R250 Advanced topics in machine learning and natural language processing

Topic: autoencoders and generative models

Damon Wischik

Arrangements

- 1 introductory lecture
- 2 sessions of paper presentations
  6 papers to present, 15-20 min each

You should *all* read the papers and participate in the discussion

Assessment:

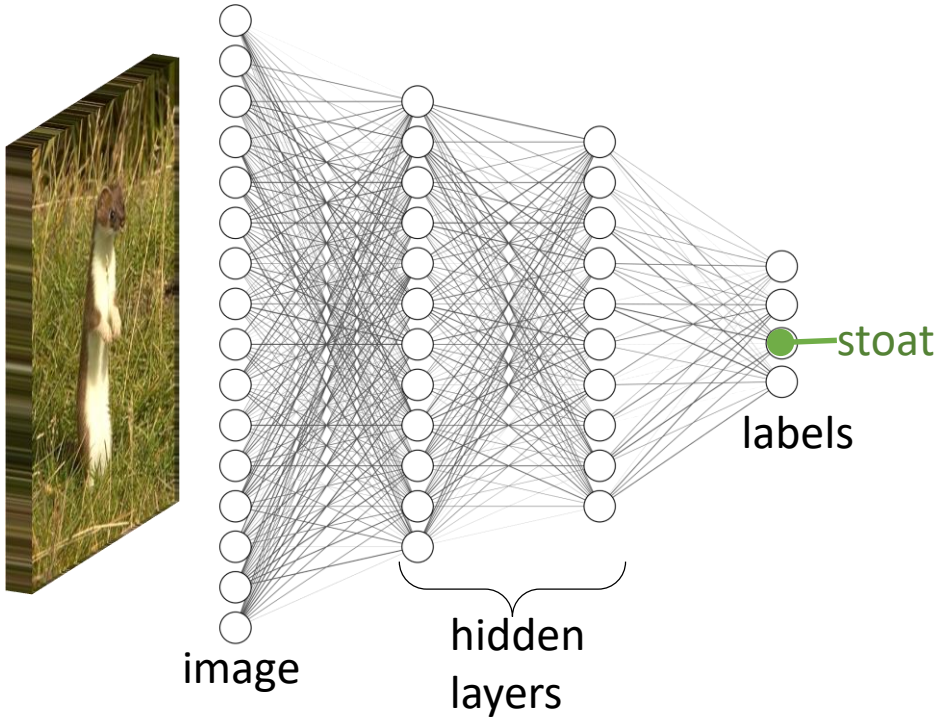
- Paper presentation (5%)
- Attendance and contribution to discussion (5%)
- Project report or essay on one of the topics (90%)

What is an autoencoder?

A classifier

Input: labelled data $(X_n, Y_n)_{n=1..N}$

Task: predict the output Y given input X

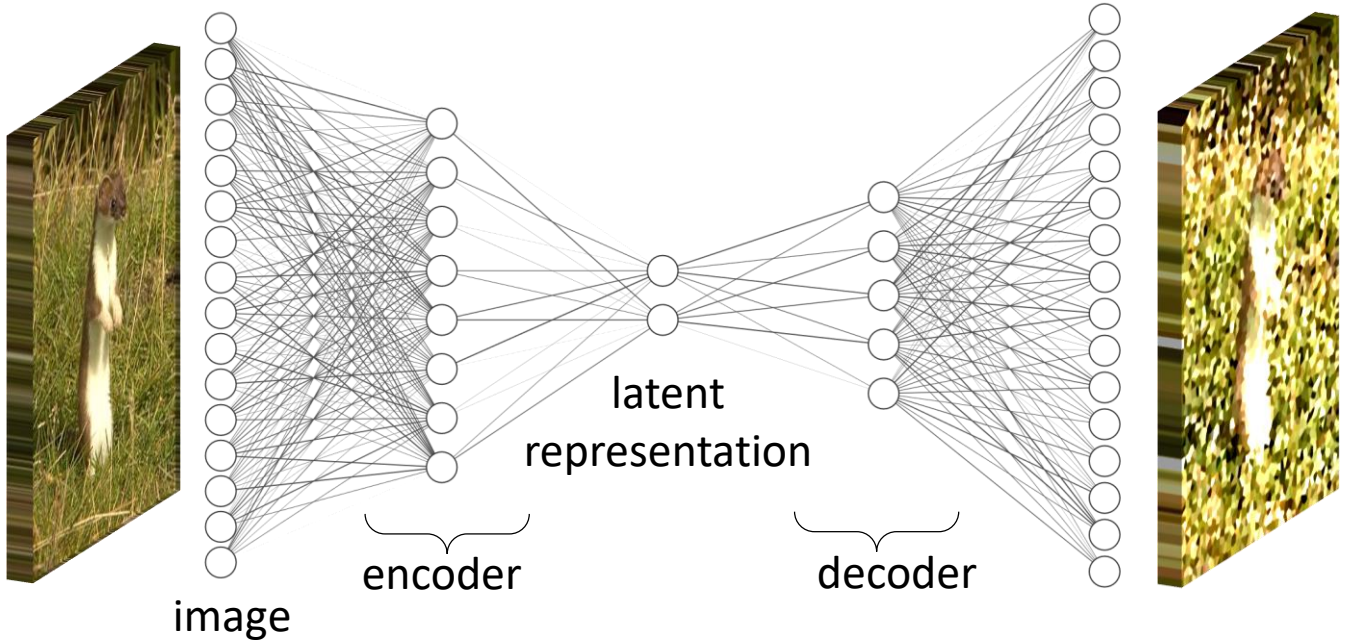


An autoencoder

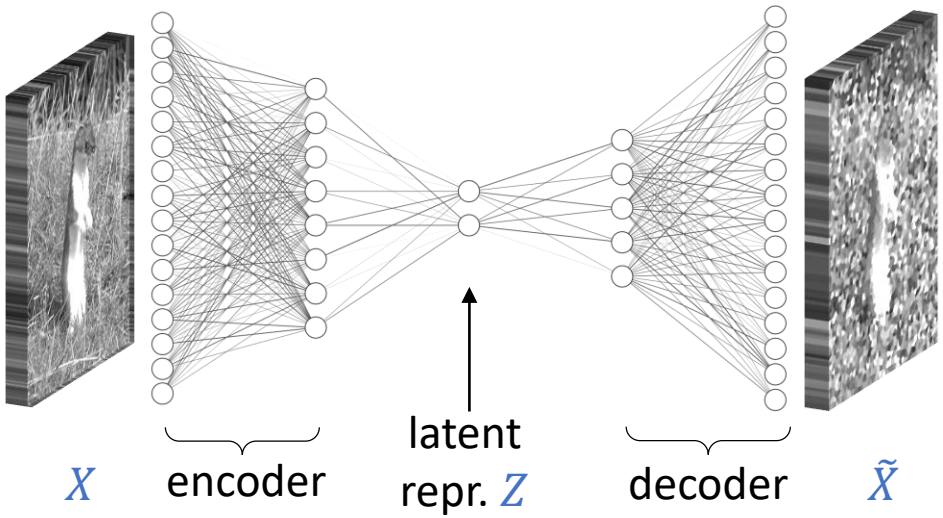
Input: unlabelled data $(X_n)_{n=1..N}$

Task: given an input, reconstruct it

Challenge: squeeze the data through a “bottleneck”



What's the point in learning to recreate the input?



