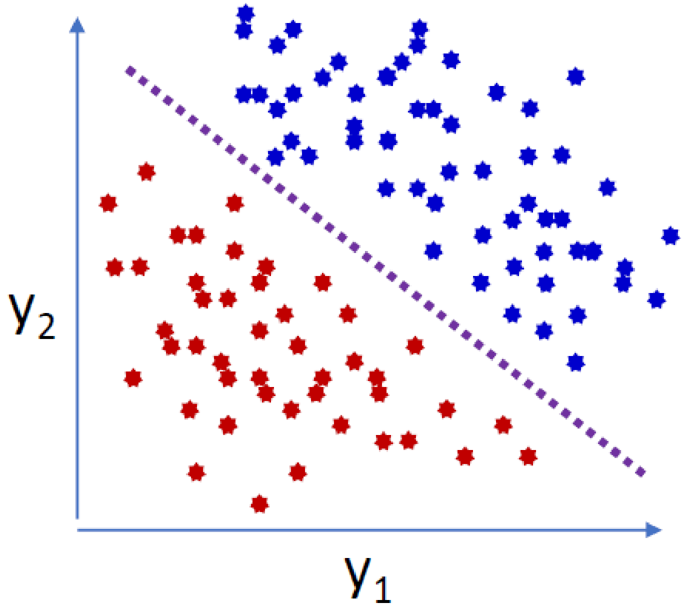
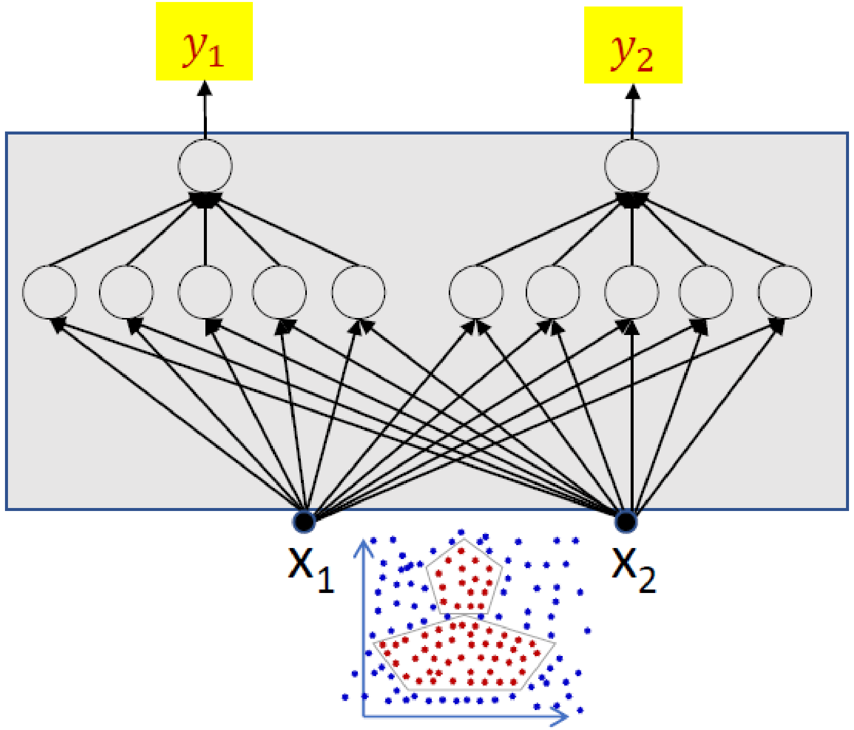
Variational Auto Encoders

Recap

- Deep Neural Models consist of 2 primary components
  - A feature extraction network – Where important aspects of the data are amplified and projected into a linearly separable form (or close to one)
  - A linear classifier to determine how to best label the data
- Neural Networks can be used for representation generation
  - One case of this is called the Autoencoder
  - Project the data into some space which is informative for reconstructing the input

Recap

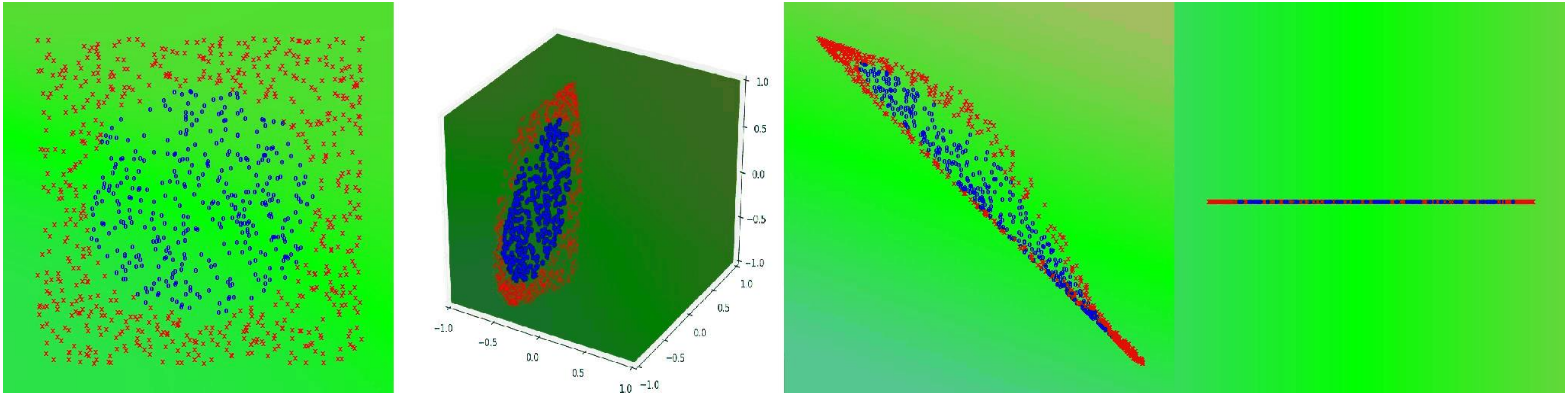




- The network learns features using layers 0 to N-1…
- Final layer used for classification

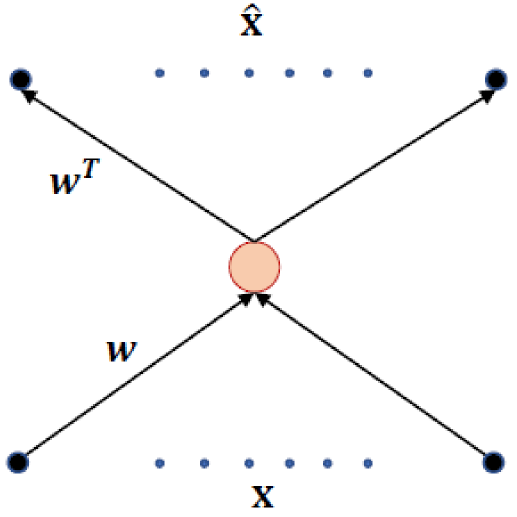
- Common in industry to take a pretrained model, fix all weights except for the last layer and retrain with a new objective.

Recap: How the layers act



Autoencoders and a bit of math

- You can think of autoencoders as performing a nonlinear PCA on the your data
  - How can we think about this?



Training: Learning W by minimizing L2 divergence

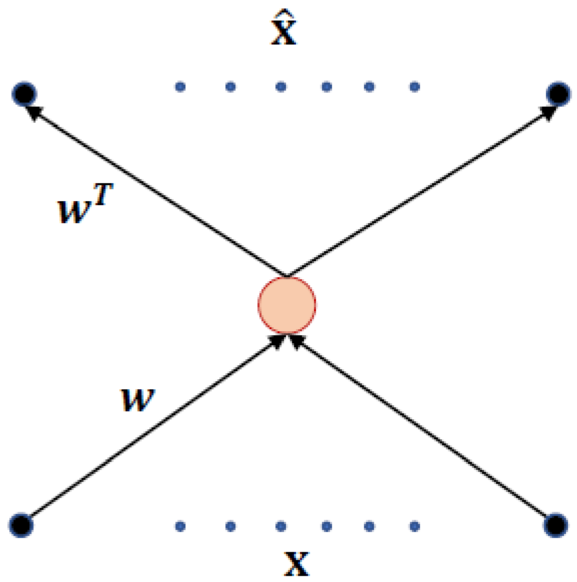
$$\hat{\mathrm{x}} = w^T w\mathrm{x}$$

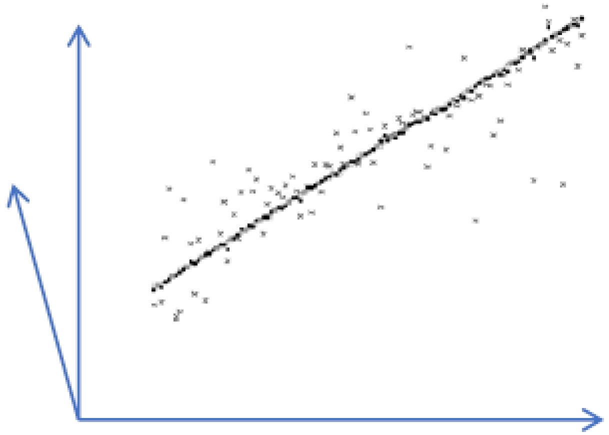
$$div(\hat{\mathrm{x}}, \mathrm{x}) = \|\mathrm{x} - \hat{\mathrm{x}}\|^2 = \|\mathrm{x} - \mathrm{w}^T w\mathrm{x}\|^2$$

$$\widehat{W} = \underset{W}{\operatorname{argmin}} E[div(\hat{\mathrm{x}}, \mathrm{x})]$$

$$\widehat{W} = \underset{W}{\operatorname{argmin}} E[\|\mathrm{x} - \mathrm{w}^T w\mathrm{x}\|^2]$$

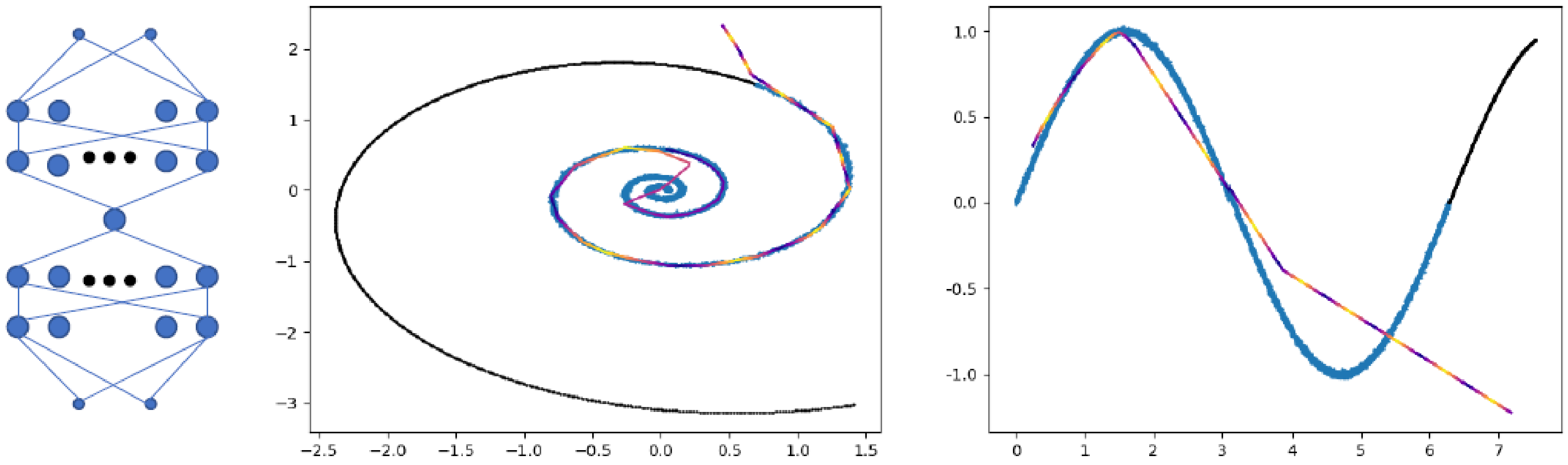
Autoencoders and a bit of math





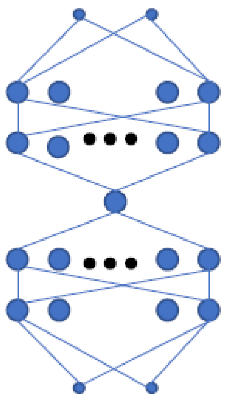
- Autoencoders find the direction of maximum energy
  - Varience if the input is a zero-mean random value
- All input vectors are mapped onto a point on the principal axis

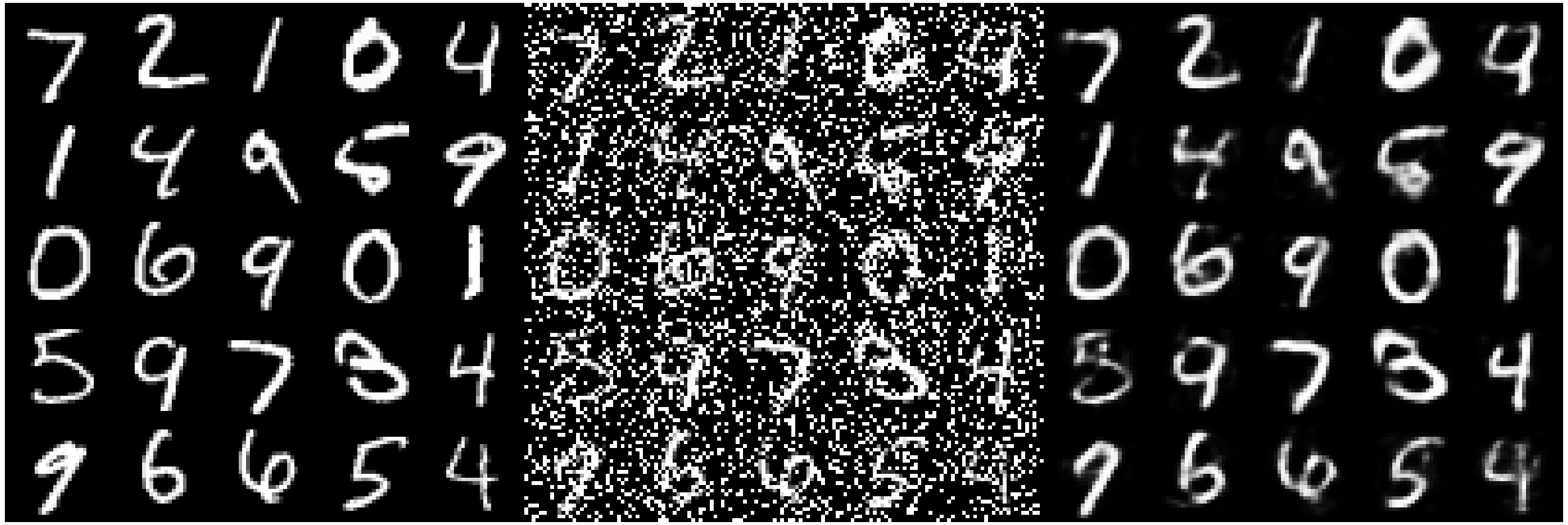
Autoencoders and a bit of math



- Varying the input to the hidden layer only generates data along the learned manifold
    - English translation – the decoder can only decode what looks like the training data
  - This may be poorly learned
  - Any and every input will result in an output along the learned manifold

Autoencoders and a bit of math





http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model

- The decoder represents a source specific generative dictionary
- Passing data to it will produce data typical from the source

Notes for the Rest of This Lecture

- Motivation for Variational Autoencoders (VAEs)
  - Just as auto encoders can be seen as performing non-linear PCA, variational auto encoders can be viewed as performing non-linear factor analysis
- VAEs get their name from variational inference, a technique that can be used for parameter estimation
- During this lecture we will discuss
  - Basics of generative models
  - Factor Analysis
  - Variational Inference and Expectation Maximization
  - VAEs

Recap: Types of Models

- Different types of neural models
  - Discriminative neural models
    - Eg. Classify this image as 1 of 10 different classes
    - CIFAR, MNIST, etc
  - Generative Neural Models
    - Eg. Given this input sequence generate a resulting output sequence
    - Machine translation, etc

