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

October, 2024

Outline

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 $x_1, x_2, \ldots$, 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*.
- Examples: *clustering, dimensionality reduction, deep learning embeddings*.

Relation of UL to Statistics and Information theory

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

$$E(x) = -\sum_i P(x_i)\log_2(P(x_i))$$

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

$$-\sum_i P(x_i)\log_2(Q(x_i))$$

Relation of UL to Statistics and Information theory

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

$$KL(P\|Q) = \sum_i P(x_i) \log(\frac{P(x_i)}{Q(x_i)})$$

- 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 $P(x) = Q(x)$.
- As Q approaches P we 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: $P(\omega_m|D) = P(\omega_m)\prod_{i=1..n} P(i|\omega_m)$, where i is a data sample

– *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 i. i. d. 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 ω_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 maximum a posteriori probability (MAP) values given the data set and the model

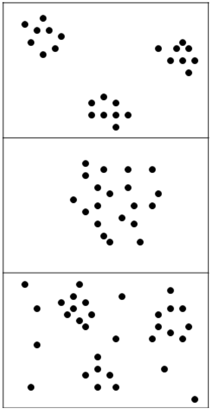
$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta}[\log(P(\theta|\omega_m)) + \sum_n \log(P(x_n|\theta,\omega_m))]$$

- Or find the parameter than maximizes likelihood:

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

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

Clustering



- ”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?

- Clustering starts in early 20th century:
  - Quantitative Expression of Cultural Relationships in tribes [1]
  - Traits grouping based personality description [2]
- "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

[1] Driver and Kroeber (1932)."Quantitative Expression of Cultural Relationships".

[2] Cattell, R. B. (1943a). The description of personality: I. Foundations of trait measurement. Psychological Review, 50, 559–594. View

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 vs. 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

k-Means

Known by different names: dynamic clustering method, iterative minimum-distance clustering, nearest centroid sorting, h-means, ... Four algorithms origin:

- Steinhaus (1956) “Sur la Division des Corps Matériels en Parties” - first that proposed explicitly algorithm for multidimensional instances.
- Lloyd (1957) @Bell Labs, problem of transmitting a random signal X in a multidimensional space, presented as a technique for PCM
- MacQueen (1967) - algorithm for partitioning data into a set of clusters: variance was small for each cluster, coined the term *k*-means.
- Jancey (1966) - “Multidimensional Group Analysis””: a clustering method characterizing species Phyllota phylicoides.

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

$$J(k) = \sum_{j=1}^{k} \sum_{i \in C_k} \|x_i - c_j\|^2$$

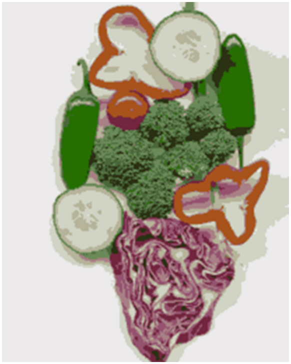
- c_j is the center of cluster C_k.
- Thus the clustering that minimizes the distances to cluster centers is retrieved.
- *NP*-Hard problem, even for $k = 2$.
- Most popular heuristic: Lloyd’s algorithm.

Lloyd's heuristic algorithm

- Algorithm:
  1. Arbitrarily choose an initial k centers $C = \{c_1, c_2, \dots, c_k\}$;
  2. For each $i \in \{1, ..., k\}$, cluster C_i: set of points in X that are closer to c_i than they are to c_j for all $j \neq i$;
  3. For each $i \in \{1, ..., k\}$, set c_i = center of mass of C_i points: $c_i = \frac{1}{|Ci|} \sum_{x \in C_i} x$;
  4. Repeat Steps 2, 3 until C no longer changes or until error $J < \theta$;
- Advantages:
  - Fast and effective method for large datasets.
- Disadvantages:
  - Works well only for convex and dense clusters.
- Time complexity: $\mathcal{O}(I\, e\, \underline{n\, k})$
  - I = number of iterations (steps)
  - e = cost of distance computation
  - k, n = number of dimensions/data points (resp)

k-Means application on image compression

Image

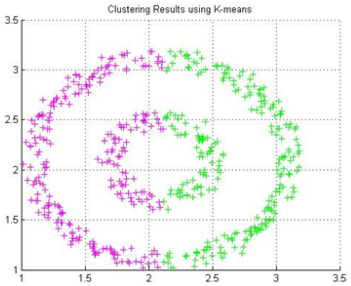

Clusters on color

k-Means clustering of RGB (3 value) pixel color intensities, $k = 11$ segments

(@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-Medoids, ...)
- Choice of distance measure
- Selection of # clusters k : gap [3] criterion i.e. assume $J(k)$ error for clustering a dataset D, k #clusters: search for $argmax_{k+1}(J(k) - J(k+1))$, k# of clusters in $[1, |D|]$

[3] Tibshirani, R. , Walther, G. and Hastie, T. (2001), Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63: 411-423. doi:10.1111/1467-9868.00293

K-means ++: the power of seeding

- choice of k centers is crucial
- many natural examples where k-means generates arbitrarily bad clusterings.
- if centers are chosen uniformly at random from the data points
- K-means++ [4] proposes a way choosing centers for the k-means algorithm:
    - $D(x)$: shortest distance from a data point to the closest center already chosen,
    - **k-means++**
      1a Assume cluster center c_1, chosen uniformly at random from X ;
      1b. Add new cluster center $c_i : x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$;
      2-4. Proceed as with the standard k-means algorithm. ;
    - with k-means++ error is bounded as: $E[J] \leq 8(\ln k + 2)J_{opt}$

[4] D. Arthur, S. Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In Proceedings of ACM-SIAM symposium on Discrete algorithms (SODA '07), pp. 1027-1035.

