Some notes on SVD, dimensionality reduction, and clustering

Guangliang Chen

Contents

1. Introduction

This handout covers some advanced linear algebra and its use in dimensionality reduction and spectral clustering in the setting of unsupervised learning.

2. Review of necessary linear algebra

Notation. Vectors are denoted by boldface lowercase letters (such as $\mathbf{a}, \mathbf{b}$). To indicate their dimensions, we use notation like $\mathbf{a} \in \mathbb{R}^n$ to represent a n-dimensional vector. The ith element of $\mathbf{a}$ is written as a_i or $\mathbf{a}(i)$. We denote the constant vector of one as $\mathbf{1}$ (with its dimension implied by the context).

Matrices are denoted by boldface uppercase letters (such as $\mathbf{A}, \mathbf{B}$). Similarly, we write $\mathbf{A} \in \mathbb{R}^{m\times n}$ to indicate its size. The (i, j) entry of $\mathbf{A}$ is denoted by a_{ij} or $\mathbf{A}(i, j)$. The ith row of $\mathbf{A}$ is denoted by $\mathbf{A}(i, :)$ while its columns are written as $\mathbf{A}(:, j)$, as in MATLAB. We use $\mathbf{I}$ to denote the identity matrix (with its dimension implied by the context).

2.1. Matrix multiplication.

Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times k}$ be two real matrices. Their product is an $m \times k$ matrix $\mathbf{C} = (c_{ij})$ with entries

$$c_{ij} = \sum_{\ell=1}^{n} a_{i\ell} b_{\ell j} = \mathbf{A}(i, :) \cdot \mathbf{B}(:, j).$$

It is possible to obtain one full row (or column) of $\mathbf{C}$ via matrix-vector multiplication:

$$\mathbf{C}(i, :) = \mathbf{A}(i, :) \cdot \mathbf{B}$$
$$\mathbf{C}(:, j) = \mathbf{A} \cdot \mathbf{B}(:, j)$$

The full matrix $\mathbf{C}$ can be written as a sum of rank-1 matrices:

$$\mathbf{C} = \sum_{\ell=1}^{n} \mathbf{A}(:, \ell) \cdot \mathbf{B}(\ell, :).$$

When one of the matrices is a diagonal matrix, we have the following rules:

$$\underbrace{\mathbf{A}}_{\text{diagonal}} \mathbf{B} = \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{pmatrix} \begin{pmatrix} \mathbf{B}(1, :) \\ \vdots \\ \mathbf{B}(n, :) \end{pmatrix} = \begin{pmatrix} a_1\mathbf{B}(1, :) \\ \vdots \\ a_n\mathbf{B}(n, :) \end{pmatrix}$$

$$\mathbf{A} \underbrace{\mathbf{B}}_{\text{diagonal}} = [\mathbf{A}(:, 1) \dots \mathbf{A}(:, n)] \begin{pmatrix} b_1 & & \\ & \ddots & \\ & & b_n \end{pmatrix} = [b_1\mathbf{A}(:, 1) \dots b_n\mathbf{A}(:, n)]$$

Finally, below are some identities involving the vector $\mathbf{1}$:

$$\mathbf{1}\mathbf{1}^T = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}, \quad \mathbf{1}^T\mathbf{1} = 1$$

$$\mathbf{A}\mathbf{1} = \sum_j \mathbf{A}(:, j), \qquad \mathbf{1}^T\mathbf{A} = \sum_i \mathbf{A}(i, :), \qquad \mathbf{1}^T\mathbf{A}\mathbf{1} = \sum_i \sum_j \mathbf{A}(i, j).$$

Example 2.1. Let

$$\mathbf{A} = \begin{pmatrix} 3 & 0 & 0 \\ 5 & 1 & -1 \\ -2 & 2 & 4 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \\ 2 & 3 \end{pmatrix}, \mathbf{\Lambda}_1 = \begin{pmatrix} 1 & & \\ & 0 & \\ & & -1 \end{pmatrix}, \mathbf{\Lambda}_2 = \begin{pmatrix} 2 & \\ & -3 \end{pmatrix}.$$

Find the products $\mathbf{AB}, \mathbf{\Lambda}_1\mathbf{B}, \mathbf{B\Lambda}_2, \mathbf{1}^T\mathbf{B}, \mathbf{B1}$ and verify the above rules.

2.2. Eigenvalues and eigenvectors. Let $\mathbf{A}$ be an $n \times n$ real matrix. The characteristic polynomial of $\mathbf{A}$ is

$$p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}).$$

The (complex) roots λ_i of the characteristic equation $p(\lambda) = 0$ are called the eigenvalues of $\mathbf{A}$. For a specific eigenvalue λ_i, any nonzero vector $\mathbf{v}_i$ satisfying

$$(\mathbf{A} - \lambda_i\mathbf{I})\mathbf{v}_i = \mathbf{0}$$

or equivalently,

$$\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

is called an eigenvector of $\mathbf{A}$ (associated to the eigenvalue λ_i). All eigenvectors associated to λ_i span a linear subspace, called the eigenspace. It is denoted as $\mathrm{E}(\lambda_i)$. The dimension g_i of $\mathrm{E}(\lambda_i)$ is called the geometric multiplicity of λ_i, while the degree a_i of the factor $(\lambda - \lambda_i)^{a_i}$ in $p(\lambda)$ is called the algebraic multiplicity of λ_i. Note that we must have $\sum a_i = n$ and for all i, $1 \le g_i \le a_i$.

Example 2.2. For the matrix $\mathbf{A}$ given in the previous example, find the above quantities.

The following theorem indicates that the trace and determinant of a square matrix can both be computed from the eigenvalues of the matrix.

Theorem 2.1. *Let* $\mathbf{A}$ *be a real square matrix whose eigenvalues are* $\lambda_1, \ldots, \lambda_n$ *(counting multiplicities). Then*

$$\det(\mathbf{A}) = \prod_{i=1}^{n} \lambda_i \qquad \textit{and} \qquad \operatorname{trace}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i.$$

Example 2.3. For the matrix $\mathbf{A}$ defined previously, verify the identities in the above theorem.

Definition 2.1. A square matrix $\mathbf{A}$ is diagonalizable if it is similar to a diagonal matrix, i.e., there exist an invertible matrix $\mathbf{P}$ and a diagonal matrix Λ such that

$$\mathbf{A} = \mathbf{P\Lambda P}^{-1}.$$

Remark. If we write $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, then the above equation can be rewritten as $\mathbf{A}\mathbf{p}_i = \lambda_i\mathbf{p}_i$, for all $1 \le i \le n$. This shows that the λ_i are the eigenvalues of $\mathbf{A}$ and $\mathbf{p}_i$ the associated eigenvectors. Thus, the above factorization is called the eigenvalue decomposition of $\mathbf{A}$.

Example 2.4. The matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}$$

is diagonalizable because

$$\begin{pmatrix}0 & 1\\ 3 & 2\end{pmatrix} = \begin{pmatrix}1 & 1\\ 3 & -1\end{pmatrix}\begin{pmatrix}3 & \\ & -1\end{pmatrix}\begin{pmatrix}1 & 1\\ 3 & -1\end{pmatrix}^{-1}$$

but $\mathbf{B} = \begin{pmatrix}0 & 1\\ -1 & 2\end{pmatrix}$ is not (how to determine this?).

The following theorem provides a way for checking the diagonalizability of a square matrix.

THEOREM 2.2. *A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is diagonalizable if and only if it has n linearly independent eigenvectors.*

This theorem immediately implies the following results.

COROLLARY 2.3. *The following matrices are diagonalizable:*

- *Any matrix whose eigenvalues all have identical geometric and algebraic multiplicities, i.e., $g_i = a_i$ for all i;*
- *Any matrix with n distinct eigenvalues;*

2.3. Symmetric matrices. A symmetric matrix is a square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ whose transpose coincides with itself: $\mathbf{A}^T = \mathbf{A}$. Recall also that an orthogonal matrix is a square matrix whose columns and rows are both orthogonal unit vectors (i.e., orthonormal vectors):

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I},$$

or equivalently,

$$\mathbf{Q}^{-1} = \mathbf{Q}^T.$$

THEOREM 2.4. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then*

- *All the eigenvalues of $\mathbf{A}$ are real;*
- *$\mathbf{A}$ is orthogonally diagonalizable, i.e., there exists an orthogonal matrix $\mathbf{Q}$ and a diagonal matrix Λ such that*

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T.$$

Remark.

