

Variational Inference and Generative Models

CS 294-112: Deep Reinforcement Learning

Sergey Levine

Class Notes

1. Homework 3 due next Wednesday
2. Accept CMT peer review invitations
 - These are required (part of your final project grade)
 - If you have not received/cannot find invitation, email Kate Rakelly!

Where we are in the course

Lecture 1: Introduction and Course Overview

Lecture 2: Supervised Learning and Imitation

Lecture 3: TensorFlow and Neural Nets Review
Session (notebook)

Lecture 4: Reinforcement Learning Introduction

Lecture 5: Policy Gradients Introduction

Lecture 6: Actor-Critic Introduction

Lecture 7: Value Functions and Q-Learning

Lecture 8: Advanced Q-Learning Algorithms

Lecture 9: Advanced Policy Gradients

Lecture 10: Optimal Control and Planning

Lecture 11: Model-Based Reinforcement Learning

Lecture 12: Advanced Model Learning and Images

Lecture 13: Learning Policies by Imitating Other
Policies

Lecture 14: Probability and Variational Inference
Primer



RL algorithms

Lecture 15: Connection between Inference and
Control

Lecture 16: Inverse Reinforcement Learning

Lecture 17: Exploration: Part 1

Lecture 18: Exploration: Part 2

Lecture 19: Transfer Learning and Multi-Task Learning

Lecture 20: Meta-Learning

Lecture 21: Parallelism and RL System Design

Lecture 22: Advanced Imitation Learning and Open
Problems

Lecture 23: Guest Lecture: Craig Boutilier

Lecture 24: Guest Lecture: Kate Rakelly & Gregory
Kahn

Lecture 25: Guest Lecture: Quoc Le

Lecture 26: Guest Lecture: Karol Hausman

Lecture 27: Final Project Presentations: Part 1

Lecture 28: Final Project Presentations: Part 2



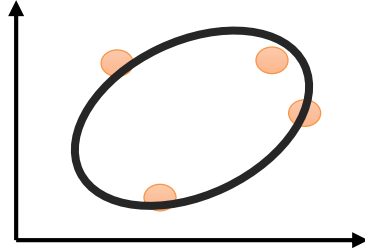
advanced topics

Today's Lecture

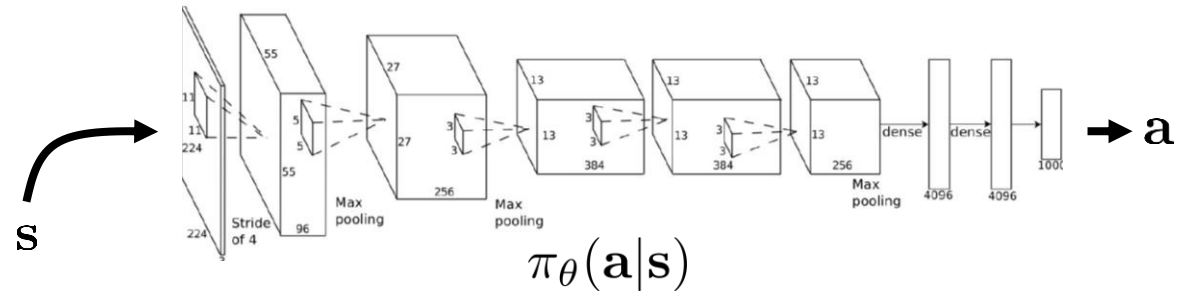
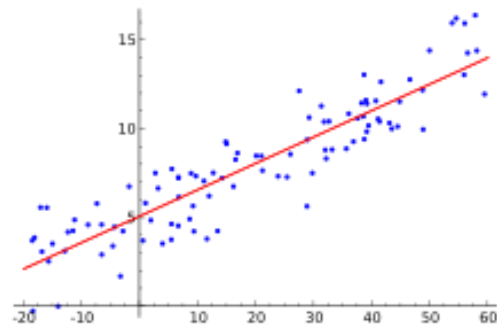
1. Probabilistic latent variable models
 2. Variational inference
 3. Amortized variational inference
 4. Generative models: variational autoencoders
- Goals
 - Understand latent variable models in deep learning
 - Understand how to use (amortized) variational inference

Probabilistic models

$p(x)$



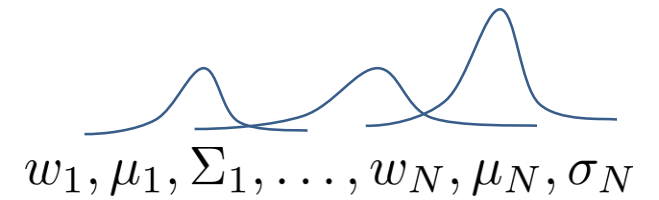
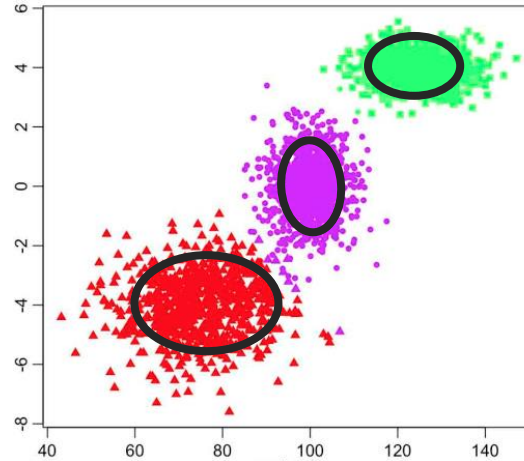
$p(y|x)$



