

# **LABEL PROPAGATION ON GRAPHS.** **SEMI-SUPERVISED LEARNING**

----Changsheng Liu

10-30-2014

# Agenda

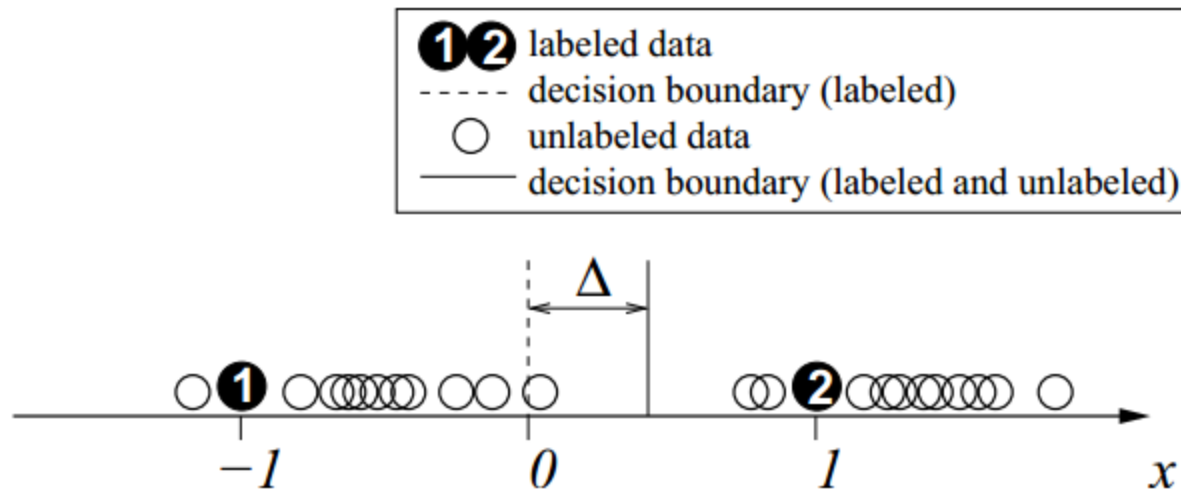
---

- Semi Supervised Learning
- Topics in Semi Supervised Learning
  - ▣ Label Propagation
  - ▣ Local and global consistency
  - ▣ Graph Kernels by Spectral Transforms
  - ▣ Gaussian field and Harmonic Function
- Reference

# Semi Supervised Learning

- **Semi-supervised learning** is a class of **supervised learning** tasks and techniques that also make use of unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data.
- **Why**
  - ▣ Labeled data is hard to get
    - Expensive, human annotation, time consuming
    - May require experts
  - ▣ Unlabeled data is cheap

# Why unlabeled data helps[1]



- assuming each class is a coherent group (e.g. Gaussian)
- with and without unlabeled data: decision boundary shift

# Label Propagation[2]

- Assumption
  - ▣ Closer data points tend to have similar class labels.
- General Idea
  - ▣ A node's labels propagate to neighboring nodes according to their proximity
  - ▣ Clamp the labels on the **labeled data**, so the labeled data could act like a sources that push out labels to unlabeled data.

# Set up

- Input  $x$ , label  $y$
- Labeled data  $(x_1, y_1), (x_2, y_2) \dots (x_l, y_l)$
- Unlabeled data  $(x_{l+1}, y_{l+1}) \dots (x_{l+u}, y_{l+u})$
- $l \ll u$
- weight  $w_{ij} = \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) = \exp\left(-\frac{\sum_{d=1}^D (x_i^d - x_j^d)^2}{\sigma^2}\right)$

# Probabilistic Transition Matrix

- Allow larger edge weight to propagate labels easier

$$T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}}$$

- $T_{ij}$  is the probability to jump from node  $j$  to  $i$
- Normalized  $T$

$$T_{ij} = \frac{T_{ij}}{\sum_k T_{ik}}$$

For example:

|     |     |     |
|-----|-----|-----|
| 0.3 | 0.2 | 0.5 |
| 0.2 | 0.1 | 0.7 |
| 0.5 | 0.7 | 0.6 |

