Clustering

Initial slides by Eamonn Keogh

What is Clustering?

Also called *unsupervised learning*, sometimes called *classification* by statisticians and *sorting* by psychologists and *segmentation* by people in marketing

- Organizing data into classes such that there is
    - high intra-class similarity
    - low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

Intuitions behind desirable distance measure properties

$D(\mathrm{A},\mathrm{B}) = D(\mathrm{B},\mathrm{A})$ *Symmetry*

Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

$D(\mathrm{A},\mathrm{A}) = 0$ *Constancy of Self-Similarity*

Otherwise you could claim "Alex looks more like Bob, than Bob does."

$D(\mathrm{A},\mathrm{B}) = 0$ If $\mathrm{A}=\mathrm{B}$ *Positivity (Separation)*

Otherwise there are objects in your world that are different, but you cannot tell apart.

$D(\mathrm{A},\mathrm{B}) \leq D(\mathrm{A},\mathrm{C}) + D(\mathrm{B},\mathrm{C})$ *Triangular Inequality*

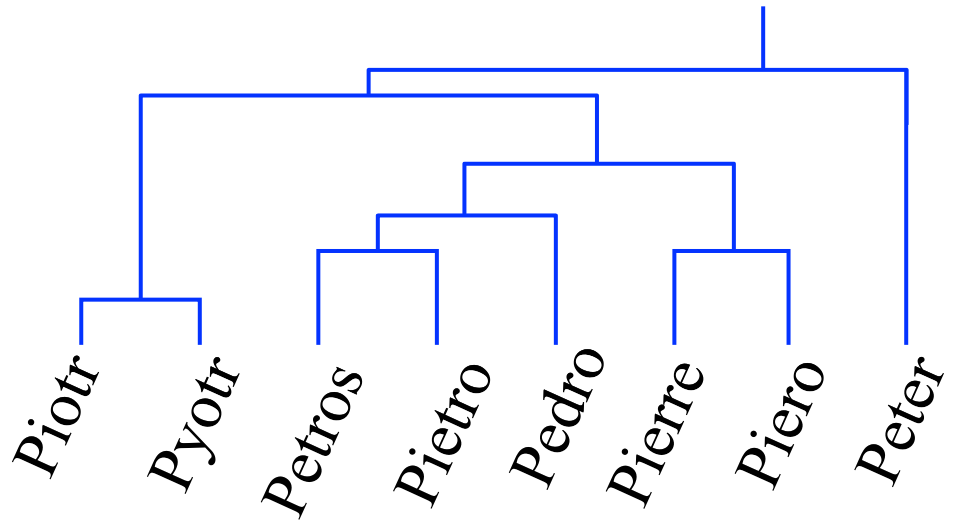
Otherwise you could claim "Alex is very like Carl, and Bob is very like Carl, but Alex is very unlike Bob."

Edit Distance Example

It is possible to transform any string *Q* into string *C*, using only *Substitution*, *Insertion* and *Deletion*.
Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the cost of the cheapest transformation from *Q* to *C*.

Note that for now we have ignored the issue of how we can find this cheapest transformation



How similar are the names "Peter" and "Piotr"?

Assume the following cost function

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(**Peter**,**Piotr**) is 3

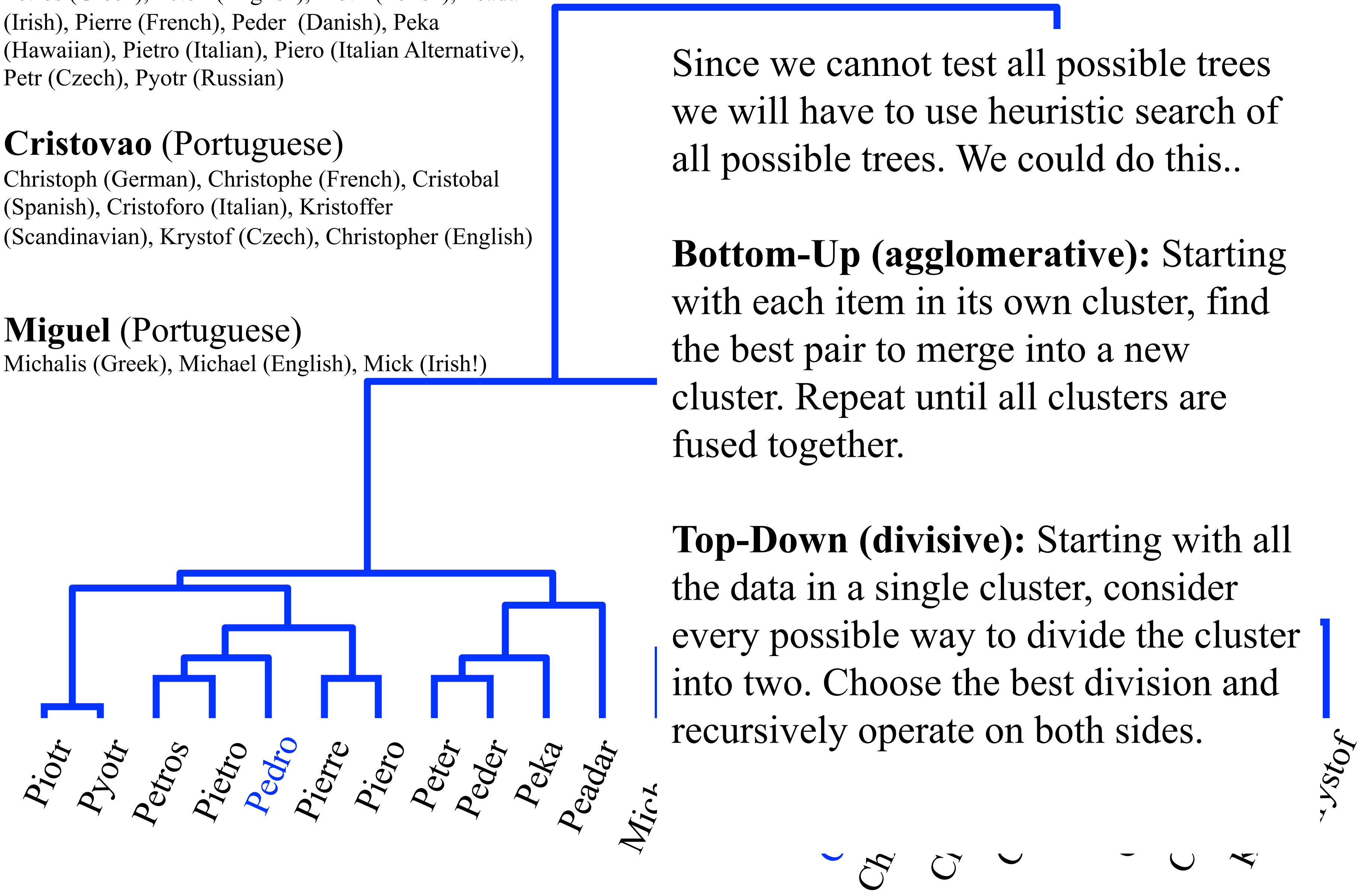
$$\begin{tikzcd}
\texttt{Peter} \arrow[d, "\text{Substitution (i for e)}"] \\
\texttt{Piter} \arrow[d, "\text{Insertion (o)}"] \\
\texttt{Pioter} \arrow[d, "\text{Deletion (e)}"] \\
\texttt{Piotr}
\end{tikzcd}$$

A Demonstration of Hierarchical Clustering using String Edit Distance

Pedro (Portuguese)
Petros (Greek), Peter (English), Piotr (Polish), Peadar (Irish), Pierre (French), Peder (Danish), Peka (Hawaiian), Pietro (Italian), Piero (Italian Alternative), Petr (Czech), Pyotr (Russian)

Cristovao (Portuguese)
Christoph (German), Christophe (French), Cristobal (Spanish), Cristoforo (Italian), Kristoffer (Scandinavian), Krystof (Czech), Christopher (English)

Miguel (Portuguese)
Michalis (Greek), Michael (English), Mick (Irish!)

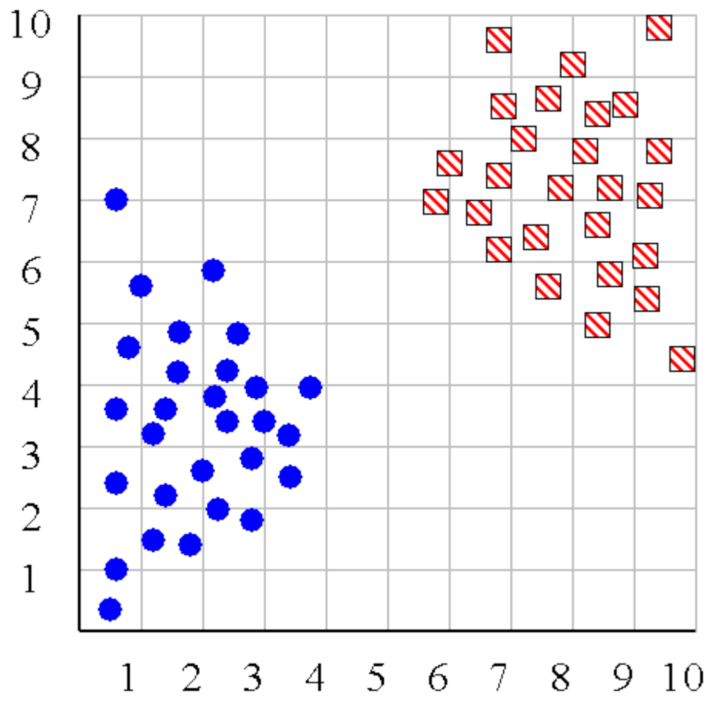


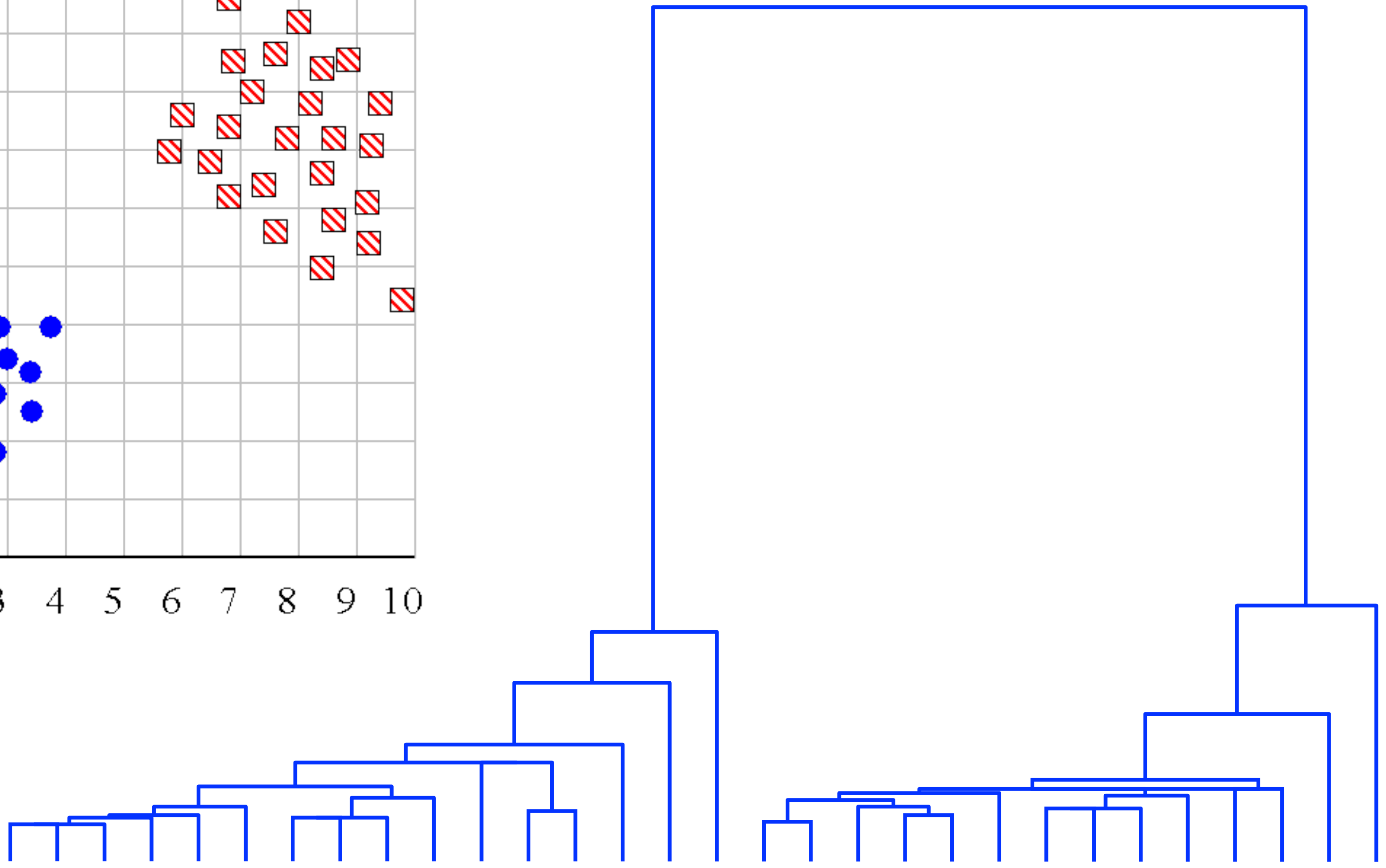
Since we cannot test all possible trees we will have to use heuristic search of all possible trees. We could do this..

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

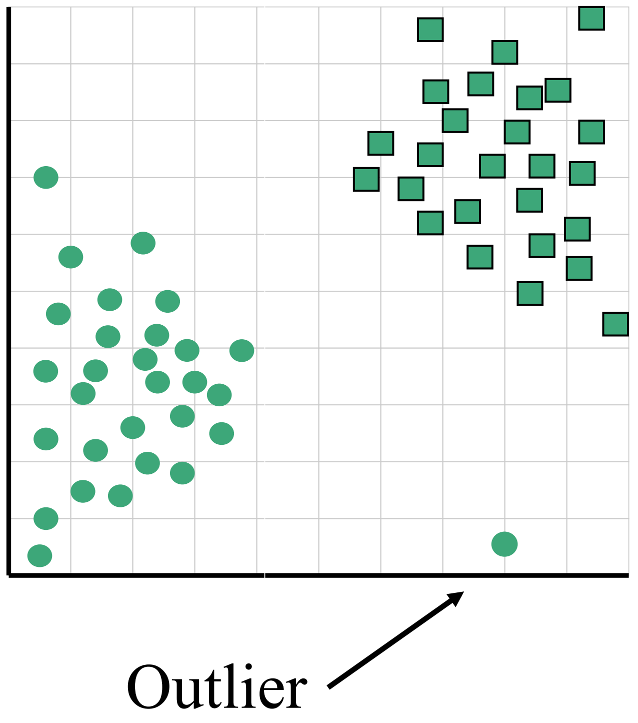
We can look at the dendrogram to determine the "correct" number of clusters. In this case, the two highly separated subtrees are highly suggestive of two clusters. (Things are rarely this clear cut, unfortunately)

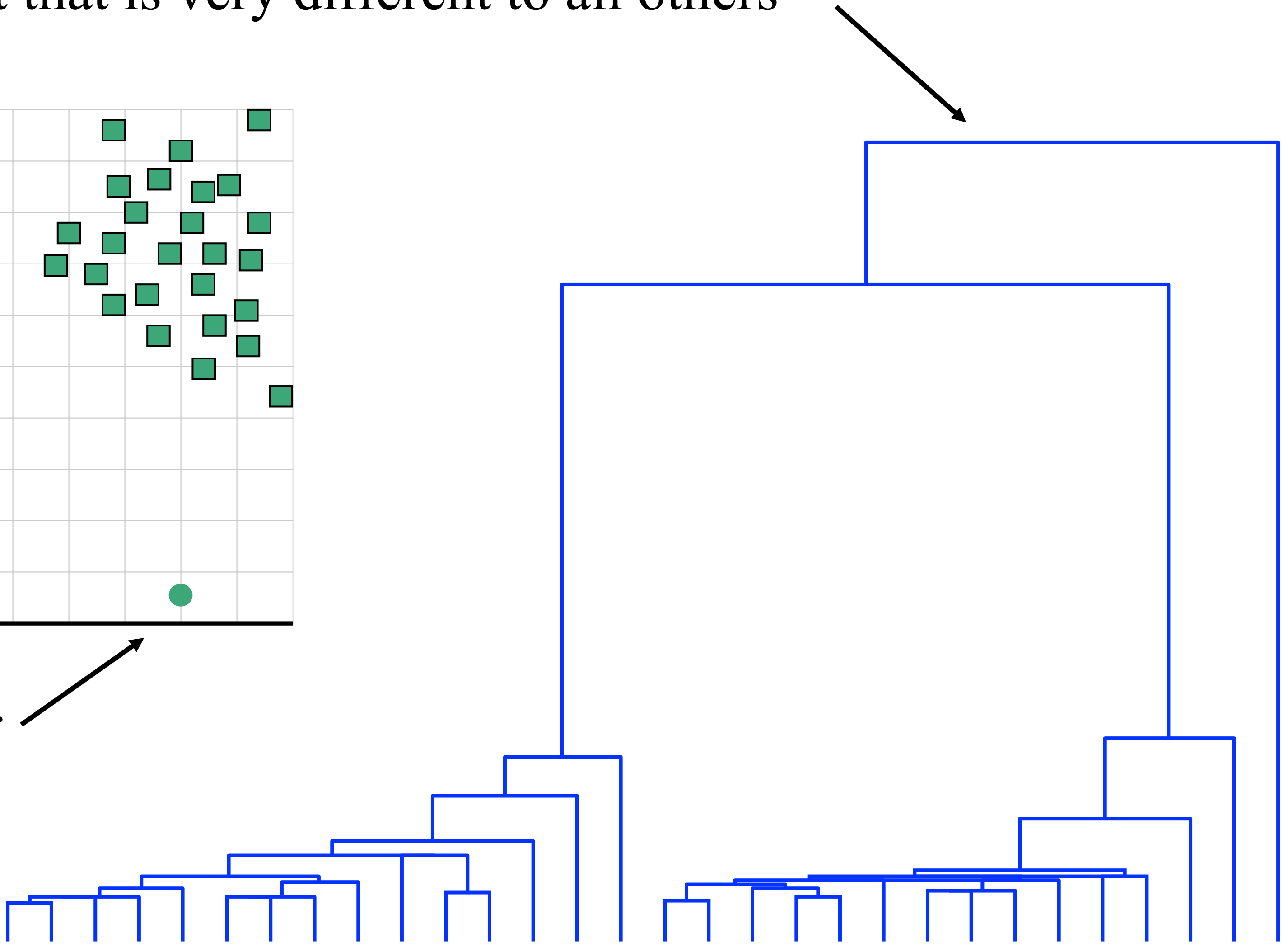




One potential use of a dendrogram is to detect outliers

The single isolated branch is suggestive of a data point that is very different to all others





Partitional Clustering

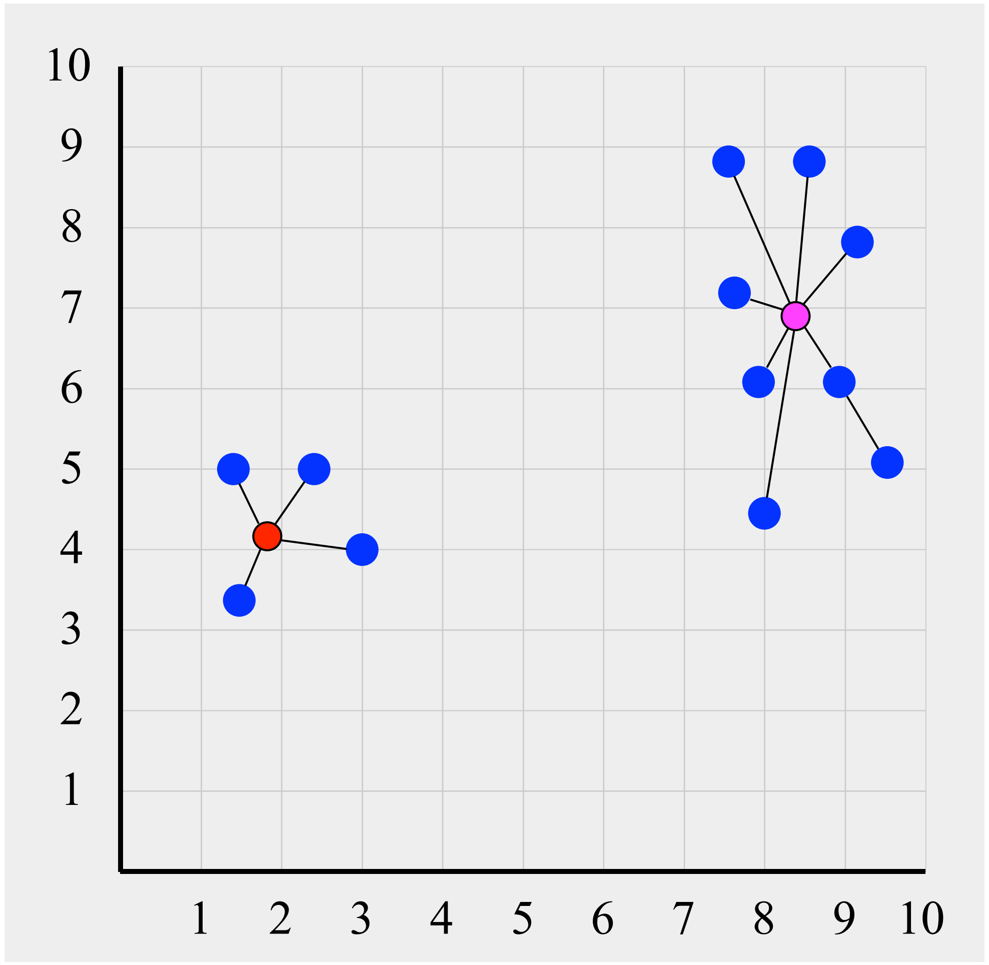
- Nonhierarchical, each instance is placed in exactly one of K nonoverlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.

Squared Error

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



Algorithm *k-means*

1. Decide on a value for *k*.

2. Initialize the *k* cluster centers (randomly, if necessary).

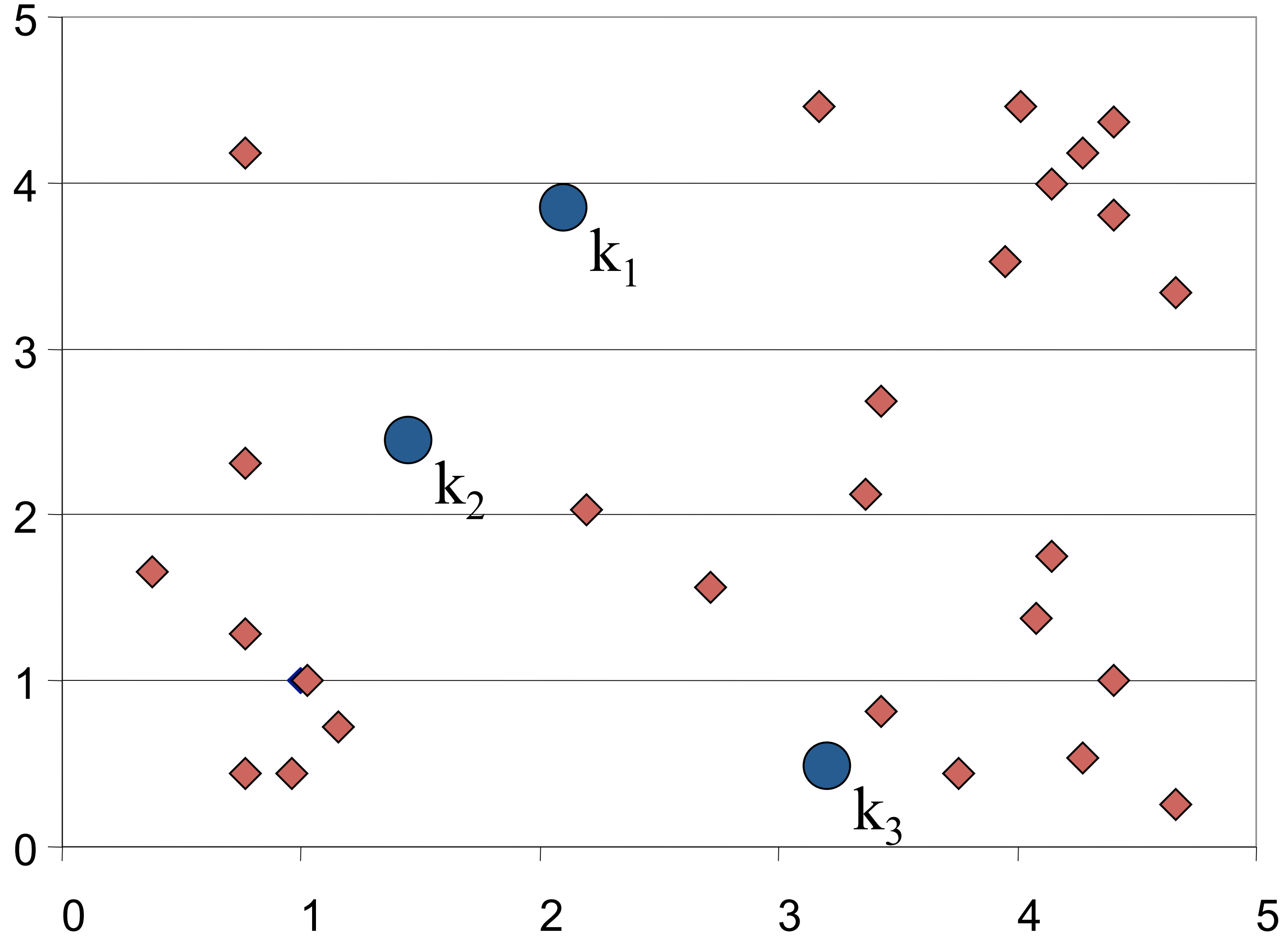
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

