Probabilistic (Graphical) Models

and inference

Oliver Obst · Autumn 2024

WESTERN SYDNEY
UNIVERSITY

Probabilistic (Graphical) Models and Inference

(PGM: Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman. MIT Press)
(PMLI: Probabilistic Machine Learning: An introduction by Kevin Murphy. MIT Press)

Week	Lecture	Required reading	Assessment
1 Monday, 4 March 2024	Introduction, Probability Theory	PGM Chapter 2, PMLI Chapter 6.1	
2 Monday, 11 March 2024	Directed and undirected networks introduction	PGM Chapter 3 & 4	Quiz 1
3 Monday, 18 March 2024	Variable elimination	PGM Chapter 9	
4 Monday, 25 March 2024	Belief propagation	PGM Chapter 10/11	Quiz 2
5 Monday, 1 April 2024	public holiday		5 April 2024: census date
6 Monday, 8 April 2024	Message passing / Graph neural networks	https://distill.pub/2021/gnn-intro/	
7 Monday, 15 April 2024	Sampling	PGM Chapter 12	Quiz 3
8 Monday, 22 April 2024	Mid-term break		
9 Monday, 29 April 2024	Variational inference	https://leimao.github.io/article/Introduction-to-Variational-Inference/	Intra-session exam
10 Monday, 6 May 2024	Autoregressive models	https://sites.google.com/view/berkeley-cs294-158-sp20/home	Quiz 4
11 Monday, 13 May 2024	Variational Auto-Encoders		
12 Monday, 20 May 2024	GANs		Quiz 5
13 Monday, 27 May 2024	Energy-based models		
14 Monday, 3 June 2024	Evaluating generative models		Quiz 6
Monday, 17 June 2024			Project due

Bayesian Inference recap

Slides based on the article “Introduction to Variational Inference”, Lei Mao, 2019

Bayesian Inference

recap

We have a set of observed random variables:

$$\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$$

and a set of latent random variables:

$$\mathbf{z} = \{z_1, z_2, \ldots, z_m\}$$

We are interesting in computing the posterior $p(\mathbf{z} \mid \mathbf{x})$.

(Sometimes also written $p(\mathbf{z} \mid \mathbf{x}, \alpha)$, where α are parameters).

Bayes' theorem

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

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

$p(\mathbf{x})$ is the marginal density (the evidence).

Bayes' theorem

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

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

$p(\mathbf{x})$ is the marginal density (the evidence).

$p(\mathbf{x})$ can be computed by "summing out" z, or, for continuous values:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

VI motivation

What is VI?

In variational inference,

we try to *approximate* the conditional density of latent variables given observed variables.

So far, we introduced exact methods (like variable elimination), and
also approximate methods (like sampling).

What is VI?

In variational inference,

we try to *approximate* the conditional density of latent variables given observed variables.

So far, we introduced exact methods (like variable elimination), and
also approximate methods (like sampling).

Exact methods may not be tractable for some problems, and while sampling is a possibility in these cases, it also has its drawbacks.

VI approximates by solving an optimisation problem.

Why Bayesian Inference can be intractable

Example (step 1: generating a dataset)

Let's say we have a mixture of K Gaussians, with means $\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}$ and $\sigma^2 = 1$.

The means are independently drawn from a common prior, $\mu_k \sim \mathcal{N}(0, \sigma^2)$.

$\rightarrow$ μ_k is random variable, σ^2 is a hyper-parameter, μ is a (column) vector.

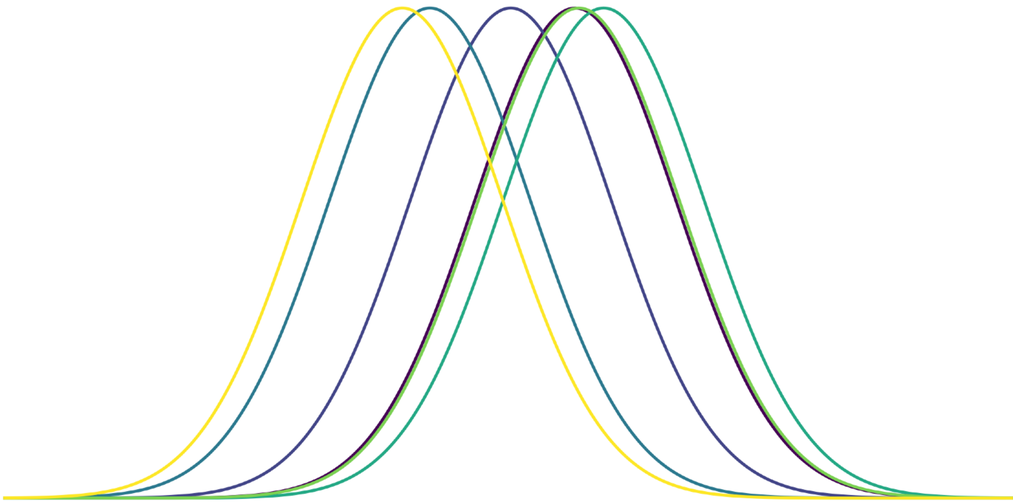
Why Bayesian Inference can be intractable

Example (step 1: generating a dataset)

Let's say we have a mixture of K Gaussians, with means $\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}$ and $\sigma^2 = 1$.

The means are independently drawn from a common prior, $\mu_k \sim \mathcal{N}(0, \sigma^2)$.

$\rightarrow$ μ_k is random variable, σ^2 is a hyper-parameter, μ is a (column) vector.