  - Both of these types of models have their place and we have explored them both in this class
    - We just haven't drawn this kind of line before
    - This will be a very important distinction when we talk about GANs later

Motivation For Generative Models

- Having a decoder of any kind means that you can arbitrarily generate data points that are similar to those in your training set

- Often easier to train than discriminative models
  - More training data available
    - People don’t go around labeling things, but they often go around associating things
      - Eg. Videos + Comments, images and captions, translation pairs, etc

Motivation For Generative Models

- Caption -> Image

https://openreview.net/pdf?id=Hyvw0L9el

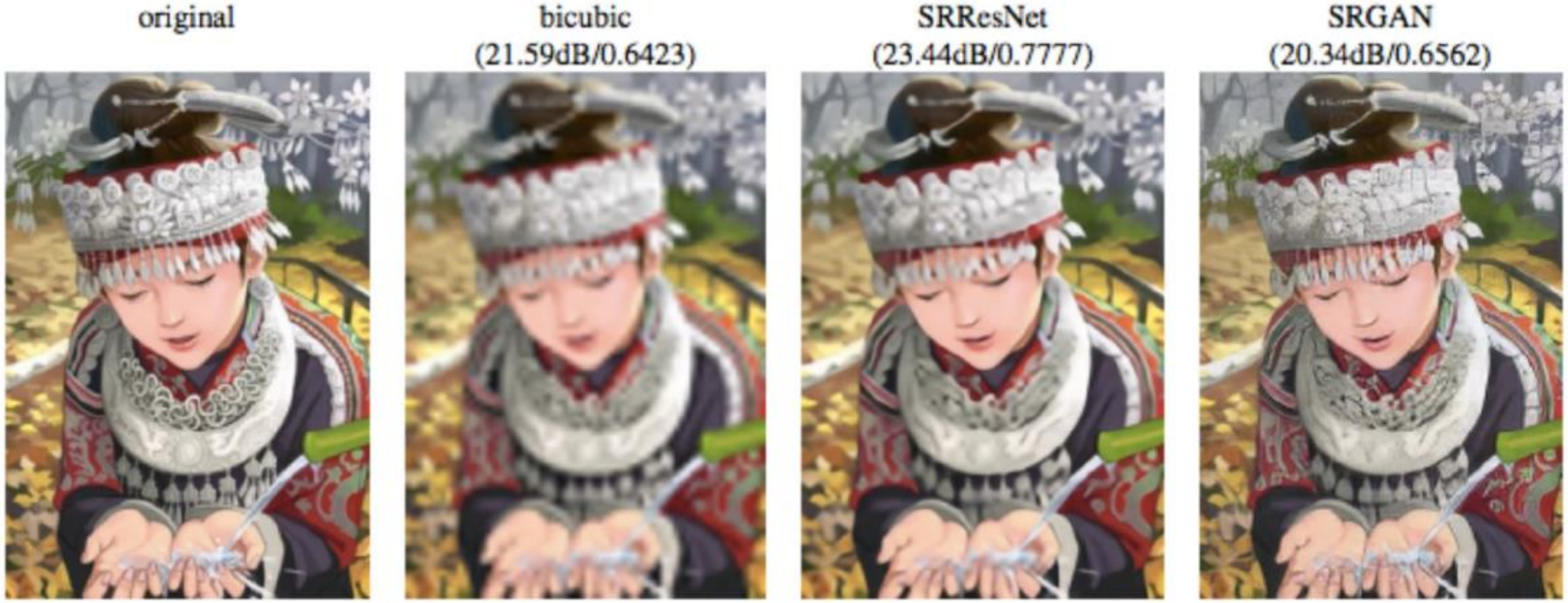
A man in an orange jacket with
sunglasses and a hat skis down a hill

- Outline -> Image

https://arxiv.org/abs/1611.07004

Motivation For Generative Models

Example: Super resolution



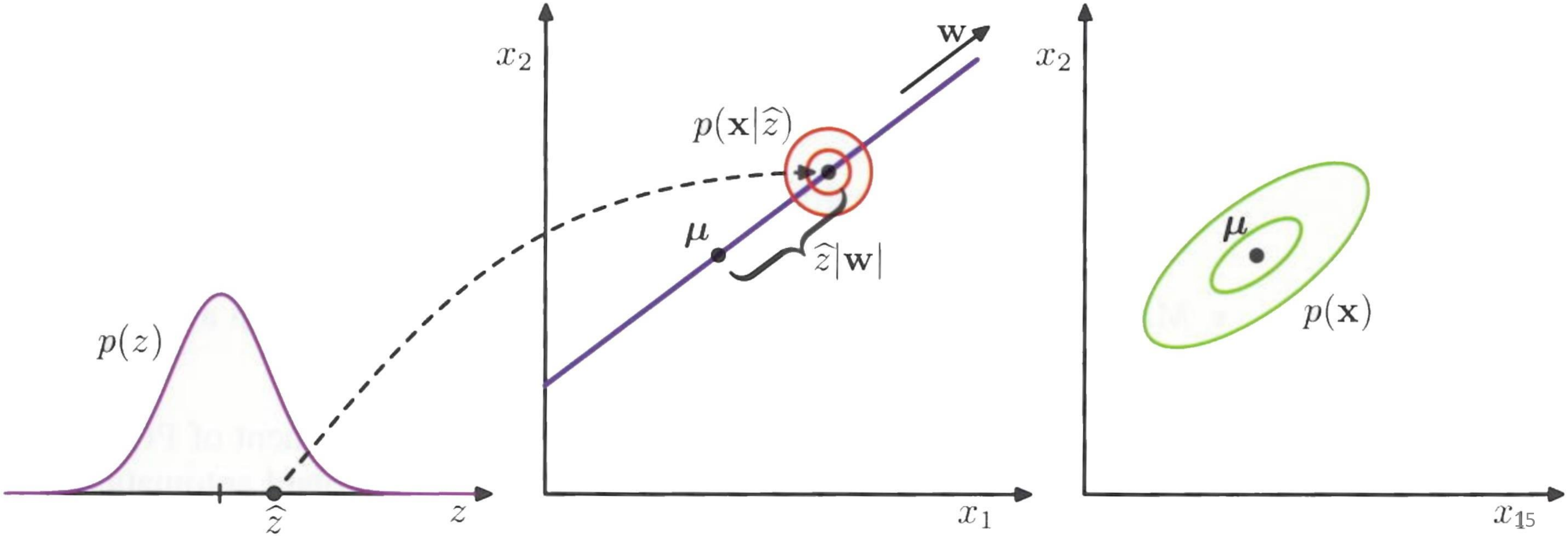
https://arxiv.org/abs/1609.04802

Motivation For Generative Models

- These models give us insight into our data
  - Is there a low level manifold underlying the data or is every variable independent?
  - What how complex is our data, is it easily predictable or is it not

Factor Analysis

- Generative model: Assumes that data are generated from real valued latent variables



Bishop – Pattern Recognition and Machine Learning

Factor Analysis model

Factor analysis assumes a generative model

- where the *ith* observation, $\boldsymbol{x_i} \in \mathbb{R}^D$ is conditioned on a vector of real valued latent variables $\boldsymbol{z_i} \in \mathbb{R}^L$.

Here we assume the prior distribution is Gaussian:

$$p(\boldsymbol{z_i}) = \mathcal{N}(\boldsymbol{z_i} | \boldsymbol{\mu_0}, \boldsymbol{\Sigma_0})$$

We also will use a Gaussian for the data likelihood:

$$p(\boldsymbol{x_i} | \boldsymbol{z_i}, \boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}) = \mathcal{N}(\boldsymbol{W} \boldsymbol{z_i} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

Where $\boldsymbol{W} \in \mathbb{R}^{D \times L}$, $\boldsymbol{\Psi} \in \mathbb{R}^{D \times D}$, $\boldsymbol{\Psi}$ is diagonal

Marginal distribution of observed $\boldsymbol{x_i}$

$$p(\boldsymbol{x_i}|\boldsymbol{W},\boldsymbol{\mu},\boldsymbol{\Psi}) = \int \mathcal{N}(\boldsymbol{W}\boldsymbol{z_i}+\boldsymbol{\mu},\boldsymbol{\Psi})\,\mathcal{N}(\boldsymbol{z_i}|\boldsymbol{\mu_0},\boldsymbol{\Sigma_0})\mathbf{d}\boldsymbol{z_i}$$

$$= \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{W}\boldsymbol{\mu_0}+\boldsymbol{\mu},\boldsymbol{\Psi}+\boldsymbol{W}\,\boldsymbol{\Sigma_0}\boldsymbol{W}^T)$$

Note that we can rewrite this as:

$$p\left(\boldsymbol{x_i}\middle|\widehat{\boldsymbol{W}},\widehat{\boldsymbol{\mu}},\boldsymbol{\Psi}\right) = \mathcal{N}\left(\boldsymbol{x_i}\middle|\widehat{\boldsymbol{\mu}},\boldsymbol{\Psi}+\widehat{\boldsymbol{W}}\widehat{\boldsymbol{W}}^T\right)$$

Where $\widehat{\boldsymbol{\mu}} = \boldsymbol{W}\boldsymbol{\mu_0}+\boldsymbol{\mu}$ and $\widehat{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{\Sigma_0}^{-\frac{1}{2}}$.

Thus without loss of generality (since $\boldsymbol{\mu_0},\boldsymbol{\Sigma_0}$ are absorbed into learnable parameters) we let:

$$p(\boldsymbol{z_i}) = \mathcal{N}(\boldsymbol{z_i}|\boldsymbol{0},\boldsymbol{I})$$

And find:

$$p(\boldsymbol{x_i}|\boldsymbol{W},\boldsymbol{\mu},\boldsymbol{\Psi}) = \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu},\boldsymbol{\Psi}+\boldsymbol{W}\boldsymbol{W}^T)$$

Marginal distribution interpretation

- We can see from $p(\boldsymbol{x_i}|\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}) = \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{W}\boldsymbol{W}^T)$ that the covariance matrix of the data distribution is broken into 2 terms
- A diagonal part $\boldsymbol{\Psi}$: **variance not shared between variables**
- A low rank matrix $\boldsymbol{W}\boldsymbol{W}^T$: **shared variance due to latent factors**

Special Case: Probabilistic PCA (PPCA)

- Probabilistic PCA is a special case of Factor Analysis
- We further restrict $\mathbf{\Psi} = \sigma^2 \boldsymbol{I}$ (assume isotropic independent variance)
- Possible to show that when the data are centered ($\boldsymbol{\mu} = \mathbf{0}$), the limiting case where $\sigma \rightarrow 0$ gives back the same solution for $\boldsymbol{W}$ as PCA
- Factor analysis is a generalization of PCA that models non-shared variance (can think of this as noise in some situations, or individual variation in others)

Inference in FA

- To find the parameters of the FA model, we use the Expectation Maximization (EM) algorithm
- EM is very similar to variational inference
- We'll derive EM by first finding a lower bound on the log-likelihood we want to maximize, and then maximizing this lower bound

Evidence Lower Bound Decomposition

- For any distributions $q(z), p(z)$ we have:

$$\mathrm{KL}\big(q(z) \,||\, p(z)\big) \triangleq \int q(z) \log \frac{q(z)}{p(z)} \mathbf{d}z$$

- Consider the KL divergence of an **arbitrary** weighting distribution $q(z)$ from a conditional distribution $p(z|x,\theta)$:

$$\mathrm{KL}\big(q(z) \,||\, p(z|x,\theta)\big) \triangleq \int q(z) \log \frac{q(z)}{p(z|x,\theta)} \mathbf{d}z$$

$$= \int q(z)[\log q(z) - \log p(z|x,\theta)] \, \mathbf{d}z$$

Applying Bayes

$$\log p(z|x,\theta) = \log\left[\frac{p(x|z,\theta)p(z|\theta)}{p(x|\theta)}\right]$$

$$= \log p(x|z,\theta) + \log p(z|\theta) - \log p(x|\theta)$$

Then:

$$\mathrm{KL}\big(q(z)\,||\,p(z|x,\theta)\big) = \int q(z)[\log q(z) - \log p(z|x,\theta)]\,\mathbf{d}z$$

$$= \int q(z)[\log q(z) - \log p(x|z,\theta) - \log p(z|\theta) + \log p(x|\theta)]\mathbf{d}z$$

Rewriting the divergence

- Since the last term does not depend on z, and we know $\int q(z)\mathrm{d}z = 1$, we can pull it out of the integration:

$$\int q(z)[\log q(z) - \log p(x|z,\theta) - \log p(z|\theta) + \log p(x|\theta)]\mathbf{d}z$$

$$= \int q(z)[\log q(z) - \log p(x|z,\theta) - \log p(z|\theta)]\mathbf{d}z + \log p(x|\theta)$$

$$= \int q(z) \log \left[\frac{q(z)}{p(x|z,\theta)p(z,\theta)}\right]\mathbf{d}z + \log p(x|\theta)$$

$$= \int q(z) \log \left[\frac{q(z)}{p(x,z\,|\theta)}\right]\mathbf{d}z + \log p(x|\theta)$$

Then we have:

$$\mathrm{KL}\big(q(z)\,||\,p(z|x,\theta)\big) = \mathrm{KL}\big(q(z)\,||\,p(x,z\,|\theta)\big) + \log p(x|\theta)$$

Evidence Lower Bound (ELBO)

- From basic probability we have:

$$\mathrm{KL}\big(q(z)\,||\,p(z|x,\theta)\big) = \mathrm{KL}\big(q(z)\,||\,p(x,z\,|\theta)\big) + \log p(x|\theta)$$

- We can rearrange the terms to get the following decomposition:

$$\log p(x|\theta) = \mathrm{KL}\big(q(z)\,||\,p(z|x,\theta)\big) - \mathrm{KL}\big(q(z)\,||\,p(x,z\,|\theta)\big)$$

- We define the *evidence lower bound* (ELBO) as:

$$\mathcal{L}(q,\theta) \triangleq -\mathrm{KL}\big(q(z)\,||\,p(x,z\,|\theta)\big)$$

Then:

$$\log p(x|\theta) = \mathrm{KL}\big(q(z)||p(z|x,\theta)\big) + \mathcal{L}(q,\theta)$$

Why the name Evidence Lower Bound (ELBO)

- Rearranging the decomposition

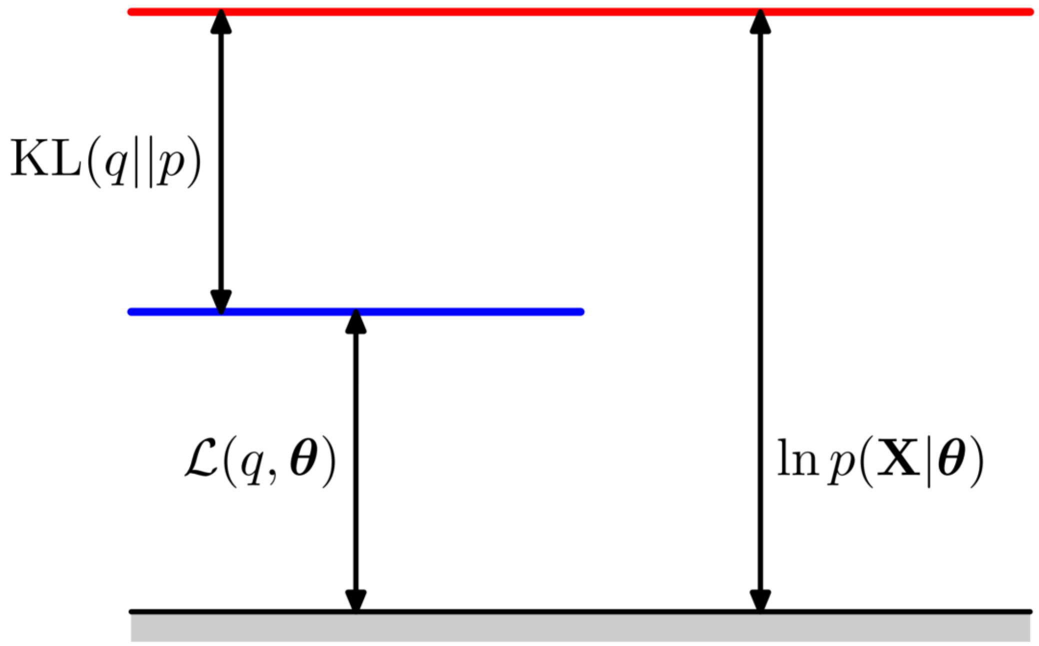
$$\log p(x|\theta) = \mathrm{KL}\big(q(z)||p(z|x,\theta)\big) + \mathcal{L}(q,\theta)$$

