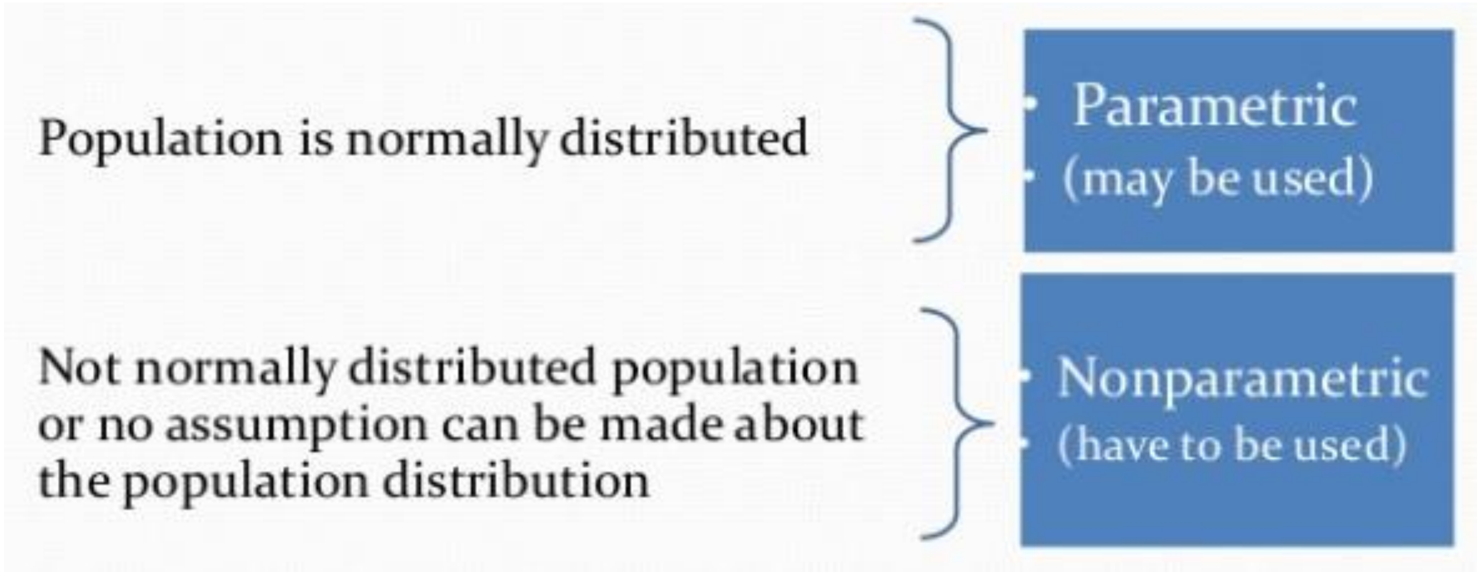
Pattern Recognition

Krystian Mikolajczyk

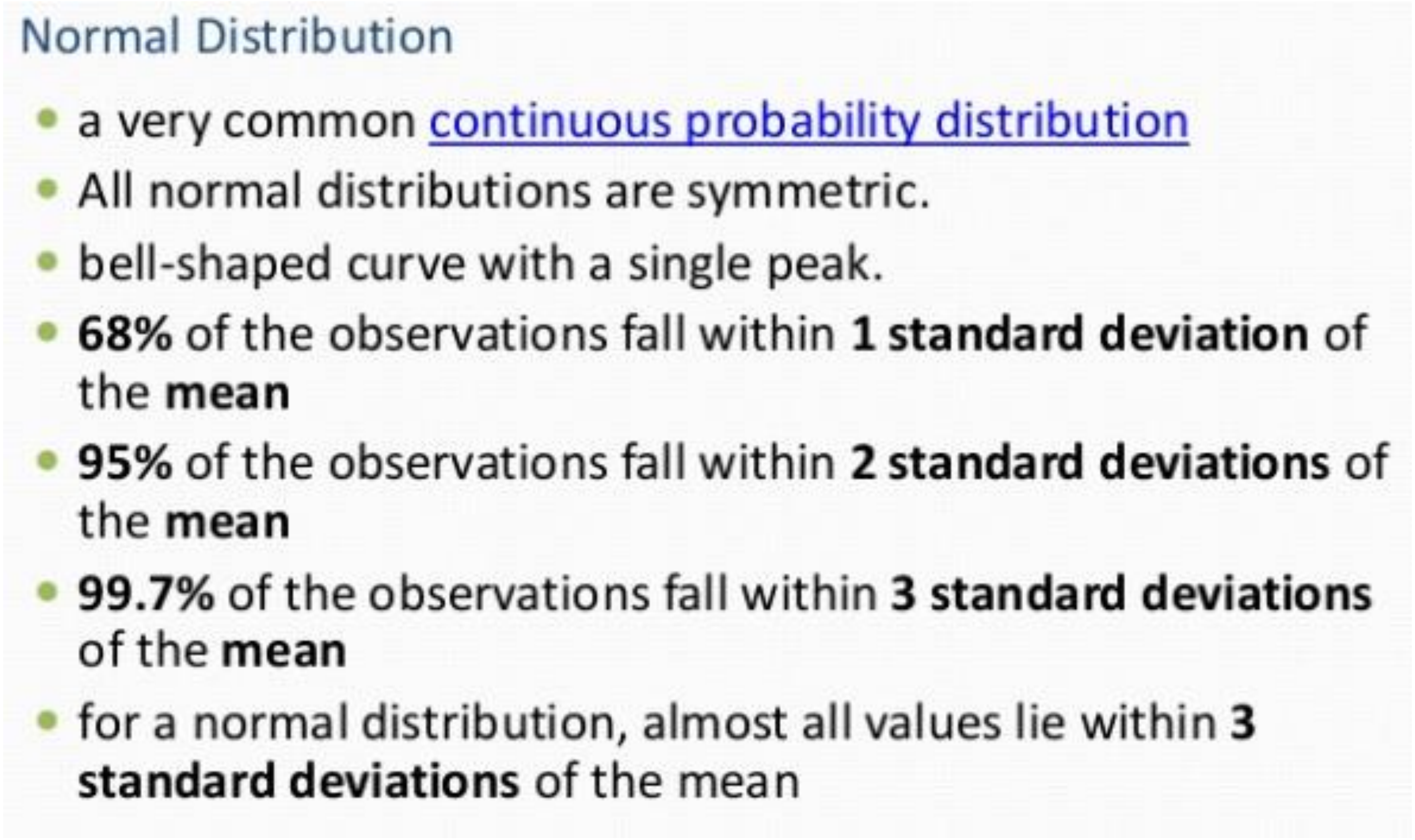
Blackboard

Data representation

Parametric vs non parametric


Population is normally distributed
Parametric
(may be used)
Not normally distributed population or no assumption can be made about the population distribution
Nonparametric
(have to be used)

Parametric vs non parametric



Normal Distribution

- a very common continuous probability distribution
- All normal distributions are symmetric.
- bell-shaped curve with a single peak.
- **68%** of the observations fall within **1 standard deviation** of the **mean**
- **95%** of the observations fall within **2 standard deviations** of the **mean**
- **99.7%** of the observations fall within **3 standard deviations** of the **mean**
- for a normal distribution, almost all values lie within **3 standard deviations** of the mean

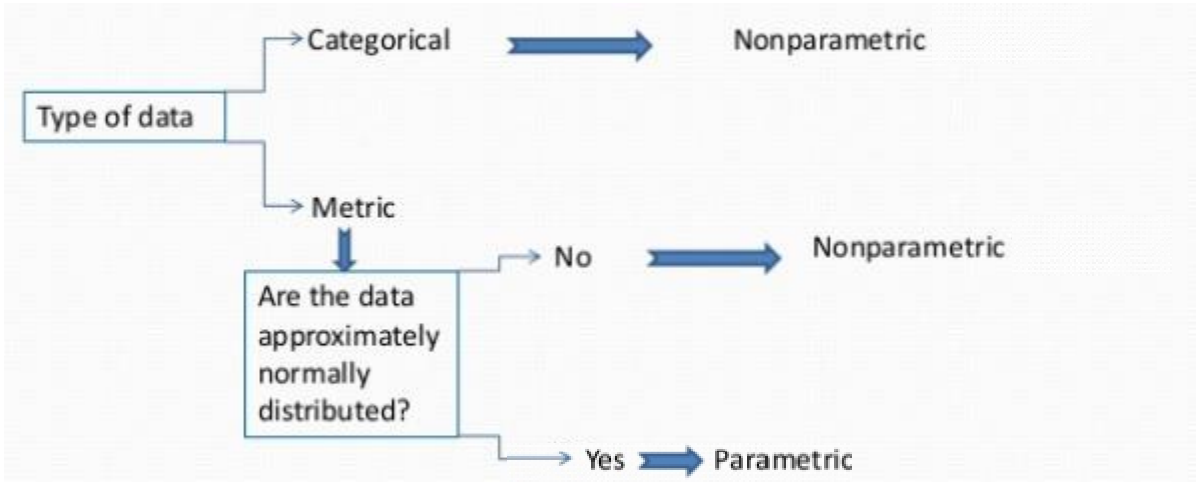
Parametric vs non parametric

- In cases where
  - the data which are measured by interval or ratio scale come from a normal distribution *

    parametric are used.
- In cases where
  - the data is nominal or ordinal
  - the assumptions of parametric tests are inappropriate

    nonparametric are used.

* - simplifying assumption, other distributions are possible



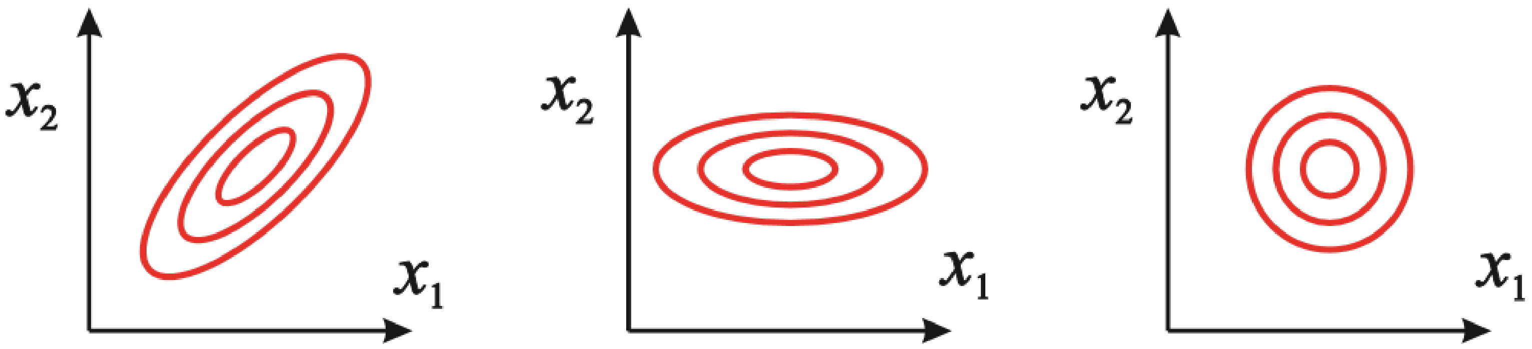
Parametric

- Multivariate Gaussian

$$\mathcal{N}(\mathbf{x}|\underset{\text{mean}}{\boldsymbol{\mu}}, \underset{\text{covariance}}{\boldsymbol{\Sigma}}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

$$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu}_{\mathrm{ML}})(\mathbf{x}_n-\boldsymbol{\mu}_{\mathrm{ML}})^{\mathsf{T}}$$



Parametric – Gaussian mixture

- Linear super-position of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Normalization and positivity require

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

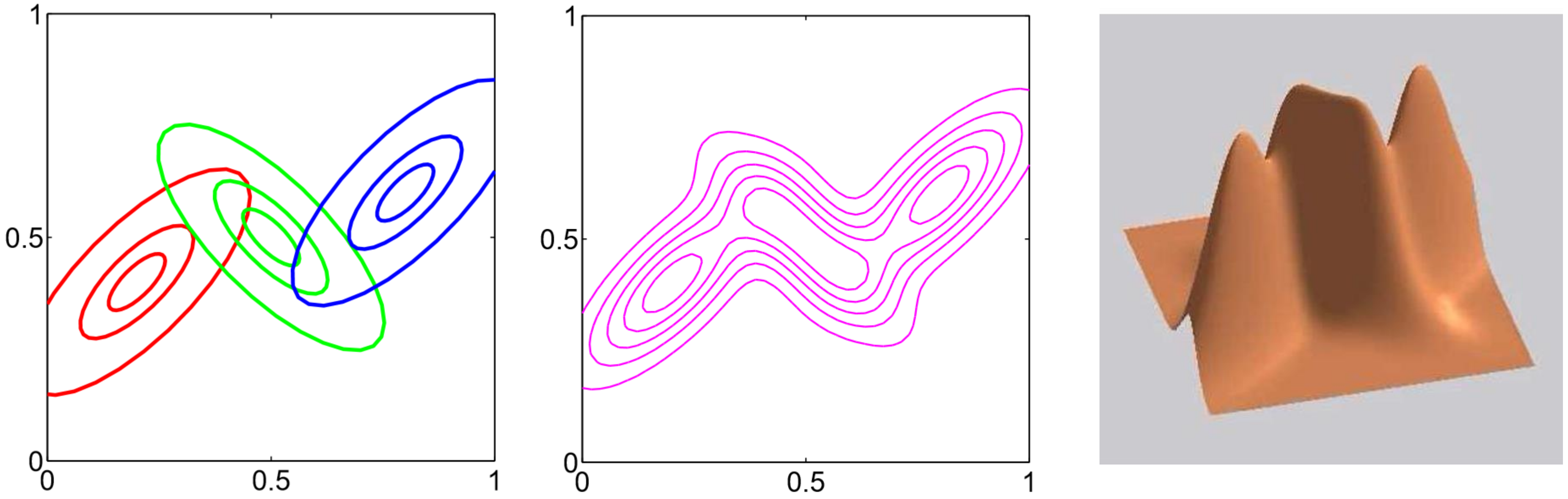
- Can interpret the mixing coefficients as prior probabilities

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k)$$

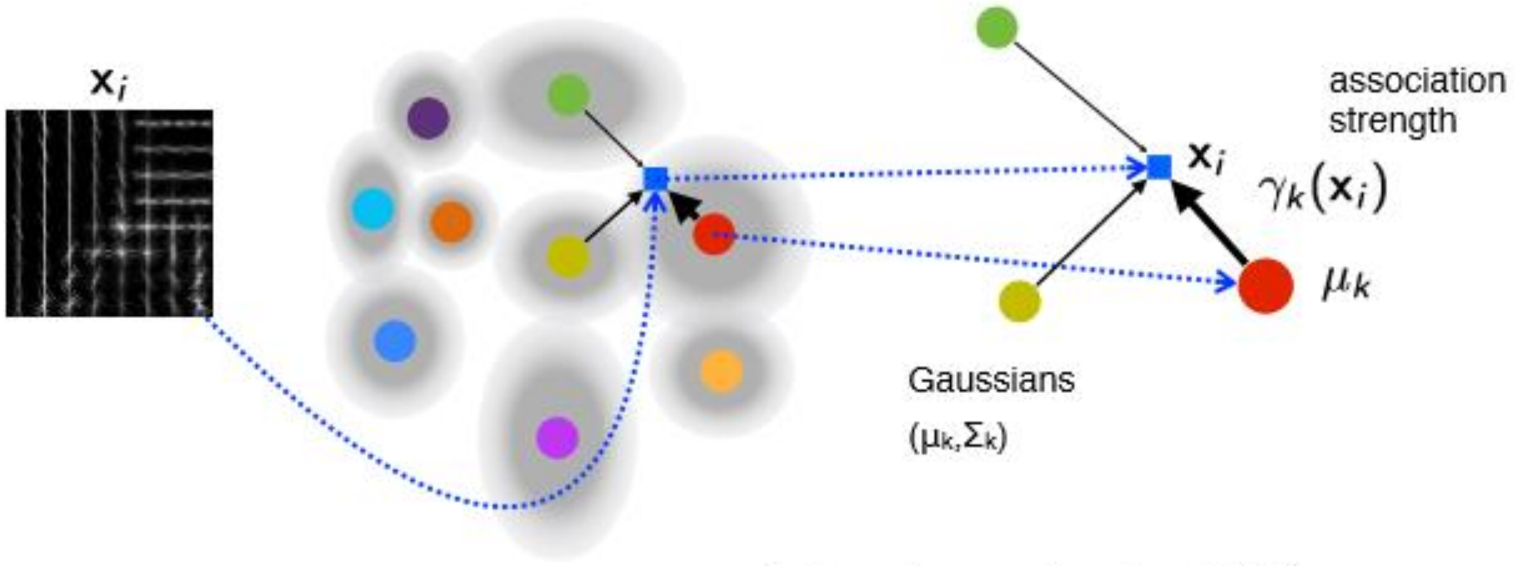
Parametric – Gaussian mixture

- 3 Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



Fisher Vectors



first and second order statistics

FV encoding $\Phi = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{u}_1 \\ \mathbf{v}_2 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{v}_K \\ \mathbf{u}_K \end{bmatrix}$

