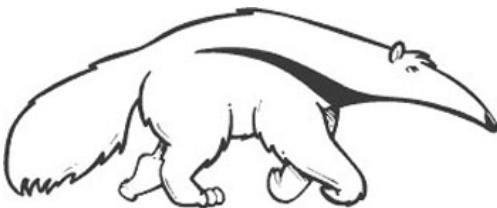


+

Machine Learning and Data Mining

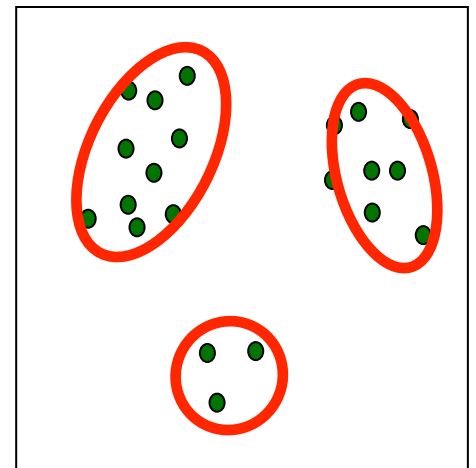
Clustering (1): Basics

Prof. Alexander Ihler



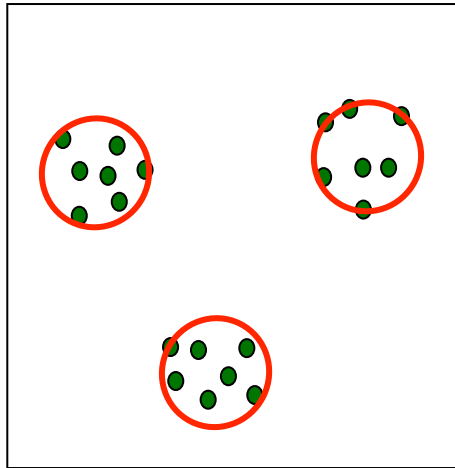
Unsupervised learning

- Supervised learning
 - Predict target value (“y”) given features (“x”)
- Unsupervised learning
 - Understand patterns of data (just “x”)
 - Useful for many reasons
 - Data mining (“explain”)
 - Missing data values (“impute”)
 - Representation (feature generation or selection)
- One example: *clustering*
 - Describe data by discrete “groups” with some characteristics

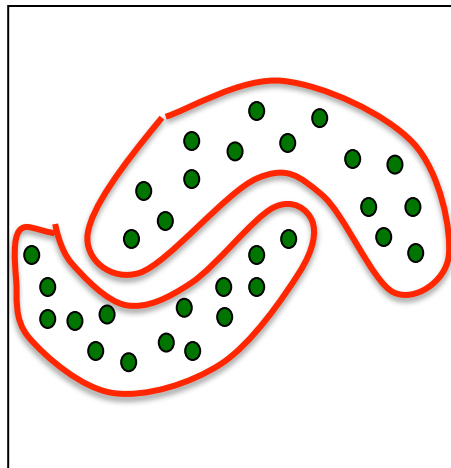


Clustering

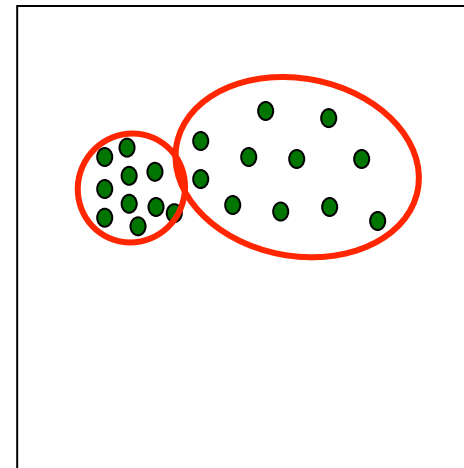
- Clustering describes data by “groups”
- The meaning of “groups” may vary by data!
- Examples



Location



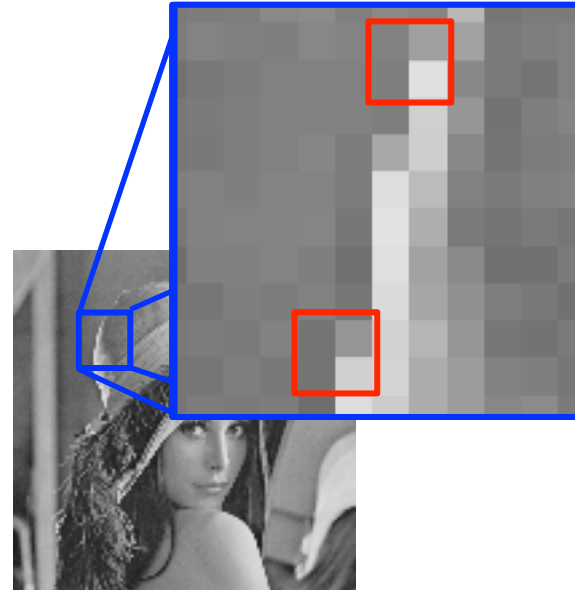
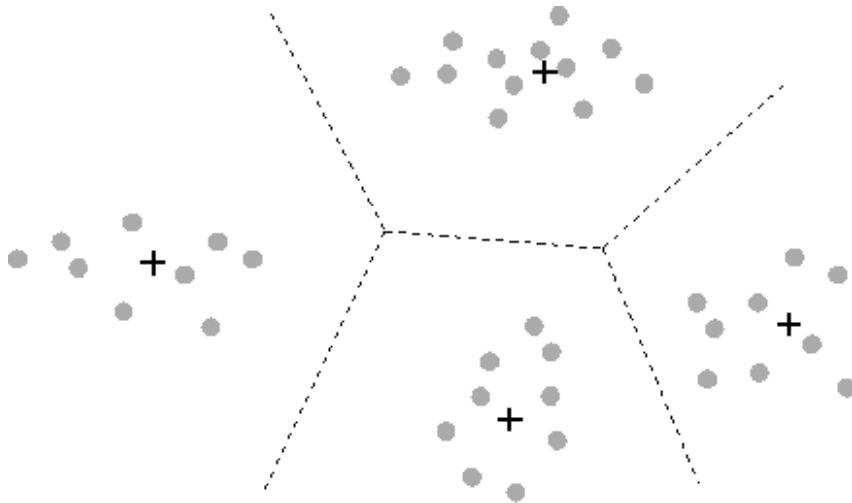
Shape



Density

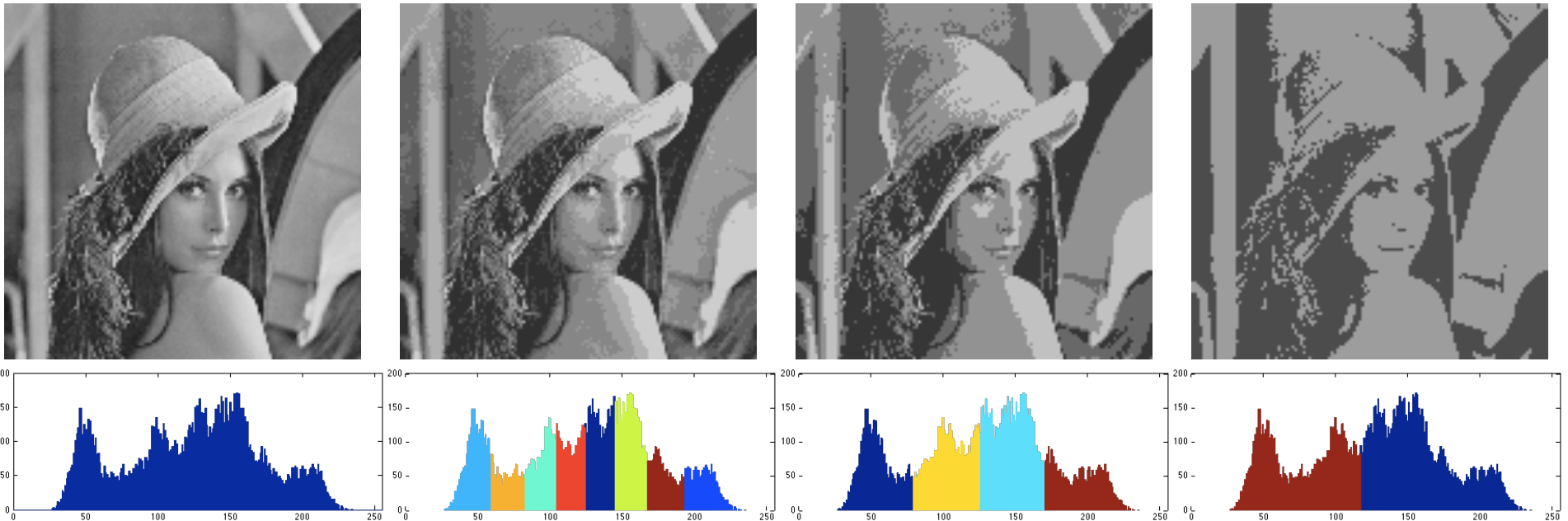
Clustering and Data Compression

- Clustering is related to vector quantization
 - Dictionary of vectors (the cluster centers)
 - Each original value represented using a dictionary index
 - Each center “claims” a nearby region (Voronoi region)



Clustering and Data Compression

- Clustering is related to vector quantization
 - Dictionary of vectors (the cluster centers)
 - Each original value represented using a dictionary index
 - Each center “claims” a nearby region (Voronoi region)
- Example in 1D: cluster pixels' grayscale values



+

Machine Learning and Data Mining

Clustering (2): Hierarchical Agglomerative Clustering

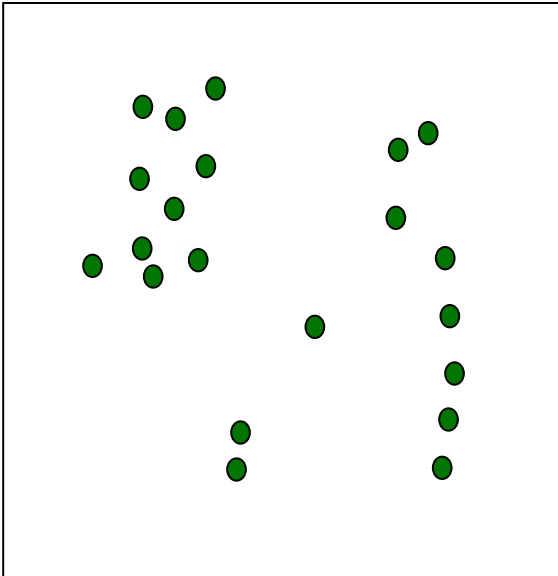
Prof. Alexander Ihler



Hierarchical Agglomerative Clustering

Initially, every datum is a cluster

Data:



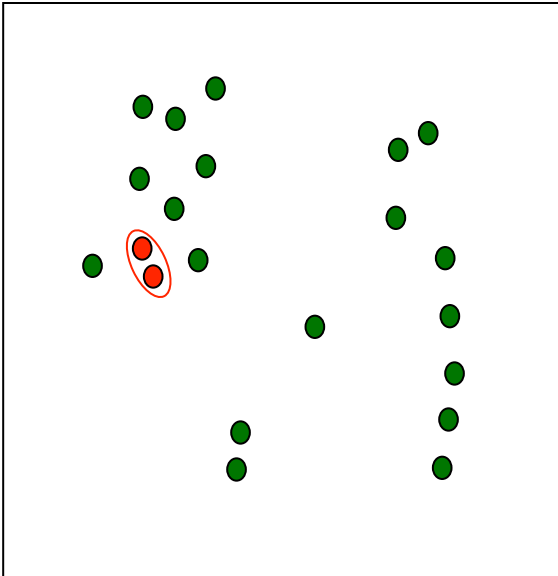
- A simple clustering algorithm
- Define a distance (or dissimilarity) between clusters (we'll return to this)
- Initialize: every example is a cluster
- Iterate:
 - Compute distances between all clusters (store for efficiency)
 - Merge two closest clusters
- Save both clustering and *sequence* of cluster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

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

Iteration 1

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



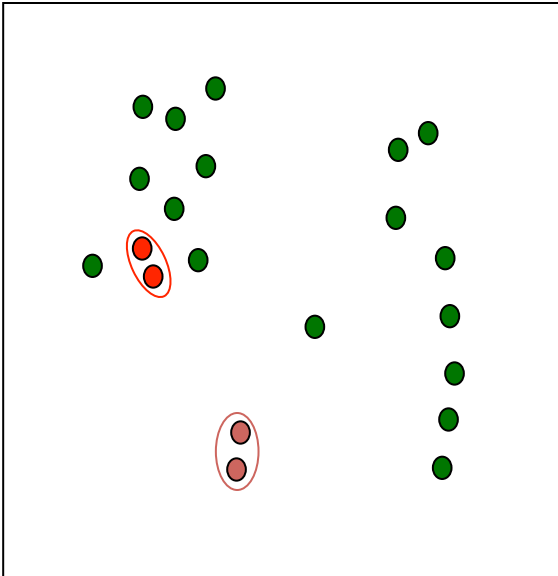
Height of the join
indicates dissimilarity

