Reducing Dimensions/Features

Dimensionality/Feature reduction

- Many modern data domains involve huge numbers of features / dimensions
  - Documents: thousands of words, millions of bigrams
  - Images: thousands to millions of pixels
  - Genomics: thousands of genes, millions of DNA polymorphisms

Why reduce dimensions/features?

- High dimensionality has many costs
  - Redundant and irrelevant features degrade performance of some ML algorithms
  - Difficulty in interpretation and visualization
  - Computation may become infeasible
    - what if your algorithm scales as $O(n^3)$?

Feature Reduction

- How to find the ‘best’ low dimension space that conveys maximum useful information?
- One answer: Find “Principal Components”

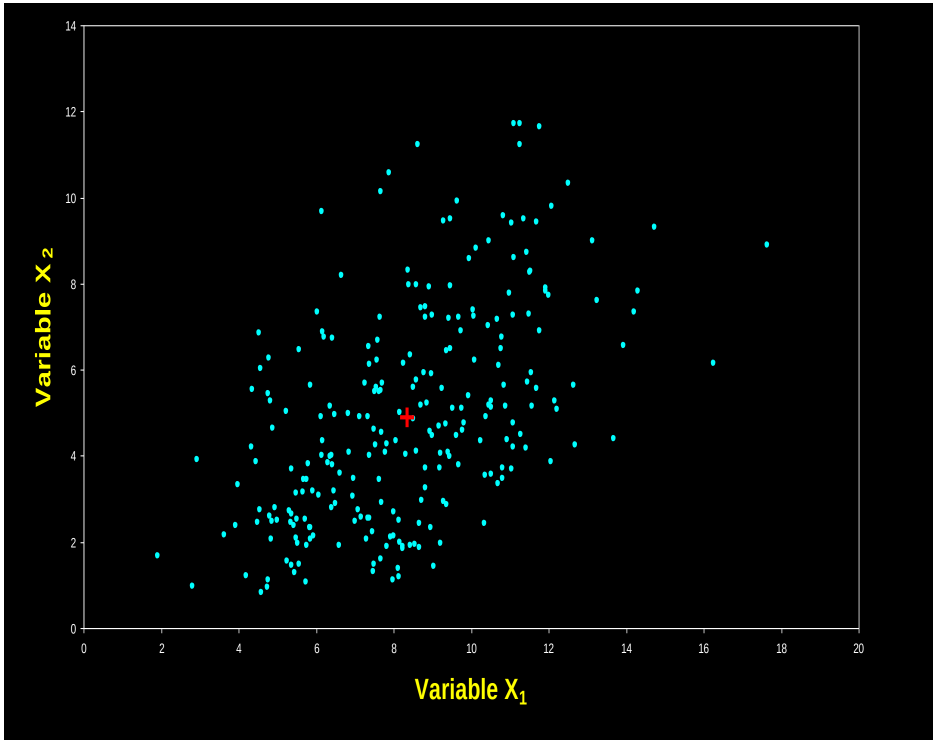
Principal Components

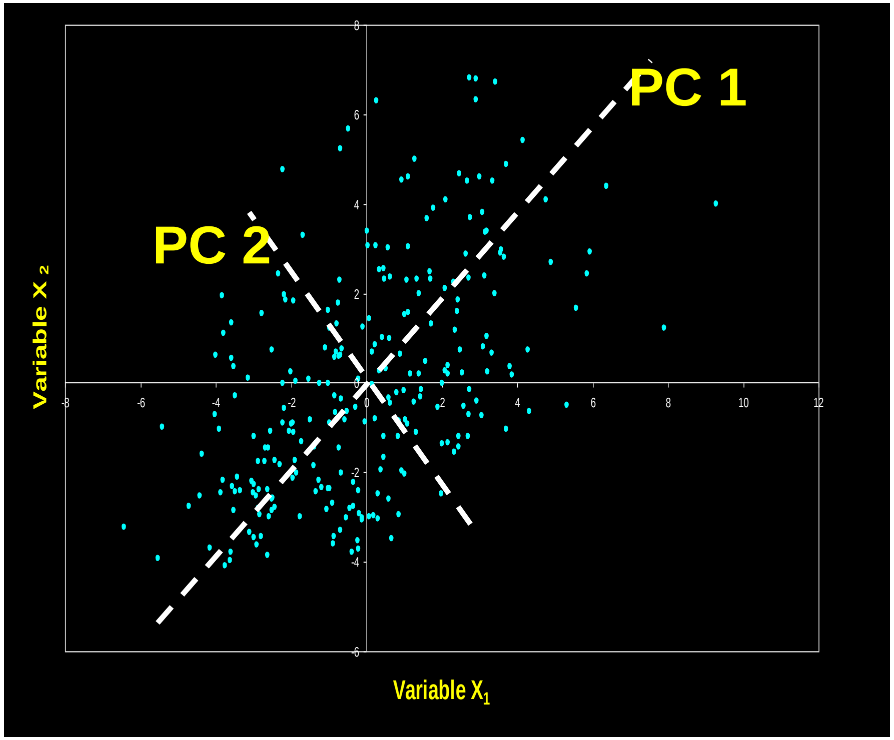
Find the new co-ordinate axes that captures the maximum variance/co-variance available in the given dataset?

The first principal component/axis captures the maximum variation and second principal component captures the maximum residual variation etc.,

Geometric Interpretation of PCA

- The goal is to rotate the axes of the p-dimensional space to new positions (principal axes) that have the following properties:
  - ordered such that principal axis 1 has the highest variance, axis 2 has the next highest variance, ... , and axis p has the lowest variance
  - covariance among each pair of the principal axes is zero (the principal axes are uncorrelated).





Note: Each principal axis is a linear combination of the original two variables

Why does it works?

Total variance along X1 and X2 = Variance of points along X1 + Variance of points along X2

Total variance along PC1 and PC2 = Variance of points along PC1 + Variance of points along PC2

Why does it works?

Total variance will be same among both bases but the variance is captured heavily among first few principal components.

Hence, we can ignore the latter principal components without loss of information.

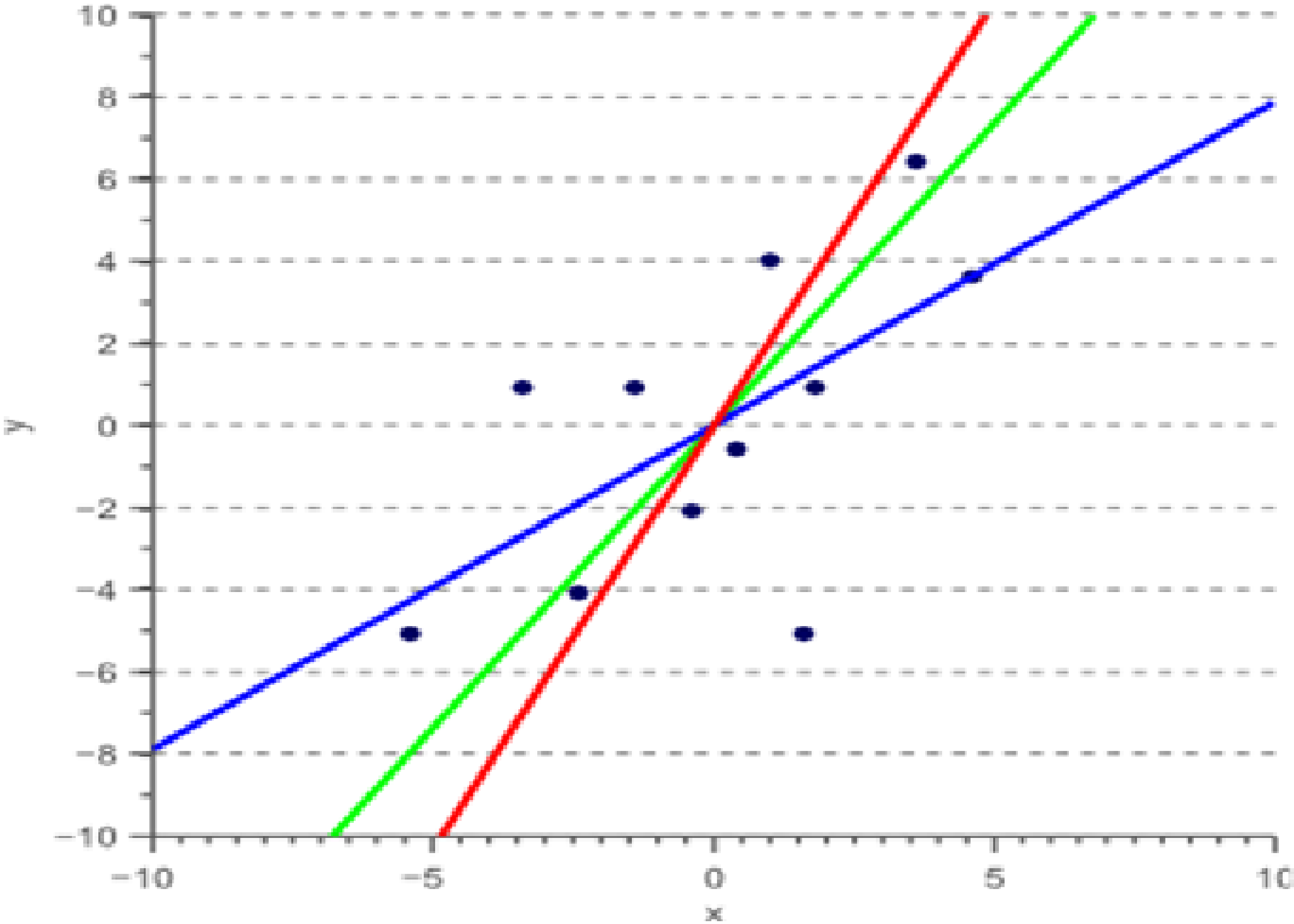
Assumption of PCA

PCA makes sense only if there exists high correlation among some of the features.

PCA problem

How do you find the axis that captures the variances in descending order (or) How do you compute the principal components/axis?

Meaning of capturing variance



Formulating PCA as Optimization: Maximizing variance among vector v

Maximize var(v) = $v^T A v$
subject to $v^T v = 1$

var(v, λ) = $v^T A v - \lambda (v^T v - 1)$

Apply calculus and solve for V and λ

Solving PCA optimization Problem

$$f(v, \lambda) = v^T A v - \lambda (v^T v - 1)$$

$$\frac{\partial f}{\partial \lambda} = v^T v - 1$$

$$v^T v - 1 = 0$$

$$v^T v = 1$$

$$\frac{\partial f}{\partial v} = 2 A v - 2 \lambda v$$

$$2 A v - 2 \lambda v = 0$$

$$A v = \lambda v$$

Interpreting the solution of PCA

- It is evident tat the maximum occurs when v is one of the eigenvectors of covariance matrix A and λ is the corresponding eigenvalues.

Interpreting the solution of PCA

- Under this condition, the corresponding total projection is:

$$var(v) = v^T A\ v = v^T \lambda\ v = \lambda$$

- So the eigenvalue represents the variation captured along its corresponding eigenvector.
- The sum of all the eigenvalues i.e., total variance along new axes will be same as total variance along original axes.

Principal components

- If we arrange eigenvalues such that:

$$\boxed{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d}$$

with the corresponding eigenvectors $v_1, v_2, \ldots, v_d$ then

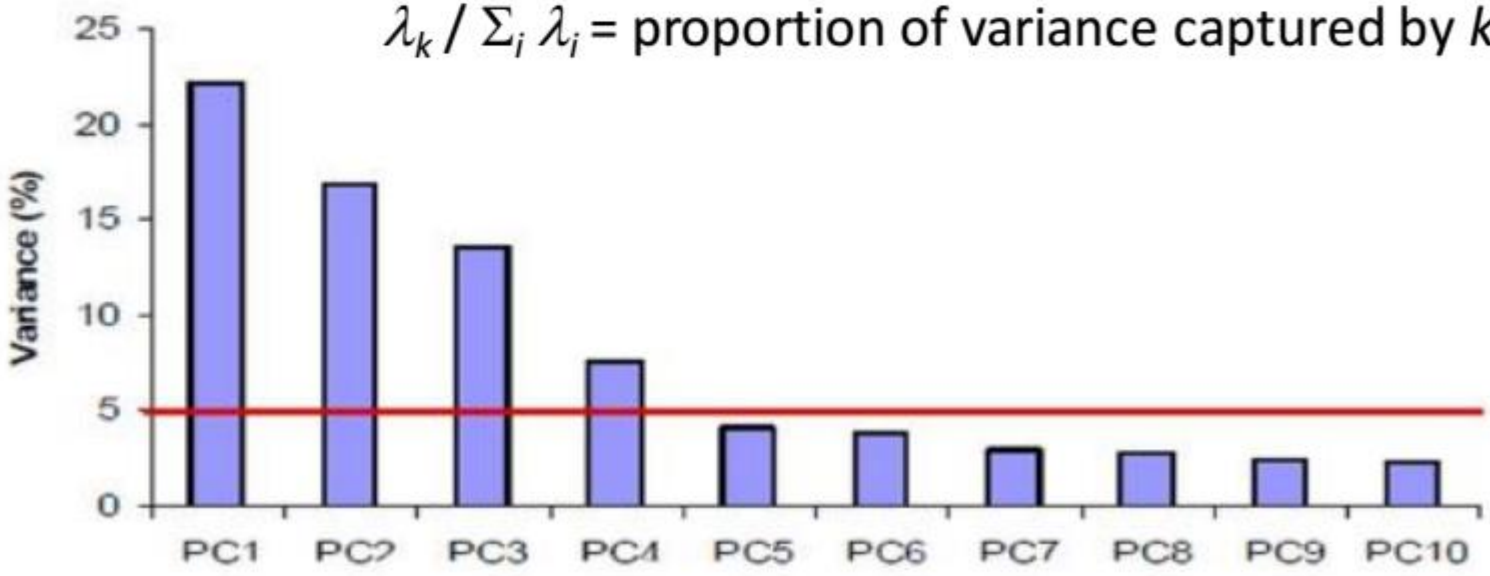
a) Max of var(v) is λ_1, which occurs at $v=v_1$
b) Min of var(v) is λ_d, which occurs at $v=v_d$

Principal Components

Eigenvector with largest eigenvalue λ_1 is 1st principal component (PC)

Eigenvector with k^{th} largest eigenvalue λ_k is k^{th} PC

$\lambda_k / \Sigma_i \lambda_i$ = proportion of variance captured by k^{th} PC



Numerical Example

Covariance Matrix - Example

Original Data

X =

X1	X2	X3
10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

Centered Data

A =

1.8	3.2	2
-6.2	-11.8	-6
-0.2	0.2	-1
0.8	3.2	2
3.8	5.2	3

Covariance Matrix

$Cov(X) = (A^T A)/n$ =

	X1	*X2*	*X3*
X1	14.2	25.3	13.5
X2	25.3	46.7	24.75
X3	13.5	24.75	13.5

Total Variation = 14.2 + 46.7 + 13.5 = 74.4

Eigenvalues and Eigenvectors

	X1	*X2*	*X3*
X1	14.2	25.3	13.5
X2	25.3	46.7	24.75
X3	13.5	24.75	13.5

Covariance Matrix

$\lambda_1 = 73.718$
$\lambda_2 = 0.384$
$\lambda_3 = 0.298$

Eigenvalues

Note: $\lambda_1 + \lambda_2 + \lambda_3 = 74.4$
(Equal to total variance in original dimensions)

Eigenvectors

Z =

Z1	Z2	Z3
0.434	0.900	-0.044
0.795	-0.406	-0.451
0.424	-0.161	0.891

Practical Algorithms

Approach1: Using co-variance matrix

1. Find the mean: $\mu = \frac{1}{n}\sum_{i=1}^{n}\mathrm{x}_i$
2. Compute the covariance matrix:

$$\mathbf{C} = \frac{1}{n}XX^T = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathrm{x}_i - \mu)^T$$

3. Find the eigenvalues of C and arrange them into descending order, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ with the corresponding eigenvectors $\{\mathrm{u}_1, \mathrm{u}_2, \cdots, \mathrm{u}_d\}$
4. The principal component matrix is:

$$\mathrm{U} = \begin{bmatrix} | & | & & | \\ \mathrm{u}_1 & \mathrm{u}_2 & \cdots & \mathrm{u}_d \\ | & | & & | \end{bmatrix}$$

Practical Issue: Scaling Up

- Covariance of the image data is BIG!
  - size of Σ = 32768 x 32768
  - finding eigenvector of such a matrix is slow.
- SVD comes to rescue!
  - Can be used to compute principal components
  - Efficient implementations available