Feature Selection / Extraction

The curse of dimensionality

Feature extraction: Principal Component Analysis, Fisher Linear Discriminant Analysis

Feature Selection: basis ideas

Summary of episodes 1-3

- **So far, we have seen that:**
  - If class-conditional densities, p(x|ω$_i$) and prior probabilities, P(ω$_i$), are known, one can set up optimal classifiers (e.g. likelihood ratio tests for dichotomy).
  - If only the parametric shape of class-conditional densities is known, it is possible to estimate their parameters using a set of labeled training samples.
  - If nothing is known, it is however possible to use non-parametric density estimation techniques to estimate class-conditional probabilities from a set of labeled training samples.
- **In the next chapter, we will go further by considering the more difficult case when training data come unlabeled…**
- **Before that, in the present chapter, we focus on what happens when the dimensionality of the problem, D, becomes large**

The issue of dimensionality

- **Intuitively, we might expect that considering more features helps increasing the discrimination between classes**
  - e.g. 2 Gaussian, equiprobable classes with same covariance: $p(\mathbf{x}|\omega_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$. It is easy to show that the probability of classification error is

$$P_e = \frac{1}{\sqrt{2pi}} \int_{\theta=\frac{r}{2}}^{+\infty} \exp\left(-\frac{y^2}{2}\right) dy$$

  where r is the Mahalanobis distance: $r^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$

  - In particular, if $\Sigma = diag(\sigma_i^2)$

$$r^2 = \sum_{i=1}^{D} \left(\frac{\boldsymbol{\mu}_{1_i} - \boldsymbol{\mu}_{2_i}}{\sigma_i} \right)^2$$

  - In other words, the larger D, the larger *r*, hence, the smaller P_e should be!
- **This is not what can be observed in practice!**
  - Classification performance degrades as D becomes large...

Trunk's experiment [Trunk, PAMI 1979]

Example where

$\boldsymbol{\omega_1} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I_D})$

$\boldsymbol{\omega_2} \sim \mathcal{N}(-\boldsymbol{\mu}, \boldsymbol{I_D})$

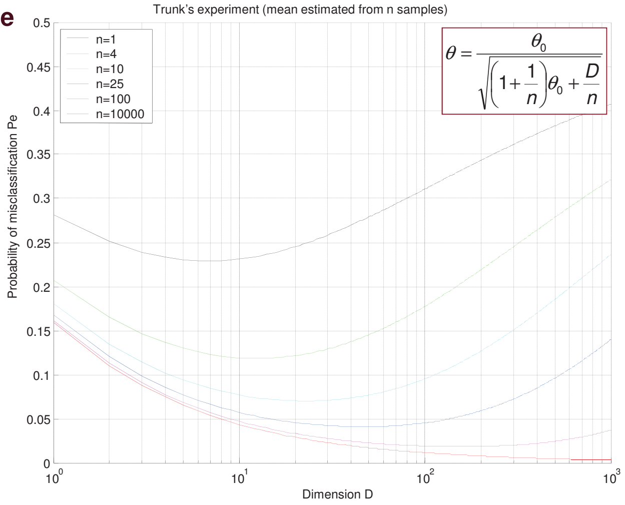
$\boldsymbol{\mu}_i = 1/\sqrt{\boldsymbol{i}}$

Then:

$$\boxed{P_e = \frac{\int_\theta^{+\infty} e^{-y^2/2}.dy}{\sqrt{2\pi}}}$$

with $\theta = r/2$

$$\theta_0 = \sum_{i=1}^{i=D} \frac{1}{i}$$

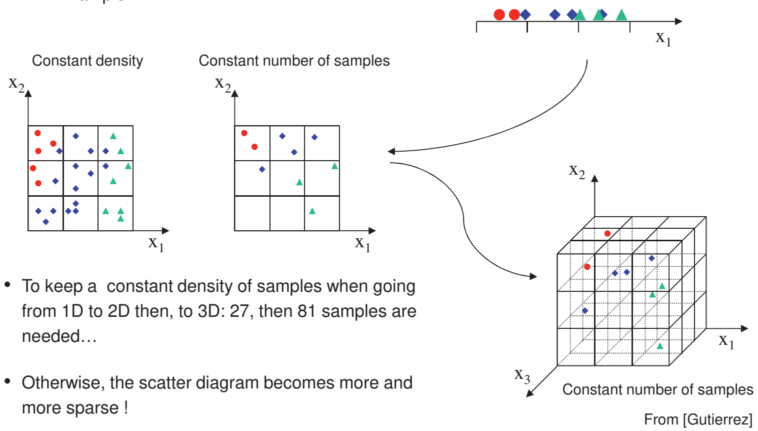


Trunk's experiment (mean estimated from n samples)

$$\theta = \frac{\theta_0}{\sqrt{\left(1+\frac{1}{n}\right)\theta_0 + \frac{D}{n}}}$$

The issue of dimensionality

- **A theoretical study of the problem is very difficult [Duda]**
  - An explanation: the number of samples is finite, so PDF estimation is not very accurate...
  - Example:



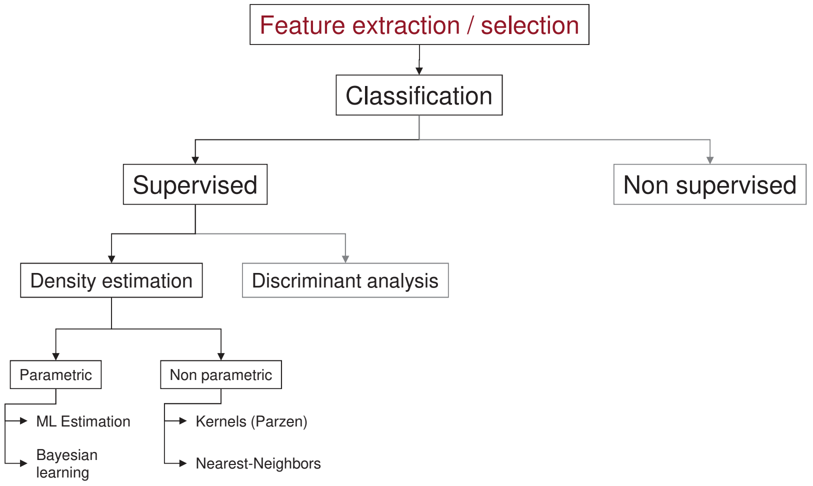
- To keep a constant density of samples when going from 1D to 2D then, to 3D: 27, then 81 samples are needed…
- Otherwise, the scatter diagram becomes more and more sparse !

From [Gutierrez]

The curse of dimensionality ?

- **Several problems occur as D increases**
  - Theoretical: scatter diagrams become more and more sparse, distances and volumes have weird properties in high dimension… [Bishop]
  - Practical: mean, covariance matrices yield heavier computational burden…
- **The set of problems that arise in data analysis when dimensionality increases is called the "curse of dimensionality" [Belmann 1961]**
  - There are several remedies (see e.g. [Duda])
  - One of them is to apply a *dimensionality reduction* technique
- **Another motivation for dimension reduction is visualization (needs $D \leq 3$ ...)**
- **Feature selection or extraction techniques**
  - Completely redesign the feature extraction algorithm…
  - Select a subset of features or re-combine features

A hierarchy of methods


Feature extraction / selection

Classification

Supervised

Non supervised

Density estimation

Discriminant analysis

Parametric

Non parametric

ML Estimation

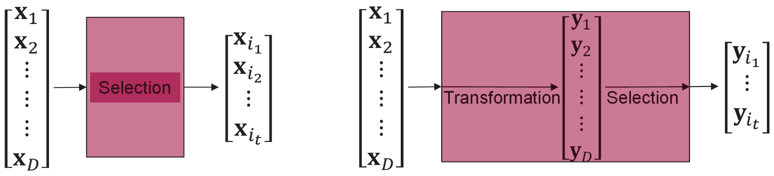
Bayesian learning

Kernels (Parzen)

Nearest-Neighbors

Dimensionality reduction techniques

- **Two approaches:**
  - Combine features and select a reduced number of the resulting ones
  - = Feature selection in transformed space or feature extraction
  - Select features directly in feature space: feature selection


Selection

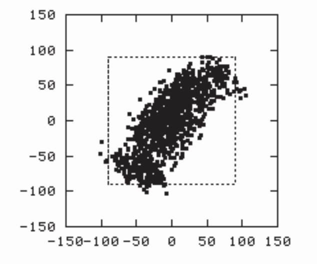
Transformation

Selection

- We will study 2 well-known, linear feature extraction techniques: Principal Component Analysis (PCA) and Fisher's Linear Discriminant Analysis (LDA).
- Then, we will introduce nonlinear extraction techniques, and selection methods

Principal Component Analysis (PCA)

- **The choice of the transformation is generally guided by an optimization criterion**



From [Gutierrez]

- **In the case of PCA,**
  - The criterion assesses the quality of representation on a smaller-dimensional subspace
  - Maximizing quality amounts to defining a rotation of the coordinate system

- **A rotation of the coordinate system in feature space may reveal the structure of the data (see above "toy example")**
  - This is what PCA does, with a particular meaning of the word "structure"...

Case 1: best representation by one point

- **Goal: find the best representation of D-dimensional vectors, $\mathbf{x}_k$, $k=1..n$, by a unique vector, $\mathbf{x}_0$**

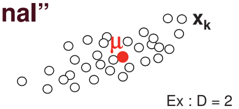
- **Representation quality criterion: mean squared error**

$$J_0(\mathbf{x}_0)=\frac{1}{n}\sum_{k=1}^{n}\left\|\mathbf{x}_k-\mathbf{x}_0\right\|^2$$

- **We seek $\mathbf{x}_0$ that minimizes this criterion, i.e. the solution of:**

$$\frac{\partial J_0(\mathbf{x}_0)}{\partial \mathbf{x}_0}=0=\frac{-2}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\mathbf{x}_0)\longrightarrow\boxed{\mathbf{x}_0=\frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k=\boldsymbol{\mu}}$$

- **The sample mean is the best "0-dimensional" representation of the point cloud in the sense of mean square estimation**



Ex : D = 2

Case 2: best representation by a straight line

- **Goal:**
  - Find the best 1D representation of the data, according to the model:

$$\mathbf{x}\approx\mu+y.\mathbf{e}\quad\text{with}\quad\|\mathbf{e}\|=1$$

  - Straight line running through the sample mean, μ with (unknown) direction $\mathbf{e}$.
  - Scalar, y: distance of any point $\mathbf{x}$ lying on the line from μ.
  - In particular, we have for any k:

$$\mathbf{x}_k\approx\mu+y_k.\mathbf{e}$$

- **Mean squared error (MSE):** $J_1(y_1,\ldots y_n,\mathbf{e})=\frac{1}{n}\sum_{k=1}^{n}\left\|(\boldsymbol{\mu}+y_k.\mathbf{e})-\mathbf{x}_k\right\|^2$

- **Remark:**

$$\begin{aligned}J_1(y_1,\ldots y_n,\mathbf{e})&=\frac{1}{n}\sum_{k=1}^{n}\left\|y_k.\mathbf{e}-(\mathbf{x}_k-\boldsymbol{\mu})\right\|^2\\&=\frac{1}{n}\sum_{k=1}^{n}\left\|y_k.\mathbf{e}\right\|^2-\frac{2}{n}\sum_{k=1}^{n}y_k.\mathbf{e}^T(\mathbf{x}_k-\boldsymbol{\mu})+\frac{1}{n}\sum_{k=1}^{n}\left\|\mathbf{x}_k-\boldsymbol{\mu}\right\|^2\end{aligned}$$

Representation by a line: the coordinates

- **The set of coefficients y_k that minimize the squared error criterion is such that:**

$$\frac{\partial J_1(y_1,\ldots y_n,\mathbf{e})}{\partial y_k}=0\quad\forall k=1..n$$

- **with** $J_1(y_1,\ldots y_n,\mathbf{e})=\frac{1}{n}\sum_{k=1}^{n}y_k{}^2-\frac{2}{n}\sum_{k=1}^{n}y_k.\mathbf{e}^T(\mathbf{x}_k-\boldsymbol{\mu})+\frac{1}{n}\sum_{k=1}^{n}\left\|\mathbf{x}_k-\boldsymbol{\mu}\right\|^2$