K-means ++: the power of seeding

K-means++ i.achieves better performance and ii. consistently finds a better clustering than k-means

	Average ϕ		Minimum ϕ		Average T	
k	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	10898	5.122	2526.9	5.122	0.48	0.05
25	787.992	4.46809	4.40205	4.41158	1.34	1.59
50	3.47662	3.35897	3.40053	3.26072	2.67	2.84

Table 1: Experimental results on the *Norm-10* dataset (n = 10000, d = 5)

	Average ϕ		Minimum ϕ		Average T	
k	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	135512	126433	119201	111611	0.14	0.13
25	48050.5	15.8313	25734.6	15.8313	1.69	0.26
50	5466.02	14.76	14.79	14.73	3.79	4.21

Table 2: Experimental results on the *Norm-25* dataset (n = 10000, d = 15)

	Average ϕ		Minimum ϕ		Average T	
k	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	7553.5	6151.2	6139.45	5631.99	0.12	0.05
25	3626.1	2064.9	2568.2	1988.76	0.19	0.09
50	2004.2	1133.7	1344	1088	0.27	0.17

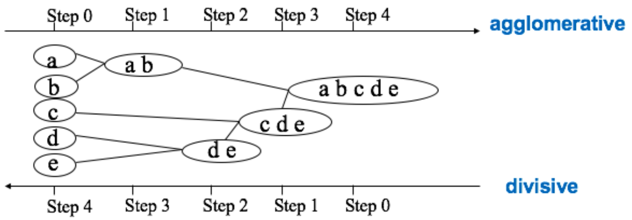
Table 3: Experimental results on the *Cloud* dataset (n = 1024, d = 10)

	Average ϕ		Minimum ϕ		Average T	
k	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	$3.45 \cdot 10^8$	$2.31 \cdot 10^7$	$3.25 \cdot 10^8$	$1.79 \cdot 10^7$	107.5	64.04
25	$3.15 \cdot 10^8$	$2.53 \cdot 10^6$	$3.1 \cdot 10^8$	$2.06 \cdot 10^6$	421.5	313.65
50	$3.08 \cdot 10^8$	$4.67 \cdot 10^5$	$3.08 \cdot 10^8$	$3.98 \cdot 10^5$	766.2	282.9

Table 4: Experimental results on the *Intrusion* dataset (n = 494019, d = 35)

Error (ϕ), execution time ($\mathcal{T}$) comparison k-means vs. k-means++

Hierarchical Clustering

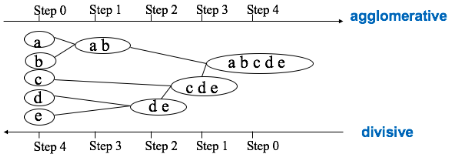


- Two basic approaches:
  - merging smaller clusters into larger ones (agglomerative),
  - splitting larger clusters (divisive)
- Visualize both via "dendograms"
  - shows nesting structure
  - merges or splits: tree nodes

Hierarchical Clustering - Merging criteria

Basic Algorithm

- Start with each point as a cluster
- Repeat
    - Find the *closest pair* of clusters
    - Merge them
- until there is only one cluster
- Return the tree of cluster-mergers



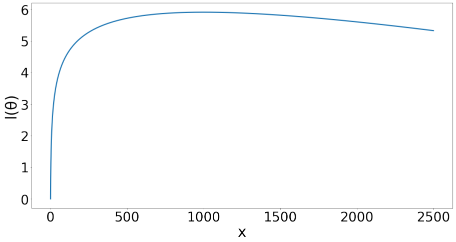
Distance computation
Assume clusters C_1, C_2, their distance can me computed with alternative ways

- *Single link* i.e. the shortest distance between any two points of the clusters.: $D(C_1, C_2) = min_{x \in C_1, y \in C_2} |x - y|$
- *Complete link* i.e. the largest distance between any two points of the clusters: $D(C_1, C_2) = max_{x \in C_1, y \in C_2} |x - y|$
- *Ward's method*: distance between two clusters, C_1, C_2: is the augmentation of error value when we merge clusters C_1, C_2 we merge them: $D(C_1, C_2) = \sum_{x_i \in C_1 \cup C_2} |x_i - m_{C_1 \cup C_2}|^2 - \sum_{i \in C_1} |x_i - m_{C_1}|^2 - \sum_{x_i \in C_2} |x_i - m_{C_2}|^2$

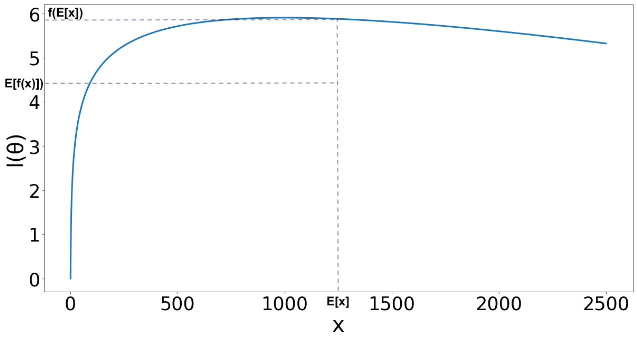
Outline

Expectation Maximization

- Assume a data set X and probability distribution $p(X|\theta)$
- Its likelihood: $\mathcal{L}(\theta) = \prod_{i=1}^{N} p(x_i|\theta)$
- Objective: Find θ that maximize $\mathcal{L}$
  - $\mathcal{L}$ tends to be a problematic function as for many points $p(x_i|\theta)$ tend to be very small numbers
- We consider log-likelihood $\ell(\theta) = log \prod_{i=1}^{N} p(x_i|\theta) = \sum_{i=1}^{N} log(p(x_i|\theta)$
- This is a much more friendly to optimize - concave function



