UNSUPERVISED LEARNING AND CLUSTERING

Jeff Robble, Brian Renzenbrink, Doug Roberts

Unsupervised Procedures

A procedure that uses *unlabeled* data in its classification process.

Why would we use these?

- Collecting and labeling large data sets can be costly
- Occasionally, users wish to group data first and label the groupings second
- In some applications, the pattern characteristics can change over time. Unsupervised procedures can handle these situations.
- Unsupervised procedures can be used to find useful features for classification
- In some situations, unsupervised learning can provide insight into the structure of the data that helps in designing a classifier

Unsupervised vs. Supervised

Unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise. How does it compare to supervised learning?

With unsupervised learning it is possible to learn larger and more complex models than with supervised learning. This is because in supervised learning one is trying to find the connection between two sets of observations, while unsupervised learning tries to identify certain latent variables that caused a single set of observations.

The difference between supervised learning and unsupervised learning can be thought of as the difference between discriminant analysis from cluster analysis.

Mixture Densities

We assume that p($\mathbf{x}|\omega_j$) can be represented in a functional form that is determined by the value of parameter vector $\boldsymbol{\theta}_j$.

For example, if we have p($\mathbf{x}|\omega_j$) ~ N($\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$), where N is the function for a normal gaussian distribution and $\boldsymbol{\theta}_j$ consists of the components $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ that characterize the average and variance of the distribution.

We need to find the probability of **x** for a given ω_j and $\boldsymbol{\theta}$, but we don't know the exact values of the $\boldsymbol{\theta}$ components that go into making the decision. We need to solve:

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j)P(\omega_j)}{p(\mathbf{x})}$$

but instead of p($\mathbf{x}|\omega_j$) we have p($\mathbf{x}|\omega_j,\boldsymbol{\theta}_j$). We can solve for the mixture density:

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, \theta_j)P(\omega_j) \quad (1)$$

Mixture Densities

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, \theta_j) P(\omega_j) \qquad (1)$$

component densities mixing parameters

We make the following assumptions:

- The samples come from a known number of c classes.
- The prior probabilities $P(\omega_j)$ for each class are known, j = 1…c.
- The forms for the class-conditional probability densities $p(\mathbf{x}|\omega_j,\boldsymbol{\theta}_j)$ are known, j = 1…c.
- The values for the c parameter vectors $\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_c$ are <u>unknown</u>.
- The category labels are unknown ➔ unsupervised learning.

Consider the following mixture density where x is binary:

$$P(x \mid \boldsymbol{\theta}) = \frac{1}{2}\theta_1^{\,x}(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^{\,x}(1-\theta_2)^{1-x}$$

Identifiability: Estimate Unknown Parameter Vector θ

$$P(x\,|\,\boldsymbol{\theta}) = \frac{1}{2}\theta_1^{\,x}(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^{\,x}(1-\theta_2)^{1-x} = \begin{cases} \frac{1}{2}(\theta_1+\theta_2) & \text{if x} = 1 \\ 1-\frac{1}{2}(\theta_1+\theta_2) & \text{if x} = 0 \end{cases}$$

Suppose we had an unlimited number of samples and use nonparametric methods to determine p(x|**θ**) such that P(x=1|**θ**)=.6 and P(x=0|**θ**)=.4:

Try to solve for θ_1 and θ_2:

$$\frac{1}{2}(\theta_1+\theta_2) = .6$$

$$-\left[1-\frac{1}{2}(\theta_1+\theta_2) = .4\right]$$

$$\text{-}1+\theta_1+\theta_2 = .2$$

$$\boxed{\theta_1+\theta_2 = 1.2}$$

We discover that the mixture distribution is <u>completely unidentifiable</u>. We cannot infer the individual parameters of **θ**.

A mixture density, p(**x**|**θ**) is <u>identifiable</u> if we can recover a unique **θ** such that p(**x**|**θ**) ≠ p(**x**|**θ’**).

