

Clustering

- Unsupervised learning task; data has no labels
- The task is to find “natural groupings” in data
- Groups include instances such that those in a cluster are more similar (“closer”) to each other than to those in another cluster
 - Minimizing the distance of all instances in a cluster from the centroid (*inter-cluster scatter*) for all clusters
 - If the number of clusters k is given, NP-complete problem
 - Relaxing optimality
 - Relaxing membership: probabilistic

- Implies the use of similarity measure or *distance*; if all attributes are numerical, the distance can be naturally *Euclidean*; if not, it is less obvious:
 - *Manhattan* distance
 - *Mahalanobis* distance
- Practically important, often the first step in exploratory data analysis: find clusters, make them into classes:
 - City mobility example
- Evaluation of clustering

Mahalanobis distance measures the distance between a point P and a distribution D .

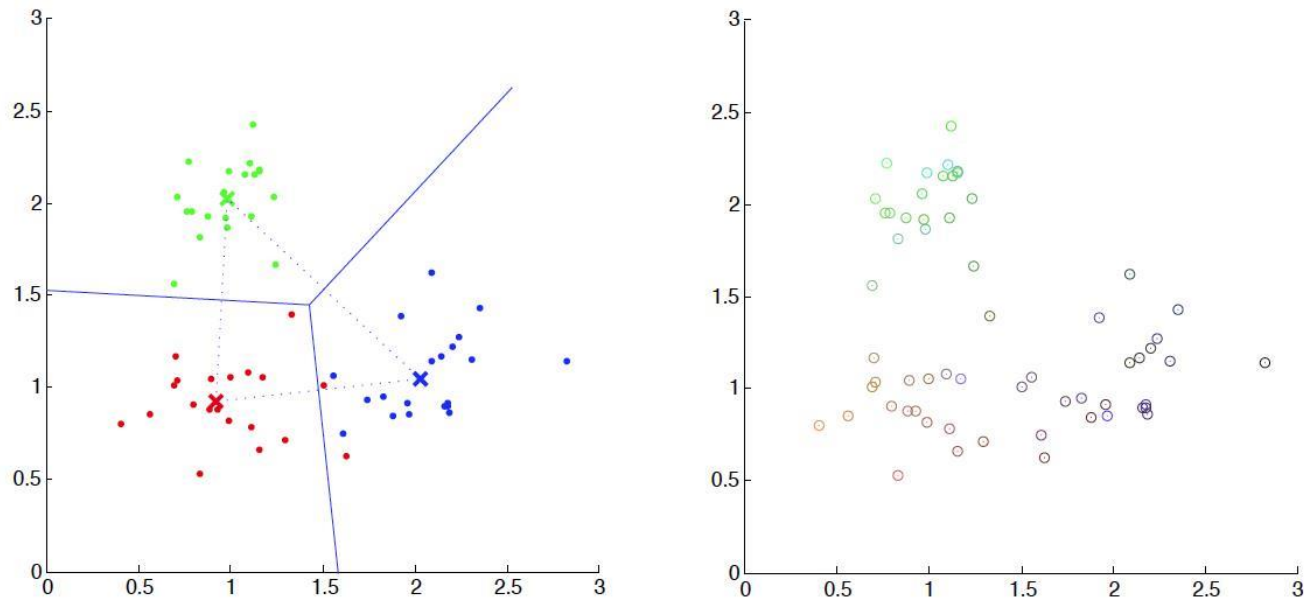
It is a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D .

This distance is zero if P is at the mean of D , and grows as P moves away from the mean:

along each principal component axis, it measures the number of standard deviations from P to the mean of D .

Types of clustering algorithms

- Partitioning methods
 - K-means
 - K-medoids
- Density-based methods
 - DBSCAN
- Model-based clustering
 - Expectation-Maximization (EM)
- Hierarchical clustering



(left) An example of a predictive clustering. The coloured dots were sampled from three bivariate Gaussians centred at **(1,1)**, **(1,2)** and **(2,1)**. The crosses and solid lines are the cluster exemplars and cluster boundaries found by 3-means. **(right)** A soft clustering of the same data found by matrix decomposition.

Partitioning Clustering – k means

1. Define k - the number of clusters
2. Choose k points randomly as cluster centres
3. For any instance, assign it to the cluster whose centre is the closest
4. If no cluster gets modified, STOP
5. Make centroids (“instances” created by taking means of all instances in the cluster) new clusters centres
6. go to 3

Algorithm $KMeans(D, K)$

Input : data $D \subseteq \mathbb{R}^d$; number of clusters $K \in \mathbb{N}$.

