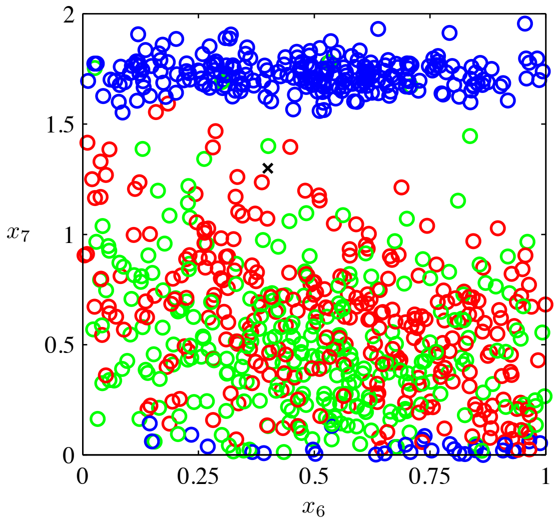
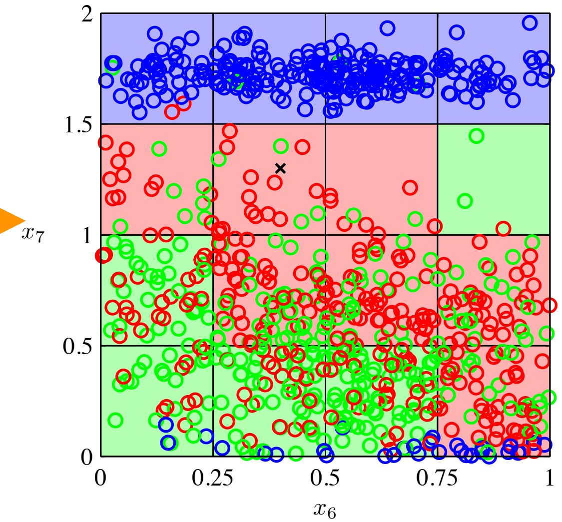
What is curse of dimensionality?

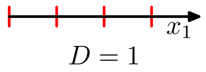
Ex: a simple classification scheme (related to decision tree)

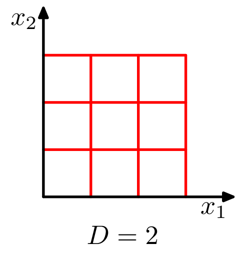


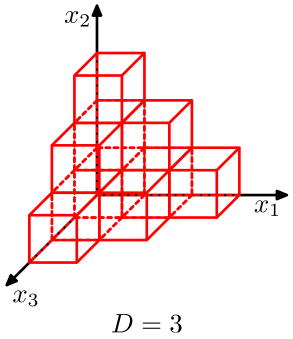
divide up
into small cells



of cells grows exponentially







# of cells $\quad r^D$

r: number of divisions in each dimension

That is a lot even if D is just moderately large!

So to cover the whole space reasonably well,
you need exponentially number of training data points!

Another example

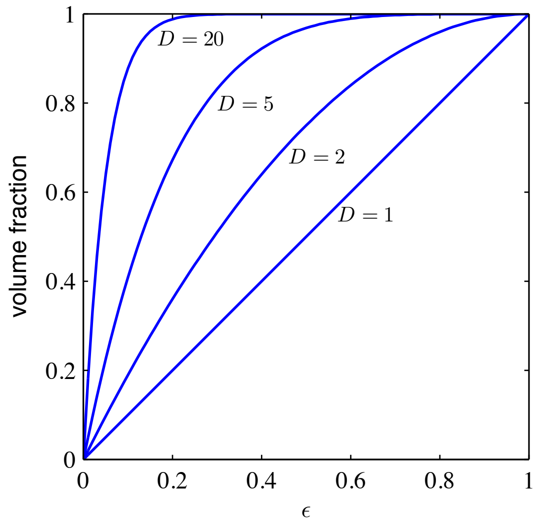
Nearest neighbor classification

Say we want points of different classes are at least ε away from each other

To put things in scale (otherwise, you can scale ε arbitrarily), assume all points are at most 1 away from origin.

