

Variational Inference and Generative Models

CS 330

10/31/2022

Course Reminders

Homework 3 due ~~Wednesday~~ Friday.

Following up on some high-res feedback:

- Extension on HW3, added clarifications & documentation links in initial HW3 Ed post
- Project mentor role clarification

This Week

A Bayesian perspective on meta-learning

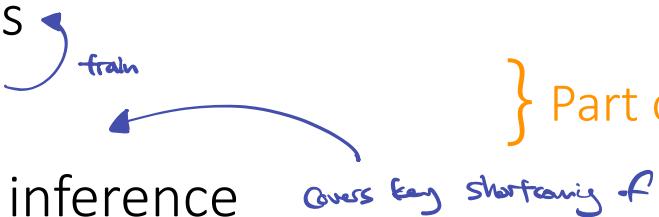
Today: Approximate Bayesian inference via variational inference



Bayes is back.

Plan for Today

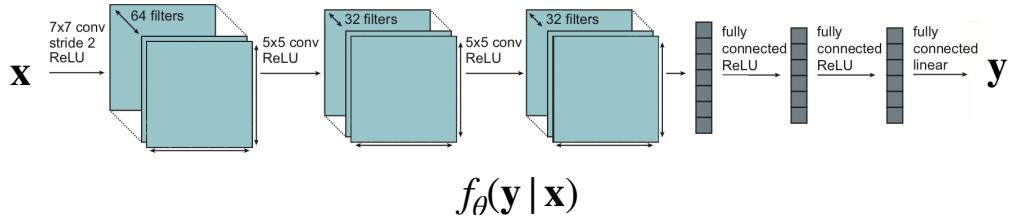
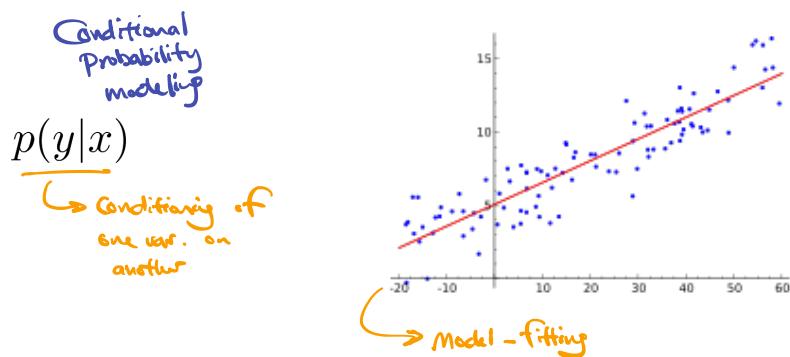
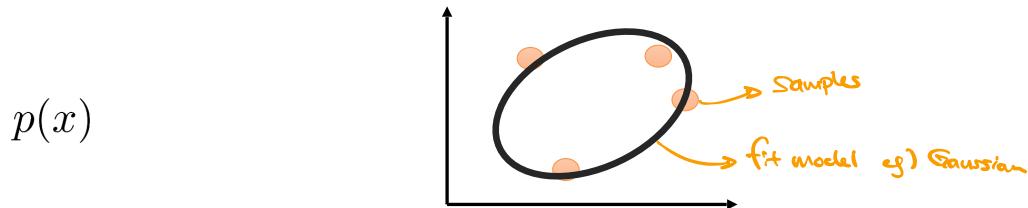
1. Latent variable models
2. Variational inference
3. Amortized variational inference
4. Example latent variables models



Goals

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

Probabilistic models



Most commonly:

- probability values of discrete **categorical distribution**
 - mean and variance of a **Gaussian**
- But it could be other distributions!

Simple examples
of prob. dist

How do we train probabilistic models?

the model: $p_\theta(x)$

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

maximum likelihood fit:

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

"best" fit

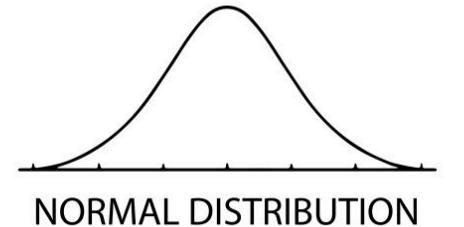
Easy to evaluate & differentiate for categorical or Gaussian distributions.

i.e. cross-entropy, MSE losses

↳ Max. likelihood for
categorical var.

↳ for regression
(e.g.) fitting a $\sim N(\mu, \sigma)$

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)
 - ↳ See next slide
- represent uncertainty over functions

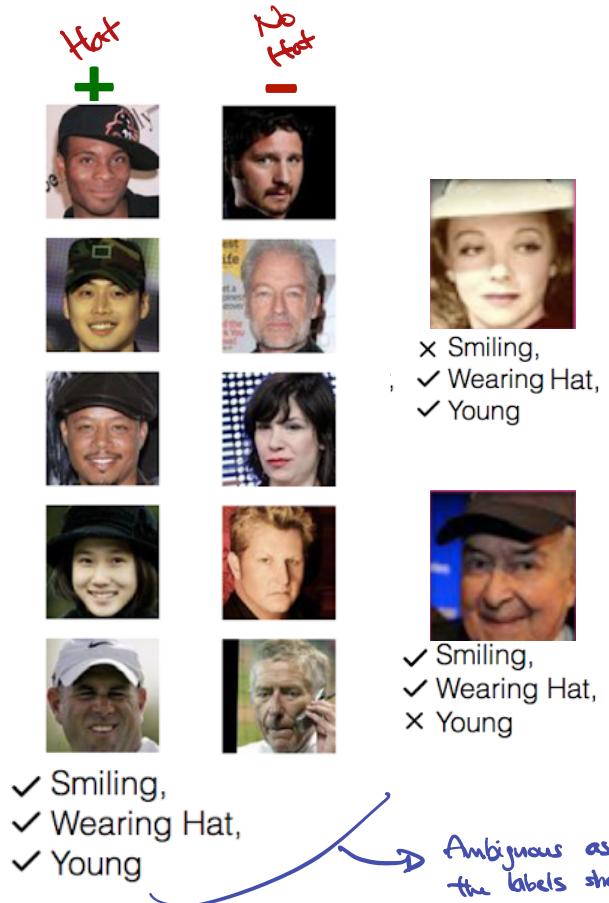
True distributions over labels
might be a complex one.
e.g.) multimodal distrib.

