Bayesian inference
in generative models

Luke Hewitt and Maddie Cusimano
BCS Computational Tutorial
2018-11-13
Overview

1. Intro to generative models (5 min)
2. Exact inference (10 min)
3. Sampling-based methods (25 min)
4. Variational inference (20 min)
5. Probabilistic programming languages (5 min)
6. Exercises (Last hour)
What is a generative model?

A probability distribution over observable variables

\[
\begin{align*}
P(S \rightarrow NP \ VP) &= 1.0 \\
P(VP \rightarrow V \ NP) &= 0.7 \\
P(VP \rightarrow VP \ PP) &= 0.3 \\
P(N \rightarrow dog) &= 0.02 \\
P(N \rightarrow fish) &= 0.01 \\
P(N \rightarrow man) &= 0.01
\end{align*}
\]

The dog saw a man in the park

\[
\text{e.g. } P(\text{pixel1}) \times P(\text{pixel2}|\text{pixel1}) \times \ldots
\]
What is a generative model?

Common to describe distribution through unobservable **latent variables**

e.g. **directed causal model:**

- **Prior** distribution of latent variables
- **Likelihood** of observations given latent variables
What is a generative model?

Common to describe distribution through unobservable **latent variables**

e.g. **directed causal model**:

- **Prior** distribution of latent variables
- **Likelihood** of observations given latent variables

Bayes rule:

Given observations, infer latent variables

\[
p(\text{shadow} | \text{img}) \propto p(\text{shadow}) \cdot p(\text{img} | \text{shadow})
\]

More generally:

\[
p(\text{shadow} | \text{img}) \propto p(\text{shadow}, \text{img})
\]

(e.g. undirected models)
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph

Simplest directed causal model: \( p(z, x) = p(z) p(x | z) \)

\[
\begin{align*}
p(z) &= N(z; \mu=0, \sigma=1) \\
p(x | z) &= N(x; \mu=z, \sigma=1)
\end{align*}
\]
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph

Simplest directed causal model: $p(z, x) = p(z) p(x | z)$

- $p(z) = N(z; \mu=0, \sigma=1)$
- $p(x | z) = N(x; \mu=z, \sigma=1)$

Latent variable $z$

Observed variable $x$
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph.

Directed causal model: \( p(z, x_1, x_2, x_3) = p(z) p(x_1 \mid z)p(x_2 \mid z) p(x_3 \mid z) \)

\[
\begin{align*}
p(z) &= N(z; \mu=0, \sigma=1) \\
For \ i = [1, 2, 3]: & \quad p(x_i \mid z) = N(x; \mu=z, \sigma=1)
\end{align*}
\]
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph

Directed causal model: $p(z, x_1, x_2, x_3) = p(z) \ p(x_1 \mid z)p(x_2 \mid z) \ p(x_3 \mid z)$

$p(z) = N(z; \mu=0, \sigma=1)$

For $i = \{1, 2, 3\}$:
$p(x_i \mid z) = N(x; \mu=z, \sigma=1)$
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph

\[
p(\text{sun}, \text{friend}, \text{raise}, \text{hot}, \text{happy}, \text{shirt}) = p(\text{sun})p(\text{friend})p(\text{raise}) \times p(\text{happy} \mid \text{sun}, \text{friend}, \text{raise}) \times p(\text{hot} \mid \text{sun})p(\text{shirt} \mid \text{hot}, \text{happy})
\]

Probabilistic programs: an extension of graphical models that express uncertainty over the structure of the graph itself
Exact Inference
Exact Inference

- When latent variable is **discrete** with finite support,
  - Can enumerate all possibilities
    \[ p(z|x) = \frac{p(z, x)}{\sum_z p(z, x)} \]

- When latent variable is **continuous**:
  - Some likelihood functions \( p(x | z) \) have **conjugate priors**, which allow the posterior to be computed analytically
  - If a conjugate prior is used, \( p(z | x) \) and \( p(z) \) will be the same type of probability distribution: simply update the prior parameters

For a list of conjugate pairs: https://en.wikipedia.org/wiki/Conjugate_prior
Graphical models

Generative model for which the conditional dependence structure between random variables is expressed as a graph

