Dimensionality Reduction

Dr.Ratnakar Dash

Department of Computer Science and Engineering
NIT Rourkela

* Slides source: Pattern Recognition by Prof. Olga Veksler, Western University

Table of Contents

Complexity

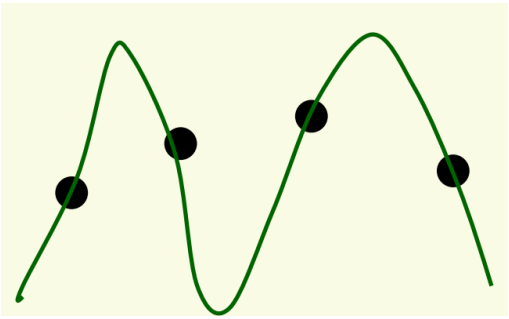
- Complexity (running time) increases with dimension d
- A lot of methods have at least $O(nd^2)$ complexity, where n is the number of samples
- For example if we need to estimate covariance matrix
- So as d becomes large, $O(nd^2)$ complexity may be too costly

Overfitting

- if d is large, n, the number of samples, may be too small for accurate parameter estimation
- For example, covariance matrix has d^2 parameters:

$$\sum = \begin{bmatrix} \sigma_1^2 & .. & \sigma_{1d} \\ . & .. & . \\ \sigma_{1d} & .. & \sigma_d^2 \end{bmatrix}$$

- For accurate estimation, n should be much bigger than d^2
- Otherwise model is too complicated for the data

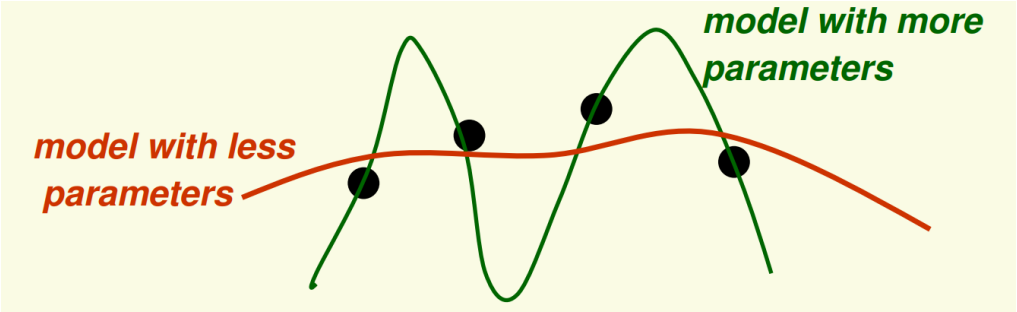


Contd..

- Paradox: if $n < d^2$ we are better off assuming that features are uncorrelated, even if we know this assumption is wrong
- In this case, the covariance matrix has only d parameters

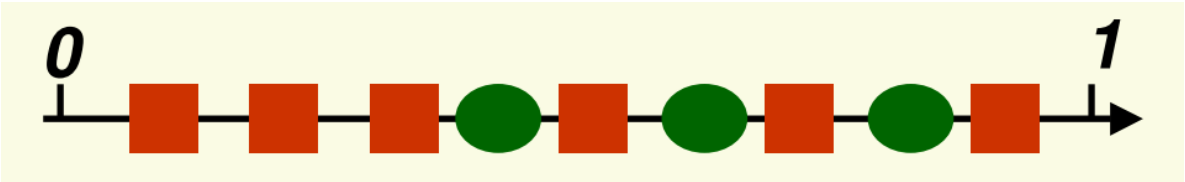
$$\sum = \begin{bmatrix} \sigma_1^2 & .. & 0 \\ . & .. & . \\ 0 & .. & \sigma_d^2 \end{bmatrix}$$

- We are likely to avoid overfitting because we fit a model with less parameters:



Number of Samples

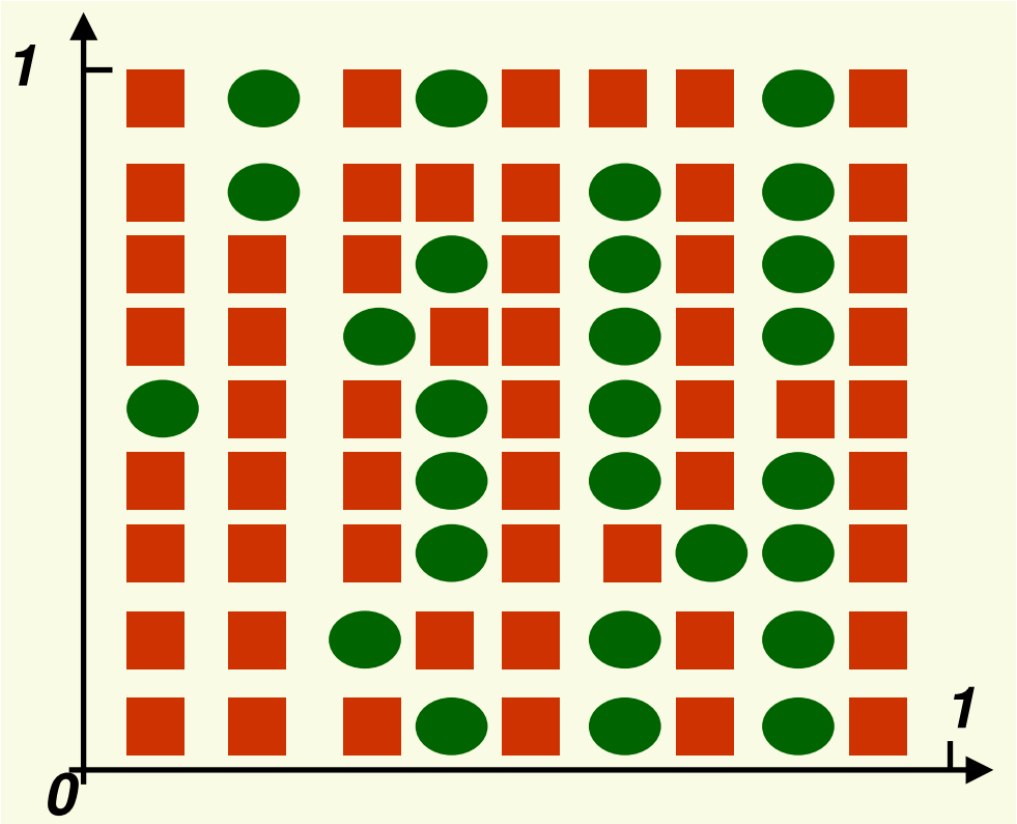
- Suppose we want to use the nearest neighbor approach with k = 1 (1NN)
- Suppose we start with only one feature



- This feature is not discriminative, i.e. it does not separate the classes well
- We decide to use 2 features. For the 1NN method to work well, need a lot of samples, i.e. samples have to be dense
- To maintain the same density as in 1D (9 samples per unit length), how many samples do we need?

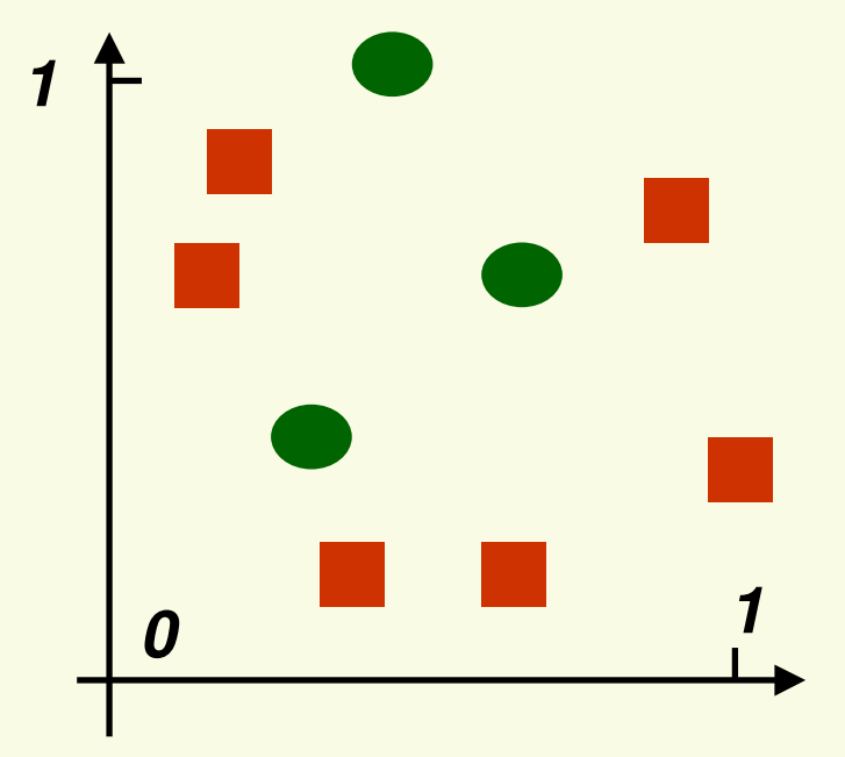
Contd..

- We need 9^2 samples to maintain the same density as in 1D



Contd..

