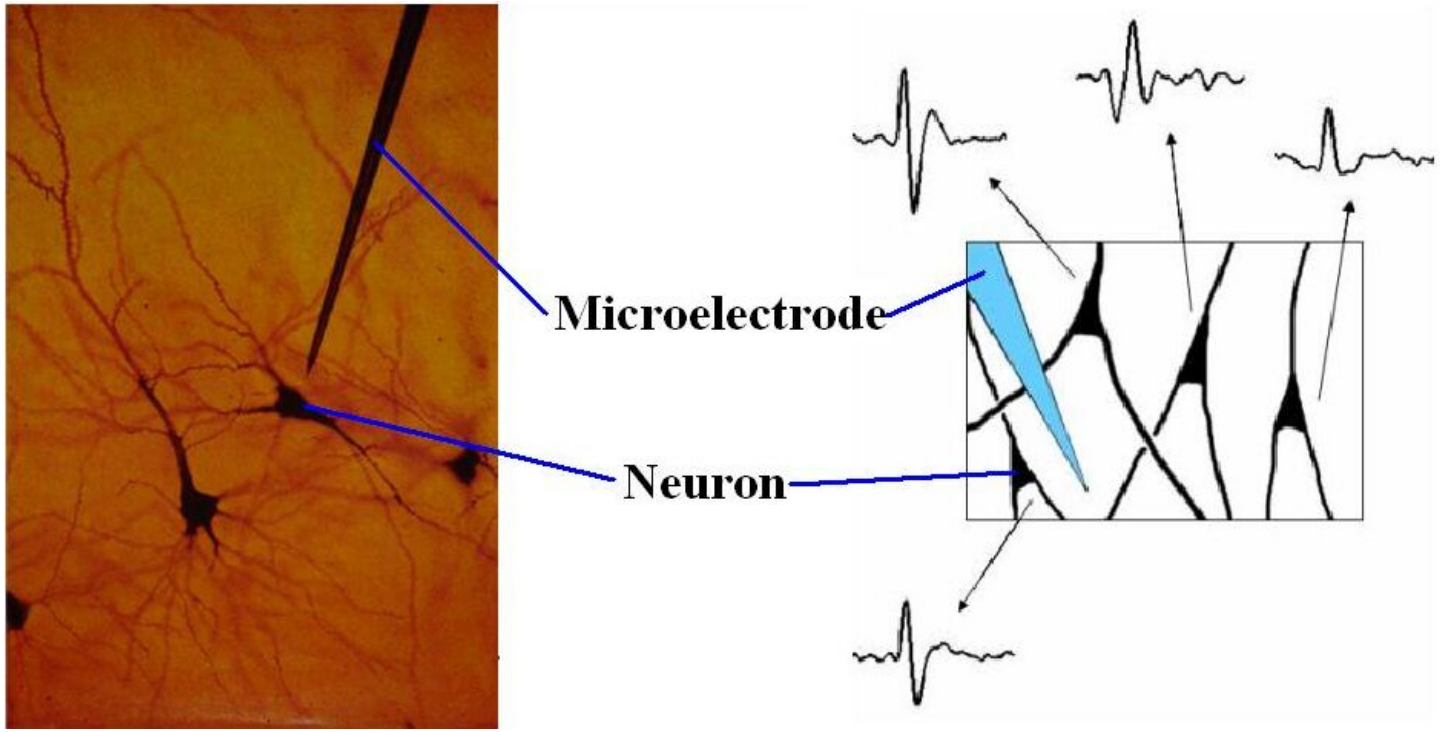
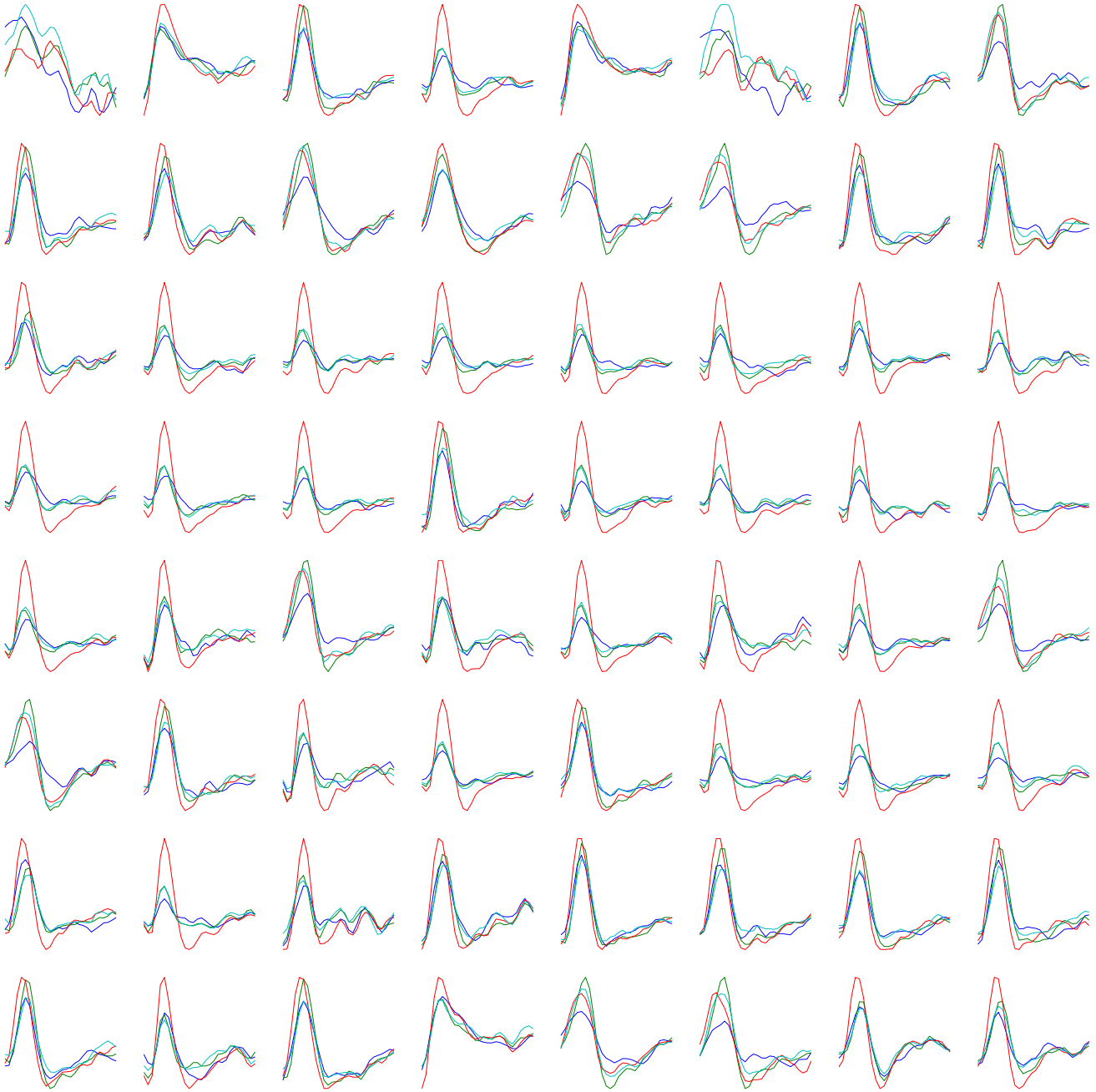
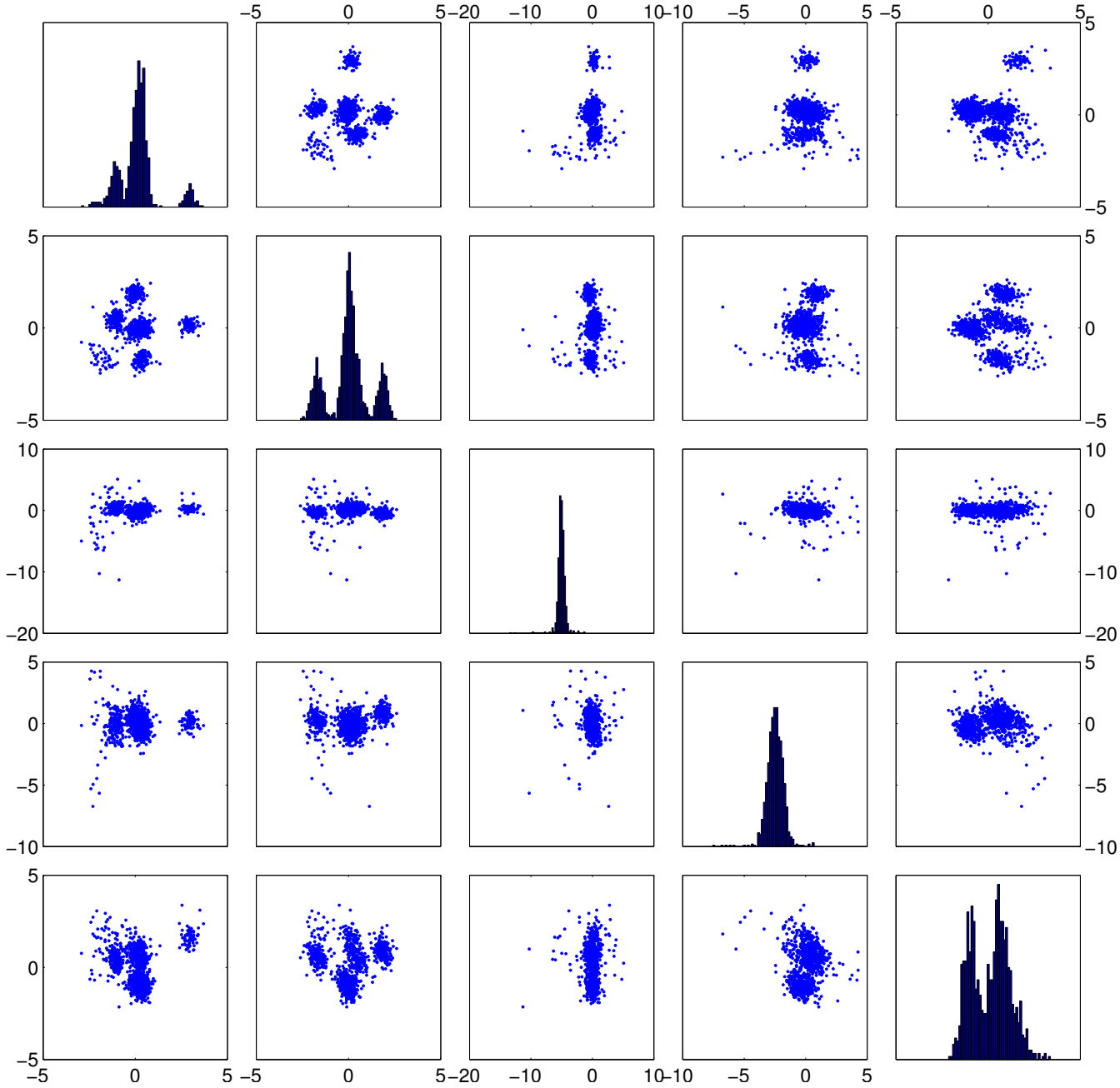
Neural Electroencephalography (EEG)



Neural Spike Waveforms



Pairs Plot of Principal Components



Clustering

- Many datasets consist of multiple heterogeneous subsets. Cluster analysis is a range of methods that reveal this heterogeneity by discovering clusters of similar points.
- Model-based clustering:
  - Each cluster is described using a probability model.
- Model-free clustering:
  - Defined by similarity among points within clusters (dissimilarity among points between clusters).
- Partition-based clustering methods:
  - Allocate points into K clusters.
  - The number of cluster is usually fixed beforehand or investigated for various values of K as part of the analysis.
- Hierarchy-based clustering methods:
  - Allocate points into clusters and clusters into super-clusters forming a hierarchy.
  - Typically the hierarchy forms a binary tree (a dendrogram) where each cluster has two “children”.

Hierarchical Clustering

- Hierarchically structured data can be found everywhere (measurements of different species and different individuals within species), hierarchical methods attempt to understand data by looking for clusters.
- There are two general strategies for generating hierarchical clusters. Both proceed by seeking to minimize some measure of dissimilarity.
  - Agglomerative / Bottom-Up / Merging
  - Divisive / Top-Down / Splitting

  Hierarchical clusters are generated where at each level, clusters are created by merging clusters at lower levels. This process can easily be viewed by a dendogram/tree.

Measuring Dissimilarity

To find hierarchical clusters, we need some way to measure the dissimilarity between clusters

- Given two points x_i and x_j, it is straightforward to measure their dissimilarity, say $d(x_i, x_j) = \|x_i - x_j\|_2$.
- It is unclear however how to extend this to measure dissimilarity between *clusters*, $D(C_i, C_j)$ for clusters C_i and C_j.

Many such proposals though no concensus as to which is best.