How many points between 1 and 1-ε from the origin?

$$1 - (1 - \epsilon)^D$$



Namely, when D is high, you cannot figure out data points that are ε away even ε is pretty big

An overview of dimensionality reduction

Parametric approaches

Linear: PCA, Fisher LDA, NMF, random projection, CCA, etc

Nonlinear: Neural networks, generative topological mapping, ICA

Nonparametric approaches

kernelized version of parametric methods: k-PCA, kICA, kCCA

manifold learning

Gaussian processes

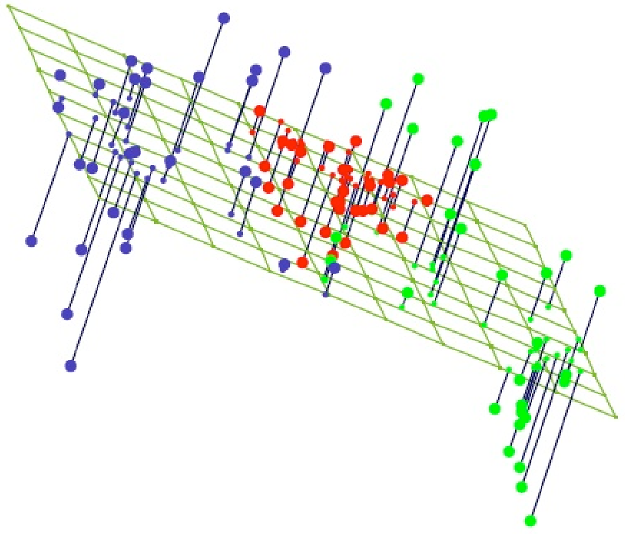
Linear dimensionality reductions

Many examples

Principal component analysis (PCA)

Fisher discriminant analysis (FDA)

Nonnegative matrix factorization (NMF)



Framework

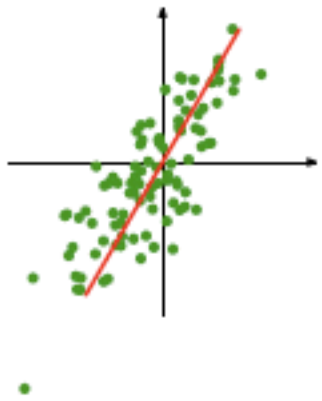
$$\boldsymbol{x} \in \Re^D \to \boldsymbol{y} \in \Re^M$$

$$D \gg M$$

$$\boldsymbol{y} = \boldsymbol{U}^\top \boldsymbol{x}$$

linear transformation of original space

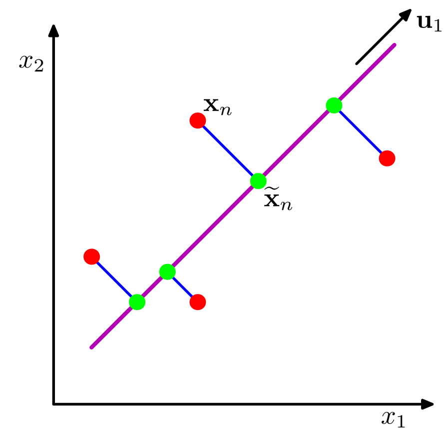
PCA: first principal component



Objective:

maximize variance of points projected on the line

Derivation on whiteboard



The framework of PCA

Assumption:

Centered inputs (if not, subtract mean) $\quad \boldsymbol{x} \in \mathbb{R}^D \quad \sum_i \boldsymbol{x}_i = \mathbf{0}$

Projection into subspace $\quad \boldsymbol{U} \in \mathbb{R}^{D \times M} \quad \boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$

$$\boldsymbol{U}^{\mathrm{T}} \boldsymbol{U} = \boldsymbol{I}$$

Solution

each row is a data sample

Computer covariance matrix $\quad \boldsymbol{S} = \frac{1}{N} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}$

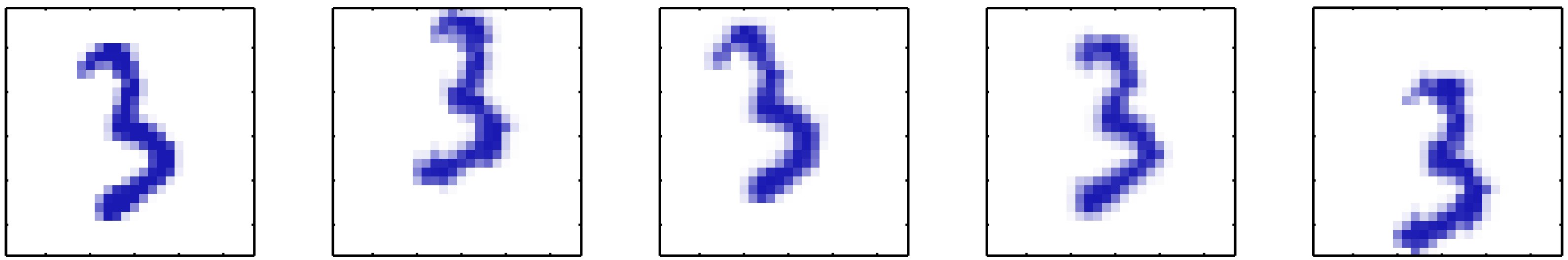
Diagonalize S: $\quad (\boldsymbol{u}_d, \lambda_d) \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$