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

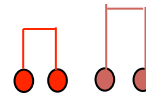
Iteration 2

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



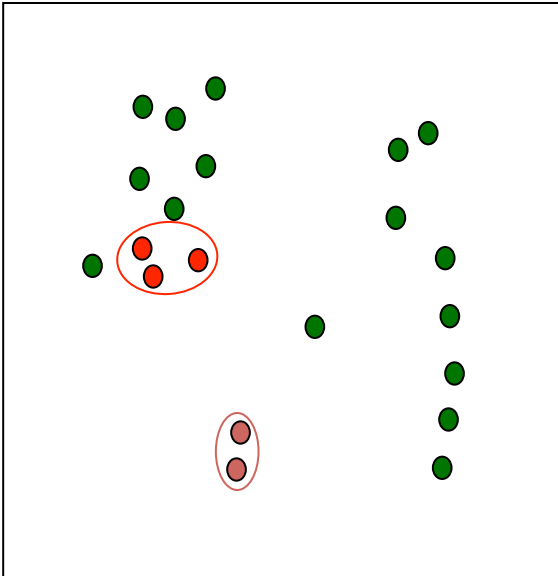
Height of the join
indicates dissimilarity

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

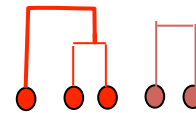
Iteration 3

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



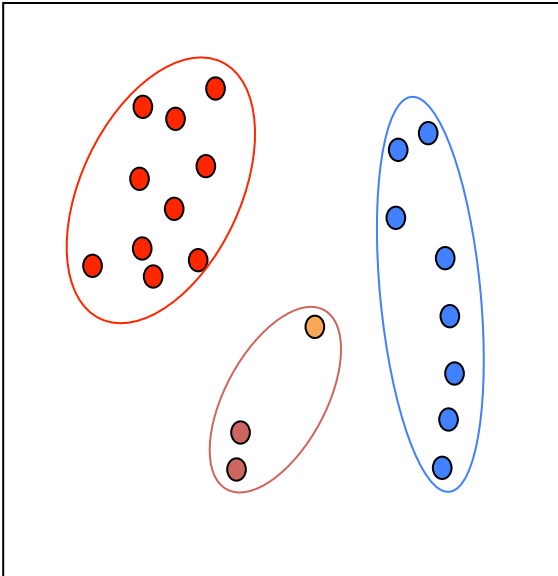
Height of the join
indicates dissimilarity

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

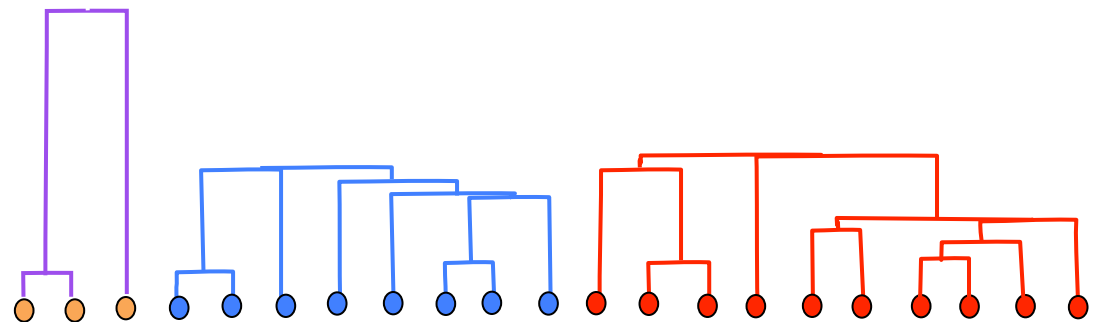
Iteration m-3

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



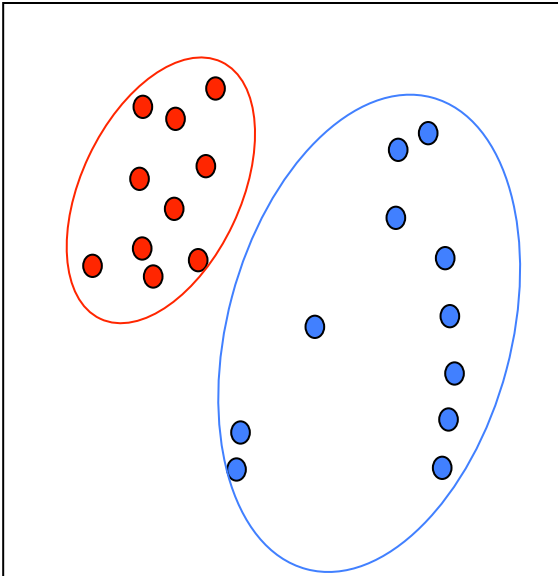
In matlab: “linkage” function (stats toolbox)

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

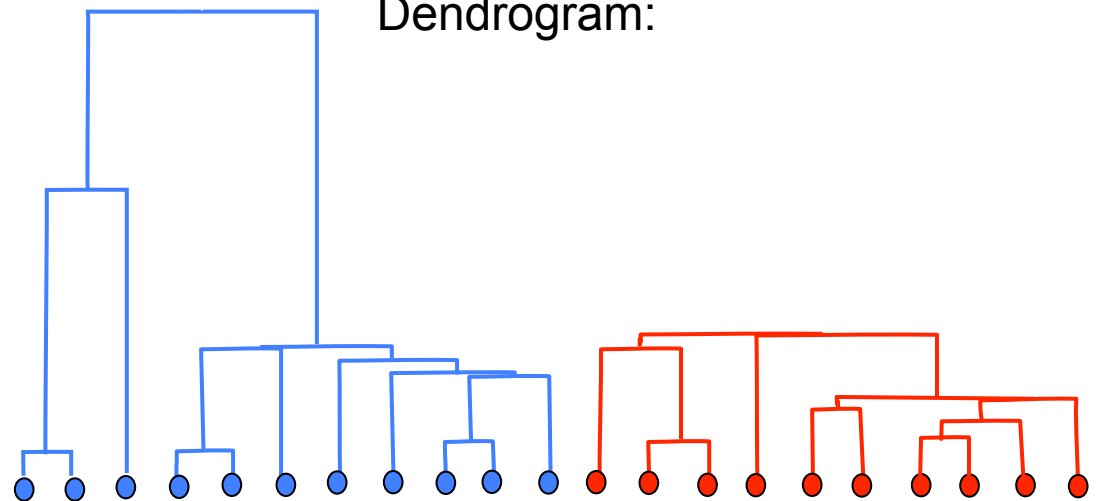
Iteration m-2

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



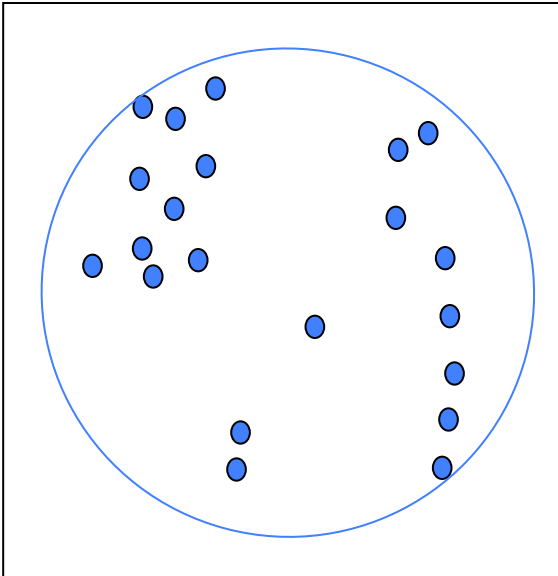
In matlab: “linkage” function (stats toolbox)

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

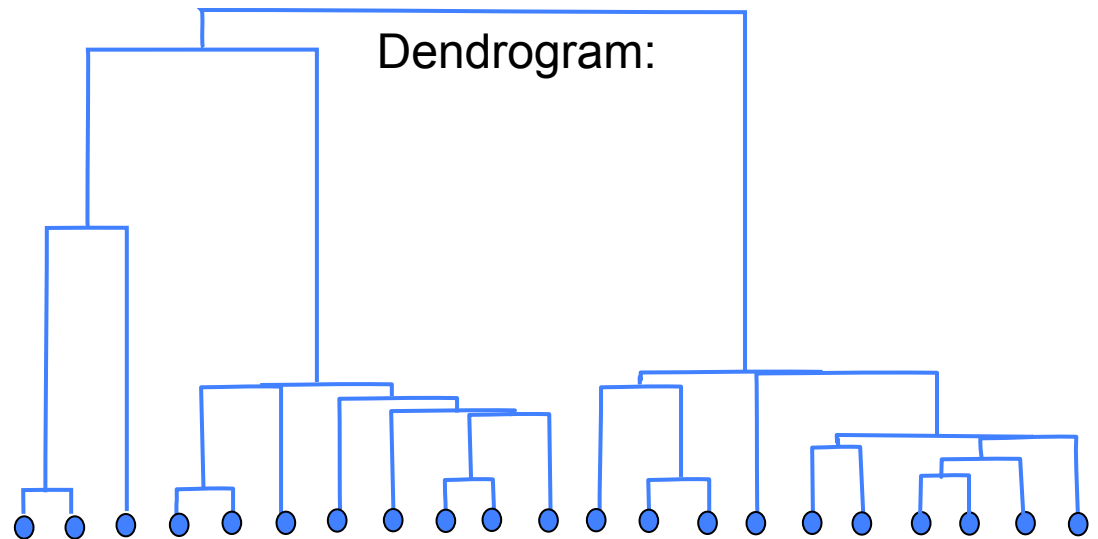
Iteration m-1

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



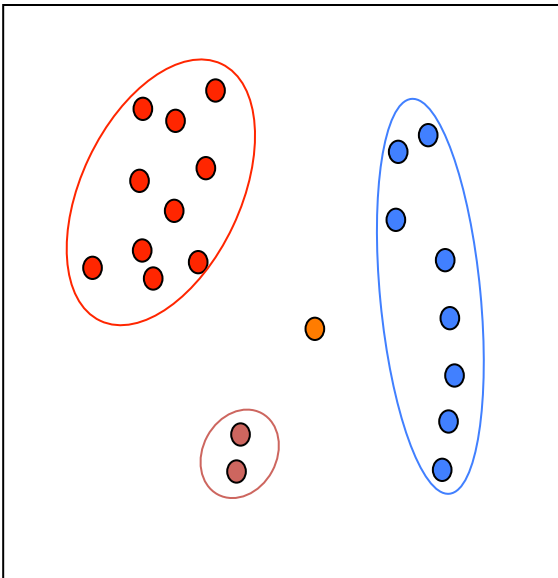
In matlab: “linkage” function (stats toolbox)

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

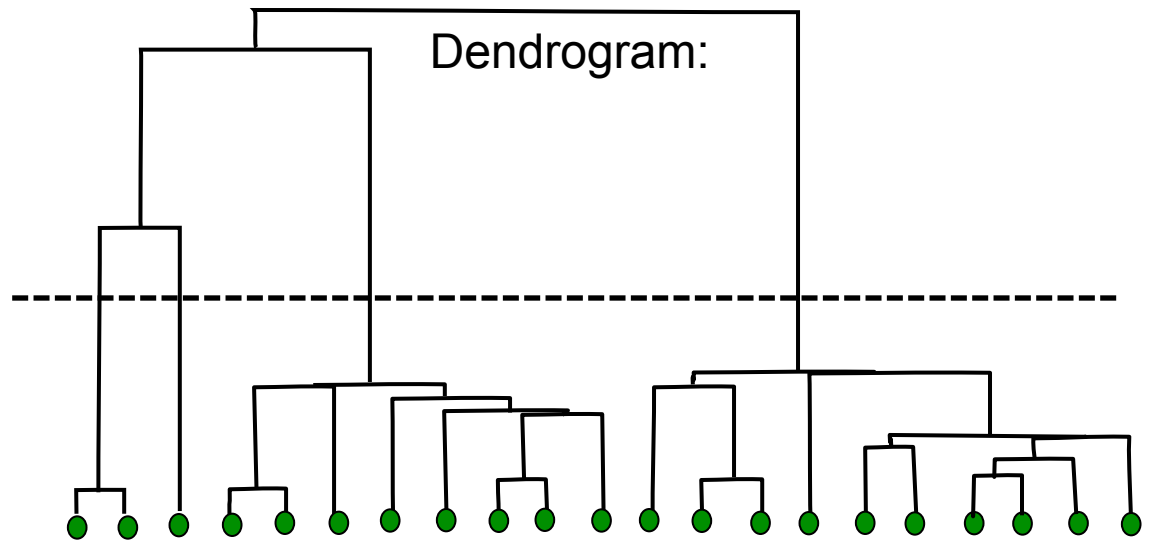
From dendrogram to clusters

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

Data:



Dendrogram:

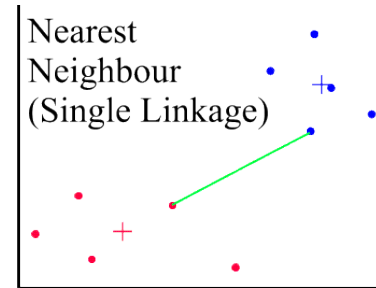


In matlab: “linkage” function (stats toolbox)

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

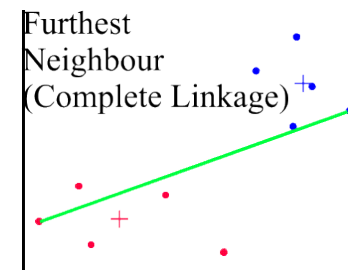
Cluster distances

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



produces minimal spanning tree.