(a) *Single Linkage*

$$D(C_i, C_j) = \min_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

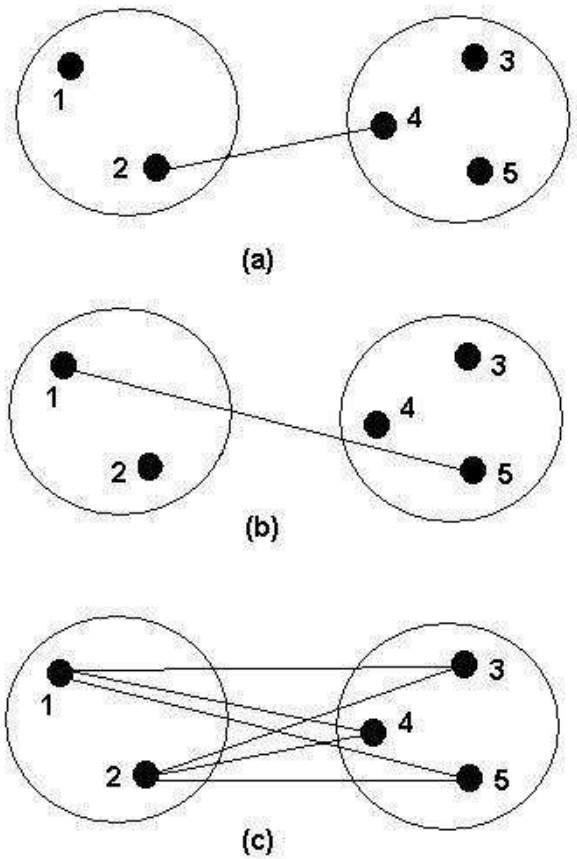
(b) *Complete Linkage*

$$D(C_i, C_j) = \max_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

(c) *Average Linkage*

$$D(C_i, C_j) = avg_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

Measuring Dissimilarity



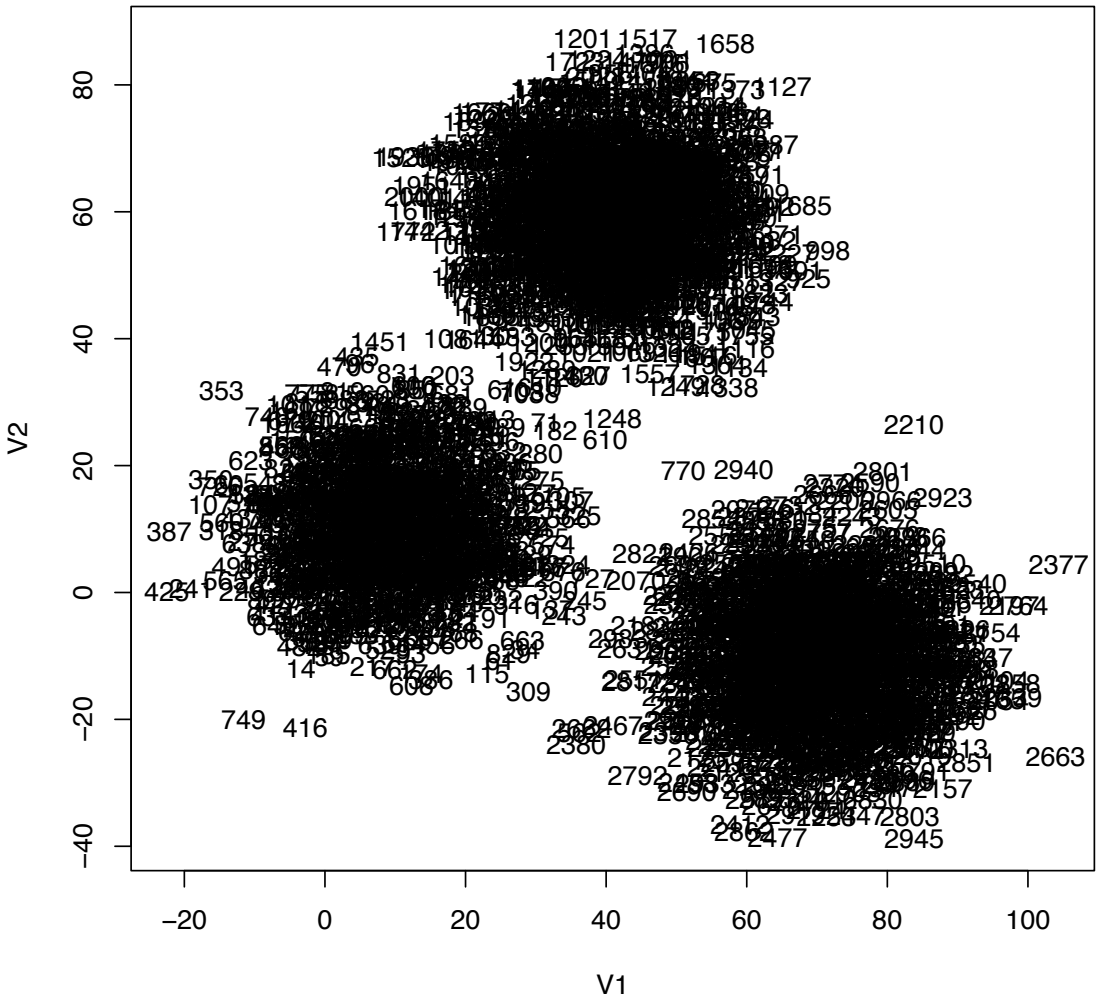
Cluster Distance

d24

d15

$$\frac{d13+d14+d15+d23+d24+d25}{6}$$

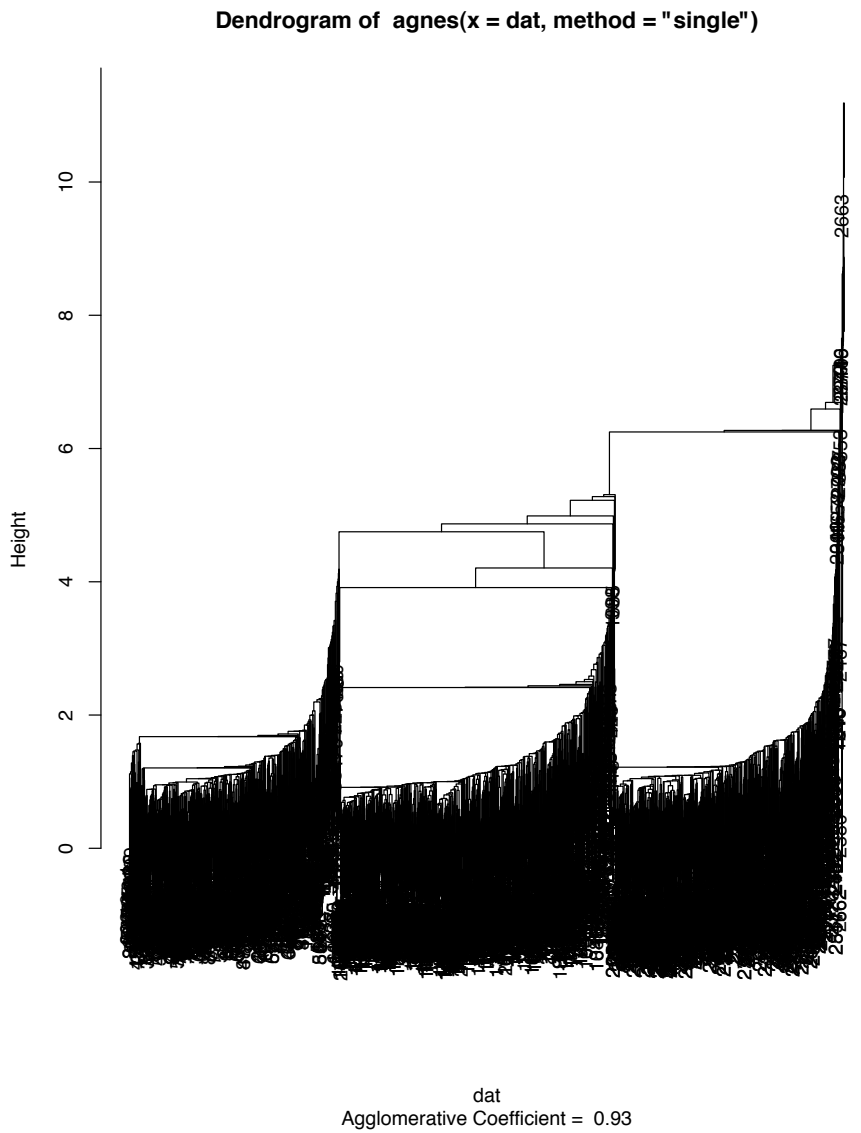
Hierarchical Clustering on Artificial Dataset



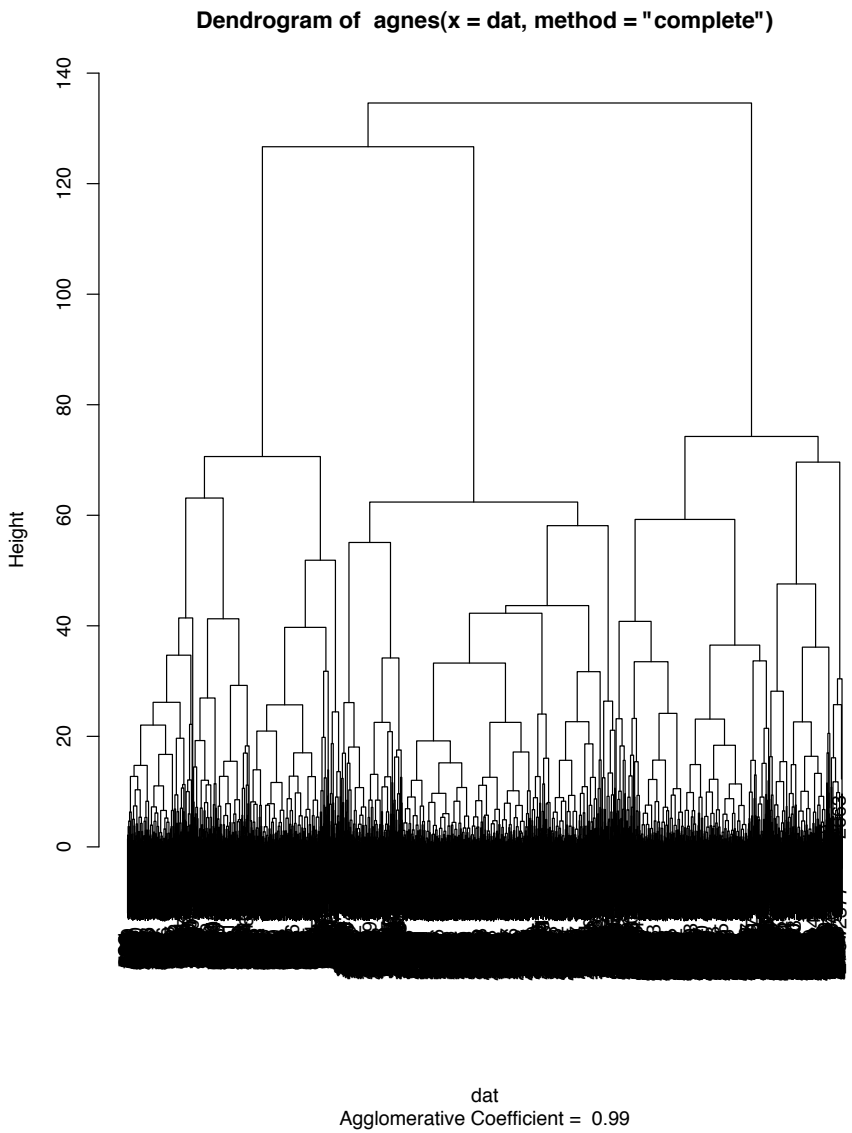
Hierarchical Clustering on Artificial Dataset

```
#start afresh
dat=xclara #3000 x 2
library(cluster)

#plot the data
plot(dat,type="n")
text(dat,labels=row.names(dat))

plot(agnes(dat,method="single"))
plot(agnes(dat,method="complete"))
plot(agnes(dat,method="average"))
```

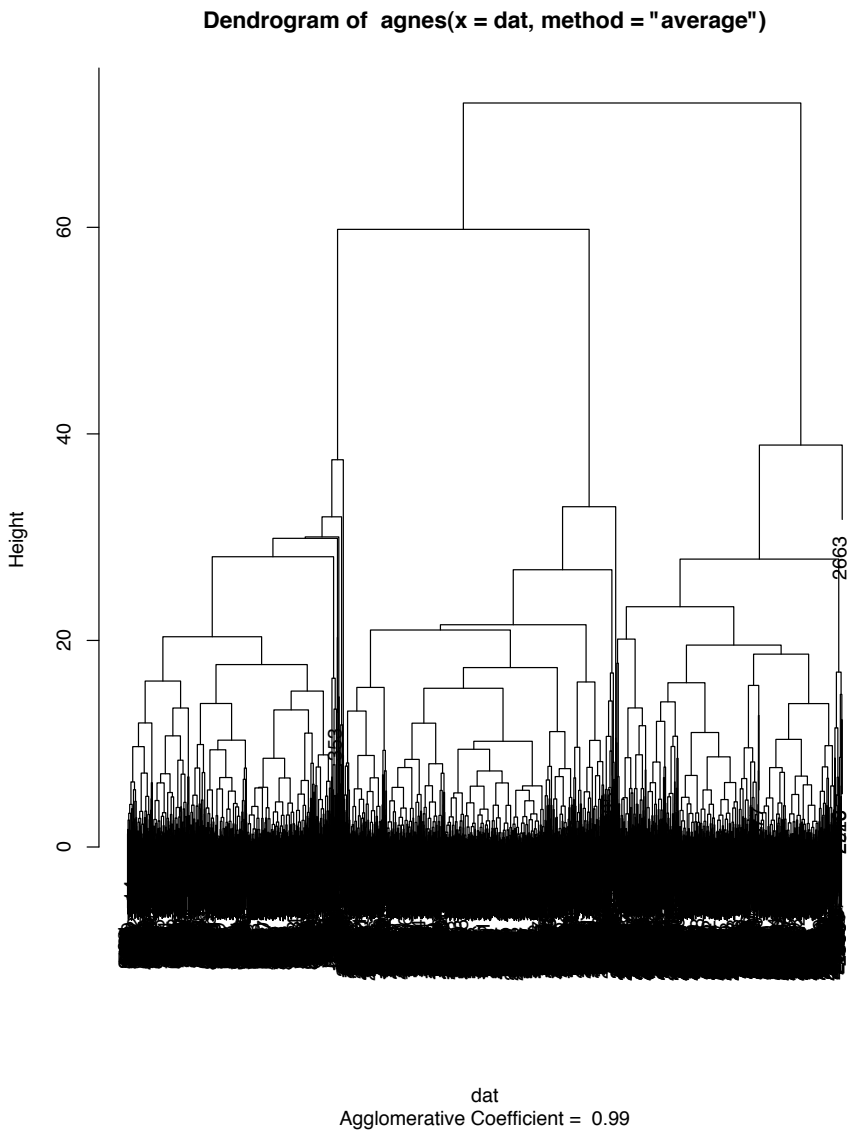
Hierarchical Clustering on Artificial Dataset