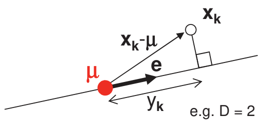
- **which leads to** $0=\frac{2}{n}y_k-\frac{2}{n}\mathbf{e}^T(\mathbf{x}_k-\boldsymbol{\mu})$

$$\boxed{y_k=\mathbf{e}^T(\mathbf{x}_k-\boldsymbol{\mu})}$$

- **The coordinates on the line are given by the orthogonal projection of the (centered) data on e.**

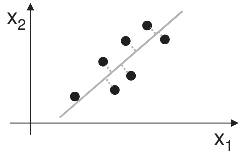
PCA vs. Orthogonal regression

$$y_k = \mathbf{e}^T(\mathbf{x}_k - \boldsymbol{\mu})$$



- **PCA is akin to orthogonal regression**

$$J_1(y_1,\dots y_n,\mathbf{e}) = \frac{1}{n}\sum_{k=1}^{n}\left\|(\boldsymbol{\mu} + y_k.\mathbf{e}) - \mathbf{x}_k\right\|^2$$



Case 2: Re-formulating the problem

- **Let us identify in J_1 the expression of the newly calculated y_k's**

$$J_1(y_1,\dots y_n,\mathbf{e}) = \frac{1}{n}\sum_{k=1}^{n} y_k^2.\overbrace{\|\mathbf{e}\|^2}^{1} - \frac{2}{n}\sum_{k=1}^{n} y_k.\overbrace{\mathbf{e}^T(\mathbf{x}_k - \boldsymbol{\mu})}^{y_k} + \frac{1}{n}\sum_{k=1}^{n}\|\mathbf{x}_k - \boldsymbol{\mu}\|^2$$

- **Thus, we obtain:**

$$J_1(y_1,\dots y_n,\mathbf{e}) = -\frac{1}{n}\sum_{k=1}^{n} y_k^2 + \frac{1}{n}\sum_{k=1}^{n}\|\mathbf{x}_k - \boldsymbol{\mu}\|^2$$

- **Then:**

$$J_1(y_1,\dots y_n,\mathbf{e}) = -\frac{1}{n}\sum_{k=1}^{n}\mathbf{e}^T(\mathbf{x}_k - \boldsymbol{\mu}).(\mathbf{x}_k - \boldsymbol{\mu})^T\mathbf{e} + \frac{1}{n}\sum_{k=1}^{n}\|\mathbf{x}_k - \boldsymbol{\mu}\|^2$$

- **And, finally :**

$$J_1(y_1,\dots y_n,\mathbf{e}) = -\frac{1}{n}\mathbf{e}^T\left(\sum_{k=1}^{n}(\mathbf{x}_k - \boldsymbol{\mu}).(\mathbf{x}_k - \boldsymbol{\mu})^T\right)\mathbf{e} + \frac{1}{n}\sum_{k=1}^{n}\|\mathbf{x}_k - \boldsymbol{\mu}\|^2$$

Analyzing the new formulation of the problem

- **We finally obtained two terms:**

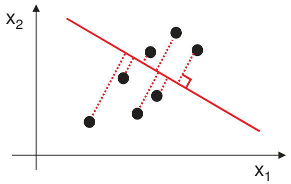
$$J_1(y_1,\dots y_n,\mathbf{e}) = -\mathbf{e}^T\underbrace{\left(\frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \boldsymbol{\mu}).(\mathbf{x}_k - \boldsymbol{\mu})^T\right)}_{\Sigma}\mathbf{e} + \underbrace{\frac{1}{n}\sum_{k=1}^{n}\|\mathbf{x}_k - \boldsymbol{\mu}\|^2}_{\text{independent from } \mathbf{e}}$$

- **where Σ is the sample covariance matrix of the x_k's**
- **But, most of all, $e^T\Sigma e$ is the variance of y, as shown below**
  - Note: y has zero mean (which can be straightforwardly checked!)

$$Var[y] = \frac{1}{n}\sum_{k=1}^{n} y_k^2 - \underbrace{\left[\frac{1}{n}\sum_{k=1}^{n} y_k\right]^2}_{0} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{e}^T(\mathbf{x}_k - \boldsymbol{\mu}).(\mathbf{x}_k - \boldsymbol{\mu})^T\mathbf{e} = \mathbf{e}^T\Sigma\mathbf{e}$$

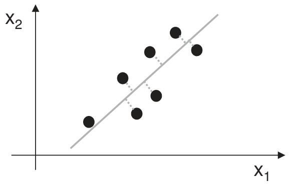
Interpretation

⇨ **Minimizing J_1 with respect to e ⇔ seeking the axis along which the variance (spread) of point coordinates is maximal...**



☹ **High** projection error

low variance



☺ **Low** projection error

high variance

Case 2: Solving the problem

- **To determine e, we must:**
  - Maximize $\mathbf{e}^T\Sigma\mathbf{e}$
  - Under the constraint: $||\mathbf{e}||^2 = \mathbf{e}^T\mathbf{e} = 1$

- **Usual method: Lagrange multipliers**
  - see Optimization course
  - Let us define: $J_1' = \mathbf{e}^T\Sigma\mathbf{e} - \lambda(\mathbf{e}^T\mathbf{e} - 1)$

  where λ is a Lagrange Multiplier

$$\frac{\partial J_1'}{\partial \mathbf{e}} = 0 = 2\Sigma\mathbf{e} - 2\lambda\mathbf{e} \longrightarrow \boxed{\Sigma\mathbf{e} = \lambda\mathbf{e}}$$

- **Remember we are trying to maximize var(y)=$\mathbf{e}^T\Sigma\mathbf{e}$= λ $\mathbf{e}^T\mathbf{e}$= λ,**
  - Hence, λ is chosen as the largest eigenvalue of the covariance matrix, Σ and **e** is the corresponding eigenvector

Case 3: generalization to d dimensions

- **In that case, the model is written as**

$$\mathbf{x} \approx \boldsymbol{\mu} + \mathcal{E}.\mathbf{y}$$

  - where **y** is a coordinate vector and $\mathcal{E}$ is a matrix whose columns are the vectors $\mathbf{e}_i$.

$$J_n(\mathbf{y}_1,\ldots,\mathbf{y}_n,\mathbf{e}_1,\ldots,\mathbf{e}_d) = \frac{1}{n}\sum_{k=1}^{n}\left\|\left(\boldsymbol{\mu} + \sum_{i=1}^{d}\mathbf{y}_{ki}\mathbf{e}_i\right) - \mathbf{x}_k\right\|^2$$

  - It can be shown that the minimum w.r.t. **y** is obtained for the projection:

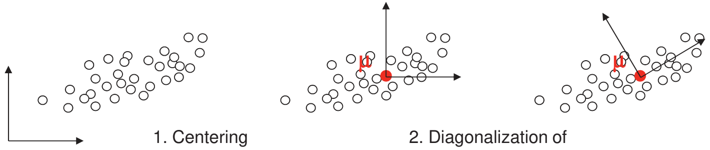
$$\boxed{\mathbf{y} = \mathcal{E}^T(\mathbf{x}_k - \boldsymbol{\mu})}$$

  - And that the column vectors of $\mathcal{E}$, which minimize J_n, are given by the eigenvectors Φ of the sample covariance matrix Σ : $\Phi^T\Sigma\Phi = \Lambda = \text{diag}\{\lambda_i\}$
  - They are orthogonal and represent the principal axes (of variation, see next slide).

Interpretation

- **2 phases:**

E.g. D = 2



1. Centering

2. Diagonalization of covariance matrix

- **PCA is a rotation because Φ is orthonormal ($\Phi^T\Phi=\Phi\Phi^T=I$)**
  - This rotation de-correlates the data because Σ_y becomes diagonal

$$\Sigma_\mathbf{y} = E\left[\mathbf{y}\mathbf{y}^T\right] = E\left[\Phi^T(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\Phi\right] = \Phi^T\Sigma\Phi = \Lambda$$

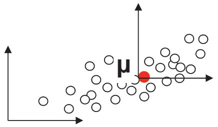
  - The variance on each axis is:

$$E\left[\mathbf{y}_{,i}^2\right] = E\left[\Phi_i^T(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\Phi_i\right] = \Phi_i^T\Sigma\Phi_i = \lambda_i$$

Reconstruction and truncation

- **Sample mean and covariance matrix**

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k \qquad \Sigma = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\boldsymbol{\mu})(\mathbf{x}_k-\boldsymbol{\mu})^T$$



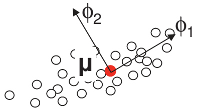
1. Centering (D=2)

- **Diagonalization (rotation)**

$$\Sigma = \Phi\Lambda\Phi^T \qquad \Phi = \begin{bmatrix} \vdots & & \vdots & & \vdots & & \vdots \\ \phi_1 & \cdots & \phi_d & \cdots & \phi_n & \cdots & \phi_D \\ \vdots & & \vdots & & \vdots & & \vdots \end{bmatrix}$$

$$\mathbf{y} = \Phi^T(\mathbf{x}-\boldsymbol{\mu})$$

$$\mathbf{x} = \boldsymbol{\mu} + \Phi\mathbf{y} \qquad \Lambda = diag(\lambda_i)_{i=1\ldots D}$$

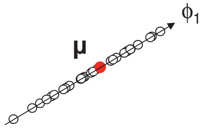


2. Covariance matrix diagonalization

- **Projection on eigenspace (with truncation)**

$$\mathbf{y} = \Phi_d^T(\mathbf{x}-\boldsymbol{\mu}) \qquad \Phi_\mathrm{d} = \begin{bmatrix} \vdots & & \vdots \\ \phi_1 & \cdots & \phi_d \\ \vdots & & \vdots \end{bmatrix}$$

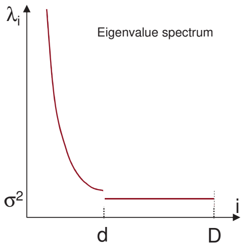
$$\mathbf{x} \cong \boldsymbol{\mu} + \Phi_d\mathbf{y}$$



3. Projection + truncation (d=1)

Variances

- **Eigenvectors are sorted by decreasing order of the corresponding eigenvalues (i.e. variances)**
  - PCA axes are ordered by decreasing order of variance
  - Variability is most often concentrated on the first axes



- **In general, $d \ll D$ ➔ dimensionality reduction.**
- **The percentage of explained variance, $\frac{\sum_1^d \lambda_i}{\sum_1^D \lambda_i}$ provides a principled way of selecting d**
- **Variances along neglected dimension can be approximated optimally (in the sense of Maximum Likelihood) using their mean**

$$\sigma^2 = \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i$$

[Moghaddam]

- **Note that $rank(\Sigma) = \min(D, n-1)$ ➔ $n-1$ eigenvalues computed if $(n-1) < D$**

Probabilistic PCA (PPCA)

- **Proposed by Tipping & Bishop (1997,1999) and Roweis (1998)**
- **Linear Generative Model**

$$\mathbf{x} = \boldsymbol{\mu} + \mathcal{E}\mathbf{y} + \boldsymbol{\varepsilon}$$

where:

- The latent variable, $\boldsymbol{y}$, is Gaussian: $p(\boldsymbol{y}) \sim \mathcal{N}(0,I)$
- The noise, $\boldsymbol{\varepsilon}$ is also Gaussian $p(\varepsilon) \sim \mathcal{N}(0,\sigma^2 I)$, so: $p(\boldsymbol{x}|\boldsymbol{y}) \sim \mathcal{N}(\mu + \mathcal{E}\boldsymbol{y}, \sigma^2 I)$
- $\boldsymbol{y}$ and $\boldsymbol{\varepsilon}$ are independent.

