Variational Inference and Generative Models

CS 330

With slides adapted from Sergey Levine, CS285

Course Reminders

Homework 3 due Monday next week.

Tutorial session on **Thursday 4:30 pm**

Be careful Azure usage — turning off machines when you are not using them!

This Week

A Bayesian perspective on meta-learning

Today: Approximate Bayesian inference via variational inference

Bayes is back.

Plan for Today

1. Latent variable models
2. Variational inference
3. Amortized variational inference
4. Example latent variables models

Part of (optional) Homework 4

Goals

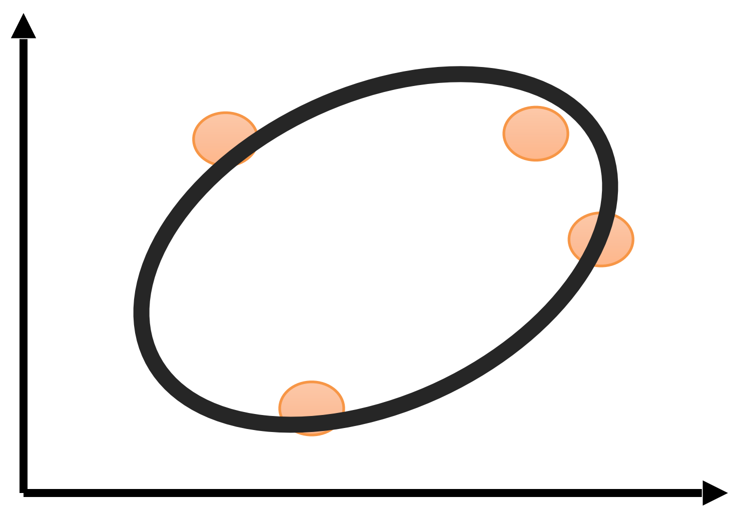
- Understand latent variable models in deep learning
- Understand how to use (amortized) variational inference

Probabilistic models

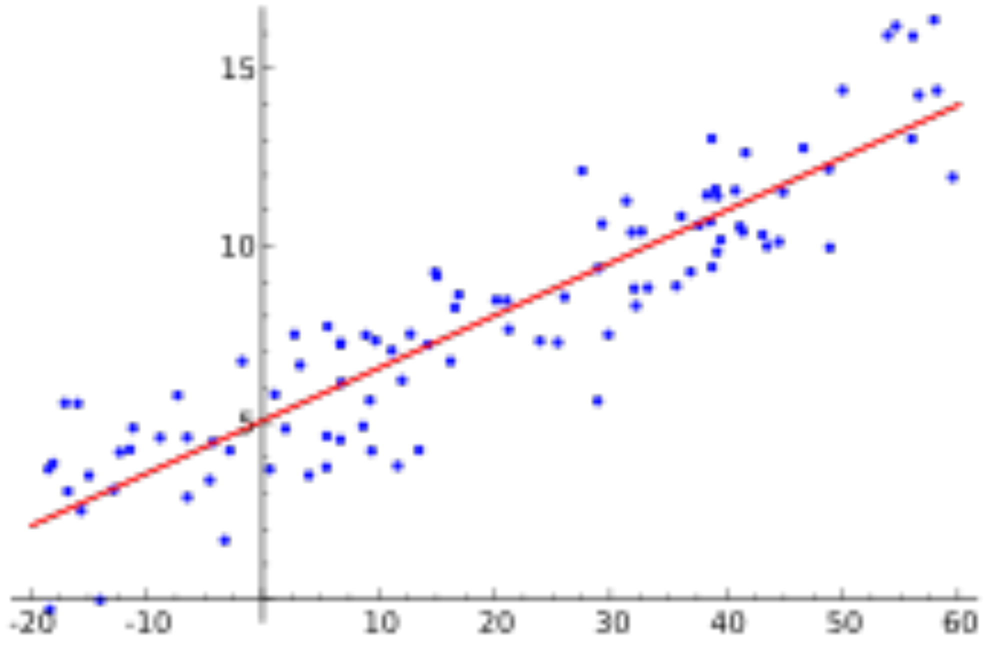
These models can approximate the data distribution. P(x, y). If we know $p_\theta(x, y)$
then we can approximate P(y|x).

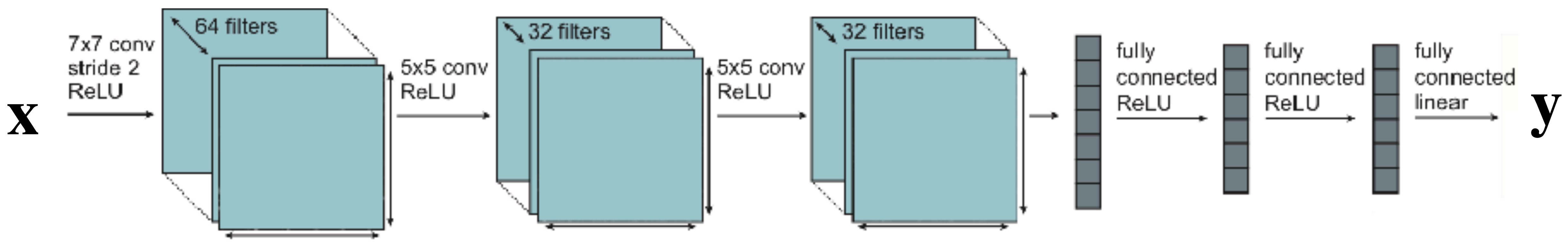
They are useful because:
1. They can fit model that can capture predictive uncertainty
2. Models are more principled by explicitly modelling data uncertainty

$p(x)$



$p(y|x)$





$f_\theta(\mathbf{y} \mid \mathbf{x})$

Most commonly:

- probability values of discrete **categorical distribution**
- mean and variance of a **Gaussian**

But it could be other distributions!

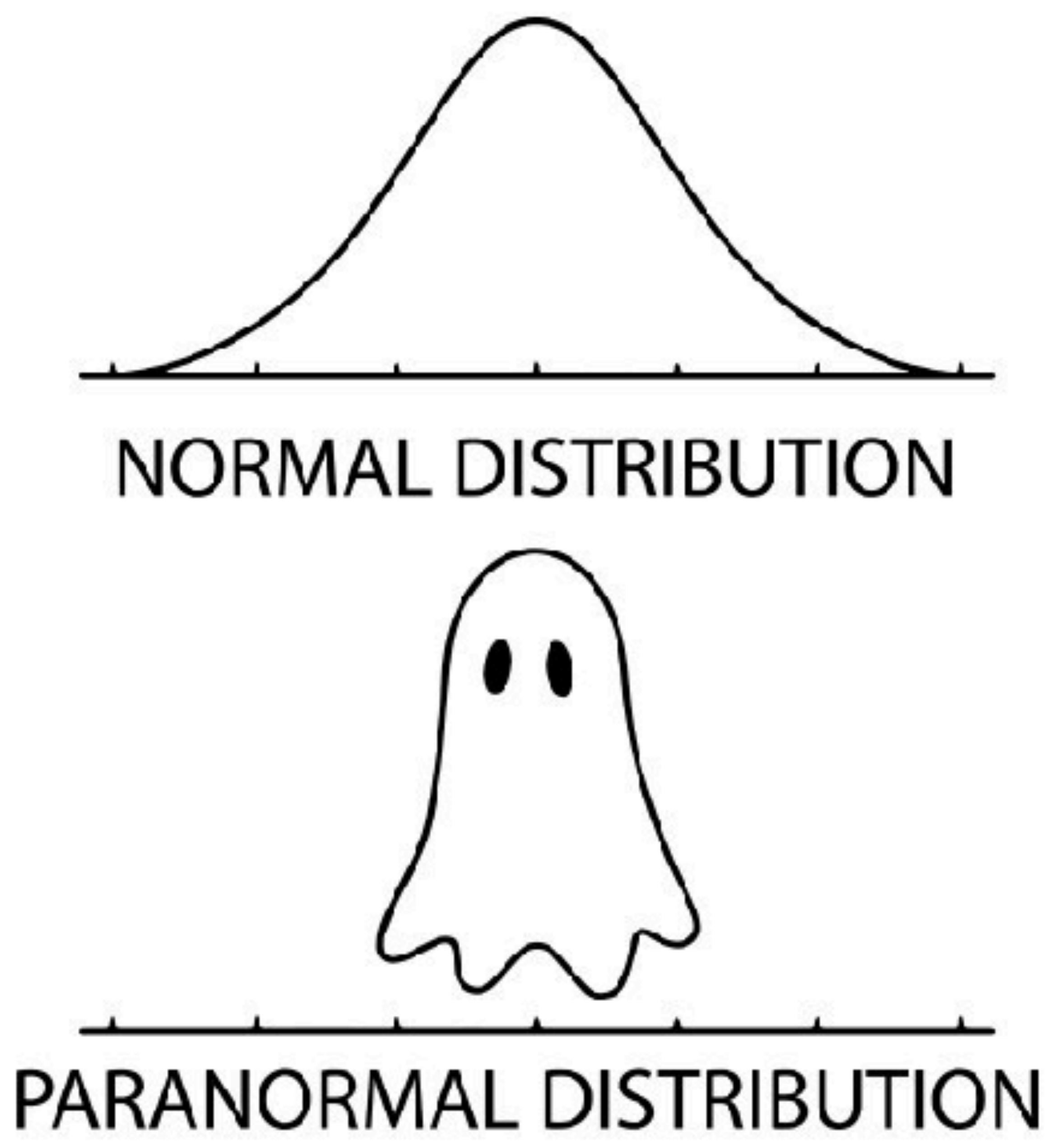
How do we train probabilistic models?

the model: $p_\theta(x)$

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

maximum likelihood fit:

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



Easy to evaluate & differentiate for **categorical** or **Gaussian** distributions.
i.e. cross-entropy, MSE losses

