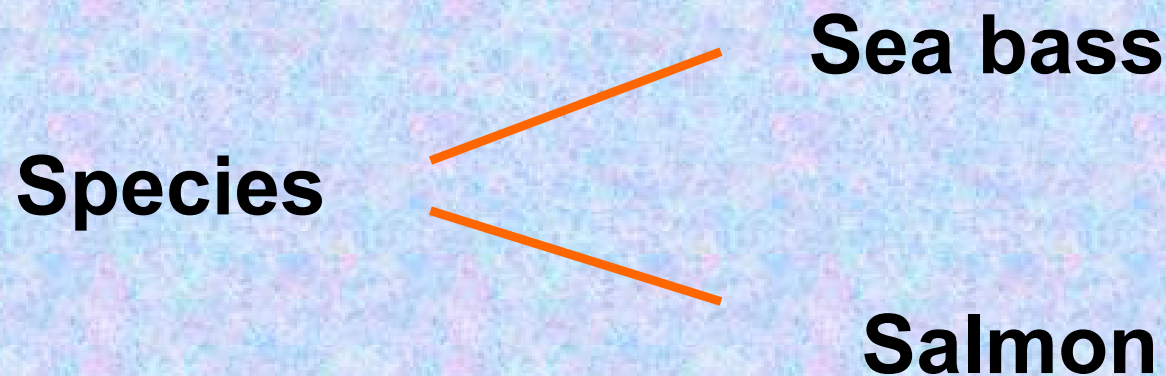


**Pattern  
Classification**

# An Example of Classification

- “Sorting incoming Fish on a conveyor according to species using optical sensing



– **Some properties that could be possibly used to distinguish between the two types of fishes is**

- **Length**
- **Lightness**
- **Width**
- **Number and shape of fins**
- **Position of the mouth, etc...**



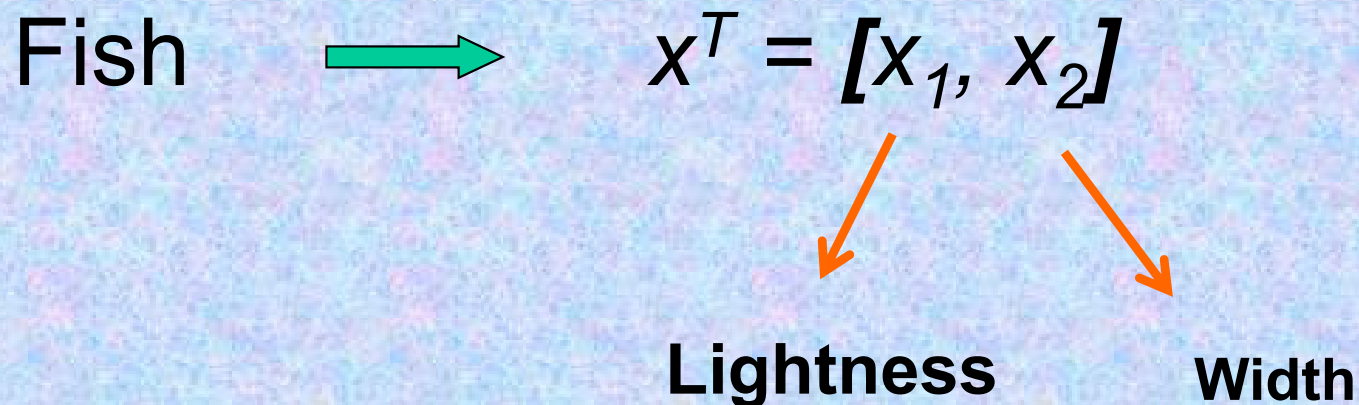
**Features**

– **This is the set of all suggested features to explore for use in our classifier!**

**Feature is a property (or characteristics) of an object (quantifiable or non quantifiable) which is used to distinguish between (or classify) two objects.**

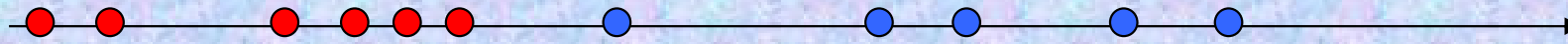
# Feature vector

- A Single feature may not be useful always for classification
- A set of features used for classification form a **feature vector**



# Feature space

- The samples of input (when represented by their features) are represented as points in the **feature space**
- If a single feature is used, then work on a one- dimensional feature space.

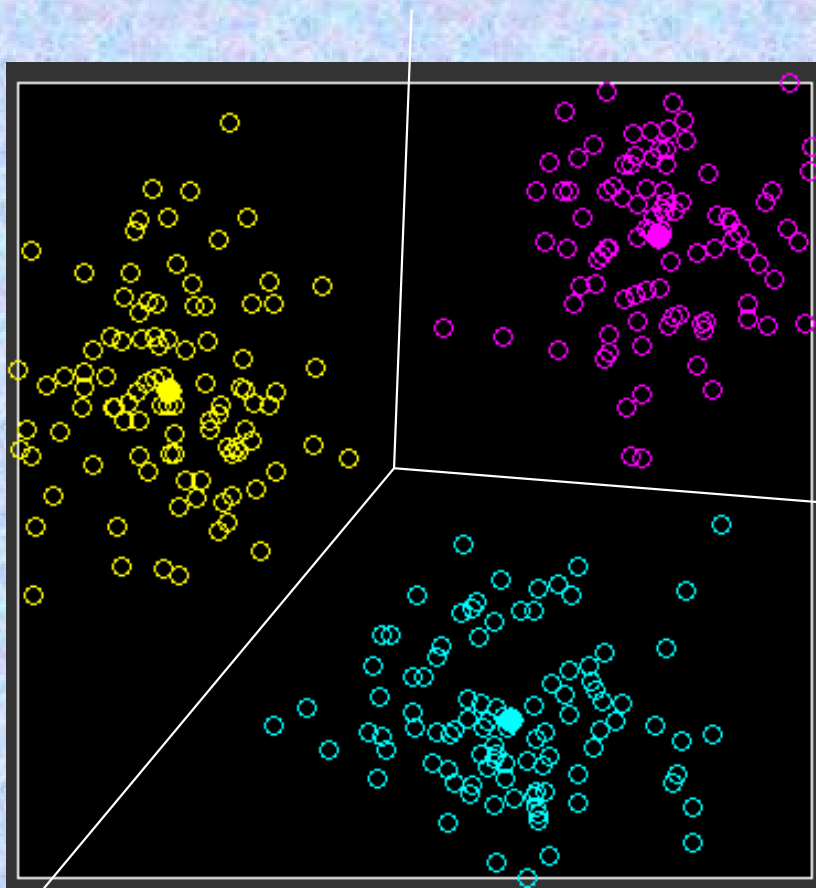


Point representing samples

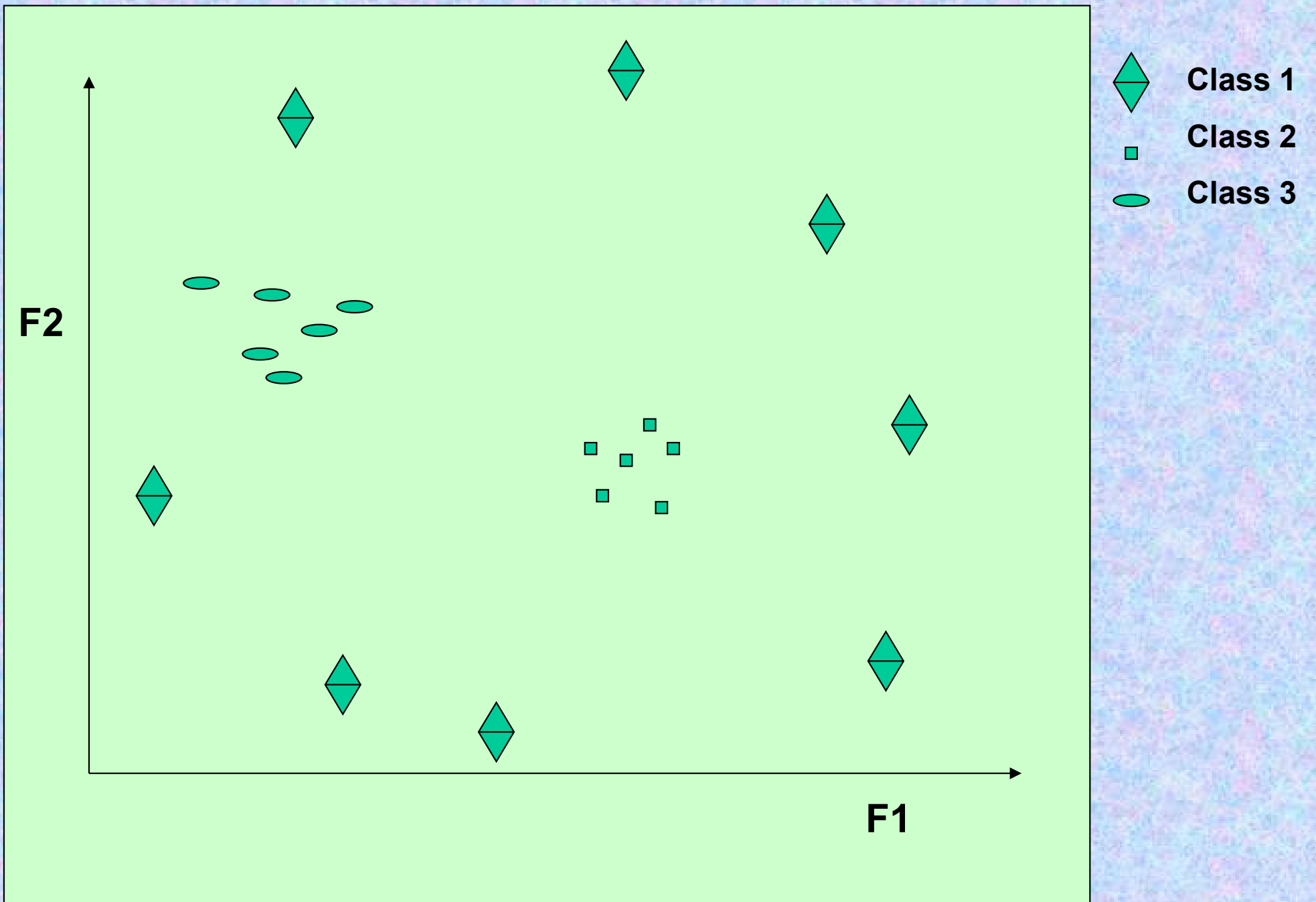
- If number of features is 2, then we get points in 2D- space as shown in the next slide.
- We can also have an n-dimensional feature space



**Decision boundary in one-dimensional case with two classes.**



**Decision boundary in 2 (or 3) dimensional case with three classes**



**Sample points in a two-dimensional feature space**

## Some Terminologies:

- **Pattern**
- **Feature**
- **Feature vector**
- **Feature space**
- **Classification**
- **Decision Boundary**
- **Decision Region**
- **Discriminant function**
- **Hyperplanes and Hypersurfaces**
- **Learning**
- **Supervised and unsupervised**
- **Error**
- **Noise**
- **PDF**
- **Baye's Rule**
- **Parametric and Non-parametric approaches**



# Decision region and Decision Boundary

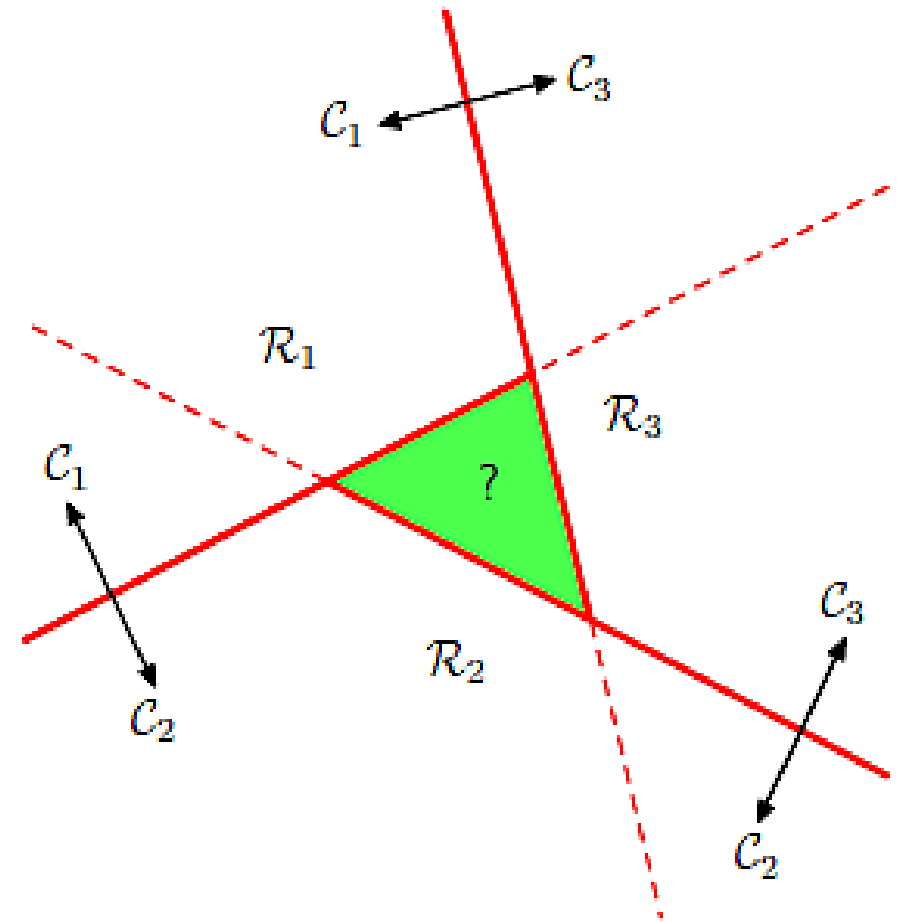
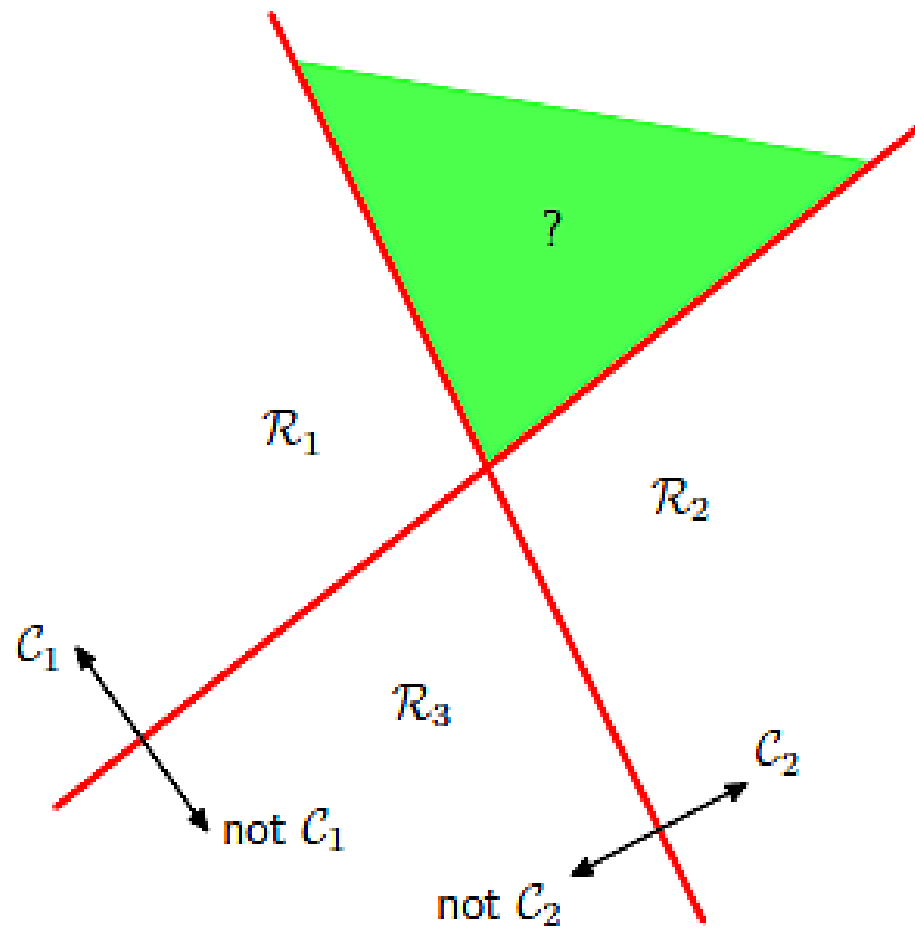
- Our goal of pattern recognition is to reach an optimal **decision rule** to categorize the incoming data into their respective categories
- The **decision boundary** separates points belonging to one class from points of other
- The decision boundary partitions the feature space into **decision regions**.
- The nature of the decision boundary is decided by the **discriminant function** which is used for decision. It is a function of the feature vector.

## *Multiple classes*

**Now consider the extension of linear discriminants to  $K > 2$  classes. We might be tempted to build a  $K$ -class discriminant by combining a number of two-class discriminant functions. However, this leads to some serious difficulties (Duda and Hart, 1973).**

**Consider the use of  $K-1$  classifiers each of which solves a two-class problem of separating points in a particular class  $C_k$  from points not in that class. This is known as a one-versus-the-rest classifier.**

**An illustration only follows; solutions follow later.**



**Figure 4.2** Attempting to construct a  $K$  class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class  $C_k$  from points not in class  $C_k$ . On the right is an example involving three discriminant functions each of which is used to separate a pair of classes  $C_k$  and  $C_j$ .

# Hyper planes and Hyper surfaces

- For two category case, a positive value of discriminant function decides class 1 and a negative value decides the other.
- If the number of dimensions is three. Then the decision boundary will be a **plane** or a 3-D surface. The decision regions become **semi-infinite volumes**
- If the number of dimensions increases to more than three, then the decision boundary becomes a **hyper-plane** or a **hyper-surface**. The decision regions become semi-infinite hyperspaces.

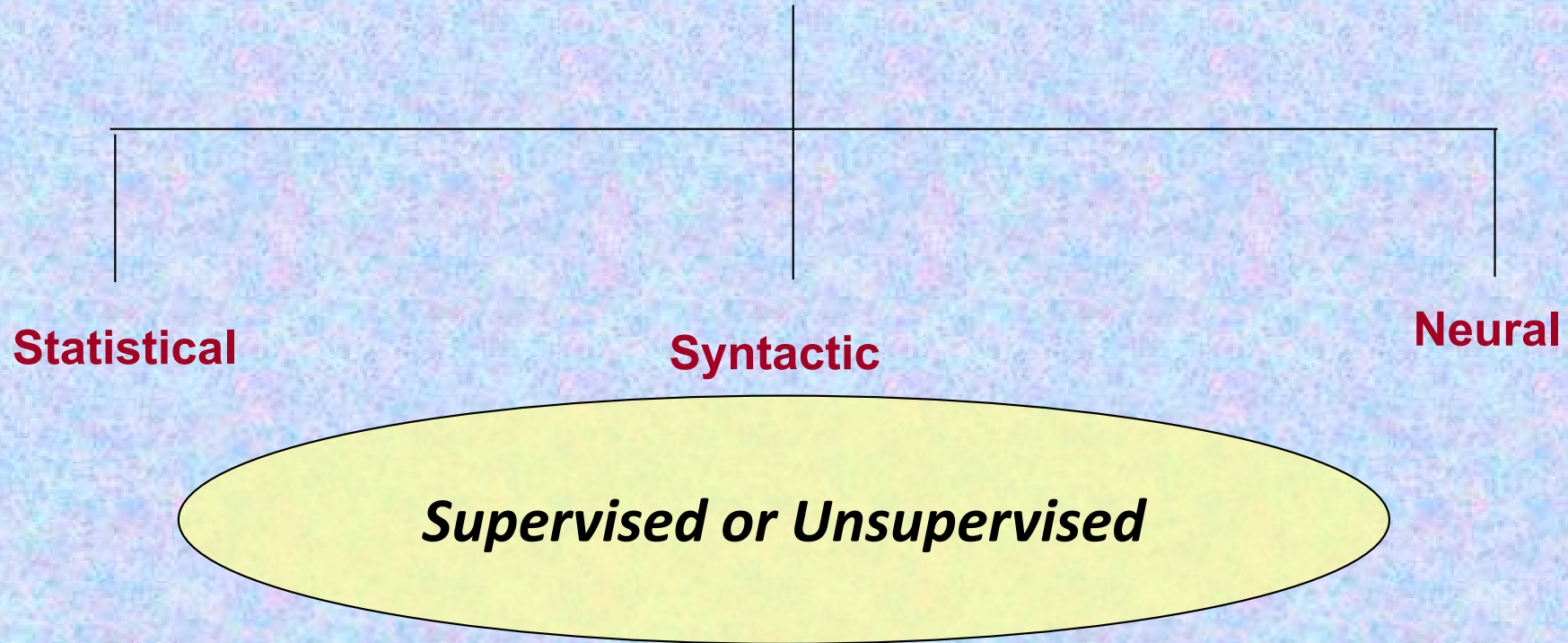
# Learning

- The classifier to be designed is built using input samples which is a mixture of all the classes.
- The classifier **learns** how to discriminate between samples of different classes.
- If the **Learning** is offline i.e. **Supervised** method then, the classifier is first given a set of training samples and the optimal decision boundary found, and then the classification is done.
- If the learning is online then there is no teacher and no training samples (**Unsupervised**). The input samples are the test samples itself. The classifier learns and classifies at the same time.

# Error

- **The accuracy of classification depends on two things**
  - **The *optimality of decision rule* used:** The central task is to find an optimal decision rules which can generalize to unseen samples as well as categorize the training samples as correctly as possible. This decision theory leads to a *minimum error-rate classification*.
  - **The *accuracy in measurements* of feature vectors:** This inaccuracy is because of presence of *noise*. Hence our classifier should deal with noisy and missing features too.

# Classifier Types



## Categories of Statistical Classifiers:

- **Linear**
- **Quadratic**
- **Piecewise**
- **Non-parametric**

# Parametric Decision making (Statistical) - Supervised

Goal of most classification procedures is to estimate the probabilities that a pattern to be classified belongs to various possible classes, based on the values of some feature or set of features.

In most cases, we decide which is the most likely class. We need a mathematical decision making algorithm, to obtain classification.

## Bayesian decision making or Bayes Theorem

This method refers to choosing the most likely class, given the value of the feature/s. Bayes theorem calculates the probability of class membership.

Define:

$P(w_i)$  - **Prior Prob.** for class  $w_i$ ;       $P(X)$  - **Prob. (Uncondl.)** for feature vector  $X$ .

$P(w_i | X)$  - **Measured-conditioned or posteriori probability**

$P(X | w_i)$  - **Prob. (Class-Condnl.)** Of feature vector  $X$  in class  $w_i$



## Bayes Theorem:

$$P(w_i | \vec{X}) = \frac{P(\vec{X} | w_i)P(w_i)}{P(\vec{X})}$$

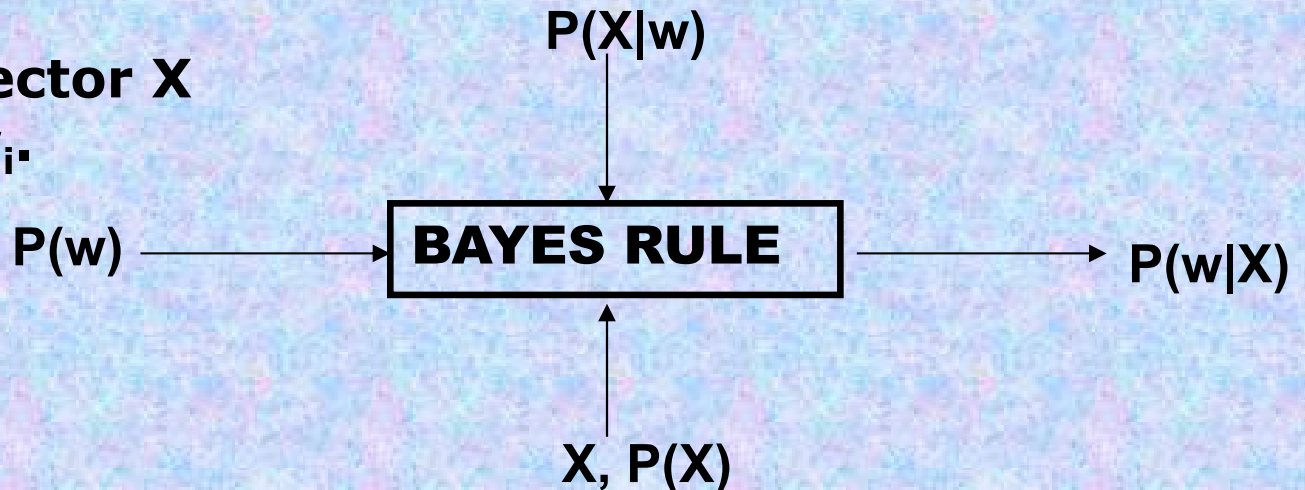
$P(\vec{X})$  is the probability distribution for feature  $\vec{X}$  in the entire population. Also called unconditional density function (or evidence).

$P(w_i)$  is the prior probability that a random sample is a member of the class  $C_i$ .

$P(\vec{X} | w_i)$  is the class conditional probability (or likelihood) of obtaining feature value  $\vec{X}$  given that the sample is from class  $w_i$ . It is equal to the number of times (occurrences) of  $\vec{X}$ , if it belongs to class  $w_i$ .

The goal is to measure:  $P(w_i | \vec{X})$  – Measured-conditioned or posteriori probability, from the above three values.

This is the Prob. of any vector  $\vec{X}$  being assigned to class  $w_i$ .



**Take an example:**

**Two class problem:**

**Cold (C) and not-cold (C'). Feature is fever (f).**

**Prior probability of a person having a cold,  $P(C) = 0.01$ .**

**Prob. of having a fever, given that a person has a cold is,  $P(f|C) = 0.4$ . Overall prob. of fever  $P(f) = 0.02$ .**

**Then using Bayes Th., the Prob. that a person has a cold, given that she (or he) has a fever is:**

$$P(C | f) = \frac{P(f | C)P(C)}{P(f)} = \frac{0.4 * 0.01}{0.02} = 0.2$$

**Not convinced that it works?**

**let us take an example with values to verify:**

**Total Population = 1000. Thus, people having cold = 10. People having both fever and cold = 4. Thus, people having only cold = 10 - 4 = 6.**

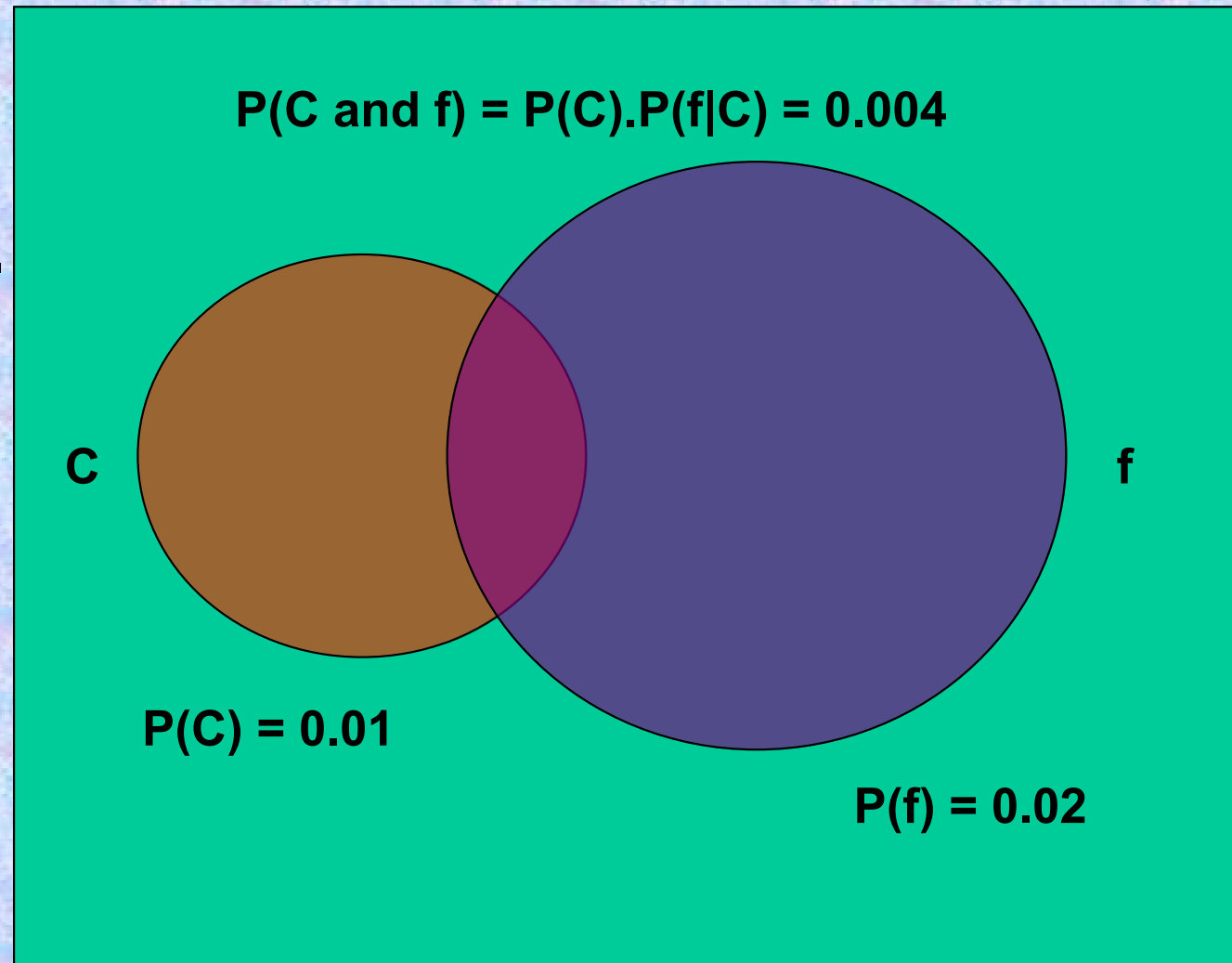
**People having fever (with and without cold) = 0.02 \* 1000 = 20.**

**People having fever without cold = 20 - 4 = 16 (may use this later).**

**So, probability (percentage) of people having cold along with fever, out of all those having fever, is:  $4/20 = 0.2$  (20%).**

***IT WORKS, GREAT***

**A Venn diagram,  
illustrating the  
two class,  
one feature problem.**



**Probability of a joint event - a sample comes from class C and has the feature value X:**

$$\begin{aligned} P(C \text{ and } X) &= P(C).P(X|C) = P(X).P(C|X) \\ &= 0.01*0.4 = 0.02*0.2 \end{aligned}$$

**Also verify, for a K class problem:**

$$P(X) = P(w_1)P(X|w_1) + P(w_2)P(X|w_2) + \dots + P(w_k)P(X|w_k)$$

**Thus:**

$$P(w_i | \vec{X}) = \frac{P(\vec{X} | w_i)P(w_i)}{P(w_1)P(X | w_1) + P(w_2)P(X | w_2) + \dots + P(w_k)P(X | w_k)}$$

**With our last example:**

$$P(f) = P(C)P(f|C) + P(C')P(f|C')$$

$$= 0.01 * 0.4 + 0.99 * 0.01616 = 0.02$$

**Decision or Classification algorithm according to Baye's Theorem:**

$$\text{Choose } \begin{cases} w_1; & \text{if } p(X | w_1)p(w_1) > p(X | w_2)p(w_2) \\ w_2; & \text{if } p(X | w_2)p(w_2) > p(X | w_1)p(w_1) \end{cases}$$

## Errors in decision making:

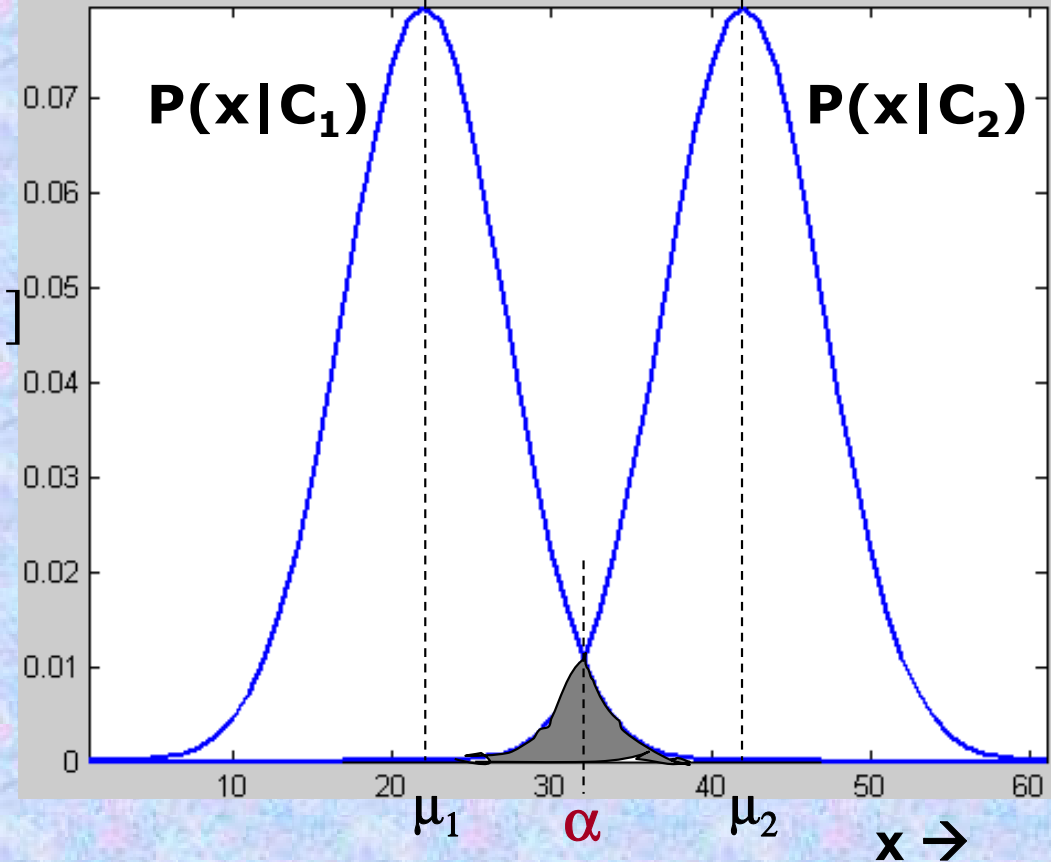
Let  $d = 1$ ,  $C = 2$ ,  
 $P(C_1) = P(C_2) = K$ ;

$$p(x | C_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x - \mu_i}{\sigma}\right)^2\right]$$

**Bayes decision rule:**

**Choose  $C_1$ , if  $P(x|C_1) > P(x|C_2)$**

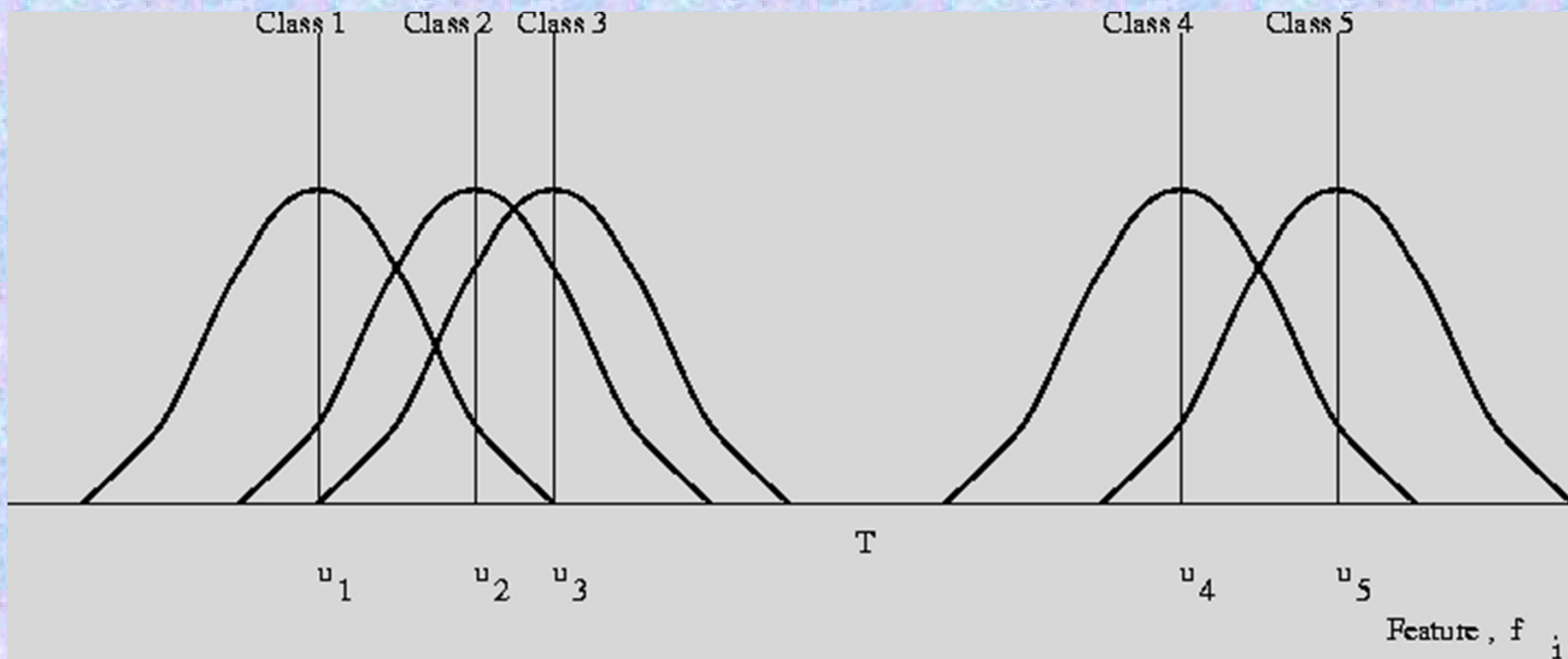
**This gives  $\alpha$ , and hence the two decision regions.**



**Classification error (the shaded region – minimum of the two curves):**

**$P(E) = P(\text{Chosen } C_1, \text{ when } x \text{ belongs to } C_2) +$   
 **$P(\text{Chosen } C_2, \text{ when } x \text{ belongs to } C_1)$****

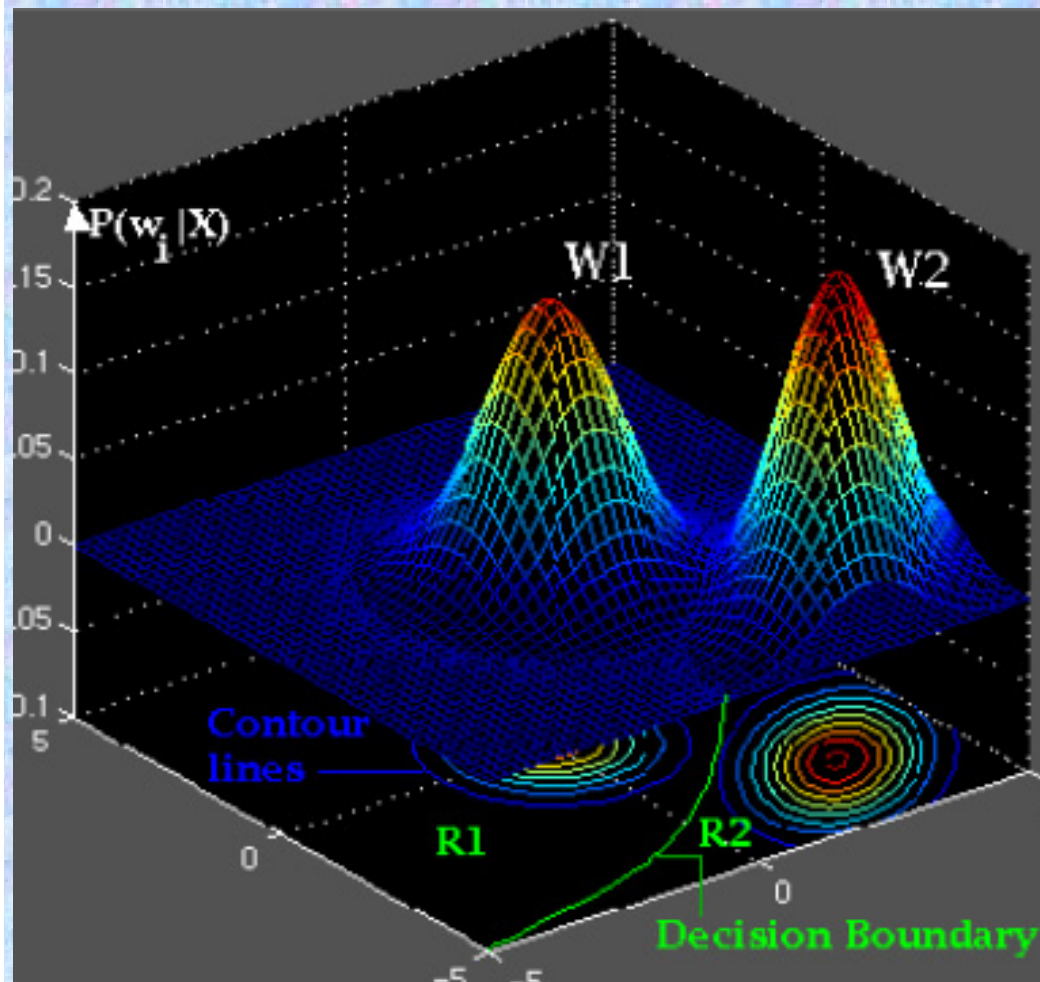
$$= P(C_2) \int_{-\infty}^{\alpha} P(\gamma | C_2) d\gamma + P(C_1) \int_{\alpha}^{\infty} P(\gamma | C_1) d\gamma$$



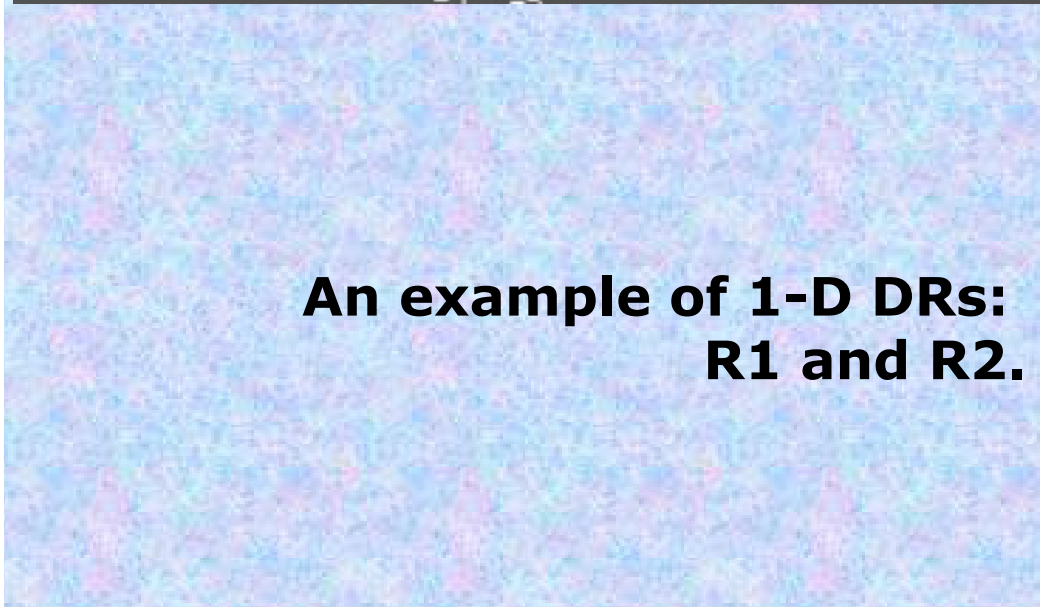
Normal distributions of feature measurement for a 5-class problem, equal variance.