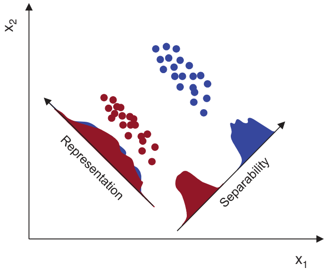
- **ML solution**

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k \qquad \mathcal{E} \propto \Phi \qquad \sigma^2 = \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i$$

- **PPCA brings several advantages compared with PCA, e.g.**
  - Derivation of an EM algorithm for fast computation of the first few eigenvectors
  - PPCA model can be run generatively to draw samples from the distribution

From PCA to LDA

- **PCA**
  - Minimize a goodness-of-representation criterion
  - Amounts to a rotation of coordinates in feature space
  - …in such a way as to represent most of the data variability on the first (few) dimensions of the transformed space.
- **Fisher Discriminant Analysis (LDA)**
  - Optimizes a class-separability, or classification criterion.
  - This does not provide the same directions
  - But it remains a linear transform: $\mathbf{y} = \mathrm{W}^T\mathbf{x}$



From [Gutierrez]

Fisher Linear Discriminant Analysis (LDA)

- **Case 1: two-class problem (dichotomy)**
  - Consider a set of n samples (D-dimensional vectors) : $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$ among which n_1 belong to class ω_1 and n_2, to class ω_2
  - Projecting one sample on an axis $\mathbf{w}$ yields the scalar $y = \mathbf{w}^T\mathbf{x}$
  - The vector $\mathbf{w}$ is a projection direction (its norm does not really matter)
- **We shall seek a direction w along which the projections are as separable as possible**
- **In order to define a separability criterion, recall that**
  - The class means in feature space and in transformed space (for i=1,2) are:

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{x\in\omega_i} \mathbf{x} \qquad \tilde{m}_i = \frac{1}{n_i}\sum_{y\in\omega_i} y = \frac{1}{n_i}\sum_{x\in\omega_i} \mathbf{w}^T\mathbf{x} = \mathbf{w}^T\mathbf{m}_i$$

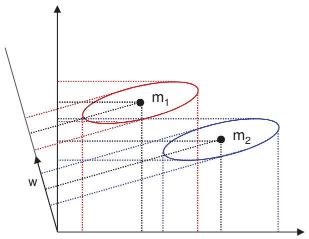
LDA, case 1: two classes

- **We should try to maximize**

$$J(\mathbf{w}) = \tilde{m}_1 - \tilde{m}_2 = \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)$$

$$\longrightarrow \mathbf{w} \propto (\mathbf{m}_1 - \mathbf{m}_2)$$

  - Does not account for intra-class variability
  - To measure intra-class variability, one may use the variance (or the dispersion, which is proportional to the variance) along the axis, **w**.



$$\tilde{s}_i^2 = \sum_{y\in\omega_i} (y - \tilde{m}_i)^2$$

- **Fishers LDA maximizes:**

$$\boxed{J(\mathbf{w}) = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}}$$

- Separate means
- Keep classes compact

along projection axis, **w**

LDA, case 1: two classes, expression of J(w)

- **Goal: obtain** $J(\mathbf{w}) = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$ **as a function of w**

1. Denominator

$$\tilde{s}_i^2 = \sum_{y\in\omega_i} (y - \tilde{m}_i)^2 = \sum_{x\in\omega_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_i)^2 = \mathbf{w}^T \sum_{x\in\omega_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \mathbf{w} = \mathbf{w}^T S_i \mathbf{w}$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^T (S_1 + S_2)\mathbf{w} = \mathbf{w}^T S_{\text{intra}} \mathbf{w}$$

2. Numerator

$$(\tilde{m}_1 - \tilde{m}_2)^2 = (\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2)^2 = \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} = \mathbf{w}^T S_{\text{inter}} \mathbf{w}$$

3. New expression

$$\boxed{J(w) = \frac{\mathbf{w}^T S_{\text{inter}} \mathbf{w}}{\mathbf{w}^T S_{\text{intra}} \mathbf{w}}}$$

LDA, case 1: maximizing J(w)

- **Canceling the derivative of J(w) leads to**

$$0 = \left[\frac{\partial(\mathbf{w}^T S_{\text{inter}} \mathbf{w})}{\partial \mathbf{w}} \mathbf{w}^T S_{\text{intra}} \mathbf{w} - \mathbf{w}^T S_{\text{inter}} \mathbf{w} \frac{\partial(\mathbf{w}^T S_{\text{intra}} \mathbf{w})}{\partial \mathbf{w}}\right]$$

$$0 = \left[\frac{\mathbf{w}^T S_{\text{intra}} \mathbf{w}}{\mathbf{w}^T S_{\text{intra}} \mathbf{w}}\right] S_{\text{inter}} \mathbf{w} - \left[\frac{\mathbf{w}^T S_{\text{inter}} \mathbf{w}}{\mathbf{w}^T S_{\text{intra}} \mathbf{w}}\right] S_{\text{intra}} \mathbf{w} = S_{\text{inter}} \mathbf{w} - J(\mathbf{w}).S_{\text{intra}} \mathbf{w}$$

$$S_{\text{intra}}^{-1} S_{\text{inter}} \mathbf{w} = J(\mathbf{w}).\mathbf{w} \qquad \text{(if } S_{\text{intra}} \text{ is non-singular)}$$

- **Remarks**
  - $S_{\text{inter}}\mathbf{w}$ is collinear to $(\mathbf{m}_1\text{-}\mathbf{m}_2)$: $S_{\text{inter}}\mathbf{w} = (\mathbf{m}_1\text{-}\mathbf{m}_2)\underbrace{(\mathbf{m}_1\text{-}\mathbf{m}_2)^T \mathbf{w}}_{\text{scalar}}$
  - Since only the direction of **w** is sought for
    → the scale factor $(\mathbf{m}_1\text{-}\mathbf{m}_2)^T \mathbf{w} / J(\mathbf{w})$ can be neglected

$$\Rightarrow \quad \hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, J(\mathbf{w}) = S_{\text{intra}}^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Fisher Linear Discriminant (1936)

- **The 1 dimensional projection direction which maximizes J is:**

$$\boxed{\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, J(\mathbf{w}) = S_{\text{intra}}^{-1}(\mathbf{m}_1 - \mathbf{m}_2)}$$

- **This is not a discriminant, but a direction… So ???**
  - Once the direction of maximal separability has been found, it remains to determine a threshold $\mathbf{x}_0$ along this axis. The classification rule is then to assign **x** to class ω_1 if:

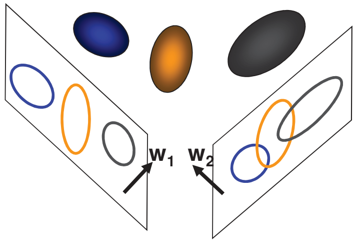
$$\hat{\mathbf{w}}^T(\mathbf{x} - \mathbf{x}_0) > 0$$

  - This resembles what we have seen in the 2-class Gaussian case with equal covariance matrices! It can be shown that this is indeed the same solution…
  - As a consequence, the above rule is optimal in the Gaussian case.

Generalization to C classes *(1/3)*

- **The generalization to C classes is quite straightforward**
- **In that case, we can find at most C–1 projection axes**
  - Tacitly, it is assumed that D ≥ C
  - The vectors $\mathbf{w}_i$ are arranged in columns to form a matrix, W
  - The projections y_i are stored in a C-1 component vector, **y**

$$y_i = \mathbf{w}_i^T\mathbf{x} \quad \rightarrow \quad \mathbf{y} = W^T\mathbf{x}$$



Three dimensional distributions are projected onto planes (described by their normal vectors $\mathbf{w}_1$ and $\mathbf{w}_2$). The best separability is in the plane associated with $\mathbf{w}_1$.

Following [Duda]

Generalization to C classes *(2/3)*

- **General dispersion matrices**
  - Generalization of intra-class dispersion matrix :

$$\boxed{S_{\text{intra}} = \sum_{i=1}^{C} S_i}$$

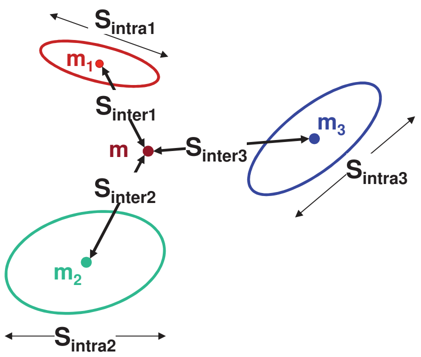
$$\textit{where } S_i = \sum_{x\in\omega_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T$$

$$\textit{and } \mathbf{m}_i = \frac{1}{N_i}\sum_{x\in\omega_i}\mathbf{x}$$

  - Generalization of inter-class dispersion matrix:

$$\boxed{S_{\text{inter}} = \sum_{i=1}^{C} N_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^T}$$

$$\textit{where } \mathbf{m} = \frac{1}{N}\sum_{\forall x}\mathbf{x} = \frac{1}{N}\sum_{i=1}^{C} N_i\mathbf{m}_i$$



m is the center of mass of class means

Following [Gutierrez]

Generalization to C classes *(3/3)*

- **In transformed space, it can be shown that if**

$$\tilde{S}_{\text{intra}} = \sum_{i=1}^{C}\sum_{y\in\omega_i}(y-\tilde{m}_i)(y-\tilde{m}_i)^T \qquad \tilde{S}_{\text{inter}} = \sum_{i=1}^{C} N_i(\tilde{m}_i-\tilde{m})(\tilde{m}_i-\tilde{m})^T$$

Then, $\tilde{S}_{\text{intra}} = W^T S_{\text{intra}} W$ **and** $\tilde{S}_{\text{inter}} = W^T S_{\text{inter}} W$

- **We need a scalar function. To this end, we may use the determinant of dispersion matrices. Why ?**
  - The determinant of a (dispersion) matrix is the product of its eigenvalues
  - i.e. product of variances in the principal directions. The determinant is thus homogeneous to the square of the hyperellipsoidal scattering volume (recall the gaussian case)
- **The criterion can be written as:**

$$\boxed{J(W) = \frac{\left|\tilde{S}_{\text{inter}}\right|}{\left|\tilde{S}_{\text{intra}}\right|} = \frac{\left|W^T S_{\text{inter}} W\right|}{\left|W^T S_{\text{intra}} W\right|}}$$

Generalization to C classes, expression of $\widehat{W}$

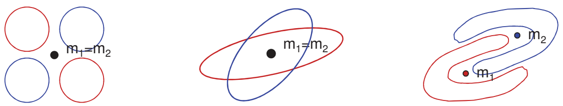
- **The optimal projection axes are the solutions of a generalized eigenvector problem :**

$$\boxed{\hat{W} = \left[\hat{w}_1\middle|\hat{w}_2\middle|\ldots\middle|\hat{w}_{c-1}\right] = \underset{w}{\text{argmax}}\frac{\left|W^T S_{\text{inter}} W\right|}{\left|W^T S_{\text{intra}} W\right|} \Rightarrow (S_{\text{inter}} - \lambda_i S_{\text{intra}})\hat{w}_i = 0}$$

- **Remarks:**
  - S_{inter} is the sum of C matrices of rank 1 or 0 related by the definition of m. It is thus at most of rank C-1. This means that only C-1eigenvalues are nonzero.
  - LDA provides at most C-1 projection axes [Fukunaga90]
  - They correspond to the largest eigenvalues of $S^{-1}_{\text{intra}}S_{\text{inter}}$ (provided S_{intra} is nonsingular).
  - The larger the eigenvalue, the higher the separability along the direction

Linear extraction methods: summary

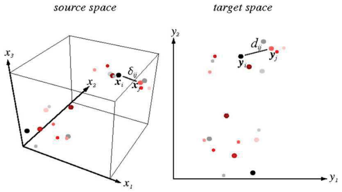
- **PCA and LDA:**
  - Are widely used in practical applications
  - Are not suited to complicated distributions (e.g. multi-modal, non Gaussian)…
  - E.g. cases where LDA fails (following [Gutierrez])



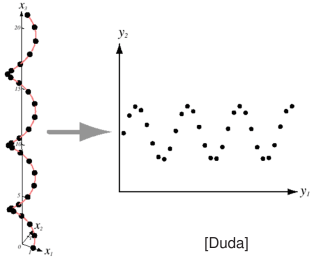
- **Other feature extraction techniques (quite numerous)**
  - Linear: Independent Component Analysis (ICA), Nonnegative Matrix Factorization (NMF), Multi-Dimensional Scaling (**MDS**)
  - Non-linear: principal curves and surfaces, Kernel PCA, auto-encoders, **ISOMAP**, Linear Local Embedding (LLE), t-Distributed Stochastic Neighbor Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), *etc.* (a.k.a. *manifold learning* methods)

Multi-Dimensional Scaling *(MDS)*

- **Input: dissimilarity matrix**
  - Distance between samples in source space: δ_{ij}
- **Output: positions**
  - low-dimensional target space (*d*=2 or 3), d_{ij}
- **Property**
  - Inter-sample distances preserved in target space
- **Optimization problem**
  - Minimize quadratic differences between δ_{ij}'s and d_{ij}'s
- **Unique up to global transformation**



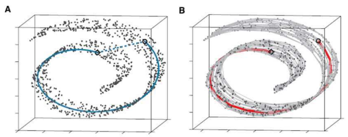
E.g. MDS from 3D to 2D



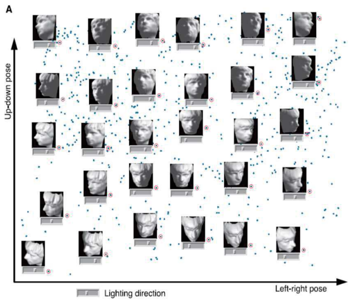
[Duda]

ISOMAP [Tenenbaum, Science, 2000]

- **Embedding manifold**
  - May be non-linear
- **ISOMAP =**
  - Similarity computation on manifold: geodesic distance
  - Application of MDS
- **Better preservation of perceptive similarity between samples**

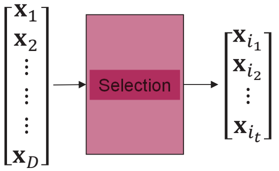


A Distance Géométrique – B distance géodésique



Feature selection methods

- ***Feature Subset Selection***
- **Basis idea:**
  - Define an **objective criterion** to evaluate the optimality of a feature subset
  - Identify such characteristics by optimizing this criterion using the training set.



