Artificial Neural Networks:
On Variational Autoencoders

Alexander G. Ororbia II

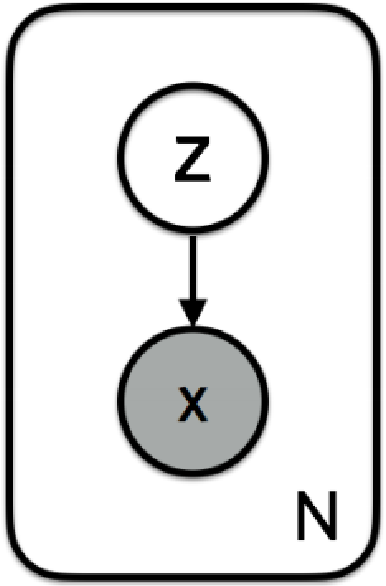
Introduction to Machine Learning

CSCI-635

12/8/2023

Probabilistic Model Perspective

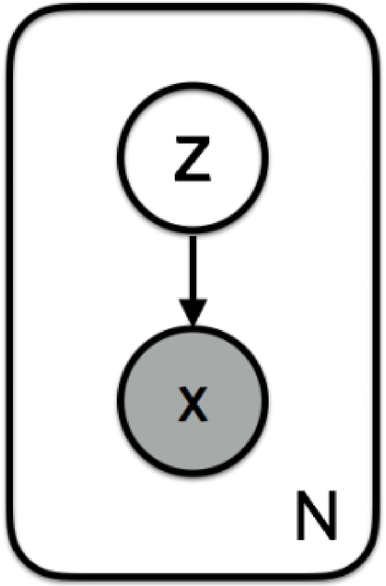
- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



To learn this model, we could appeal to Monte Carlo sampling or to the calculus of variations…

Probabilistic Model Perspective

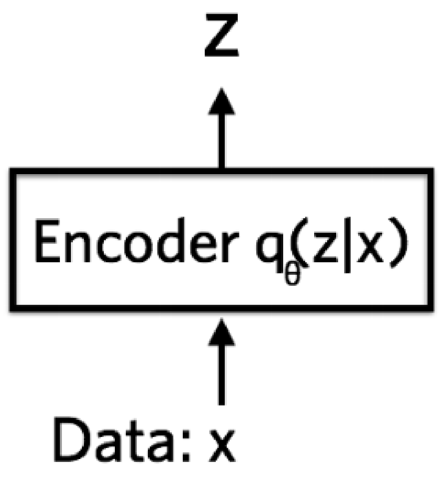
- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



Not sure if a learnable generative model…

…or an intractable waste of time.

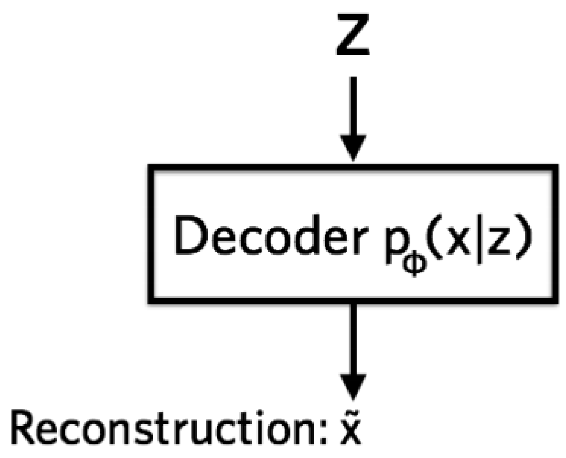
…so we ‘re going to develop a ***variational inference*** scheme using your neural building blocks!

- Goal: Build a neural network that generates digits from random (Gaussian) noise
- Define two sub-networks: Encoder and Decoder
- Define a Loss Function
  - A neural network $q_\theta(z|x)$
  - Input: datapoint x (e.g. 28×28-pixel digit)
  - Output: encoding z, drawn from Gaussian density with parameters θ
  - $|z| \ll |x|$
  - 

The variational distribution!