Latent variable models

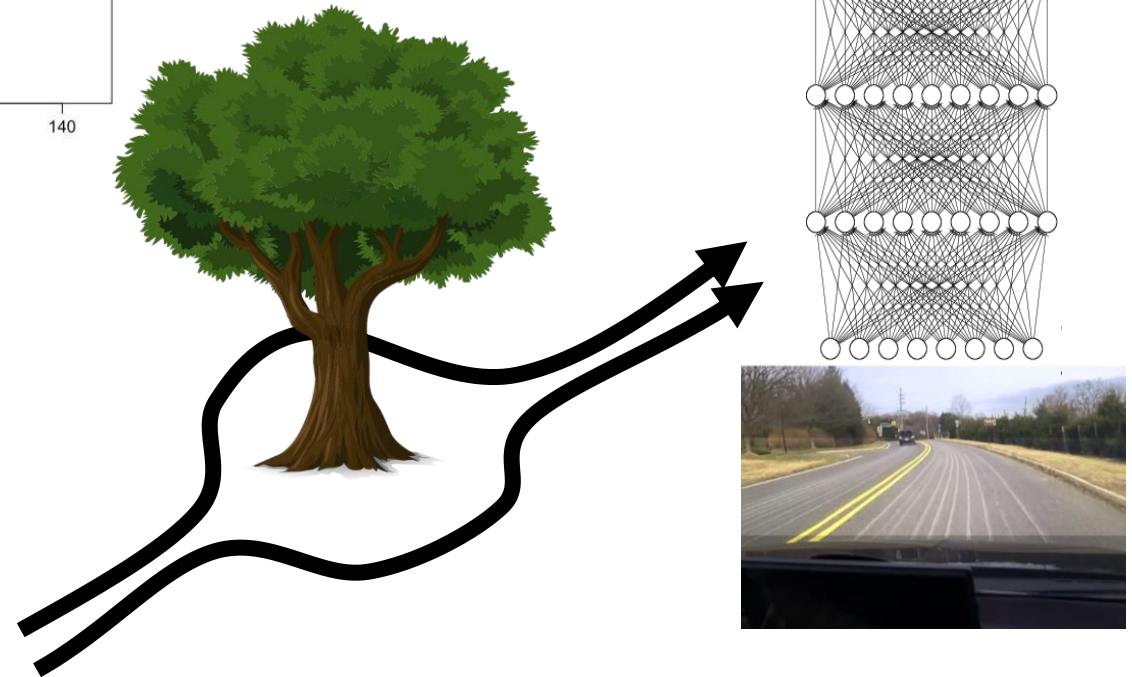
$$p(x) = \sum_z p(x|z)p(z)$$

↑
mixture
element



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

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

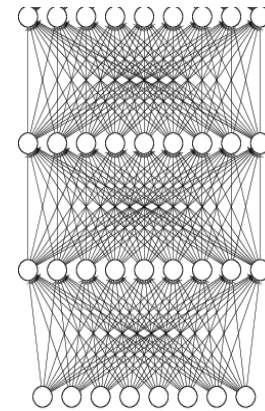
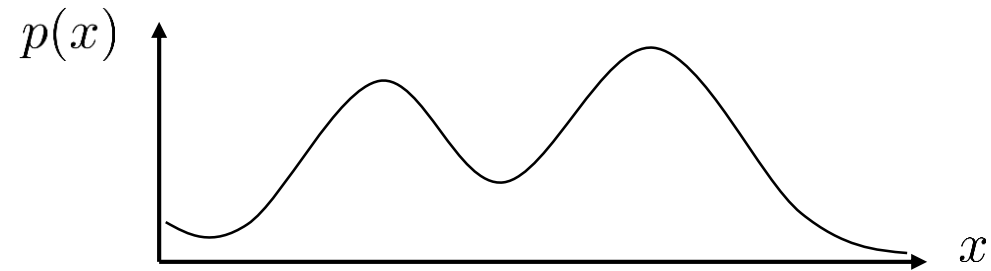


Latent variable models in general

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

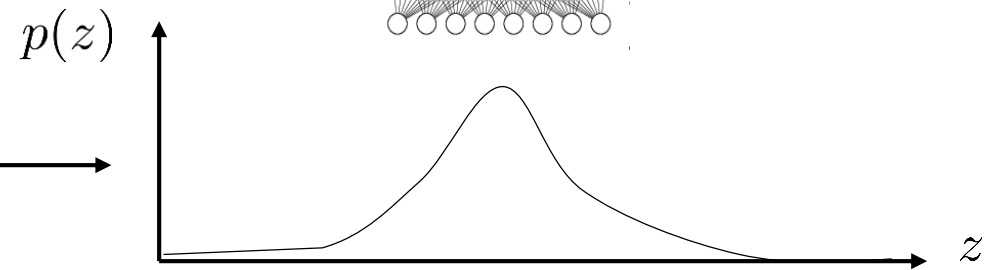
“easy” distribution
(e.g., conditional Gaussian)

“easy” distribution
(e.g., Gaussian)



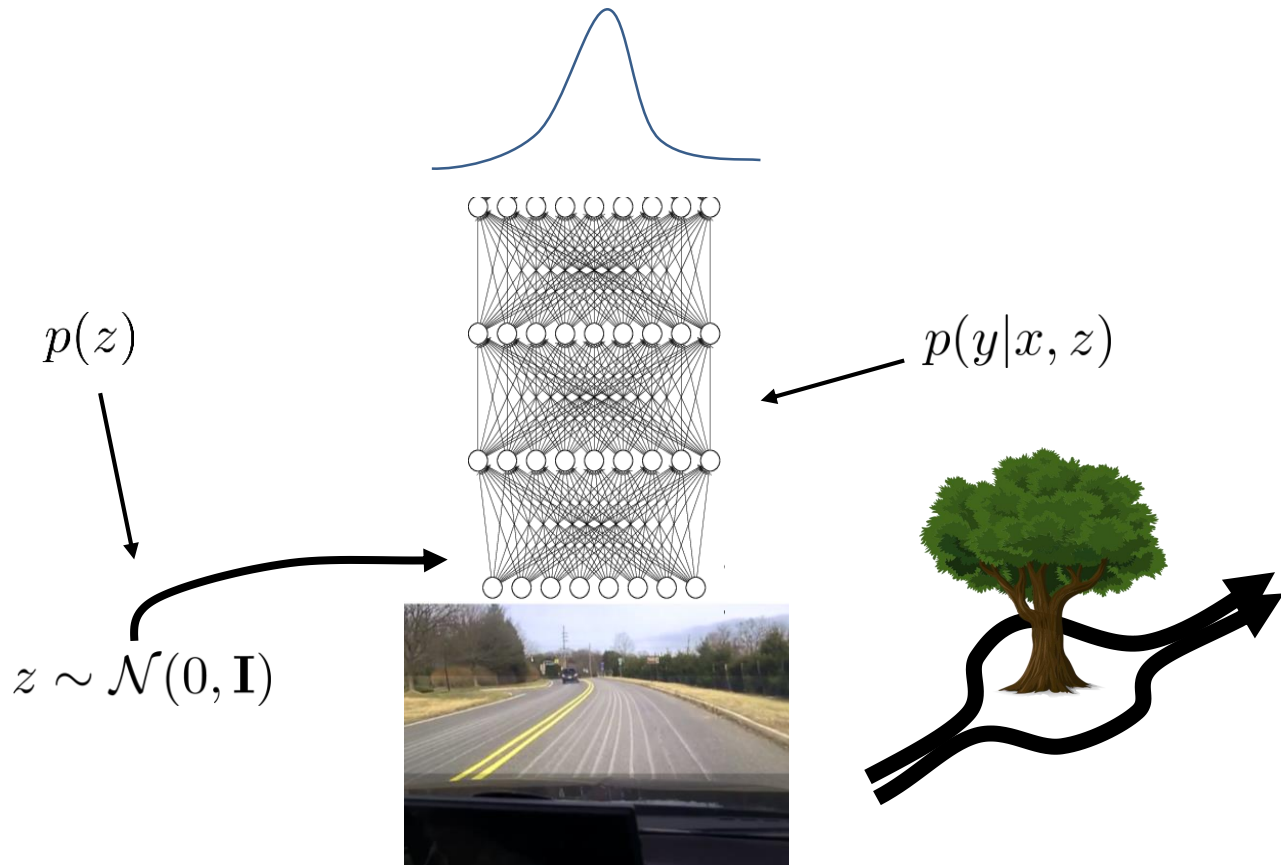
$$p(x|z) = \mathcal{N}(\mu_{\text{nn}}(z), \sigma_{\text{nn}}(z))$$