- **This seems rather simple !!! Unfortunately,**
  - Selecting t features among D involves

$$C_D^t = \binom{D}{t} = \frac{D!}{t!\,(D-t)!}$$

possibilities… E.g. 10 features among 25 = 3 268 760 possibilities !

⇨ Define a **research strategy**

Feature selection methods

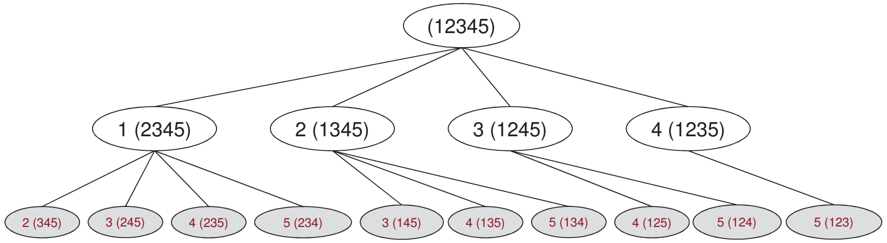
- **Possible optimality criteria:**
  - Misclassification rate over a test set (must be different from training data!)
  - Distance or separability measures:
    - ✓ Euclidean distance, Mahalanobis distance, etc...
    - ✓ Criteria based on dispersion matrices (e.g. LDA criterion)
    - ✓ Probabilistic (distance between probability densities)
  - Correlation measures:
    - ✓ Features should be correlated with class labels, but not with each others.
    - ✓ Mutual Information measures between class labels and feature vectors.
  - And more… (see [Webb, Duda, Theodoridis] for example).
- **Optimization strategies**
  - Exhaustive search: optimal, but costly
  - Tree search (e.g. "*Branch and Bound*" and variants)

Branch and bound (1/2) *(following [Webb])*

- **Goal: <u>maximize</u> a <u>monotonic</u> function to find the optimal partition, avoiding to explore all possibilities !**

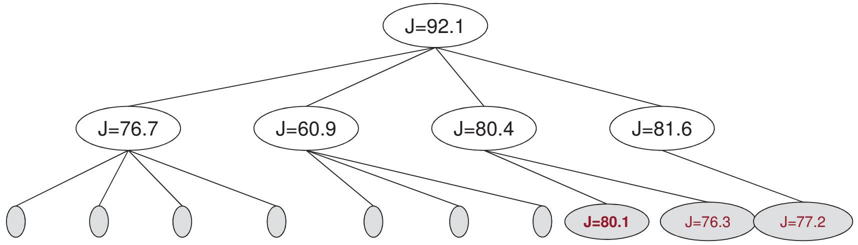
$$X \subset Y \Rightarrow J(X) < J(Y)$$

- **E.g. partition of 3 features among 5:** $C_5^3 = \begin{pmatrix} 5 \\ 3 \end{pmatrix} = \frac{5!}{3!(5-2)!} = 10$ possibilities
- **We may arrange possible partitions into a 10-leafs tree**



Branch and bound (2/2)

- **Start from the rightmost leaf (note that the figure gives examples of J values) → J=77.2 becomes the current maximum**
- **Go back to the root and explore next branch → the current maximum becomes 80.1**
- **It we try to explore the next branch, we find J=60.9 at the first node. It is useless to continue because, by construction, the leafs will have smaller values. Same case for the last branch…**
- **In this example, we found the maximum in 8 evaluations of J instead of 10.**



Optimization strategies

- **Sequential algorithms**
  - Sequential Forward Selection (SFS)
    - ✓ Starts from empty feature subset : $Y_0 = \{Ø\}$
    - ✓ Sequentially adds the feature x* which maximizes $J(Y_k + x^*)$
    - ✓ Drawback: cannot discard features becoming obsolete due to the addition of other features...
  - Sequential backward selection (SBS)
    - ✓ See SFS, but starting from the full set and discarding the feature that results in the smallest decrease of $J(Y_k - x^*)$
  - Plus L - minus R :
    - ✓ Combines SFS and SBS: at each step, discards R features and adds L features
    - ✓ Problem: no theoretical result for choosing L and R...
- **Stochastic algorithms: escape from local minima !**
  - ✓ Random generation of a feature sample, then SFS, then SBS
  - ✓ Simulated annealing, genetic algorithms, etc.

Take-away remarks

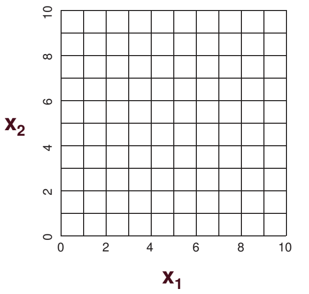
- **To overcome the curse of dimensionality / ease visualization**
  - Feature selection or extraction
- **Selection:**
  - Given an optimality criterion (e.g. class separability)
  - Choice of features = combinatory optimization problem
- **Extraction**
  - Given a criterion that measures the:
    - ✓ Goodness of data representation (e.g. PCA)
    - ✓ Separability of classes (ex : LDA)
  - Optimization leads to a process of the form:
    - ✓ Linear transformation of data
    - ✓ Truncation
  - Other linear methods: ICA, NMF, MDS
  - Nonlinear methods, ISOMAP, LLE, auto-encoders, t-SNE, UMAP are better suited to complex variability.

Exercise

Feature selection and extraction

Homework

- **Consider the following training sample:**
  - X = (x_1,x_2) = { (1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8) }
  - Draw the corresponding scatter plot
  - Calculate the sample mean and (biased) covariance matrix.
  - Calculate PCA and plot the eigenvectors v_1 and v_2
    - ✓ Trick: to calculate the first eigenvector easier, suppose that one of its coordinates is 1, calculate the other one, then normalize the resulting vector to 1.



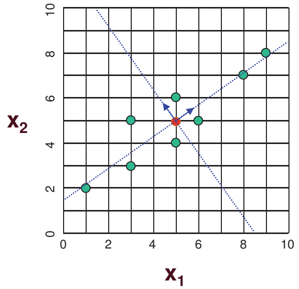
From [Gutierrez]

Solution of exercise 1

- **X = (x_1,x_2) = { (1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8) }**

x_1	x_2	$x_1-\mu_1$	$x_2-\mu_2$	$(x_1-\mu_1)^2$	$(x_1-\mu_1)(x_2-\mu_2)$	$(x_2-\mu_2)^2$
1	2	-4	-3	16	12	9
3	3	-2	-2	4	4	4
3	5	-2	0	4	0	0
5	4	0	-1	0	0	1
5	6	0	1	0	0	1
6	5	1	0	1	0	0
8	7	3	2	9	6	4
9	8	4	3	16	12	9
5	5			6,25	4,25	3,5

$$\Rightarrow \mu = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \text{and} \Sigma = \begin{bmatrix} 6{,}25 & 4{,}25 \\ 4{,}25 & 3{,}5 \end{bmatrix}$$



$$\begin{vmatrix} 6{,}25-\lambda & 4{,}25 \\ 4{,}25 & 3{,}5-\lambda \end{vmatrix} = 0 \quad \Rightarrow \quad \begin{cases} \lambda_1 \approx 9{,}34 \\ \lambda_2 \approx 0{,}41 \end{cases}$$

$$\begin{bmatrix} 6{,}25 & 4{,}25 \\ 4{,}25 & 3{,}5 \end{bmatrix} . \begin{bmatrix} 1 \\ v \end{bmatrix} = 9{,}34 . \begin{bmatrix} 1 \\ v \end{bmatrix} \xrightarrow{\text{(normalization)}} v_1 \approx \begin{pmatrix} 0{,}81 \\ 0{,}59 \end{pmatrix}, \text{ then } v_2 \perp v_1 \approx \begin{pmatrix} -0{,}59 \\ 0{,}81 \end{pmatrix}$$

[Gutierrez]

Unsupervised learning and clustering

Parametric methods: mixture distributions – EM algorithm

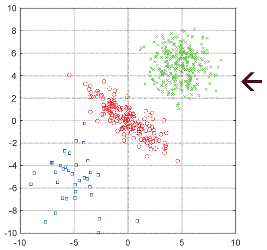
Non-parametric methods: k-means algorithms, mode-seeking methods,

Flat *vs.* hierarchical algorithms

Summary of previous episodes

- **The classification methods that we have been studying so far are supervised (i.e. training samples are labeled)**
  - When class-conditional distributions and prior probabilities are known, optimal classifiers can be designed in the Bayesian framework.
  - When only the functional forms of the pdf's are known, their parameters may be learned from a set of labeled training samples.
  - The pdf's may also be approximated in a non-parametric way from a set of labeled learning samples.
  - Dimensionality reduction techniques (feature selection or extraction) also use labeled samples.
- **The situation is often far from being so ideal!**
- **What can be done when class labels are not known ?**

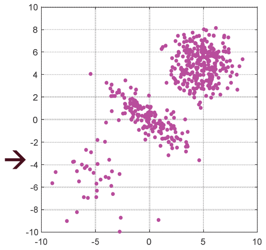
Unsupervised classification (a.k.a. clustering)



← Supervised

vs.

Unsupervised →



Following [Bontempi]

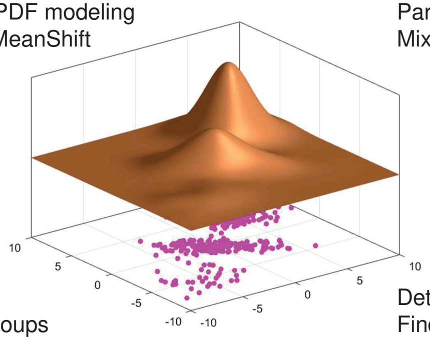
- **Goal**
  - Given a set of feature vectors $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ without class labels, capture the *structure* of the data
- **Such a situation often occurs**
  - Labeling a training set may be awkward.
  - Class labels are not always known in advance (e.g. *data mining*).
- **NB: *Clustering* methods may also be used in a supervised context, for modeling multi-modal distributions.**

Clustering: deterministic vs. probabilistic

- **"Hundreds of clustering algorithms have been proposed in the literature" [Jain2000]**

