$.

To compute both messages for any edge and the beliefs for each node:

1 - Pick a random node as the root.
2 - Send all messages from the leaves to the root.
3 - Send all messages from the root to the leaves.
4 - Compute the beliefs for each node in the graph using the messages.

At the end, each node has the marginal over the variables in its scope.

We can compute all the marginals with only twice the cost of VE.

How to Construct a Clique Tree

By running Variable Elimination:

Any resulting $\mathbf{C}_i$ which is not a maximal clique in the IG is usually collapsed.

From a chordal graph (undirected graph with no loop larger than 3):

1 - Obtain an undirected graph by **moralization**.
2 - Obtain a chordal graph by **triangulation**.
3 - Find the **maximal cliques** in the chordal graph.
4 - Make each maximal clique a node in the clique tree.
5 - Find **maximum spanning tree** with weight $|\mathbf{C}_i \cap \mathbf{C}_j|$ for $i \sim j$.

The computational cost of message passing is exponential in the number of variables of the largest clique, exponential in the tree-width.

Finding a triangulation in which the largest clique has minimum size is NP-hard. Alternative: use a heuristic method.

Approximate Inference

Exact inference is intractable if the tree-width of the clique tree is large.

What to do then?

Use approximate inference, trading off computational cost vs. accuracy.

We construct an approximation $\mathcal{Q}$ to the target distribution $\mathcal{P}$.

The approximation $\mathcal{Q}$ can be obtained by

1. Selecting a **simpler form** for $\mathcal{Q}$ that can be efficiently tuned to $\mathcal{P}$.
2. Drawing a **finite number of samples** from the distribution $\mathcal{P}$.
   $\mathcal{Q}$ is then built using these samples.

In the first case, approximate inference involves optimizing $\mathcal{Q}$ to match $\mathcal{P}$.

Some of these methods can be viewed as message passing on a graph.

Loopy Belief Propagation

We do message passing on a cluster graph rather than on a clique tree.

Cluster graph:

- Like a clique tree, but with cycles or loops.
- For the link between $\mathbf{C}_i$ and $\mathbf{C}_j$ we have the sepset $\mathbf{S}_{ij} \subseteq \mathbf{C}_i \cap \mathbf{C}_j$.

The cluster graph has to satisfy the **FPP** and a **generalized RIP**.

Running Intersection Property for Cluster Graphs

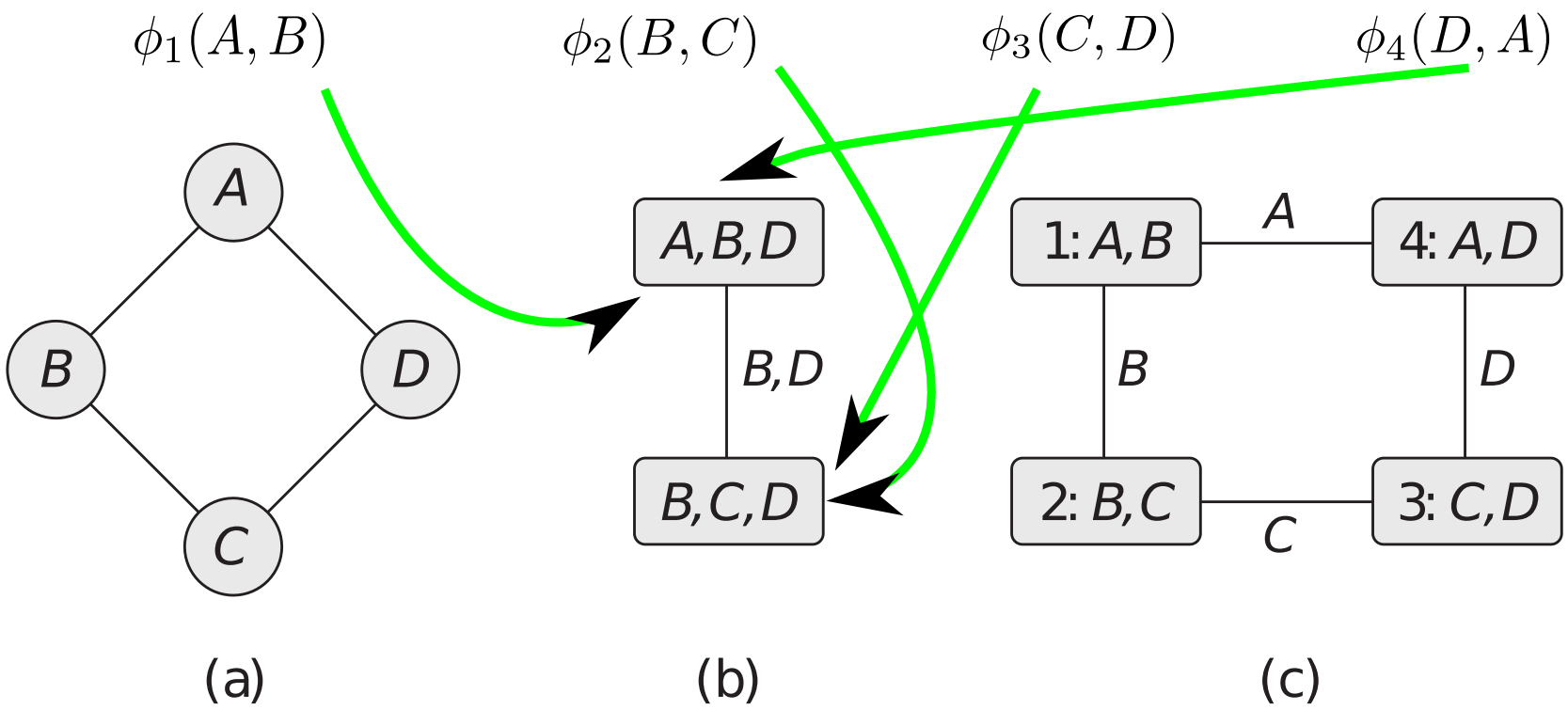
For any two nodes $\mathbf{C}_i$ and $\mathbf{C}_j$ containing variable X, there is precisely one path between them for which $X \in \mathbf{S}_e$ for all edges e in the path.

This implies that all edges associated with X form a tree that spans all the nodes containing X. Two nodes can be connected by multiple paths.

We can do inference on the cluster graph by message passing. However, because of the loops, we will often obtain only approximate answers.

Cluster Graph Example

Undirected graphical model with factors:



(a) - The undirected GM. (b) - A clique tree for the network in (a).
(c) - A cluster graph for the same network.

Figure source [Koller et al. 2009]

How to Choose the Cluster Graph?

When choosing the CG we have to consider a cost vs. accuracy trade-off.

Doing message passing in CGs that lead to more accurate results is more computationally expensive and vice-versa.

The chosen CG must also to satisfy the RIP and FPP.

A typical solution is the **Bethe cluster graph** (easily automated).

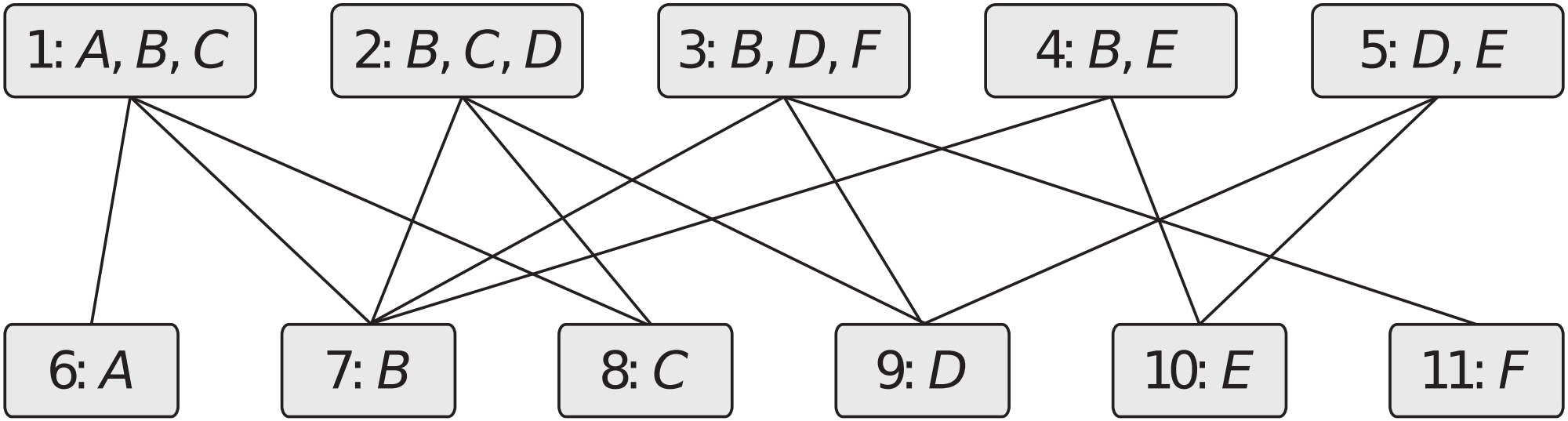
- A bipartite graph with two layers of large and small nodes.
- A large node $\mathbf{C}_{\alpha(\phi)}$ per factor ϕ in $\mathbf{\Phi}$, where $\mathbf{C}_{\alpha(\phi)} = \text{Scope}(\phi)$.
- A small node per random variable, with no associated factor.
- A large node $\mathbf{C}_i$ is connected to a small node $\mathbf{C}_j$ when $\mathbf{C}_j \subseteq \mathbf{C}_i$.

The Bethe cluster graph is guaranteed to satisfy both the RIP and FPP.

Example of Bethe Cluster Graph

Distribution with factors

$$\phi_1(A, B, C) \quad \phi_2(B, C, D) \quad \phi_3(B, D, F) \quad \phi_4(B, E) \quad \phi_5(D, E)$$



The Bethe cluster graph is limited in the sense that information between different clusters is passed through univariate marginal distributions.

Merging clusters can help capture interactions between multiple variables.

Loopy Belief Propagation Algorithm

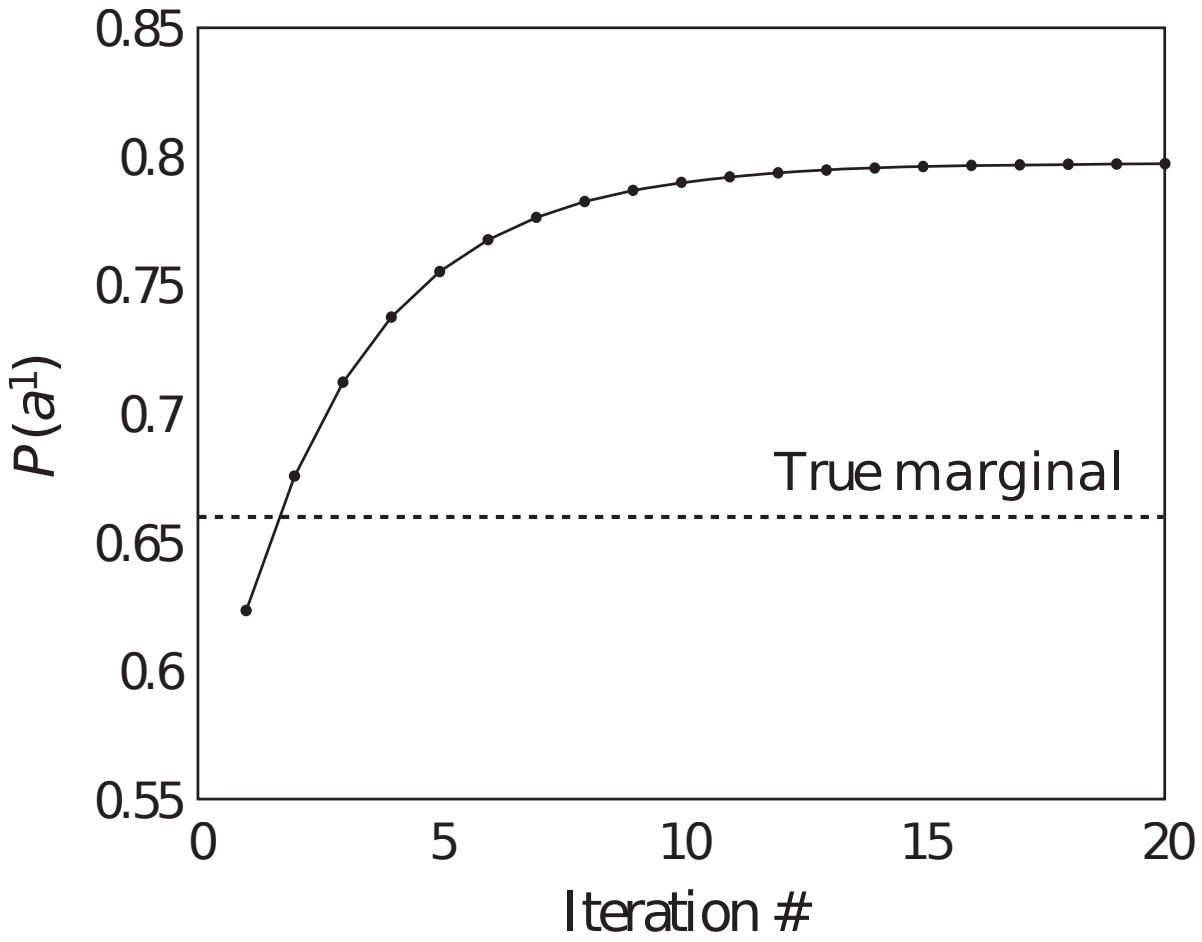
1. Assign each factor $\phi_k \in \Phi$ to a cluster $\mathbf{C}_{\alpha(k)}$.
2. Construct initial potentials $\psi_i(\mathbf{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$.
3. Initialize all messages to be non-informative (*e.g.* equal to $\mathbf{1}$).
4. Repeat until convergence (*e.g.* messages no longer change)
   1. Select edge (i, j) and pass message:

