Dimension Reduction
Dimension Reduction
Data is Too Big To Do Something..

- Limited memory size
  - Data may not be fitted to the memory of your machine

- Slow computation
  - $10^6$-dim vs. 10-dim vectors for Euclidean distance computation
Two Axes of Data Set

- **No. of data items**
  - How many data items?

- **No. of dimensions**
  - How many dimensions representing each item?

Diagram:

```
Data item index

| Dimension index |
```

- Data item index
- Dimension index
Two Axes of Data Set

- **No. of data items**
  - How many data items?

- **No. of dimensions**
  - How many dimensions representing each item?
Two Axes of Data Set

- **No. of data items**
  - How many data items?

- **No. of dimensions**
  - How many dimensions representing each item?
Two Axes of Data Set

- No. of data items
  - How many data items?

- No. of dimensions
  - How many dimensions representing each item?
Two Axes of Data Set

No. of data items
- How many data items?

No. of dimensions
- How many dimensions representing each item?

We will use this during lecture
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)

High-dim data → Dimension Reduction → low-dim data

No. of : user-specified
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)

High-dim data → Dimension Reduction → low-dim data

No. of

Other

: user-specified
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)

- High-dim data
- No. of Additional info about data
- Dimension Reduction
- low-dim data
- Other: user-specified
Dimension Reduction
Let’s Reduce Data (along Dimension Axis)

High-dim data

No. of

Dimension Reduction

low-dim data

Additional info about data

Other

Dim-reducing Transformer for a new data

: user-specified
What You Get from DR

Obviously,

- Less storage
- Faster computation
What You Get from DR

Obviously,

- Less storage
- Faster computation

More importantly,
What You Get from DR

Obviously,

- Less storage
- Faster computation

More importantly,

- Noise removal (improving quality of data)
  - Leads better performance for tasks
What You Get from DR

Obviously,
- Less storage
- Faster computation

More importantly,
- Noise removal (improving quality of data)
  - Leads better performance for tasks
- 2D/3D representation
  - Enables visual data exploration
Applications

Traditionally,
- Microarray data analysis
- Information retrieval
- Face recognition
- Protein disorder prediction
- Network intrusion detection
- Document categorization
- Speech recognition

More interestingly,
- Interactive visualization of high-dimensional data
Face Recognition
Vector Representation of Images

Images → serialized/rasterized pixel values

Dimensions can be huge.
- 640x480 size: 307,200 dimensions
Face Recognition
Face Recognition

Vector Representation of Images

Color = Person
Column = Image
Face Recognition

Vector Representation of Images

Color = Person
Column = Image
Face Recognition

Vector Representation of Images

Color = Person
Column = Image
Face Recognition

Vector Representation of Images

Dimension Reduction

PCA, LDA, etc.

Color = Person
Column = Image
Face Recognition

Vector Representation of Images

Dimension Reduction

Classification on dim-reduced data

→ Better accuracy than on original-dim data
Document Retrieval

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Document Retrieval

Latent semantic indexing

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Document Retrieval

Latent semantic indexing

Term-document matrix via bag-of-words model

- D1 = “I like data”
- D2 = “I hate hate data”

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>…</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Latent semantic indexing

Term-document matrix via bag-of-words model
- D1 = “I like data”
- D2 = “I hate hate data”

Dimensions can be hundreds of thousands
- i.e., #distinct words

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>…</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
**Document Retrieval**

Latent semantic indexing

- **Term-document matrix via bag-of-words model**
  - D1 = “I like data”
  - D2 = “I hate hate data”

- **Dimensions can be hundreds of thousands**
  - i.e., #distinct words

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Dimension Reduction**

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim1</td>
<td>1.75</td>
<td>-0.27</td>
<td>...</td>
</tr>
<tr>
<td>Dim2</td>
<td>-0.21</td>
<td>0.58</td>
<td>...</td>
</tr>
<tr>
<td>Dim3</td>
<td>1.32</td>
<td>0.25</td>
<td></td>
</tr>
</tbody>
</table>
Document Retrieval

Latent semantic indexing

Term-document matrix via bag-of-words model
- D1 = “I like data”
- D2 = “I hate hate data”

Dimensions can be hundreds of thousands
- i.e., #distinct words

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>like</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>hate</td>
<td>0</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Dimension Reduction

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim1</td>
<td>1.75</td>
<td>-0.27</td>
<td>...</td>
</tr>
<tr>
<td>Dim2</td>
<td>-0.21</td>
<td>0.58</td>
<td>...</td>
</tr>
<tr>
<td>Dim3</td>
<td>1.32</td>
<td>0.25</td>
<td></td>
</tr>
</tbody>
</table>

→ Search-Retrieval on dim-reduced data leads to better semantics
Visualizing “Map of Science”

http://www.mapofscience.com
Two Main Techniques

1. Feature selection
   - Selects a subset of the original variables as reduced dimensions
   - For example, the number of genes responsible for a particular disease may be small

2. Feature extraction
   - Each reduced dimension involves multiple original dimensions
   - Active area of research recently

Note that Feature = Variable = Dimension
Feature Selection

What are the optimal subset of $m$ features to maximize a given criterion?

