Workshop Bayesian Corpus Studies

Christoph Finkensiep
Würzburg, Feb 2024

Session 3: Understanding Inference Methods

Inference and Conditioning

Inference as Conditioning

Inference = computing a conditional distribution given the join distribution.

Model:

$$p(x, y)$$

Observing x:

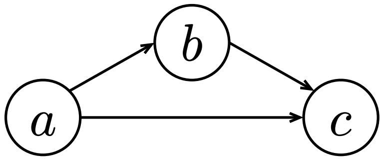
$$p(x \mid y = 4) = \frac{p(x, y = 4)}{p(x = 4)}$$

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$

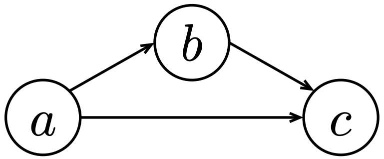
Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



Factorization

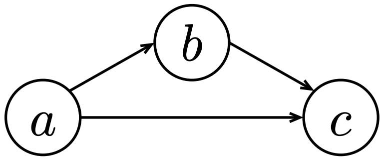
1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



A probabilistic program is a *factorization* of a joint distribution.

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



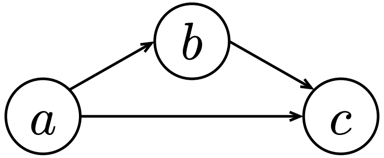
A probabilistic program is a *factorization* of a joint distribution.

$p(a, b, c)$

$= p(a) \cdot p(b, c \mid a)$

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



A probabilistic program is a *factorization* of a joint distribution.

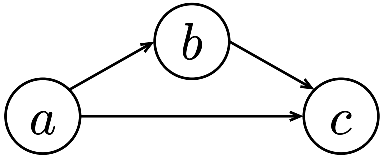
$$p(a, b, c)$$

$$= p(a) \cdot p(b, c \mid a)$$

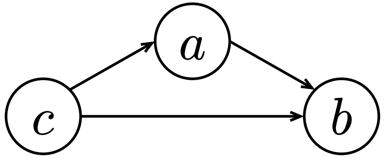
$$= p(a) \cdot p(b \mid a) \cdot p(c \mid b, a)$$

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



1. draw c
2. draw $a \mid c$
3. draw $b \mid a, c$



A probabilistic program is a *factorization* of a joint distribution.

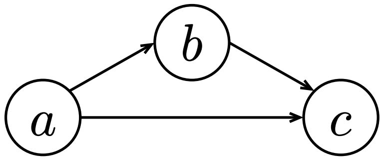
$p(a, b, c)$

$= p(a) \cdot p(b, c \mid a)$

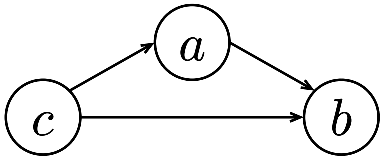
$= p(a) \cdot p(b \mid a) \cdot p(c \mid b, a)$

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



1. draw c
2. draw $a \mid c$
3. draw $b \mid a, c$



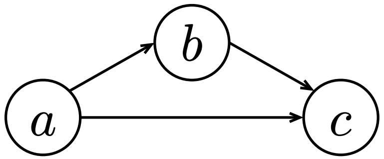
A probabilistic program is a *factorization* of a joint distribution.

$$
\begin{aligned}
&p(a, b, c) \\
&= p(a) \cdot p(b, c \mid a) \\
&= p(a) \cdot p(b \mid a) \cdot p(c \mid b, a)
\end{aligned}
$$

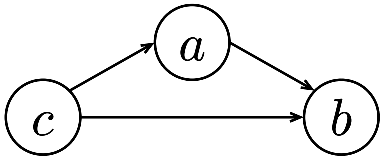
$$
\begin{aligned}
&p(a, b, c) \\
&= p(c) \cdot p(a, b \mid c)
\end{aligned}
$$

Factorization

1. draw a
2. draw $b \mid a$
3. draw $c \mid a, b$



1. draw c
2. draw $a \mid c$
3. draw $b \mid a, c$



A probabilistic program is a *factorization* of a joint distribution.

$$
\begin{aligned}
&p(a, b, c) \\
&= p(a) \cdot p(b, c \mid a) \\
&= p(a) \cdot p(b \mid a) \cdot p(c \mid b, a)
\end{aligned}
$$

$$
\begin{aligned}
&p(a, b, c) \\
&= p(c) \cdot p(a, b \mid c) \\
&= p(c) \cdot p(a \mid c) \cdot p(b \mid a, c)
\end{aligned}
$$

