MAT 280: Harmonic Analysis on Graphs & Networks
Lecture 11: Distances on Graphs II: Applications of Commute-Time Distances

Naoki Saito

Department of Mathematics
University of California, Davis

October 31, 2019

Outline

1. Setup of Classification Problems
2. Intermezzo: Classical Multidimensional Scaling
3. Commute-Time Guided Transformation
4. A Face Recognition Algorithm
5. Numerical Experiments and Some Results
6. Sparse Graphs

Outline

1. Setup of Classification Problems
2. Intermezzo: Classical Multidimensional Scaling
3. Commute-Time Guided Transformation
4. A Face Recognition Algorithm
5. Numerical Experiments and Some Results
6. Sparse Graphs

Problem Setup

- This lecture is mainly based on the paper: Y. Deng, et al.: "Commute time guided transformation for feature extraction," *Computer Vision & Image Understanding*, vol. 116, pp. 473–483, 2012.
- Let X be the training data matrix, $X := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Let $\widetilde{X} := X(I_n - \mathbf{1}_n \mathbf{1}_n^{\mathsf{T}}/n)$, i.e., the *centered* data matrix (the mean of the column vectors $\overline{\boldsymbol{x}}$ is subtracted from each column vector).
- Let $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ be a low-dimensional embedding map with $s \ll d$. Let $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{s \times n}$ be the embedded training dataset using the map Ψ, i.e., $Z = \Psi(X) = (\Psi(\boldsymbol{x}_1), \ldots, \Psi(\boldsymbol{x}_n))$. An initial graph $G = G(V = X, E)$ using the training dataset X is built using either k-NN graph with the Euclidean distances or with the Gaussian similarities, or the *sparse graphs* (more about them later).

Problem Setup

- This lecture is mainly based on the paper: Y. Deng, et al.: "Commute time guided transformation for feature extraction," *Computer Vision & Image Understanding*, vol. 116, pp. 473–483, 2012.
- Let X be the training data matrix, $X := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Let $\widetilde{X} := X(I_n - \mathbf{1}_n \mathbf{1}_n^{\top}/n)$, i.e., the *centered* data matrix (the mean of the column vectors $\overline{\boldsymbol{x}}$ is subtracted from each column vector).
- Let $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ be a low-dimensional embedding map with $s \ll d$. Let $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{s \times n}$ be the embedded training dataset using the map Ψ, i.e., $Z = \Psi(X) = (\Psi(\boldsymbol{x}_1), \ldots, \Psi(\boldsymbol{x}_n))$. An initial graph $G = G(V = X, E)$ using the training dataset X is built using either k-NN graph with the Euclidean distances or with the Gaussian similarities, or the *sparse graphs* (more about them later).

Problem Setup

- This lecture is mainly based on the paper: Y. Deng, et al.: "Commute time guided transformation for feature extraction," *Computer Vision & Image Understanding*, vol. 116, pp. 473–483, 2012.
- Let X be the training data matrix, $X := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Let $\widetilde{X} := X(I_n - \mathbf{1}_n \mathbf{1}_n^{\mathsf{T}}/n)$, i.e., the *centered* data matrix (the mean of the column vectors $\overline{\boldsymbol{x}}$ is subtracted from each column vector).
- Let $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ be a low-dimensional embedding map with $s \ll d$. Let $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{s \times n}$ be the embedded training dataset using the map Ψ, i.e., $Z = \Psi(X) = (\Psi(\boldsymbol{x}_1), \ldots, \Psi(\boldsymbol{x}_n))$. An initial graph $G = G(V = X, E)$ using the training dataset X is built using either k-NN graph with the Euclidean distances or with the Gaussian similarities, or the *sparse graphs* (more about them later).

Problem Setup

- This lecture is mainly based on the paper: Y. Deng, et al.: "Commute time guided transformation for feature extraction," *Computer Vision & Image Understanding*, vol. 116, pp. 473–483, 2012.
- Let X be the training data matrix, $X := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Let $\widetilde{X} := X(I_n - \mathbf{1}_n \mathbf{1}_n^{\mathsf{T}}/n)$, i.e., the *centered* data matrix (the mean of the column vectors $\overline{\boldsymbol{x}}$ is subtracted from each column vector).
- Let $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ be a low-dimensional embedding map with $s \ll d$. Let $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{s \times n}$ be the embedded training dataset using the map Ψ, i.e., $Z = \Psi(X) = (\Psi(\boldsymbol{x}_1), \ldots, \Psi(\boldsymbol{x}_n))$. An initial graph $G = G(V = X, E)$ using the training dataset X is built using either k-NN graph with the Euclidean distances or with the Gaussian similarities, or the *sparse graphs* (more about them later).