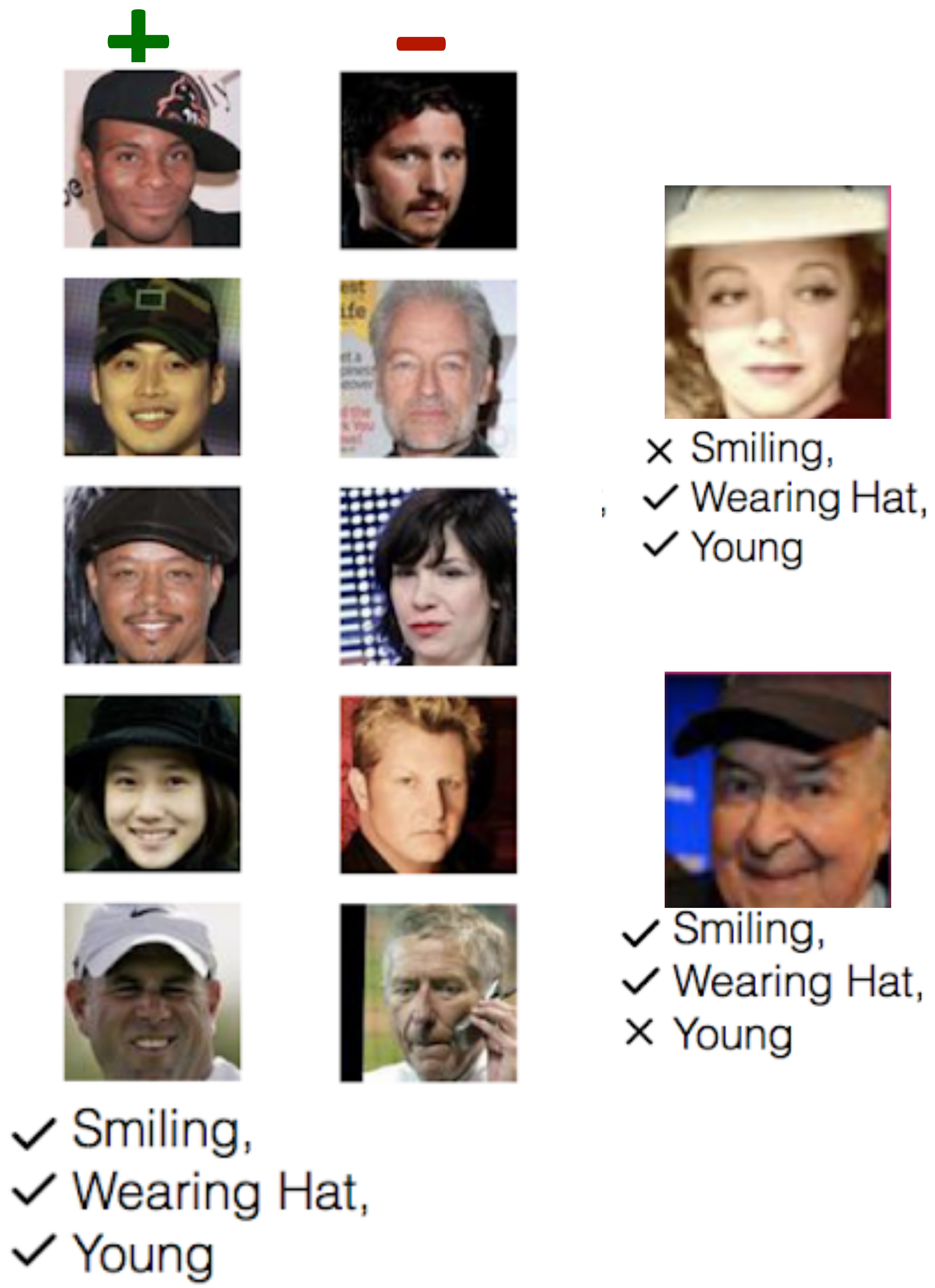
Goal: Can we model and train more complex distributions?

When might we want more complex distributions?

- **generative models** of images, text, video, or other data
- represent **uncertainty over labels** (e.g. ambiguity arising from limited data, partial observability)
- represent **uncertainty over** ***functions***

"HD Video: Riding a horse in the park at sunrise"

Meta-learning methods represent a **deterministic** $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$ (i.e. a point estimate)



Why/when is this a problem?

Few-shot learning problems may be ***ambiguous***.
(even with prior)

Can we learn to ***generate hypotheses*** about the underlying function?
i.e. sample from $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$

Important for:

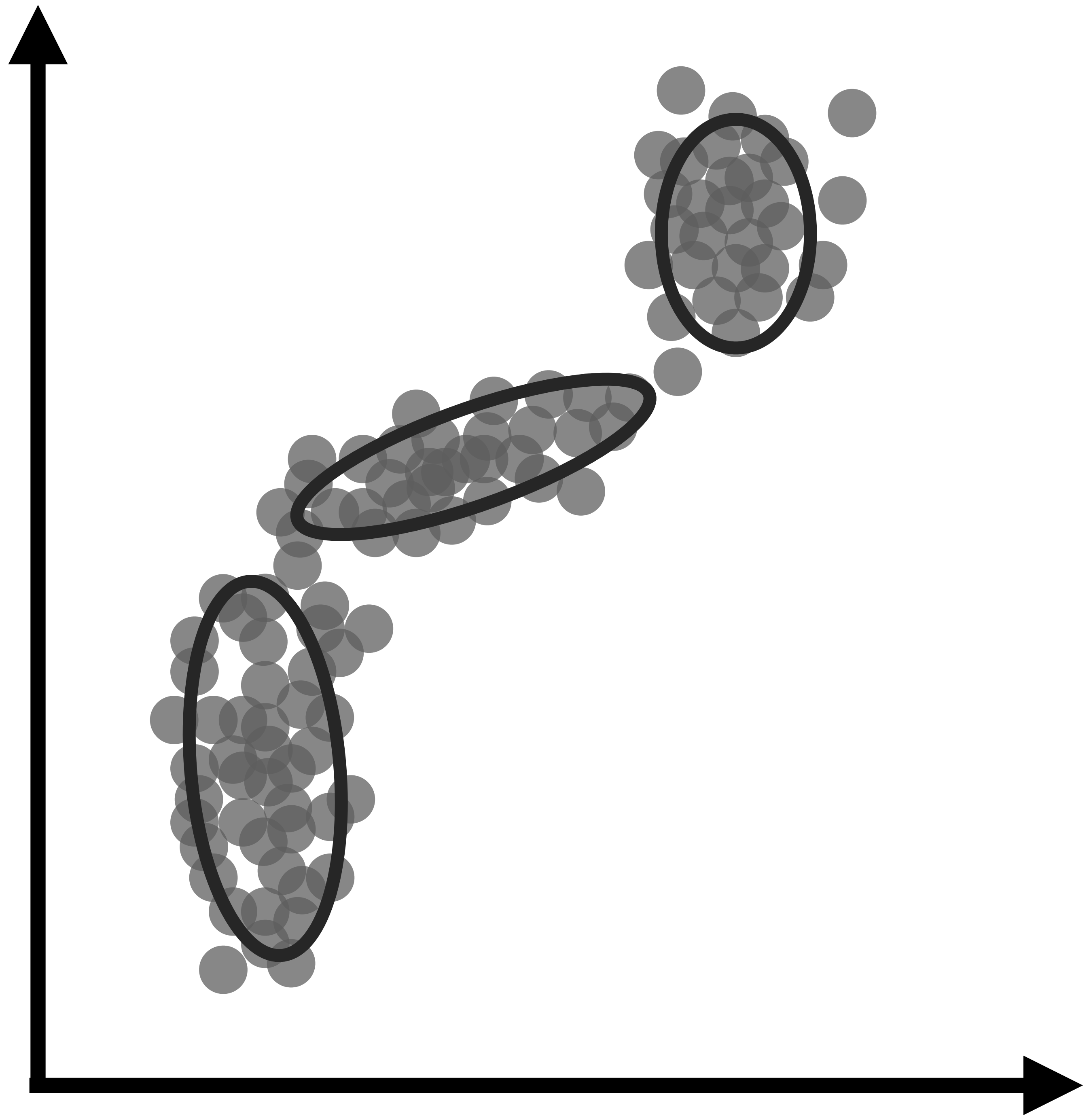
- **safety-critical** few-shot learning (e.g. medical imaging)
- learning to **actively learn**
- learning to **explore** in meta-RL

Active learning w/ meta-learning: Woodward & Finn '16, Konyushkova et al. '17, Bachman et al. '17

Goal: Can we model and train complex distributions?