Hierarchical Clustering on Artificial Dataset



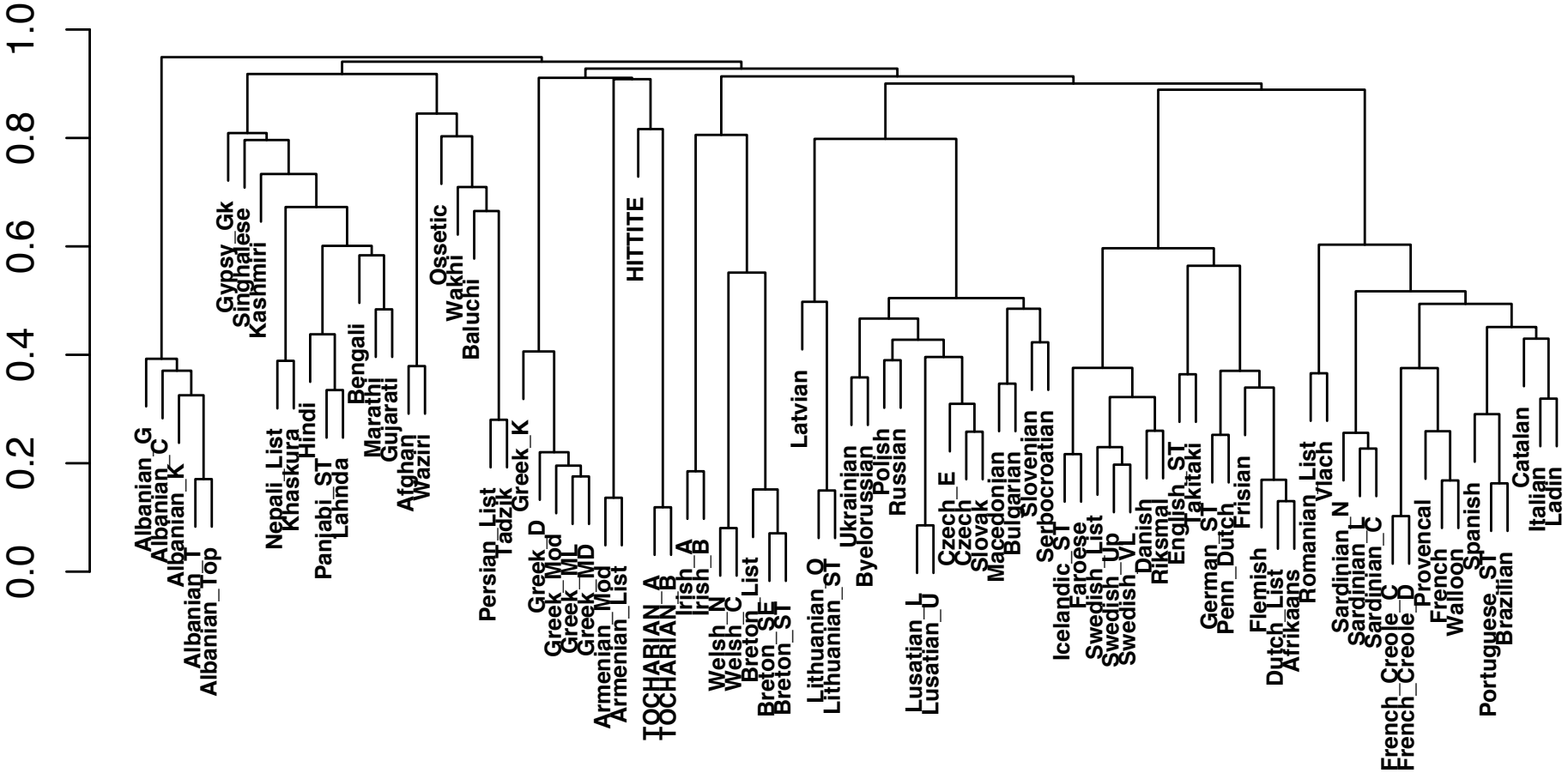
Hierarchical Clustering on Artificial Dataset



Using Dendograms

- Different ways of measuring dissimilarity result in different trees.
- Dendograms are useful for getting a feel for the structure of high-dimensional data though they don't represent distances between observations well.
- Dendograms show hierarchical clusters with respect to increasing values of dissimilarity between clusters, cutting a dendogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects. Cutting horizontally effectively reveals the state of the clustering algorithm when the dissimilarity value between clusters is no more than the value cut at.
- Despite the simplicity of this idea and the above drawbacks, hierarchical clustering methods provide users with interpretable dendograms that allow clusters in high-dimensional data to be better understood.

Hierarchical Clustering on Indo-European Languages



K-means

Partition-based methods seek to divide data points into a pre-assigned number of clusters $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

For each cluster, represent it using a *prototype* or *cluster centre* μ_k.
We can measure the quality of a cluster with its *within-cluster deviance*

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is given by the total within-cluster deviance:

$$W = \sum_{k=1}^{K} W(C_k, \mu_k).$$

The overall objective is to choose both the cluster centres and allocation of points to minimize the *objective function*.

K-means

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2$$

where $c_i = k$ if and only if $i \in C_k$.

- Given partition $\{C_k\}$, we can find the optimal prototypes easily by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

But joint minimization over both is computationally difficult.

K-means

The K-means algorithm is a well-known method that *locally optimizes* the objective function W.
Iterative and alternating minimization.

1. Randomly fix K cluster centres $\mu_1, \ldots, \mu_K$.
2. For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centre,

$$c_i := \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

3. Set $C_k := \{i : c_i = k\}$ for each k.
4. Move cluster centres $\mu_1, \ldots, \mu_K$ to the average of the new clusters:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

5. Repeat steps 2 to 4 until there is no more changes.
6. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$ at the end.

K-means

Some notes about the K-means algorithm.

- *The algorithm stops in a finite number of iterations.* Between steps 2 and 3, W either stays constant or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- *The K-means algorithm need not converge to global optimum.* K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration. Typically perform a number of runs from different configurations, and pick best clustering.

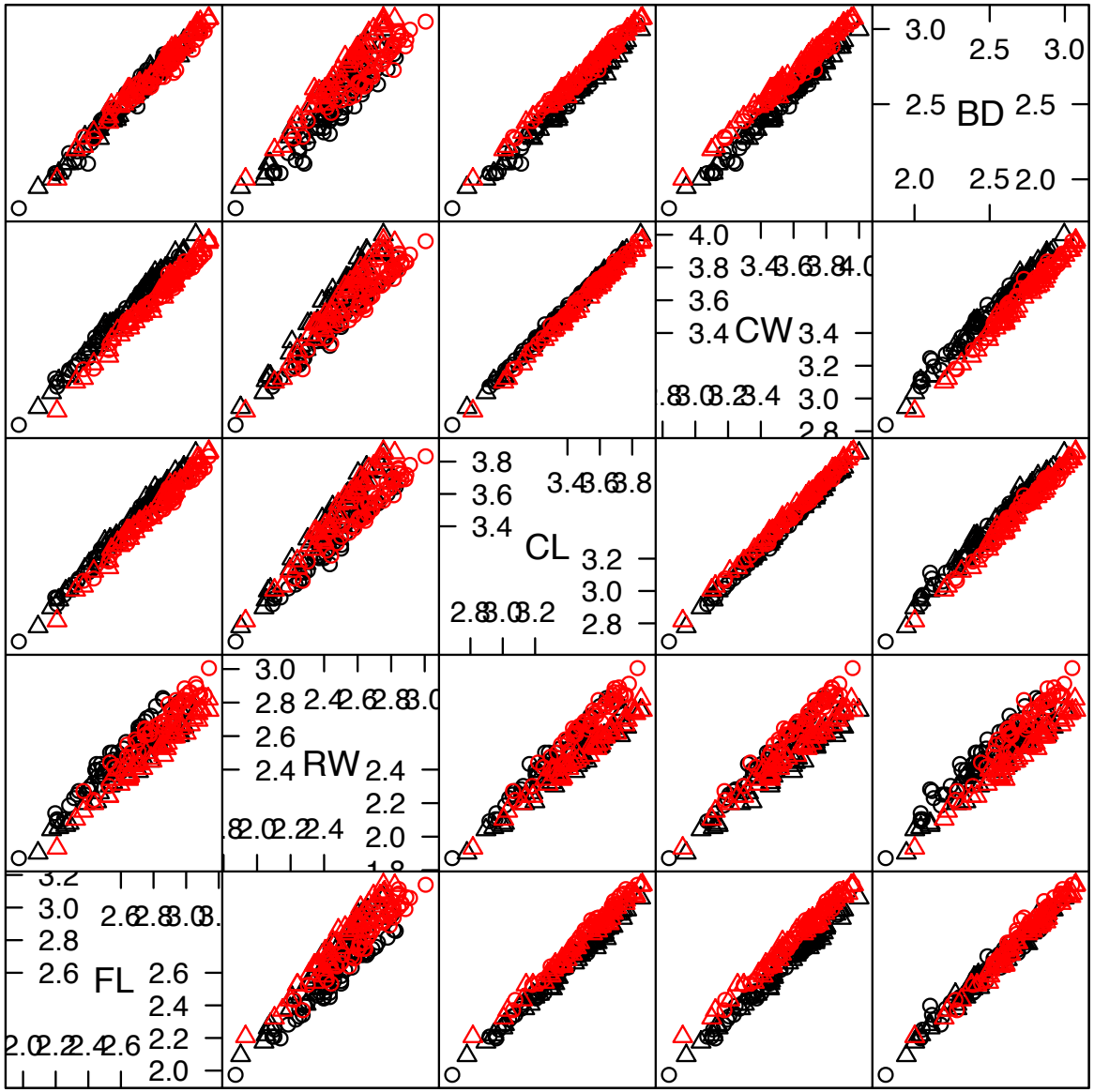
K-means on Crabs

Looking at the Crabs data again.

```
library(MASS)
library(lattice)
data(crabs)

splom(~log(crabs[,4:8]),
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

K-means on Crabs

circle/triangle is gender, black/red is species



Scatter Plot Matrix

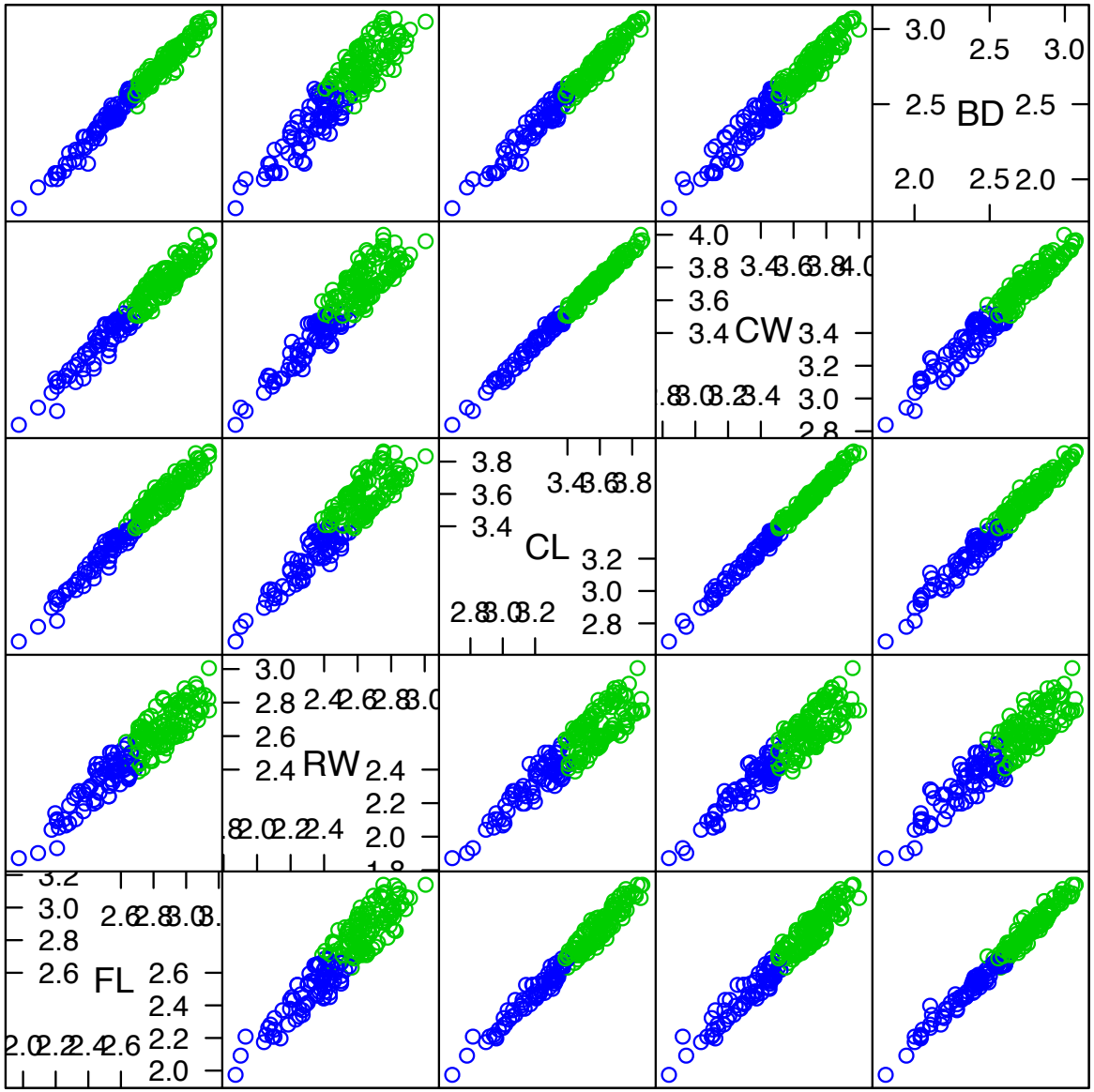
K-means on Crabs

Apply K-means with 2 clusters and plot results.

