

CS5340

Uncertainty Modeling in AI

Lecture 10: Variational Inference

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 1

Course Schedule

Week	Date	Topic	Remarks
1	10 Aug	Introduction to probabilistic reasoning	Assignment 0: Python Numpy Tutorial (Ungraded)
2	17 Aug	Bayesian networks (Directed graphical models)	
3	24 Aug	Markov random Fields (Undirected graphical models)	
4	31 Aug	Variable elimination and belief propagation	Assignment 1: Belief propagation and maximal probability (15%)
5	07 Sep	Factor graph and the junction tree algorithm	
6	14 Sep	Parameter learning with complete data	Assignment 1: Due Assignment 2: Junction tree and parameter learning (15%)
-	21 Sep	Recess week	No lecture
7	28 Sep	Mixture models and the EM algorithm	Assignment 2: Due
8	05 Oct	Hidden Markov Models (HMM)	Assignment 3: Hidden Markov model (15%)
9	12 Oct	Monte Carlo inference (Sampling)	
*	15 Oct	Variational inference	Makeup Lecture (LT15) Time: 9.30am – 12.30pm (Saturday)
10	19 Oct	Variational Auto-Encoder and Mixture Density Networks	Assignment 3: Due Assignment 4: MCMC Sampling (15%)
11	26 Oct	No Lecture	I will be traveling
12	02 Nov	Graph-cut and alpha expansion	Assignment 4: Due
13	09 Nov	-	

Acknowledgements

- A lot of slides and content of this lecture are adopted from:
 1. Christopher Bishop, “Pattern Recognition and Machine Learning”, Chapter 10.
 2. Kevin Murphy, "Machine learning: a probabilistic approach“, Chapter 21 and 22.
 3. David Barber, "Bayesian reasoning and machine learning“, Chapter 28.
 4. Daphne Koller and NirFriedman, "Probabilistic graphical models“, Chapter 11.

Learning Outcomes

- Students should be able to:
 1. Explain the concept of variational approach using **Lower-Bound** of maximum likelihood and **KL-divergence**.
 2. Use **variational approach** to do inference on graphical models containing both hidden variables and unknown parameters.
 3. Use **Loopy Belief Propagation** to do message passing.

Approximate Inference

- A central task in the application of probabilistic models is the **evaluation of the posterior distribution** $p(z|x)$.
- And the **evaluation of expectations** computed with respect to this distribution.
- Z is the **latent variables** (including the unknown parameters θ) and X is the **observed variables**.

Approximate Inference

- Could be **infeasible** to evaluate the posterior distribution or to compute expectations with respect to this distribution due to:
 1. The **dimensionality is too high** in the latent space to work with directly.
 2. The posterior distribution has a highly complex form for which expectations are **not analytically tractable**.

Approximate Inference

- In such situations, we need to resort to **approximation schemes**, and these fall broadly into two classes:
 1. **Stochastic approximation**: Markov Chain Monte-Carlo (MCMC).
 2. **Deterministic approximation**: Variational approach.

Variational Approach

- Given a joint distribution $p(x, z)$, our goal is to **find an approximation** $q(z)$ for the posterior distribution $p(z|x)$ which is intractable.
- **Key idea:** We choose the approximation $q(z)$ such that it **minimizes the KL-divergence**

$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z} \geq 0.$$

- i.e. $q(z)$ is **as close as possible** to $p(z|x)$.

Variational Approach

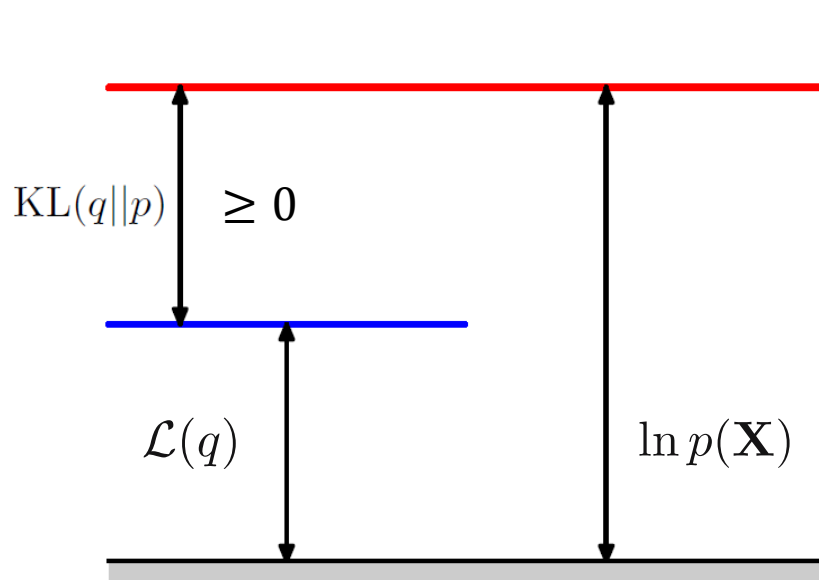
$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z} \geq 0$$

- Unfortunately, minimizing the KL-divergence **is hard** since $p(z|x)$ is intractable.
- **Solution:** **Maximize the lower-bound** of log-likelihood instead!

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

Variational Approach

- Recall in our discussion of EM, we can decompose the log marginal probability using:



$$\ln p(\mathbf{X}) = \underbrace{\mathcal{L}(q)}_{\text{Lower bound}} + \underbrace{\text{KL}(q||p)}_{\text{KL-divergence}}$$

Lower bound KL-divergence

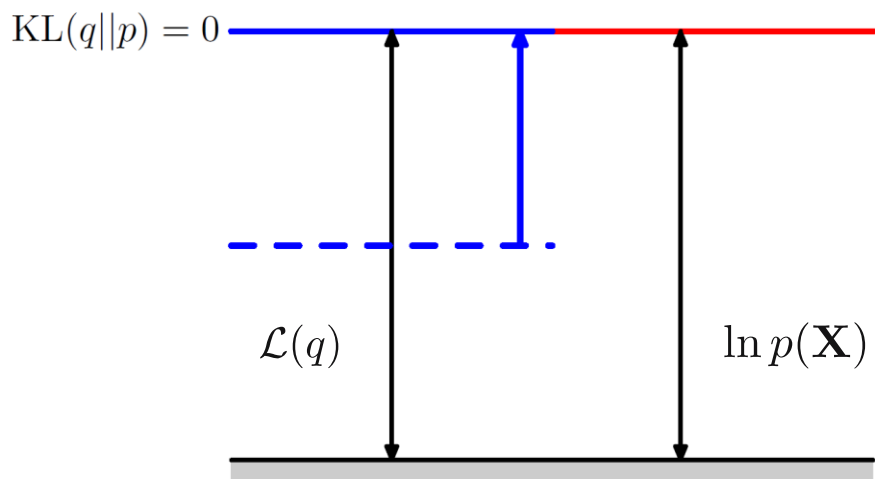
where

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}.$$

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Variational Approach



$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \text{KL}(q||p)$$

where

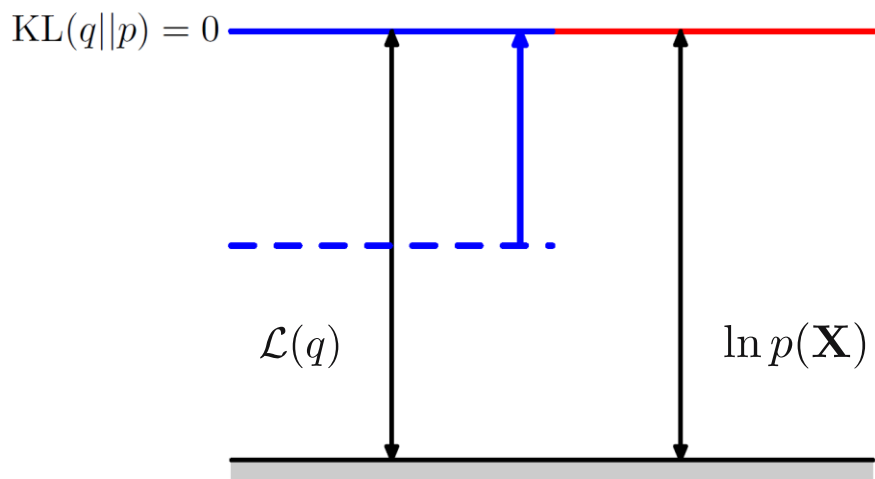
$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}.$$

- **Maximizing the lower bound $\mathcal{L}(q)$ by optimization with respect to the distribution $q(Z)$ is equivalent to minimizing the KL divergence.**

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Variational Approach



$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \text{KL}(q||p)$$

where

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}.$$

- What is a **good approximation distribution** $q(\mathbf{z})$ for the maximization of the lower-bound $\mathcal{L}(q)$?