Conditioning Probabilistic Programs

$$p(x, y) = p(x) \cdot p(y \mid x)$$

Process:

- draw x
- draw $y \mid x$

Conditioning Probabilistic Programs

$$p(x, y) = p(x) \cdot p(y \mid x)$$

Process:

- draw x
- draw $y \mid x$

Condition on x (“upstream”):

- set $x = 5$
- draw $y \mid x = 5$

$p(y \mid x)$ is already known

Conditioning Probabilistic Programs

$$p(x, y) = p(x) \cdot p(y \mid x)$$

Process:

- draw x
- draw $y \mid x$

Condition on x ("upstream"):

- set $x = 5$
- draw $y \mid x = 5$

$p(y \mid x)$ is already known

Condition on y ("downstream"):

- set $y = 4$
- draw $x \mid y = 4$???

$p(x \mid y)$ is not explicitly given

Inference Methods

Running a Probabilistic Program

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

sample from $p(x, y)$? $\rightarrow$ run program!
sample from)?

Running a Probabilistic Program

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

sample from $p(x, y)$? $\rightarrow$ run program!
sample from $p(x \mid y = 4)$?

Running a Probabilistic Program

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

sample from $p(x, y)$? $\rightarrow$ run program!
sample from $p(x \mid y = 4)$?

Idea:

- run the program many times: sample from $p(x, y)$
- select outcomes where $y = 4$: sample from $p(x \mid y = 4)$

Running a Probabilistic Program

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

sample from $p(x, y)$? $\to$ run program!
sample from $p(x \mid y = 4)$?

Idea:

- run the program many times: sample from $p(x, y)$
- select outcomes where $y = 4$: sample from $p(x \mid y = 4)$

$\Rightarrow$ “rejection sampling”

Running a Probabilistic Program

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

sample from $p(x, y)$? $\rightarrow$ run program!
sample from $p(x \mid y = 4)$?

Idea:

- run the program many times: sample from $p(x, y)$
- select outcomes where $y = 4$: sample from $p(x \mid y = 4)$

$\Rightarrow$ "rejection sampling"

How often is $y = 4$ by chance?
What if y a dataset? What if y is continuous?

Computing Relative Probabilities

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

We can run the program to compute $p(x, y)$ for fixed x and y.

Computing Relative Probabilities

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

We can run the program to compute $p(x, y)$ for fixed x and y.

- easy to compute $p(x, y = 4)$ for different x

Computing Relative Probabilities

Process:

- draw x
- draw $y \mid x$

Given: $y = 4$

We can run the program to compute $p(x, y)$ for fixed x and y.

- easy to compute $p(x, y = 4)$ for different x
- still can't compute $p(x, y = 4)$, but can compare different x:

$$\frac{p(x_1 \mid y = 4)}{p(x_2 \mid y = 4)} = \frac{\frac{p(x_1, y=4)}{p(y=4)}}{\frac{p(x_2, y=4)}{p(y=4)}} = \frac{p(x_1, y = 4)}{p(x_2, y = 4)}$$

Better Sampling: Metropolis-Hastings Algorithm

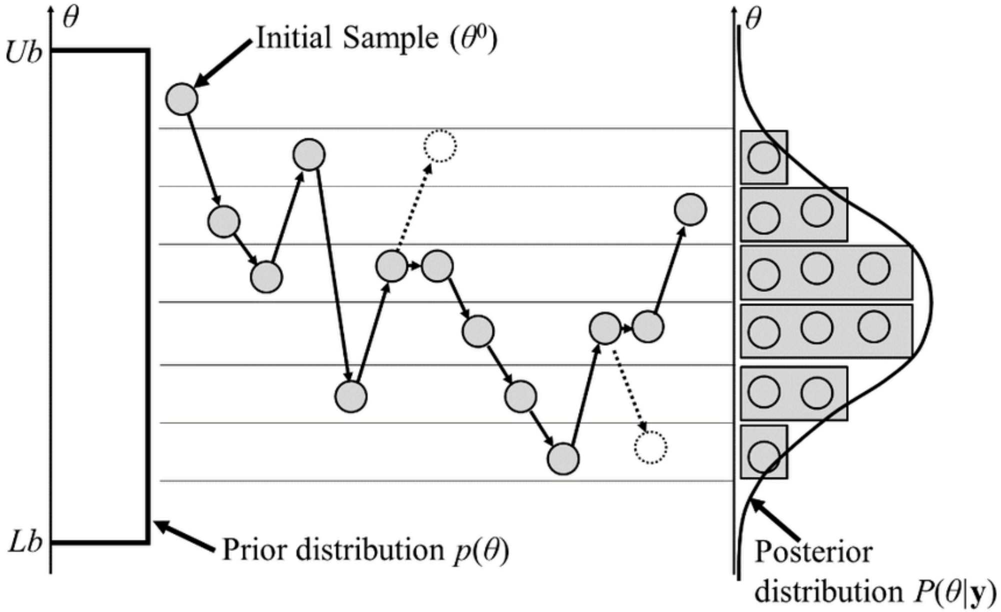
Idea: random walk, move between different values of x