Output : K cluster means $\mu_1, \dots, \mu_K \in \mathbb{R}^d$.

randomly initialise K vectors $\mu_1, \dots, \mu_K \in \mathbb{R}^d$;

repeat

 assign each $\mathbf{x} \in D$ to $\operatorname{argmin}_j \text{Dis}_2(\mathbf{x}, \mu_j)$;

for $j = 1$ to K **do**

$D_j \leftarrow \{\mathbf{x} \in D \mid \mathbf{x} \text{ assigned to cluster } j\}$;

$\mu_j = \frac{1}{|D_j|} \sum_{\mathbf{x} \in D_j} \mathbf{x}$;

end

until no change in μ_1, \dots, μ_K ;

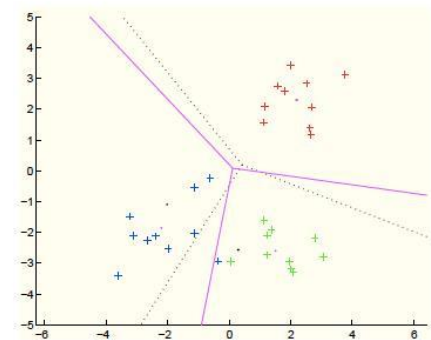
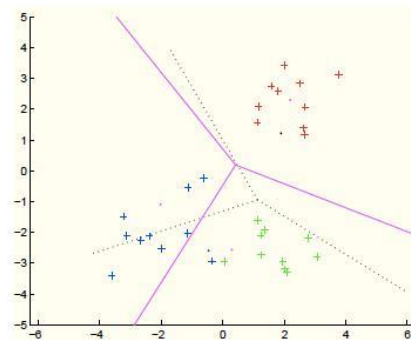
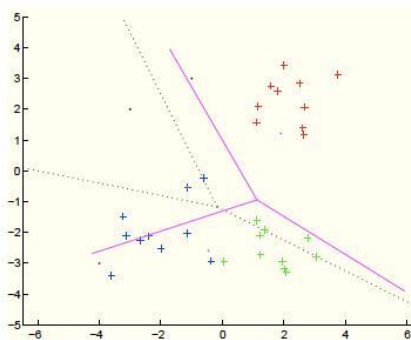
return μ_1, \dots, μ_K ;

Centroid;
notice
 Dis_2 !

k -means - example

p.248

Figure 8.11: K -means clustering



(left) First iteration of 3-means on Gaussian mixture data. The dotted lines are the Voronoi boundaries resulting from randomly initialised centroids; the **violet solid lines** are the result of the recalculated means. **(middle)** Second iteration, taking the previous partition as starting point (dotted line). **(right)** Third iteration with stable clustering.

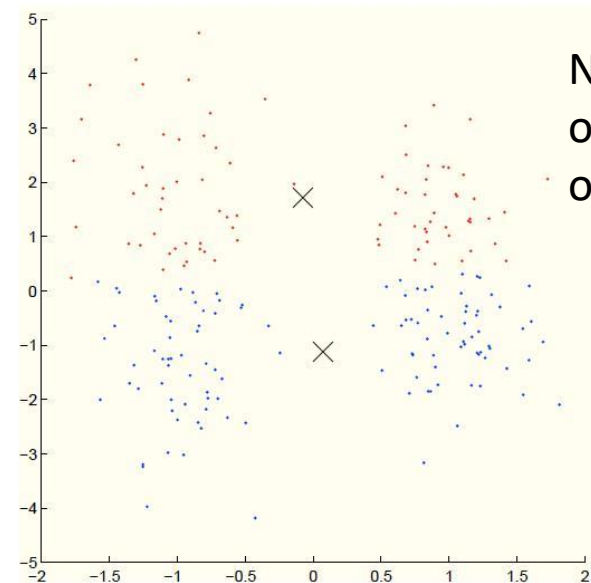
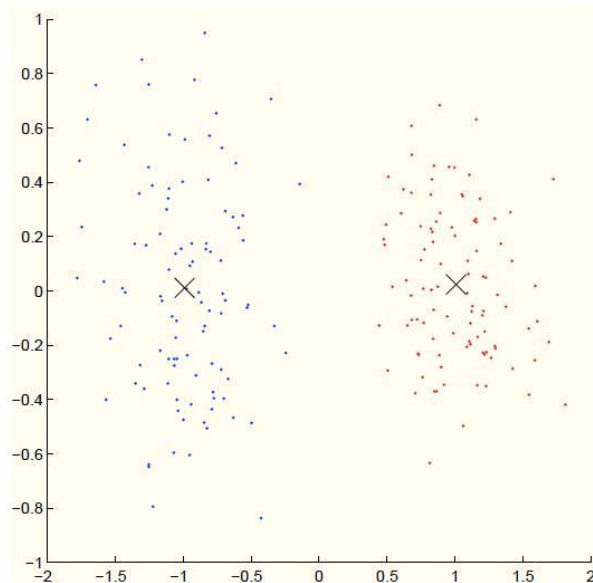
- When k -means terminates, the sum of all distances of points to their cluster centres is minimal
- This is only local, i.e. depends on the initial choice of k
- Efficiency problem - $\text{\#iterations} * k * N$; an NP-complete algorithm
- Data structures can be used for efficient implementation
- Centroid is NOT an instance!

