CS559 Machine Learning
Fisher Linear Discriminant
Principal Component Analysis

Tian Han

Department of Computer Science
Stevens Institute of Technology

Week 6

Outline

- Fisher's Linear Discriminant
- Principal Component Analysis
    - High-dimensional data, Eigenfaces
    - PCA vs FLD

Fisher's Linear Discriminant

Binary Classification

In previous lecture, we study **linear discriminant function** and use it to solve the binary classification problem.
Specifically, we have:

$$y = \begin{cases} +1, & \text{if } \mathbf{w}^T\mathbf{x} > \theta \\ -1, & \text{if } \mathbf{w}^T\mathbf{x} < \theta \end{cases}$$

We learn the model $\mathbf{w}$ using *perceptron algorithm* which is shown to be effective if the underlying two classes are *linearly separable*.

Binary Classification

In previous lecture, we study **linear discriminant function** and use it to solve the binary classification problem.
Specifically, we have:

$$y = \begin{cases} +1, & \text{if } \mathbf{w}^T\mathbf{x} > \theta \\ -1, & \text{if } \mathbf{w}^T\mathbf{x} < \theta \end{cases}$$

We learn the model $\mathbf{w}$ using *perceptron algorithm* which is shown to be effective if the underlying two classes are *linearly separable*.

We could view classification in another way......

Classification through projection

- A linear function: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ assuming in 2D, projects each point $\mathbf{x} = [x_1, x_2]^T$ to a line parallel to $\mathbf{w}$:

point in $\mathcal{R}^D$	projected point in $\mathcal{R}$
$\mathbf{x}_1$	$z_1 = \mathbf{w}^T\mathbf{x}_1$
$\mathbf{x}_2$	$z_2 = \mathbf{w}^T\mathbf{x}_2$
...	...
$\mathbf{x}_n$	$z_n = \mathbf{w}^T\mathbf{x}_n$

- We can study how well the projected points $z_1, ..., z_n$, viewed as functions of $\mathbf{w}$, are separated across the classes.

Classification through projection

- A linear function: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ assuming in 2D, projects each point $\mathbf{x} = [x_1, x_2]^T$ to a line parallel to $\mathbf{w}$:

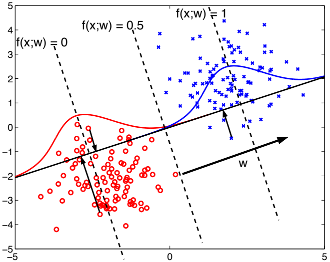

Figure: [N.Yue, CS559 S19]

- We can study how well the projected points $z_1, ..., z_n$, viewed as functions of $\mathbf{w}$, are separated across the classes.

Classification through projection

- By varying $\mathbf{w}$ we get different levels of separation between the projected points

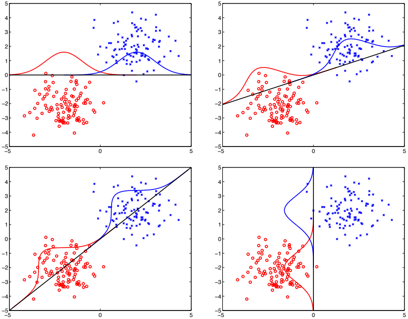

Figure: [N.Yue, CS559 S19]

Find the good projection

- We would like to find $\mathbf{w}$ that somehow maximizes the separation of the projected points across classes.

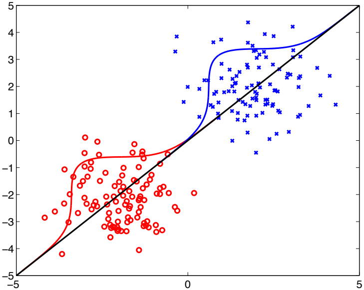

Figure: [N.Yue, CS559 S19]

Find the good projection

- We would like to find $\mathbf{w}$ that somehow maximizes the separation of the projected points across classes.

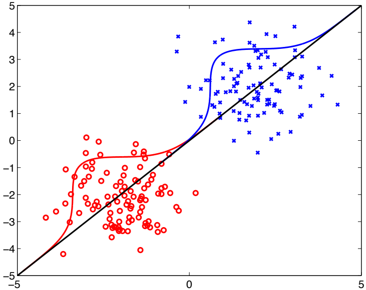

Figure: [N.Yue, CS559 S19]

- We can quantify the separation (overlap) in terms of **means** and **variances** of the resulting 1-dimensional class distributions

Fisher's Linear Discriminant

- Through projection, linear classification can be viewed in terms of dimensional reduction.

Fisher's Linear Discriminant

- Through projection, linear classification can be viewed in terms of dimensional reduction.
- Two class case: learning the linear model to project $\mathbf{x}$ into one-dimension:

$$y = \mathbf{w}^T\mathbf{x}$$

Fisher's Linear Discriminant

- Through projection, linear classification can be viewed in terms of dimensional reduction.
- Two class case: learning the linear model to project $\mathbf{x}$ into one-dimension:

$$y = \mathbf{w}^T \mathbf{x}$$

- Decision stage: select a proper threshold y_0, then

$$\begin{array}{ll} \text{if } y \geq y_0 & \text{assign } \mathbf{x} \text{ to } C_1 \\ \text{otherwise} & \text{assign } \mathbf{x} \text{ to } C_2 \end{array}$$

Fisher's Linear Discriminant

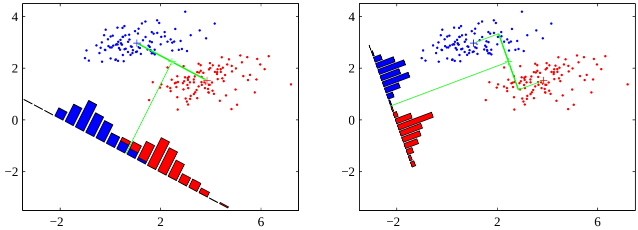

Figure: [C.Bishop PRML]

- Find an direction along which the projected samples are well separated;

Fisher's Linear Discriminant

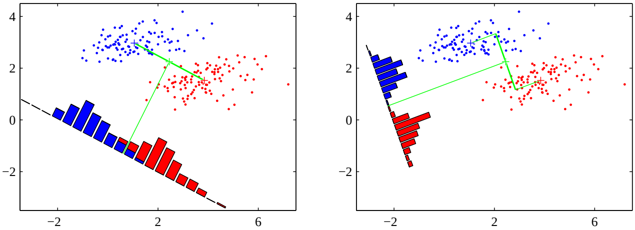

Figure: [C.Bishop PRML]

- Find an direction along which the projected samples are well separated;
- We are looking for the linear projection that best separates the data, i.e. best discriminates data of different classes.

