A brief tour of the spectral problems of data mining

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

Householder XVII, Zeuthen - June 6th, 2008
Team members involved in this work – Support

Past:
- Efi Kokiopoulou [Now at the U. of Lausanne]

Current:
- Jie Chen [grad student]
- Sofia Sakellaridi [grad student]
- Haw-Ren Fang [Post-Doc]

Support:
- National Science Foundation
Common goal of data mining methods: to extract meaningful information or patterns from data. Very broad area – includes: data analysis, machine learning, pattern recognition, information retrieval, ...

- Main tools used: linear algebra; graph theory; approximation theory; optimization; ...
- In this talk: emphasis on dimension reduction techniques and the interrelations between techniques
The problem

Given $d \ll m$ find a mapping

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

- Mapping may be explicit (e.g.,
  $$y = V^T x$$)
- Or implicit (nonlinear)

Practically:

Given $X \in \mathbb{R}^{m \times n}$, we want to find a low-dimensional representation $Y \in \mathbb{R}^{d \times n}$ of $X$

- Two classes of methods: (1) projection techniques and (2) non-linear implicit methods.
Example 1: The ‘Swill-Roll’ (2000 points in 3-D)
Example 2: Digit images (a sample of 20)
**Projection-based Dimensionality Reduction**

*Given:* a data set \( X = [x_1, x_2, \ldots, x_n] \), and \( d \) the dimension of the desired reduced space \( Y \).

*Want:* a linear transformation from \( X \) to \( Y \)

\[
X \in \mathbb{R}^{m \times n} \\
V \in \mathbb{R}^{m \times d} \\
Y = V^\top X \\
\rightarrow Y \in \mathbb{R}^{d \times n}
\]

- \( m \)-dimens. objects \((x_i)\) ‘flattened’ to \( d \)-dimens. space \((y_i)\)

*Constraint:* The \( y_i \)’s must satisfy certain properties

- Optimization problem
In PCA projected data must have maximum variance, i.e., we need to maximize over all orthogonal \( m \times d \) matrices \( V \):

\[
\sum_i \| y_i - \frac{1}{n} \sum_j y_j \|_2^2 = \cdots = \text{Tr} \left[ V^\top \bar{X} \bar{X}^\top V \right]
\]

Where: \( \bar{X} = X (I - \frac{1}{n} 11^T) \) == origin-recentered version of \( X \)

Solution \( V = \{ \text{dominant eigenvectors} \} \) of the covariance matrix

== Set of left singular vectors of \( \bar{X} \)

Solution \( V \) also minimizes ‘reconstruction error’ ..

\[
\sum_i \| x_i - VV^T x_i \|_2^2 = \sum_i \| x_i - V y_i \|_2^2
\]

.. and it also maximizes [Korel and Carmel 04] \( \sum_{i,j} \| y_i - y_j \|_2^2 \)
Laplacean Eigenmaps (Belkin-Niyogi-02)

- Not a linear (projection) method but a **Nonlinear method**
- Starts with k-nearest-neighbors graph

- Defines the graph Laplacean $L = D - W$. Simplest:

  $$D = \text{diag}(\text{deg}(i)); \quad w_{ij} = \begin{cases} 1 & \text{if } j \in N_i \\ 0 & \text{else} \end{cases}$$

  with $N_i = \text{neighborhood of } i$ (excl. $i$); $\text{deg}(i) = |N_i|$
A few properties of graph Laplacean matrices

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

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

\textbf{Property 1:} for any $x \in \mathbb{R}^n$:

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

\textbf{Property 2:} (generalization) for any $Y \in \mathbb{R}^{d \times n}$:

$$\text{Tr}[YLY^\top] = \frac{1}{2} \sum_{i,j} w_{ij} \|y_i - y_j\|^2$$
Property 3: For the particular \( L = I - \frac{1}{n} \mathbf{1} \mathbf{1}^\top \)

\[
XLX^\top = \bar{X} \bar{X}^\top = n \times \text{Covariance matrix}
\]

[Proof: 1) \( L \) is a projector: \( L^\top L = L^2 = L \), and 2) \( XL = \bar{X} \)]

- Consequence-1: PCA equivalent to maximizing \( \sum_{ij} \|y_i - y_j\|^2 \)
- Consequence-2: what about replacing trivial \( L \) with something else? [viewpoint in Koren-Carmel’04]
Property 4: (Graph partitioning) If $x$ is a vector of signs ($\pm 1$) then