1. Autoencoders are like PCA, but fancier.

   The latent representation $Z = \text{enc}(X)$ is a dimension-reduced version of the input. Data scientists love dimension reduction.

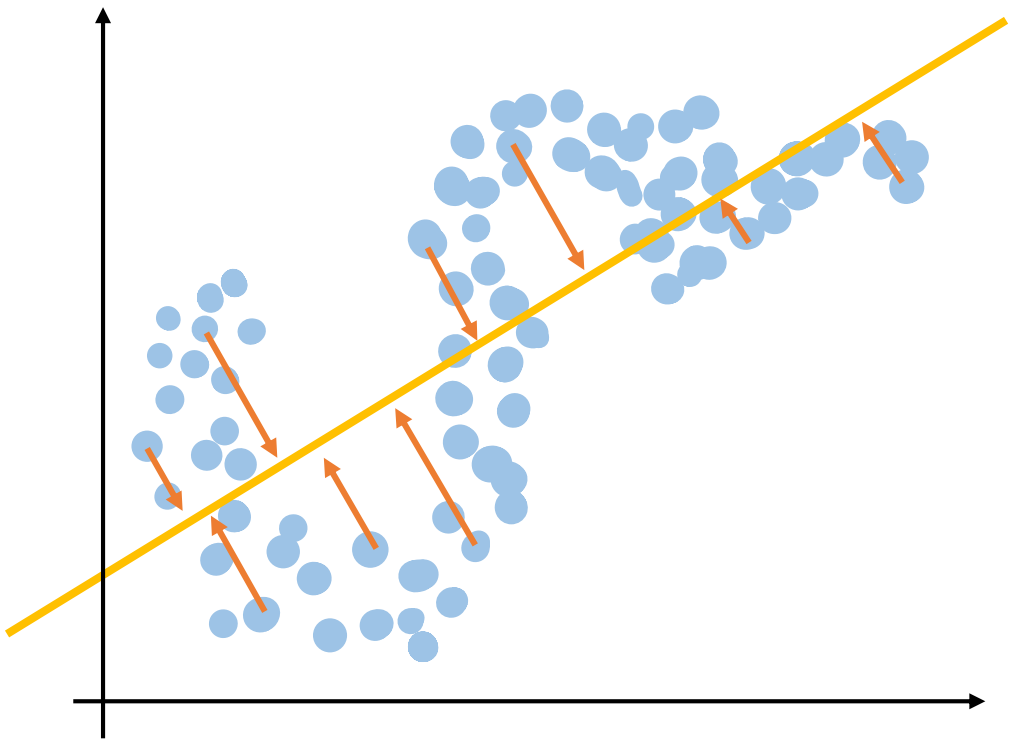
2. Autoencoders learn a useful representation of the data.

   It stands to reason that, in order to *reproduce* the input, you have to "*understand*" the input.

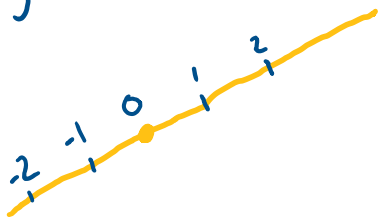
3. Autoencoders can be used to synthesize new data.

   Simply generate a random Z, then return $\text{dec}(Z)$.

1. Autoencoders are like PCA but fancier. What is PCA?



In this picture, $Z_n \in \mathbb{R}$ measures how far along the line to go, from some fixed reference point.



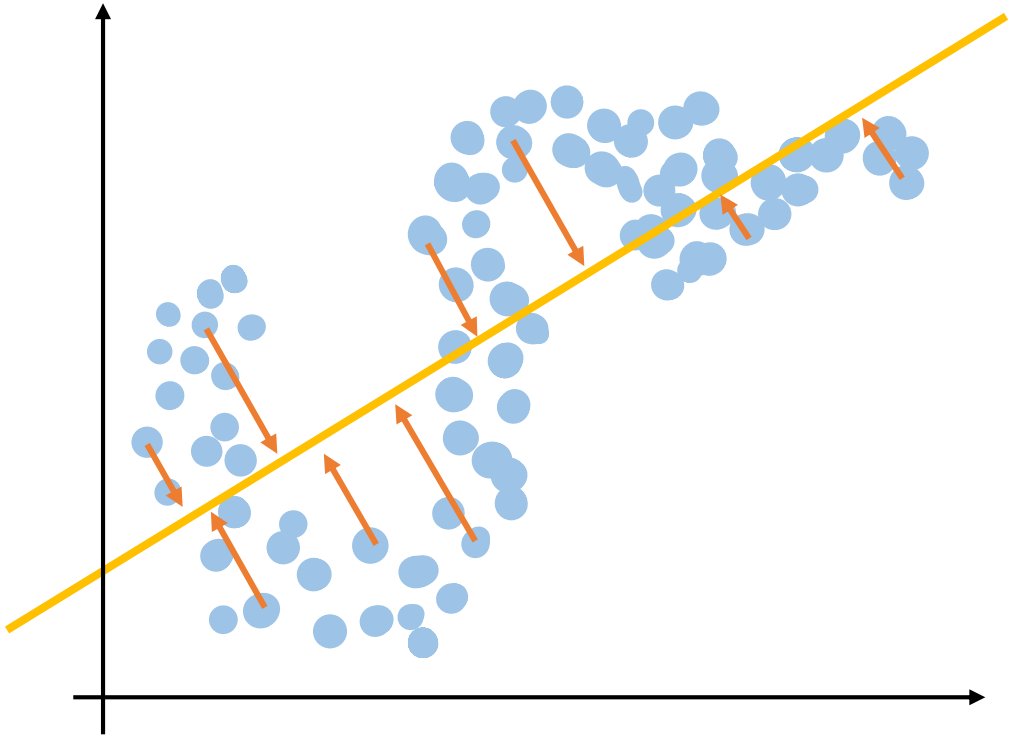
Given a collection of points $X_1, \ldots, X_N \in \mathbb{R}^d$, PCA looks for a linear subspace of dimension $e < d$ to represent the data.

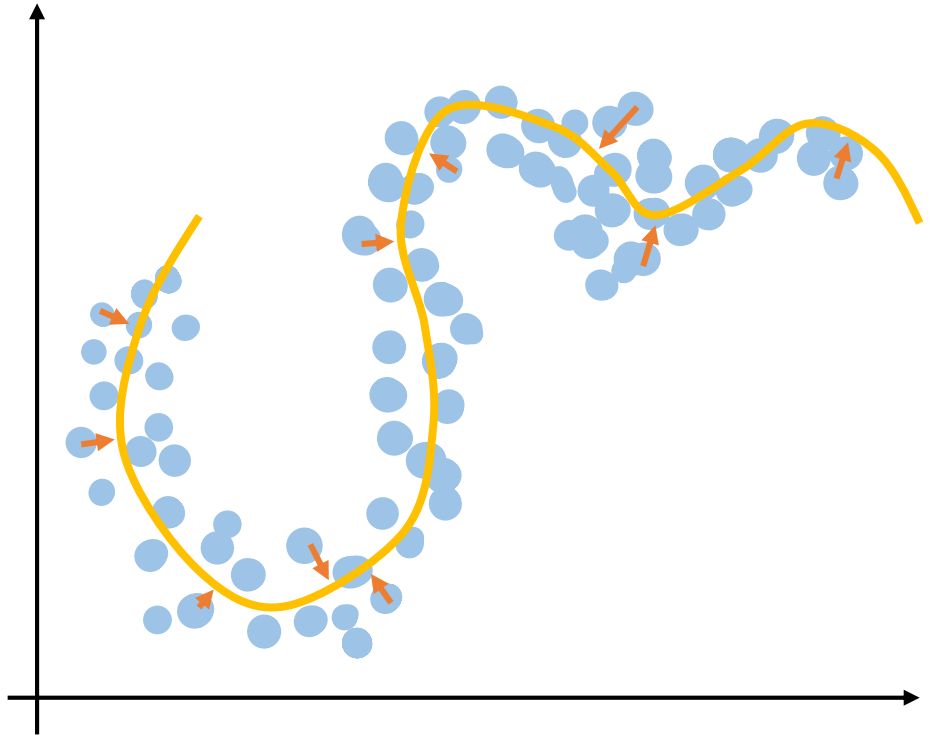
- Approximate X_n by its orthogonal projection $\tilde{X}_n$ onto the subspace
- Equivalently, encode X_n into $\mathrm{enc}(X_n) = Z_n \in \mathbb{R}^e$, and decode as $\tilde{X}_n = \mathrm{dec}(Z_n)$
- PCA picks the subspace so as to minimize the reconstruction loss