- **Widely-used criteria**
  - Information gain, correlation, …

- **Typically combinatorial optimization problems**

- Therefore, greedy methods are popular
  - **Forward selection**: Empty set $\rightarrow$ add one variable at a time
  - **Backward elimination**: Entire set $\rightarrow$ remove one variable at a time
From now on, we will only discuss about feature extraction
From now on, we will only discuss about feature extraction
Aspects of DR

- Linear vs. Nonlinear
- Unsupervised vs. Supervised
- Global vs. Local
- Feature vectors vs. Similarity (as an input)
Aspects of DR
Linear vs. Nonlinear

Linear

- Represents each reduced dimension as a linear combination of original dimensions
  - e.g., \( Y_1 = 3X_1 - 4X_2 + 0.3X_3 - 1.5X_4 \),
  \( Y_2 = 2X_1 + 3.2X_2 - X_3 + 2X_4 \)

- Naturally capable of mapping new data to the same space

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>X4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

```
\begin{center}
\begin{tabular}{c|c|c|c}
<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>1.75</td>
<td>-0.27</td>
</tr>
<tr>
<td>Y2</td>
<td>-0.21</td>
<td>0.58</td>
</tr>
</tbody>
</table>
\end{tabular}
\end{center}
```
Aspects of DR
Linear vs. Nonlinear

Linear
► Represents each reduced dimension as a linear combination of original dimensions
  ◦ e.g., \( Y_1 = 3X_1 - 4X_2 + 0.3X_3 - 1.5X_4 \),
    \( Y_2 = 2X_1 + 3.2X_2 - X_3 + 2X_4 \)
► Naturally capable of mapping new data to the same space

Nonlinear
► More complicated, but generally more powerful
► Recently popular topics
Aspects of DR
Unsupervised vs. Supervised

Unsupervised
▶ Uses only the input data

High-dim data

Dimension Reduction

No. of

low-dim data

Additional info about data

Other

Dim-reducing Transformer for a new data
Aspects of DR
Unsupervised vs. Supervised

Supervised
- Uses the input data + additional info

High-dim data

No. of

Dimension Reduction

low-dim data

Additional info about data

Other

Dim-reducing Transformer for a new data
Aspects of DR
Unsupervised vs. Supervised

Supervised
- Uses the input data + additional info
  - e.g., grouping label

High-dim data

Dimension Reduction

No. of

low-dim data

Additional info about data

Other

Dim-reducing Transformer for a new data
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

Information loss is unavoidable!

- e.g., PCA
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

Information loss is unavoidable!

- e.g., PCA
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

Information loss is unavoidable!

Then, what would you care about?

Global
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

Information loss is unavoidable!

Then, what would you care about?

Global

Treats all pairwise distances equally important
- Tends to care larger distances more
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

- Information loss is unavoidable!

Then, what would you care about?

Global
- Treats all pairwise distances equally important
  - Tends to care larger distances more

Local
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

- Information loss is unavoidable!

Then, what would you care about?

Global
- Treats all pairwise distances equally important
  - Tends to care larger distances more

Local
- Focuses on small distances, neighborhood relationships
Aspects of DR
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

- Information loss is unavoidable!

Then, what would you care about?

Global
- Treats all pairwise distances equally important
  - Tends to care larger distances more

Local
- Focuses on small distances, neighborhood relationships
- Active research area a.k.a. manifold learning
Aspects of DR
Feature vectors vs. Similarity (as an input)

Typical setup (feature vectors as an input)
Aspects of DR
Feature vectors vs. Similarity (as an input)

- Typical setup (feature vectors as an input)
- Some methods take similarity matrix instead
  - $(i,j)$-th component indicates how similar $i$-th and $j$-th data items are

Diagram:
- Similarity matrix
- No. of
- Dimension Reduction
- low-dim data
- Dim-reducing Transformer for a new data
- Additional info about data
- Other
Aspects of DR
Feature vectors vs. Similarity (as an input)