Latent variable models: examples

$p(x)$



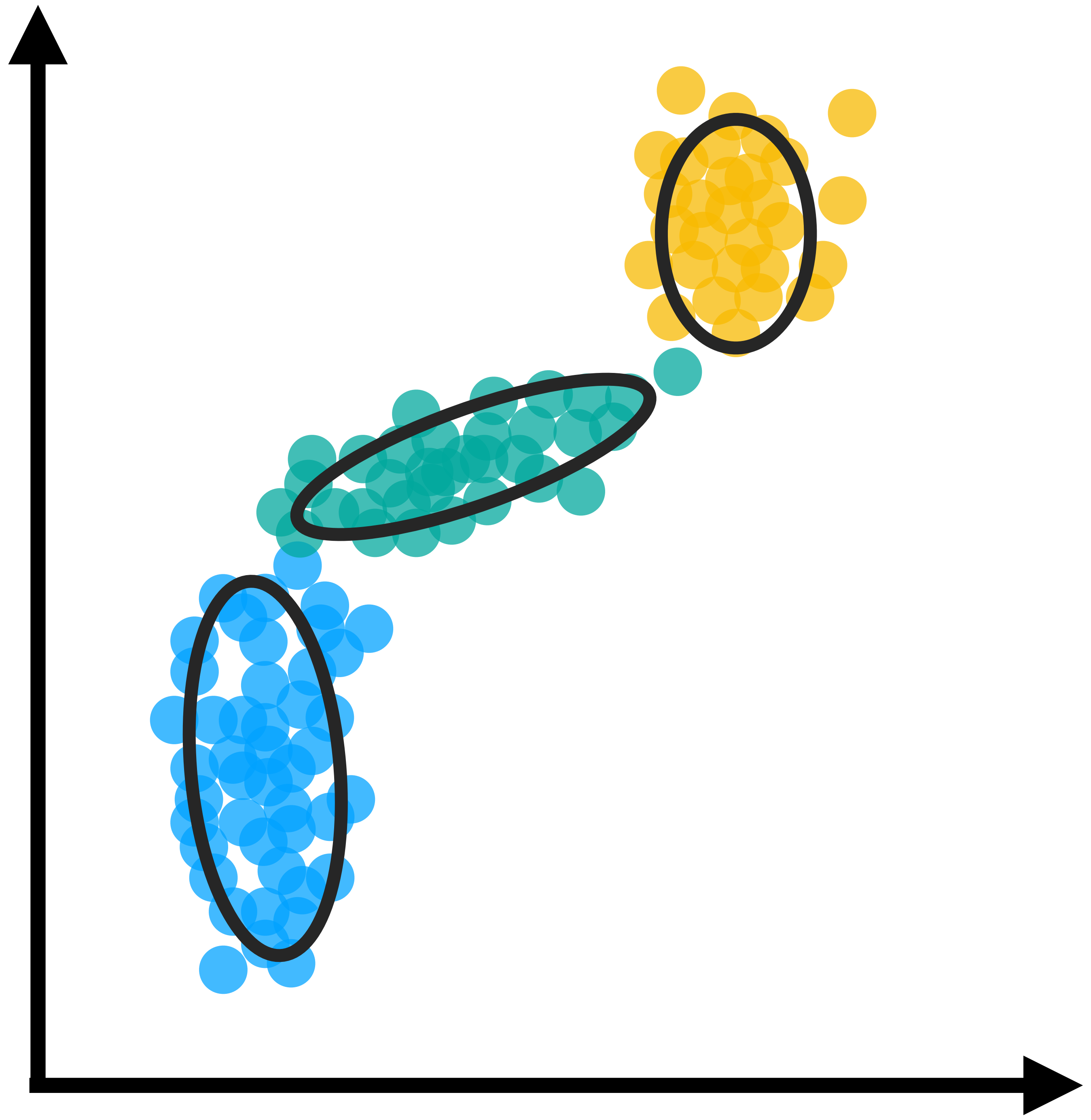
e.g. mixture model

Latent variable models: examples

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

mixture element

e.g. Gaussian



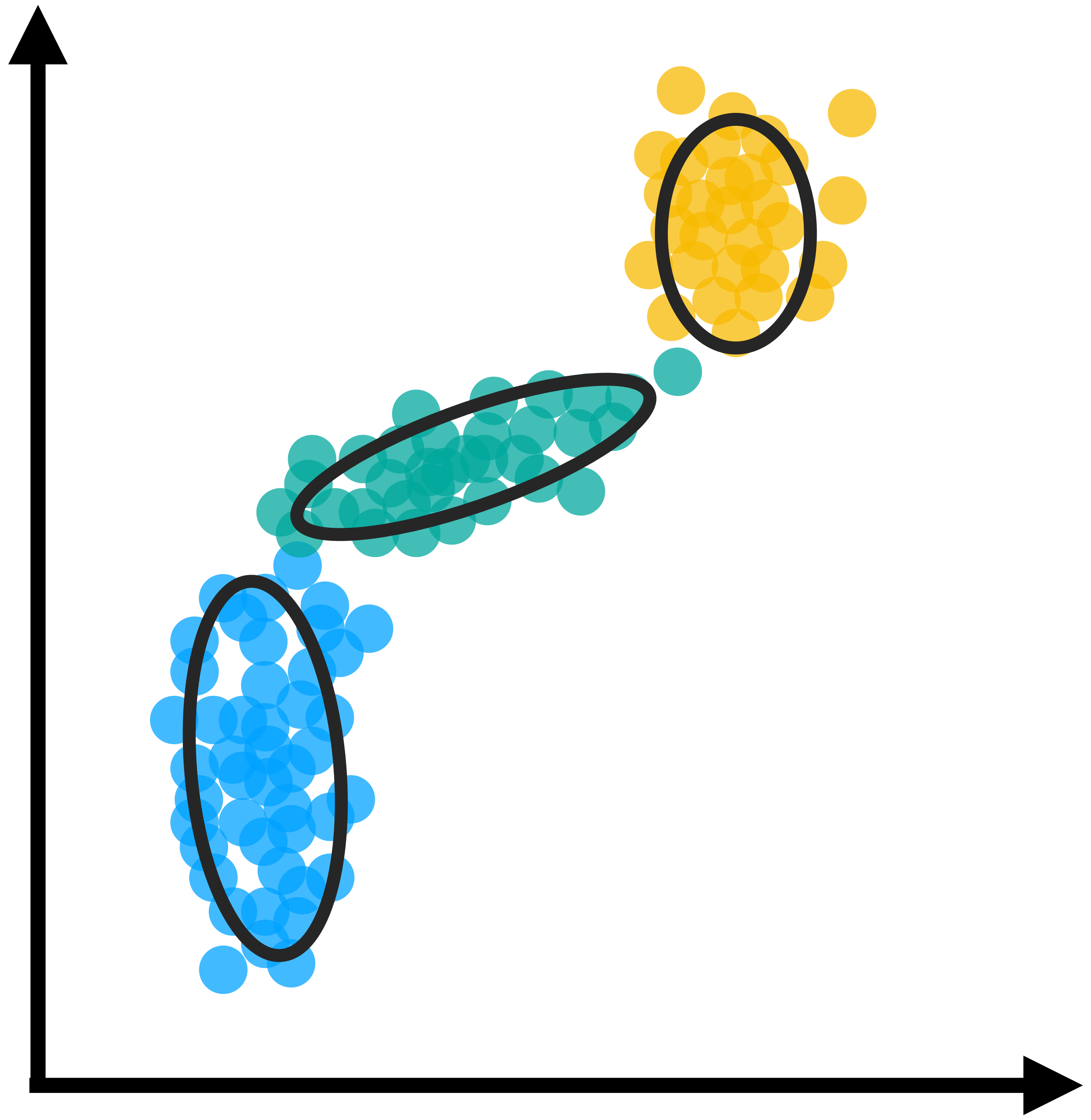
e.g. mixture model

Latent variable models: examples

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

mixture element

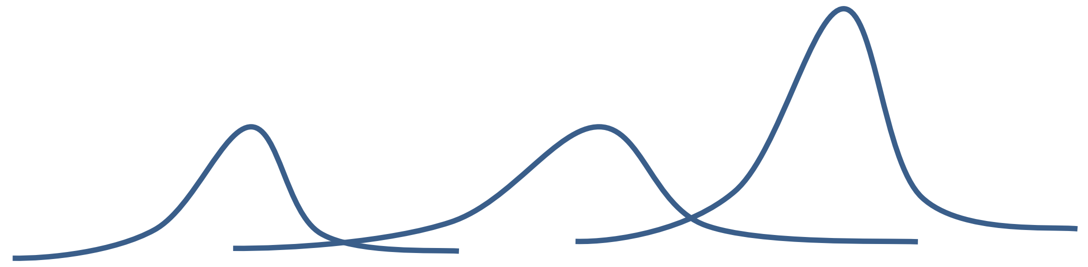
e.g. Gaussian



e.g. mixture model

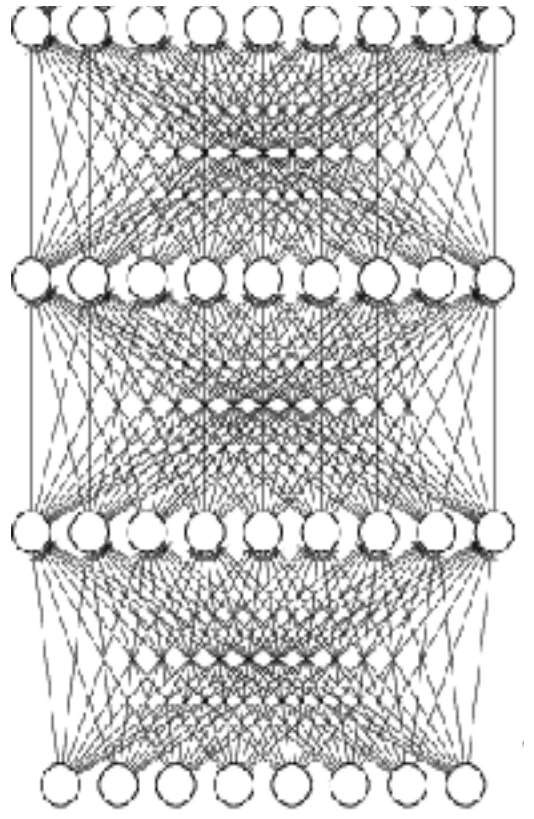
$$p(y|x) = \sum_z p(y|x,z)p(z|x)$$

e.g. mixture density network



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

length of paper



ImageNet Classification with Deep Convolutional Neural Networks

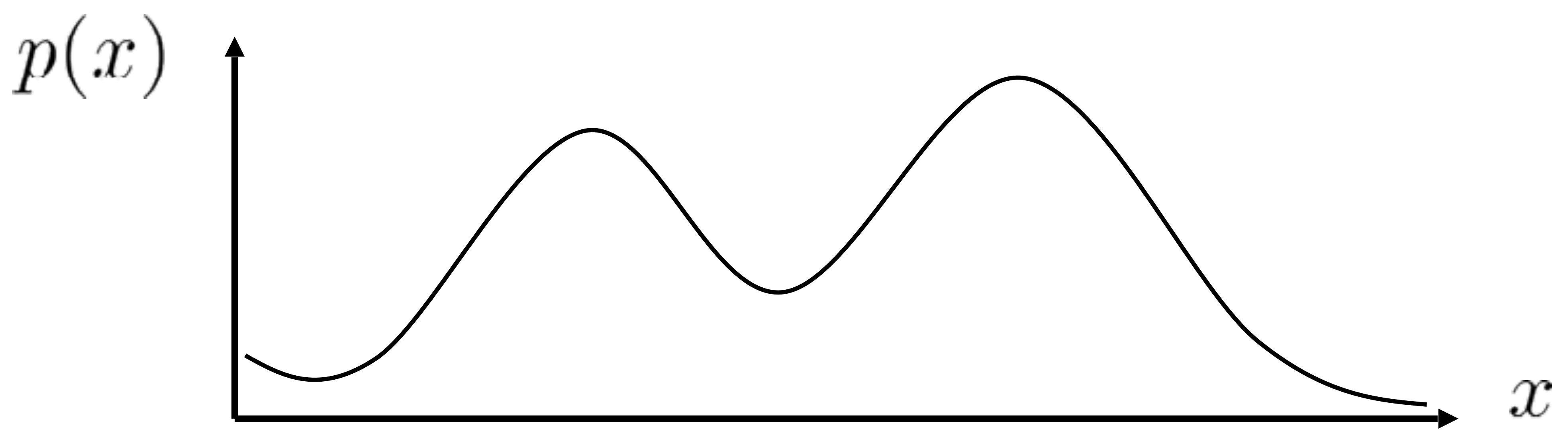
Latent variable models in general

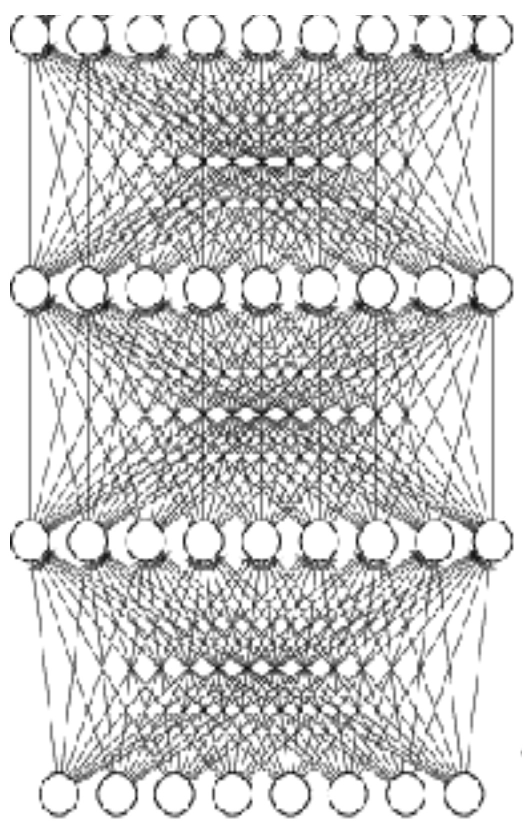
$$p(x) = \int p(x|z)p(z)dz$$

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

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

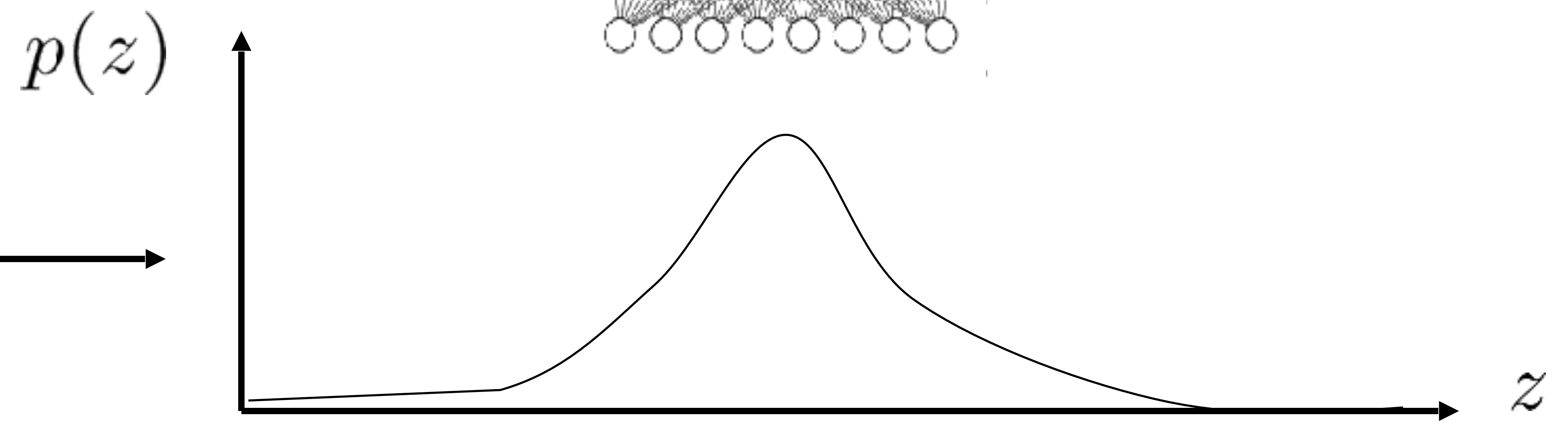
complicated distribution





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

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



