Clustering

Sriram Sankararaman

Outline

Supervised vs Unsupervised learning

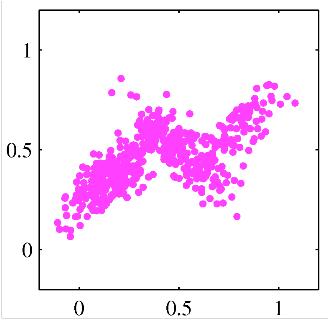
- Supervised learning: Given $(x_i, y_i), i = 1, \ldots, n$, learn a function $f : X \rightarrow Y$.
    - Categorical Y: classification
    - Continuous Y: regression
- Unsupervised learning: Given only $(x_i), i = 1, \ldots, n$, can we infer the underlying structure of X?

Why do unsupervised learning?

- Raw data cheap. Labeled data expensive.
- Save memory/computation.
- Reduce noise in high-dimensional data.
- Useful in exploratory data analysis.
- Often a pre-processing step for supervised learning.

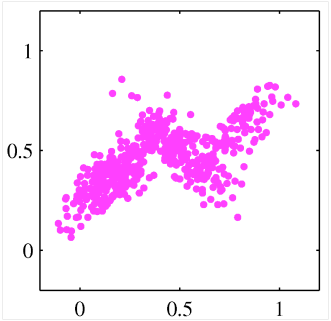
Cluster analysis

Discover groups such that samples within a group are more similar to each other than samples across groups.

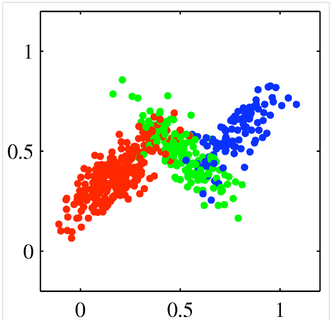


Cluster analysis

Discover groups such that samples within a group are more similar to each other than samples across groups.

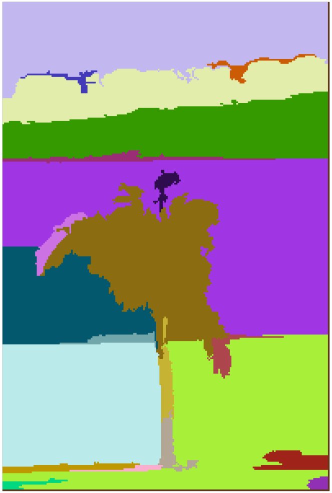


$\rightarrow$



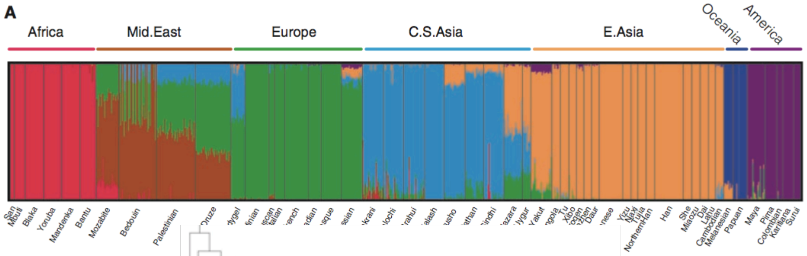
Outline

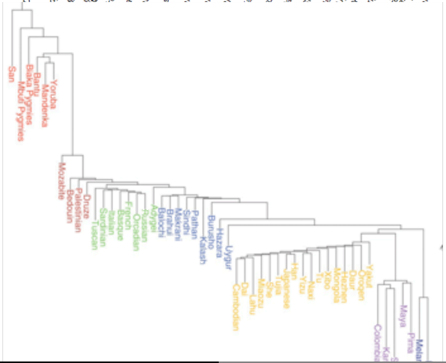
Image Segmentation



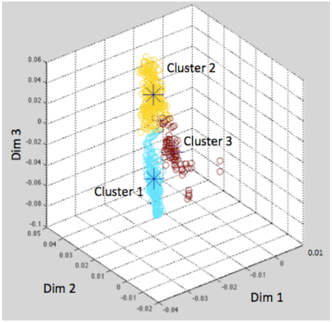
http://people.cs.uchicago.edu/ pff/segment

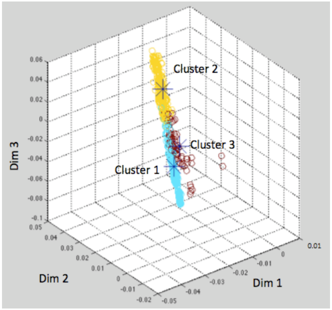
Human population structure





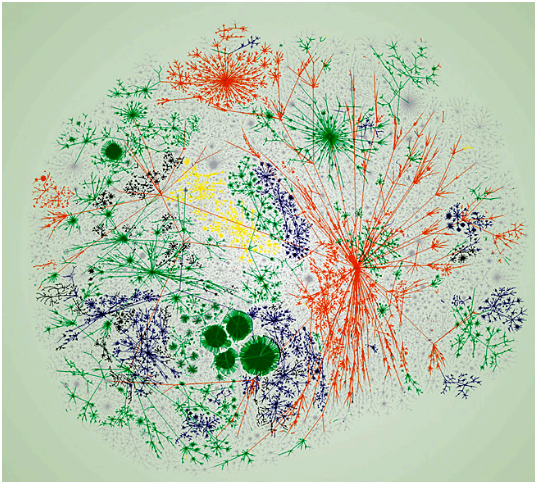
Clustering Web2.0 workloads





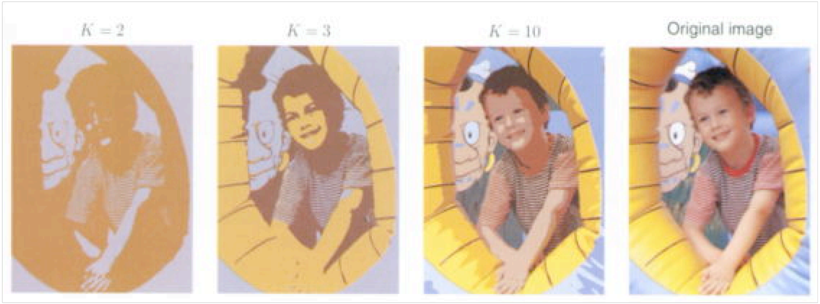
Courtesy: Archana Ganapathi

Clustering graphs



Newman, 2008

Vector quantization to compress images



Bishop, PRML

Ingredients of cluster analysis

- A dissimilarity function between samples.
- A loss function to evaluate clusters.
- Algorithm that optimizes this loss function.