K -means sensitivity

p.251



Figure 8.13: Scale-sensitivity of K -means



Notice *distance*
of the two pairs
of centroids

(left) On this data 2-means detects the right clusters. **(right)** After rescaling the y -axis, this configuration has a higher between-cluster scatter than the intended one.

K-medoids

Algorithm KMedoids(D, K, Dis)

Input : data $D \subseteq \mathcal{X}$; number of clusters $K \in \mathbb{N}$;
distance metric $\text{Dis} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

Output : K medoids $\mu_1, \dots, \mu_K \in D$, representing a predictive clustering of \mathcal{X} .
randomly pick K data points $\mu_1, \dots, \mu_K \in D$;

repeat

 assign each $\mathbf{x} \in D$ to $\arg\min_j \text{Dis}(\mathbf{x}, \mu_j)$;

for $j = 1$ to k **do**

$D_j \leftarrow \{\mathbf{x} \in D \mid \mathbf{x} \text{ assigned to cluster } j\}$;

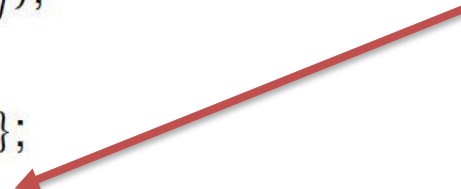
$\mu_j = \arg\min_{\mathbf{x} \in D_j} \sum_{\mathbf{x}' \in D_j} \text{Dis}(\mathbf{x}, \mathbf{x}')$;

end

until no change in μ_1, \dots, μ_K ;

return μ_1, \dots, μ_K ;

cost of finding
this instance (a
medoid) vs
cost of finding a
centroid in k-
means

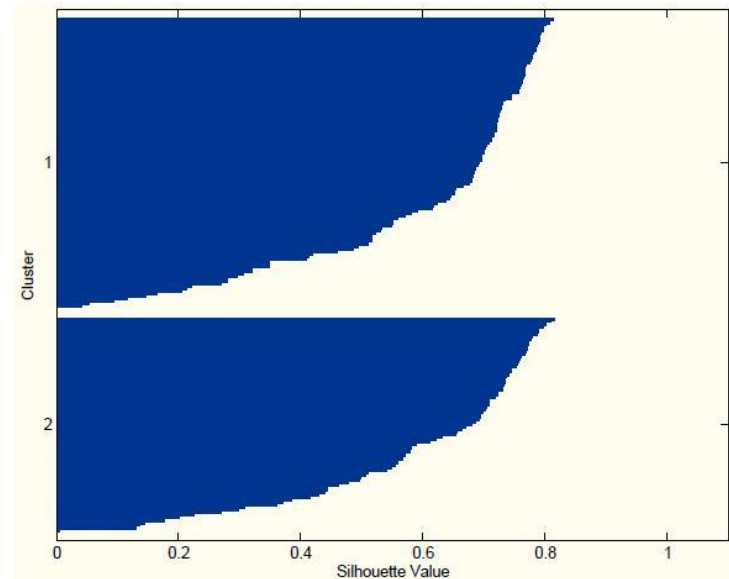
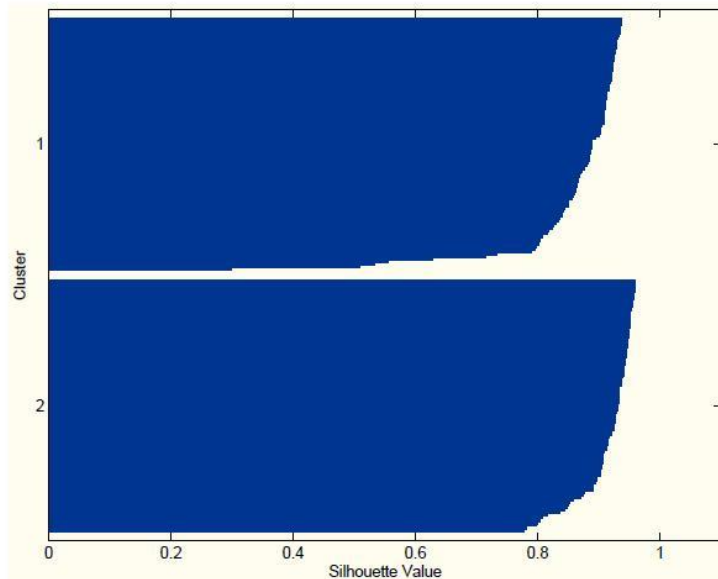


