Semi-Supervised Learning

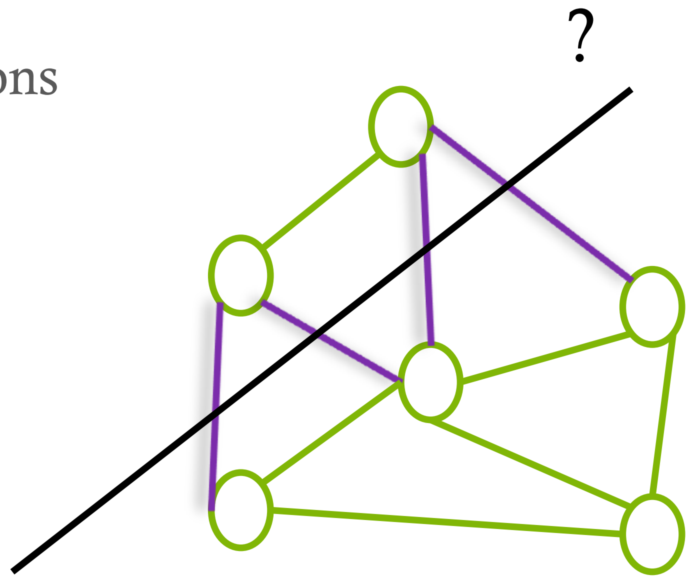
Ahmed Taha
Feb 2014

Content

- Concept Introduction
- Graph cut and Least Square solution
- Eigen vector and Eigen Functions
- Application

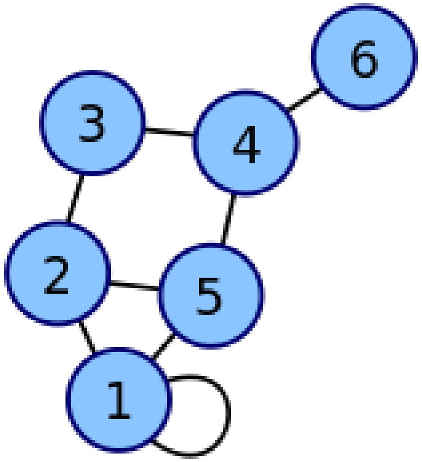
Concept Introduction

- Graph Cut
  - Divide Graph Into two divisions
  - Lowest Cut cost



Concept Introduction

- Degree Matrix / variation

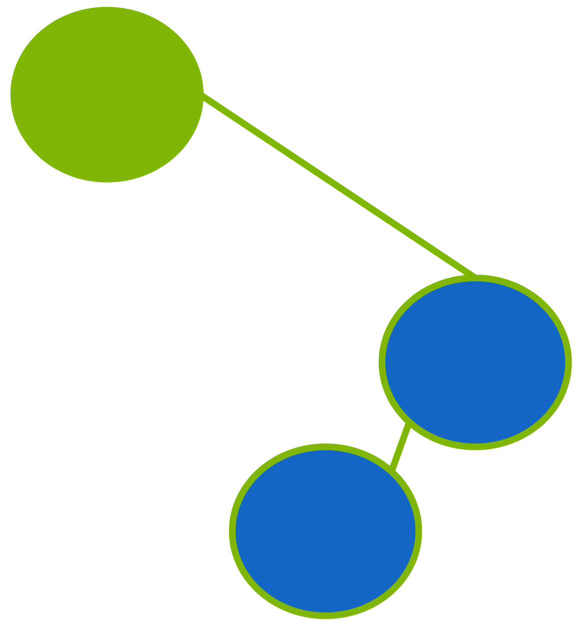


$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Concept Introduction

- Object Representation
  - 2D Point
  - 3D Point
  - Pixel
  - Or even a Whole Image