Probabilistic
Nonparametric PDF modeling
Hill Climbing = MeanShift

Probabilistic
Parametric PDF modeling
Mixture Models + EM

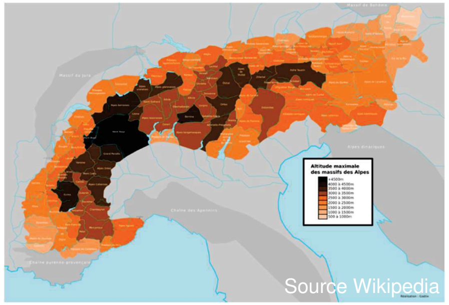


Deterministic
Find compact groups
K-Means

Deterministic
Find dense regions
DBSCAN / HDBSCAN

Clustering: partitional vs. hierarchical

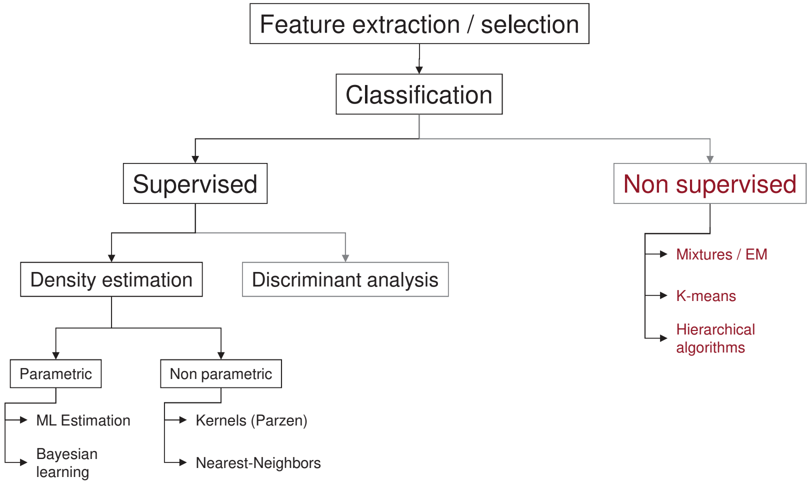
- **Partitional (flat) approaches**
  - Aim at partitioning feature space by optimizing a given functional



Source Wikipedia

- **Hierarchical approaches**
  - Organize the data into a series of nested groups that may be visualized as a tree-like representation

A hierarchy of methods



Feature extraction / selection

Classification

Supervised

Non supervised

Density estimation

Discriminant analysis

Parametric

Non parametric

ML Estimation

Bayesian learning

Kernels (Parzen)

Nearest-Neighbors

Mixtures / EM

K-means

Hierarchical algorithms

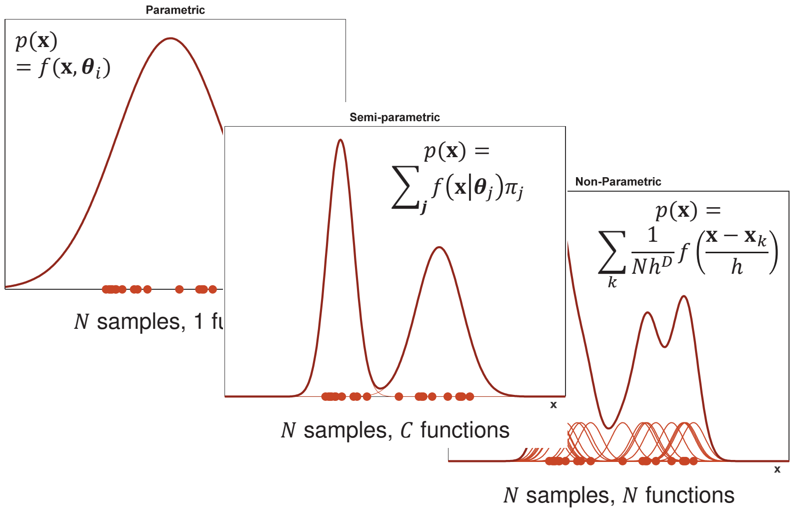
Parametric methods – Mixture Models

- **They can model either**
  - A unique, complex (e.g. multimodal) probability density function
  - **A set of pdf's, each one being associated with one of the classes in the training set**
- **Assumptions**
  - The number of classes is supposed to be known
  - The functional form of each class is known
  - Each class may be of a different type
  - Popular: **Gaussian Mixture Models** (GMM)
- **The PDF of features is a mixture of parametric densities**

$$p(\mathbf{x}|\mathbf{\Theta}) = \sum_{j=1}^{C} p(\mathbf{x}|\omega_{\mathrm{j}}, \mathbf{\theta}_{\mathrm{j}})\pi_j$$

  - Note: such approaches are sometimes dubbed *semi-parametric.*

PDF estimator families



Mixture models: the unknown parameters

- **The mixing parameters, π_j, can be viewed as prior class probabilities**

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{C} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

- **The vector of unknown parameters, $\boldsymbol{\theta}$, that concatenates:**
  - The parameters of each component $\boldsymbol{\theta}_j$ (e.g. for a Gaussian mixture: $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$)
  - The mixing parameters $\pi_j = P(\omega_j)$ with $\sum_{j=1}^{C} P(\omega_j) = 1$

  must be estimated from unlabeled samples $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

- **We first form the likelihood:**

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{k=1}^{N} \sum_{j=1}^{C} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

Mixture models: Maximum Likelihood estimation

- **It may be shown (see [DUDA], pp. 518-528) that the maximum likelihood solution:**

$$L(\Theta) = \ln[p(X|\Theta)] = \sum_{k=1}^{N} \ln\left[\sum_{j=1}^{C} p(x_k|\omega_j, \theta_j).P(\omega_j)\right]$$

satisfies in the Gaussian case, $\Sigma_j = \sigma^2_j.\mathbf{I}_d$ (in d dimensions):

$$\hat{\mu}_j = \frac{\sum_k P(\omega_j|x_k,\theta) x_k}{\sum_k P(\omega_j|x_k,\theta)} \qquad \hat{\sigma}_j^2 = \frac{\sum_k P(\omega_j|x_k,\theta)\|x_k - \hat{\mu}_j\|^2}{d\sum_k P(\omega_j|x_k,\theta)}$$

for mixing parameters (under unit sum condition $\sum_j \hat{P}(\omega_j) = 1$)

$$\hat{P}(\omega_j) = \frac{1}{N} \sum_k P(\omega_j|x_k,\theta)$$

Interpretation

- **The weights: w_{kj}=P(ω_j|x_k,θ) can be seen as an estimation of the chance that x_k belongs to the j-th class.**
  - NB: $\sum_j P(\omega_j|x_k,\theta) = 1$

- **Should P(ω_j|x_k,θ) be <u>binary</u> (⇔ one-hot encoding), then**

$$\sum_k P(\omega_j|x_k,\theta) = \sum_{x_k \in \omega_j} 1 = N_j \qquad \sum_k P(\omega_j|x_k,\theta) x_k = \sum_{x_k \in \omega_j} x_k$$

  - The ML equations would become:

$$\hat{\mu}_j = \frac{1}{N_j} \sum_{x_k \in \omega_j} x_k \qquad \hat{\sigma}_j^2 = \frac{1}{N_j d} \sum_{x_k \in \omega_j} \|x_k - \hat{\mu}_j\|^2 \qquad \hat{P}(\omega_j) = \frac{N_j}{N}$$

  - And we would get the usual estimators!

- **Most of the time, however, P(ω_j|x_k,θ) ∊ [0,1]**
  - All samples play a role in the estimation of all class parameters...

Maximum Likelihood estimation: difficulty

- **Moreover, in the ML equations:**

$$P(\omega_j|x_k,\theta) = \frac{\overbrace{p(x_k|\omega_j,\theta)}^{\frac{1}{(2\pi\hat{\sigma}_j^2)^{d/2}} \exp\left(-\frac{\|x_k - \hat{\mu}_j\|^2}{2\hat{\sigma}_j^2}\right)}.\hat{P}(\omega_j)}{\sum_c p(x_k|\omega_c,\theta).\hat{P}(\omega_c)}$$

- **The ML equations are coupled… So?**
- **Idea: use a fixed-point algorithm**
  - Take an initial parameter set
  - Use at each iteration the ML equations, inserting in their rightmost term the values found at the previous iteration…

- **The <u>EM</u> framework is a principled way of devising such an algorithm, and of proving its convergence towards a (local) maximum of the log-likelihood.**

The EM algorithm [Dempster77]

- **EM means Expectation - Maximization**
- **Iterative algorithm for parameter estimation, in the sense of Maximum Likelihood, from missing or hidden data.**
- **Missing (or corrupted) data:**
  - When the observation process is limited or corrupted.
- **Hidden data:**
  - Their knowledge would lead to a simpler form of likelihood
- **Applications:**
  - Tomographic reconstruction, mixture models…

The EM formalism

- **Notations:**
  - X denotes the incomplete, observed data
  - Z denotes the missing, unknown data.
  - (X,Z) are the complete data
  - Θ, is the parameter vector, which concatenates θ, the parameters of the components and P(ω_j), the mixing parameters.
- **We may form the likelihood of the complete data: *p(X,Z|Θ)***
- **In contrast, *p(X|Θ)*, is called likelihood of the incomplete data**
- **Maximizing L(Θ)=log[p(X|Θ)] is difficult…**
  - Maximizing log[p(X,Z|Θ)] may be easier, for a particular choice of Z.
  - But Z is unobservable… ➔ we shall try to eliminate it!

EM: basis ideas (1/2)

- **Link between the likelihood of the complete data and the likelihood of the incomplete data**
  - According to the definition of conditional probabilities, we have:

$$p(X|\Theta) = p(X,Z|\Theta)/p(Z|X,\Theta)$$

- **Taking logs, we have:**

$$L(\Theta) = \log[p(X|\Theta)] = \log[p(X,Z|\Theta)] - \log[p(Z|X,\Theta)]$$

- **1^{rst} idea: if we average this expression with respect to Z,**
  - The leftmost term (log-likelihood) does not change…
  - And we eliminate Z from the rightmost term!

EM: basis ideas (2/2)

- **In an iterative process, we know the current estimate, Θ^n**
  - The average (expectation) is taken with respect to Z, knowing X and Θ^n
- **We obtain:**

$$L(\Theta) = E_Z\left[\log p(X,Z|\Theta)\middle| X,\Theta^n\right] - E_Z\left[\log p(Z|X,\Theta)\middle| X,\Theta^n\right]$$

…which may be written as:

$$\boxed{L(\Theta) = \log[p(X|\Theta)] \equiv Q(\Theta|\Theta^n) - H(\Theta|\Theta^n)}$$

- **2nd idea: take advantage of a “good property” of the expectation to get rid of H!**

A “good property” of the expectation

- It can be shown [Lange2000] that

$$H(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n) - H(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n) \geq 0$$

- In the EM, Θ^{n+1} is chosen to maximize $Q(\boldsymbol{\Theta}|\boldsymbol{\Theta}^n)$
  - So, in particular:

$$Q(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n) - Q(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n) \geq 0$$

- Summing up:

$$\begin{array}{rc} & Q(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n) - Q(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n) \geq 0 \\ + & -H(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n) - \{-H(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n)\} \geq 0 \\ = & L(\boldsymbol{\Theta}^{n+1}) - L(\boldsymbol{\Theta}^n) \geq 0 \end{array}$$

➔ which shows that each iteration increases the log-likelihood

The iteration of the EM algorithm

- The (n+1)-th iteration is made of 2 steps
- E (expectation) step: form the conditional expectation

$$Q\left(\Theta\middle|\Theta^n\right) = E_z\left[\log p\left(X,Z\middle|\Theta\right)\middle|X,\Theta^n\right] = \int_z \log p\left(X,z\middle|\Theta\right) p\left(z\middle|X,\Theta^n\right) dz$$

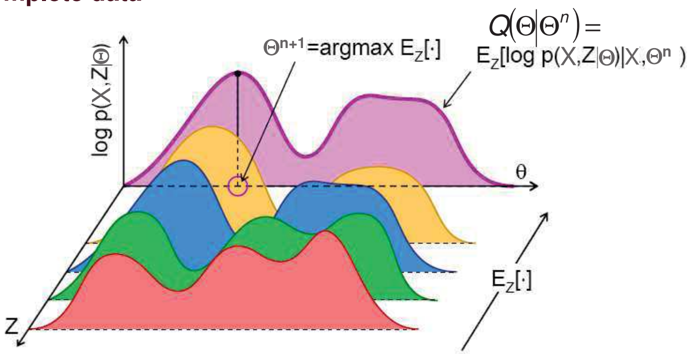
- M (maximization) step: find Θ, the maximizer of Q(Θ| Θⁿ)

$$\Theta^{n+1} = \arg\max_{\Theta} Q\left(\Theta\middle|\Theta^n\right)$$

  - Since the M-step always increases L(Θ)=log[p(x|Θ)], the algorithm converges towards a stationary point (which may not be the global maximum!)
  - In many cases, the E-step is performed analytically and grouped with the M-step (the iteration is implemented as a single closed-form expression)

The EM algorithm: interpretation

- During the E-step, we eliminate the unobserved data Z by integration (marginalization)
- During the M-step, we maximize the average likelihood of the complete data



From [Gutierrez]