k-means only works with the Euclidean distance!

Silhouette as visual cluster valuation

- For any data point \mathbf{x}_i , let $d(\mathbf{x}_i, D_j)$ denote the average distance of \mathbf{x}_i to the data points in cluster D_j , and let $j(i)$ denote the index of the cluster that \mathbf{x}_i belongs to.
- Furthermore, let $a(\mathbf{x}_i) = d(\mathbf{x}_i, D_{j(i)})$ be the average distance of \mathbf{x}_i to the points in its own cluster $D_{j(i)}$, and let $b(\mathbf{x}_i) = \min_{k \neq j(i)} d(\mathbf{x}_i, D_k)$ be the average distance to the points in its neighbouring cluster.
- We would expect $a(\mathbf{x}_i)$ to be considerably smaller than $b(\mathbf{x}_i)$, but this cannot be guaranteed.
- So we can take the difference $b(\mathbf{x}_i) - a(\mathbf{x}_i)$ as an indication of how 'well-clustered' \mathbf{x}_i is, and divide this by $b(\mathbf{x}_i)$ to obtain a number less than or equal to 1.

Silhouette



(left) Silhouette for the clustering in Figure 8.13 (left), using squared Euclidean distance. Almost all points have a high $s(\mathbf{x})$, which means that they are much closer, on average, to the other members of their cluster than to the members of the neighbouring cluster.

(right) The silhouette for the clustering in Figure 8.13 (right) is much less convincing.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Designed to find clusters with arbitrary shape
- Clusters are defined as dense regions of objects in the data space that are separated by regions of low density (representing noise)
- Generally, used with spatial databases, but not restricted to this type of data.
- No need to specify k

