

Unsupervised Learning

Simon Günter

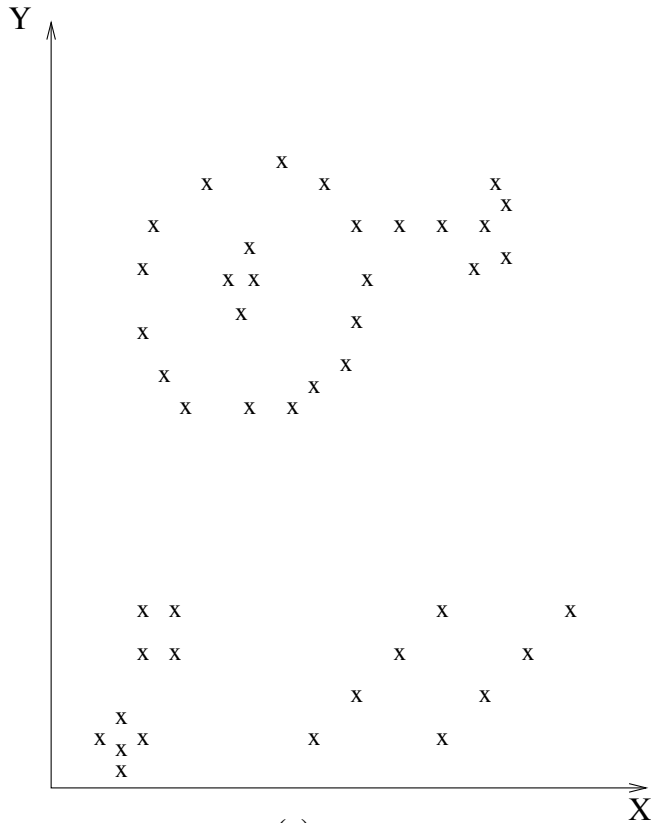
Statistical Machine Learning Program
National ICT Australia
Canberra, 0200 ACT

Based on Douglas Aberdeen's slides

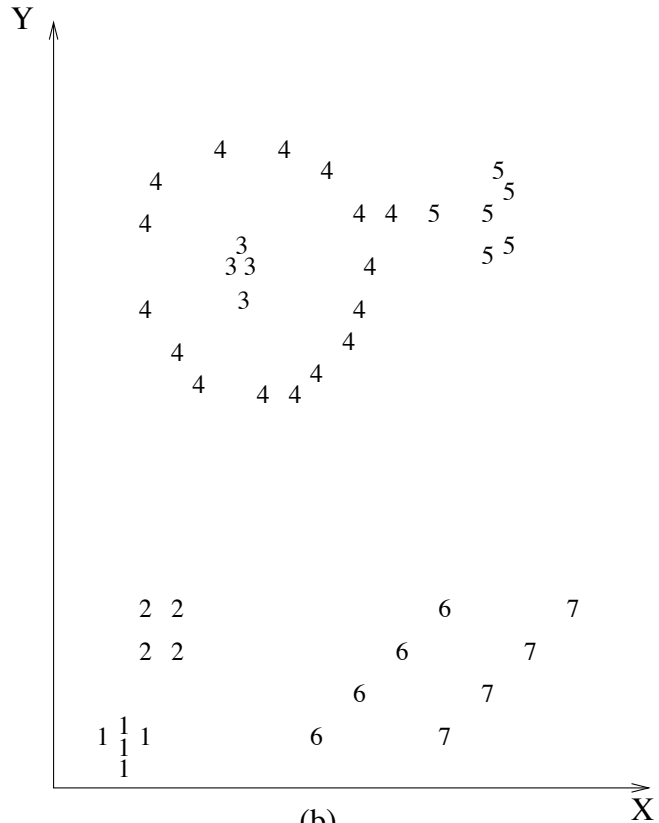
Unsupervised Learning

- Input Data without class labels or target values
- Goal: Extract some underlying structure
 - Grouping of input data → Clustering
 - Find "important" subspace of input space → PCA / ICA
 - Density estimation (how likely is pattern x) → can be used to find outliers
- Difficult to measure quality of result

Clustering Problem Example



(a)



(b)

Clustering

- Cluster elements are "close" to each other
- Types of Clustering
 - Hierarchical (subgroups)
 - 1-Level (no subgroups)
- Clustering algorithms
 - Agglomerative merges clusters, starting with individual points (bottom-up)
 - Divisive starts with one big cluster and splits (top-down)
 - Cluster modification: Element's assignment to cluster varies; also cluster number may vary
- Quality of clustering is unknown or hard to measure