→ The probability node 3  
jump to Node 1 is 0.5

← The probability node 2  
jump to Node 3 is 0.7

# Matrix Y

- Define  $(l+u) * C$  label matrix  $Y$ , whose  $i$ th row representing label probability distribution of node  $x_i$ 
  - ▣  $Y_{ij}=1$ , if the class of  $x_i$  is  $c_j$ , else 0, for labeled data
  - ▣ The initialization of row of  $Y$  corresponding to unlabeled data is not important

|   |     |     |     |  |
|---|-----|-----|-----|--|
|   | 0   | 1   | 0   |  |
| □ | 0.2 | 0.5 | 0.3 |  |
|   | 0.7 | 0.2 | 0.1 |  |

Node 1 is labeled as label 2.

The label distribution of node 3. For example, 0.7 is the probability that node 3 is label 1 .



# Algorithm

- 1 Propagate  $Y \leftarrow TY$ 
  - ▣ Labels spread information along local structure
- 2 Row normalize  $Y$ 
  - ▣ Keep proper distribution over classes
- 3 Clamp the labeled data, Repeat from step 1 until  $Y$  converges
  - ▣ Keep originally labeled points

# Convergence

- The first two steps  $Y \leftarrow \bar{T}Y$
- Split  $\bar{T}$   $\bar{T} = \begin{bmatrix} \bar{T}_{ul} & \bar{T}_{lu} \\ \bar{T}_{ul} & \bar{T}_{uu} \end{bmatrix}$
- $Y_U$   $Y_U \leftarrow \bar{T}_{uu}Y_U + \bar{T}_{ul}Y_L$
- General form  $Y_U = \lim_{n \rightarrow \infty} \bar{T}_{uu}^n Y^0 + [\sum_{i=1}^n \bar{T}_{uu}^{(i-1)}] \bar{T}_{ul} Y_L$
- $\bar{T}$  is row normalized,  $\bar{T}_{uu}$  is submatrix of  $\bar{T}$

$$\exists \gamma < 1, \sum_{j=1}^u \bar{T}_{uu_{ij}} \leq \gamma, \forall i = 1 \dots u$$

# Convergence(cont)

- Consider the row sum

$$\begin{aligned}\sum_j \bar{T}_{uu_{ij}}^n &= \sum_j \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \bar{T}_{uu_{kj}} \\ &= \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \sum_j \bar{T}_{uu_{kj}} \\ &\leq \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \gamma \\ &\leq \gamma^n\end{aligned}$$