“HD Video: Riding a horse
in the park at sunrise”



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

Task-specific params (ϕ_i)
given $\mathcal{D}_i^{\text{tr}}$ &



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 $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

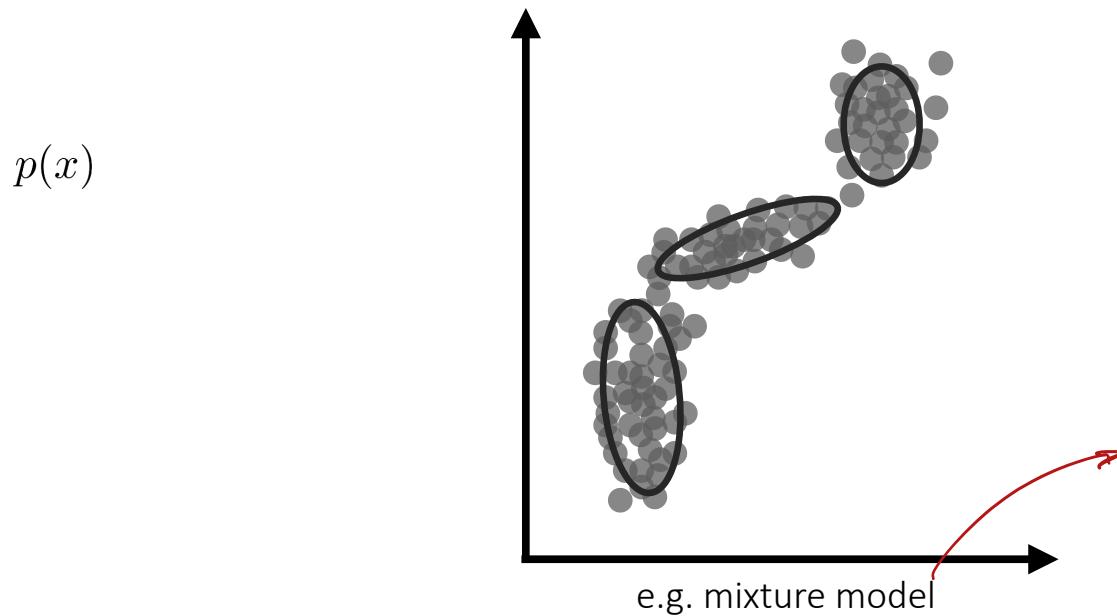
Important for:

- **safety-critical** few-shot learning
(e.g. medical imaging) → Useful to know the confidence of predictions
- learning to **actively learn** → algo tells you if it needs more training data
- learning to **explore** in meta-RL → knows what parts of the env. it needs to explore w/ confident arms

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

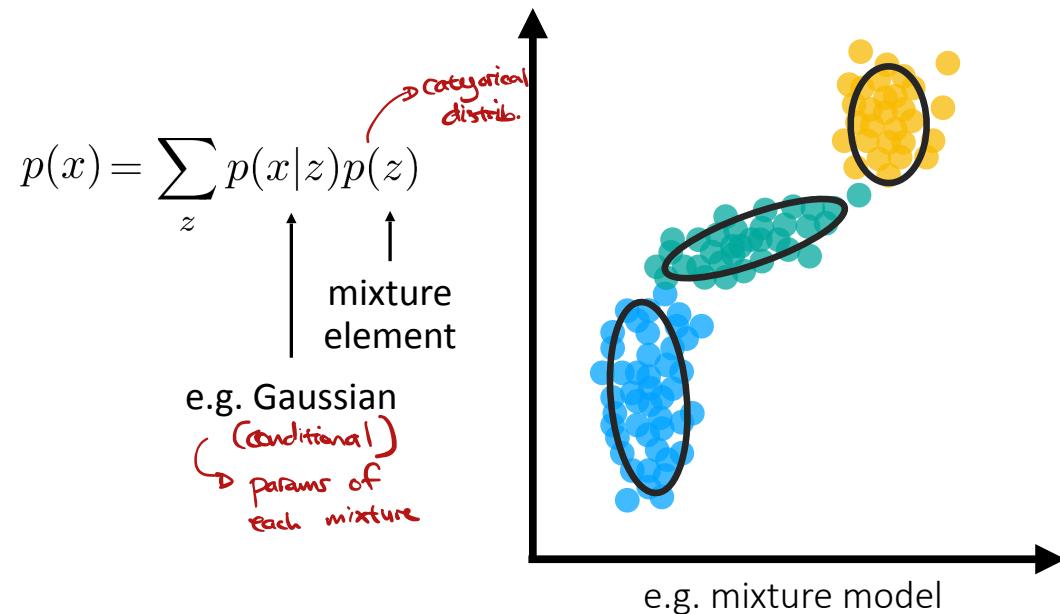
Goal: Can we model and train complex distributions?