+ sqrt-l² normalisation

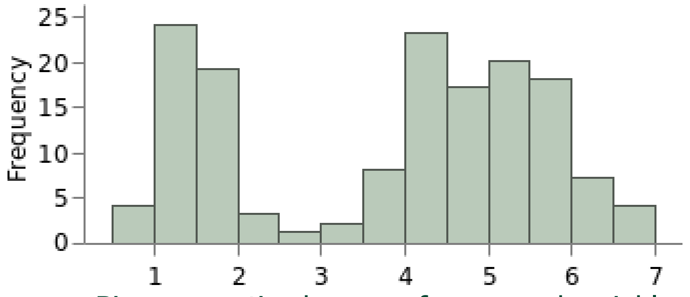
$$\mathbf{v}_k = \frac{1}{M\sqrt{\pi_k}} \sum_{i=1}^{M} \gamma_k(\mathbf{x}_i) \frac{\mathbf{x}_i - \mu_k}{\sigma_i}$$

$$\mathbf{u}_k = \frac{1}{M\sqrt{2\pi_k}} \sum_{i=1}^{M} \gamma_k(\mathbf{x}_i) \left(\frac{\mathbf{x}_i - \mu_k}{\sigma_i} - 1\right)^2$$

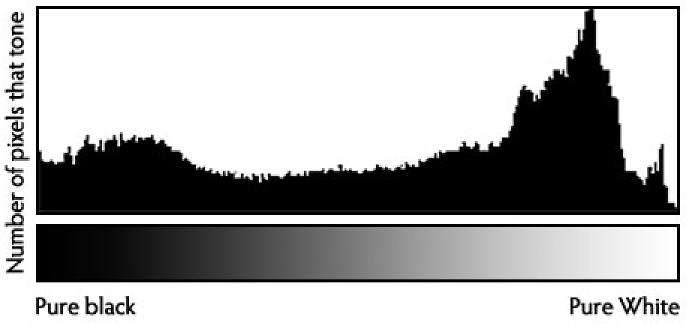
Nonparametric - histogram

- A histogram is a representation of the distribution of data.
  - an estimate of the probability distribution of a continuous variable
  - a rough sense of the density of the underlying distribution of the data
  - total area of a histogram used for probability density is always normalized to 1
    - m_i - bin value
    - n - number of observations
    - k - number of bins

$$n = \sum_{i=1}^{k} m_i \, .$$



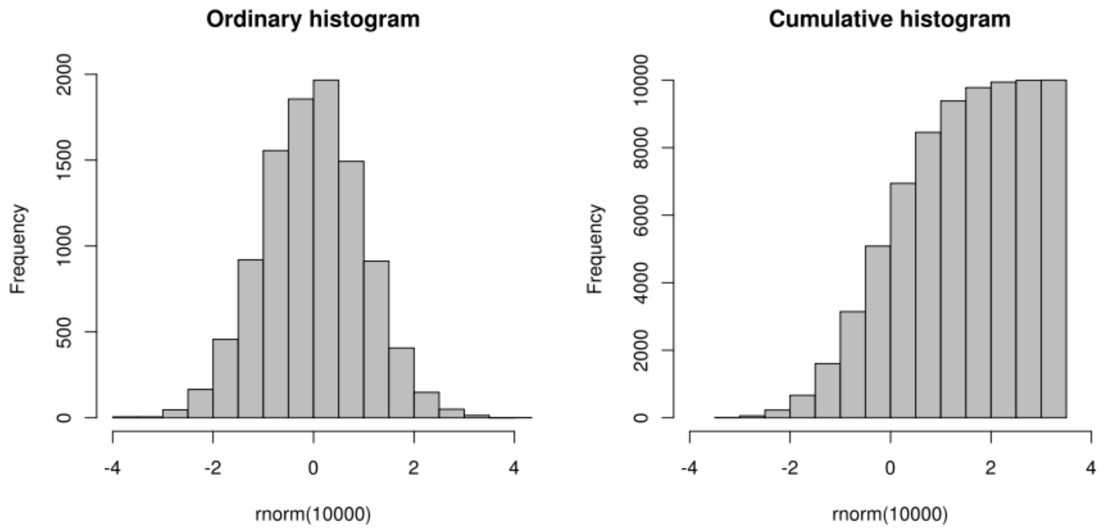
Bins – quantised range of measured variable



Nonparametric - histogram

- Cumulative histogram - counts the cumulative number of observations in all of the bins up to the specified bin

$$M_i = \sum_{j=1}^{i} m_j$$



Nonparametric - histogram

- Quantisation of data space
  - k – number of bins
  - x – data point
  - h – bin width

$$k = \left\lceil \frac{\max x - \min x}{h} \right\rceil$$

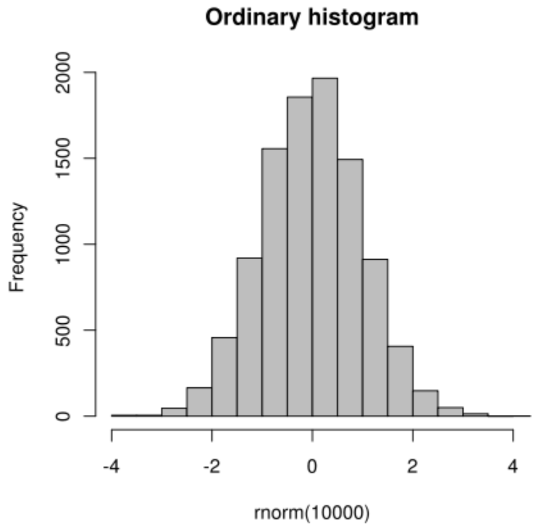
general

$$k = \sqrt{n}$$

assumes normal distribution

$$k = \lceil \log_2 n \rceil + 1$$

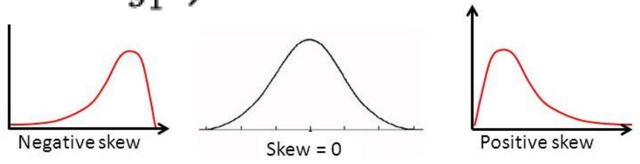
$$k = \lceil 2n^{1/3} \rceil$$



non-normal data + adds bins for skewness

$$k = 1 + \log_2(n) + \log_2\left(1 + \frac{|g_1|}{\sigma_{g_1}}\right)$$

Skew



Skew = g₁ = $\frac{\sum (x_i - \mu)^3}{n\sigma^3}$

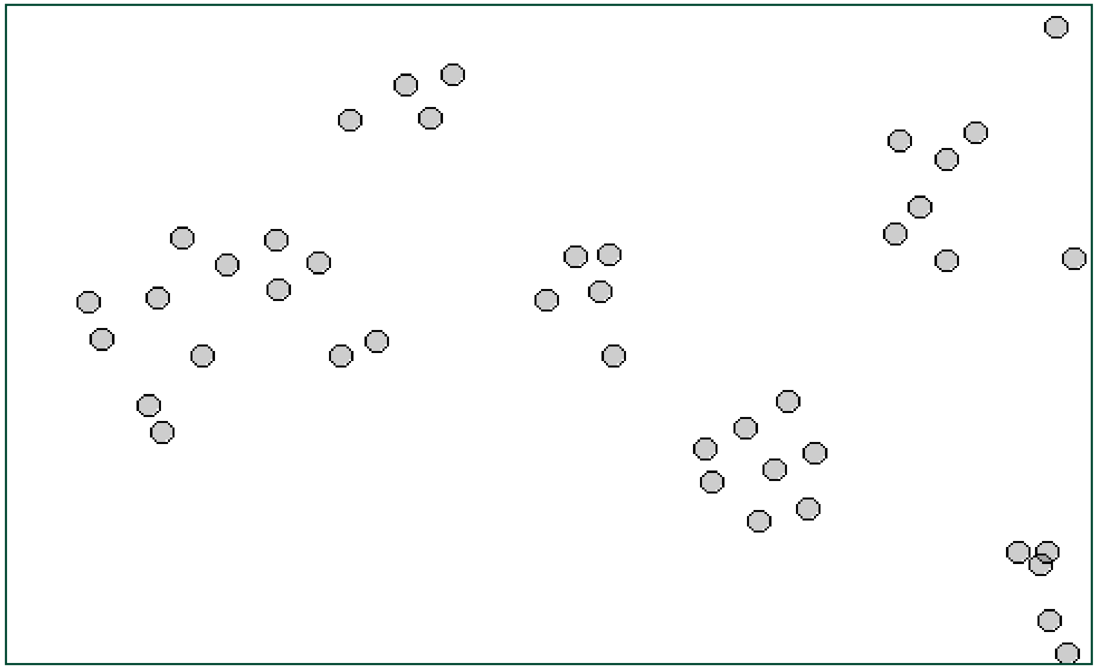
$$\sigma_{g_1} = \sqrt{\frac{6(n-2)}{(n+1)(n+3)}}$$

Clustering

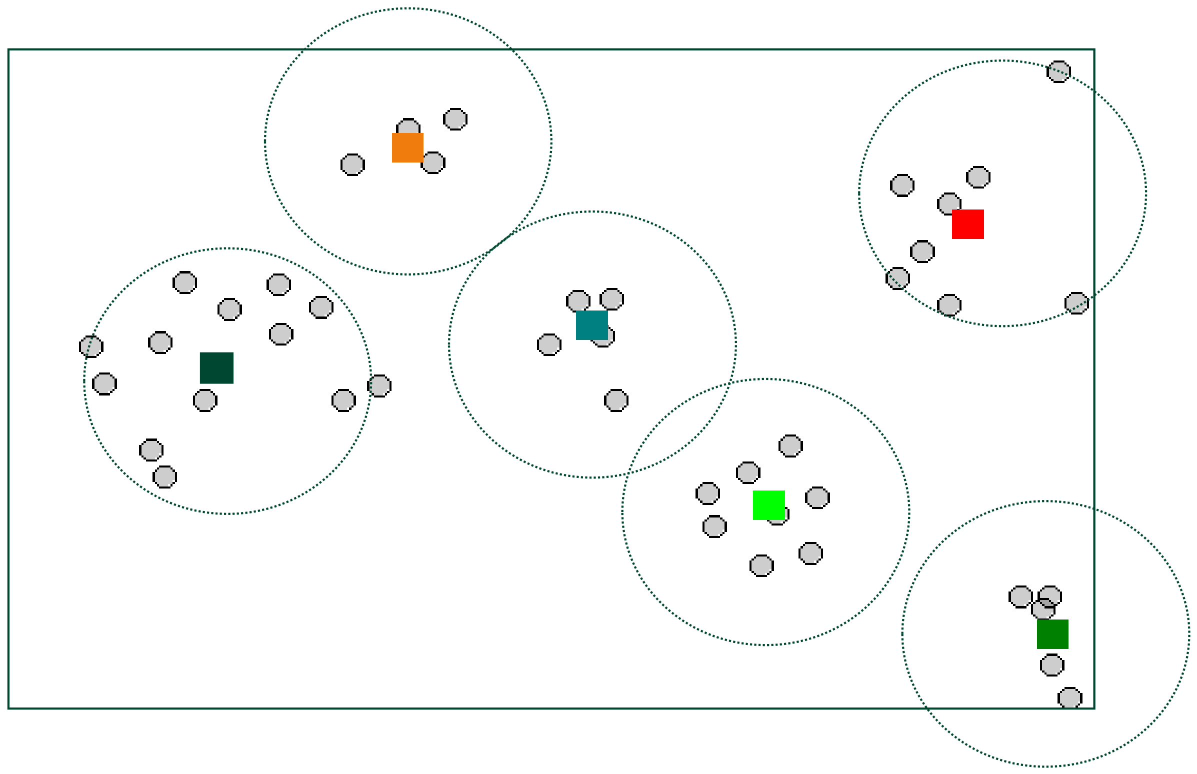
Clustering

- Classification of similar objects into different groups
- partitioning of a data set into subset (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure.

Clustering



Clustering



Clustering

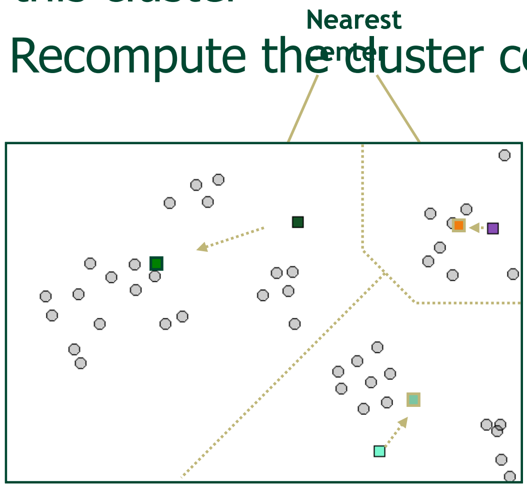
- kmeans - partitional
  - Top-down
- Agglomerative
  - Bottom-up

Top-down clustering

- Kmeans
  - Random initialization with arbitrarily set K - initial cluster centers

    For each data point find the nearest cluster center and assign the point to this cluster