- we have

$$\mathcal{L}(q,\theta) = \log p(x|\theta) - \mathrm{KL}\big(q(z)\ ||\ p(z|x,\theta)\big)$$

- Since $\mathrm{KL}\big(q(z)||p(z|x,\theta)\big) \geq 0$, $\mathcal{L}(q,\theta)$ is a lower bound on the log-likelihood we want to maximize
- $p(x|\theta)$ is sometimes called the evidence
- When is this **bound tight**? When $q(z) = p(z|x,\theta)$
- The ELBO is also sometimes called the variational bound

Visualizing ELBO decomposition



Bishop – Pattern Recognition and Machine Learning

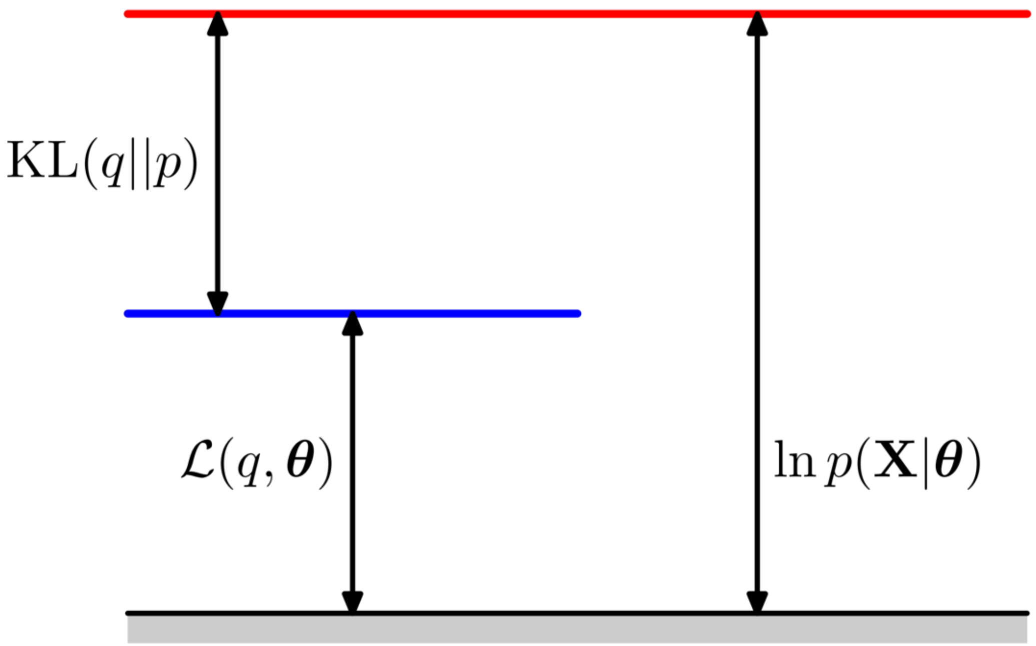
- Note: all we have done so far is decompose the log probability of the data, we still have exact equality
- This holds for any distribution q

Expectation Maximization

- Expectation Maximization alternately optimizes the ELBO, $\mathcal{L}(q, \theta)$, with respect to q (the E step) and θ (the M step)

- Initialize $\theta^{(0)}$
- At each iteration $t = 1, \ldots$
  - **E step:** Hold $\theta^{(t-1)}$ fixed, find $q^{(t)}$ which maximizes $\mathcal{L}\left(q, \theta^{(t-1)}\right)$
  - **M step:** Hold $q^{(t)}$ fixed, find $\theta^{(t)}$ which maximizes $\mathcal{L}\left(q^{(t)}, \theta\right)$

The E step

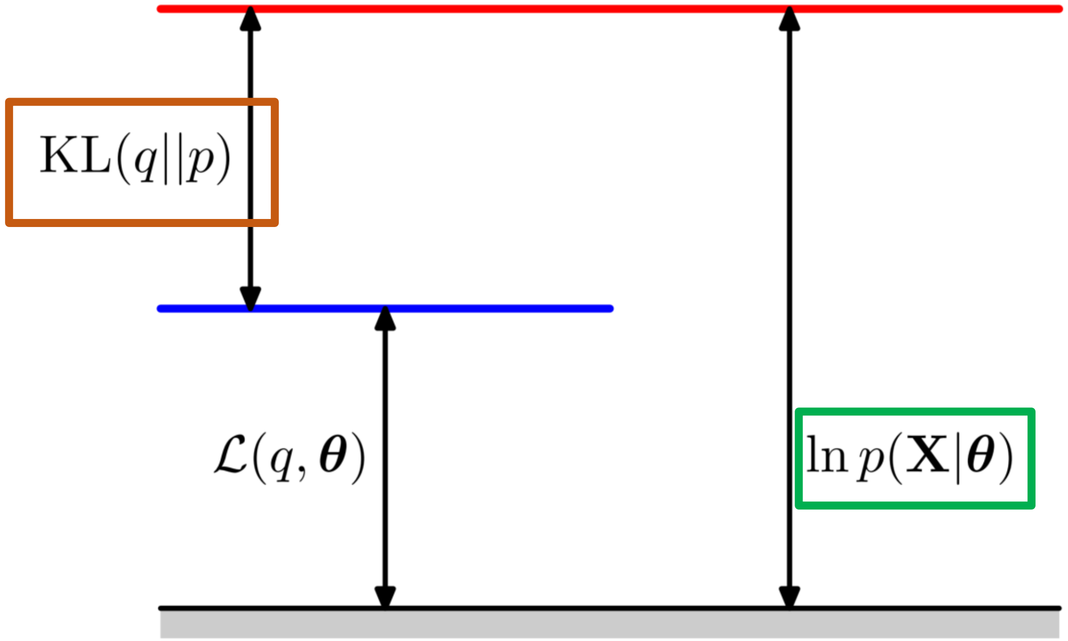


Bishop – Pattern Recognition and Machine Learning

- Suppose we are at iteration t of our algorithm. How do we maximize $\mathcal{L}\left(q, \theta^{(t-1)}\right)$ with respect to q? We know that:

$$\operatorname{argmax}_q \mathcal{L}\left(q, \theta^{(t-1)}\right) = \operatorname{argmax}_q \log p\left(x|\theta^{(t-1)}\right) - \mathrm{KL}\left(q(z) \,||\, p\left(z|x, \theta^{(t-1)}\right)\right)$$

The E step



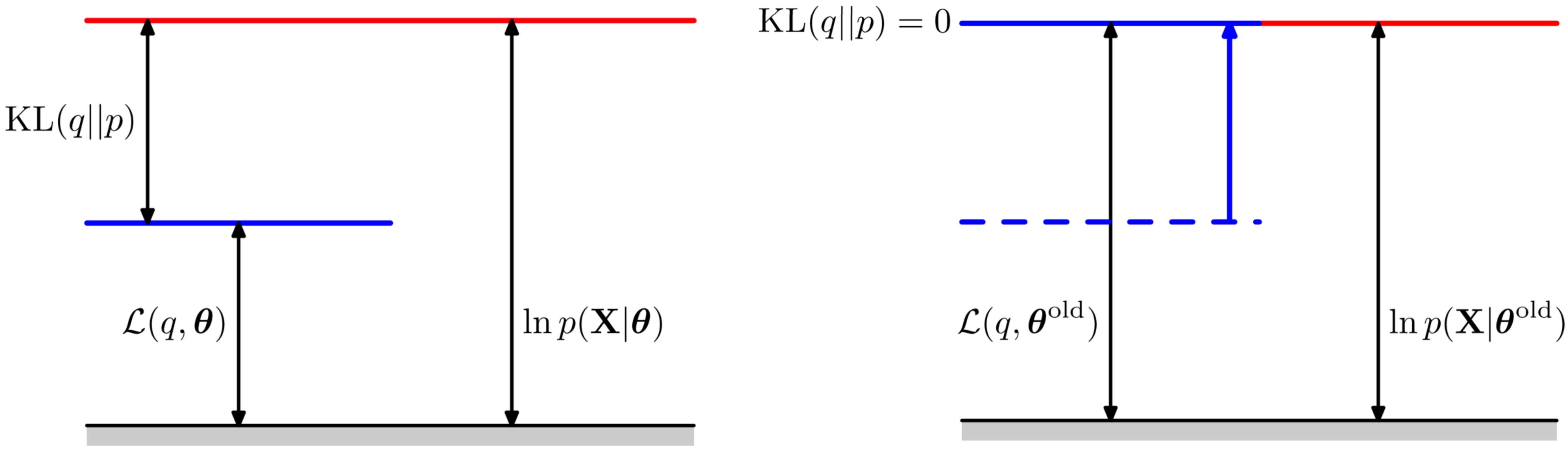
Bishop – Pattern Recognition and Machine Learning

- The first term does not involve q, and we know the KL divergence must be non-negative
- The best we can do is to make the KL divergence 0
- Thus the solution is to set $\boldsymbol{q}^{(t)}(\boldsymbol{z}) \leftarrow \boldsymbol{p}\left(\boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{(t-1)}\right)$

- Suppose we are at iteration t of our algorithm. How do we maximize $\mathcal{L}\left(q, \theta^{(t-1)}\right)$ with respect to q? We know that:

$$\operatorname{argmax}_q \mathcal{L}\left(q, \theta^{(t-1)}\right) = \operatorname{argmax}_q \log p\left(x|\theta^{(t-1)}\right) - \mathrm{KL}\left(q(z) \,||\, p\left(z|x, \theta^{(t-1)}\right)\right)$$

The E step



Bishop – Pattern Recognition and Machine Learning

- Suppose we are at iteration t of our algorithm. How do we maximize $\mathcal{L}\left(q, \theta^{(t-1)}\right)$ with respect to q? $\boldsymbol{q}^{(t)}(\boldsymbol{z}) \leftarrow \boldsymbol{p}\left(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t-1)}\right)$

The M step

- Fixing $q^{(t)}(z)$ we now solve:

$$\operatorname{argmax}_\theta \mathcal{L}(q^{(t)}, \theta) = \operatorname{argmax}_\theta -\mathrm{KL}\left(q^{(t)}(z) \,||\, p(x, z|\theta)\right)$$

$$= \operatorname{argmax}_\theta - \int q^{(t)}(z) \log\left[\frac{q^{(t)}(z)}{p(x, z|\theta)}\right] \mathbf{d}z$$

$$= \operatorname{argmax}_\theta \int q^{(t)}(z) [\log p(x, z\,|\theta) - \log q^{(t)}(z)] \mathbf{d}z$$

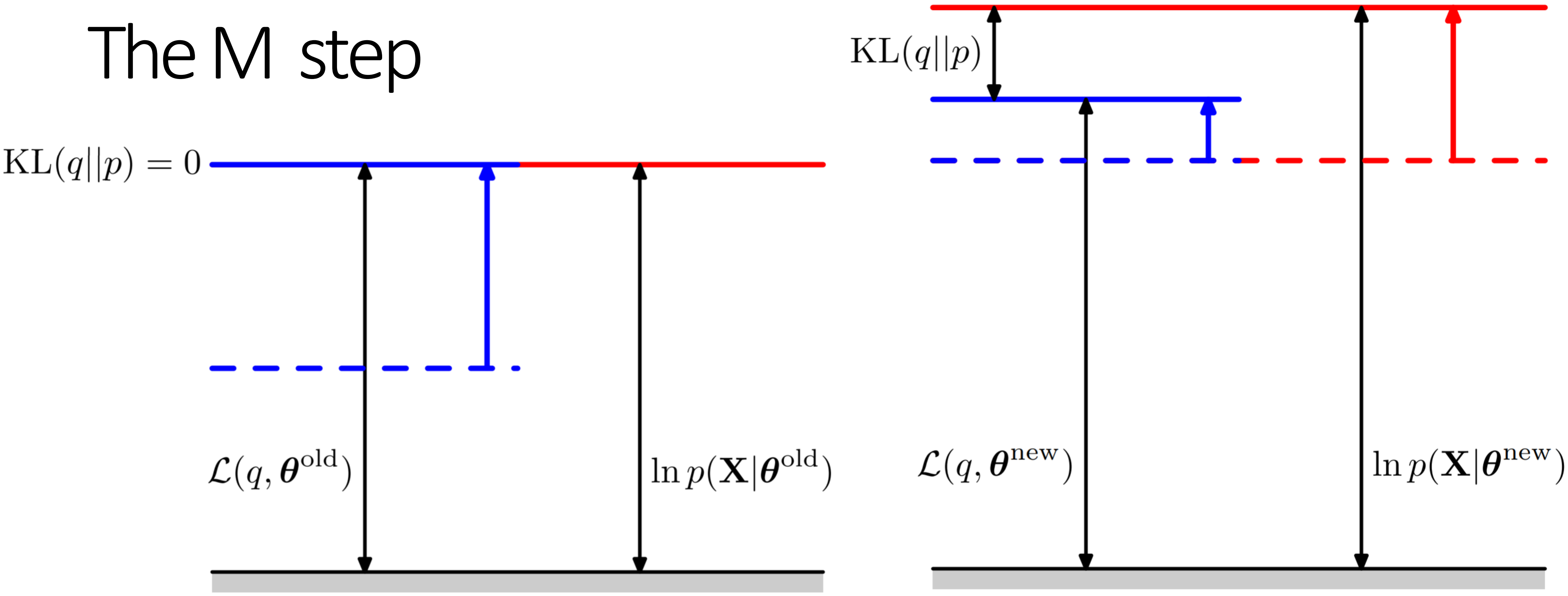
$$= \operatorname{argmax}_\theta \int q^{(t)}(z) \log p(x, z\,|\theta) \boxed{- q^{(t)}(z) \log q^{(t)}(z)\, \mathbf{d}z}$$

$$= \operatorname{argmax}_\theta \int q^{(t)}(z) \log p(x, z\,|\theta)\, \mathbf{d}z$$

$$= \operatorname{argmax}_\theta \mathbb{E}_{q^{(t)}(z)}[\log p(x, z\,|\theta)]$$

Constant w.r.t. θ

The M step



Bishop – Pattern Recognition and Machine Learning

- After applying the Estep, we increase the likelihood of the data by finding better parameters according to: $\theta^{(t)} \leftarrow \mathbf{argmax}_{\boldsymbol{\theta}} \, \mathbb{E}_{q^{(t)}(z)}[\, \log p(x, z \,|\boldsymbol{\theta})]$

The EM Algorithm

- Initialize $\theta^{(0)}$
- At each iteration $t = 1, \ldots$
  - **E step:** Update $q^{(t)}(z) \leftarrow p\left(z \middle| x, \theta^{(t-1)}\right)$
  - **M step:** Update $\theta^{(t)} \leftarrow \operatorname{argmax}_\theta \mathbb{E}_{q^{(t)}(z)}[\log p(x, z \mid \theta)]$

Why Does EM Work?

- EM does coordinate ascent on the ELBO, $\mathcal{L}(q, \theta)$
- Each iteration increases the log-likelihood until $q^{(t)}$ converges (i.e. we reach a local maximum)!
- Simple to prove

Notice after the E step:

$$\mathcal{L}\left(q^{(t)}, \theta^{(t-1)}\right)$$
$$= \log p(x|\theta^{(t-1)}) - \mathrm{KL}\left(p\left(z|x, \theta^{(t-1)}\right) \,||\, p\left(z|x, \theta^{(t-1)}\right)\right)$$
$$= \log p(x|\theta^{(t-1)})$$

The ELBO is tight!

