8 Understand Pattern Recognition & Decision Theory

Outline

- Understand what the Pattern recognition is
- Understand the process of pattern recognition
- Understand how the decision theory is developed from an intuitive process that everyone can do to a mathematical abstract theory
- The optimal decision rule: MAP and Bayesian decision rules
- Evaluation a pattern recognition system: Recognition accuracy or error rate

8 Understand Pattern Recognition & Decision Theory

- What is on earth the Pattern Recognition?
- Pattern recognition is to perceive a pattern, extract the relevant information, understand the content of the information and make decision automatically by machine or computer.
- Example 1: Is this apple or pear?

After perceiving the images, you may decide image 1 is pear and 2 is apple with 100% certainty (probability 1), based on color/shape?

- Example 2: Is this apple or pear?


After perceiving the images, you must first extract the right part from image and then make decision.

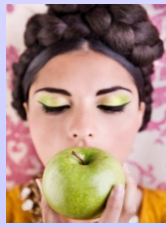
- Example 3: Is this apple or pear?

After perceiving the images, you may figure out:

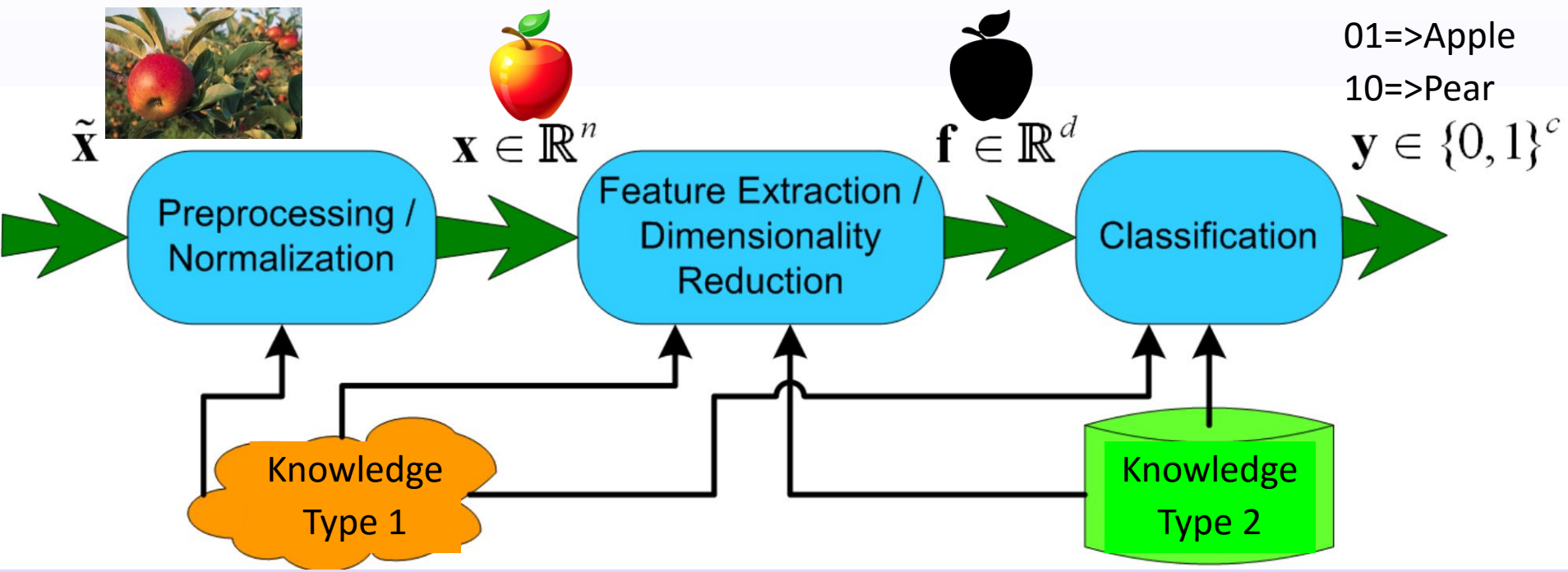
the probability of apple is 0.9, 0.4, 0.6 and 0.7.

So the probability of pear is 0.1, 0.6, 0.4 and 0.3. How do you make decision?



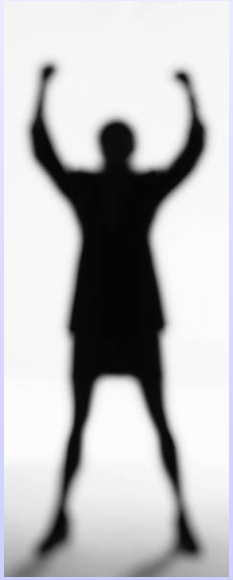
8 Understand Pattern Recognition & Decision Theory



- A pattern recognition system diagram
- Pattern recognition, especially image recognition, is very easy for human brain but very difficult for machine or computer.
- What tasks are very difficult for human brain but very easy for machine or computer?

8 Understand Pattern Recognition & Decision Theory

- How the Pattern Recognition works?
- A simple example 1: Somebody approaches you in the street who is totally disguised. You only know he/she is of height 1.72 meter. You need decide/judge if he/she a male or a female?
- What is your decision?
- if you are not silly you will judge it is a male .
- How do you work out that this person is a male?
- No female is of height of 1.72m? No.
- Or, are all male people of height 1.72m? No
- Or, are most male people are of 1.72m? No
- Is your decision definitely correct?



8 Understand Pattern Recognition & Decision Theory

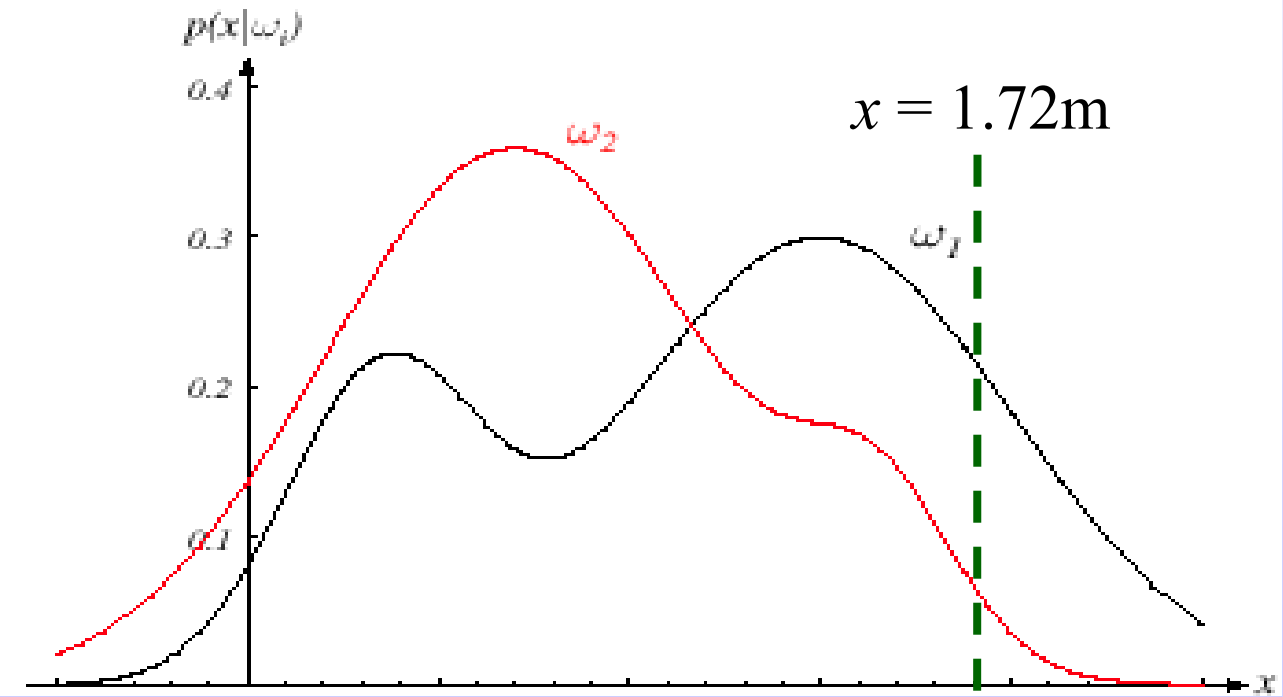
- You made a reasonable decision based on your knowledge that, for example, 30% of males are of height around 1.72m and 10% of females are of height around 1.72m.
- What is this knowledge based on which you make a reasonable decision?
- This knowledge in mathematical terms is that the probability of the height of male population is 0.3 around 1.72m, though it is less than 0.5 but the probability of the height of female population is 0.1 around 1.72m, which is much smaller than 0.3.
- Using mathematical formula, let x denote the height, ω_1 denote the class male and ω_2 denote the class female. This knowledge can be represented by $p(x|\omega_1)=0.3$ and $p(x|\omega_2)=0.1$ for $x=1.72\text{m}$.
- So you make a right decision of ω_1 because $p(x|\omega_1)>p(x|\omega_2)$

8 Understand Pattern Recognition & Decision Theory

- $p(x|\omega_1), p(x|\omega_2)$ are called class-conditional probability distribution function (discrete x) or density (continuous x). It is clear:

$$\sum_{x=0}^{3} p(x \mid \omega_1) = 1, \text{ and } \sum_{x=0}^{3} p(x \mid \omega_2) = 1 \text{ for discrete } x$$

$$\int_0^3 p(x \mid \omega_1)dx = 1, \text{ and } \int_0^3 p(x \mid \omega_2)dx = 1 \text{ for continuous } x$$



8 Understand Pattern Recognition & Decision Theory

- Is this definitely a wise decision? ω_1 : because $p(x \mid \omega_1) > p(x \mid \omega_2)$
- Yes if you meet this person in the street. Why?
- Because in the world the human population has 50% males and 50% females. Mathematically, this means the prior probability:

$$p(\omega_1) = p(\omega_2) = 0.5, \quad \text{with } p(\omega_1) + p(\omega_2) = 1$$

- Now, example 2: suppose you meet this person in the campus of a nurse school, where 95% of the people in the campus are female and 5% of the people are male. This case means

$$p(\omega_1) = 0.05, \quad p(\omega_2) = 0.95, \quad \text{so } p(\omega_1) \neq p(\omega_2)$$

- Will you in this case still judge this person is a male base on

$$\omega_1 : \text{because } p(x \mid \omega_1) > p(x \mid \omega_2) \text{ at } x = 1.72 \ ?$$

- Most likely not if you are smart. You will judge it is a female.
- Why?

8 Understand Pattern Recognition & Decision Theory

- Will you judge this person is a female just base on

$$p(\omega_1) = 0.05, \quad p(\omega_2) = 0.95, \quad \text{so } p(\omega_2) > p(\omega_1) \ ?$$

- No. it means you judge it as a female for whoever you have seen.
- The right way for the decision is to compare the probability of male and the female in this scenario after you get the information of height 1.72m.
- Mathematically, this probability is denoted by $p(\omega_1|x)$ and $p(\omega_2|x)$. They are called a posterior probability. It means the probability of a class ω_i after knowing the data value x in a particular scenario.
- Therefore, the right decision is

$$\text{Decide } \omega_k : \text{if } p(\omega_k \mid x) > p(\omega_i \mid x), i \neq k$$

$$\text{or more general: } \omega_k = \arg\max_{\omega_i}[p(\omega_i \mid x)]$$

- not $\omega_1 : \text{if } p(x \mid \omega_1) > p(x \mid \omega_2)$ nor $\omega_1 : \text{if } p(\omega_1) > p(\omega_2)$ ×

8 Understand Pattern Recognition & Decision Theory

- As you decide ω_k, base on your observed value x, the probability that you make a correct decision is hence $p(\omega_k|x)$. Therefore, the probability that you make a wrong decision or error is:

$$p(e_k \mid x) = 1 - p(\omega_k \mid x)$$

- If your decision is ω_k, because $p(\omega_k|x)$ is maximum, consequently, the probability of the decision error will be minimum.
- Therefore, this decision rule is optimal in the sense of minimizing the probability of the decision error.
- This decision rule is called the maximum a posterior (MAP) rule,

$$\text{Decide: } \omega_k = \arg\max_{\omega_i}[p(\omega_i \mid x)] = \arg\min_{\omega_i}[p(e_i \mid x)]$$

- It minimizes the probability of the decision error.

8 Understand Pattern Recognition & Decision Theory

- More generally, If there are c possible classes, obviously

$$\sum_{i=1}^{c} p(\omega_i \mid x) = 1, \quad \text{for } \forall x$$

- Therefore

$$p(e_k \mid x) = 1 - p(\omega_k \mid x) = \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i \mid x)$$

- To compute $p(\omega_k|x)$, we just need a very basic formula in the probability theory. I trust all of you know and have good understand of the formula of computing joint probability

$$p(x, \omega_i) = p(x)p(\omega_i \mid x) = p(\omega_i)p(x \mid \omega_i)$$

$$p(A,B) = p(A)p(B \mid A) = p(B)p(A \mid B)$$

- So we have: $p(\omega_i \mid x) = \dfrac{p(\omega_i)p(x \mid \omega_i)}{p(x)}$

- In addition: $p(x) = \sum_{i=1}^{c} p(x \mid \omega_i)p(\omega_i)$

8 Understand Pattern Recognition & Decision Theory

- In the example 1, $p(x|\omega_1)=0.3$ and $p(x|\omega_2)=0.1$ for x=1.72m and $p(\omega_1)=p(\omega_2)=0.5$
- So we have $p(x)=\sum_{i=1}^{c} p(x|\omega_i)p(\omega_i)=0.3\times0.5+0.1\times0.5=0.2$
- and

$$p(\omega_1|x)=\frac{p(\omega_1)p(x|\omega_1)}{p(x)}=\frac{0.5\times0.3}{0.2}=0.75$$

$$p(\omega_2|x)=\frac{p(\omega_2)p(x|\omega_2)}{p(x)}=\frac{0.5\times0.1}{0.2}=0.25$$

- In the example 2, $p(\omega_1)=0.05,\ p(\omega_2)=0.95$
- So we have $p(x)=0.3\times0.05+0.1\times0.95=0.11$

$$p(\omega_1|x)=\frac{0.05\times0.3}{0.11}=0.136,\quad p(\omega_2|x)=\frac{0.95\times0.1}{0.11}=0.864$$

- We have mathematically proven that we should decide a person with height 1.72m in case 1 as man and in the case 2 as woman.

8 Understand Pattern Recognition & Decision Theory

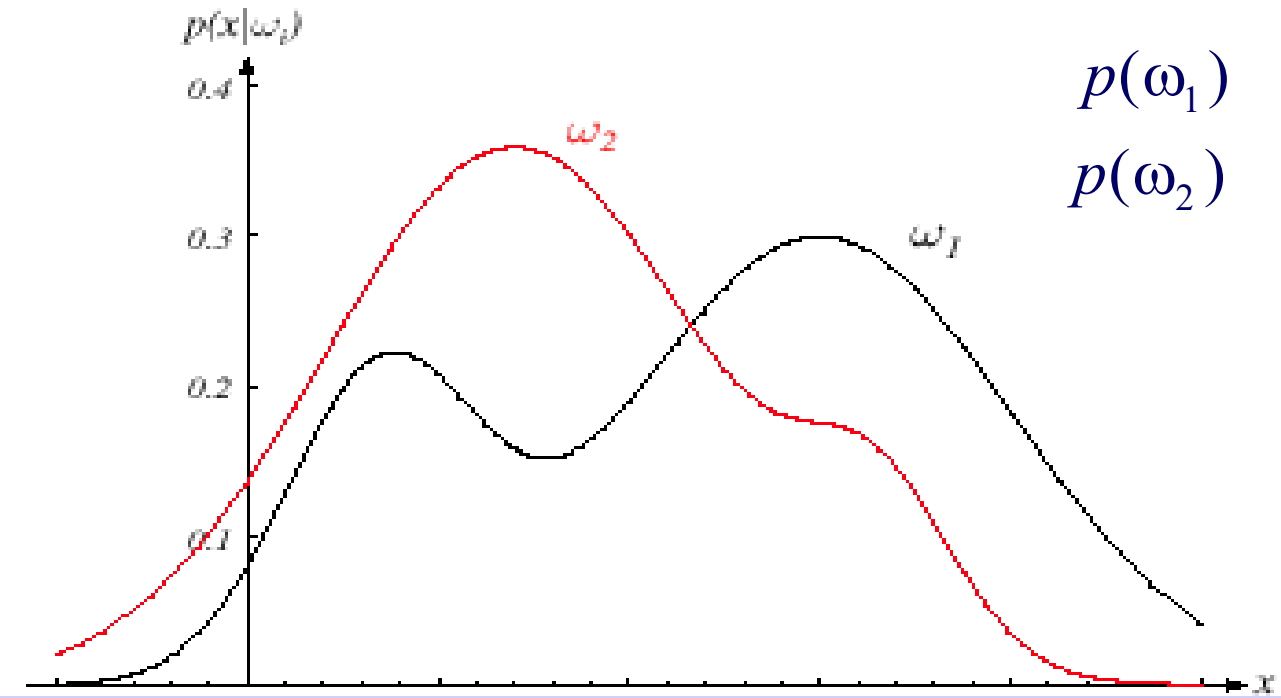
- Of cause, this decision could be correct or wrong. The probability of the wrong decision in case 1 is

$$p(e_1 \mid x) = 1 - p(\omega_1 \mid x) = 0.25$$

- And in case 2 is $p(e_2 \mid x) = 1 - p(\omega_2 \mid x) = 0.136$
- These two decisions are optimal as they minimize the decision error.
- What is the decision error if you make the opposite decision?
- The above are 2 simple pattern recognition examples with one perceiving variable and at a specific value, x=1.72m. Obviously, these two tasks can be very easily completed fully automatically by machine or computer. However, if you produce a machine to automatically do this task, you should design your machine to fully automatically recognize the person's gender for any value of x.

8 Understand Pattern Recognition & Decision Theory

- To fully automatically recognize a person's gender for all value of x, you need the prior probability $p(\omega_i)$ of all classes and the class conditional probability $p(x|\omega_i)$, of all possible value of x.



- The system will automatically classify any received value of x into male (class 1) or female (class 2).

8 Understand Pattern Recognition & Decision Theory

- The decision rule will be the same as before:

$$\text{Decide: } \omega_k = \arg\max_{\omega_i}[p(\omega_i \mid x)]$$

- But how good is the system? Or what is the performance of the system? Or how to evaluate your designed system?

- We have understand the formula to compute the error probability for a specific value of x.

$$p(e_k \mid x) = 1 - p(\omega_k \mid x) = \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i \mid x)$$

- It is not difficult to understand that the performance of a pattern recognition system can be measured by the average of $p(e_k|x)$ over all possible value of x.

8 Understand Pattern Recognition & Decision Theory

- How to average of $p(e_k|x)$ over all possible value of x?

$$p(e)? = \frac{1}{n_x}\sum_{x=0}^{3} p(e_k \mid x) \text{ for discrete } x$$

$$p(e)? = \frac{1}{3}\int_0^3 p(e_k \mid x)dx, \text{ for continuous } x$$

- The above is not a proper way because x is a random variable with different probability of occurrence at different x value.
- The right way for a random variable should be:

$$p(e) = \sum_{x=-\infty}^{\infty} p(e_k \mid x)p(x) \text{ for discrete } x$$

$$p(e) = \int_{-\infty}^{\infty} p(e_k \mid x)p(x)dx, \text{ for continuous } x$$

8 Understand Pattern Recognition & Decision Theory

- Let's take the continuous x for further working:

$$p(e)=\int_{-\infty}^{\infty}p(e_k\mid x)p(x)dx=\int_{-\infty}^{\infty}[1-p(\omega_k\mid x)]p(x)dx$$

$$=\int_{-\infty}^{\infty}[1-\frac{p(\omega_k)p(x\mid\omega_k)}{p(x)}]p(x)dx=1-\int_{-\infty}^{\infty}p(\omega_k)p(x\mid\omega_k)dx$$

- Note that for different region of x, the pattern recognition system has different decision ω_k. Pattern recognition is to partition the whole space of x into c decision regions $\Re_i$. Therefore,

$$p(e)=1-\sum_{i=1}^{c}\int_{\Re_i}p(\omega_i)p(x\mid\omega_i)dx=1-\sum_{i=1}^{c}p(\omega_i)\int_{\Re_i}p(x\mid\omega_i)dx$$

- Obviously, the probability of the correct decision is

$$p(correct)=1-p(e)=\sum_{i=1}^{c}p(\omega_i)\int_{\Re_i}p(x\mid\omega_i)dx$$

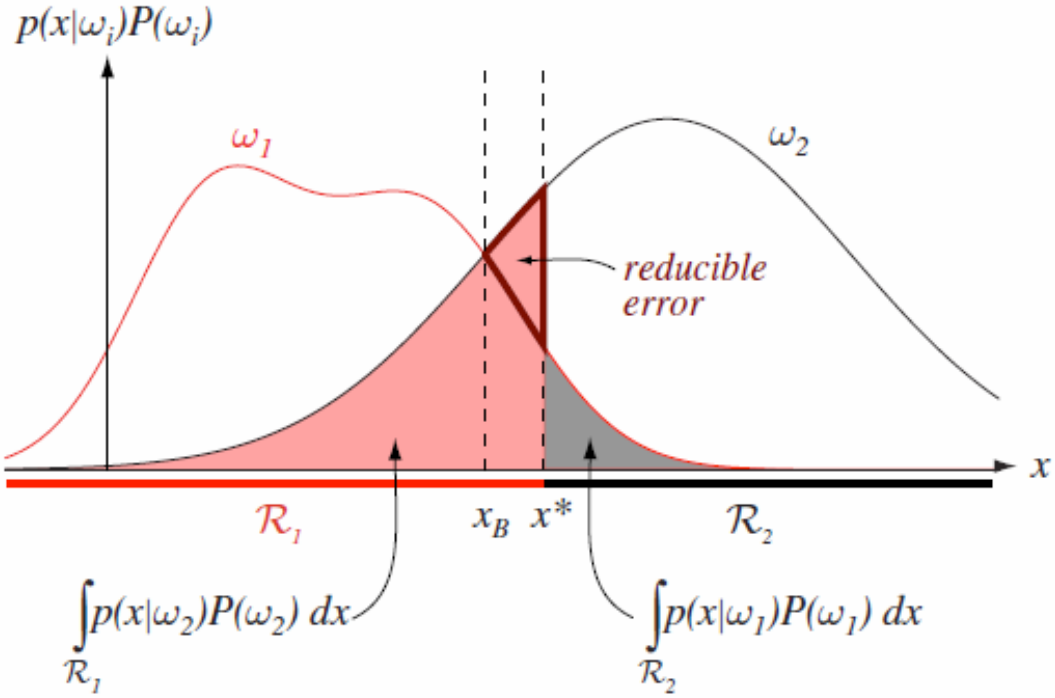


FIGURE 2.17. Components of the probability of error for equal priors and (nonoptimal) decision point x^*. The pink area corresponds to the probability of errors for deciding ω_1 when the state of nature is in fact ω_2; the gray area represents the converse, as given in Eq. 70. If the decision boundary is instead at the point of equal posterior probabilities, x_B, then this reducible error is eliminated and the total shaded area is the minimum possible; this is the Bayes decision and gives the Bayes error rate. From: Richard O.

- For the 2-class problem:

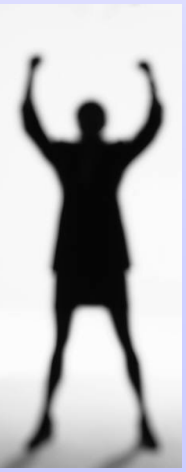
$$p(e) = 1 - p(\omega_1)\int_{\Re_1} p(x\,|\,\omega_1)dx - p(\omega_2)\int_{\Re_2} p(x\,|\,\omega_2)dx$$

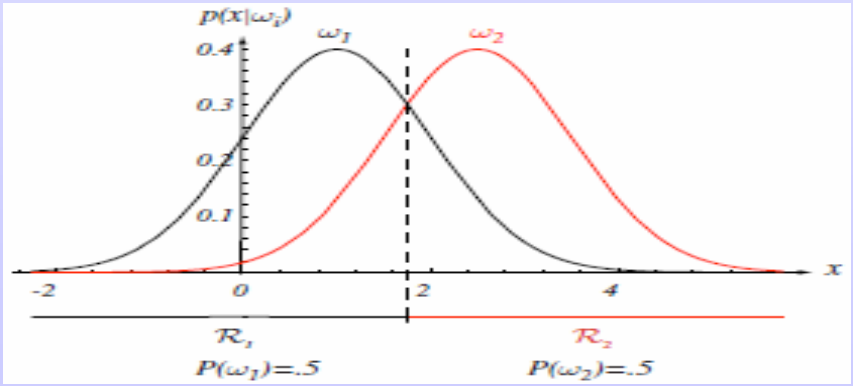
$$= p(\omega_1)\int_{\Re_2} p(x\,|\,\omega_1)dx + p(\omega_2)\int_{\Re_1} p(x\,|\,\omega_2)dx$$

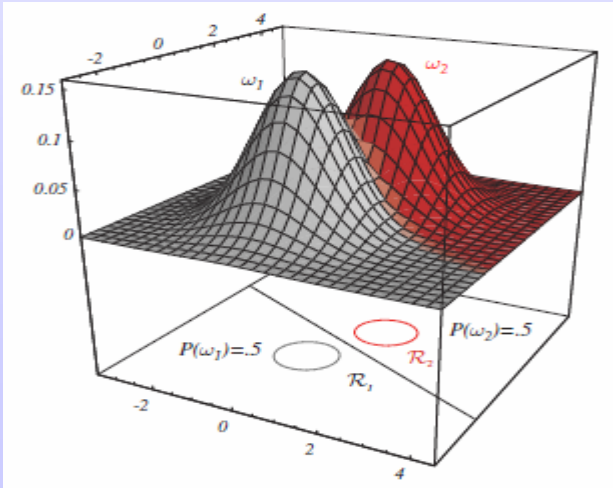
8 Understand Pattern Recognition & Decision Theory

- Now if we not only know the person's height x_1, but also the length of the hair x_2, it is not difficult to image that we will make better decision, i.e. have lower error probability. The recognition principle will be the same as before as long as now you use a vector $\mathbf{x} = [x_1, x_2]^T$

  to replace the scalar x before. The probability distribution is now two dimensional. A scalar threshold to separate the two classes in 1D case becomes a curve in the plane spanned by $\mathbf{x}$.

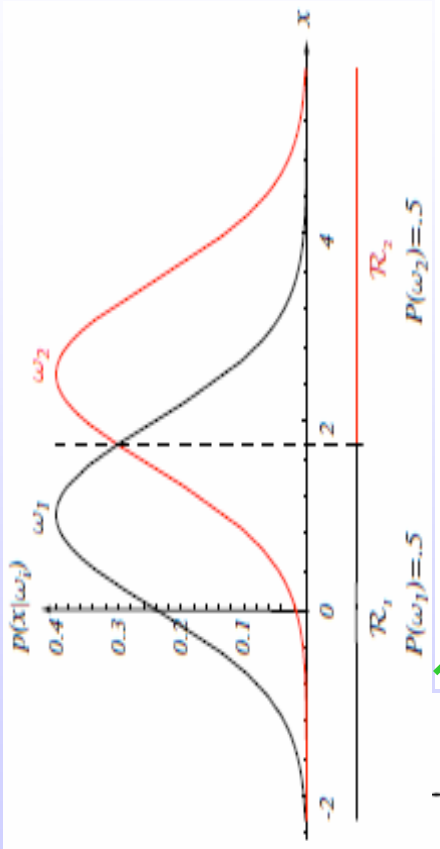


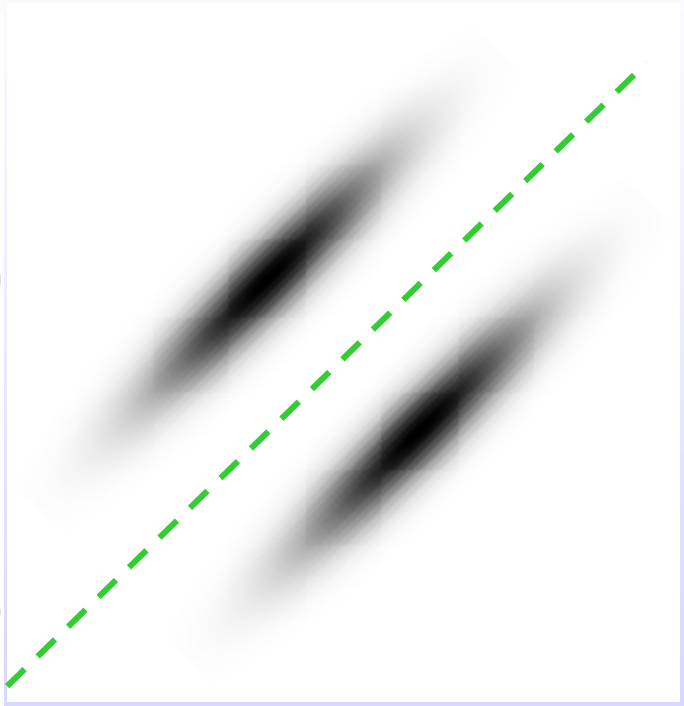


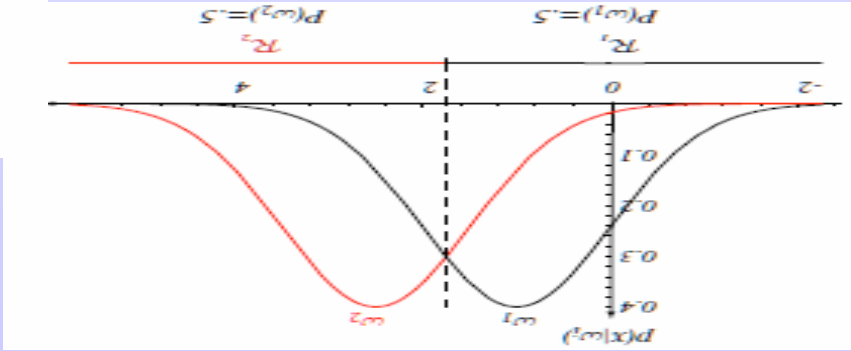


8 Understand Pattern Recognition & Decision Theory

- The increase of the information or data or feature dimension in general increases the probability of the correct decision or reduce the probability of the decision error.







8 Understand Pattern Recognition & Decision Theory

- Now I trust you have thoroughly understand the optimal decision rule. In fact, it is a very intuitive idea.
- However, to develop a simple idea into high techniques to solve complex problem, we need formulate the idea using neat, tidy and decent mathematics. Now let's formulate the simple and intuitive ideas again in another way.
- If you have to classify an object into one ω_k of the c classes ω_i before receiving any information of this particular object, how do you do?
- It is straightforward to choose the class with highest probability. Mathematically it is: $\text{Decide: } \omega_k = \arg\max_{\omega_i}[p(\omega_i)]$
- The probability of making mistake

$$p(e_k) = 1 - p(\omega_k) = \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i)$$

is then minimum.

8 Understand Pattern Recognition & Decision Theory

- Now if you need classify an object into one ω_k of the c classes ω_i after receiving information (in a column vector $\mathbf{x}$) of this particular object, it is the same straightforward to choose the class with highest probability. Mathematically it is now:

$$\text{Decide: } \omega_k = \arg\max_{\omega_i}[p(\omega_i \mid \mathbf{x})]$$

- This decision rule is called the maximum a posterior (MAP) rule
- The probability of making mistake

$$p(e_k \mid \mathbf{x}) = 1 - p(\omega_k \mid \mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i \mid \mathbf{x})$$

  is then minimum.

- Allowing the use of more than one feature merely requires replacing the scalar x by the feature vector $\mathbf{x}$, where $\mathbf{x}$ is a point in a d-dimensional Euclidean space $\mathbb{R}^d$, called the feature space.

8 Understand Pattern Recognition & Decision Theory

- Note that

$$\because \sum_{i=1}^{c} p(\omega_i \mid \mathbf{x}) = 1 \qquad \therefore p(\omega_k \mid \mathbf{x}) = 1 - \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i \mid \mathbf{x})$$

- Therefore the decision rule can also be expressed as

$$\text{Decide: } \omega_k = \arg\min_{\omega_i} [\sum_{\substack{j=1 \\ j \neq i}}^{c} p(\omega_j \mid \mathbf{x})] = \arg\min_{\omega_i} [p(e_i \mid \mathbf{x})]$$

- It is therefore pretty clear that this decision rule minimizes the probability of the mistake or error. So it is also called the minimum error rate classifier.
- We see that the error is the sum of c-1 terms.

$$p(e_k \mid \mathbf{x}) = 1 - p(\omega_k \mid \mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{c} p(\omega_i \mid \mathbf{x})$$

- It is the sum of the probabilities of all other classes.

8 Understand Pattern Recognition & Decision Theory

- In some application, wrongly classifying one class may not be at the same cost or same risk as wrongly classifying another class .
- Let's λ_{ki} denote the cost or risk of wrongly classifying the object of class ω_i into class ω_k. The cost or risk of the decision ω_k is then

$$R_k(\mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{c} \lambda_{ki} p(\omega_i \mid \mathbf{x})$$

- Therefore the decision rule that minimizes the risk or cost becomes

$$\text{Decide: } \omega_k = \arg\min_{\omega_i}[R_i(\mathbf{x})] = \arg\min_{\omega_i}[\sum_{\substack{j=1 \\ j \neq i}}^{c} \lambda_{ij} p(\omega_j \mid \mathbf{x})]$$

- This is the famous Bayesian decision rule. Cost functions allow us to treat situations where some kinds of classification mistakes are more costly than others, although we often discuss the simplest case where all errors are equally costly.

8 Understand Pattern Recognition & Decision Theory

- In some applications, the correct decision for different class may have different cost $\lambda_{kk} \neq 0$. By including the cost of correct decision, the cost or risk of a decision is then

$$R_k(\mathbf{x}) = \sum_{i=1}^{c} \lambda_{ki} p(\omega_i \mid \mathbf{x})$$

- Therefore, the famous Bayesian decision rule is generalized as:

$$\begin{aligned}
\text{Decide } \omega_k &= \arg\min_{\omega_i}[R_i(\mathbf{x})] \\
&= \arg\min_{\omega_i}[\sum_{j=1}^{c} \lambda_{ij} p(\omega_j \mid \mathbf{x})] \\
&= \arg\min_{\omega_i}[\sum_{j=1}^{c} \lambda_{ij} p(\omega_j) p(\mathbf{x} \mid \omega_j) / \ p(\mathbf{x})] \\
&= \arg\min_{\omega_i}[\sum_{j=1}^{c} \lambda_{ij} p(\omega_j) p(\mathbf{x} \mid \omega_j)]
\end{aligned}$$

8 Understand Pattern Recognition & Decision Theory

$$R_k(\mathbf{x}) = \sum_{i=1}^{c} \lambda_{ki} p(\omega_i \mid \mathbf{x})$$

is the risk or cost for the decision ω_k at a specific value of $\mathbf{x}$, called the conditional risk.

- How good is a decision rule is evaluated by the average or overall cost or risk of a pattern recognition system

$$R = \int_{\Re_x} R_k(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \int_{\Re_x} \sum_{i=1}^{c} \lambda_{ki} p(\omega_i \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$= \int_{\Re_x} \sum_{i=1}^{c} \lambda_{ki} p(\omega_i) p(\mathbf{x} \mid \omega_i) d\mathbf{x}$$

$$= \sum_{k=1}^{c} \int_{\Re_{xk}} \sum_{i=1}^{c} \lambda_{ki} p(\omega_i) p(\mathbf{x} \mid \omega_i) d\mathbf{x}$$

- This overall cost is minimized by the Bayesian decision rule. It delivers the best performance that can be achieved.

8 Understand Pattern Recognition & Decision Theory

- Two-category classification:
  - We have two possible classes : ω_1, ω_2.
  - We also have two actions:
    - α_1 corresponds to deciding that the true class is ω_1;
    - α_2 corresponds to deciding that the true class is ω_2.
  - Let $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$, we have the conditional risks as

$$R_1(\mathbf{x}) = \lambda_{11}P(\omega_1|\mathbf{x}) + \lambda_{12}P(\omega_2|\mathbf{x})$$
$$R_2(\mathbf{x}) = \lambda_{21}P(\omega_1|\mathbf{x}) + \lambda_{22}P(\omega_2|\mathbf{x})$$

  - The Bayes decision rule : decide that the true class is ω_1 if $R_1(\mathbf{x}) < R_2(\mathbf{x})$ and ω_2 otherwise.

8 Understand Pattern Recognition & Decision Theory

- From the decision rule, we obtain the following

$$(\lambda_{21} - \lambda_{11})P(\omega_1|\mathrm{x}) > (\lambda_{12} - \lambda_{22})P(\omega_2|\mathrm{x}).$$

- Normally $\lambda_{21} - \lambda_{11}$ and $\lambda_{12} - \lambda_{22}$ are positive because the loss is greater when making a mistake.
- We can also replace the posterior probability by the product of likelihood and prior , and drop the evidence term.
- Then we can write the Bayes decision rule as: Decide ω_1 if

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{(\lambda_{12} - \lambda_{22})}{(\lambda_{21} - \lambda_{11})} \frac{P(\omega_2)}{P(\omega_1)}.$$

If the above likelihood ratio is greater than ratio of loss weighted priors, we take action α_1 (decide ω_1). Otherwise take action α_2 (decide ω_2)

This is the **Likelihood Ratio Test (LRT) .**

8 Understand Pattern Recognition & Decision Theory

- We can define the loss to be zero for correct decision and one for wrong decision for simplicity.

$$\lambda_{ij} = \begin{matrix} 0 & \text{if} & i = j \\ 1 & \text{if} & i \neq j \end{matrix}$$

- In a multi-category setting, we can write the loss in the matrix form, whose elemnets are λ_{ij},

$$\begin{bmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & 0 & 1 & \dots & 1 \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & \dots & & 0 \end{bmatrix}$$

➢ With this 0/1 loss function, the Bayesian decision rule becomes MAP rule that minimizes the error rate.

8 Understand Pattern Recognition & Decision Theory

- Numerical Example 1:

In a study of two types of objects, it is found that the sizes of objects can be modeled as two Gaussian (normal) distributions. In general, there is about the same number of type A objects as there is for type B objects. For type A objects, its class conditional density is $\mathcal{N}(2,1)$ and for type B, its class conditional density is $\mathcal{N}(8,1)$. Find the decision rule that you can use to discriminate the two types of objects using the Likelihood Ratio Test if there is the same penalty for making all wrong decisions.

How would the decision rule be affected if there are twice more objects of type A than the objects of type B?

8 Understand Pattern Recognition & Decision Theory

Solution:

2 types of objects: $\{\omega_1$ for type A, ω_2 for type B$\}$; let size be x,

Type A: $p(x|\omega_1) \sim \mathcal{N}(2,1) \Rightarrow p(x|\omega_1) = \frac{1}{\sqrt{2\pi}} \exp[-\frac{1}{2}(x-2)^2]$

Type B: $p(x|\omega_2) \sim \mathcal{N}(8,1) \Rightarrow p(x|\omega_2) = \frac{1}{\sqrt{2\pi}} \exp[-\frac{1}{2}(x-8)^2]$

From the question, we know $P(\omega_2) = P(\omega_1)$.

Let penalty for making mistake be k, then $\lambda_{11} = \lambda_{22} = 0$, $\lambda_{12} = \lambda_{21} = k$.

Likelihood Ratio Test:

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})} \frac{P(\omega_2)}{P(\omega_1)} \Rightarrow \frac{\exp[-\frac{1}{2}(x-2)^2]}{\exp[-\frac{1}{2}(x-8)^2]} > 1$$

$$\begin{aligned} \text{take natural log, } -\frac{1}{2}[(x-2)^2 - (x-8)^2] &> 0 \\ \text{change sign, remove } \frac{1}{2}\text{, we get} (x-2)^2 - (x-8)^2 &< 0 \\ 12x - 60 &< 0 \\ \text{Therefore, } x &< 5 \end{aligned}$$

The rule: Decide ω_1 if $x < 5$, otherwise decide ω_2

8 Understand Pattern Recognition & Decision Theory

Solution:

If there are twice more objects of type A than objects of type B out there,

$$\frac{P(\omega_1)}{P(\omega_2)} = 2 \Rightarrow P(\omega_1) = 2P(\omega_2) \Rightarrow \frac{P(\omega_2)}{P(\omega_1)} = \frac{1}{2}$$

Following the previous solution, we now have
Likelihood Ratio Test:

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{(\lambda_{12} - \lambda_{22})}{(\lambda_{21} - \lambda_{11})} \frac{P(\omega_2)}{P(\omega_1)} = \frac{1}{2} \Rightarrow \frac{\exp[-\frac{1}{2}(x-2)^2]}{\exp[-\frac{1}{2}(x-8)^2]} > \frac{1}{2}$$

take natural log, $-\frac{1}{2}[(x-2)^2 - (x-8)^2] > \ln\frac{1}{2}$

change sign, $\times 2$, we get $(x-2)^2 - (x-8)^2 < -2\ln\frac{1}{2}$

$$12x - 60 < -2\ln\frac{1}{2}$$

Therefore, $x < \frac{60 - 2\ln(1/2)}{12} = 5.16$

The rule: Decide ω_1 if $x < 5.16$, otherwise decide ω_2

The decision boundary is shifted towards class ω_2.

8 Understand Pattern Recognition & Decision Theory

- Numerical Example 2

A set of patterns is to be classified into 2 classes based on a scalar feature. We know that the conditional density functions of the feature for the pattern classes can be represented as two Gaussians, $\mathcal{N}(2,3)$ and $\mathcal{N}(4,1)$. The patterns from the two classes are equal likely to be seen. However, if a pattern from class 1 being wrongly classified into class 2, the cost associated is 1 while the cost for a pattern from class 2 being wrongly classified into class 1 is $\sqrt{3}$. Using the Likelihood Ratio Test, find the decision rule that minimizes the probability of classification error.

8 Understand Pattern Recognition & Decision Theory

Solution:

We have 2 classes: $\{\omega_1$ for class 1, ω_2 for class 2$\}$; let the feature be x,

Class 1: $p(x|\omega_1) \sim \mathcal{N}(2,3) \Rightarrow p(x|\omega_1) = \frac{1}{\sqrt{2\pi}\sqrt{3}} \exp[-\frac{1}{2}\frac{(x-2)^2}{3}]$

Class 2: $p(x|\omega_2) \sim \mathcal{N}(4,1) \Rightarrow p(x|\omega_2) = \frac{1}{\sqrt{2\pi}} \exp[-\frac{1}{2}(x-4)^2]$

From the question, we know $P(\omega_2) = P(\omega_1)$.

Also, $\lambda_{11} = \lambda_{22} = 0$, $\lambda_{12} = 1$ and $\lambda_{21} = \sqrt{3}$.

Likelihood Ratio Test:

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})}\frac{P(\omega_2)}{P(\omega_1)} \Rightarrow \frac{\frac{1}{\sqrt{2\pi}\sqrt{3}} \exp[-\frac{1}{2}\frac{(x-2)^2}{3}]}{\frac{1}{\sqrt{2\pi}} \exp[-\frac{1}{2}(x-4)^2]} > \frac{1}{\sqrt{3}}$$

Canceling out $\frac{1}{\sqrt{2\pi}}$, $\times\sqrt{3}$, we have : $\frac{\exp[-\frac{1}{2}\frac{(x-2)^2}{3}]}{\exp[-\frac{1}{2}(x-4)^2]} > 1$

take natural log, $-\frac{1}{2}\left[\frac{(x-2)^2}{3} - (x-4)^2\right] > 0$

$\times 6$, we get : $-(x-2)^2 + 3(x-4)^2 > 0$

expand and divide by 2 : $x^2 - 10x + 22 > 0$

8 Understand Pattern Recognition & Decision Theory

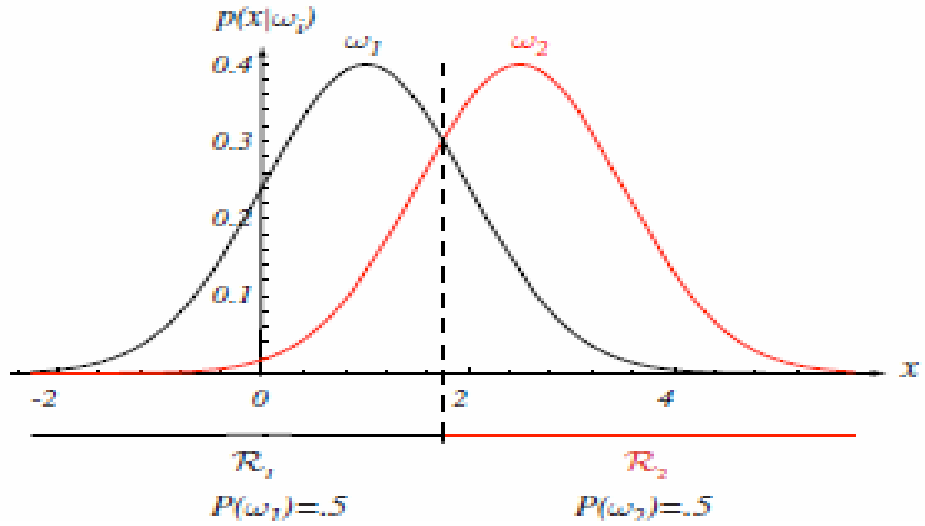
- Numerical Example 2

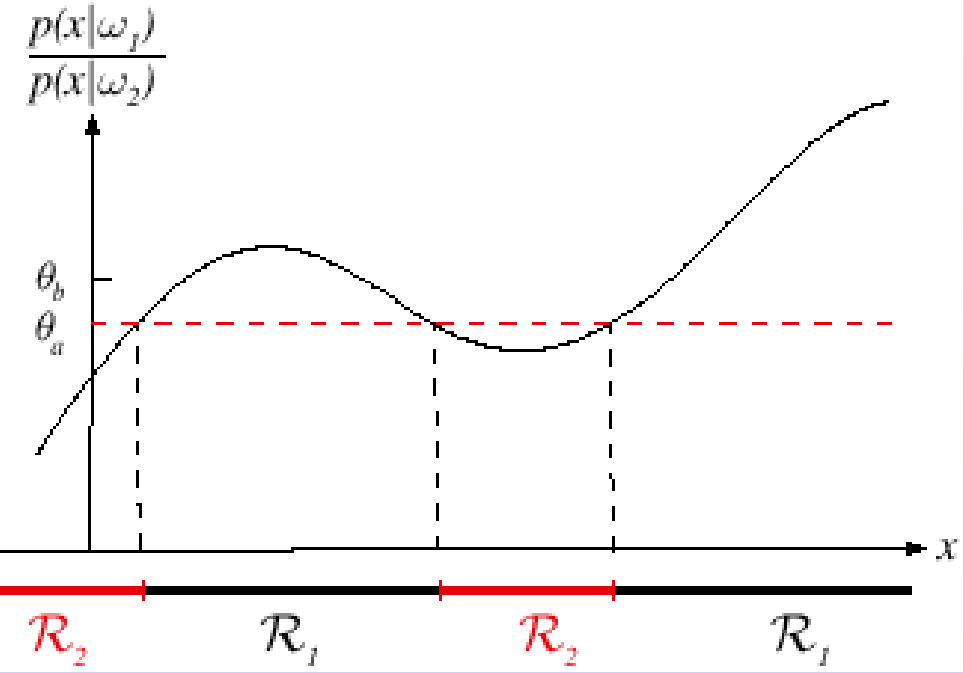
Solution (continued):
Solve the quadratic equation:
Let $f(x) = x^2 - 10x + 22$.

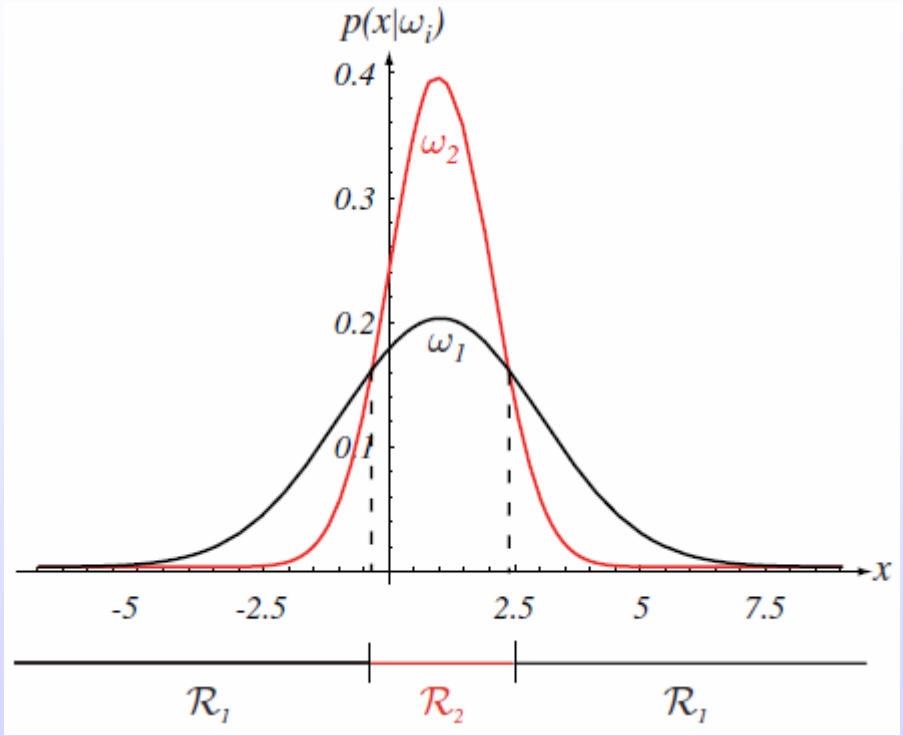
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \Rightarrow x = \frac{10 \pm \sqrt{10^2 - 4(22)}}{2} = 3.27, 6.73$$

We have $f'(x) = 2x - 10$.
Setting $f'(x) = 0$, we have $x = 5$.
We know that $f(5) < 0$.
Hence, by testing the regions, we find that
$f(x) < 0$ for $3.27 < x < 6.73$, and $f(x) > 0$ for $x < 3.27$ and $x > 6.73$.
Therefore, we have the following rule:
Decide ω_1 if $x < 3.27$ or $x > 6.73$, otherwise decide ω_2

8 Understand Pattern Recognition & Decision Theory







- Connected and disconnected decision regions

9 Statistical Estimation & Machine Learning

Outline

- Nonparametric approach to estimate the probability distribution density
  - Parzen-window approach
  - k–nearest-neighbor kNN rule
- Parametric approach to estimate the probability distribution density
  - Maximum likelihood (ML) estimation
  - Multivariate Gaussian probability distribution density

9 Statistical Estimation & Machine Learning

- We understand that the best decision or optimal classification is:

$$\text{Decide } \omega_k = \arg\min_{\omega_i}[p(e_i \mid \mathbf{x})] = \arg\min_{\omega_i}[1 - p(\omega_i \mid \mathbf{x})]$$

$$= \arg\max_{\omega_i}[p(\omega_i \mid \mathbf{x})] = \arg\max_{\omega_i}[p(\omega_i)p(\mathbf{x} \mid \omega_i)]$$

- To design an automatic pattern recognition system, we need know the probability distribution/density function (PDF) for all values of $\mathbf{x}$ so that the system can make decision for any value of received data $\mathbf{x}$. In practice, the PDF is often unknown and can only be estimated by collected examples/samples $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ for the design of the pattern recognition system.
- Determining some rules or some deterministic values on a random variable $\mathbf{x}$ based on a set of training samples $D=\{\mathbf{x}_i\}$ is the task of statistical estimation or machine learning.

9 Statistical Estimation & Machine Learning

- The probability distribution/density function (PDF) is the complete information about a random variable.
- It is difficult to estimate the posterior probability $p(\omega_k|\mathbf{x})$ function directly. So we learn the prior probability $p(\omega_k)$ and the class-conditional probability function $p(\mathbf{x}|\omega_k)$.
- The c prior probabilities can be easily estimated by

$$\hat{p}(\omega_k) = n_k / n$$

where n_k and n are the number of training samples of class ω_k and the total number of training samples.

- As the estimation method is the same for all classes, we simplifying the notation of $p(\mathbf{x}|\omega_k)$ to $p(\mathbf{x})$ and suppose we have n samples $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ drawn independently and identically distributed (i.i.d.) according to the probability law $p(\mathbf{x})$.

9 Statistical Estimation & Machine Learning

- According to the definition of probability density function, the probability P that $\mathbf{x}$ falls in a region R is:

$$P(\mathbf{x}) = \int_{R_\mathbf{x}} p(\mathbf{t})d\mathbf{t} \approx p(\mathbf{x})V$$

- where V is the volume of the region $R_\mathbf{x}$.
- If out of all n training samples of this class, there are k samples fall in the region R, the estimate of P is then. $\hat{P}(\mathbf{x}) = \frac{k}{n}$
- Therefore, the estimate of the PDF $p(\mathbf{x})$ is

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- This is called nonparametric approach to the estimation of the probability density function because no assumption of the model is applied.

