Unsupervised Learning

Michalis Vazirgiannis

DASCIM web page: http://www.lix.polytechnique.fr/dascim

Personal web page: http://www.lix.polytechnique.fr/~mvazirg

Google Scholar: https://bit.ly/2rwmvQU

Twitter: @mvazirg

November 2017

Outline

- **Introduction to unsupervised learning**
- K Means
- EM for GMMs
- Spectral Clustering
- Auto-encoders for unsupervised learning

Supervised vs. Unsupervised Learning

- **Unsupervised learning (clustering)**
  - The class labels of training data are unknown
  - Given a set of measurements, observations, etc. establish the existence of clusters in the data
- **Supervised learning (classification)**
  - **Supervision:** The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- **Semi-supervised clustering**
  - Learning approaches that use **user input** (i.e. constraints or labeled data)
  - Clusters are defined so that user-constraints are satisfied

Unsupervised learning (UL) essence

- machine simply receives inputs x_1, x_2, . . .no supervised target outputs, nor rewards from its environment.
- what can machine learn given that it *doesn't get any feedback from its environment.*
- framework for unsupervised learning: build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine.
- unsupervised learning can be thought as: *finding patterns in the data beyond what would be considered pure unstructured noise*.
- classic examples: clustering and dimensionality reduction.

Relation of UL to statistics and Information theory

- Let $P(x)$ the distribution generating the data
- coding length of an element with $P(x)$ (according to Shannon's theorem) $-log_2(P(x))$
- Expected coding cost for the distribution P

$$E = -\sum_x P(x) log_2(P(x))$$

- Data distribution generally unknown. Let model $Q(X)$ learned from the data. Optimal code length $-log_2(Q(x))$
- Expected coding cost, taking expectations with respect to the true distribution

$$-\sum_x P(x) log_2(Q(x))$$

Relation of UL to statistics and Information theory

- difference between coding costs is called the Kullback-Leibler (KL) divergence

$$KL(P||Q) = \sum_x P(x) log(\frac{P(x)}{Q(x)})$$

- Measures coding inefficiency using a model Q to compress data when true data distribution is P.
- KL divergence is non-negative and zero if and only if $P=Q$.
- the better model Q the more efficiently we can compress and communicate new data.
- link between machine learning, statistics, and information theory.

UL arising from Bayes rule

- Assume a set of possible models $\Omega = \{\omega_1, \ldots, \omega_M\}$ for an observed Dataset D with beliefs $P(\omega_i)$ where

$$\sum_{m=1}^{M} P(\omega_m) = 1$$

- Beliefs of the models based on D:

$$P(\omega_m|D) = \frac{P(\omega_m)P(D|\omega_m)}{P(D)}$$

- equivalent to:

$$P(\omega_m|D) = P(\omega_m) \prod_{i=n} P(n|\omega_m)$$

- *posterior over models is the prior multiplied by the likelihood, normalized*

UL arising from Bayes rule – how to encode new data

- The *predictive distribution over new data*, to encode new data efficiently, is

$$P(x|D) = \sum_{m=1}^{M} P(x|\omega_m)P(\omega_m|D)$$

- models are assumed to produce iid data
- Often models defined by a parametric probability distribution

$$P(x|\omega_m) = \int P(x|\theta, \omega_m)P(\theta|\omega_m)d\theta$$

- Assume a particular model m with parameters θ, and an observed data set D. The predictive distribution averages over all possible parameters weighted by the posterior:

$$P(x|D, \omega_m) = \int P(x|\theta)P(\theta|D, \omega_m)d\theta$$

UL arising from Bayes rule – how to encode new data

- Aim to find the most probable parameter (MAP) values given the data set and the model

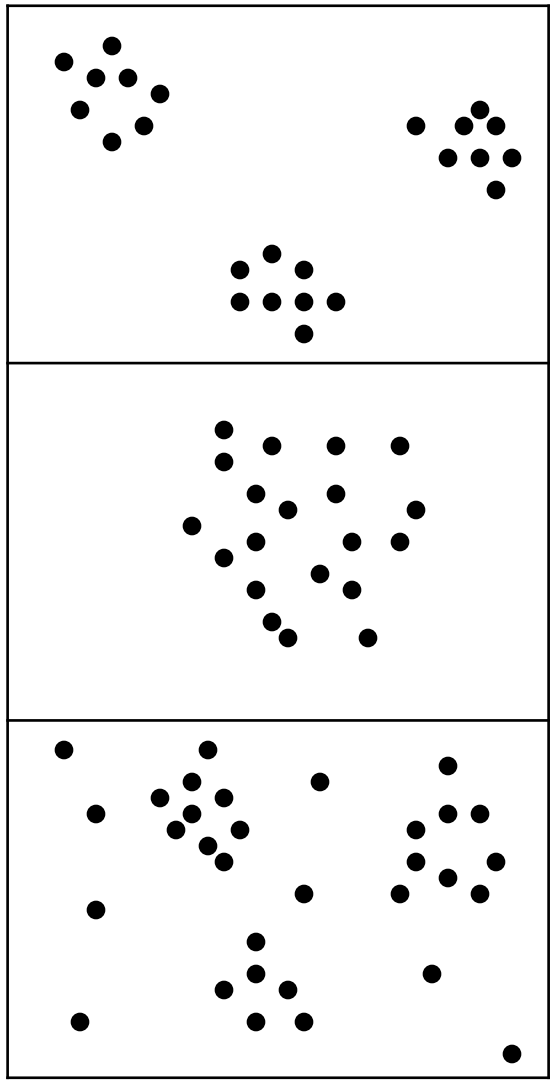
$$\hat{\theta}_{MAP} = argmax_{\theta}[logP(\theta|\omega_m) + \sum_n log(P(x_n|\theta, \omega_m))]$$

- Or find the parameter than maximizes likelihood:

$$\hat{\theta}_{ML} = argmax_{\theta} P(\theta|D, \omega_m) = argmax_{\theta} \sum_n log(P(x_n|\theta, \omega_m))$$

- ML estimation is prone to over fitting—more complex models will generally have higher maxima of the likelihood.

Clustering



Sometimes easy

Sometimes impossible

sometimes in between

- "automated detection of group structure in data"
  - Typically: partition N data points into K groups (clusters) such that the points in each group are more similar to each other than to points in other groups
  - descriptive technique (contrast with predictive)

Why is Clustering useful?

- "Discovery" of new knowledge from data
  - Contrast with supervised classification (where labels are known)
  - Long history in the sciences of categories, taxonomies, etc
  - Can be very useful for summarizing large data sets
    - For large n and/or high dimensionality

- Applications of clustering
  - Discovery of new types of galaxies in astronomical data
  - Clustering of genes with similar expression profiles
  - Cluster pixels in an image into regions of similar intensity
  - Segmentation of customers for an e-commerce store
  - Clustering of documents produced by a search engine
  - …. many more

General Issues in Clustering

- Representation: What types of clusters are we looking for?
- Score:The criterion to compare one clustering to another
- Optimization: Generally, finding the optimal clustering is NP-hard Greedy algorithms to optimize score are widely used
- Other issues
  - Distance function, $D(x(i),x(j))$ critical aspect of clustering, both
    - distance of pairs of objects
    - distance of objects from clusters
  - How is K selected?
  - Different types of data
    - Real-valued versus categorical
    - Attribute-valued vectors vs. n^2 distance matrix

Clustering Methods

- **Partitional algorithms**
  - K-Means, PAM, CLARA, CLARANS [Ng and Han, VLDB 1994]
- **Hierarchical algorithms**
  - CURE [Guha et al, SIGMOD'98], BIRCH [Zhang et al, SIGMOD'96], CHAMELEON [IEEE Computer, 1999]
- **Density based algorithms**
  - DENCLUE [Hinneburg, Keim, KDD'98], DBSCAN [Ester et al, KDD 96]
- **Subspace Clustering**
  - CLIQUE [Agrawal et al, SIGMOD'98], PROCLUS [Agrawal et al, SIGMOD'99], ORCLUS: [Aggarwal, and Yu, SIGMOD' 00], DOC: [Procopiuc, Jones, Agarwal, and Murali, SIGMOD'02]
- **Spectral clustering**
  - [Ng, Jordan, Weiss], [Shi/Malik], [Scott/Longuet-Higgins], [Perona/ Freeman]

Outline

- Introduction to unsupervised learning
- **K Means**
- EM for GMMs
- Spectral Clustering
- Auto-encoders for unsupervised learning

k-Means

- *k*-Means Clustering aims to retrieve clusters $C_1, C_2, \ldots, C_k$ that minimize objective function:

$$J_k = \sum_{j=1}^{k} \sum_{i \in C_k} ||x_i - m_j||^2$$

- m_j is the center of cluster C_k
- Thus the clustering that minimizes the distances to cluster centers is retrieved.
- *NP*-Hard problem, even for $k=2$.
- Most popular heuristic: Lloyd's algorithm.

Lloyd's heuristic algorithm

- Due to its popularity, Lloyd's heuristic is commonly referred to as *k*-Means algorithm.
- Algorithm:
  - `Randomly select k elements from the dataset-cluster centroids`
  - `Each element assigned to the nearest cluster center.`
  - `Compute new barycenters.`
  - `Each element re-assigned to the nearest cluster.`
  - `repeated until error < threshold`
- Advantages:
  - Fast and effective method for large datasets.
- Disadvantages:
  - Works well only for convex and dense clusters.
- Time complexity: $O(I\ e\ \underline{n\ k})$
  - I = number of iterations (steps)
  - e = cost of distance computation

K-means application on image compression

Image

Clusters on color

K-means clustering of RGB (3 value) pixel color intensities, K = 11 segments
(courtesy of David Forsyth, UC Berkeley)

K-means clustering issues

- Tends to select compact “isotropic” cluster shapes
- Useful for initializing more complex methods
- Different variants (k-means++, k-medoid, …)
- Choice of distance measure
  - Euclidean distance
  - Weighted Euclidean distance
  - Many others possible
- Selection of # clusters k
  - Plot error versus K, local minima

Outline

- Introduction to unsupervised learning
- K Means
- **EM for GMMs**
- Spectral Clustering
- Auto-encoders for unsupervised learning

Expectation Maximization

- (EM) algorithm
  - iterative method for finding maximum likelihood estimates of parameters in statistical models
  - unobserved latent variables.
- Assume X the data observed
- assume the data are produced by K different classes/processes
- Respective w_k weights there fore

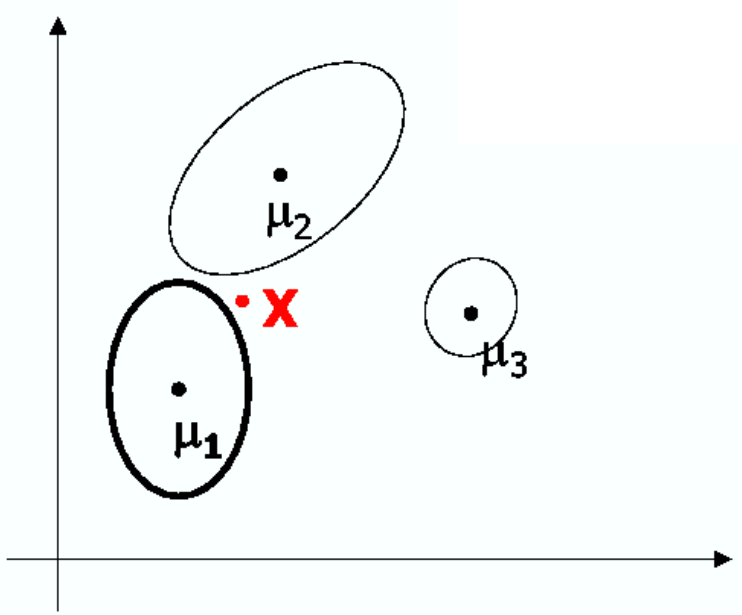
$$f(x) = \sum_k w_k f_k(x|\theta_k)$$

Gaussian Mixture Models (GMM)

- Assume the the components are normal distrubutions $N(\mu_k, \sum_k)$
  - often assume diagonal covariance: $\sum_{jj} = \sigma_j^2$, $\sum_{i \neq j} = 0$
  - or sometimes even simpler: $\sum_{jj} = \sigma^2$, $\sum_{i \neq j} = 0$

- $f(x) = \sum_{k=1 \ldots K} w_k f_k(x; \theta_k)$ with $\theta_k = <\mu_k, \sum_k>$ or $<\mu_k, \sigma_k>$

- <u>generative</u> model:
  - - randomly choose a component
  - selected with probability w_k
  - - generate $x \sim N(\mu_k, \sigma_k)$
  - - note: μ_k & σ_k both d-dim vectors



Learning Mixture Models from Data

- **Score function** Log-likelihood $L(\theta)$
  - $L(\theta) = log\ p(X|\theta) = log \sum_H p(X,H|\theta)$
  - H = hidden variables (cluster memberships of each x)
  - $L(\theta)$ cannot be optimized directly
- **Expectation Maximization Procedure**
  - General technique for maximizing log-likelihood
  - For mixtures of distributions
    - *E-step*: compute "memberships" $p(k \mid x) = w_k f_k(x;\theta_k) / f(x)$
    - *M-step*: new θ values based on the E step towards maximizing data log-likelihood
    - Iterate: guaranteed to climb to (local) maximum of $L(\theta)$

Expectation maximization (EM)

Expectation-maximization algorithm computes memberships of data points in a probability distribution

Expectation step

- initial guesses for the parameters in the mixture model
- compute "partial membership" of each data point in each constituent distribution.
- By calculating expectation for the membership variables of each data point.