Simplest directed causal model: \( p(z, x) = p(z) \ p(x \mid z) \)

\[
\begin{align*}
    p(z) &= \mathcal{N}(z; \mu=0, \sigma^2=1) \\
    p(x \mid z) &= \mathcal{N}(x; \mu=z, \sigma^2=1)
\end{align*}
\]

Gaussian conjugate prior on \( \mu \)

Gaussian likelihood

Posterior is also Gaussian! \( \mathcal{N}(z; \mu=\frac{1}{2}(x + 0), \sigma^2=\frac{1}{2}) \)

For a list of conjugate pairs: https://en.wikipedia.org/wiki/Conjugate_prior
## Conjugate priors

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Conjugate prior</th>
<th>Posterior update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \sim \text{Normal}(\mu=z, 1)$</td>
<td>$z \sim \text{Normal}(\mu_0, \sigma_0^2)$</td>
<td>$z \mid x \sim \text{Normal}\left(\frac{x + \mu_0/\sigma_0^2}{1 + \sigma_0^2}, \frac{1}{1 + \sigma_0^2}\right)$</td>
</tr>
<tr>
<td>$x \sim \text{Normal}(\mu=0, \sigma^2=z)$</td>
<td>$z \sim \text{InvGamma}(\alpha, \beta)$</td>
<td>$z \mid x \sim \text{InvGamma}(\alpha + \frac{1}{2}, \beta + \frac{1}{2} x^2)$</td>
</tr>
<tr>
<td>$x \sim \text{Bernoulli}(p=z)$</td>
<td>$z \sim \text{Beta}(\alpha, \beta)$</td>
<td>$z \mid x \sim \text{Beta}(\alpha + x, \beta + 1 - x)$</td>
</tr>
</tbody>
</table>

For a list of conjugate pairs: [https://en.wikipedia.org/wiki/Conjugate_prior](https://en.wikipedia.org/wiki/Conjugate_prior)
Belief propagation (‘Sum-product algorithm’)

If latent variables form a sequence (or a tree) can find marginals \( p(z_i) \) exactly
(Also ‘max-product’ for finding MAP)

e.g. Hidden Markov Models, Linear Gaussian State Space Models
Junction Tree Algorithm

If cyclic, must first group latent variables together.
(Still exact, but exponentially expensive...)
Exact Inference

Only possible...

1. For simple distributions which are either finite or conjugate
2. In small models, where you can enumerate all possible latent variables
3. In large models, if latent variables are not cyclic
   (see also, Junction Tree Algorithm)

Not very often..., BUT exact inference is often used as part of an approximate algorithm
Approximate Inference
Approximate Inference

To approximate the posterior, two main ideas (both from physicists):

- **Monte Carlo (1946, 1953, 1970, ...)**
  
  Represent posterior as collection of (weighted) samples, \{z_1, z_2, ...\}

- **Variational Inference (~1990s+)**
  
  Represent posterior as a parametric distribution, Q(z) (e.g. gaussian)

To supplement these ideas (2000s):

- **Amortized Inference:** Learn to do inference quickly.
  
  (‘bottom-up’, ‘data driven’, ‘pattern-recognition’)
Monte Carlo Methods

Stanisław Ulam
Monte Carlo inference

- We want to sample from some distribution
  - In this case, a posterior $p(z|x)$
- We can’t sample from $p$ directly, but maybe we can evaluate it
  - Or maybe we can only evaluate an unnormalised version of it, e.g. $p(z, x)$

Take samples from some other distribution (e.g. prior) and transform/reweight/etc. them so that they become samples from the posterior.
Likelihood Weighting

- Basic example: Sample from prior and weight by likelihood
Likelihood Weighting

- Basic example: Sample from prior and weight by likelihood
Likelihood Weighting

- Basic example: Sample from prior and weight by likelihood
Importance Sampling

- Sample from guide $q(z)$, weight by $w = \frac{p(z,x)}{q(z)}$
Importance Sampling

- Sample from guide $q(z)$, weight by $w = \frac{p(z,x)}{q(z)}$
Importance Sampling

- Sample from guide $q(z)$, weight by $w = \frac{p(z,x)}{q(z)}$
Importance Sampling

- Sample from guide $q(z)$, weight by $w = \frac{p(z,x)}{q(z)}$
Importance Sampling

- Sample from guide $q(z)$, weight by $w = \frac{p(z,x)}{q(z)}$
- Learn $q$ with a neural network
Importance Sampling

Cusumano-Towner et al. (2017)
Importance Sampling

- Works well if you can design, or learn, a guide distribution close to the true posterior
  - Should put a reasonable amount of probability mass on the true posterior
  - Much better to overshoot than undershoot!
- Usually **terrible** in high dimensional spaces
Markov Chain Monte Carlo (MCMC)

Rather than independent samples from a pre-determined guide distribution, take correlated samples that form a Markov Chain.

