

APPLICATIONS OF GRAPH LAPLACEANS: Graph Embeddings, and Dimension Reduction

- Graph Embeddings, vertex embeddings . The problem
- Use of Graph Laplaceans, Laplacean Eigenmaps
- Use of similarity graphs: Locally Linear Embeddings
- Explicit dimension reduction method: PCA
- Explicit graph-based dimension reduction method: LLP, ONPP.

Graph embeddings

Vertex embedding: map every vertex x_i to a vector $y_i \in \mathbb{R}^d$

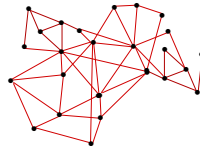
- Trivial use: visualize a graph ($d = 2$)
- Wish: mapping should preserve *similarities* in graph.
- Many applications [clustering, finding missing link, semi-supervised learning, community detection, ...]
- We will see two *nonlinear* classical methods: Eigenmaps, LLE ...
- ... and two linear (explicit) ones.

11-2

- graphEmbed

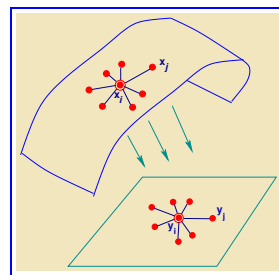
Given: a graph that models some data points x_1, x_2, \dots, x_n
[simplest case: a kNN graph of x_1, x_2, \dots, x_n]

Data: $X = [x_1, x_2, \dots, x_n]$ \rightarrow Graph:



- Graph captures similarities, closeness, ..., in data

Objective: Build a mapping of each vertex i to a data point $y_i \in \mathbb{R}^d$



- Many methods to do this. **Eigenmaps** is one of the best known

- Eigenmaps uses the **graph Laplacean**
- Recall: Graph Laplacean is a matrix defined by :

$$L = D - W$$

$$\begin{cases} w_{ij} \geq 0 & \text{if } j \in \text{Adj}(i) \\ w_{ij} = 0 & \text{else} \end{cases} \quad D = \text{diag} \left[d_{ii} = \sum_{j \neq i} w_{ij} \right]$$

with $\text{Adj}(i)$ = neighborhood of i (excludes i)

- Remember that vertex i represents data item x_i . We will use i or x_i to refer to the vertex.
- We will find the y_i 's by solving an optimization problem.

11-3

- graphEmbed

11-4

- graphEmbed

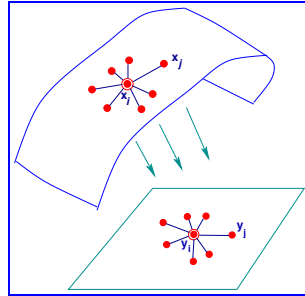
The Laplacean eigenmaps approach

Laplacean Eigenmaps [Belkin-Niyogi '01] *minimizes*

$$\mathcal{F}(Y) = \sum_{i,j=1}^n w_{ij} \|y_i - y_j\|^2 \quad \text{subject to} \quad YDY^\top = I$$

Motivation: if $\|x_i - x_j\|$ is small (orig. data), we want $\|y_i - y_j\|$ to be also small (low-Dim. data)

- Original data used indirectly through its graph
- Objective function can be translated to a trace (see Property 3 in Lecture notes 9) and will yield a sparse eigenvalue problem



11-5

- graphEmbed

- Problem translates to:

$$\begin{cases} \min & \text{Tr} [Y(D - W)Y^\top] \\ Y \in \mathbb{R}^{d \times n} \\ YDY^\top = I \end{cases}$$

- Solution (sort eigenvalues increasingly):

$$(D - W)u_i = \lambda_i D u_i; \quad y_i = u_i^\top; \quad i = 1, \dots, d$$

- An $n \times n$ sparse eigenvalue problem [In 'sample' space]
- Note: can assume $D = I$. Amounts to rescaling data. Problem becomes

$$(I - W)u_i = \lambda_i u_i; \quad y_i = u_i^\top; \quad i = 1, \dots, d$$

11-6

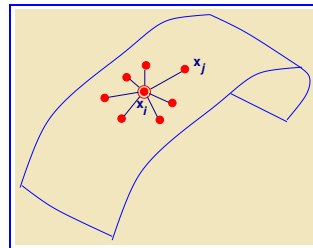
- graphEmbed

Locally Linear Embedding (Roweis-Saul-00)

- LLE is very similar to Eigenmaps. Main differences:
 - 1) Graph Laplacean matrix is replaced by an 'affinity' graph
 - 2) Objective function is changed: want to preserve graph

- 1. **Graph:** Each x_i is written as a convex combination of its k nearest neighbors:
 $x_i \approx \sum w_{ij} x_j, \quad \sum_{j \in N_i} w_{ij} = 1$
 - Optimal weights computed ('local calculation') by minimizing

$$\|x_i - \sum w_{ij} x_j\| \quad \text{for} \quad i = 1, \dots, n$$



11-7

- graphEmbed

2. Mapping:

The y_i 's should obey the same 'affinity' as x_i 's \rightsquigarrow

Minimize:

$$\sum_i \left\| y_i - \sum_j w_{ij} y_j \right\|^2 \quad \text{subject to:} \quad Y \mathbf{1} = 0, \quad YY^\top = I$$