- use a *proposal distribution*: $g(x_{t+1} \mid x_t)$
- compare values using *score*: $f(x) = p(x, y = 4)$

Better Sampling: Metropolis-Hastings Algorithm

Idea: random walk, move between different values of x

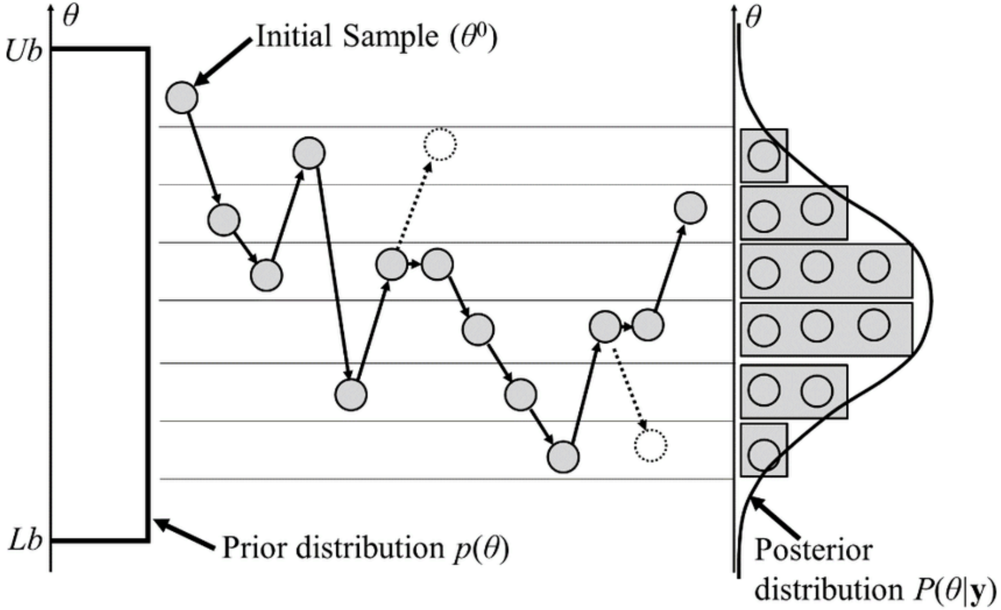
- use a *proposal distribution*: $g(x_{t+1} \mid x_t)$
- compare values using *score*: $f(x) = p(x, y = 4)$



Better Sampling: Metropolis-Hastings Algorithm

Idea: random walk, move between different values of x

- use a *proposal distribution*: $g(x_{t+1} \mid x_t)$
- compare values using *score*: $f(x) = p(x, y = 4)$