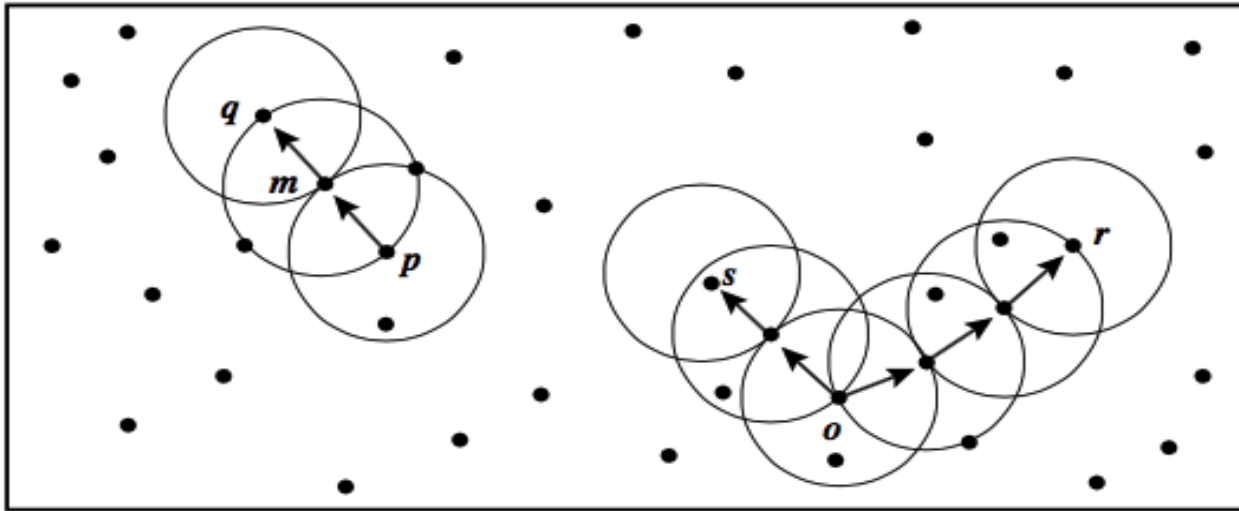
DBSCAN

- The neighborhood within a radius ϵ of a given object is called the ϵ -neighborhood of the object.
- If the ϵ -neighborhood of an object contains at least a minimum number, MinPts, of objects, then the object is called a core object.
- Given a set of objects, D , we say that an object p is directly density-reachable from object q if p is within the ϵ -neighborhood of q , and q is a core object.
- An object p is density-connected to object q with respect to ϵ and MinPts in a set of objects, D , if there is an object $o \in D$ such that both p and q are density-reachable from o with respect to ϵ and MinPts.

DBSCAN – the very idea

- Use the *closure* of the density-connectedness to find connected dense regions as clusters
- A subset $C \subset D$ is a cluster if
 - For any two objects o_1 and $o_2 \in C$, o_1 and o_2 are density connected
 - There does not exist an object $o \in C$ and $o' \in D-C$ such that o and o' are density-connected

DBSCAN



- Assume $\text{MinPts}=3$
- m , p , o and r are core objects; q is directly density-reachable from m ;
- o , r , and s are all density-connected.

DBSCAN – the algorithm

- DBSCAN searches for clusters by checking the ε - neighborhood of each point in the database.