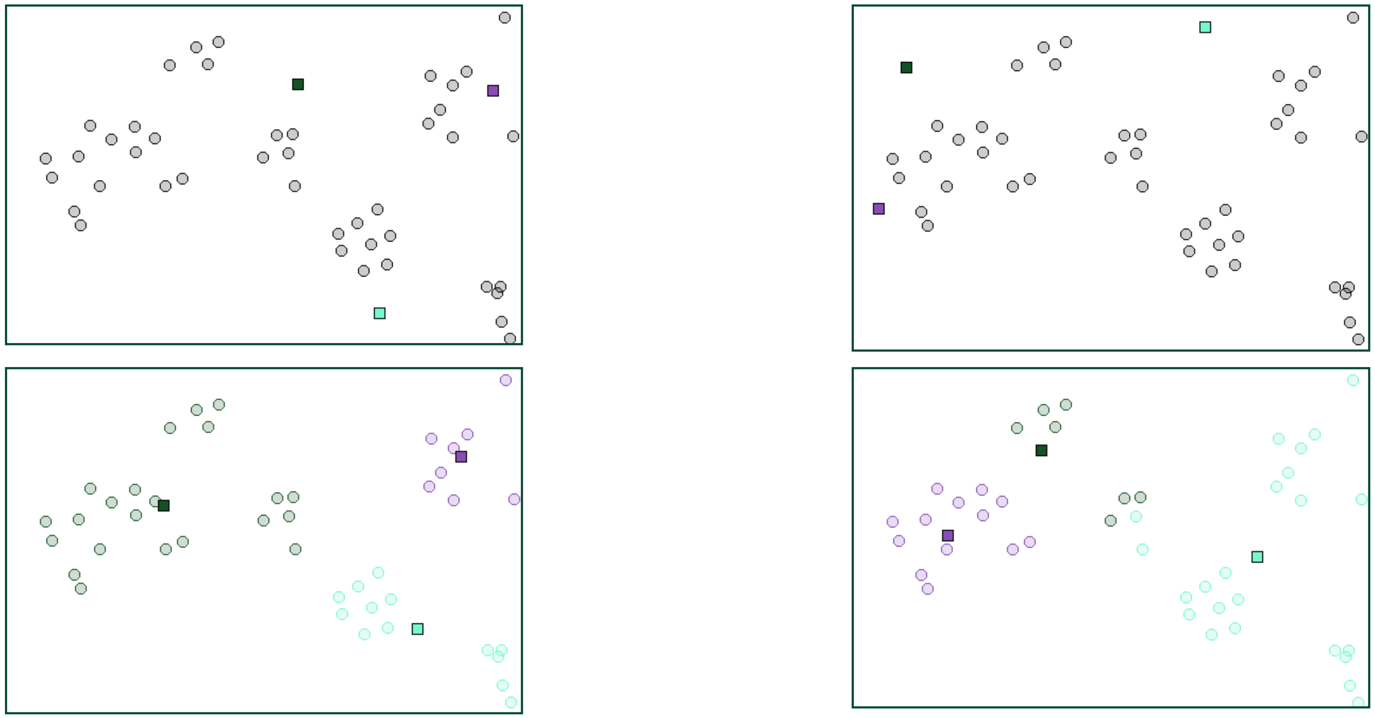
    Recompute the cluster center by computing the average of the



Top-down clustering

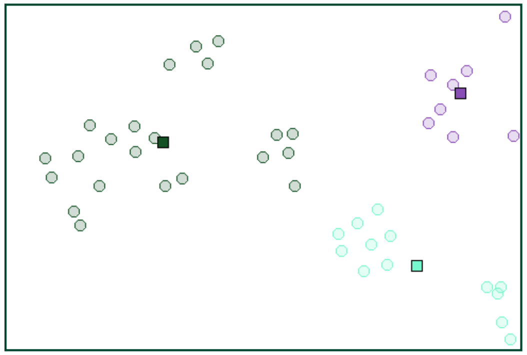
- Kmeans
  - Random initialization with arbitrarily set K

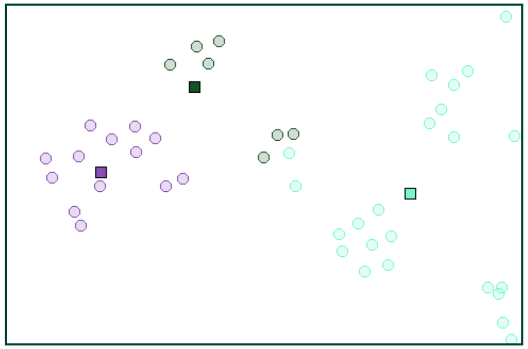
    The solution depends on initialization – local optimum



Top-down clustering

- Kmeans
  - Random initialization with arbitrarily set K
  - finds local optimum
  - sensitive to outliers
  - cluster centers might not represent well the point distribution
  - O(Nkld) - low complexity, efficient
    - Not if k is comparable with N
  - Good method for partitioning the data



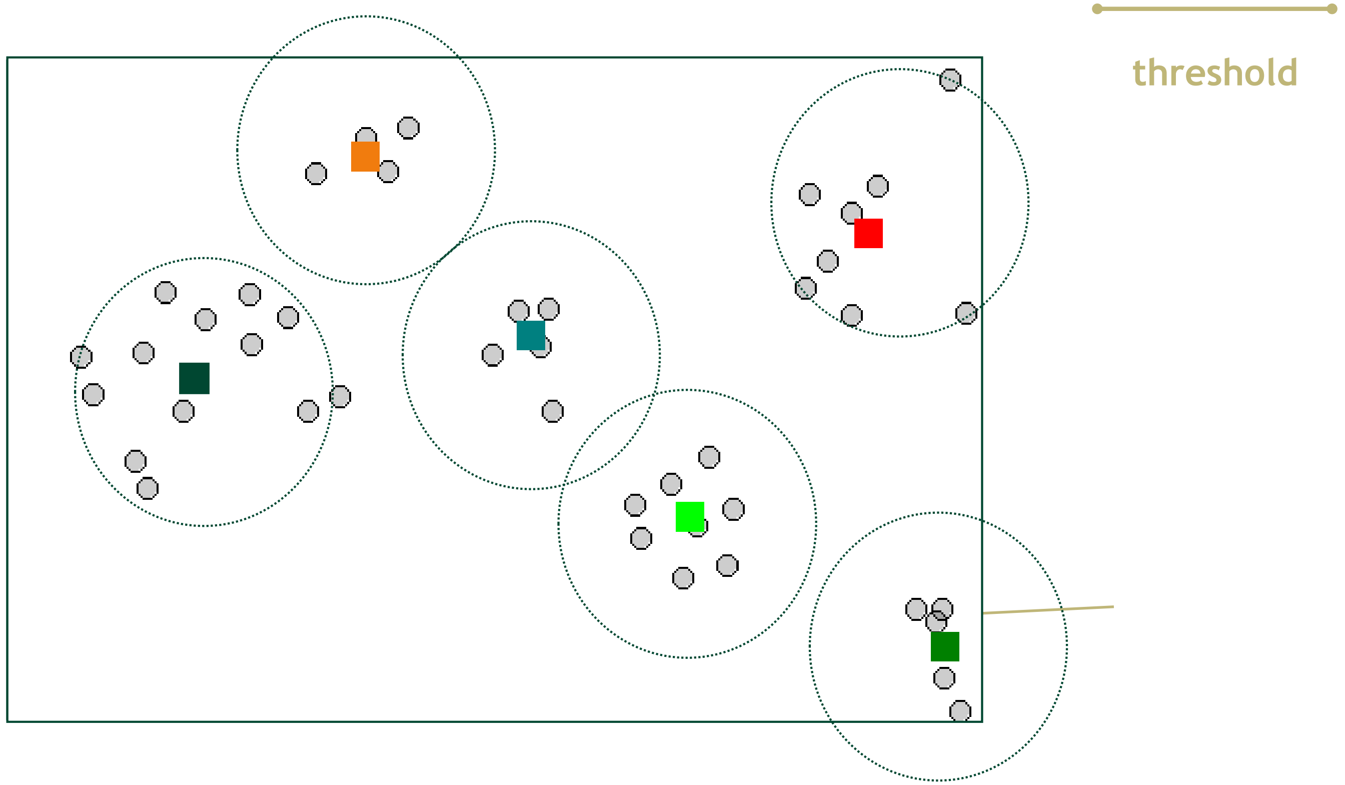


Imperial College London

Bottom-up agglomerative

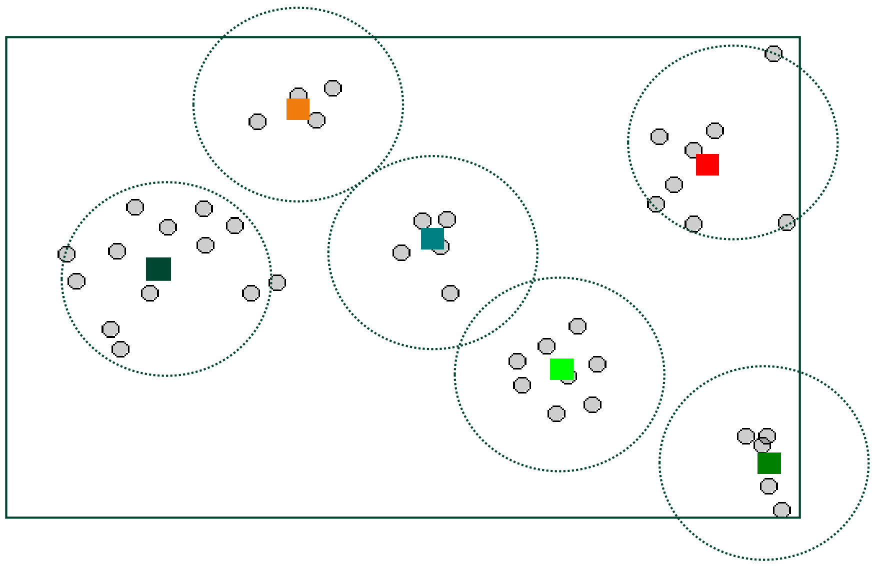
Set the threshold distance – maximum distance between two points

1. Compare all pairs and mark the with the smallest distance
2. Compare with the threshold – stop clustering if distance is larger than the threshold
3. Merge the pair and replace by its average
4. Repeat points 1 to 3



Bottom-up agglomerative

- Agglomerative,
  - Arbitrarily set maximum cluster size
  - Robust to outliers
  - Meaningful cluster centers



Distance/Similarity Metrics

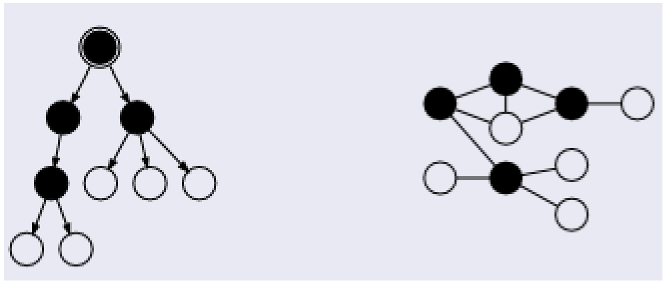
Intro

- Numerical data
  - Each data instance is a numerical feature vector.
    - Example: the age, body mass index, blood pressure, ... of a patient.

$$\mathbf{x} = \begin{pmatrix} 26 \\ 21.6 \\ 102 \\ \dots \end{pmatrix}$$

- Each instance is a structured object: a string, a tree or a graph.
  - Examples: words, DNA sequences, XML documents, molecules, social communities...

SFGAAJHSKJH



Intro

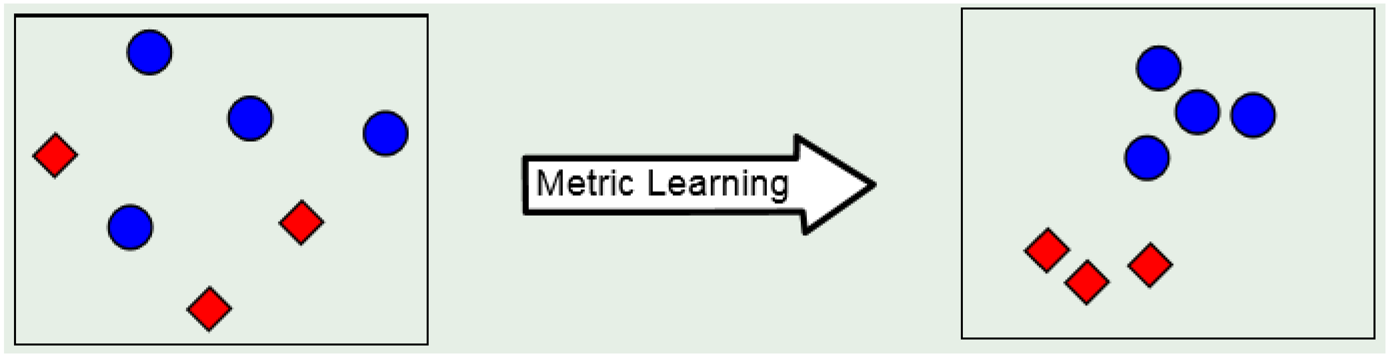
- Pairwise metric
  - Informally, a way of measuring the distance (or similarity) between object
- Metrics are ubiquitous in machine learning
  - Get yourself a good metric and you've basically solved the problem. Metrics are convenient proxies to manipulate complex objects.

Applications

- Classification
- k-Nearest Neighbors,
- Support Vector Machines...
- Clustering: K-Means and its variants.
- Information Retrieval / Ranking: search by query, image/document retrieval...
- Data visualization in high dimensions. ...
- Computer Vision: compare images or videos in ad-hoc representations. Used in image classification, object/face recognition, tracking, image annotation...
- Bioinformatics: compare structured objects such as DNA sequences or temporal series.
- Whenever the notion of metric plays an important role. Fun examples of applications include music recommendation, identity verification, cartoon synthesis and assessing the efficacy of acupuncture, to name a few

Metrics

- The notion of good metric is problem-dependent
  - Each problem has its own semantic notion of similarity, which is often badly captured by standard metrics (e.g., Euclidean distance).
- Solution: learn the metric from data
  - Basic idea: learn a metric that assigns small (resp. large) distance to pairs of examples that are semantically similar (resp. dissimilar).



Metric

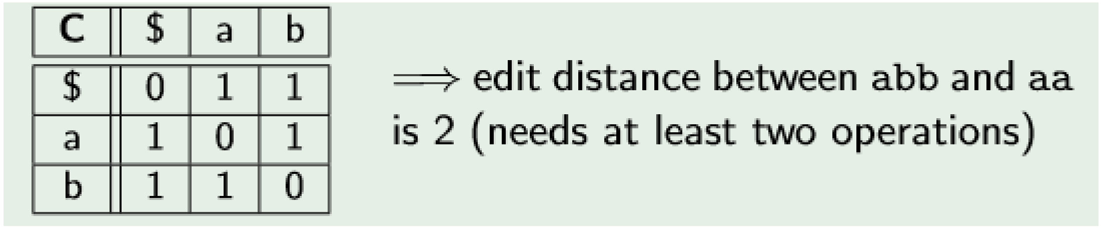
- Non-negativity:
  $D(P,Q) \geq 0$
- Identity of indiscernibles:
  $D(P,Q) = 0 \text{ iff } P = Q$
- Symmetry:
  $D(P,Q) = D(Q,P)$
- Subadditivity (triangle inequality):
  $D(P,Q) \leq D(P,K) + D(K,Q)$

Pseudo-Metric (aka Semi-Metric)

- Non-negativity:
  $D(P,Q) \geq 0$
- Property changed to:
  $D(P,Q) = 0 \; \text{if} \; P = Q$
- Symmetry:
  $D(P,Q) = D(Q,P)$
- Subadditivity (triangle inequality):
  $D(P,Q) \leq D(P,K) + D(K,Q)$

Similarity measures

- String edit distance
  - The edit distance is the cost of the cheapest sequence of operations (script) turning a string into another. Allowable operations are insertion, deletion and substitution of symbols. Costs are gathered in a matrix C.



C	\$	a	b
\$	0	1	1
a	1	0	1
b	1	1	0

$\Longrightarrow$ edit distance between abb and aa is 2 (needs at least two operations)

  - It is a proper distance if and only if C satisfies:

$$C_{ij} \geq 0, \quad C_{ij} = C_{ji}, \quad C_{ik} \leq C_{ij} + C_{jk} \quad \forall i,j,k$$

  - Generalization to trees (quadratic or cubic complexity) and graphs (NP-complete).

Minkowski-Form Distances

$$L_p(P,Q) = \left(\sum_i |P_i - Q_i|^p \right)^{\frac{1}{p}}$$

$$L_1(P,Q) = \sum_i |P_i - Q_i|$$

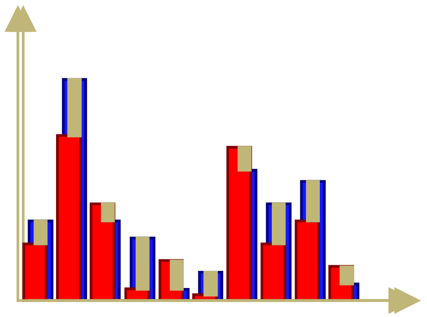
$$L_2(P,Q) = \sqrt{\sum_i (P_i - Q_i)^2}$$

$$L_\infty(P,Q) = \max_i |P_i - Q_i|$$

Similarity Measures

- Euclidean Distance

$$d(Q,V) = \sum_i (q_i - v_i)^2$$



- Properties
  - Focuses on the differences between the histograms
  - Range: [0,∞]
  - All cells are weighted equally.
  - Not very robust to outliers!