Outline

The Dissimilarity function

- Choice of dissimilarity function is application dependent.
- Need to consider the type of features.
    - Categorical, ordinal or quantitative.
- Possible to learn dissimilarity from data (later).

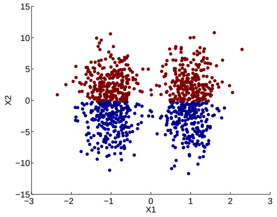
Dissimilarity based on features

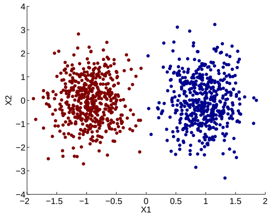
- Data point x_i has features $x_{ij}, j = 1, \ldots, p$.
- One choice of dissimilarity function is the Euclidean distance

$$D(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

- Resulting clusters invariant to rotation and translation of features but not to scaling.
- If the features have different scales, standardize the data.

Standardization

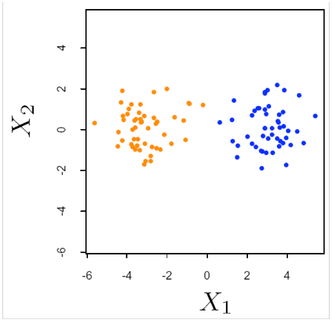




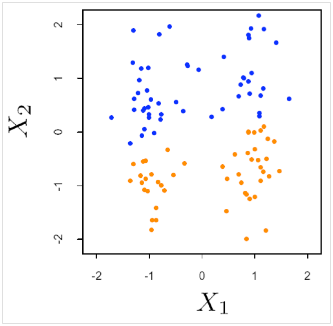
Without standardization

With standardization

Standardization not always helpful



Without standardization



With standardization

Outline

Outline

K-means: Idea

- K clusters each summarized by a prototype μ_k.
- Assignment of data x_i to a cluster represented by responsibilities $r_{ik} \in \{0, 1\}$ with $\sum_{k=1}^{K} r_{ik} = 1$.
- An example with 4 data points and 3 clusters.

$$(r_{ik}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Loss function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_2^2$.

K-means: minimizing the loss function

- How do we minimize J w.r.t (r_{ik}, μ_k)?
- Chicken and egg problem
  - If prototypes known, can assign responsibilities.
  - If responsibilities known, can compute prototypes.

K-means: minimizing the loss function

- How do we minimize J w.r.t (r_{ik}, μ_k)?
- Chicken and egg problem
    - If prototypes known, can assign responsibilities.
    - If responsibilities known, can compute prototypes.
- We use an iterative procedure.

K-means: minimizing the loss function

- E-step: Fix μ_k, minimize J w.r.t. r_{ik}.
  - Assign each data point to its nearest prototype.
- M-step: Fix r_{ik}, minimize J w.r.t. μ_k.
  - Set each prototype to the mean of the points in that cluster, i.e., $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$.
- This procedure is guaranteed to converge.

K-means: minimizing the loss function

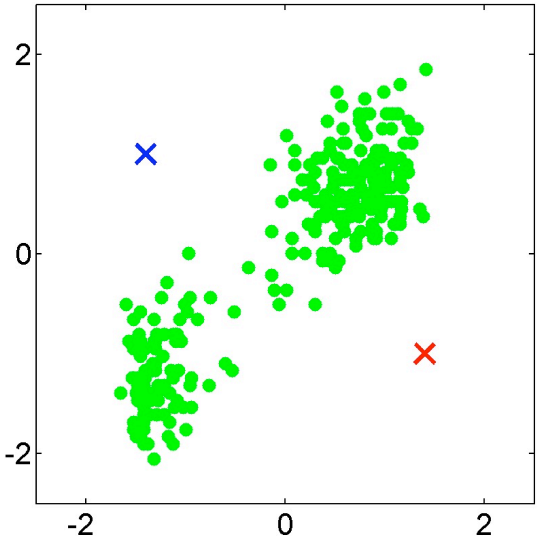
- E-step: Fix μ_k, minimize J w.r.t. r_{ik}.
  - Assign each data point to its nearest prototype.
- M-step: Fix r_{ik}, minimize J w.r.t. μ_k.
  - Set each prototype to the mean of the points in that cluster, i.e., $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$.
- This procedure is guaranteed to converge.
- Converges to a local minimum.