Aims

- The main aims of this article are to answer the following natural questions using the face image databases:
  - What embedding Ψ should be used so that the commute-time distance $c(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and the squared Euclidean distance $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 =: \delta_{ij}^2$ are preserved as much as possible after embedding?
  - How to conduct *out-of-sample extension*, i.e., once a graph is built from a given training dataset X, how can we embed a new test sample that has *not* been used to construct the graph? This consideration is particularly important in classification and regression scenarios!
- The simplest idea for such an embedding is:

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|\sqrt{c_{ij}} - \delta_{ij}\|_2^2,$$

which is the so-called *classical Multidimensional Scaling* (MDS).

Aims

- The main aims of this article are to answer the following natural questions using the face image databases:
  - What embedding Ψ should be used so that the commute-time distance $c(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and the squared Euclidean distance $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 =: \delta_{ij}^2$ are preserved as much as possible after embedding?
  - How to conduct *out-of-sample extension*, i.e., once a graph is built from a given training dataset X, how can we embed a new test sample that has *not* been used to construct the graph? This consideration is particularly important in classification and regression scenarios!
- The simplest idea for such an embedding is:

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|\sqrt{c_{ij}} - \delta_{ij}\|_2^2,$$

which is the so-called *classical Multidimensional Scaling* (MDS).

Aims

- The main aims of this article are to answer the following natural questions using the face image databases:
  - What embedding Ψ should be used so that the commute-time distance $c(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and the squared Euclidean distance $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 =: \delta_{ij}^2$ are preserved as much as possible after embedding?
  - How to conduct *out-of-sample extension*, i.e., once a graph is built from a given training dataset X, how can we embed a new test sample that has *not* been used to construct the graph? This consideration is particularly important in classification and regression scenarios!
- The simplest idea for such an embedding is:

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|\sqrt{c_{ij}} - \delta_{ij}\|_2^2,$$

which is the so-called *classical Multidimensional Scaling* (MDS).

Aims

- The main aims of this article are to answer the following natural questions using the face image databases:
  - What embedding Ψ should be used so that the commute-time distance $c(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and the squared Euclidean distance $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 =: \delta_{ij}^2$ are preserved as much as possible after embedding?
  - How to conduct *out-of-sample extension*, i.e., once a graph is built from a given training dataset X, how can we embed a new test sample that has *not* been used to construct the graph? This consideration is particularly important in classification and regression scenarios!
- The simplest idea for such an embedding is:

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|\sqrt{c_{ij}} - \delta_{ij}\|_2^2,$$

which is the so-called *classical Multidimensional Scaling* (MDS).

Outline

1. Setup of Classification Problems
2. Intermezzo: Classical Multidimensional Scaling
3. Commute-Time Guided Transformation
4. A Face Recognition Algorithm
5. Numerical Experiments and Some Results
6. Sparse Graphs

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects* $\{\boldsymbol{x}_i\}$ *themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} between the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects $\{\boldsymbol{x}_i\}$ themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} betwen the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects $\{\boldsymbol{x}_i\}$ themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} between the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects $\{\boldsymbol{x}_i\}$ themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} betwen the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects* $\{\boldsymbol{x}_i\}$ *themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} betwen the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects $\{\boldsymbol{x}_i\}$ themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} betwen the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical Multidimensional Scaling

- Is one of the earliest embedding techiniques (Torgerson, 1952)
- Originally, only dissimilarities (or similarities) among n objects are given, *not the objects* $\{\boldsymbol{x}_i\}$ *themselves*.
- MDS is a visualization technique exploring dissimilarities (or similarities) among such n objects.
- More specifically, suppose the dissimilarity d_{ij} betwen the ith and jth objects is given, $i, j = 1, \ldots, n$. Then one possible version of classical MDS embeds (or allocates) such n objects in $\mathbb{R}^s$ such that

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}\subset\mathbb{R}^s} \sum_{i,j} \|d_{ij} - \delta_{ij}\|_2^2, \quad \delta_{ij} = \delta(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2.$$

- Unfortunately, there are two significant drawbacks.
  1. No closed-form solution to the MDS optimization exists, and most of them are based on iterative approaches $\Longrightarrow$ could be computationally expensive and get stuck at local minima.
  2. It is graph-dependent, i.e., all the data including the test samples must be used to contruct an initial graph, which is often infeasible.

