Lecture 11. Autoencoders, Deep generative models

COMP90051 Statistical Machine Learning

Semester 1, 2021
Trevor Cohn

THE UNIVERSITY OF MELBOURNE

This lecture

- Autoencoders
  - Learning efficient coding
  - Methods for autoencoding: sparse, denoising, contractive
  - Use in pre-training pipelines
- Deep generative models
  - Variational autoencoder (VAE)

Autoencoder

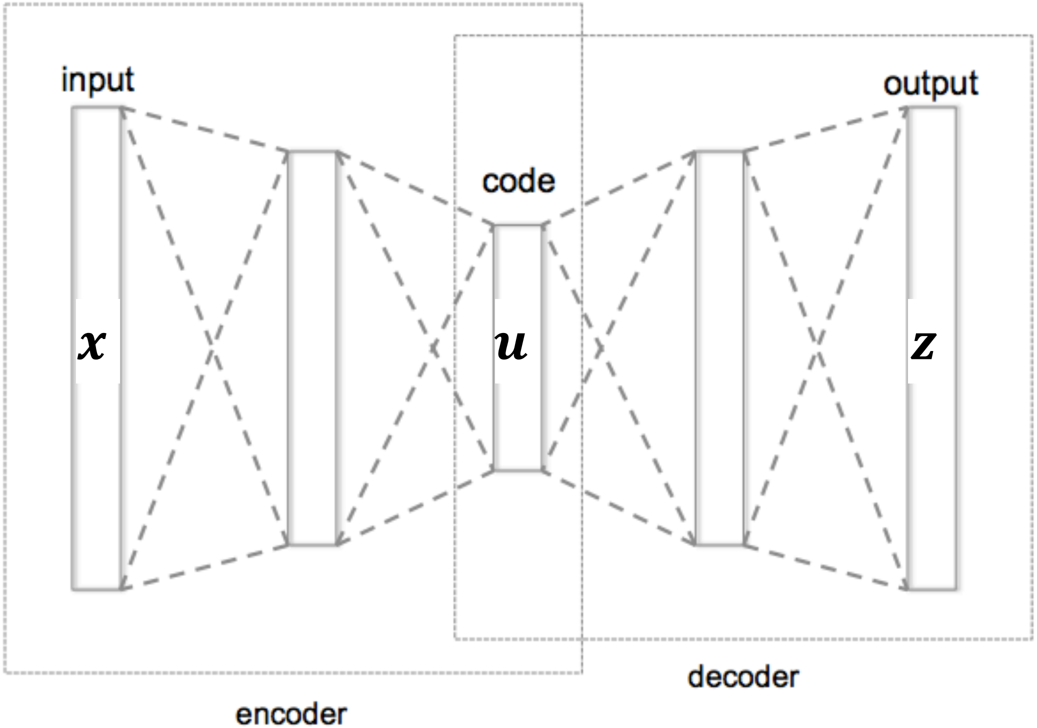
An ANN training setup that can be used for unsupervised learning, initialisation, or just efficient coding

Autoencoding idea

- Supervised learning:
  - Univariate regression: predict y from $\boldsymbol{x}$
  - Multivariate regression: predict $\boldsymbol{y}$ from $\boldsymbol{x}$
- Unsupervised learning: explore data $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$
  - No response variable
- For each $\boldsymbol{x}_i$ set $\boldsymbol{y}_i \equiv \boldsymbol{x}_i$
- Train an ANN to predict $\boldsymbol{y}_i$ from $\boldsymbol{x}_i$ i.e., model $\mathrm{p}(\boldsymbol{x}|\boldsymbol{x})$
- Pointless?