Example.

- Assume two Gaussian distributions.

$$P(x_i) = (1-f)\mathrm{N}(x_i \mid \mu_1, \sigma) + f\mathrm{N}(x_i \mid \mu_2, \sigma)$$

f : mixing coefficient in (0,1], assume variance σ is known and constant.

- Membership of x_i to each of the two Gaussians

$$y_{1,i}(x_i) = \frac{(1-f)\mathrm{N}(x_i \mid \mu_1, \sigma)}{(1-f)\mathrm{N}(x_i \mid \mu_1, \sigma) + f\mathrm{N}(x_i \mid \mu_2, \sigma)}$$

similarly for $y_{2,i}$

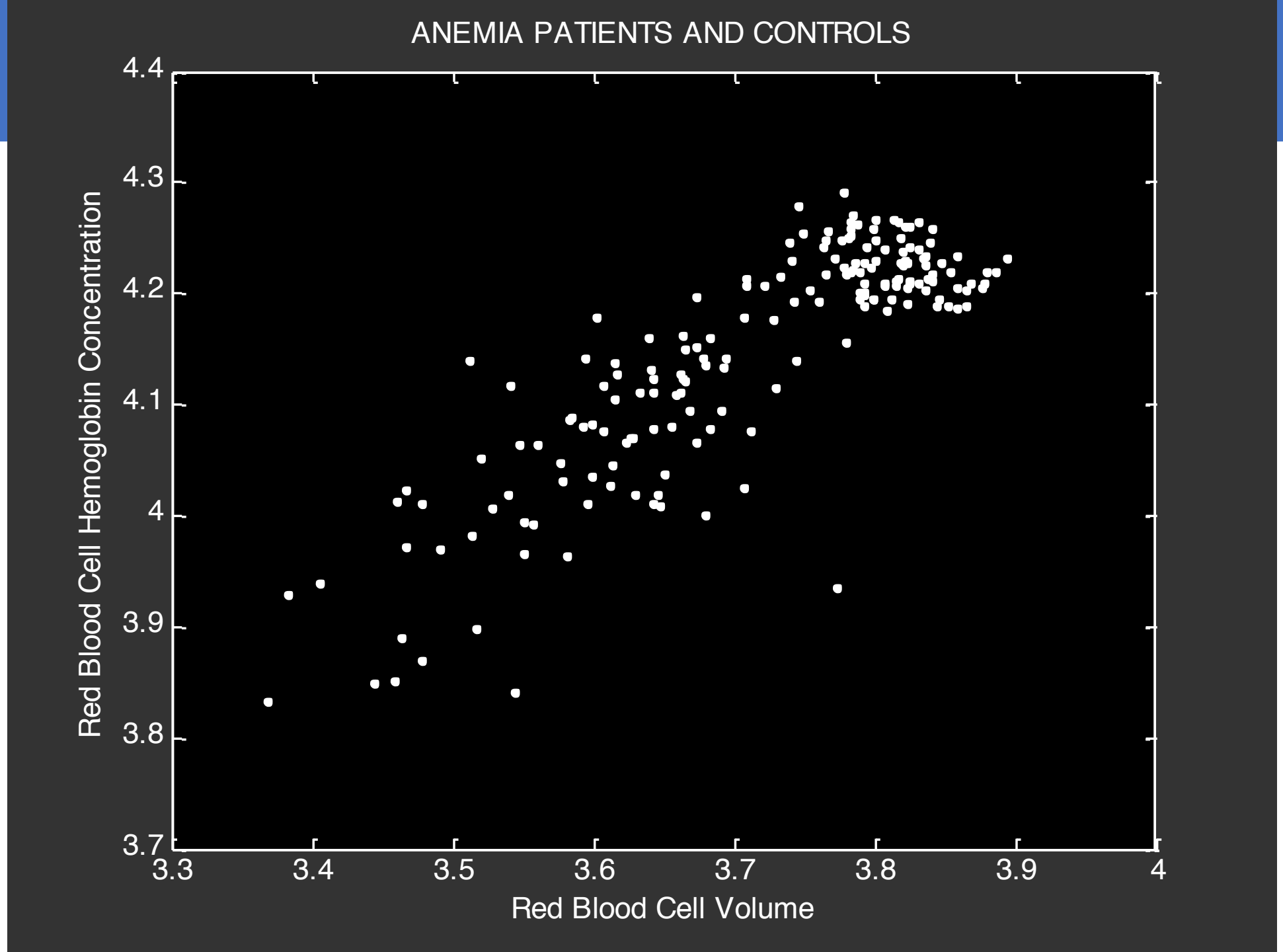
Expectation maximization (EM)