Basic setting

- Two classes: C_1 and C_2.

Basic setting

- Two classes: C_1 and C_2.
- N_1 data points in C_1, N_2 points in C_2.

Basic setting

- Two classes: C_1 and C_2.
- N_1 data points in C_1, N_2 points in C_2.
- C_1 class: mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$ and covariance Σ_1.

Basic setting

- Two classes: C_1 and C_2.
- N_1 data points in C_1, N_2 points in C_2.
- C_1 class: mean $\mu_1 = \frac{1}{N_1}\sum_{\mathbf{x}\in C_1}\mathbf{x}$ and covariance Σ_1.
- C_2 class: mean $\mu_2 = \frac{1}{N_2}\sum_{\mathbf{x}\in C_2}\mathbf{x}$ and covariance Σ_2.

Basic setting

- Two classes: C_1 and C_2.
- N_1 data points in C_1, N_2 points in C_2.
- C_1 class: mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$ and covariance Σ_1.
- C_2 class: mean $\mu_2 = \frac{1}{N_2} \sum_{\mathbf{x} \in C_2} \mathbf{x}$ and covariance Σ_2.
- Assume $||\mathbf{w}|| = 1$, then $\mathbf{w}^T \mathbf{x}$ is the projection of $\mathbf{x}$ onto $\mathbf{w}$.

Two goals

After projection, in order to obtain the best separation of the data, we need to satisfy two goals:

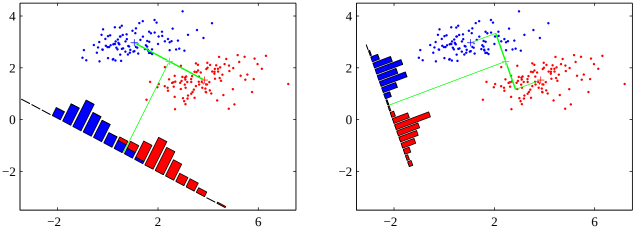

Figure: [C.Bishop PRML]

Two goals

After projection, in order to obtain the best separation of the data, we need to satisfy two goals:

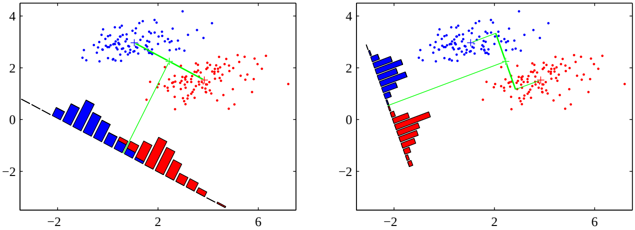

Figure: [C.Bishop PRML]

- **Maximize the distance between projected means.**

Two goals

After projection, in order to obtain the best separation of the data, we need to satisfy two goals:

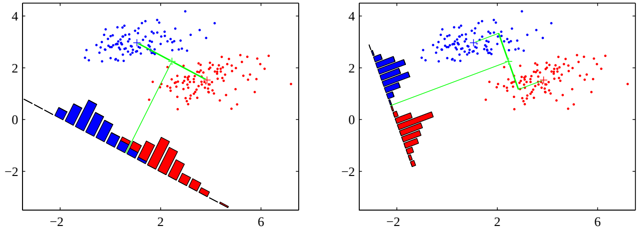

Figure: [C.Bishop PRML]

- **Maximize the distance between projected means.**
- **Minimize the variance of each class.**

Maximize the distance between projected means

- C_1: sample mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$.

Maximize the distance between projected means

- C_1: sample mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$.
- Mean for projected C_1 points: $\frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_1$

Maximize the distance between projected means

- C_1: sample mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$.
- Mean for projected C_1 points: $\frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_1$
- Similarly, mean for projected C_2 points: $\frac{1}{N_2} \sum_{\mathbf{x} \in C_2} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_2$

Maximize the distance between projected means

- C_1: sample mean $\mu_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x}$.
- Mean for projected C_1 points: $\frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_1$
- Similarly, mean for projected C_2 points: $\frac{1}{N_2} \sum_{\mathbf{x} \in C_2} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_2$
- Maximize the distance between these two projected means:

$$\max_{\mathbf{w}} (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2$$

Minimize the variance of each class

- C_1: covariance Σ_1.

Minimize the variance of each class

- C_1: covariance Σ_1.
- Q: if we know the covariance of $\mathbf{x}$, what is the variance of $\mathbf{w}^T\mathbf{x}$?

