COMPSCI 589
Lecture 22: Multidimensional Scaling and Isomap

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Created with support from National Science Foundation Award# IIS-1350522.

Outline

1. Review
2. MDS
3. Isomap

Kernel PCA and Spectral Clustering

- Last class, we saw a method to achieve non-linear dimensionality reduction by combining basis expansions with dimensionality reduction.

Kernel PCA and Spectral Clustering

- Last class, we saw a method to achieve non-linear dimensionality reduction by combining basis expansions with dimensionality reduction.
- We showed that basis expansion can be combined with PCA and SVD, but that PCA could also be used with the kernel trick.

Kernel PCA and Spectral Clustering

- Last class, we saw a method to achieve non-linear dimensionality reduction by combining basis expansions with dimensionality reduction.
- We showed that basis expansion can be combined with PCA and SVD, but that PCA could also be used with the kernel trick.
- Lastly, we saw how spectral clustering can be viewed as a modification of clustering in the kernel PCA latent space.

Outline

Multidimensional Scaling

- MDS is a non-linear dimensionality reduction method that is explicitly designed to minimize the distortion in the pairwise distances between points when projecting them into a low dimensional embedding.

Multidimensional Scaling

- MDS is a non-linear dimensionality reduction method that is explicitly designed to minimize the distortion in the pairwise distances between points when projecting them into a low dimensional embedding.
- Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1:N}$.

Multidimensional Scaling

- MDS is a non-linear dimensionality reduction method that is explicitly designed to minimize the distortion in the pairwise distances between points when projecting them into a low dimensional embedding.
- Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1:N}$.
- Let d_{ij} be the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Any distance metric can be used. Euclidean distance $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ is common.

Least Squares Multidimensional Scaling

- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.

Least Squares Multidimensional Scaling

- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Least-squares MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function, known as the *stress* function:

Least Squares Multidimensional Scaling

- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Least-squares MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function, known as the *stress* function:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} \left(d_{ij} - ||\mathbf{z}_i - \mathbf{z}_j||_2\right)^2$$

Least Squares Multidimensional Scaling

- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Least-squares MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function, known as the *stress* function:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} (d_{ij} - ||\mathbf{z}_i - \mathbf{z}_j||_2)^2$$

- Basically, this function attempts to have the regular Euclidean distances between points in $\mathbb{R}^K$ reflect the distances between the points in $\mathbb{R}^D$. This can be useful for data visualization when $K = 2$.

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Classical MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function:

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Classical MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} \left(s_{ij} - \langle \mathbf{z}_i - \bar{\mathbf{z}}, \mathbf{z}_j - \bar{\mathbf{z}} \rangle\right)^2$$

Classical Multidimensional Scaling Variants

- Let s_{ij} be the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$.
- Let $\mathbf{z}_i \in \mathbb{R}^K$ be the low-dimensional embedding of $\mathbf{x}_i$ for each data case i. We assume $K < D$.
- Classical MDS learns the embeddings $\mathbf{z}_i$ by minimizing the following objective function:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_N} \sum_{i<j} \left(s_{ij} - \langle \mathbf{z}_i - \bar{\mathbf{z}}, \mathbf{z}_j - \bar{\mathbf{z}} \rangle\right)^2$$

- If the similarities are centered inner products in $\mathbb{R}^D$, Classical MDS and PCA are equivalent. For other similarity functions, classical MDS performs non-linear dimensionality reduction.

MDS Trade-offs

- Interestingly, to use MDS we actually don't need the raw feature vectors. It's enough to have the pairwise distance or similarity matrices.

MDS Trade-offs

- Interestingly, to use MDS we actually don't need the raw feature vectors. It's enough to have the pairwise distance or similarity matrices.
- Unlike kernel PCA, the similarity or distance matrices do not need to be valid kernel matrices (ie: they do not need to correspond to inner products of some basis expansion).

MDS Trade-offs

- Interestingly, to use MDS we actually don't need the raw feature vectors. It's enough to have the pairwise distance or similarity matrices.
- Unlike kernel PCA, the similarity or distance matrices do not need to be valid kernel matrices (ie: they do not need to correspond to inner products of some basis expansion).
- A significant issue with MDS is that we need to be able to specify a global similarity or distance matrix directly. This may not actually be easy to do if the data come from a complex manifold (regular Euclidean distance will fail).

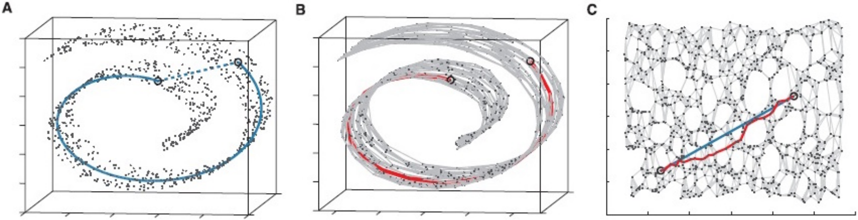
Outline

Isometric feature mapping

- Isometric feature mapping (Isomap) is a non-linear dimensionality reduction method that is designed to minimize the distortion in geodesic distances on a manifold when projecting them into a low dimensional embedding.

Isometric feature mapping

- Isometric feature mapping (Isomap) is a non-linear dimensionality reduction method that is designed to minimize the distortion in geodesic distances on a manifold when projecting them into a low dimensional embedding.



Isometric feature mapping algorithm

- The isomap algorithm begins by computing the K-nearest neighbors of each data point and building the distance weighted K-nearest neighbor graph G over the data.

Isometric feature mapping algorithm

- The isomap algorithm begins by computing the K-nearest neighbors of each data point and building the distance weighted K-nearest neighbor graph G over the data.
- For points i, j that are neighbors in the graph, isomap uses the straight line distance between them as d_{ij}.

Isometric feature mapping algorithm

- The isomap algorithm begins by computing the K-nearest neighbors of each data point and building the distance weighted K-nearest neighbor graph G over the data.
- For points i, j that are neighbors in the graph, isomap uses the straight line distance between them as d_{ij}.
- For points i, j that are not neighbors in the graph, isomap uses the length of the shortest path in G as d_{ij}. This is an approximation to the geodesic distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the manifold.

Isometric feature mapping algorithm

- The isomap algorithm begins by computing the K-nearest neighbors of each data point and building the distance weighted K-nearest neighbor graph G over the data.
- For points i, j that are neighbors in the graph, isomap uses the straight line distance between them as d_{ij}.
- For points i, j that are not neighbors in the graph, isomap uses the length of the shortest path in G as d_{ij}. This is an approximation to the geodesic distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the manifold.
- Finally, isomap plugs the distances d_{ij} into MDS with classical scaling and computes the embedding.