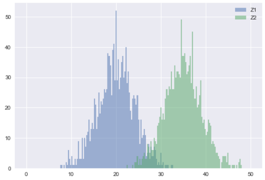
Jensen Inequality



- for a concave function $f(x)$ we know $f''(x) < 0$ and Jensen's inequality: $f(E[x]) \geq E[f(x)]$
- vice versa for a convex function $f(x)$: $f''(x) > 0$ and Jensen's inequality: $f(E[x]) \leq E[f(x)]$

Expectation Maximization

- Assume a training set **x** of n observations $x_1, ..., x_n$, n independent examples.
- we search for a parametric model $p(\mathbf{x},\mathbf{z})$ where **z** a set of k classes/clusters/latent variables parameterized on θ.
- The log-likelihood to optimize is:
$\ell(\theta) = \sum_{i=1}^{n} log(p(x_i|\theta)) = \sum_{i=1}^{n} log\left(\sum_{j=1}^{k} p(x_i, z_j|\theta)\right)$
- Maximizing $\ell(\theta)$ explicitly can be tedious: employ a greedy approach (EM)
    - (E-step) repeatedly compute a lower-bound on $\ell(\theta)$,
    - (M-step) optimize that lower-bound



Expectation Maximization

- Assume a distribution Q over the z_k clusters (i.e. each cluster is a Gaussian). Then:
$\ell(\theta) = \sum_{i=1}^{n} log(p(x_i|\theta)) = \sum_{i=1}^{n} log\left(\sum_{j=1}^{k} p(x_i, z_j|\theta)\right) =$
$\sum_{i=1}^{n} log\left(\sum_{j=1}^{k} Q(z_j)\frac{p(x_i,z_j|\theta)}{Q(z_j)}\right)$
- assuming Jensen inequality: $f(E[x]) \geq E[f(x)]$
- as $f(x) = log(x)$ a concave function, since $f''(x) = -\frac{1}{x^2} \leq 0$ for $x \in R^+$
- since $\sum_{j=1}^{k} Q(z_j)\frac{p(x_i,z_j|\theta)}{Q(z_j)}$ is the expectation of $\frac{p(x_i,z|\theta)}{Q(z)}$ wrt z over distribution Q then we deduce:

$$f(E_{Q(z)}[\frac{p(x_i,z|\theta)}{Q(z)}]) \geq E_{Q(z)}[f(\frac{p(x_i,z|\theta)}{Q(z)})]$$

Expectation Maximization - E-step

- Assume some initial values for θ
- need to set a lower bound making the inequality:
  $f(E_{Q(z)}[\frac{p(x_i,z|\theta)}{Q(z)}]) \geq E_{Q(z)}[f(\frac{p(x_i,z|\theta)}{Q(z)})]$
  to hold with equality for the particular value of θ.
- Therefore, we assume Q the posterior distribution of z given x_i with the parameters θ set.: $Q(z) = p(z|x_i, \theta) = \frac{p(x_i,z|\theta)}{\sum_j p(x_i,z_j|\theta)}$
- Thus $f(E_{Q(z)}[\frac{p(x_i,z|\theta)}{Q(z)}]) \geq E_{Q(z)}[f(\frac{p(x_i,z|\theta)}{Q(z)})]$

  gives a lower-bound on the loglikelihood that we're trying to maximize.

Expectation Maximization - M-step

- In M-step maximize $E_{Q(z)}[f(\frac{(p(x_i,z|\theta))}{Q(z)})] = \sum_{i=1}^{n} \sum_{j=1}^{k} Q(z_j) log \frac{(p(x_i,z_j|\theta))}{Q(z_j)}$ with respect to the parameters to obtain a new setting of θ.
- We repreat the E and M steps until convergence.

Repeat until convergence

1. (E-step) For each i, set $Q(z_k) = p(z_k|x_i,\theta) = \frac{p(x_i,z_k|\theta)}{\sum_j p(x_i,z_j|\theta)}$
2. (M-step) Set $\theta = argmax_{\theta} \sum_{i=1}^{n} \sum_{j=1}^{k} Q(z_j) log \frac{p(x_i,z_j|\theta)}{Q(z_j)}$

Expectation Maximization - Mixture of Gaussians

- Assume a data set X as a mixture of K different classes/processes with respective w_k weights, therefore $f(x) = \sum_k w_k N(\mu, \Sigma)$
- therefore the log likelihood to be maximized is:

$$\ell(\mathbf{w}, \mu, \Sigma) = \sum_{i=1}^{n} \sum_{j=1}^{k} \text{Ind}\,(z_i = j)\; log \left(\frac{w_j exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j))}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \right)$$

- derivative with respect to μ_k and set to 0

$$\nabla_{\mu_k} E_{Q(z)}[\ell(\mathbf{w}, \mu, \Sigma)] = \text{... proof in the lab ...} \; = \sum_{i=1}^{n} \gamma_{ik} (\Sigma_k^{-1} x_i - \Sigma_k^{-1} \mu_k),$$

where $\gamma_{ik} = \frac{w_k N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j N(x_i|\mu_j, \Sigma_j)}$ is called the “responsibility” of the normal distribution k for data point x_i.

Expectation Maximization - Mixture of Gaussians

- ... $\sum_{i=1}^{n} \gamma_{ik}(\Sigma_k^{-1} x_i - \Sigma_k^{-1}\mu_k)$ setting this to 0 we get:

$$\mu_k = \frac{\sum_{i=1}^{n} \gamma_{ik} x_i}{\sum_{i=1}^{n} \gamma_{ik}}$$

- Similarly we get:

$$w_k = \frac{1}{n}\sum_{i=1}^{n} \gamma_{ik}$$

- and:

$$\Sigma_k = \frac{\sum_{i=1}^{n} \gamma_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} \gamma_{ik}}$$

Expectation Maximization for 2 Gaussians

- 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

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

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

- Membership of x_i to each of the two Gaussians

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

similar for $y_{i,2}$.