Latent variable models: examples

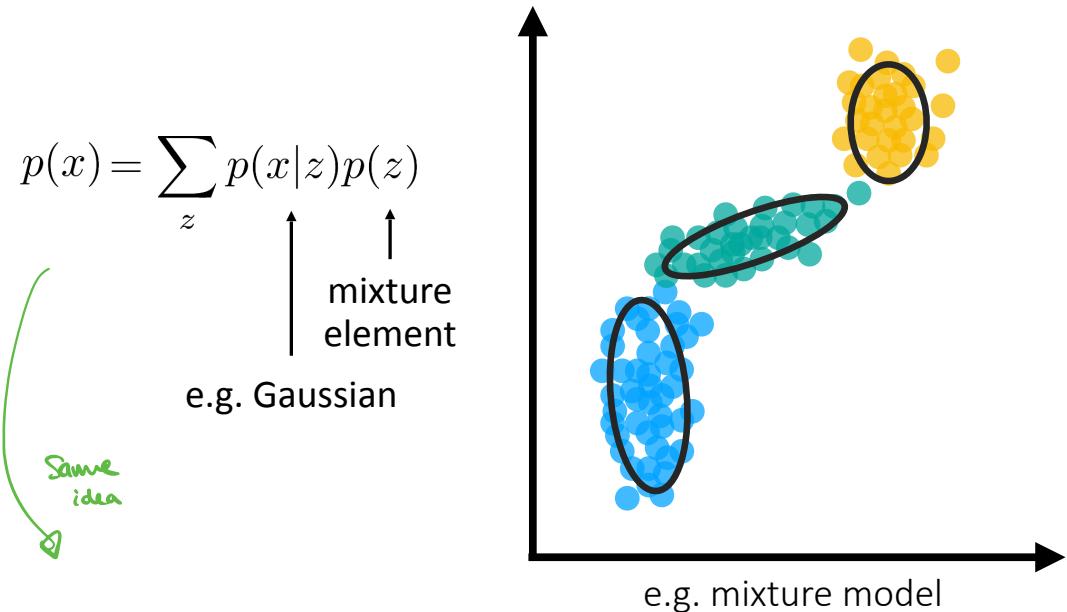


few diff. components (e.g. Gaussian mixture model)
You don't know the underlying structure in which
data points are organized (which data points
correspond to which mixture component), but
- through modeling process you'll get to both
mixture components - the params of
each mixture component.

Latent variable models: examples



Latent variable models: examples

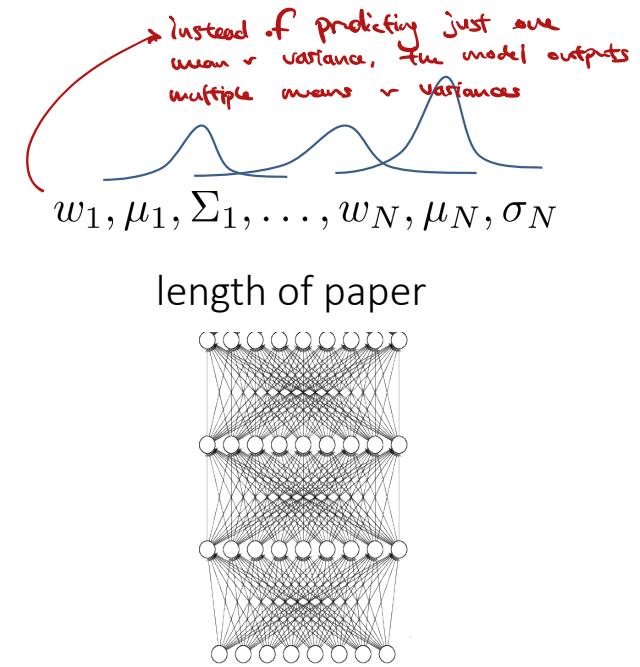


Same idea

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

e.g. mixture density network

↳ Same thing, fancier name



ImageNet Classification with Deep Convolutional Neural Networks



Latent variable models in general

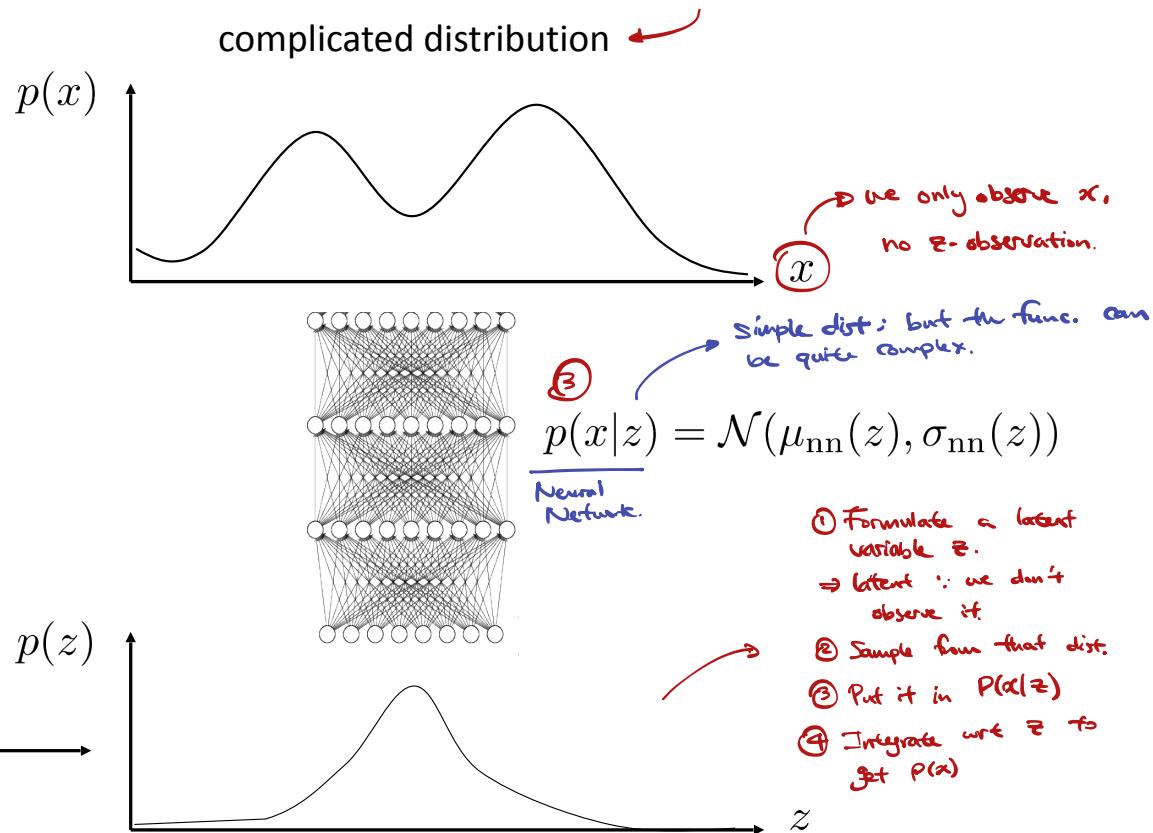
Goal: Model this!