$$L(\text{subspace}) = \frac{1}{2}\sum_{n=1}^{N} \|X_n - \tilde{X}_n\|^2$$

Hope: the coordinates in the subspace capture some important “intrinsic features” of the data.

1. Autoencoders are like PCA but fancier. Fancier in what way?



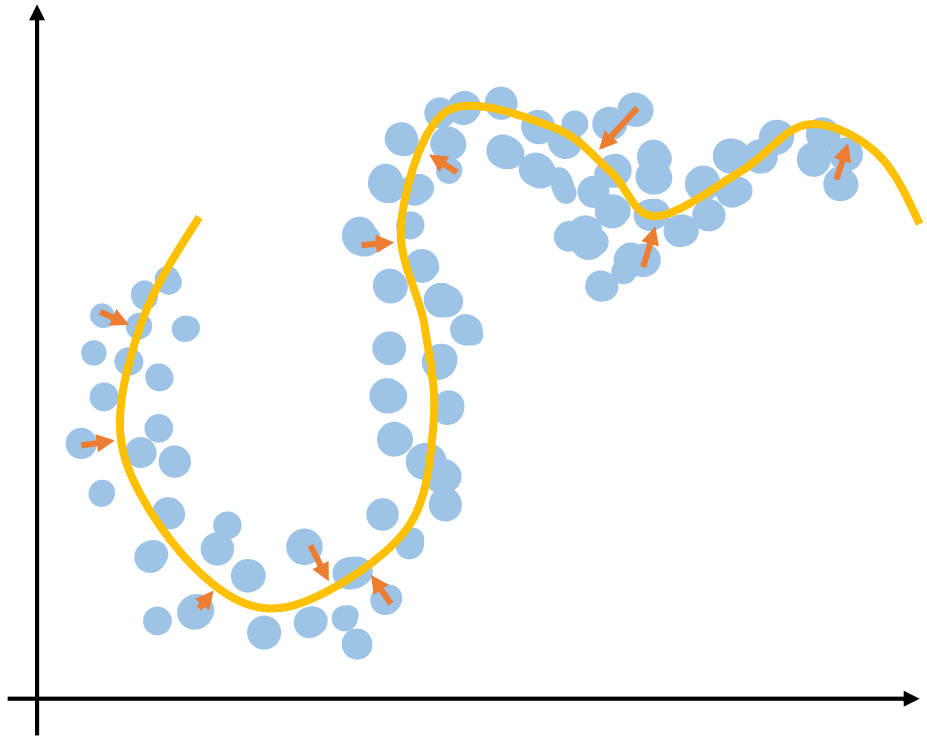


PCA only looks for linear subspaces. But if we allowed nonlinear maps $\text{enc}: \mathbb{R}^d \to \mathbb{R}^e$ and $\text{dec}: \mathbb{R}^e \to \mathbb{R}^d$, surely we'd be able to describe the data better.

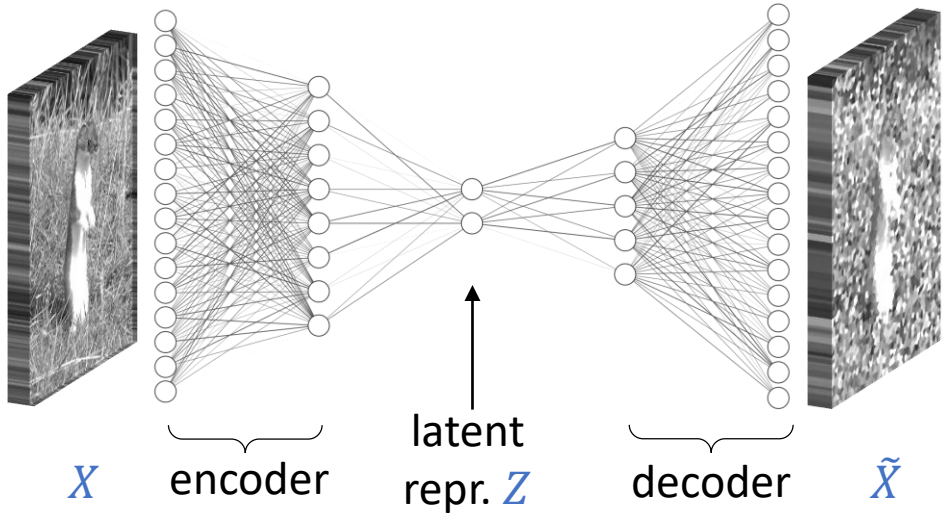
1. Autoencoders are like PCA but fancier.

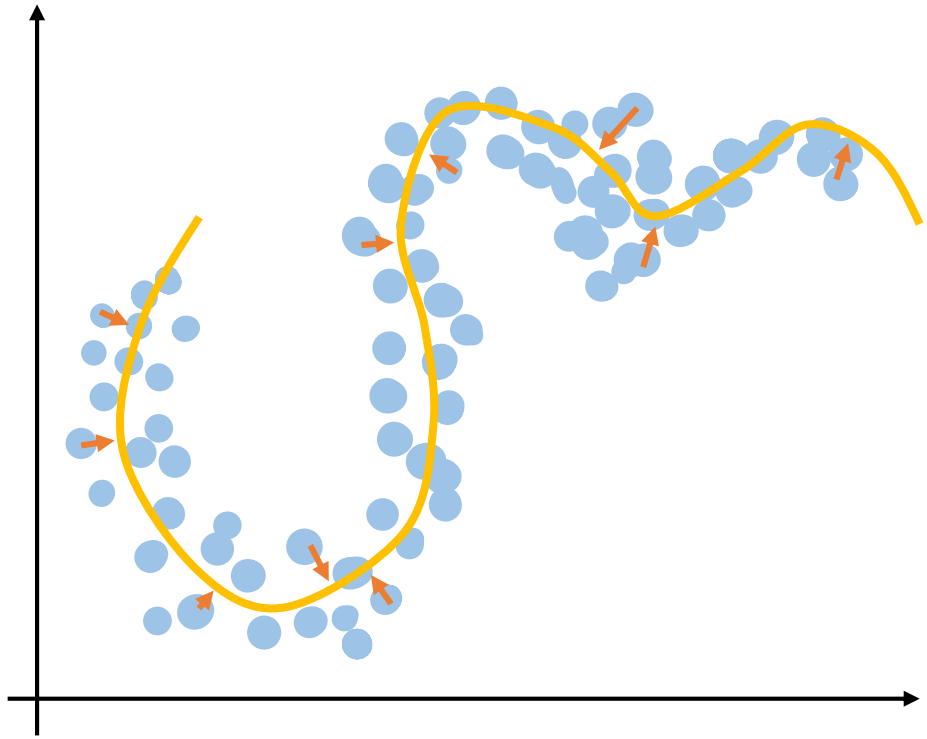
Paper 5:
“β-VAE: learning basic visual concepts with a constrained variational framework”

Autoencoders can be cajoled into learning meaningful subspaces — subspaces whose dimensions correspond to human-meaningful concepts.