Use top D eigenvectors to form U

$$\boldsymbol{U} = (\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \cdots \ \boldsymbol{u}_M)$$

Examples of running PCA

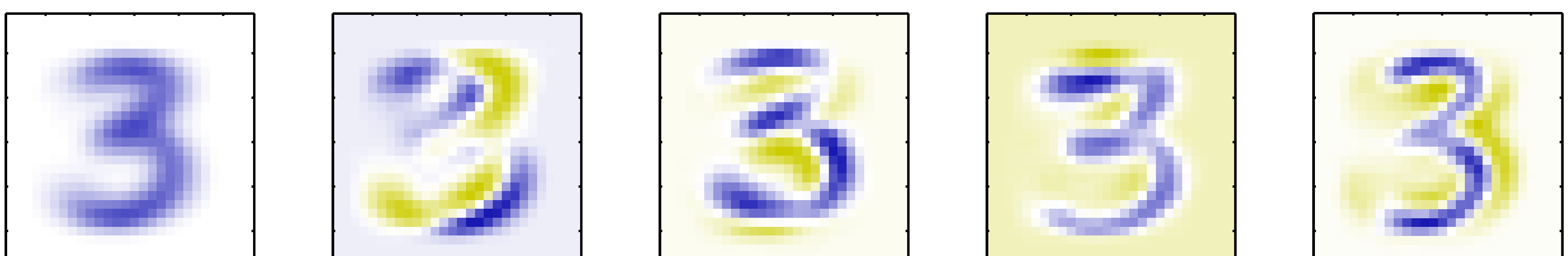
Original Images



Eigenvectors

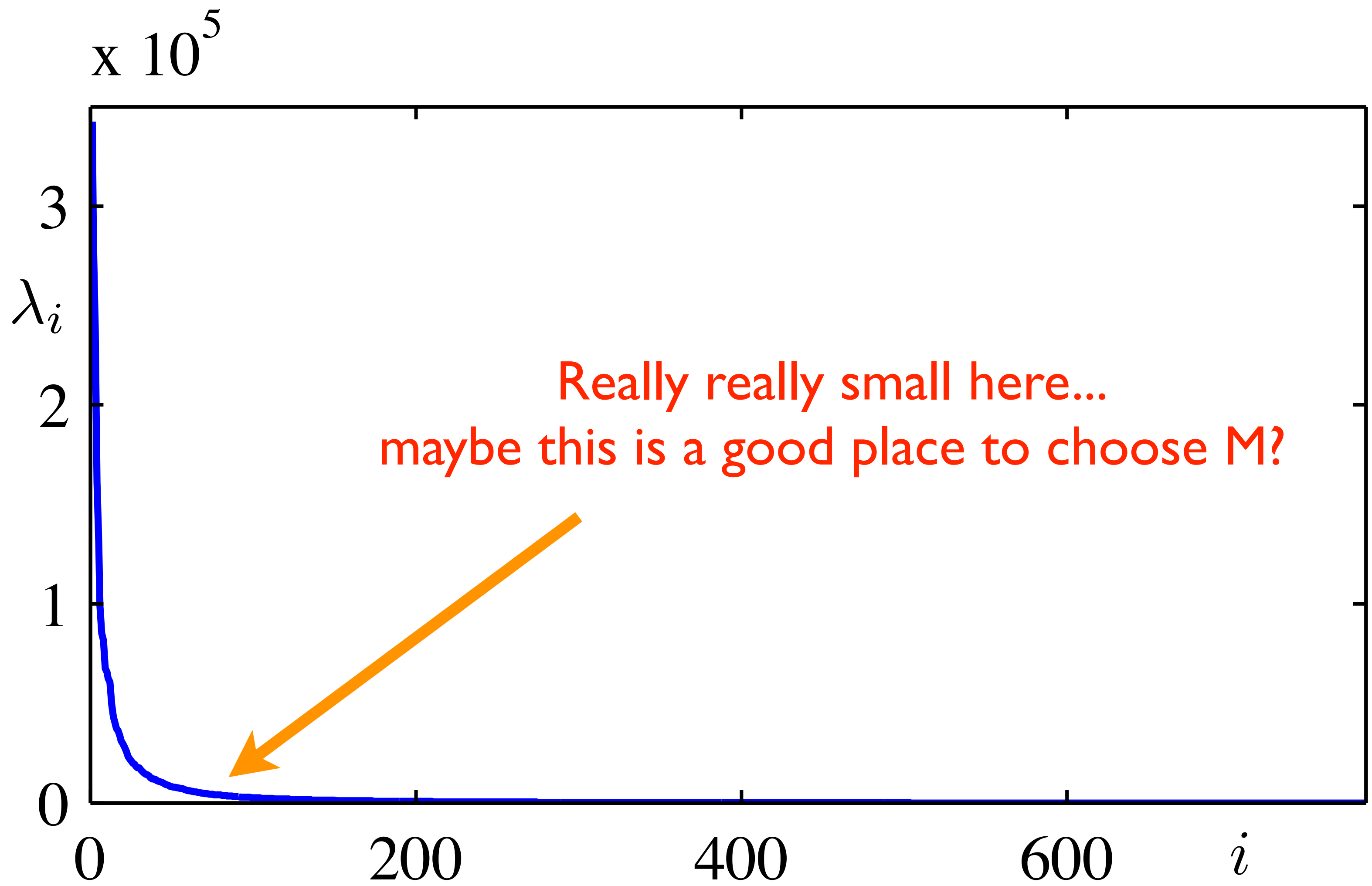
they look like blurred original images

Mean $\qquad \lambda_1 = 3.4 \cdot 10^5 \qquad \lambda_2 = 2.8 \cdot 10^5 \qquad \lambda_3 = 2.4 \cdot 10^5 \qquad \lambda_4 = 1.6 \cdot 10^5$



Used to centralize inputs

Eigenspectrum of the covariance matrix

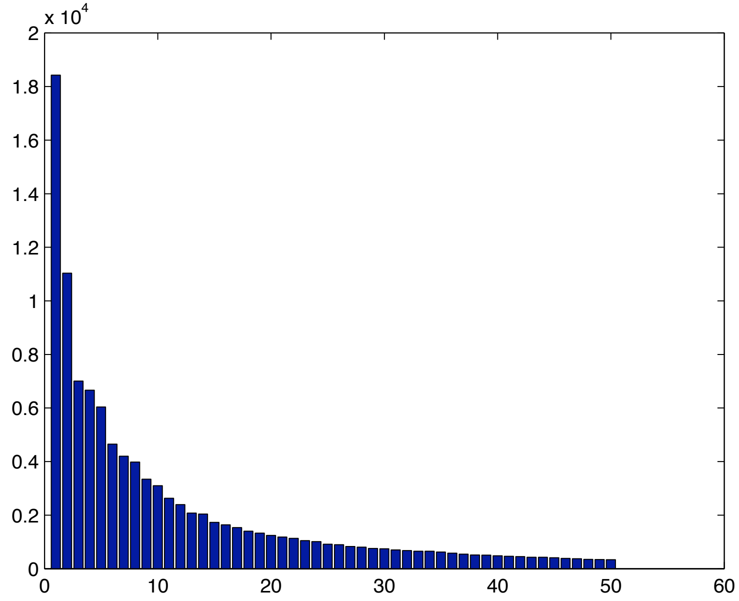


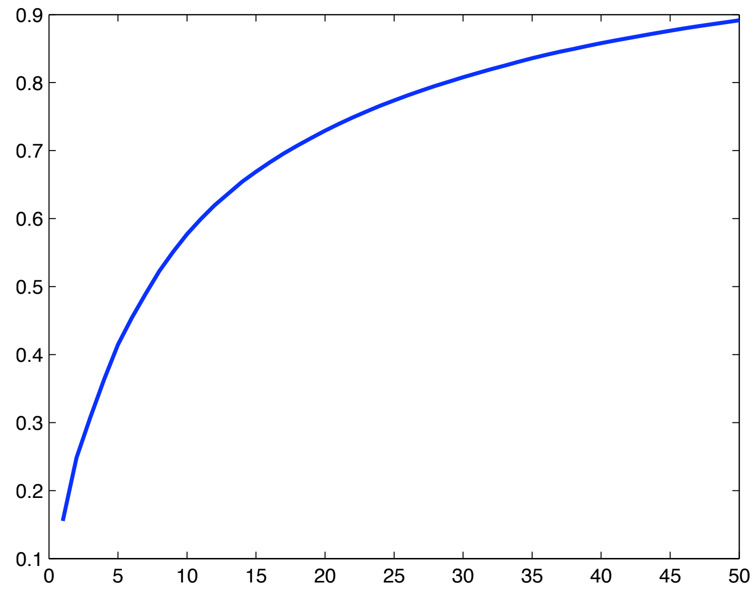
(a)

A slightly sensible approach

common choice is 95% or 90%

$$\frac{\sum_{d=1}^{M} \lambda_d}{\sum_{d=1}^{D} \lambda_d} \geq \text{Threshold}$$