Distance/Similarity Metrics

- Clustering based on notion of (dis)similarity of input data
- Most often euclidean distance is used

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_d (x_d - x'_d)^2} =_{\|\mathbf{x}\|=1, \|\mathbf{x}'\|=1} \frac{1}{\sqrt{2}} \sqrt{1 - \mathbf{x}^\top \mathbf{x}'}$$

- Mahanalobis similarity,

$$s(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \Sigma \mathbf{x}'$$

- Other metrics
- Distance metric changes clustering results a lot
- Metrics on structures: strings, graphs
- Combine feature selection and similarity: **kernels**

$$k(\mathbf{x}, \mathbf{x}')$$

Clustering evaluation

- \mathbf{c}_i is centroid of cluster i , $i \in \{1, \dots, k\}$
- \mathbf{x}_j is data point j , $j \in \{1, \dots, m\}$
- Mean square error (small is good)

$$E(X, k, \mathbf{c}) = \sum_{i=1}^k \sum_{\mathbf{x}_j \in C_i} d(\mathbf{x}_j, \mathbf{c}_i)^2$$

- Cluster separation metric (big is good)

$$\frac{\sum_{i=1}^k \sum_{i'=i}^k d(\mathbf{c}_i, \mathbf{c}_{i'})^2}{\sum_{i=1}^k \sum_{\mathbf{x}_j \in C_i} d(\mathbf{x}_j, \mathbf{c}_i)^2}$$

- Measure strongly depend on number of clusters → selection of cluster number difficult (domain knowledge?)

Algorithm of Agglomerative Clustering

- 1: Each point form an initial cluster
- 2: Compute all distances between two clusters
- 3: **while** more than one cluster left **do**
- 4: Merge closest two clusters
- 5: Update dendrogram tree
- 6: Update distances for merged cluster
- 7: **end while**

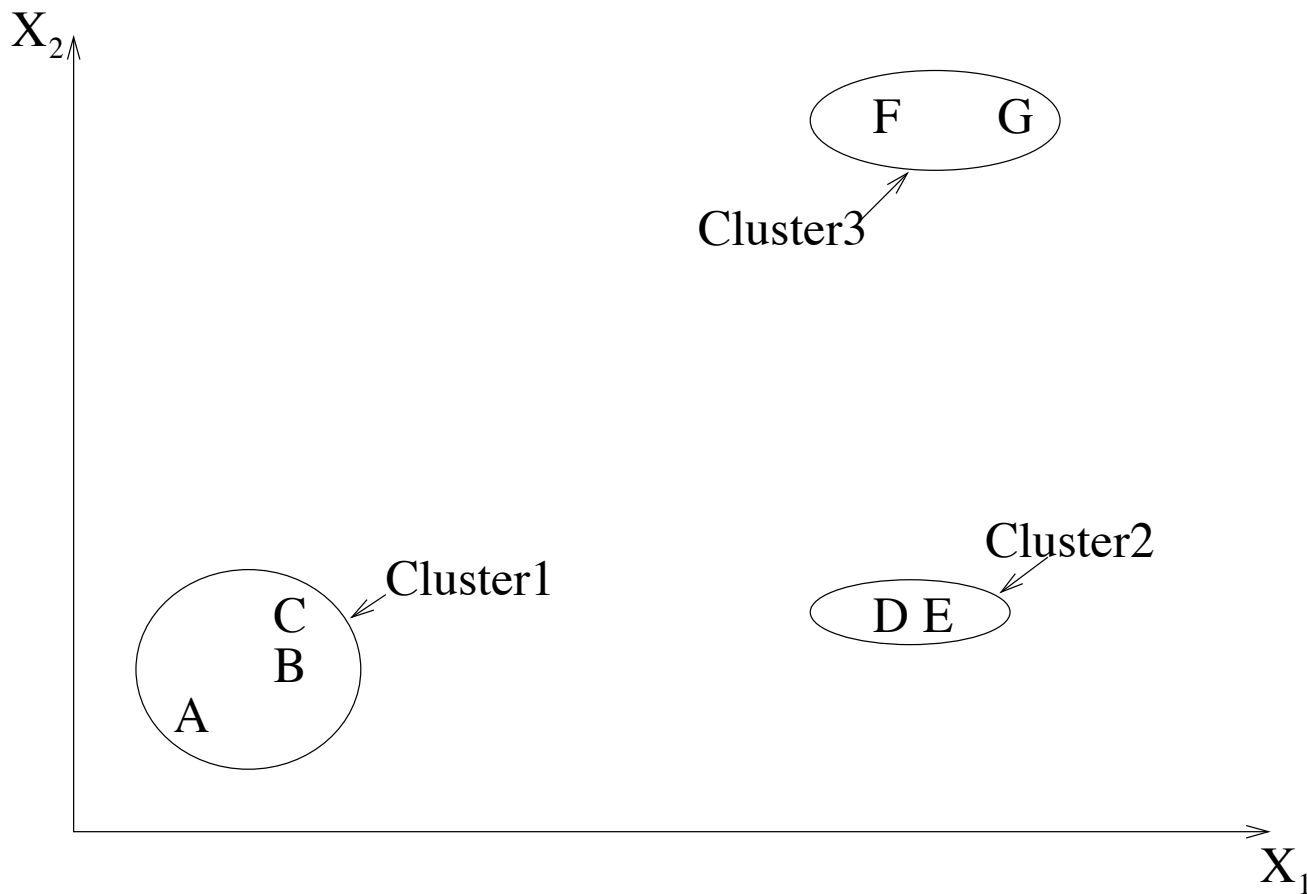
● What is the distance between two clusters ?