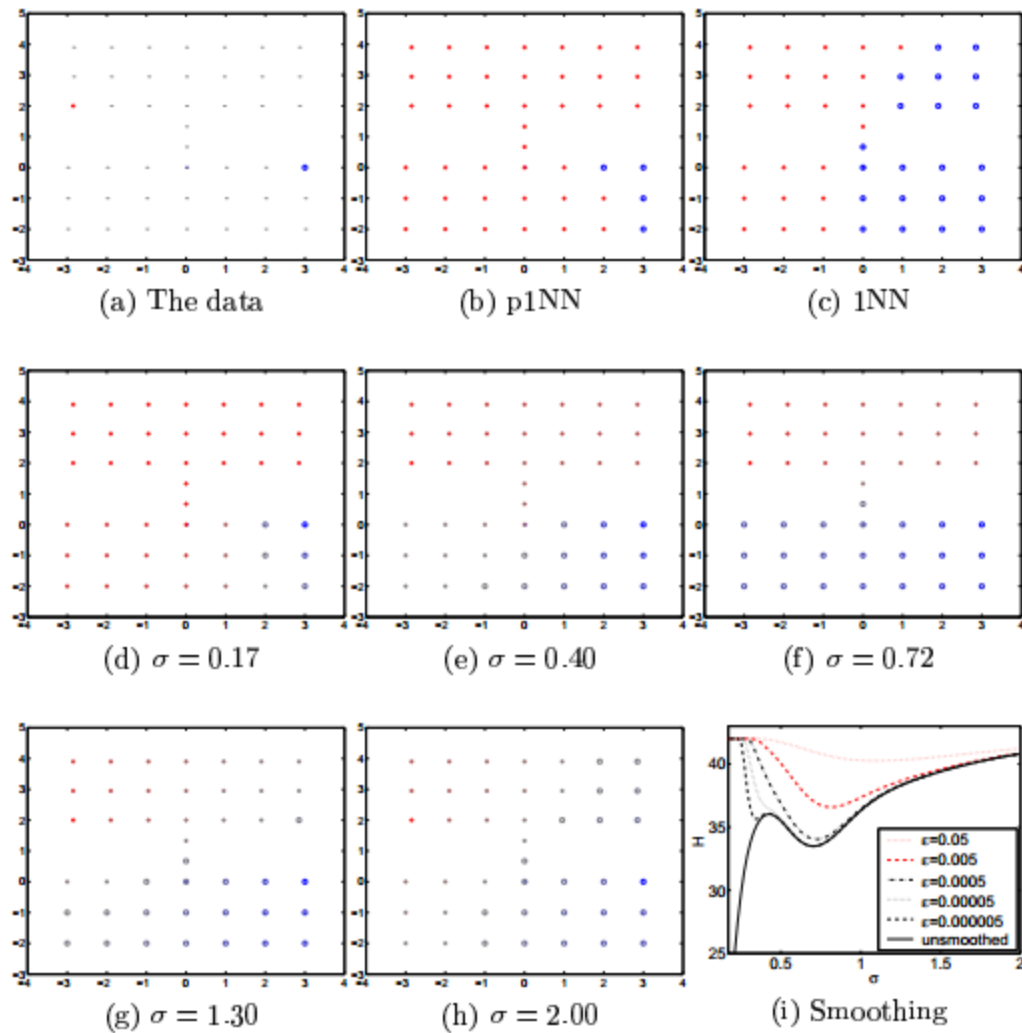
$$Y_U = (I - \bar{T}_{uu})^{-1} \bar{T}_{ul} Y_L$$

No need to iterate!

# Parameter Setting

- How to choose parameter  $\sigma$
- First, use a heuristic method. Finding a minimum spanning tree over all data points with Euclidean distances  $d_{ij}$  with Kruskal's Algorithm(The famous greedy algorithm in data structure).
- Choose the first tree edge that connect two components with different labeled points. The length is  $d_0$ .
- Set  $\sigma = d_0/3$

# The effect of $\sigma$



# Optimizing $\sigma$

- Single parameter  $\sigma$  controls spread of labels
  - ▣ For  $\sigma \rightarrow 0$ , classification of unlabeled points dominated by nearest labeled point
  - ▣ For  $\sigma \rightarrow \infty$ , class probabilities just become class frequencies (no information from label proximity)
- Can minimize entropy of class labels
- $H = -\sum_{ij} Y_{ij} \log Y_{ij}$ 
  - ▣ Leads to confident classifications
  - ▣ However, minimum entropy at  $\sigma = 0$

# Optimizing $\sigma(\text{cont})$

- Add uniform transition component ( $\mathbf{U}_{ij}=1/N$ ) to  $T$ 
$$\tilde{T} = \varepsilon \mathbf{U} + (1 - \varepsilon) T$$
- For small  $\sigma$ , uniform component dominates
  - ▣ Minimum entropy no longer at  $\sigma=0$
- Use  $\sigma_1 \dots \sigma_N$  to scale each dimension independently
- Perform gradient descent with respect to  $\sigma$ 's in order to minimize entropy

$$\frac{\partial H}{\partial \sigma_d} = \sum_{i=L+1}^{L+U} \sum_{c=1}^C \frac{\partial H}{\partial Y_{ic}} \frac{\partial Y_{ic}}{\partial \sigma_d}$$

# Rebalancing Class Proportions

- How should we assign classes to unlabeled points?
- Could choose most likely class
  - ▣ ML method does not explicitly control class proportions
- Suppose we want labels to fit a known or estimated distribution over classes
  - ▣ Normalize class mass – scale columns of  $Y_U$  to fit class distribution and then pick ML class
    - Does not guarantee strict label proportions
  - ▣ Perform label bidding – each entry  $Y_U(i,c)$  is a “bid” of sample  $i$  for class  $c$ 
    - Handle bids from largest to smallest
    - Bid is taken if class  $c$  is not full, otherwise it is discarded