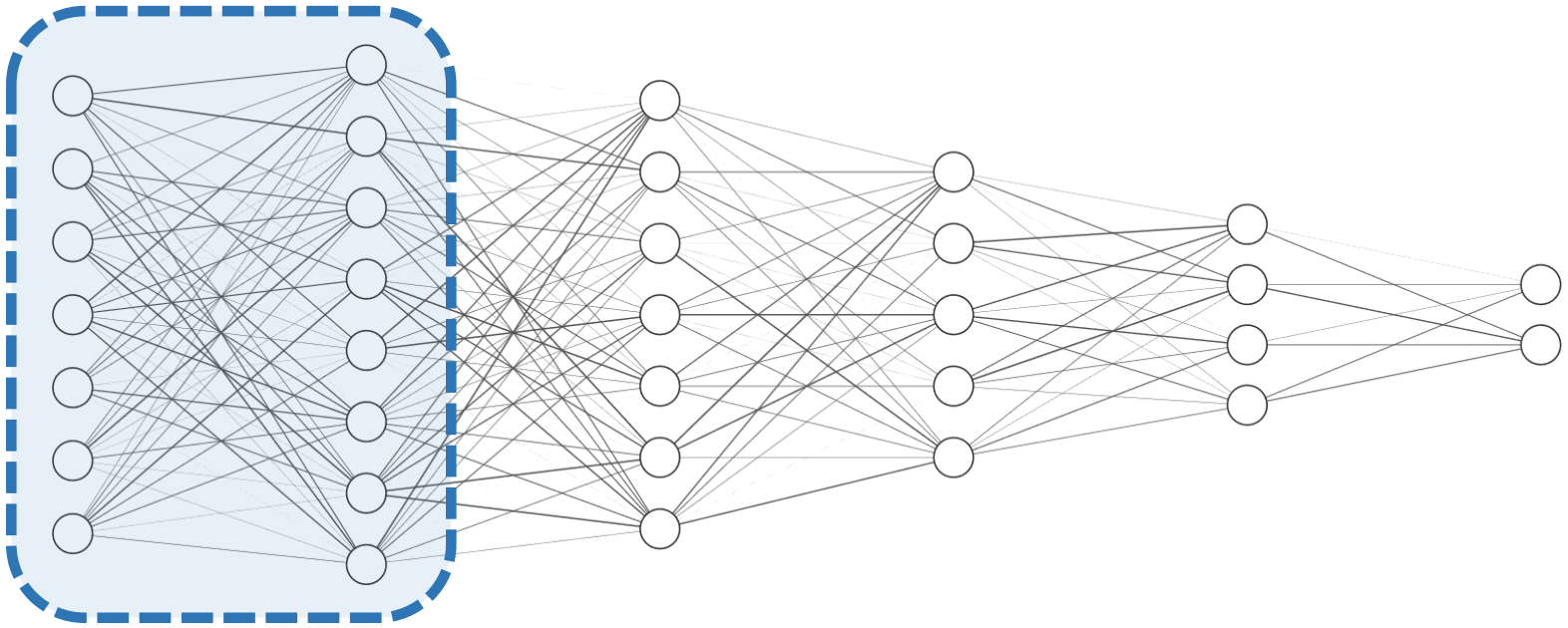
1. Autoencoders are like PCA but fancier.





- Choose the size of the latent layer to control the dimension of the reduced space
- Choose the number of hidden layers in the encoder and decoder, to control how much nonlinearity is allowed
- Choose the reconstruction loss function $L(X, \tilde{X})$ to measure what you care about keeping

2. Autoencoders learn useful representations, for initializing deep networks.



X Z

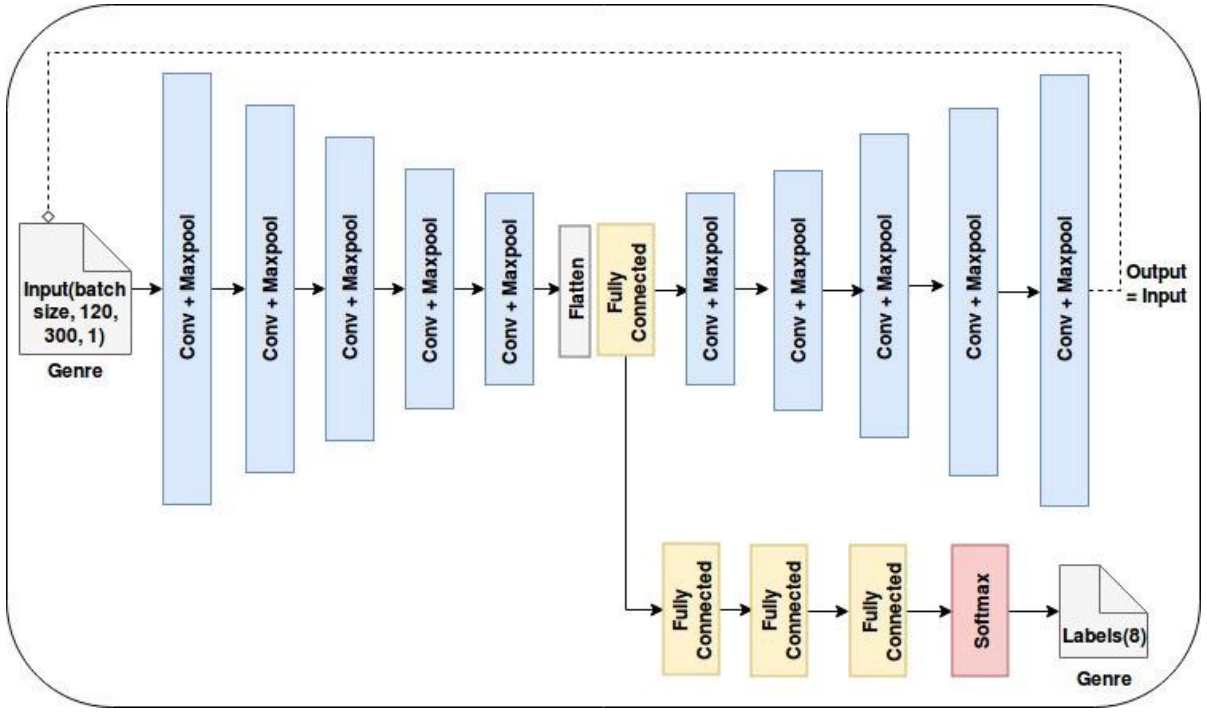
In the olden days, before people got deep learning to work smoothly, a particular type of autoencoder was used to initialize weights so that training didn't take forever.