Distance between clusters C_1 and C_2

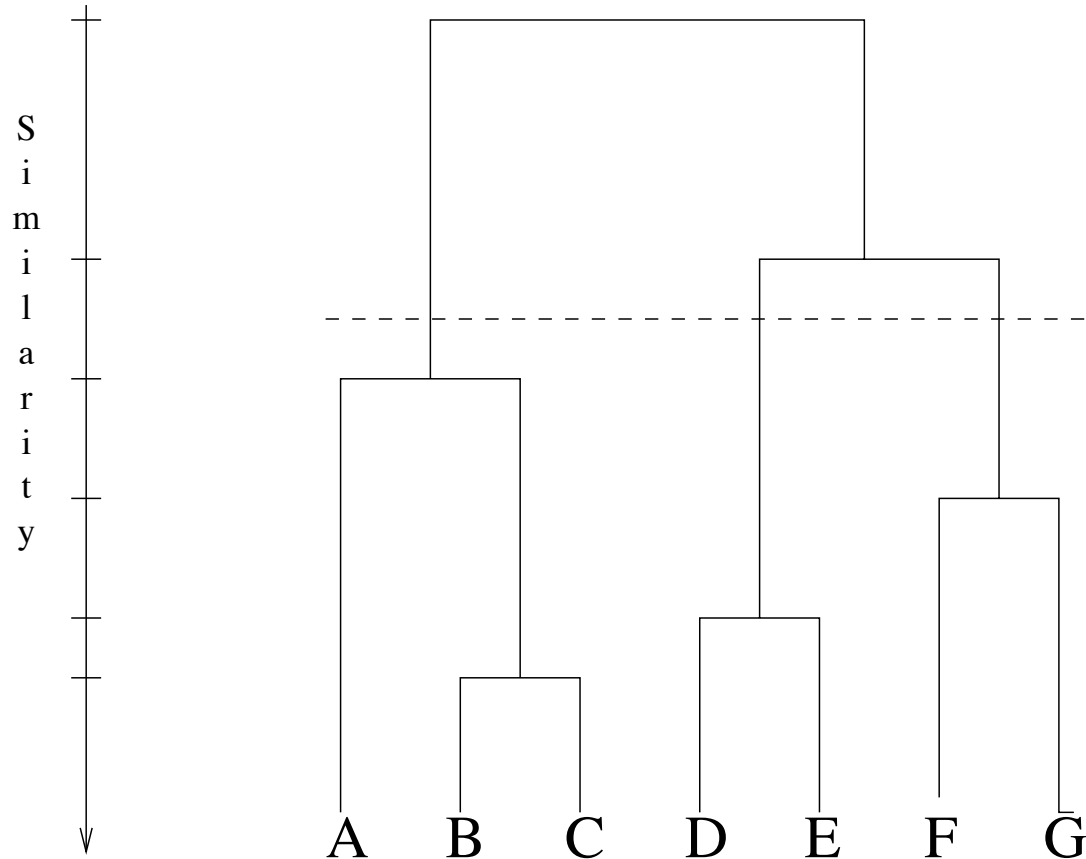
- One representative per cluster \rightarrow use distance measure of input space
 - Median of Cluster
 - Mean of Cluster (euclidean space $\mu = \frac{1}{n} \sum_{i=1}^n x_i$)
- Based on distances $d(x_i, x_j)$ where $x_i \in C_1$ and $x_j \in C_2$
 - **Complete Link Algorithm:** $d(C_1, C_2) = \max_{i,j} d(x_i, x_j)$
(compact clusters, more useful hierarchies)
 - **Single Link Algorithm:** $d(C_1, C_2) = \min_{i,j} d(x_i, x_j)$
(learns complex clusters)
 - **Average Link Algorithm:** $d(C_1, C_2) = \frac{\sum_{i,j} d(x_i, x_j)}{|C_1| \cdot |C_2|}$