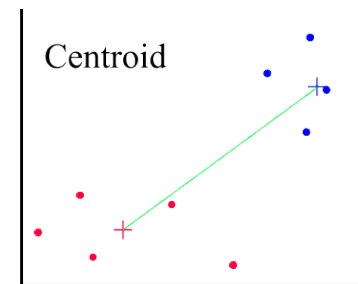
$$D_{\max}(C_i, C_j) = \max_{x \in C_i, y \in C_j} \|x - y\|^2$$



avoids elongated clusters.

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

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

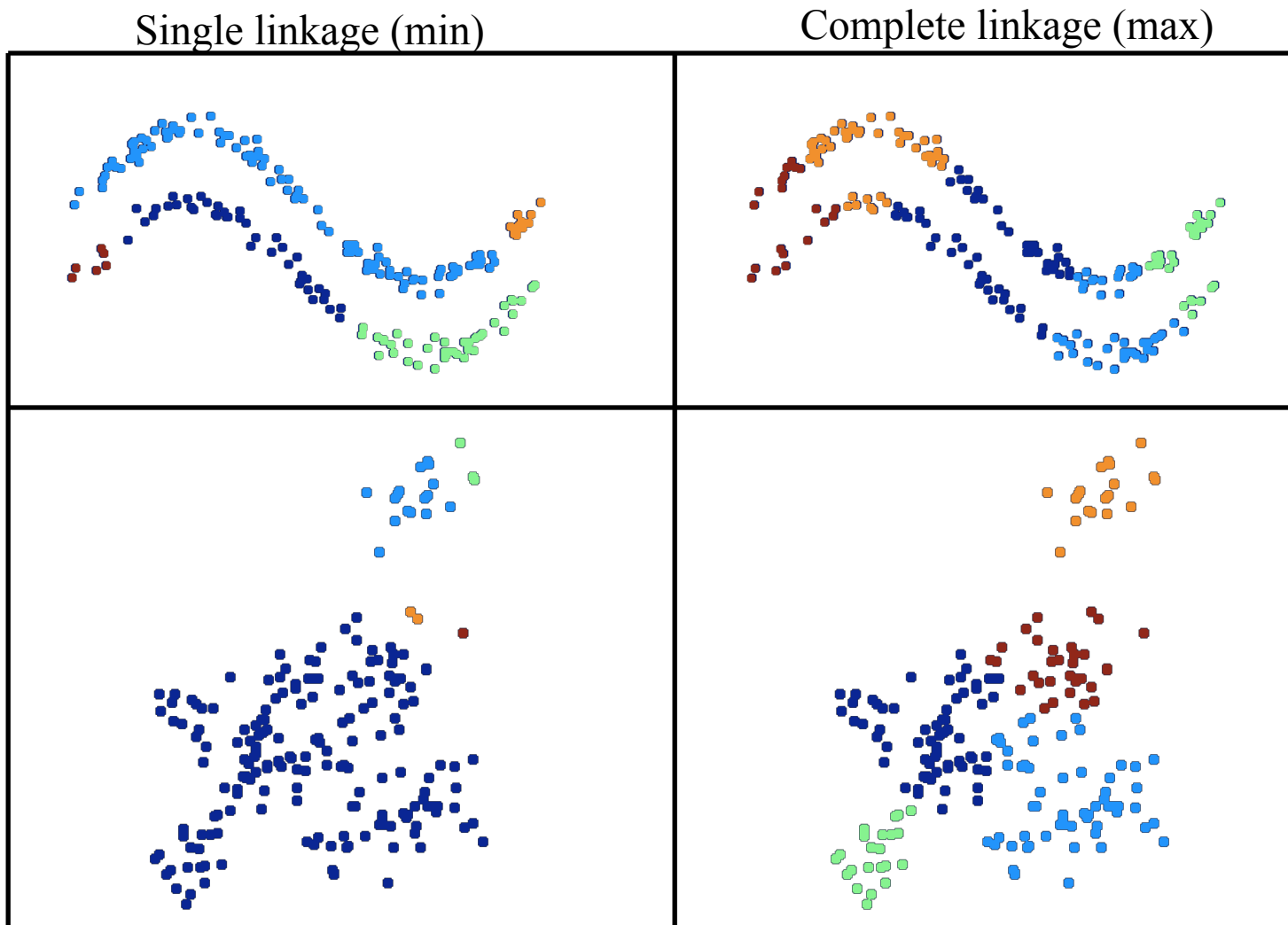


Need:

$$\begin{aligned} D(A, C) &\rightarrow D(A+B, C) \\ D(B, C) &\rightarrow D(A+B, C) \end{aligned}$$

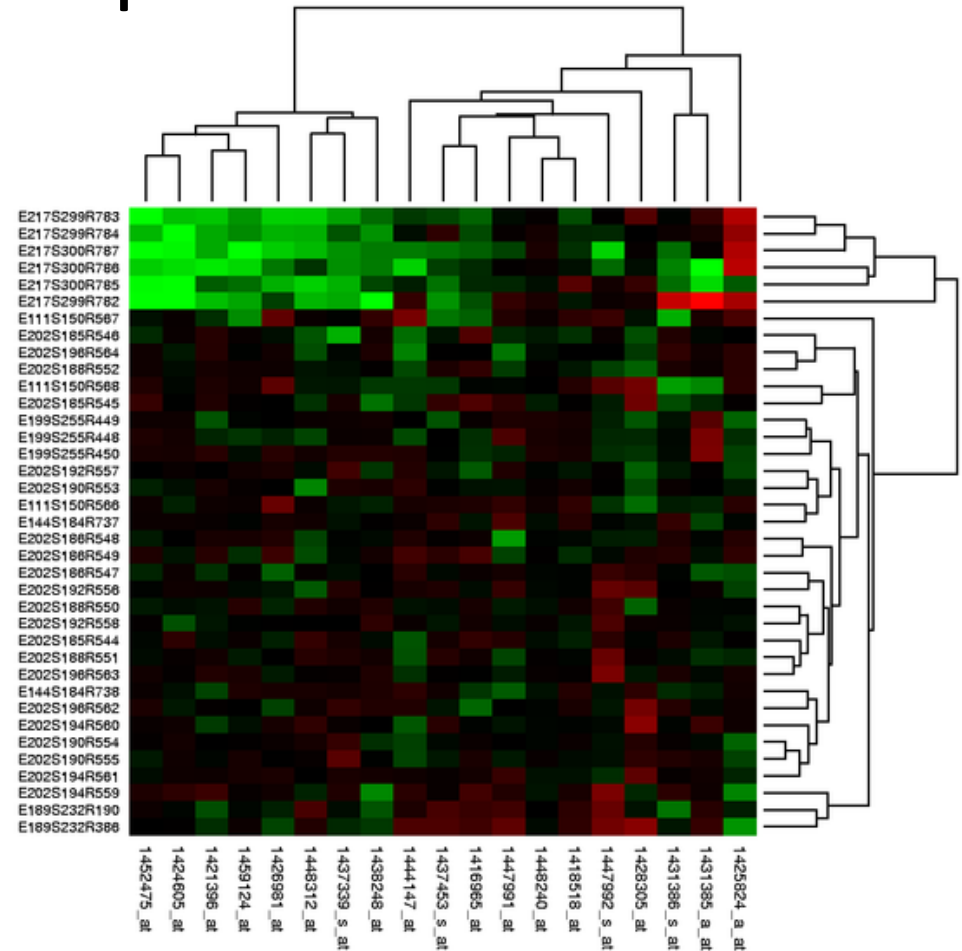
Cluster distances

- Dissimilarity choice will affect clusters created



Example: microarray expression

- Measure gene expression
- Various experimental conditions
 - Disease v. normal
 - Time
 - Subjects
- Explore similarities
 - What genes change together?
 - What conditions are similar?
- Cluster on both genes and conditions



Matlab: “clustergram” (bioinfo toolbox)

Summary

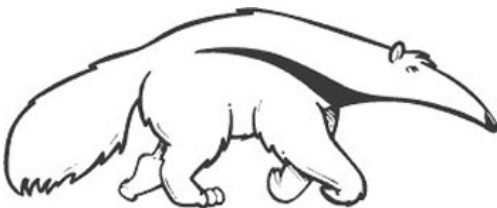
- Agglomerative clustering
 - Choose a cluster distance / dissimilarity scoring method
 - Successively merge closest pair of clusters
 - “Dendrogram” shows sequence of merges & distances
 - Complexity: $O(m^2 \log m)$
- “Clustergram” for understanding data matrix
 - Build clusters on rows (data) and columns (features)
 - Reorder data & features to expose behavior across groups
- Agglomerative clusters depend critically on dissimilarity
 - Choice determines characteristics of “found” clusters

+

Machine Learning and Data Mining

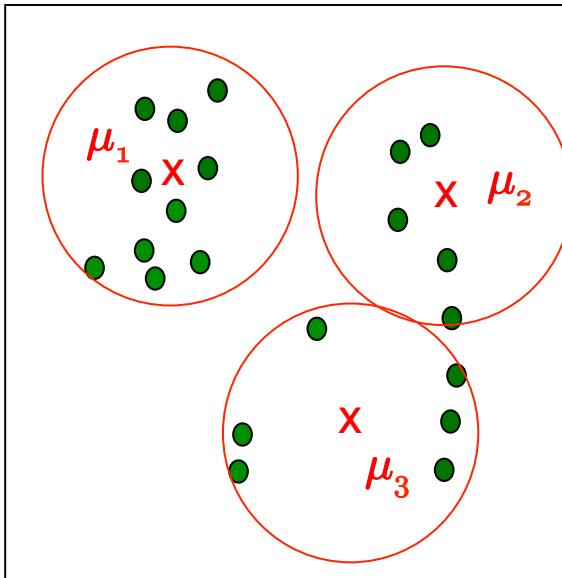
Clustering (3): k-Means Clustering

Prof. Alexander Ihler



K-Means Clustering

- A simple clustering algorithm
- Iterate between
 - Updating the assignment of data to clusters
 - Updating the cluster's summarization



Notation:

Data example i has features x_i

Assume K clusters

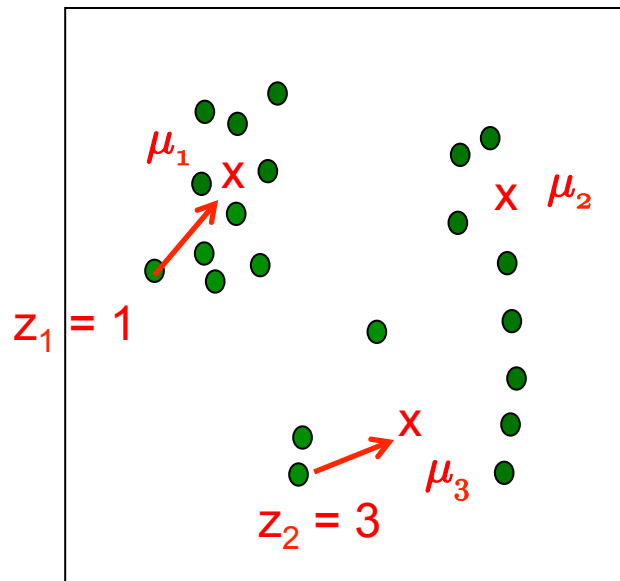
Each cluster c “described” by a center μ_c

Each cluster will “claim” a set of nearby points

Matlab: “kmeans” (stats toolbox)

K-Means Clustering

- A simple clustering algorithm
- Iterate between
 - Updating the assignment of data to clusters
 - Updating the cluster's summarization



Notation:

Data example i has features x_i

Assume K clusters

Each cluster c “described” by a center μ_c

Each cluster will “claim” a set of nearby points
“Assignment” of i^{th} example: $z_i \in 1..K$

Matlab: “kmeans” (stats toolbox)