Autoencoder topology

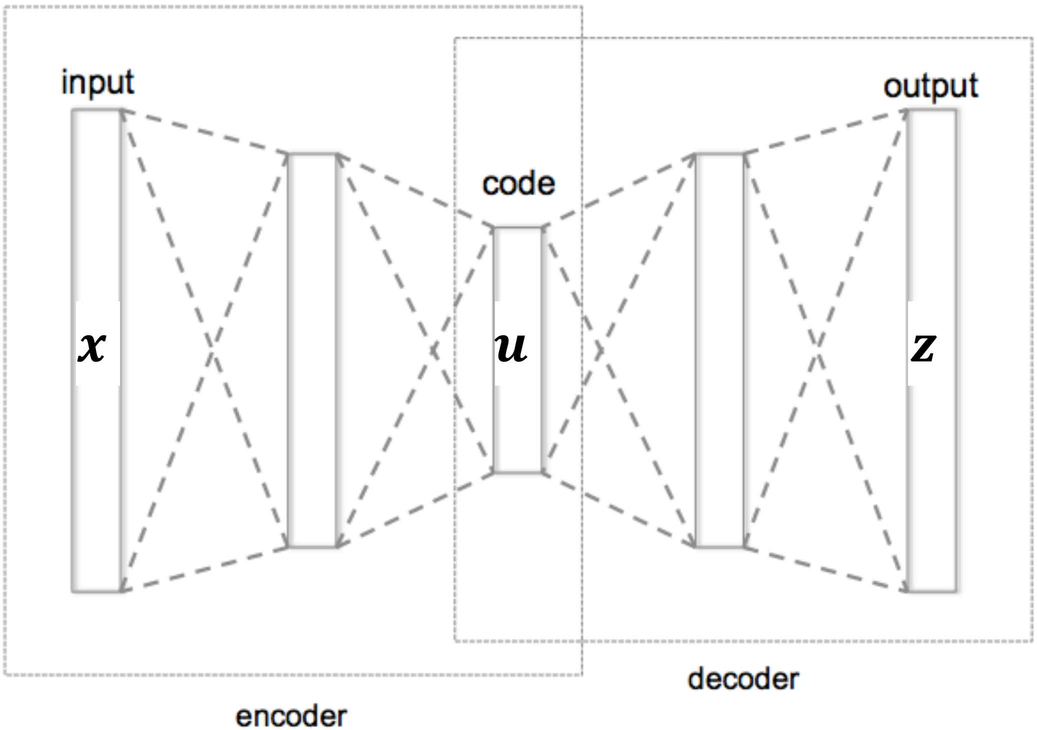
- Given data without labels $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, set $\boldsymbol{y}_i \equiv \boldsymbol{x}_i$ and train an ANN to predict $\boldsymbol{z}(\boldsymbol{x}_i) \approx \boldsymbol{x}_i$
- Set bottleneck layer $\boldsymbol{u}$ in middle "thinner" than input



adapted from: Chervinskii at Wikimedia Commons (CC4)

Introducing the bottleneck

- Suppose you managed to train a network that gives a good restoration of the original signal $\boldsymbol{z}(\boldsymbol{x}_i) \approx \boldsymbol{x}_i$
- This means that the data structure can be effectively described (encoded) by a lower dimensional representation $\boldsymbol{u}$



adapted from: Chervinskii at Wikimedia Commons (CC4)