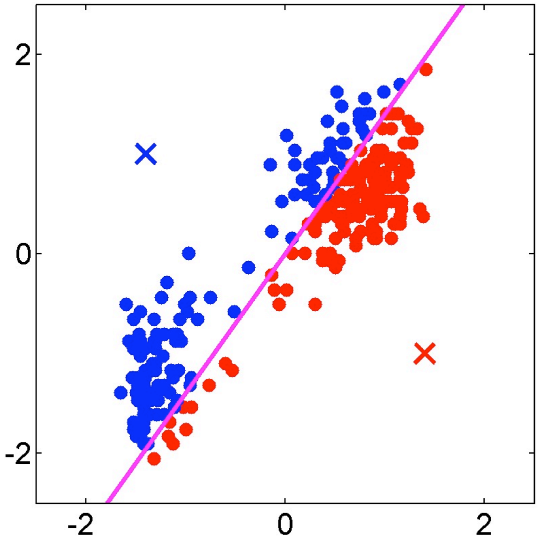
K-means: minimizing the loss function

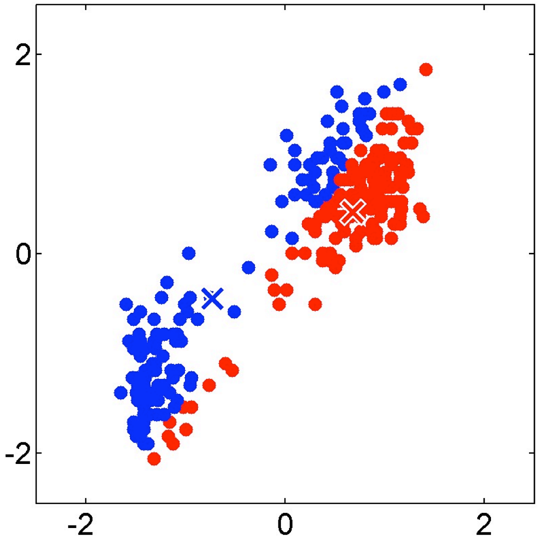
- E-step: Fix μ_k, minimize J w.r.t. r_{ik}.
  - Assign each data point to its nearest prototype.
- M-step: Fix r_{ik}, minimize J w.r.t. μ_k.
  - Set each prototype to the mean of the points in that cluster, i.e., $\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$.
- This procedure is guaranteed to converge.
- Converges to a local minimum.
  - Use different initializations and pick the best solution.
  - May still be insufficient for large search spaces.
  - Other ways include a split-merge approach.

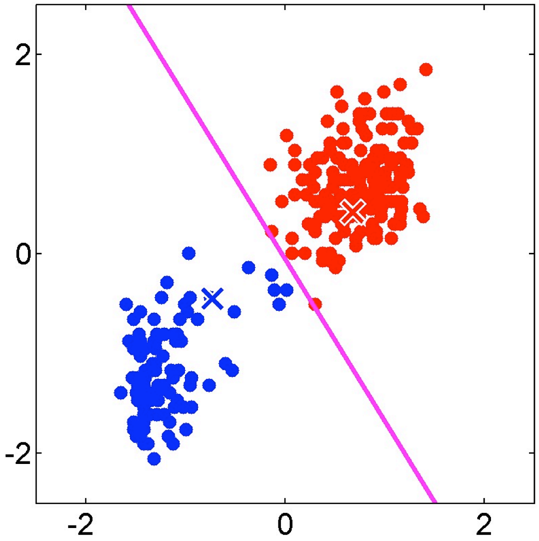
How do we initialize K-means?

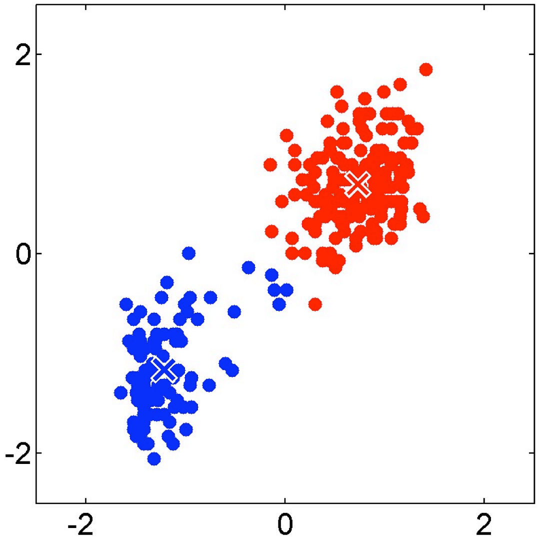
- Some heuristics
  - Randomly pick K data points as prototypes.
  - Pick prototype $i+1$ to be farthest from prototypes $\{1,\ldots,i\}$.

