Variational Inference and Generative Models

CS 285

Instructor: Sergey Levine
UC Berkeley

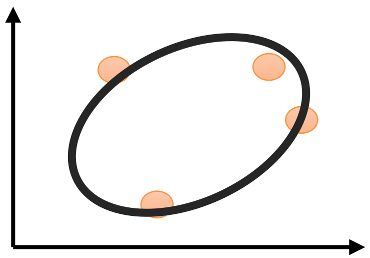
Today's Lecture

1. Probabilistic latent variable models
2. Variational inference
3. Amortized variational inference
4. Generative models: variational autoencoders

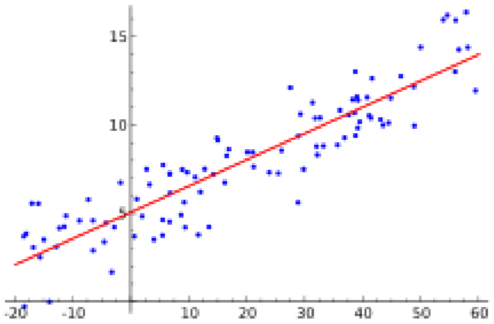
- Goals
  - Understand latent variable models in deep learning
  - Understand how to use (amortized) variational inference

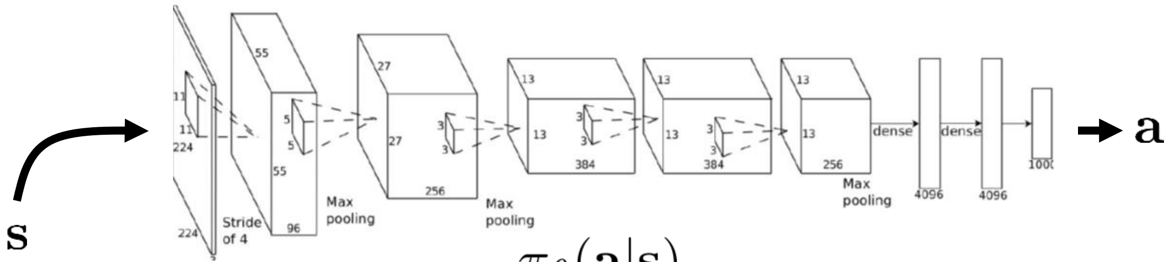
Probabilistic models

$p(x)$



$p(y|x)$



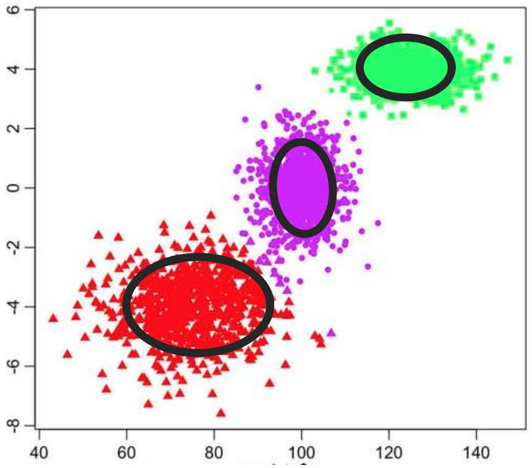


$\pi_\theta(\mathbf{a}|\mathbf{s})$

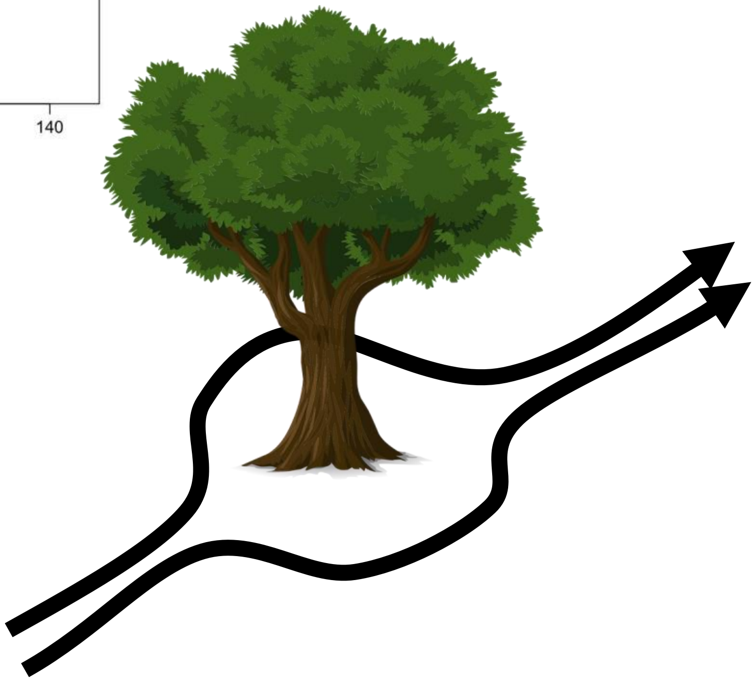
Latent variable models

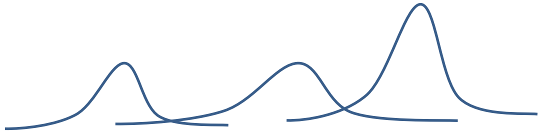
$$p(x) = \sum_z p(x|z)p(z)$$

mixture element

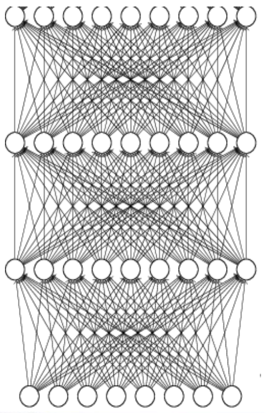


$$p(y|x) = \sum_z p(y|x,z)p(z)$$





$$w_1, \mu_1, \Sigma_1, \ldots, w_N, \mu_N, \sigma_N$$

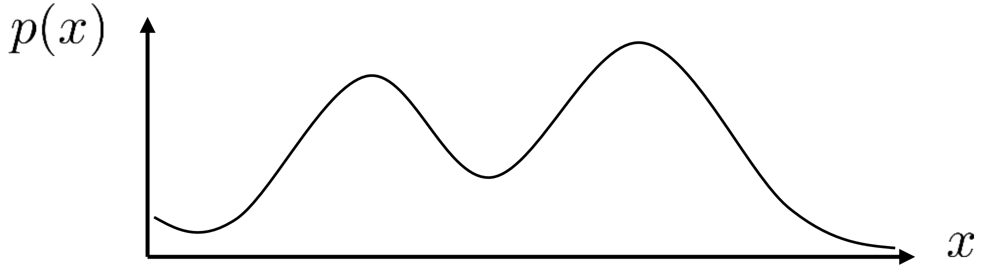


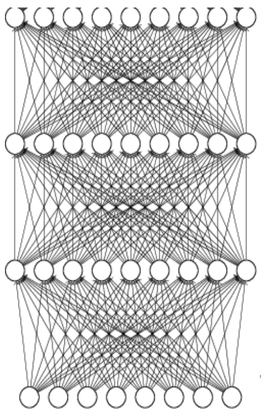
Latent variable models in general

$$p(x) = \int p(x|z)p(z)dz$$

"easy" distribution
(e.g., conditional Gaussian)

