

GRAPH LAPLACEANS AND THEIR APPLICATIONS

Graph Laplaceans - Definition

➤ “Laplace-type” matrices associated with general undirected graphs – useful in many applications

- Given a graph $G = (V, E)$ define
 - A matrix W of weights w_{ij} for each edge
 - Assume $w_{ij} \geq 0$, $w_{ii} = 0$, and $w_{ij} = w_{ji} \forall (i, j)$
 - The diagonal matrix $D = \text{diag}(d_i)$ with $d_i = \sum_{j \neq i} w_{ij}$

➤ Corresponding **graph Laplacean** of G is: $L = D - W$

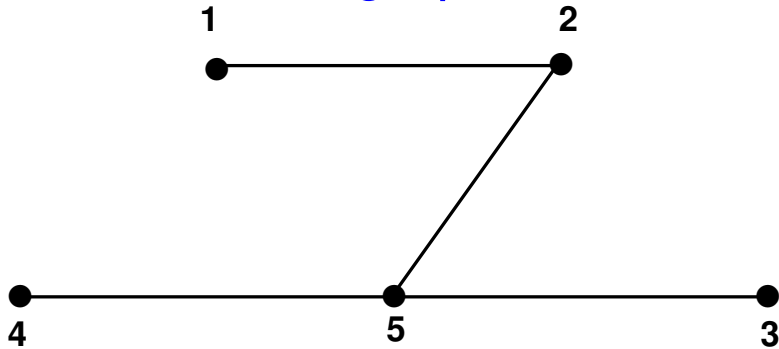
➤ Gershgorin's theorem $\rightarrow L$ is positive semidefinite.

➤ Simplest case:

$$w_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \& i \neq j \\ 0 & \text{else} \end{cases} \quad D = \text{diag} \left[d_i = \sum_{j \neq i} w_{ij} \right]$$

Example:

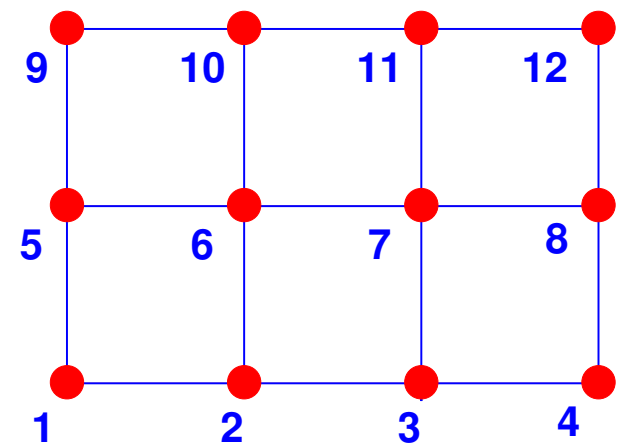
Consider the graph



$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

 33

Define the graph Laplacean for the graph associated with the simple mesh shown next. [use the simple weights of 0 or 1]. What is the difference with the discretization of the Laplace operator for case when mesh is the same as this graph?



Proposition:

- (i) L is symmetric semi-positive definite.
- (ii) L is singular with $\mathbf{1}$ as a null vector.
- (iii) If G is connected, then $\text{Null}(L) = \text{span}\{\mathbf{1}\}$
- (iv) If G has $k > 1$ connected components G_1, G_2, \dots, G_k , then the nullity of L is k and $\text{Null}(L)$ is spanned by the vectors $z^{(j)}$, $j = 1, \dots, k$ defined by:

$$(z^{(j)})_i = \begin{cases} 1 & \text{if } i \in G_j \\ 0 & \text{if not.} \end{cases}$$

Proof: (i) and (ii) seen earlier and are trivial. (iii) Clearly $u = \mathbb{1}$ is a null vector for L . The vector $D^{-1/2}u$ is an eigenvector for the matrix $D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$ associated with the smallest eigenvalue. It is also an eigenvector for $D^{-1/2}WD^{-1/2}$ associated with the largest eigenvalue. By the Perron Frobenius theorem this is a simple eigenvalue... (iv) Can be proved from the fact that L can be written as a direct sum of the Laplacian matrices for G_1, \dots, G_k . ■

A few properties of graph Laplaceans

Define: oriented incidence matrix H : (1) First orient the edges $i \sim j$ into $i \rightarrow j$ or $j \rightarrow i$. (2) Rows of H indexed by vertices of G . Columns indexed by edges. (3) For each (i, j) in E , define the corresponding column in H as $\sqrt{w(i, j)}(e_i - e_j)$.

Example: In previous example (4 p. back) orient $i \rightarrow j$ so that $j > i$ [lower triangular matrix representation]. Then matrix H is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 \end{bmatrix}$$

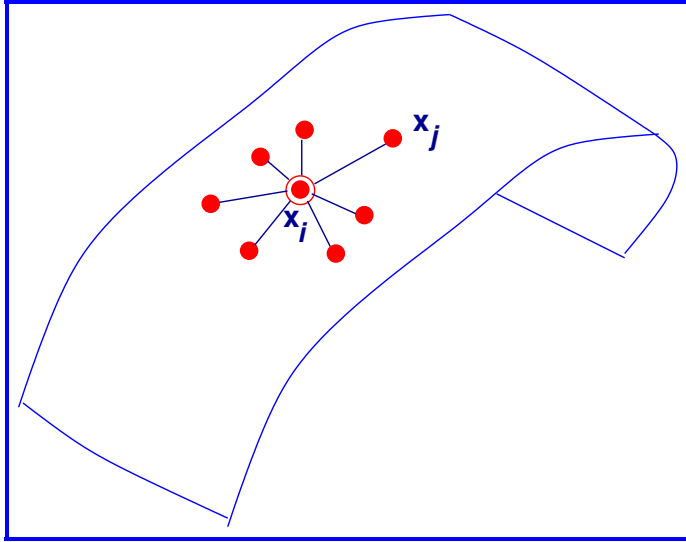
Property 1

$$L = HH^T$$

 34

Re-prove part (iv) of previous proposition by using this property.

A few properties of graph Laplaceans



Strong relation between $x^T L x$ and local distances between entries of x

► Let $L =$ any matrix s.t. $L = D - W$, with $D = \text{diag}(d_i)$ and

$$w_{ij} \geq 0, \quad d_i = \sum_{j \neq i} w_{ij}$$

Property 2: for any $x \in \mathbb{R}^n$:

$$x^T L x = \frac{1}{2} \sum_{i,j} w_{ij} |x_i - x_j|^2$$

Property 3: (generalization) for any $Y \in \mathbb{R}^{d \times n}$:

$$\text{Tr}[YLY^\top] = \frac{1}{2} \sum_{i,j} w_{ij} \|y_i - y_j\|^2$$

► Note: $y_j = j$ -th column of Y . Usually $d < n$. Each column can represent a data sample.

Property 4: For the particular $L = I - \frac{1}{n} \mathbf{1} \mathbf{1}^\top$

$$XLY^\top = \bar{X}\bar{X}^\top == n \times \text{Covariance matrix}$$

Property 5: L is singular and admits the null vector

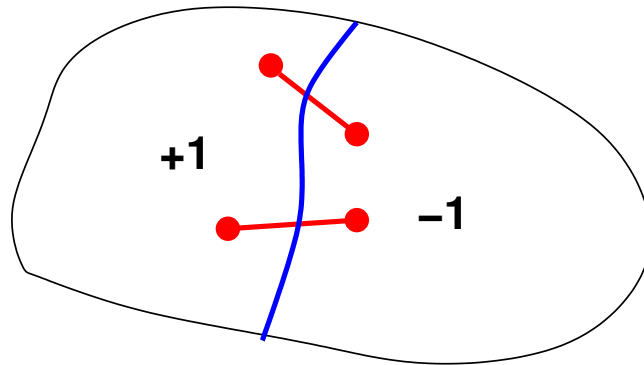
$\mathbf{1} = \text{ones}(n, 1)$

Property 6: (Graph partitioning) Consider situation when $w_{ij} \in \{0, 1\}$. If x is a vector of signs (± 1) then

$$x^\top Lx = 4 \times (\text{'number of edge cuts'})$$


edge-cut = pair (i, j) with $x_i \neq x_j$


➤ Consequence: Can be used to partition graphs



➤ Would like to minimize (Lx, x) subject to $x \in \{-1, 1\}^n$ and $e^T x = 0$
[balanced sets]

➤ Will solve a relaxed form of this problem

 35 What if we replace x by a vector of ones (representing one partition) and zeros (representing the other)?

 36 Let x be any vector and $y = x + \alpha \mathbf{1}$ and L a graph Laplacean. Compare (Lx, x) with (Ly, y) .

➤ Consider any symmetric (real) matrix A with eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ and eigenvectors u_1, \dots, u_n