9 Statistical Estimation & Machine Learning

- The procedure to estimate the PDF based on n training samples is: Given a value of $\mathbf{x}$, select a region/cell of size (volume) V centered at $\mathbf{x}$, count the number of samples k fall in the region/cell. The probability density $p(\mathbf{x})$ at $\mathbf{x}$ is then estimated as :

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- Obviously, different shape and size of the region/cell lead to different estimate of PDF.
- In general, $\mathbf{x}$ is multiple dimensional $\mathbf{x}=[x_1, x_2, \ldots, x_d]$. If we choose a d-dimensional hypercube of the side length h as the region, a sample $\mathbf{x}_i=[x_{i1}, x_{i2}, \ldots, x_{id}]$ will fall into the hypercube if

$$\frac{\left|x_j - x_{ij}\right|}{h} < \frac{1}{2} \quad \text{for } \forall j = 1,\ 2,\ldots,\ d$$

9 Statistical Estimation & Machine Learning

- Therefore, we can express the number of samples falling into the cell k mathematically as

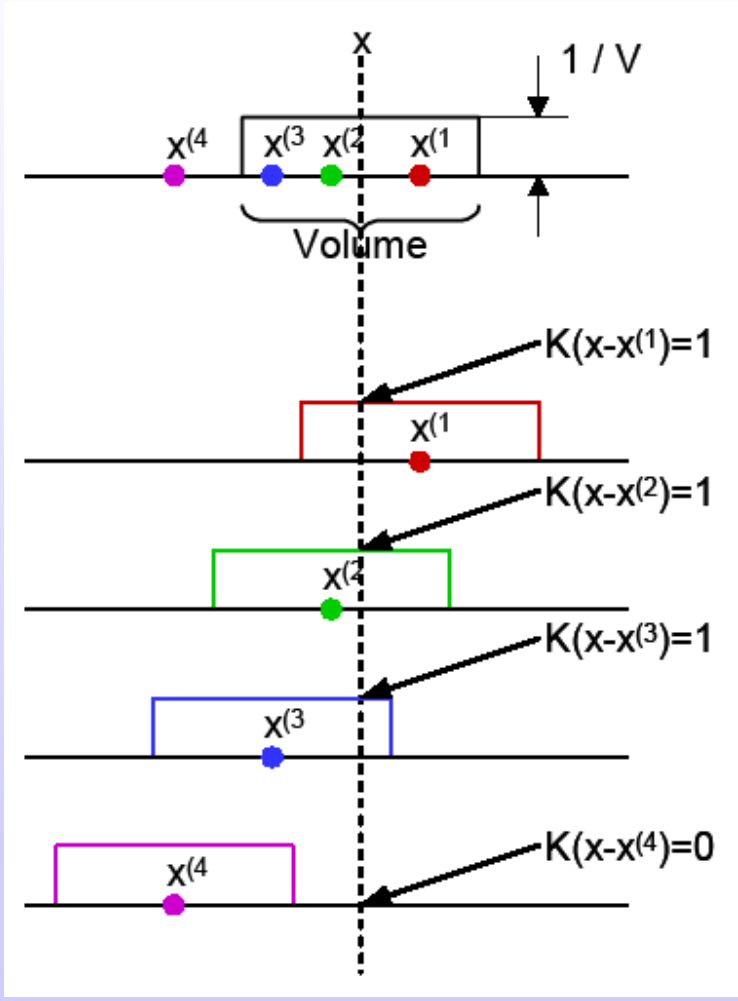
$$k = \sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

- Where the kernel function called Parzen-window is defined as

$$K(\mathbf{u}) = \begin{cases} 1 & \text{if } |u_j| < 1/2 \quad \text{for } \forall j = 1,\ 2,\ldots,\ d \\ 0 & \text{otherwise} \end{cases}$$

- The probability density $p(\mathbf{x})$ at $\mathbf{x}$ is then estimated as :

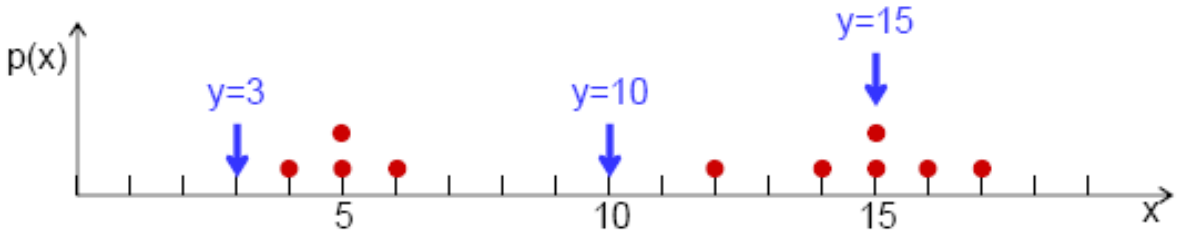
$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$



9 Statistical Estimation & Machine Learning

- **Given the dataset below, use Parzen windows to estimate the density p(x) at y=3,10,15. Use a bandwidth of h=4**
  - $X = \{x^{(1}, x^{(2}, \ldots x^{(N}\} = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$
- **Solution**
  - Let's first draw the dataset to get an idea of what numerical results we should expect

Example:



- Let's now estimate p(y=3):

$$p_{KDE}(y=3)=\frac{1}{Nh^D}\sum_{n=1}^{N}K\left(\frac{y-x^{(n}}{h}\right)=\frac{1}{10\times4^1}\left[K\underbrace{\left(\frac{3-4}{4}\right)}_{-1/4}+K\underbrace{\left(\frac{3-5}{4}\right)}_{-1/2}+K\underbrace{\left(\frac{3-5}{4}\right)}_{-1/2}+K\underbrace{\left(\frac{3-6}{4}\right)}_{-1}+\ldots+K\underbrace{\left(\frac{3-17}{4}\right)}_{-13/4}\right]=$$

$$=\frac{1}{10\times4^1}[1+0+0+0+0+0+0+0+0+0]=\frac{1}{10\times4}=0.025$$

- Similarly

$$p_{KDE}(y=10)=\frac{1}{10\times4^1}[0+0+0+0+0+0+0+0+0+0]=\frac{0}{10\times4}=0$$

$$p_{KDE}(y=15)=\frac{1}{10\times4^1}[0+0+0+0+0+1+1+1+1+0]=\frac{4}{10\times4}=0.1$$

Note here:

$$K(\mathbf{u})=\begin{cases}1 \text{ if } |u_j|<1/2\\ 0 \text{ otherwise}\end{cases}$$

9 Statistical Estimation & Machine Learning

- The rectangular kernel function produces unsmoothed PDF due to the unsmooth rectangular kernel function. In fact, we can choose any function as the kernel so long as:

$$K(\mathbf{u}) \geq 0$$

$$\int_{-\infty}^{\infty} K(\mathbf{u})d\mathbf{u} = 1 \quad \text{i.e.} \quad \int_{-\infty}^{\infty} \frac{1}{h^d} K(\frac{\mathbf{x}-\mathbf{x}_i}{h})d\mathbf{x} = 1$$

$$\because \int_{-\infty}^{\infty} K(\mathbf{u})d\mathbf{u} = \int_{-\infty}^{\infty} K(\frac{\mathbf{x}-\mathbf{x}_i}{h})d\frac{\mathbf{x}-\mathbf{x}_i}{h} = \int_{-\infty}^{\infty} \frac{1}{h^d} K(\frac{\mathbf{x}-\mathbf{x}_i}{h})d\mathbf{x}$$

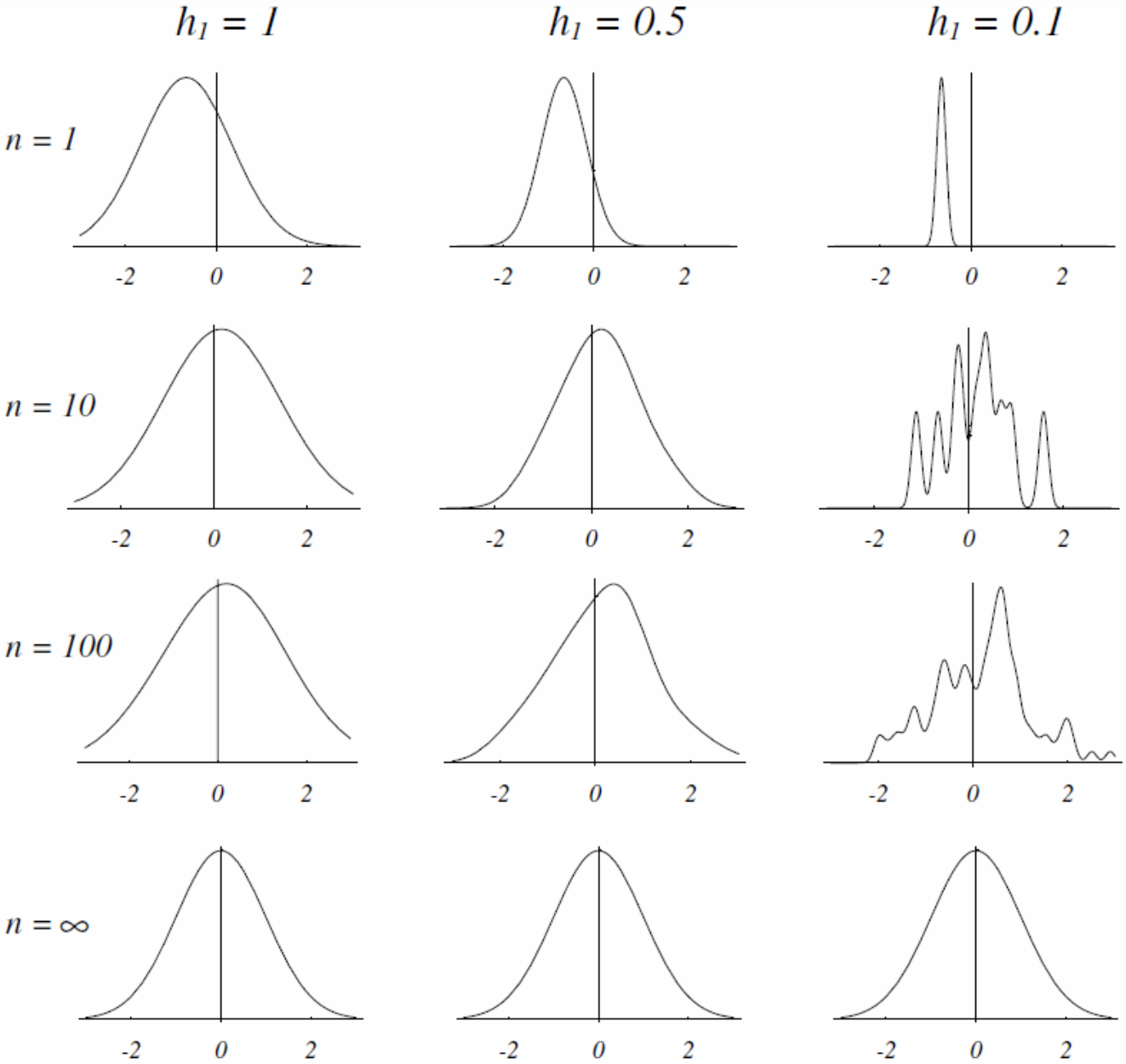
- Therefore, any DPF function can be served as the kernel function. This approach is called Parzen-window approach, it in fact interpolates the discrete points $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ into a continuous function PDF $p(\mathbf{x})$.

9 Statistical Estimation & Machine Learning

1-D example.

$$K(\mathbf{u}) = N(0,\mathbf{I})$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}\mathbf{u}^T\mathbf{u}\right]$$

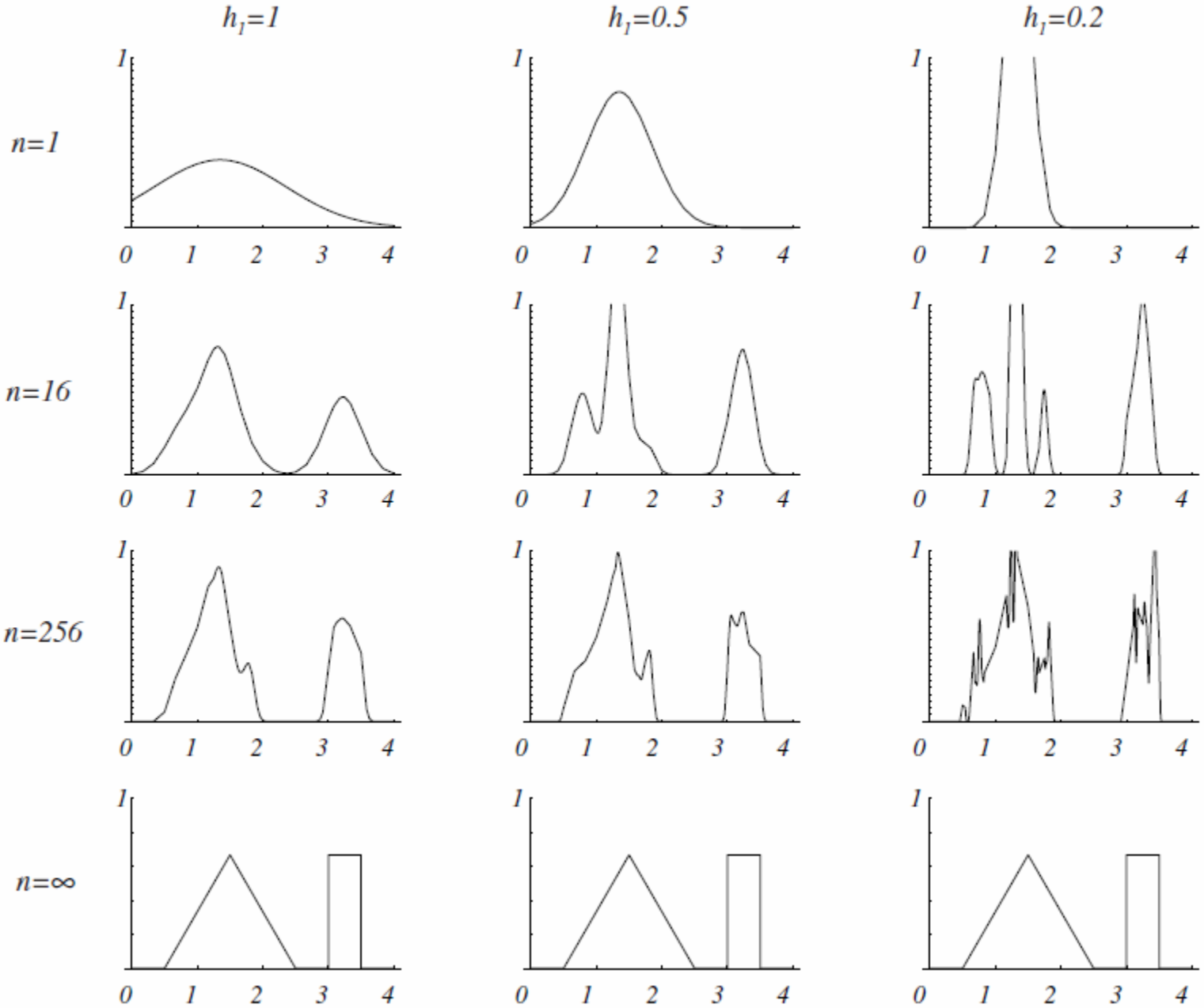


exdjiang@ntu.edu.sg

9 Statistical Estimation & Machine Learning

another
1-D example.

bimodal
distribution



exdjiang@ntu.edu

9 Statistical Estimation & Machine Learning

- A single 2-D training sample with Gaussian kernel function of different width h.

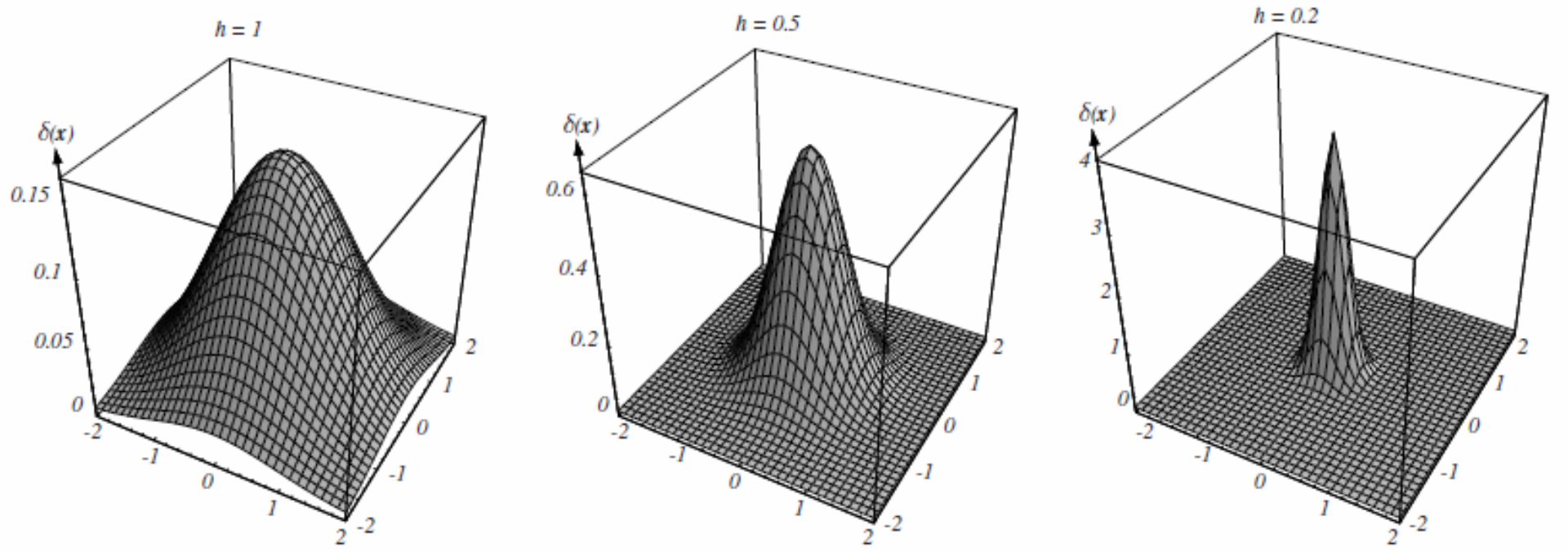


FIGURE 4.3. Examples of two-dimensional circularly symmetric normal Parzen windows for three different values of h. Note that because the $\delta(\mathbf{x})$ are normalized, different

9 Statistical Estimation & Machine Learning

- 5 2-D training samples with Gaussian kernel function of different width h.

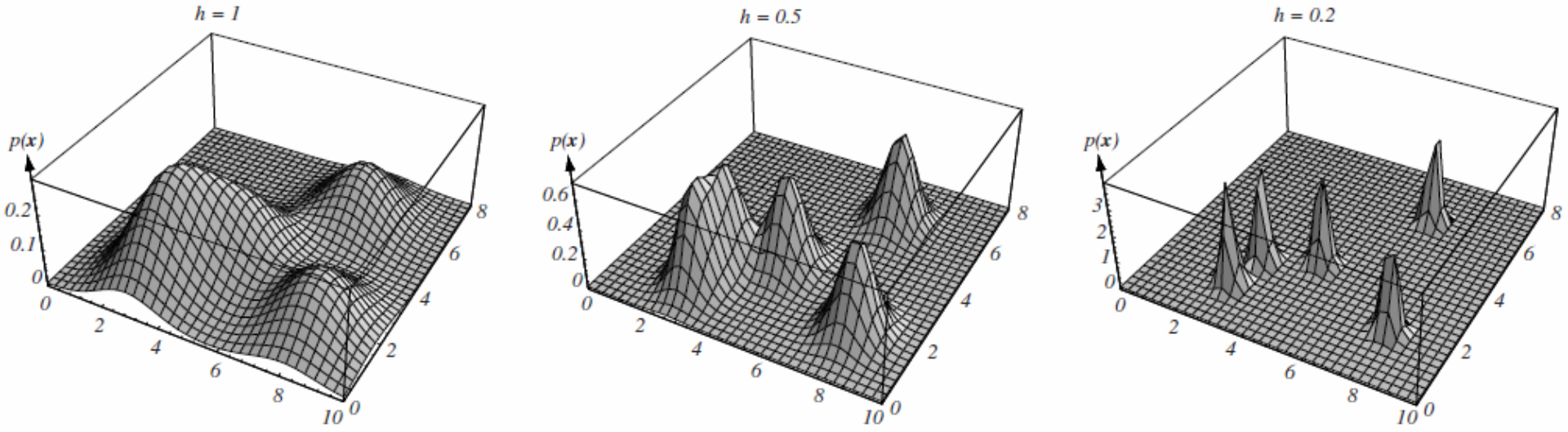


FIGURE 4.4. Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each distribution.

- The probabilistic neural networks and the RBF neural networks are almost equivalent to the Parzen-window approach.

9 Statistical Estimation & Machine Learning

- To estimate the PDF by $$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

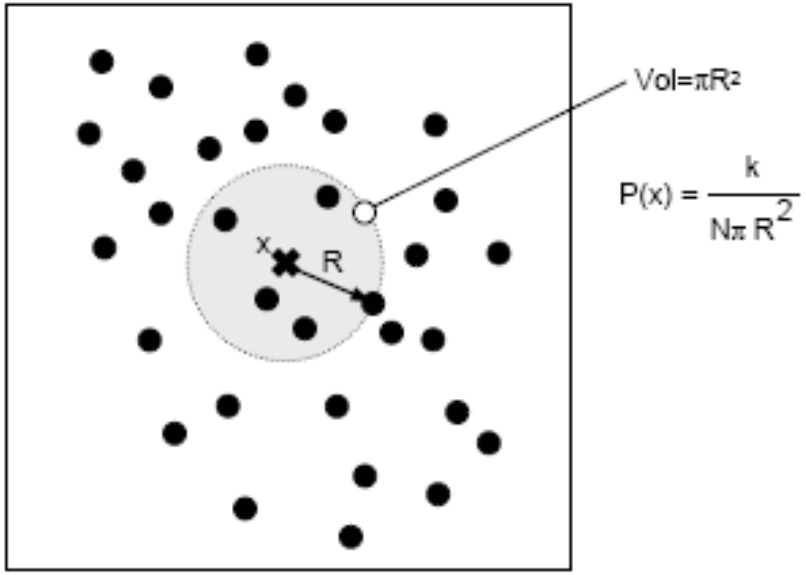
  the Parzen-window approach selects a region/cell of fixed size (volume) V centered at $\mathbf{x}$ and counts the number of samples k fall in the region/cell for the estimation of PDF.

- We can also select the fixed number of samples k and compute the size/volume that just encloses k samples to estimate the PDF by

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- This approach is called

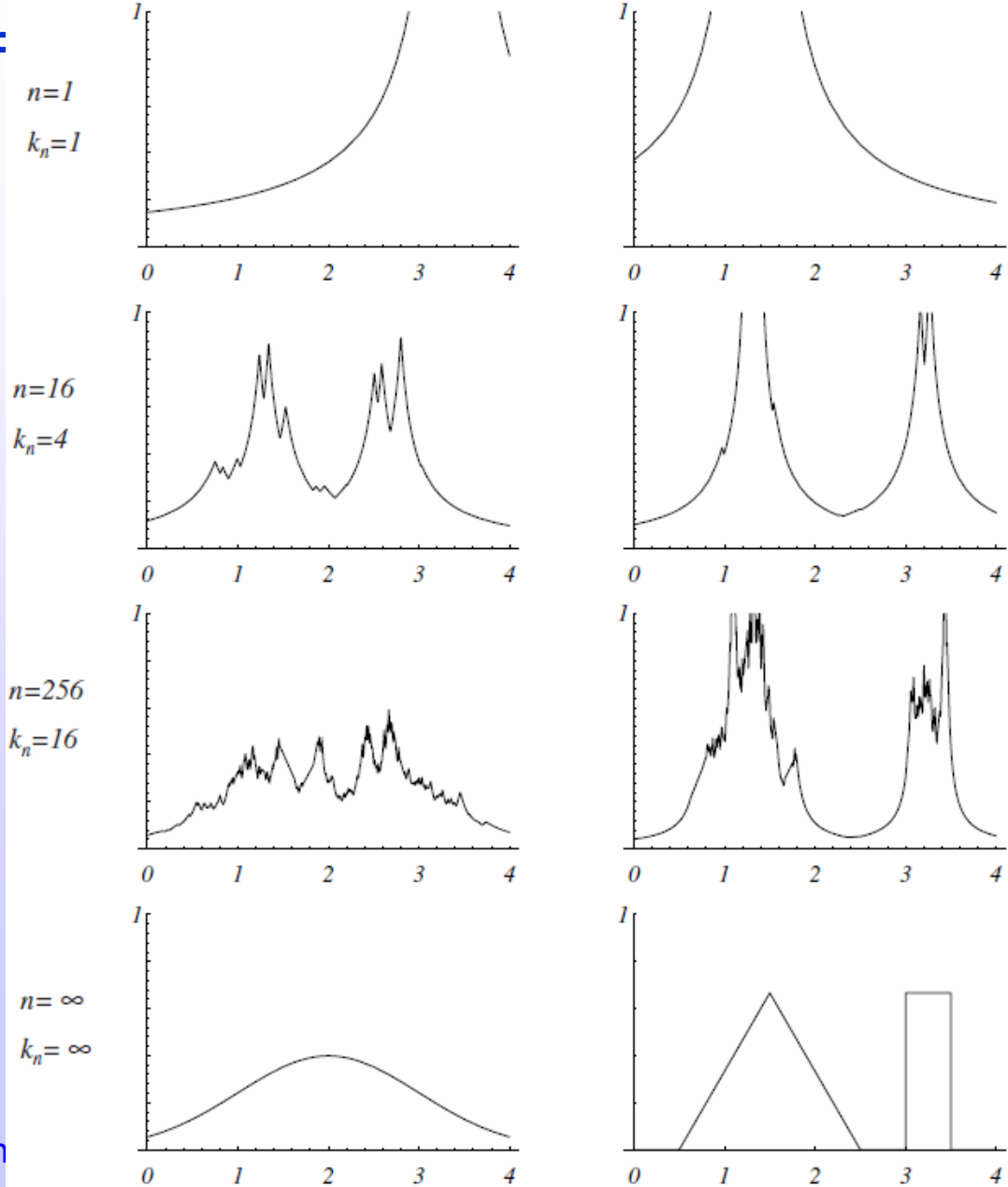
  k–nearest-neighbor kNN estimation.



9 Statistical Estimation & Machine Learning

- Several k-nearest-neighbor estimates of two uni-dimensional densities:

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$



9 Statistical Estimation & Machine Learning

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

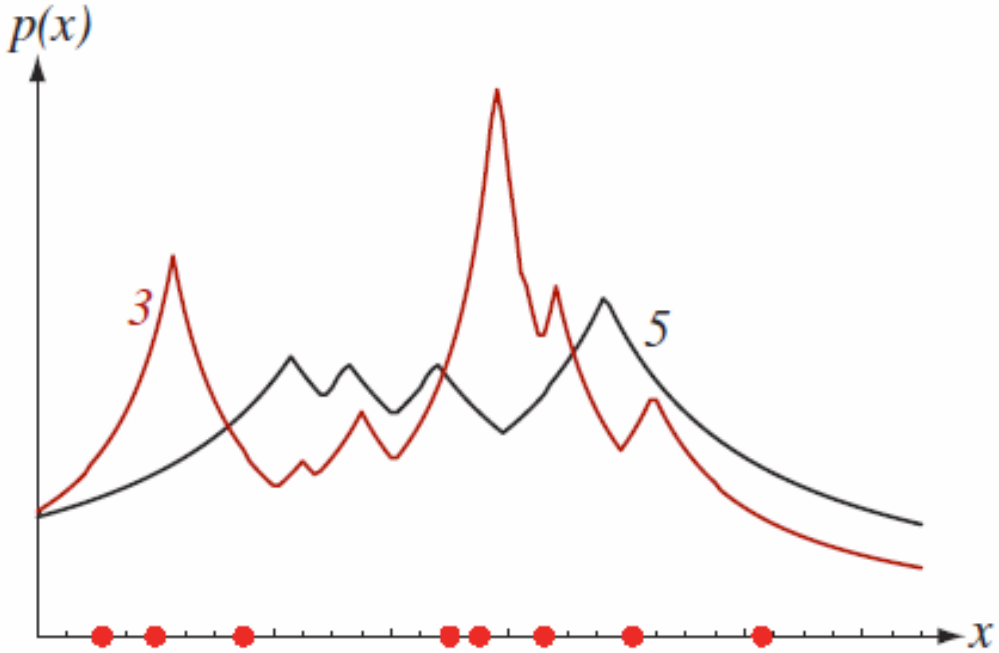


FIGURE 4.10. Eight points in one dimension and the k-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally lie *away* from the positions of the prototype points. From: Richard

9 Statistical Estimation & Machine Learning

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

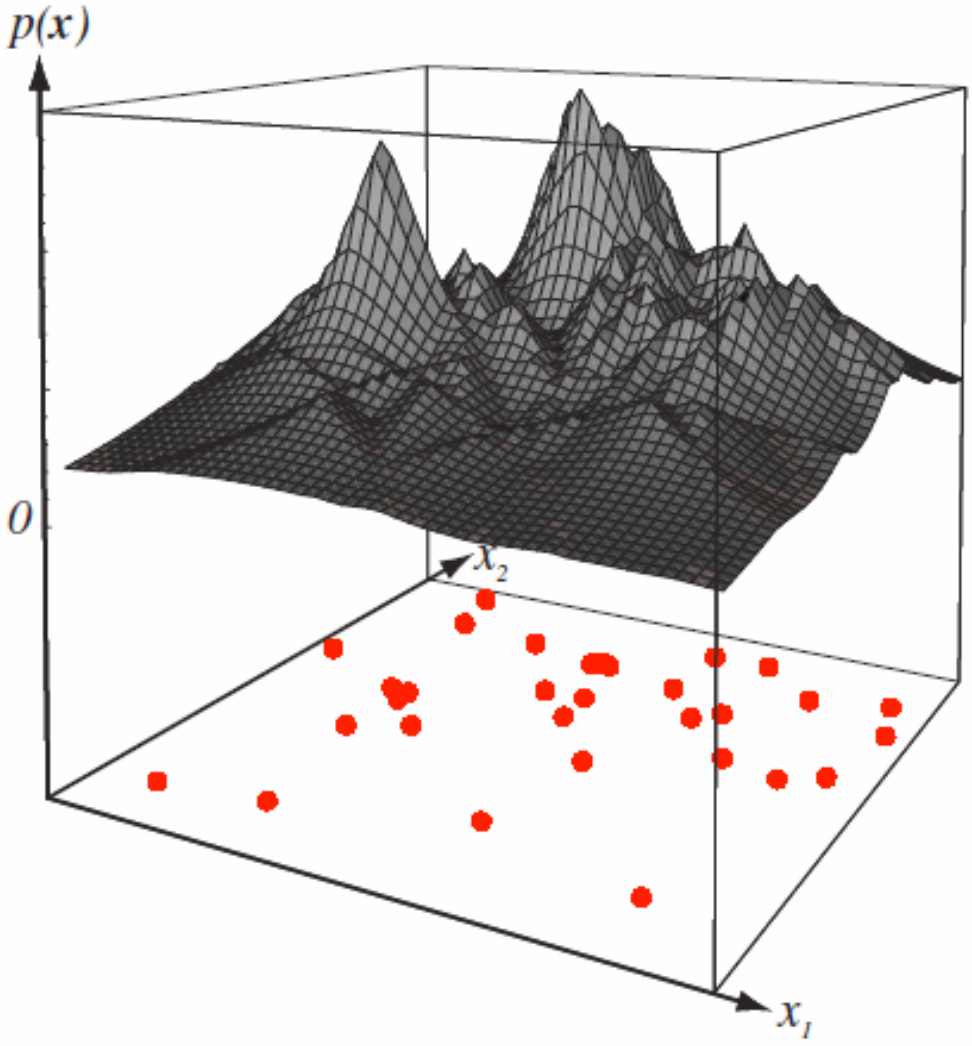


FIGURE 4.11. The k-nearest-neighbor estimate of a two-dimensional density for $k = 5$.

9 Statistical Estimation & Machine Learning

- The KNN technique can also be used for estimation of a-posteriori probabilities $P(\omega_i|\mathbf{x})$ from a set of n labeled samples. (Compare to PNNs and Parzen window estimates.)
- Suppose that we place a cell of volume V around $\mathbf{x}$ and capture k samples, k_i of which turn out to be labeled ω_i.
- An estimate for the joint probability is $p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV}$.
- Hence, we can estimate $P(\omega_i|\mathbf{x})$ by

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

$$\hat{p}(\omega_i \mid \mathbf{x}) = \hat{P}(\omega_i)\hat{p}(\mathbf{x} \mid \omega_i) / \hat{p}(\mathbf{x})$$

$$= \frac{n_i}{n}\frac{k_i}{n_i V(k)}\frac{nV(k)}{k} = \frac{k_i}{k}$$

$$\frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{k}.$$

$$\hat{P}(\omega_i) = \frac{n_i}{n}$$

$$\hat{p}(\mathbf{x} \mid \omega_i) = \frac{k_i}{n_i V(k)}$$

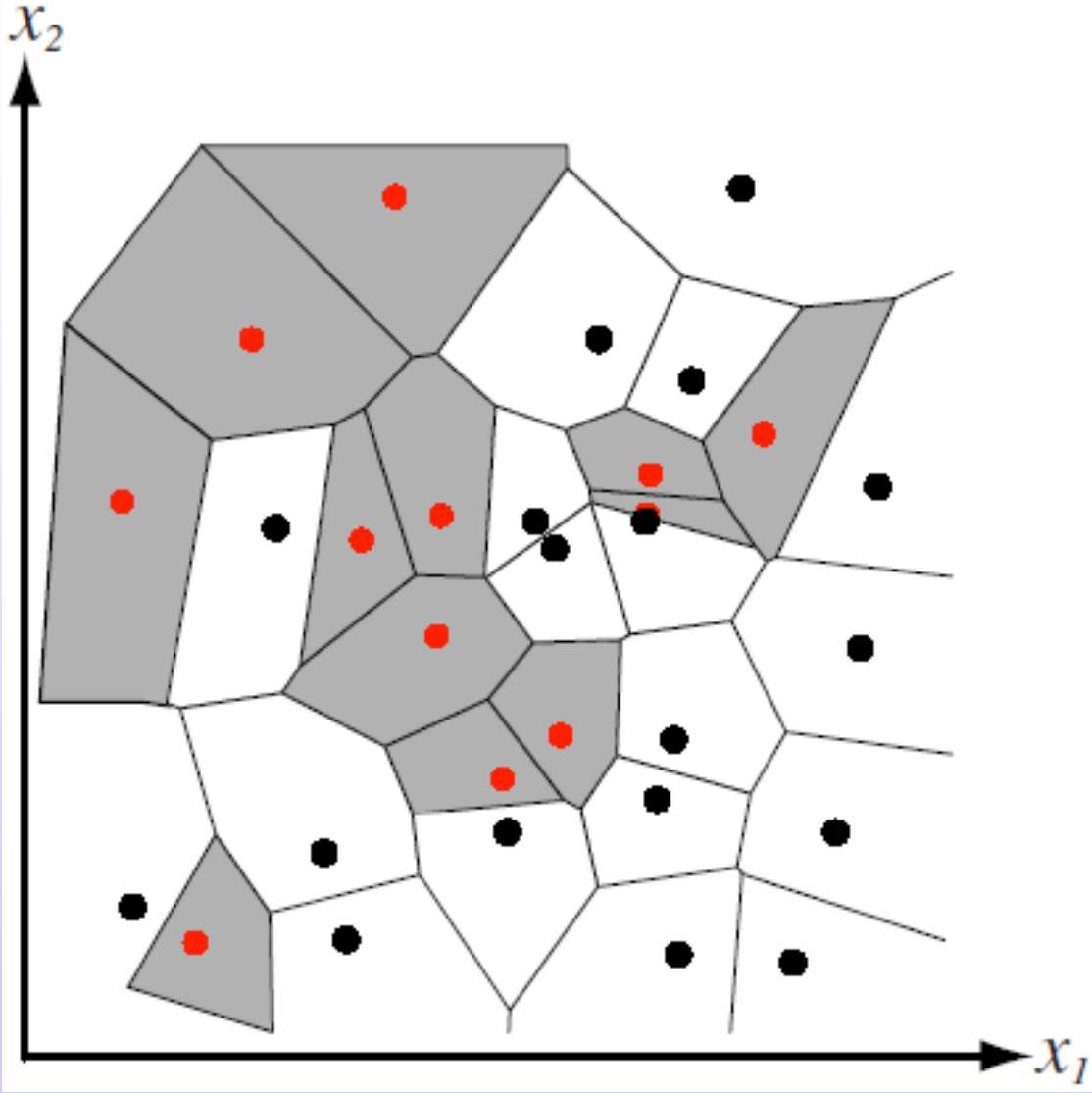
9 Statistical Estimation & Machine Learning

- Denote by $\mathcal{D}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ a set of labeled **prototypes** or training samples.
- Let $\mathbf{x}^*$ be the prototype nearest to $\mathbf{x}$.
- Then **the nearest-neighbor (NN) rule** for classifying $\mathbf{x}$ is to assign it the label associated with $\mathbf{x}^*$.
- More formally if we have a set of labeled training samples $\{(\mathbf{x}_1, \theta_1), \ldots, (\mathbf{x}_n, \theta_n)\}$, where each θ_i is one of the labels $\omega_1, \ldots, \omega_c$, then the NN decision rule is

$$\alpha_{nn}(\mathbf{x}) = \theta_k : k = \arg\min_i ||\mathbf{x} - \mathbf{x}_i||.$$

9 Statistical Estimation & Machine Learning

- Classification boundary of 1st-NN classifier, also called NN classifier.



9 Statistical Estimation & Machine Learning

- The error rate of NN classifier P for very large number of training samples are bounded as

$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$

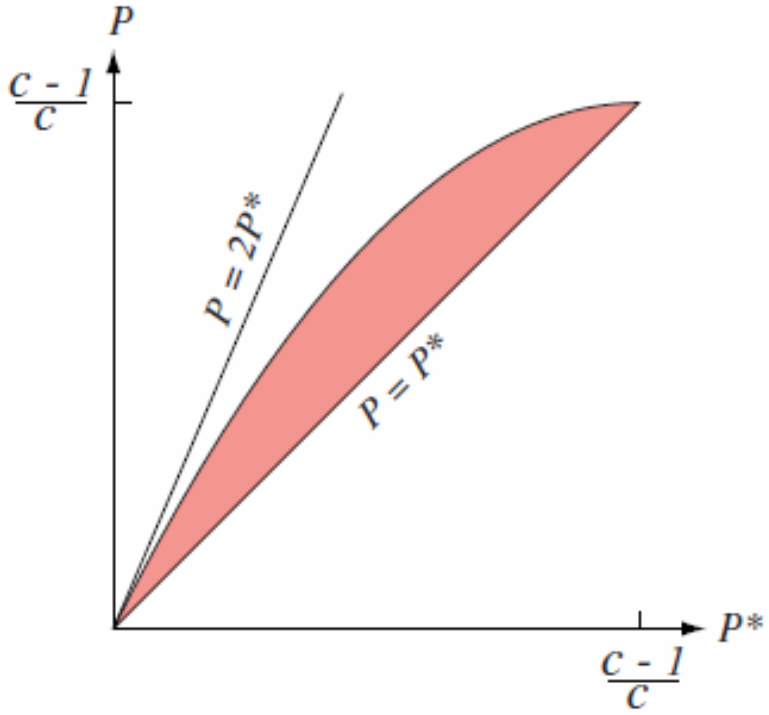


FIGURE 4.14. Bounds on the nearest-neighbor error rate P in a c-category problem given infinite training data, where P^* is the Bayes error (Eq. 52). At low error rates, the nearest-neighbor error rate is bounded above by twice the Bayes rate. From: Richard O.

9 Statistical Estimation & Machine Learning

- Generalization of the NN rule.
- The k_n nearest neighbors rule: Given a set of training samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a test point $\mathbf{x}$, find k training points closest to $\mathbf{x}$, $\mathbf{x}_1^*, \ldots, \mathbf{x}_k^*$. Collect the labels associated $\theta_1^*, \ldots, \theta_k^*$ and classify $\mathbf{x}$ to the class which has the greatest number of representatives in $\theta_1^*, \ldots, \theta_k^*$.
- In other words, the classification is performed by taking the majority vote among k nearest neighbors of $\mathbf{x}$.

9 Statistical Estimation & Machine Learning

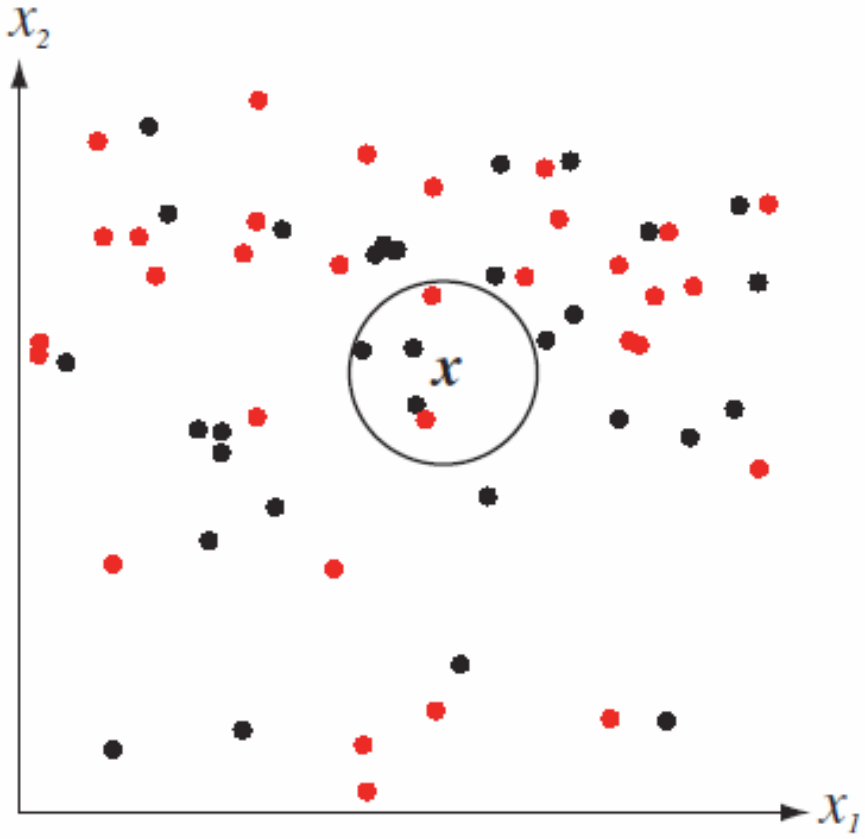


FIGURE 4.15. The k-nearest-neighbor query starts at the test point $\mathbf{x}$ and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point $\mathbf{x}$ would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

9 Statistical Estimation & Machine Learning

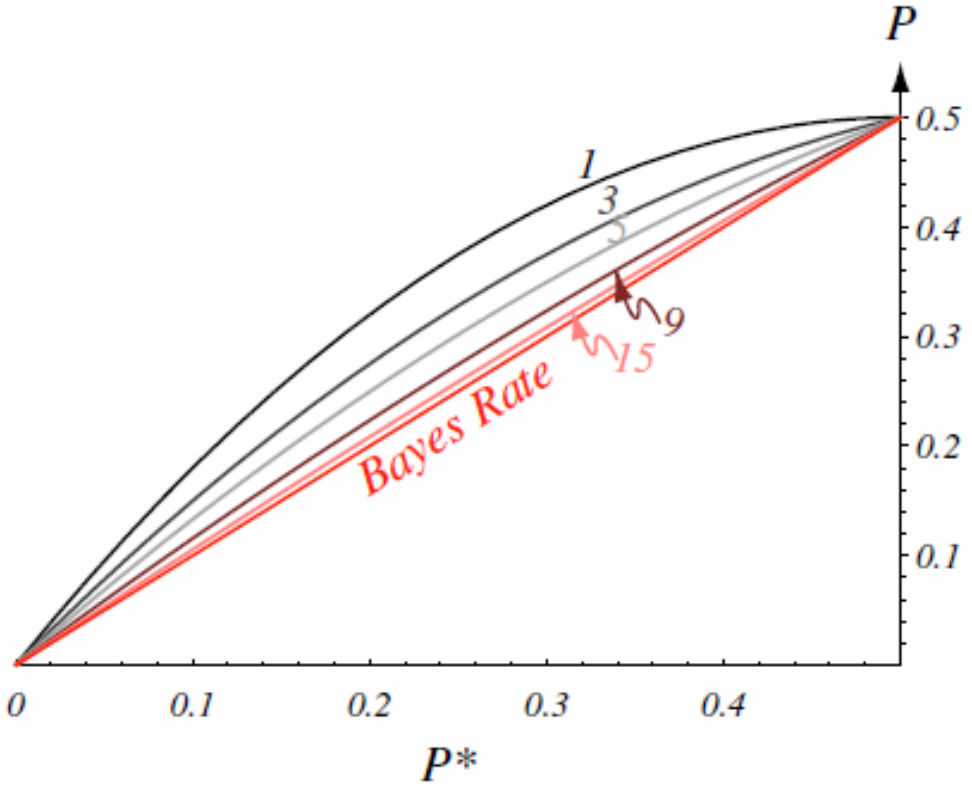


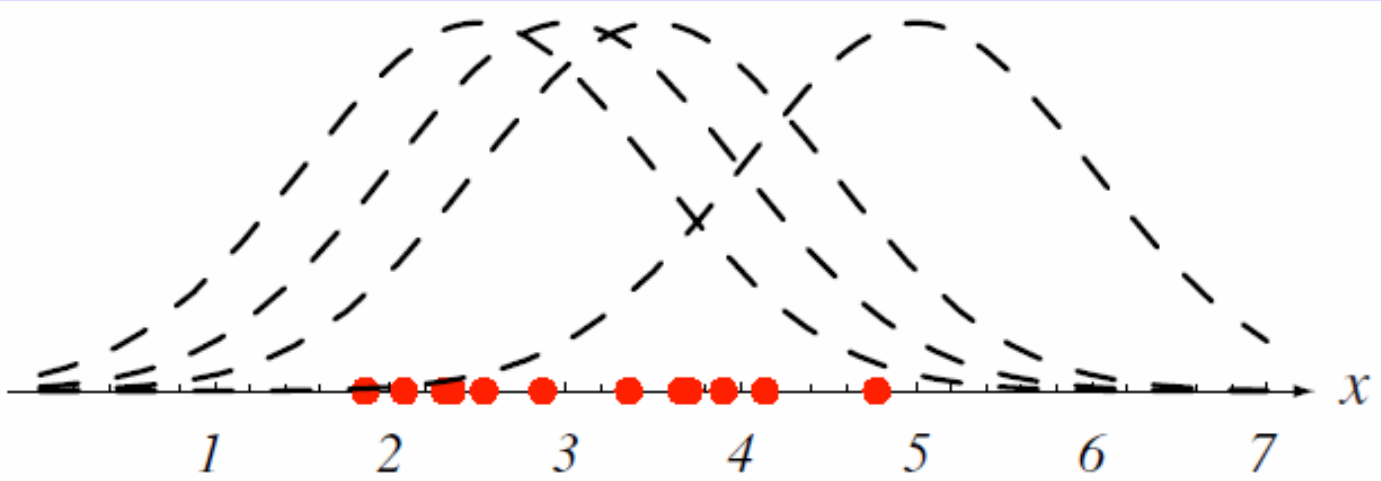
FIGURE 4.16. The error rate for the k-nearest-neighbor rule for a two-category problem is bounded by $C_k(P^*)$ in Eq. 54. Each curve is labeled by k; when $k = \infty$, the estimated probabilities match the true probabilities and thus the error rate is equal to the Bayes rate, that is, $P = P^*$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

9 Statistical Estimation & Machine Learning

- We see that the learned conditional PDF from training samples could greatly deviate from the true PDF of the population, especially in case of small number of training samples.
- If our general knowledge about the problem permits us to model the conditional PDF, i.e. using a mathematical analytical function to represent the PDF with unknown parameter. The severity of these problems can be reduced significantly. Here we parameterize the conditional PDF, which is called parametric method.
- Suppose, for example, we can reasonably assume that $p(\mathbf{x}|\omega_k)$ is a Gaussian density with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Although we do not know their values, this knowledge simplifies the problem from estimating an unknown function $p(\mathbf{x}|\omega_k)$ to estimating the unknown parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ only.

9 Statistical Estimation & Machine Learning

- Now we will use a set of training samples $D=\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ drawn independently from the probability density $p(\mathbf{x}|\boldsymbol{\theta})$ to estimate the unknown parameter vector $\boldsymbol{\theta}$.
- Lets see an example to generate idea how to estimate the parameter of a given probability density $p(\mathbf{x}|\boldsymbol{\theta})$ reasonably based on the training data D.
- The graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four PDF with 4 different means are shown in dashed lines.
- Which PDF you should choose?



exdjiang@ntu.edu.sg

9 Statistical Estimation & Machine Learning

- Now we formulate our idea mathematically.
- Obviously, the probability that a sample $\mathbf{x}_k$ occurs is $p(\mathbf{x}_k|\boldsymbol{\theta})$. As all samples in the training set are independently collected (occur), the probability that all samples occur is

$$p(D\,|\,\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k\,|\,\boldsymbol{\theta})$$

- Intuitively, we should select the parameter so that the probability density $p(\mathbf{x}|\boldsymbol{\theta})$ best supports the actually observed training samples, i.e. to make the probability of all training data occur $p(D|\boldsymbol{\theta})$ maximal. Note that $p(D|\boldsymbol{\theta})$ is called the likelihood of $\boldsymbol{\theta}$ with respect to the set of samples D. Thus, this method is called the maximum likelihood (ML) estimation.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(D\,|\,\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{k=1}^{n} p(\mathbf{x}_k\,|\,\boldsymbol{\theta})$$

9 Statistical Estimation & Machine Learning

- It is often not easy to get an analytical solution of

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(D \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{k=1}^{n} p(\mathbf{x}_k \mid \boldsymbol{\theta})$$

  due to the multiplication of the functions of $\boldsymbol{\theta}$ and $p(\mathbf{x}_k|\boldsymbol{\theta})$ is often nonlinear function of $\boldsymbol{\theta}$.

- Since the logarithm is monotonically increasing, maximizing the logarithm of a function also maximizes the function itself. The logarithm has nice property that converts the multiplication into summation and simplifies the exponential function.

- Thus, we maximize the log-likelihood instead of maximizing the likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{k=1}^{n} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta})$$

9 Statistical Estimation & Machine Learning

- The solution can be found by the standard methods of differential calculus: Solving the equation that the gradient is zero.

$$\nabla_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = 0$$

$$\sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta}) = 0$$

- If the number of parameters to be estimated is q, then $\boldsymbol{\theta}$ is a q-component vector $\boldsymbol{\theta}=(\theta_1, \theta_2, \ldots, \theta_q)^T$. The gradient is a vector that contains partial differentiation against all components of $\boldsymbol{\theta}$.

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \triangleq \begin{pmatrix} \dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_q} \end{pmatrix}$$

9 Statistical Estimation & Machine Learning

- To see how maximum likelihood methods results apply to a specific case, suppose that the samples are drawn from a multivariate Gaussian population with unknown mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

- The log-likelihood of a single sample is

$$\ln p(\mathbf{x}_k\,|\,\boldsymbol{\theta}) = -\frac{1}{2}\ln[(2\pi)^d\,|\boldsymbol{\Sigma}|] - \frac{1}{2}(\mathbf{x}_k-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_k-\boldsymbol{\mu})$$

- Consider first the univariate case with $\boldsymbol{\theta}=(\theta_1, \theta_2)^T=(\mu, \sigma^2)^T$.

$$p(x\,|\,\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

9 Statistical Estimation & Machine Learning

- Here the log-likelihood of a single sample is simplified as

$$\ln p(x_k \mid \boldsymbol{\theta}) = -\frac{1}{2}\ln[2\pi\sigma^2] - \frac{1}{2\sigma^2}(x_k - \mu)^2$$

- Its derivative is

$$\nabla_{\boldsymbol{\theta}} \ln p(x_k \mid \boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{\sigma^2}(x_k - \mu) \\ -\frac{1}{2\sigma^2} + \frac{(x_k - \mu)^2}{2\sigma^4} \end{pmatrix}$$

9 Statistical Estimation & Machine Learning

- Applying ML

$$\nabla_{\boldsymbol{\theta}} \ln p(D \mid \boldsymbol{\theta}) = \sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta}) = 0$$

- We have

$$\sum_{k=1}^{n} \frac{1}{\sigma^2}(x_k - \mu) = 0$$

$$-\sum_{k=1}^{n} \frac{1}{2\sigma^2} + \sum_{k=1}^{n} \frac{(x_k - \mu)^2}{2\sigma^4} = 0$$

- Solve these two equations we have the following maximum likelihood estimates

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{k=1}^{n} (x_k - \hat{\mu})^2$$

9 Statistical Estimation & Machine Learning

- While the analysis of the multivariate case is basically very similar, considerably more manipulations are involved. The result is that the maximum likelihood estimates for mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of multivariate Gaussian PDF

$$p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\,\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\hat{\boldsymbol{\mu}})(\mathbf{x}_k-\hat{\boldsymbol{\mu}})^T$$

10 Discriminant Functions and Classifiers

Outline

- Generalize the classification process to the discriminant function evaluation
- Discriminant function for multivariate Gaussian PDF
- Mahalanobis distance and Euclidean distance
- Linear classifier
- Minimum Mahalanobis distance classifier
- Minimum (Euclidean) distance classifier
- Classifiers by Sparse Representation

10 Discriminant Functions and Classifiers

- We understand that the Bayesian (optimal) classification with the 0/1 loss function maximizes the a posterior probability and hence minimizes the classification error by:

$$\text{Decide: } \omega_k = \arg\max_{\omega_i}[p(\omega_i \mid \mathbf{x})] = \arg\min_{\omega_i}[p(e_i \mid \mathbf{x})]$$

- Since $p(\omega_i \mid \mathbf{x}) = p(\omega_i)p(\mathbf{x} \mid \omega_i)p^{-1}(\mathbf{x})$

  and $p(\mathbf{x})$ is not a function of ω_i, the Bayesian (optimal) classification is to evaluate the called discriminant functions that can be defined as $g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln p(\omega_i)$

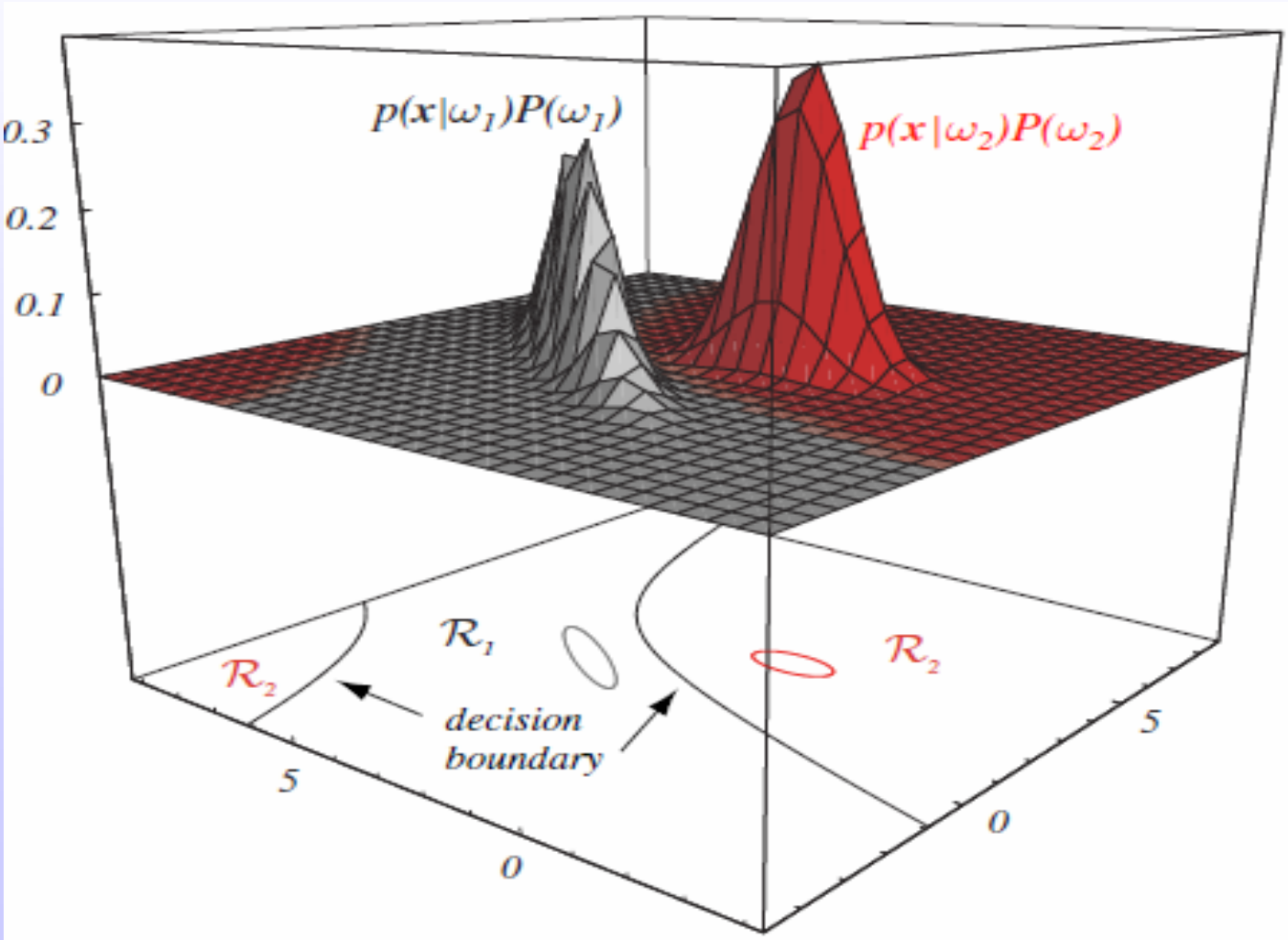
  and find the class ω_i that has the maximum value of the discriminant function for a given pattern $\mathbf{x}$.

- Here, a natural logarithm $\ln$ is applied as it is a monotonically increasing function that does not affect the decision result but will simplify its evaluation if $p(\mathbf{x} \mid \omega_i)$ is an exponential function.

10 Discriminant Functions and Classifiers

- The decision boundary or classification boundary in the space spanned by $\mathbf{x}$ between class ω_i and class ω_j is determined by:

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$



exdjiang@ntu.edu.sg

10 Discriminant Functions and Classifiers

- If the class conditional PDF is multivariate Gaussian of

$p(\mathbf{x}|\omega_i)=N(\mu_i, \Sigma_i)$

$$p(\mathbf{x}\mid\omega_i)=\frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_i\right|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

- The discriminant function becomes:

$$g_i(\mathbf{x})=-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)+\ln p(\omega_i)-\frac{1}{2}\ln\left|\boldsymbol{\Sigma}_i\right|-\frac{d}{2}\ln 2\pi$$

$$=-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)+b_i$$

$$=-\frac{1}{2}d_{\Sigma}(\mathbf{x},\boldsymbol{\mu}_i)+b_i$$

- where $d_{\Sigma_i}(\mathbf{x},\boldsymbol{\mu}_i)=(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i),\qquad b_i=\ln p(\omega_i)-\frac{1}{2}\ln\left|\boldsymbol{\Sigma}_i\right|$

10 Discriminant Functions and Classifiers

$$d_{\Sigma_i}(\mathbf{x},\boldsymbol{\mu}_i) = (\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)$$

is called the Mahalanobis distance between $\mathbf{x}$ and $\boldsymbol{\mu}_i$.