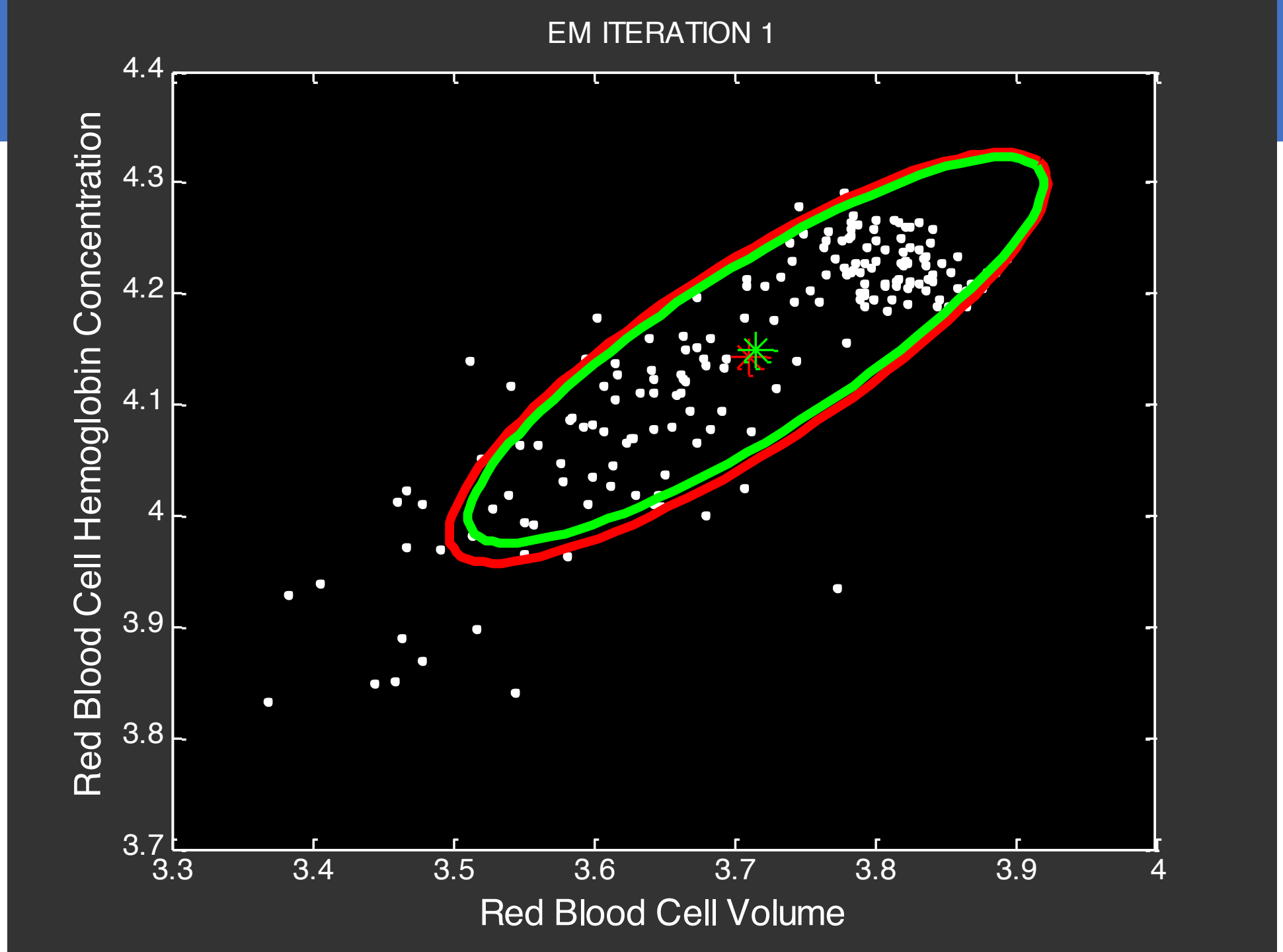
The maximization step

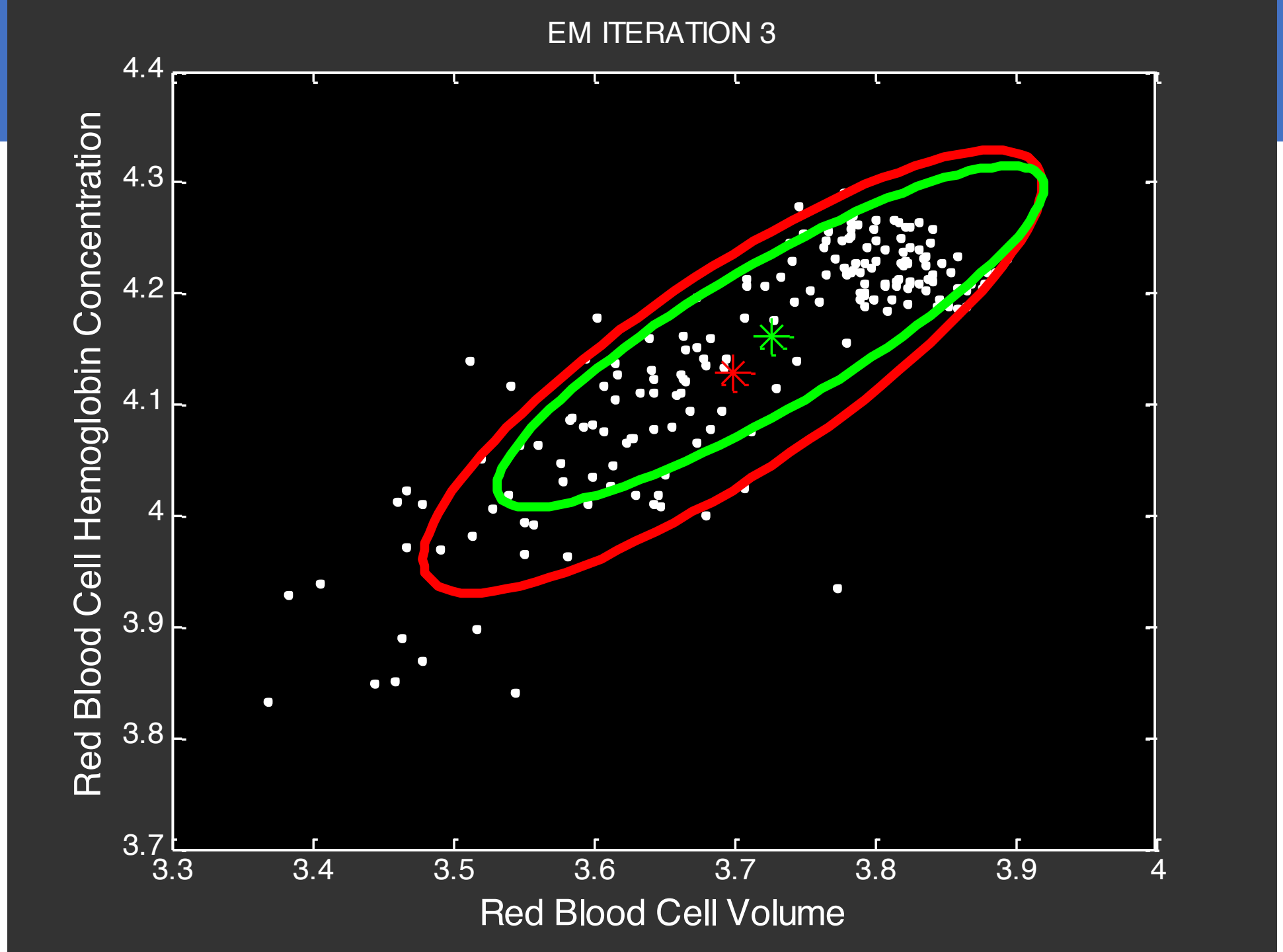
- Using values for memberships $y_{i,j}$
- compute new values for distribution parameters.

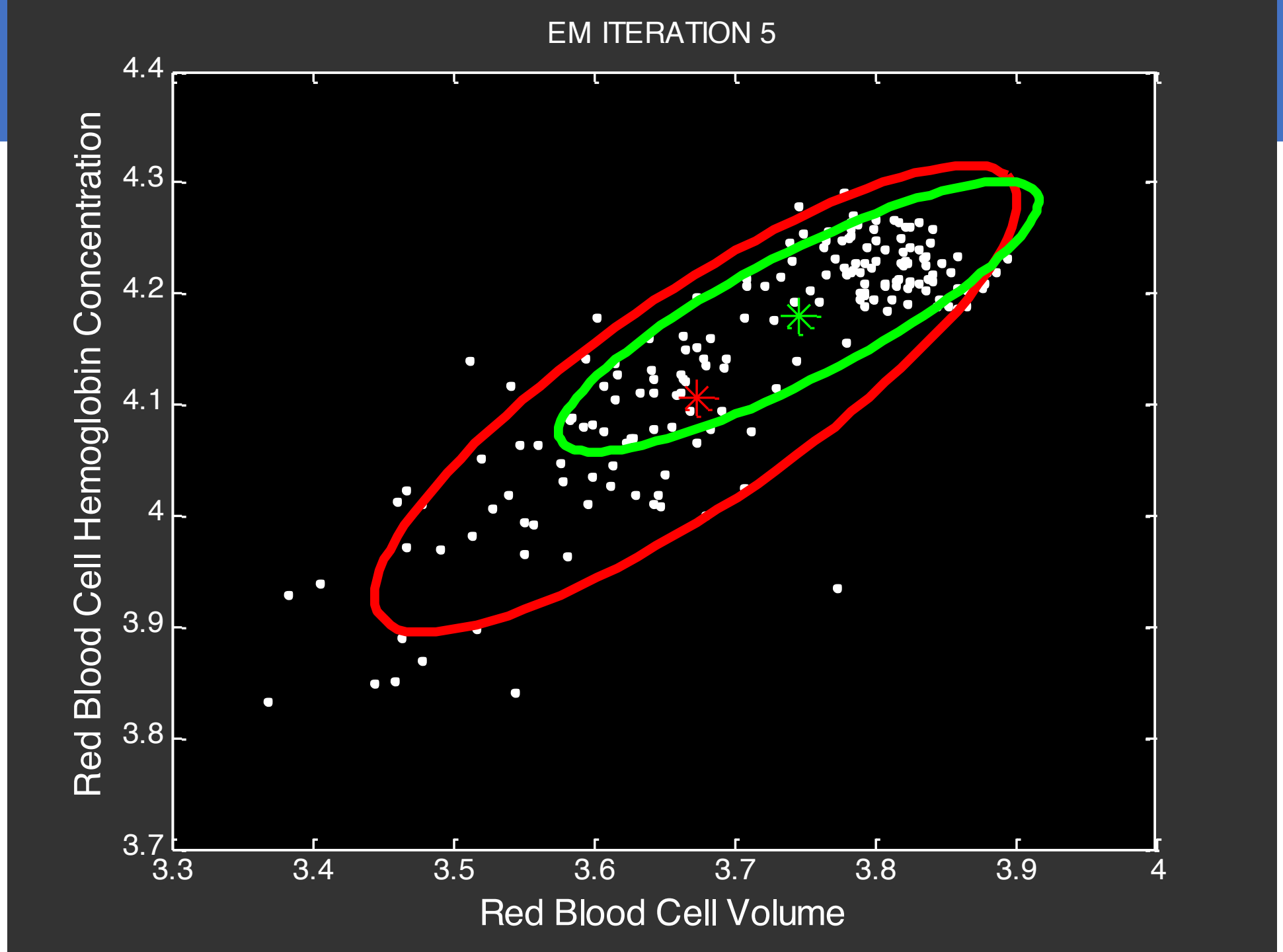
$$\begin{cases} f = \frac{\sum_i y_{i,2}}{N} \\ \\ \mu_1 = \frac{\sum_i y_{i,1} x_i}{\sum_i y_{i,1}} \end{cases}$$

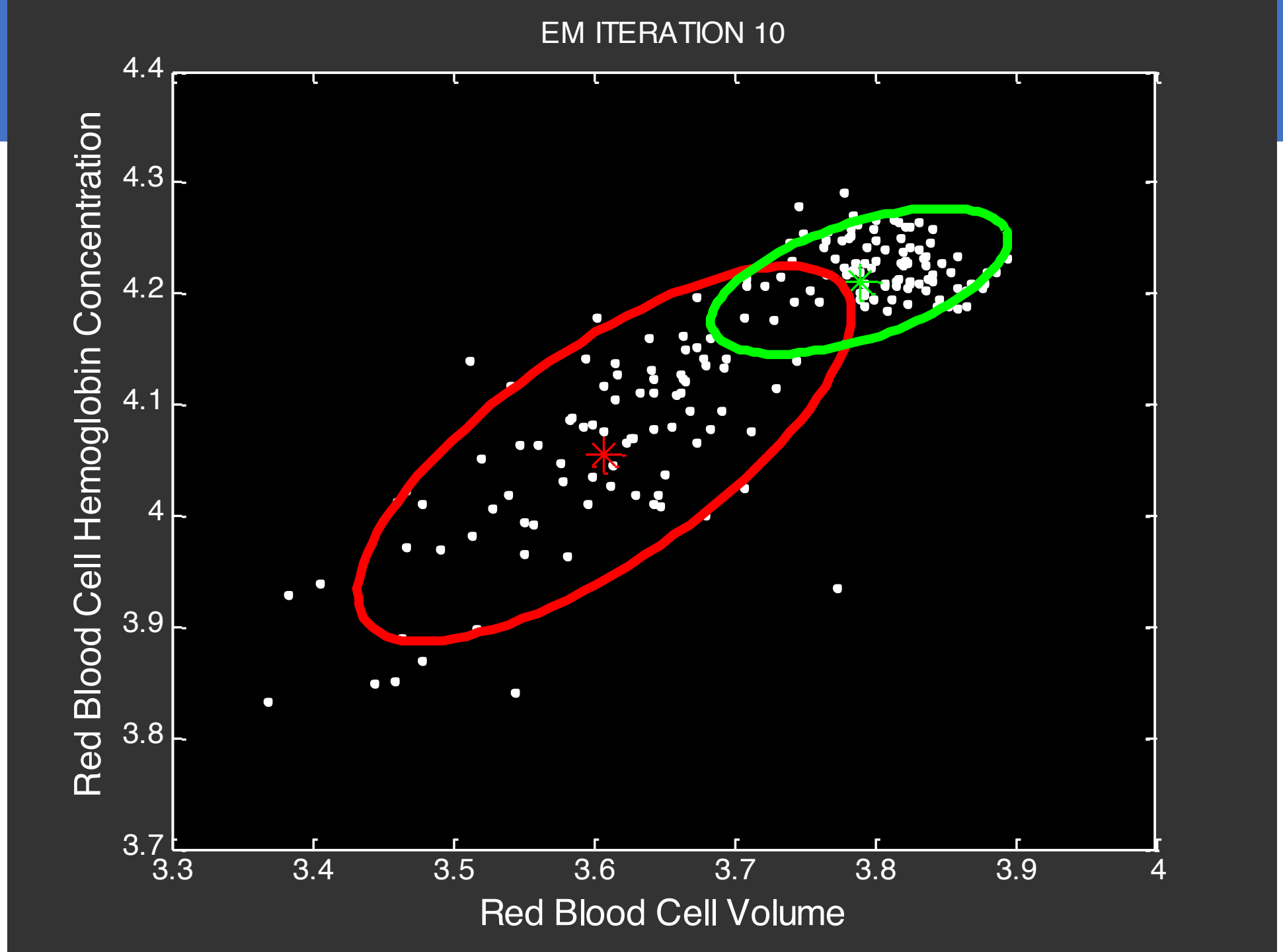
  - *N* : number of data points.
- back to Expectation step: Re-compute new membership values
- repeated until change in mixture model parameters below threshold