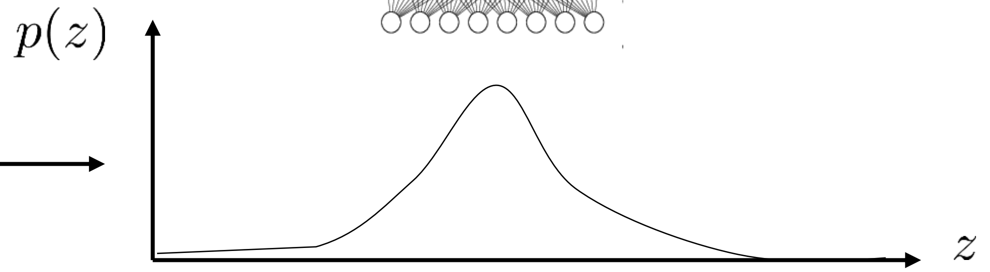
"easy" distribution
(e.g., Gaussian)





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

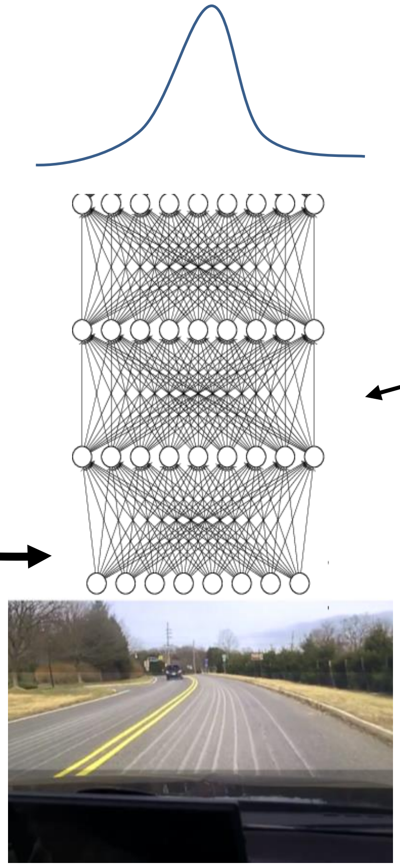
"easy" distribution
(e.g., Gaussian)



Latent variable models in RL

conditional latent variable models for multi-modal policies

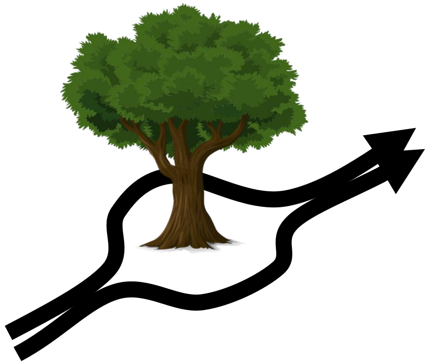
latent variable models for model-based RL

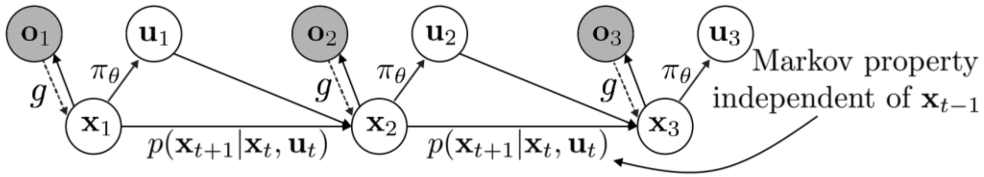


$p(z)$

$p(y|x, z)$

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



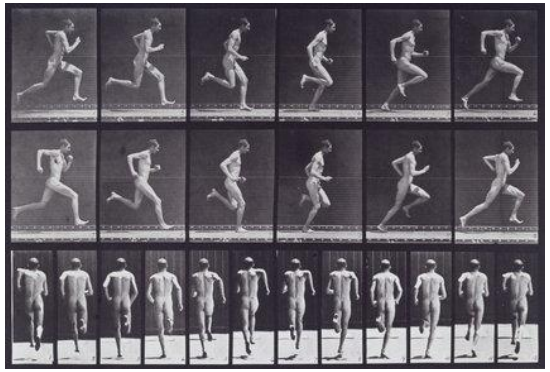


$p(o_t|x_t)$ actually models $p(x_{t+1}|x_t)$ and $p(x_1)$

$p(x_t)$ latent space has *structure*

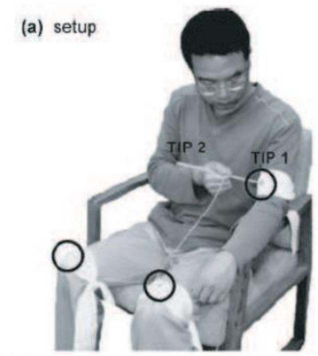
Other places we'll see latent variable models

Using RL/control + variational inference to model human behavior



Muybridge (c. 1870)

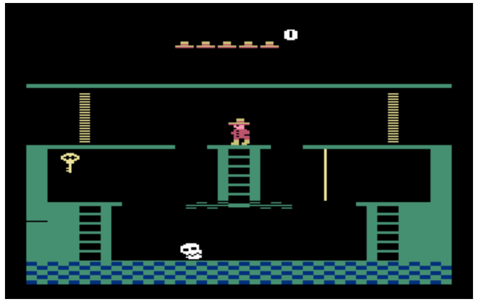
Mombaur et al. '09



Li & Todorov '06

Ziebart '08

Using generative models and variational inference for exploration



How do we train latent variable models?

the model: $p_\theta(x)$

the data: $\mathcal{D} = \{x_1, x_2, x_3, \dots, x_N\}$

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

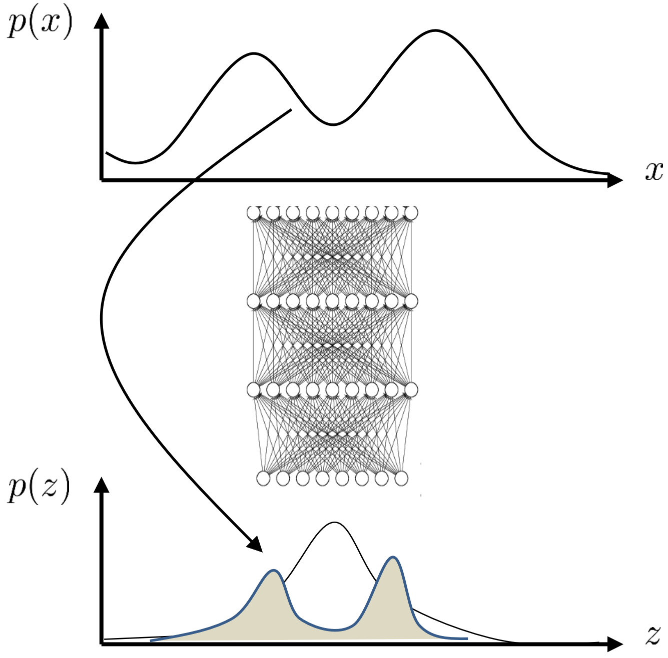
alternative: *expected* log-likelihood:

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

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 $p(z|x_i)$?