- Compare to the Euclidean distance

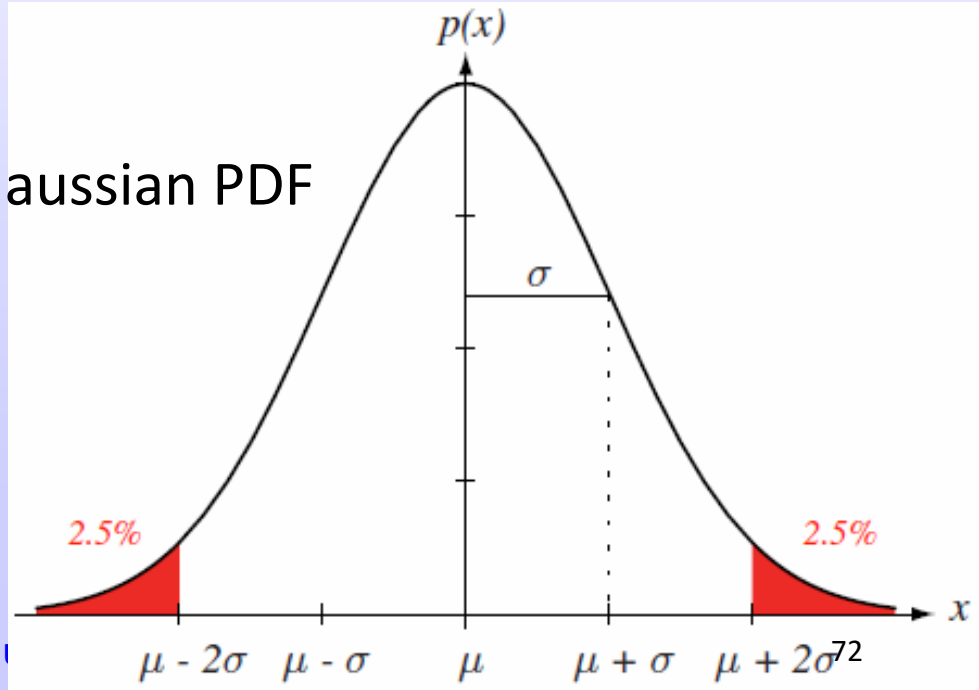
$$d_{Eu}(\mathbf{x},\boldsymbol{\mu}_i) = (\mathbf{x}-\boldsymbol{\mu}_i)^T (\mathbf{x}-\boldsymbol{\mu}_i)$$

- In1D case:

$$d_{Eu}(x,\mu_i) = (x-\mu_i)^2$$

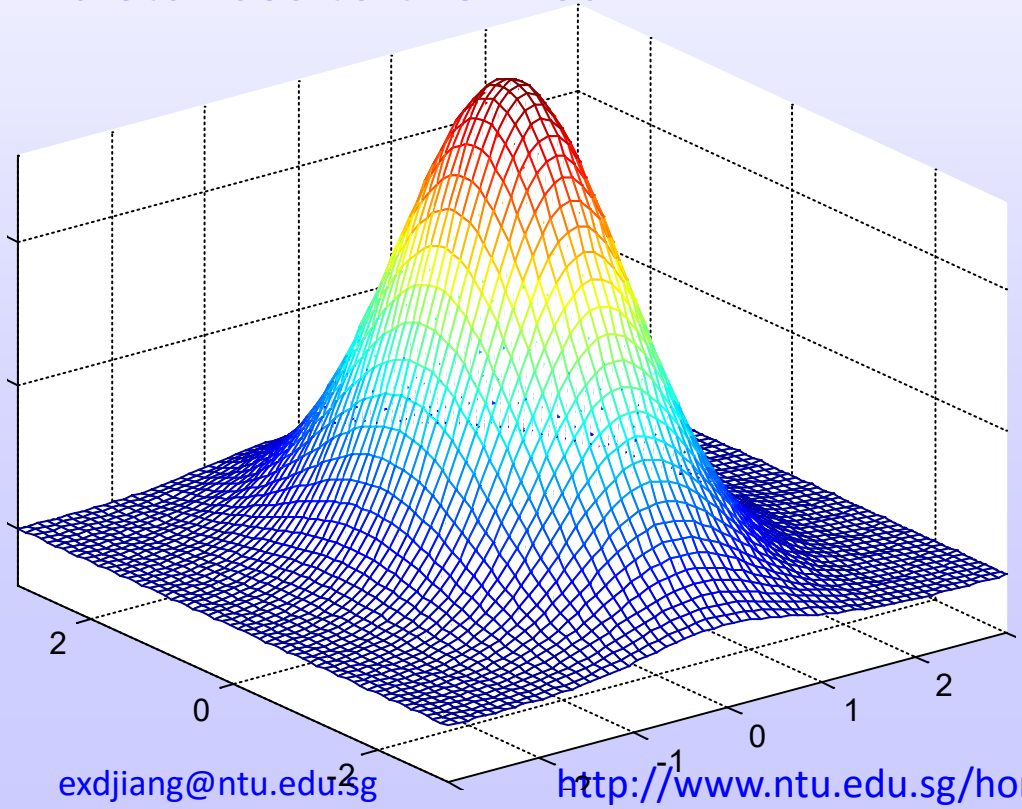
$$d_{\Sigma}(x,\mu_i) = \frac{(x-\mu_i)^2}{\sigma^2}$$

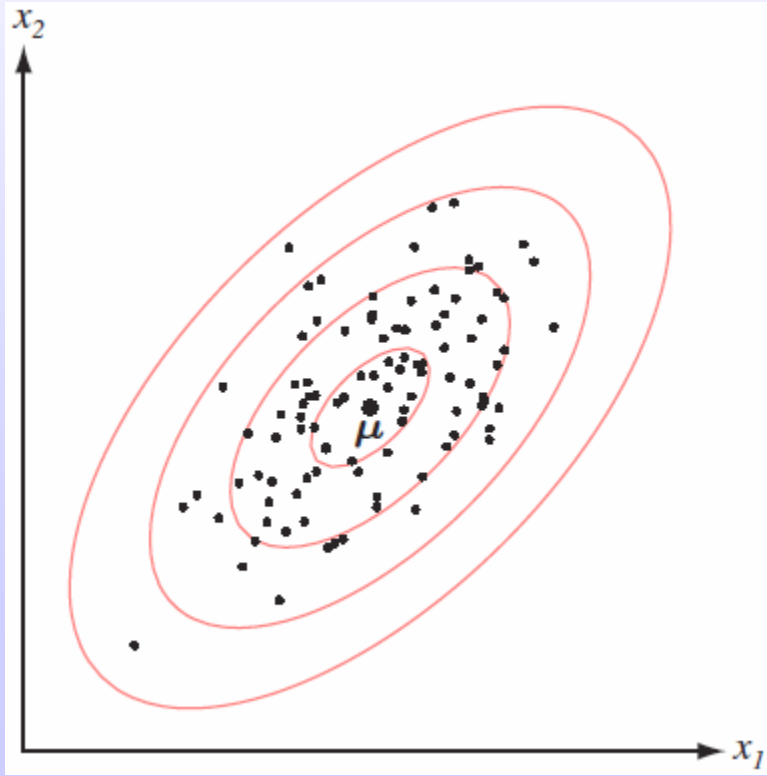
1D Gaussian PDF



10 Discriminant Functions and Classifiers

- For 2D Gaussian PDF Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean. The ellipses show lines of equal probability density of the Gaussian. All points on a ellipse has the same Mahalanobis distance but different Euclidean distances to the mean.





10 Discriminant Functions and Classifiers

- Recall the discriminant functions for class conditional PDF of multivariate Gaussian of $p(\mathbf{x}|\omega_i)=N(\mu_i, \Sigma_i)$ is

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \ln p(\omega_i) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i|$$

- This is the general multivariate normal case where the covariance matrices are different for each category. The only term that can be dropped is the $(d/2)\ln 2\pi$ term, and the resulting discriminant functions are inherently quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^t\mathbf{x} + \omega_{i0}$$

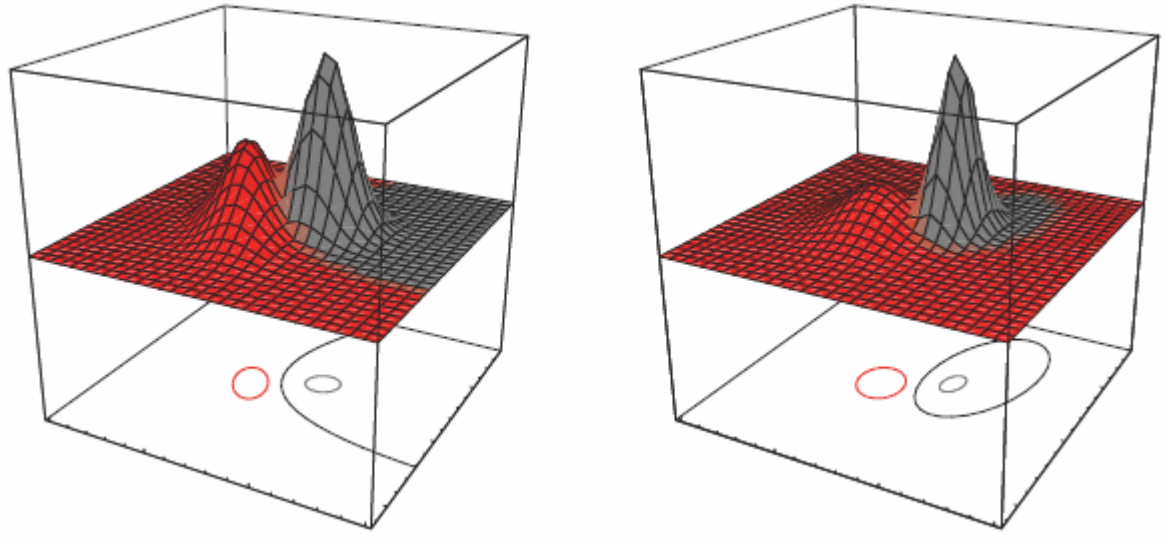
where $\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1}$ and $\mathbf{w}_i = \Sigma_i^{-1}\mu_i$ and

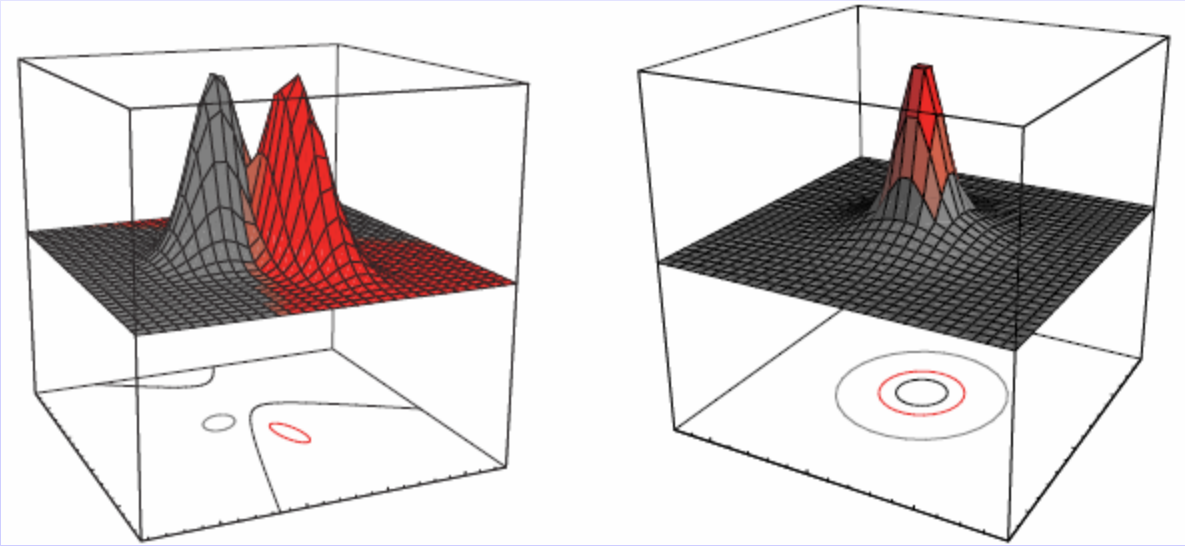
$$\omega_{i0} = -\frac{1}{2}\mu_i^t\Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

- The decision surfaces are *hyperquadrics*, and can assume any of the general forms hyperplanes, pairs of hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, and hyperhyperboloids of various types.

10 Discriminant Functions and Classifiers

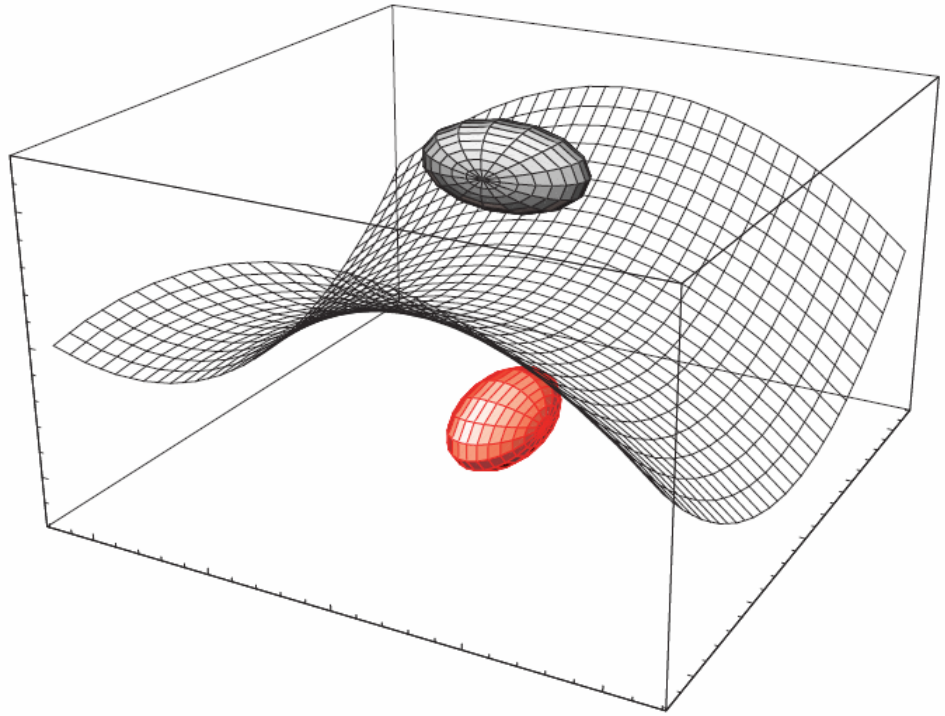
- Decision boundary examples of 2-dimensional, 2-class cases.

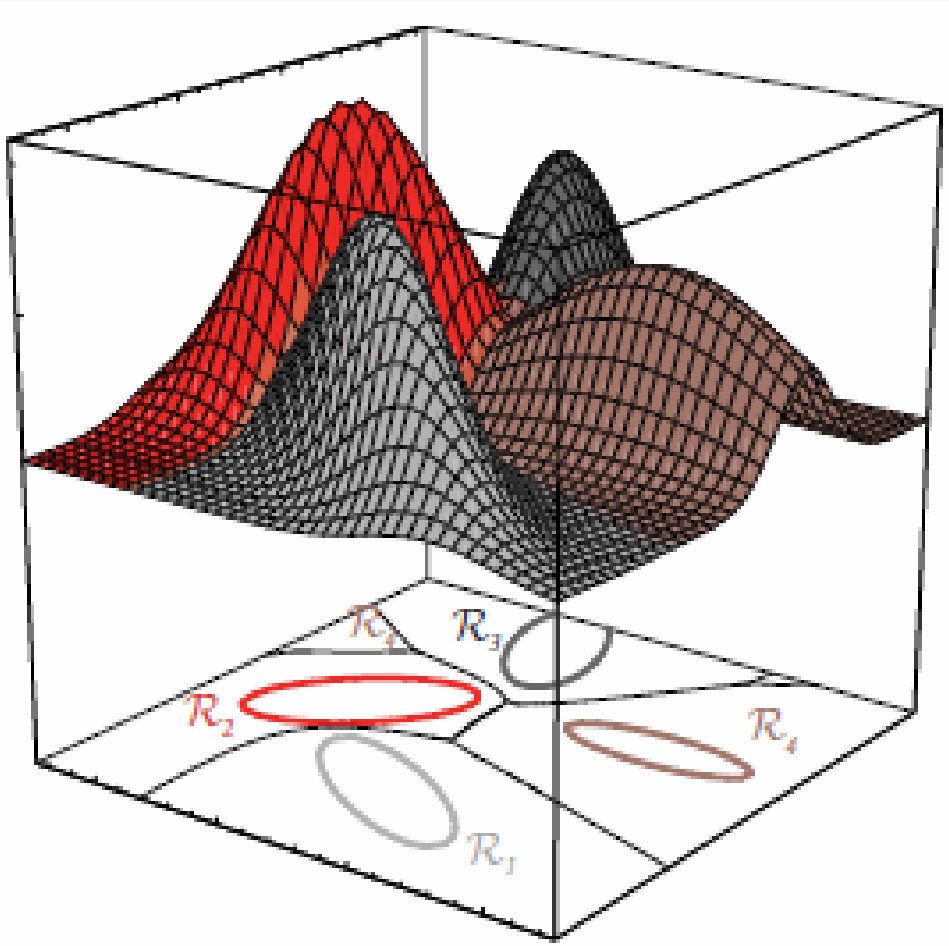




10 Discriminant Functions and Classifiers

- Complex decision boundary for 3-dimensional, 2-class case and 2-dimensional, multiclass problem





10 Discriminant Functions and Classifiers

Example: Decision regions for two-dimensional Gaussian data

- Let ω_1 be the set of the four black points, and ω_2 the red points.

$$\mu_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix} ; \Sigma_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } \mu_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix} ; \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- The inverse matrices are then,

$$\Sigma_1^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \text{ and } \Sigma_2^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

10 Discriminant Functions and Classifiers

Example : Decision regions for two-dimensional Gaussian data (Cont:)

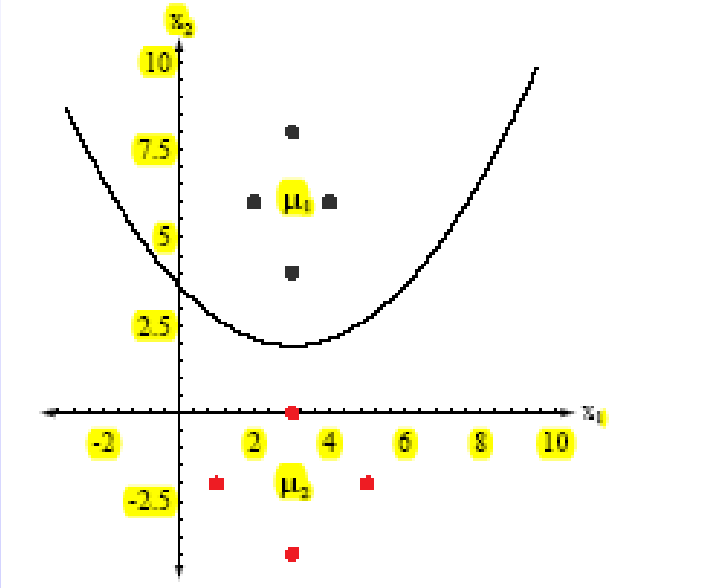
- We assume equal prior probabilities, $P(\omega_1) = P(\omega_2) = 0.5$, and substitute these into the general form for a discriminant, setting $g_1(\mathbf{x}) = g_2(\mathbf{x})$ to obtain the decision boundary:

$$x_2 = 3.514 - 1.125x_1 + 0.1875x_1^2$$

- This equation describes a parabola with vertex at $\begin{pmatrix} 3 \\ 1.83 \end{pmatrix}$ and the decision boundary does not pass through the point $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$

10 Discriminant Functions and Classifiers

Example : Decision regions for two-dimensional Gaussian data (Cont:)



The computed Bayes decision boundary for two Gaussian distributions, each based on four data points.

10 Discriminant Functions and Classifiers

- Special case 1: all classes own the same covariance matrix $\Sigma_i=\Sigma$.
- The discriminant functions are simplified as

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \ln p(\omega_i)$$

- If all classes have the same prior probability

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) = d_\Sigma(\mathbf{x},\boldsymbol{\mu}_i)$$

- The classifier is called the minimum Mahalanobis distance classifier.
- The optimal decision rule is: measure the squared Mahalanobis distance from $\mathbf{x}$ to each of the c mean vectors, and assign $\mathbf{x}$ to the category of the nearest mean. Unequal prior probabilities bias the decision in favor of the a priori more likely category.

10 Discriminant Functions and Classifiers

- Expand the quadratic term

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \ln p(\omega_i)$$

$$= -\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_i{}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln p(\omega_i)$$

- and drop terms that are independent to the class label i we have:

$$g_i(\mathbf{x}) = \boldsymbol{\mu}_i{}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln p(\omega_i)$$

$$= \mathbf{w}_i{}^T\mathbf{x} + w_{i0}$$

- The discriminant function is a linear function of $\mathbf{x}$. This results a linear classifier as we will show that the decision or classification boundary between any two classes is a hyperplane. We call w_{i0} the threshold or bias for the class ω_i.

10 Discriminant Functions and Classifiers

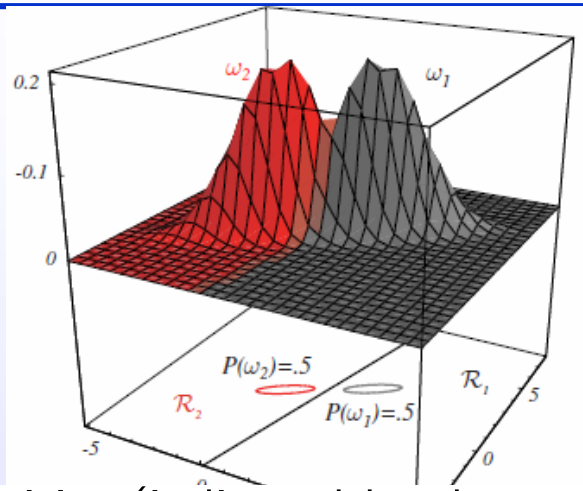
- The decision or classification boundary between any two classes is the solution of the equation.

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \Rightarrow (\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + w_{i0} - w_{j0} = 0$$

$$\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j\right) + \ln[p(\omega_i)/p(\omega_j)] = 0$$

- It is a straight line in 2D space $\mathbf{x}$, a plane in 3D space $\mathbf{x}$ and a hyperplane in higher dimensional space $\mathbf{x}$.
- We see that the decision hyperplane that separates two classes is generally not orthogonal to the line between the means. (in what case is it orthogonal?)
- The prior probabilities shift the optimal boundary hyperplane away from the more likely mean.

10 Discriminant Functions and Classifiers



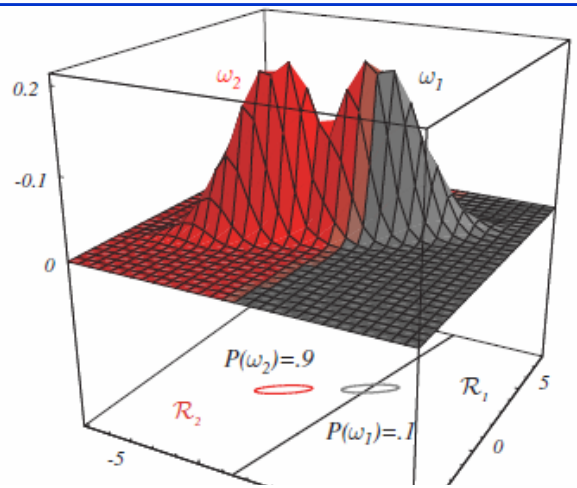
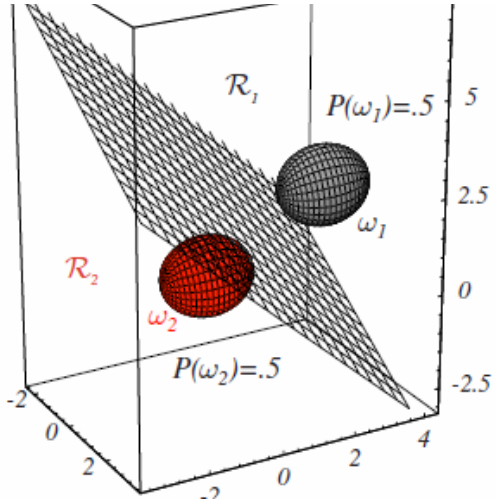
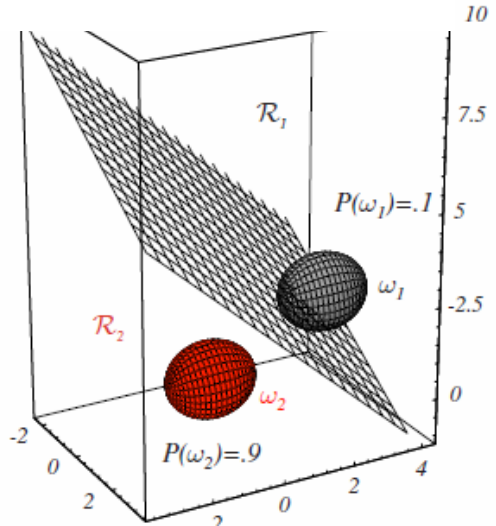


Figure 2.12: Probability densities (indicated by the surfaces in two dimensions and ellipsoidal surfaces in three dimensions) and decision regions for equal but asymmetric Gaussian distributions. The decision hyperplanes need not be perpendicular to the line connecting the means.





exdjiang@ntu.edu.sg http:/

10 Discriminant Functions and Classifiers

Example:

Given pattern vectors $(1,2)^t, (2,2)^t, (3,1)^t, (3,2)^t$, and $(2,3)^t$, are known to be in class ω_1. Another set of vectors, $(7,9)^t, (8,9)^t, (9,8)^t, (9,9)^t$, and $(8,10)^t$, are known to be in class ω_2. Find the Bayes classifier and the decision boundary. (assume all classes are equally likely to occur.)

Solution:

$c = 2, j = 1, 2.$

$$\mu_1 = \frac{1}{5}\left[\begin{pmatrix}1\\2\end{pmatrix} + \begin{pmatrix}2\\2\end{pmatrix} + \begin{pmatrix}3\\1\end{pmatrix} + \begin{pmatrix}3\\2\end{pmatrix} + \begin{pmatrix}2\\3\end{pmatrix}\right] = \frac{1}{5}\begin{bmatrix}11\\10\end{bmatrix}$$

$$\mu_2 = \frac{1}{5}\left[\begin{pmatrix}7\\9\end{pmatrix} + \begin{pmatrix}8\\9\end{pmatrix} + \begin{pmatrix}9\\8\end{pmatrix} + \begin{pmatrix}9\\9\end{pmatrix} + \begin{pmatrix}8\\10\end{pmatrix}\right] = \frac{1}{5}\begin{bmatrix}41\\45\end{bmatrix}$$

10 Discriminant Functions and Classifiers

Solution: (cont)

$$\Sigma_j = \left(\frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x}\mathbf{x}^t \right) - \mu_j \mu_j^t$$

$$\Sigma_1 = \frac{1}{5}\left[\begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix} \begin{pmatrix} 2 & 2 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \end{pmatrix} \begin{pmatrix} 3 & 1 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} \begin{pmatrix} 3 & 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} \begin{pmatrix} 2 & 3 \end{pmatrix} \right]$$

$$-\frac{1}{25} \begin{pmatrix} 11 \\ 10 \end{pmatrix} \begin{pmatrix} 11 & 10 \end{pmatrix}$$

$$\Sigma_1 = \frac{1}{25} \begin{pmatrix} 14 & -5 \\ -5 & 10 \end{pmatrix}$$

Similarly, we find $\Sigma_2 = \Sigma_1 = \Sigma$

$$\Sigma^{-1} = \frac{5}{23} \begin{pmatrix} 10 & 5 \\ 5 & 14 \end{pmatrix}$$

10 Discriminant Functions and Classifiers

Solution: (cont)

All classes are equally likely to occur means $P(\omega_1) = P(\omega_2)$. Also, $\Sigma_1 = \Sigma_2 = \Sigma$, the decision function is simplified to

$$d_j(\mathbf{x}) = \mathbf{x}^t \Sigma^{-1} \mu_j - \frac{1}{2} \mu_j^t \Sigma^{-1} \mu_j$$

$\Sigma^{-1}\mu_1 = \frac{1}{23}\begin{pmatrix} 160 \\ 195 \end{pmatrix}$ and $\mu_1^t \Sigma^{-1} \mu_1 = \frac{742}{23}$ $\quad g_i(\mathbf{x}) = -(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$

$$d_1(\mathbf{x}) = \frac{160}{23} x_1 + \frac{195}{23} x_2 - 16.13$$

Similarly, $\Sigma^{-1}\mu_2 = \frac{1}{23}\begin{pmatrix} 635 \\ 835 \end{pmatrix}$ and $\mu_2^t \Sigma^{-1} \mu_2 = 553.12$

$$d_2(\mathbf{x}) = \frac{635}{23} x_1 + \frac{835}{23} x_2 - 276.56$$

10 Discriminant Functions and Classifiers

Solution: (cont) The decision matrix is

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} \frac{160}{23} & \frac{195}{23} & -16.13 \\ \frac{635}{23} & \frac{835}{23} & -276.56 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

- To find the decision boundary, $d_{12}(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x})$
- The decision rule is : decide ω_1 if $d_{12} > 0$, otherwise, decide ω_2.

10 Discriminant Functions and Classifiers

- Special case 2: all classes own the same diagonal, scalar covariance matrix $\Sigma_i=\sigma^2\mathbf{I}$. $\mathbf{I}$ is the identity matrix.
- This case occurs when all features (components of $\mathbf{x}$) are statistically independent and each feature has the same variance, σ^2.
- Note that $\boldsymbol{\Sigma}_i = \sigma^2\mathbf{I} \Rightarrow \boldsymbol{\Sigma}_i^{-1} = \mathbf{I}/\sigma^2, \quad |\boldsymbol{\Sigma}_i| = \sigma^{2d}$
- The discriminant functions are simplified as

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)+\ln p(\omega_i)$$

$$= -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_i)^T(\mathbf{x}-\boldsymbol{\mu}_i)+\ln p(\omega_i)$$

$$= -\frac{1}{2\sigma^2}(\mathbf{x}^T\mathbf{x}-2\boldsymbol{\mu}_i{}^T\mathbf{x}+\boldsymbol{\mu}_i{}^T\boldsymbol{\mu}_i)+\ln p(\omega_i)$$

10 Discriminant Functions and Classifiers

- The quadratic term $\mathbf{x}^T\mathbf{x}$ is the same for all class i and hence can be dropped. Multiplying the constant σ^2, the discriminant functions are simplified as

$$g_i(\mathbf{x}) = \boldsymbol{\mu}_i^T\mathbf{x} - \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i / 2 + \sigma^2 \ln p(\omega_i)$$

$$= \mathbf{w}_i^T\mathbf{x} + w_{i0}$$

- We call w_{i0} the threshold or bias for the class ω_i. it is a linear function of $\mathbf{x}$. This results a linear classifier.
- The decision or classification boundary between any two classes is the solution of the equation.

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \Rightarrow (\mathbf{w}_i - \mathbf{w}_j)^T\mathbf{x} + w_{i0} - w_{j0} = 0$$

$$(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\mathbf{x} - (\boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j)/2 + \sigma^2 \ln[p(\omega_i)/p(\omega_j)] = 0$$

- This equation defines a hyperplane orthogonal to the line linking the two means.

10 Discriminant Functions and Classifiers

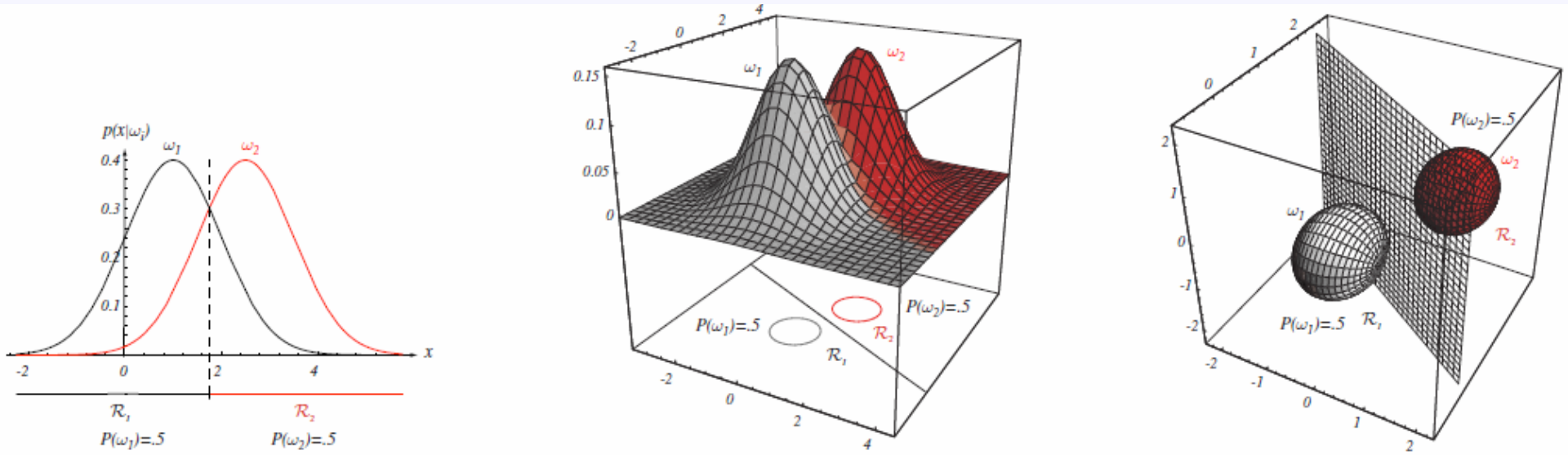
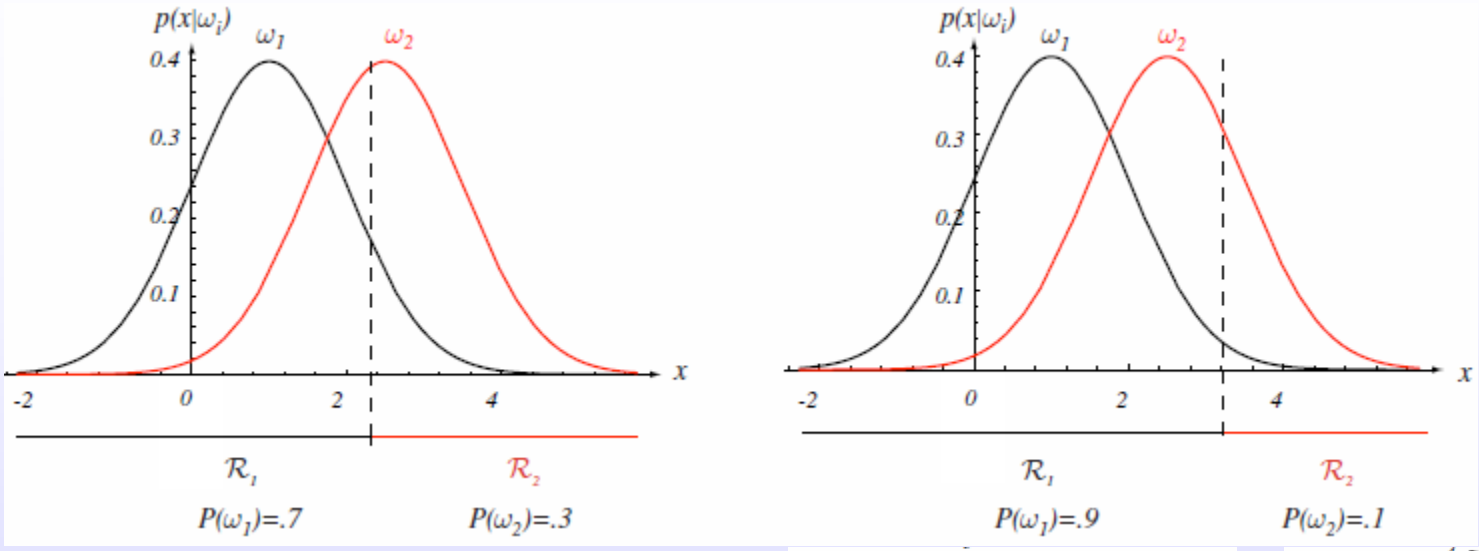
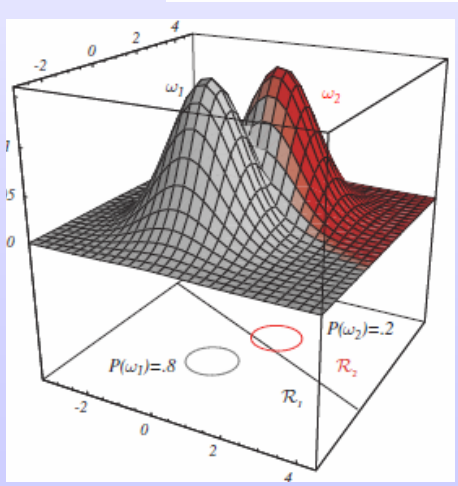
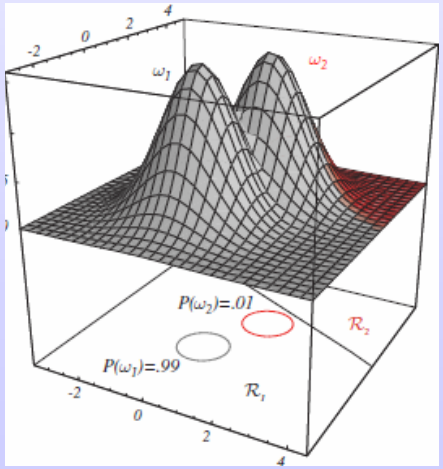


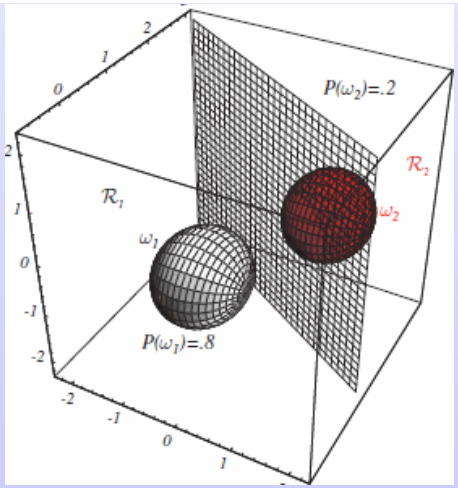
FIGURE 2.10. If the covariance matrices for two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these one-, two-, and three-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the three-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

10 Discriminant Functions and Classifiers









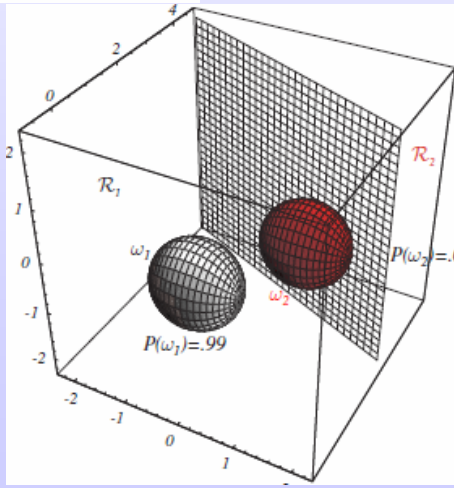


FIGURE 2.11. As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these one-, two- and three-dimensional spherical Gaussian distributions. From: Richard O. Duda, Peter E.

10 Discriminant Functions and Classifiers

- Recall the discriminant function

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_i)^T(\mathbf{x}-\boldsymbol{\mu}_i) + \ln p(\omega_i)$$

- If the prior probabilities are the same for all classes, then

$$g_i(\mathbf{x}) = -(\mathbf{x}-\boldsymbol{\mu}_i)^T(\mathbf{x}-\boldsymbol{\mu}_i) = -d_{Eu}(\mathbf{x},\boldsymbol{\mu}_i) = \| \mathbf{x}-\boldsymbol{\mu}_i \|$$

- When this happens, the optimum decision rule can be stated very simply: to classify a feature vector $\mathbf{x}$, measure the Euclidean distance from $\mathbf{x}$ to each of the c mean vectors, and assign $\mathbf{x}$ to the category of the nearest mean. Such a classifier is called a minimum distance classifier. If each mean vector is thought of as being an ideal prototype or template for patterns in its class, then this is essentially a template matching procedure.

10 Discriminant Functions and Classifiers

Example:

$\omega_1 : \mathbf{m}_1=(x_1^{(1)},x_2^{(1)})^t$

$\omega_2 : \mathbf{m}_2=(x_1^{(2)},x_2^{(2)})^t$

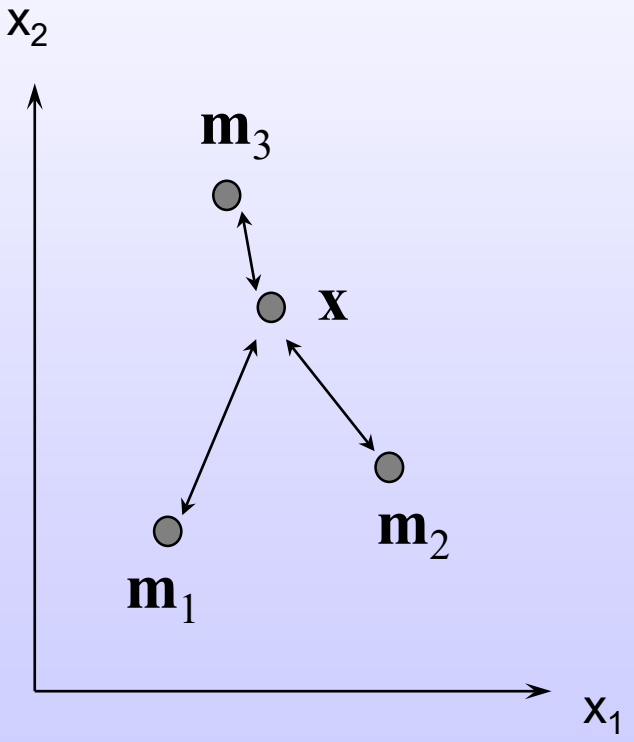
$\omega_3 : \mathbf{m}_3=(x_1^{(3)},x_2^{(3)})^t$

$\|\mathbf{x}- \mathbf{m}_3\| < \|\mathbf{x}- \mathbf{m}_1\|$

$\|\mathbf{x}- \mathbf{m}_3\| < \|\mathbf{x}- \mathbf{m}_2\|$

$\mathbf{x}$ is closest to $\mathbf{m}_3$

Therefore, $\mathbf{x}$ belongs to ω_3.



10 Discriminant Functions and Classifiers

- Example : Classify Iris versicolor and Iris setosa by petal length, x_1 and petal width, x_2.

$$d_1(\mathbf{x}) = \mathbf{x}^t\mathbf{m}_1 - \frac{1}{2}\mathbf{m}_1^t\mathbf{m}_1 = 4.3x_1 + 1.3x_2 - 10.1$$

$$d_2(\mathbf{x}) = \mathbf{x}^t\mathbf{m}_2 - \frac{1}{2}\mathbf{m}_2^t\mathbf{m}_2 = 1.5x_1 + 0.3x_2 - 1.17$$

Given:

Versicolor ω_1:

$\mathbf{m}_1 = (4.3, 1.3)^t$

Setosa ω_2 :

$\mathbf{m}_2 = (1.5, 0.3)^t$

Rewrite in matrix form,

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

If a petal has length 2 cm and width 0.5cm, i.e. $\mathbf{x} = (2, 0.5)^t$

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} 2 \\ 0.5 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.85 \\ 1.98 \end{pmatrix}$$

Therefore, the petal is from Iris setosa.

If a petal has length 5 cm and width 1.5cm, i.e. $\mathbf{x} = (5, 1.5)^t$

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 4.3 & 1.3 & -10.1 \\ 1.5 & 0.3 & -1.17 \end{pmatrix} \begin{pmatrix} 5 \\ 1.5 \\ 1 \end{pmatrix} = \begin{pmatrix} 13.35 \\ 6.78 \end{pmatrix}$$

Therefore, the petal is from Iris versicolor.

10 Discriminant Functions and Classifiers

- The decision boundary is the line that separates two classes.
- In the minimum distance classifier, it is a perpendicular bisector of the line joining two class prototypes.
- At the decision boundary (surface), $d_1(\mathbf{x}) = d_2(\mathbf{x})$. Hence, for Iris classification

$$d_{12}(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 2.8x_1 + 1.0x_2 - 8.9 = 0$$

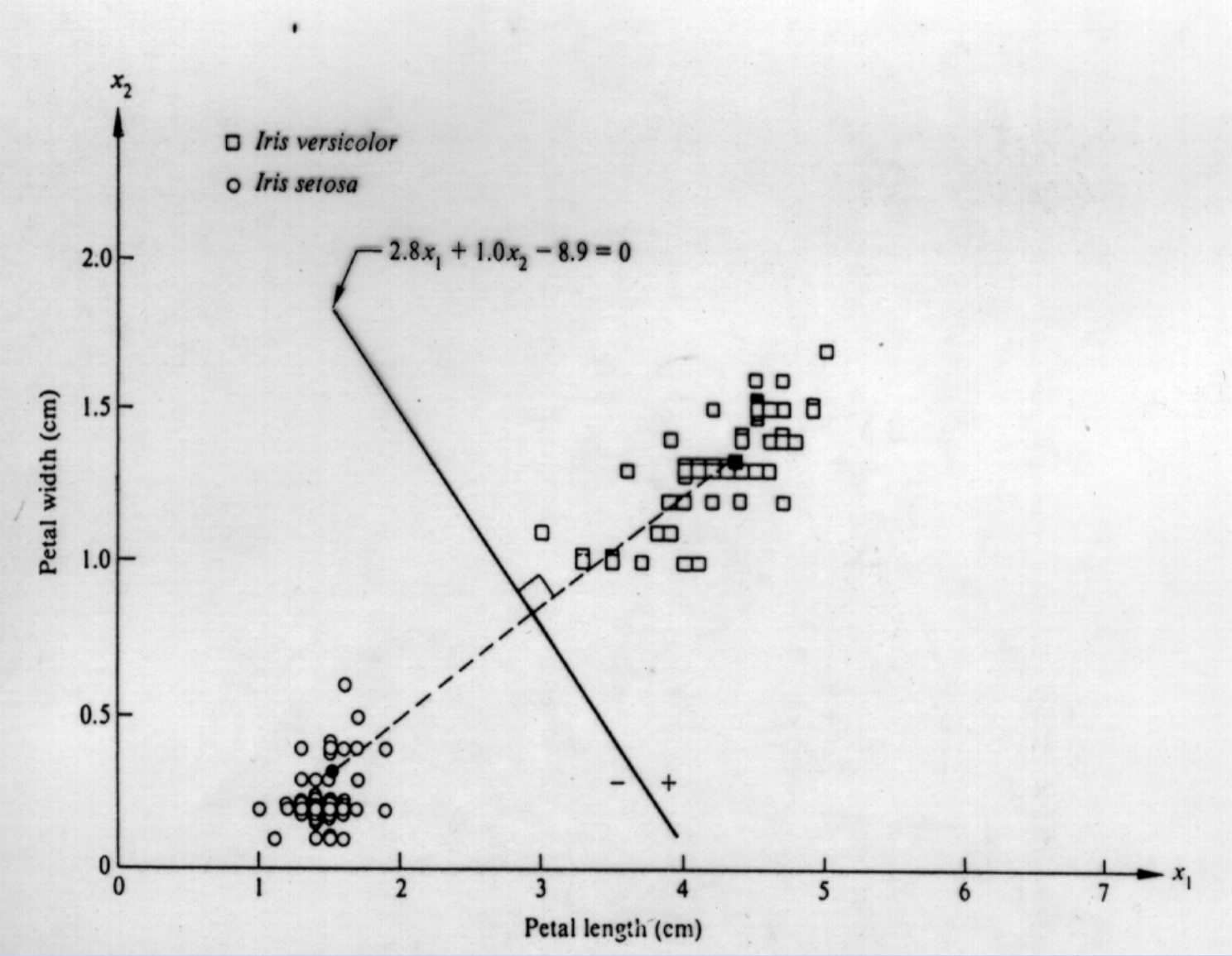
The decision rule: $\mathbf{x} \rightarrow \omega_1$ if $d_{12}(\mathbf{x}) > 0$ or, $\mathbf{x} \rightarrow \omega_2$ if $d_{12}(\mathbf{x}) < 0$.

Example:

$$\mathbf{x} = (2, 0.5)^t; d_{12}((2, 0.5)^t) = -2.8 \rightarrow \omega_2 : \text{ Iris setosa}$$

$$\mathbf{x} = (5, 1.5)^t; d_{12}((5, 1.5)^t) = 6.6 \rightarrow \omega_1 : \text{ Iris versicolor}$$

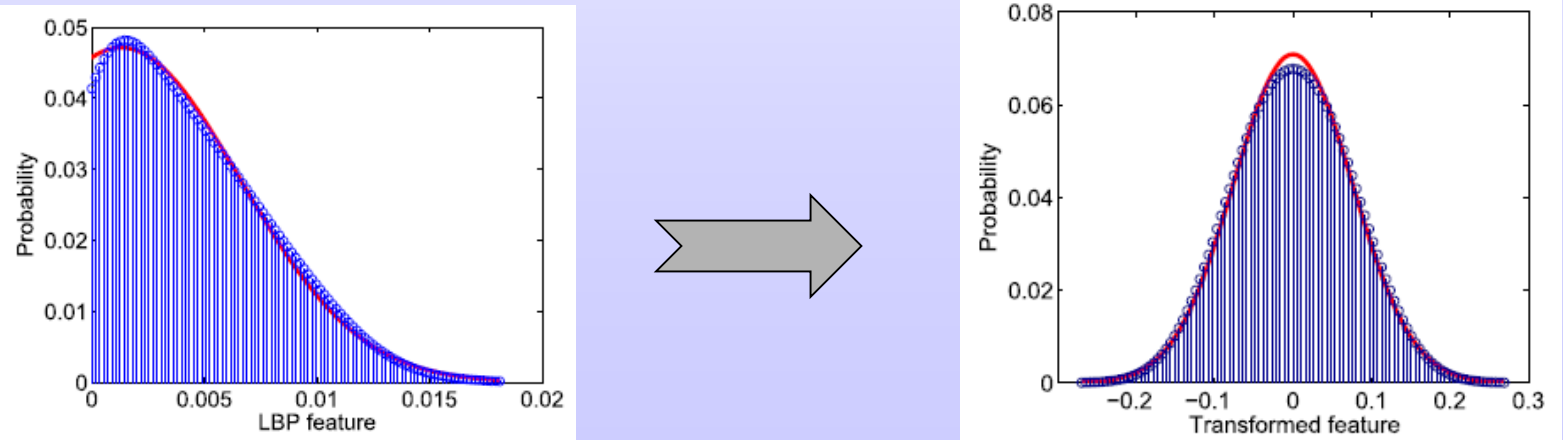
10 Discriminant Functions and Classifiers



10 Discriminant Functions and Classifiers

- We have derived several classifiers under Gaussian assumption.
- Gaussian PDF is the most natural and most common distribution.
- For Non-Gaussian distribution, it is very difficult if not impossible to derive a theoretical optimal classifier.
- One way to solve non-Gaussian data distribution is to perform some proper feature transform to convert it into Gaussian PDF. Example:

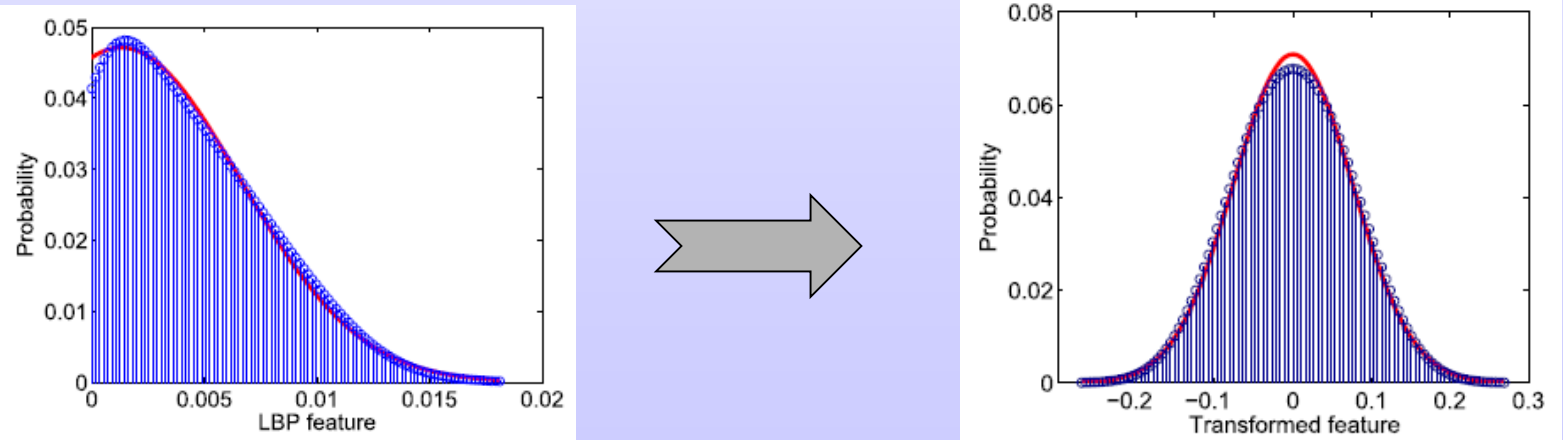
J. Ren, X.D. Jiang and J. Yuan, "A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.



10 Discriminant Functions and Classifiers

- We have derived several classifiers under Gaussian assumption.
- Gaussian PDF is the most natural and most common distribution.
- For Non-Gaussian distribution, it is very difficult if not impossible to derive a theoretical optimal classifier.
- One way to solve non-Gaussian data distribution is to perform some proper feature transform to convert it into Gaussian PDF. Example:

J. Ren, X.D. Jiang and J. Yuan, "A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.



Classifiers by Sparse Representation

Query image **y** can be well represented by a linear combination of its training images

$\mathbf{A}_i = [\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,j}, \ldots, \mathbf{a}_{i,ni}]$: training images of *i*-th class:

$$\mathbf{y} = \mathbf{A}_i\mathbf{x}_i + \mathbf{e}_i$$

$\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_i, \ldots, \mathbf{A}_c]$: training samples of all c subjects

$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$, here $\mathbf{x}=[\mathbf{x}_1;\ldots; \mathbf{x}_i;\ldots; \mathbf{x}_c]$

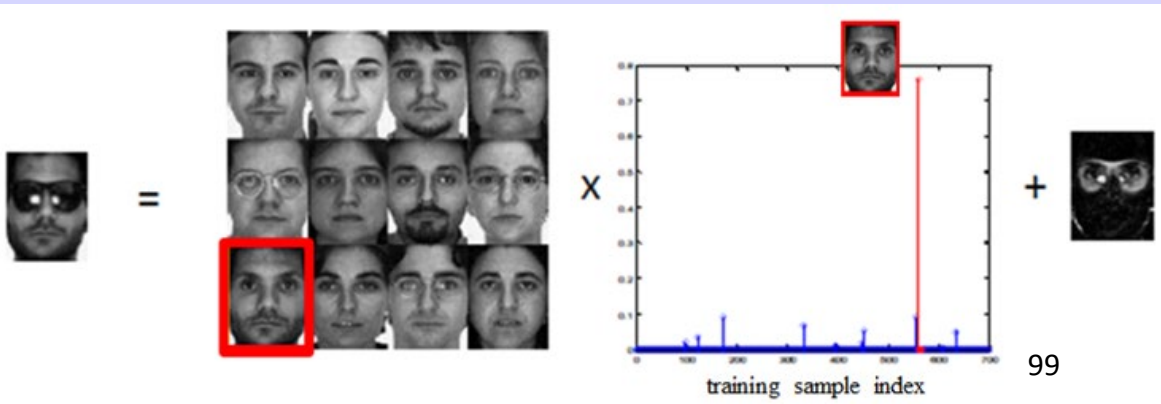
Problems!

If $\mathbf{x} = [\mathbf{0};\ldots; \mathbf{0}; \mathbf{x}_i; \mathbf{0}; \ldots; \mathbf{0}]$, we can identify the class i of query image **y**

Given **y** and **A**:

find sparse coefficients **x**

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$$



Sparse Coding or Sparse Representation

y can be well represented by a linear combination of its training images

y contains **all pixels** of the image.