Expectation Maximization for 2 Gaussians

Maximization step

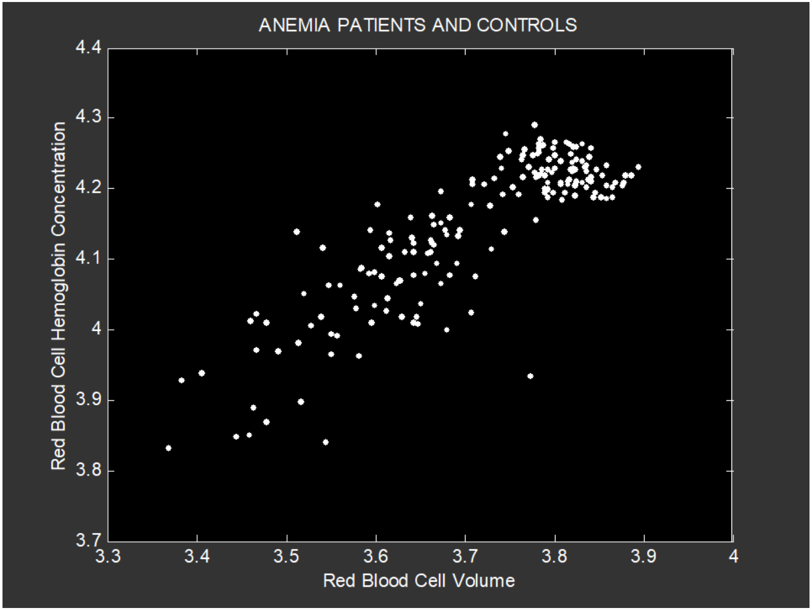
- Using values for memberships $y_{i,j}$
- compute new values for distribution parameters.

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

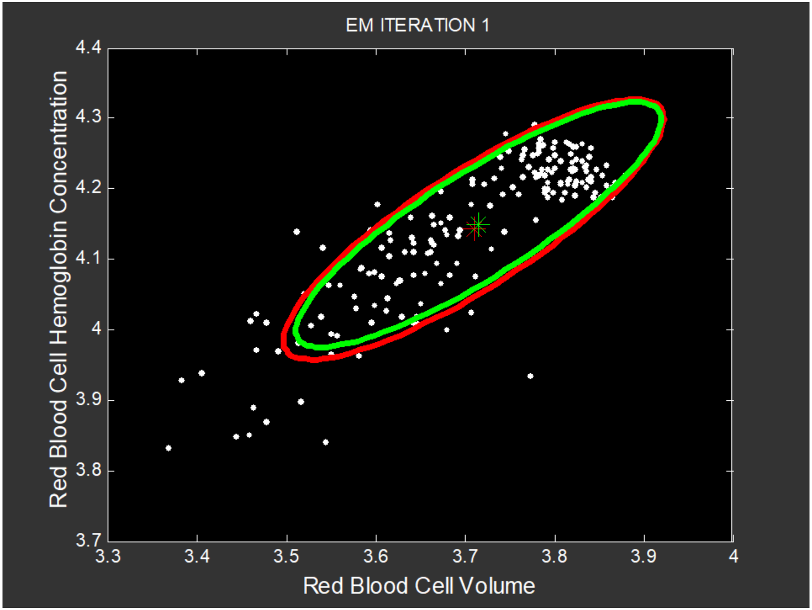
N: number of data points. Same for cluster 2.

- back to Expectation step: Re-compute new membership values
- repeated until change in mixture model parameters below threshold

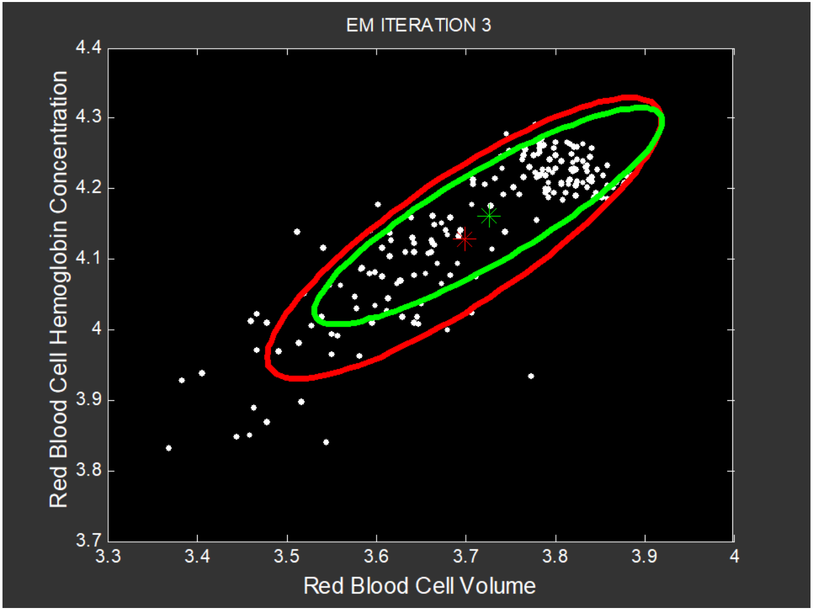
Expectation Maximization for 2 Gaussians



