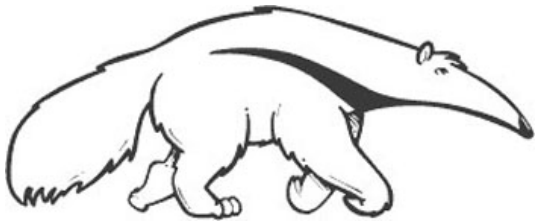
+

# CS178: Machine Learning and Data Mining

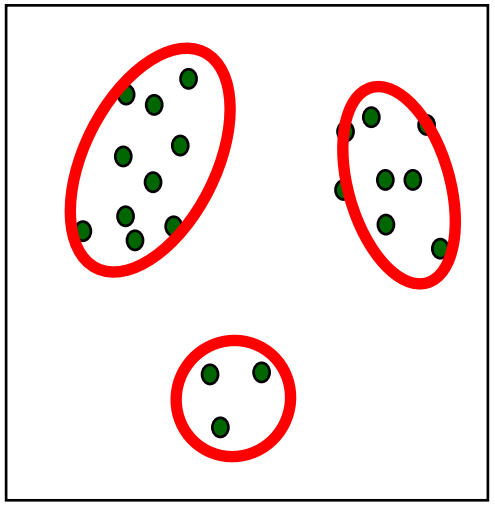
## Clustering

Prof. Alexander Ihler



# Unsupervised learning

- Supervised learning
  - Predict target value (“y”) given features (“x”)

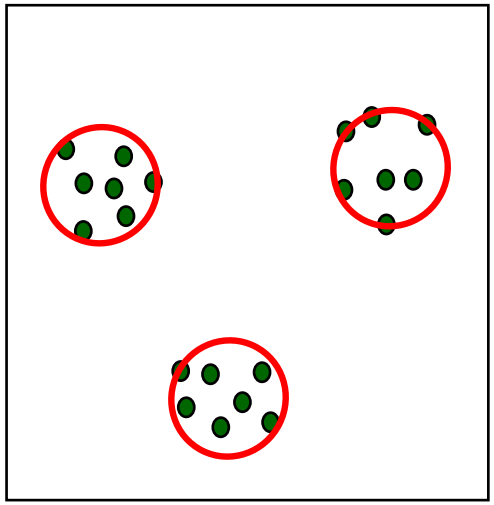


- Unsupervised learning
  - Understand patterns of data (just “x”)
  - Useful for many reasons
    - Data mining (“explain”)
    - Missing data values (“impute”)
    - Representation (feature generation or selection)

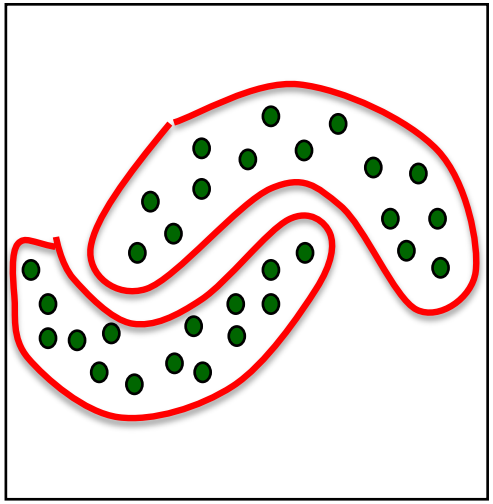
- One example: *clustering*
  - Describe data by discrete “groups” with some characteristics

# Clustering

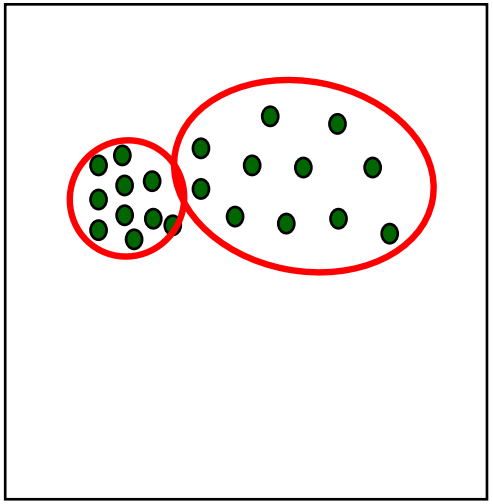
- Clustering describes data by “groups”
- The meaning of “groups” may vary by data!
- Examples



Location



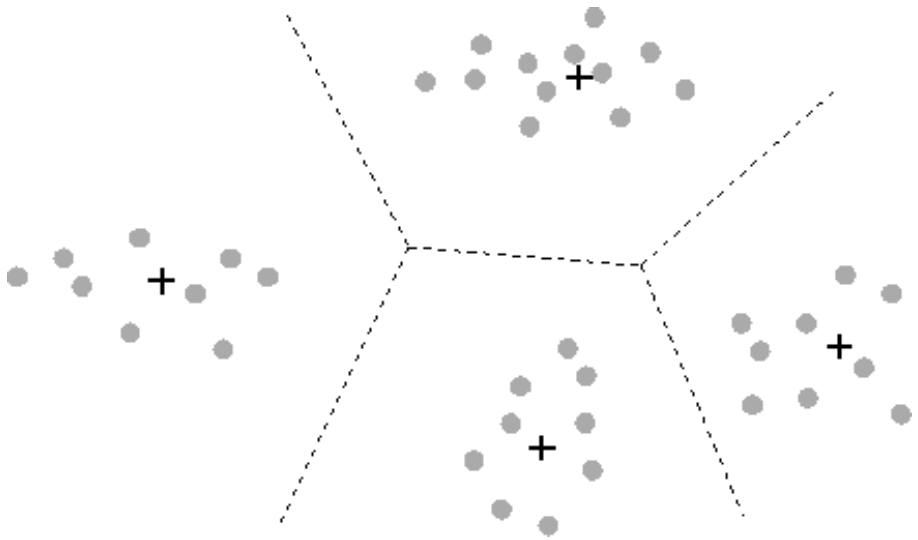
Shape

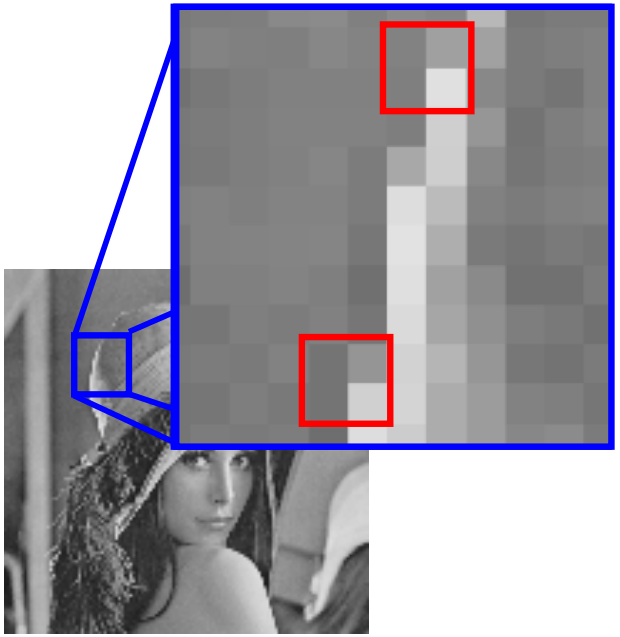


Density

# Clustering & Data Compression

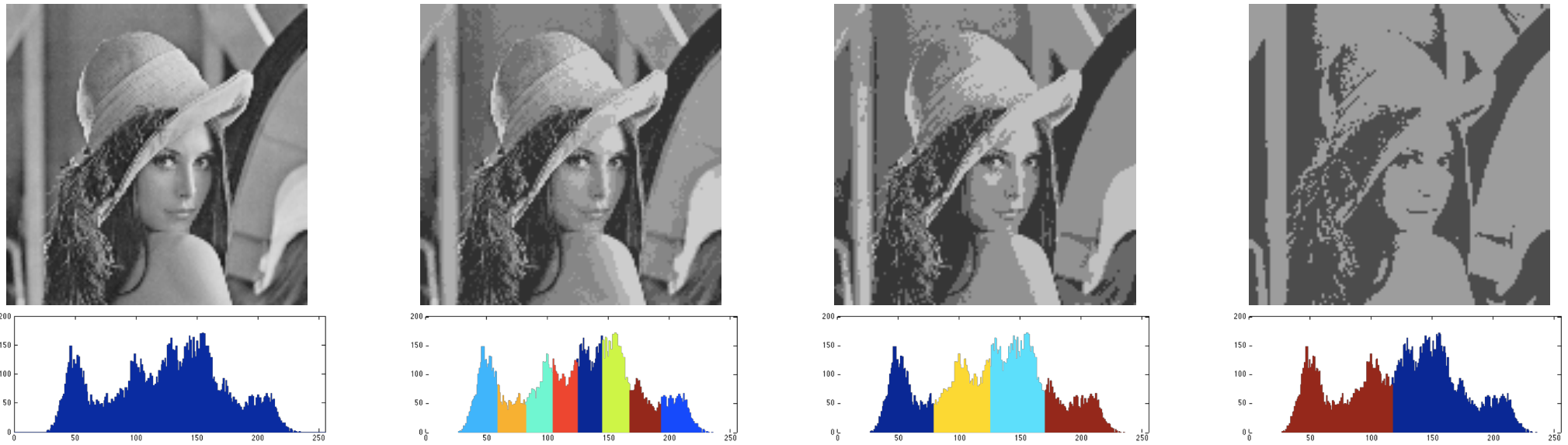
- Clustering is related to vector quantization
  - Dictionary of vectors (the cluster centers)
  - Each original value represented using a dictionary index
  - Each center “claims” a nearby region (Voronoi region)





# Clustering & 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

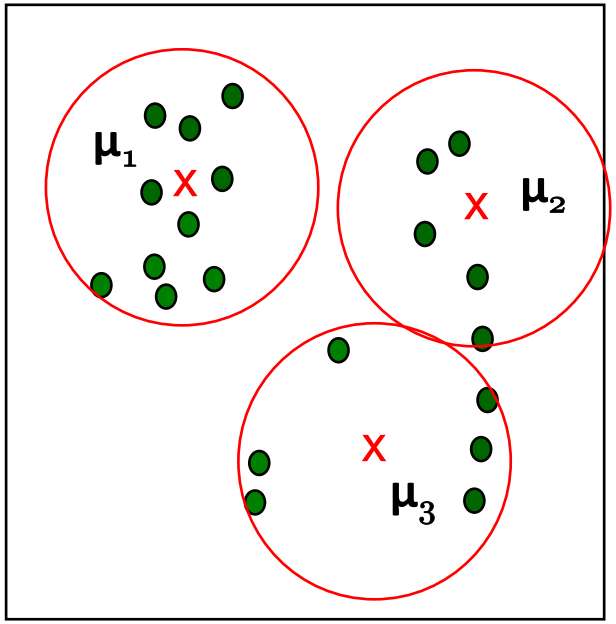
**Clustering: K-Means**

**Clustering: Agglomerative**

**Clustering: Gaussian Mixtures and EM**

# 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 $x_i$

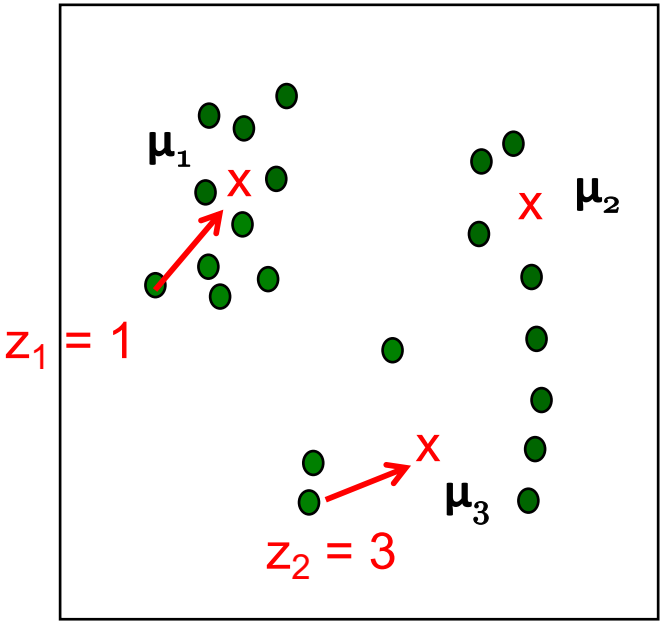
Assume K clusters

Each cluster c "described" by a center $\mu_c$

Each cluster will "claim" a set of nearby points

# 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 $\mu_c$

Each cluster will "claim" a set of nearby points

"Assignment" of $i^{th}$ example: $z_i \in 1,\ldots,K$

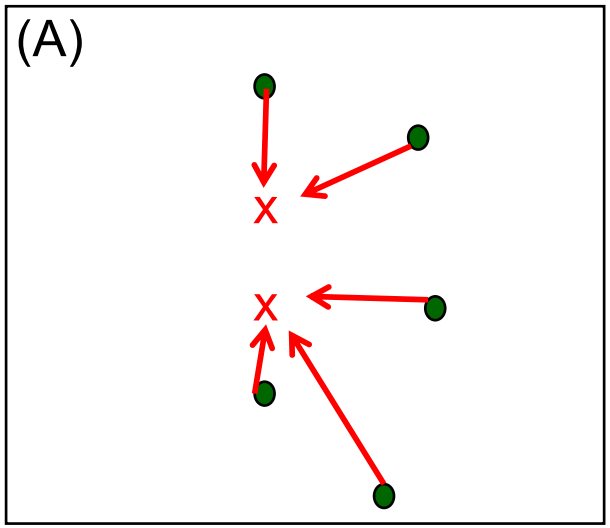
# K-Means Clustering

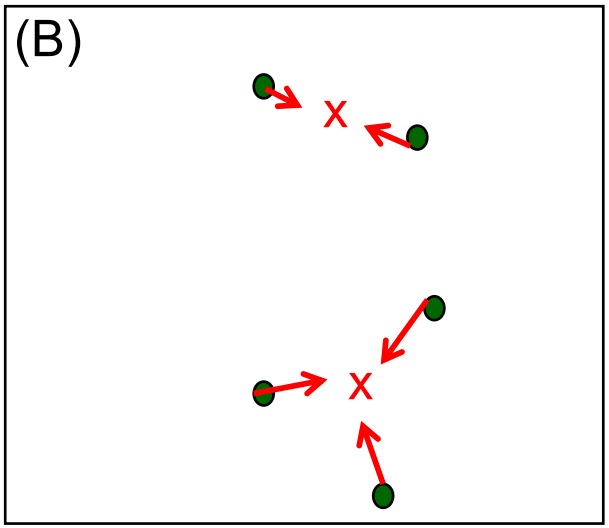
- Iterate until convergence:
  - (A) For each datum, find the closest cluster

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

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

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



# K-Means Clustering

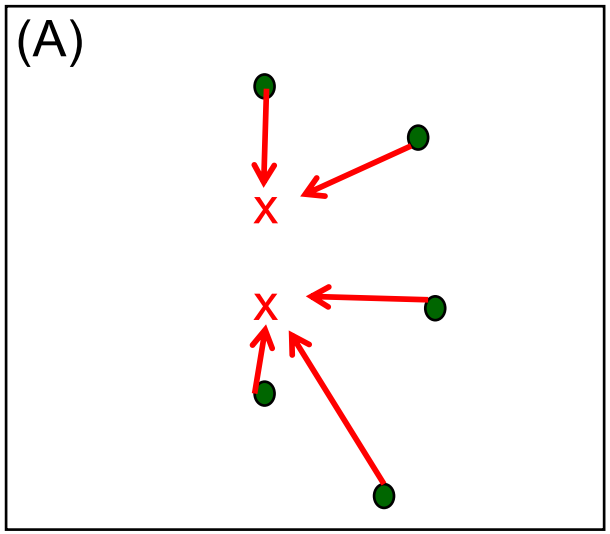
- Optimizing the cost function:

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

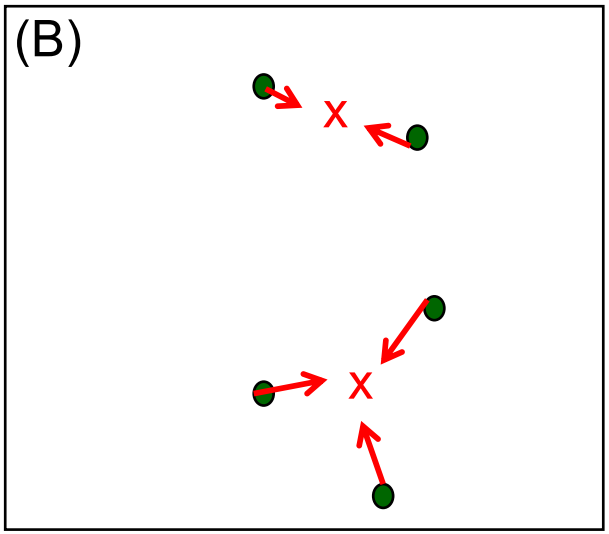
- Coordinate descent:

**Descent => guaranteed to converge**
New means = same assignments
Same assignments = same means
Same means = same assignments