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

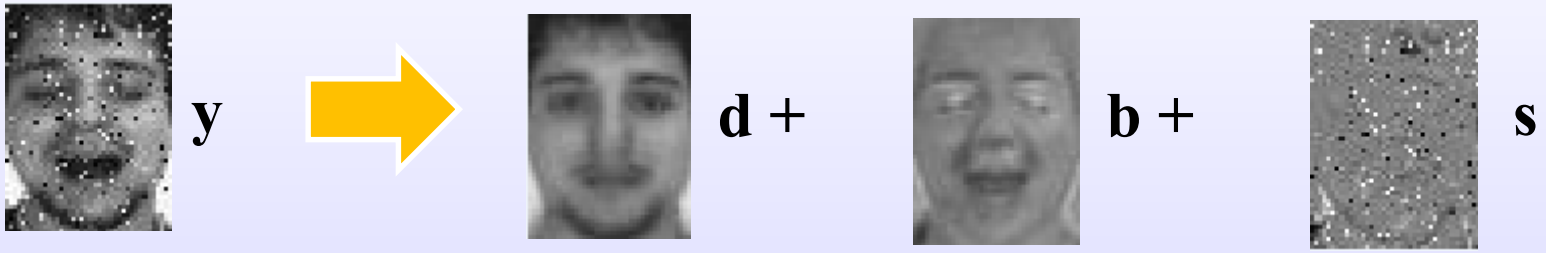
Problems!

J. Lai and X.D. Jiang, "Modular Weighted Global Sparse Representation for Robust Face Recognition," *IEEE Signal Processing letters*, vol. 19, no. 9, pp. 571-574, Sep. 2012.

Image Information Decomposition

y can be well represented by a linear combination of its training images.
What is **its training images**?

Problems!



$$\mathbf{y} = \mathbf{d} + \mathbf{b} + \mathbf{s}$$

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

d : the class-specific/identity component

b : the non-class-specific/intra-class variation component

s : the sparse noise or corruption

X.D. Jiang and J. Lai, "Sparse And Dense Hybrid Representation via Dictionary Decomposition for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 1067-1079, May, 2015.

Information Decomposition of Image Dictionary

y can be well represented by a linear combination of its training images.
What is **its training images**?

The proposed supervised low-rank decomposition (SLR):

$$\min_{\mathbf{D},\mathbf{B},\mathbf{\Gamma},\mathbf{E}} \|\mathbf{D}\|_* + \lambda\|\mathbf{B}\|_* + \tau\|\mathbf{\Gamma}\|_F^2 + \eta\|\mathbf{E}\|_1$$

$$\text{s.t. } \mathbf{A} = \mathbf{D} + \mathbf{B}\mathbf{\Gamma} + \mathbf{E}.$$

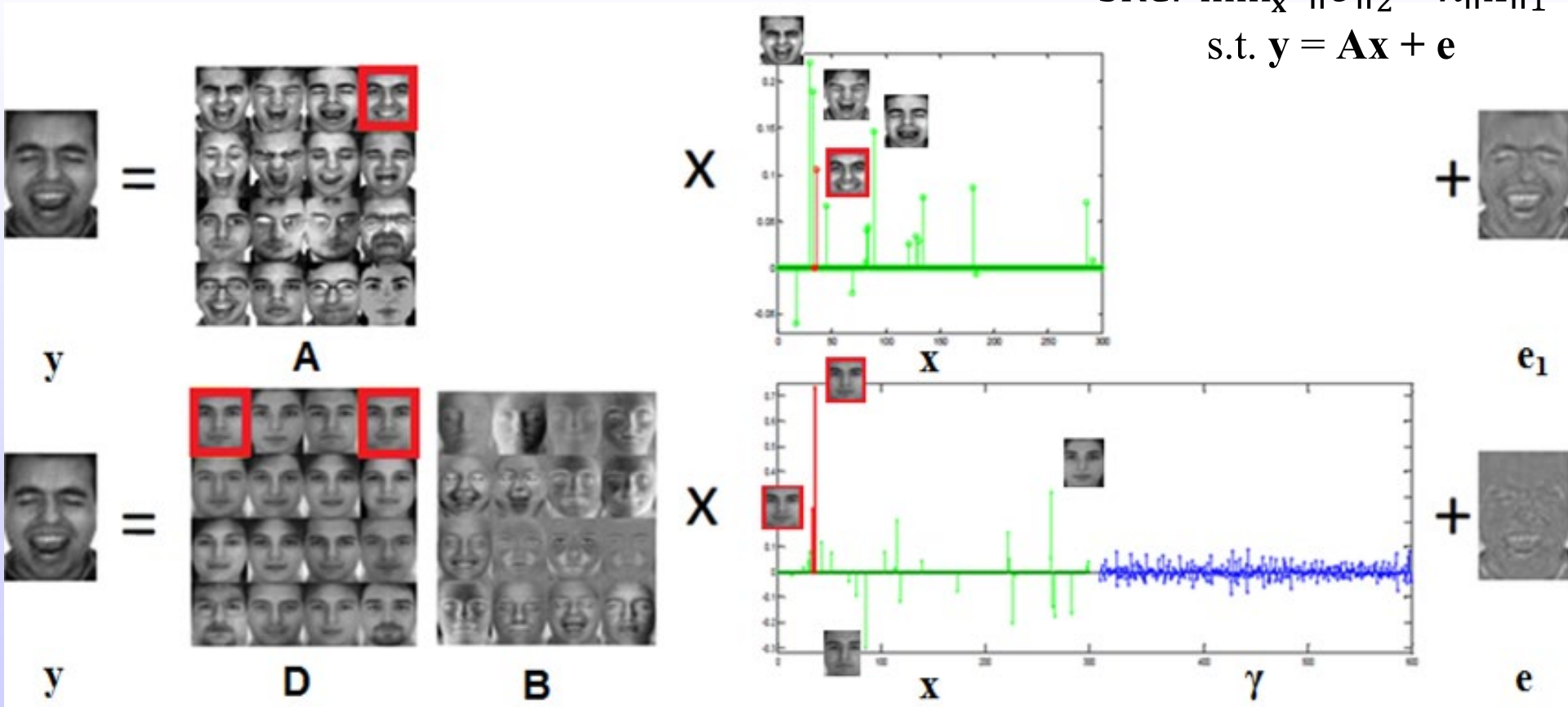
A = D + BΓ + E

X.D. Jiang and J. Lai, "Sparse And Dense Hybrid Representation via Dictionary Decomposition for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 1067-1079, May, 2015.

Sparse- and Dense- Hybrid Representation

y can be well represented by its training images. What is **its training images**?

SRC: $\min_{\mathbf{x}} \|\boldsymbol{e}\|_2^2 + \lambda\|\mathbf{x}\|_1$
s.t. $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$



$$\min_{\mathbf{x},\boldsymbol{\gamma},\mathbf{e}} \|\mathbf{x}\|_1 + \sigma\|\boldsymbol{\gamma}\|_2^2 + \beta\|\mathbf{e}\|_1$$

$$\text{s.t. } \mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{B}\boldsymbol{\gamma} + \mathbf{e}.$$

X.D. Jiang and J. Lai, "Sparse And Dense Hybrid Representation via Dictionary Decomposition for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 1067-1079, May, 2015.

11 Unsupervised Learning and Clustering

Outline

- Supervised and unsupervised learning
- Clustering as an approach to unsupervised learning
- K-means clustering algorithm
- Hierarchical agglomerative clustering algorithm
- Gaussian Mixture model and expectation-maximization (EM) algorithm

11 Unsupervised Learning and Clustering

- Previously, all our training samples were labeled: these samples were said "supervised".
- We now investigate a number of "unsupervised" procedures which use unlabeled samples because collecting and labeling a large set of sample patterns can be costly or in some cases not possible based on human knowledge.

- We can use unsupervised methods to identify features that will then be useful for categorization.
- We gain some insight into the nature (or structure) of the data.

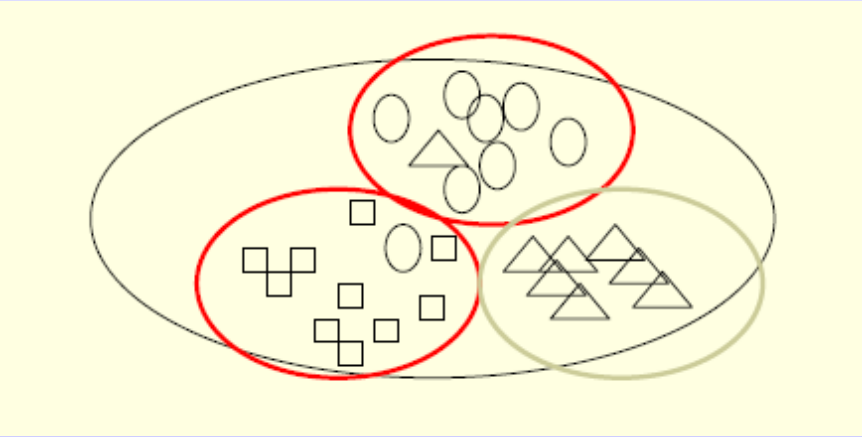
- Clustering that partitions samples into a number of groups without knowing the class membership of the samples belong to unsupervised learning.

11 Unsupervised Learning and Clustering

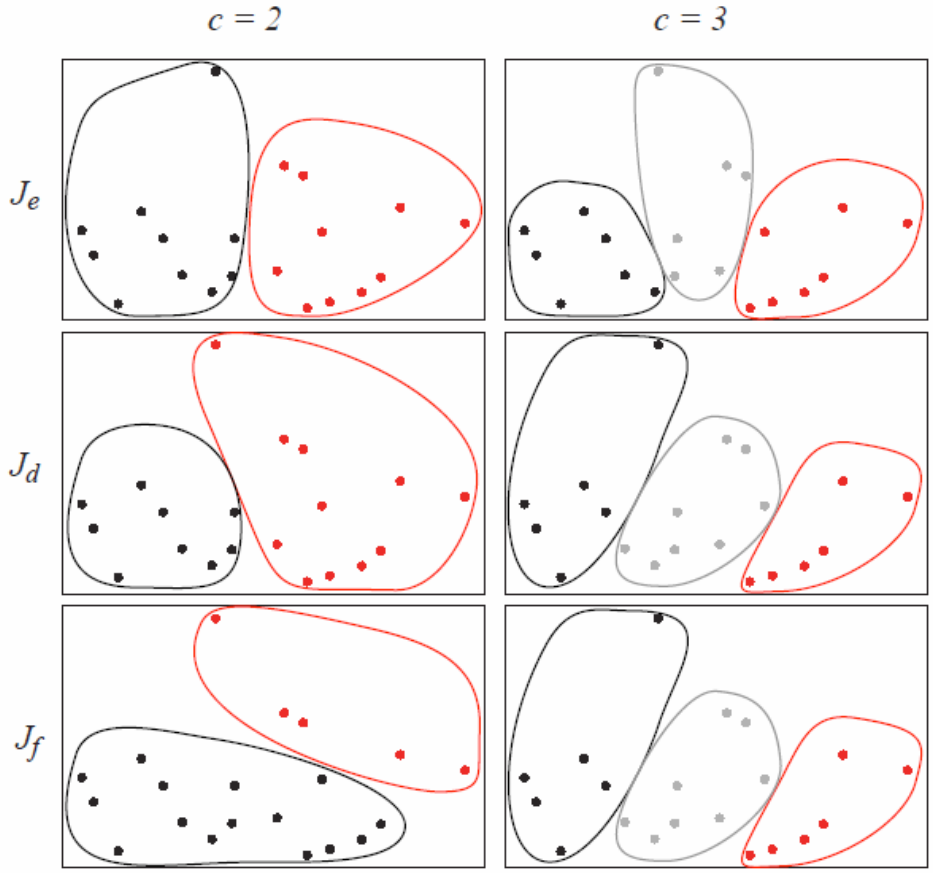
- Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called clusters. – Helps users understand the natural grouping or structure in a data set.
- Cluster: a collection of data objects that are "similar" to one another and thus can be treated collectively as one group.
- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of *n* samples and an integer value *k*, the *Clustering Problem* is to define a mapping f: D → {1,..,k} where each t_i is assigned to one cluster K_j, 1<j<k.
- A *Cluster*, K_j, contains those samples mapped to it. Unlike classification problem, clusters are not known a priori.
  - To group data into un-predefined classes
  - To make data within the same cluster have high similarity, while data points in different clusters have low similarity.

11 Unsupervised Learning and Clustering

- A good method will produce high quality clusters in which:
  - the intra-cluster similarity is high.
  - the inter-cluster similarity is low.
- The quality of a clustering result also depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover the hidden patterns.



11 Unsupervised Learning and Clustering



The clusters found by minimizing a criterion depends upon the criterion function as well as the assumed number of clusters. The sum-of-squared-error criterion J_e (Eq. 49), the determinant criterion J_d (Eq. 63) and the more subtle trace criterion J_f (Eq. 65) were applied to the 20 points in the table with the assumption of $c = 2$ and $c = 3$ clusters. (Each point in the table is shown, with bounding boxes defined

11 Unsupervised Learning and Clustering

Major Categories of Algorithms:

- Partitioning : Construct a partition of a database *D* of *n* objects into a set of *k* clusters that optimizes the chosen partitioning criterion.
  - K-means algorithm
- Hierarchy: Construct a hierarchical partition of data using some criterion
  - Agglomerative
  - Divisive
- Model-based : A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.
  - Gaussian Mixture model

11 Unsupervised Learning and Clustering

Partitioning method: Construct a partition of a database $\mathfrak{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ of n objects into a set of k clusters $\mathfrak{D}_j, j = 1, 2, \cdots, k$ that optimizes the chosen partitioning criterion.

k-means algorithm--principle:

- Each cluster is represented by the center (mean) of the cluster.

$$\mu_j = \frac{1}{|\mathfrak{D}_j|} \sum_{\mathbf{x}_i \in \mathfrak{D}_j} \mathbf{x}_i$$

- Estimate the unknown k cluster centers (means)

$$\mathfrak{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\}$$

to minimize the sum of error squares.

$$e(\mathfrak{M}) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathfrak{D}_j} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

11 Unsupervised Learning and Clustering

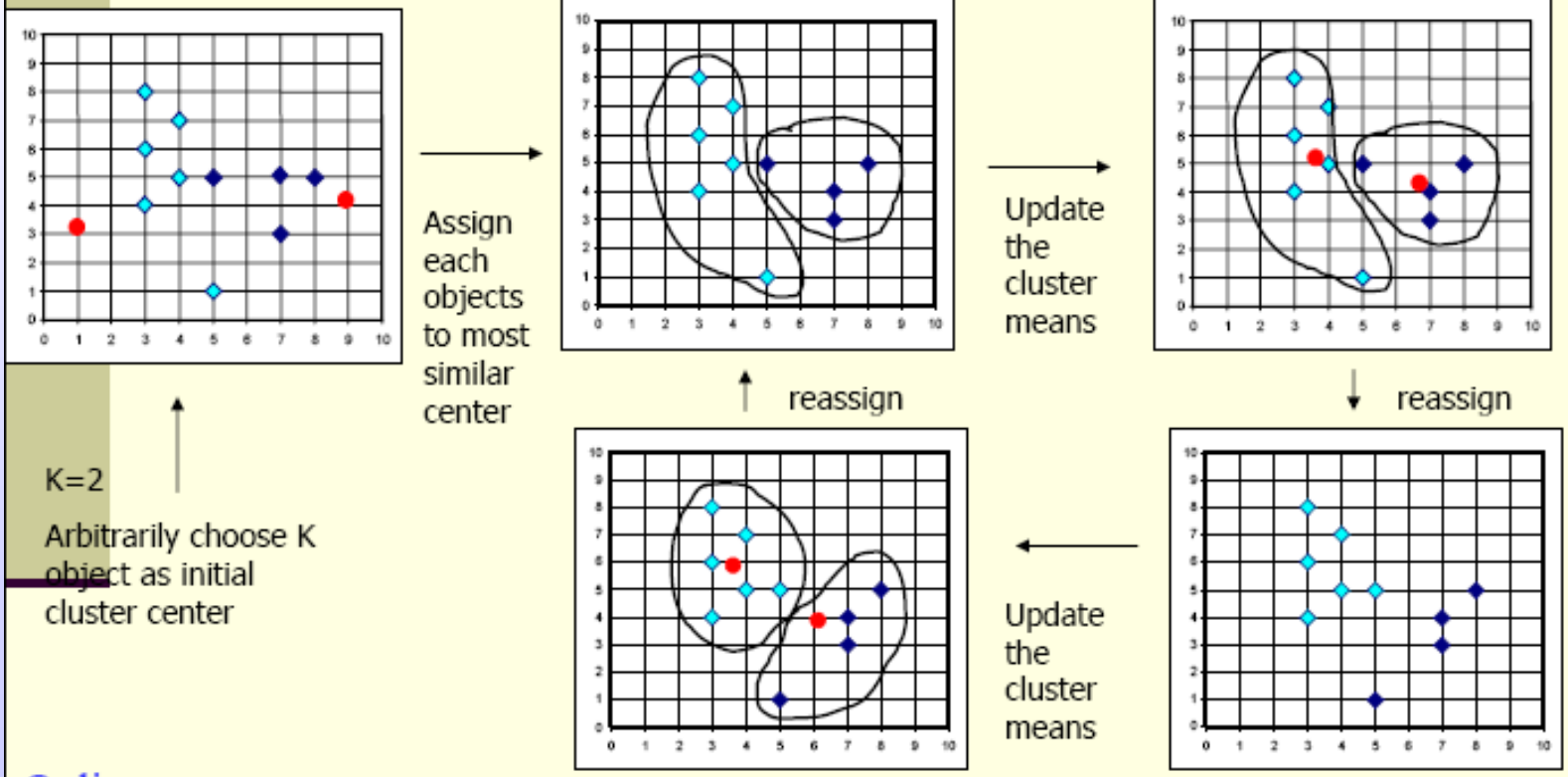
k-means algorithm--implementation:

1. Initialize k clusters, e.g. randomly pick samples.
   $$\mathfrak{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\} = \{\mathbf{x}_{r1}, \mathbf{x}_{r2}, \cdots, \mathbf{x}_{rk}\}$$
2. Classify n samples into k clusters according to nearest to $\boldsymbol{\mu}_j$.
3. Re-compute all cluster centers (means).
   $$\boldsymbol{\mu}_j = \frac{1}{|\mathfrak{D}_j|} \sum_{\mathbf{x}_i \in \mathfrak{D}_j} \mathbf{x}_i, \quad j = 1, 2, \cdots, k$$
4. Go back step 2 until no change in $\mathfrak{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\}$.

- The computational complexity of the algorithm is $O(ndkT)$ where d the number of features and T the number of iterations. In practice, the number of iterations is generally much less than the number of samples.

11 Unsupervised Learning and Clustering

Example



Online:
http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html

11 Unsupervised Learning and Clustering

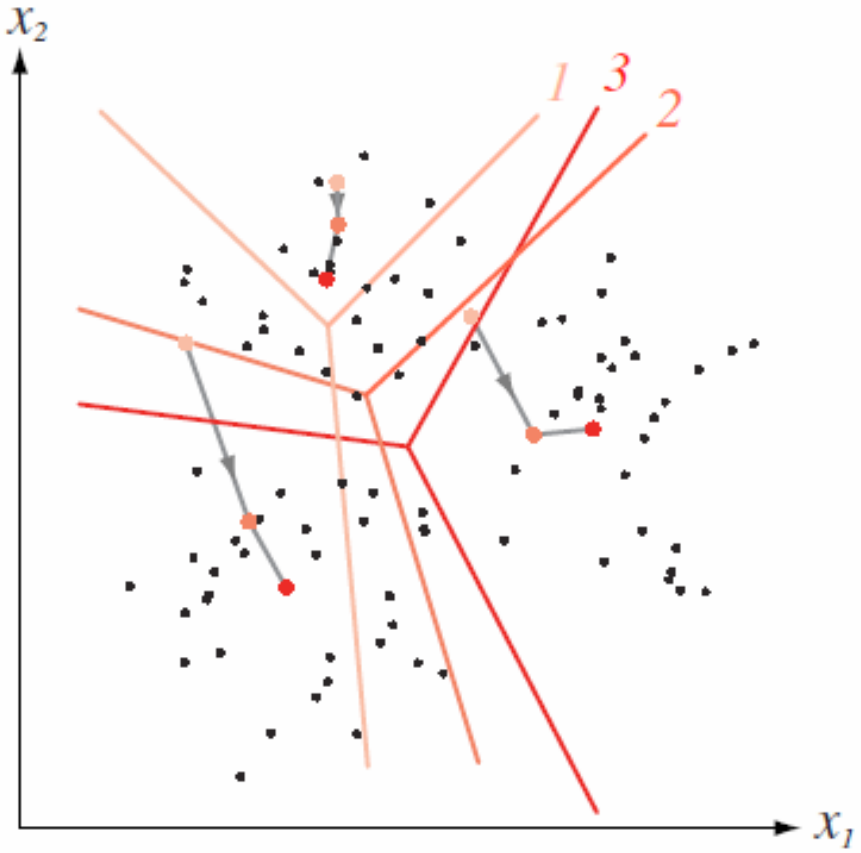
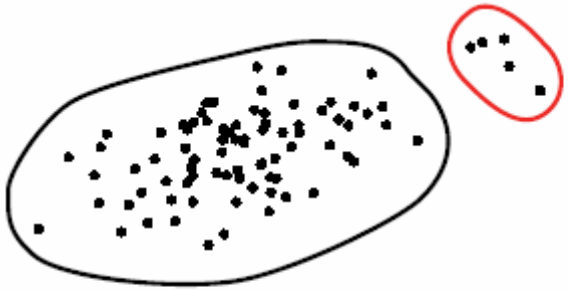


FIGURE 10.3. Trajectories for the means of the *k*-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown—the means correspond to the "centers" of the Voronoi cells. In this case, convergence is obtained in three iterations. From: Richard O. Duda, Peter E. Hart, and David G. Stork,

11 Unsupervised Learning and Clustering



J_e = *large*

J_e = *small*

FIGURE 10.10. When two natural groupings have very different numbers of points, the clusters minimizing a sum-squared-error criterion J_e of Eq. 54 may not reveal the true underlying structure. Here the criterion is smaller for the two clusters at the bottom than for the more natural clustering at the top. From: Richard O. Duda, Peter E. Hart, and

M. Liu, X.D. Jiang and A. Kot, "A Multi-Prototype Clustering Algorithm," *Pattern Recognition*, vol. 42, no. 5, pp. 689-698, May 2009.

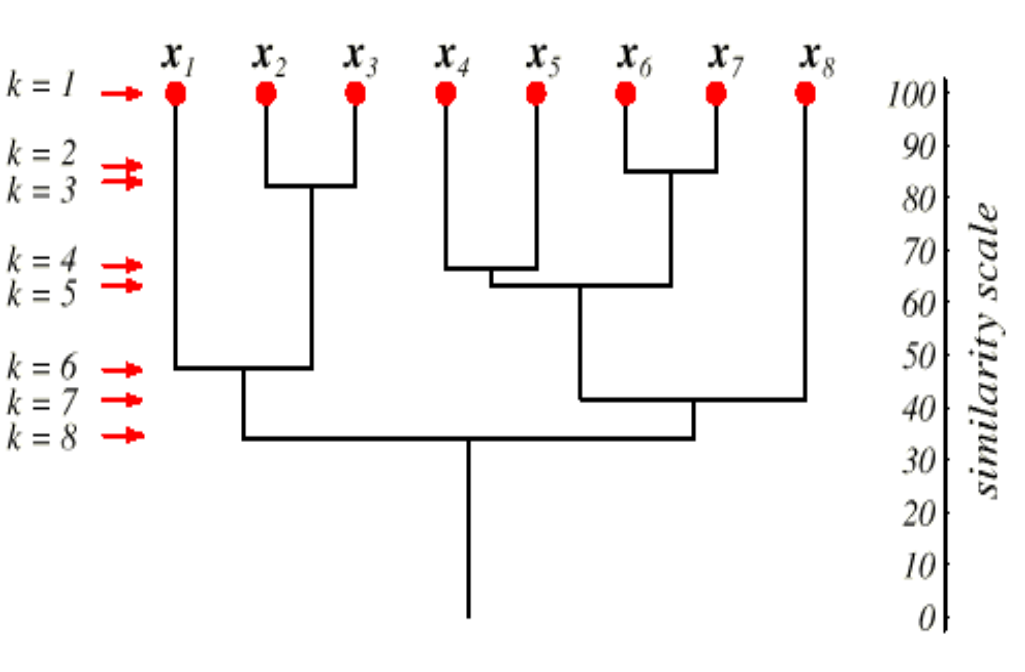
11 Unsupervised Learning and Clustering

- Hierarchical clustering can be divided in divisive and agglomerative.
- Divisive (top down, splitting): start with all of the samples in one cluster and form the sequence by successively splitting clusters
- Agglomerative (bottom up, clumping): start with n singleton cluster and form the sequence by merging clusters
  - The first is a partition into n cluster, each one containing exactly one sample
  - The second is a partition into n-1 clusters, the third into n-2, and so on, until the n-th in which there is only one cluster containing all of the samples
  - At the level k in the sequence, c = n-k+1.

11 Unsupervised Learning and Clustering

Hierarchical agglomerative clustering:

- Given any two samples x and x', they will be grouped together at some level, and if they are grouped a level k, they remain grouped for all higher levels
- A tree representation, called dendrogram, represents the sequence of clusters produced by an agglomerative algorithm.



11 Unsupervised Learning and Clustering

Example of agglomerative clustering:

- Let D = {x_i, i = 1,…,5}, with $x_1 = [1,1]^T$, $x_2 = [2,1]^T$, $x_3 = [5,4]^T$, $x_4 = [6,5]^T$, and $x_5 = [6.5,6]^T$.
  The data matrix of D is

$$D = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$$

- Its corresponding dissimilarity matrix, when Euclidean distance in use, is

$$P(D) = \begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$

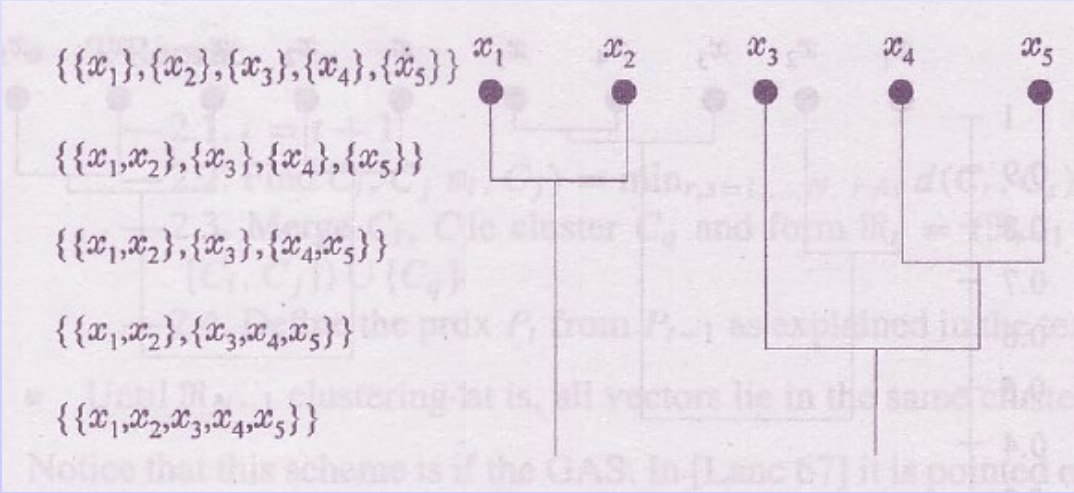
- Note that in P(D) all diagonal elements are 0, since $d_2(x_i,x_i) = 0$.

11 Unsupervised Learning and Clustering

Example of agglomerative clustering:

- Step 1. x_1 and x_2 from a new cluster.
- Step 2. x_4 and x_5 stick together, forming a single cluster.
- Step 3. x_3 joins the cluster $\{x_4, x_5\}$ and,
- Finally, at the fourth step the clusters $\{x_1, x_2\}$ and $\{x_3, x_4, x_5\}$ are merged to a single set, D.
- Cutting the dendrogram at a specific level results in a clustering

$$\begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$



11 Unsupervised Learning and Clustering

Example of agglomerative clustering:

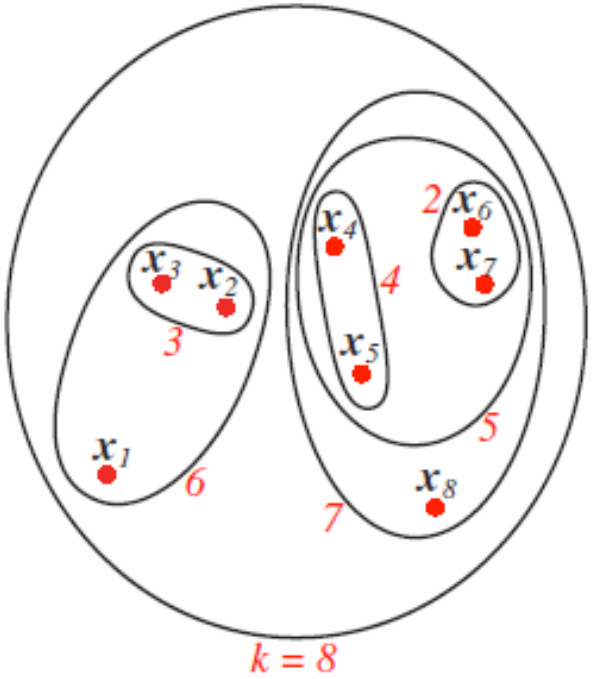


FIGURE 10.12. A set or Venn diagram representation of two-dimensional data (which was used in the dendrogram of Fig. 10.11) reveals the hierarchical structure but not the quantitative distances between clusters. The levels are numbered by k, in red. From:

11 Unsupervised Learning and Clustering

Model-based : A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.

- Assumption -- the functional forms for the underlying probability densities are known and that the only thing that must be learned is the value of an unknown parameter vector.
- Further assumptions:
  - The samples come from a known number *of* classes c
  - The prior probabilities $P(\omega_j)$ for each class are known, $(j = 1, \ldots, c)$
  - $P(x|\omega_j, \theta_j)$ $(j = 1, \ldots, c)$ are known.
  - The values of the c parameter vectors $\theta_1, \theta_2, \ldots, \theta_c$ are unknown

11 Unsupervised Learning and Clustering

Model-based :

- The category labels are unknown

$$p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \sum_{j=1}^{c} \overbrace{p(\mathbf{x}\,|\,\omega_j,\boldsymbol{\theta}_j)}^{\text{component densities}} \bullet \underbrace{P(\omega_j)}_{\text{mixing parameters}}, \quad \text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_c)^T$$

- This density function is called a mixture density.
- Our goal will be to use samples drawn from this mixture density to estimate the unknown parameter vector $\boldsymbol{\theta}$.
- Once $\boldsymbol{\theta}$ is known, we can decompose the mixture into its components and use a MAP classifier on the derived densities.
- $p(\mathbf{x}\,|\,\omega_j,\boldsymbol{\theta}_j) \sim N(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)$ → Gaussian Mixture Model (GMM).
- A popular technique to solve this problem is expectation-maximization (EM) algorithm.

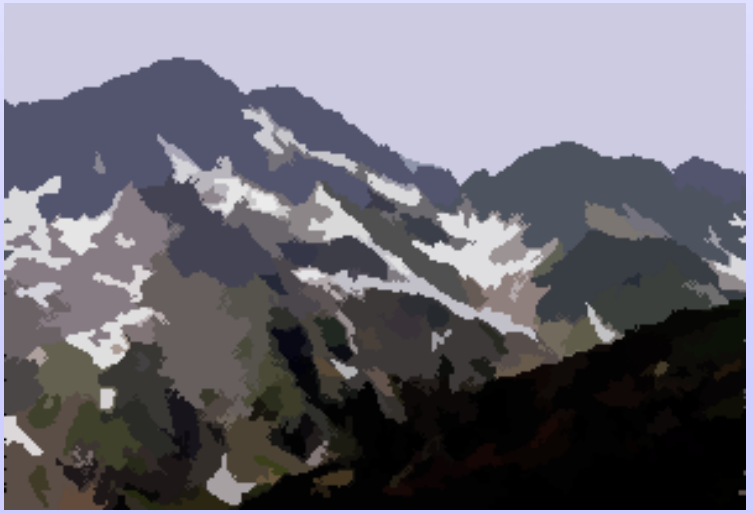
11 Unsupervised Learning and Clustering

Apply clustering approaches to image segmentation:

- Each pixel is a sample
- Select a value of k
- Select a feature vector for every pixel (color, texture, position, or combination of these etc.)
- Define a similarity measure between feature vectors (usually Euclidean distance) or assume a mixture of Gaussians distribution.
- Apply K-means or EM algorithm to find the k clusters.
- Apply Connected Components Algorithm.
- Merge any components of size less than some threshold to an adjacent component that is most similar to it.

11 Unsupervised Learning and Clustering

Example of image segmentation by clustering



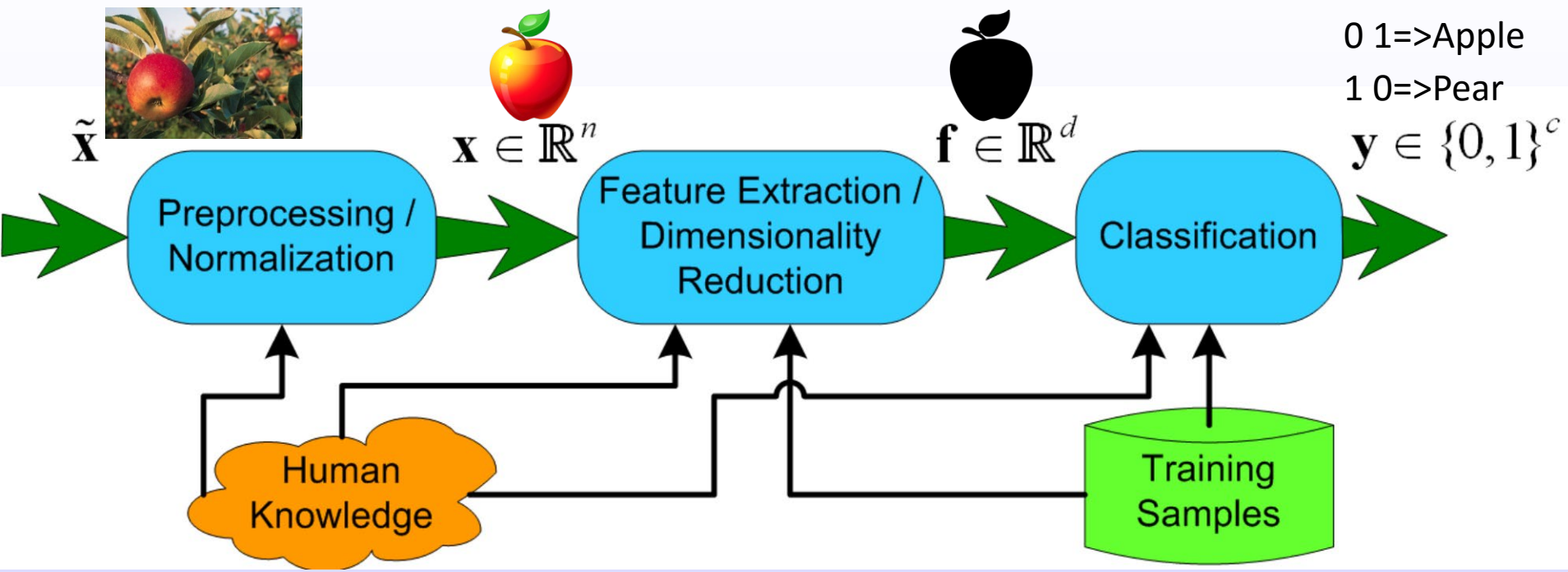
11 Unsupervised Learning and Clustering

12 Feature Extraction/Dimension Reduction v. Machine Learning

Outline:

- Restudy the Functionalities of pattern recognition modules
- Approaches to the feature extraction and dimensionality reduction
- Machine learning approaches:
  - Eigen-decomposition
  - PCA: the principal component analysis
  - LDA: the linear discriminant analysis
  - Various variants of LDA
- State-of-the-art Research on Dimensionality Reduction:
  - Questions on PCA and LDA for pattern recognition: objectives of feature extraction/dimension reduction for pattern recognition
  - Problems of statistical classification in high dimension
  - Solutions by regularization and dimensionality reduction
  - Restudy LDA

12 Feature Extraction/Dimension Reduction v. Machine Learning



$$\tilde{\mathbf{x}} \in \mathbb{R}^{w \times h}, \text{ often } w \times h > n \gg d \gg c \quad \mathbf{y} \in \{1,2,...,c\} \text{ or } \mathbf{y} \in \{0,1\}^c$$

- Pattern recognition is a series of processes that reduce the dimensionality and the variation of samples for the same class and keep their discrimination for different classes.

Dimensionality reduction and Discriminative information extraction

12 Feature Extraction/Dimension Reduction v. Machine Learning

- Pattern recognition in general is to classify an observed data into one of the classes.
- The observed data often have multiple components and the data of a same class are in general not exact same and have some variation. We thus represent the observed data by a random vector.
- The probability distribution fully characterizes a random vector.
- In most cases, the probability distribution is unknown.
- Pattern Recognition via Machine Learning is to estimate the probability distribution directly or indirectly from the known data/samples.

12 Feature Extraction/Dimension Reduction v. Machine Learning

- In some applications such as visual object detection and recognition, bioinformatics and data mining, high data dimensionality imposes great burdens on the robust and accurate recognition due to insufficient knowledge about the data population and limited number of training samples.
- Dimensionality reduction thus becomes a separate and maybe the most critical module of such recognition systems.
- Extracting the discriminative and reliable features or dimensions is always the key step for image recognition and computer vision.

- Feature extraction and dimensionality reduction can be based on
  - Human expert knowledge
  - Image local structures
  - Image global structure
  - Machine learning from training database

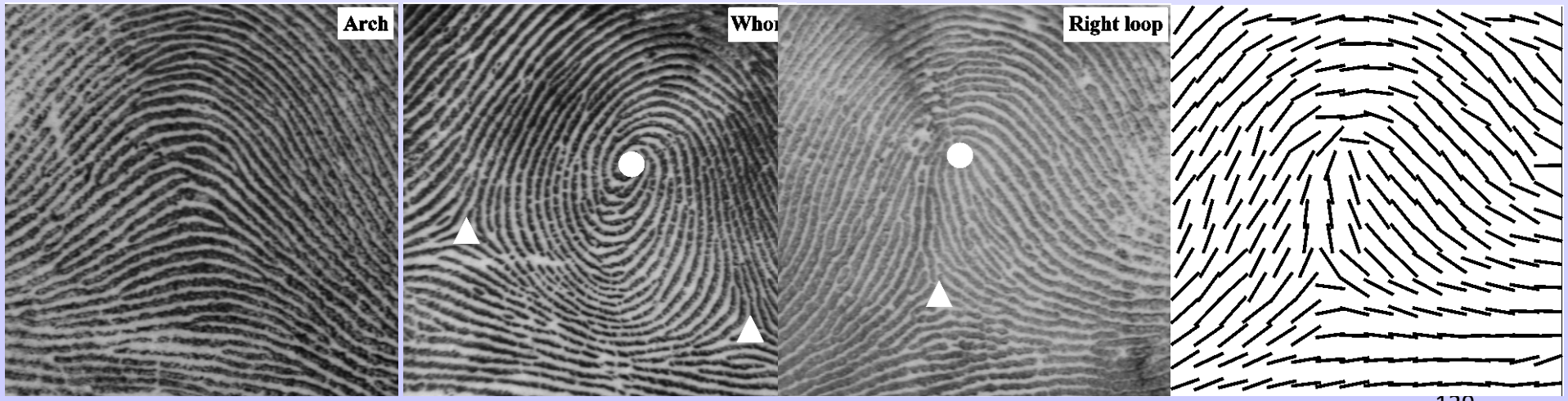
12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Human Knowledge:

- Fingerprint classification: for example, arch, whorl and loop
- Local orientations are intrinsic features for such task.(Human knowledge)
- Orientation field consisting of fingerprint local orientations represented by short lines.

X.D. Jiang, Extracting Image Orientation Feature by Using Integration Operator, *Pattern Recognition*, vol. 40, no. 2, pp. 705-717, February 2007.

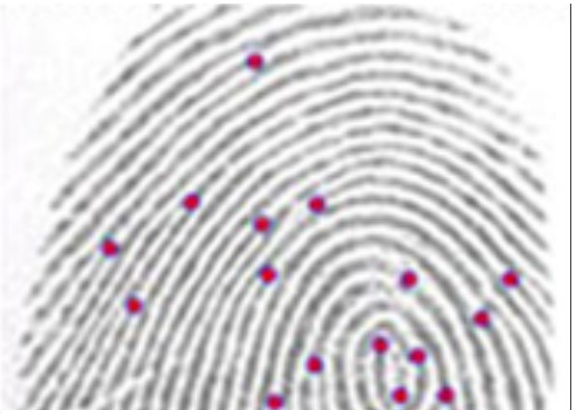
X.D. Jiang, On Orientation and Anisotropy Estimation for Online Fingerprint Authentication, *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 4038- 4049, October 2005.



12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Human Knowledge:

Fingerprint identification: Human experts teach us, the reliable and discriminative features are minutia points: where ridge terminates or bifurcates.



X.D. Jiang, W. Yau and W. Ser, "Detecting the Fingerprint Minutiae by Adaptive Tracing the Gray Level Ridge," *Pattern Recognition*, vol. 34, no. 5, pp. 999-1013, May 2001.

X.D. Jiang, M. Liu and A. Kot, "Fingerprint Retrieval for Identification," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 532-542, December 2006.

Good knowledge about the features doesn't mean the computer can reliably extract them due to poor image quality and noise.

exdjiang@ntu.edu.sg http://www.ntu.edu.sg/home/exdjiang

12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Image Local Structures:

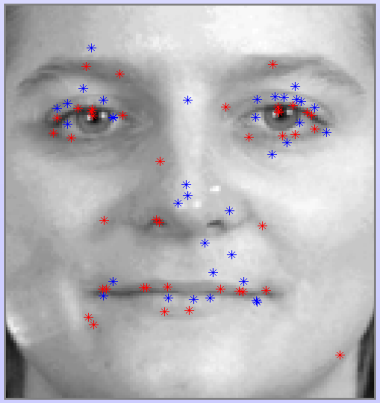
Corners and blobs called key points or interesting points have locations, scales and shapes.

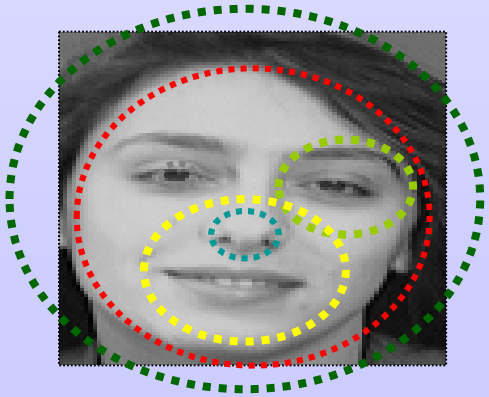
C. Geng and X.D. Jiang, "Face Recognition Based on the Multi-scale Local Image Structures," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2565-2575, October-November 2011.

C. Geng and X.D. Jiang, "Fully Automatic Face Recognition Framework based on Local and Global Features," *Machine Vision and Applications*, vol. 24, no. 3, pp. 537-549, April 2013.

Z. Miao and X.D. Jiang, "Interest Point Detection Using Rank Order LoG Filter," *Pattern Recognition*, vol. 46, no. 11, pp. 2890-2901, November 2013.

Z. Miao, X.D. Jiang and K. Yap, "Contrast Invariant Interest Point Detection by Zero-Norm LoG Filter," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 331 - 342, January 2016.





12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Image Global Structures:

Transform image $f(x,y)$ to feature $g(u,v)$

$$g(u,v) = T\{f(x,y)\}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(u,v,x,y) f(x,y) dxdy \quad \text{linear transform}$$

$$\Rightarrow \sum_1^h \sum_1^w w(u,v,x,y) f(x,y) \quad \text{for digital image}$$

$$\Rightarrow \sum_1^h \sum_1^w e^{-2\pi j(ux+vy)} f(x,y) \quad \text{Fourier transform}$$

$$\Rightarrow \sum_1^h \sum_1^w x^u y^v f(x,y) \quad \text{moments computing}$$

$$\Rightarrow \mathbf{f} = \mathbf{W}^T \mathbf{x} \quad \text{vector-matrix representation}$$

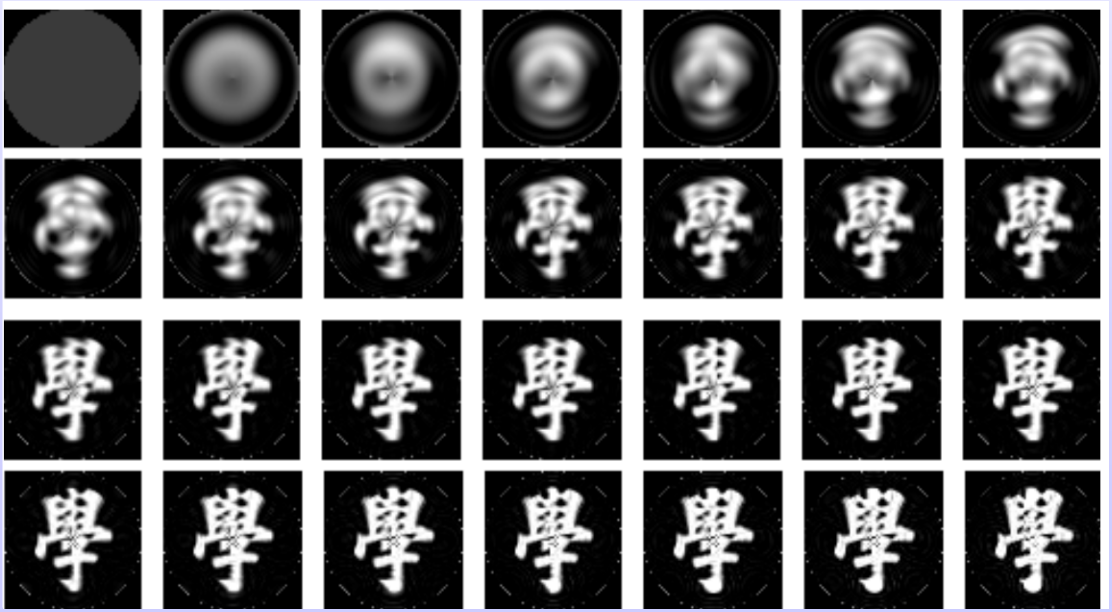
12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Image Global Structures:

Polar Complex Exponential Transform (PCET)

$$g(u,v) = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 e^{-j(2\pi ur^2 + v\theta)} f(r,\theta)\,dr\,d\theta$$

Shows good property in the image reconstruction.



P. Yap, X.D. Jiang and A. Kot, "Two Dimensional Polar Harmonic Transforms for Invariant Image Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1259-1270, July 2010.

12 Feature Extraction/Dimension Reduction v. Machine Learning

Feature Extraction Based on Image Global Structures: Histogram of gradients and histogram of LBP

J. Ren, X.D. Jiang and J. Yuan, "LBP Encoding Schemes Jointly Utilizing the Information of Current Bit and Other LBP Bits," *IEEE Signal Processing letters*, vol. 22, no. 12, pp. 2373 - 2377, Dec. 2015.

J. Ren, X.D. Jiang and J. Yuan, "A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition," *IEEE Trans. Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.

J. Ren, X.D. Jiang and J. Yuan, "Learning LBP Structure by Maximizing the Conditional Mutual Information," *Pattern Recognition*, vol. 48, no. 10, pp. 3180 - 3190, Oct. 2015.

J. Ren, X.D. Jiang, J. Yuan and W. Gang, "Optimizing LBP Structure for Visual Recognition Using Binary Quadratic Programming," *IEEE Signal Processing letters*, vol. 21, no. 11, pp. 1346-1350, Nov. 2014.

A. Satpathy, X.D. Jiang and H. Eng, "LBP Based Edge-Texture Features for Object Recognition," *IEEE Trans. Image Processing*, vol. 23, no. 5, pp. 1953-1964, May, 2014.

A. Satpathy, X.D. Jiang and H. Eng, "Human Detection by Quadratic Classification on Subspace of Extended Histogram of Gradients," *IEEE Trans. Image Processing*, vol. 23, no. 1, pp. 287-297, Jan, 2014.

J. Ren, X.D. Jiang and J. Yuan, "Noise-Resistant Local Binary Pattern with an Embedded Error-Correction Mechanism," *IEEE Trans. Image Processing*, vol. 22, no. 10, pp. 4049-4060, Oct, 2013.

12 Feature Extraction/Dimension Reduction v. Machine Learning

Problems:

- Poor or even no human knowledge about the effective features.
- Image patterns are so complex that it is impossible to incorporate all different variations into the deterministic computer program.

$$[300 \times 200] \Rightarrow \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{60000} \end{pmatrix} \Rightarrow \mathbf{f} = \begin{bmatrix} f \\ \vdots \\ f_{60} \end{bmatrix} = \mathbf{\Phi}^T \mathbf{x}$$

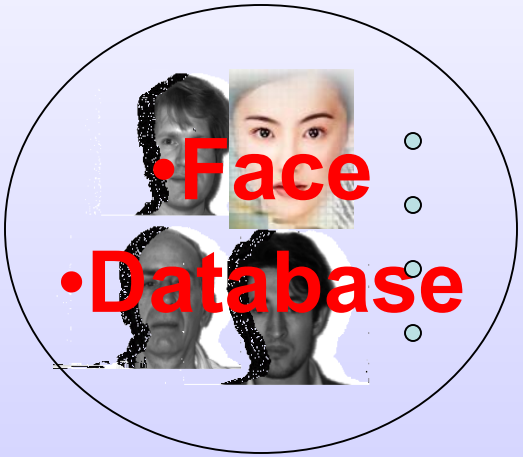
Feature Scaling,
Dimension Reduction
Feature Extraction

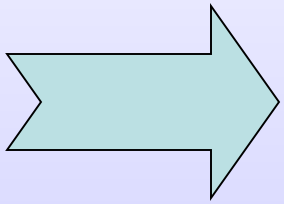
Solution: Take all pixels of the whole image as initial features and derive the effective features based on machine learning.

A real application example: J. Ren, X.D. Jiang and J. Yuan, "A Complete and Fully Automated Face Verification System on Mobile Devices," Pattern Recognition, vol. 46, no. 1, pp. 45-56 January 2013.

12 Feature Extraction/Dimension Reduction v. Machine Learning

The transform parameter W or Φ that weight the features and reduce the feature dimensionality is determined by machine (computer) learning (training) from a image database.





W Φ

- Why is the transform necessary?
- what principles are used to derive W ?
- What are the potential problems?
- How to alleviate these problems?

12 FE/DR v. ML --PCA: Principal Component Analysis

Given a data set of *q* *n*-dimensional training samples

$$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_q$$

Each sample represented by a column vector is a point in an *n*-dimensional space.

How to use one point, $\mathbf{x}_0$ to best represent all training data?

i.e. $$\varepsilon^2 = \sum_{i=1}^{q} \|\mathbf{x}_0 - \mathbf{x}_i\|^2 = \sum_{i=1}^{q} (\mathbf{x}_0 - \mathbf{x}_i)^T (\mathbf{x}_0 - \mathbf{x}_i) \Rightarrow \text{minimum}$$

- It is very easy to prove that the solution is the sample mean

$$\mathbf{x}_0 = \boldsymbol{\mu} = \frac{1}{q} \sum_{i=1}^{q} \mathbf{x}_i$$

Just for symbolic simplicity, we centralize training samples and define a training data matrix by

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i - \boldsymbol{\mu}, \qquad \mathbf{X} = \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \dots & \tilde{\mathbf{x}}_q \end{bmatrix}$$

12 FE/DR v. ML --PCA: Principal Component Analysis

- Now we want to just use one dimension $\phi, \quad \| \phi \|^2 = \phi^T \phi = 1$ to best represent all samples, $\tilde{\mathbf{x}}_i$.
- The *n*-dimensional data $\tilde{\mathbf{x}}_i$ are reduced to one-dimensional data.

$$a_i = \phi^T \tilde{\mathbf{x}}_i$$

- The best dimensionϕ makes reconstruction error minimum.

$$\varepsilon^2 = \sum_{i=1}^{q} \| \tilde{\mathbf{x}}_i - a_i \phi \|^2 \Rightarrow \text{minimum}$$

- It is easy to have

$$\varepsilon^2 = \sum_{i=1}^{q} \| \tilde{\mathbf{x}}_i - a_i \phi \|^2 = \sum_{i=1}^{q} (\tilde{\mathbf{x}}_i - a_i \phi)^T (\tilde{\mathbf{x}}_i - a_i \phi)$$

$$= \sum_{i=1}^{q} \| \tilde{\mathbf{x}}_i \|^2 - \sum_{i=1}^{q} a_i^2 \quad \left(\text{note: } a_i \phi^T \tilde{\mathbf{x}}_i = a_i^2, \; a_i \phi^T a_i \phi = a_i^2 \phi^T \phi = a_i^2 \right)$$

$$= \sum_{i=1}^{q} \| \tilde{\mathbf{x}}_i \|^2 - \sum_{i=1}^{q} \phi^T \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \phi$$

$$= \sum_{i=1}^{q} \| \tilde{\mathbf{x}}_i \|^2 - \phi^T \left(\sum_{i=1}^{q} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \right) \phi$$

12 FE/DR v. ML --PCA: Principal Component Analysis

- The sample covariance matrix of all training data,

$$\mathbf{S}^t = \frac{1}{q}\sum_{i=1}^{q}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{q}\sum_{i=1}^{q}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T$$

  is also called the total scatter matrix in the literature.

- To minimize $\varepsilon^2 = \sum_{i=1}^{q}\| \tilde{\mathbf{x}}_i \|^2 - q\phi^T\mathbf{S}^t\phi$

  is to maximize $\phi^T\mathbf{S}^t\phi$ with the constraint of $\| \phi \|^2 = \phi^T\phi = 1$

- Use Lagrange optimization method

$$f(\phi,\lambda) = \phi^T\mathbf{S}^t\phi - \lambda(\phi^T\phi - 1) \Rightarrow \max imum$$

$$\frac{\partial f}{\partial \phi} = 2\mathbf{S}^t\phi - 2\lambda\phi = 0$$

$$\mathbf{S}^t\phi = \lambda\phi$$

- The solution is eigenvalue and eigenvector of matrix $\mathbf{S}^t$

12 FE/DR v. ML --PCA: Principal Component Analysis

- We see that if ϕ is the eigenvector of the covariance matrix $\mathbf{S}^t$, $\phi^T \mathbf{S}^t \phi$ is maximum and the reconstruction error is minimum.

- The variance of the data projected onto the dimension spanned by the eigenvector is maximum.

$$\phi^T \mathbf{S}^t \phi = \frac{1}{q} \sum_{i=1}^{q} \phi^T (\mathbf{x}_i - \boldsymbol{\mu}) \left[\phi^T (\mathbf{x}_i - \boldsymbol{\mu}) \right]^T = \frac{1}{q} \sum_{i=1}^{q} a_i^2$$

- Eigenvalue of a covariance matrix is the variance of the data projected onto the eigenvetor.

$$\because \mathbf{S}^t \phi = \lambda \phi \quad \therefore \phi^T \mathbf{S}^t \phi = \phi^T \lambda \phi = \lambda \phi^T \phi = \lambda$$

12 FE/DR v. ML --PCA: Principal Component Analysis

- Reducing the n-dimensional data $\mathbf{x}_i$ into lower m-dimensional data $\mathbf{y}_i$ by

$$\mathbf{y}_i = \mathbf{\Phi}^T (\mathbf{x}_i - \boldsymbol{\mu})$$

where

$$\mathbf{\Phi} = [\phi_1 \ \phi_2 \ \ldots \phi_m], \quad m < n$$

consists of m eigenvectors corresponding to the m largest eigenvalues of the total scatter matrix $\mathbf{S}^t$.

- We can reconstruct the original n-dimensional data $\mathbf{x}_i$ using the lower m-dimensional data $\mathbf{y}_i$ using $\hat{\mathbf{x}}_i = \mathbf{\Phi}\mathbf{y}_i + \boldsymbol{\mu}$
- The reconstruction error is minumum.

$$\varepsilon^2 = \sum_{i=1}^{q} \| \mathbf{x}_i - \hat{\mathbf{x}}_i \|^2 \Rightarrow \text{minimum}$$

12 FE/DR v. ML --PCA: Principal Component Analysis

- Therefore, the famous principal component analysis PCA transforms the *n*-dimensional *X* into *m*-dimensional *Y* by:

$$\mathbf{y} = \mathbf{\Phi}^T (\mathbf{x} - \boldsymbol{\mu})$$

where $\mathbf{\Phi} = [\phi_1 \ \phi_2 \ \ldots \ \phi_m], \quad m < n$

consists of m eigenvectors corresponding to the m largest eigenvalues of the total scatter matrix $\mathbf{S}^t$.