- For symmetric matrices, the eigenvalue decomposition is also called the spectral decomposition.
- The converse is also true. Therefore, a matrix is symmetric if and only if it is orthogonally diagonalizable.
- Write $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ and $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$. Then

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^T.$$

- We often sort the diagonals of Λ in decreasing order:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

EXAMPLE 2.5. Find the spectral decomposition of the following matrix

$$\mathbf{A} = \begin{pmatrix}0 & 2\\ 2 & 3\end{pmatrix}$$

Answer.

$$\mathbf{A} = \frac{1}{\sqrt{5}}\begin{pmatrix}1 & -2\\ 2 & 1\end{pmatrix}\cdot\begin{pmatrix}4 & \\ & -1\end{pmatrix}\cdot\frac{1}{\sqrt{5}}\begin{pmatrix}1 & -2\\ 2 & 1\end{pmatrix}^T$$

DEFINITION 2.2. A symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is positive semidefinite if $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$. It is positive definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ whenever $\mathbf{x} \neq \mathbf{0}$.

THEOREM 2.5. *A symmetric matrix* $\mathbf{A}$ *is positive definite (semidefinite) if and only if all the eigenvalues are positive (nonnegative).*

EXAMPLE 2.6. Determine if the following matrix is positive definite (or semidefinite):

$$\mathbf{A} = \begin{pmatrix}2 & -1 & 0\\ -1 & 2 & -1\\ 0 & -1 & 2\end{pmatrix}$$

2.4. MATLAB programming. The following list contains some basic MATLAB functions that apply to or generate matrices:

- diag
- trace
- det
- eig, eigs
- repmat
- ones, zeros
- eye
- rand

EXAMPLE 2.7. Redo all previous examples in MATLAB (this is a chance for you to get familiar with the MATLAB matrix operations).

EXAMPLE 2.8. For the matrices $\mathbf{B}, \mathbf{\Lambda}_1, \mathbf{\Lambda}_1$ defined in Example 2.1 (suppose they are already defined in MATLAB), compare the following ways of doing matrix products in MATLAB:

- $\mathbf{\Lambda}_1\mathbf{B}$: (1) *lambda1*B* (2) *repmat(diag(lambda)).*B*
- $\mathbf{B}\mathbf{\Lambda}_2$: (1) *B*lambda2* (2) *B.*repmat(diag(lambda2)')*

Do they give you the same product matrix in each case? Which approach is faster?

3. Matrix SVD and its applications

3.1. Decomposition of rectangular matrices. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$, which can be thought of a data matrix with rows representing points. Define $\mathbf{C} = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$. Then $\mathbf{C}$ is symmetric and positive semidefinite: $\mathbf{C}^T = (\mathbf{X}^T\mathbf{X})^T = \mathbf{X}^T\mathbf{X} = \mathbf{C}$, and $\mathbf{v}^T\mathbf{C}\mathbf{v} = \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} = (\mathbf{X}\mathbf{v})^T(\mathbf{X}\mathbf{v}) = \|\mathbf{X}\mathbf{v}\|^2 \geq 0$ for any $\mathbf{v} \in \mathbb{R}^d$.

We apply the spectral decomposition to $\mathbf{C}$ to derive the singular value decomposition of $\mathbf{X}$. First, there exists an orthogonal matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_d] \in \mathbb{R}^{d\times d}$ and a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d) \in \mathbb{R}^{d\times d}$ with $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ such that

$$\mathbf{C} = \mathbf{X}^T\mathbf{X} = \mathbf{V}\Lambda\mathbf{V}^T.$$

Rewrite the above equation as

$$\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}\Lambda.$$

Consider, for each $1 \leq i \leq d$, the ith column

$$\mathbf{X}^T\mathbf{X}\mathbf{v}_i = \lambda_i\mathbf{v}_i = \sigma_i^2\mathbf{v}_i, \tag{1}$$

where $\sigma_i = \sqrt{\lambda_i}$. For all $\sigma_1 \geq \cdots \geq \sigma_r > 0$, where $r = \mathrm{rank}(\mathbf{C}) = \mathrm{rank}(\mathbf{X})$, define

$$\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{X}\mathbf{v}_i \in \mathbb{R}^n.$$

Claim: $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are orthonormal vectors. The above is equivalent to

$$\mathbf{X}\mathbf{v}_i = \sigma_i\mathbf{u}_i, \quad i = 1, \ldots, r.$$

For all $r < i \leq n$ select unit vectors $\mathbf{u}_i \in \mathbb{R}^n$ such that

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r, \mathbf{u}_{r+1}, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$$

is an orthogonal matrix. Let Σ be an $n \times d$ matrix whose entries are all zero except the top $r \times r$ block

$$\Sigma(1:r, 1:r) = \mathrm{diag}(\sigma_1, \ldots, \sigma_r).$$

It is easy to verify that with the above choices of $\mathbf{U}$ and Σ, we must have

$$\mathbf{X}\mathbf{V} = \mathbf{U}\Sigma$$

Therefore, we have proved the following result.

THEOREM 3.1. *For any matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, there exist orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{V} \in \mathbb{R}^{d\times d}$ and a "diagonal" matrix $\Sigma \in \mathbb{R}^{n\times d}$ (with nonnegative entries) such that*

$$\mathbf{X}_{n\times d} = \mathbf{U}_{n\times n}\Sigma_{n\times d}\mathbf{V}_{d\times d}^T.$$

DEFINITION 3.1. The above decomposition of any matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ is called the Singular Value Decomposition (SVD) of $\mathbf{X}$:

- The diagonals of Σ are called the **singular values** of $\mathbf{X}$
- The columns of $\mathbf{U}$ are called the **left singular vectors** of $\mathbf{X}$
- The columns of $\mathbf{V}$ are called the **right singular vectors** of $\mathbf{X}$

Remark. It is easy to see that the left and right singular vectors of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}\mathbf{X}^T$ respectively while the singular values, once squared, are the common eigenvalues of the two product matrices.

Example 3.1. Compute the SVD of

$$\mathbf{X} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Answer.

$$\mathbf{X} = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{3} & \\ & 1 \\ & \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}^T$$

Remark. The above decomposition is often called the full SVD of $\mathbf{X}$, to distinguish from other versions:

- Economic/compact SVD: Let $r = \mathrm{rank}(\mathbf{X})$. Define
$$\begin{aligned} \mathbf{U}_{n\times r} &= [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{n\times r} \\ \mathbf{V}_{d\times r} &= [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{d\times r} \\ \Sigma_{r\times r} &= \mathrm{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r} \end{aligned}$$
We then have
$$\mathbf{X} = \mathbf{U}_{n\times r}\Sigma_{r\times r}\mathbf{V}_{d\times r}^T.$$
- Rank-1 decomposition:
$$\mathbf{X} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$
This has the interpretation that $\mathbf{X}$ is a weighted sum of rank-one matrices.

In sum, $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ where both $\mathbf{U}, \mathbf{V}$ have orthonormal columns and Σ is diagonal. Furthermore, $\mathbf{X}^T = \mathbf{V}\Sigma^T\mathbf{U}^T$ is the SVD of $\mathbf{X}^T$. Lastly, for any version, the SVD of a matrix is not unique.

3.2. Low-rank approximation of matrices. Recall that a norm associated with a vector space $\mathcal{V}$ is a function $\|\cdot\| : \mathcal{V} \to \mathbb{R}$ that satisfies three conditions:

- $\|\mathbf{v}\| \geq 0$ for all $\mathbf{v} \in \mathcal{V}$ and $\|\mathbf{v}\| = 0$ iff $\mathbf{v} = \mathbf{0}$
- $\|k\mathbf{v}\| = |k|\|\mathbf{v}\|$
- $\|\mathbf{v}_1 + \mathbf{v}_2\| \leq \|\mathbf{v}_1\| + \|\mathbf{v}_2\|$ for all $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}$

Example 3.2. In $\mathbb{R}^d$, there are at least three different norms:

- 2-norm (or Euclidean norm): $\|\mathbf{x}\|_2 = \sqrt{\sum x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$
- 1-norm (Taxicab norm or Manhattan norm): $\|\mathbf{x}\|_1 = \sum |x_i|$
- ∞-norm (maximum norm): $\|\mathbf{x}\|_\infty = \max |x_i|$

Whenever unspecified, it is understood as the Euclidean 2-norm.

We next define matrix norms. Just like vector norm is used to measure the magnitude of vectors ($\|\mathbf{v}\|$) and quantify the distance between vectors ($\|\mathbf{u} - \mathbf{v}\|$), matrix norm is used similarly.

Definition 3.2. The Frobenius norm of a matrix is defined as

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

Example 3.3. In the last example, $\|\mathbf{X}\|_F = 2$.

Proposition 3.2.

$$\|\mathbf{A}\|_F^2 = \operatorname{trace}(\mathbf{A}^T\mathbf{A}) = \operatorname{trace}(\mathbf{A}\mathbf{A}^T)$$

Theorem 3.3. *For any matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$,*

$$\|\mathbf{A}\|_F^2 = \sum \sigma_i^2$$

A second matrix norm is the 2-norm, or the spectral/operator norm.

