ACC-HPC June 2025

UNSUPERVISED LEARNING & CLUSTERING

Machine Learning

Dr Kiran Waghmare
CDAC Mumbai

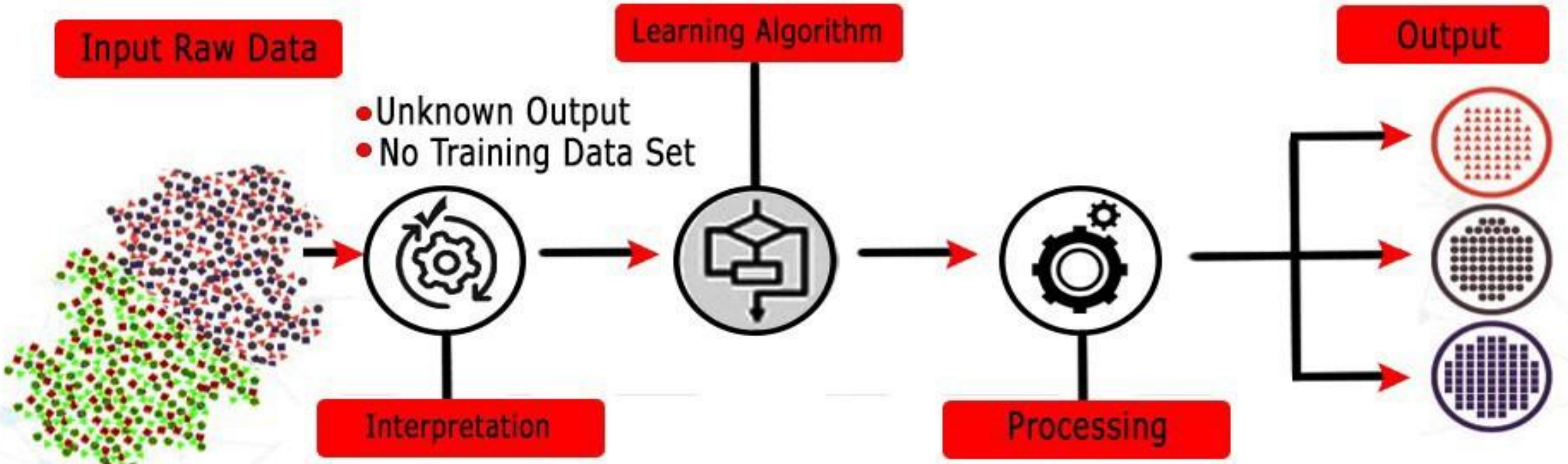
Agenda

- **Clustering**
- **K-Means**
- **Hierarchical**
- **DB-SCAN**

Machine learning:

- **Supervised vs Unsupervised.**
  - ***Supervised learning*** - the presence of the outcome variable is available to guide the learning process.
    - there **must** be a training data set in which the solution is already known.
  - ***Unsupervised learning*** - the outcomes are unknown.
    - cluster the data to reveal meaningful partitions and hierarchies

Unsupervised Learning



Clustering

Clustering:

- Unsupervised learning
- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for
- But: can get gibberish

Machine Learning: Clustering

By color

By shape

By size

etc...

Cluster by Type

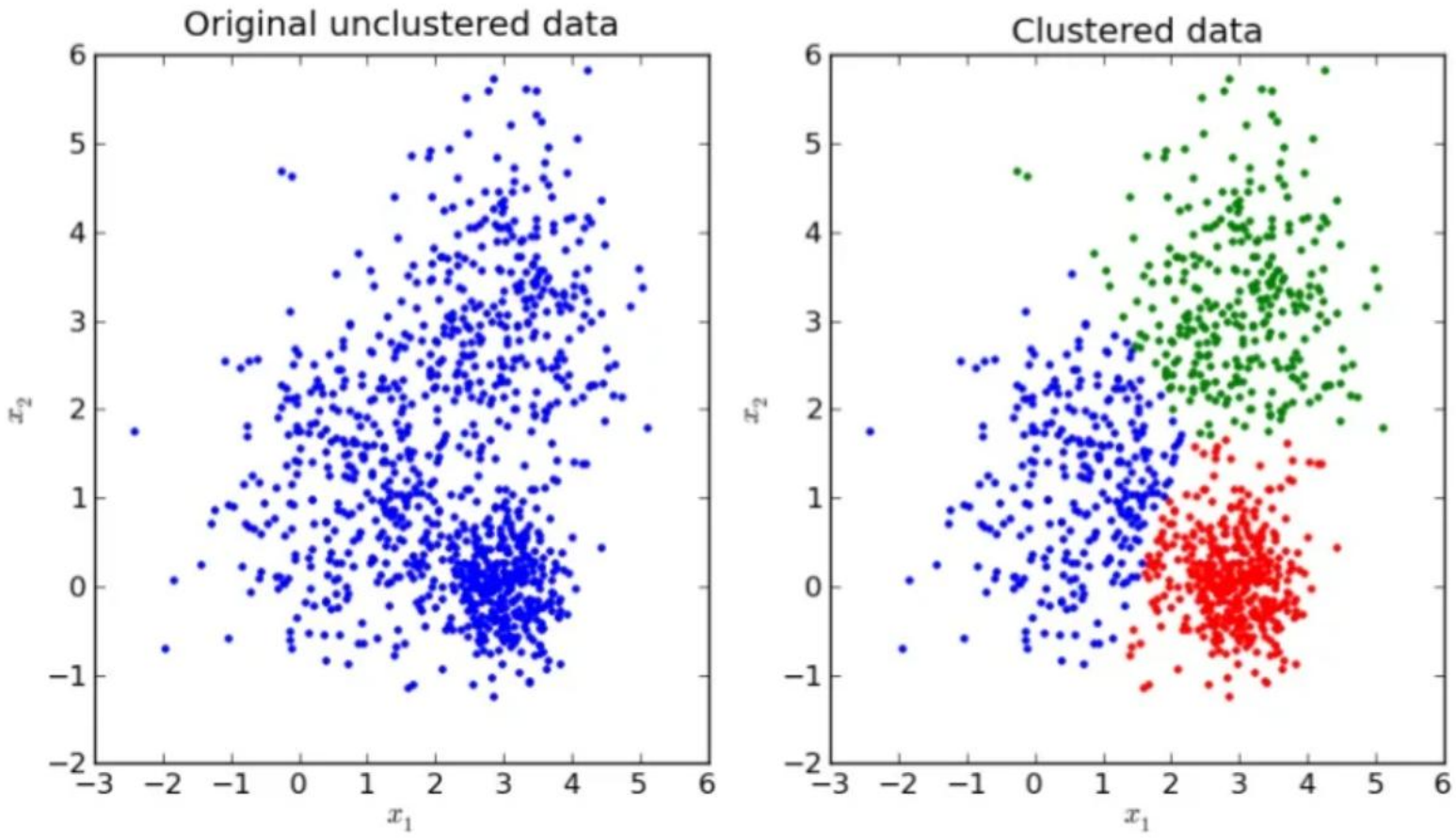
Clustering:

- Clustering is the task of gathering samples into groups of similar samples according to some predefined similarity or dissimilarity measure

sample

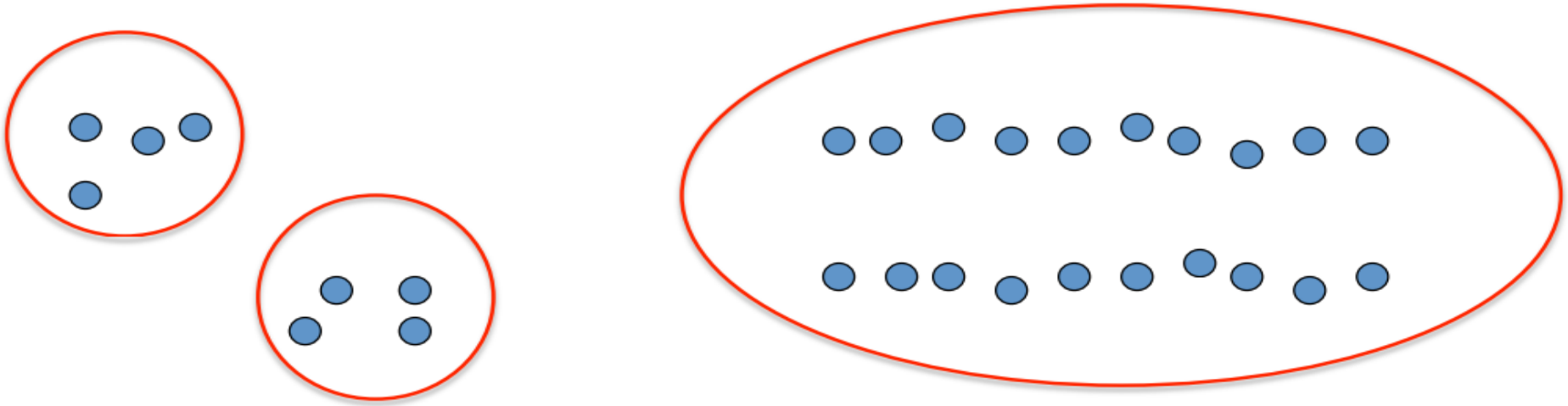
Cluster/group

- In this case clustering is carried out using the Euclidean distance as a measure.



Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



What is Similarity?

Slide based on one by Eamonn Keogh

Similarity is hard to define, but…
"*We know it when we see it*"

Partitioning Algorithms: Basic Concept

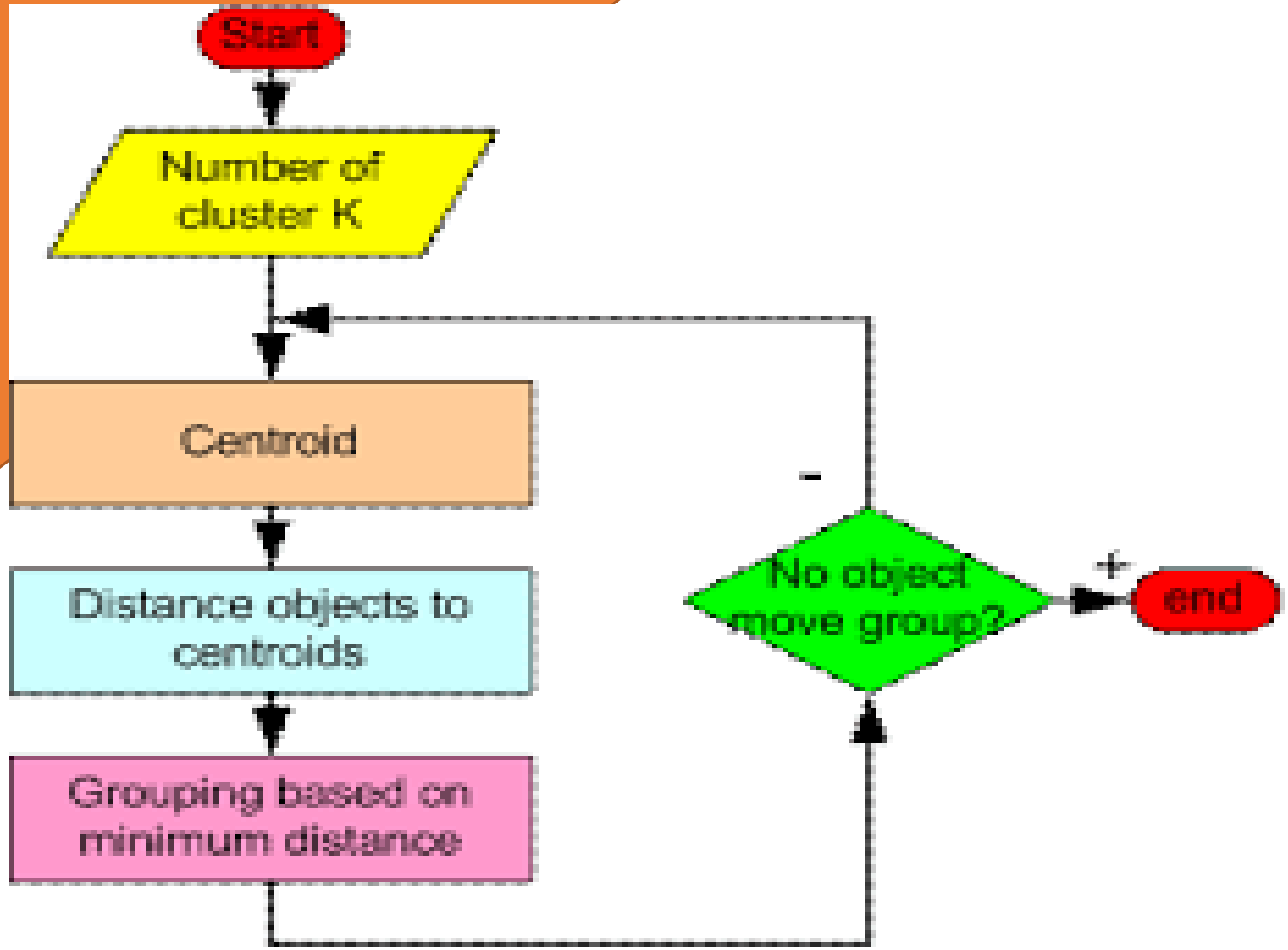
- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters, s.t., min sum of squared distance