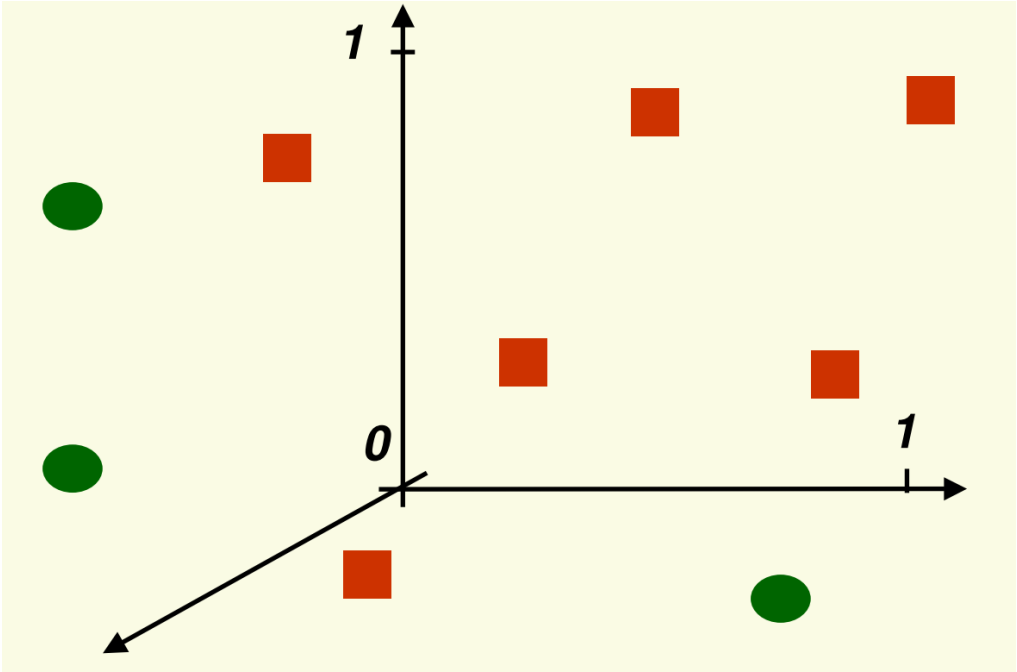
- Of course, when we go from 1 feature to 2, no one gives us more samples, we still have 9



- This is way too sparse for 1NN to work well

Contd..

- Things go from bad to worse if we decide to use 3 features:



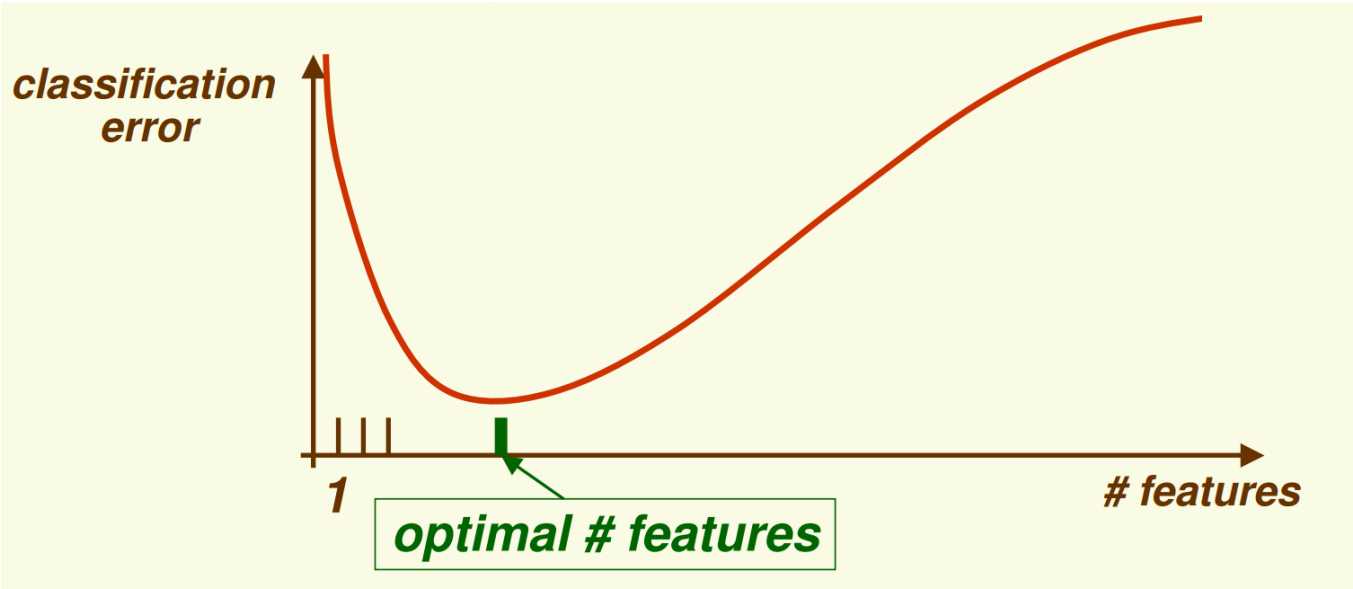
- If 9 was dense enough in 1D, in 3D we need $9^3 = 729$ samples

Contd..

- In general, if n samples is dense enough in 1D
- Then in d dimensions we need n^d samples!
- And n^d grows really really fast as a function of d
- Common pitfall:
  - If we can't solve a problem with a few features, adding more features seems like a good idea
  - However the number of samples usually stays the same
  - The method with more features is likely to perform worse instead of expected better

Contd..

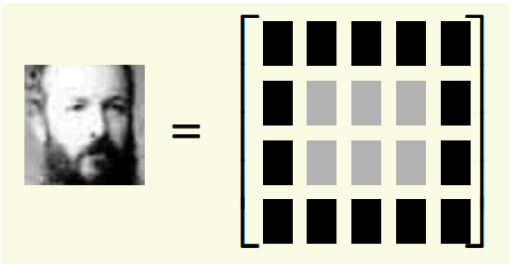
- For a fixed number of samples, as we add features, the graph of classification error:



- Thus for each fixed sample size n, there is the optimal number of features to use

Contd..

