

Introduction to variational autoencoders

Abstract

Variational autoencoders are interesting generative models, which combine ideas from deep learning with statistical inference. They can be used to learn a low dimensional representation Z of high dimensional data X such as images (of e.g. faces). In contrast to standard auto encoders, X and Z are random variables. It's therefore possible to sample X from the distribution $P(X|Z)$, thus creating e.g. [images of faces](#), [MNIST Digits](#), or [speech](#).

In this talk I will in some detail describe the paper of Kingma and Welling. “*Auto-Encoding Variational Bayes*, *International Conference on Learning Representations*.” ICLR, 2014.
[arXiv:1312.6114](#) [stat.ML].

I will also show some code. A TensorFlow notebook can be found at:
https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/vae/vae_demo.ipynb

Introduction to variational autoencoders

Oliver Dürr

Datalab-Lunch Seminar Series

Winterthur, 11 May, 2016

Motivation: Generating Faces



Other examples

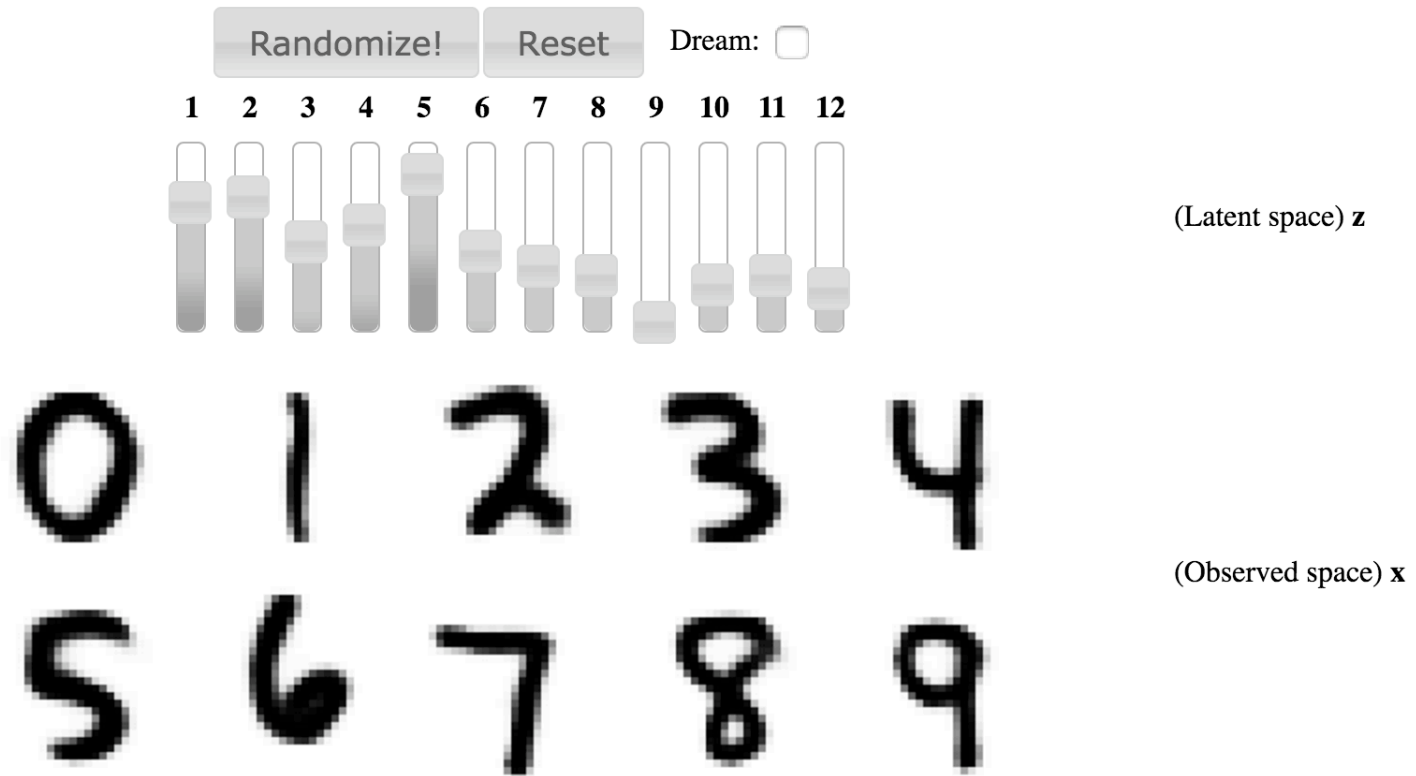
- [random faces](#)
- [MNIST](#)
- [Speech](#)

just google vae...

← These are not part of the trainingset!

<https://www.youtube.com/watch?v=XNZIN7Jh3Sg>

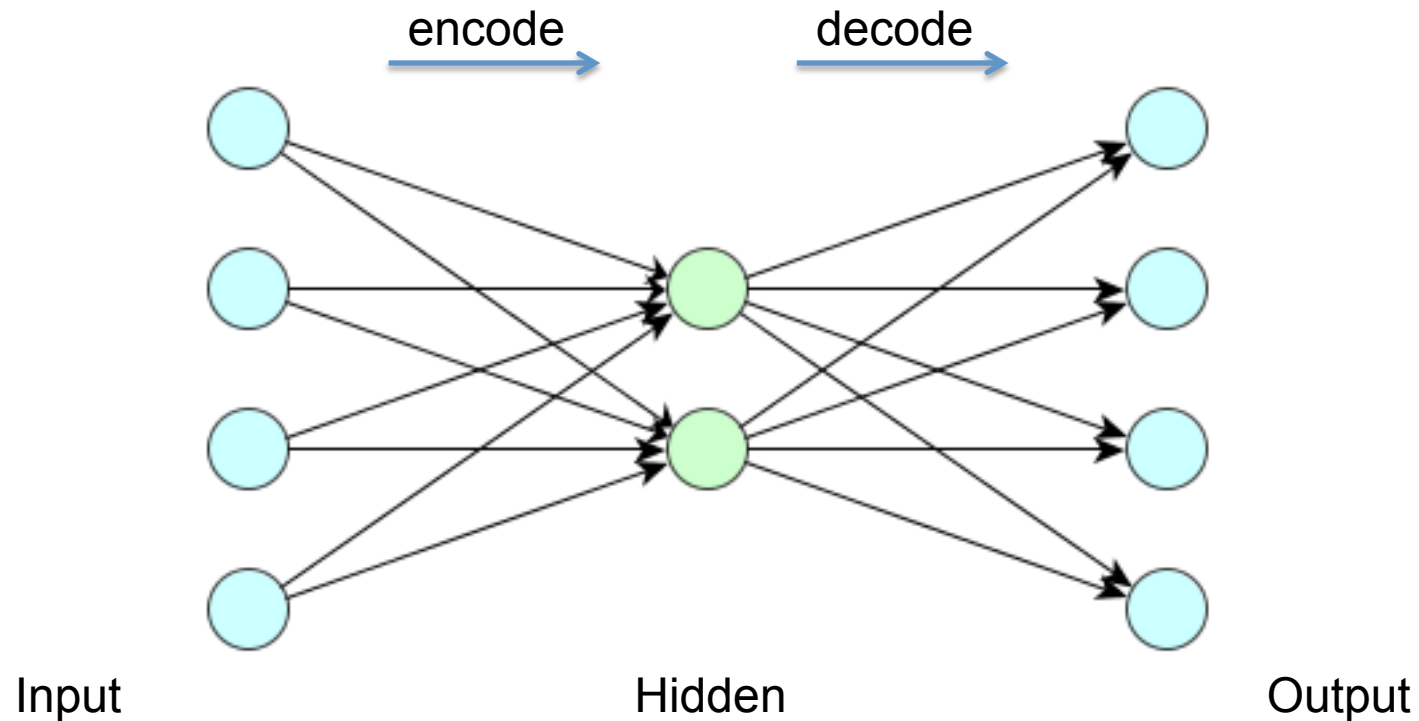
Motivation: Generating Hand Written Digits