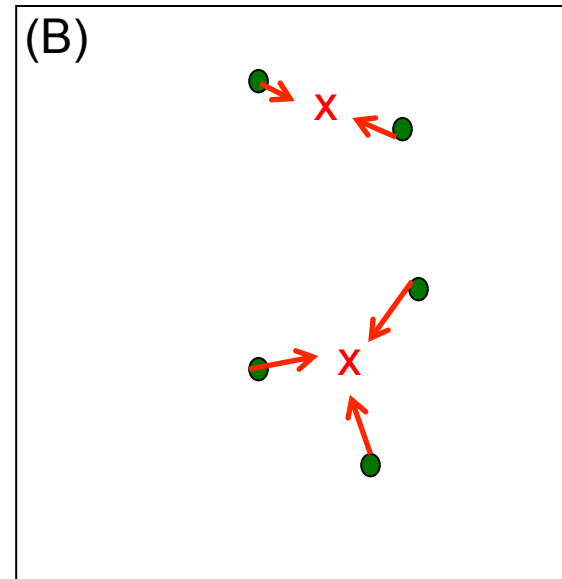
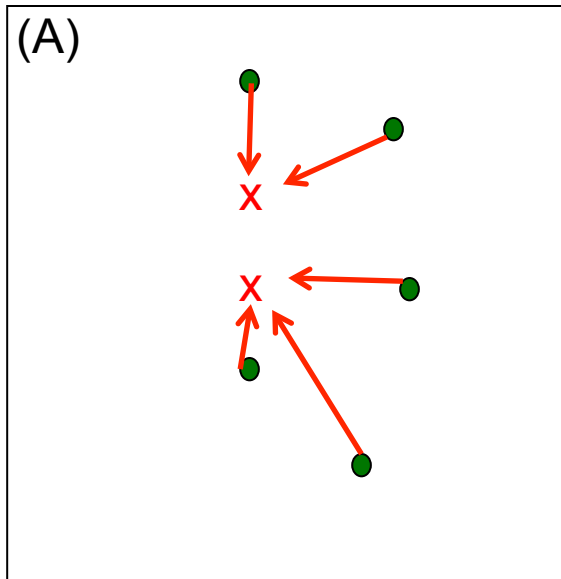
K-Means Clustering

- Iterate until convergence:
 - (A) For each datum, find the closest cluster

$$z_i = \arg \min_c \|x_i - \mu_c\|^2 \quad \forall i$$

- (B) Set each cluster to the mean of all assigned data:

$$\forall c, \quad \mu_c = \frac{1}{m_c} \sum_{i \in S_c} x_i \quad S_c = \{i : z_i = c\}, \quad m_c = |S_c|$$



K-Means Clustering

- Optimizing the cost function:

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

Descent => guaranteed to converge

New means = same assignments

Same assignments = same means

Same means = same assignments

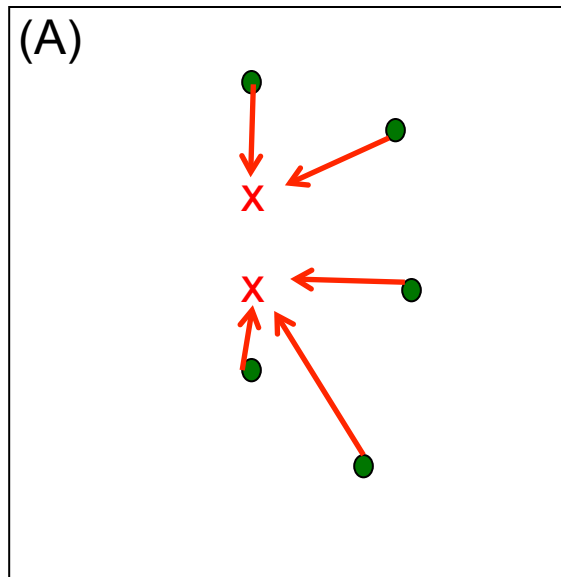
...

- Coordinate descent:

Over the cluster assignments:

Only one term in sum depends on z_i

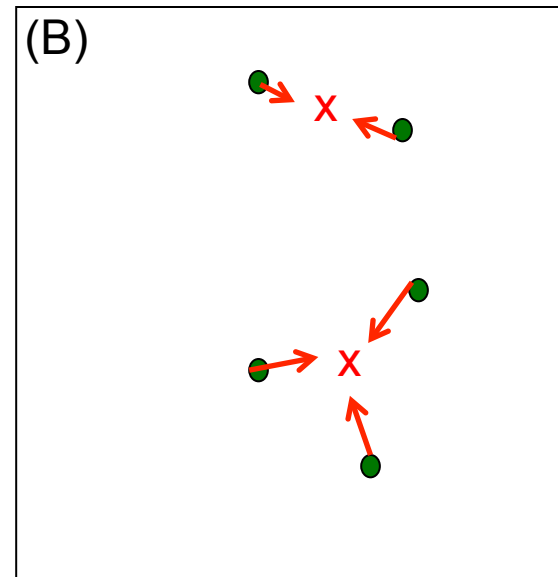
Minimized by selecting closest μ_c



Over the cluster centers:

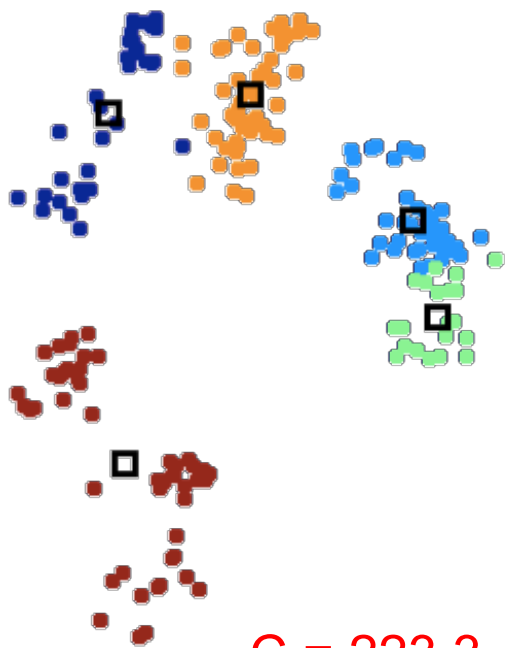
Cluster c only depends on x_i with $z_i=c$

Minimized by selecting the mean

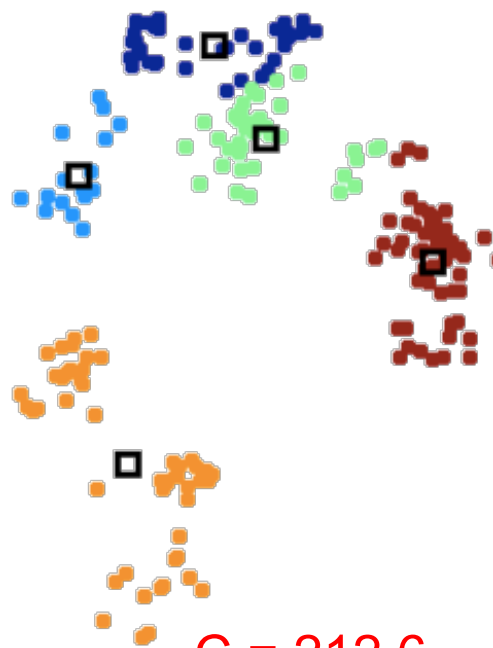


Initialization

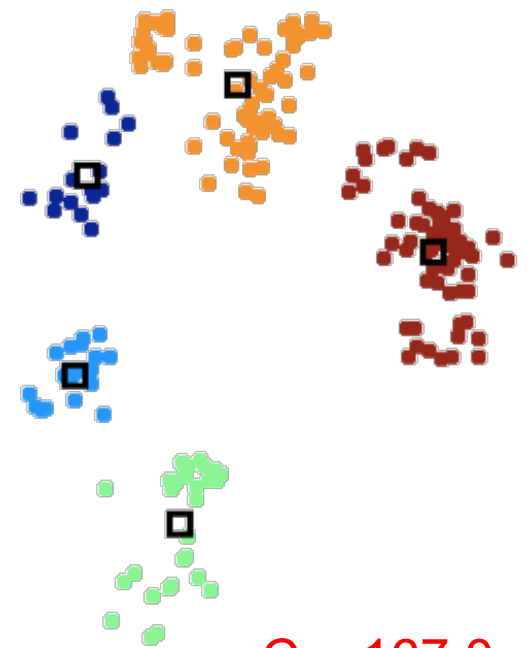
- Multiple local optima, depending on initialization
- Try different (randomized) initializations
- Can use cost C to decide which we prefer