Solution:

$$(I - W^\top)(I - W)u_i = \lambda_i u_i; \quad y_i = u_i^\top$$

- $(I - W^\top)(I - W)$ replaces the graph Laplacean of eigenmaps

11-8

- graphEmbed

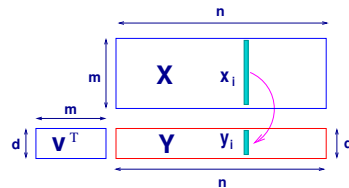
Implicit vs explicit mappings

- Background: Principal Component Analysis (PCA)

Dimension reduction via PCA: We are given a data set $X = [x_1, x_2, \dots, x_n]$, and want a linear mapping from X to Y , expressed as:

$$Y = V^T X \quad \begin{array}{l} X \in \mathbb{R}^{m \times n}; \quad V \in \mathbb{R}^{m \times d} \\ \rightarrow Y \in \mathbb{R}^{d \times n} \end{array}$$

- m -dims. objects (x_i) 'flattened' to d -dims. space (y_i)



- In PCA V is orthogonal ($V^T V = I$)

11-9

- graphEmbed

- In **Principal Component Analysis** $V \in \mathbb{R}^{m \times d}$ is computed to maximize variance of projected data:

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

- Leads to maximizing

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

- Solution $V = \{ \text{dominant eigenvectors} \}$ of the covariance matrix

11-10

- graphEmbed

Explicit (linear) vs. Implicit (nonlinear) mappings:

- In PCA the mapping Φ from high-dimensional space (\mathbb{R}^m) to low-dimensional space (\mathbb{R}^d) is explicitly known:

$$y = \Phi(x) \equiv V^T x$$

- In Eigenmaps and LLE we only know

$$y_i = \phi(x_i), i = 1, \dots, n$$

- Mapping ϕ is now **implicit**: Very difficult to compute $\phi(x)$ for an x that is not in the sample (i.e., not one of the x_i 's)
- Inconvenient for classification. Thus is known as the "The out-of-sample extension" problem

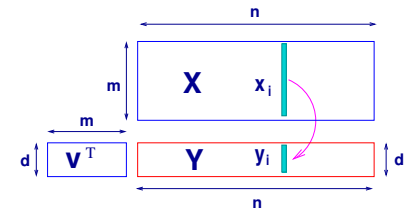
11-11

- graphEmbed

Locally Preserving Projections (He-Niyogi-03)

- LPP is a **linear** dimensionality reduction technique

- Recall the setting:
Want $V \in \mathbb{R}^{m \times d}$; $Y = V^T X$



- Starts with the same neighborhood graph as Eigenmaps: $L \equiv D - W = \text{graph 'Laplacean'}$; with $D \equiv \text{diag}(\{\sum_i w_{ij}\})$.

11-12

- graphEmbed

- Optimization problem is to solve

$$\min_{Y \in \mathbb{R}^{d \times n}, YDY^T = I} \sum_{i,j} w_{ij} \|y_i - y_j\|^2, \quad Y = V^T X.$$

- Difference with eigenmaps: Y is an explicit projection of X
- Solution (sort eigenvalues increasingly)

$$X L X^T v_i = \lambda_i X D X^T v_i \quad y_{i,:} = v_i^T X$$

- Note: essentially same method in [Koren-Carmel'04] called 'weighted PCA' [viewed from the angle of improving PCA]

11-13

- graphEmbed

ONPP (Kokopoulou and YS '05)

- Orthogonal Neighborhood Preserving Projections
- A linear (orthogonal) version of LLE obtained by writing Y in the form $Y = V^T X$
- Same graph as LLE. Objective: preserve the affinity graph (as in LLE) *but* with the constraint $Y = V^T X$
- Problem solved to obtain mapping:

$$\min_V \text{Tr} \left[V^T X (I - W^T) (I - W) X^T V \right]$$

s.t. $V^T V = I$

- In LLE replace $V^T X$ by Y

11-14

- graphEmbed

More recent methods

- Quite a bit of recent work - e.g., methods: node2vec, DeepWalk, GraRep,

See the following papers:

[1] William L. Hamilton, Rex Ying, and Jure Leskovec *Representation Learning on Graphs: Methods and Applications* arXiv:1709.05584v3

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu *GraRep: Learning Graph Representations with Global Structural Information*, CIKM, ACM Conference on Information and Knowledge Management, 24

[3] Amr Ahmed, Nino Shervashidze, and Shravan Narayanamurthy, *Distributed Large-scale Natural Graph Factorization* [Proc. WWW 2013, May 1317, 2013, Rio de Janeiro, Brazil]

... among many others

11-15

- graphEmbed