Variational Inference and Generative Models

CS 285

Instructor: Sergey Levine
UC Berkeley
Today’s Lecture

1. Probabilistic latent variable models
2. Variational inference
3. Amortized variational inference
4. Generative models: variational autoencoders
   • Goals
     • Understand latent variable models in deep learning
     • Understand how to use (amortized) variational inference
Probabilistic models

\[ p(x) \]

\[ p(y|x) \]

\[ \pi_\theta(a|s) \]
Latent variable models

\[ p(x) = \sum_z p(x|z)p(z) \]

\[ p(y|x) = \sum_z p(y|x,z)p(z) \]
Latent variable models in general

\[ p(x) = \int p(x|z)p(z)dz \]

“easy” distribution (e.g., conditional Gaussian)

“easy” distribution (e.g., Gaussian)

\[ p(x|z) = \mathcal{N}(\mu_{nn}(z), \sigma_{nn}(z)) \]

“easy” distribution (e.g., Gaussian)
Latent variable models in RL

conditional latent variable models for multi-modal policies

Latent variable models for model-based RL

$p(z)$

$z \sim \mathcal{N}(0, I)$

$p(y|x, z)$

$p(o_t|x_t)$ actually models $p(x_{t+1}|x_t)$ and $p(x_1)$

$p(x_t)$ latent space has structure
Other places we’ll see latent variable models

Using RL/control + variational inference to model human behavior

Using generative models and variational inference for exploration
How do we train latent variable models?

the model: \( p_\theta(x) \)

the data: \( \mathcal{D} = \{x_1, x_2, x_3, \ldots, x_N\} \)

maximum likelihood fit:

\[
\theta \leftarrow \arg \max_{\theta} \frac{1}{N} \sum_i \log p_\theta(x_i)
\]

\[
p(x) = \int p(x|z)p(z)dz
\]

\[
\theta \leftarrow \arg \max_{\theta} \frac{1}{N} \sum_i \log \left( \int p_\theta(x_i|z)p(z)dz \right)
\]

completely intractable
Estimating the log-likelihood

alternative: expected log-likelihood:

$$\theta \leftarrow \arg \max_\theta \frac{1}{N} \sum_i E_{z \sim p(z|x_i)} [\log p_\theta(x_i, z)]$$

but... how do we calculate $p(z|x_i)$?

intuition: “guess” most likely $z$ given $x_i$, and pretend it’s the right one
...but there are many possible values of $z$ so use the distribution $p(z|x_i)$
Variational Inference
The variational approximation

but... how do we calculate $p(z|x_i)$?

can bound $\log p(x_i)$!

\[
\log p(x_i) = \log \int_z p(x_i | z)p(z) \\
= \log \int_z p(x_i | z)p(z) \frac{q_i(z)}{q_i(z)} \\
= \log E_{z \sim q_i(z)} \left[ \frac{p(x_i | z)p(z)}{q_i(z)} \right]
\]

what if we approximate with $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$
The variational approximation

but... how do we calculate \( p(z|x_i) \)?

can bound \( \log p(x_i) \):

\[
\log p(x_i) = \log \int_z p(x_i|z)p(z)
\]

\[
= \log \int_z p(x_i|z)p(z) \frac{q_i(z)}{q_i(z)}
\]

\[
= \log E_{z \sim q_i(z)} \left[ \frac{p(x_i|z)p(z)}{q_i(z)} \right]
\]

maximizing this maximizes \( \log p(x_i) \)

\[
\geq E_{z \sim q_i(z)} \left[ \log \frac{p(x_i|z)p(z)}{q_i(z)} \right] = E_{z \sim q_i(z)} [\log p(x_i|z) + \log p(z)] + \mathcal{H}_q(q_i(z)) [\log q_i(z)]
\]
A brief aside...

Entropy:

$$\mathcal{H}(p) = -E_{x \sim p(x)}[\log p(x)] = - \int p(x) \log p(x) dx$$

Intuition 1: how random is the random variable?
Intuition 2: how large is the log probability in expectation *under itself*

what do we expect this to do?

$$E_{z \sim q_i(z)}[\log p(x_i | z) + \log p(z)] + \mathcal{H}(q_i)$$

this maximizes the first part

this also maximizes the second part (makes it as wide as possible)
A brief aside...