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Mean-Field Approximation

- **Mean-field theory**: an approximation framework developed in physics.
- Suppose we **partition the elements** of Z into disjoint groups that we denote by Z_i where $i = 1, \dots, M$.
- We then assume that **$q(\mathbf{z})$ factorizes** with respect to these groups, so that:

$$q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i).$$

Mean-Field Approximation

- We now **seek that distribution** for which the lower bound $\mathcal{L}(q)$ is largest amongst all distributions $q(z)$ having the form:

$$q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i).$$

- Make a **free form (variational) optimization** of $\mathcal{L}(q)$ with respect to all the distributions $q_i(z_i)$, which we do by optimizing w.r.t. each of the factors in turn.

Mean-Field Approximation

- Putting the factorized distribution of $q(\mathbf{z})$ into the lower-bound $\mathcal{L}(q)$, and **dissect out the dependence on one of the factors $q_j(\mathbf{z}_j)$** , we get:

$$\begin{aligned}\mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} d\mathbf{Z} \\ &= \iint q_j \prod_{i \neq j} q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i - \ln q_j \right\} d\mathbf{Z}_i d\mathbf{Z}_j \\ &= \iint q_j \prod_{i \neq j} q_i \ln p(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}_i d\mathbf{Z}_j - \iint q_j \prod_{i \neq j} q_i \sum_i \ln q_i d\mathbf{Z}_i d\mathbf{Z}_j - \\ &\quad \iint q_j \prod_{i \neq j} q_i \ln q_j d\mathbf{Z}_i d\mathbf{Z}_j\end{aligned}$$

Mean-Field Approximation

- Putting the factorized distribution of $q(\mathbf{z})$ into the lower-bound $\mathcal{L}(q)$, and **dissect out the dependence on one of the factors $q_j(\mathbf{z}_j)$** , we get:

$$\begin{aligned}
 \mathcal{L}(q) &= \iint q_j \prod_{i \neq j} q_i \ln p(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}_i d\mathbf{Z}_j - \iint q_j \prod_{i \neq j} q_i \sum_i \ln q_i d\mathbf{Z}_i d\mathbf{Z}_j - \\
 &\quad \iint q_j \prod_{i \neq j} q_i \ln q_j d\mathbf{Z}_i d\mathbf{Z}_j \quad \text{Const. (independent of } \mathbf{Z}_j) \\
 &= \int q_j \left\{ \int \prod_{i \neq j} q_i \ln p(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}_i \right\} d\mathbf{Z}_j - \boxed{\iint q_j d\mathbf{Z}_j} \prod_{i \neq j} q_i \sum_i \ln q_i d\mathbf{Z}_i - \\
 &\quad \int q_j \ln q_j \boxed{\int \prod_{i \neq j} q_i d\mathbf{Z}_i} d\mathbf{Z}_j
 \end{aligned}$$

$\xrightarrow{\text{blue arrow}} = 1$
 $\xrightarrow{\text{blue arrow}} = 1$

Mean-Field Approximation

- Putting the factorized distribution of $q(\mathbf{z})$ into the lower-bound $\mathcal{L}(q)$, and **dissect out the dependence on one of the factors $q_j(\mathbf{z}_j)$** , we get:

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \left\{ \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right\} d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const}\end{aligned}$$

We want to maximize w.r.t. each of the $q_j(\mathbf{z}_j)$!

Mean-Field Approximation

- We have defined a **new distribution** $\tilde{p}(x, z_j)$ by the relation:

$$\ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] ;$$

where

$$\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] = \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i$$

- The notation $\mathbb{E}_{i \neq j} [\dots]$ denotes an **expectation** w.r.t. the q distributions over all variables Z_i for $i \neq j$.

Mean-Field Approximation

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \\ &= \int q_j \ln \frac{\tilde{p}(\mathbf{x}, \mathbf{z}_j)}{q_j} d\mathbf{Z}_j + \text{const} \\ &= -KL(q_j \parallel \tilde{p}(\mathbf{x}, \mathbf{z}_j)) + \text{const}\end{aligned}$$

- The lower-bound is a **negative KL-divergence** that can be **maximized by choosing**

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$$

Mean-Field Approximation

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$$

- The **additive constant** is set by normalizing the distribution $q_j^*(z_j)$, taking the exponential of both sides and normalize, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\overbrace{\exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})])}^{\tilde{p}(\mathbf{x}, \mathbf{z}_j)}}{\underbrace{\int \exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j}_{\tilde{p}(\mathbf{x}, \mathbf{z}_j)}}.$$

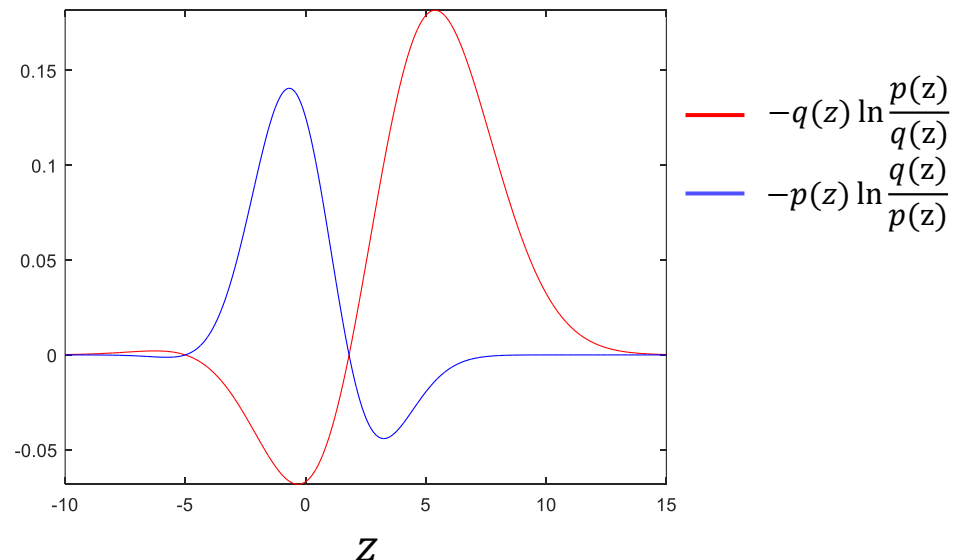
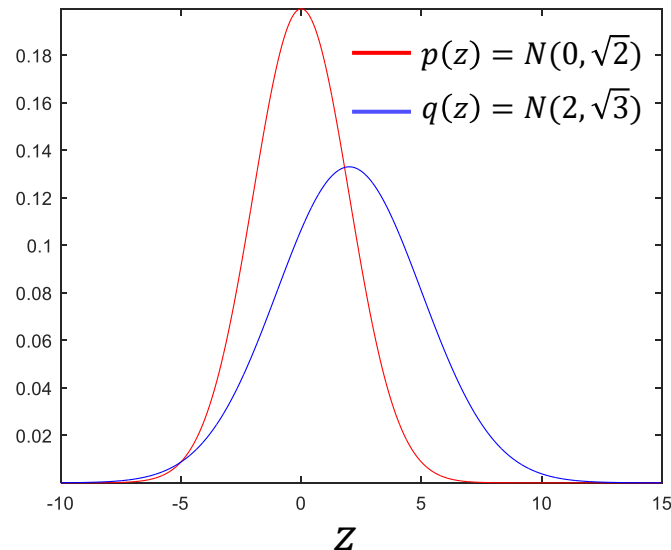
- Remark:** Although the choice of $q_j^* = \tilde{p}(\mathbf{x}, \mathbf{z}_j)$ gives a lower $\mathcal{L}(q)$, we do not choose this as the solution because q_j^* has to be a proper **normalized probability distribution**.

Forward vs Backward KL-Divergence

- KL-divergence is **not symmetrical**, minimizing $KL(q \parallel p)$ and $KL(p \parallel q)$ will give different results.

$$KL(q \parallel p) = -\sum_z q(z) \ln \frac{p(z)}{q(z)}, \quad KL(p \parallel q) = -\sum_z p(z) \ln \frac{q(z)}{p(z)}$$

$KL(. \parallel .) = \text{area under curve}$



Forward vs Backward KL-Divergence

- First, consider the **reverse KL**, $KL(q \parallel p)$, also known as an **I-projection** or **information projection**:

$$KL(q \parallel p) = \sum_z q(z) \ln \frac{q(z)}{p(z)}$$

- This is infinite if $p(z) = 0$ and $q(z) > 0$; thus if $p(z) = 0$ we must ensure $q(z) = 0$.
- We say that reverse KL is **zero forcing** for q , hence q will **typically under-estimate** the support of p .