“easy” distribution
(e.g., Gaussian)

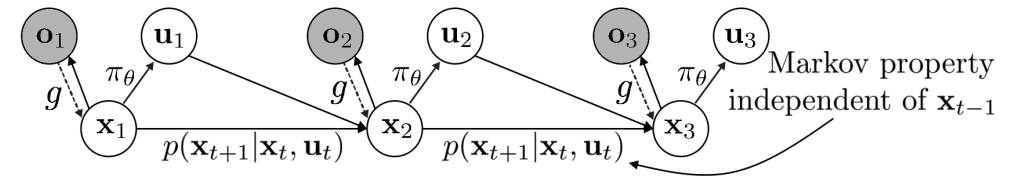


Latent variable models in RL

conditional latent variable models for multi-modal policies



latent variable models for model-based RL



$$p(o_t|x_t)$$

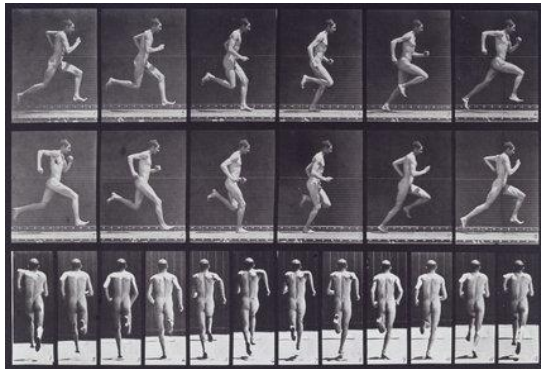
$$p(x_t)$$

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

latent space has *structure*

Other places we'll see latent variable models

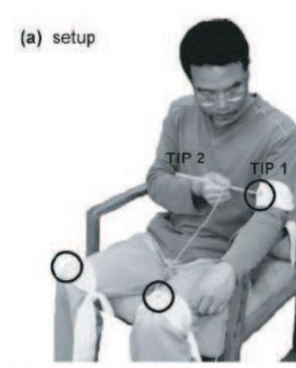
Using RL/control + variational inference to model human behavior



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

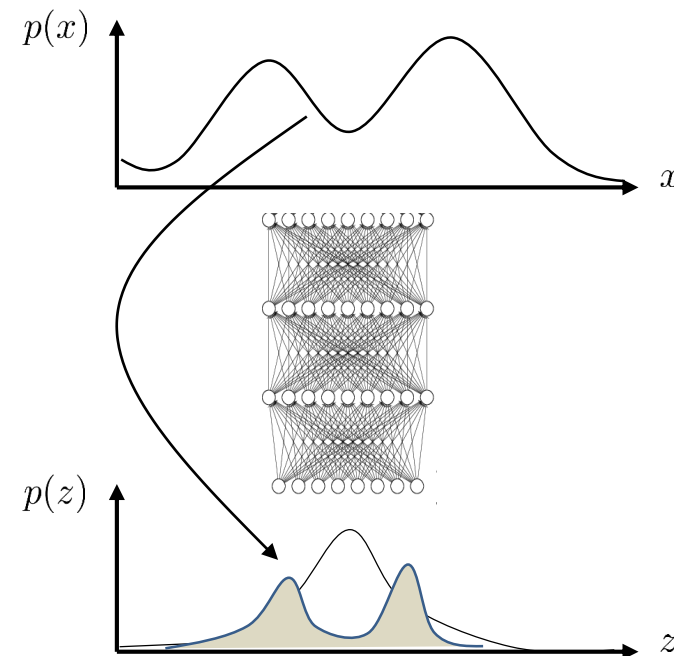
alternative: *expected* log-likelihood:

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

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 $p(z|x_i)$



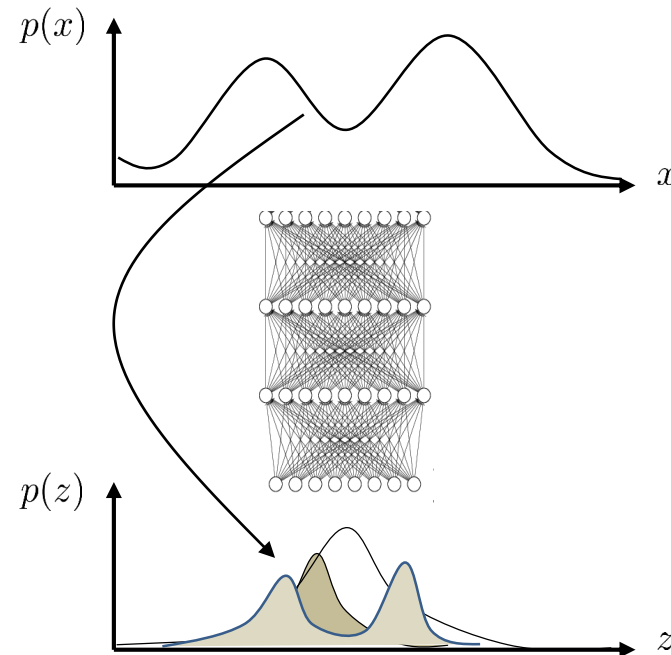
The variational approximation

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

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

can bound $\log p(x_i)$!

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



The variational approximation

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

can bound $\log p(x_i)$!

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

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

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

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

maximizing this maximizes $\log p(x_i)$



Jensen's inequality

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

A brief aside...