$C = 223.3$



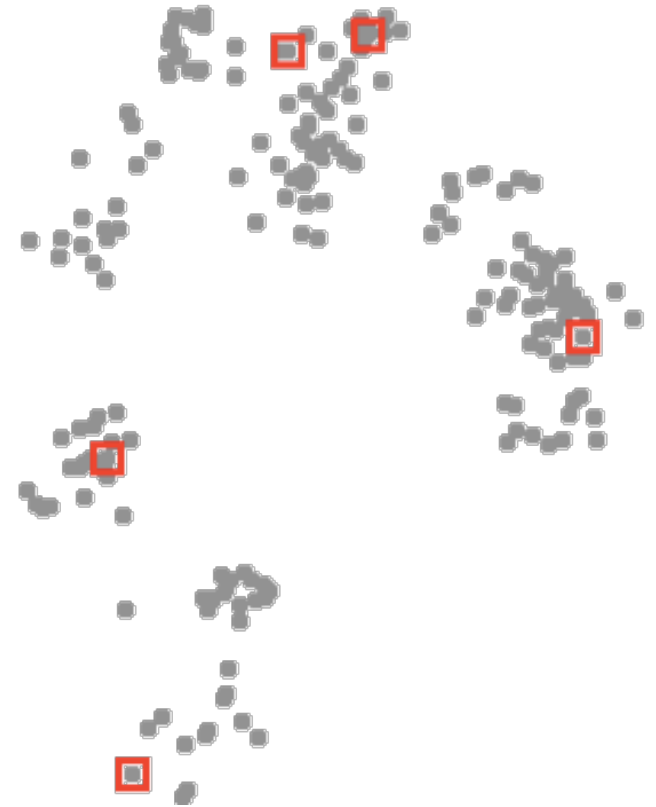
$C = 212.6$



$C = 167.0$

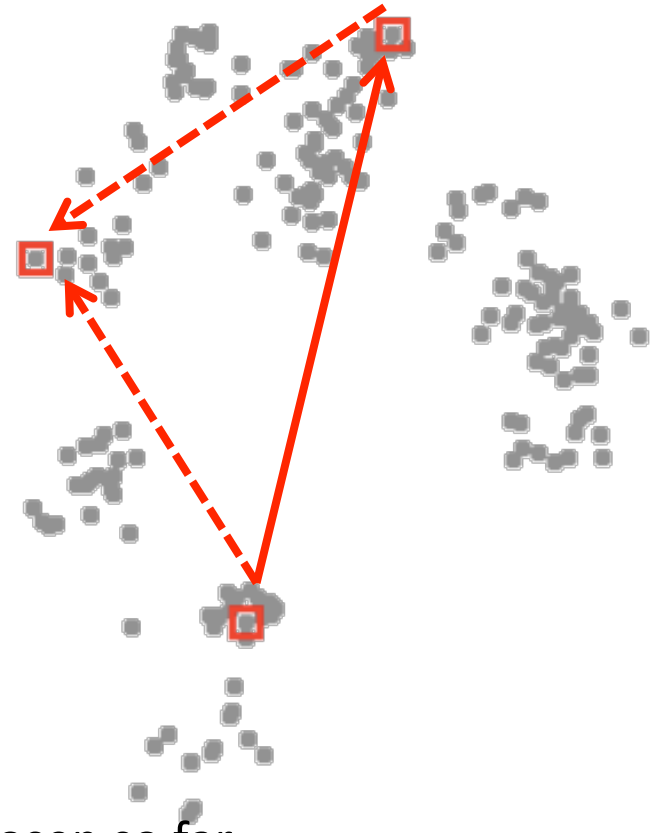
Initialization methods

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



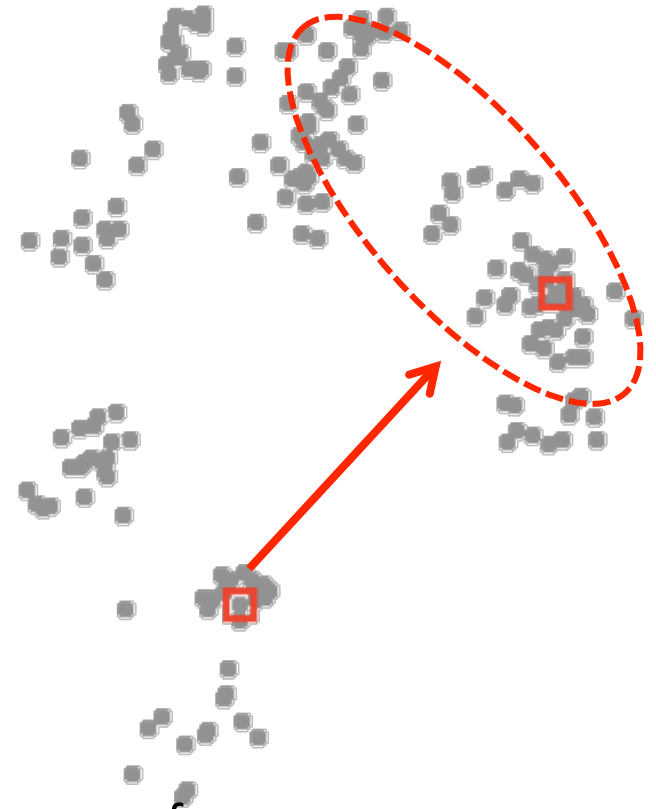
Initialization methods

- 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”
 - $p(x) \propto \text{squared distance from } x \text{ to current centers}$
 - 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
```

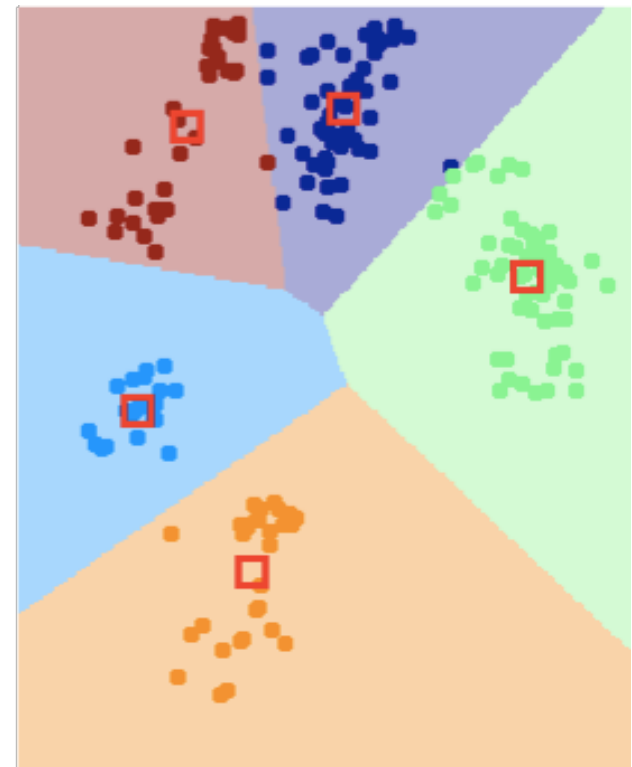
```
[Z , mu] = kmeans(X, K);
```