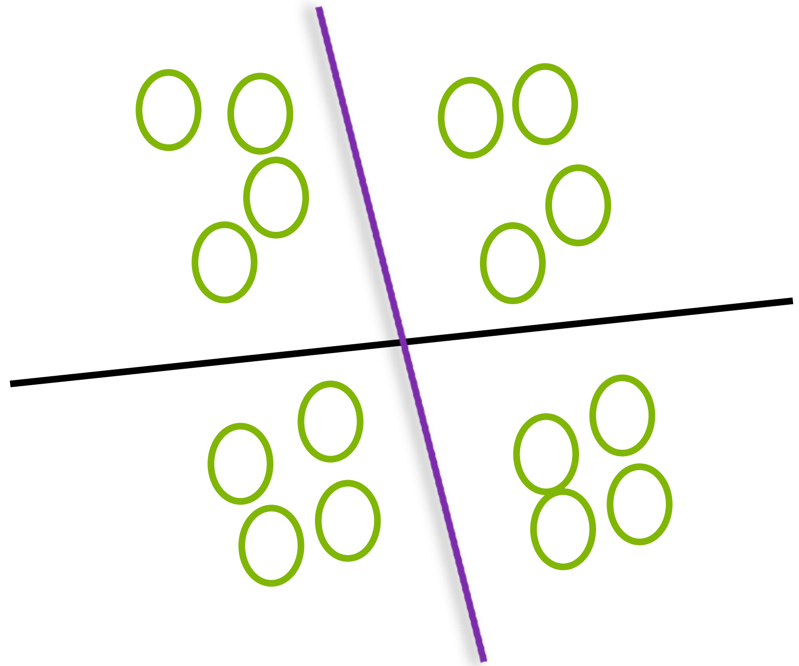
- It is ALL nodes and Edges



Concept Introduction

Semi-supervised learning vs. Un-supervised learning

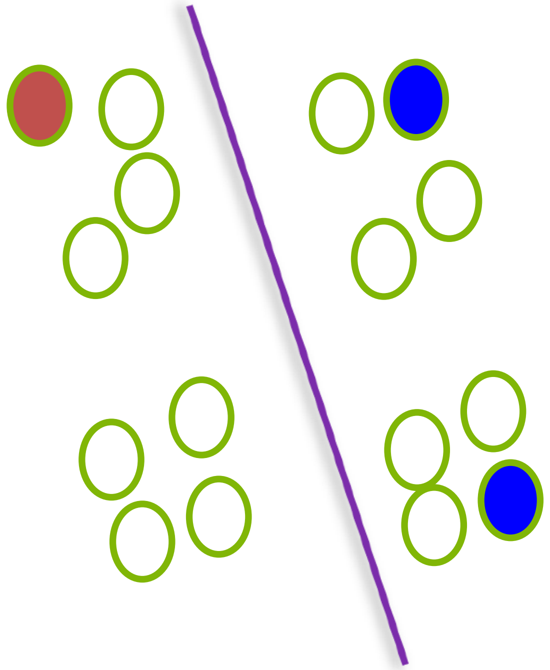
- Un-supervised Learning (No Labeled Data)



Concept Introduction

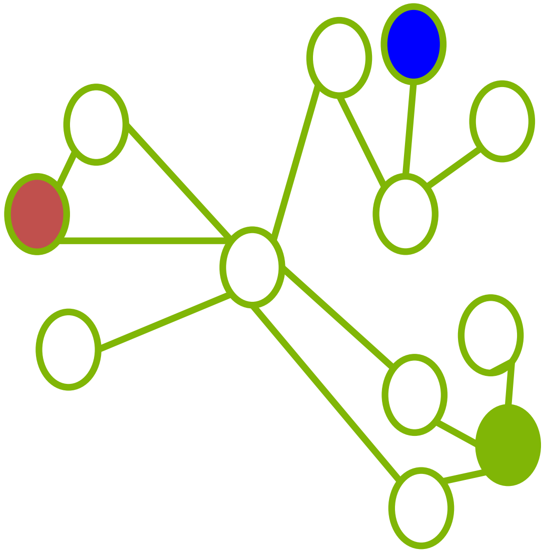
Semi-supervised learning vs. Un-supervised learning

- Semi-Supervised Learning (Labeled Data and structure of unlabeled Data)