can model for pretty much
any $p(x) \rightarrow$ v. high expressive power.

$$\textcircled{1} \quad p(x) = \int p(x|z)p(z)dz$$

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

“easy” distribution
(e.g., Gaussian)



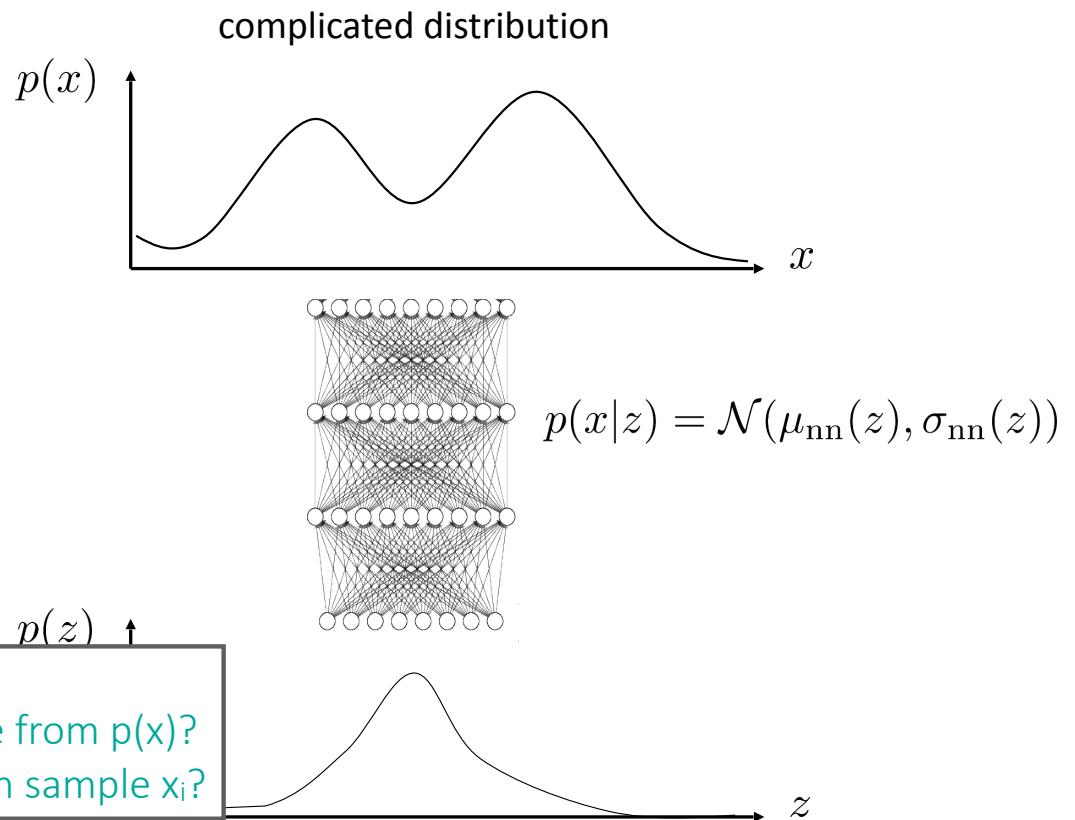
Key idea: represent complex distribution by composing two simple distributions

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)



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

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

∴ Need too many samples from z dist
To get one value of gradient
at each time step

→ Need a way to estimate the gradient
w/o having to evaluate the integral

⊗ Likelihood of parameter values occurring given the observed data.
Assumes params are unknown.

$L(\theta|x)$

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)

Diff. pros & cons

Latent Variables

- Typically latent variables don't have explicitly interpretable meaning to them.
- Sometimes when the data naturally has certain clusters that are related to interpretability, it will reflect it.
(↳ eg) face generator : may coincide w/ certain attributes (eyes slope, skin col. etc.)