Forward vs Backward KL-Divergence

- Next, consider the **forward KL**, also known as an **M-projection** or **moment projection**:

$$KL(p \parallel q) = \sum_z p(z) \ln \frac{p(z)}{q(z)}$$

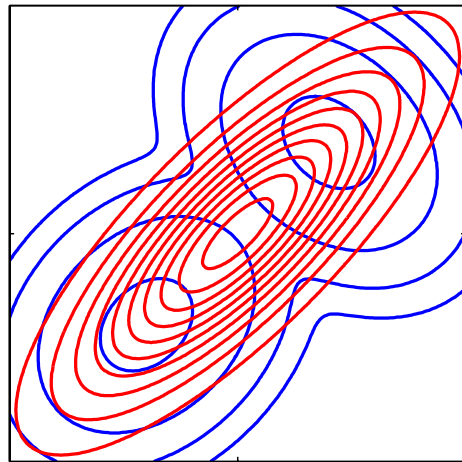
- This is infinite if $q(x) = 0$ and $p(z) > 0$, thus if $p(z) > 0$, we must ensure $q(z) > 0$.
- We say that forward KL is **zero avoiding** for q , hence q will **typically over-estimate** the support of p .

Forward vs Backward KL-Divergence

Blue contours: Bimodal distribution $p(Z)$

Red contours: $q(Z)$ that best approximates $p(Z)$

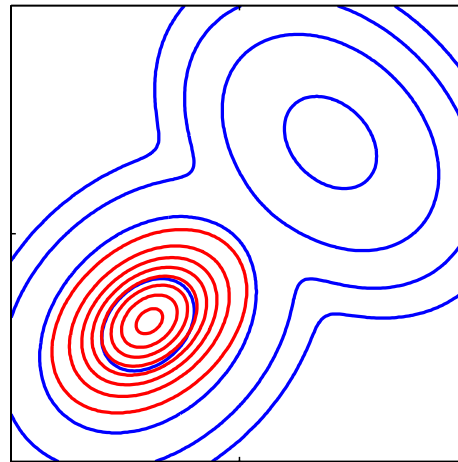
(Forward KL-divergence)
Minimizing $KL(p \parallel q)$



(a)

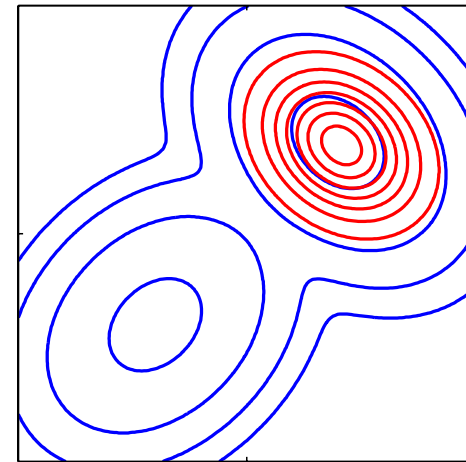
Minimizing $KL(q \parallel p)$ (Reverse KL-divergence)

Local minima 1



(b)

Local minima 2



(c)

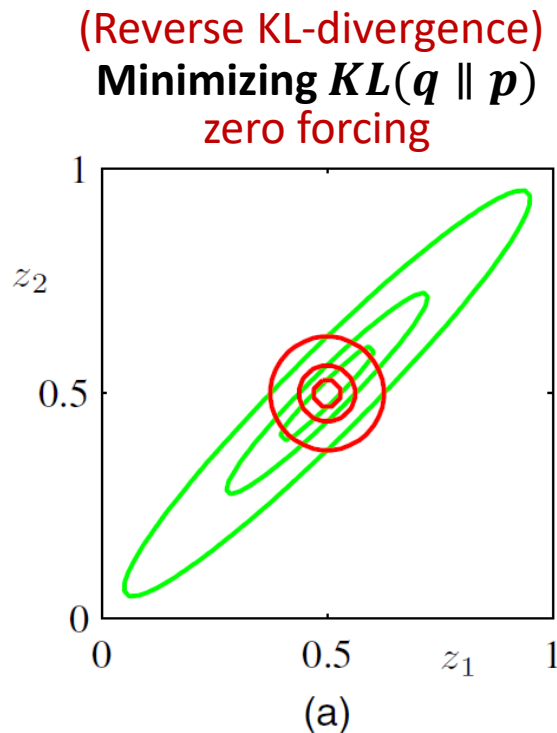
$q(Z)$ is nonzero in regions where $p(Z)$ is nonzero, i.e. **zero avoiding**

$q(Z)$ is small when $p(Z)$ is small, i.e. **zero forcing**

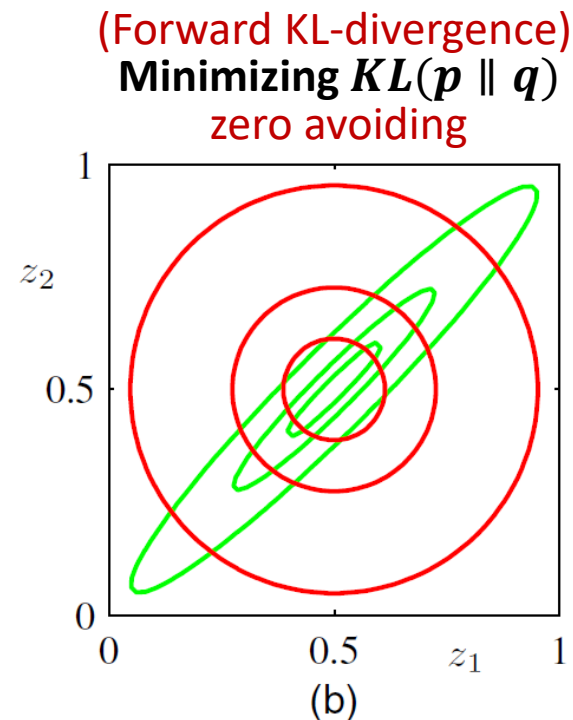
Forward vs Backward KL-Divergence

Red contours: approximating distribution $q(z)$

Green contours: Gaussian distribution $p(z)$



Under-estimation



Over-estimation

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Forward or Backward KL-Divergence?

- We use **Backward KL-divergence** $KL(q \parallel p)$ because it leads to a **tractable lower-bound**:

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

- Compared to forward **KL-divergence** $KL(p \parallel q)$ which leads to an **intractable lower-bound**:

$$\mathcal{L}(p) = \int p(\mathbf{Z}|\mathbf{X}) \ln \left\{ \frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z}|\mathbf{X})} \right\} d\mathbf{Z}$$

Intractable to compute $p(\mathbf{Z}|\mathbf{X})$

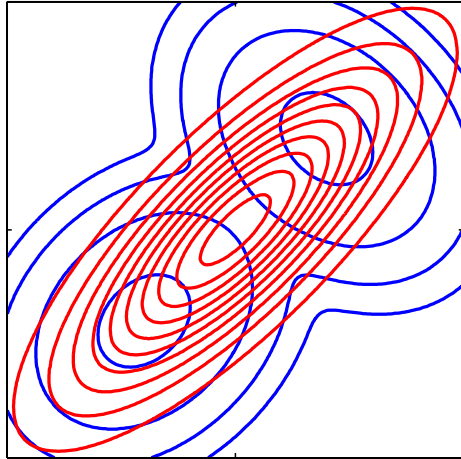
Proof:

$$\begin{aligned} & KL(p||q) + L(p) \\ &= \int p(Z|X) \{ \ln[p(Z|X)] - \ln[q(Z)] + \ln[q(Z)] \\ &\quad + \ln[p(X)] - \ln[p(Z|X)] \} dZ \\ &= \int p(Z|X) \{ \ln[p(X)] \} dZ = \ln[p(X)] \end{aligned}$$

□

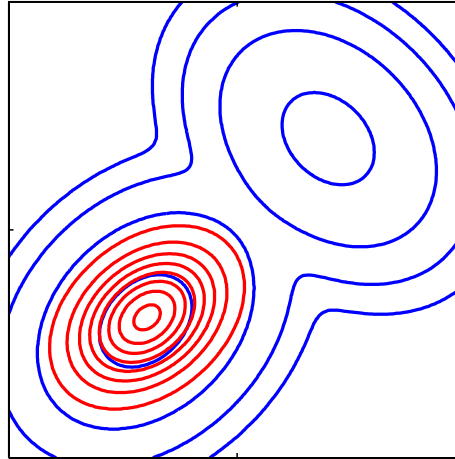
Forward or Backward KL-Divergence?