Definition 3.3. The spectral/operator norm of a matrix is defined as

$$\|\mathbf{A}\|_2 = \max_{\mathbf{q}\in\mathbb{R}^d : \|\mathbf{q}\|_2 = 1} \|\mathbf{A}\mathbf{q}\|_2$$

Theorem 3.4. *For any matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, a maximizer of the above problem is the first right singular vector $\mathbf{v}_1$ of $\mathbf{A}$ and the maximum is*

$$\|\mathbf{A}\|_2 = \sigma_1.$$

Proof. Consider the full SVD of $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$. For any unit vector $\mathbf{q} \in \mathbb{R}^d$, write $\mathbf{q} = \mathbf{V}\alpha$ for some unit vector $\alpha \in \mathbb{R}^d$. Then $\mathbf{A}\mathbf{q} = \mathbf{A}(\mathbf{V}\alpha) = \mathbf{U}\Sigma\alpha$. Accordingly, $\|\mathbf{A}\mathbf{q}\|_2 = \|\mathbf{U}\Sigma\alpha\|_2 = \|\Sigma\alpha\|_2 = \sqrt{\sum \sigma_i^2\alpha_i^2} \le \sigma_1$, where the equality holds when $\alpha = \pm\mathbf{e}_1$ and correspondingly, $\mathbf{y} = \pm\mathbf{V}\mathbf{e}_1 = \pm\mathbf{v}_1$. $\square$

Example 3.4. In the last example, $\|\mathbf{X}\|_2 = \sqrt{3}$.

Corollary 3.5. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$. Then for all $\mathbf{x} \in \mathbb{R}^d$,*

$$\|\mathbf{A}\mathbf{x}\|_2 \le \|\mathbf{A}\|_2\|\mathbf{x}\|_2 = \sigma_1\|\mathbf{x}\|_2.$$

We note that the Frobenius and spectral norms of a matrix correspond to the 2- and ∞-norms of the vector of singular values. The 1-norm of singular values is called the nuclear norm of $\mathbf{A}$.

Definition 3.4. The nuclear norm of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ is

$$\|\mathbf{A}\|_* = \sum \sigma_i.$$

Example 3.5. In the last example, $\|\mathbf{X}\|_* = \sqrt{3} + 1$.

We now consider the low rank matrix approximation problem: For any $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $k \in \mathbb{Z}^+$, solve

$$\min_{\mathbf{B}\in\mathbb{R}^{n\times d} : \operatorname{rank}(\mathbf{B}) = k} \|\mathbf{A} - \mathbf{B}\|$$

where $\|\cdot\|$ could be any matrix norm (such as Frobenius, spectral).

Definition 3.5. For any $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $1 \le k \le r = \operatorname{rank}(\mathbf{A})$, define the truncated SVD of $\mathbf{A}$ as

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \in \mathbb{R}^{n\times d}.$$

Clearly, $\operatorname{rank}(\mathbf{A}_k) = k$.

THEOREM 3.6. *For each $1 \le k \le r$, $\mathbf{A}_k$ is the best rank-k approximation to $\mathbf{A} \in \mathbb{R}^{n\times d}$ under the Frobenius or spectral norm:*

$$\min_{\mathbf{B}\,:\,\mathrm{rank}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_F = \|\mathbf{A}-\mathbf{A}_k\|_F = \sqrt{\sum_{i>k}\sigma_i^2}$$

$$\min_{\mathbf{B}\,:\,\mathrm{rank}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_2 = \|\mathbf{A}-\mathbf{A}_k\|_2 = \sigma_{k+1}.$$

Remark. The theorem still holds true if the constraint $\mathrm{rank}(\mathbf{B}) = k$ is relaxed to $\mathrm{rank}(\mathbf{B}) \le k$.

EXAMPLE 3.6. In the last example, the best rank-1 approximation (under the Frobenius/spectral norm) is

$$\mathbf{X}_1 = \begin{pmatrix} 1 & -1 \\ -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}.$$

EXAMPLE 3.7. Take a digital image (matrix) and apply SVD to obtain low-rank approximations (an display as images):

$$\mathbf{A}_{m\times n} \approx \mathbf{U}_{m\times k}\Sigma_{k\times k}\mathbf{V}_{n\times k}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

This example shows that we can use SVD for image compression: By storing $\mathbf{U}, \Sigma, \mathbf{V}$ instead of $\mathbf{A}$, storage is reduced from mn to $k(m+n+1)$.

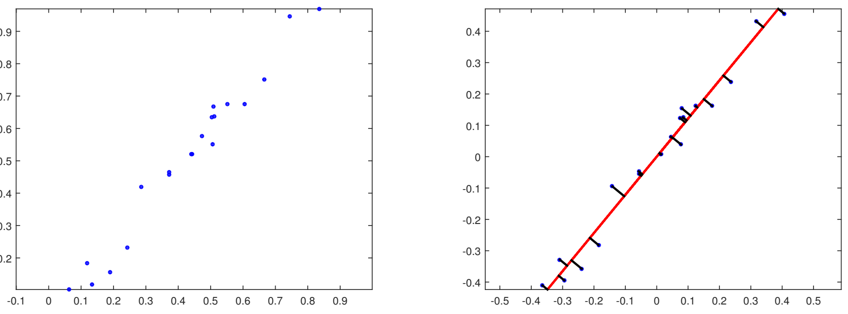


FIGURE 1. Illustration of the orthogonal fitting problem

3.3. Orthogonal Best-Fit Subspace. Consider the following problem: Given n data points $\mathbf{x}_i \in \mathbb{R}^d$, find the "best-fit" k-dimensional plane (see Fig. 1) which minimizes the total squared orthogonal error

$$\sum_{i=1}^{n} \|\mathbf{x}_i - \mathcal{P}_S(\mathbf{x}_i)\|_2^2$$

Remark. Compare with the least squares fitting problem in regression.

Suppose a k-dimensional plane $\mathcal{S}$ is used to fit the data. Let $\mathbf{m} \in \mathbb{R}^d$ represent a point in $\mathcal{S}$ and $\mathbf{B} \in \mathbb{R}^{d\times k}$ an orthonormal basis for it (i.e., $\mathbf{B}^T\mathbf{B} =$

$\mathbf{I}_{k\times k}$, but we do not have $\mathbf{B}\mathbf{B}^T = \mathbf{I}_{d\times d}$ for $k < d$) so that a parametric equation for the plane is $\mathbf{x} = \mathbf{m} + \mathbf{B}\alpha$. Since

$$\mathcal{P}_S(\mathbf{x}_i) = \mathbf{m} + \mathbf{B}\mathbf{B}^T(\mathbf{x}_i - \mathbf{m}).$$

we may rewrite the above problem as

$$\min_{\mathbf{m}\in\mathbb{R}^d,\,\mathbf{B}\in\mathbb{R}^{d\times k}} \sum \|\mathbf{x}_i - \mathbf{m} - \mathbf{B}\mathbf{B}^T(\mathbf{x}_i - \mathbf{m})\|^2$$

Using multivariable calculus, we can show that an optimal $\mathbf{m}$ is $\bar{\mathbf{x}} = \frac{1}{n}\sum \mathbf{x}_i$. Plugging in $\bar{\mathbf{x}}$ for $\mathbf{m}$ and letting $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ gives that

$$\min_{\mathbf{B}} \sum \|\tilde{\mathbf{x}}_i - \mathbf{B}\mathbf{B}^T\tilde{\mathbf{x}}_i\|^2.$$

In matrix notation, this is

$$\min_{\mathbf{B}} \|\widetilde{\mathbf{X}} - \widetilde{\mathbf{X}}\mathbf{B}\mathbf{B}^T\|_F^2$$

where $\widetilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]^T \in \mathbb{R}^{n\times d}$. The minimum is attained when $\widetilde{\mathbf{X}}\mathbf{B}\mathbf{B}^T = \widetilde{\mathbf{X}}_k$, the best rank-$k$ approximation of $\widetilde{\mathbf{X}}$, and the corresponding minimizer is the matrix consisting of the top k right singular vectors of $\widetilde{\mathbf{X}}$:

$$\mathbf{B} = \mathbf{V}(:, 1:k), \qquad \text{where} \quad \widetilde{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T.$$

We have thus proved the following result.

THEOREM 3.7. *A best-fit k-dimensional subspace to the data is given by*

$$\mathbf{x} = \mathbf{m} + \mathbf{B}\alpha$$

where

$$\mathbf{m} = \textit{center}, \quad \mathbf{B} = \textit{top } k \textit{ right singular vectors of centered data } \widetilde{\mathbf{X}}.$$

Moreover, the projection of $\widetilde{\mathbf{X}}$ onto the best-fit k-dimensional plane is $\widetilde{\mathbf{X}}_k$.

EXAMPLE 3.8. MATLAB demonstration.