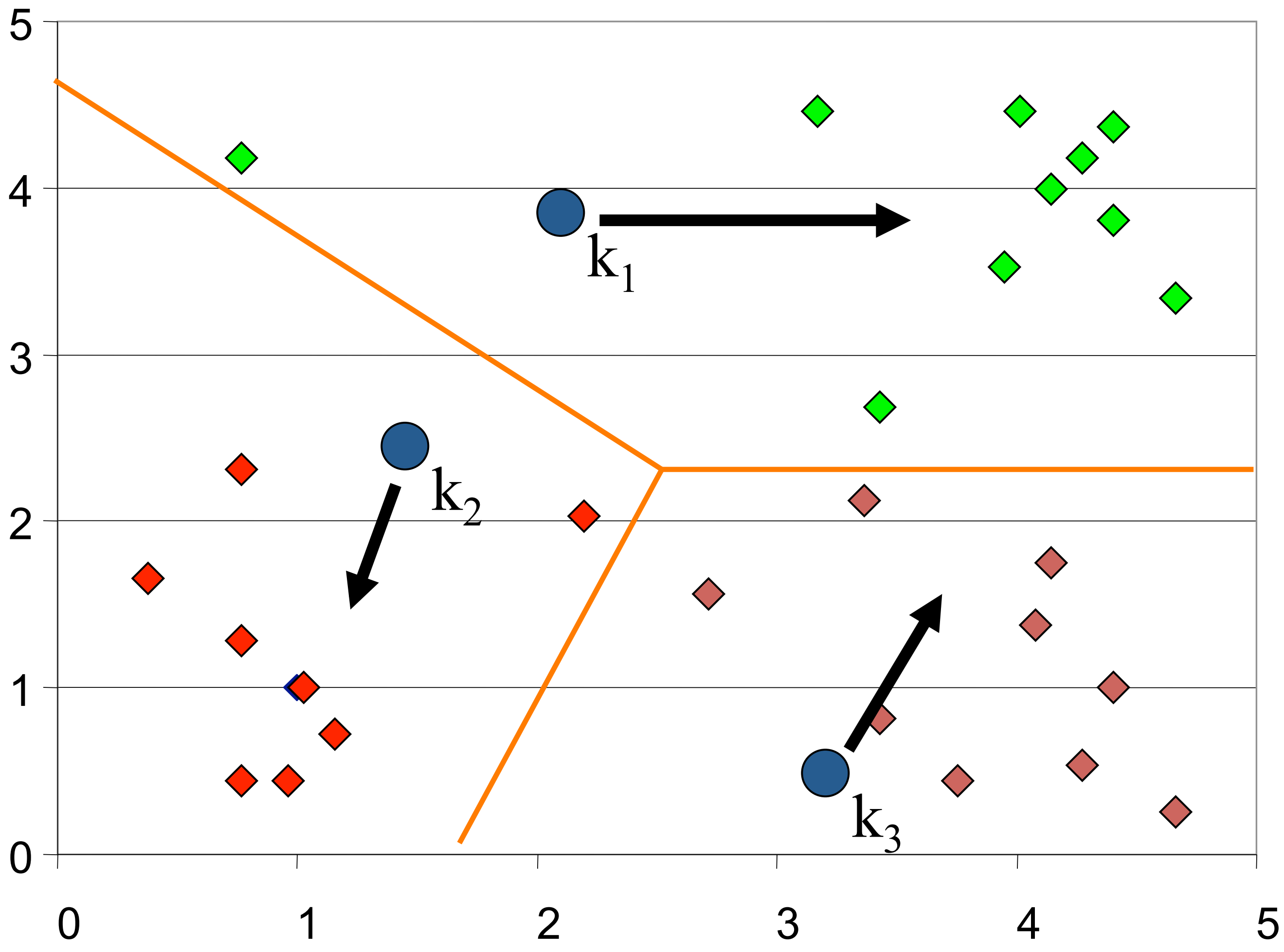
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 2

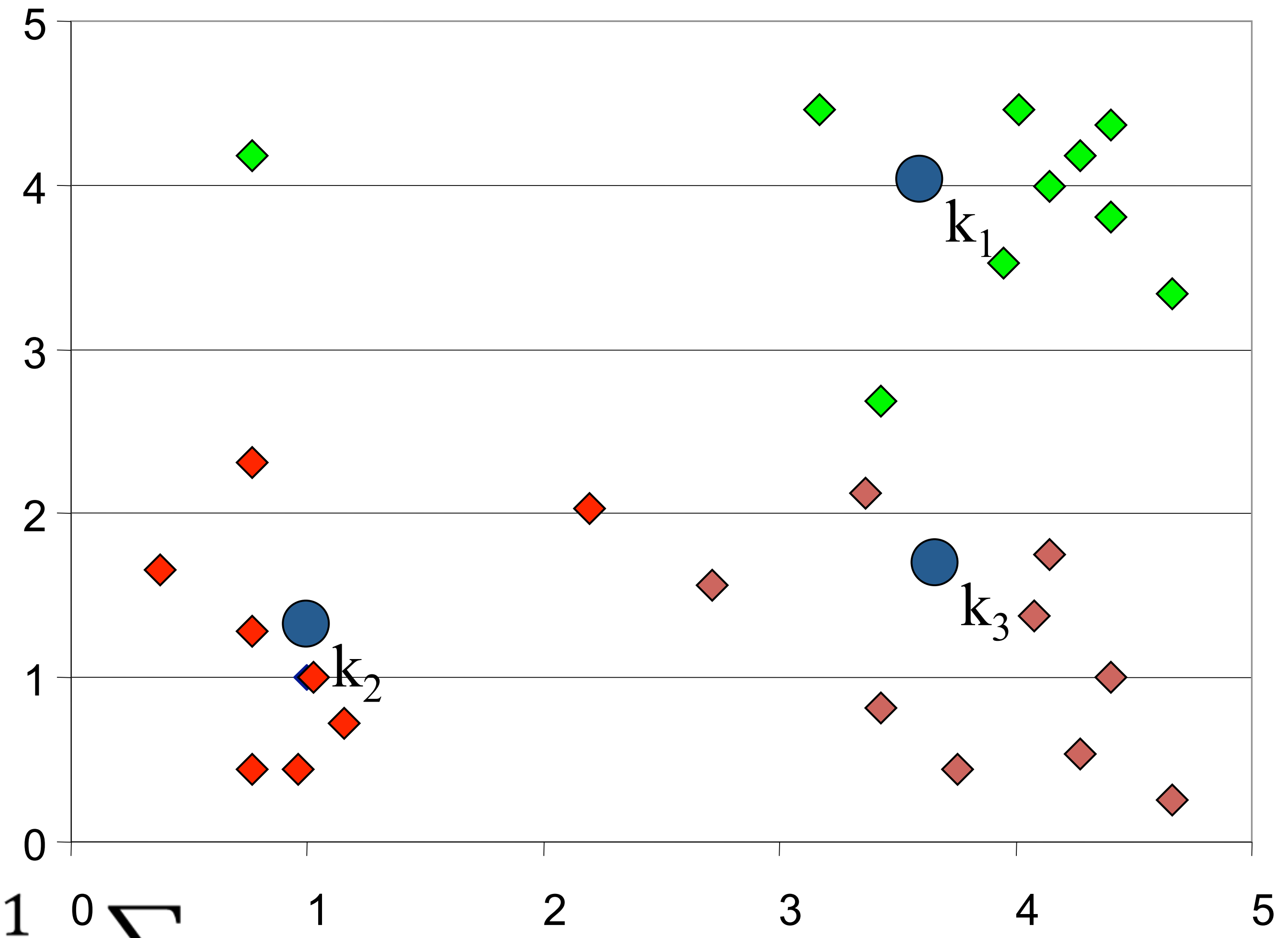
Algorithm: k-means, Distance Metric: Euclidean Distance



$$S_i^t = \left\{ xj : \left\| xj - k_i^t \right\| \le \| xj - k_r^t \| \; for\ all\ r = 1..k, r \neq i \right\}$$

K-means Clustering: Step 3

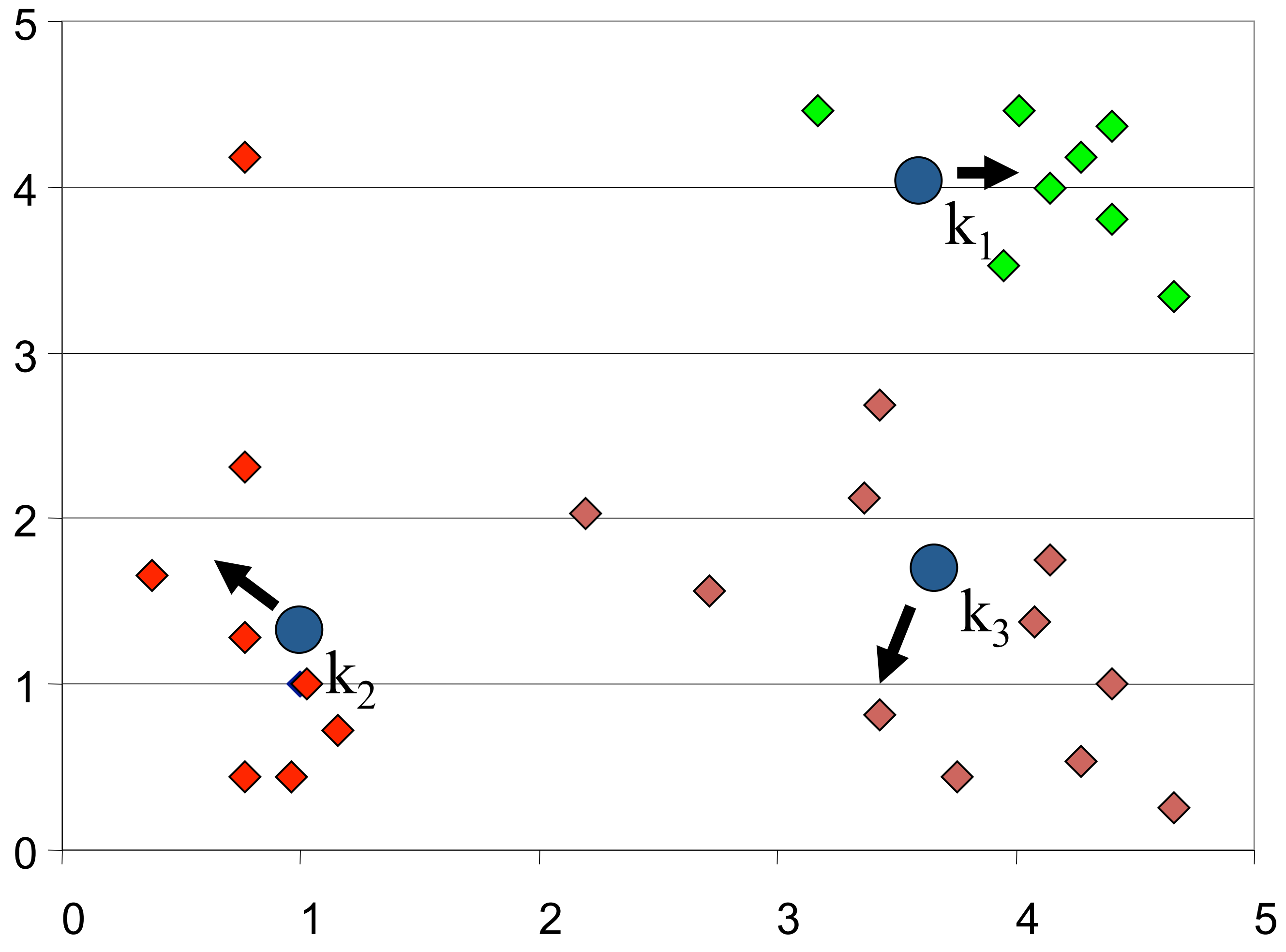
Algorithm: k-means, Distance Metric: Euclidean Distance



$$k_i^{t+1} = \frac{1}{|S_i^t|} \sum_{xj \in S_i^t} x_j$$

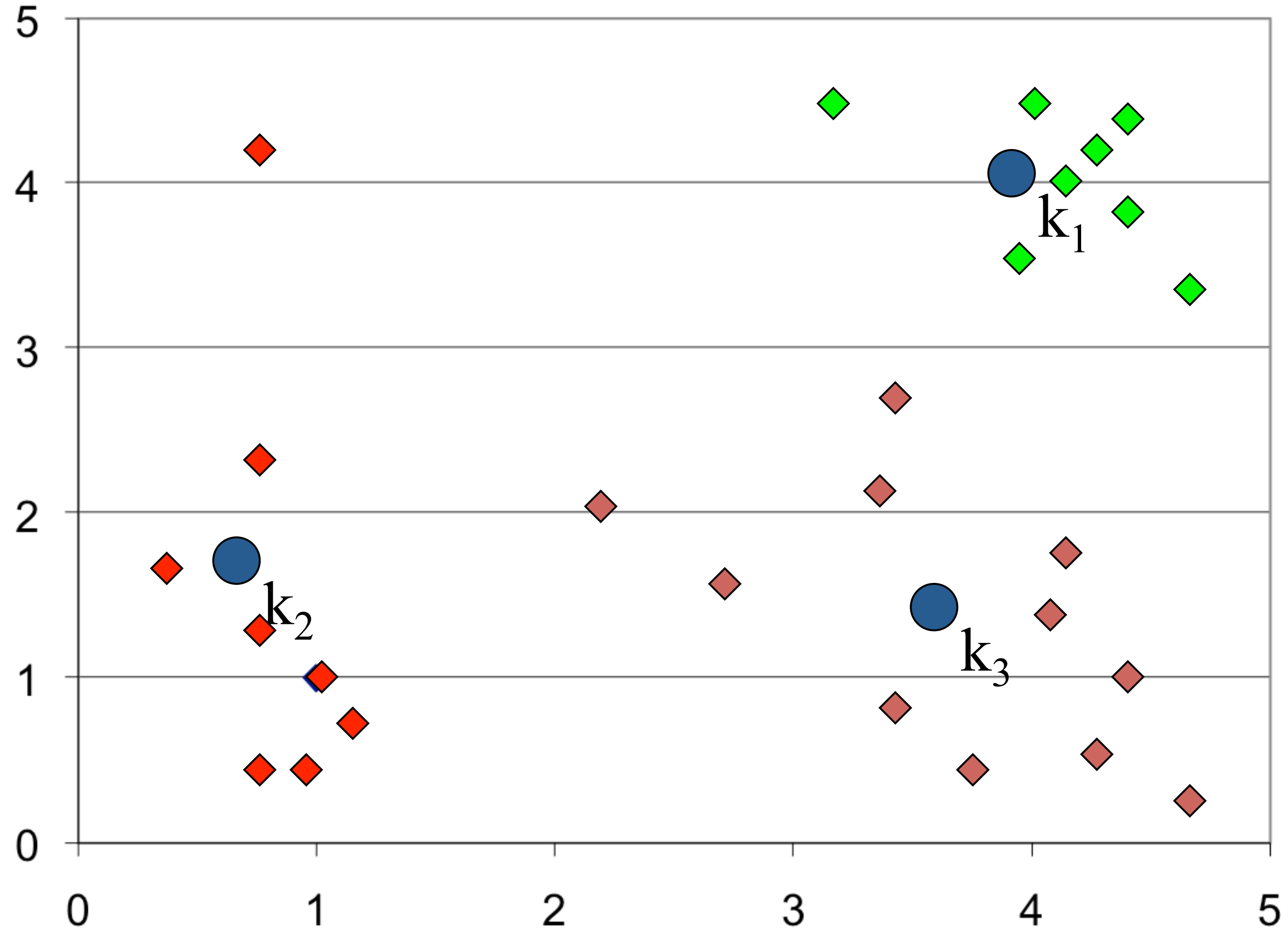
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



Comments on the *K-Means* Method

- <u>Strength</u>
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- <u>Weakness</u>
  - Applicable only when *mean* is defined, then what about categorical data? Need to extend the distance measurement.
    - Ahmad, Dey: **A k-mean clustering algorithm for mixed numeric and categorical data, Data & Knowledge Engineering, Nov. 2007**
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*
  - Tends to build clusters of equal size

EM Algorithm

Initialization: Choose means at random, etc.

E step: For all examples x_k:

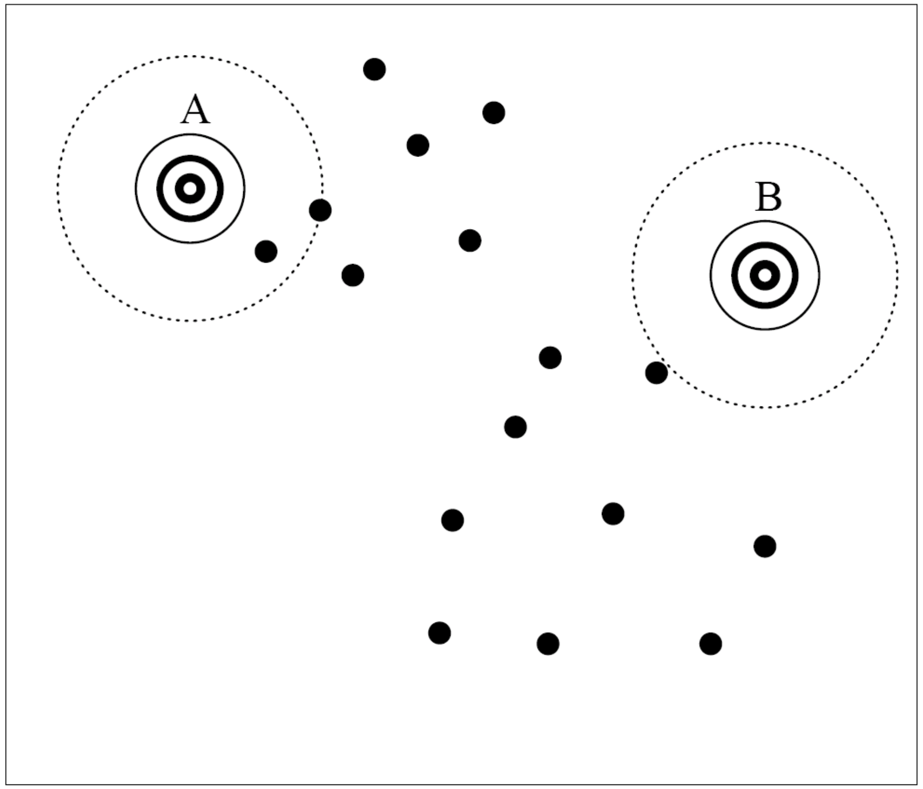
$$P(\mu_i|x_k) = \frac{P(\mu_i)P(x_k|\mu_i)}{P(x_k)} = \frac{P(\mu_i)P(x_k|\mu_i)}{\sum_{i'} P(\mu_{i'})P(x_k|\mu_{i'})}$$

M step: For all components c_i:

$$\begin{aligned} P(c_i) &= \frac{1}{n_e}\sum_{k=1}^{n_e} P(\mu_i|x_k) \\ \mu_i &= \frac{\sum_{k=1}^{n_e} x_k\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)} \\ \sigma_i^2 &= \frac{\sum_{k=1}^{n_e} (x_k - \mu_i)^2\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)} \end{aligned}$$

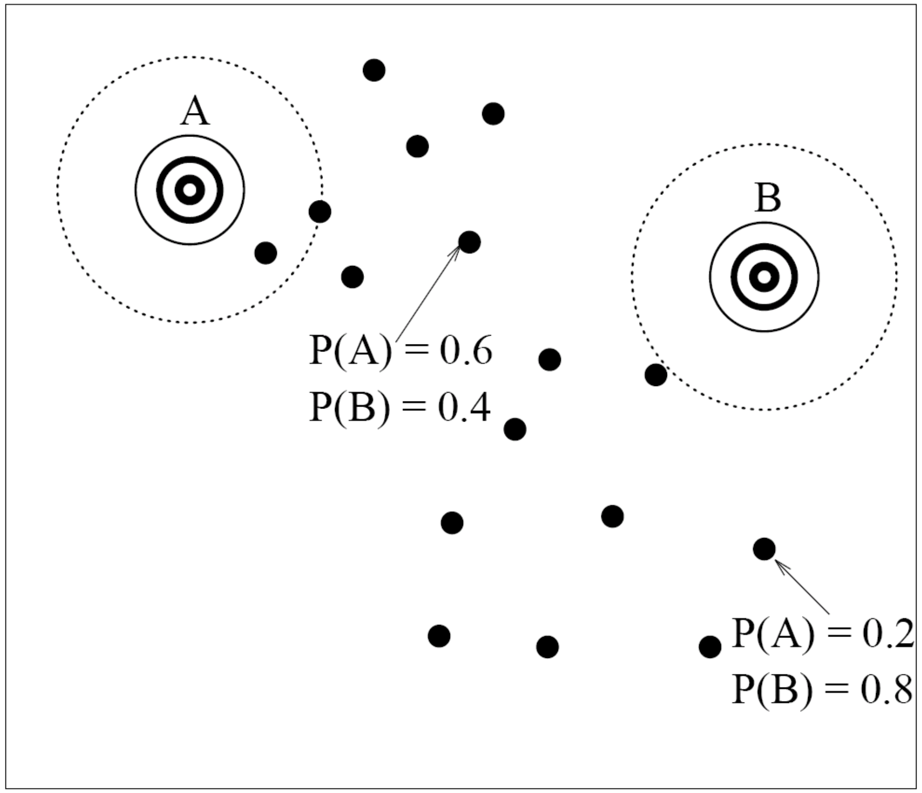
Processing : EM Initialization

– Initialization :
  - Assign random value to parameters



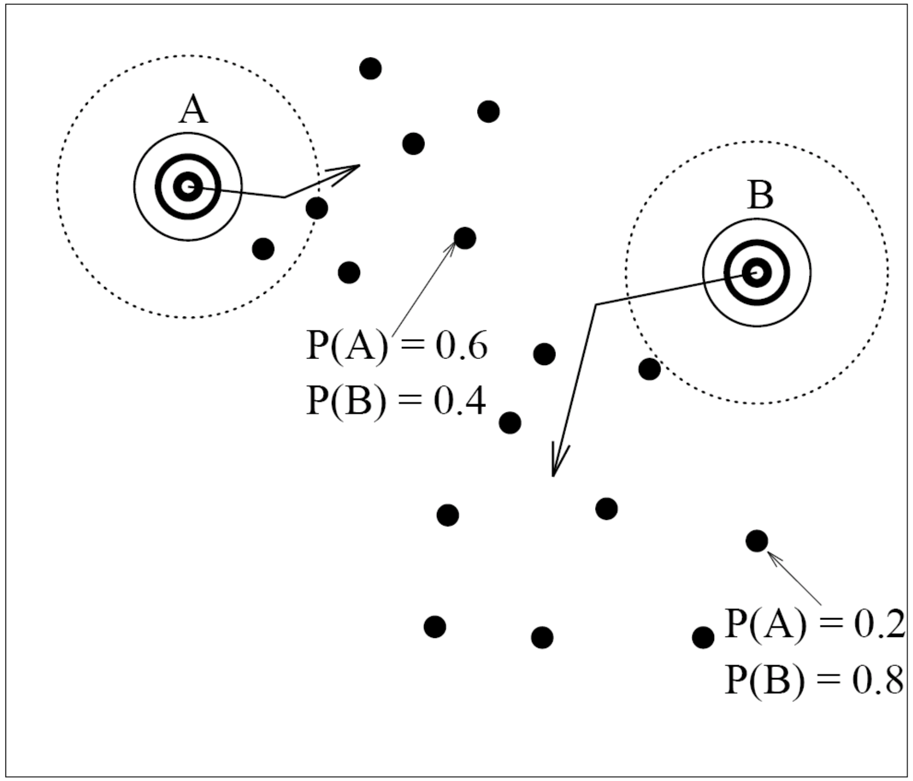
Processing : the E-Step

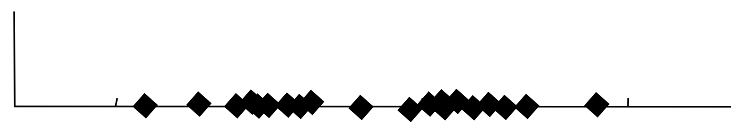
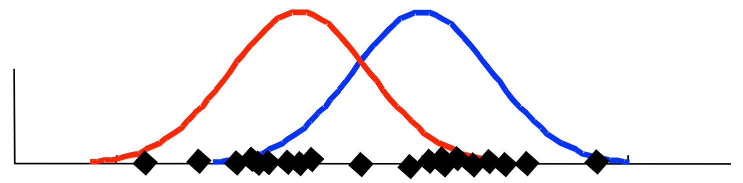
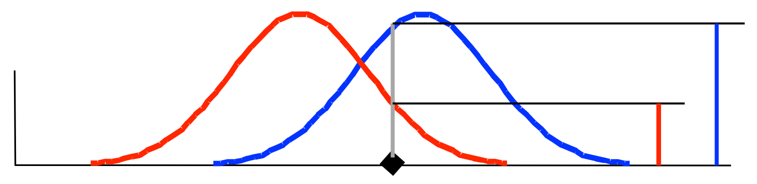
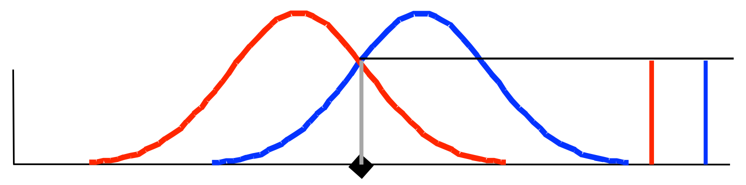
- Expectation :
  - Pretend to know the parameter
  - Assign data point to a component