➤ Recall that:
(Min reached for $x = u_1$)

$$\min_{x \in \mathbb{R}^n} \frac{(Ax, x)}{(x, x)} = \lambda_1$$

➤ In addition:
(Min reached for $x = u_2$)

$$\min_{x \perp u_1} \frac{(Ax, x)}{(x, x)} = \lambda_2$$

➤ For a graph Laplacean $u_1 = \mathbf{1}$ = vector of all ones and

➤ ...vector u_2 is called the Fiedler vector. It solves a **relaxed** form of the problem -

$$\min_{x \in \{-1, 1\}^n; \mathbb{1}^T x = 0} \frac{(Lx, x)}{(x, x)}$$

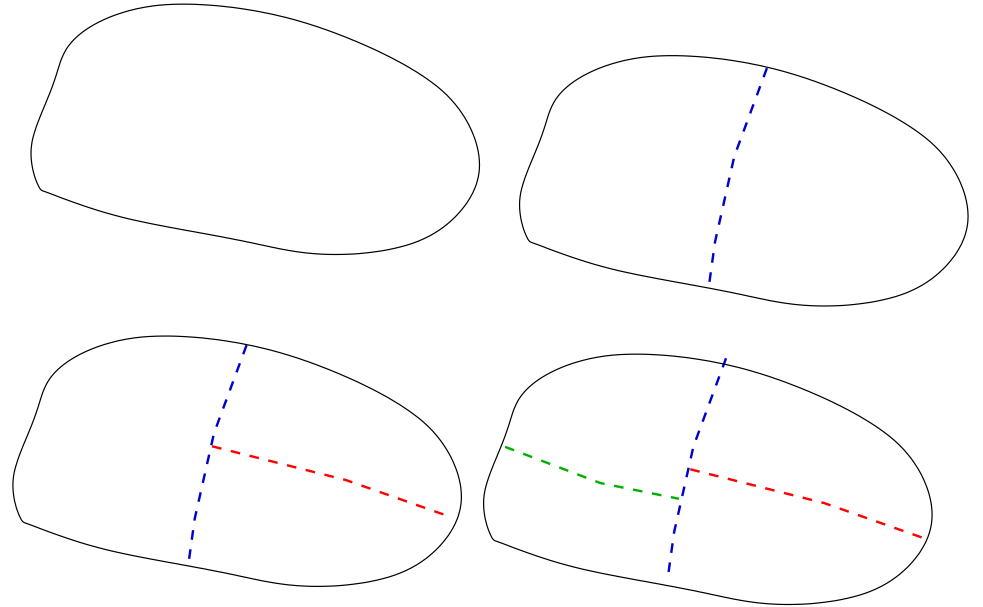
→

$$\min_{x \in \mathbb{R}^n; \mathbb{1}^T x = 0} \frac{(Lx, x)}{(x, x)}$$

➤ Define $v = u_2$ then $lab = \text{sign}(v - \text{med}(v))$

Recursive Spectral Bisection

- 1 Form graph Laplacean
- 2 Partition graph in 2 based on Fiedler vector
- 3 Partition largest subgraph in two recursively ...
- 4 ... Until the desired number of partitions is reached



Three approaches to graph partitioning:

1. Spectral methods - Just seen + add Recursive Spectral Bisection.
2. Geometric techniques. Coordinates are required. [Houstis & Rice et al., Miller, Vavasis, Teng et al.]
3. Graph Theory techniques – multilevel,... [use graph, but no coordinates]
 - Currently best known technique is Metis (multi-level algorithm)
 - Simplest idea: Recursive Graph Bisection; Nested dissection (George & Liu, 1980; Liu 1992)
 - Advantages: simplicity – no coordinates required

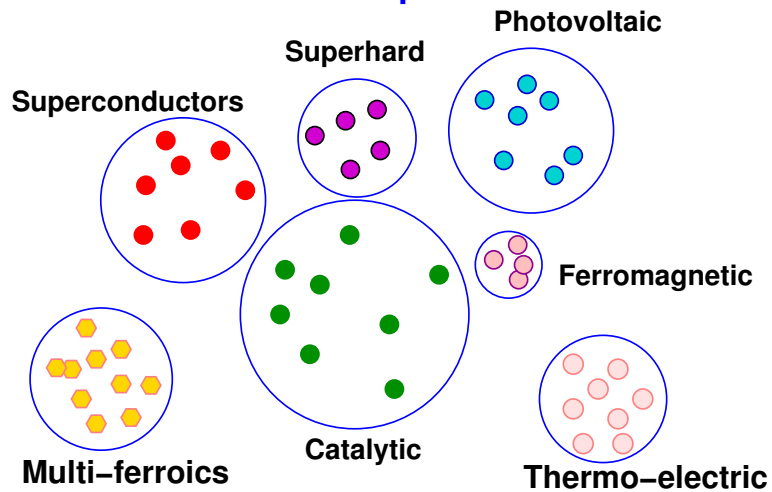
 37 Run *testBis_simple* and *testMeshPart* (in /gpartition)

APPLICATIONS OF GRAPH LAPLACEANS: CLUSTERING

Clustering

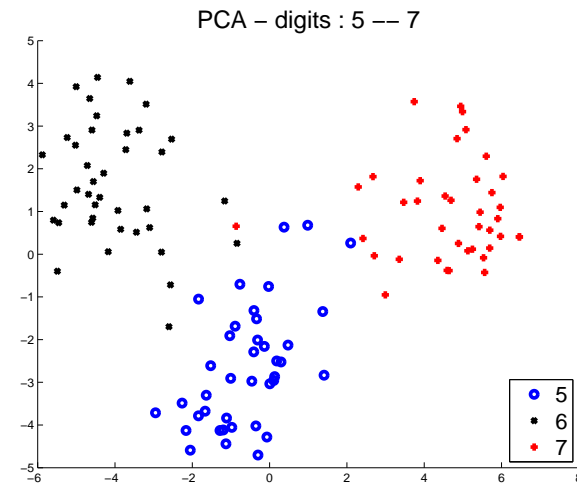
➤ Problem: we are given n data items: x_1, x_2, \dots, x_n . Would like to ‘cluster’ them, i.e., group them so that each group or cluster contains items that are similar in some sense.

➤ Example: materials



➤ Each group is a ‘cluster’ or a ‘class’

➤ Example: Digits

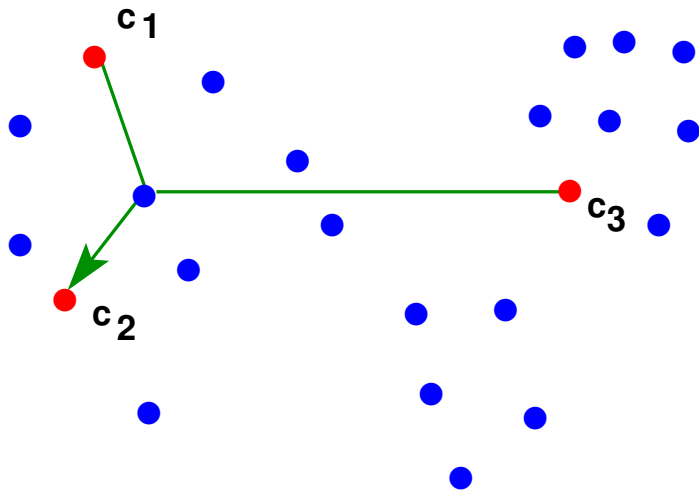


➤ ‘Unsupervised learning’

A basic method: *K*-means

➤ A basic algorithm that uses Euclidean distance

- 1 Select p initial centers: c_1, c_2, \dots, c_p for classes $1, 2, \dots, p$
- 2 For each x_i do: determine *class* of x_i as $\operatorname{argmin}_k \|x_i - c_k\|$
- 3 Redefine each c_k to be the centroid of class k
- 4 Repeat until convergence



- Simple algorithm
- Works well (gives good results) but can be slow
- Performance depends on initialization

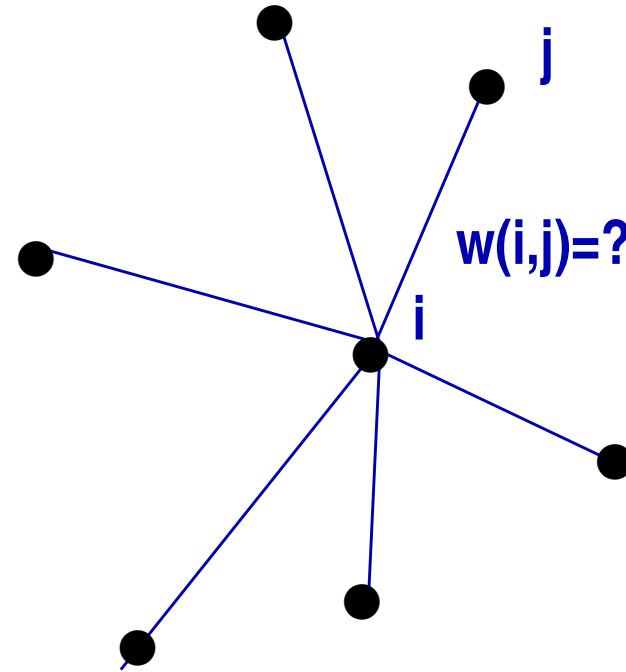
Methods based on similarity graphs

- Class of Methods that perform clustering by exploiting a graph that describes the similarities between any two items in the data.
- Need to:
 1. decide what nodes are in the neighborhood of a given node
 2. quantify their similarities - by assigning a weight to any pair of nodes.