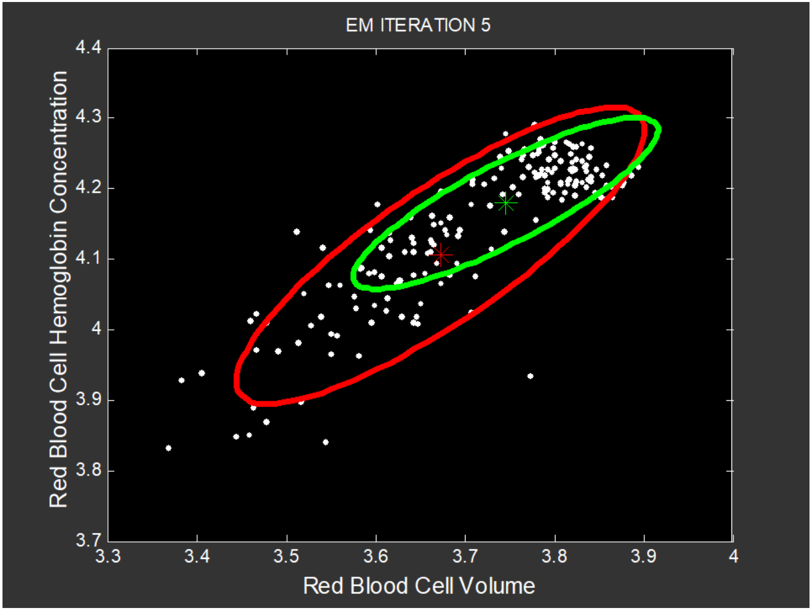
Expectation Maximization for 2 Gaussians



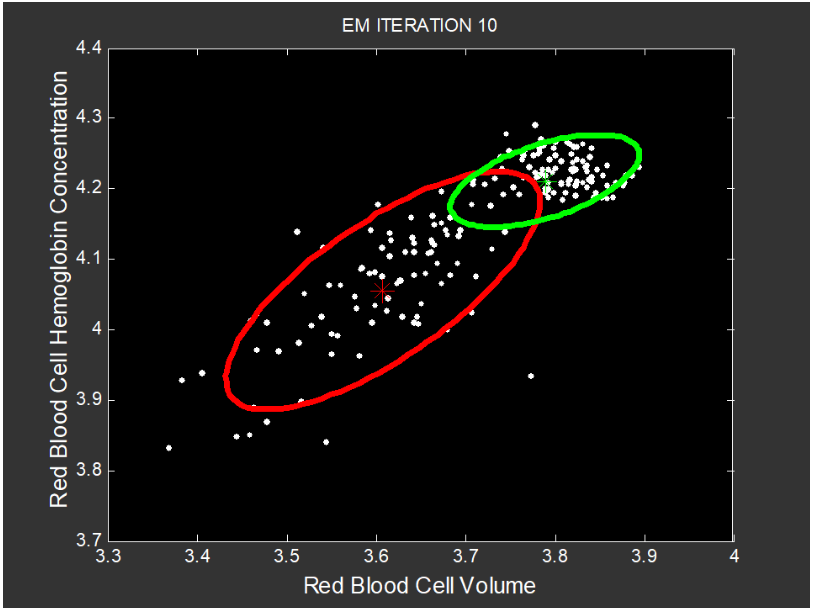
Expectation Maximization for 2 Gaussians



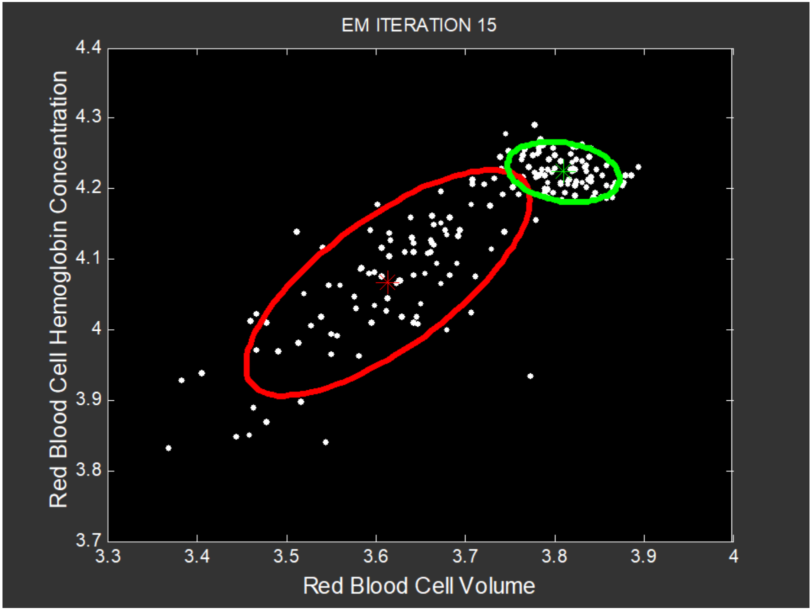
Expectation Maximization for 2 Gaussians



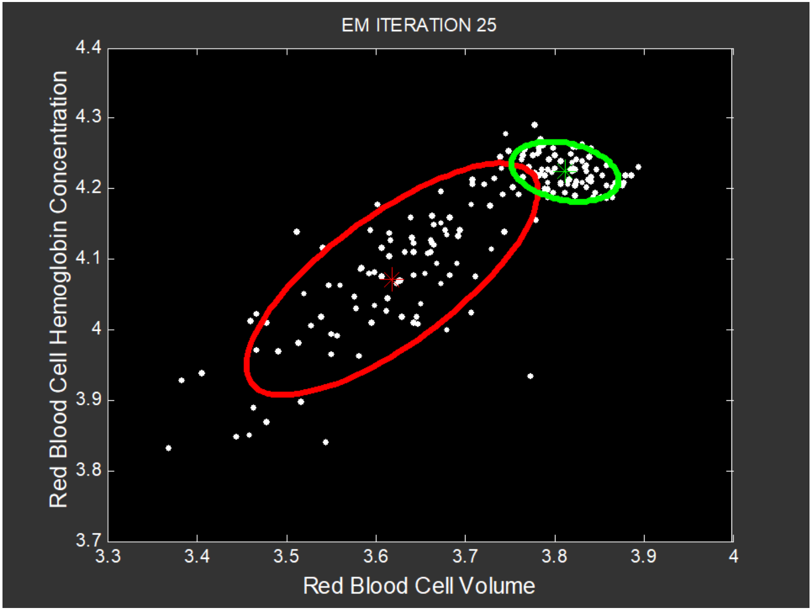
Expectation Maximization for 2 Gaussians