Minimize the variance of each class

- C_1: covariance Σ_1.
- Q: if we know the covariance of $\mathbf{x}$, what is the variance of $\mathbf{w}^T\mathbf{x}$?
- The variance of projected points in C_1 would be:

$$\mathrm{Cov}(\mathbf{w}^T\mathbf{x}) = \mathbf{w}^T\mathrm{Cov}(\mathbf{x})\mathbf{w} = \mathbf{w}^T\Sigma_1\mathbf{w}$$

Minimize the variance of each class

- C_1: covariance Σ_1.
- Q: if we know the covariance of $\mathbf{x}$, what is the variance of $\mathbf{w}^T\mathbf{x}$?
- The variance of projected points in C_1 would be:

$$\mathrm{Cov}(\mathbf{w}^T\mathbf{x}) = \mathbf{w}^T\mathrm{Cov}(\mathbf{x})\mathbf{w} = \mathbf{w}^T\Sigma_1\mathbf{w}$$

- Similarly, the variance of projected points in C_2 would be: $\mathbf{w}^T\Sigma_2\mathbf{w}$

Minimize the variance of each class

- C_1: covariance Σ_1.
- Q: if we know the covariance of $\mathbf{x}$, what is the variance of $\mathbf{w}^T\mathbf{x}$?
- The variance of projected points in C_1 would be:

$$\mathrm{Cov}(\mathbf{w}^T\mathbf{x}) = \mathbf{w}^T\mathrm{Cov}(\mathbf{x})\mathbf{w} = \mathbf{w}^T\Sigma_1\mathbf{w}$$

- Similarly, the variance of projected points in C_2 would be: $\mathbf{w}^T\Sigma_2\mathbf{w}$
- Minimize the variance of each class:

$$\min_{\mathbf{w}}(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})$$

Combine two goals

- Maximize the distance between projected means.

$$\max_{\mathbf{w}} (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2$$

Combine two goals

- Maximize the distance between projected means.

$$\max_{\mathbf{w}}(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2$$

- Minimize the variance of each class.

$$\min_{\mathbf{w}}(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})$$

Combine two goals

- Maximize the distance between projected means.

$$\max_{\mathbf{w}} (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2$$

- Minimize the variance of each class.

$$\min_{\mathbf{w}} (\mathbf{w}^T \Sigma_1 \mathbf{w} + \mathbf{w}^T \Sigma_2 \mathbf{w})$$

- How to combine these two?

Combine two goals

- Maximize the distance between projected means.

$$\max_{\mathbf{w}}(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2$$

- Minimize the variance of each class.

$$\min_{\mathbf{w}}(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})$$

- How to combine these two?

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

Optimization for $\mathbf{w}$

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

Optimization for $\mathbf{w}$

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

- Manipulate the expression to get the optimal $\mathbf{w}$.

Optimization for $\mathbf{w}$

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

- Manipulate the expression to get the optimal $\mathbf{w}$.
- A more general way....
  - $(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2 = \mathbf{w}^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\mathbf{w} = \mathbf{w}^T S_B\mathbf{w}$ (S_B: **between class covariance matrix**)

Optimization for $\mathbf{w}$

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

- Manipulate the expression to get the optimal $\mathbf{w}$.
- A more general way....
    - $(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2 = \mathbf{w}^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\mathbf{w} = \mathbf{w}^T S_B \mathbf{w}$ (S_B: **between class covariance matrix**)
    - $\Sigma_1 + \Sigma_2 = S_W$. ($S_W$: **within class covariance matrix**)

Optimization for $\mathbf{w}$

$$\max_{\mathbf{w}} \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{(\mathbf{w}^T\Sigma_1\mathbf{w} + \mathbf{w}^T\Sigma_2\mathbf{w})}$$

- Manipulate the expression to get the optimal $\mathbf{w}$.
- A more general way....
  - $(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2 = \mathbf{w}^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\mathbf{w} = \mathbf{w}^T S_B \mathbf{w}$ (S_B: **between class covariance matrix**)
  - $\Sigma_1 + \Sigma_2 = S_W$. ($S_W$: **within class covariance matrix**)
  - More compactly, we have:

$$\max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

Optimization for $\mathbf{w}$

Transform to constrained optimization problem:

$$\begin{aligned} & \max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \\ \rightarrow \quad & \max_{\mathbf{w}} \mathbf{w}^T S_B \mathbf{w}, \text{ s.t. } \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

Optimization for $\mathbf{w}$

Transform to constrained optimization problem:

$$\begin{aligned} & \max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \\ \rightarrow \quad & \max_{\mathbf{w}} \mathbf{w}^T S_B \mathbf{w}, \text{ s.t. } \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

Using Lagrange Multiplier:

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T S_B \mathbf{w} - \lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

Optimization for $\mathbf{w}$

Transform to constrained optimization problem:

$$\begin{aligned} & \max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \\ \rightarrow \quad & \max_{\mathbf{w}} \mathbf{w}^T S_B \mathbf{w}, \text{ s.t. } \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

Using Lagrange Multiplier:

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T S_B \mathbf{w} - \lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

Take $\frac{\partial L(\mathbf{w},\lambda)}{\partial \mathbf{w}} = 0$, we have the *generalized eigenvalue problem*:

$$\begin{aligned} S_B \mathbf{w} &= \lambda S_W \mathbf{w} \\ S_W^{-1} S_B \mathbf{w} &= \lambda \mathbf{w} \end{aligned}$$

Optimal $\mathbf{w}$

We have:

$$\begin{aligned} S_W^{-1} S_B \mathbf{w} &= \lambda \mathbf{w} \\ S_W^{-1}(\mu_1 - \mu_2) \underbrace{(\mu_1 - \mu_2)^T \mathbf{w}}_{\text{scalar}} &= \lambda \mathbf{w} \end{aligned}$$

Which means:

$$\mathbf{w} \propto S_W^{-1}(\mu_1 - \mu_2)$$

Optimal $\mathbf{w}$

We have:

$$\begin{aligned} S_W^{-1} S_B \mathbf{w} &= \lambda \mathbf{w} \\ S_W^{-1}(\mu_1 - \mu_2)\underbrace{(\mu_1 - \mu_2)^T \mathbf{w}}_{\text{scalar}} &= \lambda \mathbf{w} \end{aligned}$$

Which means:

$$\mathbf{w} \propto S_W^{-1}(\mu_1 - \mu_2)$$

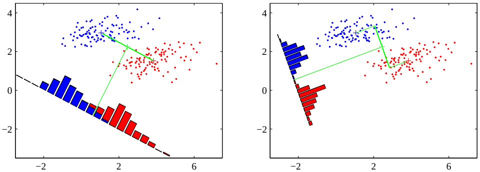

Figure: [C.Bishop PRML]

FLD summary

FLD procedure for two class classification:

Input: N_1 points from C_1 and N_2 points from C_2.