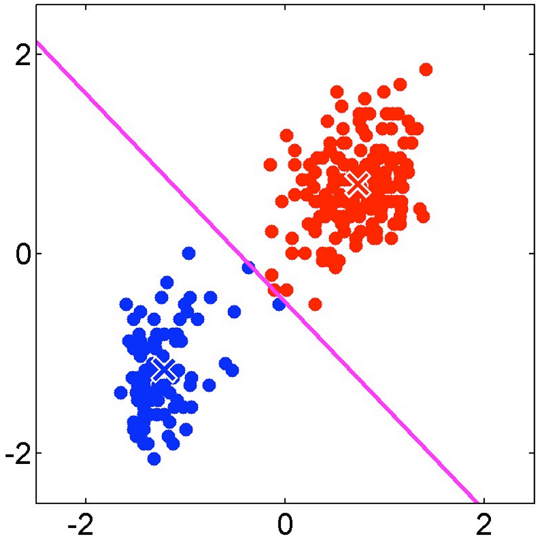

2
0
-2
-2
0
2


2
0
-2
-2
0
2

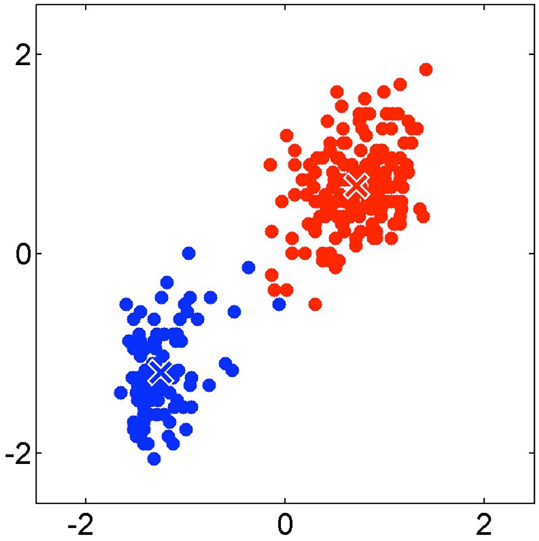

2
0
-2
-2
0
2

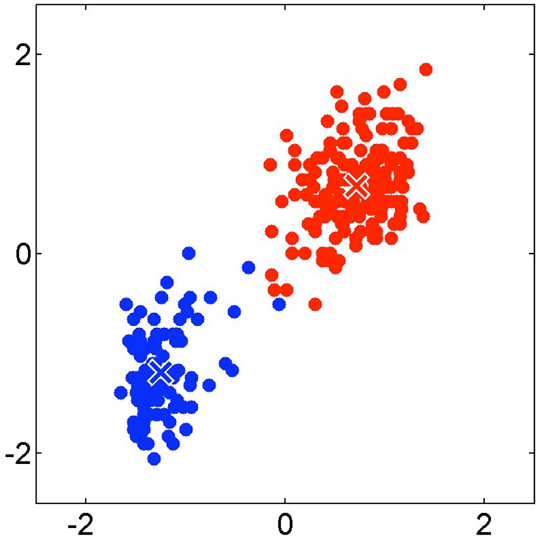

2
0
-2
-2
0
2

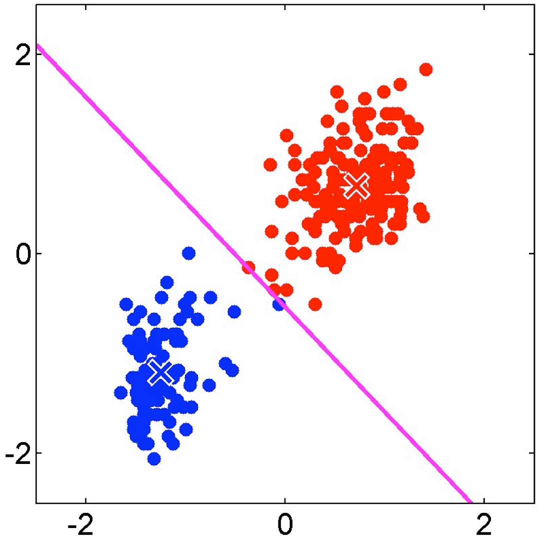

2
0
-2
-2
0
2



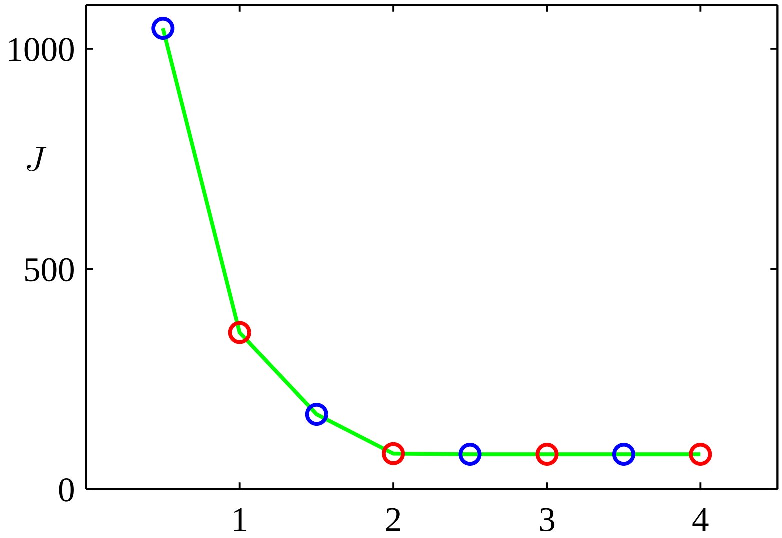
2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

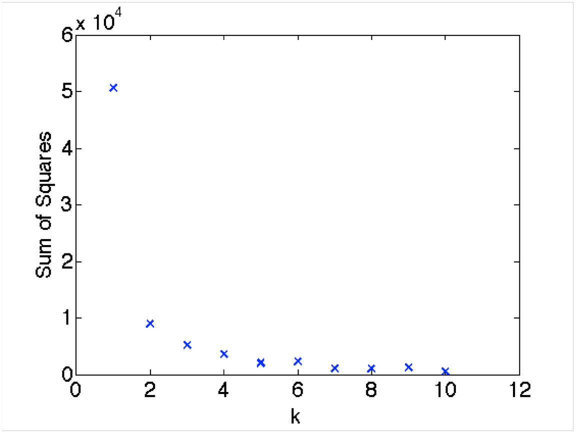

2
0
-2
-2
0
2

Loss function J after each iteration



How to choose K?

- Like choosing K in kNN.
- The loss function J generally decreases with K.



How to choose K?

- Gap statistic
- Cross-validation: Partition data into two sets. Estimate prototypes on one and use these to compute the loss function on the other.
- Stability of clusters: Measure the change in the clusters obtained by resampling or splitting the data.
- Non-parametric approach: Place a prior on K. More details in the Bayesian non-parametric lecture.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
    - Solution: GMM

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
  - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
    - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
    - Solution: GMM

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
    - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
    - Solution: GMM
- Clusters change arbitrarily for different K.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
    - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
    - Solution: GMM
- Clusters change arbitrarily for different K.
    - Solution: Hierarchical clustering

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
  - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
  - Solution: GMM
- Clusters change arbitrarily for different K.
  - Solution: Hierarchical clustering
- Sensitive to outliers.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
    - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
    - Solution: GMM
- Clusters change arbitrarily for different K.
    - Solution: Hierarchical clustering
- Sensitive to outliers.
    - Solution: Use a robust loss function.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
  - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
  - Solution: GMM
- Clusters change arbitrarily for different K.
  - Solution: Hierarchical clustering
- Sensitive to outliers.
  - Solution: Use a robust loss function.
- Works poorly on non-convex clusters.

Limitations of K-means

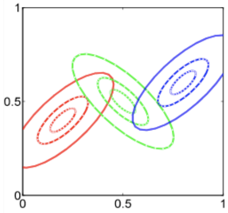
- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
  - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
  - Solution: GMM
- Clusters change arbitrarily for different K.
  - Solution: Hierarchical clustering
- Sensitive to outliers.
  - Solution: Use a robust loss function.
- Works poorly on non-convex clusters.
  - Solution: Spectral clustering.