$$\sum_{m=1}^{k}\sum_{t_{mi}\in Km}(C_m - t_{mi})^2$$

- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

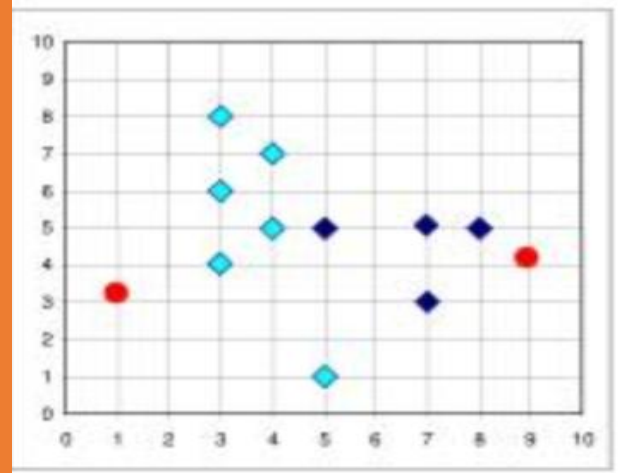
The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when no more new assignment


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

The *K-Means* Clustering Method

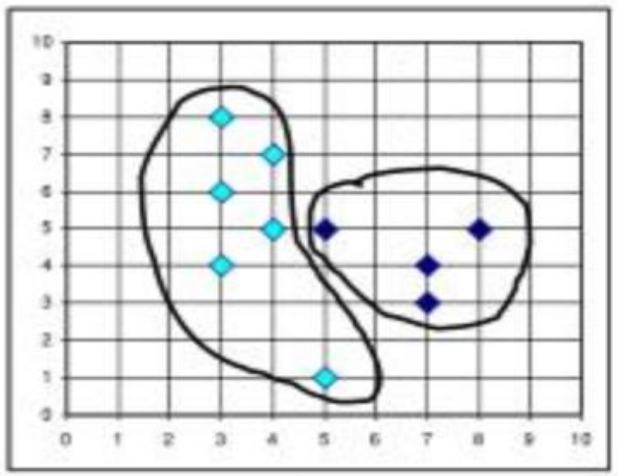
- Example



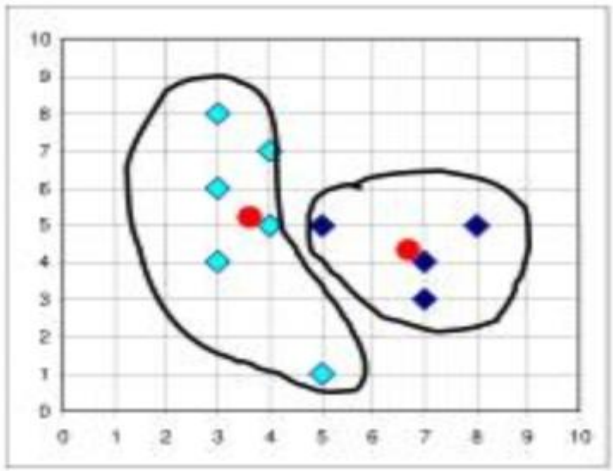
K=2

Arbitrarily choose K object as initial cluster center

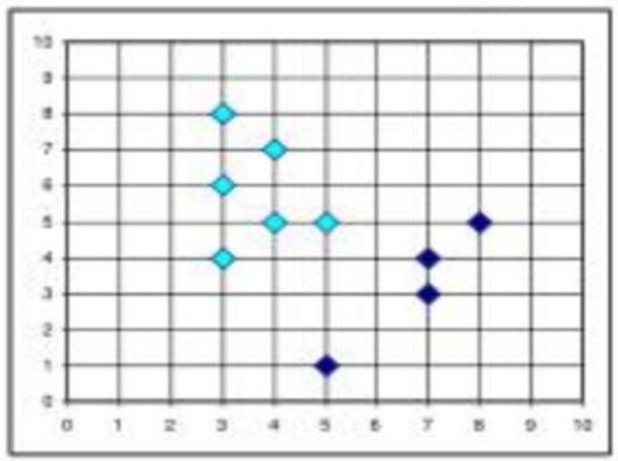
Assign each objects to most similar center



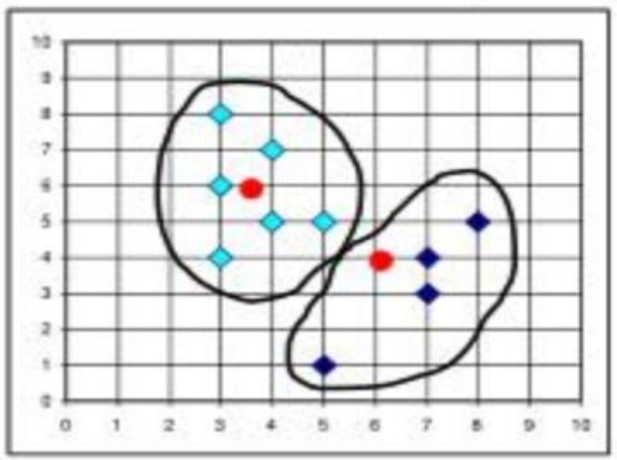
Update the cluster means



reassign



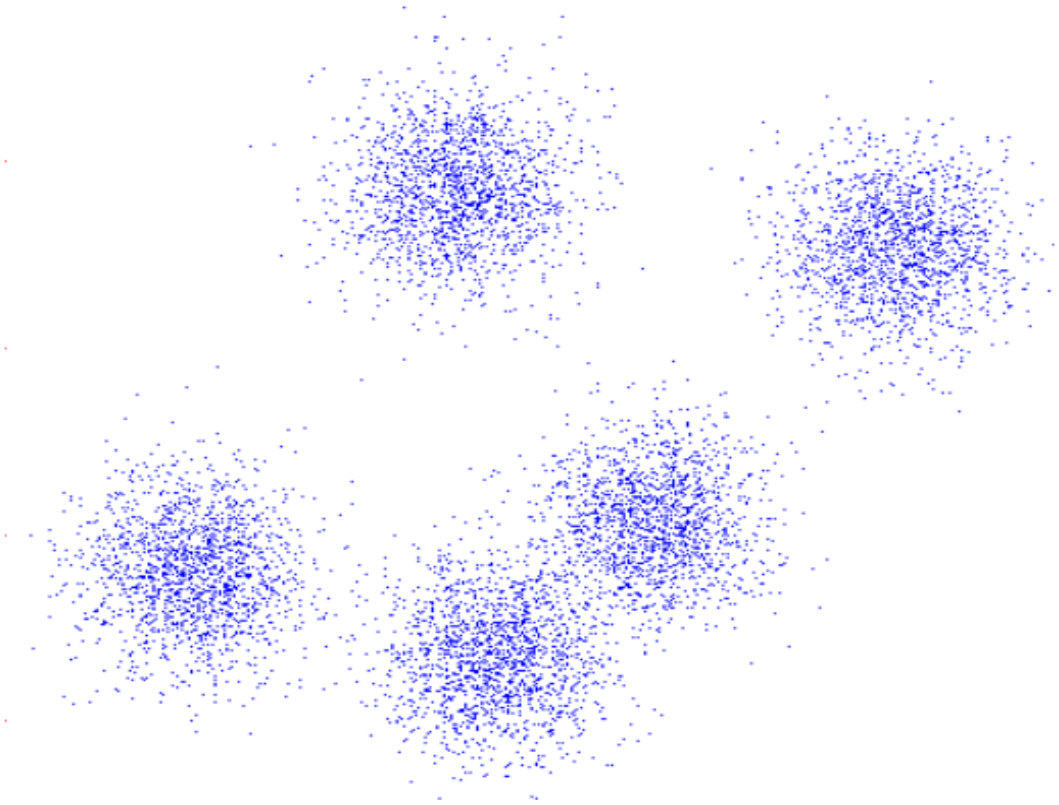
Update the cluster means



reassign

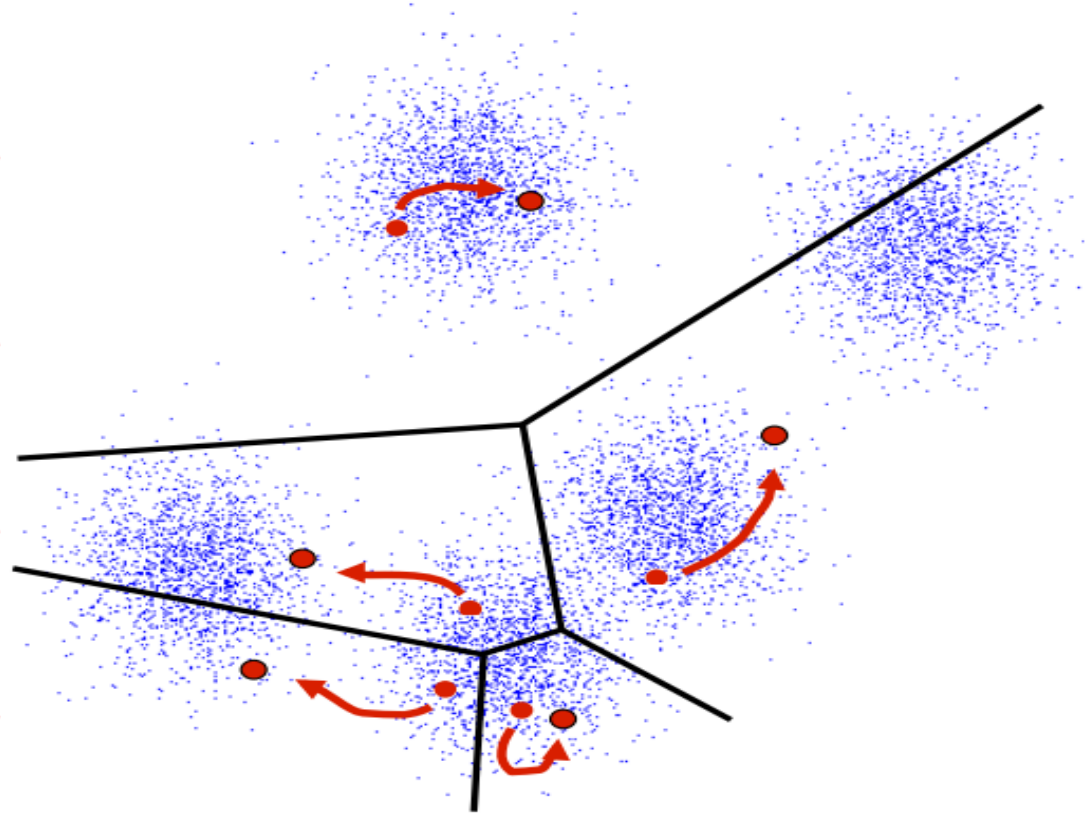
K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change