Hinton and Salakhutdinov, *Deep Boltzmann Machines*, 2009

- Define a physics-inspired joint probability distribution $\mathbb{P}_w(X, Z)$ and calculate the marginal distribution $\mathbb{P}_w(X)$
- Pick weights w to maximize the likelihood of the observed data $X_1, \ldots, X_N$
- Calculate $Z_n = \mathbb{E}(Z|X_n)$
- Repeat for each successive pair of layers

2. Autoencoders learn useful representations. Useful for multi-task learning.

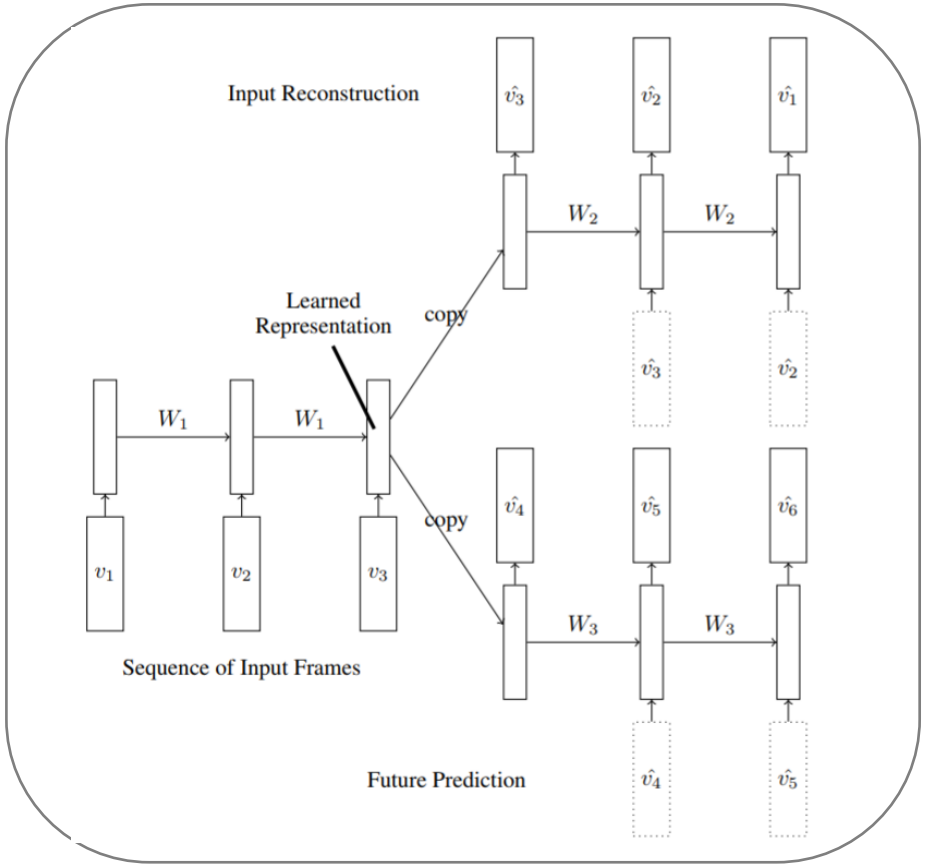
An autoencoder learns the subspace in which most of the data lies. It must be learning the useful “intrinsic features” that describe the dataset. It stands to reason that this can help us with other tasks such as classification.



- Train a neural network with two objectives:
  (a) reproduce the input
  (b) output the desired label
- This is useful if labels are low entropy
  e.g. sentiment classification of text, where you need huge amounts of labelled data to learn useful weights in a RNN
- It’s also useful if you have lots of unlabelled data and only a little labelled data

https://medium.com/@rajatheb/music2vec-generating-vector-embedding-for-genre-classification-task-411187a20820

2. Autoencoders learn useful representations.



Paper 2:
“Unsupervised learning of video representations using LSTMs”

How should sequence data be plugged into an autoencoder? For good results you need multi-task training (otherwise the LSTM decoder is too powerful...)

Paper 6:
“Grammar variational autoencoder”

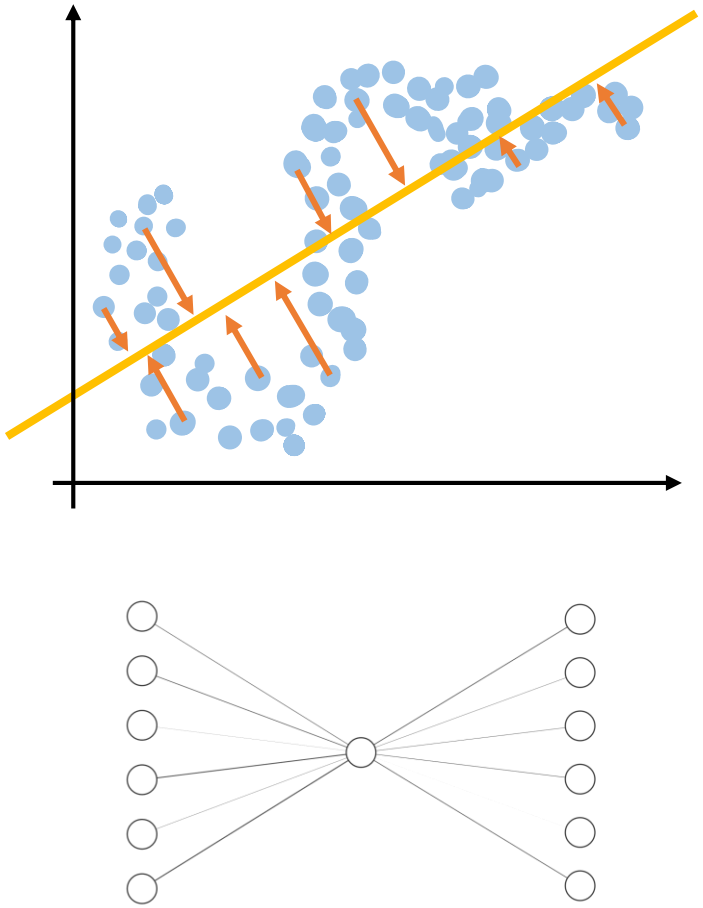
A nice application of autoencoders to structured sequence data

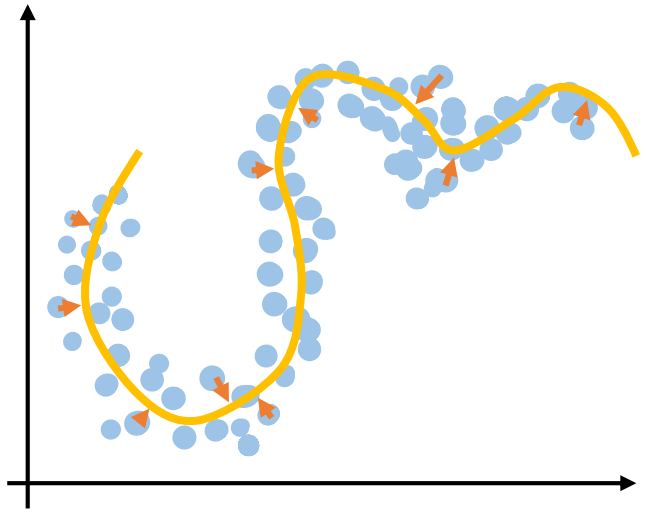
the central challenge of autoencoders:

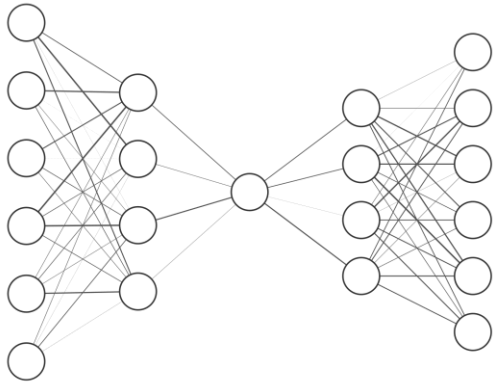
The Goldilocks Problem, or,

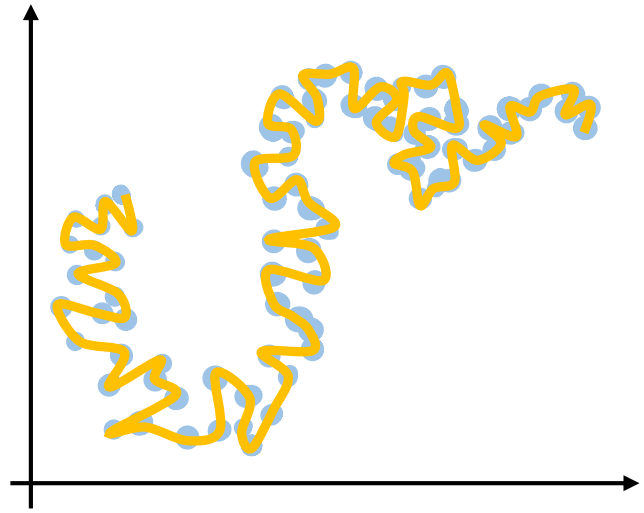
how do we make sure they learn something useful?

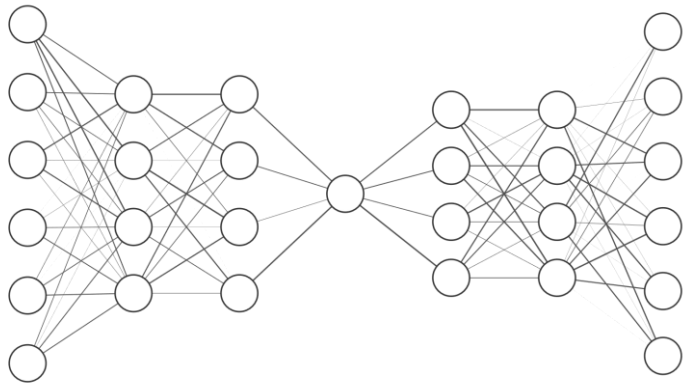
The more layers we have, the more complex the nonlinearity we can learn.



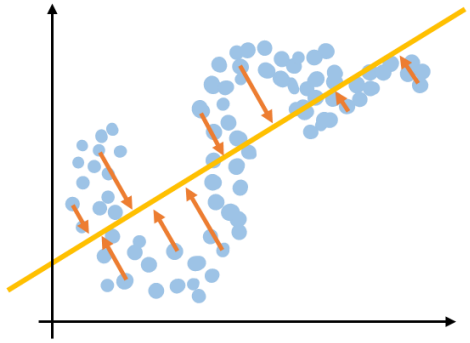


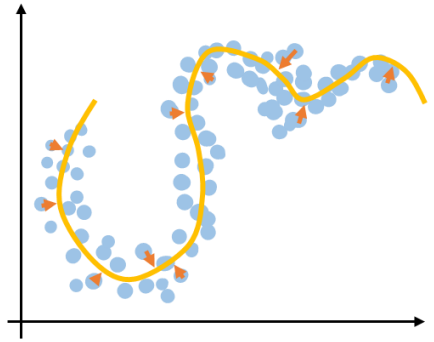






How do we prevent over-fitting?





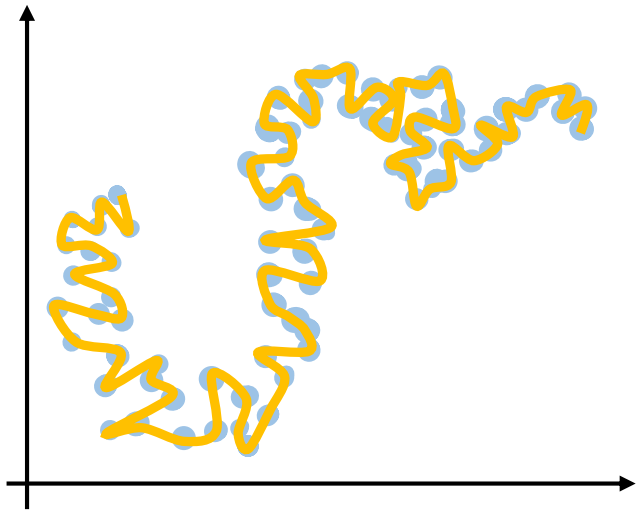
Paper 1:
"Sparse feature learning for deep belief networks"

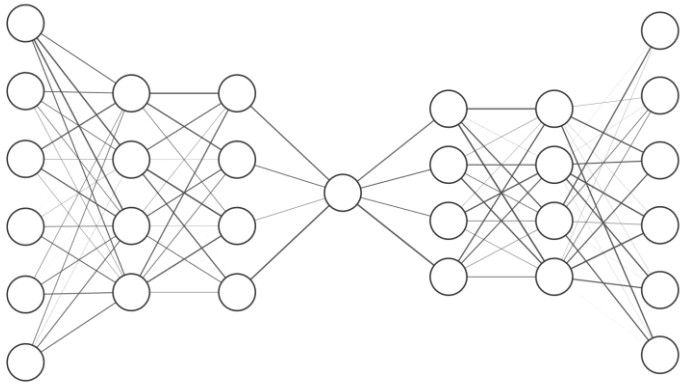
The obvious way to prevent overfitting is to include a regularizer in the loss function, which penalizes overly complex networks.

Regularizing by having limited computational capacity (e.g. PCA, no hidden layers) is just plain dumb.

It's all very well and good to invent a regularizer —
but we also need to be able to cross-validate, to work out how much regularization is needed. How do we cross-validate?

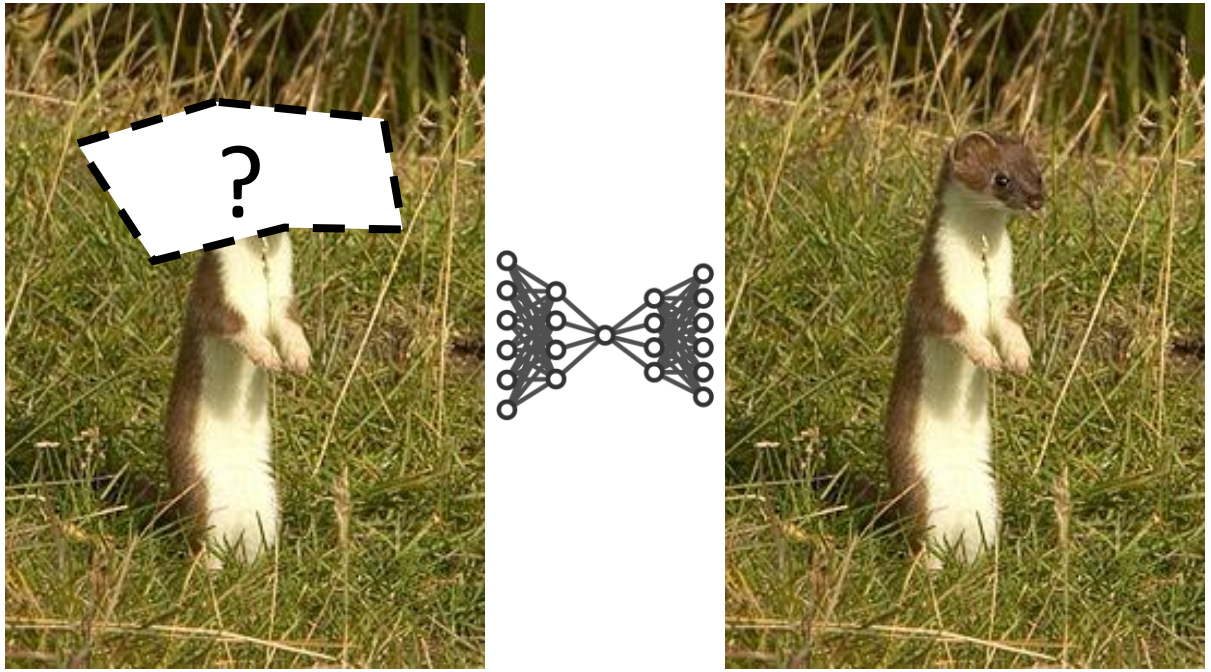
How should we validate an autoencoder? A thought experiment...