```
cl <- kmeans( log(crabs[,4:8]), 2, nstart=1, iter.max=10)

splom(~log(crabs[,4:8]),
      col=cl$cluster+2,
      main="blue/green is cluster finds big/small")
```

K-means on Crabs

blue/green is cluster finds big/small



Scatter Plot Matrix

K-means on Crabs

'Whiten' or 'sphere'[1] the data using PCA.

```
pcp <- princomp( log(crabs[,4:8]) )
spc <- pcp$scores %*% diag(1/pcp$sdev)
splom( ~spc[,1:3],
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

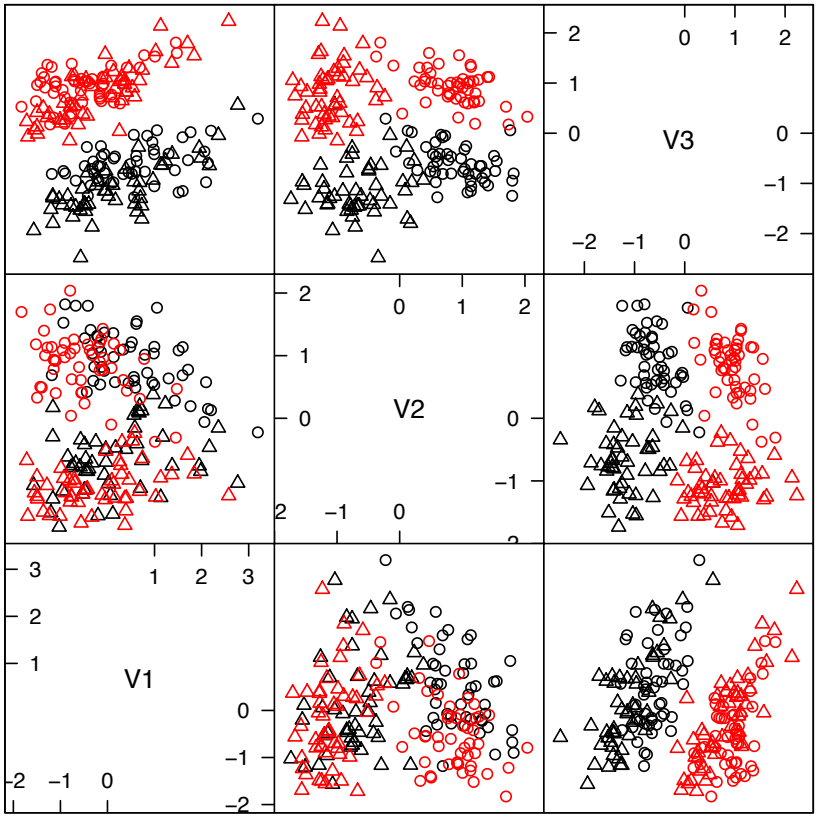
And apply K-means again.

```
cl <- kmeans(spc, 2, nstart=1, iter.max=20)
splom( ~spc[,1:3],
      col=cl$cluster+2, main="blue/green is cluster")
```

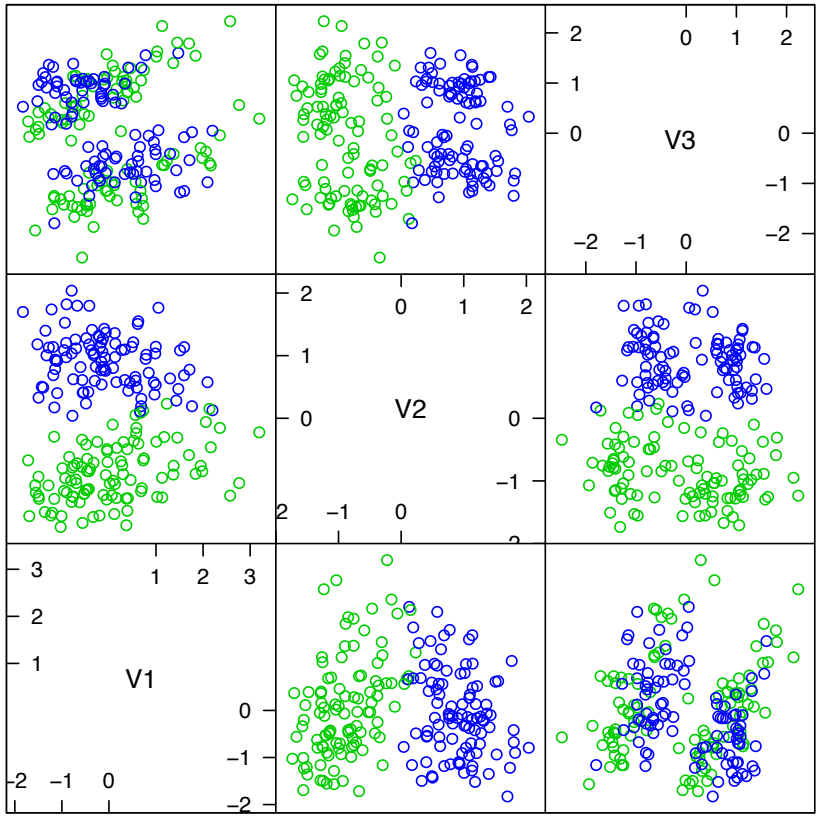
[1] Apply a linear transformation so that covariance matrix is identity.

K-means on Crabs

circle/triangle is gender, black/red is species



blue/green is cluster

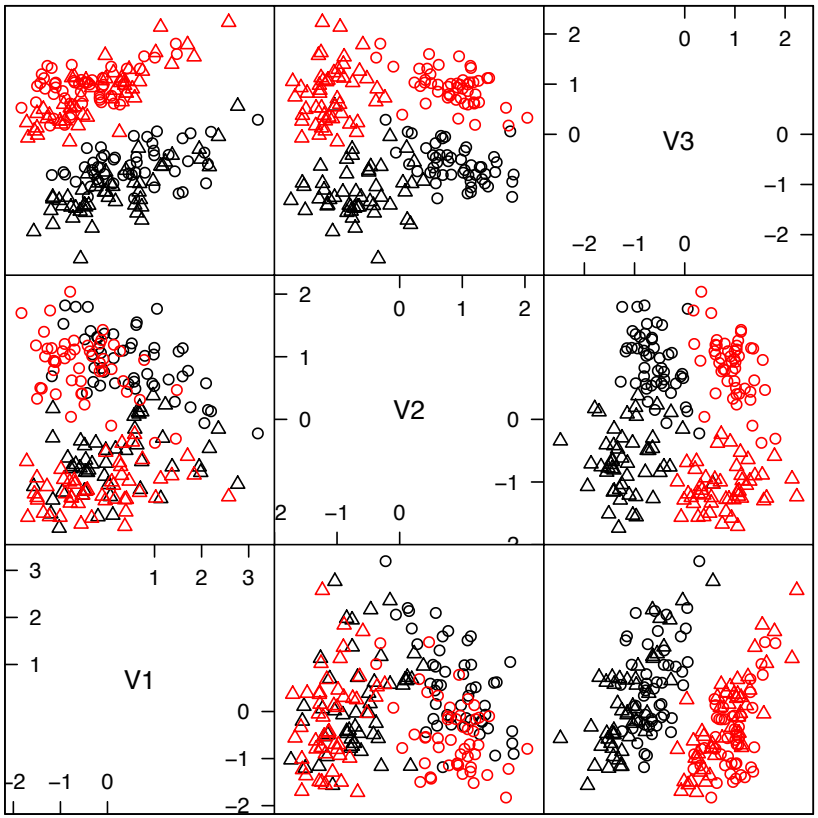


Discovers gender difference...
Results depends crucially on sphering the data first.

K-means on Crabs

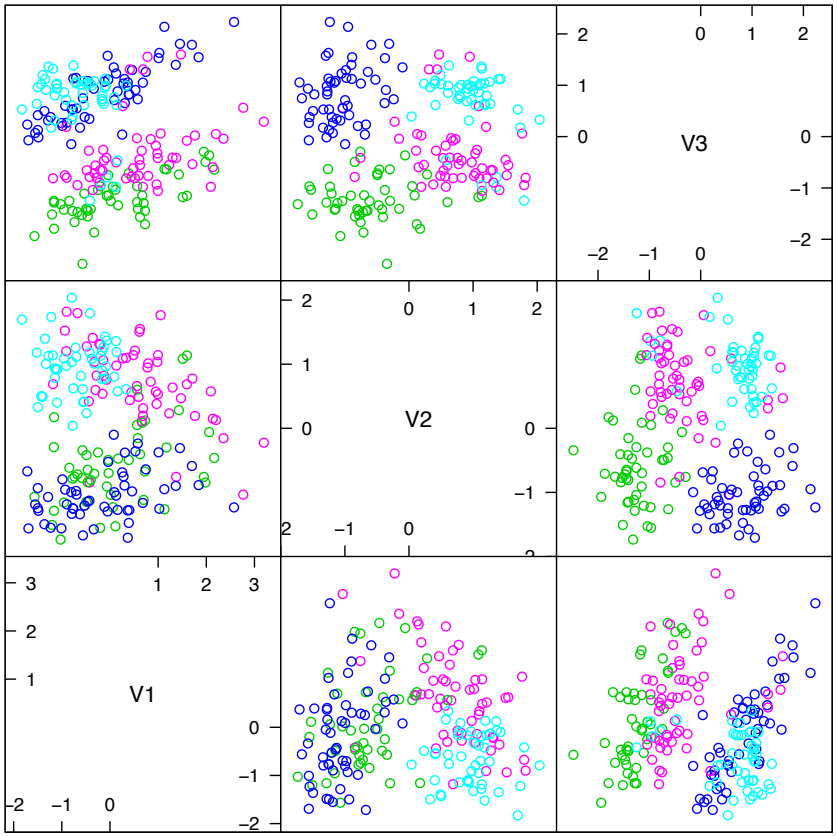
Using 4 cluster centers.

circle/triangle is gender, black/red is species



Scatter Plot Matrix

colors are clusters

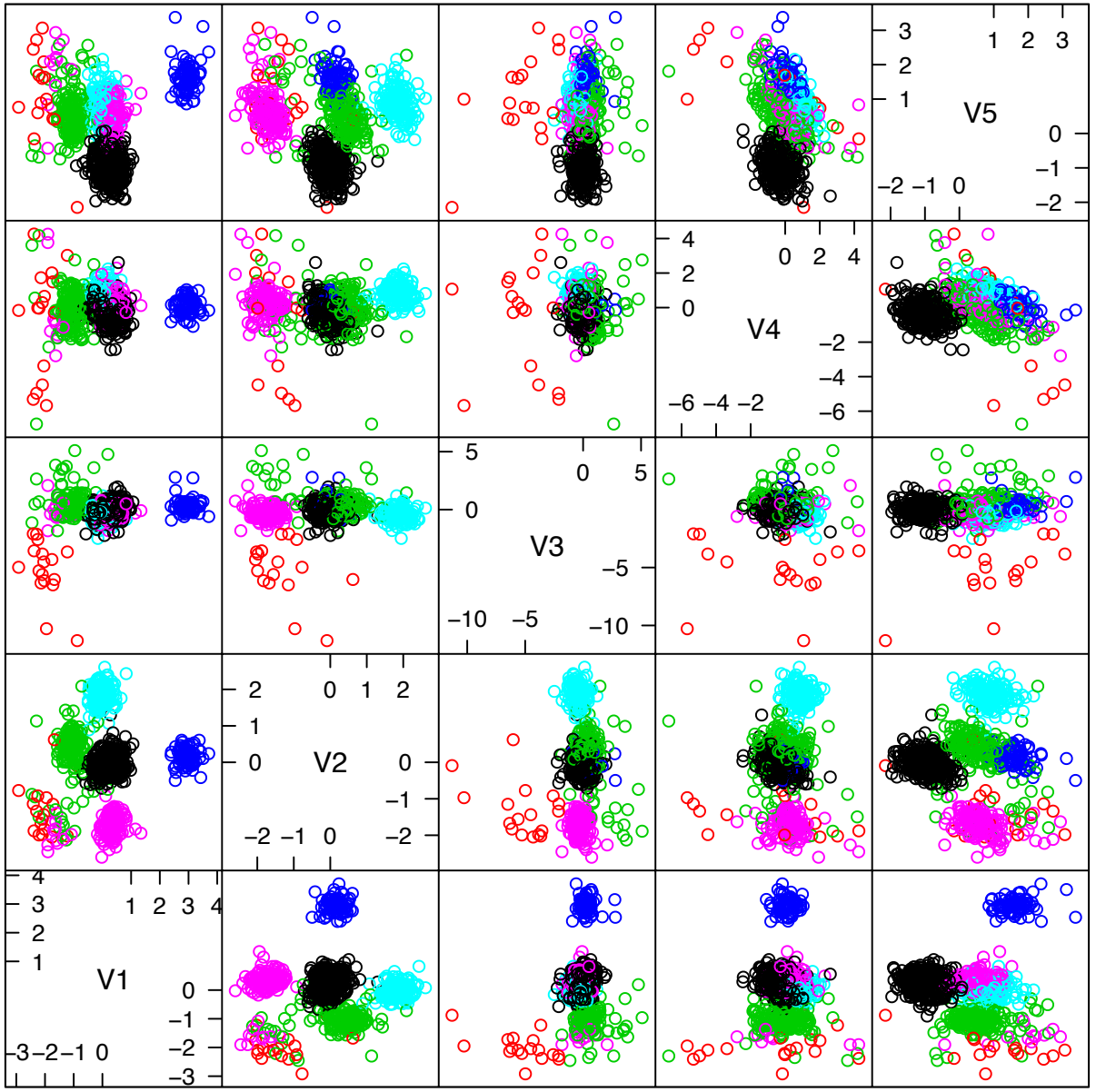


Scatter Plot Matrix

K-means on Spike Waveforms

```
library(MASS)
library(lattice)
spikespca <- read.table("spikes.txt")
cl <- kmeans(data,6,nstart=20)
splom(data,col=cl$cluster)
```