Under-/Over-completeness

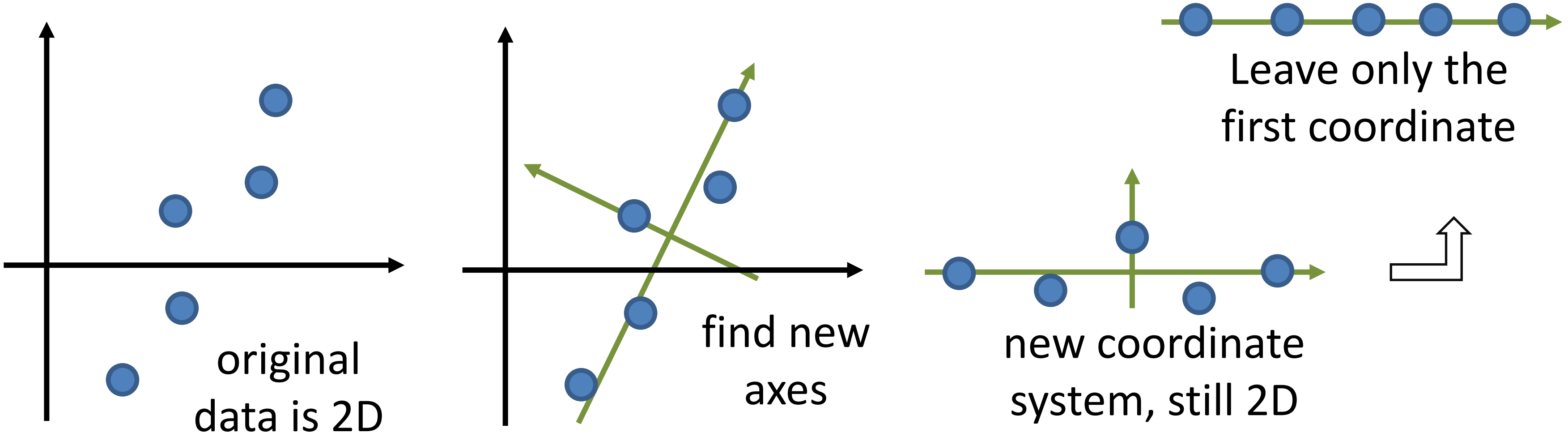
- Manner of bottleneck gives rise to:
  - undercomplete: model with thinner bottleneck than input forced to generalise
  - overcomplete: wider bottleneck than input, can just "copy" input directly to output
- Even undercomplete models can learn trivial codes, given complex non-linear encoder and decoder
- Various methods to ensure learning

Dimensionality reduction

- Autoencoders can be used for compression and dimensionality reduction via a non-linear transformation
- Closely related to principal component analysis (PCA)…

Principal component analysis

- Principal component analysis (PCA) is a popular method for dimensionality reduction and data analysis in general
- Given a dataset $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, $\boldsymbol{x}_i \in \boldsymbol{R}^m$, PCA aims to find a new coordinate system such that most of the variance is concentrated along the first coordinate, then most of the remaining variance along the second coordinate, etc.
- Dimensionality reduction is then based on discarding coordinates except the first $l < m$



PCA: Solving the optimisation

- PCA aims to find $\boldsymbol{p}_1$ that maximises $\boldsymbol{p}_1'\boldsymbol{\Sigma}_X\boldsymbol{p}_1$, subject to $\|\boldsymbol{p}_1\| = \boldsymbol{p}_1'\boldsymbol{p}_1 = 1$

- Constrained → Lagrange mulitipliers. Introduce multiplier λ_1; set derivatives of Lagrangian to zero, solve

- $L = \boldsymbol{p}_1'\boldsymbol{\Sigma}_X\boldsymbol{p}_1 - \lambda_1(\boldsymbol{p}_1'\boldsymbol{p}_1 - 1)$
- $\dfrac{\partial L}{\partial \boldsymbol{p}_1} = 2\boldsymbol{\Sigma}_X\boldsymbol{p}_1 - 2\lambda_1\boldsymbol{p}_1 = 0$
- $\boldsymbol{\Sigma}_X\boldsymbol{p}_1 = \lambda_1\boldsymbol{p}_1$

- Precisely defines $\boldsymbol{p}_1$ as an eigenvector with λ_1 being the corresponding eigenvalue

PCA vs Autoencoding

- If you use linear activation functions and only one hidden layer, then the setup becomes almost that of Principal Component Analysis (PCA)
  - PCA finds orthonormal basis where axes are aligned to capture maximum data variation
  - ANN might find a different solution, doesn't use eigenvalues (directly)

Mini-summary

- Autoencoding as modelling $\mathrm{p}(\boldsymbol{x}|\boldsymbol{x})$
- Notion of under- and over-completeness
- Relationship to PCA

Autoencoding techniques

Methods for learning Autoencoders

- Sparse autoencoders
  - includes regularisation over the code layer
- Denoising autoencoders
  - learn to recover true data given noise-corrupted input
- Contractive autoencoders
  - explicit penalty term to ensure robustness to local changes in input
- In all cases, manage the tension between compact code and accurate reconstruction