Prune tree by cluster distances to get final clustering

Agglomerative Clustering Example



Dendrogram



k -means

- Iterative algorithm which moves cluster centers until no change
- Simple and fast, therefore common
- An instance of EM
- Requires input k : number of clusters
- Local convergence, depends on initial cluster centres
- Usually produces hyper-spheric clusters
- Result is non-hierarchical clustering

k -means cont.

```
1: Initialise  $k$  cluster centres randomly
2: while any  $\mathbf{x}$  changes cluster do
3:   for each point  $\mathbf{x}_j$  do
4:     assign  $\mathbf{x}_j$  to  $\arg \min_i d(\mathbf{x}_j, \mathbf{c}_i)$ 
5:   end for
6:   for each cluster center  $\mathbf{c}_i$  do
7:     Let  $X_i =$  set of points assigned to cluster  $i$ 
8:      $\mathbf{c}_i = \frac{1}{|X_i|} \sum_{\mathbf{x}_j \in X_i} \mathbf{x}_j$ 
9:   end for
10: end while
```

- Clever ways to initialise cluster centres
- Cluster median for non euclidean input space
- Clever distance calculation

Choosing k

- k -means will find clusters even if they don't exist
- Choice of k is important
- Can choose automatically
 - Split if variance inside cluster is too high
 - Merge if centroids too close
 - Distances and variances are measured relative to other clusters / whole data set

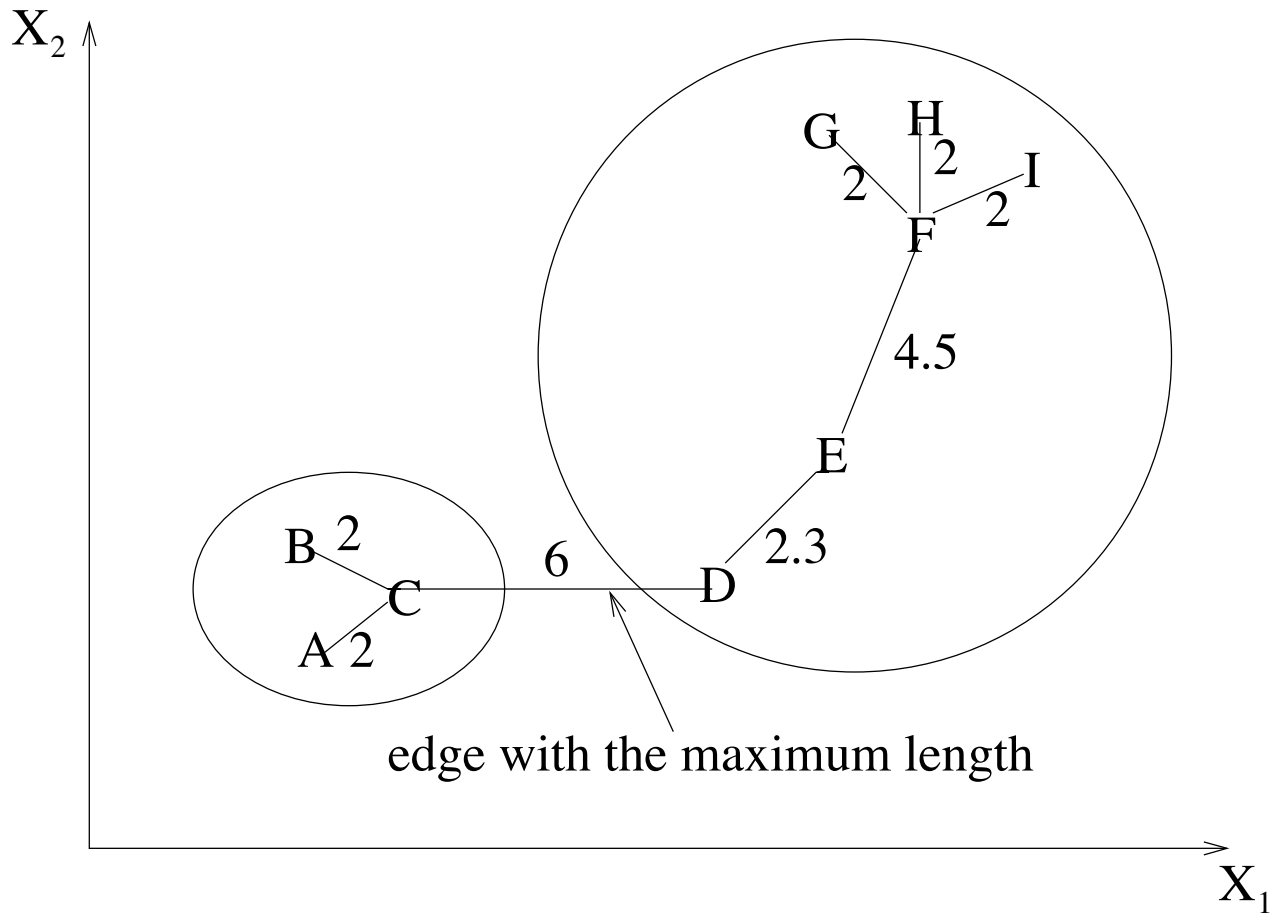
Minimum Spanning Tree (MST)