$$\delta_{i\to j}(\mathbf{S}_{i,j}) = \sum_{C_i - S_{i,j}} \psi_i \times \prod_{k\in(\mathcal{N}_i - j)} \delta_{k\to i} = \left(\sum_{C_i - S_{i,j}} \psi_i \times \prod_{k\in\mathcal{N}_i} \delta_{k\to i} \right) / \delta_{j\to i} \,.$$

5. Compute un-normalized beliefs $\beta_i(\mathbf{C}_i) = \psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$.

At convergence, the marginals over the sepsets of adjacent nodes coincide and we have a calibrated cluster graph.

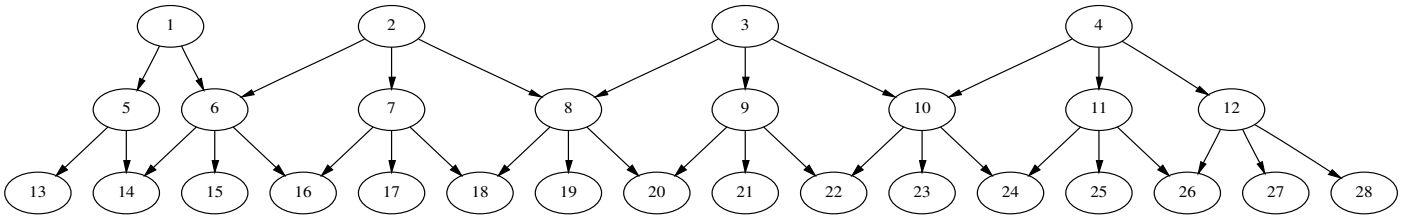
Example of Loopy Belief Propagation



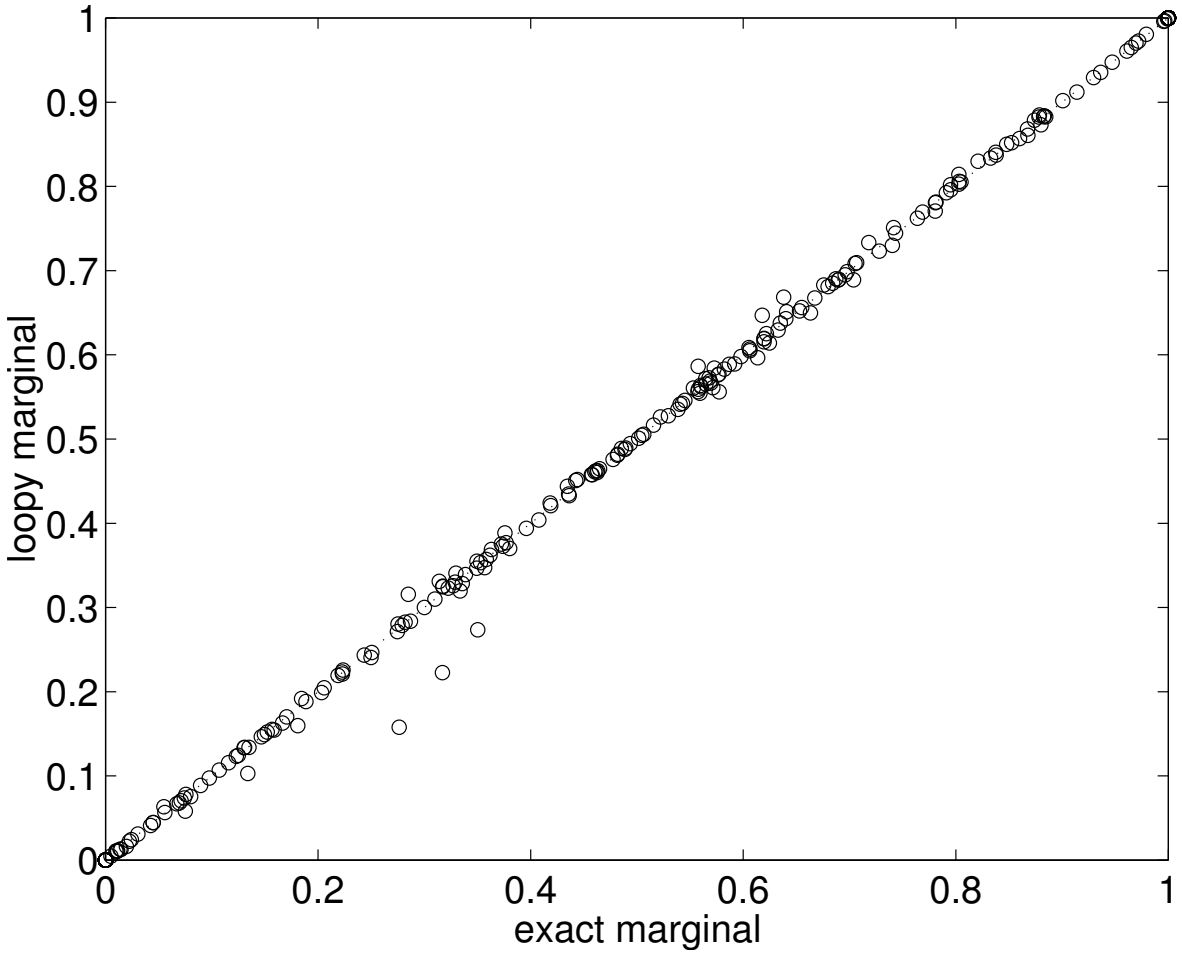
The algorithm converges but to a solution different from the exact one.

Figure source [Koller et al. 2009].

Loopy Belief Propagation: Accuracy of the Approximation



K. P. Murphy, Y. Weiss and M. I. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study, UAI 99



Summary

Inference in GMs is mainly the computation of conditional distributions.

Exact inference requires to sum an exponentially large joint distribution.

Variable elimination avoids the exponential blow up by caching intermediate results.

A clique tree allows us to do exact inference using message passing.
Message passing produces multiple answers in a single run.

When the tree-width of the graph is large we have to use approximations.

Loopy belief propagation allows us to do efficient approximate inference by passing messages in a cluster graph.

The Laplace Approximation and Variational Inference

Daniel Hernández–Lobato[1],

June 29, 2015

slides by

Daniel Hernández-Lobato and José Miguel Hernández-Lobato.

[1]Universidad Autónoma de Madrid.

The Laplace Approximation: Introduction

The **simplest deterministic method** for approximate inference. Restricted to GMs in which the variables of interest are continuous.

The factors for the continuous random variables will generally be some **continuous parametric functions**.

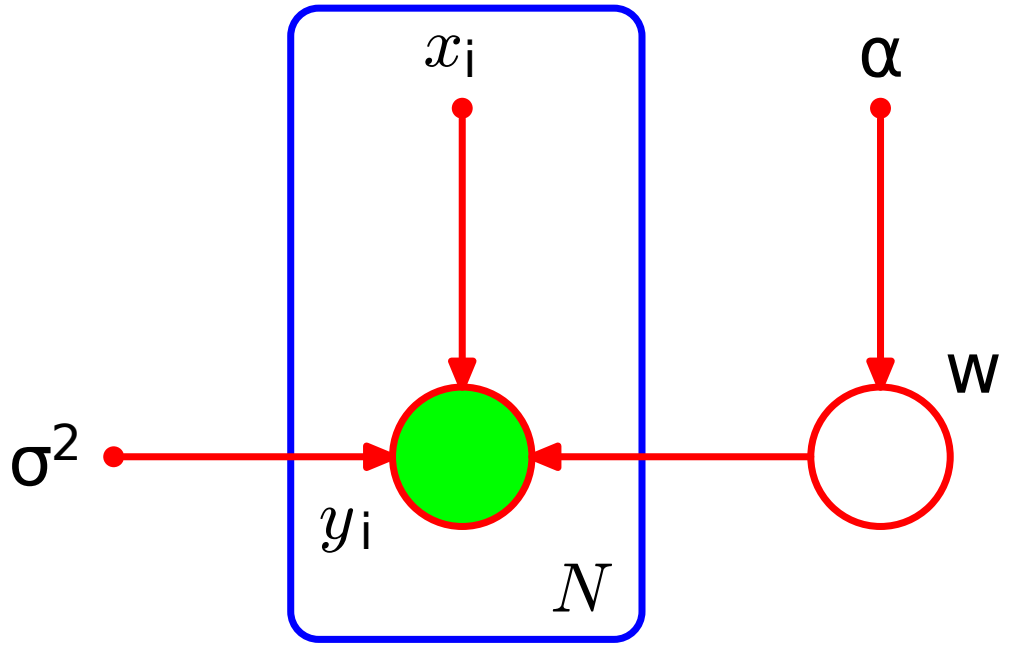
Probit regression model:

$$y_i = \begin{cases} 1 & \text{if} \quad \mathbf{w}^{\mathsf{T}}\mathbf{x}_i + \epsilon_i \geq 0 \\ -1 & \text{if} \quad \mathbf{w}^{\mathsf{T}}\mathbf{x}_i + \epsilon_i < 0 \end{cases}$$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\alpha)$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

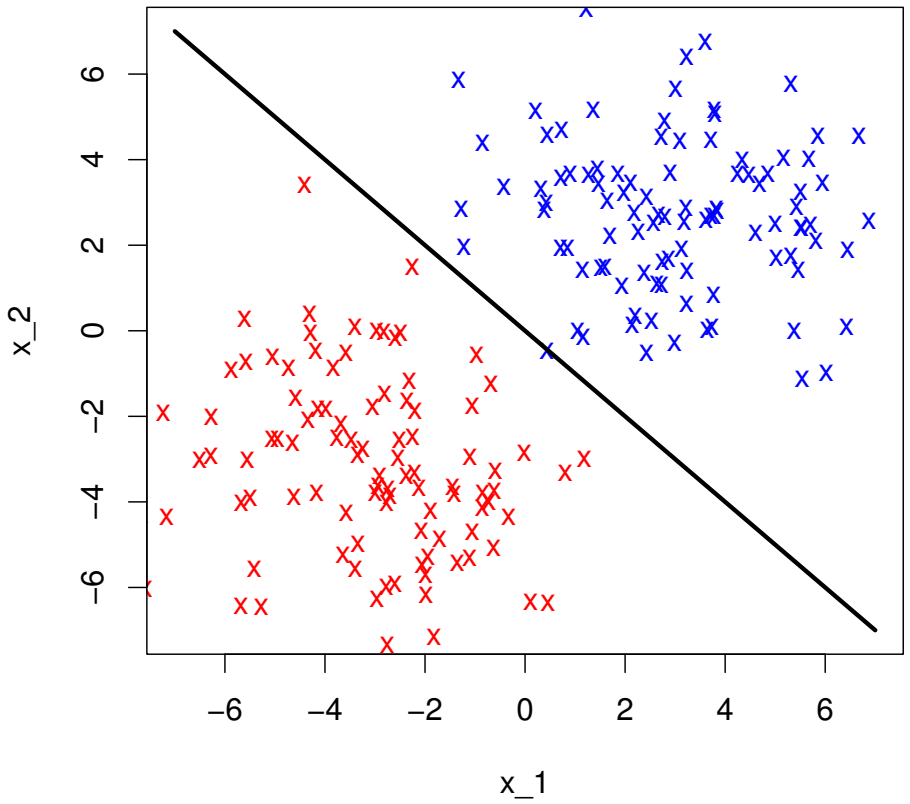
$$p(y_i|\mathbf{w}_i, \mathbf{x}_i) = \Phi(y_i \mathbf{w}^{\mathsf{T}}\mathbf{x}_i|0, \sigma^2)\,.$$



$\mathbf{w}$ are the variables of interest, and the evidence are the $\{y_i\}_{i=1}^N$. Furthermore,

$$p(\mathbf{y}, \mathbf{w}) = \prod_{i=1}^N p(y_i|\mathbf{w}, \mathbf{x}_i)p(\mathbf{w}) = \prod_{i=1}^N \Phi(y_i \mathbf{w}^{\mathsf{T}}\mathbf{x}_i|0, \sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I}\alpha)\,.$$

The Laplace Approximation: Probit Regression Model



Sample data from the corresponding GM.