http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Idea

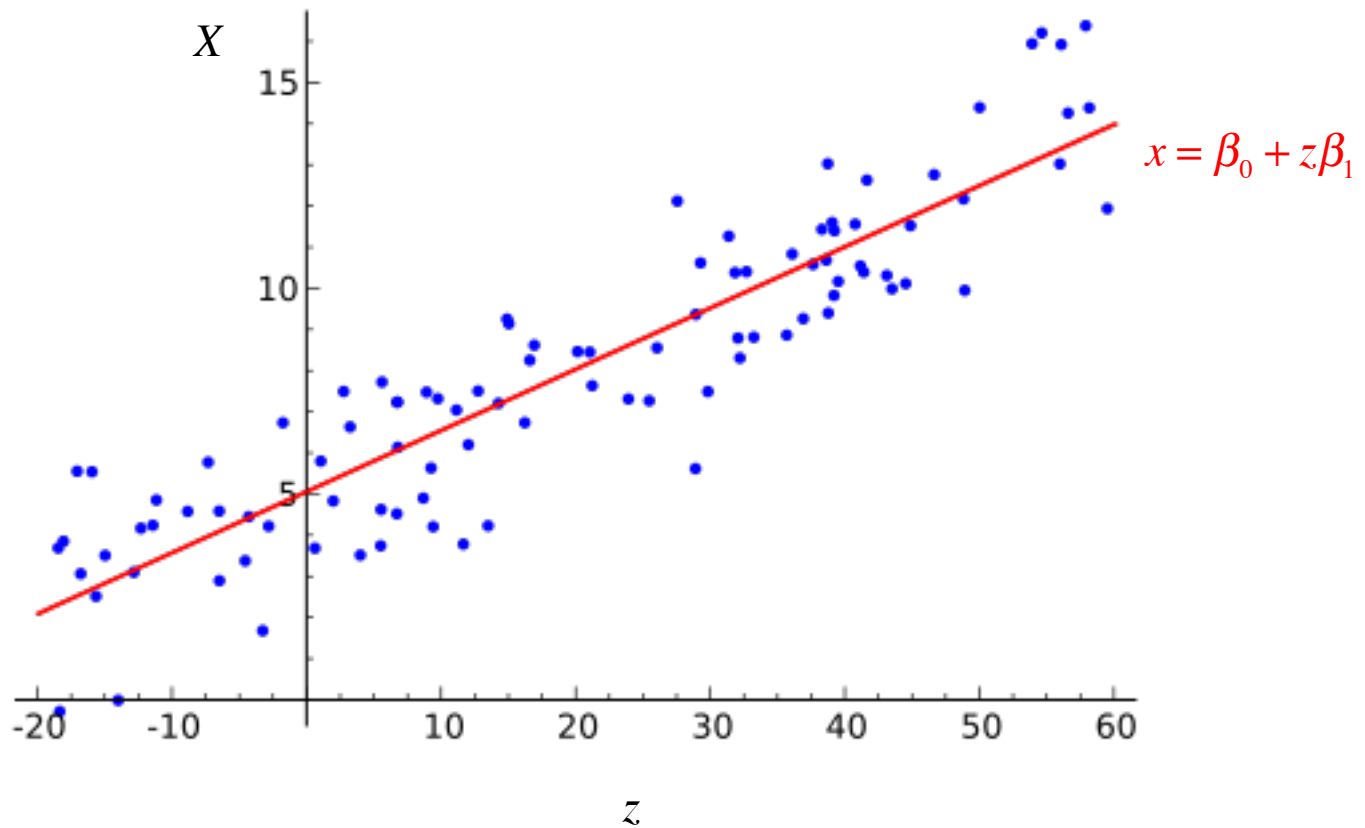
Recap: Auto Encoders ('classical')



A simple autoencoder more see [Beates talk on Autoencoders](#)

Recap: Linear Regression

Most people think of linear regression as points and a straight line:



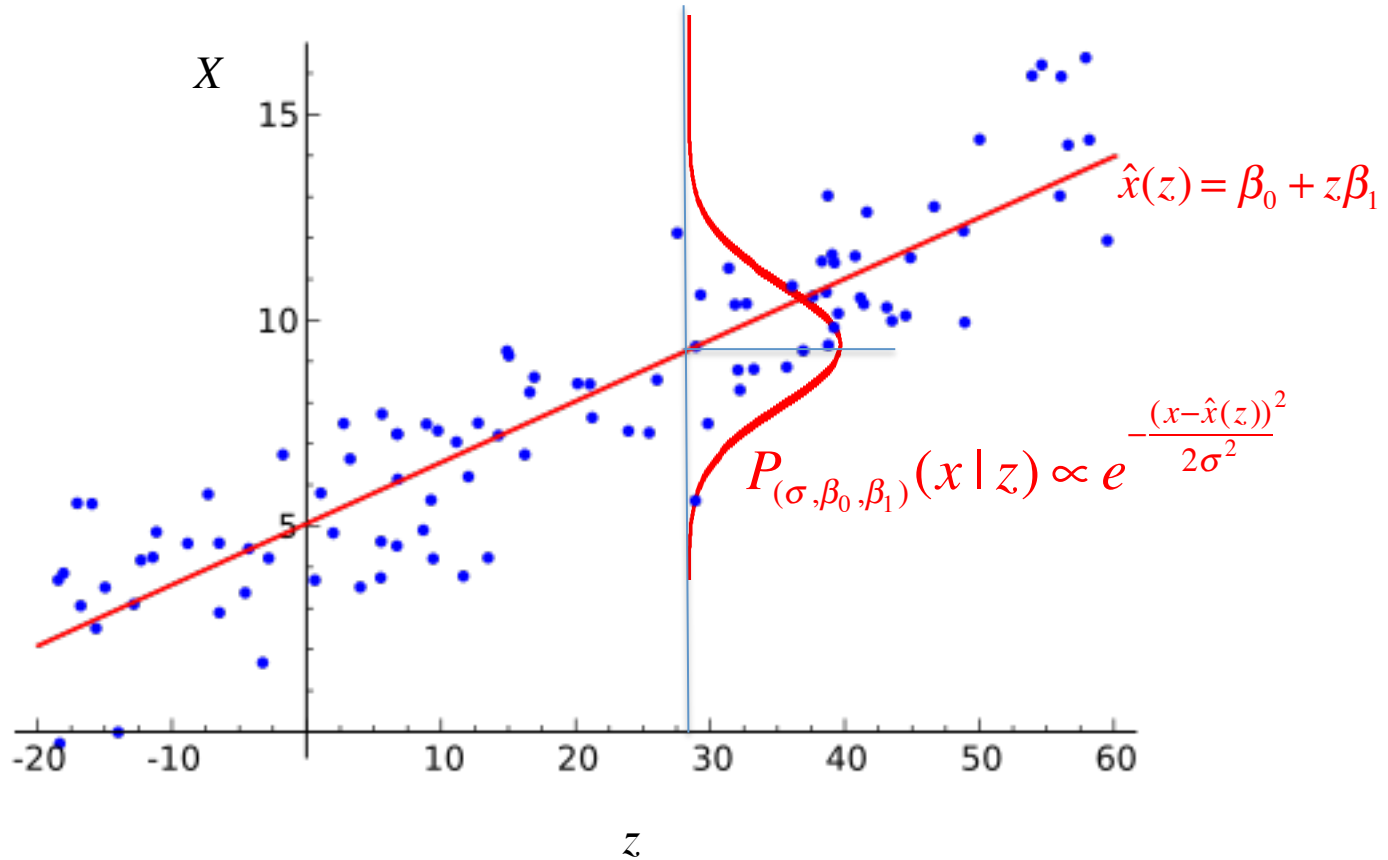
Strange axis names, to be compatible with later notation