Example: For text data: Can decide that any columns i and j with a cosine greater than 0.95 are 'similar' and assign that cosine value to w_{ij}

First task: build a 'similarity' graph

➤ Goal: to build a **similarity** graph, i.e., a graph that captures similarity between any two items



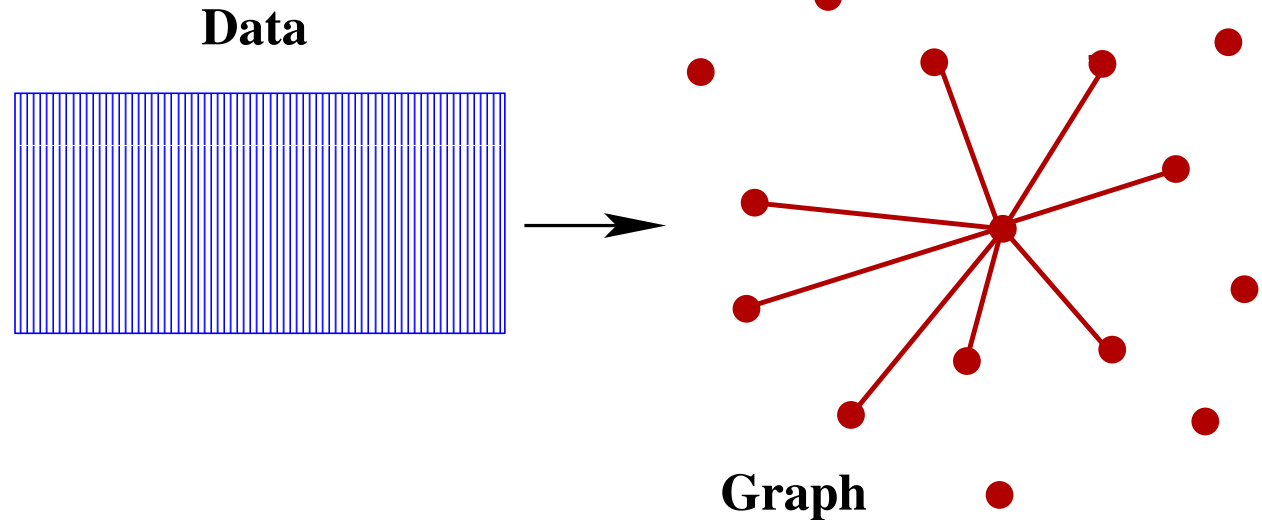
➤ Two methods: K-nearest Neighbor graphs or use Gaussian ('heat') kernel

K-nearest neighbor graphs

- Given: a set of n data points $X = \{x_1, \dots, x_n\} \rightarrow$ vertices
- Given: a **proximity** measure between two data points x_i and x_j – as measured by a quantity $dist(x_i, x_j)$
- Want: For each point x_i a list of the ‘nearest neighbors’ of x_i (edges between x_i and these nodes).
- Note: graph will usually be **directed** \rightarrow need to symmetrize

Nearest neighbor graphs

- For each node, get a few of the nearest neighbors → Graph



- Problem: How to build a nearest-neighbor graph from given data
- We will revisit this later.

Two types of nearest neighbor graph often used:

ϵ -graph:

Edges consist of pairs (x_i, x_j) such that $\rho(x_i, x_j) \leq \epsilon$

k NN graph:

Nodes adjacent to x_i are those nodes x_ℓ with the k with smallest distances $\rho(x_i, x_\ell)$.

- ϵ -graph is undirected and is geometrically motivated. Issues: 1) may result in disconnected components 2) what ϵ ?
- k NN graphs are directed in general (can be trivially fixed).
- k NN graphs especially useful in practice.

Similarity graphs: Using 'heat-kernels'

Define weight between i and j as:

$$w_{ij} = f_{ij} \times \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma_X^2}} & \text{if } \|x_i - x_j\| < r \\ 0 & \text{if not} \end{cases}$$

- Note $\|x_i - x_j\|$ could be any measure of distance...
- f_{ij} = optional = some measure of similarity - other than distance
- Only nearby points kept.
- Sparsity depends on parameters

Edge cuts, ratio cuts, normalized cuts, ...

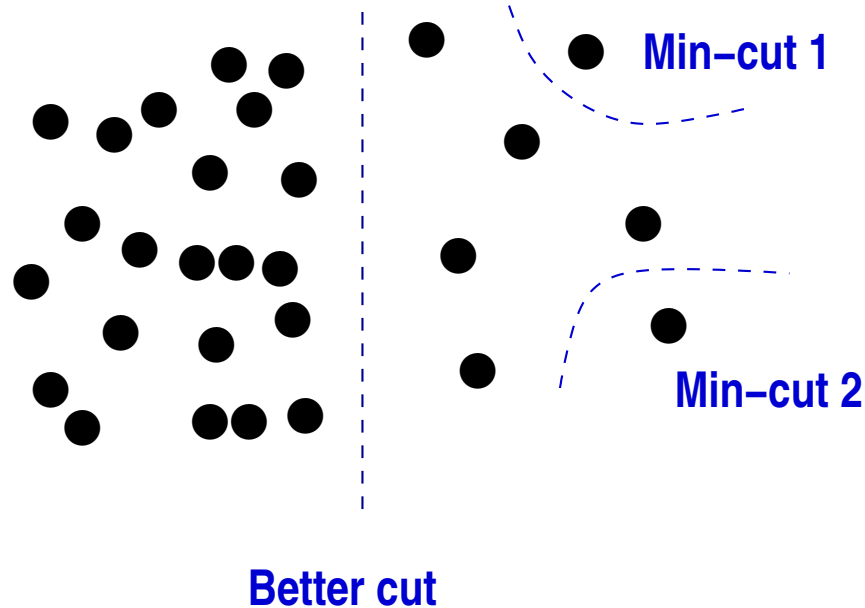
- Assume now that we have built a ‘similarity graph’
- Setting is identical with that of graph partitioning.
- Need a Graph Laplacean: $L = D - W$ with $w_{ii} = 0, w_{ij} \geq 0$ and $D = \text{diag}(W * \text{ones}(n, 1))$ [in matlab notation]
- Partition vertex set V in two sets A and B with

$$A \cup B = V, \quad A \cap B = \emptyset$$

- Define

$$\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

- First (naive) approach: use this measure to partition graph, i.e.,
... Find A and B that minimize $cut(A, B)$.
- Issue: Small sets, isolated nodes, big imbalances,



Ratio-cuts

- Standard Graph Partitioning approach: Find A, B by solving

Minimize $cut(A, B)$, subject to $|A| = |B|$

- Condition $|A| = |B|$ not too meaningful in some applications - too restrictive in others.
- Minimum Ratio Cut approach. Find A, B by solving:

Minimize $\frac{cut(A, B)}{|A| \cdot |B|}$

- Difficult to find solution (original paper [Wei-Cheng '91] proposes several heuristics)
- Approximate solution : spectral .

Theorem [Hagen-Kahng, 91] If λ_2 is the 2nd smallest eigenvalue of L , then a lower bound for the cost c of the optimal ratio cut partition, is:

$$c \geq \frac{\lambda_2}{n}.$$

➤ Idea is to use eigenvector associated with λ_2 to determine partition, e.g., based on sign of entries. Use the ratio-cut measure to actually determine where to split.

Normalized cuts [Shi-Malik,2000]

- Recall notation $w(X, Y) = \sum_{x \in X, y \in Y} w(x, y)$ - then define:

$$ncut(A, B) = \frac{cut(A, B)}{w(A, V)} + \frac{cut(A, B)}{w(B, V)}$$

- Goal is to avoid small sets A, B

 38 What is $w(A, V)$ in the case when $w_{ij} == 1$?

- Let x be an indicator vector:

$$x_i = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{if } i \in B \end{cases}$$

- Recall that: $x^T L x = \sum_{(i,j) \in E} w_{ij} |x_i - x_j|^2$ (each edge counted once)
- Using this in $ncut$ + calculations ...