3.4. Solving redundant linear systems. Consider solving a redundant linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a tall matrix ($m > n$) with full column rank and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. Such a system typically does not have an exact solution, so instead we seek an approximate solution $\mathbf{x}^*$ that optimally balance between all equations by solving

$$\min_{\mathbf{x}\in\mathbb{R}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

Using multivariable calculus, one can show that the optimal solution, often called the least squares solution, is

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

The matrix $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is called the pseudoinverse of $\mathbf{A}$: $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}_n$.

Here we show how to use the matrix SVD to solve the problem: Write $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$. Then we can rewrite the above problem as

$$\min_{\mathbf{x}\in\mathbb{R}} \|\mathbf{U}\Sigma\mathbf{V}^T\mathbf{x} - \mathbf{b}\|_2$$

The optimal $\mathbf{x}$ must satisfy

$$\Sigma\mathbf{V}^T\mathbf{x}^* = \mathbf{U}^T\mathbf{b}$$

which yields that

$$\mathbf{x}^* = \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{b}.$$

EXAMPLE 3.9. Show that $\mathbf{A}^\dagger = \mathbf{V}\Sigma^{-1}\mathbf{U}^T$. This implies that the two different methods actually yield the same solution.

Practice problems set 1

(1) Find, by hand, the economic SVD of the following matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 2 \\ 2 & 0 \\ 1 & 3 \\ 3 & 1 \end{pmatrix}$$

What are the different norms (Frobenius, Spectral and Nuclear) of this matrix?

(2) Now for the matrix in Question 1, use MATLAB to find the full SVD.

(3) Find the best-fit line (under the orthogonal error criterion) to the points in Question 1 (i.e., the rows of $\mathbf{A}$) and plot it with the data (by hand or computer). What are the coordinates of the projections of the original data onto the best-fit line?

(4) Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a square, invertible matrix and the SVD is

$$\mathbf{A} = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

Show that the inverse of $\mathbf{A}$ is

$$\mathbf{A}^{-1} = \sum_{i=1}^{n} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T.$$

(5) Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a matrix and $\vec{\sigma}$ the vector of its singular values. Which of the following norms are equal to each other? Draw a line between those that are equal.

$\|\mathbf{A}\|_F$	$\|\vec{\sigma}\|_\infty$
$\|\mathbf{A}\|_2$	$\|\vec{\sigma}\|_1$
$\|\mathbf{A}\|_*$	$\|\vec{\sigma}\|_2$

(6) First show that the product of two orthogonal matrices (of the same size) is also an orthogonal matrix. Then use this fact to show that

(a) If $\mathbf{L} \in \mathbb{R}^{m\times m}$ is orthogonal and $\mathbf{A} \in \mathbb{R}^{m\times n}$ is arbitrary, then the product $\mathbf{LA}$ has the same singular values and right singular vectors with $\mathbf{A}$.

(b) If $\mathbf{A} \in \mathbb{R}^{m\times n}$ is arbitrary and $\mathbf{R} \in \mathbb{R}^{n\times n}$ is orthogonal, then the product $\mathbf{AR}$ has the same singular values and left singular vectors with $\mathbf{A}$.

Note that an immediate consequence of the above results is that

$$\|\mathbf{LA}\| = \|\mathbf{A}\| = \|\mathbf{AR}\|$$

regardless of the norm (Frobenius/spectral/nuclear) used. Why?

(7) Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $\text{rank}(\mathbf{A}) = r$. Show that

(a) $\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_F \le \sqrt{r}\|\mathbf{A}\|_2$

(b) $\|\mathbf{A}\|_F \le \|\mathbf{A}\|_* \le \sqrt{r}\|\mathbf{A}\|_F$

(8) For any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ of full column rank, its pseudoinverse is defined as $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. Show that if the full SVD of $\mathbf{A}$ is $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$, then $\mathbf{A}^\dagger = \mathbf{V}\Sigma^{-1}\mathbf{U}^T$.

(9) Find the pseudoinverse of the matrix $\mathbf{A}$ in Question 1. What is the least squares solution of the equation $\mathbf{Ax} = \mathbf{1}$?

(10) Find a digital image from your album or the internet and compress it by using low rank approximation with various k. Which k seems to be sufficient?

4. Dimensionality Reduction

Many data sets live in high dimensional spaces, making their storage and processing very costly:

- **MNIST Handwritten Digits** (`http://yann.lecun.com/exdb/mnist/`): This data set consists of 70,000 digital images of handwritten digits that are collected from more than 250 different people. Each image is of size 28×28, and will be converted to a 784-dimensional vector.
- **USPS Zip Code Data** (`http://web.stanford.edu/~hastie/ElemStatLearn/datasets/zip.info.txt`). This data set contains 9298 handwritten digits automatically scanned from envelopes by the U.S. Postal Service. The scanned digits are saved in 16×16 grayscale images, but will be vectorized into $\mathbb{R}^{256}$.
- **Yale B Faces** (`http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html`). The extended Yale Face Database B contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions. Each image is of size 192×168, first downsampled to and then vectorized.
- **20 Newsgroups** (`http://qwone.com/~jason/20Newsgroups/`). This data set is a collection of 18774 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. Each document is also vectorized by counting the frequency of each word occurring in the document. The document vectors all have the same dimension of 61188, which is the total number of unique words in the text corpus, but they are extremely sparse.
- **ISOmap Data** (`http://web.mit.edu/cocosci/isomap/datasets.html`).

So in practice we often perform some sort of dimensionality reduction beforehand to project the data into a low dimensional space (for visualization and easy processing).

4.1. Principal component analysis (PCA). PCA projects the data onto a best-fit k-dimensional plane, which is determined by truncated SVD:

$$\widetilde{\mathbf{X}} \approx \underbrace{\mathbf{U}_{n\times k}\Sigma_{k\times k}}_{\text{coefficients}} \cdot \underbrace{\mathbf{V}^T_{d\times k}}_{\text{basis}} = \underbrace{\widetilde{\mathbf{X}}_k}_{\text{coordinates of projection}}$$

The first k columns of $\mathbf{V}$ represent the dominant directions of the data while the coefficients $\mathbf{U}_{n\times k}\Sigma_{k\times k} = \widetilde{\mathbf{X}}_{n\times d}\mathbf{V}_{d\times k}$ provide a new representation for the given data (with respect to the principal basis). We call them the principal components of the original data and use them as a low (k) dimensional embedding.

It is already known that the PCA plane minimizes the total squared orthogonal error. The following theorem indicates that PCA selects the principal directions to maximize the variances of the projections.

THEOREM 4.1. *For each $1 \le j \le k$, the variance of the projection of $\widetilde{\mathbf{X}}$ onto the $\mathbf{v}_j$ is σ_j^2. Moreover, these variances are the largest possible:*

$$\sigma_1^2 = \max_{\mathbf{v}:\, \|\mathbf{v}\|_2=1} \|\widetilde{\mathbf{X}}\mathbf{v}\|_2^2$$
$$\sigma_2^2 = \max_{\mathbf{v}:\, \|\mathbf{v}\|_2=1, \mathbf{v}_1^T\mathbf{v}=0} \|\widetilde{\mathbf{X}}\mathbf{v}\|_2^2$$
$$\vdots$$

EXAMPLE 4.1 (MNIST handwritten digit 1).

Below we address some practical questions.

Question 1: How do we select k? For visualization purposes k is normally set to 2 or 3; but for other machine learning tasks (such as clustering and classification), k needs to be much higher to avoid significant loss of information. There are two simple ways to select k practically:

- Set $k = \#$ dominant singular values (effective rank)
$$\widetilde{\mathbf{X}} = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$
- Choose k such that the top k principal directions explain a certain amount of variance of the data (e.g., 95%):
$$\frac{\sum_{i=1}^k \sigma_i^2}{\sum \sigma_i^2} > 95\%$$

Each criterion corresponds to a plot.

Question 2: What if PCA is applied to nonlinear data? PCA projection can be rewritten as follows:

$$\mathbf{y}_i = (\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{V}(:, 1:k), \quad i = 1, 2, \ldots, n$$

where $\bar{\mathbf{x}} = \frac{1}{n}\sum \mathbf{x}_i$ is the center of the data set and the columns of $\mathbf{V}$ are the principal directions directly learned from centered data:

$$\widetilde{\mathbf{X}} \approx \mathbf{U}\Sigma\mathbf{V}^T.$$

This shows that PCA is a linear embedding technique and will not capture the nonlinear geometry.

EXAMPLE 4.2 (SwissRoll).