- Goal: Build a neural network that generates digits from random (Gaussian) noise
- Define two sub-networks: Encoder and Decoder
- Define a Loss Function

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder
- Output: reconstruction $\tilde{x}$, drawn from distribution of the data
- E.g., output parameters for 28×28 Bernoulli variables



-

The Loss:

- $\tilde{x}$ is reconstructed from z where $|z| \ll |\tilde{x}|$
- How much information is lost when we go from x to z to $\tilde{x}$?
- Measure this with reconstruction log-likelihood: $\log p_\phi(x|z)$
- Measures how effectively the decoder has learned to reconstruct x given the latent representation z

- Loss function is negative reconstruction log-likelihood + regularizer
- Loss decomposes into term for each datapoint:

$$L(\theta, \phi) = \sum_{i=1}^{N} l_i(\theta, \phi)$$

- Loss for datapoint x_i:

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] + KL\left(q_\theta(z|x_i)||p(z)\right)$$

The Cost: $$L(\theta, \phi) = \sum_{i=1}^{N} \left(-\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] + KL\left(q_\theta(z|x_i)||p(z)\right)\right)$$

- Negative reconstruction log-likelihood:

$$-\mathbb{E}_{z\sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right]$$

- Encourages decoder to learn to reconstruct the data
- Expectation taken over distribution of latent representations

- KL Divergence as regularizer:

$$KL(q_\theta(z|x_i)||p(z)) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\left[\log q_\theta(z|x_i) - \log p(z)\right]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Encourages encoder to produce z's that are close to standard normal distribution
- Encoder learns a meaningful representation of MNIST digits
- Representation for images of the same digit are close together in latent space
- Otherwise could "memorize" the data and map each observed datapoint to a distinct region of space

Neural Variational Inference (NVIL)

- *Idea*: Teach neural net to approximate the posterior *p(z|x)*
  - *q(z|x)* with 'variational parameters' ϕ
  - One-shot approximate inference
  - Also known as a recognition model
    - Construct estimator of the variational (evidence) lower bound (ELBO)
    - Can optimize jointly w.r.t. ϕ jointly with θ -> Stochastic gradient ascent

D_{KL} KL-Divergence >= 0 depends on how good q(z|x) can approximate p(z|x)

Recall from the start of the semester:

KL Divergence:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]. \tag{3.50}$$

Gaussian KL Divergence:

$$KL(p,q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

$$p \sim N(\mu_1, \sigma_1)$$
$$q \sim N(\mu_2, \sigma_2)$$

Not sure if optimizing log likelihood…

…or a variational evidence lower bound!

Note: this form is in terms of log likelihood

$$\mathcal{F}(\mathbf{x}, q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})\|p(\mathbf{z})]$$

Approx. Posterior — Reconstruction — Penalty

Interpreting the bound:

- **Approximate posterior distribution** $q(z|x)$: Best match to true posterior $p(z|x)$, one of the unknown inferential quantities of interest to us.

- **Reconstruction cost**: The expected log-likelihood measures how well samples from $q(z|x)$ are able to explain the data x.

- **Penalty:** Ensures that the explanation of the data $q(z|x)$ doesn't deviate too far from your beliefs $p(z)$. A mechanism for realising Ockham's razor.