- Need to solve:

$$\begin{aligned} & \min_{y_i \in \{0, -\beta\}} \frac{y^T L y}{y^T D y} \\ & \text{Subject to } y^T D \mathbf{1} = 0 \end{aligned}$$

- + Relax \rightarrow need to solve Generalized eigenvalue problem

$$L y = \lambda D y$$

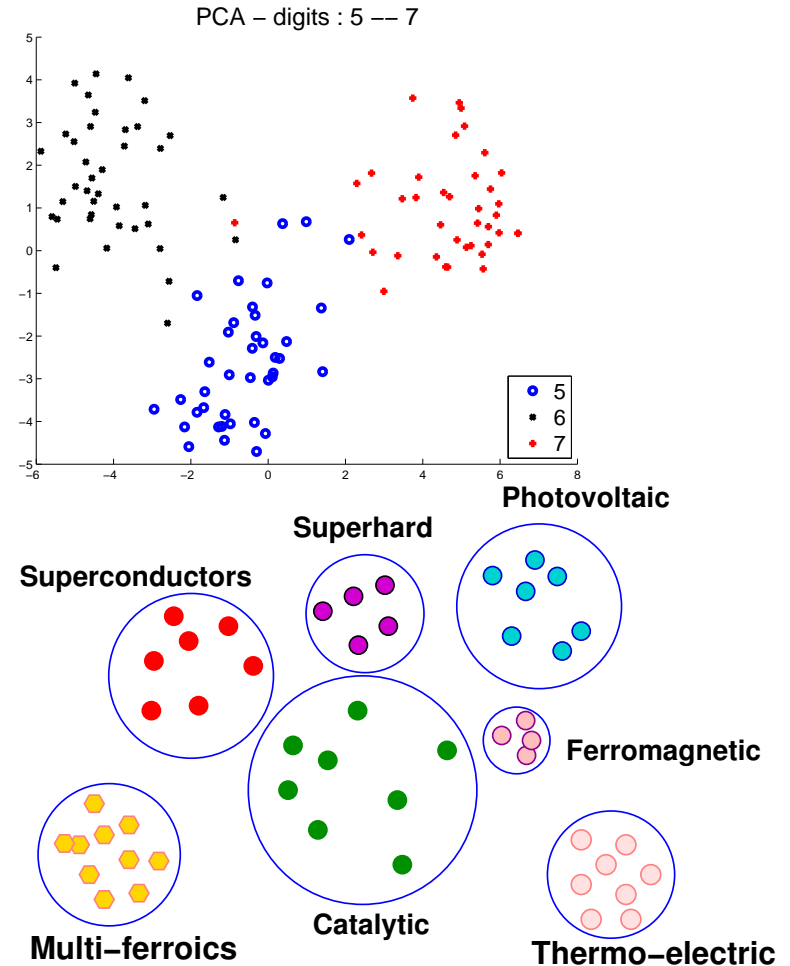
- $y_1 = \mathbf{1}$ is eigenvector associated with eigenvalue $\lambda_1 = 0$
- y_2 associated with second eigenvalue solves problem.
- Method quite popular for segmentation

DIMENSION REDUCTION - A.K.A. EMBEDDING

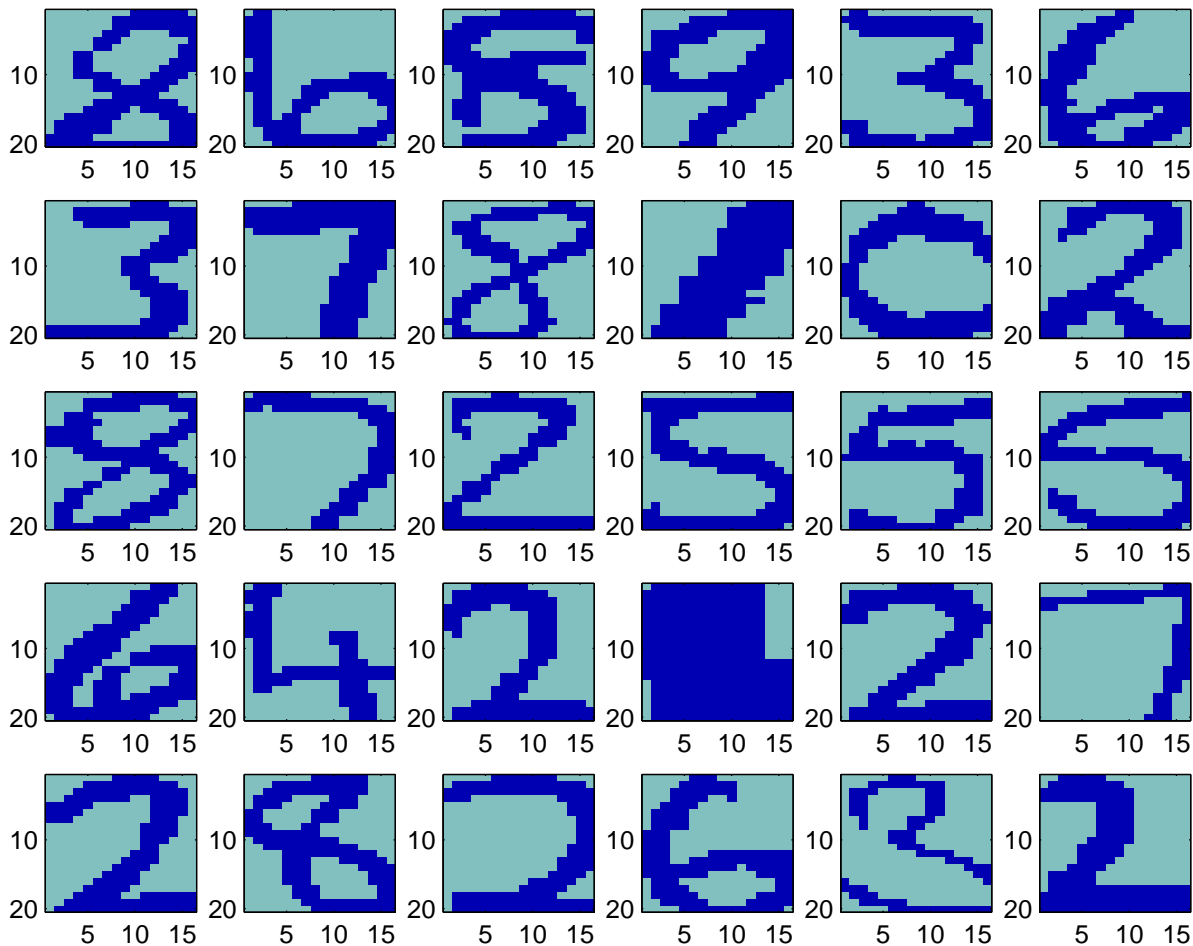
Recall: Unsupervised learning

“*Unsupervised learning*”: methods do not exploit labeled data

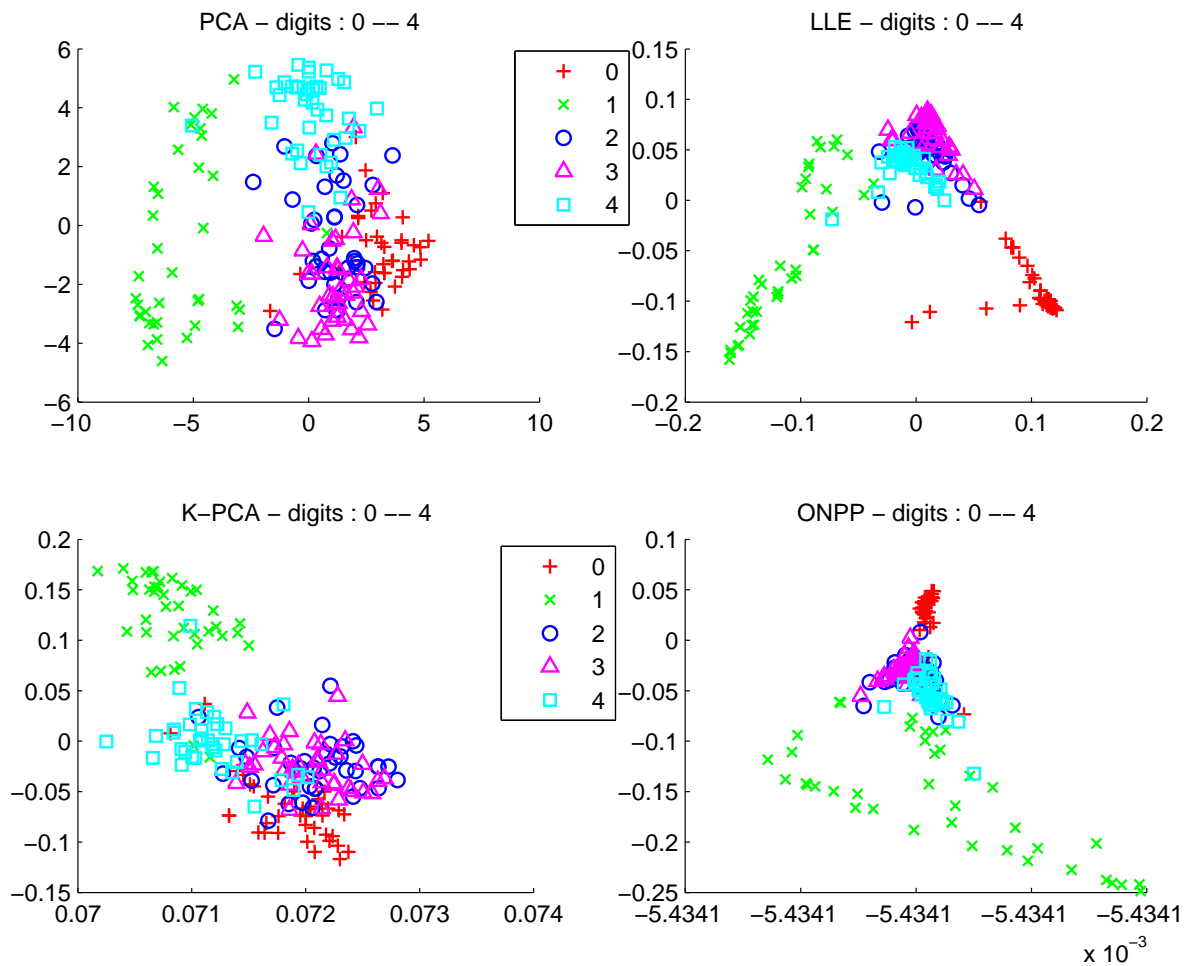
- Example of digits: perform a 2-D projection
- Images of same digit tend to cluster (more or less)
- Such 2-D representations are popular for visualization
- Can also try to find natural clusters in data, e.g., in materials
- Basic clustering technique: K-means