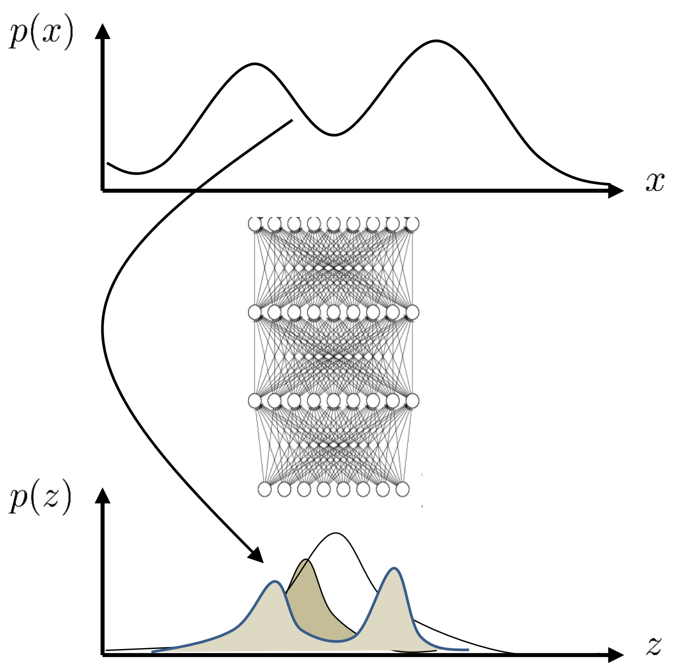
can bound $\log p(x_i)$!

$$\log p(x_i) = \log \int_z p(x_i|z)p(z)$$

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

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

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 $\log p(x_i)$!

$$\log p(x_i) = \log \int_z p(x_i|z)p(z)$$

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

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

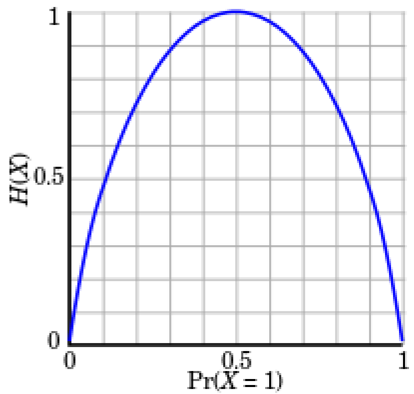
maximizing this maximizes $\log p(x_i)$

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

Jensen's inequality

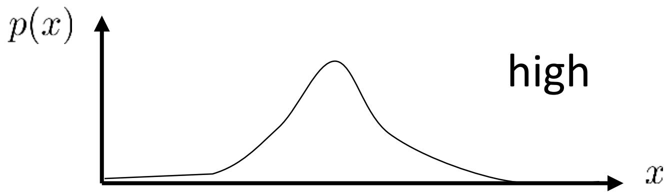
$$\log E[y] \geq E[\log y]$$

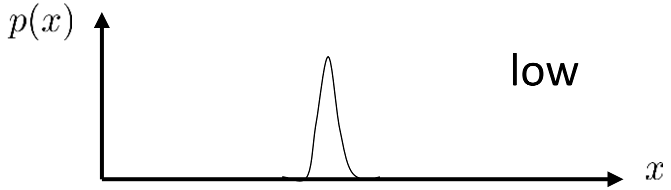
A brief aside…



Entropy:

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





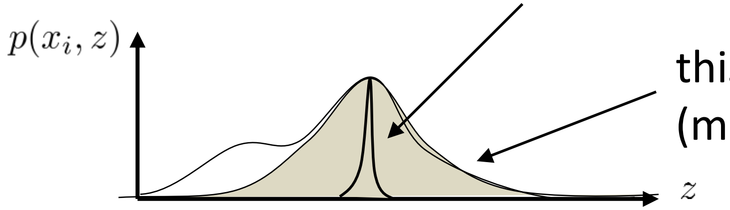
Intuition 1: how *random* is the random variable?

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:

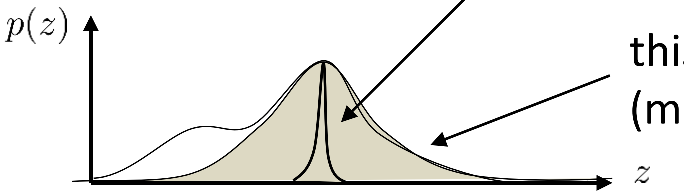
$$D_{\mathrm{KL}}(q\|p) = E_{x\sim q(x)}\left[\log\frac{q(x)}{p(x)}\right] = E_{x\sim q(x)}[\log q(x)] - E_{x\sim q(x)}[\log p(x)] = -E_{x\sim q(x)}[\log p(x)] - \mathcal{H}(q)$$

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?

this maximizes the first part



this also maximizes the second part
(makes it as wide as possible)

The variational approximation

$$\log p(x_i) \geq \overbrace{E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}^{\mathcal{L}_i(p, q_i)}$$

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: $D_{\mathrm{KL}}(q_i(z)\|p(z|x))$

why?

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

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

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

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

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

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

The variational approximation

$$\log p(x_i) \geq \overbrace{E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}^{\mathcal{L}_i(p,q_i)}$$

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

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

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

$$= \underbrace{-E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] - \mathcal{H}(q_i)}_{-\mathcal{L}_i(p,q_i)} + \log p(x_i)$$

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?

$$\log p(x_i) \geq \underbrace{E_{z\sim q_i(z)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}^{\mathcal{L}_i(p, q_i)}$$

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

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

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 $\mathcal{L}_i(p, q_i)$

how?

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 μ_i, σ_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 $\mathcal{L}_i(p, q_i)$

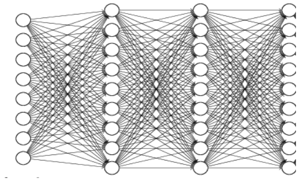
let’s say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

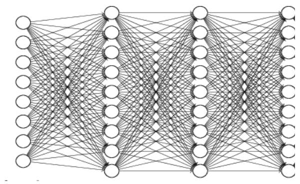
use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on μ_i, σ_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  $p_\theta(x|z)$

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

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 $\mathcal{L}_i(p, q_i)$

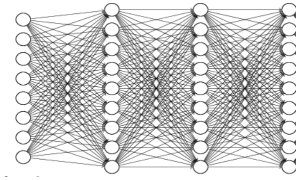
let’s say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

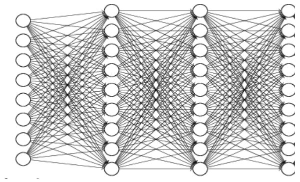
use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on μ_i, σ_i

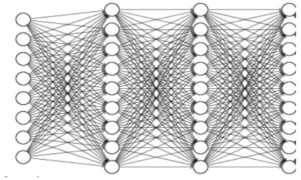
How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

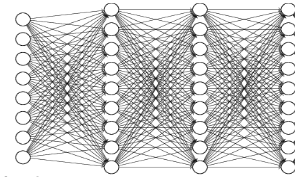
intuition: $q_i(z)$ should approximate $p(z|x_i)$ what if we learn a *network* $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

z  $p_\theta(x|z)$

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

Amortized variational inference

z  $p_\theta(x|z)$

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

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

for each x_i (or mini-batch):

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

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

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

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

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

how do we calculate this?