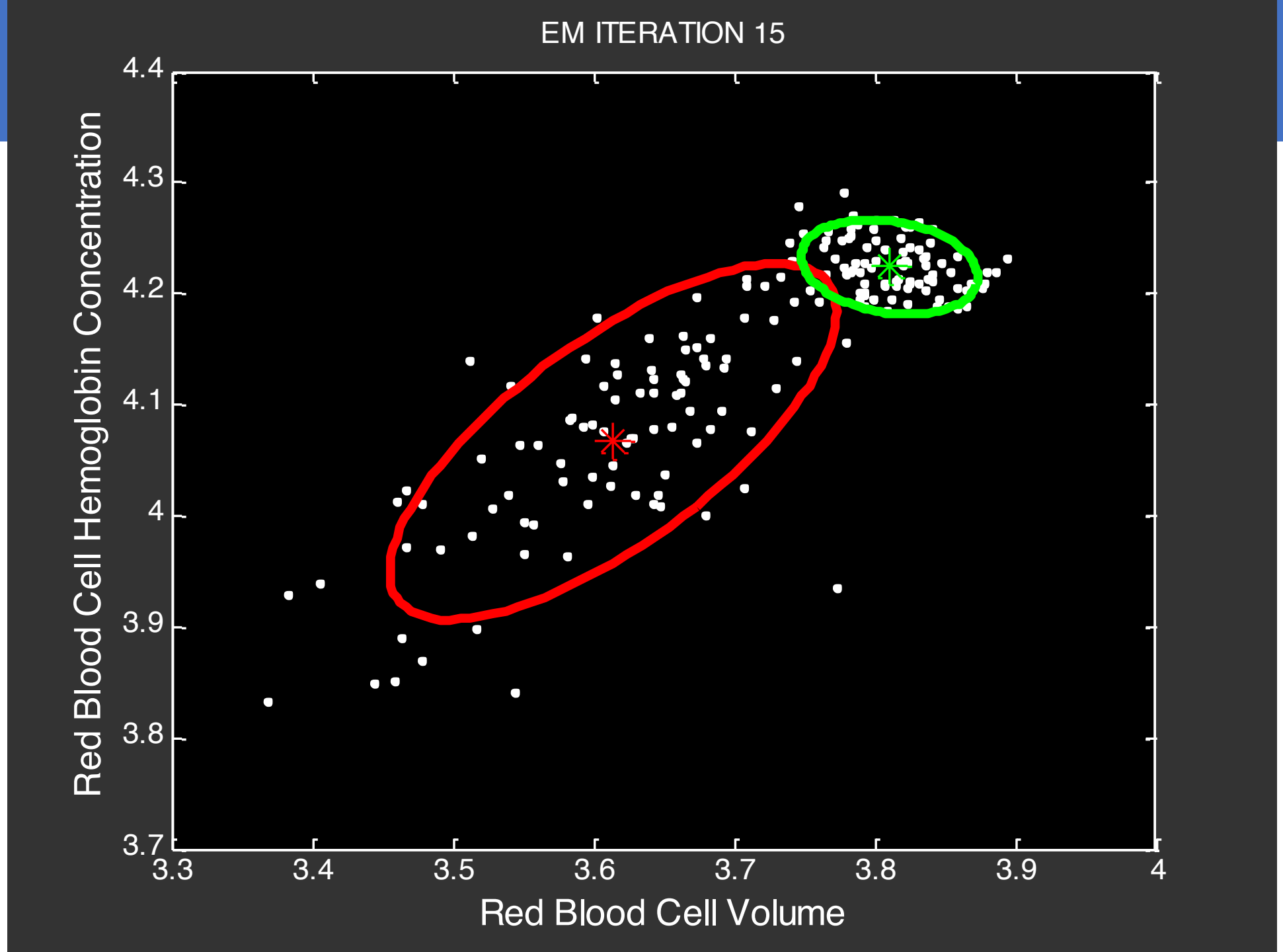

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

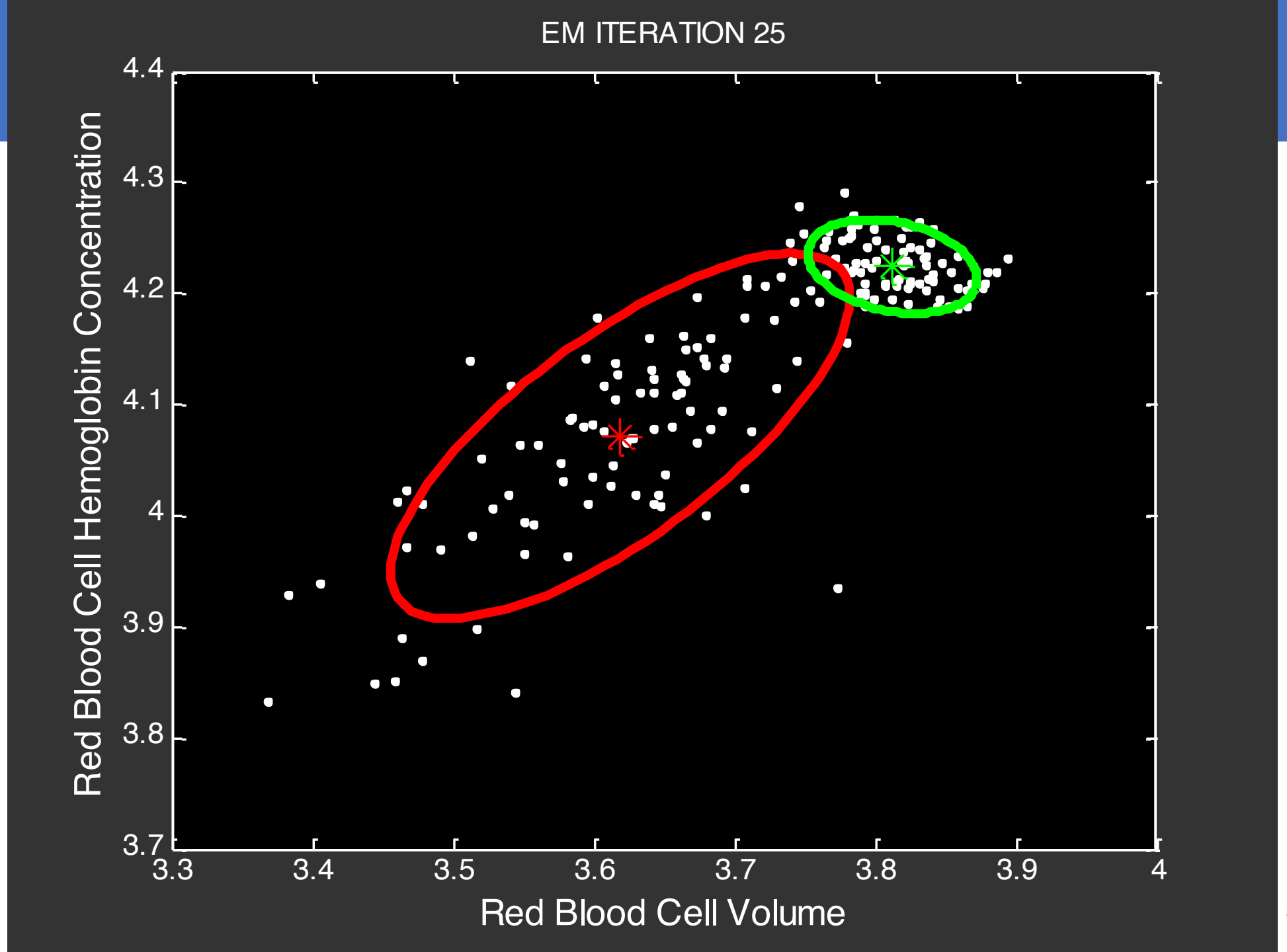

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

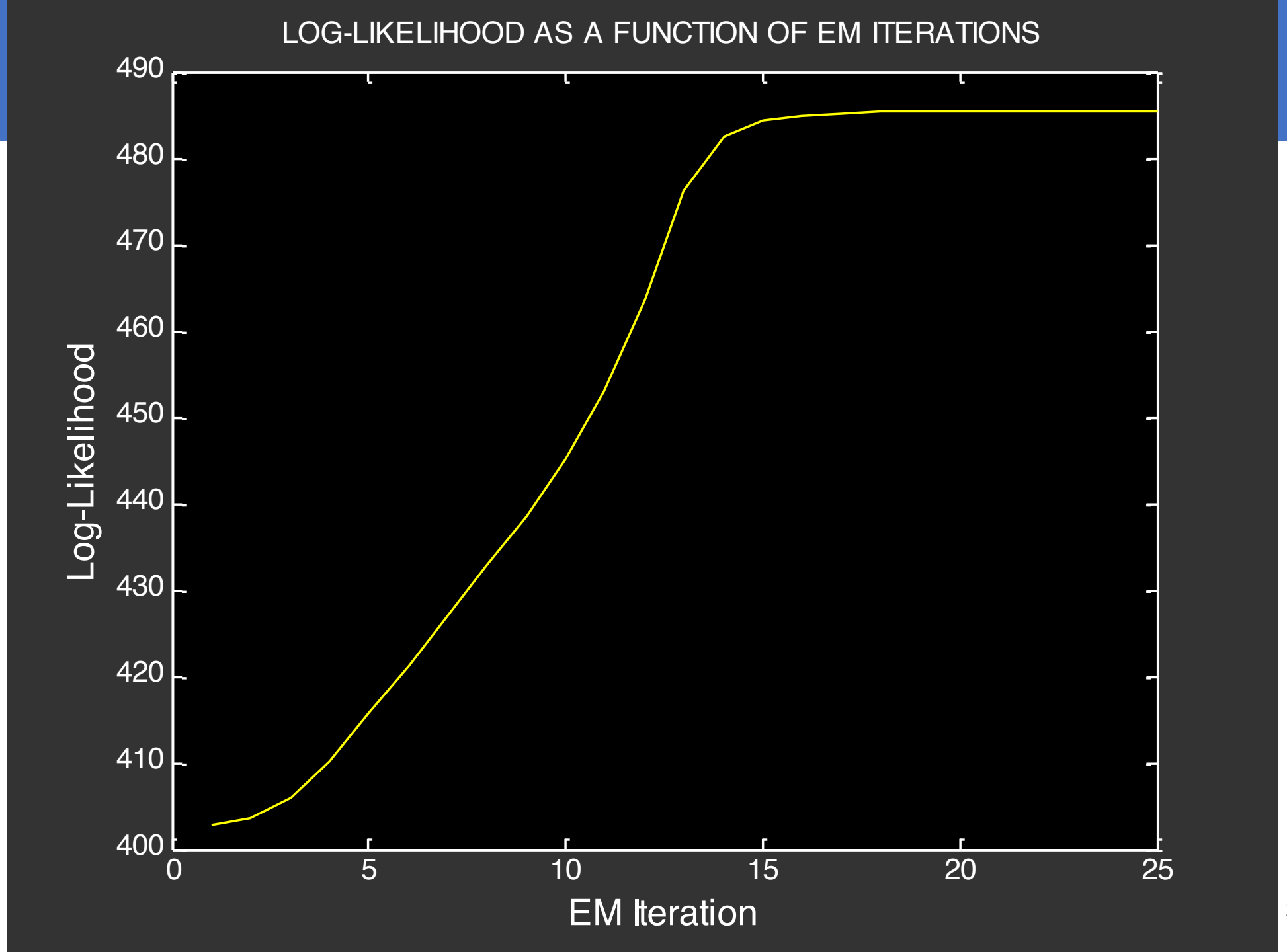

EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

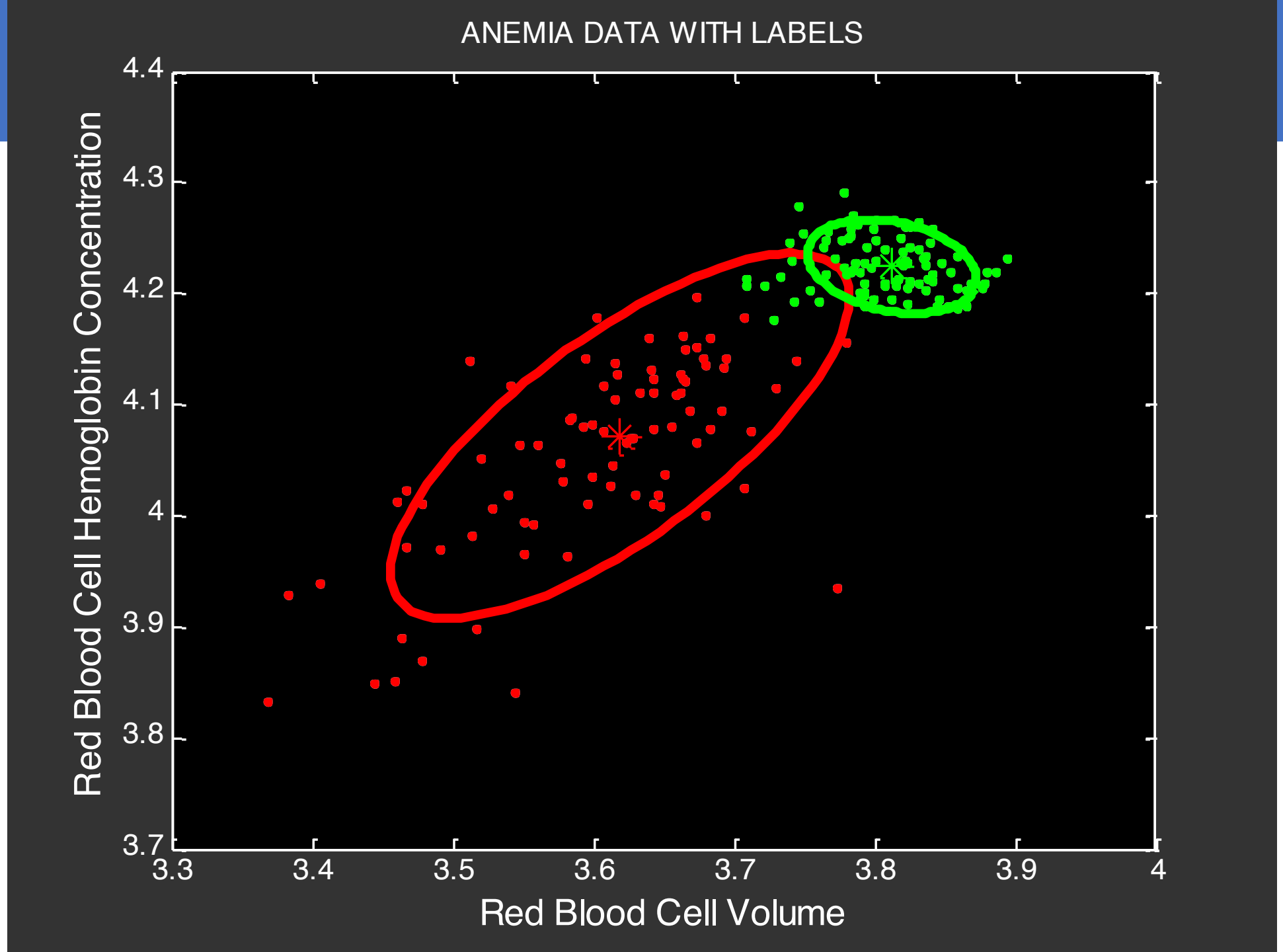

EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


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


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


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


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


ANEMIA DATA WITH LABELS
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

Outline

- Introduction to unsupervised learning
- K Means
- EM for GMMs
- **Spectral Clustering**
- Auto-encoders for unsupervised learning

Spectral *k*-Means

- *k*-Means can be stated as a Trace maximization problem in the form:

$$min_Y(\mathbf{Tr}(XX^T) - \mathbf{Tr}(Y^TXX^TY)) \equiv$$
$$max_Y(\mathbf{Tr}(Y^TXX^TY))$$

- X (*object* x *feature)* matrix, Y: discrete cluster assignments.

$$Y_{ic} = \begin{cases} \frac{1}{\sqrt{|\pi_c|}} & \text{if object i} \in \pi_c \\ 0 & \text{otherwise} \end{cases}$$

- If we relax Y to be any orthogonal matrix the solution Y will contain as columns the k dominant eigenvectors of XX^T.
- Need of extra step for discretizing solution.

- Trace of a square matrix nxn $Tr(A) = \sum_i a_{ii}$
- $Tr(A) =$ sum of the eigenvalues and also an invariant with respect to a change of basis

Graph partitioning – *the CUT concept*

- Aim is to cluster the datasets/partition the graph.
- Lets try to derive the clustering that minimizes the sum of weights between objects in different clusters.

$$cut(A, B) = \sum_{I \in A, j \in B} w_{ij}$$

- For *k*-clusters.

$$cut(A_1, ..., A_k) = \sum_{i=1}^{k} cut(A_i, \hat{A})$$

- Minimizing cut directly does consider cluster size.
  - May lead to few points clusters

Objective Functions

- Two popular objective functions that balance for cluster sizes:

$$Ratiocut(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \hat{A})}{|A_i|}$$

$$Ncut(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \hat{A})}{vol(A_i)}$$

- minimizing these objective functions is *NP*-Hard.
- can be formulated as *Trace minimization* problems.
- approximated by spectral methods.
- Notation:
  - W= the object similarity matrix.
  - D= diagonal matrix with diagonal values $d_{i,i}=\sum_{j=1}^{n} w_{ij}$

Ratio Cut /Spectral Clustering

- In graph-partitioning a popular clustering objective is Ratio Cut.

$$\mathrm{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \overline{A}_i)}{|A_i|}$$

- Equivalent Trace minimization problem

$$min_Y \mathbf{Tr}(Y^T (D - W) Y)$$

- relax Y to be any orthogonal matrix Y, solution contain as columns the *k*-eigenvectors corresponding to the smallest eigenvalues of Un-normalized Graph Laplacian.

$$L = D - W.$$

- For 2 clusters problem: clustering the values of the eigenvector that corresponds to the *second smallest eigenvalue*.