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

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

Consequence: Can be used for partitioning graphs, or ‘clustering’

[take $p = \text{sign}(u_2)$, where $u_2 = 2\text{nd smallest eigenvector..}$]
Laplacean Eigenmaps *minimizes*

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

Notes:
1. Motivation: if \( \| x_i - x_j \| \) is small (orig. data), we want \( \| y_i - y_j \| \) to be also small (low-D data)
2. Note Min instead of Max as in PCA [counter-intuitive]
3. Above problem uses original data indirectly through its graph
Problem translates to:

$$\min \quad \text{Tr} \left[ Y (D - W) Y^\top \right] .$$

$$\begin{cases} 
\quad Y \in \mathbb{R}^{d \times n} \\
\quad Y D Y^\top = I 
\end{cases}$$

Solution (sort eigenvalues increasingly):

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

Note: an $n \times n$ sparse eigenvalue problem [In ‘sample’ space]

Note: can assume $D = I$. Amounts to rescaling data. Problem becomes

$$(I - W) u_i = \lambda_i u_i ; \quad y_i = u_i^\top ; \quad i = 1, \ldots, d$$
Why smallest eigenvalues vs largest for PCA?

Intuition:
Graph Laplacean and ‘unit’ Laplacean are very different: one involves a sparse graph (More like a discr. differential operator). The other involves a dense graph. (More like a discr. integral operator). They should be treated as the inverses of each other.

Viewpoint confirmed by what we learn from Kernel approach
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
\[
\| x_i - \sum w_{ij} x_j \| \quad \text{for} \quad i = 1, \ldots, n
\]
2. Mapping:

The $y_i$’s should obey the same ’affinity’ as $x_i$’s

Minimize:

$$\sum_i \left\| y_i - \sum_j w_{ij} y_j \right\|^2$$

subject to: $Y1 = 0, \quad YY^\top = I$

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.
Locally Preserving Projections (He-Niyogi-03)

- LPP is a linear dimensionality reduction technique

- Recall the setting:
  \[ V \in \mathbb{R}^{m \times d}; \quad Y = V^\top X \]

- Starts with the same neighborhood graph as Eigenmaps: \( L \equiv D - W = \text{graph ‘Laplacean’}; \) with \( D \equiv \text{diag}(\{\Sigma_i w_{ij}\}) \).
Optimization problem is to solve

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

Difference with eigenmaps: $Y$ is a projection of $X$ data

Solution (sort eigenvalues increasingly)

$$XLX^\top v_i = \lambda_i XDX^\top v_i \quad y_i = v_i^\top X$$

Note: essentially same method in [Koren-Carmel’04] called ‘weighted PCA’ [viewed from the angle of improving PCA]
ONPP (Kokiopoulou and YS ’05)

- Orthogonal Neighborhood Preserving Projections
- Can be viewed as a linear version of LLE
- Uses the same graph as LLE. Objective: preserve the affinity graph (as in LEE) *but* by means of an orthogonal projection
- Objective function

\[ \Phi(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|^2 \quad \text{Constraint: } Y = V^T X, \quad V^T V = I \]

- Notice that

\[ \Phi(Y) = \|Y - Y W^T\|_F^2 = \cdots = \text{Tr} \left[ V^T X (I - W^T)(I - W) X^T V \right] \]
Resulting problem:

\[
\min_{V \in \mathbb{R}^{m \times d}; \quad V^\top V = I} \text{Tr} \begin{bmatrix}
V^\top X (I - W^\top)(I - W) X^\top M V
\end{bmatrix}
\]

Solution: Columns of \( V = \) eigenvectors of \( M \) associated with smallest \( d \) eigenvalues