- If the ε -neighborhood of a point p contains more than MinPts , a new cluster with p as a core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters.
- The process terminates when no new point can be added to any cluster.

DBSCAN – the algorithm

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D : a data set containing n objects,
- ϵ : the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.

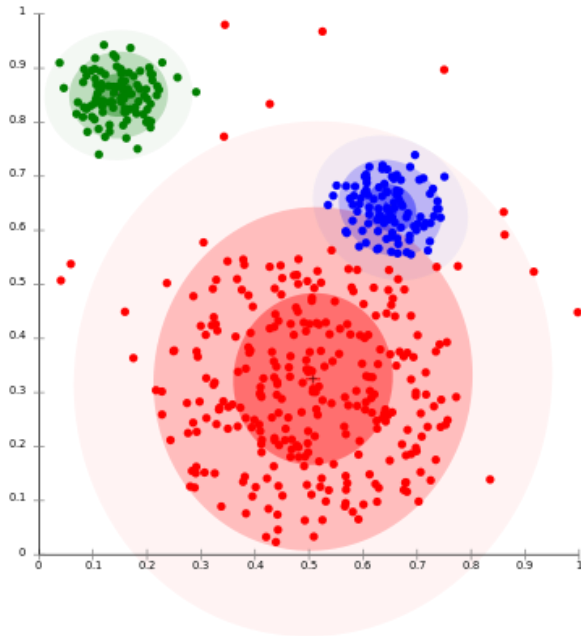
Method:

```
(1) mark all objects as unvisited;
(2) do
(3)   randomly select an unvisited object  $p$ ;
(4)   mark  $p$  as visited;
(5)   if the  $\epsilon$ -neighborhood of  $p$  has at least  $MinPts$  objects
(6)     create a new cluster  $C$ , and add  $p$  to  $C$ ;
(7)     let  $N$  be the set of objects in the  $\epsilon$ -neighborhood of  $p$ ;
(8)     for each point  $p'$  in  $N$ 
(9)       if  $p'$  is unvisited
(10)        mark  $p'$  as visited;
(11)        if the  $\epsilon$ -neighborhood of  $p'$  has at least  $MinPts$  points
(12)          add those points to  $N$ ;
(13)        if  $p'$  is not yet a member of any cluster, add  $p'$  to  $C$ ;
(14)     end for
(15)   else mark  $p$  as noise;
(16) until no object is unvisited;
```

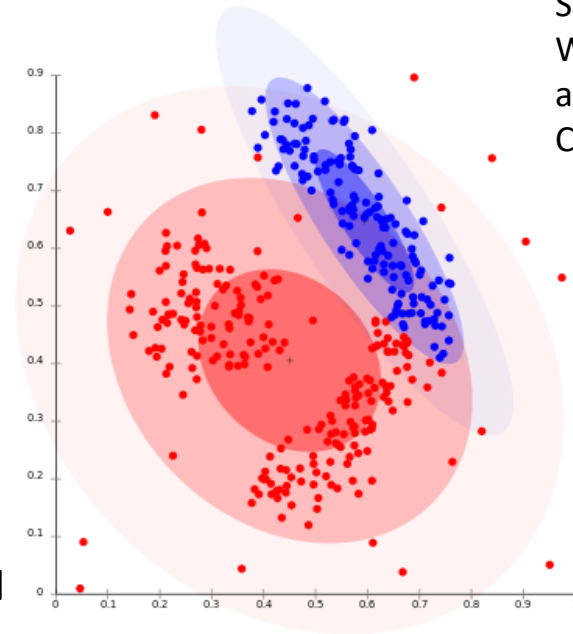
EM vs DBSCAN – Gaussian-distributed data

Source:
Wikipedia
article on
Cluster Analysis

EM



DBSCAN



DBSCAN expects some kind of density drop to detect cluster borders.