**Over the cluster assignments (fixed μ):**
Only one term in sum depends on $z_i$
Minimized by selecting closest $\mu_c$




**Over the cluster centers (fixed $z$):**
Cluster c only depends on $x_i$ with $z_i$=c
Minimized by selecting the mean

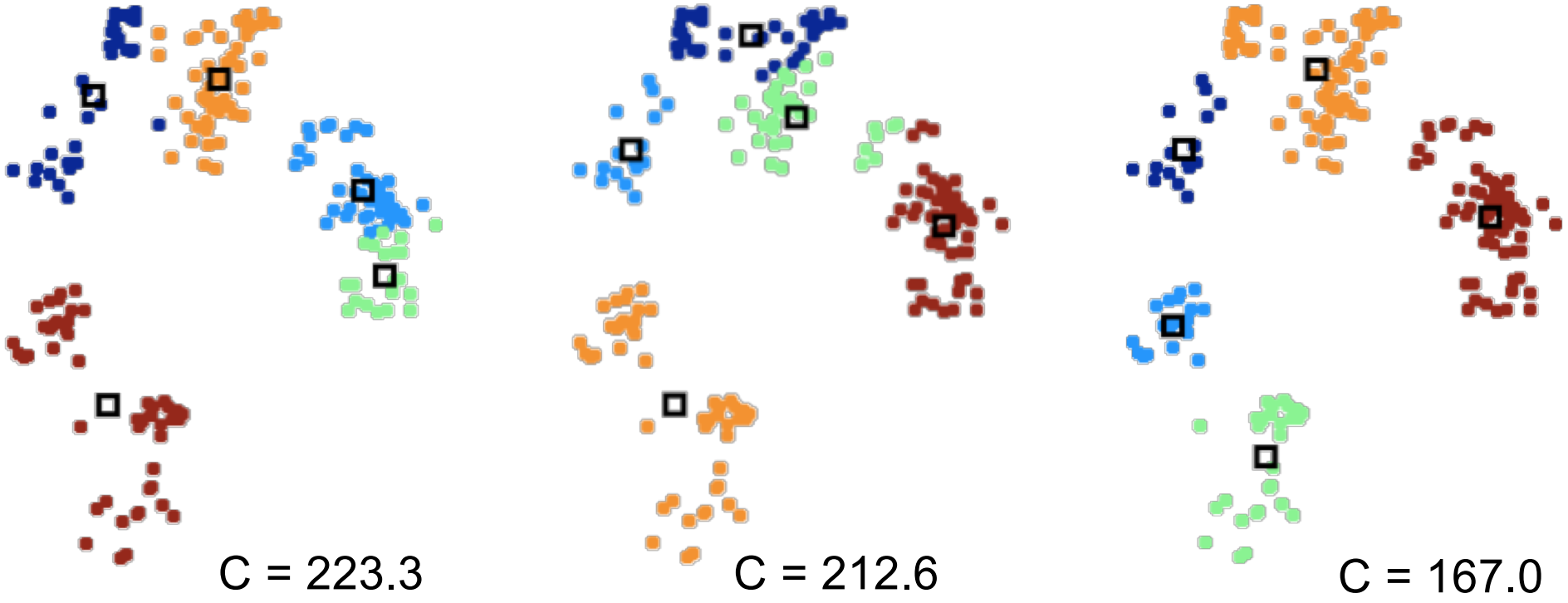


# Demo Time!

http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

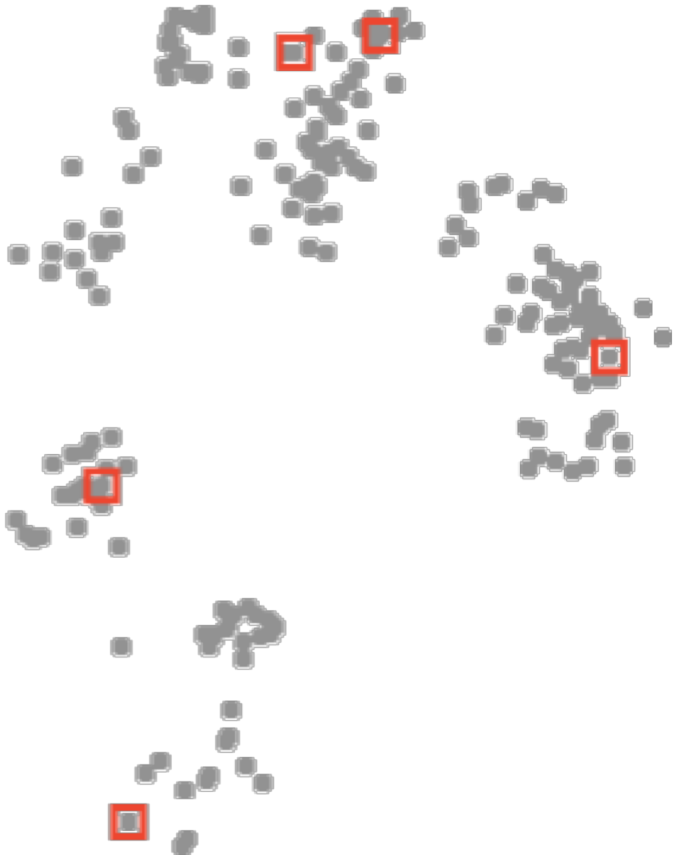
# Initialization

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



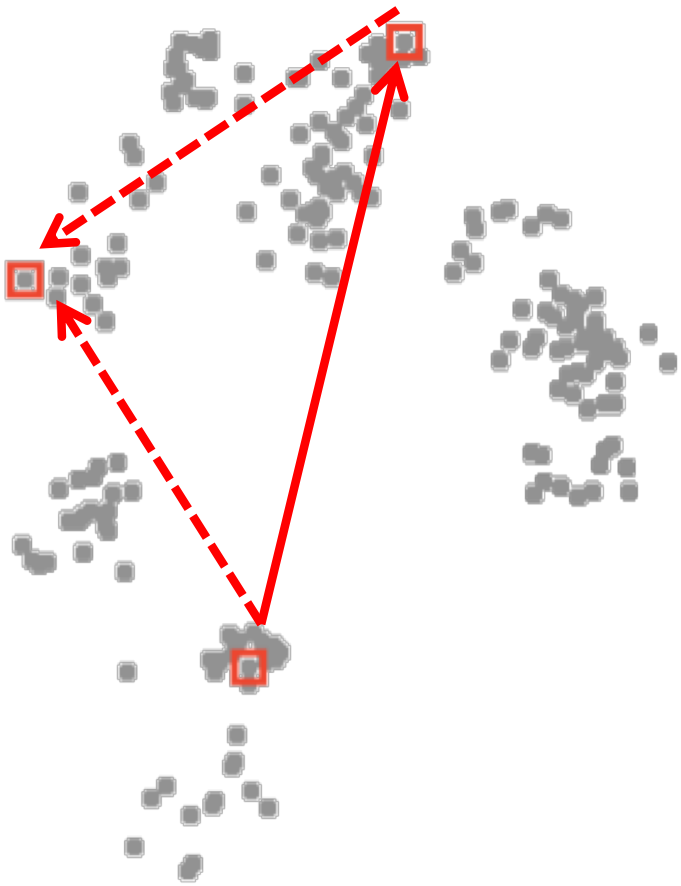
# Initialization methods

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



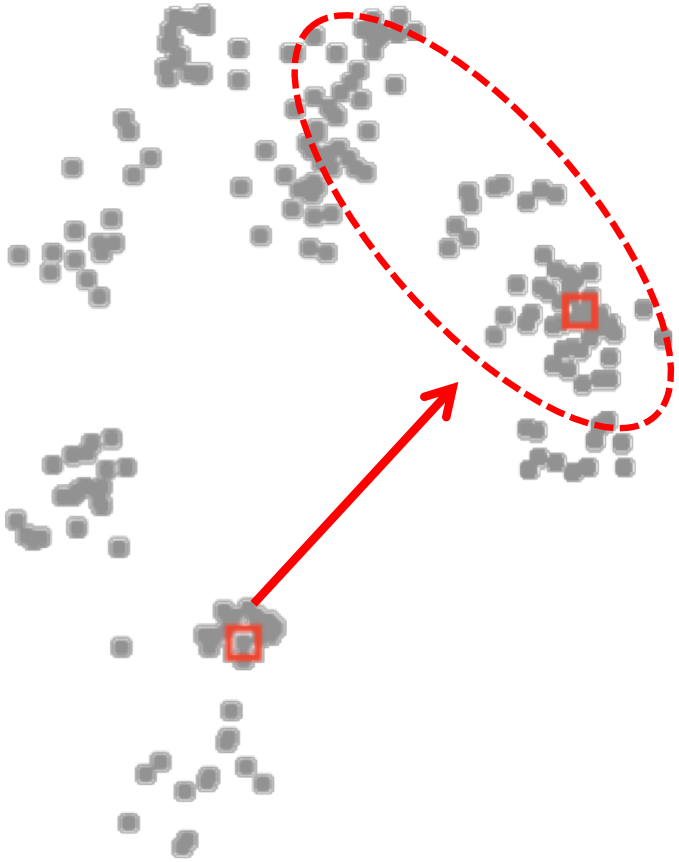
# Initialization methods

- 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 + distance ("k-means++")
  - Choose next points "far but randomly"
  - p(x) ∝ squared distance from x to current centers
  - Likely to put a cluster far away, in a region with lots of data