- Generally we want the latent variables to correspond to meaningful features. No guarantee though.
- For conditional latent models, i.e. $P(y|x, z)$
the lat. var. z is the only source of randomness (other than the input x).
So z needs to capture things in y that can't be found in 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

alternative: expected log-likelihood:

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

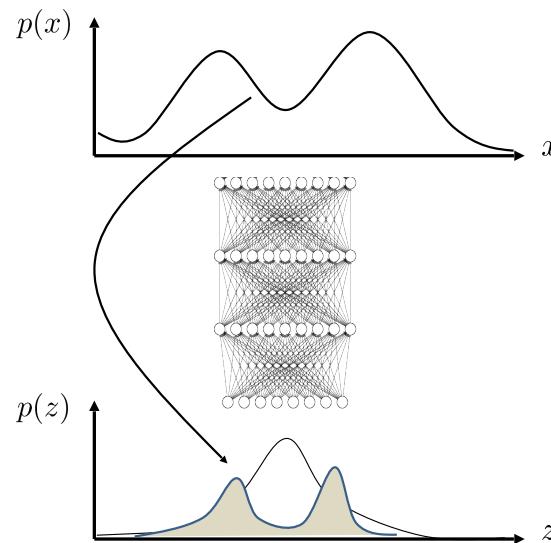
*replaced the integral
w/ an expectation.*

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 $p(z|x_i)$

↳ “posterior”

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



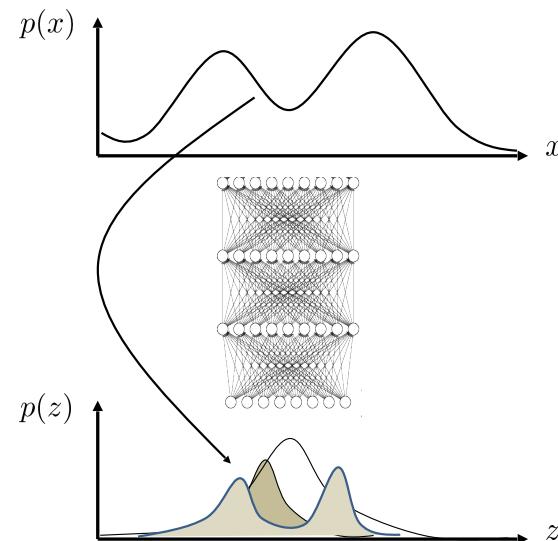
The variational approximation

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 $\log p(x_i)$!

$$\begin{aligned}\log p(x_i) &= \log \int_z p(x_i|z)p(z) dz \\ &= \log \int_z p(x_i|z)p(z) \frac{q_i(z)}{q_i(z)} dz \\ &= \log E_{\substack{z \sim q_i(z) \\ z \text{ Sampled from } g_i(z)}} \left[\frac{p(x_i|z)p(z)}{q_i(z)} \right] \\ &\xrightarrow{\text{We now have } g_i(z) \text{ we can sample from, instead of having to evaluate a complex integral}}\end{aligned}$$



The variational approximation

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

can bound $\log p(x_i)!$

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

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

We want the expectation to be outside of the log, so we can sample mini-batches instead of a ton of z's

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

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

"evidence lower bound" (ELBO)

~~JKL~~ ...

Jensen's inequality

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

- ↳ Also works for any concave function (other than \log)
- ↳ Convex works too, just flip the sign

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

maximizing this maximizes $\log p(x_i)$

↳ "entropy"

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?

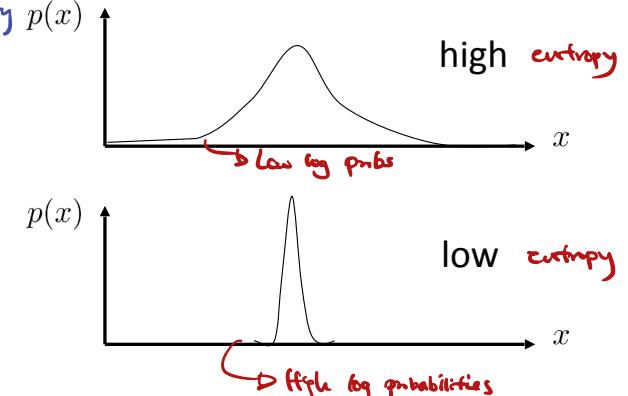
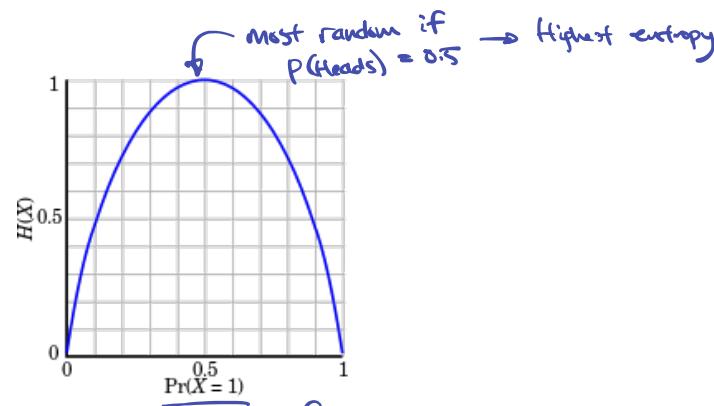
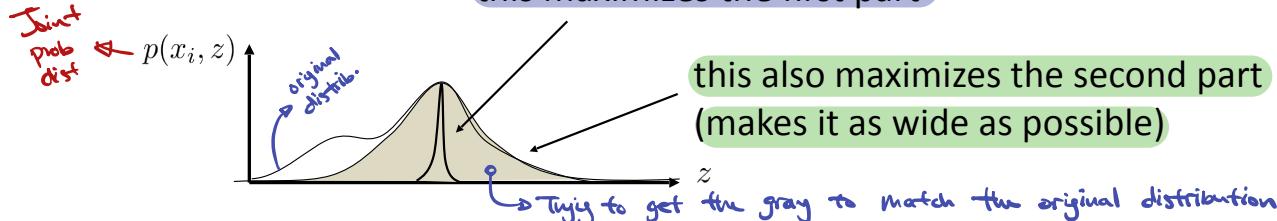
Intuition 2: how large is the log probability in expectation under itself

what do we expect this to do?