For overlapping Gaussian distributions, the cluster borders produced by the density-based clustering algorithms will often look arbitrary, because the cluster density decreases continuously

Kernelization of clustering

- Key idea: kernel can be used to measure the distance:
- $\text{Dis}_2(x,y) = \|x-y\|_2 = \text{sqrt}((x-y)(x-y)) = \text{sqrt}(x^*x - 2x^*y + y^*y) = \text{sqrt}(\kappa(x,x) - 2\kappa(x,y) + \kappa(y,y))$
- Then plug into *k-means*, but...
- ...the same issue as in *k-medoids* comes up
- Alternatively, cosine distance:
- $\cos \theta = \frac{x \cdot y}{\sqrt{(x \cdot x)(y \cdot y)}}$; $\text{distance}(x,y) = 1 - \cos \theta$
- Independent on the length of x, y ; projection points on a unit sphere and measuring distance on that sphere

Probabilistic clustering

- Finite mixture model
- Set of k probability distributions represents k clusters: each distribution determines the probability that an instance x would have a certain set of attribute values *if it was known that x belongs to this cluster*
- There also a probability distributions that reflect the relative population sizes of each cluster

Finite mixture problem cont'd

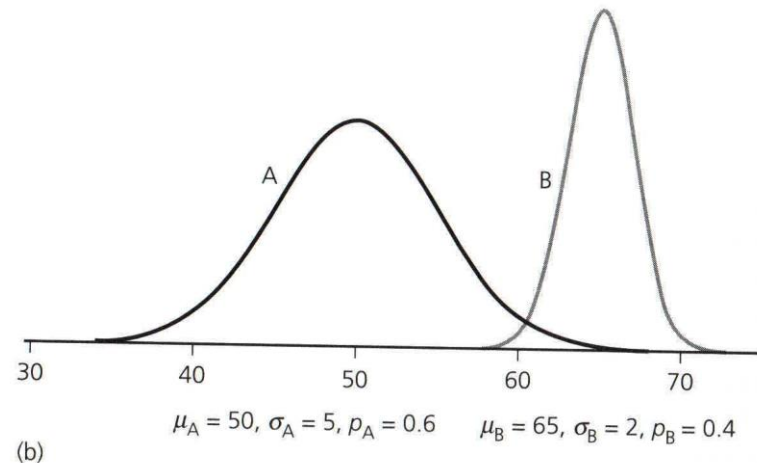
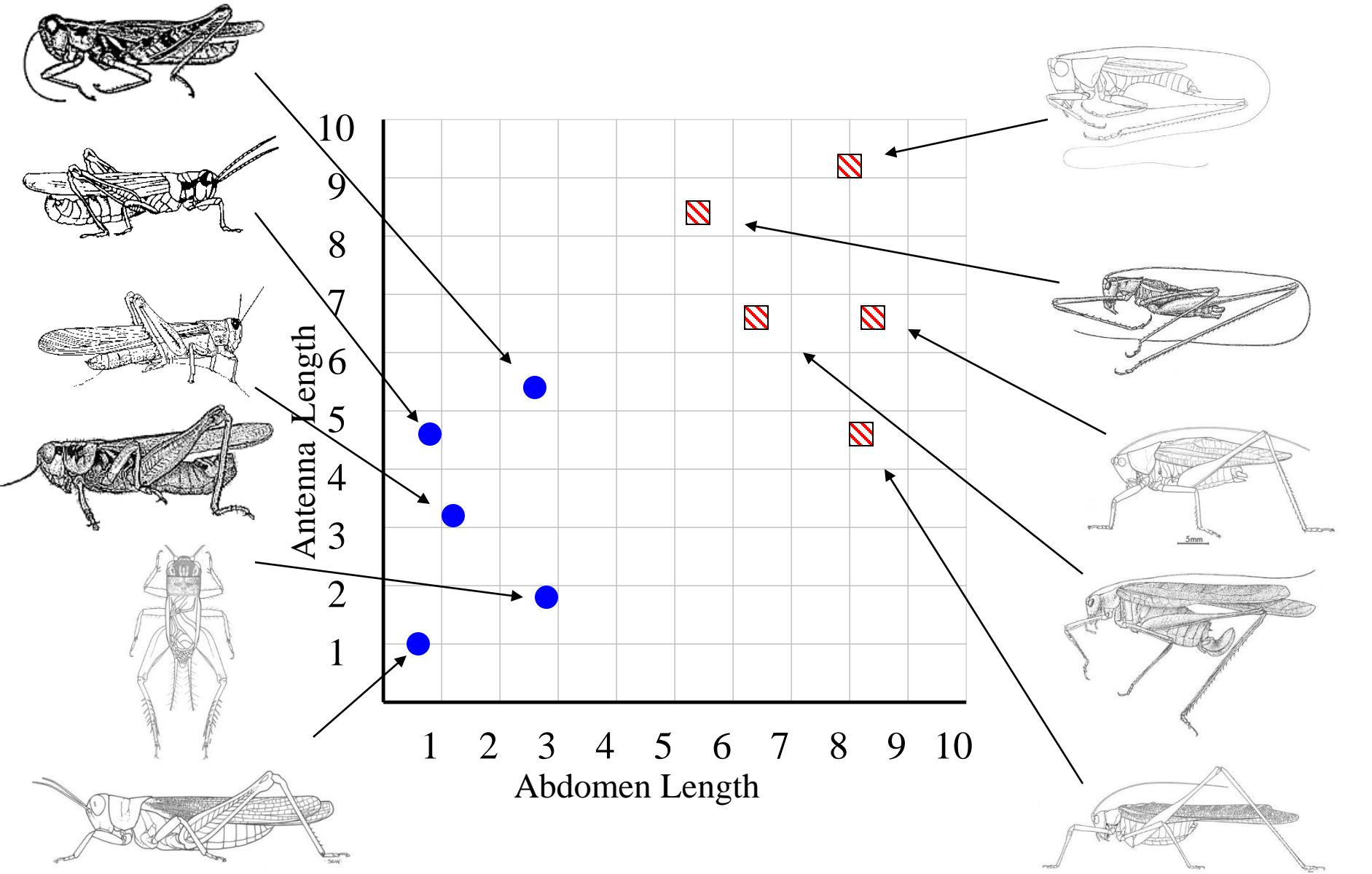