(Forward KL-divergence)
Minimizing $KL(p \parallel q)$

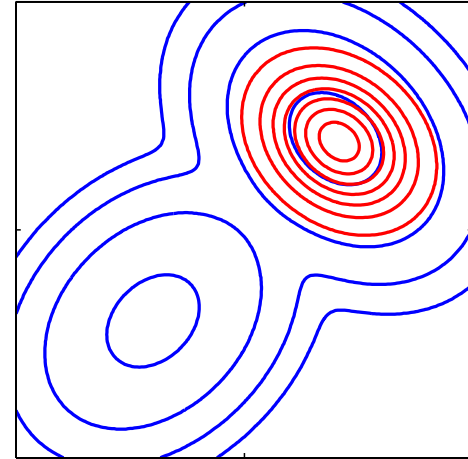


(a)

(Reverse KL-divergence)
Minimizing $KL(q \parallel p)$



(b)



(c)

- Backward KL-divergence is **statistically more sensible** because we saw that forward KL-divergence wrongly chose a mean in a region of low density.

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Example 1: The Univariate Gaussian

- **Given:** a data set $\mathcal{D} = \{x_1, \dots, x_N\}$ of **observed values** of X , which are assumed to be drawn independently from the Gaussian.
- **Goal:** infer the **posterior distribution** for the mean μ and precision τ (inverse of the covariance).
- The **likelihood function** is given by:

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

Example 1: The Univariate Gaussian

- The **conjugate prior distributions** (Gaussian-Gamma) for μ and τ given by:

$$\begin{aligned} p(\mu|\tau) &= \mathcal{N}(\mu|\mu_0, (\lambda_0\tau)^{-1}) \leftarrow (\mu_0, \lambda_0) : \text{hyperparameters} \\ p(\tau) &= \text{Gam}(\tau|a_0, b_0) \end{aligned}$$

- Gamma distribution:**

$$\text{Gam}(\tau|a_0, b_0) = \frac{1}{\Gamma(a_0)} b^{a_0} \tau^{a_0-1} \exp(-b_0\tau)$$

where a_0 and b_0 are the **hyperparameters**.

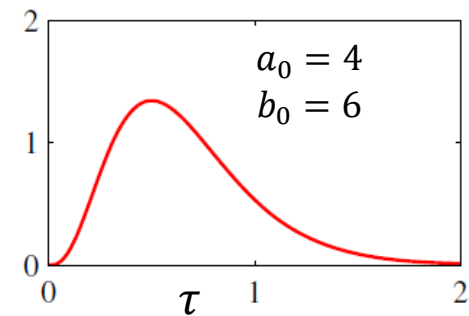
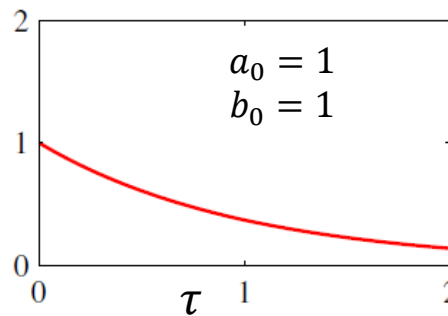
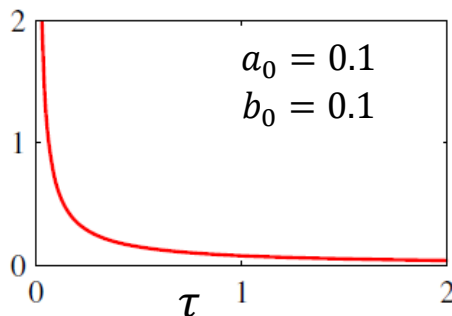


Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Example 1: The Univariate Gaussian

Note: For this simple problem, the posterior distribution can be found exactly, but we will do this with variational approach for tutorial purpose.

Example 1: The Univariate Gaussian

- We consider a **factorized variational approximation** to the posterior distribution given by:

$$q(\mu, \tau) = q_\mu(\mu)q_\tau(\tau), \quad \text{Recall: } q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i).$$

- The **optimum factors** $q_\mu(\mu)$ and $q_\tau(\tau)$ can be obtained from the general result:

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$$

Example 1: The Univariate Gaussian

- The **optimal solution** for $q_{\mu}(\mu)$ is given by:

$$\begin{aligned}\ln q_{\mu}^{\star}(\mu) &= \mathbb{E}_{\tau} [\ln p(\mathcal{D}|\mu, \tau) + \ln p(\mu|\tau)] + \text{const} \\ &= -\frac{\mathbb{E}[\tau]}{2} \left\{ \lambda_0(\mu - \mu_0)^2 + \sum_{n=1}^N (x_n - \mu)^2 \right\} + \text{const}.\end{aligned}$$

- Completing the square over μ , we see that $q_{\mu}(\mu)$ is a **Gaussian** $\mathcal{N}(\mu|\mu_N, \lambda_N^{-1})$ with means and precision given by:

$$\begin{aligned}\mu_N &= \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \\ \lambda_N &= (\lambda_0 + N)\mathbb{E}[\tau].\end{aligned}$$

$N \rightarrow \infty \Rightarrow \mu_N = \bar{x}$ and precision is infinite, i.e. maximum likelihood

Example 1: The Univariate Gaussian

- Similarly, the **optimal solution** for the factor $q_{\tau}(\tau)$ is given by:

$$\begin{aligned}\ln q_{\tau}^*(\tau) &= \mathbb{E}_{\mu} [\ln p(\mathcal{D}|\mu, \tau) + \ln p(\mu|\tau)] + \ln p(\tau) + \text{const} \\ &= (a_0 - 1) \ln \tau - b_0 \tau + \frac{N}{2} \ln \tau \\ &\quad - \frac{\tau}{2} \mathbb{E}_{\mu} \left[\sum_{n=1}^N (x_n - \mu)^2 + \lambda_0 (\mu - \mu_0)^2 \right] + \text{const}\end{aligned}$$

- Hence, $q_{\tau}(\tau)$ is a **gamma distribution** $\text{Gam}(\tau | a_N, b_N)$ with parameters:

$$\begin{aligned}a_N &= a_0 + \frac{N + 1}{2} \\ b_N &= b_0 + \frac{1}{2} \mathbb{E}_{\mu} \left[\sum_{n=1}^N (x_n - \mu)^2 + \lambda_0 (\mu - \mu_0)^2 \right].\end{aligned}$$

Example 1: The Univariate Gaussian

- It should be emphasized that we **did not assume** any specific functional forms for the optimal distributions $q_\mu(\mu)$ and $q_\tau(\tau)$.
- They **arose naturally** from the structure of the likelihood function and the corresponding conjugate priors!

Example 1: The Univariate Gaussian

Problem: solutions are coupled!

$$q_{\mu}(\mu) = \mathcal{N}(\mu | \mu_N, \lambda_N^{-1}), \text{ where } \begin{cases} \mu_N &= \frac{\lambda_0 \mu_0 + N \bar{x}}{\lambda_0 + N} \\ \lambda_N &= (\lambda_0 + N) \mathbb{E}[\tau]. \end{cases}$$

Depends on $q_{\tau}(\tau)$!

$$q_{\tau}(\tau) = \text{Gam}(\tau | a_N, b_N), \text{ where}$$

$$\begin{cases} a_N &= a_0 + \frac{N}{2} \\ b_N &= b_0 + \frac{1}{2} \mathbb{E}_{\mu} \left[\sum_{n=1}^N (x_n - \mu)^2 + \lambda_0 (\mu - \mu_0)^2 \right]. \end{cases}$$

Depends on $q_{\mu}(\mu)$!

Example 1: The Univariate Gaussian

Solution: an iterative approach

1. Make an **initial guess** for $\mathbb{E}[\tau]$ and use this to re-compute $q_{\mu}(\mu)$.
2. Use the revised $q_{\mu}(\mu)$ to **extracted the moments $\mathbb{E}[\mu]$ and $\mathbb{E}[\mu^2]$** , and use these to re-compute $q_{\tau}(\tau)$.
3. Use the revised $q_{\tau}(\tau)$ to **extract the moment $\mathbb{E}[\tau]$** and use this to re-compute $q_{\mu}(\mu)$.
4. Repeat until convergence.