New sample based on feedback from previous sample.

Those samples are based on a random walk over $Z$, such that the proportion of samples equal to $z^*$ is proportional to $p(z^* | x)$.

Need to meet conditions:

1. Markov chain is **ergodic** (eventually get to any possible $z$).
2. Posterior distribution is **stationary**.
Markov Chain Monte Carlo (MCMC)

E.g. Mixture of Gaussians

\[ z_i \sim \text{Categorical}(\ldots) \]
\[ \mu_j \sim \text{Normal}(\ldots) \]
\[ x_i \sim \text{Normal}(\mu_{z_i}) \]
Markov Chain Monte Carlo (MCMC)

E.g. Mixture of Gaussians

\[ z_i \sim \text{Categorical}(\ldots) \]
\[ \mu_j \sim \text{Normal}(\ldots) \]
\[ x_i \sim \text{Normal}(\mu_{z_i}) \]

“If we knew the zs, we could just sample \( \mu \)” (because \( \mu \) has a conjugate prior)

“If we knew the \( \mu \)s, we could just sample \( z \)” (because \( z \) has a finite prior)
Gibbs Sampling

- If we can’t sample from $p(A,B)$ but we can sample from $p(A|B)$ and $p(B|A)$
- Algorithm:
  1. Initialise A and B
  2. Repeat
     Sample $A \sim p(A|B)$
     Sample $B \sim p(B|A)$
Collapsed Gibbs Sampling

E.g. Mixture of Gaussians

\[ \pi \sim \text{Dirichlet}(...) \quad \text{← cluster proportions} \]
\[ z_i \sim \text{Categorical}(\pi) \]
\[ \mu_j \sim \text{Normal}(...) \]
\[ x_i \sim \text{Normal}(\mu_{z_i}) \]

Dirichlet is \textbf{conjugate prior} for Categorical. So: don’t sample \( \pi \), but marginalise we can evaluate \( p(z|\mu) \) exactly!
Markov Chain Monte Carlo (MCMC)

What if we don’t know the conditional distributions exactly?
Similar idea to importance sampling:
1. Propose from a ‘guide’ distribution, and
2. Accept/reject proposal using feedback from model
Metropolis-Hastings (MH)

General algorithm for obtaining MH samples to approximate $p(z \mid x)$.

1. Initialize $z^0$
2. for $s = 0, 1, 2, \ldots$ do
3.   Define $z = z^s$
4.   Sample $z' \sim q(z' \mid z) \leftarrow$ Proposal distribution (may be a mixture distribution)
5.   Compute acceptance ratio:
   \[ r = \frac{\hat{p}(z', x)q(z \mid z')}{\hat{p}(z, x)q(z' \mid z)} \leftarrow \text{Only an unnormalized posterior } \hat{p} \text{ is necessary} \]
6.   Compute $a = \min(1, r) \leftarrow$ Satisfies “detailed-balance”
7.   Sample $u \sim U(0, 1)$
8.   Set new sample to:
   \[ z^{s+1} = \begin{cases} 
   z' & \text{if } u < a \leftarrow \text{Accept} \\
   z^s & \text{if } u \geq a \leftarrow \text{Reject}
   \end{cases} \]
Some other MCMC ideas (mix ‘n match)

Data-driven proposals  e.g. Tu & Zhu (2002), Kulkarni et al. (2015)
Sample (z, x) pairs from generative model, train neural network q(z|x). Proposal distribution can be a mixture of ‘global’ network proposals and ‘local’ (e.g. gaussian) proposals

Hamilton Monte Carlo animation
Use gradient information, follow the energy landscape

Annealing
e.g. start with a ‘smoothed’ likelihood (“high temperature”) and “cool” to true likelihood

Parallel tempering
Use multiple chains at different temperatures. Proposals jump between chains.
MCMC

