Note:

- Homework #4
  - On Lecture 8-9, **Due in two weeks**
  - Accessible on *Canvas* now.
  - Submit the PDF file to "Submission for HW #4" link by 4/15 4pm. No late submission. Strictly applied.
  - **Do not procrastinate this homework which could be time-consuming**
- Fast Prototyping Exercise #2 on Mean Shift continues (second session).
  - https://bidal.sfsu.edu/~kazokada/csc872/PD2.pdf

1

Statistical Modeling

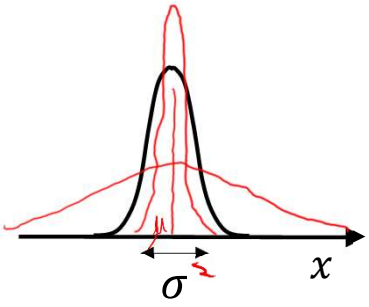
PF: Gaussian Mixture Model

CSC 872

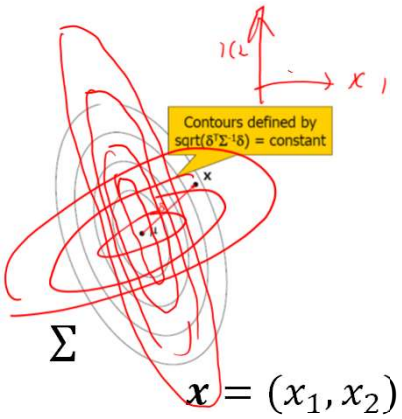
Pattern Analysis and Machine Intelligence

2

Let's be OK with Multivariate Gaussian



Variance controls the shape



Covariance controls the shape

3

Understanding Variance: Review

- **Univariate Domain:**
  - Given a random scalar variable X
  - True Mean Definition: $E[X] = \mu$
  - True Variance Definition: $Var[X] = E[(X-\mu)^2] = E[X^2] - (E[X])^2 = \sigma^2$

- **MLE of P(X) as a Gaussian Distribution**
  - Given a sample $x_1,..,x_N$ drawn from a Gaussian

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

  - MLE of mean is sample mean

$$\mu^{mle} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

  - MLE of variance is sample variance

$$\sigma^2_{mle} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu^{mle})^2$$

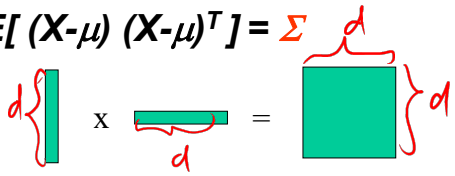
4

Covariance: Definition

- **Multivariate Domain:**
  - Given a random column-vector variable X
  - True Mean Definition:
  - True Covariance Definition:

$E[X] = \mu$

$Cov[X] = E[(X-\mu)(X-\mu)^T] = \Sigma$

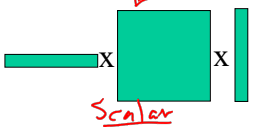


5

Covariance: MLE for Gaussian

- **Multivariate Domain:**
- **MLE of P(X) as a Gaussian Distribution**
  - Given a sample $x_1,..,x_N$ drawn from a Gaussian

$$N(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}}\exp(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))$$



$$\boldsymbol{\mu}^{mle} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}^{mle} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n-\boldsymbol{\mu}^{mle})(\boldsymbol{x}_n-\boldsymbol{\mu}^{mle})^T$$

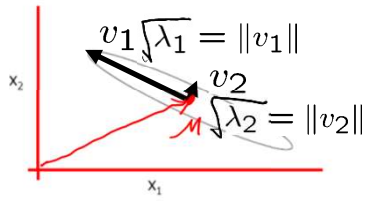
6

Understanding Covariance

- Symmetric & Square
  - Transpose of A is the same as A
- Positive semi-definite (non-negative definite)
  - Eigen values of A are all positive or zero
  - Quadratic function $x^T Ax$ is positive or zero for all x
  - The power of exponent in the multivariate Gaussian is always negative!
- Ellipsoidal shape and Cov
  - Eigen vectors = Ellipsoidal axes
  - Eigen values = Ellipsoidal axis length

