Lecture 10 – Unsupervised Learning

Cluster Analysis, The Curse of Dimensionality

Information Systems for Sustainable Society (is3) | WiSo Faculty | Univ.-Prof. Dr. Wolfgang Ketter | 12/13/22

Agenda


Clustering in research context
Soft Clustering
Dimensionality reduction using PCA

Research Context – Clustering Bidders in the Dutch Flower Auction

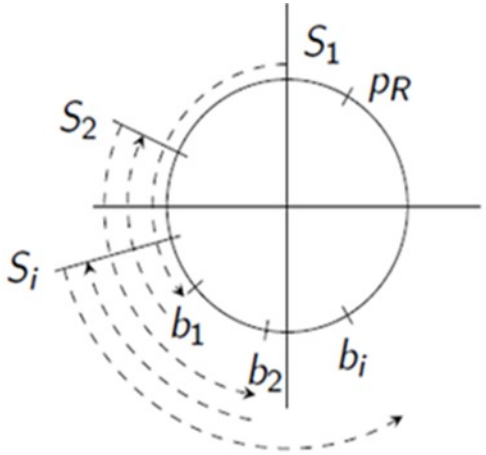
The Dutch Flower Auctions

- 60% of the global flower trades.
- Over €4 billion annual revenue.
- 9000 global suppliers, 6000 global buyers
- 6 auction sites, 40 auction clocks
- 125,000 daily transactions

The Auction Clock

The Mechanism

- Multi-unit Sequential Dutch Auction



Starting price, S_i	100	86	70	65	66
Minimum units	1	2	2	3	3

- Auctioning an 18-unit lot

Round	Transaction Time	Starting Price (cent)	Minimum Purchase Units	Available Units	Purchase Units	Price (cent)	Bidder ID	…
1	08:30:45	100	1	18	1	74	439	…
2	08:30:47	86	2	17	3	58	395	…
3	08:30:50	70	2	14	3	53	600	…
4	08:30:51	65	3	11	4	54	563	…

Clustering of buyer types

- **Input:** transaction data incl. buyer IDs
- **Question:** Which buyers are similar?

Research Design

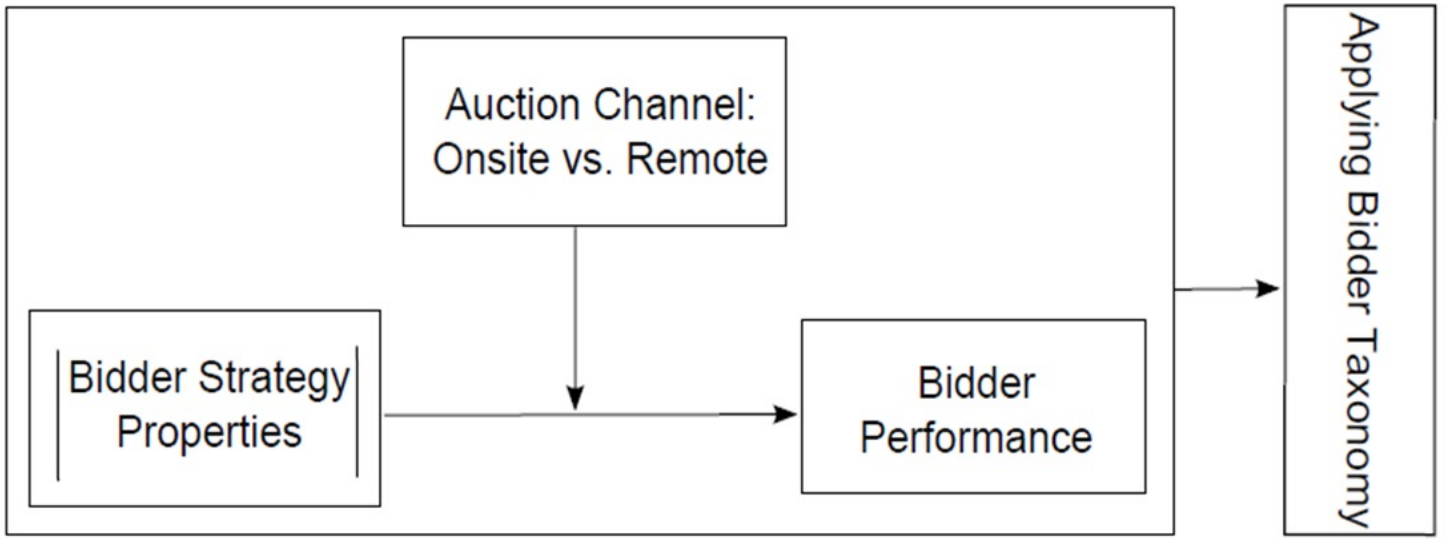


Figure: Research Model

- Classification Variables
  - **Auction-level:** Time of Entry (TOE), Frequency of Bid (FOB)
  - **Day-level:** Active Time per Day (ATD), Bidding Frequency per Day (BFD)

Data

- The dataset contains transactions of roses from June to September in 2010 at a major auction site.
  - 2010: there are 280945 transactions from 38848 lots
  - 593 bidders (305 bid onsite, 288 bid remotely)

Transaction Index	Seller ID	Flower ID	Stem Length	Stems per Units	Available Units	Minimum Purchase Units	Starting Price (cent)	Bidder ID	Purchase Units	Price (cent)
171	5644	103668	70	50	18	2	100	439	2	22
172	5644	103668	70	50	16	3	41	395	5	20
173	5644	103668	70	50	11	4	39	439	7	21
174	5644	103668	70	50	4	4	40	563	4	20

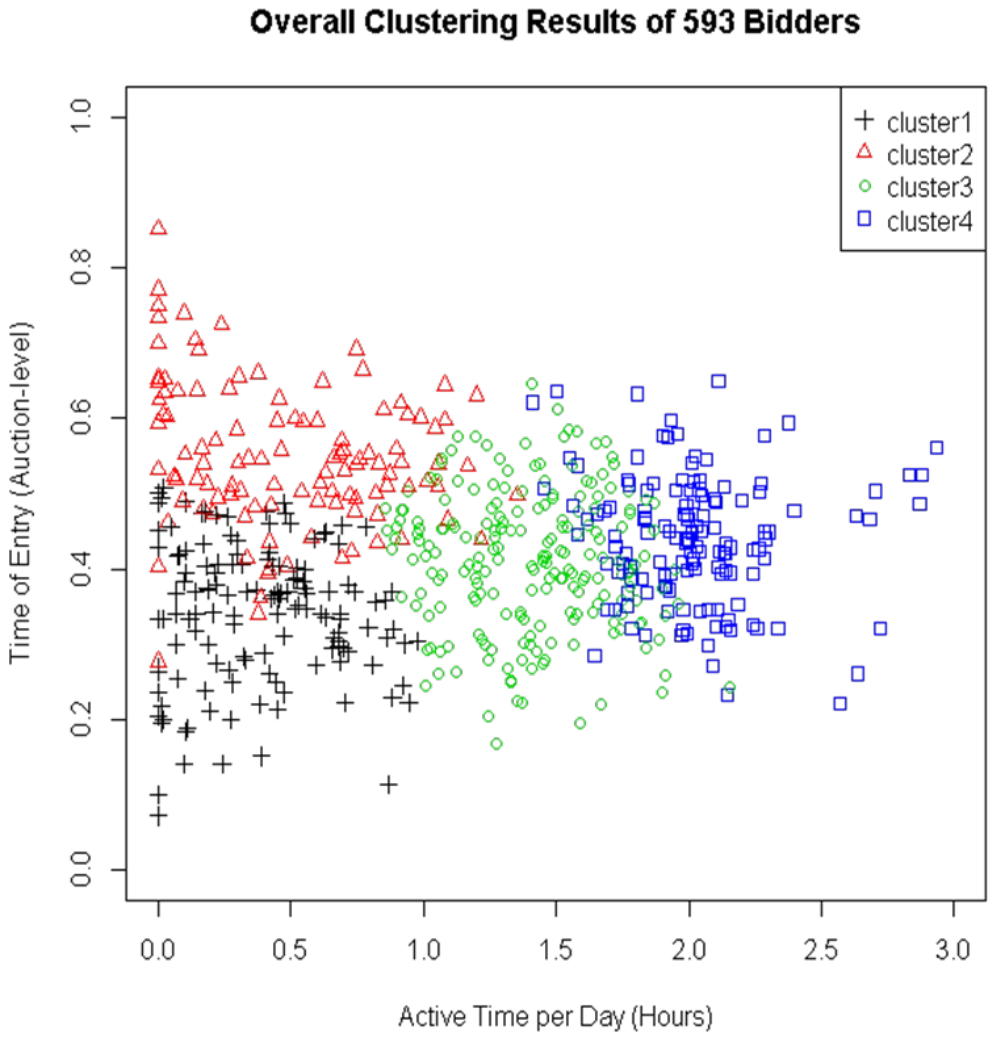
Methodology

- External Criterion (Interpretability):

	Variable	Mean Squares Cluster	Mean Squares Error	F	Significance
Auction-level	Time of Entry	0.622	0.020	31.399	0.000
	Frequency of Bid	0.000	0.001	0.107	0.744
Day-level	Active Time per Day	318.440	0.110	2830.400	0.000
	Bidding Frequency per Day	30948.500	46.300	667.830	0.000

- K-means Clustering: Automatically partition a set of observations into K disjoint subsets.
- How to determine K ?
  - Internal Criterion:
  - Repeat K-means clustering with a range of K (K_min = 2, K_max= 10)
  - According to the **Calinski-Harabasz** criterion, the optimal number of clusters is four.