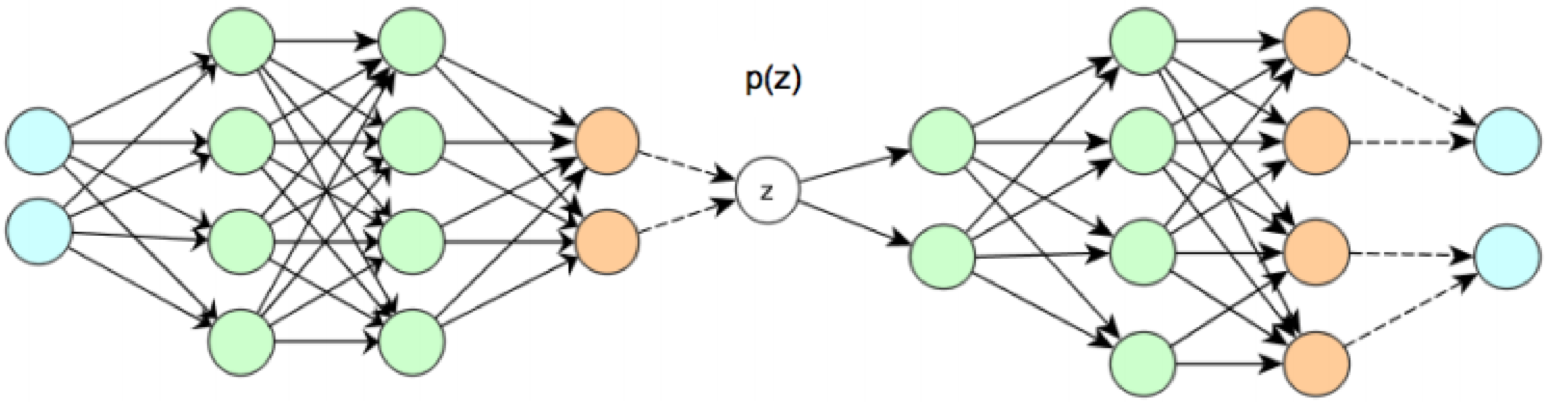
The Variational Auto-Encoder

- A feed forward NN + Gaussian

$$q_{\theta}(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$$

Just a Gaussian, with diagonal covariance.

$q_{\theta}(x|z)$ $\quad$ $p_{\varphi}(x|z)$ $\quad$ $x|z \sim N(\mu_x, \sigma_x^{\,2})$