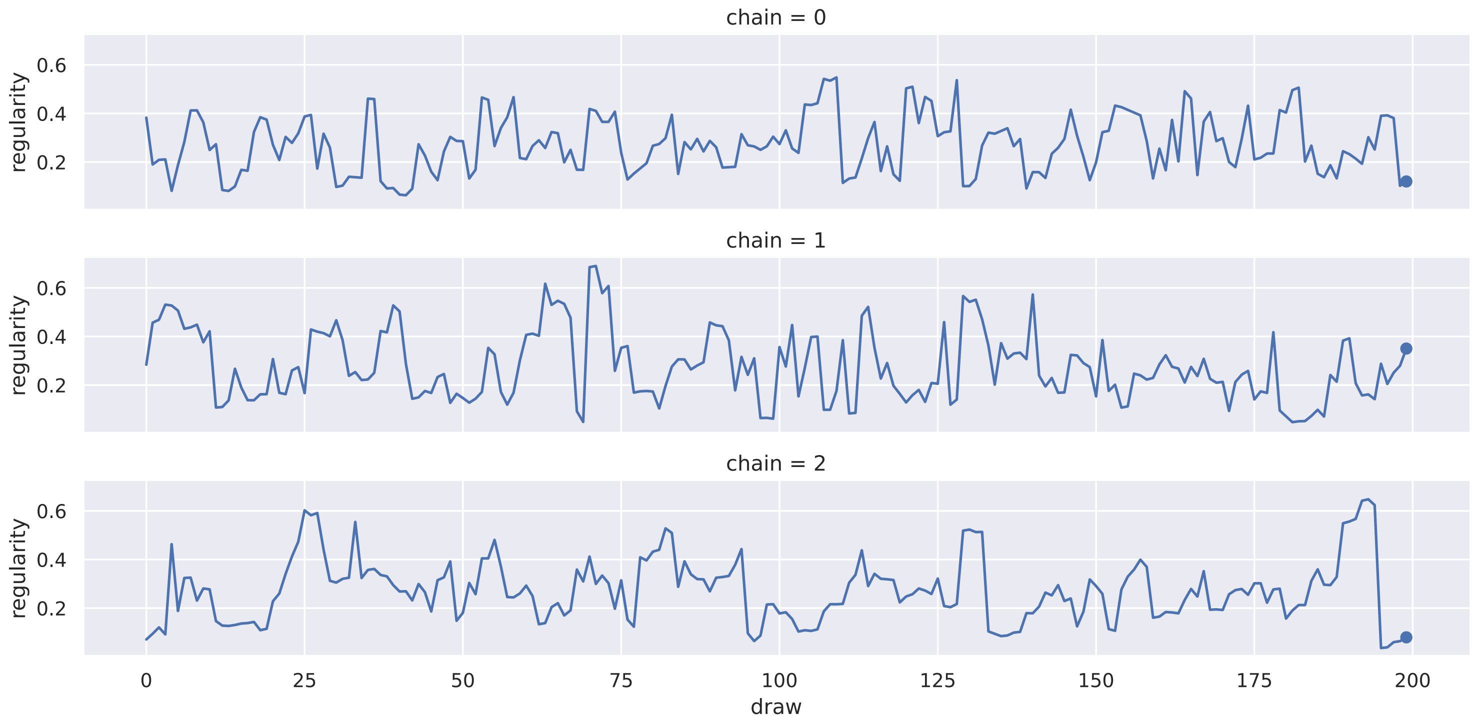


Metropolis-Hastings:

- choose x_0 random
- for each step t:
  - choose $x' \mid x_t \sim g$
  - compute $\alpha = \frac{f(x')}{f(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}$
  - draw random $u \sim \text{Uniform}(0, 1)$:
    - if $u \leq \alpha$: $x_{t+1} = x'$
    - if $u > \alpha$: $x_{t+1} = x_t$

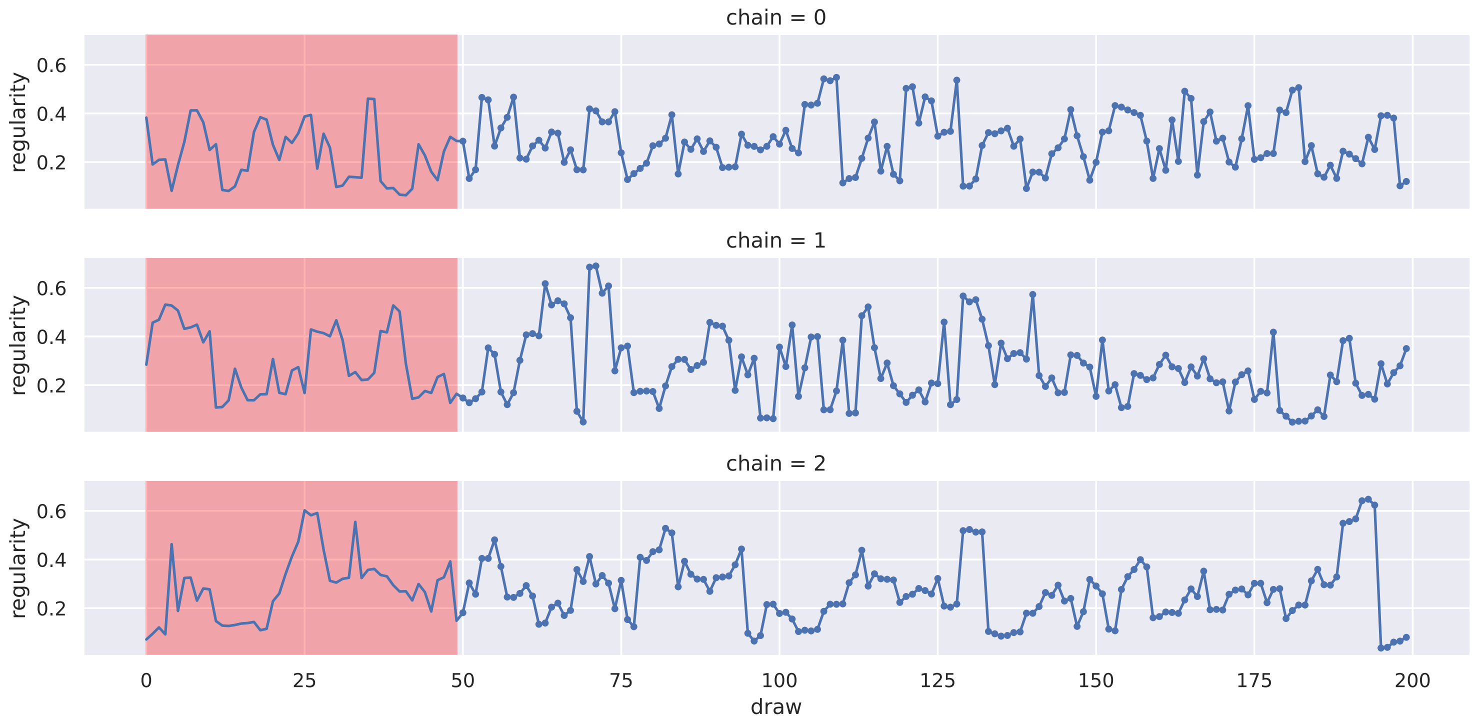
Markov-Chain Monte Carlo (MCMC)