Recap: Linear Regression

Benefits of having an error model:

- How likely is a data point
- Confidence bounds
- Compare models

Statisticians additionally have $P_{\theta}(X|Z)$



Strange axis names, to be compatible with later notation

Recap: Linear Regression (as a graphical model)

Statisticians additionally have $P_{\theta}(X|Z)$

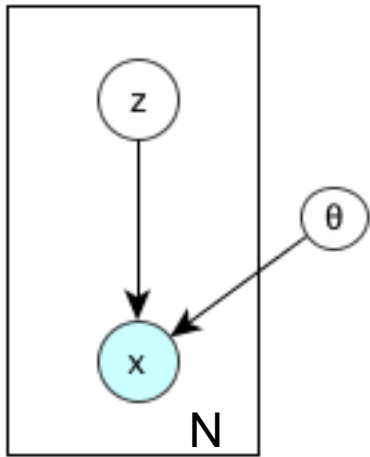
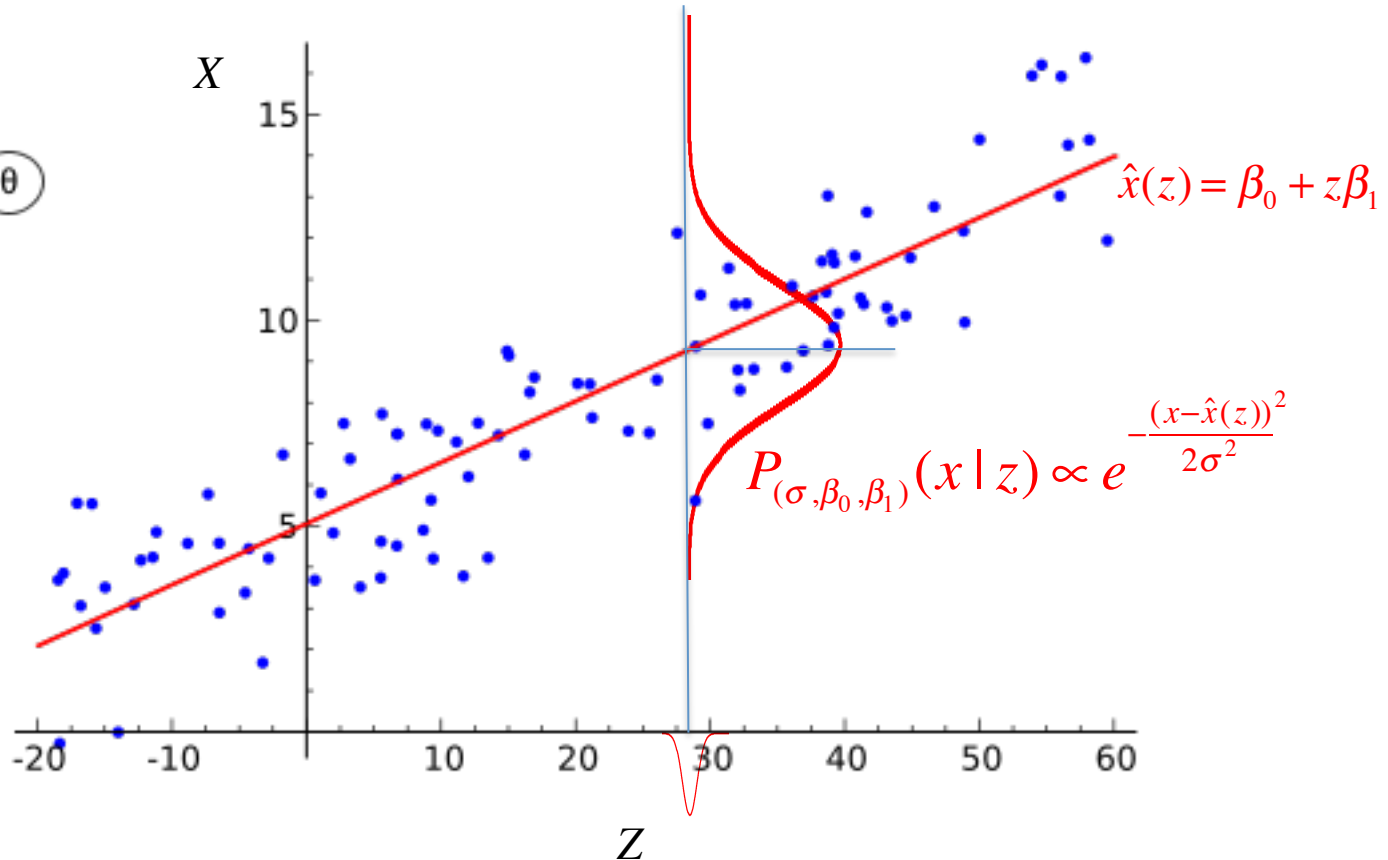
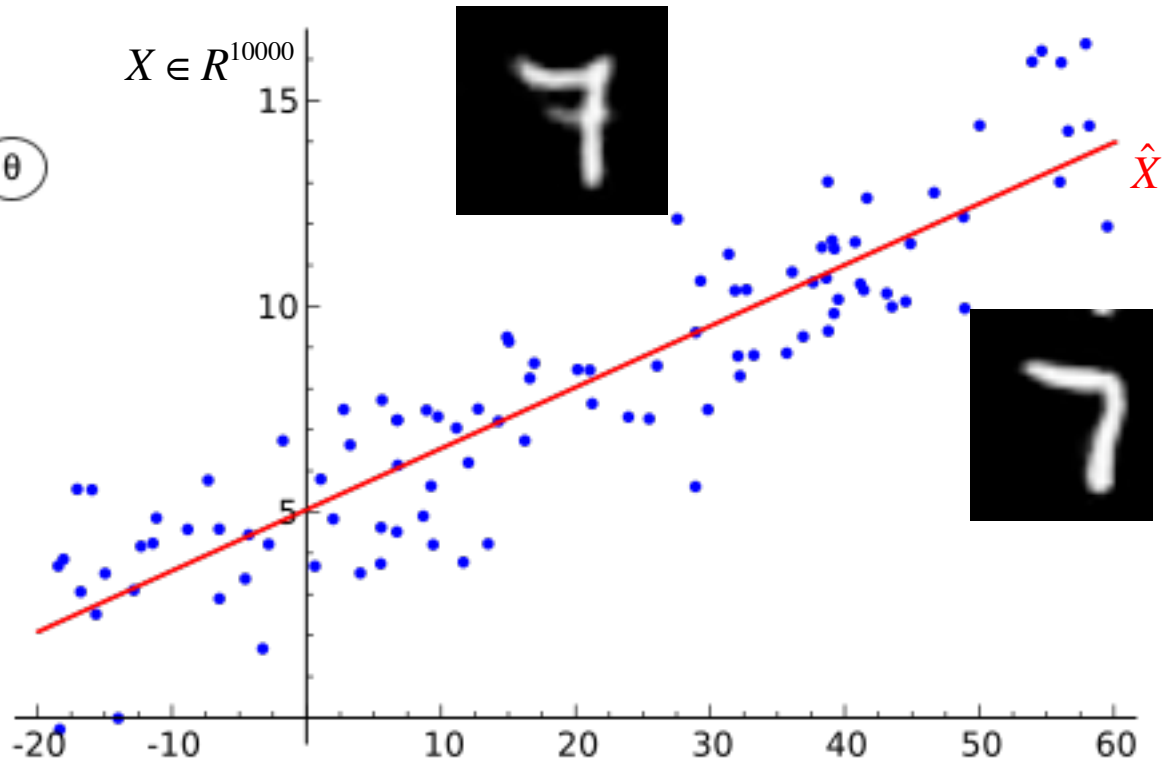
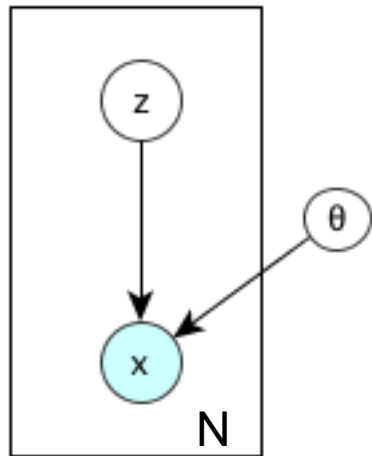