```
% cluster id = nearest center
```

```
L = knnClassify(mu, (1:K)', 1);
```

```
% assign in- or out-of-sample points
```

```
Z = predict(L, X);
```



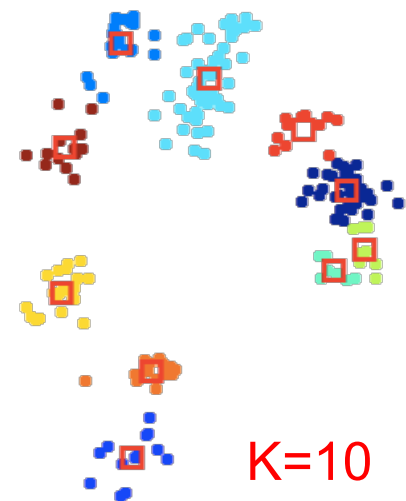
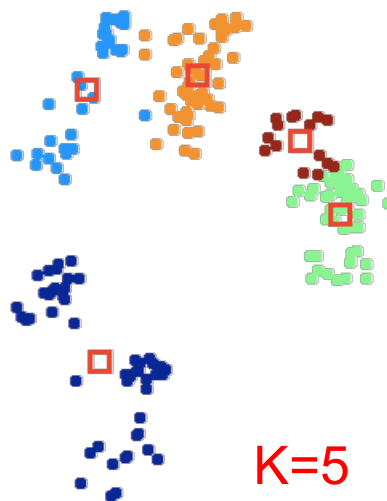
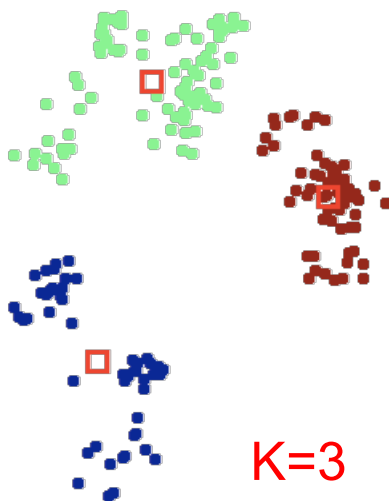
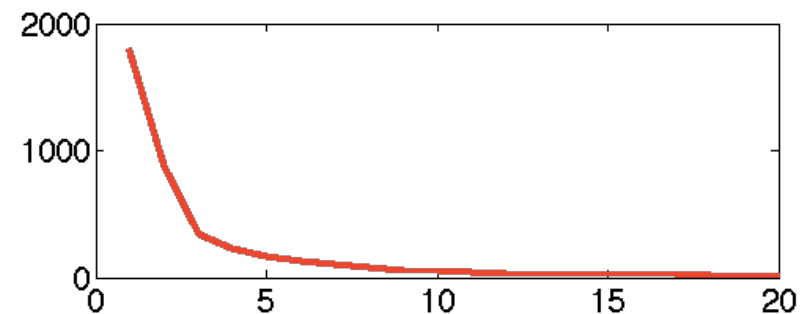
Choosing the number of clusters

- With cost function

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

what is the optimal value of k?

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



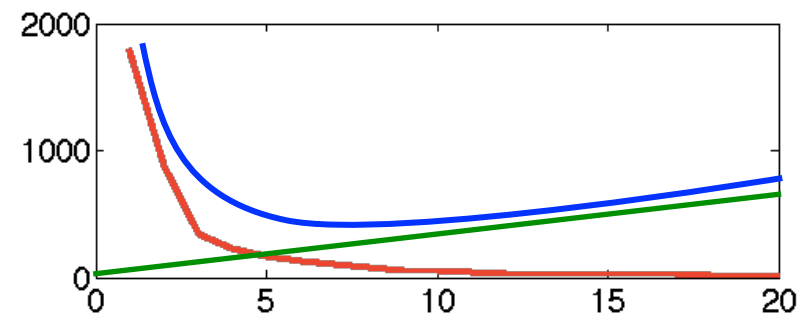
Choosing the number of clusters

- 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
$$J(\underline{z}, \underline{\mu}) = \log \left[\frac{1}{m} \sum_i \|x_i - \mu_{z_i}\|^2 \right] + k \frac{\log m}{m}$$
 - More precise version: see e.g. “X-means” (Pelleg & Moore 2000)



Summary

- K-Means clustering
 - Clusters described as locations (“centers”) in feature space
- Procedure
 - Initialize cluster centers
 - Iterate: assign each data point to its closest cluster center
 - : move cluster centers to minimize mean squared error
- Properties
 - Coordinate descent on MSE criterion
 - Prone to local optima; initialization important
- Out-of-sample data
- Choosing the # of clusters, K
 - Model selection problem; penalize for complexity (BIC, etc.)

+

Machine Learning and Data Mining

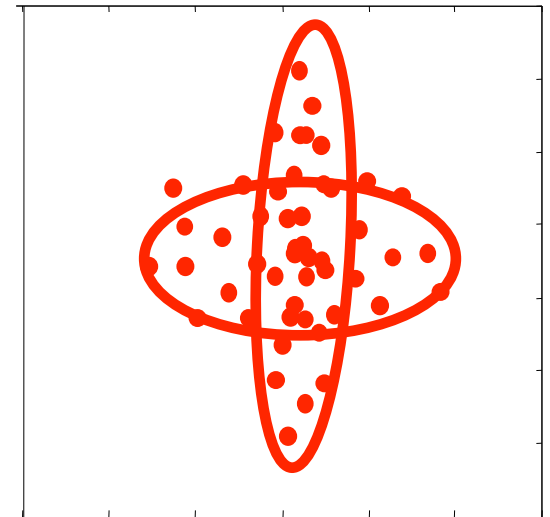
Clustering (4): Gaussian Mixtures & EM

Prof. Alexander Ihler



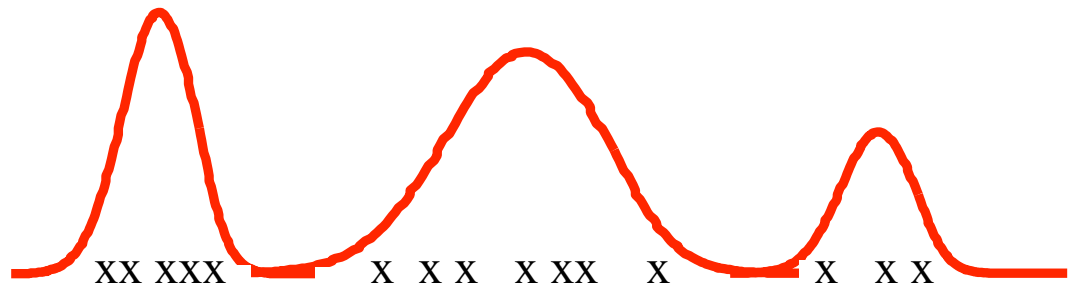
Mixtures of Gaussians

- 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”)



Mixtures of Gaussians

- Start with parameters describing each cluster
- Mean μ_c , variance σ_c , “size” π_c
- Probability distribution: $p(x) = \sum_c \pi_c \mathcal{N}(x ; \mu_c, \sigma_c)$



Mixtures of Gaussians

- Start with parameters describing each cluster
- Mean μ_c , variance σ_c , “size” π_c
- Probability distribution: $p(x) = \sum_c \pi_c \mathcal{N}(x ; \mu_c, \sigma_c)$
- Equivalent “latent variable” form:

$$p(z = c) = \pi_c$$

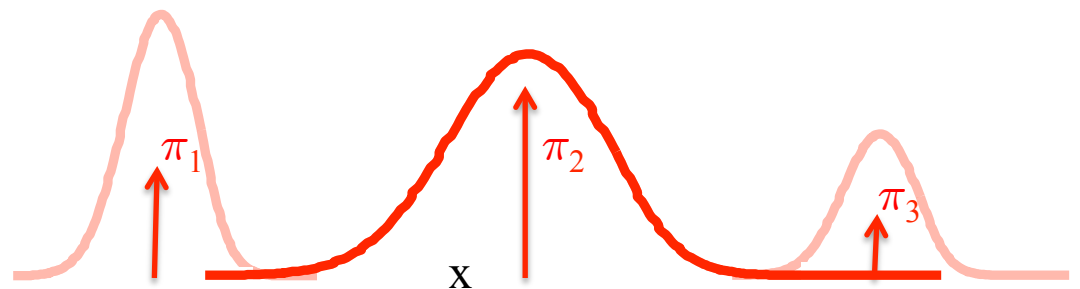
Select a mixture component with probability π

$$p(x|z = c) = \mathcal{N}(x ; \mu_c, \sigma_c)$$

Sample from that component's Gaussian

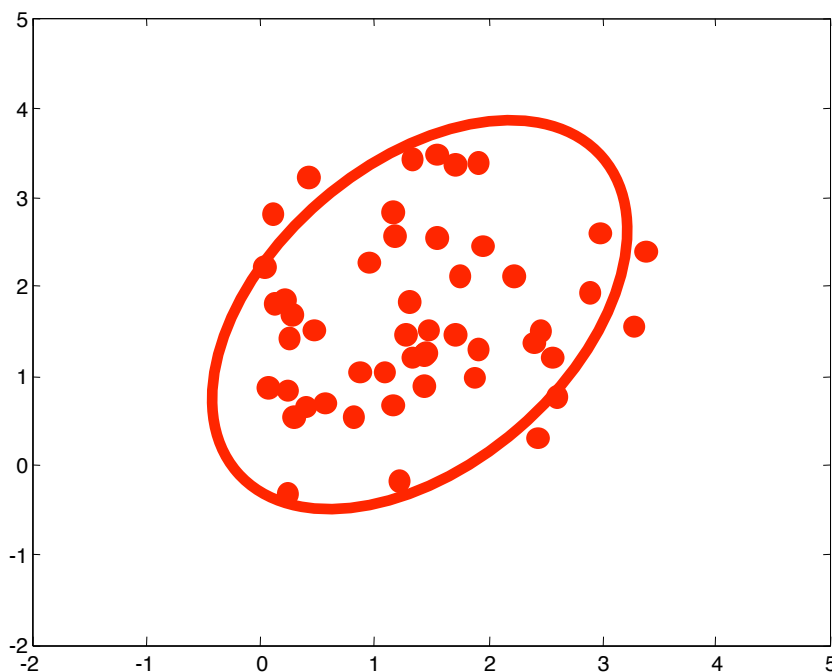
“Latent assignment” z :
we observe x , but z is hidden

$p(x)$ = marginal over x



Multivariate Gaussian models

$$\mathcal{N}(\underline{x} ; \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) \right\}$$



Maximum Likelihood estimates

$$\hat{\mu} = \frac{1}{m} \sum_i x^{(i)}$$

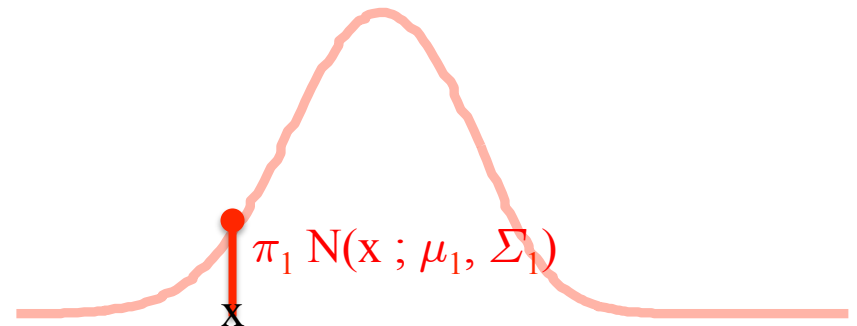
$$\hat{\Sigma} = \frac{1}{m} \sum_i (x^{(i)} - \hat{\mu})^T (x^{(i)} - \hat{\mu})$$

**We'll model each cluster
using one of these Gaussian
“bells”...**

EM Algorithm: E-step

- Start with clusters: Mean μ_c , Covariance Σ_c , “size” π_c
- E-step (“Expectation”)
 - For each datum (example) x_i ,
 - Compute “ r_{ic} ”, the probability that it belongs to cluster c
 - Compute its probability under model c
 - Normalize to sum to one (over clusters c)