Sparse Autoencoders

- Apply constraint to the hidden code
  - Use regularisation penalty in training objective, e.g.,

$$\lambda \sum_i |u_i|$$

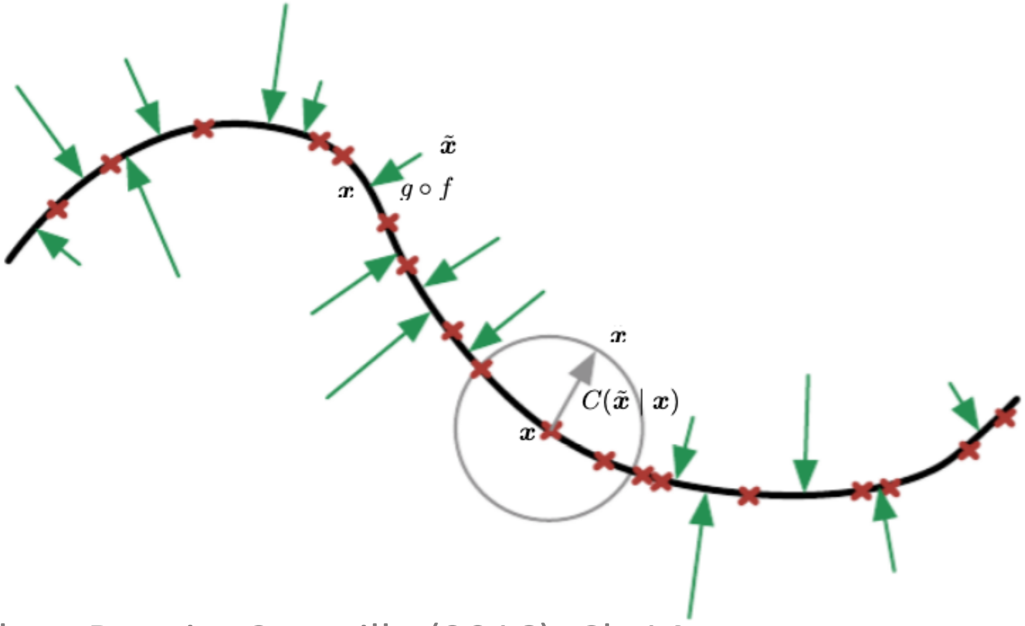
  - May need to use with over-complete hidden layer, and multi-layer encoder/decoder
  - Sparsity can simplify data analysis
- Can be thought of as a kind of as prior

Denoising Autoencoder

- Corrupt input data to the autoencoder, but applying noise (masking, additive whitenoise, etc)
  - Now modelling $\mathrm{p}(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ where $\tilde{\boldsymbol{x}}$ is noised input
  - The model must recover corrupted parts of the input, thus must learn underlying structure of the data
- Popular "BERT" model for NLP uses this idea to learn word and sentence representations
  - Based on masking of words in a transformer neural network over strings

AE and Manifold Learning

- Most of the input space does not correspond to input data, data lives on lower dim “manifold”
  - DAE learns training data manifold, by mapping nearby points onto the manifold



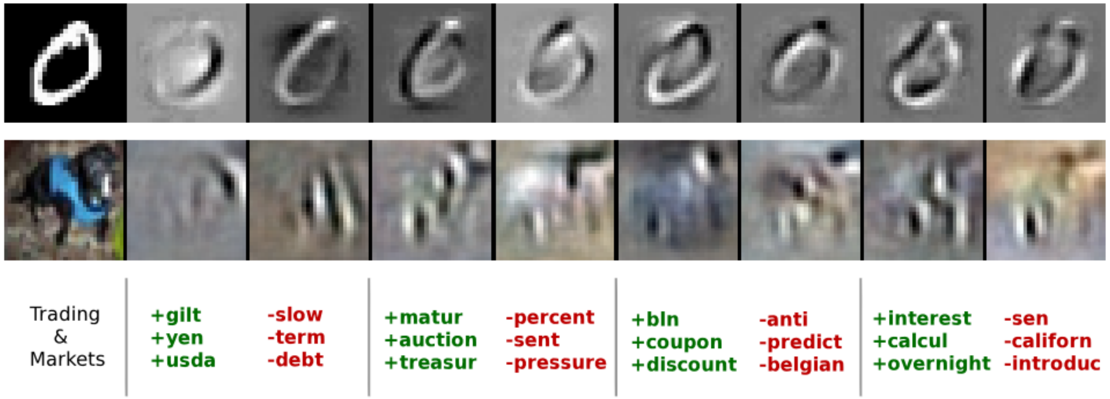
Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 14