Application of PCA

Preprocessing

Diagonalize data

$$\boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

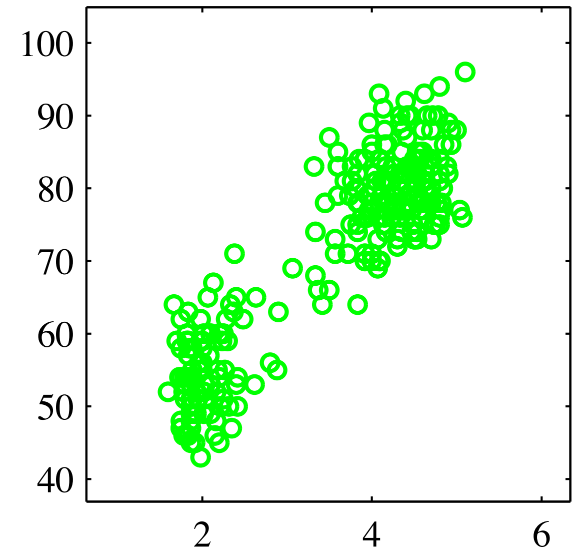
Normalize data (whitening)

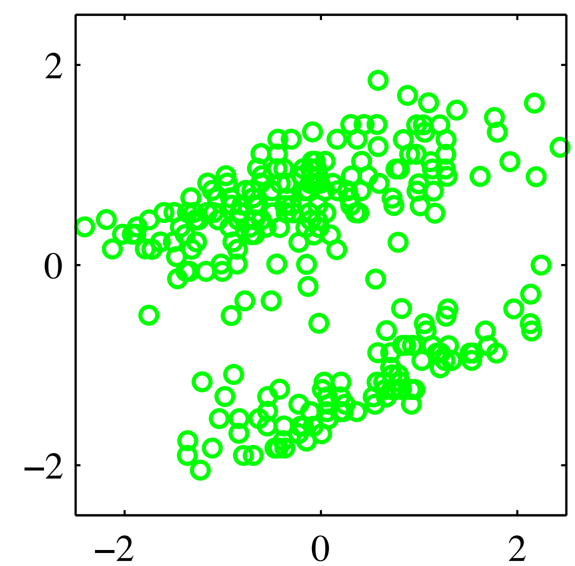
$$\boldsymbol{y}_i = \boldsymbol{\lambda}^{-1/2} \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

Benefits:

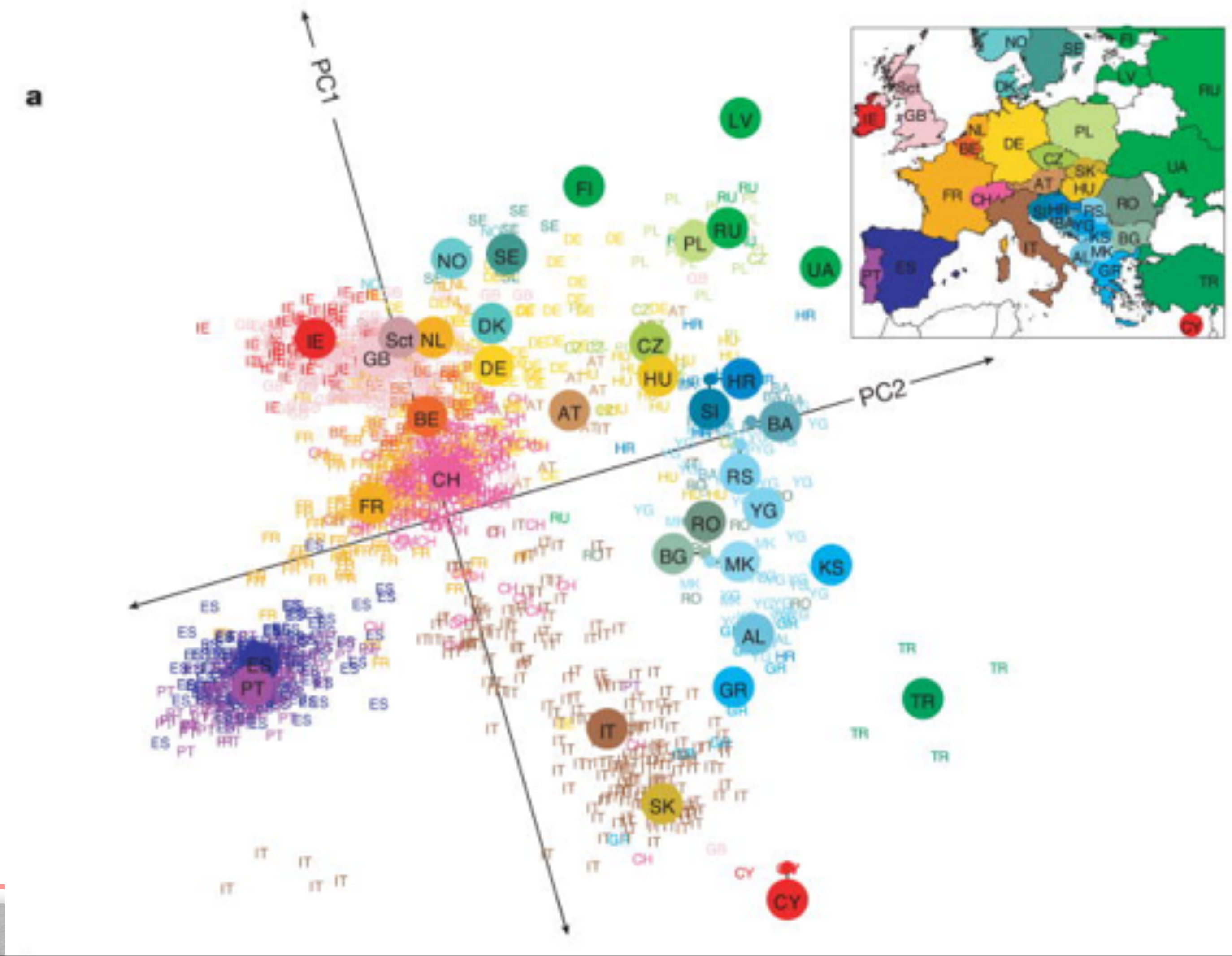
1) depress noisy features

2) couple with other models





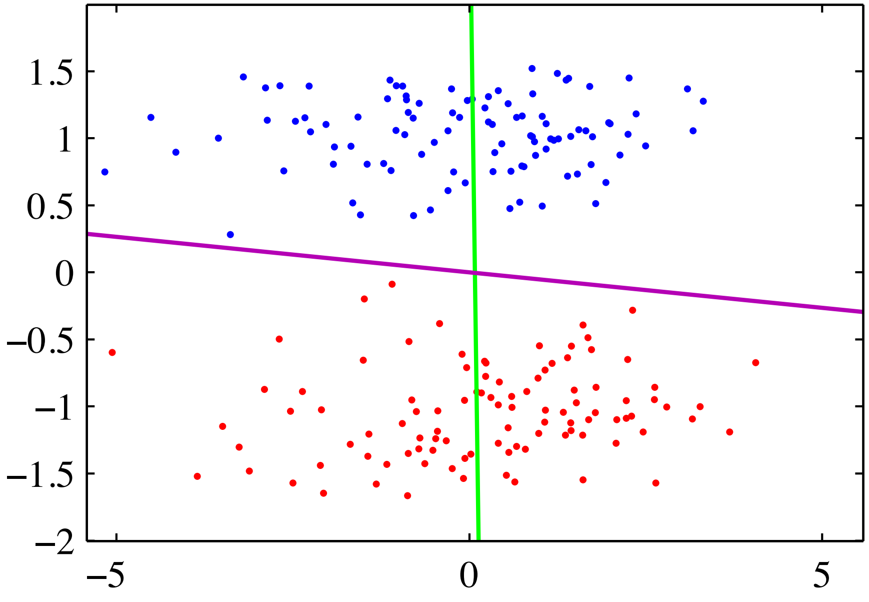
Visualizing data with PCA



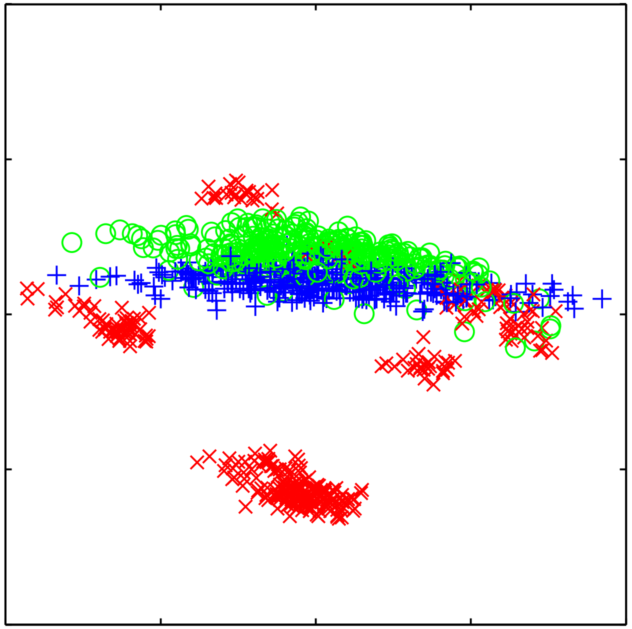
Does PCA help for classification?