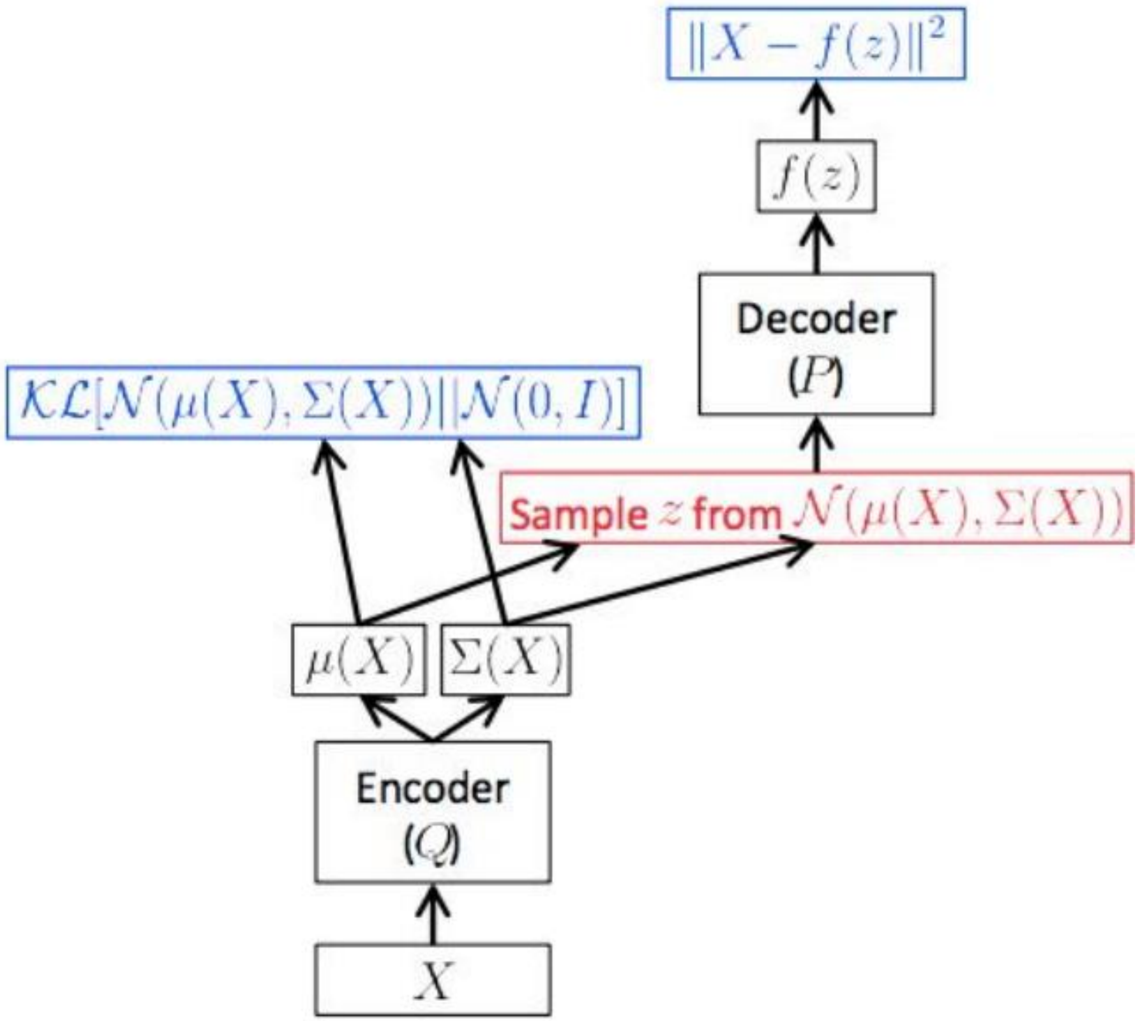


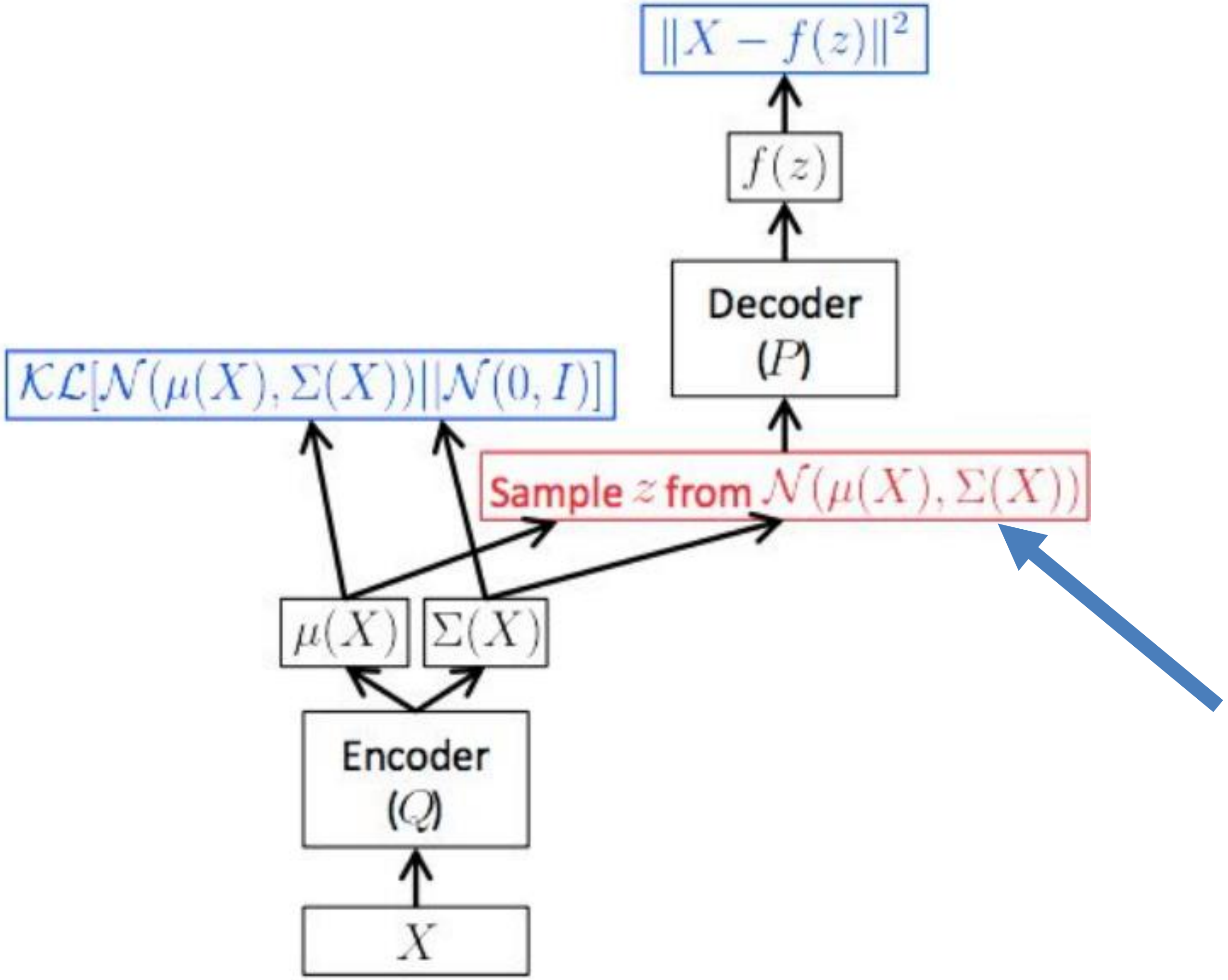
- For illustration z one dimensional x 2D
- Want a complex model of distribution of x given z