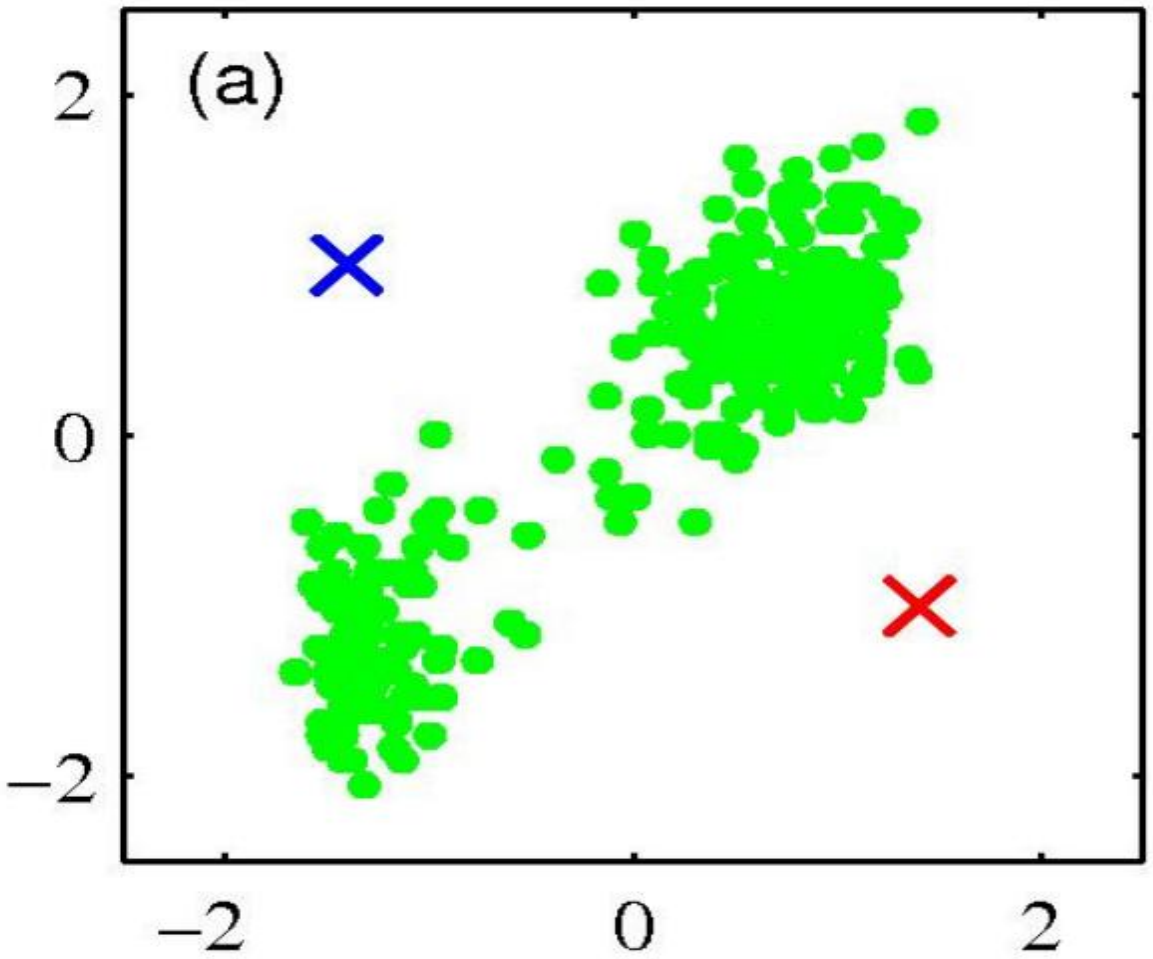


K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



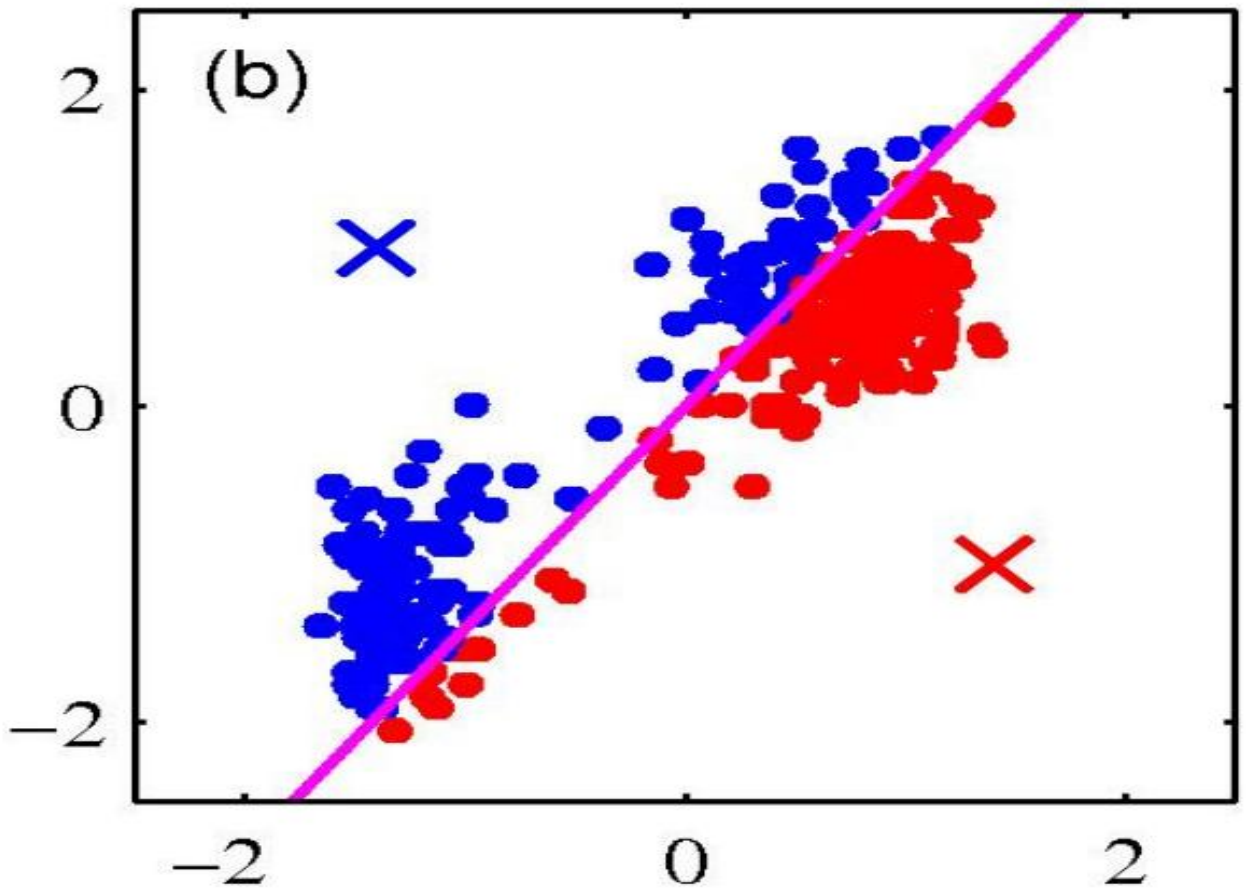
K-means clustering: Example



- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

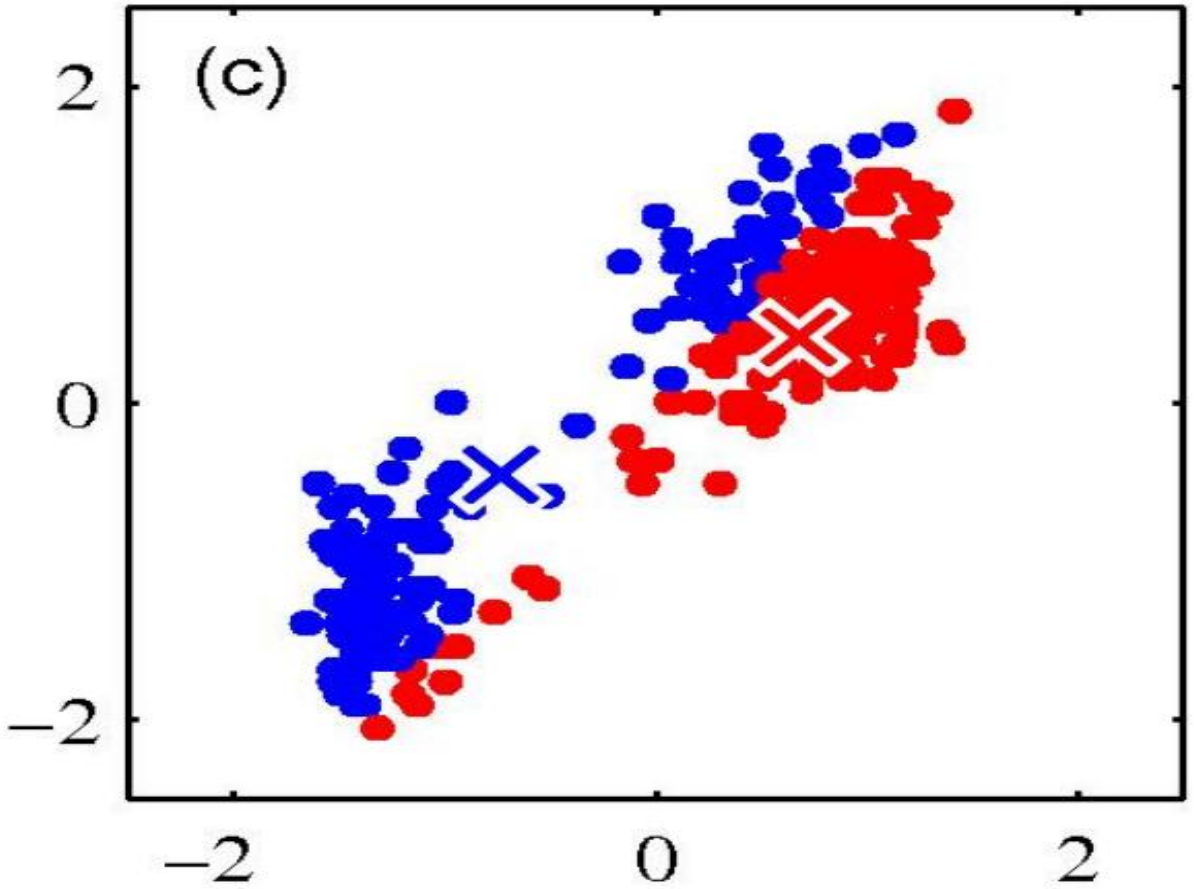