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

⊕ for a given z , ?
 $p(x_i|z)$ should be high
 \rightarrow tries to pick a z that has a high $p(x_i|z)$.

this maximizes the first part



$$\begin{aligned} & \log p(x_i|z) + \log p(z) \\ &= \log(p(z) \cdot p(x_i|z)) \\ &= \log[p(z) \wedge p(x_i|z)] \rightarrow \log p(x_i, z) \end{aligned}$$

A brief aside...

KL-Divergence:

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

Not exactly true, $\therefore D_{\text{KL}}(q\|p) + D_{\text{KL}}(p\|q)$

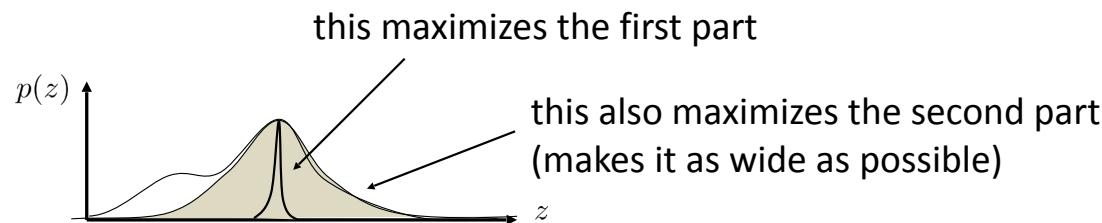
Intuition 1: how *different* are two distributions? e.g. when $q=p$, KL divergence is 0

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

↳ How much info is lost when we use one prob. dist. to approximate another.

wtf?

why entropy?



How tight is the lower bound?

$$\log p(x_i) \geq \overbrace{E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}^{\mathcal{L}_i(p, q_i) \text{ “evidence lower bound” (ELBO)}}$$

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

why?

$$\begin{aligned} D_{\text{KL}}(q_i(z)\|p(z|x_i)) &= E_{z \sim q_i(z)} \left[\log \frac{q_i(z)}{p(z|x_i)} \right] = E_{z \sim q_i(z)} \left[\log \frac{q_i(z)p(x_i)}{p(x_i, z)} \right] \\ &= -E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] + E_{z \sim q_i(z)}[\log q_i(z)] + E_{z \sim q_i(z)}[\log p(x_i)] \\ &= -E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] - \mathcal{H}(q_i) + \log p(x_i) \\ &= -\mathcal{L}_i(p, q_i) + \log p(x_i) \end{aligned}$$

$$\log p(x_i) = D_{\text{KL}}(q_i(z)\|p(z|x_i)) + \mathcal{L}_i(p, q_i) \quad \text{Note 1: If KL divergence is 0, then bound is tight.}$$

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

How tight is the lower bound?

$$\log p(x_i) \geq \overbrace{E_{z \sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}^{\mathcal{L}_i(p, q_i) \text{ “evidence lower bound” (ELBO)}}$$

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

why?

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

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

$$\log p(x_i) = D_{\text{KL}}(q_i(z)\|p(z|x_i)) + \mathcal{L}_i(p, q_i) \quad \text{Note 1: If KL divergence is 0, then bound is tight.}$$

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

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

Optimizing the ELBO

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

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

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

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

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

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?

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

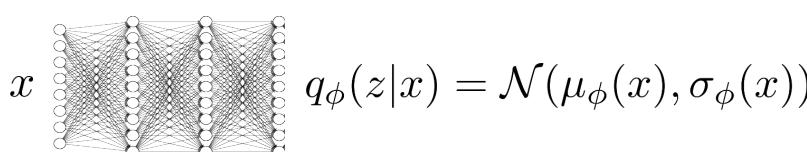
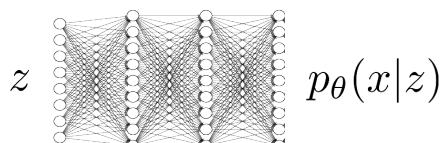
let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

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



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

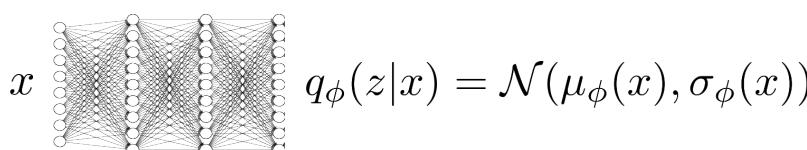
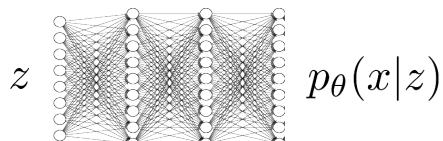
let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

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

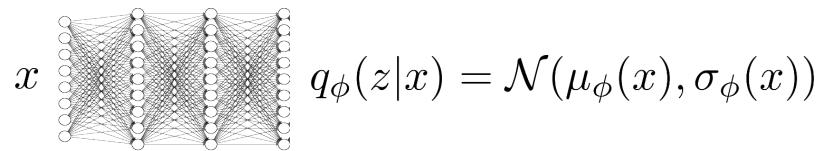
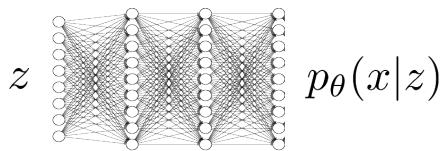
gradient ascent on μ_i, σ_i

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



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?