The reconstruction error is: $\Delta = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{\Phi y} - \boldsymbol{\mu}$

and the mean squared reconstruction error

$$E[\| \Delta \|^2] = E[\Delta^T \Delta] = \sum_{k=m+1}^{n} \lambda_k$$

where λ_k are eigenvalues sorted in descending order.

12 FE/DR v. ML --PCA: Principal Component Analysis

- Note that:

$$\mathbf{S}^t = \frac{1}{q}\sum_{i=1}^{q}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{q}\sum_{i=1}^{q}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T = \frac{1}{q}\mathbf{X}\mathbf{X}^T$$

- The rank of the total scatter matrix $\mathbf{S}^t$ is min(q-1,n) at most. Therefore, it has q-1 nonzero eigenvalues at most.
- Thus, PCA with q-1 dimensional $\mathbf{y}$ will fully represent the original n-diemsional data $\mathbf{x}$, q-1$\leq n$, because the reconstruction error is zero.

$$E[\| \Delta \|^2] = \sum_{k=q}^{n}\lambda_k = 0$$

- Obviously, the scatter matrix or covariance matrix is a *nxn* symmetry matrix.

$$\mathbf{S}^t = (\mathbf{S}^t)^T$$

12 FE/DR v. ML -- Eigen-decomposition

- Eigenvalue and eigenvector of a nxn matrix Σ is defined mathematically by:

$$\Sigma\phi_i = \lambda_i\phi_i, \qquad i = 1,\ 2,\ \ldots,\ n$$

- If Σ is a symmetry matrix, eigenvectors corresponding to the distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are orthogonal. Take the unit length of $\phi_1, \phi_2, \ldots, \phi_n$

$$\phi_i^T \phi_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

- Prove it ! (orthogonal+unit length is called orthonormal)

12 FE/DR v. ML -- Eigen-decomposition

We have $\Sigma\phi_k = \lambda_k\phi_k$ and $\Sigma\phi_j = \lambda_j\phi_j, \quad k \neq j$

$\therefore \quad \phi_j^T\left(\Sigma\phi_k\right) = \phi_j^T\left(\lambda_k\phi_k\right)$ and $\left(\Sigma\phi_j\right)^T\phi_k = \left(\lambda_j\phi_j\right)^T\phi_k$

$\therefore \quad \phi_j^T\Sigma\phi_k = \lambda_k\phi_j^T\phi_k$ and $\phi_j\Sigma^T\phi_k = \lambda_j\phi_j^T\phi_k$

$\because \Sigma^T = \Sigma$, we have $\lambda_k\phi_j^T\phi_k = \lambda_j\phi_j^T\phi_k$

$\therefore (\lambda_k - \lambda_j)\phi_j^T\phi_k = 0$

$\because (\lambda_k - \lambda_j) \neq 0$

$\therefore \phi_j^T\phi_k = 0$

12 FE/DR v. ML -- Eigen-decomposition

- Let Φ be the orthonormal matrix formed by the eigenvectors

$$\mathbf{\Phi} = [\phi_1 \ \phi_2 \ \ldots \ \phi_n]$$

$$\text{Obviously: } \mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I} \qquad \therefore \mathbf{\Phi}^{-1} = \mathbf{\Phi}^T, \qquad \therefore \mathbf{\Phi}\mathbf{\Phi}^T = \mathbf{I}$$

- Let Λ be a diagonal matrix:

$$\mathbf{\Lambda} = diag(\lambda_1 \ \lambda_2 \ \ldots \ \lambda_n) = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}$$

- From $\mathbf{\Sigma}\phi_i = \lambda_i\phi_i, \qquad i = 1, 2, \ldots, n$

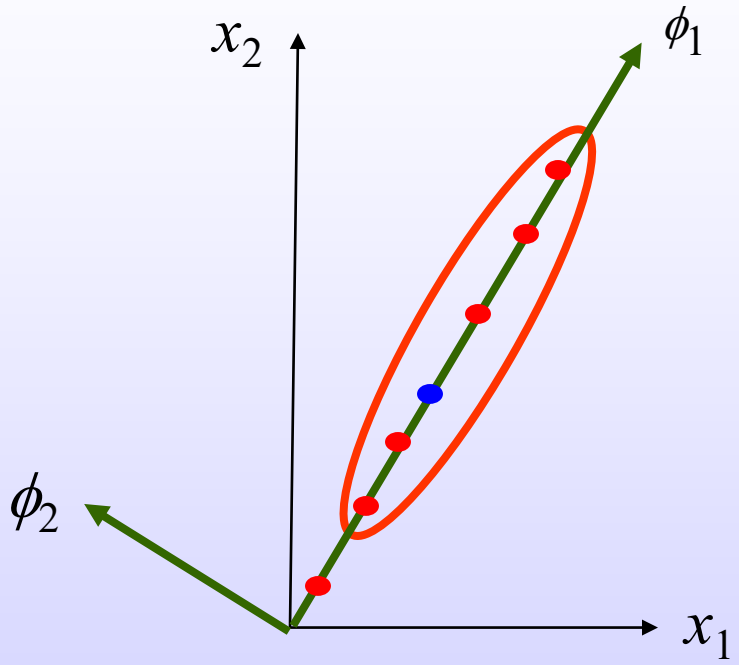
- We have $\mathbf{\Sigma\Phi} = \mathbf{\Phi\Lambda} \qquad \therefore \mathbf{\Sigma} = \mathbf{\Phi\Lambda\Phi}^T$, or $\mathbf{\Lambda} = \mathbf{\Phi}^T\mathbf{\Sigma\Phi}$

12 FE/DR v. ML – understand Eigen-decomposition

$$\boldsymbol{\Sigma}_x = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \phi_1 & \phi_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 \end{bmatrix}^T$$

$$= \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$$

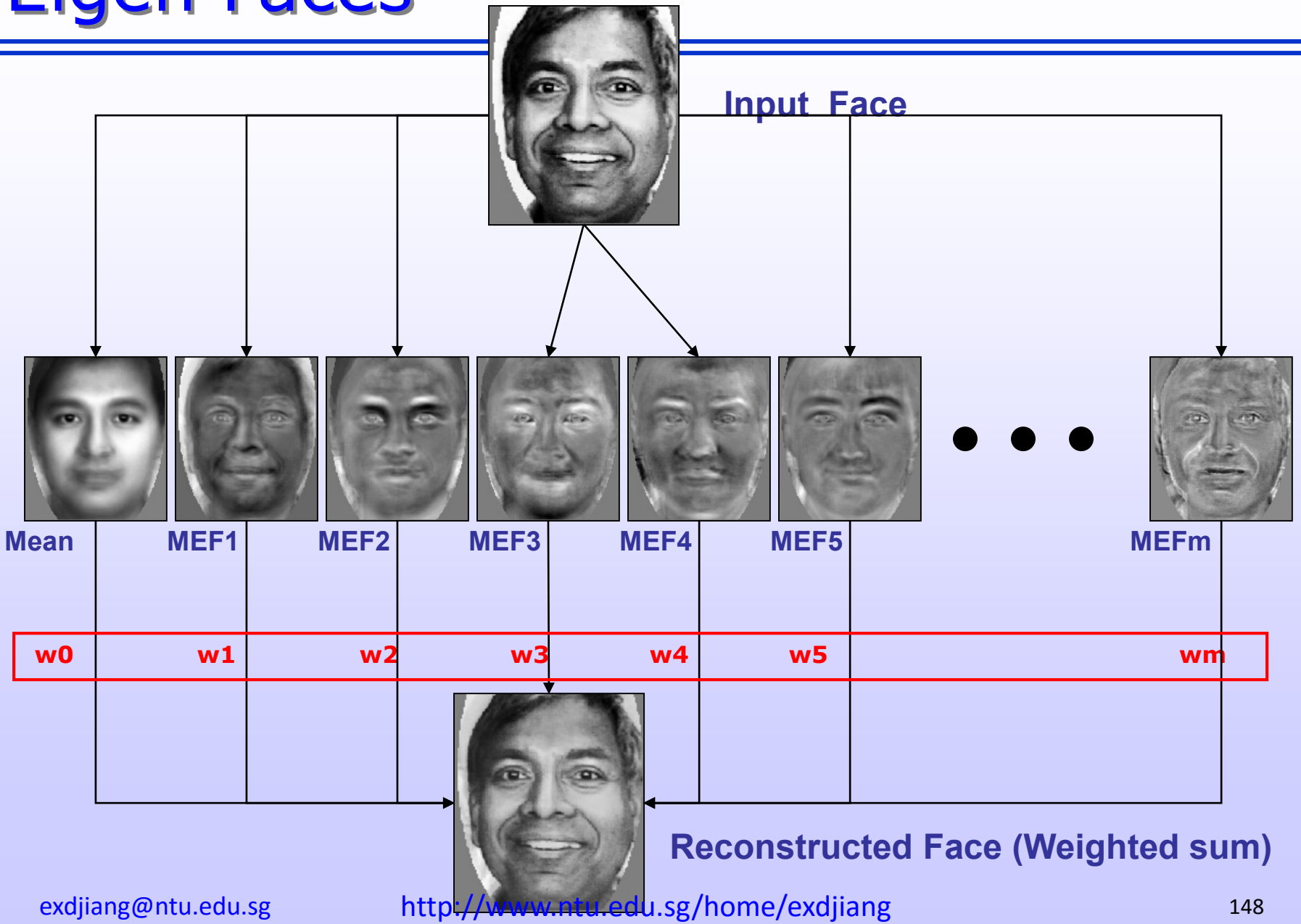


$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_x \boldsymbol{\Phi}$$

$$= \boldsymbol{\Phi}^T \mathbf{X}\mathbf{X}^T \boldsymbol{\Phi} = \boldsymbol{\Phi}^T \mathbf{X} \left(\boldsymbol{\Phi}^T \mathbf{X} \right)^T = \mathbf{Y}\mathbf{Y}^T = \boldsymbol{\Sigma}_y = \boldsymbol{\Lambda}$$

where: $\mathbf{Y} = \boldsymbol{\Phi}^T \mathbf{X} = \mathbf{W}^T \mathbf{X}$

Eigen Faces

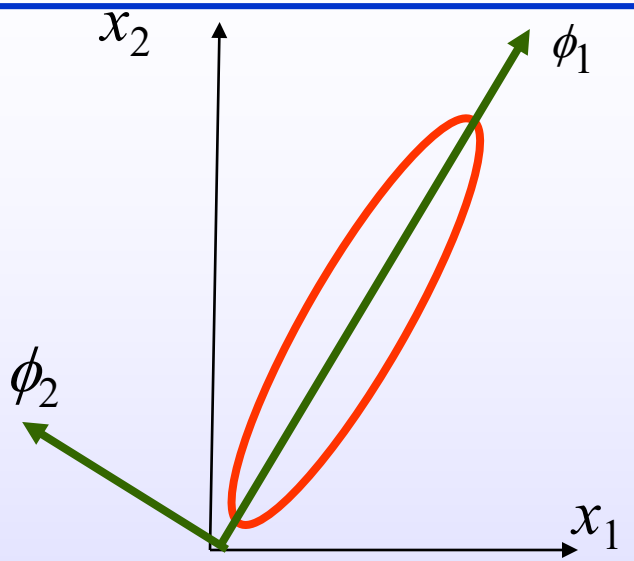

Input Face
Mean
MEF1
MEF2
MEF3
MEF4
MEF5
MEFm
w0
w1
w2
w3
w4
w5
wm
Reconstructed Face (Weighted sum)

12 FE/DR v. ML --PCA: Principal Component Analysis

$$PCA: \quad \max trace[\mathbf{\Phi}^T \mathbf{S}^t \mathbf{\Phi}] = \sum_{k=1}^{m} \lambda_k^t$$

$$\mathbf{\Phi}: \ [n \times m], \text{ e.g. } =[60000 \times 80]$$

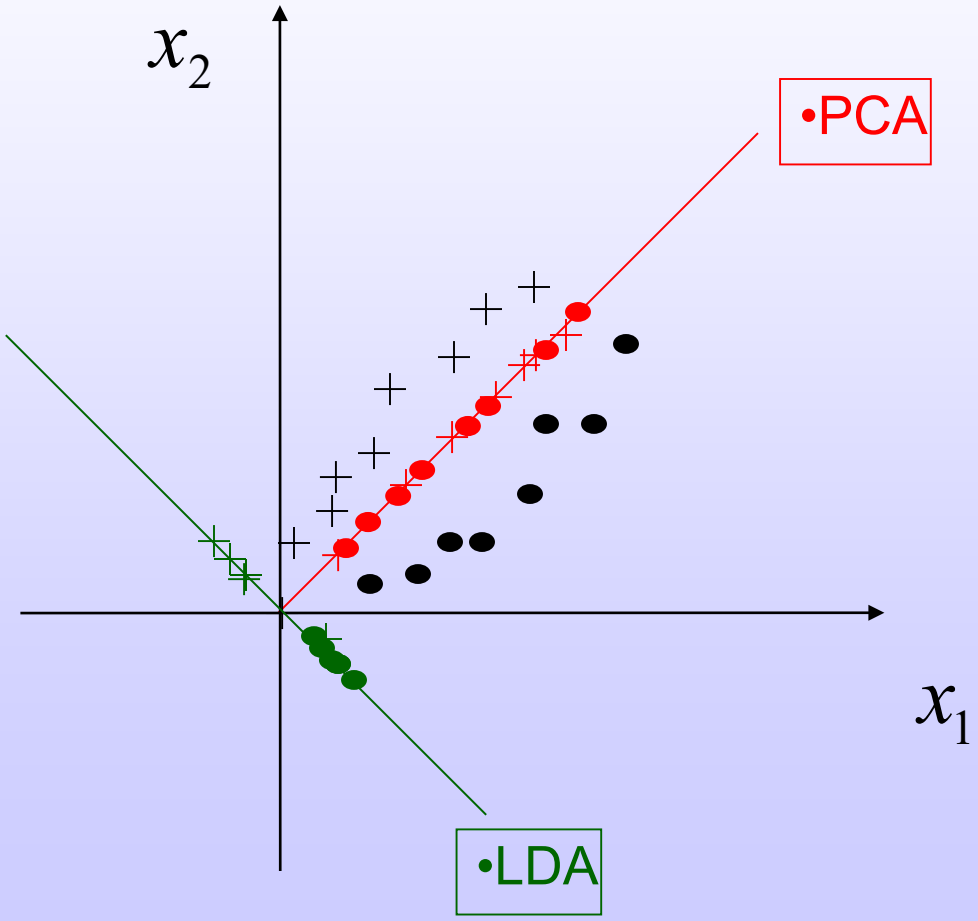
$$\mathbf{y} = \mathbf{\Phi}^T (\mathbf{x} - \boldsymbol{\mu})$$



- PCA is an unsupervised method that minimizes the reconstruction error or maximize the variation (information carried by the data) in the reduced sub-space. This dimension reduction will reduce the computational complexity of the subsequent processing.
- However, any information lost may reduces the accuracy of the subsequent processing. Further more, the lost information, though less important for data representation, could be critical for discriminating the data class membership.

12 FE/DR v. ML --LDA: Linear Discriminant Analysis

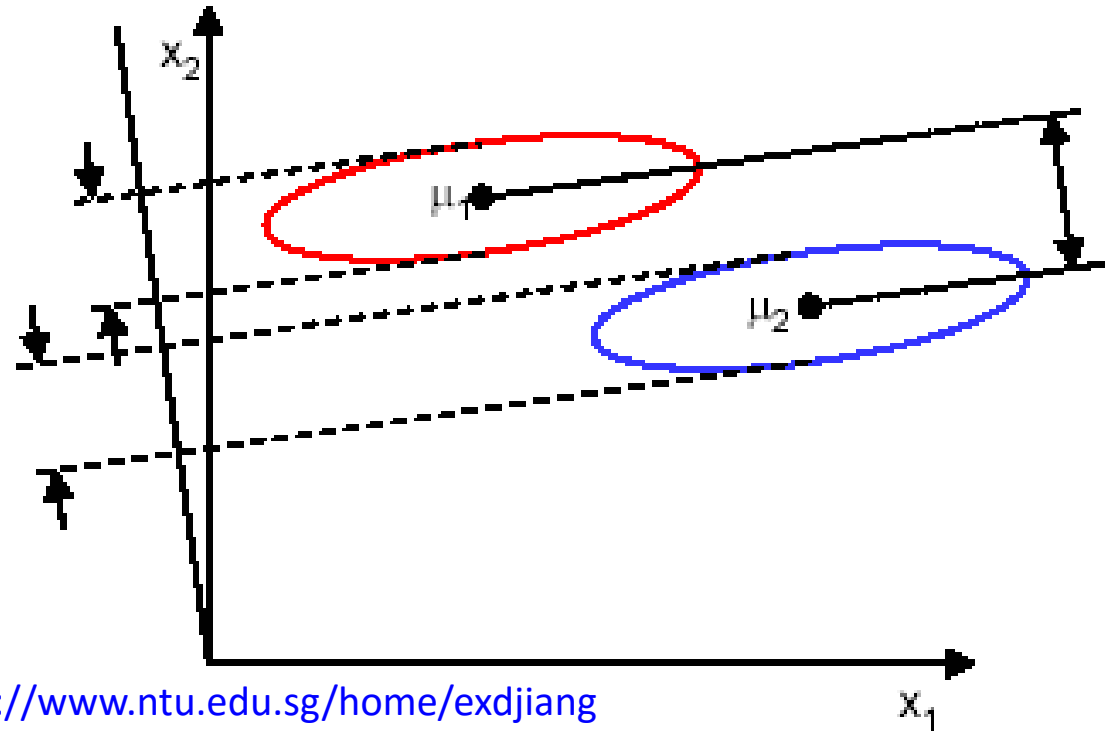
- Problem of PCA in classification?



12 FE/DR v. ML --LDA: Linear Discriminant Analysis

- Desired? properties to determine the projection vector for classification :
  - Maximize separation between projected class means
  - Minimize projected within-class scatter (variance)

$$\mathbf{y} = \mathbf{\Phi}^T (\mathbf{x} - \boldsymbol{\mu})$$



exdjiang@ntu.edu.sg

12 FE/DR v. ML --LDA: Linear Discriminant Analysis

- Given q n-dimensional training samples of c classes

$$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_q$$

- The number of training samples of class ω_j is q_j, j = 1, 2,… c.
- The covariance matrix of class ω_j is computed as

$$\boldsymbol{\Sigma}_j = \frac{1}{q_j}\sum_{X_i \in \omega_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T, \quad \text{where } \boldsymbol{\mu}_j = \frac{1}{q_j}\sum_{X_i \in \omega_j} \mathbf{x}_i$$

- The within class scatter matrix of the c classes is defined as

$$\mathbf{S}^w = \sum_{j=1}^{c} \frac{q_j}{q} \boldsymbol{\Sigma}_j$$

- The between class scatter matrix of c classes is defined as

$$\mathbf{S}^b = \sum_{j=1}^{c} \frac{q_j}{q} (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T, \text{ obviously: } \boldsymbol{\mu} = \frac{1}{q}\sum_{i=1}^{q} \mathbf{x}_i = \sum_{j=1}^{c} \frac{q_j}{q} \boldsymbol{\mu}_j$$

12 FE/DR v. ML --LDA: Linear Discriminant Analysis

- Instead of $PCA: \quad \max trace[\mathbf{\Phi}^T \mathbf{S}^t \mathbf{\Phi}] = \sum_{k=1}^{m} \lambda_k^t$

$$LDA: \quad \min trace\left[\mathbf{\Phi}^T \mathbf{S}^w \mathbf{\Phi}\right] \ \& \ \max trace\left[\mathbf{\Phi}^T \mathbf{S}^b \mathbf{\Phi}\right]$$

$$\Rightarrow LDA: \quad \max trace\left[\mathbf{\Phi}^T \mathbf{S}^{w^{-1}} \mathbf{S}^b \mathbf{\Phi}\right] = \sum_{k=1}^{m} \lambda_k^{b/w}$$

- Note that:

$trace(\mathbf{S}^t) \Rightarrow$ total variation amount of all samples

$trace(\mathbf{S}^w) \Rightarrow$ total variation amount of samples within classes

$trace(\mathbf{S}^b) \Rightarrow$ total variation amount of samples between classes

- It is easy to prove:

$$\mathbf{S}^t = \mathbf{S}^w + \mathbf{S}^b$$

12 FE/DR v. ML --LDA: Linear Discriminant Analysis

- As pattern recognition is not to represent the original data, but to discriminate the class membership of the data, obviously, LDA extracts much more discriminative features than PCA. Therefore, the vast majority of researchers prefer LDA to PCA for feature extraction and dimensionality reduction for pattern recognition.

$$LDA: \quad \max trace\left[\mathbf{\Phi}^T \mathbf{S}^{w^{-1}} \mathbf{S}^b \mathbf{\Phi}\right] = \sum_{k=1}^{m} \lambda_k^{b/w}$$

- The rank of $\mathbf{S}^w$ is $min(q\text{-}c,n)$ at most. For many applications $q\text{-}c<<n$. So, we have problem because $\mathbf{S}^w$ is singular and its inverse does not exist.
- Thousands of research papers proposed various LDA variants trying to solve this problem!

12 FE/DR v. ML – Major Efforts in LDA Variants

$$LDA: \max trace\left[\mathbf{\Phi}^T \mathbf{S}^{w^{-1}} \mathbf{S}^b \mathbf{\Phi}\right] = \sum_{k=1}^{m} \lambda_k^{b/w} \qquad \mathbf{S}^t = \mathbf{S}^w + \mathbf{S}^b$$

n-D data space of X

$(q\text{-}1)$-D range space of $\mathbf{S}^t$ $\quad GLDA: \max\left[\mathbf{\Phi}^T \mathbf{S}^{t^{-1}} \mathbf{S}^b \mathbf{\Phi}\right]$

$(q\text{-}c)$-D principal space of $\mathbf{S}^t$

$PCA + LDA: \max\left[\mathbf{\Phi}^T \tilde{\mathbf{S}}^{w^{-1}} \tilde{\mathbf{S}}^b \mathbf{\Phi}\right]$

$(q\text{-}c)$-D range space of $\mathbf{S}^w$

$PCA + LDA: \max\left[\mathbf{\Phi}^T \hat{\mathbf{S}}^{w^{-1}} \hat{\mathbf{S}}^b \mathbf{\Phi}\right]$ $\quad NDA: \max\left[\mathbf{\Phi}^T \hat{\mathbf{S}}^b \mathbf{\Phi}\right]$

(m)-D principal space of $\mathbf{S}^w$ $\quad$ $(q\text{-}1\text{-}m)$-D complement space of $\mathbf{S}^w$

dual space: $\max\left[\mathbf{\Phi}^T \hat{\mathbf{S}}^{w^{-1}} \hat{\mathbf{S}}^b \mathbf{\Phi}\right] + \max\left[\mathbf{\Phi}^T \overline{\mathbf{S}}^b \mathbf{\Phi}\right]$

$(c\text{-}1)$-D range space of $\mathbf{S}^b$

$DLDA: \max\left[\mathbf{\Phi}^T \tilde{\mathbf{S}}^{w^{-1}} \mathbf{S}^b \mathbf{\Phi}\right]$

- locality preserving projections LPP

All methods extract DT, or remove RT, same as classifiers.

Why need do so before classification ?

Extracting DT or removing RT is obvious. straightforwa rd.

12 FE/DR v. ML –Major efforts in LDA variants

- PCA extracts the most representative information in the sense of the least square error. LDA extracts the most discriminative information in the linear constrain.
- As it is widely believe that the discriminative information is the most important for pattern recognition, the vast majority of researchers prefer LDA to PCA for pattern recognition. Numerous efforts were made in the past two decades to make LDA workable or propose even more discriminative approaches than LDA.
- Although they are suboptimal (GLDA, dural space), or have lost some discriminative information before LDA (PCA+LDA, DLDA, NDA), numerous LDA variants makes LDA workable.
- No doubt, LDA is better than PCA in extracting discriminative information. The dimensionality reduction can greatly reduce the computation complexity of the subsequent classification.

12 FE/DR v. ML –Questions on DR, PCA and LDA

- However, most approaches (if not all) apply a simple classifier after PCA/LDA. The DR + classification needs even more computational resource than classification in the original high dimensional space. Thus, the claim that DR reduces the computational complexity of a pattern recognition system may not be valid.
- What is other purpose of DR? Can DR enhances or improves the recognition accuracy and robustness?
- Any DR loses information. Extracting the most discriminative information just means losing least information. If higher accuracy can be achieved with less information, why the criterion of DR is set to extract the most discriminative information?
- Superficial study will cause misunderstanding and mistakes and hence insignificant (trivial) research.

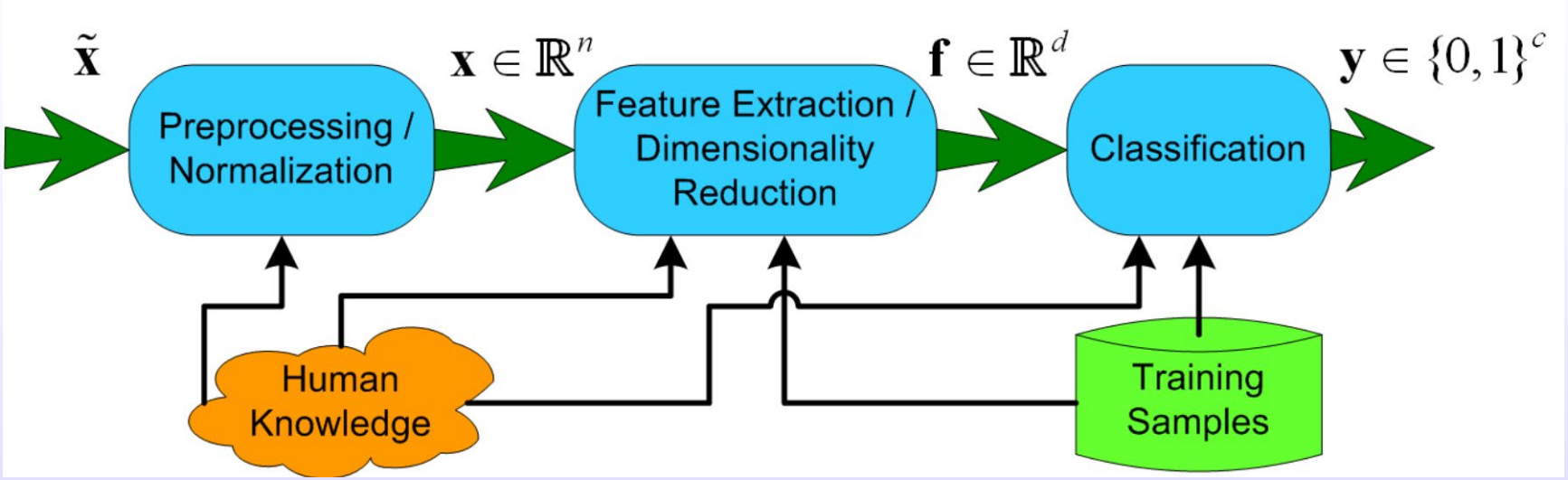
12 FE/DR v. ML –Objectives of FE/DR for PR

- We need rethink hard many related issues in-depth.
- This study (from this slide to the end of topic 12) has been published in:
  X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.
  X.D. Jiang, B. Mandal and A. Kot, "Eigenfeature Regularization and Extraction in Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 383-394, March 2008.
  X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.
- Feature extraction and dimensionality reduction has two objectives: one is to reduce the computational complexity of the subsequent classification and the other is to facilitate an even more accurate or reliable classification.
- There is no doubt that LDA is superior to PCA for the first objective.
- How can the much more important second objective be achieved?
- Can PCA and LDA help enhance the classification accuracy? Why or why not? If yes, to what extent and how? This is far from straightforward.

12 FE/DR v. ML –Objectives of FE/DR for PR

- It is theoretically proven: the probability of misclassification decreases or at least does not increase as the data dimensionality increases, as long as the decision is based on the knowledge about the whole data population.
- However, it is also well known that high dimensionality often degrades the classification performance in practice for a fixed number of training data (curse of dimensionality).
- This paradox can be resolved by distinguishing the discriminative information about the data population from that on the training data set.
- Although the ultimate objective of all modules of a pattern recognition system is to extract the most discriminative information, it is the most discriminative information about the whole data population, not on a specific training set.

12 FE/DR v. ML –Objectives of FE/DR for PR



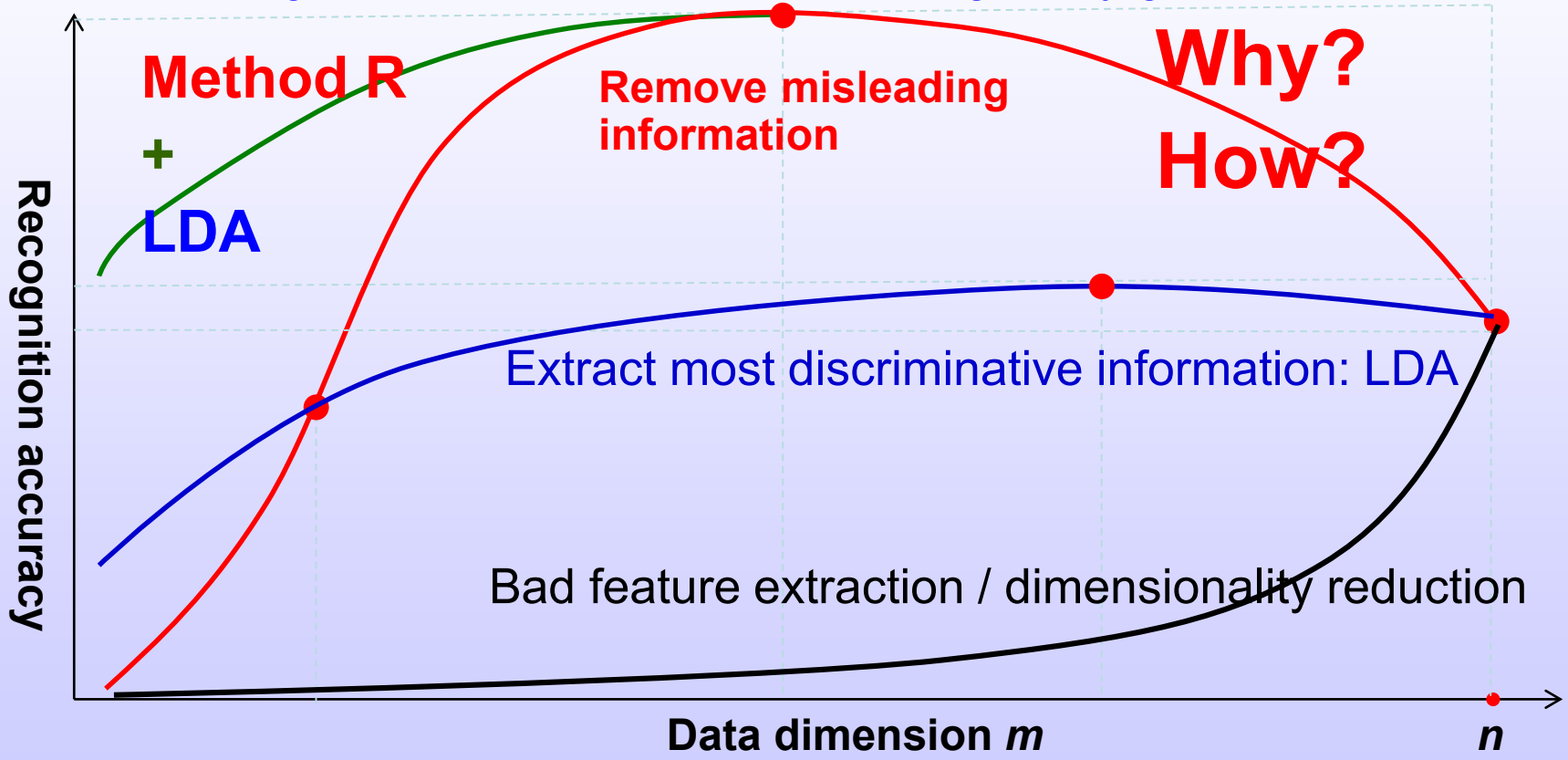
- All modules of the system, though having different physical procedures, have the **same objective**: extract the discriminative information on population (**DP**), or remove the redundant information on population (**RP**).
- A proper classifier extracts discriminative information on training data (**DT**), or remove the redundant information on training data (**RT**).

12 FE/DR v. ML –Objectives of FE/DR for PR

Recognition accuracy decreases with decreasing dimensionality

Information = discriminative information + redundant information

- Minimizing the lost doesn't mean we can get any gain!



Where can the recognition accuracy gain come from?

Information = reliable information + misleading information

A Simple Example shows DP, RP, DT, RT and MLI

How can dimension reduction enhance the classification accuracy?

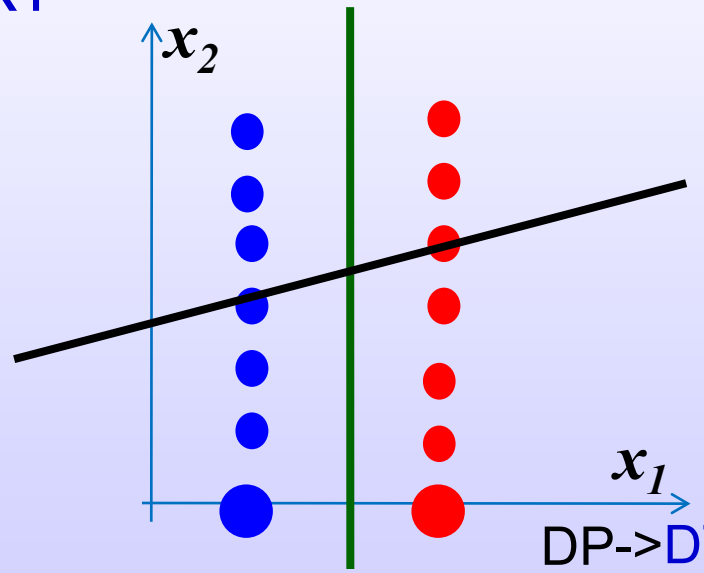
A proper Classifier extracts DT and removes / ignores RT.

RP->DT ->MLI =RT

MLI = RP∩DT

Removing RT or extracting DT before classification cannot enhance the recognition accuracy.

It can only simplify or speed up the classification.



DP->DT =DT

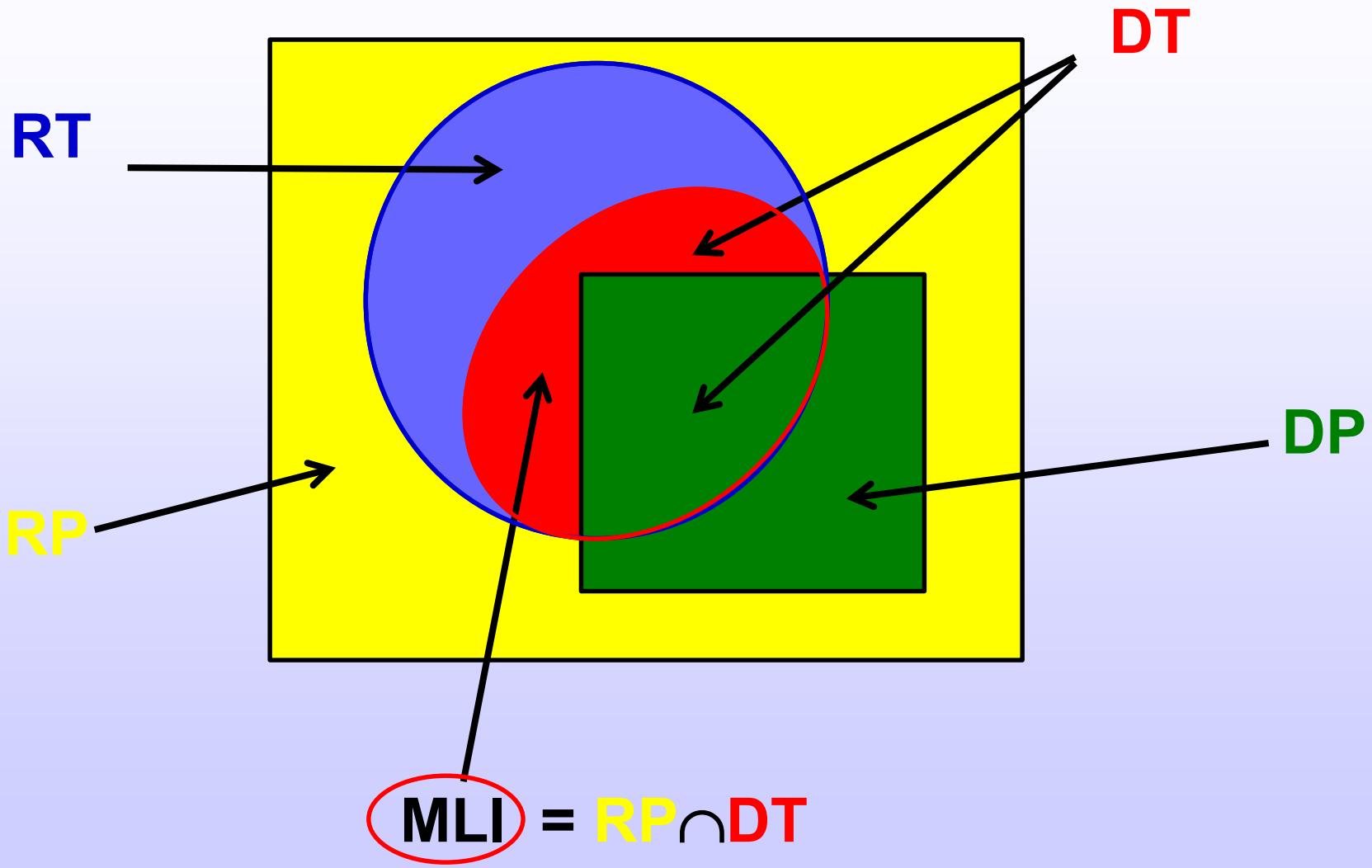
DP∩DT

Removing MLI enhances the classification /decision accuracy and robustness

However, Can this be achieved by the criteria of PCA or LDA?

How to find the unreliable or harmful or misleading dimensions?

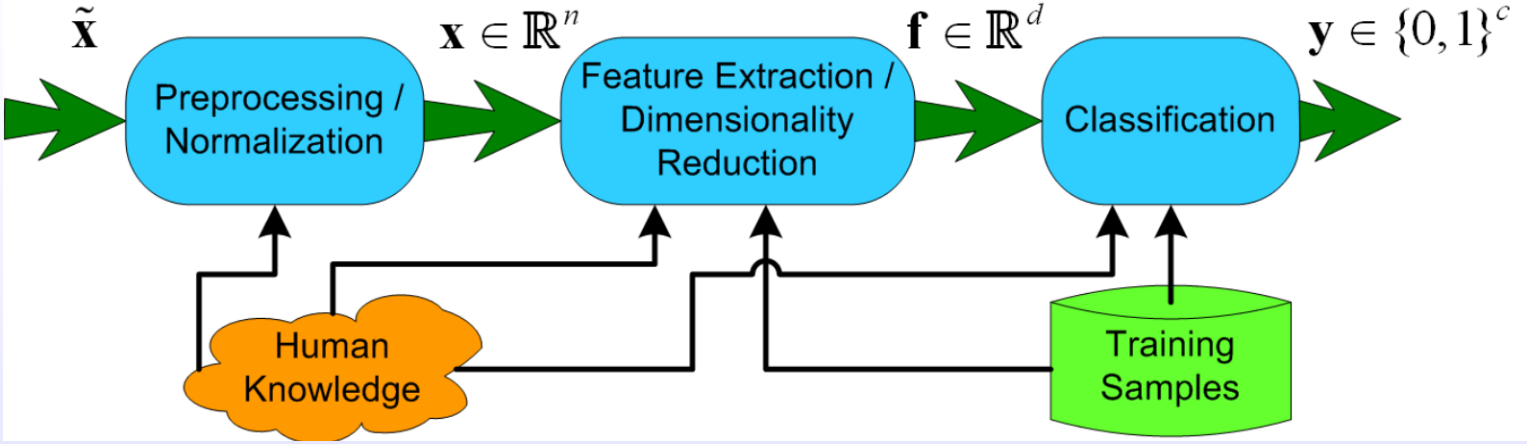
Information set of DP, RP, DT, RT & MLI



12 FE/DR v. ML –Objectives of FE/DR for PR

- Some general phenomena are well known in the pattern recognition community, such as the curse of dimensionality, small sample size problem, noise removal effect of dimensionality reduction and better generalization in a lower dimensional space. However, these have not indicated what dimensions should be extracted or what else should be removed for a more robust classification. We cannot develop an effective dimensionality reduction technique to maximize the classification accuracy just based on these general phenomena.
- For example, it is now clear that extracting maximal discriminative information just means that minimal discriminative information is removed. It is straightforward that minimum loss does not mean any gain.
- Therefore, it is unlikely that the dimensionality reduction by minimizing the loss of discriminative information can boost the classification accuracy.

12 FE/DR v. ML –Objectives of FE/DR for PR



A classifier is trained to capture the most discriminative information on the training samples. If some statistics estimated on the training data deviate from those of the data population, misclassification rate on the novel data increases. This is always the case in practice. Question is only how severe it is.

Therefore, to boost the classification accuracy, the dimensionality reduction should be targeted at removing the dimensions unreliable or harmful for the classification.

12 FE/DR v. ML –Problems of classification in high dimension

- We have learned that if data of all classes obey Gaussian distribution, the Bayes optimal decision becomes to evaluate:

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

The first part is Mahalanobis distance between X and M_i, where b_i is a threshold for user to control the error rate of a class at a price of other classes.

- Problem is that human knowledge cannot provide the class mean and covariance matrix of the data population, which can only be estimated or learned by machine from the available training samples. If some estimates largely deviate from those of the data population, we will face large misclassification rate. The discriminant function is very sensitive to the covariance matrix because the data vector is multiplied by its inverse.

12 FE/DR v. ML –Problems of classification in high dimension

- However, the inverse of the *n* by *n* covariance matrix is a great burden for us to clearly see the problems and find solutions. It is very difficult to study the problems of the covariance matrix directly as it carries two different kinds of information by n^2 estimates: data variations and correlations, which could have millions of parameters estimated by a limited number of training samples.

- Eigen-decomposition provides an effective tool to simplify the problem. It provides us a powerful tools to study, analyze and find problems of the high dimensional matrix. As the covariance matrix is symmetric, its eigenvectors provide an orthogonal basis for *n*-space.

12 FE/DR v. ML –Problems of classification in high dimension

Recall that from the basic definition of eigen-decomposition:

$$\boldsymbol{\Sigma}\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{\Lambda} \qquad \boldsymbol{\Phi}^T\boldsymbol{\Phi} = \mathbf{I} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T \qquad \boldsymbol{\Phi}^{-1} = \boldsymbol{\Phi}^T$$

$$\boldsymbol{\Phi}^T\boldsymbol{\Sigma}\boldsymbol{\Phi} = \boldsymbol{\Lambda}, \text{ or } \boldsymbol{\Sigma} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$$

- We have

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

$$= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\Phi}_i\boldsymbol{\Lambda}_i\boldsymbol{\Phi}_i^T)^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

$$= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\Phi}_i^T)^{-1}\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Phi}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_{ii}) + b_i$$

$$= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Phi}_i\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Phi}_i^T(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

$$= -\tfrac{1}{2}[\boldsymbol{\Phi}_i^T(\mathbf{x}-\boldsymbol{\mu}_i)]^T\boldsymbol{\Lambda}_i^{-1}[\boldsymbol{\Phi}_i^T(\mathbf{x}-\boldsymbol{\mu}_i)] + b_i$$

$$= -\tfrac{1}{2}\sum_{k=1}^{n}\frac{(z_k-\bar{z}_k)^2}{\lambda_k} + b_i$$

For symbolic simplicity, the index i is omitted where k is necessary. mp

$$z_k = \boldsymbol{\phi}_k^T\mathbf{x}, \quad \bar{z}_k = \boldsymbol{\phi}_k^T\boldsymbol{\mu}_i,$$

$$\boldsymbol{\Phi}_i = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n]$$

12 FE/DR v. ML –Problems of classification in high dimension

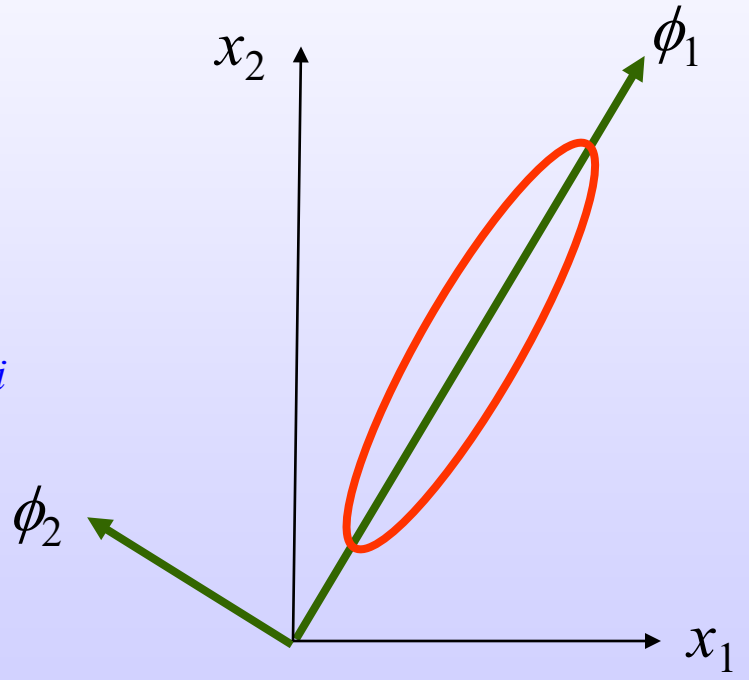
- In the eigen-space spanned by ϕ_k, we can handle the complex inverse of the *n* by *n* covariance matrix in a convenient scalar form!

$$\mathbf{z} = \{z_k\}_1^n = \mathbf{\Phi}_i{}^T \mathbf{x} = \{\phi_k\}_1^n{}^T \mathbf{x}$$

$$\overline{z}_k = \phi_k{}^T \boldsymbol{\mu}_i$$

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i{}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

$$= -\tfrac{1}{2}\sum_{k=1}^{n} \frac{(z_k - \overline{z}_k)^2}{\lambda_k} + b_i$$



- Problem is now evident if some eigenvalues are unreliable, especially if some eigenvalues are small and approach to zero.

12 FE/DR v. ML –Problems of Small Eigenvalues

Problems of the dimensions corresponding to the unreliable small eigenvalues.

$$y = x / \hat{a} = x / (a + \varepsilon)$$

$$a + \varepsilon = 100 + 0.1$$

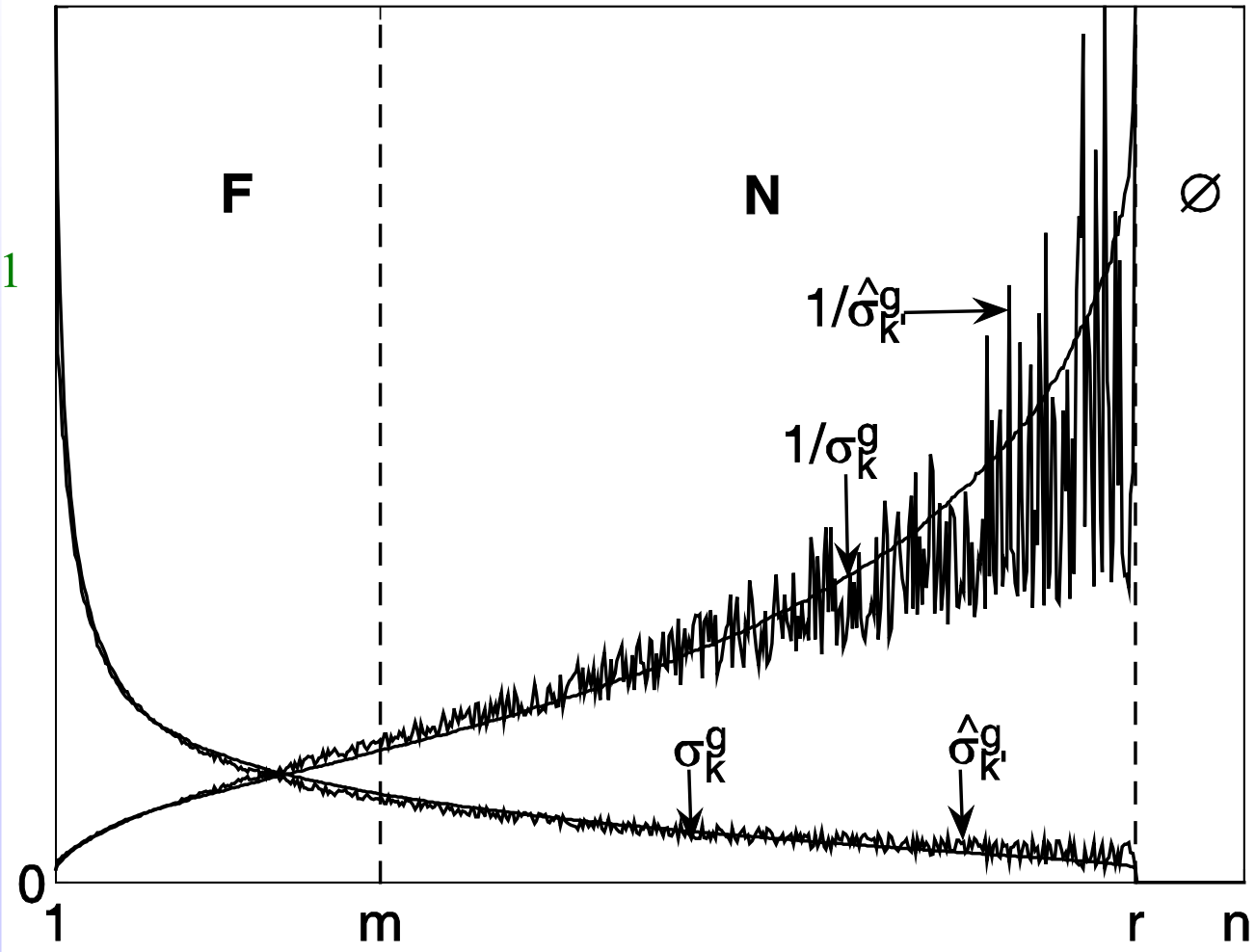
$$= 100.1 \approx 100$$

$$1 / (a + \varepsilon) = 0.00999 \approx 0.01$$

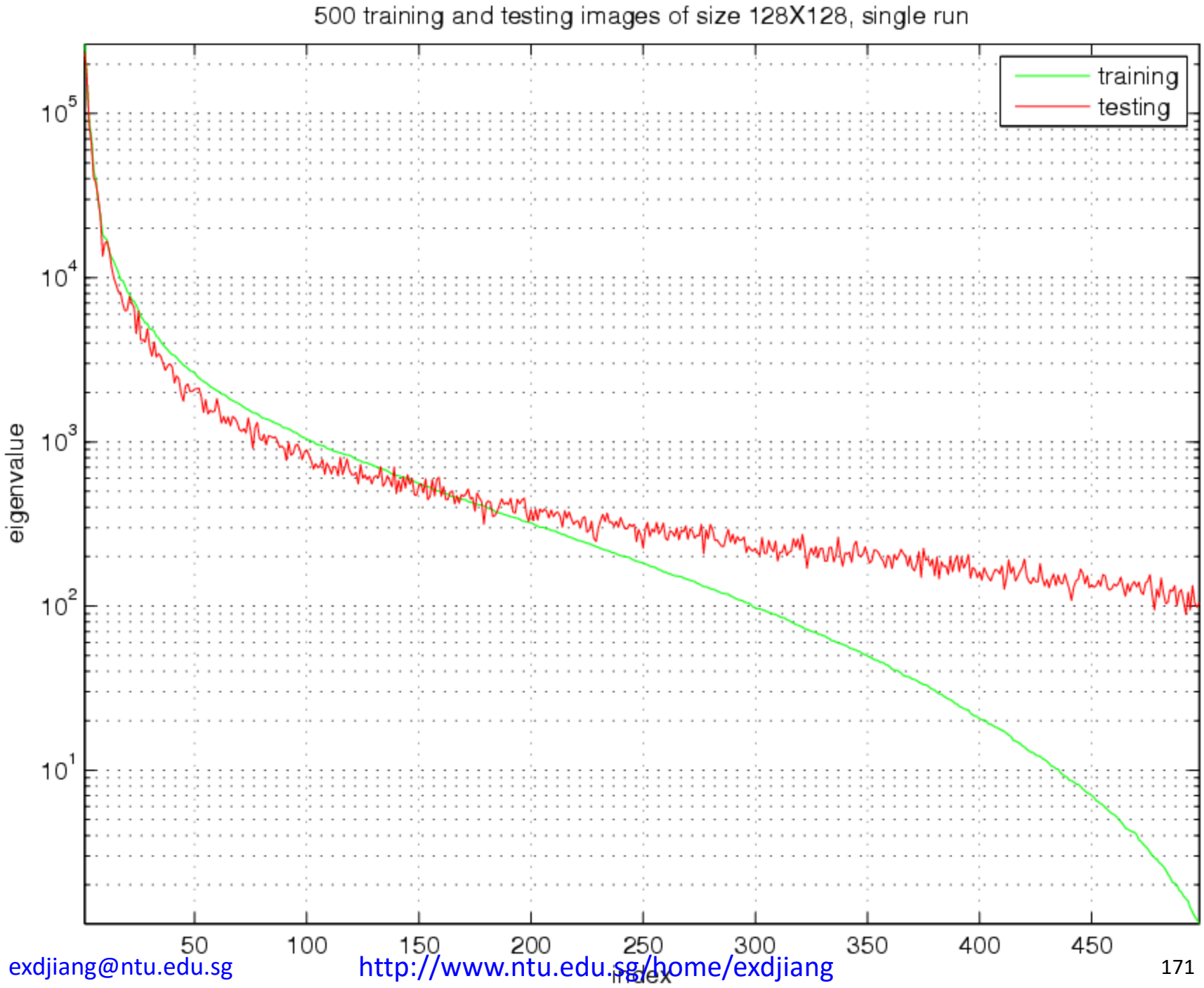
$$a + \varepsilon = 0.01 + 0.1$$

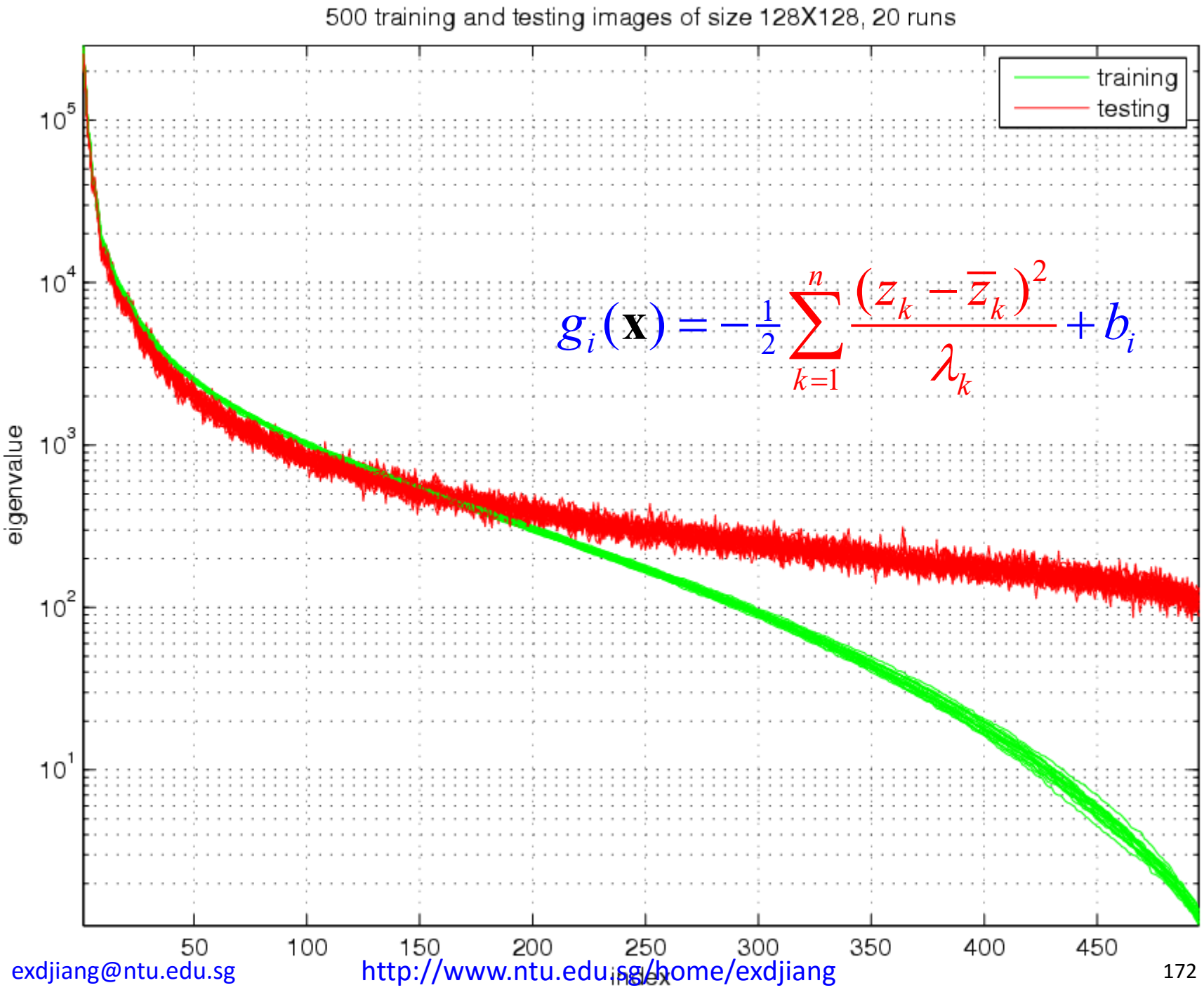
$$= 0.11 \neq 0.01$$

$$1 / (a + \varepsilon) = 9.09 \neq 100$$



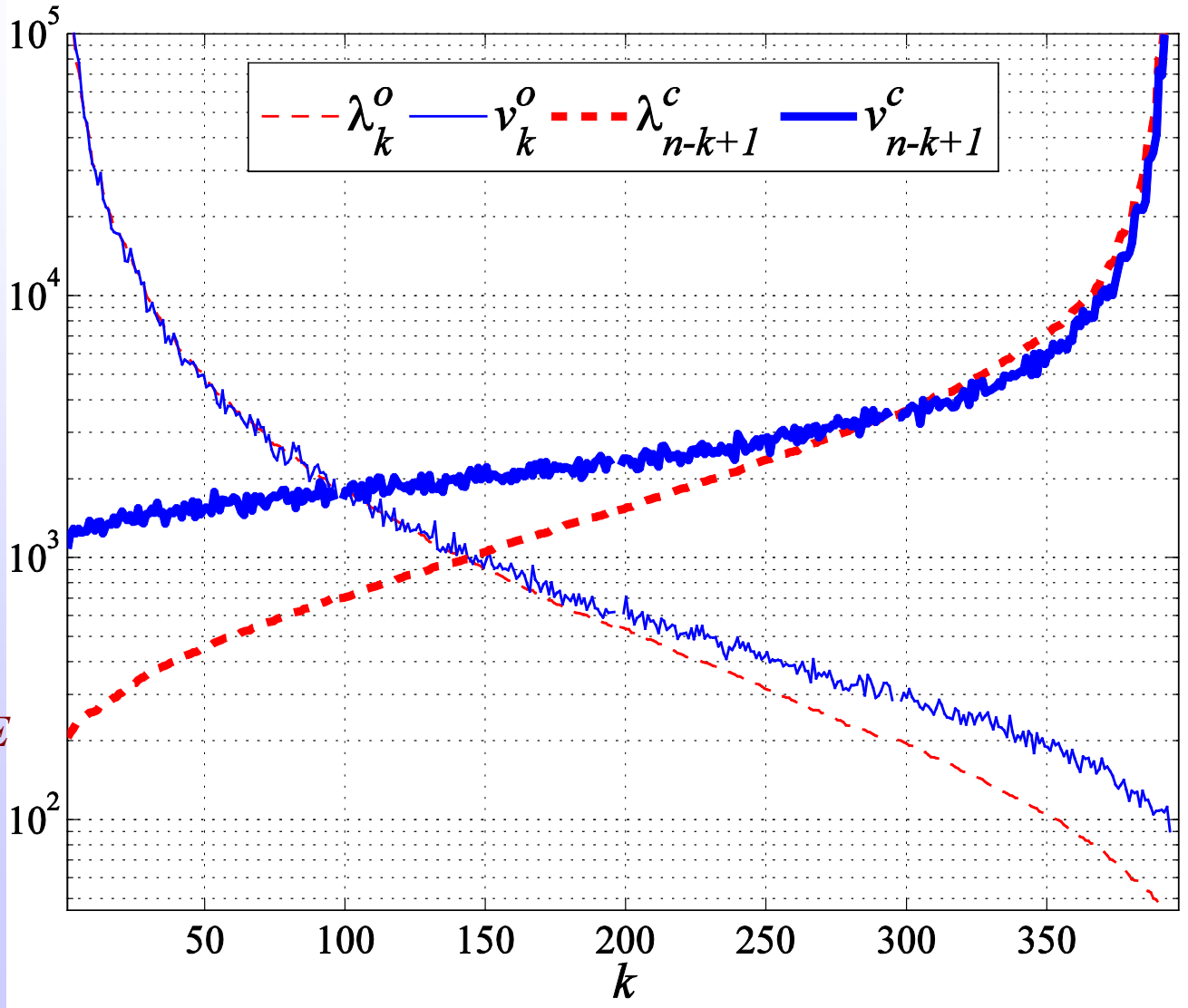
X.D. Jiang, B. Mandal and A. Kot, "Eigenfeature Regularization and Extraction in Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 383-394, March 2008.