The case of mixture models

- The hidden variable $\mathrm{Z} = \{\mathbf{z}_k\}$ for k=1…N, indicates which component (class) x_k has been generated from
  - Each $\mathbf{z}_k$ is a vector with C components, z_{kj}
  - z_{kj} is 1 if x_k belongs to class ω_j and 0 otherwise
  - z_{kj} is a Bernoulli variable with parameter P=probability that $\mathbf{x}_k$ belongs to class ω_j (knowing the parameters), i.e. $P(\omega_j|\mathbf{x}_k, \Theta)$
  - Since the expectation of a Bernoulli law with parameter P is … P, we have

$$E\left(z_{kj}\middle|\mathbf{x}_k, \Theta\right) = P\left(\omega_j\middle|\mathbf{x}_k, \Theta\right) \overset{\text{def}}{=} w_{kj}$$

  - This expression may be evaluated using Bayes’ theorem

$$w_{kj} = P\left(\omega_j\middle|\mathbf{x}_k, \Theta\right) = \frac{P\left(\mathbf{x}_k\middle|\,\omega_j, \Theta\right)P\left(\omega_j\right)}{\sum_c P\left(\mathbf{x}_k\middle|\,\omega_c, \Theta\right)P\left(\omega_c\right)}$$

- Let us note these results and form the complete likelihood…

Mixture models: the E-step

- **The likelihood of the complete data (x_k,z_k) is written as**

$$p(x_k, z_k|\Theta) = p(x_k|z_k,\Theta).P(z_k|\Theta) = p(x_k|\theta_j).P(\omega_j) = \prod_{j=1}^{C}\left[p(x_k|\theta_j).P(\omega_j)\right]^{z_{kj}}$$

- **For N independent observations:**

$$p(X,Z|\Theta) = \prod_{k=1}^{N}\prod_{j=1}^{C}\left[p(x_k|\theta_j)P(\omega_j)\right]^{z_{kj}} \longrightarrow \log(p(X,Z|\Theta)) = \sum_{k=1}^{N}\sum_{j=1}^{C} z_{kj}\left[\log(p(x_k|\theta_j)) + \log(P(\omega_j))\right]$$

- **Now, we take the expectation with respect to z**
  - Recalling the results of the previous slide, it comes that:

$$\boxed{Q(\Theta|\Theta^n) = \sum_{j=1}^{C}\sum_{k=1}^{N}\log(P(\omega_j)).w_{kj}^n + \sum_{j=1}^{C}\sum_{k=1}^{N}\log(p(x_k|\theta_j)).w_{kj}^n} \qquad w_{kj}^n = P(\omega_j|x_k,\Theta^n)$$

Gaussian mixtures: the M-step

- **Canceling the derivative of Q, with respect to each parameter, one obtains in the Gaussian case**

$$w_{kj}^n = P(\omega_j|x_k,\Theta^n) = \frac{\boxed{p(x_k|\omega_j,\theta^n)}.P^n(\omega_j)}{\sum_c p(x_k|\omega_c,\theta^n).P^n(\omega_c)}$$

$$\frac{1}{(2\pi)^{d/2}\left|\Sigma_j^n\right|^{1/2}}\exp\left(-\frac{(x_k-\mu_j^n)(\Sigma_j^n)^{-1}(x_k-\mu_j^n)}{2}\right)$$

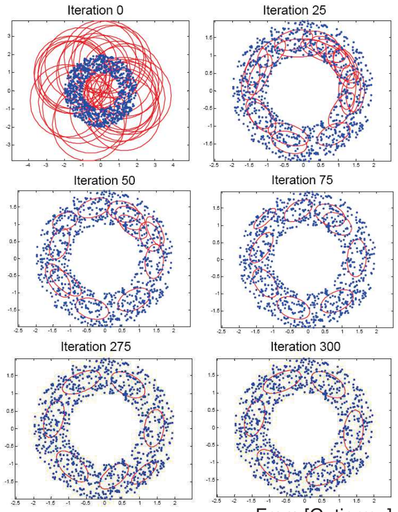
$$\boxed{\begin{aligned}\mu_j^{n+1} &= \frac{\sum_k w_{kj}^n x_k}{\sum_k w_{kj}^n}\\ \Sigma_j^{n+1} &= \frac{\sum_k w_{kj}^n.(x_k-\mu_j^{n+1})(x_k-\mu_j^{n+1})^T}{\sum_k w_{kj}^n}\\ P^{n+1}(\omega_j) &= \frac{1}{N}\sum_k w_{kj}^n\end{aligned}}$$

- **Note that**
  - We obtain weighted expressions for the mean and covariance matrix
  - The implementation is rather simple, but the convergence is slow
  - The closer x_k from the class mean, the higher its weight

Example (modeling a single, complex pdf)

- **Training data:**
  900 points drawn from an annular distribution

- **EM algorithm:**
  - 30 initial classes with random means
  - Diagonal initial covariance matrices with large variance
  - Regularization of covariance matrices
  - Components with small P(ω_j) are discarded
  - The final model is compact : only a few components

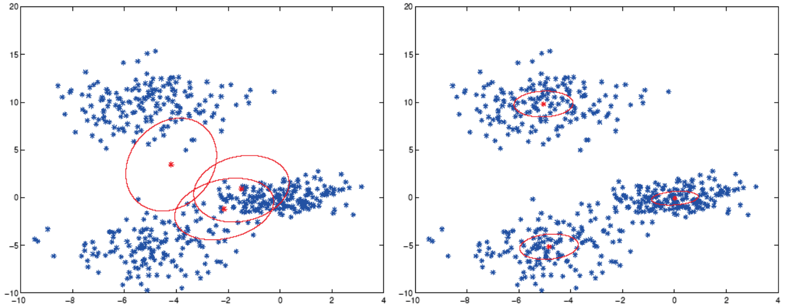


From [Gutierrez]

Unsupervised classification for 3 classes

- **Simulated data: mixture of 3 bidimensional Gaussian classes.**

- **Initialization (left), solution after 50 iterations (right)**

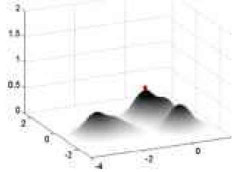


From [Bontempi]

EM clustering and classification

- **Classification method**
  - Train:
    - ✓ Specify the number of classes and the parametric form of their pdf's
    - ✓ Estimate the parameters of the mixture from training data (using EM)
  - Classify new data, x:
    - ✓ Form posterior probabilities P(ωj|x) and chose the class with highest probability.
- **Issues**
  - Choosing C
    - ✓ Minimizing a criterion with respect to C : log-likelihood + penalty term (penalize complex models i.e. large C values), k=nb parameters
      - – *Akaike Information Criterion:* $AIC = 2L(\hat{\boldsymbol{\theta}}) + 2k$
      - – *Bayesian Information Criterion:* $BIC = 2L(\hat{\boldsymbol{\theta}}) + k \ln N$
    - ✓ **See EMbic.m demo**
    - ✓ Cross-validation, etc.
  - The EM may be trapped in local maxima, and it is rather slow
    - ✓ Variants: GEM (Generalized EM), CEM (Classification EM), SEM (Stochastic EM)

Mode seeking methods

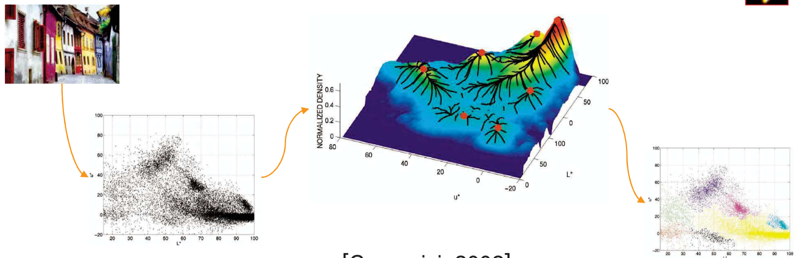


- **Assumptions**
  - In the overall PDF p(x), each cluster corresponds to a mode.
  - 2D example: "hills" separated by "valleys"
- **Principle**
  - Seek the modes using e.g. gradient ascent algorithm
  - Example: *Meanshift* [Comaniciu99] …after [Fukunaga 75] !
    - ✓ $p(x)$ is modeled using Parzen windows : $p_K(\mathbf{x}) \propto \sum_i k(\|(\mathbf{x}-\mathbf{x}_i)/h\|^2)$
    - ✓ *Meanshift* is a hill-climbing method adapted to such a representation
- **Gradient ascent and the Meanshift**
  - $\nabla p_K \propto \sum_i (\mathbf{x}-\mathbf{x}_i) k'(\|(\mathbf{x}-\mathbf{x}_i)/h\|^2)$
  - The (normalized) gradient direction, provided that $g(x) = -k'(x)$, is given by the meanshift

$$\boxed{\frac{\nabla p_K}{p_G} \propto \frac{\sum_i x_i\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}}$$

The Meanshift

- **Gradient ascent:** $\mathbf{x} \leftarrow \mathbf{x} + \nabla p_K / P_G$,
- **Algorithm**
  - Starting from each data point, iterate : $\mathbf{x} \leftarrow \frac{\sum_i x_i\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}$ until convergence
  - Group points that converge to the same maximum into a cluster



[Comaniciu2002]

Deterministic methods / K-means clustering

- **Does not involve pdf estimation**
- **Attempts to find natural grouping (clusters) in a data set by optimizing an objective function**
- **Find an assignment of data points to clusters, and a set of prototypes that represent the centers of the clusters, in order to minimize**

$$J_e = \sum_{j=1}^{C} \sum_{\mathbf{x}\in\omega_j} \left\|\mathbf{x} - \mathbf{m}_j\right\|^2$$

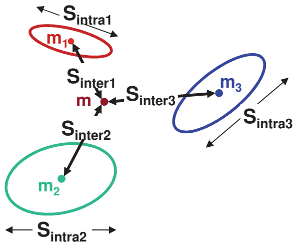
Remark

- **Dispersion matrices (see FDA)**

$$S_{\text{intra}} = \sum_{j=1}^{C} \sum_{\mathbf{x}\in\omega_j} (\mathbf{x}-\mathbf{m}_j)(\mathbf{x}-\mathbf{m}_j)^T$$

$$S_{\text{inter}} = \sum_{j=1}^{C} N_j (\mathbf{m}_j-\mathbf{m})(\mathbf{m}_j-\mathbf{m})^T$$

$$S_{\text{total}} = \sum_{j=1}^{C} \sum_{\mathbf{x}\in\omega_j} (\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T = S_{\text{intra}} + S_{\text{inter}}$$



- **Taking the trace, we obtain (Huygens theorem)**

$$J_{\text{total}} = \sum_{j=1}^{C} \sum_{\mathbf{x}\in\omega_j} \|\mathbf{x}-\mathbf{m}\|^2 = \sum_{j=1}^{C} \sum_{\mathbf{x}\in\omega_j} \|\mathbf{x}-\mathbf{m}_j\|^2 + \sum_{j=1}^{C} N_j \|\mathbf{m}_j-\mathbf{m}\|^2 = J_e + J_{\text{inter}}$$

  - Since the total inertia, J_{total}, is constant for a given data set and a constant C, minimizing J_e amounts to maximizing the inter-class variance J_{inter}

➔ Finding compact clusters ⇔ finding distinct clusters

K-means: basis idea [Lloyd 1957-1982, McQueen 1967]

- **Let us introduce binary variables to formalize the assignment of data points to clusters (one-hot-encoding)**

$$u_{kj} \in \{0,1\} \text{ and } \sum_{j=1}^{C} u_{kj} = 1$$

$$J_e(u,m) = \sum_{j=1}^{C} \sum_{k=1}^{N} \boxed{u_{kj}} \|x_k - m_j\|^2$$

- **Fix the *m_j's* and minimize with respect to *u***
  - For each x_k, only one u_{kj} may be non null (due to unit sum condition).
  - Clearly, in order to minimize J_e, we shall choose (for each k):

$$u_{kj} = \begin{cases} 1 & \text{if} \quad \|x_k - m_j\|^2 < \|x_k - m_i\|^2 \ \forall i \\ 0 & \text{otherwise} \end{cases}$$

K-means: basis idea [Lloyd 1957-1982, McQueen 1967]

- **Assignement rule = nearest-mean rule**
  - Optimal for Gaussian clusters with equal prior and same, isotropic, covariance matrix