By definition of argmax in the M step:

$$\mathcal{L}\left(q^{(t)}, \theta^{(t)}\right) \geq \mathcal{L}\left(q^{(t)}, \theta^{(t-1)}\right)$$

By simple substitution:

$$\mathcal{L}\left(q^{(t)}, \theta^{(t)}\right) \geq \log p\left(x \middle| \theta^{(t-1)}\right)$$

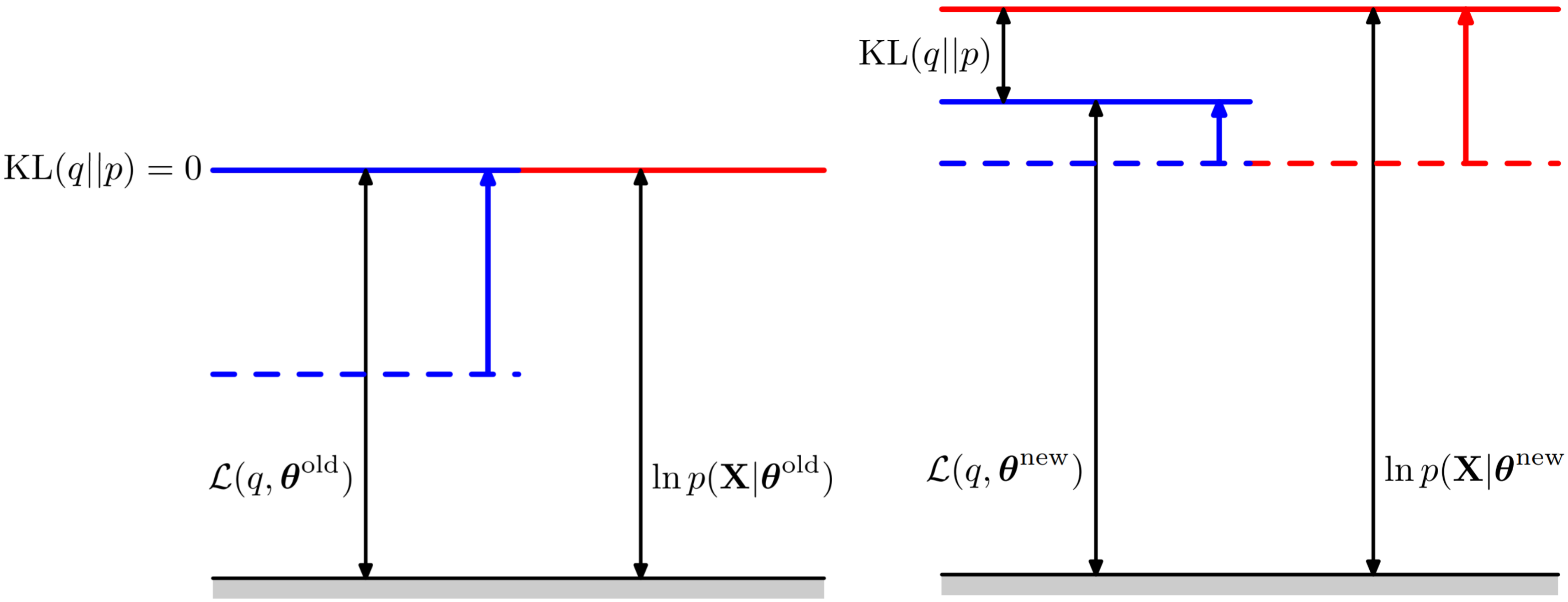
Rewriting the left hand side:

$$\log p(x|\theta^{(t)}) - \mathrm{KL}\left(p\left(z|x, \theta^{(t-1)}\right) \,||\, p\left(z|x, \theta^{(t)}\right)\right)$$
$$\geq \log p\left(x \middle| \theta^{(t-1)}\right)$$

Noting that KL is non-negative:

$$\boldsymbol{\log p\left(x \middle| \theta^{(t)}\right) \geq \log p\left(x \middle| \theta^{(t-1)}\right)}$$

Why does EM work?



Bishop – Pattern Recognition and Machine Learning

- This proof is saying the same thing we saw in pictures. Make the KL0, then improve our parameter estimates to get a better likelihood

From EM to Variational Inference

- In EM we alternately maximize the ELBO with respect to θ and probability distribution (functional) q
- In variational inference, we **drop the distinction between hidden variables and parameters** of a distribution
- I.e. we replace $p(x, z|\theta)$ with $p(x, z)$. Effectively this puts a **probability distribution on the parameters $\boldsymbol{\theta}$**, then absorbs them into z
- Fully Bayesian treatment instead of a point estimate for the parameters

Variational Inference

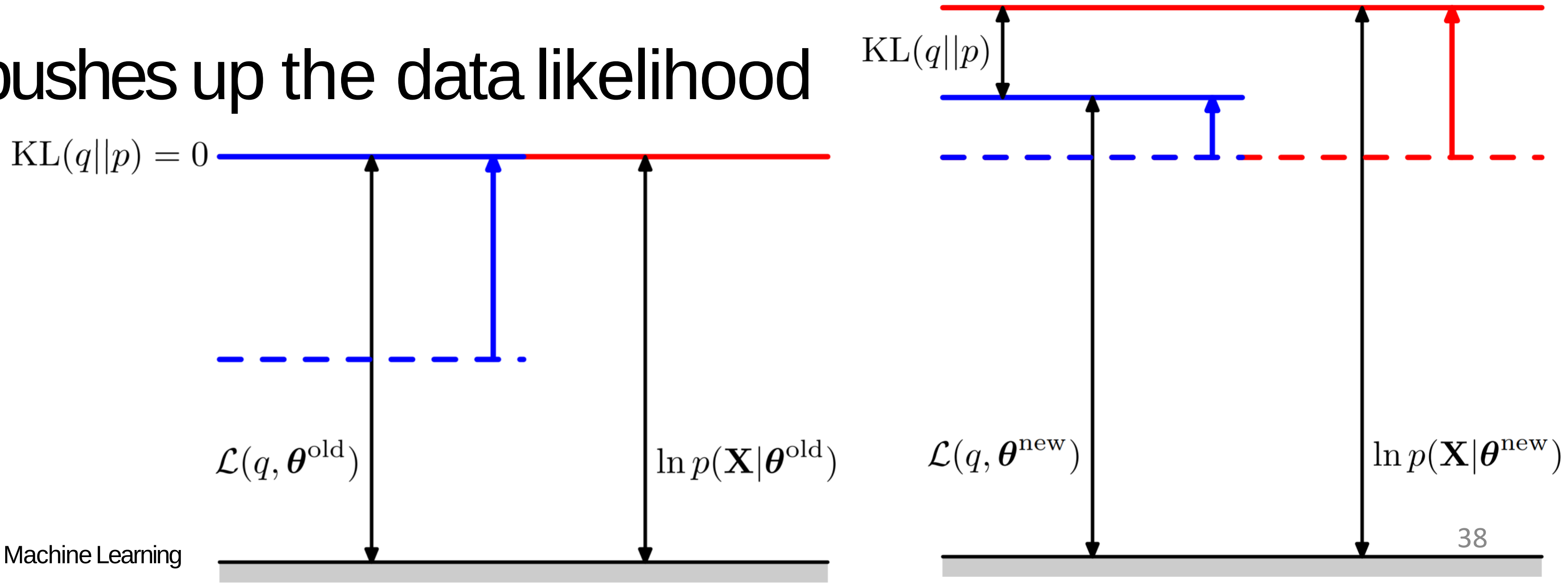
- Now the ELBO is just a function of our weighting distribution $\mathcal{L}(q)$
- We assume a form for q that we can optimize
- For example *mean field theory* assumes q factorizes:

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i)$$

- Then we optimize $\mathcal{L}(q)$ with respect to one of the terms while holding the others constant, and repeat for all terms
- By assuming a form for q we approximate a (typically) intractable true posterior

Why does Variational Inferencework?

- The argument is similar to the argument for EM
- When expectations are computed using the current values for the variables not being updated, we implicitly set the KL divergence between the weighting distributions and the posterior distributions to 0
- The update then pushes up the data likelihood



Bishop – Pattern Recognition and Machine Learning

A different perspective

- Consider the log-likelihood of a marginal distribution of the data x in a generic latent variable model with latent variable z parameterized by θ:

$$\ell(\theta) \triangleq \sum_{i=1}^{N} \log p(x_i|\theta) = \sum_{i=1}^{N} \log \int p(x_i, z_i|\theta) \mathbf{d} z_i$$

- Estimating θ is difficult because we have a log outside of the integral, so it does not act directly on the probability distribution (frequently in the exponential family)
- If we observed z_i, then our log-likelihood would be:

$$\ell_c(\theta) \triangleq \sum_{i=1}^{N} \log p(x_i, z_i|\theta)$$

This is called the *complete log-likelihood*

Expected Complete Log-Likelihood

- We can take the expectation of this likelihood over a distribution of the latent variable $q(z)$:

$$\mathbb{E}_{q(z)}[\ell_c(\theta)] = \sum_{i=1}^{N} \int q(z_i) \log p(x_i, z_i | \theta) \, \mathrm{d}z_i$$

- This looks similar to marginalizing, but now the log is inside the integral, so it's easier to deal with
- We can treat the latent variables as observed and solve this more easily than directly solving the log-likelihood
- Finding the q that maximizes this is the E step of EM
- Finding the θ that maximizes this is the M step of EM

Back to Factor Analysis

- For simplicity, assume data is centered. We want:

$$\operatorname{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \log p(\boldsymbol{X}|\boldsymbol{W},\boldsymbol{\Psi}) = \operatorname{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \sum_{i=1}^{N} \log p(\boldsymbol{x_i}|\boldsymbol{W},\boldsymbol{\Psi})$$

$$= \operatorname{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \sum_{i=1}^{N} \log \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{0},\boldsymbol{\Psi}+\boldsymbol{W}\boldsymbol{W}^T)$$

- No closed form solution in general (PPCA can be solved in closed form)
- $\boldsymbol{\Psi}$, $\boldsymbol{W}$ get coupled together in the derivative and we can't solve for them analytically

EM for Factor Analysis

$$\text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}}\, \mathbb{E}_{q^{(t)}(\boldsymbol{z})}[\log p(\boldsymbol{X},\boldsymbol{Z}\,|\boldsymbol{W},\boldsymbol{\Psi})] \;= \text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\log p(\boldsymbol{x}_i|\boldsymbol{z}_i,\boldsymbol{W},\boldsymbol{\Psi})] + \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\log p(\boldsymbol{z}_i)]$$

$$= \text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\log p(\boldsymbol{x}_i|\boldsymbol{z}_i,\boldsymbol{W},\boldsymbol{\Psi})]$$

$$= \text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\log \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i,\boldsymbol{\Psi})]$$

$$= \text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} \,\text{const} - \frac{N}{2}\log\det(\boldsymbol{\Psi}) - \sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}\left[\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{z}_i)^T\boldsymbol{\Psi}^{-1}(\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{z}_i)\right]$$

$$= \text{argmax}_{\boldsymbol{W},\boldsymbol{\Psi}} - \frac{N}{2}\log\det(\boldsymbol{\Psi}) - \sum_{i=1}^{N}\left(\frac{1}{2}\boldsymbol{x}_i^T\boldsymbol{\Psi}^{-1}\boldsymbol{x}_i - \boldsymbol{x}_i^T\boldsymbol{\Psi}^{-1}\boldsymbol{W}\mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\boldsymbol{z}_i] + \frac{1}{2}\text{tr}\left(\boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\boldsymbol{W}\mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\boldsymbol{z}_i\boldsymbol{z}_i^T]\right)\right)$$

- **We only need these 2 sufficient statistics to enable the M step.**
- **In practice, sufficient statistics are often what we compute in the E step**

Factor Analysis E step

$$\mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\boldsymbol{z}_i] = \boldsymbol{G}\boldsymbol{W}^{(\boldsymbol{t}-\boldsymbol{1})^T}\boldsymbol{\Psi}^{(t-1)^{-1}}\boldsymbol{x}_i$$

$$\mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T\right] = \boldsymbol{G} + \mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\boldsymbol{z}_i]\mathbb{E}_{q^{(t)}(\boldsymbol{z}_i)}[\boldsymbol{z}_i]^T$$

Where

$$\boldsymbol{G} = \left(\boldsymbol{I} + \boldsymbol{W}^{(t-1)^T}\boldsymbol{\Psi}^{(t-1)^{-1}}\boldsymbol{W}^{(t-1)}\right)^{-1}$$

This is derived via the Bayes rule for Gaussians

Factor Analysis M step

$$\boldsymbol{W}^{(t)} \leftarrow \left[\sum_{i=1}^{N} \boldsymbol{x}_i \, \mathbb{E}_{q^{(t)}(\mathbf{z}_i)}[\mathbf{z}_i]^T\right]\left[\sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\mathbf{z}_i)}\left[\mathbf{z}_i \mathbf{z}_i^T\right]\right]^{-1}$$

$$\boldsymbol{\Psi}^{(t)} \leftarrow \operatorname{diag}\left(\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T - \boldsymbol{W}^{(t)} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{q^{(t)}(\mathbf{z}_i)}[\mathbf{z}_i] \boldsymbol{x}_i^T\right)$$

From EM to Variational Inference

- In EM we alternately maximize the ELBO with respect to θ and probability distribution (functional) q
- In variational inference, we **drop the distinction between hidden variables and parameters** of a distribution
- I.e. we replace $p(x, z|\theta)$ with $p(x, z)$. Effectively this puts a **probability distribution on the parameters $\boldsymbol{\theta}$**, then absorbs them into z
- Fully Bayesian treatment instead of a point estimate for the parameters

Variational Inference

- Now the ELBO is just a function of our weighting distribution $\mathcal{L}(q)$
- We assume a form for q that we can optimize
- For example *mean field theory* assumes q factorizes:

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i)$$

- Then we optimize $\mathcal{L}(q)$ with respect to one of the terms while holding the others constant, and repeat for all terms
- By assuming a form for q we approximate a (typically) intractable true posterior

Mean Field Update Derivation

$$\mathcal{L}(q) = \int q(Z) \log\left[\frac{p(X,Z)}{q(Z)}\right] dZ = \int q(Z) \log p(X,Z) \; - q(Z) \log q(Z) \;\, dZ$$

$$= \int \prod_i q_i(Z_i) \left\{ \log p(X,Z) \; - \sum_k \log q_k(Z_k) \right\} dZ$$

$$= \int q_j(Z_j) \left\{ \int \prod_{i \neq j} q_i(Z_i) \left\{ \log p(X,Z) \; - \sum_k \log q_k(Z_k) \right\} dZ_i \right\} \; dZ_j$$

$$= \int q_j(Z_j) \left\{ \int \log p(X,Z) \prod_{i \neq j} q_i(Z_i) dZ_i - \int \prod_{i \neq j} \sum_k q_i(Z_i) \log q_k(Z_k) \; dZ_i \right\} \; dZ_j$$

$$= \int q_j(Z_j) \left\{ \int \log p(X,Z) \prod_{i \neq j} q_i(Z_i) dZ_i - \log q_j(Z_j) \int \prod_{i \neq j} q_i(Z_i) \;\; dZ_i \right\} \; dZ_j + \; \text{const}$$

$$= \int q_j(Z_j) \left\{ \int \log p(X,Z) \prod_{i \neq j} q_i(Z_i) dZ_i \right\} \; dZ_j - \int q_j\big(Z_j\big) \log q_j\big(Z_j\big) dZ_j + \; \text{const}$$

$$= \int q_j\big(Z_j\big) \mathbb{E}_{i \neq j}[\log p(X,Z)] \; dZ_j - \int q_j(Z_j) \log q_j\big(Z_j\big) dZ_j + \; \text{const}$$