# Experiment result

## □ 3 bands dataset and Spring dataset

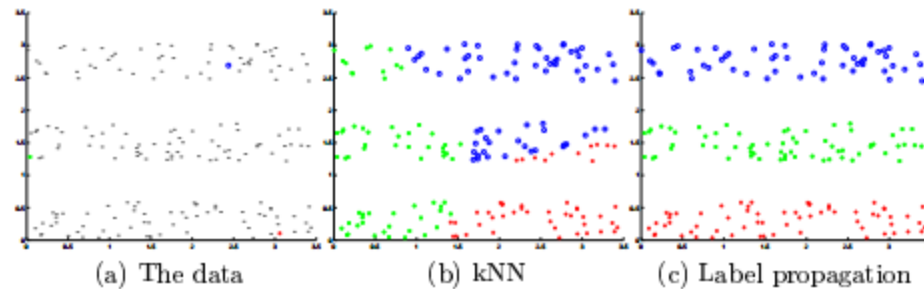


Figure 1: The 3 Bands dataset. Labeled data are color symbols and unlabeled data are dots in (a). kNN ignores unlabeled data structure, while label propagation uses it.

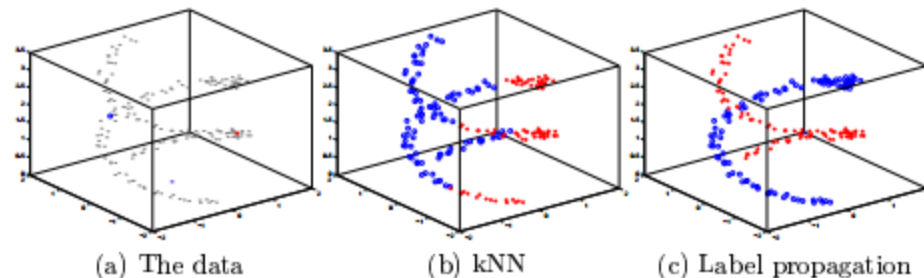


Figure 2: The Springs dataset.

# Learning with local and global consistency[4]

- The key to semi- supervised learning
  - ▣ Nearby points are likely to have the same label
  - ▣ Points on the same structure (cluster or manifold) are likely to have the same label

# A toy example

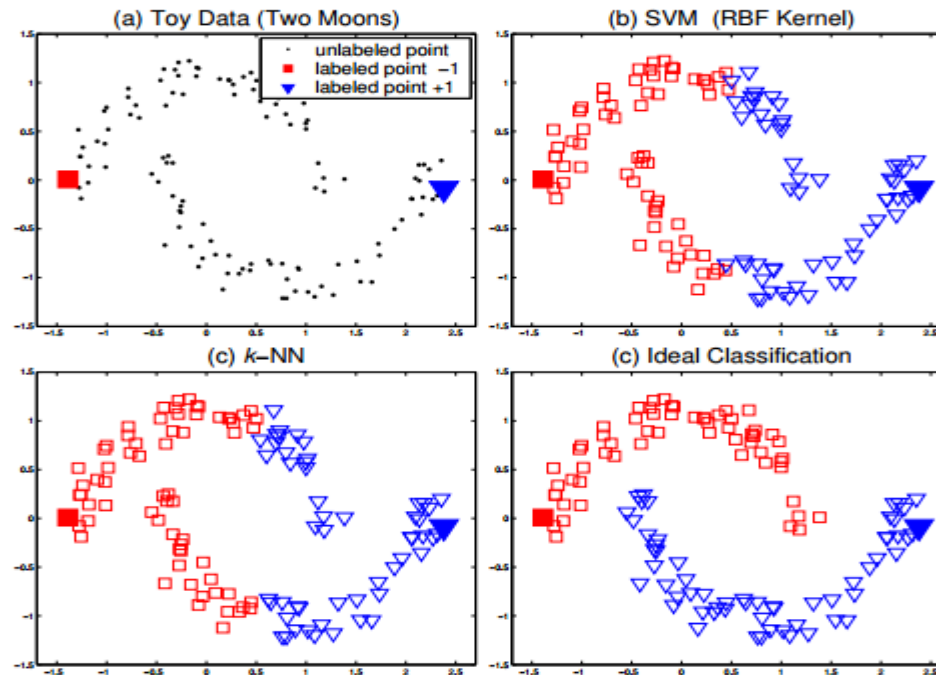


Figure 1: Classification on the two moons pattern. (a) toy data set with two labeled points; (b) classifying result given by the SVM with a RBF kernel; (c)  $k$ -NN with  $k = 1$ ; (d) ideal classification that we hope to obtain.