Similarity measures

- Cosine similarity
  - The cosine similarity measures the cosine of the angle between two instances, and can be computed a

$$K_{cos}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}.$$

  - It is widely used in data mining (better notion of similarity for bag-of-words + efficiently computable for sparse vectors).
- Bilinear similarity
  - The bilinear similarity is related to the cosine but does not include normalization and is parameterized by a matrix M:

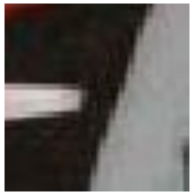
$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}',$$

  - where $M \in R^{d \times d}$ is not required to be PSD nor symmetric

Similarity Measures

- Cross correlation

$$f_1 = \begin{bmatrix} I_1(0,0) \\ I_1(0,1) \\ I_1(0,2) \\ \vdots \end{bmatrix}$$



$$f_2 = \begin{bmatrix} I_2(0,0) \\ I_2(0,1) \\ I_2(0,2) \\ \vdots \end{bmatrix}$$

$$D^2_{Euclidean}(f_1, f_2) = \| f_1 - f_2 \| = \sum_n (f_1(n) - f_2(n))^2 =$$

If the patches are normalized than $\sum_n f_1^2(n) = \sum_n f_2^2(n) = constant$

$$CrossCorrelation(f_1, f_2) = \sum_n f_1(n) f_2(n)$$

- Properties

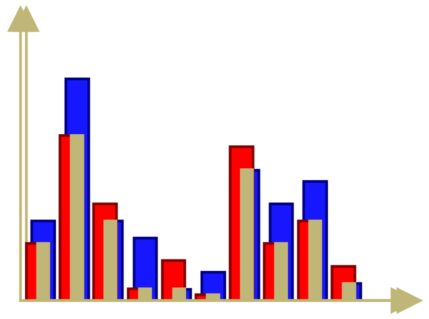
-very disctinctive,
-easy to implement

-very high dimensional,
-not very robust to noise

Similarity Measures

- Intersection

$$\cap(Q,V) = \sum_i \min(q_i, v_i)$$

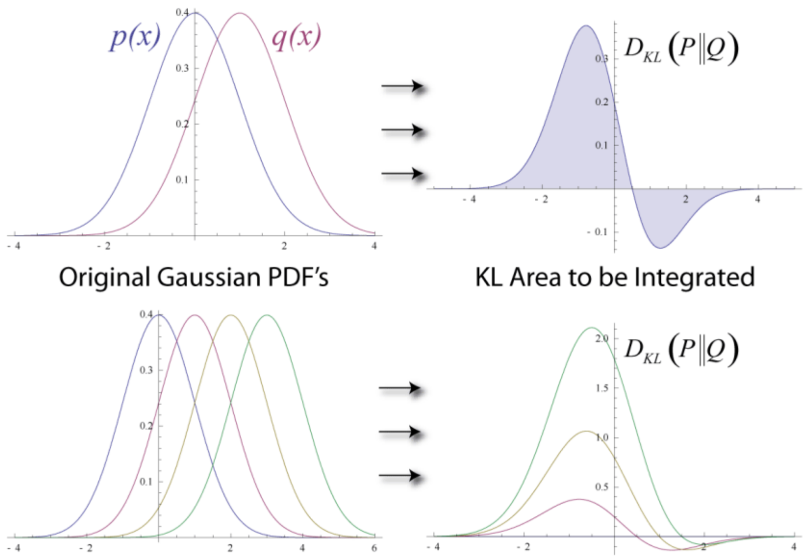


- Properties
  - Measures the common part of both histograms
  - Range: [0,1]
  - For unnormalized histograms, use the following formula

$$\cap(Q,V) = \frac{1}{2}\left(\frac{\sum_i \min(q_i, v_i)}{\sum_i q_i} + \frac{\sum_i \min(q_i, v_i)}{\sum_i v_i}\right)$$

Kullback-Leibler Divergence

$$KL(P,Q) = \sum_i P_i \log \frac{P_i}{Q_i}$$



- Properties
  - Difference between two probability distributions
  - Information theoretic origin
  - Non symmetric
  - Not a metric
  - $Q_i = 0$?

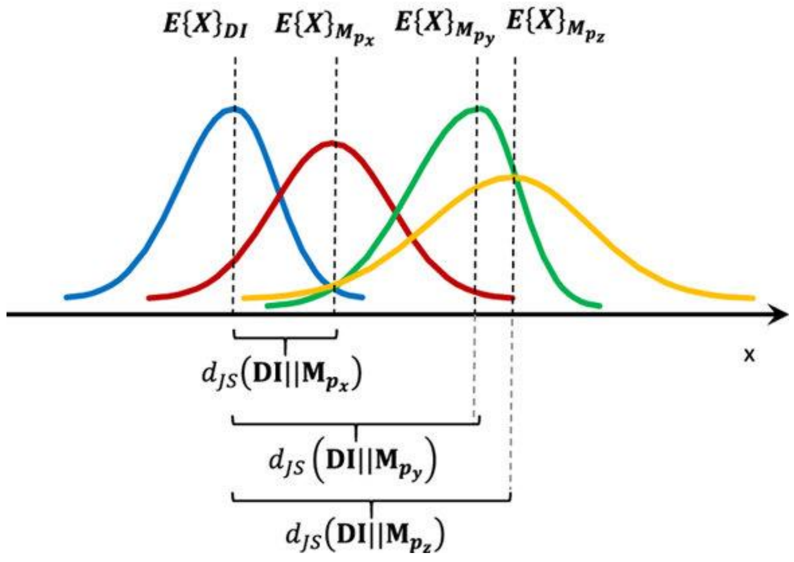
Jensen-Shannon Divergence

$$JS(P,Q) = \frac{1}{2}KL(P,M) + \frac{1}{2}KL(Q,M)$$

$$M = \frac{1}{2}(P+Q)$$

$$JS(P,Q) = \frac{1}{2}\sum_i P_i \log \frac{2P_i}{P_i+Q_i} + \frac{1}{2}\sum_i Q_i \log \frac{2Q_i}{P_i+Q_i}$$

- Properties
  - Information theoretic origin
  - Symmetric.
  - $\sqrt{JS}$ is a metric.


$E\{X\}_{DI}$
$E\{X\}_{M_{p_x}}$
$E\{X\}_{M_{p_y}}$
$E\{X\}_{M_{p_z}}$
x
$d_{JS}(\mathbf{DI}||\mathbf{M}_{p_x})$
$d_{JS}(\mathbf{DI}||\mathbf{M}_{p_y})$
$d_{JS}(\mathbf{DI}||\mathbf{M}_{p_z})$

Jensen-Shannon Divergence

- Using Taylor extension and some algebra:

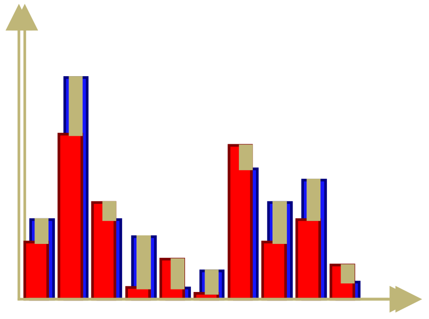
$$JS(P,Q) = \sum_{n=1}^{\infty} \frac{1}{2n(2n-1)} \sum_i \frac{(P_i - Q_i)^{2n}}{(P_i + Q_i)^{2n-1}}$$
$$= \frac{1}{2} \sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)} +$$
$$\frac{1}{12} \sum_i \frac{(P_i - Q_i)^4}{(P_i + Q_i)^3} + \dots$$

$$\chi^2(P,Q) = \frac{1}{2} \sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}$$

Similarity Measures

- Chi-square

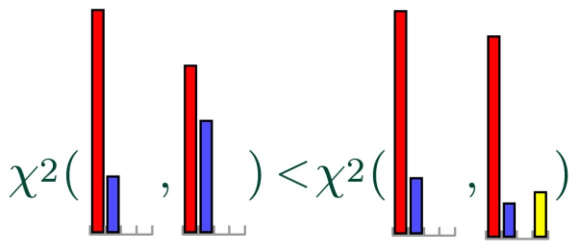
$$\chi^2(P,Q) = \frac{1}{2}\sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}$$

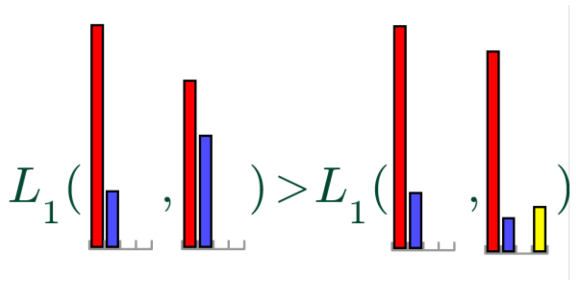


- Properties
  - Statistical background:
    - Test if two distributions are different
    - Experimentally results are very similar to JS.
  - Range: $[0,\infty]$
  - Cells are not weighted equally!
  - $\sqrt{\chi^2}$ is a metric, experimentally better than L_2
  - More robust to outliers, reduces the effect of large bins

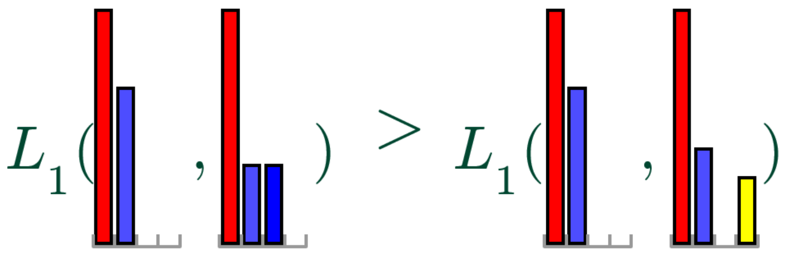
χ^2 Histogram Distance

- Different metrics may lead to opposite conclusion



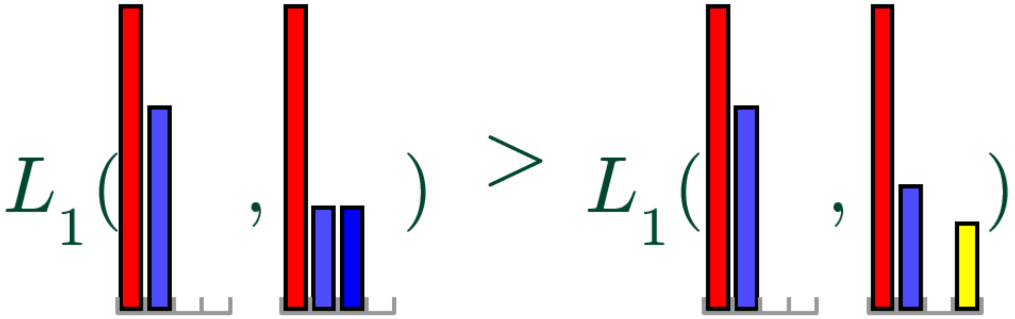


- Bin-to-Bin distances such as L2 are sensitive to quantization:



Bin-to-Bin Distances

- #bins ↓ ⇒ robustness ↑ distinctiveness ↓
- #bins ↑ ⇒ robustness ↓ distinctiveness ↑



Can we achieve robustness

and

distinctiveness ?

Quadratic-Form Histogram Distance

$$QF^A(P,Q) = \sqrt{(P-Q)^T A(P-Q)}$$

$$= \sqrt{\sum_{ij} (P_i - Q_i)(P_j - Q_j) A_{ij}}$$

- Properties
  - A_{ij} is the similarity between bin *i* and *j*.
  - If *A* is the inverse of the covariance matrix, QF is called Mahalanobis distance.
  - If A is the identity matrix, QF is the Euclidean distance

$$A = I \searrow \quad = \sqrt{\sum_{ij} (P_i - Q_i)^2} = \sqrt{L_2(P,Q)}$$

Quadratic-Form Histogram Distance

$$QF^{A}(P,Q) = \sqrt{(P-Q)^T A (P-Q)}$$

- Properties
  - Does not reduce the effect of large bins.
  - Alleviates the quantization problem.
  - Linear time computation in # of non zero A_{ij}
  - If A is positive-definite then QF is a metric
  - If A is positive-semidefinite then QF is a pseudo-metric

Quadratic-Form Histogram Distance

$$QF^A(P,Q) = \sqrt{(P-Q)^T A (P-Q)}$$

$$= \sqrt{(P-Q)^T W^T W (P-Q)}$$

$$= L_2(WP, WQ)$$

- Properties
  - We assume there is a **linear transformation** that makes bins **independent**
    - There are cases where this is **not true** *e.g.* **COLOR**
  - Converting distance to similarity (Hafner *et. al* 95):

$$A_{ij} = 1 - \frac{D_{ij}}{\max_{ij}(D_{ij})}$$

or

$$A_{ij} = e^{-\alpha \frac{D(i,j)}{\max_{ij}(D(i,j))}}$$

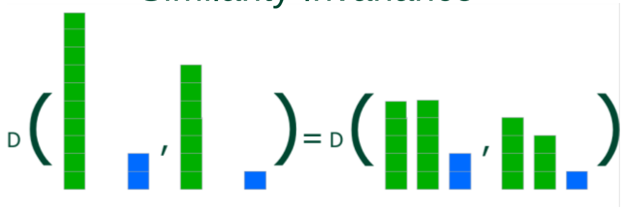
If α is large enough,
A will be positive-definitive

Quadratic-Chi Histogram Distance

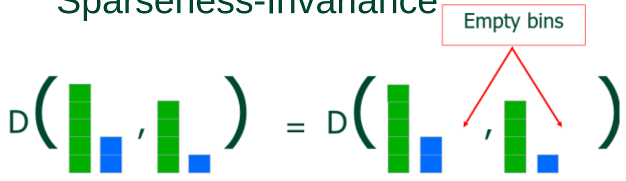
$$\mathrm{QC}_m^A(P,Q) = \sqrt{\sum_{ij} \frac{(P_i - Q_i)(P_j - Q_j)A_{ij}}{(\sum_c (P_c + Q_c)A_{ci})^m (\sum_c (P_c + Q_c)A_{cj})^m}}$$

- Properties
  - A_{ij} is the similarity between bin i and j.
  - Generalizes *QF* and χ^2
  - Reduces the effect of large bins.
  - Alleviates the quantization problem.
  - Linear time computation in # non zero A_{ij}.
  - non-negative if A is positive-semidefinite.
  - Symmetric
  - Triangle inequality unknown.
  - If we define $\frac{0}{0} = 0$ and $0 \le m < 1$, QC is continuous.
  -