Normalized Cut /Spectral Clustering

- In graph-partitioning a popular clustering objective is Normalized Cut (*NCut*)

$$NCut(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$$

- Equivalent Trace optimization problem

$$min_Y \mathbf{Tr}(Y^T(I - D^{-1/2}WD^{-1/2})Y)$$

- If we relax *Y* to be any orthogonal matrix *Y,* solution will contain as columns the *k*-eigenvectors that correspond to the smallest eigenvalues of Normalized Graph Laplacian.

$$L = I - D^{-1/2}WD^{-1/2}$$

- In the case of 2-way clustering the solution is derived by the eigenvector that corresponds to the *second smallest eigenvalue*.

Spectral Clustering

- A family of algorithms that use eigenvectors of matrices that derive from the data.
- A similarity matrix is needed
- They are called spectral because they use the spectrum of the similarity matrix to reduce the dimension.
- The selection and use of the eigenvectors differs from one algorithm to another.
- The number of clusters is defined by the user (k)
- Application in areas like :
  - Image segmentation
  - Community detection in graphs

Affinity Matrix

- Affinity Matrix $A \in R^{nxn}$
- Define $A_{ij} = e^{-\|s_i - s_j\|^2 / 2\sigma^2}$ for $i \neq j$ else $A_{ii} = 0$
- The scaling factor (σ^2) is chosen by the user
- "Closer" points will have higher weight
- The weight is a function of (σ)
- Realistically, we search over the values of σ^2 and chose the one that returns the "tightest" clusters

Laplacian Matrix

- In the case of
  - data points: distance Matrix(A).
  - Graphs: Adjacency Matrix
- Laplacian Matrix
  - Unormalized : $L = D - A$
  - Normalized (symmetric) : $L^{sym} = D^{-1/2} L D^{-1/2}$
  - Normalized (random walk) : $L^{ran\_w} = D^{-1} L$

Where D : degree matrix, $D(i,i)=sum(row(A,i))$

Laplacian Matrix

Properties of $L = D\text{-}W$

The matrix L satisfies the following properties:

1. For every vector f in R^n :

$$f'Lf = 1/2 \sum_{i,j=1}^{n} w_{ij}(f_j - f_j)^2$$

2. L is symmetric and positive semi-definite.
3. Min eigenvalue of L is 0
4. L has n non-negative, real-valued eigenvalues

$$0 = \lambda_1 <= \lambda_1 <= \ldots <= \lambda_n$$

Spectral Clustering Algorithms

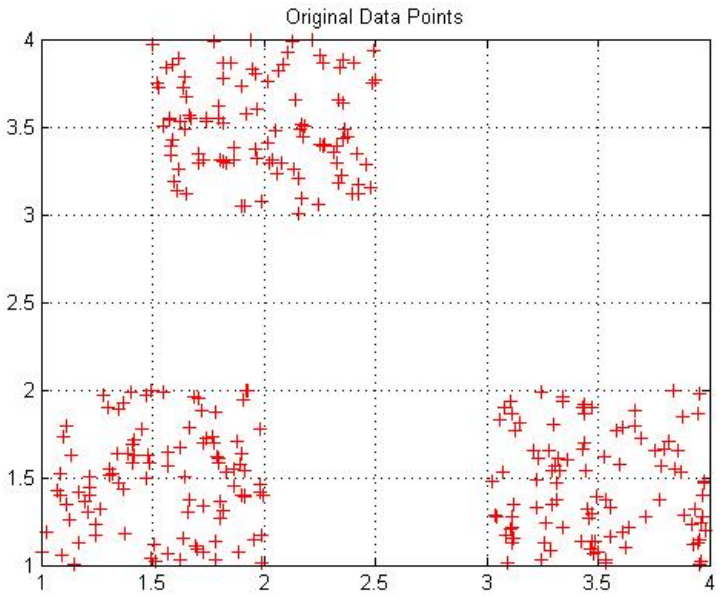
clustering (k>2) - Unnormalized

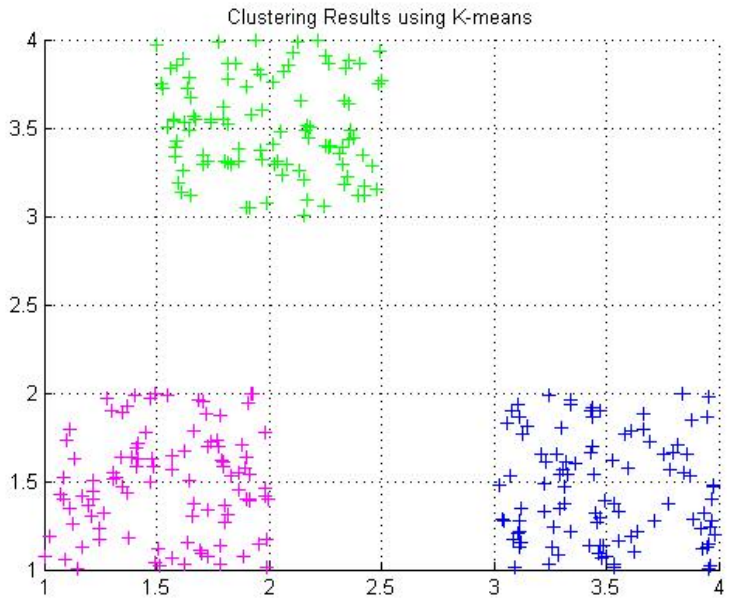
Input: Affinity matrix $W\,(n\times n)$, number of clusters k
- Compute Laplacian L.
- Compute the top k eigenvectors $u_1, \ldots, u_k$ of L.
- Let $U_{n\times k}$ matrix vectors $u_1, \ldots, u_k$ as columns.
- apply $k\text{-}means$ on the rows y_j of the $U_{n\times k}$ matrix
  - *Output*: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \; belongs \; to \; A_i\}$.

Why not K-Means?

- Last step in the example algorithm is the application of K-means
- why not just apply K-means to the original data?
  - inability of k-means to detect non-convex regions.
- Spectral/eigenvector domain is a lower-dimensional space where the points are easily separable.