Similarity matrix

No. of

Dimension Reduction

low-dim data

Additional info about data

Other

Dim-reducing Transformer for a new data
Aspects of DR
Feature vectors vs. Similarity (as an input)

Typical setup (feature vectors as an input)
Some methods take similarity matrix instead
Some methods internally converts feature vectors to similarity matrix before performing dimension reduction
Aspects of DR
Feature vectors vs. Similarity (as an input)

- Typical setup (feature vectors as an input)
- Some methods take similarity matrix instead
- Some methods internally convert feature vectors to similarity matrix before performing dimension reduction

Diagram:
- High-dim data
- Similarity matrix
- Dimension Reduction
- low-dim data

a.k.a. Graph Embedding
Aspects of DR
Feature vectors vs. Similarity (as an input)

Why called graph embedding?

Similarity matrix can be viewed as a **graph** where similarity represents edge weight

---

High-dim data → Similarity matrix → low-dim data

Dimension Reduction

a.k.a. **Graph Embedding**
Methods

Traditional

- Principal component analysis (PCA)
- Multidimensional scaling (MDS)
- Linear discriminant analysis (LDA)
- Nonnegative matrix factorization (NMF)

Advanced (nonlinear, kernel, manifold learning)

- Isometric feature mapping (Isomap)
- Locally linear embedding (LLE)
- Laplacian Eigenmaps (LE)
- Kernel PCA
- t-distributed stochastic neighborhood embedding (t-SNE)

* Matlab codes are available at http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html
Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: eigen-decomposition
- Pros: Fast
- Cons: basic limited performances

http://en.wikipedia.org/wiki/Principal_component_analysis
Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: eigen-decomposition
- Pros: Fast
- Cons: basic limited performances

Linear

http://en.wikipedia.org/wiki/Principal_component_analysis
Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: eigen-decomposition
- Pros: Fast
- Cons: basic limited performances

Linear
Unsupervised

http://en.wikipedia.org/wiki/Principal_component_analysis
Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: eigen-decomposition
- Pros: Fast
- Cons: basic limited performances

http://en.wikipedia.org/wiki/Principal_component_analysis
Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: eigen-decomposition
- Pros: Fast
- Cons: basic limited performances

Linear
Unsupervised
Global
Feature vectors

http://en.wikipedia.org/wiki/Principal_component_analysis
Principal Component Analysis

Document Visualization
Principal Component Analysis
Testbed Demo – Text Data
Multidimensional Scaling (MDS)

Intuition

- Tries to preserve given ideal pairwise distances in low-dimensional space

\[ \min_{x_1, \ldots, x_I} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2. \]

- Metric MDS
  - Preserves given ideal distance values

- Nonmetric MDS
  - When you only know/care about ordering of distances
  - Preserves only the orderings of distance values

- Algorithm: gradient-decent type

C.f. classical MDS is the same as PCA
Multidimensional Scaling (MDS)

Intuition

- Tries to preserve given ideal pairwise distances in low-dimensional space

\[
\min_{x_1, \ldots, x_I} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2.
\]

Metric MDS

- Preserves given ideal distance values

Nonmetric MDS

- When you only know/care about ordering of distances
- Preserves only the orderings of distance values

Algorithm: gradient-decent type

C.f. classical MDS is the same as PCA
Multidimensional Scaling (MDS)

Intuition

▸ Tries to preserve given ideal pairwise distances in low-dimensional space

\[
\min_{x_1, \ldots, x_I} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2.
\]

Nonlinear

▸ Metric MDS
  ▪ Preserves given ideal distance values

▸ Nonmetric MDS
  ▪ When you only know/care about ordering of distances
  ▪ Preserves only the orderings of distance values

▸ Algorithm: gradient-decent type

c.f. classical MDS is the same as PCA
Multidimensional Scaling (MDS)

Intuition

- Tries to preserve given ideal pairwise distances in low-dimensional space.

\[
\min_{x_1, \ldots, x_I} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2
\]

- Metric MDS
  - Preserves given ideal distance values

- Nonmetric MDS
  - When you only know/care about ordering of distances
  - Preserves only the orderings of distance values

Algorithm: gradient-decent type

c.f. classical MDS is the same as PCA
Multidimensional Scaling (MDS)

Intuition

▸ Tries to preserve given ideal pairwise distances in low-dimensional space

\[
\min_{x_1, \ldots, x_I} \sum_{i<j} \left( \|x_i - x_j\| - \delta_{i,j} \right)^2.
\]

▸ Metric MDS
  ▸ Preserves given ideal distance values

▸ Nonmetric MDS
  ▸ When you only know/care about ordering of distances
  ▸ Preserves only the orderings of distance values