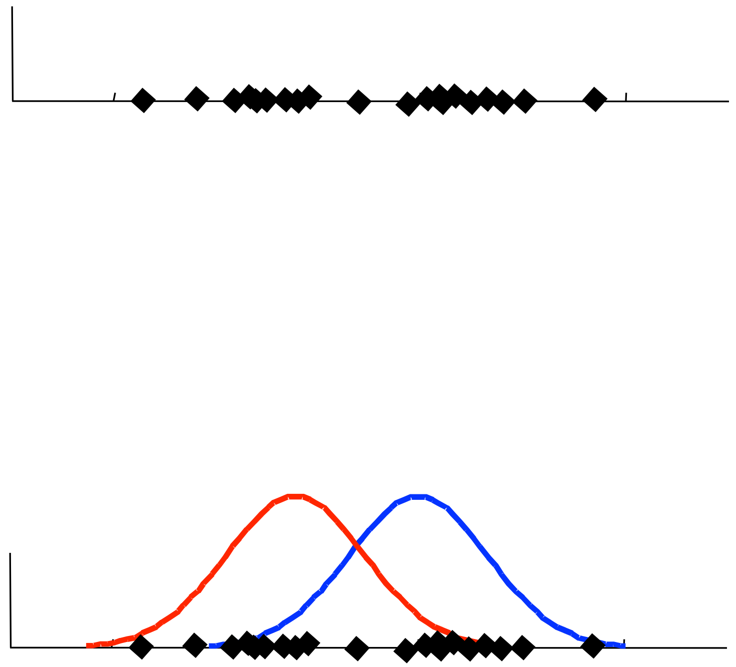


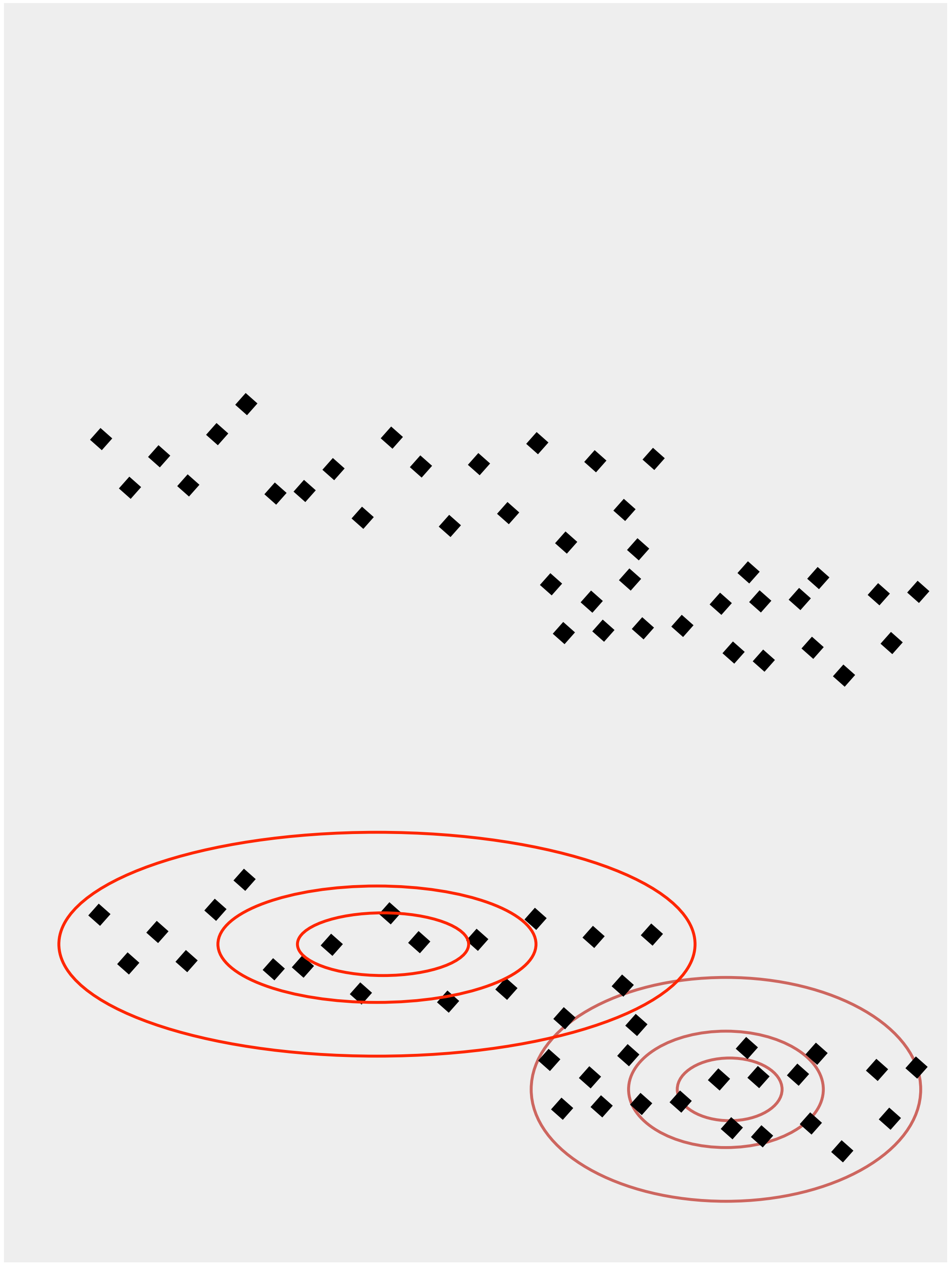
Processing : the M-Step (1/2)

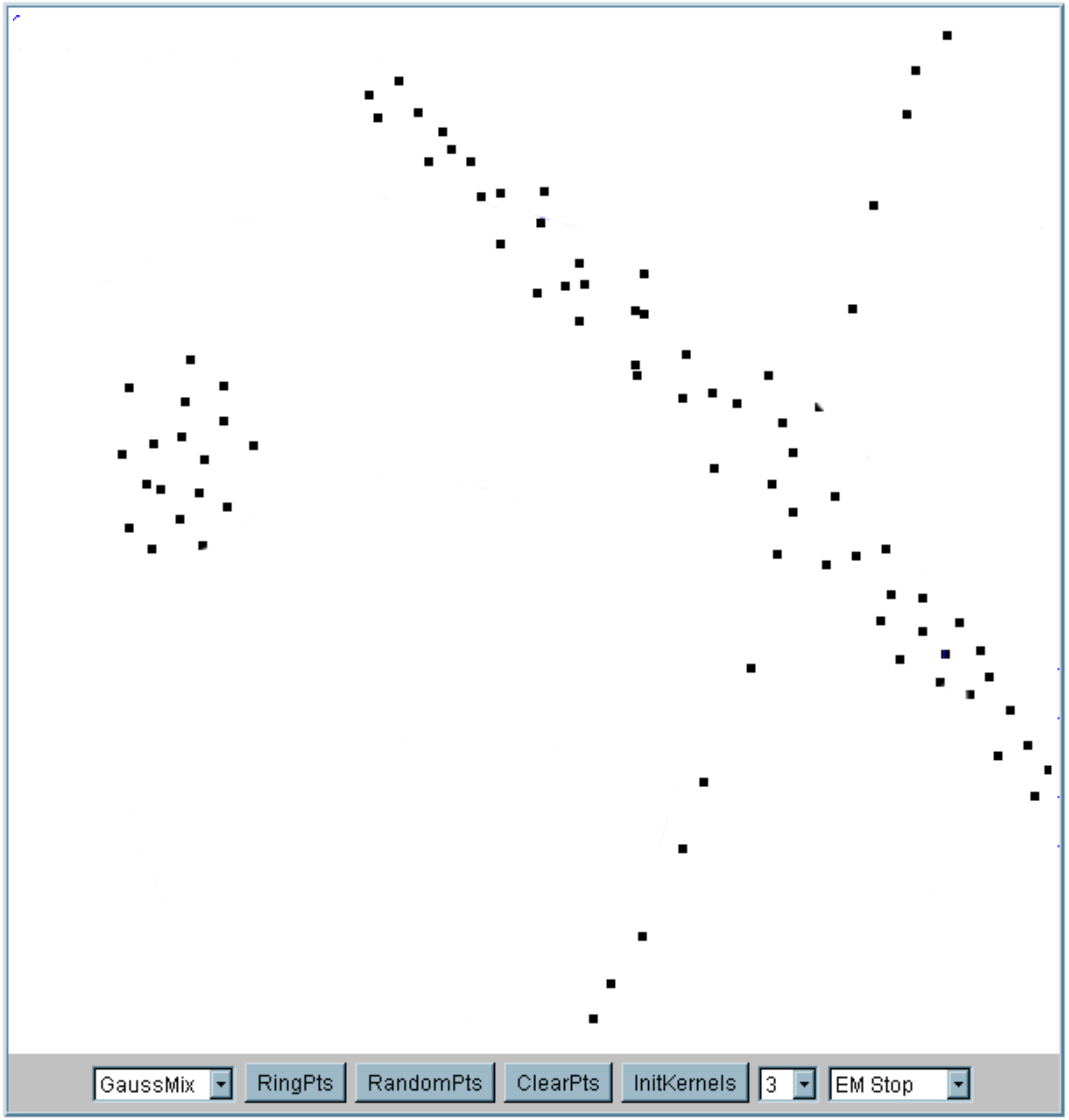
– Maximization :
  - Fit the parameter to its set of points

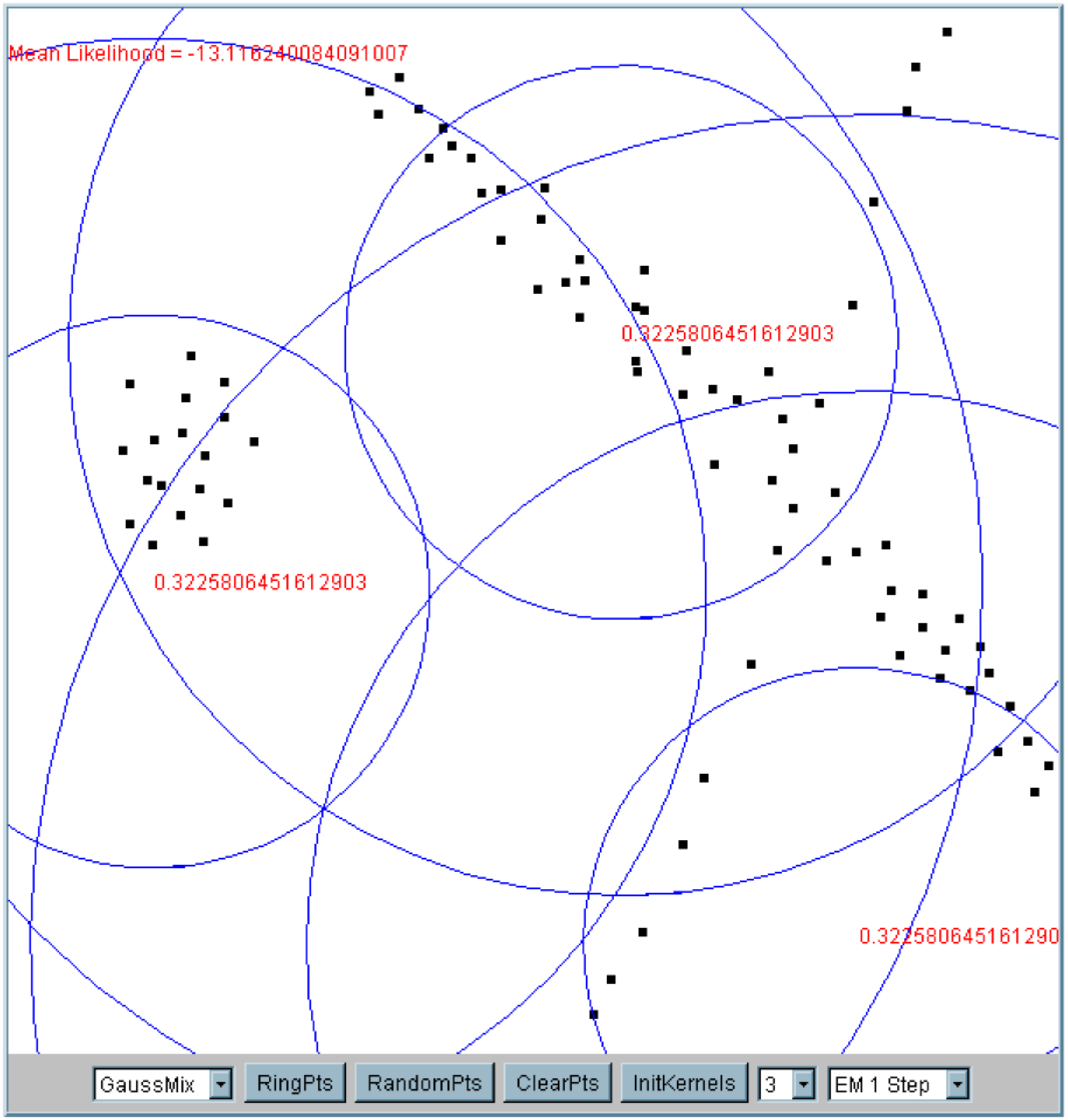







GaussMix
RingPts
RandomPts
ClearPts
InitKernels
3
EM Stop


Mean Likelihood = -13.116240084091007

0.3225806451612903

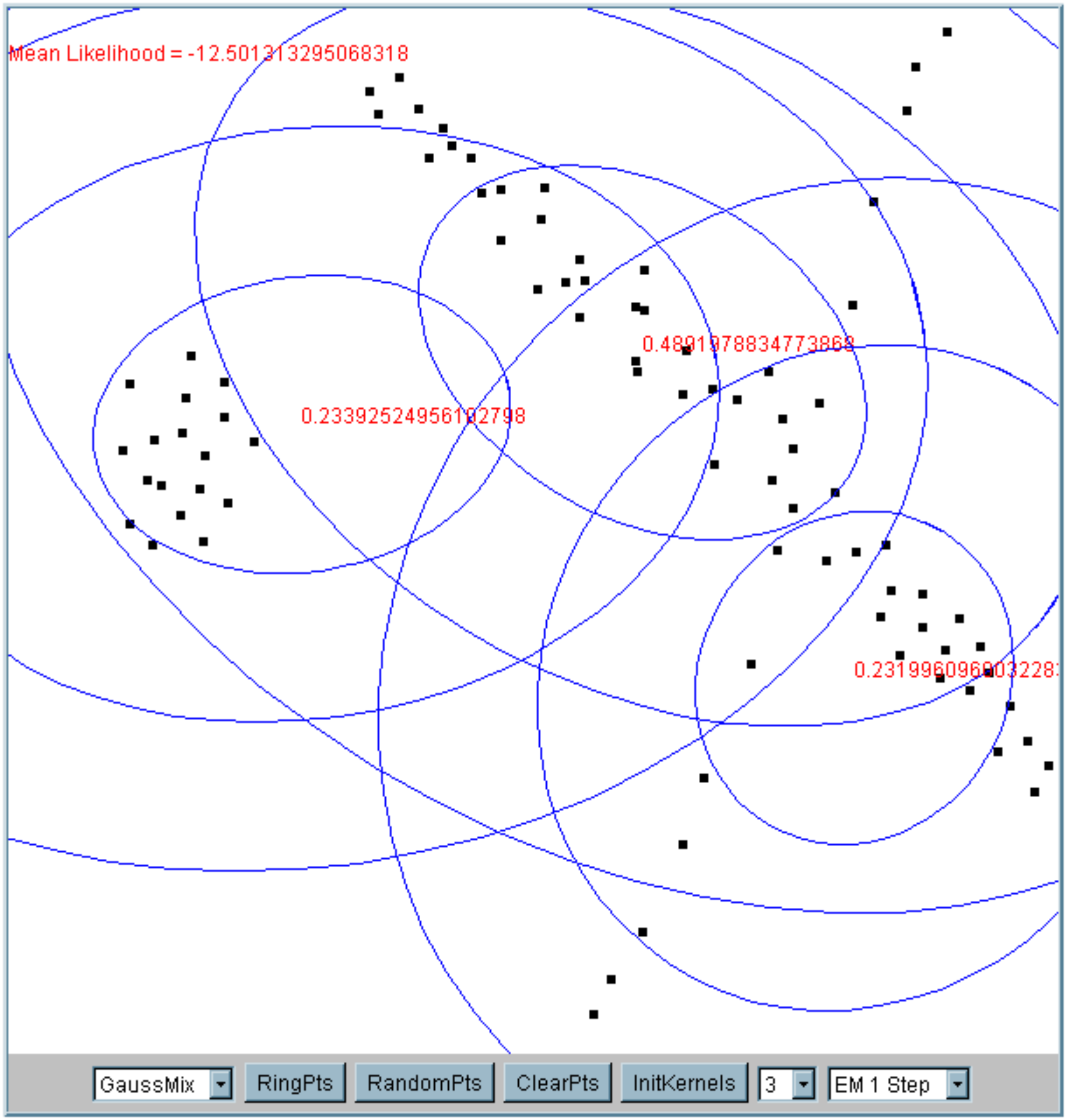
0.3225806451612903

0.322580645161290

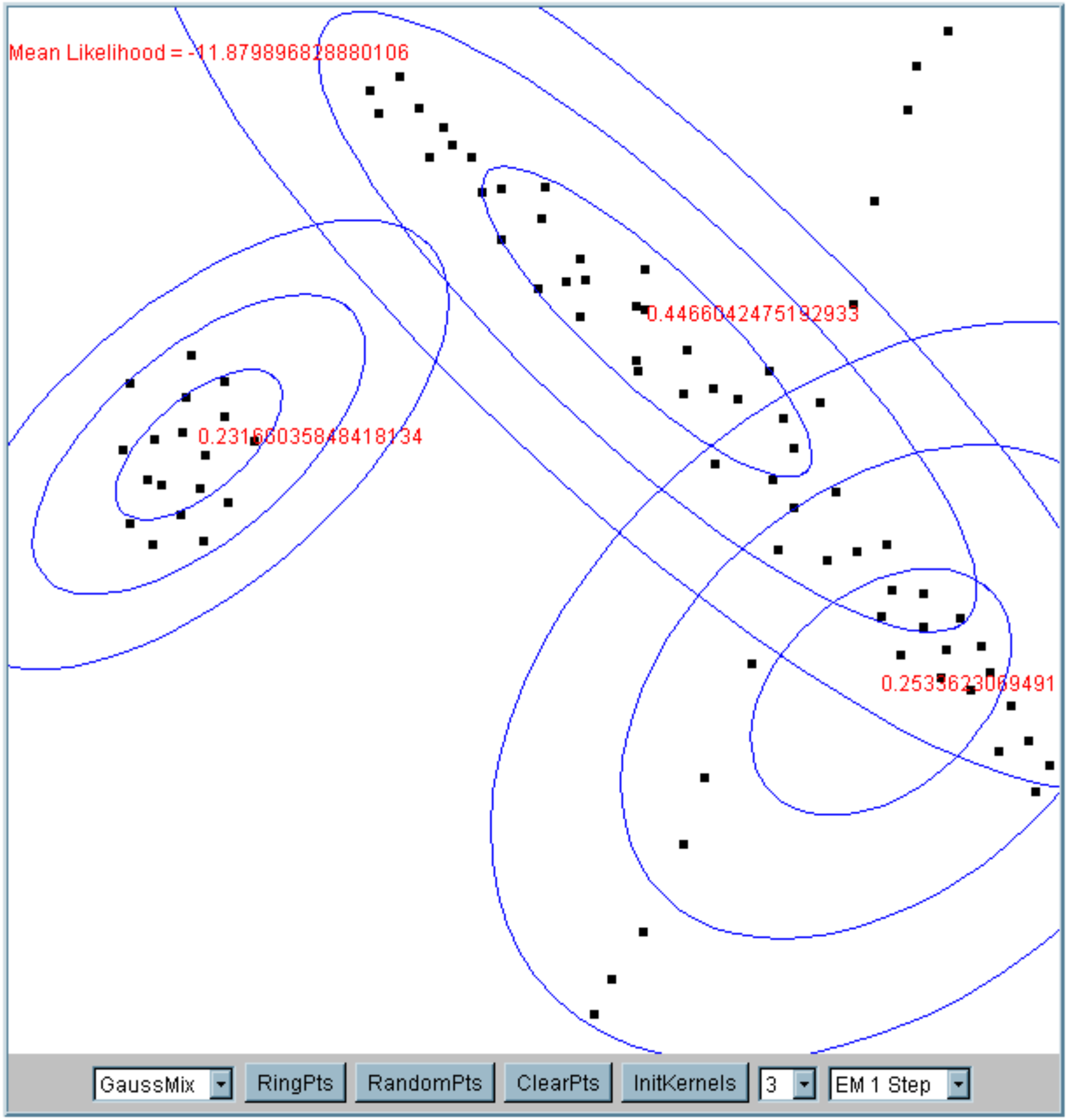
Iteration 1

The cluster means are randomly assigned

Iteration 2


Mean Likelihood = -12.501313295068318

Iteration 5

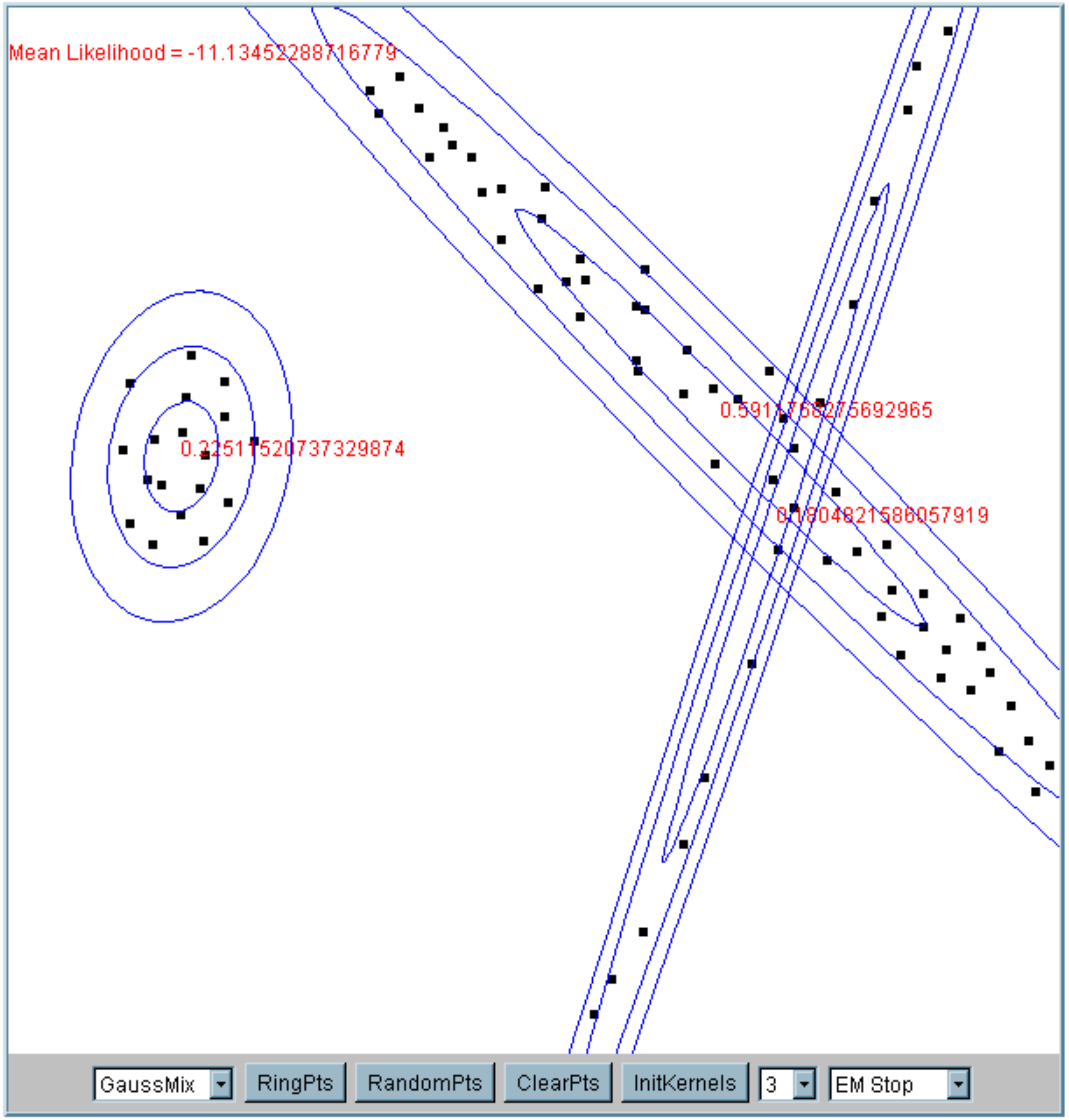

Mean Likelihood = -11.879896828880106

0.4466042475192933

0.2316503584841834

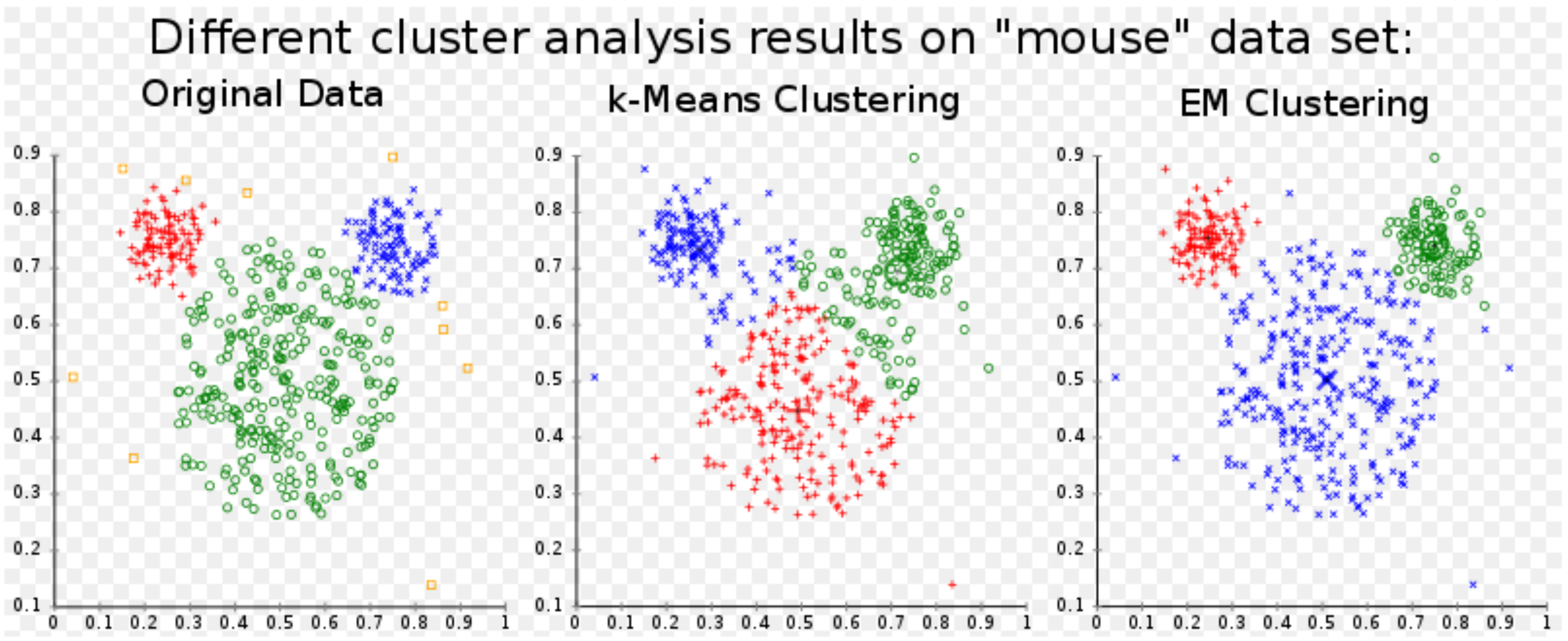
0.2535623069491

Iteration 25



Comments on the *EM*

- K-Means is a special form of EM
- EM algorithm maintains probabilistic assignments to clusters, instead of deterministic assignments, and multivariate Gaussian distributions instead of means
- Does not tend to build clusters of equal size