💡 Key idea: represent complex distribution by composing two simple distributions

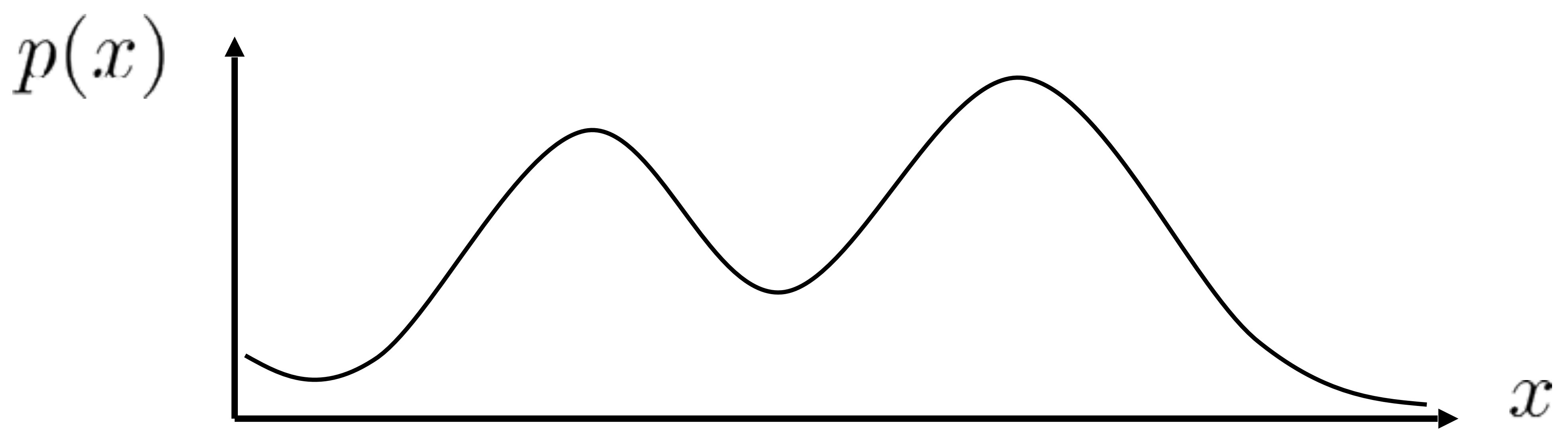
Latent variable models in general

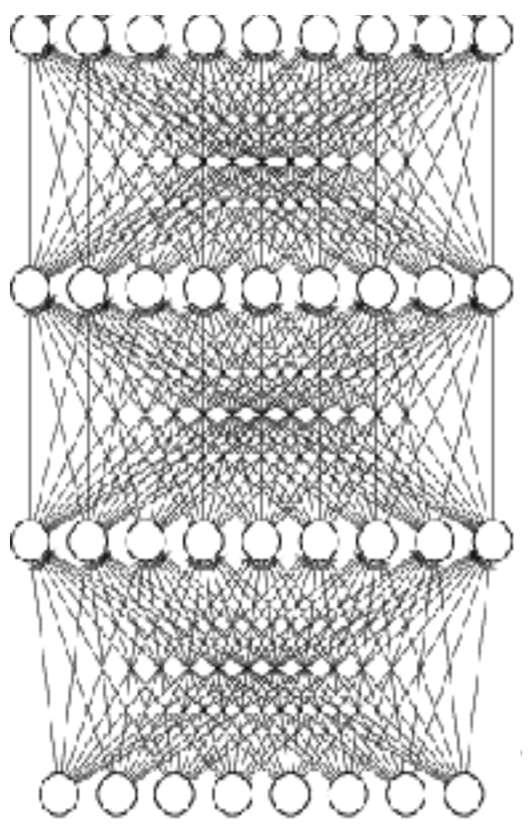
$$p(x) = \int p(x|z)p(z)dz$$

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

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

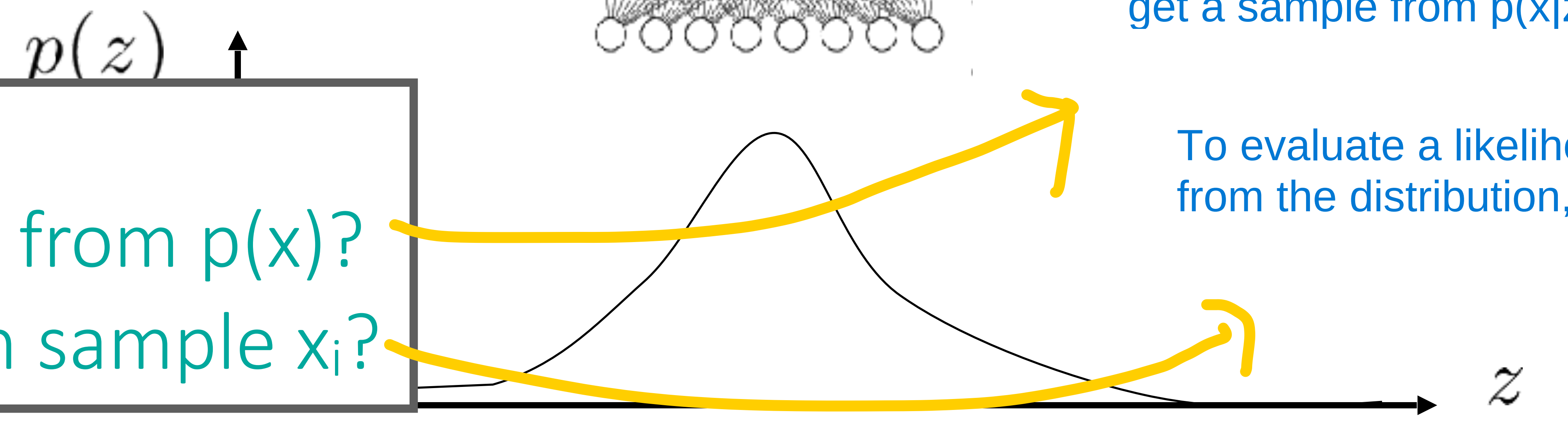
complicated distribution





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

Sample a particular z and for that particular z you can get a sample from p(x|z=Z)



To evaluate a likelihood sample out multiple z's from the distribution, and then evaluate the integral

Questions:
1. Once trained, how do you generate a sample from p(x)?
2. How do you evaluate the likelihood of a given sample x_i?

💡 Key idea: represent complex distribution by composing two simple distributions

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

Flavors of Deep Latent Variable Models

Use latent variables:
- generative adversarial networks (GANs)
- variational autoencoders (VAEs)
- normalizing flow models
- diffusion models

All differ in how they are trained.