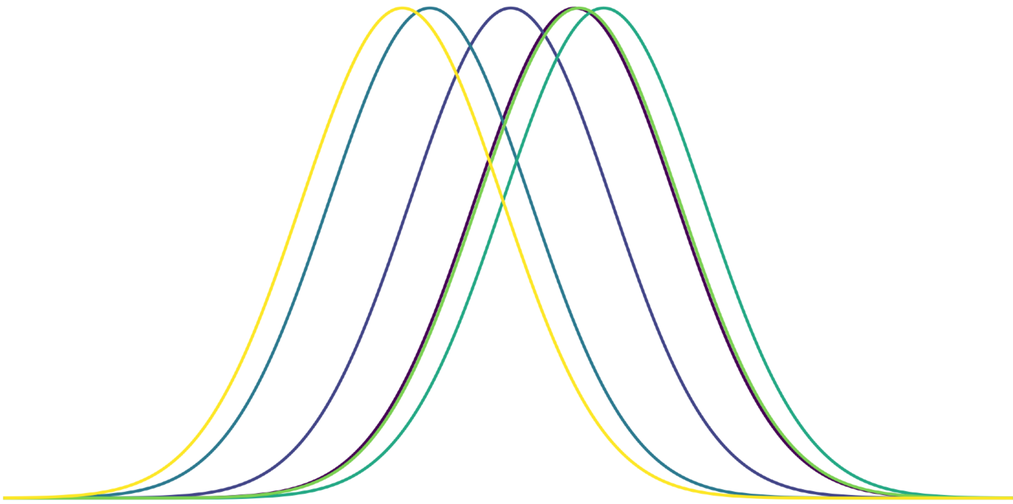


Why Bayesian Inference can be intractable

Example (step 1: generating a dataset)

Now we generate n observations: $\{x_1, x_2, \ldots, x_n\}$. For each x_i :

- we choose the assignment c_i from categorical $\{1,2,\ldots,K\}$.
- Let's say $p(c_i) = 1/K$ for all samples. c_i is represented using 1-hot encoding:

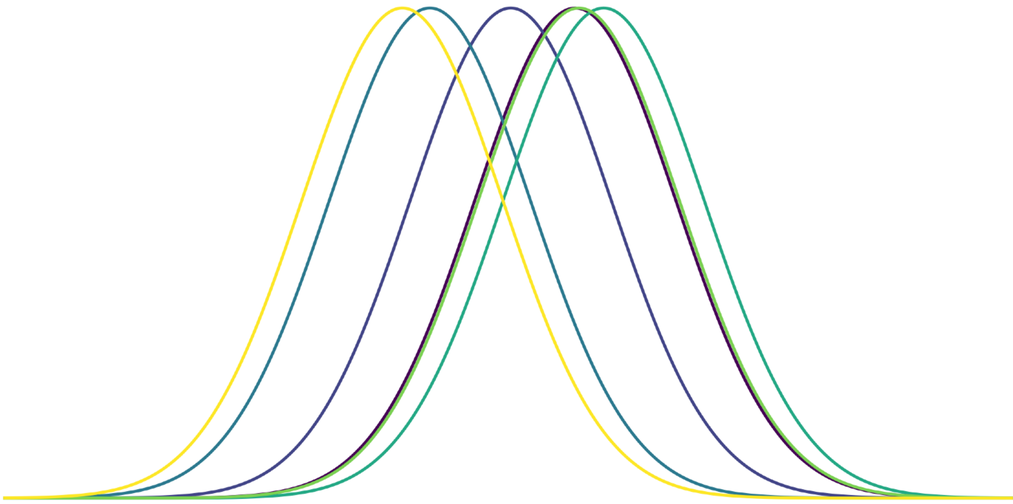


Why Bayesian Inference can be intractable

Example (step 1: generating a dataset)

Now we generate n observations: $\{x_1, x_2, \ldots, x_n\}$. For each x_i :

- we choose the assignment c_i from categorical $\{1,2,\ldots,K\}$.
- Let's say $p(c_i) = 1/K$ for all samples. c_i is represented using 1-hot encoding:
  - If $K = 5$, Gaussian mixture #3 was sampled, for observation 2: $c_2 = \{0,0,1,0,0\}$.



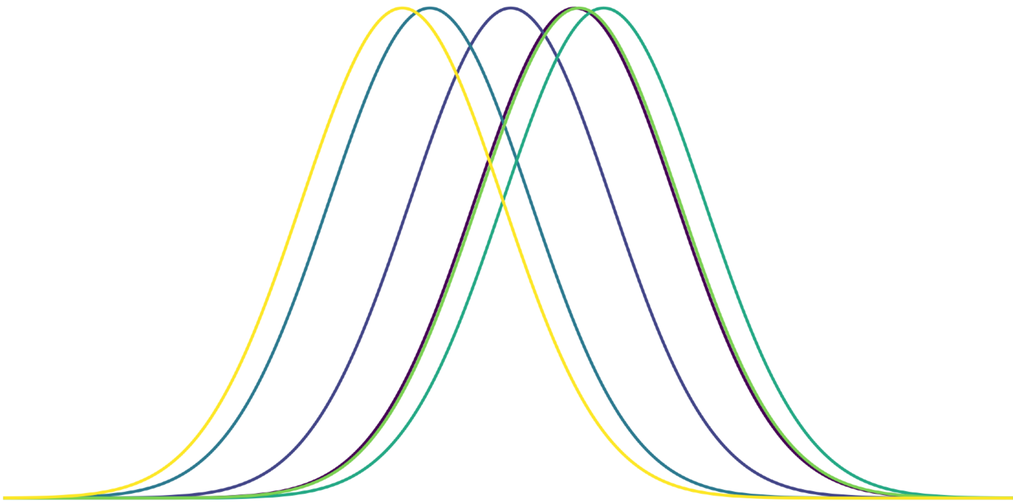
Why Bayesian Inference can be intractable

Example (step 1: generating a dataset)