Outline

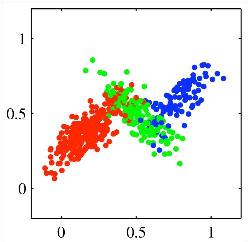
Gaussian Mixture Model

- Probabilistic story: Each cluster is associated with a Gaussian distribution. To generate data, randomly choose a cluster k with probability π_k and sample from its distribution.
- Likelihood $\Pr(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where

$$\sum_{k=1}^{K} \pi_k = 1, 0 \leq \pi_k \leq 1.$$



$\rightarrow$



Gaussian Mixture Model

- Loss function is the negative log likelihood

$$-\log \Pr(x|\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \right\}$$

- Why is this function difficult to optimize?

Gaussian Mixture Model

- Loss function is the negative log likelihood

$$-\log \Pr(x|\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \right\}$$

- Why is this function difficult to optimize?
  - Notice that the sum over the components appears inside the log, thus coupling all the parameters.

Gaussian Mixture Model

- Each x_i is associated with a latent variable $z_i = (z_{i1}, \dots, z_{iK})$.
- Given the complete data $(x, z) = (x_i, z_i), i = 1, \dots, n$
  - We can estimate the parameters by maximizing the complete data log likelihood.

$$\log \Pr(x, z|\pi, \mu, \Sigma) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \left\{ \log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k) \right\}$$

  - Notice that the π_k and (μ_k, Σ_k) decouple. Trivial closed-form solution exists.

Gaussian Mixture Model

- Each x_i is associated with a latent variable $z_i = (z_{i1}, \ldots, z_{iK})$.
- Given the complete data $(x, z) = (x_i, z_i), i = 1, \ldots, n$
  - We can estimate the parameters by maximizing the complete data log likelihood.

$$\log \Pr(x, z|\pi, \mu, \Sigma) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \left\{ \log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k) \right\}$$

  - Notice that the π_k and (μ_k, Σ_k) decouple. Trivial closed-form solution exists.
- Need a procedure that would optimize the log likelihood by working with the (easier) complete log likelihood.
  - “Fill-in” the latent variables using current estimate of the parameters.
  - Adjust parameters based on the filled-in variables.

The Expectation-Maximization (EM) algorithm

- E-step: Given parameters, compute

$$r_{ik} \triangleq E(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}$$

- M-step: Maximize the expected complete log likelihood

$$E\left[\log \Pr(x, z|\pi, \mu, \Sigma)\right] = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \left\{\log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k)\right\}$$

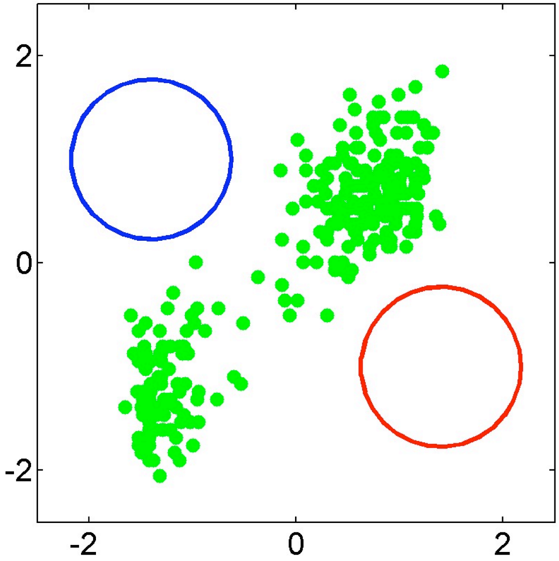
  To update the parameters

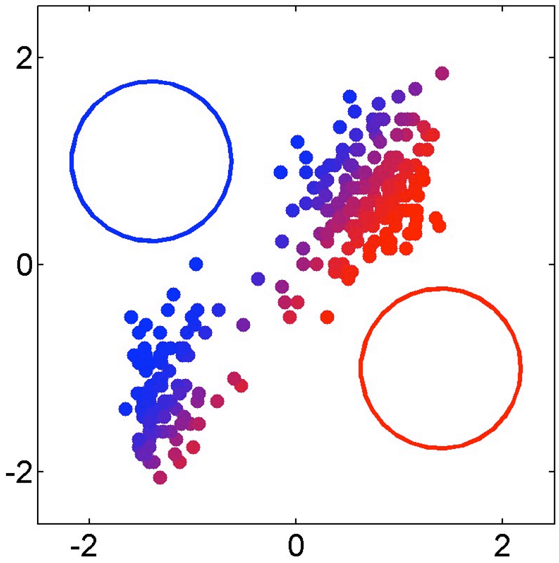
$$\pi_k = \frac{\sum_i r_{ik}}{n}, \mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}, \Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}$$

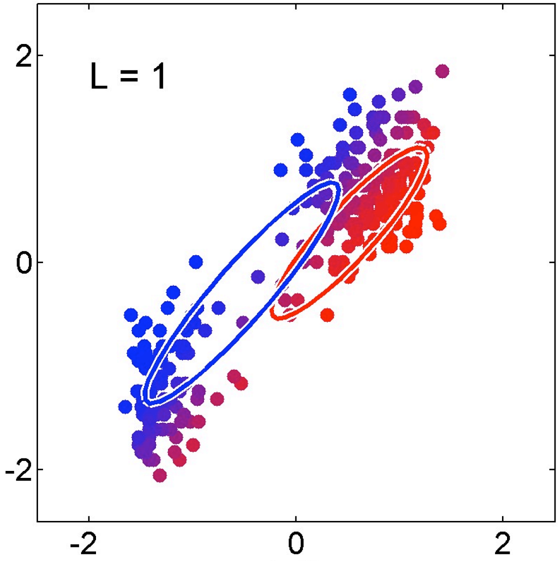
- Iterate till likelihood converges.
- Converges to local optimum of the log likelihood.

