In [1]:
```
# settings...↔
```

```
Populating the interactive namespace from numpy and matplotlib

/Applications/anaconda/lib/python3.6/site-packages/IPython/html.py:14: ShimWarni
ng: The `IPython.html` package has been deprecated. You should import from `note
book` instead. `IPython.html.widgets` has moved to `ipywidgets`.
  "`IPython.html.widgets` has moved to `ipywidgets`.", ShimWarning)
```

\usepackage*amssymb*

4.5 Prototype-based Clustering

Goal: Partitioning by assignment of N data points to $K < N$ typical prototypes (representing the clusters)

There are two variants of the assignment:

(1.) **Deterministic Assignment**:

- Each data point is exactly assigned to a single prototype (hard clustering)

(2.) **Probabilistic Assignment**:

- Each data point $\vec{x}_i$ is assigned with probability h_{ij} to the j-th prototype (soft clustering)

$$\sum_{j=1}^{K} h_{ij} = 1 \forall\, i \quad \text{(normalization)}$$

Remark: vectors

- The assignment (or membership) matrix H has N=nr of data points rows and K=nr. of prototypes columns.
  - each row sums to 1 (i.e. each data point is assigned)
  - each column k sums to the net mass of the kth cluster
- (1.) is a special case of (2.) with h_{i*} being the canonical basis

4.5.1. Hard Clustering by Vector Quantization

Idea: Each data point $\vec{x}_i$ is assigned to one of K prototypes represented by $\{\vec{w}_j\}_{j=1\ldots K}$ by using a label $l(\vec{x}_i) \in \{1, \ldots, K\}$

- This can be regarded as a compression for lossy transmission.
  1. Coding: $\vec{x} \to l \in \{1, \ldots, K\}$
  2. Decoding: $l \to \vec{w}_l$
- Goal: Minimization of the average quantization errors E by varying the $\vec{w}_j$.
- Possible Error function

$$E = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} h_{ij} \|\vec{x}_i - \vec{w}_j\|^2$$

with $h_{ij} = \begin{cases} 1 & j = l(x_i) \\ 0 & \text{else} \end{cases}$

- Gradient descent is not possible because the assignment variables h_{ij} are discrete.

The following algorithm allows optimization:

1. Initialization of prototypes, e.g. $\vec{w}_j = \vec{x}_j,\ j = 1 \dots K$
2. While keeping prototypes fixed, choose assignment h_{ij} so that E is minimized (Voronoi cells, winnter-takes all rule)
3. While keeping assignments fixed, choose prototypes so that E is minimized, i.e. $\nabla_{\vec{w}} E = 0$

$$\nabla_{\vec{w}_j} E = \frac{1}{N} \sum_{i=1}^{N} h_{ij}(2\vec{w}_j - 2\vec{x}_i) \stackrel{!}{=} 0$$

$$\Rightarrow \vec{w}_j = \frac{\sum_i h_{ij}\vec{x}_i}{\sum_i h_{ij}} \ \forall\, j$$

   - Note that the denominator counts all data points with the jth voronoi cell
   - The solution is analog to the 'principal points' before, here for each cluster
4. If the error decrease $|\Delta E| < \epsilon$: stop, else goto 2.

Remarks:

- Convergence to local minima is certain since step 2 and 3 reduce the error and $E \geq 0$ is a lower limit.
- The result depends on the initialization. Due to the risk of getting stuck in suboptimal local minima, usually different initializations are tested and the best clustering taken.

4.5.2. Soft clustering with Mixture Models

Goal: Implementing the possibility to represent uncertainty in the assignment of data to clusters (expressed in form of assignment probabilities)

Approach:

- Description of the data distribution by a probability density $p(\vec{x})$.
- Instead of disjunct partitioning into 'hard' clusters → mixture model:
  - decomposition of $p(\vec{x})$ in additive cluster terms which may overlap and thus describe a soft cluster partitioning

$$p(\vec{x}) = \sum_{j=1}^{M} g_j \mathcal{N}(\vec{x}, \mu_j, \Sigma_j)$$

$$\text{with normalization} \sum_{j=1}^{M} g_j = 1$$

- N is the normal distribution, but any other distribution is also possible

Interpretation:

- Each summand describes a soft cluster C_j with centroid $\vec{\mu}_j$ and a spatial extension which is determined by the covariance matrix Σ_j
- The parameter $g_j = p(j)$ represents the probability at which an arbitrary data point belongs to the j-th cluster, i.e. the a priori probability for the membership to a cluster
- Knowledge of a data point $\vec{x}$ allows the computation of a more precise a posteriori membership probability to cluster j:

$$p_j(\vec{x}) = \frac{g_j N(\vec{x}, \vec{\mu}_j, \Sigma_j)}{\sum_k g_k N(\vec{x}, \vec{\mu}_k, \Sigma_k)}$$

  - according to the Bayes's theorem, here with $p(\vec{x}|j) = N(\vec{\mu}_j, \Sigma)$
- Note that the probabilitites $p_j(\vec{x}_i)$ replace the binary cluster memberships h_{ij} used in hard clustering.

Question: What parameters $\Theta = \{g_j, \vec{\mu}_j, \Sigma_j\}$ are the most probable ones according to the information given in the data set D?

$$P(\Theta|\underbrace{\{\vec{x}_1 \dots \vec{x}_n\}}_{=D}) = \frac{P(D|\Theta)P(\Theta)}{\underbrace{\sum_{\Theta} P(D|\Theta)P(\Theta)}_{=P(D)}}$$

- the prior distribution for Θ without prejudice: $\Rightarrow P(\Theta) = 1$
- Likelihood function L: is the Θ-dependent part of the right side of the above equation.

$$\mathcal{L} = \log L = \log P(D|\Theta) = \log \prod_{i=1}^{N} p(\vec{x}_i)$$

$$\mathcal{L}[\{\vec{g}, \vec{\mu}, \Sigma\}] = \sum_{i=1}^{N} \log p(\vec{x}_i) = \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} g_k N(\vec{x}_i, \vec{\mu}_k, \Sigma_k) \right]$$

  - Note that this is a product since the data points are independent samples from the distribution.
- we seek the maximum of L regarding $g_k, \vec{\mu}_k, \Sigma_k$!
- side condition: $\sum_{j=1}^{K} g_j = 1$

It can be shown that a local maximization of $\mathcal{L}$ can be achieved by the following heuristic motivated iterative method:

- the trick is the application of Jensen's inequality $\ln(\sum_j \lambda_j q_j) \geq \sum_j \lambda_j \ln(q_j)$ to simplify the sum, e.g. see Bishop, Neural Networks for Pattern Recognition, Mixture Models, S. 67, or: [http://en.wikipedia.org/wiki/Expectation-maximization_algorithm (http://en.wikipedia.org/wiki/Expectation-maximization_algorithm)](http://en.wikipedia.org/wiki/Expectation-maximization_algorithm)

1. Initialization: Starting values $t = 0,\ g_k^0,\ \vec{\mu}_k^0,\ \Sigma_k$
   - e.g. obtained by one of the otehr cluster methods
2. Computation of new membership probabilities (E-Step):

$$p_j^{t+1}(\vec{x}_i) = \frac{g_j^t\, \mathcal{N}(\vec{x}_i; \mu_j^t, \Sigma_j^t)}{\sum_k g_k^t \mathcal{N}(\vec{x}_i; \mu_k^t, \Sigma_k^t)} \qquad \text{corresponds to } h_{ij}^{(t+1)}$$

   - $p_j(\vec{x}_i)$ corresponds to the (here now continuous) h_{ij} from hard clustering
3. Computation of new estimates for the cluster centers $\vec{\mu}_k^{t+1}$ as well as for the cluster covariances Σ_k^{t+1}:

$$g_k^{t+1} = \frac{1}{N} \sum_i p_k^{t+1}(\vec{x}_i)$$

(updated probability mass in cluster k)

$$\vec{\mu}_k^{t+1} = \frac{1}{g_k^{t+1}} \sum_i p_k^{t+1}(\vec{x}_i) \cdot \vec{x_i}$$

(updated centroid of cluster k)

$$\Sigma_k^{t+1} = \frac{1}{g_k^{t+1}} \sum_i \left(p_k^{t+1}(\vec{x}_i) \cdot (\vec{x}_i - \vec{\mu}_k)(\vec{x}_i - \vec{\mu}_k)^T \right)$$

(updated covariance of cluster k)