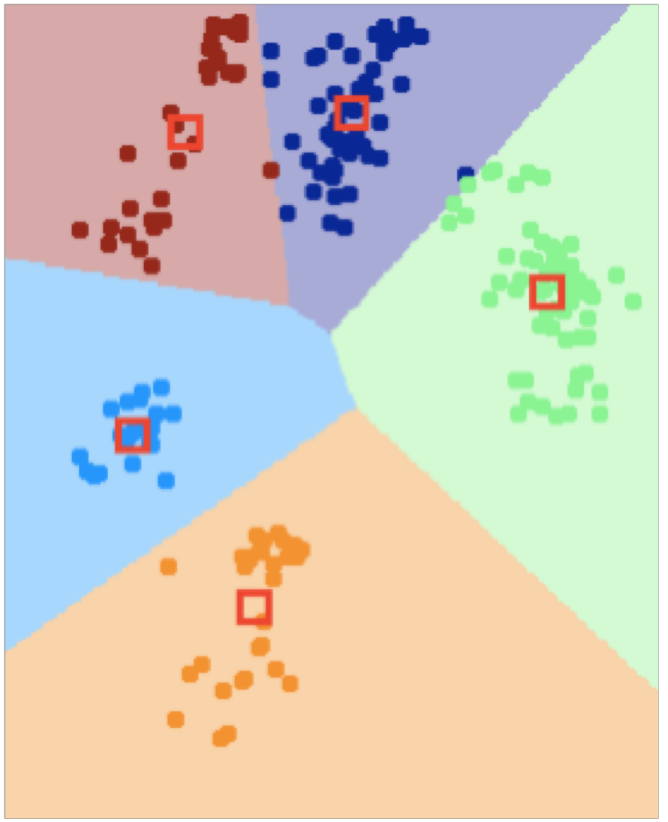
# Out-of-sample points

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

```
# perform clustering
Z , mu , score = kmeans(X, K);

# cluster id = nearest center
L = knnClassify(mu, range(K), 1);

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

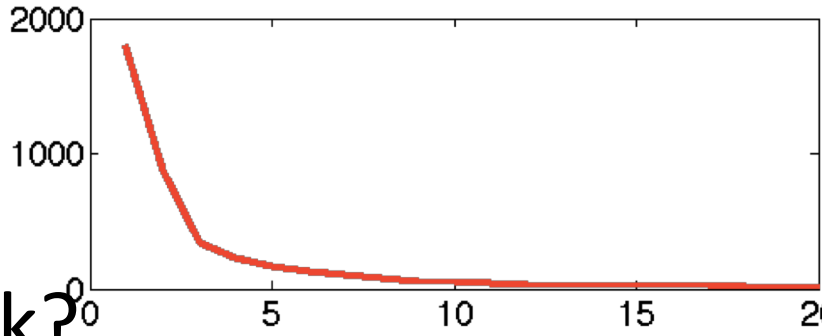


# Choosing Number of Clusters

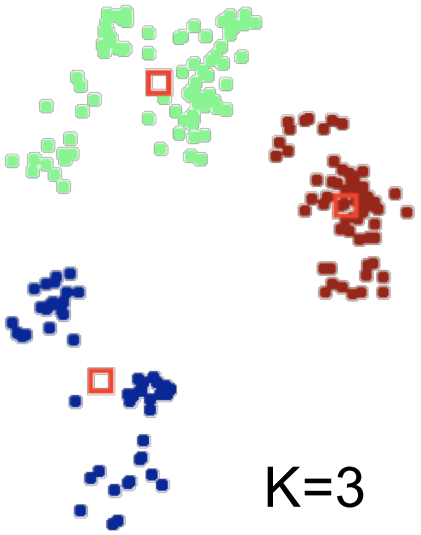
- With cost function

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

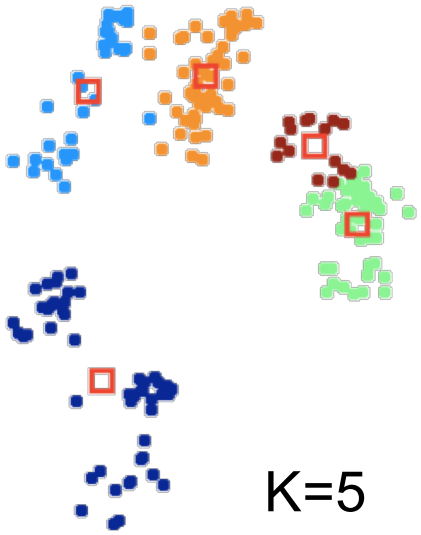
what is the optimal value of k?



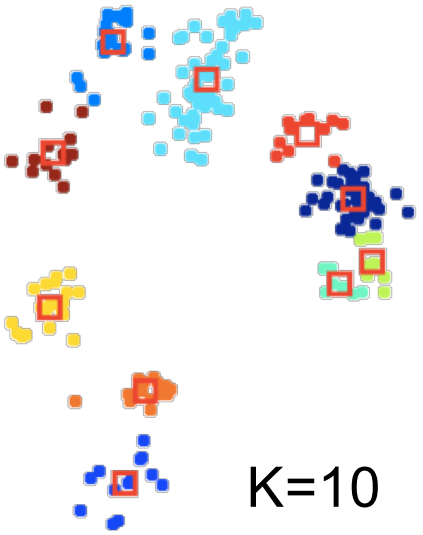

- Cost always decreases with k!
- A model complexity issue…



K=3



K=5

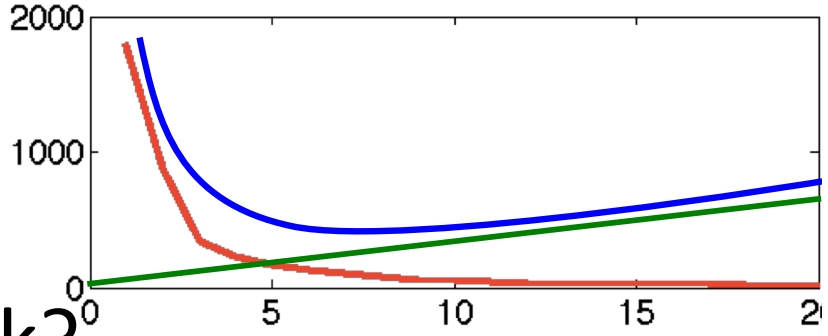


K=10

# Choosing 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 don't help "enough"
  - Example: simplified BIC penalty

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

# 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

Clustering: K-Means

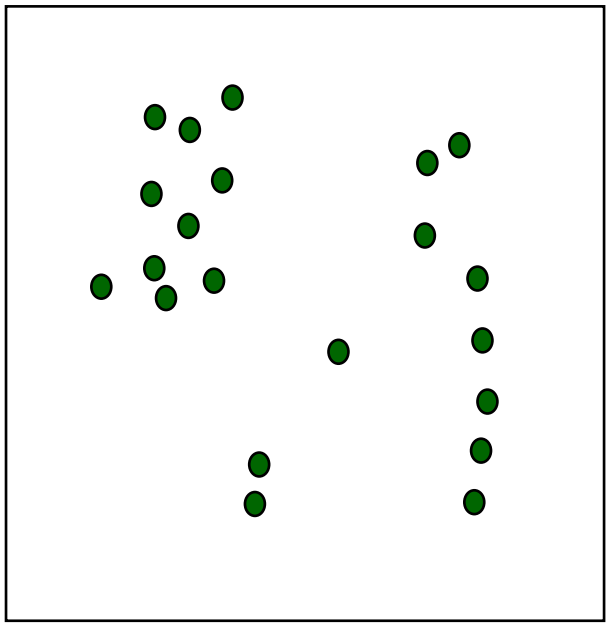
**Clustering: Agglomerative**

Clustering: Gaussian Mixtures and EM

# Hierarchical Agglomerative Clustering

Initially, every datum is a cluster

Data:



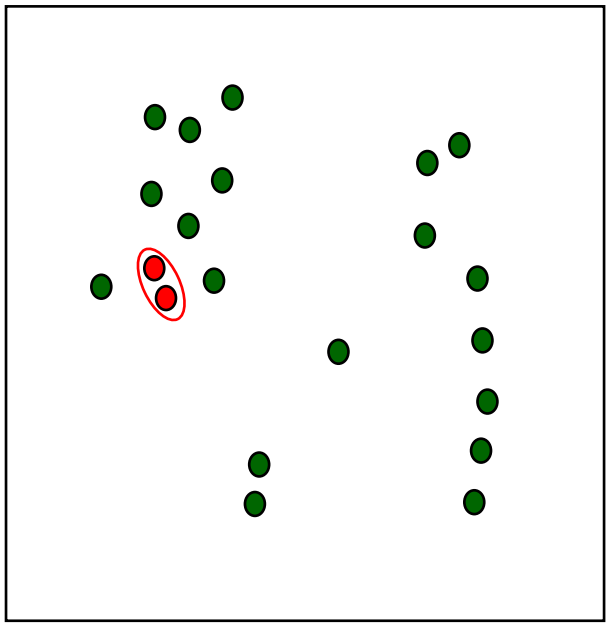
- A simple clustering algorithm
- Define a distance (or dissimilarity) between clusters (we'll return to this)
- Initialize: every example is a cluster
- Iterate:
  - Compute distances between all clusters (store for efficiency)
  - Merge two closest clusters
- Save both clustering and sequence of cluster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

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

# Iteration 1

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:

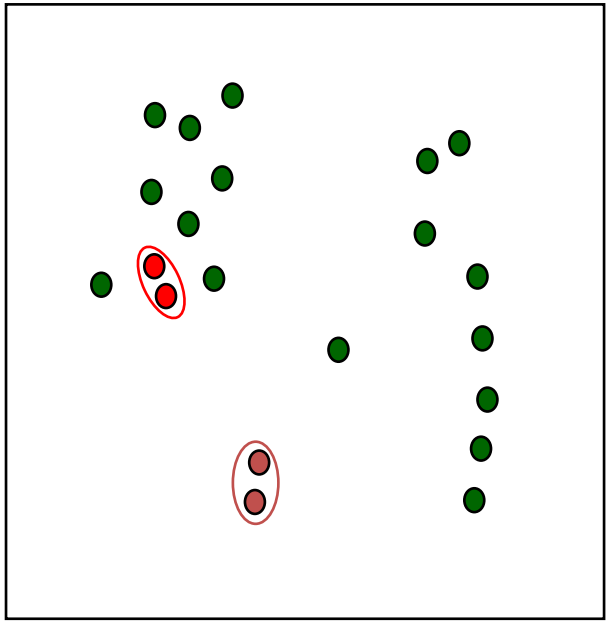
Height of the join indicates dissimilarity

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

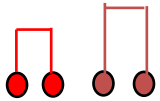
# Iteration 2

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



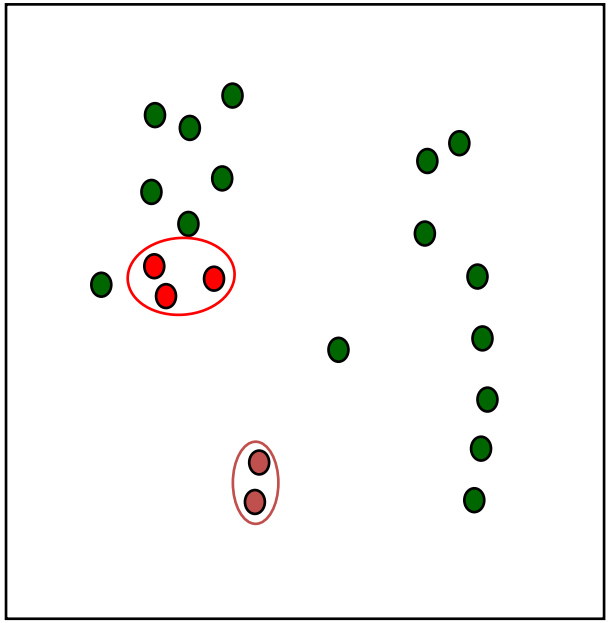
Height of the join indicates dissimilarity