▸ Algorithm: gradient-decent type

c.f. classical MDS is the same as PCA
Multidimensional Scaling (MDS)

Intuition
- Tries to preserve given ideal pairwise distances in low-dimensional space

\[
\min_{x_1, \ldots, x_I} \sum_{i < j} (\|x_i - x_j\| - \delta_{i,j})^2.
\]

- Metric MDS
  - Preserves given ideal distance values

- Nonmetric MDS
  - When you only know/care about ordering of distances
  - Preserves only the orderings of distance values

Algorithm: gradient-decent type

c.f. classical MDS is the same as PCA
Multidimensional Scaling
Sammon’s mapping

Sammon’s mapping

- Local version of MDS
- Down-weights errors in large distances

\[ E = \frac{1}{\sum_{i<j} d_{ij}^*} \sum_{i<j} (d_{ij}^* - d_{ij})^2 \]

- Algorithm: gradient-decent type

Nonlinear
Unsupervised
Local
Similarity input
Multidimensional Scaling
Force-directed graph layout

Force-directed graph layout

- Rooted from graph visualization, but essentially variant of metric MDS
- Spring-like attractive + repulsive forces between nodes
- Algorithm: gradient-decent type

Widely-used in visualization
- Aesthetically pleasing results
- Simple and intuitive
- Interactivity

Nonlinear
Unsupervised
Global
Similarity input
Multidimensional Scaling
Force-directed graph layout

Demos

➡️ Prefuse
  ➡️ http://prefuse.org/gallery/graphview/

➡️ D3: http://d3js.org/
  ➡️ http://bl.ocks.org/4062045
Multidimensional Scaling

In all variants,

- Pros: widely-used (works well in general)
- Cons: slow
  - Nonmetric MDS is even much slower than metric MDS

Nonlinear
Unsupervised
Global
Similarity input
Linear Discriminant Analysis

Maximally separates clusters by

- Putting different cluster as far as possible
- Putting each cluster as compact as possible

(a)  (b)
Linear Discriminant Analysis vs. Principal Component Analysis

2D visualization of 7 Gaussian mixture of 1000 dimensions

Linear discriminant analysis (Supervised)

Principal component analysis (Unsupervised)
Linear Discriminant Analysis

Maximally separates clusters by
- Putting different cluster as far as possible
- Putting each cluster as compact as possible

Algorithm: generalized eigendecomposition

Pros: better show cluster structure
Cons: may distort original relationship of data
Linear Discriminant Analysis

Maximally separates clusters by
- Putting different cluster as far as possible
- Putting each cluster as compact as possible

- Algorithm: generalized eigendecomposition
- Pros: better show cluster structure
- Cons: may distort original relationship of data
Linear Discriminant Analysis

Maximally separates clusters by
- Putting different cluster as far as possible
- Putting each cluster as compact as possible

- Algorithm: generalized eigendecomposition
- Pros: better show cluster structure
- Cons: may distort original relationship of data

Linear
Supervised
Linear Discriminant Analysis

Maximally separates clusters by
- Putting different cluster as far as possible
- Putting each cluster as compact as possible

Algorithm: generalized eigendecomposition
Pros: better show cluster structure
Cons: may distort original relationship of data

Linear
Supervised
Global
Linear Discriminant Analysis

Maximally separates clusters by
- Putting different cluster as far as possible
- Putting each cluster as compact as possible

- Algorithm: generalized eigendecomposition
- Pros: better show cluster structure
- Cons: may distort original relationship of data

Linear
Supervised
Global
Feature vectors
Linear Discriminant Analysis
Testbed Demo – Text Data
Nonnegative Matrix Factorization

Dimension reduction via matrix factorization

\[ A \approx WH \]

\[ \min ||A - WH||_F \]

\[ W \geq 0, \quad H \geq 0 \]

Why nonnegativity constraints?

- Better approximation vs. better interpretation
- Often physically/semantically meaningful

Algorithm: alternating nonnegativity-constrained least squares
Nonnegative Matrix Factorization as clustering

Dimension reduction via matrix factorization

\[ A \approx WH \]

\[ \min || A - WH ||_F \]

\[ W \geq 0, \ H \geq 0 \]

Often NMF performs better and faster than \( k \)-means

\( W \) : centroids, \( H \) : soft-clustering membership
In the next lecture..

More interesting topics coming up including

- Advanced methods
  - Isomap, LLE, kernel PCA, t-SNE, …
- Real-world applications in interactive visualization
- Practitioners’ guide
  - What to try first in which situations?