Benefits
- Can be applied to any probability distributions including:
  - High dimensionality
  - Discrete variables / unknown dimensionality (probabilistic programs)
- Can use unnormalized posterior

Difficulties
- ‘Burn in’ - Find region of high probability, with only local information
- ‘Mixing’ - How to move between modes?
- Difficult to assess convergence
Particle Filtering / Sequential Monte Carlo (SMC)

When your latent variables and observations form a sequence (so you get feedback over time)

Like importance sampling but reweight and resample at each timestep
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)
Particle Filtering / Sequential Monte Carlo (SMC)

Rejuvenation

MCMC steps to update the whole state sequence
Particle Filtering / Sequential Monte Carlo (SMC)

Neural network proposal functions

- Neural Adaptive Sequential Monte Carlo (Gu et al. 2015)
- Neurally Guided Procedural Models (Ritchie et al. 2016)
Particle Filtering / Sequential Monte Carlo (SMC)

Neurally Guided Procedural Models (Ritchie et al. 2016)
Monte Carlo inference: summary

- Basic idea illustrated with importance sampling

- MCMC algorithms:
  - MH: General purpose
  - Gibbs sampling: when you have exact conditional distributions
  - HMC: when you have the gradients of the (unnormalised) posterior
  - SMC: when you have sequential observations
  - Many, many others!

- Use neural networks to learn proposal distributions
Variational Inference

Leonard Euler  
Joseph-Louis Lagrange
Variational Inference (Inference as optimisation)

Monte-Carlo: Obtain samples from posterior
Variational: Approximate posterior with a parametric distribution (e.g. Gaussian)

- Optimise parameters to most closely match posterior: minimise $D_{KL}(Q\|P)$
- Trade off between ease of optimisation and accuracy of approximation
Matching the true posterior

Kullback-Leibler divergence $KL(Q||P) = \text{measure of distance}^* \text{ between } Q \text{ and } P$

- Non-negative; zero only when $P$ and $Q$ are equal

$$D_{KL}[Q(z) \parallel p(z|x)] = -E_{z \sim Q}[\log \frac{p(z|x)}{Q(z)}]$$

$$= -E_{z \sim Q}[\log p(z,x) - \log Q(z)] + \text{const.}$$

ELBO (evidence lower bound)

Lower bound on $\log p(x)$
Matching the true posterior

How to choose and optimise the variational distribution, Q?

- ‘Classical’ variational inference (1990s, 2000s)
  Find a variational family that you can optimise analytically (with the calculus of variations)

- Stochastic gradient variational Bayes (2014)
  Find a variational family that you can optimise with gradient descent

\[ \mathbb{E}_{z \sim Q} \left[ \log p(z, x) - \log Q(z) \right] \]

ELBO
‘Classical’ variational inference

1. Choose **conjugate** prior distributions in $P$ and choose $Q$ from the same family, so that we will be able to derive analytical solutions.
‘Classical’ variational inference

1. Choose **conjugate** prior distributions in $P$ and choose $Q$ from the same family, so that we will be able to derive analytical solutions.

$P$: Gaussian mixture model

- $\pi \sim \text{Dirichlet}(\ldots)$
- $z_i \sim \text{Categorical}(\pi)$
- $\mu_j \sim \text{Normal}(\ldots)$
- $\Sigma_j \sim \text{InvWishart}(\ldots)$
- $x_i \sim \text{Normal}(\mu_{z_i}, \Sigma_{z_i})$
‘Classical’ variational inference

1. Choose **conjugate** prior distributions in \( P \) and choose \( Q \) from the same family, so that we will be able to derive analytical solutions

\[ P: \text{Gaussian mixture model} \]

\[ \pi \sim \text{Dirichlet}(...) \]
\[ z_i \sim \text{Categorical} (\pi) \]
\[ \mu_j \sim \text{Normal}(...) \]
\[ \Sigma_j \sim \text{InvWishart}(...) \]
\[ x_i \sim \text{Normal}(\mu_{z_i}, \Sigma_{z_i}) \]
‘Classical’ variational inference

1. Choose **conjugate** prior distributions in $P$ and choose $Q$ from the same family, so that we will be able to derive analytical solutions