Example: Digit images (a random sample of 30)



2-D 'reductions':

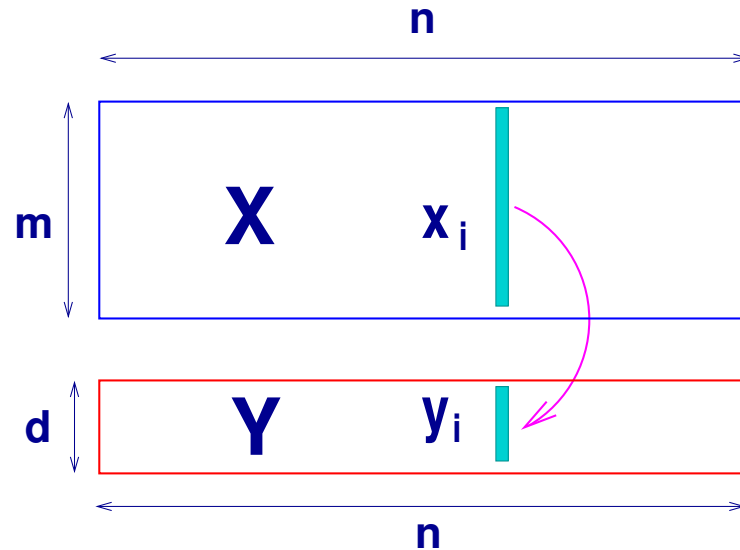
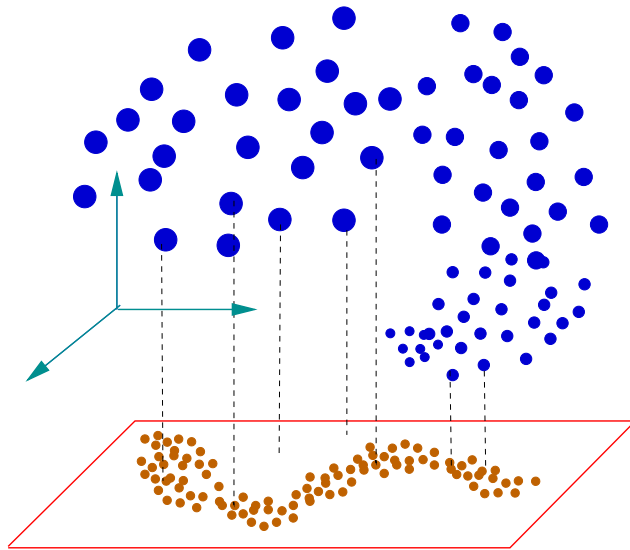


Major tool of Data Mining: Dimension reduction

➤ Given: $X = [x_1, \dots, x_n] \in \mathbb{R}^{m \times n}$, find a low-dimens. representation $Y = [y_1, \dots, y_n] \in \mathbb{R}^{d \times n}$ of X

➤ Achieved by a mapping

$$\Phi : x \in \mathbb{R}^m \longrightarrow y \in \mathbb{R}^d$$



- Φ may be linear : $y_j = W^\top x_j, \forall j, \text{ or, } Y = W^\top X$
- ... or nonlinear (implicit).
- Mapping Φ required to: Preserve proximity? Maximize variance? Preserve a certain graph?
- We say that the data (x_i 's) is embedded into \mathbb{R}^d (the y_i 's)

Basics: Principal Component Analysis (PCA)

In *Principal Component Analysis* W is computed to:

Maximize variance of projected data:

$$\max_{W \in \mathbb{R}^{m \times d}; W^T W = I} \sum_{i=1}^n \left\| y_i - \frac{1}{n} \sum_{j=1}^n y_j \right\|_2^2, \quad y_i = W^T x_i.$$

➤ Leads to maximizing

$$\text{Tr} [W^T (X - \mu e^T)(X - \mu e^T)^T W], \quad \mu = \frac{1}{n} \sum_{i=1}^n x_i$$

➤ Solution $W = \{ \text{dominant eigenvectors} \}$ of the covariance matrix \equiv Set of left singular vectors of $\bar{X} = X - \mu e^T$

SVD:

$$\bar{X} = U\Sigma V^\top, \quad U^\top U = I, \quad V^\top V = I, \quad \Sigma = \text{Diag}$$

- Optimal $W = U_d \equiv$ matrix of first d columns of U
- Solution W also minimizes 'reconstruction error' ..

$$\sum_i \|x_i - WW^\top x_i\|^2 = \sum_i \|x_i - Wy_i\|^2$$

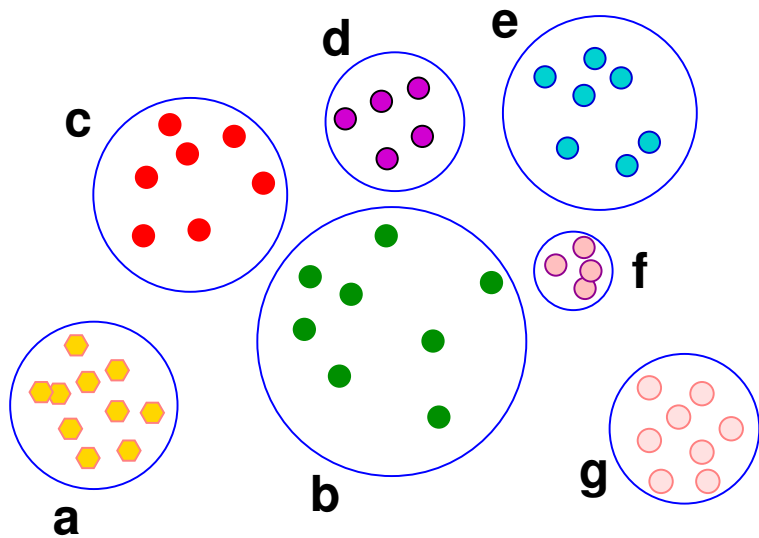
- In some methods recentering to zero is not done, i.e., \bar{X} replaced by X .

SUPERVISED LEARNING

Supervised learning

➤ We now have data that is 'labeled'

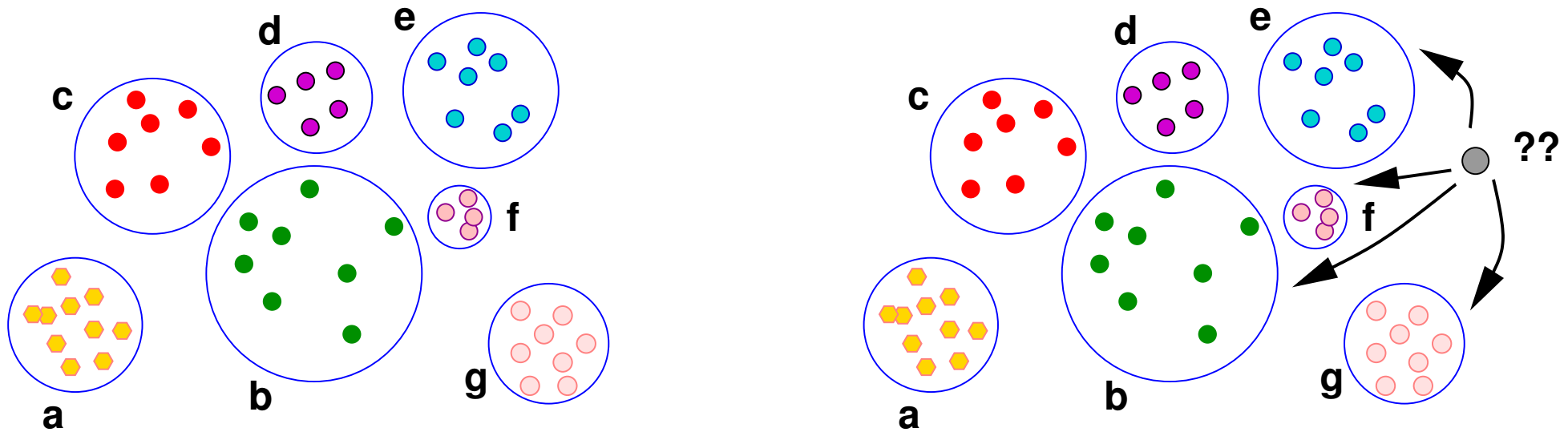
Examples: Health Sciences ('malignant'- 'non malignant') ; Materials ('photovoltaic', 'hard', 'conductor', ...) ; Digit Recognition ('0', '1', ..., '9')