## A minimum distance (NN) supervised classifier

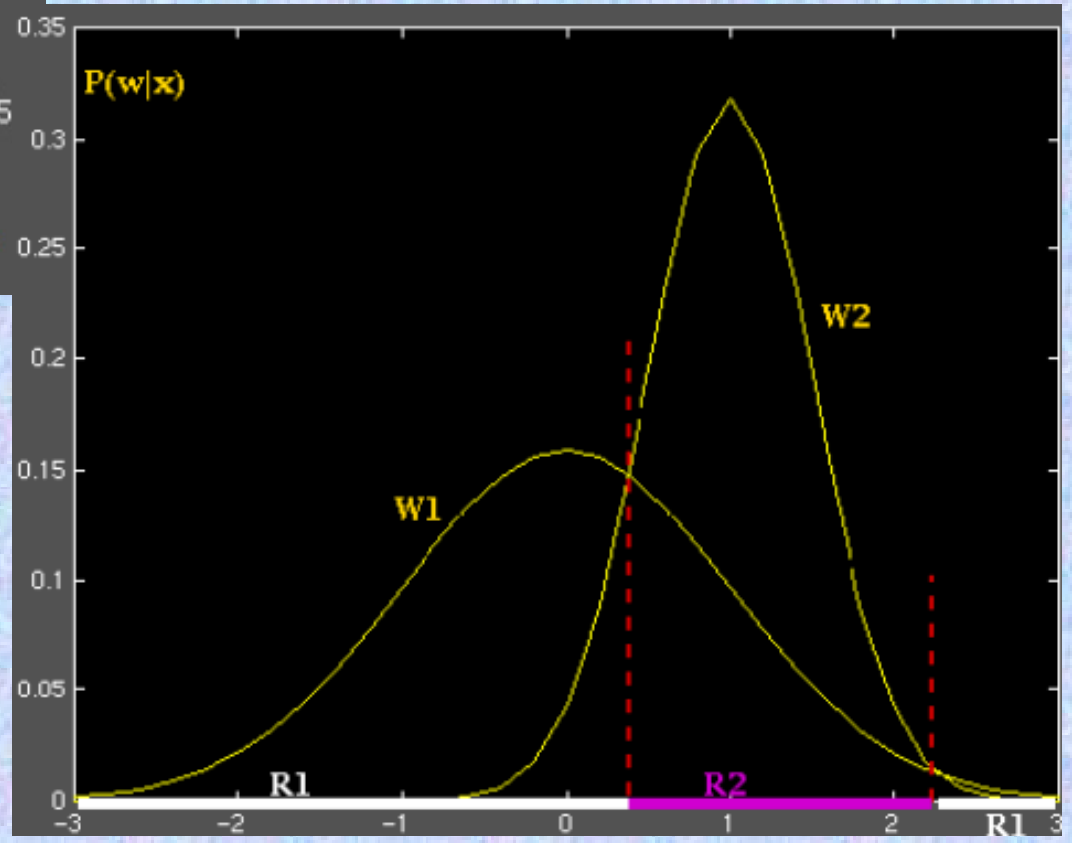
**Rule: Assign  $X$  to  $R_i$ , where  $X$  is closest to  $\mu_i$ .**

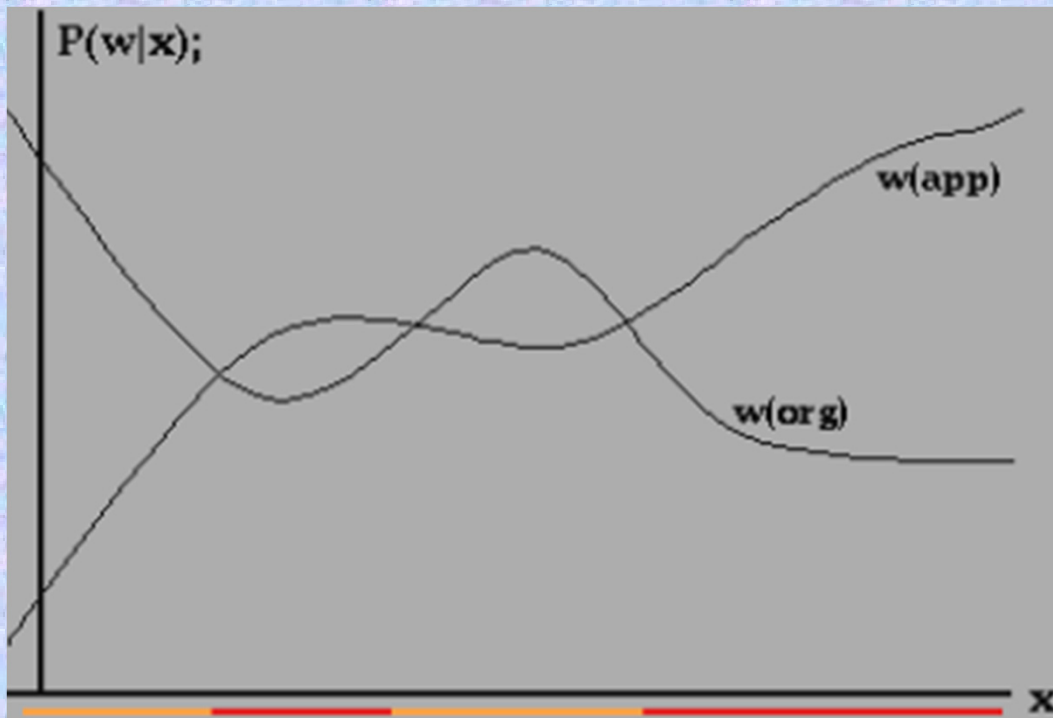


**An example of 2-D DRs:  
R1 and R2; with a non-linear DB.**



**An example of 1-D DRs:  
R1 and R2.**





Decision based on  
**arbitrary Posteriors**,  
 for an example:  
 Apples  
 Vs. Oranges.

Commonly used Discriminant functions  
 based on Baye's decision rule:

$$g_i(x) = P(w_i | x)$$

$$g_i(x) = \frac{p(x|w_i)P(w_i)}{\sum_{j=1}^c p(x|w_j)P(w_j)}$$

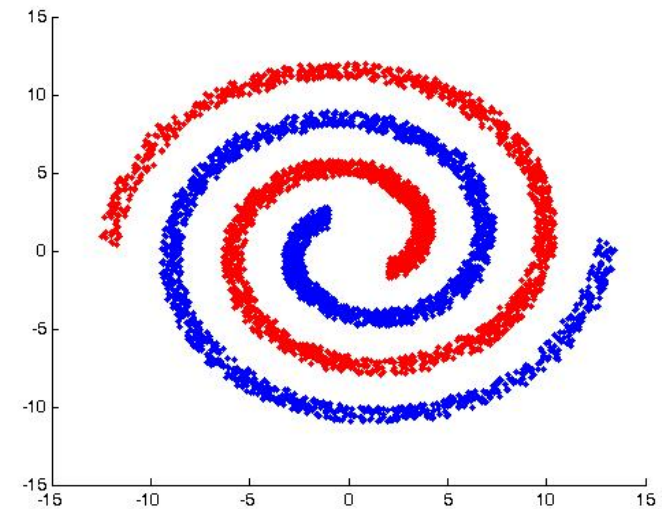
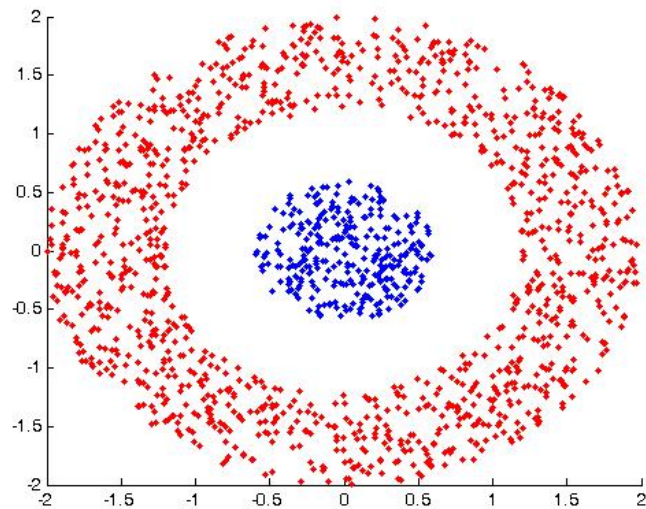
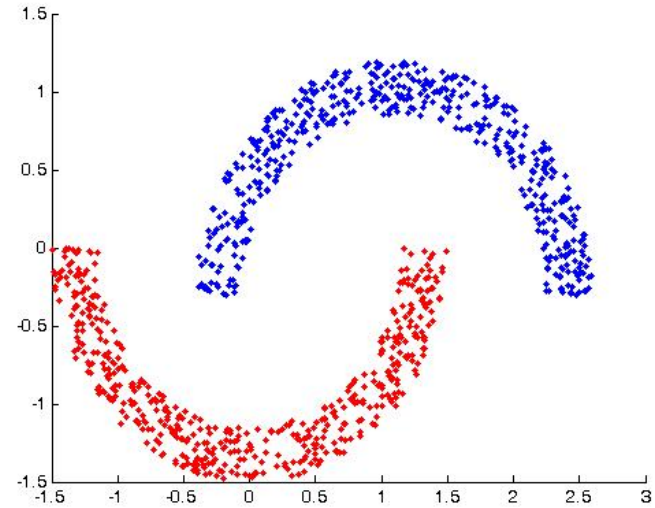
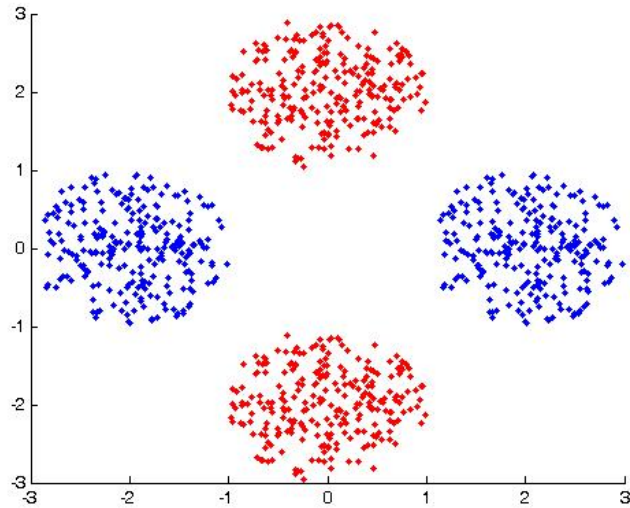
$$g_i(x) = p(x|w_i)P(w_i)$$

$$g_i(x) = \ln p(x|w_i) + \ln P(w_i)$$

**Naïve Bayes** – left for self-study:  
 Ref: Bishop – Sec 4.2.3 , pp -202..



# Some examples of dense distribution of instances, with non-linear decision boundaries



# K-means Clustering (unsupervised)

- Given a fixed number of **k clusters**, assign observations to those clusters so that the means across clusters for all variables are as different from each other as possible.
- **Input**
  - Number of Clusters,  $k$
  - Collection of  $n$ ,  $d$  dimensional vectors  $x_j$ ,  $j=1, 2, \dots, n$
- **Goal:** find the  $k$  mean vectors  $\mu_1, \mu_2, \dots, \mu_k$
- **Output**
  - $k \times n$  binary membership matrix  $U$  where

$$u_{ij} = \begin{cases} 1 & \text{if } x_i \in G_j \\ 0 & \text{else} \end{cases}$$

&  $G_j$ ,  $j=1, 2, \dots, k$  represent the  $k$  clusters

If  $n$  is the number of known patterns and  $c$  the desired number of clusters, the k-means algorithm is:

Begin

initialize  ~~$n, \epsilon$~~ ,  $\mu_1, \mu_2, \dots, \mu_c$  (randomly selected)

do

1. classify  $n$  samples according to nearest  $\mu_i$

2. recompute  $\mu_i$

until no change in  $\mu_i$

return  $\mu_1, \mu_2, \dots, \mu_c$

End

**Distance measures used:**

**Euclidean, Manhattan, Minkowski, Canberra, Hamming, Chessboard/Checkerboard & Maximum (*Chebychev*), Cosine-Sim, Spearman-correlation.**

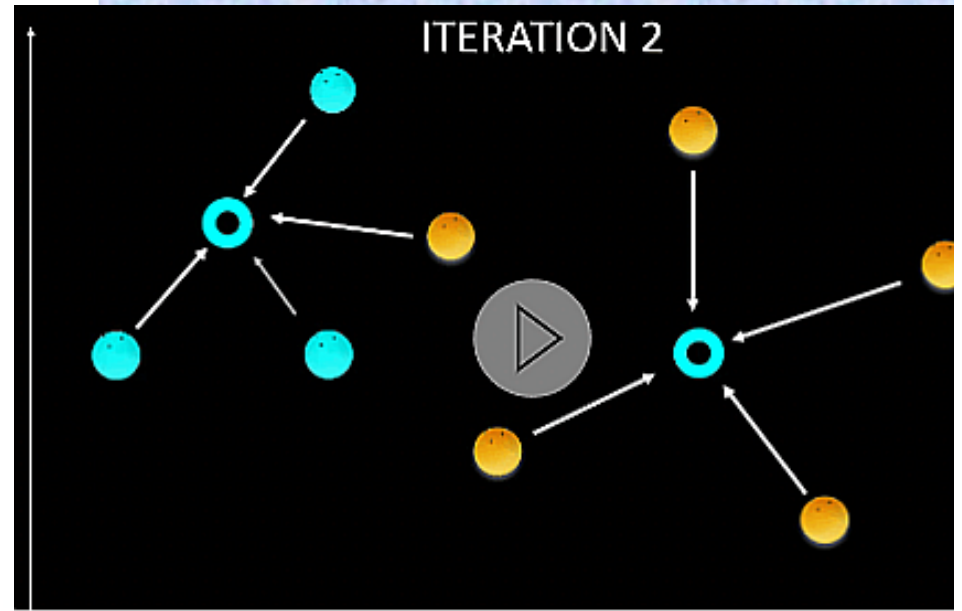
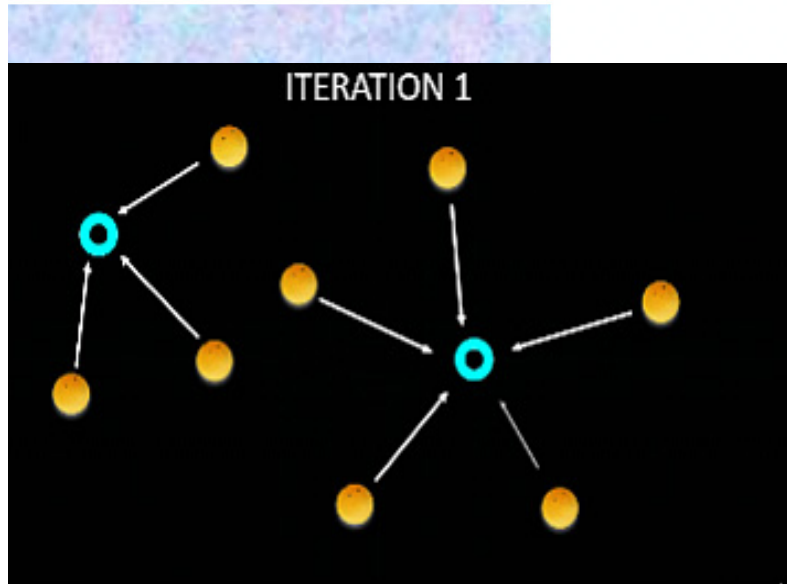
# Classification Stage

- The samples have to be assigned to clusters in order to **minimize the cost function** which is:

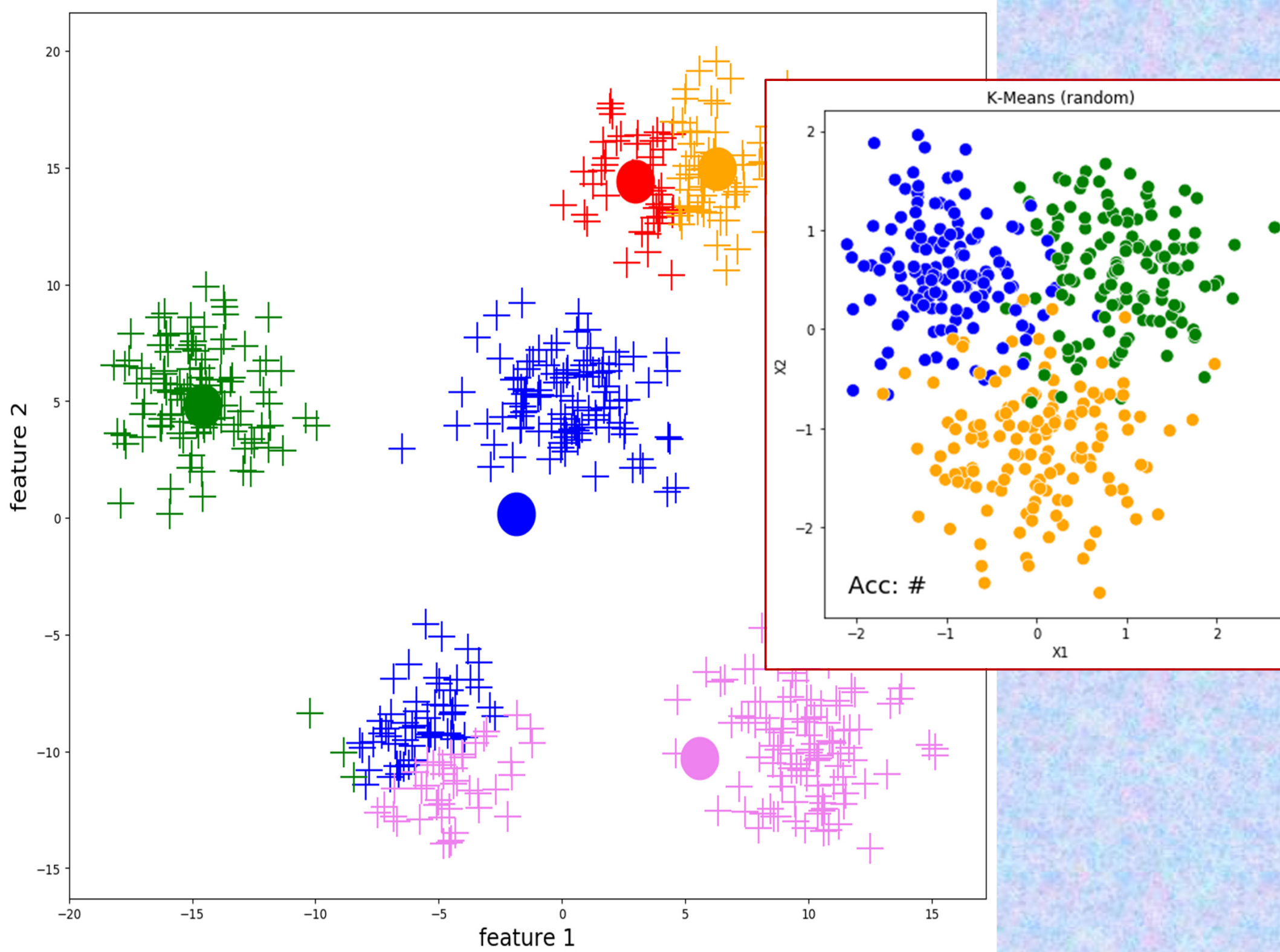
$$J = \sum_{i=1}^c J_i = \sum_{i=1}^c \left[ \sum_{k, x_k \in G_i} \|x_k - \mu_i\|^2 \right]$$

- This is the **Euclidian Distance** of the samples from its cluster center; for all clusters this sum should be minimum
- The classification of a point  $x_k$  is done by:

$$u_i = \begin{cases} 1 & \text{if } \|x_k - \mu_i\|^2 \geq \|x_k - \mu_j\|^2, \forall k \neq i \\ 0 & \text{otherwise} \end{cases}$$











# Re-computing the Means

- The means are recomputed according to:

$$\mu_i = \frac{1}{|G_i|} \left( \sum_{k, x_k \in G_i} x_k \right)$$

- Disadvantages
  - What happens when there is overlap between classes... that is a **point is equally close to two cluster centers**..... Algorithm will not terminate
  - The Terminating condition is modified to “Change in cost function (computed at the end of the Classification) is below some threshold rather than 0”.



**Back to study of**  
**Decision Boundaries (DBs)**  
**under Bayes paradigm;**  
**Supervised Statistical Classifiers**

**Normal Density:**

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

**Bivariate Normal Density:**

$$p(x, y) = \frac{e^{-\frac{1}{2(1-\rho_{xy}^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - \frac{2\rho_{xy}(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]}}{2\pi\sigma_x\sigma_y\sqrt{(1-\rho_{xy}^2)}}$$

$\mu$  - Mean;  $\sigma$  - S.D.;  $\rho_{xy}$  - Correlation Coefficient

Visualize  $\rho$  as equivalent to the orientation of a 2-D tilted Gaussian or Gabor filter.

For  $x$  as a discrete random variable, the expected value of  $x$ :

$$E(x) = \sum_{i=1}^n x_i P(x_i) = \mu_x$$

$E(x)$  is also called the first moment of the distribution.

The  $k^{\text{th}}$  moment is defined as:

$$E(x^k) = \sum_{i=1}^n x_i^k P(x_i)$$

$P(x_i)$  is the probability of  $x = x_i$ .

**Multi-variate Case:**  $X = [x_1 \ x_2 \ \dots \ x_d]^T$

Mean vector:

$$\mu = E(X) =$$

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdot \\ \cdot \\ \mu_d \end{bmatrix}$$

Covariance matrix (symmetric):

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdot & \cdot & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{d1} & \sigma_{d2} & \cdot & \cdot & \sigma_{dd} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdot & \cdot & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{1d} & \sigma_{2d} & \cdot & \cdot & \sigma_d^2 \end{bmatrix}$$

**d-dimensional normal density is:**

$$p(X) = \frac{1}{\sqrt{\det(\Sigma)}(2\pi)^d} \exp\left[-\frac{(X - \mu)^T \Sigma^{-1} (X - \mu)}{2}\right]$$
$$= \frac{1}{\sqrt{\det(\Sigma)}(2\pi)^d} \exp\left[-\frac{1}{2} \sum_{ij} (x_i - \mu_i) s_{ij} (x_j - \mu_j)\right]$$

$$p(\mathbf{X}) = \frac{1}{\sqrt{\det(\Sigma)(2\pi)^d}} \exp\left[-\frac{(\mathbf{X} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu})}{2}\right]$$

$$= \frac{1}{\sqrt{\det(\Sigma)(2\pi)^d}} \exp\left[-\frac{1}{2} \sum_{ij} (x_i - \mu_i) s_{ij} (x_j - \mu_j)\right]$$

where,  $s_{ij}$  is the  $i$ - $j$ <sup>th</sup> component of  $\Sigma^{-1}$  (the inverse of covariance matrix  $\Sigma$ ).

Special case,  $d = 2$ ; where  $\mathbf{X} = (x \ y)^T$ ;      Then:  $\boldsymbol{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$

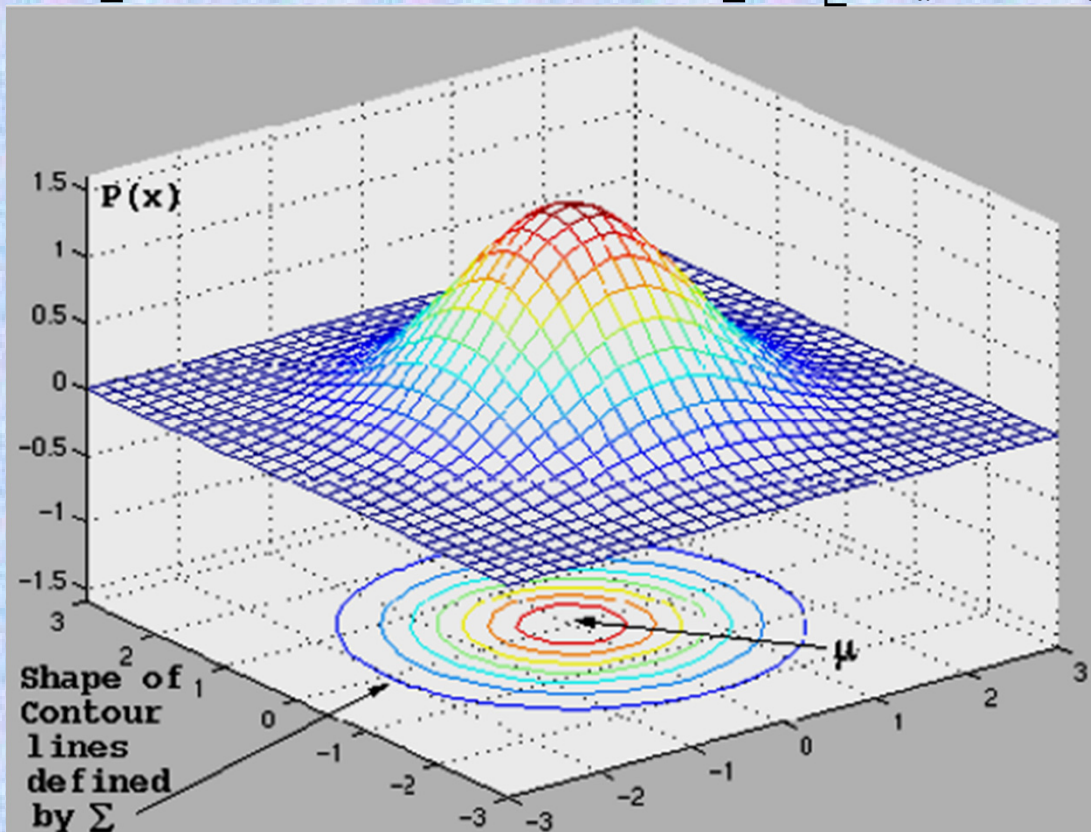
and

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} = \begin{pmatrix} \sigma_x^2 & \rho_{xy} \sigma_x \sigma_y \\ \rho_{xy} \sigma_x \sigma_y & \sigma_y^2 \end{pmatrix}$$

Can you now obtain this,  
as given earlier:

$$p(x, y) = \frac{e^{-\frac{1}{2(1-\rho_{xy}^2)} \left[ \left(\frac{x-\mu_x}{\sigma_x}\right)^2 - \frac{2\rho_{xy}(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \left(\frac{y-\mu_y}{\sigma_y}\right)^2 \right]}}{2\pi\sigma_x\sigma_y\sqrt{(1-\rho_{xy}^2)}}$$

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdot & \cdot & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{d1} & \sigma_{d2} & \cdot & \cdot & \sigma_{dd} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdot & \cdot & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdot & \cdot & \sigma_{2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sigma_{1d} & \sigma_{2d} & \cdot & \cdot & \sigma_d^2 \end{bmatrix} \quad \mu = E(X) = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdot \\ \cdot \\ \mu_d \end{bmatrix}$$



Contours have constant density of the distant term (d=2):

$$d(X) = (X - \mu)^T \Sigma_D^{-1} (X - \mu);$$

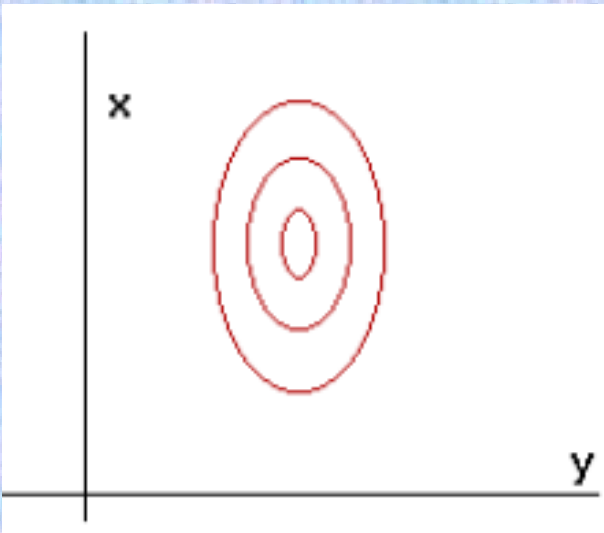
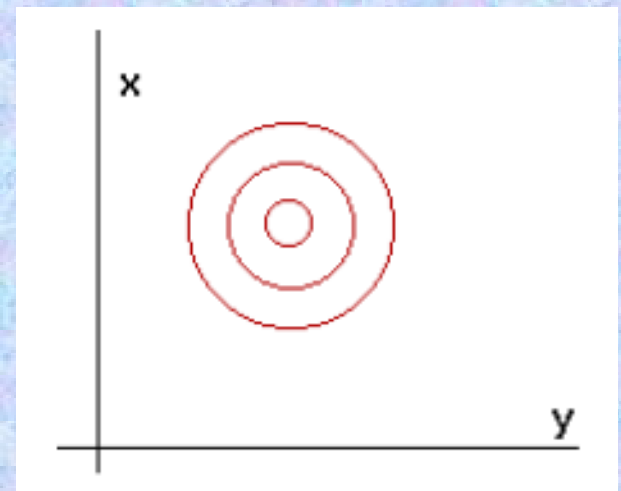
The contours are lines of constant **Mahalanobis distance** (determined by the matrix  $\Sigma$ ), and are quadratic functions.

The contours of constant density may also be hyper-ellipsoids (non-diagonal  $\Sigma$ ) of constant Mahalanobis distance to  $\mu$ .

**Diagonal covariance;**

$$\sigma_x = \sigma_y;$$

$$\rho_{xy} = 0;$$



**Diagonal covariance;**

$$\sigma_x > \sigma_y;$$

$$\rho_{xy} = 0;$$

**Asymmetric and oriented  
Gaussians**

**Non-Diagonal  
covariance;**

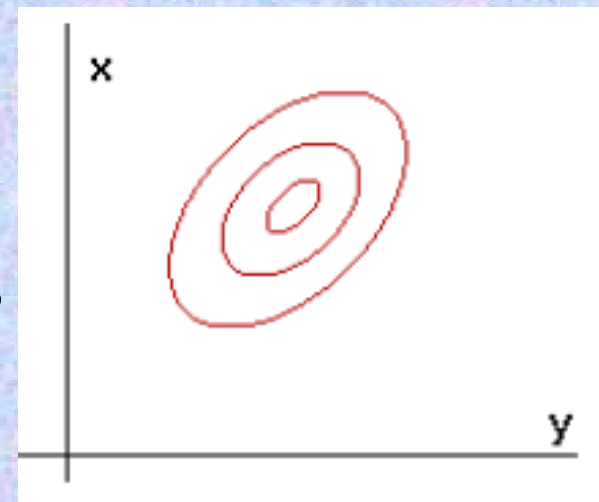
$$\sigma_x = \sigma_y;$$

$$\rho_{xy} < 0;$$



$$\sigma_x = \sigma_y;$$

$$\rho_{xy} > 0;$$





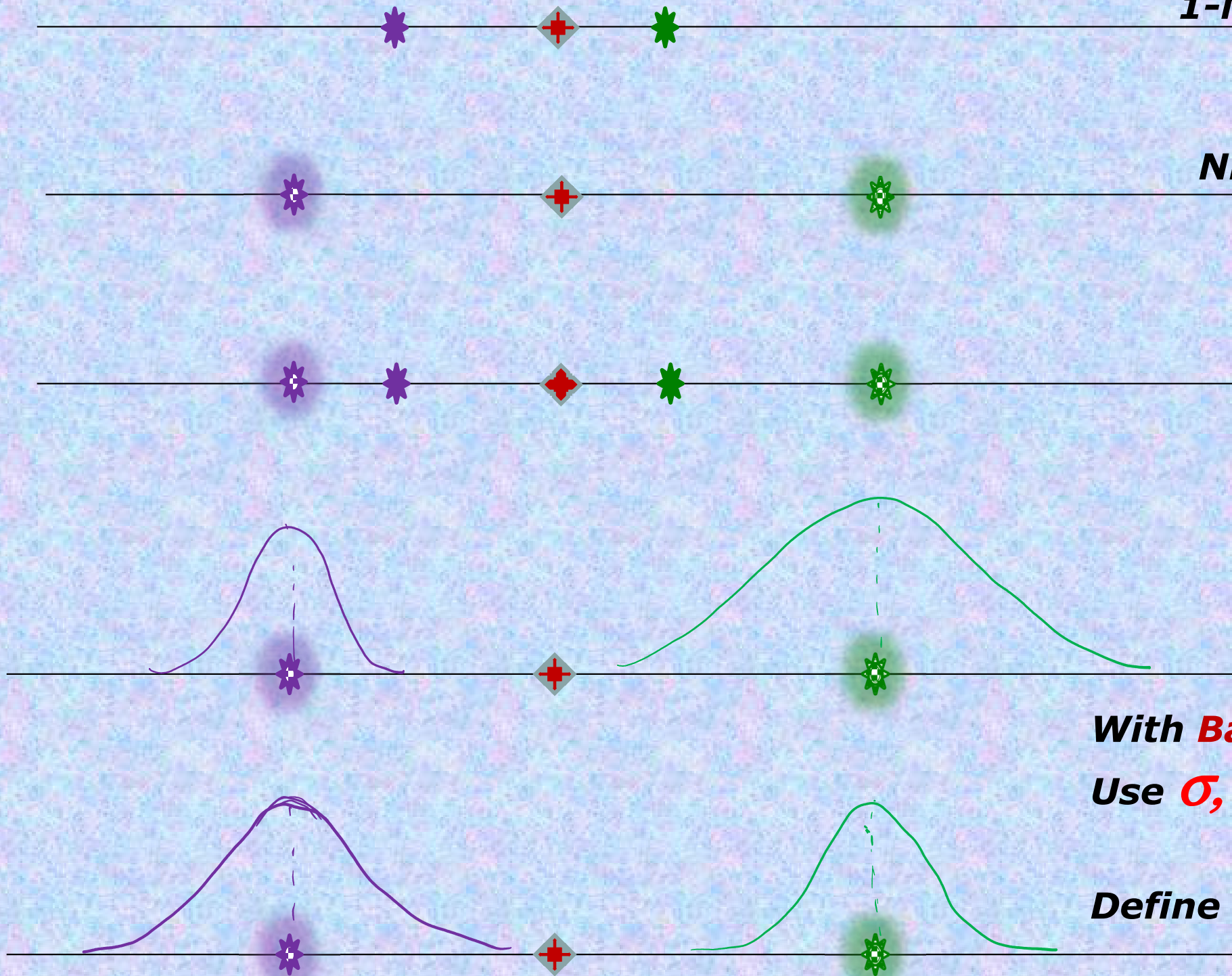
**1-NN**

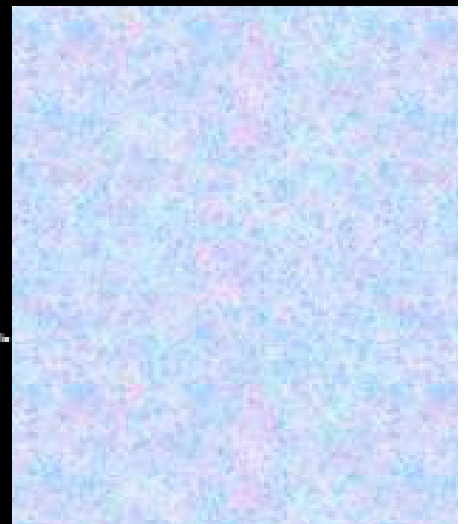
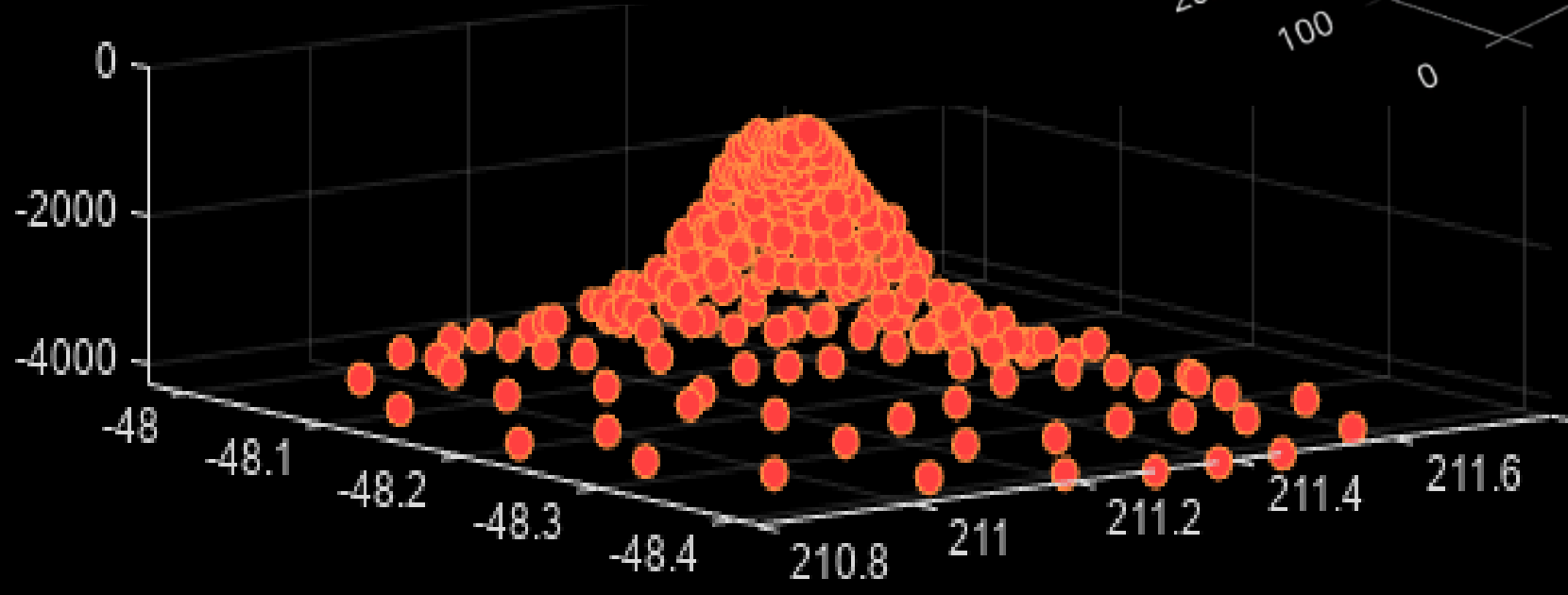
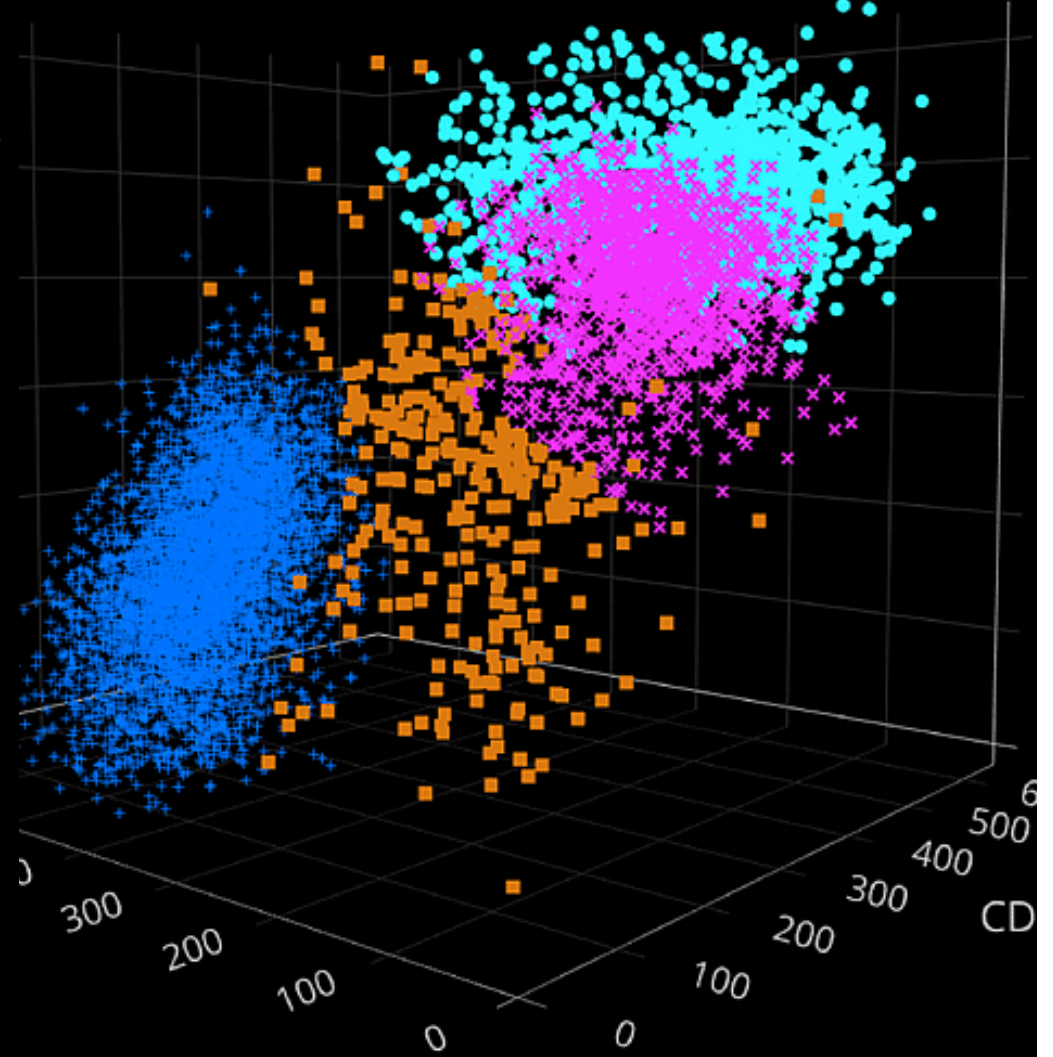
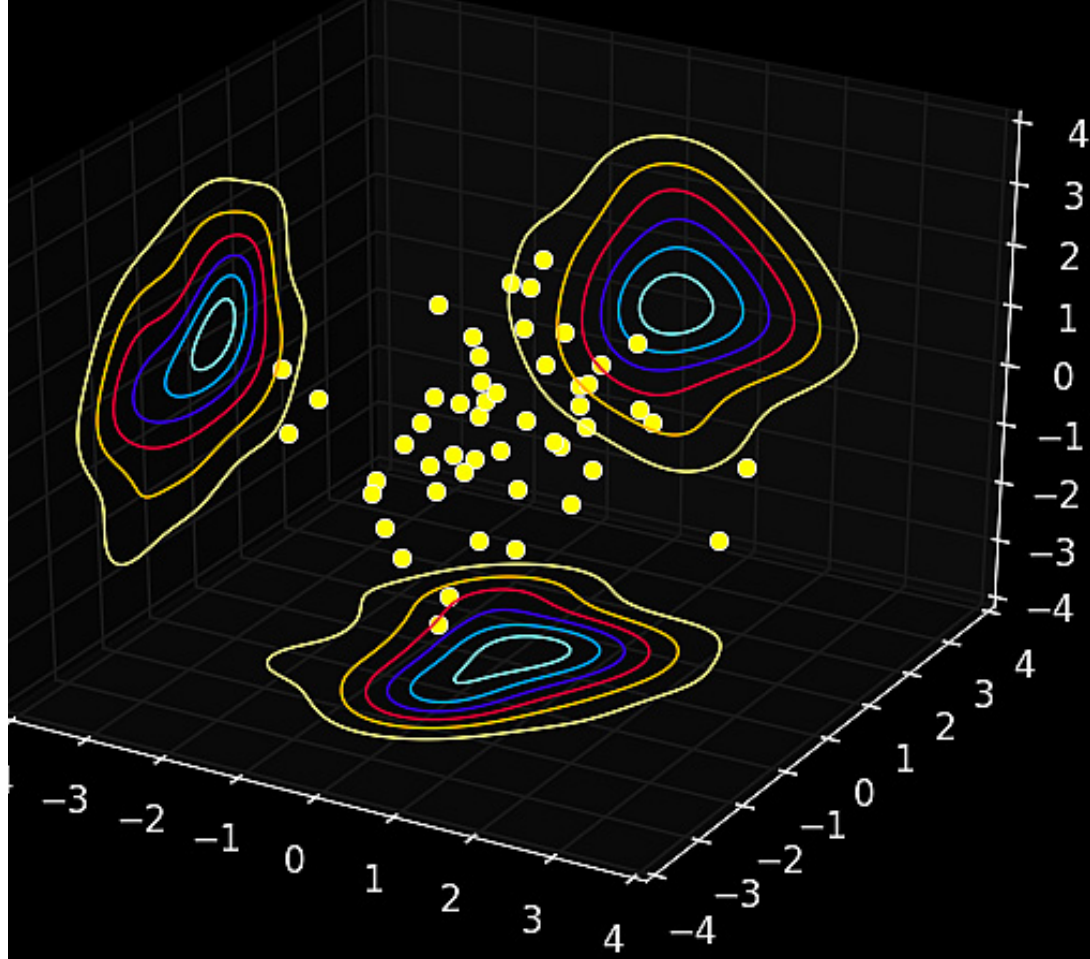
**ND**

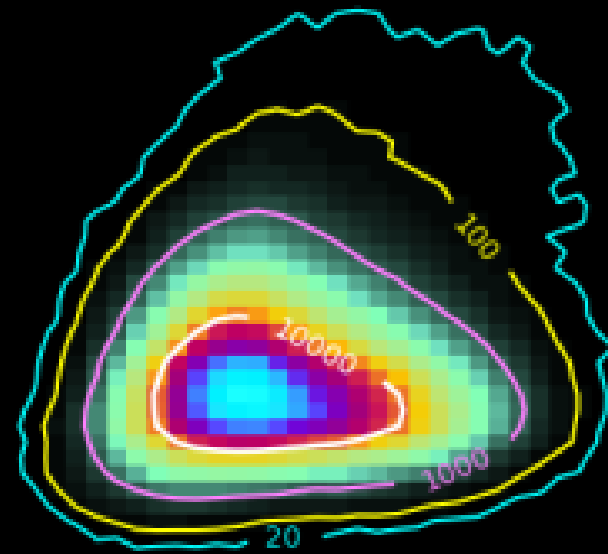
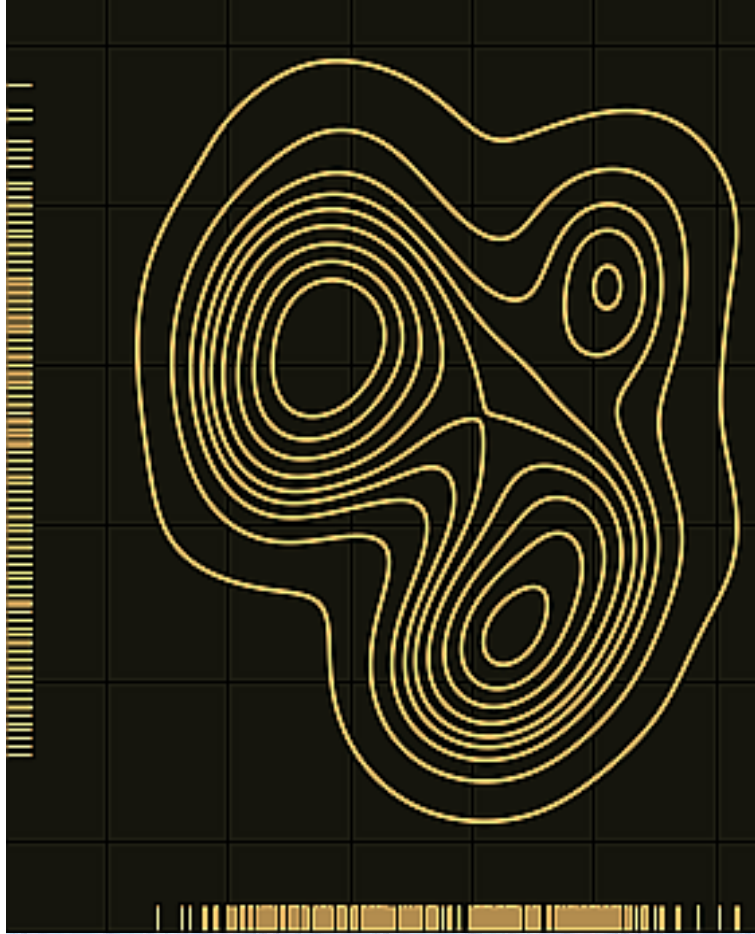
**??**

**With Bayes,  
Use  $\sigma$ ,  $\Sigma$ ,  $\mu$  :**

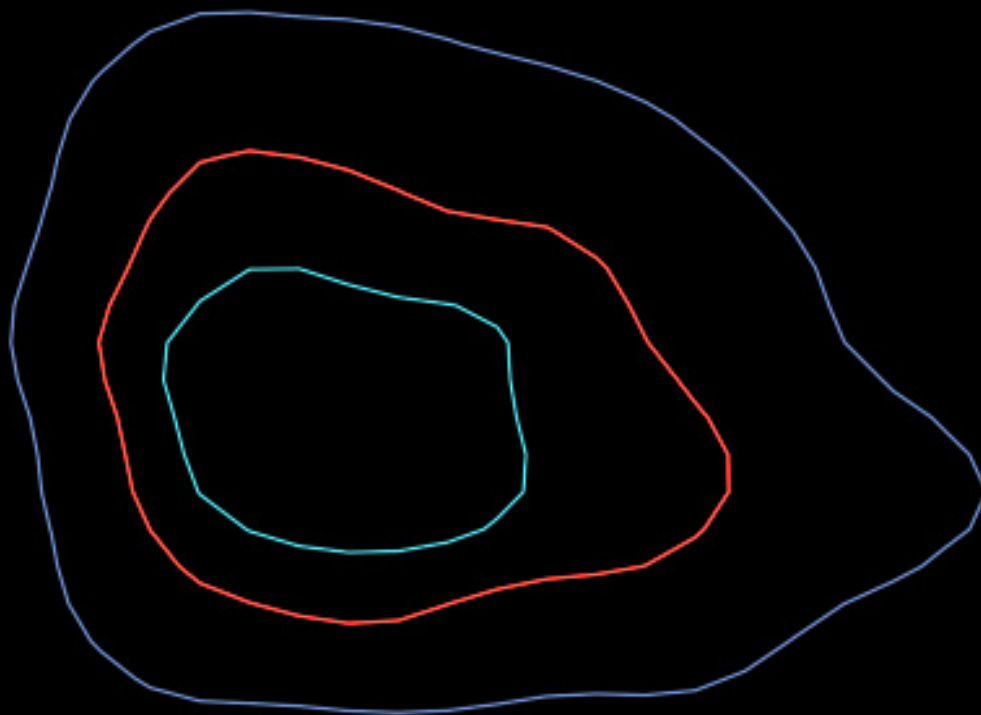
**Define DB**







Read about:  
**T-SNE plot**  
**Parzen window**



## Decision Regions and Boundaries

A classifier partitions a feature space into class-labeled decision regions (DRs).