KL-Divergence:

$$D_{KL}(q\|p) = E_{x \sim q(x)} \left[ \log \frac{q(x)}{p(x)} \right] = E_{x \sim q(x)} [\log q(x)] - E_{x \sim q(x)} [\log p(x)] = -E_{x \sim q(x)} [\log p(x)] - \mathcal{H}(q)$$

Intuition 1: how **different** are two distributions?

Intuition 2: how small is the expected log probability of one distribution under another, **minus entropy**?

why entropy?

![Diagram showing p(z) distribution with labels](image)
The variational approximation

\[ \mathcal{L}_i(p, q_i) \]

\[ \log p(x_i) \geq E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i) \]

what makes a good \( q_i(z) \)?

intuition: \( q_i(z) \) should approximate \( p(z|x_i) \)

approximate in what sense?

compare in terms of KL-divergence: \( D_{KL}(q_i(z)||p(z|x)) \)

why?

\[ D_{KL}(q_i(x_i)||p(z|x_i)) = E_{z \sim q_i(z)} \left[ \log \frac{q_i(z)}{p(z|x_i)} \right] = E_{z \sim q_i(z)} \left[ \log \frac{q_i(z)p(x_i)}{p(x_i, z)} \right] \]

\[ = -E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] + E_{z \sim q_i(z)}[\log q_i(z)] + E_{z \sim q_i(z)}[\log p(x_i)] \]

\[ = -E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] - \mathcal{H}(q_i) + \log p(x_i) \]

\[ = -\mathcal{L}_i(p, q_i) + \log p(x_i) \]

\[ \log p(x_i) = D_{KL}(q_i(z)||p(z|x_i)) + \mathcal{L}_i(p, q_i) \]

\[ \log p(x_i) \geq \mathcal{L}_i(p, q_i) \]
The variational approximation

\[
\mathcal{L}_i(p, q_i)
\]

\[
\log p(x_i) \geq E_{z \sim q_i(z)}[\log p(x_i | z) + \log p(z)] + \mathcal{H}(q_i)
\]

\[
\log p(x_i) = D_{KL}(q_i(z) || p(z | x_i)) + \mathcal{L}_i(p, q_i)
\]

\[
\log p(x_i) \geq \mathcal{L}_i(p, q_i)
\]

\[
D_{KL}(q_i(z) || p(z | x_i)) = E_{z \sim q_i(z)} \left[ \log \frac{q_i(z)}{p(z | x_i)} \right] = E_{z \sim q_i(z)} \left[ \log \frac{q_i(z)p(x_i)}{p(x_i, z)} \right]
\]

\[
= -E_{z \sim q_i(z)}[\log p(x_i | z) + \log p(z)] - \mathcal{H}(q_i) + \log p(x_i)
\]

\[
-\mathcal{L}_i(p, q_i)
\]

independent of \(q_i\)!

\[\Rightarrow\] maximizing \(\mathcal{L}_i(p, q_i)\) w.r.t. \(q_i\) minimizes KL-divergence!
How do we use this?

\[ \mathcal{L}_i(p, q_i) \]

\[
\log p(x_i) \geq E_{z \sim q_i(z)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_i)
\]

\[
\theta \leftarrow \arg \max_\theta \frac{1}{N} \sum_i \log p_\theta(x_i)
\]

for each \( x_i \) (or mini-batch):

- calculate \( \nabla_\theta \mathcal{L}_i(p, q_i) \):
  - sample \( z \sim q_i(z) \)
  - \( \nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z) \)
  - \( \theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i) \)
  - update \( q_i \) to maximize \( \mathcal{L}_i(p, q_i) \)

\[
\theta \leftarrow \arg \max_\theta \frac{1}{N} \sum_i \mathcal{L}_i(p, q_i)
\]

let’s say \( q_i(z) = \mathcal{N}(\mu_i, \sigma_i) \)

use gradient \( \nabla_\mu_i \mathcal{L}_i(p, q_i) \) and \( \nabla_\sigma_i \mathcal{L}_i(p, q_i) \)

gradient ascent on \( \mu_i, \sigma_i \)
What’s the problem?

for each $x_i$ (or mini-batch):

- calculate $\nabla_\theta \mathcal{L}_i(p, q_i)$:
  - sample $z \sim q_i(z)$
  - $\nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z)$
  - $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i)$
  - update $q_i$ to maximize $\mathcal{L}_i(p, q_i)$

let’s say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on $\mu_i, \sigma_i$