Theory: run chain as long as possible, take last state, repeat (slow)



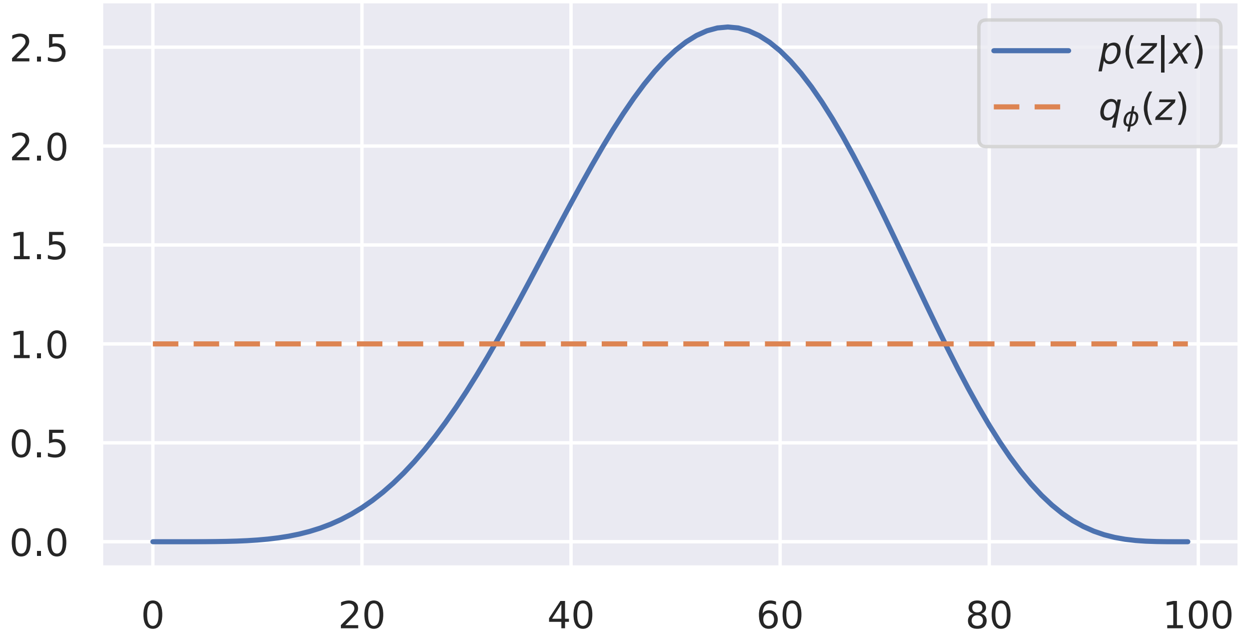
Markov-Chain Monte Carlo (MCMC)

Practice: “burn-in” / “tune” phase, then take all samples (correlated but faster)