4.2. Multidimensional scaling (MDS).

Consider the following problem: Assume a collection of n objects (e.g., cities) with pairwise distances $\{\ell_{ij}\}_{1\le i,j\le n}$. Represent them as points in some Euclidean space, $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^k$, such that

$$\ell_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|_2 \text{ (or as close as possible)}, \quad \forall\, i, j$$

Remark. Possible distance metrics that can be used by MDS:

- Euclidean distance (ℓ_2)
- Manhattan / Cityblock distance (ℓ_1)
- Minkowski distance (ℓ_p)
- Chebyshev / maximum coordinate difference (ℓ_∞)
- Cosine of the angle: $\|\frac{\mathbf{x}}{\|\mathbf{x}\|} - \frac{\mathbf{y}}{\|\mathbf{y}\|}\|^2 = 2 - 2\cos\theta$

- Geodesic distance (along curved dimensions)

We illustrate this problem with some examples.

Example 4.3. Given the distances between 20 cities in the U.S., display them on a (two-dimensional) map to preserve, as closely as possible, all the distances.

Example 4.4. Suppose we are given points in a very high dimensional space $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with some kind of distance $\ell_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$. We would like to find low dimensional representations, $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^k$ for some $k \ll d$, which can (approximately) preserve the given distances.

If the points are known to lie on a manifold (curve, surface, etc.), then one can use geodesic distance (shortest distance along the manifold) and try to preserve them in a low-dimensional Euclidean space. This is called the manifold learning problem.

To solve the MDS problem, first observe that the solutions are not unique, as any translation of the new points preserves the pairwise distances. We thus remove the translational invariance by adding a constraint

$$\sum \mathbf{y}_i = \mathbf{0}.$$

After squaring the above equations, we may expand them to get

$$\ell_{ij}^2 = \|\mathbf{y}_i\|^2 + \|\mathbf{y}_j\|^2 - 2\langle \mathbf{y}_i, \mathbf{y}_j \rangle$$

Summing over i and j separately to get

$$\sum_i \ell_{ij}^2 = \sum_i \|\mathbf{y}_i\|^2 + n\|\mathbf{y}_j\|^2$$
$$\sum_j \ell_{ij}^2 = n\|\mathbf{y}_i\|^2 + \sum_j \|\mathbf{y}_j\|^2$$

Denoting by

$$\ell_{\cdot j}^2 = \sum_i \ell_{ij}^2$$
$$\ell_{i\cdot}^2 = \sum_j \ell_{ij}^2$$
$$\ell_{\cdot\cdot}^2 = \sum_i \sum_j \ell_{ij}^2$$

we continue to sum over i, j separately:

$$\ell_{\cdot\cdot}^2 = n\sum_i \|\mathbf{y}_i\|^2 + n\sum_j \|\mathbf{y}_j\|^2 = 2n\sum_t \|\mathbf{y}_t\|^2$$

This implies that

$$\sum_t \|\mathbf{y}_t\|^2 = \frac{1}{2n}\ell_{\cdot\cdot}^2$$

Plugging back we then find

$$\|\mathbf{y}_j\|^2 = \frac{1}{n}\ell_{\cdot j}^2 - \frac{1}{2n^2}\ell_{\cdot\cdot}^2$$
$$\|\mathbf{y}_i\|^2 = \frac{1}{n}\ell_{i\cdot}^2 - \frac{1}{2n^2}\ell_{\cdot\cdot}^2$$

and finally

$$\langle \mathbf{y}_i, \mathbf{y}_j \rangle = \underbrace{\frac{1}{2}\left(\frac{1}{n}\ell_{i\cdot}^2 + \frac{1}{n}\ell_{\cdot j}^2 - \frac{1}{n^2}\ell_{\cdot\cdot}^2 - \ell_{ij}^2\right)}_{:=g_{ij}}, \quad \forall\, i, j$$

Let $\mathbf{G} = (g_{ij})$ be an $n \times n$ matrix and define $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n\times d}$. Then the last equation may be rewritten as

$$\mathbf{Y}\mathbf{Y}^T = \mathbf{G}.$$

Remark. It can be shown that

$$\mathbf{G} = -\frac{1}{2}\mathbf{J}\mathbf{L}\mathbf{J}, \quad \text{where } \mathbf{L} = (\ell_{ij}^2) \text{ and } \mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

From this we conclude that the matrix $\mathbf{G}$ is symmetric with all row and column sums equal to zero.

The solution of the problem, if it exists, is still not unique, because any rotation of a solution $\mathbf{Y}$, i.e., $\mathbf{YQ}$ for some orthogonal matrix $\mathbf{Q}$, is also a solution. In practice, we only want to find one solution in order to represent the original data in a Euclidean space. Suppose

$$\mathbf{G} = \mathbf{U}\Lambda\mathbf{U}^T$$

is the spectral decomposition. An exact solution of the above equation is

$$\mathbf{Y} = \mathbf{U}\Lambda^{1/2} = [\sqrt{\lambda_1}\mathbf{u}_1 \dots \sqrt{\lambda_r}\mathbf{u}_r].$$

We have thus proved the following result.

Theorem 4.2. *The problem*

$$\|\mathbf{y}_i - \mathbf{y}_j\|_2 = \ell_{ij}, \ \forall\, i, j \qquad \textit{and} \qquad \sum \mathbf{y}_i = \mathbf{0}$$

has the following exact solution

$$\mathbf{Y} = \mathbf{U}\Lambda^{1/2} = [\sqrt{\lambda_1}\mathbf{u}_1 \dots \sqrt{\lambda_r}\mathbf{u}_r]$$

where $(\lambda_i, \mathbf{u}_i)$ *are the eigenpairs of the* $\mathbf{G}$ *matrix.*

Remark. If the eigenvalues decay quickly, we then truncate the columns to obtain an approximate solution

$$\mathbf{Y}_k \approx [\sqrt{\lambda_1}\mathbf{u}_1 \dots \sqrt{\lambda_k}\mathbf{u}_k]$$

where $k < r$ is the number of dominant eigenvalues.

Remark. In the special case of the input data being the Euclidean distances between points $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$, the MDS approach is equivalent to PCA. To see this, first note that $\mathbf{Y} = \widetilde{\mathbf{X}}$ is already a solution to the MDS problem. Accordingly, $\mathbf{G} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$. From the SVD of $\widetilde{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T$ we obtain the eigenvalue decomposition of $\mathbf{G} = \mathbf{U}\Sigma^2\mathbf{U}^T$. Therefore, another exact solution is $\mathbf{Y} = \mathbf{U}\Sigma$, which differs from $\widetilde{\mathbf{X}}$ by a rotation represented by the orthogonal matrix $\mathbf{V}$ (but this solution allows to find low-dimensional approximate solutions based on the decay of the singular values). This remark also shows that PCA, in addition to preserving variances, preserves the pairwise Euclidean distances of the original data.

Example 4.5. Consider a collection of mutual distances (along the earth surface) among 20 US cities. Use MDS to find a two-dimensional representation of the cities (i.e., draw a map).

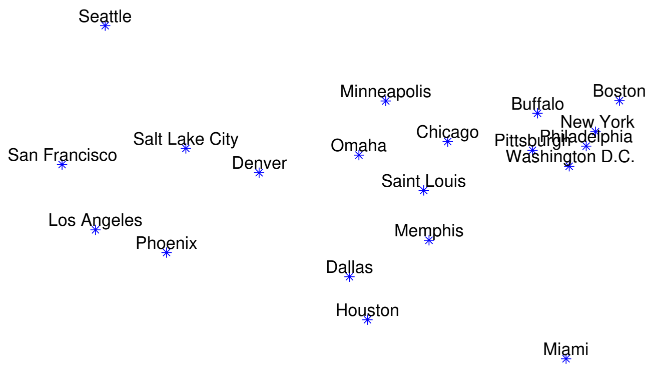


Figure 2. Locations of 20 US cities on a map found by MDS

Example 4.6 (MNIST handwritten digit 1). Apply MDS with ℓ_1 distance to embed the images into 2 dimensions.

In general, to select k and evaluate the quality of approximation, one can use the following measure.

Definition 4.1. The Kruskal stress is defined as

$$\text{Stress} = \sqrt{\frac{\sum_{i,j}(\ell_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|_2)^2}{\sum_{i,j}\ell_{ij}^2}}.$$

Empirically, the fit is good if stress < 0.1, and unacceptable if stress $>$ 0.15.

4.3. ISOmap. Briefly, ISOmap is MDS with geodesic distances (it assumes that the data lives on some smooth manifold).

4.3.1. *Motivation.* Consider Swissroll and MNIST digits 1. There are at least three drawbacks about PCA:

- The PCA dimension is bigger than the manifold dimension
- PCA may project faraway points inside the manifold together
- PCA does not capture the curved dimensions (its principal directions are generally not meaningful).

4.3.2. *How to find geodesic distances.* Two steps:

- Build a neighborhood graph from the given data by connecting only "nearby points" (by using either ϵ-ball or kNN approaches)
- Apply Disjkstra's algorithm with the graph to find all geodesic distances

Practice problems set 2

(1) Download all the code and data from my webpage (`http://www.math.sjsu.edu/~gchen/`) and make sure you can execute them successfully and reproduce all results obtained in class.