If decision regions are used for a possible and unique class assignment, the regions must cover  $R^d$  and be disjoint (non-overlapping). In Fuzzy theory, decision regions may be overlapping.

The border of each decision region is a Decision Boundary (DBs).

**Typical classification approach is as follows:**

**Determine the decision region (in  $R^d$ ) into which  $X$  falls, and assign  $X$  to this class.**

**This strategy is simple. But determining the DRs is a challenge.**

**It may not be possible to visualize, DRs and DBs, in a general classification task with a large number of classes and higher feature space (dimension).**

Classifiers are based on Discriminant functions.

In a C-class case, Discriminant functions are denoted by:  $g_i(X)$ ,  $i = 1, 2, \dots, C$ .

This partitions the  $R^d$  into C distinct (disjoint) regions, and the process of classification is implemented using the Decision Rule:

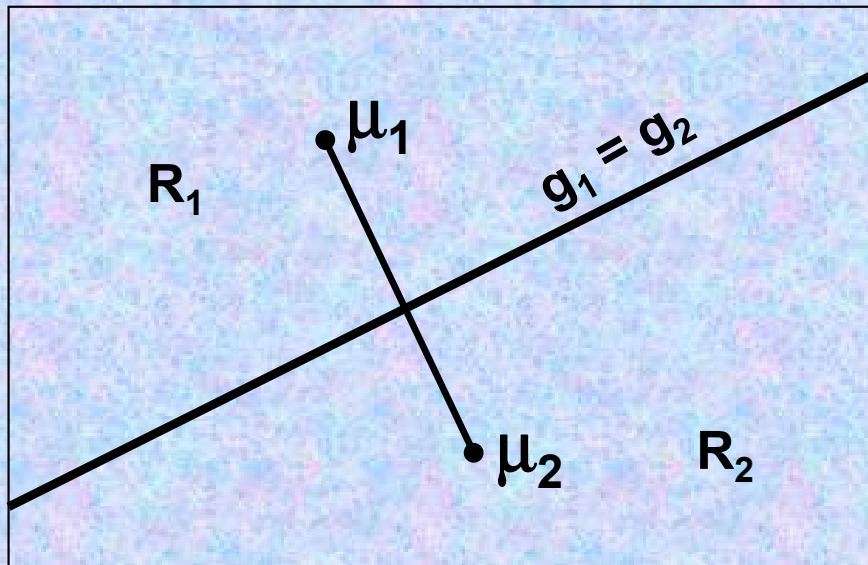
Assign X to class  $C_m$  (or region m), where:  $g_m(X) > g_i(X), \forall i, i \neq m$ .

Decision Boundary is defined by the locus of points, where:

$$g_k(X) = g_l(X), k \neq l$$

Minimum distance (also NN) classifier:

Discriminant function is based on the distance to the class mean:



$$g_1(X) = \left\| \vec{X} - \vec{\mu}_1 \right\|; \quad g_2(X) = \left\| \vec{X} - \vec{\mu}_2 \right\|$$

This does not take into account class PDFs and priors.

**Remember Baye's:** 
$$P(w_i | \vec{X}) = \frac{P(\vec{X} | w_i)P(w_i)}{P(\vec{X})}$$

**Consider  
discriminant function as:**

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|w_i) + \ln P(w_i)$$

**and class-conditional Prob. as:**

$$p(X | w_i) = \frac{1}{\sqrt{\det(\Sigma_i)(2\pi)^d}} \exp\left[-\frac{(X - \mu)^T \Sigma_i^{-1} (X - \mu)}{2}\right]$$

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

**Many cases arise, due to the varying nature of  $\Sigma$ :**

- **Diagonal (equal or unequal elements);**
- **Off-diagonal (+ve or -ve).**

Let the discrimination function for the  $i^{\text{th}}$  class be:

$$g_i(\vec{X}) = P(C_i | \vec{X}), \text{ and assume } P(C_i) = P(C_j), \forall i, j; i \neq j.$$

Remember, multivariate Gaussian density?

$$g_i(X) = P(X | C_i) = \frac{1}{\sqrt{\det(\Sigma_i)(2\pi)^d}} \exp\left[-\frac{(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)}{2}\right]$$

**Define:**

$$\begin{aligned} G_i(X) &= \log[P(X | C_i)] = \log\left[\frac{1}{\sqrt{\det(\Sigma_i)(2\pi)^d}}\right] - \frac{(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)}{2} \\ &= k \cdot \vec{d}_i^2 + q \end{aligned}$$

Thus the classification is now influenced by the square distance (hyper-dimensional) of  $X$  from  $\mu_i$ , weighted by the  $\Sigma^{-1}$ .

Let us examine:

$$\vec{d}_i^2 = (X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)$$

This quadratic term (scalar) is known as the **Mahalanobis distance** (the distance from  $X$  to  $\mu_i$  in feature space).

$$\vec{d}_i^2 = (X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)$$

For a given  $X$ , some  $G_m(X)$  is largest where  $(d_m)^2$  is the smallest, for a class  $i = m$  (assign  $X$  to class  $m$ , based on NN Rule) .

**Simplest case:**  $\Sigma = \mathbf{I}$ , the criteria becomes the Euclidean distance norm (and hence the NN classifier).

This is equivalent to obtaining the mean  $\mu_m$ , for which  $X$  is the nearest, for all  $\mu_i$ . The distance function is then:

$$\vec{d}_i^2 = \|X - \mu_i\|^2 = X^T X - 2\mu_i^T X + \mu_i^T \mu_i \quad (\text{all vector notations})$$

$$\begin{aligned} \text{Thus, } G_i(X) &= d_i^2 / 2 = (X^T X) / 2 - \mu_i^T X + (\mu_i^T \mu_i) / 2 \\ &= \omega_i^T X + \omega_{i0} \end{aligned}$$

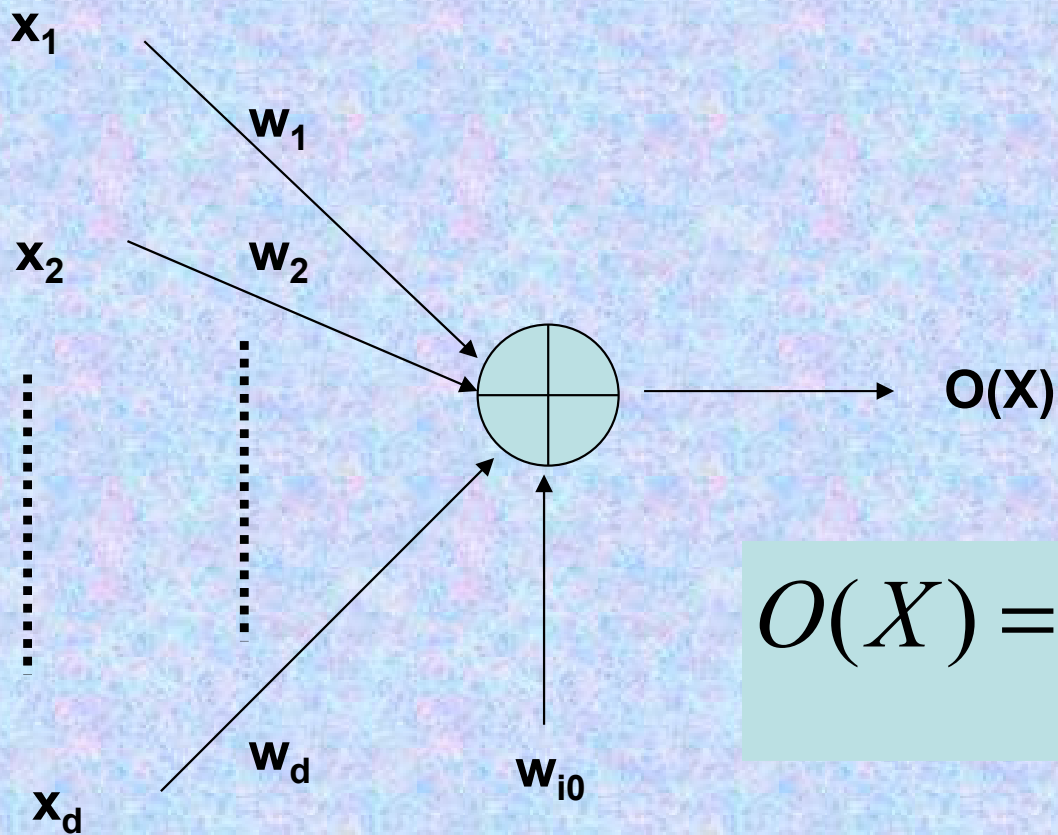
*Neglecting the class-invariant term.*

$$\text{where, } \omega_i^T = \mu_i \text{ and } \omega_{i0} = -\frac{\mu_i^T \mu_i}{2}$$

This gives the simplest **linear discriminant function or correlation detector.**



# The perceptron (ANN) built to form the linear discriminant function



$$O(X) = \left( \sum_i w_i x_i \right) + w_{i0}$$

View this as (in 2-D space):

$$G = MX - Y + C$$

The decision region boundaries are determined by solving :

$$G_i(X) = G_j(X), \text{ which gives : } (\omega_i^T - \omega_j^T)X + (\omega_{i0} - \omega_{j0}) = 0$$

This is an expression of a hyperplane separating the decision regions in  $\mathbb{R}^d$ . The hyperplane will pass through the origin, if:

$$\omega_{i0} = \omega_{j0}$$

Generalized results (Gaussian case) of a discriminant function:

$$\begin{aligned} G_i(X) &= \log[P(X | C_i)] = \log\left[\frac{1}{\sqrt{\det(\Sigma_i)}(2\pi)^d}\right] - \frac{(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)}{2} \\ &= -\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) - \left(\frac{d}{2}\right) \log(2\pi) - \frac{1}{2} \log(\Sigma_i) \end{aligned}$$

The **mahalanobis distance** (quadratic term) spawns a number of different surfaces, depending on  $\Sigma^{-1}$ . It is basically a vector distance using a  $\Sigma^{-1}$  norm. It is denoted as:

$$\left\| X - \mu_i \right\|_{\Sigma_i^{-1}}^2$$

**Make the case of Baye's rule more general for class assignment.**

**Earlier we has assumed that:**

$$g_i(\vec{X}) = P(C_i | \vec{X}), \text{ assuming } P(C_i) = P(C_j), \forall i, j; i \neq j.$$

**Now,**  $G_i(\vec{X}) = \log[P(C_i | \vec{X}).P(\vec{X})] = \log[P(\vec{X} | C_i)] + \log[P(C_i)]$

$$G_i(X) = \log\left[\frac{1}{\sqrt{\det(\Sigma_i)}(2\pi)^d}\right] - \frac{(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)}{2} + \log[P(C_i)]$$

$$= -\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) - \left(\frac{d}{2}\right) \log(2\pi) - \frac{1}{2} \log(\Sigma_i) + \log[P(C_i)]$$

$$= -\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) - \frac{1}{2} \log(\Sigma_i) + \log[P(C_i)] \quad \text{Neglecting the constant term}$$

**Simpler case:**  $\Sigma_i = \sigma^2 \mathbf{I}$ , and eliminating the class-independent bias, we have:

$$G_i(X) = -\frac{1}{2\sigma^2}(X - \mu_i)^T (X - \mu_i) + \log[P(C_i)]$$

**These are loci of constant hyper-spheres, centered at class mean.**

**More on this later on.....**

If  $\Sigma$  is a diagonal matrix, with equal/unequal  $\sigma_{ii}^2$ :

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdot & \cdot & 0 \\ 0 & \sigma_2^2 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \sigma_d^2 \end{bmatrix} \quad \text{and} \quad \Sigma^{-1} = \begin{bmatrix} 1/\sigma_1^2 & 0 & \cdot & \cdot & 0 \\ 0 & 1/\sigma_2^2 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & 1/\sigma_d^2 \end{bmatrix}$$

Considering the discriminant function:

$$G_i(X) = -\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) - \frac{1}{2} \log(\Sigma_i) + \log[P(C_i)]$$

This now will yield a weighted distance classifier. Depending on the covariance term (*more spread/scatter or not*), we tend to put more emphasis on some feature vector components than the other.

Check out the following:

This will give hyper-elliptical surfaces in  $\mathbb{R}^d$ , for each class.

It is also possible to linearise it.

## More general decision boundaries

Take  $P(C_i) = K$  for all  $i$ , and eliminating the class independent terms yield:

$$G_i(X) = (X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)$$

$$\overset{\rightarrow 2}{d}_i = (X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) = -X^T \Sigma_i^{-1} X + 2\mu_i^T \Sigma_i^{-1} X - \mu_i^T \Sigma_i^{-1} \mu_i$$

$$G_i(X) = (\Sigma^{-1} \mu_i)^T X - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i \quad \text{as } \Sigma_i = \Sigma, \text{ and are symmetric.}$$

$$\text{Thus, } G_i(X) = \omega_i^T X + \omega_{i0}$$

$$\text{where } \omega_i = \Sigma^{-1} \mu_i \text{ and } \omega_{i0} = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i$$

Thus the decision surfaces are hyperplanes and decision boundaries will also be linear (use  $G_i(X) = G_j(X)$ , as done earlier)

Beyond this, if a diagonal  $\Sigma$  is class-dependent or off-diagonal terms are non-zero, we get **non-linear DFs, DRs or DBs**.

The discriminant function (DF) for **linearly separable** classes is:

$$g_i(X) = \omega_i^T X + \omega_{i0}$$

where,  $\omega_i$  is a dx1 vector of weights used for class i.

This function leads to DBs that are hyperplanes. It's a point in 1D, line in 2-D, planar surfaces in 3-D, and .....

3-D case:  $(\omega_1 \omega_2 \omega_3) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 0$  is a plane passing through the origin.

In general, the equation:  $\omega^T (\vec{X} - \vec{X}_d) = 0; \Rightarrow \omega^T \vec{X} - d = 0$

represents a plane H passing through any point (position vector)  $X_d$ .

This plane partitions the space into two mutually exclusive regions, say  $R_p$  and  $R_n$ . The assignment of the vector X to either the +ve side, or -ve side or along H, can be implemented by:

$$\omega^T \vec{X} - d \begin{cases} > 0 & \text{if } X \in R_p \\ = 0 & \text{if } X \in H \\ < 0 & \text{if } X \in R_n \end{cases}$$

### 4.1.1 Two classes

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector so that

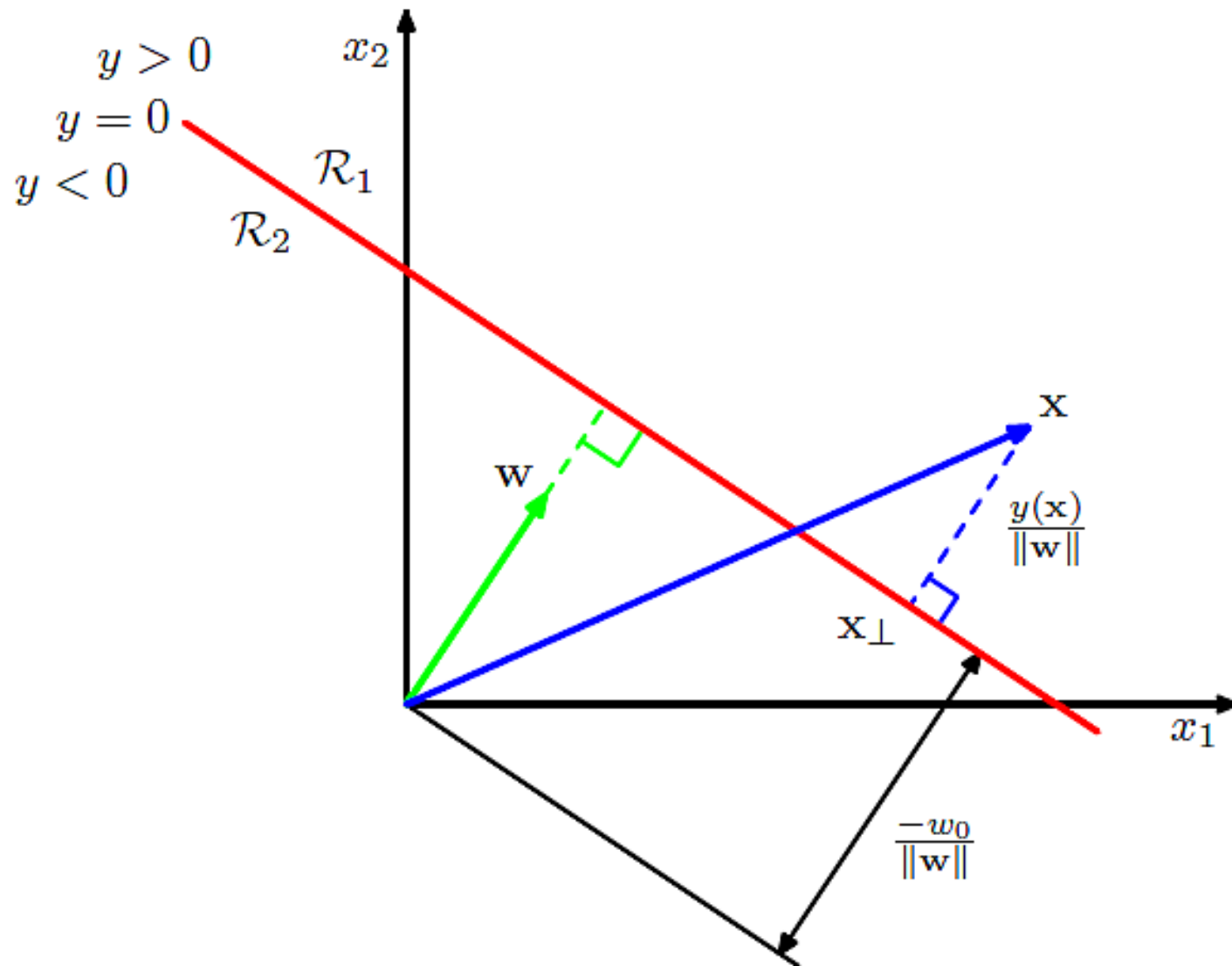
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (4.4)$$

where  $\mathbf{w}$  is called a *weight vector*, and  $w_0$  is a *bias* (not to be confused with bias in the statistical sense). The negative of the bias is sometimes called a *threshold*. An input vector  $\mathbf{x}$  is assigned to class  $\mathcal{C}_1$  if  $y(\mathbf{x}) \geq 0$  and to class  $\mathcal{C}_2$  otherwise. The corresponding decision boundary is therefore defined by the relation  $y(\mathbf{x}) = 0$ , which corresponds to a  $(D - 1)$ -dimensional hyperplane within the  $D$ -dimensional input space. Consider two points  $\mathbf{x}_A$  and  $\mathbf{x}_B$  both of which lie on the decision surface. Because  $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$ , we have  $\mathbf{w}^T (\mathbf{x}_A - \mathbf{x}_B) = 0$  and hence the vector  $\mathbf{w}$  is orthogonal to every vector lying within the decision surface, and so  $\mathbf{w}$  determines the orientation of the decision surface. Similarly, if  $\mathbf{x}$  is a point on the decision surface, then  $y(\mathbf{x}) = 0$ , and so the normal distance from the origin to the decision surface is given by

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}. \quad (4.5)$$

We therefore see that the bias parameter  $w_0$  determines the location of the decision surface. These properties are illustrated for the case of  $D = 2$  in Figure 4.1.

**Figure 4.1** Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to  $\mathbf{w}$ , and its displacement from the origin is controlled by the bias parameter  $w_0$ . Also, the signed orthogonal distance of a general point  $\mathbf{x}$  from the decision surface is given by  $y(\mathbf{x})/\|\mathbf{w}\|$ .





An alternative is to introduce  $K(K - 1)/2$  binary discriminant functions, one for every possible pair of classes. This is known as a *one-versus-one* classifier. Each point is then classified according to a majority vote amongst the discriminant functions. However, this too runs into the problem of ambiguous regions, as illustrated in the right-hand diagram of Figure 4.2.

We can avoid these difficulties by considering a single  $K$ -class discriminant comprising  $K$  linear functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad (4.9)$$

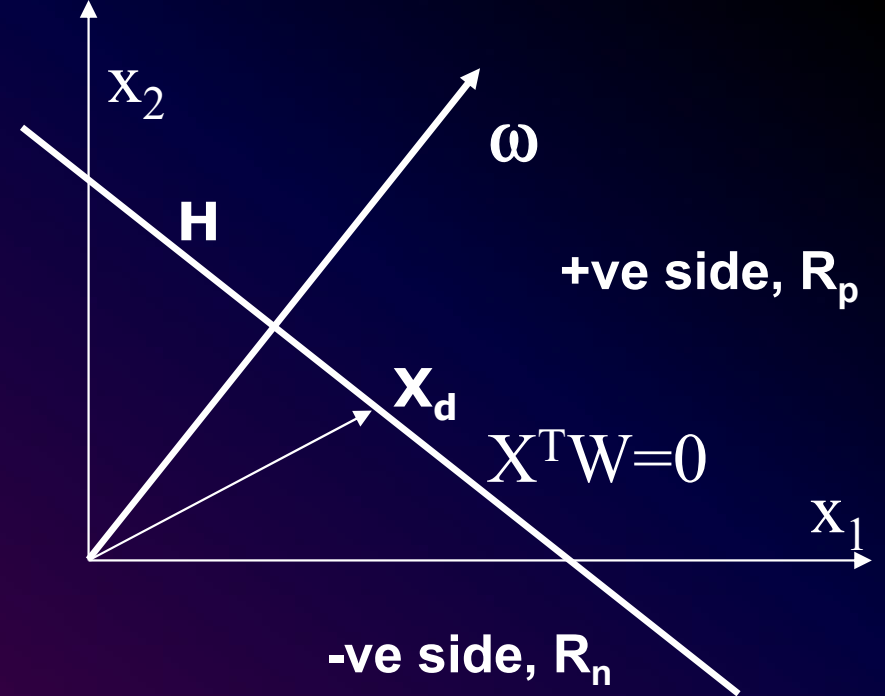
and then assigning a point  $\mathbf{x}$  to class  $\mathcal{C}_k$  if  $y_k(\mathbf{x}) > y_j(\mathbf{x})$  for all  $j \neq k$ . The decision boundary between class  $\mathcal{C}_k$  and class  $\mathcal{C}_j$  is therefore given by  $y_k(\mathbf{x}) = y_j(\mathbf{x})$  and hence corresponds to a  $(D - 1)$ -dimensional hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0. \quad (4.10)$$

This has the same form as the decision boundary for the two-class case discussed in Section 4.1.1, and so analogous geometrical properties apply.

**A relook at,  
Linear Discriminant Function  $g(\mathbf{X})$ :**

$$g(\mathbf{X}) = \omega^T \vec{\mathbf{X}} - d$$



**Orientation of H is determined by  $\omega$ .**

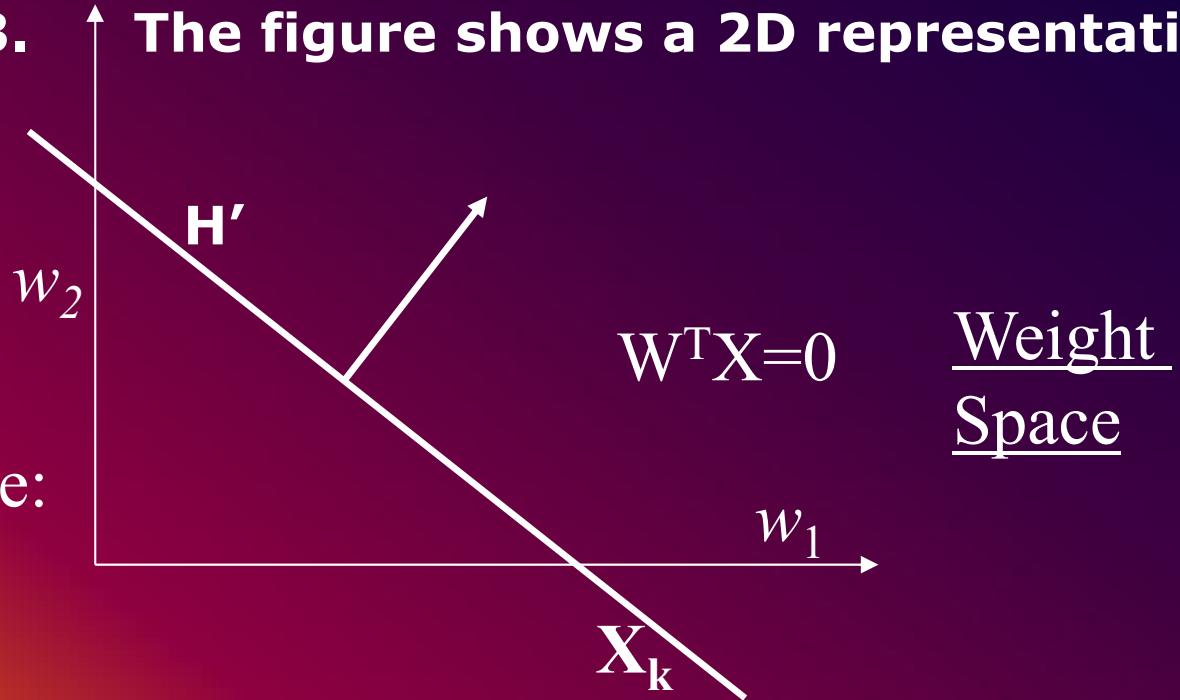
**Location of H is determined by  $d$ .**

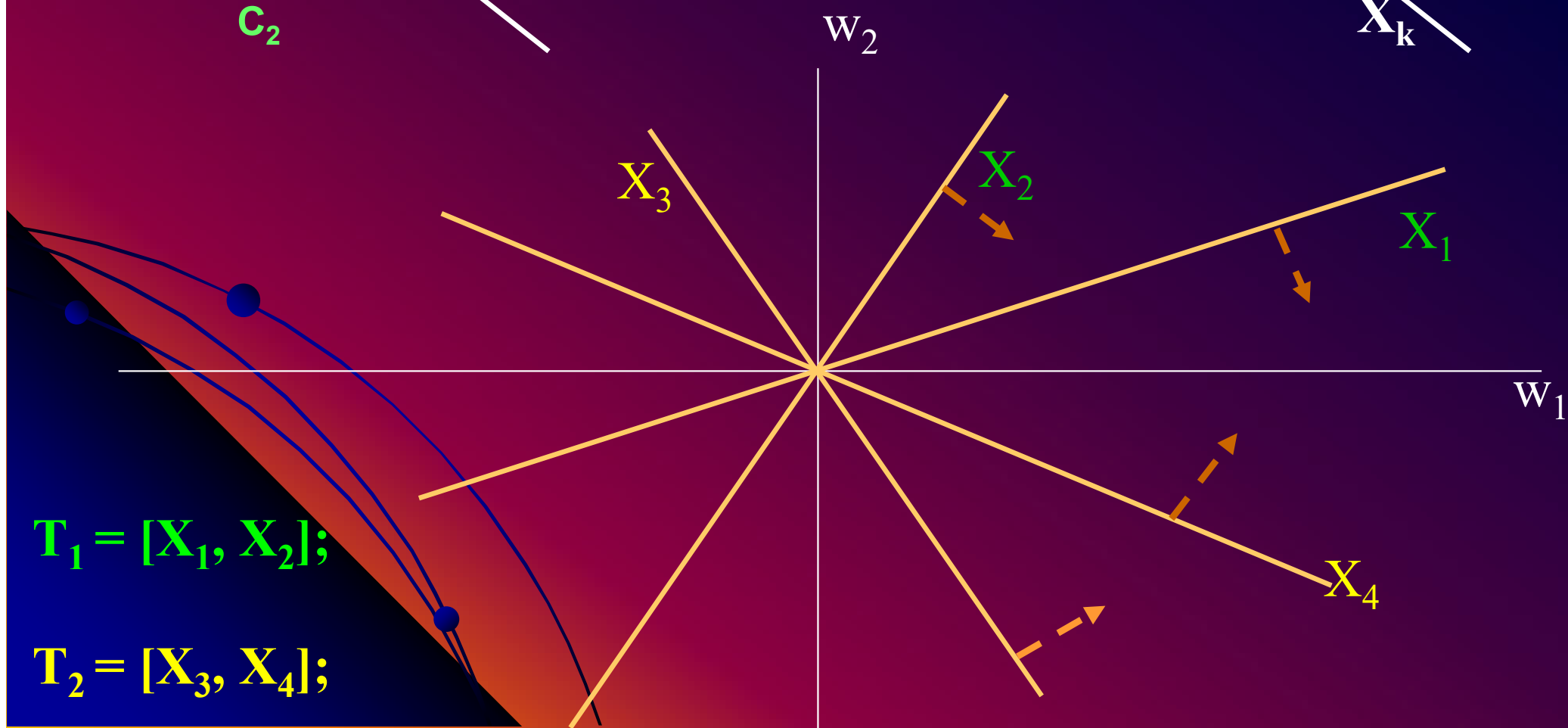
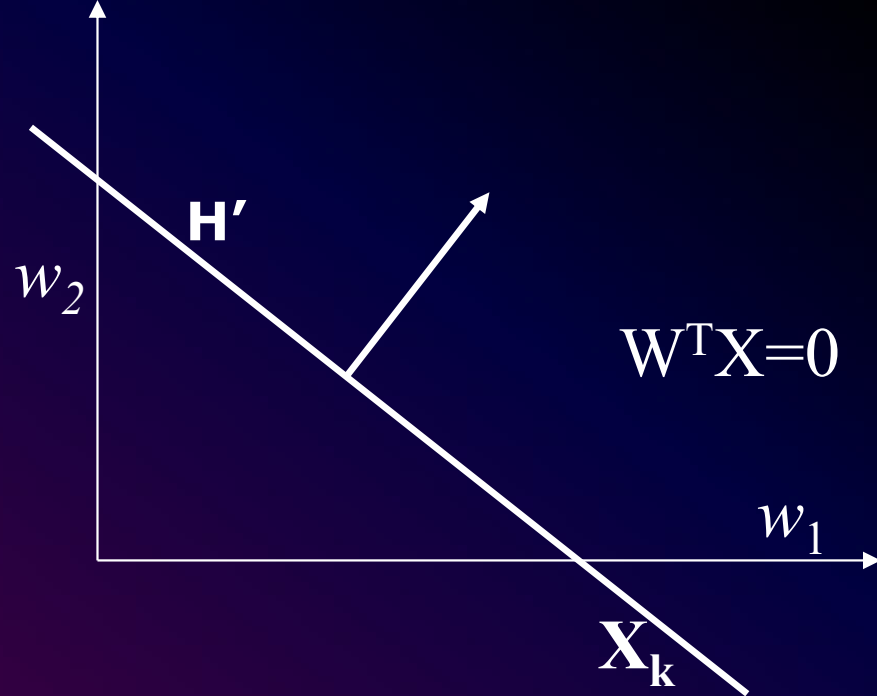
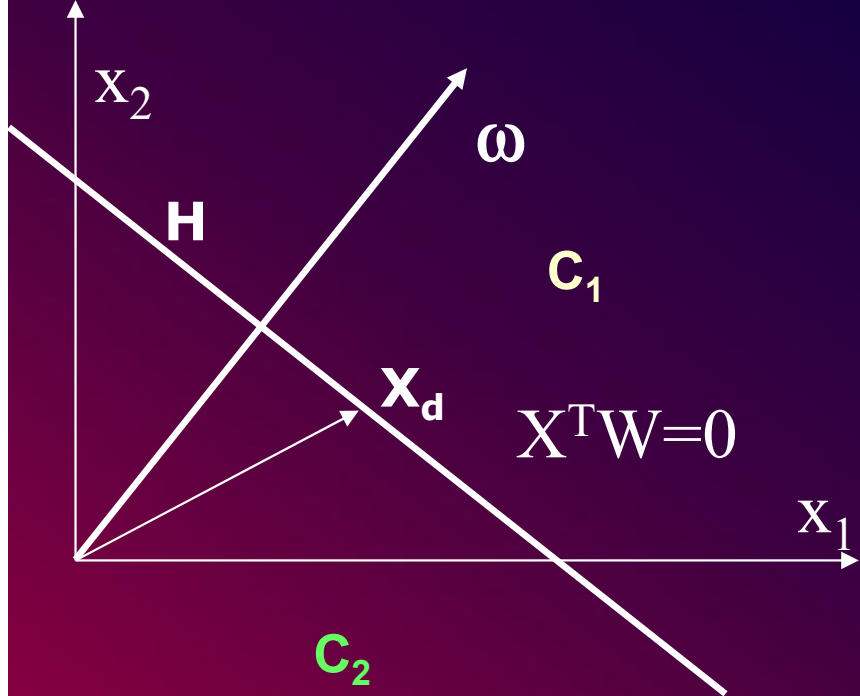
Pattern/feature Space

**H is a hyperplane for  $d > 3$ .**

**The figure shows a 2D representation.**

The complementary role of  
a sample in parametric space:





$$T_1 = [X_1, X_2];$$

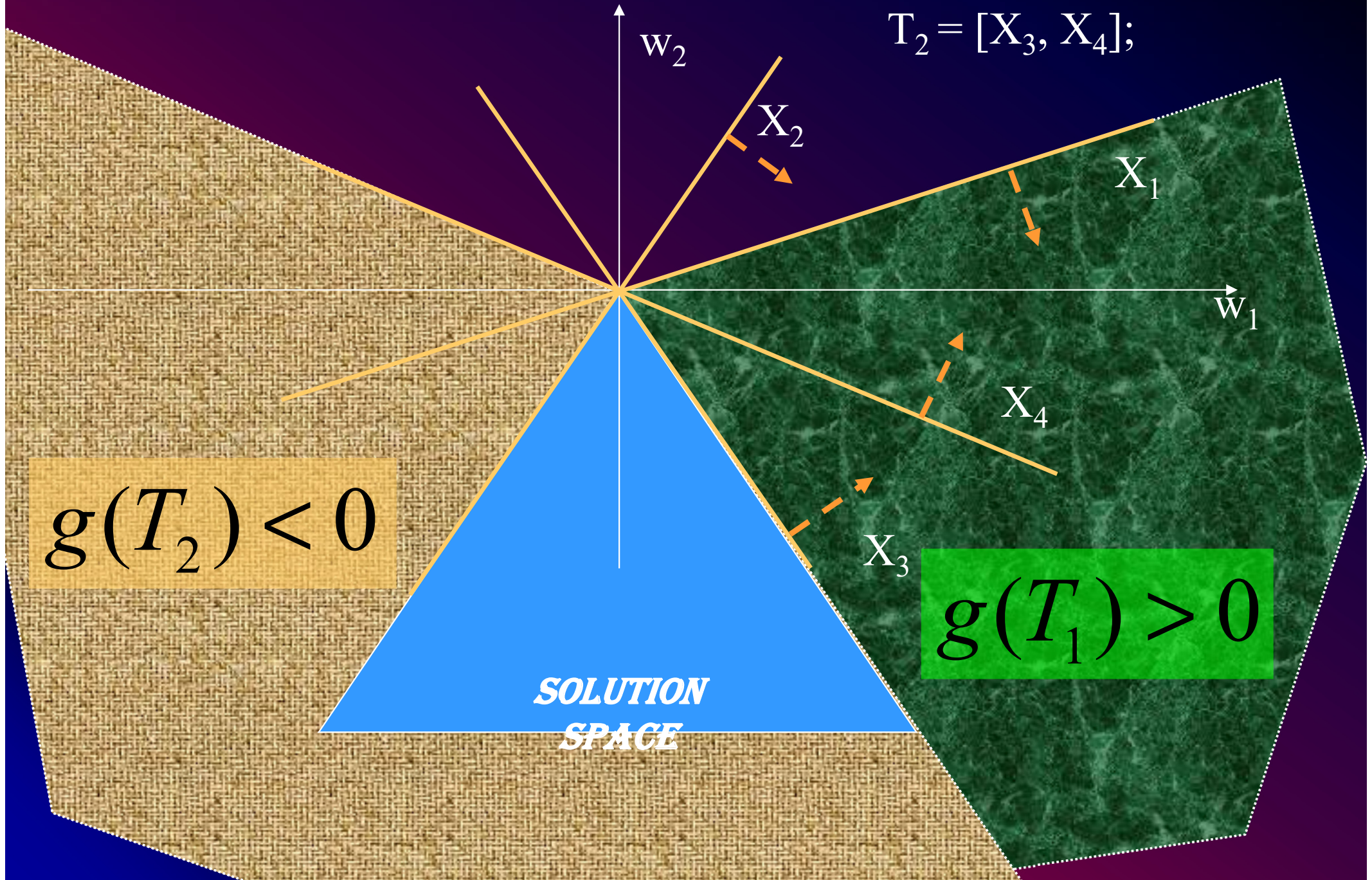
$$T_2 = [X_3, X_4];$$

 $w_2$  $X_2$  $X_1$  $w_1$  $X_4$  $X_3$ 

$$g(T_2) < 0$$

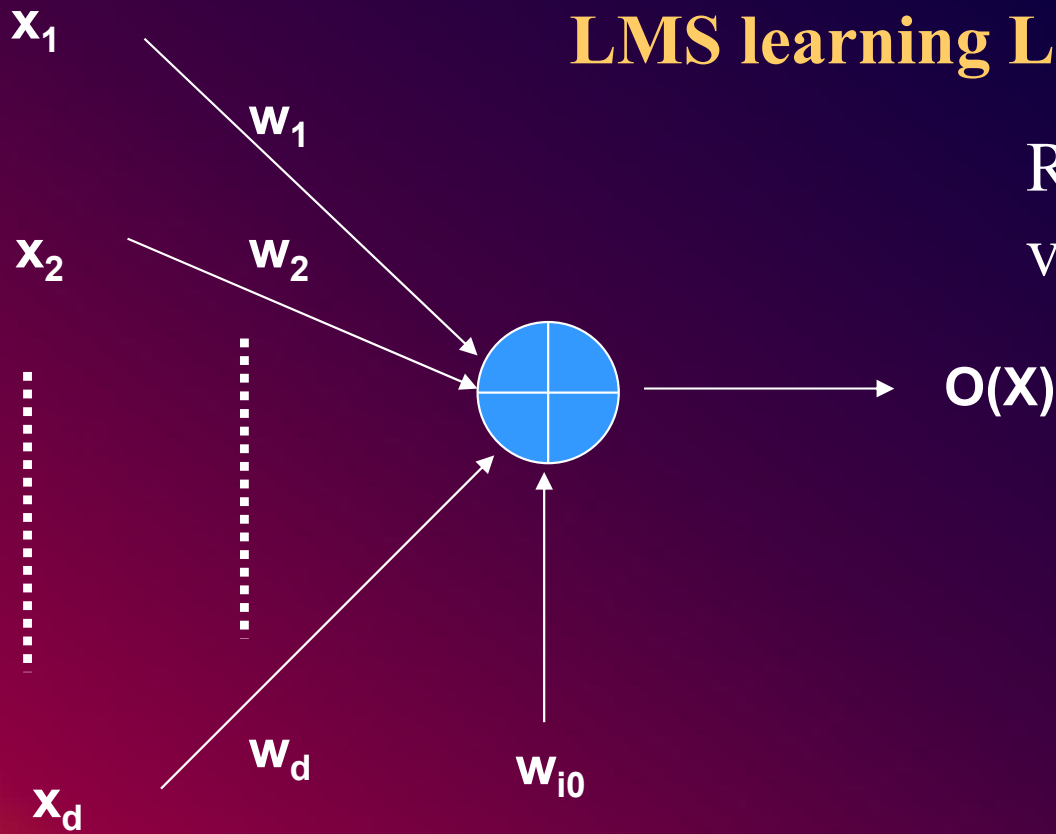
$$g(T_1) > 0$$

*SOLUTION  
SPACE*



# LMS learning Law in BPNN or FFNN models

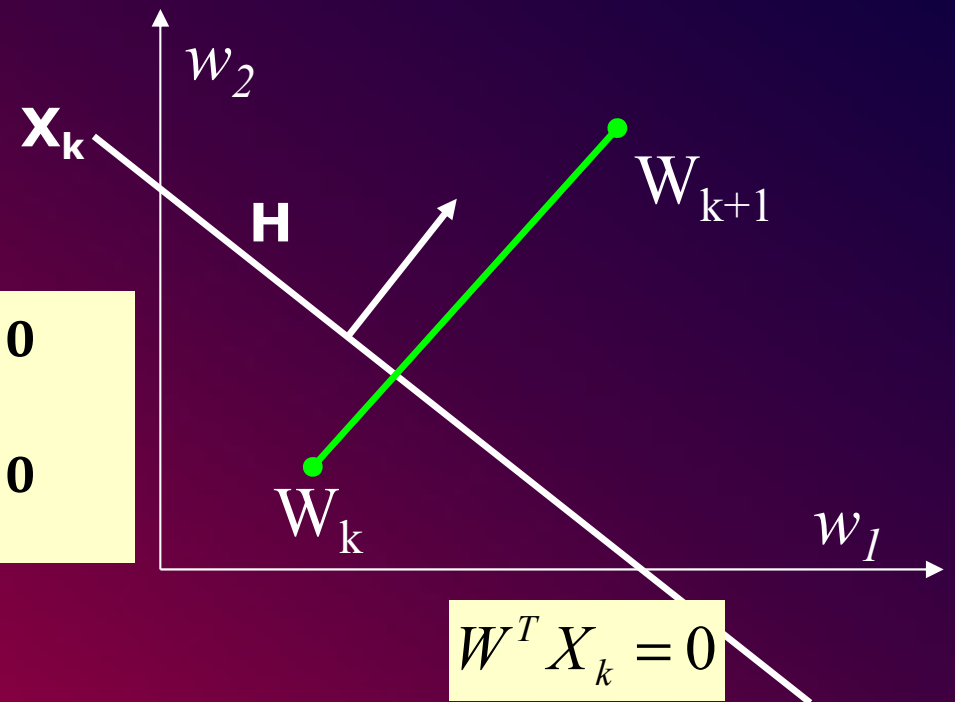
Read about [perceptron](#) vs. multi-layer feedforward network



$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } X_k^T W_k \leq 0 \\ W_k & \text{if } X_k^T W_k \geq 0 \end{cases}$$