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \dots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} (\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j)))^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \ldots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} (\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j)))^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \ldots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} \left(\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j))\right)^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

  where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \ldots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} (\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j)))^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

  where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \ldots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} (\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j)))^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

  where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Intermezzo: Classical MDS + Input Data Vectors ≡ PCA

- One simplification happens if instead of just similarities among objects *actual n objects are given* as a set of column vectors of $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$.
- Define the *similarity* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by the *centered correlation*

$$\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) := (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}).$$

- Suppose the centered correlation is also used to measure the similarity among the embedded objects $\boldsymbol{z}_i = \Psi(\boldsymbol{x}_i) \in \mathbb{R}^s$, $i = 1, \ldots, n$.
- Then, the classical MDS seeks the mapping Ψ that minimizes:

$$J_{\mathrm{CS}}(\Psi) := \sum_{i,j} (\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j) - \alpha(\Psi(\boldsymbol{x}_i), \Psi(\boldsymbol{x}_j)))^2 = \left\| \widetilde{X}^{\mathsf{T}} \widetilde{X} - \Psi(\widetilde{X})^{\mathsf{T}} \Psi(\widetilde{X}) \right\|_F^2.$$

- We can find this map using the *SVD* of $\widetilde{X} = U \Sigma V^{\mathsf{T}}$ as

$$\Psi(\widetilde{X}) = U_s^{\mathsf{T}} \widetilde{X} = \Sigma_s V_s^{\mathsf{T}},$$

  where U_s and V_s correpond to the first s left and right singular vectors, and Σ_s contains the corresponding singular values. This is *exactly the same as* using the first s components of *PCA*!

Outline

1. Setup of Classification Problems
2. Intermezzo: Classical Multidimensional Scaling
3. Commute-Time Guided Transformation
4. A Face Recognition Algorithm
5. Numerical Experiments and Some Results
6. Sparse Graphs

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^{\mathsf{T}} \boldsymbol{x}_i - \Psi^{\mathsf{T}} \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^{\mathsf{T}} \boldsymbol{x}_i - \Psi^{\mathsf{T}} \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation

- Recap: the classical MDS trying to preserve the commute-time distances is difficult to compute.
- Hence, Deng et al. introduced a new notion called "commute-time guided transformation."
- Find a unitary matrix $\Psi : \mathbb{R}^d \to \mathbb{R}^s$ that minimizes:

$$J_{CTG}(\Psi) := \sum_{i,j} \frac{\delta_{ij}^2}{c_{ij}} = \sum_{i,j} \frac{\|\Psi^{\mathsf{T}} \boldsymbol{x}_i - \Psi^{\mathsf{T}} \boldsymbol{x}_j\|_2^2}{c_{ij}}.$$

- If c_{ij} is small, then δ_{ij} should also be small enough to minimize $J_{CTG}(\Psi)$. A small c_{ij} with a large δ_{ij} may be penalized.
- On the other hand, if c_{ij} is large, then it allows a comparably large δ_{ij} in $\mathbb{R}^s$.
- In other words, the value of c_{ij} is used as a penalty to *guide* the optimization of $J_{CTG}(\Psi)$; hence the name: the "*commute-time guided transformation*."

Commute-Time Guided Transformation ...

$J_{CTG}(\Psi)$ can be simplified using matrices and trace:

$$\begin{aligned} J_{CTG}(\Psi) &= \sum_{i,j} \frac{1}{c_{ij}} \operatorname{tr}\left[\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)^\top\right] \\ &= \operatorname{tr}\left[\sum_{i,j} \frac{\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)^\top}{c_{ij}}\right] \\ &= 2\operatorname{tr}\left[\sum_i \frac{\Psi^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \Psi}{c_{i\bullet}} - \sum_{i,j} \frac{\Psi^\top \boldsymbol{x}_i \boldsymbol{x}_j^\top \Psi}{c_{ij}}\right] \quad \text{via symmetry} \\ &= 2\operatorname{tr}\left[\Psi^\top X(\Gamma - K)X^\top \Psi\right], \end{aligned}$$

where $c_{i\bullet} := \sum_j c_{ij}$, $K := \left(1/c_{ij}\right)$, and $\Gamma := \operatorname{diag}(1/c_{1\bullet}, \ldots, 1/c_{n\bullet})$.

The larger the Γ_{ii} is, the more important the ith vertex (i.e., the data vector $\boldsymbol{x}_i$) and its embedded point $\boldsymbol{z}_i$ become for the minimization problem.

Commute-Time Guided Transformation . . .

