DDA5001 Machine Learning

Unsupervised Leanring: Dimensionality Reduction

Xiao Li

School of Data Science
The Chinese University of Hong Kong, Shenzhen

Dimensionality Reduction

- Observe samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, without labels.

Dimensionality reduction: Find a closest point to $\boldsymbol{x}_i$ in a lower dimensional space, i.e.,

$$\mathbb{R}^d \ni \boldsymbol{x}_i \to \tilde{\boldsymbol{x}}_i \in \mathbb{R}^k,$$

where $k \ll d$.

- Contrary to kernel methods in supervised learning.

The motivation of dimensionality reduction:

- Reducing redundant information.
- Help algorithms to be more computationally efficient (in lower dimension).
- Preventing overfitting, especially when $n < d$ (data preprocessing for supervised learning).

Dimensionality reduction is an important unsupervised learning technique. The main methods for dimensionality reduction are feature selection and feature extraction. We will focus on the later.

Dimensionality Reduction

Principal Component Analysis (PCA)

Apply PCA to Real Image Dataset

Principal Component Analysis

Principal component analysis (PCA): Find a low-dimensional approximation of high-dimensional data by minimizing the squared norms of distances.

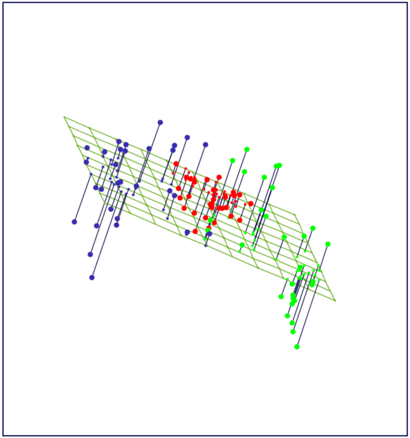
PCA modeling of date:

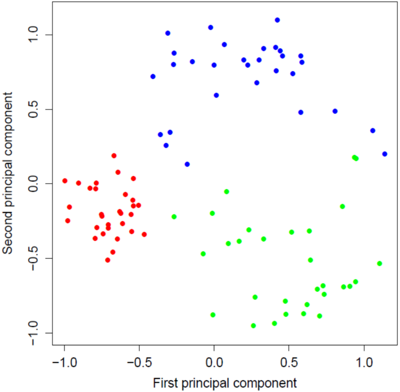
$$\boxed{\boldsymbol{x} \approx \boldsymbol{A}\boldsymbol{\theta} + \boldsymbol{\mu}}$$

- $\boldsymbol{x} \in \mathbb{R}^d$ is the original sample.
- $\boldsymbol{A} = [\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k] \in \mathbb{R}^{d \times k}$ with orthogonal columns, i.e., satisfying $\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}_k$. Matrix $\boldsymbol{A}$ is often called basis.
- $\boldsymbol{\theta} \in \mathbb{R}^k$ is the principle component.
- $\boldsymbol{\mu}$ is the mean of the samples.

Interpretation: $\boldsymbol{x}$ (after removing the mean $\boldsymbol{\mu}$) can be approximated by a k-dimensional point $\tilde{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{\theta}$.

Principal Component Analysis - Illustration





The PCA Learning Problem

PCA boils down to

$$\boxed{\underset{\boldsymbol{\mu}, \{\boldsymbol{\theta}_i\}, \boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}_k}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{\theta}_i - \boldsymbol{\mu}\|_2^2}$$

- Only samples $\{\boldsymbol{x}_i\}$ are known. Others are unknowns.
- It is a nonconvex optimization problem.
- The hard part is to solve for $\boldsymbol{A}$.
- Given $\boldsymbol{A}$, finding $\boldsymbol{\mu}$ and $\{\boldsymbol{\theta}_i\}$ is easy.

Solve for $\boldsymbol{\theta}_i$

Given $\boldsymbol{A}, \boldsymbol{\mu}$

$$\underset{\{\boldsymbol{\theta}_i\}}{\text{minimize}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{\theta}_i - \boldsymbol{\mu}\|_2^2.$$

Solution:

$$\boldsymbol{\theta}_i = \boldsymbol{A}^\top(\boldsymbol{x}_i - \boldsymbol{\mu})$$

Why? It is just a standard least square problem with $\boldsymbol{A}$ being semi-orthogonal.

