Aprendizagem não supervisonada

Introdução

- No aprendizado **supervisionado**, todas os exemplos de treinamento eram **rotulados**.

$$\underbrace{0.51\ \ 0.14\ \ 0.12\ \ 0.04\ \ 0.65\ \ 0.01\ \ 0.08}_{\text{Vetor de Atributos}}\ \ \underbrace{\mathbf{2}}_{\textbf{Classe}}$$

- Estes exemplos são ditos “supervisionados”, pois, contém tanto a **entrada** (atributos), quanto a **saída** (classe).

Introdução

- Porém, muitas vezes temos que lidar com exemplos "**não–supervisionados**", isto é, exemplos **não rotulados**.

- **Por que?**
  - Coletar e rotular um grande conjunto de exemplos pode custar muito tempo, esforço, dinheiro...

Introdução

- Entretanto, podemos utilizar grandes quantidades de dados **não rotulados** para encontrar padrões existentes nestes dados. E somente depois supervisionar a rotulação dos agrupamentos encontrados.

- Esta abordagem é bastante utilizada em aplicações de **mineração de dados** (datamining), onde o conteúdo de grandes bases de dados não é conhecido antecipadamente.

Introdução

- O principal interesse do aprendizado não-supervisionado é desvendar a organização dos padrões existentes nos dados através de **clusters** (agrupamentos) consistentes.

- Com isso, é possível descobrir **similaridades e diferenças** entre os padrões existentes, assim como derivar conclusões úteis a respeito deles.

Clusterização

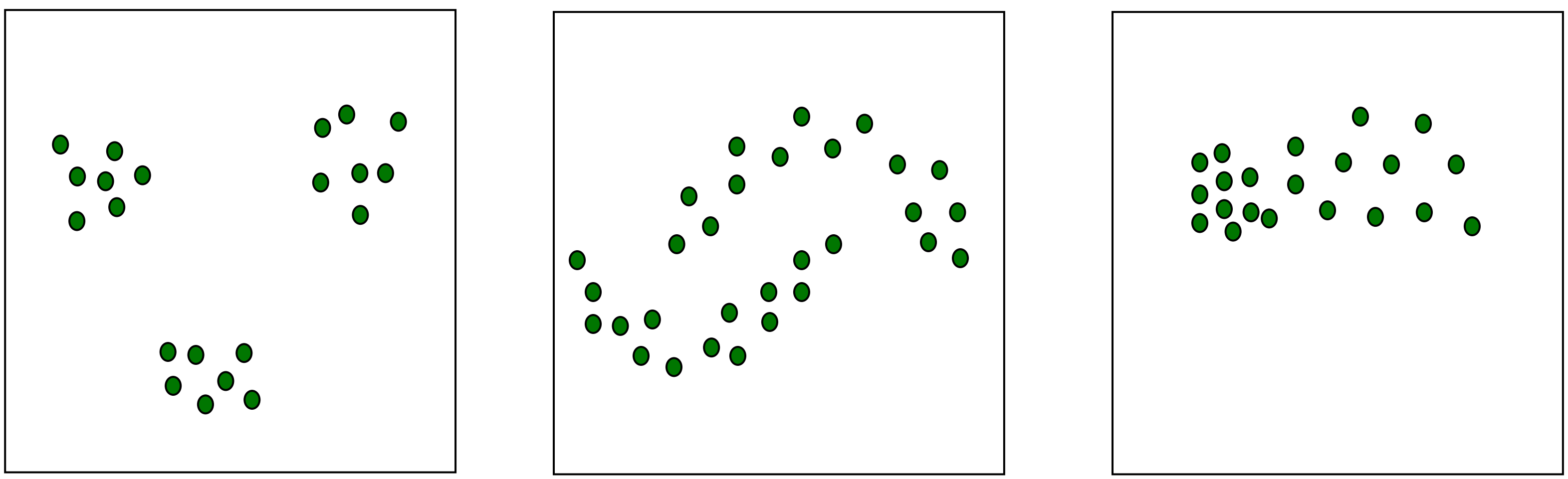
- A **clusterização** é o processo de **agrupar** um conjunto de objetos físicos ou abstratos em classes de objetos **similares**.

- Um cluster é uma coleção de objetos que são similares uns aos outros (de acordo com algum **critério de similaridade** pré-definido) e dissimilares a objetos pertencentes a outros clusters.

Clustering

- Clustering describes data by “groups”
- The meaning of “groups” may vary by data!

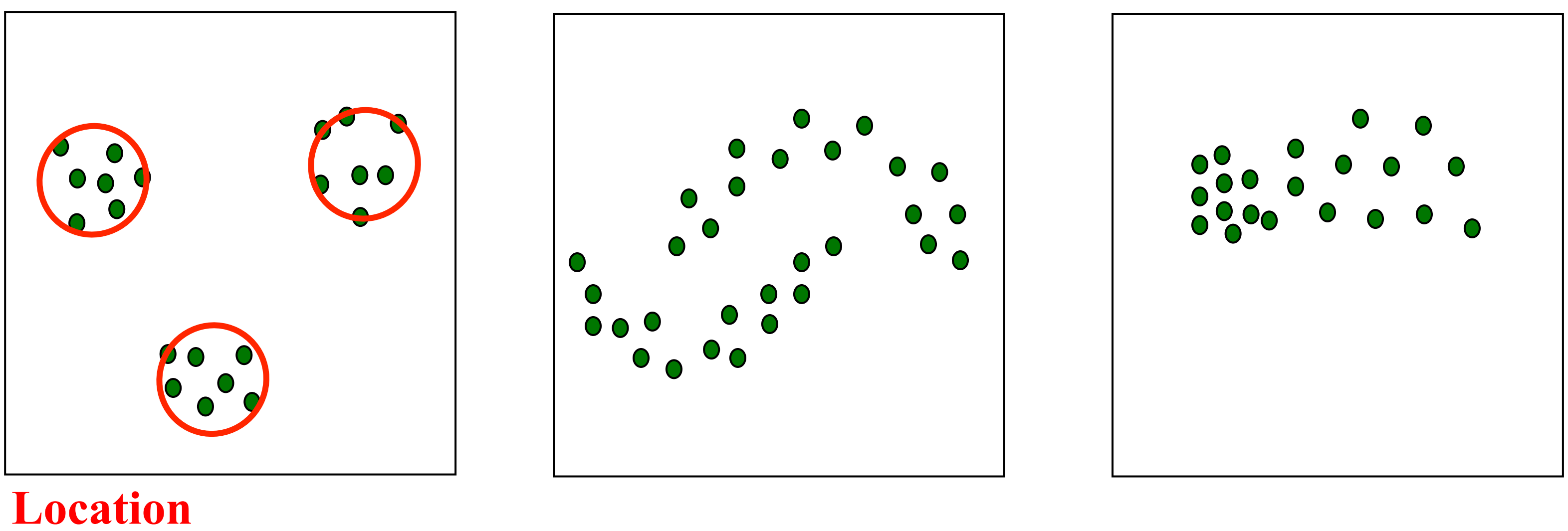
- Examples



Clustering

- Clustering describes data by "groups"
- The meaning of "groups" may vary by data!

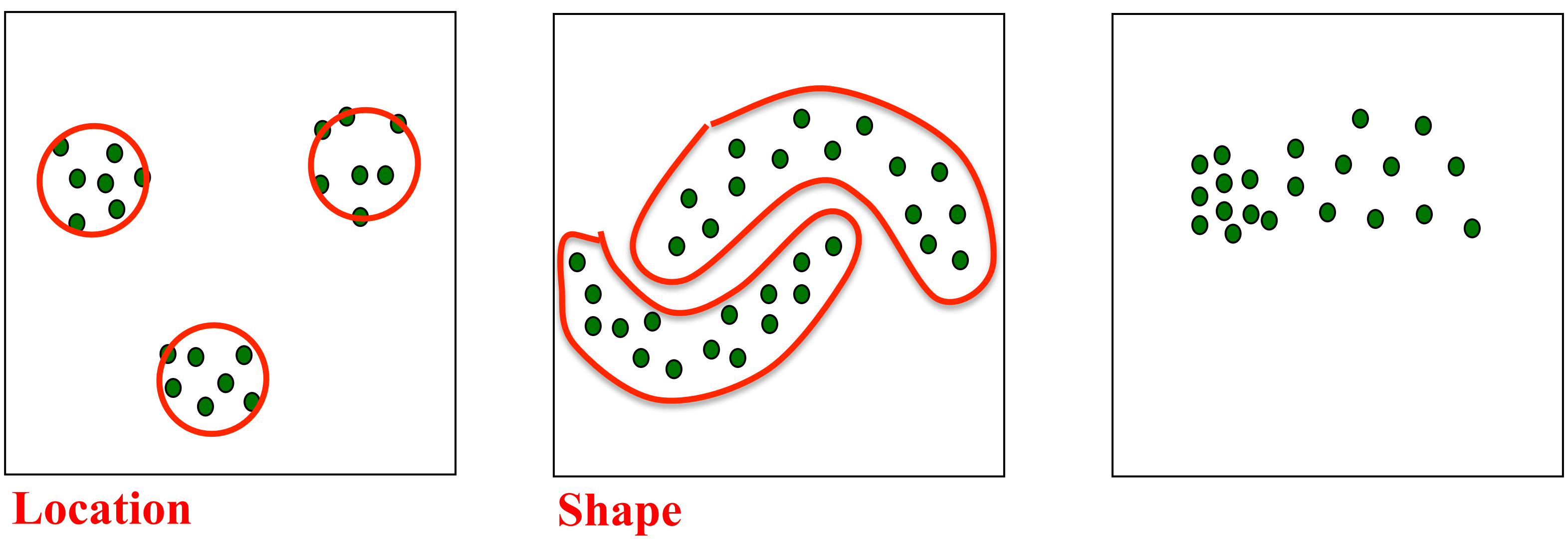
- Examples



Clustering

- Clustering describes data by "groups"
- The meaning of "groups" may vary by data!

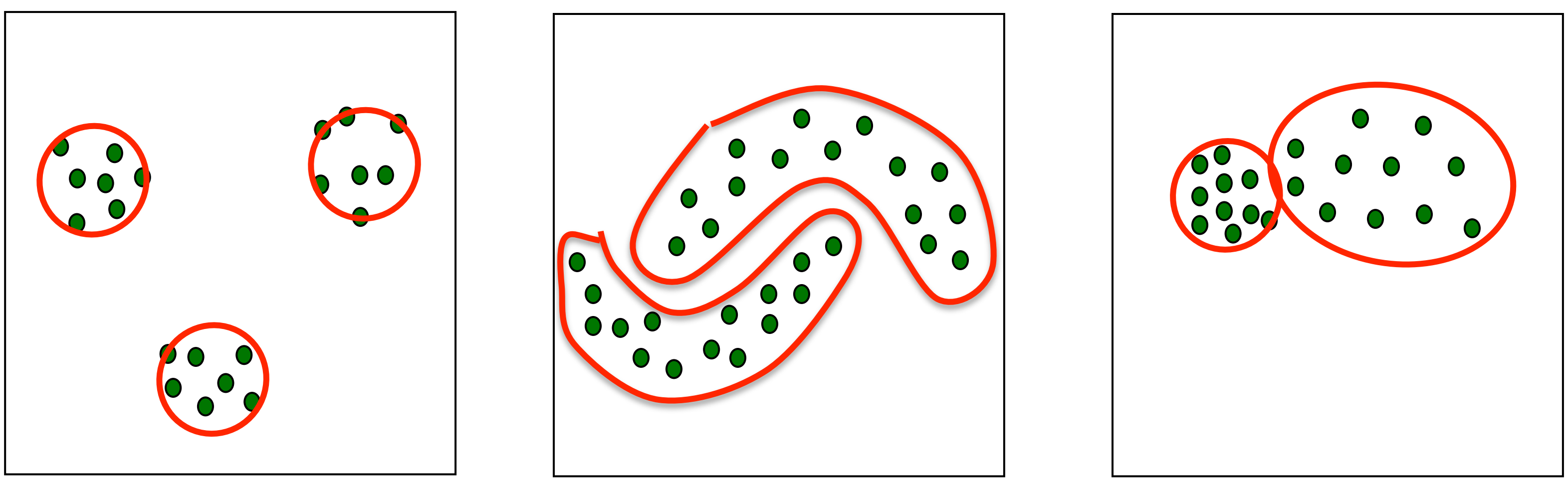
- Examples



Clustering

- Clustering describes data by “groups”
- The meaning of “groups” may vary by data!

- Examples



Location **Shape** **Density**

Critério de Similaridade

- A similaridade é difícil de ser definida...

Algoritmos de Clustering

- Os **algoritmos de clusterização** buscam identificar padrões existentes em conjuntos de dados.

- Os algoritmos de clusterização podem ser divididos em varias categorias:
  - Sequenciais;
  - Hierárquicos;
  - Baseados na otimização de funções custo;
  - Outros: Fuzzy, SOM, LVQ...

Algoritmos Sequenciais

- São algoritmos diretos e rápidos.
- Geralmente, todos os vetores de características são apresentados ao algoritmo uma ou várias vezes.
- O resultado final geralmente depende da ordem de apresentação dos vetores de características.

Algoritmos Sequenciais

- Basic Sequential Algorithmic Scheme (BSAS)
  - Todos os vetores são apresentados uma única vez ao algoritmo.
  - Número de clusters não é conhecido inicialmente.
  - Novos clusters são criados enquanto o algoritmo evolui.

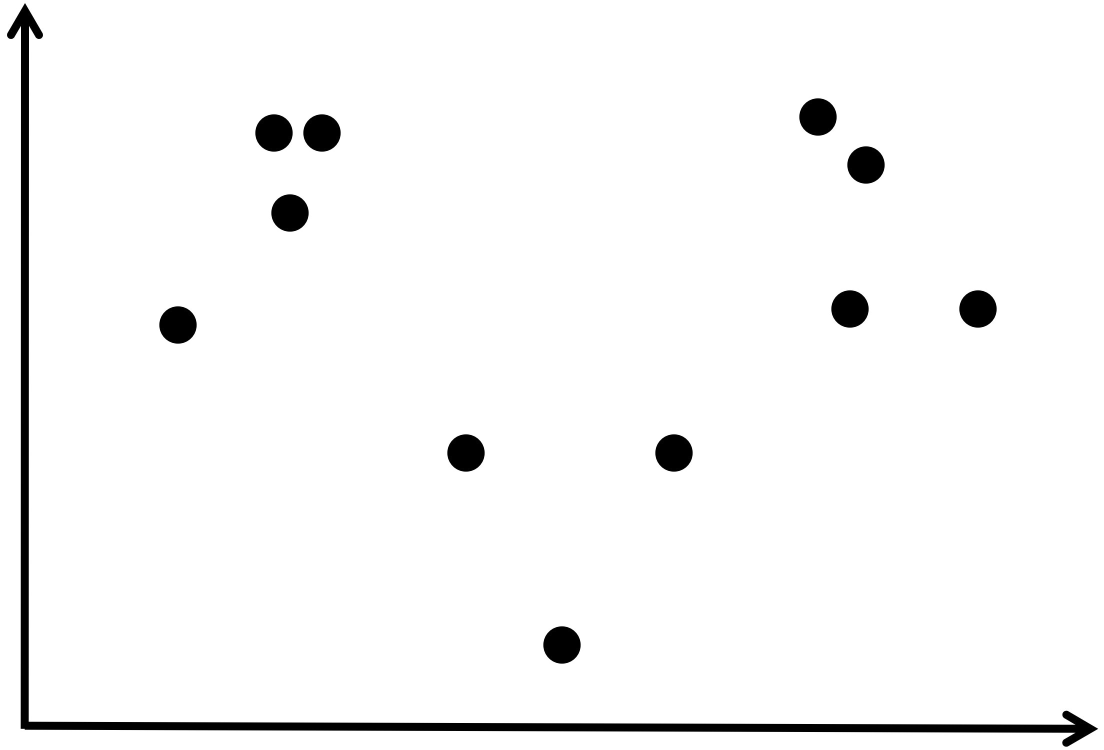
Basic Sequential Algorithmic Scheme (BSAS)

- **Parâmetros do BSAS:**
  - **d(x, C):** métrica de distância entre um vetor de características x e um cluster C.
  - **Θ:** limiar de dissimilaridade.
  - **q:** número máximo de clusters.

- **Ideia Geral do Algoritmo:**
  - Para um dado vetor de características, designá–lo para um cluster existente ou criar um novo cluster (depende da distância entre o vetor e os clusters já formados).

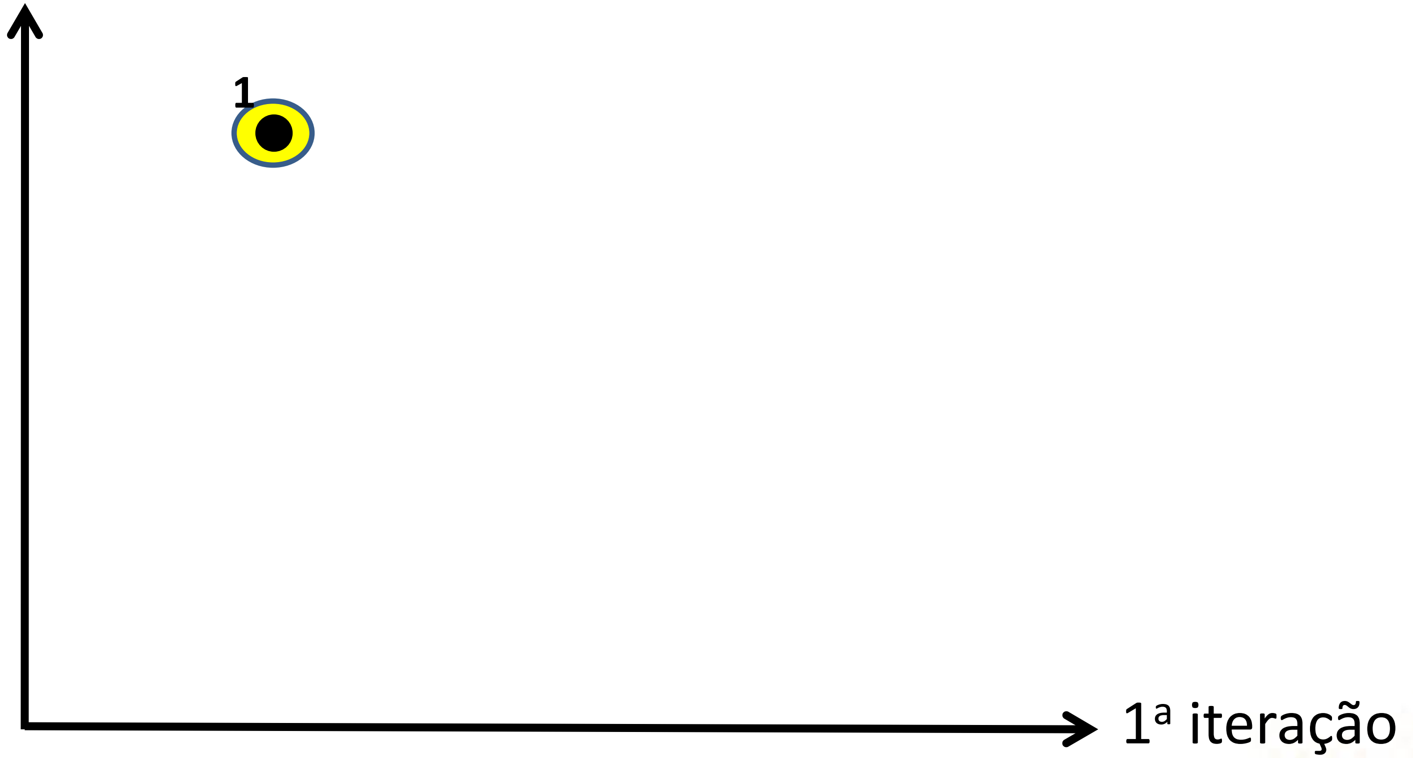
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



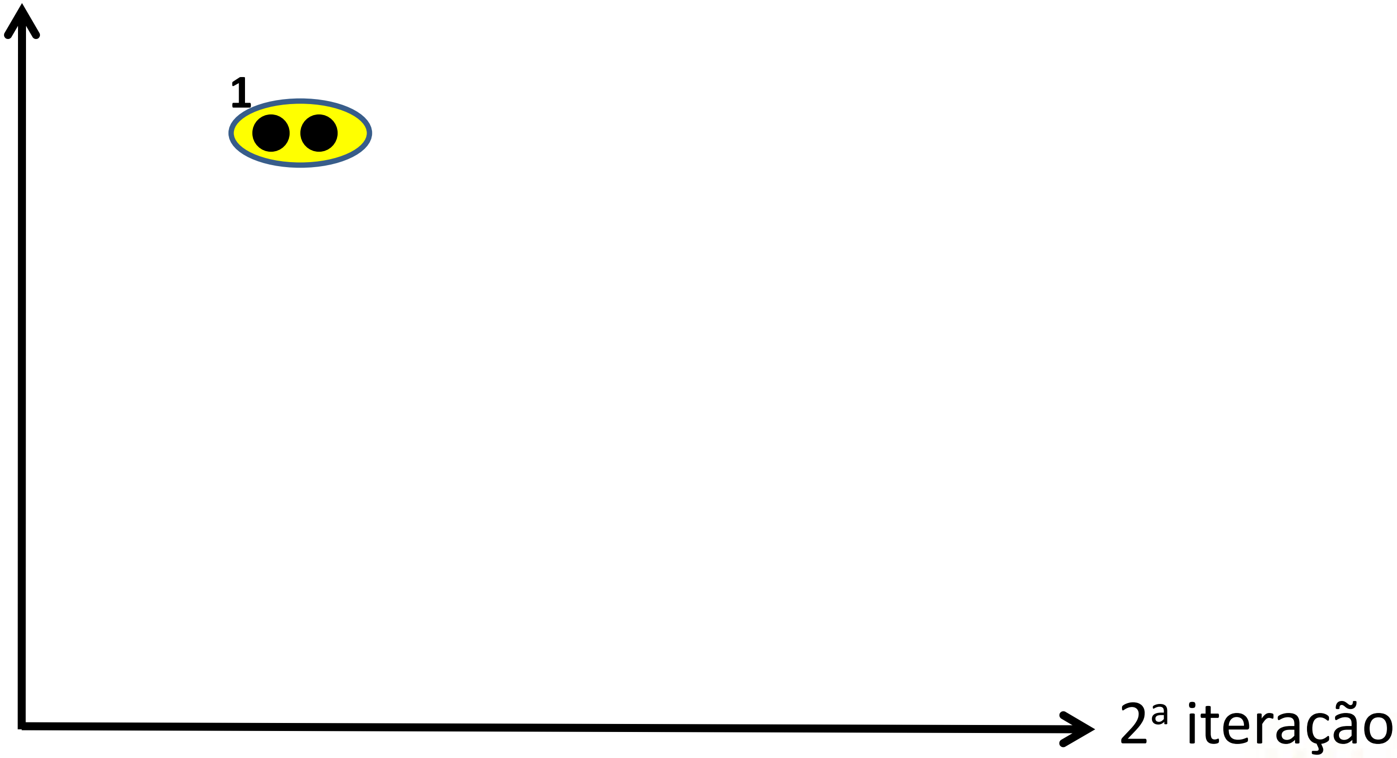
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