K-means on Spike Waveforms



Scatter Plot Matrix

Stochastic Optimization

- Each iteration of K-means requires a pass through whole dataset. In extremely large datasets, this can be computationally prohibitive.
- Stochastic optimization: update cluster means after assigning each data point to the closest cluster.
- Repeat for $t = 1, 2, \ldots$ until satisfactory convergence:
  1. Pick data item x_i either randomly or in order.
  2. Assign x_i to the cluster with the nearest centre,

$$c_i := \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

  3. Update cluster centre:

$$\mu_k := \mu_k + \alpha_t(x_i - \mu_k)$$

  where $\alpha_t > 0$ are *step sizes*.
- Algorithm stochastically minimizes the objective function. Convergence requires slowly decreasing step sizes:

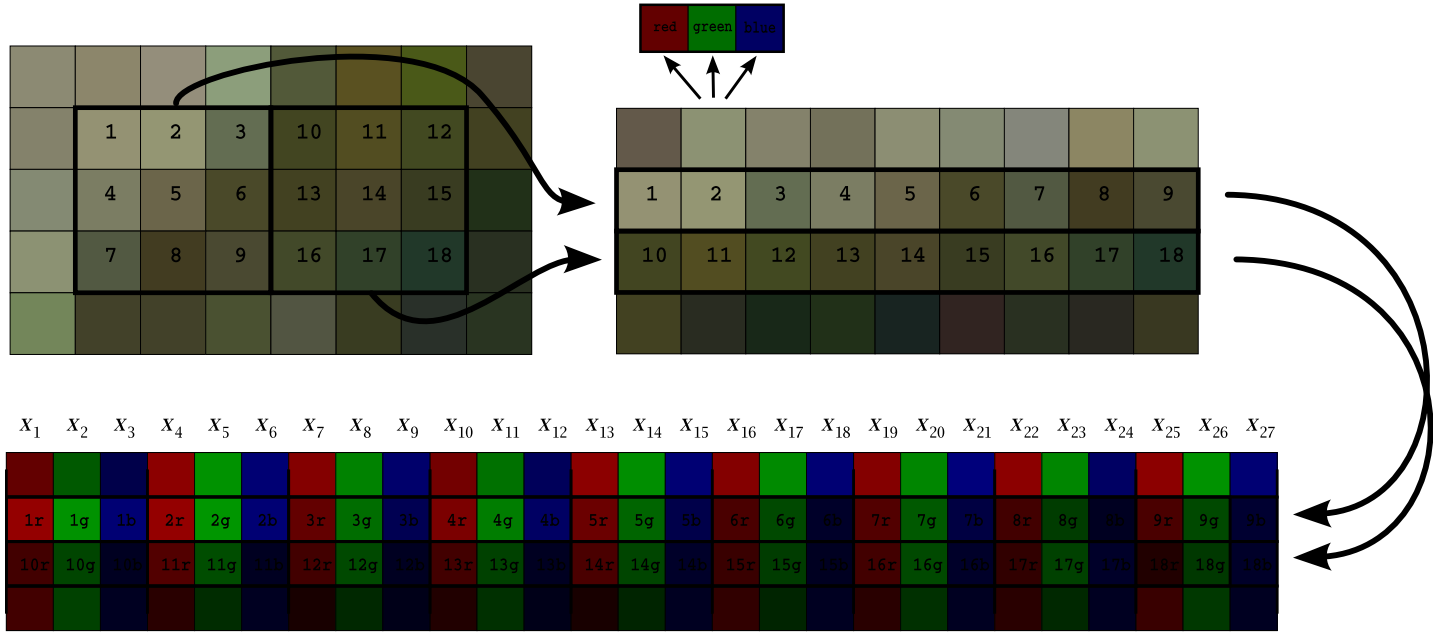
$$\sum_{t=1}^{\infty} \alpha_t = \infty \qquad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Vector Quantization

- A related algorithm developed in the signal processing literature for *lossy data compression*.
- If $K \ll n$, we can store the *codebook* of *codewords* $\mu_1, \ldots, \mu_K$, and each vector x_i is encoded using c_i, which only requires $\lceil \log K \rceil$ bits.
- As with K-means, K must be specified. Increasing K improves the quality of the compressed image but worsens the data compression rate, so there is a clear tradeoff.
- Some audio and video codecs use this method.
- Stochastic optimization algorithm for K-means was originally developed for VQ.

VQ Image Compression

3×3 block VQ: View each block of 3×3 pixels as single observation



VQ Image Compression

Original image (24 bits/pixel, uncompressed size 1,402 kB)

VQ Image Compression

Codebook length 1024 (1.11 bits/pixel, total size 88kB)

VQ Image Compression

Codebook length 128 (0.78 bits/pixel, total size 50kB)

VQ Image Compression

Codebook length 16 (0.44 bits/pixel, total size 27kB)

K-means Additional Comments

- *Sensitivity to distance measure.* Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance,

$$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

  where M is positive semi-definite matrix, e.g. sample covariance.
- *Other partition based methods.* There are many other partition based methods that employ related ideas. For example K-medoids differs from K-means in requiring cluster centres μ_i to be an observation x_i[2], K-medians (use median in each dimension) and K-modes (use mode).
- *Determination of K.* The K-means objective will always improve with larger number of clusters K. Determination of K requires an additional *regularization* criterion. E.g., in DP-means[3], use

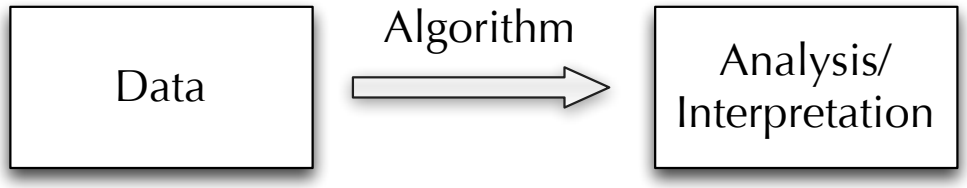
$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K$$

[2] See also Affinity propagation.

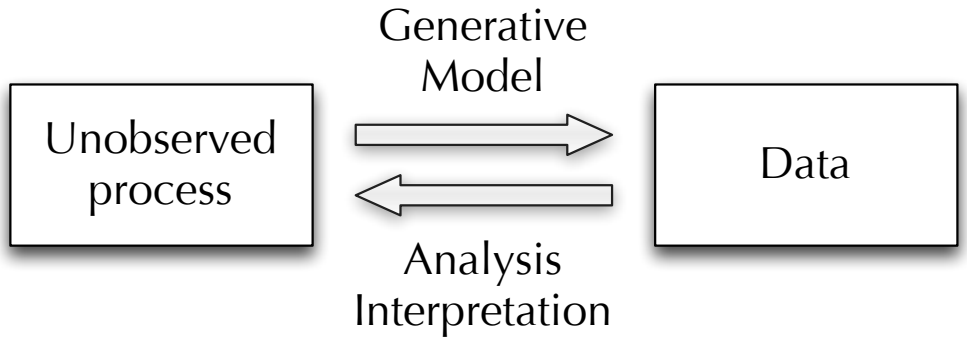
[3] DP-means paper.

Probabilistic Methods

- Algorithmic approach:



- Probabilistic modelling approach:



Mixture Models

- Mixture models suppose that our dataset was created by sampling iid from K distinct populations (called *mixture components*).
- Typical samples in population k can be modelled using a distribution $F(\phi_k)$ with density $f(x|\phi_k)$. For a concrete example, consider a Gaussian with unknown mean ϕ_k and known symmetric covariance $\sigma^2 I$,

$$f(x|\phi_k) = |2\pi\sigma^2|^{-\frac{p}{2}} \exp\left(-\frac{1}{2\sigma^2}\|x-\phi_k\|_2^2\right).$$

- Generative process: for $i = 1, 2, \ldots, n$:
  - First determine which population item i came from (independently):

$$Z_i \sim \text{Discrete}(\pi_1, \ldots, \pi_K) \qquad \text{i.e. } \mathbb{P}(Z_i = k) = \pi_k$$

    where *mixing proportions* are $\pi_k \geq 0$ for each k and $\sum_{k=1}^K \pi_k = 1$.
  - If $Z_i = k$, then $X_i = (X_{i1}, \ldots, X_{ip})^\top$ is sampled (independently) from corresponding population distribution:

$$X_i | Z_i = k \sim F(\phi_k)$$

- We observe that $X_i = x_i$ for each i, and would like to learn about the unknown parameters of the process.

Mixture Models

- Unknowns to learn given data are
  - *Parameters*: $\pi_1, \ldots, \pi_K$, $\phi_1, \ldots, \phi_K$, as well as
  - *Latent variables*: $z_1, \ldots, z_K$.
- The joint probability over all cluster indicator variables $\{Z_i\}$ are:

$$p_Z((z_i)_{i=1}^n) = \prod_{i=1}^n \pi_{z_i} = \prod_{i=1}^n \prod_{k=1}^K \pi_k^{\mathbb{1}(z_i=k)}$$

- The joint density at observations $X_i = x_i$ given $Z_i = z_i$ are:

$$p_X((x_i)_{i=1}^n | (Z_i = z_i)_{i=1}^n) = \prod_{i=1}^n \prod_{k=1}^K f(x_i|\phi_k)^{\mathbb{1}(z_i=k)}$$

- So the joint probability/density[4] is:

$$p_{X,Z}((x_i, z_i)_{i=1}^n) = \prod_{i=1}^n \prod_{k=1}^K (\pi_k f(x_i|\phi_k))^{\mathbb{1}(z_i=k)}$$

[4] In this course we will treat probabilities and densities equivalently for notational simplicity. In general, the quantity is a density with respect to the product base measure, where the base measure is the counting measure for discrete variables and Lebesgue for continuous variables.

Mixture Models - Posterior Distribution

- Suppose we know the parameters $(\pi_k, \phi_k)_{k=1}^K$.
- Z_i is a random variable, so the posterior distribution given data set $\mathbf{X}$ tells us what we know about it:

$$Q_{ik} := p(Z_i = k|x_i) = \frac{p(Z_i = k, x_i)}{p(x_i)} = \frac{\pi_k f(x_i|\phi_k)}{\sum_{j=1}^K \pi_j f(x_i|\phi_j)}$$

where the marginal probability is:

$$p(x_i) = \sum_{j=1}^K \pi_j f(x_i|\phi_j)$$

- The posterior probability Q_{ik} of $Z_i = k$ is called the *responsibility* of mixture component k for data point x_i.
- The posterior distribution *softly partitions* the dataset among the k components.

Mixture Models - Maximum Likehood

- How can we learn about the parameters $\theta = (\pi_k, \phi_k)_{k=1}^K$ from data?
- Standard statistical methodology asks for the *maximum likelihood* estimator (MLE).
- The log likelihood is the log marginal probability of the data:

$$\ell((\pi_k, \phi_k)_{k=1}^K) := \log p((x_i)_{i=1}^n | (\pi_k, \phi_k)_{k=1}^K) = \sum_{i=1}^n \log \sum_{j=1}^K \pi_j f(x_i|\phi_j)$$

$$\begin{aligned}\nabla_{\phi_k} \ell((\pi_k, \phi_k)_{k=1}^K) &= \sum_{i=1}^n \frac{\pi_k f(x_i|\phi_k)}{\sum_{j=1}^K \pi_j f(x_i|\phi_j)} \nabla_{\phi_k} \log f(x_i|\phi_k) \\ &= \sum_{i=1}^n Q_{ik} \nabla_{\phi_k} \log f(x_i|\phi_k)\end{aligned}$$

- A difficult equation to solve, as Q_{ik} depends implicitly on ϕ_k...

Mixture Models - Maximum Likehood

$$\sum_{i=1}^{n} Q_{ik} \nabla_{\phi_k} \log f(x_i|\phi_k) = 0$$

- What if we ignore the dependence of Q_{ik} on the parameters?
- Taking the mixture of Gaussian with covariance $\sigma^2 I$ as example,

$$\sum_{i=1}^{n} Q_{ik} \nabla_{\phi_k} \left(-\frac{p}{2} |2\pi\sigma^2| - \frac{1}{2\sigma^2} \|x_i - \phi_k\|_2^2 \right)$$

$$= \frac{1}{\sigma^2} \sum_{i=1}^{n} Q_{ik}(x_i - \phi_k) = \frac{1}{\sigma^2} \left(\left(\sum_{i=1}^{n} Q_{ik} x_i \right) - \phi_k \left(\sum_{i=1}^{n} Q_{ik} \right) \right) = 0$$

$$\phi_k^{MLE?} = \frac{\sum_{i=1}^{n} Q_{ik} x_i}{\sum_{i=1}^{n} Q_{ik}}$$

Mixture Models - Maximum Likehood

- The estimate is a weighted average of data points, where the estimated mean of cluster k uses its responsibilities to data points as weights.

$$\phi_k^{MLE?} = \frac{\sum_{i=1}^{n} Q_{ik} x_i}{\sum_{i=1}^{n} Q_{ik}}$$

- Makes sense: Suppose we knew that data point x_i came from population z_i. Then $Q_{iz_i} = 1$ and $Q_{ik} = 0$ for $k \neq z_i$ and:

$$\pi_k^{MLE} = \frac{\sum_{i:z_i=k} x_i}{\sum_{i:z_i=k} 1}$$

- Our best guess of the originating population is given by Q_{ik}.

Mixture Models - Maximum Likehood

- For the mixing proportions, we can similarly derive an estimator.
- Include a Lagrange multiplier λ to enforce constraint $\sum_k \pi_k = 1$.

$$\begin{aligned}
&\nabla_{\log \pi_k} \left(\ell((\pi_k, \phi_k)_{k=1}^K) - \lambda(\textstyle\sum_{k=1}^K \pi_k - 1) \right) \\
&= \sum_{i=1}^n \frac{\pi_k f(x_i|\phi_k)}{\sum_{j=1}^K \pi_j f(x_i|\phi_j)} - \lambda\pi_k \\
&= \sum_{i=1}^n Q_{ik} - \lambda\pi_k \pi_k^{MLE?} = \frac{\sum_{i=1}^n Q_{ik}}{n}
\end{aligned}$$

- Again makes sense: the estimate is simply (our best guess of) the proportion of data points coming from population k.