Source: http://en.wikipedia.org/wiki/K-means_algorithm

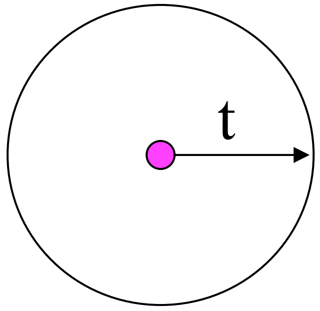
What happens if the data is streaming…

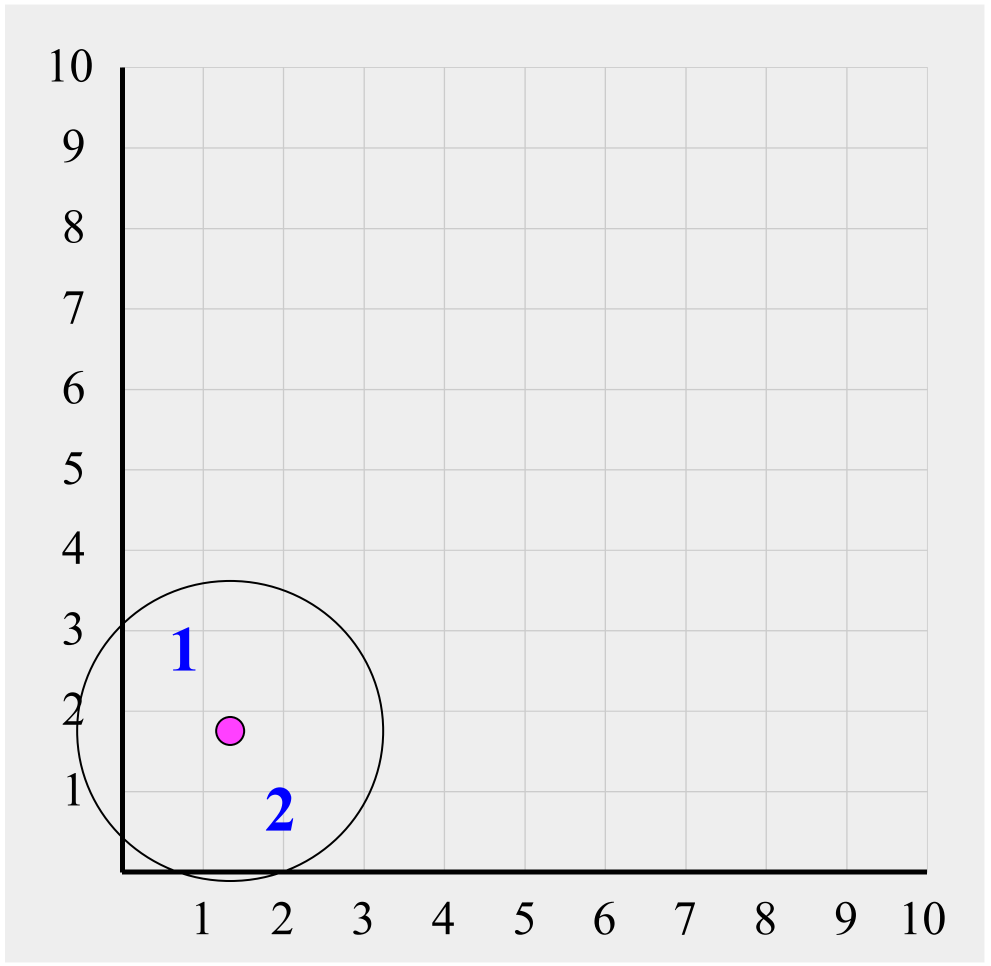
Nearest Neighbor Clustering

Not to be confused with Nearest Neighbor **Classification**

- Items are iteratively merged into the existing clusters that are closest.
- Incremental
- Threshold, t, used to determine if items are added to existing clusters or a new cluster is created.

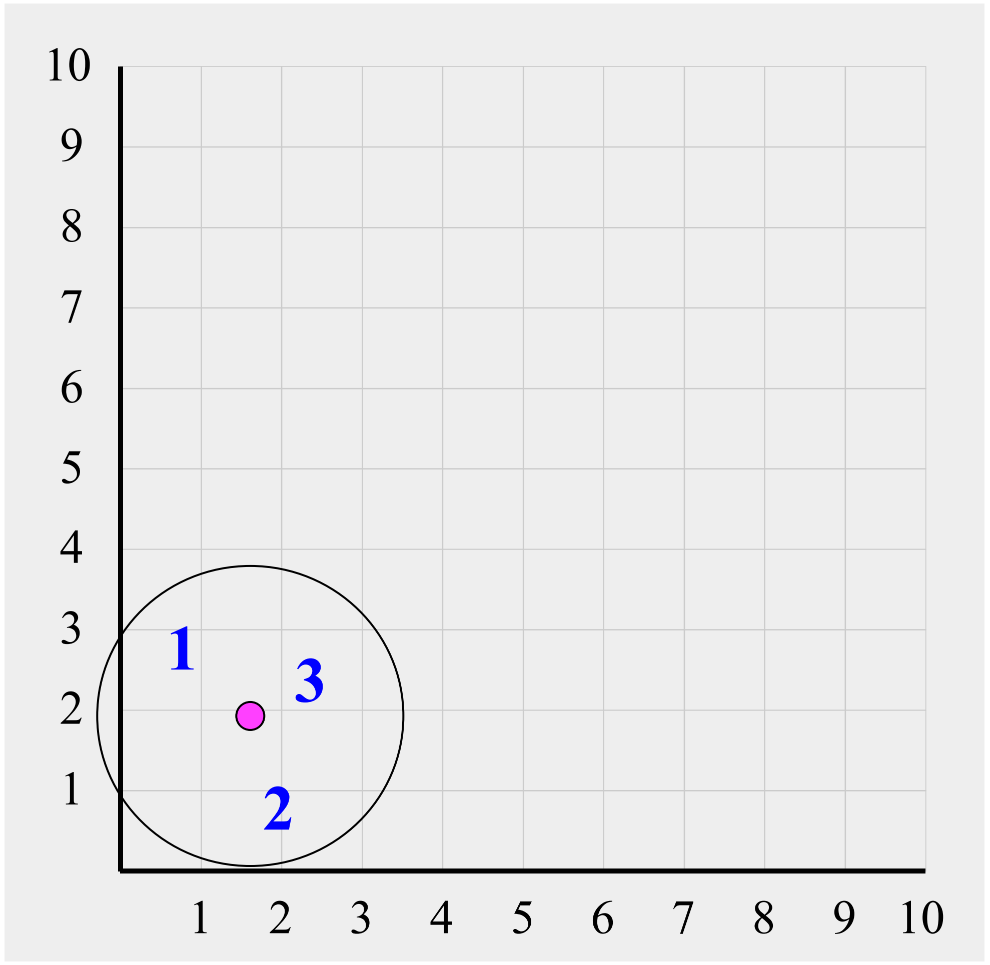
Threshold t





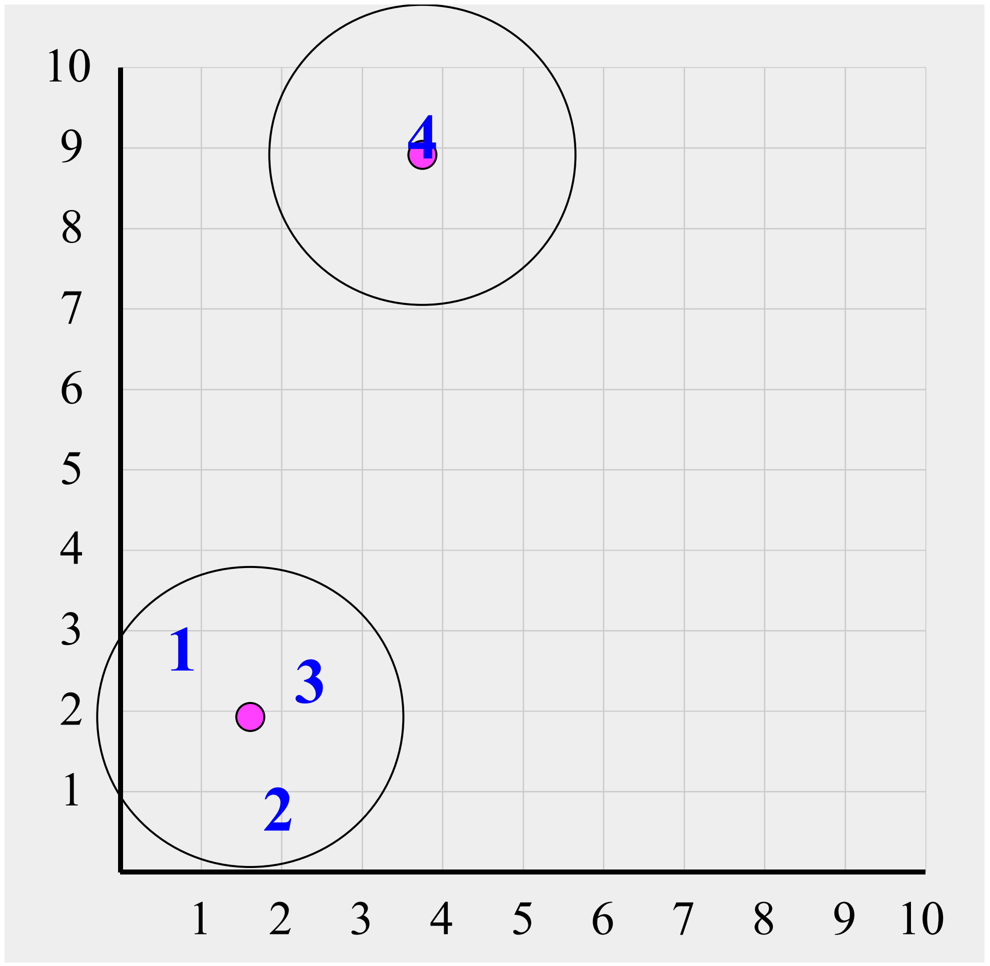
New data point arrives…

It is within the threshold for cluster 1, so add it to the cluster, and update cluster center.



New data point arrives…

It is **not** within the threshold for cluster 1, so create a new cluster, and so on..

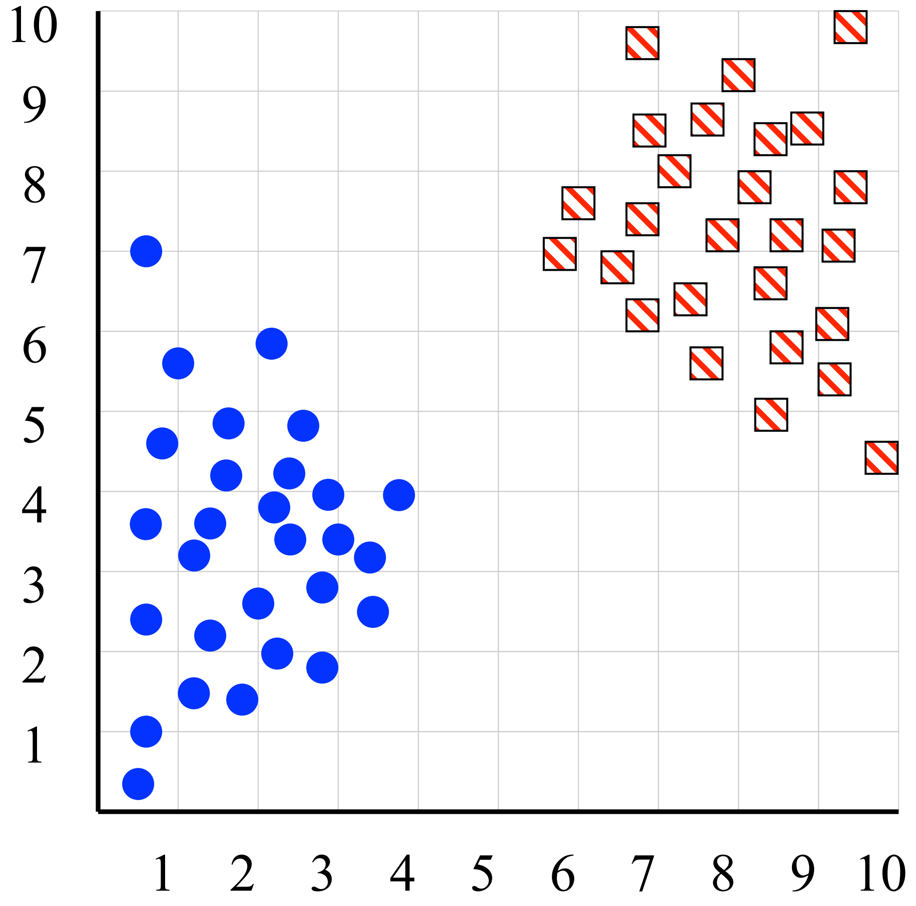


Algorithm is highly order dependent…

It is difficult to determine t in advance…

How can we tell the *right* number of clusters?

In general, this is a unsolved problem. However there are many approximate methods. In the next few slides we will see an example.



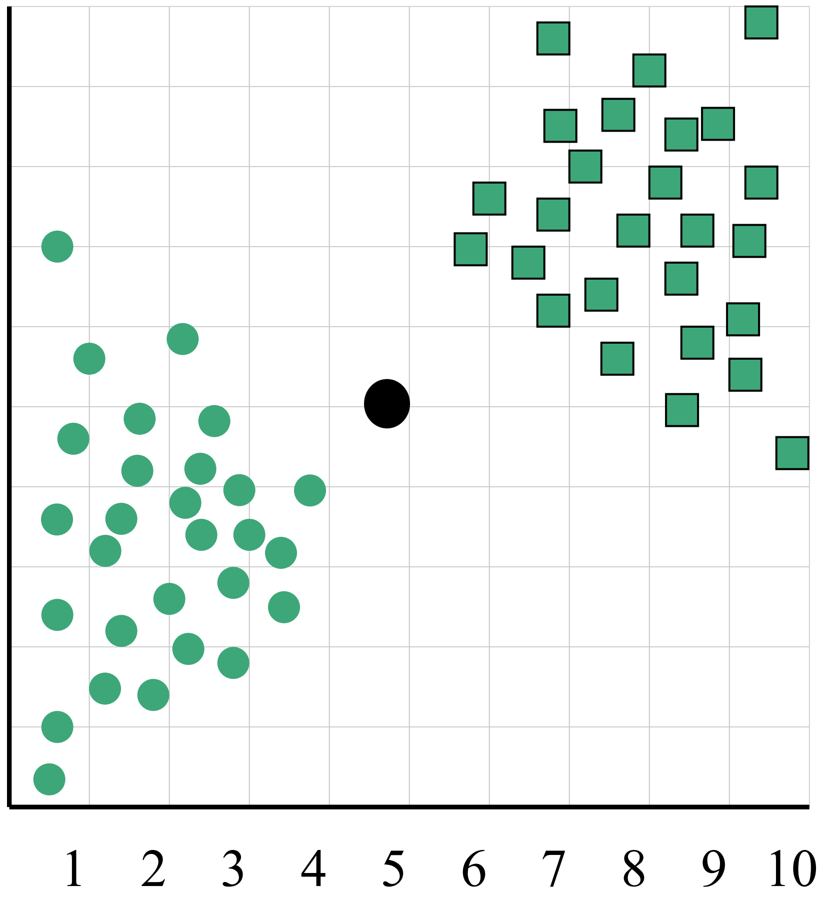
For our example, we will use the familiar katydid/grasshopper dataset.

However, in this case we are imagining that we do NOT know the class labels. We are only clustering on the X and Y axis values.

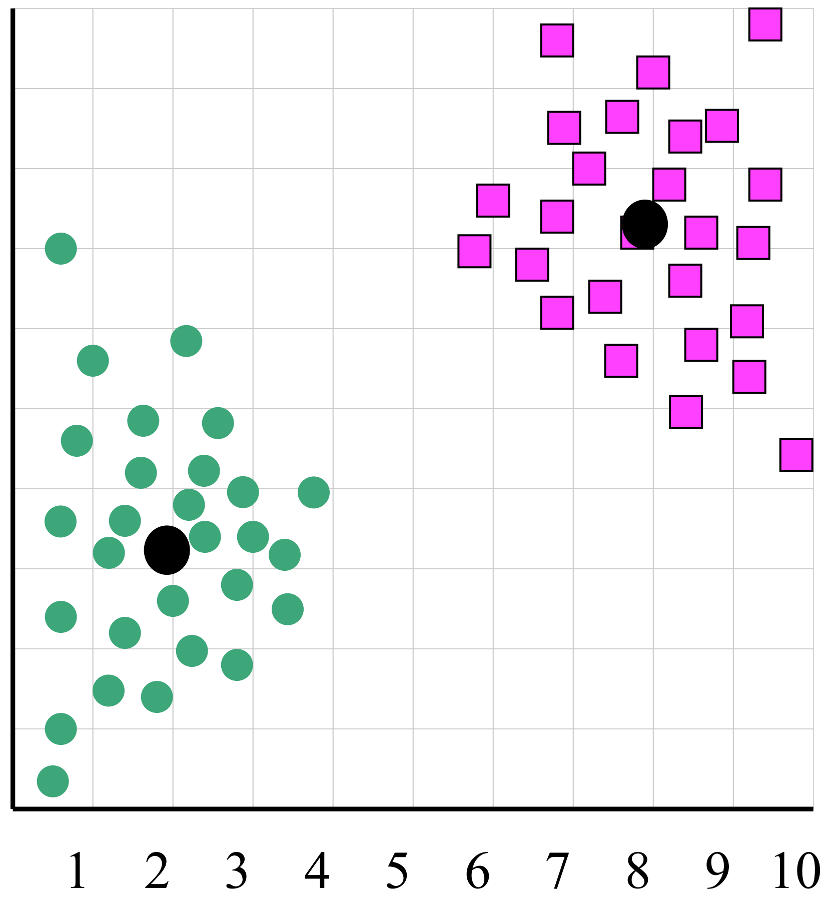
When k = 1, the objective function is 873.0

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

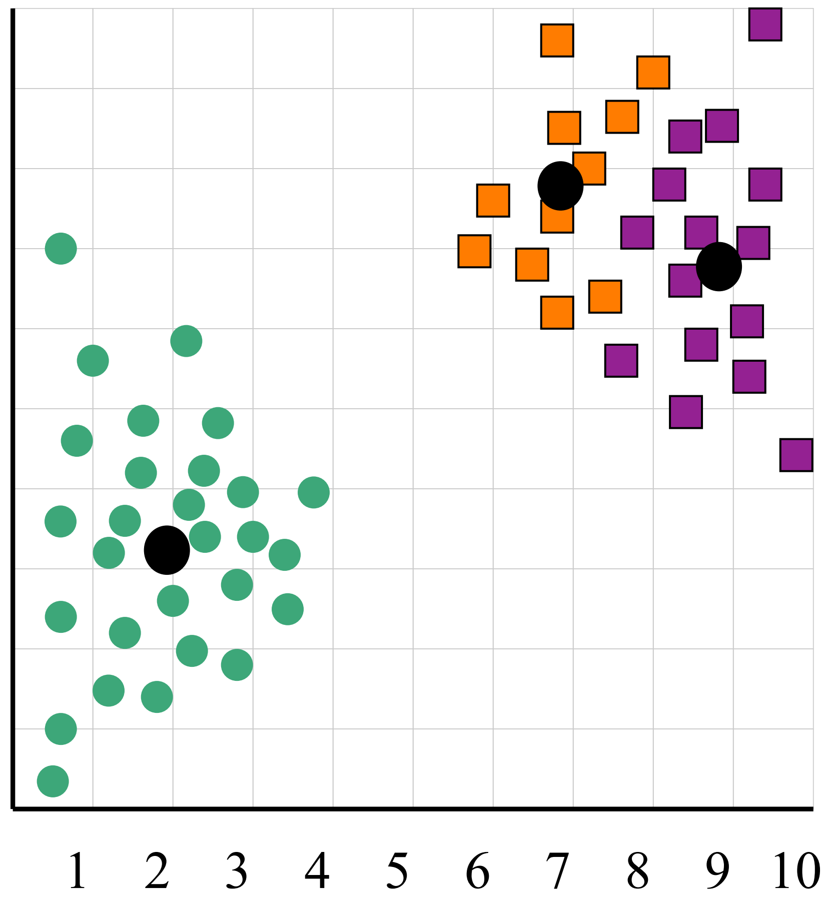
$$se_K = \sum_{j=1}^{k} se_{K_j}$$



When k = 2, the objective function is 173.1

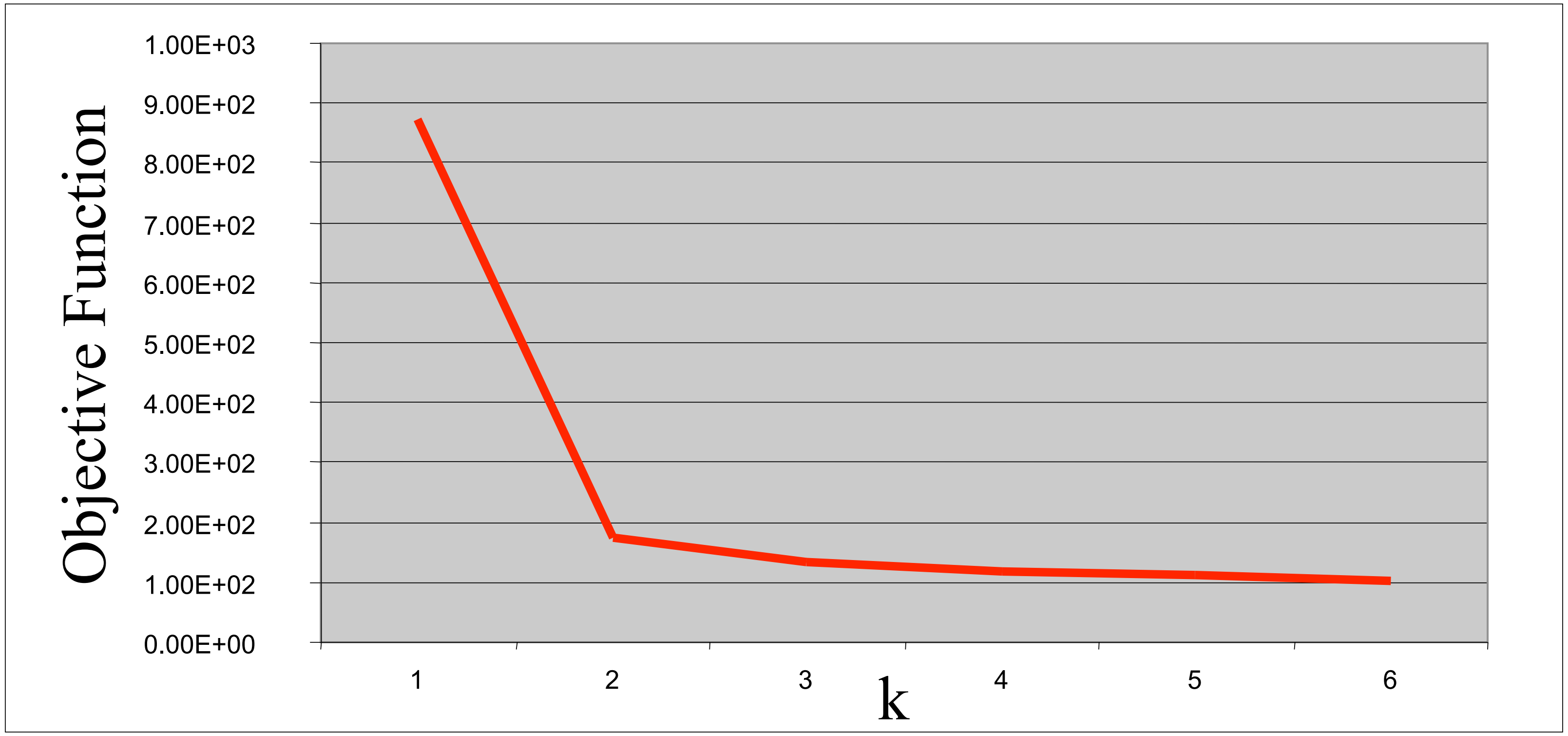


When k = 3, the objective function is 133.6



We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as "knee finding" or "elbow finding".