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_1 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma^2_2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma^2_m \end{pmatrix}$$

$v_1, \ldots, v_n$
$\lambda_1, \ldots, \lambda_n$

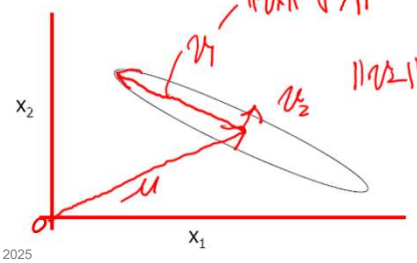


7

Understanding Covariance

- **General Gaussian: Fully-valued Covariance**
- Any oriented ellipsoidal shape

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_1 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{12} & \sigma^2_2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m} & \sigma_{2m} & \cdots & \sigma^2_m \end{pmatrix}$$



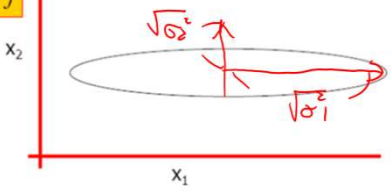
8

Understanding Covariance

- **Axis-Aligned Gaussian: Diagonal Covariance**
- Any axis-aligned ellipsoidal shape
- Every X_i are independent to each other

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2_{m-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2_m \end{pmatrix}$$

$X_i \perp X_i$ for $i \neq j$



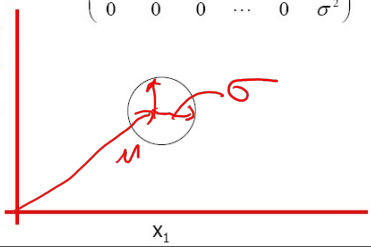
9

Understanding Covariance

- **Spherical Gaussian: $\Sigma = \sigma^2 I$**
- Spherical shape
- Independent & identical size

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_m \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2 \end{pmatrix}$$

$X_i \perp X_i$ for $i \neq j$



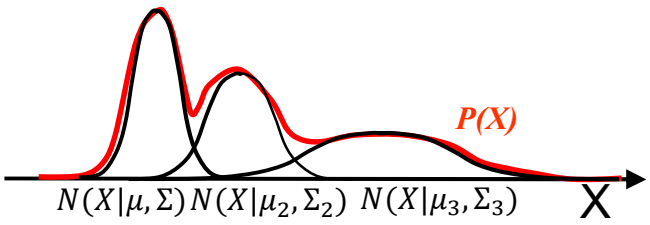
10

Two Statistical Modeling Approaches

- **Non-Parametric**: *Histogram & KDE*
  - **Yes**: Flexible, accurately describe arbitrary distributions
  - **No**: High Time and Space Complexity
- **Parametric**: *MLE & MAP*
  - **Yes**: Low Time and Space Complexity
  - **No**: Rigid, it may not be accurate
- **Any flexible but economic parametric model???**

YES

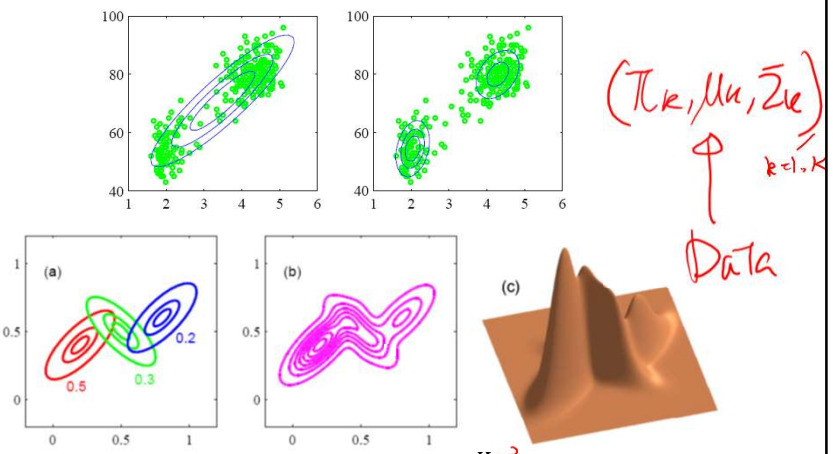
- Gaussian Mixture Model
  - Suppose P(X) takes a form of a **weighted sum** of *K* different Gaussian components