Do not use latent variables:
- autoregressive models

(recall generative pre-training lecture)

Let X be a random variable and g(X) be a function of that random variable then
E(g(X)) = summation (g(X) pmf(X))
and here in our case in the defination below the pmf = p(X)

Variational Inference

A. Formulate **a lower bound** on the log likelihood objective.
B. Check **how tight** the bound is.
C. Variational inference -> *Amortized* variational inference
D. How to **optimize**

Estimating the log-likelihood

H(X) = -E(log(P(X))) $= -\int p(x) \log p(x) \, dx$

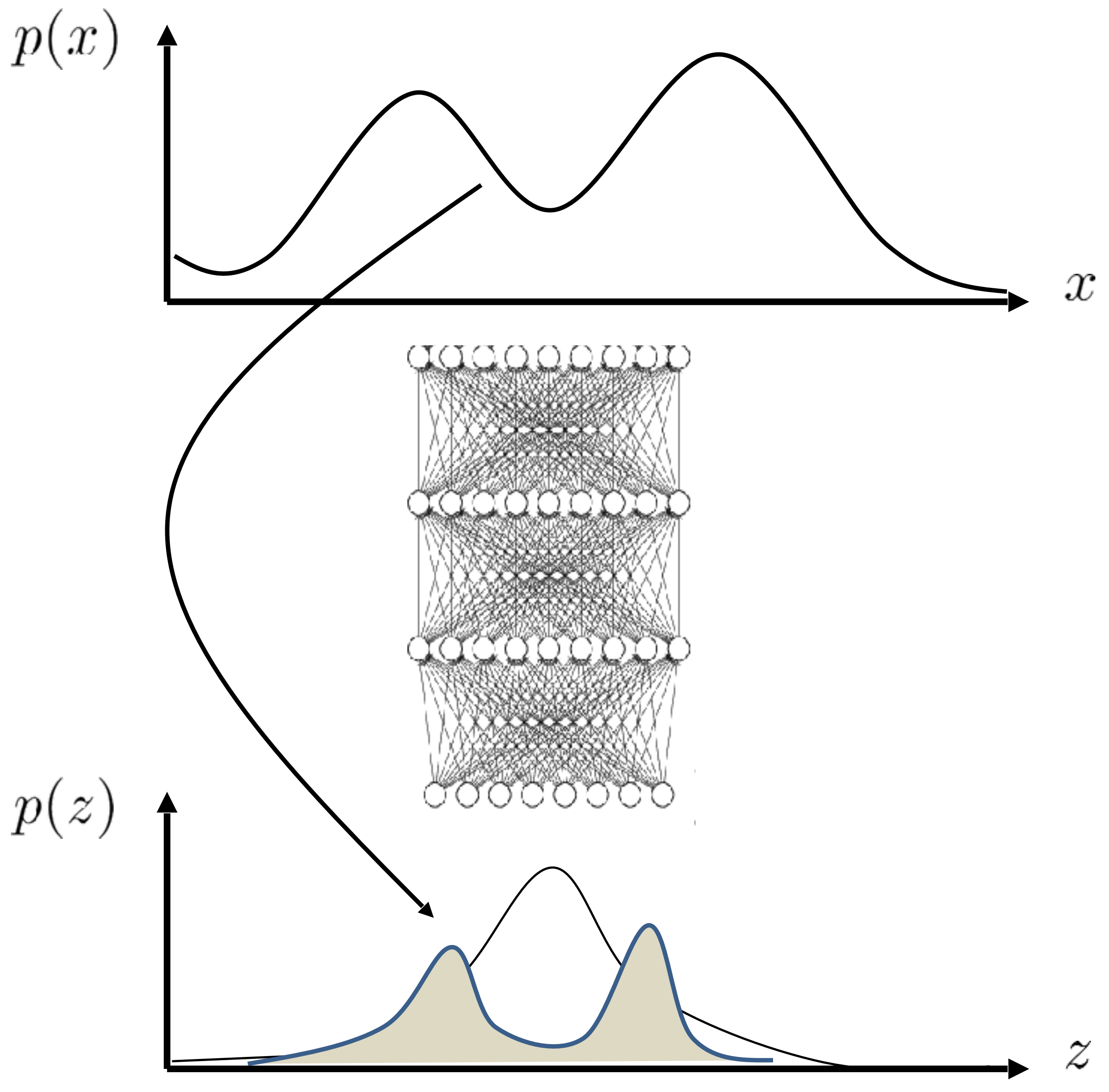
alternative: *expected* log-likelihood:

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

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 $p(z|x_i)$?

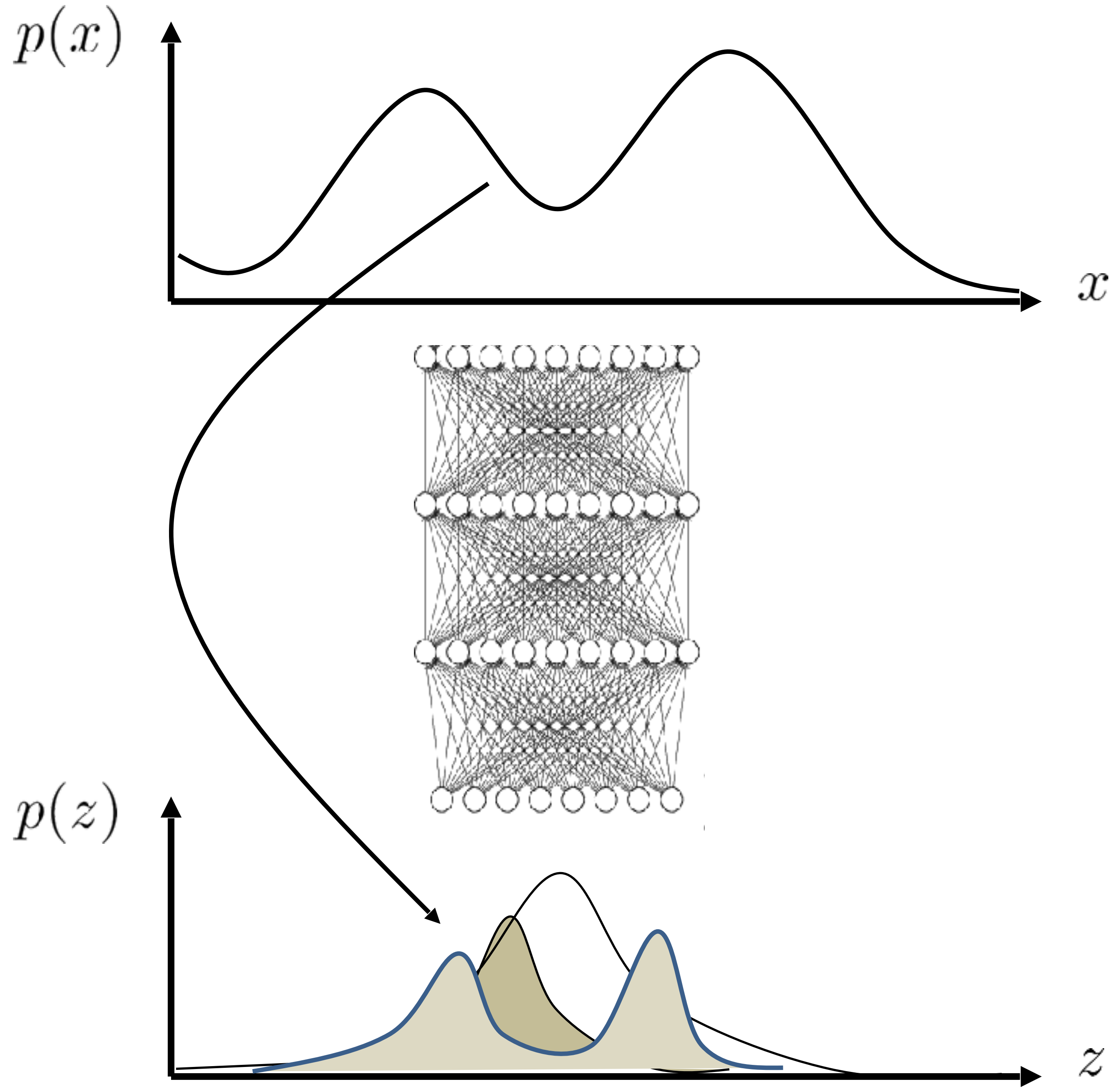
what if we approximate with $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

can bound $\log p(x_i)$!

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

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

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



The variational approximation

Jensen's inequality

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

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

can bound $\log p(x_i)$!

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

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

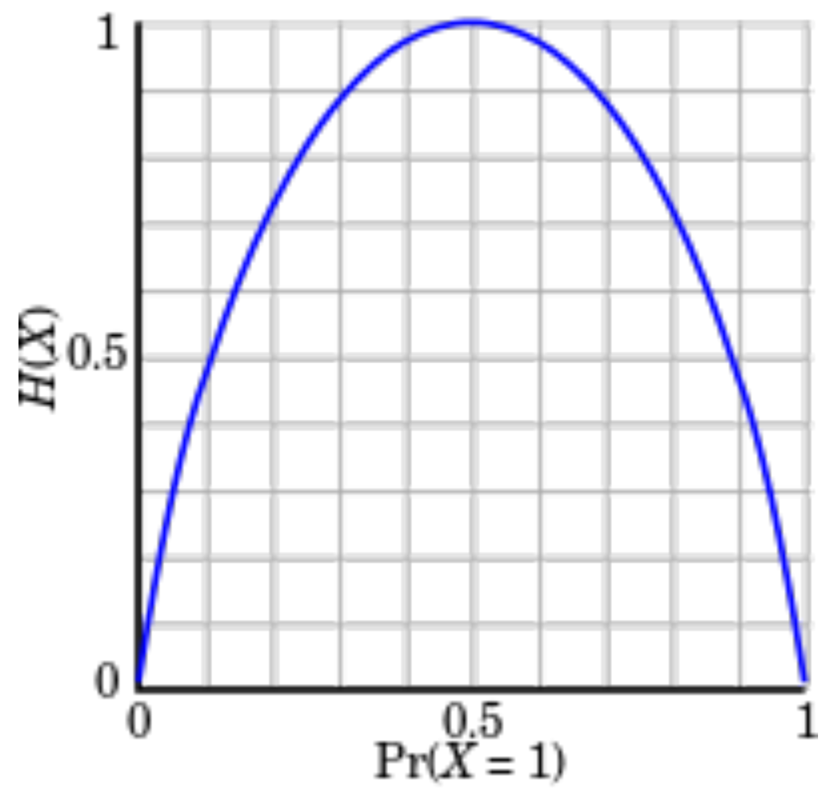
$$= \log E_{z\sim q_i(z)}\left[\frac{p(x_i|z)p(z)}{q_i(z)}\right]$$

maximizing this maximizes $\log p(x_i)$

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

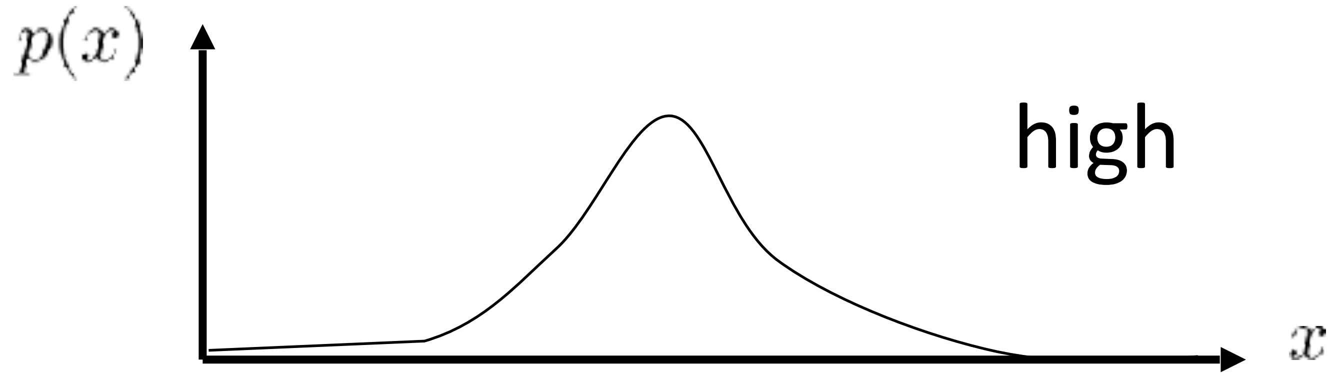
"evidence lower bound" (ELBO)