Plate notation
of a graphical
model
to show off



See Beates talk on [Causal inference with graphical models](#)

Going from \mathbb{R}^1 to \mathbb{R}^{10000}

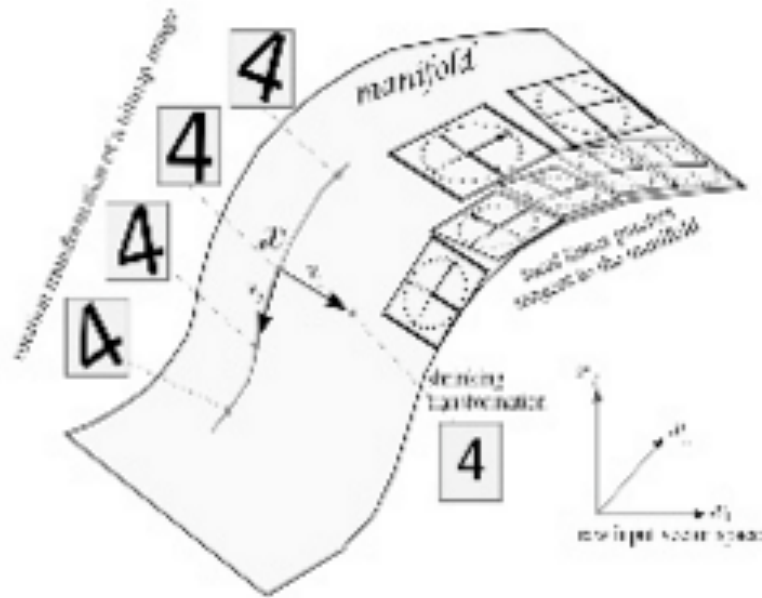


$Z \in \mathbb{R}^n$ typically $n = 2, \dots, 50$ “latent space”

Is \mathbb{R}^2 “big enough” to create images from \mathbb{R}^{100000} ?...

Manifold hypothesis

- X high dimensional vector
- Data is concentrated around a low dimensional manifold



- Hope finding a representation Z of that manifold.

Variational auto encoders (idea of low dim manifold)

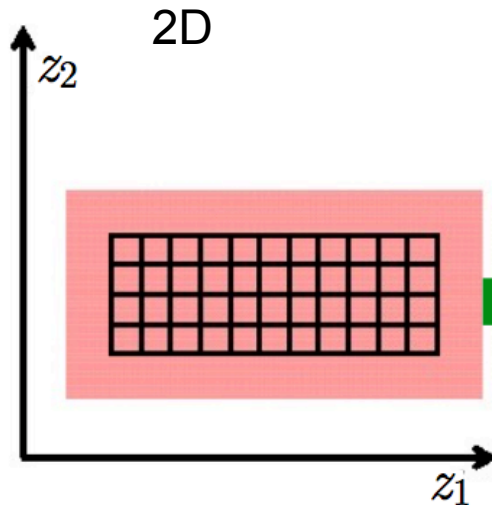
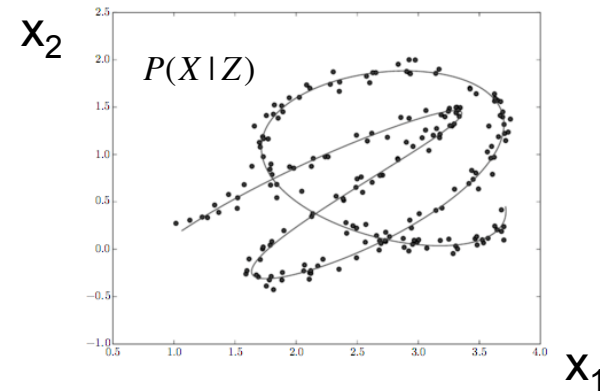
1D

Low Dimensional
representation a line

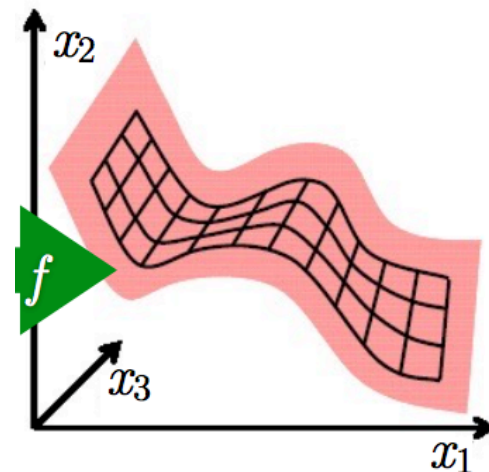


2D

High Dimensional (number of pixels)

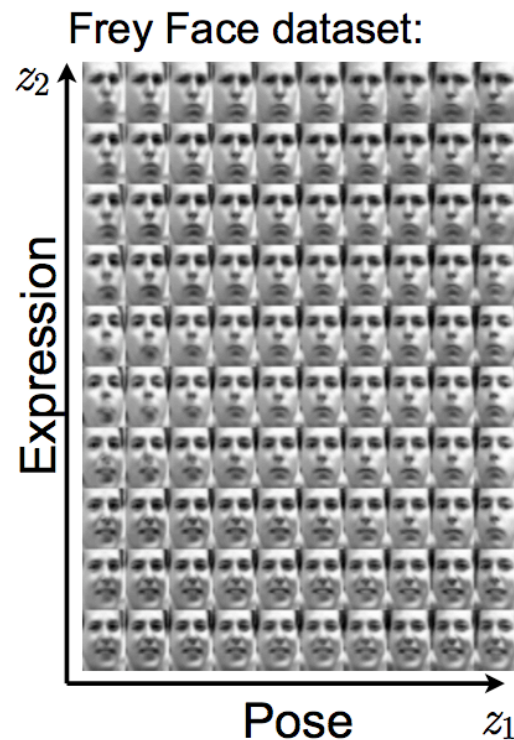
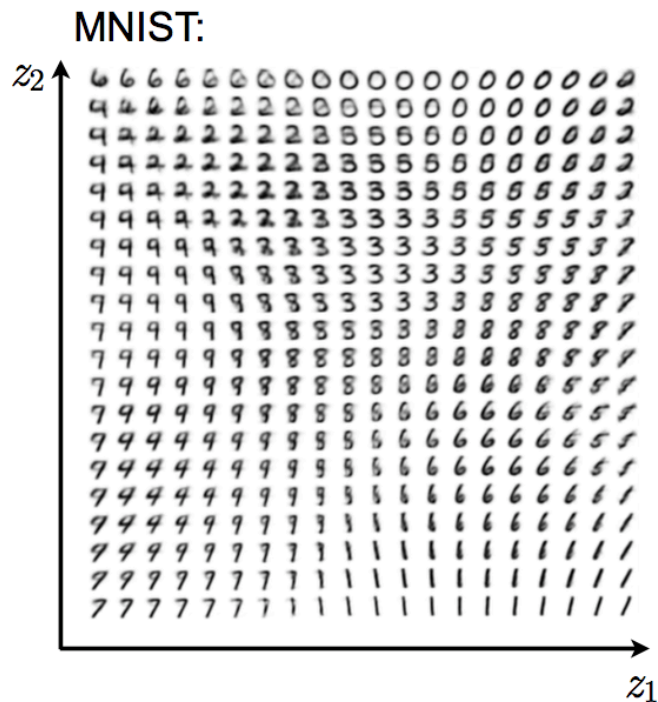