$P$: Gaussian mixture model

$\pi \sim \text{Dirichlet}(...)$
$z_i \sim \text{Categorical}(\pi)$
$\mu_j \sim \text{Normal}(...)$
$\Sigma_j \sim \text{InvWishart}(...)$
$x_i \sim \text{Normal}(\mu_{z_i}, \Sigma_{z_i})$
'Classical' variational inference

1. Choose **conjugate** prior distributions in \( P \) and choose \( Q \) from the same family, so that we will be able to derive analytical solutions

\[ p(\pi, \mu, z, x) = p(\pi) \prod p(\mu^k) \prod p(z^i|\pi) p(x|\mu^z) \]

**P**: Gaussian mixture model

- \( \pi \sim \text{Dirichlet}(...) \)
- \( z_i \sim \text{Categorical}(\pi) \)
- \( \mu_j \sim \text{Normal}(...) \)
- \( \Sigma_j \sim \text{InvWishart}(...) \)
- \( x_i \sim \text{Normal}(\mu_{z_i}, \Sigma_{z_i}) \)
‘Classical’ variational inference

1. Choose conjugate prior distributions in $P$ and choose $Q$ from the same family, so that we will be able to derive analytical solutions

2. Mean-field approximation: make all latent variables independent in $Q$

$$Q(z_1, ..., z_n) = \prod_{i=1}^{n} q(z_i)$$

$$Q(\pi, \mu, \Sigma, z) = q(\pi) \prod q(\mu_k) q(\Sigma_k) \prod q(z_i)$$
‘Classical’ variational inference

Mean-field variational inference:

“Find the closest match to the posterior without allowing any correlations”
‘Classical’ variational inference

Mean-field variational inference:
“Find the closest match to the posterior without allowing any correlations”
‘Classical’ variational inference

Gaussian mixture model

\[ z_i \sim \text{Categorical}(\pi) \]
\[ \mu_j \sim \text{Normal}(...) \]
\[ x_i \sim \text{Normal}(\mu_{z_i}) \]

Mean field approximation:
\[ Q(z, \mu) = q(z)q(\mu) \]

ELBO
\[ \mathbb{E}_{z \sim q} \left[ \log p(z, x) - \log q(z) \right] \]

\[ \arg\max_{q(z), q(\mu)} \mathbb{E}_{z \sim q(z), \mu \sim q(\mu)} \left[ \log p(z, \mu, x) - \log q(z)q(\mu) \right] \]
‘Classical’ variational inference

\[
\text{ELBO} \quad \mathbb{E}_{z \sim \mathcal{Q}} \left[ \log p(z, x) - \log Q(z) \right]
\]

Gaussian mixture model

\[
z_i \sim \text{Categorical}(\pi) \\
\mu_j \sim \text{Normal}(...)
\]

\[
x_i \sim \text{Normal}(\mu_{z_i})
\]

Mean field approximation:

\[
Q(z, \mu) = \mathbb{Q}(z) \mathbb{Q}(\mu)
\]

\[
\text{Argmax}_{\mathbb{Q}(z), \mathbb{Q}(\mu)} \quad \mathbb{E}_{z \sim \mathbb{Q}(z), \mu \sim \mathbb{Q}(\mu)} \left[ \log p(z, \mu, x) - \log \mathbb{Q}(z) \mathbb{Q}(\mu) \right]
\]

“If we knew \(\mathbb{Q}(z)\) we could find the best \(\mathbb{Q}(\mu)\)”

\[
\mathbb{Q}^*(\mu) \propto \exp \left[ \mathbb{E}_{\mathbb{Q}(z)} \log p(z, \mu, x) \right]
\]

“If we knew \(\mathbb{Q}(\mu)\) we could find the best \(\mathbb{Q}(z)\)”

\[
\mathbb{Q}^*(z) \propto \exp \left[ \mathbb{E}_{\mathbb{Q}(\mu)} \log p(z, \mu, x) \right]
\]

So: Alternate updates to \(\mathbb{Q}(z)\) and \(\mathbb{Q}(\mu)\)
‘Classical’ variational inference

• Algorithm:
  1. Initialise $q(A)$ and $q(B)$
  2. Repeat
     Update $q(A)$ to minimise $D_{KL}(q(A)q(B) \| P(A,B))$
     Update $q(B)$ to minimise $D_{KL}(q(A)q(B) \| P(A,B))$
   Until convergence
“But can’t we just do
$$\text{Argmax}_Q \mathbb{E}_{z \sim Q} [ \log p(z,x) - \log Q(z) ]$$
with gradient descent?”