Maximum Likelihood Estimates

The posterior probability becomes:

$$P(\omega_i \mid \mathbf{x}_k, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i) P(\omega_i)}{p(\mathbf{x}_k \mid \boldsymbol{\theta})} \qquad (6)$$

We make the following assumptions:

- The elements of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are functionally independent if $i \neq j$.
- $p(D|\boldsymbol{\theta})$ is a differentiable function of $\boldsymbol{\theta}$, where $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n independently drawn unlabeled samples.

The search for a maximum value of $p(D|\boldsymbol{\theta})$ extending over $\boldsymbol{\theta}$ and $P(\omega_j)$ is constrained so that:

$$P(\omega_i) \geq 0 \quad i = 1, \ldots, c \quad \text{and} \quad \sum_{i=1}^{c} P(\omega_i) = 1$$

Let $\hat{P}(\omega_i)$ be the maximim likelihood estimate for $P(\omega_i)$.

Let $\hat{\theta}_i$ be the maximim likelihood estimate for θ_i.

$$\text{If } \hat{P}(\omega_i) \neq 0 \text{ for any i then } \hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta}) \qquad (11)$$

Maximum Likelihood Estimates

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) \qquad (11)$$

The MLE of the probability of a category is the average over the entire data set of the estimate derived from each sample (weighted equally)

$$\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k \mid \omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k \mid \omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)} \qquad (13)$$

Bayes theorem. When estimating the probability for ω_i, the numerator depends on $\hat{\theta}_i$ and not on the full $\hat{\theta}$.

Maximum Likelihood Estimates

The gradient must vanish at the value of $\boldsymbol{\theta}_i$ that maximizes the logarithm of the likelihood, so the MLE $\hat{\boldsymbol{\theta}}_i$ must satsify the following conditions:

$$\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k \mid \omega_i, \hat{\boldsymbol{\theta}}_i) = 0 \quad \mathrm{i} = 1, \ldots, \mathrm{c} \qquad (12)$$

Consider one sample, so $\mathrm{n} = 1$. Since we assumed $\hat{P} \neq 0$, the probability is maximized as a function of $\boldsymbol{\theta}_i$ so $\nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k \mid \omega_i, \hat{\boldsymbol{\theta}}_i) = 0$. Note that $\ln(1) = 0$, so we are trying to find the a value of $\hat{\boldsymbol{\theta}}_i$ that maximizes p(.).

Applying MLE to Normal Mixtures

Case 1: The only unknown quantities are the mean vectors μ_i .

θ_i consists of components of μ_i

The likelihood of this particular sample is

$$\ln p(\mathbf{x} \mid \omega_i, \mu_i) = -\ln[(2\pi)^{d/2} \mid \Sigma_i \mid^{1/2}] - \frac{1}{2}(\mathbf{x} - \mu_i)^t \Sigma_i^{-1}(\mathbf{x} - \mu_i) \qquad (14)$$

and its derivative is

$$\nabla_{\mu_i} \ln p(\mathbf{x} \mid \omega_i, \mu_i) = \boxed{\Sigma_i^{-1}(\mathbf{x} - \mu_i)} \qquad (15)$$

Thus, according to Equation 8 in the book the MLE estimate $\hat{\mu}_i$

must satisfy: $$\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\mu}) \boxed{\Sigma_i^{-1}(\mathbf{x}_k - \hat{\mu}_i)} = 0$$

where $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_c)^t$ (16)

Applying MLE to Normal Mixtures

If we multiply the above equation by the covariance matrix Σ_i and rearranging terms, we obtain the equation for the maximum likelihood estimate of the mean vector

$$\hat{\mu}_i = \frac{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\mu})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\mu})} \qquad (17)$$

However, we still need to calculate explicitly. If we have a good initial estimate $\hat{\mu}_i(0)$ we can use a hill climbing procedure to improve our estimates

$$\hat{\mu}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\mu}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\mu}(j))} \qquad (18)$$

Applying MLE to Normal Mixtures

Case 2: The mean vector μ_i, the covariance matrix Σ_i, and the prior probabilities $P(\omega_i)$ are all unknown

In this case the maximum likelihood principle only gives singular solutions. Usually, singular solutions are unusable. However, if we restrict our attention to the largest of the finite local maxima of the likelihood function we can still find meaningful results.