Amortized variational inference

for each x_i (or mini-batch):

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

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

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

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

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

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

look up formula for entropy of a Gaussian

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

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

can just use policy gradient!

What's wrong with this gradient?

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

The reparameterization trick

Is there a better way?

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

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

$$z = \mu_\phi(x) + \epsilon\sigma_\phi(x)$$

$$\epsilon \sim \mathcal{N}(0,1)$$

independent of ϕ!

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

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

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

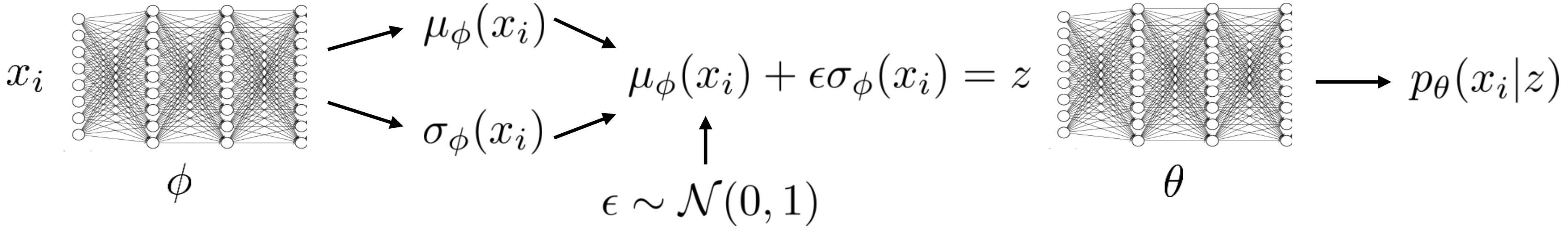
most autodiff software (e.g., TensorFlow) will compute this for you!

Another way to look at it...

$$
\begin{aligned}
\mathcal{L}_i &= E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_\phi(z|x_i)) \\
&= E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] + \underbrace{E_{z\sim q_\phi(z|x_i)}[\log p(z)] + \mathcal{H}(q_\phi(z|x_i))}_{-D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))}
\end{aligned}
$$

this often has a convenient analytical form (e.g., KL-divergence for Gaussians)

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



Reparameterization trick vs. policy gradient

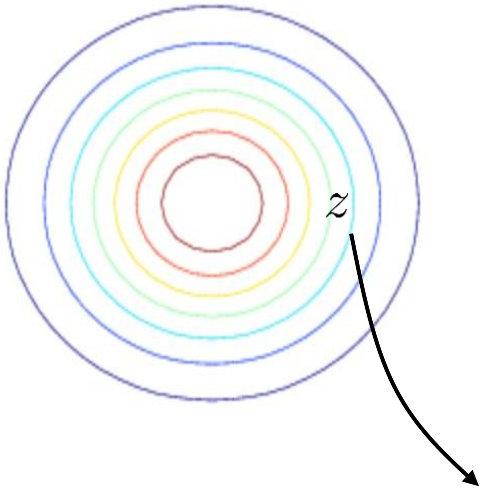
- Policy gradient
  - Can handle both discrete and continuous latent variables
  - High variance, requires multiple samples & small learning rates
- Reparameterization trick
  - Only continuous latent variables
  - Very simple to implement
  - Low variance

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

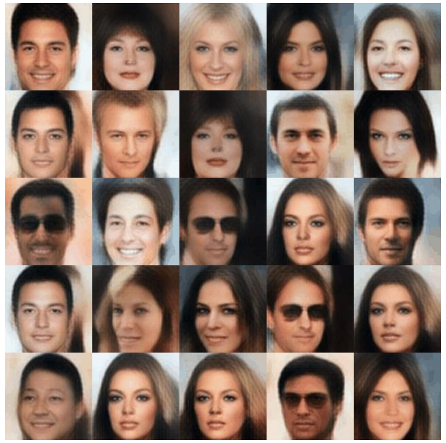
$$\nabla_\phi J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi r(x_i, \mu_\phi(x_i) + \epsilon_j \sigma_\phi(x_i))$$

Variational Inference in Deep RL

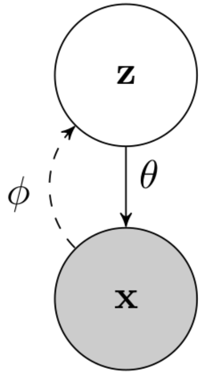
The *variational* autoencoder



$p_\theta(x|z)$

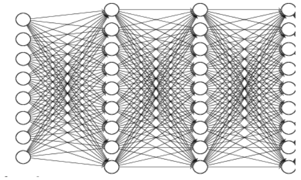


$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$



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

x_i



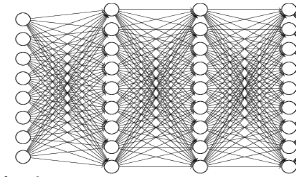
ϕ

$\mu_\phi(x_i)$

$\sigma_\phi(x_i)$

$$\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i) = z$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

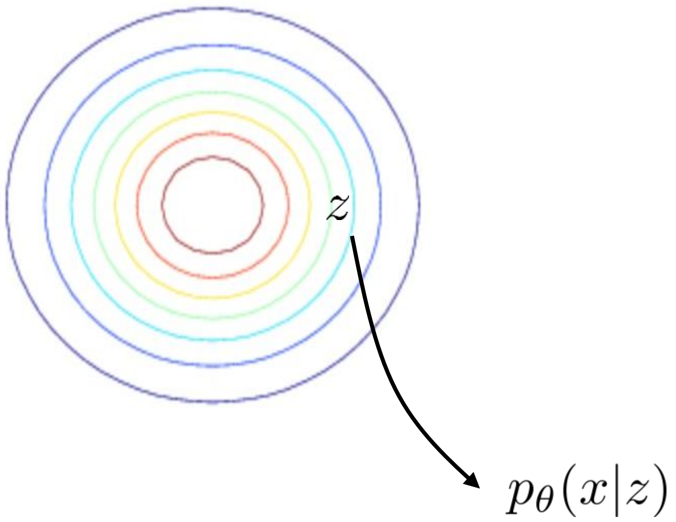


θ

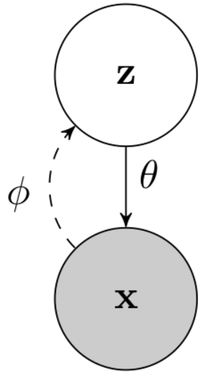
$\longrightarrow p_\theta(x|z)$

$$\max_{\theta,\phi} \frac{1}{N} \sum_i \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{\mathrm{KL}}(q_\phi(z|x_i) \| p(z))$$