Contractive Autoencoder

- Include penalty term over derivatives wrt input

$$\lambda \sum_i \left|\left|\nabla_x u_i\right|\right|^2$$

  - local changes to input $\boldsymbol{x}$ have limited effect on code $\boldsymbol{u}$
  - tangent vectors to the manifold encode data variation



Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011c). The manifold tangent classifier. In NIPS'2011 .

Uses of Autoencoders

- Data visualisation & clustering
  - Unsupervised first step towards understanding properties of the data
- As a feature representation
  - Allowing the use of off-the-shelf ML methods, applied to much smaller and informative representations of input
- Pre-training of deep models
  - Warm-starting training by initialising model weights with encoder parameters
  - In some fields like vision, mostly replaced with supervised pre-training on very large datasets

Mini-summary

- Autoencoding methods
  - Sparse
  - Denoising
  - Contractive
- Relation to manifold learning
- Uses of autoencoders

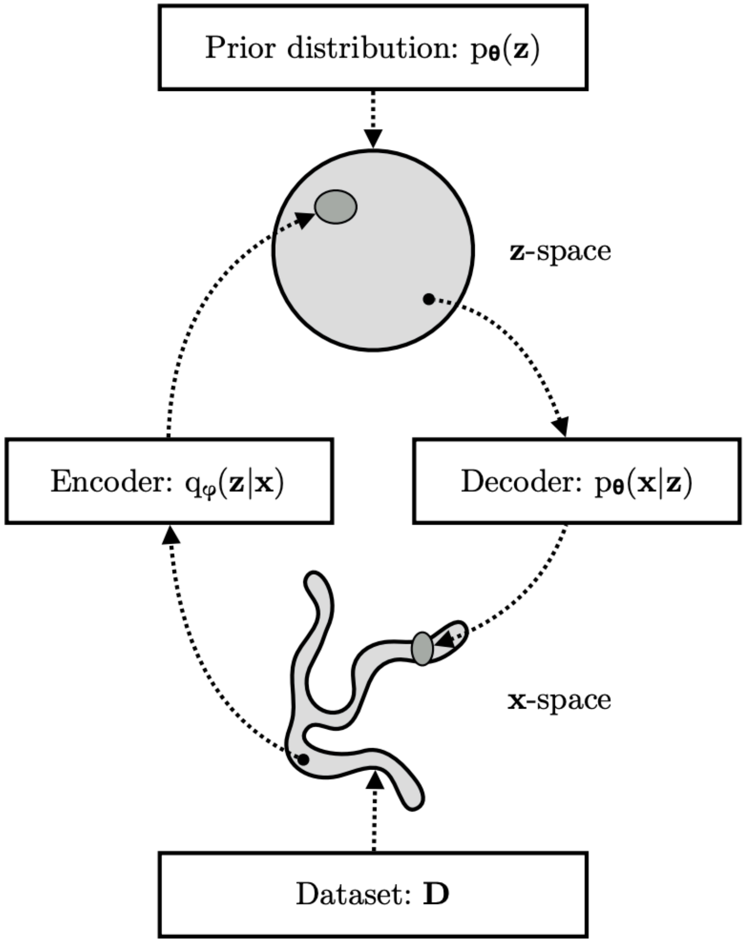
Variational “autoencoder”

A form of deep generative model
whose inference method resembles
that of an autoencoder