$J_{CTG}(\Psi)$ can be simplified using matrices and trace:

$$\begin{aligned} J_{CTG}(\Psi) &= \sum_{i,j} \frac{1}{c_{ij}} \operatorname{tr}\left[\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)^\top\right] \\ &= \operatorname{tr}\left[\sum_{i,j} \frac{\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)\left(\Psi^\top \boldsymbol{x}_i - \Psi^\top \boldsymbol{x}_j\right)^\top}{c_{ij}}\right] \\ &= 2\operatorname{tr}\left[\sum_i \frac{\Psi^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \Psi}{c_{i\bullet}} - \sum_{i,j} \frac{\Psi^\top \boldsymbol{x}_i \boldsymbol{x}_j^\top \Psi}{c_{ij}}\right] \quad \text{via symmetry} \\ &= 2\operatorname{tr}\left[\Psi^\top X(\Gamma - K)X^\top \Psi\right], \end{aligned}$$

where $c_{i\bullet} := \sum_j c_{ij}$, $K := \left(1/c_{ij}\right)$, and $\Gamma := \operatorname{diag}(1/c_{1\bullet}, \ldots, 1/c_{n\bullet})$.
The larger the Γ_{ii} is, the more important the ith vertex (i.e., the data vector $\boldsymbol{x}_i$) and its embedded point $\boldsymbol{z}_i$ become for the minimization problem.

Commute-Time Guided Transformation . . .

- With the constraints $Z\Gamma Z^{\mathsf{T}} = I_s$, we have the following constrained minimization problem:

$$\min_{\Psi \in \mathbb{R}^{d\times s}; \Psi^{\mathsf{T}}\Psi = I_s} \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] \quad \text{subject to } \Psi^{\mathsf{T}} X \Gamma X^{\mathsf{T}} \Psi = I_s.$$

- This can be solved by the method of Lagrange multipliers as follows:

$$J_{CTG}(\Psi, \Lambda) := \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] - \langle \Lambda, \Psi^{\mathsf{T}} X \Gamma X^{\mathsf{T}} \Psi - I_s \rangle,$$

  where $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix containing the Lagrange multipliers.

- Setting $\nabla_\Psi J_{CTG}(\Psi, \Lambda) = \mathbf{0}$ leads to the following generalized eigenvalue problem:

$$\underbrace{X(\Gamma - K)X^{\mathsf{T}}}_{P} \Psi = \underbrace{X \Gamma X^{\mathsf{T}}}_{Q} \Psi \Lambda, \quad \text{i.e., } P\boldsymbol{\psi}_j = \lambda_j Q \boldsymbol{\psi}_j,\ j = 1, \ldots, s.$$

- Compare this with the *Locality Preserving Projection* (LPP) of He and Niyogi (a.k.a. Laplacianfaces): $XLX^{\mathsf{T}}\Psi = XDX^{\mathsf{T}}\Psi\Lambda$.

- Hence, the correspondence: $A \Leftrightarrow K$, i.e., $a_{ij} \Leftrightarrow 1/c_{ij}$.

Commute-Time Guided Transformation ...

- With the constraints $Z\Gamma Z^{\mathsf{T}} = I_s$, we have the following constrained minimization problem:

$$\min_{\Psi \in \mathbb{R}^{d\times s}; \Psi^{\mathsf{T}}\Psi = I_s} \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K)X^{\mathsf{T}}\Psi\right] \quad \text{subject to } \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}}\Psi = I_s.$$

- This can be solved by the method of Lagrange multipliers as follows:

$$J_{CTG}(\Psi, \Lambda) := \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K)X^{\mathsf{T}}\Psi\right] - \left\langle \Lambda, \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}}\Psi - I_s \right\rangle,$$

where $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix containing the Lagrange multipliers.

- Setting $\nabla_\Psi J_{CTG}(\Psi, \Lambda) = \mathbf{0}$ leads to the following generalized eigenvalue problem:

$$\underbrace{X(\Gamma - K)X^{\mathsf{T}}}_{P}\Psi = \underbrace{X\Gamma X^{\mathsf{T}}}_{Q}\Psi\Lambda, \quad \text{i.e., } P\boldsymbol{\psi}_j = \lambda_j Q\boldsymbol{\psi}_j,\ j = 1, \ldots, s.$$

- Compare this with the *Locality Preserving Projection* (LPP) of He and Niyogi (a.k.a. Laplacianfaces): $XLX^{\mathsf{T}}\Psi = XDX^{\mathsf{T}}\Psi\Lambda$.
- Hence, the correspondence: $A \Leftrightarrow K$, i.e., $a_{ij} \Leftrightarrow 1/c_{ij}$.

Commute-Time Guided Transformation . . .