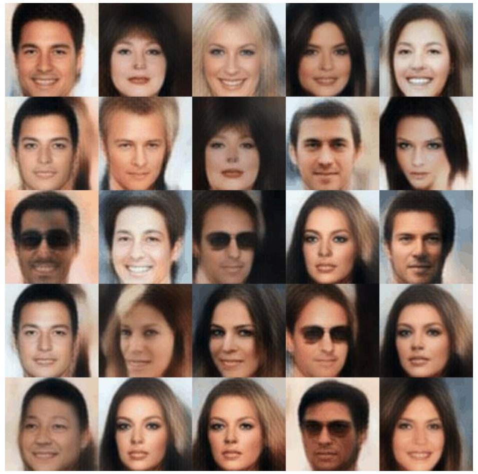
Using the variational autoencoder



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



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



$$p(x) = \int p(x|z)p(z)dz$$

why does this work?

sampling:

$$z \sim p(z)$$

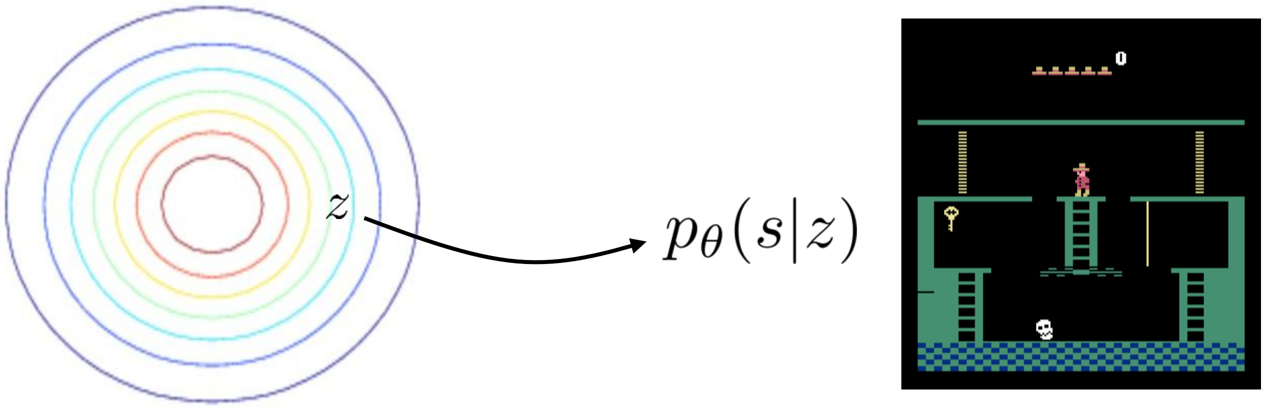
$$x \sim p(x|z)$$

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

Example applications

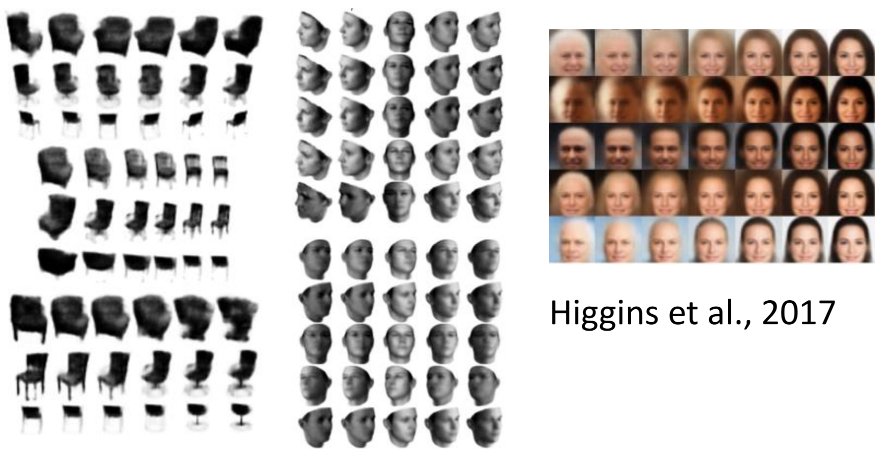
Representation learning

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?



Higgins et al., 2017

Sample algorithm

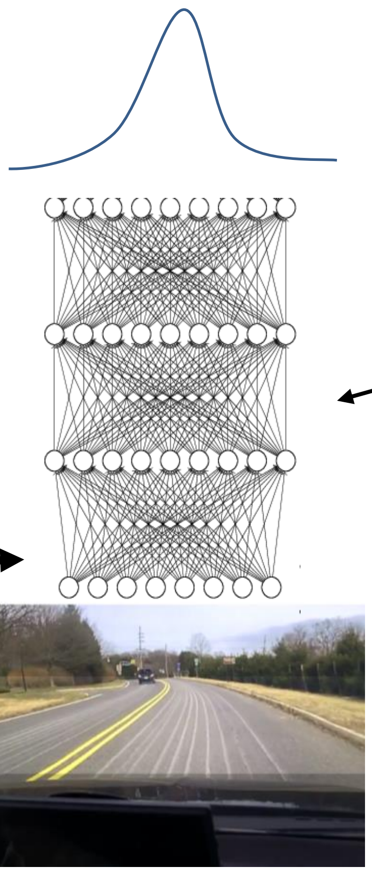
1. Collect transition (s, a, s', r), add to $\mathcal{R}$
2. Update $p_\theta(s|z)$ and $q_\phi(z|s)$ w/ batch from $\mathcal{R}$
3. Update $Q(z, a)$ w/ batch from $\mathcal{R}$

This also provides a great way to use prior data!

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



$p(y|x,z)$

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

$p(z)$

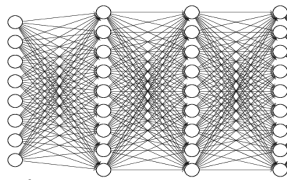
can *optionally* depend on x

at test time:

$z \sim p(z|x_i)$

$y \sim p(y|x_i,z)$

x_i
y_i



ϕ

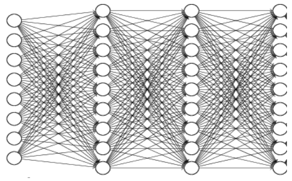
$\mu_\phi(x_i,y_i)$

$\sigma_\phi(x_i,y_i)$

$\mu_\phi + \epsilon\sigma_\phi = z$

$\epsilon \sim \mathcal{N}(0,1)$

x_i

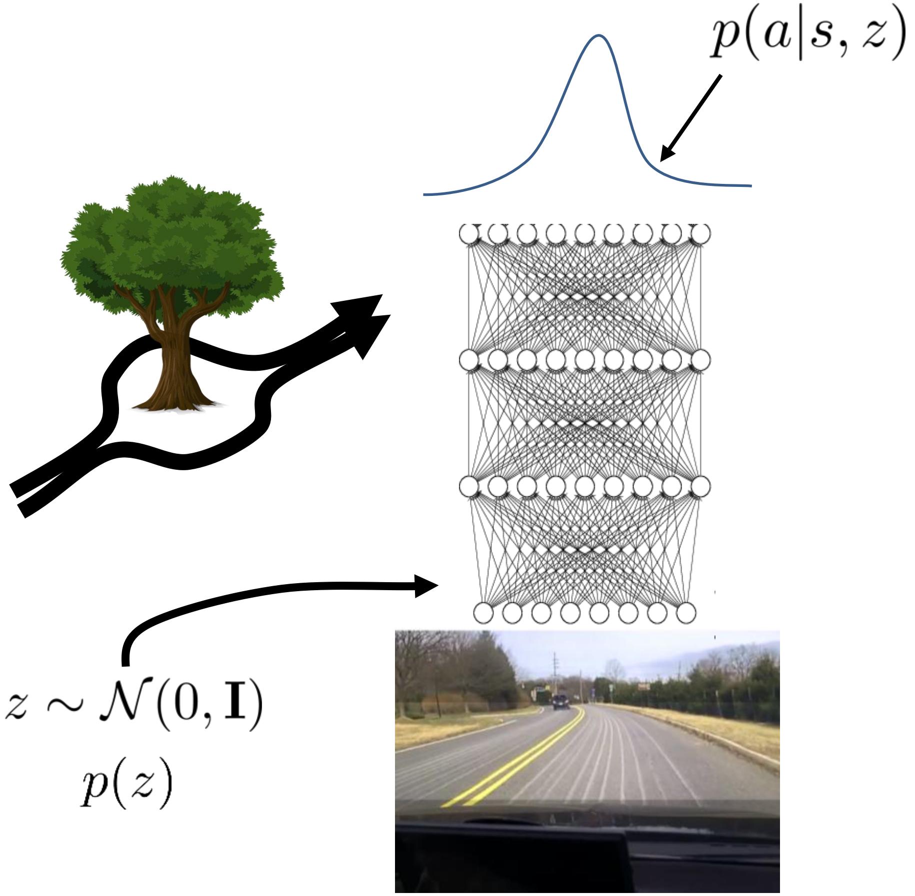


θ

$p_\theta(y_i|x_i,z)$

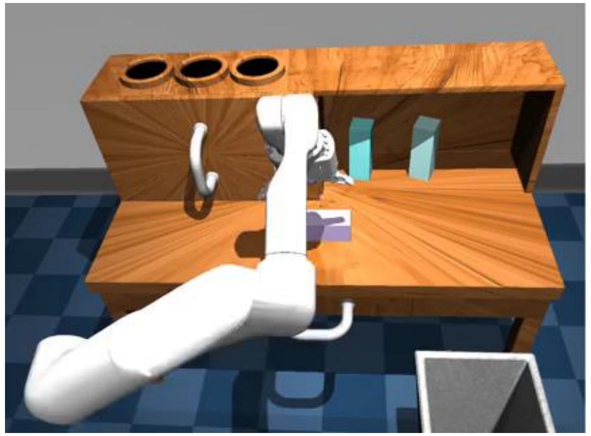
Example applications

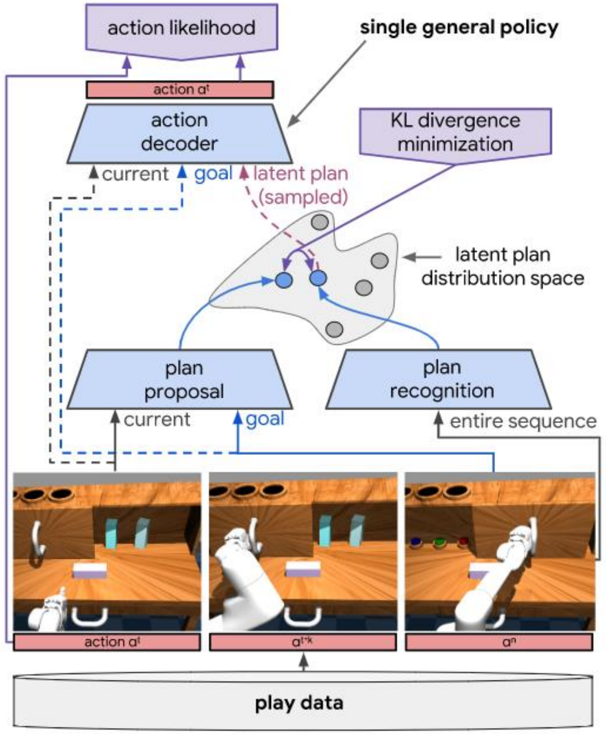
Multimodal imitation learning



Learning Latent Plans from Play

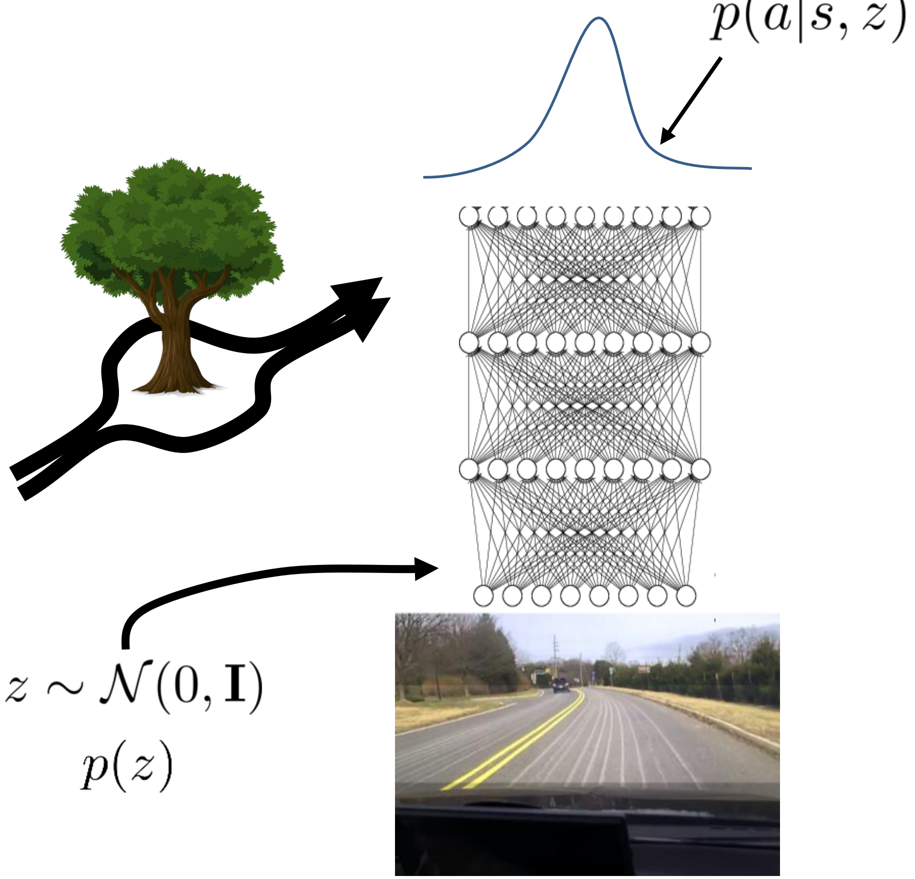
COREY LYNCH Google Brain
MOHI KHANSARI Google X
TED XIAO Google Brain
VIKASH KUMAR Google Brain
JONATHAN TOMPSON Google Brain
SERGEY LEVINE Google Brain
PIERRE SERMANET Google Brain