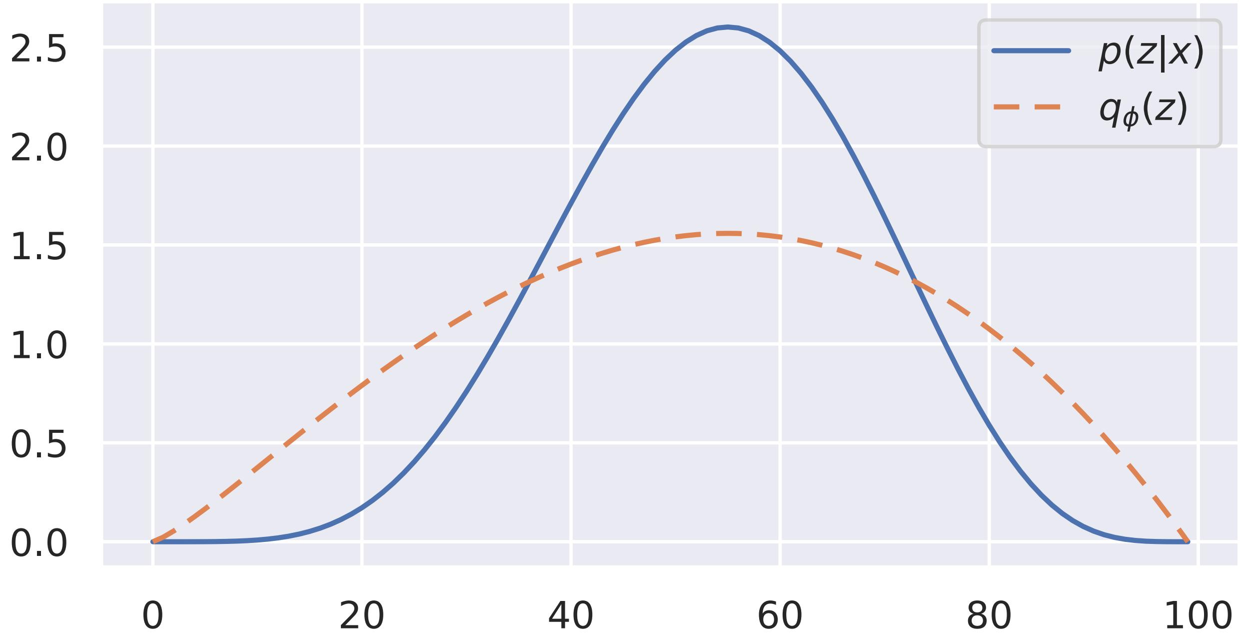
Variational Inference: Intuition

Idea: approximate the posterior through optimization / gradient descent



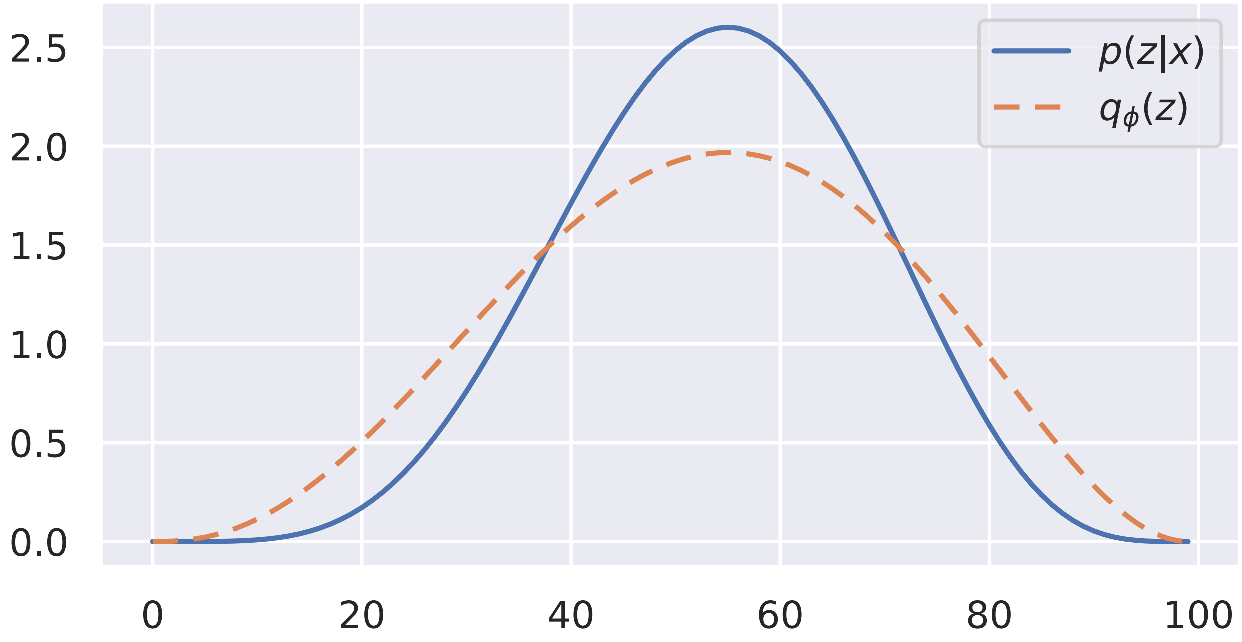
Variational Inference: Intuition

Idea: approximate the posterior through optimization / gradient descent



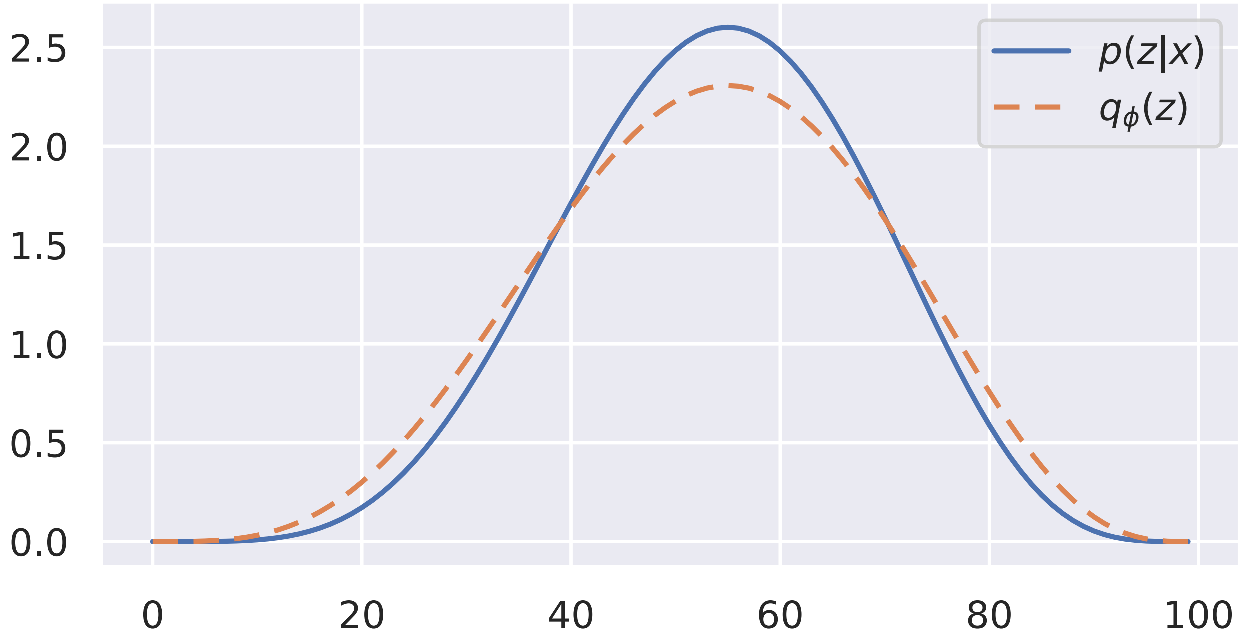
Variational Inference: Intuition

Idea: approximate the posterior through optimization / gradient descent



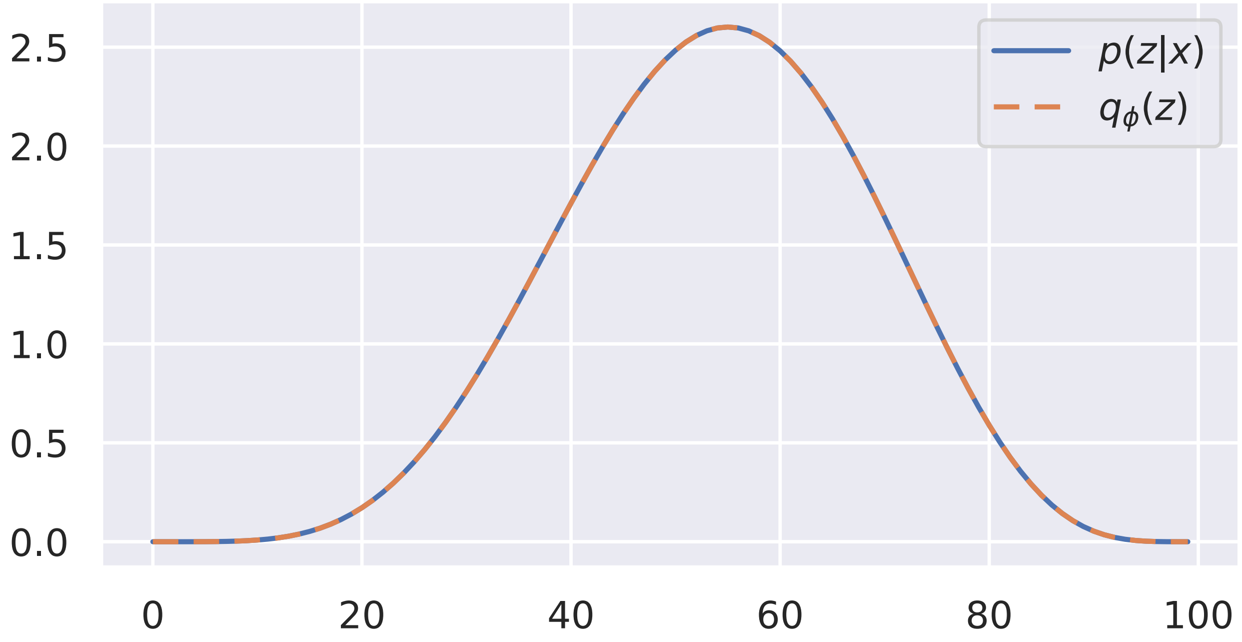
Variational Inference: Intuition

Idea: approximate the posterior through optimization / gradient descent



Variational Inference: Intuition

Idea: approximate the posterior through optimization / gradient descent



Variational Inference: Math

Define a *variational family*: $q_\varphi(z)$

- known shape (can be simpler than true posterior)
- parameters φ

Variational Inference: Math

Define a *variational family*: $q_\varphi(z)$

- known shape (can be simpler than true posterior)
- parameters φ

Optimize φ:

- loss: KL divergence $D_{\mathrm{KL}}\left(q_\varphi(z) \parallel p(z \mid x)\right) = \mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z \mid x)}\right]$

Variational Inference: Math

Define a *variational family*: $q_\varphi(z)$