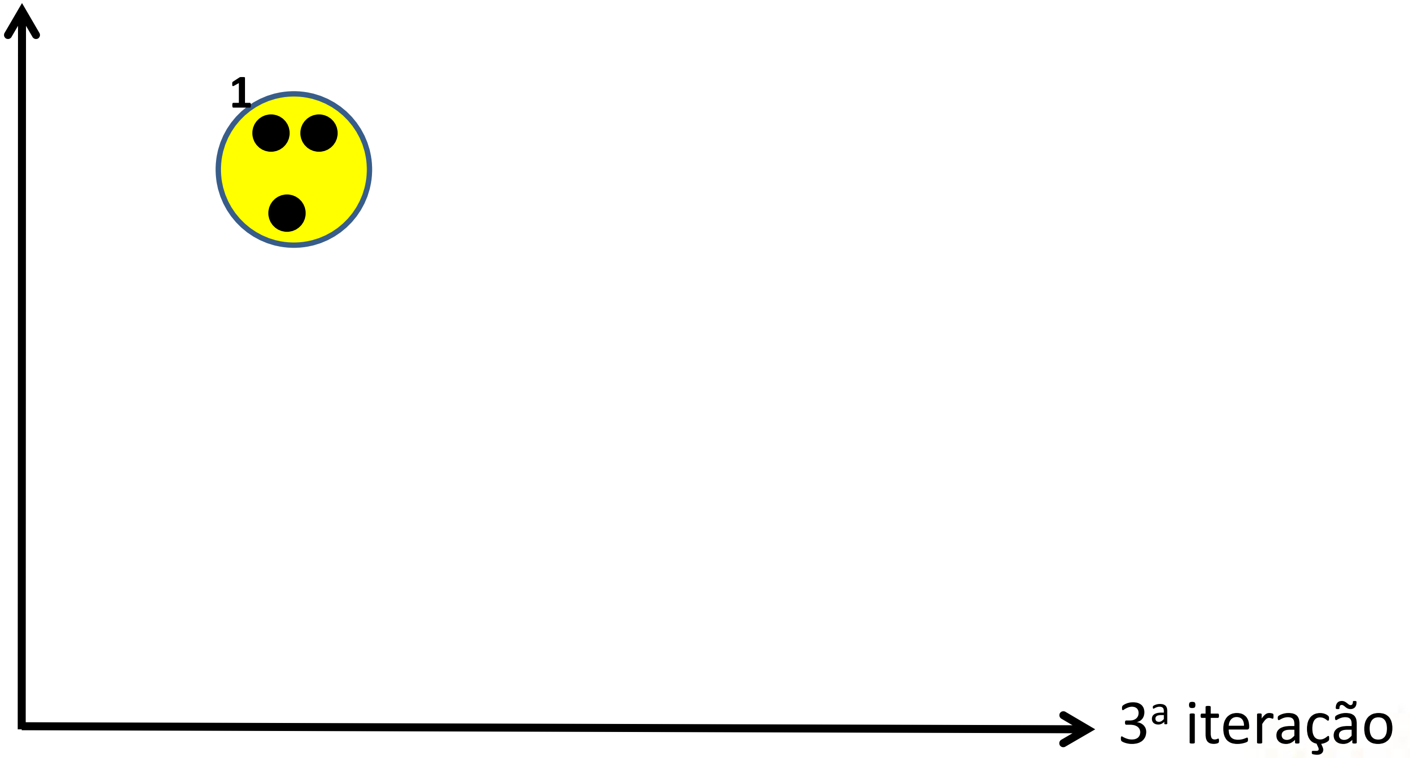
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



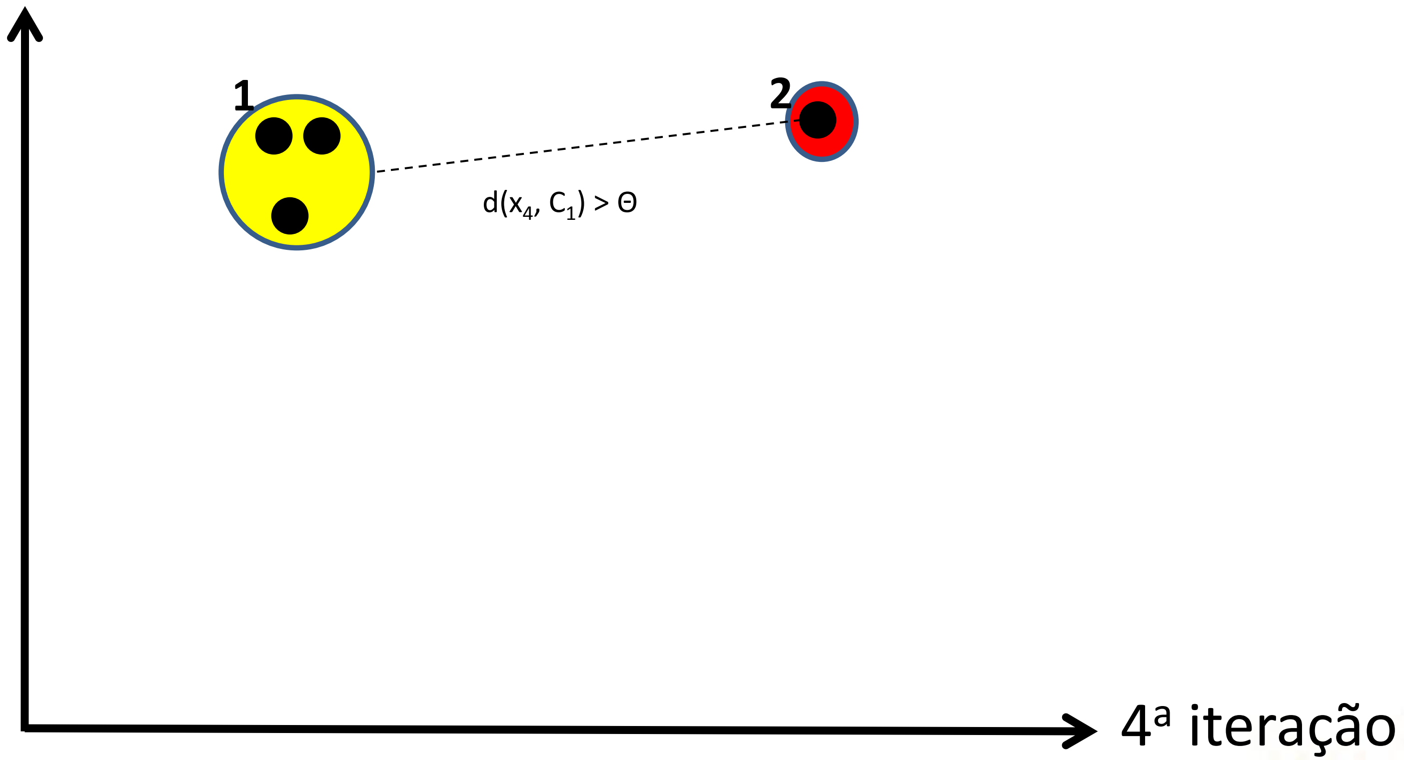
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



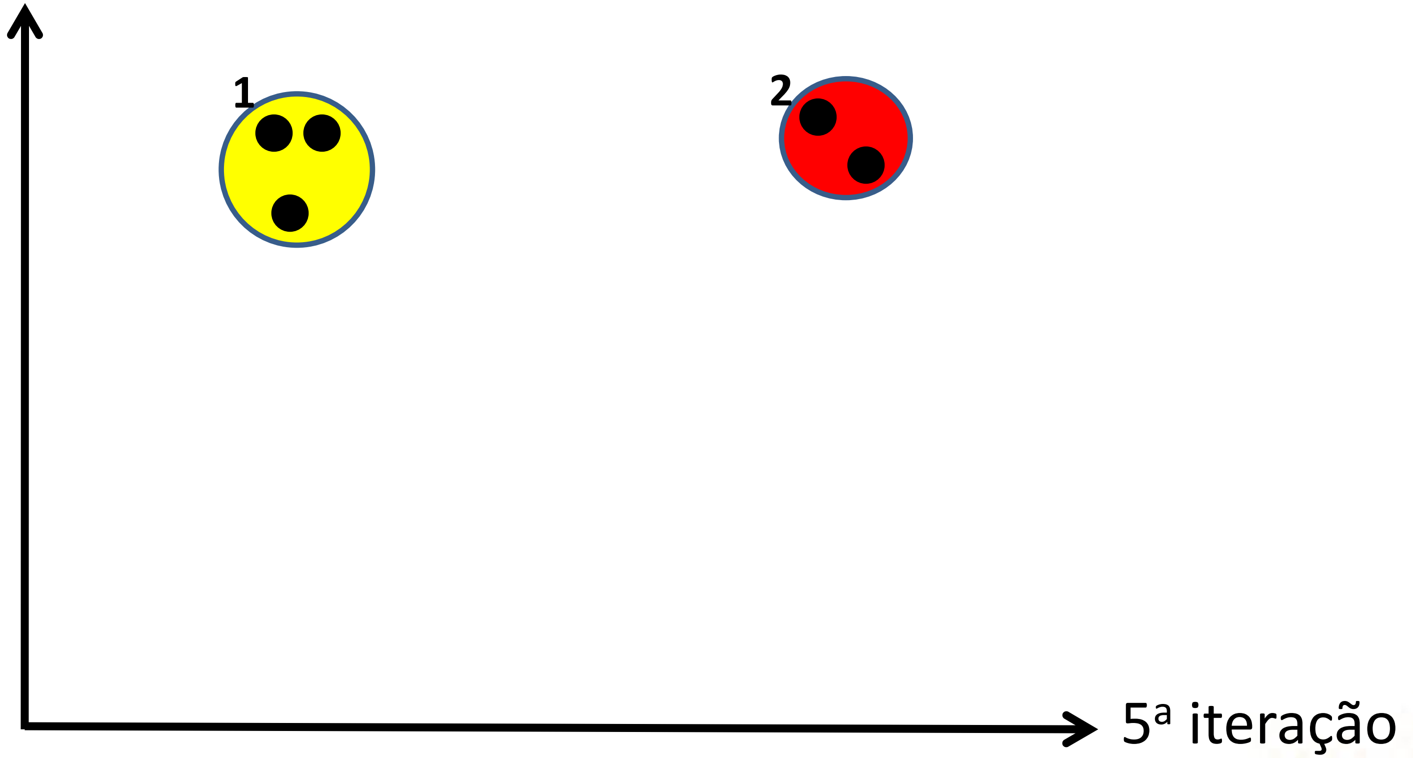
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



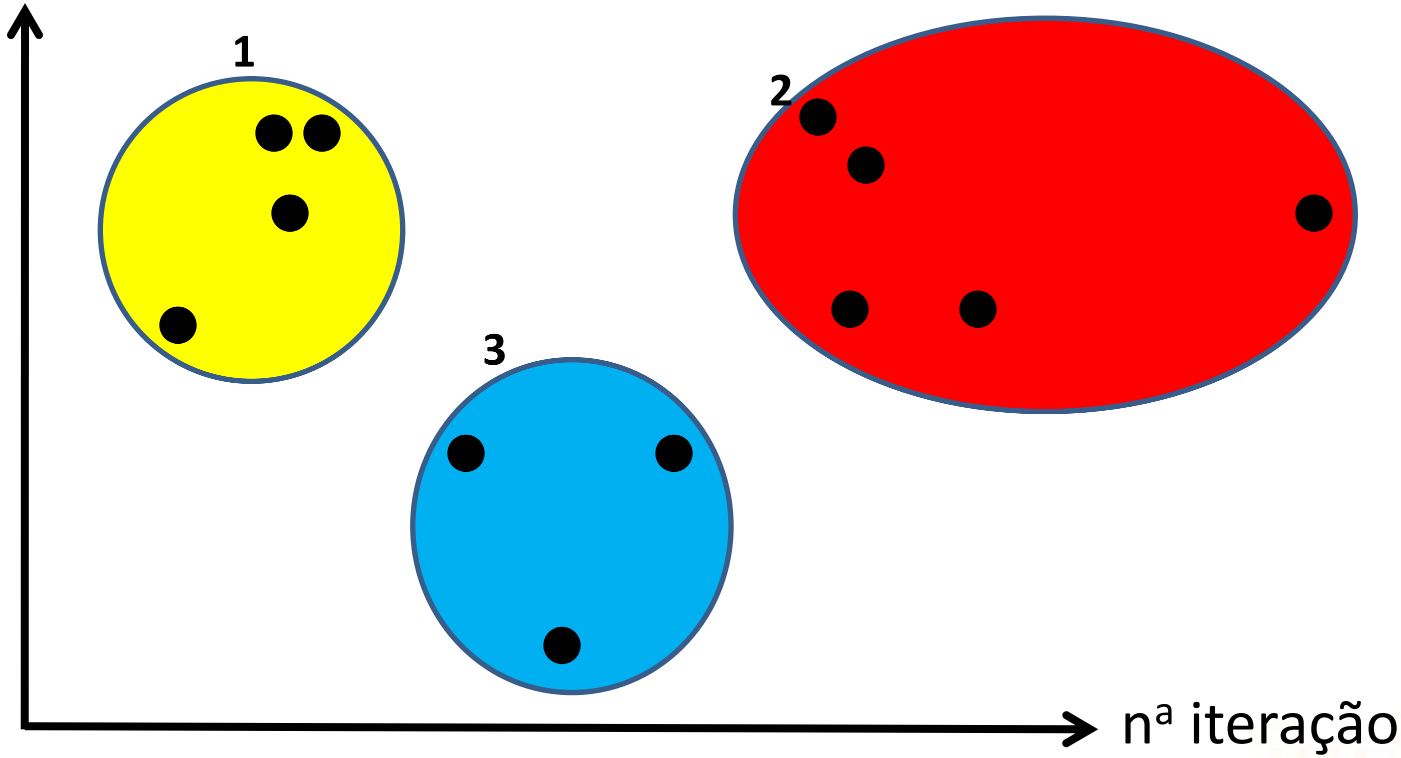
Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



Basic Sequential Algorithmic Scheme (BSAS)

- Exemplo 1:



Clusterização Hierárquica

- Os algoritmos de **clusterização hierárquica** pode ser divididos em 2 subcategorias:

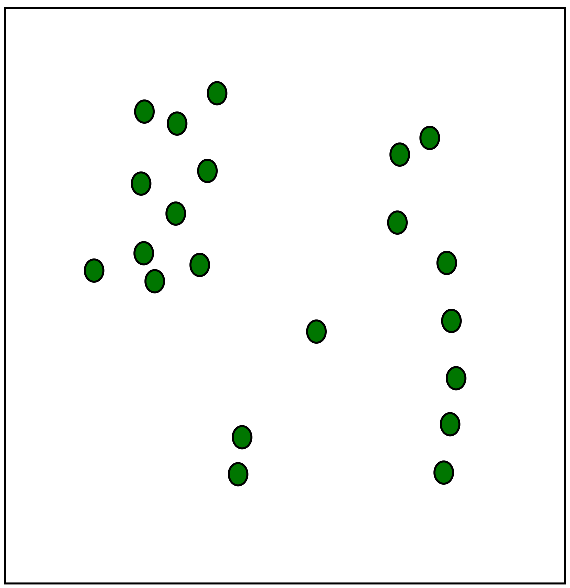
- **Aglomerativos:**
  - Produzem uma sequência de agrupamentos com um número decrescente de clusters a cada passo.
  - Os agrupamentos produzidos em cada passo resultam da fusão de dois clusters em um.

- **Divisivos:**
  - Atuam na direção oposta, isto é, eles produzem uma sequência de agrupamentos com um número crescente de clusters a cada passo.
  - Os agrupamentos produzidos em cada passo resultam da partição de um único cluster em dois.

Hierarchical Agglomerative Clustering

Initially, every datum is a cluster

Data:



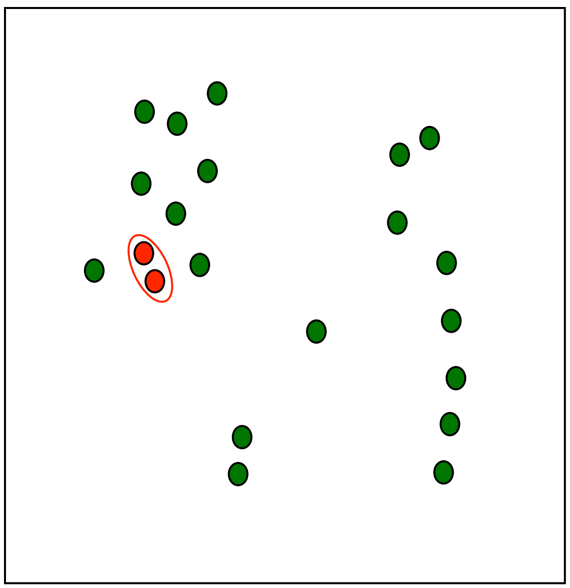
Algorithmic Complexity: O(m^2 log m) +

- A simple clustering algorithm
- Define a distance (or dissimilarity) between clusters (we'll return to this)
- Initialize: every example is a cluster
- Iterate:
  - Compute distances between all clusters (store for efficiency)
  - Merge two closest clusters
- Save both clustering and *sequence* of cluster operations
- "Dendrogram"

Iteration 1

Builds up a sequence of clusters ("hierarchical")

Data:



Dendrogram:

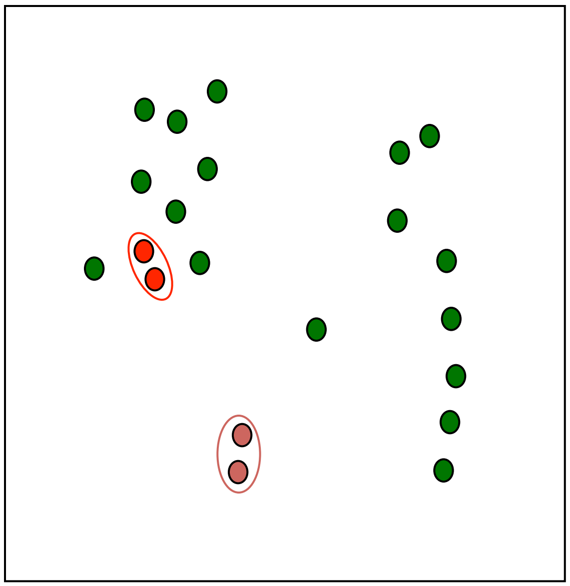
Height of the join indicates dissimilarity

Algorithmic Complexity: $O(m^2 \log m) + O(m \log m) +$

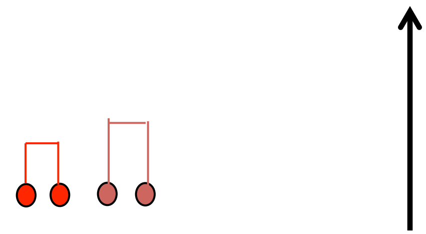
Iteration 2

Builds up a sequence of clusters ("hierarchical")

Data:



Dendrogram:



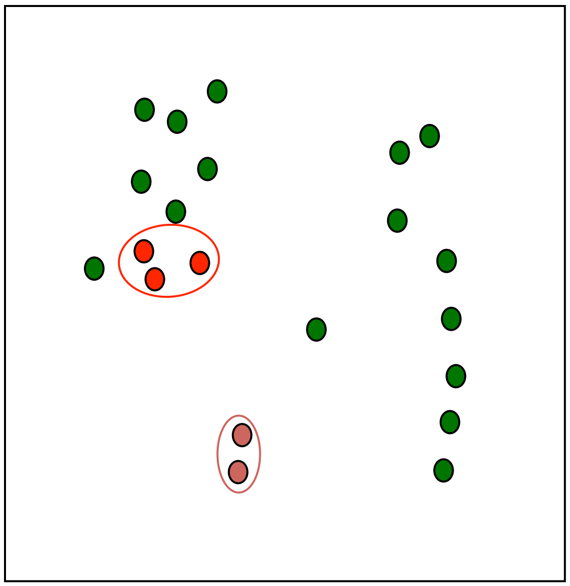
Height of the join
indicates dissimilarity

Algorithmic Complexity: $O(m^2 \log m) + 2*O(m \log m) +$

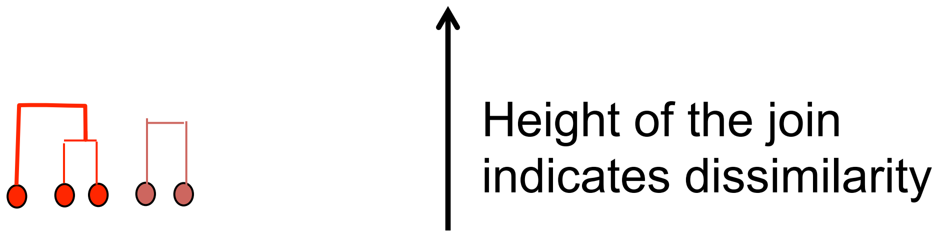
Iteration 3

Builds up a sequence of clusters ("hierarchical")

Data:



Dendrogram:

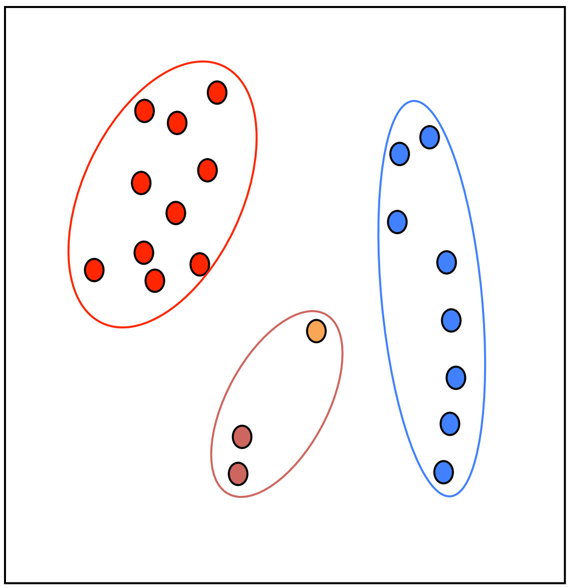


Algorithmic Complexity: $O(m^2 \log m) + 3*O(m \log m) +$

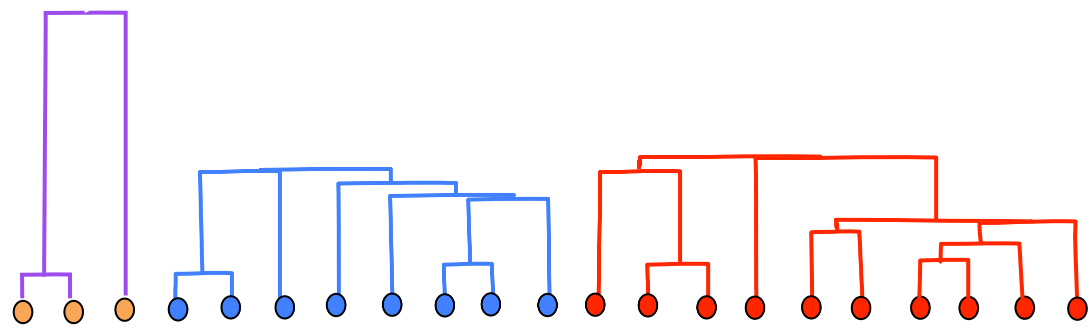
Iteration m-3

Builds up a sequence of clusters ("hierarchical")

Data:



Dendrogram:



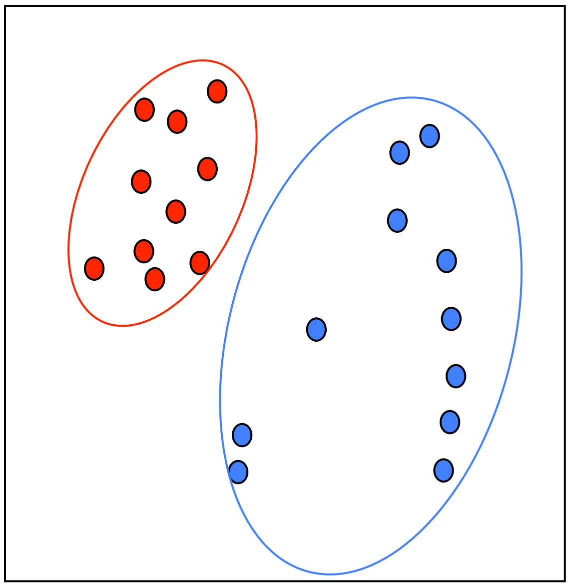
In matlab: "linkage" function (stats toolbox)