Results of K-means Clustering



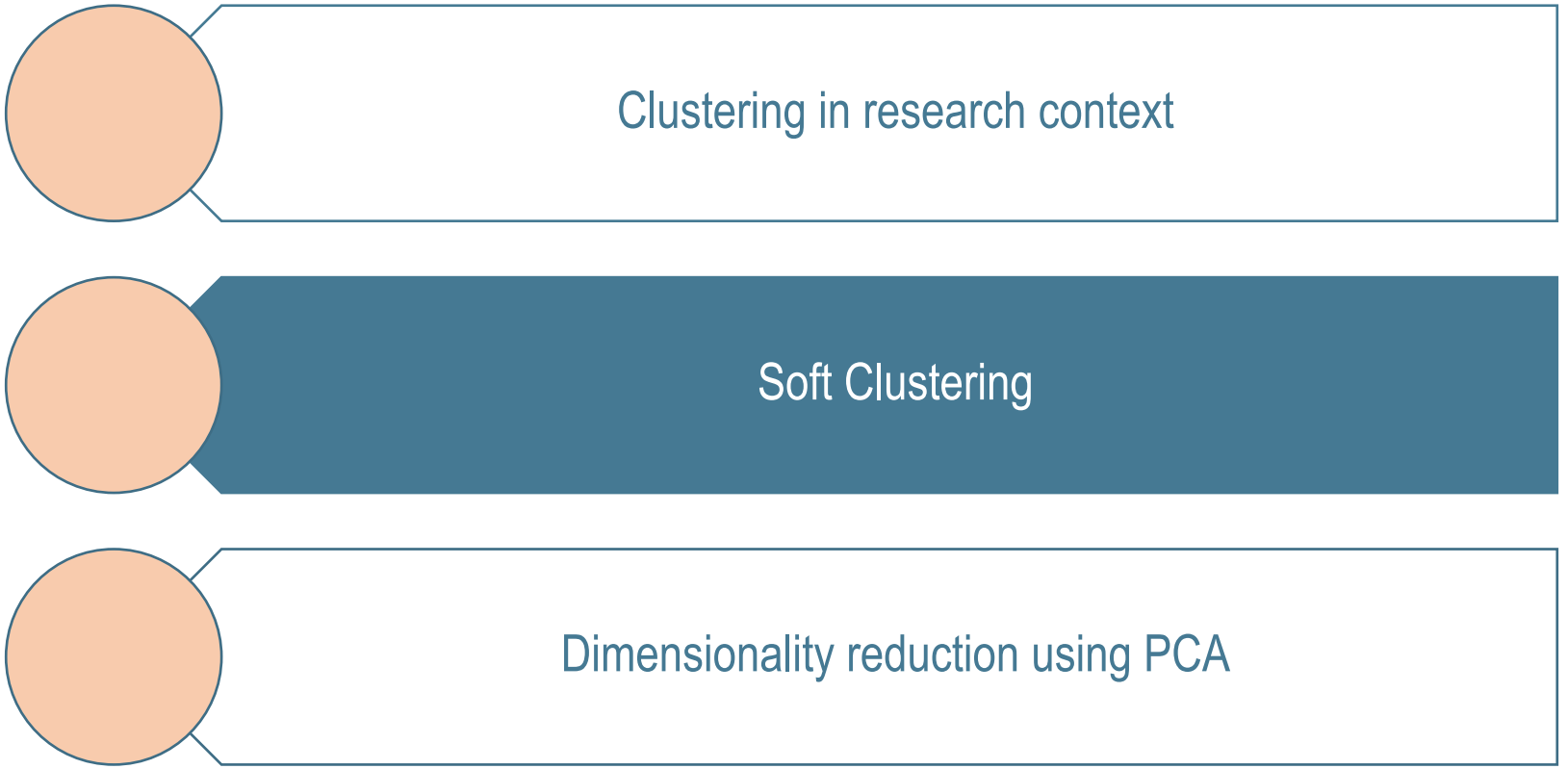
How would you interpret the four flower buyer clusters derived via k-means clustering from a domain perspective?

Bidder Types

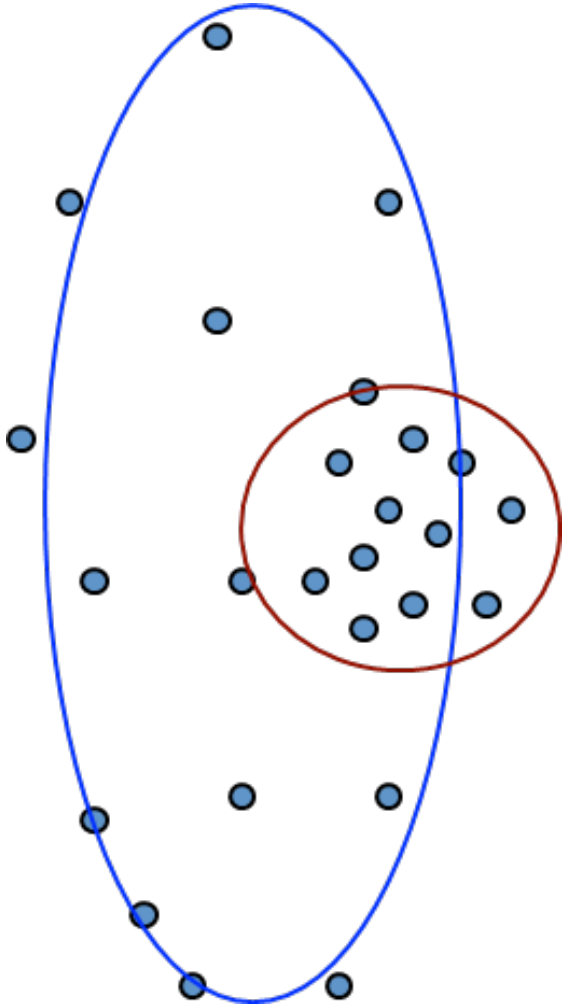
Bidder Types	Description
Early Bidders (Cluster 1)	Submit bids early; Low involvement;
Opportunists (Cluster 2)	Submit bids late; Low involvement;
Participants (Cluster 3)	Submit bids neither early nor late; Medium involvement;
Analytical Bidders (Cluster 4)	Submit bids neither early nor late; High involvement;

Read the full paper here.

Agenda

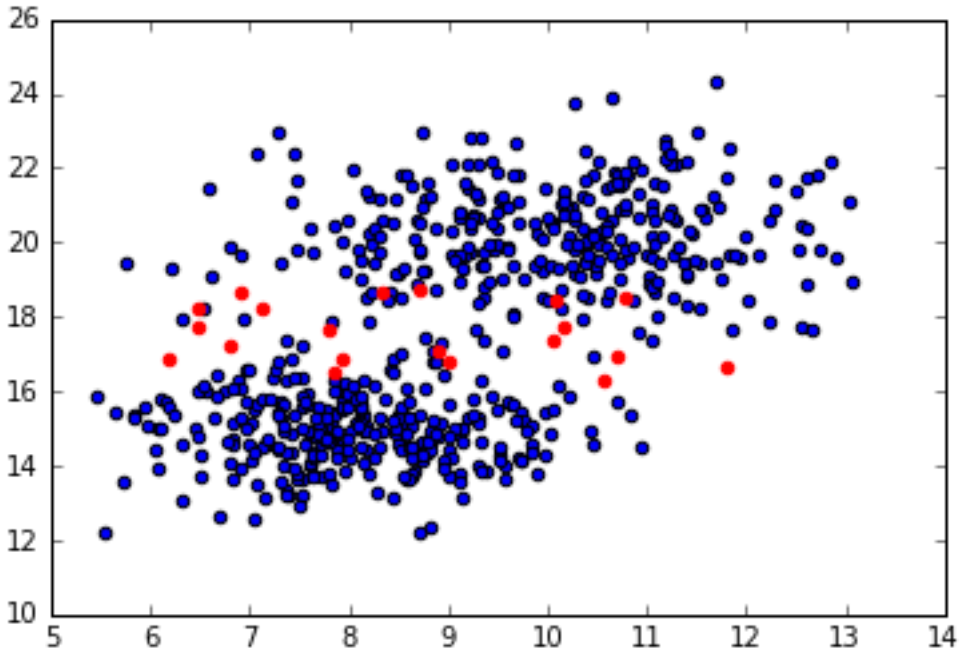

Clustering in research context
Soft Clustering
Dimensionality reduction using PCA

The Evils of “Hard Assignments”?



- Sometimes when we're performing clustering on a dataset, there exist points which don't belong strongly to any given cluster. If we were to use something like k-means clustering, we're forced to decide as to which group an observation belongs to
- Some clusters may be “wider” than others
- Distances can be deceiving!

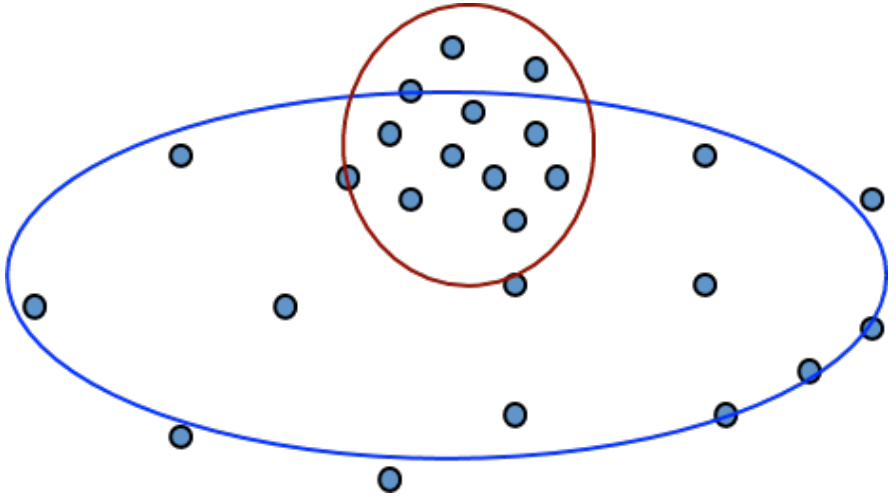
The Evils of “Hard Assignments”?