Learning the parameters φ and θ via backpropagation

Determining the loss function

But…we have a **problem**!



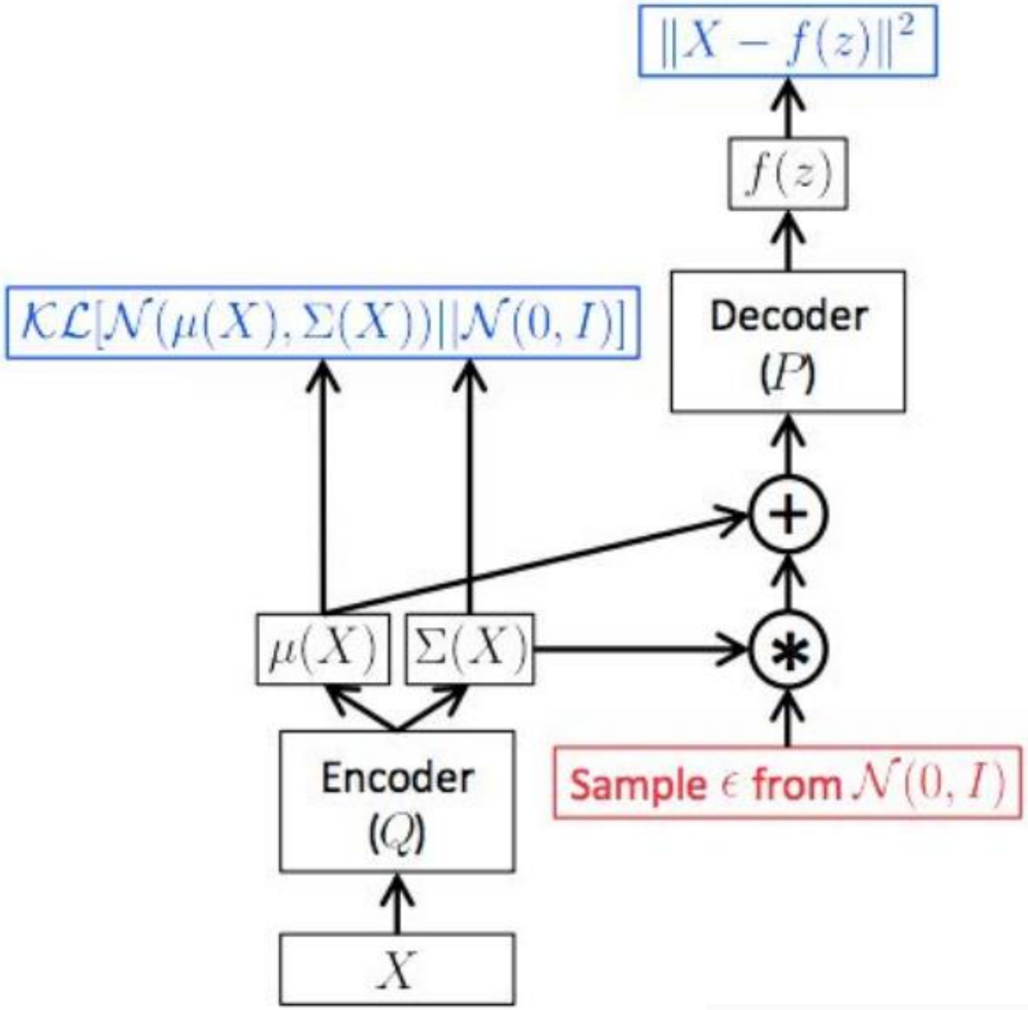


?? Backprop through numpy's Gaussian RNG**??**

The Reparameterization 'Trick'

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product
- Can take derivatives of (functions of) z w.r.t. μ and σ
- Output of $q_\theta(z|x)$ is vector of μ's and vector of σ's

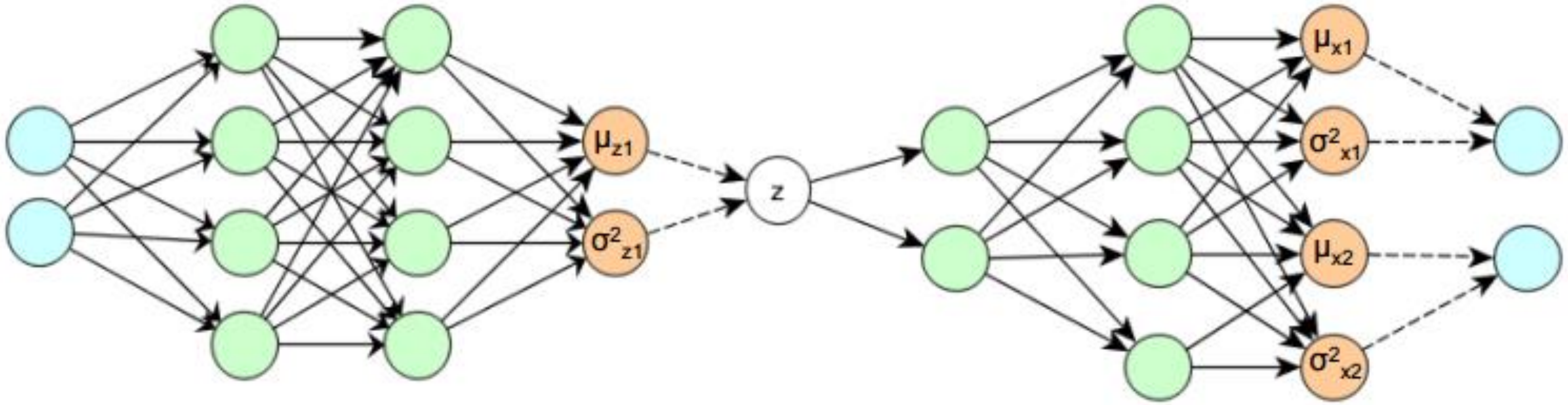
The Reparameterization Trick



z ~ N(μ, σ) is equivalent to
μ + σ · ε, where ε ~ N(0, 1)

Putting It All Together!