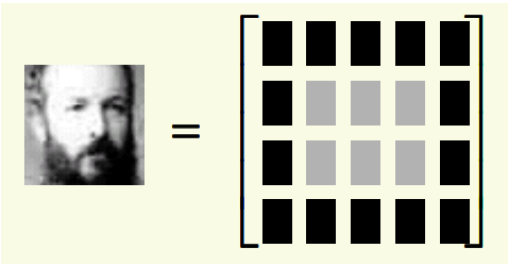
- We should try to avoid creating lot of features
- Often no choice, problem starts with many features
- Example: Face Detection
  - One sample point is k by m array of pixels



  - Feature extraction is not trivial, usually every pixel is taken as a feature
  - Typical dimension is 20 by 20 = 400
  - Suppose 10 samples are dense enough for 1 dimension. Need only 10^{400} samples

Contd..

- Face Detection, dimension of one sample point is km



- The fact that we set up the problem with km dimensions (features) does not mean it is really a km-dimensional problem
- Space of all k by m images has km dimensions
- Space of all k by m faces must be much smaller, since faces form a tiny fraction of all possible images
- Most likely we are not setting the problem up with the right features
- If we used better features, we are likely need much less than km-dimensions

Dimentionality Reduction

- High dimensionality is challenging and redundant
- It is natural to try to reduce dimensionality
- Reduce dimensionality by feature combination: combine old features x to create new features y

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \end{bmatrix} \rightarrow \left(\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \end{bmatrix} \right) = \begin{bmatrix} y_1 \\ . \\ . \\ x_k \end{bmatrix} = y$$

with $k < d$

- For example

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \end{bmatrix} = y$$

- Ideally, the new vector y should retain from x all information important for classification

Contd..

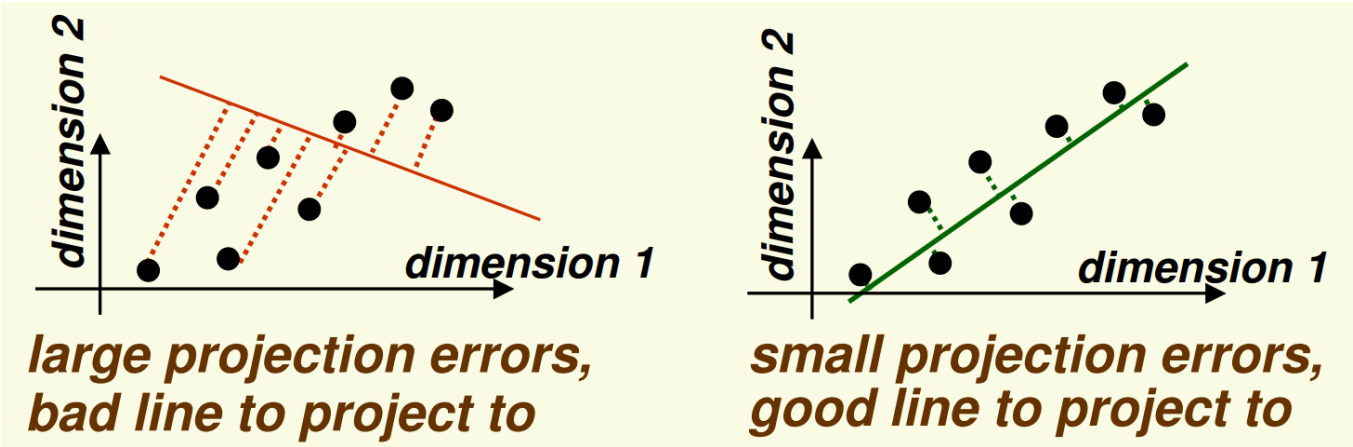
- The best f(x) is most likely a non-linear function
- Linear functions are easier to find though
- For now, assume that f(x) is a linear mapping
- Thus it can be represented by a matrix W:

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ x_d \end{bmatrix} \Rightarrow W \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_d \end{bmatrix} = \begin{bmatrix} w_{11} & .. & w_{1d} \\ . & .. & . \\ w_{k1} & .. & w_{kd} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_k \end{bmatrix}$$

with $k < d$

Principle Component Analysis (PCA)

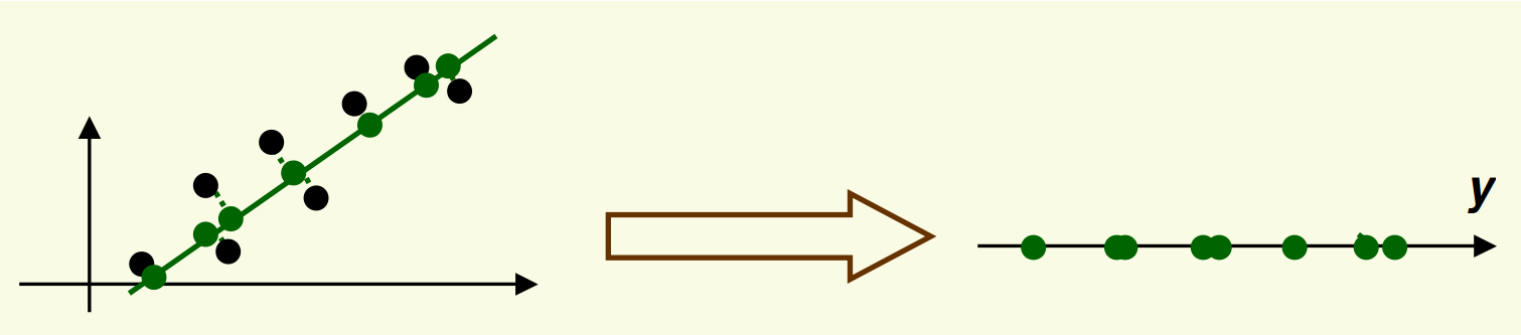
- Main idea: seek most accurate data representation in a lower dimensional space
- Example in 2-D
  - Project data to 1-D subspace (a line) which minimize the projection error