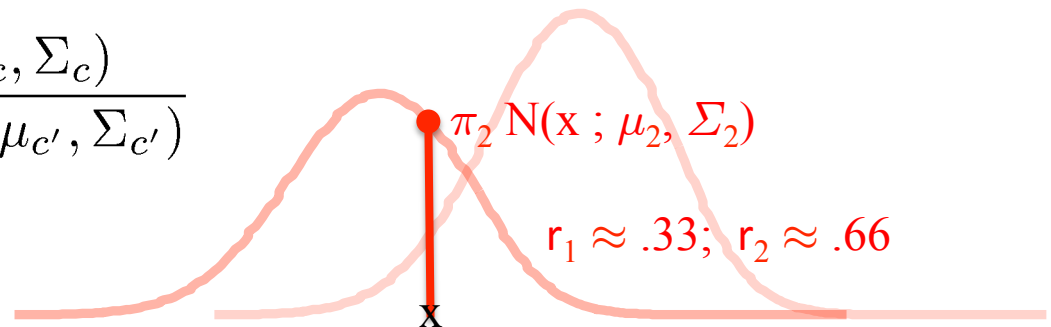
$$r_{ic} = \frac{\pi_c \mathcal{N}(x_i ; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i ; \mu_{c'}, \Sigma_{c'})}$$



EM Algorithm: E-step

- Start with clusters: Mean μ_c , Covariance Σ_c , “size” π_c
- E-step (“Expectation”)
 - For each datum (example) x_i ,
 - Compute “ r_{ic} ”, the probability that it belongs to cluster c
 - Compute its probability under model c
 - Normalize to sum to one (over clusters c)

$$r_{ic} = \frac{\pi_c \mathcal{N}(x_i ; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i ; \mu_{c'}, \Sigma_{c'})}$$



- If x_i is very likely under the c^{th} Gaussian, it gets high weight
- Denominator just makes r 's sum to one

EM Algorithm: M-step

- Start with assignment probabilities r_{ic}
- Update parameters: mean μ_c , Covariance Σ_c , “size” π_c
- M-step (“Maximization”)
 - For each cluster (Gaussian) $z = c$,
 - Update its parameters using the (weighted) data points

$$m_c = \sum_i r_{ic} \quad \text{Total responsibility allocated to cluster } c$$

$$\pi_c = \frac{m_c}{m} \quad \text{Fraction of total assigned to cluster } c$$

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)} \quad \Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

Weighted mean of assigned data

Weighted covariance of assigned data
(use new weighted means here)

Expectation-Maximization

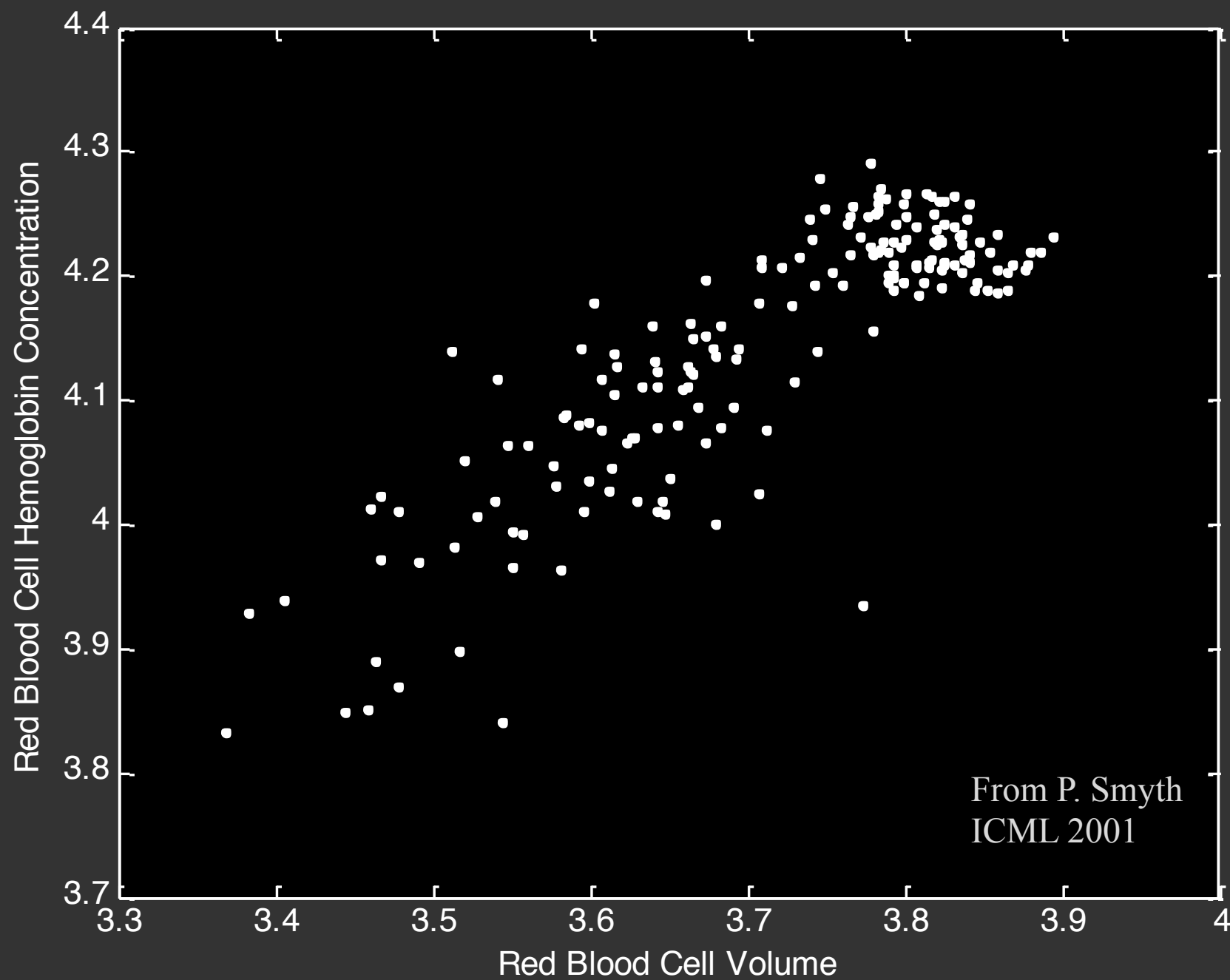
- Each step increases the log-likelihood of our model

$$\log p(\underline{X}) = \sum_i \log \left[\sum_c \pi_c \mathcal{N}(x_i ; \mu_c, \Sigma_c) \right]$$

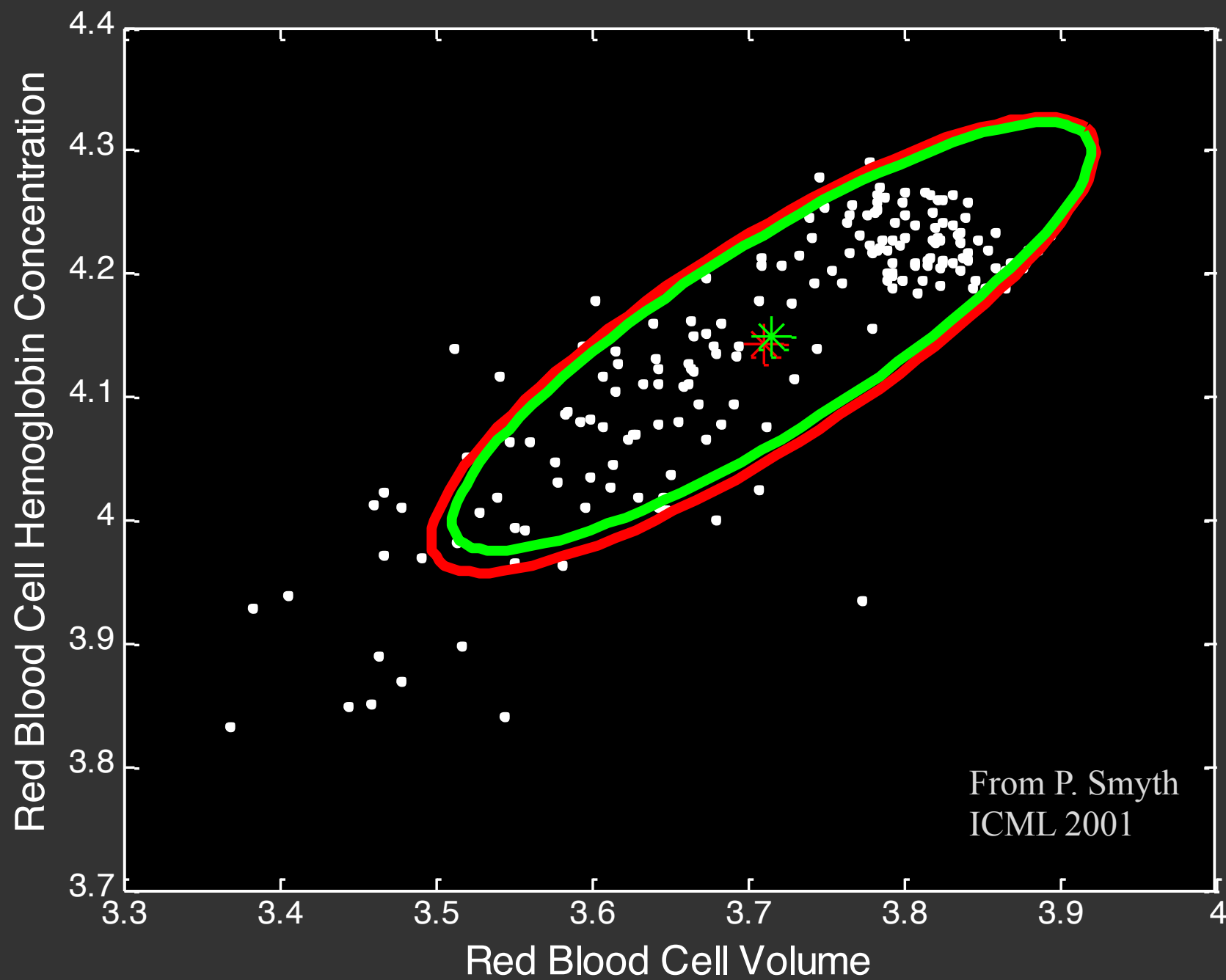
(we won't derive this here, though)

- Iterate until convergence
 - Convergence guaranteed – another ascent method
 - Local optima: initialization often important
- What should we do
 - If we want to choose a single cluster for an “answer”?
 - With new data we didn't see during training?
- Choosing the number of clusters
 - Can use penalized likelihood of training data (like k-means
 - True probability model: can use log-likelihood of test data, $\log p(x')$

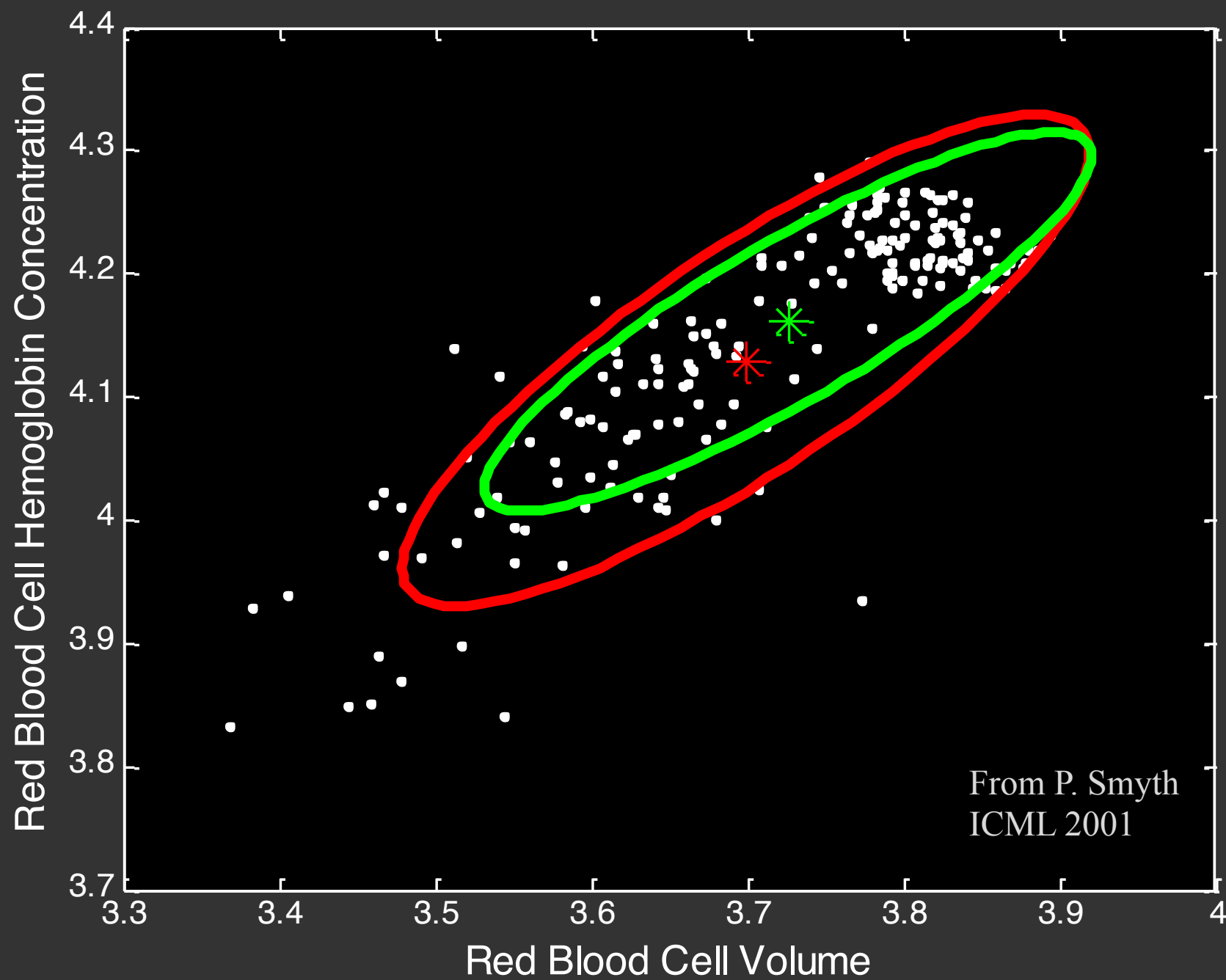
ANEMIA PATIENTS AND CONTROLS



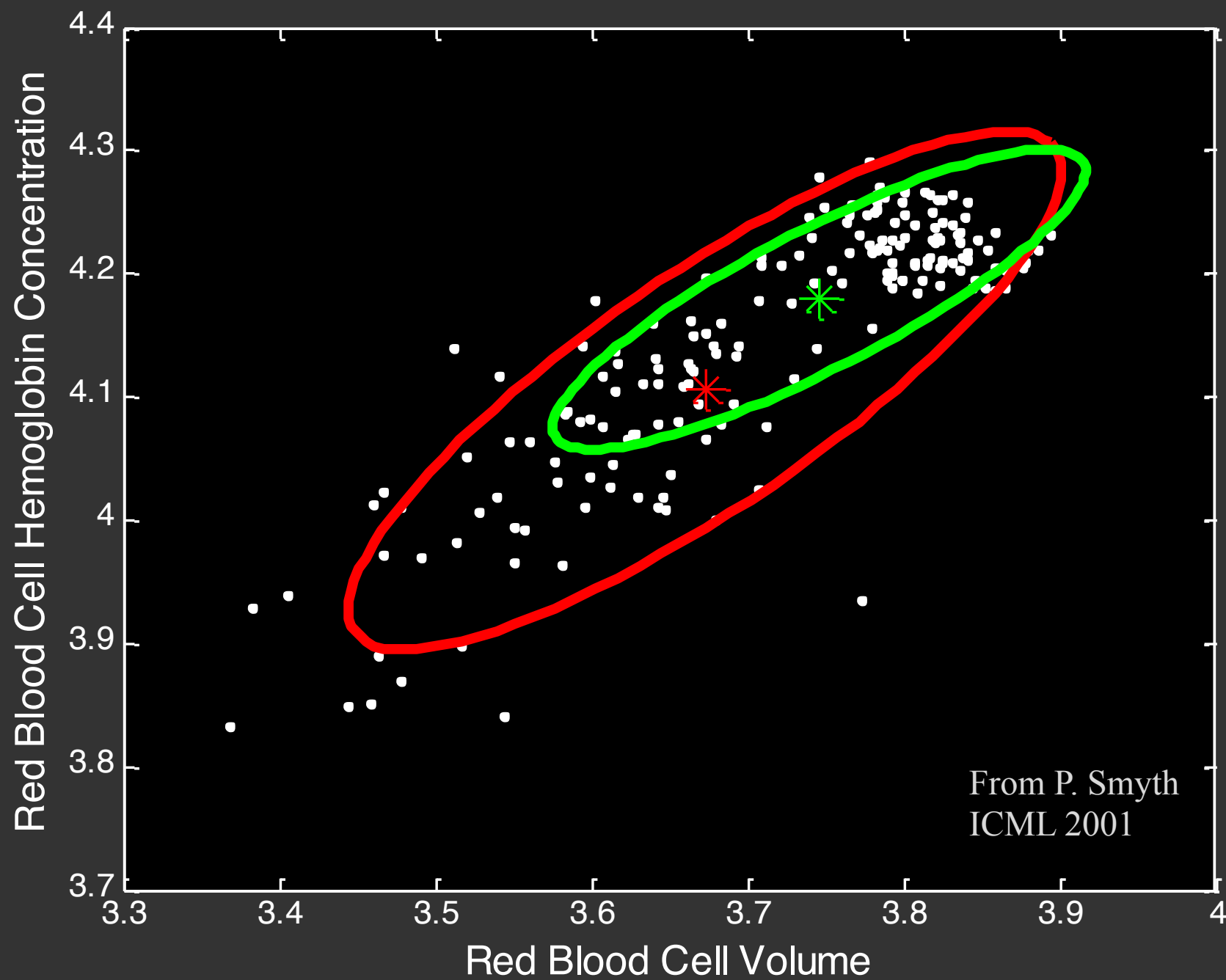
EM ITERATION 1



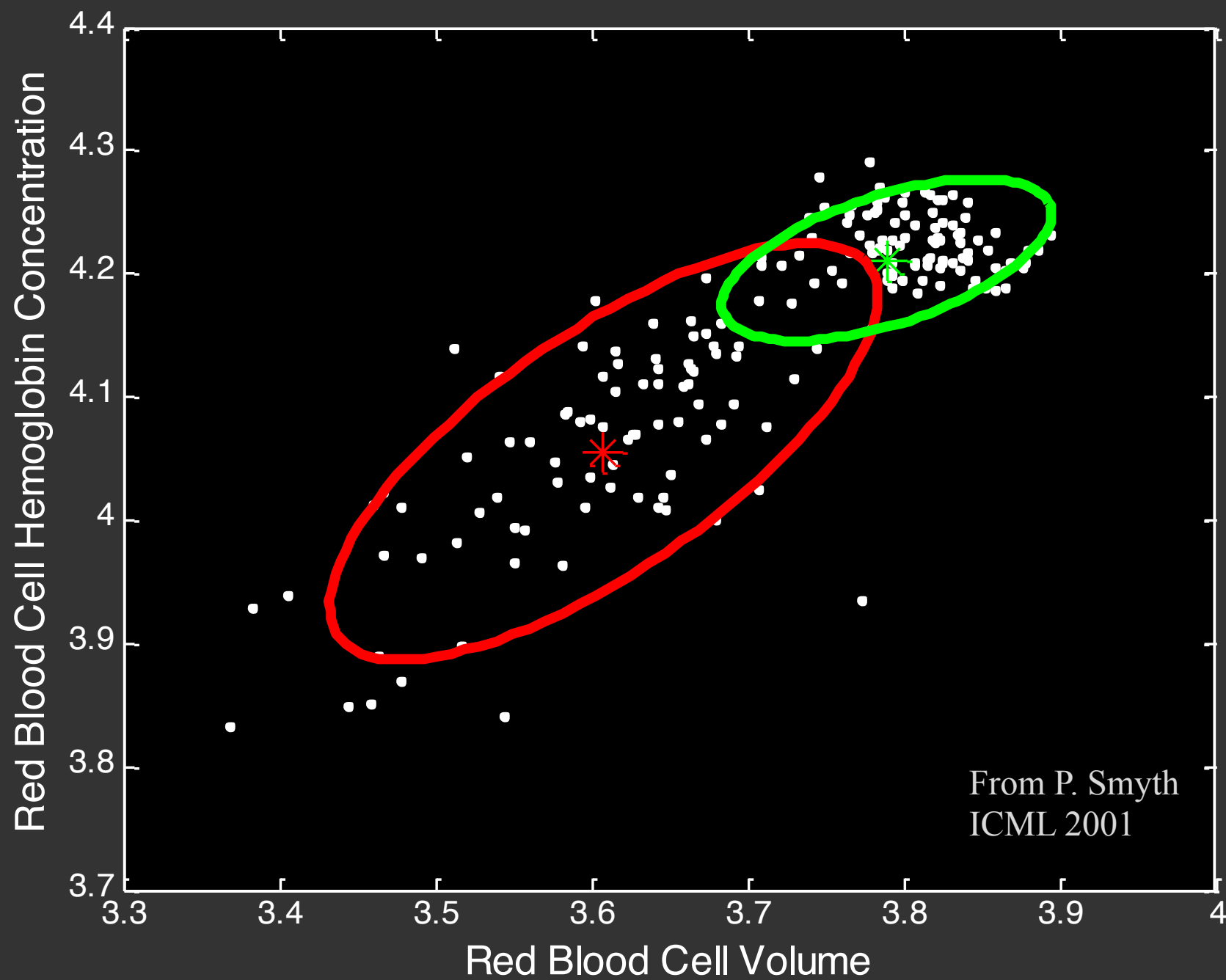
EM ITERATION 3



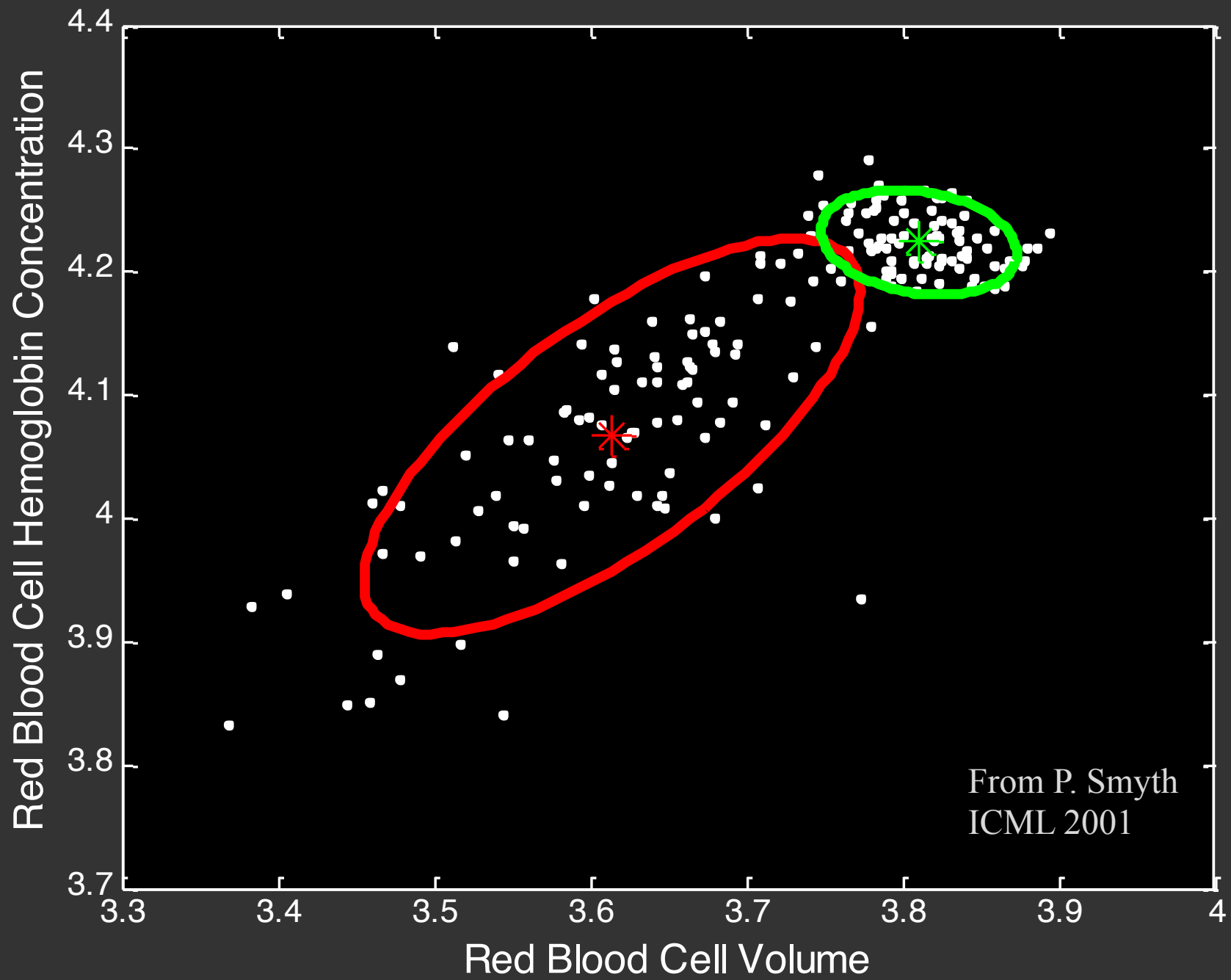
EM ITERATION 5



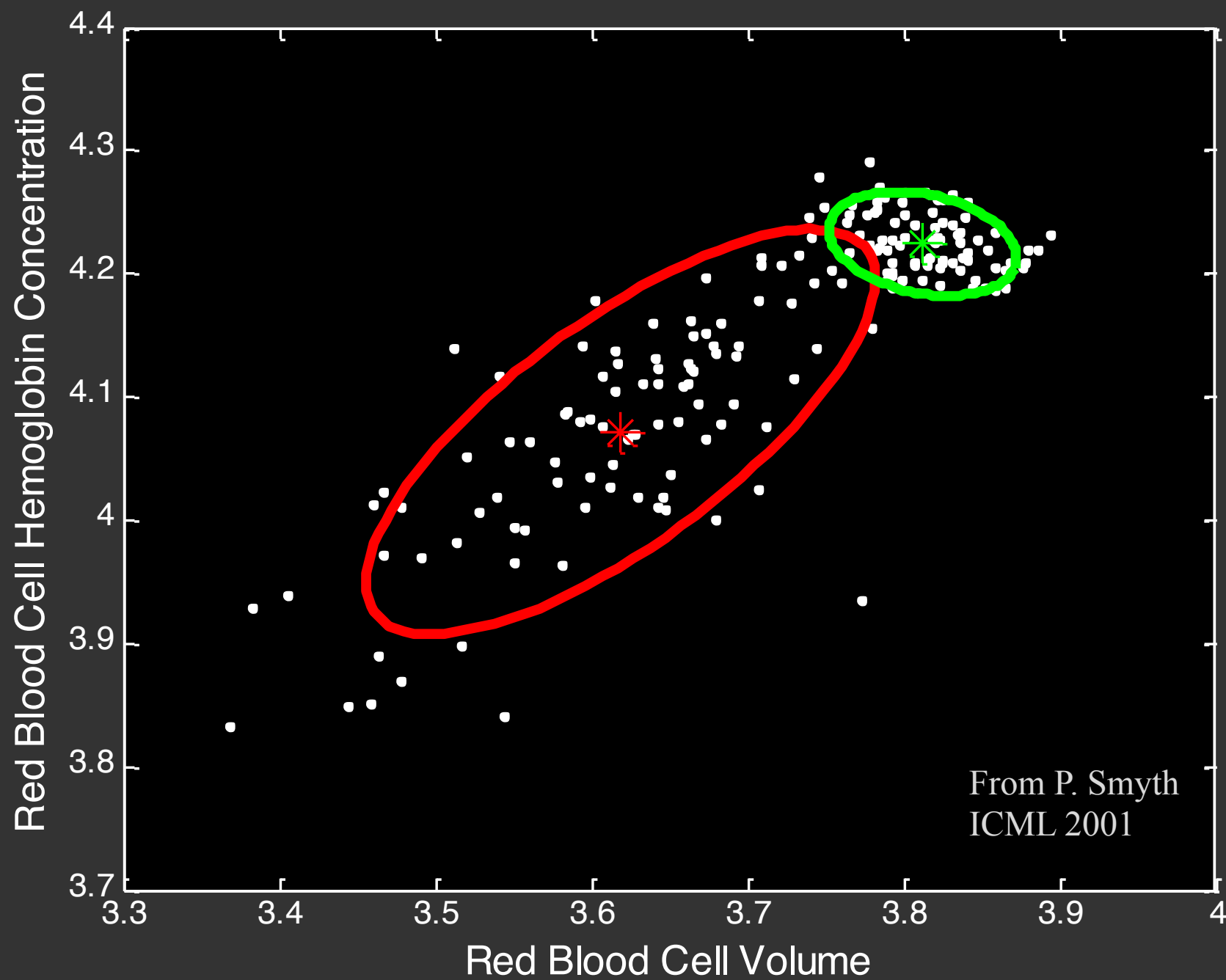
EM ITERATION 10



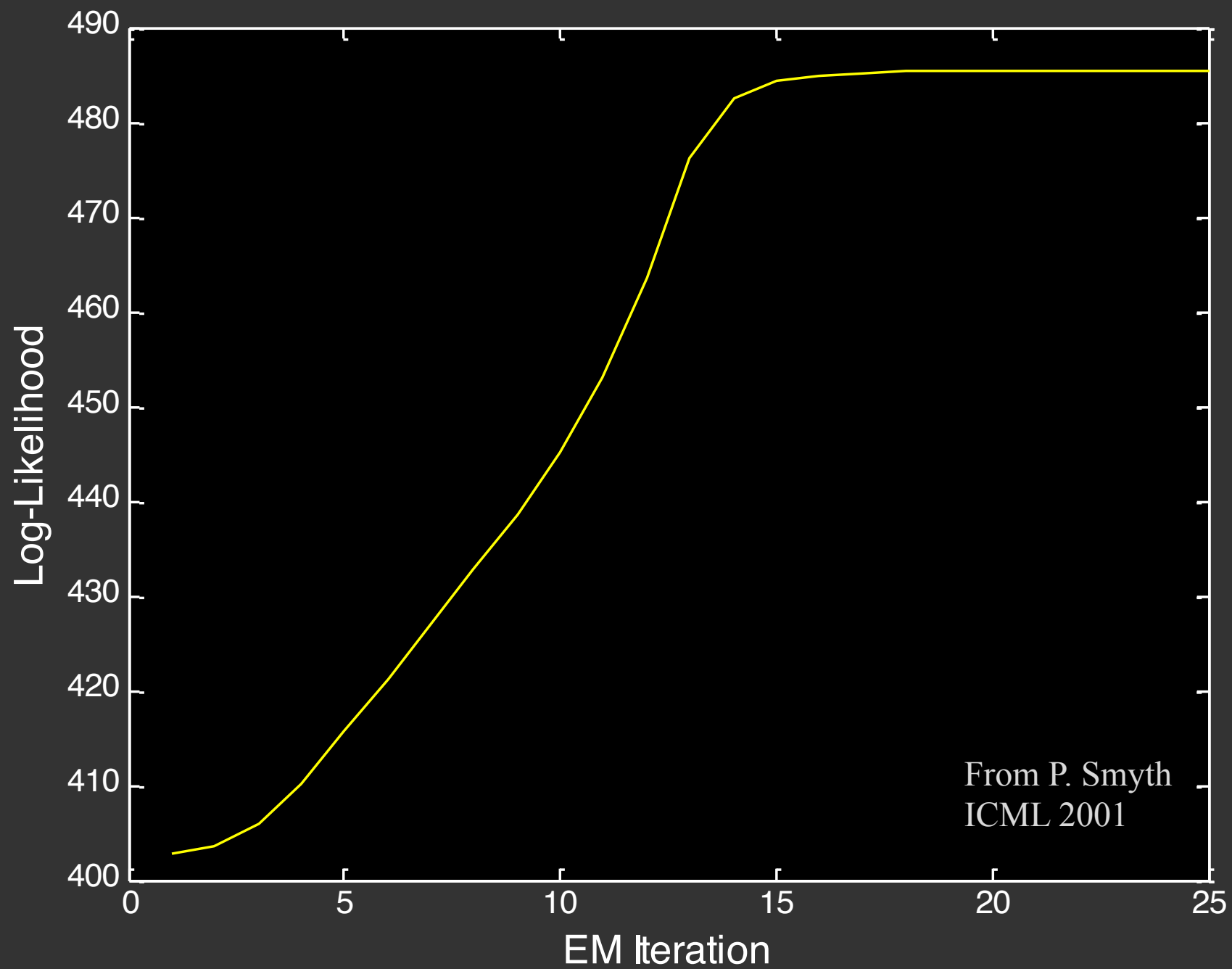
EM ITERATION 15



EM ITERATION 25



LOG-LIKELIHOOD AS A FUNCTION OF EM ITERATIONS



EM and missing data

- EM is a general framework for partially observed data
 - “Complete data” x_i, z_i – features and assignments
 - Assignments z_i are missing (unobserved)
- EM corresponds to
 - Computing the distribution over all z_i given the parameters
 - Maximizing the “expected complete” log likelihood
 - GMMs = plug in “soft assignments”, but not always so easy
- Alternatives: Stochastic EM, Hard EM
 - Instead of expectations, just sample the z_i or choose best (often easier)