Prior p(z) ~ N(0,1) and p, q Gaussian, extension to dim(z) > 1 trivial



Cost: Regularisation

$$-D_{KL}\left(q(z|x^{(i)})\|p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1+\log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$

Cost: Reproduction

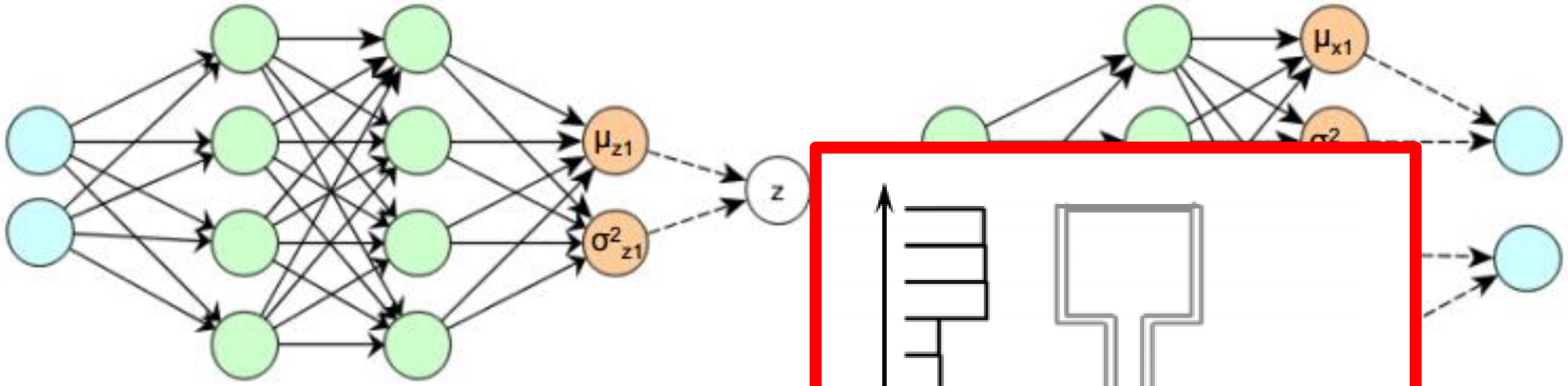
$$-\log\left(p(x^{(i)}|z^{(i)})\right) = \sum_{j=1}^{D}\frac{1}{2}\log(\sigma_{x_j}^2) + \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$

We use mini batch gradient decent to optimize the cost function over all $x^{(i)}$ in the mini batch

Least Square for constant variance

Putting It All Together!

Prior p(z) ~ N(0,1) and p, q Gaussian, extension to dim(z) > 1 trivial

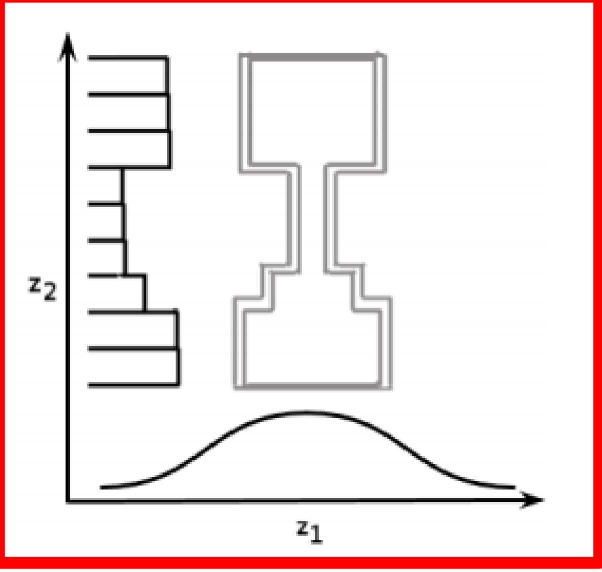


Cost: Regularisation

$$-D_{KL}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2}\right.$$

Why not capture modes (in text) with piecewise linear variables?
(Serban & Ororbia, 2016)