- **Now, fix *u* and minimize w.r.t. the *m_j's***

$$J_e(\mathbf{u},\mathbf{m}) = \sum_{j=1}^{C} \sum_{k=1}^{N} \mathbf{u}_{\mathrm{kj}} \|\mathbf{x}_{\mathrm{k}} - \mathbf{m}_{\mathrm{j}}\|^2$$

$$0 = \sum_{k=1}^{N} \mathbf{u}_{\mathrm{kj}} \frac{\partial \|\mathbf{x}_{\mathrm{k}} - \mathbf{m}_{\mathrm{j}}\|^2}{\partial \mathbf{m}_{\mathrm{j}}} \quad \forall j = 1 \ldots C \;\longrightarrow\; \sum_{k=1}^{N} u_{kj}\mathbf{x}_k = \sum_{k=1}^{N} u_{kj}\mathbf{m}_j = N_j \mathbf{m}_j \;\longrightarrow\; \mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x}\in\omega_j} \mathbf{x}$$

➔ Optimal class centers are class means

The k-means algorithm

- **Batch k-means algorithm**
  - Initialize class centers
    - ✓ At random, or simply take the first C samples, for example…
  - Repeat
    - ✓ Classify the n samples according to their distance to class centers
      - ⇨ **if $\|\mathbf{x}_k - \mathbf{m}_j\| < \|\mathbf{x}_k - \mathbf{m}_i\| \ \forall i \neq j$, then $\mathbf{x}_k$ is assigned to ω_j**
    - ✓ Update the positions of class centers
      - ⇨ $m_j = \frac{1}{N_i} \sum_{x\in\omega_i} x \quad \forall j = 1 \ldots C$
  - Until the partition does not change

Remarks (1)

- **The K-means algorithm is rather fast O(NdCT)**
  - where N is the number of samples, d the dimensionality of feature space, C the number of classes and T the number of iterations (in general, T<< N).
- **The K-means algorithm converges to a local minimum of J_e, which depends on the initial choice of class centers.**
  - One may run the algorithm several times, with different initializations and retain the best result (e.g. the one that yields the smallest J_e final value)
- ***Dynamic clouds* generalize k-means. They involve:**
  - A function that allows computing the prototype for each cluster
  - A function that allows assigning each point to one class
  - The general form of the algorithm remains the same…

Remarks (2)

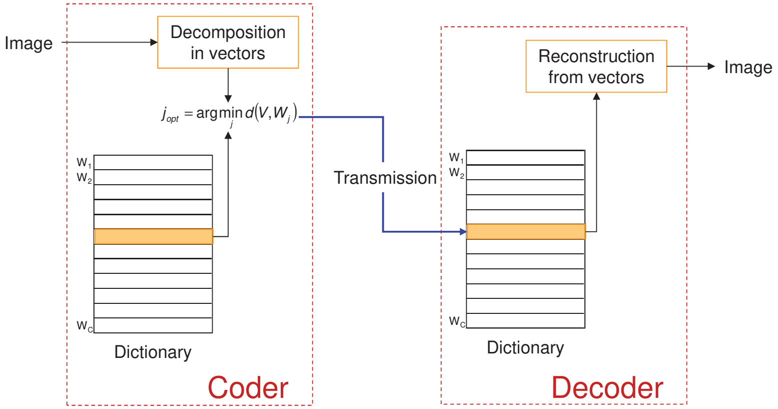
- **There exists a sequential version of the k-means algorithm [Duda, p. 549]**
  - Initialize class centers
  - Repeat
    - ✓ Pick one sample at random, x*
    - ✓ Assign x* to a new class according to its distance to class centers
    - ✓ Update the positions of the centers of the old and new class for x* and the value of the overall error, J_e
  - Until J_e does not change in n attempts
- **This version is (even) less robust with respect to local minima…**

The ISODATA algorithm [Ball & Hall, 1965]

- **ISODATA is a more sophisticated version of k-means**
  - K-means are sometimes called “basic ISODATA”
  - Supplementary mechanisms allow the automatic selection of the number of clusters, C
- **Algorithm:**
  - Repeat
    - ✓ Perform clustering using k-means
    - ✓ Discard clusters smaller than a certain threshold
    - ✓ Split highly dispersed clusters
    - ✓ Merge sufficiently close clusters
  - Until stabilization
- **Thresholds must be set, but the algorithm is more automatic**

K-means and vector quantization

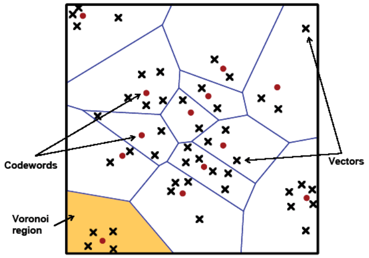
- **Vector quantization**



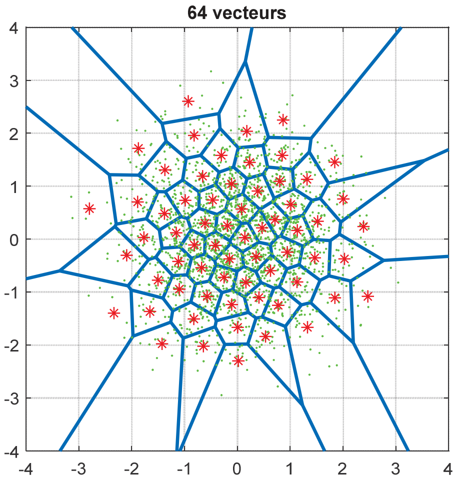
K-means and vector quantization

- **Construction of the dictionary: minimization of total distortion**
  - Most often, distortion = quadratic error (i.e. k-means criterion)

- **This is a clustering problem!**
  - A famous algorithm (called LBG) was proposed in [Linde, Buzo & Gray 1980]
  - This is essentially a version of k-means with a hierarchical initialization of class centers (codewords)
    - ✓ Start with 1 class
    - ✓ Split the codeword
    - ✓ Classify with 2 classes, and so on…



LBG at work



K-means and EM

- **There is a close similarity between them. They both**
  - Compute a class membership function
    - ✓ k-means algorithm performs a *hard* assignment (u_{kj})
    - ✓ EM performs a *soft* assignment (w_{kj})
  - Use it to weight the computation of the mean (EM: also dispersion & priors)

- **The variable u_{kj} plays a role similar to the one of w_{kj} in the EM**
  - Some authors (e.g. Duda) denote them both as $P(\omega_j|\mathbf{x}_k, \Theta)$

- **k-means may be derived as a limit case of EM for Gaussian mixtures with diagonal, isotropic covariance matrices [Bishop]**
  - The variance is fixed and the same for all classes.
  - Consider the limit where the variance goes to 0
    - ✓ The EM weights w_{kj} go to 0 except for the closest class, for which it goes to 1.
    - ✓ One obtains the same expression for the mean

Fuzzy C-means [DUNN,1973]

- **In the K-means algorithm**
  - The class membership coefficient is binary

- **In the fuzzy C-means,** $u_{kj} \in [0,1]$ and $\sum_{j=1}^{C} u_{kj} = 1$
  - i.e. one sample has a graded membership to all clusters

- **One minimizes a heuristic global cost function**

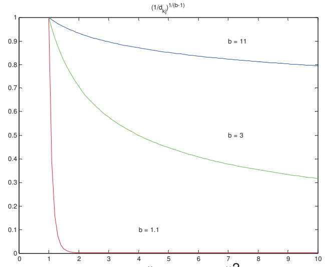
$$J_{fuzzy} = \sum_{j=1}^{C}\sum_{k=1}^{N} u_{kj}^{\boldsymbol{b}} \left\| x_k - m_j \right\|^2$$

- **The exponent, b tunes the "fuzziness" of the algorithm, b > 1**

Fuzzy C-means: algorithm

- **Same structure as the K-means**
- **Fix class centers and minimize w.r.t. u**
  - Use a Lagrangian to incorporate unit sum condition
  - Canceling derivatives:

$$u_{kj}^{b} = \frac{\left(1/d_{kj}\right)^{1/(b-1)}}{\sum_{c=1}^{C}\left(1/d_{kc}\right)^{1/(b-1)}}$$



$$d_{kj} = \left\| x_k - m_j \right\|^2$$

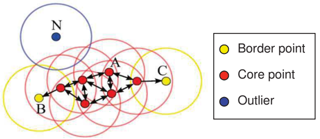
- **Fix u and update centers**
  - Canceling derivatives:

$$m_j = \frac{\sum_{k=1}^{N} u_{kj}^{b} x_k}{\sum_{k=1}^{N} u_{kj}^{b}}$$

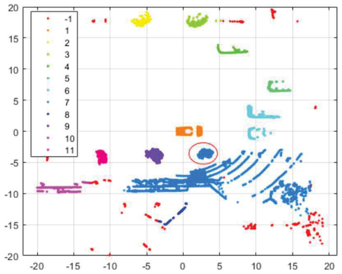
- **Convergence may be better than for the K-means...**

Density-based clustering

- **Clusters: dense regions separated by regions of lower density**
- **DBSCAN (Ester et al, 1996)**
  - Neighborhood distance, ϵ
  - Minimal neighbor number, minPoint
  - Repeat
    - ✓ Find core point and expand cluster
  - Until all points left are isolated (outliers)
- **Pros / cons**
  - ☺ Number of clusters not needed
  - ☺ Captures arbitrarily shaped clusters
  - ☺ Handles noisy datasets / outliers
  - Sensitive to dimensionality
  - Difficulties with varying densities



DBSCAN cluster model
[Schubert2020]



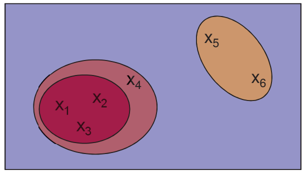
LIDAR data analysis using DBSCAN
© Matlab 2020

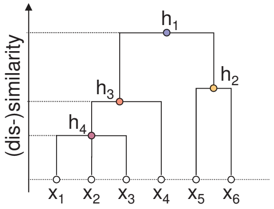
Fast clustering techniques

- **Useful for initializing more sophisticated clustering algorithms**
- **Random selection of C samples**
- **Division of variables [Webb] :**
  - Choose one variable (e.g. principal component), divide its range into C bins. Every sample is then classified according to the bin its falls into. Finally, cluster centers are chosen as the means of the C resulting groups.
- **Incremental algorithms**
  - E.g. Leader Algorithm [Hartigan, 1975]
    - ✓ The first sample becomes the leader of the first cluster
    - ✓ Samples are considered one by one, in turn until they are all labeled
      - – If a sample is close enough to the leader of one class, then it is assigned to that class.
      - – Otherwise, it becomes the leader of a new class
  - Low algorithmic complexity, the number of classes is not required, but a threshold must be tuned
  - Leader - follower : when a sample is assigned to a cluster, its leader moves in the direction of the newly classified point. Neural implementation : ART.

Hierarchical clustering

- **Some algorithms provide a hierarchical representation of data**
  - Such a tree-like representation is often called a dendogram
  - The root of the tree is the whole data set, the nodes represent sub-clusters
  - The leafs are the individual samples





- **The dendogram also provides a measure of (dis-)similarity between groups**
  - …which is not the case for Venn diagrams

1rst family: splitting algorithms

- ***Top-down*** **or “divisive” algorithms**
  - Start with a single cluster that contains the whole data set
  - Repeat
    - ✓ Choose the “worst” cluster
    - ✓ Split it
  - Until the number of clusters is equal to the number of samples, N
- **Choosing the “worst” cluster ?**
  - E.g. according to the number of elements, to the variance
- **Splitting clusters ?**
  - E.g. perpendicularly to the direction of greatest variance...
- **This technique may be complicated and rather costly.**

⇨ **Ascending techniques are often preferred.**

2nd family: merging algorithms

- **B*ottom-up*** **or “agglomerative” algorithms**
  - Start with N singleton clusters
  - Repeat
    - ✓ Choose the 2 closest clusters
    - ✓ Merge them
  - Until the number of clusters is 1
- **Choosing the closest clusters ?**
  - According to the distance between groups
  - Some popular choices are:

$$d_{\min}(\omega_i,\omega_j)=\min_{\substack{\mathbf{x}\in\omega_i\\ \mathbf{y}\in\omega_j}}\|x-y\|$$

$$d_{average}(\omega_i,\omega_j)=\frac{1}{n_in_j}\sum_{\mathbf{x}\in\omega_i}\sum_{\mathbf{y}\in\omega_j}\|x-y\|$$

$$d_{\max}(\omega_i,\omega_j)=\max_{\substack{\mathbf{x}\in\omega_i\\ \mathbf{y}\in\omega_j}}\|x-y\|$$

$$d_{centroid}(\omega_i,\omega_j)=\|\mu_i-\mu_j\|$$

Main algorithms

- **Single linkage (or minimum algorithm)**
  - Uses the minimum distance between clusters: “min(min)”
  - Is rather versatile (e.g. it can handle concentric clusters)
  - Produces elongated clusters by linking clusters, without loops or circuits,
  - The result is dubbed *Minimal Spanning Tree*.
- **Complete linkage (or maximum algorithm)**
  - Uses the maximum distance between clusters: “min(max)”
  - Produces compact clusters
  - This produces graphs in which edges connect all of the nodes in a cluster.
  - Each resulting cluster is a *complete subgraph*.
- **Both distances are sensitive to outliers**
  - Using the average distance or the distance between centroids may be better.