 - Called “imputing” the values of z
 - Hard EM: similar to EM, but less “smooth”, more local minima
 - Stochastic EM: similar to EM, but with extra randomness
 - Not obvious when it has converged

Summary

- Gaussian mixture models
 - Flexible class of probability distributions
 - Explain variation with hidden groupings or clusters of data
 - Latent “membership” $z^{(i)}$
 - Feature values $x^{(i)}$ are Gaussian given $z^{(i)}$
- Expectation-Maximization
 - Compute soft membership probabilities, “responsibility” r_{ic}
 - Update mixture component parameters given soft memberships
 - Ascent on log-likelihood: convergent, but local optima
- Selecting the number of clusters
 - Penalized likelihood or validation data likelihood

Gibbs sampling for clustering

- Another technique for inferring uncertain cluster assignments
 - K-means: take the best assignment
 - EM: assign “partially”
 - Stochastic EM: sample assignment
 - All: choose best cluster descriptions given assignments
- Gibbs sampling (“Markov chain Monte Carlo”)
 - Assign randomly, probability equal to EM’s weight
 - *Sample* a cluster description given assignment
 - Requires a probability model over cluster parameters
- This doesn’t really find the “best” clustering
 - It eventually samples almost all “good” clusterings
 - Converges “in probability”, randomness helps us explore configurations
 - Also tells us about uncertainty of clustering
 - Disadvantage: not obvious when “done”

“Infinite” mixture models

- How many clusters are there?
- Gibbs sampling has an interesting solution
 - Write a distribution over k , the # of clusters
 - Sample k also
- Can do our sampling sequentially
 - Draw each z_i given all the others
 - Instead of sampling cluster parameters, marginalize them
 - Defines a distribution over groupings of data
- Now, for each z_i , sample
 - Join an existing cluster? Or, join a new cluster?
- What are these probabilities?
 - “Dirichlet process” mixture models

Parametric and Nonparametric Models

- Every model has some parameters
 - “The stuff you have to store to make your prediction”
 - Logistic regression: weights
 - Decision tree: feature to split, value at each level
 - Gaussian mixture model: means, covariances, sizes
- Parametric vs Nonparametric models
 - Parametric: fixed # of parameters
 - Nonparametric: # of parameters grows with more data
- What type are
 - Logistic regression?
 - Nearest neighbor prediction?
 - Decision trees?
 - Decision trees of depth < 3 ?
 - Gaussian mixture model?

Summary

- Clustering algorithms
 - Agglomerative clustering
 - K-means
 - Expectation-Maximization

Open questions for each application:

- What does it mean to be “close” or “similar”?
 - Depends on your particular problem...
- “Local” versus “global” notions of similarity
 - Former is easy, but we usually want the latter...
- Is it better to “understand” the data itself (unsupervised learning), to focus just on the final task (supervised learning), or both?
- Do we need a generative model? Out-of-sample assignments?