- The naive way to validate is to run the network on new unseen data (the validation dataset), and measure the reconstruction loss $\mathbb{E}\, L(X, \tilde{X})$
- Consider a super-intelligent autoencoder, which has learnt to encode input pixel i into bit i of the latent variable $Z \in \mathbb{R}$.
- This autoencoder is surely not what we want — but it will score perfectly on our naive validation test.

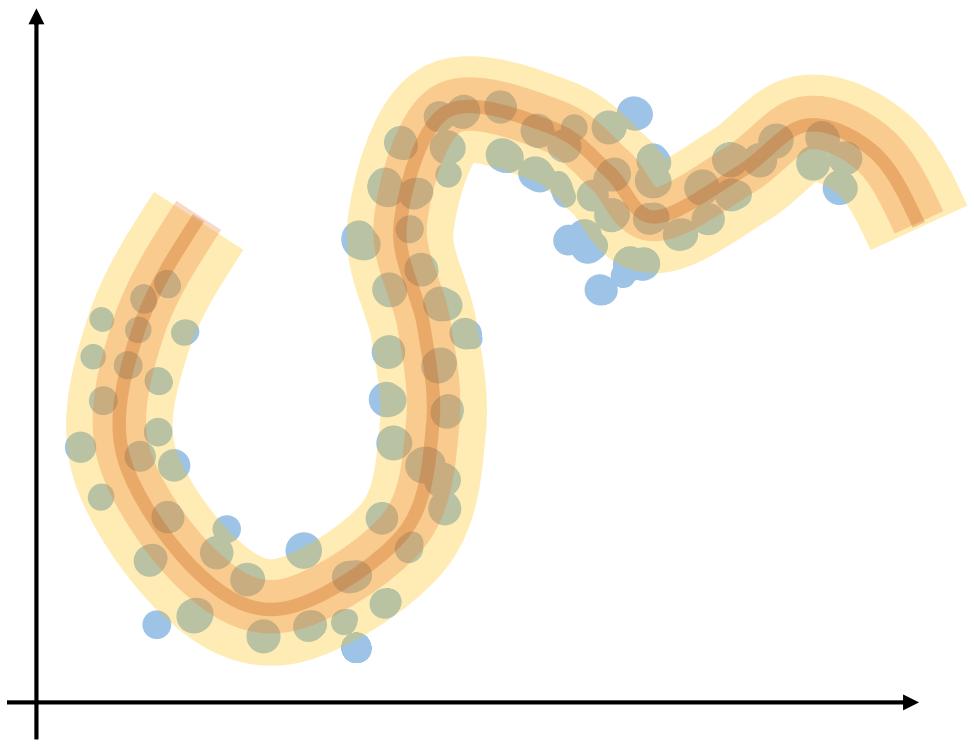
How should we validate an autoencoder?



Paper 3:
“Extracting and composing robust features with denoising autoencoders”

A cunning way to validate:
train an autoencoder whose task is to predict a complete image given its masked version; then we can measure its accuracy on unseen validation data.

The generative view of autoencoders

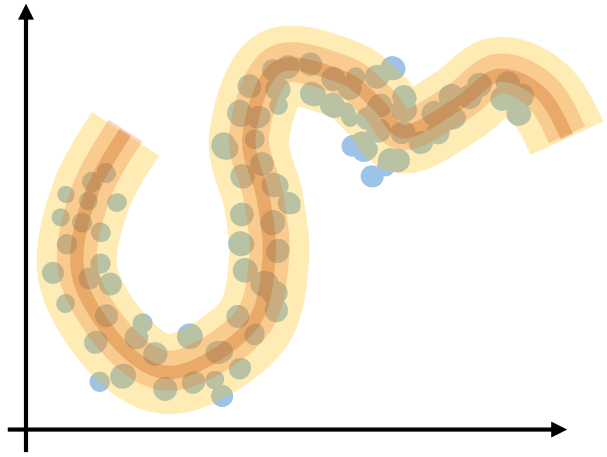


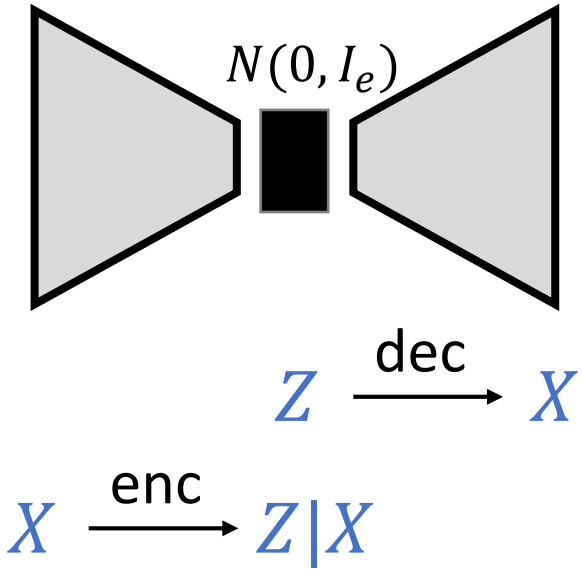
- Suppose we see our goal as "Find a distribution over $\mathbb{R}^d$ that fits the data well"
- Let's call our best-guess distribution $P(\cdot)$
- We want $P(\cdot)$ to put its probability mass where there is data. We can train it by maximizing the log likelihood of the training dataset

$$\log \text{lik}(X_1, \ldots, X_N) = \Sigma_n \log P(X_n)$$

- A distribution that puts its mass in useless places, or spreads it out too thin over the entirety of $\mathbb{R}^d$, will have less to put where there is data, so it will have a bad log likelihood

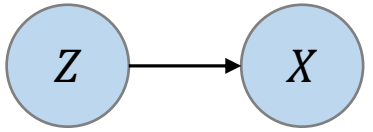
The generative view of autoencoders





Here's a family of distributions that seems to work well for fitting:

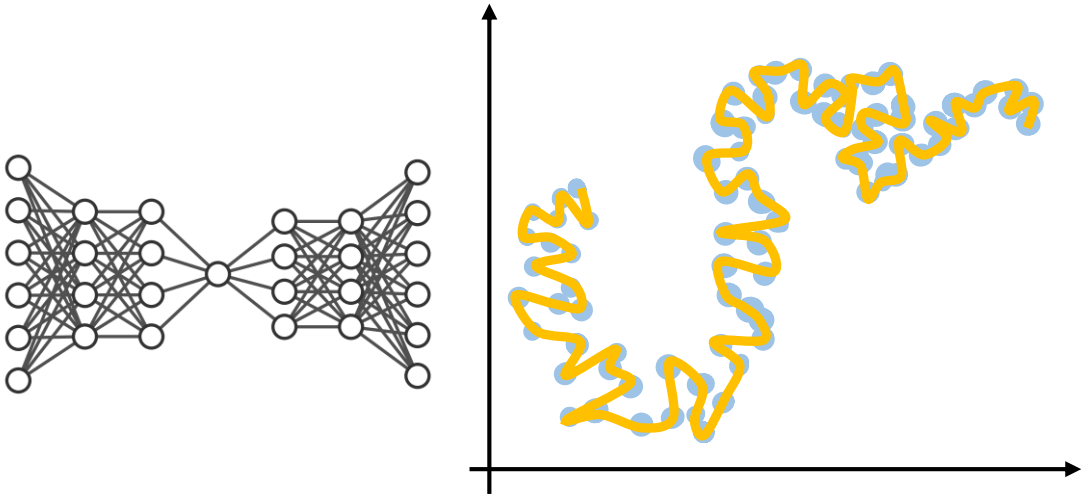
- Let $Z \sim \text{Normal}(0, I_e)$
- Let $X \sim \text{Normal}(\text{dec}\mu(Z), \text{dec}\sigma(Z))$

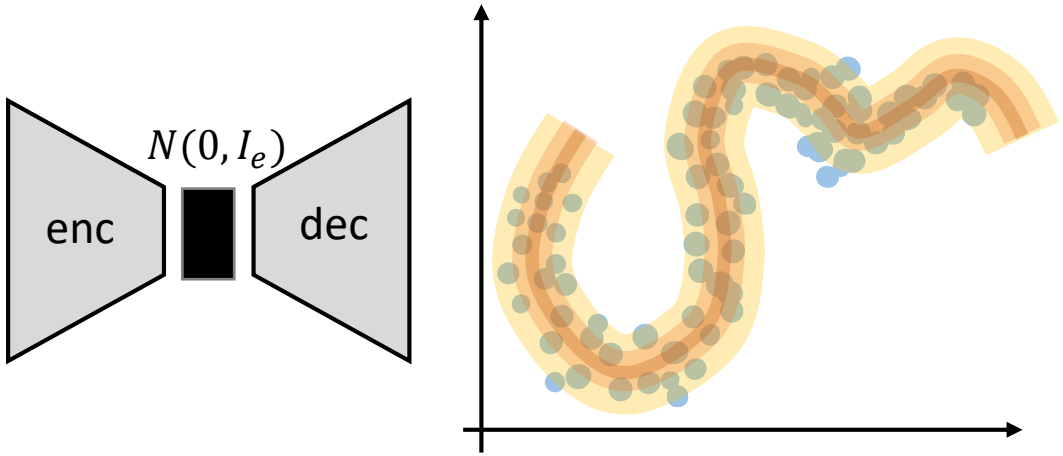


- The decoder functions $(\text{dec}\mu, \text{dec}\sigma)$ are computed by a neural network, and we tune its weights to maximize the likelihood of the training data
- For maximizing the likelihood, it's useful to be able to approximate the conditional distribution of Z given X,

$$Z|X \approx \text{Normal}(\text{enc}\mu(X), \text{enc}\sigma(X))$$

The generative view of autoencoders





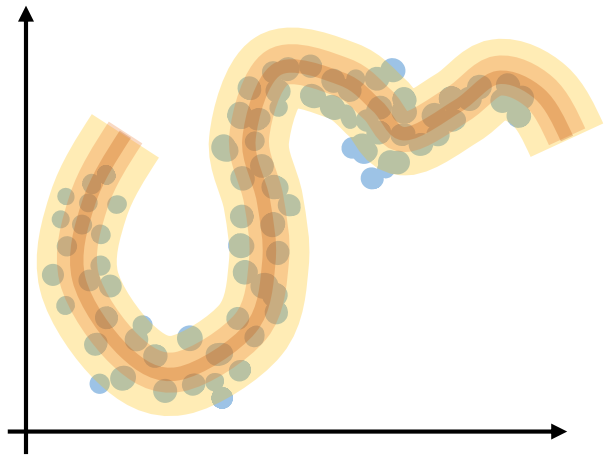
Question: does the noise in this probabilistic autoencoder prevent it from overfitting? Do we still need cross validation?

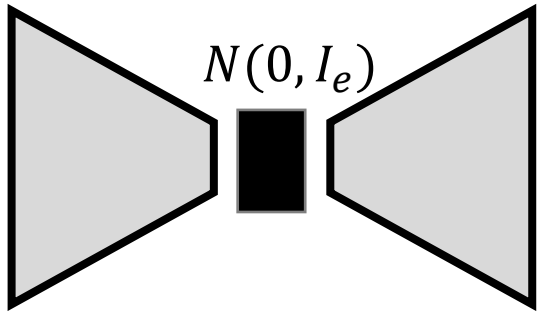
- There is a standard way to measure how well a proposed distribution fits a new (validation) dataset:

  $$\text{score} = \mathbb{E} \log P(X_{\text{new}})$$

  (This is equivalent to *perplexity* in NLP)
- A distribution that overfits and puts all its probability mass on the training datapoints will score badly on new data
- If P depends on hyperparameters, we can use this score for cross-validation

The generative view of autoencoders





$Z \xrightarrow{\text{dec}} X$

$X \xrightarrow{\text{enc}} Z|X$

Paper 4:
"Auto-Encoding Variational Bayes"

This is an important paper, very deep and very subtle. It will take many readings.

Presentation schedule

Friday 8 March 3–5pm	vrs26	Sparse feature learning for deep belief networks *Naive regularization*
	sat62	Unsupervised learning of video representations using LSTMs *How to hook up autoencoders to other neural networks*
	zz362	Extracting and composing robust features with denoising autoencoders *Denoising, as a form of validation*
	mfb37	Auto-Encoding Variational Bayes *Autoencoders as generative models*
Tue 12 March 2–3pm	dai24	β-VAE: learning basic visual concepts with a constrained variational framework *Interpretable latent spaces*
	er513	Grammar variational autoencoder *A little bit of everything...*

- Present for 15–20 minutes
- Find a key message and explain that idea
- Provoke discussion
- Don’t try to summarize everything in the paper
- Don’t believe everything you read
- Working code is always good!
- Don’t show us slide after slide of evaluation numbers.

Autoencoder projects

You all pick one of the six topics, and write a 5000-word project for it. It could be

- a survey of existing methods
- replication of a method from previous work
- small novel experiment

If you want to do your project on autoencoders, send me a 500-word proposal by 11 March.
Project due date is 24 April 16:00

Example project:

Why does no one do cross-validation on autoencoders? What are the methods in use for “validation”?

Example project:

PCA gives us a collection of principal components, ordered by importance. Current autoencoders only give a “blob” latent variable, not ordered components. How can we fix this?

Example project:

We are brainwashed into supervised learning, $P(Y|X)$. Autoencoders can learn the joint distribution $P(X, Y)$. Can we extract the conditional distribution $P(Y|X)$? Or $P(X|Y)$?