(2) Select all images of the digit 0 (or another digit you prefer) in the MNIST dataset to perform PCA. Display the following:
- The center of the handwritten 0's as an image of size 28×28 (this is how the "average" writer writes the digit 0)
- The first 50 singular values and their explained variances (two plots). How large does k need to be if the goal is to preserve 90% of the variance in the data set?
- The top 20 principal directions (i.e., right singular vectors) as images of size 28×28
- The top two principal components of the data (in order to visualize the data)

(3) Perform PCA with all images of digits 0 and 1 in the MNIST data set and display the top two principal components with the clusters colored differently. How good is the separation between them in 2 dimensions?

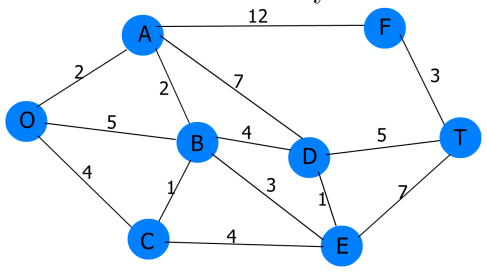
(4) Repeat the previous question with digits 4 and 9. Which pair seems easier to be separated?

(5) Perform MDS with the city block (ℓ_1) distance to the pair of digits 4 and 9 and display the 2 dimensional embedding. Is it different from what you obtained by PCA?

(6) Perform MDS on a data set called *ChineseCityData.mat* that contains the mutual distances of 12 Chinese cities to produce a two-dimensional map (for clarity let's place the City of Urumqi in the top left corner). Find also the stress number. How good is your map (compared with the Google map of China)?

(7) Download the ISOmap code from my website (note that the code provided on the ISOmap website has an error; it has been fixed in the version I provide). Perform ISOmap on the images of digit 4 and interpret the low dimensional representation you got.

(8) Use Dijkstra's algorithm to calculate, by hand, the shortest distance from the node O to every other node in the following graph:



(9) Verify that the $\mathbf{G}$ matrix used in MDS satisfies

$$\mathbf{G} = -\frac{1}{2}\mathbf{JLJ}, \quad \text{where } \mathbf{L} = (\ell_{ij}^2) \text{ and } \mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

Use this formula to show that the matrix $\mathbf{G}$ is symmetric with all row and column sums equal to zero.

5. Introduction to clustering

5.1. Clustering basics.

Problem 5.1. Assume a data set $\mathbf{X} \in \mathbb{R}^{n\times d}$. We would like to divide the data set into k disjoint subsets (called clusters) such that within every cluster points are "similar" (at least to their near neighbors) and between clusters points are "dissimilar".

Remark. Clustering is an unsupervised machine learning task, often called learning without a teacher. In contrast, classification is supervised (learning with a teacher) because training examples from each class are available.

5.1.1. *Different types of clusters:*

- Center-based (kmeans)
- Distribution-based (mixture of Gaussians)
- Geometry-based (subspace clustering)
- Contiguity-based (manifold clustering)
- Content-based (document clustering, image segmentation)

5.1.2. *Applications of clustering.*

- Digits clustering
- Face clustering
- Document grouping
- Image segmentation

5.1.3. *Some necessary components of clustering:*

- Objects and their attributes (i.e., data matrix)
- Number of clusters
- Similarity or dissimilarity measure, e.g.,
  - Euclidean distance (ℓ_2)
  - Manhattan distance (ℓ_1)
  - Maximum direction (ℓ_∞)
  - Cosine of the angle
  - Correlation coefficient
- Objective function

5.1.4. *Clustering is hard!*

- Similarity measure hard to pick
- Number of clusters often unknown and hard to determine (consider hierarchical clustering)

5.2. *k*means clustering. The $k$means algorithm aims to solve the following clustering problem.

Problem 5.2. Given a set of n points in $\mathbb{R}^d$, $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, and a positive integer k, find a partition of X into k disjoint clusters $C_1, \dots, C_k$ such that the total scatter is the smallest possible:

$$\min_{X=C_1\cup\cdots\cup C_k} \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j} \|\mathbf{x} - \mathbf{c}_j\|^2, \quad \text{where } \mathbf{c}_j = \frac{1}{n_j}\sum_{\mathbf{x}\in C_j} \mathbf{x}$$

The original problem is combinatorial in nature (as the naive approach of checking every possible partition has a complexity of $\mathcal{O}(n^k)$). $k$means is a fast, approximate solution that is based on iterations.

5.2.1. *Intuition.* We rewrite the above problem as a more "complicated" one:

$$\min_{\{C_j\},\{\mathbf{a}_j\}} \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j} \|\mathbf{x}-\mathbf{a}_j\|^2$$

where each $\mathbf{a}_j$ represents a landmark point for C_j. Choosing the clusters and landmark points simultaneously to minimize the total scatter is difficult; however,

- Given the clusters $C_1, \dots, C_k$, the optimal landmark points are their centers: $\mathbf{a}_j = \mathbf{c}_j$, $\forall j$;
- Given the landmark points $\mathbf{a}_1, \dots, \mathbf{a}_k$, the optimal clusters are formed by assigning points to nearest landmark points.

We thus adopt an iterative procedure, starting with a random guess of the landmark points, to solve the above problem.

5.2.2. *Algorithm.* See Alg. 1.

Algorithm 1 Psuedocode of $k$means clustering (Lloyd's algorithm)

```
Input: Data $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$, number of clusters $k$
Output: A partition of $X$ into $k$ clusters $C_1, \dots, C_k$
Steps:
1: Initialization: Randomly select $k$ initial centers $\mathbf{c}_1^{(0)}, \dots, \mathbf{c}_k^{(0)}$
2: Let $t \leftarrow 1$ be the iteration index.
   Repeat
       • Cluster update: For each $1 \le j \le k$, assign to $C_j^{(t)}$ the points
         that are closest to $\mathbf{c}_j^{(t-1)}$
       • Center update: $\mathbf{c}_j^{(t)} = \frac{1}{|C_j^{(t)}|} \sum_{\mathbf{x}\in C_j^{(t)}} \mathbf{x}, j = 1, \dots, k$
       • $t \leftarrow t+1$
   until some stopping criterion has been reached, e.g., when $t > 100$ or
   total scatter stops decreasing
3: Return the final clusters $C_j^{(t)}, \dots, C_j^{(t)}$
```

5.2.3. *Experiments.* See Matlab demonstration.

5.2.4. *Advantages and disadvantages.*

- pros: simple and fast (often, but not always), and always converges
- cons: not always the best partition; cannot handle nonconvex data

5.2.5. *How to initialize kmeans.* A few methods:

- Randomly select points from data set
- Pick faraway points
- Pick high density points
- $k$means++

5.2.6. *How to determine k.*

- The elbow method
- Xmeans

5.2.7. *Other measures of cluster quality.* $\ell - 1$ distance and maximum distance, which lead to k-medians and k-balls clustering.

6. Spectral clustering

Spectral clustering refers to a family of clustering algorithms that are based on the spectral decomposition of similarity matrices.

6.1. The perturbation perspective of clustering. Given a data set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ to be separated into k clusters, the Ng-Jordan-Weiss (NJW) version of spectral clustering starts by computing a matrix $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{n\times n}$ of pairwise similarities:

$$w_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, & i \neq j \\ 0, & i = j \end{cases}$$

where $\sigma > 0$ is a fixed parameter whose value will be set by the user. It then performs a symmetric normalization of $\mathbf{W}$

$$\widetilde{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}},$$

where

$$\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n), \quad d_i = \sum_{j=1}^n w_{ij}$$

or equivalently, in compact matrix notation

$$\mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{1}).$$

Afterward, the top k eigenvectors of $\widetilde{\mathbf{W}} \in \mathbb{R}^{n\times n}$ are taken to form $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_k] \in \mathbb{R}^{n\times k}$, and further $\widetilde{\mathbf{U}} \in \mathbb{R}^{n\times k}$ by normalizing the rows of $\mathbf{U}$ to have unit length. Lastly, k-means is applied to the row vectors of $\widetilde{\mathbf{U}}$ (which represent the original points in the same order) to cluster them into k disjoint subsets. We list all the steps in Alg. 2.