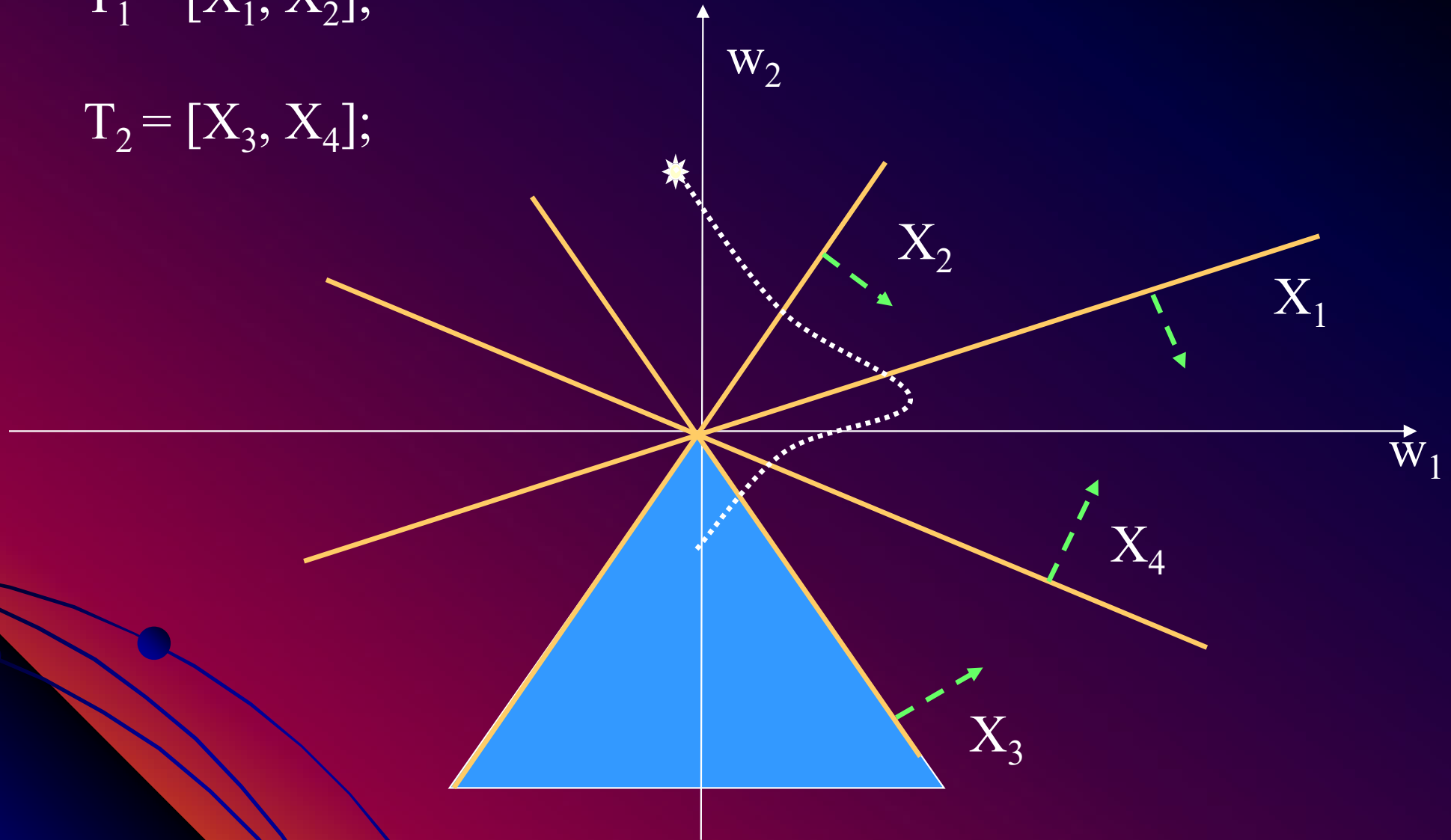
$\eta_k$  is the learning rate parameter

$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } X_k \in C_1 \text{ and } X_k^T W_k \leq 0 \\ W_k - \eta_k X_k & \text{if } X_k \in C_0 \text{ and } X_k^T W_k \geq 0 \end{cases}$$



$$T_1 = [X_1, X_2];$$

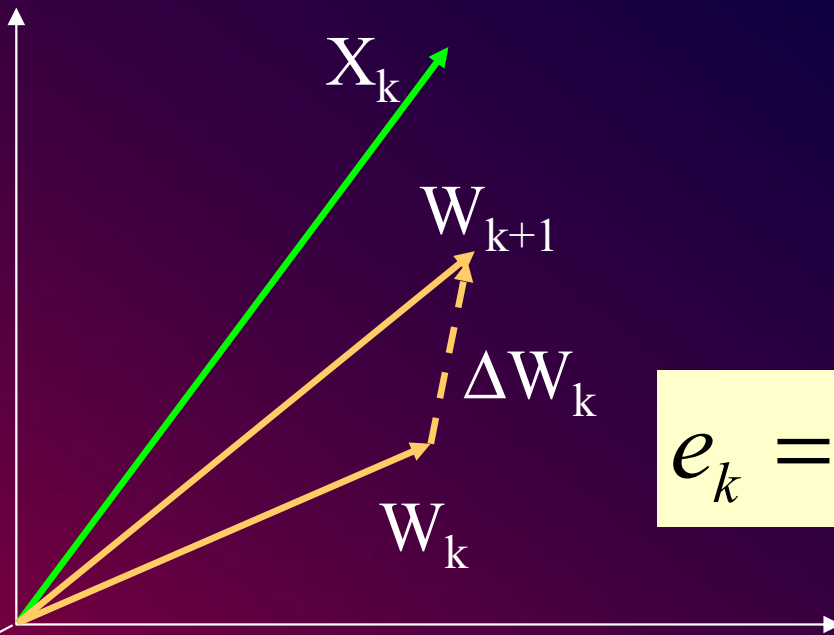
$$T_2 = [X_3, X_4];$$



$\eta_k$  decreases with each iteration

$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } X_k \in C_1 \text{ and } X_k^T W_k \leq 0 \\ W_k - \eta_k X_k & \text{if } X_k \in C_0 \text{ and } X_k^T W_k \geq 0 \end{cases}$$

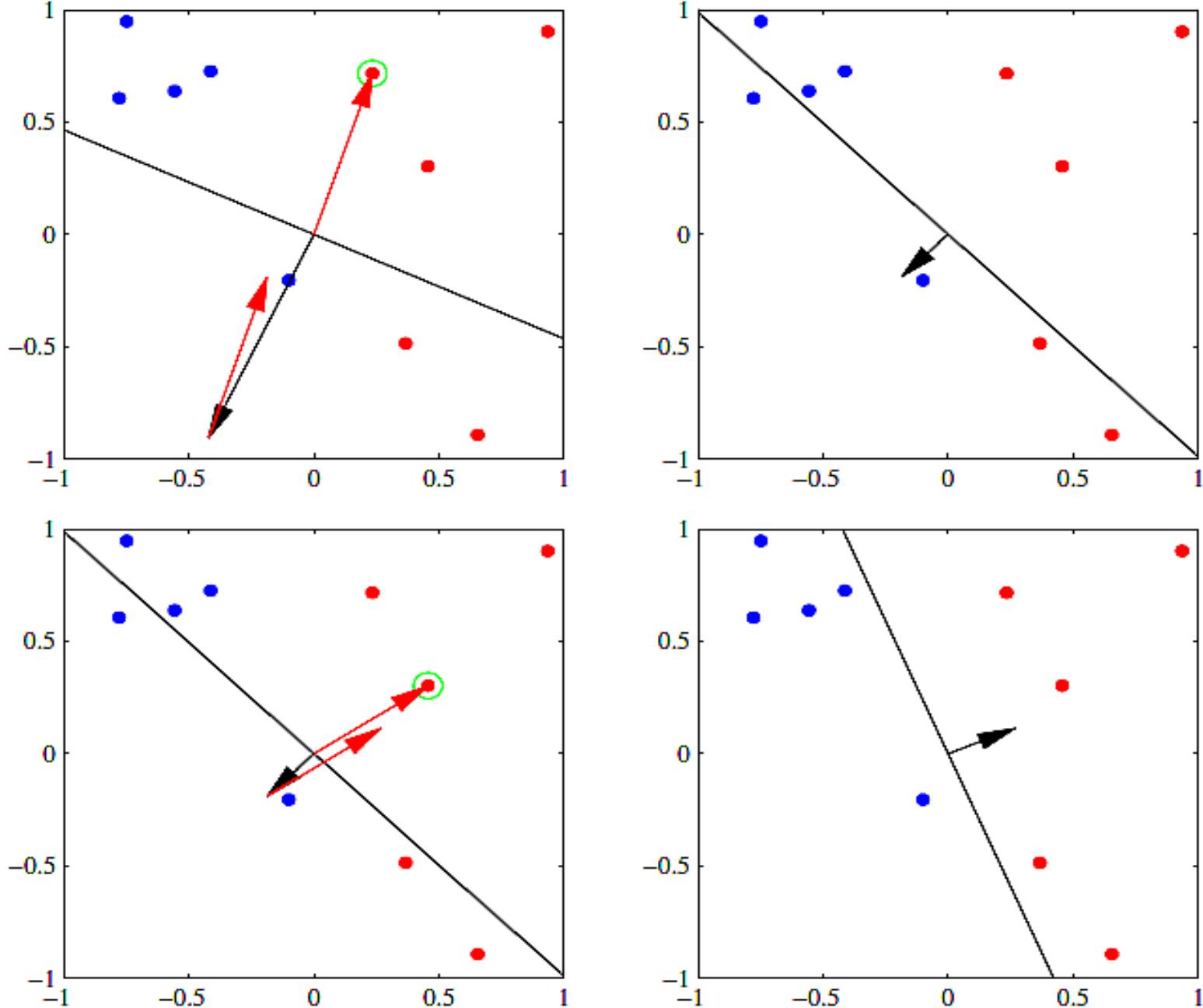
In case of FFNN, the objective is to minimize the error term:



$$e_k = d_k - s_k = d_k - X_k^T W_k$$

$\alpha$ -LMS Learning Algorithm:

$$\Delta W_k = \eta e_k \hat{X}_k$$



**Figure 4.7** Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space  $(\phi_1, \phi_2)$ . The top left plot shows the initial parameter vector  $\mathbf{w}$  shown as a black arrow together with the corresponding decision boundary (black line), in which the arrow points towards the decision region which classified as belonging to the red class. The data point circled in green is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary shown in the top right plot. The bottom left plot shows the next misclassified point to be considered, indicated by the green circle, and its feature vector is again added to the weight vector giving the decision boundary shown in the bottom right plot for which all data points are correctly classified.



***Self-study - Bishop chap 5;***

***Start at Sec. 4.1.7, pp 192.***

## MSE error surface (in case of multi-layer perceptron):

$$\xi_k = \frac{1}{2} [d_k - X_k^T W_k]^2 = E / 2 - P^T W + (1/2) W^T R W.$$

$$P^T = E[d_k X_k^T];$$

$$R = E[X_k X_k^T] = E \left[ \begin{array}{ccc} 1 & x_1^k & x_n^k \\ x_1^k & x_1^k x_1^k & x_1^k x_n^k \\ x_n^k & x_n^k x_1^k & x_n^k x_n^k \end{array} \right]$$

$$\nabla \xi = \left( \frac{\delta \xi}{\delta w_0}, \frac{\delta \xi}{\delta w_1}, \dots, \frac{\delta \xi}{\delta w_n} \right)^T = -P + RW$$

Thus,

$$\hat{W} = R^{-1} P$$

## Effect of class Priors – revisiting DBs in a more general case.

$$p(X | w_i) = \frac{1}{\sqrt{\det(\Sigma)(2\pi)^d}} \exp\left[-\frac{(X - \mu)^T \Sigma^{-1} (X - \mu)}{2}\right]$$
$$P(w_i | \vec{X}) = \frac{P(\vec{X} | w_i)P(w_i)}{P(\vec{X})}$$

$$g_i(x) = \ln p(x|w_i) + \ln P(w_i)$$

$$g_i(x) = \frac{-1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma| + \ln P(w_i)$$

## CASE A. – Same diagonal $\Sigma$ , with identical diagonal elements.

Canceling in class-invariant terms:

$$g_i(X) = \frac{-1}{2\sigma^2} [(X - \mu_i)^T (X - \mu_i)] + \ln P(w_i)$$

$$g_i(X) = \frac{-1}{2\sigma^2} [X^T X - 2\mu_i^T X + \mu_i^T \mu_i] + \ln P(w_i)$$

$$g_i(X) = \frac{-1}{2\sigma^2} [X^T X - 2\mu_i^T X + \mu_i^T \mu_i] + \ln P(w_i)$$

Thus,  $g_i(X) = \omega_i^T X + \omega_{i0}$

where  $\omega_i = \mu_i / \sigma^2$  and  $\omega_{i0} = -\frac{\mu_i^T \mu_i}{2\sigma^2} + \ln P(w_i)$

**The linear DB is thus:**  $g_k(X) = g_l(X), k \neq l$

**which is:**  $(\omega_k^T - \omega_l^T)X + (\omega_{k0} - \omega_{l0}) = 0;$

**Prove that the 2<sup>nd</sup> constant term:**

$$(\omega_{k0} - \omega_{l0}) = (\omega_l - \omega_k)^T X_0; \text{ where}$$

$$X_0 = \frac{1}{2}(\mu_k + \mu_l) - \sigma^2 \frac{\mu_k - \mu_l}{\|\mu_k - \mu_l\|^2} \ln \frac{P(\omega_k)}{P(\omega_l)}$$

**Thus the linear DB is:**  $W^T (X - X_0) = 0;$

where,  $W = \mu_k - \mu_l$

**Nothing new,  
seen earlier**

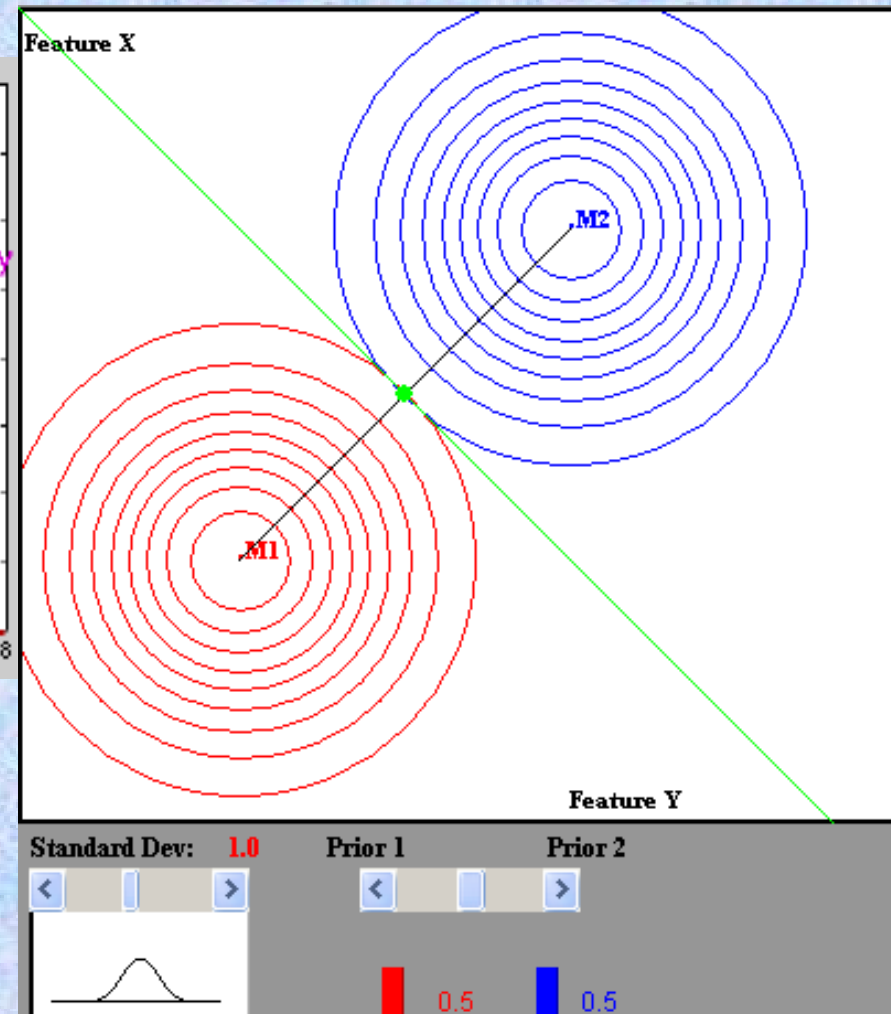
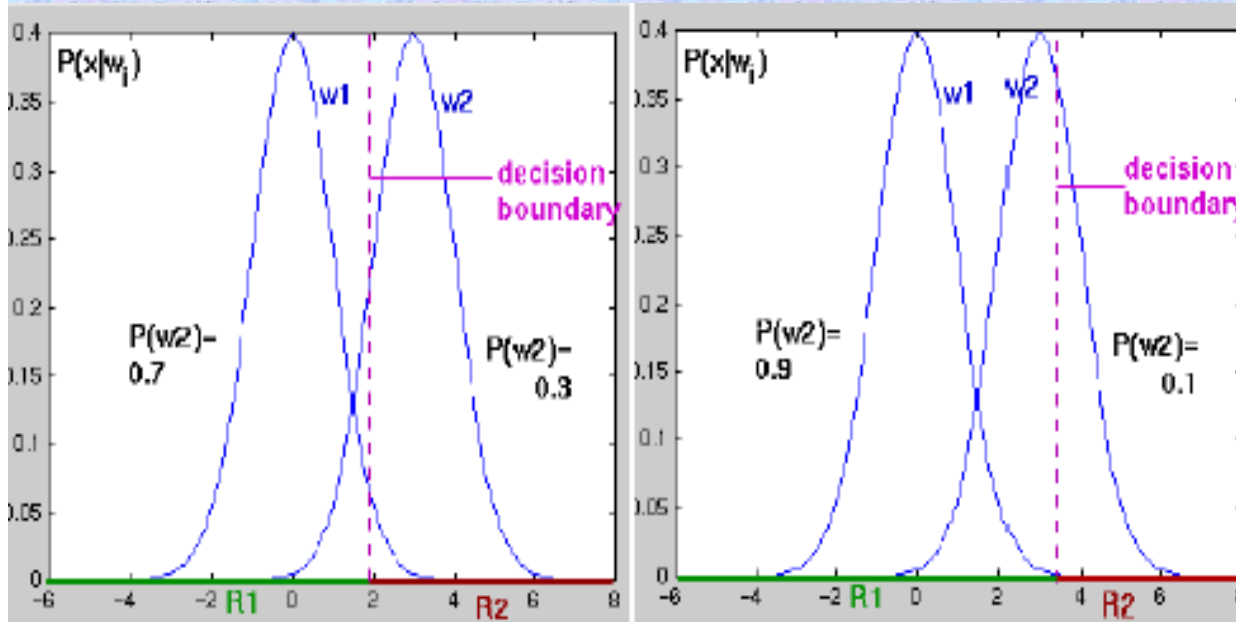
# CASE – A. – Same diagonal $\Sigma$ , with identical diagonal elements (Contd.)

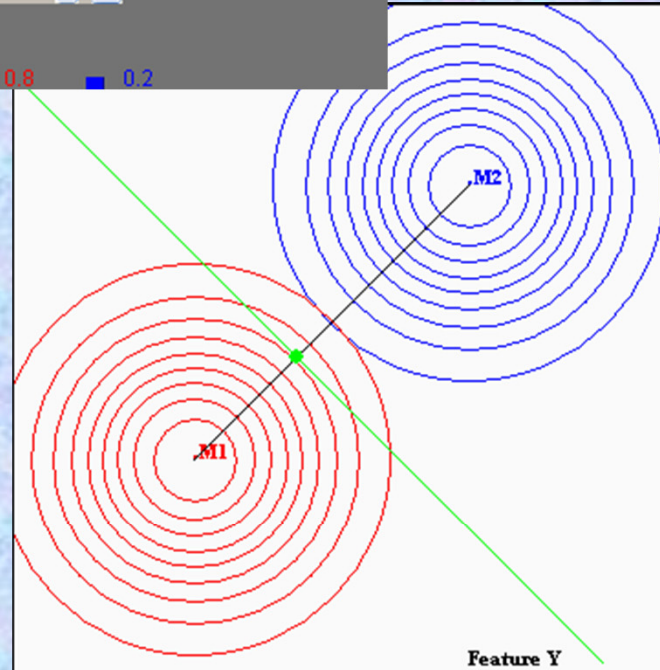
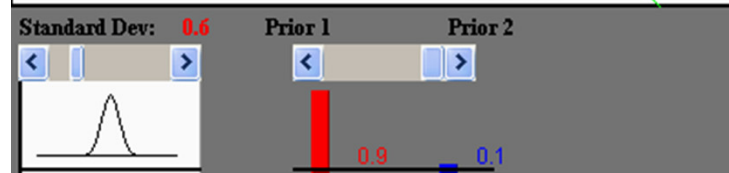
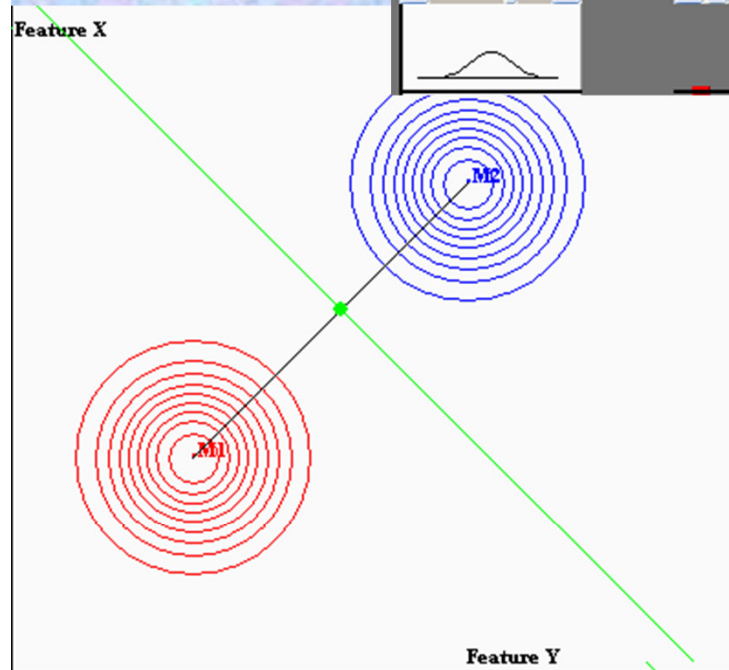
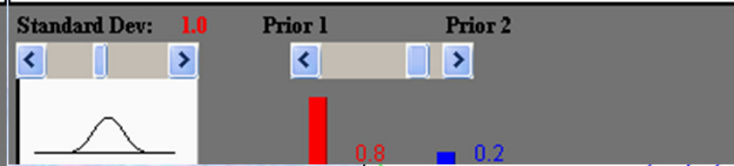
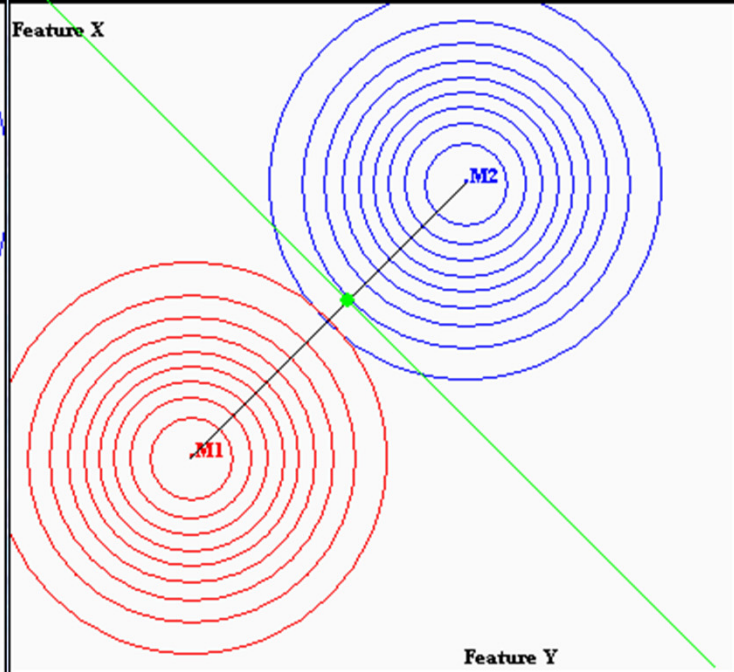
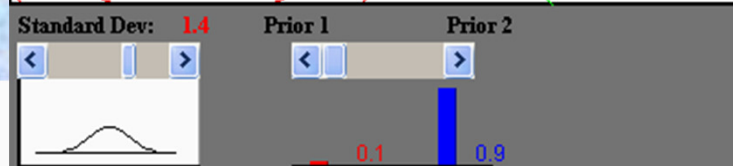
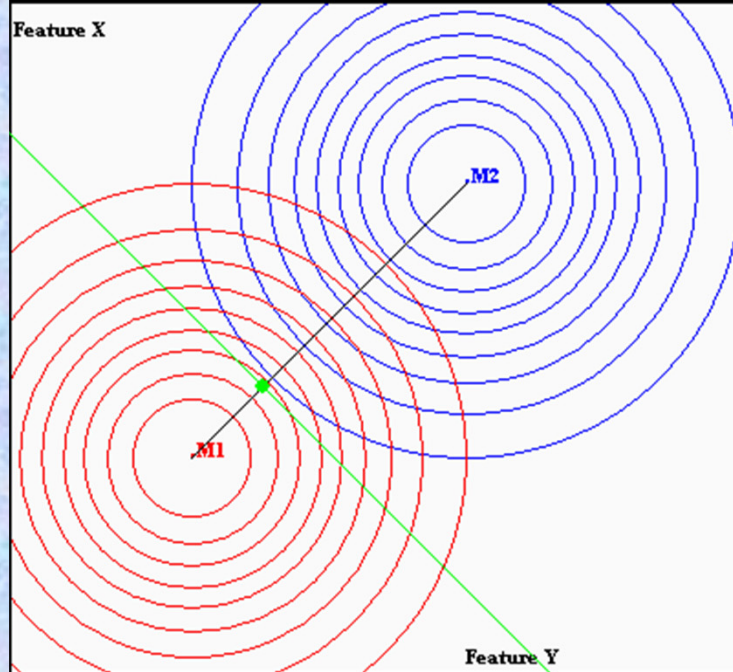
## Linear DB:

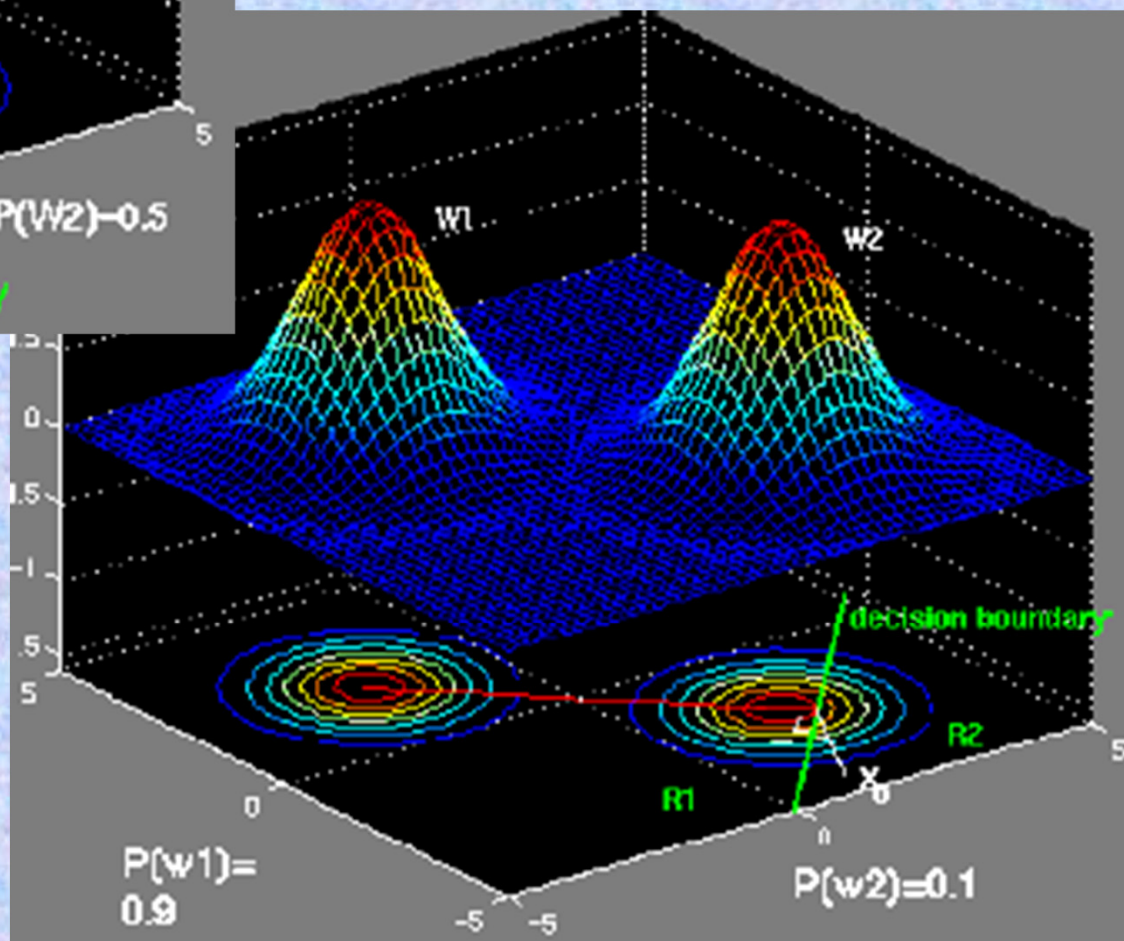
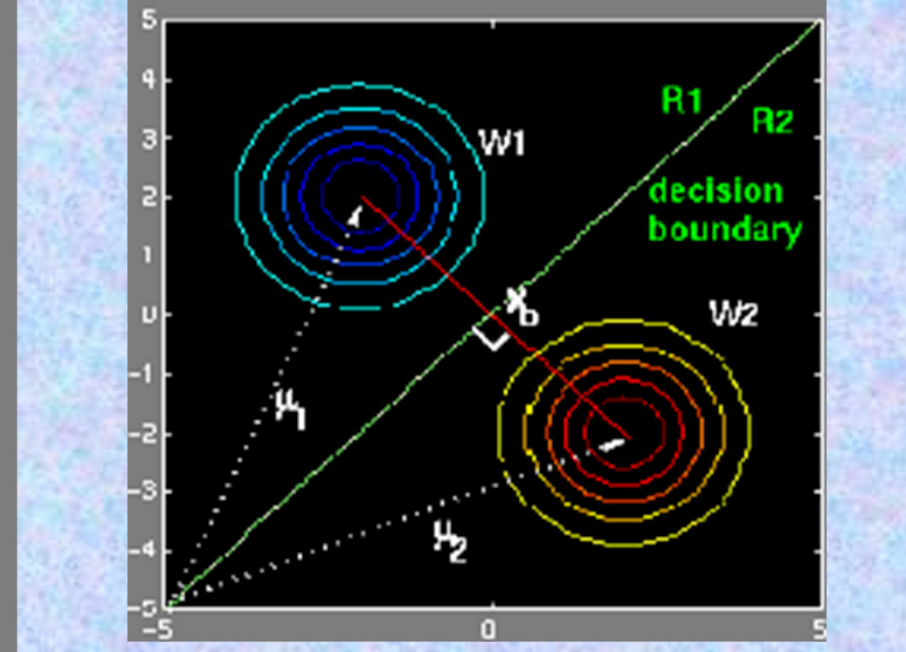
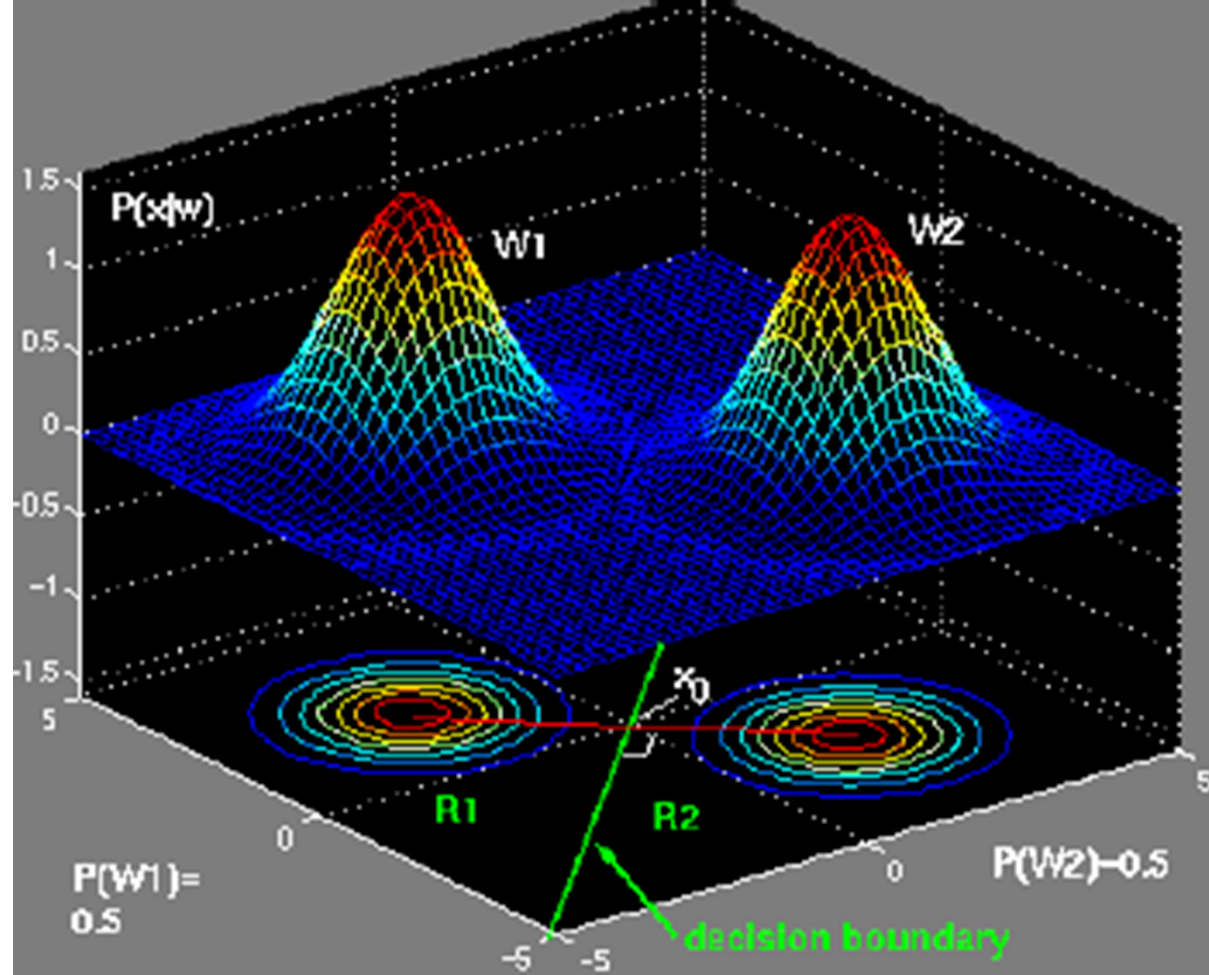
$$W^T (X - X_0) = 0;$$

where,  $W = \mu_k - \mu_l$

$$X_0 = \frac{1}{2}(\mu_k + \mu_l) - \sigma^2 \frac{\mu_k - \mu_l}{\|\mu_k - \mu_l\|^2} \ln \frac{P(\omega_k)}{P(\omega_l)}$$







## CASE – B. – Arbitrary $\Sigma$ , but identical for all class.

$$g_i(X) = \frac{-1}{2} [(X - \mu_i)^T \Sigma^{-1} (X - \mu_i)] + \ln P(w_i)$$

**Removing the class-invariant quadratic term:**

$$g_i(X) = \frac{-1}{2} \mu_i^T \Sigma^{-1} \mu_i + (\Sigma^{-1} \mu_i)^T X + \ln P(w_i)$$

Thus,  $g_i(X) = \omega_i^T X + \omega_{i0}$

where  $\omega_i = \Sigma^{-1} \mu_i$  and  $\omega_{i0} = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + \ln P(w_i)$

**The linear DB is thus:**  $g_k(X) = g_l(X), k \neq l$

**which is:**  $(\omega_k^T - \omega_l^T)X + (\omega_{k0} - \omega_{l0}) = 0;$

$(\omega_{k0} - \omega_{l0}) = (\omega_l - \omega_k)^T X_0;$  where

$$X_0 = \frac{1}{2} (\mu_k + \mu_l) - \frac{\mu_k - \mu_l}{(\mu_k - \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l)} \ln \frac{P(\omega_k)}{P(\omega_l)} \quad \leftarrow \text{Prove it.}$$



**Thus the linear DB is:**  $W^T (X - X_0) = 0$ ;

where,  $W = \omega_k - \omega_l$  where  $\omega_i = \Sigma^{-1} \mu_i$

Thus,  $W = \Sigma^{-1} (\mu_k - \mu_l)$ ;

**The normal to the DB, "W", is thus the transformed line joining the two means.**

**The transformation matrix is a symmetric  $\Sigma^{-1}$ .**

**The DB is thus -**

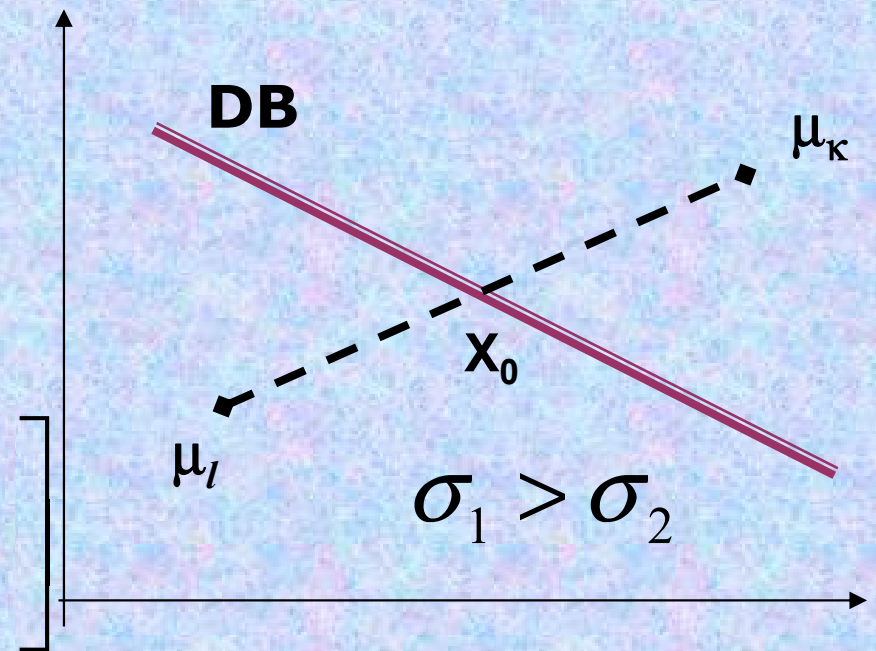
**a tilted (rotated) vector joining the two means.**

**Let  $\Sigma$  (2-D) be diagonal, with non-identical diagonal elements:  $\sigma_1$  and  $\sigma_2$ .**

Then,  $W_D = \begin{bmatrix} \phantom{0} \\ \phantom{0} \end{bmatrix}$ ;

$d = 2$  case. Direction of DB =

$\begin{bmatrix} \phantom{0} \\ \phantom{0} \end{bmatrix}$



**Thus the linear DB is:**  $W^T (X - X_0) = 0;$

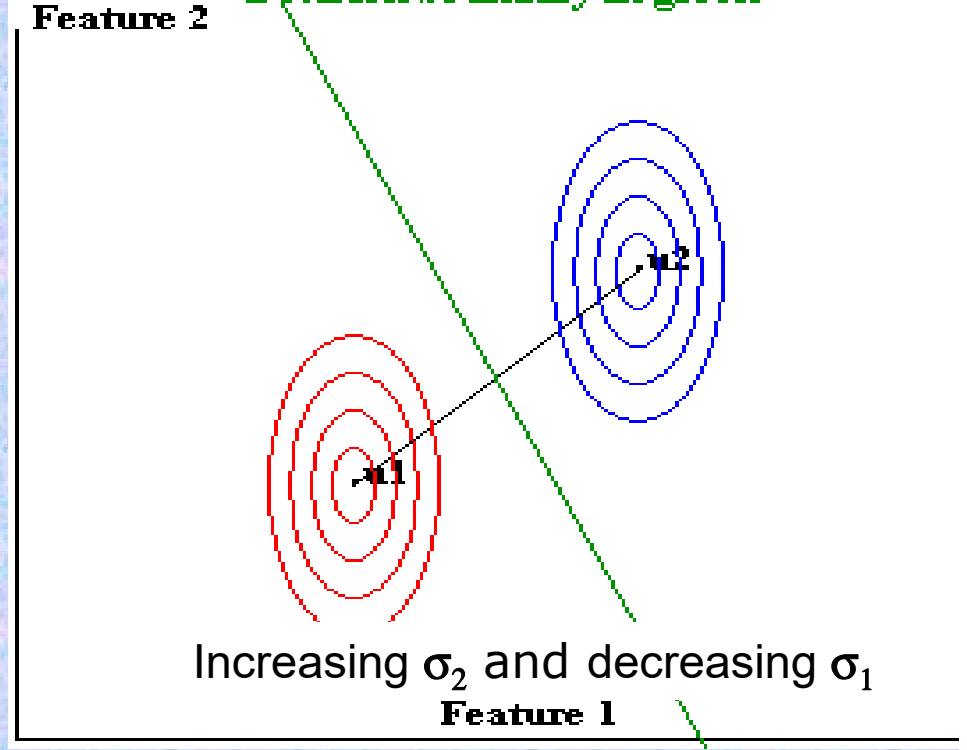
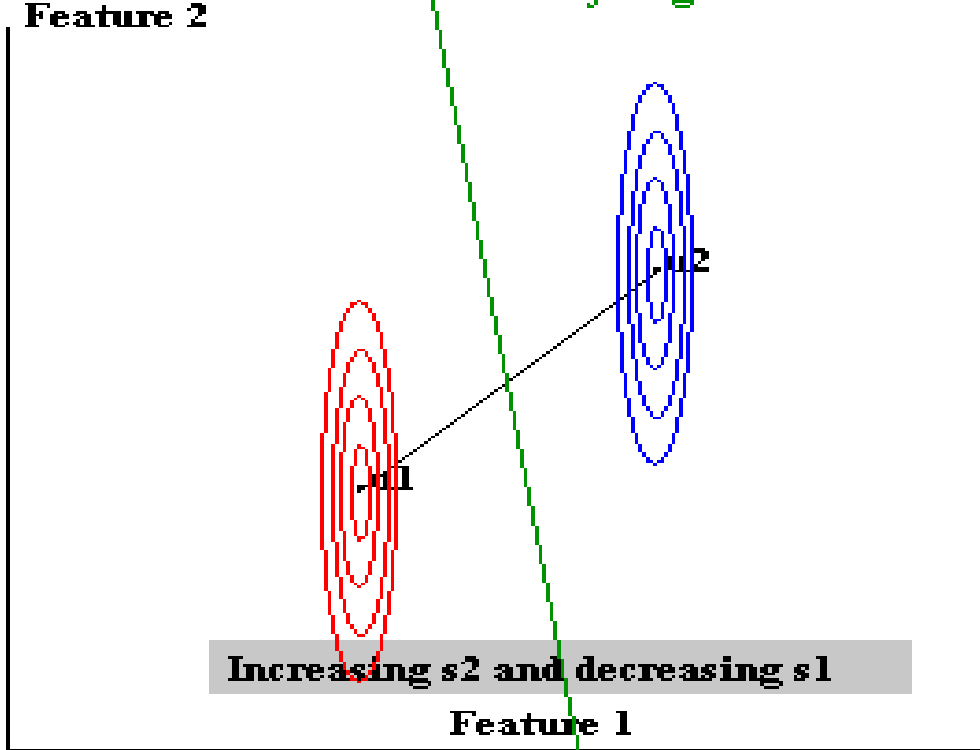
where,  $W = \omega_k - \omega_l$  where  $\omega_i = \Sigma^{-1} \mu_i$

*Thus,  $W = \Sigma^{-1} (\mu_k - \mu_l);$*

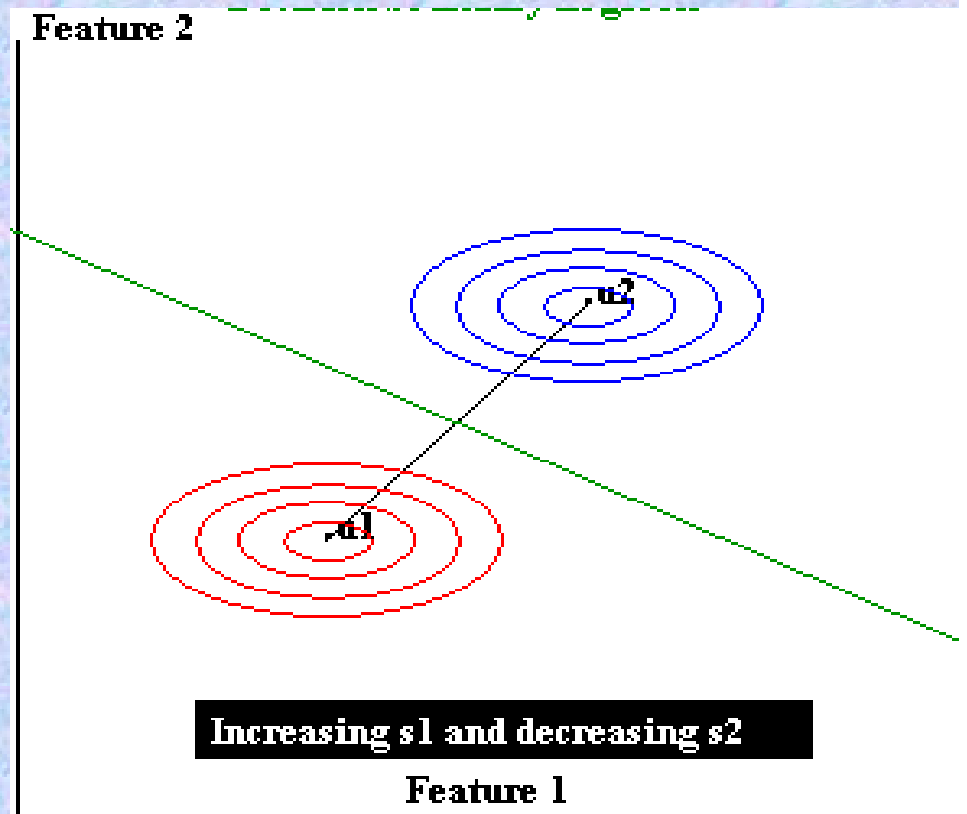
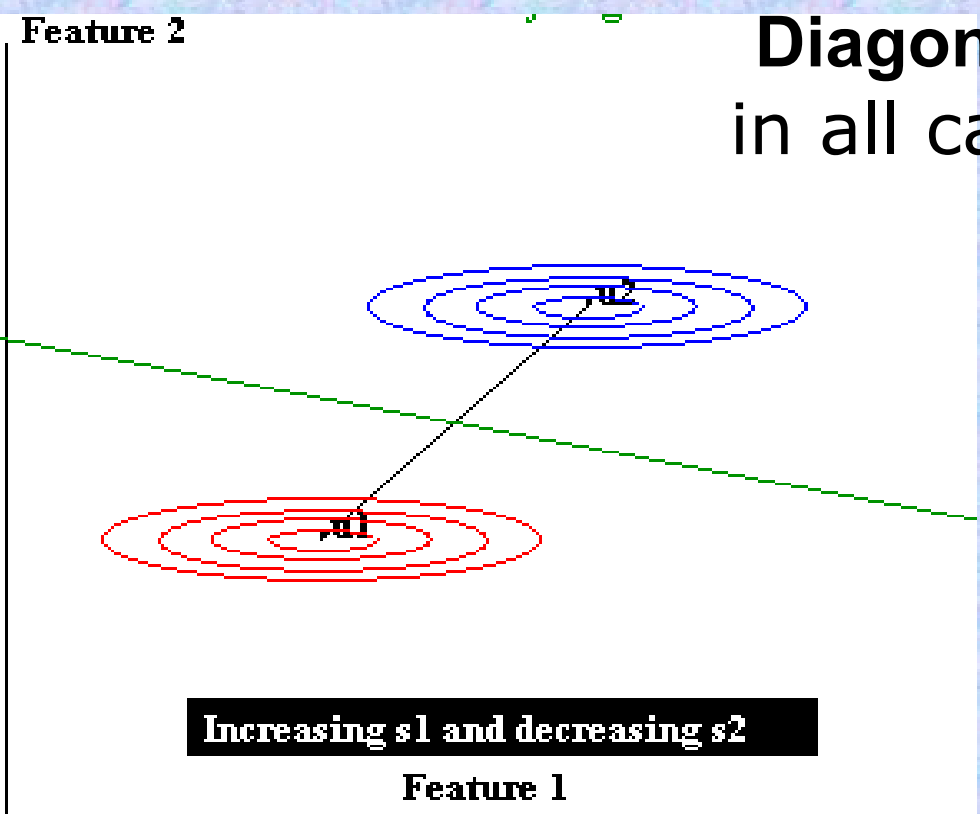
### **Special case:**

**Let,  $\Sigma$  (2-D) be arbitrary, but with diagonal elements (=1).**

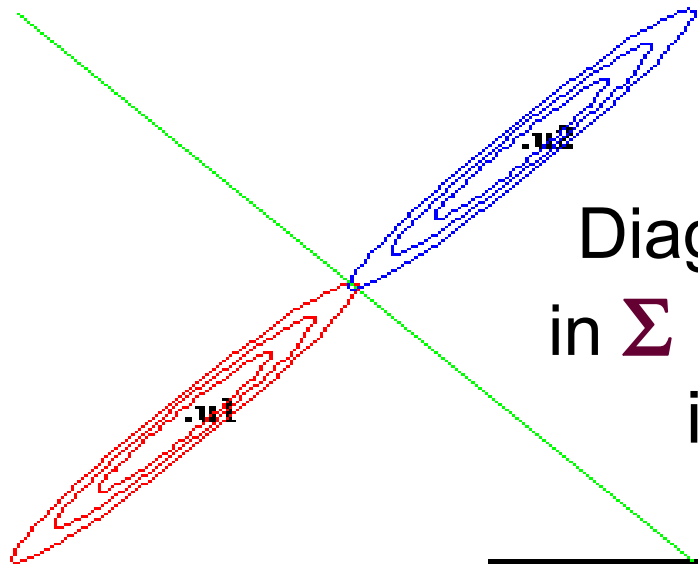
Solve for **W** in this case, and compare with the diagonal  $\Sigma$  case.