Algorithmic Complexity: $O(m^2 \log m) + (m\text{-}3)*O(m \log m) +$

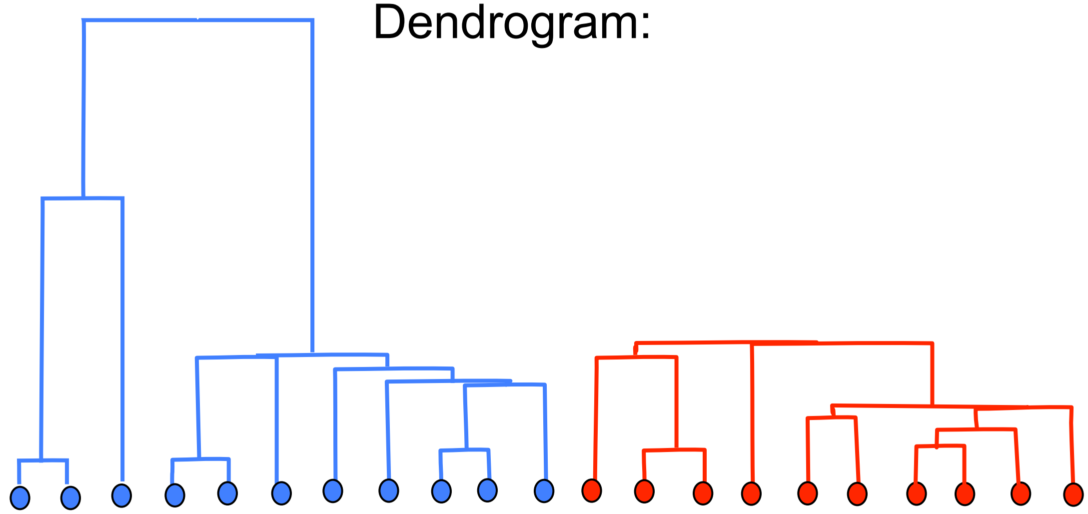
Iteration m-2

Builds up a sequence of clusters ("hierarchical")

Data:



Dendrogram:

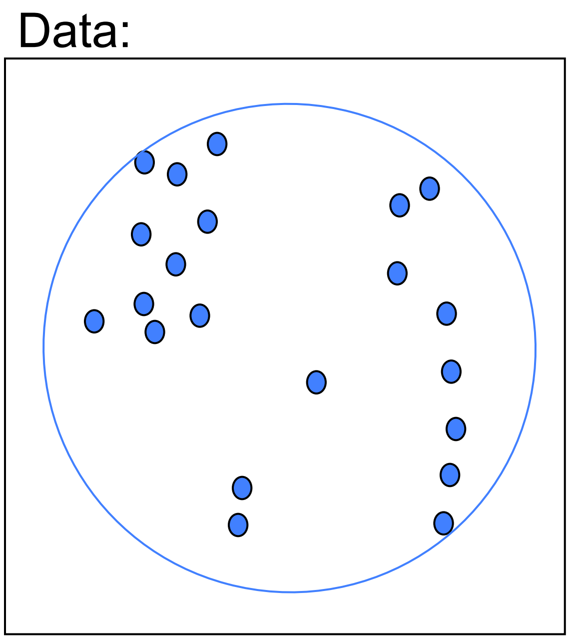


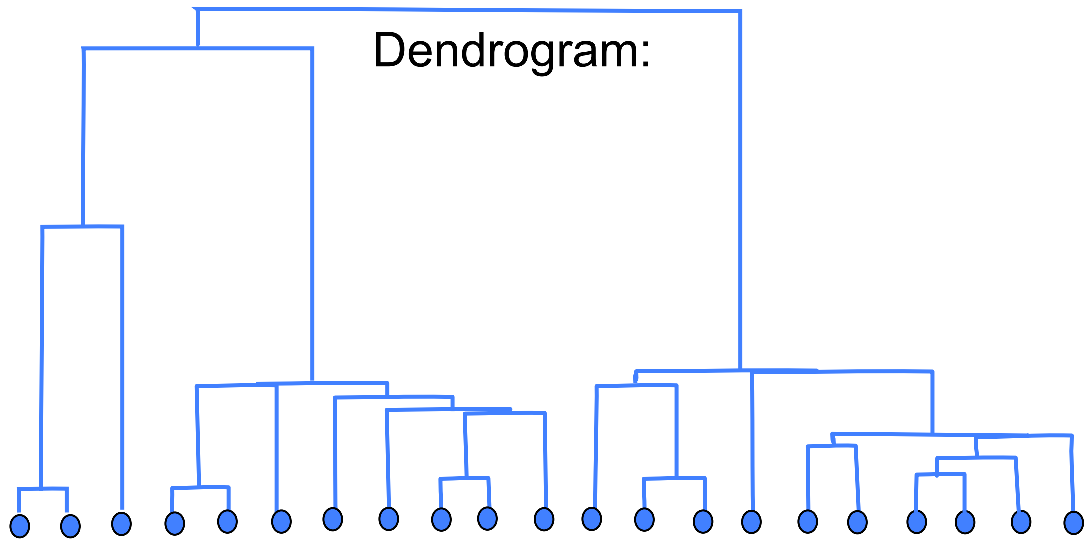
In matlab: "linkage" function (stats toolbox)

Algorithmic Complexity: $O(m^2 \log m) + (m-2)*O(m \log m) +$

Iteration m-1

Builds up a sequence of clusters ("hierarchical")



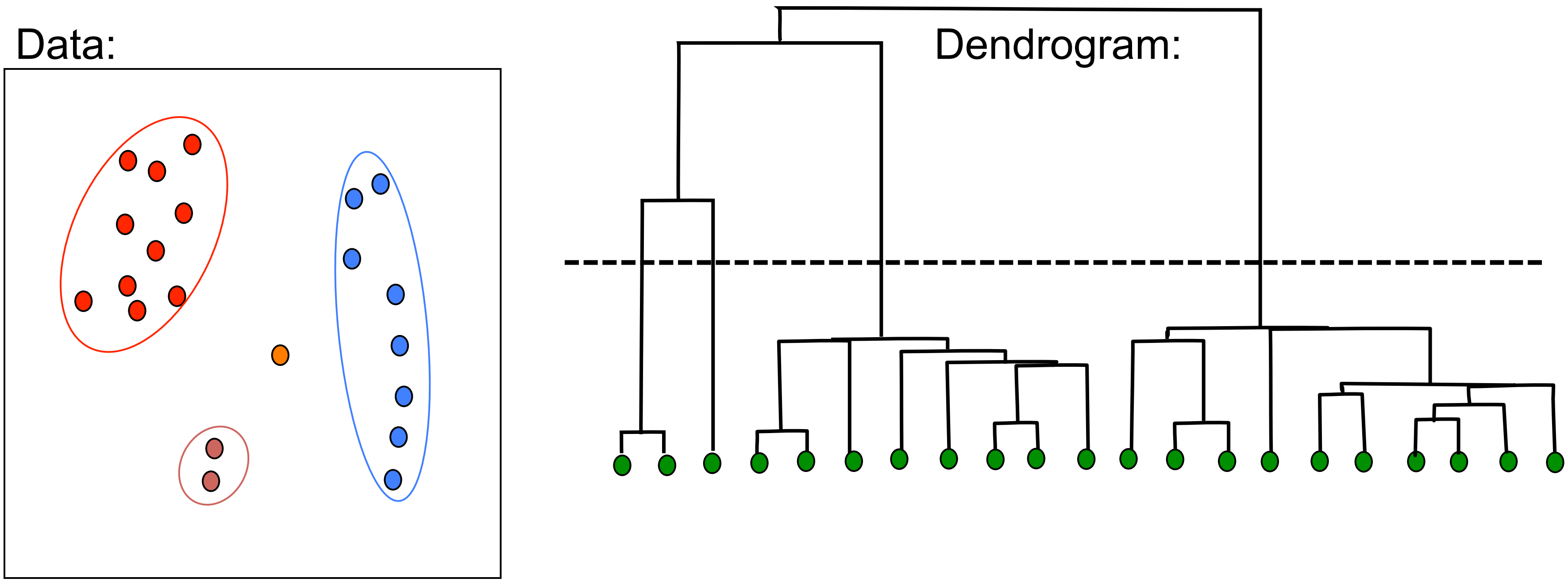


In matlab: "linkage" function (stats toolbox)

Algorithmic Complexity: $O(m^2 \log m) + (m-1)*O(m \log m) = O(m^2 \log m)$

From dendrogram to clusters

Given the sequence, can select a number of clusters or a dissimilarity threshold:

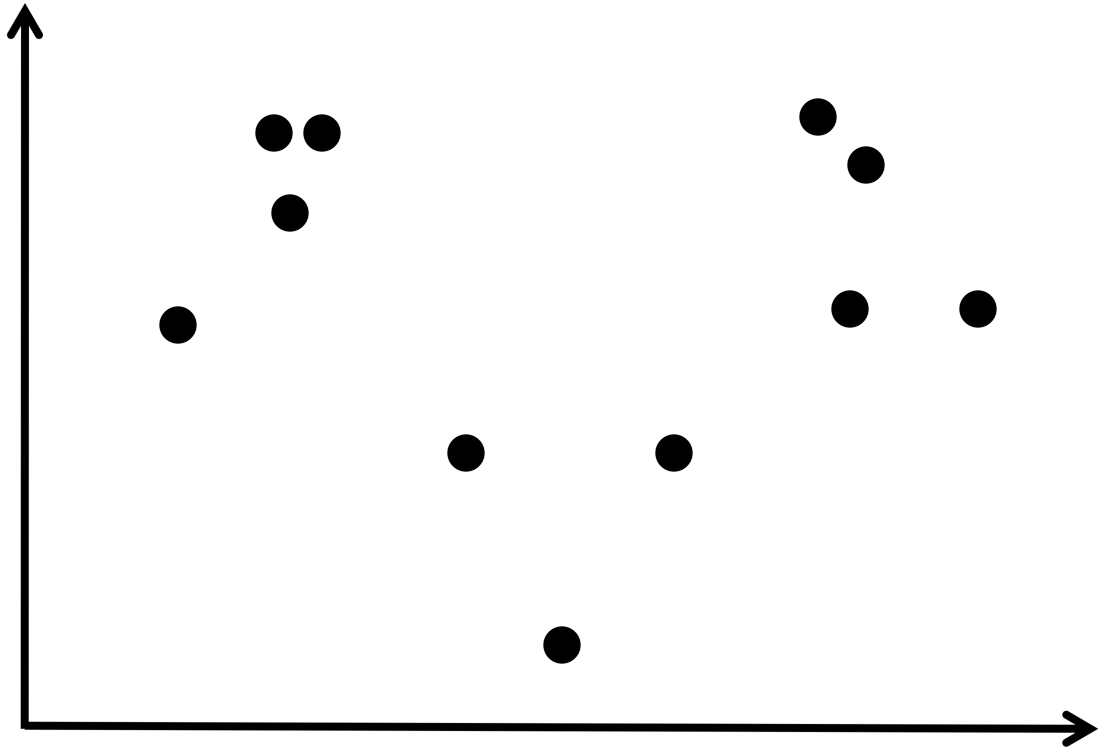


In matlab: “linkage” function (stats toolbox)

Algorithmic Complexity: $O(m^2 \log m) + (m-1)*O(m \log m) = O(m^2 \log m)$

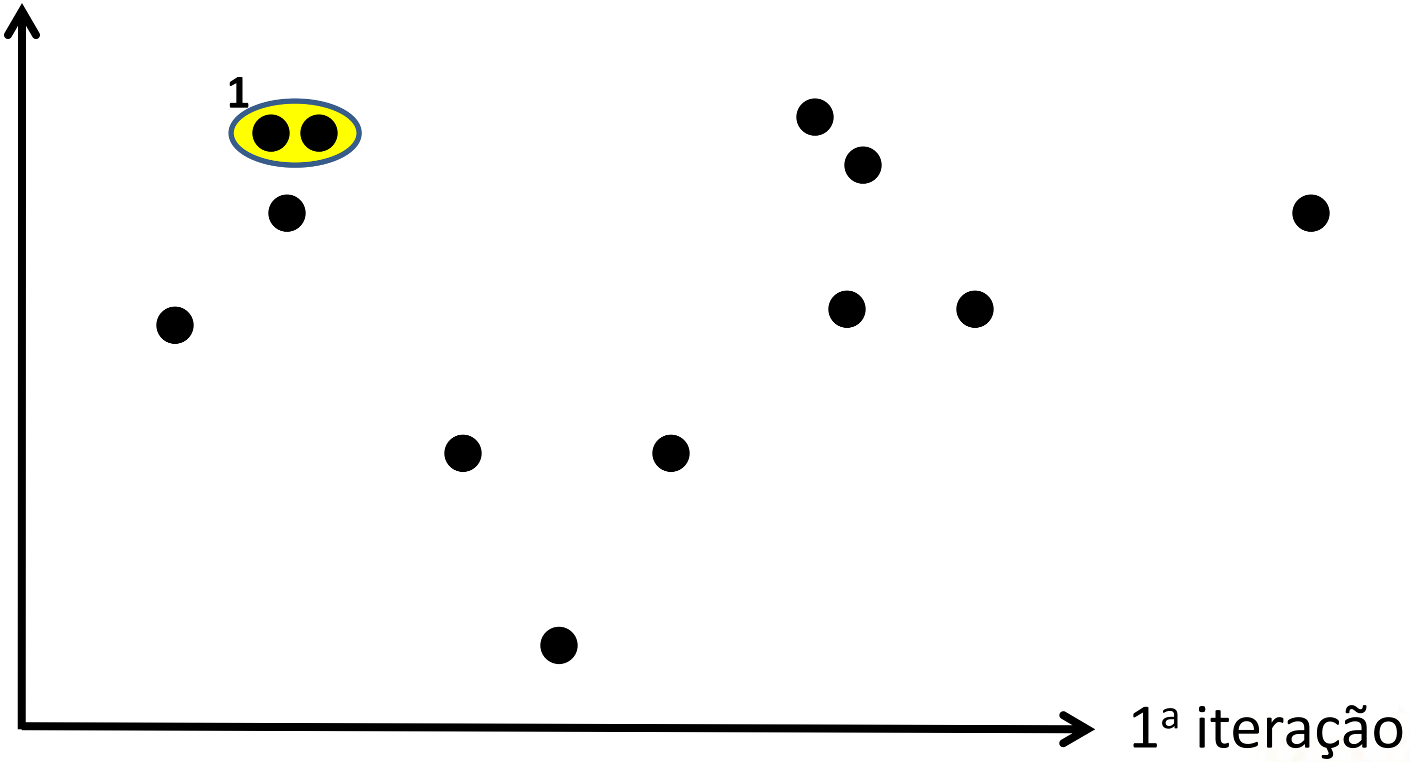
Clusterização Hierárquica (outro exemplo)

- Exemplo 1 – Aglomerativo:



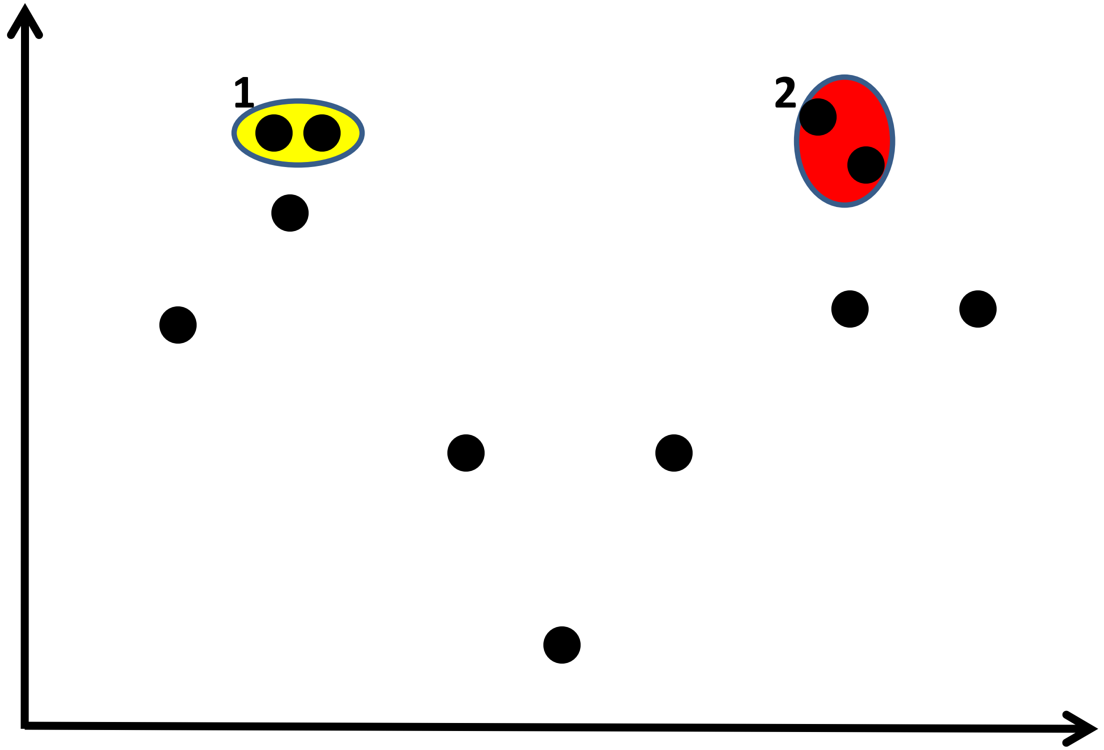
Clusterização Hierárquica

- Exemplo 1 – Aglomerativo:



Clusterização Hierárquica

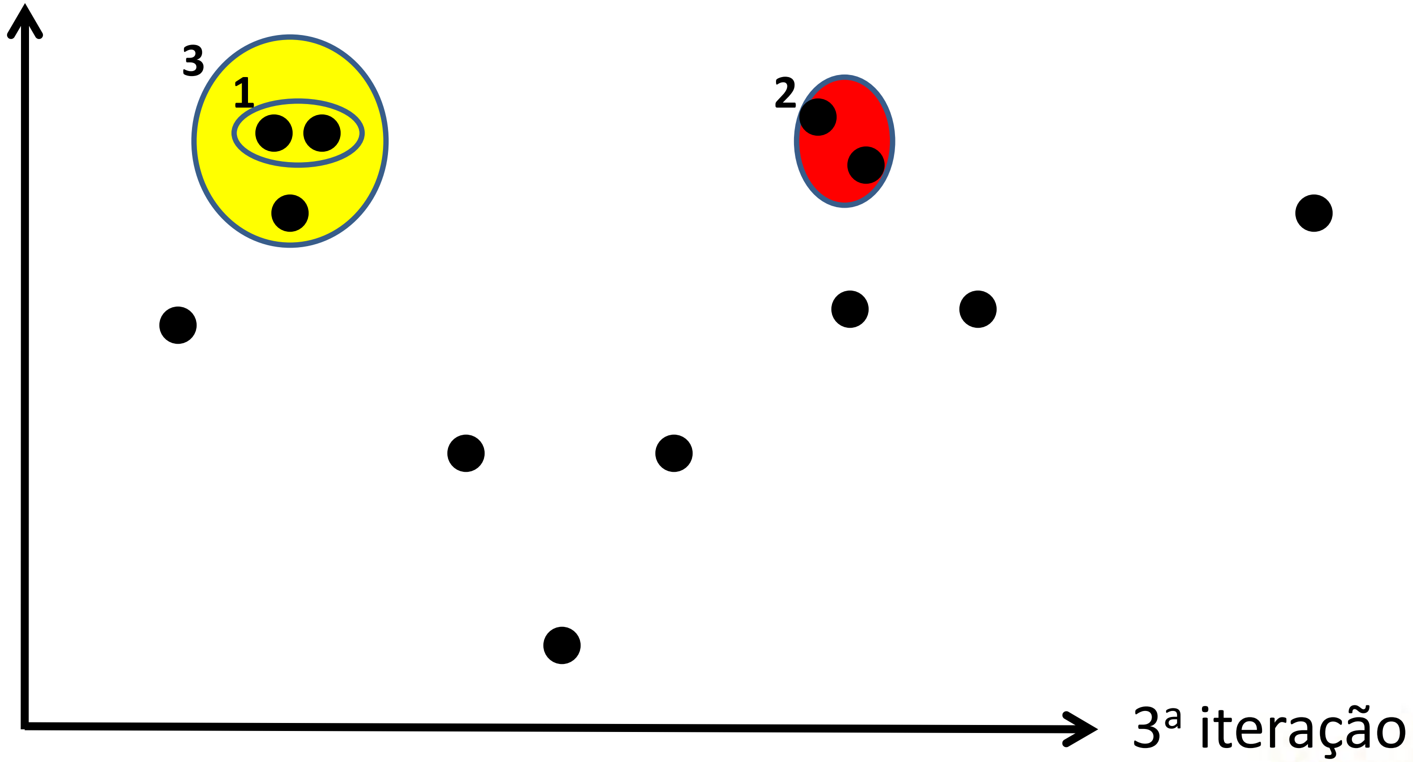
- Exemplo 1 – Aglomerativo:



2ª iteração

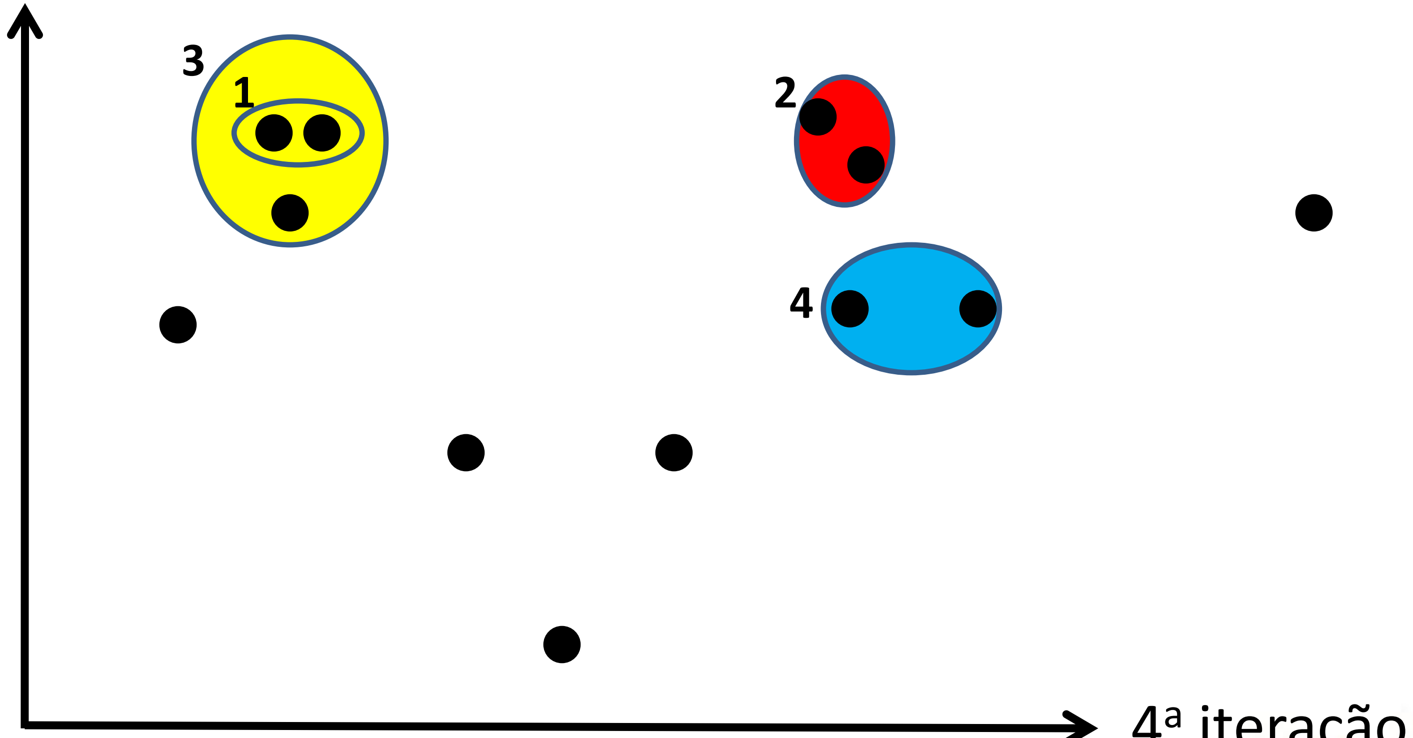
Clusterização Hierárquica

- Exemplo 1 – Aglomerativo:



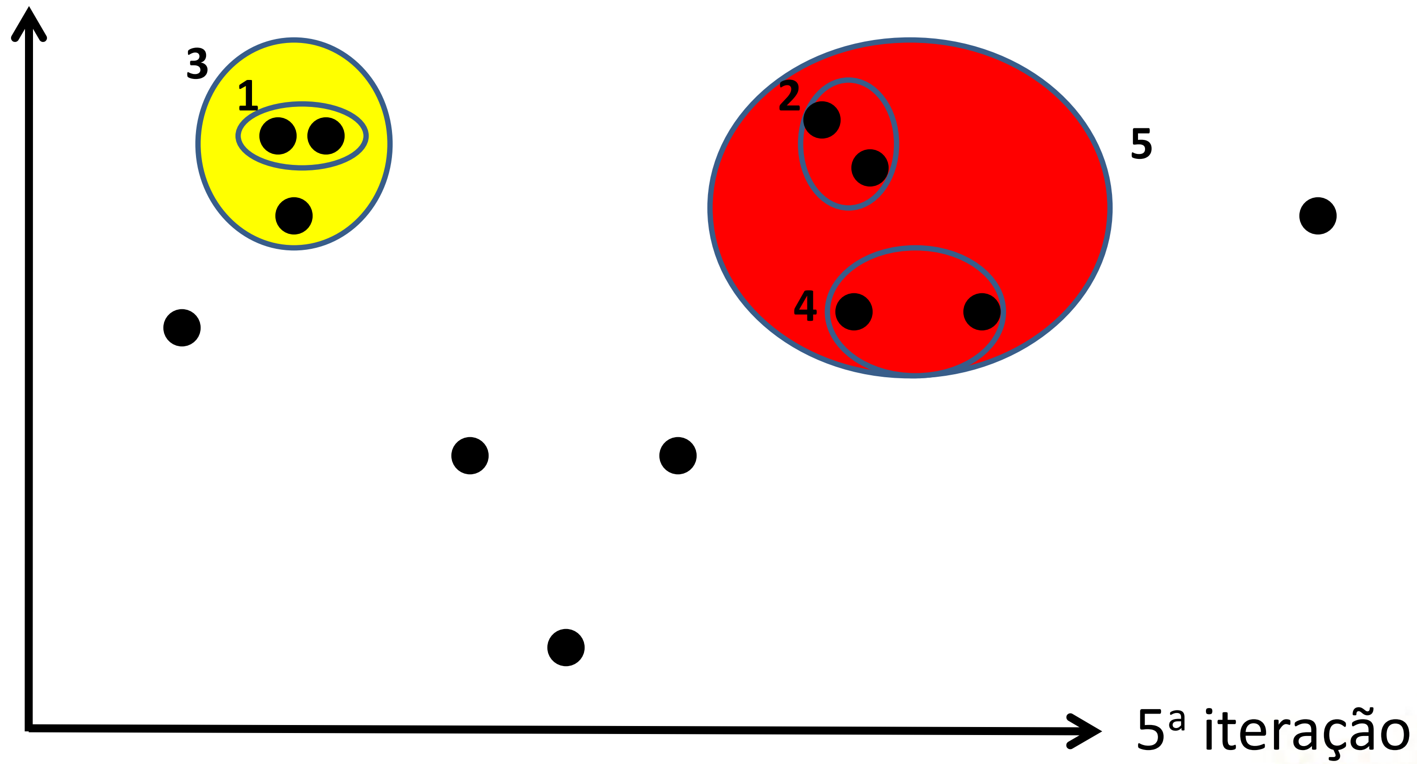
Clusterização Hierárquica

- Exemplo 1 – Aglomerativo:



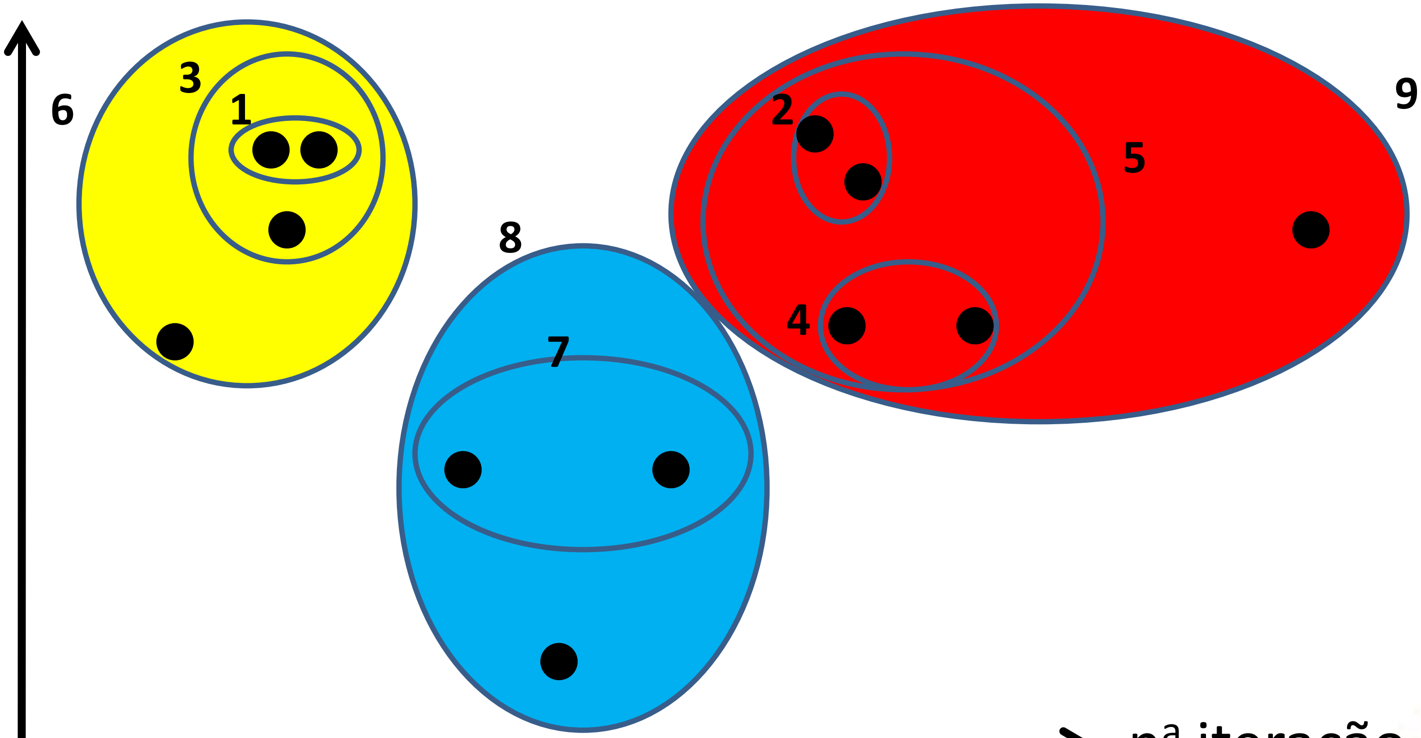
Clusterização Hierárquica

- Exemplo 1 – Aglomerativo:



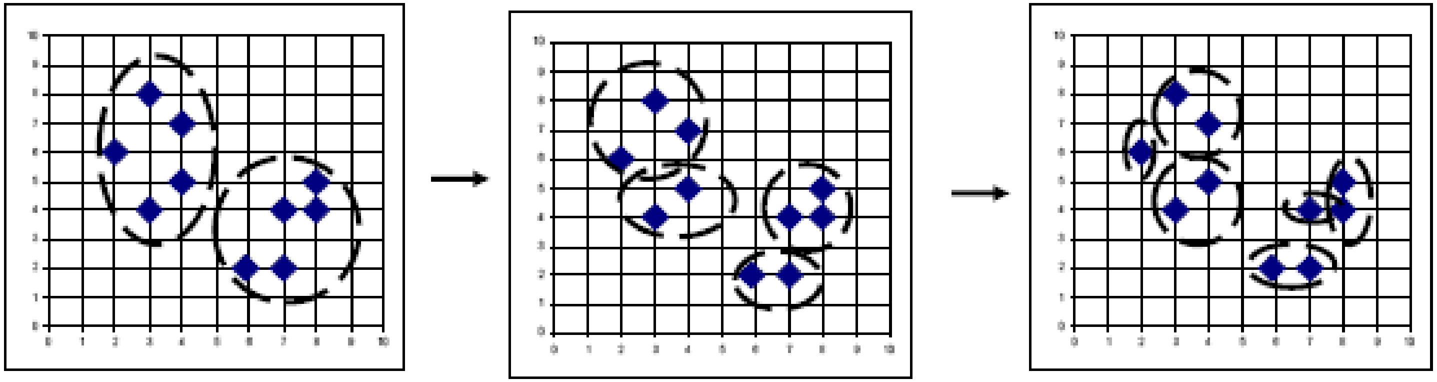
Clusterização Hierárquica

- Exemplo 1 – Aglomerativo:



Clusterização Hierárquica

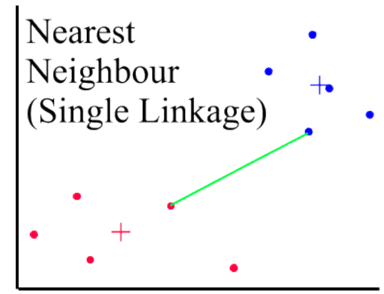
- Exemplo 2 – Divisivo:



- Processo inverso.

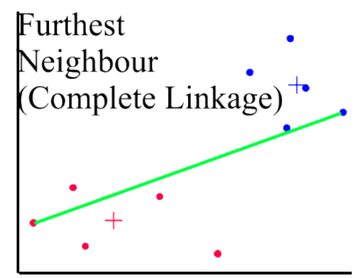
Cluster distances

$$D_{\min}(C_i, C_j) = \min_{x \in C_i,\ y \in C_j} \|x - y\|^2$$



produces minimal spanning tree.

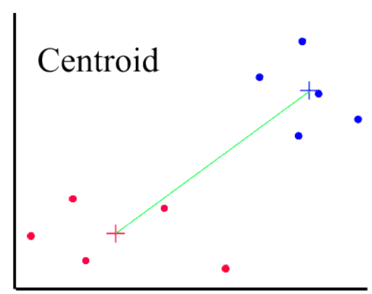
$$D_{\max}(C_i, C_j) = \max_{x \in C_i,\ y \in C_j} \|x - y\|^2$$



avoids elongated clusters.

$$D_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i,\ y \in C_j} \|x - y\|^2$$

$$D_{\text{means}}(C_i, C_j) = \|\mu_i - \mu_j\|^2$$



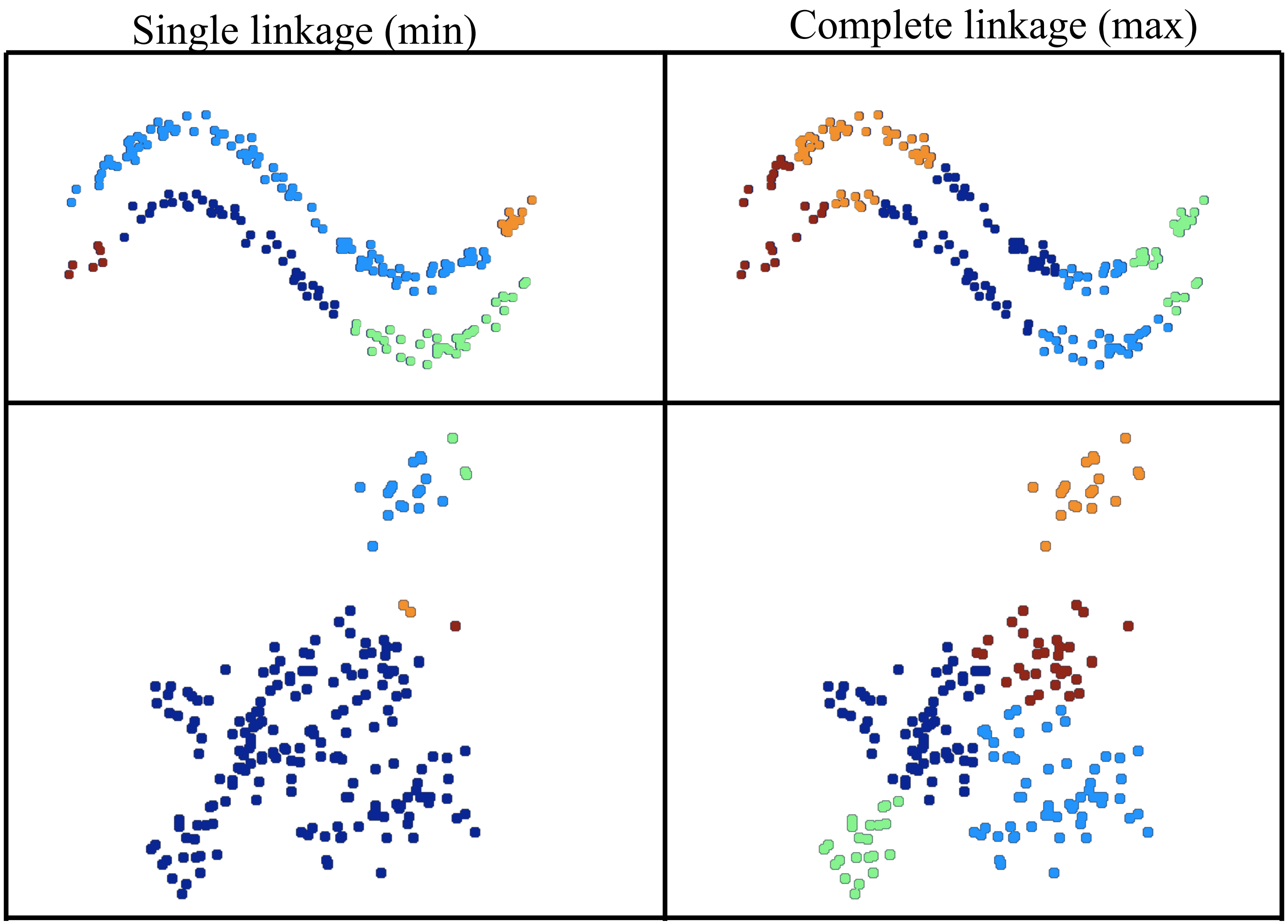
Need:

D(A,C) → D(A+B,C)

D(B,C) →

Cluster distances

- Dissimilarity choice will affect clusters created



K-Means

- É a técnica **mais simples** de aprendizagem não supervisionada.
- Consiste em fixar **k centróides** (de maneira aleatória), um para cada grupo (clusters).
- Associar cada indivíduo ao seu **centróide mais próximo**.
- Recalcular os centróides com base nos indivíduos classificados.

Algoritmo K-Means

(1) Selecione k centróides iniciais.

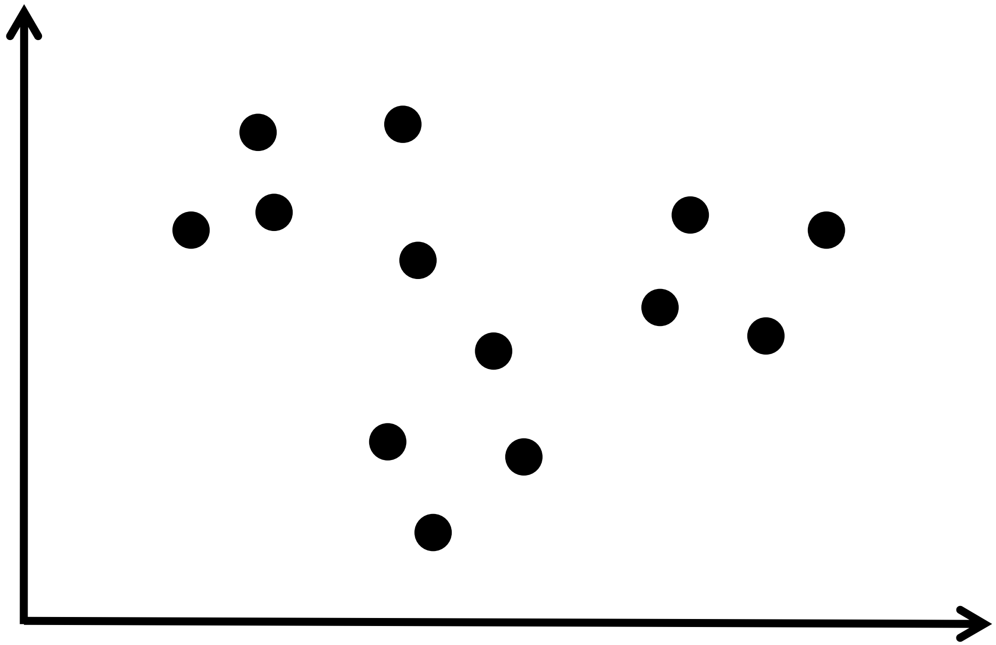
(2) Forme k clusters associando cada exemplo ao seu centróide mais próximo.

(3) Recalcule a posição dos centróides com base no centro de gravidade do cluster.

(4) Repita os passos 2 e 3 até que os centróides não sejam mais movimentados.

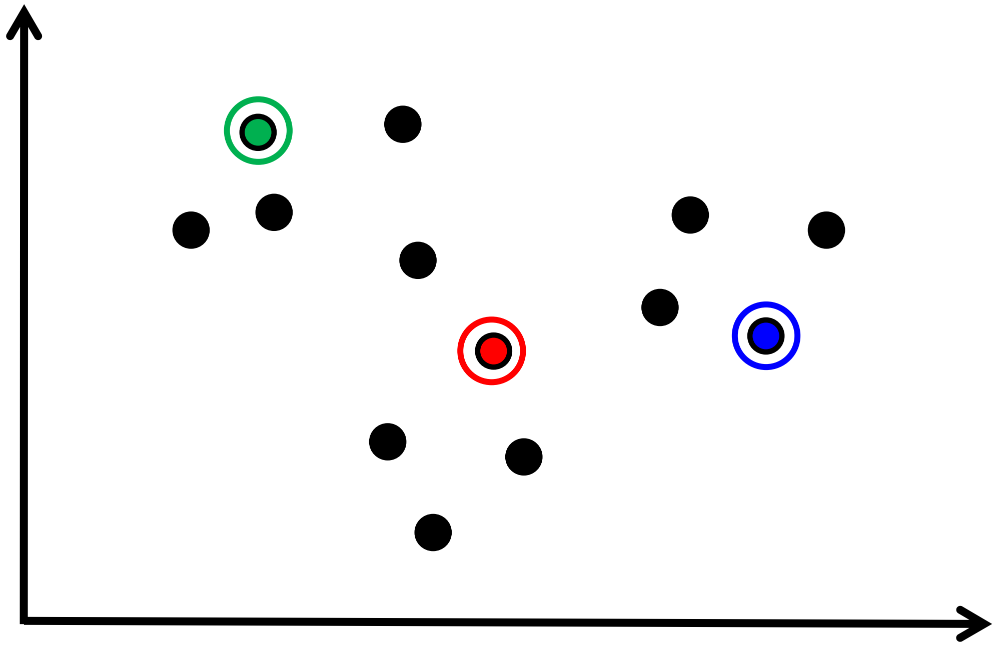
Algoritmo K-Means

- Exemplo:



Algoritmo K-Means

- Exemplo: $k = 3$

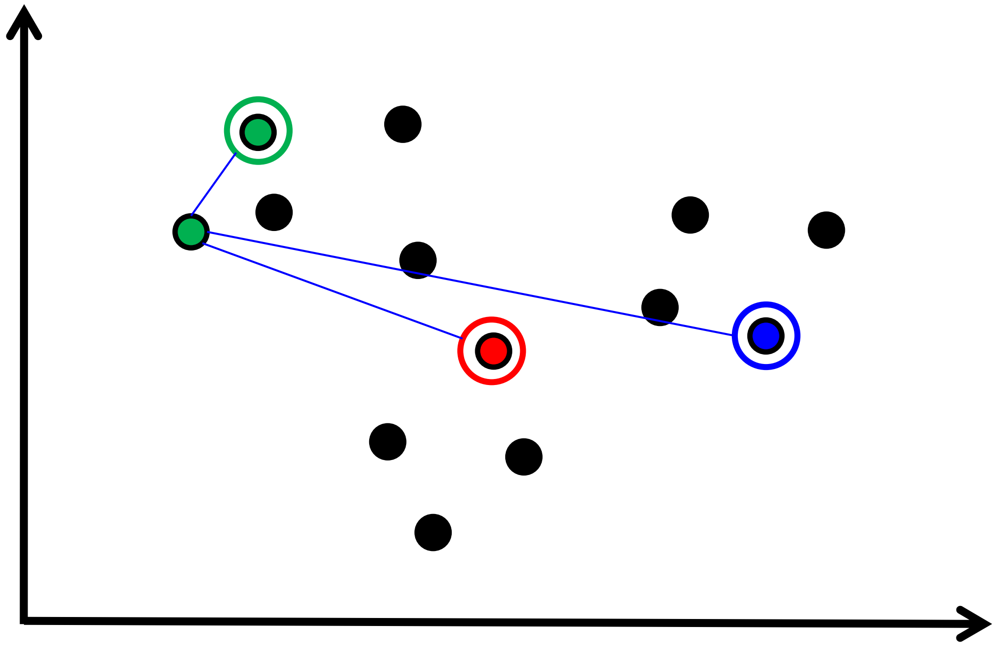


Seleciona-se k centróides iniciais.

Algoritmo K-Means

- Exemplo:

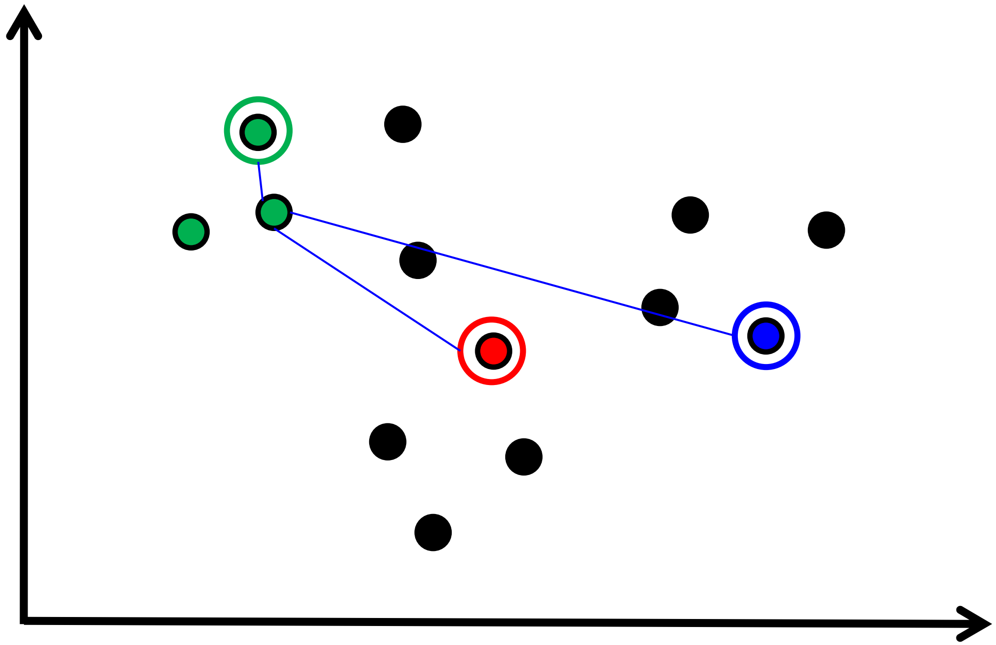
k = 3



1ª iteração

Algoritmo K-Means

- Exemplo: k = 3

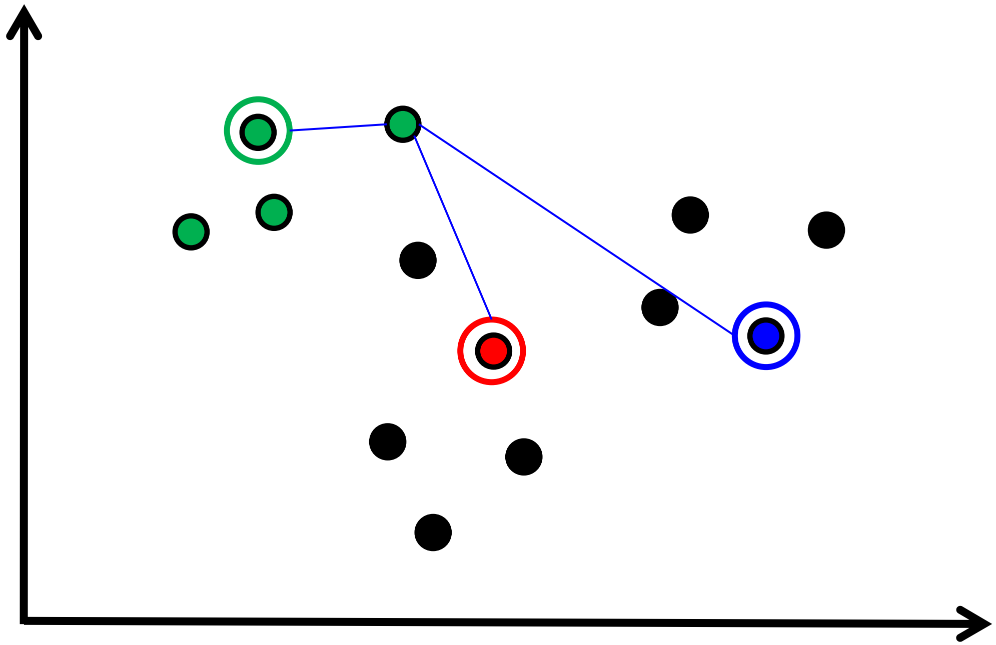


2ª iteração

Algoritmo K-Means

- Exemplo:

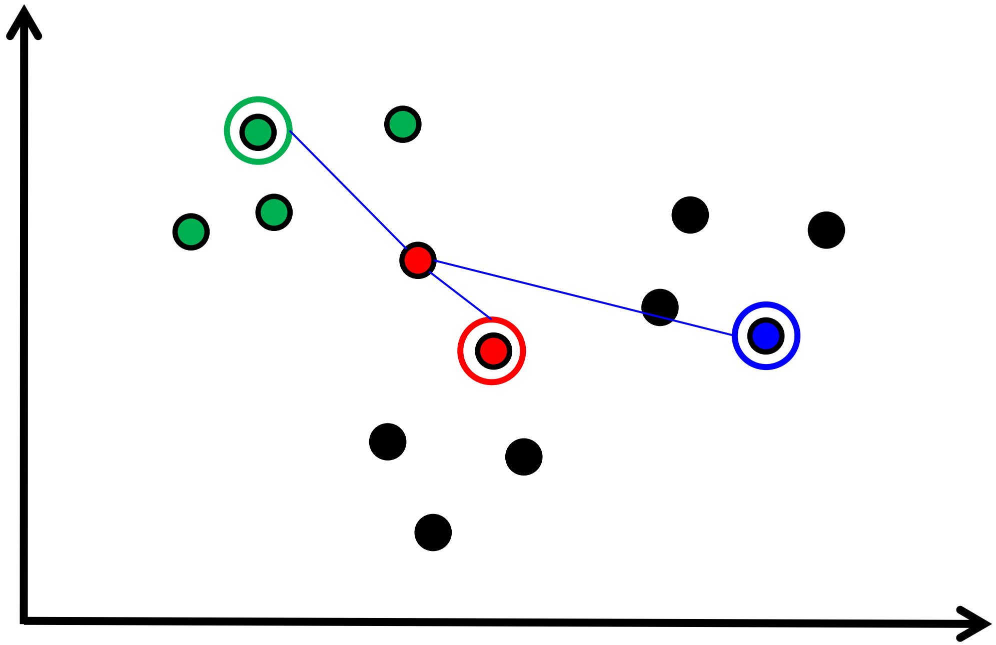
k = 3



3ª iteração

Algoritmo K-Means

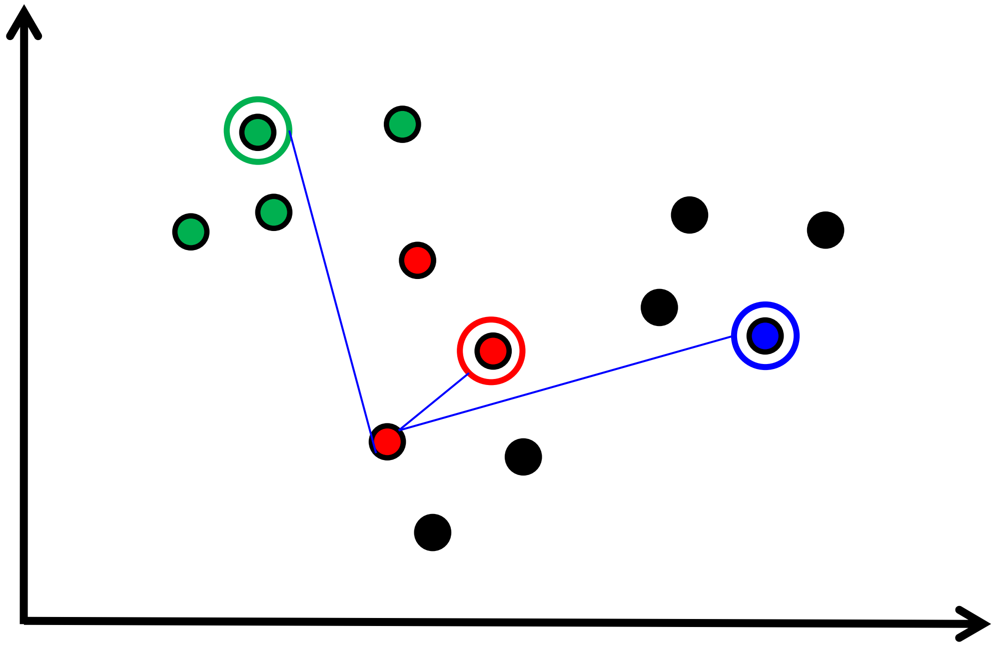
- Exemplo: k = 3



4ª iteração

Algoritmo K-Means

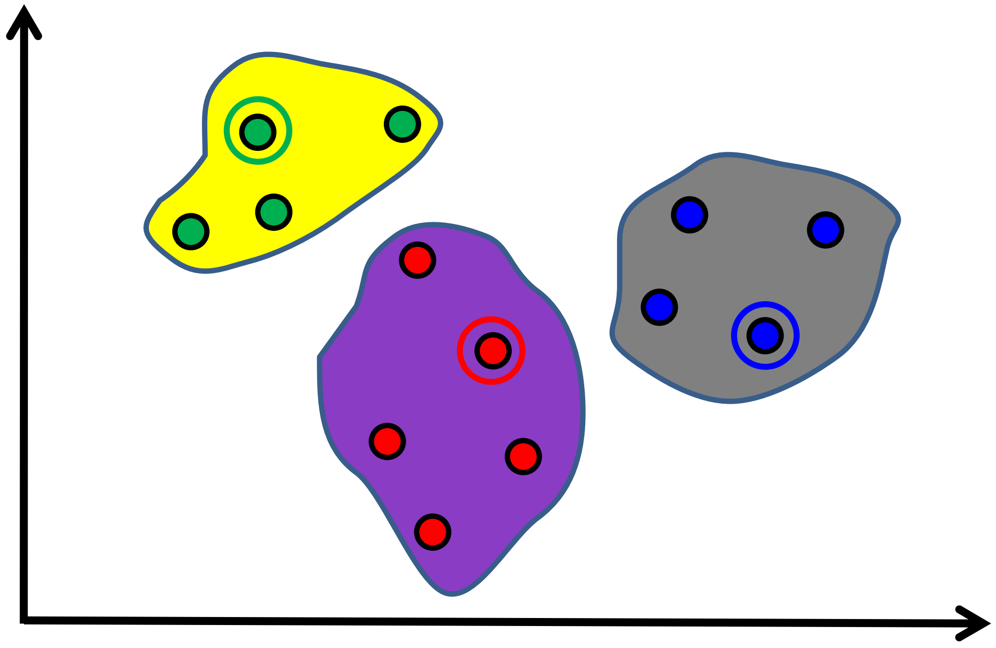
- Exemplo: k = 3



5ª iteração

Algoritmo K-Means

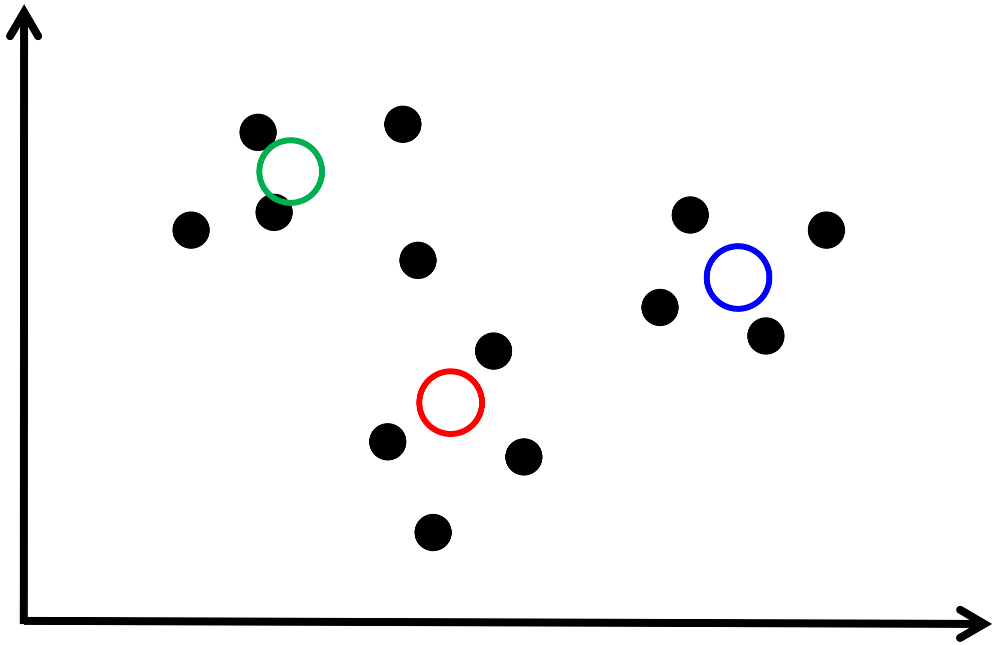
- Exemplo: k = 3



nª iteração

Algoritmo K-Means

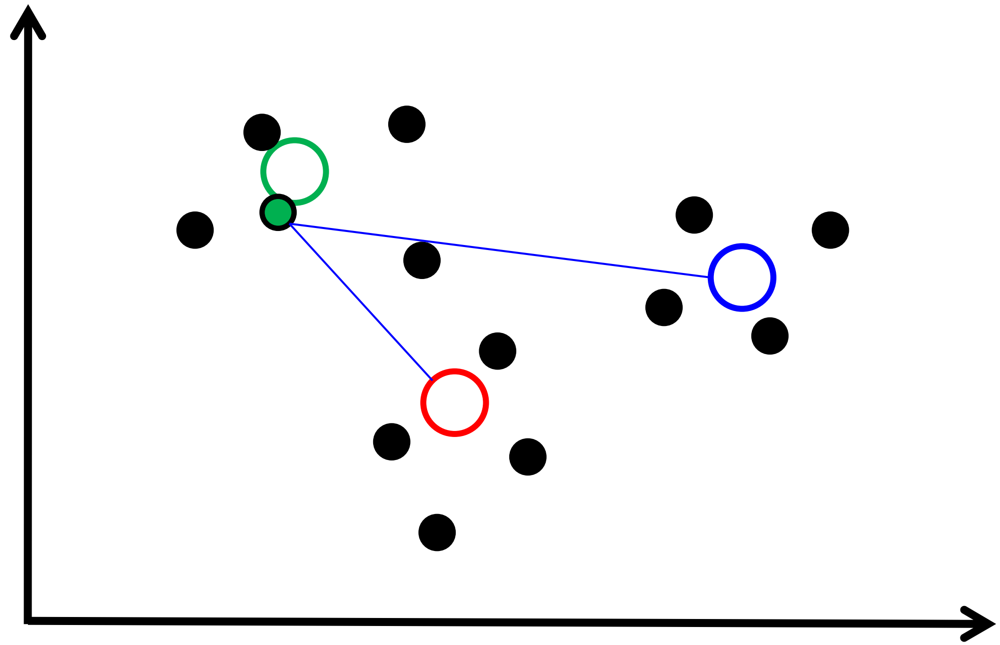
- Exemplo: k = 3



Repite-se os passos anteriores até que os centróides não se movam mais.

Algoritmo K-Means

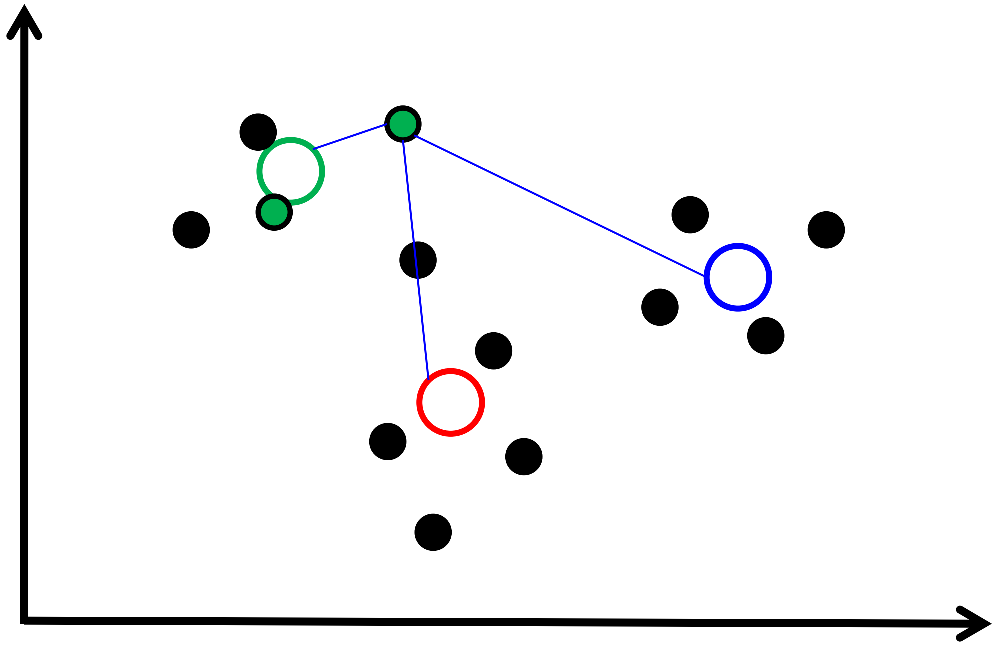
- Exemplo: k = 3



1ª iteração

Algoritmo K-Means

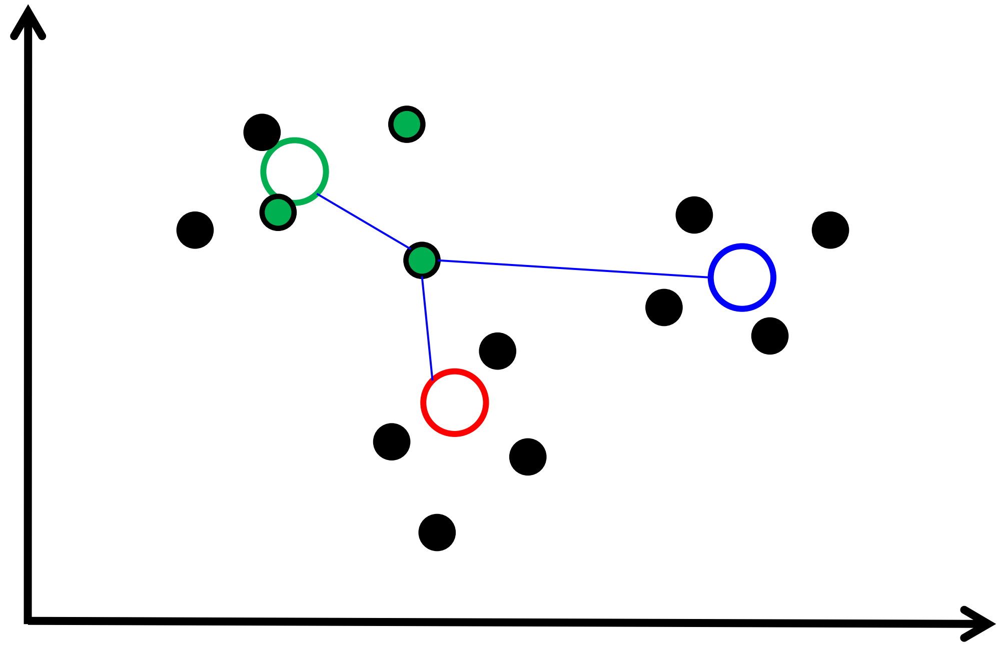
- Exemplo: k = 3



2ª iteração

Algoritmo K-Means

- Exemplo: k = 3



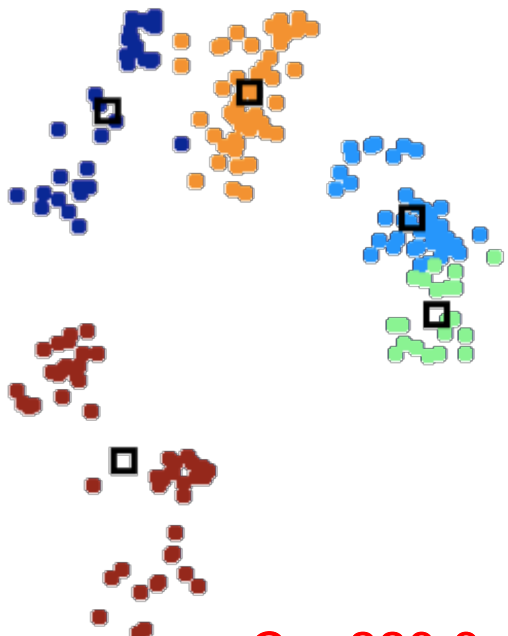
3ª iteração

Initialization

- Multiple local optima, depending on initialization

- Try different (randomized) initializations
- Can use cost C to decide which we prefer

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$



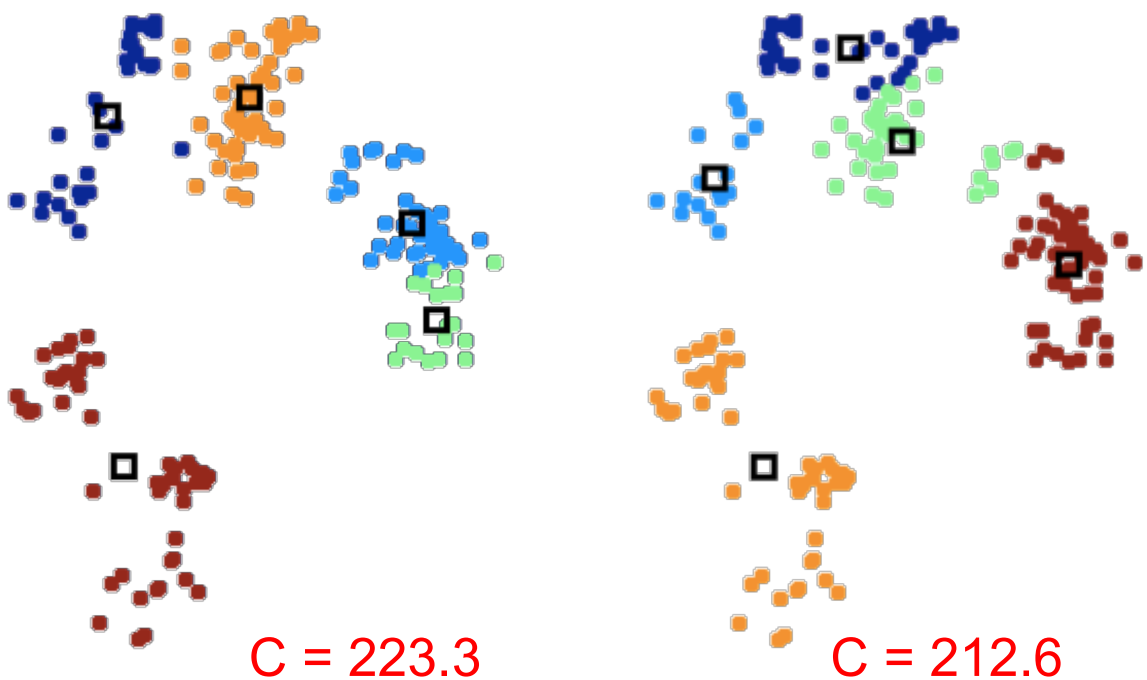
C = 223.3

Initialization

- Multiple local optima, depending on initialization

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

- Try different (randomized) initializations
- Can use cost C to decide which we prefer

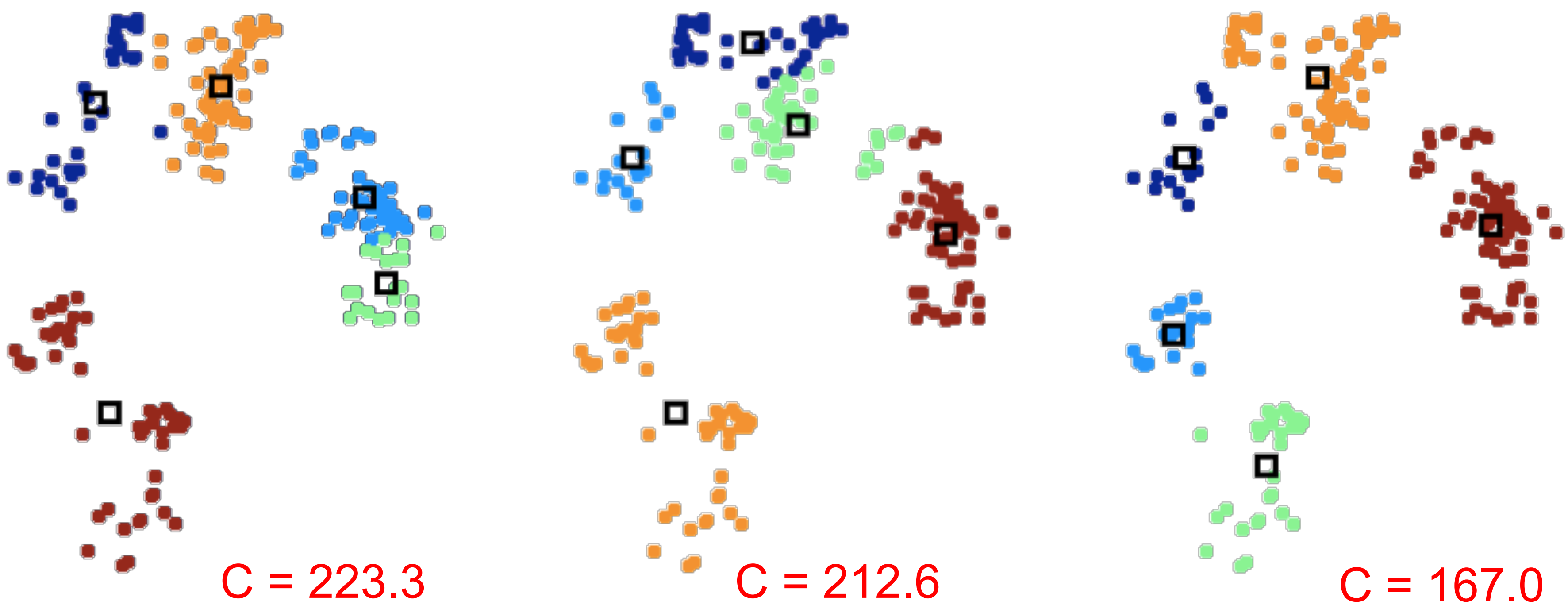


Initialization

- Multiple local optima, depending on initialization

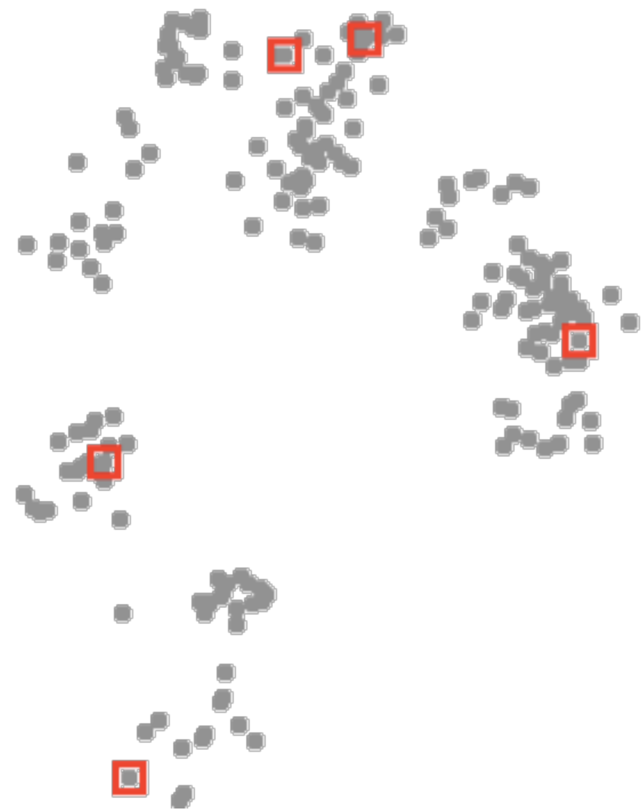
- Try different (randomized) initializations
- Can use cost C to decide which we prefer