1. Compute sample means for each class, i.e., μ_1, μ_2, and compute the S_W.
2. Compute $\mathbf{w} \leftarrow S_W^{-1}(\mu_1 - \mu_2)$
3. Normalize: $\mathbf{w} \leftarrow \frac{\mathbf{w}}{||\mathbf{w}||}$
4. Select a suitable threshold θ, then classify $\mathbf{x}$ to be C_1 if $\mathbf{w}^T\mathbf{x} > \theta$, otherwise, classify it to be C_2.

Note:

- In the literature, people always use within class scatter matrix for S_W, which is:

$$S_W = \sum_{\mathbf{x} \in C_1} (\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^T + \sum_{\mathbf{x} \in C_2} (\mathbf{x} - \mu_2)(\mathbf{x} - \mu_2)^T$$

FLD summary

- Fisher's linear discriminant can be generalized to multiple classes.

FLD summary

- Fisher's linear discriminant can be generalized to multiple classes.
- If number of samples N is greater than data dimensionality D, i.e., $N > D$, S_W is very likely to be invertible.

FLD summary

- Fisher's linear discriminant can be generalized to multiple classes.
- If number of samples N is greater than data dimensionality D, i.e., $N > D$, S_W is very likely to be invertible.
- However, when $D > N$, e.g., in vision problems like face recognition, S_W is singular. Could use pseudo-inverse or *Fisher face* algorithm.

FLD summary

- Fisher's linear discriminant can be generalized to multiple classes.
- If number of samples N is greater than data dimensionality D, i.e., $N > D$, S_W is very likely to be invertible.
- However, when $D > N$, e.g., in vision problems like face recognition, S_W is singular. Could use pseudo-inverse or *Fisher face* algorithm.
- Fisher's linear discriminant can be used to do dimension reduction. The found $\mathbf{w}$ can be treated as new axis. Previous $\mathcal{R}^D$ problem becomes one dimensional.

FLD summary

- Fisher's linear discriminant can be generalized to multiple classes.
- If number of samples N is greater than data dimensionality D, i.e., $N > D$, S_W is very likely to be invertible.
- However, when $D > N$, e.g., in vision problems like face recognition, S_W is singular. Could use pseudo-inverse or *Fisher face* algorithm.
- Fisher's linear discriminant can be used to do dimension reduction. The found $\mathbf{w}$ can be treated as new axis. Previous $\mathcal{R}^D$ problem becomes one dimensional.
- What about other **dimensionality reduction** techniques? What if we do not have class label information?

Principal Component Analysis

Dimensionality Reduction

- In many applications, the observed data has very high dimensionality, e.g., images, videos, DNA sequences...

Dimensionality Reduction

- In many applications, the observed data has very high dimensionality, e.g., images, videos, DNA sequences...
- **Assumption:** the data points lie close to a subspace of much *lower dimensionality* than that of the original data space.

Low-dimensional subspace

- E.g., the points in 3D space may form a line or plane.

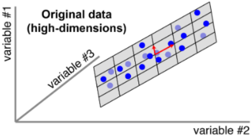

Figure: From Source

Low-dimensional subspace

- E.g., the points in 3D space may form a line or plane.

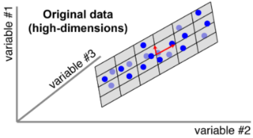

Figure: From Source

- E.g., a small 100×100 gray-scaled image has $10,000$ dimensions! While the intrinsic dimensionality might be low.

Principal Component Analysis

- Principal Component Analysis, or PCA, is a widely used technique for dimensionality reduction which transform the data from D-dimensional space into a new coordinate system of dimension K, where $K \leq D$.

Principal Component Analysis

- Principal Component Analysis, or PCA, is a widely used technique for dimensionality reduction which transform the data from D-dimensional space into a new coordinate system of dimension K, where $K \leq D$.
- PCA can be extremely useful since it can find features based on the *hidden structure* of the data.

Principal Component Analysis

- Principal Component Analysis, or PCA, is a widely used technique for dimensionality reduction which transform the data from D-dimensional space into a new coordinate system of dimension K, where $K \leq D$.
- PCA can be extremely useful since it can find features based on the *hidden structure* of the data.
- PCA aims to find a new representation (coordinate system) of the data by extracting combinations of features (components) from the data that are uncorrelated with each other and ordered by *importance*.

Principal Component Analysis

- Principal Component Analysis, or PCA, is a widely used technique for dimensionality reduction which transform the data from D-dimensional space into a new coordinate system of dimension K, where $K \leq D$.
- PCA can be extremely useful since it can find features based on the *hidden structure* of the data.
- PCA aims to find a new representation (coordinate system) of the data by extracting combinations of features (components) from the data that are uncorrelated with each other and ordered by *importance*.
- PCA allows us to approximate the data with a reduced number of features (i.e., only keep the very significant ones).

Principal Component Analysis

- Principal Component Analysis, or PCA, is a widely used technique for dimensionality reduction which transform the data from D-dimensional space into a new coordinate system of dimension K, where $K \leq D$.
- PCA can be extremely useful since it can find features based on the *hidden structure* of the data.
- PCA aims to find a new representation (coordinate system) of the data by extracting combinations of features (components) from the data that are uncorrelated with each other and ordered by *importance*.
- PCA allows us to approximate the data with a reduced number of features (i.e., only keep the very significant ones).
- Useful tools in many applications such as face recognition, data compression, feature extraction etc.

Principal Component Analysis

- **Goal:** preserve as much of the information (variance) in the original data as possible in the new coordinate system.

Principal Component Analysis

- **Goal:** preserve as much of the information (variance) in the original data as possible in the new coordinate system.
- The new variables that form a new coordinate system are called **principal components** (PCs). PCs are orthogonal (uncorrelated) to each other.

Principal Component Analysis

- **Goal:** preserve as much of the information (variance) in the original data as possible in the new coordinate system.
- The new variables that form a new coordinate system are called **principal components** (PCs). PCs are orthogonal (uncorrelated) to each other.
- For D-dimensional data, we have at most D PCs, denoted as $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$.