- Kingma et al. (2014), Rezende et al. (2014)

- Answer: Yes, if we use samples to approximate the expectation (SGD)!
  But only if:
  - Unnormalized posterior $p(z,x)$ is differentiable in $z$ (no discrete random variables, unless you can marginalise them out)
  - $Q(z)$ is ‘reparametrisable’ (e.g. multivariate Normal)

- No need for conjugate priors!
- No need for mean-field approximation!
Stochastic Gradient Variational Bayes

\[ p(z) = \text{Gamma}(z; 1,1) \]
\[ q(z) = \text{Gamma}(z; a, b) \]
\[ p(x|z) \propto \exp[-(z-x)^4] \]

```python
import torch
theta = torch.nn.Parameter(torch.zeros(2))
optimizer = torch.optim.Adam([theta], lr=1e-3)

for i in range(100000):
    # Model
    log_prior = torch.distributions.gamma.Gamma(1,1).log_prob
    log_likelihood = lambda z, x: -(z-x)**4

    # Variational Distribution
    a, b = torch.exp(theta)
    Q = torch.distributions.gamma.Gamma(a, b)

    # Optimise ELBO
    z = Q.sample()
    elbo = log_prior(z) + log_likelihood(z, x=10) - Q.log_prob(z)
    (elbo).backward(); optimizer.step(); optimizer.zero_grad()

print(Q.mean, Q.variance)  # Returns (9.73, 1.18)
```
1. Normalizing flows (Rezende and Mohamed, 2016)
   ○ To make an expressive variational distribution $Q$, start with a simple distribution (e.g. Normal) and then run it through a bunch of invertible transformations.

2. Amortized variational distribution
   ○ Use a neural network $f_\theta$ to parametrize $Q(z)$
     Variational Autoencoders (Kingma et al, 2014)
   ○ Afterwards, you not only solve your inference problem, but you also have a recognition model for future observations

\[
q(z_i) = \text{Normal}(z_i; f_\theta(x_i))
\]
Variational inference: summary

Can mix methods: some latents stochastic, others analytical (e.g. Belief Propagation)

**Benefits**
- Fast and easy to assess convergence
- Provides model evidence log P(X)
- Normalising flows: Can handle multimodality

**Limitations**
- Unlike MH, restrictions on model:
  - Classical VI: Mean-field approximation and conjugate priors
    Often can’t express multimodal posteriors
  - SGVB: everything has to be differentiable/enumerable
- Unlike MCMC, not exact in the limit, convergence $\neq$ exact posterior
Probabilistic Programming Languages

Programming languages which contain useful abstractions and functions for defining generative models and performing inference in them.

<table>
<thead>
<tr>
<th>PPL</th>
<th>Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stan (Carpenter, Gelman et al.)</td>
<td>Optimised for graphical models</td>
</tr>
<tr>
<td>+  Edward (Tran, Blei et al.)</td>
<td>+  Deep generative models, SVI</td>
</tr>
<tr>
<td>WebPPL (Goodman &amp; Stuhlmüller)</td>
<td>Universal (dynamic computation graph)</td>
</tr>
<tr>
<td>+  Pyro (Bingham, Goodman et al.)</td>
<td>+  Deep generative models, SVI</td>
</tr>
<tr>
<td>Gen (Cusumano-Towner &amp; Mansinghka)</td>
<td>Programmable Inference</td>
</tr>
</tbody>
</table>

Learn ourselves Pyro - This Thursday @ 2:30pm, 46-5165
Exercises: Bayesian polynomial regression

\[ y \sim \text{Normal}(\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3, \sigma^2) \]

Inference exercises
1. Metropolis Hasting sampler
2. ‘Classical’ Variational Inference
3. Stochastic Gradient Variational Bayes
Defining the model

Fix prior parameters $a_0, b_0, \mu_0$.