- Notice that the the good line to use for projection lies in the direction of largest variance

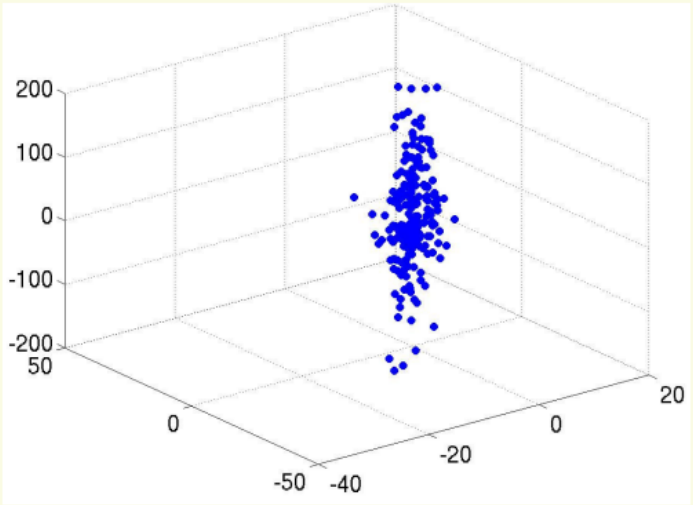
PCA

- After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector y

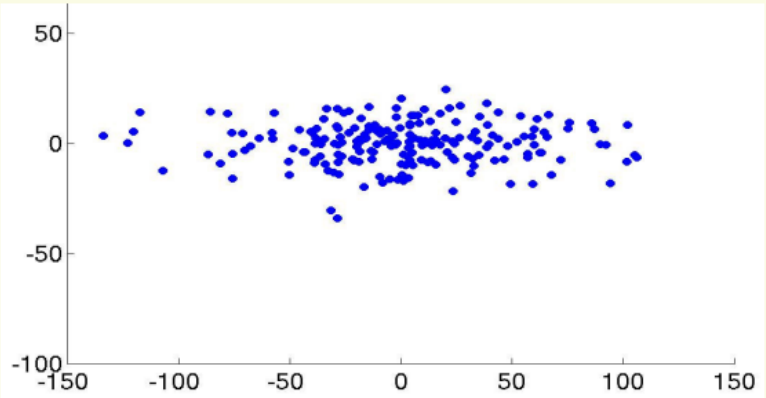


- Note that new data y has the same variance as old data x in the direction of the green line
- PCA preserves largest variances in the data. We will prove this statement, for now it is just an intuition of what PCA will do

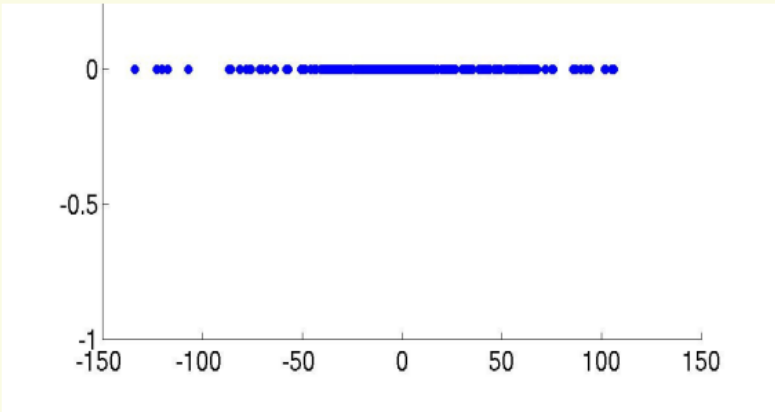
PCA: Approximation of Elliptical Cloud in 3D



