Large Scale Manifold Learning: Review and Progress

Yan-Ming Zhang

National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences

Qingdao, Dec. 23 2015

Contents

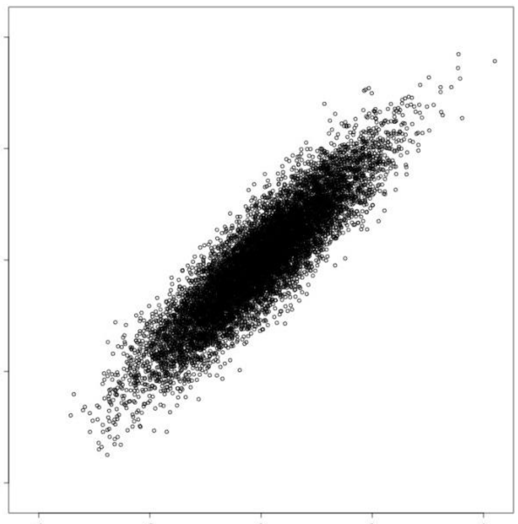
- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

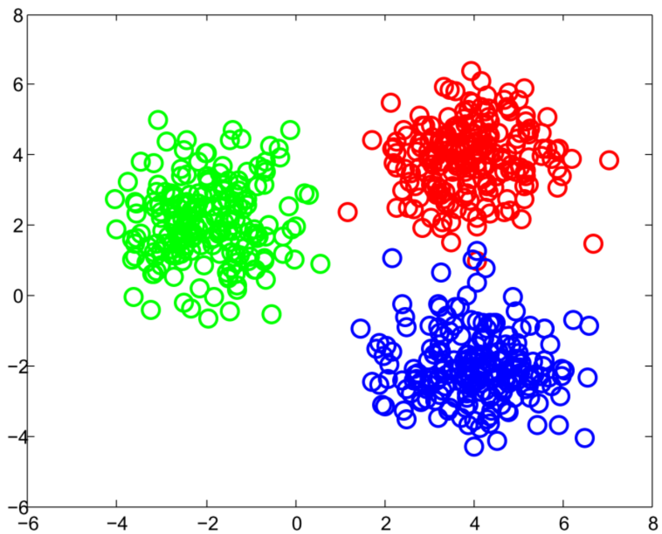
Contents

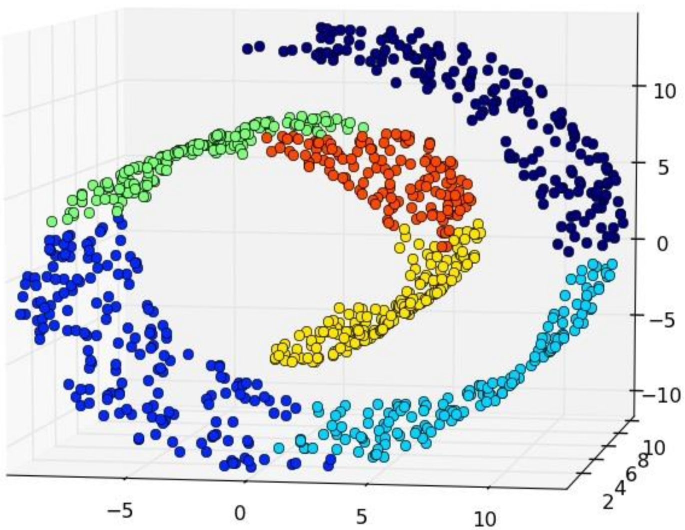
- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

Introduction

- Manifold is a geometric representation of data distribution.

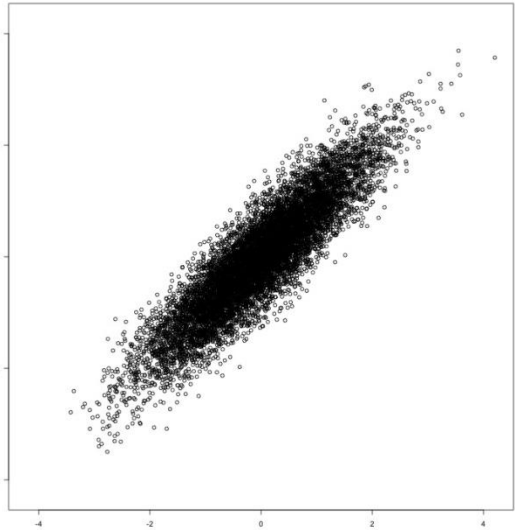


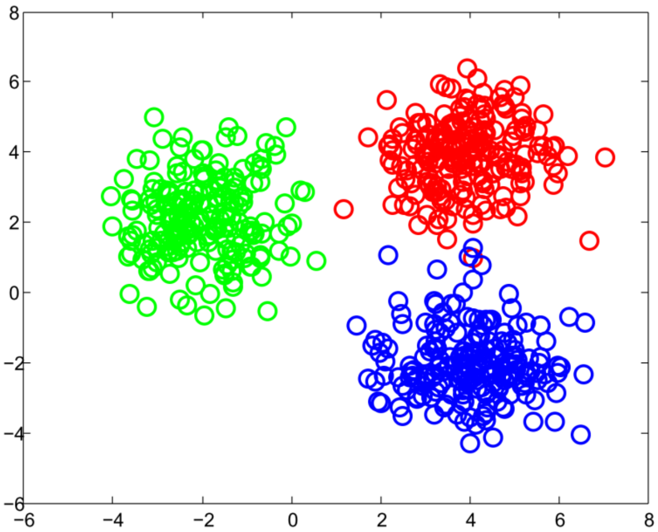


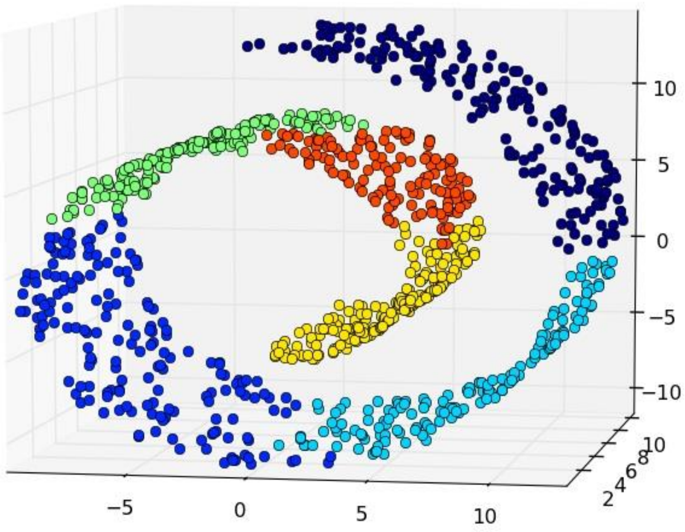


Introduction

- Manifold is a geometric representation of data distribution.



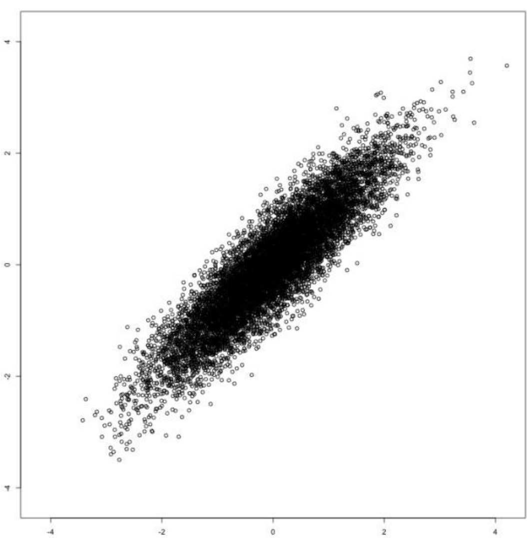


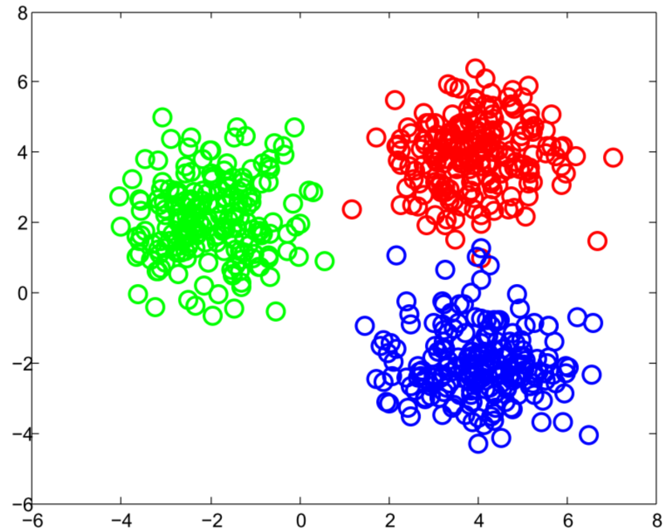


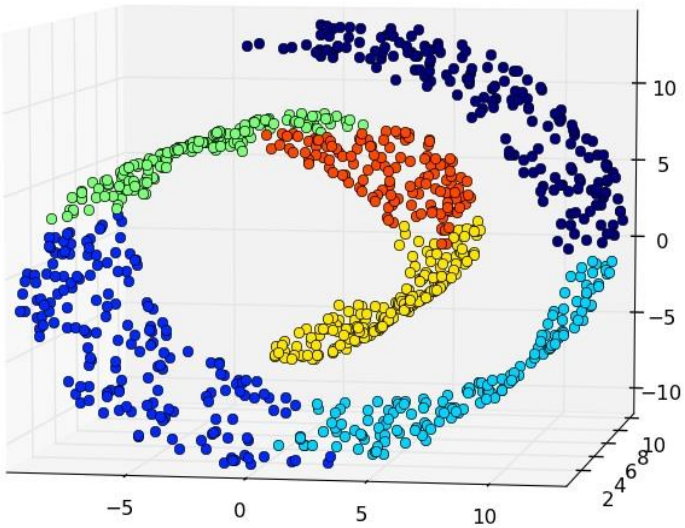
PCA,
Gaussian distribution

Introduction

- Manifold is a geometric representation of data distribution.





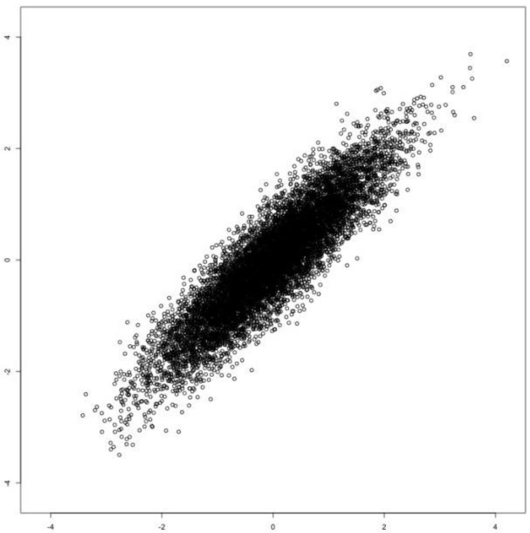


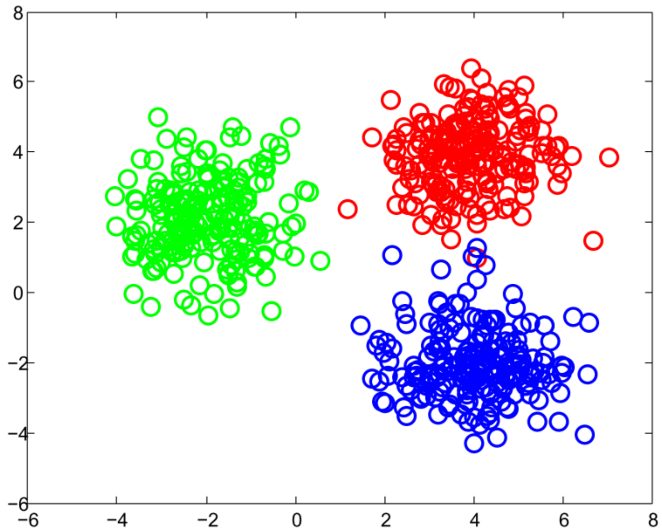
PCA,
Gaussian distribution

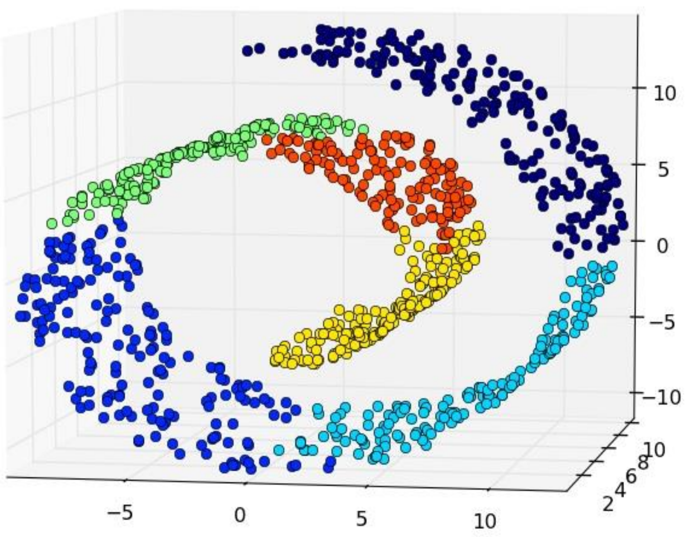
Mixture of Gaussian,
K-means,
Linear SVM

Introduction

- Manifold is a geometric representation of data distribution.







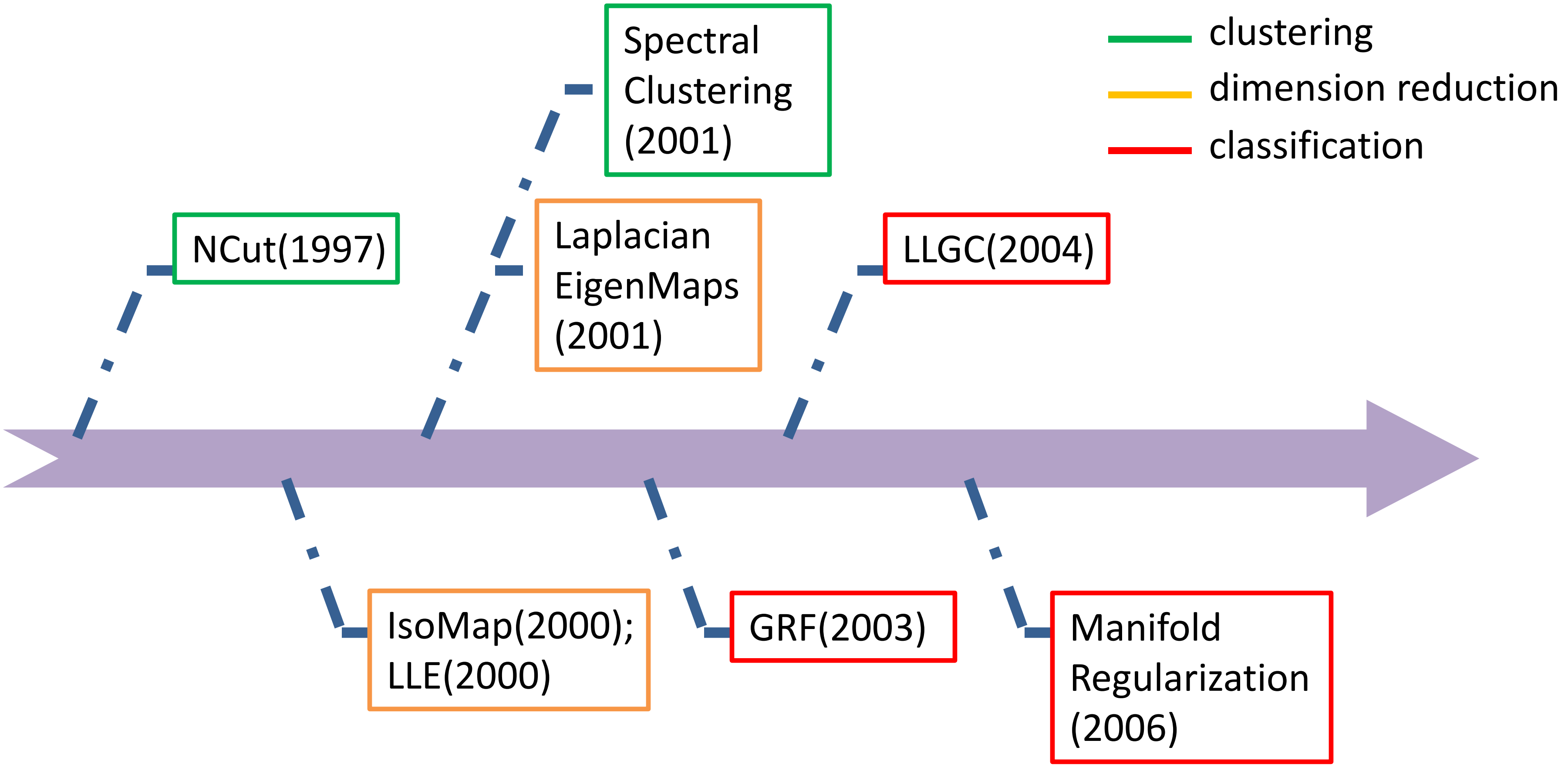
PCA,
Gaussian distribution

Mixture of Gaussian,
K-means,
Linear SVM

Manifold learning
methods

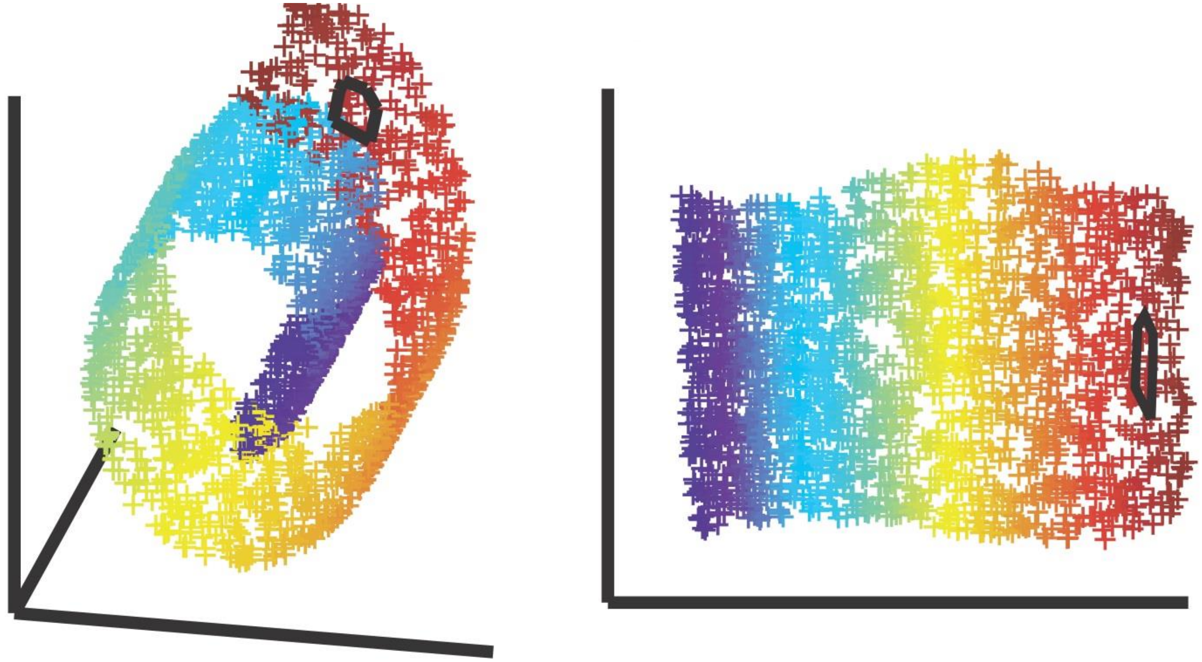
Introduction

- A brief history of manifold learning



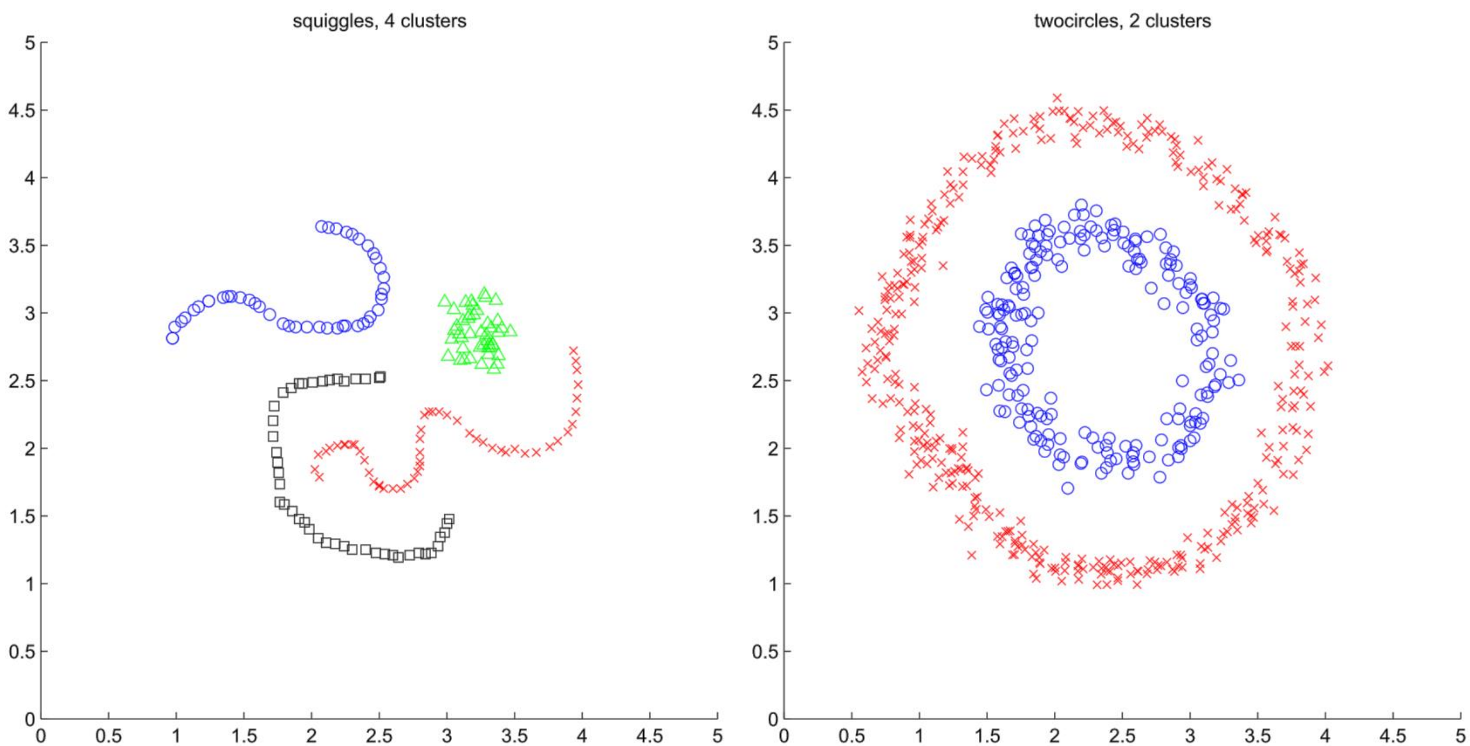
Introduction

- Manifold learning methods
  - Dimension reduction
    - IsoMap [Tenenbaum'00], LLE [Roweis'00], Laplacian Eigenmaps [Belkin'01]



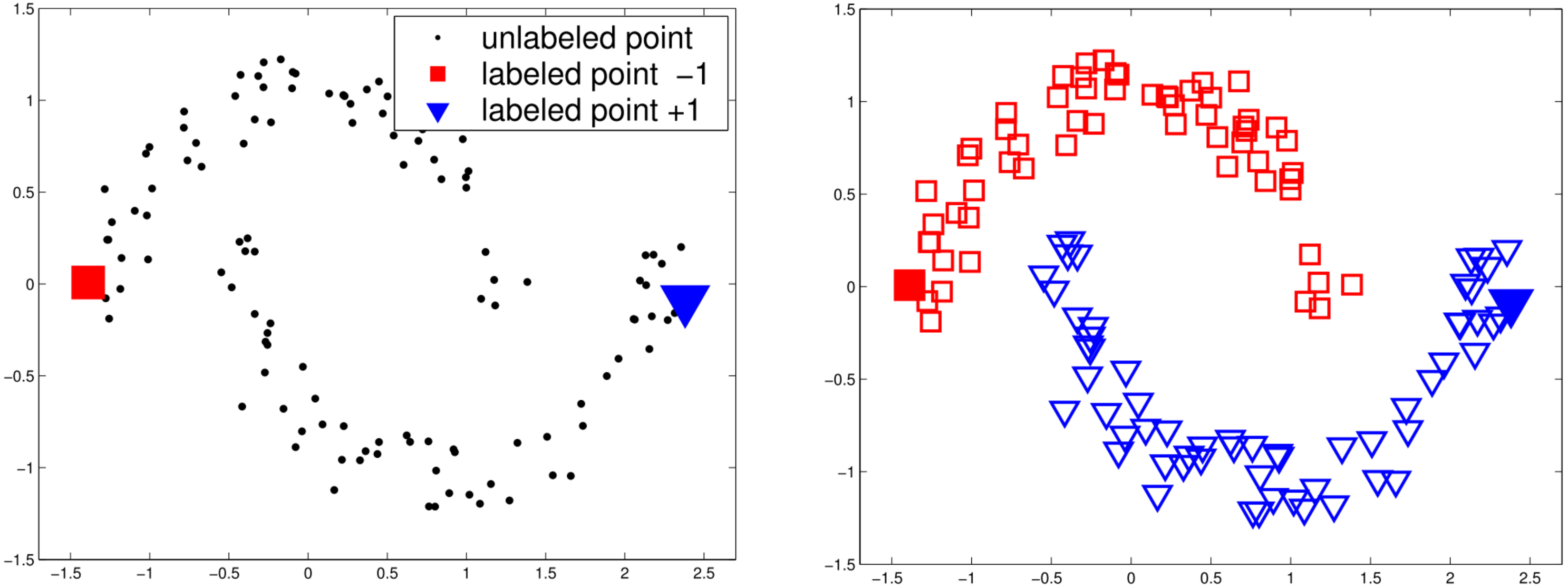
Introduction

- Manifold learning methods
  - Clustering
    - Normalized cut [Shi'97], Spectral clustering [Ng'01]



Introduction

- Manifold learning methods
  - Classification
    - GRF[Zhu'03], LLGC[Zhou'04], Manifold Regularization[Belkin'06]



Manifold Learning Methods for Classification

- Problem definition

Given l labeled data $(x_1, y_1), \ldots, (x_l, y_l)$, and $n - l$ unlabeled data $x_{l+1}, \ldots, x_n$, where $x_i \in \mathbb{R}^d$ $y_i \in \{-1, +1\}$

- Semi-supervised learning:

  learn a decision function $f: \mathbb{R}^d \to \{-1, +1\}$

- Transductive learning:

  predict labels of $x_{l+1}, \ldots, x_n$

Manifold Learning Methods for Classification

- Method
  - Step 1. graph construction

    Approximate the data manifold by a graph

    a. Define the edge set

    k-nearest-neighbor search, ε-nearest-neighbor search

    $O(n^2)$

    b. Weight the edges

    e.g. RBF weighting function $W_{ij} = \exp\{-\frac{\|x_i - x_j\|^2}{\sigma^2}\}$

Manifold Learning Methods for Classification

- Method
  - Step 2. training/optimization

Define $\overline{y} \in \mathbb{R}^n$ as $\overline{y}_i = \begin{cases} y_i & x_i \text{ is labeled} \\ 0 & \text{otherwise} \end{cases}$

$$\min_{f \in R^d} \sum_{i=1}^{n} c_i (f_i - \overline{y}_i)^2 + \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$$

$$\Leftrightarrow \quad \min_{f \in R^d} \underbrace{(f - \overline{y})' C (f - \overline{y})}_{\text{training error}} \; + \; \underbrace{f' L f}_{\text{manifold regularization}}$$

L : graph Laplacian matrix

C : diagonal matrix with $C_{ii} = c_i$

Manifold Learning Methods for Classification

- Method
  - Step 2. training/optimization

Define $\overline{y} \in \mathbb{R}^n$ as $\overline{y}_i = \begin{cases} y_i & x_i \text{ is labeled} \\ 0 & \text{otherwise} \end{cases}$

$$\min_{f \in R^d} \sum_{i=1}^{n} c_i (f_i - \overline{y}_i)^2 + \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$$

$$\Leftrightarrow \quad \min_{f \in R^d} \underbrace{(f - \overline{y})' C (f - \overline{y})}_{\text{training error}} + \underbrace{f' L f}_{\text{manifold regularization}}$$

training error **manifold regularization**

L : graph Laplacian matrix

C : diagonal matrix with $C_{ii} = c_i$

Methods differ from each other in how to define C and L

Manifold Learning Methods for Classification

- Method
  - Step 2. training/optimization cont.

    $(f - \overline{y})' C (f - \overline{y}) \; + \; f' L f$ is quadratic and convex.

    Setting its gradient to zero, we get a closed-form solution:

$$(C + L) f = C \overline{y}$$
$$\Leftrightarrow \quad f = (C + L)^{-1} C \overline{y}$$

Can be solved by the conjugate gradient method.

Manifold Learning Methods for Classification

- Advantages
  - Nonparametric
  - Nonlinear

Manifold Learning Methods for Classification

- Advantages
  - Nonparametric
  - Nonlinear
- Disadvantages
  - Un-scalable
    - Graph construction: kNN search
    - Training algorithm: compute the eigendecomposition or inversion of L

Contents

- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

Manifold embedding Methods

- Method

Step 1. Represent data points in a new space by

$$\min_{U'U=I} \mathrm{tr}(U'LU)$$

The solution U* is the *d* eigenvectors corresponding to the smallest eigenvalues.

M. Belkin, et al. Semi-supervised learning on Riemannian manifolds. Machine Learning, 2004.

Manifold embedding Methods

- Method

Step 1. Represent data points in a new space by

$$\min_{U'U=I} \mathrm{tr}(U'LU)$$

The solution U* is the *d* eigenvectors corresponding to the smallest eigenvalues.

Step 2. Train a linear classifier in the new space.

$$\min_{w} \|y - U_l w\|^2 + \alpha \|w\|^2$$

$$w^* = (U_l' U_l + \alpha I)^{-1} U_l' y$$

in $O(d^3)$ time

M. Belkin, et al. Semi-supervised learning on Riemannian manifolds. Machine Learning, 2004.

Manifold embedding Methods

- How to compute U efficiently ?
  - Nystrom method

k>d

$$L = \begin{array}{c} \overbrace{\phantom{L_{11}}}^{k} \\ k\left\{ \left[\begin{array}{cc} \boxed{\begin{array}{c} L_{11} \\ L_{21} \end{array}} & \begin{array}{c} L_{12} \\ L_{22} \end{array} \end{array} \right] \right. \\ C \end{array}$$

$$L \approx C L_{11}^{-1} C'$$

$$L_{11} = U_k \Sigma_k U_k'$$

$$\Rightarrow U_L = \sqrt{\frac{k}{n}} C U_k \Sigma_k$$

in $O(k^3 + nk^2)$ time

Pick the bottom d eigenvectors as U.

Manifold embedding Methods

- How to compute U efficiently ?
  - Column sampling method

k>d

$$L = \begin{array}{c} \\ k\left\{ \begin{array}{c} \\ \\ \end{array} \right. \\ \\ \end{array} \begin{bmatrix} \overbrace{\boxed{\begin{matrix} L_{11} \\ L_{21} \end{matrix}}}^{k} & \begin{matrix} L_{12} \\ L_{22} \end{matrix} \end{bmatrix}$$

C

$$C = U_d \Sigma_d V_d'$$

in O(nk^2) time

Pick the bottom d eigenvectors as U.

Contents

- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

Prototype-based methods

- Core ideas

  Select a small set of prototypes to build the prediction function and the adjacency graph.

- How to select prototypes $\mathcal{U} = \{\boldsymbol{u}_k\}_{k=1}^{m}$ $m \ll n$
  - random sampling
  - *k*-means clustering
- Reference

O. Delalleau, et al. Efficient non-parametric function induction in semi-supervised learning. AISTATS, 2005.
K. Zhang, et al. Prototype vector machine for large scale semi-supervised learning. ICML, 2009.
W. Liu, et al. Large graph construction for scalable semi-supervised learning. ICML, 2010.

Prototype-based methods

- Predict the labels using prototypes

$$f(\boldsymbol{x}_i) = \sum_{k=1}^{m} Z_{ik} f(\boldsymbol{u}_k)$$

$$\boldsymbol{f} = Z\boldsymbol{a} \qquad \boldsymbol{a} = [f(\boldsymbol{u}_1), \cdots, f(\boldsymbol{u}_m)]^\top$$

Reduce the number of variables from n-*l to m*

W. Liu, et al. Large graph construction for scalable semi-supervised learning. ICML, 2010.

Prototype-based methods

- Predict the labels using prototypes

$$f(\boldsymbol{x}_i) = \sum_{k=1}^{m} Z_{ik} f(\boldsymbol{u}_k)$$

$$\boldsymbol{f} = Z\boldsymbol{a} \qquad \boldsymbol{a} = [f(\boldsymbol{u}_1), \cdots, f(\boldsymbol{u}_m)]^\top$$

Reduce the number of variables from n-*l to m*

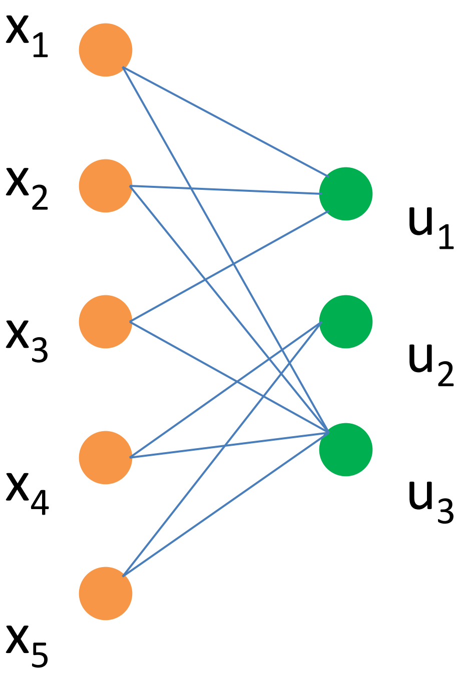
- Design of Z

$$Z_{ik} = \frac{K_h(\boldsymbol{x}_i, \boldsymbol{u}_k)}{\sum_{k' \in \langle i \rangle} K_h(\boldsymbol{x}_i, \boldsymbol{u}_{k'})} \qquad K_h(x_i, u_k) = \exp\{-\frac{\|x_i - u_k\|^2}{2h^2}\}$$

W. Liu, et al. Large graph construction for scalable semi-supervised learning. ICML, 2010.

Prototype-based methods

- Building the anchor graph



$$W = Z\Lambda^{-1}Z' \qquad \Lambda_{kk} = \sum_{i=1}^{n} Z_{ik}$$

$$L = D - W = I - Z\Lambda^{-1}Z'$$

W. Liu, et al. Large graph construction for scalable semi-supervised learning. ICML, 2010.

Prototype-based methods

- Combine the two techniques

Objective function:

$$\min_{f \in R^n} \|f_l - y\|^2 + \gamma f' L f$$

$$\Leftrightarrow \quad \min_{a \in R^m} \|Z_l a - y\|^2 + \gamma a' Z' (I - Z \Lambda^{-1} Z') Z a$$

Set the gradient to 0

$$a^* = (Z_l' Z_l + \gamma (Z' Z - Z' Z \Lambda^{-1} Z' Z))^{-1} Z_l' y$$

Can be compute in $O(m^3 + m^2 n)$ time.

W. Liu, et al. Large graph construction for scalable semi-supervised learning. ICML, 2010.

Prototype-based methods

- Advantages
  - Its complexity scales linearly with *n*
  - No need to construct graphs explicitly.

Prototype-based methods

- Advantages
  - Its complexity scales linearly with n
  - No need to construct graphs explicitly.
- Disadvantages
  - In practice, #prototypes should grow with n
  - The selection of prototypes is very important. But random sampling leads to bad results, while k-means is slow.

Contents

- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

Minimum Tree Cut

- Basic idea
  - Use a spanning tree to approximate the graph
    - ✓ Minimum spanning tree, Shortest path tree, Random spanning tree
  - Use the minimum cut idea to label tree

$$\min_{\mathbf{f}} \boxed{\sum_{(i,j)\in E(T)} w_{ij} I\{f_i \neq f_j\}} \longrightarrow \text{cut size of } \mathbf{f} \text{ on tree T}$$

$$\text{s.t.} \quad \boxed{\begin{array}{l} f_i = y_i \quad i = 1, \ldots, l \\ \mathbf{f} \in \{1, \ldots, K\}^n, \end{array}} \longrightarrow \text{label constraints}$$

Y.-M. Zhang, et al. MTC: a fast and robust graph-based transductive learning method. TNNLS, 2014.

Minimum Tree Cut

- Optimization algorithm

$$\min_{\mathbf{f}} \sum_{(i,j)\in E(T)} w_{ij} I\{f_i \neq f_j\}$$
$$\text{s.t.} \quad f_i = y_i \quad i = 1, \ldots, l$$
$$\mathbf{f} \in \{1, \ldots, K\}^n,$$

  - Step 1. compute the cutsize(i,k) value for each node from leaves to root
  - Step 2. find the labeling to achieve the minimum cut size from root to leaves.

  Each step needs O(n) time.

Y.-M. Zhang, et al. MTC: a fast and robust graph-based transductive learning method. TNNLS, 2014.

Minimum Tree Cut

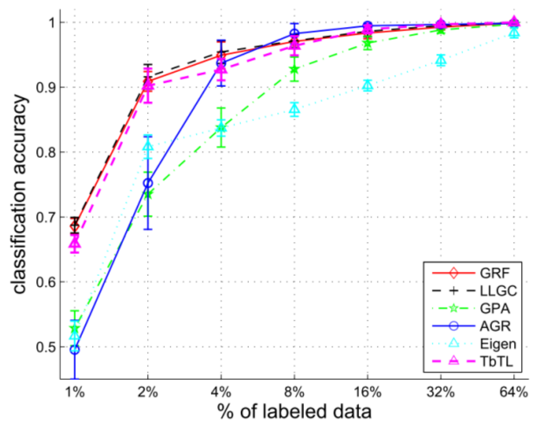
Proposition 1. *Suppose* $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n : \mathbf{x}_i \in \mathbb{R}^d\}$ *is a set of data points.* G *and* G' *are two connected graphs built on* S *by the* ϵ*-graph method with different* ϵ*, and* $E(G)$*,* $E(G')$ *are weighted by* $w_{ij} = \exp\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\}$ *with same* σ*. For each edge* e_{ij}*, let its cost* $\pi_{ij} = -w_{ij}$*. If* T *and* T' *are minimum spanning trees of* G *and* G' *respectively, then* $E(T) = E(T')$.

Proposition 2. *Suppose* G*,* G' *are two connected graphs built on a data set* $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n : \mathbf{x}_i \in \mathbb{R}^d\}$*. Furthermore,* G *and* G' *have the same edge set* $E(G) = E(G')$ *which are weighted by* $w_{ij} = \exp\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\}$ *with different* σ*. For each edge* e_{ij}*, let its cost* $\pi_{ij} = -w_{ij}$*. Then, if* T *and* T' *are minimum spanning trees of* G *and* G' *respectively, then* $E(T) = E(T')$.

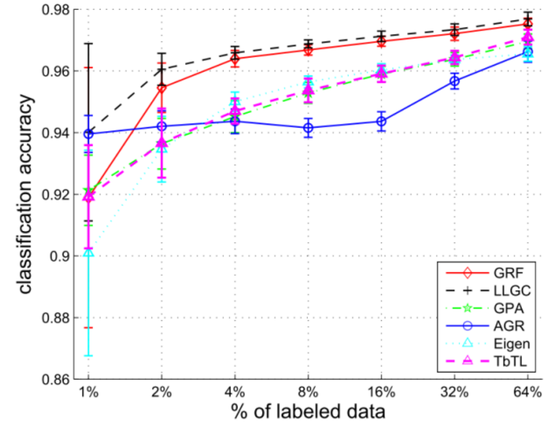
Y.-M. Zhang, et al. MTC: a fast and robust graph-based transductive learning method. TNNLS, 2014.

Minimum Tree Cut

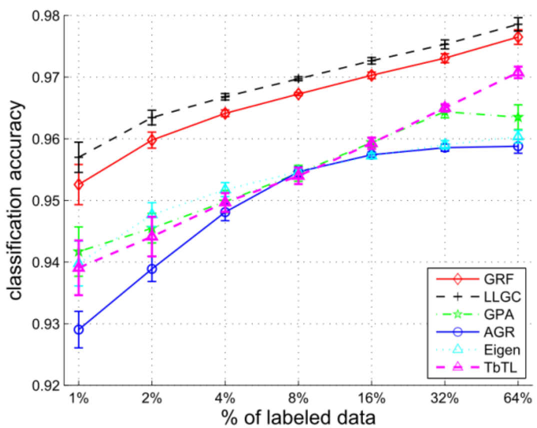
- Experiments : precision



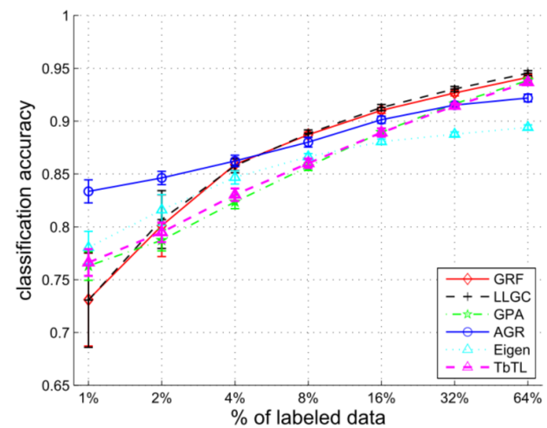
(a) COIL20



(b) USPS



(d) MNIST

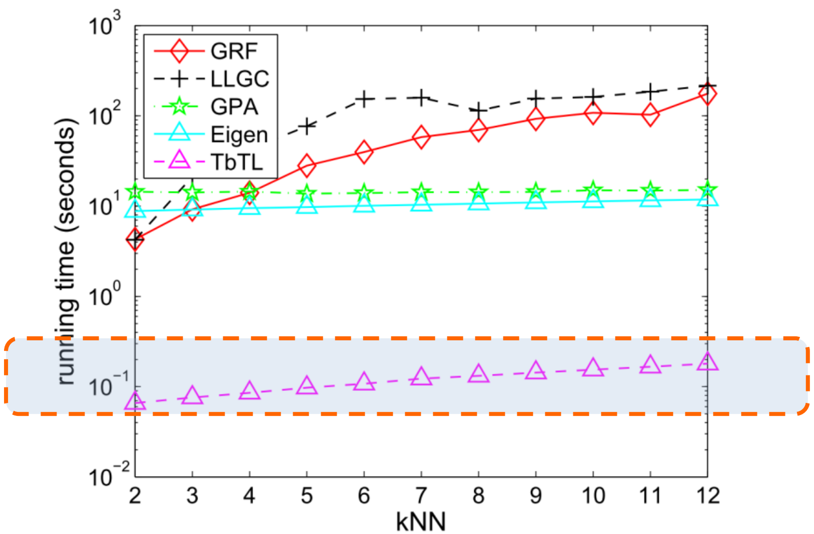


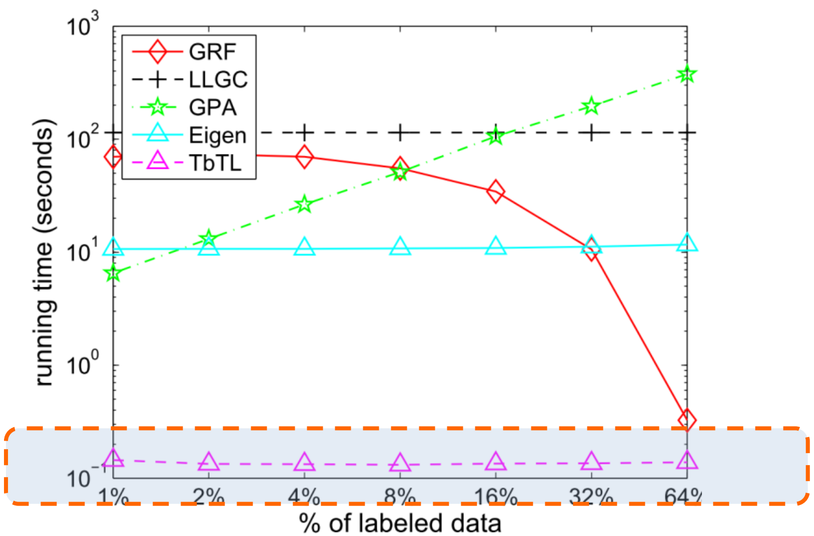
(e) RCV1

Minimum Tree Cut

- Experiments : speed

	USPS	Letter	MNIST	RCV1	News20
GRF	0.16	1.48	73.65	6.39	5.37
LLGC	0.60	0.82	114.26	6.76	13.34
GPA	0.28	6.92	14.29	0.45	0.45
Eigen	1.34	2.93	10.71	2.83	2.79
TbTL	0.01	0.05	0.15	0.03	0.03





Minimum Tree Cut

- Interactive image segmentation
  - 2560*1600=4,096,000 nodes(pixels)
  - 163,465,800 edges

Contents

- An Introduction to Manifold Learning
- Large Scale Manifold Learning
  - Manifold embedding
  - Prototype-based methods
  - Tree-based methods
- Conclusion

Conclusions

- Manifold learning methods provide powerful tools for handling highly nonlinear problems.

- With the help of low-rank approximation and graph sparsification, these methods can enjoy linear complexity.

Thank you!