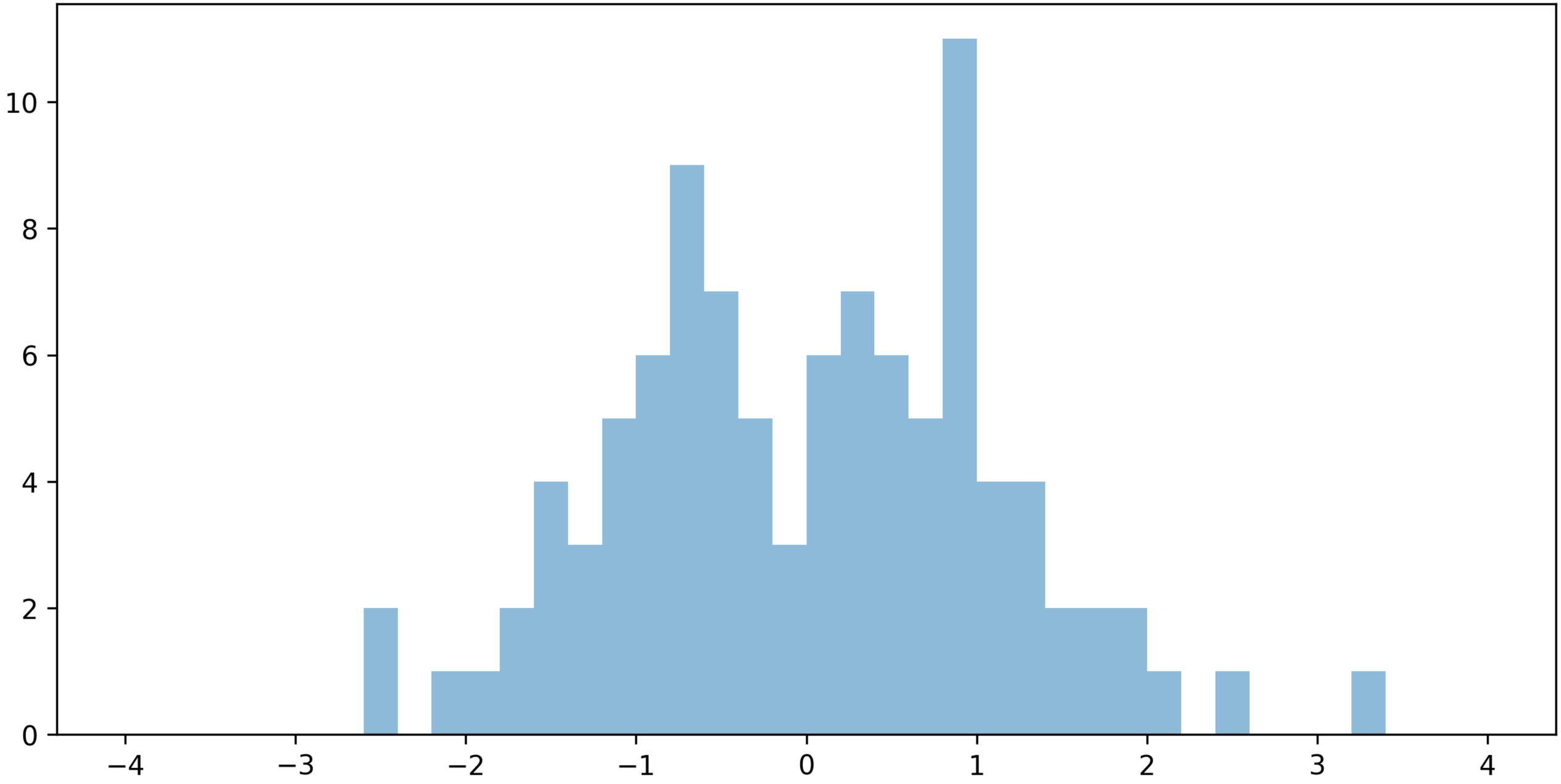
Now we generate n observations: $\{x_1, x_2, \ldots, x_n\}$. For each x_i :

- we choose the assignment c_i from categorical $\{1,2,\ldots,K\}$.
- Let's say $p(c_i) = 1/K$ for all samples. c_i is represented using 1-hot encoding:
- If $K = 5$, Gaussian mixture #3 was sampled, for observation 2: $c_2 = \{0,0,1,0,0\}$.

The mean of the sampled Gaussian can be written as the dot product of c_i and μ: $x_i \mid c_i, \mu \sim \mathcal{N}(c_i^\top \mu, 1)$



```
K = 5
n = 100

# Generate μ_k from a Gaussian distribution
mu_k = np.random.normal(0, 1, K)

# Initialize the data matrix D and the one-hot encoded assignments matrix C
D = np.zeros((n, 1))
C = np.zeros((n, K))

# Generate n observations
for i in range(n):
    # Choose assignment c_i from categorical 1,2,...,K with equal probability
    c_i = np.random.choice(K)

    # Represent c_i using one-hot encoding
    one_hot_c_i = np.zeros(K)
    one_hot_c_i[c_i] = 1

    # Store the one-hot encoded assignment in the matrix C
    C[i, :] = one_hot_c_i

    # Generate x_i from the Gaussian distribution with mean μ_k and variance 1
    x_i = np.random.normal(mu_k[c_i], 1)

    # Store the generated observation in the data matrix D
    D[i] = x_i

print("Data matrix D:\n", D)
print("\nOne-hot encoded assignments matrix C:\n", C)
```


10
8
6
4
2
0
−4
−3
−2
−1
0
1
2
3
4

Why Bayesian Inference can be intractable

Example: model (the prior)

We have this (assumed) model, containing prior information

- $\mu_k \sim \mathcal{N}(0, \sigma^2)$ $\quad k = 1,2,...,K$
- $c_i \sim \text{Categorical}(1/K, \ldots, 1/K)$ $\quad i = 1,2,...,n$
- $x_i \,|\, c_i, \mu \sim \mathcal{N}(c_i^\top \mu, 1)$ $\quad i = 1,2,...,n$

To calculate the joint probability density of $\left(\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}, \mathbf{c} = \{c_1, c_2, \ldots, c_n\}\right)$

Note: $\mathbf{c}$ has K^n possible values. Observation $\left(\mathbf{x} = \{x_1, x_2, \ldots, x_n\}\right)$

Why Bayesian Inference can be intractable

Example: model (the prior)

To calculate the joint probability density of $\left(\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}, \mathbf{c} = \{c_1, c_2, \ldots, c_n\}\right)$

Note: $\mathbf{c}$ has K^n possible values.

Observation $\left(\mathbf{x} = \{x_1, x_2, \ldots, x_n\}\right)$

Why Bayesian Inference can be intractable

Example: model (the prior)

To calculate the joint probability density of $\left(\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}, \mathbf{c} = \{c_1, c_2, \ldots, c_n\}\right)$

Note: $\mathbf{c}$ has K^n possible values.

Observation $\left(\mathbf{x} = \{x_1, x_2, \ldots, x_n\}\right)$

So $p(\mu, \mathbf{c}, \mathbf{x}) = p(\mu) \prod_{i=1}^{n} p(c_i)\, p(x_i \mid c_i, \mu).$

$\mu, \mathbf{c}, \mathbf{x}$ are all random variables, the latent variables here are $\mathbf{z} = \{\mu, \mathbf{c}\}$.

Why Bayesian Inference can be intractable

Example: model (the prior)

To calculate the joint probability density of $\left(\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}, \mathbf{c} = \{c_1, c_2, \ldots, c_n\}\right)$

Note: $\mathbf{c}$ has K^n possible values.