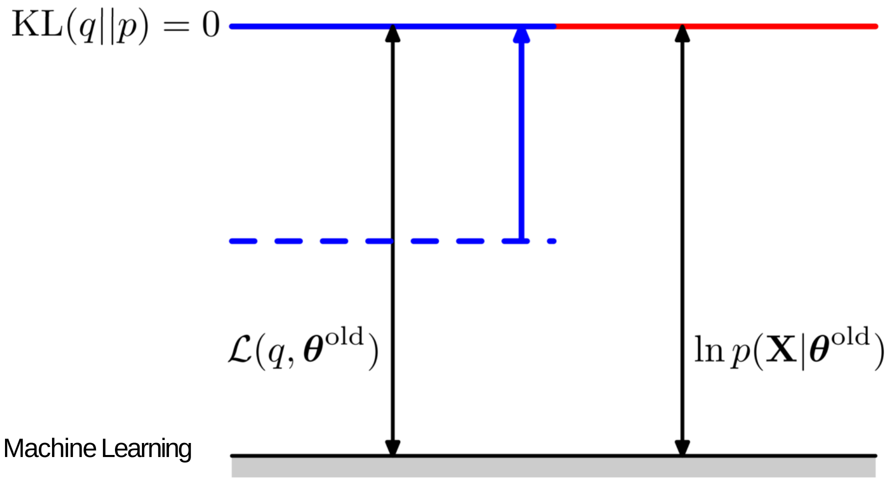
Mean Field Update

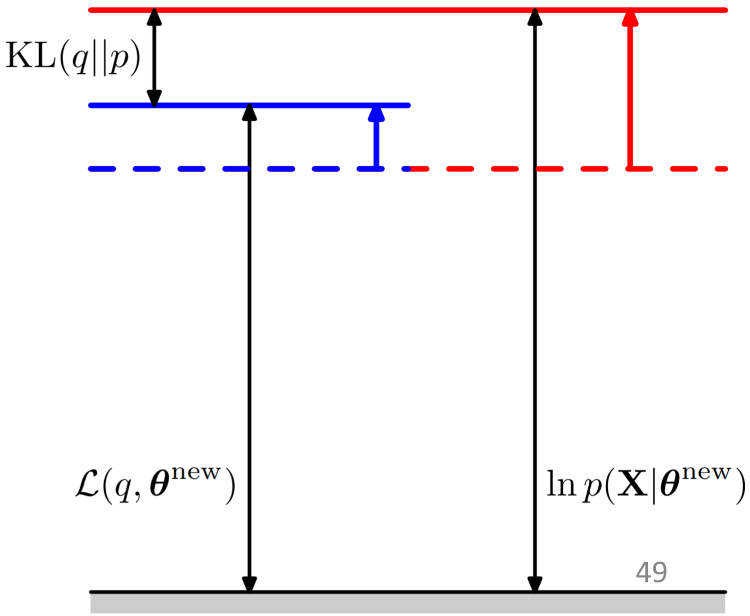
$$q_j(Z_j)^{(t)}$$
$$\leftarrow \operatorname{argmax}_{q_j(Z_j)} \int q_j(Z_j) \mathbb{E}_{i \neq j}[\log p(X, Z)]\, dZ_j$$
$$- \int q_j(Z_j) \log q_j(Z_j) dZ_j$$

- The point of this is not the update equations themselves, but the general idea:
  - **freeze some of the variables, compute expectations over those**
  - **update the rest using these expectations**

Why does Variational Inferencework?

- The argument is similar to the argument for EM
- When expectations are computed using the current values for the variables not being updated, we implicitly set the KL divergence between the weighting distributions and the posterior distributions to 0
- The update then pushes up the data likelihood



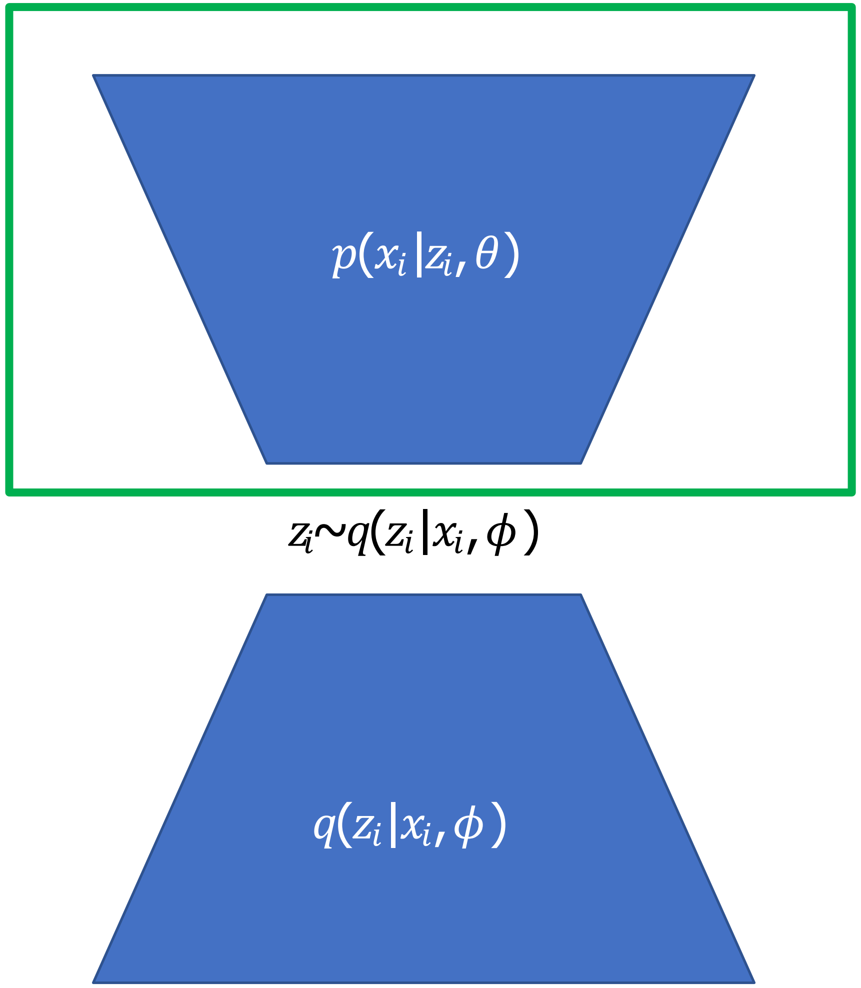


Bishop – Pattern Recognition and Machine Learning

Variational Autoencoder

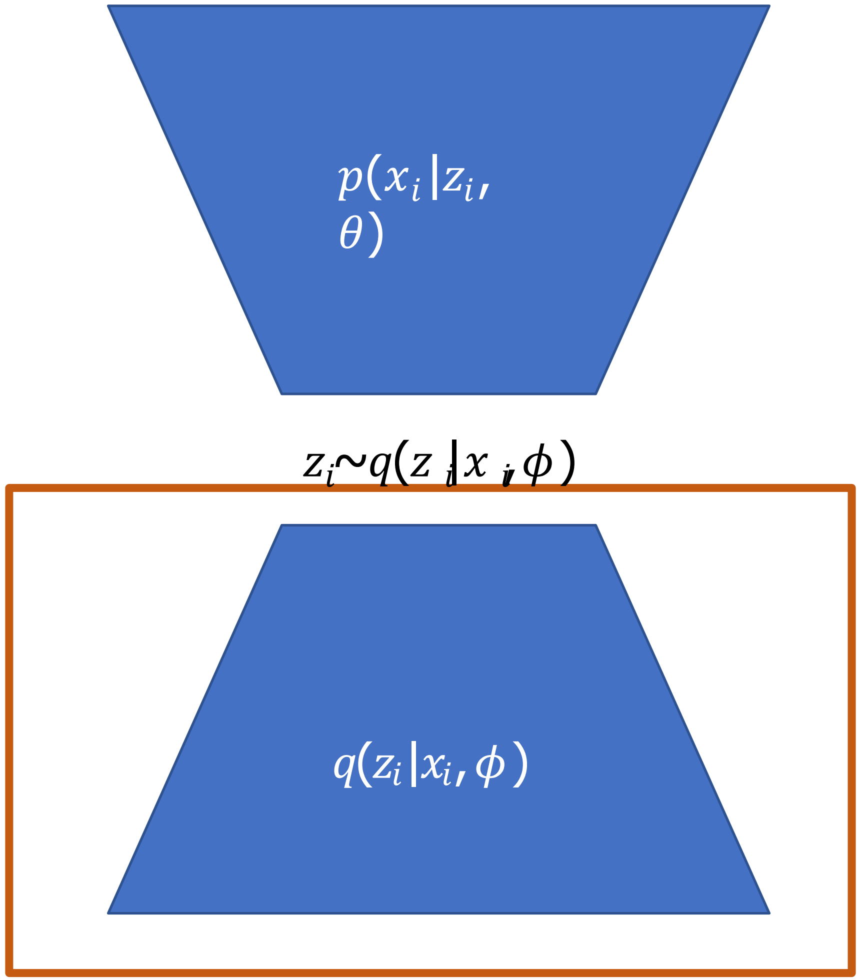
- Kingma & Welling: Auto-Encoding Variational Bayes proposes maximizing the ELBO with a trick to make it differentiable
- Discusses both the variational autoencoder model using parametric distributions and fully Bayesian variational inference, but we will only discuss the variational autoencoder

Problem Setup



- Assume a generative model with a latent variable distributed according to some distribution $p(z_i)$
- The observed variable is distributed according to a conditional distribution $p(x_i|z_i,\theta)$
- Note the similarity to the Factor Analysis (FA) setup so far

Problem Setup

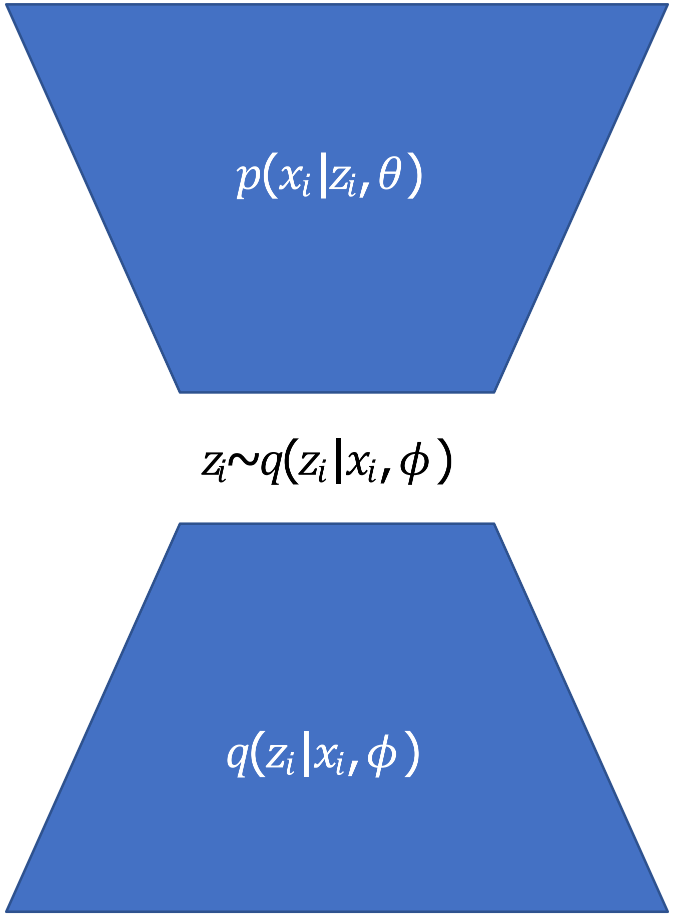


- We also create a weighting distribution $q(z_i|x_i, \phi)$
- This will play the same role as $q(z_i)$ in the EM algorithm, as we will see.
- Note that when we discussed EM, this weighting distribution could be ***arbitrary***: we choose to condition on x_i here. **This is a choice**.

Using a conditional weighting distribution

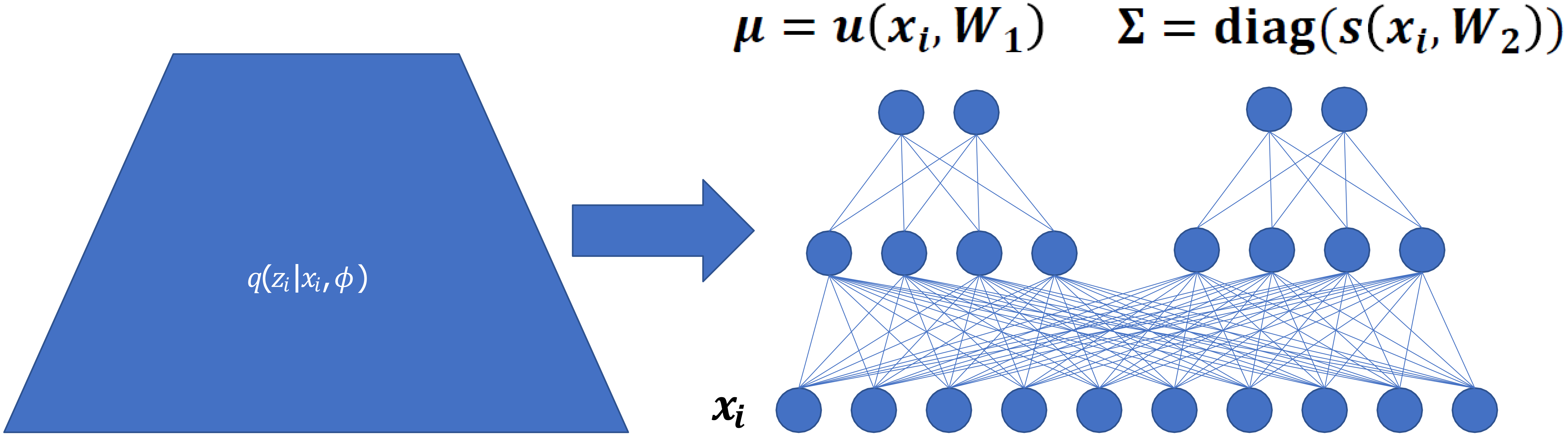
- There are many values of the latent variables that don't matter in practice – by conditioning on the observed variables, we emphasize the latent variable values we actually care about: the ones most likely given the observations
- We would like to be able to encode our data into the latent variable space. This conditional weighting distribution enables that encoding

Problem setup



- Implement $p(x_i|z_i,\theta)$ as a neural network, this can also be seen as a **probabilistic decoder**
- Implement $q(z_i|x_i,\phi)$ as a neural network, we also can see this as a **probabilistic encoder**
- Sample z_i from $q(z_i|x_i,\phi)$ in the middle

Unpacking the encoder

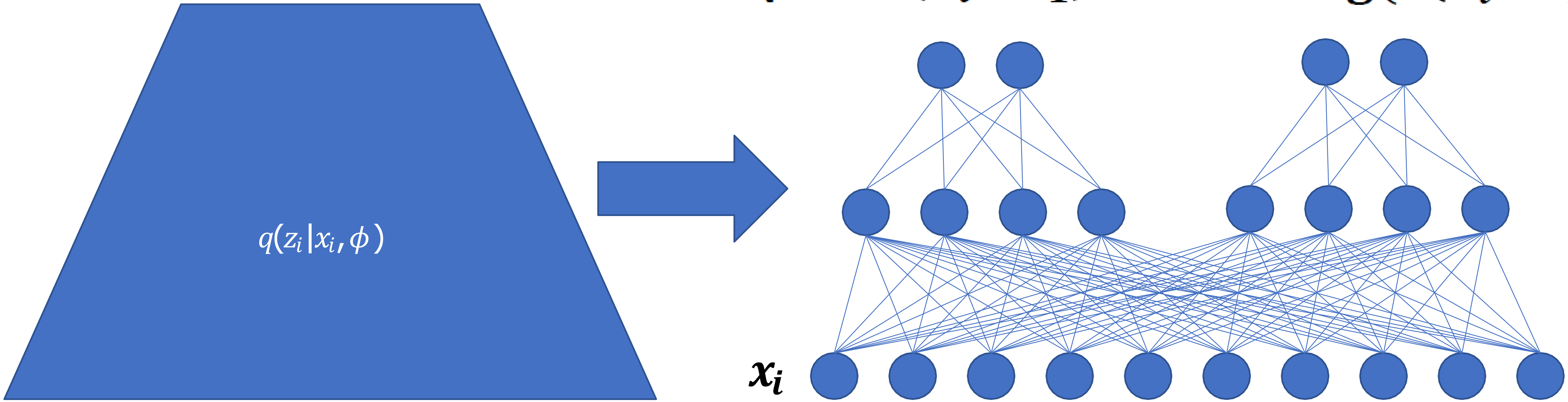


- We choose a family of distributions for our conditional distribution q. For example Gaussian with diagonal covariance:

$$q(z_i|x_i,\phi) = \mathcal{N}(z_i|\mu = u(x_i, W_1), \Sigma = \mathrm{diag}(s(x_i, W_2)))$$