Principal Component Analysis

- **Goal:** preserve as much of the information (variance) in the original data as possible in the new coordinate system.
- The new variables that form a new coordinate system are called **principal components** (PCs). PCs are orthogonal (uncorrelated) to each other.
- For D-dimensional data, we have at most D PCs, denoted as $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$.
- Normally, not all D PCs are used but rather a subset of K "most important" PCs, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$.

Principal Component Analysis

- **Goal:** preserve as much of the information (variance) in the original data as possible in the new coordinate system.
- The new variables that form a new coordinate system are called **principal components** (PCs). PCs are orthogonal (uncorrelated) to each other.
- For D-dimensional data, we have at most D PCs, denoted as $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$.
- Normally, not all D PCs are used but rather a subset of K "most important" PCs, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$.
- **Key assumption**: the direction within the data that shows the *most variance* contains the *most information* and therefore, is likely the most important.

First principal component

- Consider the projection of original data $\mathbf{x}$ onto a one-dimensional space, define the direction of this space using a D-dimensional vector $\mathbf{u}_1$.

First principal component

- Consider the projection of original data $\mathbf{x}$ onto a one-dimensional space, define the direction of this space using a D-dimensional vector $\mathbf{u}_1$.
- The scale of $\mathbf{u}_1$ doesn't matter, care more about direction, assume $\mathbf{u}_1^T\mathbf{u}_1 = 1$.

First principal component

- Consider the projection of original data $\mathbf{x}$ onto a one-dimensional space, define the direction of this space using a D-dimensional vector $\mathbf{u}_1$.
- The scale of $\mathbf{u}_1$ doesn't matter, care more about direction, assume $\mathbf{u}_1^T \mathbf{u}_1 = 1$.
- Each data point $\mathbf{x}_n$ is projected onto a scalar value $\mathbf{u}_1^T \mathbf{x}_n$

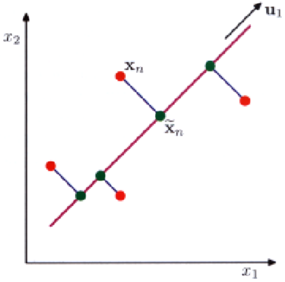

Figure: [C.Bishop, PRML]

Maximize the variance

- Suppose we have N observations $\{\mathbf{x}_n, n = 1, 2, \ldots, N\}$, $\mathbf{x}_n \in \mathcal{R}^D$.

Maximize the variance

- Suppose we have N observations $\{\mathbf{x}_n, n = 1, 2, \ldots, N\}$, $\mathbf{x}_n \in \mathcal{R}^D$.
- The projected data: $\{\mathbf{u}_1^T \mathbf{x}_n, n = 1, 2, \ldots, N\}$

Maximize the variance

- Suppose we have N observations $\{\mathbf{x}_n, n = 1, 2, \ldots, N\}$, $\mathbf{x}_n \in \mathcal{R}^D$.
- The projected data: $\{\mathbf{u}_1^T \mathbf{x}_n, n = 1, 2, \ldots, N\}$
- The mean of the projected data:

$$\frac{1}{N} \sum_{n=1}^{N} \mathbf{u}_1^T \mathbf{x}_n = \mathbf{u}_1^T \bar{\mathbf{x}}$$

Maximize the variance

- Suppose we have N observations $\{\mathbf{x}_n, n = 1, 2, \ldots, N\}$, $\mathbf{x}_n \in \mathcal{R}^D$.
- The projected data: $\{\mathbf{u}_1^T \mathbf{x}_n, n = 1, 2, \ldots, N\}$
- The mean of the projected data:

$$\frac{1}{N} \sum_{n=1}^{N} \mathbf{u}_1^T \mathbf{x}_n = \mathbf{u}_1^T \bar{\mathbf{x}}$$

- The variance of the projected data:

$$\frac{1}{N} \sum_{n=1}^{N} [\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}]^2 = \mathbf{u}_1^T S \mathbf{u}_1$$

$$S = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

Maximize the variance

Find $\mathbf{u}_1$, such that the projected data has **maximum variance**.

Maximize the variance

Find $\mathbf{u}_1$, such that the projected data has **maximum variance**.

- $\max_{\mathbf{u}_1} \mathbf{u}_1^T S \mathbf{u}_1$: would render trivial solution $||\mathbf{u}_1|| \to \infty$.

Maximize the variance

Find $\mathbf{u}_1$, such that the projected data has **maximum variance**.

- $\max_{\mathbf{u}_1} \mathbf{u}_1^T S \mathbf{u}_1$: would render trivial solution $||\mathbf{u}_1|| \to \infty$.
- Consider normalization condition:

$$\begin{array}{ll} \max_{\mathbf{u}_1} & \mathbf{u}_1^T S \mathbf{u}_1 \\ \text{s.t.} & \mathbf{u}_1^T \mathbf{u}_1 = 1 \end{array}$$

Maximize the variance

Find $\mathbf{u}_1$, such that the projected data has **maximum variance**.

- $\max_{\mathbf{u}_1} \mathbf{u}_1^T S \mathbf{u}_1$: would render trivial solution $||\mathbf{u}_1|| \to \infty$.
- Consider normalization condition:

$$\begin{array}{ll} \max_{\mathbf{u}_1} & \mathbf{u}_1^T S \mathbf{u}_1 \\ \text{s.t.} & \mathbf{u}_1^T \mathbf{u}_1 = 1 \end{array}$$

- Using Lagrange Multiplier:

$$L(\mathbf{u}_1, \lambda) = \mathbf{u}_1^T S \mathbf{u}_1 - \lambda_1(\mathbf{u}_1^T \mathbf{u}_1 - 1)$$

Optimization for $\mathbf{u}_1$

Take $\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 0$, we have:

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

Optimization for $\mathbf{u}_1$

Take $\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 0$, we have:

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- $\mathbf{u}_1$ must be eigenvector of S. λ_1 is the eigenvalue of S.