Observation $\left(\mathbf{x} = \{x_1, x_2, \ldots, x_n\}\right)$

So $p(\mu, \mathbf{c}, \mathbf{x}) = p(\mu) \prod_{i=1}^{n} p(c_i)\, p(x_i \mid c_i, \mu).$

$\mu, \mathbf{c}, \mathbf{x}$ are all random variables, the latent variables here are $\mathbf{z} = \{\mu, \mathbf{c}\}$.

This is the model (the prior) we *assume* for our observations
– we usually don't know how the data was generated.

Why Bayesian Inference can be intractable

Example: model (the prior)

To calculate the joint probability density of $\left(\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}, \mathbf{c} = \{c_1, c_2, \ldots, c_n\}\right)$

Note: $\mathbf{c}$ has K^n possible values.

Observation $\left(\mathbf{x} = \{x_1, x_2, \ldots, x_n\}\right)$

So $p(\mu, \mathbf{c}, \mathbf{x}) = p(\mu) \prod_{i=1}^{n} p(c_i)\, p(x_i \mid c_i, \mu).$

$\mu, \mathbf{c}, \mathbf{x}$ are all random variables, the latent variables here are $\mathbf{z} = \{\mu, \mathbf{c}\}$.

This is the model (the prior) we *assume* for our observations – we usually don't know how the data was generated.

Our goal is to calculate $p(\mathbf{z} \mid \mathbf{x})$.

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z} \mid \mathbf{x})$.

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z} \mid \mathbf{x})$.

Using Bayes' theorem:

$$
\begin{aligned}
p(\mathbf{z} \mid \mathbf{x}) &= p(\mu, \mathbf{c} \mid \mathbf{x}) \\
&= \frac{p(\mathbf{x} \mid \mu, \mathbf{c})\, p(\mu, \mathbf{c}))}{p(\mathbf{x})} \\
&= \frac{p(\mu, \mathbf{c}, \mathbf{x})}{p(\mathbf{x})}
\end{aligned}
$$

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z} \mid \mathbf{x})$.

Using Bayes' theorem:

$$
\begin{aligned}
p(\mathbf{z} \mid \mathbf{x}) &= p(\mu, \mathbf{c} \mid \mathbf{x}) \\
&= \frac{p(\mathbf{x} \mid \mu, \mathbf{c})\, p(\mu, \mathbf{c}))}{p(\mathbf{x})} \\
&= \frac{p(\mu, \mathbf{c}, \mathbf{x})}{p(\mathbf{x})}
\end{aligned}
$$

$$
\begin{aligned}
p(\mathbf{x}) &= \int_{\mu} \int_{\mathbf{c}} p(\mu, \mathbf{c}, \mathbf{x})\, d\mathbf{c}\, d\mu \\
&= \ldots \\
&= \int_{\mu} p(\mu) \left[\prod_{i=1}^{n} \sum_{c_i} p(c_i)\, p(x_i \mid c_i, \mu) \right] d\mu
\end{aligned}
$$

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z}\,|\,\mathbf{x})$.

Using Bayes' theorem:

$$p(\mathbf{z}\,|\,\mathbf{x}) = p(\mu, \mathbf{c}\,|\,\mathbf{x}) = \frac{p(\mathbf{x}\,|\,\mu, \mathbf{c})\, p(\mu, \mathbf{c}))}{p(\mathbf{x})} = \frac{p(\mu, \mathbf{c}, \mathbf{x})}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \int_{\mu} \int_{\mathbf{c}} p(\mu, \mathbf{c}, \mathbf{x})\, d\mathbf{c}\, d\mu = \ldots = \int_{\mu} p(\mu) \left[\prod_{i=1}^{n} \sum_{c_i} p(c_i)\, p(x_i\,|\,c_i, \mu) \right] d\mu$$

Possible values of c_i is $\{0,1,\ldots,K\}$. If we expand

there will be K^n terms (independent integrals) to compute $p(\mathbf{x})$.

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z}\,|\,\mathbf{x})$.

Using Bayes' theorem:

$$p(\mathbf{z}\,|\,\mathbf{x}) = p(\mu, \mathbf{c}\,|\,\mathbf{x}) = \frac{p(\mathbf{x}\,|\,\mu, \mathbf{c})\,p(\mu, \mathbf{c}))}{p(\mathbf{x})} = \frac{p(\mu, \mathbf{c}, \mathbf{x})}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \int_{\mu}\int_{\mathbf{c}} p(\mu, \mathbf{c}, \mathbf{x})\, d\mathbf{c}\, d\mu = \ldots = \int_{\mu} p(\mu)\left[\prod_{i=1}^{n}\sum_{c_i} p(c_i)\,p(x_i\,|\,c_i, \mu)\right] d\mu$$

Possible values of c_i is $\{0,1,\ldots,K\}$. If we expand

there will be K^n terms (independent integrals) to compute $p(\mathbf{x})$.

The complexity of computing $p(\mathbf{x})$ is $O(K^n)$, which is intractable.

Why Bayesian Inference can be intractable

Example: model (the prior)

Our goal is to calculate $p(\mathbf{z}\,|\,\mathbf{x})$.

Using Bayes' theorem:

$$p(\mathbf{z}\,|\,\mathbf{x}) = p(\mu, \mathbf{c}\,|\,\mathbf{x}) = \frac{p(\mathbf{x}\,|\,\mu, \mathbf{c})\,p(\mu, \mathbf{c}))}{p(\mathbf{x})} = \frac{p(\mu, \mathbf{c}, \mathbf{x})}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \int_{\mu}\int_{\mathbf{c}} p(\mu, \mathbf{c}, \mathbf{x})\, d\mathbf{c}\, d\mu = \ldots = \int_{\mu} p(\mu)\left[\prod_{i=1}^{n}\sum_{c_i} p(c_i)\,p(x_i\,|\,c_i, \mu)\right] d\mu$$

Possible values of c_i is $\{0,1,\ldots,K\}$. If we expand

there will be K^n terms (independent integrals) to compute $p(\mathbf{x})$.

Computing the posterior $p(\mathbf{z}\,|\,\mathbf{x})$ is intractable in this case.

The complexity of computing $p(\mathbf{x})$ is $O(K^n)$, which is intractable.

Approximation by optimisation

VI tries to find the best distribution from a family of distributions.