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

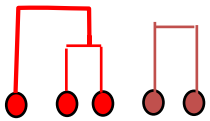
# Iteration 3

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



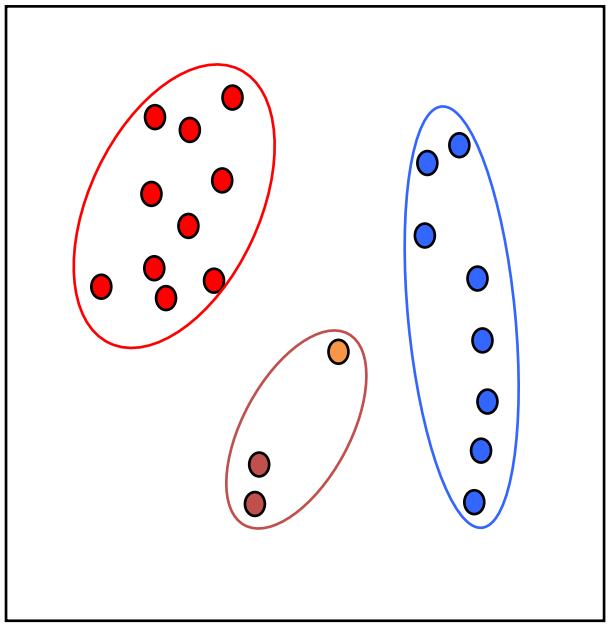
Height of the join indicates dissimilarity

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

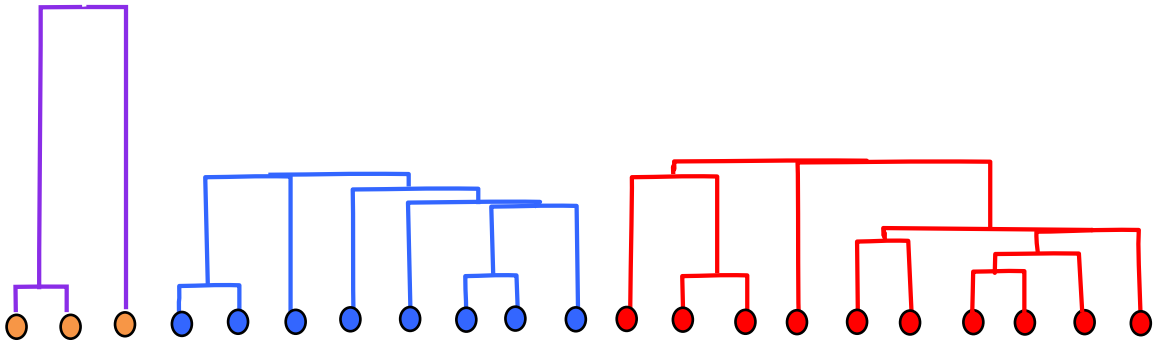
# Iteration m-3

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



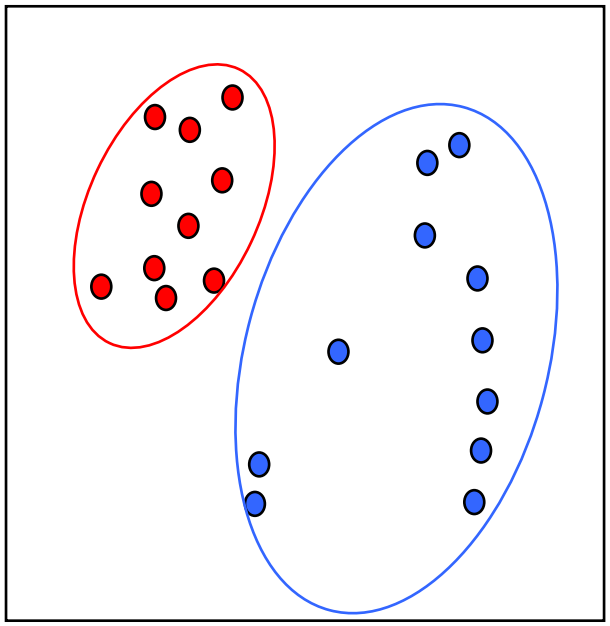
In mltools: “agglomerative”

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

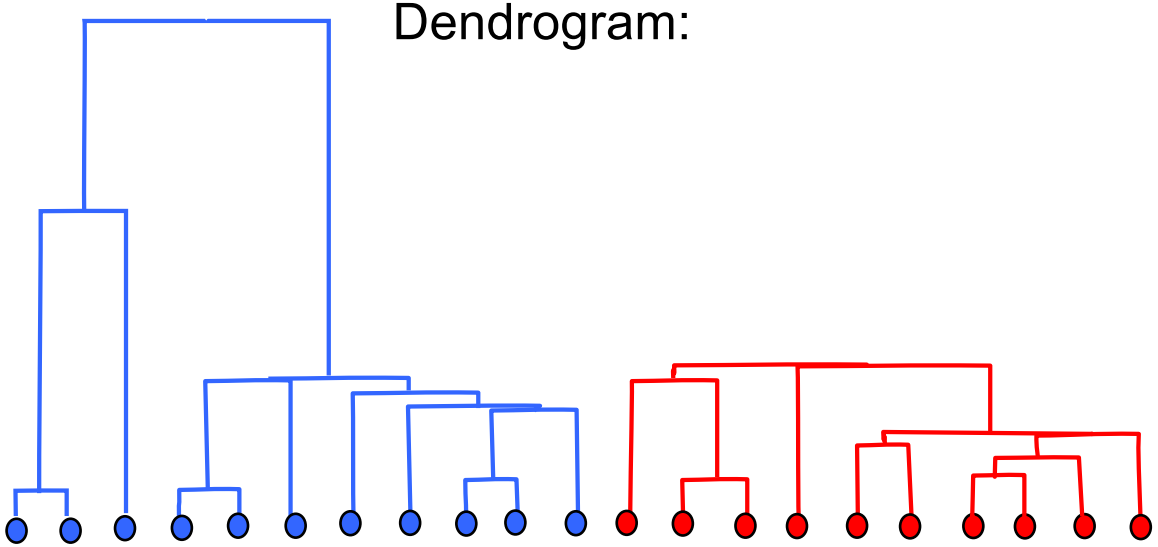
# Iteration m-2

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:



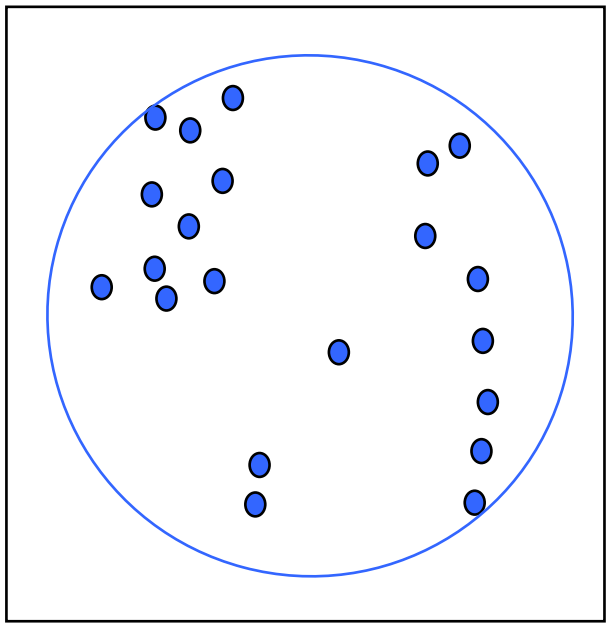
In mltools: “agglomerative”

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

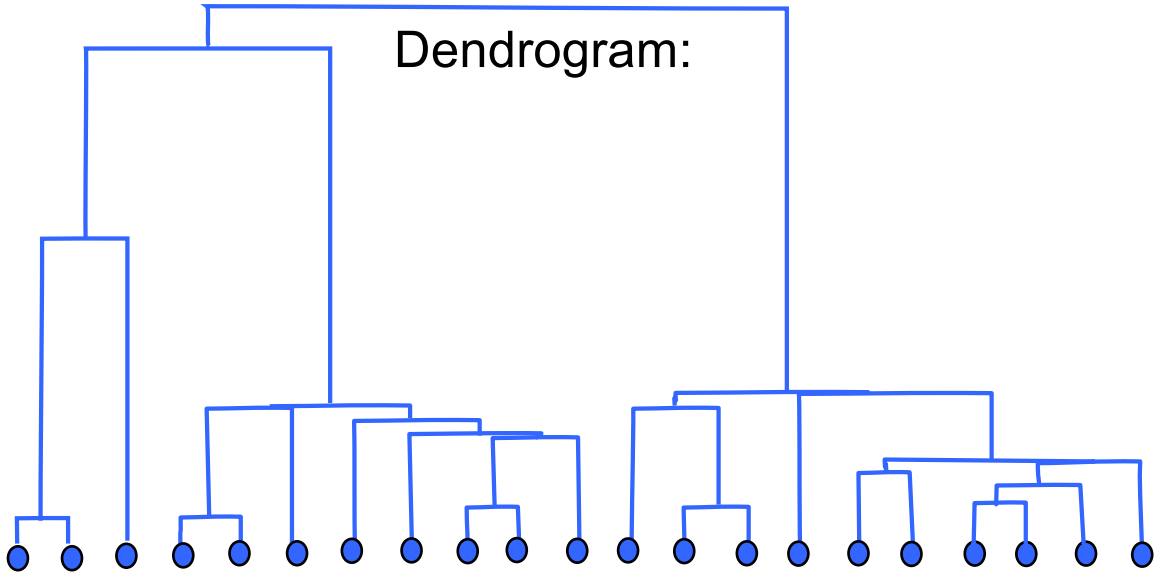
# Iteration m-1

Builds up a sequence of clusters (“hierarchical”)

Data:



Dendrogram:

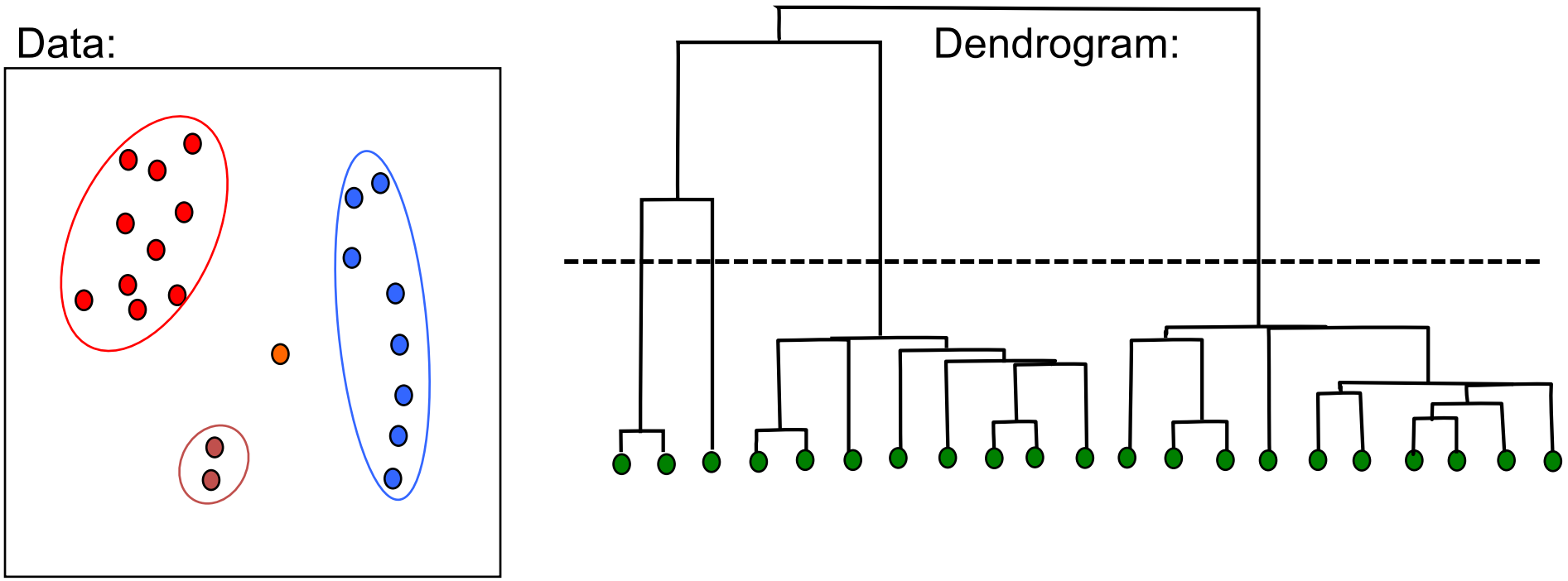


In mltools: “agglomerative”

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

# From dendrogram to clusters

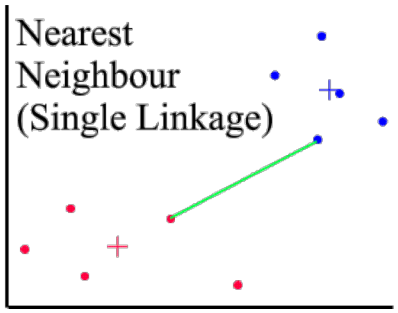
Given the sequence, can select a number of clusters or a dissimilarity threshold:




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

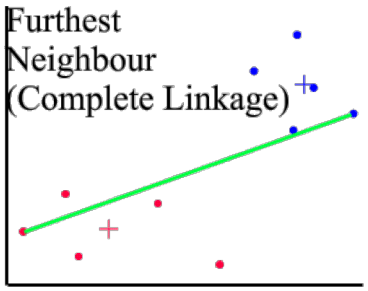
# Cluster distances

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




produces minimal spanning tree.

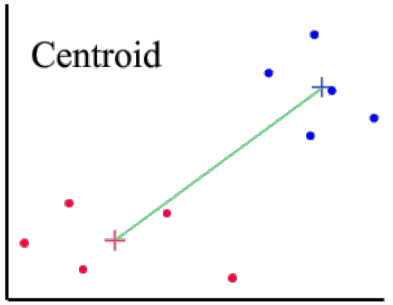
$$D_{\max}(C_i, C_j) = \max_{x \in C_i,\ y \in C_j} \|x - y\|^2$$




avoids elongated clusters.

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

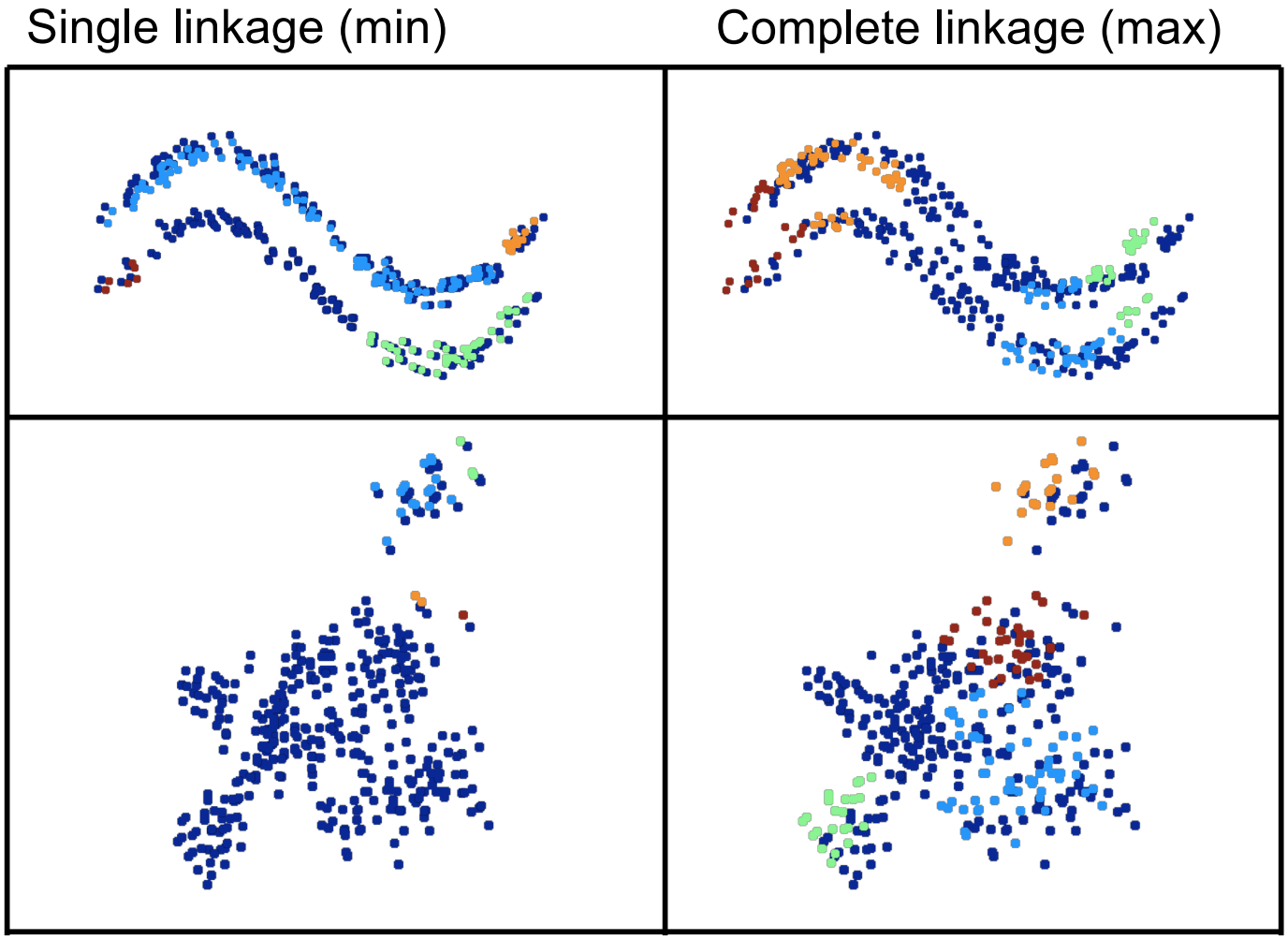
$$D_{\text{means}}(C_i, C_j) = \|\mu_i - \mu_j\|^2$$




**Constant time:** D(A,C) → D(A+B,C)
D(B,C) →

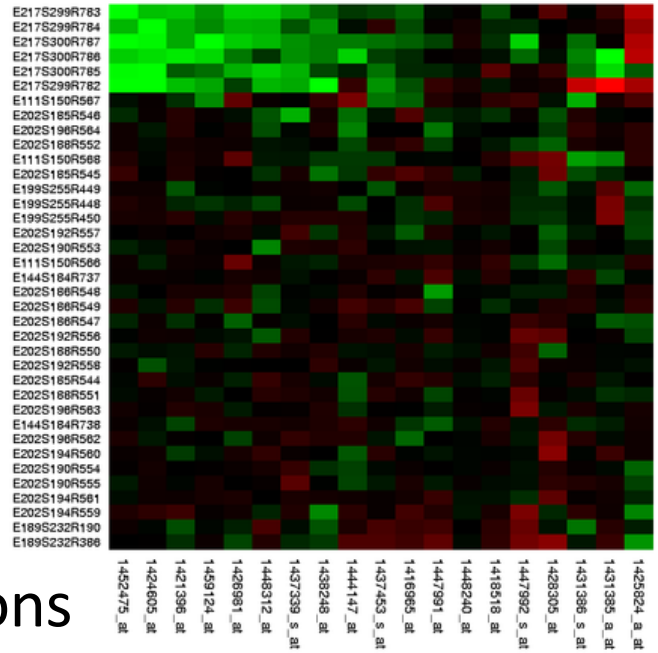
# Cluster distances

- 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



# Summary

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

# Machine Learning

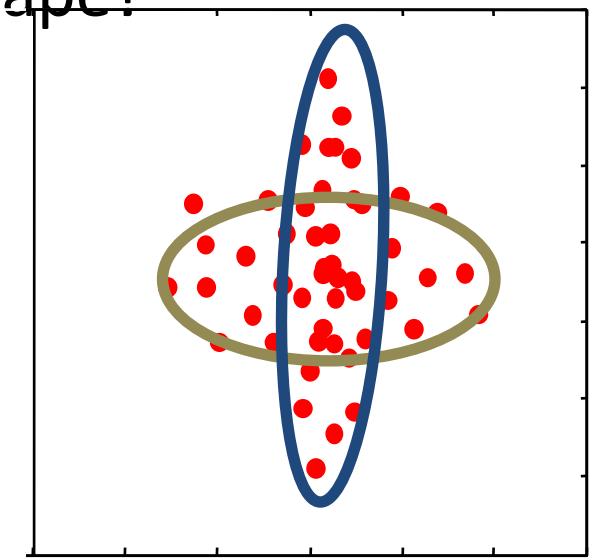
Clustering: K-Means

Clustering: Agglomerative