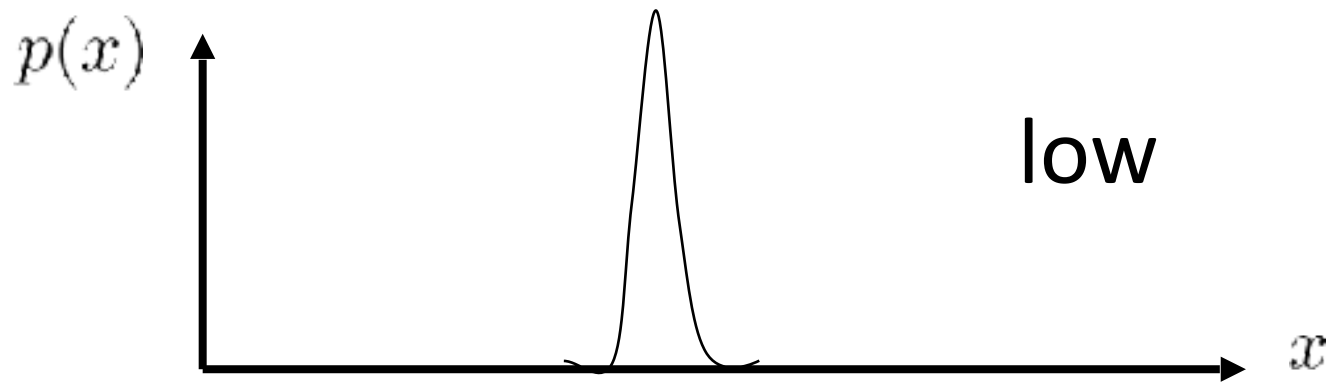
A brief aside...



Entropy:

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





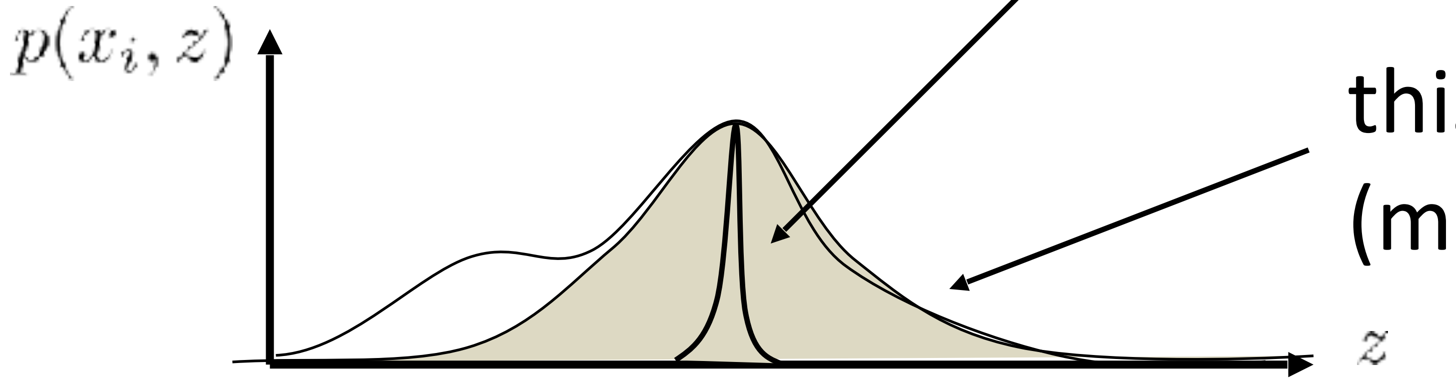
Intuition 1: how *random* is the random variable?

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:

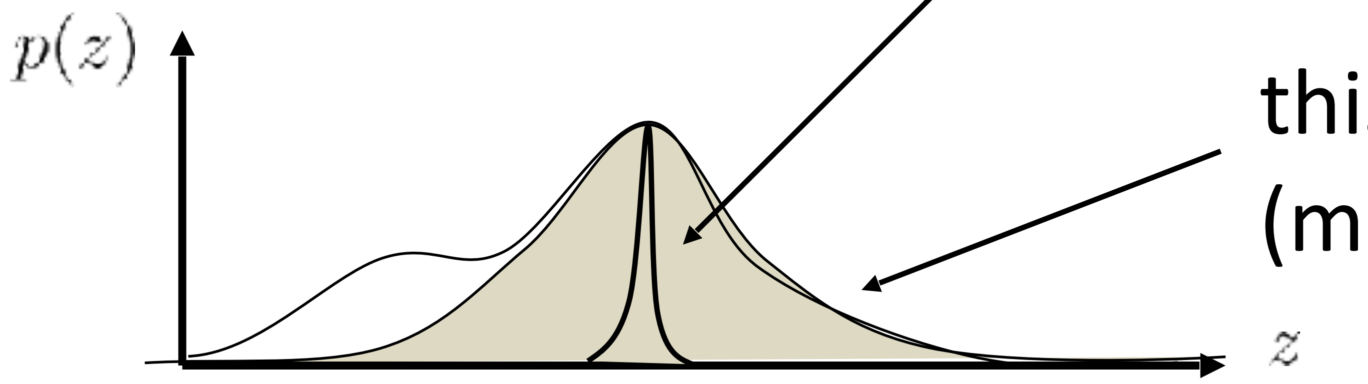
$$D_{\mathrm{KL}}(q||p) = E_{x\sim q(x)}\left[\log \frac{q(x)}{p(x)}\right] = E_{x\sim q(x)}[\log q(x)] - E_{x\sim q(x)}[\log p(x)] = -E_{x\sim q(x)}[\log p(x)] - \mathcal{H}(q)$$

Intuition 1: how *different* are two distributions? e.g. when q=p, KL divergence is 0

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)

How tight is the lower bound?

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

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

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?

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

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

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

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

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

Note 1: If KL divergence is 0, then bound is tight.

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

How tight is the lower bound?

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

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

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?

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

Note 2: Maximizing $L(p, q_i)$ w.r.t. q_i minimizes the KL divergence.

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

Note 1: If KL divergence is 0, then bound is tight.

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

Optimization objective: $\max_{\theta, q_i} \frac{1}{N} \sum_i \mathcal{L}_i(p_\theta, q_i)$

Imp (Both θ, q_i)

Optimizing the ELBO

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

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

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

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

for each x_i (or mini-batch):

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

sample $z \sim q_i(z)$ Here they are sampling one 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

Question: How many parameters are there?

in terms of $|\theta|$, $|z|$, N

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

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 $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

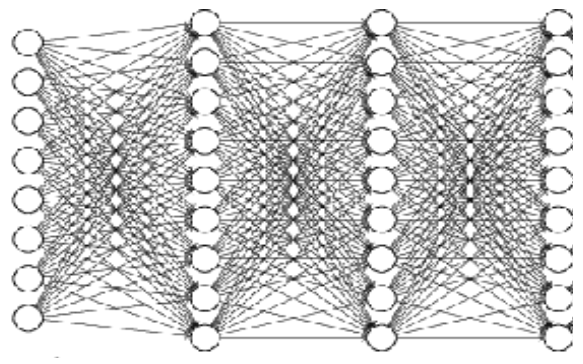
use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

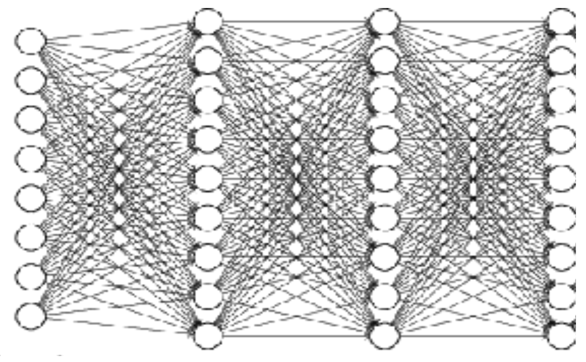
gradient ascent on μ_i, σ_i

Question: How many parameters are there?

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

A. Formulate **a lower bound** on the log likelihood objective.
B. Check **how tight** the bound is.
C. Variational inference -> *Amortized* variational inference
D. How to **optimize**

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

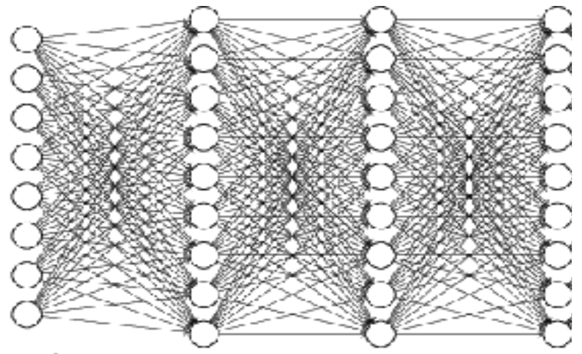
let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

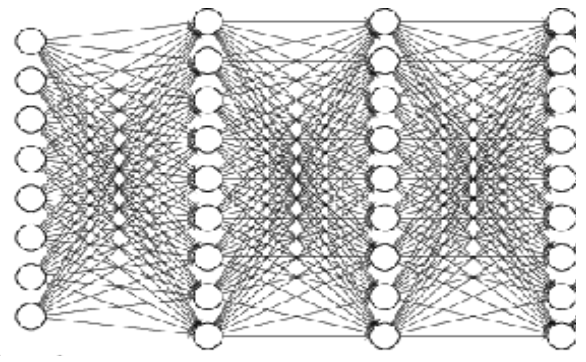
use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on μ_i, σ_i

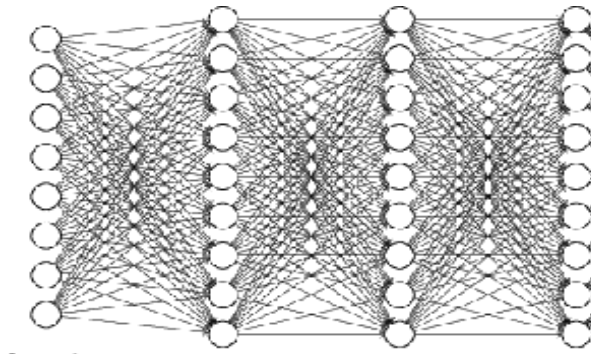
Question: How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

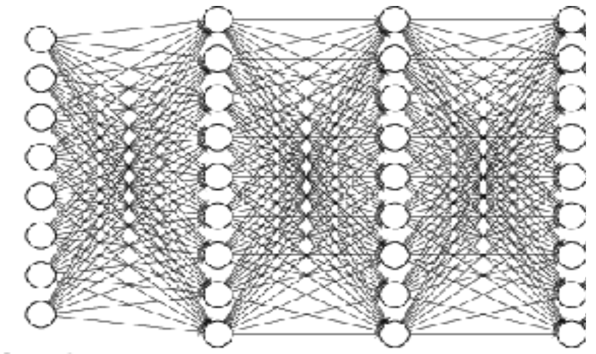
intuition: $q_i(z)$ should approximate $p(z|x_i)$ what if we learn a *network* $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

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

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

The reparameterization trick

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

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

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

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

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

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

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

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

independent of ϕ!