Example 1: The Univariate Gaussian

Contours of the true posterior distribution $p(\mu, \tau | \mathcal{D})$ are shown in green

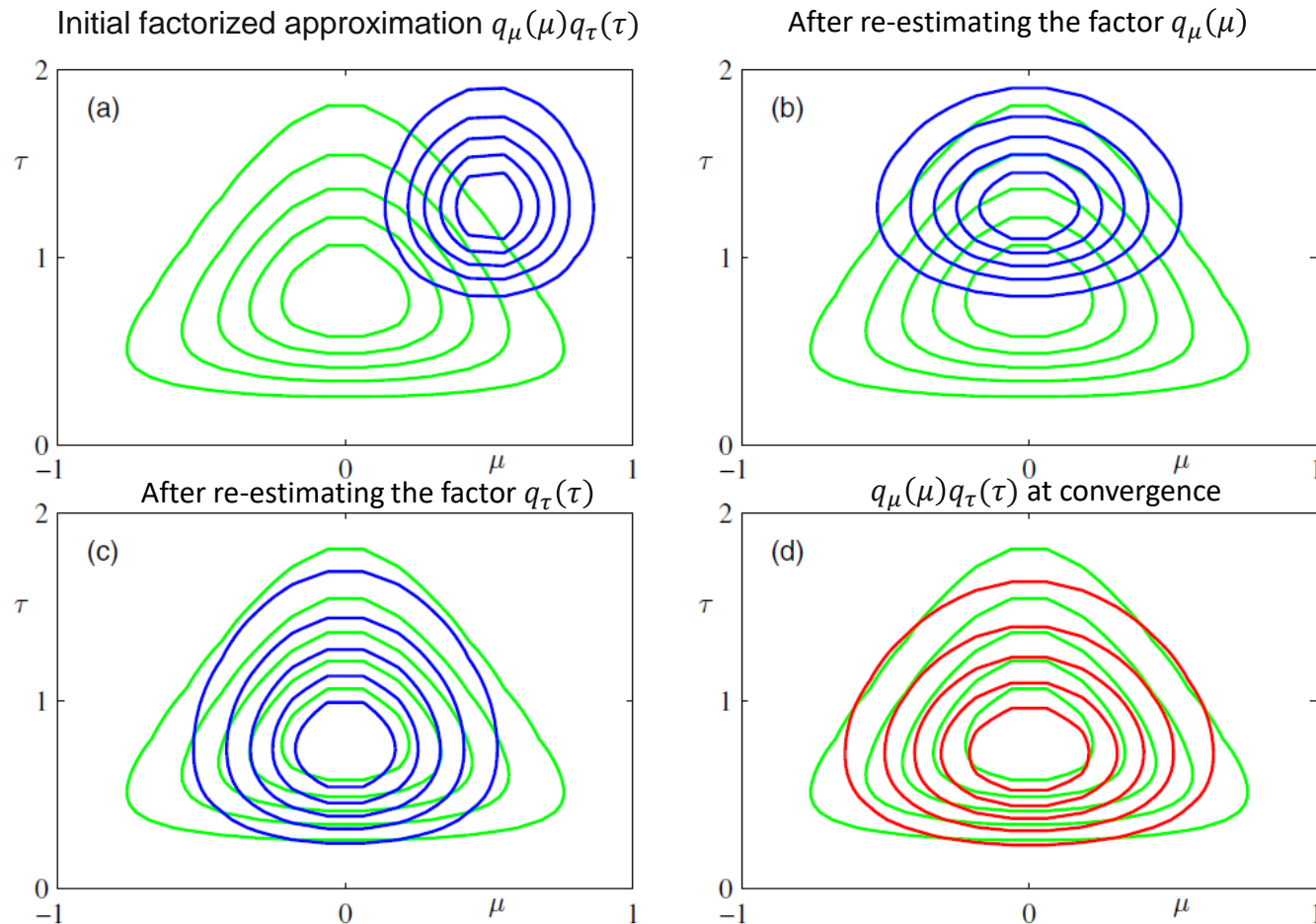


Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Example 2: Mixture of Gaussian

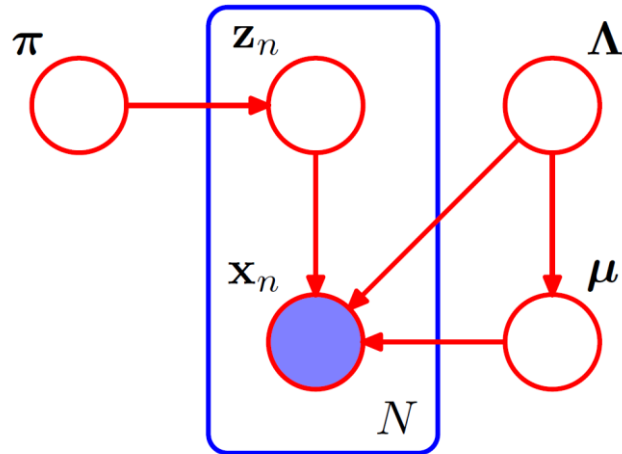


Plate denotes a set of N i.i.d. observations
 μ denotes $\{\mu_k\}$ and Λ denotes $\{\Lambda_k\}$.

- Let's look at a slightly more complicated example of applying variational inference on **mixture of Gaussian**.

Example 2: Mixture of Gaussian

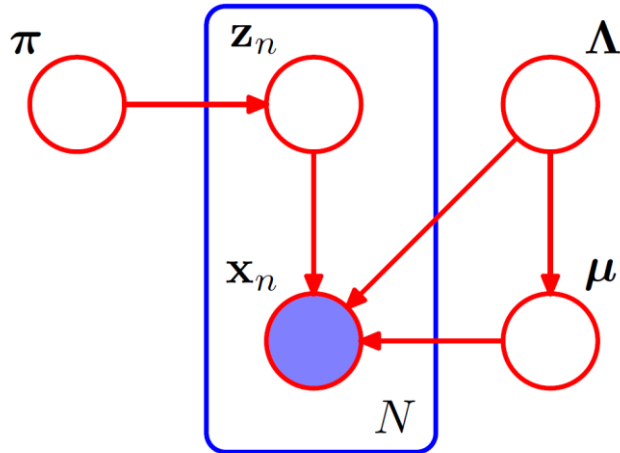


Plate denotes a set of N i.i.d. observations
 μ denotes $\{\mu_k\}$ and Λ denotes $\{\Lambda_k\}$.

- For each **observation** X_n , we have a corresponding **latent variable** Z_n comprising a **1-of- K binary vector** with elements Z_{nk} for $k = 1, \dots, K$.
- We denote the **observed data set** by $X = \{X_1, \dots, X_N\}$, and similarly we denote the **latent variables set** by $Z = \{Z_1, \dots, Z_N\}$.

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop


Example 2: Mixture of Gaussian

- **Conditional distribution of \mathbf{Z}** , given the mixing coefficients π :

$$p(\mathbf{Z}|\pi) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{z_{nk}}. \quad (\text{Categorical distribution})$$

- **Dirichlet distribution** conjugate prior over the mixing coefficients π :

Normalizing constant (see Lecture 1)



$$p(\pi) = \text{Dir}(\pi|\alpha_0) = C(\alpha_0) \prod_{k=1}^K \pi_k^{\alpha_0 - 1}$$

- Where we chose the **same hyperparameter α_0** for each component.

Example 2: Mixture of Gaussian

- Conditional distribution of **observed data vectors**, given latent variables and component parameters:

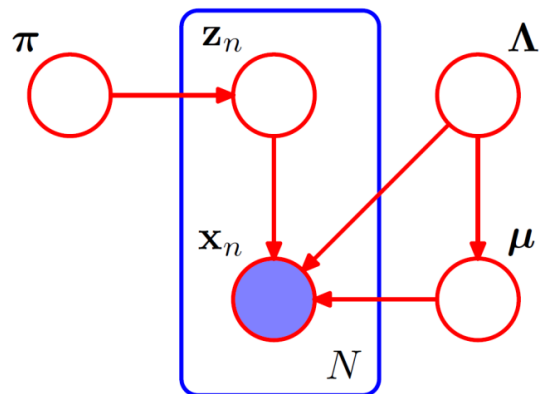
$$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^N \prod_{k=1}^K \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$$

- Independent **Gaussian-Wishart prior** governing the mean and precision of each Gaussian component:

$$\begin{aligned} p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) &= p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}) \\ &= \prod_{k=1}^K \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_0, \nu_0) \end{aligned}$$

- Where $\mathbf{m}_0, \mathbf{W}_0, \nu_0$ are the **hyperparameters**.