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$



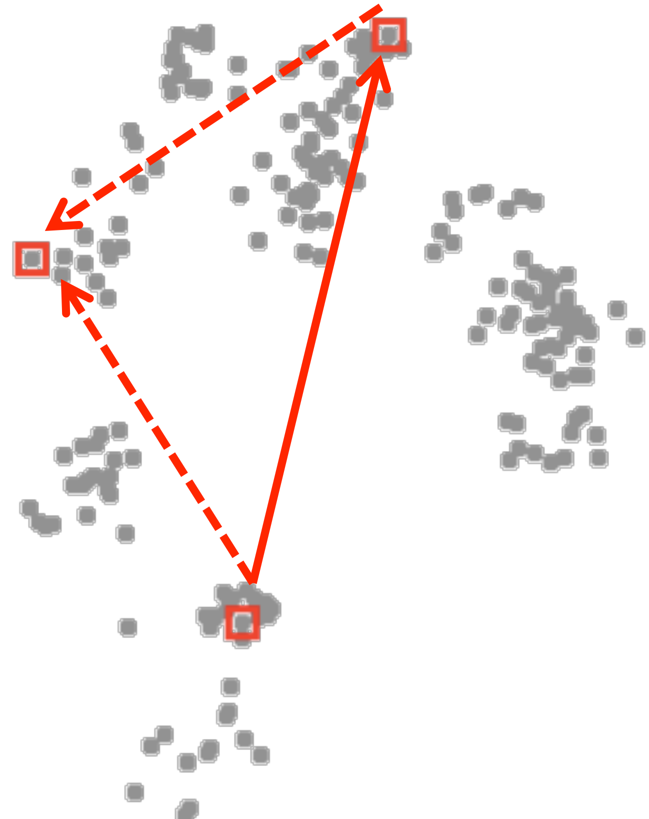
Initialization methods

- Random
  - Usually, choose random data index
  - Ensures centers are near some data
  - Issue: may choose nearby points



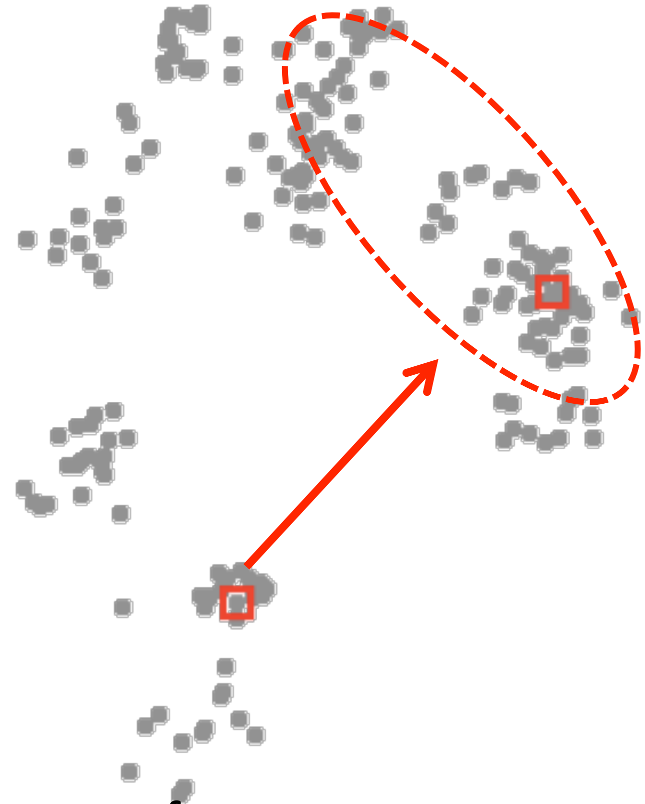
Initialization methods

- Random
  - Usually, choose random data index
  - Ensures centers are near some data
  - Issue: may choose nearby points



- Distance-based
  - Start with one random data point
  - Find the point farthest from the clusters chosen so far
  - Issue: may choose outliers

Initialization methods



- Random
  - Usually, choose random data index
  - Ensures centers are near some data
  - Issue: may choose nearby points
- Distance-based
  - Start with one random data point
  - Find the point farthest from the clusters chosen so far
  - Issue: may choose outliers
- Random + distance (“k-means++”) (Arthur & Vassilvitskii, 2007)
  - Choose next points “far but randomly”

    $p(x) \propto$ squared distance from x to current centers
  - Likely to put a cluster far away, in a region with lots of data

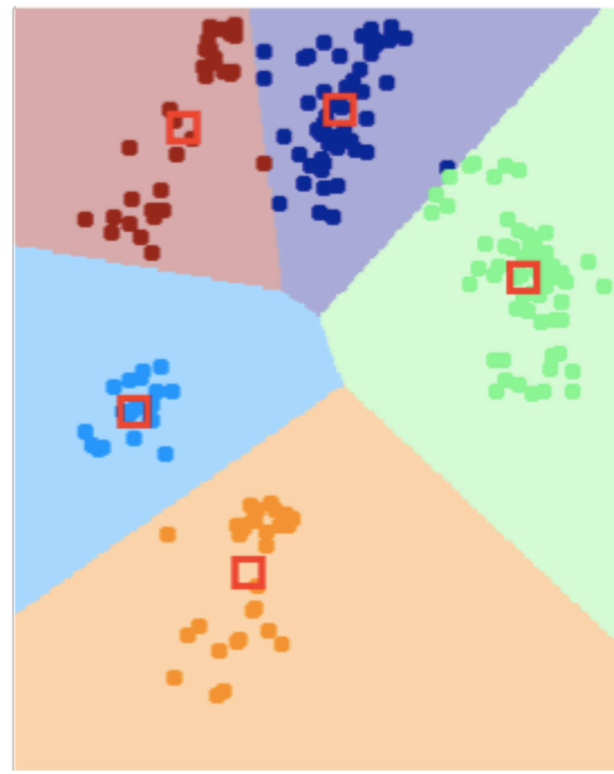
Out-of-sample points

- Often want to use clustering on new data
- Easy for k-means: choose nearest cluster center

```
% perform clustering
[Z , mu] = kmeans(X, K);

% cluster id = nearest center
L = knnClassify(mu, (1:K)', 1);

% assign in- or out-of-sample points
Z = predict(L, X);
```



Choosing the number of clusters

- With cost function

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

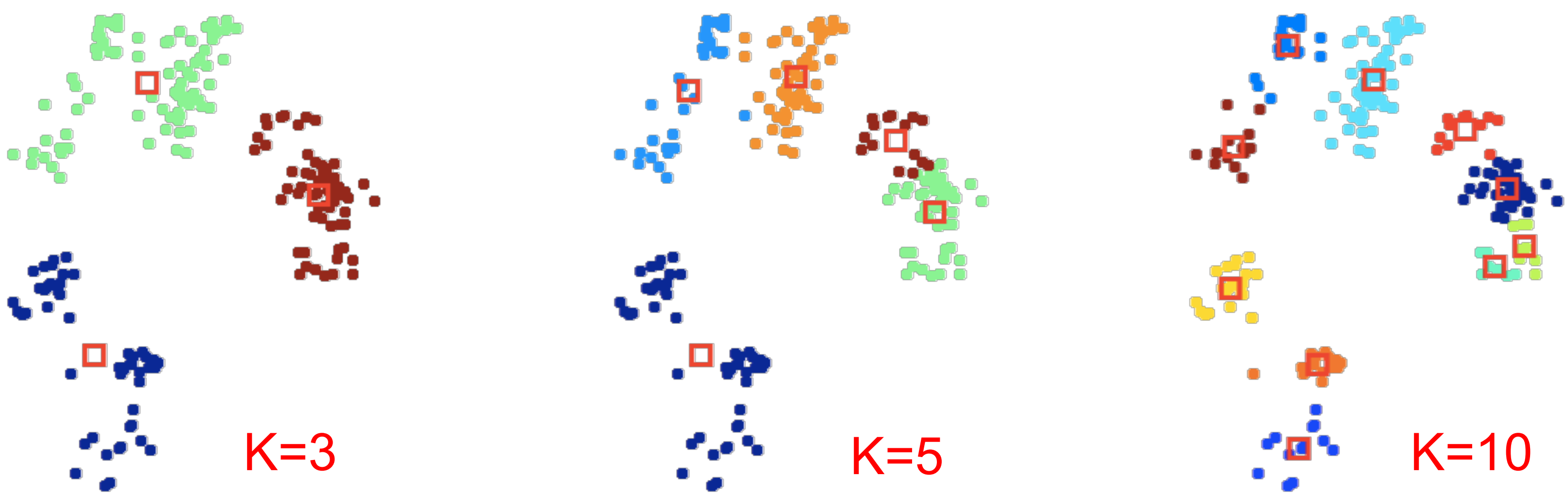
what is the optimal value of k?

Choosing the number of clusters

- With cost function

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

what is the optimal value of k?



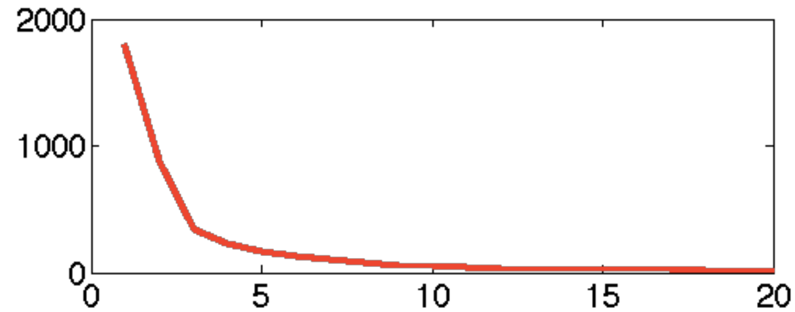
Choosing the number of clusters

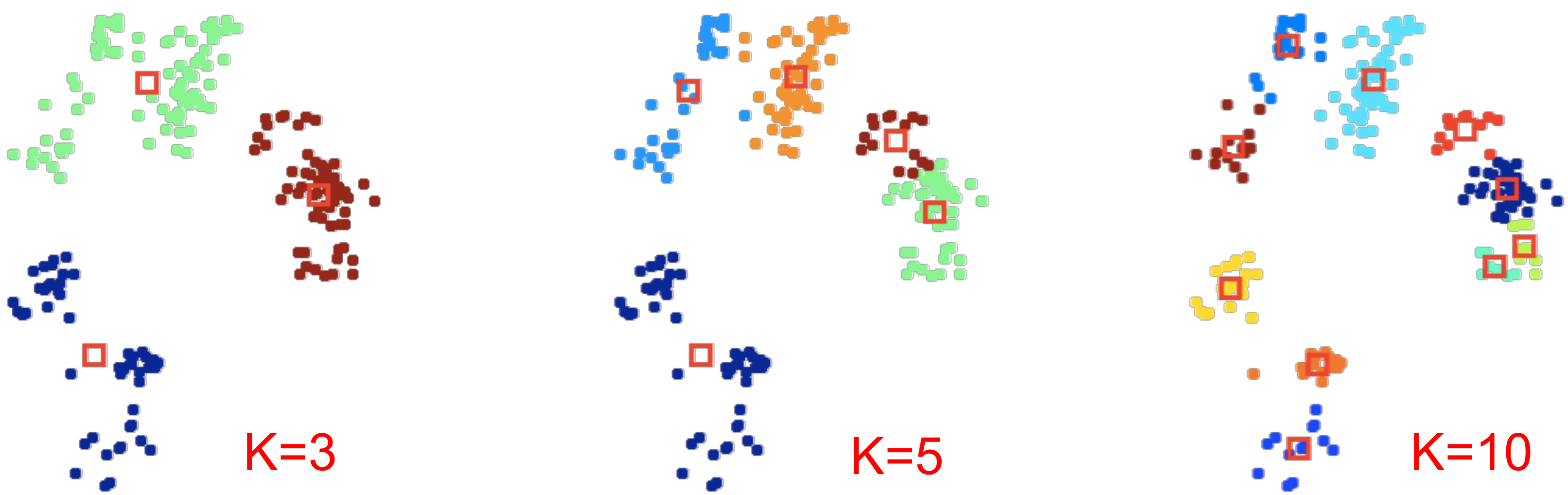
- With cost function

$$C(\underline{z},\underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

what is the optimal value of k?

- Cost always decreases with k!



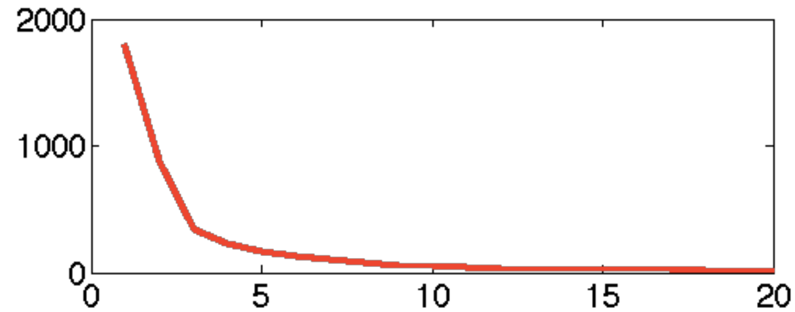


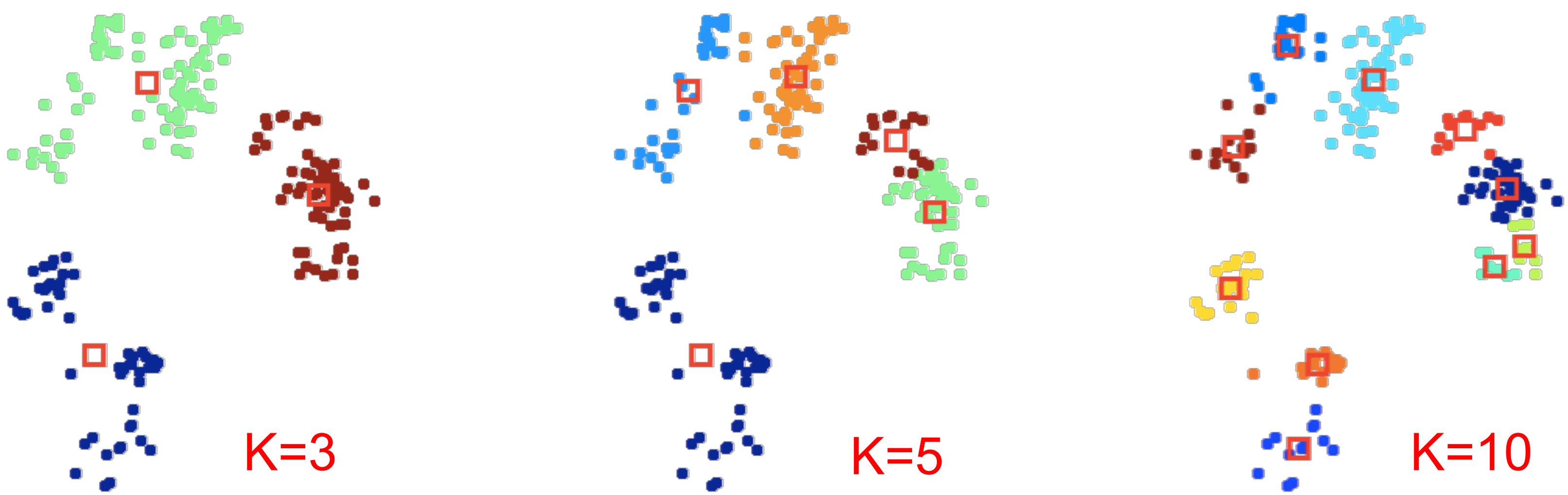
Choosing the number of clusters

- With cost function

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

  what is the optimal value of k?
- Cost always decreases with k!
- A model complexity issue…



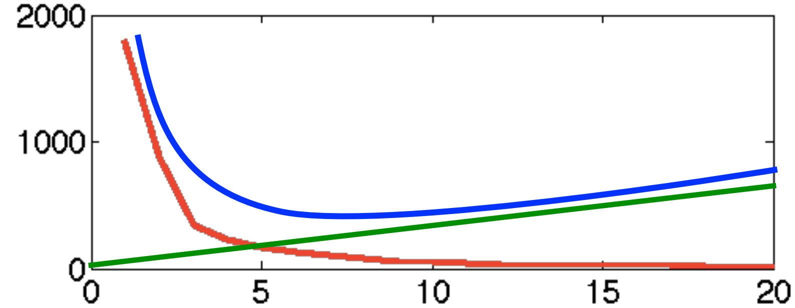


Choosing the number of clusters

- With cost function

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

what is the optimal value of k?



- Cost always decreases with k!
- A model complexity issue…

- One solution is to penalize for complexity
  - Add penalty: Total = Error + Complexity
  - Now more clusters can increase cost, if they don't help "enough"
  - Ex: simplified BIC penalty

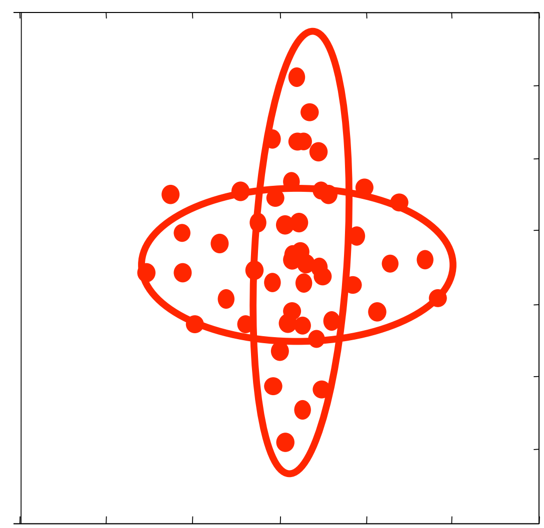
$$J(\underline{z}, \underline{\mu}) = \log \left[\frac{1}{m\,d} \sum_i \|x_i - \mu_{z_i}\|^2 \right] + k \frac{\log m}{m}$$

  - More precise version: see e.g. "X-means" (Pelleg & Moore 2000)

Mixtures of Gaussians

- K-means algorithm
  - Assigned each example to exactly one cluster
  - What if clusters are overlapping?
    - Hard to tell which cluster is right
    - Maybe we should try to remain uncertain
  - Used Euclidean distance
  - What if cluster has a non-circular shape?

- Gaussian mixture models
  - Clusters modeled as Gaussians
    - Not just by their mean
  - EM algorithm: assign data to cluster with some *probability*
  - Gives probability model of x! ("generative")



Mixture models in 1-d

- Observations $x_1 \ldots x_n$
  - K=2 Gaussians with unknown μ, σ^2
  - estimation trivial if we know the source of each observation

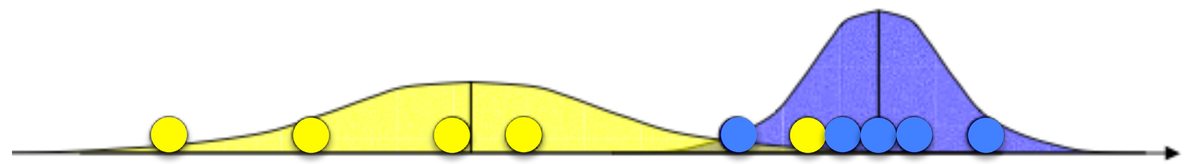
Mixture models in 1-d

Observations $x_1 \ldots x_n$

- K=2 Gaussians with unknown μ, σ^2
- estimation trivial if we know the source of each observation

$$\mu_b = \frac{x_1 + x_2 + \ldots + x_{n_b}}{n_b}$$

$$\sigma_b^2 = \frac{(x_1 - \mu_1)^2 + \ldots + (x_n - \mu_n)^2}{n_b}$$

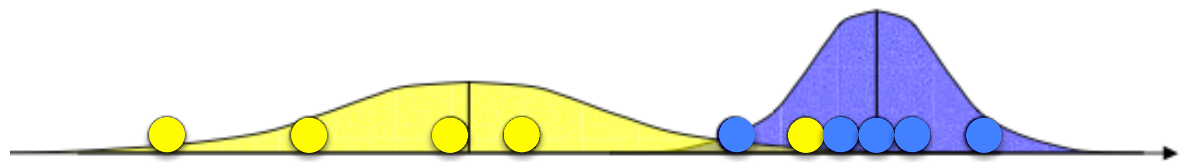


Mixture models in 1-d

- Observations $x_1 \ldots x_n$
  - K=2 Gaussians with unknown μ, σ^2
  - estimation trivial if we know the source of each observation

$$\mu_b = \frac{x_1 + x_2 + \ldots + x_{n_b}}{n_b}$$

$$\sigma_b^2 = \frac{(x_1 - \mu_1)^2 + \ldots + (x_n - \mu_n)^2}{n_b}$$



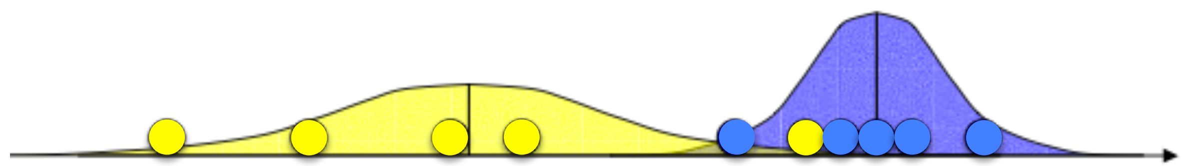
- What if we don't know the source?

