Autoencoders

want f(x) = x,

Goal: set up the task being more difficult, want to have some useful representation has to happen in middle

g: element wise function
W1: K*D matrix, W2: D*K (is X . W^T)
want bottleneck, K << D

$$\mathbf{h} = g^{(1)}(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\mathbf{f} = g^{(2)}(W^{(2)}\mathbf{h} + \mathbf{b}^{(2)}),$$

Applications:

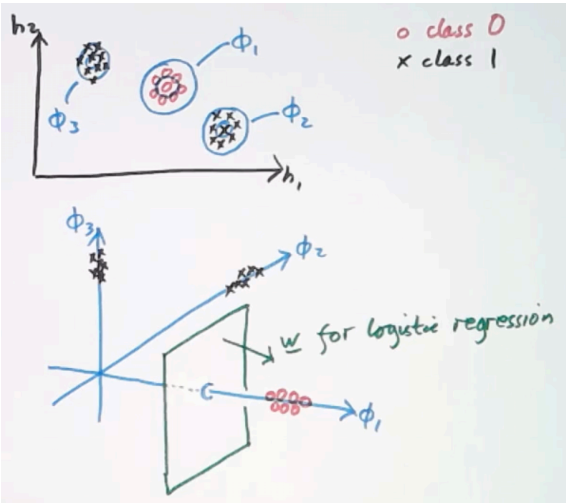
Could plot when k=2;
encoder weight = word vectors / image filters
use transformed vectors as input to another model (word embeddings)
Outlier detection: in testing, if auto encoder produces weird (not similar to input) (MSE of the reconstruction is much larger than the rest, decoder does not know how to decode this outlier), input might be outlier

Sparse autoencoders:

hidden layer is high dimension, K >= D

since in high dim, different class of datapoints could be pushed to different dimensions, easier to deal with by simple ML methods

e.g. put 3 basis functions, transform to basis function space, easier to classify



But require constrain: encourage most elements in middle hidden dimension approximately 0

sparsity = fewer neurons can be activated (neuron output ≠ 0) at the same time, creating an information bottleneck

(prevent from not learning anything useful)

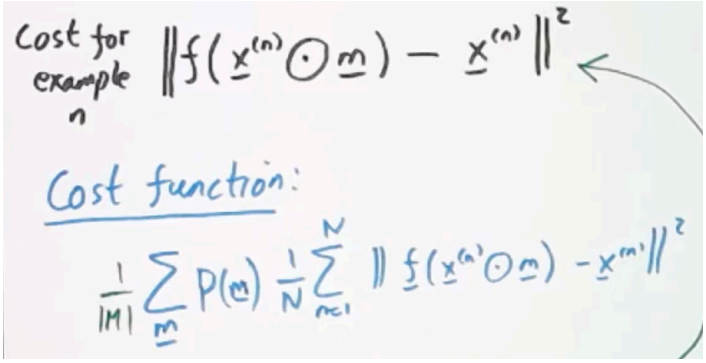
Use L1 or KL divergence to encourage sparsity

Denoting auto encoder:

create sparse representation, though not explicit cost function that define sparse

with mask m {0,1}, f reconstruct original

cost function = expectation over possible masks (E_p(m))



Learning new methods: think of **"do I know how to implement every stage of this?"**

e.g. denoting autoencoder mask: if see the same image again, use the original mask or new mask? (Answer: New mask, not overfit)

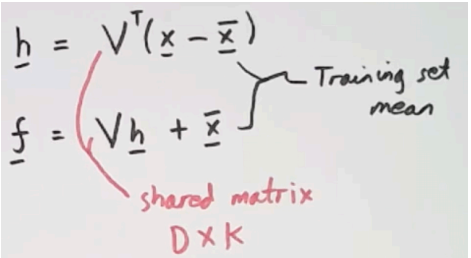
To derive the **cost function** (blue above):