Single linkage: example [Webb]

Consider the dissimilarity matrix

	1	2	3	4	5	6
1	0	4	13	24	12	8
2		0	10	22	11	10
3			0	7	**3**	9
4				0	6	18
5					0	8,5
6						0

1. We merge 5 and 3

	1	2	(3,5)	4	6
1	0	**4**	12	24	8
2		0	10	22	10
(3,5)			0	6	8,5
4				0	18
6					0

2. We merge 1 and 2

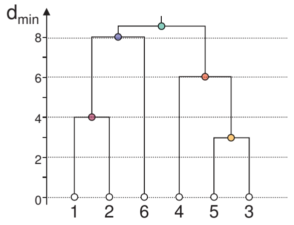
	(1,2)	(3,5)	4	6
(1,2)	0	10	22	8
(3,5)		0	**6**	8,5
4			0	18
6				0

3. We merge (3,5) and 4

	(1,2)	(3,4,5)	6
(1,2)	0	10	**8**
(3,4,5)		0	8,5
6			0

4. We merge (1,2) and 6

	(1,2,6)	(3,4,5)
(1,2,6)	0	**8,5**
(3,4,5)		0

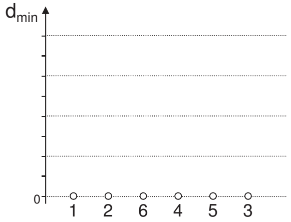


Complete linkage: exercise

- Same exercise with the following dissimilarity matrix

	1	2	3	4	5	6
1	0	4	13	24	12	8
2		0	10	22	11	10
3			0	7	3	9
4				0	6	18
5					0	8,5
6						0

Recall that we always merge the closest clusters (even if a max is used to evaluate the distances)



Remarks

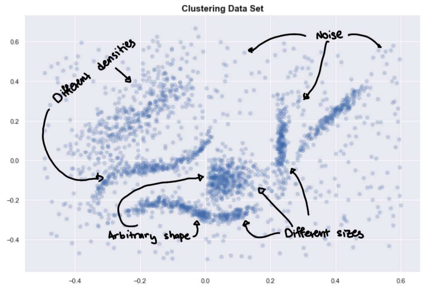
- Different distances between groups were proposed
  - Each one has a particular effect on the results of the clustering algorithm
  - Some distances may be related to the minimization of an objective function
  - E.g. Ward’s distance corresponds to the quadratic error criterion
    - ✓ Merging groups necessarily increases intra-cluster variance
    - ✓ Using Ward’s distance minimizes this increase

$$d_{Ward}(\omega_i,\omega_j)=\frac{n_i n_j}{n_i+n_j}\|m_i-m_j\|^2 \qquad J_e=\sum_{i=1}^{C}\sum_{x\in\omega_i}\|x-m_i\|^2$$

- Choosing the best number of groups ?
  - It is a recurring and difficult question… some solutions are
    - ✓ Monitoring the “lifetime” or persistence of clusters in the dendogram
    - ✓ When an objective criterion is used (ex. J_e), see how it evolves as the number of clusters changes
    - ✓ Use a self-similarity criterion
    - ✓ Use hypothesis testing [Duda]...

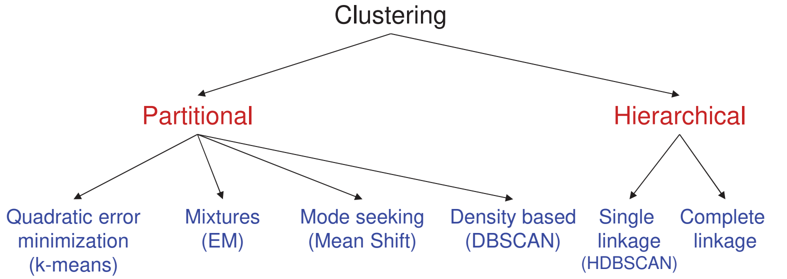
Hierarchical Density Based Clustering (HDBSCAN)

- Introduced by Campello et al (2013, 2015)
- Hierarchical version of (a reinterpreted version of) DBSCAN
- Core distance
  - d_{core}=distance from a point to its k-th neighbor (same as DBSCAN)
- Hierarchical clustering
  - Single linkage
  - Mutual reachability distance $\min\{d_{core}(x), d_{core}(y), d(x,y)\}$
- Flattening the hierarchy
  - Based on cluster persistence (cluster stability)



https://towardsdatascience.com/a-gentle-introduction-to-hdbscan-and-density-based-clustering-5fd79329c1e8

Overview (following [Jain99])


Clustering
Partitional
Hierarchical
Quadratic error minimization (k-means)
Mixtures (EM)
Mode seeking (Mean Shift)
Density based (DBSCAN)
Single linkage (HDBSCAN)
Complete linkage

- Clustering methods:
  - Deterministic vs. probabilistic
  - Flat vs. hierarchical
  - Monothetical vs. polythetical (use a single coordinate of x or the whole x)
  - Hard vs. fuzzy
  - Incremental or non-incremental

Supplementary material

A "good property" of the expectation [Lange2000]

- **Jensen's inequality for expectation:**

$$E[h(x)] \geq h(E[x]) \text{ for } h \text{ convex}$$

- **Taking $h = -\ln$, one shows that for two pdf's, p and q**

$$E_p[\ln(p)] - E_p[\ln(q)] \geq 0$$

  - Demo: $-\ln(.)$ being convex, $E_p[\ln(p)] - E_p[\ln(q)] = E_p\left[-\ln\left(\frac{q}{p}\right)\right] \geq -\ln\left(E_p\left[\frac{q}{p}\right]\right)$

$$-\ln E_p\left[\frac{q}{p}\right] = -\ln\left(\int \frac{q}{p} p\right) = -\ln\left(\int q\right) = \ln(1) = 0$$

- **Now take $p = p_{\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\Theta}^n}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\Theta}^n)$ and $q = p_{\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\Theta}^n}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\Theta}^{n+1})$,**

i.e. $E_p[\ln(p)] = H(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n)$ and $E_p[\ln(q)] = H(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n)$

$$H(\boldsymbol{\Theta}^n|\boldsymbol{\Theta}^n) - H(\boldsymbol{\Theta}^{n+1}|\boldsymbol{\Theta}^n) \geq 0$$

M-step, Gaussian Mixture Model (scalar case)

- **The function that is maximized during the M-step is**

$$\mathrm{Q}(\Theta|\Theta^n) = \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log P(\omega_j) + \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log p(x_k|\boldsymbol{\theta}_j)$$

where $\Theta = \{P(\omega_j), \boldsymbol{\theta}_j\}$ are the parameters of the mixture, and $\boldsymbol{\theta}_j$ is the set of parameters of component ω_j.

- **In the following slides, we develop the calculation of the optimal parameters, for the Gaussian Mixture Model (GMM)**
  - As is well-known, the method consists in cancelling the derivative of the function with respect to (w.r.t.) the parameter in question.

M-step, Gaussian Mixture Model (scalar case)

- **Recall that if $x \sim \mathcal{N}(\mu, \sigma^2)$**

$$\log p(x|\mu,\sigma^2) = -\frac{(x-\mu)^2}{2\sigma^2} - \log\sigma - \frac{1}{2}\log 2\pi$$

- **Then, neglecting the additive constant, we have**

$$\mathrm{Q}(\Theta|\Theta^n) = \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log P(\omega_j) - \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \frac{(x_k-\mu_j)^2}{2\sigma_j^2} - \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log\sigma_j$$

- **Maximizing $\mathrm{Q}(\Theta|\Theta^n)$ w.r.t. μ_j only involves the 2nd term**

$$2\sum_{k=1}^{N} w_{kj}^n \frac{(x_k-\mu_j)}{2\sigma_j^2} = 0 \rightarrow \sum_{k=1}^{N} w_{kj}^n x_k = \mu_j \sum_{k=1}^{N} w_{kj}^n \Rightarrow \boxed{\mu_j = \frac{\sum_{k=1}^{N} w_{kj}^n x_k}{\sum_{k=1}^{N} w_{kj}^n}}$$

M-step, Gaussian Mixture Model (scalar case)

$$Q(\Theta|\Theta^n) = \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log P(\omega_j) - \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \frac{\left(x_k-\mu_j\right)^2}{2\sigma_j^2} - \sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log\sigma_j$$

- **Maximizing $Q(\Theta|\Theta^n)$ w.r.t. $\boldsymbol{\sigma_j}$ involves the last two terms**

$$0 = \sum_{k=1}^{N} w_{kj}^n \frac{\left(x_k-\mu_j\right)^2}{\sigma_j^3} - \sum_{k=1}^{N} \frac{w_{kj}^n}{\sigma_j}$$

- **Assuming $\boldsymbol{\sigma_j}$ to be non-null, we obtain**

$$\boxed{\sigma_j^2 = \frac{\sum_{k=1}^{N} w_{kj}^n\left(x_k-\mu_j\right)^2}{\sum_{k=1}^{N} w_{kj}^n}}$$

M-step, Gaussian Mixture Model (scalar case)

- **For $P(\omega_j)$ we use the constraint $\sum_{j=1}^{j=C} P(\omega_j) = 1$ and maximize**

$$\sum_{j=1}^{C}\sum_{k=1}^{N} w_{kj}^n \log P(\omega_j) + \lambda\left(1 - \sum_{j=1}^{C} P(\omega_j)\right)$$

- **Cancelling the derivative w.r.t. $P(\omega_j)$, we obtain**

$$0 = \sum_{k=1}^{N} \frac{w_{kj}^n}{P(\omega_j)} - \lambda \ \rightarrow\ P(\omega_j) = \frac{1}{\lambda}\sum_{k=1}^{N} w_{kj}^n$$

- **Summing over j and since $\sum_{j=1}^{j=C} P(\omega_j) = 1$**

$$\lambda = \sum_{j=1}^{j=C}\sum_{k=1}^{N} w_{kj}^n = N \ \Rightarrow\ \boxed{P(\omega_j) = \frac{1}{N}\sum_{k=1}^{N} w_{kj}^n}$$

Solutions of exercises

Unsupervised hierarchical clustering algorithms

Complete linkage: exercise

Consider the dissimilarity matrix

	1	2	3	4	5	6
1	0	4	13	24	12	8
2		0	10	22	11	10
3			0	7	**3**	9
4				0	6	18
5					0	8,5
6						0

1. We merge 5 and 3

	1	2	(3,5)	4	6
1	0	**4**	13	24	8
2		0	11	22	10
(3,5)			0	7	9
4				0	18
6					0

2. We merge 1 and 2

	(1,2)	(3,5)	4	6
(1,2)	0	13	24	10
(3,5)		0	**7**	9
4			0	18
6				0

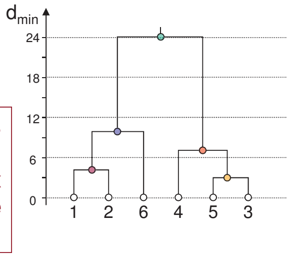
3. We merge (3,5) and 4

	(1,2)	(3,5,4)	6
(1,2)	0	24	**10**
(3,5,4)		0	18
6			0

4. We merge (1,2) and 6

	(1,2,6)	(3,5,4)
(1,2,6)	0	**24**
(3,5,4)		0

Recall that one always merge the <u>closest</u> clusters (even if a max is used to evaluate the distances)


d_{min}