- For example, examine the scatterplot showing two clusters (in blue) and some fringe observations (in red) that may belong to either of the two clusters.
- Ideally, we'd like to be as true to the data as possible when assigning observations to clusters; allowing partial assignment to multiple clusters allows us to more accurately describe the data.

Probabilistic Clustering

- **Try a probabilistic model!**
  - allows overlaps, clusters of different size, etc.
- Can tell a **generative story** for data
  - P(X|Y) P(Y)
- **Challenge:** we need to estimate model parameters without labeled Ys



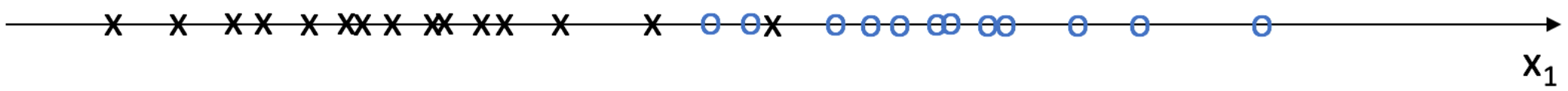
Y	X_1	X_2
??	0.1	2.1
??	0.5	--1.1
??	0.0	3.0
??	--0.1	--2.0
??	0.2	1.5
…	…	…

Soft Clustering

- Formally, soft clustering (also known as fuzzy clustering) is a form of clustering where observations may belong to multiple clusters.
- A common soft clustering technique is known as **expectation maximization** (EM) of a **Gaussian mixture model**.
- Essentially, the process goes as follows:
  - Identify the number of clusters you'd like to split the dataset into.
  - Define each cluster by generating a Gaussian model.
  - For every observation, calculate the probability that it belongs to each cluster (ex. Observation 23 has a 21% chance that it belongs to Cluster A, a 0.1% chance that it belongs to Cluster B, a 48% chance of Cluster C, ... and so forth).
  - Using the above probabilities, recalculate the Gaussian models.
  - Repeat until observations more or less "converge" on their assignments.

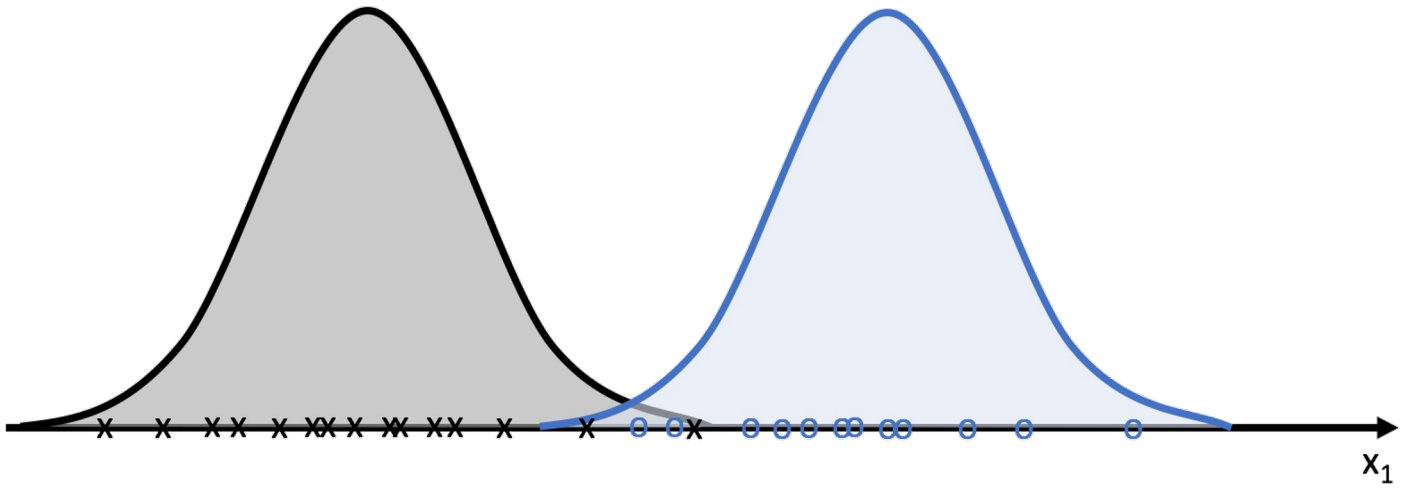
(Simple) Example

- Suppose I have data for a set of observations with one feature, x_1, that come from two distinct classes.



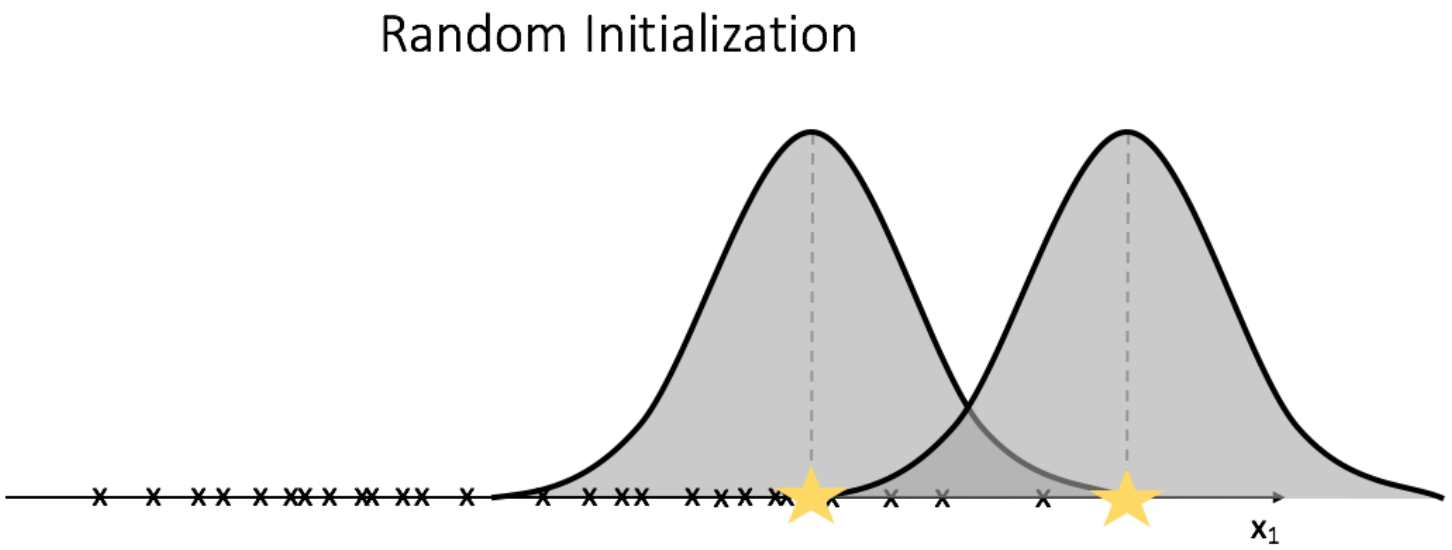
- We can use this data to build a Gaussian model for each class which would allow us to calculate the probability of a new observation belonging to each class. The class denoted by black x's would be used to build one Gaussian model and the class denoted by blue o's would be used to build a separate Gaussian model.

Gaussian Mixture Models (GMM)



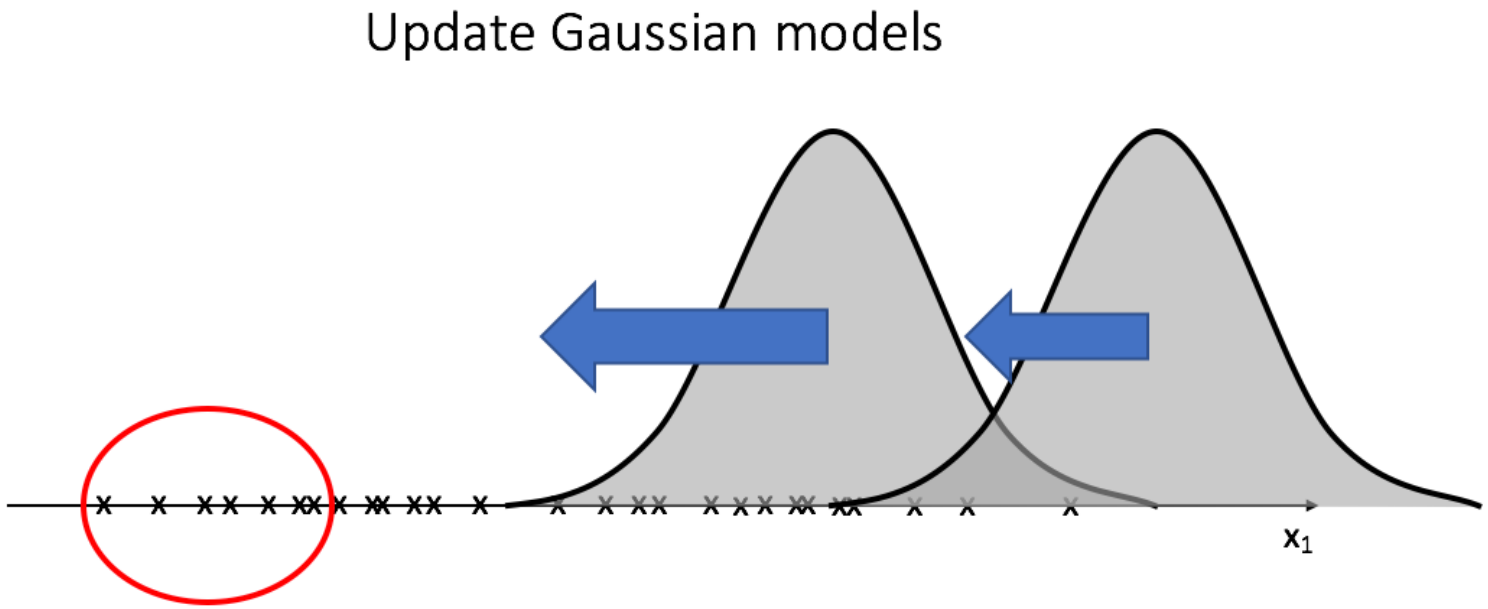
- Unfortunately, we usually perform clustering because we don't have a labeled dataset and thus don't know which class any of the observations belong to- that's what we're hoping to learn!
- Since we don't know which class each observation belongs to, we don't have an easy way to build multiple Gaussian models to partition the data. We can no longer simply calculate the mean and variance of observations belonging to each class.