We want to make inference on $\mathbf{w}$ given some observed labels $\mathbf{y}$. For this, we can use Bayes theorem:

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}\,.$$

where $p(\mathbf{y})$ is a normalization constant which can be used for model selection.

We also want to compute a predictive distribution for new unlabeled instances:

$$p(y_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{y}) = \int p(y_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{w})p(\mathbf{w}|\mathbf{y})d\mathbf{w}\,.$$

Unfortunately the required computations are intractable.

The Laplace Approximation: Univariate Case I

The Laplace approximation will find a Gaussian approximation to the conditional distribution of a set of continuous variables:

Consider first a single scalar variable z:

$$p(z) = \frac{1}{Z} f(z)\,,$$

where $f(z) = p(z, \mathbf{e})$, $\mathbf{e}$ are observed variables and $Z = \int f(z)dz$.

How do we set the parameters of $\mathcal{Q}(z)$, the Gaussian approximation, so that it is similar to $p(z)$ given that we **do not know** Z?

The first step is to find a mode (*i.e.*, a local maximum z_0) of $p(z)$

$$\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0$$

Any optimization algorithm can be used for this purpose.

The Laplace Approximation: Univariate Case II

The logarithm of a Gaussian is a quadratic function of the variables.

We consider a truncated Taylor expansion of $\log f(z)$ center at the mode:

$$\log f(z) \approx \log f(z_0) - \frac{1}{2} A (z - z_0)^2 , \qquad A = - \frac{d^2}{dz^2} \log f(z) \Big|_{z=z_0}$$

Taking the exponential we obtain:

$$f(z) \approx f(z_0) \exp \left\{ - \frac{A}{2} (z - z_0)^2 \right\} \qquad \mathcal{Q}(z) = \mathcal{N}(z | z_0, A^{-1})$$

The normalization constant Z can be approximated by $f(z_0)\sqrt{2\pi/A}$.

The **mean** of $\mathcal{Q}$ is z_0 and the **variance** is A^{-1}.

The Gaussian approximation will only be defined if $A > 0$, *i.e.*, z_0 must be a local maximum of $\log f$ and hence f with negative second derivative.

The Laplace Approximation: Multi-variate Case

The same principle can be applied to approximate a M-dimensional distribution $p(\mathbf{z}) = f(\mathbf{z})/Z$ defined over a vector of real values.

$$\log f(\mathbf{z}_0) \approx \log f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathsf{T}} \mathbf{A} (\mathbf{z} - \mathbf{z}_0)\,, \qquad \mathbf{A} = - \bigtriangledown \bigtriangledown \log f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

Taking the exponential we have:

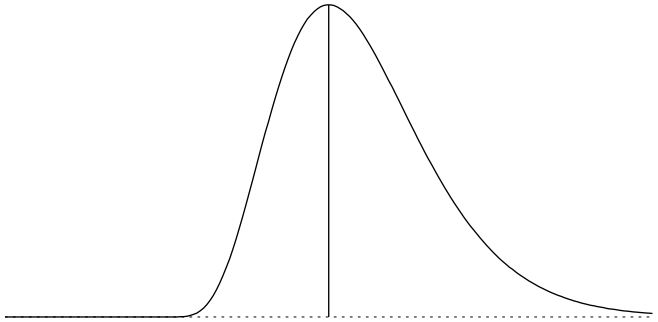
$$f(\mathbf{z}) \approx f(\mathbf{z}_0) \exp\left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathsf{T}} \mathbf{A} (\mathbf{z} - \mathbf{z}_0) \right\}\,, \qquad \mathcal{Q}(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1})$$

The normalization constant is approximated by $f(z_0)\sqrt{(2\pi)^M/|\mathbf{A}|}$.

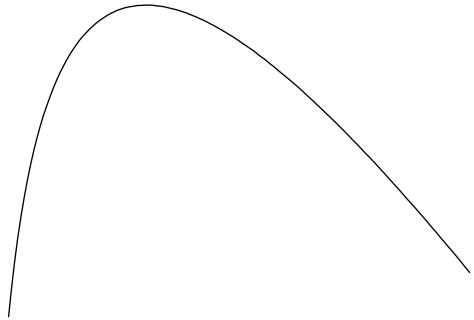
The **mean** of $\mathcal{Q}$ is $\mathbf{z}_0$ and the **covariance matrix** is $\mathbf{A}^{-1}$.

The Gaussian approximation will only be defined if $\mathbf{A}$ is positive semidefinite, *i.e.*, z_0 must be a local maximum not a minimum or a saddle point.

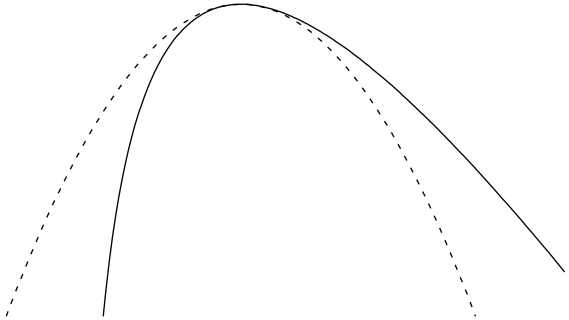
The Laplace Approximation: Example



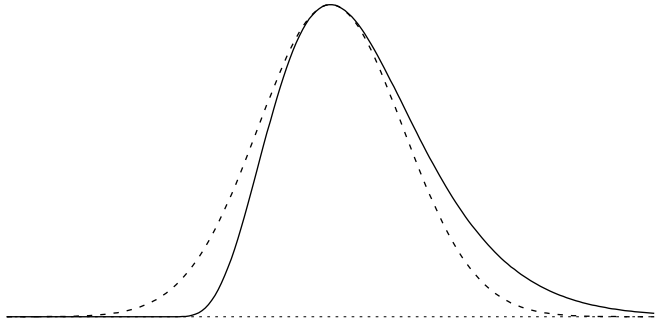
f(z)



log f(z)



log f(z) & log ~Q(z)



f(z) & ~Q(z)

The Laplace Approximation: Probit Regression I

For simplicity we consider that $\sigma^2 = 1$ and that $\alpha = 1$.

The posterior distribution is:

$$p(\mathbf{w}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) = f(\mathbf{w})\,, \qquad \log f(\mathbf{w}) = \log p(\mathbf{y}|\mathbf{w}) + \log p(\mathbf{w})\,.$$

Furthermore, we have that:

$$\log f(\mathbf{w}) = \sum_{i=1}^{N} \log \Phi(y_i \mathbf{w}^{\mathsf{T}} \mathbf{x}_i) - \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}\,.$$

Let $\mathbf{w}_0$ a maximum of f. Computing the negative Hessian at $\mathbf{w}_0$:

$$\mathbf{A} = -\nabla\nabla \log f(\mathbf{w}_0) = \sum_{i=1}^{N} \left[v_i(s_i + v_i)\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}\right] + \mathbf{I}\,, \quad v_i = \frac{\mathcal{N}(s_i|0,1)}{\Phi(s_i)}\,, \quad s_i = y_i \mathbf{w}_0^{\mathsf{T}} \mathbf{x}_i\,.$$

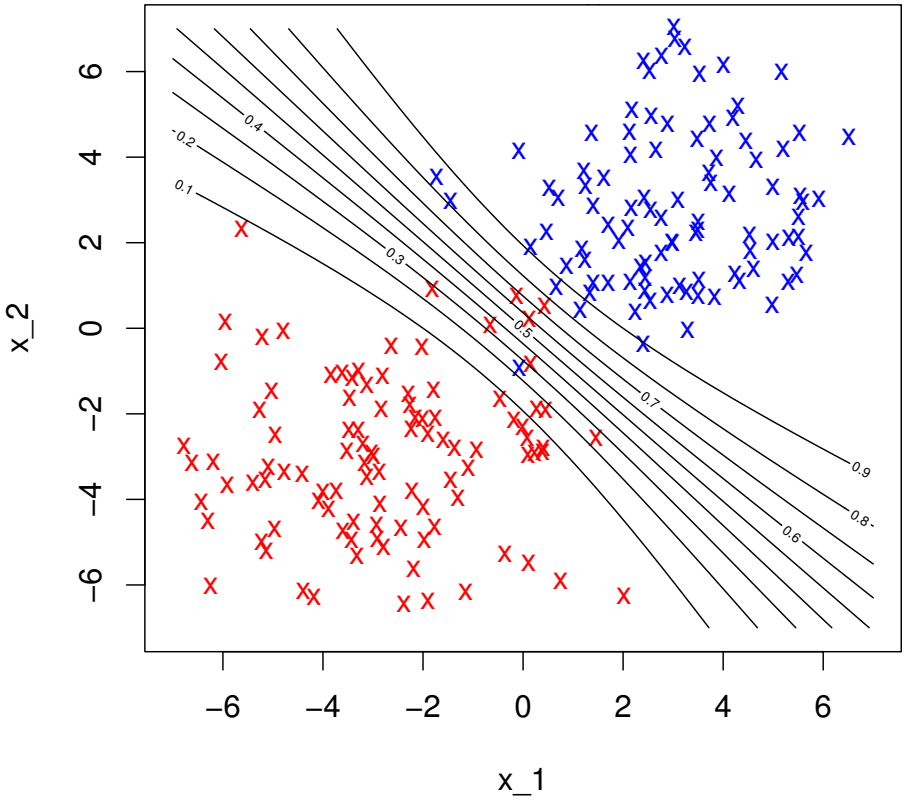
The approximate posterior is $\mathcal{Q}(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{A}^{-1})$. The normalization constant is $Z \approx f(\mathbf{w}_0)\sqrt{\frac{(2\pi)^M}{|\mathbf{A}|}}$, where M is the dimensionality of $\mathbf{w}$.

The Laplace Approximation: Probit Regression II

It is also possible to compute an approximate predictive distribution:

$$p(y_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{y}) \approx \int p(y_{\text{new}}|\mathbf{x}_{\text{new}}, \mathbf{w})\mathcal{Q}(\mathbf{w})d\mathbf{w} = \int \Phi(y_{\text{new}}\mathbf{w}^{\mathsf{T}}\mathbf{x}_{\text{new}})\mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{A}^{-1})d\mathbf{w}\,,$$

$$= \Phi\left(\frac{y_{\text{new}}\mathbf{w}_0^{\mathsf{T}}\mathbf{x}_{\text{new}}}{\sqrt{\mathbf{x}_{\text{new}}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{x}_{\text{new}} + 1}}\right).$$



Uncertainty is high near the decision boundary and progressively decreases as we move away from it.

Uncertainty is significantly larger in regions where there is no data.

MAP solutions do not consider this uncertainty.