- known shape (can be simpler than true posterior)
- parameters φ

Optimize φ:

- loss: KL divergence $D_{\mathrm{KL}}\left(q_\varphi(z) \parallel p(z \mid x)\right) = \mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z \mid x)}\right]$
    - not tractable, but equivalent to "evidence lower bound" (ELBO):

$$\mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z \mid x)}\right] = \mathbb{E}_q\left[\log \frac{q_\varphi(z)p(x)}{p(z,x)}\right] = \underbrace{\mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z,x)}\right]}_{\substack{\text{-"ELBO"} \\ \text{(computable!)}}} + \underbrace{\log p(x)}_{\substack{\text{"evidence"} \\ \text{(const!)}}}$$

Variational Inference: Math

Define a *variational family*: $q_\varphi(z)$
- known shape (can be simpler than true posterior)
- parameters φ

Optimize φ:
- loss: KL divergence $D_{\mathrm{KL}}\left(q_\varphi(z) \parallel p(z \mid x)\right) = \mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z \mid x)}\right]$
  - not tractable, but equivalent to "evidence lower bound" (ELBO):

$$\mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z \mid x)}\right] = \mathbb{E}_q\left[\log \frac{q_\varphi(z)p(x)}{p(z,x)}\right] = \underbrace{\mathbb{E}_q\left[\log \frac{q_\varphi(z)}{p(z,x)}\right]}_{\substack{\text{-"ELBO"} \\ \text{(computable!)}}} + \underbrace{\log p(x)}_{\substack{\text{“evidence”} \\ \text{(const!)}}}$$