Using μ_i, Σ_i, and $P(\omega_i)$ derived from Equations 11-13 we can find the likelihood of θ_i using

$$\ln p(\mathbf{x} \mid \omega_i, \theta_i) = \ln \frac{|\Sigma_i^{-1}|^{1/2}}{(2\pi)^{d/2}} - \frac{1}{2}(\mathbf{x} - \mu_i)^t \Sigma_i^{-1} (\mathbf{x} - \mu_i) \qquad (21)$$

Applying MLE to Normal Mixtures

The differentiation of the previous equation gives

$$\nabla_{\mu_i} \ln p(\mathbf{x} \mid \omega_i, \theta_i) = \Sigma_i^{-1}(\mathbf{x}_k - \mu_i) \qquad (22)$$

and

$$\frac{\partial \ln p(\mathbf{x}_k \mid \omega_i, \theta_i)}{\partial \sigma^{pq}(i)} = (1 - \frac{\delta_{pq}}{2})[\sigma_{pq}(i) - (x_p(k) - \mu_p(i))(x_q(k) - \mu_q(i))] \qquad (23)$$

Where δ_{pq} is the Kronecker delta, $x_p(k)$ is the *p*th element of $\mathbf{x}_k$, $\mu_p(i)$ is the *p*th element of μ_i , and σ_{pq} is the *pq*th element of Σ_i and Σ_i^{-1}

Applying MLE to Normal Mixtures

Using the above differentiation along with Equation 12 we can find the following equations for the MLE of $\hat{\mu}_i$, $\hat{P}(\omega_i)$, and $\hat{\Sigma}_i$

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta}) \tag{24}$$

$$\hat{\mu}_i = \frac{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\theta})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\theta})} \tag{25}$$

$$\hat{\Sigma}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta})(\mathbf{x}_k - \hat{\mu}_i)(\mathbf{x}_k - \hat{\mu}_i)^t}{\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta})} \tag{26}$$

Applying MLE to Normal Mixtures

These equations work where

$$\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta}) = \frac{p(\mathbf{x}_k \mid \omega_i, \hat{\theta}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k \mid \omega_j, \hat{\theta})\hat{P}(\omega_j)} \tag{27}$$

$$= \frac{|\hat{\Sigma}_i|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_k - \hat{\mu}_i)^t]\hat{\Sigma}_i^{-1}(\mathbf{x}_k - \hat{\mu}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} |\hat{\Sigma}_i|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_k - \hat{\mu}_i)^t]\hat{\Sigma}_i^{-1}(\mathbf{x}_k - \hat{\mu}_i)\hat{P}(\omega_i)}$$

To solve the equation for the MLE, we should again start with an initial estimate to evaluate Equation 27, and use Equations 24-26 to update these estimates.

k-Means Clustering

Clusters numerical data in which each cluster has a center called the mean

The number of clusters *c* is assumed to be fixed

The goal of the algorithm is to find the *c* mean vectors $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$