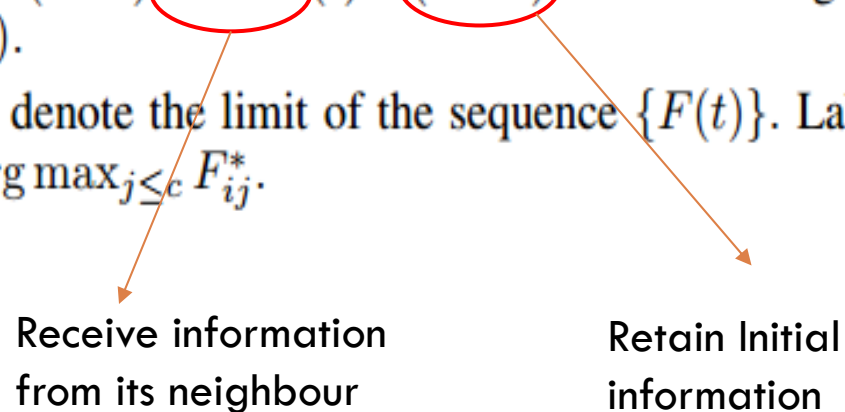
# Objective



Design a classifying function which is sufficiently smooth with respect to the intrinsic structure

# Algorithm

1. Form the affinity matrix  $W$  defined by  $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$  if  $i \neq j$  and  $W_{ii} = 0$ .
2. Construct the matrix  $S = D^{-1/2}WD^{-1/2}$  in which  $D$  is a diagonal matrix with its  $(i, i)$ -element equal to the sum of the  $i$ -th row of  $W$ .
3. Iterate  $F(t+1) = \alpha SF(t) + (1 - \alpha)Y$  until convergence, where  $\alpha$  is a parameter in  $(0, 1)$ .
4. Let  $F^*$  denote the limit of the sequence  $\{F(t)\}$ . Label each point  $x_i$  as a label  $y_i = \arg \max_{j \leq c} F_{ij}^*$ .



Receive information  
from its neighbour

Retain Initial  
information

# Convergence

- The sequence  $\{F(t)\}$  converges, suppose  $F(0)=Y$

$$F^* = (1 - \alpha)(I - \alpha S)^{-1} Y$$

The proof is similar to Label Propagation

# Regularization Framework(A different perspective)

$$Q(F) = \frac{1}{2} \sum_{i,j=1}^n W_{ij} \left\| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \right\|^2 + \mu \sum_{i=1}^n \|F_i - Y_i\|^2$$

Smoothness term, capture the local variations, a good function should not change too much between nearby points

Fitting constraints, loss function, a good classifying function should not change too much from initial label assignment

# Regularization Framework

Property of  
Laplacian Matrix

$$\frac{1}{2} \sum_{i,j=1}^n W_{ij} \left\| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \right\|^2 = f^T D^{-1/2} L_{sym} D^{-1/2} f$$

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

□ Differentiating  $Q(F)$  with respect to  $F$  = I - S

□  $\left. \frac{\partial Q}{\partial F} \right|_{F=F^*} = F^* - S F^* + \mu(F^* - Y) = 0$