Supervised learning

➤ We now have data that is 'labeled'

Examples: Health Sciences ('malignant'- 'non malignant') ; Materials ('photovoltaic', 'hard', 'conductor', ...) ; Digit Recognition ('0', '1', ..., '9')

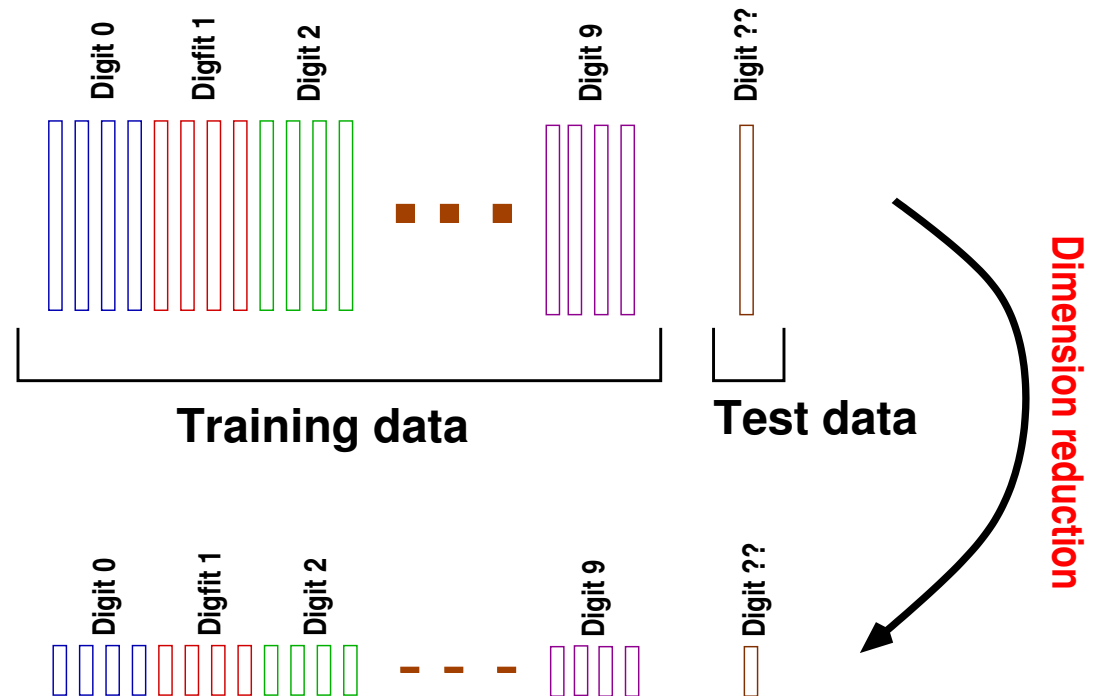


Supervised learning: classification

- Best illustration: written digits recognition example

Given: set of labeled samples (training set), and an (unlabeled) test image x .

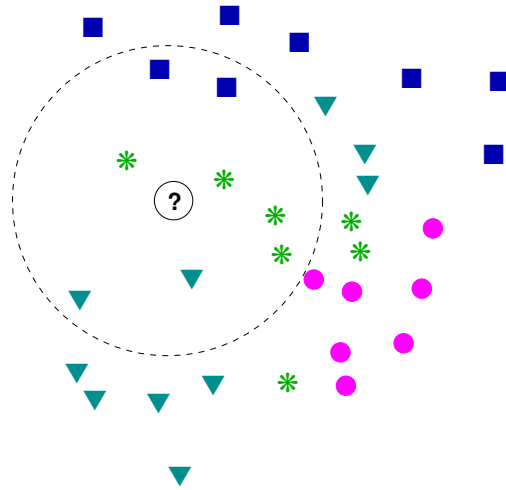
Problem: label of $x = ?$



- Roughly speaking: we seek dimension reduction so that recognition is 'more effective' in low-dim. space

Basic method: K -nearest neighbors (KNN) classification

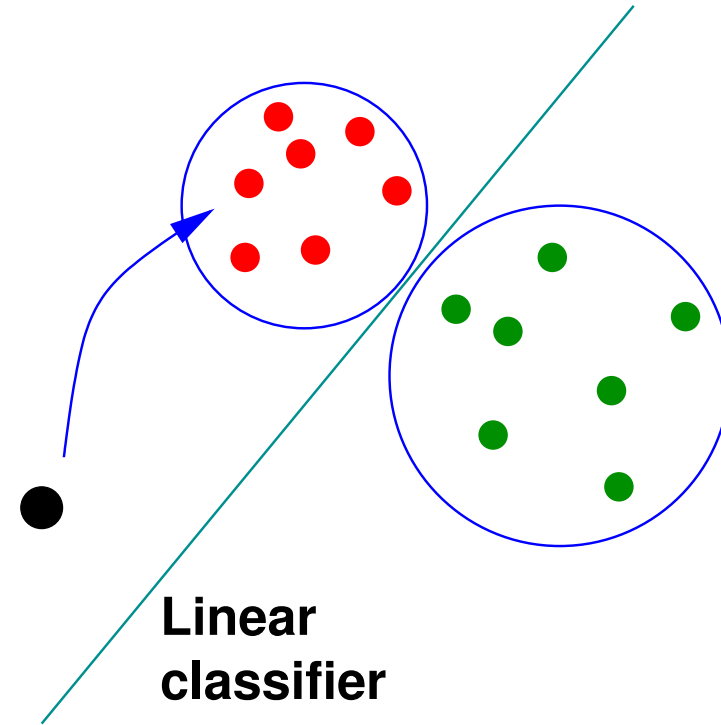
- Idea of a voting system: get distances between test sample and training samples
- Get the k nearest neighbors (here $k = 8$)
- Predominant class among these k items is assigned to the test sample (“*” here)



Supervised learning: Linear classification

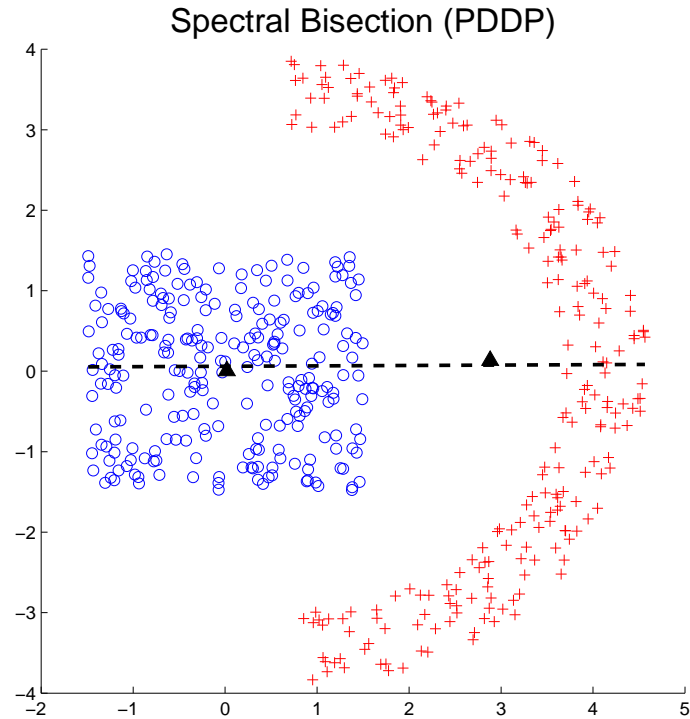
Linear classifiers: Find a hyperplane which best separates the data in classes A and B.