These are parameterised by free “variational parameters”, to approximate $p(\mathbf{z} \mid \mathbf{x})$.

Approximation by optimisation

VI tries to find the best distribution from a family of distributions.

These are parameterised by free “variational parameters”, to approximate $p(\mathbf{z} \mid \mathbf{x})$.

The optimisation finds the best settings for these parameters, and

therefore we then have found the best distribution (from the family of distributions).

Approximation by optimisation

VI tries to find the best distribution from a family of distributions.

These are parameterised by free “variational parameters”, to approximate $p(\mathbf{z} \mid \mathbf{x})$.

The optimisation finds the best settings for these parameters, and

therefore we then have found the best distribution (from the family of distributions).

To optimise anything, we need to have a measure (for how good we are).

Optimisation goal

In VI, we specify a family of distributions $\mathscr{L}$ over latent random variables.

- Each $q(\mathbf{z}) \in \mathscr{L}$ is a candidate approximation to the posterior.
- But how good is (each) approximation?
- Difference between two distributions: $D_{\mathrm{KL}}(\,\cdot \| \cdot\,)$ – aka the KL divergence.

Optimisation goal

In VI, we specify a family of distributions $\mathscr{L}$ over latent random variables.

- Each $q(\mathbf{z}) \in \mathscr{L}$ is a candidate approximation to the posterior.
- But how good is (each) approximation?
- Difference between two distributions: $D_{\mathrm{KL}}(\,\cdot \| \cdot\,)$ – aka the KL divergence.

Our goal:

$$q^*(\mathbf{z}) = \underset{q(\mathbf{z}) \in \mathscr{L}}{\arg\min}\, D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \,|\, \mathbf{x}))$$

Optimisation goal

In VI, we specify a family of distributions $\mathscr{L}$ over latent random variables.

- Each $q(\mathbf{z}) \in \mathscr{L}$ is a candidate approximation to the posterior.
- But how good is (each) approximation?
- Difference between two distributions: $D_{\mathrm{KL}}(\,\cdot \| \cdot\,)$ – aka the KL divergence.

Our goal:

$$q^*(\mathbf{z}) = \underset{q(\mathbf{z}) \in \mathscr{L}}{\arg\min}\, D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \,|\, \mathbf{x}))$$

where $q^*(\,\cdot\,)$ is the best approximation to the posterior in distribution family $\mathscr{L}$.

KL-divergence

Kullback–Leibler divergence

$$D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) = \int_{\mathbf{z}} q(\mathbf{z})\log\left[\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] d\mathbf{z}$$

KL-divergence

Kullback–Leibler divergence

$$D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) = \int_{\mathbf{z}} q(\mathbf{z})\log\left[\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] d\mathbf{z}$$

$$= \int_{\mathbf{z}} \left[q(\mathbf{z})\log q(\mathbf{z})\right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z}\log p(\mathbf{z}\,|\,\mathbf{x})\right] d\mathbf{z}$$

KL-divergence

Kullback–Leibler divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z}) \log \left[\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z}) \log q(\mathbf{z}) \right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z} \log p(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{z} \mid \mathbf{x}) \right]
\end{aligned}
$$

KL-divergence

Kullback–Leibler divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z}) \log \left[\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z}) \log q(\mathbf{z}) \right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z} \log p(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{z} \mid \mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log \left[\frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right] \right]
\end{aligned}
$$

KL-divergence

Kullback–Leibler divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z}) \log \left[\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z}) \log q(\mathbf{z}) \right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z} \log p(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{z} \mid \mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log \left[\frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right] \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \mathbb{E}_q \left[\log p(\mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \log p(\mathbf{x})
\end{aligned}
$$

KL-divergence

Kullback–Leibler divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z}) \log \left[\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z}) \log q(\mathbf{z}) \right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z} \log p(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{z} \mid \mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log \left[\frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right] \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \mathbb{E}_q \left[\log p(\mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \log p(\mathbf{x})
\end{aligned}
$$

z will not affect prior distribution of **x**, i.e., w can omit the expectation.

KL-divergence

Kullback–Leibler divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z})\log\left[\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z})\log q(\mathbf{z})\right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z}\log p(\mathbf{z}\,|\,\mathbf{x})\right] d\mathbf{z} \\
&= \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{z}\,|\,\mathbf{x})\right] \\
&= \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log\left[\frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})}\right]\right] \\
&= \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z})\right] + \mathbb{E}_q\left[\log p(\mathbf{x})\right] \\
&= \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z})\right] + \log p(\mathbf{x})
\end{aligned}
$$

$D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x}))$ contains a term $\log p(\mathbf{x})$, so using it for optimisation directly is also not tractable...

However, $\log p(\mathbf{x})$ is a constant, so we can just ignore it during optimisation.

$\mathbf{z}$ will not affect prior distribution of $\mathbf{x}$, i.e., w can omit the expectation.

KL-divergence

Kullback–Leibler divergence

The KL-divergence is strictly non-negative.

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) &= \int_{\mathbf{z}} q(\mathbf{z}) \log \left[\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})} \right] d\mathbf{z} \\
&= \int_{\mathbf{z}} \left[q(\mathbf{z}) \log q(\mathbf{z}) \right] d\mathbf{z} - \int_{\mathbf{z}} \left[q(\mathbf{z} \log p(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{z} \mid \mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log \left[\frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} \right] \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \mathbb{E}_q \left[\log p(\mathbf{x}) \right] \\
&= \mathbb{E}_q \left[\log q(\mathbf{z}) \right] - \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z}) \right] + \log p(\mathbf{x})
\end{aligned}
$$

$D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x}))$ contains a term $\log p(\mathbf{x})$, so using it for optimisation directly is also not tractable...

However, $\log p(\mathbf{x})$ is a constant, so we can just ignore it during optimisation.

$\mathbf{z}$ will not affect prior distribution of $\mathbf{x}$, i.e., w can omit the expectation.