The Laplace Approximation: Considerations

- The mode of $\log f$ can be found using a numerical optimization method. The Hessian can be approximated by differences.
- Many distributions can be multi-modal, what leads to many different Laplace approximations, depending on the mode.
- In many cases, the posterior distribution of $\mathbf{z}$ will converge to a Gaussian as the number of observations (evidence) increases.
- Only applicable on real variables.
- Only focuses around the mode and can fail to capture global properties.
- No need to know Z. Furthermore, it provides an estimate of Z.
- Depends on the basis. If z is mapped to $u(z)$, the density is transformed to $p(u) = p(z)|dz/du|$ and the approximation will be different.

Variational Inference: Introduction

Based on the calculus of variations, *i.e.*, a generalization of standard calculus. Deals with functionals, functions and derivatives of functionals rather than functions, variables and derivatives. Similar rules apply.

Can be applied to models of either continuous or discrete random variables.

Approximates both the posterior distribution and its normalization constant: $p(\mathbf{z}|\mathbf{e})$ and $p(\mathbf{e})$.

It is based on the following **decomposition**:

$$\log p(\mathbf{e}) = \mathcal{L}(\mathcal{Q}) + \mathrm{KL}(\mathcal{Q}||p)$$

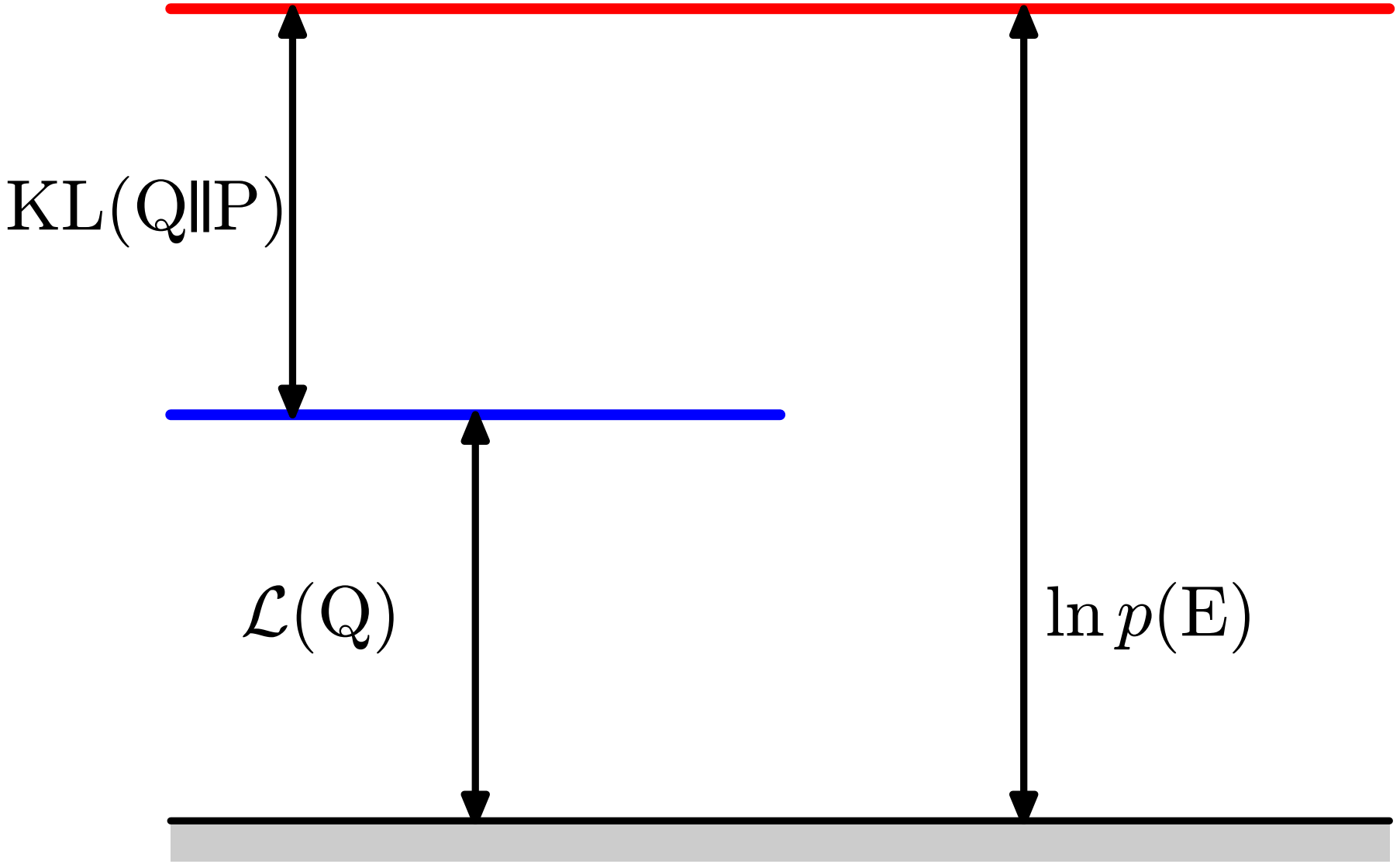
where

$$\mathcal{L}(\mathcal{Q}) = \sum_{\mathbf{z}} \mathcal{Q}(\mathbf{z}) \log \left\{ \frac{p(\mathbf{z}, \mathbf{e})}{\mathcal{Q}(\mathbf{z})} \right\}, \quad \mathrm{KL}(\mathcal{Q}||p) = -\sum_{\mathbf{z}} \mathcal{Q}(\mathbf{z}) \log \left\{ \frac{p(\mathbf{z}|\mathbf{e})}{\mathcal{Q}(\mathbf{z})} \right\} \geq 0$$

The Kullback Leibler divergence measures the **fit** of $\mathcal{Q}$ to $p(\mathbf{z}|\mathbf{e})$.

$\mathcal{L}(\mathcal{Q})$ approximates $\log p(\mathbf{e})$.

Decomposition of the Marginal Likelihood



Variational Inference: Choosing the approximation $\mathcal{Q}$

$\mathcal{Q}$ minimizes KL($\mathcal{Q}||p$) (maximizes $\mathcal{L}(\mathcal{Q})$) when $\mathcal{Q} = p(\mathbf{z}|\mathbf{e})$.

In practice, one selects $\mathcal{Q}$ to be a parametric distribution $\mathcal{Q}(\mathbf{z}|\boldsymbol{\theta})$, for which $\mathcal{L}(\mathcal{Q})$ can be computed analytically, *e.g.*, a Gaussian.

The lower bound then becomes a function of $\boldsymbol{\theta}$ and can be optimized using non-linear optimization techniques such as gradient descent.

An alternative is to assume that $\mathcal{Q}$ factorizes with respect to a partition of $\mathbf{z}$ into M disjoint groups $\mathbf{z}_i$, with $i = 1, \ldots, M$:

$$\mathcal{Q}(\mathbf{z}) = \prod_{i=1}^{M} \mathcal{Q}_i(\mathbf{z}_i)$$

and **no further assumptions are made** about $\mathcal{Q}$.

This approach is known in the literature as variational mean field.

Variational Inference: Variational Mean-Field

Substituting $\mathcal{Q}$ in $\mathcal{L}(\cdot)$ and looking for the dependence with respect to $\mathcal{Q}_j$:

$$\begin{aligned}\mathcal{L}(\mathcal{Q}) &= \sum_{\mathbf{z}} \prod_{i=1}^{M} \mathcal{Q}_i(\mathbf{z}_i) \left\{ \log p(\mathbf{z}, \mathbf{e}) - \sum_{i=1}^{M} \log \mathcal{Q}_i(\mathbf{z}_i) \right\} \\ &= \sum_{\mathbf{z}_j} \left[\mathcal{Q}_j(\mathbf{z}_j) \left\{ \sum_{\mathbf{z}_{i \neq j}} \log p(\mathbf{z}, \mathbf{e}) \prod_{i \neq j}^{M} \mathcal{Q}_i(\mathbf{z}_i) \right\} - \mathcal{Q}_j(\mathbf{z}_j) \log \mathcal{Q}_j(\mathbf{z}_j) \right] + \text{const} \\ &= \sum_{\mathbf{z}_j} \left[\mathcal{Q}_j(\mathbf{z}_j) \log \hat{p}(\mathbf{z}_j, \mathbf{e}) - \mathcal{Q}_j(\mathbf{z}_j) \log \mathcal{Q}_j(\mathbf{z}_j) \right] + \text{const}\end{aligned}$$

which is a negative KL divergence and we have defined

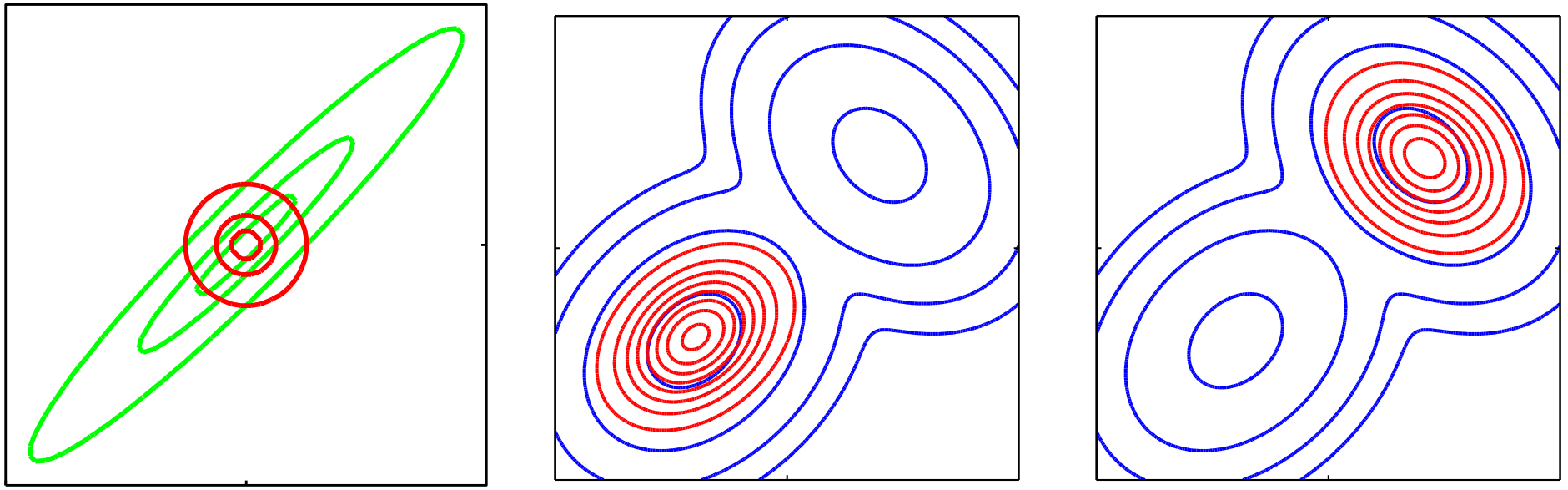
$\log \hat{p}(\mathbf{z}_j, \mathbf{e}) = \mathbb{E}_{i \neq j} \left[\log p(\mathbf{z}, \mathbf{e}) \right] + \text{const}$.

The optimal $\mathcal{Q}_j$ given that the other factors are kept fixed is:

$$\log \mathcal{Q}_j^{\star}(\mathbf{z}_j) = \mathbb{E}_{i \neq j} \left[\log p(\mathbf{z}, \mathbf{e}) \right] + \text{const} \tag{1}$$

To optimize $\mathcal{Q}$ we iterate, optimizing each $\mathcal{Q}_j$ using (1).

Properties of Variational Approximations



The KL divergence $\mathrm{KL}(\mathcal{Q}||p)$ favors solutions that take high probability where p takes high probability, but can ignore important regions.

The optimization problem is not convex and can have multiple local optima.

Variational Inference Example: 2D Ising Model I

Ising models (ferromagnetic or anti-ferromagnetic) are arrays of spins, *e.g.*, atoms that can take states ± 1, that are magnetically coupled to each other.