- Graph theoretic clustering
- Minimum spanning tree:
Shortest set of links to connect all vertices
- Combination of a distance measure and MST algorithm
- Each data point is treated as a vertex in the graph
- Runs in $O(m^2)$ time
- Hierarchical
- Produces a tree with distance pruning interpretation

MST Cont.

- 1: **while** Any unconnected data points **do**
 - 2: Join two closest points x, x' with edge
 - 3: **while** $x \text{---} x'$ induces a loop **do**
 - 4: Try next two closest points
 - 5: **end while**
 - 6: **end while**
 - 7: Remove longest edges to produce clusters
- Each removal of an edge is a split of a cluster
 - Related to agglomerate algorithms
 - No cluster distances defined
 - Related to Nearest neighbour algorithms

MST Example



Nearest Neighbour Clustering

```
1: Select distance threshold  $t$ 
2: All  $\mathbf{x}$  start unassigned
3: for each  $\mathbf{x} \in X$  do
4:     Find neighbour  $\mathbf{x}_{NN} = \arg \min_{\mathbf{x}' \in X} d(\mathbf{x}, \mathbf{x}')$ 
5:     if  $d(\mathbf{x}, \mathbf{x}_{NN}) > t$  then
6:         assign  $\mathbf{x}$  to a new cluster
7:     else
8:         assign  $\mathbf{x}$  to the same cluster as  $\mathbf{x}_{NN}$ 
9:     end if
10: end for
```

- Good for online implementation
- Automatically chooses number of clusters k
- Strongly depends on order of input data and t

Other Clustering Methods

- k -means with simulated annealing
- Neural Networks (gradient descent)
 - Combine feature selection and cluster assignment
 - Kohonen's Self Organising Feature Map
- Genetic Algorithms
 - Swap cluster point assignments
 - Mutate cluster centroids
 - Make sure you have appropriate fitness function