best 2D approximation

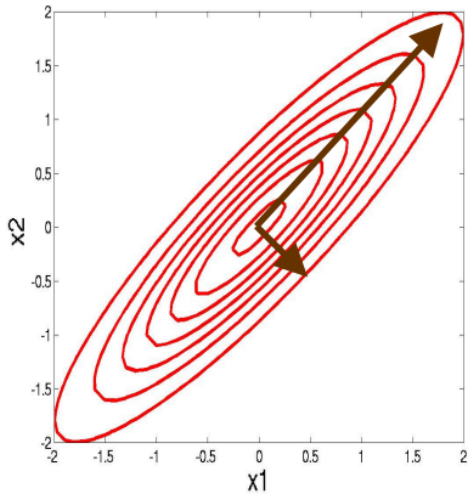


best 1D approximation



PCA

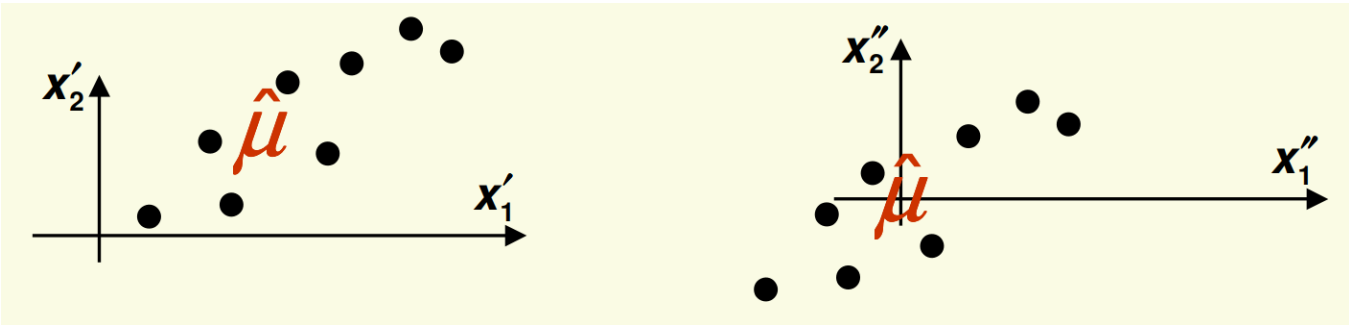
- What is the direction of largest variance in data?
- Recall that if x has multivariate distribution $N(\mu, \Sigma)$, direction of largest variance is given by eigenvector corresponding to the largest eigenvalue of Σ



- This is a hint that we should be looking at the covariance matrix of the data (note that PCA can be applied to distributions other than Gaussian)

PCA Derivation

- Before PCA, subtract sample mean from the data
  $x - \frac{1}{n}\sum_{i=1}^{n} x_i = x - \hat{\mu}$
- The new data has zero mean:
  $E(X - E(X)) = E(X) - E(X) = 0$
- All we did is change the coordinate system



- Another way to look at it: first step of getting y is to subtract the mean of x
  $x \rightarrow y = f(x) = g(x - \hat{\mu})$

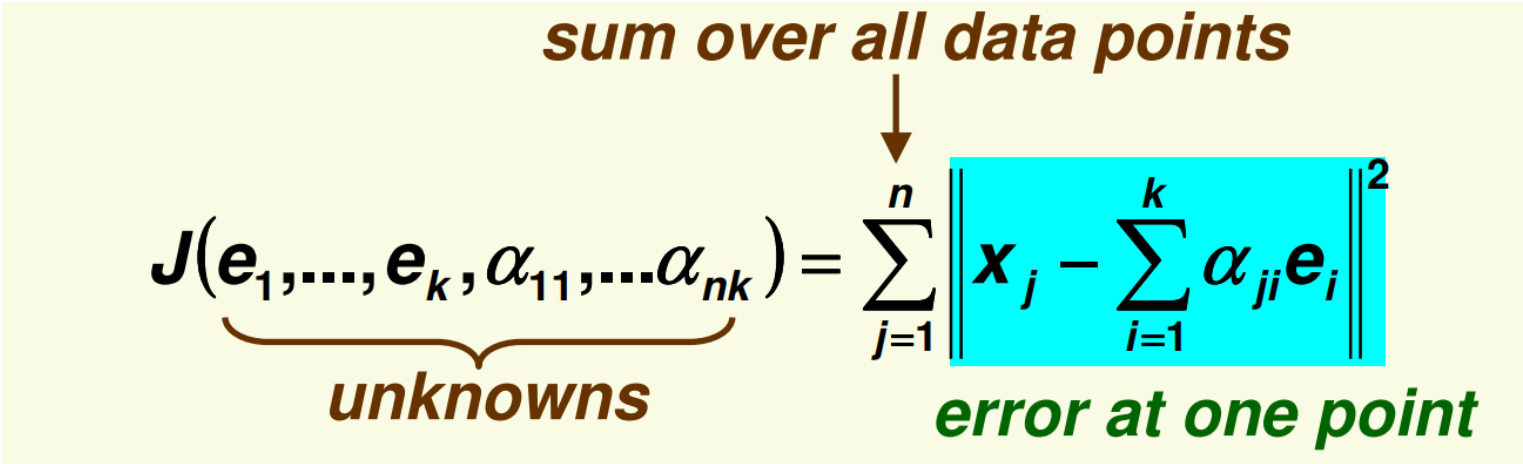
Contd..

- We want to find the most accurate representation od data $D = x_1, x_2, ..., x_n$ in some subspace W which has dimension $k < d$
- Let $(e_1, e_2, ..., e_k)$ be the orthonormal basis for W. Any vector in W can be written as $\sum_{i=1}^{k} \alpha_i e_i$
- Error this representation:

$$error = \left\| x_1 - \sum_{i=1}^{k} \alpha_{1i} e_i \right\|^2$$

Contd..