How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

intuition: $q_i(z)$ should approximate $p(z|x_i)$

what if we learn a network $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

$z \quad p_\theta(x|z) \quad x \quad q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$
Amortized Variational Inference
What’s the problem?

for each $x_i$ (or mini-batch):

- calculate $\nabla_\theta \mathcal{L}_i(p, q_i)$:
  - sample $z \sim q_i(z)$
  - $\nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z)$
  - $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i)$
  - update $q_i$ to maximize $\mathcal{L}_i(p, q_i)$

let’s say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on $\mu_i, \sigma_i$

How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

intuition: $q_i(z)$ should approximate $p(z|x_i)$ what if we learn a network $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

$z \quad p_\theta(x|z) \quad x \quad q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$
Amortized variational inference

for each $x_i$ (or mini-batch):

1. calculate $\nabla_\theta \mathcal{L}(p_\theta(x_i|z), q_\phi(z|x_i))$:
   - sample $z \sim q_\phi(z|x_i)$
   - $\nabla_\theta \mathcal{L} \approx \nabla_\theta \log p_\theta(x_i|z)$
   - $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}$
   - $\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}$

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$

$$\mathcal{L}(p_\theta(x_i|z), q_\phi(z|x_i))$$

$$\log p(x_i) \geq E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_\phi(z|x_i))$$

how do we calculate this?
Amortized variational inference

for each $x_i$ (or mini-batch):

calculate $\nabla_\theta \mathcal{L}(p_\theta(x_i \mid z), q_\phi(z \mid x_i))$:

sample $z \sim q_\phi(z \mid x_i)$

$\nabla_\theta \mathcal{L} \approx \nabla_\theta \log p_\theta(x_i \mid z)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}$

$q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$

$\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}$

look up formula for entropy of a Gaussian

$\mathcal{L}_i = E_{z \sim q_\phi(z \mid x_i)}[\log p_\theta(x_i \mid z) + \log p(z)] + \mathcal{H}(q_\phi(z \mid x_i))$

$J(\phi) = E_{z \sim q_\phi(z \mid x_i)}[r(x_i, z)]$

What’s wrong with this gradient?

$\nabla J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi \log q_\phi(z_j \mid x_i) r(x_i, z_j)$

can just use policy gradient!
The reparameterization trick

Is there a better way?

\[ J(\phi) = E_{z \sim q_\phi(z|x_i)}[r(x_i, z)] \]
\[ q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x)) \]
\[ = E_{\epsilon \sim \mathcal{N}(0,1)}[r(x_i, \mu_\phi(x_i) + \epsilon \sigma_\phi(x_i))] \]
\[ z = \mu_\phi(x) + \epsilon \sigma_\phi(x) \]

estimating \( \nabla_\phi J(\phi) \):

sample \( \epsilon_1, \ldots, \epsilon_M \) from \( \mathcal{N}(0,1) \) \quad (a single sample works well!)

\[ \nabla_\phi J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi r(x_i, \mu_\phi(x_i) + \epsilon_j \sigma_\phi(x_i)) \]

most autodiff software (e.g., TensorFlow) will compute this for you!
Another way to look at it...

\[ L_i = E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)] + H(q_\phi(z|x_i)) \]

\[ = E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] + E_{z \sim q_\phi(z|x_i)}[\log p(z)] + H(q_\phi(z|x_i)) \]

\[ -D_{KL}(q_\phi(z|x_i)||p(z)) \]

\[ = E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)||p(z)) \]

\[ = E_{\epsilon \sim \mathcal{N}(0,1)}[\log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i))] - D_{KL}(q_\phi(z|x_i)||p(z)) \]

\[ \approx \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{KL}(q_\phi(z|x_i)||p(z)) \]

\[ \phi \quad \mu_\phi(x_i) \quad \sigma_\phi(x_i) \quad \epsilon \sim \mathcal{N}(0,1) \quad \theta \quad p_\theta(x_i|z) \]
Reparameterization trick vs. policy gradient

- **Policy gradient**
  - Can handle both discrete and continuous latent variables
  - High variance, requires multiple samples & small learning rates