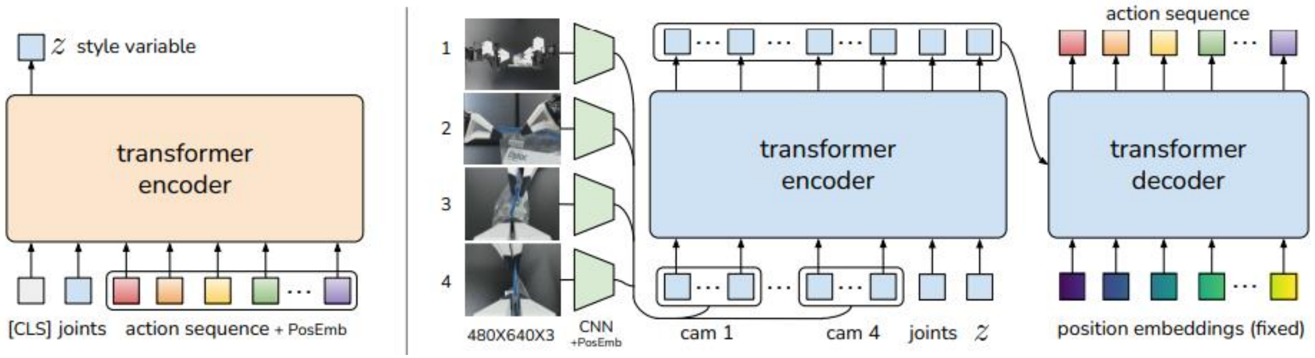
Example applications

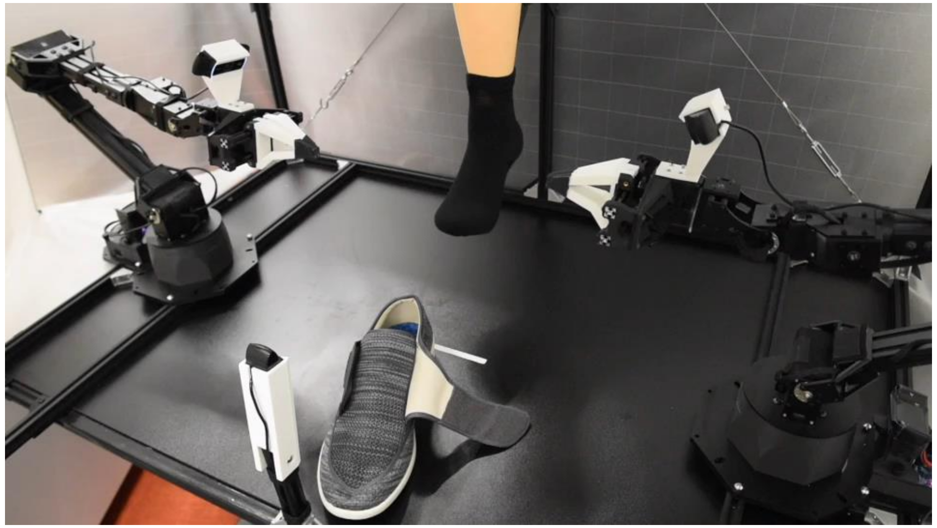
Multimodal imitation learning



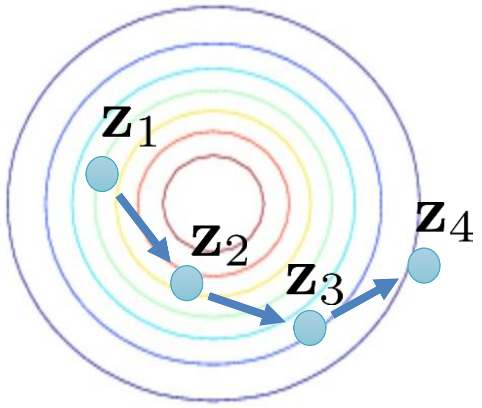
Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

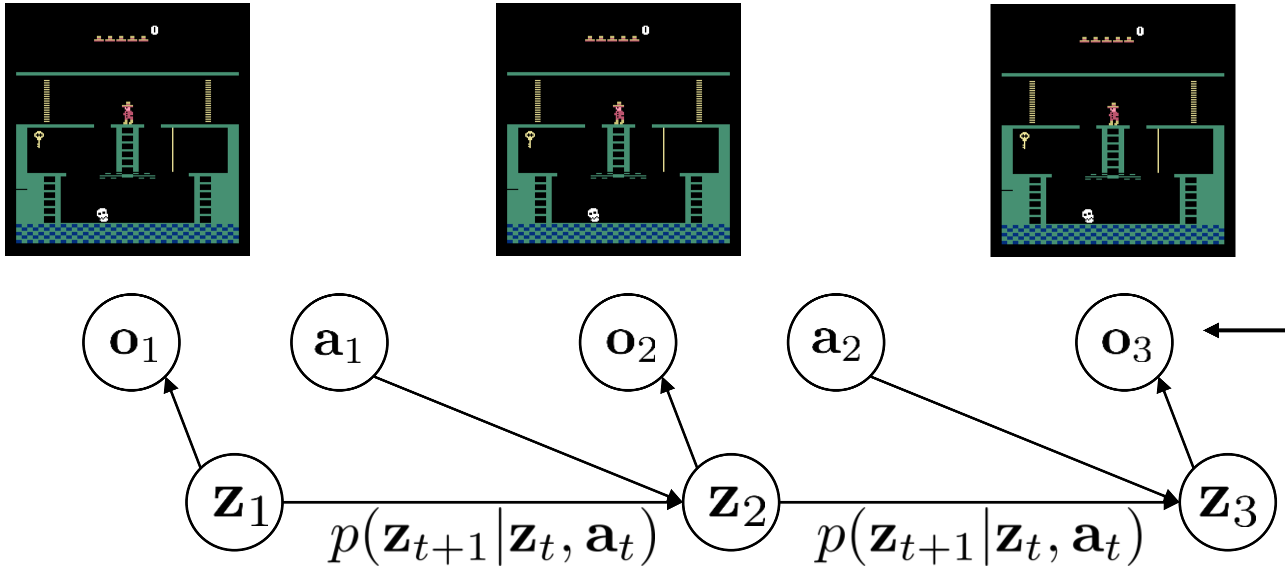
Tony Z. Zhao[1] Vikash Kumar[3] Sergey Levine[2] Chelsea Finn[1]
[1] Stanford University [2] UC Berkeley [3] Meta



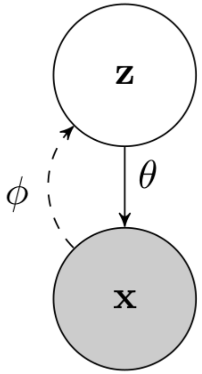


State space models





we are not in partially observed setting



$(\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_T)$

$(\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T)$

What is the **prior**?

What is the **decoder**?

What is the **encoder**?