- To find the total error, we need to sum over all x_j's
- Any x_j can be written as $\sum_{i=1}^{k} \alpha_{ji} e_i$
- Thus the total error for representation of all data D is:



Contd..

- To minimize J, need to take partial derivatives and also enforce constraint that $e_1, e_2, ...e_k$ are orthogonal

$$J(e_1, ..., e_k, \alpha_{11}, ..., \alpha_{nk}) = \sum_{j=1}^{n} \left\| x_j - \sum_{i=1}^{k} \alpha_{ji} E_i \right\|^2$$

- After simplifying J and differentiation

$$J(e_1, ..., e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t S e_i$$

- Where $S = \sum_{j=1}^{n} x_j x_j^t$
- S is a scatter matrix, it is just n-1 times the sample covariance matrix

Contd..

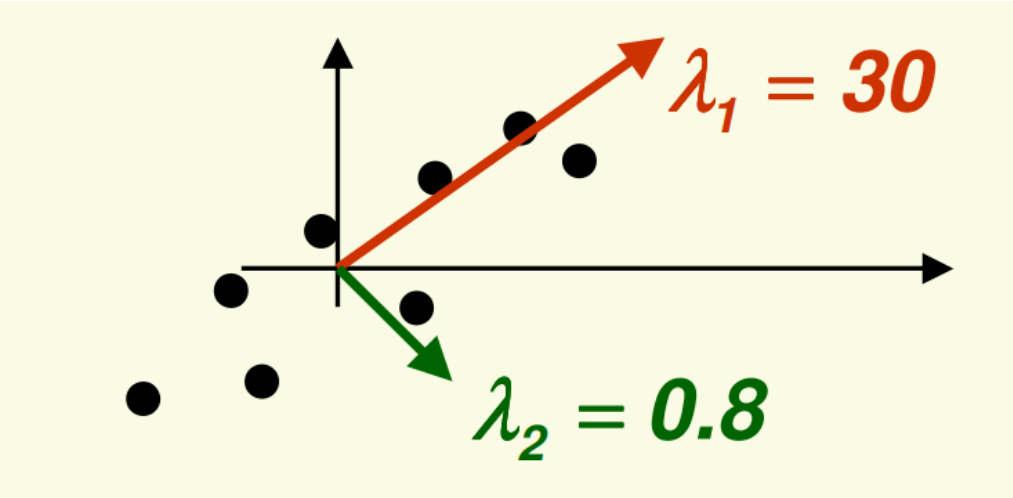
- Further simplification and differentiation will give

$$J(e_1, ..., e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} \lambda_i \|e_i\|^2 = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} \lambda_i$$

- Thus to minimize J take for the basis of W the k eigenvectors of S corresponding to the k largest eigenvalues

Contd..

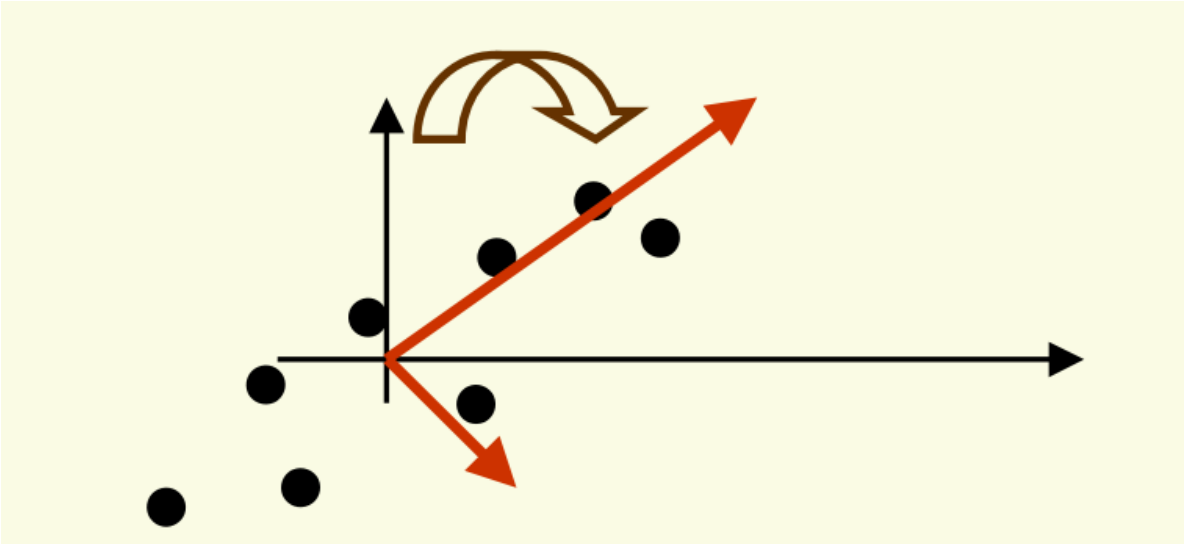
- The larger the eigenvalue of S, the larger is the variance in the direction of corresponding eigenvector



- This result is exactly what we expected: project x into subspace of dimension k which has the largest variance
- This is very intuitive: restrict attention to directions where the scatter is the greatest

Contd..

- Thus PCA can be thought of as finding new orthogonal basis by rotating the old axis until the directions of maximum variance are found



PCA as Data Approximation

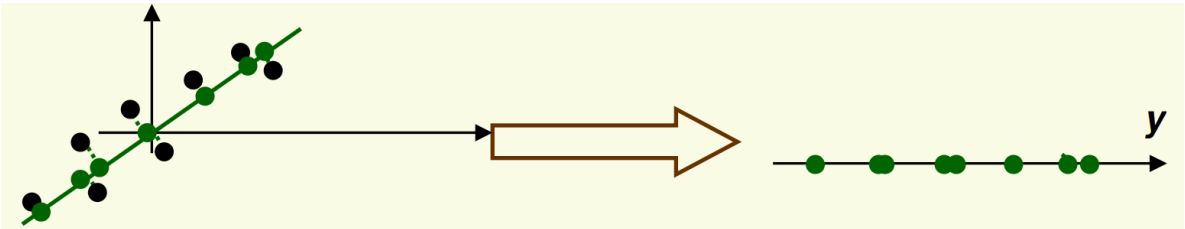
- Let $e_1, e_2, , e_d$ be all d eigenvectors of the scatter matrix S, sorted in order od decreasing corresponding eigenvalue
- Without any approximation, for any sample X_i:

$$\boldsymbol{x}_i = \sum_{j=1}^{d} \alpha_j \boldsymbol{e}_j = \underbrace{\alpha_1 \boldsymbol{e}_1 + \ldots + \alpha_k \boldsymbol{e}_k}_{\text{approximation of } x_i} + \overbrace{\alpha_{k+1} \boldsymbol{e}_{k+1} \ldots + \alpha_d \boldsymbol{e}_d}^{\text{error of approximation}}$$

- coefficients $\alpha_m = x^t{}_i e_m$ are called principle components
  - The larger k, the better is the approximation
  - Components are arranged in order of importance more important components come first
- Thus PCA takes the first k most important components of X_i as an approximation to x_i

PCA: Last Step

- Now we know how to project the data
- Last step is to change the coordinates to get final k-dimensional vector y



- Let Matrix $E = [e_1 .... e_k]$
- Then the coordinate transformation is $y = E^t x$
- Under E^t, the eigenvectors become the standard basis:

$$E^t e_i = \begin{bmatrix} e_1 \\ . \\ e_i \\ . \\ e_i \end{bmatrix} \quad e_i = \begin{bmatrix} 0 \\ . \\ 1 \\ . \\ 0 \end{bmatrix}$$

Recipe for Dimension Reduction with PCA

Data $D = x_1, x_2, , x_n$. Each x_i is a d-dimensional vector. Wish to use PCA to reduce dimension to k

1. Find the sample mean $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2. Subtract sample mean from the data $Z_i = x_i - \hat{\mu}$
3. Compute the scatter matrix $S = \sum_{i=1}^{n} z_i z_i^t$
4. Compute eigenvectors $e_1, e_2...., e_k$ corresponding to the k largest eigenvalues of S
5. Let $e_1, e_2...., e_k$ be the columns of matrix $E = [e_1...., e_k]$
6. The desired y which is the closest approximation to x is $y = E^t z$

PCA Example

- Let D = (1,2),(2,3),(3,2),(4,4),(5,4),(6,7),(7,6),(9,7)

$$x = \begin{bmatrix} 1 & 2 \\ . & . \\ 9 & 7 \end{bmatrix} = \begin{bmatrix} x_1 \\ . \\ x_8 \end{bmatrix}$$

- Mean $\mu = mean(X) = [4.64.4]$
- Subtract mean from data to get new data array Z

$$Z = X - \begin{bmatrix} \mu \\ . \\ \mu \end{bmatrix} = X - repmat(\mu, 8, 1) = \begin{bmatrix} -3.6 & -4.4 \\ . & . \\ 4.4 & 2.6 \end{bmatrix}$$

- Compute The scatter matrix S

$$S = 7 * cov(Z) = [-3.6 - 4.4] \begin{bmatrix} -3.6 \\ -4.4 \end{bmatrix} + ... + [4.42.6] \begin{bmatrix} 4.4 \\ 2.6 \end{bmatrix} =$$

$$\begin{bmatrix} 57 & 40 \\ 40 & 34 \end{bmatrix}$$

PCA Example

- Use [V,D] =eig(S) to get eigenvalues and eigenvectors of

  $\lambda_1 = 87$ and $e_1 = \begin{bmatrix} -0.8 \\ -0.6 \end{bmatrix}$

  $\lambda_2 = 3.8$ and $e_2 = \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}$

- Projection to 1D space in the direction of e_1

  $Y = e_1^t Z^t = \left([-0.8 - 0.6] \begin{bmatrix} -3.6 & \dots & 4.4 \\ -4.4 & \dots & 2.6 \end{bmatrix} \right) =$

  $[4.3.... - 5.1] = [y_1....y_8]$

Drawbacks of PCA

- PCA was designed for accurate data representation, not for data classification
  - Preserves as much variance in data as possible
  - If directions of maximum variance is important for classification, will work
- However the directions of maximum variance may be useless for classification