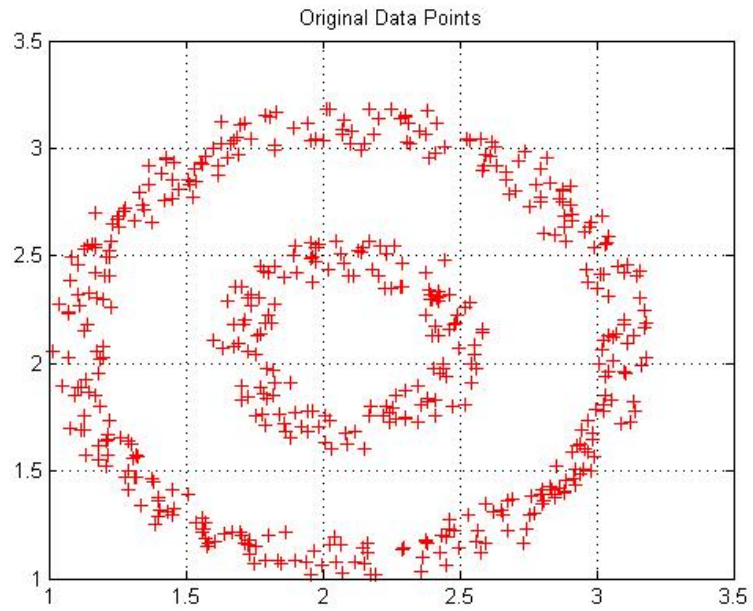
K-means

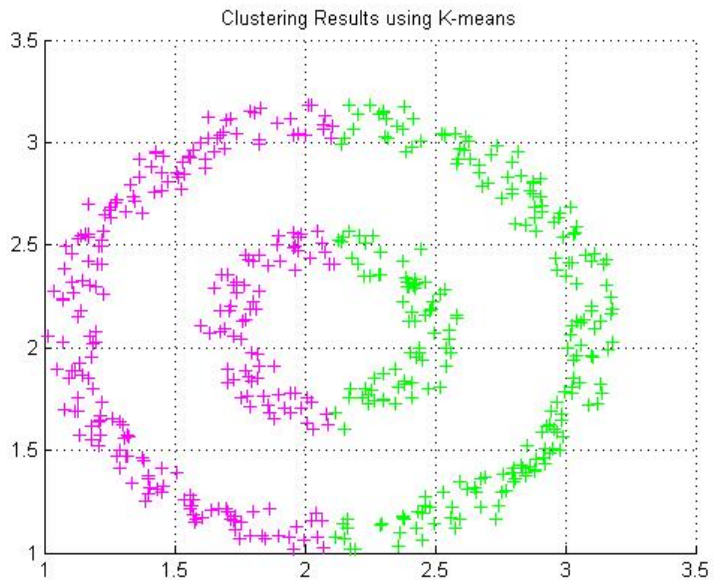




- K-means is good at finding dense areas that are defined by a convex region

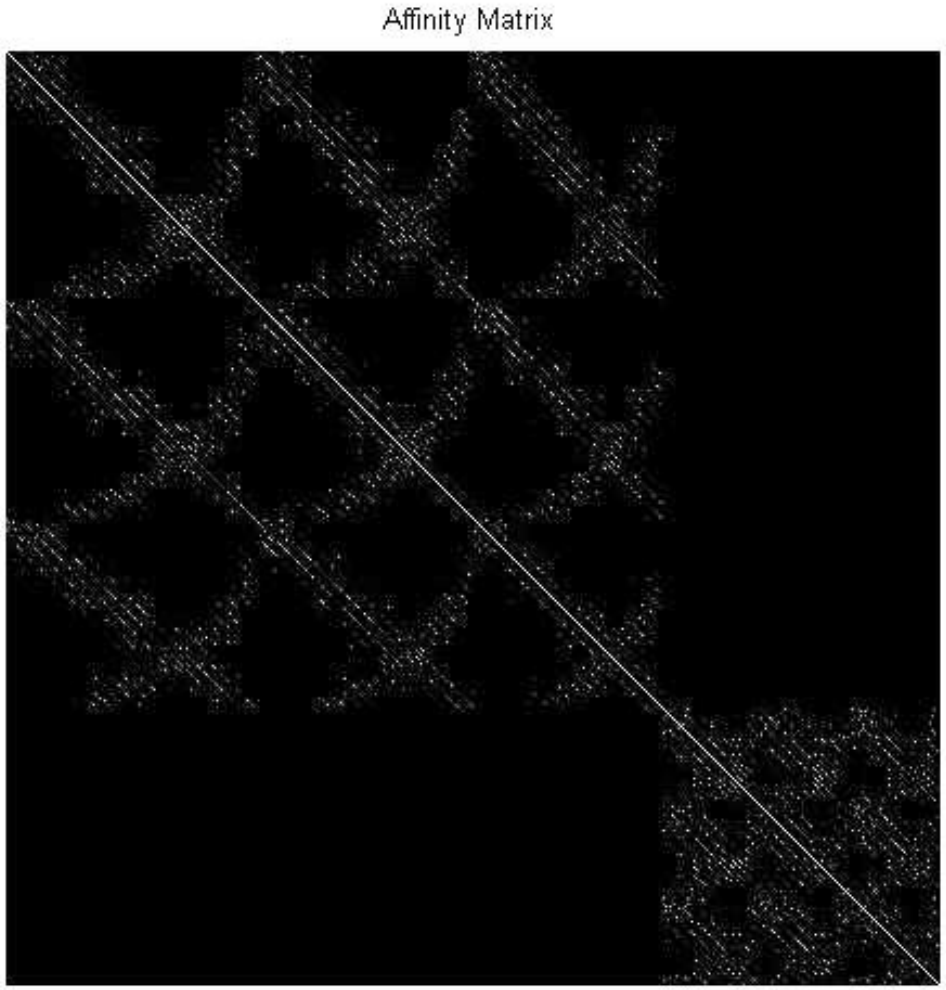
K-means





- On non-convex clusters k-means will give bad results due to the tendency to find equal-sized clusters.

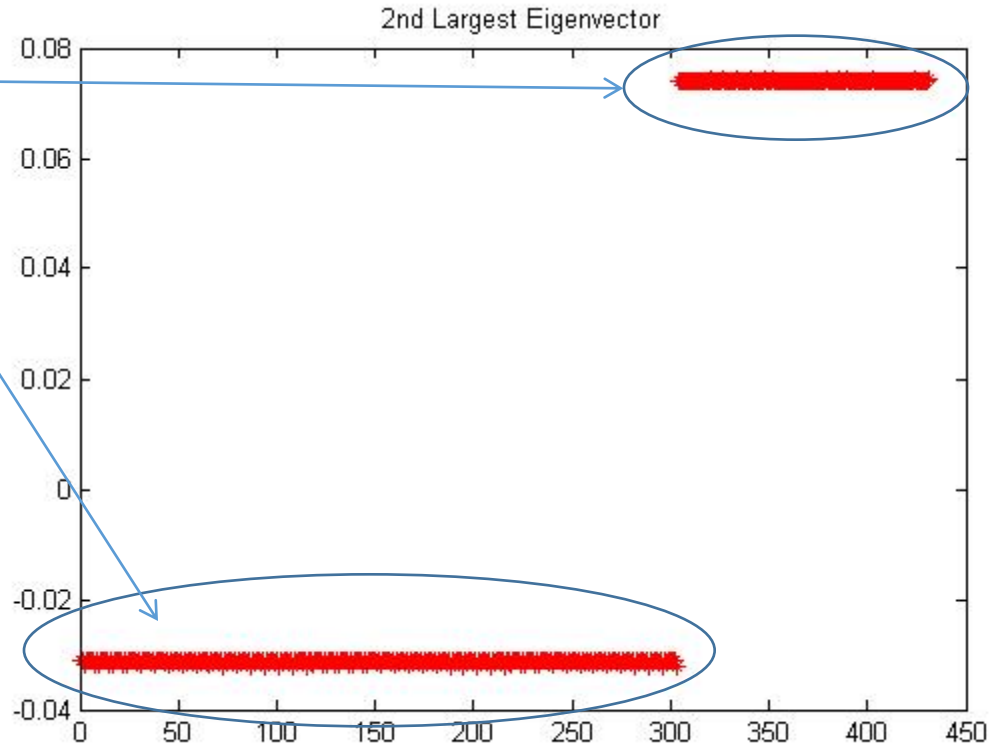
Example 1 - “Good” Scaling Factor (0.1)



Example 1 - “Good” Scaling Factor (0.1)

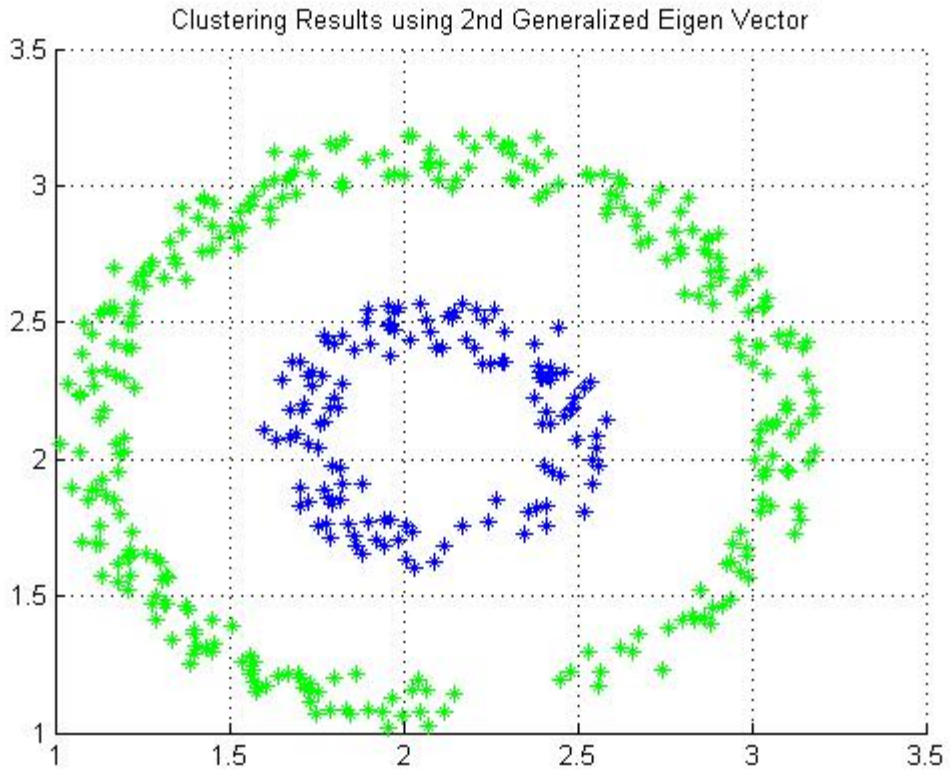
- 2nd Largest EigenVector

The values of the eigenvector are easily separable into two clusters

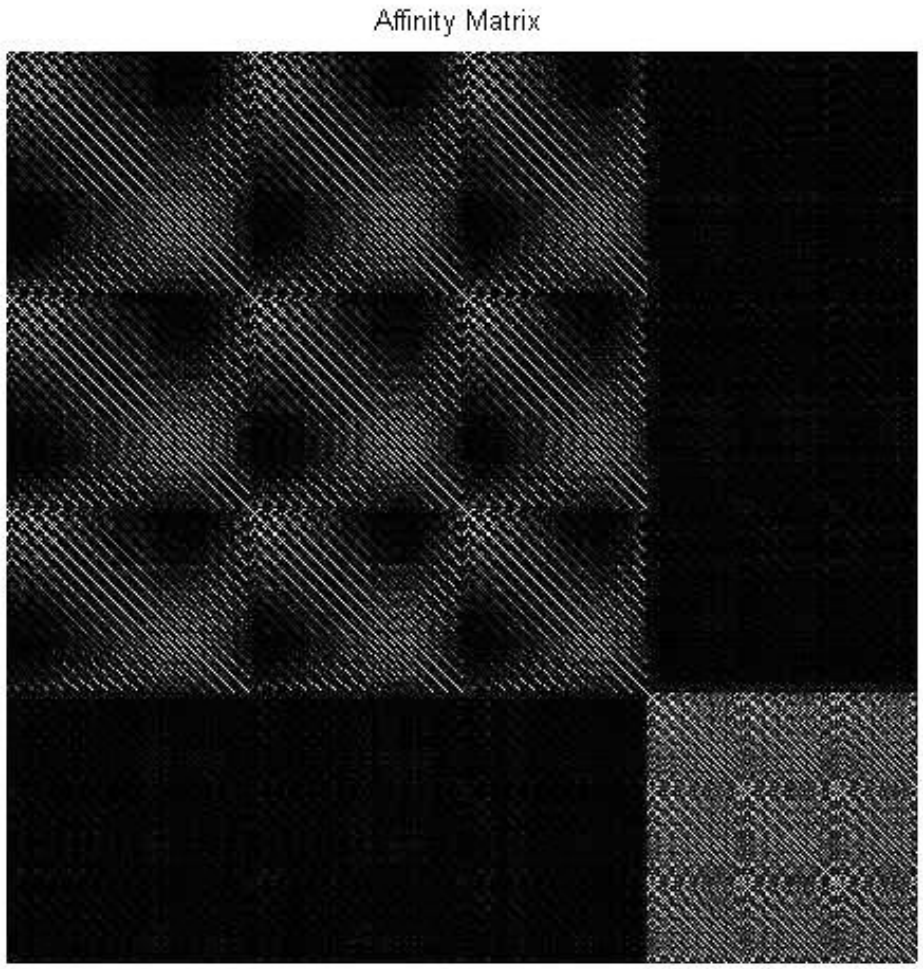


Example 1 “Good” Scaling Factor (0.1)