- With the constraints $Z\Gamma Z^{\mathsf{T}} = I_s$, we have the following constrained minimization problem:

$$\min_{\Psi \in \mathbb{R}^{d\times s}; \Psi^{\mathsf{T}}\Psi = I_s} \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] \quad \text{subject to } \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}} \Psi = I_s.$$

- This can be solved by the method of Lagrange multipliers as follows:

$$J_{CTG}(\Psi,\Lambda) := \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] - \left\langle \Lambda, \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}} \Psi - I_s \right\rangle,$$

  where $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix containing the Lagrange multipliers.

- Setting $\nabla_{\Psi} J_{CTG}(\Psi,\Lambda) = \mathbf{0}$ leads to the following generalized eigenvalue problem:

$$\underbrace{X(\Gamma - K)X^{\mathsf{T}}}_{P}\Psi = \underbrace{X\Gamma X^{\mathsf{T}}}_{Q}\Psi\Lambda, \quad \text{i.e., } P\boldsymbol{\psi}_j = \lambda_j Q\boldsymbol{\psi}_j,\ j = 1,\ldots,s.$$

- Compare this with the *Locality Preserving Projection* (LPP) of He and Niyogi (a.k.a. Laplacianfaces): $XLX^{\mathsf{T}}\Psi = XDX^{\mathsf{T}}\Psi\Lambda$.

- Hence, the correspondence: $A \Leftrightarrow K$, i.e., $a_{ij} \Leftrightarrow 1/c_{ij}$.

Commute-Time Guided Transformation . . .

- With the constraints $Z\Gamma Z^{\mathsf{T}} = I_s$, we have the following constrained minimization problem:

$$\min_{\Psi \in \mathbb{R}^{d\times s};\Psi^{\mathsf{T}}\Psi = I_s} \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K)X^{\mathsf{T}}\Psi\right] \quad \text{subject to } \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}}\Psi = I_s.$$

- This can be solved by the method of Lagrange multipliers as follows:

$$J_{CTG}(\Psi,\Lambda) := \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K)X^{\mathsf{T}}\Psi\right] - \left\langle \Lambda, \Psi^{\mathsf{T}} X\Gamma X^{\mathsf{T}}\Psi - I_s\right\rangle,$$

  where $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix containing the Lagrange multipliers.

- Setting $\nabla_\Psi J_{CTG}(\Psi,\Lambda) = \mathbf{0}$ leads to the following generalized eigenvalue problem:

$$\underbrace{X(\Gamma - K)X^{\mathsf{T}}}_{P}\Psi = \underbrace{X\Gamma X^{\mathsf{T}}}_{Q}\Psi\Lambda, \quad \text{i.e., } P\boldsymbol{\psi}_j = \lambda_j Q\boldsymbol{\psi}_j,\ j = 1,\ldots,s.$$

- Compare this with the *Locality Preserving Projection* (LPP) of He and Niyogi (a.k.a. Laplacianfaces): $XLX^{\mathsf{T}}\Psi = XDX^{\mathsf{T}}\Psi\Lambda$.

- Hence, the correspondence: $A \Leftrightarrow K$, i.e., $a_{ij} \Leftrightarrow 1/c_{ij}$.

Commute-Time Guided Transformation ...

- With the constraints $Z\Gamma Z^{\mathsf{T}} = I_s$, we have the following constrained minimization problem:

$$\min_{\Psi \in \mathbb{R}^{d\times s}; \Psi^{\mathsf{T}}\Psi = I_s} \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] \quad \text{subject to } \Psi^{\mathsf{T}} X \Gamma X^{\mathsf{T}} \Psi = I_s.$$

- This can be solved by the method of Lagrange multipliers as follows:

$$J_{CTG}(\Psi, \Lambda) := \operatorname{tr}\left[\Psi^{\mathsf{T}} X(\Gamma - K) X^{\mathsf{T}} \Psi\right] - \left\langle \Lambda, \Psi^{\mathsf{T}} X \Gamma X^{\mathsf{T}} \Psi - I_s \right\rangle,$$

  where $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix containing the Lagrange multipliers.

- Setting $\nabla_\Psi J_{CTG}(\Psi, \Lambda) = \mathbf{0}$ leads to the following generalized eigenvalue problem:

$$\underbrace{X(\Gamma - K)X^{\mathsf{T}}}_{P} \Psi = \underbrace{X\Gamma X^{\mathsf{T}}}_{Q} \Psi \Lambda, \quad \text{i.e., } P\boldsymbol{\psi}_j = \lambda_j Q \boldsymbol{\psi}_j,\ j = 1, \ldots, s.$$

- Compare this with the *Locality Preserving Projection* (LPP) of He and Niyogi (a.k.a. Laplacianfaces): $XLX^{\mathsf{T}}\Psi = XDX^{\mathsf{T}}\Psi\Lambda$.
- Hence, the correspondence: $A \Leftrightarrow K$, i.e., $a_{ij} \Leftrightarrow 1/c_{ij}$.