Note that the results are not always as clear cut as in this toy example

Goals

- Estimate class-conditional densities

$$p(\mathbf{x} \mid \omega_i)$$

- Estimate posterior probabilities

$$P(\omega_i \mid \mathbf{x})$$

Density Estimation

$$\boxed{E[K] = nP_{\mathcal{R}}}$$

Assume $p(\mathbf{x})$ is continuous & $\mathcal{R}$ is small

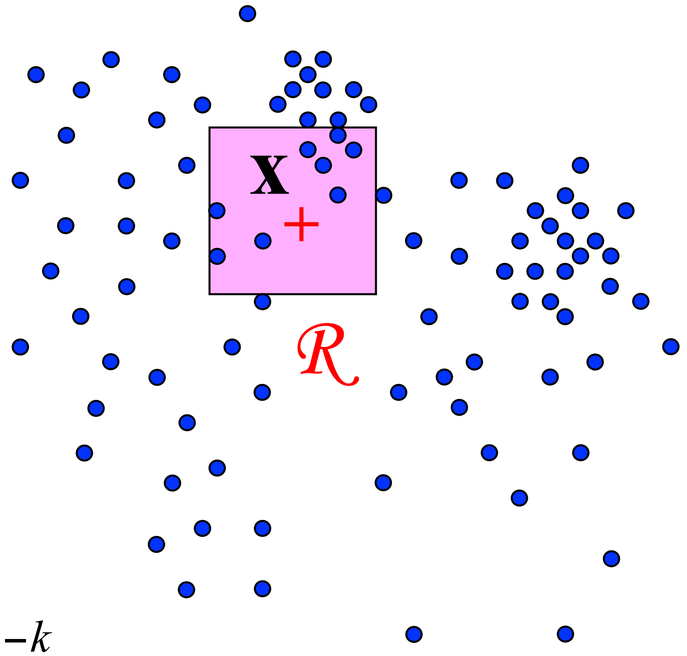
$$P(\mathbf{X} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}' = p(\mathbf{x})\int_{\mathcal{R}} d\mathbf{x}'$$

$$= p(\mathbf{x})V_{\mathcal{R}} \; \boxed{= P_{\mathcal{R}}}$$

Randomly take n samples, let K denote the number of samples inside $\mathcal{R}$.

$$\Rightarrow K \sim B(n, P_{\mathcal{R}})$$

$$P(K = k) = \binom{n}{k} P_{\mathcal{R}}{}^{k}(1 - P_{\mathcal{R}})^{n-k}$$



n samples

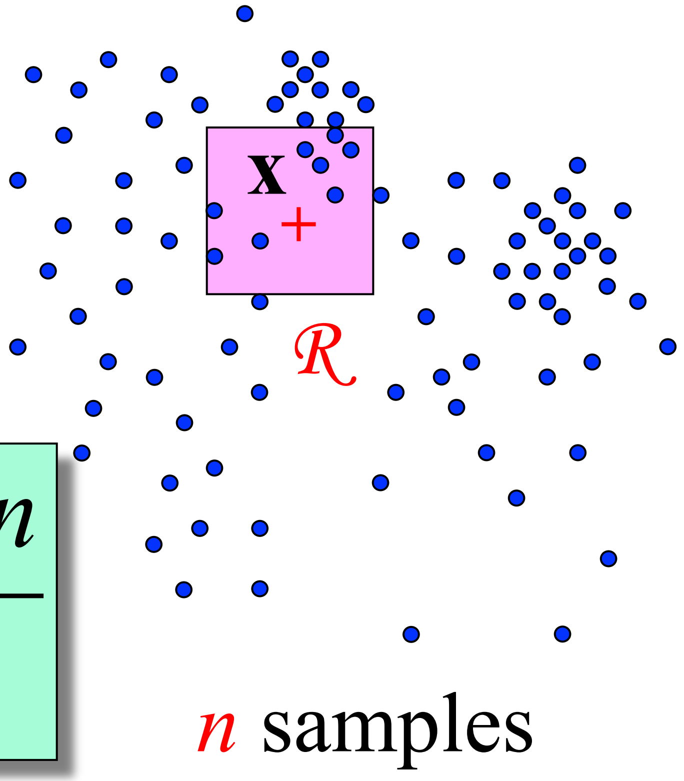
Density Estimation

$$\boxed{E[K] = nP_{\mathcal{R}}}$$

Assume $p(\mathbf{x})$ is continuous & $\mathcal{R}$ is small

$$P(\mathbf{X} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}' = p(\mathbf{x})\int_{\mathcal{R}} d\mathbf{x}'$$

$$= p(\mathbf{x})V_{\mathcal{R}} \boxed{= P_{\mathcal{R}}}$$



Let $k_{\mathcal{R}}$ denote the number of samples in $\mathcal{R}$.

$$E[K] \approx k_{\mathcal{R}}$$

$$P_{\mathcal{R}} \approx k_{\mathcal{R}} / n$$

$$\boxed{p(\mathbf{x}) \approx \frac{k_{\mathcal{R}} / n}{V_{\mathcal{R}}}}$$

Density Estimation

What items can be controlled?

How?

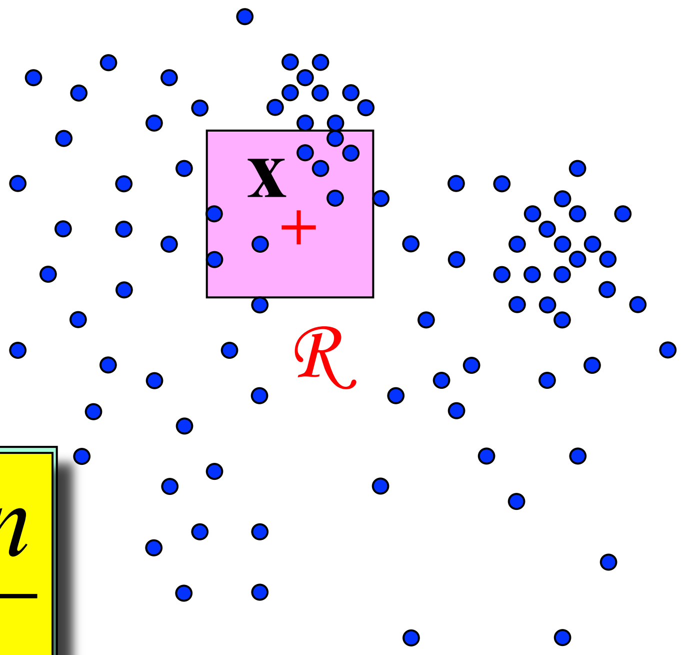
Use subscript n to take sample size into account.

We hope $\boxed{\lim_{n\to\infty} p_n(\mathbf{x}) = p(\mathbf{x})}$

To this, we should have

1. $\lim_{n\to\infty} V_n = 0$
2. $\lim_{n\to\infty} k_n = \infty$
3. $\lim_{n\to\infty} k_n / n = 0$

$$\boxed{p_n(\mathbf{x}) = \frac{k_n / n}{V_n}}$$



n samples

Two Approaches

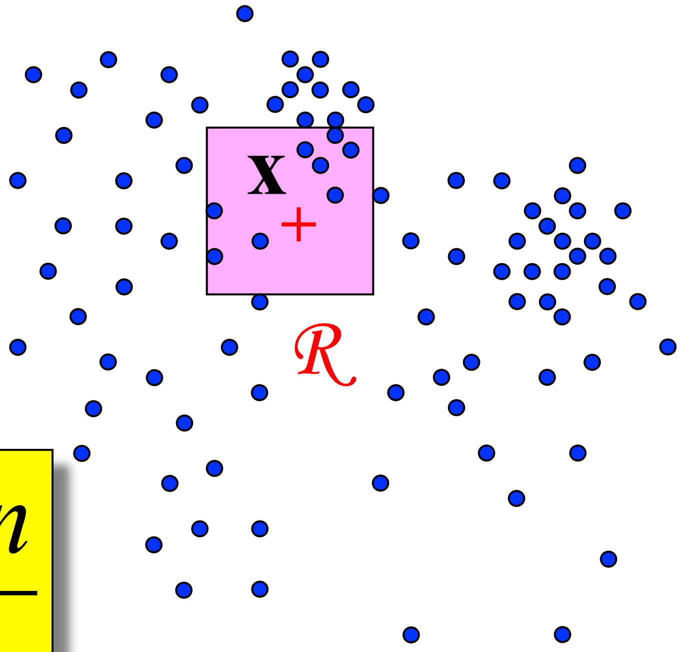
What items can be controlled?

How?

- Parzen Windows
  - Control V_n
- k_n-Nearest-Neighbor
  - Control k_n

1. $\lim_{n\to\infty} V_n = 0$
2. $\lim_{n\to\infty} k_n = \infty$
3. $\lim_{n\to\infty} k_n / n = 0$

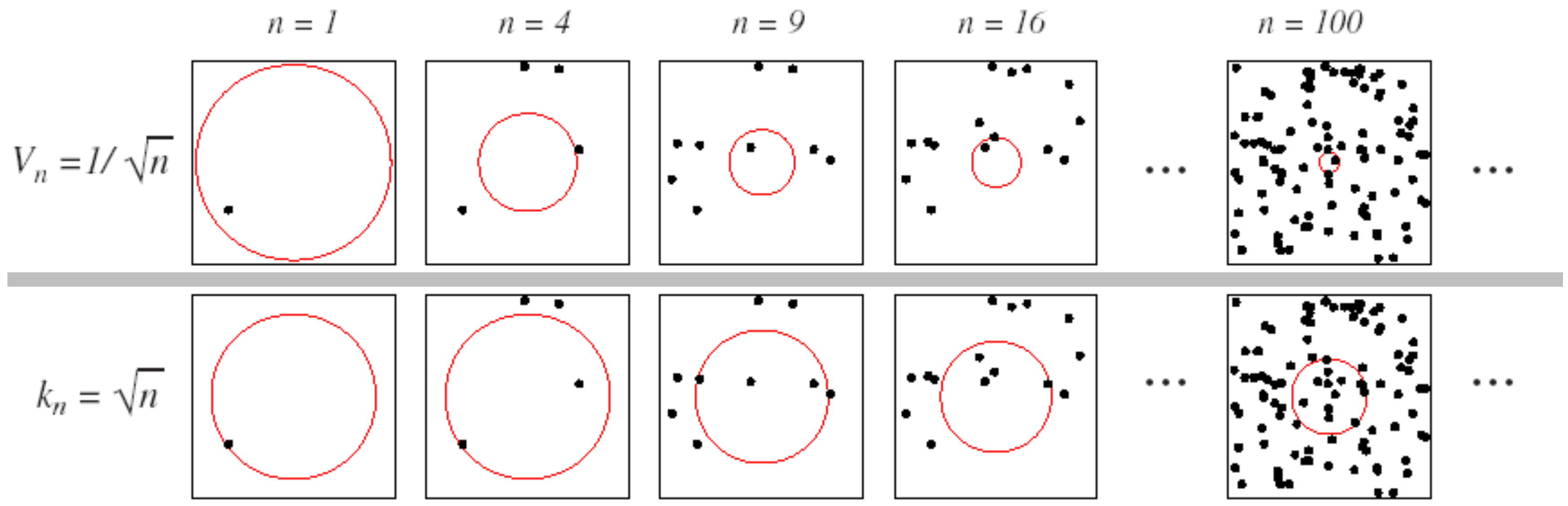
$$\boxed{p_n(\mathbf{x}) = \frac{k_n / n}{V_n}}$$



n samples

Two Approaches

Parzen Windows

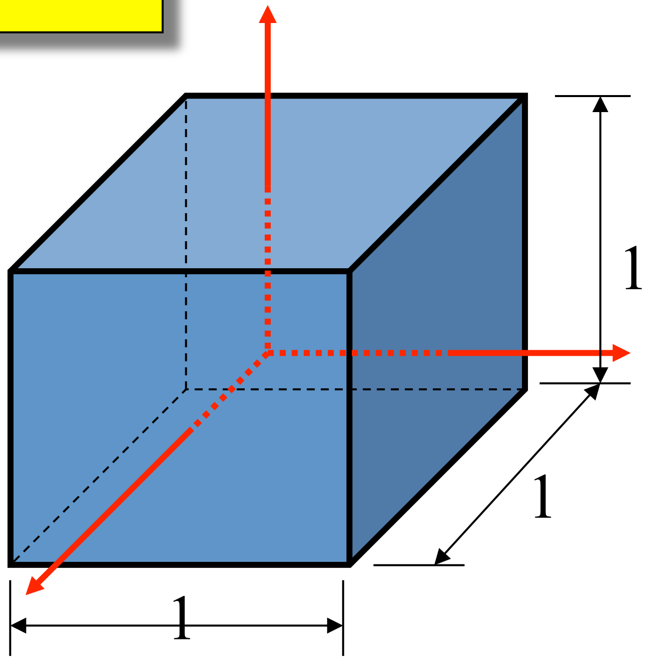


k_n-Nearest-Neighbor

Parzen Windows

$$\int \varphi(\mathbf{u})\,d\mathbf{u} = 1$$

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \le 1/2, \quad j = 1,2,\ldots,d \\ 0 & \text{otherwise} \end{cases}$$

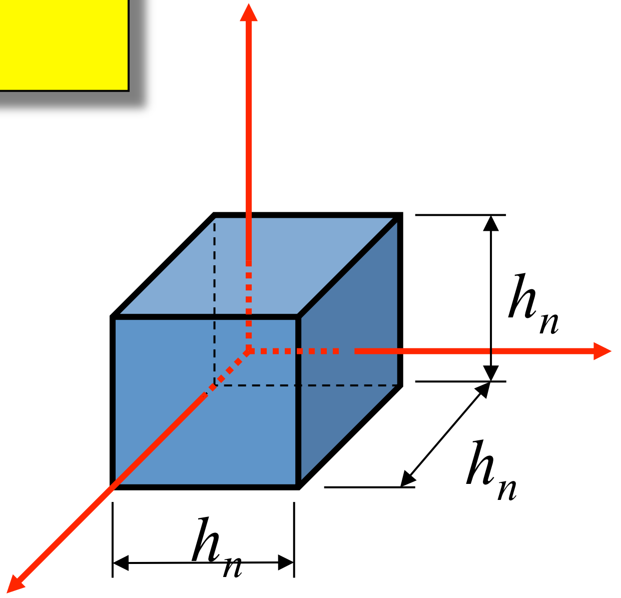


Window Function

$$\boxed{\int \varphi(\mathbf{u})d\mathbf{u} = 1}$$

$$\boxed{\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \le 1/2, \quad j = 1,2,\ldots,d \\ 0 & \text{otherwise} \end{cases}}$$

$$\varphi\left(\frac{\mathbf{x}}{h_n}\right) = \begin{cases} 1 & |x_j| \le h_n/2 \\ 0 & \text{otherwise} \end{cases}$$



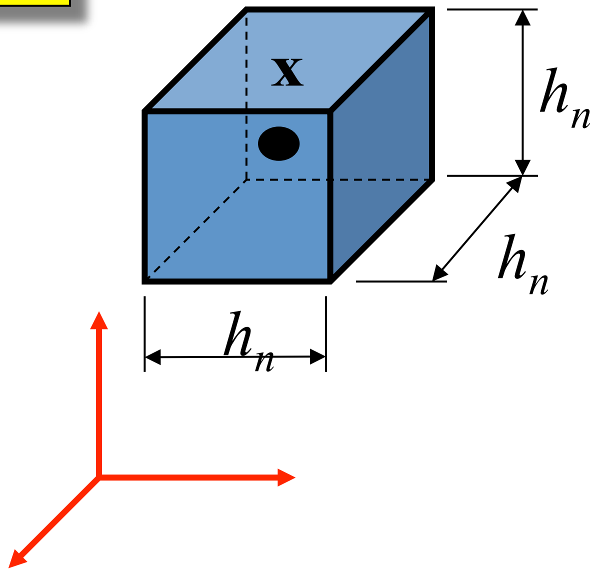
Window Function

$$\boxed{\int \varphi(\mathbf{u})d\mathbf{u} = 1}$$

$$\boxed{\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \le 1/2, \quad j = 1,2,\ldots,d \\ 0 & \text{otherwise} \end{cases}}$$

$$\varphi\left(\frac{\mathbf{x}}{h_n}\right) = \begin{cases} 1 & |x_j| \le h_n/2 \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi\left(\frac{\mathbf{x}-\mathbf{x}'}{h_n}\right) = \begin{cases} 1 & |x_j - x'_j| \le h_n/2 \\ 0 & \text{otherwise} \end{cases}$$



Parzen-Window Estimation

$$\boxed{\int \varphi(\mathbf{u})\,d\mathbf{u} = 1}$$

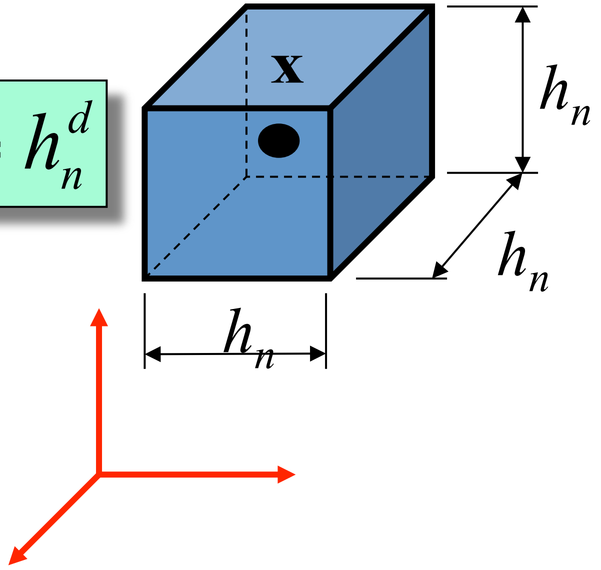
$$\boxed{\varphi\left(\frac{\mathbf{x}-\mathbf{x}'}{h_n}\right) = \begin{cases} 1 & |x_j - x'_j| \le h_n/2 \\ 0 & \text{otherwise} \end{cases}}$$

$$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

k_n: # samples inside hypercube centered at $\mathbf{x}$.

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

$$\boxed{V_n = h_n^d}$$



$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

Generalization

$$\boxed{\int \varphi(\mathbf{u})d\mathbf{u} = 1}$$

$$\boxed{\varphi\left(\frac{\mathbf{x}-\mathbf{x}'}{h_n}\right) = \begin{cases} 1 & |x_j - x'_j| \le h_n/2 \\ 0 & \text{otherwise} \end{cases}}$$

Requirement $\int p_n(\mathbf{x})d\mathbf{x} = 1$

Set $\mathbf{x}/h_n=\mathbf{u}$.

$$\int \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)d\mathbf{x} = \frac{1}{n}\sum_{i=1}^{n}\int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)d\mathbf{x} = \frac{1}{n}\sum_{i=1}^{n}\int \varphi(\mathbf{u})d\mathbf{u}$$

$$\Rightarrow \boxed{\int \varphi(\mathbf{u})d\mathbf{u} = 1}$$

The window is not necessarily a hypercube.

$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

h_n is a important parameter.

It depends on sample size.

Interpolation Parameter

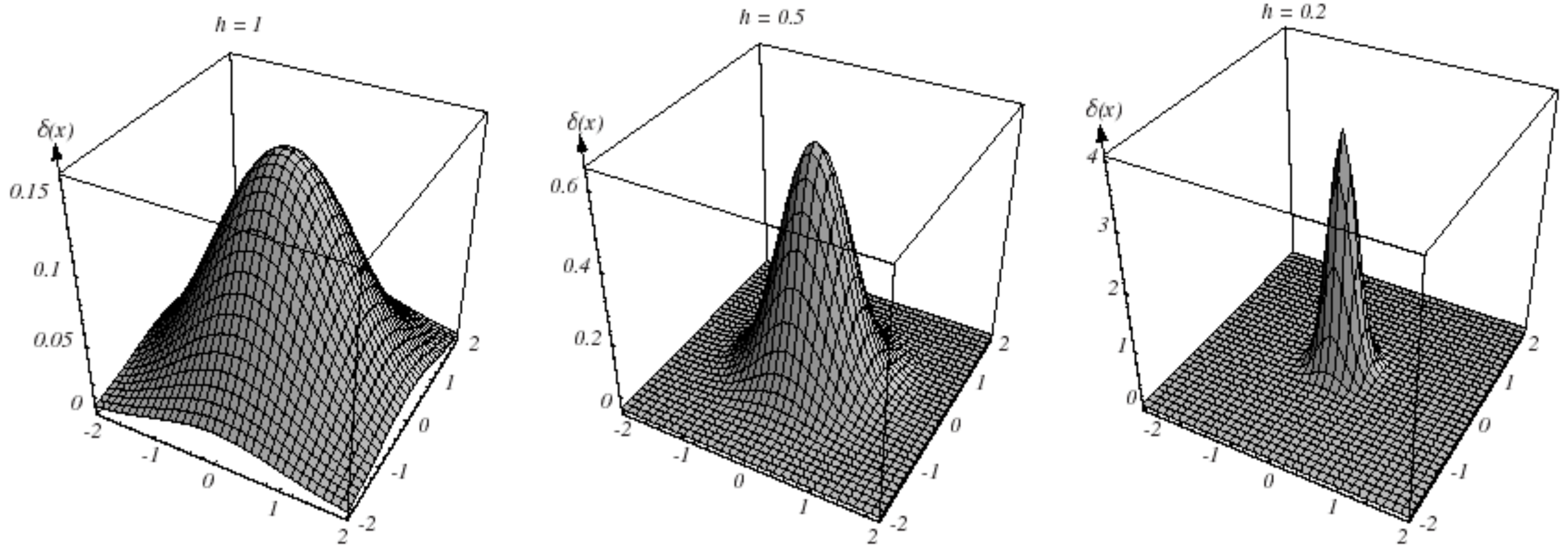
$$\int \varphi(\mathbf{u})\, d\mathbf{u} = 1$$

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \boxed{\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

$$\delta_n(\mathbf{x}) = \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$$

$h_n \rightarrow 0$

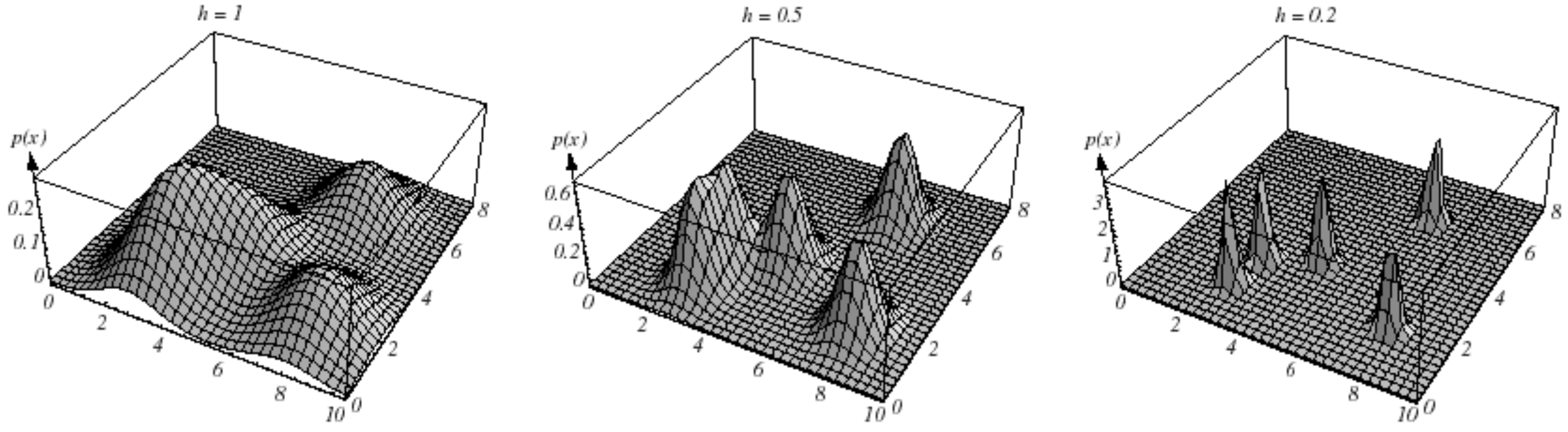
$\delta_n(\mathbf{x})$ is a *Dirac delta function.*



Example

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

Parzen-window estimations for five samples



Convergence Conditions

$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

To assure convergence, i.e.,

$$\boxed{\lim_{n\to\infty} E[p_n(\mathbf{x})] = p(\mathbf{x})} \quad \text{and} \quad \boxed{\lim_{n\to\infty} Var[p_n(\mathbf{x})] = 0}$$

we have the following additional constraints:

$$\sup_{\mathbf{u}} \varphi(\mathbf{u}) < \infty \qquad \lim_{n\to\infty} V_n = 0$$

$$\lim_{\|\mathbf{u}\|\to\infty} \varphi(\mathbf{u})\prod_{i=1}^{d} u_i = 0 \qquad \lim_{n\to\infty} nV_n = \infty$$

Illustrations

$$\boxed{X \sim N(0,1)}$$

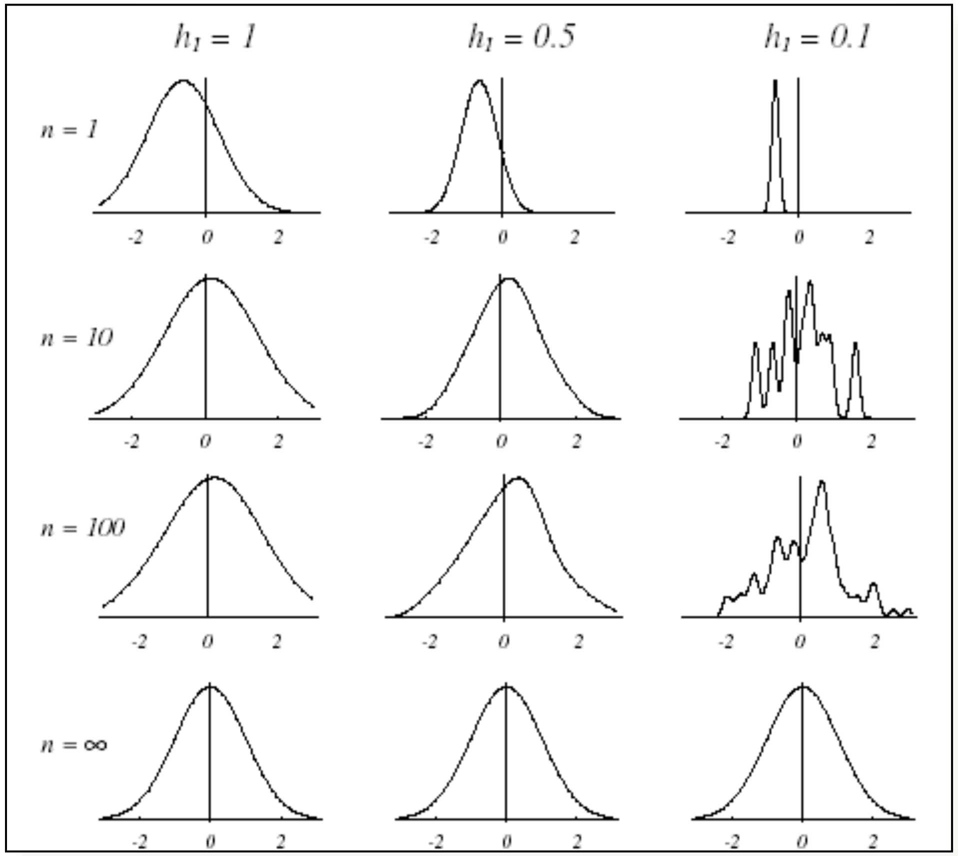
$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