Graph Cut
Least square Solution

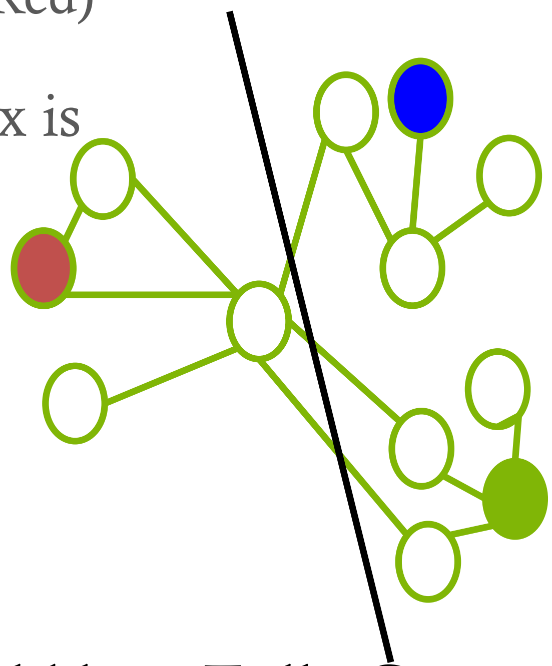
- Semi-Supervised Learning (Labeled Data)
  - We have 3 Objects Now



This Should be a Fully Connected Graph

Graph Cut
Least square Solution

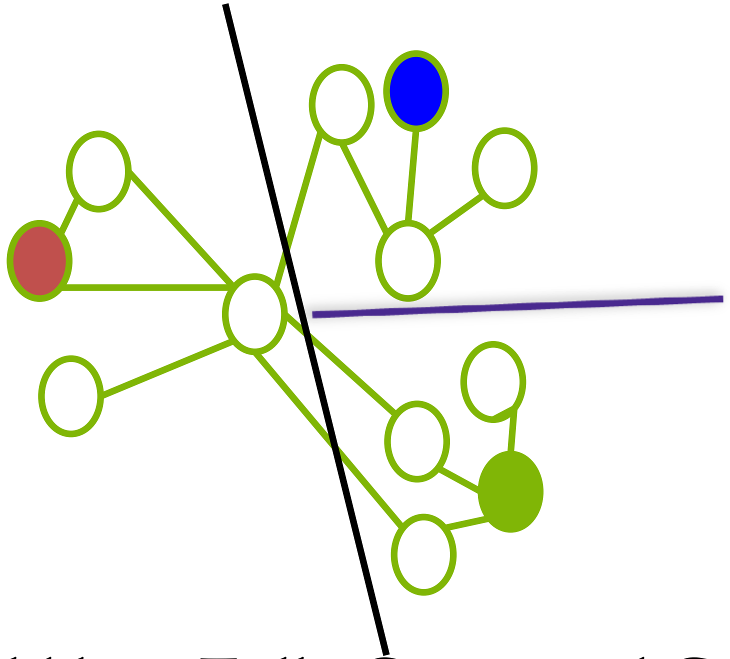
- Objective separate graph into two part
  - (Red and Non-Red)
- Size if this Matrix is
  - N^2
  - Not sparse



This Should be a Fully Connected Graph

Graph Cut
Least square Solution

- We can after that divide the rest of graph into blue and not blue and so on
- NP Problem ?



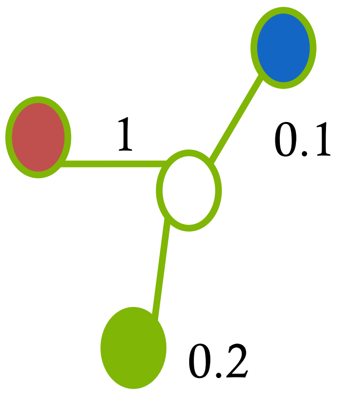
This Should be a Fully Connected Graph

Graph Cut
Least square Solution

- Current Situation , we have a fully connected Graph , represented in NxN Matrix = W (Similarity Matrix)
- We expect each object to be assigned {1,-1} ➔{Red, non-red} with lowest cost assignment cost
- But this is NP ???