Outline

1 Setup of Classification Problems

2 Intermezzo: Classical Multidimensional Scaling

3 Commute-Time Guided Transformation

4 A Face Recognition Algorithm

5 Numerical Experiments and Some Results

6 Sparse Graphs

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d\times n}$; Test faces $Y \in \mathbb{R}^{d\times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d\times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (v_1, \ldots, v_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of v_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to v_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\upsilon_1, \ldots, \upsilon_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of υ_k from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to υ_k.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of $\boldsymbol{v}_k$ from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to $\boldsymbol{v}_k$.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^\dagger(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^\mathsf{T} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = \Psi^\mathsf{T} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of $\boldsymbol{v}_k$ from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to $\boldsymbol{v}_k$.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1:m$ do select the nearest neighbor of $\boldsymbol{v}_k$ from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to $\boldsymbol{v}_k$.
- Output: The list of labels of the test faces.

A Face Recognition Algorithm

- Input: Training faces $X \in \mathbb{R}^{d \times n}$; Test faces $Y \in \mathbb{R}^{d \times m}$.
- Training:
  1. Build a graph G from X;
  2. Compute the commute-time matrix $C = (c_{ij})$ using $L^{\dagger}(G)$.
  3. Compute matrices K and Γ.
  4. Solve the above generalized eigenvalue problem to obtain $\Psi \in \mathbb{R}^{d \times s}$.
  5. Embed the training faces via $Z = \Psi^{\mathsf{T}} X$.
- Recognition/Test:
  1. Embed the test faces via $\Upsilon = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = \Psi^{\mathsf{T}} Y$.
  2. For $k = 1 : m$ do select the nearest neighbor of $\boldsymbol{v}_k$ from the embedded training faces Z using the ℓ^2-distance in the embedded space $\mathbb{R}^s$. Then assign its label to $\boldsymbol{v}_k$.
- Output: The list of labels of the test faces.

Outline

1. Setup of Classification Problems
2. Intermezzo: Classical Multidimensional Scaling
3. Commute-Time Guided Transformation
4. A Face Recognition Algorithm
5. Numerical Experiments and Some Results
6. Sparse Graphs

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color $\rightarrow$ grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color $\rightarrow$ grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color $\rightarrow$ grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color → grayscale; normalization to 64×64 pixel resolution; histogram equalization, . . .
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color → grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color → grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments

- Face recognition rates over four different face databases were computed.
  - Yale face dataset: 165 faces of 15 individuals with various lighting conditions.
  - CMU PIE face dataset: 41,368 faces of 68 subjects under varying pose, illumination, expression.
  - AR dataset: over 4,000 faces of 126 individuals with varying illumination, expression, and occlusion.
  - FERET dataset: From NIST. More than 1,100 individuals with varying pose, illumination, expression.
- Each face image was preprocessed, e.g., color → grayscale; normalization to 64×64 pixel resolution; histogram equalization, ...
- Compared methods include: PCA, LDA, NMF (nonnegative matrix factorization), SR (sparse representation), LPP (locality preserving projection), GEO (geodesic projection), and CTG (commute-time guided transformation).

Numerical Experiments . . .

- For each face database, 50% of the faces (randomly selected) are used as the training faces, and the rest as the test faces.
- Repeat such random selection of the training faces and recognition of test faces 10 times for each method in each face database.
- For graph-based methods, k-NN graphs and *sparse graphs* were used.
- k of the k-NN graphs was fixed to be $k = n_t - 1$ where n_t is the average number of training samples for one individual.
- Various values of the dimension of the embedded space (or feature dimensionality) s were tested.

Numerical Experiments . . .

- For each face database, 50% of the faces (randomly selected) are used as the training faces, and the rest as the test faces.
- Repeat such random selection of the training faces and recognition of test faces 10 times for each method in each face database.
- For graph-based methods, k-NN graphs and *sparse graphs* were used.
- k of the k-NN graphs was fixed to be $k = n_t - 1$ where n_t is the average number of training samples for one individual.
- Various values of the dimension of the embedded space (or feature dimensionality) s were tested.

Numerical Experiments . . .