Solve for $\boldsymbol{\mu}$

Suppose given $\boldsymbol{A}$, setting $\boldsymbol{\theta}_i = \boldsymbol{A}^\top(\boldsymbol{x}_i - \boldsymbol{\mu})$, we have

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \ \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top(\boldsymbol{x}_i - \boldsymbol{\mu}) - \boldsymbol{\mu}\|_2^2.$$

It is equivalent to

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \ \sum_{i=1}^{n} \|(\mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top)(\boldsymbol{x}_i - \boldsymbol{\mu})\|_2^2.$$

It is further equivalent to

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \ \sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})^\top(\mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top)^\top(\mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top)(\boldsymbol{x}_i - \boldsymbol{\mu}).$$

Let $\boldsymbol{B} = (\mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top)^\top(\mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top) = \mathbf{I} - \boldsymbol{A}\boldsymbol{A}^\top$.

Solve for $\boldsymbol{\mu}$

Take the gradient with respect to $\boldsymbol{\mu}$ gives

$$\nabla_{\boldsymbol{\mu}} = 2\sum_{i=1}^{n} \boldsymbol{B}(\boldsymbol{x}_i - \boldsymbol{\mu}).$$

Set the gradient to zero yields one solution

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i.$$

Solve for $\boldsymbol{A}$

It remains to solve

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{minimize}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top(\boldsymbol{x}_i - \boldsymbol{\mu}) - \boldsymbol{\mu}\|_2^2$$

- We can assume $\boldsymbol{\mu} = \mathbf{0}$ without loss of generality, as we can set $\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$ (removing the mean from the data).

The problem reduces to

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{minimize}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i\|_2^2$$

This is one form of PCA.

- Interpretation: Find the closest k-dimensional data point $\boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i$ to $\boldsymbol{x}_i$ (projecting $\boldsymbol{x}_i$ onto the k-dimensional subspace spanned by the columns of $\boldsymbol{A}$).

Derivation of the Second Equivalent PCA Form

Given the PCA formulation

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{minimize}} \ \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i\|_2^2.$$

Expanding the objective function yields

$$\begin{aligned}
&\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i\|_2^2 = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i)^\top (\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i) \\
&= \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{x}_i - 2\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i + \boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i \right) \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i.
\end{aligned}$$

We reduce to the second form of PCA

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \sum_{i=1}^{n} \boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i = \sum_{i=1}^{n} \|\boldsymbol{A}^\top \boldsymbol{x}_i\|_2^2.$$

Derivation of the Third Equivalent PCA Form

The second form is further equivalent to

$$\begin{aligned}
&\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \sum_{i=1}^{n} \|\boldsymbol{A}^\top \boldsymbol{x}_i\|_2^2 \\
\iff &\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \sum_{i=1}^{n} \operatorname{trace}(\boldsymbol{A}^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{A}) \\
\iff &\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{A}^\top \left(\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right) \boldsymbol{A}\right).
\end{aligned}$$

Let

$$\boldsymbol{S} = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top = \boldsymbol{X}\boldsymbol{X}^\top, \quad \text{where } \boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$$

be the empirical covariance matrix. We have the following third form of PCA

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{A}^\top \boldsymbol{S} \boldsymbol{A}\right).$$

We use the third form to derive the solution for $\boldsymbol{A}$. We need to consult a matrix computation tool called eigen decomposition.

Eigenvalue Analysis

Eigenvalue problem: Given matrix $\boldsymbol{S}$, find a vector $\boldsymbol{u}$ and a scaler λ such that

$$\boldsymbol{S}\boldsymbol{u} = \lambda \boldsymbol{u}$$

- λ characterizes the behavior of $\boldsymbol{S}$ in $\boldsymbol{u}$.
- $\boldsymbol{u}$ is called the eigenvector, while λ is called the eigenvalue.

Eigen decomposition for real PSD matrix

Suppose matrix $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ is real, symmetric, and positive semidefinite (PSD), it always admits an eigen decomposition:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$$

where $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ is an orthogonal matrix satisfying $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\top = \mathbf{I}$ containing eigenvectors and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$ is a diagonal matrix containing the corresponding eigenvalues.

Return to The Solution of PCA

Consider the third form of PCA

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{A}^\top \boldsymbol{S} \boldsymbol{A}\right),$$

where $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$.

- Note that $\boldsymbol{S}$ is constructed from the data matrix $\boldsymbol{X}$, it is computable.
- Further, $\boldsymbol{S}$ is real and must be PSD (why?).