➤ Example of application: Distinguish between SPAM and non-SPAM e-mails



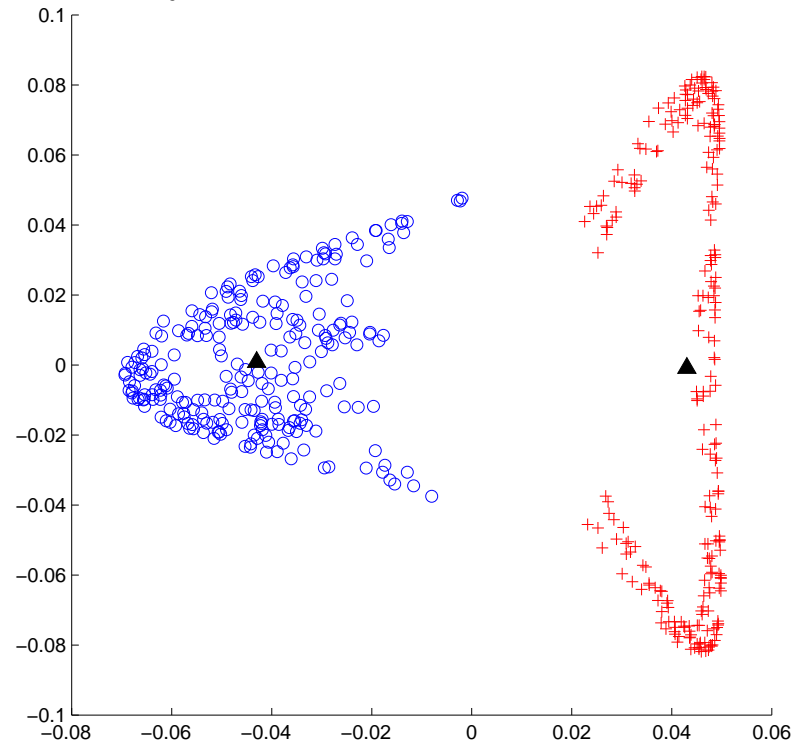
➤ Note: The world is non-linear. Often this is combined with **Kernels** – amounts to changing the inner product

A harder case:



➤ Use kernels to transform

Projection with Kernels -- $\sigma^2 = 2.7463$

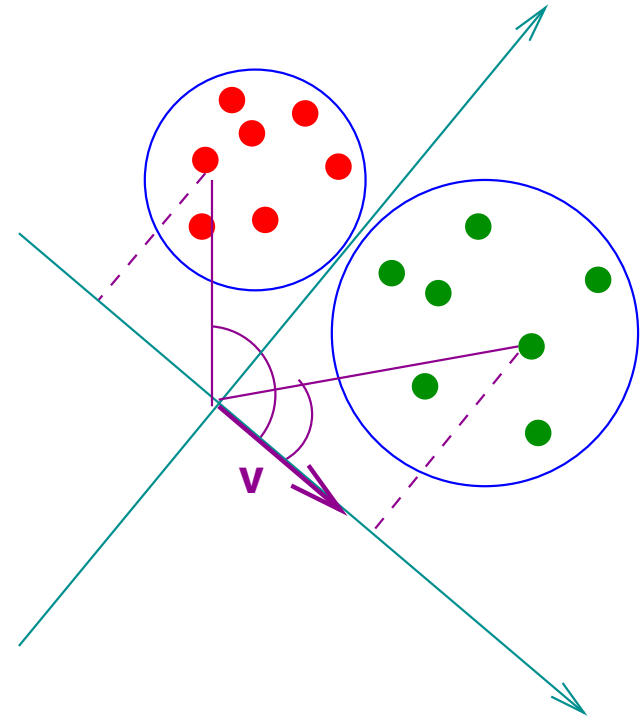


Transformed data with a Gaussian Kernel

Simple linear classifiers

- Let $X = [x_1, \dots, x_n]$ be the data matrix.
- and $L = [l_1, \dots, l_n]$ labels. $l_i = \pm 1$
- 1st Solution: Find a vector u such that $u^T x_i$ close to $l_i, \forall i$
- Common solution: SVD to reduce dimension of data [e.g. 2-D] then do comparison in this space. e.g.

$$A: u^T x_i \geq 0, B: u^T x_i < 0$$



[For clarity: principal axis u drawn below where it should be]

Fisher's Linear Discriminant Analysis (LDA)

Principle: Use label information to build a good projector, i.e., one that can 'discriminate' well between classes

- Define “**between scatter**”: a measure of how well separated two distinct classes are.
- Define “**within scatter**”: a measure of how well clustered items of the same class are.
- Objective: make “between scatter” measure large **and** “within scatter” small.

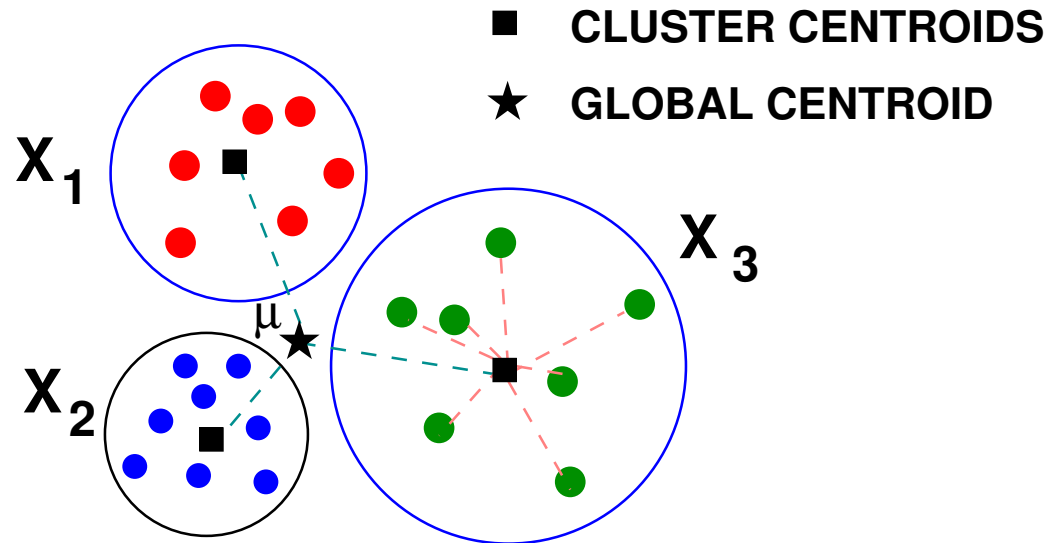
Idea: Find projector that maximizes the ratio of the “between scatter” measure over “within scatter” measure

$$S_B = \sum_{k=1}^c n_k (\mu^{(k)} - \mu) (\mu^{(k)} - \mu)^T,$$

$$S_W = \sum_{k=1}^c \sum_{x_i \in X_k} (x_i - \mu^{(k)}) (x_i - \mu^{(k)})^T$$

where:

- $\mu = \text{mean}(X)$
- $\mu^{(k)} = \text{mean}(X_k)$
- $X_k = k\text{-th class}$
- $n_k = |X_k|$



$$a^T S_B a = \sum_{i=1}^c n_k |a^T (\mu^{(k)} - \mu)|^2,$$

$$a^T S_W a = \sum_{k=1}^c \sum_{x_i \in X_k} |a^T (x_i - \mu^{(k)})|^2$$

➤ Consider 2nd moments for a vector a :

➤ $a^T S_B a \equiv$ weighted variance of projected μ_j 's

➤ $a^T S_W a \equiv$ w. sum of variances of projected classes X_j 's

➤ LDA projects the data so as to maximize the ratio of these two numbers:

$$\max_a \frac{a^T S_B a}{a^T S_W a}$$

➤ Optimal $a =$ eigenvector associated with top eigenvalue of:

$$S_B u_i = \lambda_i S_W u_i .$$

LDA – Extension to arbitrary dimensions

➤ Criterion: maximize the ratio of two traces:

$$\frac{\text{Tr} [U^T S_B U]}{\text{Tr} [U^T S_W U]}$$

➤ Constraint: $U^T U = I$ (orthogonal projector).

➤ Reduced dimension data: $Y = U^T X$.

Common viewpoint: hard to maximize, therefore ...

➤ ... alternative: Solve instead the ('easier') problem:

$$\max_{U^T S_W U = I} \text{Tr} [U^T S_B U]$$

➤ Solution: largest eigenvectors of $S_B u_i = \lambda_i S_W u_i$.

In Brief: Support Vector Machines (SVM)

➤ Similar in spirit to LDA. Formally, SVM finds a hyperplane that best separates two training sets belonging to two classes.