3D



Variational auto encoders (idea of low dim manifold)

Examples of successful unfolding (2D $\rightarrow \mathbb{R}^{28 \times 28}$, $\mathbb{R}^{20 \times 26}$)



[Frey Face dataset](#)

2000 pictures of Brendan Frey (20x26)

How did they do that?

Variational Autoencoders (“history”)

Simultaneously discovered by

- Kingma and Welling. “*Auto-Encoding Variational Bayes*, *International Conference on Learning Representations*.” ICLR, 2014.
[arXiv:1312.6114](#) [stat.ML] (20 December 2013, Amsterdam University)
[Talk](#)
- Rezende, Mohamed and Wierstra. “*Stochastic back-propagation and variational inference in deep latent Gaussian models*.” ICML, 2014
[arXiv:1401.4082](#) [stat.ML] (16 January 2014, Google DeepMind)

Alternative approach (for binary distributions)

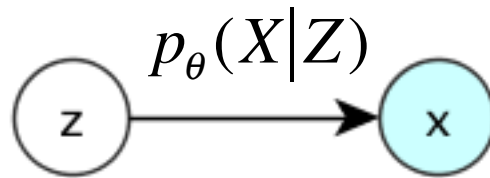
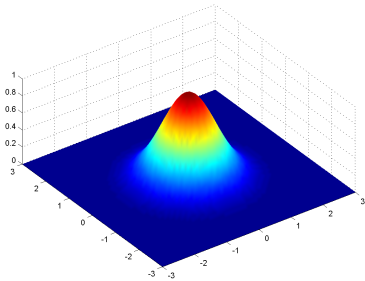
- Gregor, Danihelka et al. “*Deep autoregressive networks*.” ICML 2014
 - Has a more information theoretic ansatz (codings length)
 - Lecture given at [Nando de Freitas ML Course \(University of Oxford\)](#) (a bit hand waving argument but with nice examples)
- We focus on the approach as in Kingma, Welling

Principle Idea encoder network (graphical model)

- We have a set of N-observations (e.g. images) $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$
- Complex model parameterized with θ
- There is a latent space z with

$z \sim p(z)$ multivariate Gaussian

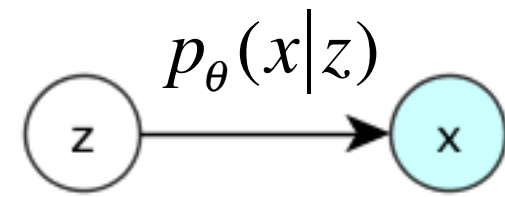
$x|z \sim p_{\theta}(x|z)$



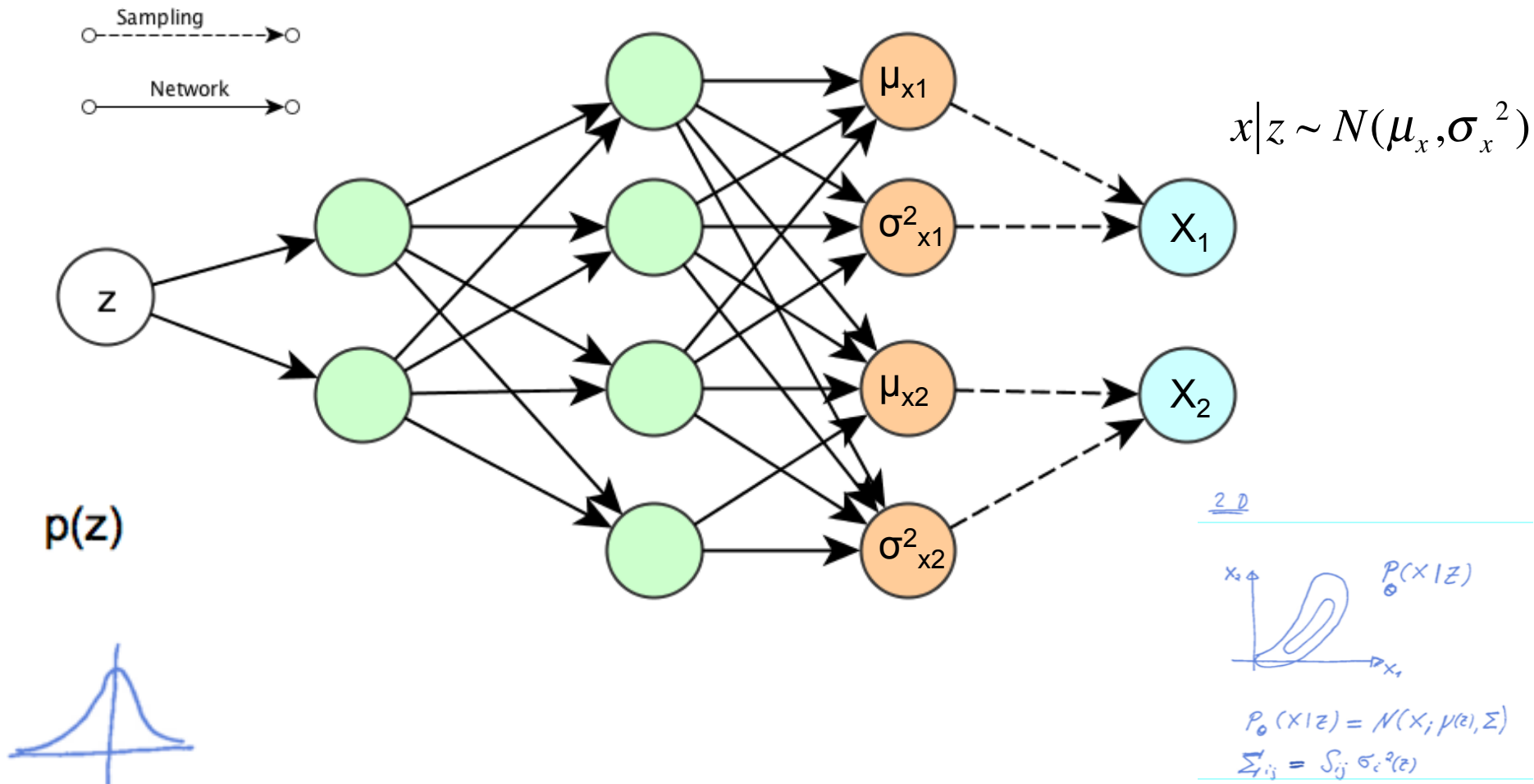
One Example

Wish to learn θ from the N training observations $x^{(i)}$ $i=1, \dots, N$

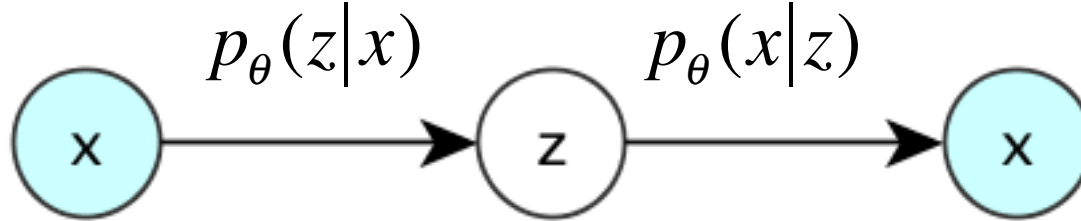
The model for the decoder network



- For illustration z one dimensional x 2D
- Want a complex model of distribution of x given z
- Idea: **NN** + Gaussian (or Bernoulli) here with diagonal covariance Σ