Entropy:

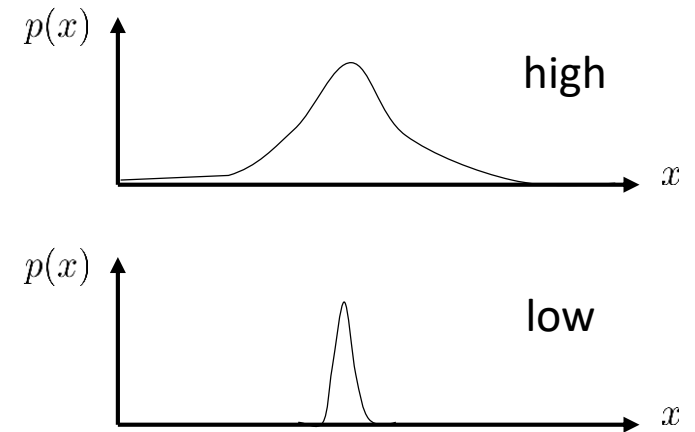
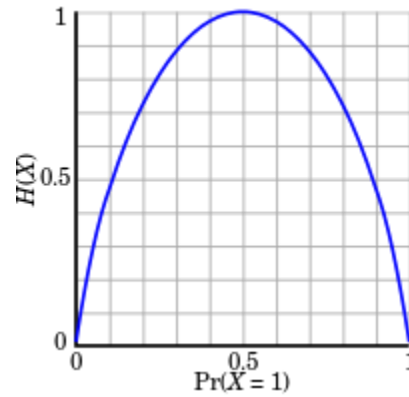
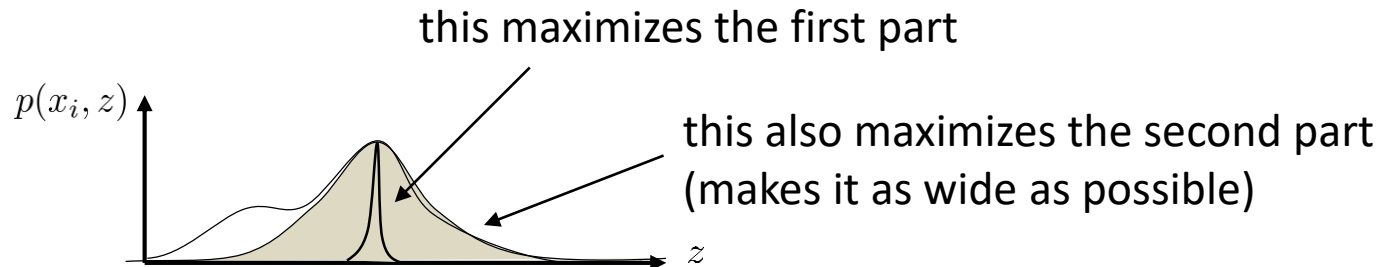
$$\mathcal{H}(p) = -E_{x \sim p(x)}[\log p(x)] = - \int_x p(x) \log p(x) dx$$

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?

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



A brief aside...

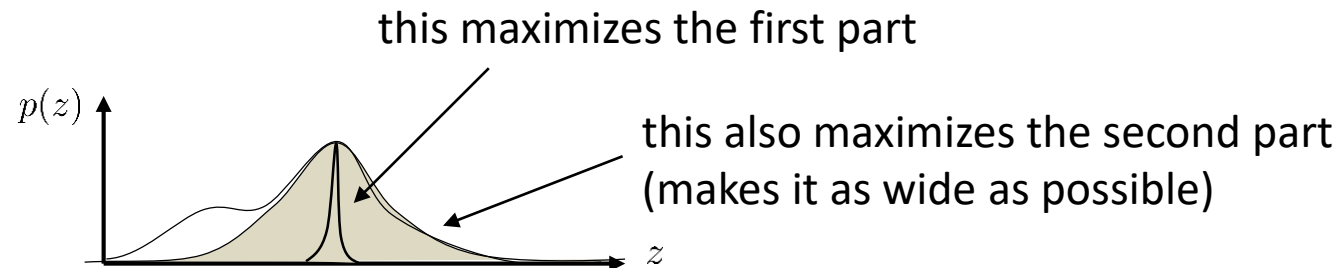
KL-Divergence:

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

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?



The variational approximation

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

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

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

why?

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

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

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

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

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

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

The variational approximation

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

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

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

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

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

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?

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

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

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

for each x_i (or mini-batch):

calculate $\nabla_{\theta} \mathcal{L}_i(p, q_i)$:

sample $z \sim q_i(x_i)$

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

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathcal{L}_i(p, q_i)$$

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 $z \sim q_i(x_i)$

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

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathcal{L}_i(p, q_i)$$

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

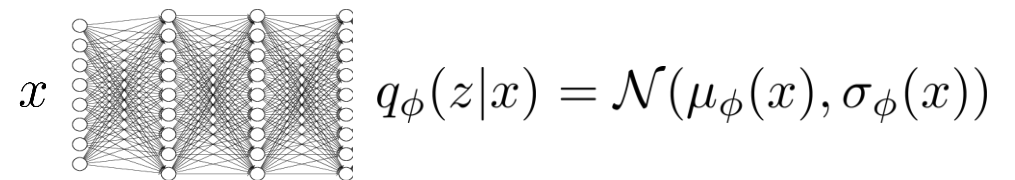
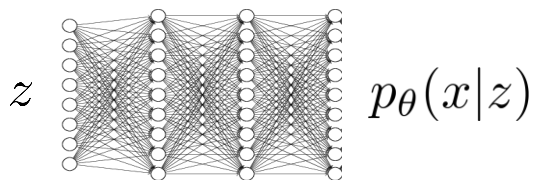
gradient ascent on μ_i, σ_i

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* $q_i(z) = q(z|x_i) \approx p(z|x_i)$?



Break

What's the problem?

for each x_i (or mini-batch):

calculate $\nabla_{\theta} \mathcal{L}_i(p, q_i)$:

sample $z \sim q_i(x_i)$

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

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathcal{L}_i(p, q_i)$$

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

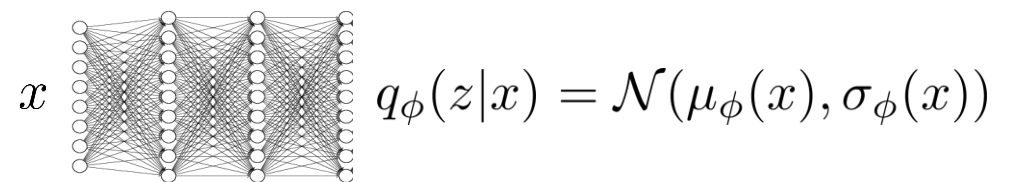
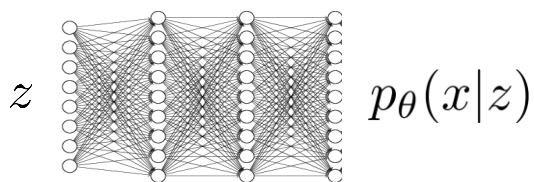
gradient ascent on μ_i, σ_i

How many parameters are there?