Gaussian Mixture Models (GMM)



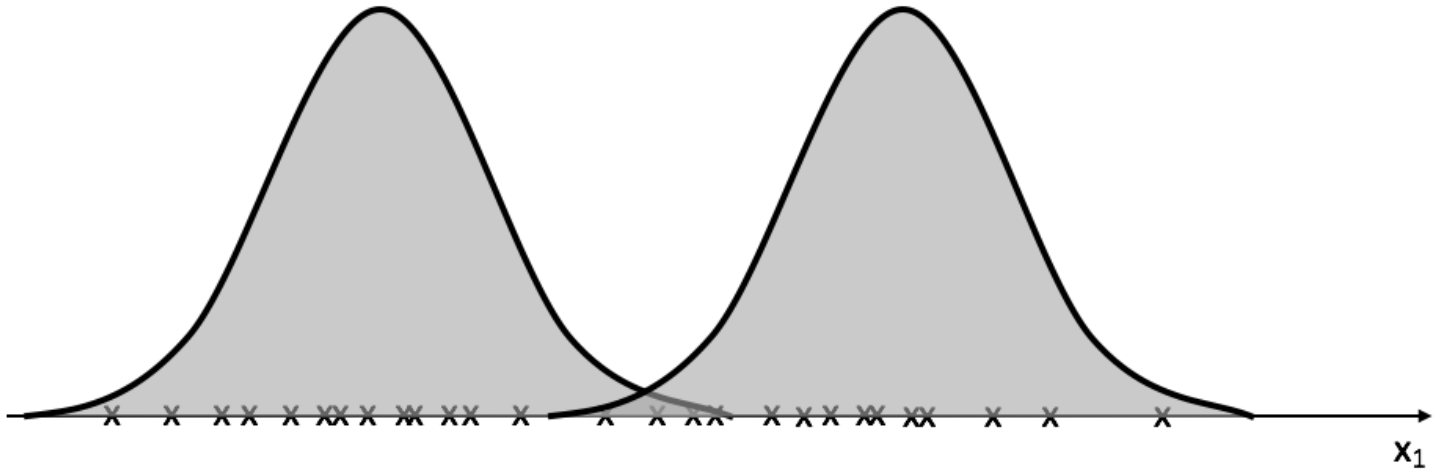
- Let's start with a random guess of our Gaussian models and then iteratively optimize the attributes, in a similar fashion as we did for k-mean clustering, to find the optimal Gaussian model to express the data.

Gaussian Mixture Models (GMM)



- After initializing two random Gaussians, we'll compute the likelihood of each observation being expressed in both Gaussian models.
- We'll then recalculate the Gaussian models; we'll use all observations in calculating the mean and variance for each Gaussian, but the observations will be weighted by the likelihood of existing in the given model.

Gaussian Mixture Models (GMM)



- We'll continue this cycle of recalculating probabilities and then using them to update the Gaussians until we "converge" on optimal clusters.

- This example visualizes a univariate Gaussian example, we can extend this logic, however, to accommodate multivariate datasets.

Steps of the algorithm: Probabilistic assignment to clusters (expectation)

- After initializing k random Gaussian models, we can calculate our expectation of z_i, a vector of probabilities that x_i belongs to the jth cluster for j=1 to j=k. As mentioned earlier, we don't know the true probabilistic cluster assignments for z, so we'll start with a guess and iteratively refine it.

$$z_i = \begin{bmatrix} P(x_i \; belongs \; to \; Cluster \; 1) \\ \vdots \\ P(x_i \; belongs \; to \; Cluster \; k) \end{bmatrix}$$

$$E[z_{i,j}] = \frac{p(x = x_i | \mu = \mu_j)}{\sum_{m=1}^{k} p(x = x_i | \mu = \mu_m)}$$

Steps of the algorithm: Probabilistic assignment to clusters (expectation)

We can calculate the probability of x_i belonging to cluster j using the probability distribution function of a Gaussian distribution.

$$p(x = x_i | \mu = \mu_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-(\frac{x_i - \mu_j}{2\sigma_j^2})}$$

Reformulating the Gaussian models (maximization)

- We'll then recalculate our Gaussian models leveraging the weights we found in the expectation step. The expectation, $E[z_{i,j}]$, represents the likelihood that the *i*th observation belongs to cluster j.

$$\mu_j = \frac{\sum_i E[z_{i,j}] x_i}{\sum_i E[z_{i,j}]}$$

$$\sigma_j = \frac{\sum_i E[z_{i,j}] (x_i - \mu_j)(x_i - \mu_k)}{\sum_i E[z_{i,j}]}$$

Defining a stopping criterion

- With k-means clustering, we iteratively recalculated means and reassigned observations until convergence and observations stopped moving between clusters.
- However, because we're now dealing with continuous probabilities, and because we never give a hard cluster assignment, we can't rely on this convergence.
- Instead, we'll set a stopping criterion to end the iterative cycle once the observation probabilities stop changing by above some threshold.

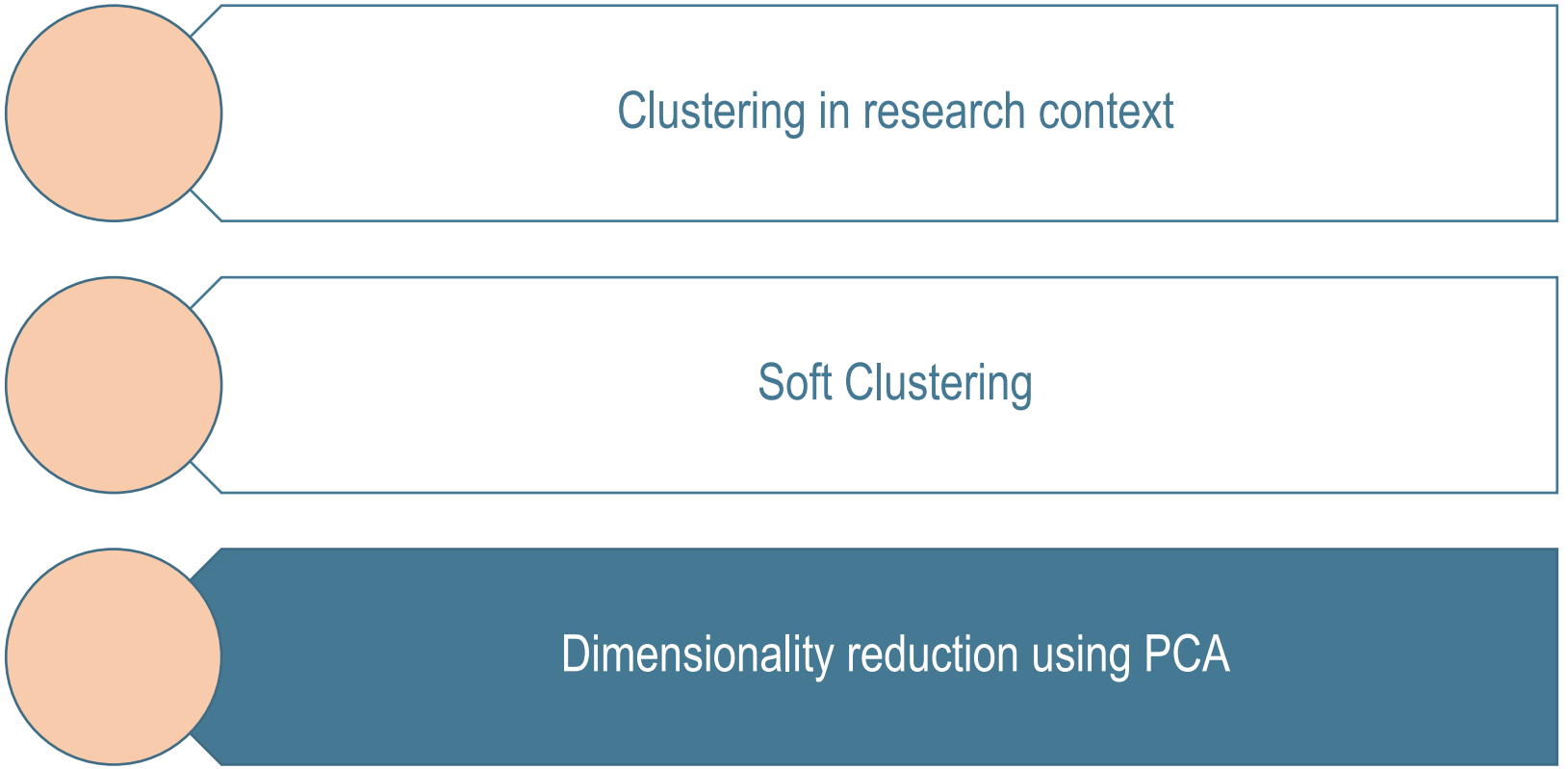
Summary of GMM

- Interpretability: learns a generative model of each cluster
  - you can generate new data based on the learned model
- Relatively efficient: O(tkn), where n is # objects, k is # clusters, and t is # iterations. Normally, k, t << n.
- Intuitive (?) objective function: optimizes data likelihood
- Extensible to other mixture models for other data types
  - e.g. mixture of multinomial for categorical data
  - maximization instead of mean
  - sensitivity to noise and outliers depend on the distribution

Comparison Clustering Methods

	Hierarchical	K-means	GMM
Clustering type	Hard	Hard	Soft
Running time	naively, $O(N^3)$	fastest (each iteration is linear)	fast (each iteration is linear)
Assumptions	requires a similarity / distance measure	strong assumptions	strongest assumptions
Input parameters	none	K (number of clusters)	K (number of clusters)
Clusters	subjective (only a tree is returned)	exactly K clusters	exactly K clusters