Diagonal  $\Sigma$   
in all cases.



Feature 2

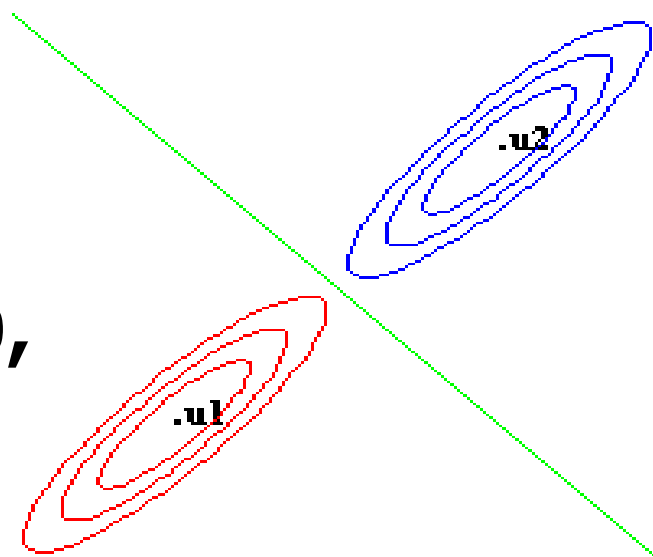


Feature 1

Covariance is 0.8

Diagonal elements  
in  $\Sigma$  are both **1.0**,  
in all cases

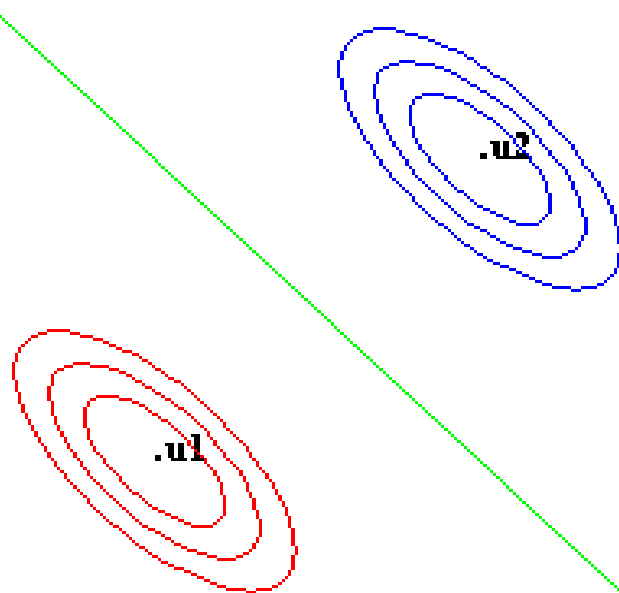
Feature 2



Feature 1

Covariance is 0.600

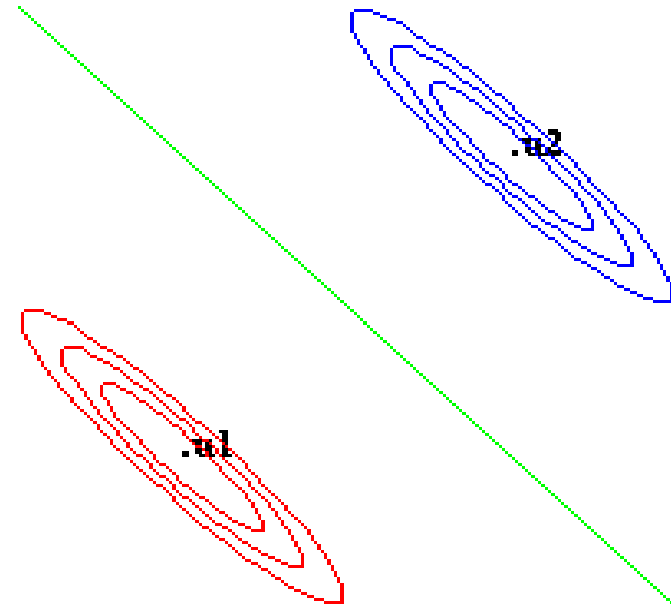
Feature 2



Feature 1

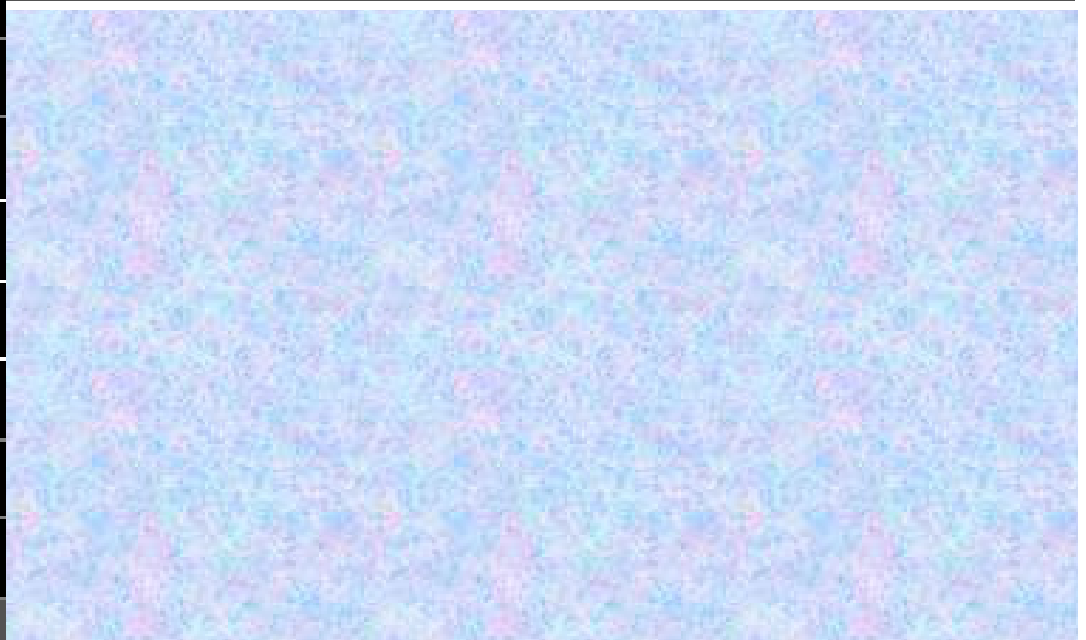
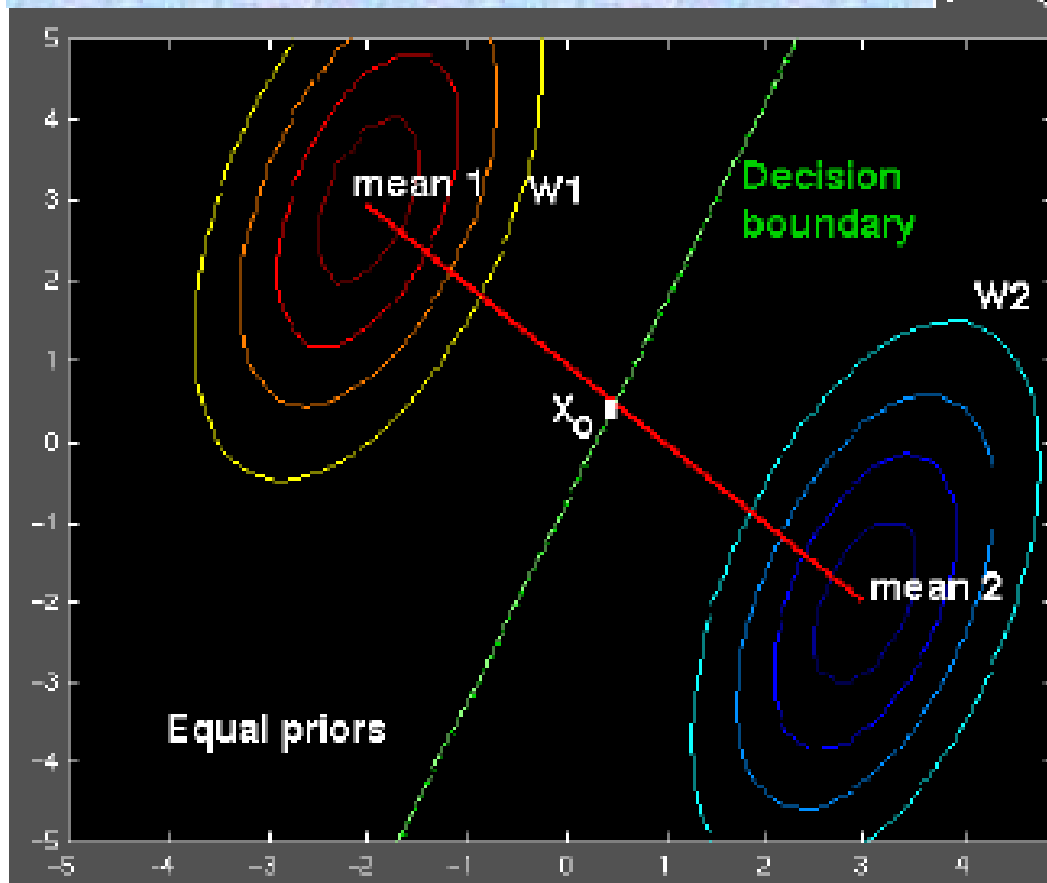
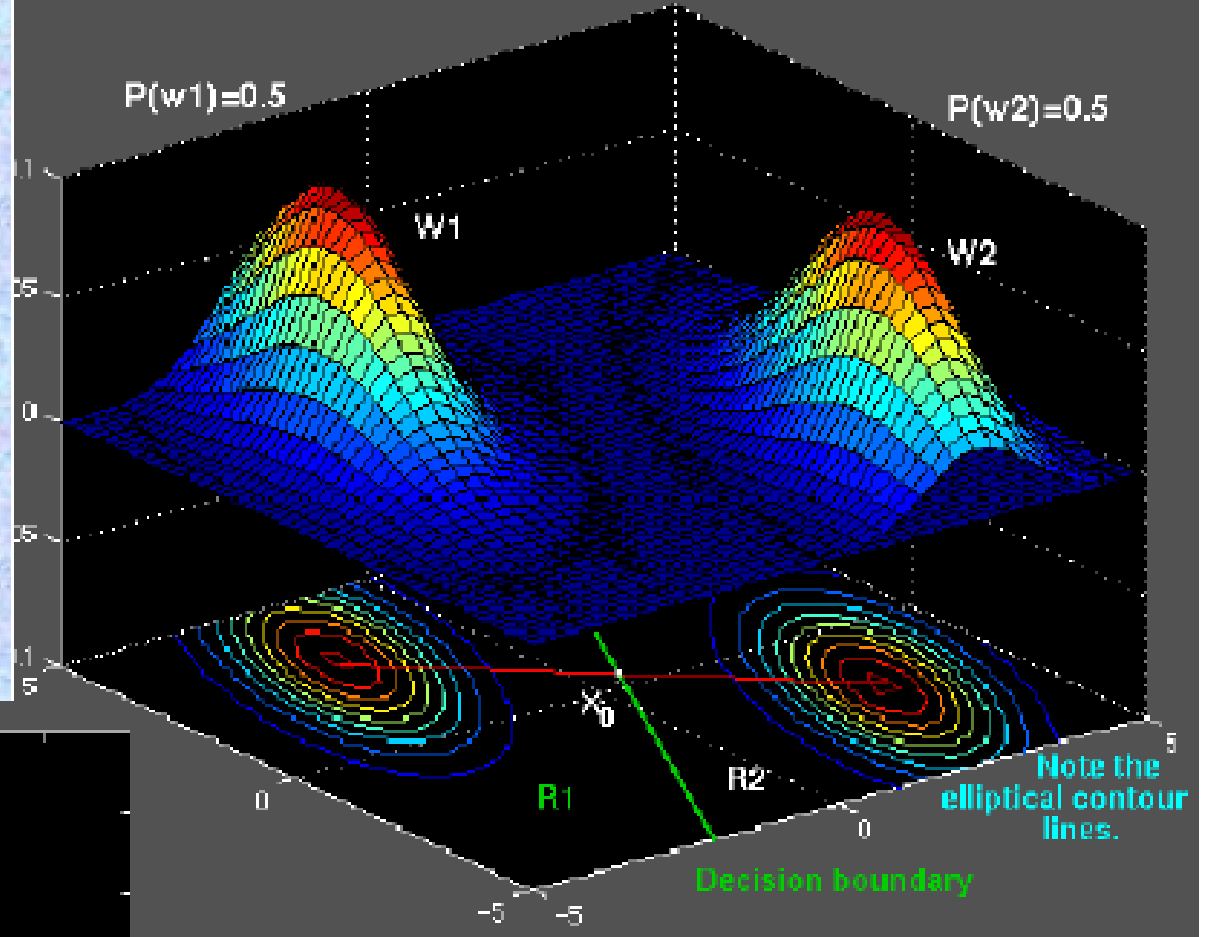
Covariance is -0.39

Feature 2



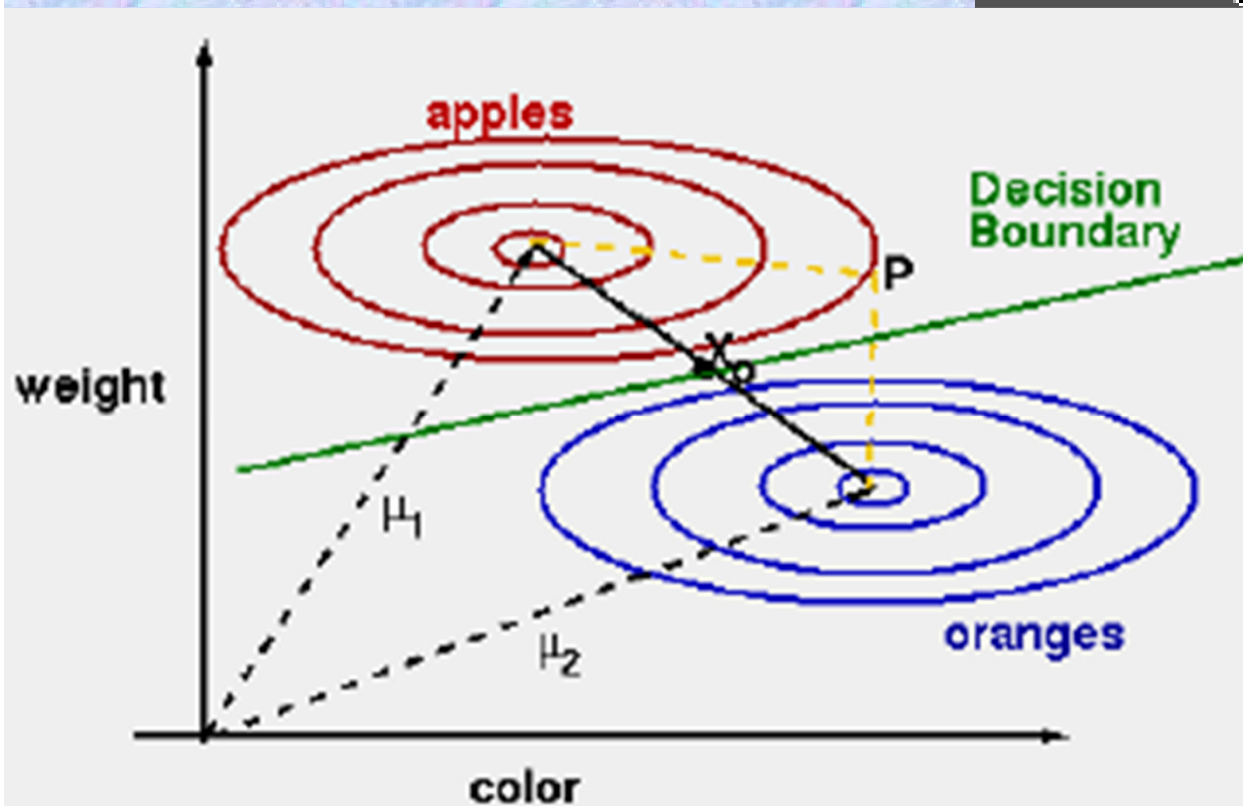
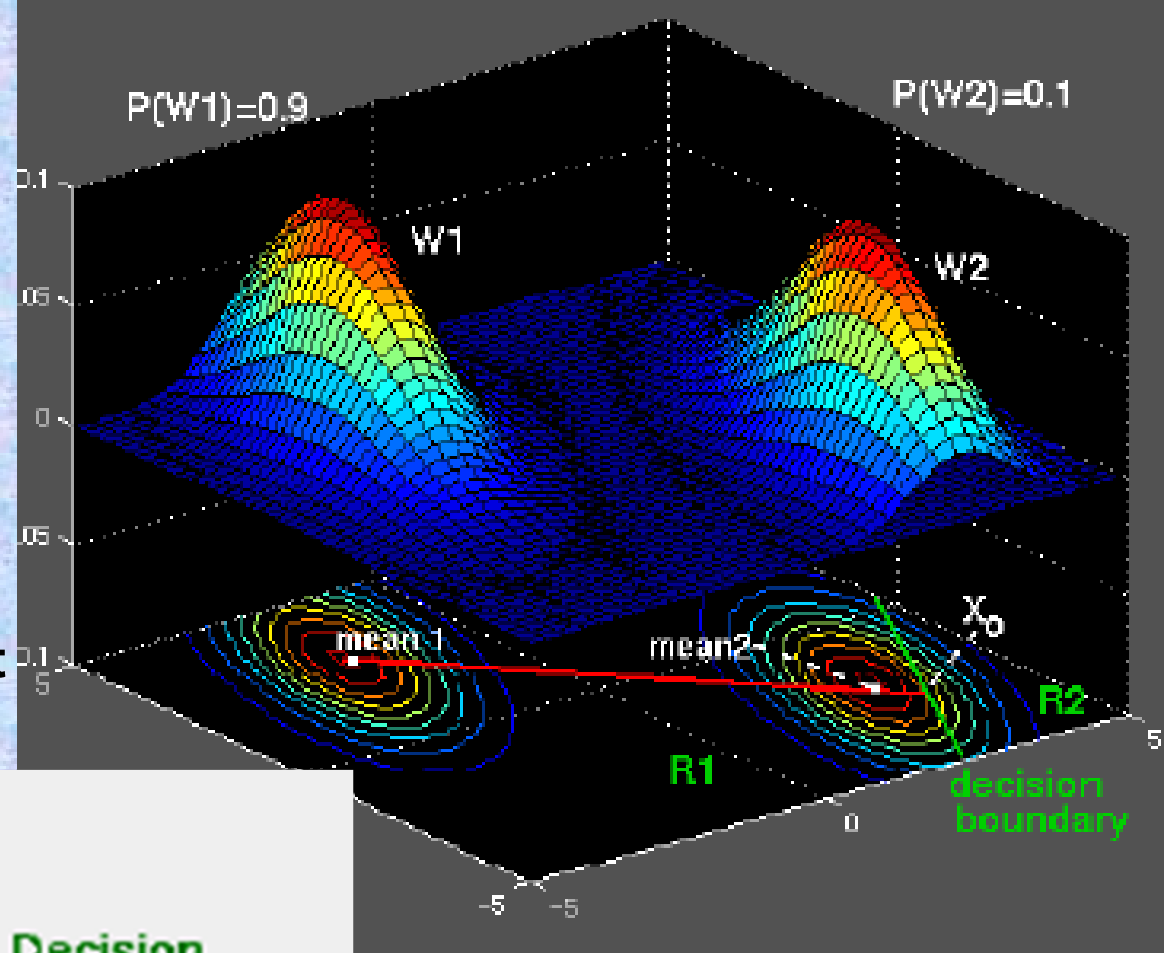
Feature 1

Covariance is -0.69



Point P is actually closer (in the Euclidean sense) to the mean for the Orange class.

The discriminant function evaluated at P is smaller for class 'apple' than it is for class 'orange'.



## CASE C. – Arbitrary $\Sigma$ , all parameters are class dependent.

$$g_i(X) = \frac{-1}{2} [(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)] - \frac{-1}{2} \ln |\Sigma_i| + \ln P(w_i)$$

Thus,  $g_i(X) = X^T W_i X + \omega_i^T X + \omega_{i0}$ ;

where  $W_i = \frac{-1}{2} \Sigma_i^{-1}$ ;

$\omega_i = \Sigma_i^{-1} \mu_i$  and

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

**The DBs and DFs are hyper-quadrics.  $g_k(X) = g_l(X), k \neq l$**


**We shall first look into a few cases of such surfaces next.**

**Example [Duda, Hart]:**

$$\mu_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}; \quad \Sigma_1 = \begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix};$$

**Draw and Visualize (qualitatively)  
the iso-contours**

$$\mu_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}; \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix};$$

$$\Sigma_1^{-1} =$$

$$\Sigma_2^{-1} =$$

**Assume;  $P(w_1) = P(w_2) = 0.5$ ;**

**Get expression of DB:**



# Quadratic Decision Boundaries

In  $\mathbb{R}^d$  with  $\mathbf{X} = (x_1, x_2, \dots, x_d)^T$ , consider the equation:

$$\sum_{i=1}^d w_{ii} x_i^2 + \sum_{i=1}^{d-1} \sum_{j=i+1}^d w_{ij} x_i x_j + \sum_{i=1}^d w_i x_i + w_0 = 0 \quad ..1$$

The above equation is defined by a quadric discriminant function, which yields a quadric surface.

If  $d=2$ ,  $\mathbf{X} = (x_1, x_2)^T$  equation (1) becomes:

$$w_{11} x_1^2 + w_{22} x_2^2 + w_{12} x_1 x_2 + w_1 x_1 + w_2 x_2 + w_0 = 0 \quad ..2$$

## Special cases of equation:

$$w_{11}x_1^2 + w_{22}x_2^2 + w_{12}x_1x_2 + w_1x_1 + w_2x_2 + w_0 = 0 \quad ..2$$

Case 1:

$w_{11} = w_{22} = w_{12} = 0$ ; Eqn. (2) defines a line.

Case 2:

defines a circle.

Case 3:

$w_{11} = w_{22} = 1$ ;  $w_{12} = w_1 = w_2 = 0$ ; defines a circle whose center is at the origin.

Case 4:

$w_{11} = w_{22} = 0$ ; defines a bilinear constraint.

Case 5:

$w_{11} = w_{12} = w_2 = 0$ ; defines a parabola with a specific orientation.

Case 6:

$w_{11} \neq 0, w_{22} \neq 0, w_{11} \neq w_{22}; w_{12} = w_1 = w_2 = 0$   
defines a simple ellipse.

Selecting suitable values of  $w_i$ 's, gives other conic sections; Hyperbolic ??

For  $d \geq 3$ , we define a family of hyper-surfaces in  $\mathbb{R}^d$ .

$$\sum_{i=1}^d w_{ii} x_i^2 + \sum_{i=1}^{d-1} \sum_{j=i+1}^d w_{ij} x_i x_j + \sum_{i=1}^d w_i x_i + \omega_o = 0 \quad ..1$$

In the above equation, the total number of parameters is: ??

$$2d + 1 + d(d-1)/2 = (d+1)(d+2)/2.$$

Organize these parameters, and manipulate the equation to obtain:

$$\overline{X}^T W \overline{X} + w^T \overline{X} + \omega_o = 0 \quad ..3$$

$w$  has  $d$  terms,  $\omega_o$  has **one** term, and  $W$  ( $w_{ij}$ ) is a  $d \times d$  matrix as:

$(d^2-d)$  non-diagonal terms of the matrix  $W$ ,  
is obtained by duplicating (split into two parts):  
 $d(d-1)/2$   $w_{ij}$ s.

$$w_{ij} = \begin{cases} w_{ii} & \text{if } i = j \\ \frac{1}{2} w_{ij} & \text{if } i \neq j \end{cases}$$

In equation 3, the symmetric part of matrix  $W$ , contributes to the Quadratic terms. Equation 3 generally defines a hyperhyperboloidal surface.

If  $W = I/0$ , we get a hyper-spheres/planes.

$$\overline{X}^T W \overline{X} + w^T \overline{X} + \omega_o = 0$$

$$\vec{d}_i^2 = (X - \mu_i)^T \Sigma^{-1} (X - \mu_i) = -X^T \Sigma^{-1} X + 2\mu_i^T \Sigma^{-1} X - \mu_i^T \Sigma^{-1} \mu_i$$

**Example of linearization:**

$$g(X) = x_2 - x_1^2 - 3x_1 + 6 = 0$$

To **Linearize**, let  $x_3 = x_1^2$ . Then:

$$g(X) = x_2 - x_3 - 3x_1 + 6 = W^T X + w_o$$

where,  $X = [x_1, x_2, x_3]^T$

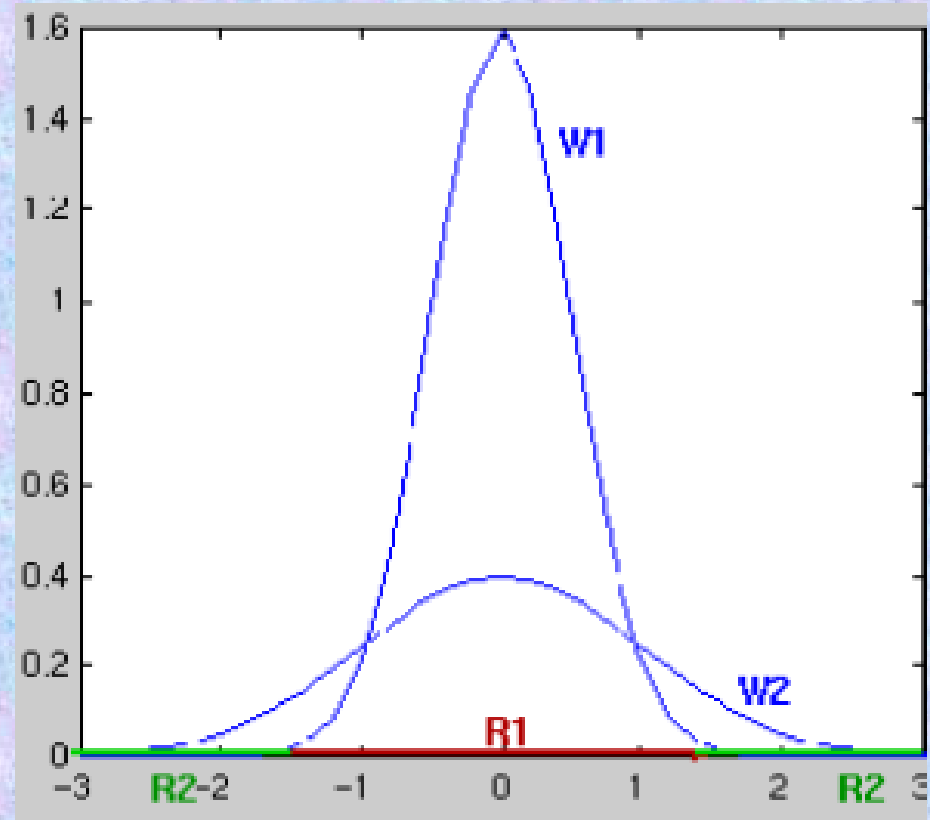
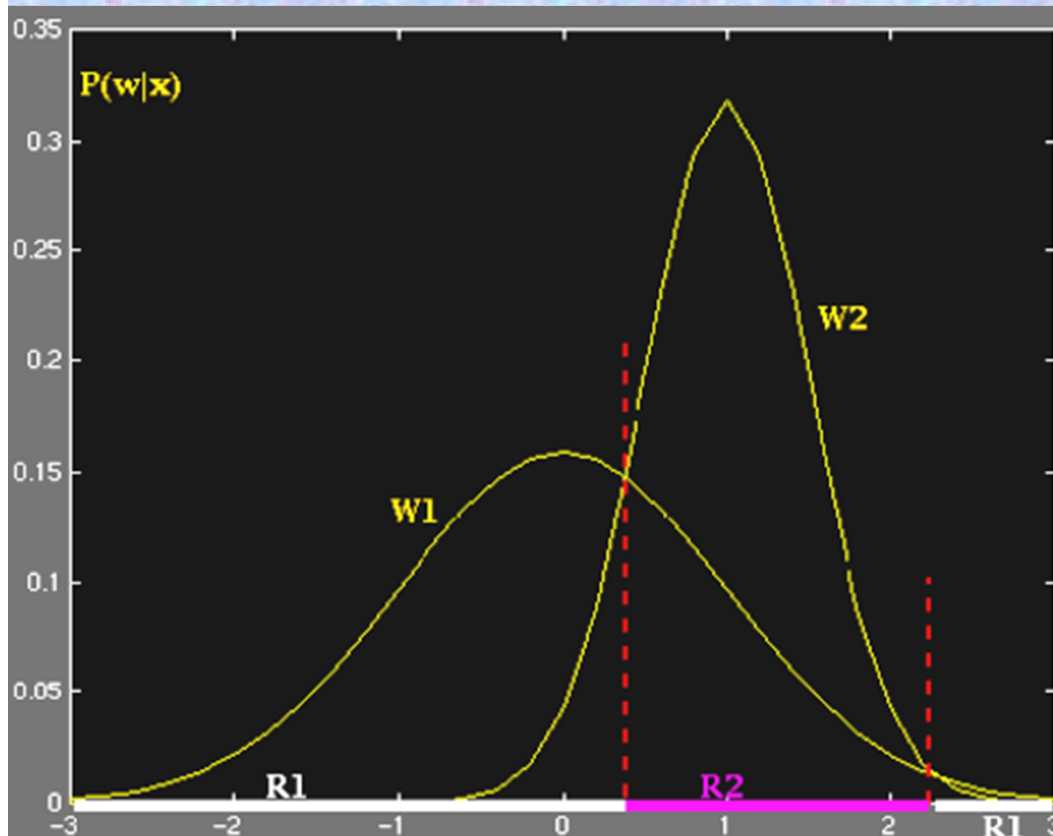
and  $W^T = [-3, 1, -1]$

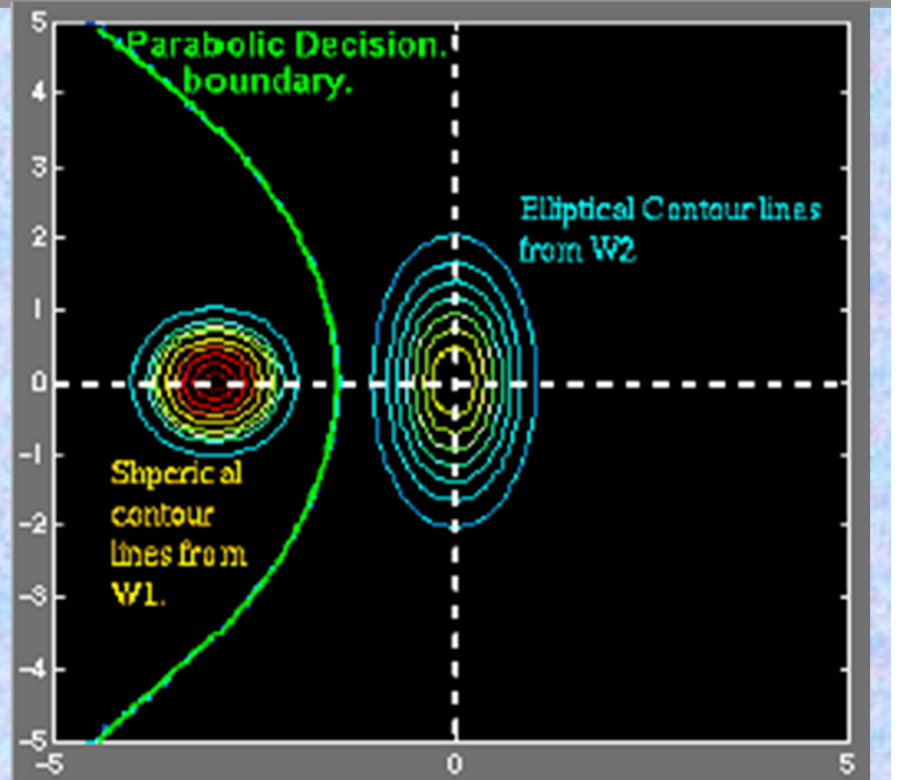
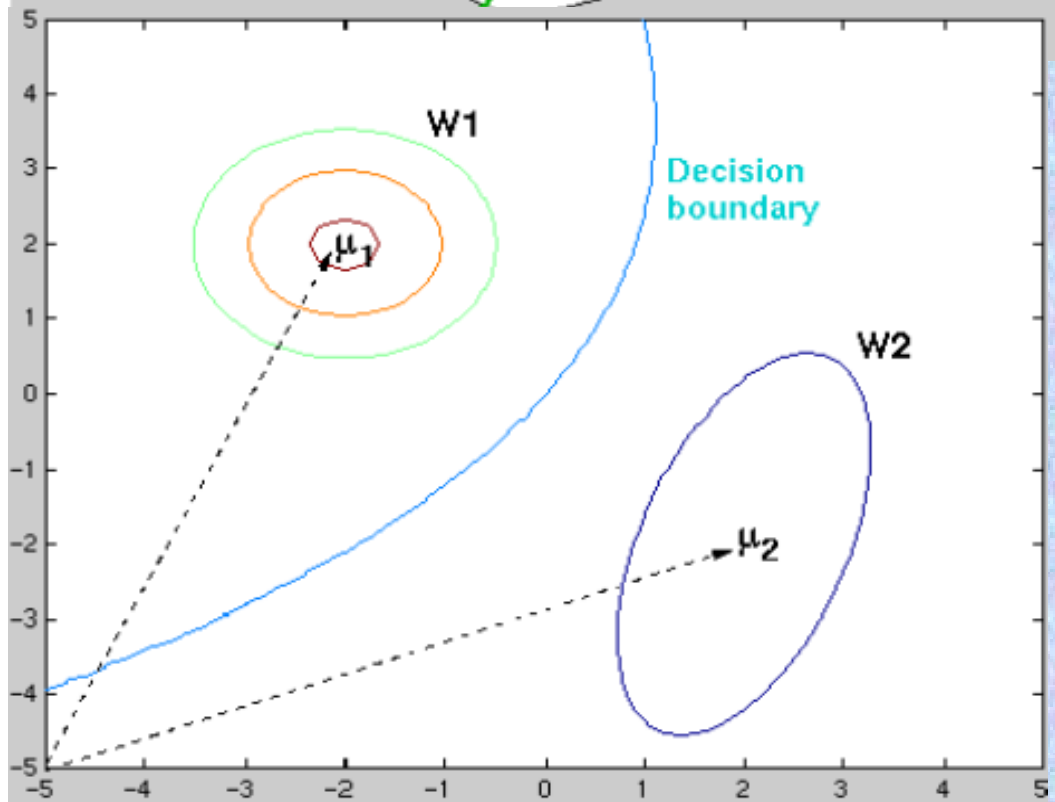
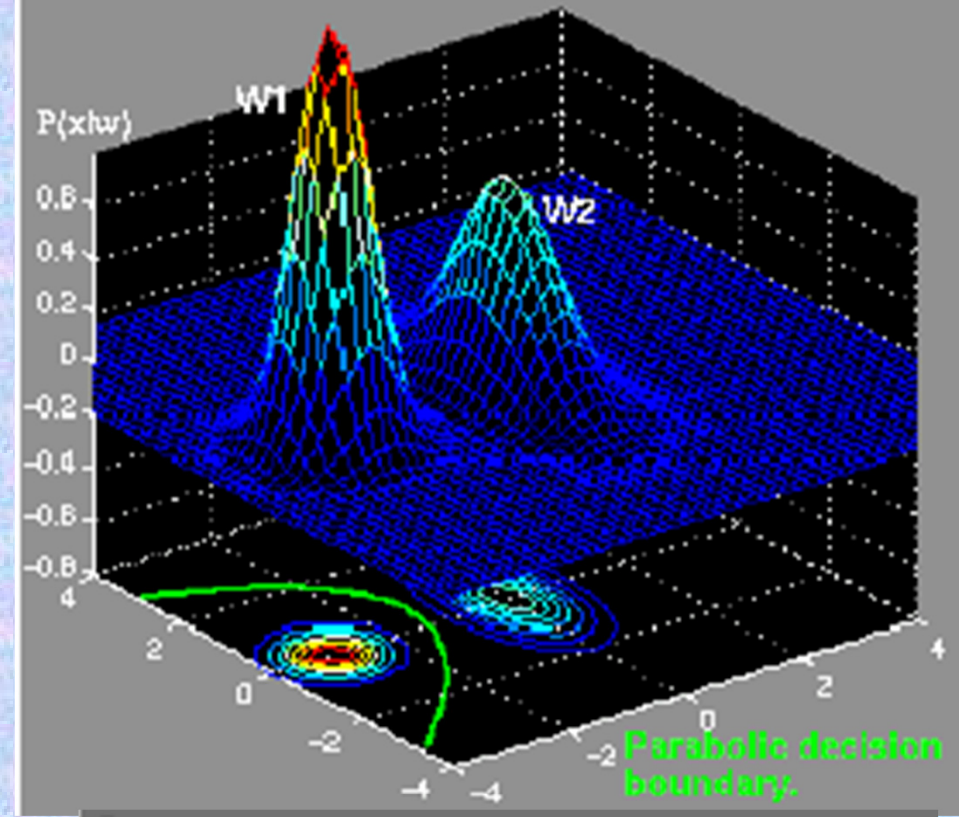
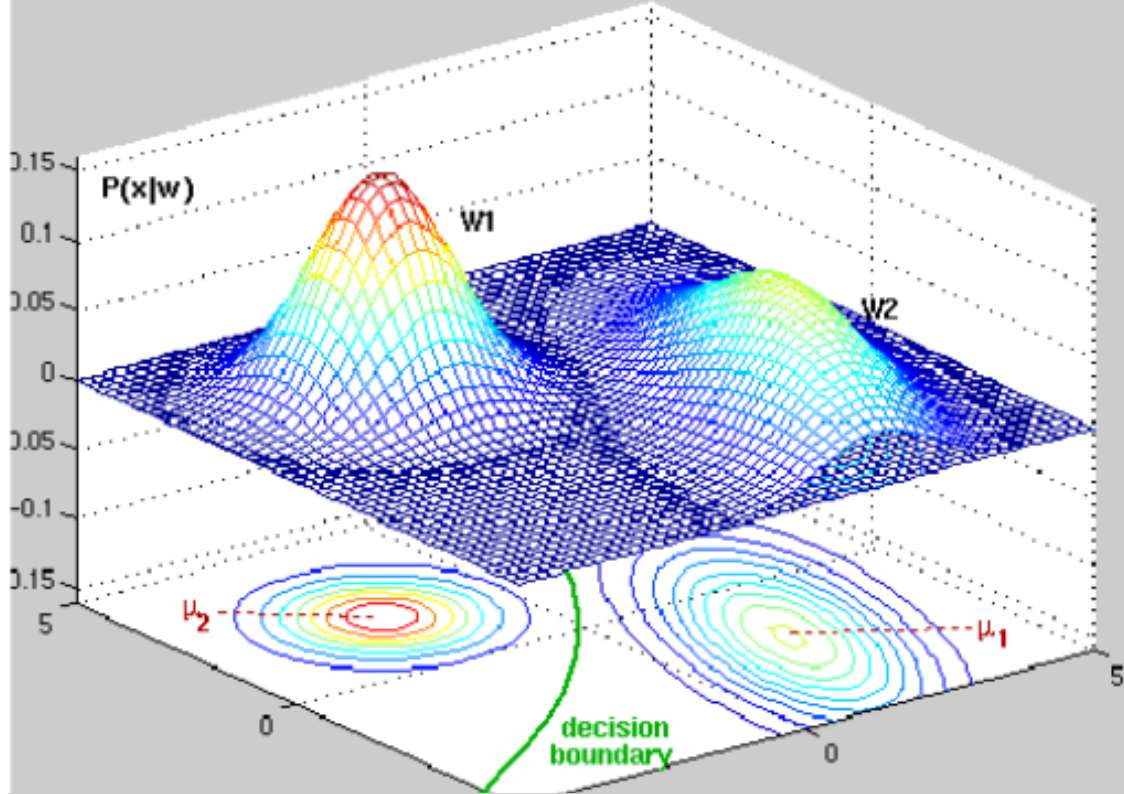
**CASE – C. – Arbitrary  $\Sigma$ , all parameters are class dependent – contd..**

$$g_i(X) = \frac{-1}{2} [(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i)] - \frac{-1}{2} \ln |\Sigma_i| + \ln P(w_i)$$