**Clustering: Gaussian Mixtures and EM**

# 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 $\mu_c$, variance $\Sigma_c$, "size" $\pi_c$
- 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 $\mu_c$, variance $\Sigma_c$, "size" $\pi_c$
- Probability distribution:

$$p(x) = \sum_c \pi_c \, \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$$

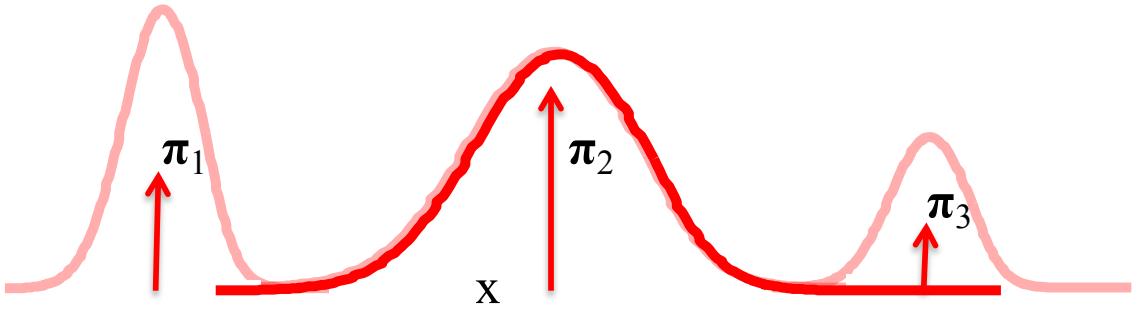
- Equivalent "latent variable" form:

Select a mixture component with probability $\pi$ $\quad p(z = c) = \pi_c$

Sample from that component's Gaussian $\quad p(x|z = c) = \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$

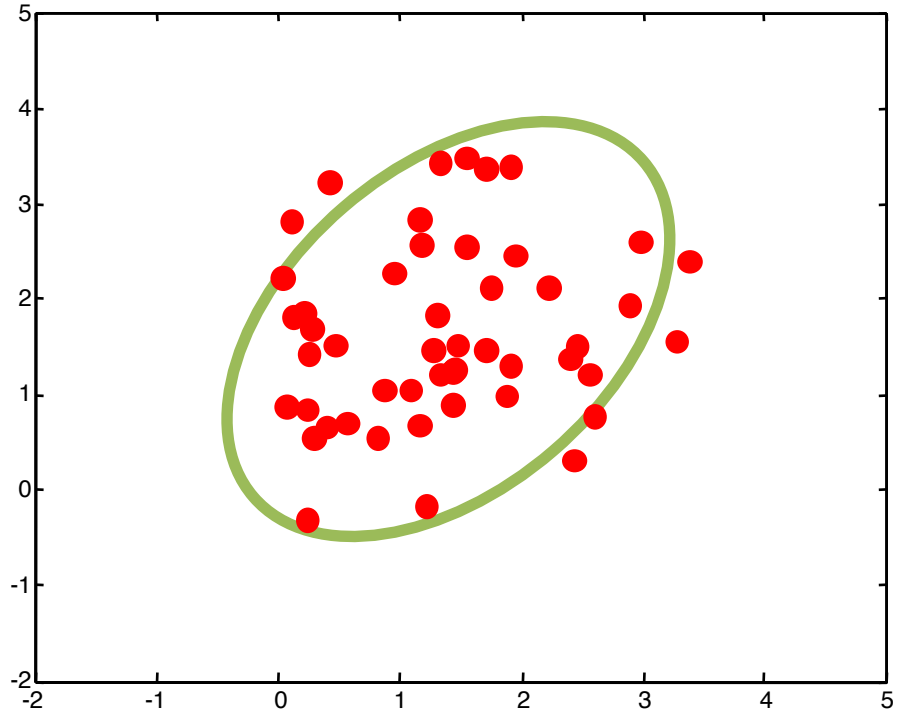
"Latent assignment" z:
we observe x, but z is *hidden*

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

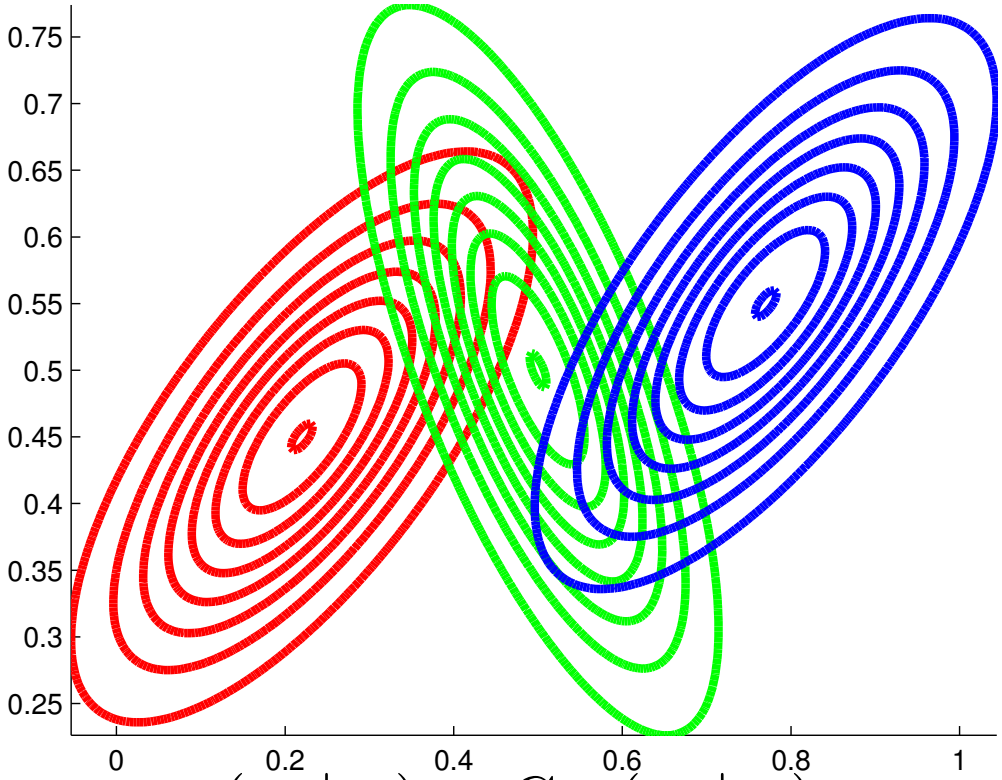
$$\hat{\mu} = \frac{1}{m}\sum_i x^{(i)}$$

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

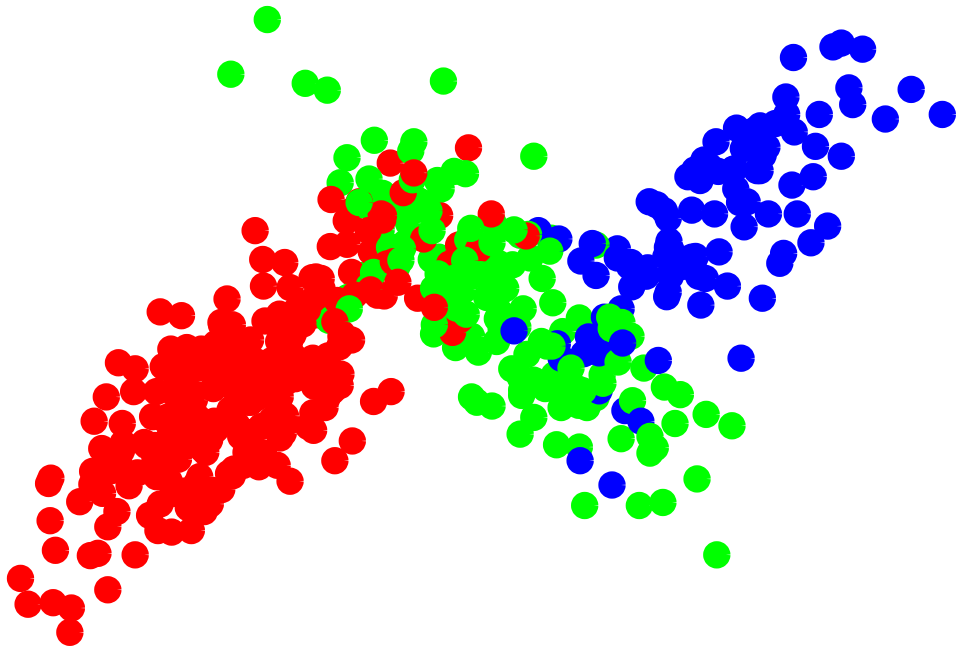
We'll model each cluster using one of these Gaussian "bells"…

# Gaussian Mixture Models

***Mixture of K=3 Gaussian Distributions in 2D***




***Example Data with True Cluster Labels***



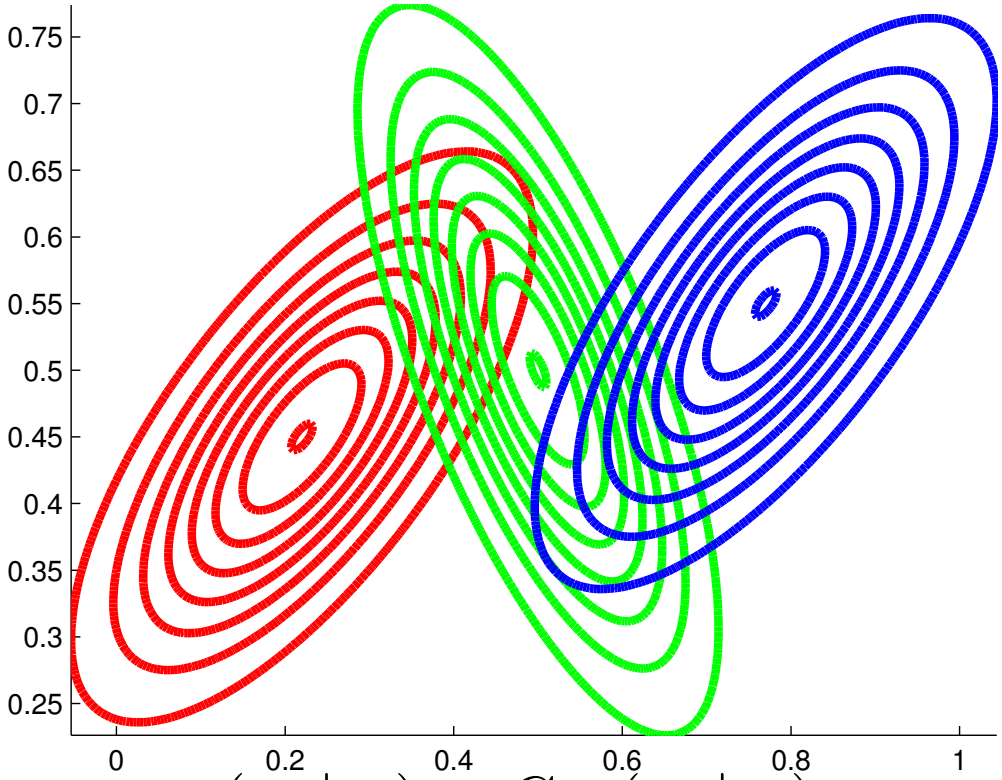
$$p(z_i \mid \pi) = \mathrm{Cat}(z_i \mid \pi)$$

$$p(x_i \mid z_i, \mu, \Sigma) = \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

$$p(x_i \mid \pi, \mu, \Sigma) = \sum_{z_i=1}^{K} \pi_{z_i} \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

# Gaussian Mixture Models

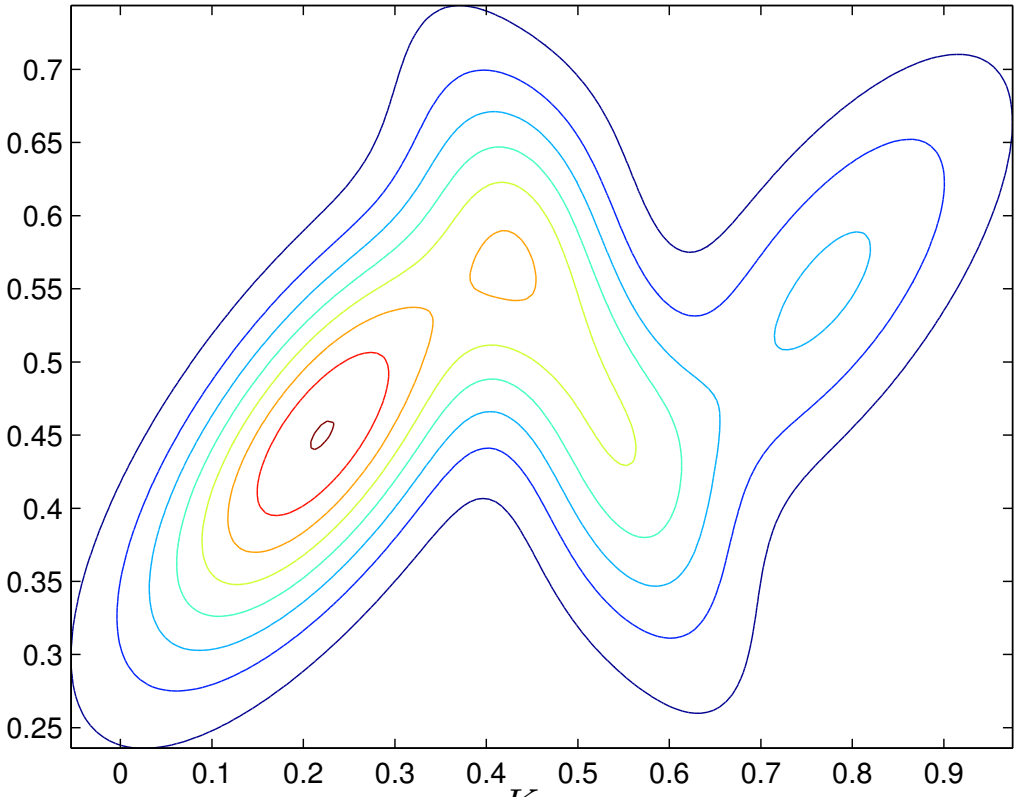
***Mixture of K=3 Gaussian Distributions in 2D***




$$p(z_i \mid \pi) = \mathrm{Cat}(z_i \mid \pi)$$

$$p(x_i \mid z_i, \mu, \Sigma) = \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

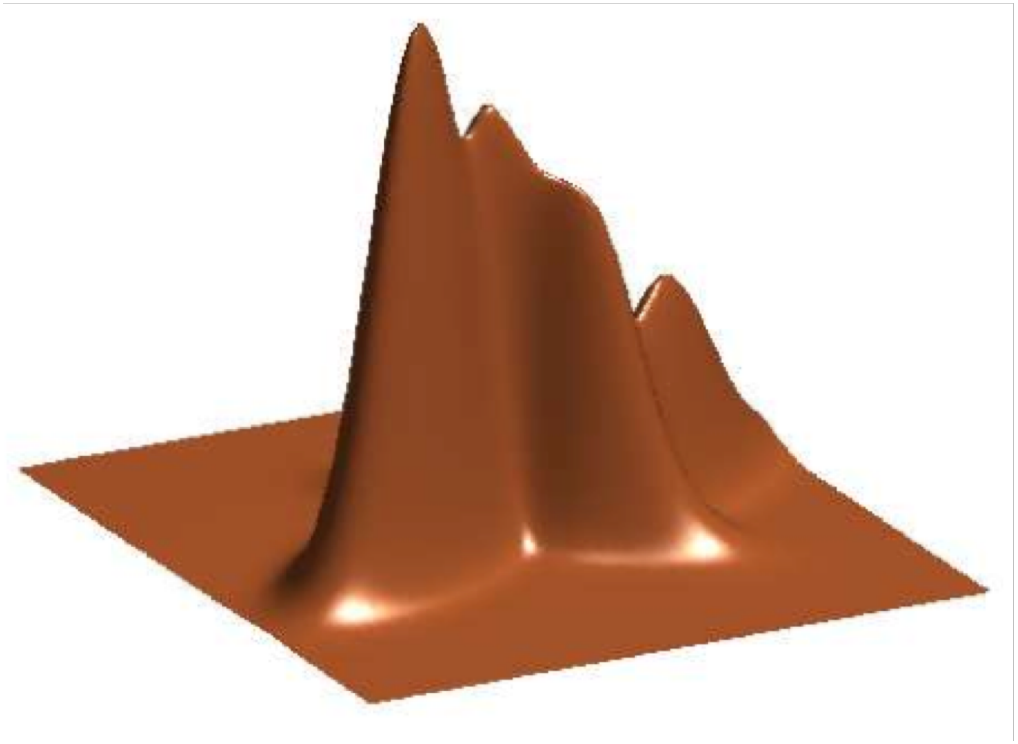
***Contour Plot of Density, Marginalizing Clusters***



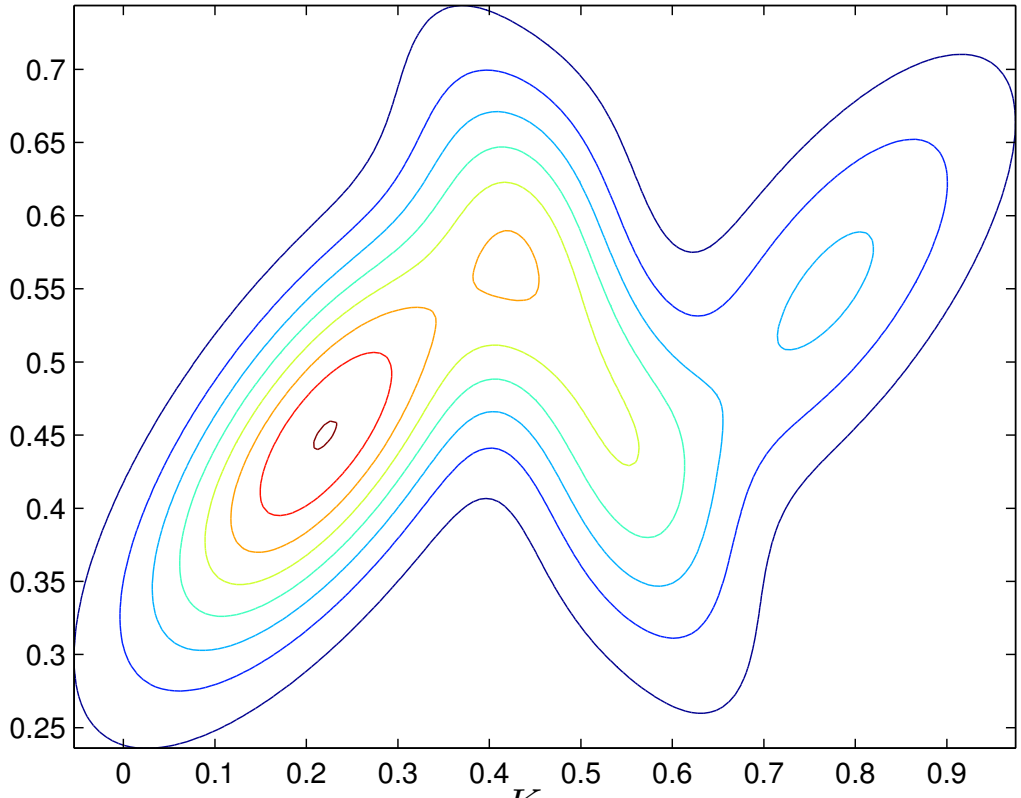

$$p(x_i \mid \pi, \mu, \Sigma) = \sum_{z_i=1}^{K} \pi_{z_i} \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

# Gaussian Mixture Models

***Surface Plot of Density, Marginalizing Clusters***



***Contour Plot of Density, Marginalizing Clusters***




$$p(z_i \mid \pi) = \mathrm{Cat}(z_i \mid \pi)$$