500 training and testing images of size 128X128, single run
training
testing
eigenvalue
index
10^5
10^4
10^3
10^2
10^1
50
100
150
200
250
300
350
400
450


500 training and testing images of size 128X128, 20 runs
training
testing
eigenvalue
index
10^5
10^4
10^3
10^2
10^1
50
100
150
200
250
300
350
400
450
$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{n}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} + b_i$

12 FE/DR v. ML –Problems of Small Eigenvalues

Problems of the dimensions corresponding to the unreliable small eigenvalues.



X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

12 FE/DR v. ML –Solution1: adding a small constant

- One solution is to regularize the covariance matrix. A common practice in classification and data regression is to add a constant to its diagonal elements.

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\Sigma}_i + a\mathbf{I})^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

- Although this method was originally proposed to circumvent the singularity of $\boldsymbol{\Sigma}_i$ and the numerical instability of its inverse, numerous algorithms for classification, data regression, dimensionality reduction and manifold learning adopt this classical technique.
- Why this technique can improve the classification accuracy?
- The underlying principle can been seen by its equivalence to adding the constant to all eigenvalues

$$(\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\Sigma}_i + a\mathbf{I})^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) = \sum_{k=1}^{n}\frac{1}{\lambda_k + a}(z_k - \overline{z}_k)^2$$

12 FE/DR v. ML –Solution1: adding a small constant

- This is easy to see from $trace(\boldsymbol{\Sigma}_i + a\mathbf{I}) = \sum_{k=1}^{n}(\lambda_k + a)$
- From $\frac{(\lambda_k + a)}{v_k} = (1 + \frac{a}{\lambda_k})\frac{\lambda_k}{v_k}$

  we see that the factor $(1 + \frac{a}{\lambda_k})$ is larger for smaller λ_k and smaller for larger λ_k.
- Therefore, the regularized eigen-spectrum λ^a_k can be very close to the population variances as shown in the Figures latter.
- Problems of this method are the increased disparity of large eigenvalues and no dimensionality reduction effect. Either the *nxn* covariance matrix or the *nxn* eigenvector matrix is needed to compute the discriminant function.

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T(\boldsymbol{\Sigma}_i + a\mathbf{I})^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + b_i = \sum_{k=1}^{n}\frac{-1/2}{\lambda_k + a}(z_k - \overline{z}_k)^2 + b_i$$

12 FE/DR v. ML –Solution2: probabilistic subspace learning

Another solution, called probabilistic subspace learning decomposes the discriminant function into two parts and replaces the small eigenvalues by a constant

$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{n}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} + b_i \quad \Downarrow$$

$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{m}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} - \frac{1}{2}\sum_{k=m+1}^{n}\frac{(z_k - \bar{z}_k)^2}{\rho_{av}} + b_i$$

The constant is computed by $$\rho_{av} = \frac{1}{n-m}\sum_{k=m+1}^{n}\lambda_k$$

as it is the optimal approximation to λ_k *for* $m < k < n$.

This method leads to one of the best performers called Bayesian algorithm in the face recognition community

12 FE/DR v. ML –Solution3: enhanced probab. subspace learning

However, the purpose of regularization is not best approximating to the eigenspectrum but to the population variances.

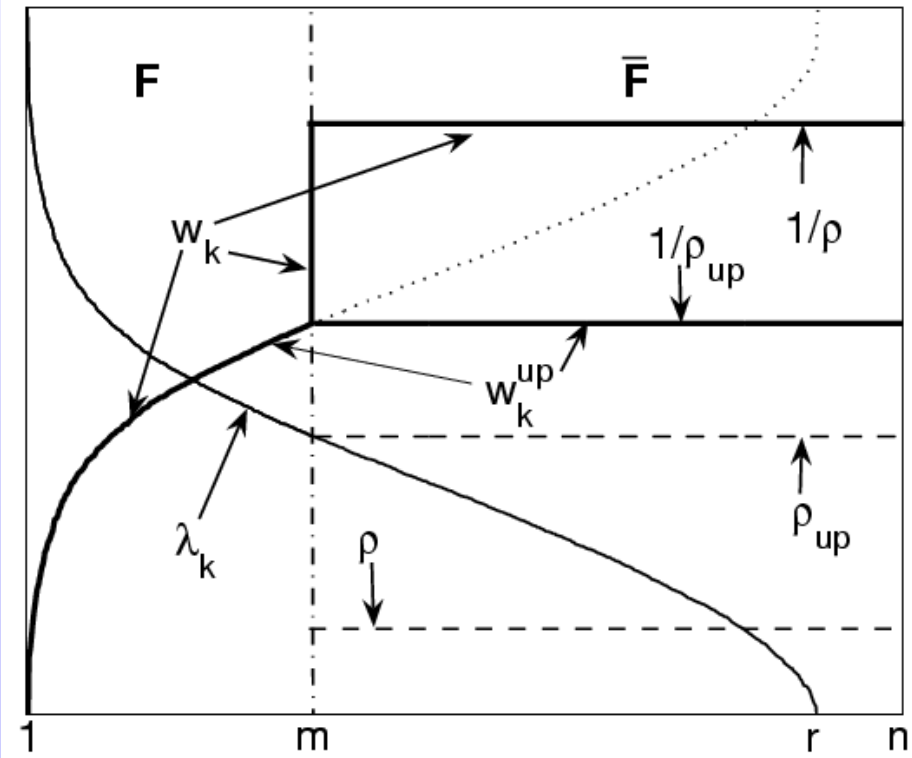
$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{m}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} - \frac{1}{2}\sum_{k=m+1}^{n}\frac{(z_k - \bar{z}_k)^2}{\rho_{up}} + b_i$$

$$\rho_{up} = \lambda_{m+1}$$

$$= \max(\lambda_k,\ k = m+1,\ldots,n)$$

X.D. Jiang, B. Mandal and A. Kot, Enhanced Maximum Likelihood Face Recognition, *Electronics Letters*, vol. 42, no. 19, pp. 1089-1090, September 2006.

recognition rate	number of training images		
method	500	1000	1400
ρ_{av}	89.76	90.44	89.70
ρ_{up}	91.03	93.03	93.64
$\rho_{up}\,\mathbb{R}^r$	92.90	95.94	96.46



12 FE/DR v. ML –Solution2 & 3: probab. subspace learning

These two regularization techniques have some role of dimensionality reduction as it is not necessary to project the data to the eigenvectors for $k > m$. Euclidian distance between two vectors in the eigen-space is identical to that in the data space. Thus, we only need nxm eigenvector matrix for classification. However, the n-dimensional class mean vectors are still required. I call it semi-dimensionality reduction.

$$\because (\mathbf{x}-\boldsymbol{\mu}_i)^T(\mathbf{x}-\boldsymbol{\mu}_i) = \sum_{k=1}^{m}(z_k-\overline{z}_k)^2 + \sum_{k=m+1}^{n}(z_k-\overline{z}_k)^2$$

$$\therefore \sum_{k=m+1}^{n}\frac{(z_k-\overline{z}_k)^2}{\rho_{up}} = \frac{1}{\rho_{up}}\left[(\mathbf{x}-\boldsymbol{\mu}_i)^T(\mathbf{x}-\boldsymbol{\mu}_i) - \sum_{k=1}^{m}(z_k-\overline{z}_k)^2\right]$$

12 FE/DR v. ML –Solution4: Dimensionality Reduction

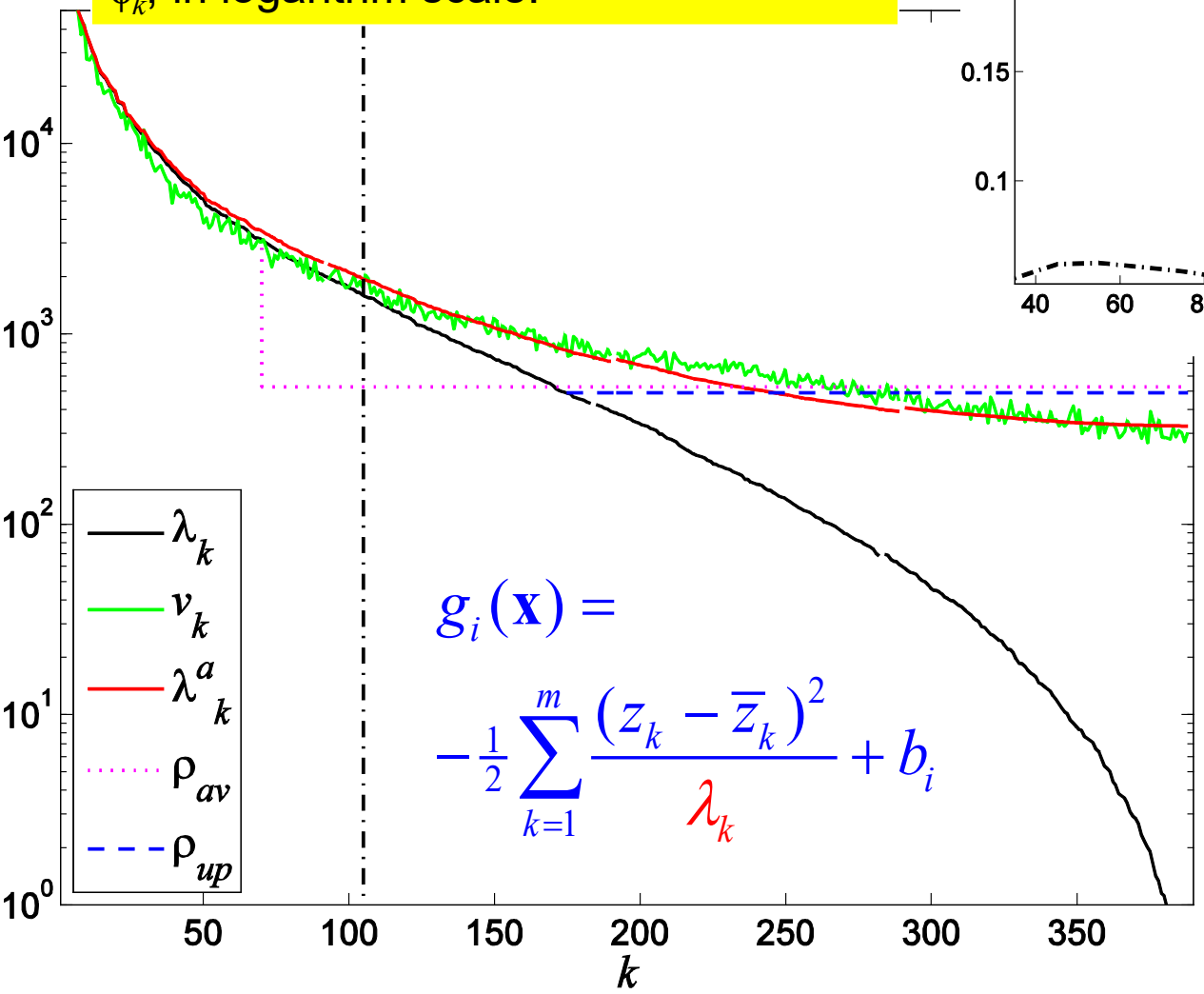
$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{m}\frac{(z_k - \overline{z}_k)^2}{\lambda_k} - \frac{1}{2}\sum_{k=m+1}^{n}\frac{(z_k - \overline{z}_k)^2}{\rho} + b_i$$

- Why PCA, though an unsupervised method that minimizes the reconstruction error rather than maximizes the discrimination of classes, can improve the classification accuracy?

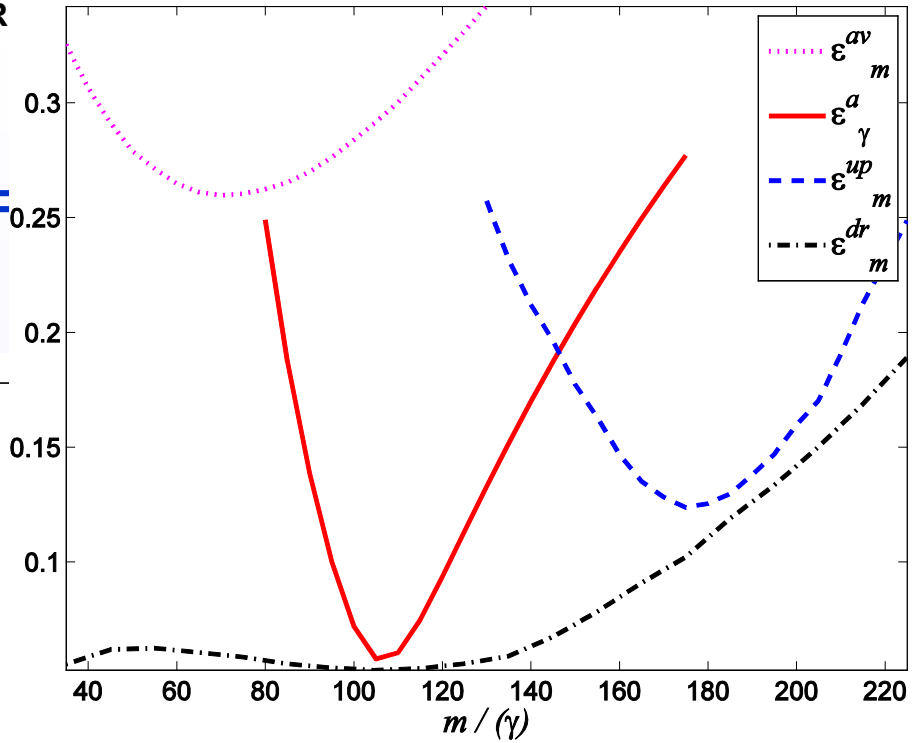
The underline principle behind PCA for classification is to remove dimensions corresponding to the unreliable small eigenvalues of the class-conditional covariance matrices.

$$\mathbf{S}^t = \mathbf{S}^w + \mathbf{S}^b = \sum_{j=1}^{c}\frac{L_j}{L}\boldsymbol{\Sigma}_j + \mathbf{S}^b$$

The black curve shows an eigen-spectrum obtained from 400 face images and the green curve shows the variances v_k of other 8,500 face images projected on the eigenvectors ϕ_k, in logarithm scale.



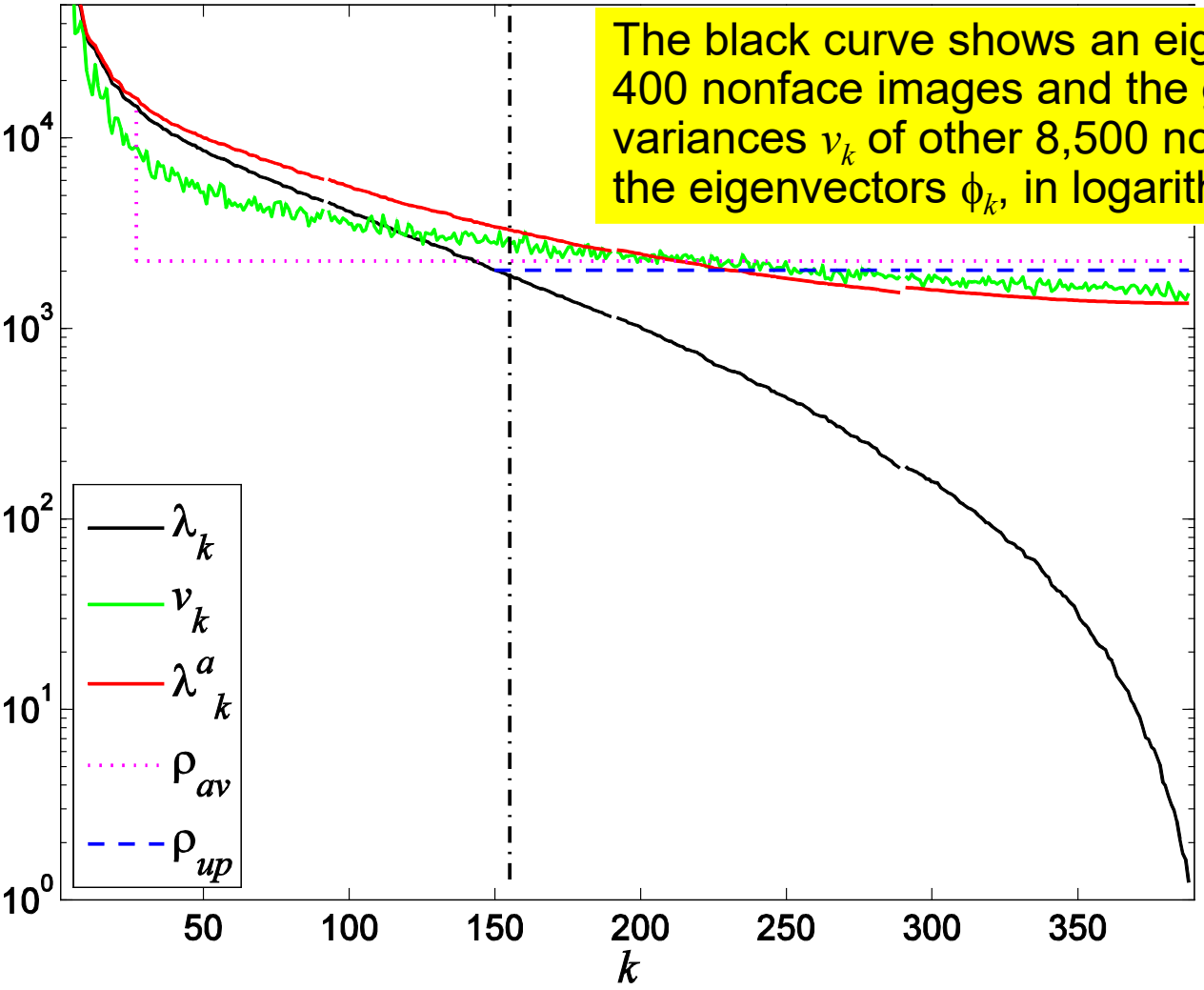
$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{m}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} + b_i$$



the normalized disparity between the regularized eigen-spectrum and the variances

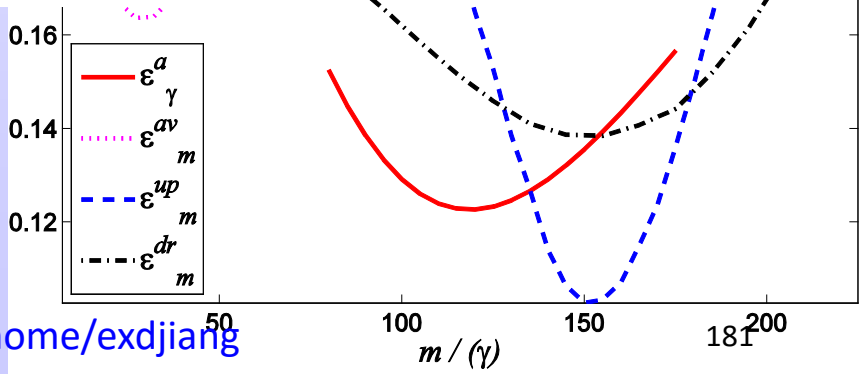
X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

The black curve shows an eigen-spectrum obtained from 400 nonface images and the green curve shows the variances v_k of other 8,500 nonface images projected on the eigenvectors ϕ_k, in logarithm scale.



$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{k=1}^{m}\frac{(z_k - \bar{z}_k)^2}{\lambda_k} + b_i$$

the normalized disparity between the regularized eigen-spectrum and the variances



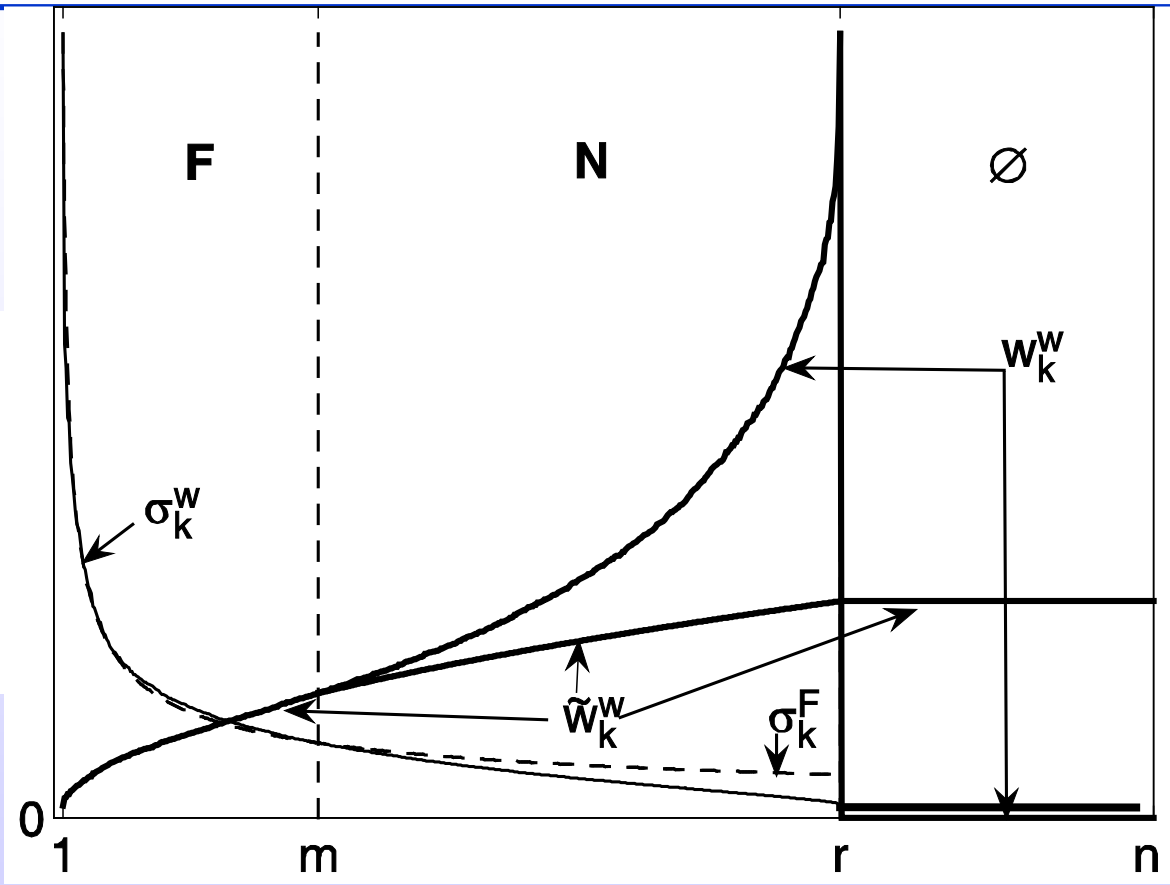
X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

exdjiang@ntu.edu.sg http://www.ntu.edu.sg/home/exdjiang

12 FE/DR v. ML –Solution5: Eigenspectrum Model

Eigenspectrum model

$$\tilde{\lambda}_k^w = \begin{cases} \lambda_k^w, & k < m \\ \frac{\alpha}{k+\beta}, & m \le k \le r \\ \frac{\alpha}{r+1+\beta}, & r < k \le n \end{cases}$$



Was proposed in

X.D. Jiang, B. Mandal and A. Kot, "Eigenfeature Regularization and Extraction in Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 383-394, March 2008.

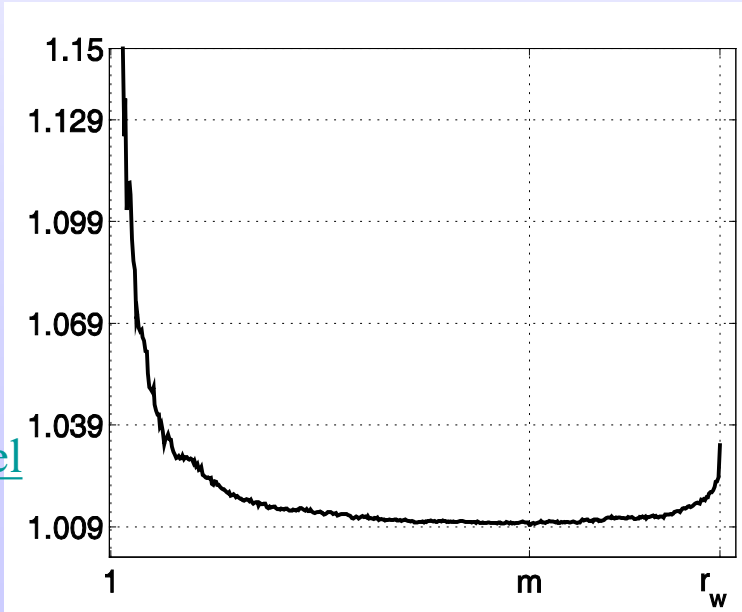
12 FE/DR v. ML –Solution5: Eigenspectrum Model

Determine the unreliable eigenvalues in ERE method by

$$\lambda_k^w < \lambda_{r/2}^w + \lambda_{r/2}^w - \lambda_r^w = 2\lambda_{r/2}^w - \lambda_r^w$$

Determine the unreliable eigenvalues by the minimum eigen-ratio:

$$m = \arg\{\min_k(\lambda_k^w / \lambda_{k+1}^w)\}$$

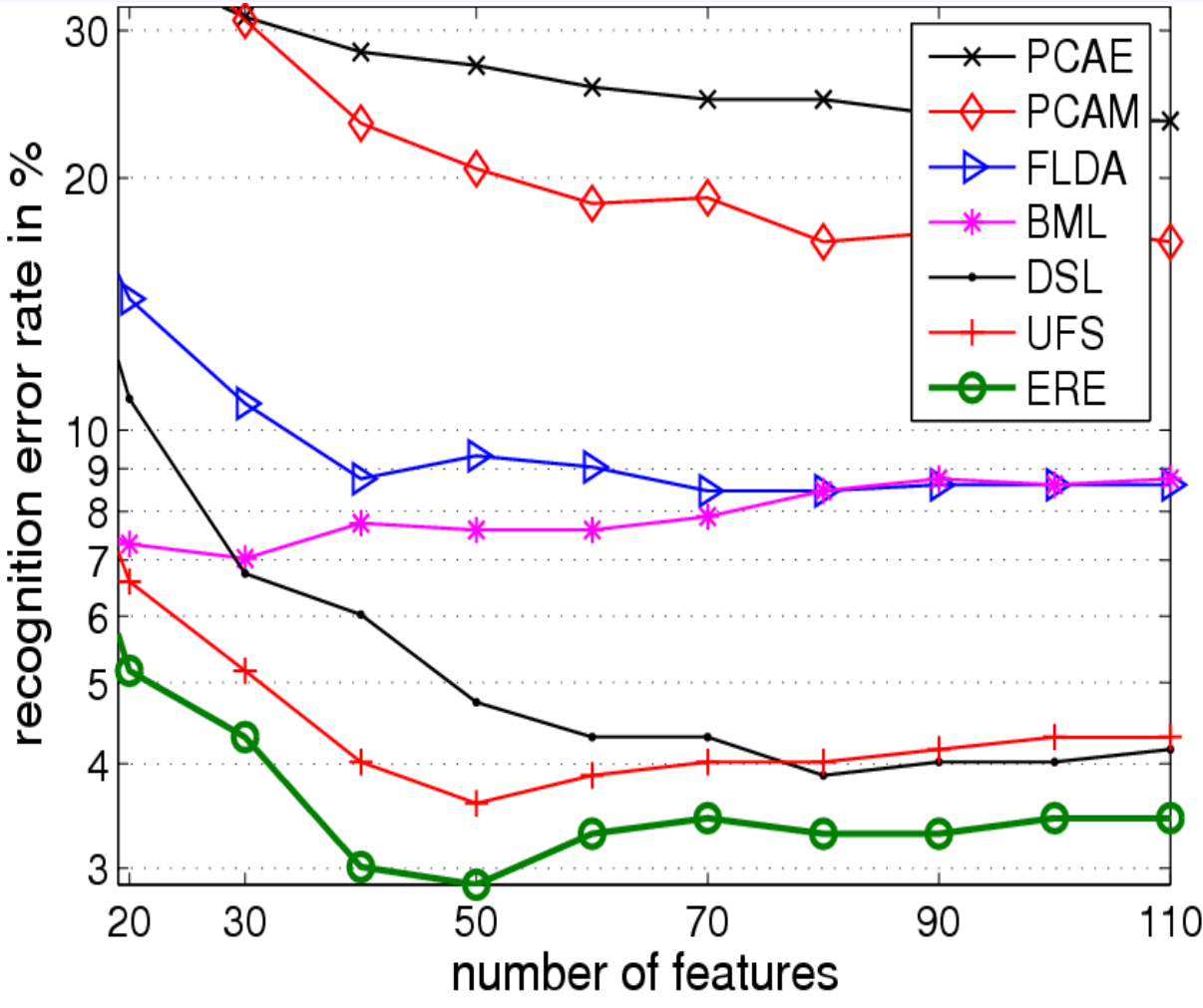


X.D. Jiang, B. Mandal and A. Kot, "Complete Discriminant Evaluation and Feature Extraction in Kernel Space for Face Recognition," *Machine Vision and Applications*, vol. 20, no. 1, pp. 35-46, January 2009.

12 FE/DR v. ML –Solution5: Eigenspectrum Model

Face recognition performance comparison of different approaches

on the FERET database of 994 training images (497 subjects) and 1394 testing images (697 subjects).



12 FE/DR v. ML –Solution6: Supervised/Asymmetric PCA

- Is the unsupervised PCA that minimizes the data reconstruction error optimal in the dimensionality reduction for classification?

$$\mathbf{S}^t = \frac{1}{q}\sum_{i=1}^{L}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T$$

$$= \mathbf{S}^w + \mathbf{S}^b = \sum_{j=1}^{c}\frac{q_j}{q}\boldsymbol{\Sigma}_j + \mathbf{S}^b$$

- What is the problem of PCA for classification?

  Classes with more training samples are heavier weighted so more dimensions unreliable for theses classes are removed.
- How to modify PCA from an unsupervised method to a supervised one for an even better classification?
- How to modify PCA to handle unbalanced data and asymmetric classes?

12 FE/DR v. ML –Solution6: Supervised/Asymmetric PCA

PCA

$$\mathbf{S}^t = \frac{1}{q}\sum_{i=1}^{L}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T$$

$$= \mathbf{S}^w + \mathbf{S}^b = \sum_{j=1}^{c}\frac{q_j}{q}\boldsymbol{\Sigma}_j + \mathbf{S}^b$$

•It is not the more but the less reliable covariance matrix that should be heavier weighted in the covariance mixture so that more dimensions characterized by the small variances of this class can be removed.

New methods were proposed in

X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

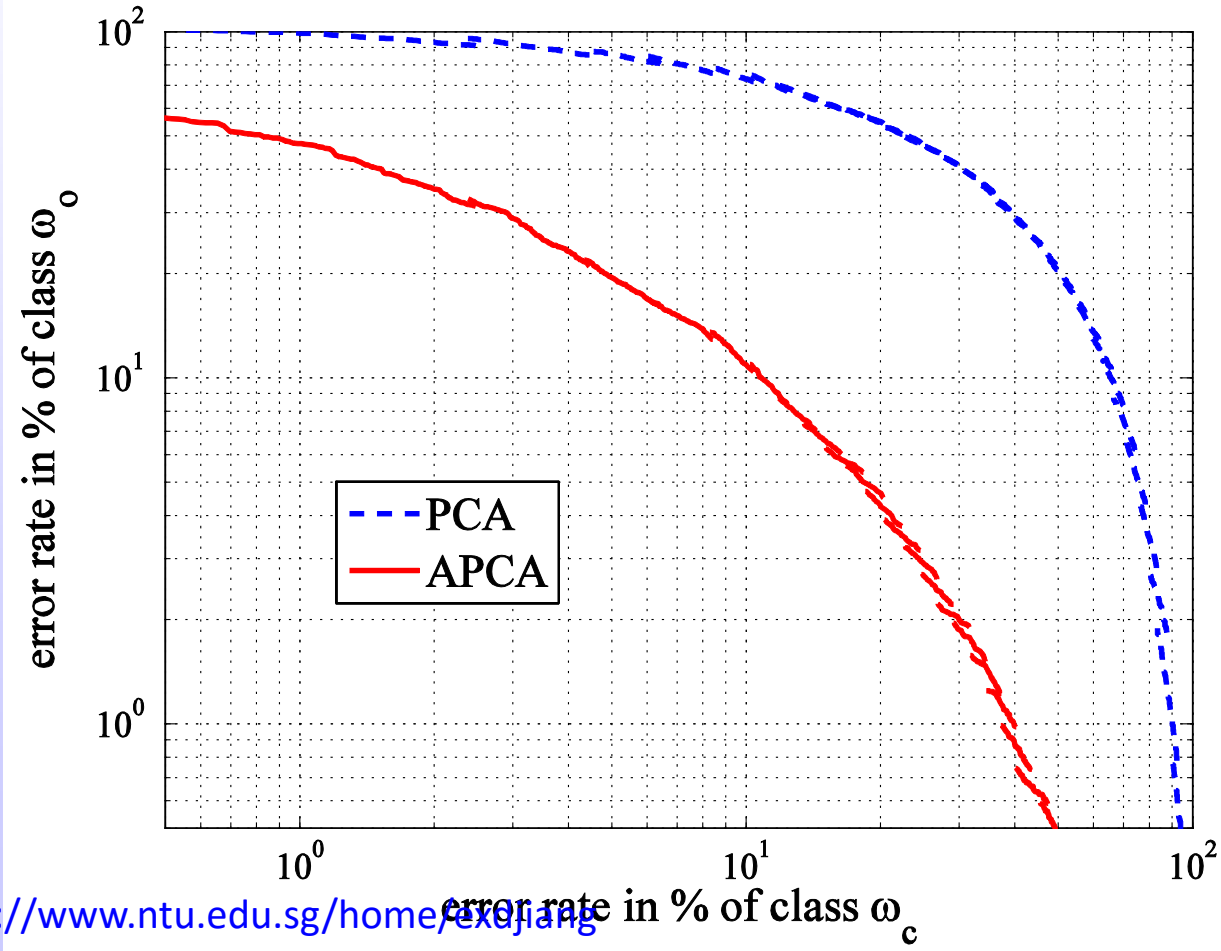
Asymmetric PCA: APCA $\boldsymbol{\Sigma}_a = a\boldsymbol{\Sigma}_o + \beta\boldsymbol{\Sigma}_c + \eta\mathbf{S}^b$

Supervised PCA: SPCA $\boldsymbol{\Sigma}_a = \sum_{j=1}^{c} a_j\boldsymbol{\Sigma}_j + \eta\mathbf{S}^b$

12 FE/DR v. ML –Solution6: Supervised/Asymmetric PCA

Classification performance comparison of different approaches

2,000/500 training samples and 20,000/5,000 testing samples of the 400 dimensional data of classes ω_o and ω_c. m=200, a=0.2



X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

12 FE/DR v. ML –Summary: DR for Accuracy Enhancement

- It is the regularization technique or the dimensionality reduction by the supervised/asymmetric principal component analysis (SPCA, APCA) that plays the most vital role in boosting the classification accuracy, the most important objective of the dimensionality reduction.
- The underlying principle behind the regularization is that it corrects the unreliable statistics that are harmful for classification. Similarly, SPCA/APCA removes dimensions that are harmful for the classification.
- However, these techniques may not be able to greatly reduce the dimension for a fast classification.
- The discriminative method can greatly reduce the dimension with the minimum loss of the discriminative information, i.e. it removes redundant dimensions. It can be used for reducing the classification complexity, another objective of the dimensionality reduction.
- However, the vast majority of researchers so far prefer discriminative approaches and the most published approaches stem from the most discriminative criterion. To make the above conclusions more convincing, let's restudy the discriminative method.

12 FE/DR v. ML –Restudy the Insights of LDA

- In the identification applications, we often have a large number of classes with only a few samples per class for training so that each individual class covariance matrix is extremely unreliable. One solution to regularize them is to pool them together to form a common covariance matrix.

$$\boldsymbol{\Sigma}_w = \sum_{j=1}^{c} \frac{L_j}{L} \boldsymbol{\Sigma}_j = \mathbf{S}^w$$

- Recall that under Gaussian assumption with the same covariance matrix of all classes, the Bayes optimal decision becomes to evaluate the discriminant function, which becomes a linear function of $\mathbf{x}$:

$$g_i(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + b_i$$

$$\Rightarrow \quad g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{S}^{w-1}\boldsymbol{\mu}_i + t_i = \mathbf{x}^T \mathbf{w}_i + t_i$$

where t_i absorbs all terms that is either constant to $\mathbf{x}$ or constant to i.

12 FE/DR v. ML –Restudy the Insights of LDA

- The decision boundary in the $\mathbf{x}$-space between any two classes ω_i and ω_j

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \quad \Rightarrow \quad \mathbf{x}^T \mathbf{S}^{w-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + t_i - t_j = 0$$

is a hyperplane specified by its normal vector

$$\psi_{ij} = \mathbf{S}^{w-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

and the threshold t_i - t_j.

- This means that for the optimal classification between the two classes ω_i and ω_j, only one dimension spanned by ψ_{ij} is necessary. Thus, under the constraint of the linear classification, this dimension ψ_{ij} contains the most (in fact, all) discriminative information to differentiate class ω_i and ω_j.

12 FE/DR v. ML –Restudy the Insights of LDA

- Although we need c(*c*-1)/2 hyperplanes to classify *c* classes, their normal vectors

$$\psi_{ij} = \mathbf{S}^{w-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

only span a (*c*-1)-dimensional subspace as only (*c*-1) of them are linear independent. Therefore, we can reduce the *n*-dimensional data space to this (*c*-1)-dimensional subspace without losing any discriminative information as the linear classification

$$g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{S}^{w-1} \boldsymbol{\mu}_i + t_i$$

produces exactly the same results in the two spaces.

- This (*c*-1)-dimensional subspace is spanned by

$$\psi_i = \mathbf{S}^{w-1} \boldsymbol{\mu}_i$$

- Same as a training sample $\mathbf{x}_i$, ψ_i is also a point in the *n*-dimensional $\mathbf{x}$-space.

12 FE/DR v. ML –Restudy the Insights of LDA

- However, if the dimensionality is reduced to *d*, *d*<*c*-1, some discriminative information will be lost. Based on what we have learned from PCA, the minimal information is lost if we keep the subspace spanned by the eigenvectors corresponding to the *d* largest eigenvalues of the covariance or scatter matrix of the c samples ψ_i

$$\Sigma_\psi = \mathbf{S}^{w-1}\Sigma_\mu = \mathbf{S}^{w-1}\mathbf{S}^b$$

- This is the well-known linear discriminant analysis (LDA) that performs the eigen-decomposition

$$LDA: \quad \max trace\left[\mathbf{\Phi}^T \mathbf{S}^{w^{-1}} \mathbf{S}^b \mathbf{\Phi}\right] = \sum_{k=1}^{d} \lambda_k^{b/w}$$

- Here I claim that LDA is just PCA on samples ψ_i, which determine a linear classifier. The properties of LDA is exactly the same as those of PCA, just with different data.

12 FE/DR v. ML –Restudy the Insights of LDA

- We see from the above analysis that the objective of LDA is to find the one among all possible *d*-dimensional subspaces in which the linear classification

$$g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{S}^{w-1} \boldsymbol{\mu}_i + t_i$$

achieves the closest result to that in the original *n*-space. It is undoubtedly an effective method to largely reduce the data dimensionality with the minimum loss of the classification capability in a linear sense.

- For a two-category classification problem, LDA can only extract one dimension. It is insufficient for a reasonable classification for some problems such as various tasks of verification and object detection because the two class-conditional covariance matrices are significantly different and hence the optimal classification is obviously not linear.

12 FE/DR v. ML –Asymmetric Discriminant Analysis, ADA

- To apply the discriminant analysis in such problems, an asymmetric discriminant analysis (ADA) is proposed in

  X.D. Jiang, "Asymmetric Principal Component and Discriminant Analyses for Pattern Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

  to extract a rich number of features. It solves the following eigen-decomposition problem:

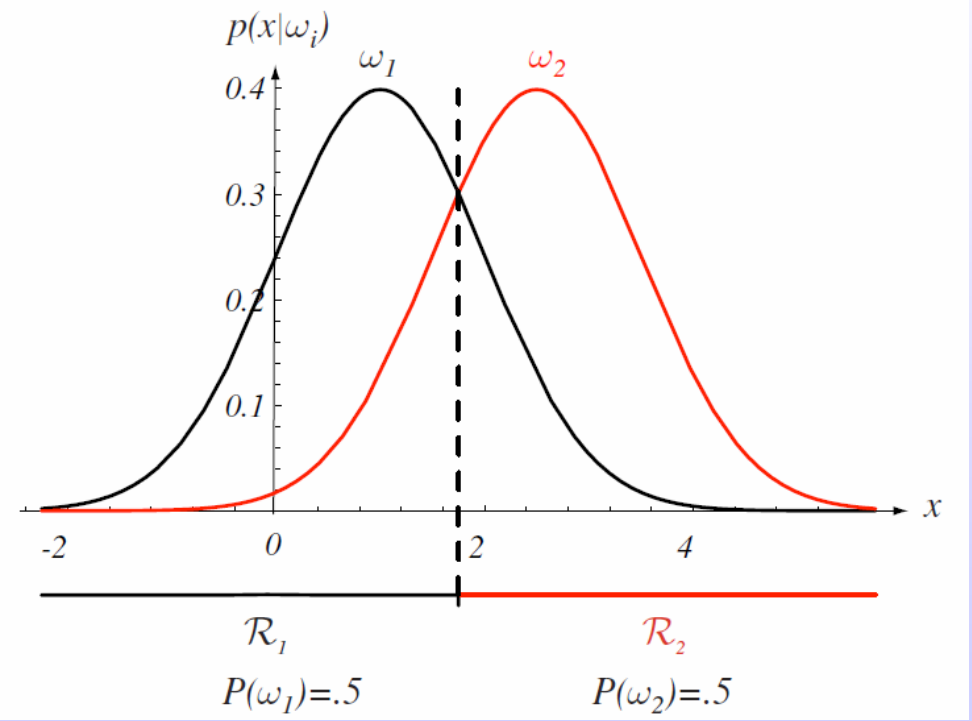
$$\mathbf{\Phi}^T(\mathbf{\Sigma}_o + \beta\mathbf{\Sigma}_c)^{-1}(\mathbf{\Sigma}_o + \gamma\mathbf{\Sigma}_\mu)\mathbf{\Phi} = \mathbf{\Lambda}$$

  The ADA with $\beta=\gamma=1$ maximizes the Bhattacharyya distance between two classes in the subspace spanned by the eigenvectors corresponding to the largest $\max(\lambda_k, 1-\lambda_k)$.

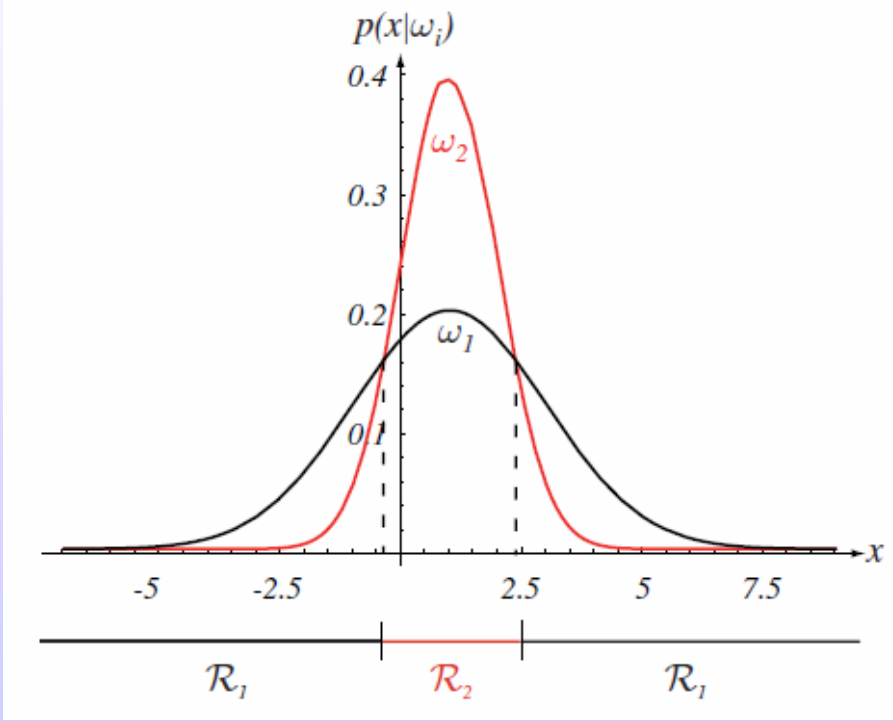
- The underlying principle of ADA is that the discriminative information is not only carried by the distinction of the two class means but also by the distinction of the two class variances.

12 FE/DR v. ML –Asymmetric Discriminant Analysis, ADA

- The underlying principle of ADA is that the discriminative information is not only carried by the distinction of the two class means but also by the distinction of the two class variances.



The discriminative information about class mean (LDA)



The discriminative information about class variance (ADA)

12 FE/DR v. ML –Comments and Conclusion of LDA

- The discriminative analysis extracts the most discriminative dimensions on the training data. it in principle just duplicates the classification process. Therefore, the most discriminative criterion may not help boost the classification accuracy and robustness.
- It is the regularization technique or the dimensionality reduction by the supervised principal component analysis that plays the most vital role in boosting the classification accuracy while the discriminative method can greatly reduce the dimensionality for a fast classification with the minimum loss of the accuracy.

- **To boost the classification accuracy, we need remove the harmful dimensions or modify the unreliable statistics of the training data in these dimensions.**
- **To speed up the classification, we need remove the redundant dimensions, i.e, extract the most discriminative dimensions.**

12 FE/DR v. ML –Comments and Conclusion of LDA

- Question may arise as to why NLDA can work well in some applications if the smallest and zero eigenvalues are the most unreliable. The reason behind it is that the classification of the NLDA features does not use the variance due to zero eigenvalues in all dimensions of the null space. Thus, it implicitly circumvents the problem of the unreliable small eigenvalues to a certain extent by evenly weighting all features.

- Another question is why some approaches using LDA alone can also work well on some data sets. The underlying causes include the avoidance of feature scaling by the variance in the classification and the linearity of LDA but the nonlinearity of the classifier.

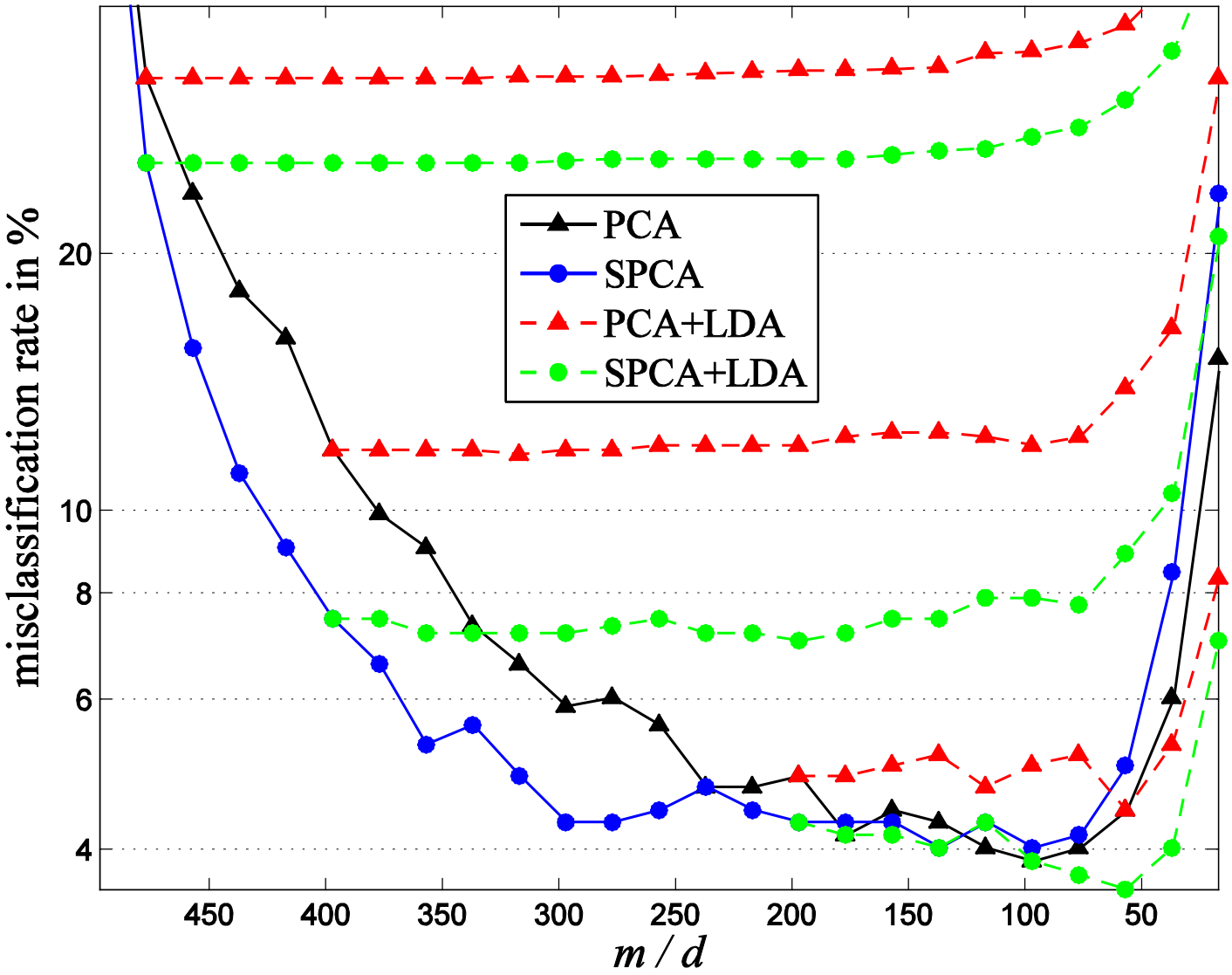
12 FE/DR v. ML –Comments and Conclusion of LDA

- These approaches, though applying LDA for feature extraction, do not apply its origin as classifier. Most of them apply the nearest-neighbor classifier (NNC) with Euclidian distance. While the simple Euclidian distance ignores the data variance and hence circumvents the problem of the unreliable small eigenvalues to a certain extent, the complex data distribution is captured by NNC that computes all distances from a novel pattern to all training samples. The NNC, though very simple, is highly nonlinear, can form arbitrary complex, nonlinear decision boundary and classifies all training samples without error. LDA restricts such highly nonlinear classifier to a subspace, which is, though the most discriminative, only in a linear sense. This restriction has similar role to the regularization.
- Therefore, **the improvement of the classification accuracy by LDA is most likely contributed by its linearity constraint rather than its most discriminative nature.**

Misclassification rate against the reduced dimensionality by PCA/SPCA and PCA/SPCA+LDA of face identification problems.

497 people are randomly selected for training and other 697 people are used for testing

Image size: 33X38.