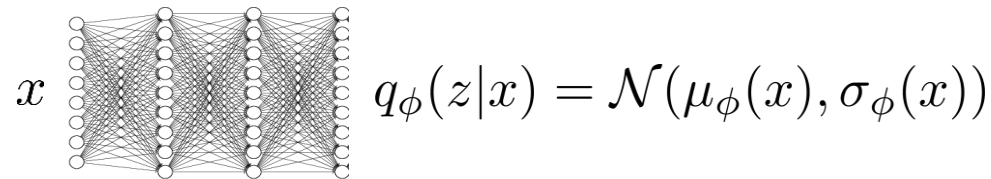
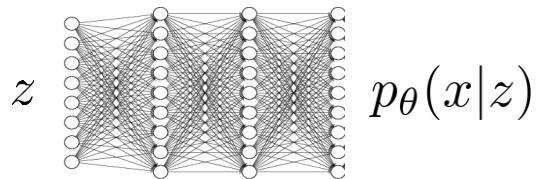
$$|\theta| + (|\mu_i| + |\sigma_i|) \times N$$

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

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



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 $z \sim q_{\phi}(z|x_i)$

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

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

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

how do we calculate this?

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

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



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

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

can just use policy gradient!

What's wrong with this gradient?

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

The reparameterization trick

Is there a better way?

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

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

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

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

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

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



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

independent of ϕ !

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

Another way to look at it...

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

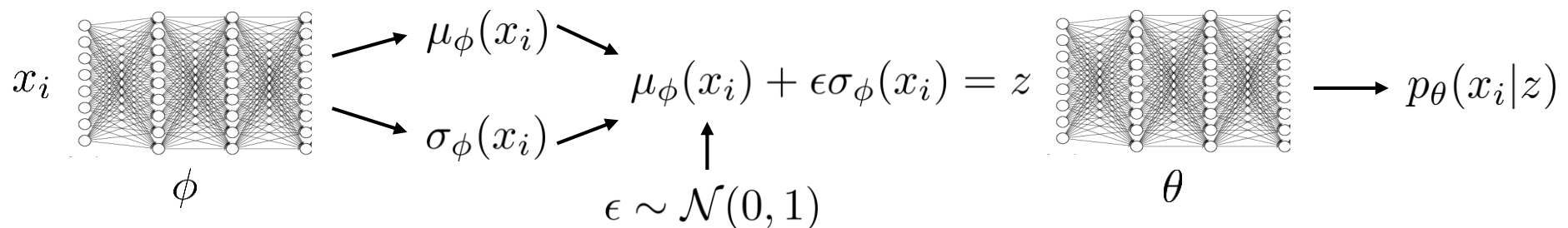
$$= E_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i|z)] + \underbrace{E_{z \sim q_\phi(z|x_i)} [\log p(z)] + \mathcal{H}(q_\phi(z|x_i))}_{-D_{\text{KL}}(q_\phi(z|x_i) \| p(z))}$$

$-D_{\text{KL}}(q_\phi(z|x_i) \| p(z))$ ← this often has a convenient analytical form (e.g., KL-divergence for Gaussians)

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

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

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



Reparameterization trick vs. policy gradient

- Policy gradient

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

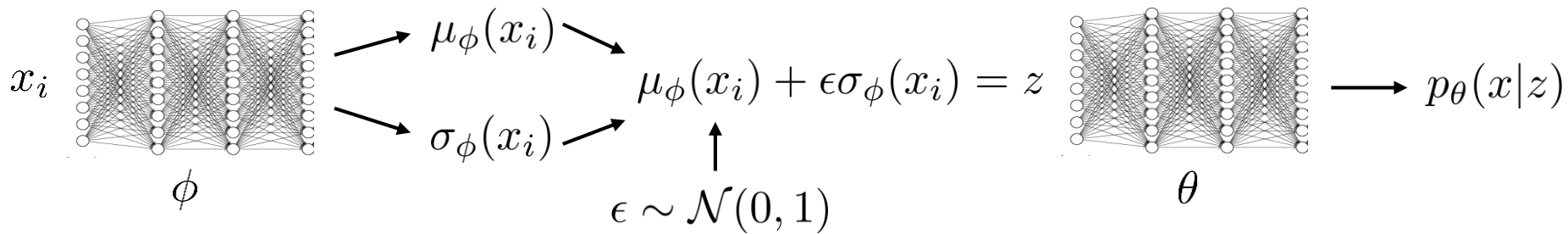
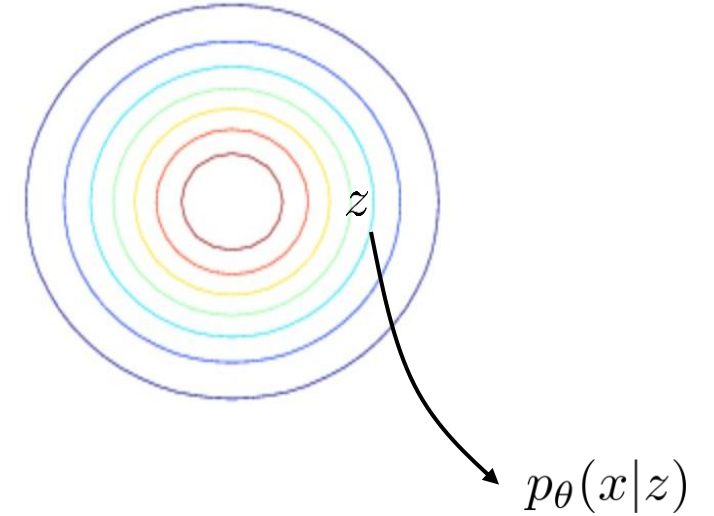
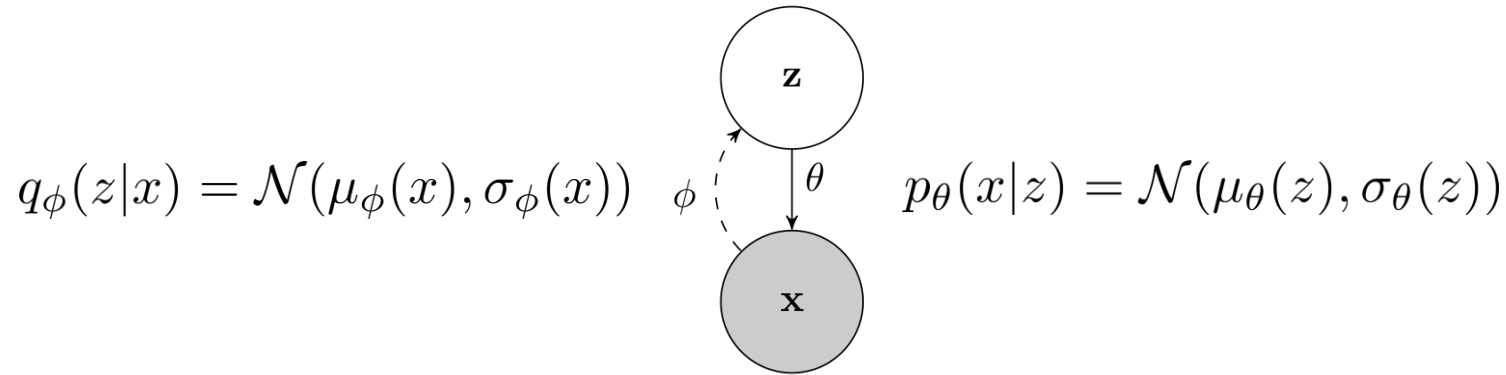
$$J(\phi) \approx \frac{1}{M} \sum_j \nabla_{\phi} \log q_{\phi}(z_j|x_i) r(x_i, z_j)$$