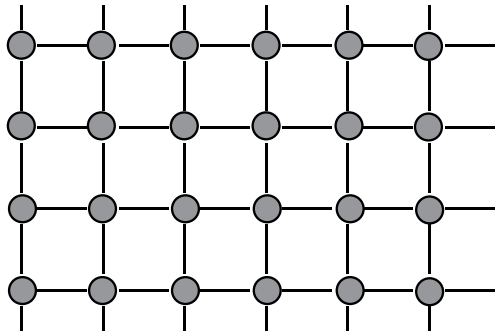
At temperature T, the probability of a spin configuration $\mathbf{z} \in \{-1, 1\}^N$ is

$$p(\mathbf{z}|\beta, J, H) = \frac{1}{Z(\beta, J, H)} \exp\left\{-\beta E(\mathbf{z}; J, H)\right\} ,$$

where $\beta = 1/(k_B T)$, k_B is Boltzmann's constant and Z is a normalizer,

$$E(\mathbf{z}; J, H) = -\left[\frac{1}{2}\sum_{m,n} J_{m,n} z_m z_n + \sum_n H z_n\right]$$

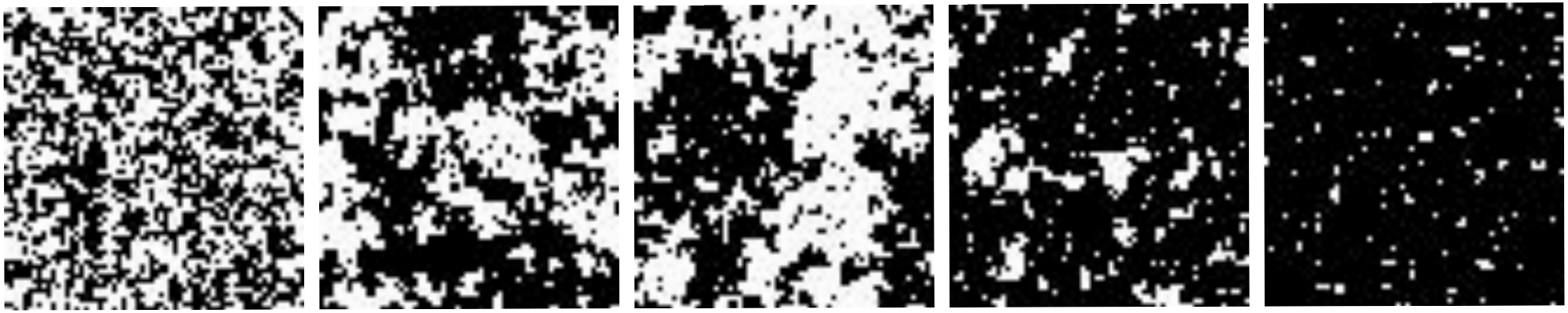
H is the applied field, $J_{m,n} = J$ is m and n are neighbors and 0 otherwise.



Can be described by an un-directed GM.

The evaluation of $p(\mathbf{z}|\beta, J, H)$ is intractable since computing Z requires summing out $\mathbf{z}$.

Variational Inference Example: 2D Ising Model II



As the temperature increases the probability of any state becomes uniform.

For low temperatures it is likely to find all the spins in the same position.

2D Ising Model: Mean Field Approximation I

We choose a factorizing approximation :

$$\mathcal{Q}(\mathbf{z}) = \prod_{i=1}^{N} \mathcal{Q}_i(z_i)\,, \qquad \mathcal{Q}_j(z_j) \propto \exp\{\mathbb{E}_{i\neq j}[\log \tilde{p}(\mathbf{z}|\beta, J, H)]\}\,.$$

which gives a closed form solution for $\mathcal{Q}_j$:

$$\mathcal{Q}_j(z_j = 1) = \frac{\exp(a_j)}{exp(a_j) + exp(-a_j)} = \frac{1}{1 + exp(-2a_j)}\,, \quad a_j = \beta\left(J \sum_{n\in \mathrm{Nb}_j} \overline{z}_n + H\right)\,.$$

where $\overline{z}_n = 1 \cdot \mathcal{Q}_n(z_n = 1) - 1 \cdot \mathcal{Q}_n(z_n = -1) = \tanh(a_n)$.

This process has to be iterated for $j = 1, \ldots, N$ until convergence of $\mathcal{Q}$.

Given $\mathcal{Q}$ it is easy to evaluate the lower bound on $Z(\beta, J, H)$, which can be used to approximate the value of this constant.

Example: Unknown Mean and Variance of a Gaussian

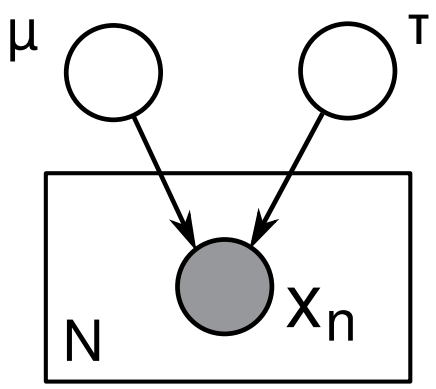
Goal: infer the posterior distribution of the mean μ and precision τ of a Gaussian distribution given a dataset $\mathcal{D} = \{x_1, \ldots, x_N\}$ of independent samples.

The posterior satisfies $p(\mu, \tau|\mathcal{D}) \propto p(\mathcal{D}|\mu, \tau)p(\mu)p(\tau) = p(\mathcal{D}, \mu, \tau)$.

The log likelihood of μ and τ is:

$$\log p(\mathcal{D}|\mu, \tau) = -\frac{N}{2}\log 2\pi\tau^{-1} - \frac{\tau}{2}\sum_{n=1}^{N}(x_n - \mu)^2$$

$$= -\frac{N}{2}\log\tau - \frac{\tau}{2}\left[N(\mu - \overline{x})^2 + S\right] + \text{const}\,,$$



where $S = \sum_n (x_n - \overline{x})^2$ and $\overline{x}$ is the empirical mean.

The priors for μ and τ are uniform:

$$p(\mu) = 1/\sigma_\mu\,, \qquad p(\tau) = 1/\tau$$

Mean Field: Unknown Mean and Variance of a Gaussian I

We enforce that the posterior approximation factorizes $\mathcal{Q}(\mu, \tau) = \mathcal{Q}_\mu(\mu)\mathcal{Q}_\tau(\tau)$ and solve for the optimal factors.

$$\log \mathcal{Q}_\mu(\mu) = \mathbb{E}_{\mathcal{Q}_\tau}\left[\log p(\mathcal{D}, \mu, \tau)\right] + \text{const.}\,,$$
$$\log \mathcal{Q}_\tau(\tau) = \mathbb{E}_{\mathcal{Q}_\mu}\left[\log p(\mathcal{D}, \mu, \tau)\right] + \text{const.}\,,$$

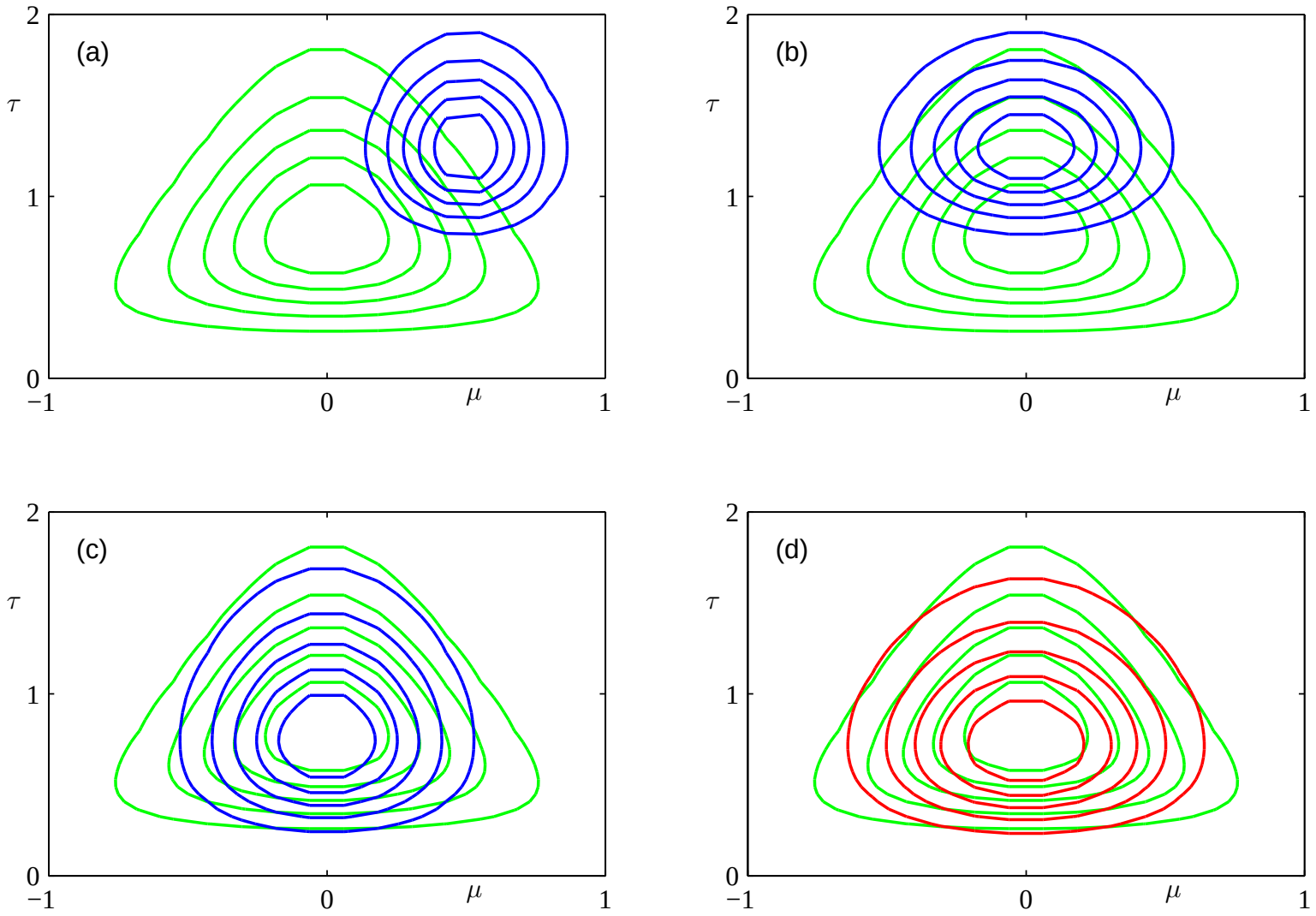
This gives the following optimal factors given that the other factor is fixed:

$$\mathcal{Q}_\mu(\mu) = \mathcal{N}(\mu|\overline{x}, \lambda^{-1})\,, \qquad \mathcal{Q}_\tau(\tau) = \text{Gamma}(\tau|a, b)$$
$$\mathcal{Q}_\tau(\tau) = b^a \frac{1}{\Gamma(a)} \tau^{a-1} \exp\{-b\tau\}\,,$$

where $\lambda = N\mathbb{E}_{\mathcal{Q}_\tau}[\tau] = Na/b$, $a = N/2$ and $b = 1/2(N\lambda^{-1} + S)$.

We iteratively optimize $\mathcal{Q}_\mu$ and $\mathcal{Q}_\tau$ until convergence.

Mean Field: Unknown Mean and Variance of a Gaussian II



Variational Inference: Local Methods I

Bounds over specific factors can be useful to deal with intractable expectations.