Optimization for $\mathbf{u}_1$

Take $\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 0$, we have:

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- $\mathbf{u}_1$ must be eigenvector of S. λ_1 is the eigenvalue of S.
- Left-multiply by $\mathbf{u}_1^T$, and use $\mathbf{u}_1^T \mathbf{u}_1 = 1$, we have:

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

Optimization for $\mathbf{u}_1$

Take $\frac{\partial L(\mathbf{u}_1,\lambda_1)}{\partial \mathbf{u}_1} = 0$, we have:

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- $\mathbf{u}_1$ must be eigenvector of S. λ_1 is the eigenvalue of S.
- Left-multiply by $\mathbf{u}_1^T$, and use $\mathbf{u}_1^T\mathbf{u}_1 = 1$, we have:

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

- The variance will be **maximum** when we take $\mathbf{u}_1$ equal to the eigenvector that having the **largest eigenvalue** λ_1.

Optimization for $\mathbf{u}_1$

Take $\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 0$, we have:

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- $\mathbf{u}_1$ must be eigenvector of S. λ_1 is the eigenvalue of S.
- Left-multiply by $\mathbf{u}_1^T$, and use $\mathbf{u}_1^T \mathbf{u}_1 = 1$, we have:

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

- The variance will be **maximum** when we take $\mathbf{u}_1$ equal to the eigenvector that having the **largest eigenvalue** λ_1.
- This eigenvector $\mathbf{u}_1$ is known as the first principal component.

Alternative view

- Centering the data to have mean 0: $\mathbf{x}_n - \bar{\mathbf{x}}$ (simplify the notation, PCA is more like rotation operation)

Alternative view

- Centering the data to have mean 0: $\mathbf{x}_n - \bar{\mathbf{x}}$ (simplify the notation, PCA is more like rotation operation)
- For unit vector $\mathbf{u}_1$, the **residual of projection** for any data $\mathbf{x}_n$:

$$||\mathbf{x}_n - (\mathbf{u}_1^T \mathbf{x}_n)\mathbf{u}_1||^2$$

Alternative view

- Centering the data to have mean 0: $\mathbf{x}_n - \bar{\mathbf{x}}$ (simplify the notation, PCA is more like rotation operation)
- For unit vector $\mathbf{u}_1$, the **residual of projection** for any data $\mathbf{x}_n$:

$$||\mathbf{x}_n - (\mathbf{u}_1^T \mathbf{x}_n)\mathbf{u}_1||^2$$

- Consider N points, we have **Residual-Sum-of-Square (RSS)**, it can be shown that for centred data :

$$\min_{\mathbf{u}_1} RSS(\mathbf{u}_1) \equiv \max_{\mathbf{u}_1} \mathrm{Var}(\mathbf{u}_1^T \mathbf{x})$$

Alternative view

- Centering the data to have mean 0: $\mathbf{x}_n - \bar{\mathbf{x}}$ (simplify the notation, PCA is more like rotation operation)
- For unit vector $\mathbf{u}_1$, the **residual of projection** for any data $\mathbf{x}_n$:

$$||\mathbf{x}_n - (\mathbf{u}_1^T \mathbf{x}_n)\mathbf{u}_1||^2$$

- Consider N points, we have **Residual-Sum-of-Square (RSS)**, it can be shown that for centred data :

$$\min_{\mathbf{u}_1} RSS(\mathbf{u}_1) \equiv \max_{\mathbf{u}_1} \mathrm{Var}(\mathbf{u}_1^T \mathbf{x})$$

- Find $\mathbf{u}_1$, such that the **projection error is minimized**.

Alternative view

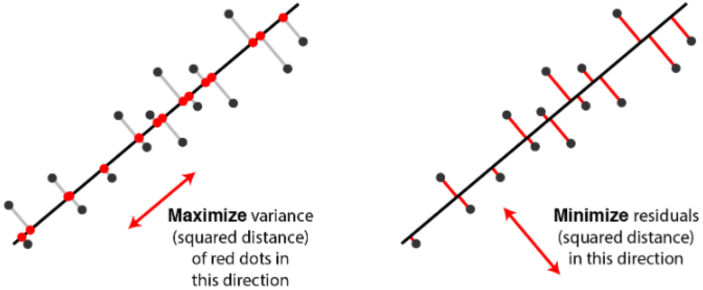

Figure: [Source]

Finding additional principal components

- To find second principal component $\mathbf{u}_2$: find direction that **maximize** the projected variance amongst all possible directions **orthogonal** to $\mathbf{u}_1$.
  - $\mathbf{u}_2$ is the eigenvector of S having the second largest eigenvalue λ_2.

Finding additional principal components

- To find second principal component $\mathbf{u}_2$: find direction that **maximize** the projected variance amongst all possible directions **orthogonal** to $\mathbf{u}_1$.
  - $\mathbf{u}_2$ is the eigenvector of S having the second largest eigenvalue λ_2.
- To find third principal component $\mathbf{u}_3$: find direction that **maximize** the projected variance amongst all possible directions **orthogonal** to $\mathbf{u}_1$ and $\mathbf{u}_2$.
  - $\mathbf{u}_3$ is the eigenvector of S having the third largest eigenvalue λ_3.

Finding additional principal components

- To find second principal component $\mathbf{u}_2$: find direction that **maximize** the projected variance amongst all possible directions **orthogonal** to $\mathbf{u}_1$.
  - $\mathbf{u}_2$ is the eigenvector of S having the second largest eigenvalue λ_2.
- To find third principal component $\mathbf{u}_3$: find direction that **maximize** the projected variance amongst all possible directions **orthogonal** to $\mathbf{u}_1$ and $\mathbf{u}_2$.
  - $\mathbf{u}_3$ is the eigenvector of S having the third largest eigenvalue λ_3.
- In general, we get D eigenvalues of covariance matrix S, ordered from largest to smallest:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

Then the corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$ are **principal components**.