$\tau \sim \text{Gamma}(a_0, b_0) \quad \rightarrow \quad \sigma^2 = 1/\tau$

$\beta_k \sim \text{Normal}(\mu_0, 1/\tau)$

$y \sim \text{Normal}(\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3, 1/\tau)$

$\mathbf{X}$ $\rightarrow$ $\mathbf{y}$ $\rightarrow$ $\tau$ $\rightarrow$ $\beta_k$ $\rightarrow$ $k = 1..K$
Exact inference

For details see: https://en.wikipedia.org/wiki/Bayesian_linear_regression
Metropolis-Hastings (MH)

1. **propose** samples z’ and computes these probabilities:
   a. q(z’ | z) and q(z | z’)
   b. p(z’) and p(z)

2. **propose** requires
   transition_dist q(z’ | z)
   a. p(z) is already defined for you
Metropolis-Hastings (MH)

1. Initialize $z^0$
2. for $s = 0, 1, 2, ...$ do
   3. Define $z = z^s$
   4. Sample $z' \sim q(z'|z)$
   5. Compute acceptance ratio:
      $r = \frac{\hat{p}(z', x)q(z'|z)}{\hat{p}(z, x)q(z'|z)}$
   6. Compute $a = \min(1, r)$
   7. Sample $u \sim U(0, 1)$
   8. Set new sample to:
      $z^{s+1} = \begin{cases} 
      z' \text{ if } u < a & \leftarrow \text{Accept} \\
      z^s \text{ if } u \geq a & \leftarrow \text{Reject}
      \end{cases}$

3. $p(z', x)$ needs to be calculated
   a. $p(z', x) = p(z')p(x | z)$
   b. E.g., $p(z, x)$

4. Combine outputs of propose and unnormalized posteriors to calculate acceptance ratio

5. Apply Metropolis-Hastings to accept or reject the sample

6. Do inference!
Classic Variational Inference

1. Write out the mean-field approximation
2. Derive the evidence lower bound (ELBO)
3. Derive analytical latent variable updates by either:
   a. Directly optimizing the ELBO
   b. Through the calculus of variations
4. Iterate updating each latent variable
Classic Variational Inference

1. Write out the mean-field approximation

**Model P**
\[ \tau \sim \text{Gamma}(a_0, b_0) \quad \rightarrow \sigma^2 = 1/\tau \]
\[ \beta_k \sim \text{Normal}(\mu_0, 1/\tau) \]
\[ y \sim \text{Normal}(\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3, 1/\tau) \]

**Variational Posterior Q**
\[ Q(\tau, \beta) = Q(\tau) \prod^K Q(\beta_k) \]
\[ \tau \sim \text{Gamma}(a, b) \]
\[ \beta_k \sim \text{Normal}(\nu_k, 1/\omega_k) \]
2. Derive the evidence lower bound (ELBO)

\[
\log P(x) \geq \mathbb{E}_{\tau \sim Q} \left[ \log P(\tau) - \log Q(\tau) \right] \\
+ \sum_{k=1}^{K} \mathbb{E}_{\tau, \beta_k \sim Q} \left[ \log P(\beta_k \mid \tau) - \log Q(\beta_k \mid \tau) \right] \\
+ \sum_{i=1}^{N} \mathbb{E}_{\tau, \beta \sim Q} \left[ \log P(y_i \mid \beta, x_i, \tau) \right]
\]
Classic Variational Inference

2. Derive the evidence lower bound (ELBO)

To compute the first term:

\[
E_{\tau \sim Q} \left[ \log P(\tau) - \log Q(\tau) \right] = -D_{KL}(Q_\tau || P_\tau)
\]

**Gamma distribution**
Classic Variational Inference

2. Derive the evidence lower bound (ELBO)

To compute the first term:

\[
\mathbb{E}_{\tau \sim Q} \left[ \log P(\tau) - \log Q(\tau) \right] = -D_{KL}(Q_\tau || P_\tau)
\]

**Gamma distribution**

\[
= (a_0 - a) \psi(a) + \log \Gamma(a) - \log \Gamma(a_0)
\]

\[
- a_0 (\log(b) - \log(b_0)) - a \left( \frac{b_0 - b}{b} \right)
\]
Classic Variational Inference

2. Derive the evidence lower bound (ELBO)

To compute the second term:

$$\mathbb{E}_{\beta, \tau \sim Q} \left[ \log P(\beta_k) - \log Q(\beta_k) \right] = \mathbb{E}_{\tau \sim Q} \left[ -D_{KL}(Q_{\beta_k|\tau} || P_{\beta_k|\tau}) \right]$$