Consider the logistic function:

$$\sigma(z) = \frac{1}{1+e^{-z}}, \qquad \log\sigma(z) = \log\left(1+e^{-z}\right) = \frac{z}{2} - \log\left(e^{\frac{z}{2}} + e^{-\frac{z}{2}}\right).$$

The function $f(z) = -\log\left(e^{\frac{z}{2}} + e^{-\frac{z}{2}}\right)$ is convex in terms of z^2. We can hence approximate this function by a tangent line at ξ which is a global lower bound:

$$f(z) \geq f(\xi) + \left.\frac{df(z)}{d(z^2)}\right|_{z=\xi} (z^2 - \xi^2) = -\frac{\xi}{2} + \log\sigma(\xi) + \lambda(\xi)(z^2 - \xi^2).$$

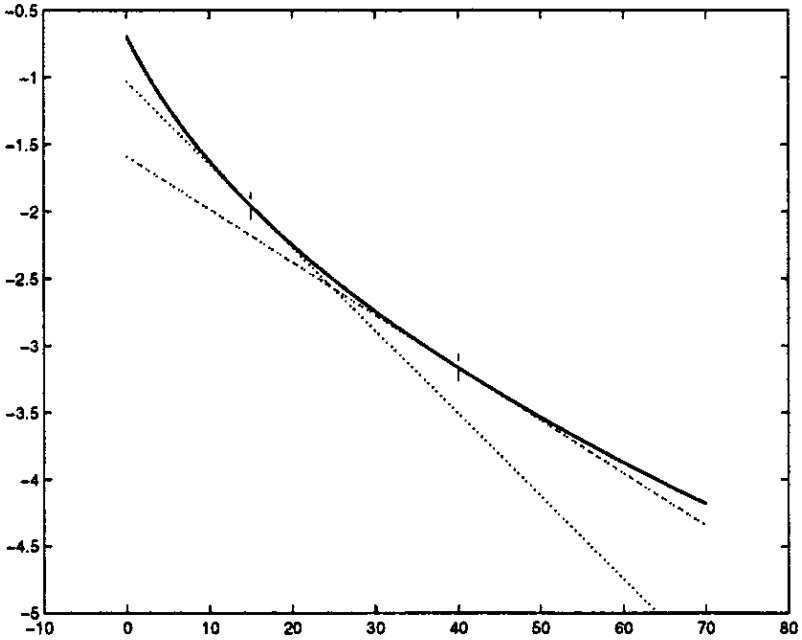
which is tight for $z^2 = \xi^2$ and gives

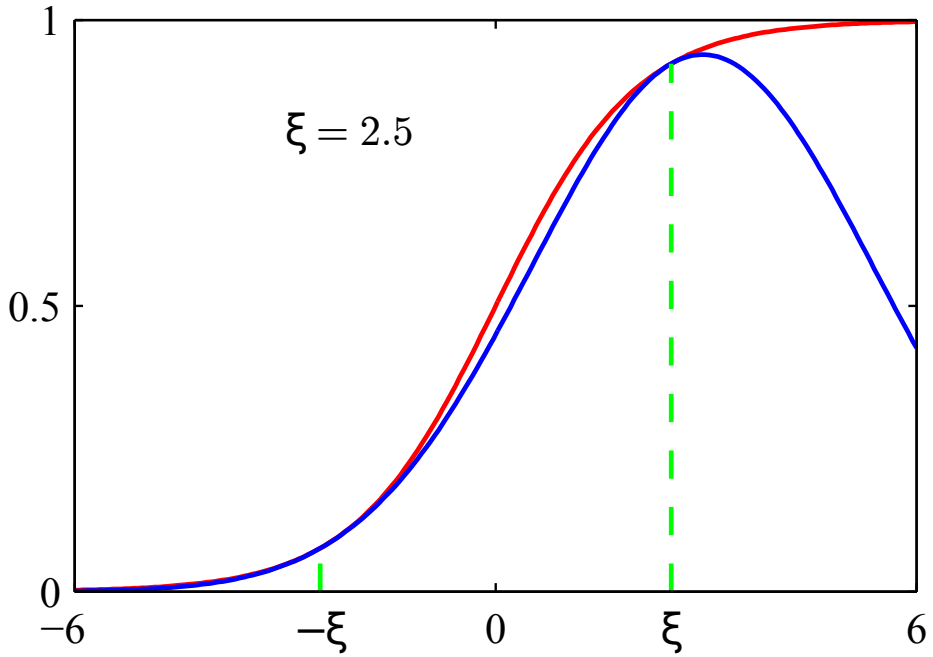
$$\sigma(z) \geq \sigma(\xi)\exp\left\{\frac{z-\xi}{2} - \lambda(\xi)(z^2-\xi^2)\right\} = \underline{\sigma}(z,\xi), \quad \lambda(\xi) = \frac{1}{2\xi}\left[\sigma(\xi) - \frac{1}{2}\right].$$

The lower bound is Gaussian with respect to z.

Can be **maximized** with respect to ξ to find the best fit.

Variational Inference: Local Methods II





A tangent line is a global lower bound of any convex function.

The bound on the right is tight for $z = \xi$ and $z = -\xi$ (green dashed lines).

Variational Inference: Considerations

- The VI approximation $\mathcal{Q}$ tends to be more compact than the exact p.
- Takes into account global properties of the exact distribution, unlike the Laplace approximation.
- Independent of the basis used. The KL divergence is invariant to any variable transformation.
- Computing each factor given that the others are kept fixed is a convex optimization problem.
- However, the global optimization problem need not be convex and there could be local optima.
- Only suitable when the the logarithm of p is tractable.
- If this is not the case, sometimes further approximations can be made.

Expectation Propagation

Daniel Hernández–Lobato[1],

June 29, 2015

slides by

Daniel Hernández-Lobato and José Miguel Hernández-Lobato.

[1]Universidad Autónoma de Madrid.

Introduction

EP can be used to approximate an un-normalized distribution by a simpler parametric distribution, in a similar way as VI.

Also based on the minimization of the KL-divergence, but in its direct way $\mathrm{KL}(p||\mathcal{Q})$ instead of $\mathrm{KL}(\mathcal{Q}||p)$ (the one used by VI).

EP is a generalization of LBP to GM which may contain continuous variables.

The distribution $\mathcal{Q}$ is restricted to belong to a family of probability distributions that is closed under the product operation. This is the exponential family:

$$\mathcal{Q}(\mathbf{z}) = \exp\left(\boldsymbol{\eta}^{\mathsf{T}}\mathbf{u}(\mathbf{z}) - g(\boldsymbol{\eta})\right), \qquad g(\boldsymbol{\eta}) = \log \int \exp\left(\boldsymbol{\eta}^{\mathsf{T}}\mathbf{u}(\mathbf{z})\right) d\mathbf{z}$$

where $\boldsymbol{\eta}$ is a vector of natural parameters of $\mathcal{Q}$, $\mathbf{u}(\mathbf{z})$ are the sufficient statistics and $g(\boldsymbol{\eta})$ is a log normalizer.

Examples of Distributions in the Exponential Family

Gaussian $\mathcal{N}(z|\mu,\sigma^2) = 1/\sqrt{2\pi\sigma^2}\exp\left\{-\frac{1}{2\sigma^2}(z-\mu)^2\right\}$:

$$\boldsymbol{\eta} = (\mu/\sigma^2, -1/(2\sigma^2))^{\mathsf{T}}, \qquad \mathbf{u}(z) = (z, z^2)^{\mathsf{T}}, \qquad g(\boldsymbol{\eta}) = \frac{1}{2}\log\frac{\pi}{-\eta_2} - \frac{\eta_1^2}{4\eta_2}.$$

Multinomial for a single observation $p(\mathbf{z}) = \prod_{k=1}^{M}\mu_k^{z_k}$:

$$\boldsymbol{\eta} = (\log\mu_1, \ldots, \log\mu_M)^{\mathsf{T}}, \qquad \mathbf{u}(\mathbf{z}) = \mathbf{z}, \qquad g(\boldsymbol{\eta}) = 0.$$

Bernoulli $\mathrm{Bern}(z|\mu) = \mu^z(1-\mu)^{1-z}$:

$$\eta = \log\left(\frac{\mu}{1-\mu}\right), \qquad u(z) = z, \qquad g(\eta) = \log(1+\exp(\eta)).$$

Most of the simplest parametric distributions belong to the exponential family.

KL Divergence Minimization

Consider any distribution $p(\mathbf{z})$ and the KL-divergence between p and $\mathcal{Q}$:

$$\mathrm{KL}(p||\mathcal{Q}) = -\int p(\mathbf{z}) \log \left\{ \frac{\mathcal{Q}(\mathbf{z})}{p(\mathbf{z})} \right\} d\mathbf{z} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathsf{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{z})] + \text{const}\,.$$

To minimize $\mathrm{KL}(p||\mathcal{Q})$ with respect to the natural parameters $\boldsymbol{\eta}$ we do

$$\frac{\partial \mathrm{KL}(p||\mathcal{Q})}{\partial \boldsymbol{\eta}} = 0 \Longleftrightarrow \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_p[\mathbf{u}(\mathbf{z})]\,, \qquad \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_{\mathcal{Q}}[\mathbf{u}(\mathbf{z})]\,.$$

Minimizing $\mathrm{KL}(p||\mathcal{Q})$ is equivalent to matching expected sufficient statistics.

If $\mathcal{Q}$ is Gaussian, then we have to match $\mathbb{E}_{\mathcal{Q}}[\mathbf{z}] = \mathbb{E}_p[\mathbf{z}]$ and $\mathbb{E}_{\mathcal{Q}}[\mathbf{z}\mathbf{z}^{\mathsf{T}}] = \mathbb{E}_p[\mathbf{z}\mathbf{z}^{\mathsf{T}}]$.

This result is systematically exploited in EP to carry out approximate inference.

Problem: computing $\mathbb{E}_p[\mathbf{u}(\mathbf{z})]$ is intractable!

Factorization of the Joint Distribution

In many GMs (mainly those that assume i.i.d. data) the joint distribution $p(\mathbf{z}, \mathbf{e})$ of the observed variables $\mathbf{e}$ and the latent variables $\mathbf{z}$ factorizes as

$$p(\mathbf{z}, \mathbf{e}) = \prod_i f_i(\mathbf{z}) \,,$$

where each factor f_1 depends on $\mathbf{z}$ or a subset of these variables.

The factors f_i can be produced by a likelihood or a prior for $\mathbf{z}$.

Given $p(\mathbf{z}, \mathbf{e})$, the posterior for $\mathbf{z}$ is obtained after normalizing by $p(\mathbf{e})$:

$$p(\mathbf{z}|\mathbf{e}) = \frac{1}{p(\mathbf{e})} \prod_i f_i(\mathbf{z}) \,, \qquad p(\mathbf{e}) = \int \prod_i f_i(\mathbf{z}) d\mathbf{z} \,,$$

The Approximation to the Joint Distribution

EP approximates $p(\mathbf{z}, \mathbf{e})$ using a product of simpler factors:

$$p(\mathbf{z}, \mathbf{e}) = \prod_i f_i(\mathbf{z}) \approx \prod_i \tilde{f}_i(\mathbf{z}) \,.$$

Each approximate factor $\tilde{f}_i$ approximates the corresponding exact factor f_i.

The $\tilde{f}_i$ are in an exponential family but need not be normalized. For example, the $\tilde{f}_i$ can be unnormalized Gaussians.

Because the exponential family is closed under the product operation, the product of the $\tilde{f}_i(\mathbf{z})$ has a simple form and can be easily normalized:

$$p(\mathbf{z}|\mathbf{e}) = \frac{1}{p(\mathbf{e})} \prod_i f_i(\mathbf{z}) \approx \frac{1}{Z} \prod_i \tilde{f}_i(\mathbf{z}) = \mathcal{Q}(\mathbf{z}) \,,$$

where $Z = \int \prod_i \tilde{f}_i(\mathbf{z}) d\mathbf{z}$ approximates $p(\mathbf{e})$, the model evidence. Importantly, $\mathcal{Q}$ has the same form as the approximate factors $\tilde{f}_i$.

Updating the Approximate Factors I

How do we adjust the parameters of the approximate factors $\tilde{f}_i$?

Ideally, we would like to minimize $\mathrm{KL}(p||\mathcal{Q})$. However, this involves computing averages with respect to the exact posterior which is intractable. EP minimizes the KL divergence between f_j and $\tilde{f}_j$ in the context of all the other approximate factors $\tilde{f}_i$, $i \neq j$. This ensures that $\tilde{f}_j$ is accurate where $\mathcal{Q}^{\setminus j} = \prod_{i \neq j} \tilde{f}_i$ takes large values.

To refine $\tilde{f}_j$, we first remove it from $\mathcal{Q}$: $\mathcal{Q}^{\setminus j}(\mathbf{z}) \propto \prod_{i \neq j} \tilde{f}_i(\mathbf{z}) = \mathcal{Q}(\mathbf{z})/\tilde{f}_j(\mathbf{z})$.