Deep generative models

- Often need a model which can *generate inputs*
  - trained to model $\mathrm{p}(\boldsymbol{x})$ directly. I.e., unsupervised model trained as density estimator
  - sometimes we also model a label, $\mathrm{p}(\boldsymbol{x}, y)$ — we will return to this in a later lecture
- Use a probabilistic model with a latent random variable $\boldsymbol{z}$ to explain data variation
  - $p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z})$ (*sum denotes integral or summation*)
  - latent variable to capture structure in data, such as clusters, style, etc
  - framework includes several models, e.g., RBMs, DBMs etc

Structure of Variational Autoencoder



Kingma and Welling, An Introduction to Variational Autoencoders, FTML (2019)
https://arxiv.org/pdf/1906.02691.pdf

Variational Autoencoder

- Given generative model, $p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z})$, how to fit to data?
  - $\log p(\boldsymbol{x})$

assume model factorises

introduce q distribution (chosen carefully)

$$= \log \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z})$$

$$= \log \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) \frac{q(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z}|\boldsymbol{x})}$$

$$= \log E_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})} \left[p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) \frac{1}{q(\boldsymbol{z}|\boldsymbol{x})} \right]$$

$$\geq E_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})} \left[\log p(\boldsymbol{x}|\boldsymbol{z}) \frac{p(\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})} \right]$$

use Jensen's inequality to move log inside expectation

  - This is known as the "evidence lower bound" or ELBO. We seek to maximise this quantity.

VAE (cont.)

- Have to choose distributions, most commonly:
  - prior $p(\boldsymbol{z})$ a unit Gaussian, $N(0,1)$
  - $q(\boldsymbol{z}|\boldsymbol{x})$ also a Gaussian, $N(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$ as neural network with vector outputs, and parameters ϕ
  - likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ another neural network, parameters θ
- Optimisation problem is now

$$\underset{\theta,\phi}{\arg\max}\, E_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z}) + \log\frac{p(\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right]$$

reconstruction term

Kullbach-Leibler divergence
$-D_{\mathrm{KL}}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$