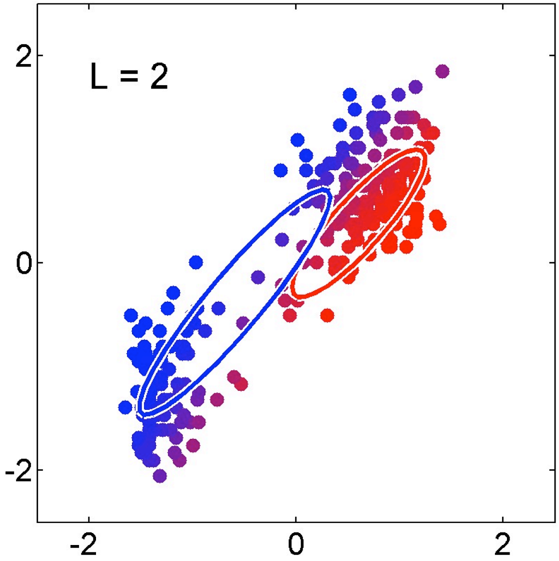
GMM: Relation to K-means

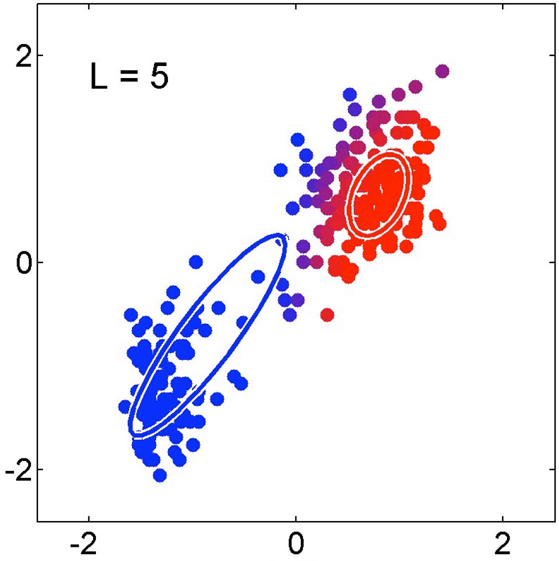
- E-step in GMM a soft version of K-means. $r_{ik} \in [0, 1]$ instead of $\{0, 1\}$.
- M-step in GMM estimates the probabilities and the covariance matrix of each cluster in addition to the means.
- All π_k are equal. $\Sigma_k = \delta^2 I$. As $\delta^2 \to 0$, $r_{ik} \to \{0, 1\}$, and the two methods coincide.

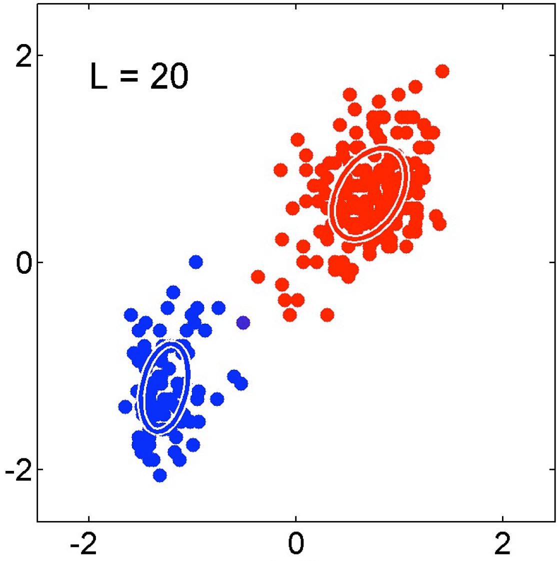

2
0
-2
-2
0
2


2
0
-2
-2
0
2


L = 1
2
0
-2
-2
0
2


L = 2
2
0
-2
-2
0
2


L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

K-means vs GMM

- Loss function: minimize sum of squared distance.
- Hard assignment of points to clusters.
- Assumes spherical clusters with equal probability of a cluster.

- Minimize negative log likelihood.
- Soft assignment of points to clusters.
- Can be used for non-spherical clusters with different probabilities.

Outline

K-medoids

- Squared Euclidean distance loss function of K-means not robust.
- Only the dissimilarity matrix may be given.
- Attributes not quantitative.

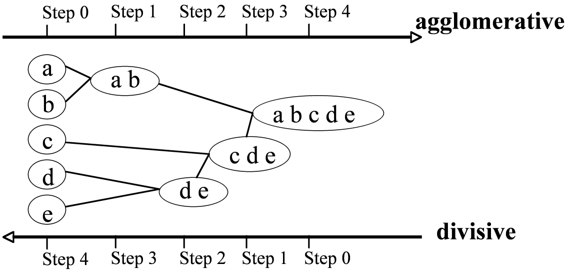
K-medoids

- Use L1 loss function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_1$ instead of squared Euclidean distance.
  - Recall connection between linear and L1 regression.
- Use an iterative procedure as before.
  - Prototype is the median of the points assigned to a cluster.

Outline

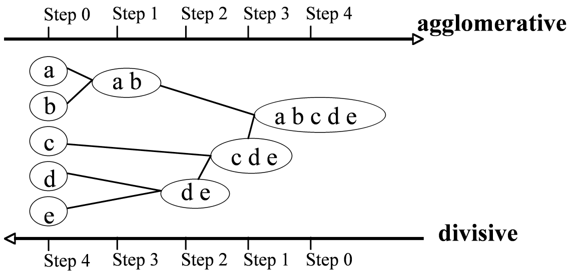
Hierarchical Clustering

- Organize the clusters in a hierarchical way.
- Produces a rooted binary tree (dendrogram).



Hierarchical Clustering

- Bottom-up (agglomerative): Recursively merge two groups with the smallest between-cluster similarity.
- Top-down (divisive): Recursively split a least-coherent (e.g. largest diameter) cluster.
- Users can then choose a cut through the hierarchy to represent the most natural division into clusters (e.g. where intergroup similarity exceeds some threshold).