- For each face database, 50% of the faces (randomly selected) are used as the training faces, and the rest as the test faces.
- Repeat such random selection of the training faces and recognition of test faces 10 times for each method in each face database.
- For graph-based methods, k-NN graphs and *sparse graphs* were used.
- k of the k-NN graphs was fixed to be $k = n_t - 1$ where n_t is the average number of training samples for one individual.
- Various values of the dimension of the embedded space (or feature dimensionality) s were tested.

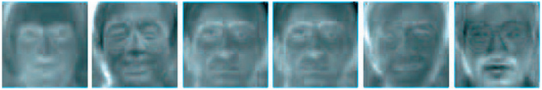
Numerical Experiments . . .

- For each face database, 50% of the faces (randomly selected) are used as the training faces, and the rest as the test faces.
- Repeat such random selection of the training faces and recognition of test faces 10 times for each method in each face database.
- For graph-based methods, k-NN graphs and *sparse graphs* were used.
- k of the k-NN graphs was fixed to be $k = n_t - 1$ where n_t is the average number of training samples for one individual.
- Various values of the dimension of the embedded space (or feature dimensionality) s were tested.

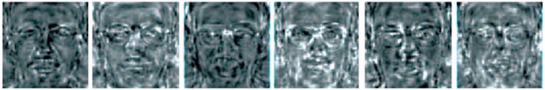
Numerical Experiments . . .

- For each face database, 50% of the faces (randomly selected) are used as the training faces, and the rest as the test faces.
- Repeat such random selection of the training faces and recognition of test faces 10 times for each method in each face database.
- For graph-based methods, k-NN graphs and *sparse graphs* were used.
- k of the k-NN graphs was fixed to be $k = n_t - 1$ where n_t is the average number of training samples for one individual.
- Various values of the dimension of the embedded space (or feature dimensionality) s were tested.

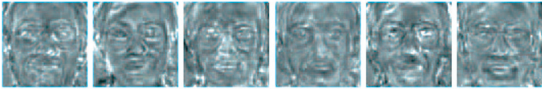
Some Results



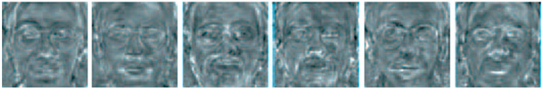
(a) Eigen-faces



(b) Fisher-faces



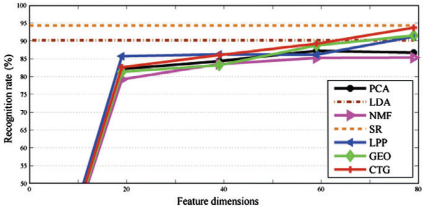
(c) Laplacian-faces



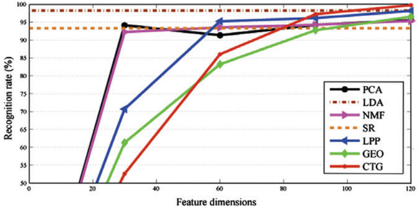
(d) CTG-faces

Fig. 2. The first six projections extracted from the Yale dataset based on (a) PCA, (b) LDA, (c) LPP, and (d) CTG.

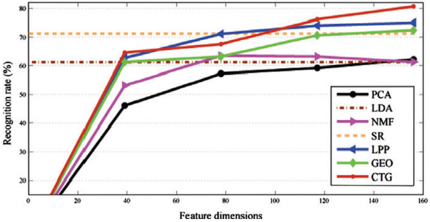
Some Results . . .



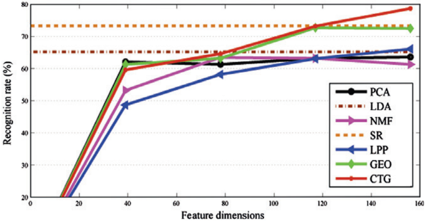
(a) Face recognition results in Yale dataset.



(b) Face recognition results in PIE dataset.



(c) Face recognition results in AR dataset.



(d) Face recognition results in FERET.

Fig. 4. Recognition rate versus different feature dimensionality based on the four different datasets.