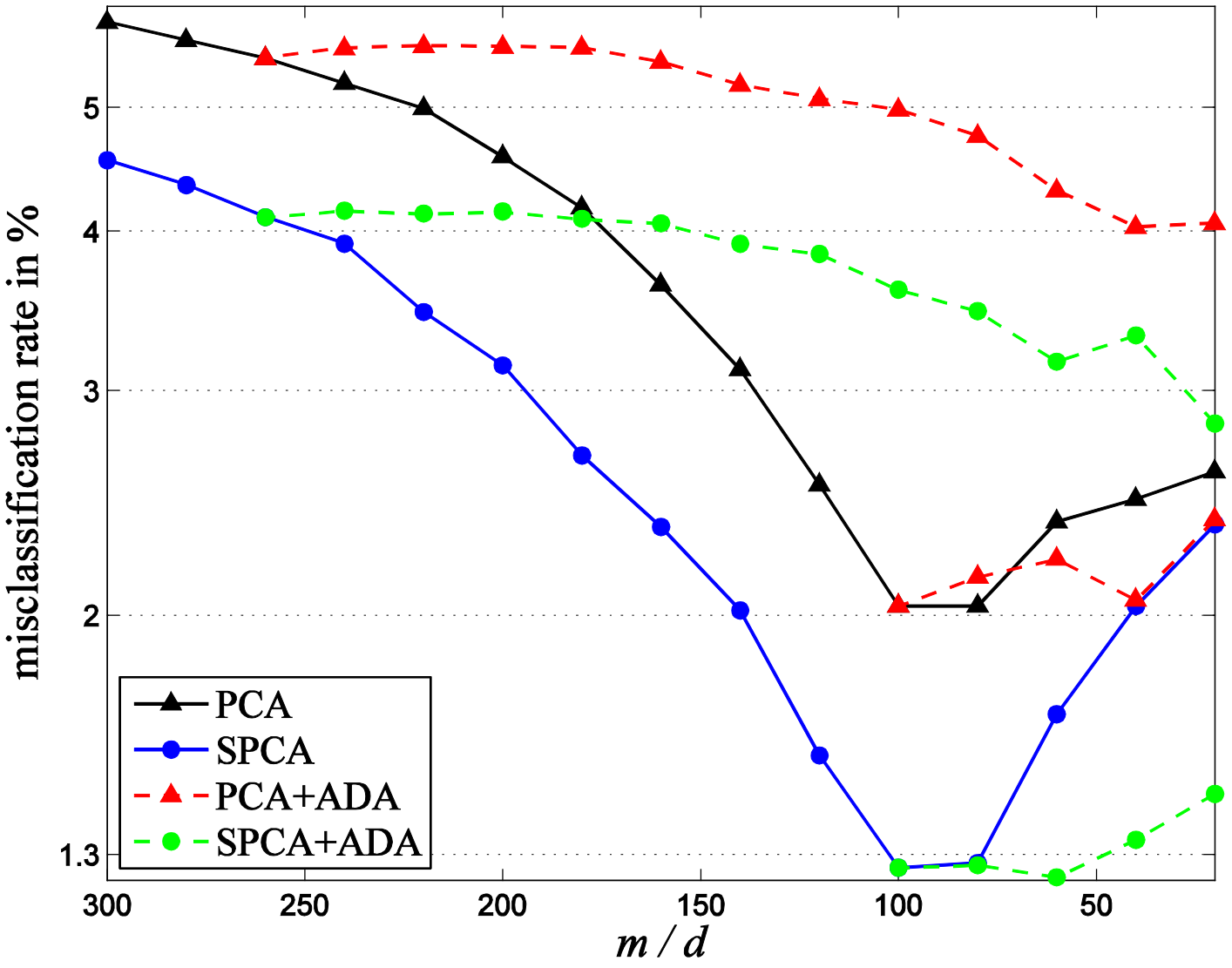
The most left point of each dashed curve indicates the dimensionality m of the PCA/SPCA subspace in which LDA/ADA further reduces it to d indicated by the other points on the same dashed curve.

X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

Misclassification rate against the reduced dimensionality by PCA/SPCA and PCA/SPCA+ADA of face detection problems.

9000 face images and 9000 non-face images are used in training (75%) and testing (25%).

Image size: 20X20



The most left point of each dashed curve indicates the dimensionality m of the PCA/SPCA subspace in which LDA/ADA further reduces it to d indicated by the other points on the same dashed curve.

X.D. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

12 Feature Extraction/Dimension Reduction v. Machine Learning

- Dimensionality reduction and feature extraction has two objectives. One is to reduce the computational complexity of the subsequent classification with the minimum loss of information needed for classification and the other is to circumvent the generalization (prediction) problem of the subsequent classification and hence enhance its accuracy and robustness.
- For machine learning from training samples based dimensionality reduction and feature extraction:
- There is no doubt that various discriminant analyses can effectively achieve the first objective.
- The second objective of the dimensionality reduction is much more important than the first one in most applications with the rapid growth of the computation power.
- However, the second objective of the dimensionality reduction is far from straightforward.

12 Feature Extraction/Dimension Reduction v. Machine Learning

- Although the ultimate objective of a pattern recognition system is to extract the most discriminative information, it is the most discriminative information about the whole data population, not on a specific training set. A classifier is trained to capture the most discriminative information on the training samples. If some statistics estimated on the training data deviate from those of the data population, misclassification rate on the novel data increases.
- The most discriminative criterion cannot solve this problem because it in general just repeats the classification process. Redundant information does not mean harmful for classification.
- Therefore, to boost the classification accuracy, the dimensionality reduction by machine learning should be targeted at removing the dimensions unreliable (harmful) for the classification or regularizing (correcting) the unreliable statistics in these dimensions.

13 Connectionist Approaches & Neural Networks

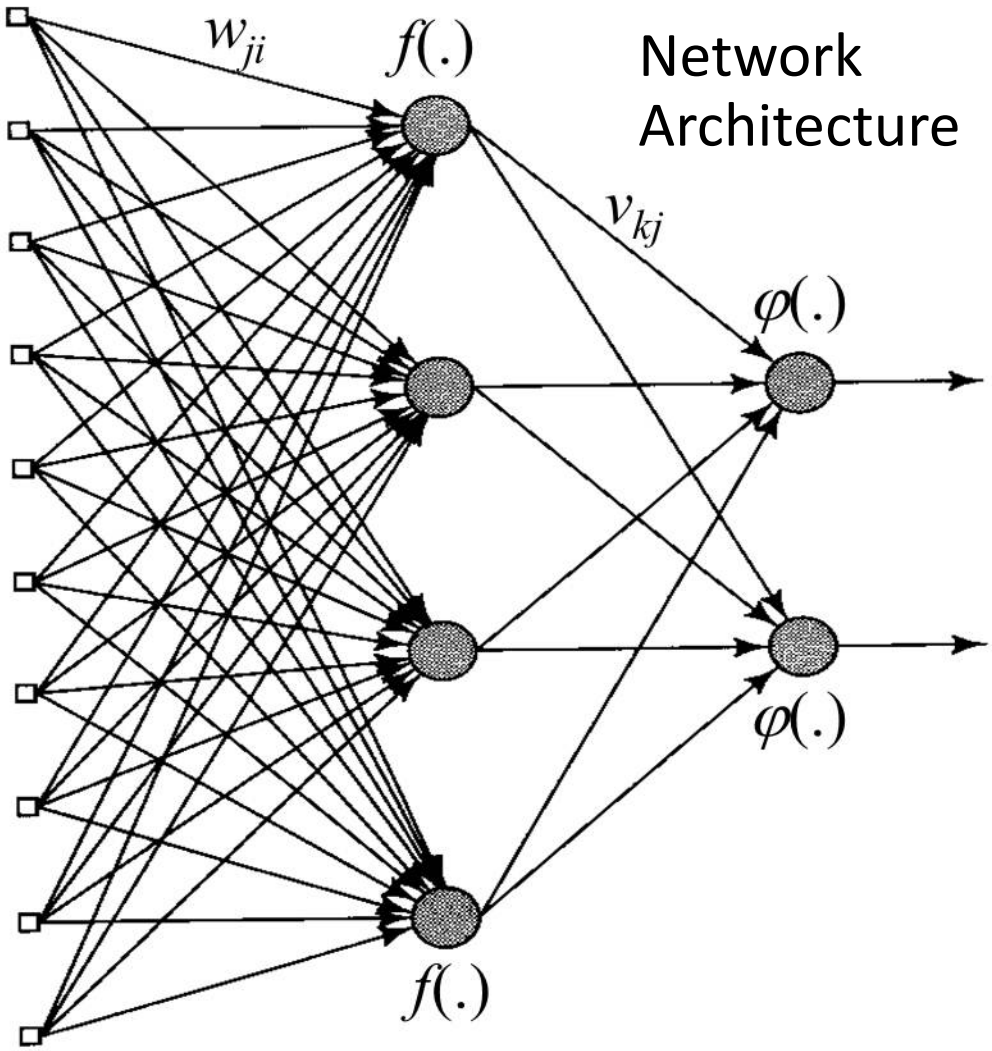
Outline

- Motivation of the Connectionist Approaches and Neural Networks
- Network Architecture and Neuron Model
- Optimization and Iterative (Error Correction) Learning
- Multilayer Perceptron (MLP) and Backpropagation Learning Algorithm
- Radial Basis Function (RBF) Neural Networks
- Understanding Deep Learning and Convolutional neural Networks (CNN)

13 Connectionist Approaches & Neural Networks

What are the connectionist approaches and neural networks?

Connectionist approaches model the pattern recognition systems as interconnected networks of simple units. Artificial neural networks (ANN), or simply, neural networks are typical connectionist models.



13 ANN -- Characteristics

Why is it called (artificial) neural network?

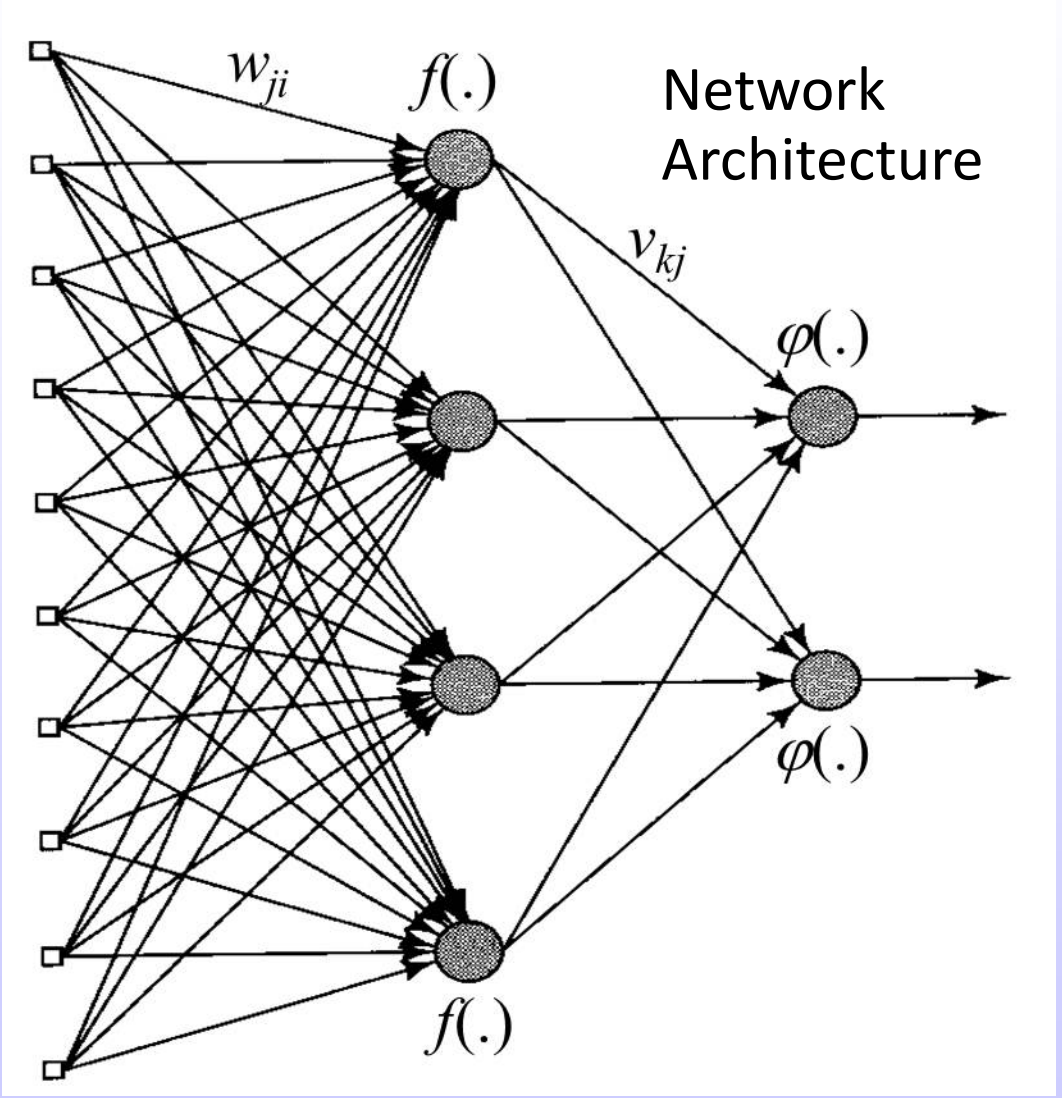
It has certain characteristics in common with biological neuron networks:

(1) Information processing occurs at many same simple elements called neurons;

(2) Information are processed in parallel;

(3) Nonlinear information processing;

(4) Learning capability.



13 ANN -- Motivation

Why do people create and study artificial neural networks?

Traditional, sequential, logic-based computers excel in arithmetic, but is less effective than human brains in many fields. A good example is the processing of visual information. A one-year-old baby is much better and faster at recognizing objects, faces and so on than even the most advanced artificial intelligence (AI) systems running on the fastest supercomputer.

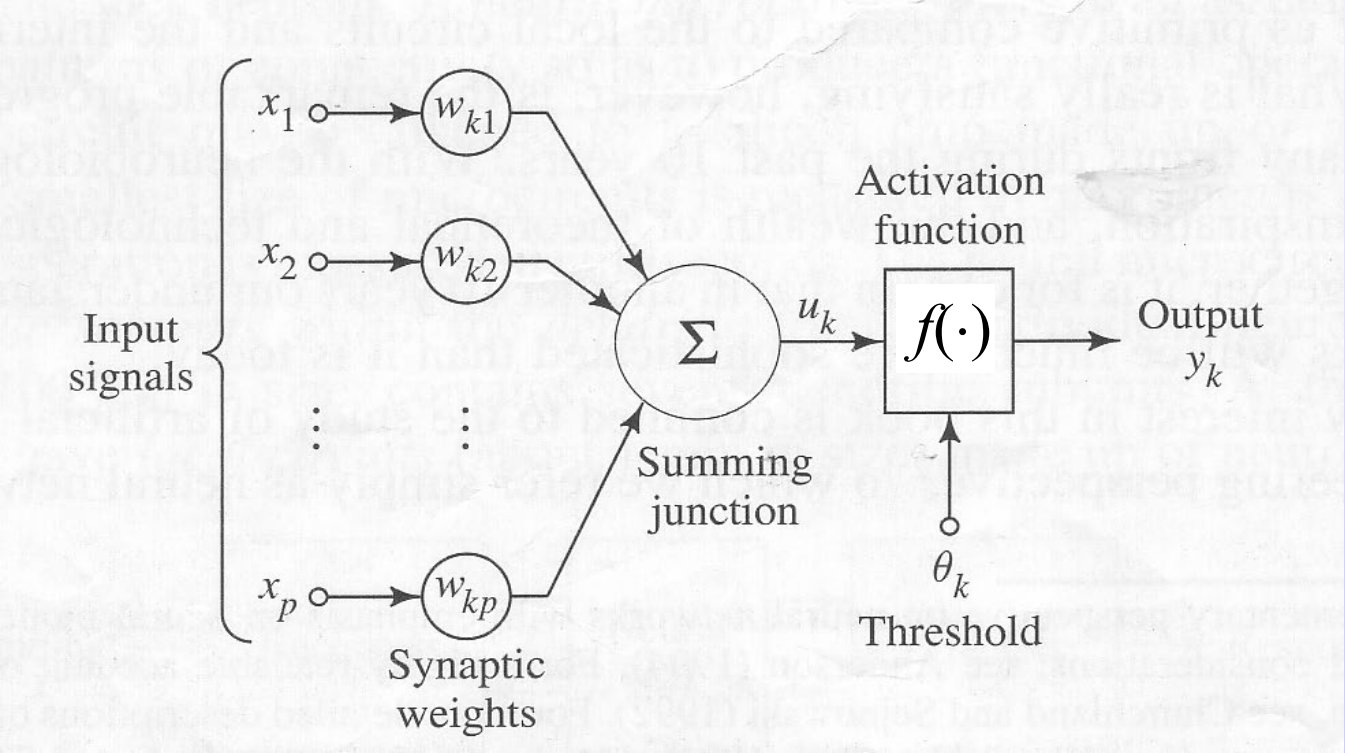
As modern computers become ever more powerful, scientists continue to be challenged to use machines effectively for some tasks that are relatively simple to humans. Thus, people try to build a machine information processing system to simulate human brains.

However, after over 3 decades of hot research, ANN seems not as promising as people expected at beginning. Nevertheless, it provides one alternative approaches to learning and recognition.

13 ANN -- Neuron Model

Neuron Model

A neuron is an information processing unit that is fundamental to the operation of a neural network. The model of a neuron is shown below:



13 ANN -- Neuron Model

In mathematical terms, we can describe the neuron as:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j$$

$$y_k = f(u_k - \theta_k)$$

Where $x_1, x_2, \ldots, x_p$ are the input signals, $w_{k1}, w_{k2}, \ldots, w_{kp}$ are the synaptic weights of neuron k, u_k is the linear combiner output, θ_k is the threshold, $f(.)$ is the activation function and y_k is the neurons output.

θ_k is an external parameter, we can consider this parameter as an input variable:

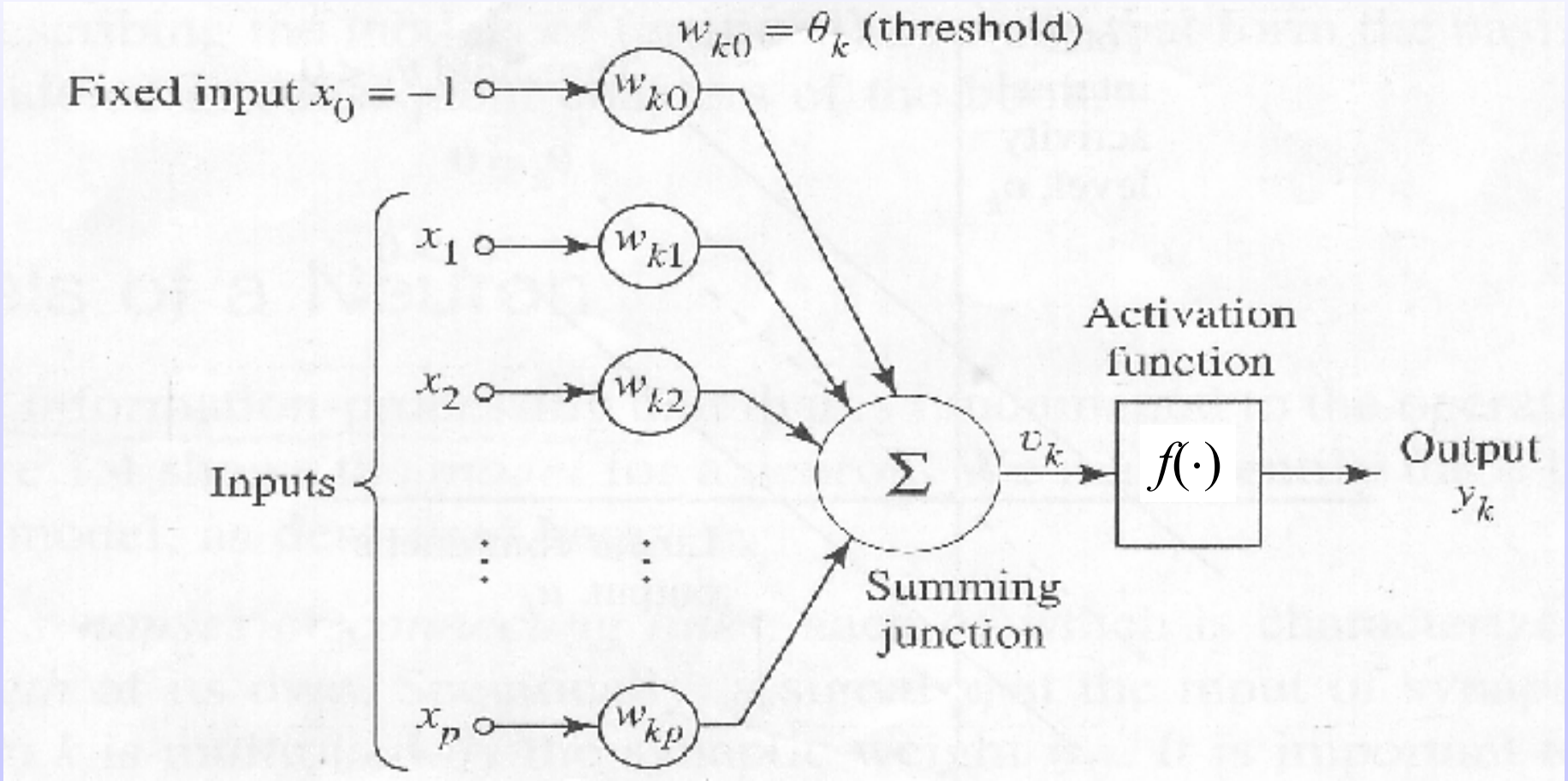
$$x_0 = 1, \qquad w_{k0} = -\theta_k$$

Then we have:

13 ANN -- Neuron Model

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \qquad y_k = f(v_k)$$

Thus, the diagram of the model becomes:



13 ANN -- Activation Functions

Types of Activation Functions

The activation function defines the output of a neuron in terms of the activation level at its input. We may identify three basic types of activation functions:

Binary Function:

$$f(v)=\begin{cases}1 & v\geq 0\\ 0 & v<0\end{cases}$$

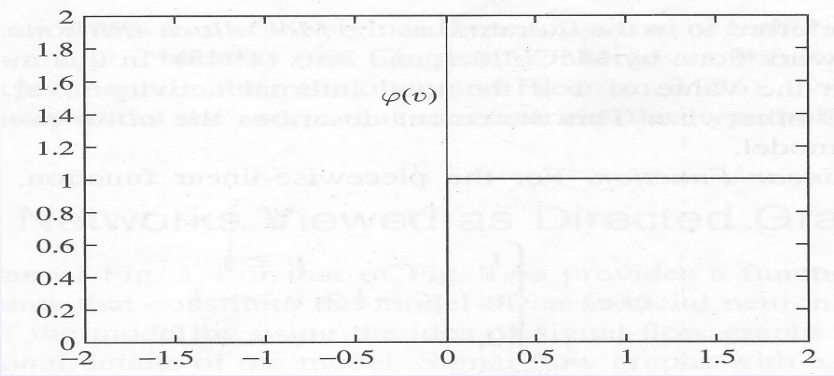
Piecewise-Linear Function

$$f(v)=\begin{cases}1 & v\geq 0.5\\ v & -0.5<v<0.5\\ 0 & v\leq -0.5\end{cases}$$

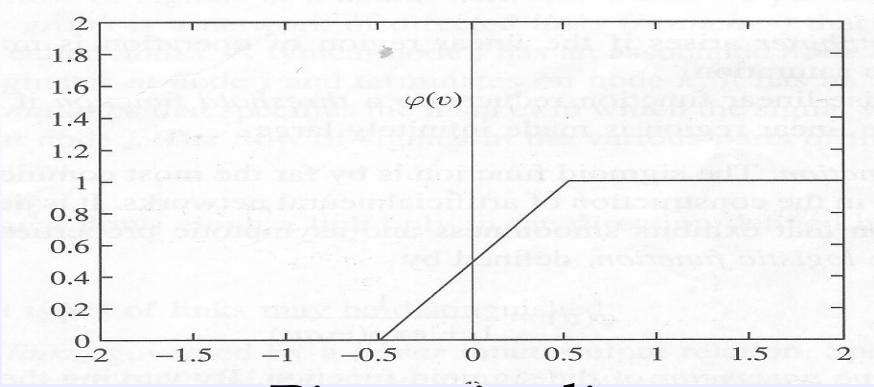
Sigmoid Function

$$f(v)=\frac{1}{1+\exp(-av)}$$

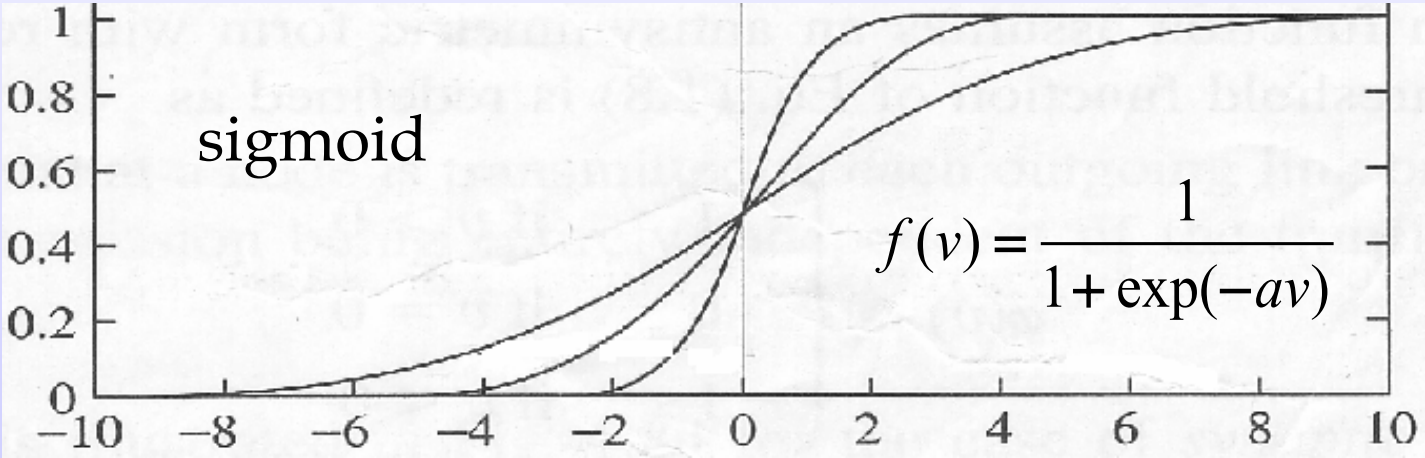
13 ANN -- Activation Functions



binary



Piecewise linear



"a" is the parameter that controls the slope of the sigmoid. A bigger value of "a" results in a steeper slope of the curve. The suitable value of the parameter is application dependent.

13 ANN – Learning from Samples

The major task of a neural network is to learn a model of the world based on the known knowledge of the world. Knowledge of the world consists of two kinds of information:

(1) The known world state, represented by facts about what is and what has been known. This form of knowledge is often referred to as *prior information*.

(2) Observations (measurements) of the world, obtained by sensors designed to probe the world. The observations so obtained provide the pool of information from which neural networks learn. These observations are often referred to as *examples*.

The examples (samples) can be *labeled* or *unlabeled*.

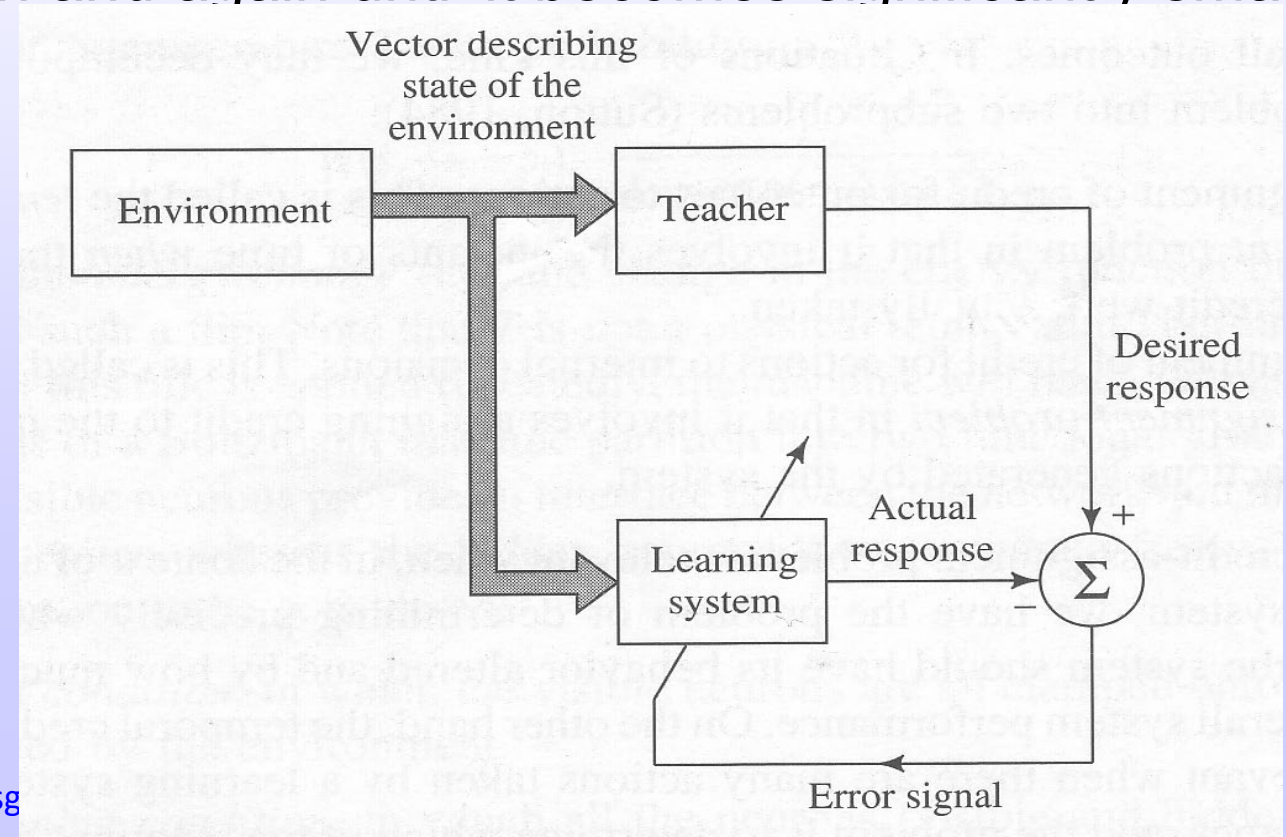
Supervised Learning

During the training session of a neural network, an input is applied to the network, and a response of the network is obtained. The response is compared with an *a priori* target response. If the actual response differs from the target response, the neural network generates an error signal, which is then used to compute the adjustment that should be made to the network's synaptic weights so that the actual response matches the target output. In other words, the error is minimized, possibly to zero. Since the minimization process requires a teacher (supervisor), this kind of training is named *supervised learning*.

The notion of teacher comes from biological observations. For example, when learning a language, we hear the sound of a word from a teacher. The sound is stored in the

Supervised Learning

memory banks of our brain, and we try to reproduce the sound. When we hear our own sound, we mentally compare it (actual response) with the stored sound (desired response) and note the error. If the error is large, we try again and again until it becomes significantly small.

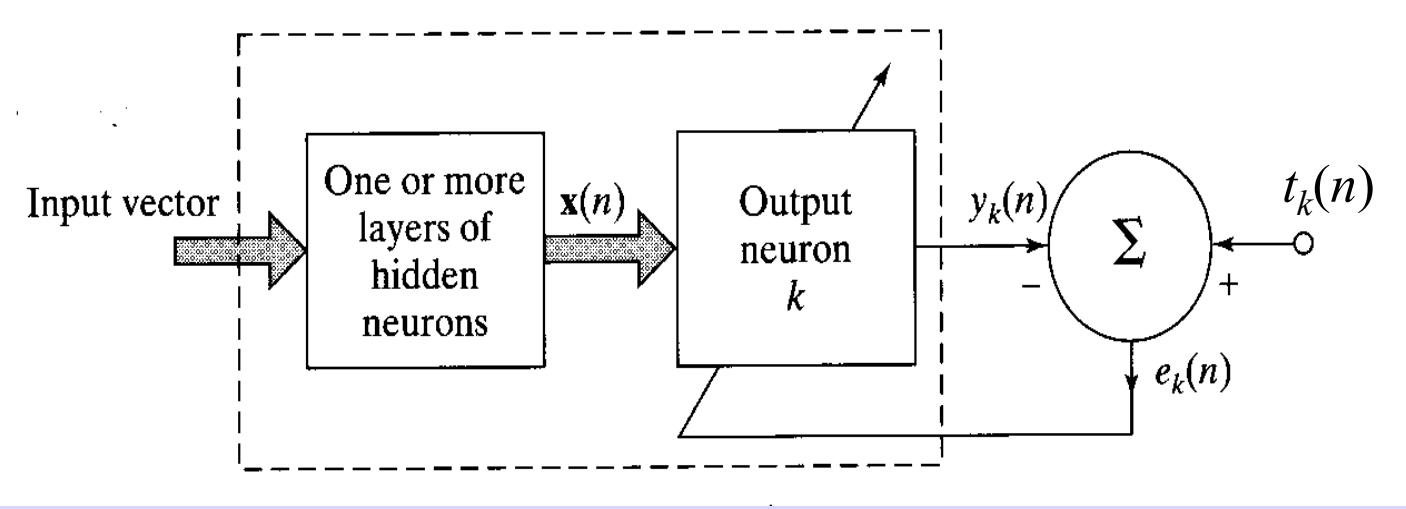


Unsupervised Learning

In contrast to supervised learning, *unsupervised learning* does not require a teacher, *i.e.* there is no target response. During the training stage, the neural network receives its input patterns and it arbitrarily organizes the pattern into categories. When an input is later applied, the neural network provides an output response to indicate the class to which the input pattern belongs. For example, show a person a set of different objects. Then ask him to separate them into different groups, such that objects in a group have one or more common features that distinguish them from other groups. When this (training) is done, show the same person an object that is unseen and ask him to place the object in one of the groups, he would put it in the group with which the object has most common features.

13 ANN – Error-Correction Learning

To illustrate the error-correction learning rule, let's consider the simple case that the output layer of a feed-forward neural network consists of only one neuron, say neuron *k*, as shown below:



Where neuron *k* is driven by a signal vector *X*(n) produced by one or more layers of hidden neurons. The argument *n* denotes the discrete time. The output and desired response of neuron *k* at *n* are denoted by $y_k(n)$ and $t_k(n)$ respectively.

Error-correction Learning

Typically, the actual response is different from the desired response, and the difference between them, *i.e.* error signal, is defined as:

$$e_k(n) = t_k(n) - y_k(n)$$

The error signal actuates a control mechanism, based on which weights are adjusted. The adjustment of weights are designed to make the output signal comes closer to the desired signal in a step-by-step manner. This goal is achieved by minimizing the cost function:

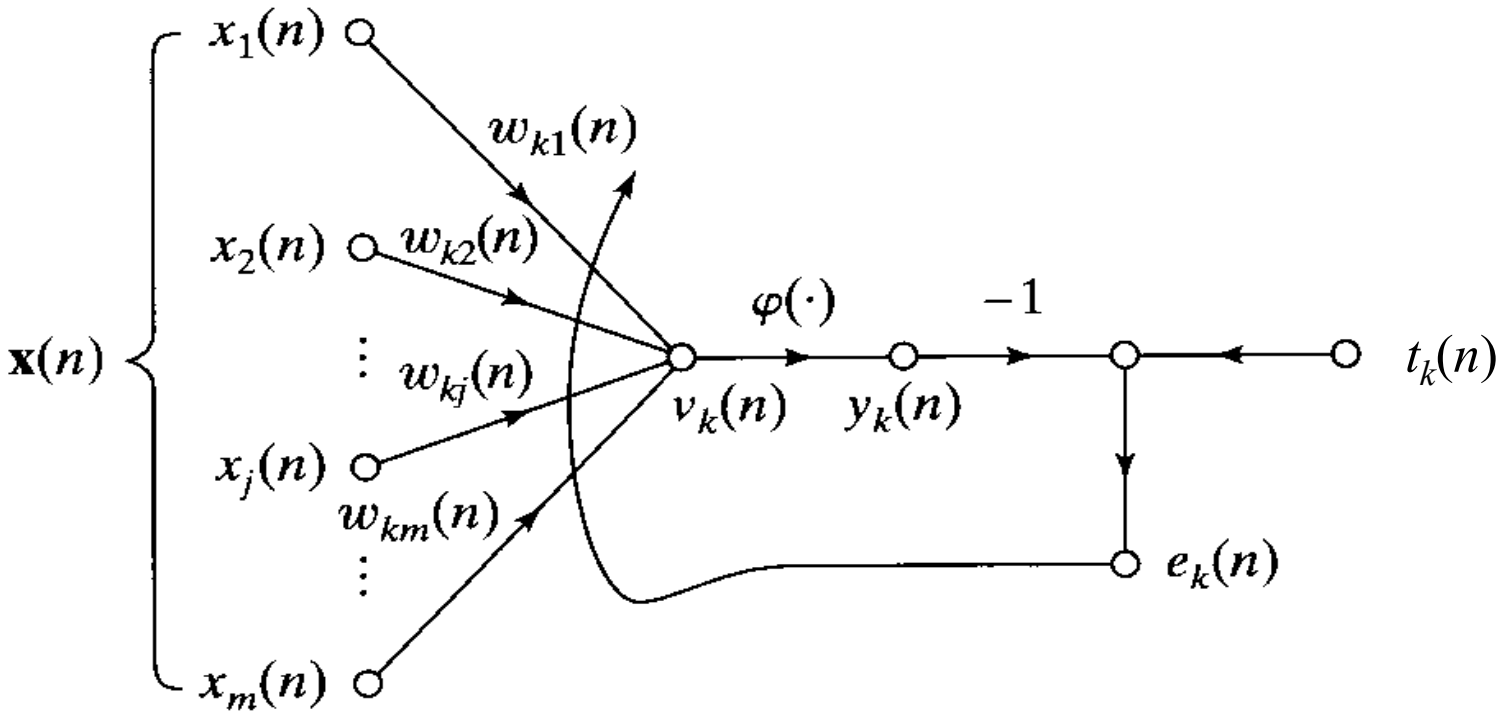
$$J(n) = \frac{1}{2} e_k^2(n) = \frac{1}{2}[t_k(n) - y_k(n)]^2$$

Where J(n) is the instantaneous value of error energy. The step-by-step adjustments to the weights of neuron k are

Error-correction Learning

continued until the weights are stabilized. At that point, the learning process finishes. Minimization of cost function of J(n) leads to a learning rule referred to as the *Widrow-Hoff rule*.

To describe in detail the learning rule, let's consider the signal-flow graph of the output neuron k.



Error-correction Learning

Let $w_{kj}(n)$ denote the value of weight w_{kj} of neuron k excited by signal $x_j(n)$ at time instant n. According to the Widrow-Huff rule, the adjustment applied to weight w_{jk} at time instant n is defined as:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

where η is a positive constant that determines the *rate of learning*. The adjustment of the weight is proportional to the product of the error signal and the corresponding input signal.

Thus, the updated value of the weight w_{kj} is determined by the following rule:

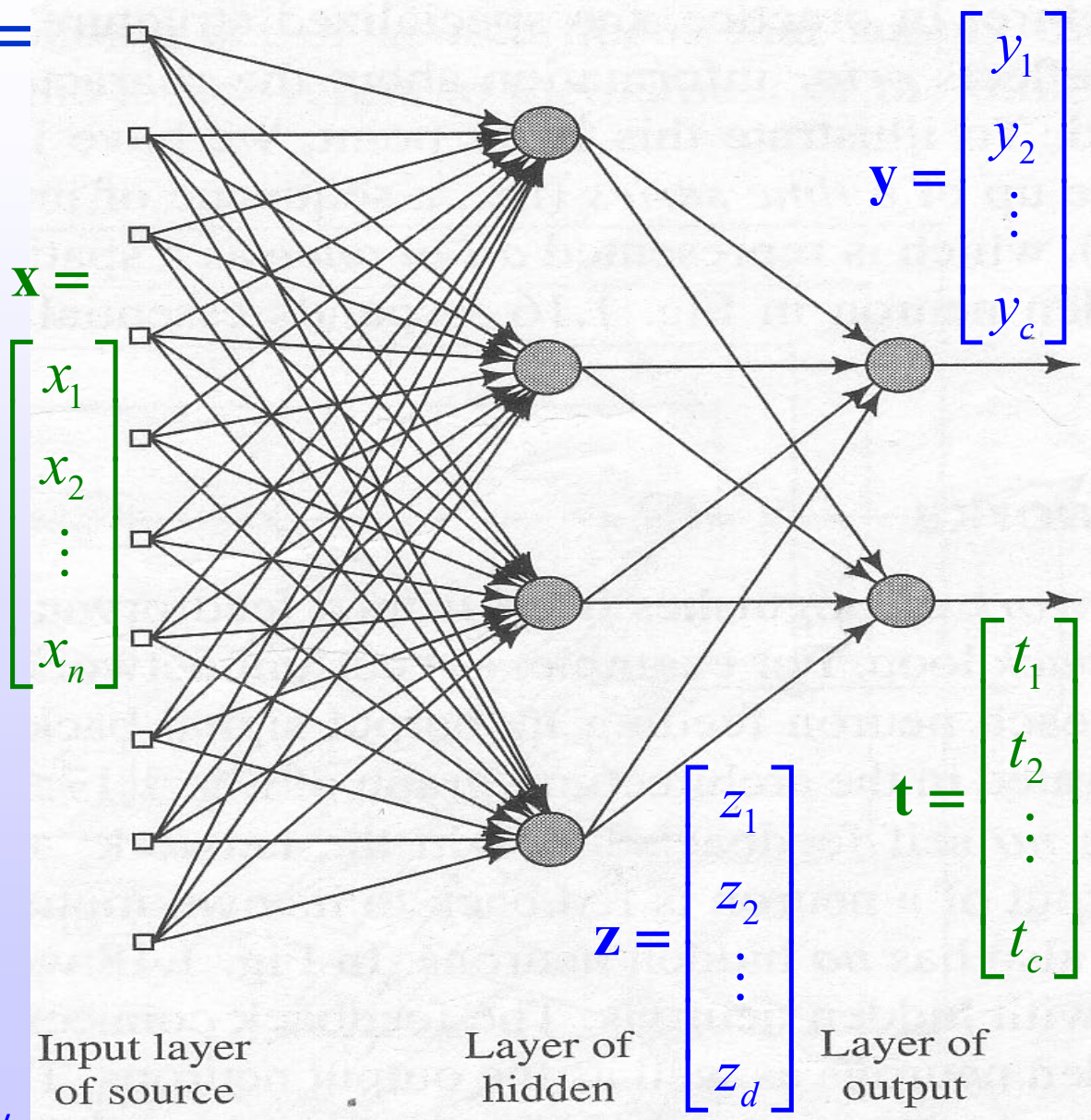
$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

13 ANN – Multilayer Perceptron

$$\mathbf{w}_j = \begin{bmatrix} w_{j1} \\ w_{j2} \\ \vdots \\ w_{jn} \end{bmatrix}, \mathbf{v}_k = \begin{bmatrix} v_{k1} \\ v_{k2} \\ \vdots \\ v_{kd} \end{bmatrix}$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_d]$$

$$\mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_c]$$



exdjiang@ntu.edu.sg http://w

13 ANN – Multilayer Perceptron

Mathematical expression of outputs from inputs of neural networks in scalar, vector and matrix forms

$$z_j = f\left(\sum_{i=1}^{n} w_{ji} x_i\right) = f\left(\mathbf{w}_j{}^T \mathbf{x}\right)$$

$$y_k = f\left(\sum_{j=1}^{d} v_{kj} z_j\right) = f\left(\mathbf{v}_k{}^T \mathbf{z}\right) = f\left[\sum_{j=1}^{d} v_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i\right)\right]$$

$$\Downarrow$$

$$\mathbf{y} = f\left(\mathbf{V}^T \mathbf{z}\right) = f\left[\mathbf{V}^T f\left(\mathbf{W}^T \mathbf{x}\right)\right]$$

13 ANN – Multilayer Perceptron

- If the activation faction f is a linear function,

$$f(r) = r$$

$$\Downarrow$$

$$\mathbf{y} = f\left[\mathbf{V}^T f\left(\mathbf{W}^T \mathbf{x}\right)\right] = \mathbf{V}^T \mathbf{W}^T \mathbf{x} = (\mathbf{WV})^T \mathbf{x} = \mathbf{U}^T \mathbf{x}$$

$$\text{where } \mathbf{U} = \mathbf{WV}$$

- Multilayer network will be just the same as a single layer network.
- To design a multi-layer neural network, it is thus very important that the activation function should be a nonlinear function, at least for hidden neurons.

13 ANN – Single Layer Neural Network

- Let's first study the single layer network of single output

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

- If we have a target output t, the error of the output is:

$$e = t - y = t - \mathbf{w}^T \mathbf{x}$$

- The mean square error is:

$$J(\mathbf{w}) = E\{e^2\} = E\{(t-y)^2\} = E\{(t-\mathbf{w}^T\mathbf{x})^2\}$$

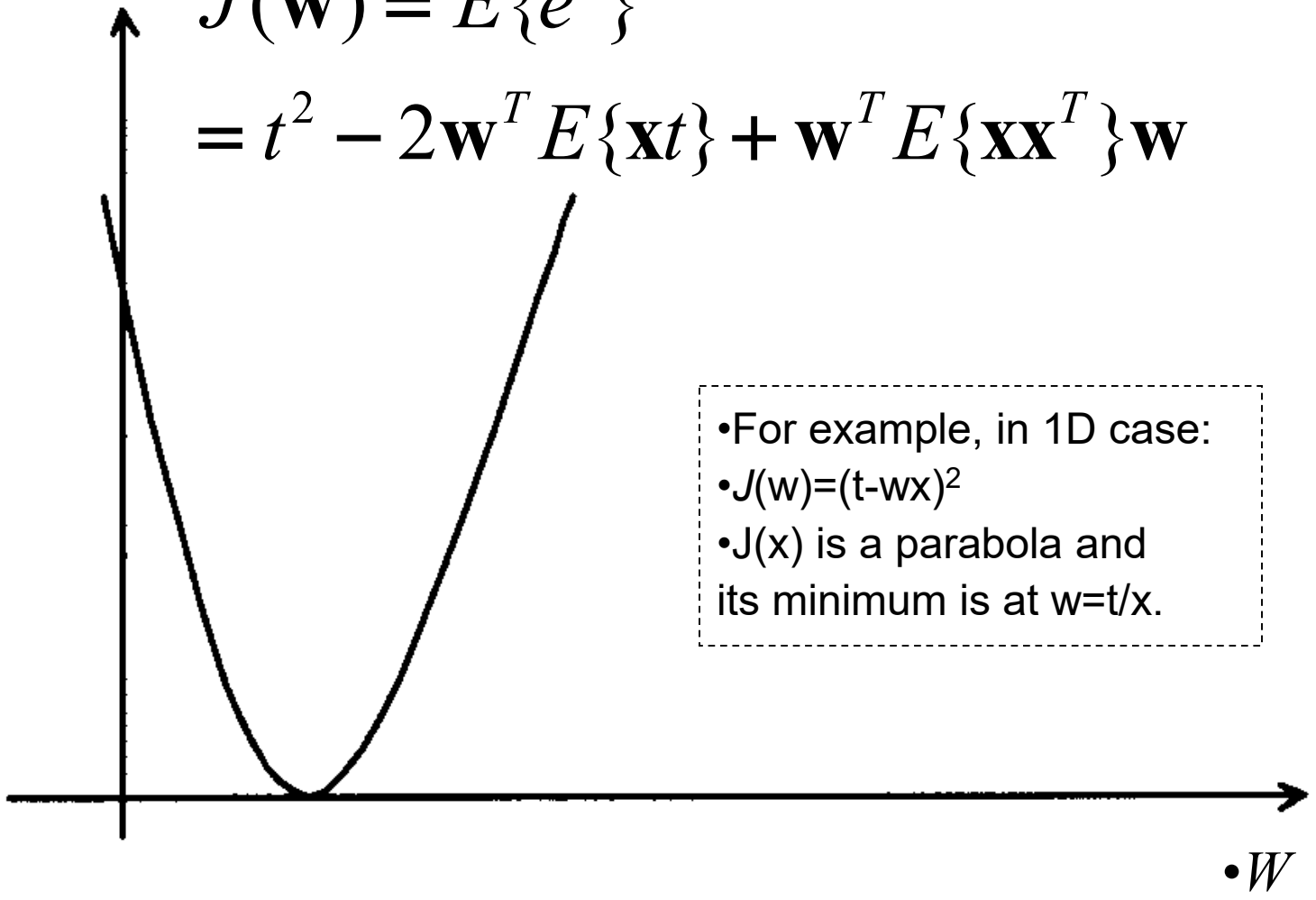
$$= t^2 - 2\mathbf{w}^T E\{\mathbf{x}t\} + \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w}$$

- Obviously, the error function is a quadratic function (second order) of $\mathbf{w}$. It has only one minimum point.

13 ANN – Single Layer Neural Network

$$J(\mathbf{w}) = E\{e^2\}$$

$$= t^2 - 2\mathbf{w}^T E\{\mathbf{x}t\} + \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w}$$



- For example, in 1D case:
- $J(w)=(t-wx)^2$
- J(x) is a parabola and its minimum is at w=t/x.

13 ANN – Single Layer Neural Network

- The optimal network parameter is obtained by minimizing the mean square error. This leads to the least mean square solution of $\mathbf{w}$:

$$\frac{\partial E\{e^2\}}{\partial \mathbf{w}} = \frac{\partial (t^2 - 2\mathbf{w}^T E\{\mathbf{x}t\} + \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w})}{\partial \mathbf{w}} = 0$$

$$\Downarrow$$

$$E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} = E\{\mathbf{x}t\}$$

- Suppose we have q samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L$,

$$E\{\mathbf{x}\mathbf{x}^T\} \cong \frac{1}{q}\sum_{i=1}^{q} \mathbf{x}_i \mathbf{x}_i^T, \quad E\{\mathbf{x}t\} \cong \frac{1}{q}\sum_{i=1}^{q} \mathbf{x}_i t_i$$

13 ANN – Single Layer Neural Network

- Let $$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_q], \quad \mathbf{t} = [t_1 \quad t_2 \quad \cdots \quad t_q]$$
- Then: $$q \cdot E\{\mathbf{x}\mathbf{x}^T\} \cong \sum_{i=1}^{q} \mathbf{x}_i \mathbf{x}_i^T = \mathbf{X}\mathbf{X}^T$$

$$q \cdot E\{\mathbf{x}t\} \cong \sum_{i=1}^{q} \mathbf{x}_i t_i = \mathbf{X}\mathbf{t}^T$$

$$E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} = E\{\mathbf{x}t\}$$

$$\approx \Downarrow$$

$$\mathbf{X}\mathbf{X}^T\mathbf{w} = \mathbf{X}\mathbf{t}^T \quad \Rightarrow \quad \mathbf{w} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{t}^T$$

- It is also called least square method.