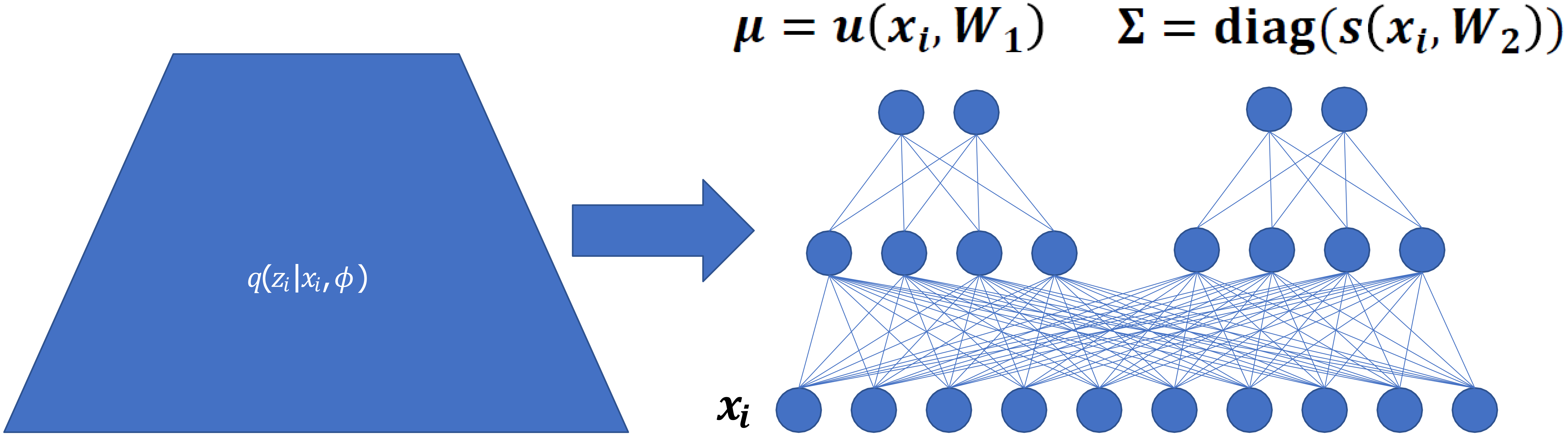
Unpacking the encoder

$$\mu = u(x_i, W_1) \qquad \Sigma = \mathrm{diag}(s(x_i, W_2))$$



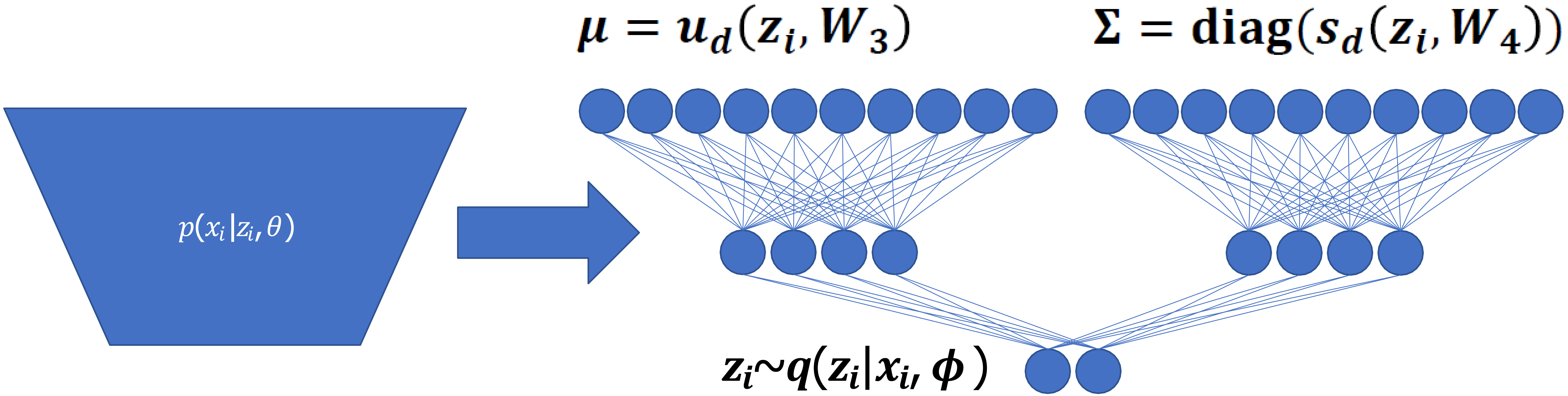
- We create neural networks to predict the parameters of q from our data
- In this case, the outputs of our networks are μ and Σ

Unpacking the encoder



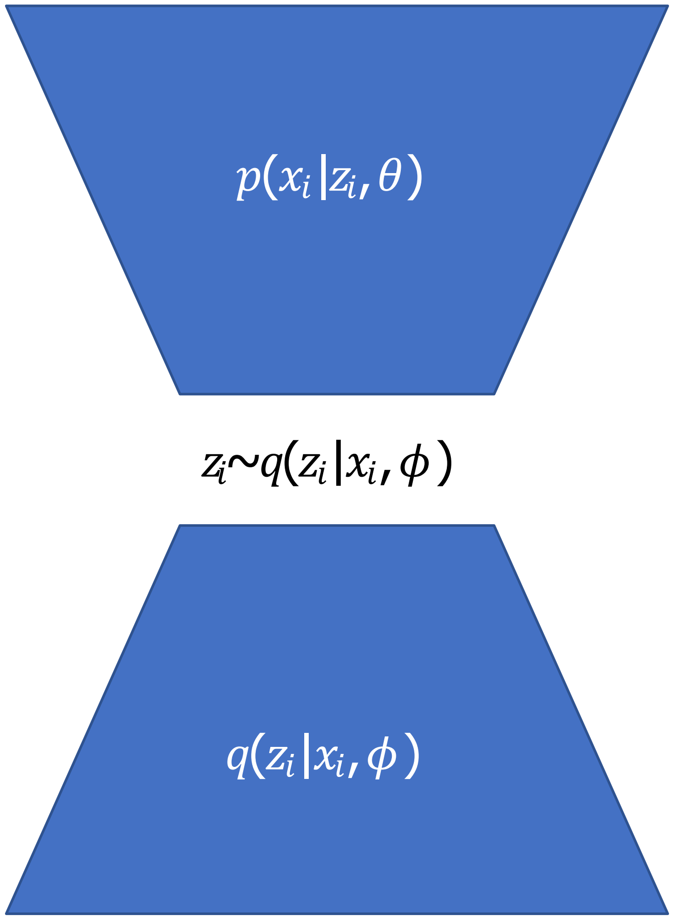
- We refer to the parameters of our networks, $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ collectively as ϕ
- Together, networks $\boldsymbol{u}$ and $\boldsymbol{s}$ parameterize a distribution, $q(z_i|x_i, \phi)$, of the latent variable $\boldsymbol{z_i}$ that depends in a complicated, non-linear way on $\boldsymbol{x_i}$

Unpacking the decoder



- The decoder follows the same logic, just swapping $\boldsymbol{x_i}$ and $\boldsymbol{z_i}$
- We refer to the parameters of our networks, $\boldsymbol{W_3}$ and $\boldsymbol{W_4}$ collectively as θ
- Together, networks $\boldsymbol{u_d}$ and $\boldsymbol{s_d}$ parameterize a distribution, $p(x_i|z_i, \theta)$, of the latent variable $\boldsymbol{x_i}$ that depends in a complicated, non-linear way on $\boldsymbol{z_i}$

Understanding the setup



- Note that p and q do not have to use the same distribution family, this was just an example
- This basically looks like an autoencoder, but the outputs of both the encoder and decoder are parameters of the distributions of the latent and observed variables respectively
- We also have a sampling step in the middle

Using EM for training

- Initialize $\theta^{(0)}$
- At each iteration $t = 1, \ldots, T$
    - **E step:** Hold $\theta^{(t-1)}$ fixed, find $q^{(t)}$ which maximizes $\mathcal{L}\left(q, \theta^{(t-1)}\right)$
    - **M step:** Hold $q^{(t)}$ fixed, find $\theta^{(t)}$ which maximizes $\mathcal{L}\left(q^{(t)}, \theta\right)$

- We will use a modified EM to train the model, but we will transform it so we can use standard back propagation!

Using EM for training

- Initialize $\theta^{(0)}$
- At each iteration $t = 1, \ldots, T$
  - **E step:** Hold $\theta^{(t-1)}$ fixed, find $\phi^{(t)}$ which maximizes $\mathcal{L}\left(\phi, \theta^{(t-1)}, x\right)$
  - **M step:** Hold $\phi^{(t)}$ fixed, find $\theta^{(t)}$ which maximizes $\mathcal{L}\left(\phi^{(t)}, \theta, x\right)$

- First we modify the notation to account for our choice of using a parametric, conditional distribution q

Using EM for training

- Initialize $\theta^{(0)}$
- At each iteration $t = 1, \ldots, T$
  - **E step:** Hold $\theta^{(t-1)}$ fixed, find $\frac{\partial \mathcal{L}}{\partial \phi}$ to increase $\mathcal{L}\left(\phi, \theta^{(t-1)}, x\right)$
  - **M step:** Hold $\phi^{(t)}$ fixed, find $\frac{\partial \mathcal{L}}{\partial \theta}$ to increase $\mathcal{L}\left(\phi^{(t)}, \theta, x\right)$

- Instead of fully maximizing at each iteration, we just take a step in the direction that increases $\mathcal{L}$

Computing the Loss

- We need to compute the gradient for each mini-batch with B data samples using the ELBO/variational bound $\mathcal{L}(\phi, \theta, x_i)$ as the loss

$$\sum_{i=1}^{B} \mathcal{L}(\phi, \theta, x_i) = \sum_{i=1}^{B} -\mathrm{KL}\big(q(z_i|x_i, \phi) \,||\, p(x_i, z_i|\theta)\big) = \sum_{i=1}^{B} -\mathbb{E}_{q(z_i|x_i, \phi)}\left[\log\left[\frac{q(z_i|x_i, \phi)}{p(x_i, z_i|\theta)}\right]\right]$$

- Notice that this involves an **intractable integral** over all values of z
- We can use Monte Carlo sampling to approximate the expectation using L samples from $q(z_i|x_i, \phi)$:

$$\mathbb{E}_{q(z_i|x_i,\phi)}[f(z_i)] \simeq \frac{1}{L}\sum_{j=1}^{L} f(z_{i,j})$$

$$\mathcal{L}(\phi, \theta, x_i) \simeq \tilde{\mathcal{L}}^A(\phi, \theta, x_i) = \frac{1}{L}\sum_{j=1}^{L} \log p\big(x_i, z_{i,j}|\theta\big) - \log q(z_{i,j}|x_i, \phi)$$

A lower variance estimator of the loss

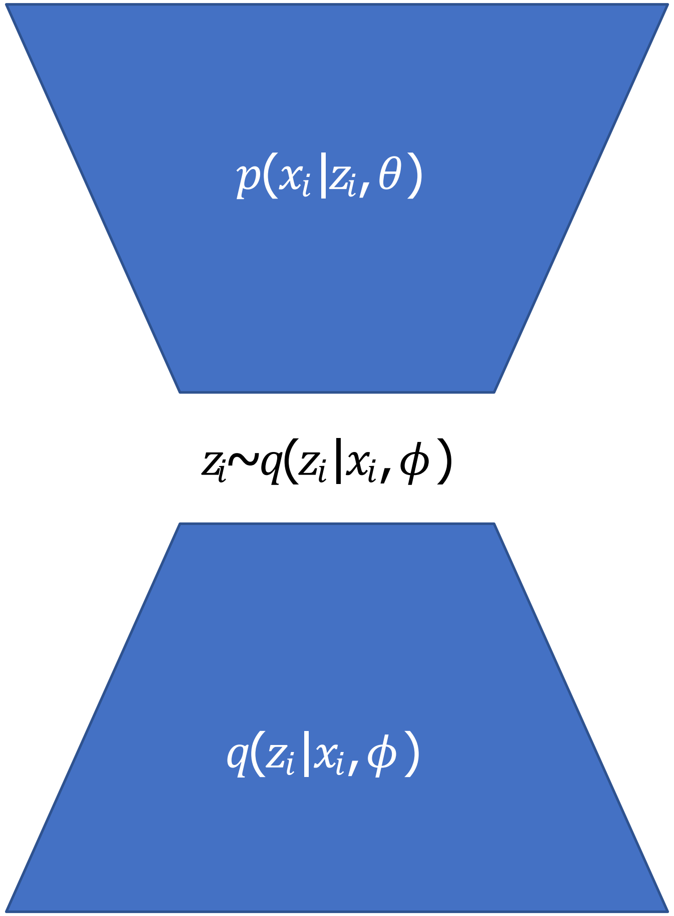
- We can rewrite

$$\begin{aligned}\mathcal{L}(\phi,\theta,x) &= -\mathrm{KL}\big(q(z|x,\phi)\,||\,p(x,z|\theta)\big)\\ &= -\int q(z|x,\phi)\log\left[\frac{q(z|x,\phi)}{p(x|z,\theta)p(z)}\right]\mathbf{d}z\\ &= -\int q(z|x,\phi)\left[\log\left[\frac{q(z|x,\phi)}{p(z)}\right]-\log p(x|z,\theta)\right]\mathbf{d}z =\\ &= -\mathrm{KL}\big(q(z|x,\phi)\,||\,p(z)\big)+\mathbb{E}_{q(z|x,\phi)}[\log p(x|z,\theta)]\end{aligned}$$

- The first term can be computed analytically for some families of distributions (e.g. Gaussian); only the second term must be estimated

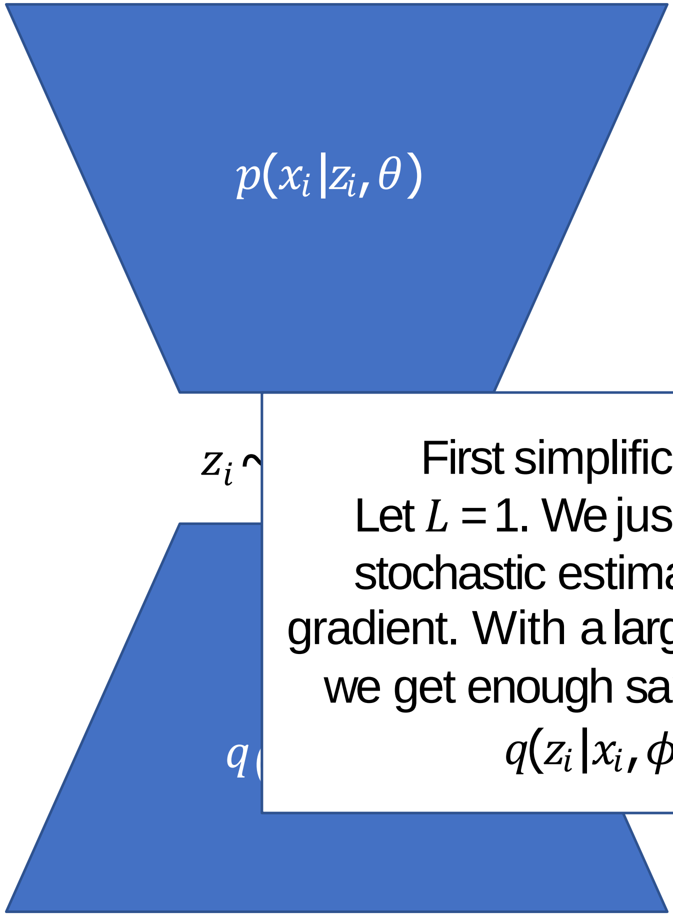
$$\begin{aligned}&\mathcal{L}(\phi,\theta,x_i)\\ &\simeq \tilde{\mathcal{L}}^B(\phi,\theta,x_i) = -\mathrm{KL}\big(q(z_i|x_i,\phi)\,||\,p(z_i)\big)+\frac{1}{L}\sum_{j=1}^{L}\log p\big(x_i|z_{i,j},\theta\big)\end{aligned}$$