Thus,  $g_i(X) = X^T W_i X + \omega_i^T X + \omega_{i0}$ ; where  $W_i = \frac{-1}{2} \Sigma_i^{-1}$ ;

$$\omega_i = \Sigma_i^{-1} \mu_i \quad \text{and} \quad \omega_{i0} = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln P(w_i)$$

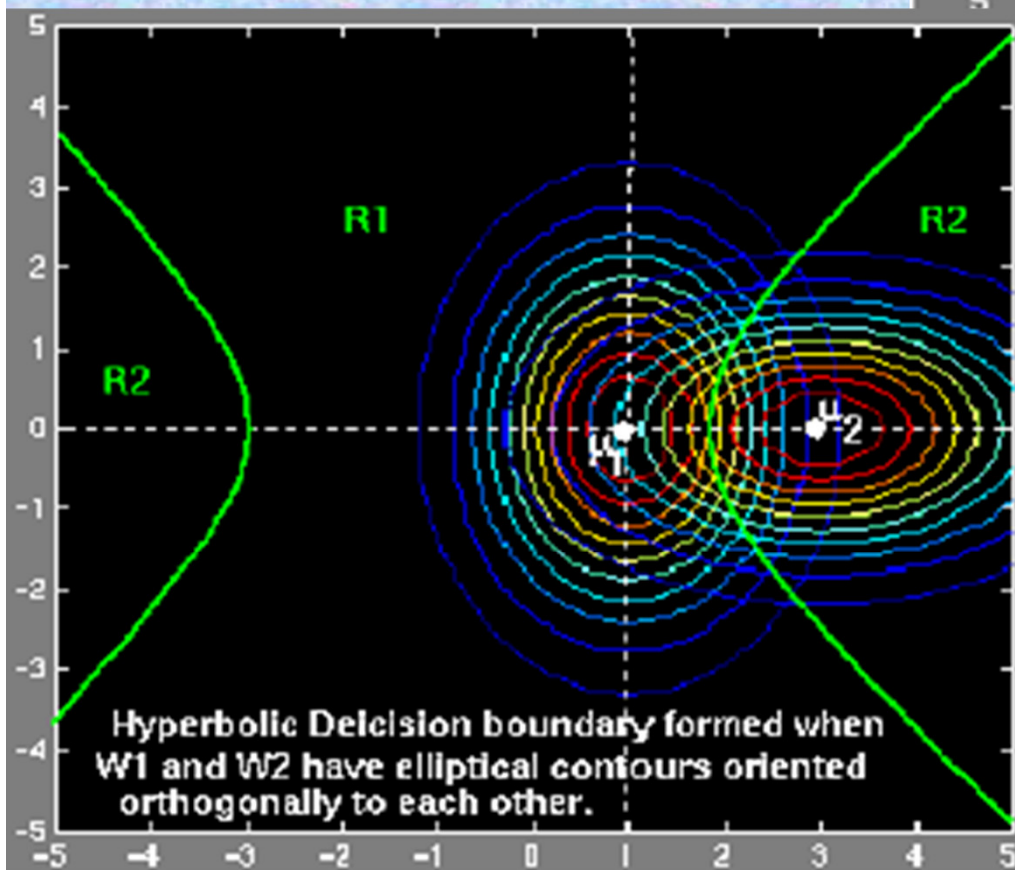
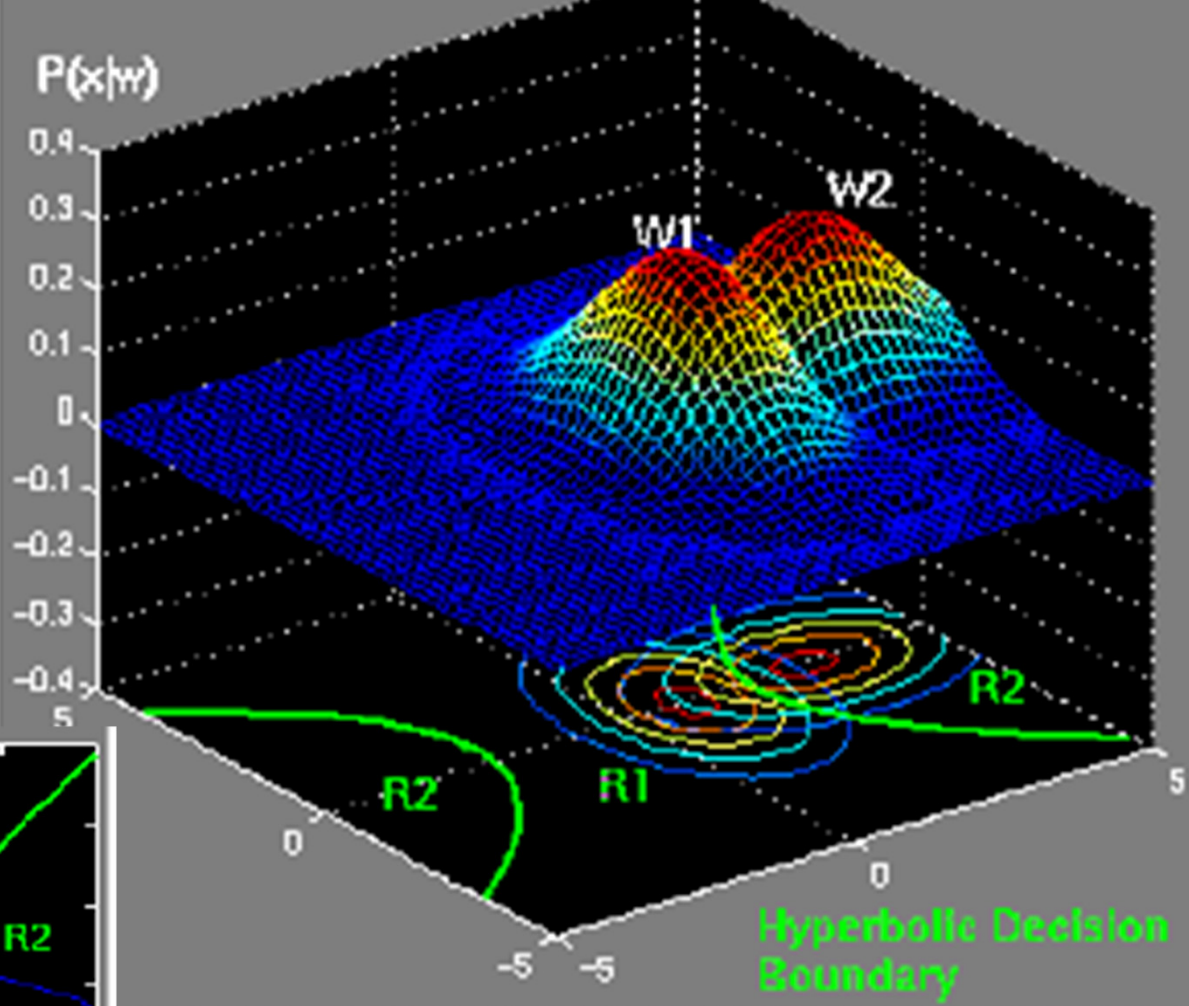


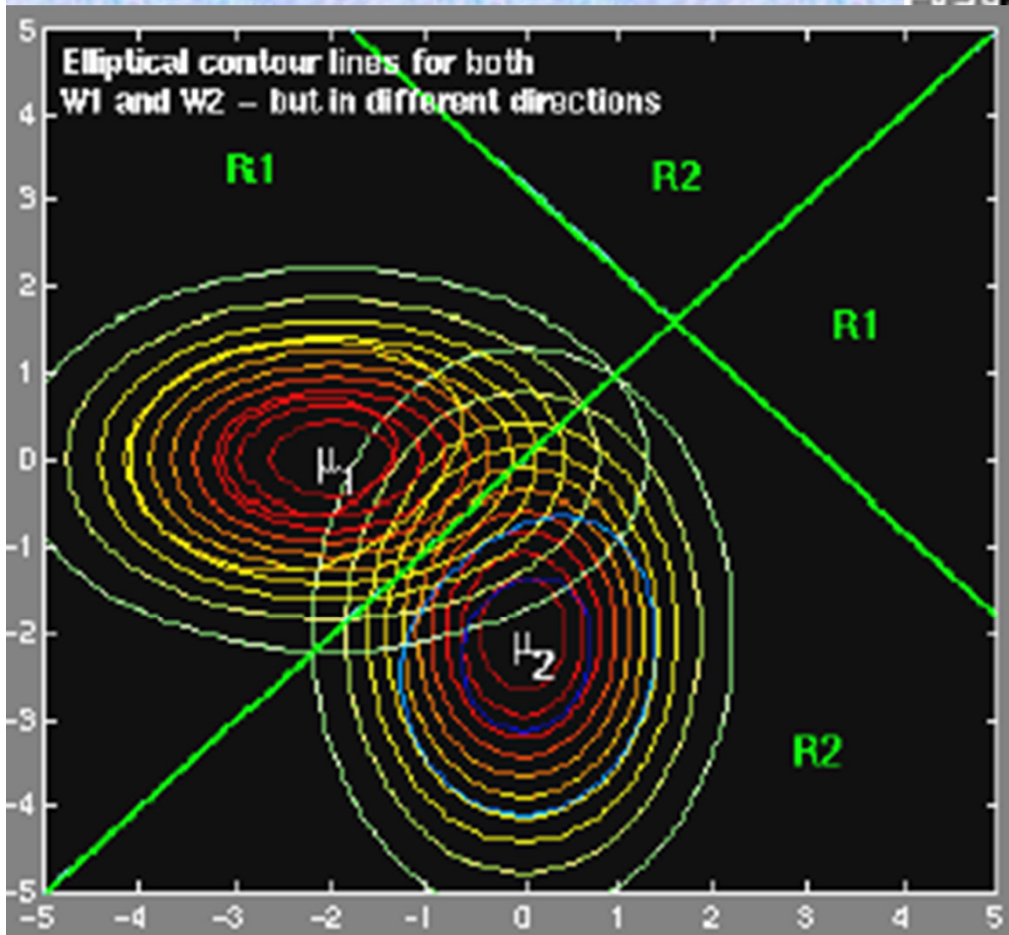
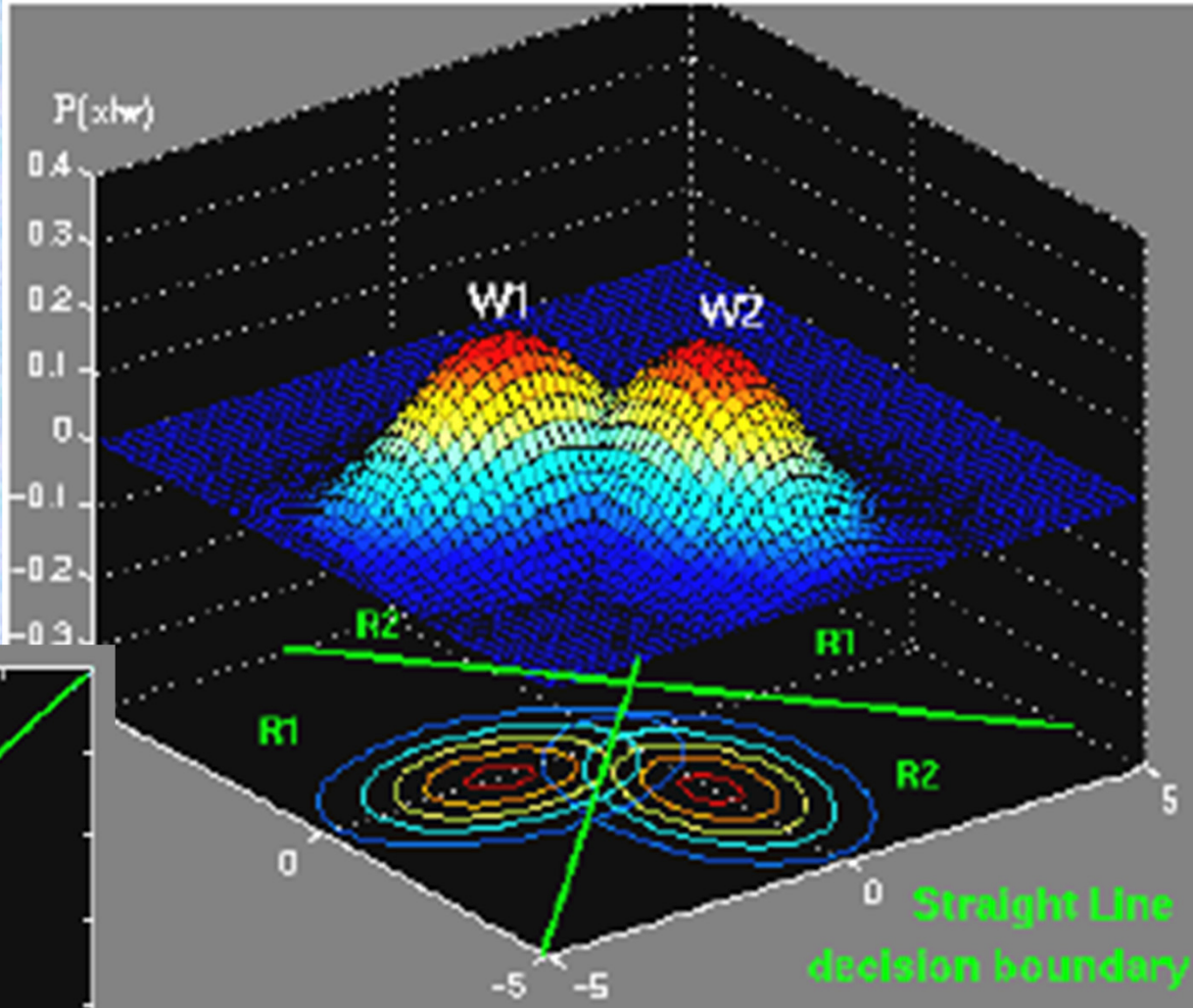


$$\sigma_1^x = \sigma_2^y; \sigma_1^y = \sigma_2^x;$$

$$\rho_1 = \rho_2 = 0;$$

$$\mu_1^x < \mu_2^x; \mu_1^y = \mu_2^y;$$





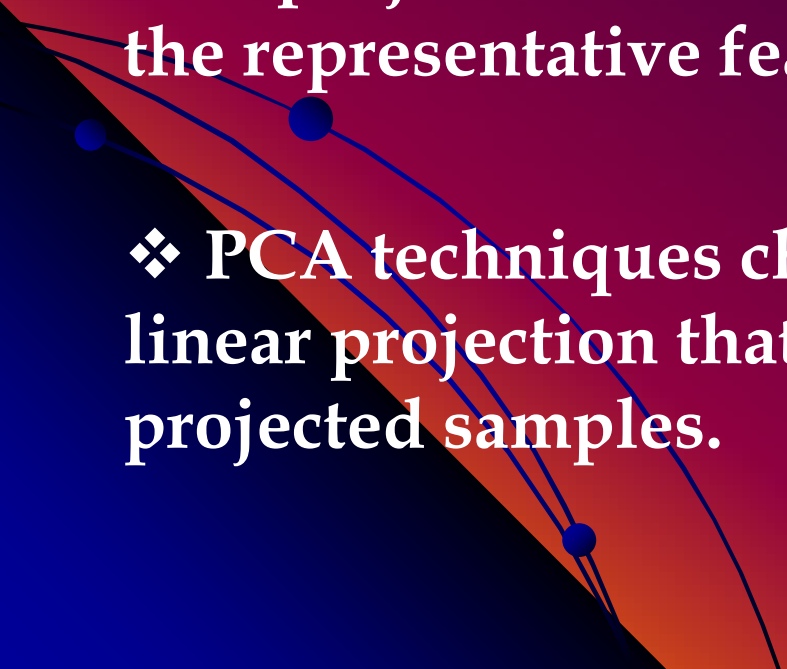
$$\sigma_1^x = \sigma_2^y; \sigma_1^y = \sigma_2^x;$$

$$\rho_1 = \rho_2 = 0;$$

$$\mu_1^x = \mu_2^x \pm C; \mu_1^y = \mu_2^y \mp C;$$



# Principal Component Analysis

- ❖ Eigen analysis, Karhunen-Loeve transform
  - ❖ **Eigenvectors:** derived from Eigen decomposition of the **scatter matrix**
  - ❖ A projection set that best explains the distribution of the representative features of an object of interest.
  - ❖ PCA techniques choose a dimensionality-reducing linear projection that maximizes the scatter of all projected samples.
- 

# Principal Component Analysis Contd.

- Let us consider a set of  $N$  sample images  $\{x_1, x_2, \dots, x_N\}$  taking values in  $n$ -dimensional image space.
- Each image belongs to one of  $c$  classes  $\{X_1, X_2, \dots, X_c\}$ .
- Let us consider a linear transformation, mapping the original  $n$ -dimensional *image space* to  $m$ -dimensional *feature space*, where  $m < n$ .
- The new feature vectors  $y_k \in R^m$  are defined by the linear transformation –

$$y_k = W^T x_k \quad k = 1, 2, \dots, N$$

where,  $W \in R^{n \times m}$  is a matrix with orthogonal columns representing the basis in feature space.

# Principal Component Analysis Contd..

- Total scatter matrix  $S_T$  is defined as

$$S_T = \sum_{k=1}^N (x_k - \mu)(x_k - \mu)^T$$

where,  $N$  is the number of samples, and  $\mu \in R^n$  is the mean image of all samples.

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$$

- The scatter of transformed feature vectors  $\{y_1, y_2, \dots, y_N\}$  is  $W^T S_T W$ .

- In PCA,  $W_{opt}$  is chosen to maximize the determinant of the total scatter matrix of projected samples, *i.e.*,

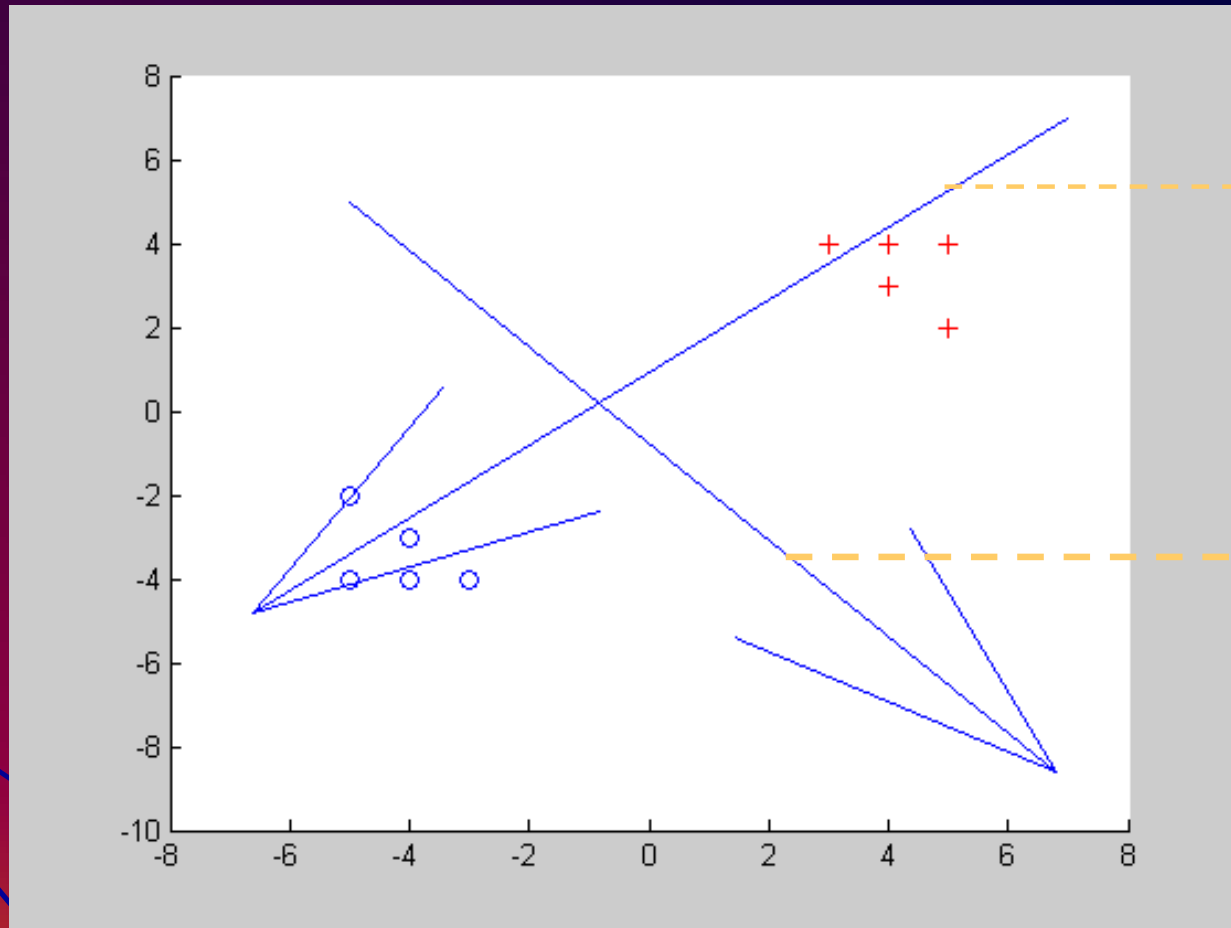
$$W_{opt} = \arg \max_W |W^T S_T W|$$

where  $\{w_i \mid i=1,2,\dots,m\}$  is the set of  $n$  dimensional eigenvectors of  $S_T$  corresponding to  $m$  largest eigenvalues (check proof).

# Principal Component Analysis Contd.

- Eigenvectors are called eigen images/pictures and also basis images/facial basis for faces.
  - Any data (say, face) can be reconstructed approximately as a weighted sum of a small collection of images that define a facial basis (eigen images) and a mean image of the face.
    - Data form a scatter in the feature space through projection set (eigen vector set)
    - Features (eigenvectors) are extracted from the training set without prior class information
- Unsupervised learning

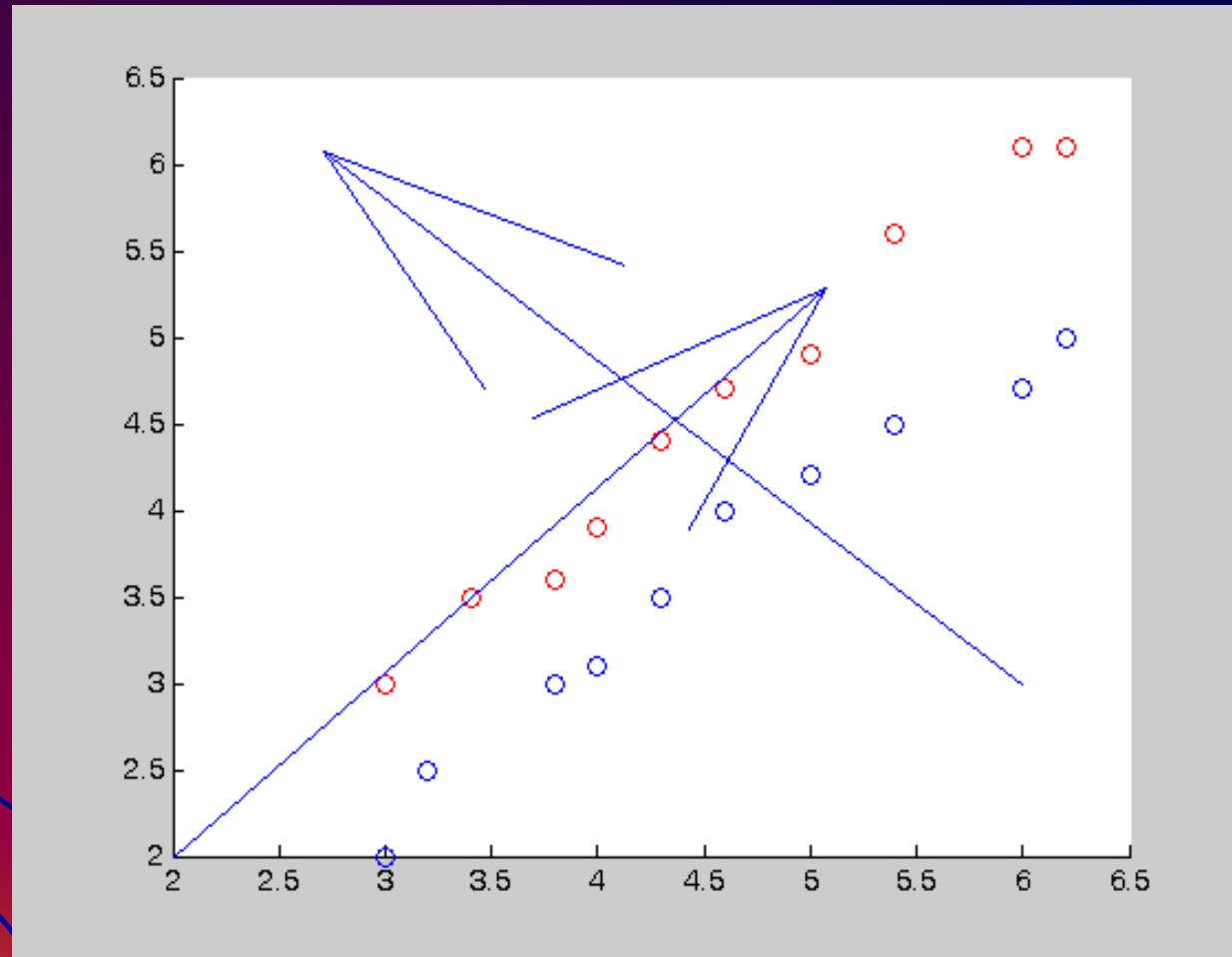
# Demonstration of KL Transform

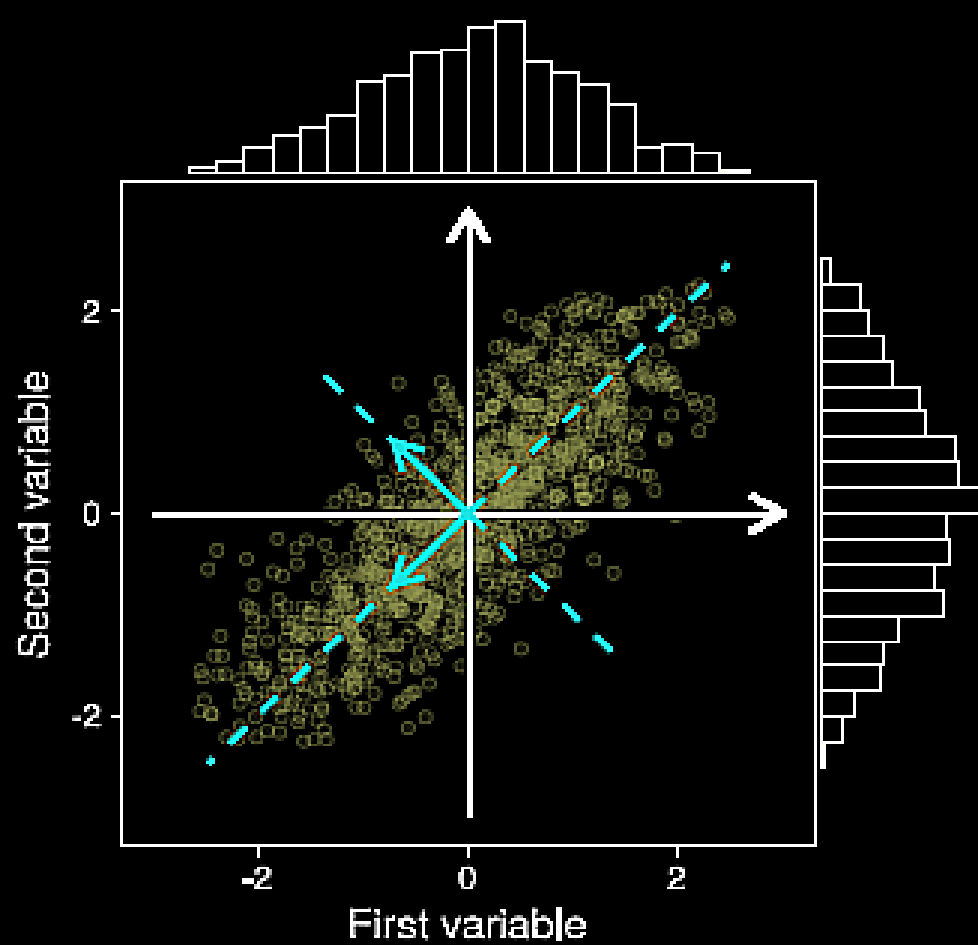


First  
eigen  
vector

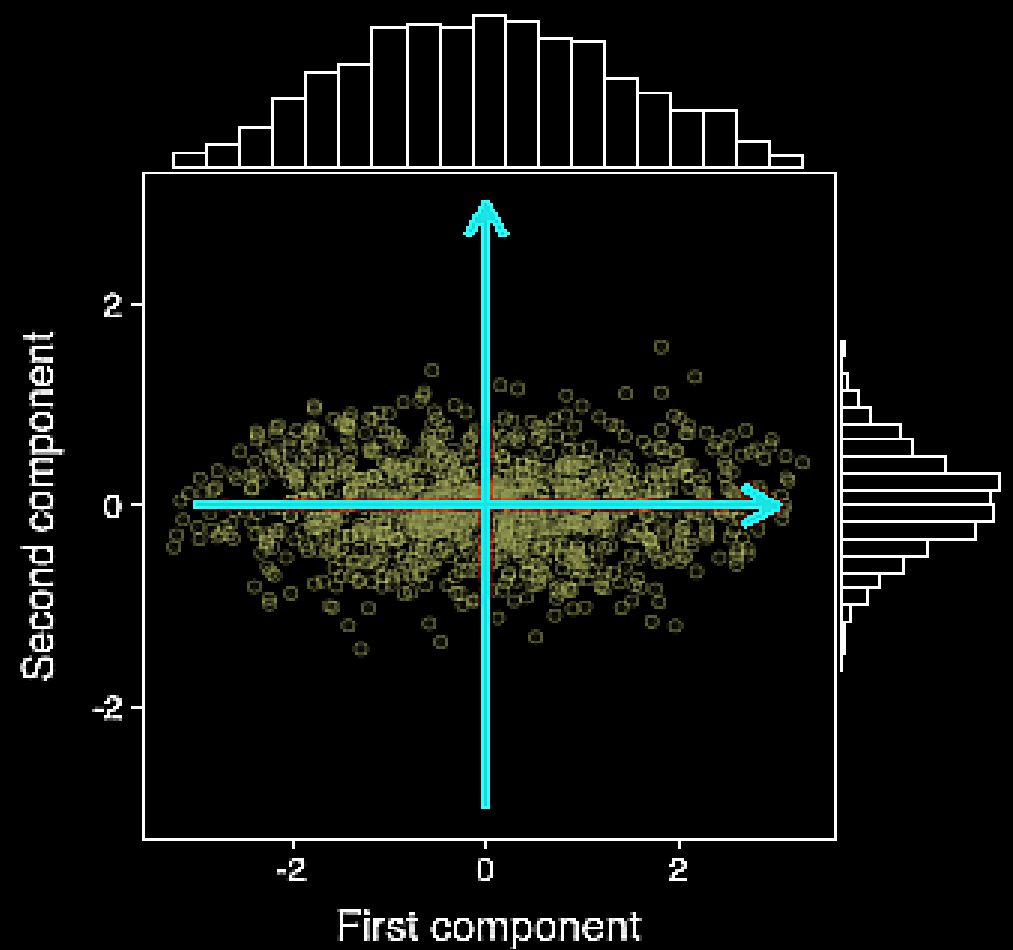
Second  
eigen  
vector

# Another One

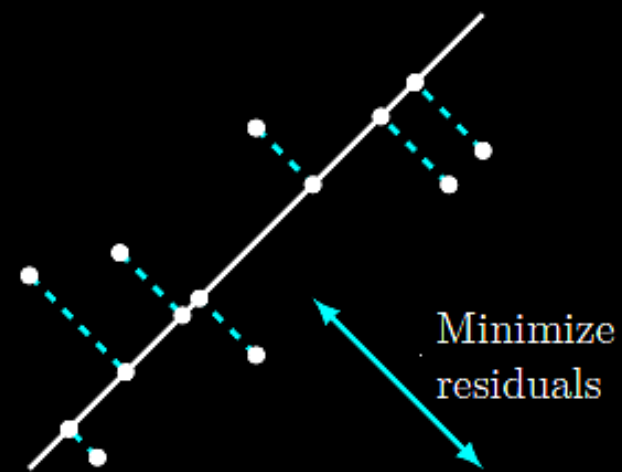
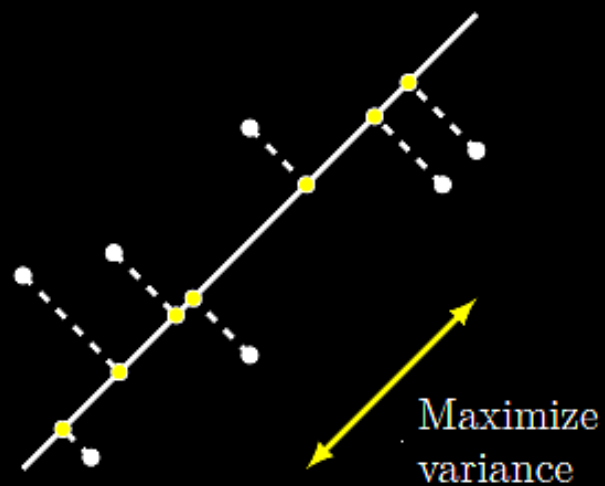




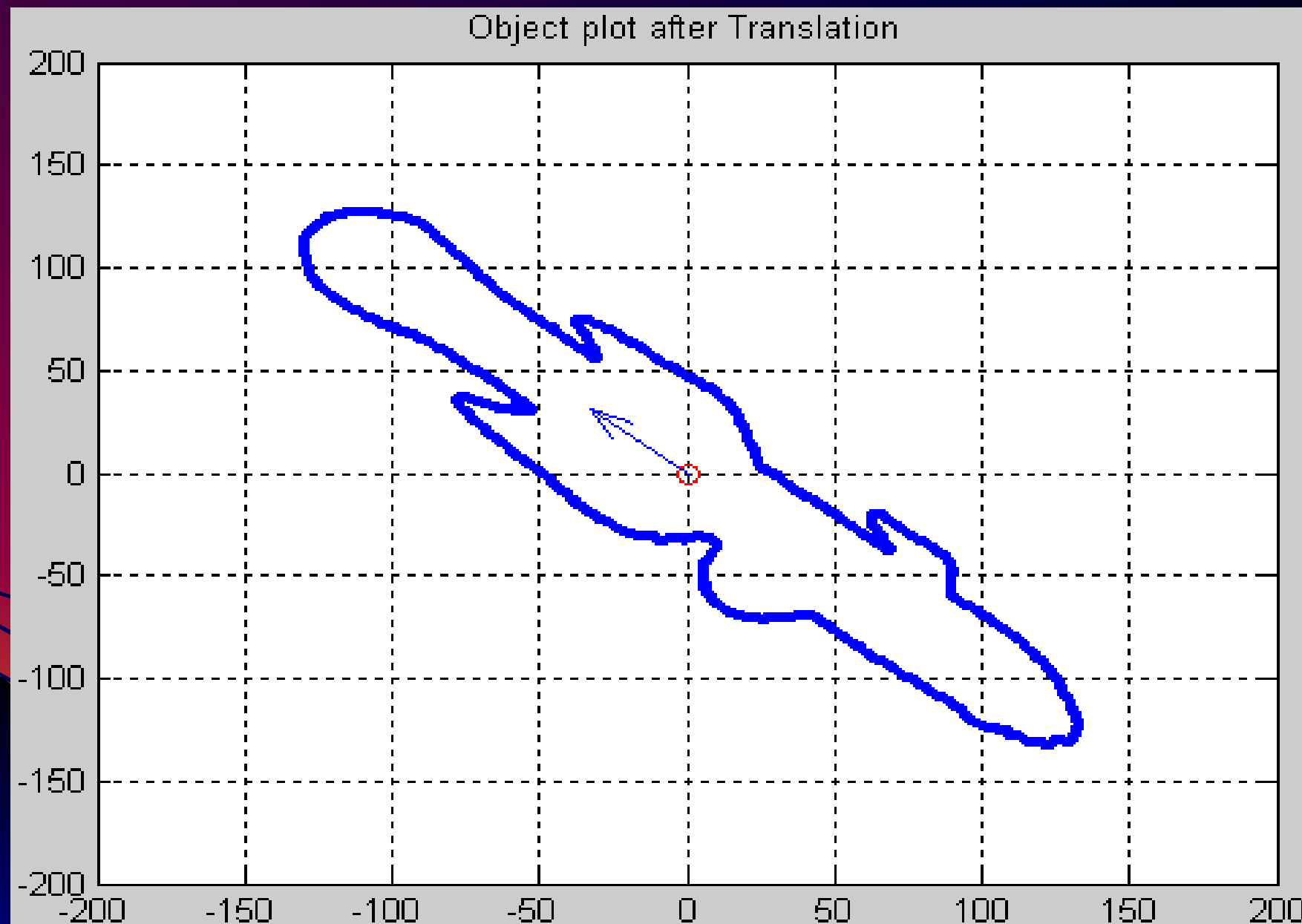
(a) Original (standardized) data.



(b) Rotated data.



# Another Example



Source: SQUID Homepage



**Principal components analysis (PCA)** is a technique used to reduce multi-dimensional data sets to lower dimensions for analysis.

The applications include exploratory data analysis and generating predictive models. PCA involves the computation of the eigenvalue decomposition or Singular value decomposition of a data set, usually after mean centering the data for each attribute.

PCA is mathematically defined as an orthogonal linear transformation, that transforms the data to a new coordinate system such that the **greatest variance** by any projection of the data comes to lie on the **first coordinate** (called the first principal component), the second greatest variance on the second coordinate, and so on.

PCA can be used for dimensionality reduction in a data set by retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. Such low-order components often contain the "most important" aspects of the data. But this is not necessarily the case, depending on the application.

For a data matrix,  $X^T$ , with zero empirical mean (the empirical mean of the distribution has been subtracted from the data set), where each *column* is made up of results for a different subject, and each *row* the results from a different probe. This will mean that the PCA for our data matrix  $X$  will be given by:

$$Y = W^T X = \Sigma V^T,$$

where  $W\Sigma V^T$  is the singular value decomposition (SVD) of  $X$ .

**Goal of PCA:**

Find some orthonormal matrix  $W^T$ , where  $Y = W^T X$ ;  
such that

**$\text{COV}(Y) \equiv (1/(n-1))YY^T$  is diagonalized.**

The rows of  $W$  are the principal components of  $X$ , which are also the eigenvectors of  $\text{COV}(X)$ .

Unlike other linear transforms (DCT, DFT, DWT etc.), PCA does not have a fixed set of basis vectors. Its basis vectors depend on the data set.

## SVD – the theorem (Src; WIKI ++)

Suppose  $M$  is an  **$m$ -by- $n$**  matrix whose entries come from the field  $K$ , which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form

$$M = U\Sigma V^*$$

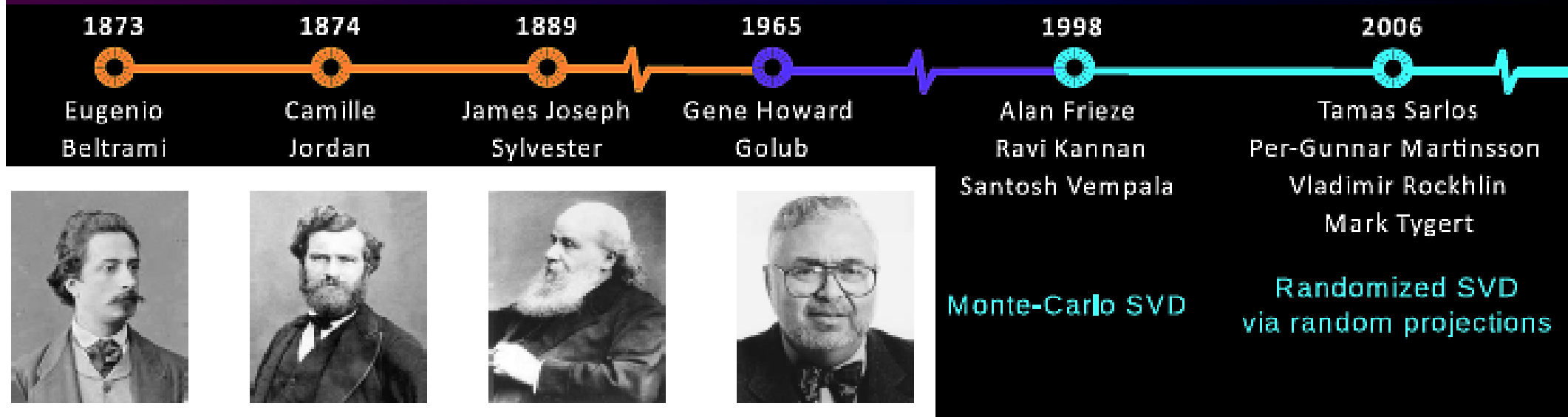
where,  **$U$  is an  $m$ -by- $m$**  unitary matrix, the matrix  **$\Sigma$  is  $m$ -by- $n$**  with nonnegative numbers on the diagonal and zeros off the diagonal, and  **$V^*$**  denotes the conjugate transpose of  **$V$ , an  $n$ -by- $n$**  unitary matrix over  $K$ . Such a factorization is called a (Full) singular-value decomposition of  $M$ .

The matrix  $V$  thus contains a set of orthonormal "input" or "analysing" basis vector directions for  $M$ .

The matrix  $U$  contains a set of orthonormal "output" basis vector directions for  $M$ . The matrix  $\Sigma$  contains the singular values, which can be thought of as scalar "gain controls" by which each corresponding input is multiplied to give a corresponding output.

A common convention is to order the values  $\Sigma_{i,i}$  in non-increasing fashion. In this case, the diagonal matrix  $\Sigma$  is uniquely determined by  $M$  (though the matrices  $U$  and  $V$  are not).

For  $p = \min(m,n)$  -  **$U$  is  $m$ -by- $p$ ,  $\Sigma$  is  $p$ -by- $p$ , and  $V$  is  $n$ -by- $p$ .**



A timeline of major singular value decomposition developments.

Erichson, N. B., Voronin, S., **Brunton, S. L.**, & Kutz, J. N. (2019). Randomized Matrix

The columns of  $U$  ( $m \times m$ ) are **eigenvectors of  $AA^T$** , and the columns of  $V$  ( $n \times n$ ) are **eigenvectors of  $A^T A$**

Given a real matrix  $A \in \mathbb{R}^{m \times n}$  with  $m \geq n$ , the singular value decomposition takes the form

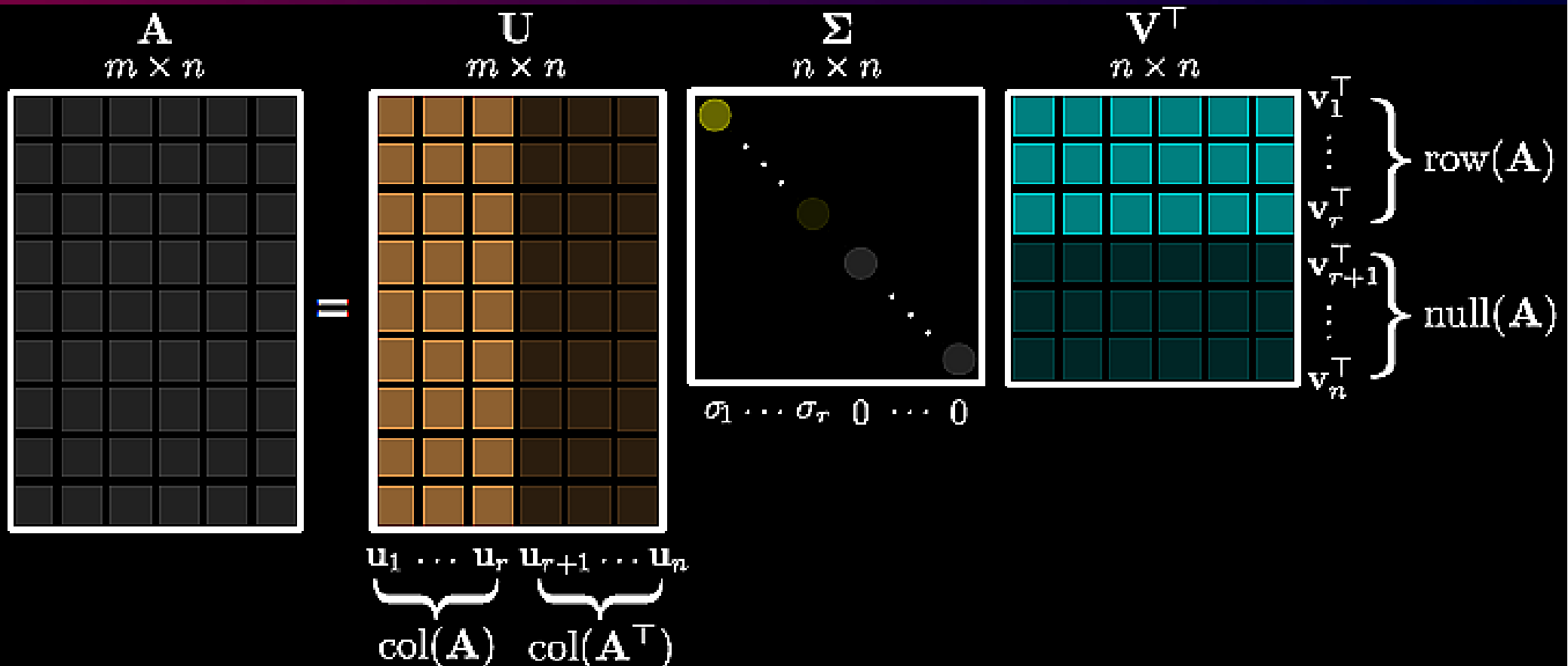
$$A = U \Sigma V^T.$$

The matrices  $U = [u_1, \dots, u_m] \in \mathbb{R}^{m \times m}$  and  $V = [v_1, \dots, v_n] \in \mathbb{R}^{n \times n}$  are orthonormal so that  $U^T U = I$  and  $V^T V = I$ . The left singular vectors in  $U$  provide a basis for the range (column space), and the right singular vectors in  $V$  provide a basis for the domain (row space) of the matrix  $A$ . The rectangular diagonal matrix  $\Sigma \in \mathbb{R}^{m \times n}$  contains the corresponding non-negative singular values  $\sigma_1 \geq \dots \geq \sigma_n \geq 0$ , describing the spectrum of the data.