Training as an autoencoder



Training use maximum likelihood
of $p(x)$ given the training data

Problem:

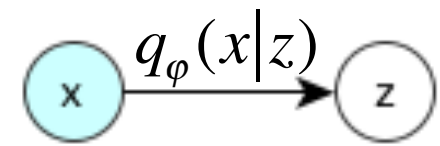
$$p_{\theta}(z|x)$$

Cannot be calculated:

Solution:

- MCMC (too costly)
- Approximate $p(z|x)$ with $q(z|x)$

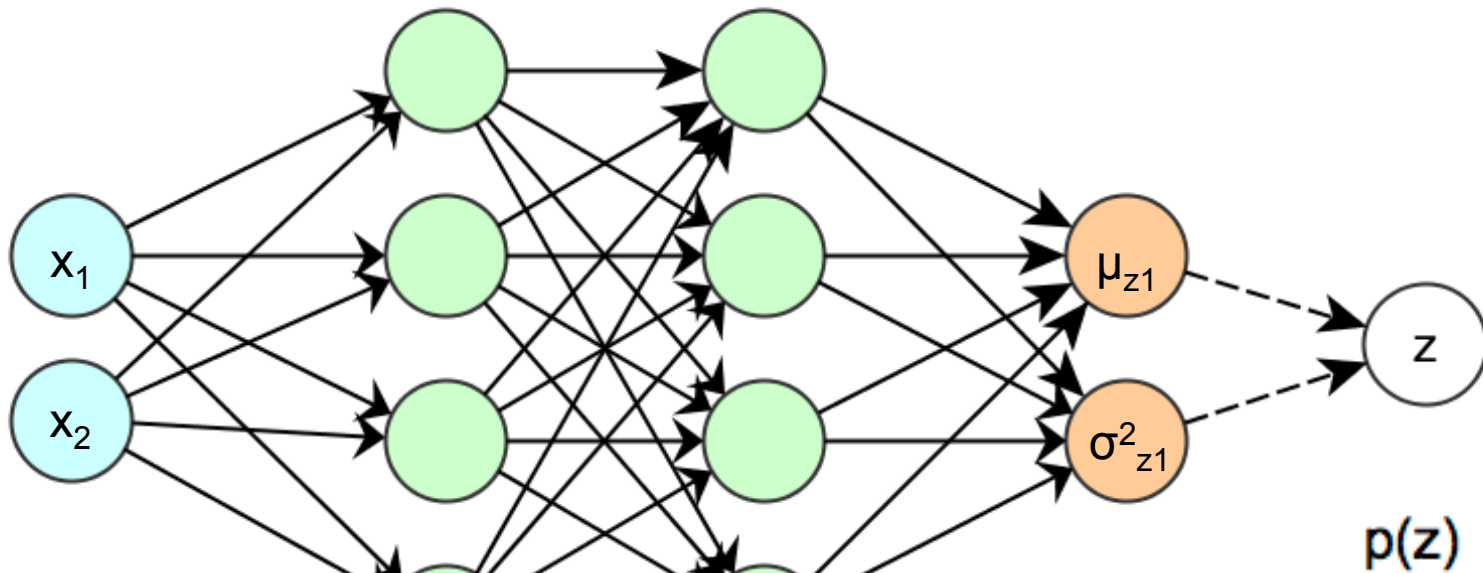
The model for the decoder



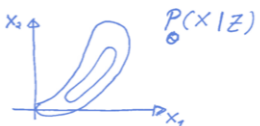
- A feed forward NN + Gaussian

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

Just a Gaussian, with diagonal covariance.



2D

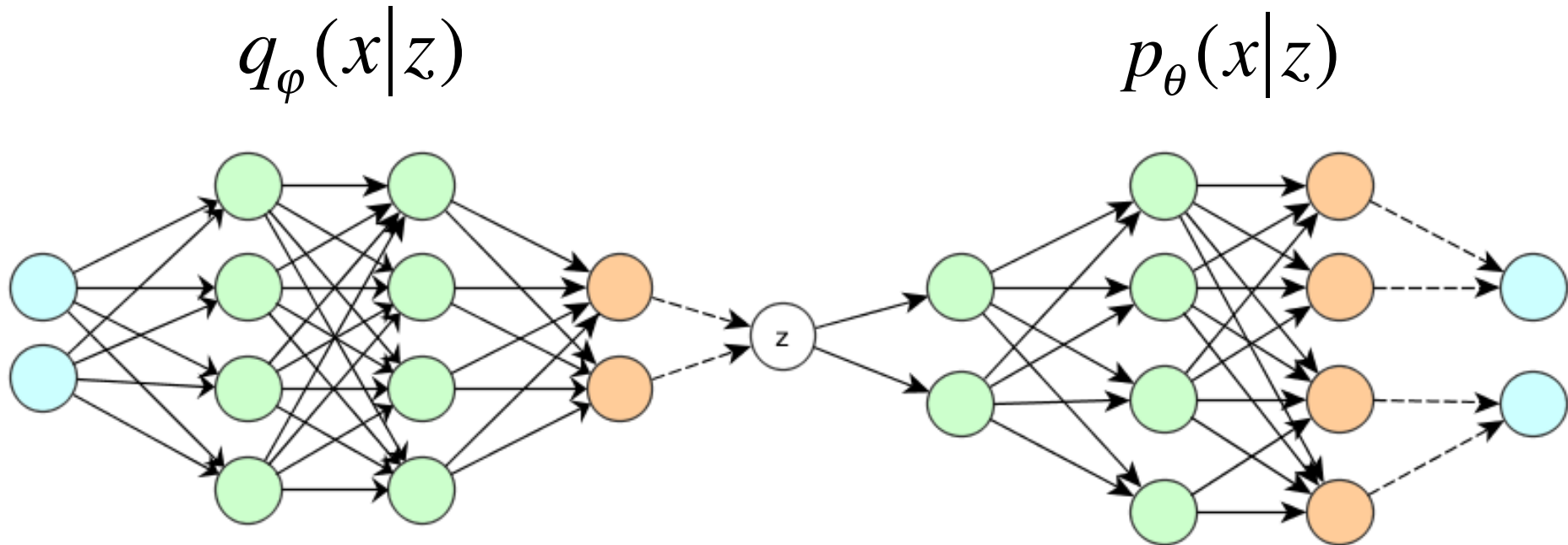


$$p_{\theta}(x|z) = \mathcal{N}(x; \mu(z), \Sigma)$$

$$\Sigma_{i,j} = \delta_{i,j} \sigma_i^2(z)$$



The complete auto-encoder



Learning the parameters ϕ and θ via backpropagation

Determining the loss function

Training: Loss Function

- What is (one of the) most beautiful idea in statistics?
- Max-Likelihood, tune Φ , θ to maximize the likelihood
- We maximize the (log) likelihood of a given “image” $x^{(i)}$ of the training set. Later we sum over all training data (using minibatches)