Dimension reduction

The principal components (PCs) $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$ form the new coordinate system, so the original data $\mathbf{x}_n$ can be projected onto new system:

$$\mathbf{x}_n = \sum_{i=1}^{D} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i$$

Dimension reduction

The principal components (PCs) $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$ form the new coordinate system, so the original data $\mathbf{x}_n$ can be projected onto new system:

$$\mathbf{x}_n = \sum_{i=1}^{D} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i$$

- If we keep all D PCs, then there is *no dimensionality reduction* but simply a *rotation* of the coordinate axes to align with the principal components.

Dimension reduction

The principal components (PCs) $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D$ form the new coordinate system, so the original data $\mathbf{x}_n$ can be projected onto new system:

$$\mathbf{x}_n = \sum_{i=1}^{D} (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i$$

- If we keep all D PCs, then there is *no dimensionality reduction* but simply a *rotation* of the coordinate axes to align with the principal components.
- If keep $K(< D)$ PCs, then we have *lower dimensional* representation, and we could approximate $\mathbf{x}_n$ by:

$$\mathbf{x}_n \approx \sum_{i=1}^{K} (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i$$

PCA for High-dimensional Data

High-dimensional data

- In modern AI and vision applications, the number of data points is smaller than the dimensionality of the data space, i.e., $N < D$.

High-dimensional data

- In modern AI and vision applications, the number of data points is smaller than the dimensionality of the data space, i.e., $N < D$.
- E.g., we might have only 100 gray-scaled images of size 256×256 where $N = 100$, $D = 256 \times 256 = 65,536$.

High-dimensional data

- In modern AI and vision applications, the number of data points is smaller than the dimensionality of the data space, i.e., $N < D$.
- E.g., we might have only 100 gray-scaled images of size 256×256 where $N = 100$, $D = 256 \times 256 = 65,536$.
- N points in a D-dimensional space defines a subspace whose dimensionality is at most $N - 1$, therefore, apply PCA, we will find at least $D - N + 1$ of the eigenvalues to be 0.

High-dimensional data

- In modern AI and vision applications, the number of data points is smaller than the dimensionality of the data space, i.e., $N < D$.
- E.g., we might have only 100 gray-scaled images of size 256×256 where $N = 100$, $D = 256 \times 256 = 65,536$.
- N points in a D-dimensional space defines a subspace whose dimensionality is at most $N - 1$, therefore, apply PCA, we will find at least $D - N + 1$ of the eigenvalues to be 0.
- Further, covariance matrix S is $D \times D$, computational infeasible when D is large.

Eigenfaces

- Consider N face images of size $[h, w]$. In order to apply PCA, we view the data as long vector of size $D = h \times w$.

Eigenfaces

- Consider N face images of size $[h, w]$. In order to apply PCA, we view the data as long vector of size $D = h \times w$.
- Define $(N \times D)$ data matrix $\mathbf{X}$ where n-th row is given by $(\mathbf{x}_n - \bar{\mathbf{x}})^T$ which is the centred face image. $\bar{\mathbf{x}}$ is the mean image.

Eigenfaces

- Consider N face images of size $[h, w]$. In order to apply PCA, we view the data as long vector of size $D = h \times w$.
- Define $(N \times D)$ data matrix $\mathbf{X}$ where n-th row is given by $(\mathbf{x}_n - \bar{\mathbf{x}})^T$ which is the centred face image. $\bar{\mathbf{x}}$ is the mean image.
- The covariance matrix defined before can be written as $S = N^{-1}\mathbf{X}^T\mathbf{X}$.

Eigenfaces

- Consider N face images of size $[h, w]$. In order to apply PCA, we view the data as long vector of size $D = h \times w$.
- Define $(N \times D)$ data matrix $\mathbf{X}$ where n-th row is given by $(\mathbf{x}_n - \bar{\mathbf{x}})^T$ which is the centred face image. $\bar{\mathbf{x}}$ is the mean image.
- The covariance matrix defined before can be written as $S = N^{-1}\mathbf{X}^T\mathbf{X}$.
- The eigenvector equation for PCA:

$$\frac{1}{N} \underbrace{\mathbf{X}^T\mathbf{X}}_{D \times D} \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Eigenfaces

- Consider N face images of size $[h, w]$. In order to apply PCA, we view the data as long vector of size $D = h \times w$.
- Define $(N \times D)$ data matrix $\mathbf{X}$ where n-th row is given by $(\mathbf{x}_n - \bar{\mathbf{x}})^T$ which is the centred face image. $\bar{\mathbf{x}}$ is the mean image.
- The covariance matrix defined before can be written as $S = N^{-1}\mathbf{X}^T\mathbf{X}$.
- The eigenvector equation for PCA:

$$\frac{1}{N}\underbrace{\mathbf{X}^T\mathbf{X}}_{D\times D}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

- The eigenvectors of S, i.e., $\mathbf{u}_1, \mathbf{u}_2, \ldots$ forms the **eigenfaces**.

Eigenfaces

- The direct way for PCA:

$$\frac{1}{N}\underbrace{\mathbf{X}^T\mathbf{X}}_{D\times D}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

- Multiply by $\mathbf{X}$:

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{u}_i) = \lambda_i(\mathbf{X}\mathbf{u}_i)$$

Eigenfaces

- The direct way for PCA:

$$\frac{1}{N}\underbrace{\mathbf{X}^T\mathbf{X}}_{D\times D}\mathbf{u}_i = \lambda_i\mathbf{u}_i$$

- Multiply by $\mathbf{X}$:

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{u}_i) = \lambda_i(\mathbf{X}\mathbf{u}_i)$$

- Let $\mathbf{v}_i = \mathbf{X}\mathbf{u}_i$:

$$\frac{1}{N}\underbrace{\mathbf{X}\mathbf{X}^T}_{N\times N}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

Eigenfaces

- After we obtained the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$, we can use $(\mathbf{X}^T \mathbf{v}_i)$ to get eigenvector of S with eigenvalue λ_i:

$$\underbrace{(\frac{1}{N}\mathbf{X}^T\mathbf{X})}_{S}\underbrace{(\mathbf{X}^T\mathbf{v}_i)}_{D\times 1} = \lambda_i(\mathbf{X}^T\mathbf{v}_i)$$

Eigenfaces

- After we obtained the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots$, we can use $(\mathbf{X}^T\mathbf{v}_i)$ to get eigenvector of S with eigenvalue λ_i:

$$\underbrace{(\frac{1}{N}\mathbf{X}^T\mathbf{X})}_{S}\underbrace{(\mathbf{X}^T\mathbf{v}_i)}_{D\times 1} = \lambda_i(\mathbf{X}^T\mathbf{v}_i)$$

- Rescale $\mathbf{u}_i \propto \mathbf{X}^T\mathbf{v}_i$ such that $||\mathbf{u}_i|| = 1$.