Similarity-Invariance



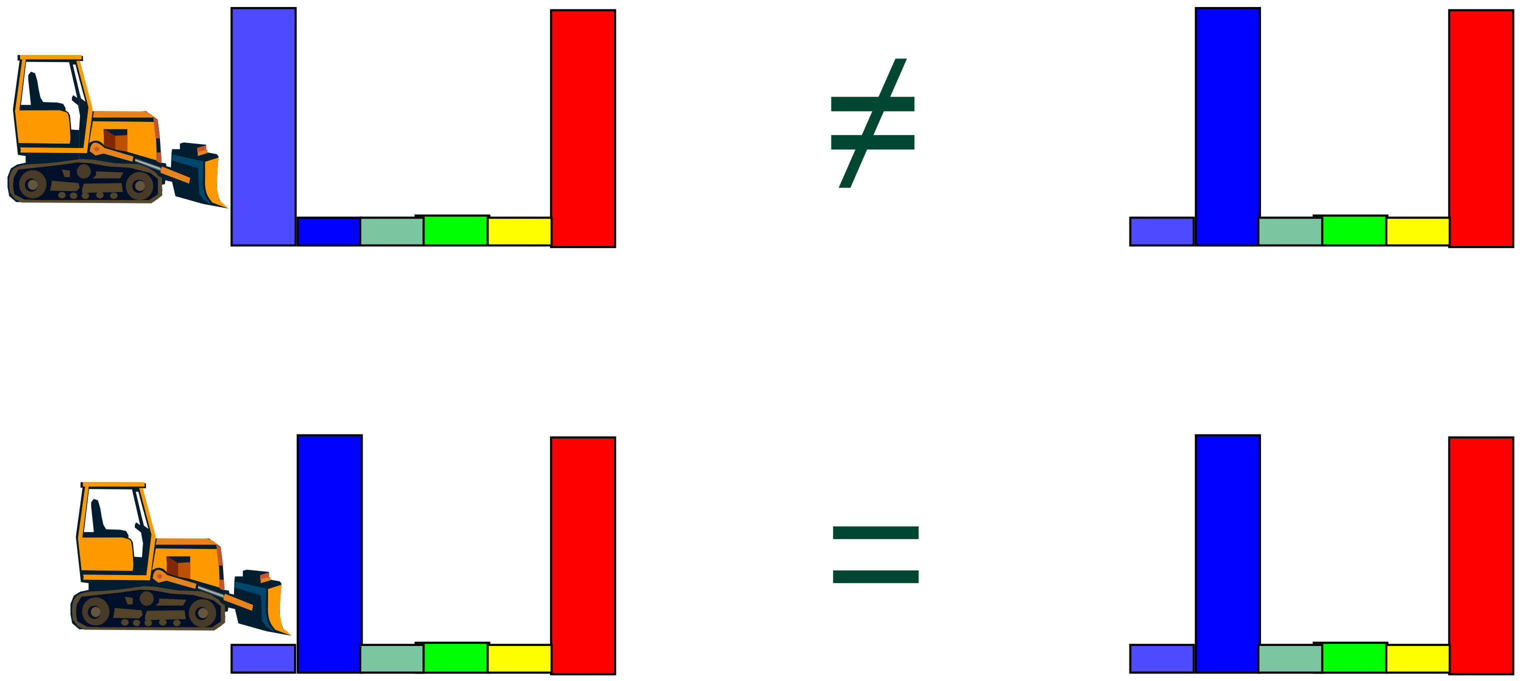
Sparseness-Invariance



The Earth Mover’s Distance

The Earth Mover’s Distance

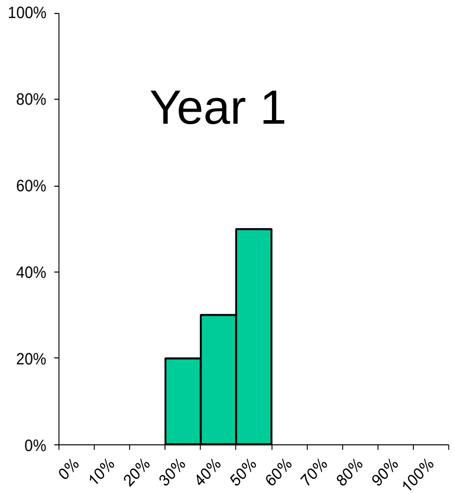
- The Earth Mover’s Distance is defined as the minimal cost that must be paid to transform one histogram into the other, where there is a “ground distance” between the basic features that are aggregated into the histogram.

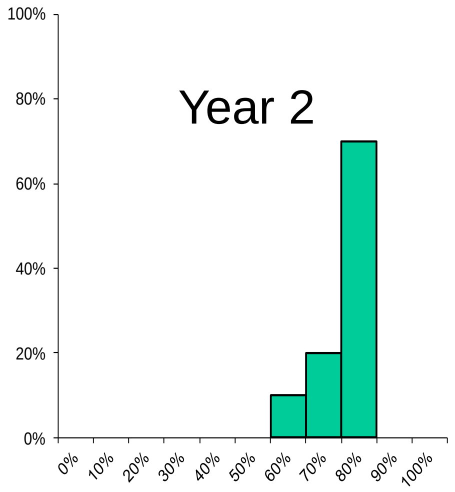


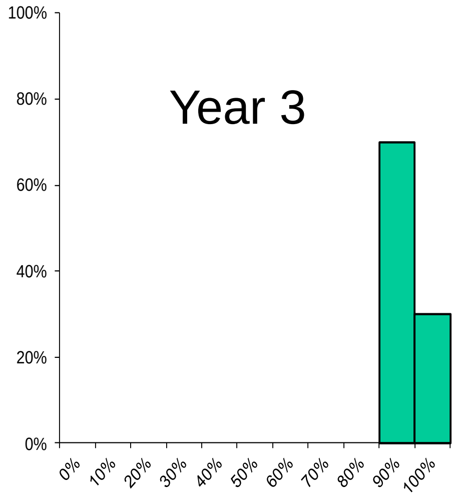
Earth Mover's Distance - example

[Rubner *et al.* 1998]

- We are given the distribution of grades for a course over the past three years and we want to compare the distributions:







- If we just compare theses as vectors, the results from Year 3 are as similar to the results from Year 2 as they they are to the results of Year1.

Earth Mover's Distance - approach

- Instead of comparing the values in each bin (e.g. Euclidean distance), compute the amount of work needed to transform one distribution into the other.
- Define the cost of moving *d* values from bin to *i* to bin *j* as:

$$\mathrm{Work}(d, i, j) = d \cdot |i - j|$$

- Find the minimal amount of work that needs to be done to transform one distribution into the other.

Earth Mover's Distance - definition

- Given distributions $X=\{x_1,\ldots,x_n\}$ and $Y=\{y_1,\ldots,y_n\}$, set $c_{ij}=|i-j|$ and find the values for f_{ij} that minimize:

$$\text{EMD}_{\text{work}} = \min_{F=\{F_{ij}\}} \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}c_{ij}$$

  - subject to the constraints:

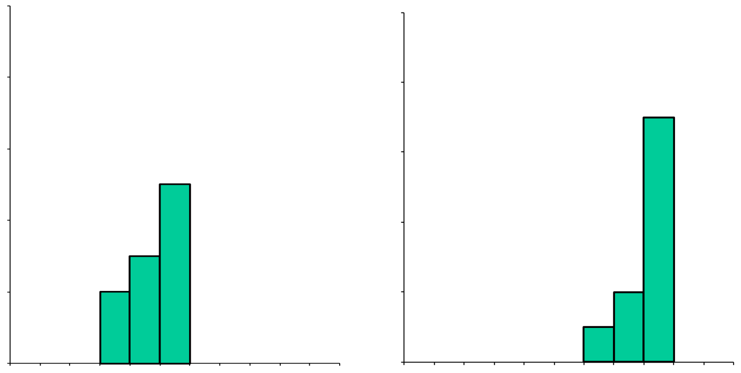
$c_{ij} \geq 0$ cost of moving from bin i to bin j

$f_{ij} \geq 0$ amount of data moved from bin i to bin j

$$\sum_{j=1}^{n} f_{ij} = x_i \quad \text{and} \quad \sum_{i=1}^{n} f_{ij} = y_j$$

Earth Mover’s Distance - illustration

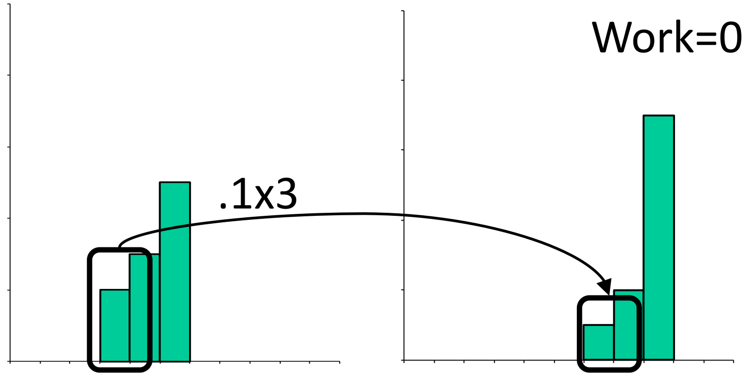
- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

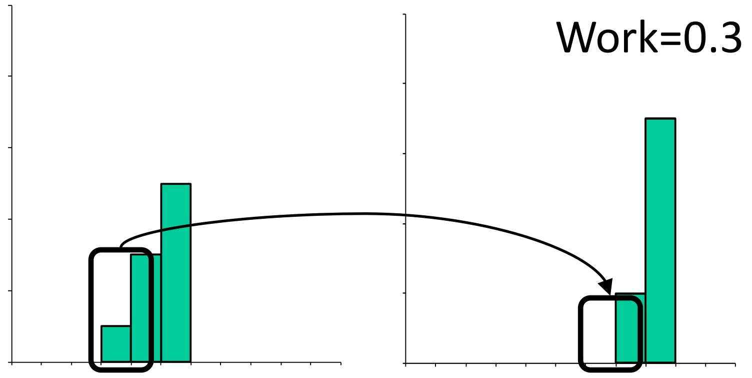
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover’s Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

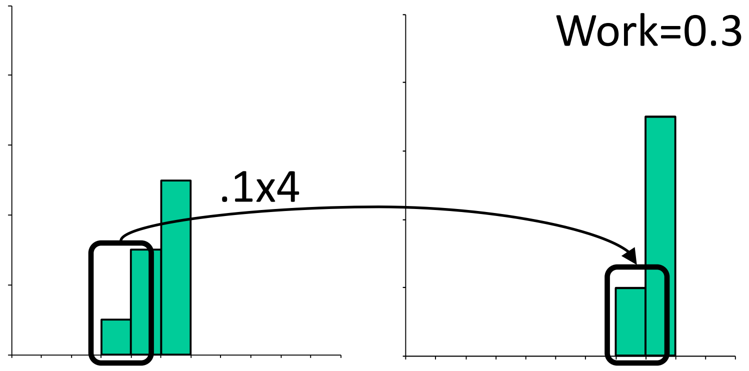
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover’s Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

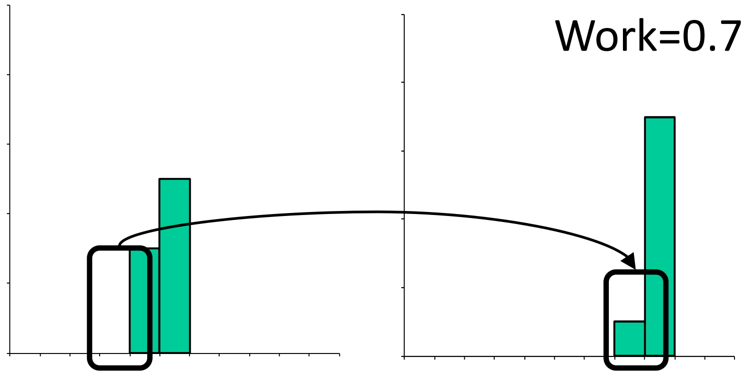
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

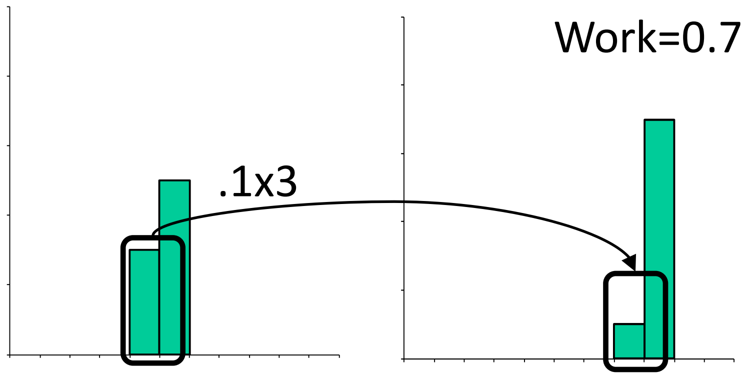
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

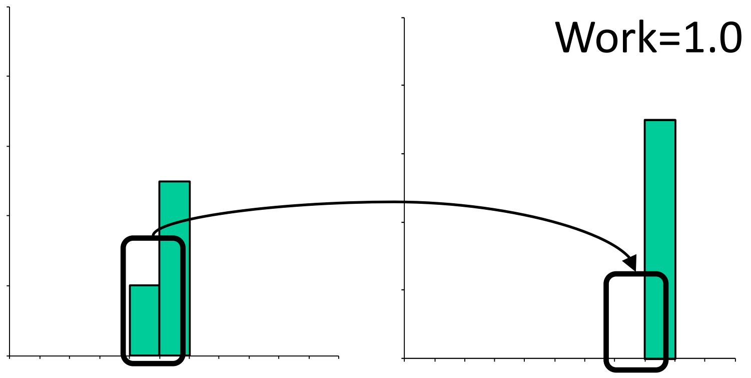
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

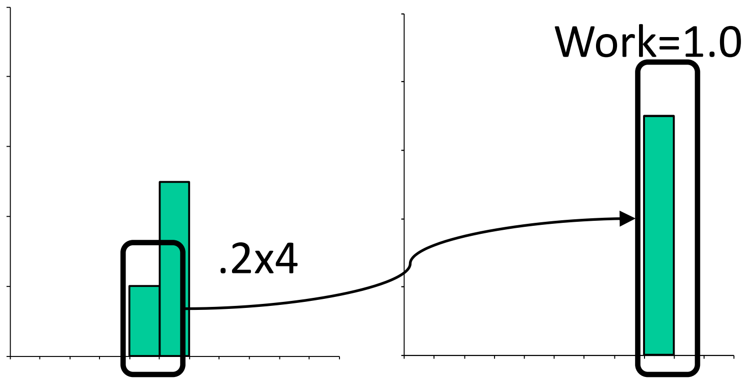
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover’s Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

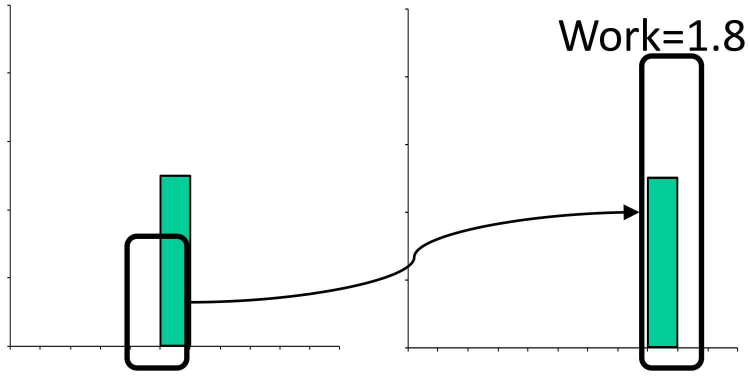
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

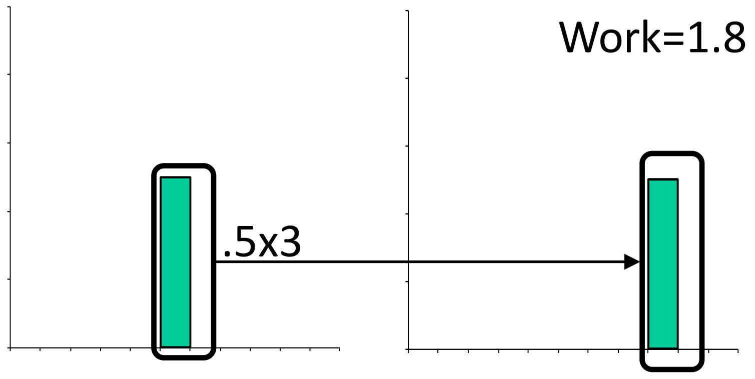
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

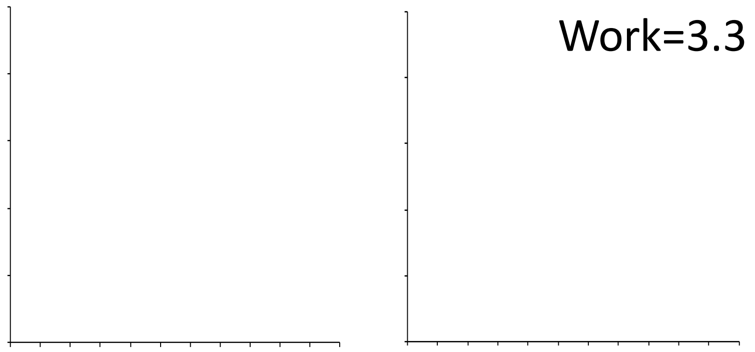
- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover's Distance - illustration

- In general, this is the *transportation problem* and can be solved using linear programming.
- For 1D histograms, this can be solved using the greedy algorithm.