Label Propagation Least square Solution

- Weighted Average concept
  - New Node
    - (Red Now)1 * 1
    - (Blue) -1 * 0.1
    - Green -1 * 0.2
- 1-0.1-0.2 = 0.7 ,
- so it is probably a Red Object {1}



Label Propagation
Least square Solution

- Here comes the first Equation , Lets define
  - Matrix W (NxN) , Similarity Between Objects
  - Matrix D (NxN), degree of each Object
  - Matrix L (Laplacian Matrix) = D – W
- Label vector F (Nx1), assignment of each object [-1,1] and not {-1,1}
- Objective Function ➔ Min ½ $\sum_{i,j} w_{ij} (f_i - f_j)$

Least square Solution

- Objective Function ➔ Min ½ $\sum_{i,j} w_{ij} (f_i - f_j)$
- But this doesn't't consider Label data yet

$$½ \sum_{i,j} W_{ij} (f_i - f_j) + \sum_i \lambda (f_i - y)$$

- After some Equation manipulation

$$f = (L + \Lambda)^{-1} \Lambda Y$$

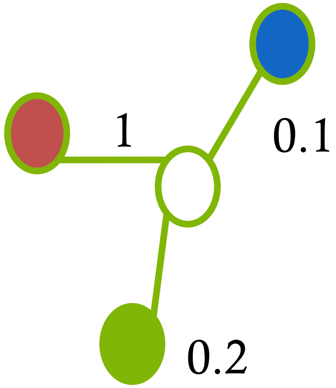
Least square Solution

- We need to solve NxN
  - NxN matrix Inverse
  - NxN matrix multiplication

- Need to reduce dimensions by using Eigenvectors of Graph Laplacian

EigenVector

- As mentioned before we want to have a Label vector f
- f = *α U , so once we have U, we can get α and then we get f*
- Laplacian Eigenmap dimension reduction
- L have the characteristic is this Graph
- *Mapping the objects into a new dimension*



EigenVector

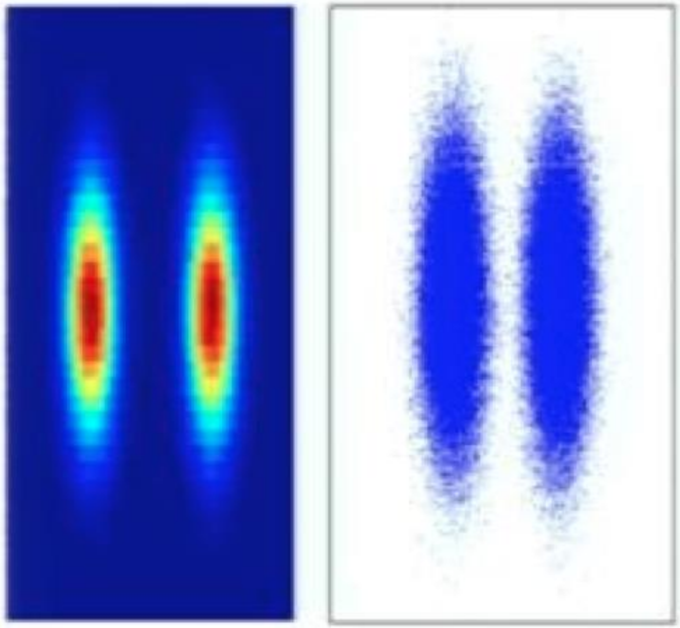
- As mentioned before we want to have a Label vector f
- Get the EigenVectors (U) of Laplacian Matrix (L)
- f = αU *, so once we have U, we can get* α *and then we get f*

$$(\Sigma + U^T \Lambda U)\alpha = U^T \Lambda y$$

- We still need to work with NxN Matrix, at least we compute its Eigen vectors

Eigen Function

- Eigenfunction are limit of Eigenvectors as n ➔ ∞
- For each dimension (2),
  - we calculate the Eigenvector by interpolating the Eigen function from the histogram of this dimension
- Which takes a lot less than
- Need more explanation ☹