Not always!

no help



help



Visualization

The second interpretation of PCA

minimum reconstruction error

Consider a basis of a subspace with M-dimension where M < D

$$\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M \in \mathbb{R}^D$$

Consider the approximation error by linearly combining the bassis

$$\sum \|\boldsymbol{x}_n - \sum_m \alpha_{nm} \boldsymbol{u}_m\|_2^2$$

If we minimize the error by optimizing both the basis and the coefficients, we get

$$\alpha_{nm} = \boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{u}_m$$

$\boldsymbol{u}_m$ is the m-th largest eigenvector of S(ample) covariance matrix

Unexplained variance and residual error

Unexplained variance

Each projection direction u_i contributes λ_i variance

With D-dimensional data, M projection directions, the "leftover" is

$$\sum_{d=1}^{D} \lambda_d - \sum_{m=1}^{M} \lambda_m = \sum_{m=M+1}^{D} \lambda_m$$

Reconstruction error

$$\sum \|\boldsymbol{x}_n - \sum_m \alpha_{nm} \boldsymbol{u}_m\|_2^2$$

The two are exactly the same!!!

Issues with PCA

Choose dimensionality

ad hoc approach: we have seen that

more principled approach: bayesian PCA, etc

Missing values

what if data is missing?

Probabilistic PCA: a little bit later

Computational complexity

For large input dimensionality D

the covariance matrix is D x D

finding eigenvalue and eigenvector for the covariance matrix can be very expensive

The trick

section 12.1.4

Same set of eigenvalues

This is called Multidimensional scaling (MDS)

$$\frac{1}{N}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{u} = \lambda\boldsymbol{u} \rightarrow \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boxed{\boldsymbol{X}\boldsymbol{u}} = \lambda\boxed{\boldsymbol{X}\boldsymbol{u}}$$

Same low dimensional representation

$$\boldsymbol{u} \propto \boldsymbol{X}^{\mathrm{T}}\boldsymbol{v}$$

Different computational cost

PCA's matrix scales quadratically in D

MDS's matrix scales quadratically in N

Big win for MDS when D is much greater than N !

Use nonlinear kernel?

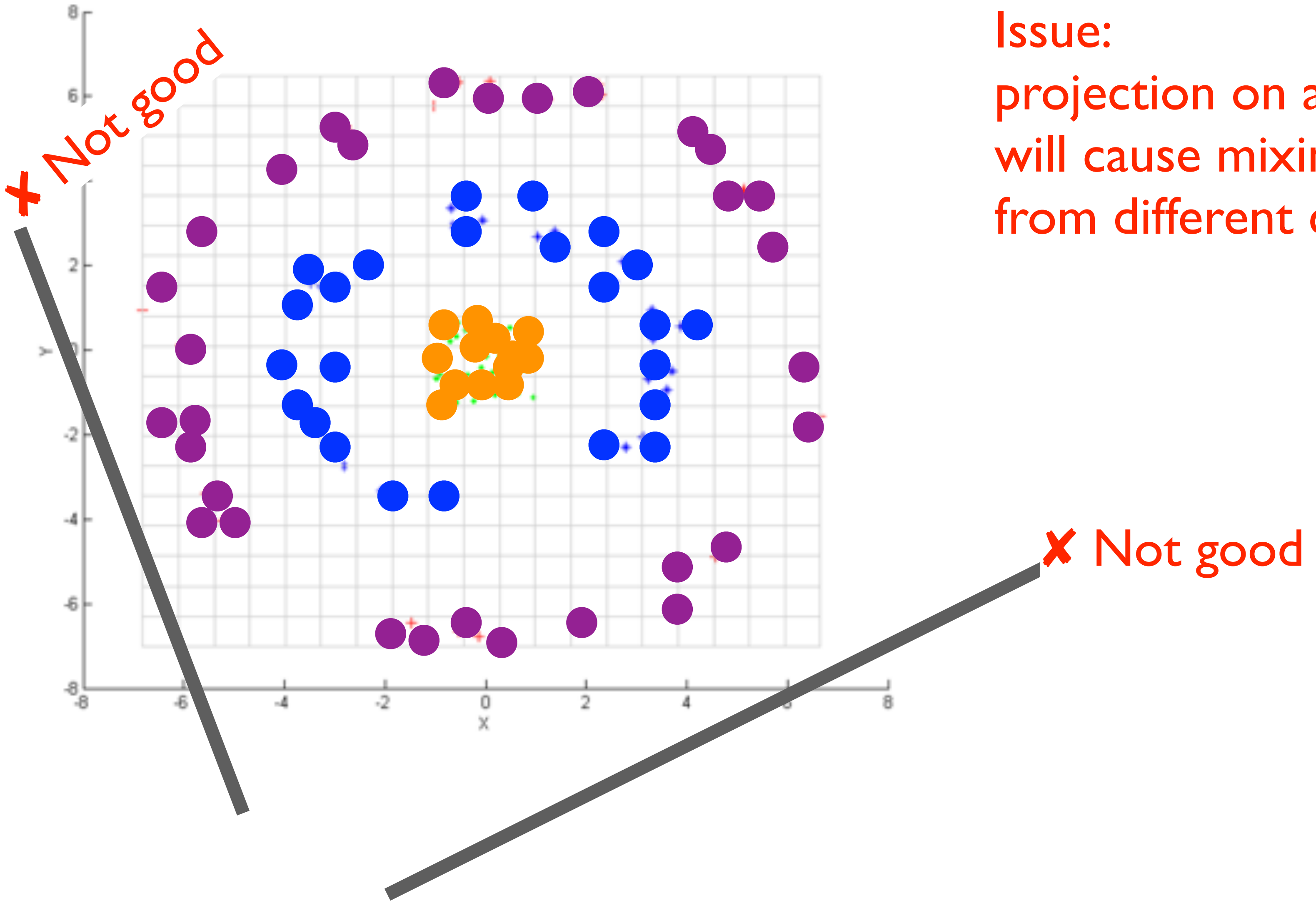
$$\frac{1}{N}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{u} = \lambda\boldsymbol{u} \rightarrow \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{u} = \lambda\boldsymbol{X}\boldsymbol{u}$$