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

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

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

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

The reparameterization trick

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

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

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 !

+ Very simple to implement

+ Low variance

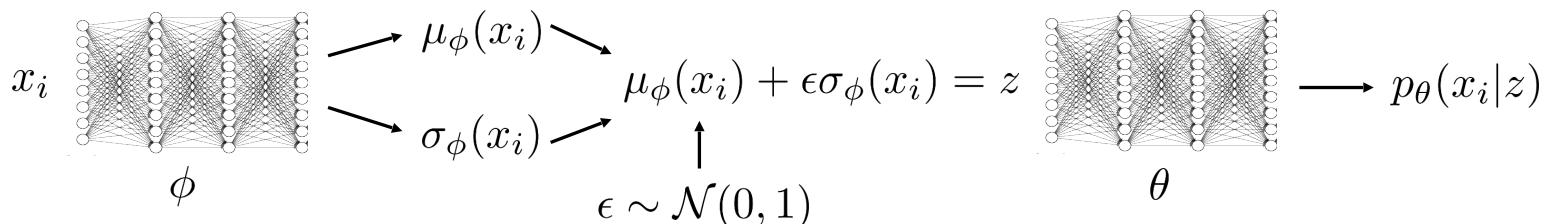
- Only continuous latent variables

Discrete latent variables:

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

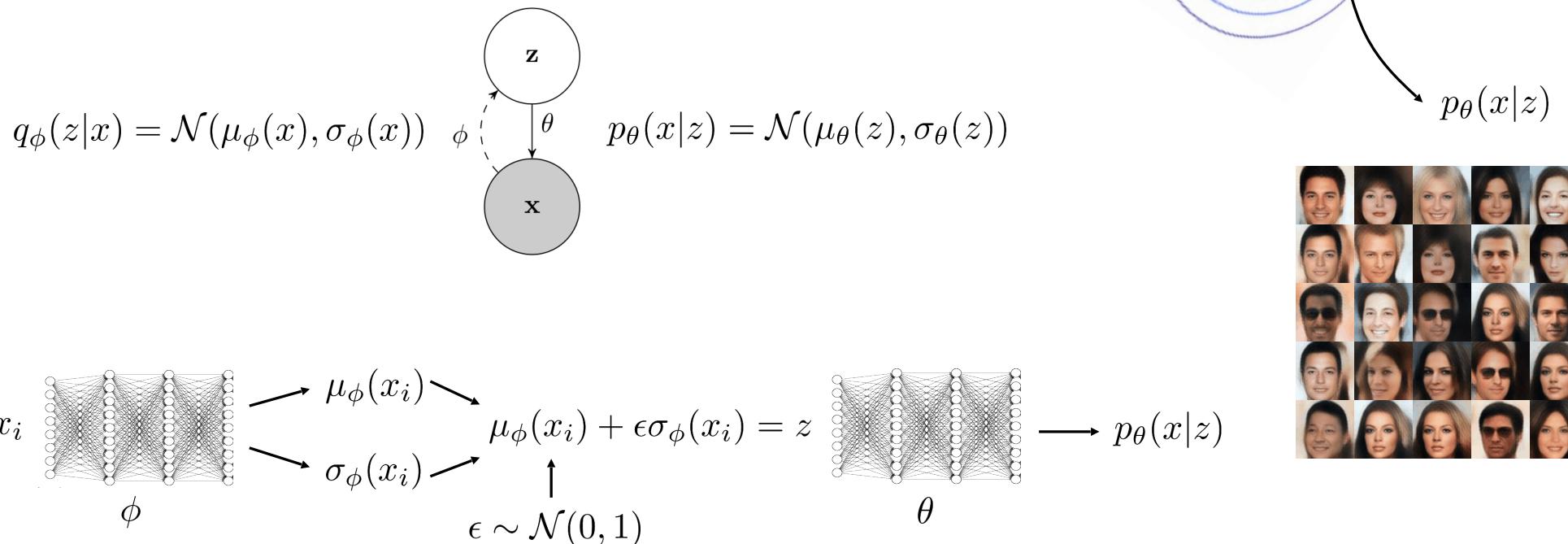
Another way to look at everything...

$$\begin{aligned}
\mathcal{L}_i &= E_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_\phi(z|x_i)) \\
&= E_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i|z)] + \underbrace{E_{z \sim q_\phi(z|x_i)} [\log p(z)] + \mathcal{H}(q_\phi(z|x_i))}_{-D_{\text{KL}}(q_\phi(z|x_i)\|p(z))} \\
&\quad \leftarrow \text{this has a convenient analytical form for Gaussians} \\
&= E_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i|z)] - D_{\text{KL}}(q_\phi(z|x_i)\|p(z)) \\
&= E_{\epsilon \sim \mathcal{N}(0,1)} [\log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i))] - D_{\text{KL}}(q_\phi(z|x_i)\|p(z)) \\
&\approx \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{\text{KL}}(q_\phi(z|x_i)\|p(z))
\end{aligned}$$