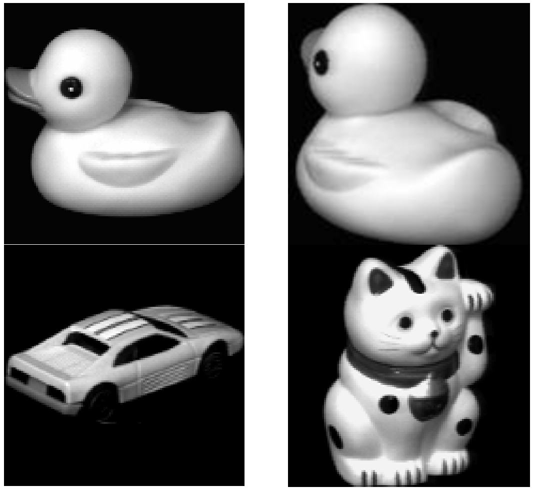


Eigen Function

- Eigenfunction are limit of Eigenvectors as n ➔ ∞
- Notice solution of Eigenfunction is based on the number of Dimensions, while Eigenfunction is based on number of Objects
  - Images Pixels as Object
  - Images with local features as dimension

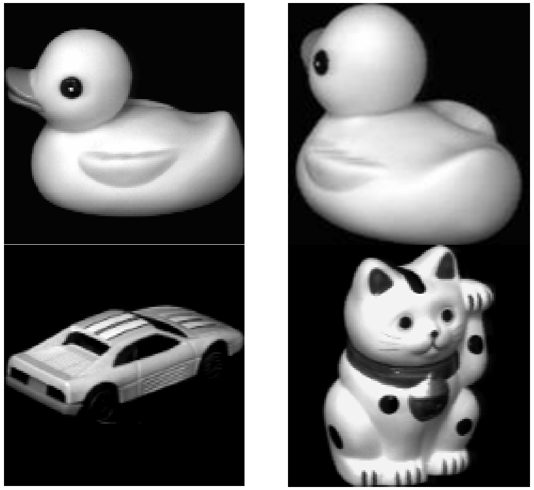
Application
Object Classification

- Coil 20 Dataset
  - 20 Different Object
  - Each Object has 72 different pose



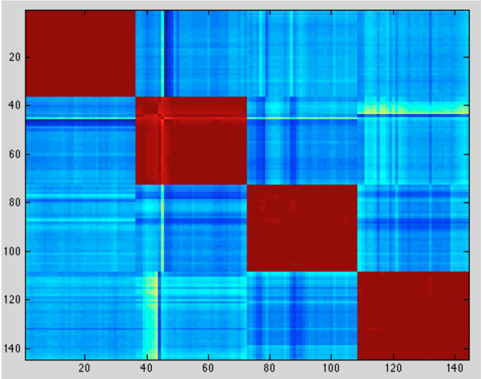
Application
Object Classification

- Our Experiment
  - Label some of these Images
    - Both Positive and Negative Labels
  - Use the LSQ , EigenVector, EigenFunction to compute the labels of the Unlabled data