We then adjust $\tilde{f}_j$ so that the distributions

$$\mathcal{Q}_{\text{new}}(\mathbf{z}) \propto \tilde{f}_j(\mathbf{z})\mathcal{Q}^{\setminus j}(\mathbf{z}) \quad \text{and} \quad \hat{p}(\mathbf{z}) = \frac{1}{Z_j} f_j(\mathbf{z})\mathcal{Q}^{\setminus j}(\mathbf{z}), \quad Z_j = \int f_j(\mathbf{z})\mathcal{Q}^{\setminus j}(\mathbf{z})d\mathbf{z},$$

are as close as possible in terms of the KL divergence, where $\mathcal{Q}^{\setminus j}$ is kept fixed.

Updating the Approximate Factors II

First, we minimize $\mathrm{KL}(Z_j^{-1} f_j(\mathbf{z}) \mathcal{Q}^{\setminus j}(\mathbf{z}) || \mathcal{Q}_{\text{new}}(\mathbf{z}))$ with respect to $\mathcal{Q}_{\text{new}}$.

Done by matching expected sufficient statistics between $\mathcal{Q}_{\text{new}}$ and $1/Z_j f_j \mathcal{Q}^{\setminus j}$.
For this, expectations with respect to $1/Z_j f_j \mathcal{Q}^{\setminus j}$ must be tractable.

Then $\tilde{f}_j$ is updated using

$$\tilde{f}_j(\mathbf{z}) = Z_j \frac{\mathcal{Q}_{\text{new}}(\mathbf{z})}{\mathcal{Q}^{\setminus j}(\mathbf{z})}, \qquad \text{recall that} \quad \mathcal{Q}_{\text{new}} \propto \tilde{f}_j(\mathbf{z}) \mathcal{Q}^{\setminus j}(\mathbf{z}),$$

which ensures that $\tilde{f}_j(\mathbf{z}) \mathcal{Q}^{\setminus j}(\mathbf{z})$ and $f_j(\mathbf{z}) \mathcal{Q}^{\setminus j}(\mathbf{z})$ integrate the same.

Several passes are made trough the factors until they converge.

The model evidence is approximated by the normalizing constant of the product of all the $\tilde{f}_i$.

The Expectation Propagation Algorithm

Computes $\mathcal{Q}$ and an approximation to the model evidence.

1. Initialize $\mathcal{Q}$ and each $\tilde{f}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{f}_i$:
   1. Choose a factor $\tilde{f}_j$ to refine.
   2. Remove $\tilde{f}_j$ from $\mathcal{Q}$ by division $\mathcal{Q}^{\setminus j} = \mathcal{Q}/\tilde{f}_j$.
   3. Compute Z_j and find $\mathcal{Q}_{\text{new}}$ by minimizing $\text{KL}(\hat{p}||\mathcal{Q}_{\text{new}})$.
   4. Compute and store the new factor $\tilde{f}_j = Z_j \mathcal{Q}_{\text{new}}/\mathcal{Q}^{\setminus j}$.
3. Evaluate the approximation to the model evidence:

$$p(\mathbf{e}) \approx Z = \int \prod_i \tilde{f}_j(\mathbf{z}) d\mathbf{z} \,.$$

A simplification is known as assumed density filtering (ADF).

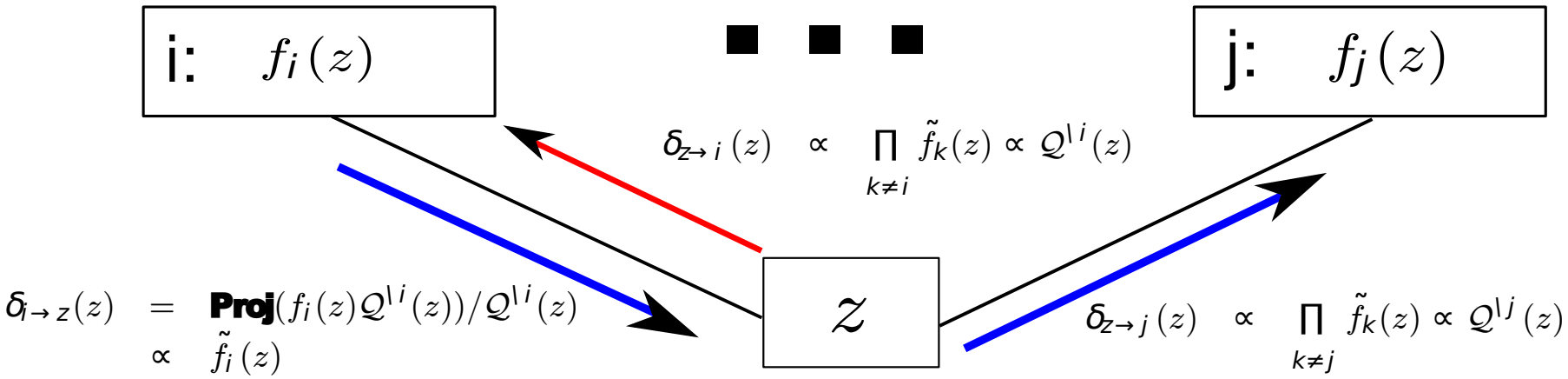
In ADF only one pass is done for each factor (faster but less accurate).

Expectation Propagation as a Message Passing Algorithm

EP is a generalization of LBP with **approximate messages** in a cluster graph, often the Bethe cluster graph. If there is no approximation they are **equivalent**.

In **LBP** the messages are **factors** (the product of factors is **another factor**). **EP** keeps messages consistent by **projecting** to the chosen **exponential family**.

The **approximate messages** sent in EP are the approximate factors $\tilde{f}_i$.



Node i sends a message $\tilde{f}_i$ to node z. This last node sends a message with $\mathcal{Q}^{\setminus j}$ to node j. At convergence, the clusters are approximately calibrated and the **product of the messages** in node z give $\mathcal{Q}$. Note the **division** in the computations carried out at node i.

Expectation Propagation: Considerations

- The minimization of the KL is done by moment matching.
- EP may not converge and the $\tilde{f}_i$ may oscillate forever (same as in LBP).
- Convergence can be improved by damping the EP updates.
- As with loopy BP, the convergence points of EP can be shown to be stationary points of a particular energy function which need not be convex. There can be multiple convergence points of EP.
- It is possible to design convergent versions of EP that directly attempt to optimize the energy function. However, they are much more expensive and most times EP converges successfully.
- No need to replace all the factors in the joint distribution with approximations. For example, if one factor is already in the exponential family, the approximate factor is always the same and exact.
- EP considers global aspects of p by approximately minimizing $\mathrm{KL}(p|\mathcal{Q})$.

EP Example: The Clutter Problem

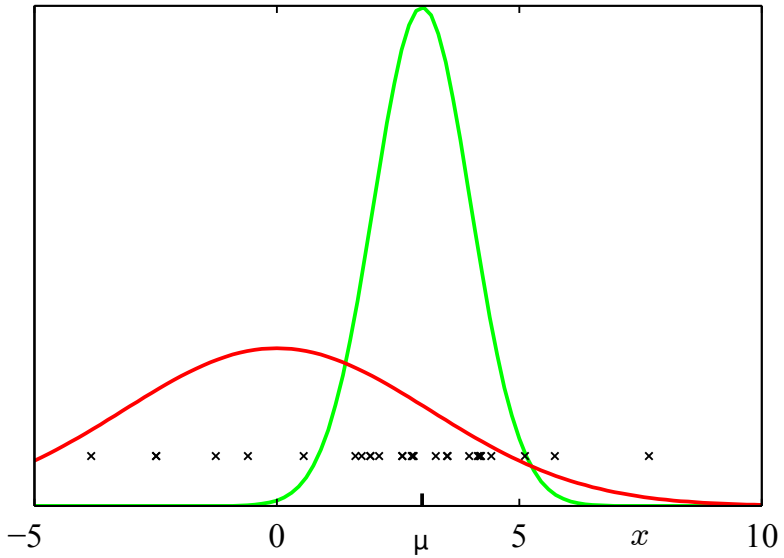
We consider the problem of inferring the mean $\boldsymbol{\mu}$ of a multivariate Gaussian when the Gaussian observations are embedded in background Gaussian clutter.

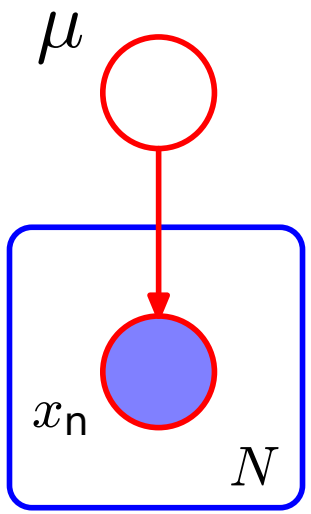
In this problem $\mathbf{z} = \boldsymbol{\mu}$ and $\mathbf{e}$ are the observations $\mathbf{x}$, which are generated from:

$$p(\mathbf{x}|\boldsymbol{\mu}) = (1 - w)\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{I}) + w\mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}a)\,,$$

where $w = 0.5$ is the proportion of clutter and $a = 10$.

The prior for $\boldsymbol{\mu}$ is $p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|0, \mathbf{I}b)$ with $b = 100$ (little informative).





Factorization of the Joint Distribution

The joint distribution of $\boldsymbol{\mu}$ and the evidence $\mathbf{e} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is

$$p(\boldsymbol{\mu}, \mathbf{e}) = p(\boldsymbol{\mu}) \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\mu}) = f_0(\boldsymbol{\mu}) \prod_{i=1}^{N} f_i(\boldsymbol{\mu}) \,,$$

a mixture of 2^N terms. Computing $p(\boldsymbol{\mu}|\mathbf{e})$ is intractable for large N.

We choose a parametric form for $\mathcal{Q}$ that belongs to the exponential family:

$$\mathcal{Q}(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}, v\mathbf{I}) \,, \qquad \tilde{f}_i(\boldsymbol{\mu}) = \tilde{s}_i \mathcal{N}(\boldsymbol{\mu}|\tilde{\mathbf{m}}_i, \tilde{v}_i \mathbf{I}) \,,$$

with parameters $\mathbf{m}$, $\{\tilde{\mathbf{m}}_i\}_{i=0}^N$, $\{\tilde{s}_i\}_{i=0}^N$, $\{\tilde{v}_i\}_{i=0}^N$ and v.

The $\tilde{f}_i$ are not densities and negative values for $\tilde{v}_i$ are valid.

f_0 can be approximated exactly and the optimal choice for $\tilde{f}_0$ is $\tilde{f}_0 = f_0$.

Once initialized, this term needs not be updated by EP anymore.

Gaussian Identities I

The product and ratio of Gaussians is again Gaussian.

$$\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \cdot \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = C\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}, \qquad \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right),$$

$$C = \sqrt{\frac{|\boldsymbol{\Sigma}|}{(2\pi)^d|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\mu}_1^{\mathsf{T}}\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^{\mathsf{T}}\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right\}.$$

$$\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)/\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = C\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

$$\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})^{-1}, \qquad \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right),$$

$$C = \sqrt{\frac{|\boldsymbol{\Sigma}||\boldsymbol{\Sigma}_2|(2\pi)^d}{|\boldsymbol{\Sigma}_1|}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\mu}_1^{\mathsf{T}}\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^{\mathsf{T}}\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right\}.$$

Gaussian Identities II

Let $f(\mathbf{x})$ be an arbitrary factor of $\mathbf{x}$ and let

$$Z = \int t(\mathbf{x})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\,, \qquad \hat{p}(\mathbf{x}) = \frac{1}{Z} t(\mathbf{x})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\,,$$

Then, we have that

$$\mathbb{E}_{\hat{p}}[\mathbf{x}] = \boldsymbol{\mu} + \boldsymbol{\Sigma}\frac{\partial \log Z}{\partial \boldsymbol{\mu}}\,,$$

$$\mathbb{E}_{\hat{p}}[\mathbf{x}\mathbf{x}^\mathsf{T}] - \mathbb{E}_{\hat{p}}[\mathbf{x}]\mathbb{E}_{\hat{p}}[\mathbf{x}]^\mathsf{T} = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\left(\frac{\partial \log Z}{\partial \boldsymbol{\mu}}\left(\frac{\partial \log Z}{\partial \boldsymbol{\mu}}\right)^\mathsf{T} - 2\frac{\partial \log Z}{\partial \boldsymbol{\Sigma}}\right)\boldsymbol{\Sigma}\,.$$

These expressions are very useful to find the parameters of $\mathcal{Q}_{\text{new}}$ in EP.