Lower bound of likelihood (mathematical sleight of hand)

Likelihood, for an image $x^{(i)}$ from training set. Writing $x=x^{(i)}$ for short.

$$\begin{aligned} L &= \log(p(x)) \\ &= \sum_z q(z|x) \log(p(x)) && \text{multiplied with 1} \\ &= \sum_z q(z|x) \log\left(\frac{p(z, x)}{p(z|x)}\right) \\ &= \sum_z q(z|x) \log\left(\frac{p(z, x)}{q(z|x)} \frac{q(z|x)}{p(z|x)}\right) \\ &= \sum_z q(z|x) \log\left(\frac{p(z, x)}{q(z|x)}\right) + \sum_z q(z|x) \log\left(\frac{q(z|x)}{p(z|x)}\right) \\ &= L^v + D_{\text{KL}}(q(z|x) \| p(z|x)) \\ &\geq L^v \end{aligned}$$

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

L^v “lower variational bound of the (log) likelihood” $L^v = L$ for perfect approximation

Approximate Inference (rewriting L^v)

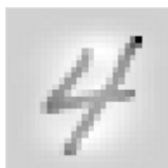
$$\begin{aligned}
 L^v &= \sum_z q(z|x) \log \left(\frac{p(z, x)}{q(z|x)} \right) && \text{with } p(z, x) = p(x|z) p(z) \\
 &= \sum_z q(z|x) \log \left(\frac{p(x|z)p(z)}{q(z|x)} \right) \\
 &= \sum_z q(z|x) \log \left(\frac{p(z)}{q(z|x)} \right) + \sum_z q(z|x) \log (p(x|z)) \\
 &= -D_{\text{KL}}(q(z|x) \| p(z)) + \mathbb{E}_{q(z|x)} (\log (p(x|z))) && \text{putting in } x^{(i)} \text{ for } x \\
 &= -D_{\text{KL}}(q(z|x^{(i)}) \| p(z)) + \mathbb{E}_{q(z|x^{(i)})} (\log (p(x^{(i)}|z)))
 \end{aligned}$$

Regularisation

$p(z)$ is usually a
simple prior $N(0,1)$

Reconstruction quality, $\log(1)$ if $x^{(i)}$ gets always
reconstructed perfectly (z produces $x^{(i)}$)

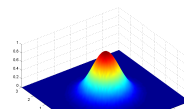
Example $x^{(i)}$



$$q_{\phi}(z|x^{(i)}) \longrightarrow$$



$$p_{\theta}(x^{(i)}|z) \longrightarrow$$

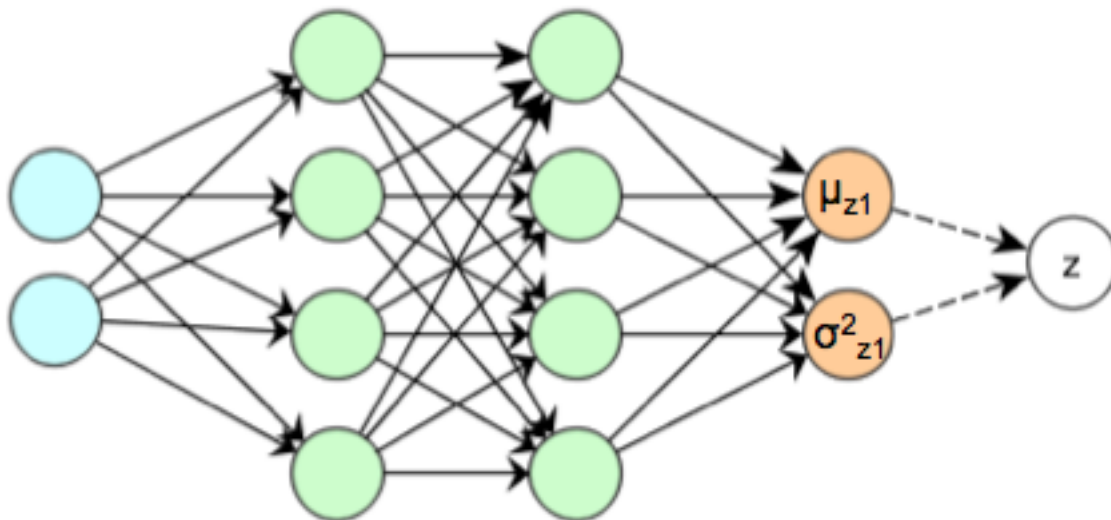


Calculation the regularization $-D_{\text{KL}}(q(z|x^{(i)})||p(z))$

Use $N(0,1)$ as prior for $p(z)$

$q(z|x^{(i)})$ is Gaussian with parameters $(\mu^{(i)}, \sigma^{(i)})$ determined by NN

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



Sampling to calculate $\mathbb{E}_{q(z|x^{(i)})} (\log(p(x^{(i)}|z)))$

Approximating $\mathbb{E}_{q(z|x^{(i)})}$ with sampling from the distribution $q(z|x^{(i)})$

With $z^{(i,l)}$ $l = 1, 2, \dots, L$ sampled from $z^{(i,l)} \sim q(z|x^{(i)})$

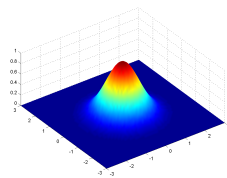
$$L^v = -D_{\text{KL}}(q(z|x^{(i)}) \| p(z)) + \mathbb{E}_{q(z|x^{(i)})} (\log(p(x^{(i)}|z)))$$

$$L^v \approx -D_{\text{KL}}(q(z|x^{(i)}) \| p(z)) + \frac{1}{L} \sum_{i=1}^L \log(p(x^{(i)}|z^{(i,l)}))$$

Example $x^{(i)}$



$q_\phi(z|x^{(i)})$



$\log(p_\theta(x^{(i)}|z^{(i,1)}))$ where $z^{(i,1)} \sim N(\mu_z^{(i)}, \sigma_z^{2(i)})$

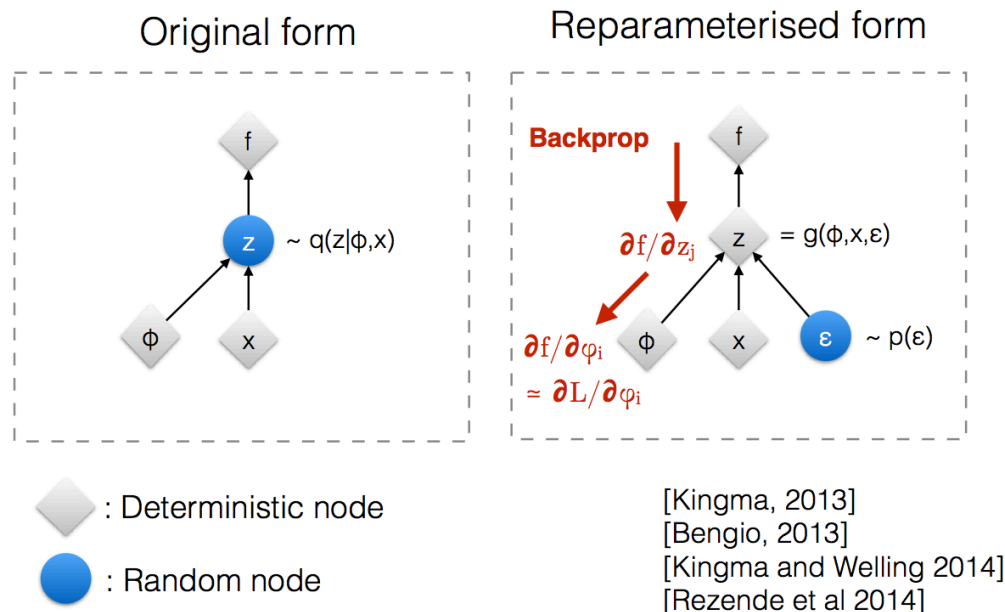
...