13 ANN – Single Layer Neural Network

- We can also obtained the optimal weights W by iteratively adjust/update the value of W using gradient descent method:

$$\mathbf{w} \Leftarrow \mathbf{w} + \Delta\mathbf{w}$$

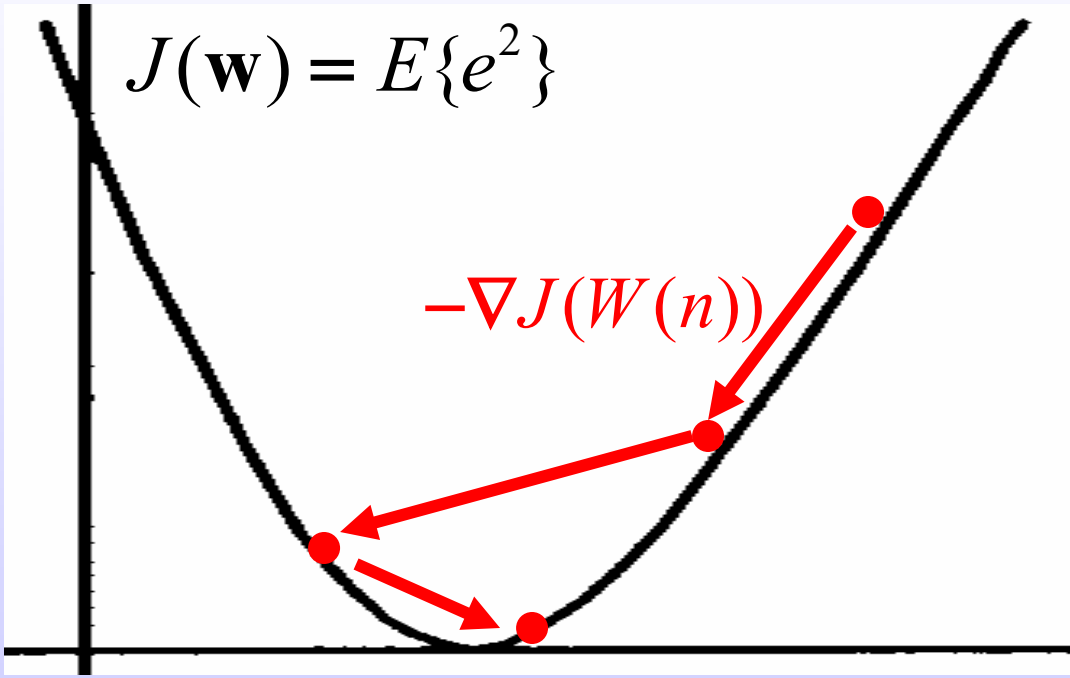
$$\Delta\mathbf{w} = -\eta \nabla J(\mathbf{w})$$

$$\nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

$$= \frac{\partial E\{(t - \mathbf{w}^T\mathbf{x})^2\}}{\partial \mathbf{w}}$$

$$= -2E\{(t - \mathbf{w}^T\mathbf{x})\mathbf{x}\}$$

$$= -2E\{e\mathbf{x}\} \cong -2e_i\mathbf{x}_i$$

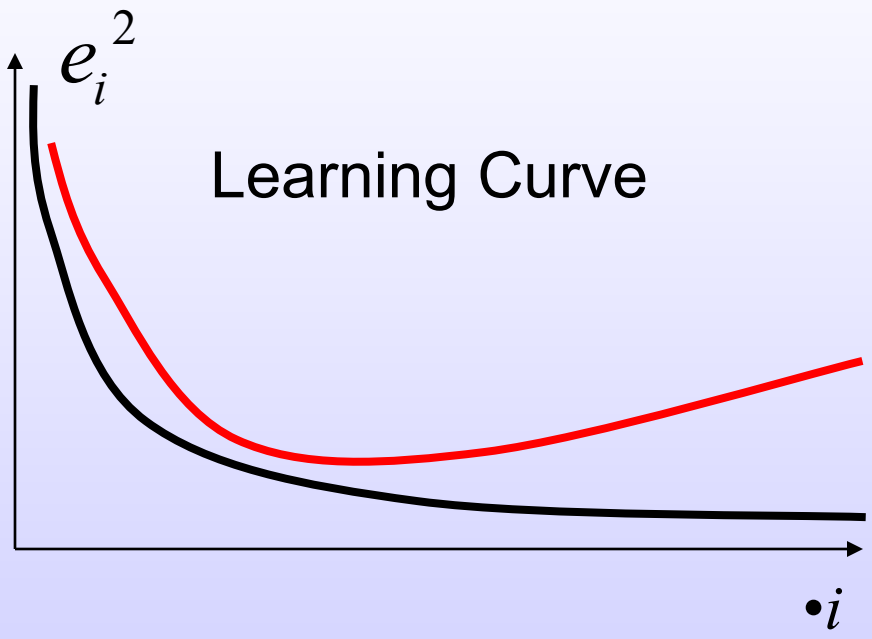


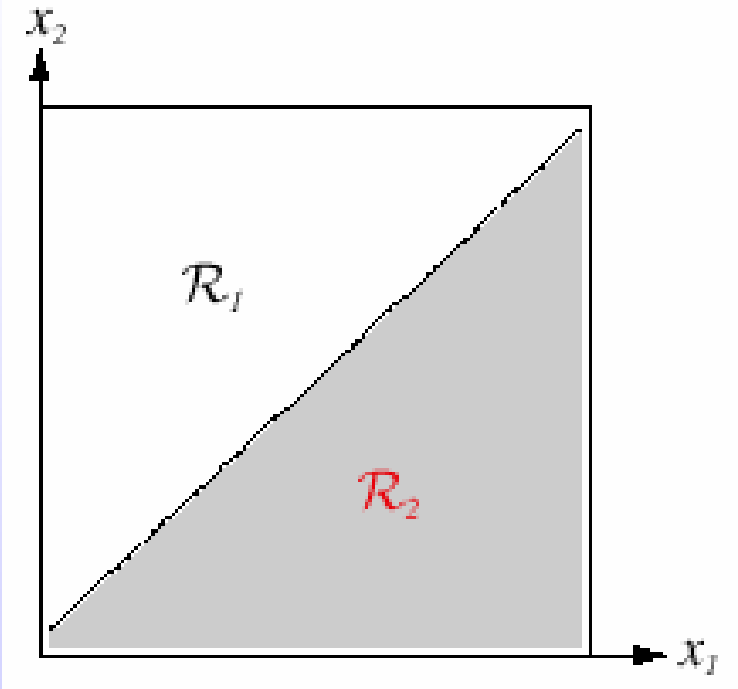
$$\mathbf{w} \Leftarrow \mathbf{w} + \eta e_i\mathbf{x}_i = \mathbf{w} + \eta(t_i - y_i)\mathbf{x}_i$$

$$\rightarrow \mathbf{w} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{t}^T$$

13 ANN – Learning Curve and Decision Boundary

$$\mathbf{w} \Leftarrow \mathbf{w} + \eta(t_i - y_i)\mathbf{x}_i$$





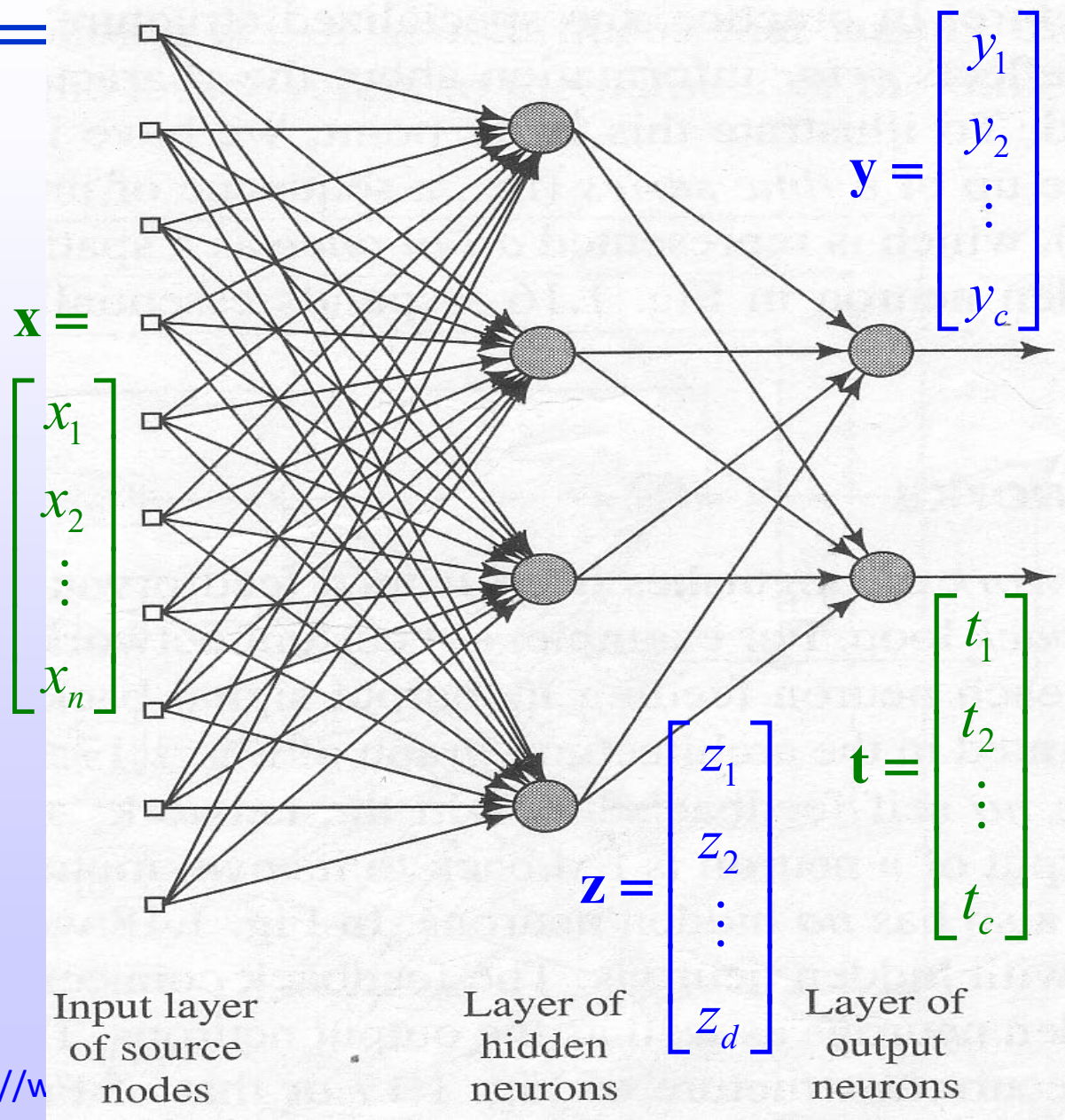
Decision boundary is a strait line or plane or hyper-plane.

$$\mathbf{w}^T\mathbf{x} = 0$$

13 ANN – Multilayer Perceptron

$$\mathbf{y} = f\left(\mathbf{V}^T\mathbf{z}\right)$$
$$= f\left[\mathbf{V}^T f\left(\mathbf{W}^T\mathbf{x}\right)\right]$$

Any decision can be implemented by a two-layer network. Any function from input to output can be implemented in a two-layer net, given sufficient number of hidden units, proper nonlinearities, and weights.



13 ANN – Multilayer Perceptron

- These results are of greater theoretical interest than practical, since the construction of such a network requires the nonlinear functions and the weight values which are unknown!
- How to find the nonlinear functions based on data is the central problem in network-based pattern recognition.
- Our goal now is to set the interconnection weights based on the training patterns and the desired outputs
- It is a straightforward matter to understand how the output, and thus the error, depend on the hidden-to-output layer weights
- The power of backpropagation is that it enables us to compute an effective error for each hidden unit, and thus derive a learning rule for the weights.

13 ANN – Multilayer Perceptron/backpropagation

$$\mathbf{y} = f\left(\mathbf{V}^T\mathbf{z}\right) = f\left[\mathbf{V}^T f\left(\mathbf{W}^T\mathbf{x}\right)\right]$$

$$J(\mathbf{w},\mathbf{v}) = E\{e^2\} = E\{\|\mathbf{t}-\mathbf{y}\|^2\} = E\left\{\left\|\mathbf{t} - f\left[\mathbf{V}^T f\left(\mathbf{W}^T\mathbf{x}\right)\right]\right\|^2\right\}$$

To minimize $J(\mathbf{w},\mathbf{v})$, let $\nabla J(\mathbf{w},\mathbf{v}) = 0$

There is no analytical solution to this equation due to the nonlinear function f.

However, we can use gradient decent method so long as the gradient is computable for a given sample $\mathbf{x}_i$.

$$\mathbf{w} \Leftarrow \mathbf{w} - \eta\nabla J(\mathbf{w})$$

$$\mathbf{v} \Leftarrow \mathbf{v} - \eta\nabla J(\mathbf{v})$$

13 ANN – Multilayer Perceptron/backpropagation

- We use scalar form of weights to derive the learning formula.

$$J(\mathbf{w},\mathbf{v}) = E\{e^2\} = E\{\| \mathbf{t}-\mathbf{y} \|^2\} = \frac{1}{2}\sum_{k=1}^{c}(t_k - y_k)^2$$

$$= \frac{1}{2}\sum_{k=1}^{c}\left[t_k - f(q_k)\right]^2 = \frac{1}{2}\sum_{k=1}^{c}\left[t_k - f\left(\sum_{j=1}^{d} v_{kj} z_j\right)\right]^2$$

$$\frac{\partial J(\mathbf{w},\mathbf{v})}{\partial v_{kj}} = \frac{\partial J}{\partial q_k}\cdot\frac{\partial q_k}{\partial v_{kj}} = -(t_k - y_k) f'(q_k) z_j$$

13 ANN – Multilayer Perceptron/backpropagation

$$J(\mathbf{w},\mathbf{v}) = \frac{1}{2}\sum_{k=1}^{c}\left[t_k - f\left(\sum_{j=1}^{d} v_{kj} z_j\right)\right]^2 = \frac{1}{2}\sum_{k=1}^{c}\left[t_k - f\left[\sum_{j=1}^{d} v_{kj} f\left(q_j\right)\right]\right]^2$$

$$= \frac{1}{2}\sum_{k=1}^{c}\left[t_k - f\left[\sum_{j=1}^{d} v_{kj} f\left(\sum_{i=1}^{n} w_{ji} x_i\right)\right]\right]^2$$

$$\frac{\partial J(\mathbf{w},\mathbf{v})}{\partial w_{ji}} = \frac{\partial J}{\partial z_j}\cdot\frac{\partial z_j}{\partial q_j}\cdot\frac{\partial q_j}{\partial w_{ji}}$$

$$= -\left[\sum_{k=1}^{c}(t_k - y_k) f'(q_k) v_{kj}\right] f'(q_j) x_i$$

13 ANN – Multilayer Perceptron/backpropagation

$$v_{kj} \Leftarrow v_{kj} - \eta \frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial v_{kj}}$$

$$w_{ji} \Leftarrow w_{ji} - \eta \frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial w_{ji}}$$

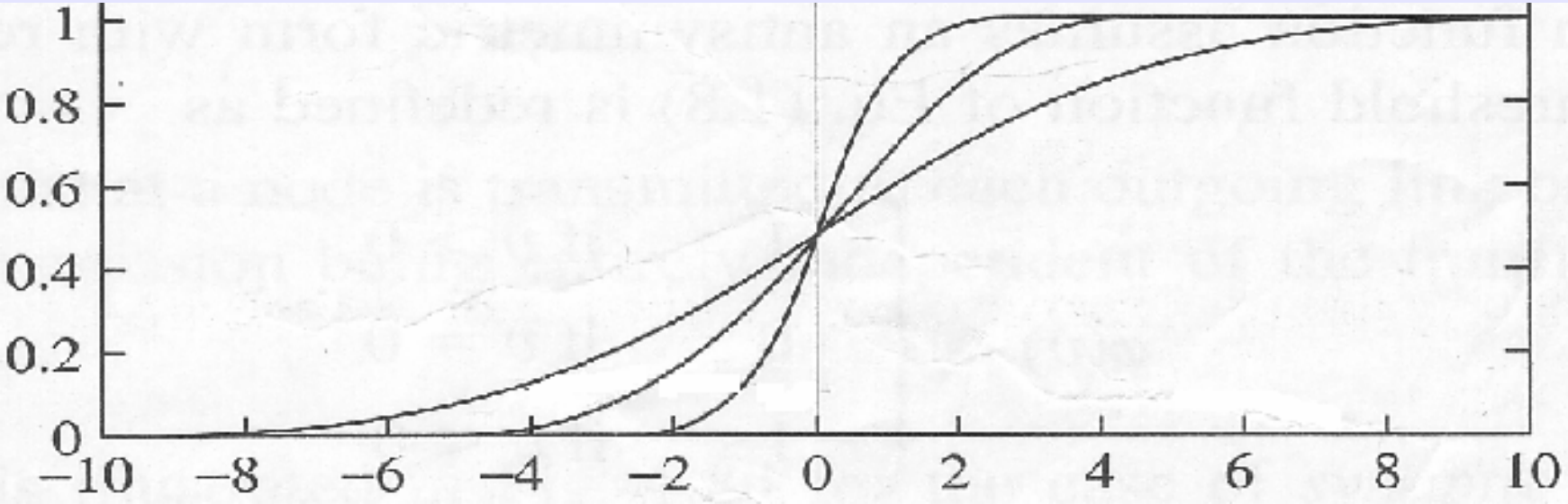
$$v_{kj} \Leftarrow v_{kj} + \eta (t_k - y_k) f'(q_k) z_j$$

$$w_{ji} \Leftarrow w_{ji} + \eta \left[\sum_{k=1}^{c} v_{kj} (t_k - y_k) f'(q_k) \right] f'(q_j) x_i$$

13 ANN – Multilayer Perceptron/backpropagation

$$f(q) = \frac{1}{1+\exp(-aq)}, \quad 0 \le f(q) \le 1$$

$$f'(q) = \frac{a\exp(-aq)}{[1+\exp(-aq)]^2} = af(q)[1-f(q)]$$



13 ANN – Multilayer Perceptron/backpropagation

- Such network is also called multilayer perceptron (MLP) having two modes of operation:

  - Feedforward

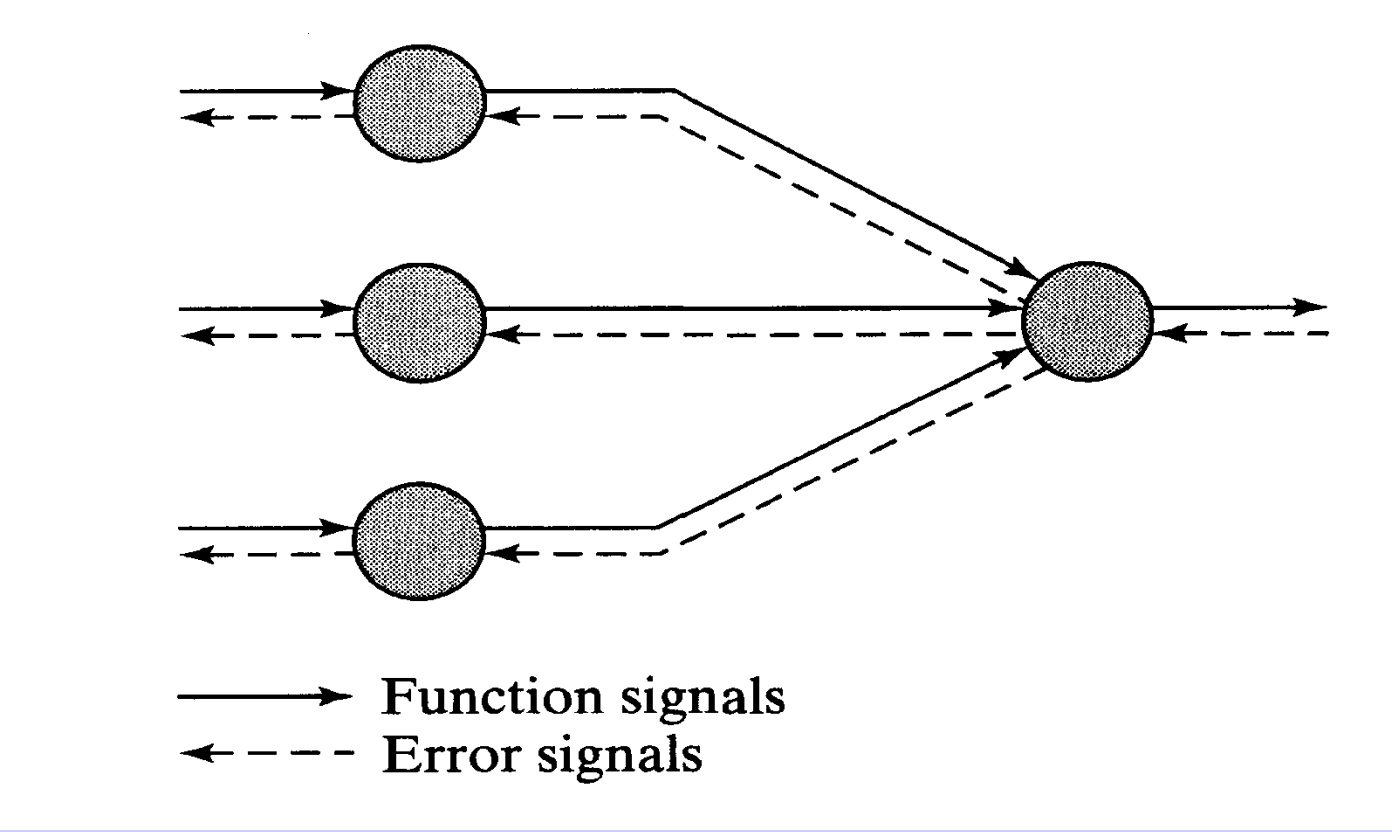
    The feedforward operations consists of presenting a pattern to the input units and passing (or feeding) the signals through the network in order to get outputs units (no cycles!)

  - Learning

    The supervised learning consists of presenting an input pattern and modifying the network parameters (weights) to reduce distances between the computed output and the desired output

13 ANN – Multilayer Perceptron/backpropagation

- This learning algorithm is called backpropagation:
  The following figure shows a portion of the MLP. Two kinds of signals are identified in this network.



13 ANN – Multilayer Perceptron/backpropagation

Backpropagation training procedure:

Assume there are *n* the training samples:

$$\{\mathbf{x}(1), \mathbf{t}(1)\}, \{\mathbf{x}(2), \mathbf{t}(2)\}, \cdots, \{\mathbf{x}(n), \mathbf{t}(n)\}$$

(1) Initialization.

Assuming that no prior information is available, pick the synaptic weights from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the activation signals lie at the transition between linear and saturated parts of the sigmoidal activation function.

(2) Presentation of training samples

Present the network with an epoch of training samples. For each sample in the set, perform the sequence of forward and backward computations.

13 ANN – Multilayer Perceptron/backpropagation

(3) Forward computation

Let a training samples in the epoch be denoted by $\{\mathbf{x}(i), \mathbf{t}(i)\}$, with the $\mathbf{x}(i)$ applied to the input layer and the desired response $\mathbf{t}(i)$ presented to the output layer. Compute the activation signals and function signals of the network by proceeding through the network, layer by layer.

(4) Backward computation

Compute the local gradient of the network, and adjust the synaptic weights of network.

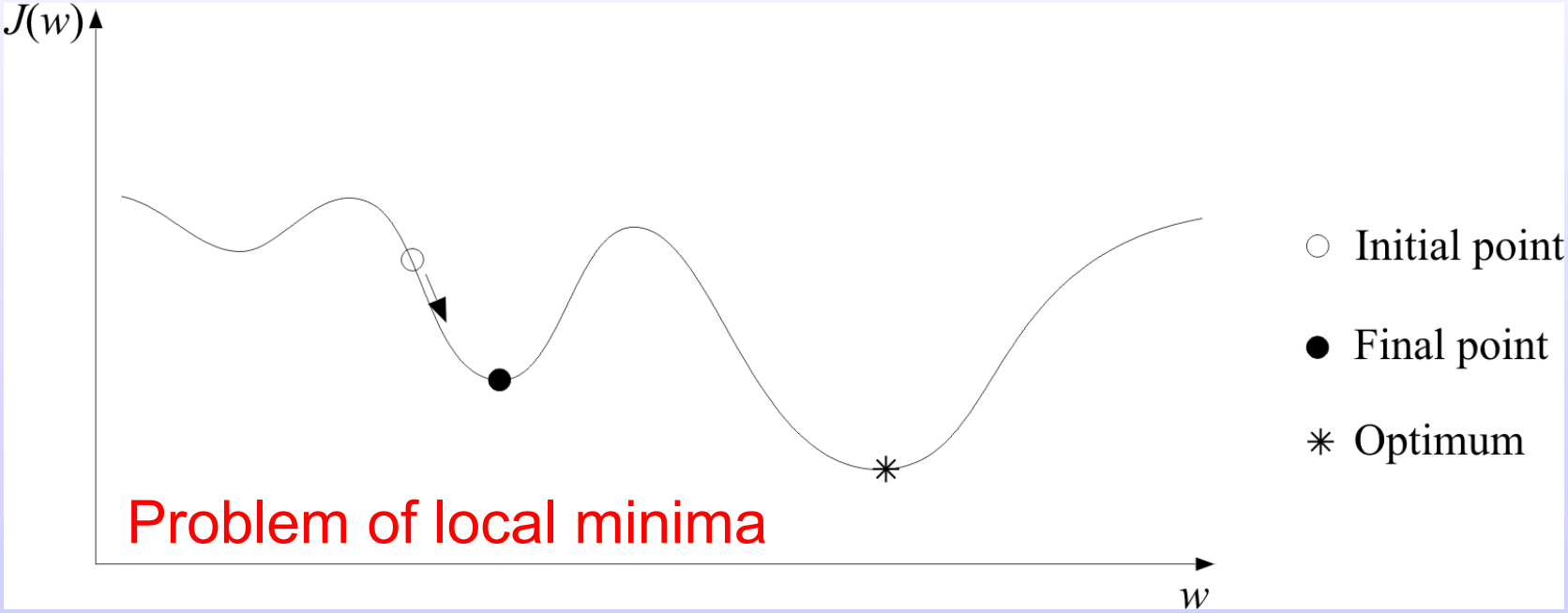
(5) Iteration

Iterate the forward and backward computations in steps (3)-(4) by representing new epochs of training samples to the network until the stopping criterion is met.

13 ANN – Problem of Local Minima

Note, the order of presentation of training samples should be randomized from epoch to epoch.

The stopping criterion can be the number of iterations, or the rate of change of the average error small enough.



X.D. Jiang and A. Kam, Constructing and Training Feed-Forward Neural Networks for Pattern Classification, *Pattern Recognition*, vol. 36, no. 4, pp. 853-867, April 2003.

13 ANN – Learning Curve

Learning Curve

- Before training starts, the error on the training set is high; through the learning process, the error becomes smaller
- The error per pattern depends on the amount of training data and the expressive power (such as the number of weights) in the network
- The average error on an independent test set is always higher than on the training set, and it can decrease as well as increase. (Over-fitting/generalization problem)
- A validation set is used in order to decide when to stop training ; we do not want to over fit the network and decrease the power of the classifier generalization

"We stop training at a minimum of the error on the validation set"

13 ANN – Learning Curve

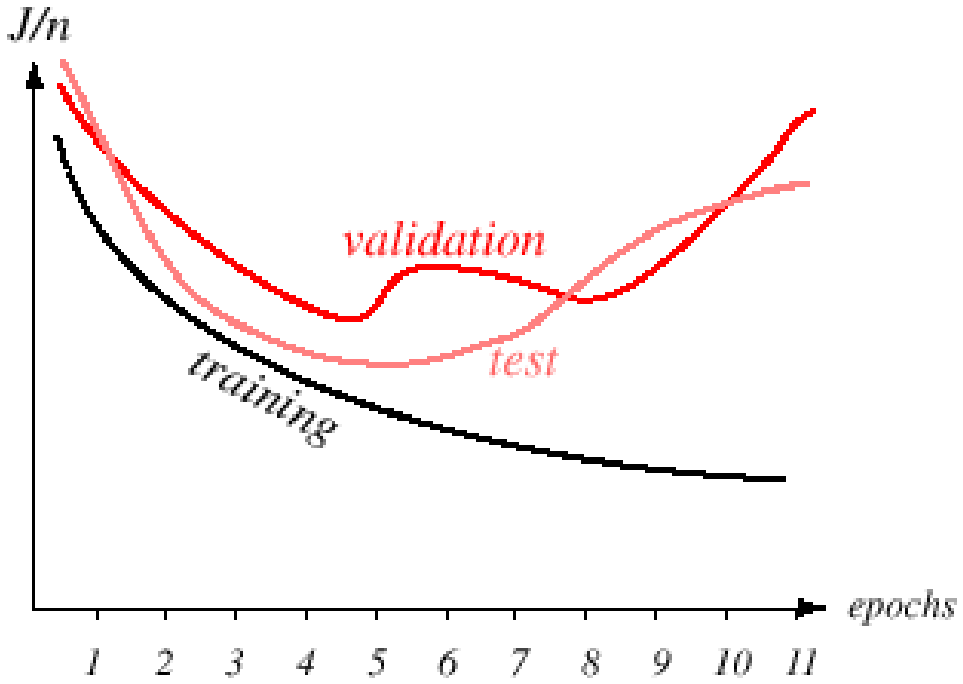


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

13 ANN – Radial Basis Function (RBF) NN

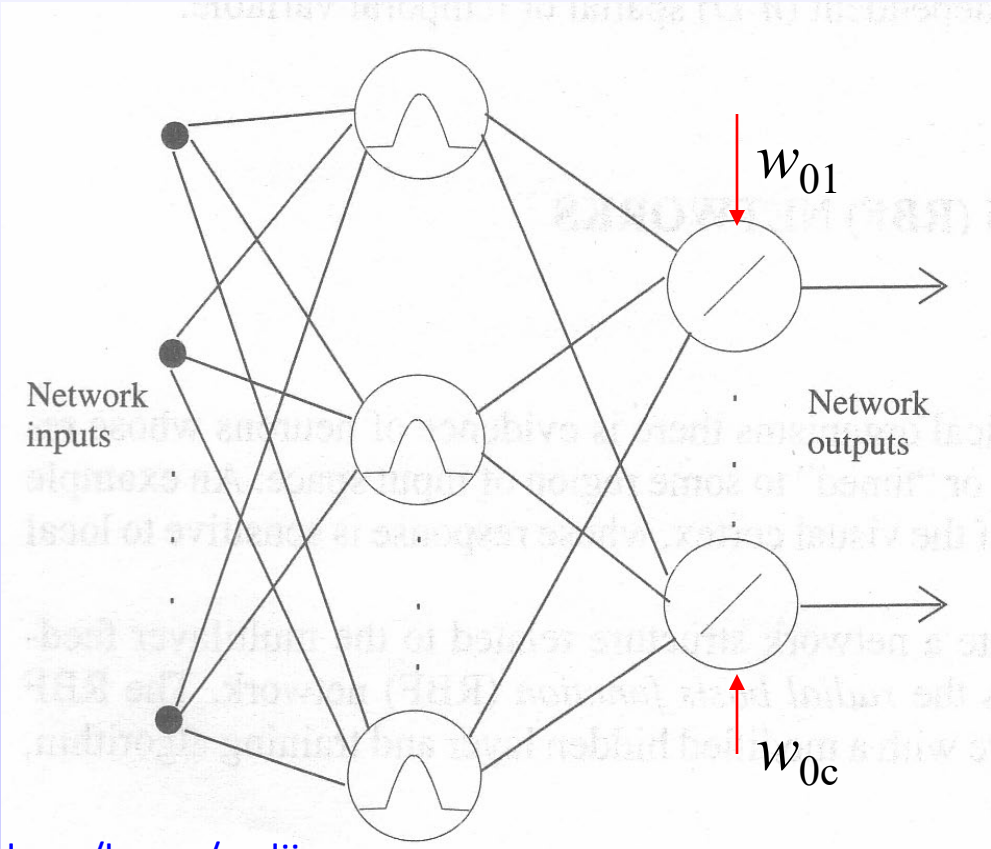
In the nervous system of biological organisms, there is evidence of neurons whose response characteristics are "local" or 'tuned' to some region of input space. An example is the orientation sensitive cells of the visual cortex, whose response is sensitive to local region in the retina.

In this section, we will investigate a network structure related to the multi-layer feed-forward network, known as the radial basis function (RBF) neural network. The RBF neural network has a feed-forward structure with a modified hidden layer and training algorithms.

13 ANN – Radial Basis Function (RBF) NN

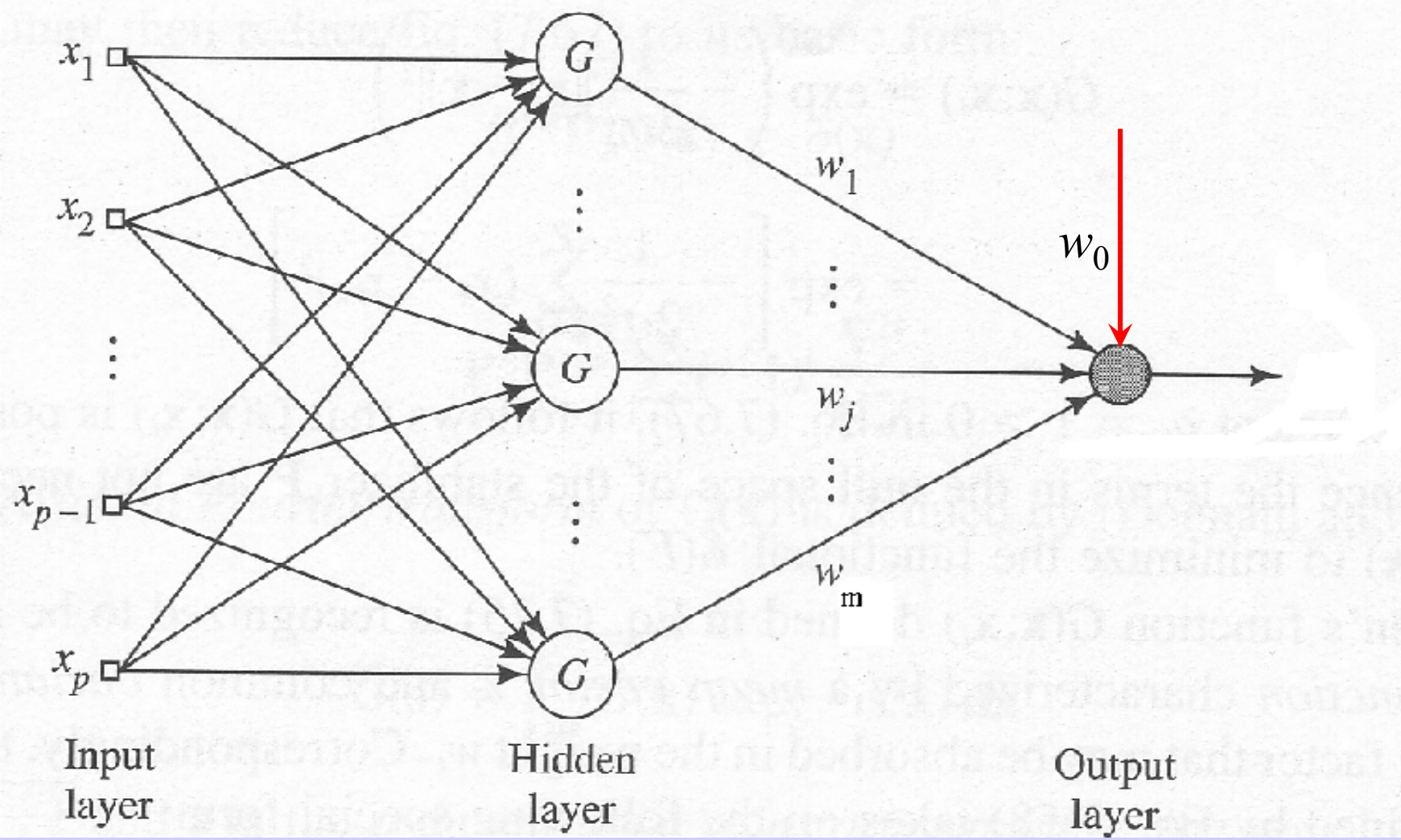
As mentioned, RBF networks emulate the behavior of certain biological networks. Basically, the hidden layer consists of the locally tuned or locally sensitive neurons, and the output layer consists of linear units.

In hidden-layer neurons, the neuron response (output) is localized and decreases as function of the distance of inputs from the neuron's receptive field center.



RBF Neural Network Structure

13 ANN – Radial Basis Function (RBF) NN

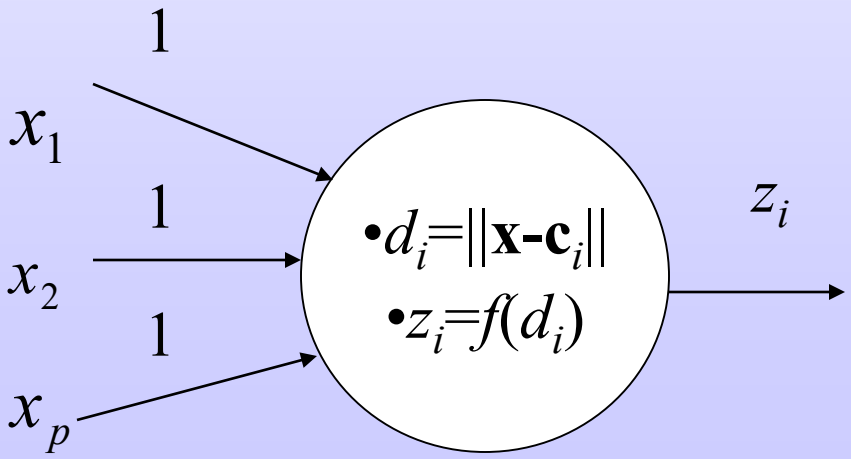


The architecture of RBF with a single output

13 ANN –RBF Neuron Characteristics

Input layer: The same as the input layer of feed-forward network, the input layer neurons do not perform any computation and just distribute the input variables to the hidden layer. But note the weights between input layer neurons and hidden layer neurons in the RBF network are all set to 1.

Hidden layer neurons: a general form for hidden layer neurons of the RBF neural network may be described by:



13 ANN –RBF Neuron Characteristics

In mathematic terms, we can describe the operation of the hidden layer neuron as:

$$d_j = \left\| \mathbf{c}_j - \mathbf{x} \right\| = \sqrt{(\mathbf{c}_j - \mathbf{x})^T (\mathbf{c}_j - \mathbf{x})}$$

$$z_j(X) = f(d_j) = f(\left\| \mathbf{c}_j - \mathbf{x} \right\|)$$

Where C_j is called center vector of neuron j; f is the radial basis function. Some commonly used basis functions are as follows:

(1) The Gaussian function

$$f(d) = \exp\left(-\frac{d^2}{2\sigma^2}\right)$$

where $2\sigma^2$ is the width of the basis function. The suitable value of σ is application dependent.

13 ANN –RBF Neuron Characteristics

(2) The multi-quadratic function:

$$f(d) = \sqrt{d^2 + \sigma^2}$$

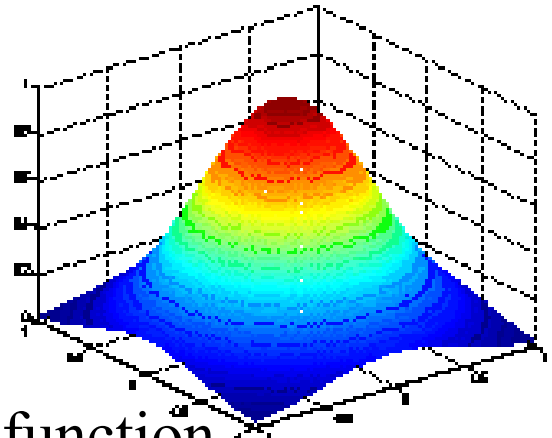
(3) The inverse multi-quadratic function

$$f(d) = 1/\sqrt{d^2 + \sigma^2}$$

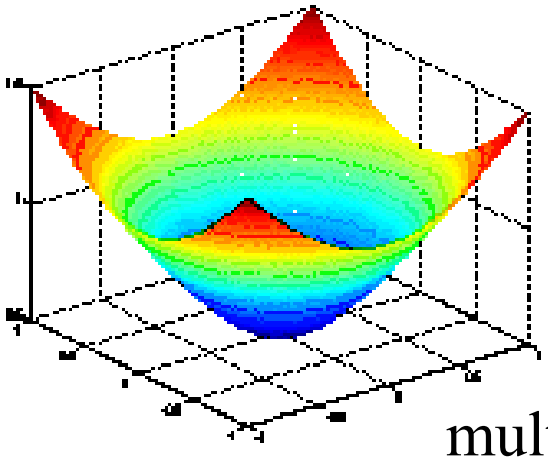
(4) The thin plate spline

$$f(d) = d^2 \log(d)$$

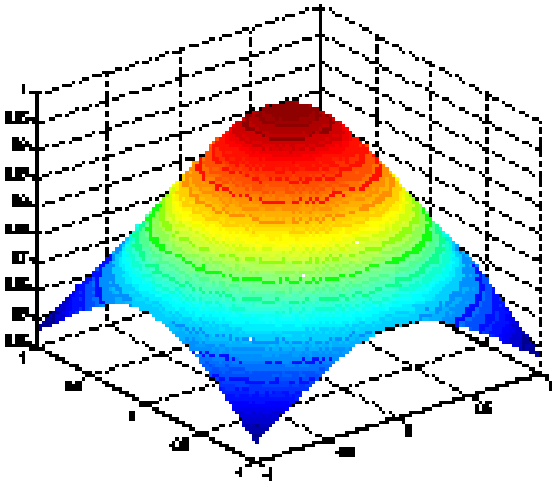
13 ANN –RBF Neuron Characteristics



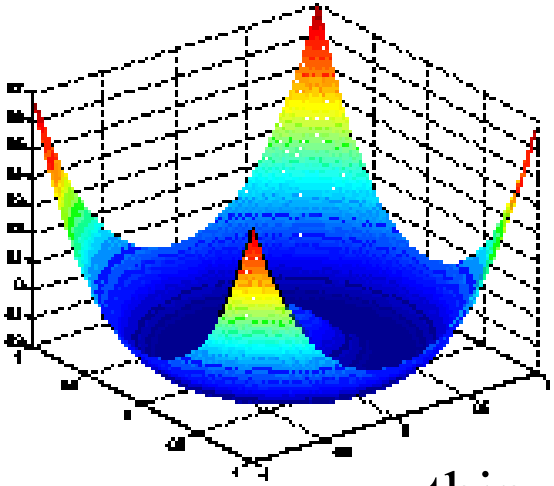
Gaussian function



multi-quadratic



inverse multi-quadratic

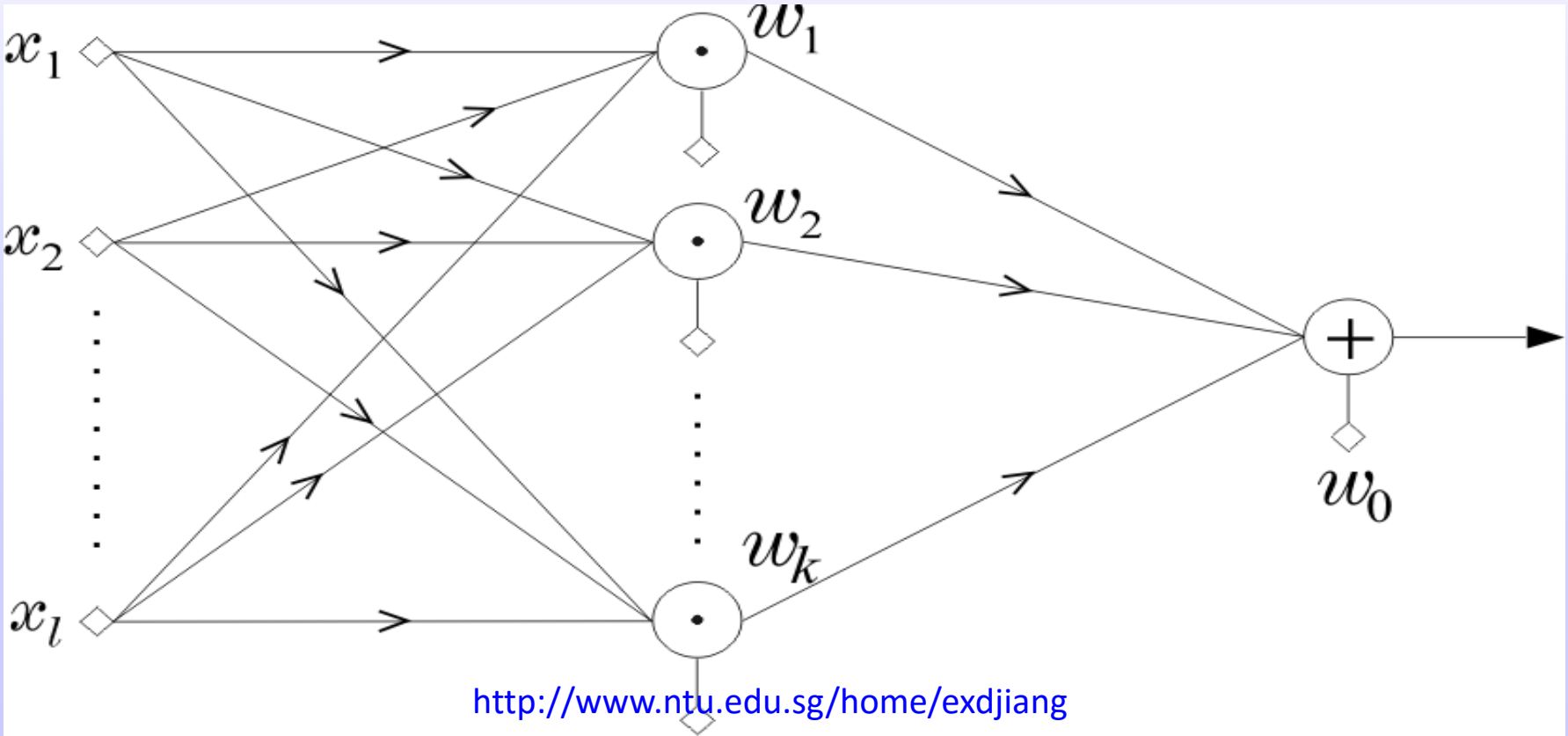


thin-plate spline

13 ANN –RBF Neuron Characteristics

The output layer: The output layer neuron is a linear combiner. The output of the network is the weighted sum of the hidden layer neuron outputs:

$$y(\mathbf{x}) = \sum_{j=0}^{m} w_j z_j(\mathbf{x})$$



13 ANN – Training RBF NN

If Gaussian basis function is used, we have:

$$y(\mathbf{x}) = \sum_{j=0}^{m} w_j z_j(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \exp\left(-\frac{\|\mathbf{c}_j - \mathbf{x}\|^2}{2\sigma^2} \right)$$

The parameters that determine the RBF are:

(1) The center vectors $\mathbf{c}_j$, j=1,2, …, m.

(2) The width of the basis function $2\sigma^2$.

(3) The weights w_j, j=1,2, …, m.

13 ANN – Training RBF NN

An important point here is that the different layers of an RBF network perform different tasks, and it is reasonable to separate the optimization of the hidden layer and the output layer by using different techniques. Based on this idea, the training of the RBF is usually done using a two-step procedure:

(1) In the first step, the RBF centers and widths are determined.

(2) In the second step, the weights are estimated using some algorithm, with the goal of minimizing the difference between the desired output and the actual output of the RBF neural network.

13 ANN – Training RBF NN

It is noted that the output of the RBF neural network has a linear relationship with the output of hidden layer neurons, thus the estimation of weights in the second step can be easily done using a linear least square estimation algorithm.

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \exp\left(-\frac{\|\mathbf{c}_j - \mathbf{x}\|^2}{2\sigma^2}\right) = \sum_{j=0}^{m} w_j z_j(\mathbf{x}) = \mathbf{w}^T \mathbf{z}$$

If we have a target output t, the error of the output is:

$$e = t - y = t - \mathbf{w}^T \mathbf{z}$$

The mean square error is:

$$J(\mathbf{w}) = E\{e^2\} = E\{(t-y)^2\} = E\{(t - \mathbf{w}^T \mathbf{z})^2\}$$

13 ANN – Training RBF NN

- The optimal weights are obtained by minimizing the mean square error. This leads to the least mean square solution of W:

$$\frac{\partial E\{e^2\}}{\partial \mathbf{w}} = \frac{\partial E\{(t-\mathbf{w}^T\mathbf{z})^2\}}{\partial \mathbf{w}} =$$

$$\frac{\partial (t^2 - 2\mathbf{w}^T E\{\mathbf{z}t\} + \mathbf{w}^T E\{\mathbf{z}\mathbf{z}^T\}\mathbf{w})}{\partial \mathbf{w}} = 0$$

$$\Rightarrow E\{\mathbf{z}\mathbf{z}^T\}\mathbf{w} = E\{\mathbf{z}t\}$$

- Suppose we have q samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q$, which produce q output: $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_q$

$$E\{\mathbf{z}\mathbf{z}^T\} \cong \frac{1}{q}\sum_{i=1}^{q}\mathbf{z}_i\mathbf{z}_i^T, \quad E\{\mathbf{z}t\} \cong \frac{1}{q}\sum_{i=1}^{q}\mathbf{z}_i t_i$$

13 ANN – Training RBF NN

- Let

$$\mathbf{Z} = [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_L], \quad \mathbf{t} = [t_1 \quad t_2 \quad \cdots \quad t_q]$$

- Then:

$$q \cdot E\{\mathbf{z}\mathbf{z}^T\} \cong \sum_{i=1}^{q} \mathbf{z}_i \mathbf{z}_i^T = \mathbf{Z}\mathbf{Z}^T$$

$$q \cdot E\{\mathbf{z}t\} \cong \sum_{i=1}^{q} \mathbf{z}_i t_i = \mathbf{Z}\mathbf{t}^T$$

$$E\{\mathbf{z}\mathbf{z}^T\}W = E\{\mathbf{z}t\}$$

$$\approx \Downarrow$$

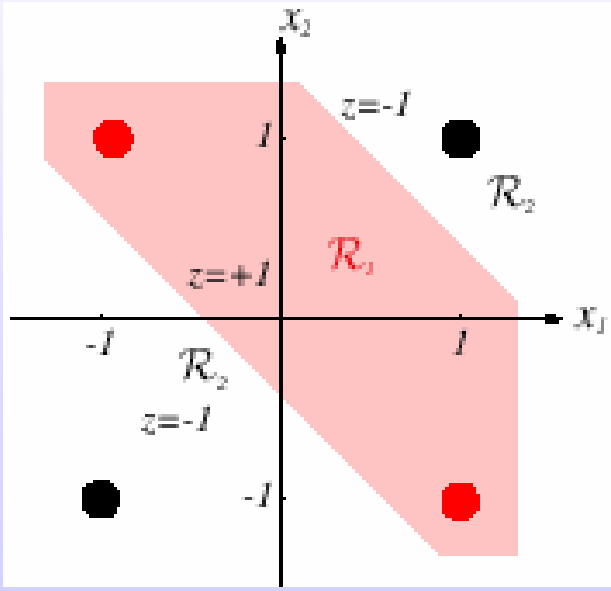
$$\mathbf{Z}\mathbf{Z}^T W = \mathbf{Z}\mathbf{t}^T \quad \Rightarrow \quad W = (\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{t}^T$$

- It is also called least square method.

13 ANN – Example of Training RBF NN

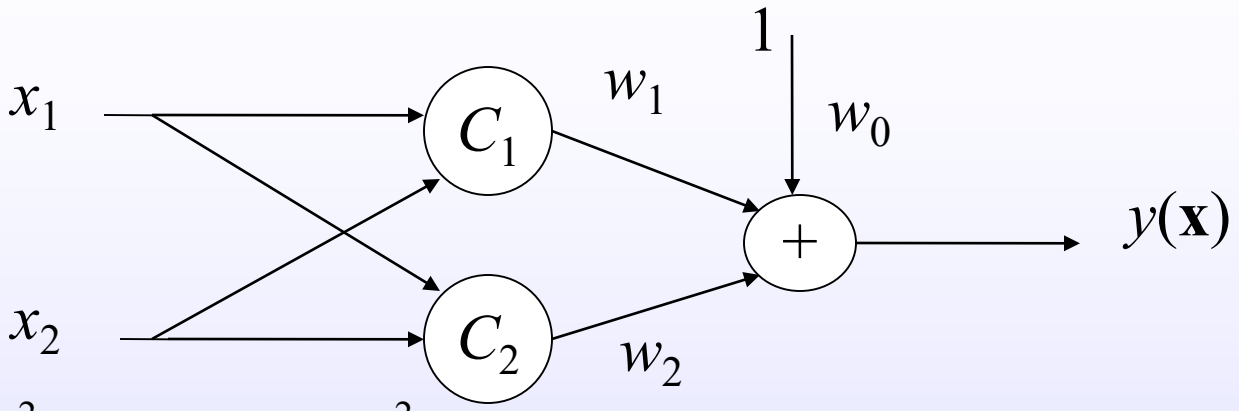
Consider the exclusive OR (XOR) problem again. The input and output of the logic operator are as follows:

Index of the data	inputs	target outputs
1	$[1\ 1]^T$	0
2	$[0\ 1]^T$	1
3	$[0\ 0]^T$	0
4	$[1\ 0]^T$	1



Assume two hidden layer neurons are used, the centers of two neurons are set to: $\mathbf{c}_1=[1\ 1]^T$, $\mathbf{c}_2=[0\ 0]^T$, and the width of the Gaussian basis function is set to 1.

13 ANN – Example of Training RBF NN



$$y(\mathbf{x}) = \sum_{i=1}^{2} w_i z_i + w_0 = \sum_{i=1}^{2} w_i \exp(-\| \mathbf{x} - \mathbf{c}_i \|^2) + w_0 = \mathbf{w}^T \mathbf{z}$$

Substituting all the 4 data points to the above equation, we obtain:

$$z_1 = \exp\left[-(\mathbf{x}-\mathbf{c}_1)^T(\mathbf{x}-\mathbf{c}_1)\right] = \{1.0000,\ 0.3679,\ 0.1353,\ 0.3679\}$$

$$z_2 = \exp\left[-(\mathbf{x}-\mathbf{c}_2)^T(\mathbf{x}-\mathbf{c}_2)\right] = \{0.1353,\ 0.3679,\ 1.0000,\ 0.3679\}$$

$$\mathbf{Z} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1.0000 & 0.3679 & 0.1353 & 0.3679 \\ 0.1353 & 0.3679 & 1.0000 & 0.3679 \end{pmatrix}$$

13 ANN – Example of Training RBF NN

$$\mathbf{Z}\mathbf{Z}^T =$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1.0000 & 0.3679 & 0.1353 & 0.3679 \\ 0.1353 & 0.3679 & 1.0000 & 0.3679 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1.0000 & 0.3679 & 0.1353 & 0.3679 \\ 0.1353 & 0.3679 & 1.0000 & 0.3679 \end{pmatrix}^T$$

$$= \begin{pmatrix} 4.0000 & 1.8711 & 1.8711 \\ 1.8711 & 1.2890 & 0.5413 \\ 1.8711 & 0.5413 & 1.2890 \end{pmatrix}$$

$$\mathbf{Z}\mathbf{t}^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1.0000 & 0.3679 & 0.1353 & 0.3679 \\ 0.1353 & 0.3679 & 1.0000 & 0.3679 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2.0000 \\ 0.7358 \\ 0.7358 \end{pmatrix}$$

13 ANN – Example of Training RBF NN

$$\mathbf{Z}\mathbf{Z}^T\mathbf{w} = \mathbf{Z}\mathbf{t}^T \quad \Rightarrow \quad \mathbf{w} = (\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{t}^T$$

$$\Downarrow$$

$$\begin{pmatrix} 4.0000 & 1.8711 & 1.8711 \\ 1.8711 & 1.2890 & 0.5413 \\ 1.8711 & 0.5413 & 1.2890 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 2.0000 \\ 0.7358 \\ 0.7358 \end{pmatrix}$$

Solving the above matrix equation, we obtain:

$$\mathbf{w} = \begin{pmatrix} 2.8413 \\ -2.5027 \\ -2.5027 \end{pmatrix}$$

13 ANN – Example of Training RBF NN

Let's check the performance of the network next.

$$\mathbf{Y} = \mathbf{w}^T \mathbf{Z} = (0.0000 \quad 1.0000 \quad 0.0000 \quad 1.0000)$$

It means:

If $\mathbf{x}=[\,1\ 1\,]^T$, we have $y(\mathbf{x})=0.0000$

If $\mathbf{x}=[\,0\ 1\,]^T$, we have $y(\mathbf{x})=1.0000$

If $\mathbf{x}=[\,0\ 0\,]^T$, we have $y(\mathbf{x})=0.0000$

If $\mathbf{x}=[\,1\ 0\,]^T$, we have $y(\mathbf{x})=1.0000$

Obviously the network with just two neurons approximates the XOR logic operation perfectly well.

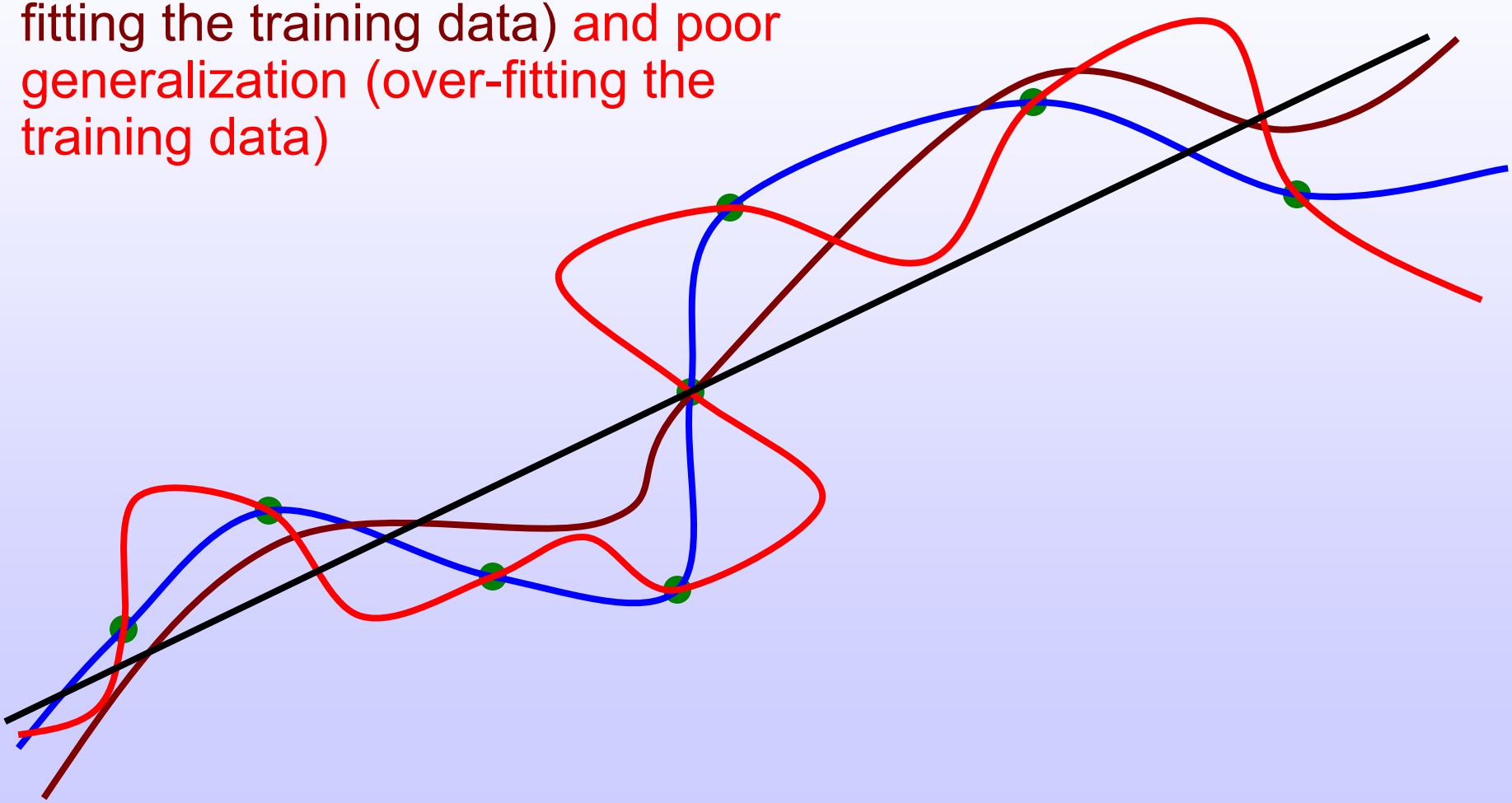
13 ANN – Strategies for neuron center selection

Depending on how RBF neuron centers are determined, there are a few strategies that we can follow in the design of an RBF network, such as

- Random selection from training samples
- k-means clustering algorithm
- Other clustering algorithm

13 ANN – Understand Under- and Over-fitting

Problems of local minima (under-fitting the training data) and poor generalization (over-fitting the training data)



13 ANN – Conclusions of Neural Networks

- Neural networks are Universal Approximators

It has been shown that any nonlinear continuous function can be approximated arbitrarily close, both, by a two layer perceptron, with sigmoid activations, and an RBF network, provided a large enough number of nodes are used.

However, these results are of greater theoretical interest than practical, since the construction of such a network requires the nonlinear functions and the weight values which are unknown! It is still an open question how to find the nonlinear functions based on that training data. Problems of local minima (under-fitting the training data) and poor generalization (over-fitting the training data) are central issues of pattern recognition and machine learning.