Mask m isn't part of training data, is generated randomly, so if cost function is going to be defined, need to have a way to sum m so they disappear from the math

mask comes from P(m), average those (by sum over m), want reconstruction to work well under any mask (= distribution of masks)

PCA

for g(a) = a in linear auto encoder,



Sample covariance: of centered data (subtracted mean)

$$\text{cov}(X - \bar{X}) = \frac{1}{N}\sum_{n}(\underline{x}^{(n)} - \bar{\underline{x}})(\underline{x}^{(n)} - \bar{\underline{x}})^{T}$$

Set columns V: take sample covariance, take eigenvectors (eigenvalue decomposition of covariance matrix), select top K eigenvectors

get square error as good as autoencoders

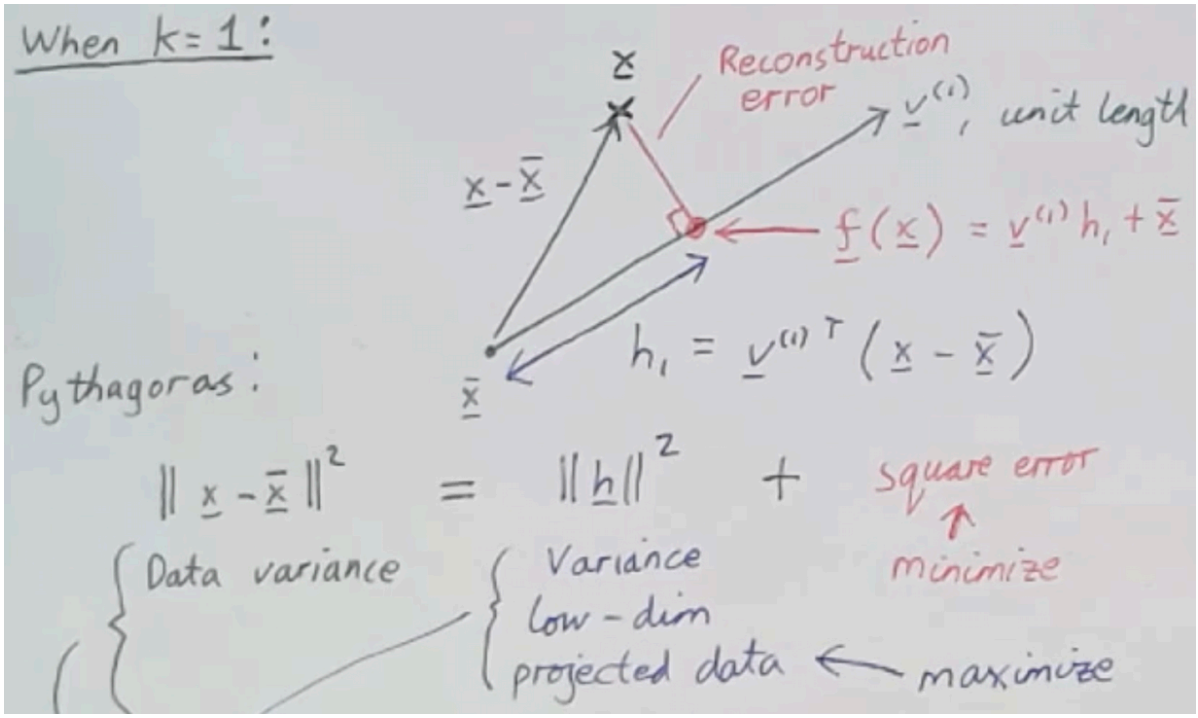
Advantages of PCA:

use standard matrix functions

same answer every time

solution nested: largest 1,2,…k eigenvectors the same; but not in auto encoders

v^(1): eigenvector (principle component)
x: datapoint, x hat: mean
PCA projects point onto PC