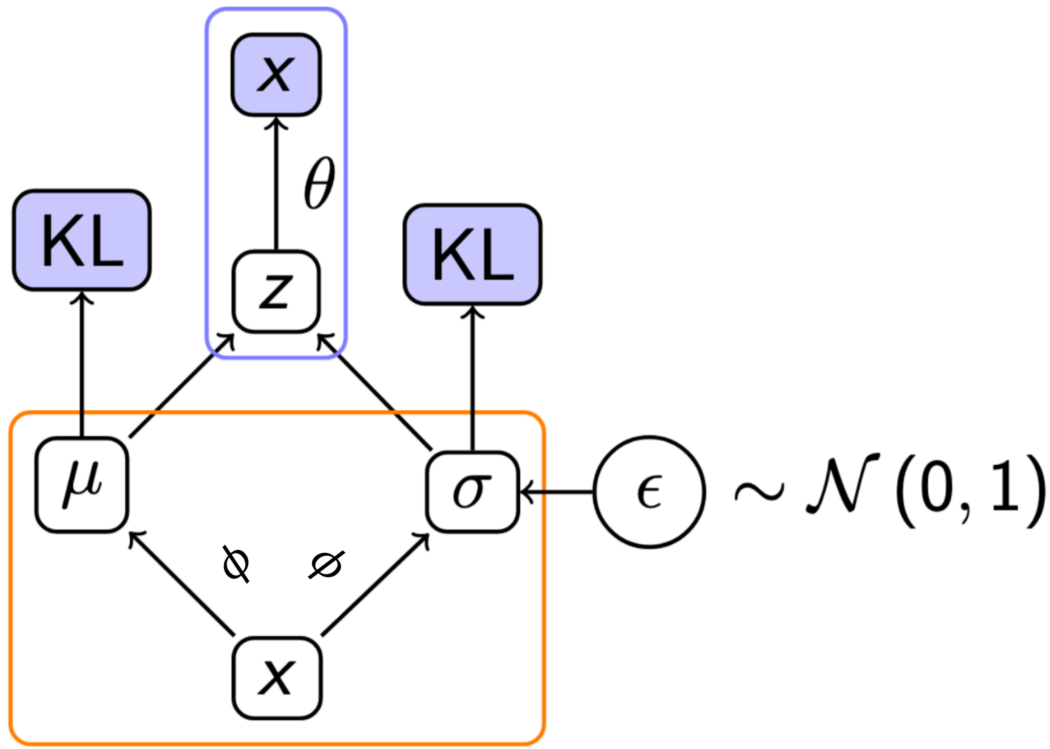
Optimising the VAE

- Intractable objective due to expectation
  - approximate expectation with single sample $\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}|\boldsymbol{x})$
  - but sampling is not a differentiable operator, can't backpropagate gradients (to learn ϕ)
- Reparameterisation trick
  - sample instead $\epsilon \sim N(0,1)$, then rescale and offset,
    $$\hat{\boldsymbol{z}} = \mu(\boldsymbol{x}) + \hat{\epsilon} \odot \sigma^2(\boldsymbol{x})$$
  - back-propagation now side-steps sampling
  - N.b., trick only works for a limited set of distributions

VAE as a Computation Graph

generation model
a.k.a., "decoder"

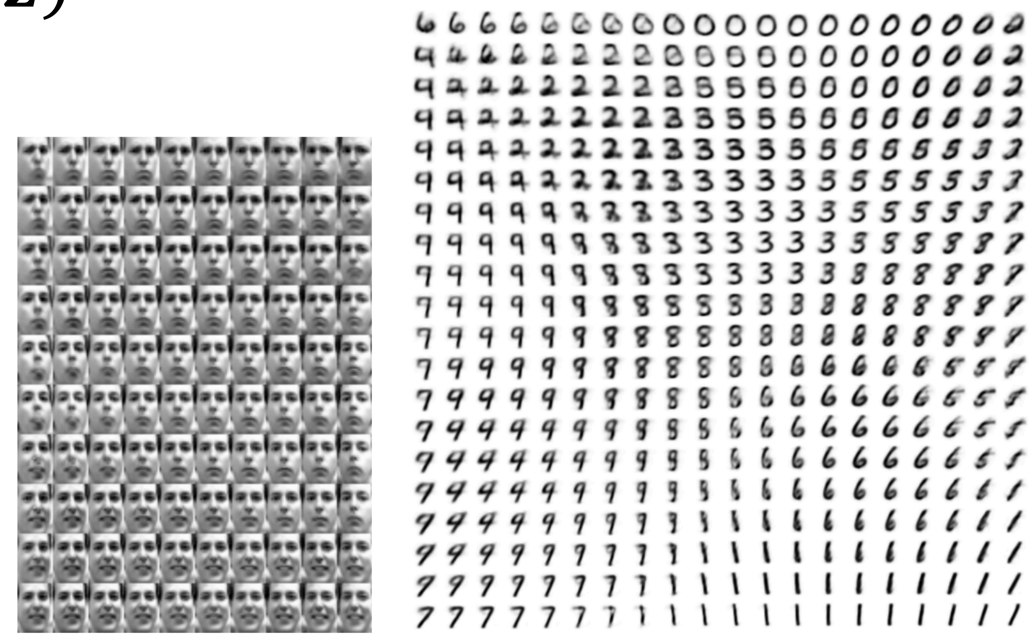
inference model
a.k.a., "encoder"



VI Tutorial, Wilker Aziz et al,
https://github.com/vitutorial/tour/blob/UA2020/modules/DGMs-Continuous/master.pdf

Generating from a VAE

- Simply sample from prior, $\hat{\boldsymbol{z}} \sim p(\boldsymbol{z})$, then sample from decoder, $\widehat{\boldsymbol{x}} \sim p(\boldsymbol{x}|\hat{\boldsymbol{z}})$
- Examples



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *ICLR 204*

This lecture

- Autoencoders
  - Learning efficient coding
  - Methods for autoencoding
  - Their uses
- Variational autoencoders
- Workshops Week #6: Convolution & Autoencoding
- Next lectures: Recurrent neural networks