Thus we apply eigen decomposition to $\boldsymbol{S}$ to obtain

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top.$$

The PCA problem becomes

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{A}^\top \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top \boldsymbol{A}\right).$$

Return to The Solution of PCA

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{A}^\top \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top \boldsymbol{A}\right)$$

Let $\boldsymbol{\Phi} = \boldsymbol{A}^\top \boldsymbol{U} \in \mathbb{R}^{k \times d}$, it is a semi-orthogonal matrix since $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \mathbf{I}$
We can rewrite the optimization problem as

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top\right) = \underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \operatorname{trace}\left(\sum_{i=1}^{d} \lambda_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top\right)$$

put the trace inside, we have

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I}}{\text{maximize}} \ \sum_{i=1}^{d} \lambda_i \boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i$$

Fact: This optimization problem has upper bound $\leq \sum_{i=1}^{k} \lambda_i$. Hence, it attains its maximum when $\boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i = 1, i = 1, \ldots, k$ and $\boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i = 0, i = k+1, \ldots, d$. This is achieved by

$$\boxed{\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]}$$

i.e., the first k eigenvectors.

Process of Computing The PCA

Given samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, without labels.

- Remove the mean

$$\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{\mu},$$

  where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$.

- Form the empirical covariance matrix from data

$$\boldsymbol{S} = \sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^\top = \boldsymbol{X}\boldsymbol{X}^\top.$$

- Compute the eigen decomposition of $\boldsymbol{S}$

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$$

  and the PCA solution is given by

$$\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k].$$

- Compute principle component and low-dimensional sample

$$\boldsymbol{\theta}_i = \boldsymbol{A}^\top \boldsymbol{x}_i, \quad \tilde{\boldsymbol{x}}_i = \boldsymbol{A}\boldsymbol{\theta}_i = \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}_i.$$

Connection to Singular Value Decomposition

Singular value decomposition (SVD)

Given any real matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$, there exists a 3-tuple $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \in \mathbb{R}^{d\times d} \times \mathbb{R}^{d\times n} \times \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal and $\boldsymbol{\Sigma}$ takes the form

$$\boldsymbol{\Sigma}(i,j) = \begin{cases} \sigma_i, & i = j \\ 0, & i \neq j \end{cases}$$

$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, $p = \min(d, n)$.

- σ_i are called singular value.
- $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are called left and right singular vectors.

PCA via SVD

Recall $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$, the eigen decomposition is $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, and the solution to the PCA problem is

$$\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$$

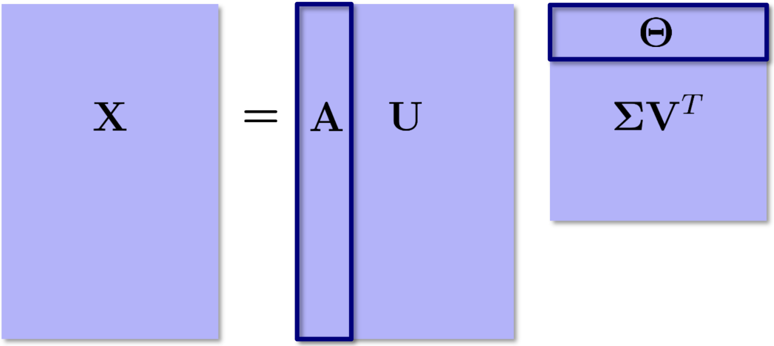
We can instead compute the SVD of matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and we also have

$$\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k].$$

- Computing PCA via SVD can be more favorable, as it can be more numerically reliable than eigen decomposition.
- It might also save computation time as we do not need to compute $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$, which can be expensive when $\boldsymbol{X}$ is a large matrix.

PCA via SVD: Illustration

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$



PCA From the Matrix Factorization Perspective

Recall the (first form) PCA (we assume without loss of generality that $\boldsymbol{\mu} = 0$)

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \{\boldsymbol{\theta}_i\}}{\text{minimize}} \ \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{\theta}_i\|_2^2.$$

It can be written in a matrix form:

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times k}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{k \times n}$, and $\|\cdot\|_F$ is the Frobenius norm.

- The above problem is also called low-rank matrix factorization.
- Interpretation: Factorize $\boldsymbol{X}$ into two factors' multiplication, where the latter is a low-rank matrix.

PCA From the Matrix Factorization Perspective

Low-rank matrix factorization

$$\underset{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I},\boldsymbol{\Theta}}{\text{minimize}} \; \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

- Calculate the SVD of $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$.