Outline

1 Setup of Classification Problems

2 Intermezzo: Classical Multidimensional Scaling

3 Commute-Time Guided Transformation

4 A Face Recognition Algorithm

5 Numerical Experiments and Some Results

6 Sparse Graphs

Sparse Graphs

- New graph construction methods that were proposed relatively recently by H. Cheng et al. (2009) and by B. Cheng et al. (2010).
- Influenced by the idea of *compressed sensing*.
- *ℓ^1-graph* of B. Cheng et al. uses the sparse approximation of each $\boldsymbol{x}_i$ using all the other vectors $X^{(i)} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times (n-1)}$ via the following ℓ^1-minimization:

$$\min_{\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^{n-1}} \left\| \boldsymbol{\alpha}^{(i)} \right\|_1 \quad \text{subject to } \boldsymbol{x}_i = X^{(i)} \boldsymbol{\alpha}^{(i)},\ i = 1, \ldots, n.$$

  Then, if $\alpha_j^{(i)} > 0$, then set $a_{ij} = 1$. So, ℓ^1-graph is a sparse unweighted graph constructed from the input data vectors.
- *Sparseness Induced Graph* (SIG) of H. Cheng et al. uses the same ℓ^1 sparse approximation, but assigns weights via:

$$a_{ij} = \frac{\max\left(\alpha_j^{(i)}, 0\right)}{\sum_{k=1}^{n-1} \max\left(\alpha_k^{(i)}, 0\right)}.$$

Sparse Graphs

- New graph construction methods that were proposed relatively recently by H. Cheng et al. (2009) and by B. Cheng et al. (2010).
- Influenced by the idea of *compressed sensing*.
- *ℓ^1-graph* of B. Cheng et al. uses the sparse approximation of each $\boldsymbol{x}_i$ using all the other vectors $X^{(i)} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times (n-1)}$ via the following ℓ^1-minimization:

$$\min_{\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^{n-1}} \left\| \boldsymbol{\alpha}^{(i)} \right\|_1 \quad \text{subject to } \boldsymbol{x}_i = X^{(i)} \boldsymbol{\alpha}^{(i)}, \; i = 1, \ldots, n.$$

  Then, if $\alpha_j^{(i)} > 0$, then set $a_{ij} = 1$. So, ℓ^1-graph is a sparse unweighted graph constructed from the input data vectors.
- *Sparseness Induced Graph* (SIG) of H. Cheng et al. uses the same ℓ^1 sparse approximation, but assigns weights via:

$$a_{ij} = \frac{\max\left(\alpha_j^{(i)}, 0\right)}{\sum_{k=1}^{n-1} \max\left(\alpha_k^{(i)}, 0\right)}.$$

Sparse Graphs

- New graph construction methods that were proposed relatively recently by H. Cheng et al. (2009) and by B. Cheng et al. (2010).
- Influenced by the idea of *compressed sensing*.
- *ℓ^1-graph* of B. Cheng et al. uses the sparse approximation of each $\boldsymbol{x}_i$ using all the other vectors $X^{(i)} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times(n-1)}$ via the following ℓ^1-minimization:

$$\min_{\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^{n-1}} \left\| \boldsymbol{\alpha}^{(i)} \right\|_1 \quad \text{subject to } \boldsymbol{x}_i = X^{(i)} \boldsymbol{\alpha}^{(i)}, \; i = 1, \ldots, n.$$

  Then, if $\alpha_j^{(i)} > 0$, then set $a_{ij} = 1$. So, ℓ^1-graph is a sparse unweighted graph constructed from the input data vectors.
- *Sparseness Induced Graph* (SIG) of H. Cheng et al. uses the same ℓ^1 sparse approximation, but assigns weights via:

$$a_{ij} = \frac{\max\left(\alpha_j^{(i)}, 0\right)}{\sum_{k=1}^{n-1} \max\left(\alpha_k^{(i)}, 0\right)}.$$

Sparse Graphs

- New graph construction methods that were proposed relatively recently by H. Cheng et al. (2009) and by B. Cheng et al. (2010).
- Influenced by the idea of *compressed sensing*.
- *ℓ^1-graph* of B. Cheng et al. uses the sparse approximation of each $\boldsymbol{x}_i$ using all the other vectors $X^{(i)} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times (n-1)}$ via the following ℓ^1-minimization:

  $$\min_{\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^{n-1}} \left\| \boldsymbol{\alpha}^{(i)} \right\|_1 \quad \text{subject to } \boldsymbol{x}_i = X^{(i)} \boldsymbol{\alpha}^{(i)}, \ i = 1, \ldots, n.$$

  Then, if $\alpha_j^{(i)} > 0$, then set $a_{ij} = 1$. So, ℓ^1-graph is a sparse unweighted graph constructed from the input data vectors.
- *Sparseness Induced Graph* (SIG) of H. Cheng et al. uses the same ℓ^1 sparse approximation, but assigns weights via:

  $$a_{ij} = \frac{\max\left(\alpha_j^{(i)}, 0\right)}{\sum_{k=1}^{n-1} \max\left(\alpha_k^{(i)}, 0\right)}.$$