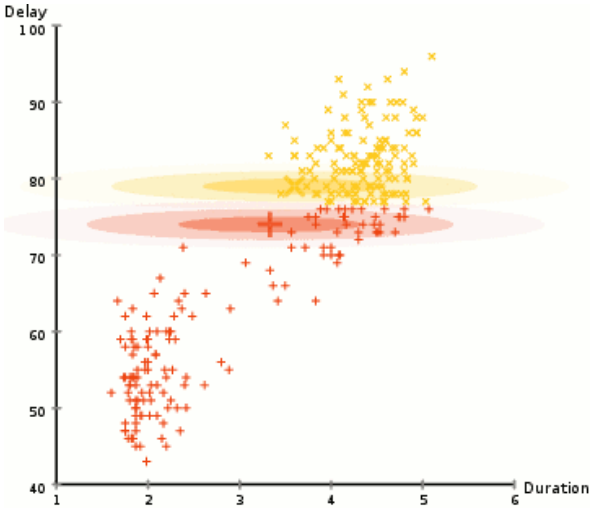
4. If the maximal number of iterations has not yet been reached: goto 2.

This method is known as EM algorihtm (Expectation-Maximization algorithm).

Remarks:

- EM-algorithm delivers local optima and no garantuee for finding a global optimum
- remedy: repeat with different initializations, take the best result.
- Convergence problems with 'degenerated clusters', i.e. if Σ_k is almost singular
  - this can happen if too few data points $< d$ are within a cluster
  - or is all points lie within a linear subspace
- here: special solution under the assumption of normal distributed clusters and vectorial represented data.
- The results of the clustering is similar to the Ward-clustering, but now the membership is not greedy, but globally and thus more effort, but potentially better
- Attention: minimization makes only sense for a given number of clusters c. Optimizing additionally w.r.t. c is useless: this leads always to the trivial solution of $c = N$ clusters centered at data points.

Example (from wikipedia)



4.6. Evaluation of Clustering results

Questions:

- How many clusters exist?
- How unique is the found clustering?

4.6.1. The number of clusters

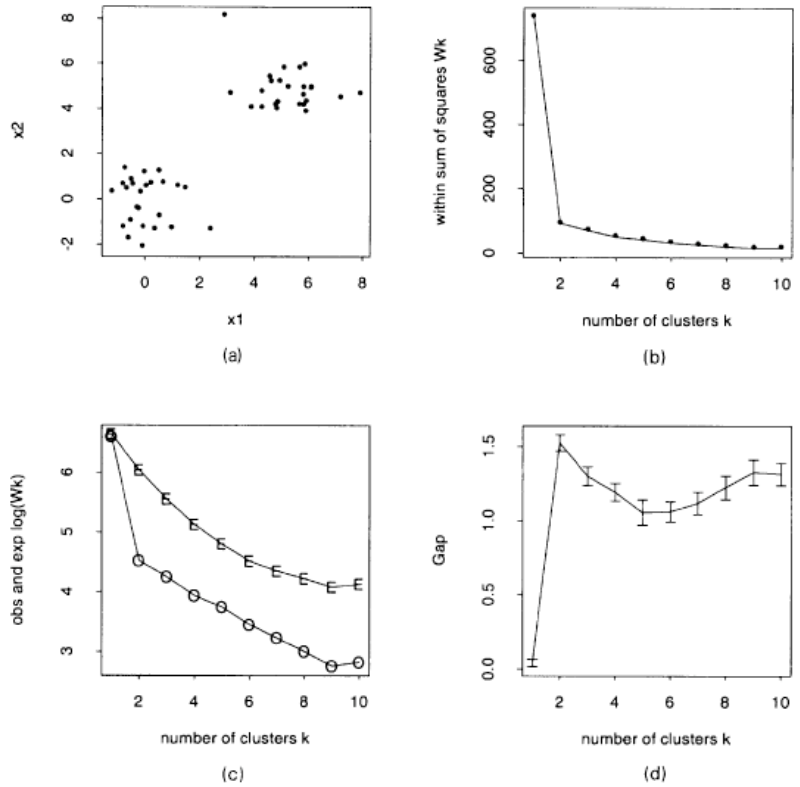
Basic Idea:

- Compare the cluster quality for a given number of clusters c
- choose the smallest number of clusters with acceptable quality.
- $\rightarrow$ this shifts the problem towards the definition of a suitable cluster quality measure.