Mixture Models - The EM Algorithm

- Putting all the derivations together, we get an iterative algorithm for learning about the unknowns in the mixture model.
- Start with some initial parameters $(\pi_k^{(0)}, \phi_l^{(0)})_{k=1}^K$.
- Iterate for $t = 1, 2, \ldots$:
  - *Expectation Step*:

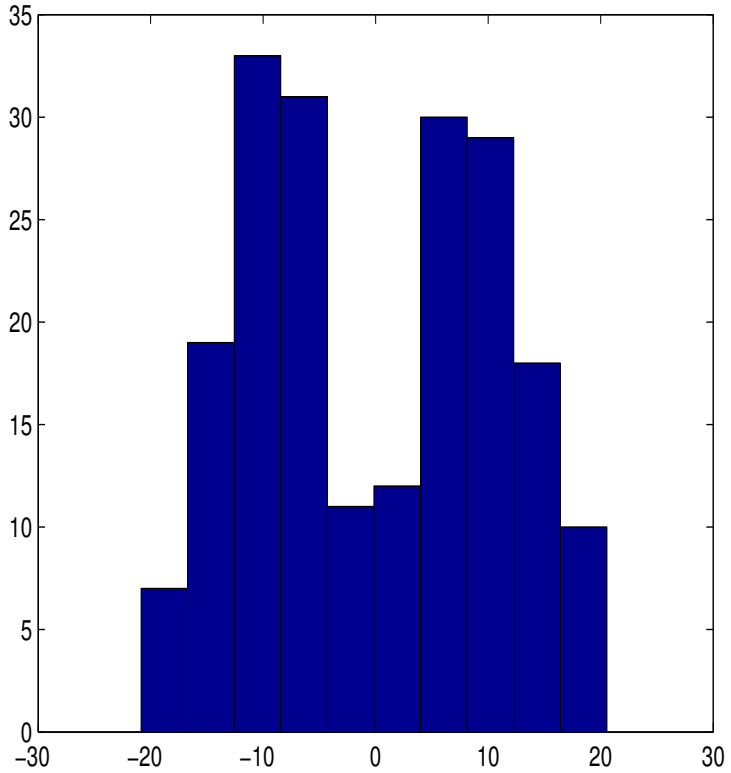
$$Q_{ik}^{(t)} := \frac{\pi_k^{(t-1)} f(x_i|\phi_k^{(t-1)})}{\sum_{j=1}^K \pi_j^{(t-1)} f(x_i|\phi_j^{(t-1)})}$$

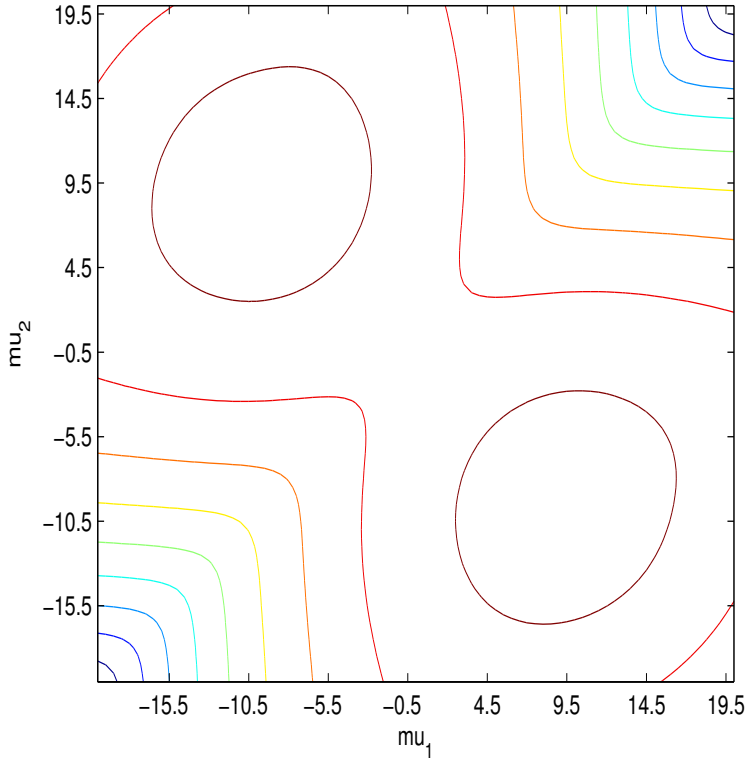
  - *Maximization Step*:

$$\pi_k^{(t)} = \frac{\sum_{i=1}^n Q_{ik}^{(t)}}{n} \qquad \phi_k^{(t)} = \frac{\sum_{i=1}^n Q_{ik}^{(t)} x_i}{\sum_{i=1}^n Q_{ik}^{(t)}}$$

- Will the algorithm converge?
- What does it converge to?

Likelihood Surface for a Simple Example

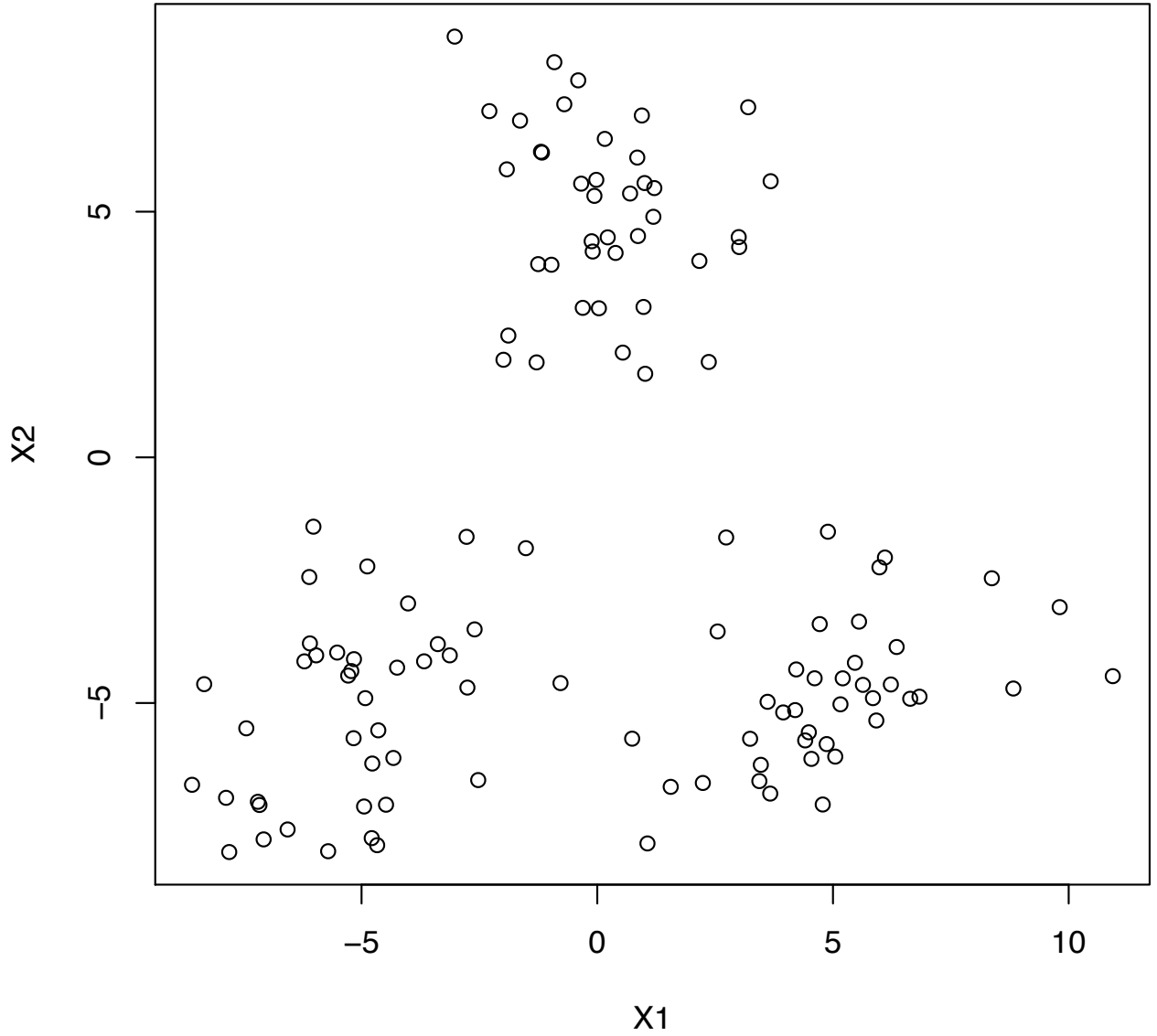




(left) $n = 200$ data points from a mixture of two 1D Gaussians with $\pi_1 = \pi_2 = 0.5$, $\sigma = 5$ and $\mu_1 = 10, \mu_2 = -10$.
(right) Log likelihood surface $\ell(\mu_1, \mu_2)$, all the other parameters being assumed known.

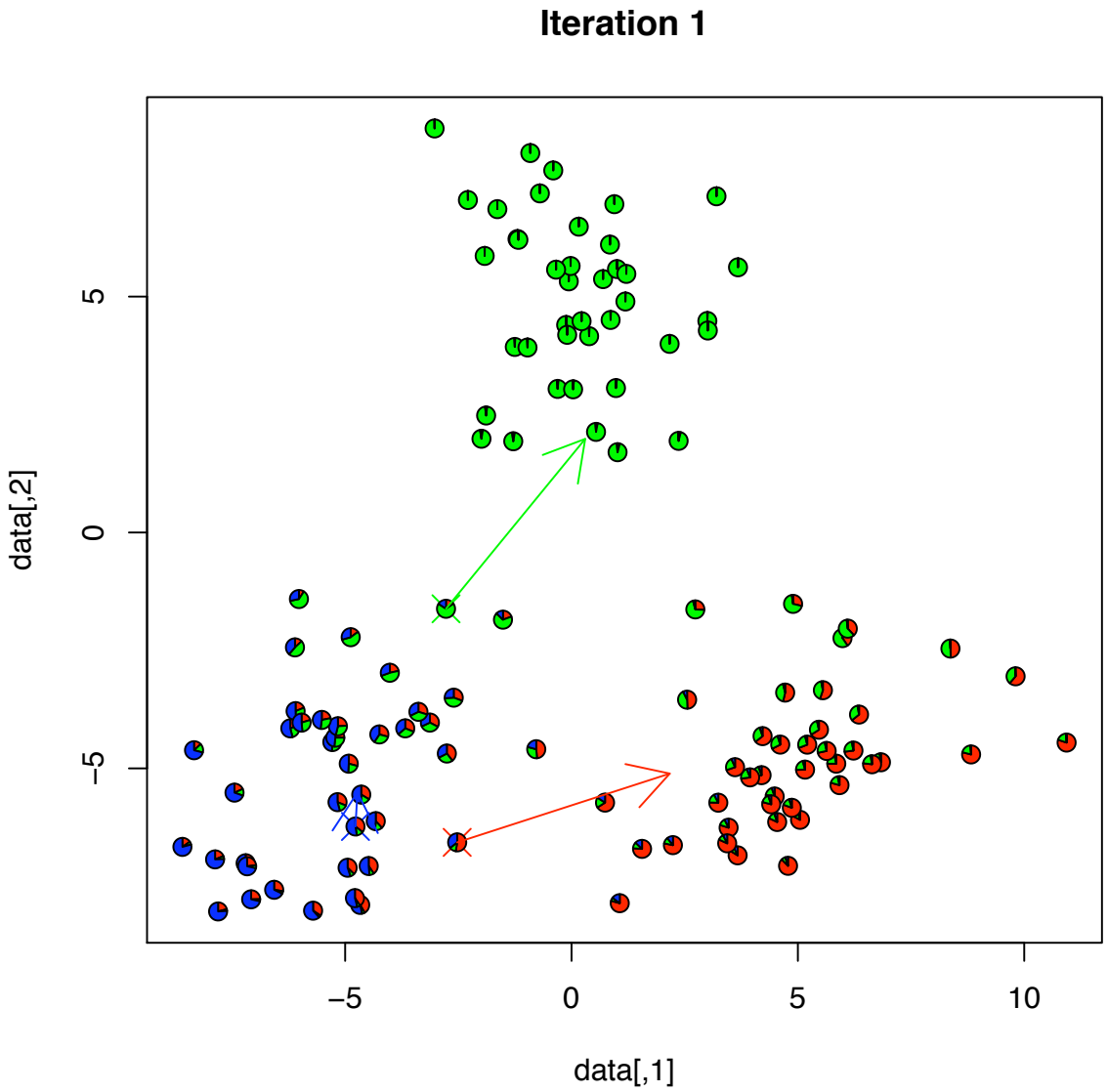
Example: Mixture of 3 Gaussians

An example with 3 clusters.