Full EM training procedure (not really used)



- For $t = 1 : b : T$
  - Estimate $\frac{\partial \mathcal{L}}{\partial \phi}$ (How do we do this? We'll get to it shortly)
  - Update ϕ
  - Estimate $\frac{\partial \mathcal{L}}{\partial \theta}$:
    - Initialize $\Delta\theta = 0$
    - For $i = t : t + b - 1$
      - Compute the outputs of the encoder (parameters of q) for x_i
      - For $\ell = 1, \ldots L$
        - Sample $z_i \sim q(z_i|x_i,\phi)$
        - $\Delta\theta_{i,\ell} \leftarrow$ Run forward/backward pass on the decoder (standard back propagation) using either $\mathcal{L}^A$ or $\mathcal{L}^B$ as the loss
        - $\Delta\theta \leftarrow \Delta\theta + \Delta\theta_{i,\ell}$
  - Update θ

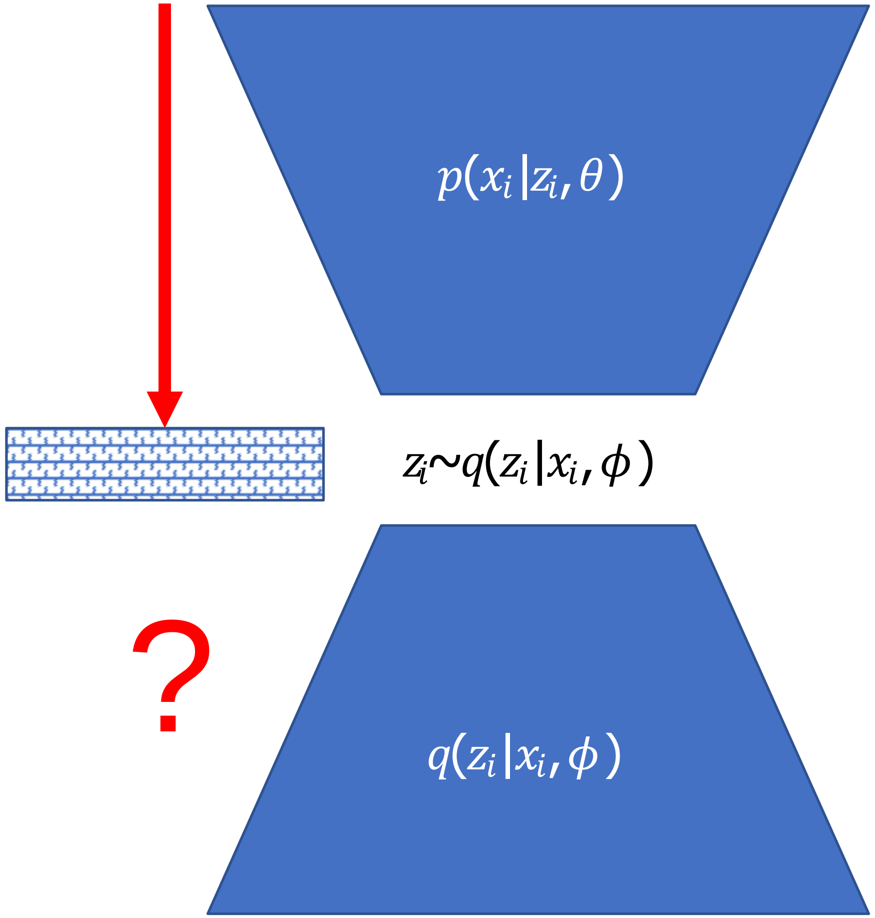
Full EM training procedure (not really used)



First simplification:
Let $L=1$. We just want a stochastic estimate of the gradient. With a large enough B, we get enough samples from $q(z_i|x_i,\phi)$

- For $t = 1 : b : T$
  - Estimate $\frac{\partial \mathcal{L}}{\partial \phi}$ (How do we do this? We'll get to it shortly)
  - Update ϕ
  - Estimate $\frac{\partial \mathcal{L}}{\partial \theta}$:
    - Initialize $\Delta\theta = 0$
    - For $i = t : t + b - 1$
      - Compute the outputs of the encoder (parameters of q) for x_i
      - Sample $z_i \sim q(z_i|x_i,\phi)$
      - $\Delta\theta_i \leftarrow$ Run forward/backward pass on the decoder (standard back propagation) using either $\mathcal{L}^A$ or $\mathcal{L}^B$ as the loss
      - $\Delta\theta \leftarrow \Delta\theta + \Delta\theta_i$
  - Update θ

The E step



- We can use standard back propagation to estimate $\frac{\partial \mathcal{L}}{\partial \theta}$
- How do we estimate $\frac{\partial \mathcal{L}}{\partial \phi}$?
- The sampling step blocks the gradient flow
- Computing the derivatives through q via the chain rule gives a very high variance estimate of the gradient

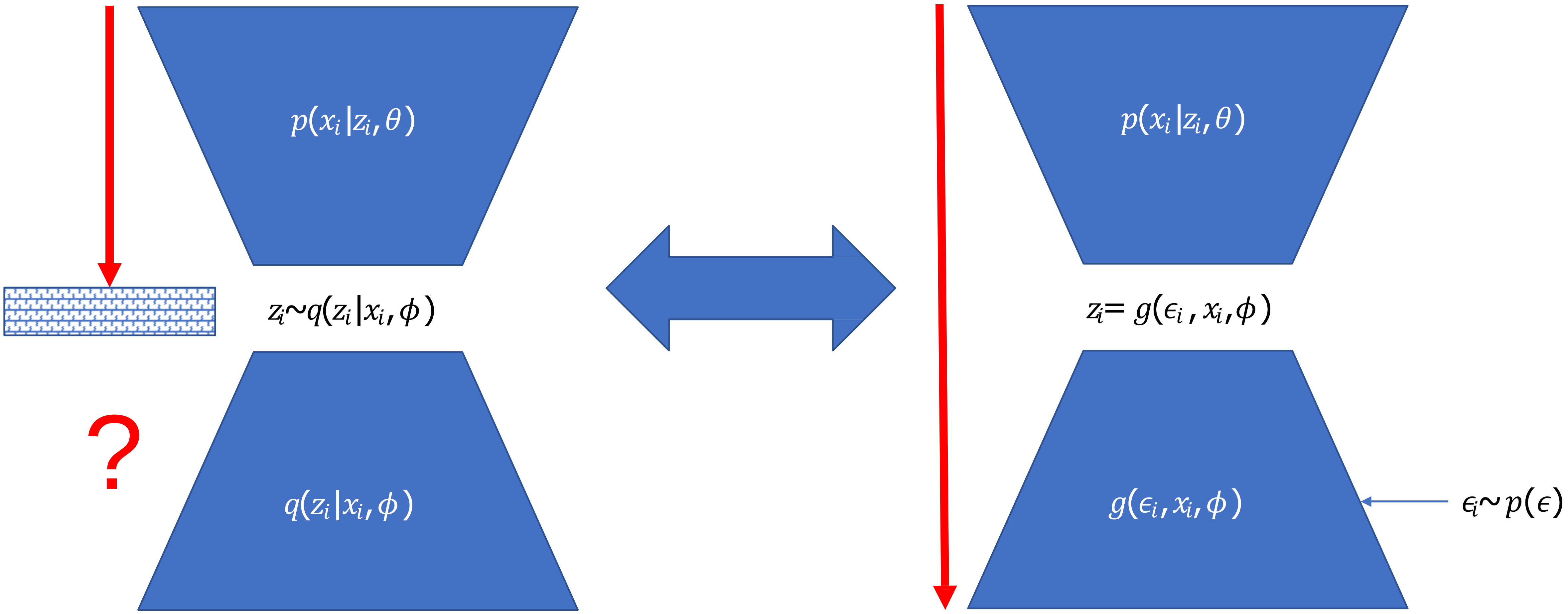
Reparameterization

- Instead of drawing $z_i \sim q(z_i | x_i, \phi)$,
  let $z_i = \mathrm{g}(\epsilon_i, x_i, \phi)$, and draw $\epsilon_i \sim p(\epsilon)$
- z_i is still a random variable but depends on ϕ **deterministically**
- Replace $\mathbb{E}_{q(z_i|x_i,\phi)}[f(z_i)]$ with $\mathbb{E}_{p(\epsilon)}[f(\mathrm{g}(\epsilon_i, x_i, \phi))]$
- Example – univariate normal:

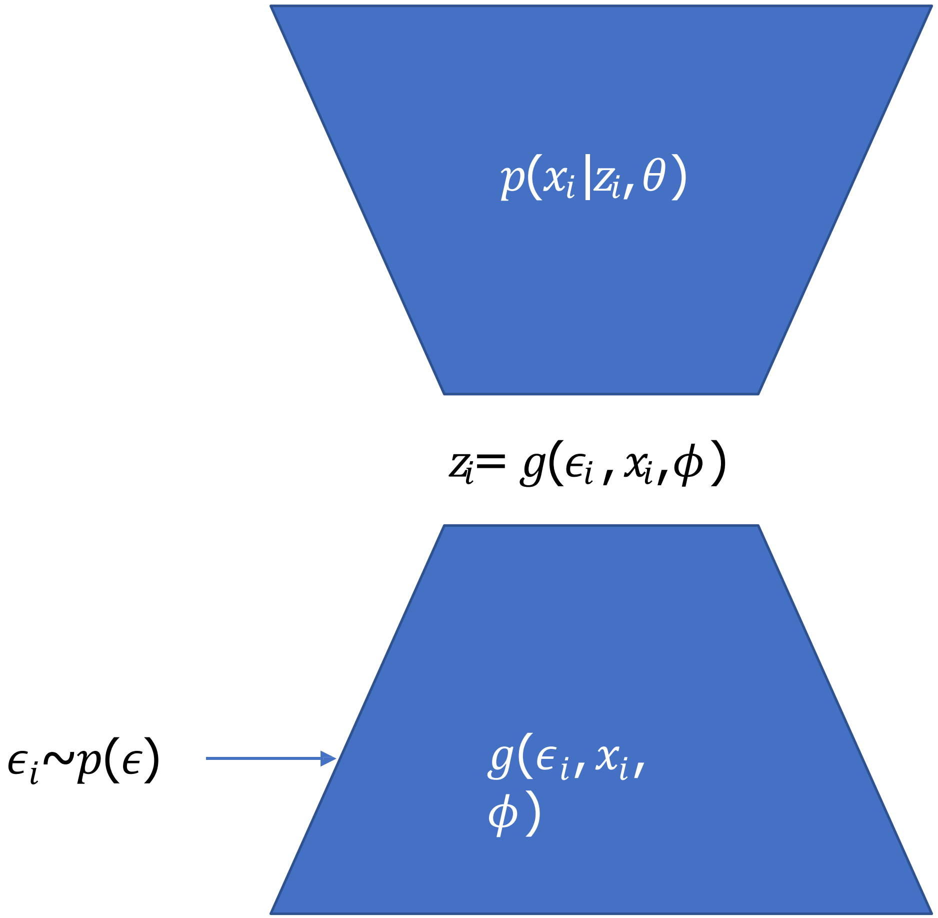
  $a \sim \mathcal{N}(\mu, \sigma^2)$ is equivalent to

  $a = \mathrm{g}(\epsilon), \epsilon \sim \mathcal{N}(0, 1), \mathrm{g}(b) \triangleq \mu + \sigma b$

Reparameterization

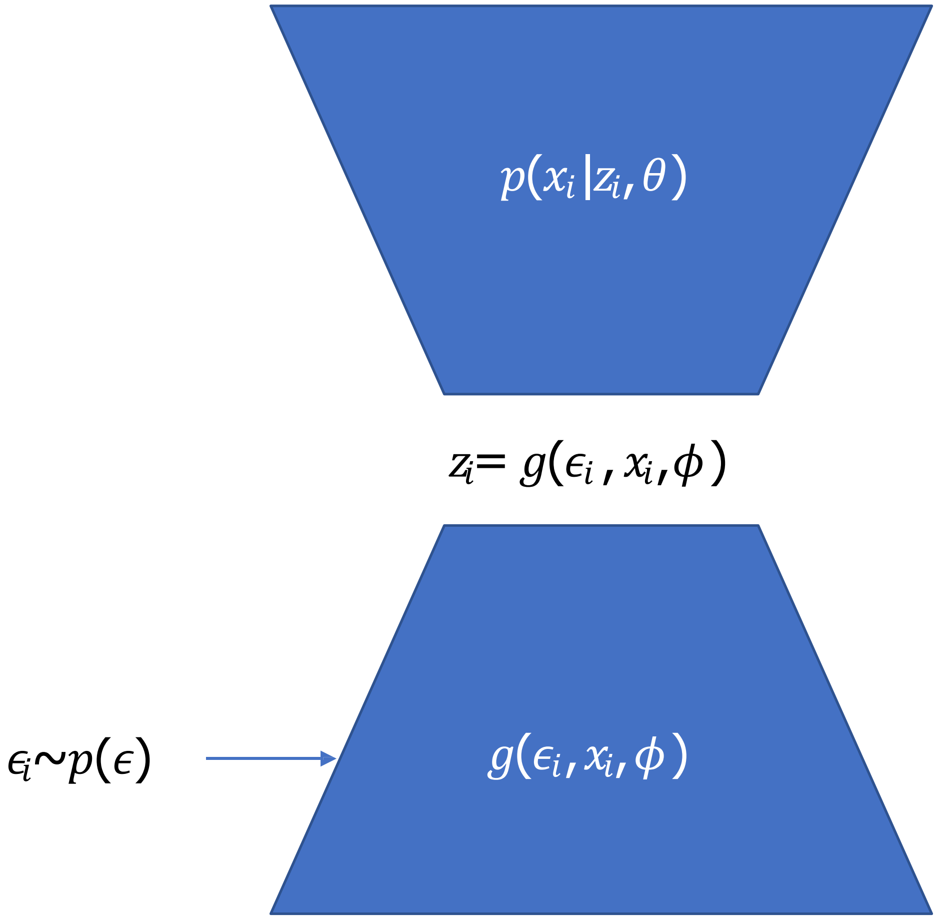


Full E M training procedure (not really used)



- For $t = 1 : b : T$
  - E Step
    - Estimate $\frac{\partial \mathcal{L}}{\partial \phi}$ using standard back propagation with either $\tilde{\mathcal{L}}^A$ or $\tilde{\mathcal{L}}^B$ as the loss
    - Update ϕ
  - M Step
    - Estimate $\frac{\partial \mathcal{L}}{\partial \theta}$ using standard back propagation with either $\tilde{\mathcal{L}}^A$ or $\tilde{\mathcal{L}}^B$ as the loss
    - Update θ

Full training procedure



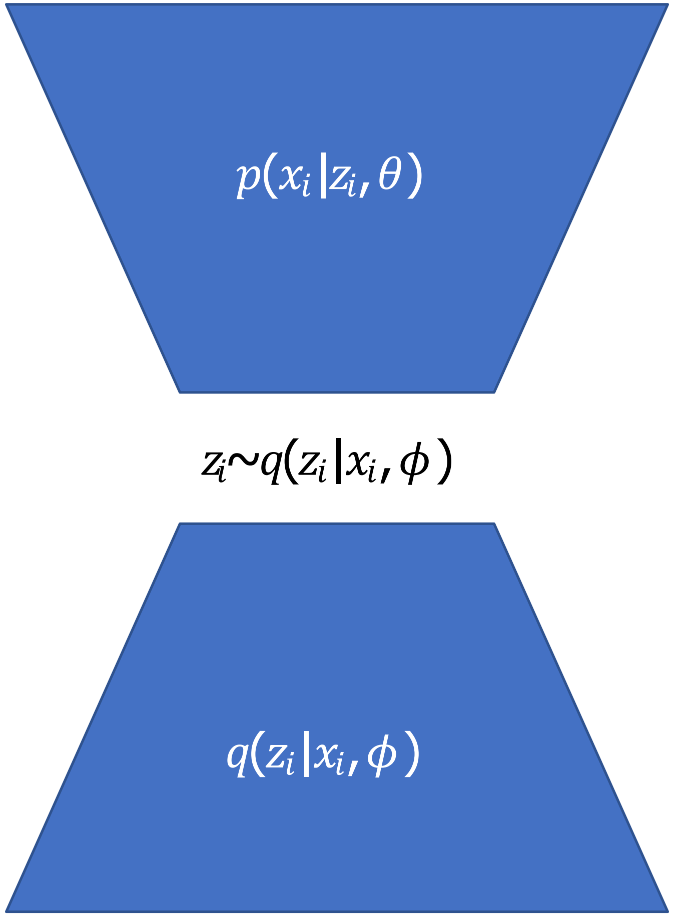
- For $t = 1 : b : T$
  - Estimate $\frac{\partial \mathcal{L}}{\partial \phi}, \frac{\partial \mathcal{L}}{\partial \theta}$ with either $\tilde{\mathcal{L}}^A$ or $\tilde{\mathcal{L}}^B$ as the loss
  - Update ϕ, θ