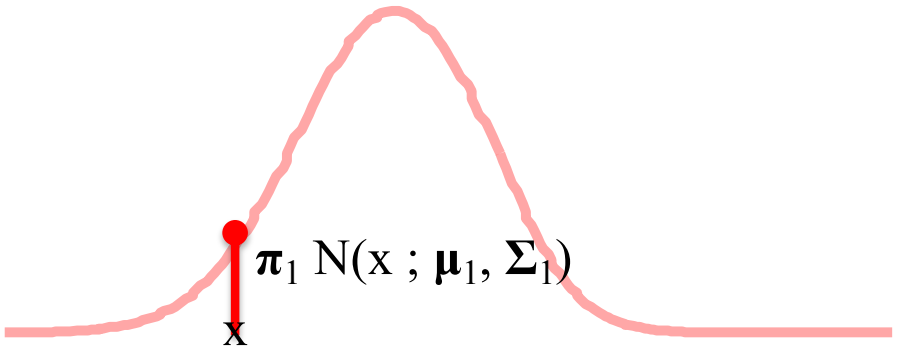
$$p(x_i \mid z_i, \mu, \Sigma) = \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

$$p(x_i \mid \pi, \mu, \Sigma) = \sum_{z_i=1}^{K} \pi_{z_i} \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

# EM Algorithm: E-step

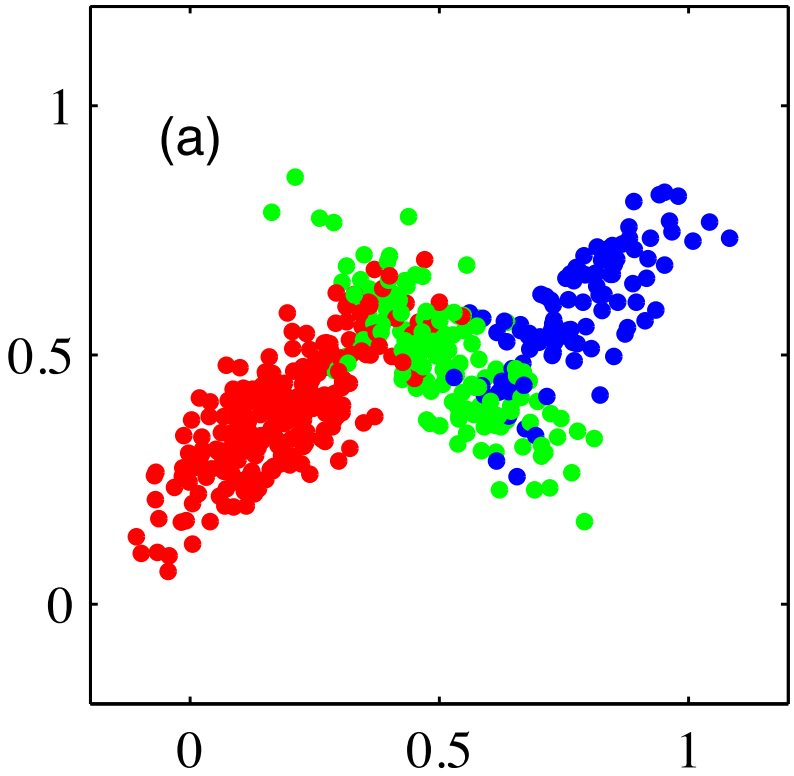
- Start with clusters: Mean $\mu_c$, Covariance $\Sigma_c$, "size" $\pi_c$
- E-step ("Expectation")
  - For each datum (example) $x_i$,
  - Compute "$r_{ic}$", the probability that it belongs to cluster c
    - Compute its probability under model c
    - Normalize to sum to one (over clusters c)

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

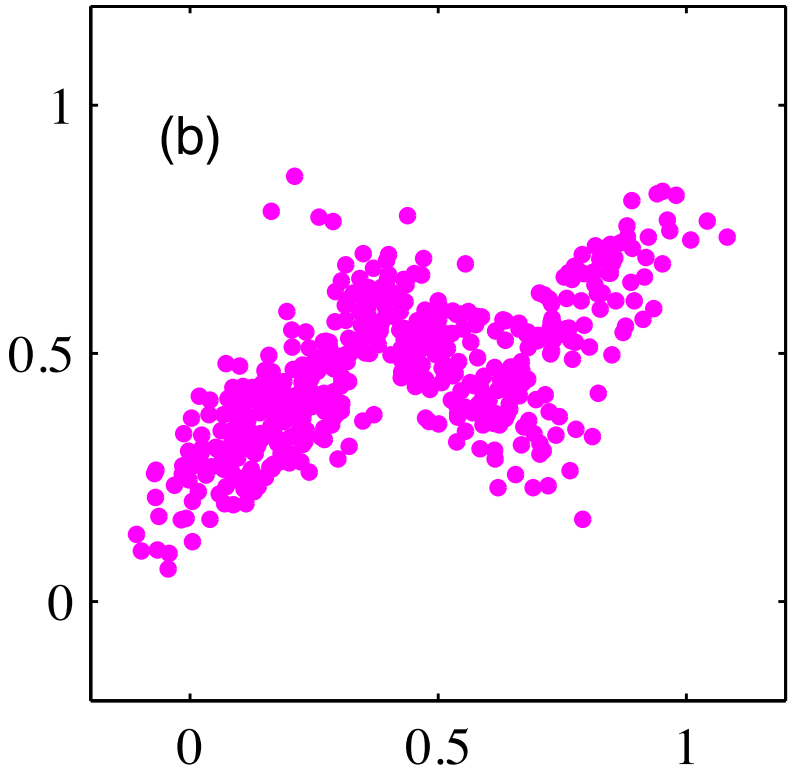



  - If xi is very likely under the $c^{th}$ Gaussian, it gets high weight
  - Denominator just makes r's sum to one

# Learning Gaussian Mixtures

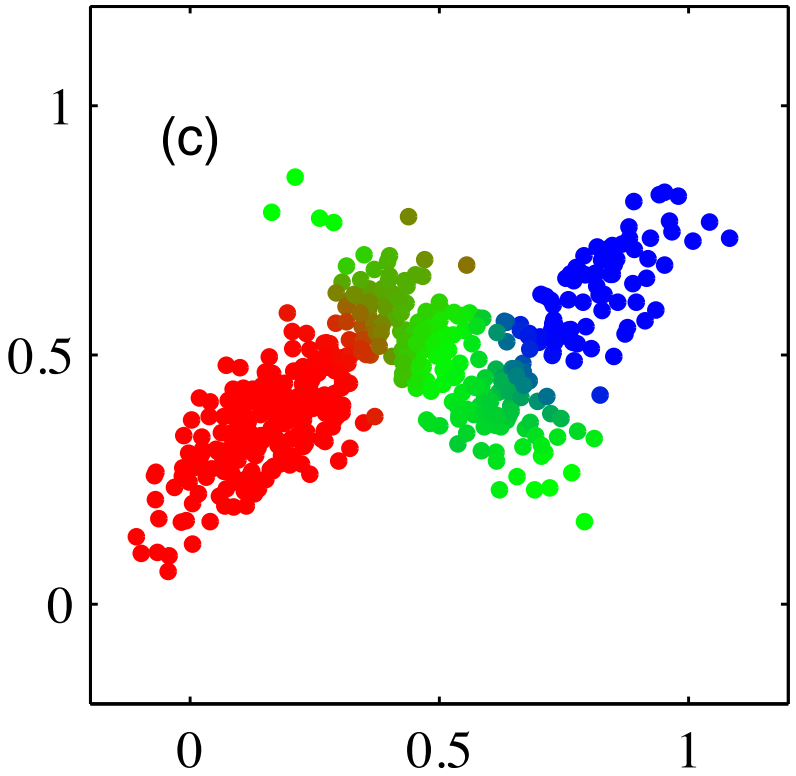



***"Complete Data" Labeled by True Cluster Assignments***

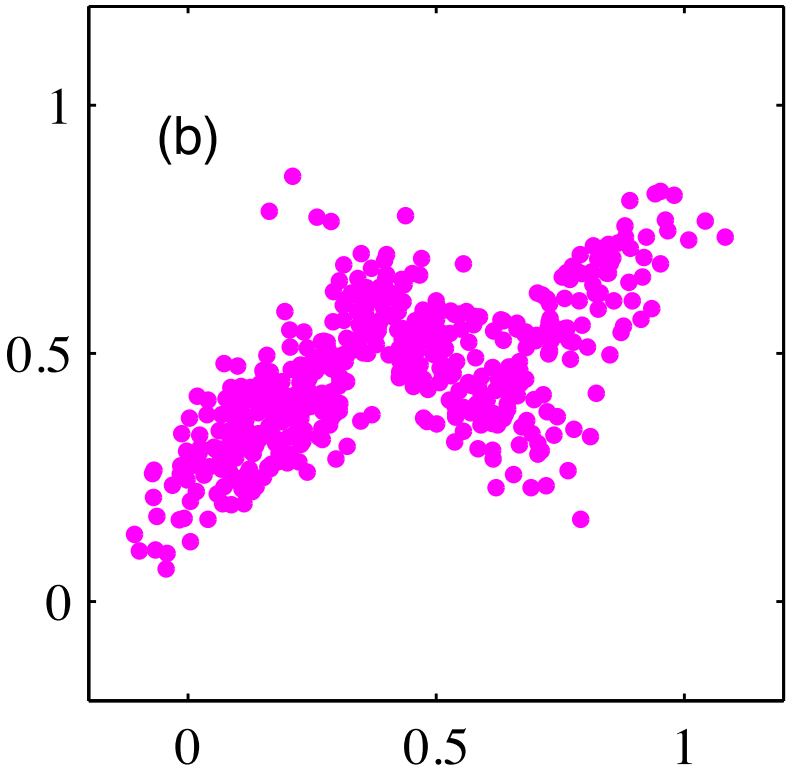



***"Incomplete Data": Points to be Clustered***

# Inference Given Cluster Parameters




***Posterior Probabilities of Assignment to Each Cluster***




***"Incomplete Data": Points to be Clustered***

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

# EM Algorithm: M-step

- Start with assignment probabilities $r_{ic}$
- Update parameters: mean $\mu_c$, Covariance $\Sigma_c$, "size" $\pi_c$
- M-step ("Maximization")
  - For each cluster (Gaussian) z = c,
  - Update its parameters using the (weighted) data points

$$m_c = \sum_i r_{ic}$$

Total responsibility allocated to cluster c

$$\pi_c = \frac{m_c}{m}$$

Fraction of total assigned to cluster c

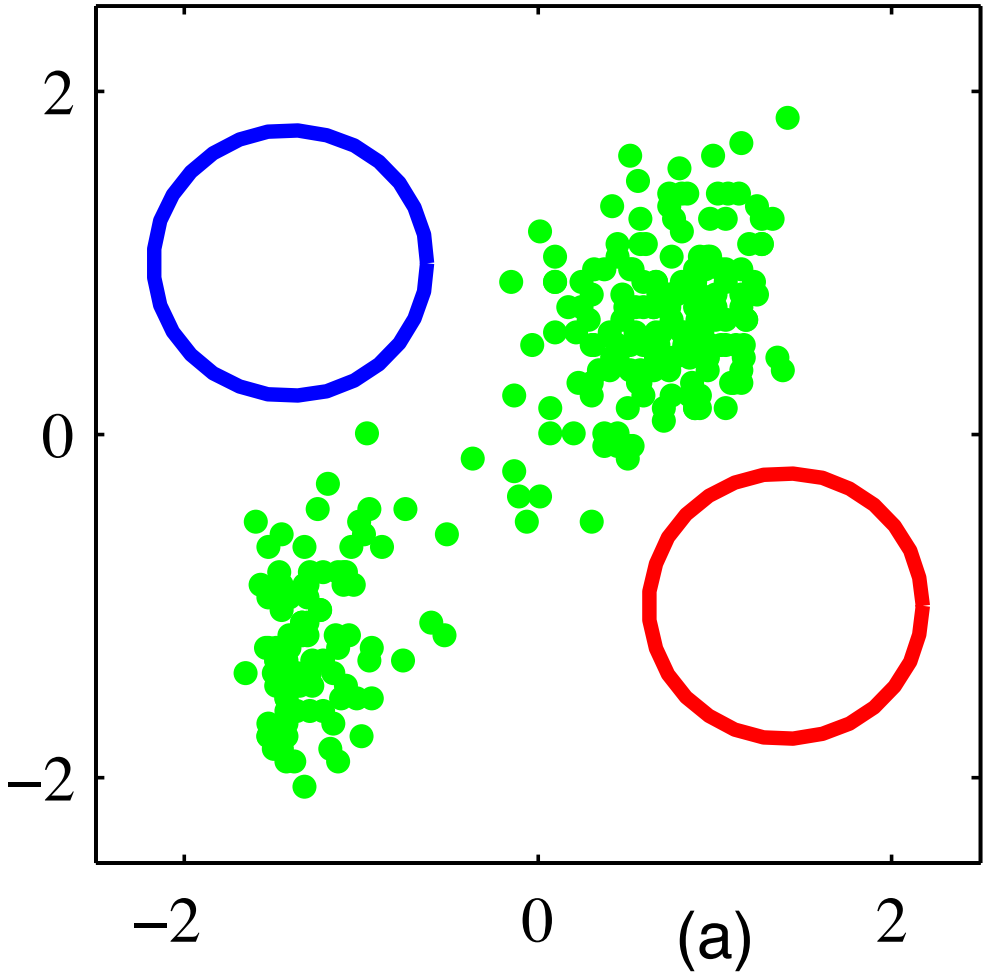
$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)}$$

Weighted mean of assigned data

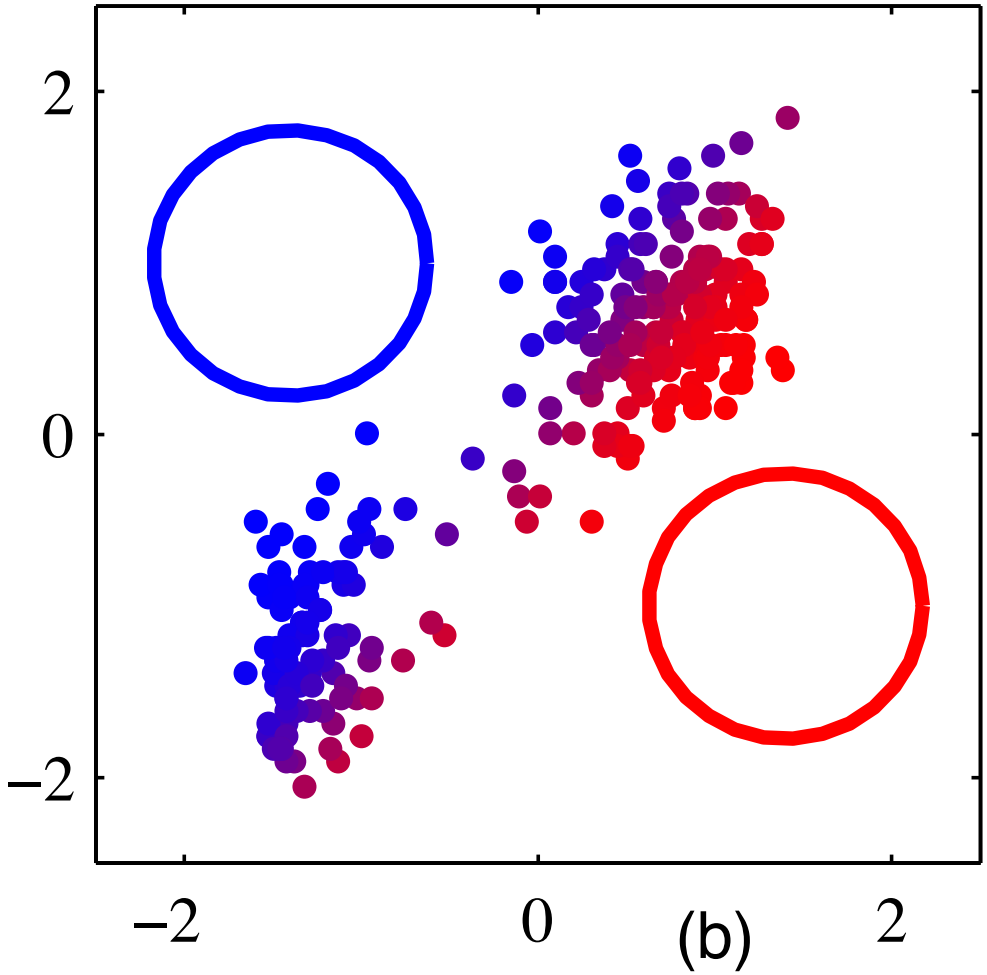
$$\Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

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

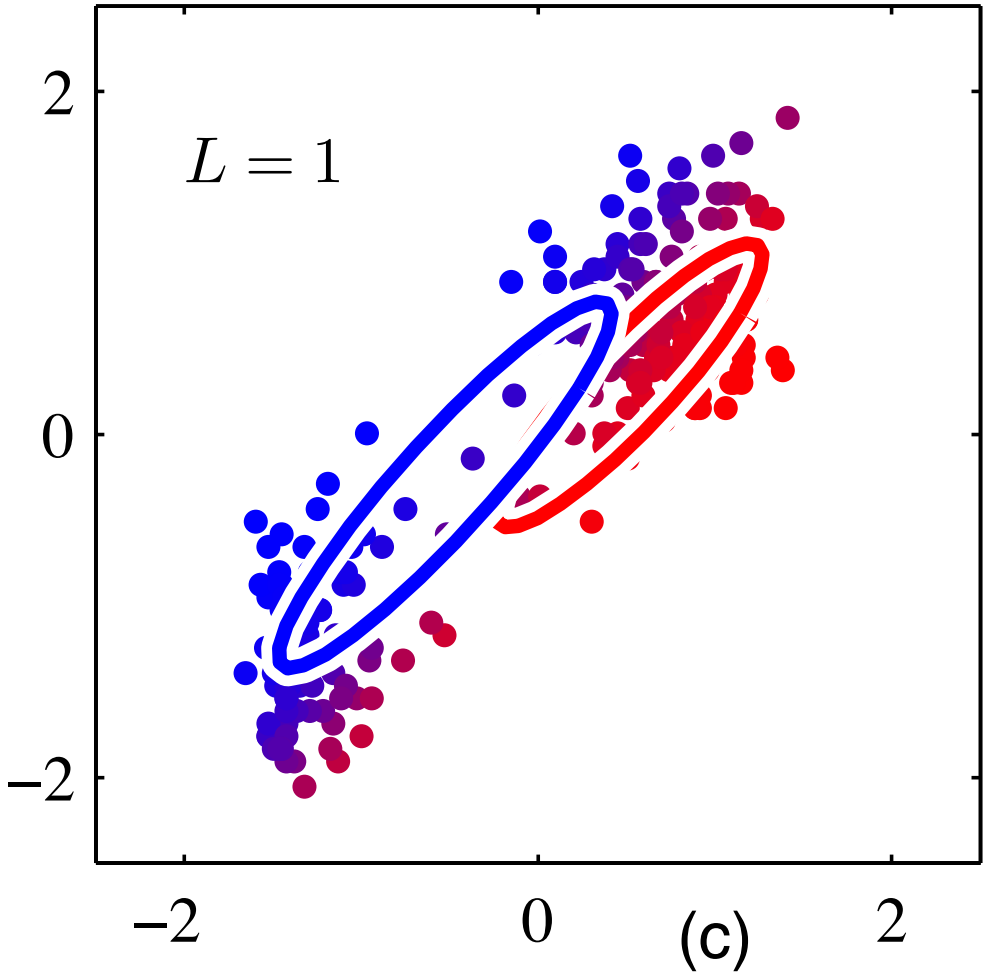
# EM Algorithm: 2D Example



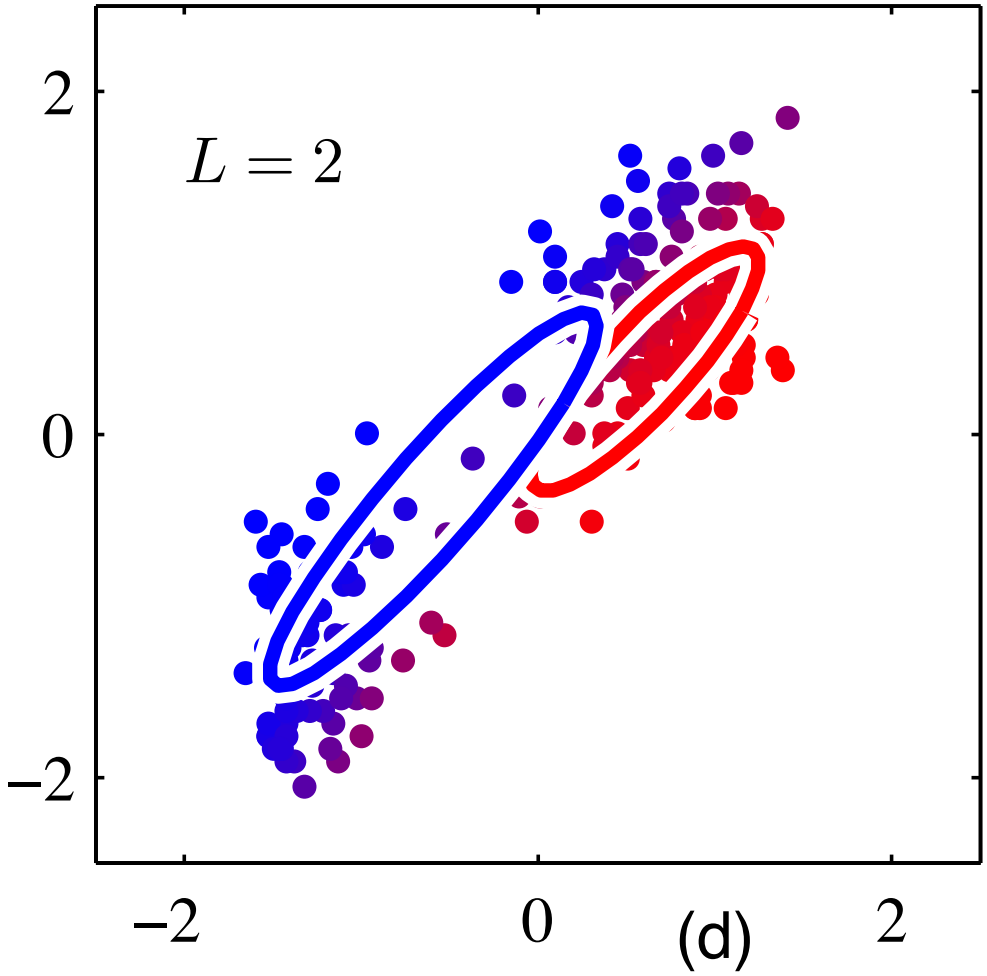