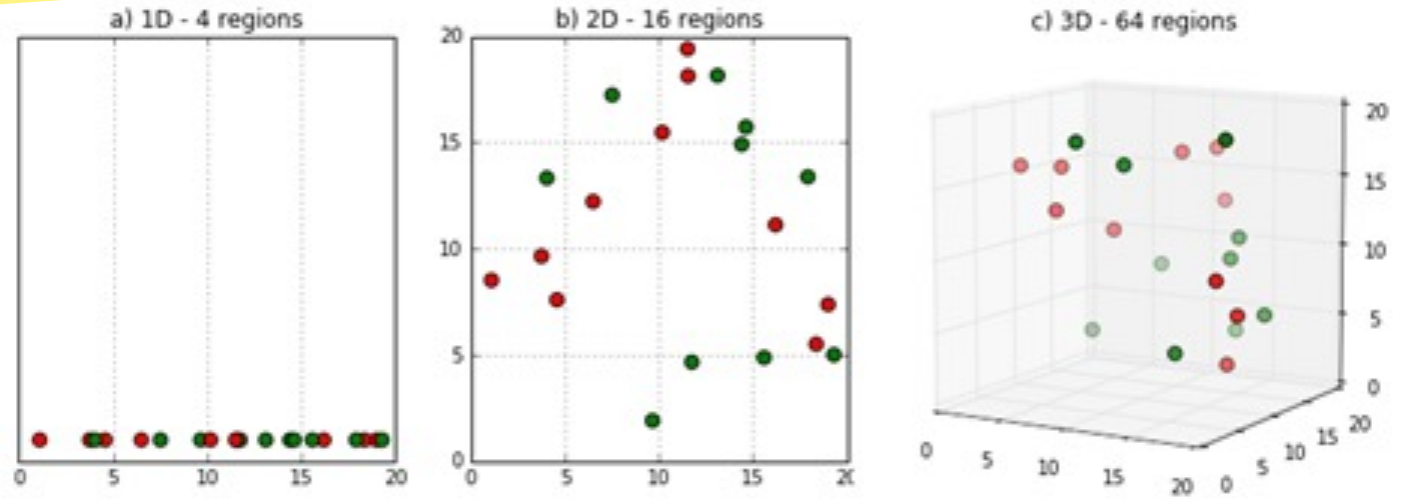
Agenda


Clustering in research context
Soft Clustering
Dimensionality reduction using PCA

The Curse of Dimensionality

- The curse of dimensionality refers to the phenomena that occur when classifying, organizing, and analyzing high dimensional data that does not occur in low dimensional spaces, specifically the issue of data sparsity and “closeness” of data.

- Sparsity of data occurs when moving to higher dimensions. the volume of the space represented grows so quickly that the data cannot keep up and thus becomes sparse, as seen below.



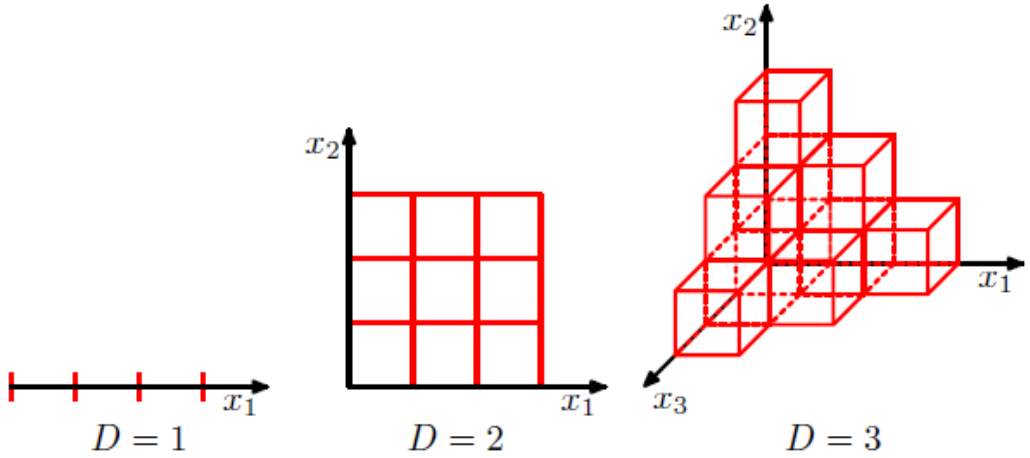
The Curse of Dimensionality

How to classify new data points?

- Divide the feature space into regular cells

Drawback:

- The number of cells increases exponentially with the dimensionality



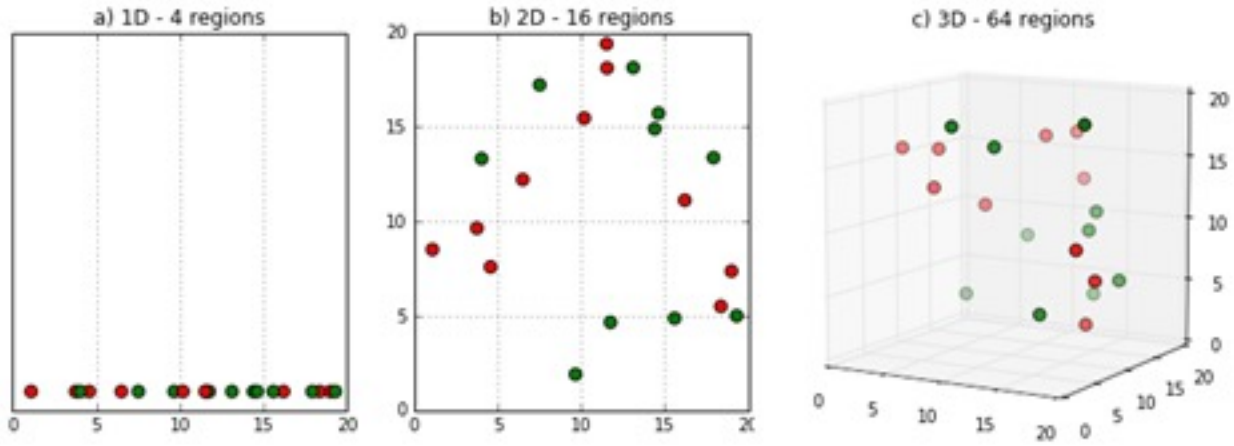
Reference: Bishop (2006)

Implication:

The number of observations needed increases significantly (due to density of observations in individual cells)

The Curse of Dimensionality

- The second issue that arises is related to sorting or classifying the data. In low dimensional spaces, data may seem very similar but the higher the dimension the further these data points may seem to be.
- Most disconcerting, the number of training data needed increases exponentially with each added feature.



In other words…

- If we have more features than observations than we run the risk of massively overfitting our model — this would generally result in terrible out of sample performance.

- When we have too many features, observations become harder to cluster, too many dimensions causes every observation in your dataset to appear equidistant from all the others. And because clustering uses a distance measure such as Euclidean distance to quantify the similarity between observations, this is a big problem. If the distances are all approximately equal, then all the observations appear equally alike (as well as equally different), and no meaningful clusters can be formed.

Application & Example

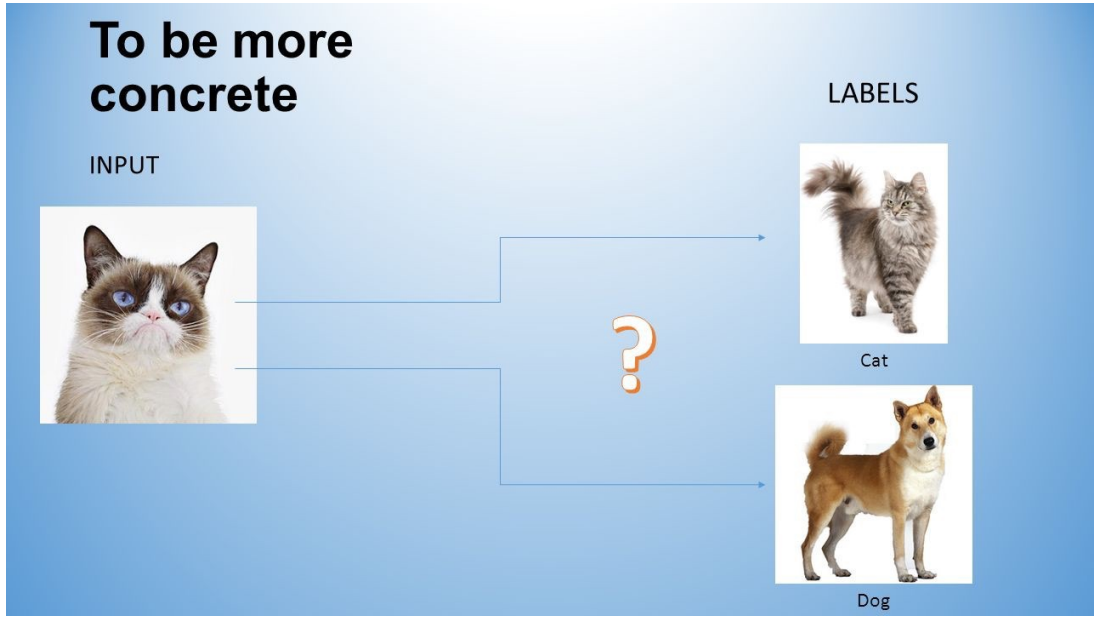
Classification Problem:

Sets of images which either shows a cat or a dog

Idea: Use different colors for classification

Activation Function using average color shares:
red + 0.6*green + 0.4*blue

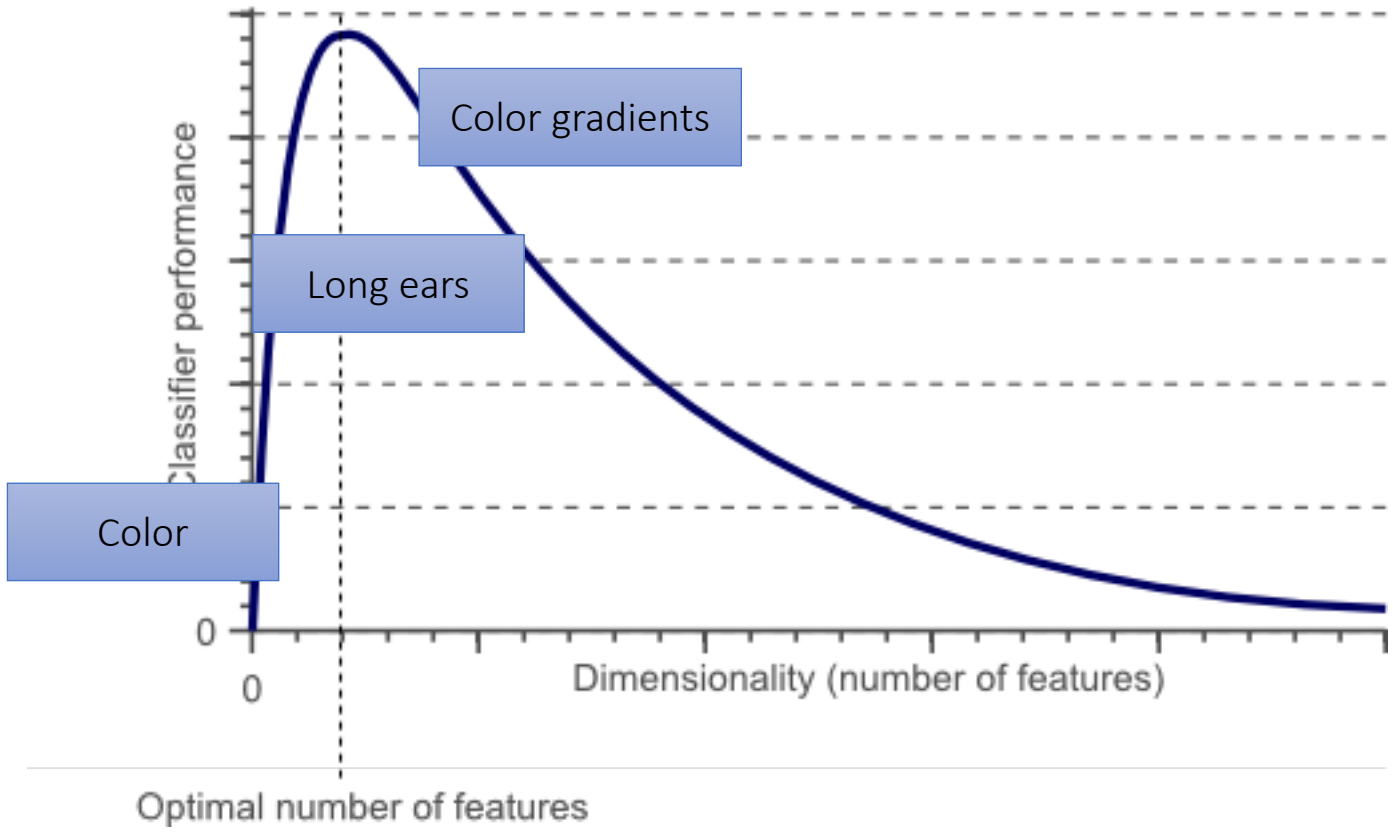
Threshold : 0.8



Application & Example

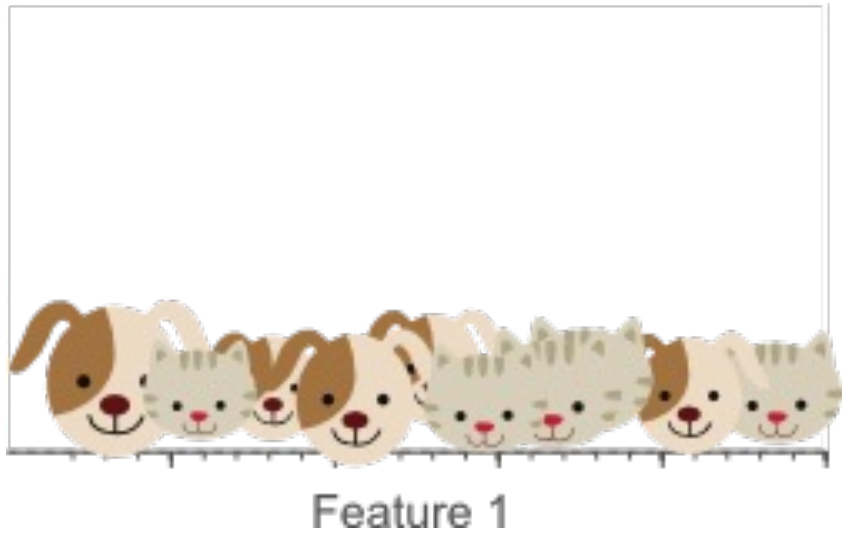
Sufficient explanatory power?

Application & Example



Reference: Computer Vision for Dummies

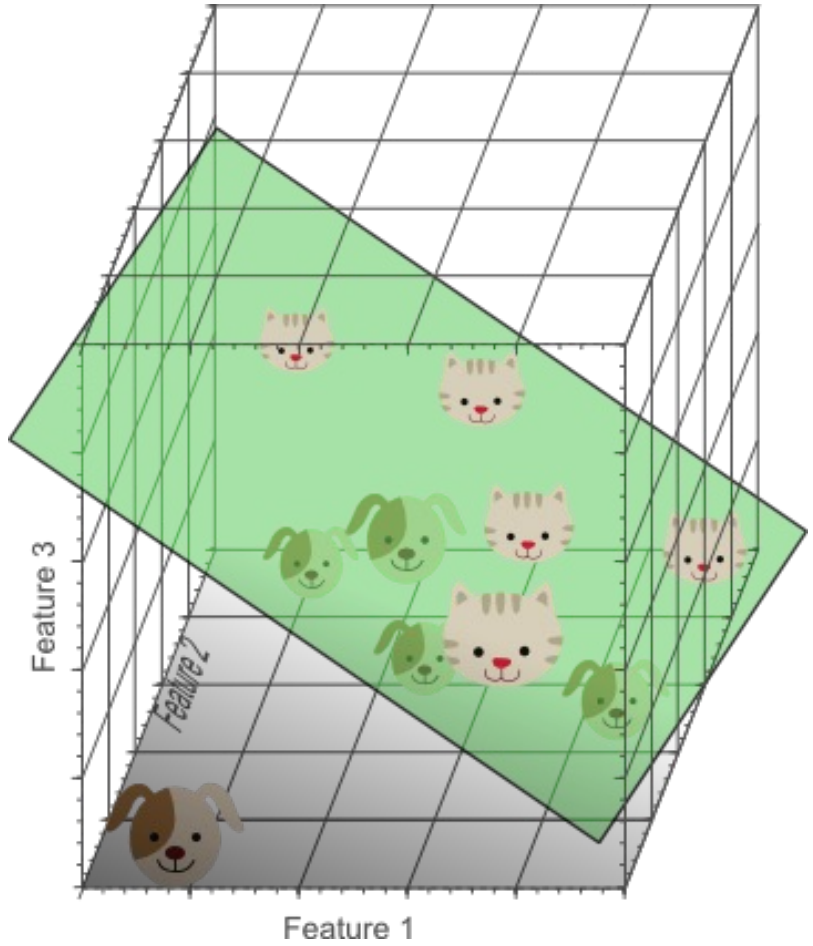
Application & Example



Reference: Computer Vision for Dummies

Only one feature gives weak results

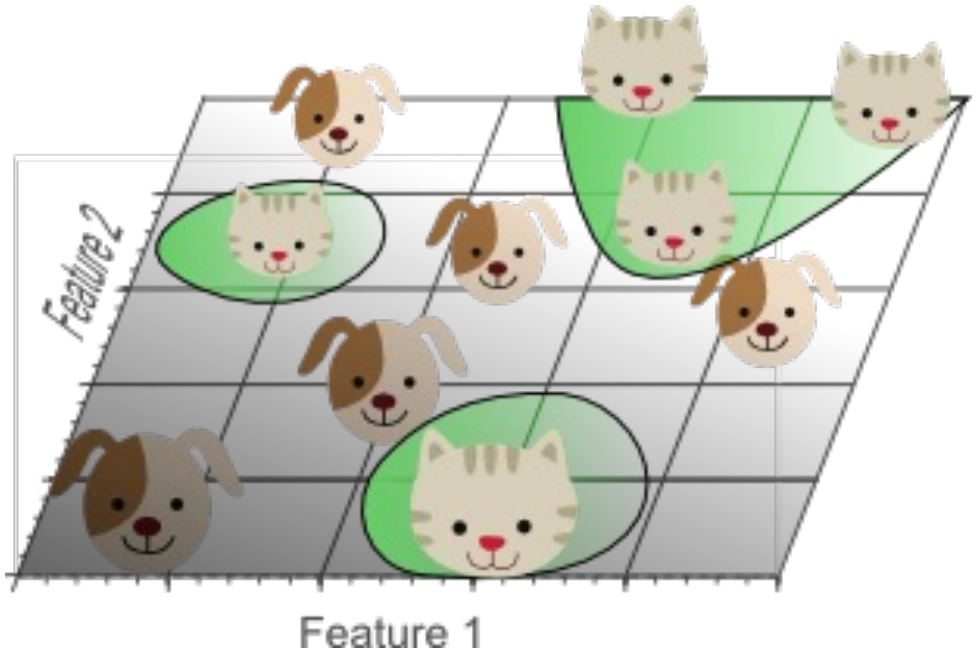
Application & Example



Reference: Computer Vision for Dummies

- There exists a perfect classifier (hyperplane)
- The density of observations, however, decreases exponentially

Application & Example



Reference: Computer Vision for Dummies

Going back to the two-dimensional space, the problem of overfitting becomes obvious:

Exceptions may be learned (few samples in each cell) which are not generalizable

Dimensionality Reduction

Goals:

- Obtain fewer dimensions
- Drop redundant or irrelevant but not important information
- Remove correlations (numerical stability)
- Improved statistical properties enhance post processing quality

Dimension Reduction

Two Approaches:

1. **Feature Selection**
   - Choose a subset of all features
2. **Feature Extraction**
   - Create new features by creating existing ones

Intuitive/Naive Approaches for Feature Selection

- Remove variables with low variance compared to other features
- Decide which features to take based on correlation coefficients
- Filtering: Pretend that only one variable exists and see how well it performs

Drawbacks/Weaknesses?