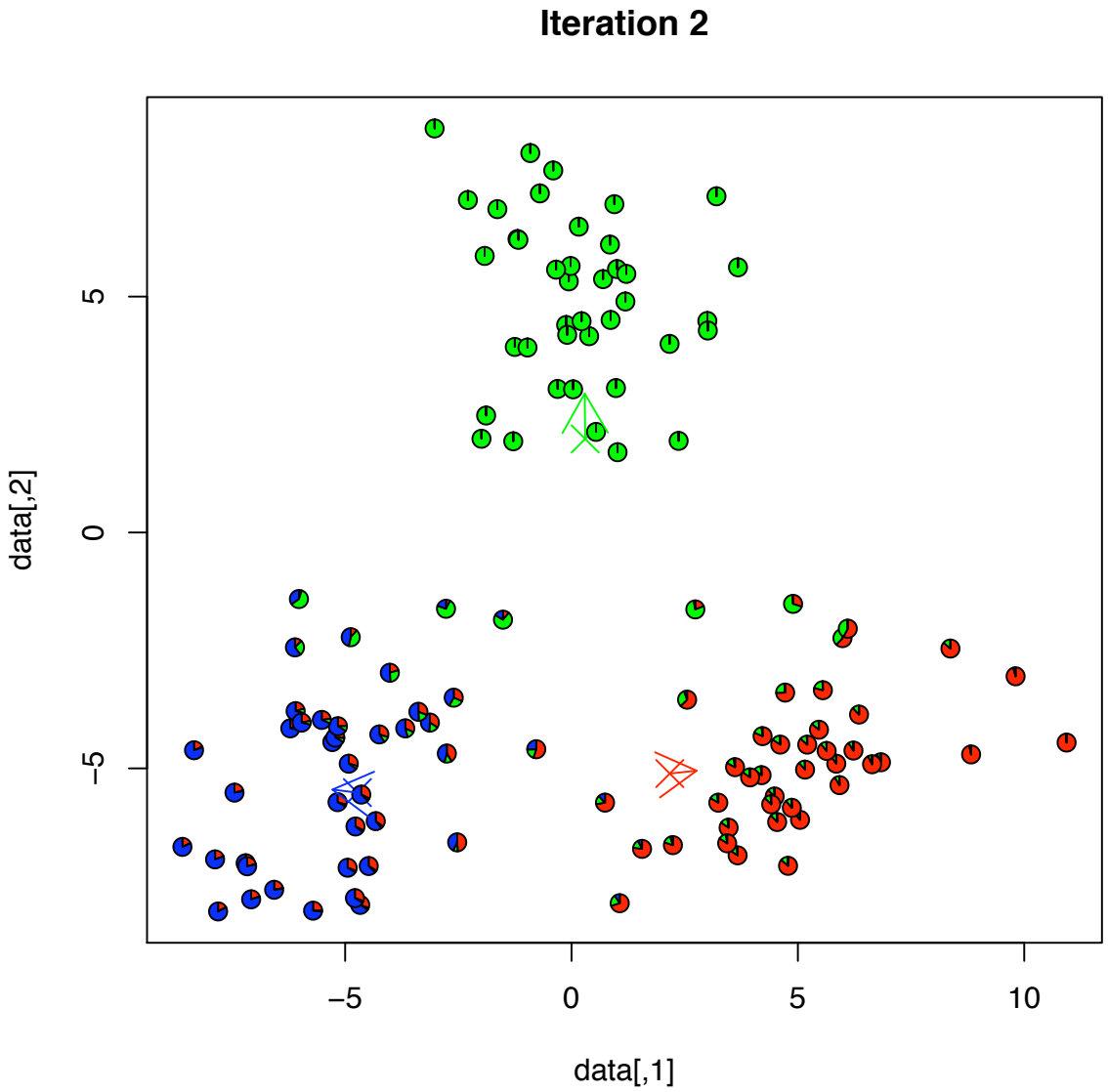
Example: Mixture of 3 Gaussians

After 1st E and M step.



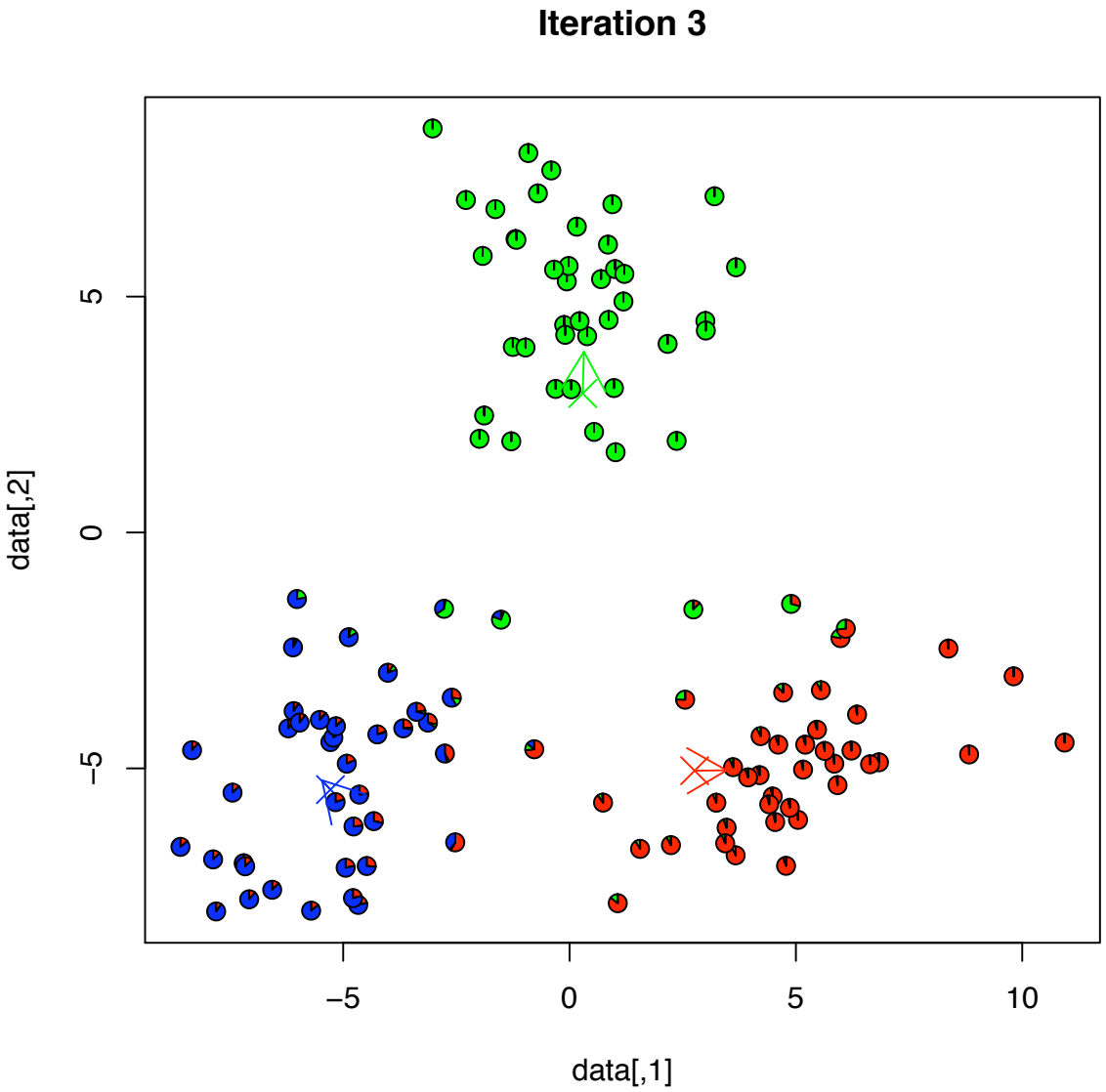
Example: Mixture of 3 Gaussians

After 2nd E and M step.



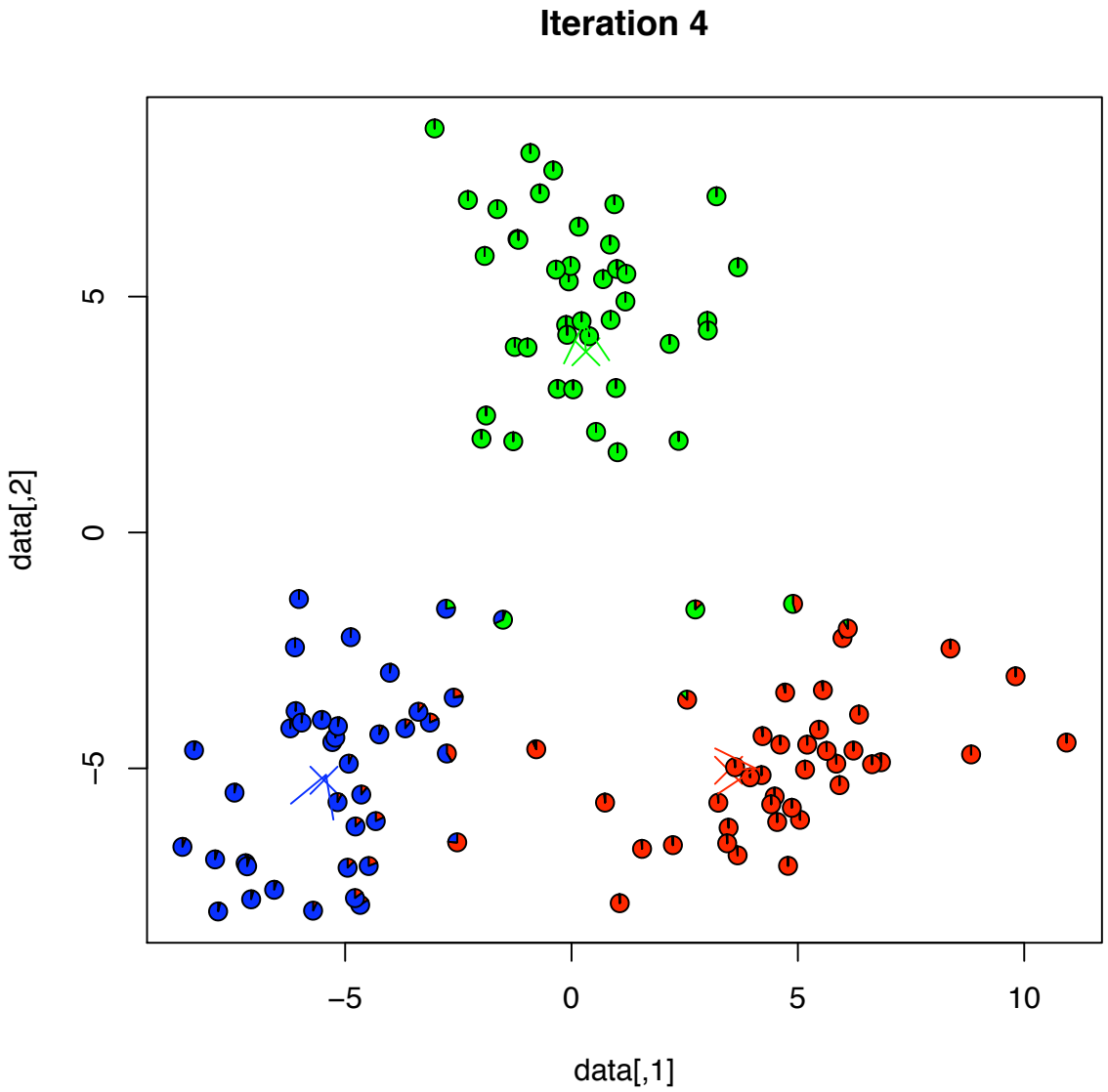
Example: Mixture of 3 Gaussians

After 3rd E and M step.



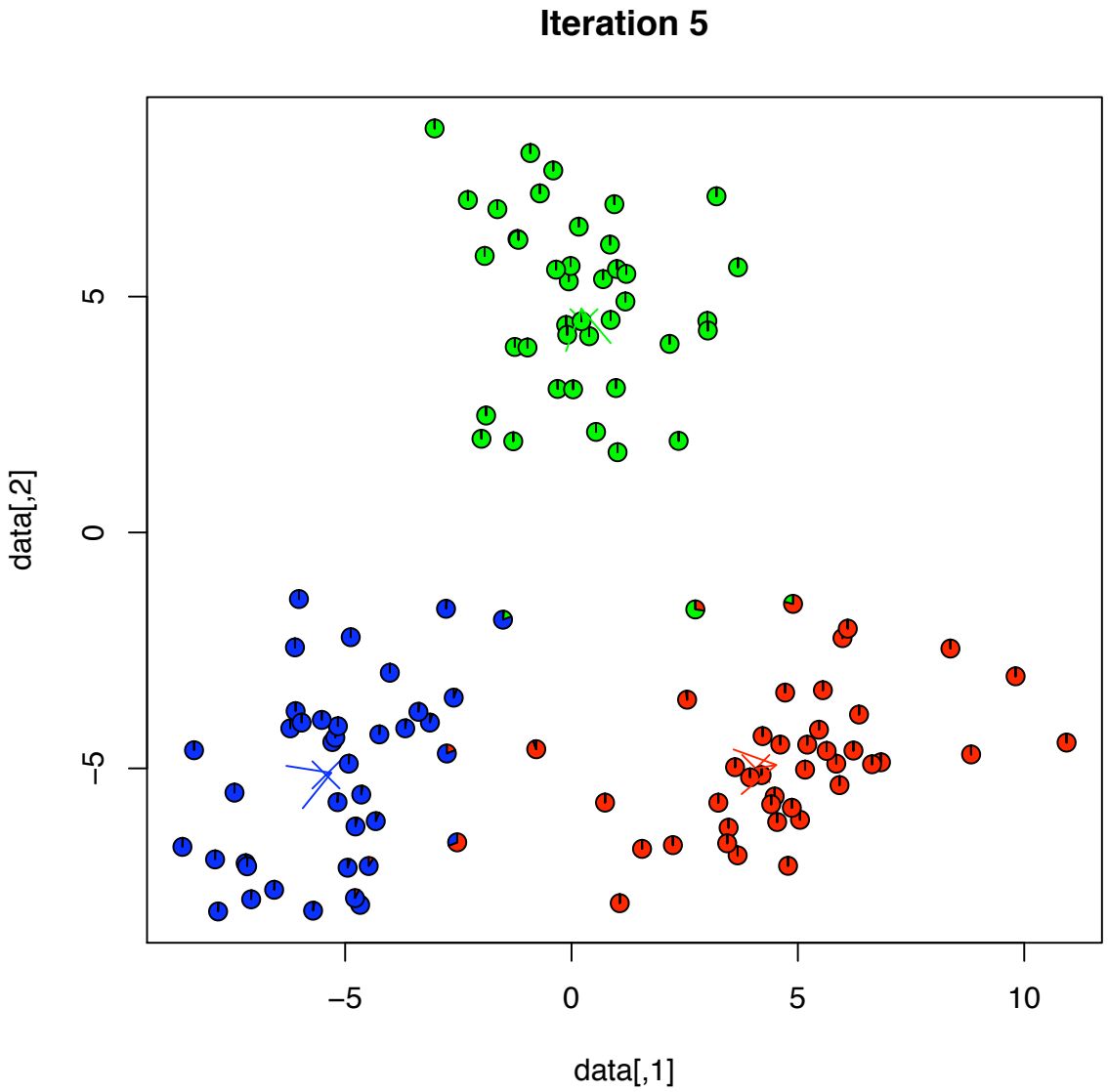
Example: Mixture of 3 Gaussians

After 4th E and M step.