Eigenfaces

- After we obtained the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots,$ we can use $(\mathbf{X}^T\mathbf{v}_i)$ to get eigenvector of S with eigenvalue λ_i:

$$\underbrace{(\frac{1}{N}\mathbf{X}^T\mathbf{X})}_{S}\underbrace{(\mathbf{X}^T\mathbf{v}_i)}_{D\times 1} = \lambda_i(\mathbf{X}^T\mathbf{v}_i)$$

- Rescale $\mathbf{u}_i \propto \mathbf{X}^T\mathbf{v}_i$ such that $||\mathbf{u}_i|| = 1$.
- Keep top-K eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K$ to form K eigenfaces. Reshape the D-dimensional $\mathbf{u}_i$ to $[h, w]$ to visualize the eigenfaces.

General procedures of obtaining eigenfaces

1. Construct the data matrix $\mathbf{X}$, each row contains one centred face image represented by long-vector.
2. Consider $\mathbf{X}\mathbf{X}^T$ $(N \times N)$, and find its $K(< N)$ eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$.
3. Get K eigenfaces by normalizing the corresponding $\mathbf{X}^T\mathbf{v}_1, \ldots, \mathbf{X}^T\mathbf{v}_K$.

Representing new data using eigenfaces

- Let $W = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$ which contains K eigenfaces that forms the new coordinate system.

Representing new data using eigenfaces

- Let $W = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$ which contains K eigenfaces that forms the new coordinate system.
- Given test/new face image $\mathbf{x}$, centering it using mean image, i.e., $\mathbf{x} - \bar{\mathbf{x}}$.

Representing new data using eigenfaces

- Let $W = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$ which contains K eigenfaces that forms the new coordinate system.
- Given test/new face image $\mathbf{x}$, centering it using mean image, i.e., $\mathbf{x} - \bar{\mathbf{x}}$.
- Project it onto eigenfaces: $(\mathbf{x} - \bar{\mathbf{x}})W$ (think of it as coordinates/projected value on the new system).

Representing new data using eigenfaces

- Let $W = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$ which contains K eigenfaces that forms the new coordinate system.
- Given test/new face image $\mathbf{x}$, centering it using mean image, i.e., $\mathbf{x} - \bar{\mathbf{x}}$.
- Project it onto eigenfaces: $(\mathbf{x} - \bar{\mathbf{x}})W$ (think of it as coordinates/projected value on the new system).
- Approximate the centred test/new face image using eigenfaces: $(\mathbf{x} - \bar{\mathbf{x}})WW^T$.

Representing new data using eigenfaces

- Let $W = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K)$ which contains K eigenfaces that forms the new coordinate system.
- Given test/new face image $\mathbf{x}$, centering it using mean image, i.e., $\mathbf{x} - \bar{\mathbf{x}}$.
- Project it onto eigenfaces: $(\mathbf{x} - \bar{\mathbf{x}})W$ (think of it as coordinates/projected value on the new system).
- Approximate the centred test/new face image using eigenfaces: $(\mathbf{x} - \bar{\mathbf{x}})WW^T$.
- Finally, add back the mean image to obtain the approximation/reconstruction for test/new image: $\bar{\mathbf{x}} + (\mathbf{x} - \bar{\mathbf{x}})WW^T$

Eigenface representation

Figure: [Source]

PCA vs FLD

- Both methods can be viewed as techniques for linear **dimensionality reduction**.

PCA vs FLD

- Both methods can be viewed as techniques for linear **dimensionality reduction**.
- Fisher's Linear Discriminant uses class-label information which is **supervised** learning. Can be used for classification.

PCA vs FLD

- Both methods can be viewed as techniques for linear **dimensionality reduction**.
- Fisher's Linear Discriminant uses class-label information which is **supervised** learning. Can be used for classification.
- Principal Component Analysis depends only on data $\mathbf{x}$ without label information which is **unsupervised**.

PCA vs FLD

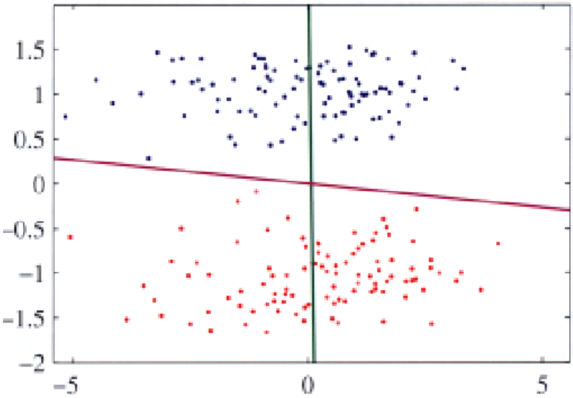

Figure: Red line: PCA. Green line: FLD [C.Bishop, PRML]

Acknowledgement and Further Reading

First few slides of Fisher Linear Discriminant are taken from Dr. Y. Ning's Spring 19 offering of CS-559.

Part of the discussion on different views of PCA is inspired by [bioramble]

Further Reading:
Chapter 4.1 and 12.1 of *Pattern Recognition and Machine Learning* by C. Bishop.