- use gradient descent (in practice: autodiff)

Bonus: Why "ELBO"?

"Evidence lower bound":

$$D_{\mathrm{KL}}\left(q_\varphi \parallel p\right) = -\mathrm{ELBO}(\varphi) + \log p(x)$$

$$\Rightarrow$$

$$D_{\mathrm{KL}}\left(q_\varphi \parallel p\right) + \mathrm{ELBO}(\varphi) = \log p(x)$$

$$\Rightarrow$$

$$\mathrm{ELBO}(\varphi) \leq \log p(x)$$

Bonus: Why "ELBO"?

"Evidence lower bound":

$$D_{\mathrm{KL}}\left(q_\varphi \parallel p\right) = -\mathrm{ELBO}(\varphi) + \log p(x)$$
$$\Rightarrow$$
$$D_{\mathrm{KL}}\left(q_\varphi \parallel p\right) + \mathrm{ELBO}(\varphi) = \log p(x)$$
$$\Rightarrow$$
$$\mathrm{ELBO}(\varphi) \le \log p(x)$$

(should be "log-evidence lower bound")

Comparison: Sampling vs Variational Inference

Sampling:

- result: sample of $p(z \mid x)$
- very flexible
- can be slow on large data
- can be tricky to get right
- use: PyMC / numpyro / …

Variational Inference:

- result: $q_{\varphi(z)}$
- integrates with deep learning (VAE)
- fast on large data, can be slow to converge
- can be tricky to get right
- use: pyro / numpyro