□  $F^* - \frac{1}{1+\mu} S F^* - \frac{\mu}{1+\mu} Y = 0$

□  $\alpha = \frac{1}{1+\mu}, \text{ and } \beta = \frac{\mu}{1+\mu}$

□  $(I - \alpha S) F^* = \beta Y$

□  $I - \alpha S$  is invertible,  $F^* = \beta(I - \alpha S)^{-1} Y$



# Experiment

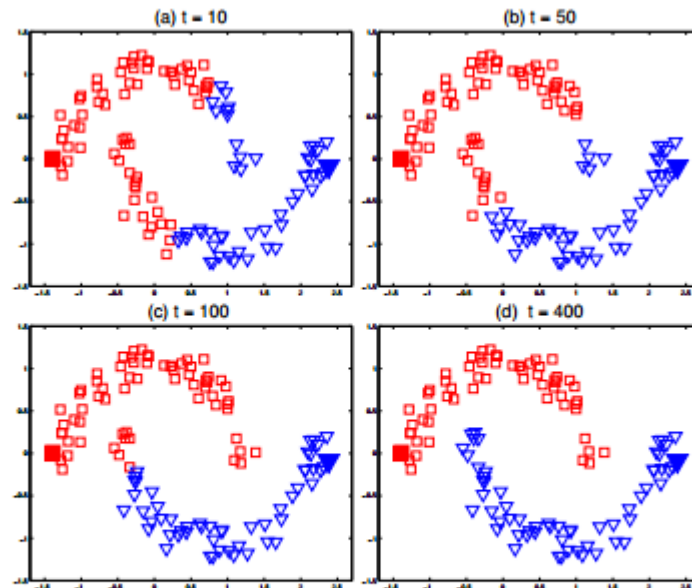


Figure 2: Classification on the pattern of two moons. The convergence process of our iteration algorithm with  $t$  increasing from 1 to 400 is shown from (a) to (d). Note that the initial label information are diffused along the moons.

# Graph Kernels by Spectral Transforms[5]

- Graph-based semi-supervised learning methods can be viewed as imposing smoothness conditions on the target function
- Eigenvectors with small eigenvalues are smooth, and ideally represent large cluster structures within the data.

# Smoothness

## □ Consider the Laplacian $L$

**Proposition 1 (Properties of  $L$ )** *The matrix  $L$  satisfies the following properties:*

1. *For every vector  $f \in \mathbb{R}^n$  we have*

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

2.  *$L$  is symmetric and positive semi-definite.*

3. *The smallest eigenvalue of  $L$  is 0, the corresponding eigenvector is the constant one vector  $\mathbf{1}$ .*

4.  *$L$  has  $n$  non-negative, real-valued eigenvalues  $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ .*

# Smoothness

- Semi-supervised learning creates a smooth function over unlabeled points

$$f : [n] \rightarrow \mathbb{R},$$

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^N W_{ij} (f(i) - f(j))^2$$

- Generally, smooth if  $f(i) \approx f(j)$  for pairs with large  $W_{ij}$
- The smoothness of an eigenvector is

$$\phi_i^T L \phi_i = \lambda_i$$

Eigenvectors with  
smaller eigenvalues  
are smoother

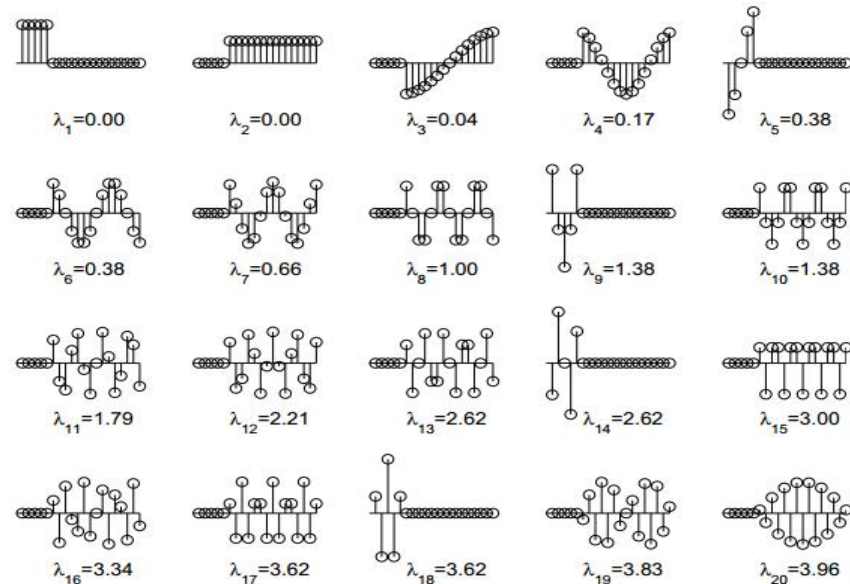


# Smoothness of Eigenvectors

- The complete orthonormal set of eigenvectors  $\phi_1, \phi_2, \dots, \phi_n$

$$L = \sum_{i=1}^n \lambda_i \phi_i \phi_i^T$$

(a) a linear unweighted graph with two segments



(b) the eigenvectors and eigenvalues of the Laplacian  $L$

**Figure 1.1** A simple graph and its Laplacian spectral decomposition. Note the eigenvectors become rougher with larger eigenvalues.

# Kernels by Spectral Transform

- Different weightings (i.e. spectral transforms) of Laplacian eigenvalues leads to different smoothness measures
- We want a kernel  $K$  that respects smoothness
  - ▣ Define using eigenvectors of Laplacian ( $\phi$ ) and eigenvalues of  $K$  ( $\mu$ )

$$K = \sum_{i=1}^N \mu_i \phi_i \phi_i^T$$

- ▣ Can also define in terms of a spectral transform of Laplacian eigenvalues

$$K = \sum_{i=1}^N r(\lambda_i) \phi_i \phi_i^T$$

# Types of Transforms

- $r(\lambda_i)$  is a non-negative and decreasing transform

Regularized Laplacian

$$r(\lambda) = \frac{1}{\lambda + \varepsilon}$$

Diffusion Kernel

$$r(\lambda) = \exp\left(-\frac{\sigma^2}{2} \lambda\right)$$

1 - step Random Walk

$$r(\lambda) = (\alpha - \lambda), \alpha \geq 2$$

p - step Random Walk

$$r(\lambda) = (\alpha - \lambda)^p, \alpha \geq 2$$

Inverse Cosine

$$r(\lambda) = \cos(\lambda\pi / 4)$$

Step Function

$$r(\lambda) = 1 \text{ if } \lambda \leq \lambda_{cut}$$

- Reverses order of eigenvalues, so smooth eigenvectors have larger eigenvalues in  $K$

- Is there an optimal transform?

We need to find a regularizer here

# Kernel Alignment

- Assess fitness of a kernel to training labels
- Empirical kernel alignment compares kernel matrix  $K_{tr}$  for training data to target matrix  $T$  for training data

- ▣  $T_{ij} = 1$  if  $y_i = y_j$ , otherwise  $T_{ij} = -1$

$$\hat{A}(K_{tr}, T) = \frac{\langle K_{tr}, T \rangle_F}{\sqrt{\langle K_{tr}, K_{tr} \rangle_F \langle T, T \rangle_F}}$$

$$\langle M, N \rangle_F = \text{Tr}(MN)$$

Frobenius Product

- Alignment measure computes cosine between  $K_{tr}$  and  $T$
- Find the optimal spectral transformation  $r(\lambda_i)$  using the kernel alignment notion



# Convex Optimization

- Convex set
- Convex function
- Convex Optimization

- ▣ Linear Programming

- Minimize  $c^T x + d$
- Subject to  $Gx \leq h$   
 $Ax = b$

- ▣ Quadratically Constrained Quadratic Programming

- Minimize  $1/2 x^T P x + c^T x + d$
- Subject to  $1/2 x^T Q_i x + r_i^T x + s_i \leq 0$   
 $Ax = b$

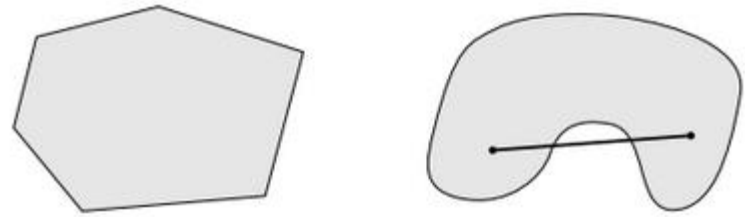
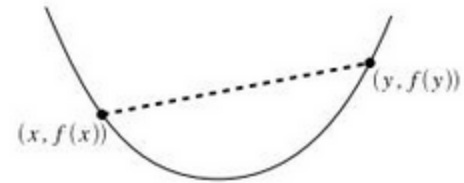


Figure 1: Examples of a convex set (a) and a non-convex set (b).



# QCQP

- Kernel alignment between  $K_{tr}$  and  $T$  is a convex function of kernel eigenvalues  $\mu_i$ 
  - ▣ No assumption on parametric form of transform  $r(\lambda_i)$
- Need  $K$  to be positive semi-definite
  - ▣ Restrict eigenvalues of  $K$  to be  $\geq 0$
- Leads to computationally efficient Quadratically Constrained Quadratic Program
  - ▣ Minimize convex quadratic function over smaller feasible region
  - ▣ Both objective function and constraints are quadratic
  - ▣ Complexity comparable to linear programs

# Impose Order Constraints

- We would like to keep decreasing order on spectral transformation
  - ▣ Smooth functions are preferred – bigger eigenvalues for smoother eigenvectors
- An order constrained semi-supervised kernel  $K$  is the solution to the following convex optimization problem.

$$\begin{array}{ll} \max_K & \hat{A}(K_{tr}, T) \\ \text{subject to} & K = \sum_{i=1}^n \mu_i K_i \\ & \mu_i \geq 0 \\ & \text{Tr}(K) = 1 \\ & \mu_i \geq \mu_{i+1}, \quad i = 1 \cdots n-1 \end{array}$$

# Improved Order Constraints

- Constant eigenvectors act as a bias term in the graph kernel
  - ▣  $\lambda_1=0$ , corresponding eigenvector  $\phi_1$  is constant
  - ▣ Need not constrain bias terms
- Improved Order constrains
  - ▣ Ignore the constant eigenvectors

$\mu_i \geq \mu_{i+1}, i = 1 \dots n - 1, \text{ and } \phi_i \text{ not constant}$

# Harmonic Functions (Zhu, 2003)[3]

- Now define class labeling  $f$  in terms of a Gaussian over continuous space, instead of random field over discrete label set
- Distribution on  $f$  is a Gaussian field

$$p_{\beta}(f) = \frac{e^{-\beta E(f)}}{Z_{\beta}}$$
$$Z_{\beta} = \int_{f|_{L=f_l}} \exp(-\beta E(f)) df$$

- Useful for multi-label problems (NP-hard for discrete random fields)
  - ▣ ML configuration is now unique, attainable by matrix methods, and characterized by harmonic functions

# Harmonic Energy

- “Energy” of solution labeling  $f$  is defined as:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} (f(i) - f(j))^2$$

- Nearby points should have similar labels
- Solution which minimizes  $E(f)$  is harmonic
  - $\Delta f = 0$  for unlabeled points, where  $\Delta = D - W$  (combinatorial Laplacian)
  - $\Delta f = f_l$  for labeled points
  - Value of  $f$  at an unlabeled point is the average of  $f$  at neighboring points

$$f(j) = \frac{1}{d_j} \sum_{i \sim j} w_{ij} f(i), \text{ for } j = L+1, \dots, L+U$$

$$f = D^{-1} W f$$

# Harmonic Solution

- As before, split problem into:

$$f = \begin{bmatrix} f_l \\ f_u \end{bmatrix} \quad W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix} \quad P = D^{-1}W$$

- Solve using  $\Delta f = 0$ ,  $f|_L = f_l$ :

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l$$

- Can be viewed as heat kernel classification, but independent of time parameter

# Summary

- Label Propagation
  - ▣ Propagate and clamp data
- Local and global consistency
  - ▣ Allow  $f(X_i)$  to be different from  $Y_i$ , but penalize it
  - ▣ Introduce a balance between labeled data fit and graph energy
- Graph Kernels by Spectral Transforms
  - ▣ Smoothness, using eigenvector of Laplacian to keep smooth
  - ▣ Use kernel alignment
- Gaussian field and Harmonic Function
  - ▣ The label is discrete (Gaussian)



# Reference

- [1] Zhu, Semi supervised learning tutorial (<http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf>)
- [2] Zhu, Ghahramani [Learning from labeled and unlabeled data](#)
- [3]Zhu, Ghahramani, Lafferty [Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions](#)
- [4]Zhou at al [Learning with Local and Global Consistency](#)
- [5]Zhu et al [Semi-supervised learning](#)
- [6]Matt Stokes, Semi-Supervised Learning