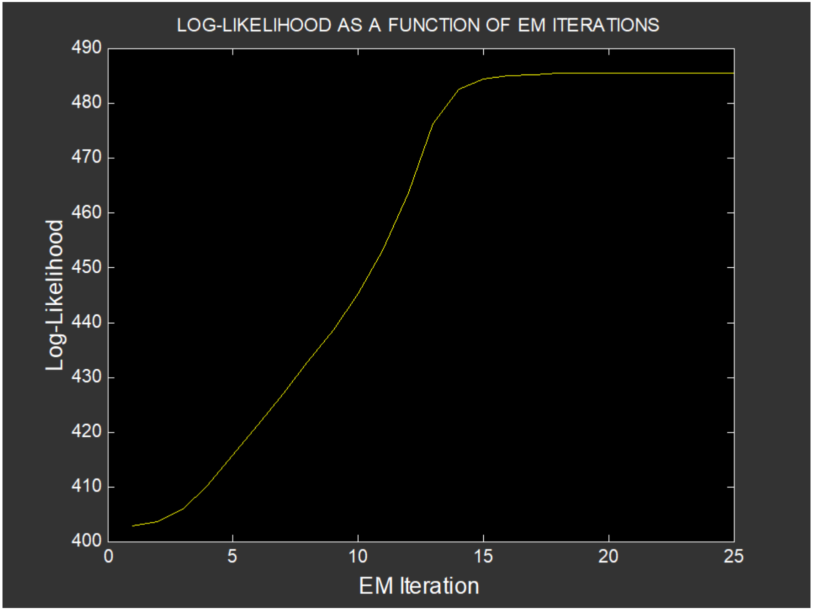
Expectation Maximization for 2 Gaussians



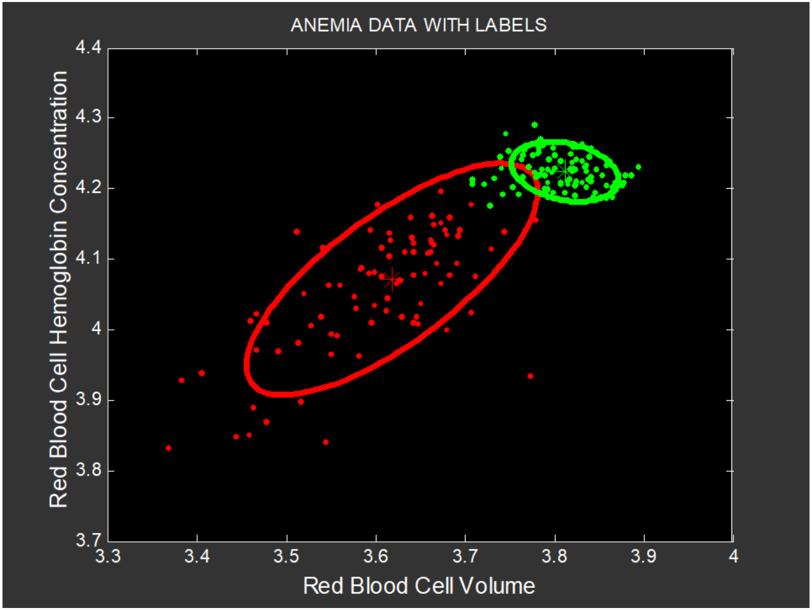
Expectation Maximization for 2 Gaussians



Expectation Maximization for 2 Gaussians



Expectation Maximization for 2 Gaussians



Outline

Spectral Clustering

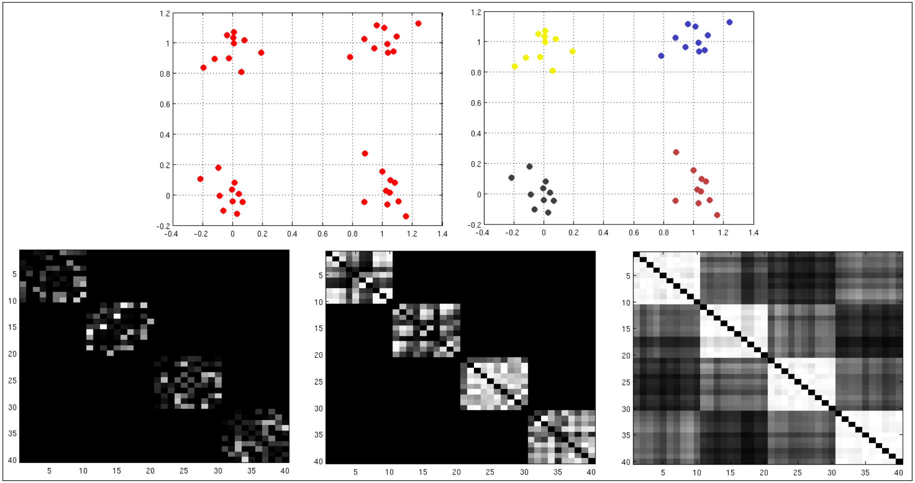
Aim is to cluster/partition datasets mapped to a graph. Graphs can be obtained from datasets using one of the the following three approaches:

- the ϵ-*neighbourhood graph* data points are joined by an edge if their pairwise distance is smaller than ϵ.
- In the *k-nearest neighbour graphs* each data point or vertex v_i is connected to another vertex v_j if v_j is amongst the k-nearest neighbours of v_i.
- In the *fully connected graph* all pairs of data points with a positive similarity in a given similarity measure are connected. All edges are weighted by their similarity using for example in the Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$, where the parameter σ controls the width of the neighbourhood.