The so called “economy” or “thin” SVD computes only the left singular vectors and singular values corresponding to the number (i.e.,  $n$ ) of right singular vectors

$$A = U\Sigma V = [u_1, \dots, u_n] \text{diag}(\sigma_1, \dots, \sigma_n) [v_1, \dots, v_n]^T.$$

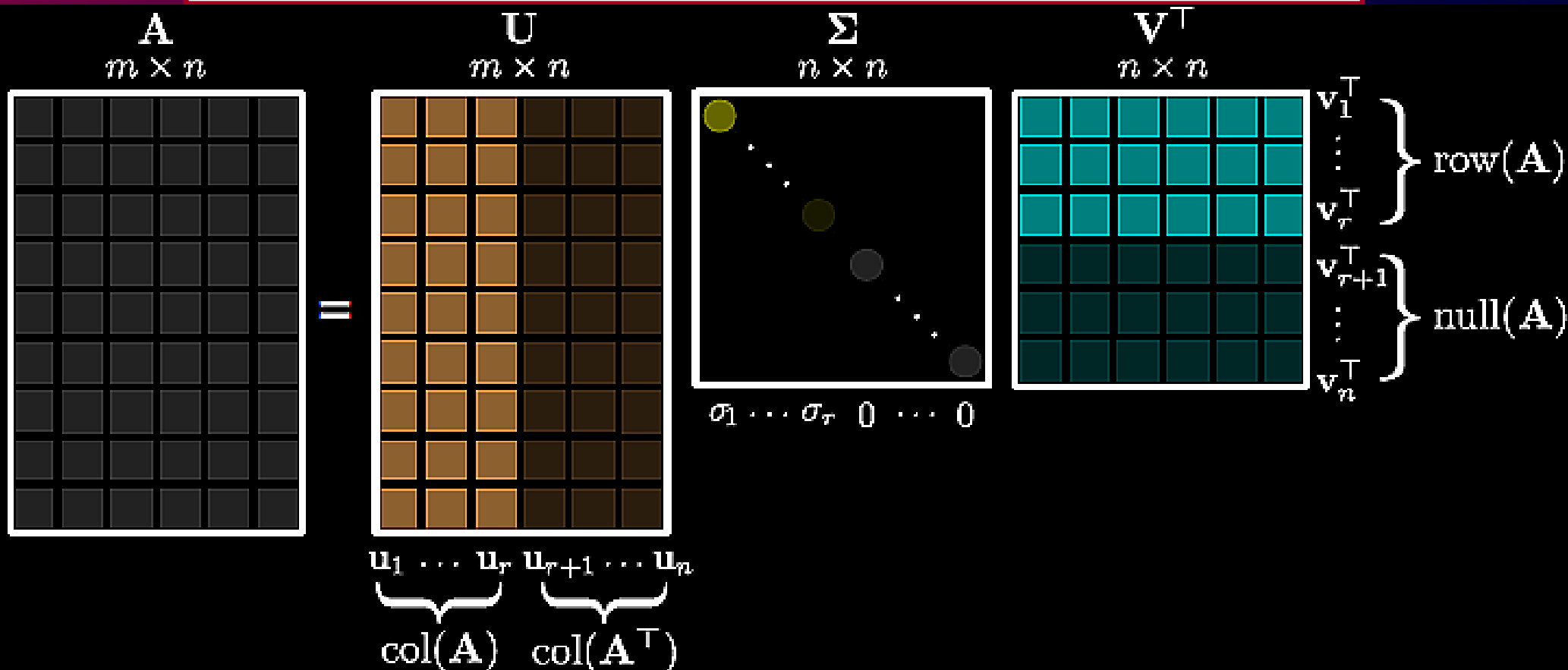
**If the number of right singular vectors is small (i.e.  $n \ll m$ ), this is a more compact factorization than the full SVD.**



Schematic of the “economy” SVD for a rank- $r$  matrix, where  $m \geq n$ .

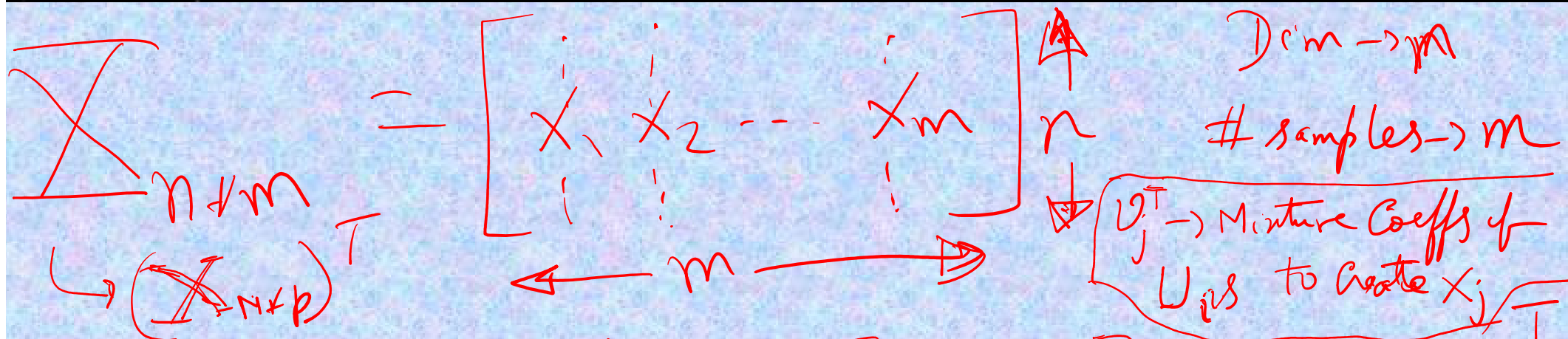
Low-rank matrices feature a rank ( $r$ ) that is smaller than the dimension of the matrix. The first  $r$  columns of  $U$  are the column space of  $A$  and the last  $m - r$  columns of  $U$  are the left nullspace of  $A$ . The first  $r$  columns of  $V$  are the row space of  $A$  and the last  $n - r$  columns of  $V$  are the nullspace of  $A$ .

first  $r$  columns of  $U$ : column space of  $A$   
 last  $m - r$  columns of  $U$ : left nullspace of  $A$   
 first  $r$  columns of  $V$ : row space of  $A$   
 last  $n - r$  columns of  $V$ : nullspace of  $A$



Schematic of the “economy” SVD for a rank- $r$  matrix, where  $m \geq n$ .

$$U \Sigma V^T = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ \hline & & & & 0 \end{bmatrix} \begin{bmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_n \\ | & | & & | \end{bmatrix}^T$$



$$= \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{bmatrix}_{n \times n} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ \hline & & & & 0 \end{bmatrix}_{n \times m} \begin{bmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_m \\ | & | & & | \end{bmatrix}_{m \times m}^T$$

$v_i \in \mathbb{R}^m; u_i, x_i \in \mathbb{R}^n$

$$U^T U = U U^T = I_{n \times n} \quad \parallel \quad \text{L.S.V.}$$

$$V^T V = V V^T = I_{m \times m} \quad \parallel \quad \text{R.S.V.}$$

**In practical applications matrices are often contaminated by errors, and the effective rank of a matrix can be smaller than its exact rank  $r$ .**

**In this case, the matrix can be well approximated by including only those singular vectors which correspond to singular values of a significant magnitude. Hence, it is often desirable to compute a reduced version of the SVD, as:**

$$\mathbf{A}_k := \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k = [\mathbf{u}_1, \dots, \mathbf{u}_k] \text{diag}(\sigma_1, \dots, \sigma_k) [\mathbf{v}_1, \dots, \mathbf{v}_k]^\top,$$

where  $k$  denotes the desired target rank of the approximation. In other words, this reduced form of the SVD allows one to express  $\mathbf{A}$  approximately by the sum of  $k$  rank-one matrices

$$\mathbf{A}_k \approx \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^\top.$$

Choosing an optimal target rank  $k$  is highly dependent on the task.

**For massive datasets, however, the truncated/reduced SVD is costly to compute. The cost to compute the full SVD of an  $m \times n$  matrix is of the order  $O(mn^2)$ , from which the first  $k$  components can then be extracted to form  $\mathbf{A}_k$ .**

**$k$  should be chosen close to the effective rank – data representation Applcn.; while, chosen much smaller ( $\ll r$ ) for dimension reduction (PCA).**



$$\begin{array}{|c|} \hline A \\ \hline n \times d \\ \hline \end{array} = \begin{array}{|c|} \hline U \\ \hline n \times r \\ \hline \end{array} \begin{array}{|c|} \hline D \\ \hline r \times r \\ \hline \end{array} \begin{array}{|c|} \hline V^T \\ \hline r \times d \\ \hline \end{array}$$

B.Tech, CSE\_ IIT Madras (1997);  
Ph.D., MIT (2001);

Miller Research Fellow,  
UC Berkeley (2001-02);  
CMU; Berkeley

The SVD decomposition of an  $n \times d$  matrix.

*Theorem 1.5 Let  $A$  be an  $n \times d$  matrix with right singular vectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$ , left singular vectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ , and corresponding singular values  $\sigma_1, \sigma_2, \dots, \sigma_r$ . Then*

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

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \text{ and } \mathbf{A}^T = \mathbf{V} \mathbf{\Sigma} \mathbf{U}^T$$

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}_{r \times r} \quad \diamond \quad \mathbf{U} \mathbf{U}^T$$

$$\mathbf{V} \mathbf{V}^T \diamond \mathbf{I}_{r \times r} = \mathbf{V}^T \mathbf{V}$$

$$\mathbf{A}^T \mathbf{A} = \mathbf{V} \mathbf{\Sigma} \mathbf{U}^T \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T$$

$$\mathbf{A}^T \mathbf{A} \mathbf{V} = \mathbf{V} \mathbf{\Sigma}^2; \quad \mathbf{V}^T \mathbf{A}^T \mathbf{A} \mathbf{V} = \mathbf{\Sigma}^2;$$

$$\mathbf{A} \mathbf{A}^T = ??$$

The diagonal entries  $\sigma_i = \sum_{ii}$  of  $\Sigma$  are uniquely determined by  $\mathbf{M}$  and are known as the **singular values** of  $\mathbf{M}$ . The number of non-zero singular values is equal to the **rank** of  $\mathbf{M}$ . The columns of  $\mathbf{U}$  and the columns of  $\mathbf{V}$  are called **left-singular vectors** and **right-singular vectors** of  $\mathbf{M}$ , respectively. They form two sets of **orthonormal bases**  $\mathbf{u}_1, \dots, \mathbf{u}_m$  and  $\mathbf{v}_1, \dots, \mathbf{v}_n$ , and if they are sorted so that the singular values  $\sigma_i$  with value zero are all in the highest-numbered columns (or rows), the singular value decomposition can be written as 
$$\mathbf{M} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^*$$
, where  $r \leq \min\{m, n\}$  is the rank of  $\mathbf{M}$ .

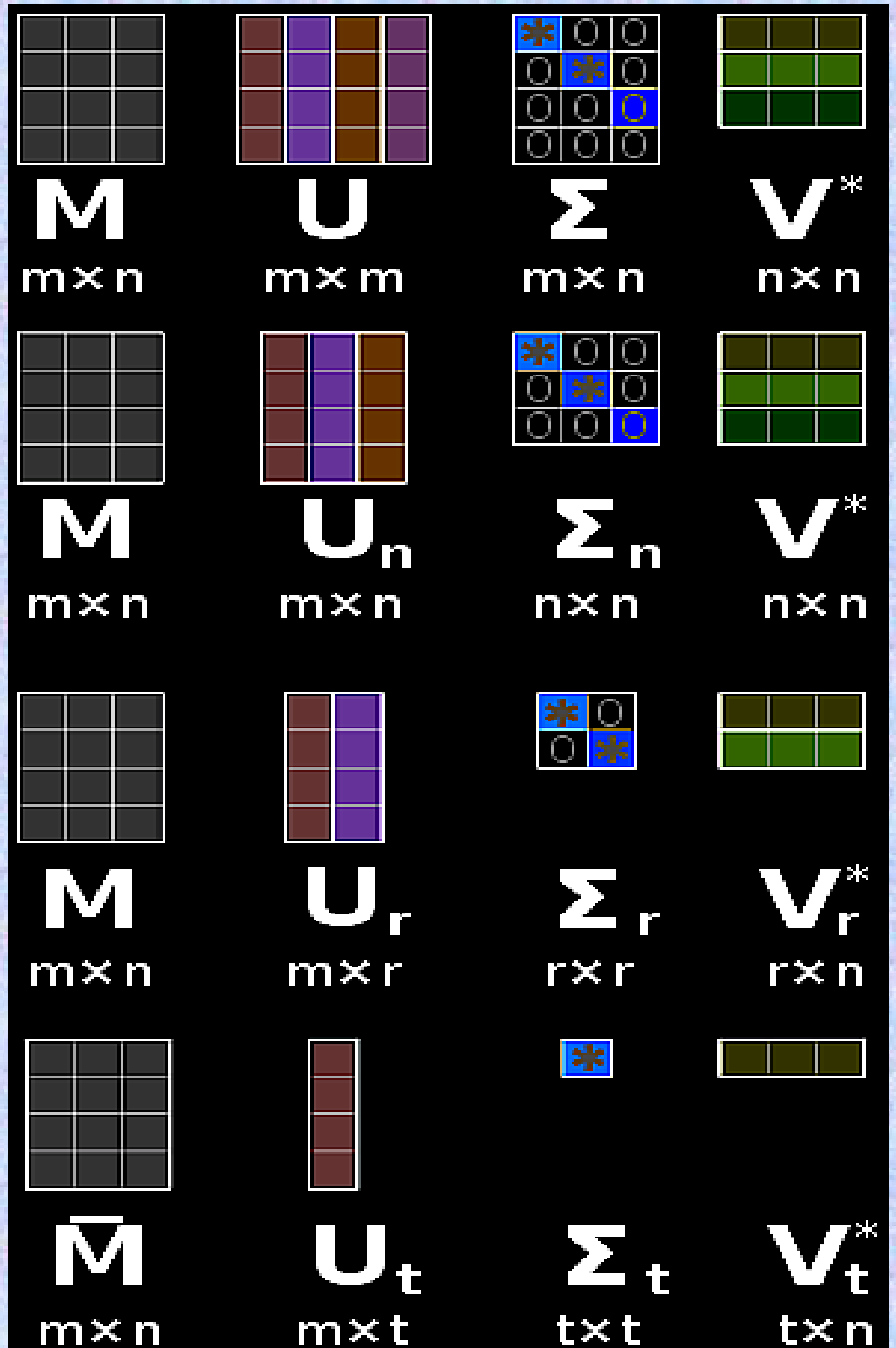
The term sometimes refers to the **compact SVD**, a similar decomposition  $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^*$  in which  $\Sigma$  is square diagonal of size  $r \times r$ , where  $r \leq \min\{m, n\}$  is the rank of  $\mathbf{M}$ , and has only the non-zero singular values. In this variant,  $\mathbf{U}$  is an  $m \times r$  **semi-unitary matrix** and  $\mathbf{V}$  is an  $n \times r$  **semi-unitary matrix**, such that 
$$\mathbf{U}^* \mathbf{U} = \mathbf{V}^* \mathbf{V} = \mathbf{I}_r.$$

1: Full SVD,

2: Thin SVD (remove columns of U not corresponding to rows of V\*),

3: Compact SVD (remove vanishing singular values and corresponding columns/rows in U and V\*),

4: Truncated SVD (keep only largest t singular values and corresponding columns/rows in U and V\*)



The Karhunen-Loève transform is therefore equivalent to finding the singular value decomposition of the data matrix  $X$ , and then obtaining the reduced-space data matrix  $Y$  by projecting  $X$  down into the reduced space defined by only the first  $L$  singular vectors,  $W_L$ :

$$X = W\Sigma V^T; \quad Y = W_L^T X = \Sigma_L V_L^T$$

The matrix  $W$  of singular vectors of  $X$  is equivalently the matrix  $W$  of eigenvectors of the matrix of observed covariances  $C = X X^T$  (find out?) =:

$$COV(X) = X X^T = W\Sigma\Sigma^T W^T = W D W^T$$

The eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the data set. PCA is equivalent to empirical orthogonal functions (EOF).

PCA is a popular technique in pattern recognition. But it is not optimized for class separability. An alternative is the linear discriminant analysis, which does take this into account. PCA optimally minimizes reconstruction error under the  $L_2$  norm.

## PCA by COVARIANCE Method

We need to find a  $d \times d$  orthonormal transformation matrix  $W^T$ , such that:

$$Y = W^T X$$

with the constraint that:

$\text{Cov}(Y)$  is a diagonal matrix, and  $W^{-1} = W^T$ .

$$\begin{aligned} \text{COV}(Y) &= E[YY^T] = E[(W^T X)(W^T X)^T] \\ &= E[(W^T X)(X^T W)] = W^T E[XX^T] W \\ &= W^T \text{COV}(X) W = W^T (W D W^T) W = D \end{aligned}$$

$$W \text{COV}(Y) = W W^T \text{COV}(X) W = \text{COV}(X) W$$

Can you derive from the above, that:

$$\begin{aligned} [\lambda_1 W_1, \lambda_2 W_2, \dots, \lambda_d W_d] &= \\ [\text{COV}(X) W_1, \text{COV}(X) W_2, \dots, \text{COV}(X) W_d] \end{aligned}$$

The covariance matrix  $\mathbf{C}_S$  is real and symmetric. Hence it can be diagonalized, which simplifies the covariance structure. We could do this in the normal way by finding its eigenvalues and eigenvectors. However, there is also a close relationship between  $\mathbf{C}_S$  and the SVD of the data matrix which is advantageous to exploit.

Let the SVD of the data matrix be:

$$\begin{aligned}\mathbf{COV}(X) &= \mathbf{X}\mathbf{X}^T = \mathbf{W}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{W}^T = \mathbf{W}\mathbf{D}\mathbf{W}^T \\ \mathbf{COV}(Y) &= \mathbf{D}\end{aligned}$$

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

Basic properties of the SVD give the diagonalization of  $\mathbf{C}_S$ :

$$\mathbf{C}_S = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

where  $\mathbf{\Lambda} = \text{diag}(\sigma_k^2)/(n-1)$ . The left singular vectors  $\mathbf{u}_k$  of the data matrix are the eigenvectors of the covariance matrix. If we rotate the data matrix into this basis by setting  $\hat{\mathbf{S}} = \mathbf{U}^T\mathbf{S}$ , then the rotated data has covariance matrix

$$\mathbf{C}_{\hat{\mathbf{S}}} = \frac{1}{n-1}\hat{\mathbf{S}}\hat{\mathbf{S}}^T = \mathbf{U}^T\mathbf{C}_S\mathbf{U} = \mathbf{\Lambda}$$

$$\mathbf{W}_L^T \mathbf{X} = \sum_L \mathbf{V}_L^T$$

In this rotated coordinate system, the variance decouples (is mutually uncorrelated) between coordinate directions, with direction  $k$  contributing a variance  $\sigma_k^2/(n-1)$ . The time-dependent amplitude of the  $k$ 'th rotated data component is  $\mathbf{u}_k^T\mathbf{S} = \sigma_k\mathbf{v}_k$ .

This decomposition of the covariance matrix is called principal component analysis (PCA). The vectors  $\mathbf{u}_k$  are called the loading vectors or patterns. The vectors  $\mathbf{v}_k$  are called the principal components or PCs. Various application-dependent normalizations are applied to the PCs

Maximise  $\mathbf{u}^T \mathbf{X} \mathbf{X}^T \mathbf{u}$  s.t  $\mathbf{u}^T \mathbf{u} = 1$

Construct Lagrangian  $\mathbf{u}^T \mathbf{X} \mathbf{X}^T \mathbf{u} - \lambda \mathbf{u}^T \mathbf{u}$

Vector of partial derivatives set to zero

$$\mathbf{X} \mathbf{X}^T \mathbf{u} - \lambda \mathbf{u} = (\mathbf{X} \mathbf{X}^T - \lambda \mathbf{I}) \mathbf{u} = \mathbf{0}$$

As  $\mathbf{u} \neq \mathbf{0}$  then  $\mathbf{u}$  must be an eigenvector of  $\mathbf{X} \mathbf{X}^T$  with eigenvalue  $\lambda$

let  $\mathbf{u}$  be an arbitrary vector of length 1, so that  $\mathbf{u}^T \mathbf{S}$  (a column vector of length  $n$ ) is its projection on the data matrix. Let  $\hat{\mathbf{u}} = \mathbf{U}^T \mathbf{u}$  be this unit vector expressed into the rotated basis (in which it will also have length 1). Then

$$\begin{aligned} \text{var}[\mathbf{u}^T \mathbf{S}] &= \frac{1}{n-1} \mathbf{u}^T \mathbf{S} \mathbf{S}^T \mathbf{u} \\ &= \mathbf{u}^T \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T \mathbf{u} \\ &= \hat{\mathbf{u}}^T \mathbf{\Lambda} \hat{\mathbf{u}} \\ &= \frac{1}{n-1} \sum_k \sigma_k^2 \hat{u}_k^2 \\ &\leq \frac{\sigma_1^2}{n-1} \sum_k \hat{u}_k^2 = \frac{\sigma_1^2}{n-1}. \end{aligned}$$

# Steps for principal component analysis

- Principal component analysis using the covariance function should only be considered if all of the variables have the same units of measurement.

If the variables have different units of measurement, (i.e., pounds, feet, gallons, etc), or if we wish each variable to receive equal weight in the analysis, then the variables should be standardized before conducting a principal components analysis. To standardize a variable, subtract the mean and divide by the standard deviation:

$$Z_{ij} = \frac{X_{ij} - \bar{x}_j}{s_j}$$

where

- $X_{ij}$  = Data for variable  $j$  in sample unit  $i$
- $\bar{x}_j$  = Sample mean for variable  $j$
- $s_j$  = Sample standard deviation for variable  $j$

**Note!** The variance-covariance matrix of the standardized data is equal to the correlation matrix for the unstandardized data. Therefore, principal component analysis using the standardized data is equivalent to principal component analysis using the correlation matrix.



## A Summary of the PCA Approach

- Standardize the data.
- Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Value Decomposition.
- Eigenvalues from SVD are sorted in descending order; so choose the  $\mathbf{k}$  eigenvectors that correspond to the  $\mathbf{k}$  largest eigenvalues where  $\mathbf{k}$  is the number of dimensions of the new feature subspace ( $\mathbf{k} \leq d$ ).
- Construct the projection matrix  $W$  from the selected  $\mathbf{k}$  eigenvectors.

Using SVD on the data matrix has two advantages over just calling the Matlab function `eig` on the covariance matrix  $C_S$ , which would give the  $\sigma_k^2)/(n - 1)$ s as the eigenvalues, and the patterns  $u_k$  as the eigenvectors. First, if  $m > n$  (more variables than samples),  $C_S$  is  $m \times m$ , which can become very large ( $m$  is over 2000 in our Pacific SST example, while  $n$  is only 396). Only  $n$  or less of these eigenvalues will be nonzero, but this can choke Matlab. Note there is a short version `svds` that, like `eigs` will just return a small number of leading singular modes, which is all we usually care about in PCA. That can minimize computation and memory requirements if the dataset is large.

Second, the right singular vectors automatically give the principal component time series for the patterns. To get these by eigendecomposition of the covariance matrix requires an extra step of projecting the data at each time onto the eigenvectors  $u_k$ .

## Example of PCA

Samples:  $x_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}; x_2 = \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix}; x_3 = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix};$   $X = \begin{bmatrix} -1 & -2 & 4 \\ 1 & 3 & 0 \\ 2 & 1 & 3 \end{bmatrix}$

3-D problem, with  $N = 3$ .

Each column is an observation (sample) and each row a variable (dimension),

Mean of the samples:  $\mu_x = \begin{bmatrix} 1/3 \\ 4/3 \\ 2 \end{bmatrix}; \tilde{x}_1 = \begin{bmatrix} -4/3 \\ -1/3 \\ 0 \end{bmatrix}; \tilde{x}_2 = \begin{bmatrix} -7/3 \\ 5/3 \\ -1 \end{bmatrix}; \tilde{x}_3 = \begin{bmatrix} 11/3 \\ -4/3 \\ 1 \end{bmatrix};$

**Method – 1** (easiest)

$$\tilde{X} = \begin{bmatrix} -4/3 & -7/3 & 11/3 \\ -1/3 & 5/3 & -4/3 \\ 0 & -1 & 1 \end{bmatrix}; \text{COVAR} = (\tilde{X} \tilde{X}^T) / 2 = (1/2) \begin{bmatrix} 62/3 & -25/3 & 6 \\ -25/3 & 14/3 & -3 \\ 6 & -3 & 2 \end{bmatrix}$$

## Method – 2 (PCA defn.)

$$S_T = \left(\frac{1}{N-1}\right) \sum_{k=1}^N (x_k - \mu)(x_k - \mu)^T$$

C1 =

$$\begin{bmatrix} 1.7778 & 0.4444 & 0 \\ 0.4444 & 0.1111 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

C2 =

$$\begin{bmatrix} 5.4444 & -3.8889 & 2.3333 \\ -3.8889 & 2.7778 & -1.6667 \\ 2.3333 & -1.6667 & 1.0000 \end{bmatrix}$$

SigmaC =

$$\begin{bmatrix} 20.6667 & -8.3333 & 6.0000 \\ -8.3333 & 4.6667 & -3.0000 \\ 6.0000 & -3.0000 & 2.0000 \end{bmatrix}$$

$$\tilde{x}_1 = \begin{bmatrix} -4/3 \\ -1/3 \\ 0 \end{bmatrix}; \tilde{x}_2 = \begin{bmatrix} -7/3 \\ 5/3 \\ -1 \end{bmatrix}; \tilde{x}_3 = \begin{bmatrix} 11/3 \\ -4/3 \\ 1 \end{bmatrix};$$

C3 =

$$\begin{bmatrix} 13.4444 & -4.8889 & 3.6667 \\ -4.8889 & 1.7778 & -1.3333 \\ 3.6667 & -1.3333 & 1.0000 \end{bmatrix}$$

COVAR =

SigmaC/2 =

$$\begin{bmatrix} 10.3333 & -4.1667 & 3.0000 \\ -4.1667 & 2.3333 & -1.5000 \\ 3.0000 & -1.5000 & 1.0000 \end{bmatrix}$$

Next do SVD, to get vectors.

For a face image with N samples and dimension d (=w\*h, very large), we have:

The array X or  $X_{avg}$  of size d\*N (N vertical samples stacked horizontally)

Thus  $XX^T$  will be of d\*d, which will be very large. To perform eigen-analysis on such large dimension is time consuming and may be erroneous.

Thus often  $X^T X$  of dimension N\*N is considered for eigen-analysis. Will it result in the same, after SVD? Lets check:

$$S = \tilde{X} \tilde{X}^T = (1/2) \begin{bmatrix} 62/3 & -25/3 & 6 \\ -25/3 & 14/3 & -3 \\ 6 & -3 & 2 \end{bmatrix} = \begin{bmatrix} 10.3333 & -4.1667 & 3.0000 \\ -4.1667 & 2.3333 & -1.5000 \\ 3.0000 & -1.5000 & 1.0000 \end{bmatrix}$$

$$S^m = \tilde{X}^T \tilde{X} = \begin{bmatrix} 0.9444 & 1.2778 & -2.2222 \\ 1.2778 & 4.6111 & -5.8889 \\ -2.2222 & -5.8889 & 8.1111 \end{bmatrix}$$

*Lets do SVD of both:*

$$S = X \tilde{X}^T =$$

10.3333	-4.1667	3.0000
-4.1667	2.3333	-1.5000
3.0000	-1.5000	1.0000

U =

-0.8846	-0.4554	-0.1010
0.3818	-0.8313	0.4041
-0.2680	0.3189	0.9091

S =

13.0404	0	0
0	0.6263	0
0	0	0.0000

V =

-0.8846	-0.4554	0.1010
0.3818	-0.8313	-0.4041
-0.2680	0.3189	-0.9091

$$S^m = \tilde{X}^T \tilde{X} =$$

0.9444	1.2778	-2.2222
1.2778	4.6111	-5.8889
-2.2222	-5.8889	8.1111

U =

-0.2060	0.7901	0.5774
-0.5812	-0.5735	0.5774
0.7872	-0.2166	0.5774

S =

13.0404	0	0
0	0.6263	0
0	0	0.0000

V =

-0.2060	0.7901	0.5774
-0.5812	-0.5735	0.5774
0.7872	-0.2166	0.5774

Samples:

Example, where  $d \neq N$ :

$$x_1 = \begin{bmatrix} -3 \\ -3 \end{bmatrix}; x_2 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}; x_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}; x_4 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}; x_5 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}; x_6 = \begin{bmatrix} 6 \\ 7 \end{bmatrix};$$

2-D problem ( $d=2$ ), with  $N = 6$ .

Each column is an observation (sample) and each row a variable (dimension),

$$X = \begin{bmatrix} -3 & -2 & -1 & 4 & 5 & 6 \\ -3 & -2 & -1 & 4 & 5 & 7 \end{bmatrix}$$

Mean of the samples:

$$\mu_x = \begin{bmatrix} 3 / 2 \\ 5 / 3 \end{bmatrix};$$

$XM =$

$$\begin{bmatrix} -4.5000 & -3.5000 & -2.5000 & 2.5000 & 3.5000 & 4.5000 \\ -4.6667 & -3.6667 & -2.6667 & 2.3333 & 3.3333 & 5.3333 \end{bmatrix}$$

$XM^T * XM =$

$COVAR(X) = XM * XM^T$

$$= \begin{bmatrix} 77.5000 & 82.0000 \\ 82.0000 & 87.3333 \end{bmatrix}$$

$$\begin{bmatrix} 42.0278 & 32.8611 & 23.6944 & -22.1389 & -31.3056 & -45.1389 \\ 32.8611 & 25.6944 & 18.5278 & -17.3056 & -24.4722 & -35.3056 \\ 23.6944 & 18.5278 & 13.3611 & -12.4722 & -17.6389 & -25.4722 \\ -22.1389 & -17.3056 & -12.4722 & 11.6944 & 16.5278 & 23.6944 \\ -31.3056 & -24.4722 & -17.6389 & 16.5278 & 23.3611 & 33.5278 \\ -45.1389 & -35.3056 & -25.4722 & 23.6944 & 33.5278 & 48.6944 \end{bmatrix}$$

$$\text{COVAR}(X) = XM * XM^T$$

$$= \begin{bmatrix} 77.5000 & 82.0000 \\ 82.0000 & 87.3333 \end{bmatrix}$$

$$U =$$

$$\begin{bmatrix} -0.6856 & -0.7280 \\ -0.7280 & 0.6856 \end{bmatrix}$$

$$S =$$

$$\begin{bmatrix} 164.5639 & 0 \\ 0 & 0.2694 \end{bmatrix}$$

$$V =$$

$$\begin{bmatrix} -0.6856 & -0.7280 \\ -0.7280 & 0.6856 \end{bmatrix}$$

$$XM^T * XM =$$

$$\begin{bmatrix} 42.0278 & 32.8611 & 23.6944 & -22.1389 & -31.3056 & -45.1389 \\ 32.8611 & 25.6944 & 18.5278 & -17.3056 & -24.4722 & -35.3056 \\ 23.6944 & 18.5278 & 13.3611 & -12.4722 & -17.6389 & -25.4722 \\ -22.1389 & -17.3056 & -12.4722 & 11.6944 & 16.5278 & 23.6944 \\ -31.3056 & -24.4722 & -17.6389 & 16.5278 & 23.3611 & 33.5278 \\ -45.1389 & -35.3056 & -25.4722 & 23.6944 & 33.5278 & 48.6944 \end{bmatrix}$$

$$U =$$

$$\begin{bmatrix} -0.5053 & -0.1469 & -0.7547 & 0.3882 & 0.0214 & 0.0486 \\ -0.3951 & -0.0654 & 0.3632 & 0.0984 & -0.4091 & 0.7284 \\ -0.2849 & 0.0162 & -0.0433 & -0.3456 & -0.7396 & -0.5002 \\ 0.2660 & 0.4241 & -0.5083 & -0.5306 & -0.1150 & 0.4429 \\ 0.3762 & 0.5057 & -0.0258 & 0.6601 & -0.4043 & -0.0539 \\ 0.5432 & -0.7337 & -0.1938 & 0.0541 & -0.3293 & 0.1332 \end{bmatrix}$$

$$S =$$

$$\begin{bmatrix} 164.5639 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2694 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0 \end{bmatrix}$$

$V = U ??$

**X:**

**[5 5 0 0 1]  
[4 5 1 1 0]  
[5 4 1 1 0]  
[0 0 4 4 4]  
[0 0 5 5 5]  
[1 1 4 4 4]**

**Covariance Matrix for X:**

**[ 4.917 4.75 -4.083 -4.083 -4.333]  
[ 4.75 4.917 -4.083 -4.083 -4.333]  
[-4.083 -4.083 3.583 3.583 3.667]  
[-4.083 -4.083 3.583 3.583 3.667]  
[-4.333 -4.333 3.667 3.667 4.222]**

**SVD applied on Covariance Matrix:**

**U:**

**[-0.482 0.076 -0.707 -0.511 0. ]  
[-0.482 0.076 0.707 -0.511 0. ]  
[ 0.413 -0.365 -0. -0.443 0.707]  
[ 0.413 -0.365 -0. -0.443 -0.707]  
[ 0.44 0.85 -0. -0.289 0. ]**

**D:**

**[20.611 0.308 0.167 0.137 0. ]**

**$V^T$ :**

**[-0.482 -0.482 0.413 0.413 0.44 ]  
[ 0.076 0.076 -0.365 -0.365 0.85 ]  
[-0.707 0.707 -0. -0. -0. ]  
[-0.511 -0.511 -0.443 -0.443 -0.289]  
[ 0. -0. 0.707 -0.707 -0. ]**



**X:**  
[5 5 0 0 1]  
[4 5 1 1 0]  
[5 4 1 1 0]  
[0 0 4 4 4]  
[0 0 5 5 5]  
[1 1 4 4 4]

**Covariance Matrix for  $X^T$ :**  
[ 5.36 4.16 4.16 -4.48 -5.6 -3.36]  
[ 4.16 3.76 3.56 -3.68 -4.6 -2.76]  
[ 4.16 3.56 3.76 -3.68 -4.6 -2.76]  
[-4.48 -3.68 -3.68 3.84 4.8 2.88]  
[-5.60 -4.60 -4.6 4.8 6.0 3.6 ]  
[-3.36 -2.76 -2.76 2.88 3.6 2.16]

**SVD applied on Covariance Matrix of  $X^T$ :**

**U:**  
[-0.462 0.669 -0. -0.486 0.31 0.087]  
[-0.383 -0.518 0.707 -0.243 0.155 0.043]  
[-0.383 -0.518 -0.707 -0.243 0.155 0.043]  
[ 0.397 -0.071 0. -0.289 0.492 -0.715]  
[ 0.497 -0.088 -0. -0.72 -0.3 0.37 ]  
[ 0.298 -0.053 0. 0.21 0.723 0.584]

**D:**  
[24.292 0.388 0.2 0. 0. 0. ]

**$v^T$ :**  
[-0.462 -0.383 -0.383 0.397 0.497 0.298]  
[ 0.669 -0.518 -0.518 -0.071 -0.088 -0.053]  
[-0. 0.707 -0.707 0. -0. 0. ]  
[-0.462 -0.231 -0.231 -0.253 -0.74 0.261]  
[-0.328 -0.164 -0.164 -0.608 0.298 -0.616]  
[-0.135 -0.067 -0.067 0.635 -0.33 -0.679]

# Scatter Matrices and Separability criteria

Scatter matrices used to formulate criteria of class separability:

❖ **Within-class scatter Matrix:** It shows the scatter of samples around their respective class expected vectors.

$$S_W = \sum_{i=1}^c \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

❖ **Between-class scatter Matrix:** It is the scatter of the expected vectors around the mixture mean..... $\mu$  is the mixture mean..

$$S_B = \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

# Scatter Matrices and Separability criteria

- ❖ **Mixture scatter matrix:** It is the covariance matrix of all samples regardless of their class assignments.

$$S_T = \sum_{k=1}^N (x_k - \mu)(x_k - \mu)^T = S_W + S_B$$

- The criteria formulation for class separability needs to **convert these matrices into a number**.
- This number should be larger when between-class scatter is larger or the within-class scatter is smaller.

Several Criteria are..

$$J_1 = \text{tr}(S_2^{-1} S_1)$$

$$J_2 = \ln |S_2^{-1} S_1| = \ln |S_1| - \ln |S_2|$$

$$J_3 = \text{tr}(S_1) - \mu(\text{tr} S_2 - c)$$

$$J_4 = \frac{\text{tr} S_1}{\text{tr} S_2}$$

# Linear Discriminant Analysis

- Learning set is labeled – supervised learning
- Class specific method in the sense that it tries to ‘shape’ the scatter in order to make it more reliable for classification.
- Select  $W$  to maximize the ratio of the between-class scatter and the within-class scatter.

Between-class scatter matrix is defined by-

$$S_B = \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$\mu_i$  is the mean of class  $X_i$

$N_i$  is the no. of samples in class  $X_i$

Within-class scatter matrix

is:

$$S_W = \sum_{i=1}^c \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

# Linear Discriminant Analysis

- If  $S_W$  is nonsingular,  $W_{opt}$  is chosen to satisfy

$$W_{opt} = \arg \max \frac{|W^T S_B W|}{|W^T S_W W|}$$

$$W_{opt} = [w_1, w_2, \dots, w_m]$$

$\{w_i \mid i = 1, 2, \dots, m\}$  is the set of eigenvectors of  $S_B$  and  $S_W$  corresponding to  $m$  largest eigen values. i.e.

$$S_B w_i = \lambda_i S_W w_i$$

- There are at most  $(c-1)$  non-zero eigen values. So upper bound of  $m$  is  $(c-1)$ .

# Linear Discriminant Analysis

$S_W$  is singular most of the time. It's rank is at most  $N-c$

Solution – Use an alternative criterion.

- Project the samples to a lower dimensional space.
- Use PCA to reduce dimension of the feature space to  $N-c$ .
- Then apply standard FLD to reduce dimension to  $c-1$ .

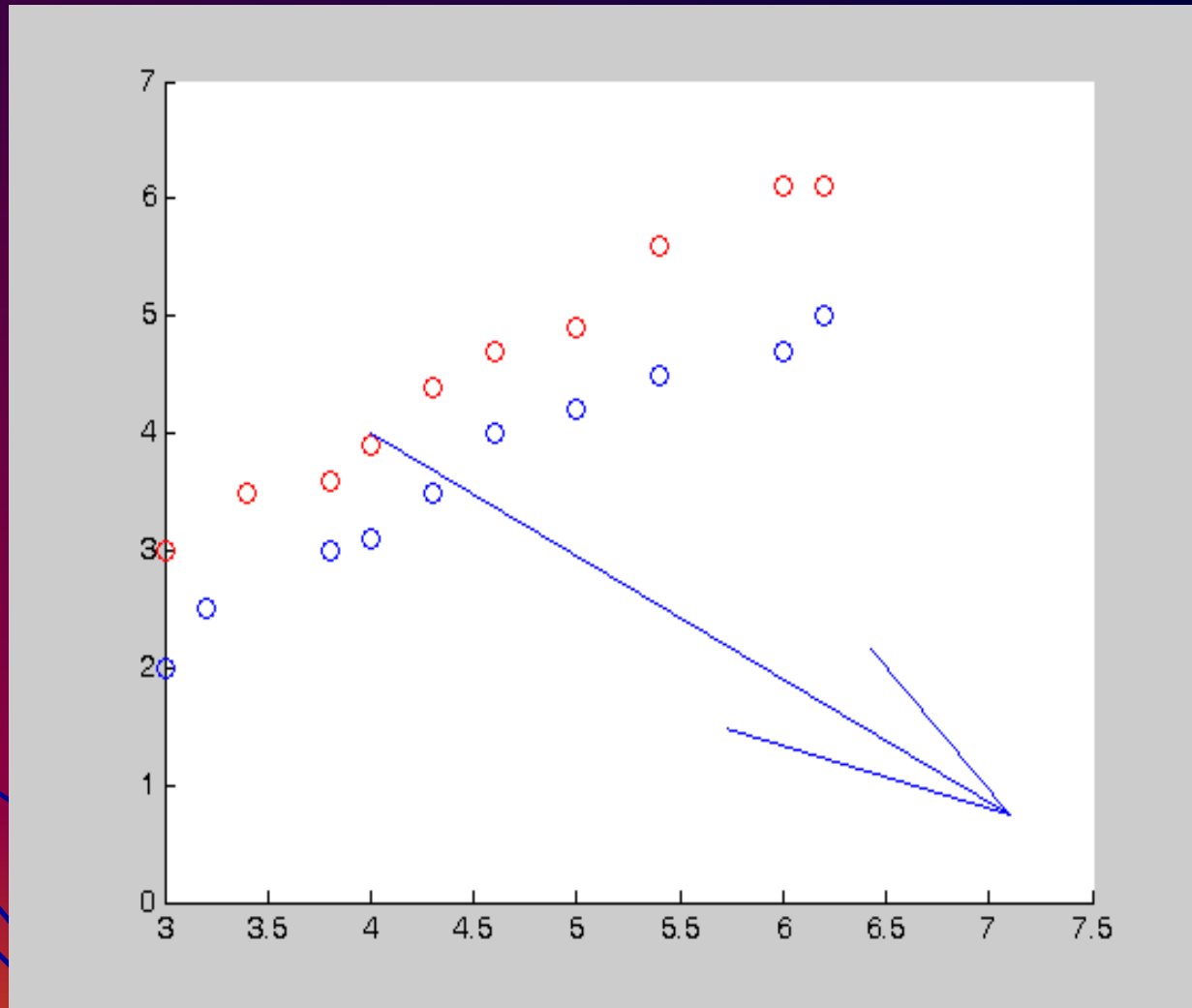
$W_{opt}$  is given by

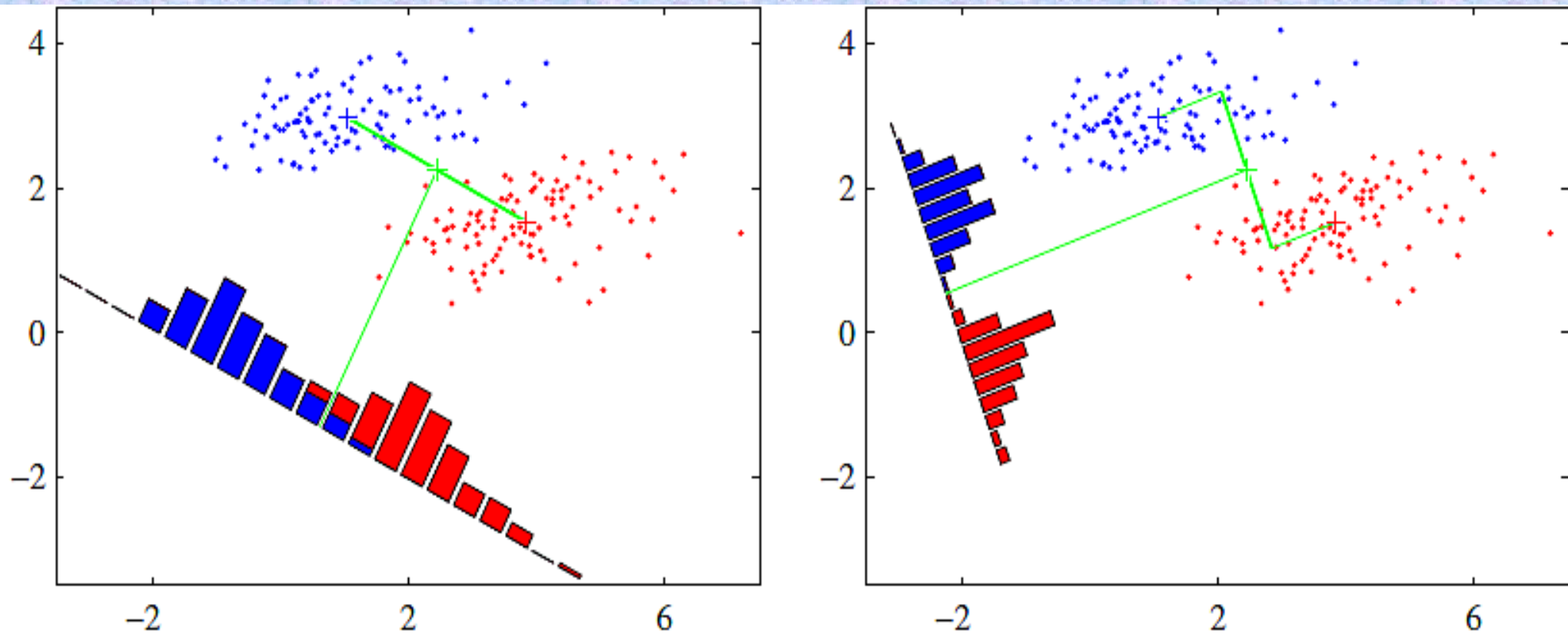
$$W_{opt} = W_{fld}^T W_{pca}^T$$

$$W_{pca} = \arg \max_W |W^T S_T W|$$

$$W_{fld} = \arg \max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

# Demonstration for LDA





**Figure 4.6** The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.



consider a two-class problem in which there are  $N_1$  points of class  $\mathcal{C}_1$  and  $N_2$  points of class  $\mathcal{C}_2$ , so that the mean vectors of the two classes are given by

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n. \quad (4.21)$$

The simplest measure of the separation of the classes, when projected onto  $\mathbf{w}$ , is the separation of the projected class means. This suggests that we might choose  $\mathbf{w}$  so as to maximize

into a labelled set in the one-dimensional space  $y$ . The within-class variance of the transformed data from class  $\mathcal{C}_k$  is therefore given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \quad (4.24)$$

where  $y_n = \mathbf{w}^T \mathbf{x}_n$ . We can define the total within-class variance for the whole data set to be simply  $s_1^2 + s_2^2$ . The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}. \quad (4.25)$$

rewrite the Fisher criterion in the form  $J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$  (4.26)

where  $\mathbf{S}_B$  is the *between-class* covariance matrix and is given by

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \quad (4.27)$$

and  $\mathbf{S}_W$  is the total *within-class* covariance matrix, given by

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T. \quad (4.28)$$

Differentiating (4.26) with respect to  $\mathbf{w}$ , we find that  $J(\mathbf{w})$  is maximized when

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}. \quad (4.29)$$

From (4.27), we see that  $\mathbf{S}_B \mathbf{w}$  is always in the direction of  $(\mathbf{m}_2 - \mathbf{m}_1)$ . Furthermore, we do not care about the magnitude of  $\mathbf{w}$ , only its direction, and so we can drop the scalar factors  $(\mathbf{w}^T \mathbf{S}_B \mathbf{w})$  and  $(\mathbf{w}^T \mathbf{S}_W \mathbf{w})$ . Multiplying both sides of (4.29) by  $\mathbf{S}_W^{-1}$  we then obtain

$$\mathbf{w} \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1). \quad (4.30)$$

Note that if the within-class covariance is isotropic, so that  $\mathbf{S}_W$  is proportional to the unit matrix, we find that  $\mathbf{w}$  is proportional to the difference of the class means, as discussed above.