- **Reparameterization trick**
  - Only continuous latent variables
  - Very simple to implement
  - Low variance

\[ J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi \log q_\phi(z_j|x_i) r(x_i, z_j) \]

\[ \nabla_\phi J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi r(x_i, \mu_\phi(x_i) + \epsilon_j \sigma_\phi(x_i)) \]
Variational Inference in Deep RL
The variational autoencoder

\[ q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x)) \]

\[ p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z)) \]

\[ x_i \xrightarrow{\phi} \mu_\phi(x_i) \xrightarrow{\sigma_\phi(x_i)} \mu_\phi(x_i) + \epsilon \sigma_\phi(x_i) = z \xrightarrow{\theta} p_\theta(x|z) \]

\[ \max_{\theta, \phi} \frac{1}{N} \sum_i \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon \sigma_\phi(x_i)) - D_{KL}(q_\phi(z|x_i)||p(z)) \]
Using the variational autoencoder

\[ q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x)) \]
\[ p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z)) \]

\[ p(x) = \int p(x|z)p(z)dz \]

why does this work?

\[ \mathcal{L}_i = E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)\parallel p(z)) \]

sampling:
\[ z \sim p(z) \]
\[ x \sim p(x|z) \]
Example applications

Representation learning

$z$ is a representation of $s$

1. Train VAE on states in replay buffer $\mathcal{R}$
2. Run RL, using $z$ as the state instead of $s$

Why is this a good idea?

Sample algorithm

1. Collect transition $(s, a, s', r)$, add to $\mathcal{R}$
2. Update $p_\theta(s|z)$ and $q_\phi(z|s)$ w/ batch from $\mathcal{R}$
3. Update $Q(z, a)$ w/ batch from $\mathcal{R}$

This also provides a great way to use prior data!

Higgins et al., 2017
Conditional models

\[ \mathcal{L}_i = E_{z \sim q_\Phi(z|x_i,y_i)} [\log p_\Theta(y_i|x_i,z) + \log p(z|x_i)] + \mathcal{H}(q_\Phi(z|x_i,y_i)) \]

just like before, only now generating \( y_i \)
and everything is conditioned on \( x_i \)
at test time:

\[ z \sim p(z|x_i) \]
\[ y \sim p(y|x_i,z) \]

can optionally depend on \( x \)

\[ x_i \quad \phi \quad \mu_\Phi(x_i,y_i) \quad \sigma_\Phi(x_i,y_i) \quad \mu_\Phi + \epsilon\sigma_\Phi = z \quad \epsilon \sim \mathcal{N}(0,1) \quad p_\Theta(y_i|x_i,z) \]

\[ p(y|x,z) \]

\[ z \sim \mathcal{N}(0,1) \]

\[ p(z) \]
Example applications

Multimodal imitation learning

\[ p(a|s, z) \]

\[ z \sim \mathcal{N}(0, I) \]

\[ p(z) \]
Example applications

Multimodal imitation learning

Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

Tony Z. Zhao, Vikash Kumar, Sergey Levine, Chelsea Finn
Stanford University, UC Berkeley, Meta
State space models

- What is the prior? $p(z) = p(z_1) \prod_t p(z_{t+1}|z_t, a_t)$
- What is the decoder? $p_\theta(o|z) = \prod_t p(o_t|z_t)$
- What is the encoder? $q_\phi(z|o) = \prod_t q_\phi(z_t|o_{1:t})$

We are not in partially observed setting learned $\mathcal{N}(0, I)$
State space models

What is the decoder?

\[ p_\theta(o|z) = \prod_t p(o_t|z_t) \]

What is the encoder?

\[ q_\phi(z|o) = \prod_t q_\phi(z_t|o_{1:t}) \]

What is the encoder?
Example applications

Representation learning and model-based RL

1. Learn state space model and plan in the latent space

Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images

Manuel Watter* Editorial Tobias Springenberg* Joschka Boedecker
University of Freiburg, Germany

Martin Riedmiller
Google DeepMind
London, UK

Swing-up with the E2C algorithm

Learning Latent Dynamics for Planning from Pixels

Danijar Hafner 1 2 Timothy Lillicrap 3 4 Ian Fischer 1 Ruben Villegas 1 5
David Ha 1 Honglak Lee 1 James Davidson 1
(c) Cheetah (d) Finger (e) Cup (f) Walker
Example applications

Representation learning and model-based RL

1. Learn state space model and run RL in the state space

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

true rollouts

samples