Fully connected graph — Affinity Matrix: pairwise similarity

- Affinity Matrix $A \in \mathbb{R}^{n \times n}$
- Define $A_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ 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

Affinity Matrix

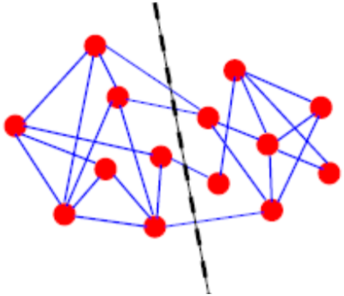


http://www.cs.utah.edu/~jfishbau/advimproc/project6/images/pts_dist_example.png

Graph partitioning – *the CUT concept*

- Derive clustering minimizing the sum of weights on edges between objects in different clusters.

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



- For *k*-clusters.

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

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

Objective Functions

- Two popular objective functions that balance for cluster sizes:

$$\text{RatioCut}(A_1, \ldots, A_k) = \sum_{1}^{k} \frac{cut(A_i, \hat{A})}{|A_i|}$$

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{1}^{k} \frac{cut(A_i, \hat{A})}{vol(A_i)}$$

- 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.

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

- Equivalent Trace minimization problem

$$\min_{Y} \text{Tr}(Y^T(D - W)Y)$$

- 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)

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$$

- Equivalent Trace optimization problem

$$\min_{Y} \text{Tr}(Y^T(I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}})Y)$$

- 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.

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

- 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

Laplacian Matrix

- In the case of
  - data points: distance Matrix (A)
  - graphs: adjacency matrix
- Laplacian Matrix

Un-normalized:	$L = D - A$
Normalized (symmetric):	$L^{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
Normalized (random walk):	$L^{\text{ran-w}} = D^{-1} L$

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

Laplacian Matrix

The matrix $L = D - W$ satisfies the following properties:

1. For every vector $\boldsymbol{f}$ in $\mathbb{R}^n$:

$$\boldsymbol{f}' L \boldsymbol{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (\boldsymbol{f}_i - \boldsymbol{f}_j)^2$$

2. L is symmetric and positive semi-definite
3. Min eigenvalue of L is 0
4. L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
5. multiplicity of 0 eigenvalue is the number of connected components in the graph. The corresponding eigenvectors identify these connected components.

Graphs with sparsely connected to each other clusters can be seen as a perturbation of a graphs of connected components. Perturbed Laplacian eigenvectors are close the the ideal indicator vectors. (von Luxburg, 2007, p.406)

Spectral Clustering Algorithms

clustering ($k > 2$) – Un-normalized

Input: Affinity matrix $W \in \mathbb{R}^{n \times n}$, number of clusters n

Compute Laplacian L;

Compute the top k eigenvectors $u_1, \dots, u_k$ of L;

Let $U^{n \times k}$ matrix with vectors $u_1, \dots, u_k$ as columns;

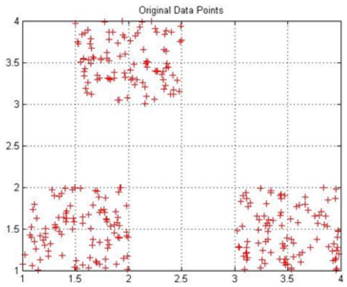
Apply k-means on the rows y_j of the $U^{n \times k}$ matrix;

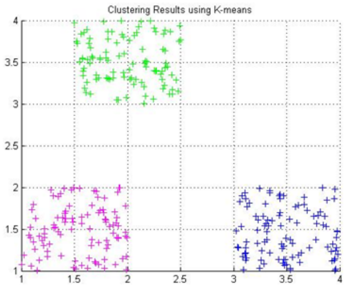
Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j | y_j \text{ belongs to } A_i\}$

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

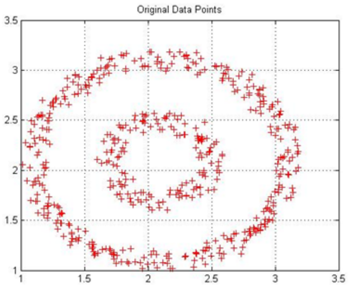
k-Means

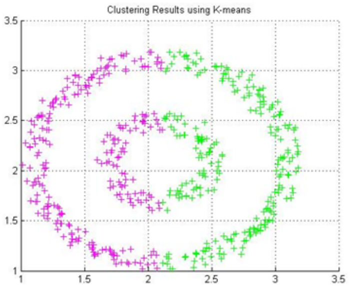




- *k*-Means is good at finding dense areas that are defined by a convex region.

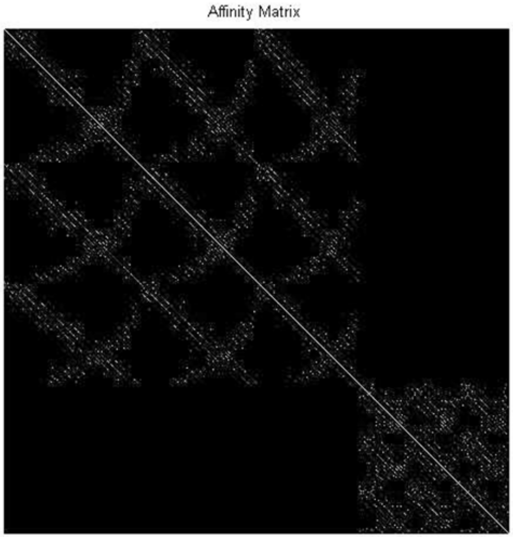
k-Means





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

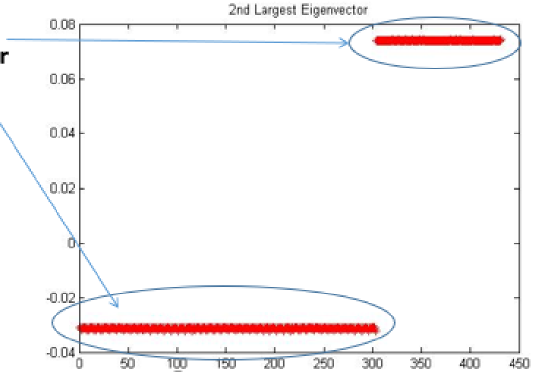
Example 1 – ”Good” Scaling Factor (0.1)