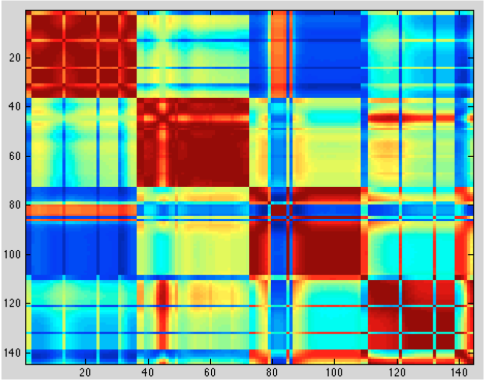


Application
Object Classification

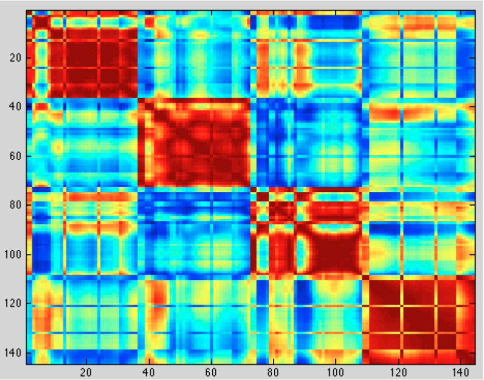
- Our Results



LSQ Solution



EigenVector Solution



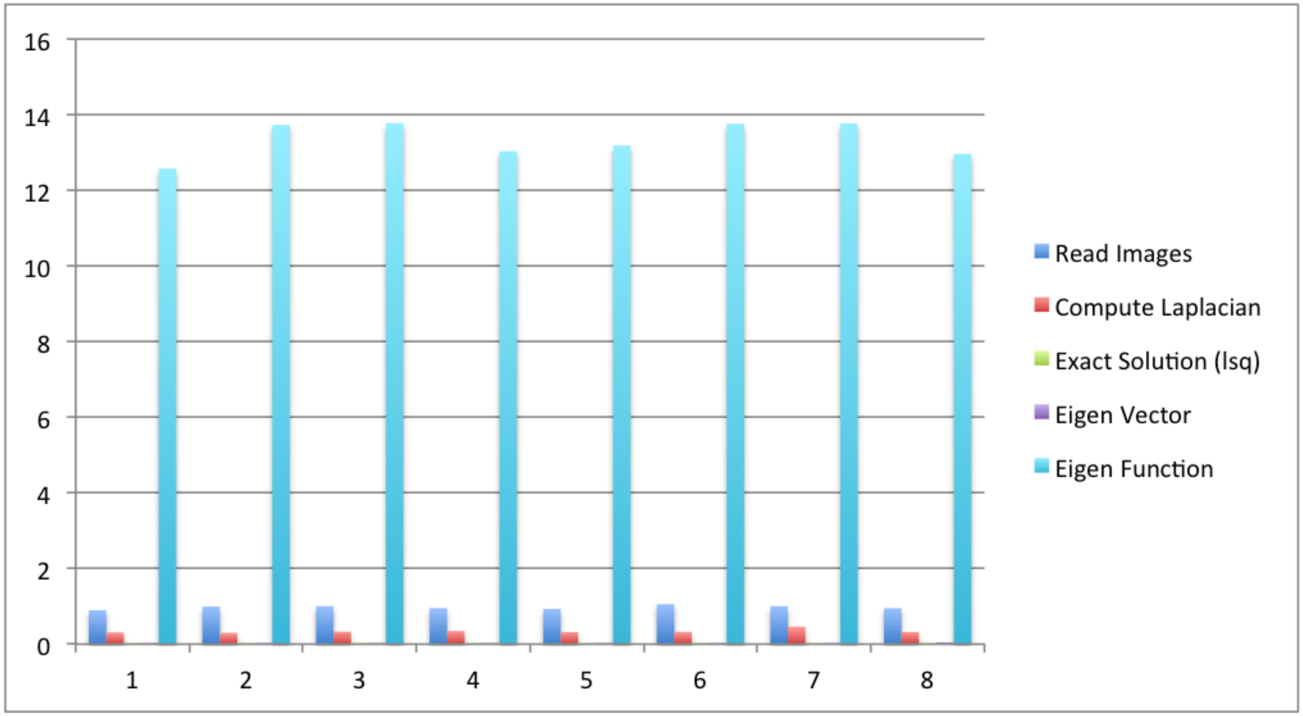
Eigenfunction Solution

Application
Object Classification

- Results Analysis
  - LSQ solution is almost perfect since it is almost exact Solution
  - EigenVector generate approximate solution but in less time, which makes more sense it is just solving one NxN Matrix to get Eigen Vectors
  - Eigen Function method also generated an approximate solution but its time was worse

Application
Object Classification

- Time Results Analysis



Application
Object Classification

- Results Explanation
  - We have 4 (Object) * 36 (pose per object) so total of 144 Object so Matrix laplaican is of size 144
  - Each Image has 128*128 (gray scale) pixel so total of 16384 , so each object have 16384 dimension
- 144 Object vs 16384 dimension

Application
Object Classification

- Results Explanation
- 144 Object vs 16384 dimension
- So it is expected that LSQ , EigenVector method to finish faster since Matrix L is not that big
- While Eigen-function will take a long time to compute the Eigen-function for each dimension 16384

Thanks ☺