$$P(X) = \sum_{k=1}^{K} \pi_k N(X|\mu_k, \Sigma_k)$$

π_k: *mixing weights* $\sum_{k=1}^{K} \pi_k = 1$

In 2D



(π_k, μ_k, Σ_k) $k=1,K$

Data

$$P(X) = \sum_{k=1}^{K=3} \pi_k N(X|\mu_k, \Sigma_k)$$

13

PF: Mixture Model

- More general form of a Mixture Model
- A mixture (=weighted sum) of arbitrary component distributions parameterized by θ_k

$$P(X) = \sum_{k=1}^{N} \pi_k P(X|\theta_k)$$

Data → $\{\pi_1, \cdots \pi_N\}$ $\{\theta_1, \cdots \theta_N\}$

14

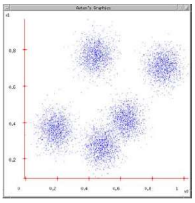
Sampling from GMM

- Sampling: The Inverse Problem of Modeling
  - Draw a set of data points from a known prob. dist.
- Sampling from GMM *P(X)* is two-step!
  1) **Pick one of Gaussians according to $\pi = \pi_1,..,\pi_K$**
  2) **Generate x ~ the chosen Gaussian component**
- **Latent variable** $Z \in \{1,..,K\}$
  - A random variable that picks one of the Gaussians!
  - π_k **= P(Z=k|π)**

$$P(X) = \sum_{k=1}^{K} P(Z = k|\pi) N(X|\mu_k, \Sigma_k)$$

STOP: Problem

- You have N data points
- You know they all come from *K* Gaussian Mixture
- Let me ask you. Can I get MLE of the μ's?
- No problem!
- MLE of Gaussian is sample mean! so I just need to compute K sample means
- Oh there's one thing.
- None of the data are labeled. I don't know which Gaussian each point is from…
- Oh oh. You cannot do the MLE…



Problem: More Formally

- Latent variable Z is not observable
- Data is unlabeled >>> Unsupervised Learning
- Data is labeled then >>> Supervised Learning (Later)

- Incomplete data likelihood $f(\theta) = \log P(x_1, \ldots, x_N|\theta)$
- Unknown $z_1,..,z_N$ and θ
- Good old MLE recipe

$$\frac{\partial f(\theta)}{\partial \theta} = 0 \quad \frac{\partial f(\theta)}{\partial z} = 0$$

$$= \sum_n \log P(\boldsymbol{x}_n|\theta)$$
$$= \sum_n \log \sum_k P(\boldsymbol{x}_n, \boldsymbol{z}_k|\theta)$$
$$= \sum_n \log \sum_k P(\boldsymbol{z}_k|\pi) N(\boldsymbol{x}_n|\boldsymbol{z}_k, \theta)$$

- Well, it yields non-linear eq. You cannot solve them

Expectation Maximization Algorithm

- **Goal**: solve MLE problem of *f(θ) iteratively*
- Basic Idea
  - MLE of incomplete likelihood is difficult due to the unknown labels Z so…
- **First find best label Z guess for each data point X**
  - **E-step**: expectation
  - Expected value of label Z is computed, solving this problem probabilistically
- **With the guessed Z, you can find MLE of θ with X**
  - **M-step**: maximization
  - Optimize complete likelihood instead of incomplete likelihood
- Iterate these two steps (EM-algorithm) probabilistically
- You can prove that this converges to a nearest local optimum/peak ! (without step-size tuning like Gradient D)