Hierarchical Clustering

- Dissimilarity for two disjoint groups G and H, $d(G, H)$ is computed from pairwise dissimilarities $D(i, j), i\ inG, j \in H$.
  - Single linkage: tends to yield extended clusters.

$$D_{SL}(G, H) = min_{i \in G, j \in H} D(i, j)$$

  - Complete linkage: tends to yield round clusters.

$$D_{CL}(G, H) = max_{i \in G, j \in H} D(i, j)$$

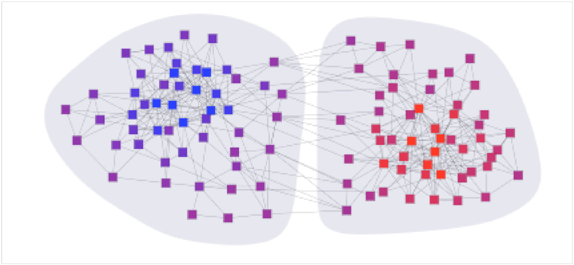
  - Group average: tradeoff between the two. Not invariant under monotone increasing transform.

$$D_{GA}(G, H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} D(i, j)$$

Outline

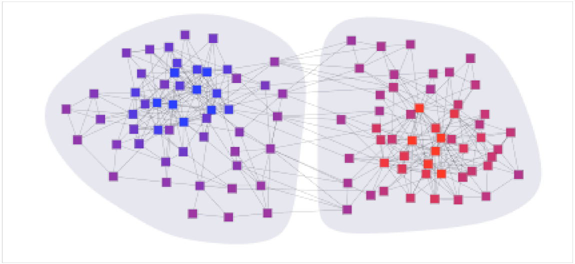
Spectral Clustering

- Represent datapoints as vertices V of a graph G.
- Each pair of vertices is connected by an edge.
- Edges have weights W. Large weights mean that adjacent vertices are similar.
- The graph construction depends on the application.



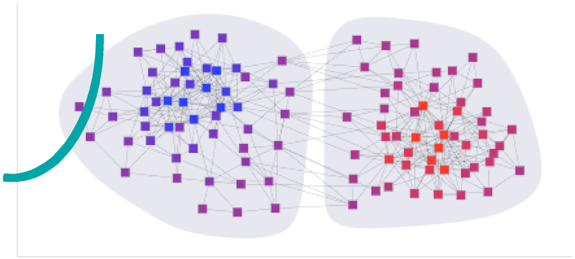
Graph partitioning

- Clustering partitions the vertices of the graph. A good clustering places dissimilar vertices in different partitions.
- The loss function for a partition of $(A, V - A)$ is given by the cut $cut(A, V - A) = \sum_{i \in A, j \in V - A} W_{ij}$.
- Find a partition that minimizes the cut (Mincut criterion).
- Does this criterion produce good clusters?



Graph partitioning

- Clustering partitions the vertices of the graph. A good clustering places dissimilar vertices in different partitions.
- The loss function for a partition of $(A, V-A)$ is given by the cut $cut(A, V-A) = \sum_{i \in A, j \in V-A} W_{ij}$.
- Find a partition that minimizes the cut (Mincut criterion).
- Does this criterion produce good clusters?



Graph partitioning

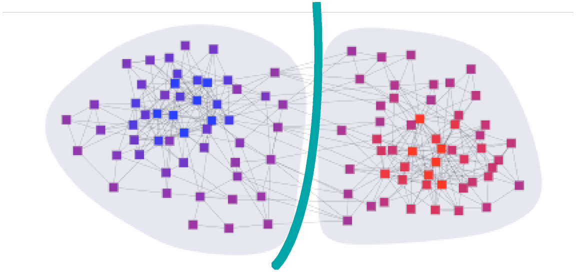
- A good partition should separate dissimilar vertices and should produce balanced clusters.
- A loss function that favors such clusters is Normalized cut

$$Ncut(A, B) = \frac{cut(A, B)}{\sum_{i \in A, j} W_{ij}} + \frac{cut(B, A)}{\sum_{i \in B, j} W_{ij}}$$

Graph partitioning

- A good partition should separate dissimilar vertices and should produce balanced clusters.
- A loss function that favors such clusters is Normalized cut

$$Ncut(A, B) = \frac{cut(A, B)}{\sum_{i \in A, j} W_{ij}} + \frac{cut(B, A)}{\sum_{i \in B, j} W_{ij}}$$

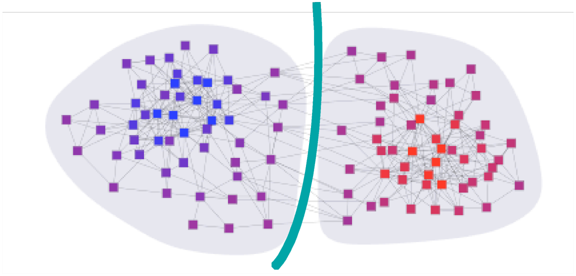


Graph partitioning

- A good partition should separate dissimilar vertices and should produce balanced clusters.
- A loss function that favors such clusters is Normalized cut

$$Ncut(A, B) = \frac{cut(A, B)}{\sum_{i \in A, j} W_{ij}} + \frac{cut(B, A)}{\sum_{i \in B, j} W_{ij}}$$

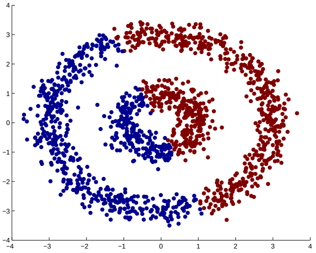
- Minimizing normalized cut is NP-hard.



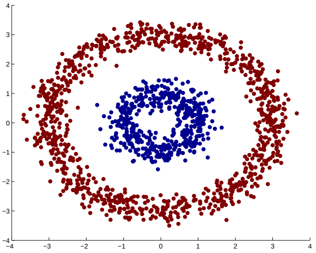
Spectral Clustering

- One way of approximately optimizing normalized cuts leads to spectral clustering.
- Overview
  - Build a weighted graph $G = (V, E, W)$.
  - Construct a matrix $L = f(W)$ (different variants of spectral clustering result from different functions f.
  - Compute the eigenvectors of the k smallest eigenvalues of L. These provide a new representation of the original data points.
  - Cluster the points in this new representation (e.g. using K-means).
- Note that there is no guarantee on the quality of the solution.