Example 2: Mixture of Gaussian



- Joint distribution:

$$p(\mathbf{X}, \mathbf{Z}, \pi, \mu, \Lambda) = p(\mathbf{X}|\mathbf{Z}, \mu, \Lambda)p(\mathbf{Z}|\pi)p(\pi)p(\mu|\Lambda)p(\Lambda)$$

- Consider a **variational distribution** which factorizes between the latent variables and parameters:

$$q(\mathbf{Z}, \pi, \mu, \Lambda) = q(\mathbf{Z})q(\pi, \mu, \Lambda).$$

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Example 2: Mixture of Gaussian

- The log of the **optimized factor for $q(\mathbf{z})$** is given by:

$$\ln q^*(\mathbf{Z}) = \mathbb{E}_{\pi, \mu, \Lambda} [\ln p(\mathbf{X}, \mathbf{Z}, \pi, \mu, \Lambda)] + \text{const.}$$

- Substituting the joint distribution and absorbing all terms that do not depend on Z into the additive constant, we get:

$$\ln q^*(\mathbf{Z}) = \mathbb{E}_{\pi} [\ln p(\mathbf{Z}|\pi)] + \mathbb{E}_{\mu, \Lambda} [\ln p(\mathbf{X}|\mathbf{Z}, \mu, \Lambda)] + \text{const.}$$

Example 2: Mixture of Gaussian

- Substituting the two conditional distributions, and again absorbing all terms independent of Z into the additive constant, we have:

$$\ln q^*(\mathbf{Z}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \ln \rho_{nk} + \text{const}$$

where we have defined

Dimensionality of data variable \mathbf{x}

$$\begin{aligned} \ln \rho_{nk} &= \mathbb{E}_{\pi_k}[\ln \pi_k] + \frac{1}{2} \mathbb{E}_{\Lambda_k}[\ln |\Lambda_k|] - \frac{D}{2} \ln(2\pi) \\ &\quad - \frac{1}{2} \mathbb{E}_{\mu_k, \Lambda_k} [(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \Lambda_k (\mathbf{x}_n - \boldsymbol{\mu}_k)] \end{aligned}$$

Example 2: Mixture of Gaussian

- Taking the exponential of both sides, we obtain:

$$q^*(\mathbf{Z}) \propto \prod_{n=1}^N \prod_{k=1}^K \rho_{nk}^{z_{nk}}.$$

- **Normalizing this distribution**, and noting that $z_{nk} \in \{0,1\}$ and sum to 1 over all values of k , we obtain:

$$q^*(\mathbf{Z}) = \prod_{n=1}^N \prod_{k=1}^K r_{nk}^{z_{nk}} \quad \leftarrow \text{Same functional form as } p(\mathbf{z}|\pi)!$$

where

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^K \rho_{nj}}.$$

Note that quantities r_{nk} are playing the role of responsibilities (i.e. this is similar to the E-Step in EM)!

Example 2: Mixture of Gaussian

- Let us **define three statistics** of the observed data set evaluated with respect to the responsibilities:

$$\begin{aligned}N_k &= \sum_{n=1}^N r_{nk} \\ \bar{\mathbf{x}}_k &= \frac{1}{N_k} \sum_{n=1}^N r_{nk} \mathbf{x}_n \\ \mathbf{S}_k &= \frac{1}{N_k} \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^T.\end{aligned}$$

- These are analogous to quantities evaluated in the **maximum likelihood** EM algorithm for the Gaussian mixture model.

Example 2: Mixture of Gaussian

- The log of the **optimized factor for $q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$** is given by:

$$\ln q^*(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \boxed{\ln p(\boldsymbol{\pi})} + \boxed{\sum_{k=1}^K \ln p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)} + \boxed{\mathbb{E}_{\mathbf{Z}} [\ln p(\mathbf{Z}|\boldsymbol{\pi})]}$$

Depends only on $\boldsymbol{\pi}$

Depends only on $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$

$$+ \boxed{\sum_{k=1}^K \sum_{n=1}^N \mathbb{E}[z_{nk}] \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})} + \text{const.}$$



Further factorizes into

$$q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\boldsymbol{\pi}) \prod_{k=1}^K q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$$

Example 2: Mixture of Gaussian

$\ln q^*(\boldsymbol{\pi})$ is given by:

$$\ln q^*(\boldsymbol{\pi}) = (\alpha_0 - 1) \sum_{k=1}^K \ln \pi_k + \sum_{k=1}^K \sum_{n=1}^N r_{nk} \ln \pi_k + \text{const}$$

$$\begin{aligned} \mathbb{E}[z_{nk}] &= \sum_Z z_{nk} q(Z) \\ &= \sum_Z z_{nk} \prod_n \prod_k r_{nk}^{z_{nk}} \\ &= r_{nk} \end{aligned}$$

Taking exponential of both sides, we get:

$$q^*(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi} | \boldsymbol{\alpha}) \quad \text{Dirichlet distribution!}$$

where $\boldsymbol{\alpha}$ has components α_k given by:

$$\alpha_k = \alpha_0 + N_k.$$

Example 2: Mixture of Gaussian

And $\ln q^*(\mu, \Lambda)$ is given by:

$$\ln q^*(\mu, \Lambda) = \sum_{k=1}^K \underbrace{\ln p(\mu_k, \Lambda_k)}_{\mathcal{N}(\mu_k | \mathbf{m}_0, (\beta_0 \Lambda_k)^{-1})} + \sum_{k=1}^K \sum_{n=1}^N \boxed{\mathbb{E}[z_{nk}]} \ln \mathcal{N}(\mathbf{x}_n | \mu_k, \Lambda_k^{-1}) + \text{const.}$$

r_{nk} : responsibility defined earlier

Proof:

$$\begin{aligned} \mathbb{E}[z_{nk}] &= \sum_{\mathbf{z}} q(\mathbf{z}) z_{nk} \quad , \quad \text{where } q^*(\mathbf{Z}) = \prod_{n=1}^N \prod_{k=1}^K r_{nk}^{z_{nk}} \\ &= \sum_{\mathbf{z}} z_{nk} \prod_n \prod_k r_{nk}^{z_{nk}} \\ &= r_{nk} \end{aligned}$$

Example 2: Mixture of Gaussian

Taking exponent on both sides, we get $q^*(\mu_k, \Lambda_k) = q^*(\mu_k|\Lambda_k)q^*(\Lambda_k)$, where:

$$q^*(\mu_k, \Lambda_k) = \underbrace{\mathcal{N}(\mu_k | \mathbf{m}_k, (\beta_k \Lambda_k)^{-1})}_{q^*(\mu_k|\Lambda_k)} \underbrace{\mathcal{W}(\Lambda_k | \mathbf{W}_k, \nu_k)}_{q^*(\Lambda_k)} \quad \text{Gaussian-Wishart distribution!}$$

with:

$$\begin{aligned}\beta_k &= \beta_0 + N_k \\ \mathbf{m}_k &= \frac{1}{\beta_k} (\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k) \\ \mathbf{W}_k^{-1} &= \mathbf{W}_0^{-1} + N_k \boxed{\mathbf{S}_k} + \frac{\beta_0 N_k}{\beta_0 + N_k} (\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^T \\ \nu_k &= \nu_0 + N_k.\end{aligned}$$

Similar to M-step of the EM algorithm for mixture of Gaussians!

Dependent on responsibility $\mathbb{E}[z_{nk}]$

Example 2: Mixture of Gaussian

- As before, the **solutions are coupled**; we will use the **iterative approach** (similar to EM iterations).

$$\ln q^*(\mathbf{Z}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \ln \rho_{nk} + \text{const}$$

where

$$\ln \rho_{nk} = \boxed{\mathbb{E}[\ln \pi_k]} + \frac{1}{2} \boxed{\mathbb{E}[\ln |\mathbf{\Lambda}_k|]} - \frac{D}{2} \ln(2\pi) - \frac{1}{2} \boxed{\mathbb{E}_{\mu_k, \mathbf{\Lambda}_k} [(\mathbf{x}_n - \mu_k)^T \mathbf{\Lambda}_k (\mathbf{x}_n - \mu_k)]}$$