- Find the first non-empty bin.
- Move earth into first non-empty bin in the other histogram.



Earth Mover’s Distance

Alternatively:

Compute the cumulative distributions:

$$CDX(i) = \sum_{j=1}^{i} x_j$$

Then the Earth Mover’s Distance between *X* and *Y* is:

$$EMD(X,Y) = \sum_{i=1}^{n} \left| CDX(i) - CDY(i) \right|$$

So that for 1D histograms, the EMD can be expressed as a normed difference.

Distance/Similarity measures

- Euclidean L_2
- Manhattan L_1
- Chesboard L_{inf}
- Euclidean L_2
- Cosine
- Correlation
- Intersection
- Kullback-Leibler divergence
- Jenses-Shanon divergence
- Chi square
- Quadratic form distance
- Earth movers distance
- **Mahalanobis distance**

Distance Metric Learning

Distance Metric Learning - motivation

Learn metric for scaling dimensions/features

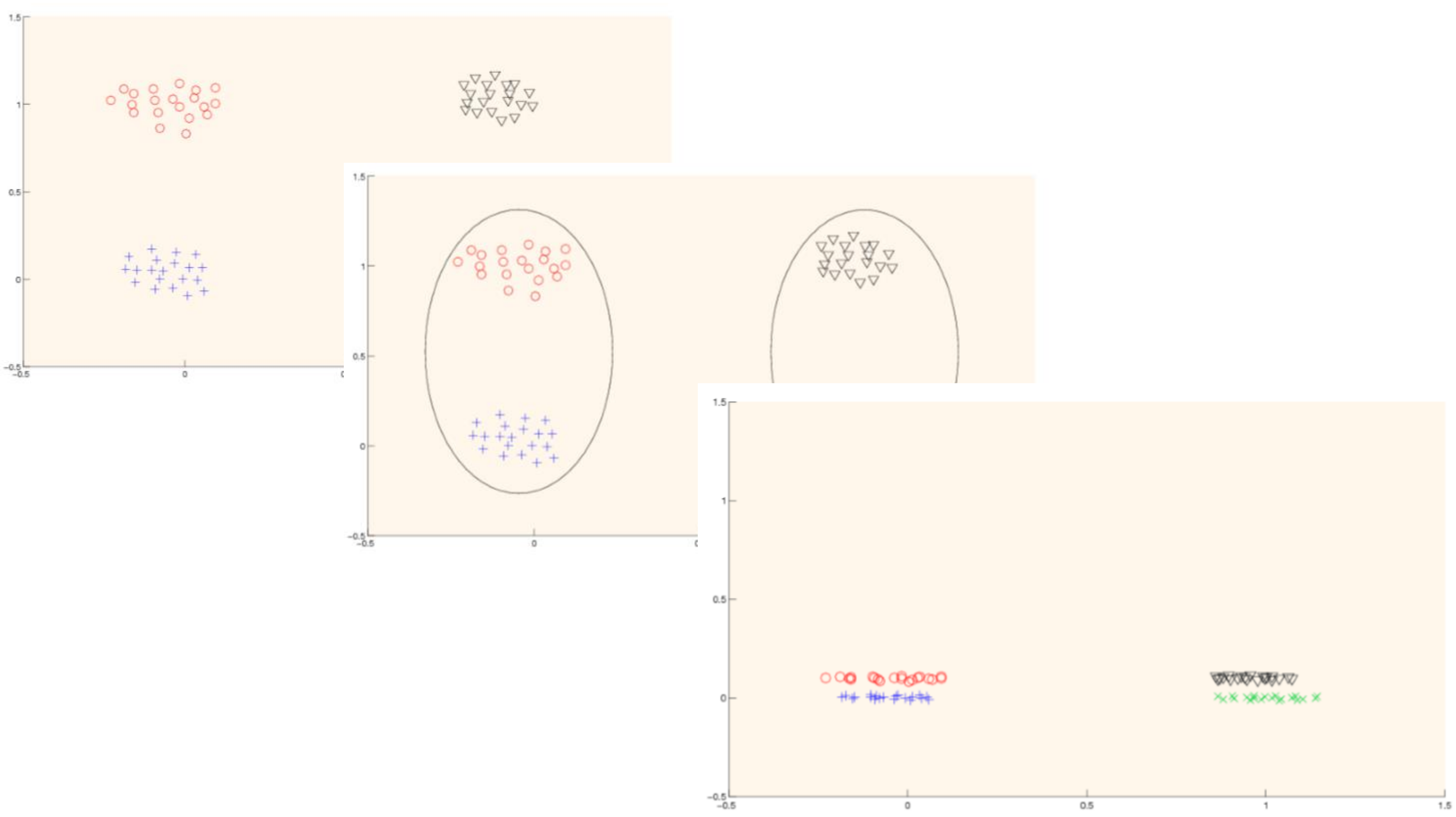
- Example: UCI Wine data set
  - 13 features 9/13 features have mean value in [0,10]
  - 3/13 features have mean value in [10,100]
  - One feature has a mean value of 747 (with std 315)

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

- Using a standard distance such as Euclidean distance, the largest feature dominates the computation
  - That feature may not be important for classification
- Need a weighting of the features that improves classification or other tasks

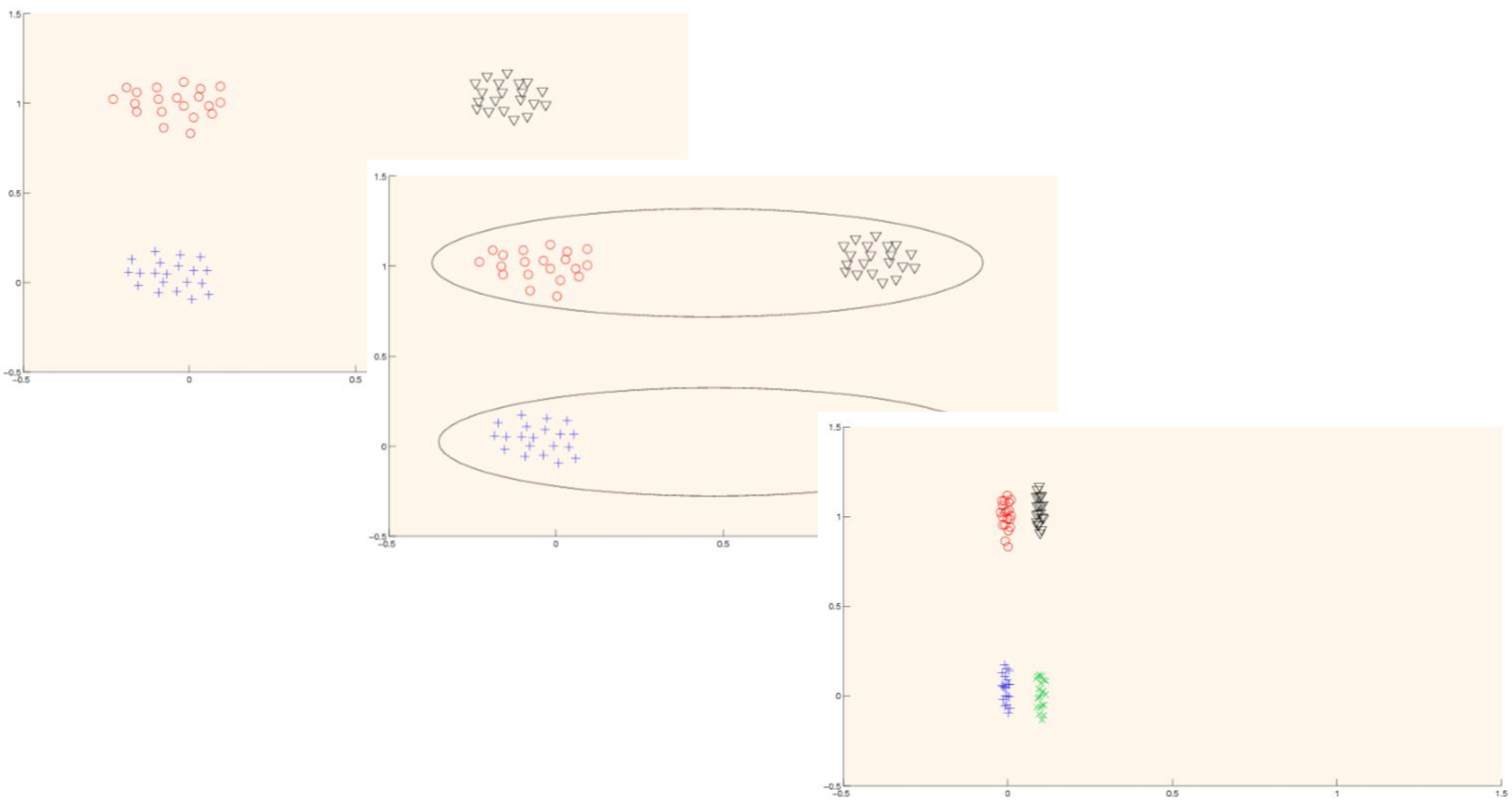
Distance Metric Learning - motivation

- Learn metric for reducing/grouping dimensions/features



Distance Metric Learning - motivation

- Learn metric for reducing/grouping dimensions/features

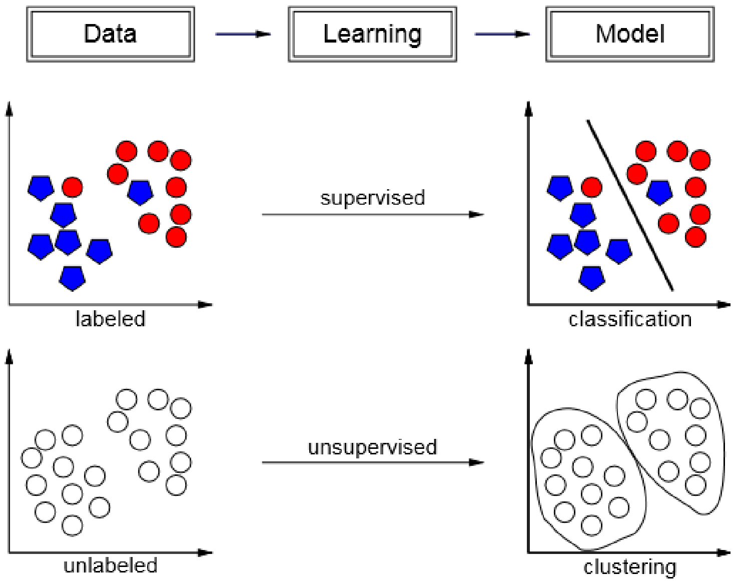


Distance Metric Learning

- Feature re-weighting/scaling
  - Learn weightings (scaling) over the features, then use standard distance (e.g., Euclidean) after re-weighting
  - Diagonal Mahalanobis methods
  - Number of parameters grows linearly with the dimensionality d
- Full linear transformation
  - In addition to scaling of features, also rotates the data
  - Linear dimensionality reduction for transformations to r < d dimensions
  - For transformations from d dimensions to d dimensions, number of parameters grows quadratically in d
- Non-linear transformation
  - Variety of methods
  - Kernelization of linear transformations
  - Neural nets
  - Complexity varies from method to method

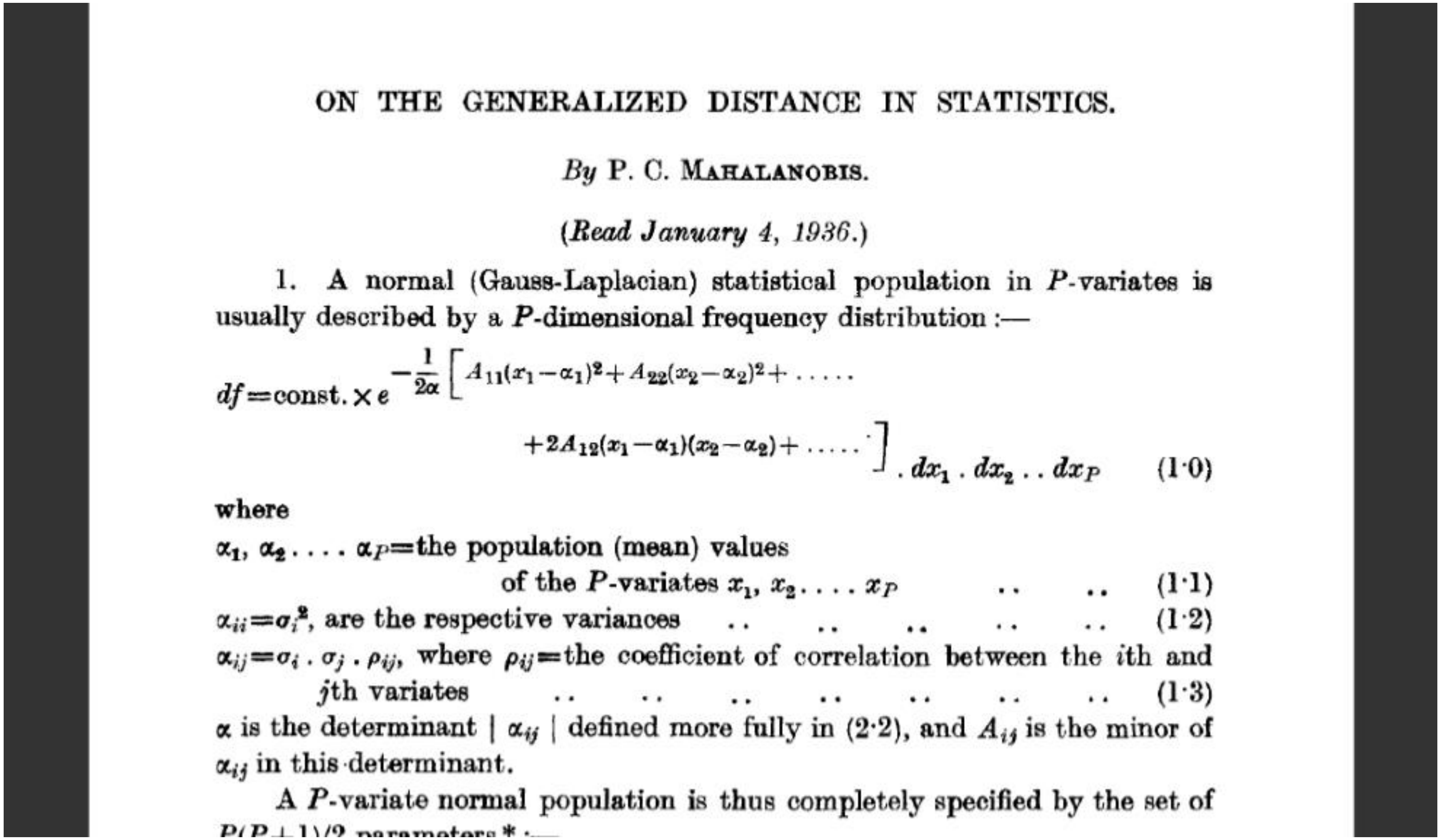
Distance Metric Learning

- Unsupervised Metric Learning
  - Dimensionality reduction
  - Principal Components Analysis
  - Kernel PCA
  - Multidimensional Scaling



- Supervised and Semi-supervised Metric Learning
  - Constraints or labels given to the algorithm
  - Example: set of similarity and dissimilarity constraints

Mahalanobis distance – original covariance based



ON THE GENERALIZED DISTANCE IN STATISTICS.