Example: Mixture of 3 Gaussians

After 5th E and M step.



The EM Algorithm

- In a maximum likelihood framework, the objective function is the log likelihood,

$$\ell(\theta) = \sum_{i=1}^{n} \log \sum_{j=1}^{K} \pi_j f(x_i|\phi_j)$$

  Direct maximization is not feasible.
- Consider another objective function $\mathcal{F}(\theta, q)$ such that:

$$\mathcal{F}(\theta, q) \leq \ell(\theta) \text{ for all } \theta, q,$$
$$\max_q \mathcal{F}(\theta, q) = \ell(\theta)$$

  $\mathcal{F}(\theta, q)$ is a lower bound on the log likelihood.
- We can construct an alternating maximization algorithm as follows: For $t = 1, 2 \ldots$ until convergence:

$$q^{(t)} := \operatorname*{argmax}_q \mathcal{F}(\theta^{(t-1)}, q)$$
$$\theta^{(t)} := \operatorname*{argmax}_\theta \mathcal{F}(\theta, q^{(t)})$$

EM Algorithm

- The lower bound we use is called the *variational free energy*.
- q is a probability mass function for some distribution over (Z_i) and

$$\begin{aligned}
\mathcal{F}(\theta, q) =& \mathbb{E}_q[\log p((x_i, z_i)_{i=1}^n) - \log q((z_i)_{i=1}^n)] \\
=& \mathbb{E}_q \left[\left(\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \left(\log \pi_k + \log f(x_i | \phi_k) \right) \right) - \log q(\mathbf{z}) \right] \\
=& \sum_{\mathbf{z}} q(\mathbf{z}) \left[\left(\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \left(\log \pi_k + \log f(x_i | \phi_k) \right) \right) - \log q(\mathbf{z}) \right]
\end{aligned}$$

Using $\mathbf{z} := (z_i)_{i=1}^n$ to shorten notation.

EM Algorithm - Solving for q

- Introducing Lagrange multiplier to enforce $\sum_{\mathbf{z}} q(\mathbf{z}) = 1$, and setting derivatives to 0,

$$\begin{aligned}
\nabla_{q(\mathbf{z})} \mathcal{F}(\theta, q) &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \left(\log \pi_k + \log f(x_i|\phi_k)\right) - \log q(\mathbf{z}) - 1 - \lambda \\
&= \sum_{i=1}^{n} \left(\log \pi_{z_i} + \log f(x_i|\phi_{z_i})\right) - \log q(\mathbf{z}) - 1 - \lambda = 0 \\
q^*(\mathbf{z}) &= \frac{\prod_{i=1}^{n} \pi_{z_i} f(x_i|\phi_{z_i})}{\sum_{\mathbf{z}'} \prod_{i=1}^{n} \pi_{z_i'} f(x_i|\phi_{z_i'})} = \prod_{i=1}^{n} \frac{\pi_{z_i} f(x_i|\phi_{z_i})}{\sum_k \pi_k f(x_i|\phi_k)} = \prod_{i=1}^{n} p(z_i|x_i, \theta)
\end{aligned}$$

- Optimal q^* is simply the posterior distribution.
- Plugging in optimal q^* into the variational free energy,

$$\mathcal{F}(\theta, q^*) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k f(x_i|\phi_k) = \ell(\theta)$$

EM Algorithm - Solving for θ

- Setting derivative with respect to ϕ_k to 0,

$$\nabla_{\phi_k}\mathcal{F}(\theta,q) = \sum_{\mathbf{z}} q(\mathbf{z}) \sum_{i=1}^{n} \mathbb{1}(z_i = k)\nabla_{\phi_k} \log f(x_i|\phi_k)$$

$$= \sum_{i=1}^{n} q(z_i = k)\nabla_{\phi_k} \log f(x_i|\phi_k) = 0$$

- This equation can be solved quite easily. E.g., for mixture of Gaussians,

$$\phi_k^* = \frac{\sum_{i=1}^{n} q(z_i = k)x_i}{\sum_{i=1}^{n} q(z_i = k)}$$

- If it cannot be solved exactly, we can use *gradient ascent* algorithm:

$$\phi_k^* = \phi_k + \alpha \sum_{i=1}^{n} q(z_i = k)\nabla_{\phi_k} \log f(x_i|\phi_k)$$

- This leads to *generalized EM algorithm*. Further extension using *stochastic optimization* method leads to *stochastic EM algorithm*.
- Similar derivation for optimal π_k as before.

EM Algorithm

- Start with some initial parameters $(\pi_k^{(0)}, \phi_l^{(0)})_{k=1}^K$.
- Iterate for $t = 1, 2, \ldots$:
  - *Expectation Step*:

$$q^{(t)}(z_i = k) := \frac{\pi_k^{(t-1)} f(x_i|\phi_k^{(t-1)})}{\sum_{j=1}^K \pi_j^{(t-1)} f(x_i|\phi_j^{(t-1)})} = \mathbb{E}_{p(z_i|x_i,\theta^{(t-1)})}[\mathbb{1}(z_i = k)]$$

  - *Maximization Step*:

$$\pi_k^{(t)} = \frac{\sum_{i=1}^n q^{(t)}(z_i = k)}{n} \qquad \phi_k^{(t)} = \frac{\sum_{i=1}^n q^{(t)}(z_i = k) x_i}{\sum_{i=1}^n q^{(t)}(z_i = k)}$$

- Each step increases the log likelihood:

$$\ell(\theta^{(t-1)}) = \mathcal{F}(\theta^{(t-1)}, q^{(t)}) \leq \mathcal{F}(\theta^{(t)}, q^{(t)}) \leq \mathcal{F}(\theta^{(t)}, q^{(t+1)}) = \ell(\theta^{(t)}).$$

- Additional assumption, that $\nabla_\theta^2 \mathcal{F}(\theta^{(t)}, q^{(t)})$ are negative definite with eigenvalues $< -\epsilon < 0$, implies that $\theta^{(t)} \to \theta^*$ where θ^* is a local MLE.

Notes on Probabilistic Approach and EM Algorithm

Some good things:

- Guaranteed convergence to locally optimal parameters.
- Formal reasoning of uncertainties, using both Bayes Theorem and maximum likelihood theory.
- Rich language of probability theory to express a wide range of generative models, and straightforward derivation of algorithms for ML estimation.

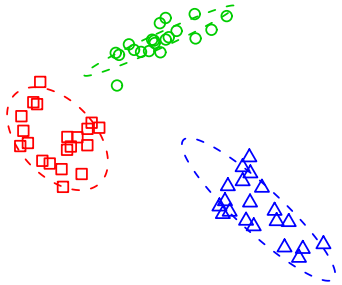
Some bad things:

- Can get stuck in local minima so multiple starts are recommended.
- Slower and more expensive than K-means.
- Choice of K still problematic, but rich array of methods for model selection comes to rescue.

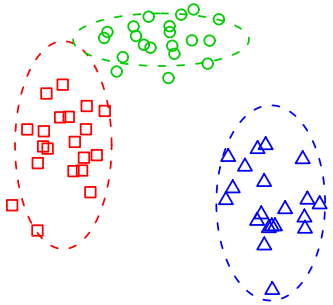
Flexible Gaussian Mixture Models

- We can allow each cluster to have its own mean and covariance structure allows greater flexibility in the model.

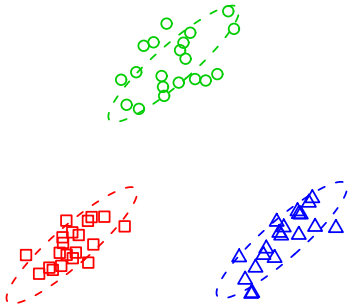
Different covariances



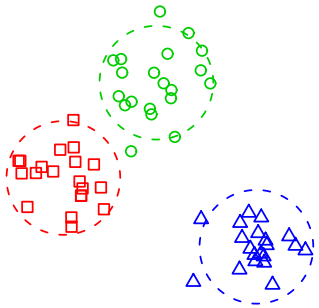
Different, but diagonal covariances



Identical covariances



Identical and spherical covariances



Probabilistic PCA

- A probabilistic model related to PCA has the following generative model: for $i = 1, 2, \ldots, n$:
  - Let $k < n, p$ be given.
  - Let Y_i be a k-dimensional normally distributed random variable with 0 mean and identity covariance:
    $$Y_i \sim \mathcal{N}(0, I_k)$$
  - We model the distribution of the ith data point given Y_i as a p-dimensional normal:
    $$X_i \sim \mathcal{N}(\mu + LY_i, \sigma^2 I)$$
    where the parameters are a vector $\mu \in \mathbb{R}^p$, a matrix $L \in \mathbb{R}^{p \times k}$ and $\sigma^2 > 0$.
- EM algorithm can be used for ML estimation, but PCA can more directly give a MLE (note this is not unique).
- Let $\lambda_1 \geq \cdots \geq \lambda_p$ be the eigenvalues of the sample covariance and let $V \in \mathbb{R}^{p \times k}$ have columns given by the eigenvectors of the top k eigenvalues. Let $R \in \mathbb{R}^{k \times k}$ be orthogonal. Then a MLE is:
  $$\mu^{\text{MLE}} = \bar{x} \qquad (\sigma^2)^{\text{MLE}} = \frac{1}{p-k} \sum_{j=k+1}^{p} \lambda_j$$
  $$L^{\text{MLE}} = V \operatorname{diag}((\lambda_1 - (\sigma^2)^{\text{MLE}})^{\frac{1}{2}}, \ldots, (\lambda_k - (\sigma^2)^{\text{MLE}})^{\frac{1}{2}}) R$$

Mixture of Probabilistic PCAs

- We have learnt two types of unsupervised learning techniques:
  - Dimensionality reduction, e.g. PCA, MDS, Isomap.
  - Clustering, e.g. K-means, linkage and mixture models.
- Probabilistic models allow us to construct more complex models from simpler pieces.
- Mixture of probabilistic PCAs allows both clustering and dimensionality reduction at the same time.

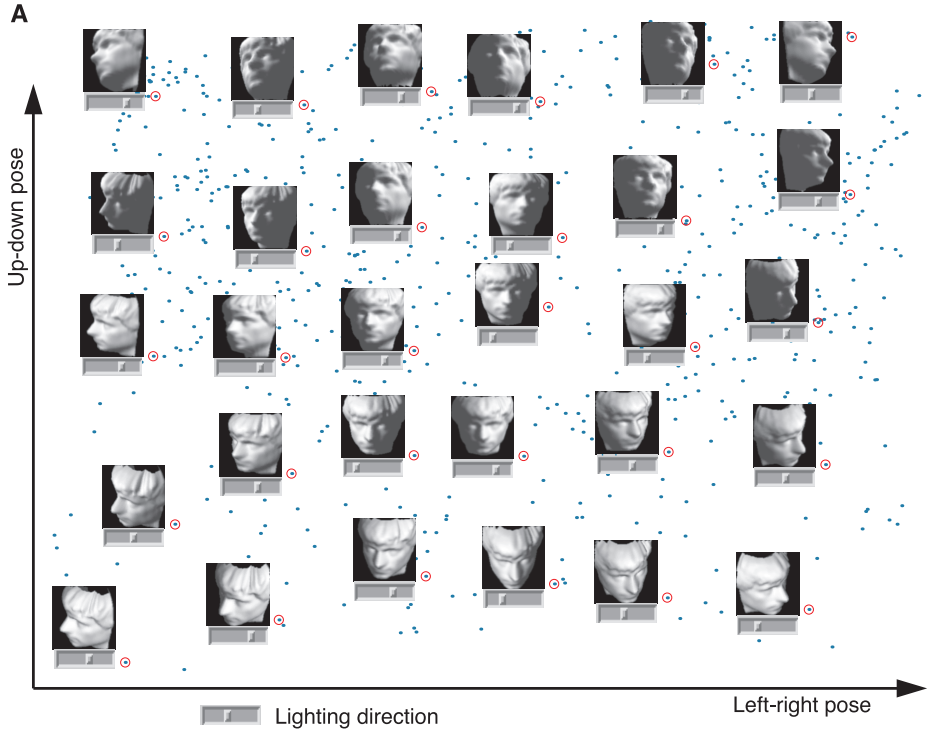
$$
\begin{aligned}
Z_i &\sim \text{Discrete}(\pi_1, \ldots, \pi_K) \\
Y_i &\sim \mathcal{N}(0, I_d) \\
X_i | Z_i = k, Y_i = y_i &\sim \mathcal{N}(\mu_k + L y_i, \sigma^2 I_p)
\end{aligned}
$$

- Allows flexible modelling of covariance structure without using too many parameters.

Ghahramani and Hinton 1996

Mixture of Probabilistic PCAs

- PCA can reconstruct x given low dimensional embedding z, but is linear.
- Isomap is non-linear, but cannot reconstruct x given any z.



- We can learn a probabilistic mapping between the k-dimensional Isomap embedding space and the p-dimensional data space.
- Demo: [Using LLE instead of Isomap, and Mixture of factor analysers instead of Mixture of PPCAs.]

Teh and Roweis 2002

Further Readings—Unsupervised Learning

- Hastie et al, Chapter 14.
- James et al, Chapter 10.
- Venables and Ripley, Chapter 11.
- Tukey, John W. (1980). We need both exploratory and confirmatory. The American Statistician 34 (1): 23-25.