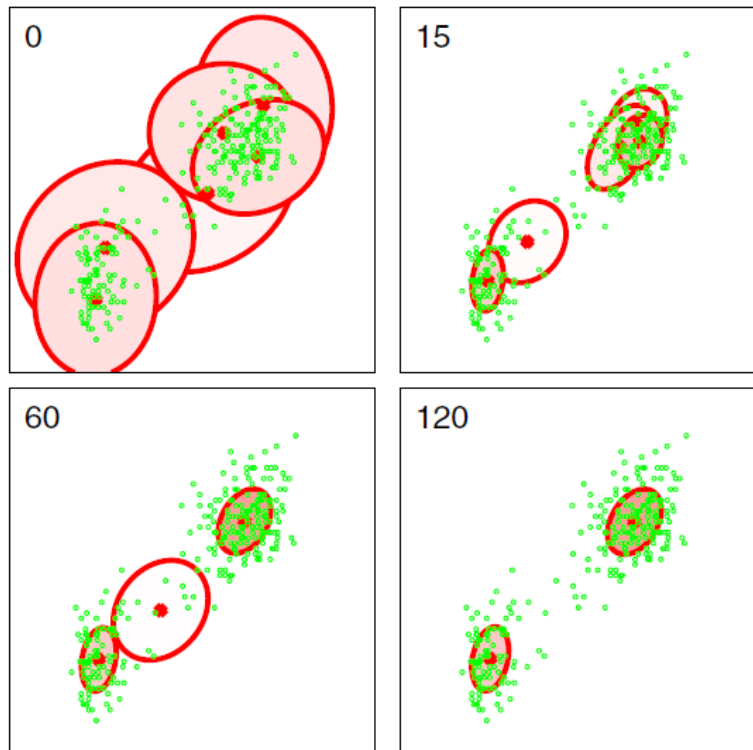
Depends on $q(\pi, \mu, \mathbf{\Lambda})$

$$\begin{aligned} \ln q^*(\pi, \mu, \mathbf{\Lambda}) &= \ln p(\pi) + \sum_{k=1}^K \ln p(\mu_k, \mathbf{\Lambda}_k) + \boxed{\mathbb{E}_{\mathbf{Z}} [\ln p(\mathbf{Z}|\pi)]} \\ &\quad + \sum_{k=1}^K \sum_{n=1}^N \boxed{\mathbb{E}[z_{nk}]} \ln \mathcal{N}(\mathbf{x}_n | \mu_k, \mathbf{\Lambda}_k^{-1}) + \text{const.} \end{aligned}$$

Depends on $q(\mathbf{Z})$

Example 2: Mixture of Gaussian

- Initialized with $K = 6$, but eventually **converged to two unique clusters**.
- All other unused components have the respective $r_{nk} \approx 0$.

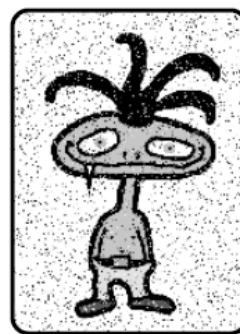
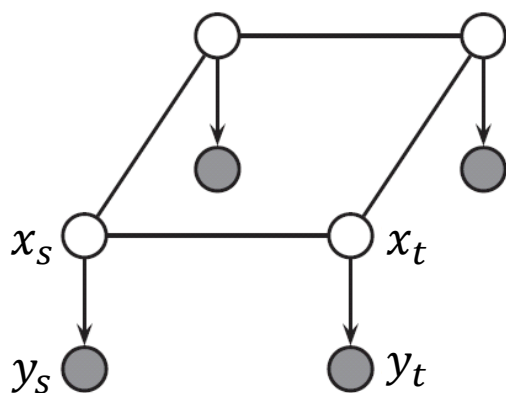


Model selection!

Image Source: "Pattern Recognition and Machine Learning", Christopher Bishop

Example 3: Ising Model

Image denoising problem: Get the “clean” image!



Observed Variables
 $y \in \{-1, +1\}$



Latent Variables
 $x \in \{-1, +1\}$

The **joint distribution** is given by:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$$

Image Source: “Computer Vision: Models, Learning, and Inference”, Simon Prince

Example 3: Ising Model

- The **prior** has the form:

$$p(\mathbf{x}) = \frac{1}{Z_0} \exp(-E_0(\mathbf{x}))$$
$$E_0(\mathbf{x}) = - \sum_{i=1}^D \sum_{j \in \text{nbr}_i} W_{ij} x_i x_j$$

Ising prior normally set to 1

“**Pairwise smoothness**” term, i.e. we want all neighbors to take same value

- The **likelihood** has the form:

$$p(\mathbf{y}|\mathbf{x}) = \prod_i p(\mathbf{y}_i|x_i) = \exp\left(\sum_i L_i(x_i)\right)$$

High energy $-L(x_i)$ when $x_i \neq y_i$, and vice-versa.

“**Unary potential**” term, i.e. we want latent variable to agree with the observation.

Example 3: Ising Model

- Therefore, the **joint probability** has the form:

$$\begin{aligned} p(x, y) &= p(x)p(y|x) = \frac{1}{Z_0} \exp(-E_0(x) + \sum_i L_i(x_i)) \\ &= \frac{1}{Z_0} \exp(-E(x)) , \end{aligned}$$

where

$$E(x) = \sum_i \left\{ -x_i \sum_{j \in nbr_i} W_{ij} x_j - L_i(x_i) \right\} .$$

$$\Rightarrow \ln p(x, y) = \sum_i \left\{ x_i \sum_{j \in nbr_i} W_{ij} x_j + L_i(x_i) \right\} + \ln Z_0$$

Example 3: Ising Model

- We will now approximate this by a **fully factored approximation**:

$$q(x) = \prod_i q(x_i)$$

- The log of the **optimized factor $q_i(x_i)$** is given by:

$$\log q_i(x_i) = \mathbb{E}_{x_{\setminus i}}[\log p(x, y)] + \text{const}$$

$$= \mathbb{E}_{x_{\setminus i}} \left[x_i \sum_{j \in \text{nbr}_i} W_{ij} x_j + L_i(x_i) \right] + \underbrace{\text{const}}$$

All terms that do not include x_i
go to const (including Z_0)

Example 3: Ising Model

- We get:

$$q_i(x_i) \propto \exp \left(x_i \sum_{j \in \text{nbr}_i} W_{ij} \mu_j + L_i(x_i) \right)$$

where $\mu_j = \sum_j x_j q(x_j)$ is the **mean value** of node j .

The results are **coupled** as before, do **iterative update** with an initialization of all μ_j !

Example 3: Ising Model

- We define m_i as the **mean field influence** on node i :

$$m_i = \sum_{j \in \text{nbr}_i} W_{ij} \mu_j$$

- And let:

$$L_i^+ \triangleq L_i(+1) \quad \text{and} \quad L_i^- \triangleq L_i(-1).$$

- The **approximate marginal posterior** is given by

$$q_i(x_i = 1) = \frac{e^{m_i + L_i^+}}{e^{m_i + L_i^+} + e^{-m_i + L_i^-}} = \frac{1}{1 + e^{-2m_i + L_i^- - L_i^+}} = \text{sigm}(2a_i)$$
$$a_i \triangleq m_i + 0.5(L_i^+ - L_i^-)$$

$$q_i(x_i = -1) = \text{sigm}(-2a_i)$$

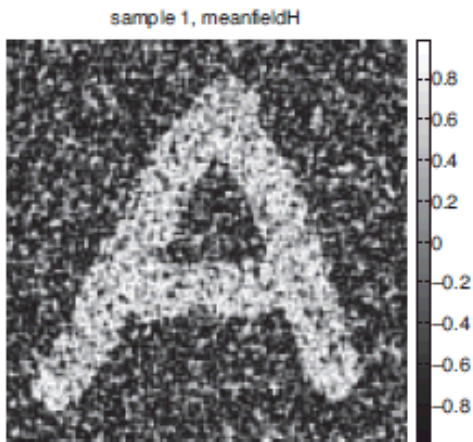
Example 3: Ising Model

- we can compute the **new mean** for site i :

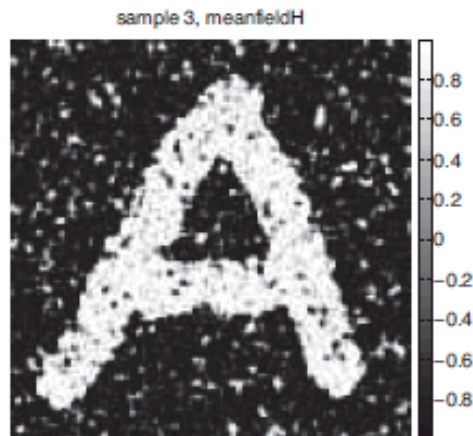
$$\begin{aligned}\mu_i &= \mathbb{E}_{q_i} [x_i] = q_i(x_i = +1) \cdot (+1) + q_i(x_i = -1) \cdot (-1) \\ &= \frac{1}{1 + e^{-2a_i}} - \frac{1}{1 + e^{2a_i}} = \frac{e^{a_i}}{e^{a_i} + e^{-a_i}} - \frac{e^{-a_i}}{e^{-a_i} + e^{a_i}} = \tanh(a_i) \\ &= \tanh \left(\sum_{j \in \text{nbr}_i} W_{ij} \mu_j + 0.5(L_i^+ - L_i^-) \right)\end{aligned}$$

Iterate until convergence!