By P. C. MAHALANOBIS.

(*Read January 4, 1936.*)

1. A normal (Gauss-Laplacian) statistical population in P-variates is usually described by a P-dimensional frequency distribution :—

$$df = \text{const.} \times e^{-\frac{1}{2\alpha}\left[A_{11}(x_1-\alpha_1)^2 + A_{22}(x_2-\alpha_2)^2 + \ldots\ldots + 2A_{12}(x_1-\alpha_1)(x_2-\alpha_2) + \ldots\ldots\right]} . dx_1 . dx_2 .. dx_P \qquad (1{\cdot}0)$$

where

$\alpha_1, \alpha_2 \ldots . \alpha_P$ = the population (mean) values of the P-variates $x_1, x_2 \ldots . x_P$ .. .. (1·1)

$\alpha_{ii} = \sigma_i^2$, are the respective variances .. .. (1·2)

$\alpha_{ij} = \sigma_i . \sigma_j . \rho_{ij}$, where ρ_{ij} = the coefficient of correlation between the ith and jth variates .. .. (1·3)

α is the determinant $|\alpha_{ij}|$ defined more fully in (2·2), and A_{ij} is the minor of α_{ij} in this determinant.

A P-variate normal population is thus completely specified by the set of $P(P+1)/2$ parameters * :—

Mahalanobis distance – original covariance based

Assume the data is represented as N vectors of length d:
$X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$

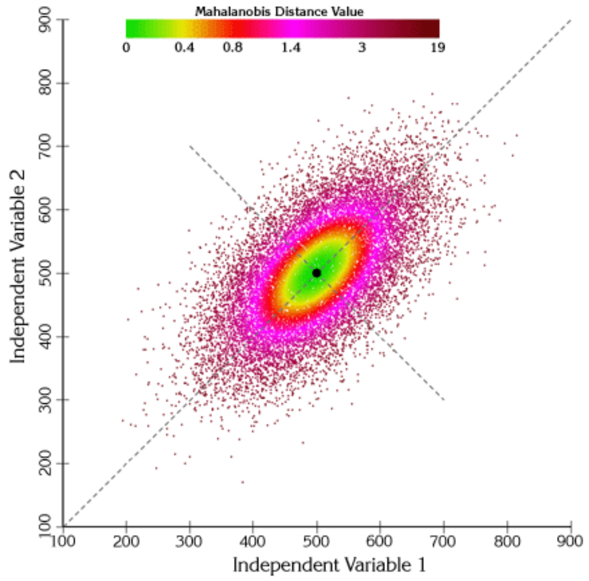
Squared Euclidean distance

$$\begin{aligned} d(\mathbf{x}_1, \mathbf{x}_2) &= \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \\ &= (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) \end{aligned}$$

Let $\Sigma = \sum_{i,j}(\mathbf{x}_i - \mu)(\mathbf{x}_j - \mu)^T$

The "original" Mahalanobis distance:

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1}(\mathbf{x}_1 - \mathbf{x}_2)$$



Mahalanobis distance

- Weighting/scaling of dimensions/features
  - Diagonal covariance matrix

$f_1 = [v_1, \ldots, v_N]$

$$D^2_{Euclidean}(f_1, f_2) = \| f_1 - f_2 \| = \sum_n (f_1(n) - f_2(n))^2$$

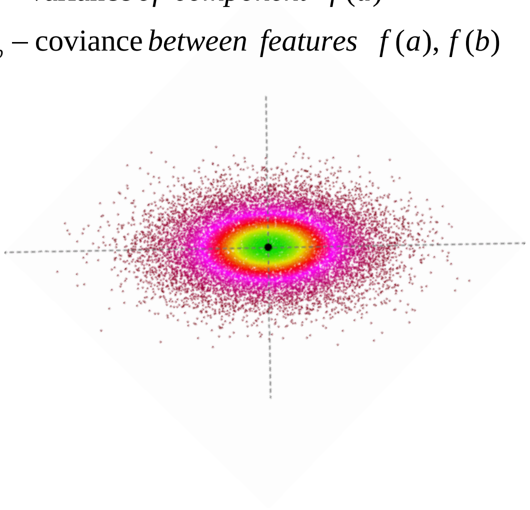
$f_2 = [v_1, \ldots, v_N]$

$$D^2_{Mahalanobis}(f_1, f_2) = [f_1 - f_2]^T \cdot \Sigma^{-1} \cdot [f_1 - f_2] =$$

$$= \begin{bmatrix} f_1(0) - f_2(0) \\ f_1(1) - f_2(1) \\ \vdots \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots \\ \sigma_{12} & \sigma_2^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}^{-1} \begin{bmatrix} f_1(0) - f_2(0) \\ f_1(1) - f_2(1) \\ \vdots \end{bmatrix}$$

σ_a^2 – variance of component $f(a)$

σ_{ab} – coviance between features $f(a), f(b)$



$$if \quad \sigma_{ab} = 0 \ \forall a \neq b$$

$$then\ D^2_{Mahalanobis}(f_1, f_2) = \sum_n \frac{(f_1(n) - f_2(n))^2}{\sigma_n^2}$$

$$if \ \sigma_{ab} = const \ \forall a = b, and \ \ \sigma_{ab} = 0 \ \forall a \neq b$$

$$D^2_{Mahalanobis}(f_1, f_2) = \frac{1}{const^2} \sum_n (f_1(n) - f_2(n))^2 = D^2_{Euclidean}(f_1, f_2)$$

Mahalanobis distance - general

- General form

$$d_A(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2)$$

- Why is A positive semi-definite (PSD)?
  - If A is not PSD, then d_A could be negative
  - Suppose $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_2$ is an eigenvector corresponding to a negative eigenvalue λ of A

$$\begin{aligned} d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\ &= \mathbf{v}^T A \mathbf{v} \\ &= \lambda \mathbf{v}^T \mathbf{v} = \lambda < 0 \end{aligned}$$

Mahalanobis distance - general

- Properties of a metric:
  - $d(\mathbf{x}, \mathbf{y}) \geq 0$
  - $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$
  - $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
  - $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$
- d_A is not technically a metric $\qquad d_A(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T A(\mathbf{x}_1 - \mathbf{x}_2)$
  - Analogous to Euclidean distance, need the square root:

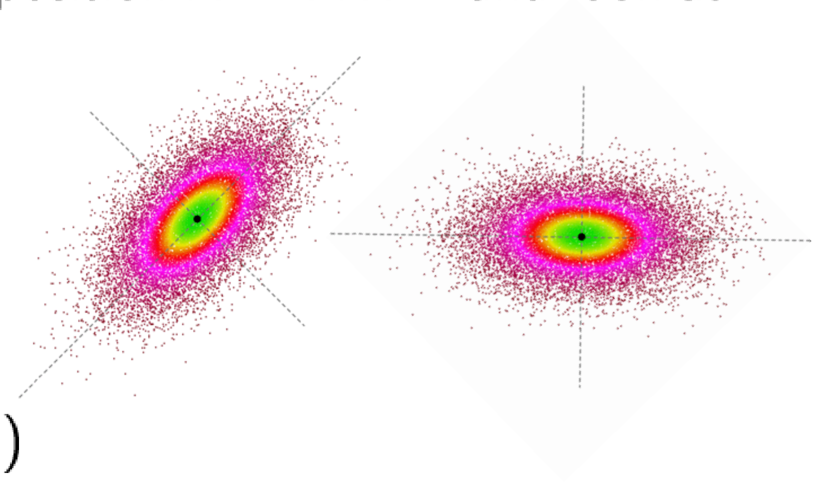
$$\sqrt{d_A(\mathbf{x}_1, \mathbf{x}_2)} = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T A(\mathbf{x}_1 - \mathbf{x}_2)}$$

- Square root of the Mahalanobis distance satisfies all properties if A is strictly positive definite, but if A is positive semi-definite then second property is not satisfied
  - Called a *pseudo-metric*
- In practice, most algorithms work only with d_A

Mahalanobis distance - general

- Can view d_A as the squared Euclidean distance after applying a linear transformation
  - Decompose $A = G^T G$ via Cholesky decomposition
  - (Alternatively, take eigenvector decomposition $A = V \Lambda V^T$ and look at $A = (\Lambda^{1/2} V^T)^T (\Lambda^{1/2} V^T)$)

$$\begin{aligned} d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\ &= (\mathbf{x}_1 - \mathbf{x}_2)^T G^T G (\mathbf{x}_1 - \mathbf{x}_2) \\ &= (G\mathbf{x}_1 - G\mathbf{x}_2)^T (G\mathbf{x}_1 - G\mathbf{x}_2) \\ &= \|G\mathbf{x}_1 - G\mathbf{x}_2\|_2^2 \end{aligned}$$



- Mahalanobis distance is just the squared Euclidean distance after applying the linear transformation G
  - Unitless and scale invariant

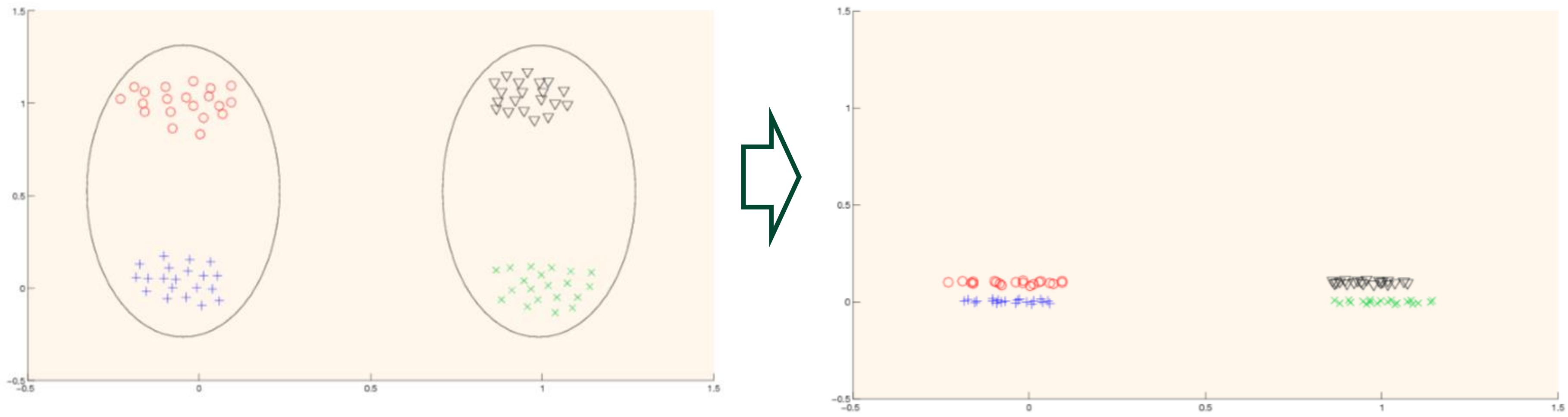
Mahalanobis distance learning - illustration

- Given this grouping we may want to learn matrix *A* such that it brings closer the related groups

$$
\begin{aligned}
d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\
&= (\mathbf{x}_1 - \mathbf{x}_2)^T G^T G (\mathbf{x}_1 - \mathbf{x}_2) \\
&= (G\mathbf{x}_1 - G\mathbf{x}_2)^T (G\mathbf{x}_1 - G\mathbf{x}_2) \\
&= \|G\mathbf{x}_1 - G\mathbf{x}_2\|_2^2
\end{aligned}
$$

learn:

$$
A = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} \qquad G = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{\epsilon} \end{pmatrix}
$$



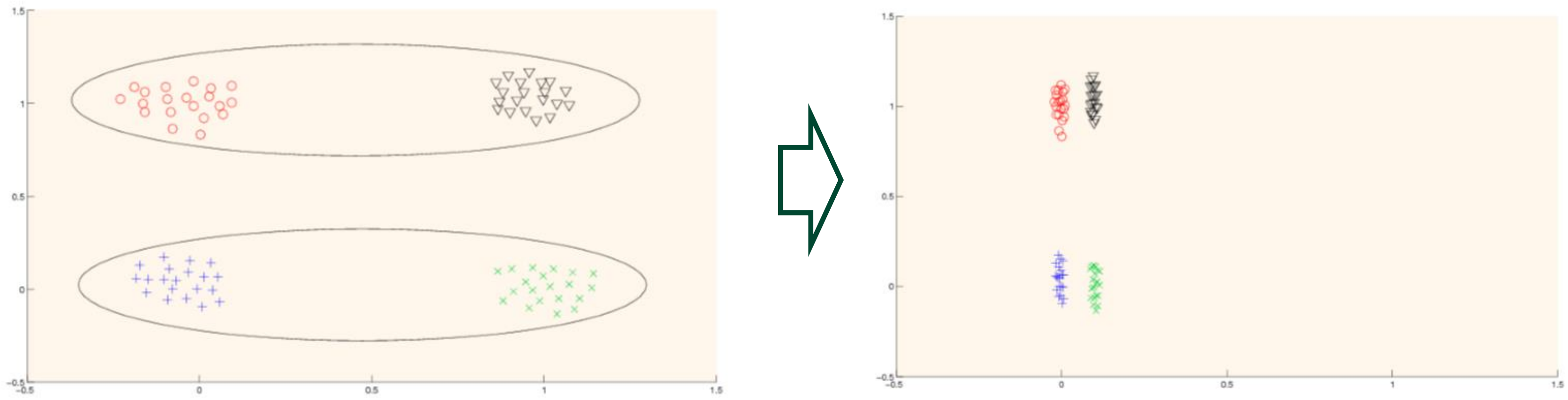
Mahalanobis distance learning - illustration

- Given this grouping we may want to learn matrix *A* such that it brings closer the related groups

$$
\begin{aligned}
d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\
&= (\mathbf{x}_1 - \mathbf{x}_2)^T G^T G (\mathbf{x}_1 - \mathbf{x}_2) \\
&= (G\mathbf{x}_1 - G\mathbf{x}_2)^T (G\mathbf{x}_1 - G\mathbf{x}_2) \\
&= \|G\mathbf{x}_1 - G\mathbf{x}_2\|_2^2
\end{aligned}
$$

learn:

$$A = \begin{pmatrix} \epsilon & 0 \\ 0 & 1 \end{pmatrix} \qquad G = \begin{pmatrix} \sqrt{\epsilon} & 0 \\ 0 & 1 \end{pmatrix}$$



Mahalanobis distance learning - formulation

- Typically 2 main pieces to a Mahalanobis metric learning problem
  - A set of constraints on the distance
  - A regularizer on the distance / objective function
- In the constrained case, a general problem may look like:

$$\begin{array}{ll} \min_A & r(A) \\ \text{s.t.} & c_i(A) \leq 0 \quad 0 \leq i \leq C \\ & A \succeq 0 \end{array}$$

- r is a regularizer/objective on A and c_i are the constraints on A

Mahalanobis distance learning - constraints

- Similarity / Dissimilarity constraints
  - Given a set of pairs $\mathcal{S}$ of points that should be similar, and a set of pairs of points $\mathcal{D}$ of points that should be dissimilar
  - A single constraint would be of the form

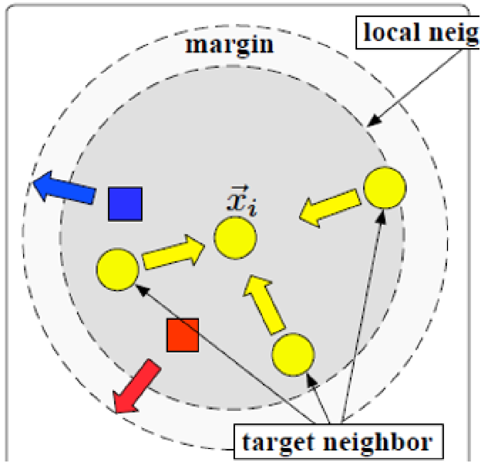
$$d_A(\mathbf{x}_i, \mathbf{x}_j) \leq \ell$$

for $(i, j) \in \mathcal{S}$ or

$$d_A(\mathbf{x}_i, \mathbf{x}_j) \geq u$$

for $(i, j) \in \mathcal{D}$

  - Easy to specify given class labels



- Aggregate distance constraints
  - Constrain the sum of all pairs of same-class distances to be small, e.g.,

$$\sum_{ij} y_{ij} d_A(\mathbf{x}_i, \mathbf{x}_j) \leq 1$$

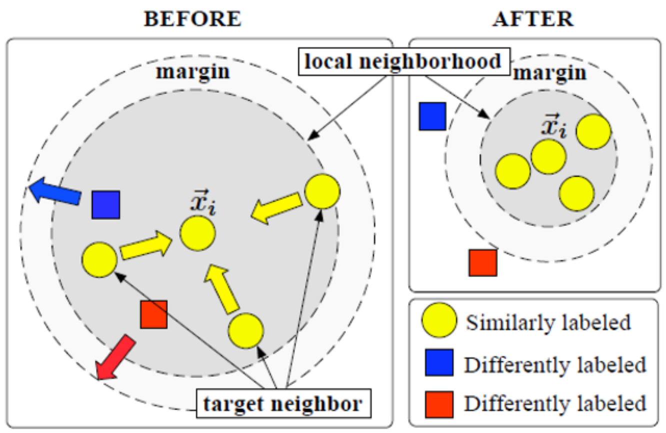
where $y_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class, and 0 otherwise

Mahalanobis distance learning - constraints

- Relative distance constraints
  - Given a triple $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ such that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_k$, a single constraint is of the form

$$d_A(\mathbf{x}_i, \mathbf{x}_j) \leq d_A(\mathbf{x}_i, \mathbf{x}_k) - m,$$

    where m is the margin
  - Popular for ranking problems



Define constraints tailored to k-NN in a local way

- the k nearest neighbours should be of same class ("target neighbors"), while examples of different classes should be kept away ("impostors"):

$S = \{(x_i, x_j) : y_{ij} = 1$ and x_j belongs to the k-neighborhood of $x_i\}$,

$R = \{(x_i, x_j, x_k) : y_{ij} = 1 , y_{ik} = 0\}$.