Solution from PCA

- One optimal solution to the above (low-rank) matrix factorization problem is given by

$$\boldsymbol{A} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k], \quad \boldsymbol{\Theta} = [\sigma_1 \boldsymbol{v}_1, \ldots, \sigma_k \boldsymbol{v}_k]^\top$$

- It has infinitely many equivalent optimal solutions.

It is a closed-form solution to a nonconvex optimization problem.

More General Matrix Factorization

We can also remove the orthogonal constraint on $\boldsymbol{A}$ to allow more flexibility

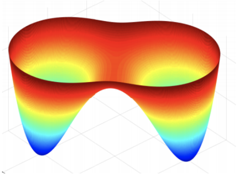
$$\underset{\boldsymbol{A}\in\mathbb{R}^{d\times k},\boldsymbol{\Theta}\in\mathbb{R}^{k\times n}}{\text{minimize}} \quad \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2.$$

One optimal solution to the above (low-rank) matrix factorization problem is given by

$$\boldsymbol{A} = [\sqrt{\sigma_1}\boldsymbol{u}_1, \ldots, \sqrt{\sigma_k}\boldsymbol{u}_k], \quad \boldsymbol{\Theta} = [\sqrt{\sigma}_1\boldsymbol{v}_1, \ldots, \sqrt{\sigma}_k\boldsymbol{v}_k]^\top,$$

and it has infinitely many equivalent optimal solution.

- A nonconvex optimization problem



- Fortunately, closed-form solution exists.

LoRA: Low-rank Adaptation

In the post-training stage of large models (like large language models), we often need to learn an incremental to the learned model to incorporate new knowledge. That is

$$\underset{\Delta\boldsymbol{\Theta}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad \mathcal{L}(\widehat{\boldsymbol{\Theta}} + \Delta\boldsymbol{\Theta})$$

Low-rank Adaptation (LoRA) uses the simple idea of PCA. It does not aim to learn the full $\Delta\boldsymbol{\Theta}$, it instead learns a PCA decomposition of $\Delta\boldsymbol{\Theta}$.

$$\underset{\boldsymbol{A}\in\mathbb{R}^{m\times r},\boldsymbol{B}\in\mathbb{R}^{r\times n}}{\text{minimize}} \quad \mathcal{L}(\widehat{\boldsymbol{\Theta}} + \boldsymbol{A}\boldsymbol{B})$$

where $r \ll \min\{m, n\}$.

- LoRA approach can save computation memory.
- It is now widely utilized in large language models.
- We will use LoRA in our final project.

Dimensionality Reduction

Principal Component Analysis (PCA)

Apply PCA to Real Image Dataset

The ORL Database of Faces

- 40 persons.
- Each has 10 distinct face images.
- Each image is of size 92×112.
- The images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses).

Form the Data Matrix

92x112

$$\xrightarrow{\text{vectorize}} \boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{x}_i[1] \\ \vdots \\ \boldsymbol{x}_i[92 \times 112] \end{bmatrix} \in \mathbb{R}^{10304}$$

- Do the same thing for all the face images, we get

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n},$$

where $d = 10304$, $n = 10 \times 40 = 400$.

Apply PCA to $\boldsymbol{X}$ with $k = 40$

Perform PCA, i.e., solving

$$\underset{\boldsymbol{A}^\top \boldsymbol{A} = \mathbf{I}, \boldsymbol{\Theta}}{\text{minimize}} \ \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{\Theta}\|_F^2$$

We get the extracted features (i,e., $\boldsymbol{A} \in \mathbb{R}^{d \times k}$)

where we resize each column of $\boldsymbol{A}$ (of dimension $d = 10304$) to a 92×112 image and show it.

Application

$$\boxed{\boldsymbol{x}_i \quad = \quad \boldsymbol{A}\boldsymbol{\theta}_i}$$

- Each face image $\boldsymbol{x}_i$ can be interpreted as a linear combination of the columns in $\boldsymbol{A}$.
- The importance of each feature is implied in $\boldsymbol{\theta}_i$.
- $\boldsymbol{x}_i \in \mathbb{R}^d$ ($d = 10304$ here) has been nicely represented by a k-dimensional vector $\boldsymbol{A}\boldsymbol{\theta}_i$ ($k = 40$ here) — Dimensionality reduction.

$\rightsquigarrow$ Next lecture: Unsupervised learning: Clustering and k-means.