One dimension case:

$$\varphi(\mathbf{u}) = \frac{1}{\sqrt{2\pi}}e^{-u^2/2}$$

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

$$\boxed{h_n = h_1/\sqrt{n}}$$



Illustrations

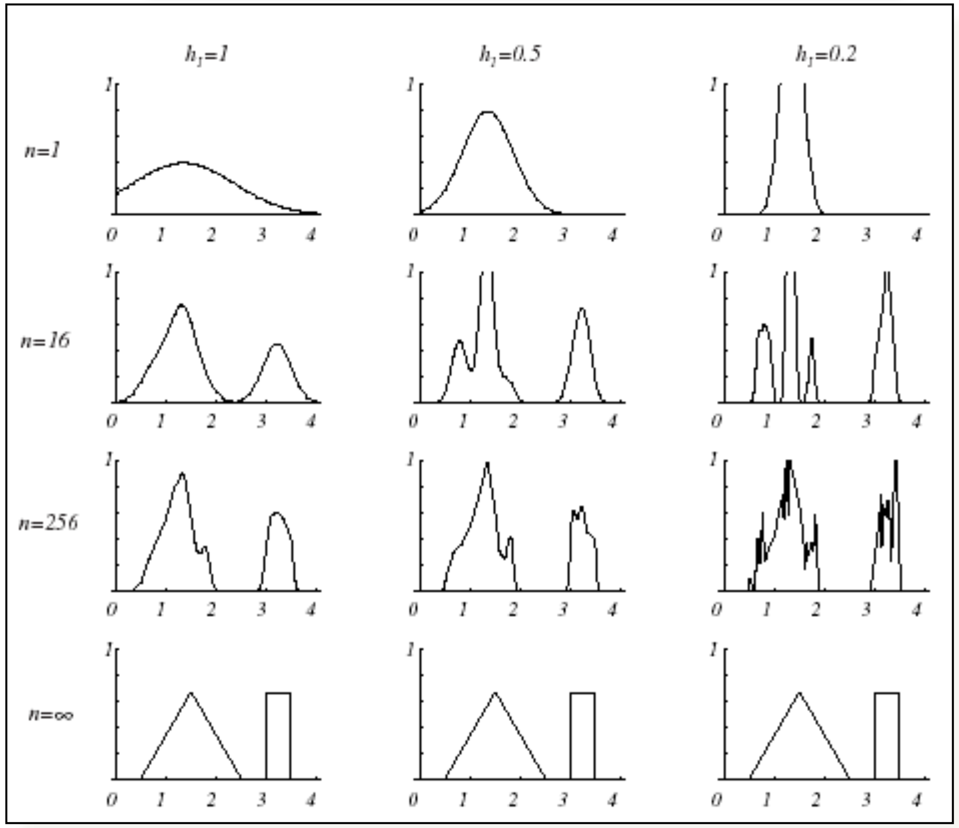
$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

One dimension case:

$$\varphi(\mathbf{u}) = \frac{1}{\sqrt{2\pi}}e^{-u^2/2}$$

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

$$\boxed{h_n = h_1/\sqrt{n}}$$

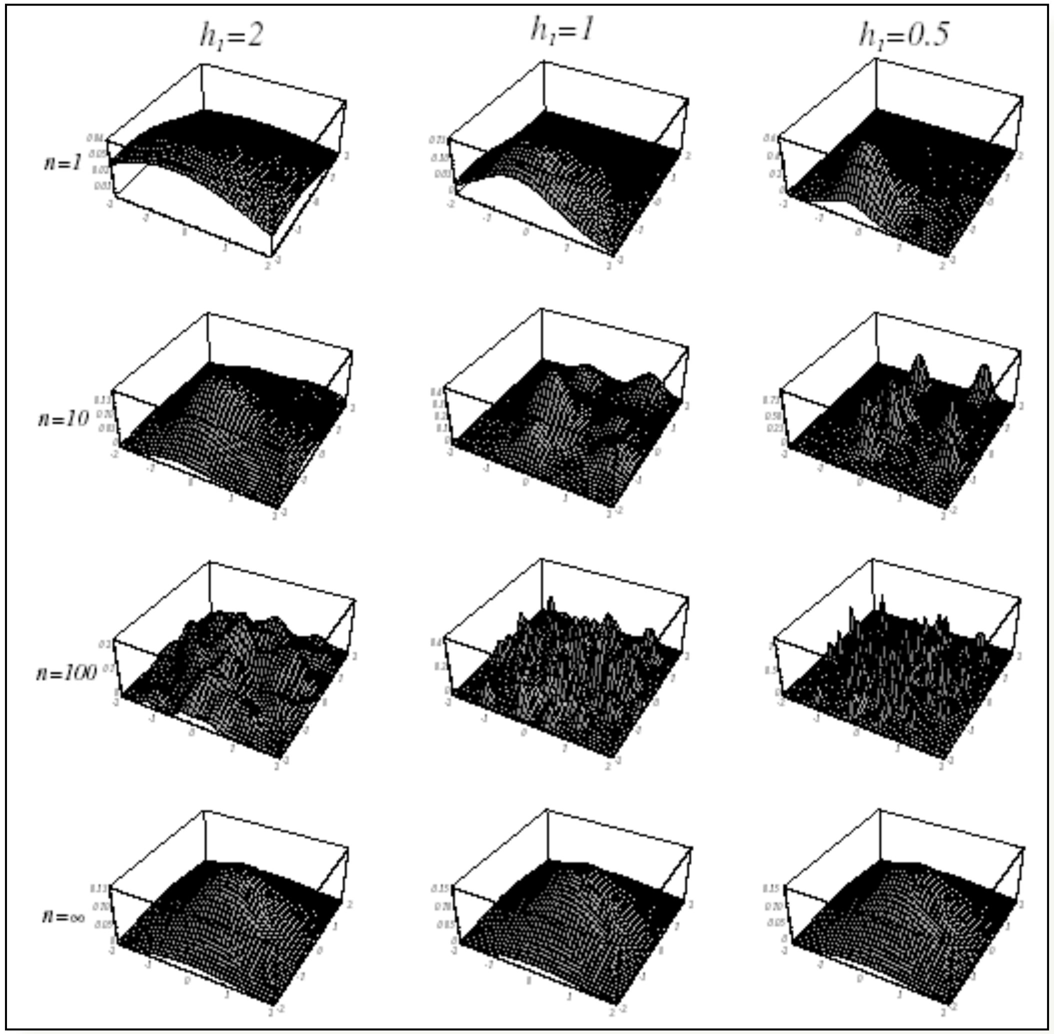


Illustrations

Two dimension case:

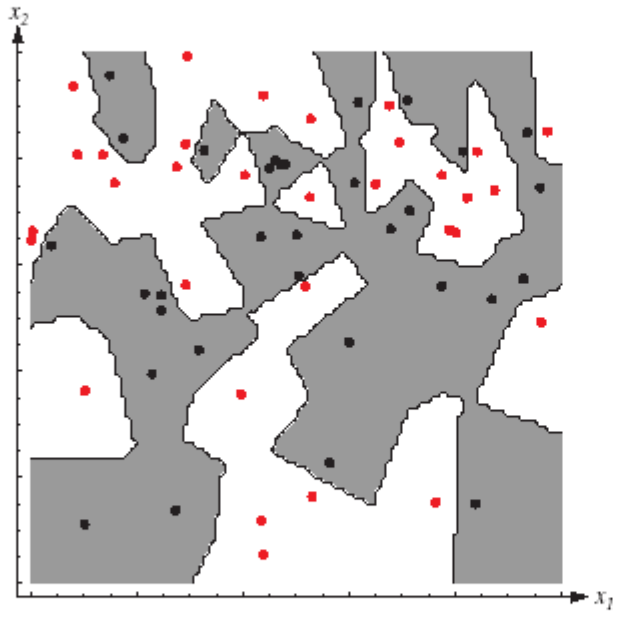
$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n^2}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

$$\boxed{h_n = h_1/\sqrt{n}}$$

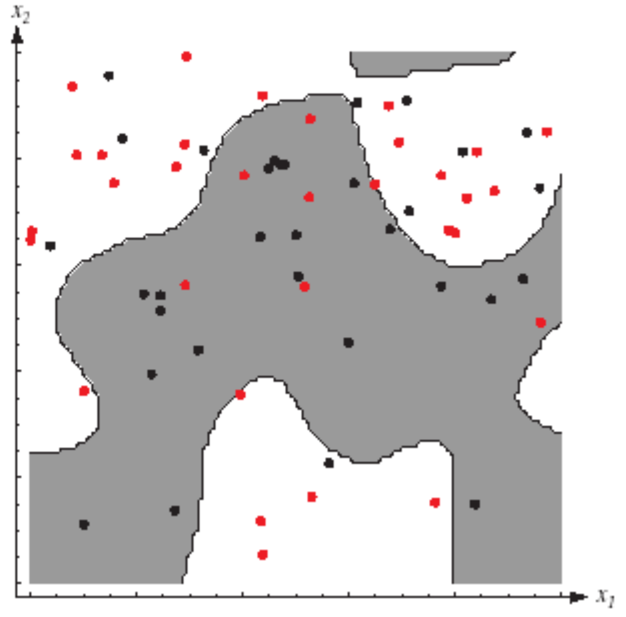


Classification Example

Smaller window



Larger window



Choosing the Window Function

- V_n must approach zero when $n \rightarrow \infty$, but at a rate slower than $1/n$, e.g.,

$$\boxed{V_n = V_1 / \sqrt{n}}$$

- The value of initial volume V_1 is important.
- In some cases, a cell volume is proper for one region but unsuitable in a different region.

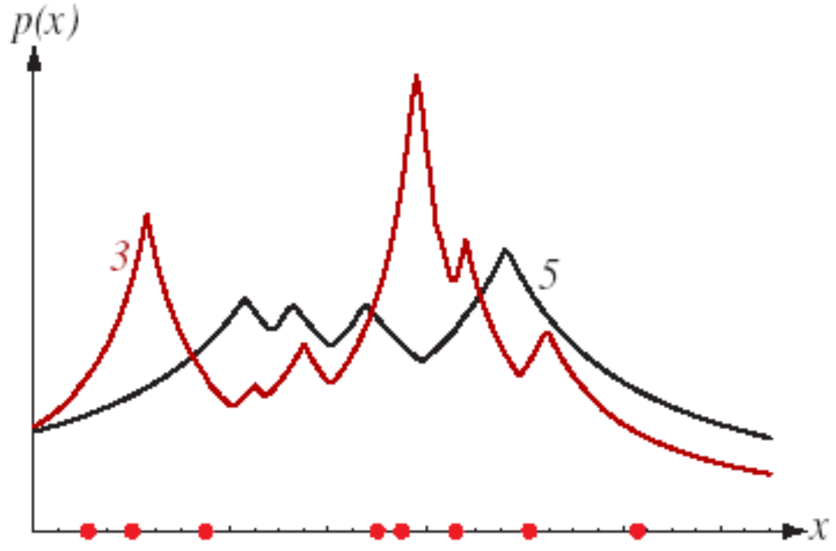
k_n-Nearest-Neighbor Estimation

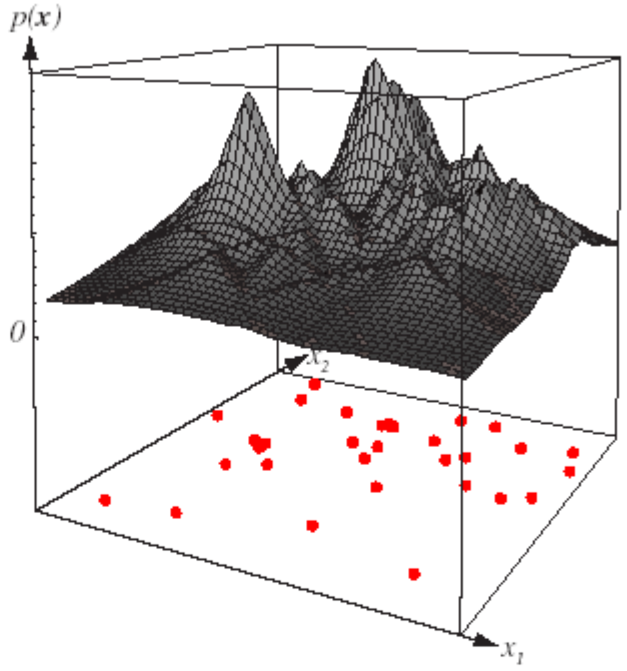
- Let the cell volume depend on the *training data*.
- To estimate $p(\mathbf{x})$, we can center a cell about $\mathbf{x}$ and let it grow until it captures k_n samples, where is some specified function of n, e.g.,

$$\boxed{k_n = \sqrt{n}}$$

Example

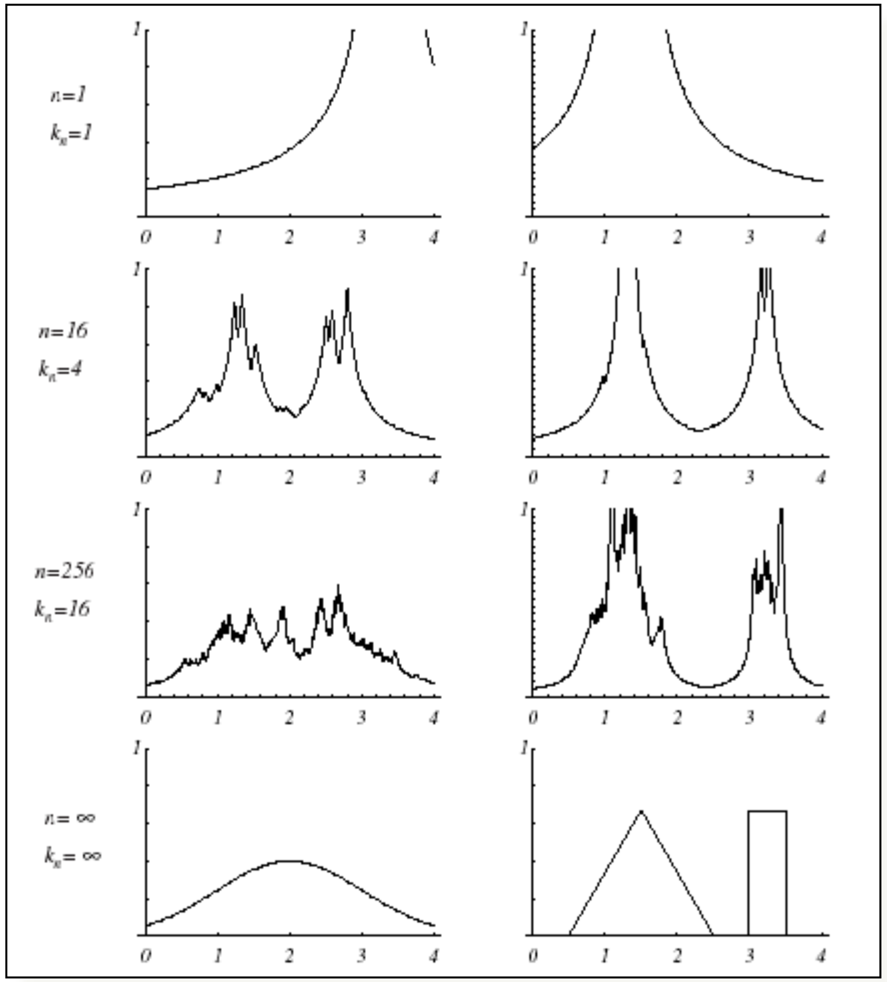
$k_n=5$





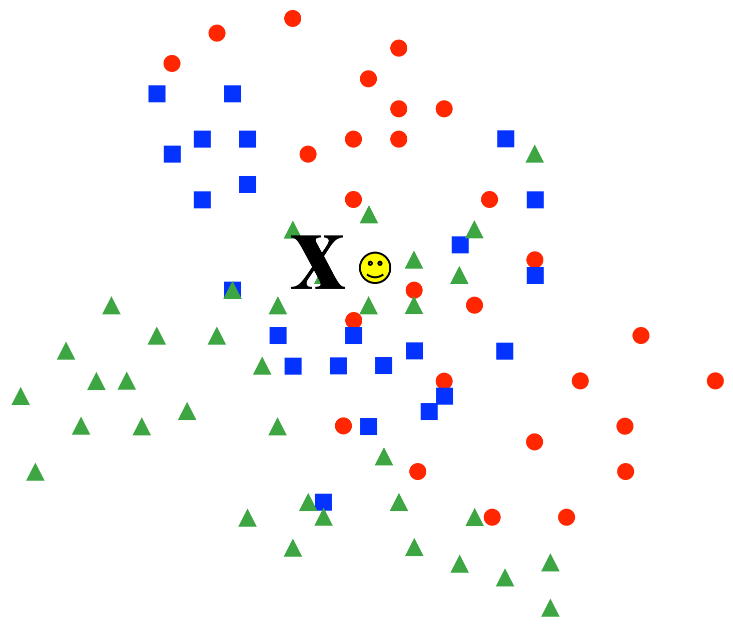
Example

$$\boxed{k_n = \sqrt{n}}$$



Estimation of A Posteriori Probabilities

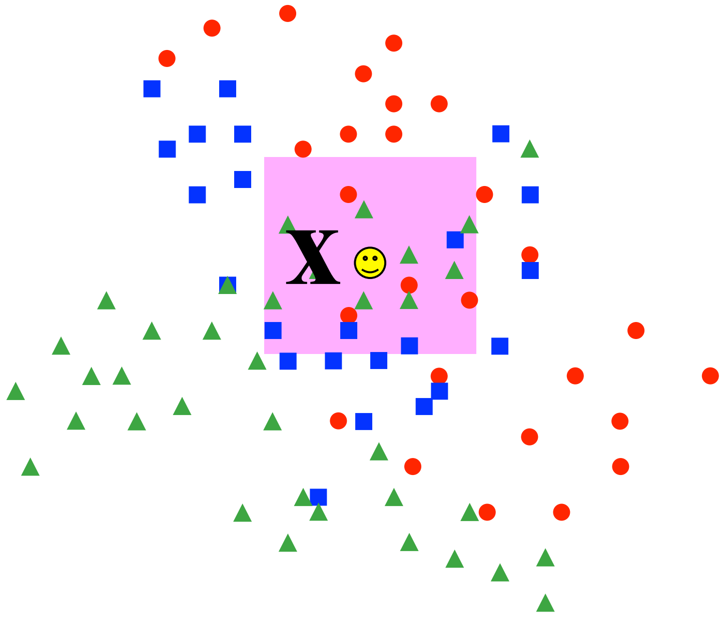
$$P_n(\omega_i|\mathbf{x})=?$$



Estimation of A Posteriori Probabilities

$$P_n(\omega_i|\mathbf{x})=?$$

$$P_n(\omega_i \mid \mathbf{x}) = \frac{p_n(x,\omega_i)}{\sum_{j=1}^{c} p_n(x,\omega_j)} = \frac{k_i}{k_n}$$



$$p_n(x_n,\omega_i) = \frac{k_i / n}{V_n}$$

$$\sum_{j=1}^{c} p_n(x_n,\omega_j) = \frac{k_n / n}{V_n}$$

Estimation of A Posteriori Probabilities

$$P_n(\omega_i \mid \mathbf{x}) = \frac{p_n(x, \omega_i)}{\sum_{j=1}^{c} p_n(x, \omega_j)} = \frac{k_i}{k_n}$$

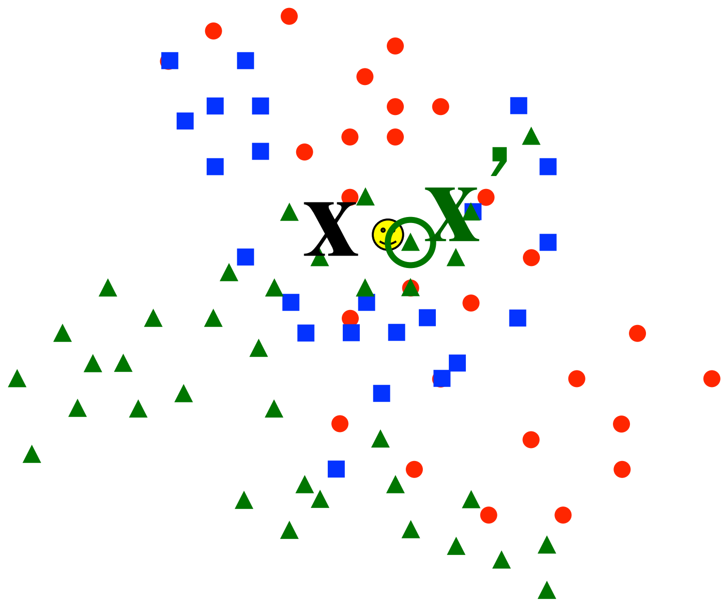
$$\left.\begin{aligned} p_n(x_n, \omega_i) &= \frac{k_i / n}{V_n} \\ \sum_{j=1}^{c} p_n(x_n, \omega_j) &= \frac{k_n / n}{V_n} \end{aligned}\right\}$$

The value of V_n or k_n can be determined base on Parzen window or k_n-nearest-neighbor technique.