Feature Extraction

Linear transformations:

- **Principal Component Analysis**
- **Linear Discriminant Analysis (Fisher)**
- Factor Analysis

Nonlinear transformations:

- Kernel-based nonlinear transformations
- Multidimensional scaling

Principal Component Analysis

Two Definitions

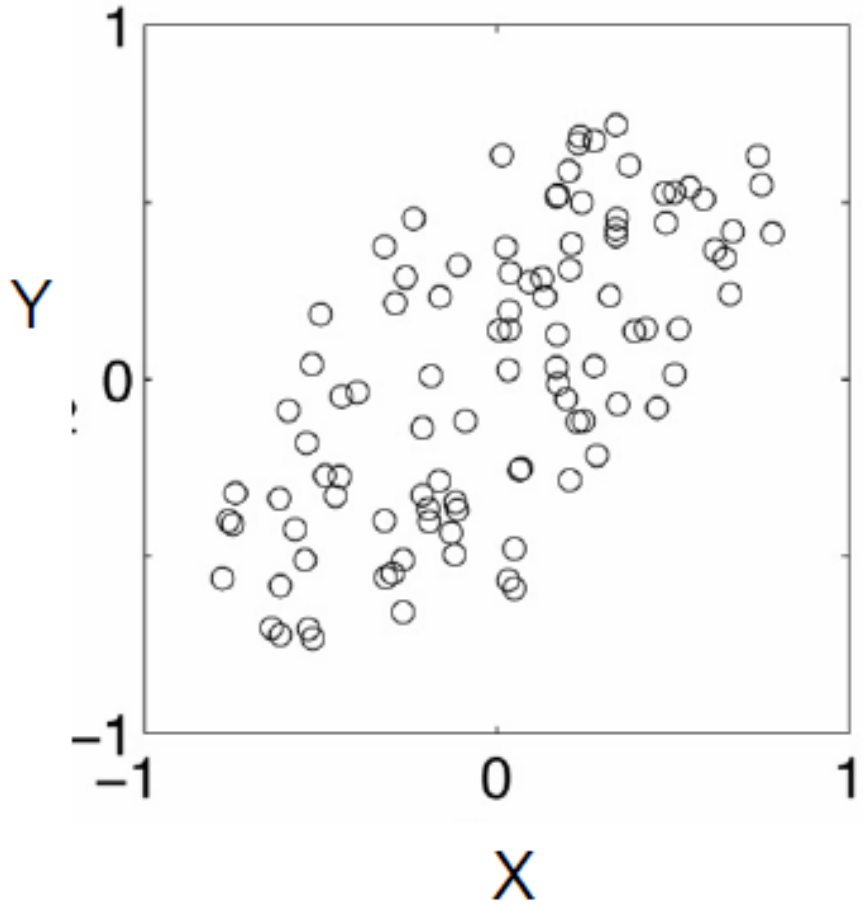
1. Orthogonal projection of data onto a lower dimensional linear space (principal subspace) subject to maximizing the variance of the projected data

2. Linear projection minimizing average projection cost (mean squared distance between data points and projections)

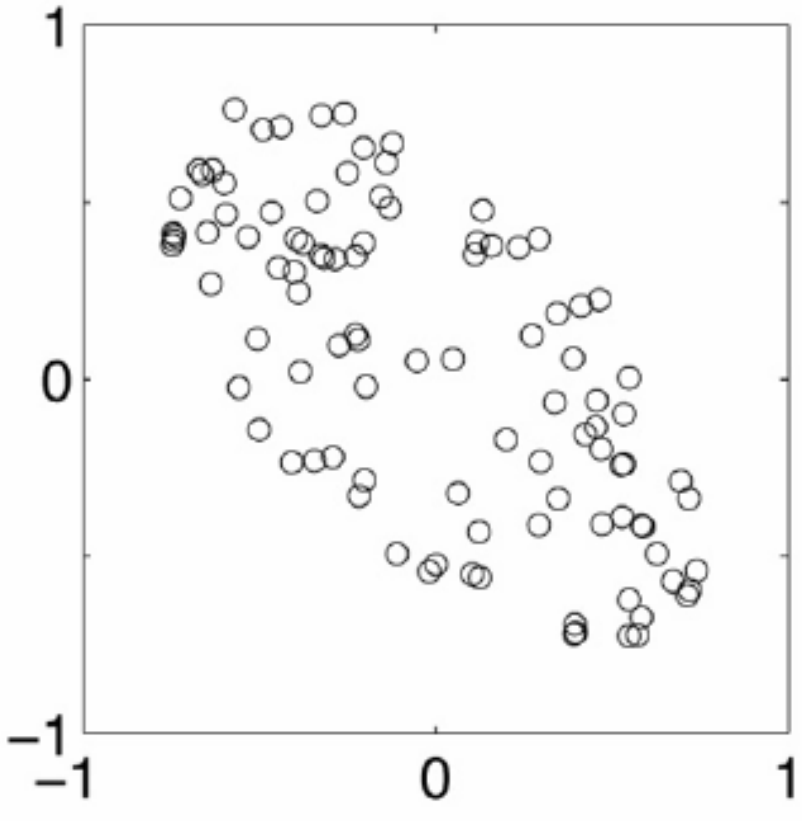
Interpretation: Minimizing information loss when executing projection

Remember

Positive Covariance



Negative Covariance



Reference: PennState College of Engineering

Principal Component Analysis

Mathematically:

- Finding eigenvalues and eigenvectors of the covariance matrix
  1. Find the empirical mean of the data
  2. Compute the covariance matrix
  3. Perform eigenvector decomposition, and select sufficient eigenvectors to preserve the chosen amount of variation in the data

Eigenvalues and eigenvectors of matrices

- Consider the linear transformation of *n*-dimensional vectors defined by an *n* by *n* matrix *A*

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

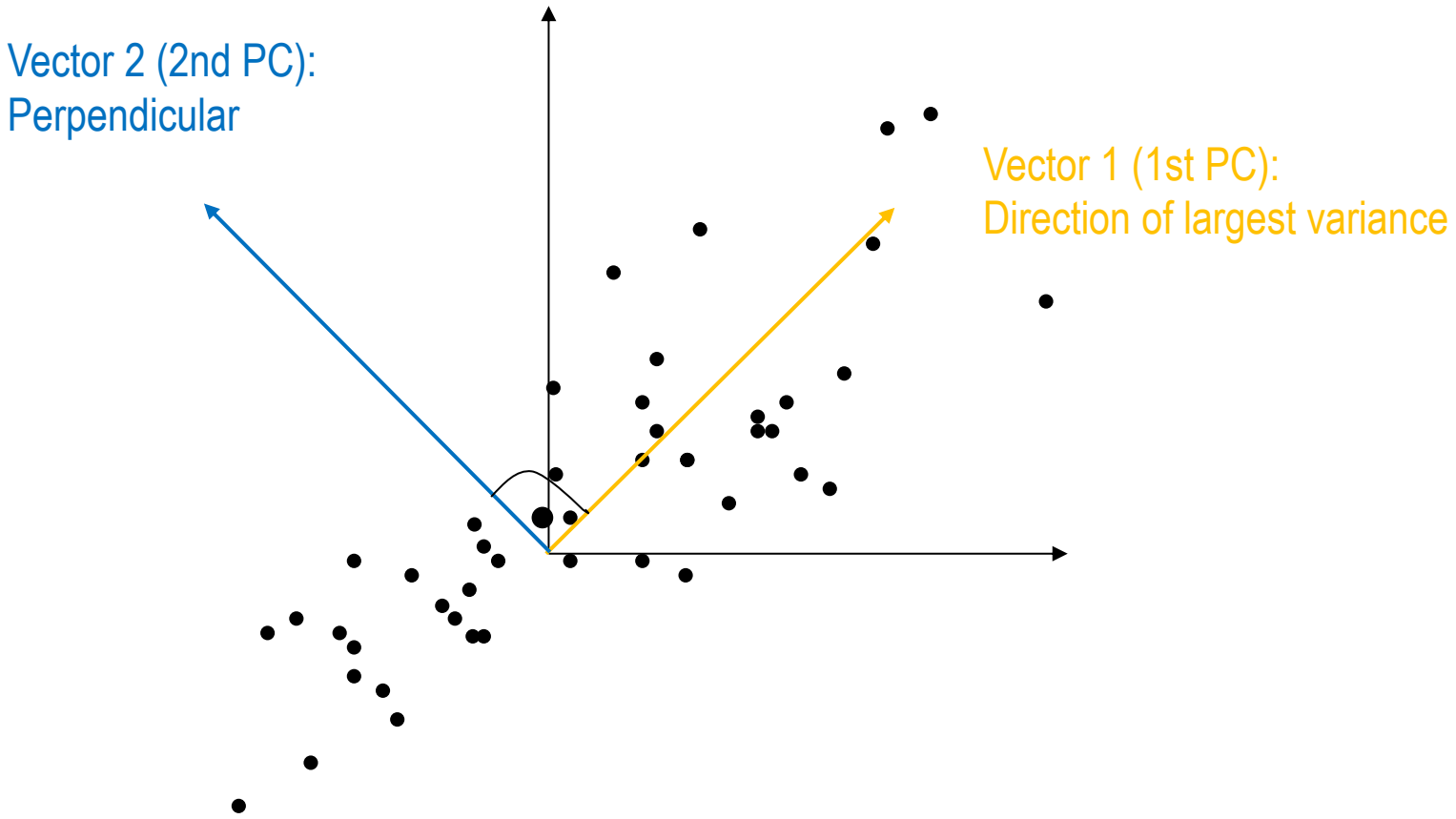
$$w_i = A_{i1}v_1 + A_{i2}v_2 + \cdots + A_{in}v_n = \sum_{j=1}^{n} A_{ij}v_j$$