Mixture models in 1-d

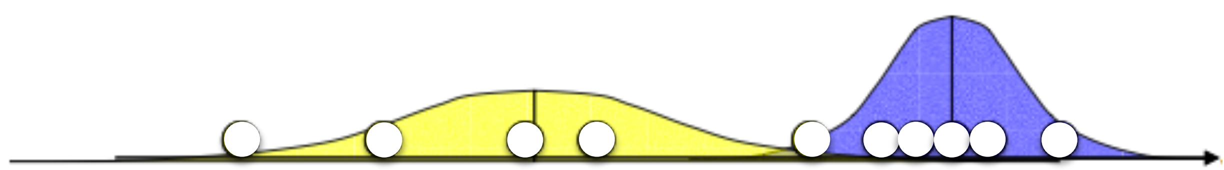
- Observations $x_1 \ldots x_n$
  - K=2 Gaussians with unknown μ, σ^2
  - estimation trivial if we know the source of each observation

$$\mu_b = \frac{x_1 + x_2 + \ldots + x_{n_b}}{n_b}$$

$$\sigma_b^2 = \frac{(x_1 - \mu_1)^2 + \ldots + (x_n - \mu_n)^2}{n_b}$$



- What if we don’t know the source?
- If we knew parameters of the Gaussians (μ, σ^2)
  - can guess whether point is more likely to be a or b

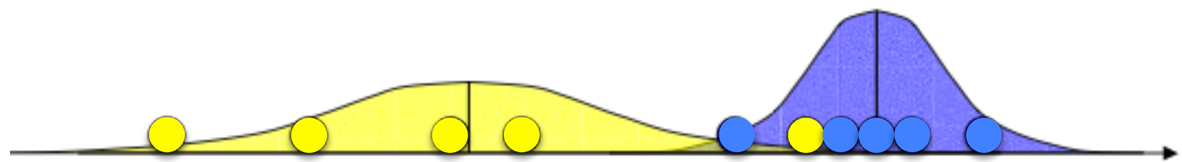


Mixture models in 1-d

- Observations $x_1 \ldots x_n$
  - K=2 Gaussians with unknown μ, σ^2
  - estimation trivial if we know the source of each observation

$$\mu_b = \frac{x_1 + x_2 + \ldots + x_{n_b}}{n_b}$$

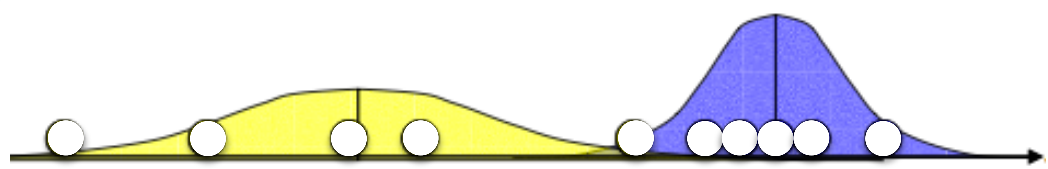
$$\sigma_b^2 = \frac{(x_1-\mu_1)^2 + \ldots + (x_n-\mu_n)^2}{n_b}$$



- What if we don't know the source?
- If we knew parameters of the Gaussians (μ, σ^2)
  - can guess whether point is more likely to be a or b

$$P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i-\mu_b)^2}{2\sigma_b^2}\right)$$



Expectation Maximization (EM)

- Chicken and egg problem
  - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
  - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)

Expectation Maximization (EM)

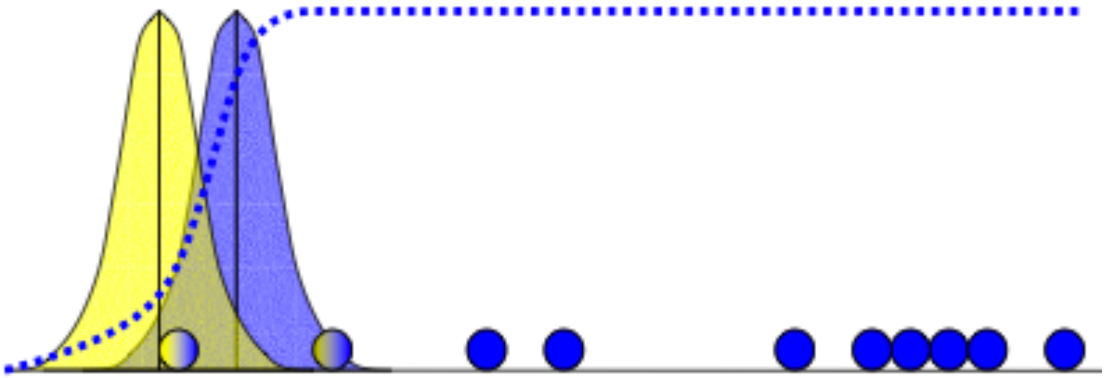
- Chicken and egg problem
  - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
  - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM algorithm
  - start with two randomly placed Gaussians (μ_a, σ_a^2), (μ_b, σ_b^2)
  - for each point: $P(b|x_i)$ = does it look like it came from b?
  - adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them
  - iterate until convergence

Expectation Maximization (EM)

- Chicken and egg problem
  - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
  - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM algorithm
  - start with two randomly placed Gaussians (μ_a, σ_a^2), (μ_b, σ_b^2)
  - E-step: for each point: $P(b|x_i)$ = does it look like it came from b?
  - M-step: adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them
  - iterate until convergence

EM: 1-d example

EM: 1-d example



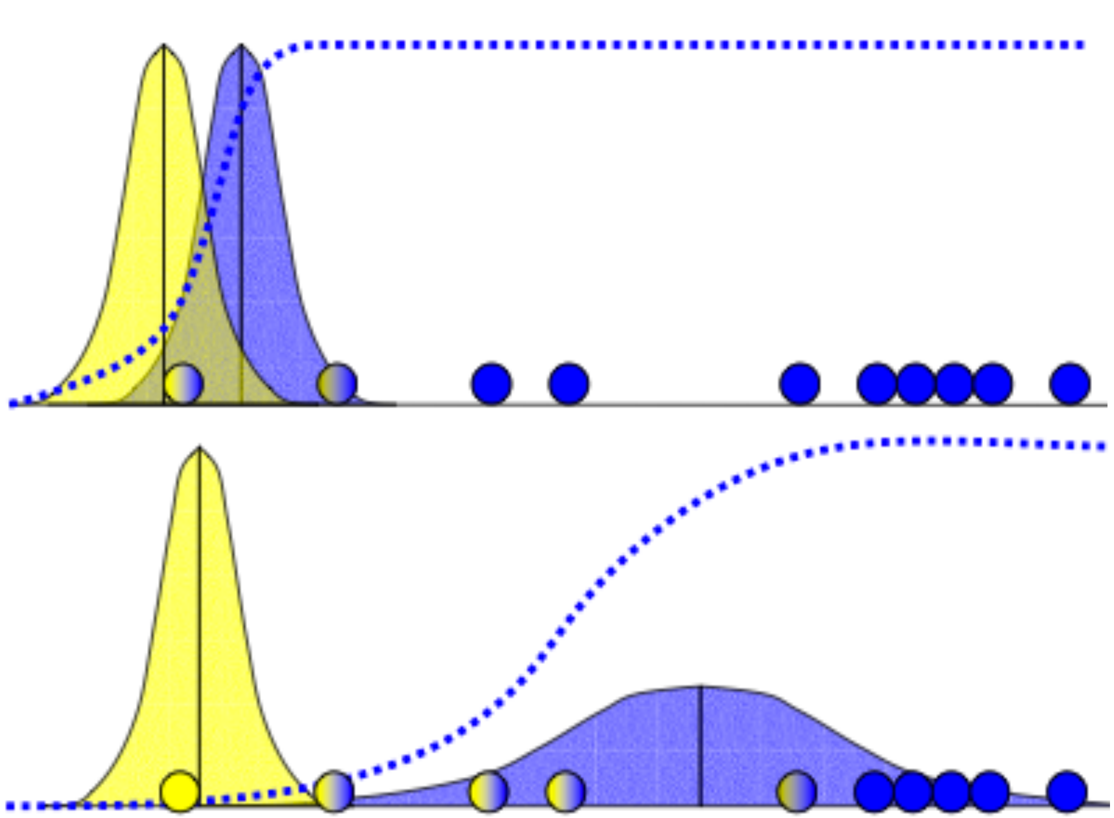
$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

P(blue|x) << P(yellow|x)

EM: 1-d example



$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

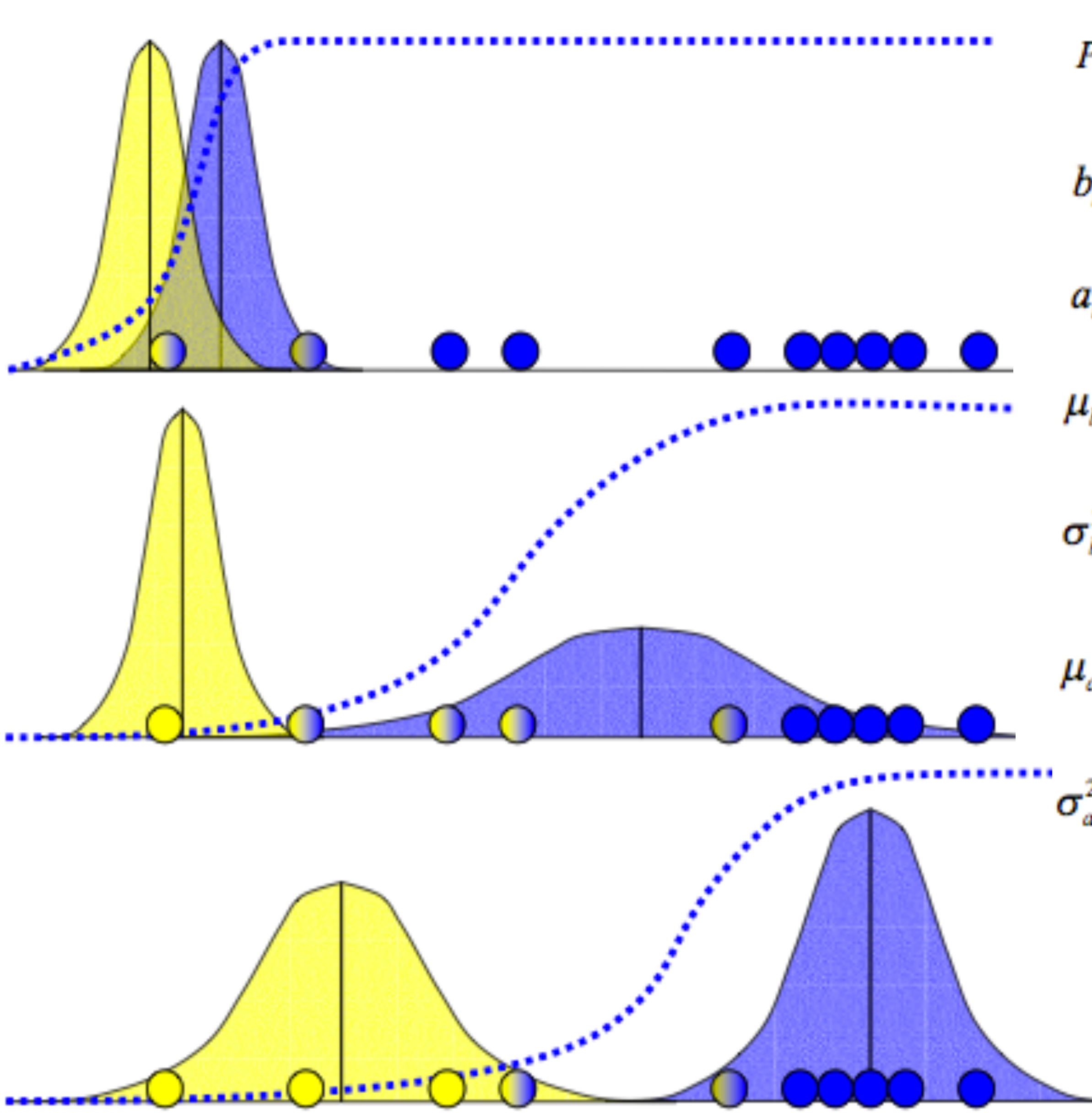
$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_{n_b}}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1(x_1 - \mu_1)^2 + \ldots + b_n(x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

EM: 1-d example



$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_{n_b}}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_1)^2 + \ldots + b_n (x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \ldots + a_n x_{n_b}}{a_1 + a_2 + \ldots + a_n}$$

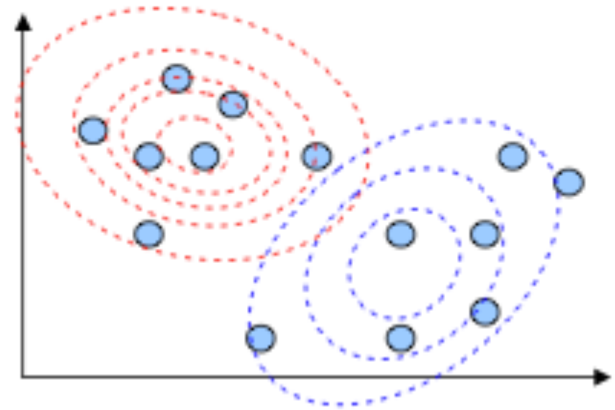
$$\sigma_a^2 = \frac{a_1 (x_1 - \mu_1)^2 + \ldots + a_n (x_n - \mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

could also estimate priors:

P(b) = (b_1 + b_2 + … b_n) / n P(a) = 1 – P(b)

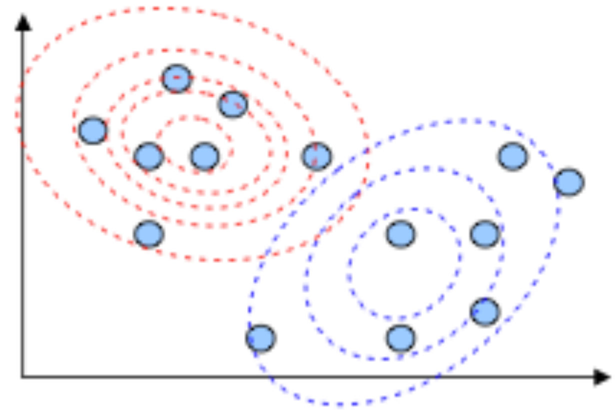
Gaussian mixture models: d>1

- Data with d attributes, from k sources
- Each source c is a Gaussian
- Iteratively estimate parameters:
  - prior: what % of instances came from source c?



$$P(c) = \frac{1}{n}\sum_{i=1}^{n} P(c \mid \vec{x}_i)$$

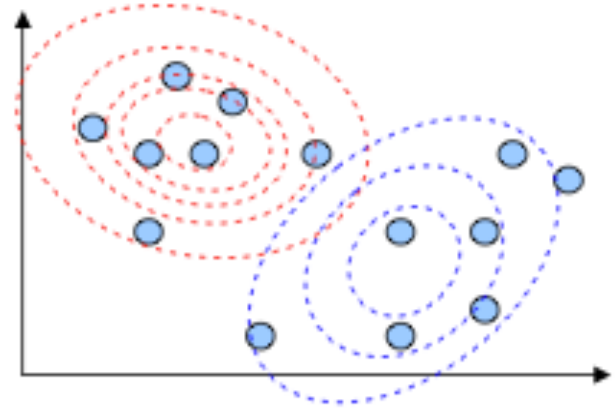
Gaussian mixture models: d>1



- Data with d attributes, from k sources
- Each source c is a Gaussian
- Iteratively estimate parameters:
  - prior: what % of instances came from source c?
    $$P(c) = \tfrac{1}{n}\sum_{i=1}^{n} P(c \mid \vec{x}_i)$$
  - mean: expected value of attribute j from source c
    $$\mu_{c,j} = \sum_{i=1}^{n} \left(\frac{P(c|\vec{x}_i)}{nP(c)}\right) x_{i,j}$$

Gaussian mixture models: d>1

- Data with d attributes, from k sources
- Each source c is a Gaussian
- Iteratively estimate parameters:
  - prior: what % of instances came from source c?
  - mean: expected value of attribute j from source c
  - covariance: how correlated are attributes j and k in source c?



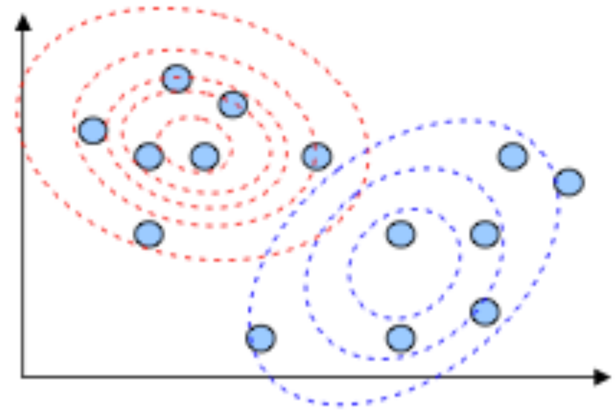
$$P(c) = \frac{1}{n}\sum_{i=1}^{n} P(c \mid \vec{x}_i)$$

$$\mu_{c,j} = \sum_{i=1}^{n} \left(\frac{P(c \mid \vec{x}_i)}{nP(c)}\right) x_{i,j}$$

$$(\Sigma_c)_{j,k} = \sum_{i=1}^{n} \left(\frac{P(c \mid \vec{x}_i)}{nP(c)}\right) (x_{i,j} - \mu_{c,j})(x_{i,k} - \mu_{c,k})$$

Gaussian mixture models: d>1

- Data with d attributes, from k sources
- Each source c is a Gaussian
- Iteratively estimate parameters:
  - prior: what % of instances came from source c?
    $$P(c) = \tfrac{1}{n}\sum_{i=1}^{n} P(c \mid \vec{x}_i)$$
  - mean: expected value of attribute j from source c
    $$\mu_{c,j} = \sum_{i=1}^{n}\left(\frac{P(c|\vec{x}_i)}{nP(c)}\right) x_{i,j}$$
  - covariance: how correlated are attributes j and k in source c?
    $$(\Sigma_c)_{j,k} = \sum_{i=1}^{n}\left(\frac{P(c|\vec{x}_i)}{nP(c)}\right)\left(x_{i,j} - \mu_{c,j}\right)\left(x_{i,k} - \mu_{c,k}\right)$$
  - based on: our guess of the source for each instance



$$P(c \mid \vec{x}_i) = \frac{P(\vec{x}_i \mid c)P(c)}{\sum_{c'=1}^{k} P(\vec{x}_i \mid c')P(c')}$$

$$P(\vec{x}_i \mid c) = \frac{1}{\sqrt{2\pi|\Sigma_c|}}\exp\left(-\frac{1}{2}\underbrace{(\vec{x}_i - \vec{\mu}_c)^T \Sigma_c^{-1}(\vec{x}_i - \vec{\mu}_c)}_{\sum_{j=1}^{d}\sum_{k=1}^{d}(x_{i,j}-\mu_{c,j})(\Sigma_c^{-1})_{j,k}(x_{i,k}-\mu_{c,k})}\right)$$

How to pick K?

- Probabilistic model $$L = \log P(x_1 \ldots x_n) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} P(x_i \mid k) P(k)$$
  - tries to "fit" the data (maximize likelihood)

How to pick K?

- Probabilistic model

$$L = \log P(x_1 \ldots x_n) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} P(x_i \mid k) P(k)$$

  - tries to “fit” the data (maximize likelihood)
- Pick K that makes *L* as large as possible?
  - *K* = *n*: each data point has its own “source”
  - may not work well for new data points

How to pick K?

- Probabilistic model

$$L = \log P(x_1 \ldots x_n) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} P(x_i \mid k) P(k)$$

  - tries to "fit" the data (maximize likelihood)
- Pick K that makes *L* as large as possible?
  - *K* = *n*: each data point has its own "source"
  - may not work well for new data points
- Split points into training set T and validation set V
  - for each *K*: fit parameters of T, measure likelihood of V
  - sometimes still best when *K* = *n*

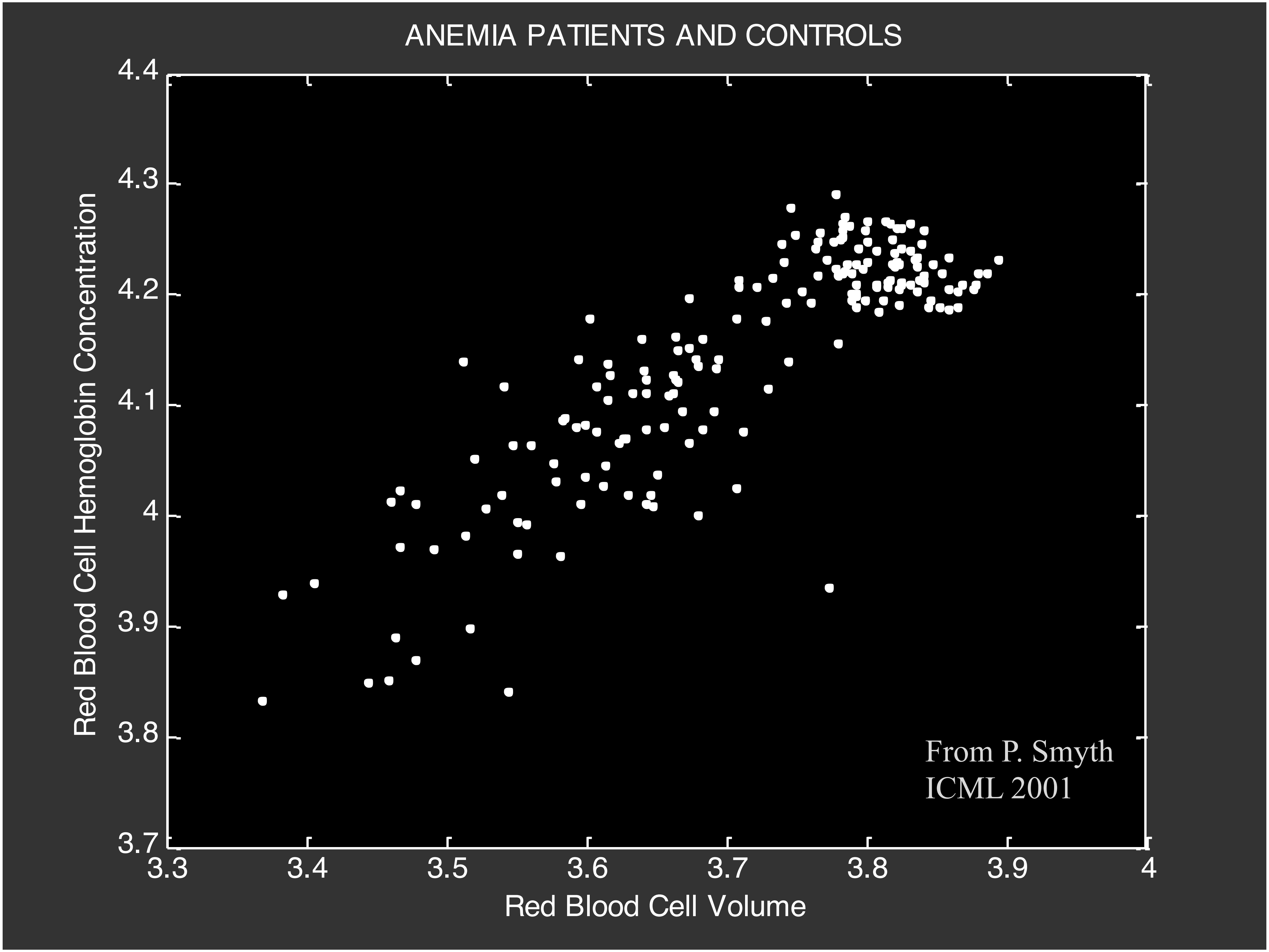
How to pick K?

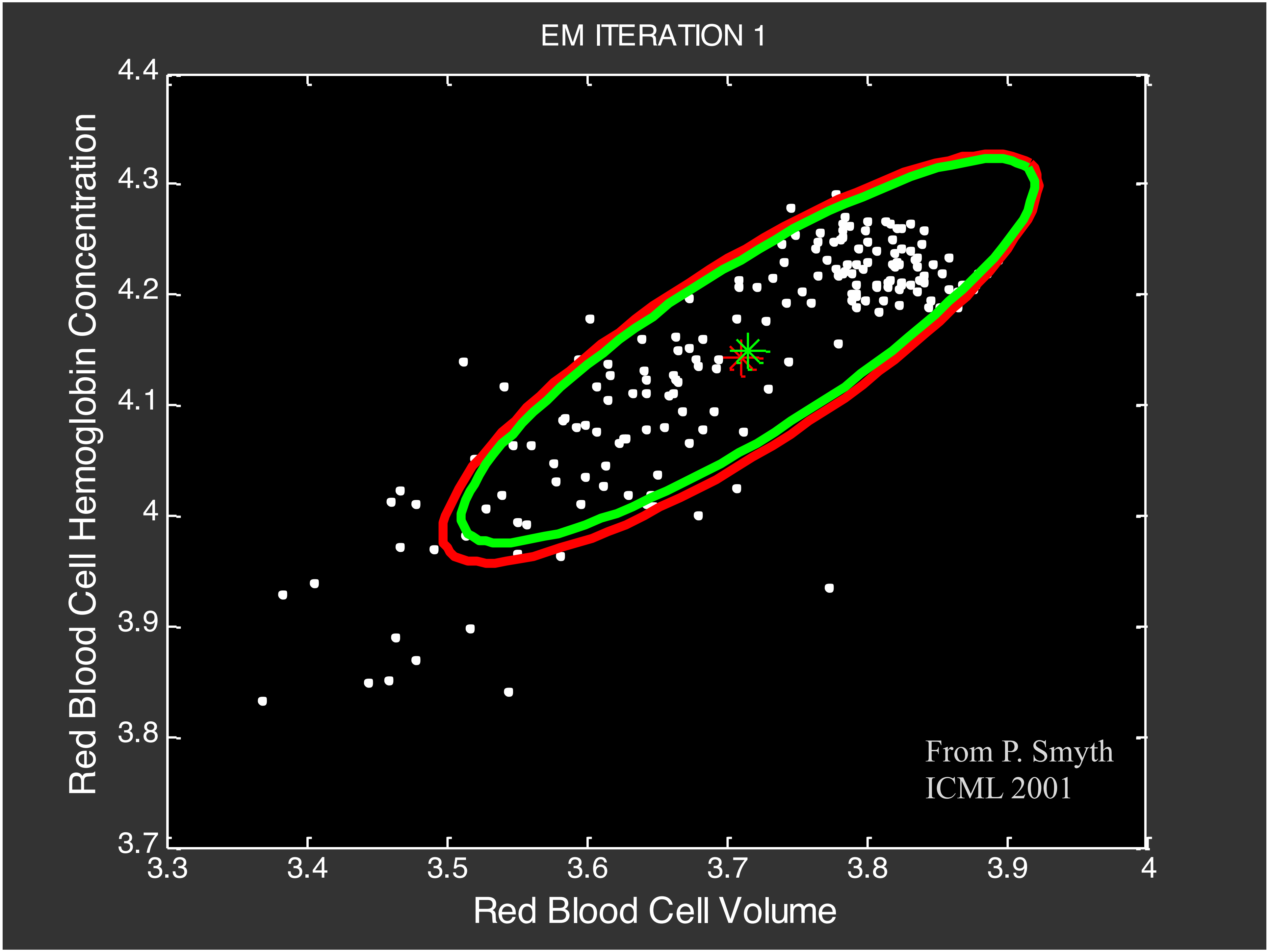
- Probabilistic model

$$L = \log P(x_1 \ldots x_n) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} P(x_i \mid k) P(k)$$