Classification:
The Nearest-Neighbor Rule

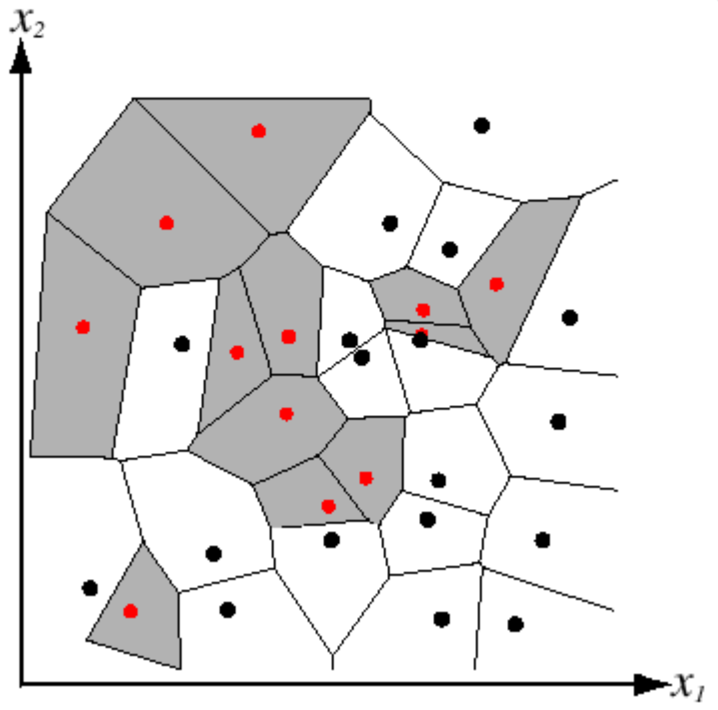
$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ - A set of labeled *prototypes*

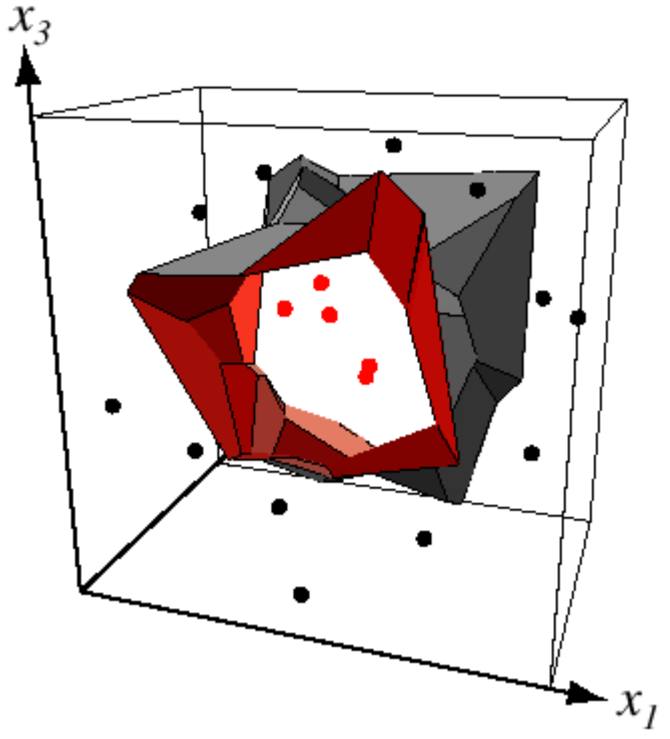
Classify 🙂 as ▲



The Nearest-Neighbor Rule

Voronoi Tessellation





Error Rate

Optimum: $P^*(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})$

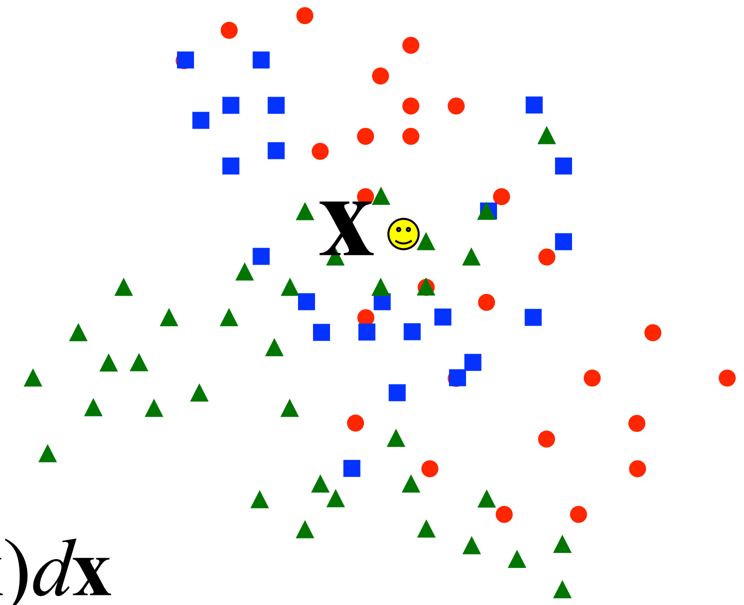
$$P^*(error) = \int P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

Baysian (optimum):

$$\omega_m = \arg\max_i P(\omega_m \mid \mathbf{x})$$

$$P(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})$$

$$P(error) = \int p(error, \mathbf{x}) d\mathbf{x}$$
$$= \int P(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$



Error Rate

$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$

$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

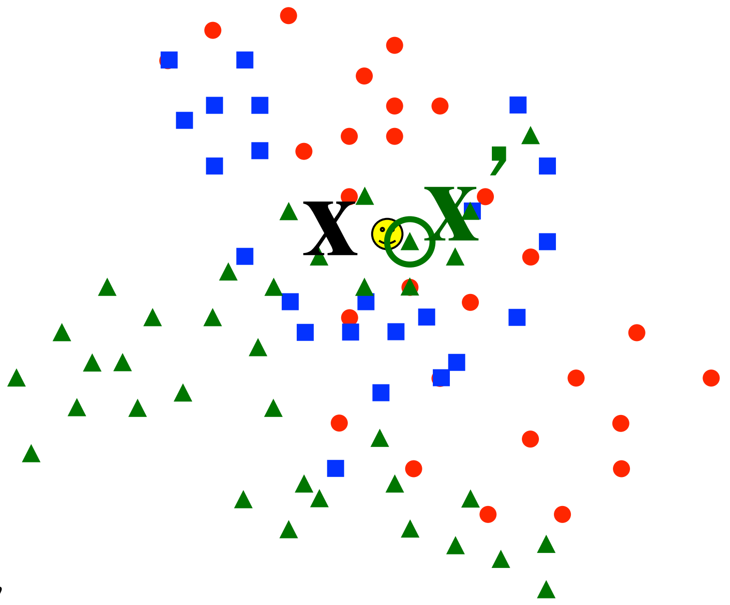
1-NN

Suppose the true class for **x** is θ

$$P(error \mid \mathbf{x}, \mathbf{x}')$$

$$= 1 - \sum_{i=1}^{c} P(\theta = \omega_i, \theta' = \omega_i \mid \mathbf{x}, \mathbf{x}')$$

$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x}) P(\omega_i \mid \mathbf{x}')$$



$$P(error \mid \mathbf{x}) = \int P(error \mid \mathbf{x}, \mathbf{x}') p(\mathbf{x}' \mid \mathbf{x}) d\mathbf{x}'$$

Error Rate

$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$

$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

1-NN

As $n \rightarrow \infty$, $\quad \mathbf{x} \approx \mathbf{x}'$

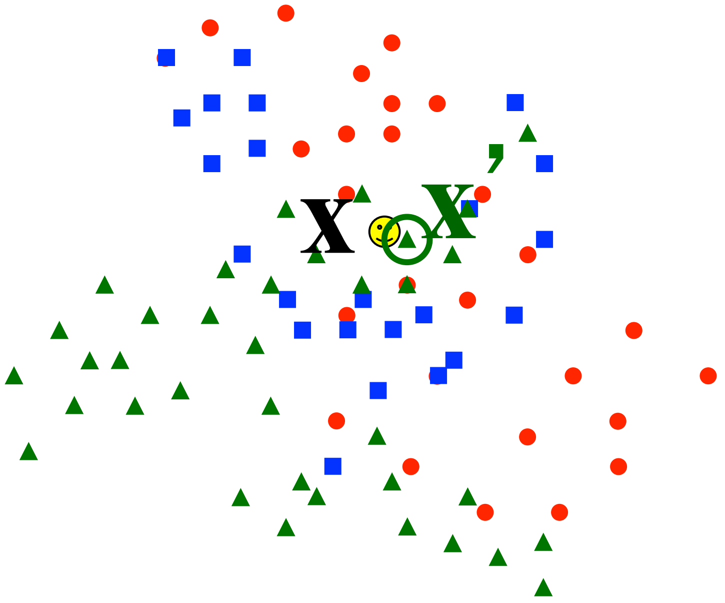
$$p(\mathbf{x}' \mid \mathbf{x}) \approx \delta(\mathbf{x}' - \mathbf{x})$$

$$P(error \mid \mathbf{x}, \mathbf{x}') = 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x}) P(\omega_i \mid \mathbf{x}')$$

$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2$$

$$P(error \mid \mathbf{x}) = \int P(error \mid \mathbf{x}, \mathbf{x}') \underbrace{p(\mathbf{x}' \mid \mathbf{x})}_{?} d\mathbf{x}'$$

$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2$$



Error Rate

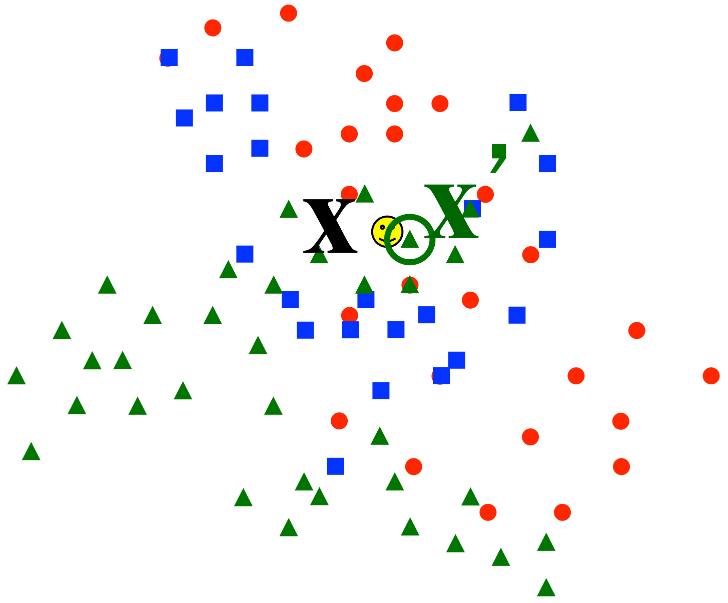
$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$

$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

1-NN

$$P(error) = \int P(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$= \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right] p(\mathbf{x}) d\mathbf{x}$$



$$P(error \mid \mathbf{x}) = 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2$$

Error Bounds

Consider the most complex classification case:

$$P(\omega_i \mid \mathbf{x}) \approx 1/c$$

Bayesian

$$P^*(error) = \int [1 - P(\omega_m \mid \mathbf{x})] p(\mathbf{x}) d\mathbf{x}$$
$$= \int [1 - 1/c)] p(\mathbf{x}) d\mathbf{x}$$
$$= 1 - 1/c$$

1-NN

$$P(error) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right] p(\mathbf{x}) d\mathbf{x}$$
$$= \int [1 - 1/c)] p(\mathbf{x}) d\mathbf{x}$$
$$= 1 - 1/c^2$$

$$P^*(error) \leq P(error) \leq ?$$

Error Bounds

Consider the opposite case: $\boxed{P(\omega_m \mid \mathbf{x}) \approx 1}$

$$\boxed{P^*(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})}$$

Bayesian

$$P^*(error) = \int \left[1 - P(\omega_m \mid \mathbf{x})\right] p(\mathbf{x}) d\mathbf{x}$$

1-NN

Minimized this term

$$\underbrace{P(error) = \int \left[1 - \overbrace{\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2}^{\text{Minimized this term}}\right]}_{} p(\mathbf{x}) d\mathbf{x}$$

Maximized this term to find the upper bound

$$\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 = P(\omega_m \mid \mathbf{x})^2 + \underbrace{\sum_{i \neq m} P(\omega_i \mid \mathbf{x})^2}$$

This term is minimum when all elements have the same value *i.e.,*

$$P(\omega_i \mid \mathbf{x}) = \frac{1 - P(\omega_m \mid \mathbf{x})}{c-1} = \frac{P^*(error)}{c-1}$$

$$P(\omega_m \mid \mathbf{x})^2 = [1 - P^*(error \mid \mathbf{x})]^2 = 1 - 2P^*(error \mid \mathbf{x}) + P^*(error \mid \mathbf{x})^2$$

$$\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \geq 1 - 2P^*(error \mid \mathbf{x}) + \frac{c}{c-1} P^*(error \mid \mathbf{x})^2$$

Error Bounds

Consider the opposite case: $\boxed{P(\omega_m \mid \mathbf{x}) \approx 1}$

$$\boxed{P^*(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})}$$

Bayesian

$$P^*(error) = \int \left[1 - P(\omega_m \mid \mathbf{x})\right] p(\mathbf{x}) d\mathbf{x}$$

$$= \int P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

1-NN

$$P(error) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2\right] p(\mathbf{x}) d\mathbf{x}$$

$$\le \int 2P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$\boxed{P^*(error) \le P(error) \le 2P^*(error)}$$

$$1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \le 2P^*(error \mid \mathbf{x}) - \frac{c}{c-1} P^*(error \mid \mathbf{x})^2 \le 2P^*(error \mid \mathbf{x})$$

$$\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \ge 1 - 2P^*(error \mid \mathbf{x}) + \frac{c}{c-1} P^*(error \mid \mathbf{x})^2$$

Error Bounds

Consider the opposite case: $\boxed{P(\omega_m \mid \mathbf{x}) \approx 1}$

$$\boxed{P^*(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})}$$

Bayesian

$$P^*(error) = \int \left[1 - P(\omega_m \mid \mathbf{x})\right] p(\mathbf{x}) d\mathbf{x}$$

$$= \int P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

1-NN

$$P(error) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2\right] p(\mathbf{x}) d\mathbf{x}$$

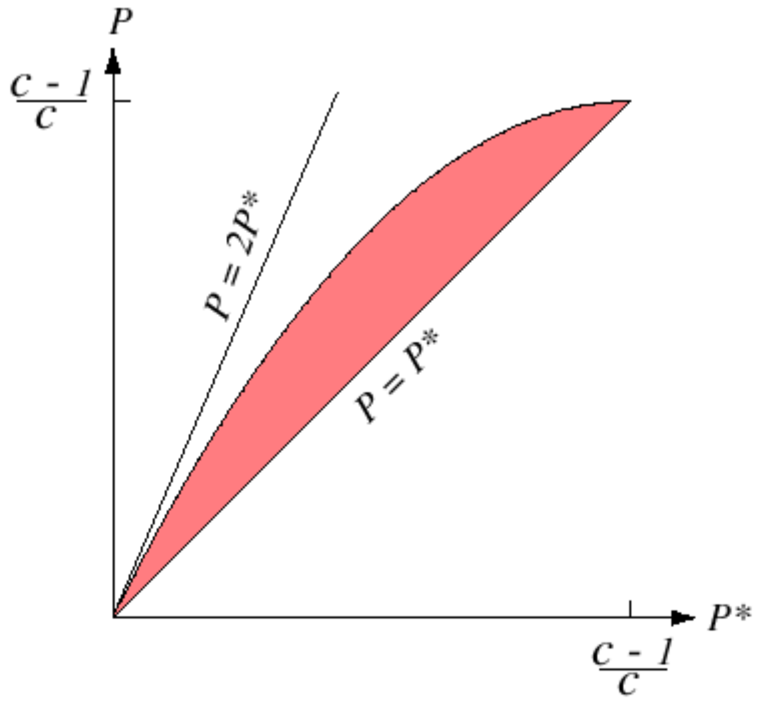
$$\leq \int 2P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$\boxed{P^*(error) \leq P(error) \leq 2P^*(error)}$$

The nearest-neighbor rule is a *suboptimal* procedure.

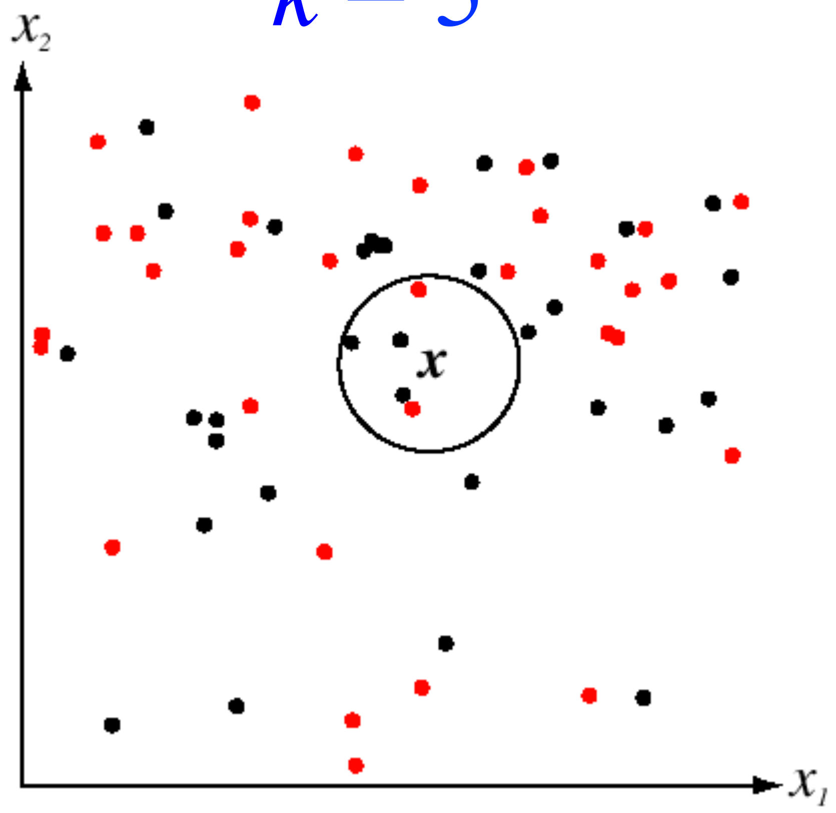
The error rate is never worse than *twice* the Bayes rate.

Error Bounds



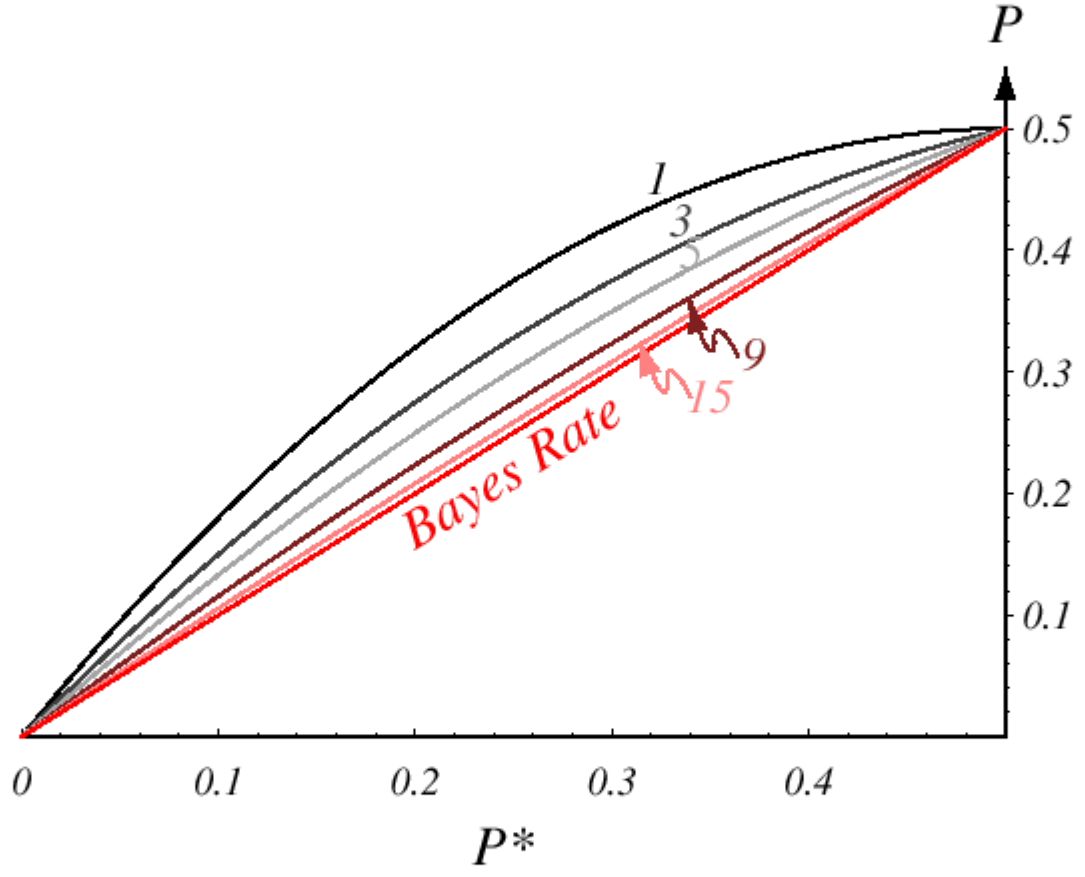
Classification: The k-Nearest-Neighbor Rule

$k = 5$



Assign pattern to the class wins the majority.

Error Bounds



Computation Complexity

- The computation complexity of the nearest-neighbor algorithm (both in *time* and *space*) has received a great deal of analysis.

- Require $O(dn)$ space to store n prototypes in a training set.
  - Editing, pruning or condensing

- To search the nearest neighbor for a d-dimensional test point $\mathbf{x}$, the time complexity is $O(dn)$.
  - Partial distance
  - Search tree

Partial Distance

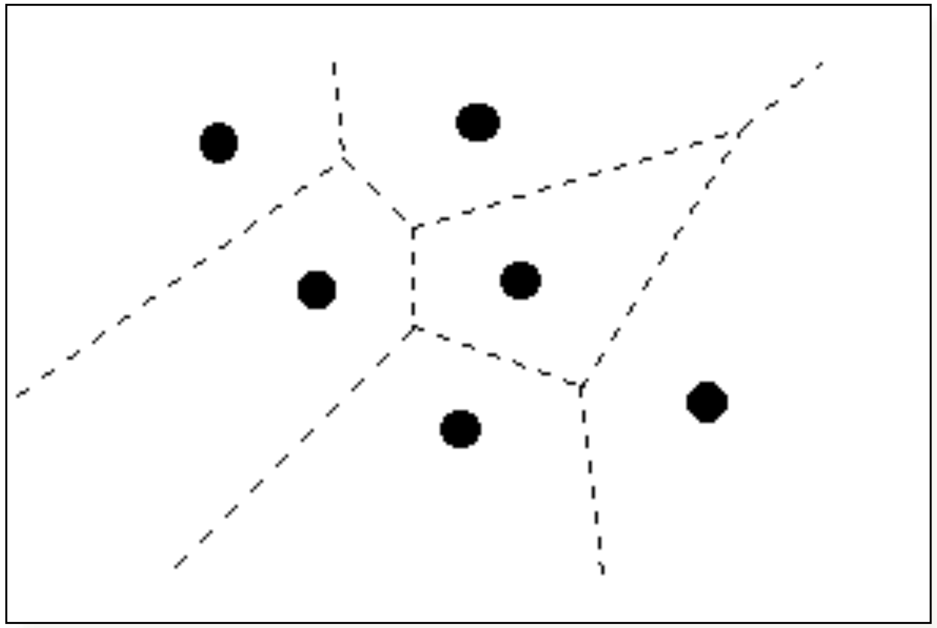
Using the following fact to early throw *far-away* prototypes

$$r \le d$$

$$D_r(\mathbf{a},\mathbf{b}) = \left(\sum_{k=1}^{r}(a_k - b_k)^2\right)^{1/2} \le \left(\sum_{k=1}^{d}(a_k - b_k)^2\right)^{1/2}$$

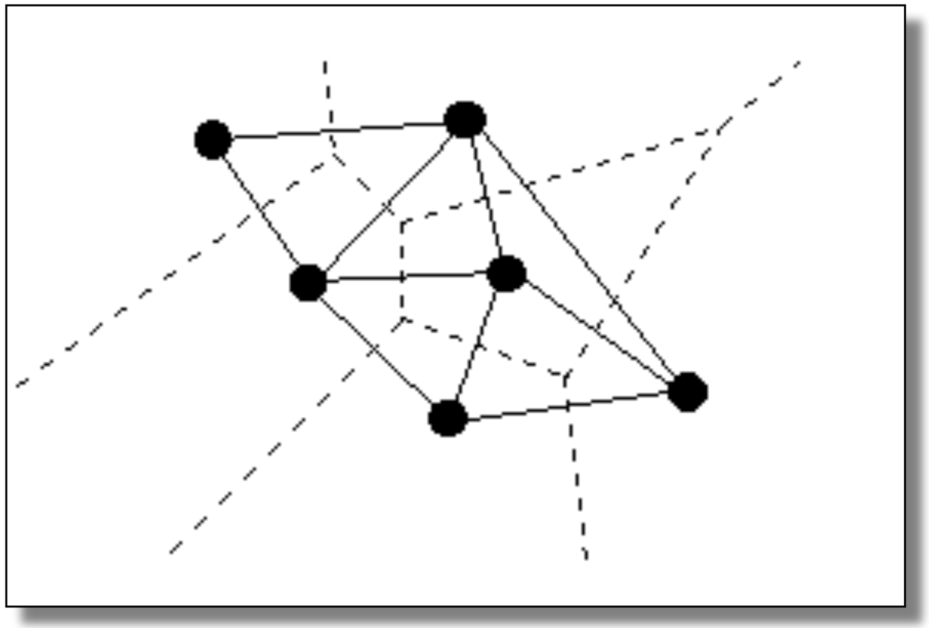
Editing Nearest Neighbor

Given a set of points, a *Voronoi diagram* is a partition of space into *regions*, within which all points are closer to some particular node than to any other node.



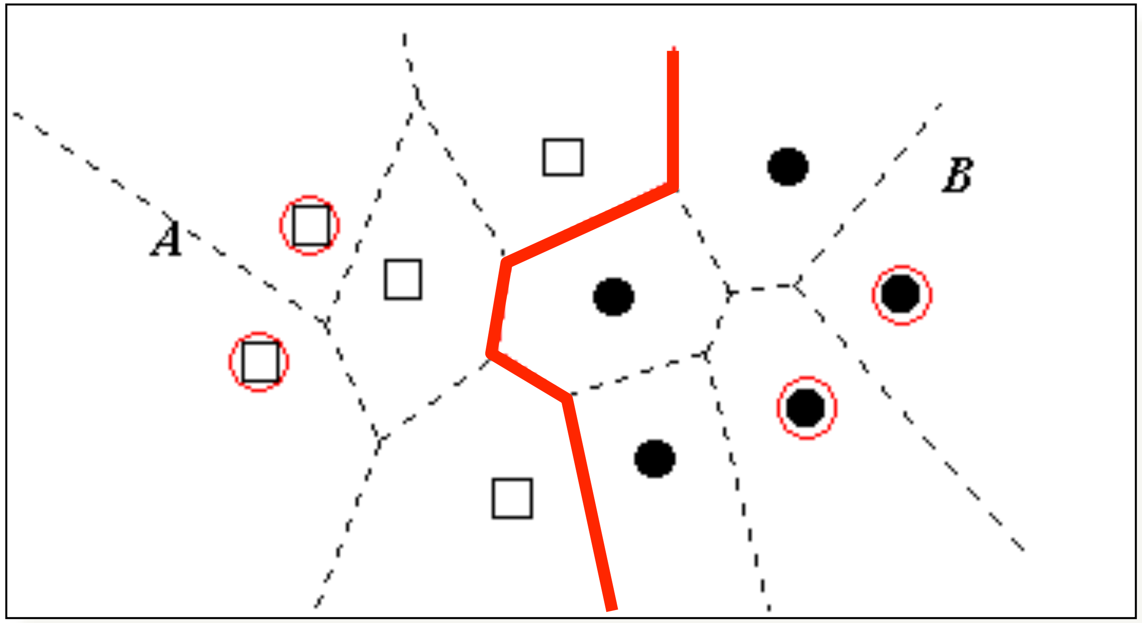
Delaunay Triangulation

If two Voronoi regions share a boundary, the nodes of these regions are connected with an edge. Such nodes are called the *Voronoi neighbors* (or *Delaunay neighbors*).

