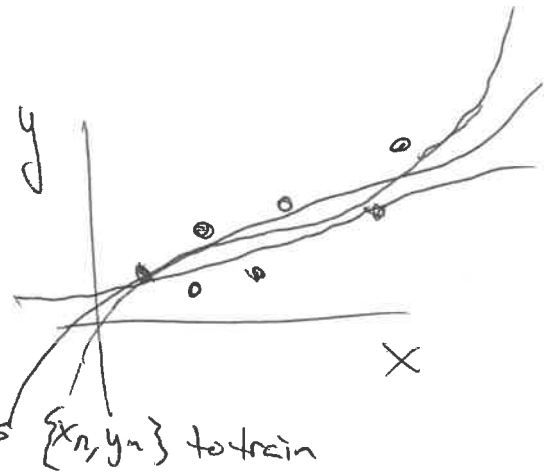
Autoencoders + Unsupervised Learning

Plan: 3 - 3:45 Intro to AEs + Unsupervised Learning

3:45 - 4:15 Project Brainstorming

Summary until today:

Focus on regression task



"supervised learning"

requires input-data pairs $\{x_n, y_n\}$ to train

Model: "Bayesian neural net"

- Prior on weights
- Likelihood of data given weights

Goals of fitting model to data

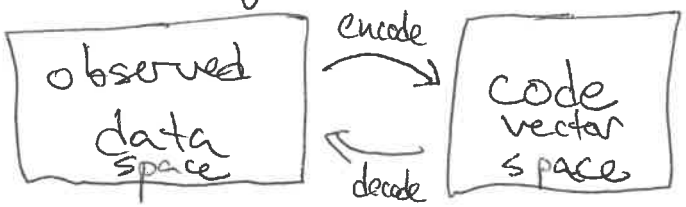
- Make predictions $y_* | x_*$
- Maybe interpreting function itself — scientific insight about trends?
- Maybe interpreting weights themselves (which input dimensions are important?)

Today: What if we have x_n but no y_n.
How could we train useful representations?

Autoencoders via Neural Networks

Observe: N data vectors $x_1, \dots x_N$

Goal: Learn mapping/transformation between



size D x_n $\quad$ c_n size C

Want codes to be compact, yet retain information about x_n

$C < D$ is compact, "undercomplete"

Method

Achieve dimensionality reduction

Encoder network w/ 1 layer

$$c_n = \sigma(Wx_n + b)$$

Decoder network w/ 1 layer

$$x_n' = \sigma(W'c_n + b')$$

Remark: if the activation function $\sigma(\,)$ is just identity, and AE is trained on sq. error reconstruction, the code space of size C that is learned will be same subspace as first C principal components

Towards a Probabilistic model

if x_n is real-valued, decoder last layer no activation! (now-longer!)

$$p(x_n|c_n) = \text{Normal}(x_n \mid \underbrace{\text{decode}(c_n, W', b')}_{\text{mean}}, \underbrace{\sigma^2 I}_{\text{var}})$$

if x_n is Binary vector, decoder last layer squash to $(0,1)$ interval

$$p(x_n|c_n) = \text{Bern}(\underbrace{\text{decode}(c_n, W', b')}_{\text{mean}})$$

Thus, decoders can be used to build likelihoods

Vincent et al 2010 show how to train these models via maximum likelihood

$$\max_{\substack{W,b \\ W',b'}} \sum_n \log p(x_n \mid \text{decode}(\text{encode}(x_n, W, b), W', b'))$$

Remark on PCA goes here

Extension: Denoising AE

Instead of passing clean input vector x_n to encoder, what if we perturb it slightly (flip pixels, etc)

Intuitively, this should not change code c_n too much

For next time: Variational Auto encoders

Imagine a generative model for images:

PRIOR $c_n \sim N(0, I_G)$

LIK $x_n \sim N(\underset{\text{decode}(c_n, W', b')}{\text{mean}}, \tau^2 I)$

How could we train such a model?

esp. to manage uncertainty about code space in reasonable way

Hint: approximate posterior $q(c_n)$ will be fit w/ VI