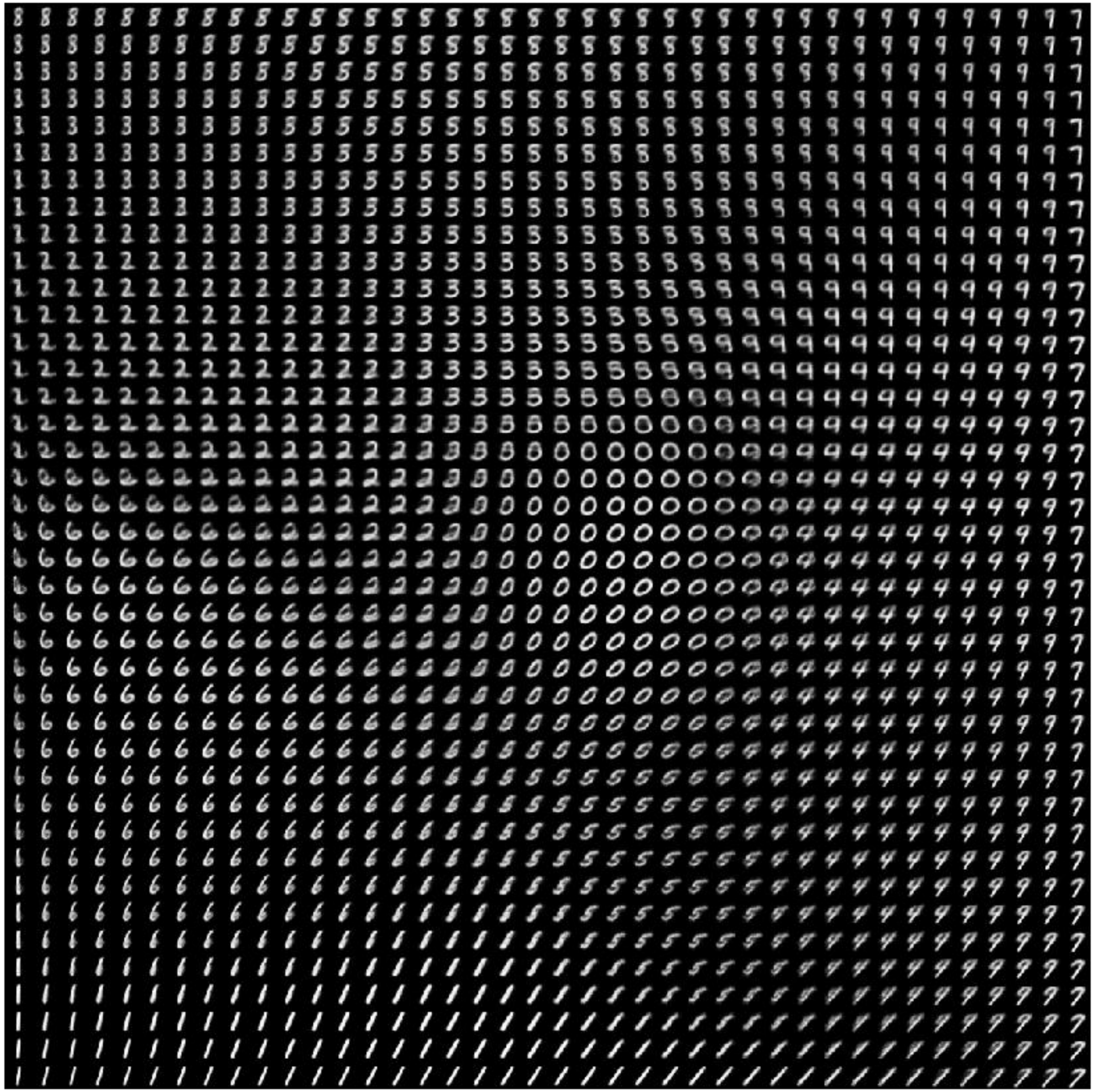
$$\sum_{j=1} 2 \quad \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$



lient
cost
e mini

Least Square for constant variance

Samples….



QUESTIONS?