Cost function for EM

- Given observed *X* and unobserved *Z*
- Complete log data likelihood

$$\log P(X,Z|\theta) = \log \prod_n P(z_n|\pi)P(x_n|z_n,\theta)$$
$$= \sum_n \log P(z_n|\pi)N(x_n|z_n,\theta)$$

- Auxiliary function Q(θ|θ') that you are going to optimize

Marginalizing !

$$Q(\theta|\theta') = E_{Z|X,\theta'}[\log P(X,Z|\theta)]$$

PS: EM Algorithm for GMM

- Given $X=x_1,..,x_n,..,x_N$, $\theta = (\{\pi_k\}, \{\mu_k\}, \{\Sigma_k\}), K$
- E-step: calculate $P(z_n|x_n,\theta_{old})$ for each x_n

$$P(z_n = k|x_n,\theta) = \frac{N(x_n|\mu_k,\Sigma_k)\pi_k}{\sum_{k'} N(x_n|\mu_{k'},\Sigma_{k'})\pi_{k'}} = r_{nk}$$

- M-step: replace current θ_{old} by solving

$$\theta_{new} \leftarrow \text{argmax}_{\theta'} Q(\theta'|\theta_{old})$$

$$\pi_k^{new} = \frac{1}{N}\sum_n r_{nk}$$

$$\mu_k^{new} = \frac{\sum_n r_{nk}x_n}{\sum_n r_{nk}}$$

$$\Sigma_k^{new} = \frac{\sum_n r_{nk}(x_n-\mu_k^{new})(x_n-\mu_k^{new})^T}{\sum_n r_{nk}}$$

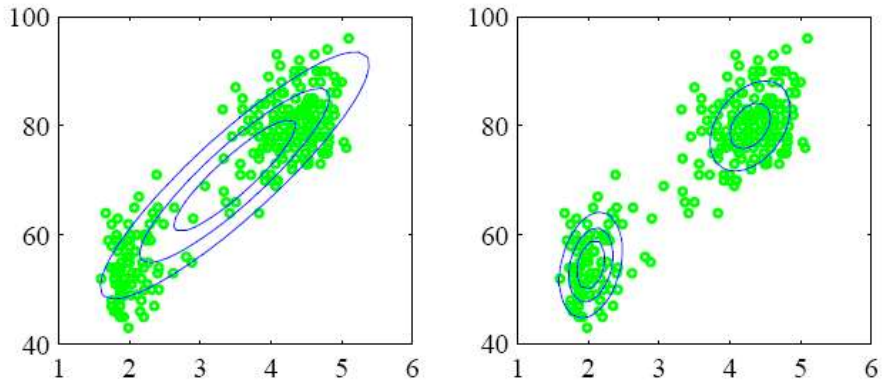
Rather Simple !

Lots of algebra here. If you want to master EM, you must once derive these by hand!!! Takes time but rewarding, I assure you.

Relation to Clustering

- It is a parametric clustering!!!
- Bayesian way of doing clustering
  - After running GMM-EM on your data, you have a Gaussian prob. distribution for each cluster
  - Any given point, you can do MAP
  - **Which cluster label *k* maximizes $P(z_n=k|x_n,\theta)$?**
- Let's compare this with mean shift
- Better?: yes, faster and more economical
- Worse?: yes, well both can get stuck at local optima but really you need to know *K* beforehand for EM

Number of Clusters…



PS: K-Mean Clustering

- What about other clustering methods I learned in your other favorite courses???
- Key Observation: Notice that EM is not really computing the value for *Z* at each step.
- It only computes a probability distribution of *Z*
- This is called **Soft-Assignment**
- Replace it with **Hard-Assignment?**
- At each iteration, you do the **MAP** and get *Z* value
- Then use regular **MLE** formulae to get parameters
- **It is the *K*-Mean Clustering!!! (with $\Sigma = \sigma I$, $\sigma \to 0$)**

- **Moral: GMM-EM is a generalization of *K*-Mean**

E-step: calculate $P(z_n|x_n, \theta_{old})$

Simple mean
Simple variance

Curse of Dimensionality

- **Here is A MAJOR problem!!!**
- Linear increase in dimension of domain yields
- **Exponential increase in volume**