The number of clusters *c*

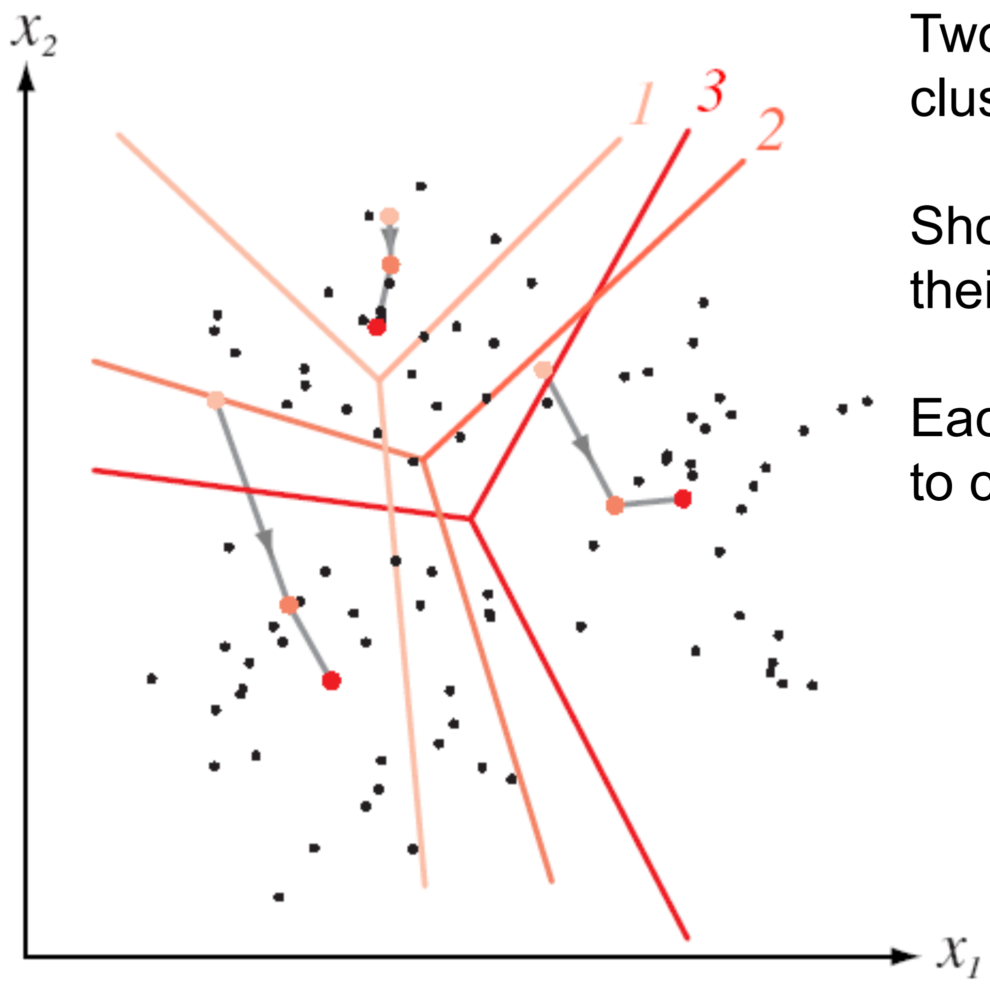
- May be guessed
- Assigned based on the final application

k-Means Clustering

The following pseudo code shows the basic functionality of the k-Means algorithm

```
begin initialize $n$, c, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
        do classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
                recompute $\boldsymbol{\mu}_i$
        until no change in $\boldsymbol{\mu}_i$
    return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
end
```

k-Means Clustering



Two dimensional example with *c = 3* clusters

Shows the initial cluster centers and their associated Voronoi tesselation

Each of the three Voronoi cells are used to calculate new cluster centers

Fuzzy k-Means

The algorithm assumes that each sample x_i has a fuzzy membership in a cluster(s)

The algorithm seeks a minimum of a heuristic global cost function

$$J_{fuz} = \sum_{i=1}^{c} \sum_{j=1}^{n} [\hat{P}(\omega_i \mid \boldsymbol{x}_j, \hat{\theta})]^b \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2$$

Where:

- *b* is a free parameter chosen to adjust the “blending” of clusters
- *b > 1* allows each pattern to belong to multiple clusters (fuzziness)

Fuzzy k-Means

Probabilities of cluster membership for each point are normalized as

$$\sum_{i=1}^{c} \hat{P}\left(\omega_i \middle| \boldsymbol{x}_j\right) = 1,\ j = 1,\dots,n \qquad (30)$$

Cluster centers are calculated using Eq. 32

$$\mu_j = \frac{\sum_{j=1}^{n} [\hat{P}(\omega_i | \boldsymbol{x}_j)]^b \boldsymbol{x}_j}{\sum_{j=1}^{n} [\hat{P}(\omega_i | \boldsymbol{x}_j)]^b} \qquad (32)$$

Where: $$\hat{P}\left(\omega \middle| \boldsymbol{x}_j\right) = \frac{\left(1/d_{ij}\right)^{1/(b-1)}}{\sum_{r=1}^{c} (1/d_{rj})^{1/(b-1)}} \quad \textit{and} \quad d_{ij} = \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2 \qquad (33)$$

Fuzzy k-Means

The following is the pseudo code for the Fuzzy k-Means algorithm

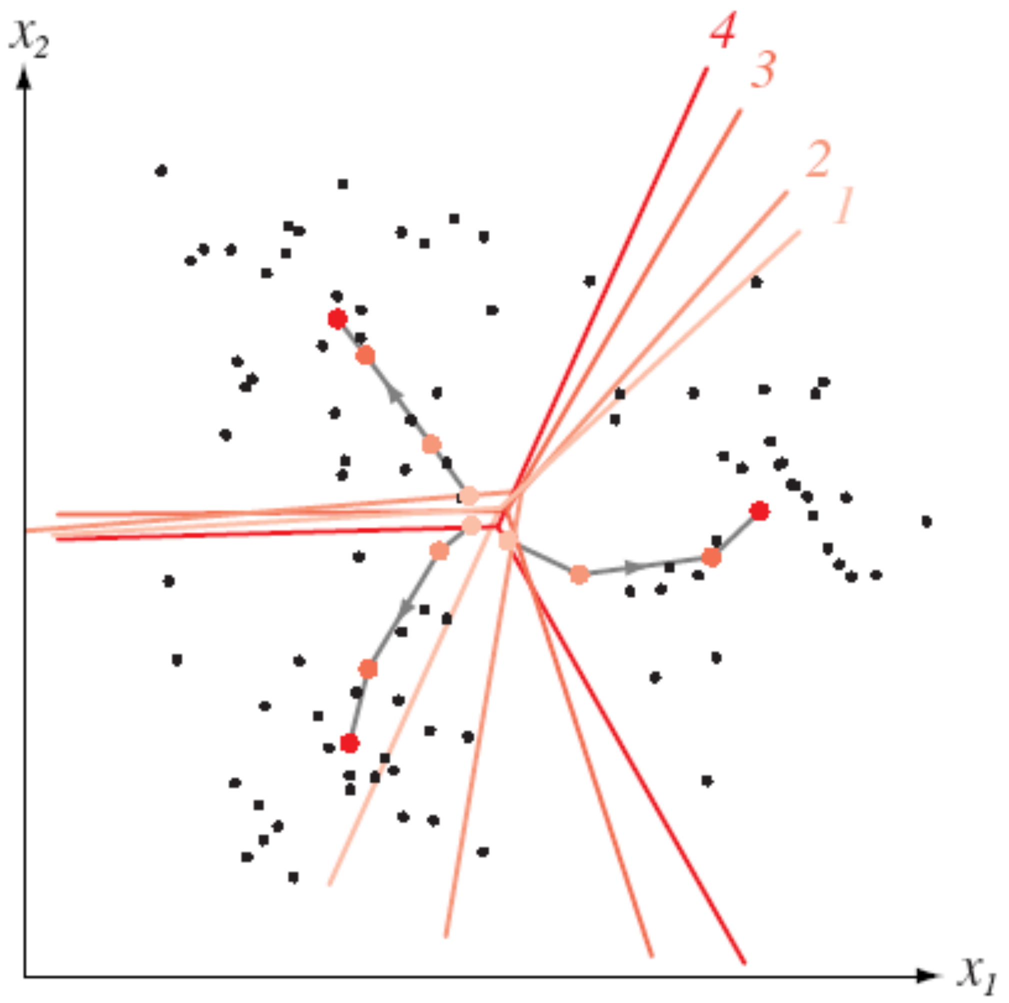
```
begin initialize $n, c, b, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, \hat{P}(\omega_i|\boldsymbol{x}_j)$, $i = 1,\ldots,c; j = 1,\ldots,n$
    normalize $\hat{P}(\omega_i|\boldsymbol{x}_j)$ by Eq. 30
        do recompute $\boldsymbol{\mu}_i$ by Eq. 32
            recompute $\hat{P}(\omega_i|\boldsymbol{x}_j)$ by Eq. 33
        until small change in $\boldsymbol{\mu}_i$ and $\hat{P}(\omega_i|\boldsymbol{x}_j)$
  return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
end
```

Fuzzy k-means

Illustrates the progress of the algorithm

Means lie near the center during the first iteration since each point has non-negligible “membership”

Points near the cluster boundaries can Have membership in more that one cluster



x-Means

In k-Means the number of clusters is chosen before the algorithm is applied

In x-Means the Bayesian information criterion (BIC) is used globally and locally to find the best number of clusters *k*

BIC is used globally to choose the best model it encounters and locally to guide all centroid splits

x-Means

The algorithm is supplied:

- A data set $D = \{x_1, x_2, \ldots, x_n\}$ containing n objects in d-dimensional space
- A set of alternative models $M_i = \{C_1, C_2, \ldots, C_k\}$ which correspond to solutions with different values of k
  - Posterior probabilities $P(M_i \mid D)$ are used to score the models

x-Means

The BIC is defined as

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2}\log(n)$$

Where

- $\hat{l}_j(D)$ is the loglikelihood of D according to the jth model and taken at the maximum likelihood point
- p_j is the number of parameters in M_j

The maximum likelihood estimate is

$$\hat{\sigma}^2 = \frac{1}{n-k}\sum_{i=1}^{n}(\boldsymbol{x}_i - \mu_{(i)})^2$$

Where $\mu_{(i)}$ is the centroid associated with $\boldsymbol{x}_i$

x-Means

The point probabilities are

$$\hat{P}(\boldsymbol{x}_i) = \frac{|C_{(i)}|}{n} \cdot \frac{1}{\sqrt{2\pi}\hat{\sigma}^d} \exp\left(-\frac{1}{2\hat{\sigma}^2} \left\| \boldsymbol{x}_i - \mu_{(i)} \right\|^2 \right)$$

Finally the loglikelihood of the data is

$$l(D) = \log \prod_{i=1}^{n} P(\boldsymbol{x}_j) = \sum_{i=1}^{n} \left[\log\left(\frac{1}{\sqrt{2\pi}\hat{\sigma}^d} \right) - \frac{1}{2\hat{\sigma}^2} \left\| \boldsymbol{x}_i - \mu_{(i)} \right\|^2 + \frac{|C_{(i)}|}{n} \right]$$

x-Means

Basic functionality of the algorithm

- Given a range for k, $[k_{min}, k_{max}]$
- Start with $k = k_{min}$
- Continue to add centroids as needed until k_{max} is reached
- Centroids are added by splitting some centroids in two according to BIC
- The centroid set with the best score is used as the final output

References

Duda, R., Hart, P., Stork, D. Pattern Classification, 2nd ed. John Wiley & Sons, 2001.

G. Gan, C. Ma, and J. Wu. Data Clustering Theory, Algorithms, and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA. 2007

Samet, H., 2008. K-Nearest Neighbor Finding Using MaxNearestDist. *IEEE Trans. Pattern Anal. Mach. Intell. 30, 2 (Feb. 2008), 243-252.*

Yu-Long Qiao, Jeng-Shyang Pan, Sheng-He Sun .Improved partial distance search for k nearest-neighbor classification. IEEE International Conference on Multimedia and Expo, 2004. June 2004, 1275 – 1278.

Ghahramani, Z. Unsupervised Learning. Advanced Lectures on Machine Learning, 2003, 72-112.