ELBO

The evidence lower bound

Defined as:

$$\text{ELBO}(q) = \mathbb{E}_q\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q\left[\log q(\mathbf{z})\right]$$

ELBO

The evidence lower bound

Defined as:

$$\mathrm{ELBO}(q) = \mathbb{E}_q\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q\left[\log q(\mathbf{z})\right]$$

It is called the evidence lower bound, because it is the lower bound of logarithmic evidence:

$$\begin{aligned}\log p(\mathbf{x}) &= \mathrm{ELBO}(q) + D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) \\ &\geq \mathrm{ELBO}(q)\end{aligned}$$

ELBO

The evidence lower bound

$$D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) = \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z})\right] + \log p(\mathbf{x})$$

Defined as:

$$\mathrm{ELBO}(q) = \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z})\right] - \mathbb{E}_q\left[\log q(\mathbf{z})\right]$$

It is called the evidence lower bound, because it is the lower bound of logarithmic evidence:

$$\begin{aligned}\log p(\mathbf{x}) &= \mathrm{ELBO}(q) + D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) \\ &\geq \mathrm{ELBO}(q)\end{aligned}$$

Minimising D_{KL}

$$\begin{aligned}\log p(\mathbf{x}) &= \mathrm{ELBO}(q) + D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) \\ &\geq \mathrm{ELBO}(q)\end{aligned}$$

Minimising D_{KL}

$$\begin{aligned}\log p(\mathbf{x}) &= \mathrm{ELBO}(q) + D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x})) \\ &\geq \mathrm{ELBO}(q)\end{aligned}$$

Minimising $D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z} \mid \mathbf{x}))$ is equivalent to maximising $\mathrm{ELBO}(q)$.

Minimising D_{KL}

$$\begin{aligned}\log p(\mathbf{x}) &= \mathrm{ELBO}(q) + D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})) \\ &\geq \mathrm{ELBO}(q)\end{aligned}$$

Minimising $D_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x}))$ is equivalent to maximising $\mathrm{ELBO}(q)$.

$$\begin{aligned}q^*(\mathbf{z}) &= \arg\min_{q(\mathbf{z})\in\mathscr{L}} D_{\mathrm{KL}}\left(q(\mathbf{z})\|p(\mathbf{z}\,|\,\mathbf{x})\right) \\ &= \arg\max_{q(\mathbf{z})\in\mathscr{L}} \mathrm{ELBO}(q) \\ &= \arg\max_{q(\mathbf{z})\in\mathscr{L}} \left\{\mathbb{E}_q[\log p(\mathbf{x},\mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z})]\right\}\end{aligned}$$

Families of distributions

For variational inference (examples)

- **Mean-field family**: this variational distribution factorises over the latent variables.
  It means that each latent variable has its own variational distribution, and these distributions are assumed to be independent. The simplest and most commonly used family in VI.

- **Gaussian family**: Gaussian distributions are widely used in VI due to their simplicity and nice properties, such as being closed under linear transformations and having a well-defined mean and covariance matrix. The multivariate Gaussian distribution, in particular, is a common choice for continuous latent variables.

- **Exponential family**: The exponential family of distributions includes Gaussian, exponential, gamma, beta, Dirichlet, Bernoulli, Poisson, and many others. These distributions have some desirable properties, such as conjugacy and sufficient statistics, which can simplify the VI algorithm and lead to closed-form updates.

- **Mixture models**: Mixture models are a combination of multiple simpler distributions, typically from the same family, and can be used to approximate more complex distributions. Gaussian mixture models, for example, are widely used in VI.

- **Neural networks**: In variational autoencoders we use neural networks to parameterise the variational distribution. This allows for a highly flexible family of distributions, to model complex high-dimensional data.

Mean-field variational family

In a mean-field variational family of distributions, each variable z_j has its own variational factor. The variational distribution factorises over the latent variables:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$$

Mean-field variational family

In a mean-field variational family of distributions, each variable z_j has its own variational factor. The variational distribution factorises over the latent variables:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$$

Here, $q_j(z_j)$ is the variational factor for the variable z_j. The mean-field assumption treats each latent variable as independent, which simplifies the optimisation problem in variational inference.

Potential dependencies among the latent variables can not be captured by this approximation.

Example

Mean-field approximation

- Bayesian mixture of Gaussians. Using mean-field:

$$q(\mathbf{z}) = q(\mu, \mathbf{c})$$
$$= \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i)$$

Example

Mean-field approximation

- Bayesian mixture of Gaussians. Using mean-field:

$$q(\mathbf{z}) = q(\mu, \mathbf{c})$$
$$= \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i)$$

- The factor $q(\mu_k; m_k, s_k^2)$ is a Gaussian distribution for μ_k – the mean of the kth Gaussian in the mixture.

Example

Mean-field approximation

- Bayesian mixture of Gaussians. Using mean-field:

$$q(\mathbf{z}) = q(\mu, \mathbf{c})$$
$$= \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i)$$

- The factor $q(\mu_k; m_k, s_k^2)$ is a Gaussian distribution for μ_k – the mean of the kth Gaussian in the mixture.

- The factor $q(c_i; \varphi_i)$ is a distribution for c_i – the assigned cluster ID for the Gaussian, for observation x_i sampling.

Example

Mean-field approximation

- Bayesian mixture of Gaussians. Using mean-field:

$$q(\mathbf{z}) = q(\mu, \mathbf{c})$$
$$= \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i)$$

- The factor $q(\mu_k; m_k, s_k^2)$ is a Gaussian distribution for μ_k – the mean of the kth Gaussian in the mixture.

- The factor $q(c_i; \varphi_i)$ is a distribution for c_i – the assigned cluster ID for the Gaussian, for observation x_i sampling.

- φ_i is a vector of length K. It contains the probabilities of selecting each Gaussian for x_i sampling.

Coordinate ascent optimisation

Idea

- An iterative optimisation technique
- At each step, we optimise one variable while keeping the other variables fixed
- We cycle through all the variables, updating one at a time.
- Repeat this until we reach a maximum number of iterations, or until the change in the objective function becomes very small).
- Breaks down the optimisation problem into a series of smaller, easier-to-solve subproblems.