- Reparameterization trick

- Only continuous latent variables
- Very simple to implement
- Low variance

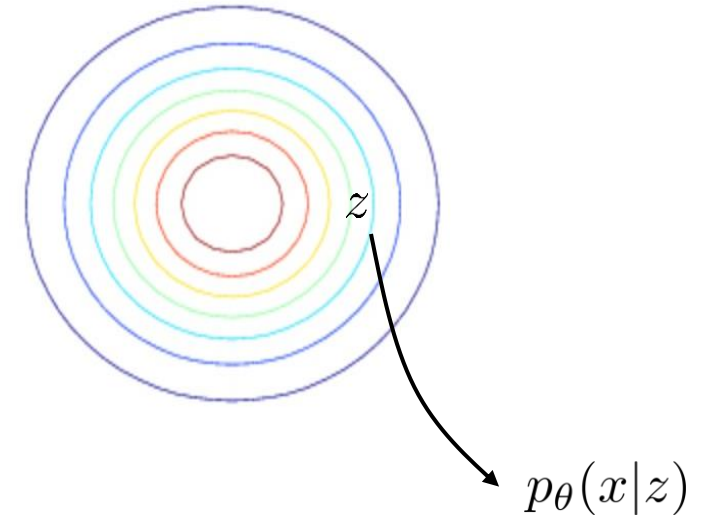
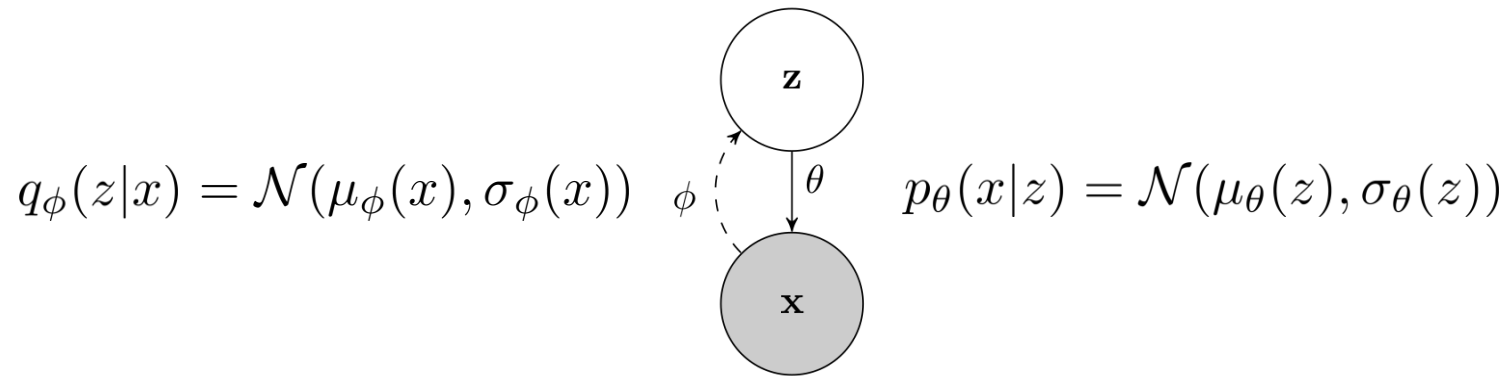
$$\nabla_{\phi} J(\phi) \approx \frac{1}{M} \sum_j \nabla_{\phi} r(x_i, \mu_{\phi}(x_i) + \epsilon_j \sigma_{\phi}(x_i))$$

The *variational* autoencoder



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

Using the variational autoencoder



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

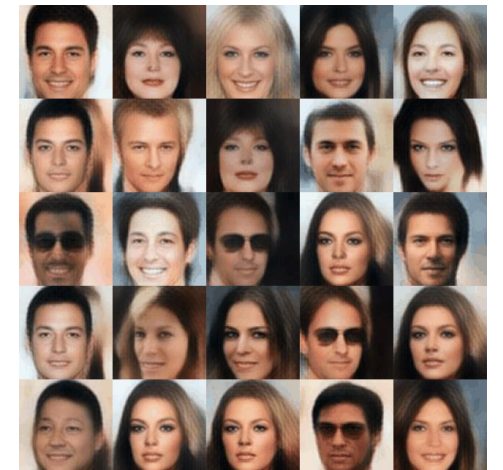
why does this work?

sampling:

$$z \sim p(z)$$

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

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



Conditional models

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

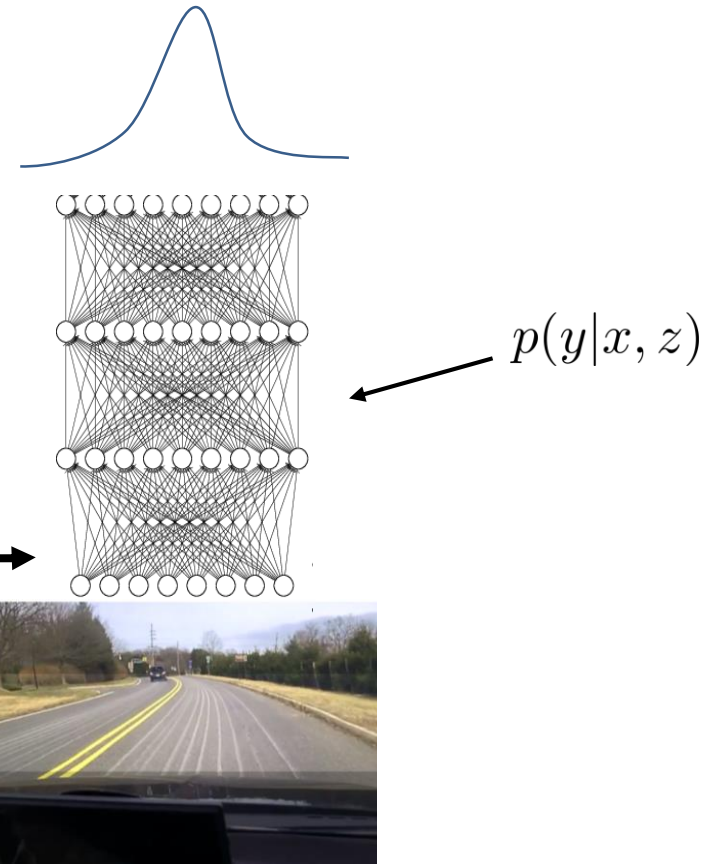
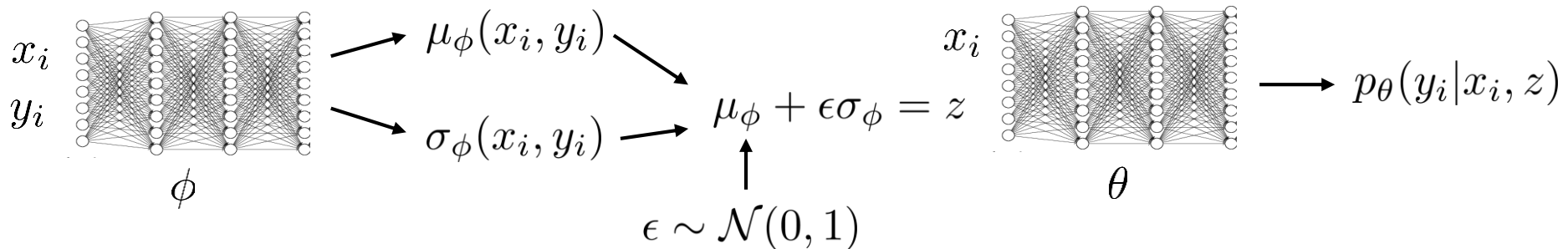
just like before, only now generating y_i
and *everything* is conditioned on x_i

at test time:

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

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

can *optionally* depend on x



Examples

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

Manuel Watter*

Jost Tobias Springenberg*

Martin Riedmiller

Joschka Boedecker

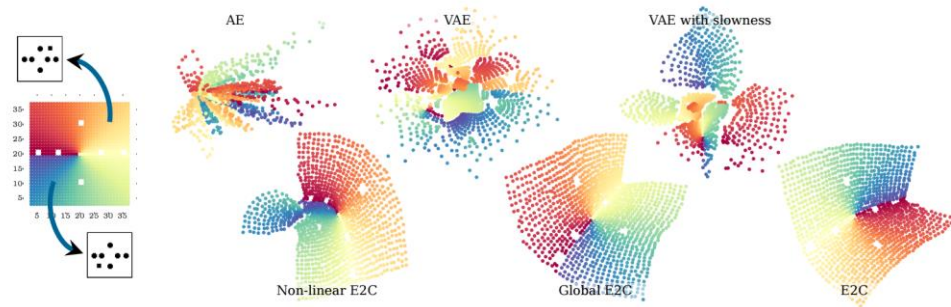
Google DeepMind

University of Freiburg, Germany

London, UK

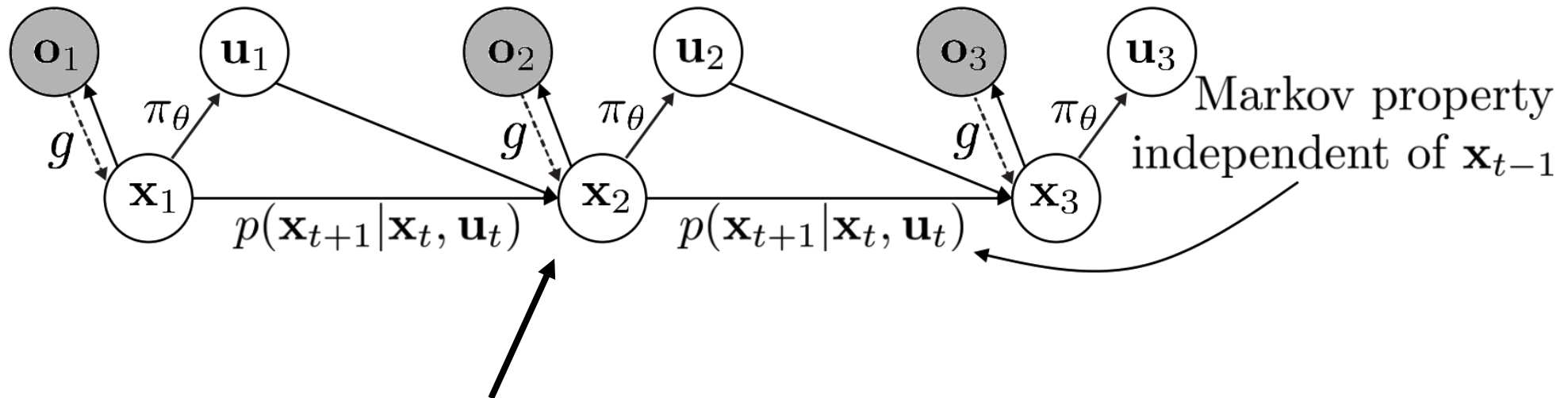
{watterm, springj, jboedeck}@cs.uni-freiburg.de