Algorithm 2 NJW spectral clustering

```
Input: Data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$, # clusters $k$, and scale parameter $\sigma$
Output: A partition of $X = C_1 \cup \cdots \cup C_k$
Steps:
1: Construct the similarity matrix $\mathbf{W} = (w_{ij})$ by
       $w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, \quad i \neq j \quad (0 \text{ otherwise})$
2: Compute $\mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{1})$ and use it to normalize $\mathbf{W}$ to get $\widetilde{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$
3: Find the top $k$ eigenvectors of $\widetilde{\mathbf{W}}$ to form an eigenvectors matrix $\mathbf{U} \in \mathbb{R}^{n\times k}$
4: Renormalize the rows of $\mathbf{U}$ to have unit length (denote the new matrix by $\widetilde{\mathbf{U}}$)
5: Apply $k$-means to cluster the row vectors of $\widetilde{\mathbf{U}}$ into $k$ groups and assign labels to the original data accordingly.
```

If we ignore the two normalization steps (i.e., 2 and 4), then NJW spectral clustering essentially consists of calculating two matrices followed by kmeans clustering:

- **W**: This matrix encodes the similarity information in the data: $0 \le w_{ij} \le 1$ for all i, j and for any fixed σ, the closer two points are, the larger their similarity score is. This actually defines an undirected, weighted graph $\mathcal{G} = (V, E)$ with each vertex $v_i \in V$ representing a data point $\mathbf{x}_i$ and each edge $e_{ij} \in E$ (from $\mathbf{x}_i$ to $\mathbf{x}_j$) being weighted by w_{ij}. As we shall see, such interpretation of **W** will lead to a graph cut formulation of spectral clustering.
- **U**: This matrix provides a k-dimensional embedding of the given data for easy clustering by kmeans. To see this, we regard the weight matrix as a feature matrix and note that its left singular vectors are the same as the eigenvectors (because **W** is symmetric).

We carry out all the steps of Alg. 2 by hand in an ideal scenario where the weight matrix **W** is given by

$$w_{ij} = \begin{cases} 1, & x_i, x_j \text{ in same true cluster;} \\ 0, & \text{otherwise} \end{cases}$$

In reality, any **W** calculated from real data can be seen as a perturbed version of this ideal matrix.

6.1.1. *Experiments.*

Example 6.1. Computer demonstration.

How to choose σ:

- Average distance from jth nearest neighbor
- Median distance from jth nearest neighbor
- Self-tuning spectral clustering

6.2. Applications.

6.2.1. *Image segmentation.*

6.2.2. *face clustering.*

6.2.3. *Text documents clustering.*

6.3. The graph cut perspective of clustering. Given a data set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ to be clustered, we represent it by a similarity graph $\mathcal{G} = (V, E)$. Each vertex $v_i \in V$ represents a data point $\mathbf{x}_i$. Two vertices $v_i, v_j \in V$ are connected by an edge e_{ij} if the corresponding data points $\mathbf{x}_i$ and $\mathbf{x}_j$ are "sufficiently similar".

The problem of clustering can now be reformulated as a graph cut problem: we want to find a partition of the graph such that

- edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and
- edges within a group have high weights (which means that points within the same cluster are similar to each other).

Below are the different kinds of similarity graphs:

- The ϵ neighborhood graph: connect any two points $\mathbf{x}_i, \mathbf{x}_j$ whose distance is less than ϵ
- The kNN graph (and mutual kNN): connect any two points $\mathbf{x}_i, \mathbf{x}_j$ if one is within the k nearest neighbors of the other (weak kNN) or both are within the k nearest neighbors of each other (strong/mutual kNN);
- The fully connected graph: connect any two points $\mathbf{x}_i, \mathbf{x}_j$ but assign a weight to the edge
$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, \ \forall i \neq j$$
where $\sigma > 0$ is a scale parameter whose value is fixed.

The first two methods typically yield undirected and unweighted graphs, while the last one generates undirected, weighted graphs.

DEFINITION 6.1. The set of all pairwise weights in a weighted graph define a weight matrix
$$\mathbf{W} \in \mathbb{R}^{n\times n} \qquad \text{with} \quad w_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \ \forall i \neq j$$

PROPOSITION 6.1. *Properties of the weight matrix.*

- *All entries of* $\mathbf{W}$ *are between 0 and 1 (in particular,* $w_{ii} = 0$*);*
- $\mathbf{W}$ *is symmetric;*
- $\mathbf{W}$ *is nearly block-diagonal (if the data is ordered according to the clusters).*

6.4. Graph terminology. Let $\mathcal{G} = (V, E, \mathbf{W})$ be an undirected, weighted graph with vertices $V = \{v_1, \ldots, v_n\}$ and weights $w_{ij} \geq 0$ (there is an edge e_{ij} connecting v_i and v_j if and only if $w_{ij} > 0$). For example, a graph is completely determined by the following weight matrix:
$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & .0 & .8 & 0 & 0 \\ .8 & .8 & 0 & .1 & 0 \\ 0 & 0 & .1 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}$$

Note that unweighted graphs are also included if 0/1 weights are used. We introduce some terminology commonly used in graph theory below.

Two vertices are adjacent if they are connected by an edge (i.e., $w_{ij} > 0$). An edge is incident on a vertex if the vertex is an endpoint of the edge. Therefore, the weight matrix is also referred to as the adjacency or incidence matrix.

The degree of a vertex $v_i \in V$ is defined as

$$d_i = \deg(v_i) = \sum_{j=1}^{n} w_{ij}.$$

It measures the connectivity of a point. The degrees of all vertices yield a degree matrix

$$\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}.$$

Given a subset of vertices $A \subset V$ we denote its complement by $\bar{A} = V - A$. We define the indicator vector $\mathbf{1}_A$ associated to A by

$$\mathbf{1}_A = (a_1, \ldots, a_n)^T, \quad a_i = 1 \text{ (if } v_i \in A) \text{ and } a_i = 0 \text{ (if } v_i \in \bar{A}).$$

For any two subsets $A, B \subset V$ (not necessarily disjoint), we define

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

When $B = \bar{A}$, the total weight $W(A, B) = W(A, \bar{A})$ is called a cut.

There are two ways to measure the "size" of $A \subset V$:

$$|A| = \#\text{vertices in } A;$$
$$\operatorname{vol}(A) = \sum_{i \in A} d_i = W(A, V).$$

The former simply counts the number of vertices in A while the latter measures how strongly the vertices in A are connected to all vertices of $\mathcal{G}$.

A path in the graph is a sequence of vertices and edges in between such that no vertex or edge can repeat. Other terms include walk, cycle, etc.

A subset $A \subset V$ of a graph is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A. A is called a connected component if it is connected and if there are no connections between vertices in A and $\bar{A}$. The nonempty sets $A_1, \ldots, A_k$ form a partition of the graph if $A_i \cap A_j = \emptyset$, $\forall i \neq j$ and $A_1 \cup \cdots \cup A_k = V$.

6.5. Graph Laplacian. Let $\mathcal{G}$ be an undirected, weighted graph with weight matrix $\mathbf{W}$ and degree matrix $\mathbf{D}$.

Definition 6.2. The unnormalized graph Laplacian is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

Remark. $\mathbf{L}_{ii} = -\sum_{j \neq i} w_{ij}$ and $\mathbf{L}_{ij} = -w_{ij}$ for all $i \neq j$.

The graph Laplacian has many important properties.

Proposition 6.2. *Let $\mathbf{L} \in \mathbb{R}^{n \times n}$ represent a graph Laplacian. Then*

- $\mathbf{L}$ *is symmetric.*
- *All the rows (and columns) sum to 0, i.e.,* $\mathbf{L}\mathbf{1} = \mathbf{0}$. *This implies that* $\mathbf{L}$ *has a eigenvalue 0 with eigenvector* $\mathbf{1}$.

- *For every vector* $\mathbf{f} \in \mathbb{R}^d$ *we have*
$$\mathbf{f}'\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$
This implies that $\mathbf{L}$ *is positive semidefinite and accordingly, its eigenvalues are all nonnegative:* $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$.
- *The algebraic multiplicity of the eigenvalue 0 equals the number of connected components in the graph.*

Example 6.2. Consider the modified graph

$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & .0 & .8 & 0 & 0 \\ .8 & .8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}$$

It can be shown that

$$\det(\lambda\mathbf{I} - \mathbf{L}) = \lambda(\lambda - 2.4)^2 \cdot \lambda(\lambda - 1.8).$$

Thus, the unnormalized graph Laplacian has a repeated eigenvalue 0, with multiplicity equal to the number of connected components. (The original connected graph has eigenvalues $0, 0.0788, 1.8465, 2.4000, 2.4747$)

We next define two normalized graph Laplacians.

Definition 6.3.

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W};$$
$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}.$$

Remark.