\( \Rightarrow \) Can be computed as \( d \) lowest left singular vectors of 

\[
X (I - W^\top)
\]
## A unified view

<table>
<thead>
<tr>
<th>Method</th>
<th>Object. (min)</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA/MDS</td>
<td>$\text{Tr} \left[ V^\top X \left(-I + ee^\top\right) X^\top V \right]$</td>
<td>$V^\top V = I$</td>
</tr>
<tr>
<td>LLE</td>
<td>$\text{Tr} \left[ Y(I - W^\top)(I - W)Y^\top \right]$</td>
<td>$YY^\top = I$</td>
</tr>
<tr>
<td>Eigenmaps</td>
<td>$\text{Tr} \left[ Y(I - W)Y^\top \right]$</td>
<td>$YY^\top = I$</td>
</tr>
<tr>
<td>LPP</td>
<td>$\text{Tr} \left[ V^\top X(I - W)X^\top V \right]$</td>
<td>$V^\top XX^\top V = I$</td>
</tr>
<tr>
<td>ONPP</td>
<td>$\text{Tr} \left[ V^\top X(I - W^\top)(I - W)X^\top V \right]$</td>
<td>$V^\top V = I$</td>
</tr>
<tr>
<td>LDA</td>
<td>$\text{Tr} \left[ V^\top X(I - H)X^\top V \right]$</td>
<td>$V^\top XX^\top V = I$</td>
</tr>
</tbody>
</table>
Let $M = I - W$ = a Laplacean matrix ($-I + ee^\top$ for PCA/MDS); or the LLE matrix $(I - W)(I - W^\top)$, or geodesic distance matrix (ISOMAP).

All techniques lead to one of two types of problems

First type is:

$$\min_{Y \in \mathbb{R}^{d \times n}} \quad \text{Tr} \left[ Y M Y^\top \right]$$

$$Y Y^\top = I$$

$Y$ obtained from solving an eigenvalue problem

LLE, Eigenmaps (normalized), ..
And the second type is:

\[
\begin{align*}
\min_{V \in \mathbb{R}^{m \times d}} & \quad \text{Tr} [V^\top X M X^\top V] \\
\text{subject to} & \quad V^\top G V = I
\end{align*}
\]

- \(G\) is either the identity matrix or \(XDX^\top\) or \(XX^\top\).
- Low-Dim. data: \(Y = V^\top X\)

**Important observation:** 2nd is just a projected version of the 1st, i.e., approximate eigenvectors are sought in Span \(\{X\}\) [Rayleigh-Ritz procedure]

- Problem is of dim. \(m\) (dim. of data) not \(n\) (# of samples).
- This difference can be mitigated by resorting to Kernels..
TIME FOR A MATLAB DEMO
A brief tour of Kernels

- Kernels employ an implicit nonlinear map of original data into a higher dimensional feature space $\mathbb{H}$.

- Mapping $\Phi$ only known through its Kernel on data:

  $$ < \phi(x_i), \phi(x_j) > \equiv K(x_i, x_j) $$

- Can do PCA, eigenmaps, ..., on this data without using $\Phi$
Kernel PCA (Ham et. al. 2004)

- Classical PCA on the set \( \{ \Phi \} \)

\[
\min \text{Tr} \left[ V^\top \Phi \Phi^\top V \right] \quad \text{subject to} \quad V^\top V = I
\]

- Projected data \( Y = V^\top \Phi \)

- Problem to solve \( \Phi \Phi^\top u_i = \lambda u_i \)
Right singular vector approach. Multiply both sides by $\phi^T$:

$$
\begin{bmatrix}
\Phi^T \\
\phi
\end{bmatrix}
\Phi^T u_i = \lambda_i \Phi^T u_i
$$

Note

1. $\Phi^T \Phi = (I - ee^T) K (I - ee^T)$ Denoted by $\bar{K}$
2. $\Phi^T u_i = y_i^T$ (recall $Y = V^T \phi$)

Result: columns of $Y^T$ are largest eigenvectors of $\bar{K}$

$$
\bar{K} y_i^T = \lambda_i y_i^T \quad \text{or} \quad y_i \bar{K} = \lambda_i y_i
$$
Compare with Eigenmaps: the columns of $Y^\top$ ($n$-vectors) are smallest eigenvectors of $L = I - W$

Interpretation [see Ham, Mika, and Scölkopf, 2004]: Eigenmaps can be interpreted as Kernel PCA with Kernel $K = L^\dagger$. 
Proceed similarly to PCA.

Assumption & notation: \( \Phi \equiv \Phi(X) \), \( K \equiv \Phi^\top \Phi \) is invertible

**LPP:** Problem in feature space:

\[
\begin{align*}
\min & \text{Tr} \left[ V^\top \Phi(X) L \Phi(X)^\top V \right] \\
\text{Subj. to} & \quad V^\top \Phi D \Phi^\top V = I
\end{align*}
\]

Leads to the eigenvalue problem:

\[
\Phi L \Phi^\top u_i = \lambda_i \Phi D \Phi^\top u_i
\]

Left multiply by \( \Phi^\top \), then by \( K^{-1} \), + recall that \( y_i^\top = \Phi^\top u_i \):

\[
Ly_i^\top = \lambda_i D y_i^\top
\]