Using mean-field variational approximation, we can decompose the terms in ELBO as

$$\mathbb{E}_q \left[\log q(\mathbf{z})\right] = \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right]$$

Using mean-field variational approximation, we can decompose the terms in ELBO as

$$\mathbb{E}_q \left[\log q(\mathbf{z})\right] = \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right]$$

Because $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}) \prod_{i=1}^{m} p(z_i \mid z_{1:(i-1)}, \mathbf{x})$:

$$\mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z})\right] = \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q \left[\log p(z_j \mid z_{1:(i-1)}, \mathbf{x})\right]$$

Using mean-field variational approximation, we can decompose the terms in ELBO as

$$\mathbb{E}_q \left[\log q(\mathbf{z})\right] = \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right]$$

Because $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}) \prod_{i=1}^{m} p(z_i \mid z_{1:(i-1)}, \mathbf{x})$:

$$\mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z})\right] = \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q \left[\log p(z_j \mid z_{1:(i-1)}, \mathbf{x})\right]$$

We could write the ELBO as:

$$\begin{aligned}
\text{ELBO}(q) &= \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q \left[\log q(\mathbf{z})\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q \left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \left[\mathbb{E}_q \left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \mathbb{E}_{q_i} \left[\log q(z_i)\right]\right]
\end{aligned}$$

Using mean-field variational approximation, we can decompose the terms in ELBO as

$$\mathbb{E}_q \left[\log q(\mathbf{z})\right] = \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right]$$

Because $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}) \prod_{i=1}^{m} p(z_i \mid z_{1:(i-1)}, \mathbf{x})$:

$$\mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z})\right] = \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q \left[\log p(z_j \mid z_{1:(i-1)}, \mathbf{x})\right]$$

We could write the ELBO as:

$$\begin{aligned}
\text{ELBO}(q) &= \mathbb{E}_q \left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q \left[\log q(\mathbf{z})\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q \left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[\log q(z_i)\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \left[\mathbb{E}_q \left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \mathbb{E}_{q_i} \left[\log q(z_i)\right]\right]
\end{aligned}$$

In every step of the coordinate ascent, we optimise for one variable z_j at a time.

$$\text{ELBO}(q_j) = \log p(\mathbf{x}) + \mathbb{E}_{q_{-j}} \left[\log p(z_j \mid z_{1:(j-1)}, \mathbf{x})\right] - \mathbb{E}_{q_j} \left[\log q(z_j)\right]$$

Using mean-field variational approximation, we can decompose the terms in ELBO as

$$\mathbb{E}_q\left[\log q(\mathbf{z})\right] = \sum_{i=1}^{m} \mathbb{E}_{q_i}\left[\log q(z_i)\right]$$

Because $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}) \prod_{i=1}^{m} p(z_i \mid z_{1:(i-1)}, \mathbf{x})$:

$$\mathbb{E}_q\left[\log p(\mathbf{x}, \mathbf{z})\right] = \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q\left[\log p(z_j \mid z_{1:(i-1)}, \mathbf{x})\right]$$

We could write the ELBO as:

$$\begin{aligned}
\text{ELBO}(q) &= \mathbb{E}_q\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_q\left[\log q(\mathbf{z})\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \mathbb{E}_q\left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \sum_{i=1}^{m} \mathbb{E}_{q_i}\left[\log q(z_i)\right] \\
&= \log p(\mathbf{x}) + \sum_{i=1}^{m} \left[\mathbb{E}_q\left[\log p(z_i \mid z_{1:(i-1)}, \mathbf{x})\right] - \mathbb{E}_{q_i}\left[\log q(z_i)\right]\right]
\end{aligned}$$

In every step of the coordinate ascent, we optimise for one variable z_j at a time.

$$\text{ELBO}(q_j) = \log p(\mathbf{x}) + \mathbb{E}_{q_{-j}}\left[\log p(z_j \mid z_{1:(j-1)}, \mathbf{x})\right] - \mathbb{E}_{q_j}\left[\log q(z_j)\right]$$

$-D_{\text{KL}}$

Update step

$$\mathrm{ELBO}(q_j) = -D_{\mathrm{KL}}(q_j(z_j) \| \widetilde{p}_j(z_j, \mathbf{x})) + \mathrm{const}$$

Update step

$$\log \widetilde{p}_j(z_j, \mathbf{x}) = \mathbb{E}_{q_{-j}}\left[\log p(z, \mathbf{x})\right] + \text{const}$$

$$\text{ELBO}(q_j) = -D_{\text{KL}}(q_j(z_j) \| \widetilde{p}_j(z_j, \mathbf{x})) + \text{const}$$

Update step

$$\log \widetilde{p}_j(z_j, \mathbf{x}) = \mathbb{E}_{q_{-j}}\left[\log p(z, \mathbf{x})\right] + \text{const}$$

$$\text{ELBO}(q_j) = -D_{\text{KL}}(q_j(z_j) \| \widetilde{p}_j(z_j, \mathbf{x})) + \text{const}$$

Now, since D_{KL} is non-negative, the $\text{ELBO}(q_j)$ is maximised when $q_j(z_j) = \widetilde{p}(z_j, \mathbf{x})$

(because then it is 0).

Update step

$$\log \widetilde{p}_j(z_j, \mathbf{x}) = \mathbb{E}_{q_{-j}}\left[\log p(z, \mathbf{x})\right] + \text{const}$$

$$\text{ELBO}(q_j) = -D_{\text{KL}}(q_j(z_j) \| \widetilde{p}_j(z_j, \mathbf{x})) + \text{const}$$

Now, since D_{KL} is non-negative, the $\text{ELBO}(q_j)$ is maximised when $q_j(z_j) = \widetilde{p}(z_j, \mathbf{x})$ (because then it is 0).

$q_j(z_j) = \widetilde{p}(z_j, \mathbf{x})$ is equivalent to

$$q_j^*(z_j) \propto \exp\left\{ \mathbb{E}_{q_{-j}}\left[\log p(\mathbf{z}, \mathbf{x})\right] \right\}$$

Coordinate ascent variational inference (CAVI)

Pseudo-code

Algorithm 1: Coordinate ascent variational inference (CAVI)

Input: A model $p(\mathbf{x}, \mathbf{z})$, a data set $\mathbf{x}$

Output: A variational density $q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$