Gap-Statistic (Tibshirani, G. Walther, T. Hastie, 2001, https://web.stanford.edu/~hastie/Papers/gap.pdf (https://web.stanford.edu/~hastie/Papers/gap.pdf)):

- compares the logarithmized intra cluster variance
$$g_c(D) = \log \mathrm{Tr}(\mathbf{W}_c(D))$$
for c clusters with their centroid $\langle g_c(U_i)\rangle_i$ for enforced clustering of M uniform distributed random sets U_i $(i = 1, \dots M)$ into likewise c clusters.
- original formulation:
$$= \log \sum_{r=1}^{c} \frac{1}{2n_r} D_r$$
  - with $n_r = |C_r|$ (size of clusters)
  - and $D_r = \sum_{i,i' \in C_r} d(\vec{x}_i, \vec{x}_{i'})$ (data point distances)
  - interestingly D_r is $2E$, two times the MSE to cluster centers, if d is the squared Euclidean distance
- the first 'clear/distinct' peak of the difference
$$\mathrm{Gap}(c) = \langle g_c(U_i)\rangle - g_c(D)$$
points at the 'true' number of clusters c_{opt}:

- Example illustration from Tibshirani:



  - Figure (b) depicts the gap for the data set in Figure (a)
- more accurate is c_{opt} = smallest value c, for which

$$\text{Gap}(c) \geq \text{Gap}(c+1) - s_{c+1}$$

where

$$s_c = \sqrt{\left(1 + \frac{1}{M}\right) \text{Var}(g_c(U_i))}$$

- The U_i are randomly generated as equally large ($|U_i| = |D|$) random sets of a D-containing box in feature space.
- If the data distribution is longer than thick, it is recommended to align the box along the principal axes of the data distribution.

Further, more simple propositions :

- Approach by Calinsky & Harabasz:

$$c_{opt} = \arg\max_{c \geq 2} \frac{n-c}{c-1} \frac{\text{trace}(\mathbf{B}_c)}{\text{trace}(\mathbf{W}_c)}$$

  - Only applicable for $c \geq 2$
  - performed best in a comparison of 20 alternative approaches
- Alternative approach by Hartican (1975):

$$c_{opt} = \min_c \left\{ c \middle| \left(\frac{\text{trace}(\mathbf{W}_c)}{\text{trace}(\mathbf{W}_{c+1})} - 1 \right) \geq 10(n-c-1) \right\}$$

  - The idea is to start with 1 cluster and add more as long as $H(c) = \left(\frac{\text{trace}(\mathbf{W}_c)}{\text{trace}(\mathbf{W}_{c+1})} - 1 \right) \frac{1}{n-c-1}$ is sufficiently large.
- For mixture models the Bayes-Criterion is applicable:

$$\text{BIC}(c) = 2 \log P(D|\theta_c) - m_c \log(n)$$

  - First term: Log-Likelihood of the data D at model parameters θ_c.
  - Second term: penalty for models with large number of parameters m_c.
  - c_{opt} = value of c at the first distinct maximum of BIC(·).

4.6.2. Uniqueness of Clustering results [...]

- Basic Idea: Compare results of repeated clustering runs under slight variations (see below)
- Measure for comparison for two clusterings C_1 and C_2:

Rand-Index

$$R_g = \frac{N_{++} + N_{--}}{N_{++} + N_{--} + N_{+-} + N_{-+}} = \frac{2(N_{++} + N_{--})}{N(N-1)}$$

with the following definitions:

- N_{++} is the number of data point pairs that are in the same cluster in C_1 and also in the same cluster in C_2
- N_{+-} is the number of data point pairs that are in the same cluster in C_1 but in different clusters in C_2
- N_{-+} is the number of data point pairs that are in different clusters in C_1 but in the same cluster in C_2
- N_{--} is the number of data point pairs that are in different clusters in C_1 and also in different clusters in C_2
- The denominator is the number of all possible point pairs, i.e. $\binom{N}{2} = N(N-1)/2$
- $R_g = 1 \Rightarrow$ both clusterings are "`the same"'.
- Possible modes of variation for the examination of clustering stability
  - different initializations
  - Repetition of the method on different data subsets (equal size)
  - ditto, but variable large data set size
  - elimination of few features should affect the result not too strongly
  - using new variables