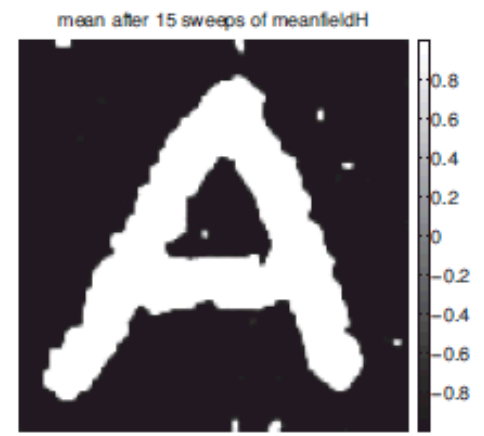
Example 3: Ising Model



(a)



(b)



(c)

Image source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Loopy Belief Propagation

- Loopy Belief Propagation is a simple algorithm for belief propagation in **graphs with cycles**.
- We will just look at the algorithm **without proof** of a variational point of view.
- Proof of the Loopy Belief Propagation algorithm from a **variational point of view** can be found in a 300 pages paper (Wainwright and Jordan 2008).

Loopy Belief Propagation

The **basic idea** is extremely simple: we apply the belief propagation algorithm to the graph, even if it has loops, i.e. even if it is **not a tree**.

Convergence is however **NOT Guaranteed!**

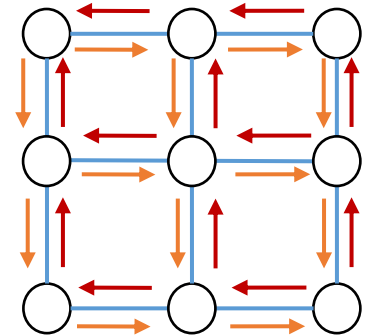
Loopy Belief Propagation

All messages and beliefs initialized to 1: $\begin{cases} m_{s \rightarrow t}(x_t) = 1 \\ \text{bel}_s(x_s) = 1 \end{cases}$

Algorithm: Loopy belief propagation for a pairwise MRF

- 1 Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
 - 2 Initialize messages $m_{s \rightarrow t}(x_t) = 1$ for all edges $s - t$;
 - 3 Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes s ;
 - 4 **repeat**
 - 5 Send message on each edge

$$m_{s \rightarrow t}(x_t) = \sum_{x_s} \left(\psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \rightarrow s}(x_s) \right);$$
 - 6 Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \rightarrow s}(x_s)$;
 - 7 **until** *beliefs don't change significantly*;
 - 8 Return marginal beliefs $\text{bel}_s(x_s)$;
-



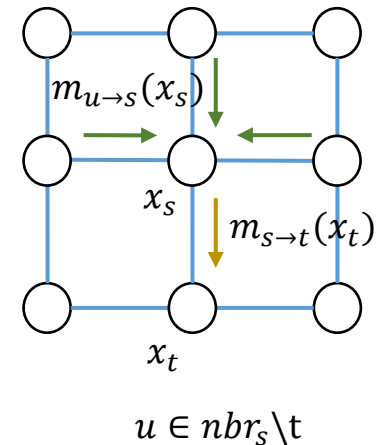
Source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Loopy Belief Propagation

Algorithm: Loopy belief propagation for a pairwise MRF

- 1 Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
- 2 Initialize messages $m_{s \rightarrow t}(x_t) = 1$ for all edges $s - t$;
- 3 Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes s ;
- 4 **repeat**
- 5 Send message on each edge
- 6 Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \rightarrow s}(x_s)$;
- 7 **until** *beliefs don't change significantly*;
- 8 Return marginal beliefs $\text{bel}_s(x_s)$;

Message passing



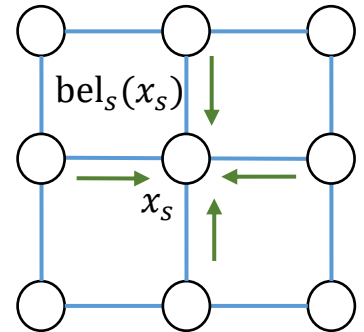
Source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Loopy Belief Propagation

Algorithm: Loopy belief propagation for a pairwise MRF

- 1 Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
 - 2 Initialize messages $m_{s \rightarrow t}(x_t) = 1$ for all edges $s - t$;
 - 3 Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes s ;
 - 4 **repeat**
 - 5 Send message on each edge **Belief is updated at every iteration!**

$$m_{s \rightarrow t}(x_t) = \sum_{x_s} \left(\psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \rightarrow s}(x_s) \right);$$
 - 6 Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \rightarrow s}(x_s);$
 - 7 **until** *beliefs don't change significantly*;
 - 8 Return marginal beliefs $\text{bel}_s(x_s)$;
-



Source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Loopy Belief Propagation

Algorithm: Loopy belief propagation for a pairwise MRF

- 1 Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
 - 2 Initialize messages $m_{s \rightarrow t}(x_t) = 1$ for all edges $s - t$;
 - 3 Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes s ;
 - 4 **repeat**
 - 5

Send message on each edge

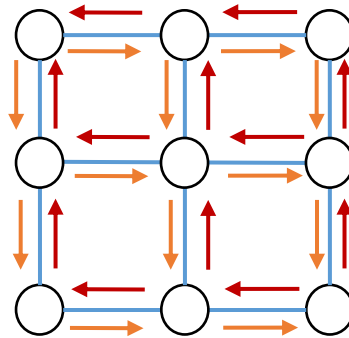
Can be asynchronous or synchronous!

$$m_{s \rightarrow t}(x_t) = \sum_{x_s} \left(\psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \rightarrow s}(x_s) \right);$$
 - 6 Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \rightarrow s}(x_s)$;
 - 7 **until** *beliefs don't change significantly*;
 - 8 Return marginal beliefs $\text{bel}_s(x_s)$;
-

Source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Loopy Belief Propagation: Synchronous

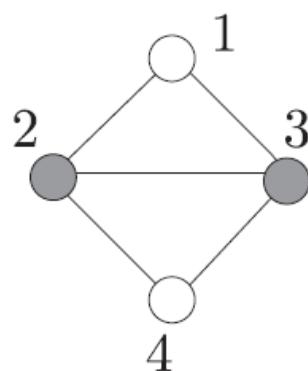
All messages are passed concurrently!



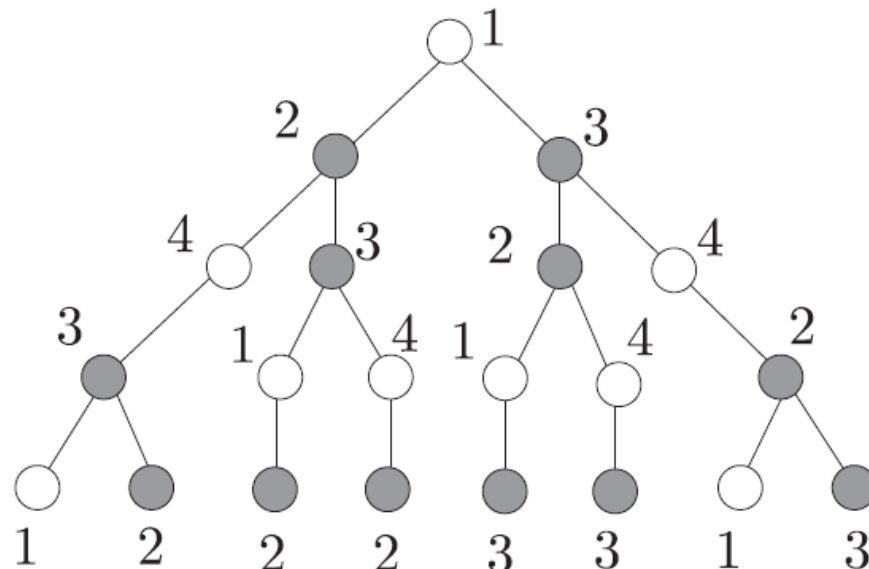
$$m_{s \rightarrow t}^{(k)}(x_t) = \sum_{x_s} \left(\psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \rightarrow s}^{(k-1)}(x_s) \right);$$

- All **current messages** $m_{s \rightarrow t}^{(k)}$ are updated with messages from the **previous time step** $m_{u \rightarrow s}^{(k-1)}$

Loopy Belief Propagation: Asynchronous



(a)



(b)

- Messages are passed asynchronously according to the **herarchical order given by the tree**.
- Notice that a node would **never pass a message** to its parent in the tree.

Image Source: "Machine Learning: A probabilistic Perspective", Kevin Murphy

Summary

- We have looked at how to:
 1. Explain the concept of variational approach using **Lower-Bound** of maximum likelihood and **KL-divergence**.
 2. Use **variational approach** to do inference on graphical models containing both hidden variables and unknown parameters.
 3. Use **Loopy Belief Propagation** to do message passing.