- Examples
  - Joint distribution of random variables with 10 attributes $\mathbf{x}=(x_1,..,x_{10})$
  - **2** variables P(x,y): 10^2 = **100 possible combinations**
  - **10** variables P(x,y,z,a,..,g): 10^{10} = **10 billion possible combinations!!!**

- Examples
  - 1D time series analysis to 3D RBG color analysis
  - Bioinformatics: a study on a few markers to a full set of genes

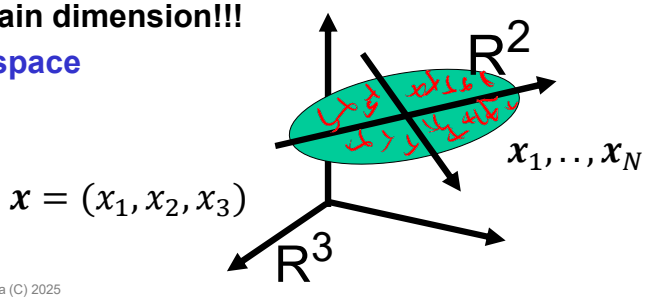
The deal is that…

- **Because of this, solving a real-world problem is difficult!!!**
  - **Cannot model** joint distribution in high-dimension
  - **Cannot sample** sufficiently from a joint distribution
  - Takes **a lot of time** to compute
  - Takes **a lot of space** to keep in memory

- We learned that **Parametric Modeling** makes things better …..
- But you may **not have enough number of samples** to accurately estimate the covariance ….
- Oh NO… **What we gonna do???**

Dimensionality Reduction

- **A SERIOUSLY GOOD NEWS OF TODAY**
  - **You can describe entire information of your data by using much smaller number of variables**
  - **Intrinsic Dimensionality**
  - **Variance of your data is (most of time) confined to a space whose dimensionality is lower than your domain dimension!!!**
  - **Subspace**



Dimensionality Reduction

- **Example**
  - Consider a 100x100 pixel image
  - We take it as 10,000 dimensional vector $\in R^{10000}$
  - Take 100 images
  - We have 100 points in R^{10000}
  - What is the dimensionality of the subspace that contains the 100 points?
  - Typically ~100 <<< 10,000 !!!
- **Linear Dimension Reduction**
  - Find a linear transform W given $x_1,..,x_N$ such that

$$y = Wx$$

$$100 = L < M = 10{,}000$$

$$x \in R^M \quad (= 10{,}000)$$

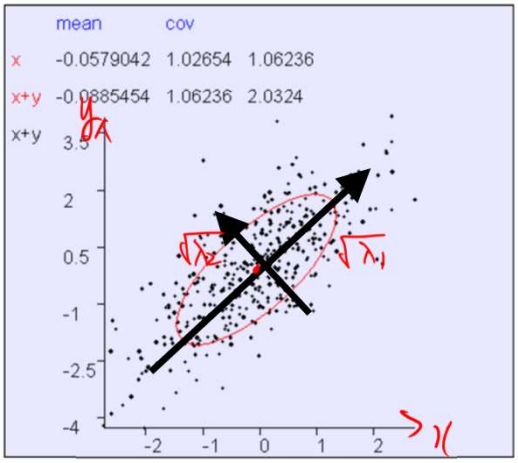
$$y \in R^L \quad (= 100)$$

$$W \in R^{L \times M} \quad (100 \times 10{,}000)$$

PS: Principal Component Analysis

- **PCA:** Find a linear subspace with a lower-dimension that captures most data variance of *N* *d*-variate samples
- **Steps (Pseudo Code)**
  1) Compute *d* x *d* sample covariance matrix
  2) Solve its eigen-value problem
  3) Resulting a set of eigen-values and corresponding eigen-vectors
  4) Sort the eigen-vectors according to the eigen-values
  5) Choose top-*K* eigen-vectors with highest eigen-values
  6) Set each row of W by the *K* eigen-vectors (called principal components)