Initialize: Variational factors $q_j(z_j)$

while *the* ELBO *has not converged* **do**

  for $j \in \{1, \ldots, m\}$ **do**

    Set $q_j(z_j) \propto \exp\{\mathbb{E}_{-j}[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})]\}$

  end

  Compute $\text{ELBO}(q) = \mathbb{E}[\log p(\mathbf{z}, \mathbf{x})] - \mathbb{E}[\log q(\mathbf{z})]$

end

return $q(\mathbf{z})$

Lei Mao, https://leimao.github.io/article/Introduction-to-Variational-Inference/

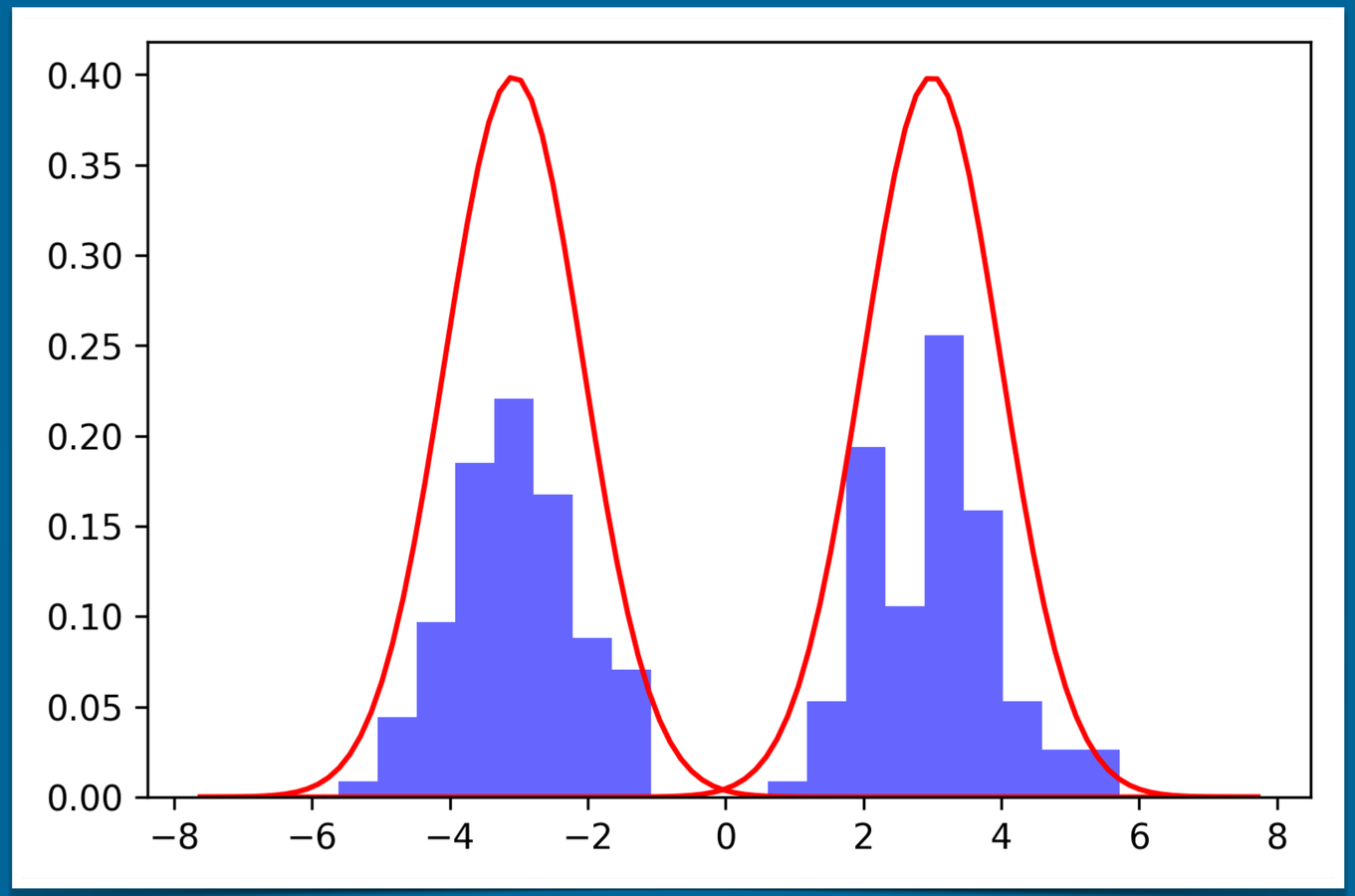
In Python

```
## simple example using mean-field VI with two components

# Generate synthetic data
np.random.seed(42)
N = 200
X = np.concatenate([np.random.normal(-3, 1, N//2), np.random.normal(3, 1, N//2)])

# Initialise variational parameters
mu_params = np.random.normal(0, 1, 2)
the_c = np.random.rand(N, 2)
the_c /= the_c.sum(axis=1, keepdims=True)

# Parameters for the prior distributions of the means
prior_mu = np.array([0, 0])
prior_sigma2 = np.array([1, 1])

# Known variances for each Gaussian component
component_variances = np.array([1, 1])
```

```
## CAVI algorithm (not explicitly computing the ELBO)

max_iter = 1000
tolerance = 1e-6
for iter_idx in range(max_iter):
    mu_params_old = mu_params.copy()

    # Update q(z_i)
    for i in range(N):
        for k in range(2):
            the_c[i, k] = np.exp(-(X[i] - mu_params[k])**2 /
                                 (2 * component_variances[k])) / np.sqrt(2 * np.pi *
                                                                         component_variances[k])
        the_c[i] /= the_c[i].sum()

    # Update q(μ_k)
    for k in range(2):
        numerator = (the_c[:, k] * X).sum()
        denominator = the_c[:, k].sum()
        mu_params[k] = (prior_sigma2[k] * numerator + prior_mu[k]) / (prior_sigma2[k] *
                                                                      denominator + 1)

    # Check convergence
    if np.linalg.norm(mu_params - mu_params_old) < tolerance:
        break

print("CAVI converged after", iter_idx, "iterations")
print("Estimated means:", mu_params)
```

```
## Plot the results

plt.hist(X, bins=20, density=True, alpha=0.6, color='blue')
x_range = np.linspace(X.min() - 2, X.max() + 2, 100)
for k in range(2):
    plt.plot(x_range, np.exp(-(x_range - mu_params[k])**2 /
                             (2 * component_variances[k])) / np.sqrt(2 * np.pi *
component_variances[k]), color='red')
plt.show()
```

```
## Plot the results

plt.hist(X, bins=20, density=True, alpha=0.6, color='blue')
x_range = np.linspace(X.min() - 2, X.max() + 2, 100)
for k in range(2):
    plt.plot(x_range, np.exp(-(x_range - mu_params[k])**2 /
                             (2 * component_variances[k])) / np.sqrt(2 * np.pi *
component_variances[k]), color='red')
plt.show()
```



The end

for today