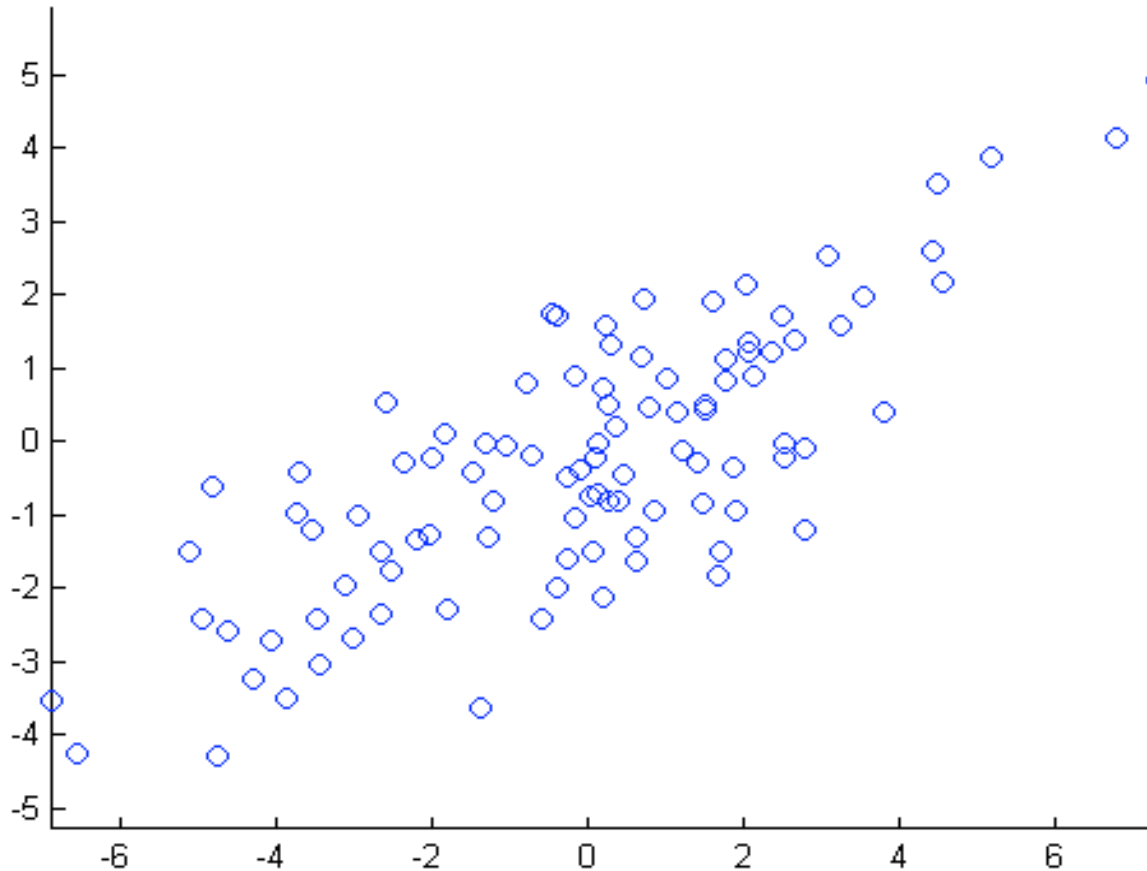
Principal Component Analysis (PCA)

- Input data: $X = x_1, \dots, x_m \in \mathbb{R}^n$
- Find the orthogonal directions v_1, \dots, v_r in \mathbb{R}^n with highest variance
- This leads to
 - V subspace spanned by v_1, \dots, v_r

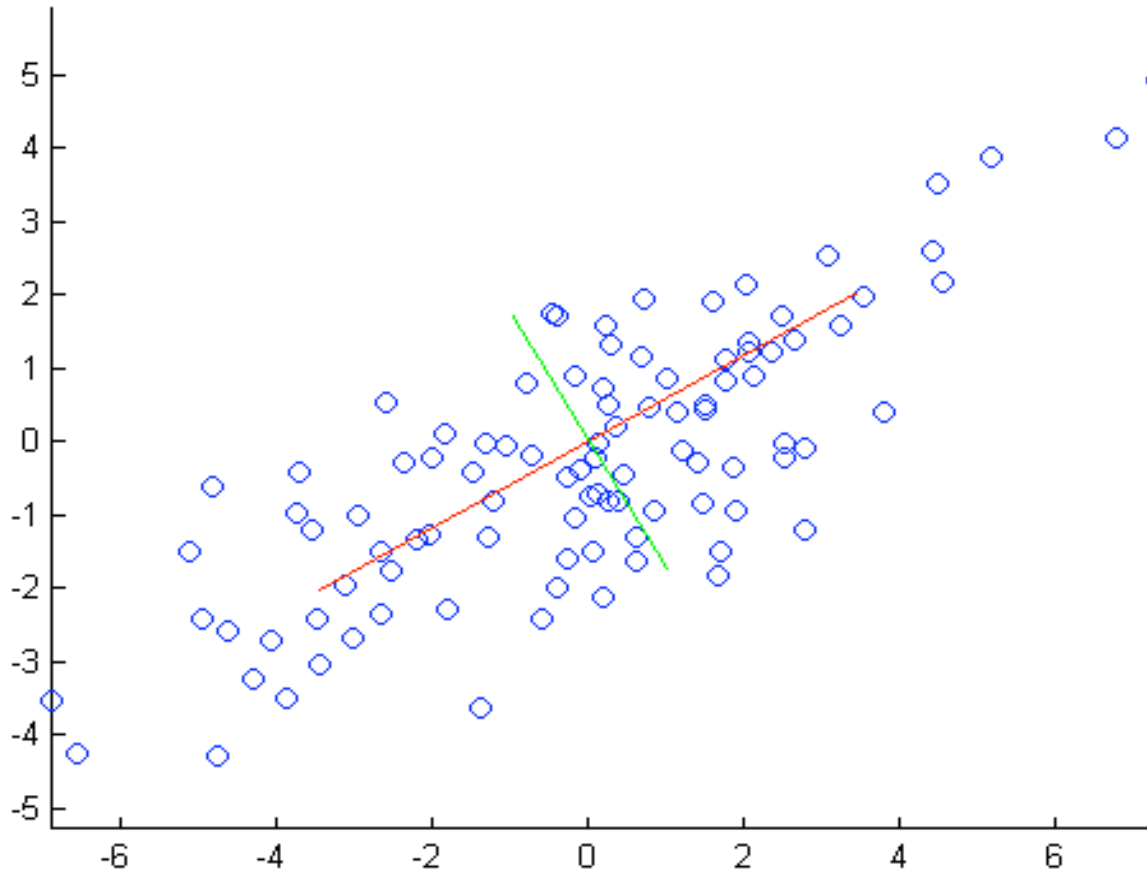
$$V = \left\{ v \in \mathbb{R}^n \mid \exists a_1, \dots, a_r \quad v = \sum_{i=1}^r a_i \cdot v_i \right\}$$