$\log(p_\theta(x^{(i)}|z^{(i,L)}))$ where $z^{(i,L)} \sim N(\mu_z^{(i)}, \sigma_z^{2(i)})$

L is often very small (often just $L=1$)

One last trick

Backpropagation not possible through random sampling!



Sampling (reparametrization trick)

$$z^{(i,l)} \sim N(\mu^{(i)}, \sigma^{2(i)})$$

$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon_i \quad \epsilon_i \sim N(0,1)$$

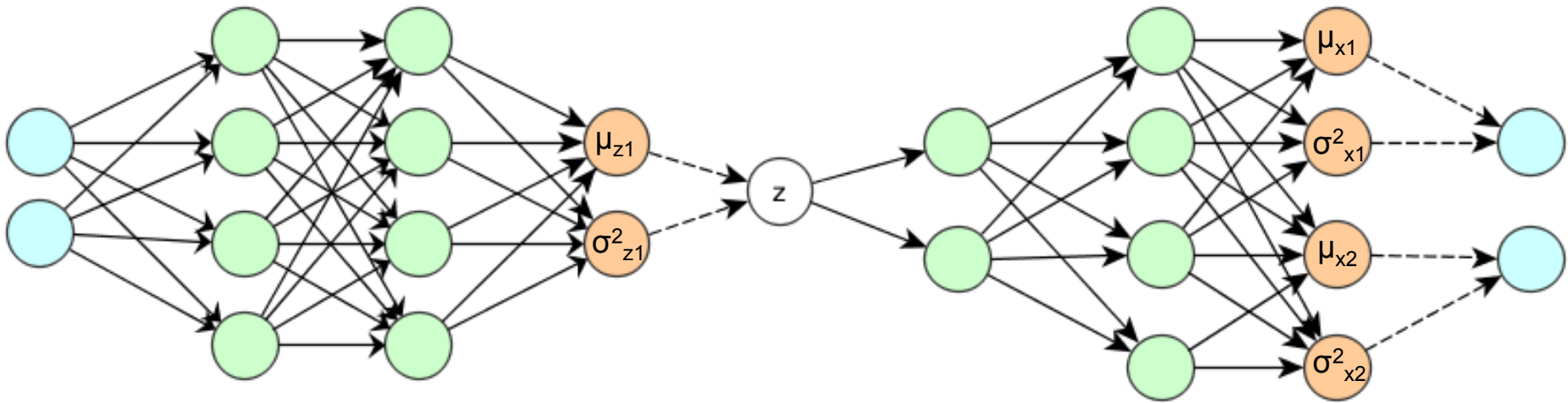
Cannot back propagate through a random drawn number

z has the same distribution, but now one can back propagate.

Writing z in this form, results in a deterministic part and noise.

Putting it all together

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



Cost: Regularisation

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

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

Cost: Reproduction

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

Least Square for constant variance

Use the source Luke

Simple example 2-D distribution

https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/vae/vae_demo-2D.ipynb

Simple MNIST Example

https://github.com/oduerr/dl_tutorial/blob/master/tensorflow/vae/vae_demo.ipynb

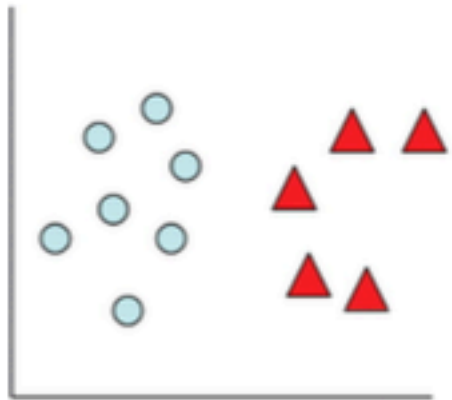
Recent developments of VAE

Recent developments in VAE / generative models

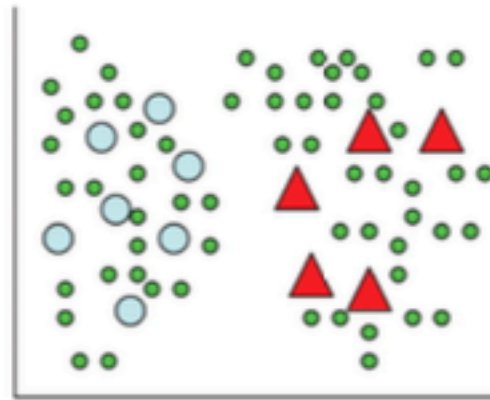
(subjective overview)

- Authors of VAE Amsterdam University and Google DeepMind teamed up and wrote a paper on semi-supervised learning:
 - Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, Max Welling.
["Semi-supervised learning with deep generative models"](#) (2014)
- Karl Gregor et al. extended the (binary autoencoder) with attention
 - *DRAW: A Recurrent Neural Network For Image Generation*
<https://arxiv.org/abs/1502.04623> (2015)
 - <https://www.youtube.com/watch?v=Zt-7MI9eKEo>
- Adversarial networks as a non-statistical way to generate high dimensional data
 - Play a game:
 - First network invents some data $\rightarrow P(X)$ to fool second network
 - Second network tells if first network is a liar.

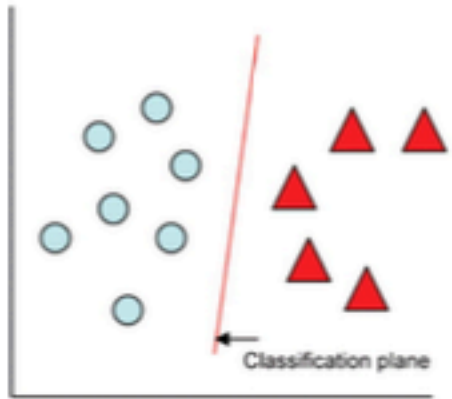
Semisupervised learning



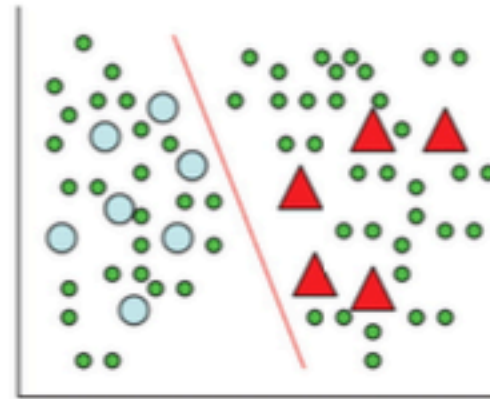
Labeled Data
(a)



Labeled and Unlabeled Data
(b)



Supervised Learning
(c)



Semi-Supervised Learning
(d)

Semisupervised learning

VAEs are SOTA
on semi-supervised learning on MNIST

That's 10
per class!

	100 labels
AtlasRBF (Pitelis et al., 2014)	8.10% (± 0.95)
Deep Generative Model (M1+M2) (Kingma et al., 2014)	3.33% (± 0.14)
Virtual Adversarial (Miyato et al., 2015)	2.12%
Ladder (Rasmus et al., 2015)	1.06% (± 0.37)
Auxiliary Deep Generative Model (1 MC)	2.25% (± 0.08)
Auxiliary Deep Generative Model (10 MC)	0.96% (± 0.02)



“Improving Semi-Supervised Learning with Auxiliary
Deep Generative Models”

[Maaløe, Sønderby, Sønderby and Winter, 2015]

Thank you, questions?