- Clusters

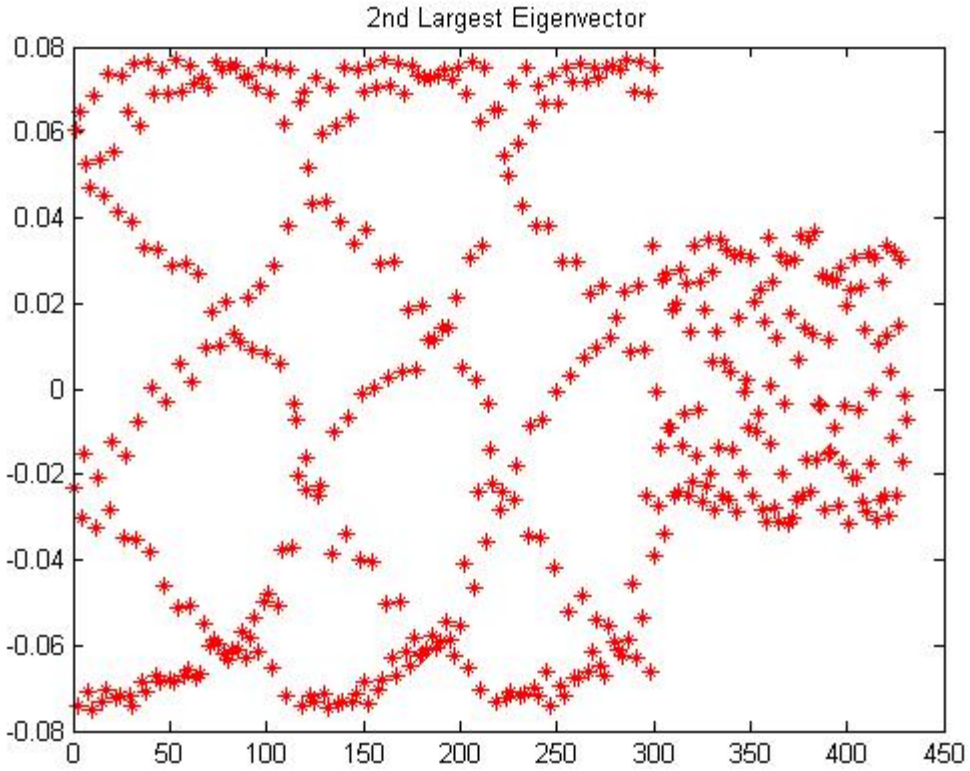


Example 2 “Bad” Scaling Factor (1.0)



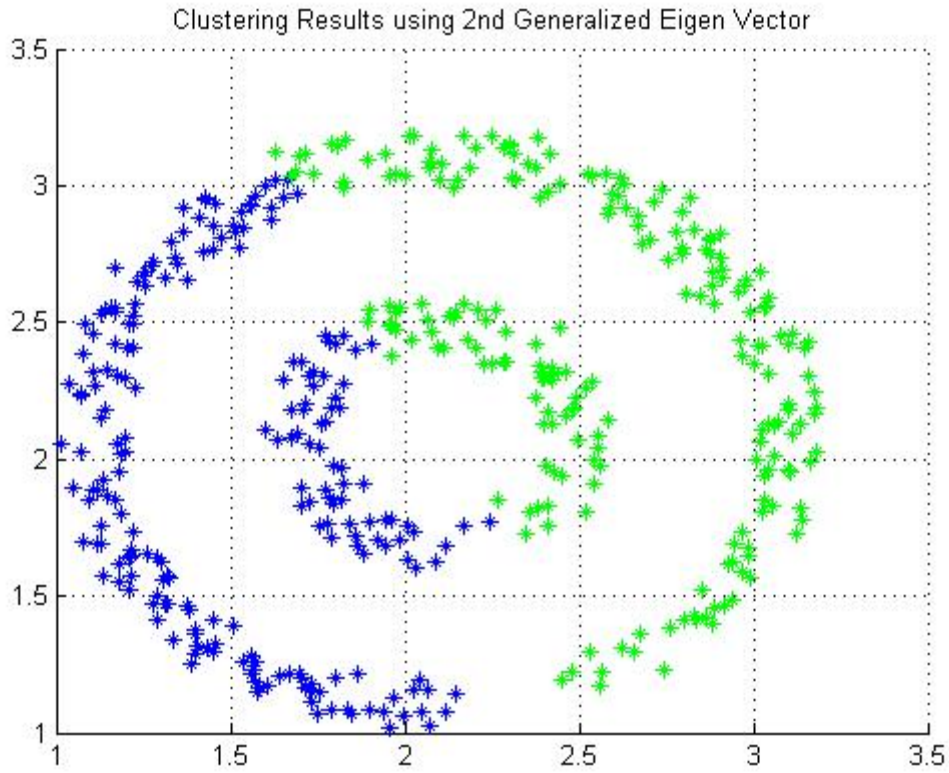
Example 2 “Bad” Scaling Factor (1.0)

- 2nd Largest EigenVector
- It is harder to separated the clusters



Example 2 “Bad” Scaling Factor (1.0)

- Clusters:
  The result is similar to what k-means would give



Outline

- Introduction to unsupervised learning
- K Means
- EM for GMMs
- Spectral Clustering
- Autoencoders for unsupervised learning

Auto-encoders for unsupervised learning

Autoencoders – the concept

- introduced by Hinton
  - address the problem of “backpropagation without a teacher”
  - Need to formulate an error function – using input data as the teacher [RUM1986]
- more recently, “deep architecture” approach (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Bengio and LeCun, 2007;Erhan et al., 2010)
  - Restricted Boltzmann Machines (RBMS), stacked, trained bottom up in unsupervised fashion.

Autoencoders

- Assume a set of $n\text{-}dim$ data X_n – we assume transformations f, g
  - f : *mapping data to a lower dim space* Z_m *(code)*
  - g: *mapping* Z_m *to the an original dim data* $X_n' \sim X_n$
- *Learning: minimize the loss* $L(x(f(g(x))$
  - *|code|< |input|: undercomplete AE*
  - *Sparse autoencoders:* $L(x(f(g(x)) + \Omega(Z_m)$

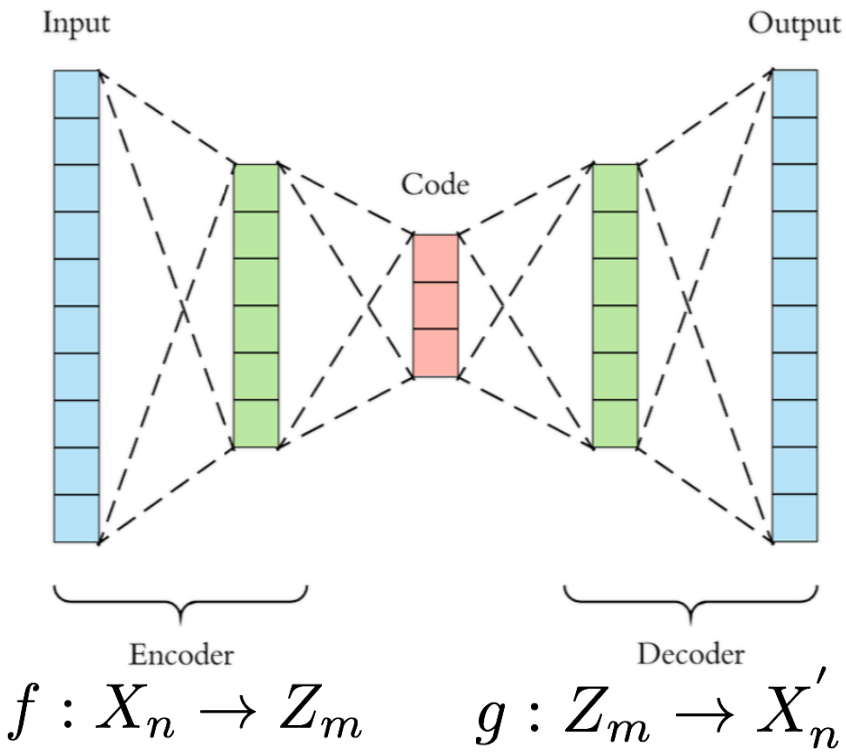
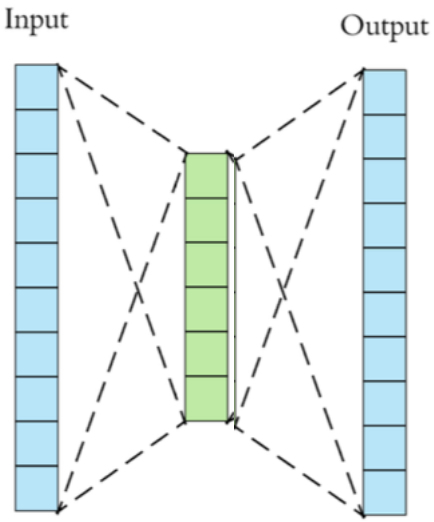


Image from [Dertat 2017]

Applications of AEs

- *Learn a low dimensional Z_m representation of the data X*
  - *learning low dimensional space with less reconstruction error than PCA (Hinton 2006)*
  - *Features used in classification and improve generalization (Hinton 2007)*
- *Perform clustering – unsupervised learning*
- *Information Retrieval*
  - *Semantic hashing for images and NLP (Hinton 2007, Torralba 2007)*
- *Learn data features in the absence of a supervised task*

Sparse Autoencoders (AE)



- Assume AE: n *input neurons*, m neurons in hidden layer and k training data
- Assume $a_j^{(2)}(x^{(i)})$ activation of j-*th* neuron in hidden layer for $x(i)$-*th* data point
- average output of j-*th* neuron of the hidden layer after data training:

$$\hat{\rho} = \frac{1}{k}\sum_{i}^{k}(a_j^{(2)}(x^{(i)}))$$

- Assume only few hidden layer neurons should fire for each training sample
  - output $\boldsymbol{\rho}$ of a neuron should be small (i.e. 0.02)
  - Penalize $\hat{\boldsymbol{\rho}}$ deviating significantly from $\boldsymbol{\rho}$:

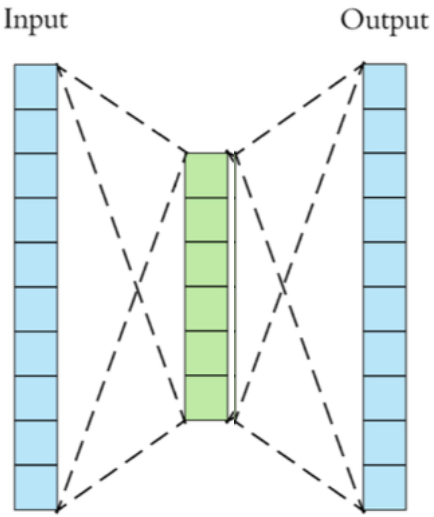
$$\sum_{j=1}^{m}\rho log\frac{\rho}{\hat{\rho}_j} + (1-\rho)log\frac{1-\rho}{1-\hat{\rho}_j} = \sum_{j=1}^{m}KL(\rho||\hat{\rho}_j)$$

  - KL divergence metric of distribution difference – sparsity term

Sparse Autoencoders (AE)

- The overall cost function:

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j)$$



- Need to incorporate the KL divergence in the back-propagation:
- The back-prop at the second (hidden) layer: $\delta_i^{(2)} = (\sum_{j=1}^{m} W_{ji}^{(2)} \delta_j^{(3)}) f'(z_i^{(2)})$
- Take into account sparsity:

$$\delta_i^{(2)} = (\sum_{j=1}^{m} W_{ji}^{(2)} \delta_j^{(3)}) + \beta(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}) f'(z_i^{(2)})$$

- Images that enable hidden units are image fundamental features (i.e. basic shapes)

Auto-encoders for clustering [Hie te al 2016]

- Deep Embedding Clustering (DEC)
  - Assume data X pf n points, cluster them into k. clusters in a lower dimensional space via a nonlinear mapping $f_\theta : X \to Z$, Z, θ: code, trainable parameters
  - learning simultaneously i. the center of the k clusters in the Z space and ιι. θ
  - *DEC phases:*
    - parameter initialization with a deep stacked autoencoder (Vincent 2010)
    - parameter optimization (i.e., clustering), where we iterate between computing an auxiliary target distribution and minimizing the Kullback–Leibler (KL) divergence to ιτ

Autoencoders for clustering [Hie te al 2016]

- soft assignment of embedded point $z_i = f_\theta(x_i)$ and the cluster centroids μ_j

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}$$

- iteratively refine clusters by learning from their high confidence assignments with the help of an auxiliary target distribution

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$$

- Where $f_j = \sum_i q_{ij}$ are the soft cluster frequencies.