\+ Very simple to implement

\+ Low variance

\- Only continuous latent variables

Discrete latent variables:

- vector quantization & straight-through estimator (“VQ-VAE”)
- policy gradients / “REINFORCE”

Another way to look at everything...

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

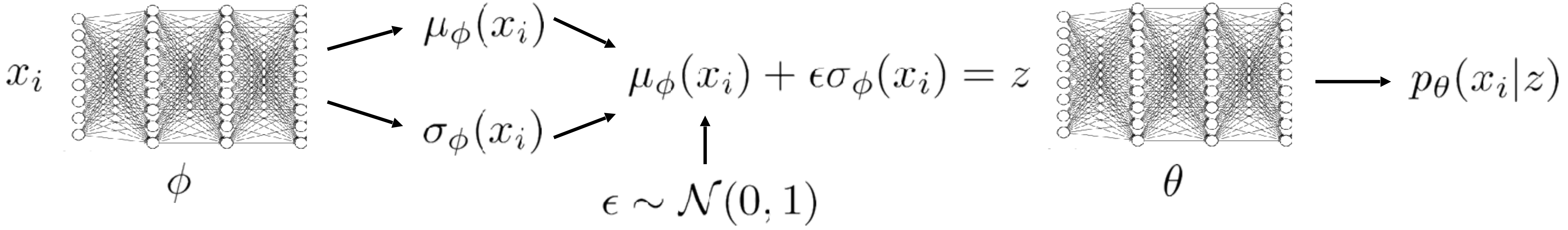
$$= E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] + \underbrace{E_{z\sim q_\phi(z|x_i)}[\log p(z)] + \mathcal{H}(q_\phi(z|x_i))}_{-D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))}$$

this has a convenient analytical form for Gaussians

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

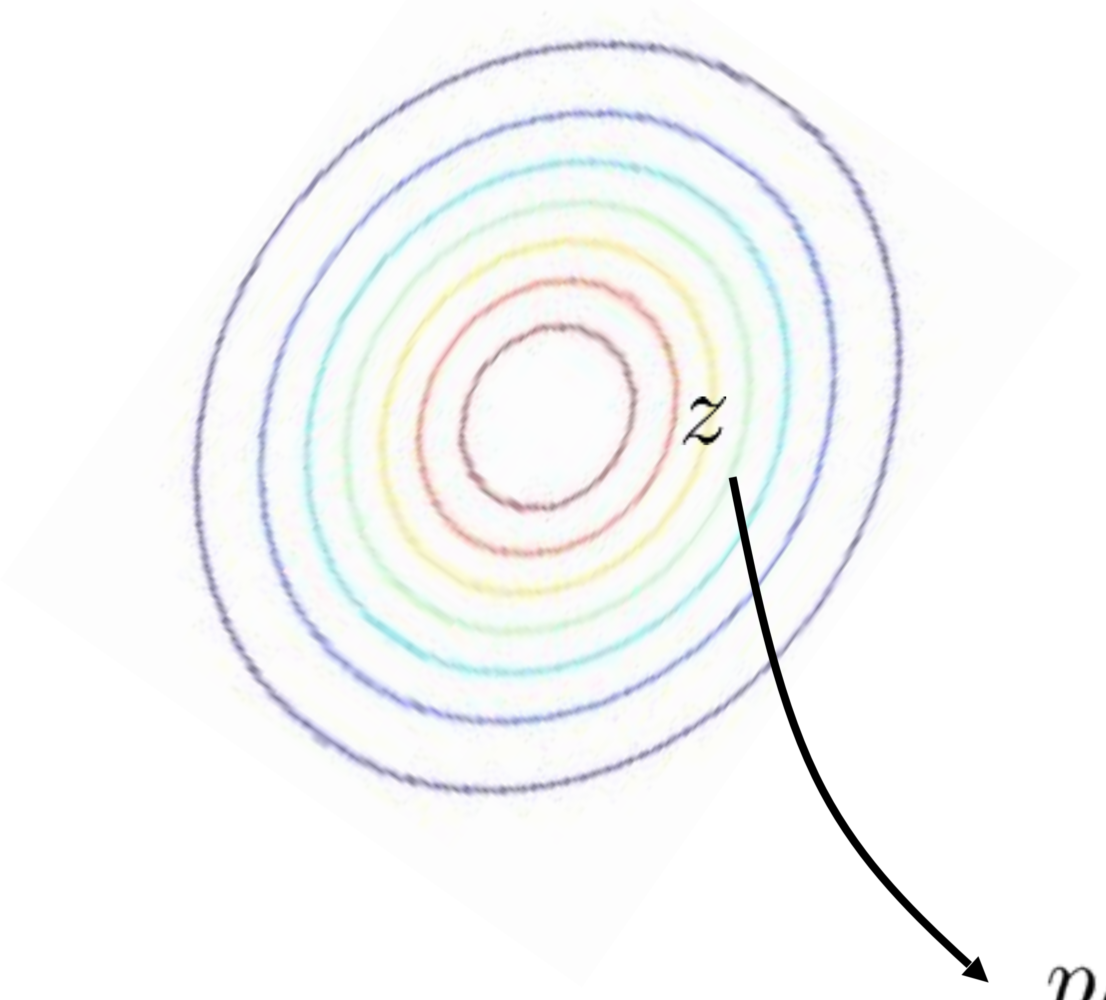
$$= E_{\epsilon\sim\mathcal{N}(0,1)}[\log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i))] - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

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



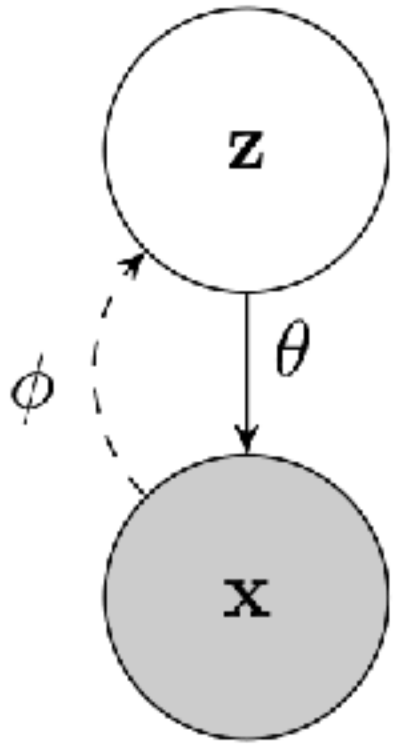
Example Models

The variational autoencoder



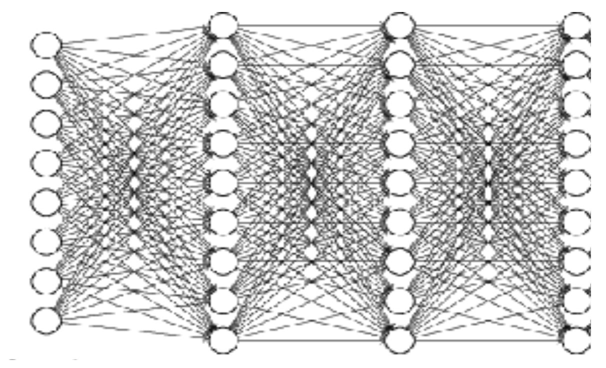
$p_\theta(x|z)$

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



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

x_i



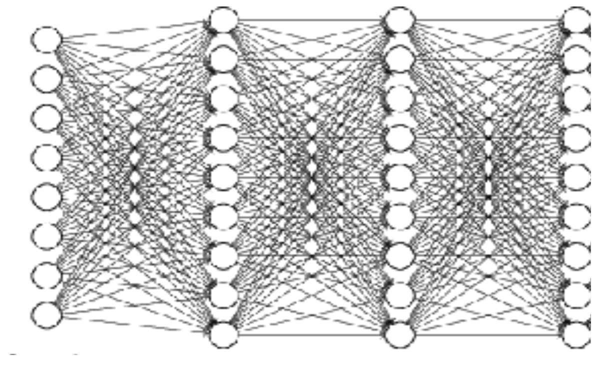
ϕ

$\mu_\phi(x_i)$

$\sigma_\phi(x_i)$

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

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



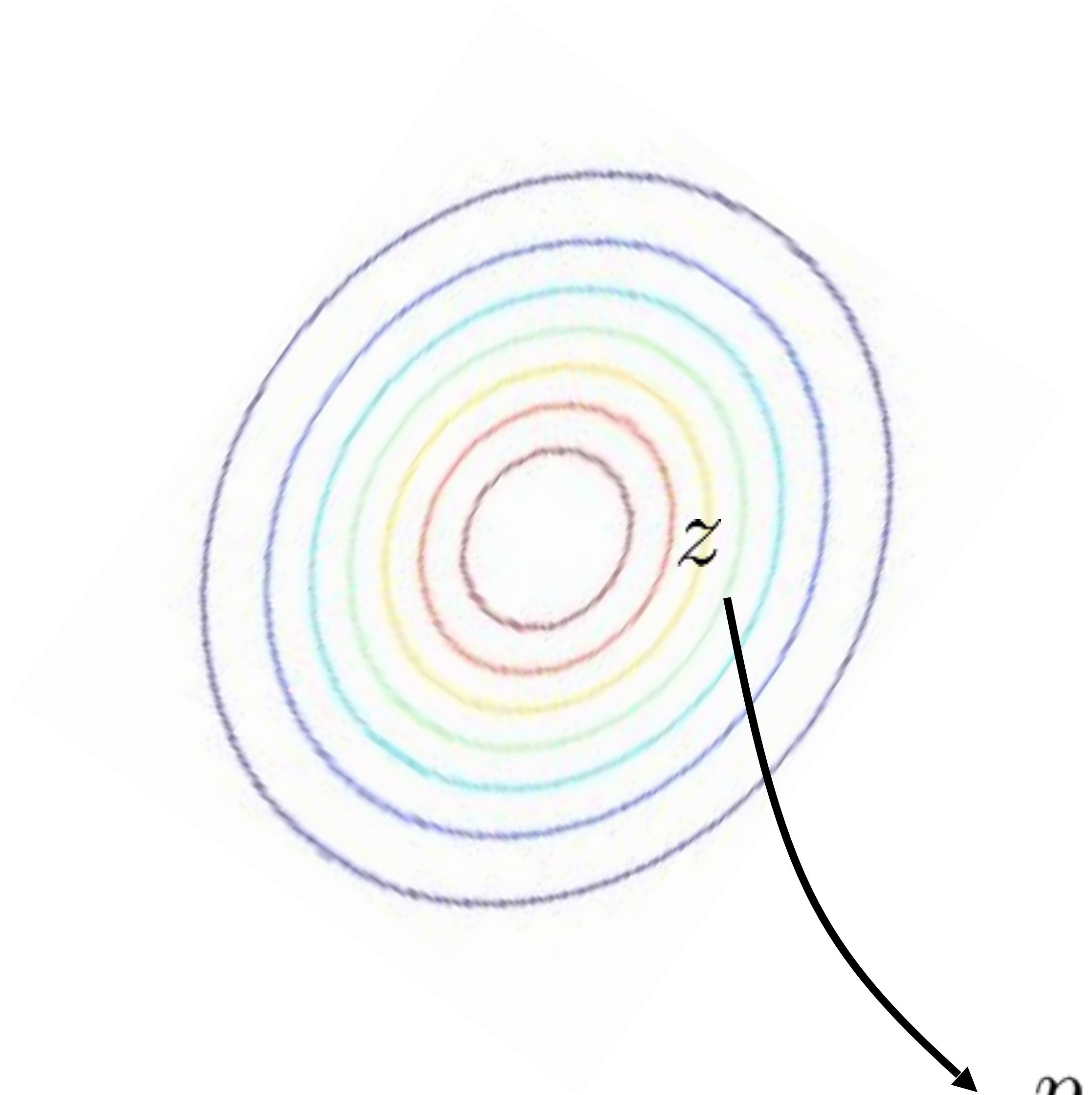
θ

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

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

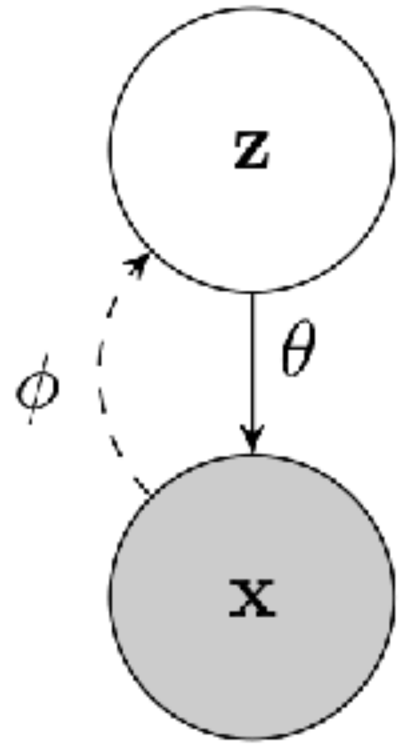
This KL Divergence term acts as a regularizer that ensures that the posterior distribution is closed to the prior ensuring that too much info is not from a single example

Using the variational autoencoder

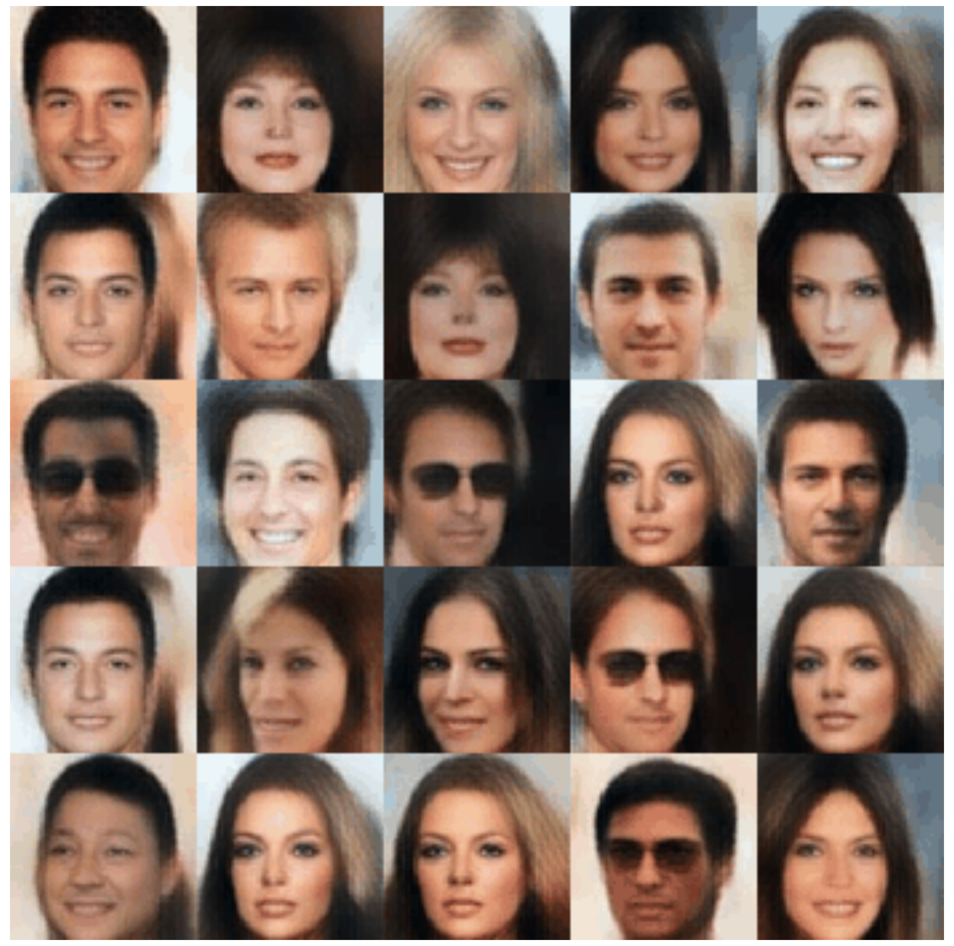


$p_\theta(x|z)$

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



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



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

why does this work?

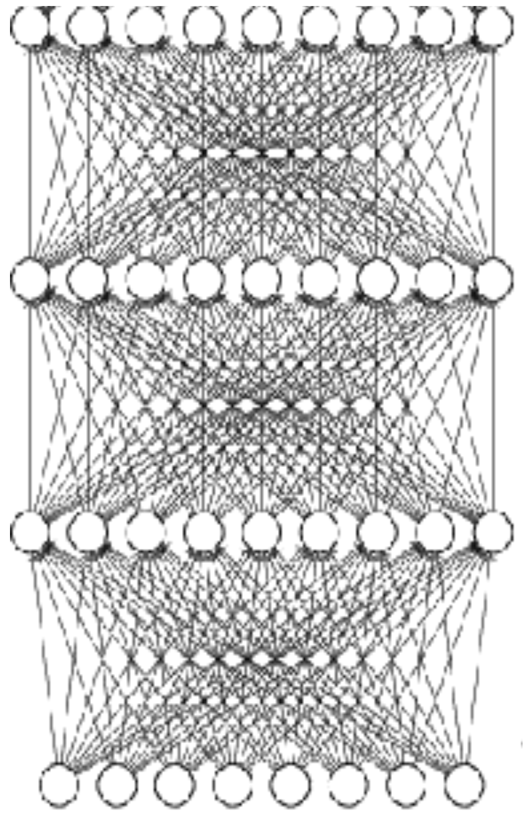
sampling:

$$z \sim p(z)$$

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

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

Conditional models



$p(y|x, z)$

class 109
(brain coral)

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

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

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

$p(z)$

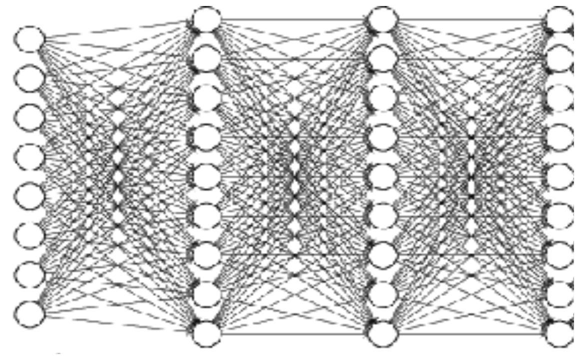
can *optionally* depend on x

at test time:

$z \sim p(z|x_i)$

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

x_i
y_i



ϕ

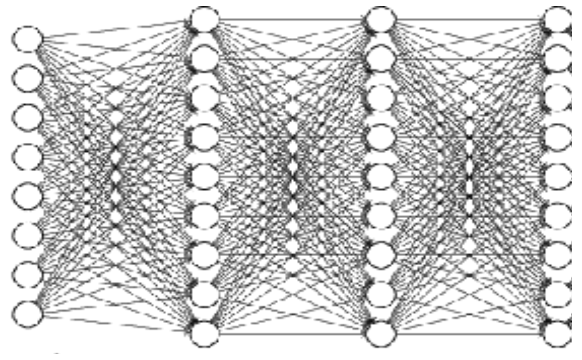
$\mu_\phi(x_i, y_i)$

$\sigma_\phi(x_i, y_i)$

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

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

x_i



θ

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

Images from Razavi, van den Oord, Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. ‘19

Plan for Today

1. Latent variable models
2. Variational inference
3. Amortized variational inference
4. Example latent variables models

Part of (optional) Homework 4

Goals

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

Course Reminders

Homework 3 due Monday next week.

Tutorial session on **Thursday 4:30 pm**

Next time: Bayesian meta-learning

Be careful Azure usage — turning off machines when you are not using them!