- Error of projection of input data on V is minimal for all subspaces of $\dim(V)$
- Values of a_i and a_j are not correlated (for X)

PCA example (1)



PCA example (2)



PCA Applications

- Data compression (only values a_i must be saved)
- Noise elimination. All information is assumed to be in V and rest is noise
- Decorrelation: Some models, like naive Bayes, assume uncorrelated features
- Example noisy samples $[X, X]$:
 $x_1 = [0.95, 1], x_2 = [1.5, 1.5], x_3 = [2.1, 2]$
- Reconstruction using only 1 PCA:
 $x_1 = [0.98492, 0.96434], x_2 = [1.51567, 1.48400],$
 $x_3 = [2.07194, 2.02866]$
- Results are closer to the diagonal (noise was removed)

PCA Solution

- Input matrix \mathbf{X} with x_1, \dots, x_m as rows
- Singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ (related to eigenvalue decomposition)
- \mathbf{U} and \mathbf{V} are orthogonal matrices
- \mathbf{D} is diagonal matrix with $\forall i < j \quad \mathbf{D}[i, i] \geq \mathbf{D}[j, j]$
- $d_i = \mathbf{D}[i, i]$ are called singular values
- Solution to PCA are the r first rows of \mathbf{V}
- time complexity is $O(m^3)$

Independent Component Analysis (ICA)

- Input data: $X = x_1, \dots, x_m \in \mathbb{R}^n$
- Find the directions v_1, \dots, v_r in \mathbb{R}^n which are independent and have highest variance
- Independence of v_1 and v_2 : distribution of coefficient for v_1 is same regardless of value of the coefficient for v_2
- Most famous application: Blind Source Separation (BSS)
- PCAs are often very different from ICAs (example on next slide)
- No analytical solution
- Algorithms minimize dependence of v_i (e.g. measured by mutual information)