**Gamma distribution**

**Normal distribution**
Classic Variational Inference

2. Derive the evidence lower bound (ELBO)

To compute the second term:

$$\mathbb{E}_{\beta, \tau \sim Q} \left[ \log P(\beta_k) - \log Q(\beta_k) \right] = \mathbb{E}_{\tau \sim Q} \left[ -D_{KL}(Q_{\beta_k | \tau} \parallel P_{\beta_k | \tau}) \right]$$

\[= -\frac{1}{2} \left[ \frac{a(v_k - \mu_0)^2}{b} + \left( \frac{a}{\omega_k b} - 1 - \Psi(a) + \log(b) + \log(\omega_k) \right) \right] \]

**Gamma distribution**

**Normal distribution**
2. Derive the evidence lower bound (ELBO)

We won’t compute the third term today, it has long (but manageable) integrals.

\[
\mathbb{E}_{\tau, \beta \sim Q} \left[ \log P(y_i|\beta, x_i, \tau) \right] = \mathbb{E}_{\tau \sim Q} \left[ \mathbb{E}_{\beta \sim Q} \left( \log P(y_i|\beta, x_i, \tau) \right) \right] \\
= \mathbb{E}_{\tau \sim Q} \left[ \frac{1}{2} (\log(\tau) - \log(2\pi)) - \frac{y_i^2 \tau}{2} + y_i \tau \sum_{k=0}^{K} x_{ik} \nu_k \right] \\
- \frac{\tau}{2} \left[ \sum_{k=0}^{K} x_{ik}^2 \left( \frac{1}{\omega_k} + \nu_k^2 \right) + 2 \sum_{k=0}^{K} \sum_{j=0}^{k-1} x_{ik} x_{ij} \nu_k \nu_j \right] \\
= \frac{1}{2} (\Psi(a) - \log(b) - \log(2\pi)) - \frac{y_i^2 a}{2b} + \frac{y_i a}{b} \sum_{k=0}^{K} x_{ik} \nu_k \\
- \frac{a}{2b} \left[ \sum_{k=0}^{K} x_{ik}^2 \left( \frac{1}{\omega_k} + \nu_k^2 \right) + 2 \sum_{k=0}^{K} \sum_{j=0}^{k-1} x_{ik} x_{ij} \nu_k \nu_j \right]
\]
Classic Variational inference

3. Derive analytical latent variable updates by:
   a. Directly optimizing the ELBO

   Take the partial derivatives of the ELBO with respect to $\nu_k$ and $\omega_k$ to derive each of their updates. (Remember to sum over N and K in deriving the entire ELBO!)

This will not lead to closed form solutions for a and b, though.
Classic Variational inference

3. Derive analytical latent variable updates by:
   b. Through the calculus of variations (we won’t do this today)

   \[ Q(\tau) \propto \exp \left( \mathbb{E}_{\beta \sim Q} \left[ \log P(\tau, \beta, y, x) \right] \right) \]

   \[ \log Q(\tau) = \mathbb{E}_{\beta \sim Q} \left[ \log(P(\tau)) + \log(P(\beta|\tau)) \log(P(y|\beta, \tau, x)) \right] + \text{Const}. \]

   Doing these expectations and folding anything constant with respect to \( \tau \) into \( \text{Const} \) eventually gets you to an expression with the form:

   \[ \log Q(\tau) = f_1(\nu, \omega, y, x) \log(\tau) - f_2(\nu, \omega, y, x) \tau + \text{Const}. \]

   This is log of a Gamma distribution with \( a = f_1 \) & \( b = f_2 \), which are the updates.
Classic Variational inference

4. Enter your iterative updates and do inference!
Stochastic Gradient Variational Bayes

The ELBO is:
\[
E \left[ \log p(\tau, \beta, y | x) - \log q(\tau)q(\beta) \right]
\]

For each SGD iteration, we need a unbiased (monte-carlo) estimate of the ELBO, so:

1. Sample \((\tau, \beta)\) from \(q\)
   - Note: use \texttt{dist.rsampole()} for a ‘reparametrized’ (differentiable) sample
2. Evaluate the term inside the expectation
3. Repeat until convergence!