Figure 6.19 A two-class mixture model.

- Given set of instances without knowing which gaussian generated which instance, determine $\mu_A, \sigma_A, p_A, \mu_B, \sigma_B$ ($p_B = 1 - p_A$)

Grasshoppers

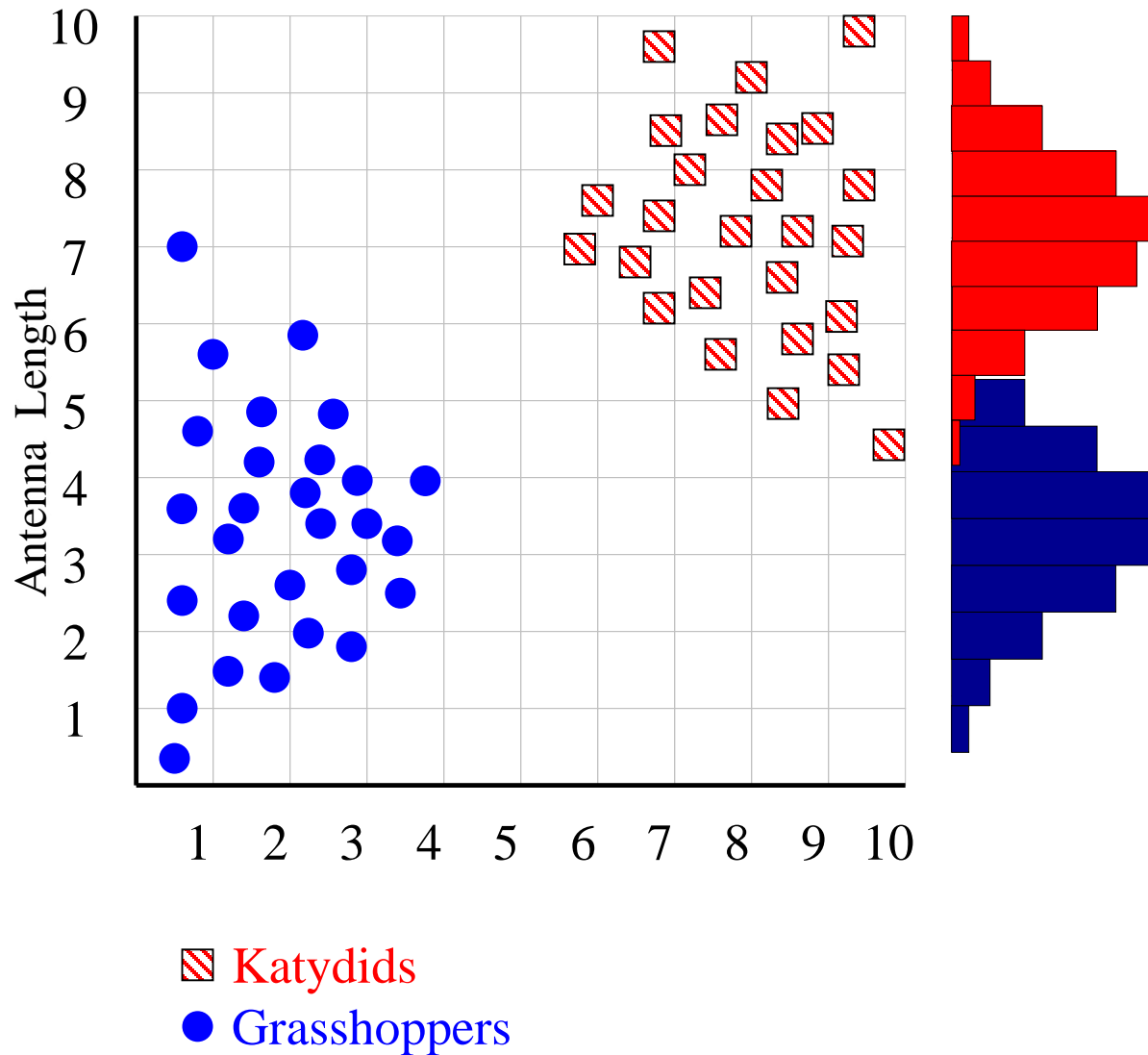
Katydids



Naïve Bayes classifier: the very foundation

Courtesy of Eammon Keogh, UCR,
eamonn@cs.ucr.edu

With a lot of data, we can build a histogram. Let us just build one for “Antenna Length” for now...



Mixed model cont'd

- Had we known from which distribution (A or B) a instance comes from, we could easily compute the two μ , σ , and p
- If we knew the five parametrs, we would assign a new x to cluster A if

$$\frac{\Pr[A | x]}{\Pr[B | x]} = \frac{f(x, \mu_A, \sigma_A)}{f(x, \mu_B, \sigma_B)} > 1$$

- where

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The EM algorithm

- Since we do not know any of the five parameters, we estimate and maximize:
 - Start with a random assignment of the 5
 - Compute cluster probabilities for each instance (Expectation step - expected cluster assignments)
 - Use these cluster assignments to compute the 5 parameters (Maximize step - maximize the likelihood of the distribution given the data)
- Note that the same algorithm, with label assignment instead of cluster assignment, can be used to assign labels to unlabeled data generated by a mixture model!

EM cont'd

- But when to stop?
- Essentially, when the learning curve flattens. Specifically, when the overall probability that the data comes from this model

$$\prod_i (p_A \Pr[x_i | A] + p_B \Pr[x_i | B])$$

(where the cluster probabilities are given by the $f(x, \mu, \sigma)$ starts to yield very small differences in a number of consecutive iterations

- in practice EM works with log-likelihoods to avoid multiplications
- K-means is a special case of EM

EM cont'd

- The framework is extended to mixtures of k gaussians (two-class to k -class, but k must be known)
- The framework is further easily extended to multiple attributes, under the assumption of independence of attributes...
- ...and further extended with dropping the independence assumption and replacing the standard deviation by the covariance matrix

EM cont'd

- Parameters: for n independent attributes, $2n$ parameters; for covariant attributes, $n+n(n+1)/2$ parameters: n means and the symmetric $n \times n$ covariance matrix
- For (independent) nominal attributes, EM is like Naïve Bayes: instead of normal distribution, k_v parameters per attribute are estimated, where v is the number of values of the attribute:
 - Expectation: determine the cluster (like the class in NB)
 - Maximization: like estimating NB priors (attribute-value probabilities) from data

Visualization of k-means and DBSCAN

- <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>