- Final simplification: update all of the parameters at the same time instead of using separate E, M steps
- This is standard back propagation. Just use $-\tilde{\mathcal{L}}^A$ or $-\tilde{\mathcal{L}}^B$ as the loss, and run your favorite SGD variant

Running the model on new data

- To get a MAP estimate of the latent variables, just use the mean output by the encoder (for a Gaussian distribution)
- No need to take a sample
- Give the mean to the decoder
- **At test time, this is used just as an auto-encoder**
- You can optionally take multiple samples of the latent variables to estimate the uncertainty

Relationship to Factor Analysis



- VAE performs **probabilistic, non-linear dimensionality reduction**
- It uses a **generative model** with a latent variable distributed according to some prior distribution $p(z_i)$
- The observed variable is distributed according to a conditional distribution $p(x_i|z_i,\theta)$
- Training is approximately running **expectation maximization** to maximize the data likelihood
- This can be seen as a **non-linear version of Factor Analysis**

Regularization by a prior

- Looking at the form of $\mathcal{L}$ we used to justify $\mathcal{L}^{ELBO}$ gives us additional insight

$$\mathcal{L}(\phi, \theta, x) = -\mathrm{KL}\big(q(z|x, \phi) \,||\, p(z)\big) + \mathbb{E}_{q(z|x,\,\phi)}[\log p(x|z, \theta)]$$

- We are making the latent distribution as close as possible to a prior on z
- While maximizing the conditional likelihood of the data under our model
- In other words this is an approximation to **Maximum Likelihood Estimation regularized by a prior on the latent space**

Practical advantages of a VAE vs. an AE

- The prior on the latent space:
  - Allows you to inject domain knowledge
  - Can make the latent space more interpretable
- The VAE also makes it possible to estimate the variance/uncertainty in the predictions

What does this look like in code

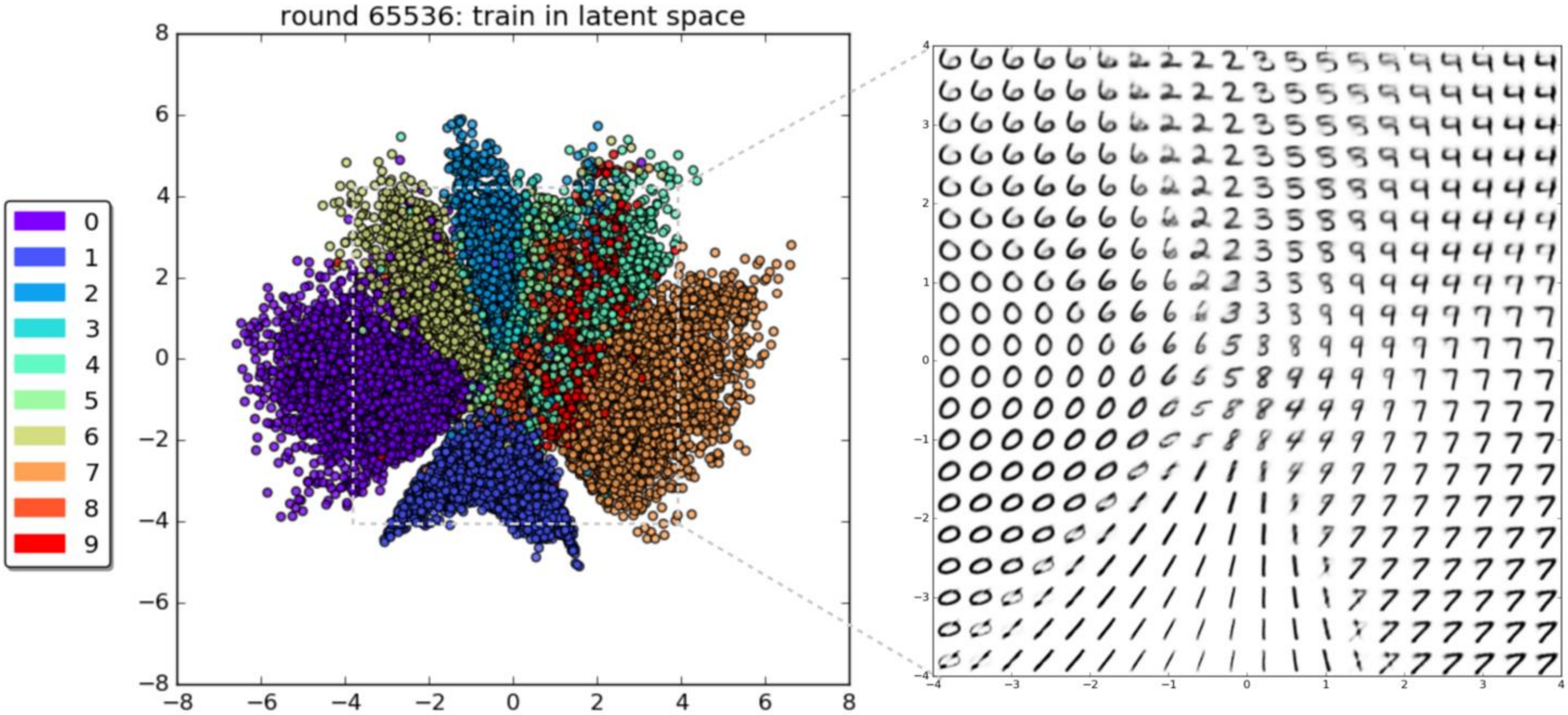
```python
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.fc1 = nn.Linear(784, 400)
        self.fc21 = nn.Linear(400, 20)
        self.fc22 = nn.Linear(400, 20)
        self.fc3 = nn.Linear(20, 400)
        self.fc4 = nn.Linear(400, 784)

        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
```

```python
def encode(self, x):
    h1 = self.relu(self.fc1(x))
    return self.fc21(h1), self.fc22(h1)

def reparameterize(self, mu, logvar):
    if self.training:
        std = logvar.mul(0.5).exp_()
        eps = Variable(std.data.new(std.size()).normal_())
        return eps.mul(std).add_(mu)
    else:
        return mu

def decode(self, z):
    h3 = self.relu(self.fc3(z))
    return self.sigmoid(self.fc4(h3))
```

```python
def forward(self, x):
    mu, logvar = self.encode(x.view(-1, 784))
    z = self.reparameterize(mu, logvar)
    return self.decode(z), mu, logvar
```

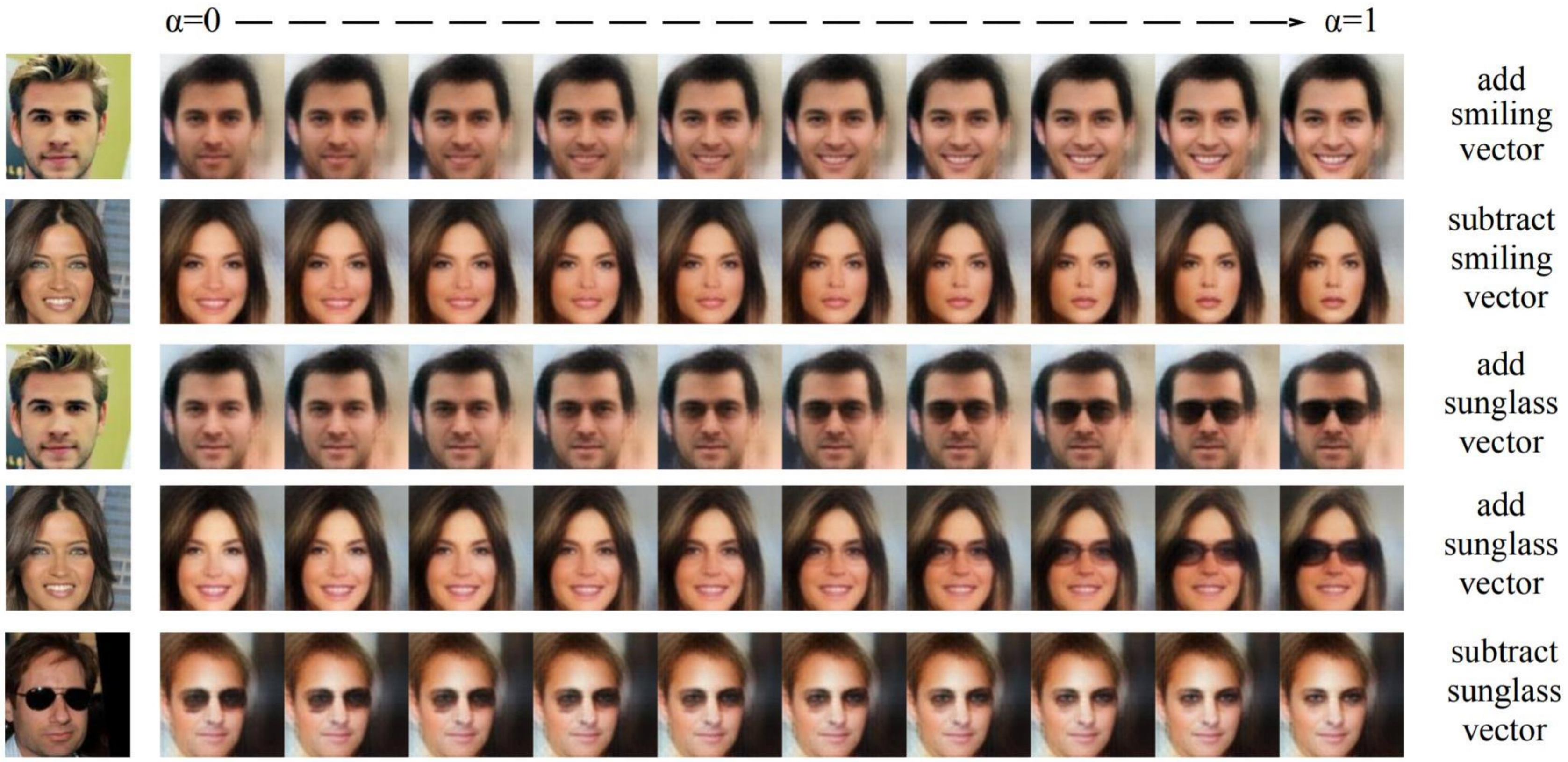

PyTorch Example Code: https://github.com/pytorch/examples/blob/master/vae/main.py

What does our hidden space look like now?



http://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and.html

Interpreting the latent space



https://arxiv.org/pdf/1610.00291.pdf

Requirements of the VAE

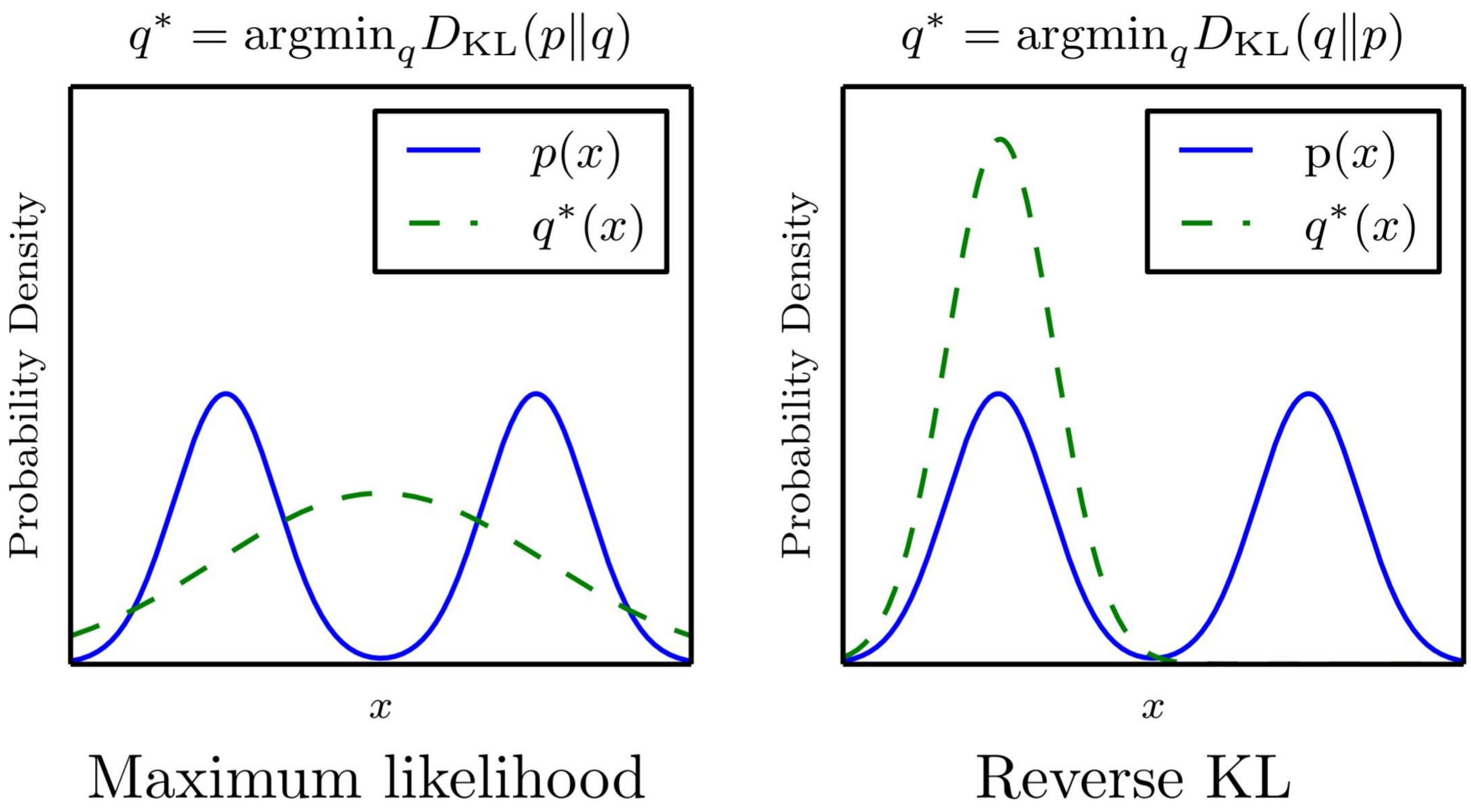
- Note that the VAE requires 2 tractable distributions to be used:
  - The prior distribution $p(z)$ must be easy to sample from
  - The conditional likelihood $p(x|z, \theta)$ must be computable
- In practice this means that the 2 distributions of interest are often simple, for example uniform, Gaussian, or even isotropic Gaussian

The blurry image problem

https://blog.openai.com/generative-models/

- The samples from the VAE look blurry
- Three plausible explanations for this
  - Maximizing the likelihood
  - Restrictions on the family of distributions
  - The lower bound approximation

The maximum likelihood explanation



- Recent evidence suggests that this is not actually the problem
- GANs can be trained with maximum likelihood and still generate sharp examples

https://arxiv.org/pdf/1701.00160.pdf

Investigations of blurriness

- Recent investigations suggest that both the simple probability distributions and the variational approximation lead to blurry images
- Kingma & colleages: Improving Variational Inference with Inverse Autoregressive Flow
- Zhao & colleagues: Towards a Deeper Understanding of Variational Autoencoding Models
- Nowozin & colleagues: f-gan: Training generative neural samplers using variational divergence minimization