Principal Components in 2D



Eigenvalues = σ^2

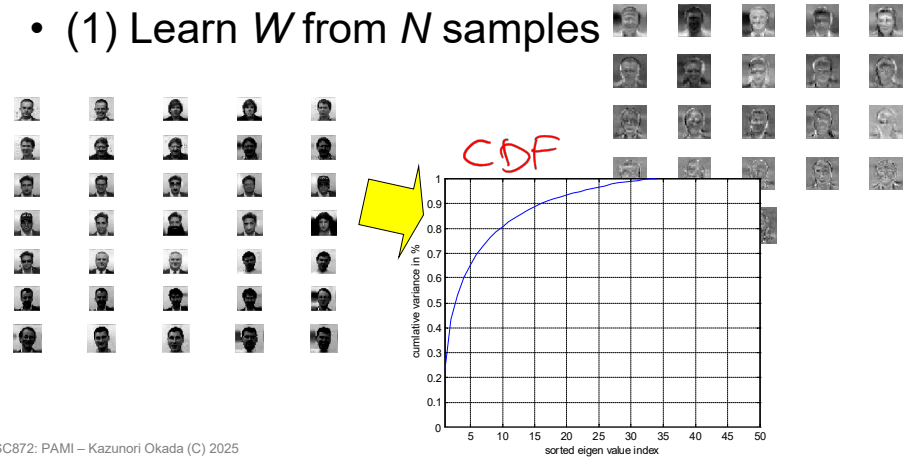
PCs are perpendicular to each other

PCs are independent to each other

29

Eigenface: Learning

- Use PCA to extract economic feature for describing facial images
- (1) Learn *W* from *N* samples



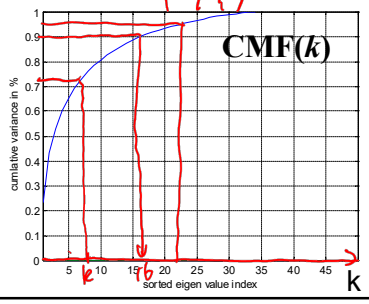
30

PCA: Principal Component Anal.

- Steps
  1) Collect Training Images (must be aligned)
  2) Vectorize the Images: $X = \{x_1,..,x_N\}$
  3) Construct Covariance Matrix: $C = XX^T$
  4) Solve Eigenvalue Problem: $Cv_i = \lambda_i v_i$
  5) Select Top Eigenvectors $W = \{v_1,..,v_K\}^T$

$$\lambda_1 > \lambda_2 > \cdots > \lambda_k > \cdots > \lambda_M$$

$$\mathrm{CMF}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{M} \lambda_i}$$

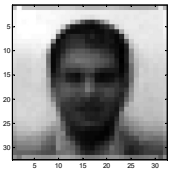


Eigenface: Feature Extraction

- Use PCA to extract economic feature for describing facial images
- (2) Extract a feature *f* of a face ***x*** using ***W***

$$\boldsymbol{f} = \boldsymbol{W}(\boldsymbol{x} - \boldsymbol{\mu})$$

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$



Face Recognition with Eigen Face

- Preprocess:
  - Prepare DB of *N* known face by using $\boldsymbol{W}$ and μ
- Recognition: Given an input face $\boldsymbol{x}$, which entry of the known person DB is closest to the input?
- Steps (Pseudo Code)
  1) Extract the feature of input using $\boldsymbol{W}$ and μ
  2) Compute the Euclidean distance of the input feature to the DB features.
  3) Find the DB entry with the smallest Euclidean distance to the input
  4) Output the index of the best match entry
  5) Show the image of the best match entry

Summary

- Gaussian Mixture Model
  - Gaussian Mixture Model
  - EM Algorithm
  - K-Mean Clustering
  - Curse of Dimensionality
  - Dimensionality Reduction
  - Principal Component Analysis
  - Eigenface and Face Recognition
- Next
  - General Framework for Machine Learning
  - Pattern Classification Examples