PCA vs ICA

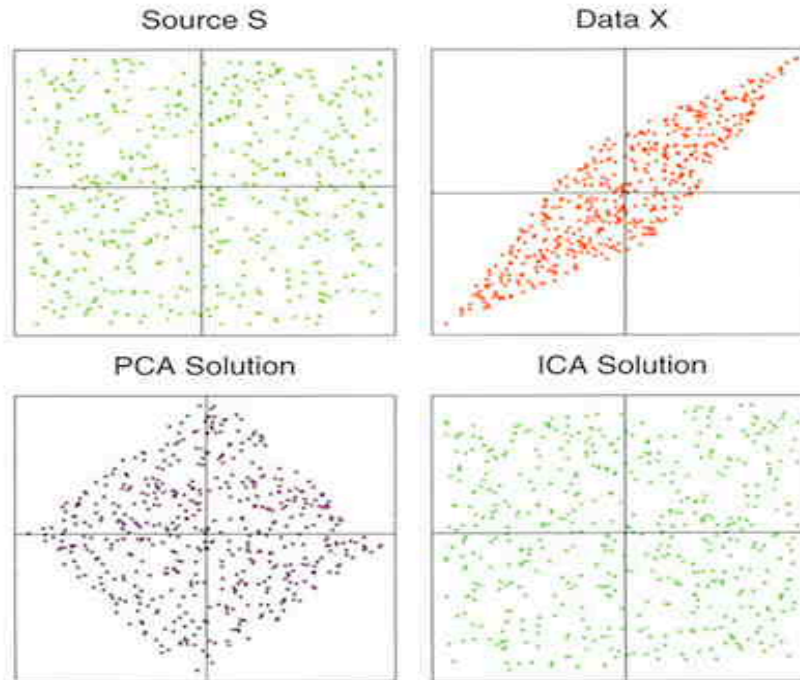


FIGURE 14.28. *Mixtures of independent uniform random variables. The upper left panel shows 500 realizations from the two independent uniform sources, the upper right panel their mixed versions. The lower two panels show the PCA and ICA solutions, respectively.*

copied from "The Elements of Statistical Learning" (Springer)

Blind Source Separation (BSS)

- Have K sources $\{s_k[t]\}$, and K signals $\{x_k[t]\}$. Both $\{s_k[t]\}$ and $\{x_k[t]\}$ are time series (t is a discrete time index)
- Each signal is a linear mixture of the sources $x_k[t] = \mathbf{A}s_k[t] + n_k[t]$ where $n_k[t]$ is the noise contribution in the k -th signal $x_k[t]$, and \mathbf{A} is a mixture matrix
- The problem: given $x_k[n]$, determine \mathbf{A} and $s_k[n]$
- Solution: We assume sources are independent and use ICA
- The Cocktail Party: We want to separate individual voices from a cocktail party

Density estimation

- Input data $T = \{x_1, \dots, x_m\}$
- Measure of how likely a pattern x is, given T
- x_1, \dots, x_m are samples from random variable X
- Problem: $P(x = X) = 0$ for continuous x
- Use probability density $p(x) := \lim_{\epsilon \rightarrow 0} \frac{P(x \leq X < x + \epsilon)}{\epsilon}$
- Naive approach: Use the empirical density

$$p_{emp}(x) = \frac{1}{m} \sum_{i=1}^m \delta(x, x_i)$$

$$\forall x \neq x_0 \quad \delta(x, x_0) = 0 \quad \text{and} \quad \forall \epsilon > 0 \quad \int_{x_0 - \epsilon}^{x_0 + \epsilon} \delta(x, x_0) dx = 1$$

- Problem: $p_{emp}(x) = 0$ for $x \notin T$

Parzen Windows

- Idea: Smear out p_{emp} by convolving it with a kernel $k(x, x_0)$

$$p_{emp}(x) = \frac{1}{m} \sum_{i=1}^m k(x, x_i)$$

- Restriction to k : $\int_{\mathbb{R}^n} k(x, x') dx' = 1 \quad \forall x \in \mathbb{R}^n$
- Kernel should have a parameter to modify its "width" w
- If $\|x - x'\| < w$ then $k(x, x')$ is not too close to 0

Kernel Types

- Gaussian Kernel

$$k(x, x') = (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right)$$

- Laplacian Kernel

$$k(x, x') = \lambda 2^{-n} \exp(-\lambda \|x - x'\|_1)$$

- Indicator Kernel

$$k(x, x') = \begin{cases} \frac{1}{w^n} & \text{if } \forall i \in \{1, \dots, n\} \quad -\frac{1}{2w} \leq x'_i - x_i \leq \frac{1}{2w} \\ 0 & \text{otherwise} \end{cases}$$

- Width (parameters σ, λ, w) of the kernel is usually much more important than type.

Selection of Kernel Width

- We need a method for adjusting the kernel width
- The likelihood of the input data increases for small width
→ not good measure
- Possible Solution:
 - Check the likelihood of the density estimate on an un-seen part of the data
 - Select width for which we obtained highest likelihoods
- Density estimation approach without kernel
 - Assume distribution (e.g. Gaussian) and optimize its parameters on input data

Application: Novelty Detection

- Goal: Find the least likely observations x_i from a dataset X .
- Perform density estimate $p(x)$ using X and declare all x_i with $p(x_i) < p_0$ as novel.
- Note that $p(x_i) \geq \frac{1}{m}k(x_i, x_i) \rightarrow$ choice of p_0 not so easy
 \rightarrow Do not use x_i for calculating $p(x_i)$
- Also the likelihood $p(x')$ of a new pattern $x' \notin X$ can be calculated
- Application: Jet Engine Failure Detection
 - We don't know exactly how engines fail
 - .. yet the sensor data will be somehow different