# EM Algorithm: 2D Example



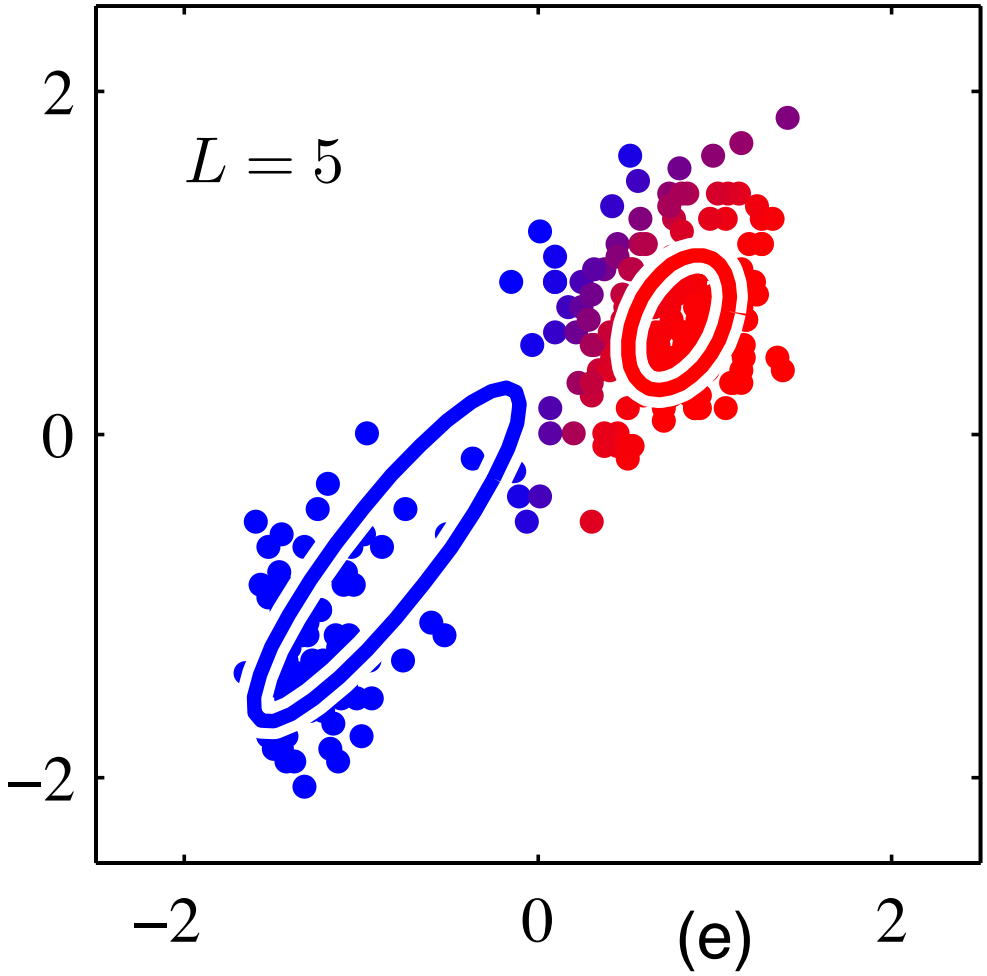
# EM Algorithm: 2D Example



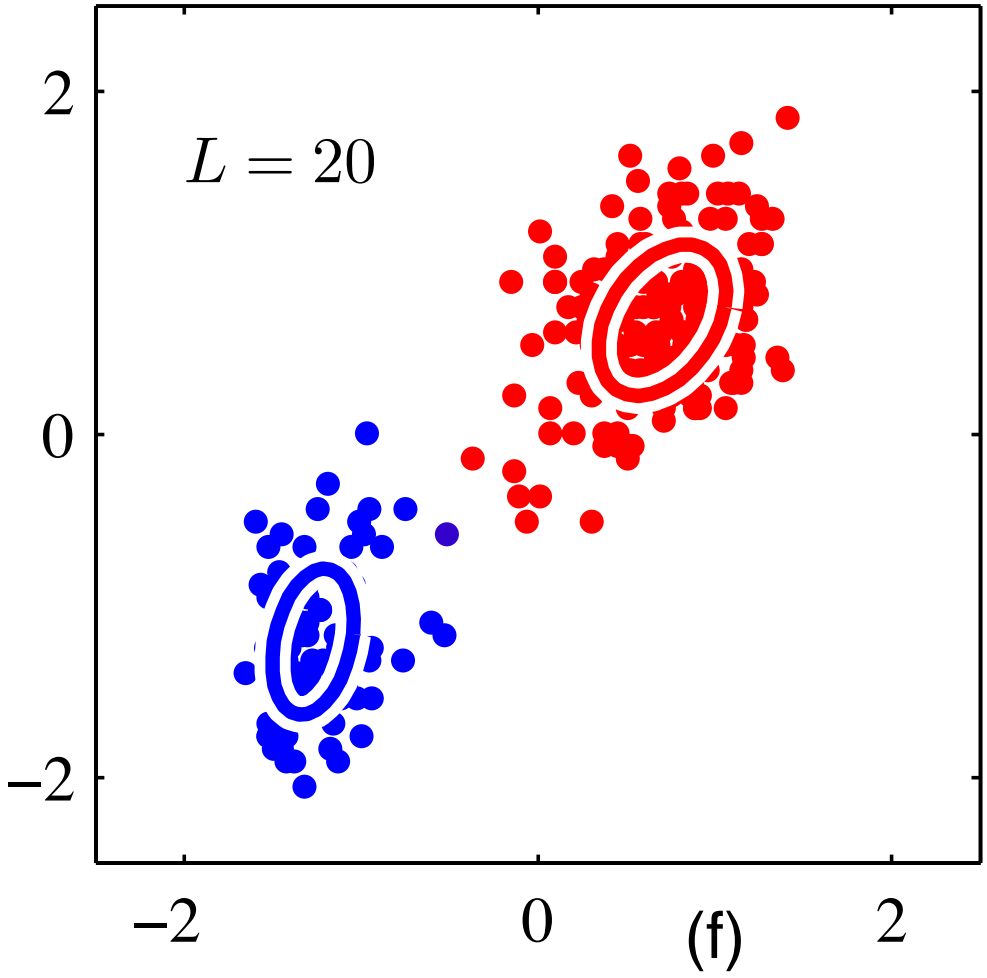
# EM Algorithm: 2D Example



# EM Algorithm: 2D Example



# EM Algorithm: 2D Example

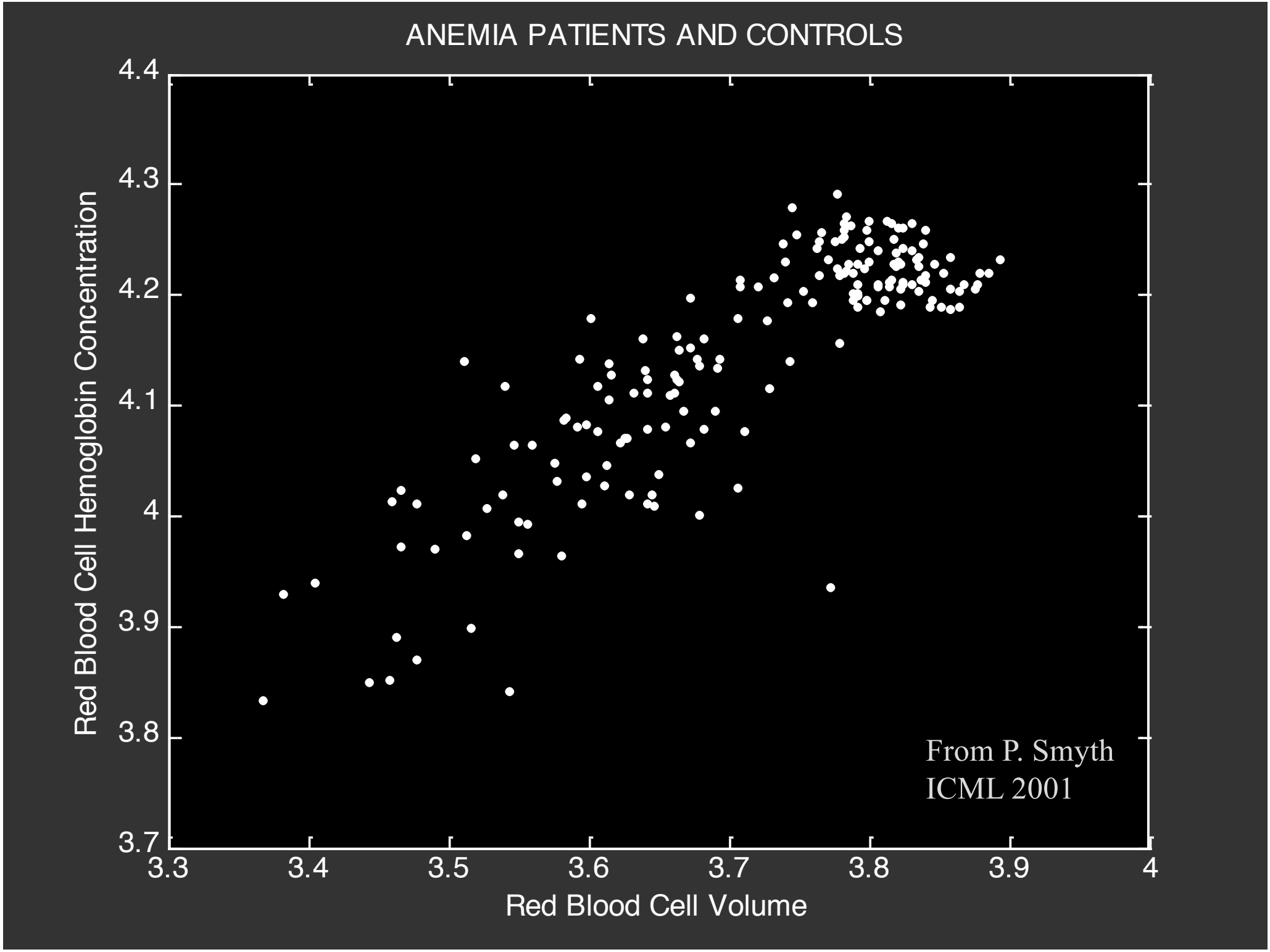


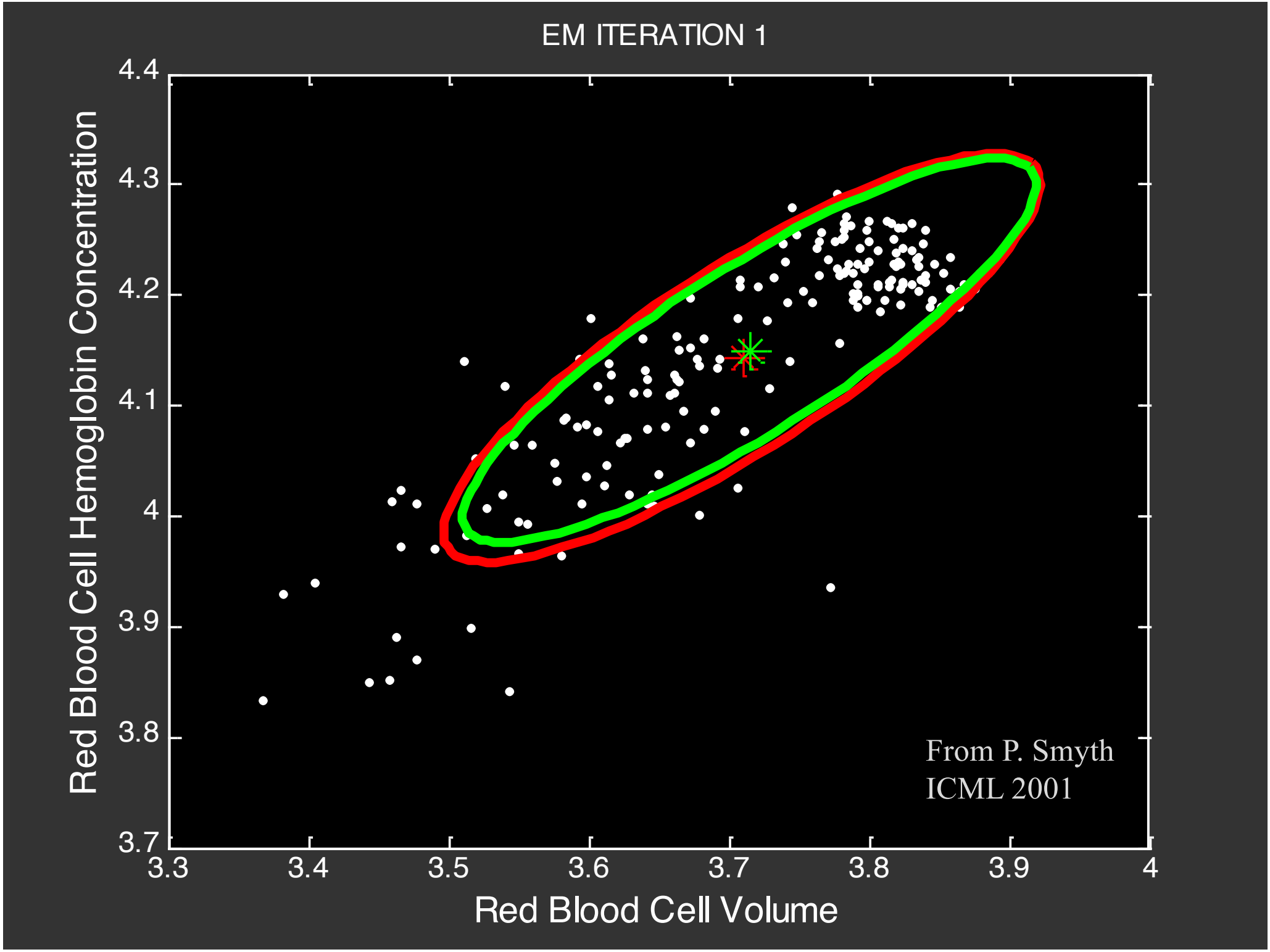
# Expectation-Maximization

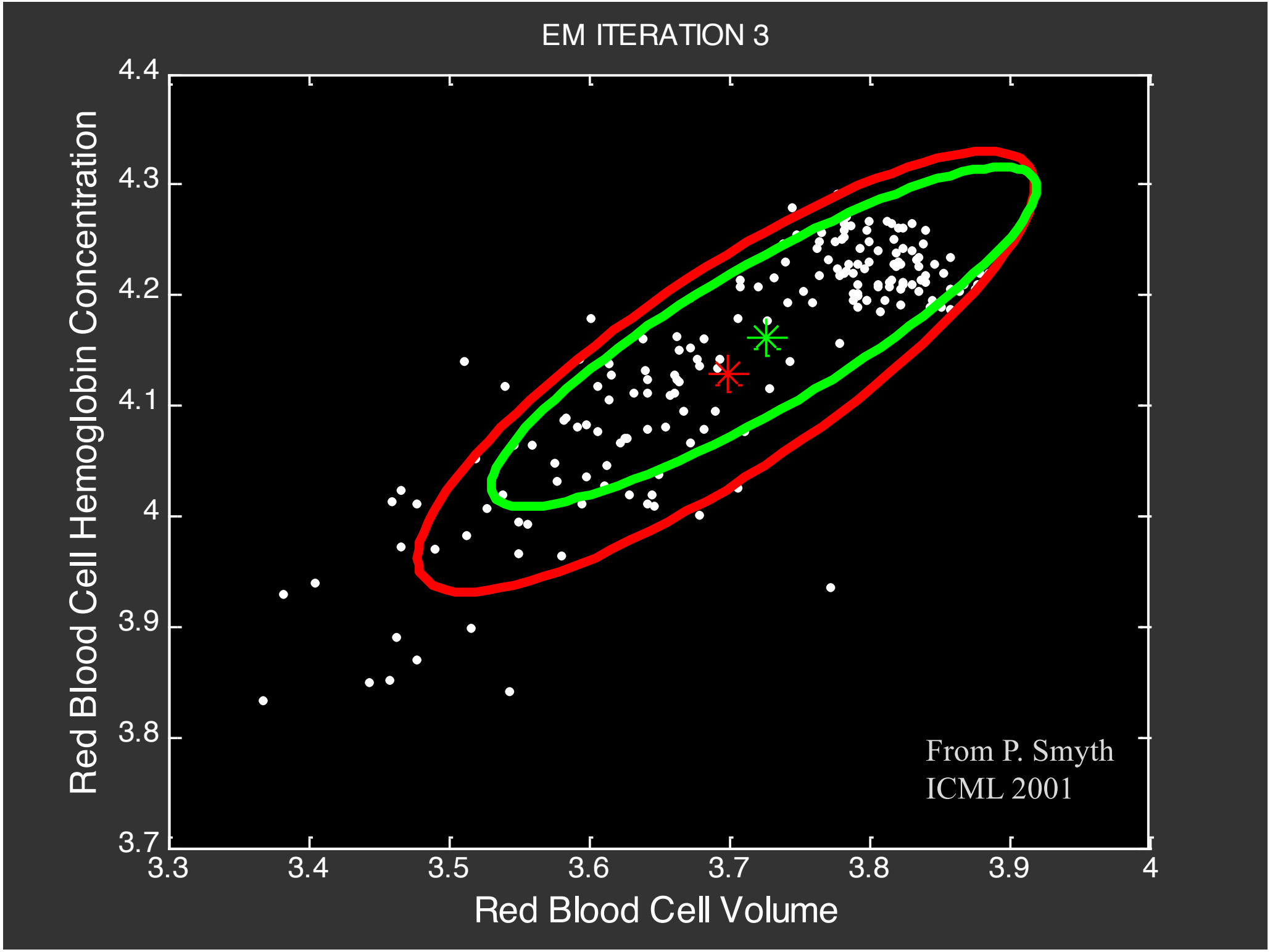
- Each step increases the log-likelihood of our model

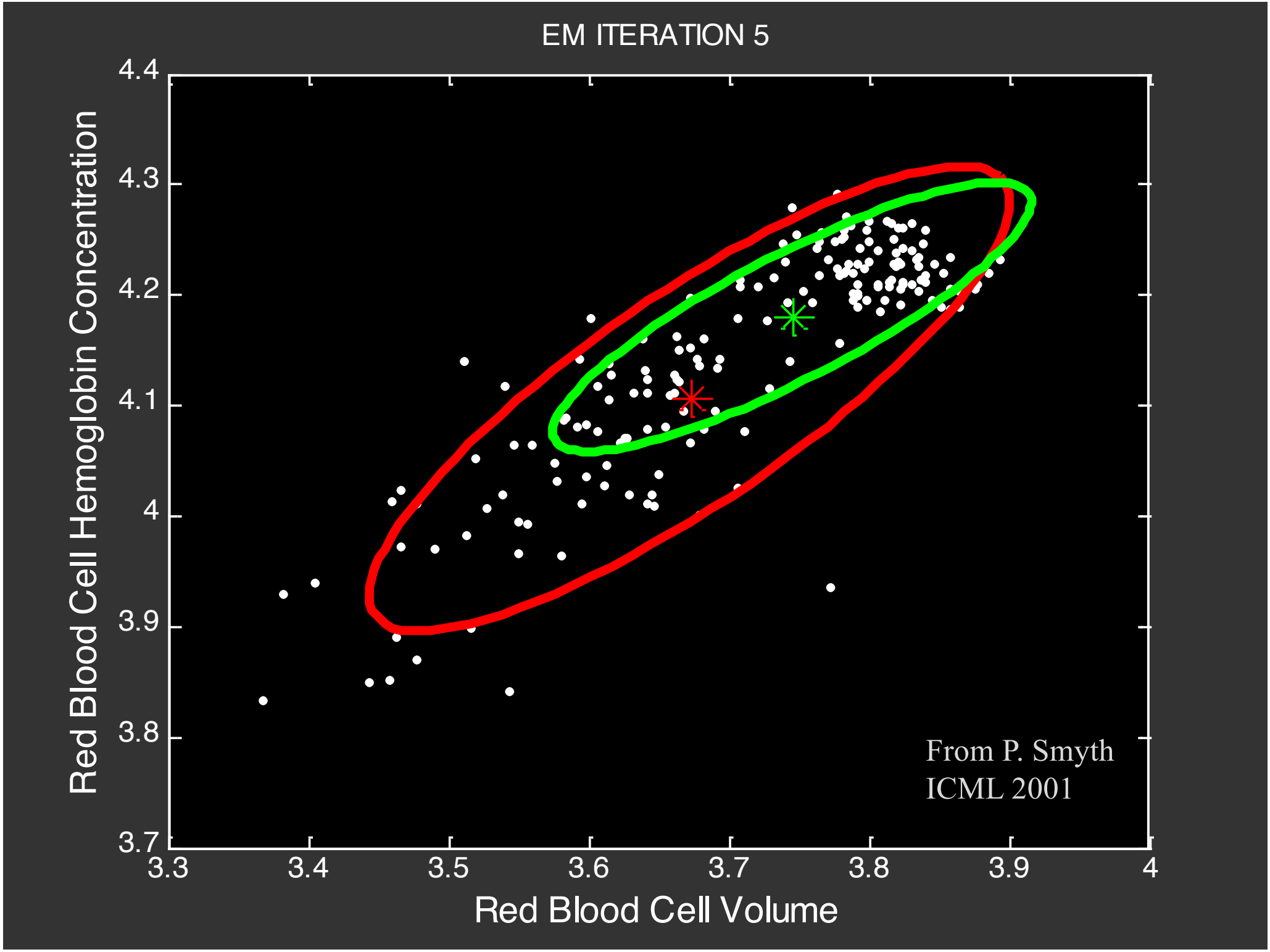
$$\log p(\underline{X}) = \sum_i \log \left[ \sum_c \pi_c \, \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c) \right]$$

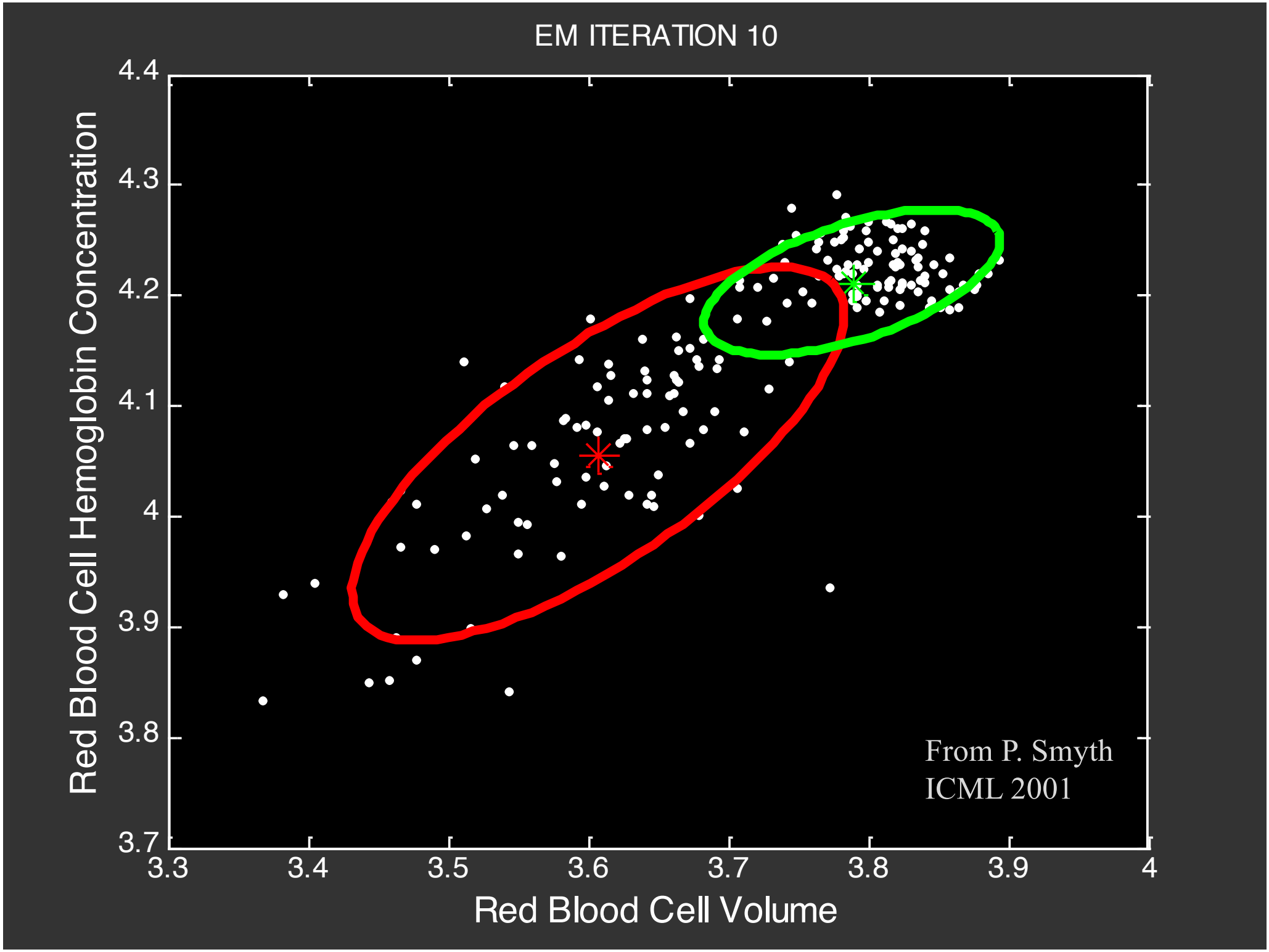
  (we won't derive this here, though)
- Iterate until convergence
  - Convergence guaranteed – another ascent method
  - Local optima: initialization often important
- What should we do
  - If we want to choose a single cluster for an "answer"?
  - With new data we didn't see during training?
- Choosing 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')

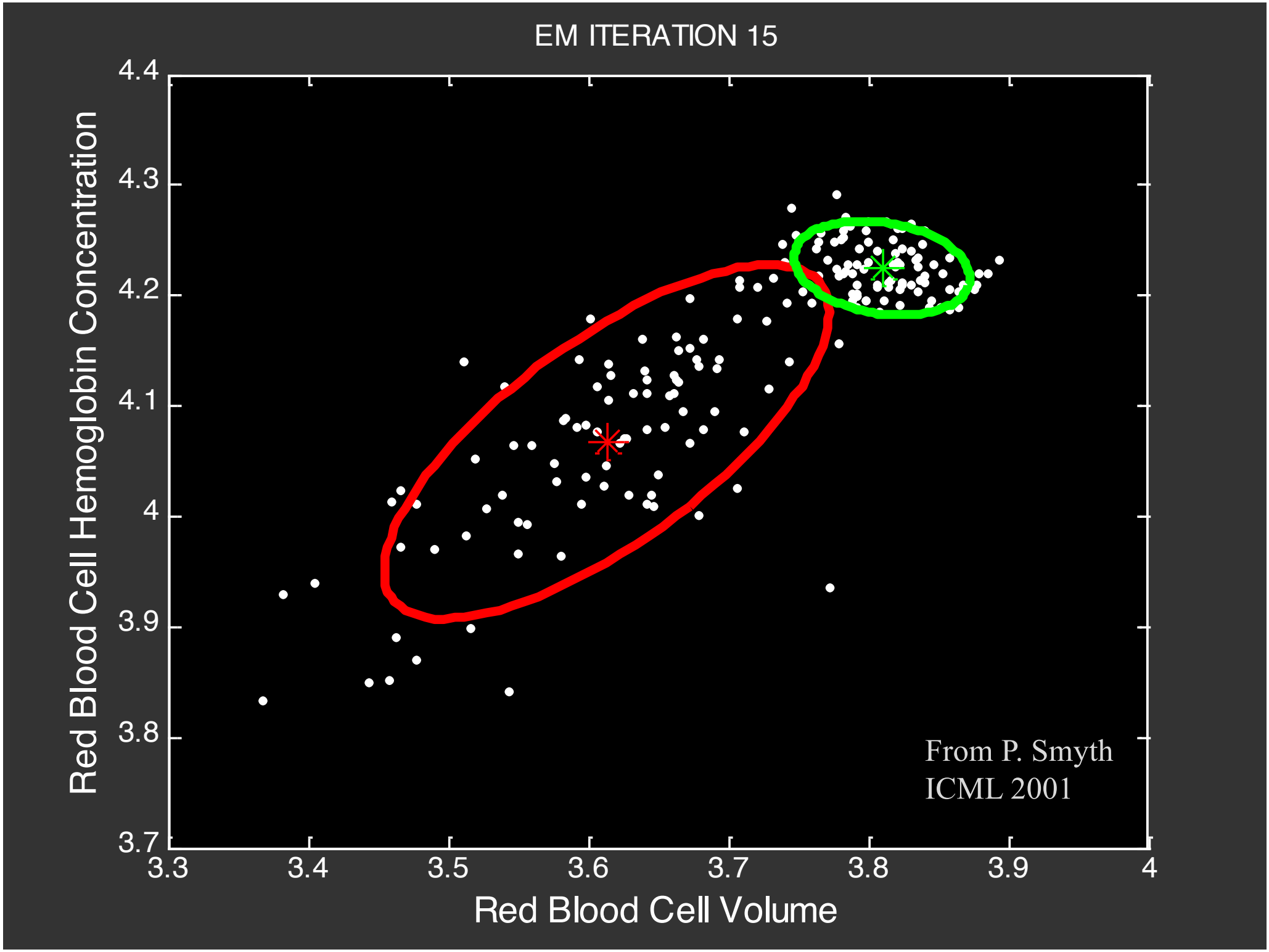

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001

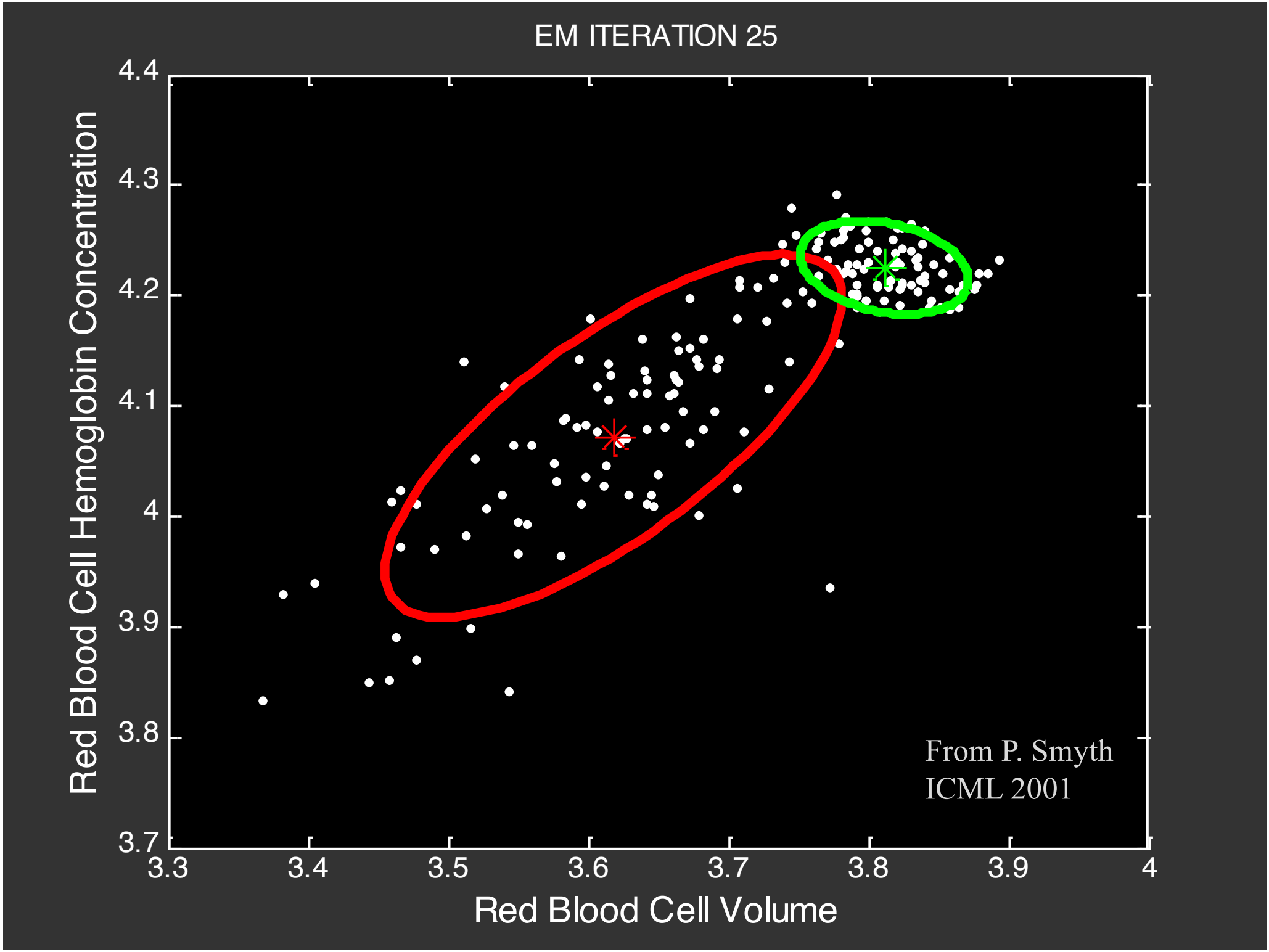

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001

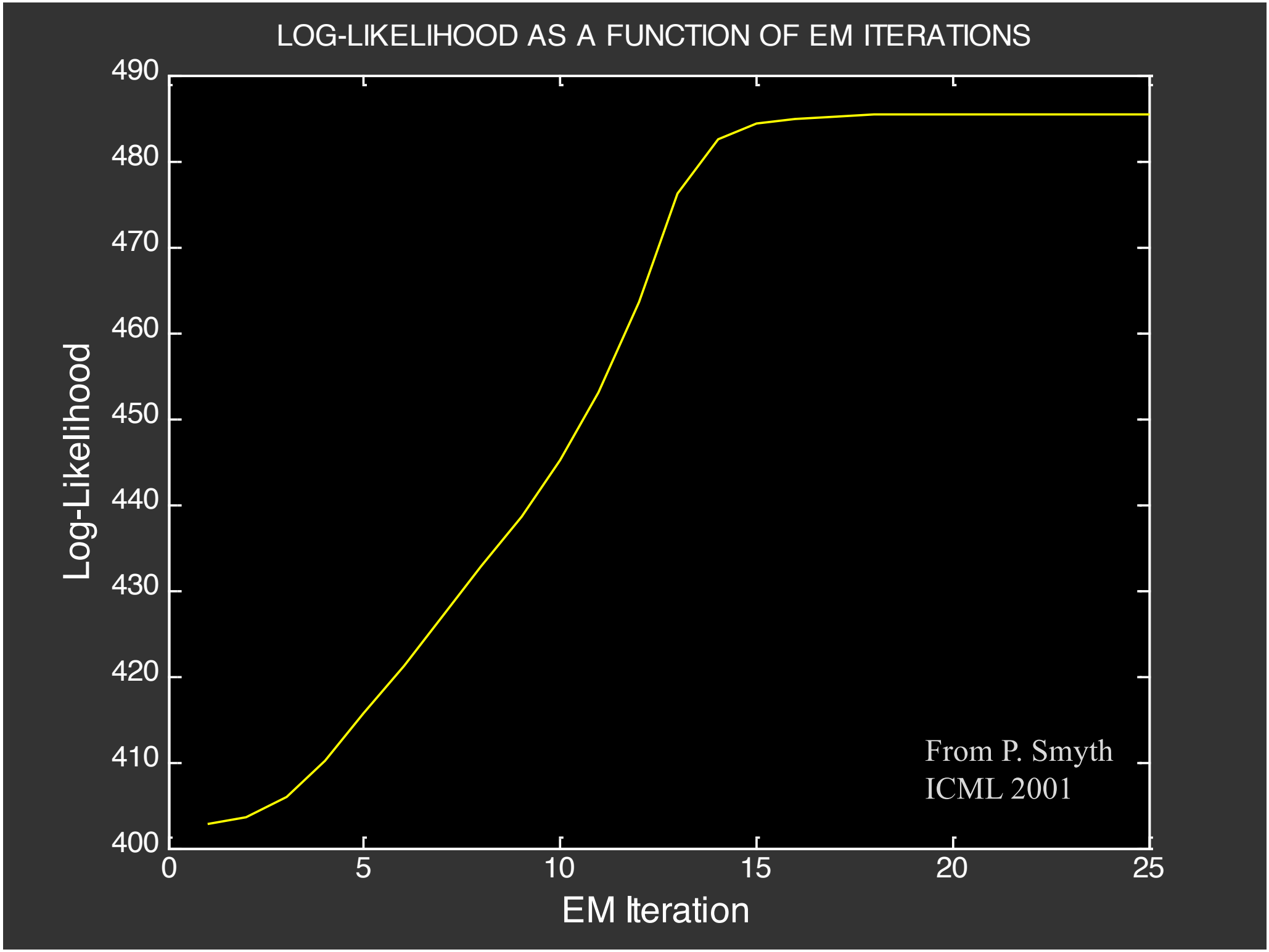

EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001


EM ITERATION 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
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 15
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

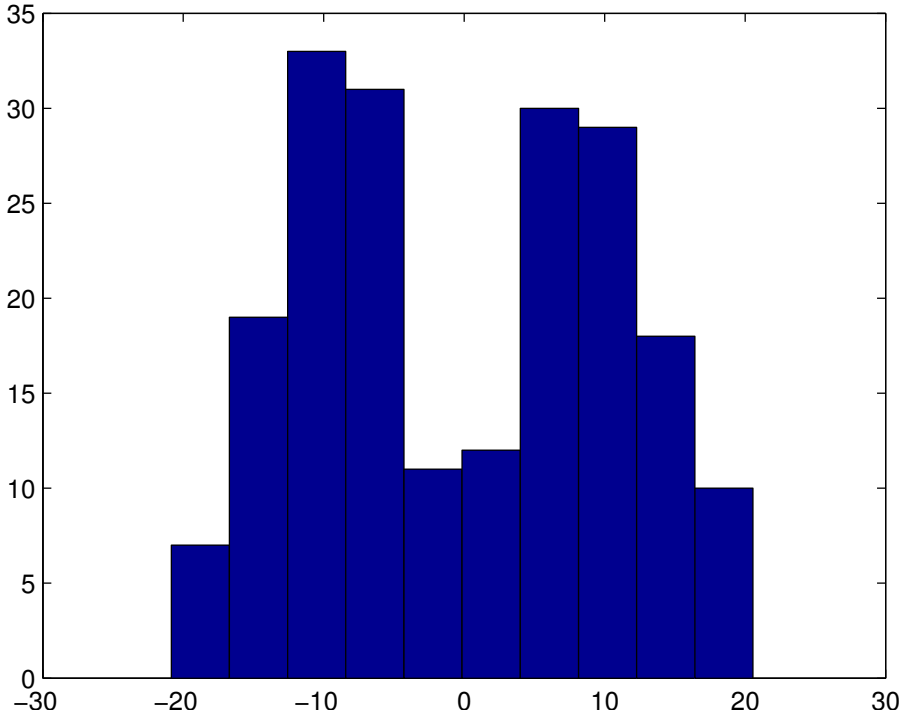

EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001


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

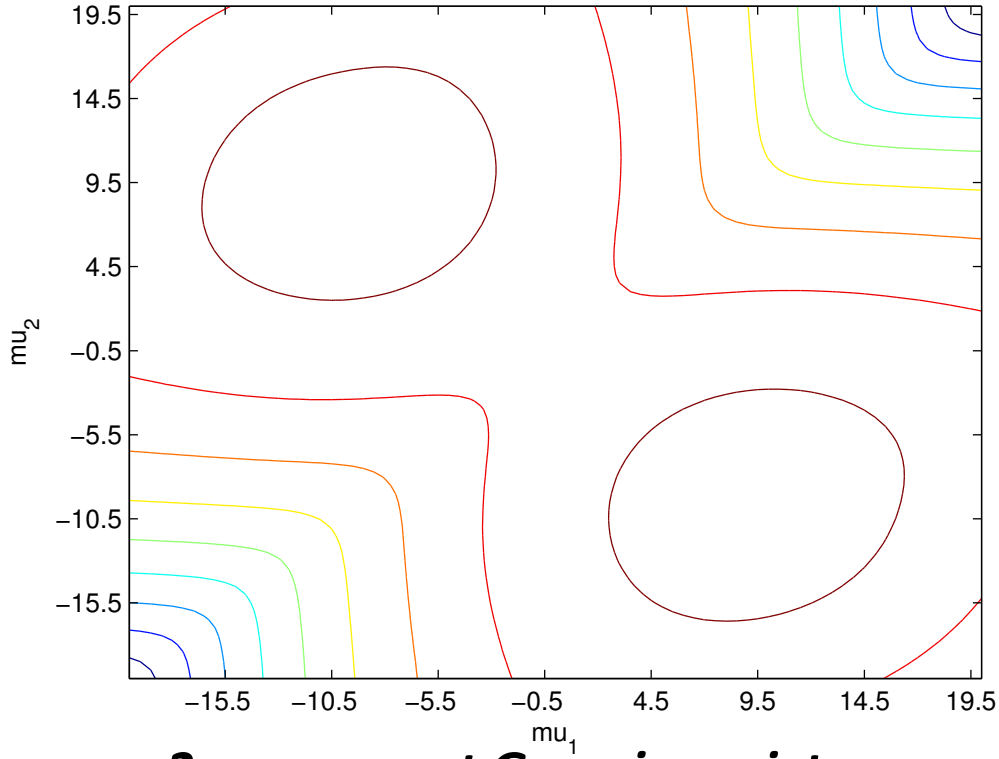
# Demo Time!

https://lukapopijac.github.io/gaussian-mixture-model/

# Label Switching in Mixture Models




***Histogram of 200 samples from a mixture of two 1D Gaussians***




***2-component Gaussian mixture likelihood surface as function of means, for fixed variances***

***Likelihood is non-convex, learning biased by initialization.***

# EM for Diagonal Gaussian Mixtures

$$p(z_i \mid \pi) = \text{Cat}(z_i \mid \pi)$$