riedmiller@google.com



Swing-up with the E2C algorithm

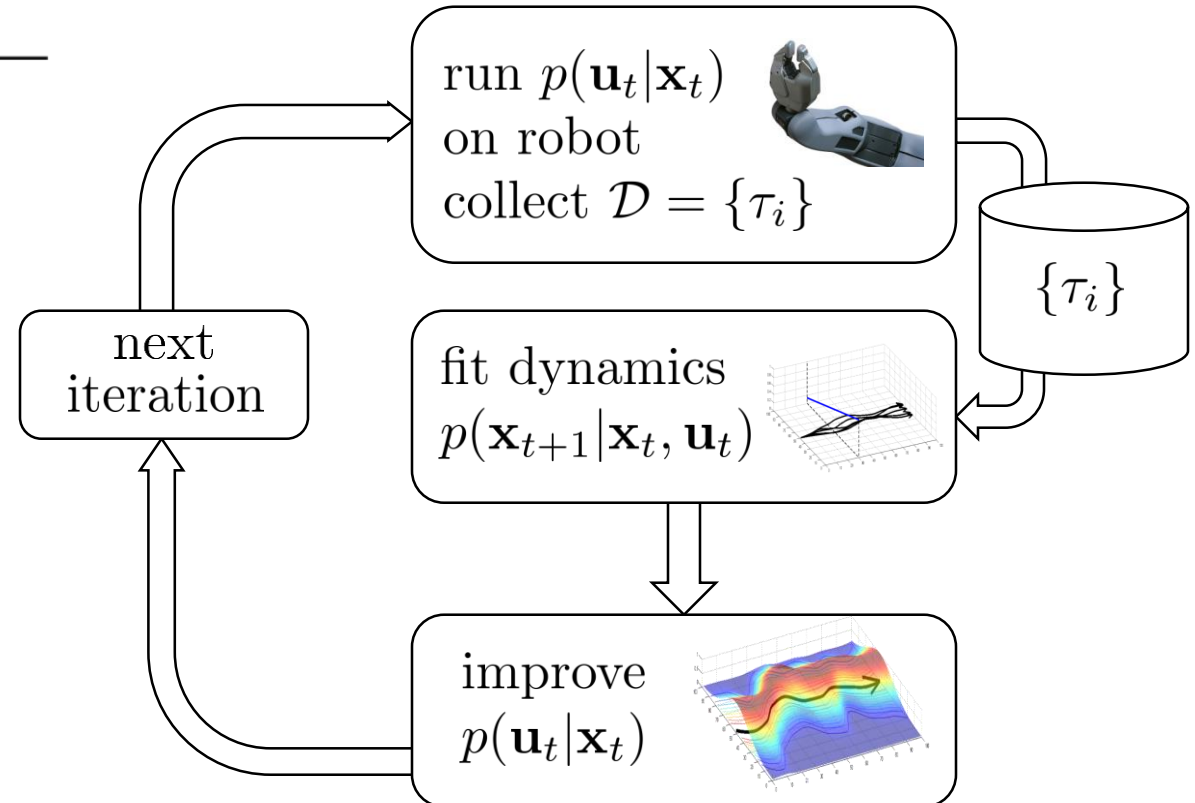
1. collect data
2. learn embedding of image & dynamics model (**jointly**)
3. run iLQG to learn to reach image of goal



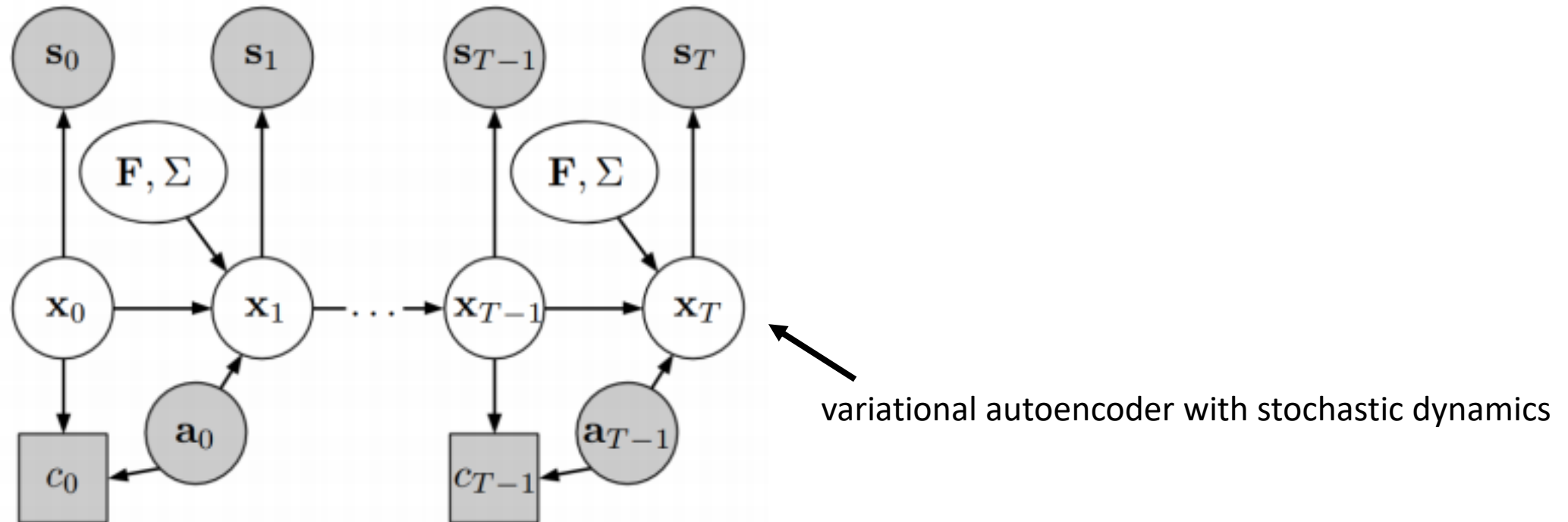
a type of variational autoencoder with temporally decomposed latent state!

Local models with images

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

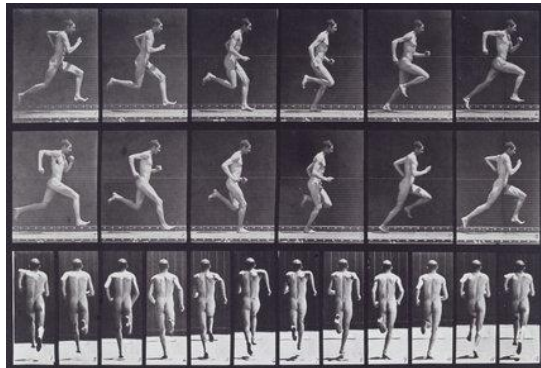


Local models with images



We'll see more of this for...

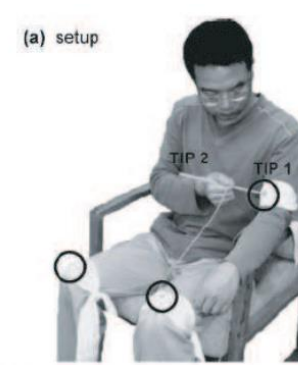
Using RL/control + variational inference to model human behavior



Muybridge (c. 1870)



Mombaur et al. '09



Li & Todorov '06



Ziebart '08

Using generative models and variational inference for exploration