The result (4.30) is known as *Fisher's linear discriminant*, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension. However, the projected data can subsequently be used

$$\mathbf{S}_W = \sum_{k=1}^K \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^T \quad (4.47)$$

and

$$\mathbf{S}_B = \sum_{k=1}^K N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \quad (4.48)$$

where

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{y}_n, \quad \boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^K N_k \boldsymbol{\mu}_k. \quad (4.49)$$

Again we wish to construct a scalar that is large when the between-class covariance is large and when the within-class covariance is small. There are now many possible choices of criterion (Fukunaga, 1990). One example is given by

$$J(\mathbf{W}) = \text{Tr} \{ \mathbf{S}_W^{-1} \mathbf{S}_B \}. \quad (4.50)$$

Again we wish to construct a scalar that is large when the between-class covariance is large and when the within-class covariance is small. There are now many possible choices of criterion (Fukunaga, 1990). One example is given by

$$J(\mathbf{W}) = \text{Tr} \{ \mathbf{S}_W^{-1} \mathbf{S}_B \}. \quad (4.50)$$

This criterion can then be rewritten as an explicit function of the projection matrix  $\mathbf{W}$  in the form

$$J(\mathbf{w}) = \text{Tr} \{ (\mathbf{W} \mathbf{S}_W \mathbf{W}^T)^{-1} (\mathbf{W} \mathbf{S}_B \mathbf{W}^T) \}. \quad (4.51)$$

Maximization of such criteria is straightforward, though somewhat involved, and is discussed at length in Fukunaga (1990). The weight values are determined by those eigenvectors of  $\mathbf{S}_W^{-1} \mathbf{S}_B$  that correspond to the  $D'$  largest eigenvalues.

There is one important result that is common to all such criteria, which is worth emphasizing. We first note from (4.46) that  $\mathbf{S}_B$  is composed of the sum of  $K$  matrices, each of which is an outer product of two vectors and therefore of rank 1. In addition, only  $(K - 1)$  of these matrices are independent as a result of the constraint (4.44). Thus,  $\mathbf{S}_B$  has rank at most equal to  $(K - 1)$  and so there are at most  $(K - 1)$  nonzero eigenvalues. This shows that the projection onto the  $(K - 1)$ -dimensional subspace spanned by the eigenvectors of  $\mathbf{S}_B$  does not alter the value of  $J(\mathbf{w})$ , and so we are therefore unable to find more than  $(K - 1)$  linear 'features' by this means (Fukunaga, 1990).

# Hand workout EXAMPLE:

Data Points:      **1 2 3 5 4 6 8 -2 -1 1 3 4 2 5**  
                         **1 2 3 4 5 6 7 3 4 5 6 7 8 9**

Class:              **1 1 1 1 1 1 1 2 2 2 2 2 2 2**

Lets try **PCA** first :

Overall data mean:    **2.9286**  
                             **5.0000**

COVAR of the mean-subtracted data:

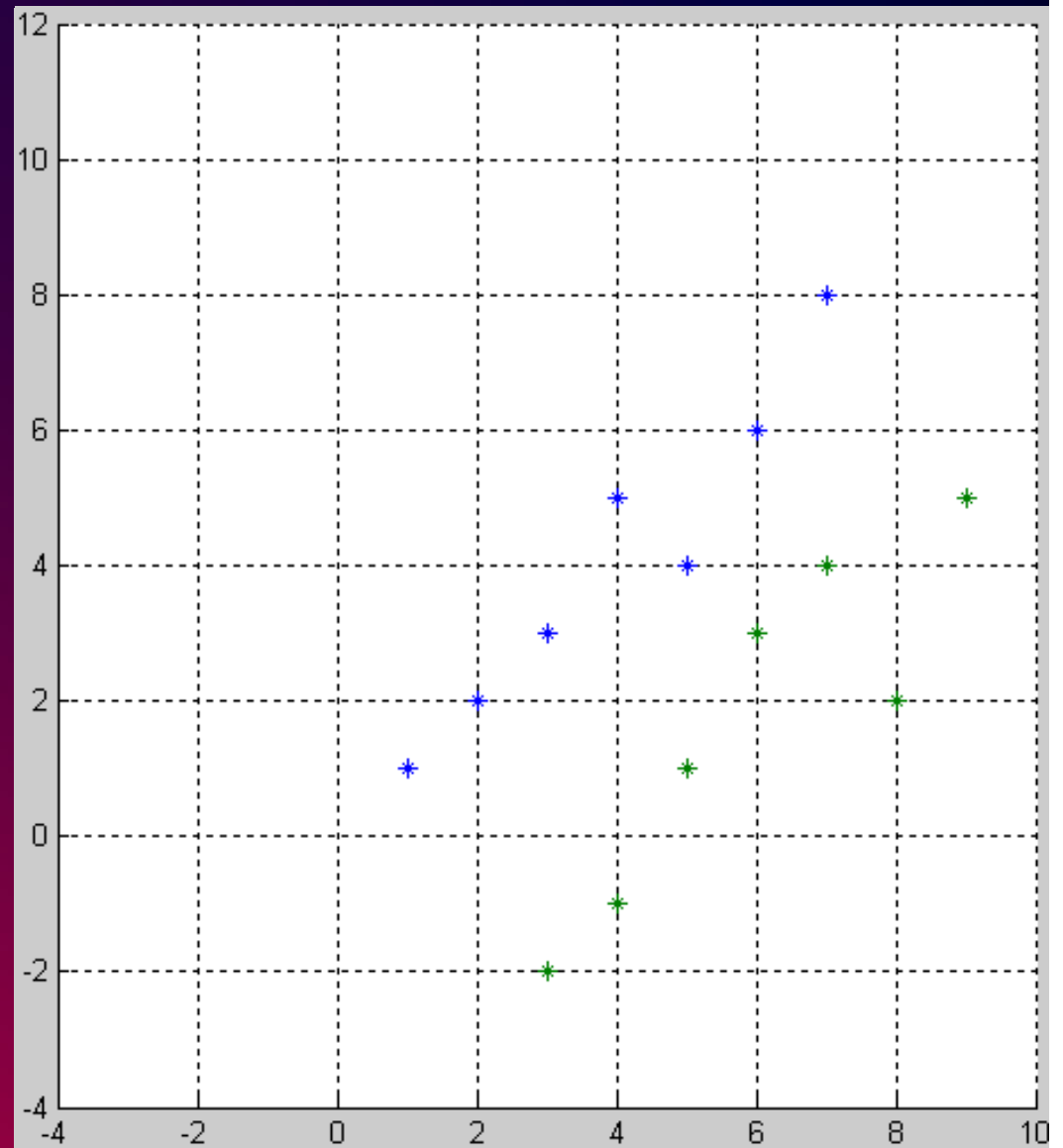
**7.3022 3.3077**  
**3.3077 5.3846**

Eigenvalues after SVD of above:

**9.7873 2.8996**

Finally, the eigenvectors:

**-0.7995 -0.6007**  
**-0.6007 0.7995**



Same EXAMPLE for LDA :

Data Points:	1	2	3	5	4	6	8	-2	-1	1	3	4	2	5
	1	2	3	4	5	6	7	3	4	5	6	7	8	9
Class:	1	1	1	1	1	1	1	2	2	2	2	2	2	2

$$S_w = \begin{bmatrix} 10.6122 & 8.5714 \\ 8.5714 & 8.0000 \end{bmatrix}$$

$$S_b = \begin{bmatrix} 20.6429 & -17.00 \\ -17.00 & 14.00 \end{bmatrix}$$

$$\text{INV}(S_w) \cdot S_b = \begin{bmatrix} 27.20 & -22.40 \\ -31.268 & 25.75 \end{bmatrix}$$

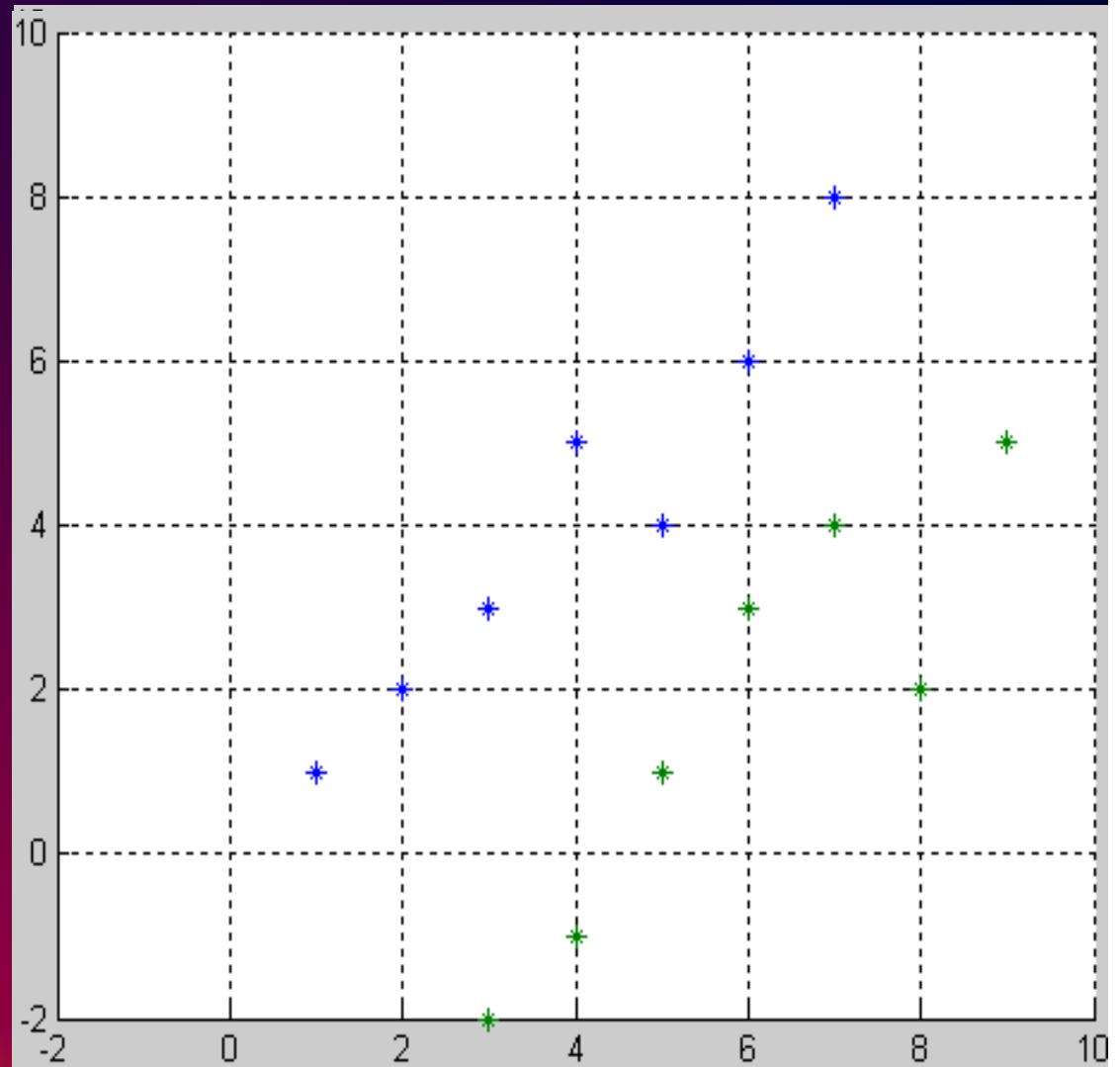
$$\begin{bmatrix} 27.20 & -22.40 \\ -31.268 & 25.75 \end{bmatrix}$$

Perform Eigendecomposition on above:

$$\text{Eigenvalues of } S_w^{-1} S_b : \begin{bmatrix} 53.687 \\ 0 \end{bmatrix}$$

Eigenvectors:

$$\begin{bmatrix} -0.7719 & 0.6357 \\ 0.6357 & 0.7719 \end{bmatrix}$$

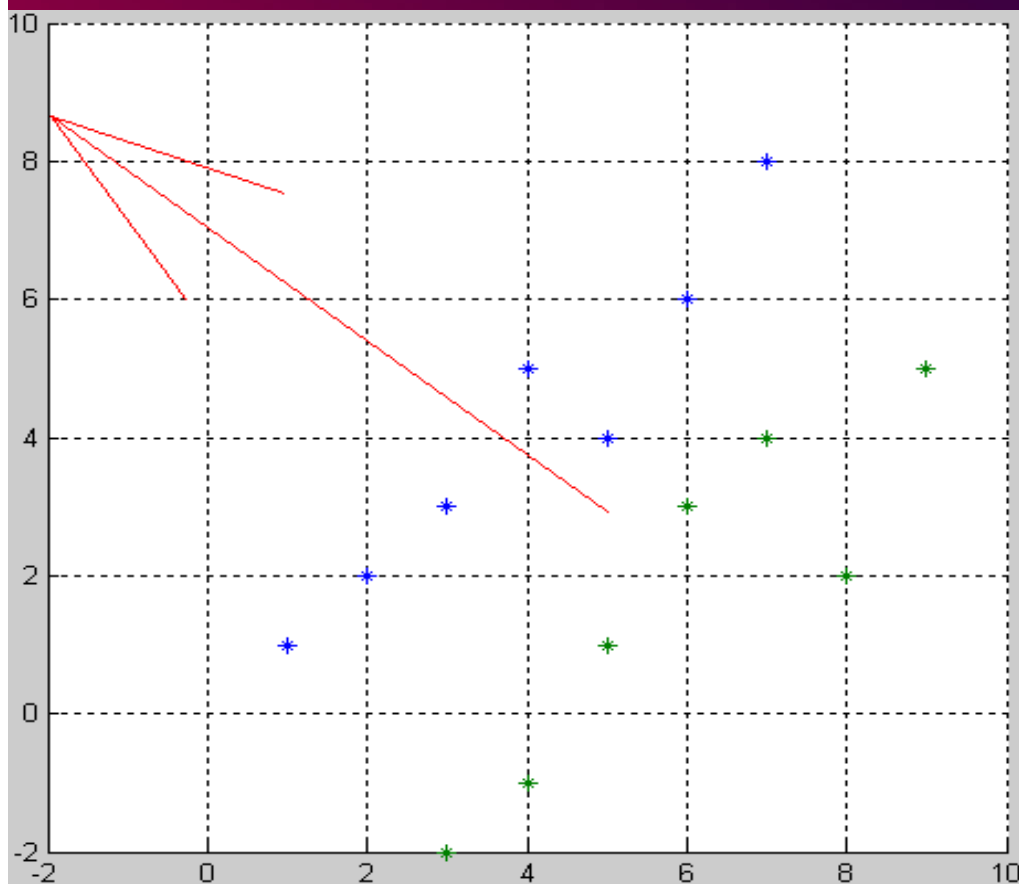


$$S_w = \begin{bmatrix} 10.6122 & 8.5714 \\ 8.5714 & 8.0000 \end{bmatrix}$$

$$S_b = \begin{bmatrix} 20.6429 & -17.00 \\ -17.00 & 14.00 \end{bmatrix}$$

$$\text{Eigenvalues of } S_w^{-1} S_b : \begin{matrix} 53.687 \\ 0 \end{matrix}$$

$$\text{Eigenvectors: } \begin{bmatrix} -0.7719 & 0.6357 \\ 0.6357 & 0.7719 \end{bmatrix}$$

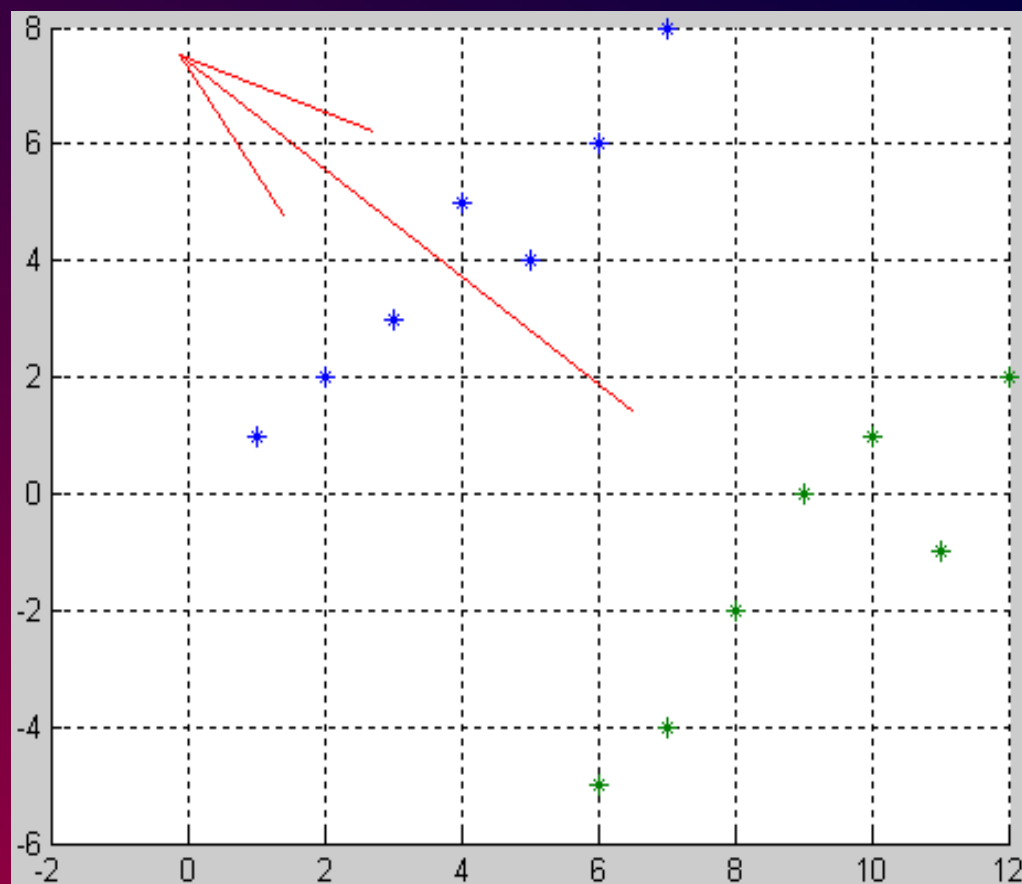


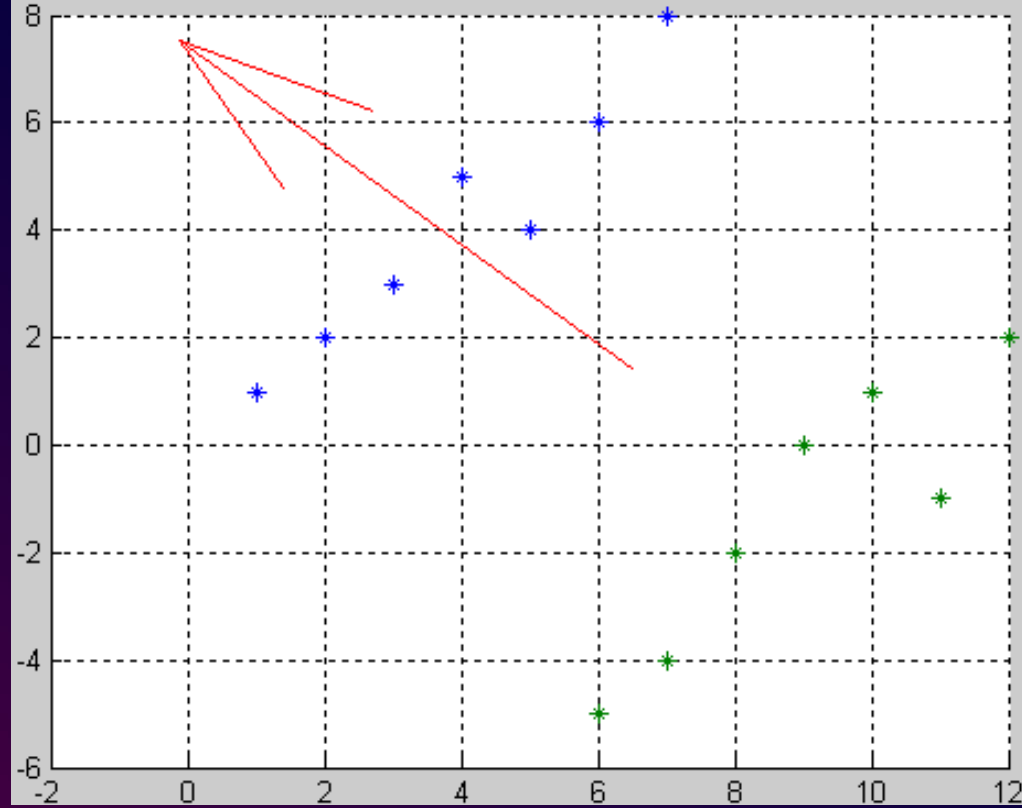
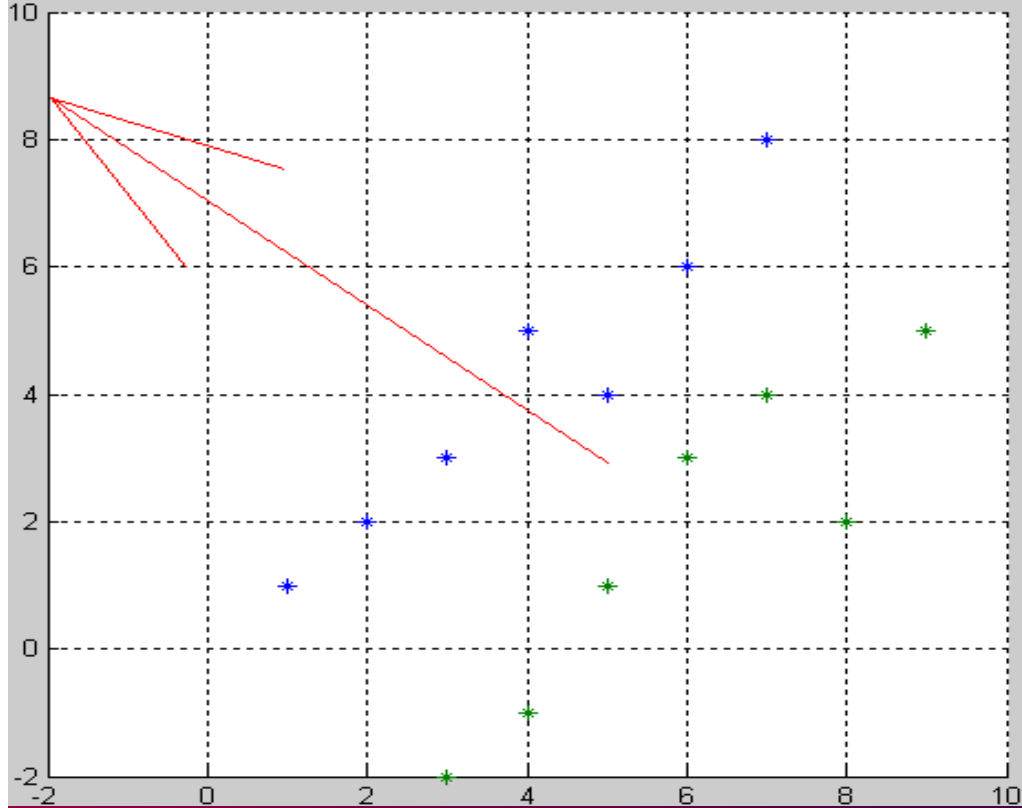
$$S_w = \begin{bmatrix} 10.6122 & 8.5714 \\ 8.5714 & 8.0000 \end{bmatrix}$$

$$S_b = \begin{bmatrix} 203.143 & -95.00 \\ -95.00 & 87.50 \end{bmatrix}$$

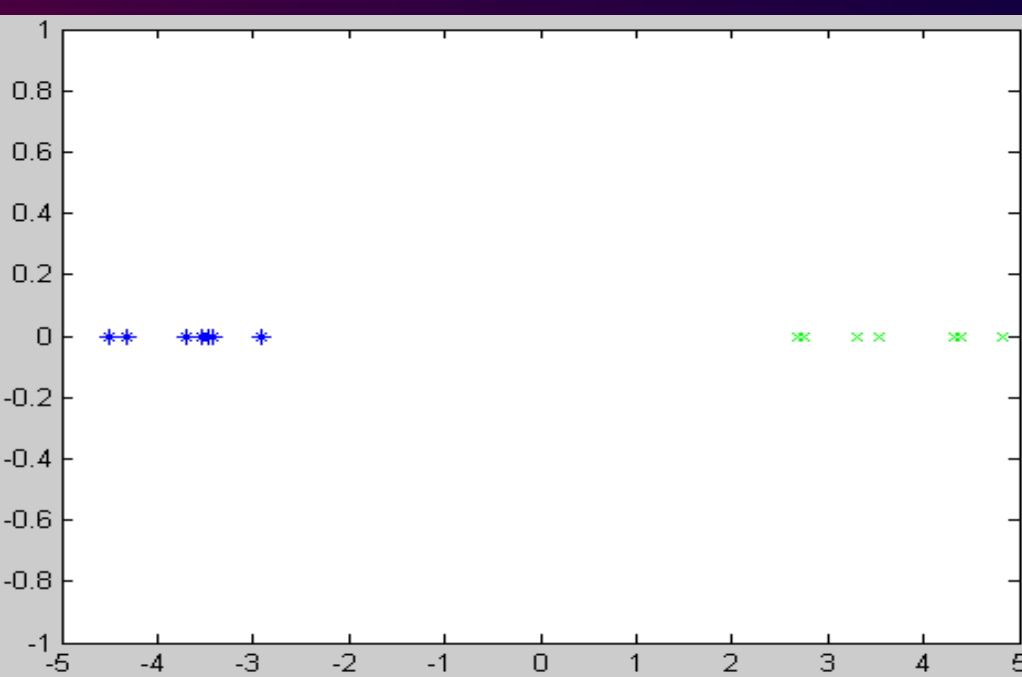
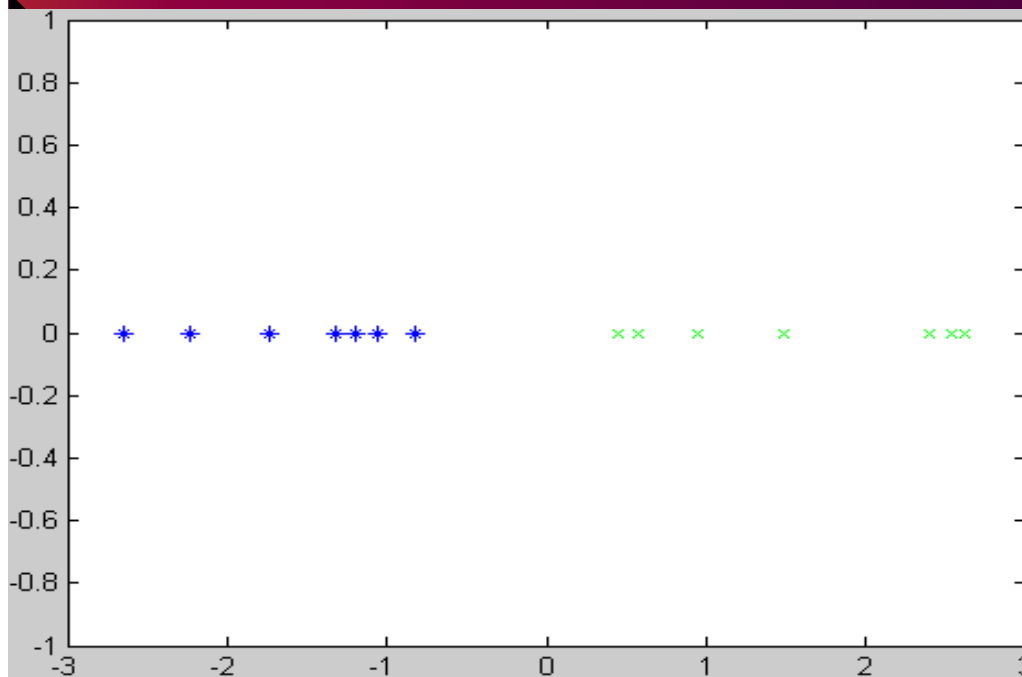
$$\text{Eigenvalues of } S_w^{-1} S_b : \begin{matrix} 297.83 \\ 0.0 \end{matrix}$$

$$\text{Eigenvectors: } \begin{bmatrix} -0.7355 & -0.6775 \\ 0.6775 & 0.7355 \end{bmatrix}$$





**After linear projection, using LDA:**





Same EXAMPLE for LDA, with  $C = 3$ :

Data Points:	1	2	3	5	4	6	8	-2	-1	1	3	4	2	5
	1	2	3	4	5	6	7	3	4	5	6	7	8	9
Class:	1	1	1	2	2	3	3	1	1	1	2	2	3	3

$$S_w = \begin{bmatrix} 8.0764 & -2.125 \\ -2.125 & 4.1667 \end{bmatrix}$$

$$S_b = \begin{bmatrix} 56.845 & 52.50 \\ 52.50 & 50.00 \end{bmatrix}$$

$$\text{INV}(S_w) \cdot S_b =$$

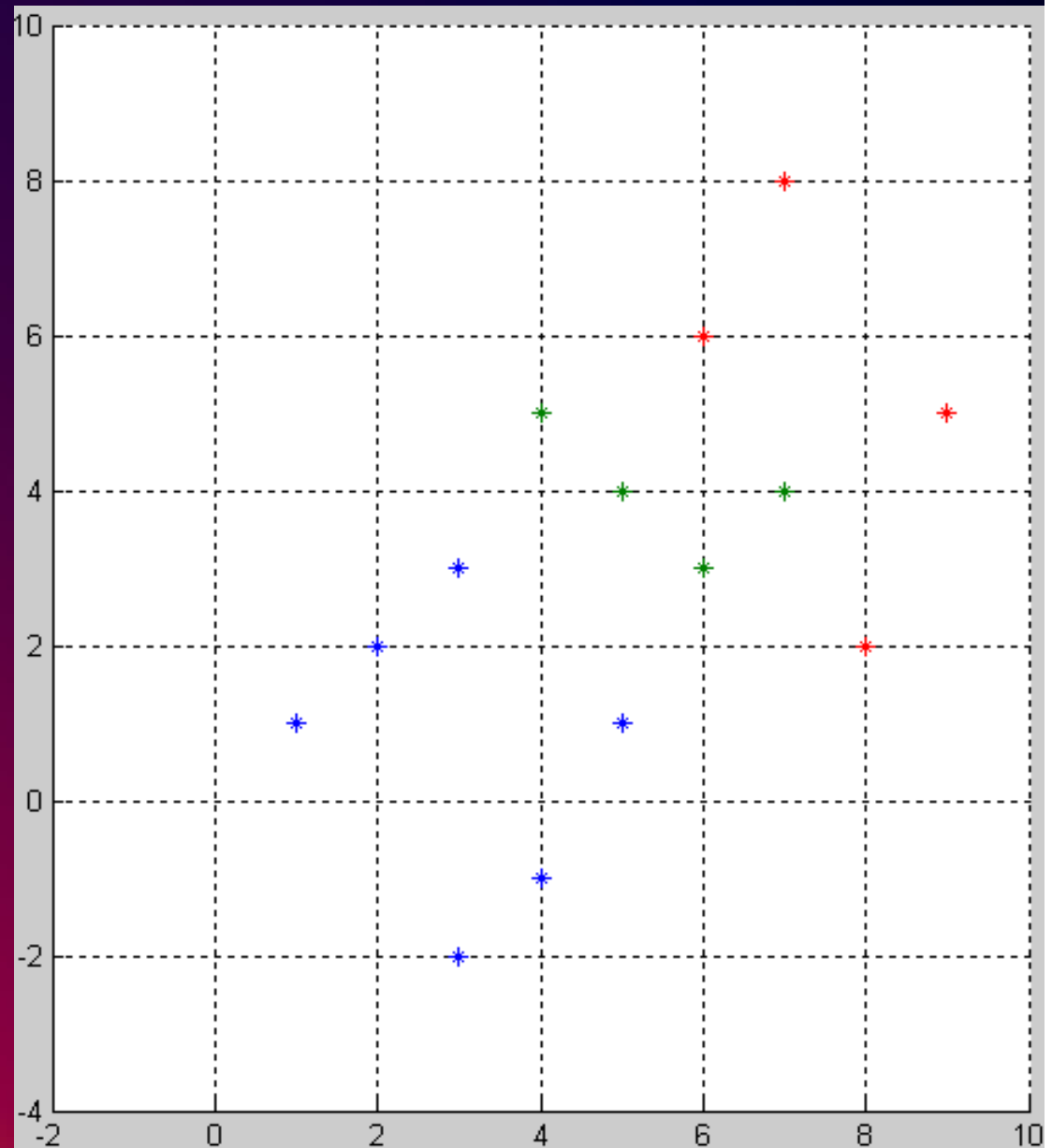
$$\begin{bmatrix} 11.958 & 11.155 \\ 18.7 & 17.69 \end{bmatrix}$$

Perform Eigendecomposition on above:

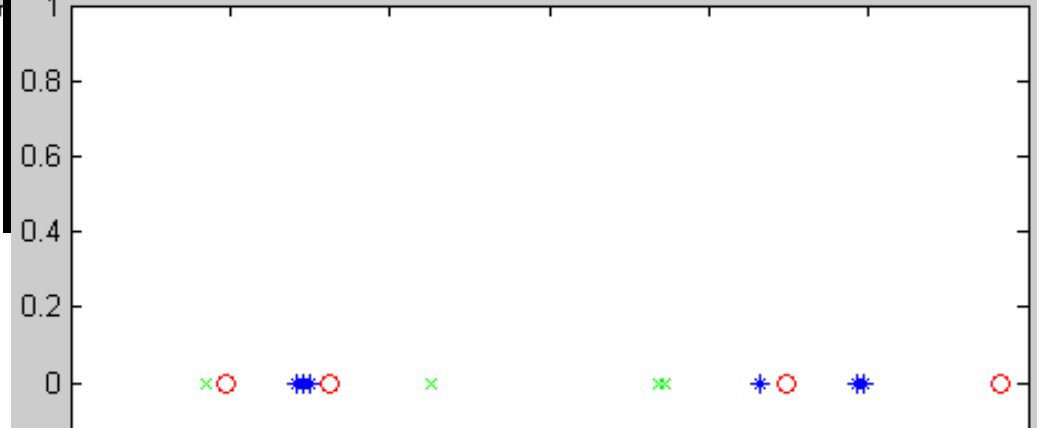
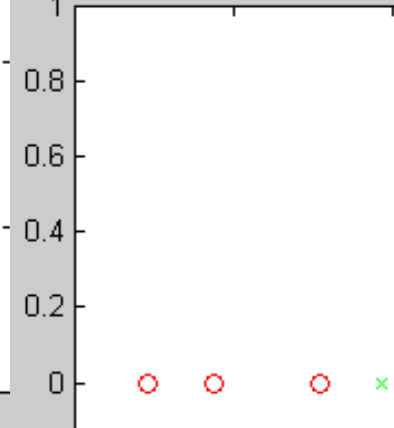
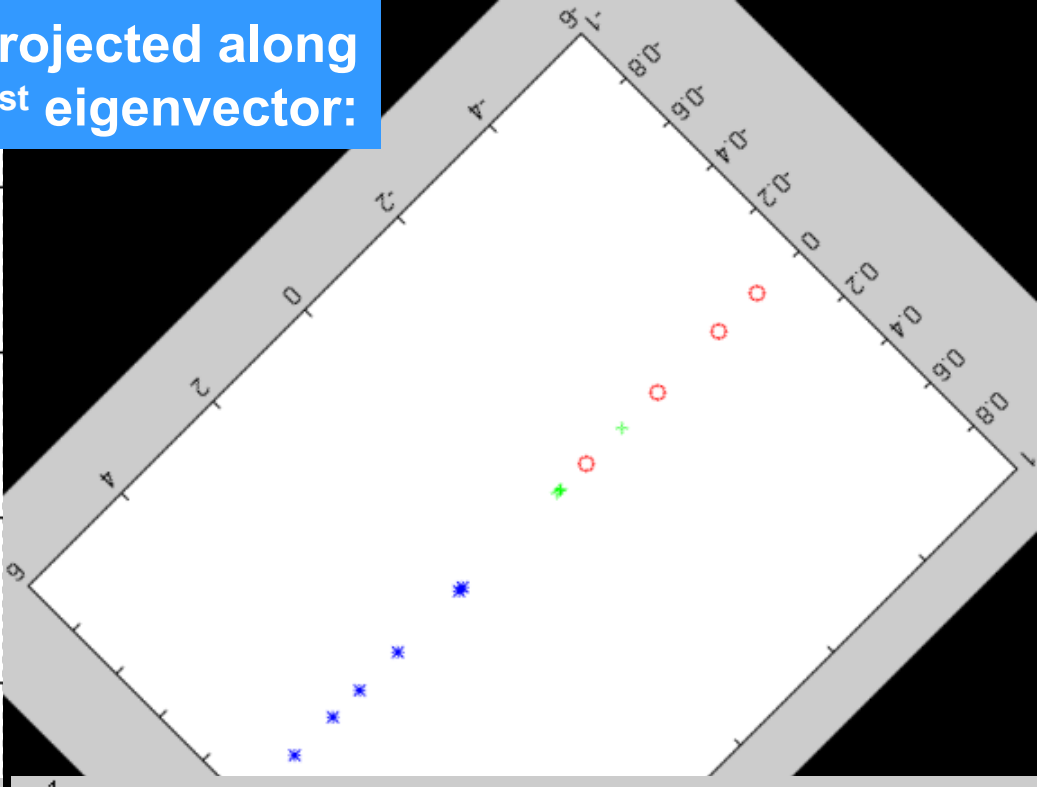
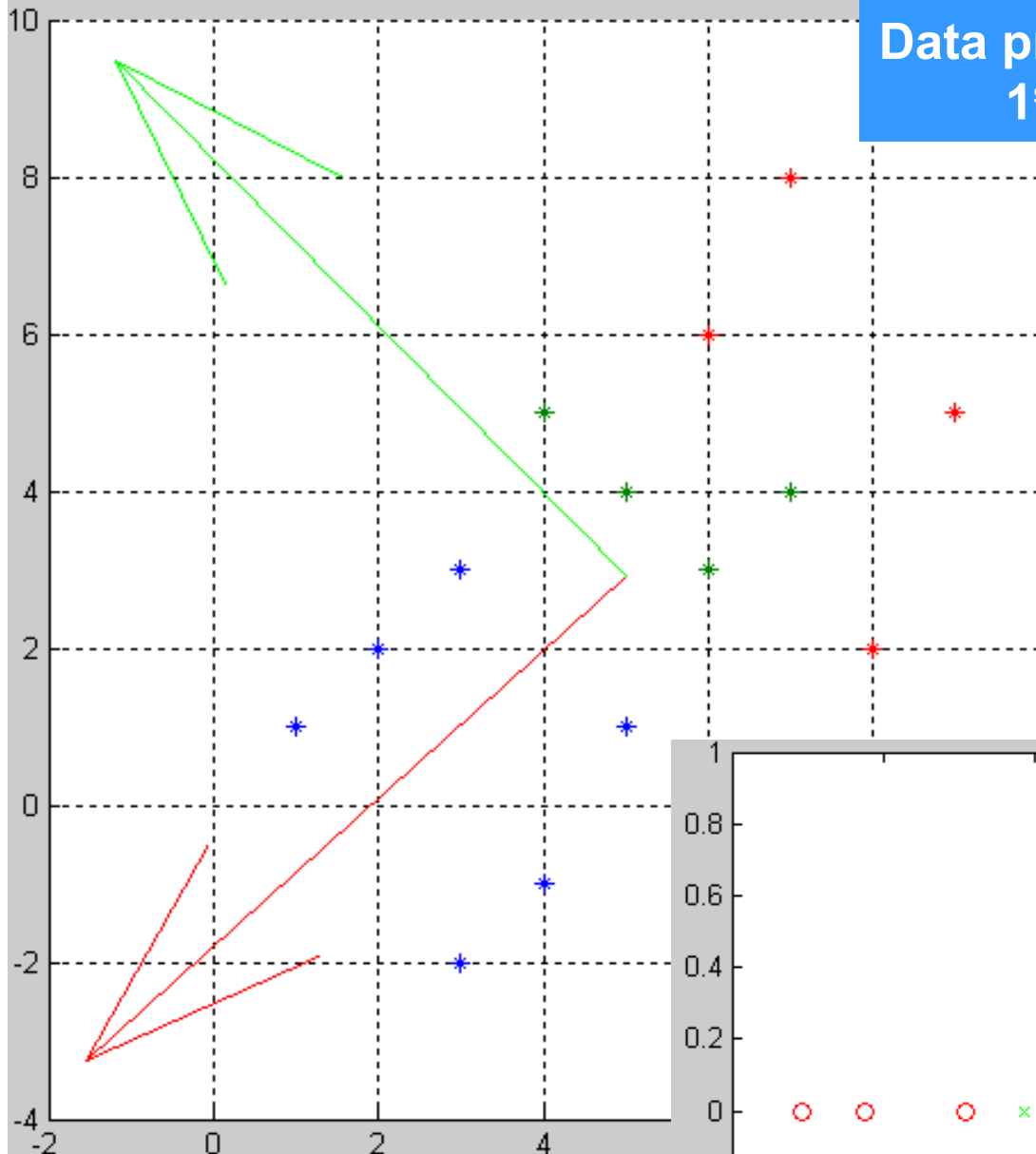
$$\text{Eigenvalues of } S_w^{-1} S_b : \begin{matrix} 30.5 \\ 0.097 \end{matrix}$$

Eigenvectors:

$$\begin{bmatrix} -0.728 & -0.69 \\ -0.69 & 0.728 \end{bmatrix}$$



Data projected along  
1<sup>st</sup> eigenvector:



Data projected along  
2<sup>nd</sup> eigenvector:

Hence, one may need ICA

Some of the latest **advancements in Pattern recognition** technology deal with:

- **Neuro-fuzzy (soft computing) concepts**
- **Multi-classifier Combination – decision and feature fusion**
- **Reinforcement learning**
- **Learning from small data sets**
- **Generalization capabilities**
- **Evolutionary Computations**
- **Genetic algorithms**
- **Pervasive computing**
- **Neural dynamics**
- **Support Vector machines - kernel methods**
- **Modern ML methods – transfer learning, domain adaptation, Manifold based learning, MKL, Co-training; Deep learning, semi-supervised, self-supervised, Weakly supervised, FSL,.....**

## REFERENCES

- **Statistical pattern Recognition; S. Fukunaga; Academic Press, 2000.**
- **Bishop – PR**
- **Satish Kumar - ANN**

References (for SVD)

**Golub, G.H., and Van Loan, C.F. (1989) Matrix Computations, 2nd ed. (Baltimore: Johns Hopkins University Press).**

**Greenberg, M. (2001) Differential equations & Linear algebra (Upper Saddle River, N.J. : Prentice Hall).**

**Strang, G. (1998) Introduction to linear algebra (Wellesley, MA : Wellesley-Cambridge Press).**