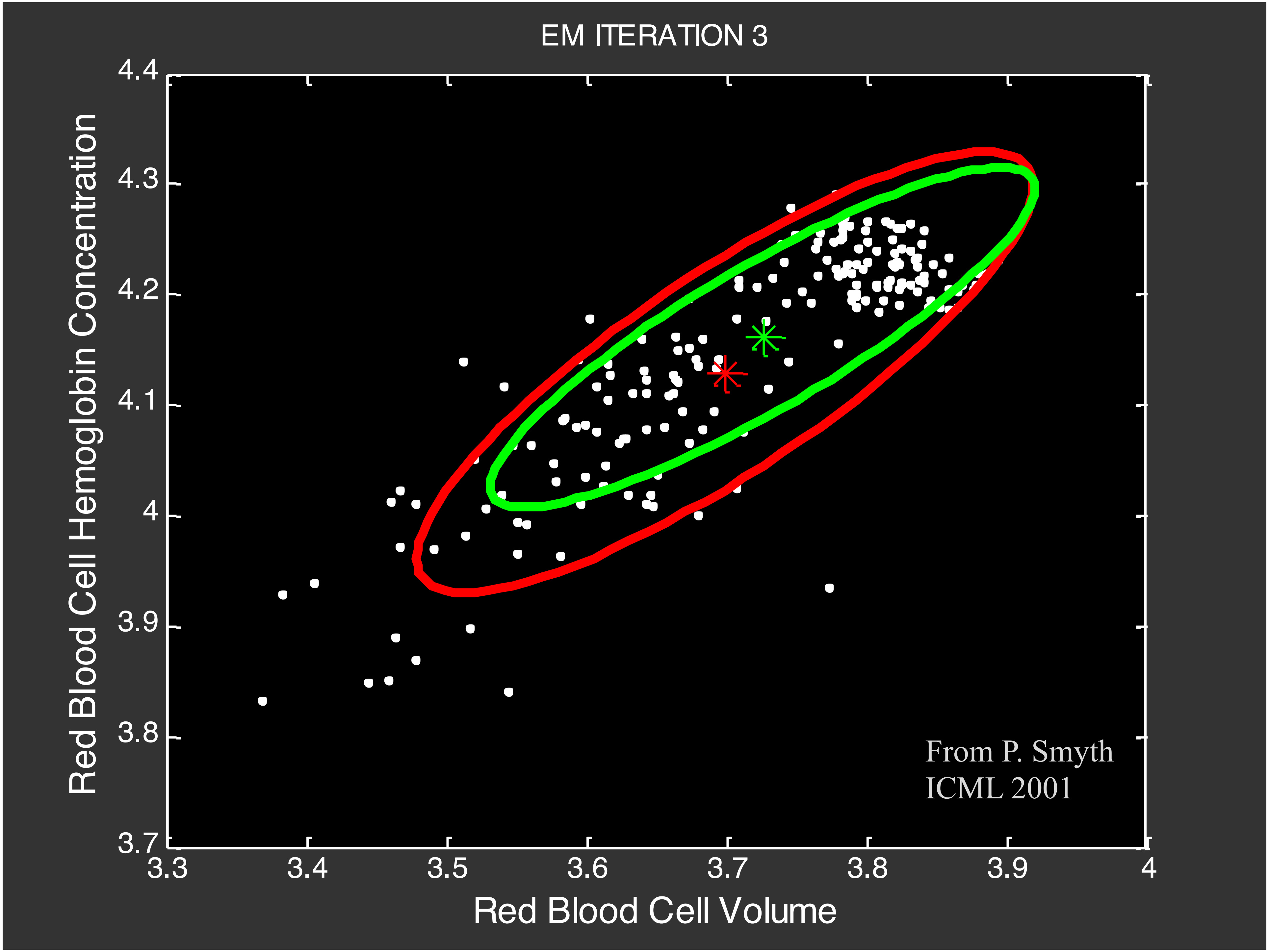
  - tries to “fit” the data (maximize likelihood)
- Pick K that makes *L* as large as possible?
  - *K* = *n*: each data point has its own “source”
  - may not work well for new data points
- Split points into training set T and validation set V
  - for each *K*: fit parameters of T, measure likelihood of V
  - sometimes still best when *K* = *n*
- Occam’s razor: pick “simplest” of all models that fit
  - Bayes Inf. Criterion (BIC): $\max_p \{ L - \frac{1}{2} p \log n \}$
  - Akaike Inf. Criterion (AIC): $\min_p \{ 2p - L \}$

L … **likelihood**, how well our model fits the data ***p*** … **number of parameters** how “simple” is the model

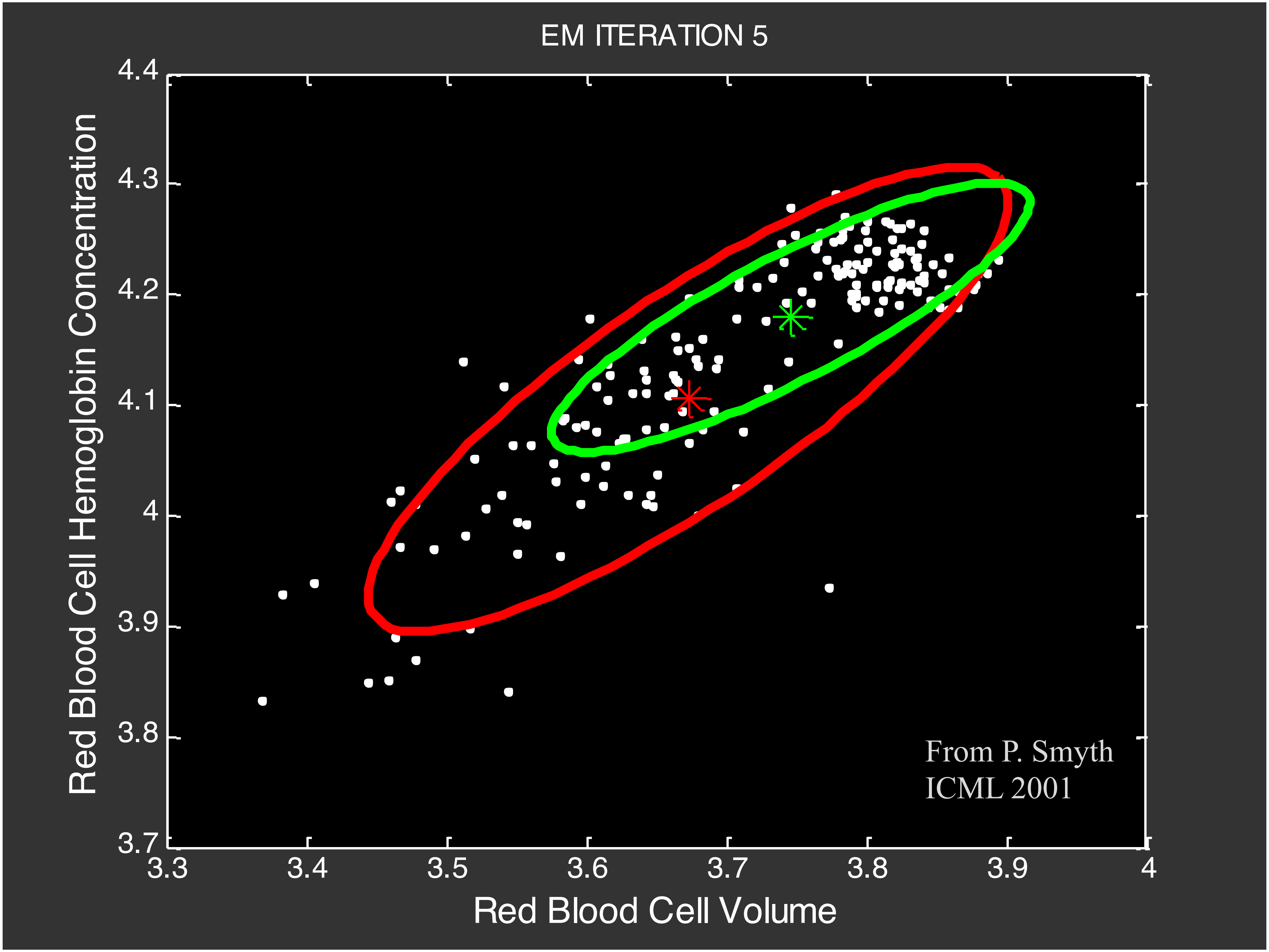

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Volume
From P. Smyth
ICML 2001



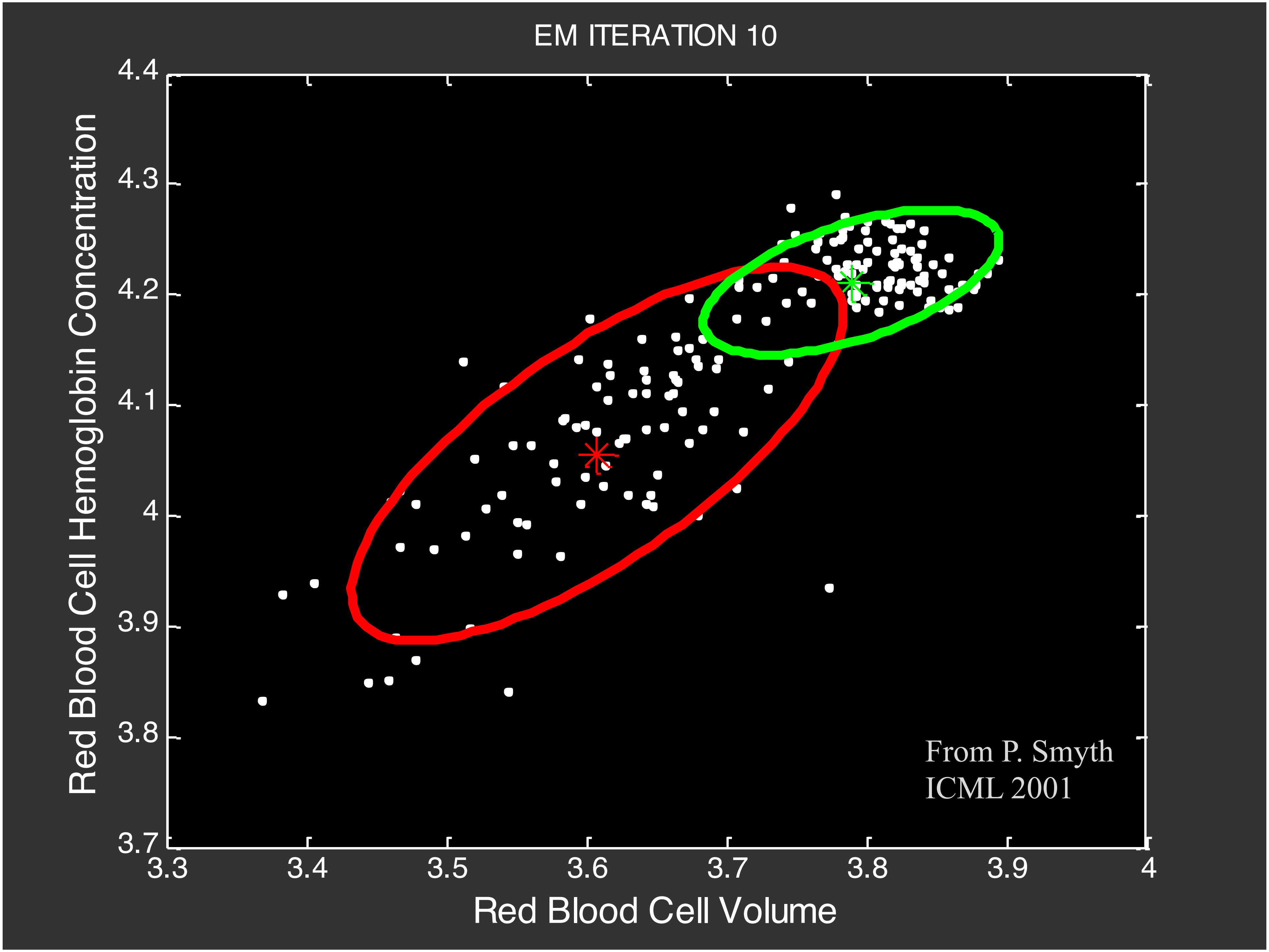
EM ITERATION 1
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001



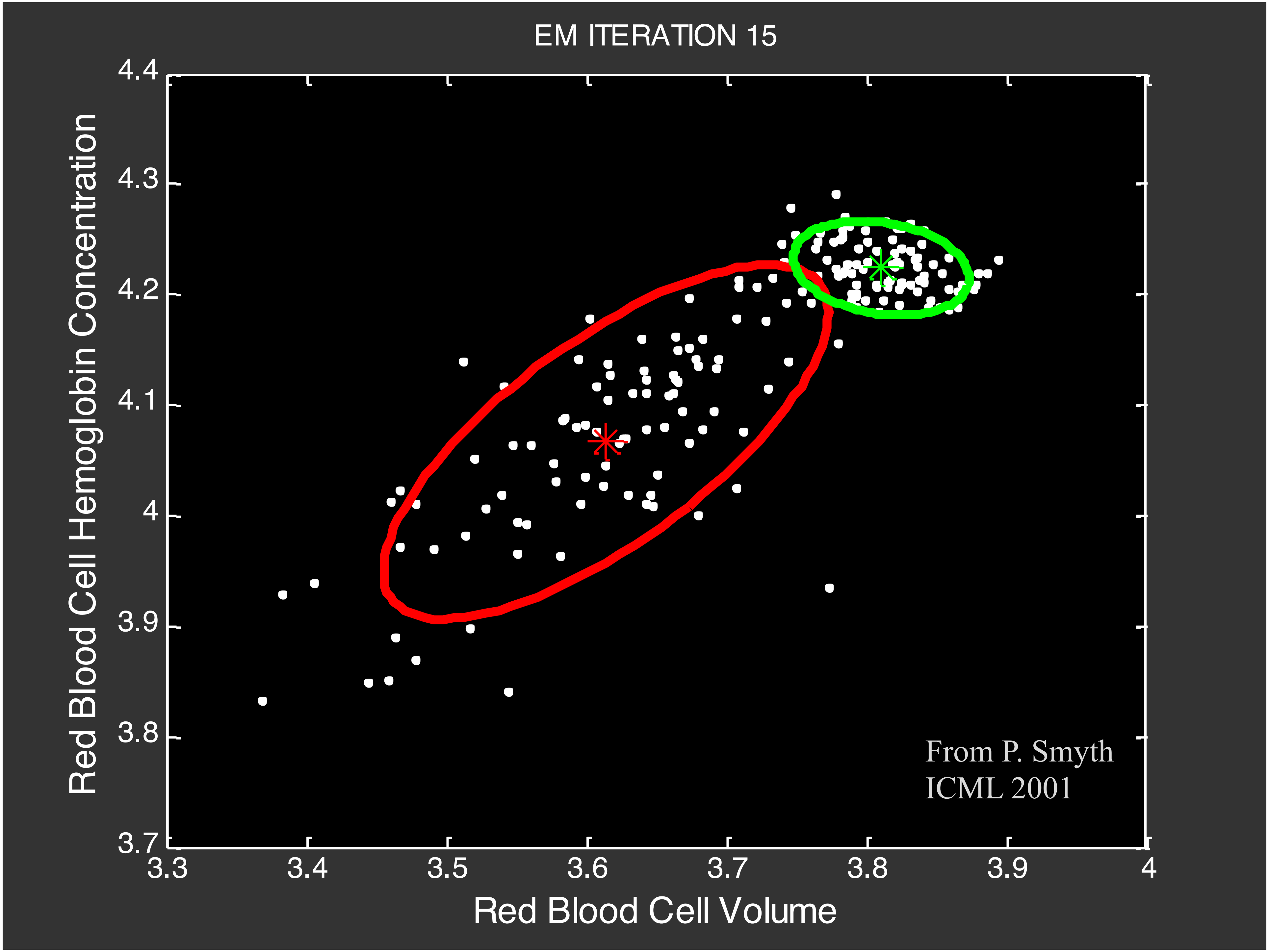
EM ITERATION 3
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
Red Blood Cell Hemoglobin Concentration
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Volume
From P. Smyth
ICML 2001



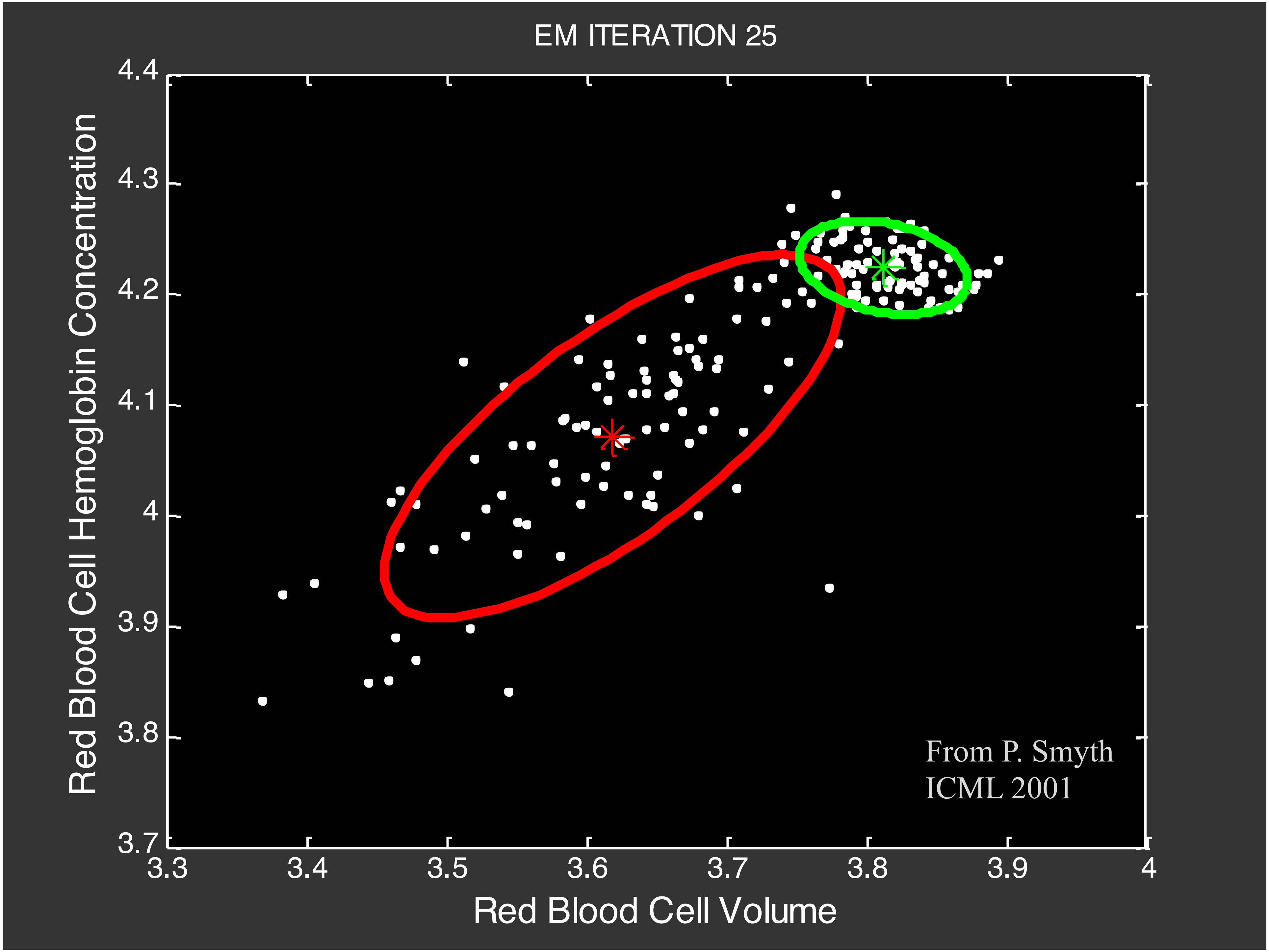
EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001

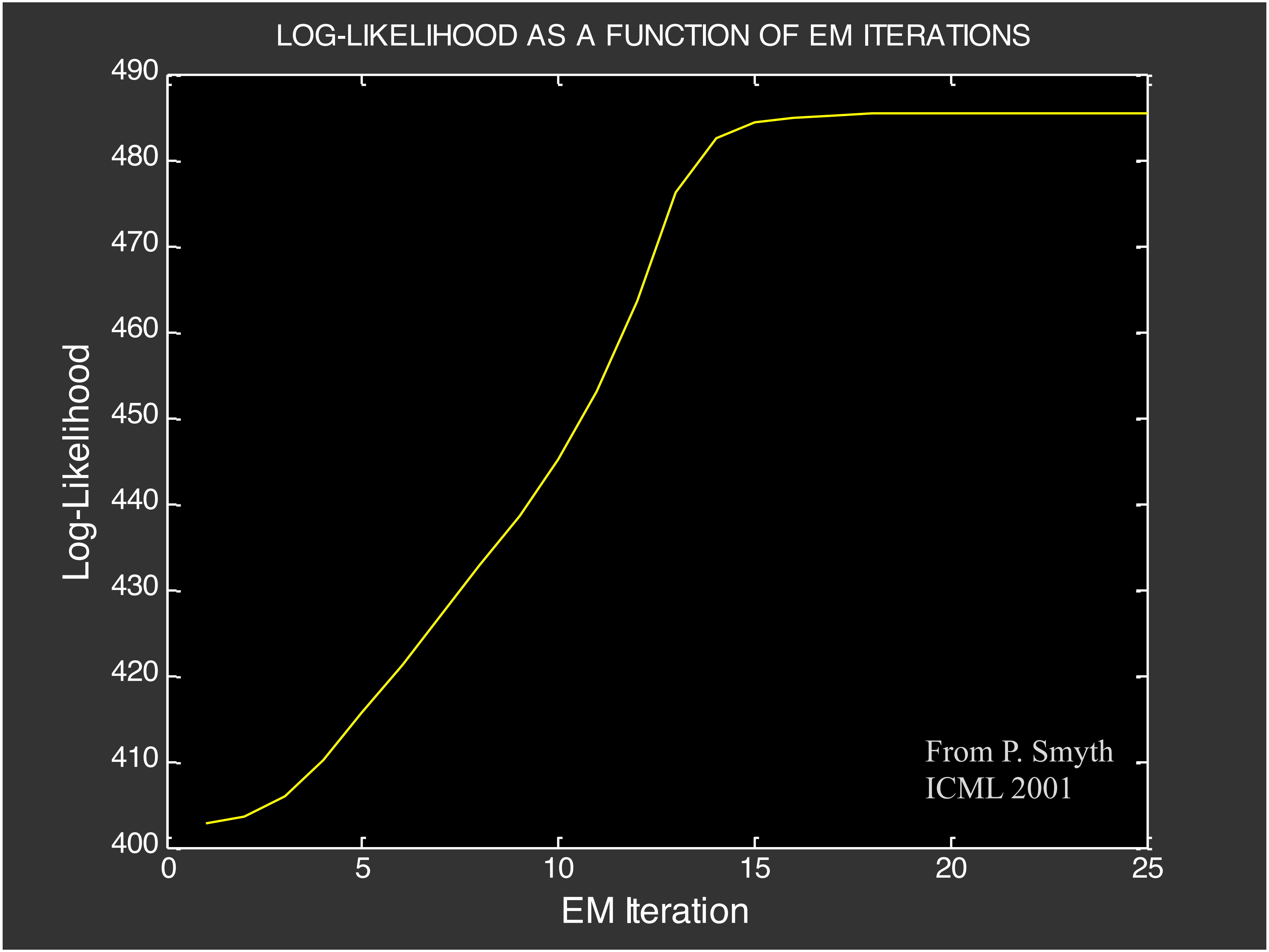


EM ITERATION 10
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
From P. Smyth
ICML 2001



EM ITERATION 15
Red Blood Cell Hemoglobin Concentration
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Volume
From P. Smyth
ICML 2001


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001



LOG-LIKELIHOOD AS A FUNCTION OF EM ITERATIONS
Log-Likelihood
490
480
470
460
450
440
430
420
410
400
0
5
10
15
20
25
EM Iteration
From P. Smyth
ICML 2001