Initialization and Computation of $\mathcal{Q}^{\setminus i}$

The $\tilde{f}_i$ are initialized to be non-informative, $\mathcal{Q}$ is also non-informative:

$$\tilde{s}_i = (2\pi\tilde{v}_i)^{\frac{D}{2}}, \quad \tilde{\mathbf{m}}_i = \mathbf{0}, \quad \tilde{v}_i \to \infty, \quad \mathbf{m} = \mathbf{0}, \quad v = b, \qquad \text{for } i = 1, \ldots, N.$$

where we have used the Gaussian identities.

After refining $\tilde{f}_0$, $\mathcal{Q}$ is equal to the prior $p(\boldsymbol{\mu})$.

The first step to refine $\tilde{f}_i$ with $i = 1, \ldots, N$, is to compute $\mathcal{Q}^{\setminus i}$ using

$$\mathcal{Q}^{\setminus i}(\boldsymbol{\mu}) \propto \mathcal{Q}(\boldsymbol{\mu})/\tilde{f}_i(\boldsymbol{\mu}) \propto \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}^{\setminus i}, \mathbf{I}v^{\setminus i}),$$

where we use the Gaussian identities again to get

$$\mathbf{m}^{\setminus i} = v^{\setminus i}(\mathbf{m}v^{-1} - \tilde{\mathbf{m}}_i\tilde{v}_i^{-1}), \qquad (v^{\setminus i})^{-1} = v^{-1} - \tilde{v}_i^{-1}.$$

Computation of the New Posterior $\mathcal{Q}_{\text{new}}$

The first step to update $\tilde{f}_i$ is to compute Z_i:

$$Z_i = \int f_i(\boldsymbol{\mu}) \mathcal{Q}^{\setminus i}(\boldsymbol{\mu}) d\boldsymbol{\mu} = (1-w)\mathcal{N}(\mathbf{x}_i|\mathbf{m}^{\setminus i}, (v^{\setminus i}+1)\mathbf{I}) + w\mathcal{N}(\mathbf{x}_i|\mathbf{0}, a\mathbf{I})\,.$$

which is obtained from the convolution of two Gaussians.

Next, we compute $\mathcal{Q}_{\text{new}}$ by finding the mean and the variance of $f_i \mathcal{Q}^{\setminus i}$:

$$\mathbf{m}_{\text{new}} = \mathbf{m}^{\setminus i} + \rho_i \frac{v^{\setminus i}}{v^{\setminus i}+1}(\mathbf{x}_i - \mathbf{m})\,,$$

$$v_{\text{new}} = v^{\setminus i} - \rho_i \frac{(v^{\setminus i})^2}{v^{\setminus i}+1} + \rho_i(1-\rho_i)\frac{(v^{\setminus i})^2 ||\mathbf{x}_i - \mathbf{m}^{\setminus i}||^2}{D(v^{\setminus i}+1)^2}\,,$$

where we have used again the Gaussian identities and

$$\rho_i = 1 - \frac{w}{Z_i}\mathcal{N}(\mathbf{x}_i|\mathbf{0}, a\mathbf{I})$$

can be interpreted as the probability of $\mathbf{x}_i$ not being clutter.

Update of the Approximate Factor $\tilde{f}_i$

$\tilde{f}_i$ is updated to be equal to $Z_i \mathcal{Q}_{\text{new}} / \mathcal{Q}^{\setminus i}$:

$$
\begin{aligned}
(\tilde{v}_i)^{-1} &= (v_{\text{new}})^{-1} - (v^{\setminus i})^{-1} , \\
\tilde{\mathbf{m}}_i &= \tilde{v}_i \left(v_{\text{new}}^{-1} \mathbf{m}_{\text{new}} - (v^{\setminus i})^{-1} \mathbf{m}^{\setminus i} \right) , \\
\tilde{s}_i &= \frac{Z_i}{\mathcal{N}(\tilde{\mathbf{m}}_i | \mathbf{m}^{\setminus i}, (\tilde{v}_i + v^{\setminus i}) \mathbf{I})} ,
\end{aligned}
$$

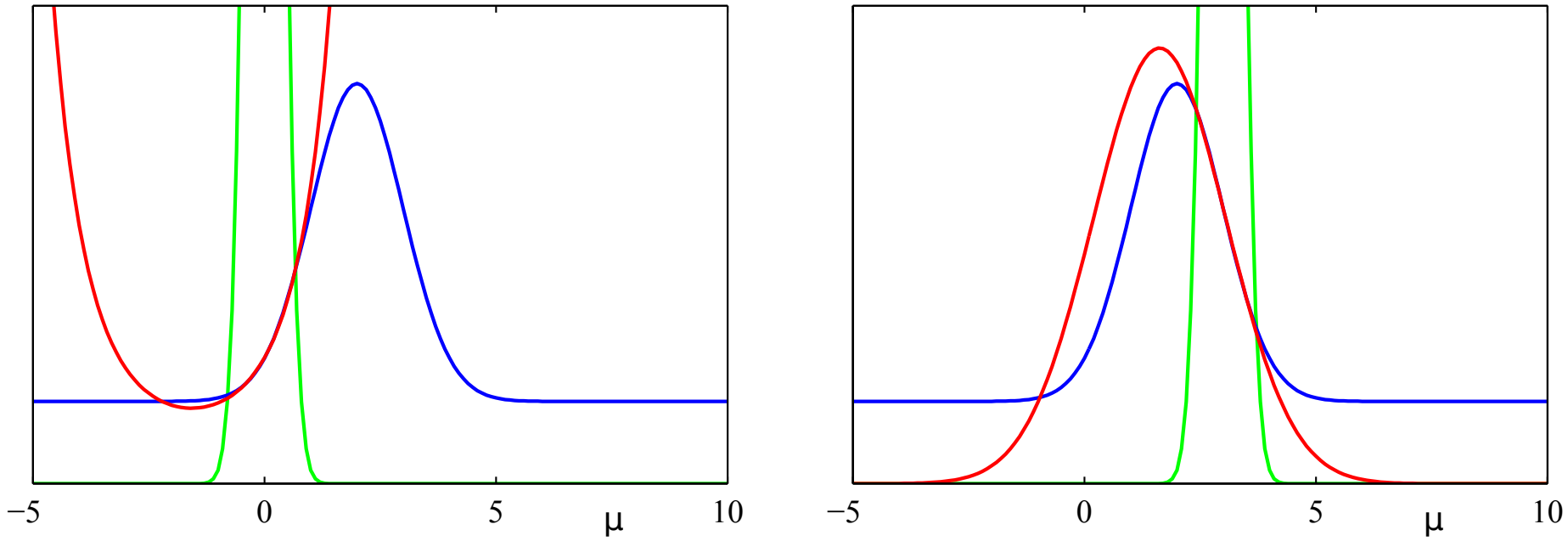
where we used the Gaussian identities.

At convergence we evaluate the approximation of the marginal likelihood:

$$
p(\mathbf{e}) \approx \int \prod_{i=0}^{N} \tilde{f}_i(\boldsymbol{\mu}) d\boldsymbol{\mu} = (2\pi v_{\text{new}})^{D/2} \exp(B/2) \prod_{i=0}^{N} \left[\tilde{s}_i (2\pi \tilde{v}_i)^{-D/2} \right] ,
$$

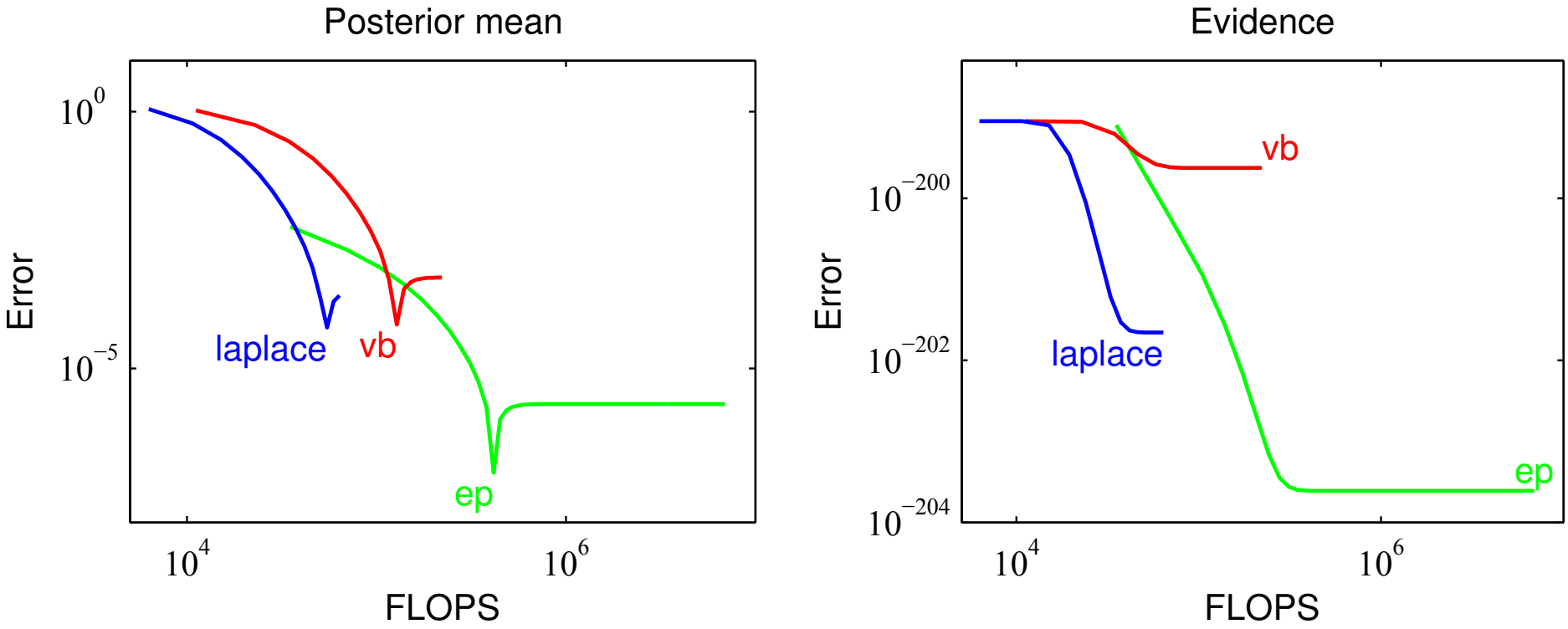
where $B = \mathbf{m}_{\text{new}}^{\mathsf{T}} v_{\text{new}}^{-1} \mathbf{m}_{\text{new}} - \sum_{i=0}^{N} \tilde{\mathbf{m}}^{\mathsf{T}} (\tilde{v}_i)^{-1} \tilde{\mathbf{m}}$ and we have used the Gaussian identities.

EP Example: Computed Approximations of f_i



Approximation of specific factors f_i when $D = 1$. Exact factor $f_i(\mu)$ is shown in blue (a Gaussian plus a constant), approximate factor $\tilde{f}_i(\mu)$ is shown in red, and $\mathcal{Q}^{\setminus i}(\mu)$ in green. The Gaussian approximation is accurate in regions of high posterior probability as estimated by $\mathcal{Q}^{\setminus i}$.

EP Example: Comparison with VI and Laplace



Comparison of EP with Laplace's method and Variational Inference (mean field) on the clutter problem. Accuracy is measured in absolute difference from the true mean and the true integral. Cost is measured in FLOPS (floating point operations).

References

- Koller, D. and Friedman, N. Probabilistic Graphical Models: Principles and Techniques MIT Press, 2009.
- Bishop, C. M. Pattern Recognition and Machine Learning (Information Science and Statistics) Springer, 2006.
- MacKay, D. J. C. Information Theory, Inference and Learning Algorithms Cambridge University Press, 2003.
- Barber, D. Bayesian Reasoning and Machine Learning Cambridge University Press, 2011.
- Kevin Patrick Murphy, Machine Learning: A Probabilistic Perspective, 2012.

Acknowledgment of figures: Several of the figures contained in these slides have been extracted for educational purposes from the references above.