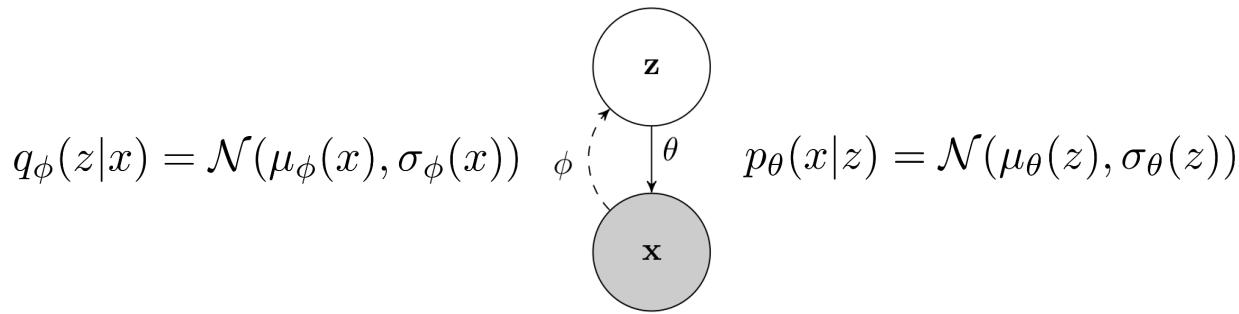
Example Models

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



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

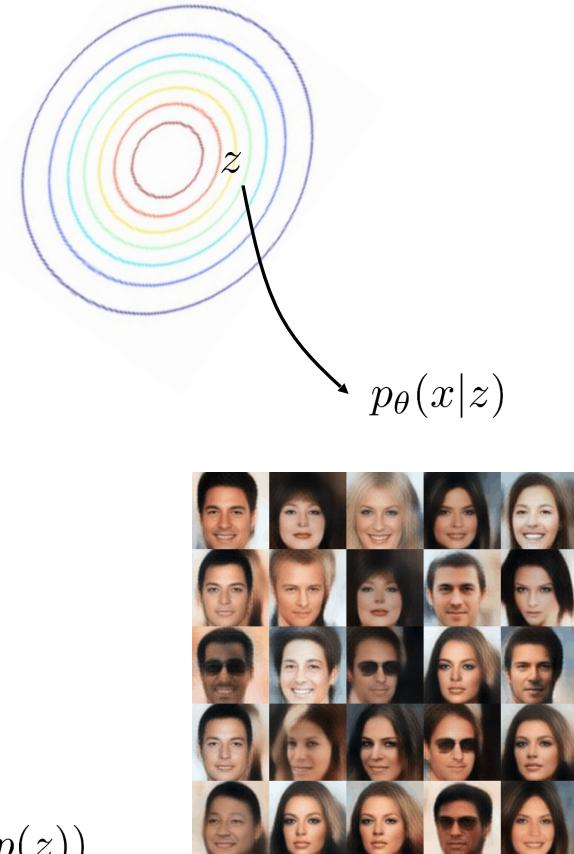
why does this work?

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



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



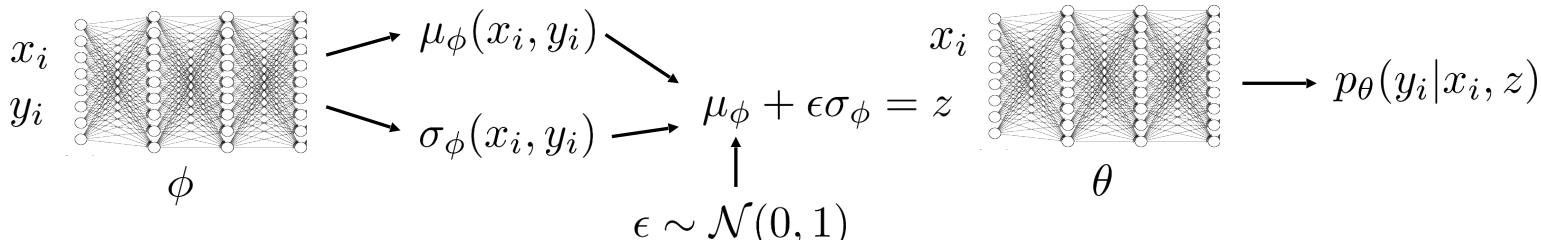
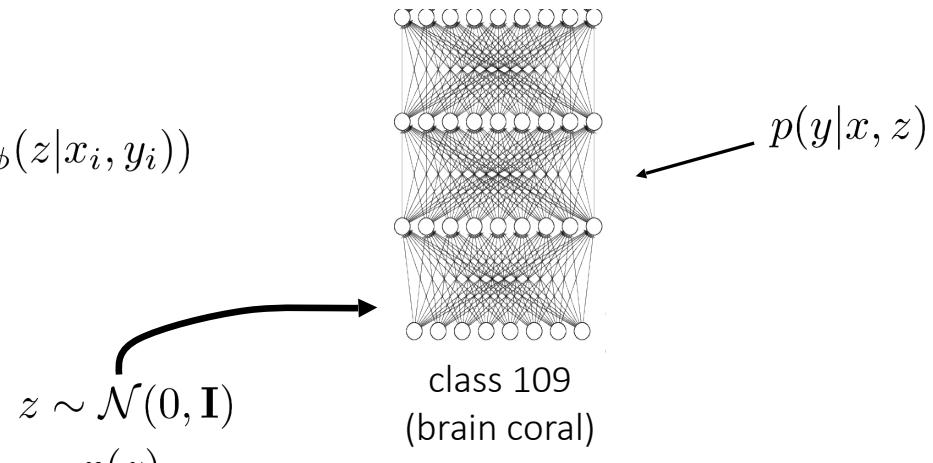
just like before, only now generating y_i
and *everything* is conditioned on x_i

at test time:

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

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

can *optionally* depend on x



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 ~~Wednesday~~ Friday.

Next time: Bayesian meta-learning