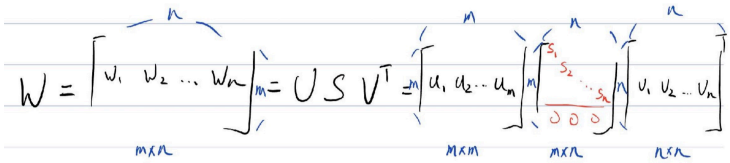


Think of PCA in 2 ways: auto encoding objective (minimize square error, = equation above),

OR finding PCs

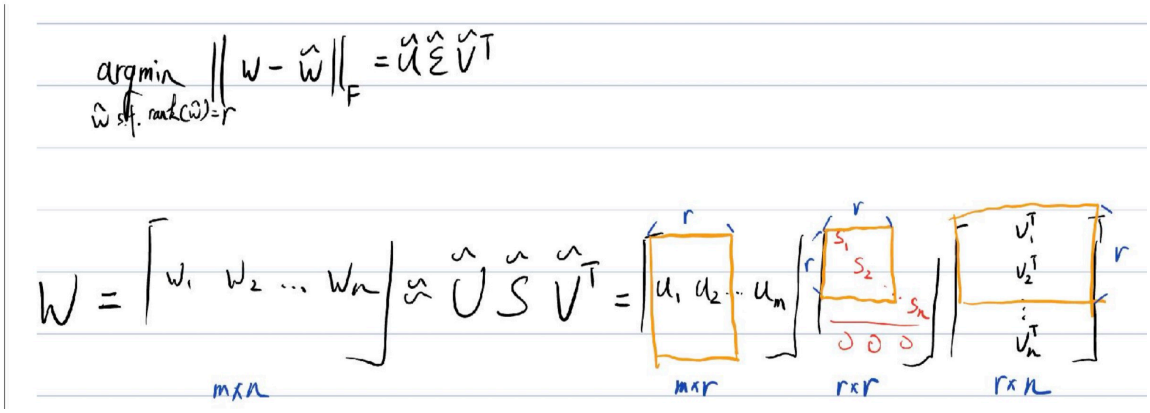
How much variance of data explained by PCs: how much ||x-x||^2 put into ||h||^2, else is square error (reconstruction error, red above)

SVD:



Truncated SVD:

Eckart Young theorem:
The best approximation to matrix W of rank r is given by:



each element in W could be approximated by: corresponding row in U * element in S * corresponding column in V

when applied to centered data (subtracted mean), SVD gives the same solution as PCA

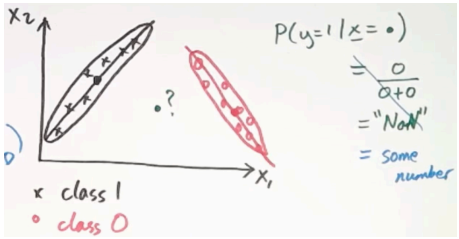
- columns of US are PCs
- diagonal values of S are singular values (could be transformed to eigenvalues)
- columns of V are principle directions (axis)

Probabilistic PCA:
transform K dimensional compressed data into original D dimension

covariance is low rank, rank K, because it only has K independent rows or columns

so vector x generated from this model will lie exactly on a linear subspace of dimension K (e.g. the line bellow)

so the likelihood of such a model will be zero if any data-points lie outside the K-dimensional subspace (e.g. probability of every points outside the line = 0)



Solution: add some noise with variance σ in dimension D (what variance?), then have full rank covariance

$$\tilde{\mathbf{x}} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, WW^\top + \sigma^2 \mathbb{I})$$

Since point x^m is linear combination of embedding h^m, distribution of x is [cov changes under linear combination]

Since V is D*K matrix, V.V^T is low rank, so a 1D line in 2D space, so probability of every points outside the line = 0

So add some gaussian noise for every point in x^m

P(x^m | h^m), prob of point x^m given embedding we have seen, h^m

X^m is mean, noise is cov

P(x), integrate joint probability of X^m and h^m (marginalizing (removing) h^m)

apply chain rule