Gram matrix
(matrix of inner products!)

Can we use kernel here?
Just like we did for kernel regression?

it is called kernel PCA!

Why? ---- Linear is not enough



Issue:
projection on any 1D line will cause mixing of data from different classes.

We need nonlinear projection

Intuition:

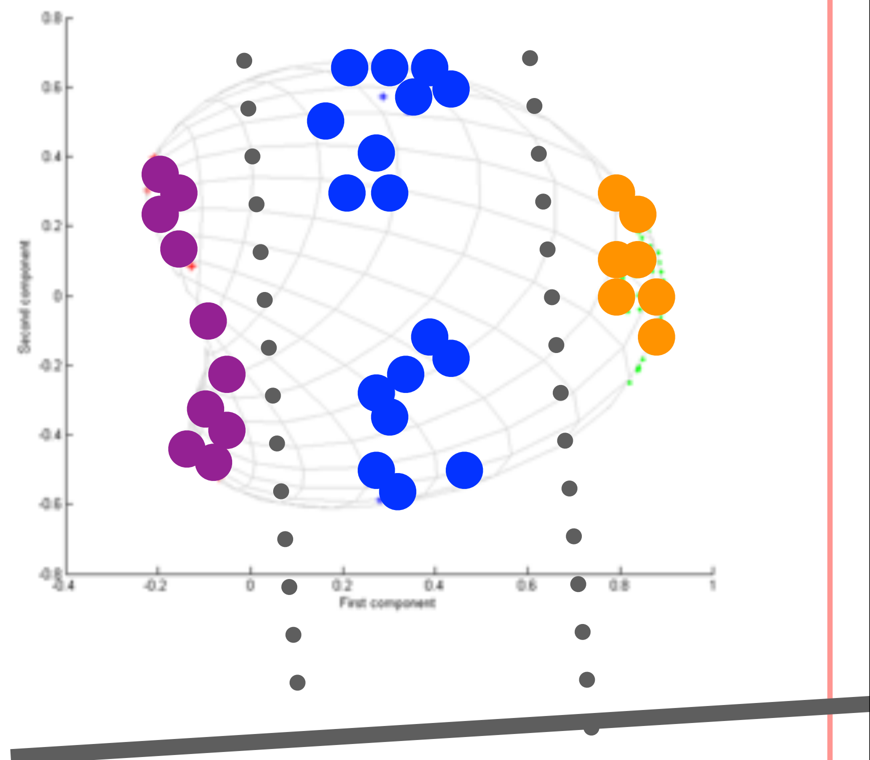
Find a good space through

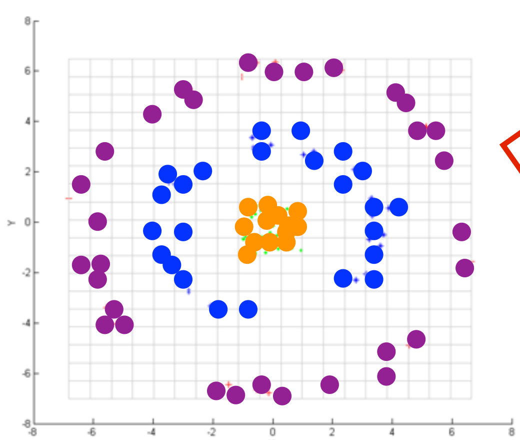
nonlinear mapping;

Then do linear projection

(as in PCA)

✓ Data separated!





How to find this
nonlinear mapping?

Details of kernel PCA

Derivation

see note

Demo