- If it occurs that *v* and *w* are scalar multiples, that is if

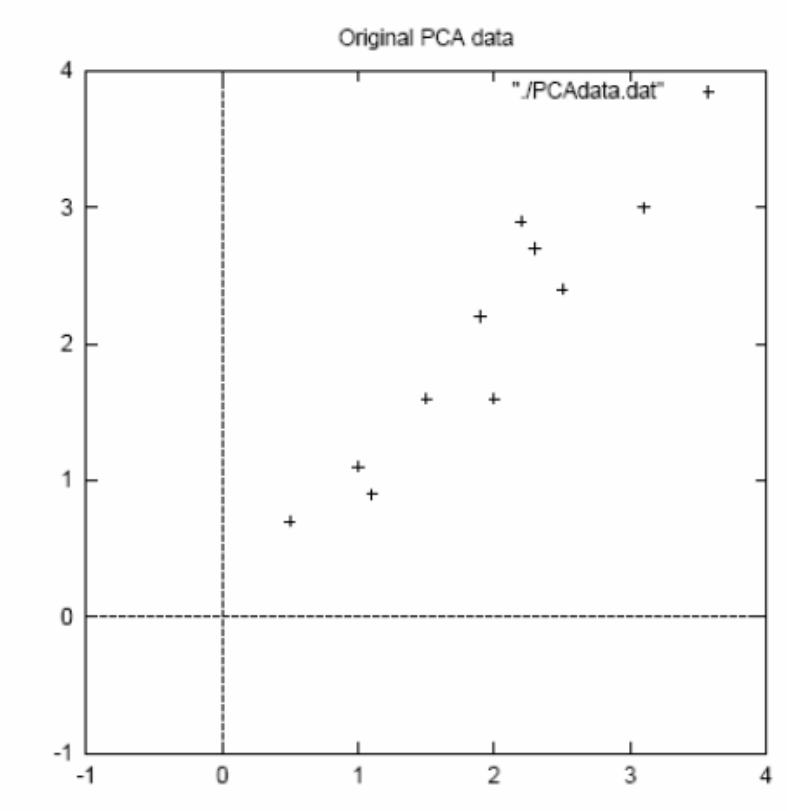
$$A\mathbf{v} = W = \lambda\mathbf{v}$$

- then **v** is an **eigenvector** of the linear transformation *A* and the scale factor *W* is the **eigenvalue** corresponding to that eigenvector. Equation above is the **eigenvalue equation** for the matrix *A*.

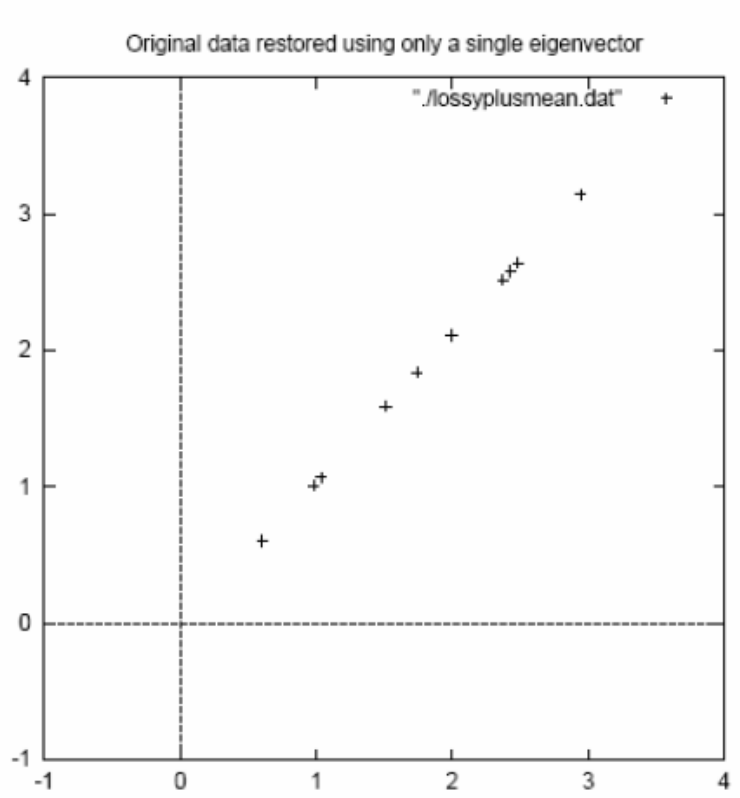
Principal Component Analysis: Intuition



Principal Component Analysis



2D point cloud

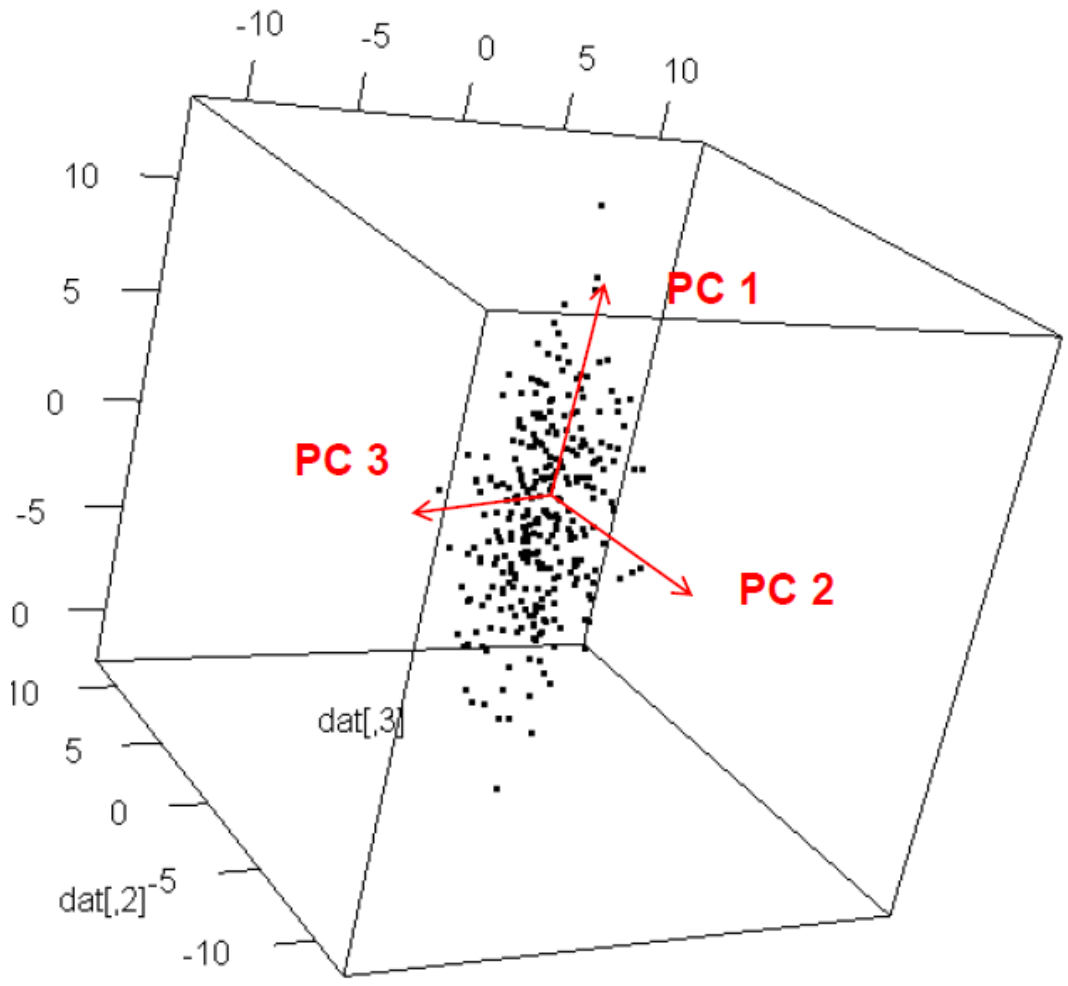


Approximation using one eigenvector basis

Reference: PennState College of Engineering

Principal Component Analysis

- PC1: Direction of largest variance
- PC2: Perpendicular; again direction of largest variance
- PC3: Perpendicular on PC1 & PC2; direction of largest variance



How many PCs?

Only rule of thumbs available:

- E.g. such that at least 85% of variance is captured
- Keep only principal components with variance above mean variance

Linear Feature Extraction

- Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events.

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \xrightarrow{\text{Linear Feature Extraction}} \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_m \end{bmatrix} = \begin{bmatrix} w_{11} & . & w_{1N} \\ w_{21} & . & . \\ . & . & . \\ w_{M1} & . & w_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

Criteria:

- Signal Representation: Represent samples accurately in lower-dimensional space
- **Classification: enhance class-discriminatory information**

Fisher‘s Linear Discriminant Analysis

Target:

Keep as much class discriminatory information as possible

Choose a line (y=wTx) such that separability is maximized

Measure for separability:

Difference between the means normalized by a measure of the within-class scatter (variance)

Contact

For general questions and enquiries on **research**, **teaching**, **job openings** and new **projects** refer to our website at www.is3.uni-koeln.de

For specific enquiries regarding this course contact us by sending an email to the **IS3 teaching** address at is3-teaching@wiso.uni-koeln.de

Follow us on **Twitter** at **@IS3_UniCologne** to stay up to date with recent publication and presentations of our group