➤ If the hyperplane is:

$$w^T x + b = 0$$

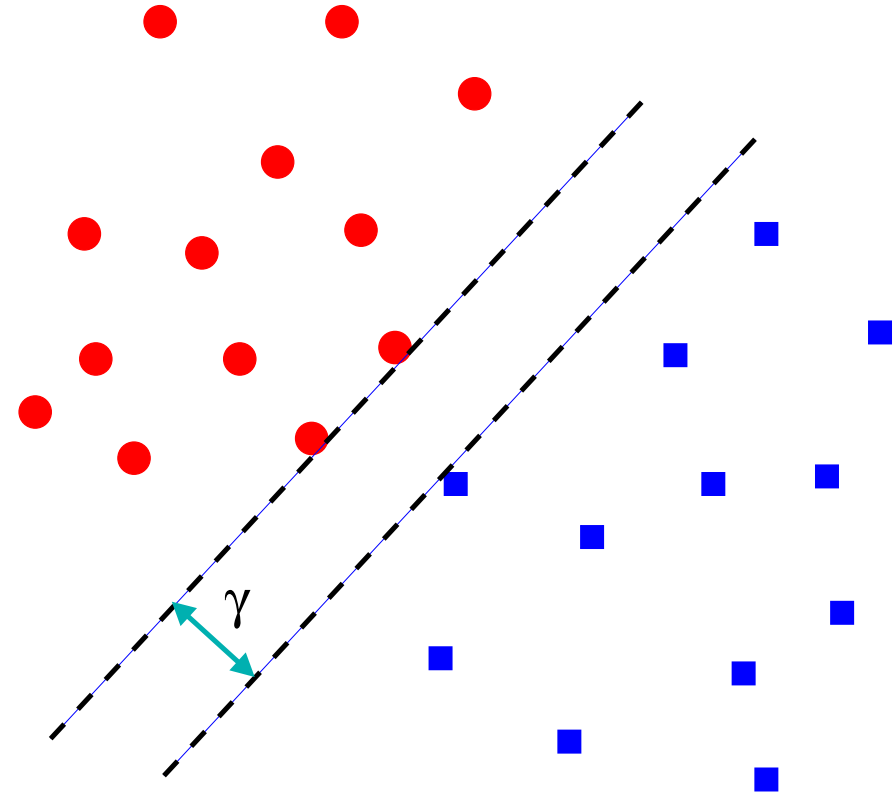
➤ Then the classifier is $f(x) = \text{sign}(w^T x + b)$: assigns $y = +1$ to one class and $y = -1$ to other

➤ Normalize parameters w, b by looking for hyperplanes of the form $w^T x + b \geq 1$ to include one set and $w^T x + b \leq -1$ to include the other.

➤ With $y_i = +1$ for one class and $y_i = -1$ for the other, we can write the constraints as $y_i(w^T x_i + b) \geq 1$.

➤ The margin is the maximum distance between two such planes: goal find w, b to maximize margin.

➤ Maximize margin subject to the constraint $y_i(w^T x_i + b) \geq 1$.



➤ As it turns out the margin is equal to:

$$\gamma = \frac{2}{\|w\|_2}$$

 39 Prove it.


➤ Need to solve the constrained quadratic programming problem:

$$\begin{array}{ll} \min_{w,b} & \frac{1}{2} \|w\|_2^2 \\ \text{s.t.} & y_i(w^T x_i + b) \geq 1, \quad \forall x_i. \end{array}$$

Modification 1: Soft margin. Consider hinge loss: $\max\{0, 1 - y_i[w^T x_i + b]\}$

➤ Zero if constraint satisfied for pair x_i, y_i . Otherwise proportional to distance from corresponding hyperplane. Hence we can minimize

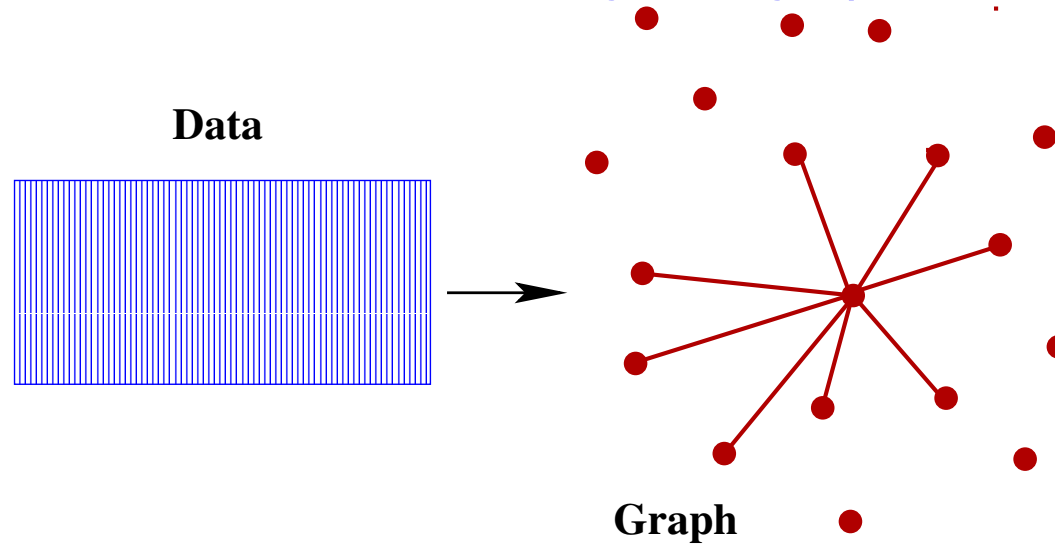
$$\lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i[w^T x_i + b]\}$$

 40 Suppose $y_i = +1$ and let $d_i = 1 - y_i[w^T x_i + b]$. Show that the distance between x_i and hyperplane $w^T x_i + b = +1$ is $d_i / \|w\|$.

Modification 2: Use in combination with a Kernel to improve separability

Building a nearest neighbor graph

- Question: How to build a nearest-neighbor graph from given data?



- Will demonstrate the power of a divide a conquer approach combined with the Lanczos algorithm.

Recall: Two common types of nearest neighbor graphs

ϵ -graph:

Edges consist of pairs (x_i, x_j) such that $\rho(x_i, x_j) \leq \epsilon$

k NN graph:

Nodes adjacent to x_i are those nodes x_ℓ with the k with smallest distances $\rho(x_i, x_\ell)$.

- ϵ -graph is undirected and is geometrically motivated. Issues: 1) may result in disconnected components 2) what ϵ ?
- k NN graphs are directed in general (can be trivially fixed).
- k NN graphs especially useful in practice.

Divide and conquer KNN: key ingredient

- Key ingredient is *Spectral bisection*
- Let the data matrix $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$
- Each column == a data point.
- Center the data: $\hat{X} = [\hat{x}_1, \dots, \hat{x}_n] = X - ce^T$
where c == centroid; $e = \text{ones}(d, 1)$ (matlab)

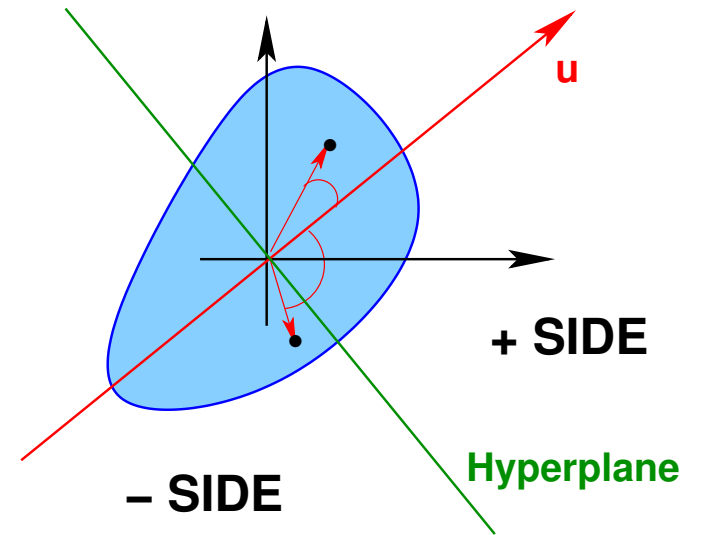
Goal: Split \hat{X} into halves using a hyperplane.

Method: Principal Direction Divisive Partitioning D. Boley, '98.

Idea: Use the (σ, u, v) = largest singular triplet of \hat{X} with: $u^T \hat{X} = \sigma v^T$.

Hyperplane $\langle u, x \rangle = 0$
splits data points in 2
subsets:

$$X_+ = \{x_i \mid u^T \hat{x}_i \geq 0\}$$
$$X_- = \{x_i \mid u^T \hat{x}_i < 0\}$$



► Note that $u^T \hat{x}_i = u^T \hat{X} e_i = \sigma v^T e_i \rightarrow$

$$X_+ = \{x_i \mid v_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid v_i < 0\},$$

where v_i is the i -th entry of v .

- In practice: replace above criterion by

$\mathbf{X}_+ = \{x_i \mid v_i \geq \text{med}(v)\}$ & $\mathbf{X}_- = \{x_i \mid v_i < \text{med}(v)\}$ where $\text{med}(v) ==$ median of the entries of v .

- For largest singular triplet (σ, u, v) of \hat{X} : use Golub-Kahan-Lanczos algorithm or Lanczos applied to $\hat{X}\hat{X}^T$ or $\hat{X}^T\hat{X}$
- Cost (assuming s Lanczos steps) : $O(n \times d \times s)$; Usually: d very small

Reference:

Jie Chen, Haw-Ren Fang and YS, “Fast Approximate k NN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection” JMLR, vol. 10, pp. 1989-2012 (2009).