- $\mathbf{L}_{\text{sym}}$ is symmetric while $\mathbf{L}_{\text{rw}}$ is not, but they are similar to each other: $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1/2}\mathbf{L}_{\text{sym}}\mathbf{D}^{1/2}$.
- Both have the same eigenvalues, but different eigenvectors: $\mathbf{v}$ is an eigenvector of $\mathbf{L}_{\text{rw}}$ if and only if $\mathbf{D}^{1/2}\mathbf{v}$ is an eigenvector of $\mathbf{L}_{\text{sym}}$. In particular, for the eigenvalue 0, the associated eigenvectors are $\mathbf{1}$ (for $\mathbf{L}_{\text{rw}}$) and $\mathbf{D}^{1/2}\mathbf{1}$ (for $\mathbf{L}_{\text{sym}}$).
- The multiplicity of the zero eigenvalue is also equal to the number of connected components in the graph.
- $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ defines a random walk on the graph; in contrast, $\widetilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ has practical advantages because of positive semidefiniteness.
- $\mathbf{L}_{\text{rw}}$ is adopted by Shi and Malik while $\widetilde{\mathbf{W}}$ is used by Ng, Jordan and Weiss.

6.6. Shi and Malik's algorithm. Shi and Malik propose to perform 2-way spectral clustering by seeking an optimal balanced cut:

$$\min_{A\cup B=V, A\cap B=\emptyset} \text{cut}(A,B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right) = \text{Ncut}(A,B).$$

Remark. Other measures of the cut include

- Cheeger constant
$$\text{CheegerCut}(A,B) = \frac{\text{cut}(A,B)}{\min(\text{vol}(A), \text{vol}(B))}$$
- Min-max cut
$$\text{MinMaxCut}(A,B) = \text{cut}(A,B)\left(\frac{1}{W(A,A)} + \frac{1}{W(B,B)}\right)$$
- Ratio cut
$$\text{RatioCut}(A,B) = \text{cut}(A,B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$$

We show that the normalized cut can be expressed in terms of the graph Laplacian.

THEOREM 6.3. *Given a similarity graph $\mathcal{G} = \{V, \mathbf{W}\}$ and a partition $A \cup B = V$, we have*
$$\text{Ncut}(A,B) = \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{D}\mathbf{x}},$$
where
$$\mathbf{x} = \sqrt{\frac{\text{vol}(B)}{\text{vol}(A)}}\mathbf{1}_A - \sqrt{\frac{\text{vol}(A)}{\text{vol}(B)}}\mathbf{1}_B, \quad x_i = \begin{cases} \sqrt{\frac{\text{vol}(B)}{\text{vol}(A)}}, & i \in A \\ -\sqrt{\frac{\text{vol}(A)}{\text{vol}(B)}}, & i \in B \end{cases}$$

PROOF. It can shown by direct calculation:
$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \text{vol}(V)\text{Ncut}(A,B)$$
$$\mathbf{x}^T\mathbf{D}\mathbf{x} = \text{vol}(V).$$
□

Remark. The vector $\mathbf{x}$ is completely defined by the partition, contains only two distinct values, and satisfies a hidden constraint:
$$\mathbf{x}^T\mathbf{D}\mathbf{1} = 0.$$
To see this, write
$$\mathbf{x}^T\mathbf{D}\mathbf{1} = \sum_i x_i d_i = \sqrt{\frac{\text{vol}(B)}{\text{vol}(A)}}\sum_{i\in A} d_i - \sqrt{\frac{\text{vol}(A)}{\text{vol}(B)}}\sum_{i\in B} d_i = \sqrt{\text{vol}(A)\text{vol}(B)} - \sqrt{\text{vol}(A)\text{vol}(B)} = 0.$$

Remark. The choice of $\mathbf{x}$ is not unique. In fact,
$$\mathbf{x} = \frac{1}{\text{vol}(A)}\mathbf{1}_A - \frac{1}{\text{vol}(B)}\mathbf{1}_B, \quad x_i = \begin{cases} \frac{1}{\text{vol}(A)}, & i \in A \\ \frac{-1}{\text{vol}(B)}, & i \in B \end{cases}$$
would work too and have the same properties. This is left as homework for you to verify.

We have thus arrived at the following equivalent problem:
$$\min_{\substack{\mathbf{x}\in\{a,-b\}^n:\\ \mathbf{x}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{D}\mathbf{x}}.$$

This problem is NP-hard, and in order to solve it efficiently, we relax the requirement that $\mathbf{x}$ take only two distinct values to consider only

$$\min_{\substack{\mathbf{x}\in\mathbb{R}^n\\ \mathbf{x}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{D}\mathbf{x}}.$$

Theorem 6.4. *A minimizer of the above Rayleigh quotient problem is given by the second smallest eigenvector of* $\mathbf{L}_{rw}$*:*

$$\mathbf{L}_{rw}\,\mathbf{x} = \lambda_2\mathbf{x}.$$

Proof. Define $\mathbf{y} = \mathbf{D}^{1/2}\mathbf{x}$. Then the above problem can be rewritten as

$$\min_{\substack{\mathbf{y}\in\mathbb{R}^n\\ \mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1}=0}} \frac{\mathbf{y}^T\mathbf{L}_{\text{sym}}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}.$$

Since $\mathbf{D}^{1/2}\mathbf{1}$ is an eigenvector of $\mathbf{L}_{\text{sym}}$ corresponding to 0, the minimizer is given by the second smallest eigenvector of $\mathbf{L}_{\text{sym}}$. In terms of $\mathbf{x}$, this is

$$\mathbf{L}_{\text{sym}}\mathbf{D}^{1/2}\mathbf{x} = \lambda_2\mathbf{D}^{1/2}\mathbf{x}$$

or

$$\mathbf{L}_{\text{rw}}\mathbf{x} = \lambda_2\mathbf{x}.$$

□

Remark. The RatioCut criterion leads to the following relaxed problem:

$$\min_{\substack{\mathbf{x}\in\mathbb{R}^n:\\ \mathbf{x}^T\mathbf{1}=0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}.$$

The solution is obviously given by the second smallest eigenvector of the unnormalized graph Laplacian $\mathbf{L}$:

$$\mathbf{L}\mathbf{x} = \lambda_2\mathbf{x}.$$

The resulting algorithm is equivalent to that of Ncut when the clusters have comparable sizes and is worse otherwise (you will see this in a homework problem).

Algorithm 3 Normalized Cut for 2-way clustering (by Shi and Malik)

```
Input: Data $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$, number of clusters $k$, and scale parameter $\sigma$
Output: A partition of $X = C_1 \cup C_2$
Steps:
 1: Construct a weighted graph by assigning weights
        $w_{ij} = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
 2: Find the second smallest eigenvector $\mathbf{v}$ of the normalized graph Laplacian $\mathbf{L}_{rw}$
 3: Assign labels based on the sign of the coordinates of $\mathbf{v}$
```

Practice problems set 3

(1) Let

$$\mathbf{A} = \begin{pmatrix} a & b & \cdots & b \\ b & a & \cdots & b \\ \vdots & & & \\ b & b & \cdots & a \end{pmatrix} \in \mathbb{R}^{m\times m}$$

Show that $\mathbf{A}$ has eigenvalues $a+(m-1)b$ and $a-b$ with algebraic multiplicities of 1 and $m-1$ respectively. Moreover, an eigenvector associated to the eigenvalue $a+(m-1)b$ is $\mathbf{1}/\sqrt{m}$.

(2) First apply the kmeans algorithm to cluster the MNIST digits 0 and 1 directly. Now perform PCA on the two groups to reduce the dimensionality to $k \in \{50, 100, 150\}$ and run kmeans again to separate the two groups of digits. What are your accuracy rates?

(3) We know that kmeans clustering tries to minimize the total scatter. Let $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$ represent any cluster with mean $\bar{\mathbf{y}}$. Then

$$\sum_{i=1}^{m} \|\mathbf{y}_i - \bar{\mathbf{y}}\|^2 = \frac{1}{m} \sum_{1\le i<j\le m} \|\mathbf{y}_i - \mathbf{y}_j\|^2$$

This implies that kmeans also minimizes the total sum of squared pairwise distances.

(4) Implement on your own the NJW version of spectral clustering and evaluate it on the two-Gaussians data generated by the following piece of code:
n = 100;
*X = [randn(n,2)*0.5 + 1; randn(n,2)*0.5 - 1];*
labels = reshape(repmat(1:2, n, 1), 1, []);
What is the value of the scale parameter σ you used?

(5) Consider NJW spectral clustering with similarity matrix $\mathbf{W}$ and degree matrix $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$. Define $\mathbf{L} = \mathbf{D} - \mathbf{W}$, the graph Laplacian matrix. Show that
- $\mathbf{L}$ is symmetric.
- All the rows (and columns) of $\mathbf{L}$ sum to 0, i.e., $\mathbf{L1} = \mathbf{0}$. This implies that $\mathbf{L}$ has a eigenvalue 0 with eigenvector $\mathbf{1}$.
- For every vector $\mathbf{f} \in \mathbb{R}^d$ we have

$$\mathbf{f}'\mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

  This implies that $\mathbf{L}$ is positive semidefinite and accordingly, its eigenvalues are all nonnegative: $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$.