Spectral Clustering

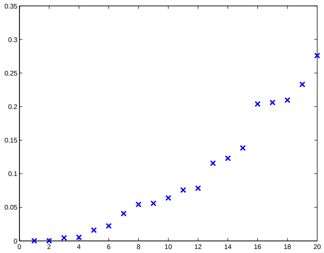


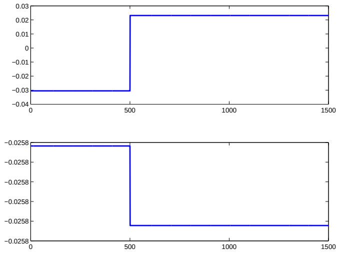
K-means, K=2



Spectral clustering

Spectral Clustering





Eigenvalues of L

Eigenvectors of L

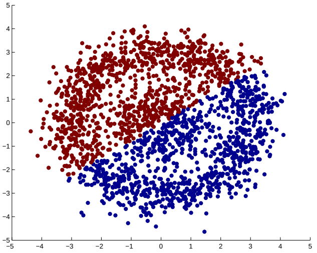
Spectral Clustering

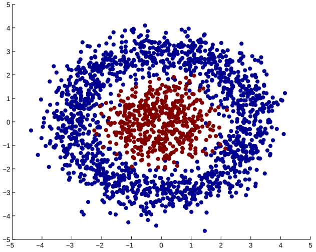
- Set $D = diag(W\mathbf{1})$ and compute the Laplacian $L = I - D^{-1}W$.
- Intuition:
  - Ideal case: If the graph has K components (clusters are separable), the k smallest eigenvalues are 0.
  - The indicator vectors on each of the components span the eigenspace of 0. Trivial to assign datapoints to clusters.

$$\begin{aligned} L &= \begin{pmatrix} L_1 & & \cdots \\ & L_2 & \cdots \\ & & \cdots \\ \cdots & & L_k \end{pmatrix} \\ L\mathbf{1}_i &= 0, i = 1, \ldots, k \end{aligned}$$

  - Can be extended to a case where the clusters are not completely distinct. The eigenvectors of L will still be close to the indicator vectors provided the eigengap $|\lambda_k - \lambda_{k+1}|$ is large relative to the perturbation.

Spectral Clustering

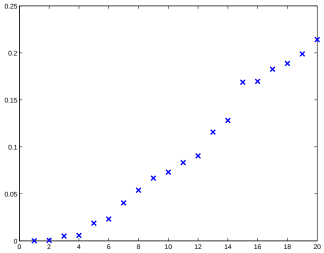


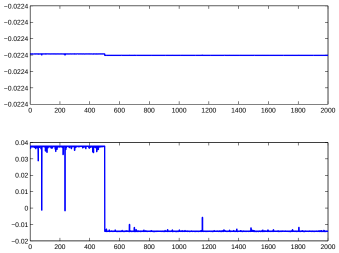


K-means, K=2

Spectral clustering

Spectral Clustering

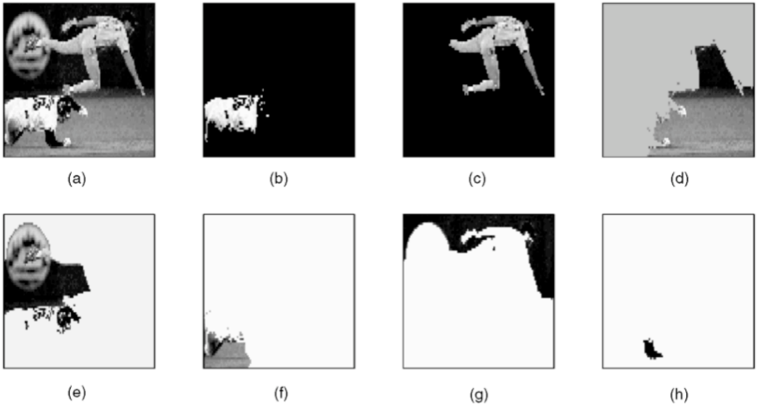




Eigenvalues of L

Eigenvectors of L

Image segmentation using Spectral Clustering



Shi and Malik, 2000

Learning Dissimilarity

- Suppose a user indicates that certain objects are “similar”: $(x_i, x_j) \in \mathcal{S}$ if x_i and x_j are similar
- Consider learning a dissimilarity that respects this subjectivity

$$D(x_i, x_j) = \|x_i - x_j\|_A = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

- Learning such a dissimilarity is equivalent to replacing x by $\sqrt{A}x$ and then applying standard Euclidean distance.

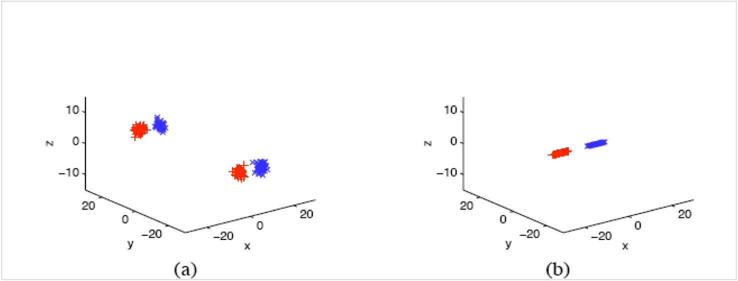
Learning Dissimilarity

- A simple way to define a criterion for the dissimilarity

$$\begin{aligned} & min_A \sum_{(x_i,x_j)\in\mathcal{S}} \|x_i - x_j\|_A{}^2 \\ s.t. \quad & \sum_{(x_i,x_j)\in\mathcal{D}} \|x_i - x_j\|_A \geq 1 \\ & A \succeq 0 \end{aligned}$$

- A convex optimization problem. Can be solved by gradient descent and iterative projection.
- For more details, see [Xing, Ng, Jordan, Russell, 2003].

Learning dissimilarity



Original 2-class data

Projected 2-class data

References

- Hastie, Tibshirani and Friedman, The Elements of Statistical Learning, Chapter 14
- Bishop, Pattern Recognition and Machine Learning, Chapter 9