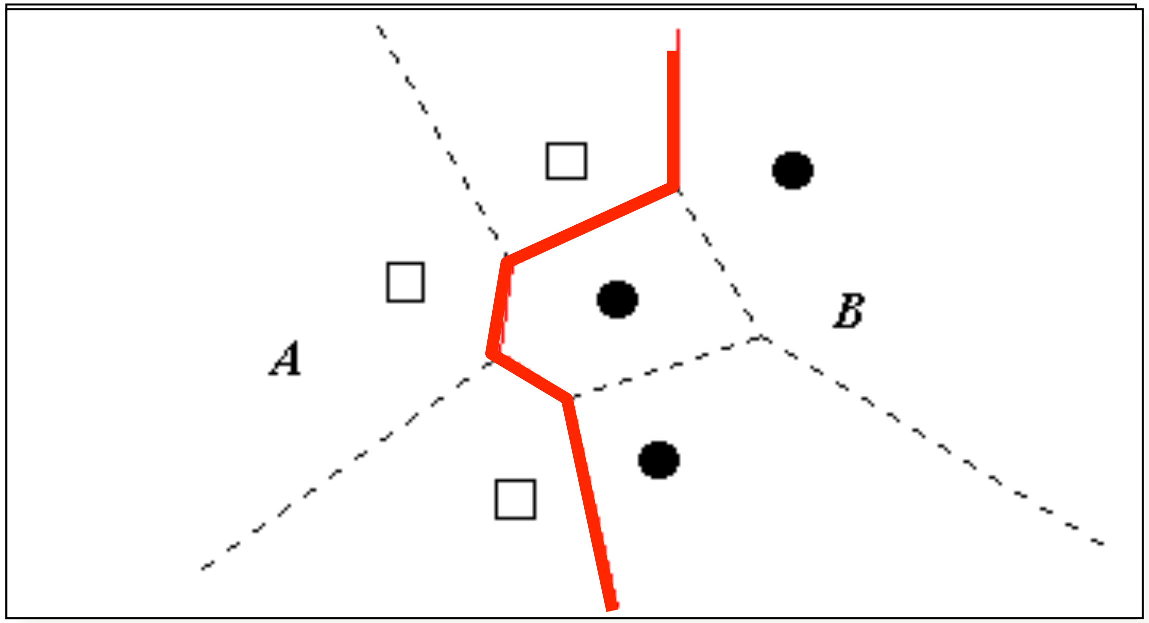


The Decision Boundary

The circled prototypes are redundant.



The Edited Training Set



Editing: The Voronoi Diagram Approach

- Compute the *Delaunay triangulation* for the training set.
- Visit each node, *marking* it if all its *Delaunay neighbors* are of the same class as the current node.
- *Delete all marked nodes*, exiting with the remaining ones as the edited training set.

Spectral Clustering

Acknowledgements for subsequent slides to

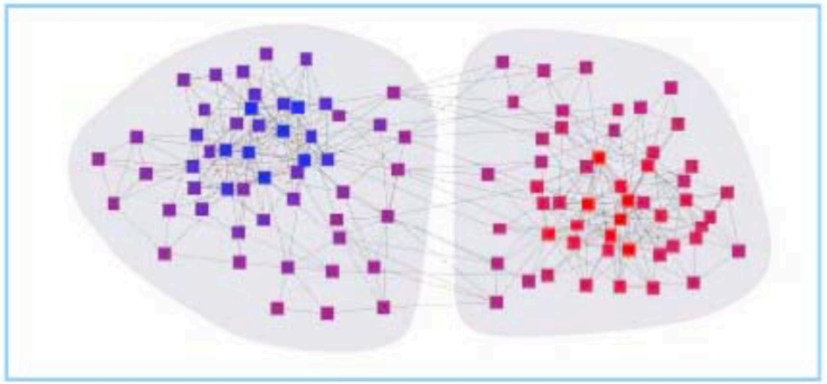
Xiaoli Fern

CS 534: Machine Learning 2011

http://web.engr.oregonstate.edu/~xfern/classes/cs534/

Spectral Clustering

- Represent data points as the vertices V of a graph G.
- Vertices are connected by edges E
- Edges have weights described by matrix *W*
  - Large weight $W(i, j)$ mean that the points i and j are very similar; small weights imply dissimilarity

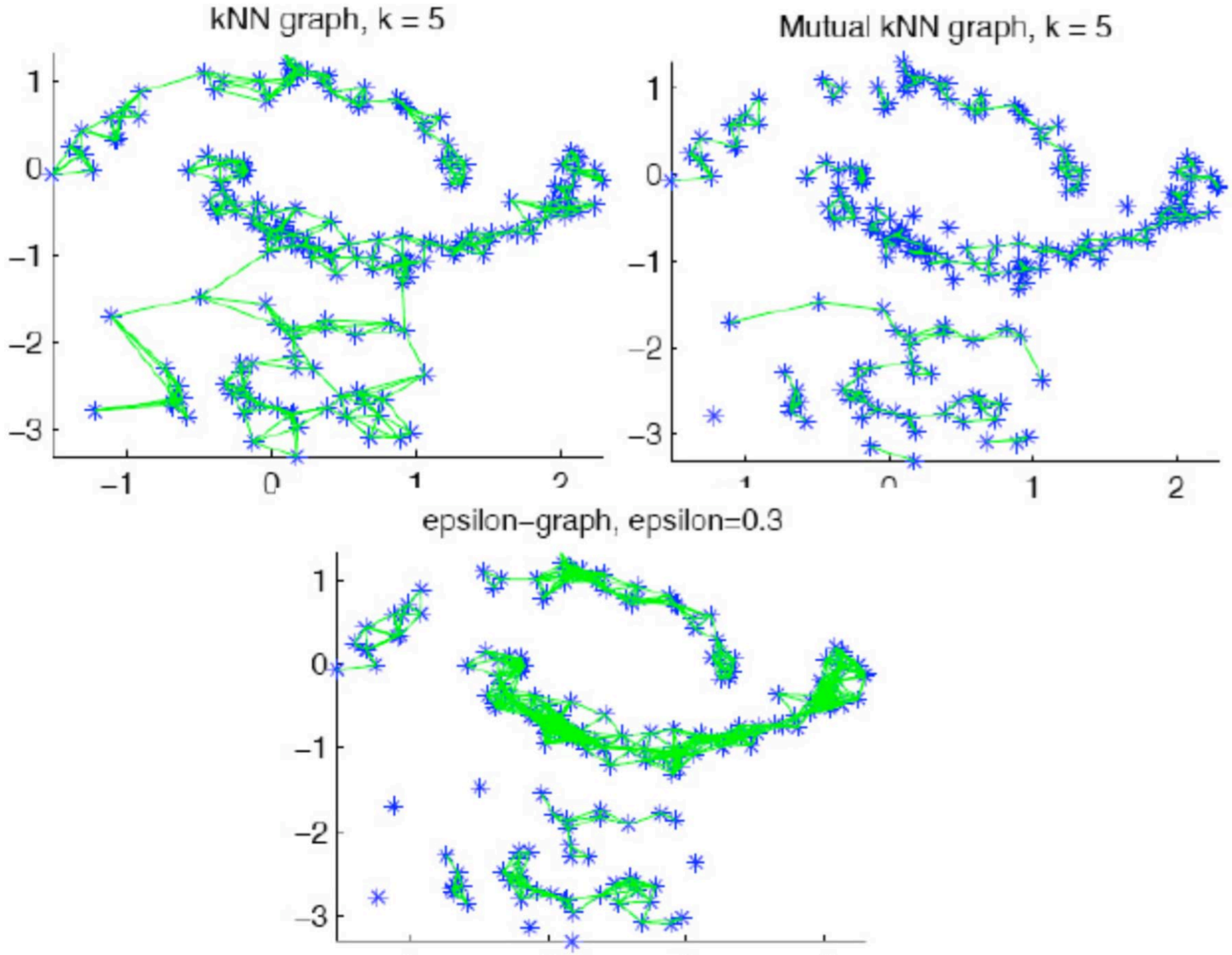


Methods that use the spectrum of the similarity matrix *W* to cluster are known as ***spectral clustering***

How to Create the Graph?

- One could create
  - A fully connected graph
  - K-nearest neighbor graph (each node is only connected to its K-nearest neighbors)
  - ϵ-neighborhood graph (each node is only connected to points within ϵ distance)
- It is common to use a Gaussian Kernel to compute similarity between objects

$$W(i,j) = \exp \frac{-\left|x_i - x_j\right|^2}{\sigma^2}$$


kNN graph, k = 5
Mutual kNN graph, k = 5
epsilon−graph, epsilon=0.3
1
0
−1
−2
−3
−1
0
1
2

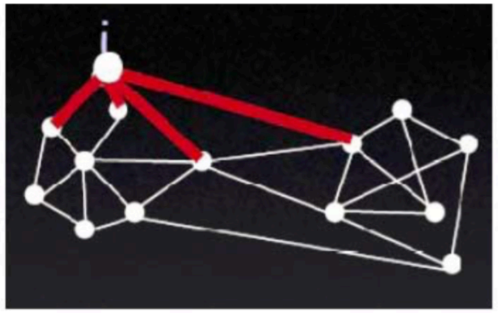
Motivations / Objectives

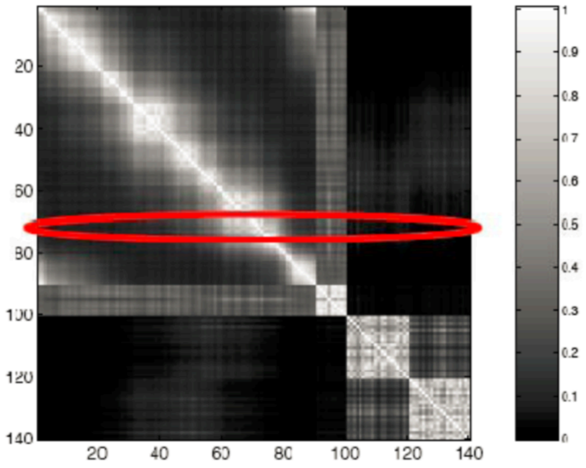
- There are different ways to interpret the spectral clustering
- One can view spectral clustering as finding partitions of the graph that minimizes Normalized Cut
- Alternatively, we can also view this as performing a random walk on the graph

Graph Terminologies

- Degree of nodes

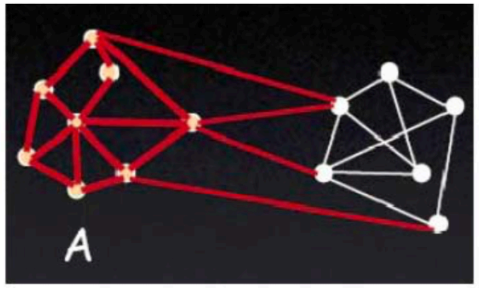
$$d_i = \sum_j w_{i,j}$$

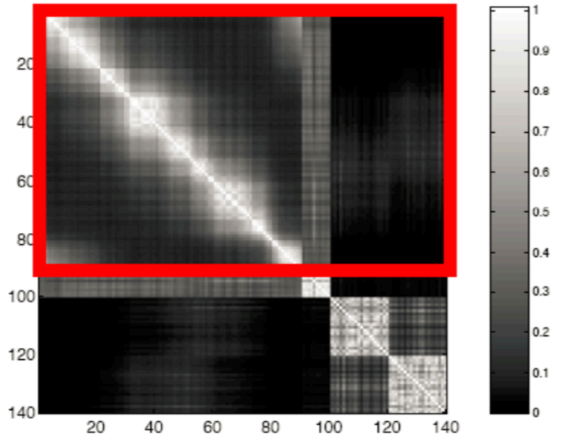




- Volume of a set

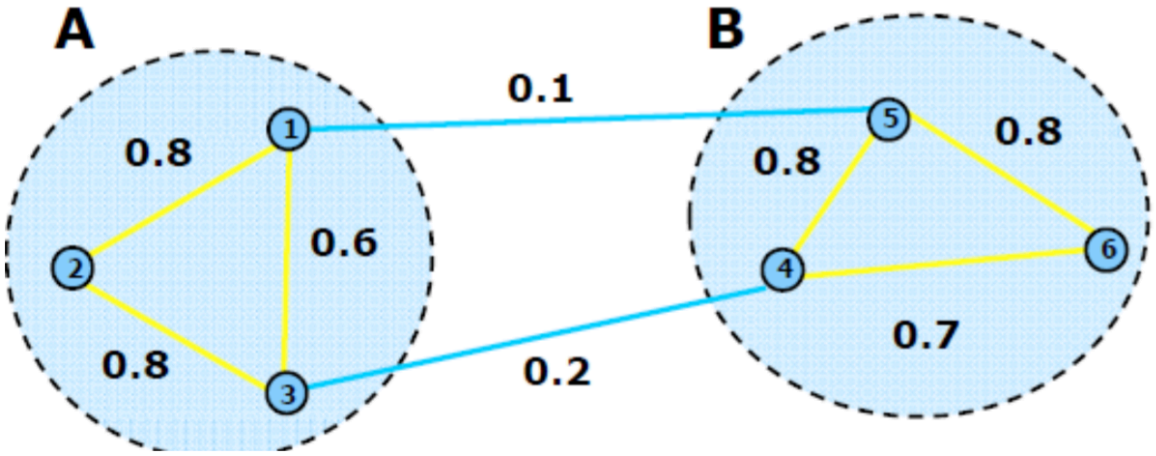
$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





Graph Cut

- Consider a partition of the graph into two parts A and B



- ***Cut(A, B)***: sum of the weights of the set of edges that connect the two groups

$$cut(A,B) = \sum_{i\in A, j\in B} w_{ij} \quad \boldsymbol{= 0.3}$$

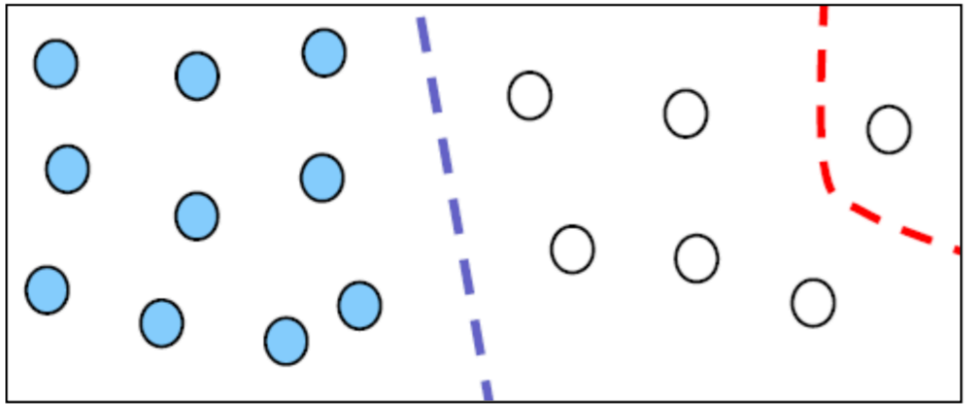
- An intuitive goal is find the partition that minimizes the cut

Min Cut Objective

- **Mincut:** Minimize weight of connections between groups

$$\min_{A \cap B = \emptyset, A \cup B = V} Cut(A, B)$$

- Problem:
  - Prefer degenerate solution (e.g. the red partition)

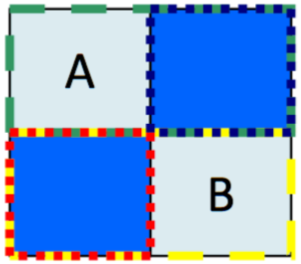


  - Need to express preference for more balanced solution

Normalized Cut

- Consider the connectivity between groups relative to the volume of each group

$$Ncut(A,B) = \frac{cut(A,B)}{Vol(A)} + \frac{cut(A,B)}{Vol(B)}$$



$$Ncut(A,B) = cut(A,B)\frac{Vol(A)+Vol(B)}{Vol(A)Vol(B)}$$

Maximized when Vol(A) and Vol(B) are equal.
Thus encourage balanced cut

Optimizing Ncut Objective

- How to minimize *Ncut*?

  Let W be the similarity matrix, $W(i, j) = W_{i,j}$;
  Let D be the diag. matrix, $D(i,i) = \sum_j W(i, j)$;
  Let x be a vector in $\{1,-1\}^N$, $x(i) = 1 \Leftrightarrow i \in A$.

- With some simplifications, we can show:

$$\min_x Ncut(x) = \min_y \frac{y^T (D-W) y}{y^T D y}$$

Rayleigh quotient

Subject to: $y^T D1 = 0$ (*$\mathbf{y}$ takes discrete values*)

NP-Hard!

Solving Ncut

- Relax the optimization problem into the continuous domain by solving generalized eigenvalue system:
$$\min_y y^T(D-W)y \text{ subject to } y^TDy = 1$$
- Lagrangian: $L(y,\lambda) = y^T(D-W)y - \lambda(y^TDy - 1)$
- Taking partial derivative w.r.t. y and set it to zero:
$$(D-W)y = \lambda Dy$$
- Note that $(D-W)1 = 0$, so the first eigenvector is $y_0 = 1$ with eigenvalue 0.
- The second smallest eigenvector is the real valued solution to this problem!!

2-way Normalized Cuts

1. Compute the affinity matrix W, compute the degree matrix (D), D is diagonal and $D(i,i) = \sum_{j \in V} W(i,j)$
2. Solve $(D - W)y = \lambda D y$, where $D - W$ is called the Laplacian matrix
3. Use the eigenvector with the second smallest eigen-value to bipartition the graph into two parts.

Creating Bi-Partition Using 2nd Eigenvector

- Sometimes there is not a clear threshold to split based on the second vector since it takes continuous values
- How to choose the splitting point?

a) Pick a constant value (0, or 0.5).

b) Pick the median value as splitting point.

c) Look for the splitting point that has the minimum *Ncut* value:

1. Choose *n* possible splitting points.
2. Compute *Ncut* value.
3. Pick minimum.

K-way Partition?

- Recursive bi-partitioning
  - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
  - Disadvantages: Inefficient, unstable
- Cluster using multiple eigenvectors
  - Build a reduced space from multiple eigenvectors.
  - Commonly used in recent papers
  - A preferable approach… its like doing dimension reduction then k-means

Spectral Clustering

(Ng, Jordan, and Weiss 2001)

- Form the affinity matrix W

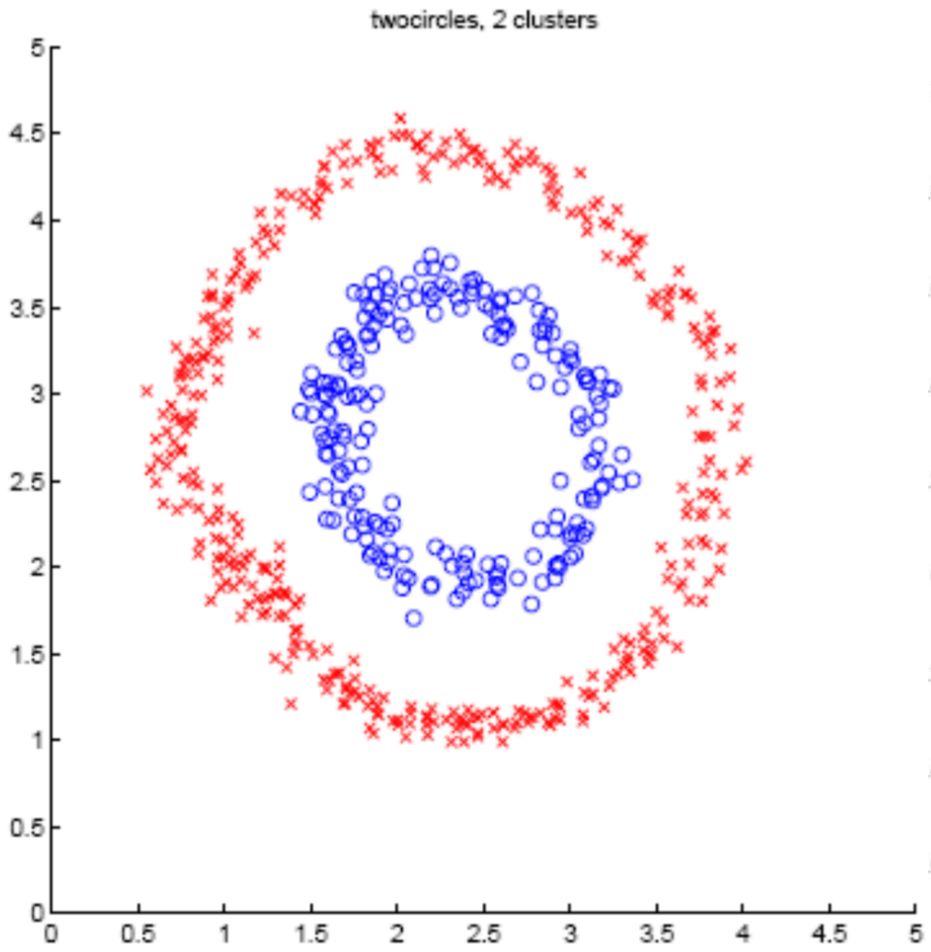
$$W(i,j) = \exp(-\frac{|x_i - x_j|^2}{2\sigma}), W(i,i) = 0$$

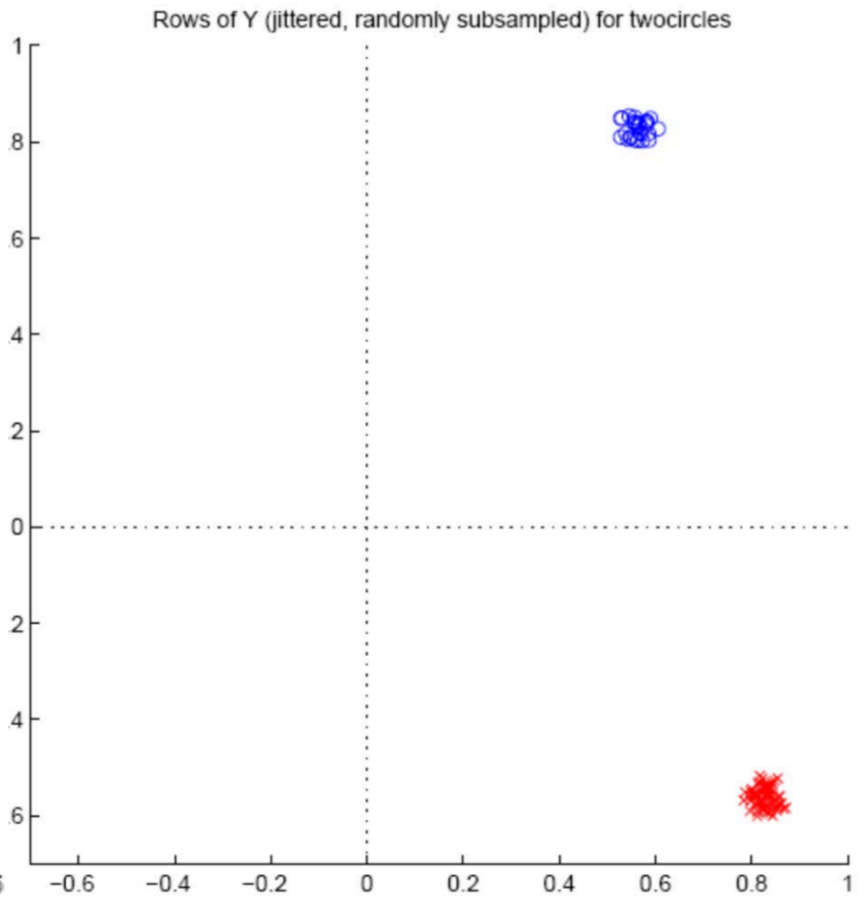
- Compute the degree matrix $D = diag(W \cdot \mathbf{1})$
- Compute the normalized graph Laplacian

$$L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

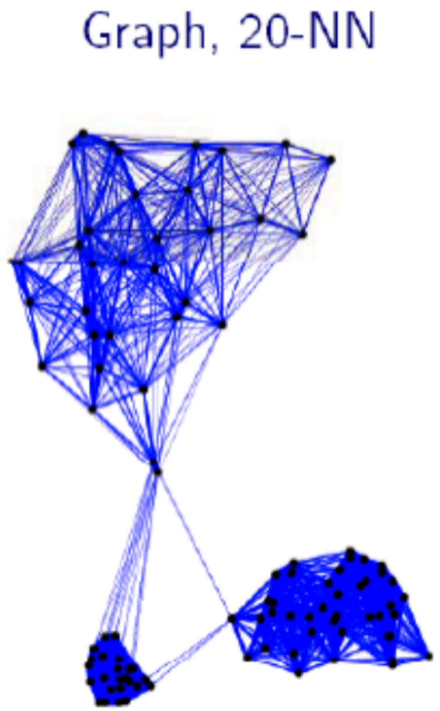
- Find the k largest eigenvectors, for new data matrix $X'_{n\times k}$
- Normalize each row(each example) to have unit length
  - $x_i \leftarrow \frac{x_i}{|x_i|}$
- Treating each row as a data point in k-d space and cluster the data into k clusters via kmeans

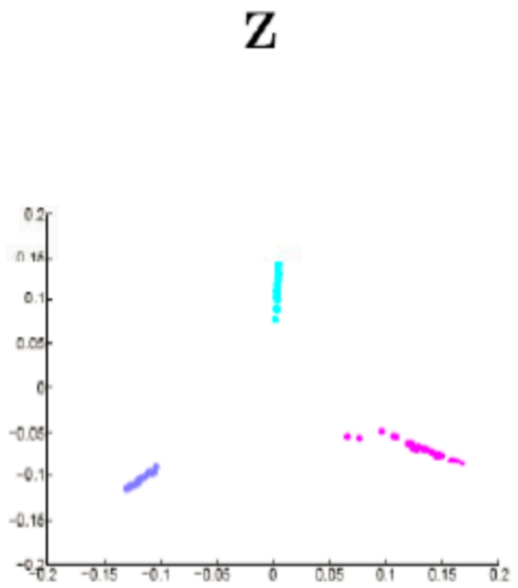
Ng A.Y., Jordan, M.I., and Weiss Y
On Spectral Clustering: Analysis and an algorithm
In Proc. Neural Information Processing Systems **2001**

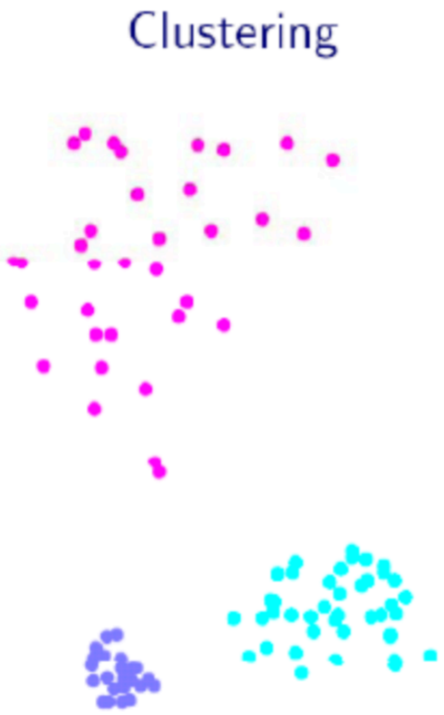




Spectral embedding of the data







Spectral embedding of the data