Example 1 – ”Good” Scaling Factor (0.1)

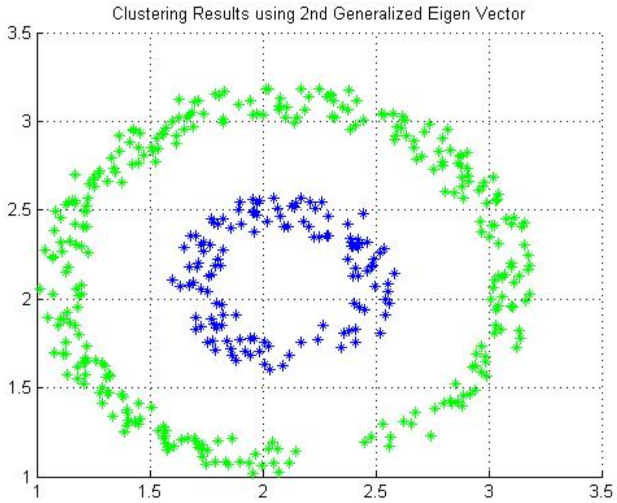
- 2nd Largest EigenVector

The values of the eigenvector are easily separable into two clusters

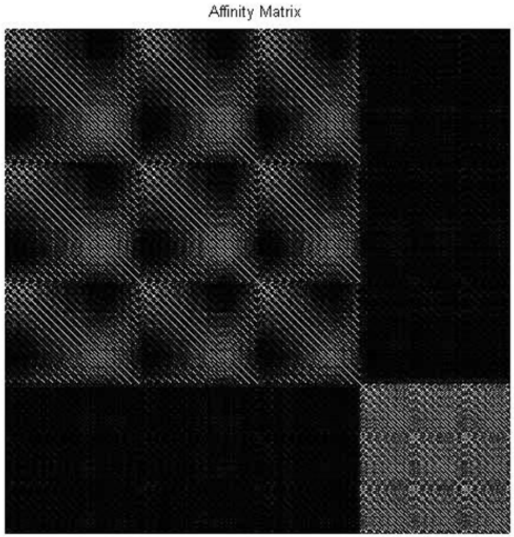


Example 1 – ”Good” Scaling Factor (0.1)

- Clusters

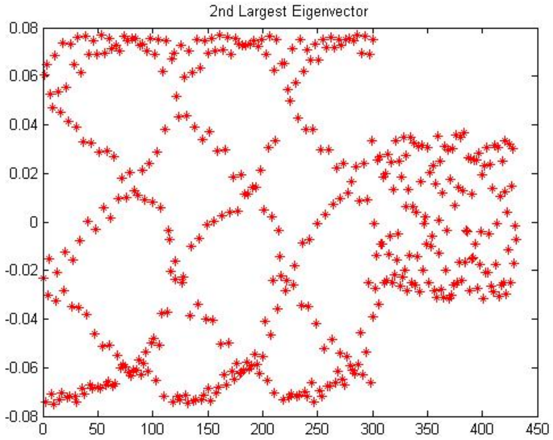


Example 2 – ”Bad” Scaling Factor (1.0)



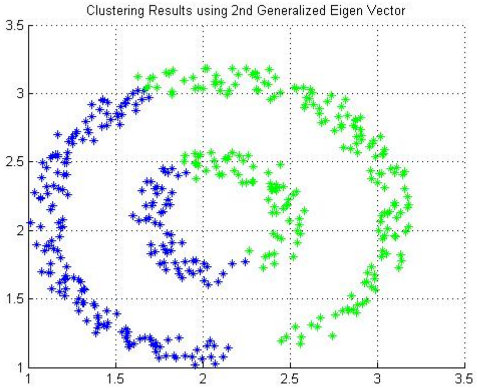
Example 2 – ”Bad” Scaling Factor (1.0)

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



Example 2 – ”Bad” Scaling Factor (1.0)

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



- Clearly when $\sigma = 1$ the edge weights connecting data points in the inner and outer ring are too large.

Unsupervised Learning - Conclusions

Unsupervised learning is the most promising part of AI...

- Supervised learning - require ”teaching” computer concepts that matter to humans - human learning based on observation and unsupervised interaction (action-perception loop)
- Training sets increasingly difficult to keep the pace with data production Can we trust humans ?
- Unsupervised learning captures all possible dependencies between *any subset of variables - will revisit this in deep leaning/autoencoders*

“. . . 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**

Acknowledgments

Thanks to *Dr. Johannes Lutzeyer@DaSciM* for the comments and improvement on the EM and Spectral clustering parts.

References Clustering I

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 (6 November 1998).
Self-organizing maps
Neurocomputing Volume 21. Issues 1–3.

Jon Kleinberg (2003)
An Impossibility Theorem for Clustering
Advances in Neural Information Processing Systems

References Spectral Clustering I

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. (2001).
On spectral clustering: Analysis and an algorithm
Proc. of NIPS-14

J. Shi and J. Malik. (August 2000)
Normalized cuts and image segmentation
IEEE Trans. Pattern Analysis and Machine Intelligence 22(8):888–905.

S.X.Yu and J.Shi. (2003)
Multiclass spectral clustering
International Conference on Computer Vision

Inderjit S. Dhillon, Yuqiang Guan, Brian Kulis (2004)
Kernel k-means, Spectral Clustering and Normalized Cuts
proceedings of the ACM KDD

References Spectral Clustering II

Satu Elisa Schaeffer (2007)
Graph clustering
COMPUTER SCIENCE REVIEW 1(2007)27–64.

Ulrike von Luxburg (March 2007)
A Tutorial on Spectral Clustering

P.Dempster, N.M.Laird, and D.B.Rubin (1977)
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.