K-means clustering: Example



Iterative Step 1

- Assign data points to closest cluster center

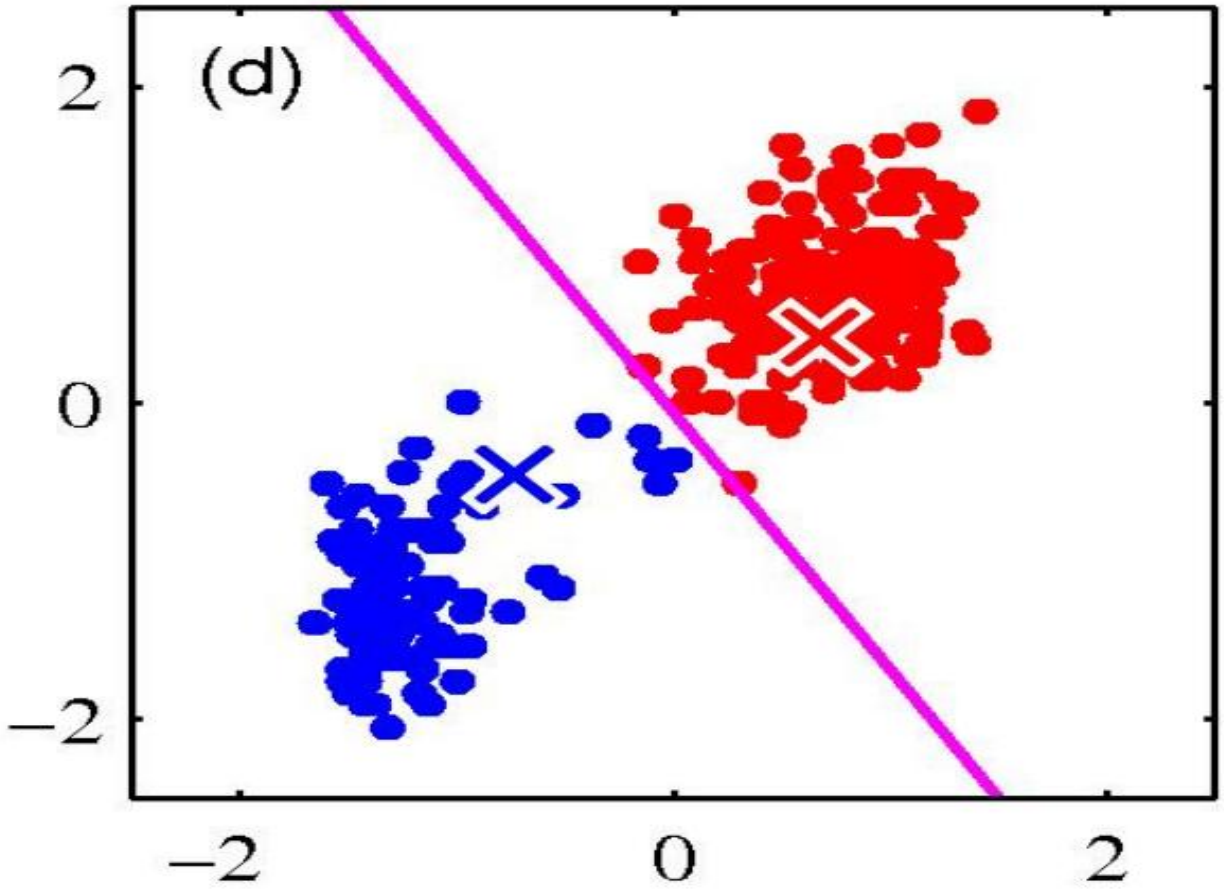
K-means clustering: Example



Iterative Step 2

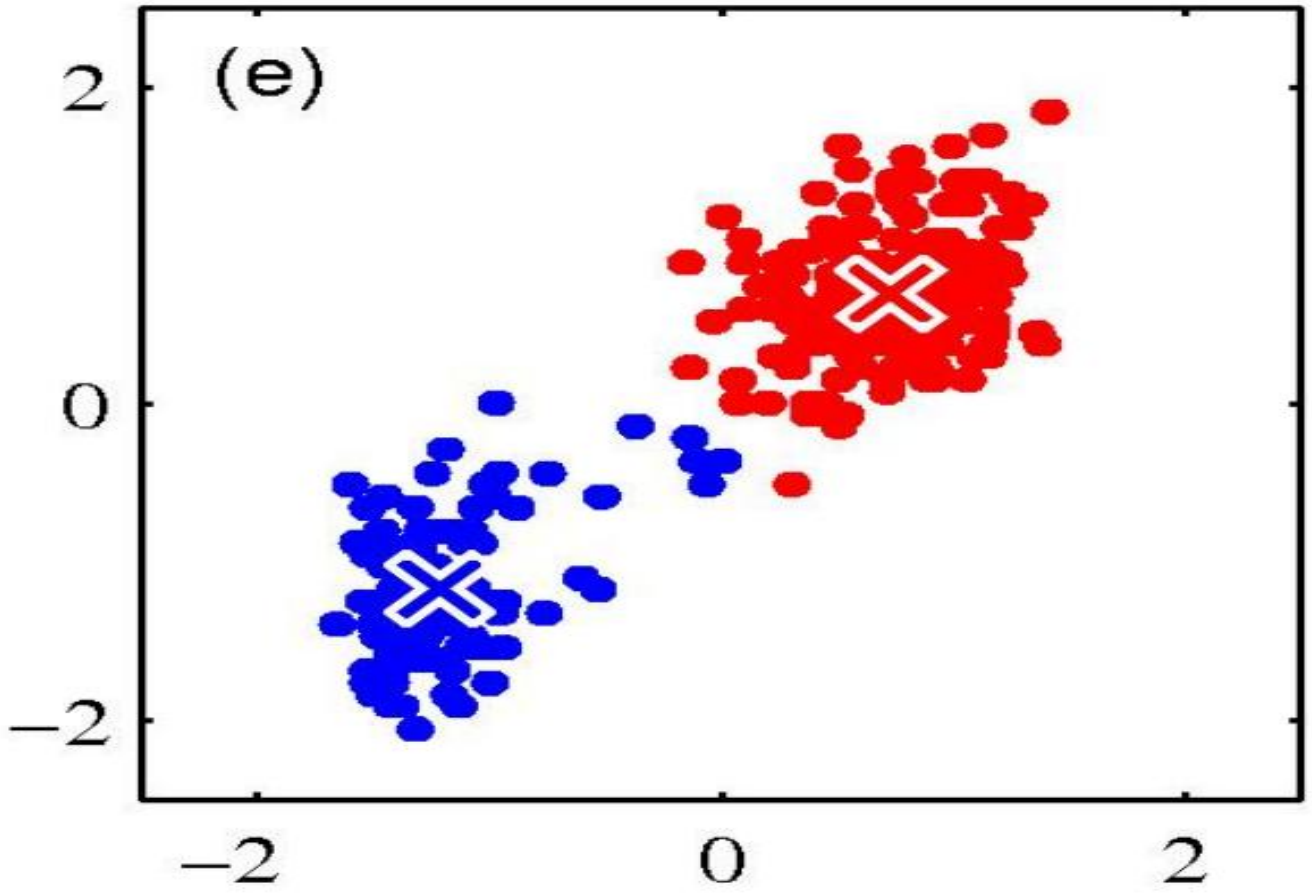
- Change the cluster center to the average of the assigned points

K-means clustering: Example

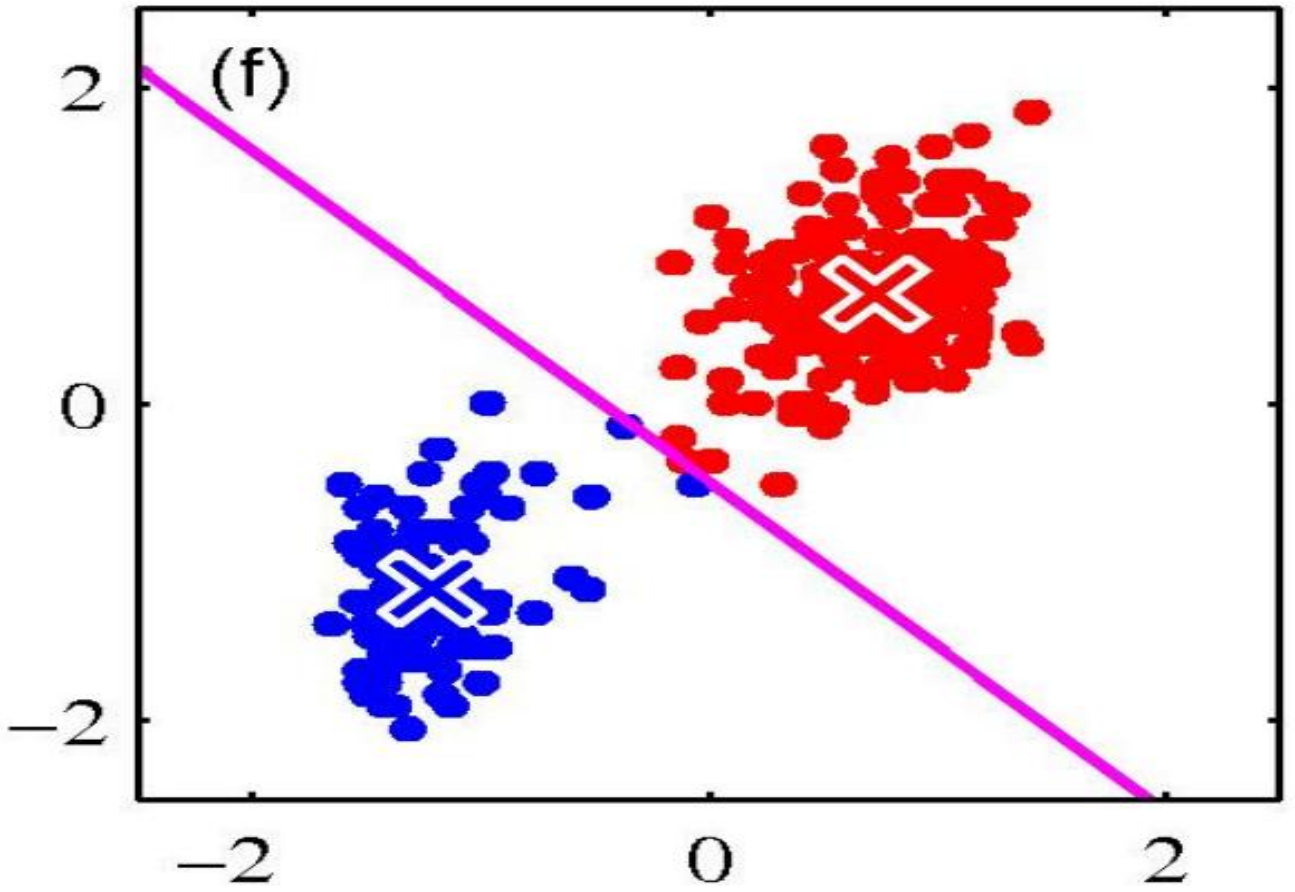


- Repeat until convergence

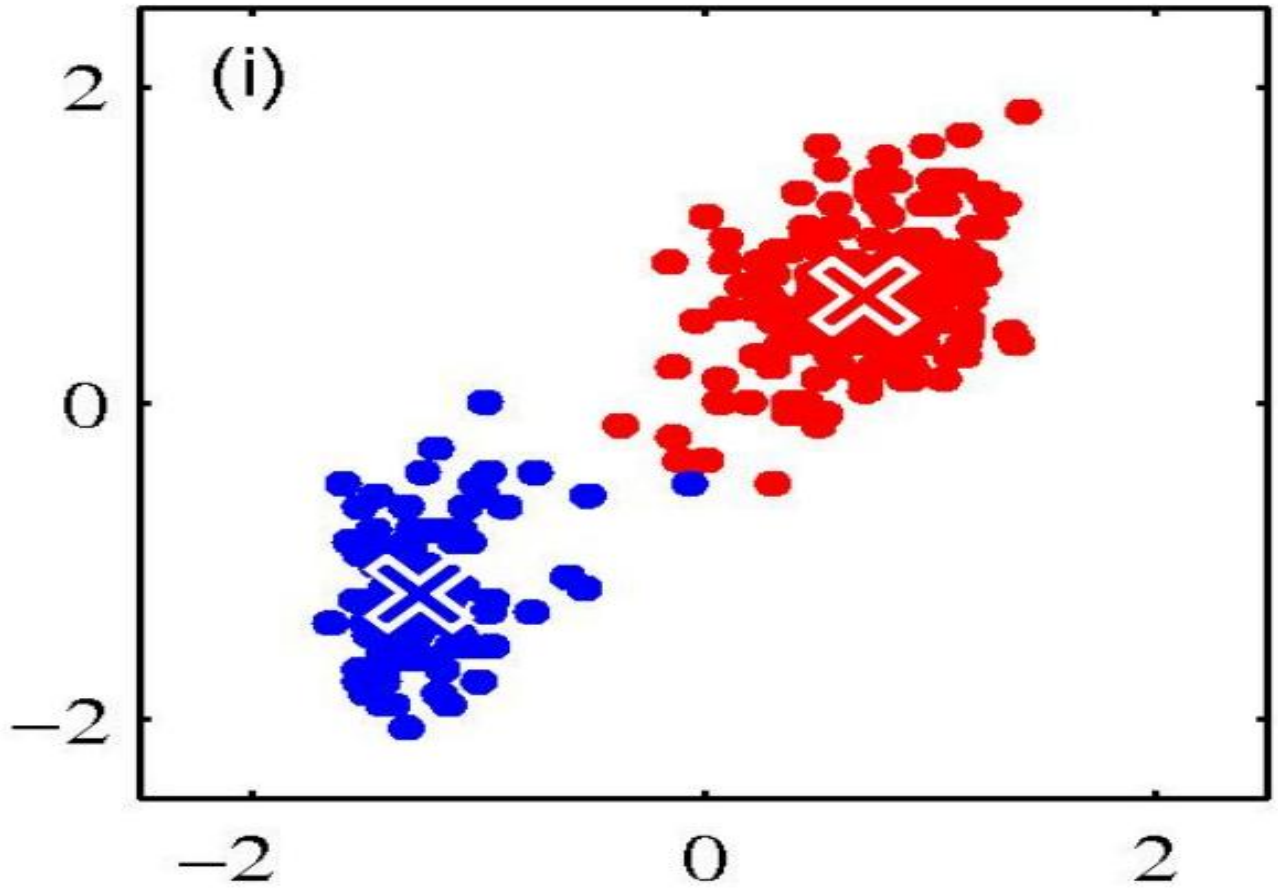
K-means clustering: Example



K-means clustering: Example



K-means clustering: Example



Comments on the *K-Means* Method

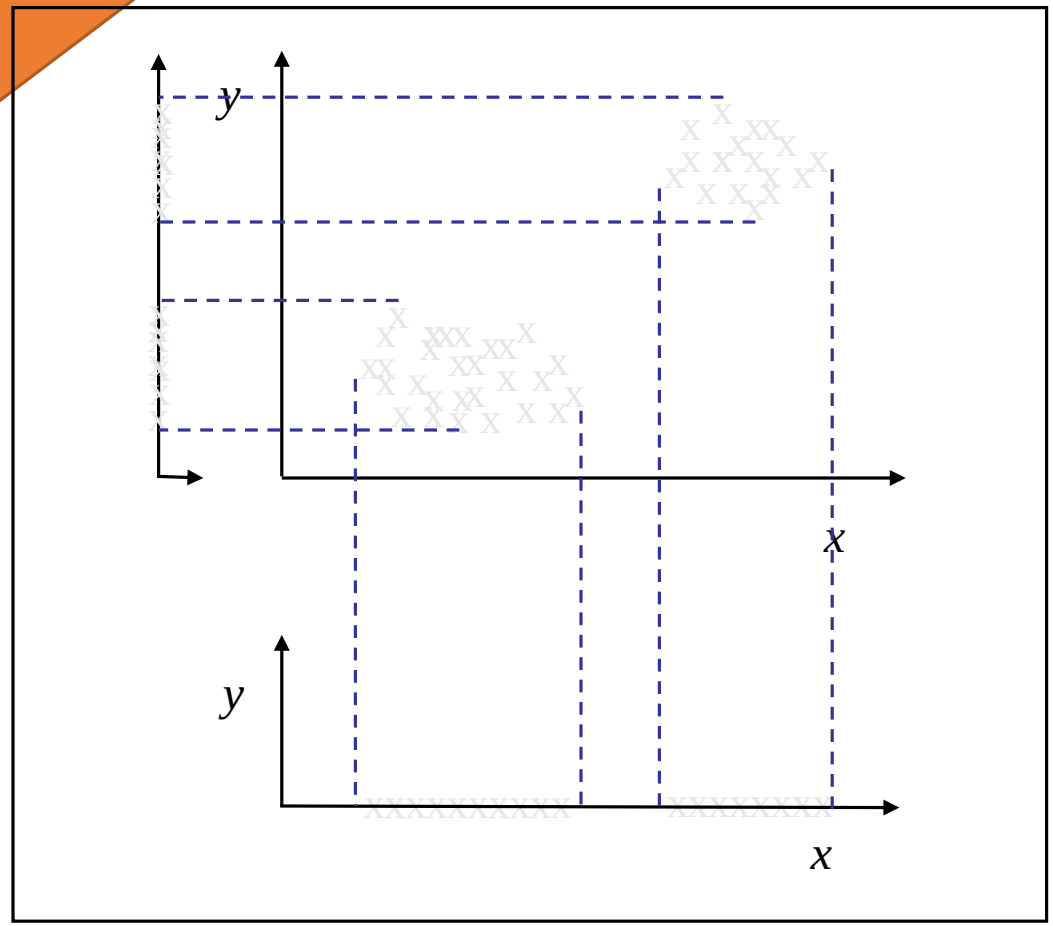
- Strength: *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k*, the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

High-Dimensional Data poses Problems for Clustering

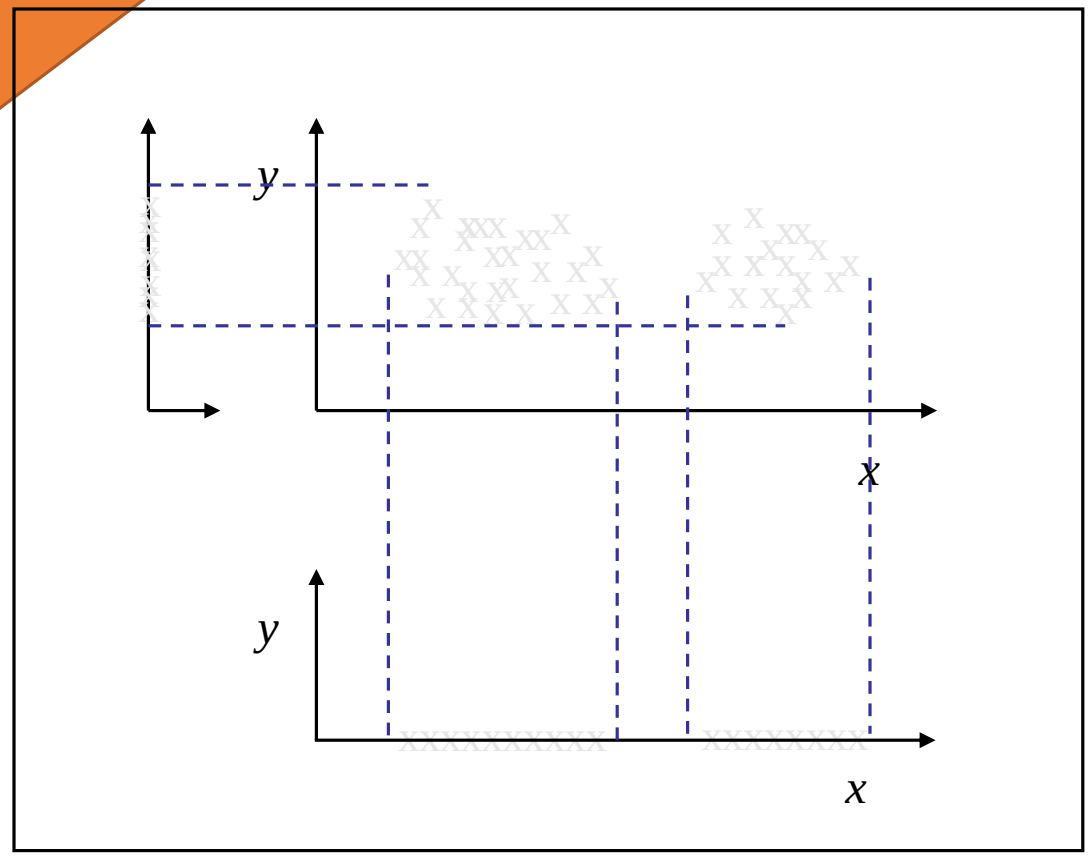
- **Difficult to find true clusters**
  - Irrelevant and redundant features
  - All points are equally close

- **Solutions: Dimension Reduction**
  - Feature subset selection
  - Cluster ensembles using random projection (in a later lecture….)

Redundant



Irrelevant



Curse of Dimensionality

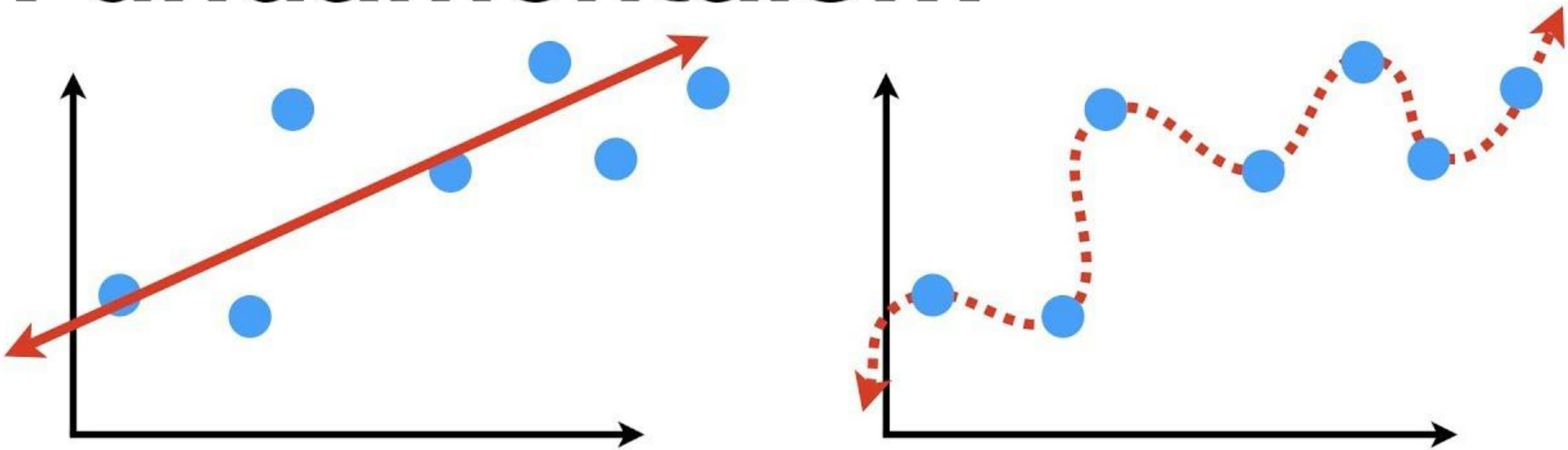
100 observations cover the 1-D unit interval [0,1] well

Consider the 10-D unit hypersquare, 100 observations are now isolated points in a vast empty space.

Consequence of the Curse

- **Suppose the number of samples given to us in the total sample space is fixed**
- **Let the dimension increase**
- **Then the distance of the k nearest neighbors of any point increases**

Machine Learning Fundamentals...

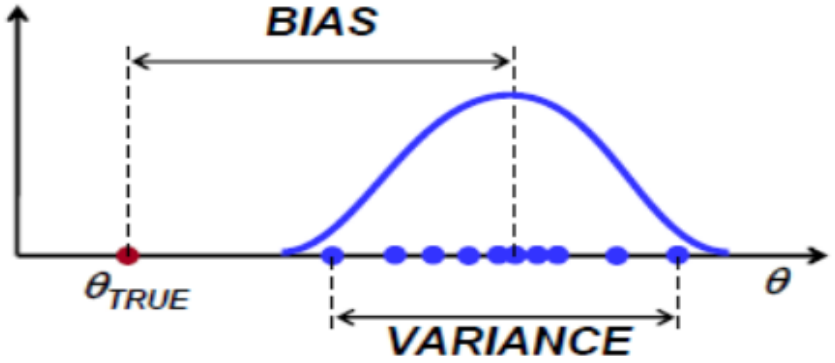


...Bias and Variance!!!

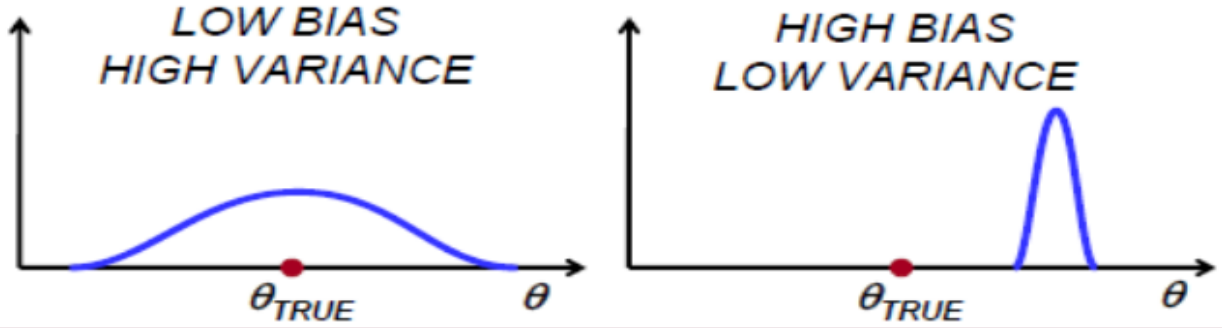
- In previous lectures we showed how to build classifiers when the underlying densities are known
  - Bayesian Decision Theory introduced the general formulation
- In most situations, however, the true distributions are unknown and must be estimated from data.
  - **Parameter Estimation** (we saw the Maximum Likelihood Method)
    - Assume a particular form for the density (e.g. Gaussian), so only the parameters (e.g., mean and variance) need to be estimated
    - Maximum Likelihood
    - Bayesian Estimation
  - **Non-parametric Density Estimation** (not covered)
    - Assume NO knowledge about the density
    - Kernel Density Estimation
    - Nearest Neighbor Rule

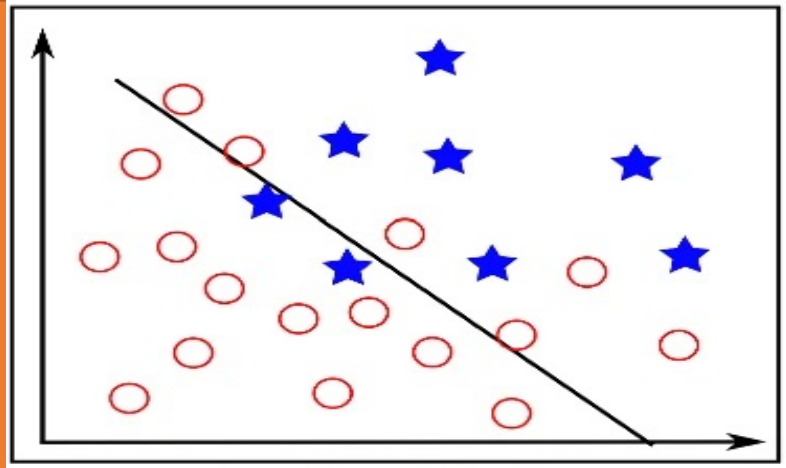
Bias and variance (1)

- **How good are these estimates? Two measures of "goodness" are used for statistical estimates**
  - **BIAS**: how close is the estimate to the true value?
  - **VARIANCE**: how much does the estimate change for different runs (e.g. different datasets)?

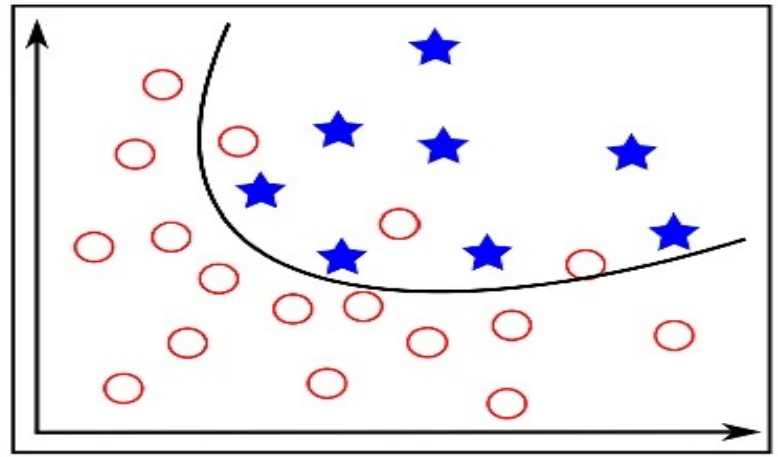


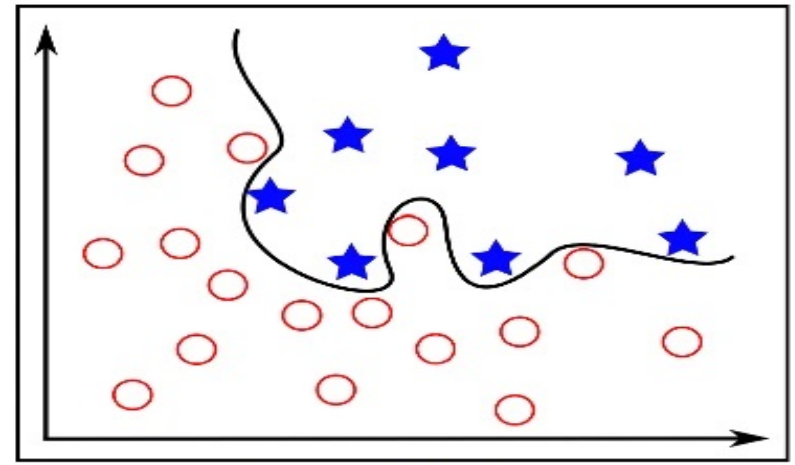
- **The bias-variance tradeoff**
  - In most cases, you can only decrease one of them at the expense of the other



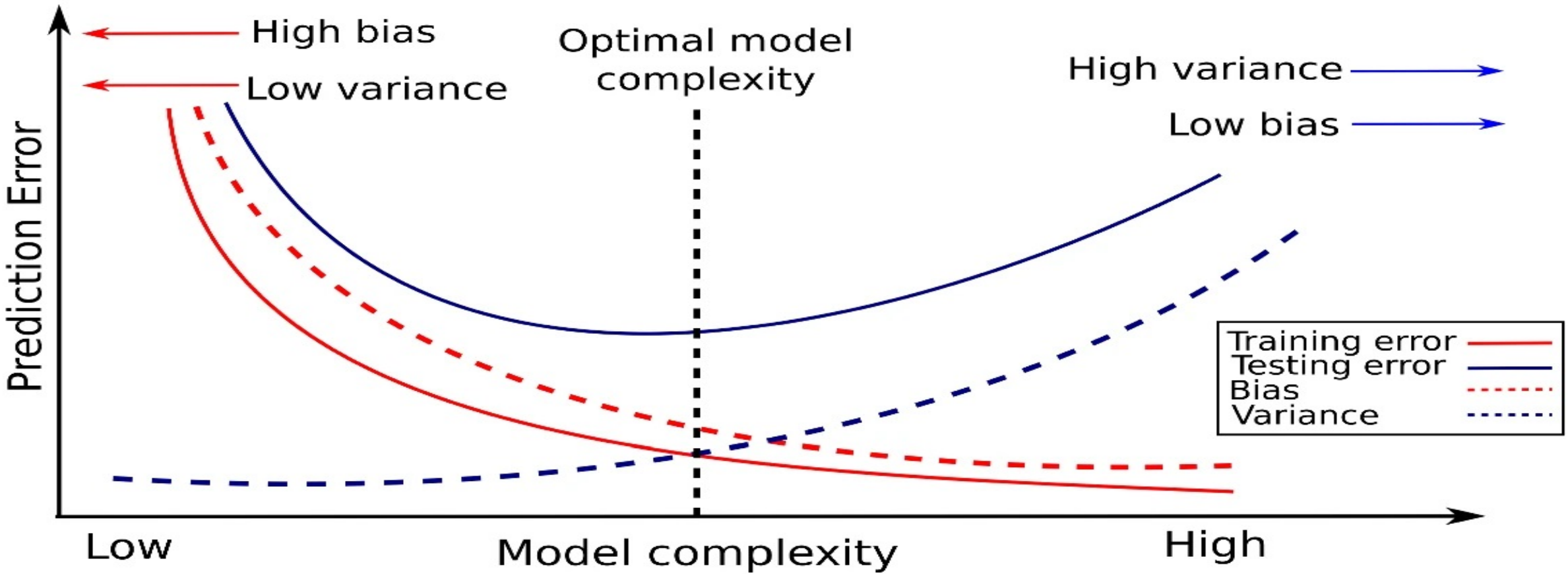


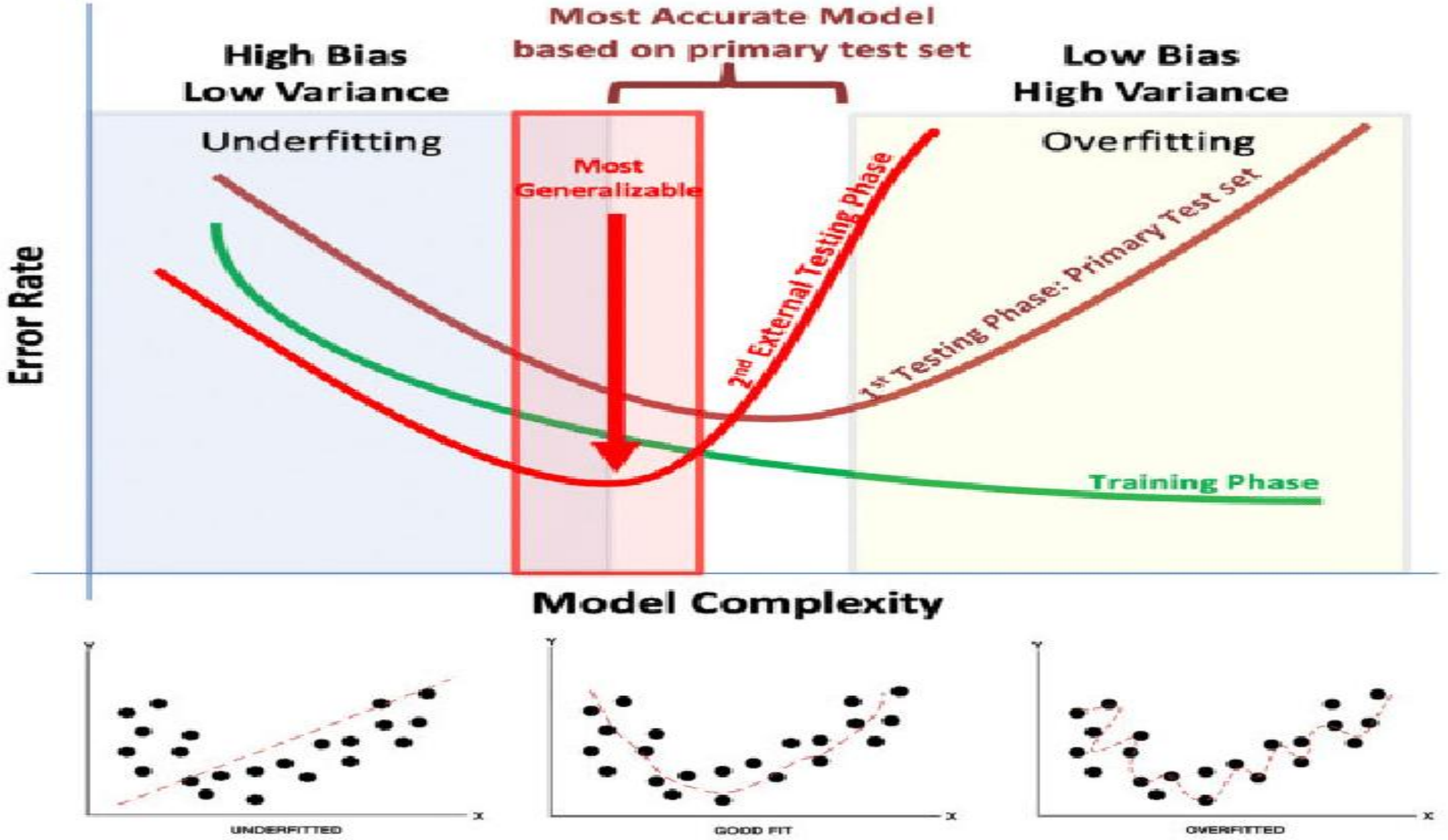
Underfitting





Overfitting



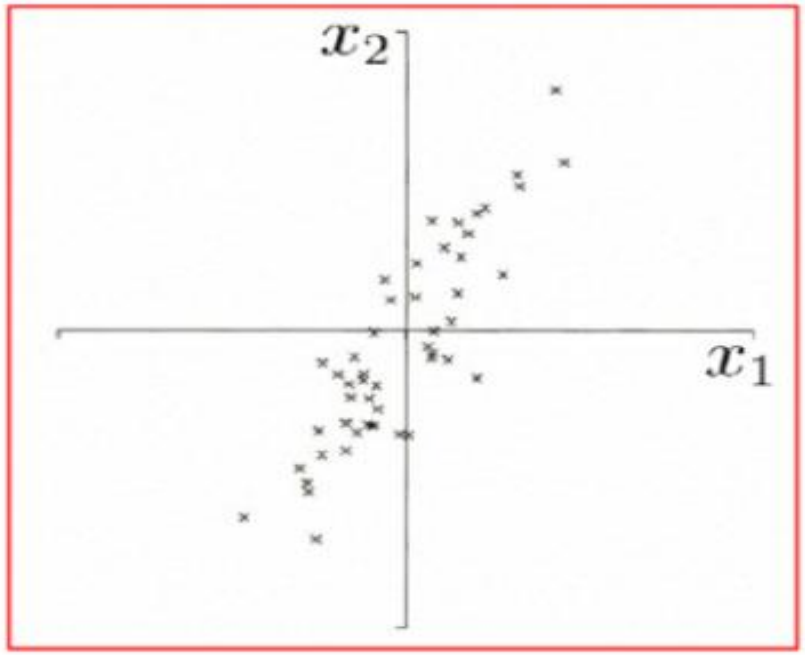

Most Accurate Model
based on primary test set
High Bias
Low Variance
Underfitting
Most
Generalizable
Low Bias
High Variance
Overfitting
2nd External Testing Phase
1st Testing Phase: Primary Test set
Training Phase
Error Rate
Model Complexity
UNDERFITTED
GOOD FIT
OVERFITTED

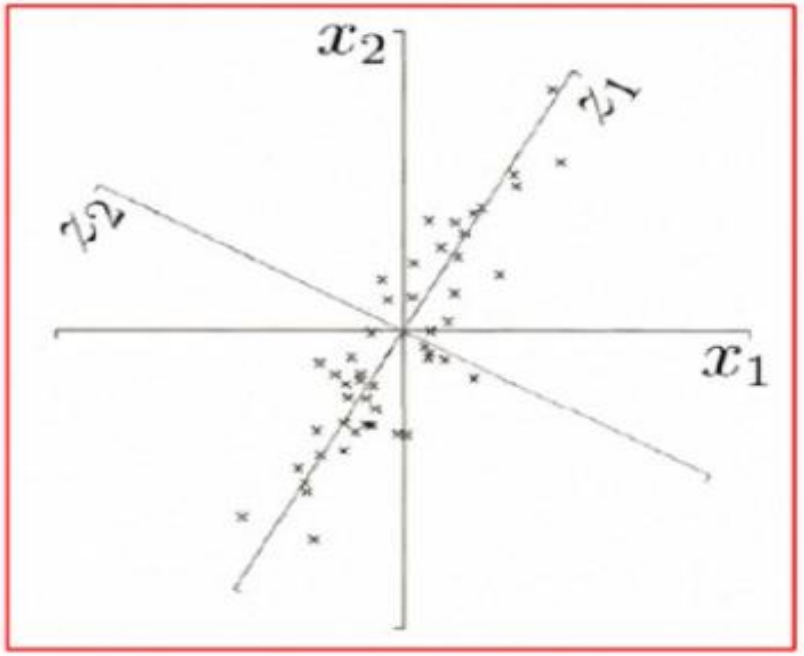
Principal component analysis (PCA)

- Widely used method for unsupervised, linear dimensionality reduction
- GOAL: account for variance of data in as few dimensions as possible (using linear projection)

Geometric picture of principal components (PCs)

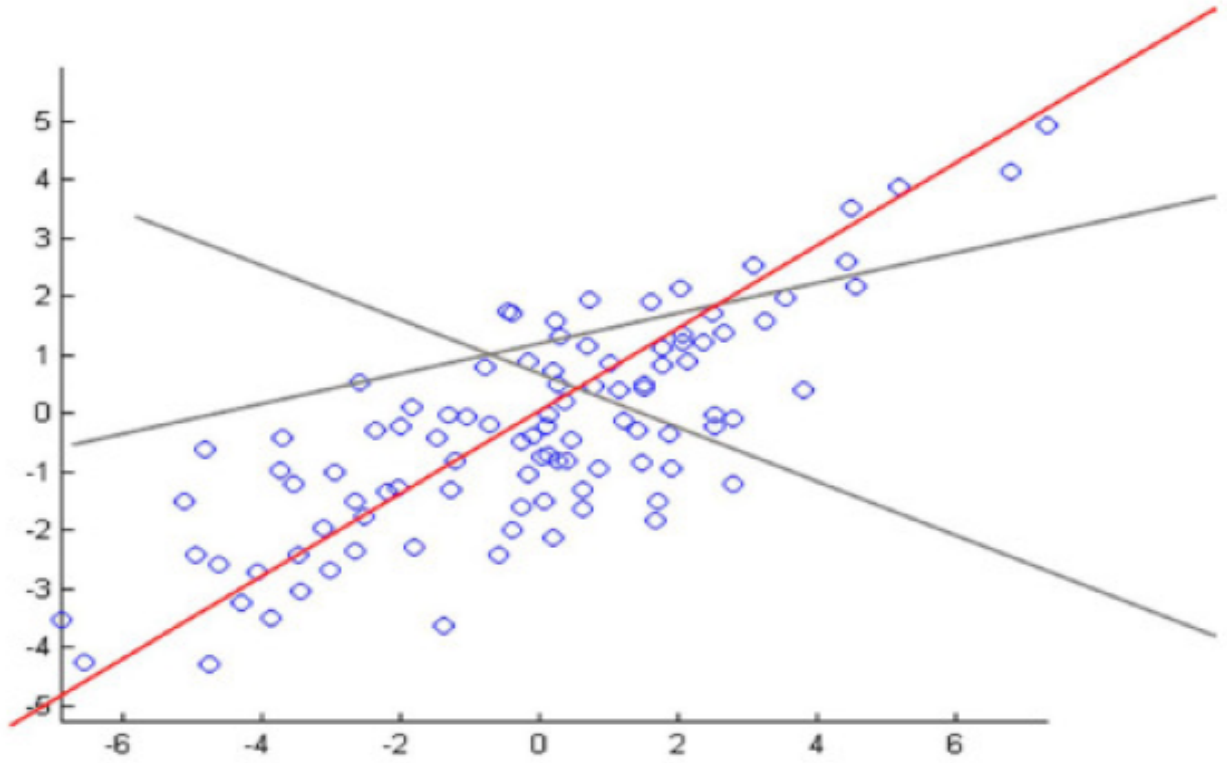
- First PC is the projection direction that maximizes the variance of the projected data
- Second PC is the projection direction that is orthogonal to the first PC and maximizes variance of the projected data





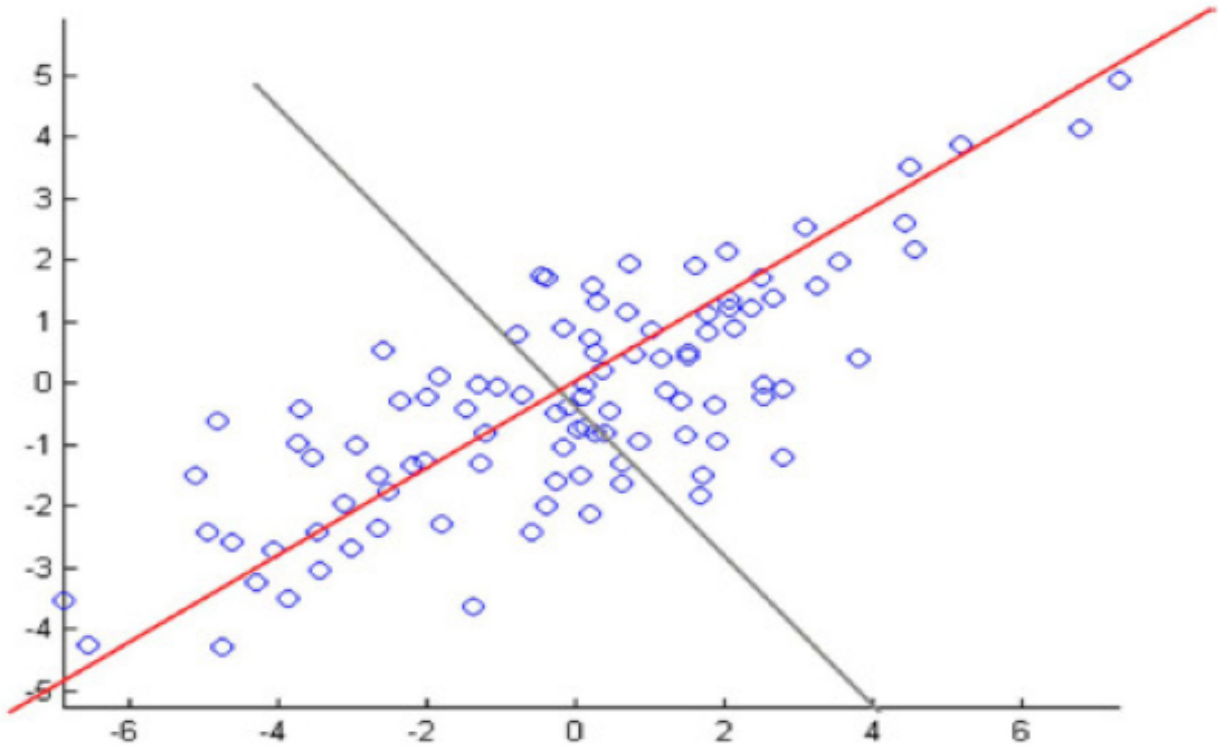
PCA: conceptual algorithm

- Find a line, such that when the data is projected onto that line, it has the maximum variance.



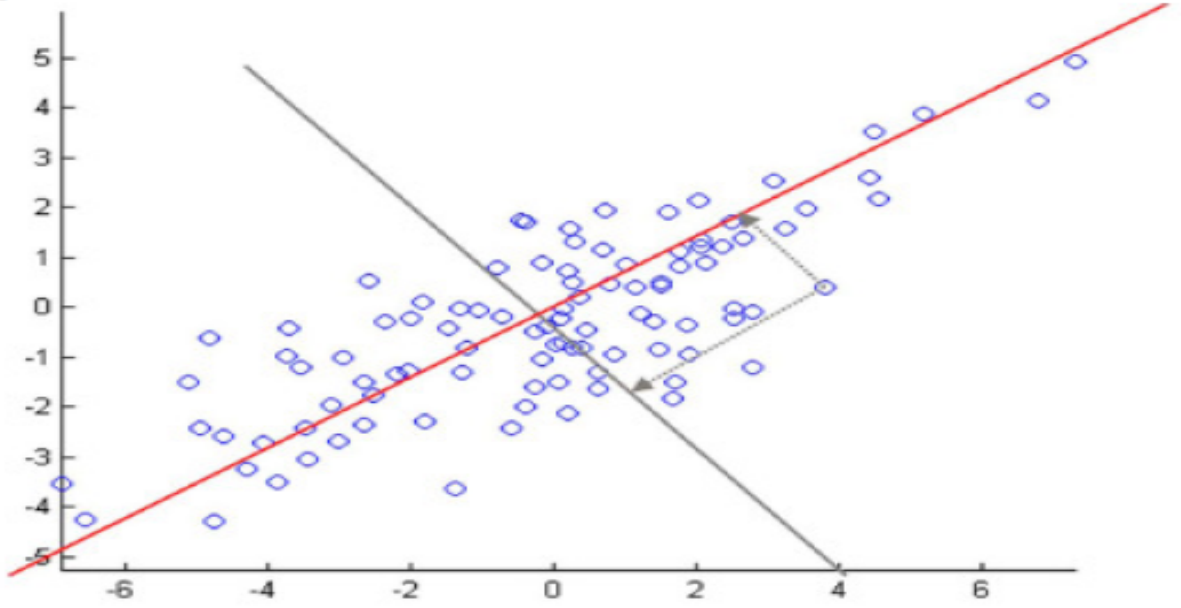
PCA: conceptual algorithm

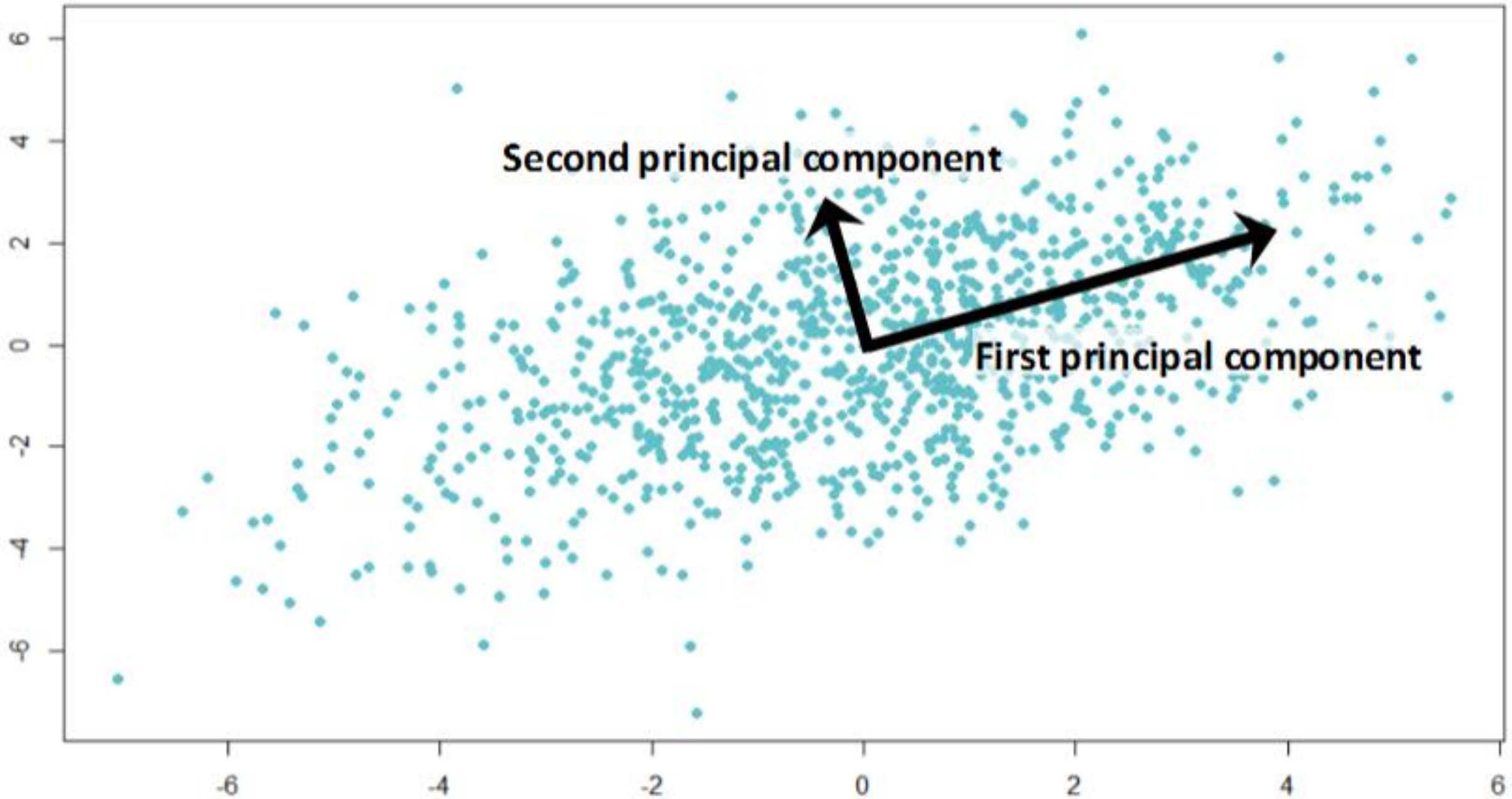
- Find a second line, orthogonal to the first, that has maximum projected variance.



PCA: conceptual algorithm

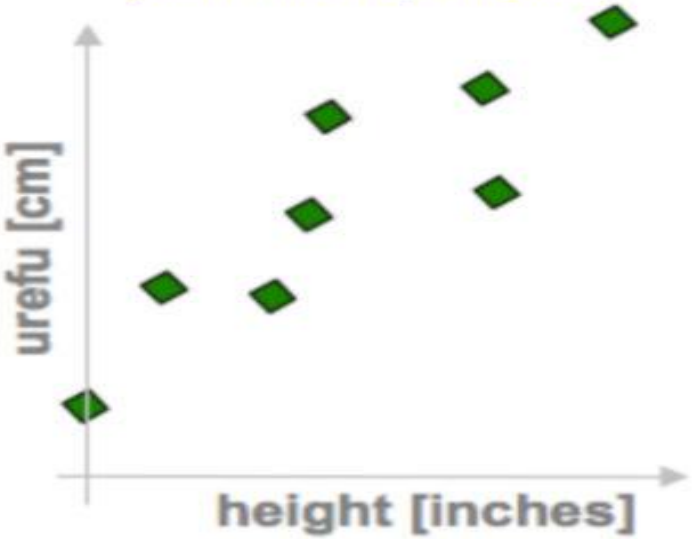
- Repeat until have *k* orthogonal lines
- The projected position of a point on these lines gives the coordinates in the *k*-dimensional reduced space.



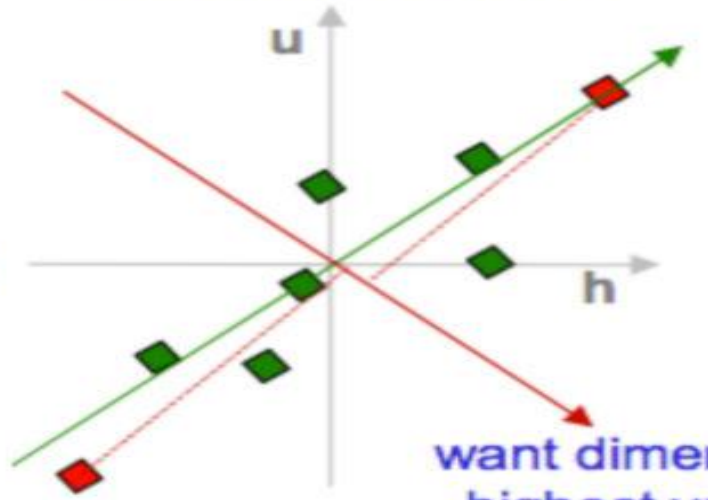

Second principal component
First principal component
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6

PCA in a nutshell

1. correlated hi-d data
("urefu" means "height" in Swahili)



2. center the points



want dimension of highest variance

3. compute covariance matrix

	h	u
h	2.0	0.8
u	0.8	0.6

$$\text{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

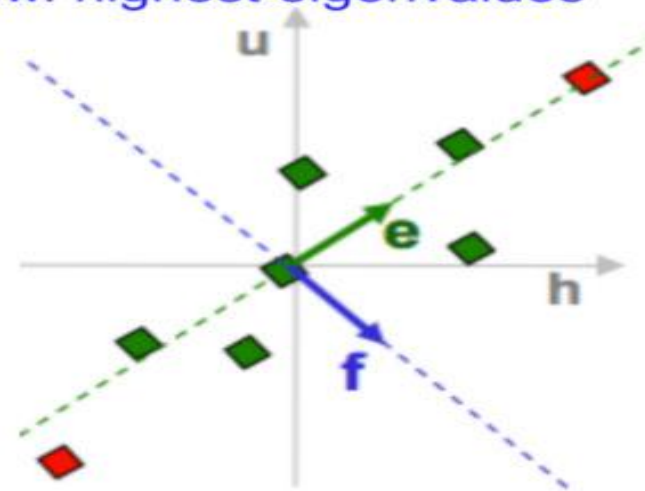
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

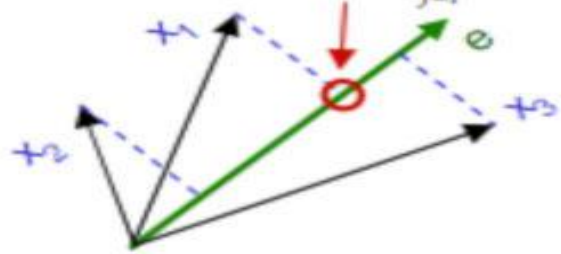
```
eig(cov(data))
```

5. pick m<d eigenvectors
w. highest eigenvalues

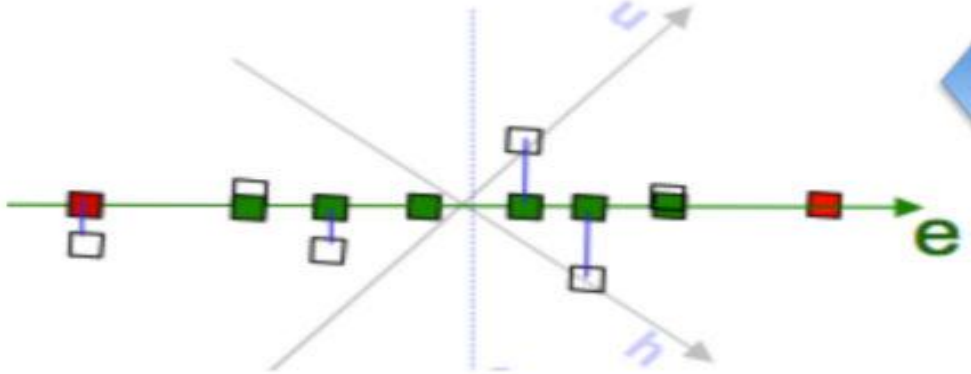


6. project data points to those eigenvectors

$$x_e' = x^T e = \sum_{j=1}^{a} x_{ij} e_j$$



7. uncorrelated low-d data



Steps in principal component analysis

- Mean center the data
- Compute covariance matrix Σ
- Calculate eigenvalues and eigenvectors of Σ
  - 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

Applying a principal component analysis

- Full set of PCs comprise a new orthogonal basis for feature space, whose axes are aligned with the maximum variances of original data.
- Projection of original data onto first *k* PCs gives a reduced dimensionality representation of the data.
- Transforming reduced dimensionality projection back into original space gives a reduced dimensionality *reconstruction* of the original data.
- Reconstruction will have some error, but it can be small and often is acceptable given the other benefits of dimensionality reduction.