Mahalanobis distance learning - objectives

- General loss functions D(A,A_0)
  - Distance between matrices A and A_0
  - A is a matrix we learn
  - A_0 is a baseline matrix, which we start the learning with

- Loss/divergence functions
  - Squared Frobenius $\|A - A_0\|_F^2$
  - LogDet divergence: $\text{tr}(AA_0^{-1}) - \log\det(AA_0^{-1}) - d$
  - $\|A\|_F^2$ Frobenius norm
  - $\text{tr}(AC_0)$ (i.e., if C_0 is the identity, this is the trace norm)

Mahalanobis distance learning - objectives

- Frobenius norm – “Euclidean” norm used for matrices

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

- LogDet divergence

$$\begin{aligned} D_{\ell d}(X,Y) &= \operatorname{trace}(XY^{-1}) - \log\det(XY^{-1}) - d, \\ &= \operatorname{trace}(Y^{-1/2}XY^{-1/2}) - \log\det(Y^{-1/2}XY^{-1/2}) - d. \end{aligned}$$

- Can be used as a measure of distance
  - Positive, and zero iff $X = Y$
  - But not symmetric, and triangle inequality does not hold
  - Scale-invariance

$$D_{\ell d}(X,Y) = D_{\ell d}(\alpha X, \alpha Y), \quad \alpha \geq 0$$

- Trace - sum of diagonal elements

$$\operatorname{tr}(A) = \sum_i a_{ii}$$

Mahalanobis distance learning - objectives

- Example 1: $\text{tr}(A)$
  - Trace function is the sum of the eigenvalues
  - Analogous to the ℓ_1 penalty, promotes sparsity
  - Leads to low-rank A
- Example 2: LogDet Divergence
  - Defined only over positive semi-definite matrices
  - Makes computation simpler
  - Possesses other desirable properties
- Example 3: $\|A\|_F^2$
  - Arises in many formulations
  - Easy to analyze and optimize

Mahalanobis distance learning – example A

Problem posed as follows:

$$\begin{aligned} \max_{A} \quad & \textstyle\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \sqrt{d_A(\mathbf{x}_i,\mathbf{x}_j)} \\ \text{s.t.} \quad & c(A) = \textstyle\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d_A(\mathbf{x}_i,\mathbf{x}_j) \le 1 \\ & A \succeq 0. \end{aligned}$$

- Here, $\mathcal{D}$ is a set of pairs of dissimilar pairs, $\mathcal{S}$ is a set of similar pairs
- Objective tries to maximize sum of dissimilar distances
- Constraint keeps sum of similar distances small
  - Use square root in regularizer to avoid trivial solution

[Xing, Ng, Jordan, and Russell; NIPS 2002]

Mahalanobis distance learning – example B

Problem formulated as follows:

$$\begin{aligned} \min_{A} \quad & \|A\|_F^2 \\ \text{s.t.} \quad & d_A(\mathbf{x}_i,\mathbf{x}_k) - d_A(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \quad \forall (i,j,k) \in \mathcal{R} \\ & A \succeq 0. \end{aligned}$$

- Constraints in $\mathcal{R}$ are relative distance constraints
- There may be no solution to this problem; introduce slack variables ξ_{ijk}

$$\begin{aligned} \min_{A,\xi} \quad & \|A\|_F^2 + \gamma \textstyle\sum_{(i,j,k)\in\mathcal{R}} \xi_{ijk} \\ \text{s.t.} \quad & d_A(\mathbf{x}_i,\mathbf{x}_k) - d_A(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall (i,j,k) \in \mathcal{R} \\ & \xi_{ijk} \geq 0 \quad \forall (i,j,k) \in \mathcal{R} \\ & A \succeq 0. \end{aligned}$$

[Schultz and Joachims; NIPS 2002]

Mahalanobis distance learning – example C

- Problem Formulation

$$\begin{aligned} \min_{A} \quad & \textstyle\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d_A(\mathbf{x}_i,\mathbf{x}_j) \\ \text{s.t.} \quad & d_A(\mathbf{x}_i,\mathbf{x}_k) - d_A(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \quad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R} \\ & A \succeq 0. \end{aligned}$$

- Also define set $\mathcal{S}$ of pairs of points $(\mathbf{x}_i, \mathbf{x}_j)$ such that $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors in the same class
- Want to minimize sum of distances of pairs of points in $\mathcal{S}$
- Also want to satisfy the relative distance constraints

- Introduce slack variables

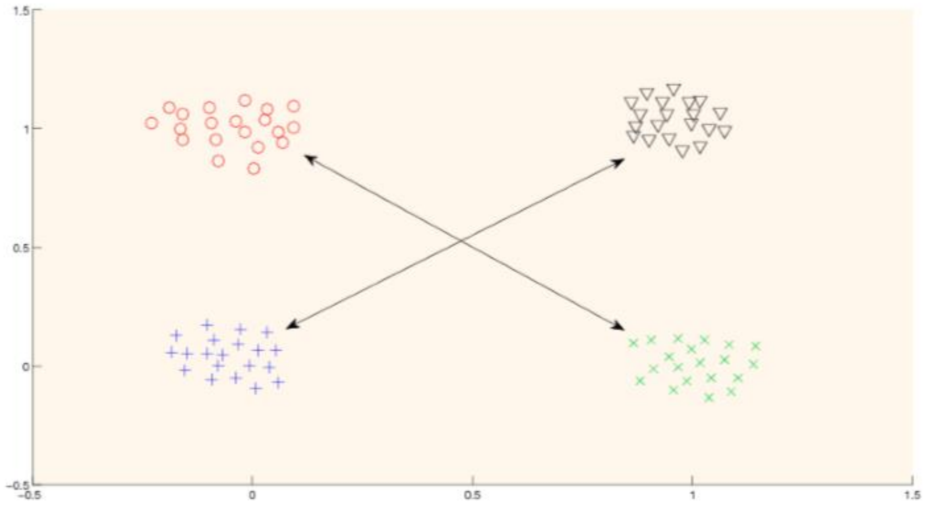
$$\begin{aligned} \min_{A,\xi} \quad & \textstyle\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d_A(\mathbf{x}_i,\mathbf{x}_j) + \gamma \sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R}} \xi_{ijk} \\ \text{s.t.} \quad & d_A(\mathbf{x}_i,\mathbf{x}_k) - d_A(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R} \\ & A \succeq 0, \xi_{ijl} \geq 0. \end{aligned}$$

Imperial College London

Mahalanobis distance learning

- Many existing Mahalanobis distance learning methods can be obtained simply by choosing a regularizer/objective and constraints
    - Memory overhead grows quadratically with the dimensionality of the data
        - Does not scale to high-dimensional data ($d = O(10^6)$ for many image embeddings)
    - Only works for linearly separable data

but
There is no linear transformation for such grouping problem



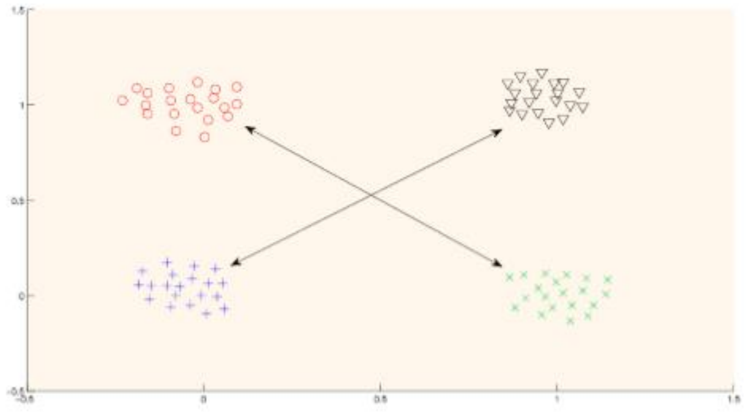
Kernel Distance learning - nonlinear

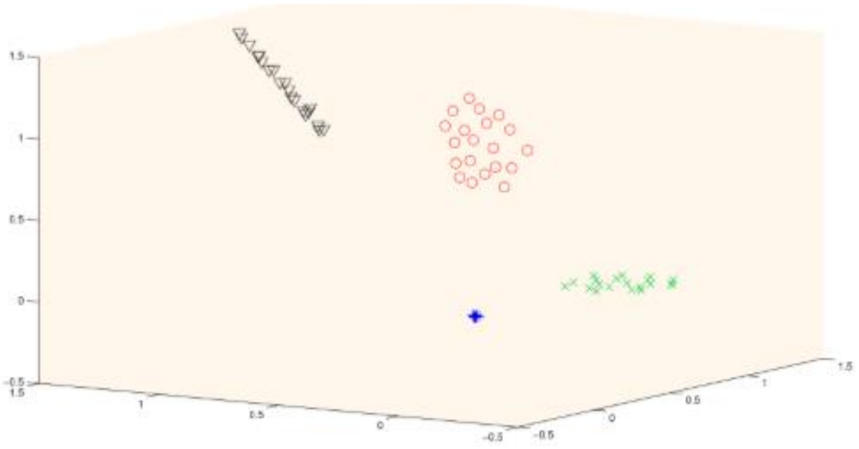
- Map input data to higher-dimensional "feature" space:

$$\mathbf{x} \rightarrow \varphi(\mathbf{x})$$

- Idea: Run machine learning algorithm in feature space
- Use the following mapping:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$





Kernel Distance learning - nonlinear

- Map input data to higher-dimensional "feature" space:

$$\mathbf{x} \rightarrow \varphi(\mathbf{x})$$

- Idea: Run machine learning algorithm in feature space
- Use the following mapping:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Kernel function: $\kappa(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$
- "Kernel trick" — no need to explicitly form high-dimensional features
- In this example: $\langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle = (\mathbf{x}^T\mathbf{y})^2$

Kernel Distance learning - nonlinear

- Kernel function (Definition)
  - A symmetric similarity function K is a kernel if there exist a (possibly implicit) mapping function $\phi : X \to H$ from instance space *X* to a Hilbert space *H* such that *K* can be written as an inner product in *H:* $K(x_i , x_j) = < \phi (x_i) , \phi(x_j) >$
  - *K* is a kernel if it is *positive semi-definite* i.e. the following

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

holds for any $n \in \mathbb{N}, x_1, \ldots, x_n \in \mathcal{X}, c_1, \ldots, c_n \in \mathbb{R}$

Kernel Distance learning - nonlinear

- Main idea
  - Take an existing learning algorithm
  - Write it using inner products
  - Replace inner products $\mathbf{x}^T\mathbf{y}$ with kernel functions $\varphi(\mathbf{x})^T\varphi(\mathbf{y})$
  - If $\varphi(\mathbf{x})$ is a non-linear function, then algorithm has been *implicitly* non-linearly mapped
- Examples of kernel functions

$$\begin{aligned}
\kappa(\mathbf{x},\mathbf{y}) &= (\mathbf{x}^T\mathbf{y})^p \quad \text{Polynomial Kernel}\\
\kappa(\mathbf{x},\mathbf{y}) &= \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right) \quad \text{Gaussian Kernel}\\
\kappa(\mathbf{x},\mathbf{y}) &= \tanh(c(\mathbf{x}^T\mathbf{y})+\theta) \quad \text{Sigmoid Kernel}
\end{aligned}$$

- Kernel functions also defined over objects such as images, trees, graphs, etc.

Kernel Distance learning - kmeans example

Recall the k-means clustering algorithm

- Repeat until convergence:
  - Compute the means of every cluster π_c

$$\mu_c = \frac{1}{|\pi_c|} \sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}_i$$

  - Reassign points to their closest mean by computing

$$\|\mathbf{x} - \mu_c\|_2^2$$

  for every data point $\mathbf{x}$ and every cluster π_c

Kernelization of k-means

- Expand $\|\mathbf{x} - \mu_c\|_2^2$ as $\quad \mathbf{x}^T\mathbf{x} - \frac{2\sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}^T\mathbf{x}_i}{|\pi_c|} + \frac{\sum_{\mathbf{x}_i,\mathbf{x}_j \in \pi_c} \mathbf{x}_i^T\mathbf{x}_j}{|\pi_c|^2}$
  - No need to explicitly compute the mean; just compute this for every point to every cluster
- Replace inner products with kernels, and this is kernel k-means

$$\kappa(\mathbf{x},\mathbf{x}) - \frac{2\sum_{\mathbf{x}_i \in \pi_c} \kappa(\mathbf{x},\mathbf{x}_i)}{|\pi_c|} + \frac{\sum_{\mathbf{x}_i,\mathbf{x}_j \in \pi_c} \kappa(\mathbf{x}_i,\mathbf{x}_j)}{|\pi_c|^2}$$

Kmeans finds linear separators for cluster boundaries, kernel Kmeans finds non-linear separators

Distance learning – Mahalanobis kernel summary

Consider the following *kernelized* problem

- You are given a kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y})$
- You want to run a metric learning algorithm in kernel space
  - Optimization algorithm cannot use the explicit feature vectors $\varphi(\mathbf{x})$
  - Must be able to compute the distance/kernel over arbitrary points (not just training points)

- Mahalanobis distances:

$$d_A(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T A(\mathbf{x} - \mathbf{y}) \qquad d_A(\mathbf{x}, \mathbf{y}) = (\varphi(\mathbf{x}) - \varphi(\mathbf{y}))^T A(\varphi(\mathbf{x}) - \varphi(\mathbf{y}))$$

- Inner products / kernels:

$$\kappa_A(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T A\mathbf{y} \qquad \kappa_A(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T A\varphi(\mathbf{y})$$

- Algorithms for constructing A learn both measures

- Can be thought of as a kind of *kernel learning* problem