Note: \( K \) disappeared from picture; What’s the catch??.
\[ \min_{V \in \mathbb{R}^{L \times d}} V^T V = I \quad \text{Tr} \left[ V^T \Phi(X) M \Phi(X)^T V \right] \]

- Leads to the eigenvalue problem:

\[ \Phi M \Phi^T u_i = \lambda_i u_i \]

- Multiply by \( \Phi^T \) and note as before \( K = \Phi^T \Phi \), \( y_i^T = \Phi^T u_i \):

\[ K M y_i^T = \lambda_i y_i^T \quad \text{or} \quad M y_i^T = K^{-1} y_i^T \]

- Solution is set of eigenvectors of Matrix \( M \) – but constraint: \( K^{-1} \) - orthogonality
Conclusion

➢ So how is this related to initial title of “efficient algorithms in data mining”?

➢ Answer: All these eigenvalue problems are not cheap to solve..

➢ .. and cost issue does not seem to bother practitioners too much for now..

➢ Ingredients that will become mandatory:

1. Avoid the SVD
2. Fast algorithms that do not sacrifice quality.
3. In particular: Multilevel approaches
4. Multilinear algebra [tensors]
Multilevel techniques in brief

➢ Divide and conquer paradigms as well as multilevel methods in the sense of ‘domain decomposition’

➢ Main principle: very costly to do an SVD [or Lanczos] on the whole set. Why not find a smaller set on which to do the analysis – without too much loss?

➢ Tools used: graph coarsening, divide and conquer –

➢ For information retrieval we use hypergraphs
**Main Idea:** coarsen for a few levels. Use the resulting data set $\hat{X}$ to find a projector $P$ from $\mathbb{R}^m$ to $\mathbb{R}^d$. $P$ can be used to project original data or new data.

- **Gain:** Dimension reduction is done with a much smaller set.
- **Hope:** not much loss compared to using whole data.
Recall common approach:

1. Scale data [e.g. TF-IDF scaling]:
2. Perform a (partial) SVD on resulting matrix $X \approx U_d\Sigma_d V_d^T$
3. Process query by same scaling (e.g. TF-IDF)
4. Compute similarities in $d$-dimensional space:
   $$s_i = \langle \hat{q}, \hat{x}_i \rangle / \|\hat{q}\| \|\hat{x}_i\|$$
   where $[\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n] = V_d^T \in \mathbb{R}^{d \times n}$; $\hat{q} = \Sigma_d^{-1} U_d^T \bar{q} \in \mathbb{R}^d$

Multilevel approach: replace SVD (or any other dim. reduction) by dimension reduction on coarse set. Only difference: TF-IDF done on the coarse set not original set.
Three public data sets used for experiments: Medline, Cran and NPL (cs.cornell.edu)

- Coarsening to a max. of 4 levels.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Medline</th>
<th>Cran</th>
<th>NPL</th>
</tr>
</thead>
<tbody>
<tr>
<td># documents</td>
<td>1033</td>
<td>1398</td>
<td>11429</td>
</tr>
<tr>
<td># terms</td>
<td>7014</td>
<td>3763</td>
<td>7491</td>
</tr>
<tr>
<td>sparsity (%)</td>
<td>0.74%</td>
<td>1.41%</td>
<td>0.27%</td>
</tr>
<tr>
<td># queries</td>
<td>30</td>
<td>225</td>
<td>93</td>
</tr>
<tr>
<td>avg. # rel./query</td>
<td>23.2</td>
<td>8.2</td>
<td>22.4</td>
</tr>
</tbody>
</table>
### Results with NPL

#### Statistics

<table>
<thead>
<tr>
<th>Level</th>
<th>coarsen.</th>
<th>#doc.</th>
<th>#dim.</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>N/A</td>
<td>11429</td>
<td>736</td>
<td>23.5%</td>
</tr>
<tr>
<td>#2</td>
<td>3.68</td>
<td>5717</td>
<td>592</td>
<td>23.8%</td>
</tr>
<tr>
<td>#3</td>
<td>2.19</td>
<td>2861</td>
<td>516</td>
<td>23.9%</td>
</tr>
<tr>
<td>#4</td>
<td>1.50</td>
<td>1434</td>
<td>533</td>
<td>23.3%</td>
</tr>
</tbody>
</table>
CPU times for preprocessing (Dim. reduction part)