$\mathcal{N}(0, \mathbf{I})$

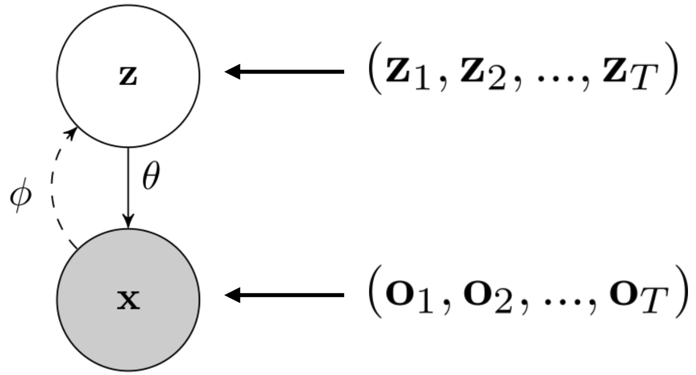
learned

$$p(\mathbf{z}) = p(\mathbf{z}_1) \prod_t p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$$

$$p_\theta(\mathbf{o}|\mathbf{z}) = \prod_t p(\mathbf{o}_t|\mathbf{z}_t)$$

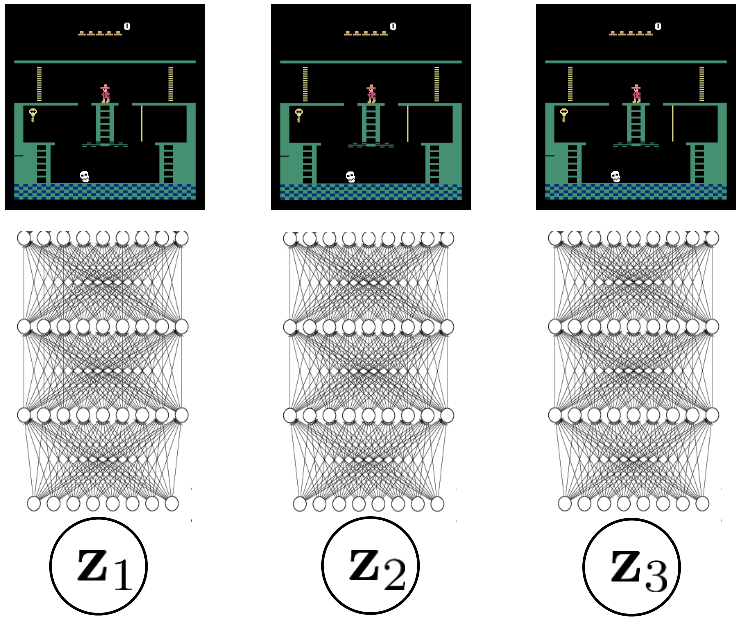
$$q_\phi(\mathbf{z}|\mathbf{o}) = \prod_t q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t})$$

State space models



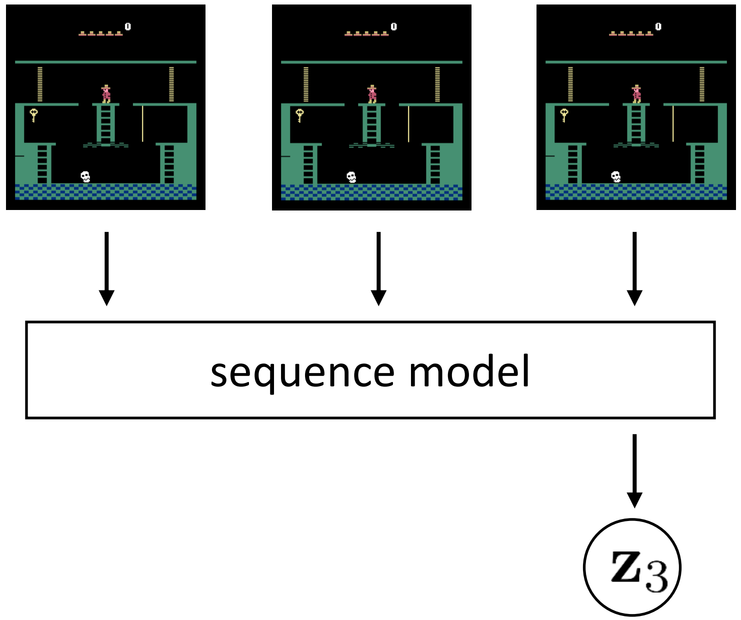
What is the **decoder**?

$$p_\theta(\mathbf{o}|\mathbf{z}) = \prod_t p(\mathbf{o}_t|\mathbf{z}_t)$$



What is the **encoder**?

$$q_\phi(\mathbf{z}|\mathbf{o}) = \prod_t q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t})$$



LSTM
Transformer

Example applications

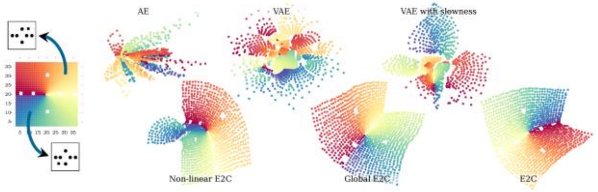
Representation learning and model-based RL

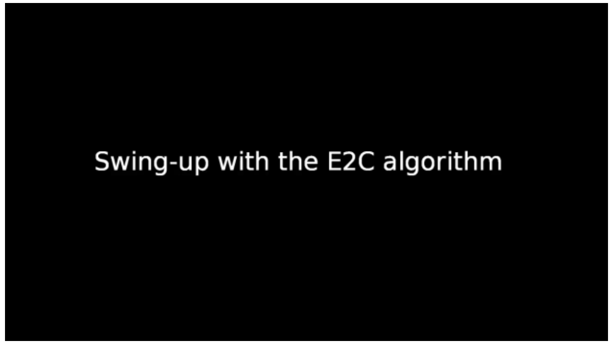
1. Learn state space model and plan in the latent space

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

Manuel Watter* Jost Tobias Springenberg* Joschka Boedecker
University of Freiburg, Germany
{watterm, springj, jboedeck}@cs.uni-freiburg.de

Martin Riedmiller
Google DeepMind
London, UK
riedmiller@google.com

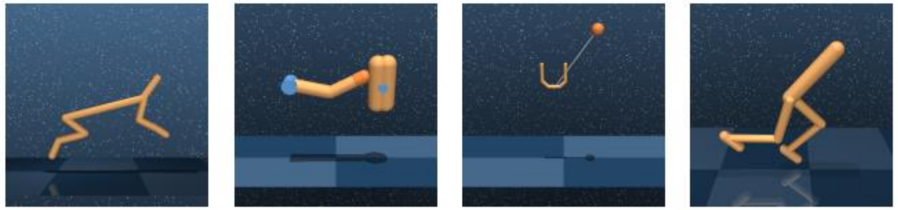




SOLAR: Deep Structured Latent Representations for Model-Based Reinforcement Learning

Learning Latent Dynamics for Planning from Pixels

Danijar Hafner [1 2] Timothy Lillicrap [3] Ian Fischer [4] 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

Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model

Alex X. Lee[1,2] Anusha Nagabandi[1] Pieter Abbeel[1] Sergey Levine[1]
[1]University of California, Berkeley
[2]DeepMind
{alexlee_gk,nagaban2,pabbeel,svlevine}@cs.berkeley.edu

true rollouts

samples

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner * University of Toronto Google Brain
Timothy Lillicrap DeepMind
Jimmy Ba University of Toronto
Mohammad Norouzi Google Brain