$$p(x_i \mid \mu_k) = \text{Norm}(x_i \mid \mu_k, \sigma^2 I)$$
$$\pi_k = \frac{1}{K}$$

***Constrain clusters to be equally common, and have same circular shape.***

- **Initialization: Randomly select starting parameters** $\mu^{(0)}$
- **E-Step: Given parameters, assignments probabilities equal**

$$r_{ik} = p(z_i = k \mid x_i, \mu) = \frac{\text{Norm}(x_i \mid \mu_k, \sigma^2 I)}{\sum_{\ell=1}^{K} \text{Norm}(x_i \mid \mu_\ell, \sigma^2 I)}$$

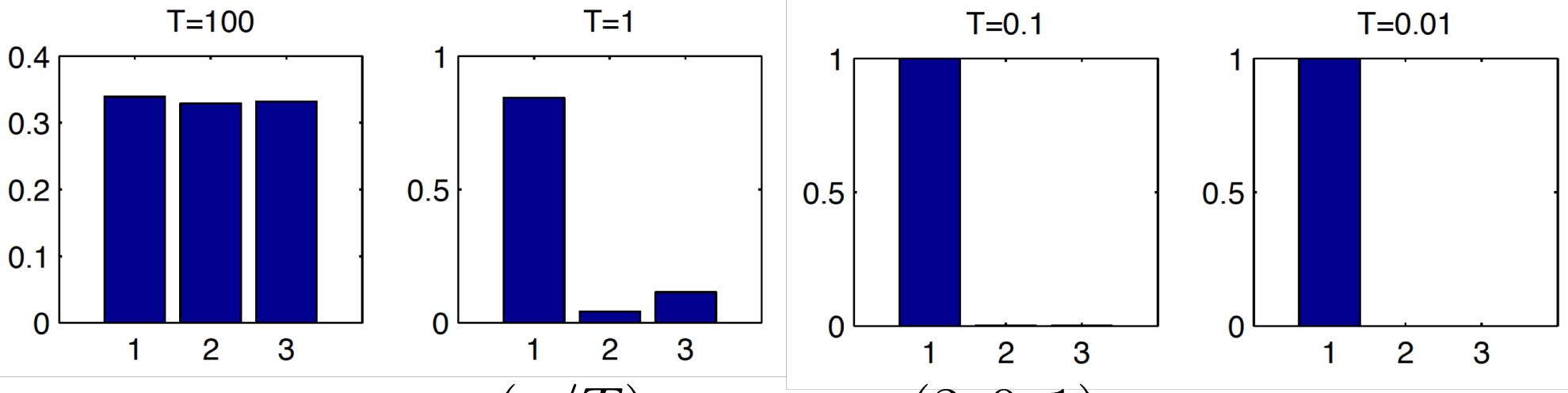
- **M-Step: Given posterior distributions, find likely parameters**

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik} x_i \qquad N_k = \sum_{i=1}^{N} r_{ik}$$

***M-Step update is identical to the K-Means algorithm.***

# Softmax Function

*"uniform"* ← *probabilities* → *"max"*




$$s_k(z/T), \qquad z = (3, 0, 1)$$

$$s_k(z) = \frac{\exp(z_k)}{\sum_{\ell=1}^{K} \exp(z_\ell)}, \qquad k = 1, \ldots, K.$$

$$z_k \in \mathbb{R}, \quad s_k(z) > 0, \quad \sum_{k=1}^{K} s_k(z) = 1.$$

# K-Means is a Limit of EM Algorithm

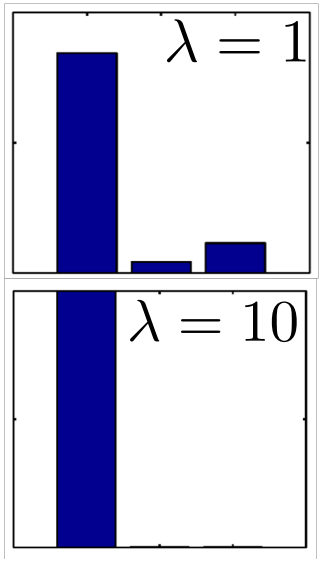
$$p(z_i \mid \pi) = \mathrm{Cat}(z_i \mid \pi) \qquad \pi_k = \frac{1}{K}$$
$$p(x_i \mid \mu_k) = \mathrm{Norm}(x_i \mid \mu_k, \sigma^2 I)$$

***Constrain clusters to be equally common, and have same circular shape.***

**E-Step: Given parameters, assignments probabilities equal**

$$r_{ik} = p(z_i = k \mid x_i, \mu) = \frac{\mathrm{Norm}(x_i \mid \mu_k, \sigma^2 I)}{\sum_{\ell=1}^{K} \mathrm{Norm}(x_i \mid \mu_\ell, \sigma^2 I)}$$

$$r_{ik} = \frac{\exp\{-\lambda ||x_i - \mu_k||^2\}}{\sum_{\ell=1}^{K} \exp\{-\lambda ||x_i - \mu_\ell||^2\}} \qquad \lambda = \frac{1}{2\sigma^2}$$




- **In limit as variance approaches zero, *softmax* assigns weight (probability) one to the single, closest cluster**
- **K-Means is equivalent to EM algorithm when clusters have circular covariance, and cluster means are widely separated**

# 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 (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