- Then the error is $L = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$

Autoencoders for clustering [Hie te al 2016]

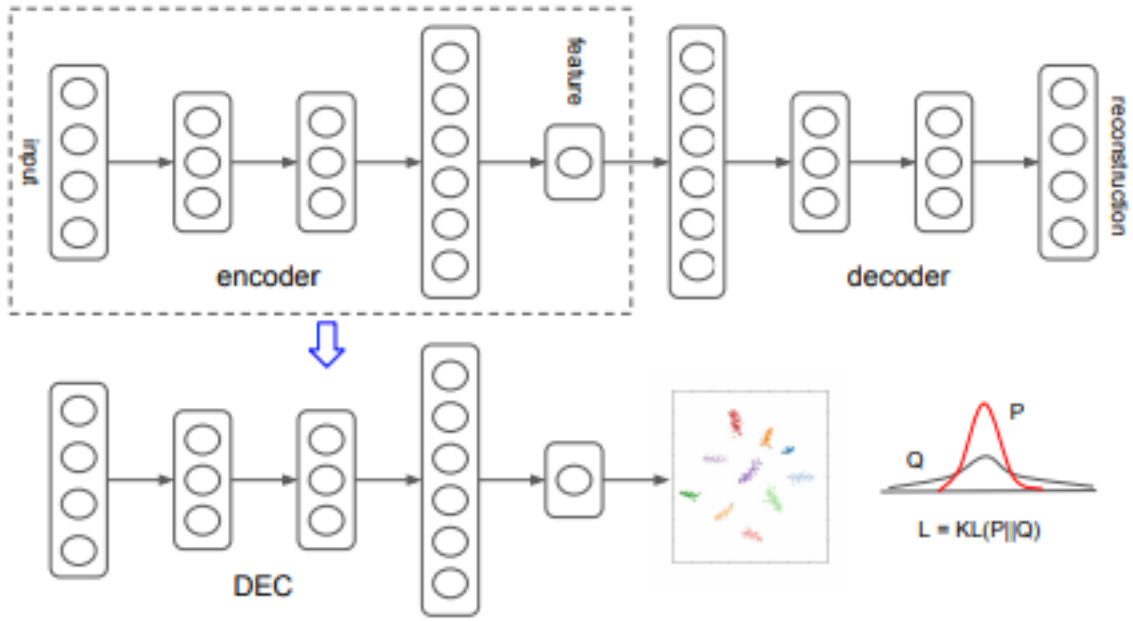
- jointly optimize cluster centers $\{\mu_j\}$ and DNN parameters θ using Stochastic Gradient Descent (SGD):

$$\frac{\partial L}{\partial z_i} = \frac{\alpha+1}{\alpha}\sum_j (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \times (p_{ij}-q_{ij})(z_i-\mu_j),$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha+1}{\alpha}\sum_i (1+\frac{\|z_i-\mu_j\|^2}{\alpha})^{-1} \times (p_{ij}-q_{ij})(z_i-\mu_j).$$

- gradients $\partial L/\partial zi$ are passed down to the DNN
- standard backpropagation compute DNN's parameter gradient $\partial L/\partial \theta$.
- discovering cluster assignments, stop when less than $tol\%$ of points change cluster assignment between consecutive iterations.

Autoencoders for clustering [Hie te al 2016]



- To initialize the cluster centers:
  - Pass data through the initialized DNN to get embedded data points and then perform standard k-means clustering in the feature space Z to obtain k initial centroids {μj} k j=1.

Autoencoders for clustering [Hie te al 2016]

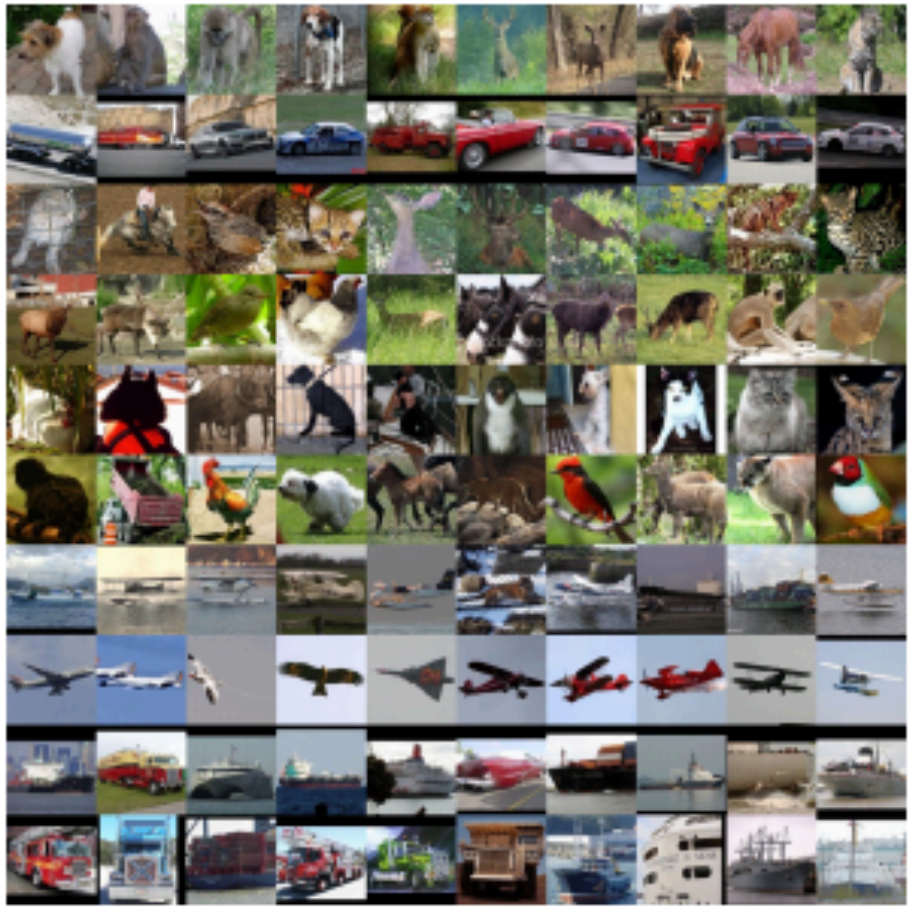
Table 1. Dataset statistics.

Dataset	# Points	# classes	Dimension	% of largest class
MNIST (LeCun et al., 1998)	70000	10	784	11%
STL-10 (Coates et al., 2011)	13000	10	1428	10%
REUTERS-10K	10000	4	2000	43%
REUTERS (Lewis et al., 2004)	685071	4	2000	43%

Method	MNIST	STL-HOG	REUTERS-10k	REUTERS
k-means	53.49%	28.39%	52.42%	53.29%
LDMGI	84.09%	33.08%	43.84%	N/A
SEC	80.37%	30.75%	60.08%	N/A
DEC w/o backprop	79.82%	34.06%	70.05%	69.62%
DEC (ours)	**84.30%**	**35.90%**	**72.17%**	**75.63%**

Autoencoders for clustering [Hie te al 2016]

(a) MNIST



(b) STL-10

- Top 10 results for each of the 10 clusers for the MNIST and STL data sets

Why is Unsupervised learning important

- Machine learning: learning representations (features, latent variables) to capture the statistical dependencies
- Supervised learning – from input variables to output variables, – i.e. categories
  - supervised learning require "teaching" computer concepts that matter to **humans**
  - supervised deep learning discover meaningful intermediate representations in their layers.
- Unsupervised learning
  - capture all possible dependencies between any subset of variables
  - human learning: based on observation and unsupervised interaction (action-perception loop)
- Issues
  - Training sets increasingly difficult to keep the pace with data production
  - Can we trust humans ?

"…hope is that deep unsupervised learning will be able to discover (possibly with a little bit of help from the few labeled examples we can provide) all of the concepts and underlying causes that matter (some being explicitly labeled, some remaining unnamed) to explain what we see around us. So I believe it is essential for approaching AI to make progress in this direction. And we are ;-)" Y. Bengio

References - Clustering

- *Arthur, D.; Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding", 18th symposium on Discrete algorithms. SIAM. pp. 1027–1035.*
- Teuvo Kohonen, Panu Somervuo, Self-organizing maps: Neurocomputing Volume 21, Issues 1–3, 6 November 1998, Pages
- An Impossibility Theorem for Clustering, Jon Kleinberg, NIPS 2003.

References Spectral Clustering

- *Lloyd, Stuart P. (1982), "Least squares quantization in PCM" IEEE Transactions on Information Theory, **28** (2): 129–137, doi:10.1109/TIT.1982.1056489.*
- A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of NIPS-14, 2001*
- J. Shi and J. Malik. Normalized cuts and image segmentation, *IEEE Trans. Pattern Analysis and Machine Intelligence, 22(8):888–905, August 2000.*
- S. X. Yu and J. Shi. "Multiclass spectral clustering", in *International Conference on Computer Vision, 2003.*
- *Inderjit S. Dhillon, Yuqiang Guan, Brian Kulis , "Kernel k-means, Spectral Clustering and Normalized Cuts"*, in proceedings of the ACM KDD 2004
- "Graph clustering", Satu Elisa Schaeffer, COMPUTER SCIENCE REVIEW1(2007)27–64
- A Tutorial on Spectral Clustering, Ulrike von Luxburg, March 2007.
- P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," Journal of the Royal Statistical Society, Series B (Methodological), vol. 39, no. 1, pp. 1–38, 1977.

References - Autoencoders

- [Baldi, 2012] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In Proceedings of ICML workshop on unsupervised and transfer learning, pages 37–49.
- [Rum1986] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.
- [Bengio and LeCun, 2007] Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards ai. Large-scale kernel machines, 34(5) :1–41.
- [Dertat, 2017] Dertat, A. (2017). Applied deep learning - part 3 : Autoencoders, medium post, towards data science.
- [Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. science, 313(5786) :504–507.
- [Hou et al., 2017] Hou, X., Shen, L., Sun, K., and Qiu, G. (2017). Deep feature consistent variational autoencoder. In Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, pages 1133–1141. IEEE.
- [Hie te al 2016] Unsupervised Deep Embedding for Clustering Analysis, J. Xie, R. Girschick, A. Farhadi, ICML 2016

References - Autoencoders

- [Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pages 1096–1103. ACM.
- [Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(Dec) :3371–3408.
- Restricted Boltzman Machines, https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf