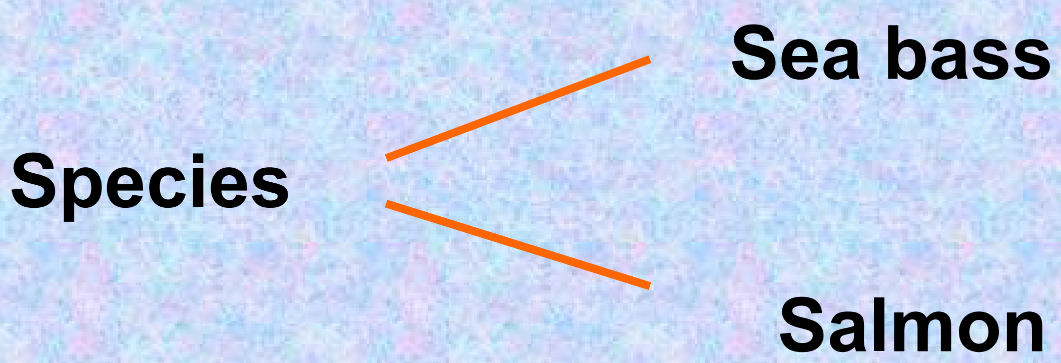
Pattern

Classification

An Example of Classification

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



Species

Sea bass

Salmon

- 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

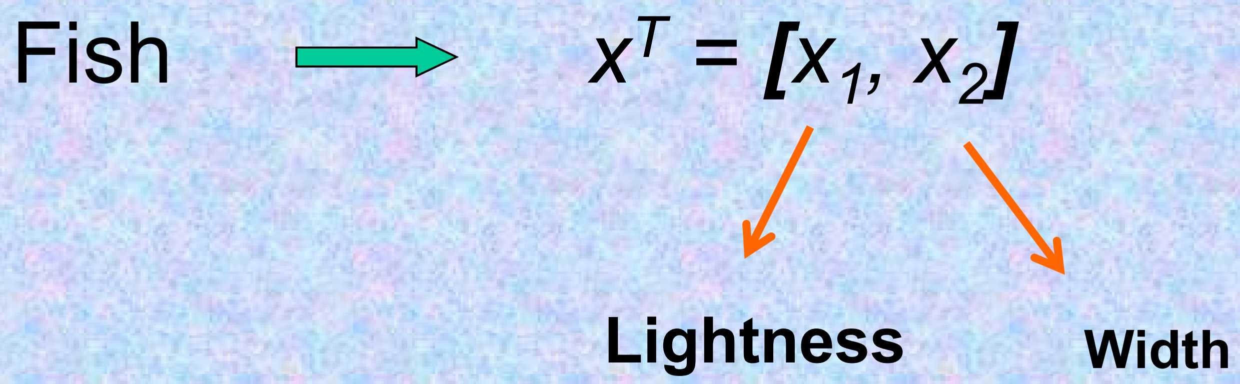
- This is the set of all suggested features to explore for use in our classifier!

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



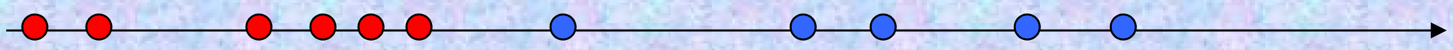
Fish $\longrightarrow$ $x^T = [x_1, x_2]$

Lightness

Width

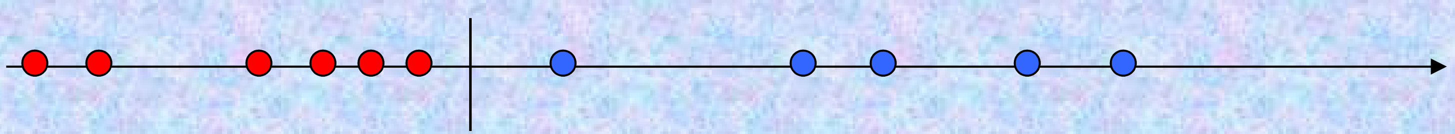
Feature space

- 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.

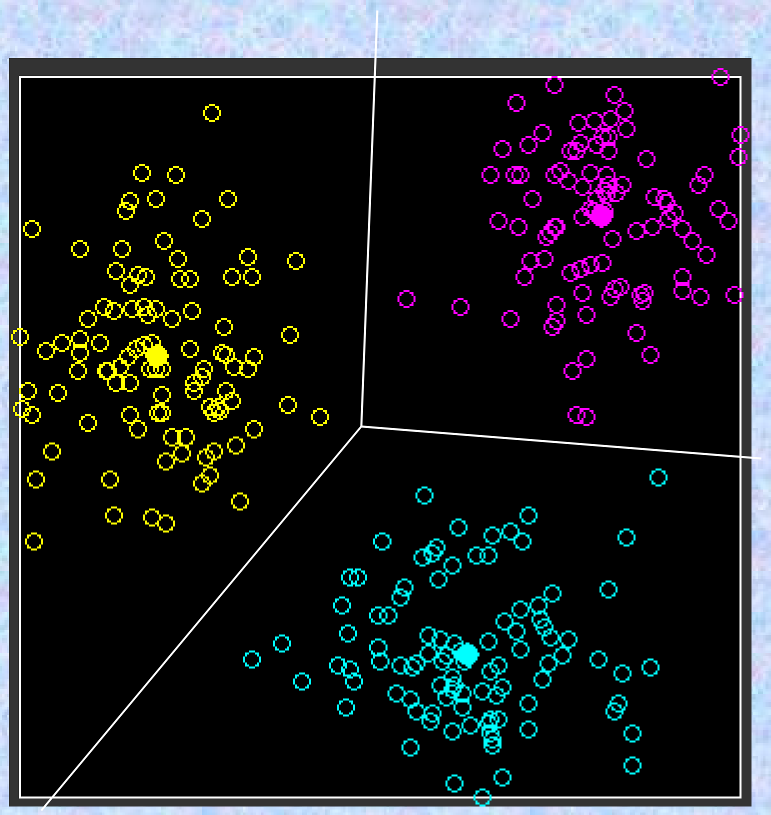


Point representing samples

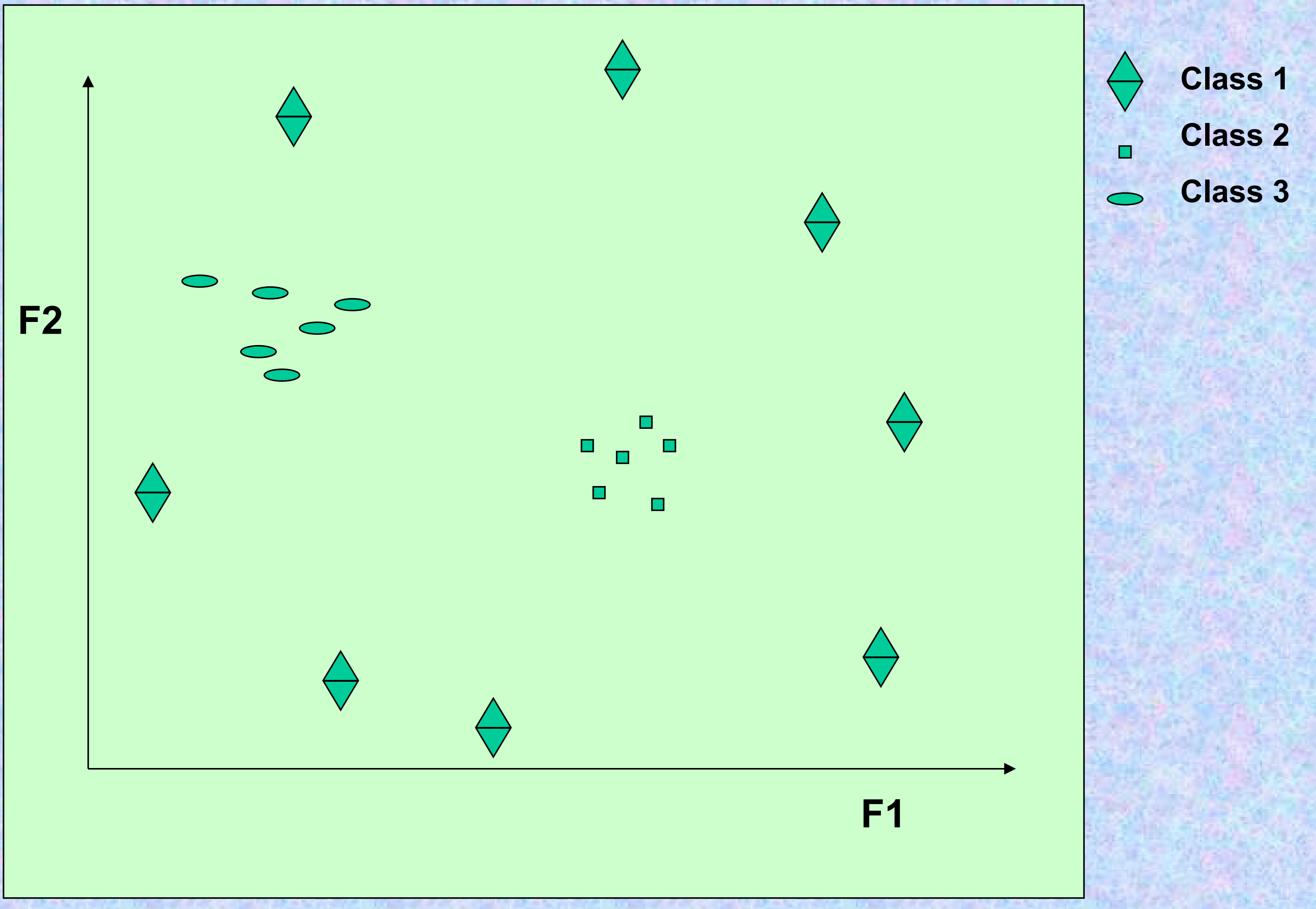
- If number of features is 2, then we get points in 2D-space as shown in the next slide.
- We can also have an n-dimensional feature space



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 be 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.

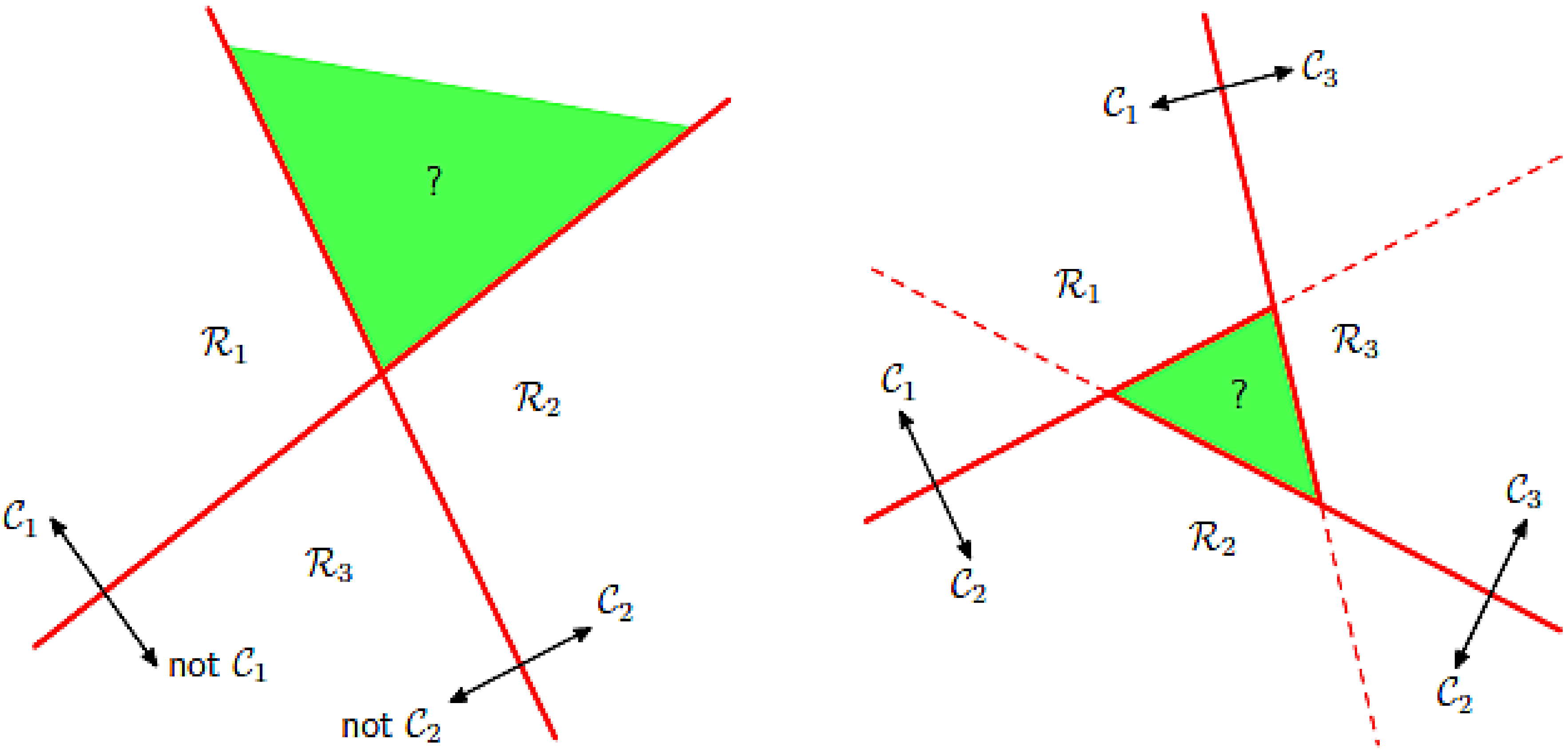


Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$.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

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.

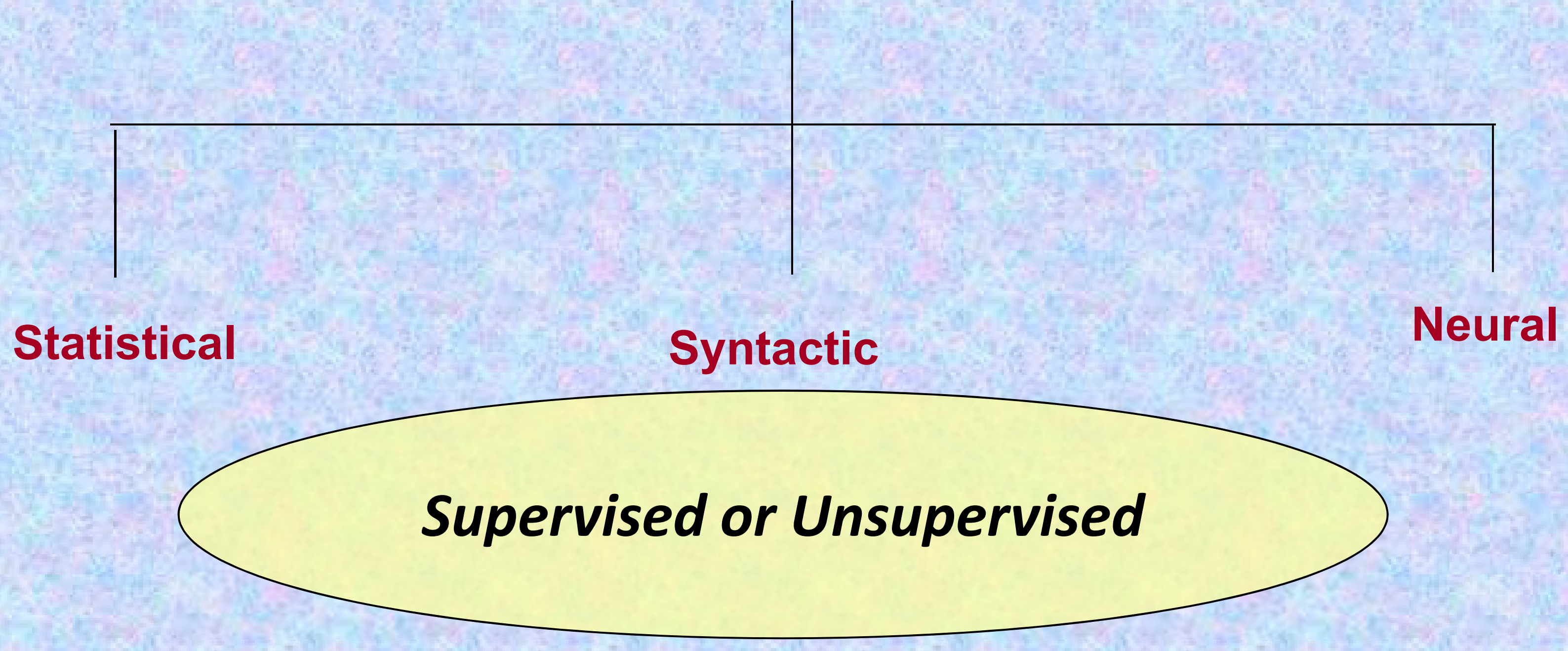
- If the Learning is offline i.e. Supervised method then, the classifier is first given a set of training samples and the optimal decision boundary found, and then the classification is done.

- 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



Statistical

Syntactic

Neural

Supervised or Unsupervised

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:

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

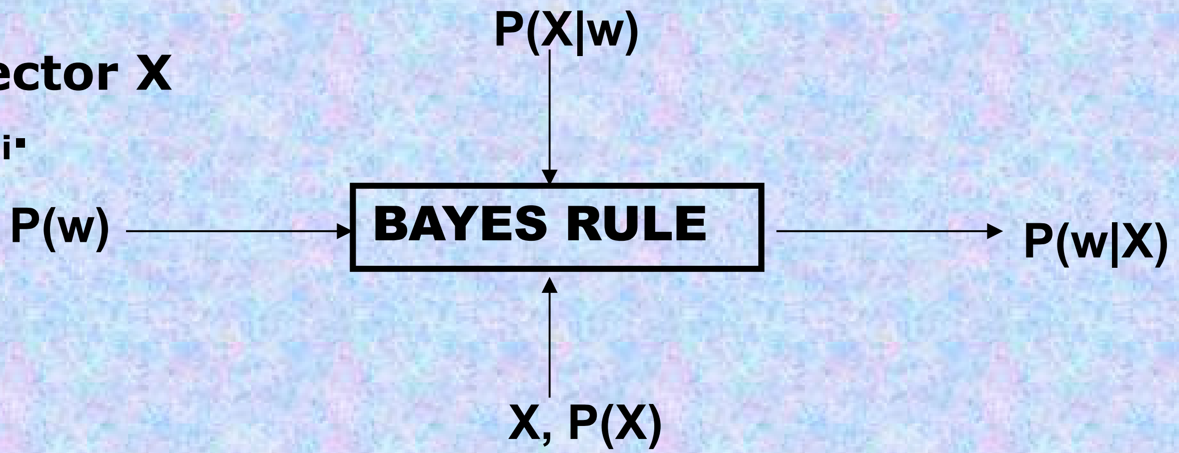
P(X) is the probability distribution for feature X in the entire population. Also called *unconditional density function (or evidence)*.

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

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

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

This is the Prob. of any vector X being assigned to class w_i.



P(X|w)

P(w)

BAYES RULE

P(w|X)

X, P(X)

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:

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

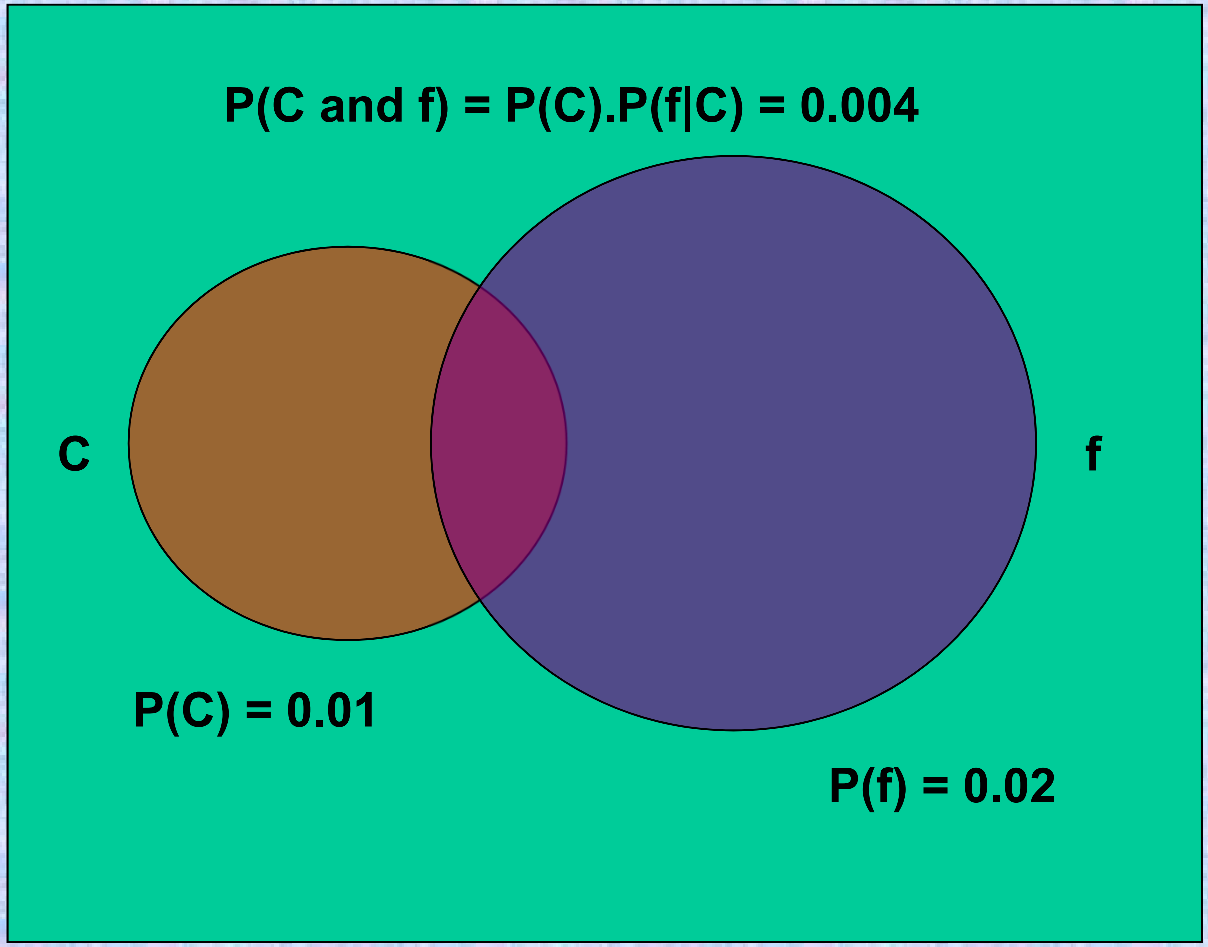
Not convinced that it works?
let us take an example with values to verify:

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:

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

Also verify, for a K class problem:

$P(X) = P(w_1)P(X|w_1) + P(w_2)P(X|w_2) + ... + P(w_k)P(X|w_k)$

Thus:

$$P(w_i \mid \overrightarrow{X}) = \frac{P(\overrightarrow{X} \mid w_i)P(w_i)}{P(w_1)P(X \mid w_1) + P(w_2)P(X \mid w_2) + ... + P(w_k)P(X \mid w_k)}$$

With our last example:

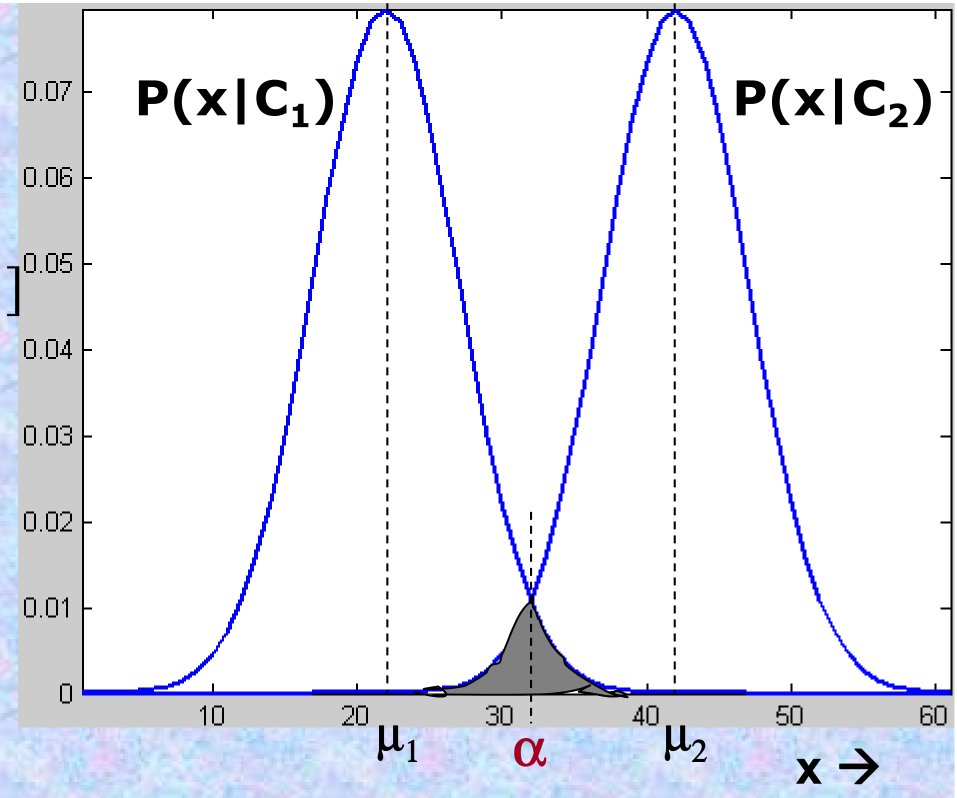
P(f) = P(C)P(f|C) + P(C')P(f|C')

= 0.01 *0.4 + 0.99 *0.01616 = 0.02

Decision or Classification algorithm according to Baye's Theorem:

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

Errors in decision making:



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

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

Bayes decision rule:

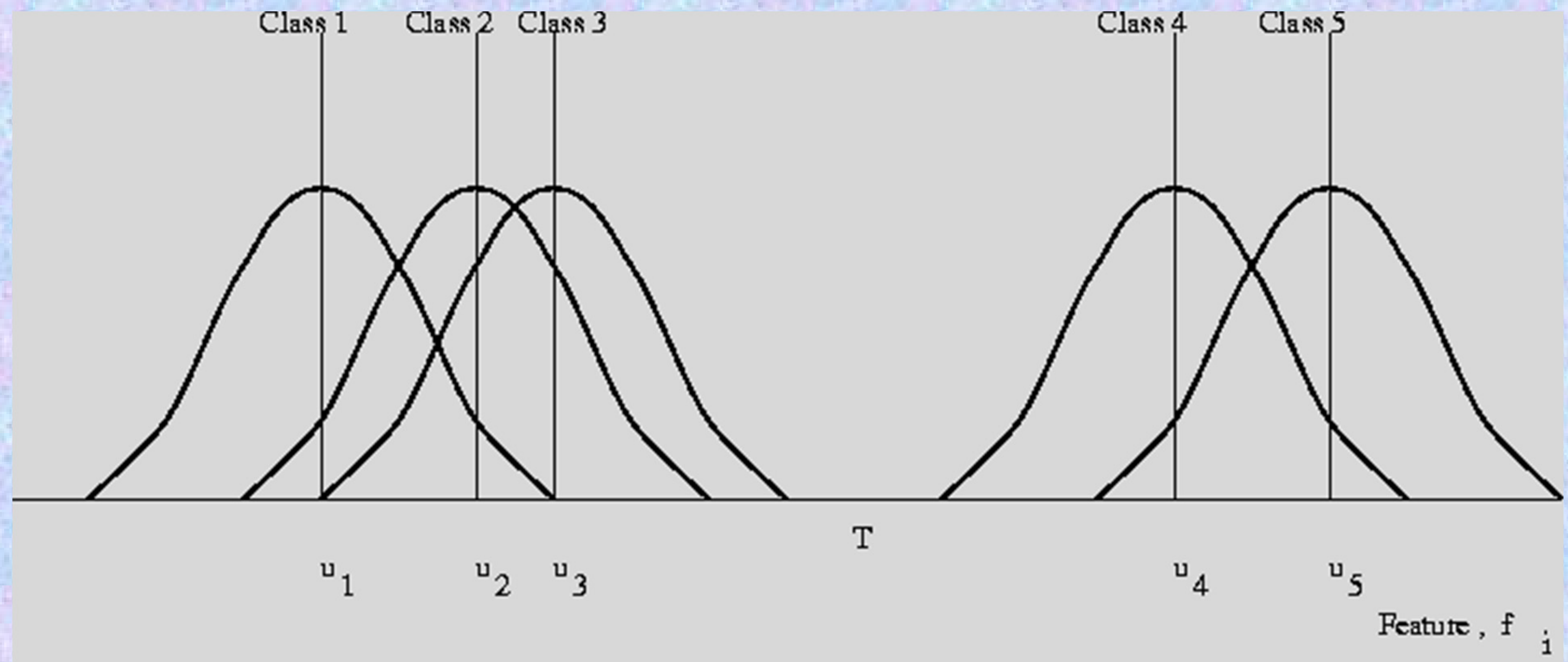
Choose C_1 , if $P(x|C_1) > P(x|C_2)$

This gives α, and hence the two decision regions.

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

P(E) = P(Chosen C_1, when x belongs to C_2) +
P(Chosen C_2, when x belongs to C_1)

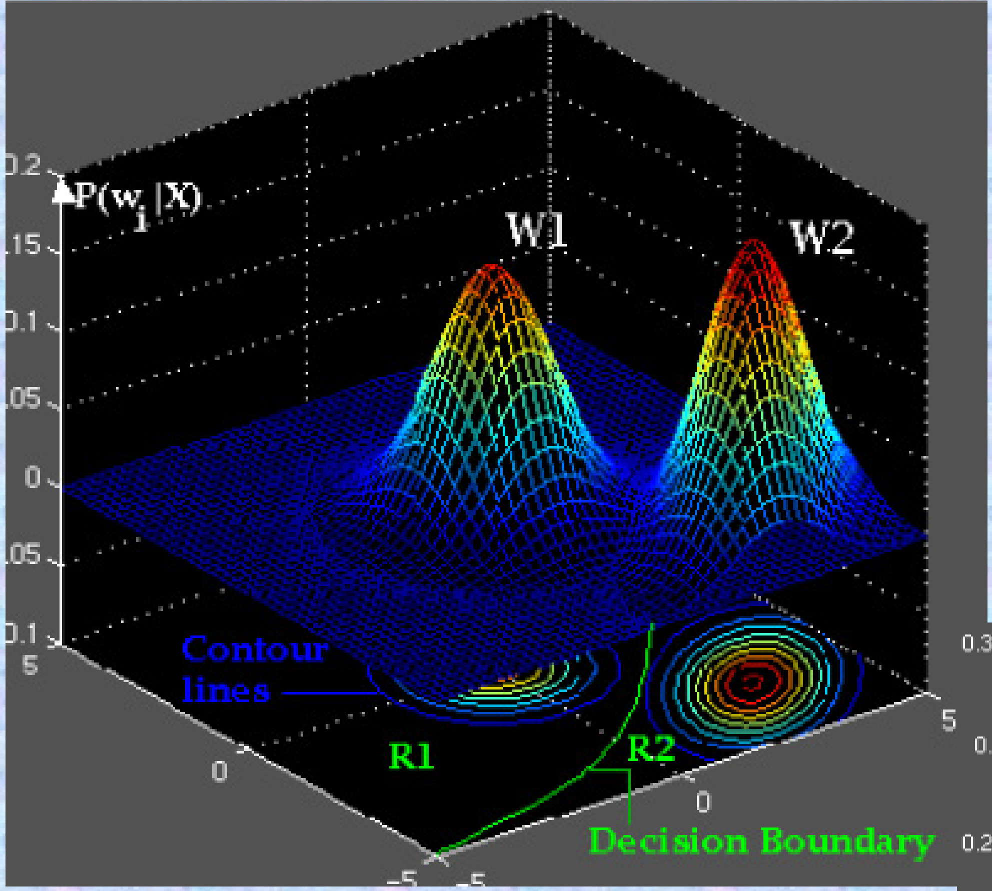
$$= P(C_2)\int_{-\infty}^{\alpha} P(\gamma \mid C_2)d\gamma + P(C_1)\int_{\alpha}^{-\infty} P(\gamma \mid C_1)d\gamma$$



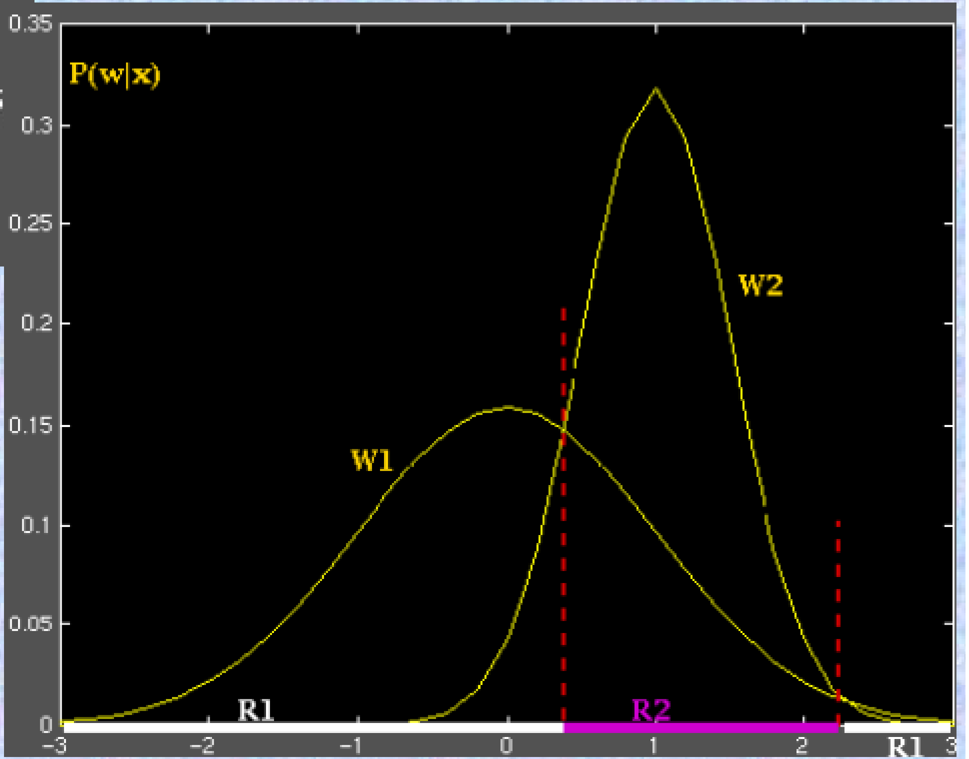
Normal distributions of feature measurement for a 5-class problem, equal variance.

A minimum distance (NN) supervised classifier

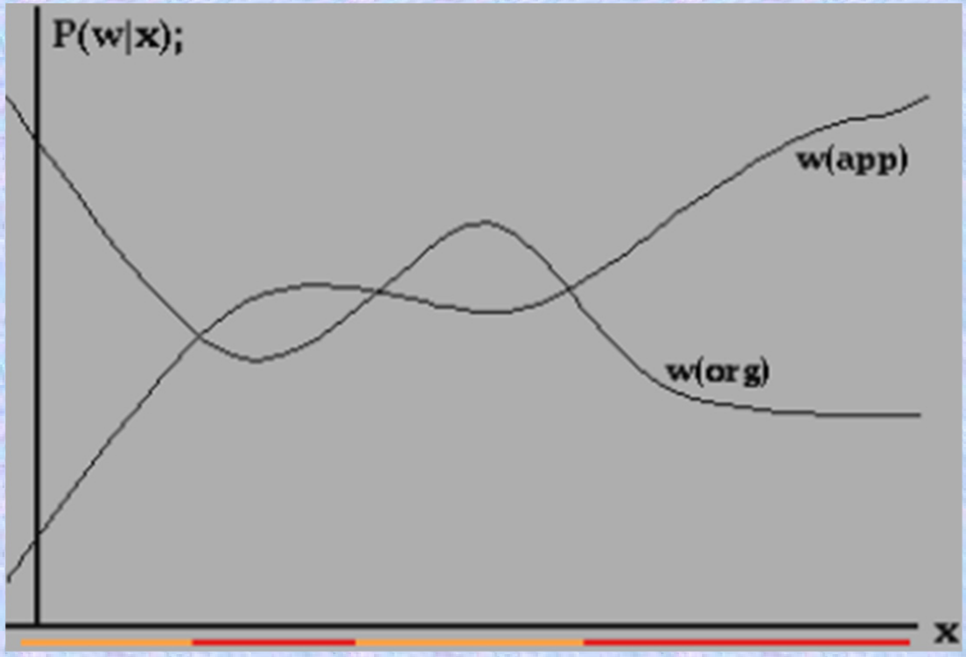
Rule: Assign X to R_i, where X is closest to μ_i.



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 <u>Discriminant functions</u> based on Baye’s decision rule:

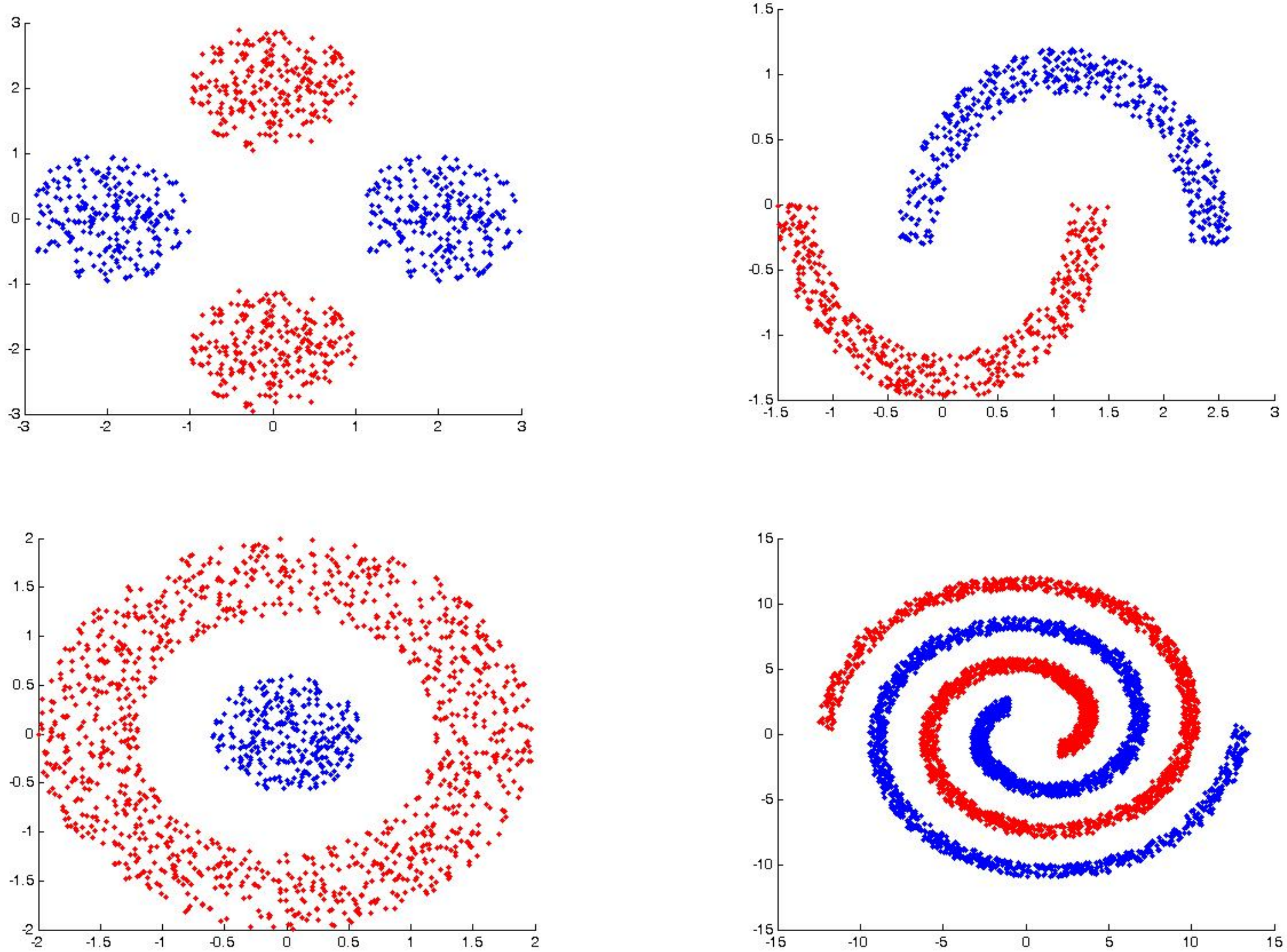
$$g_i(x) = P(w_i|x)$$

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

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

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

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, …, n**
- **Goal: find the k mean vectors $\mu_1, \mu_2, \ldots, \mu_k$**
- **Output**
  - **k x n binary membership matrix U where**

$$\boldsymbol{u}_{ij} = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \in \boldsymbol{G}_i \\ 0 & \text{else} \end{cases}$$

& G_j, j=1, 2, …, 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, c, $\mu_1, \mu_2, …, \mu_c$ (randomly selected)
        do
            1. classify n samples according to nearest $\mu_i$
            2. recompute $\mu_i$
        until no change in $\mu_i$
    return $\mu_1, \mu_2, …, \mu_c$
End
```

Classification Stage

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

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

- 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:

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

Re-computing the Means

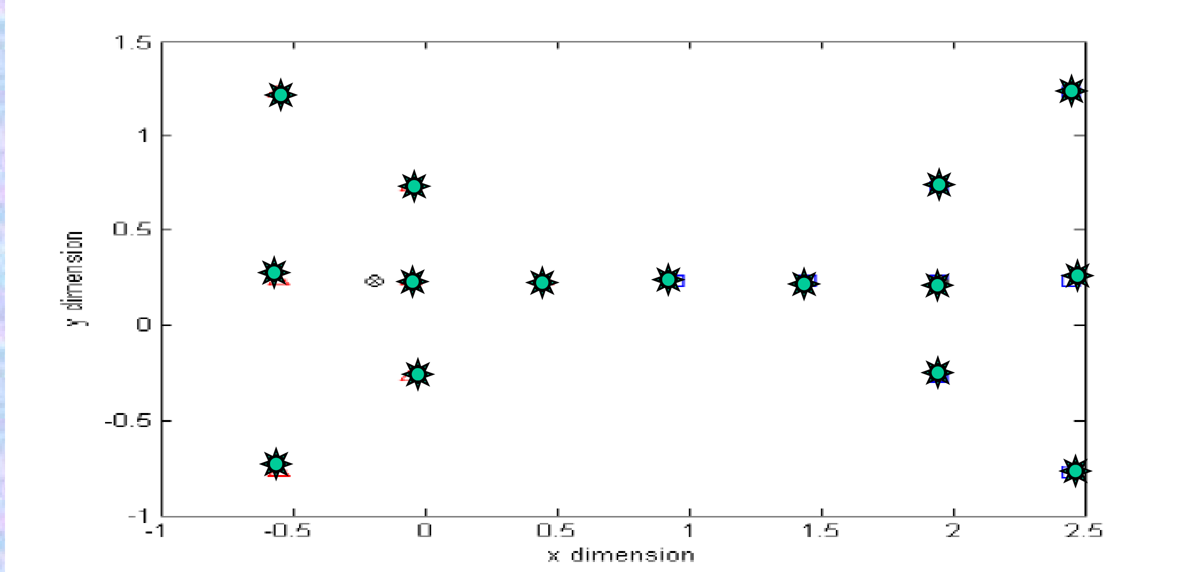
- The means are recomputed according to:

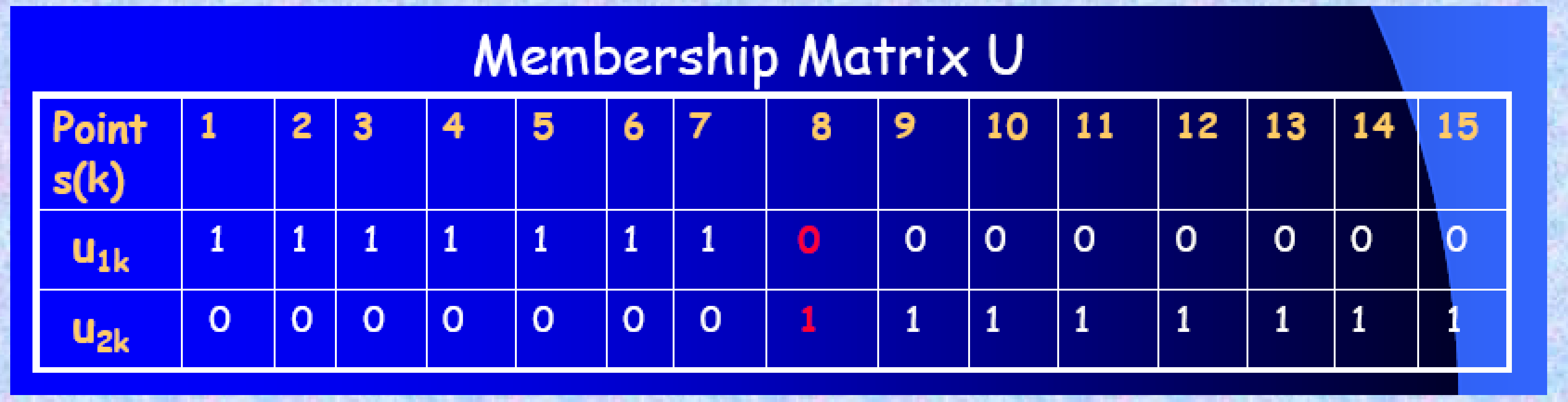
$$\mu_i = \frac{1}{|G_i|}\left(\sum_{k, x_k \in G_i} x_k\right)$$

- 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”.

An Example

- The no of clusters is two in this case.
- But still there is some overlap





Membership Matrix U

Point s(k)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
u_{1k}	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
u_{2k}	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

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

Bivariate Normal Density:

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

μ - Mean; σ - S.D.; ρ_{xy} - Correlation Coefficient

Visualize ρ as equivalent to the orientation of the 2-D 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 kth 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.

<u>Multi-variate Case</u>: $X = [x_1 \;\; x_2 \ldots\ldots x_d]^T$

Mean vector:

$$\mu = E(X) = \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ \mu_d \end{bmatrix}$$

Covariance matrix (symmetric):

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

d-dimensional normal density is:

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

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

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

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

where, s_{ij} is the i-jth component of Σ^{-1} (the inverse of covariance matrix Σ).

Special case, d = 2; where X = (x y)^T; Then:

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$$

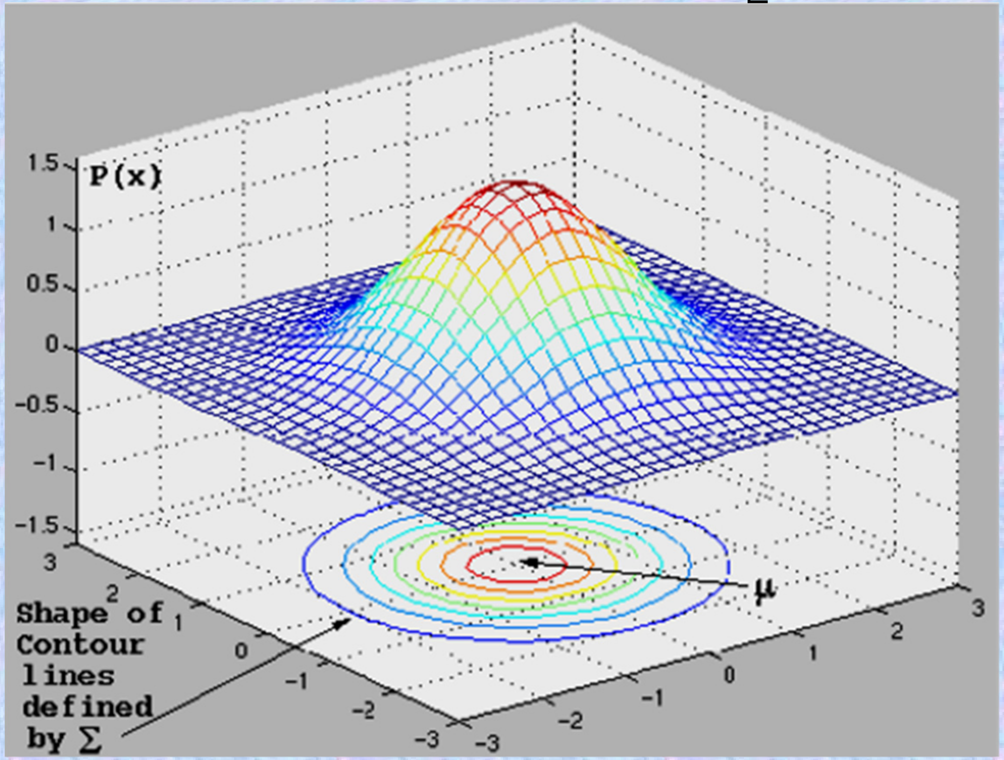
and

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

Can you now obtain this, as given earlier:

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

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



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

$$d(X) =$$

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

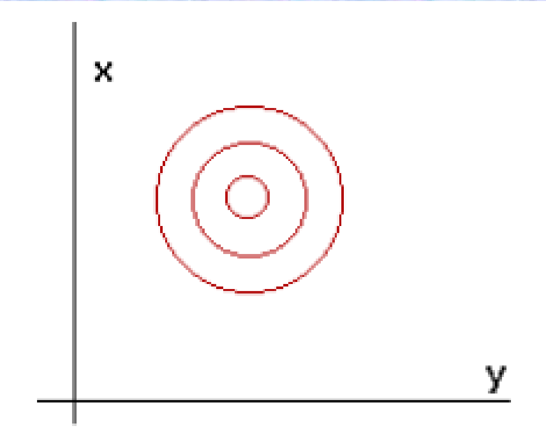
The contours are lines of constant Mahalanobis distance (determined by the matrix Σ), and are quadratic functions.

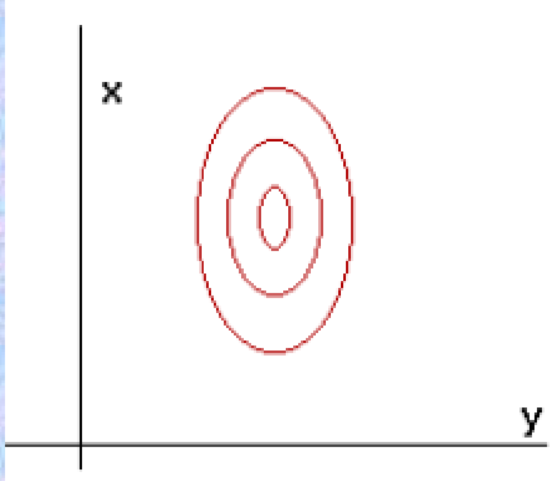
The contours of constant density may also be hyper-ellipsoids (non-diagonal Σ) of constant Mahalanobis distance to μ.

Diagonal covariance;

$$\sigma_x = \sigma_y;$$

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





Diagonal covariance;

$$\sigma_x > \sigma_y;$$

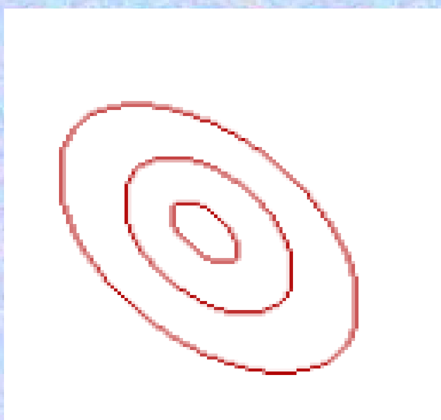
$$\rho_{xy} = 0;$$

Remember, asymmetric and oriented Gaussians

Non-Diagonal covariance;

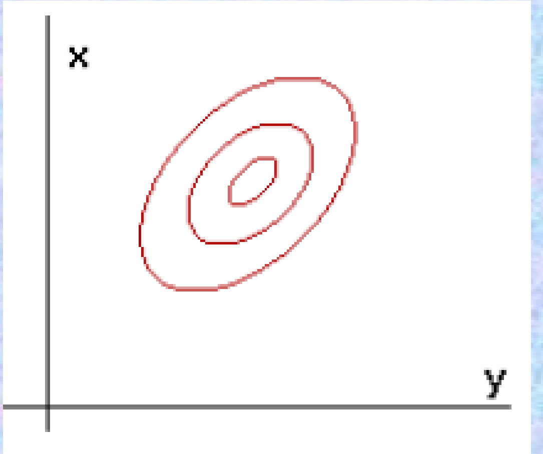
$$\sigma_x = \sigma_y;$$

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



$$\sigma_x = \sigma_y;$$

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



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,…,C.

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

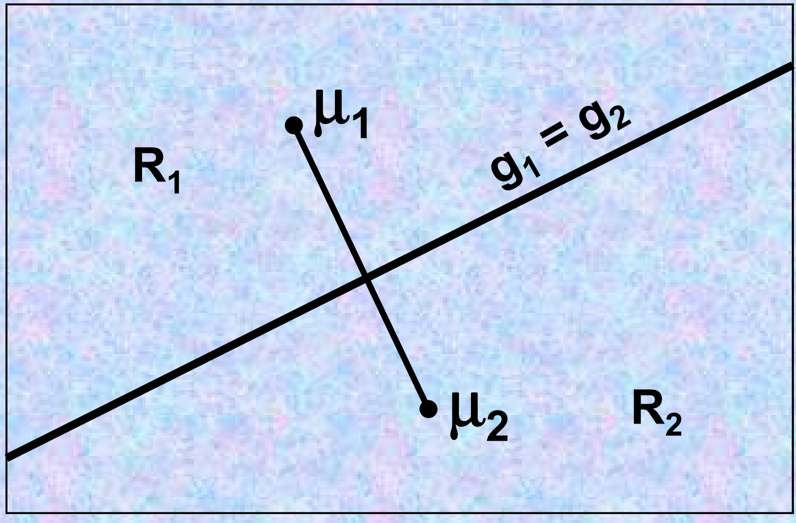
Assign X to class C_m (or region m), where: $g_m(X) > g_i(X), \forall i, i \neq m.$

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:



$$g_1(X) = \left\| \overrightarrow{X} - \overrightarrow{\mu_1} \right\|; \quad g_2(X) = \left\| \overrightarrow{X} - \overrightarrow{\mu_2} \right\|$$

This does not take into account class PDFs and priors.

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

Consider discriminant function as: $$g_i(x) = \ln p(x|w_i) + \ln P(w_i)$$

and class-conditional Prob. as:

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

$g_i(x) =$

Many cases arise, due to the varying nature of Σ:

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

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

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

Remember, multivariate Gaussian density?

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

Define:

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

$$= k.\overrightarrow{d}_i^{\,2} + q$$

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

Let us examine:

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

This quadratic term (scalar) is known as the *Mahalanobis distance* (the distance from X to μ_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 μ_m, for which X is the nearest, for all μ_i. The distance function is then:

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

$$\text{Thus,} \quad G_i(X) = d_i^2 / 2 = (X^T X)/2 - \mu_i^T X + (\mu_i^T \mu_i)/2$$

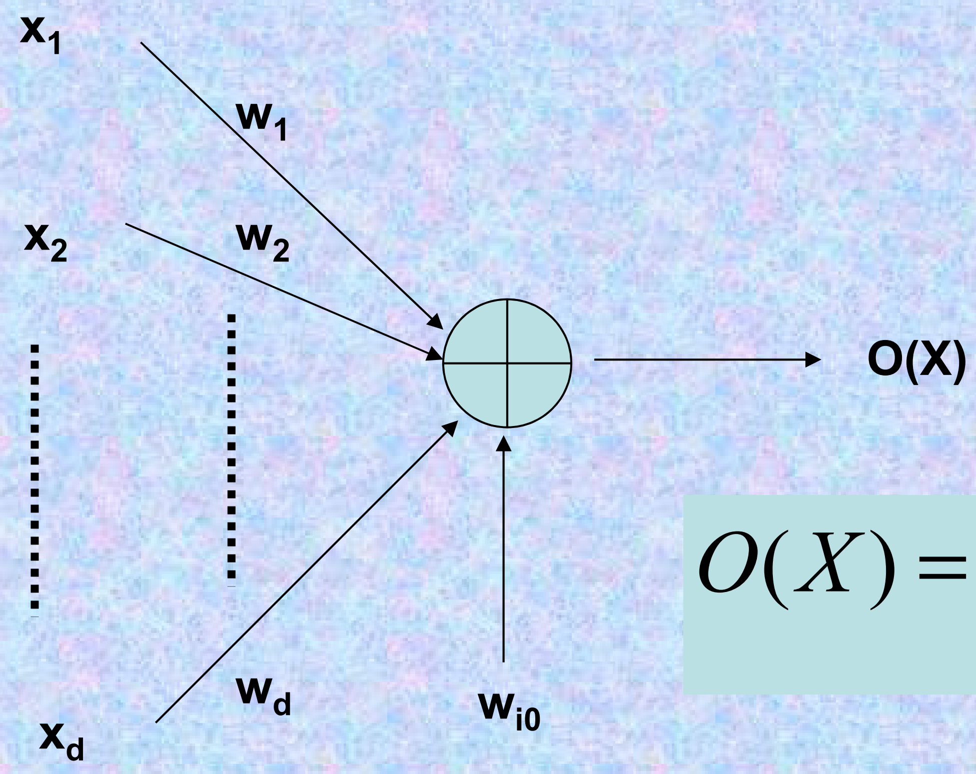
$$= \omega_i^T X + \omega_{i0}$$

Neglecting the class-invariant term.

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

This gives the simplest linear discriminant function or correlation detector.

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



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

View this as (in 2-D space):

$$G = MX - Y + C$$

The decision region boundaries are determined by solving :

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

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

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

Generalized results (Gaussian case) of a discriminant function:

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

$$= -\frac{1}{2}(X-\mu_i)^T \Sigma_i^{-1}(X-\mu_i) - (\frac{d}{2})\log(2\pi) - \frac{1}{2}\log(\sum_i)$$

The mahalanobis distance (quadratic term) spawns a number of different surfaces, depending on Σ^{-1}. It is basically a vector distance using a Σ^{-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:

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

Now, $G_i(\overrightarrow{X}) = \log[P(C_i \mid \overrightarrow{X}).P(\overrightarrow{X})] = \log[P(\overrightarrow{X} \mid C_i)] + \log[\mathrm{P}(C_i)]$

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

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

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

Neglecting the constant term

***<u>Simpler case</u>*: $\Sigma_i = \sigma^2 I$, 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 Σ is a diagonal matrix, with equal/unequal σ_{ii}^2:

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

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 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:

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

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

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

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

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

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 Σ 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, ω_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 ……. .

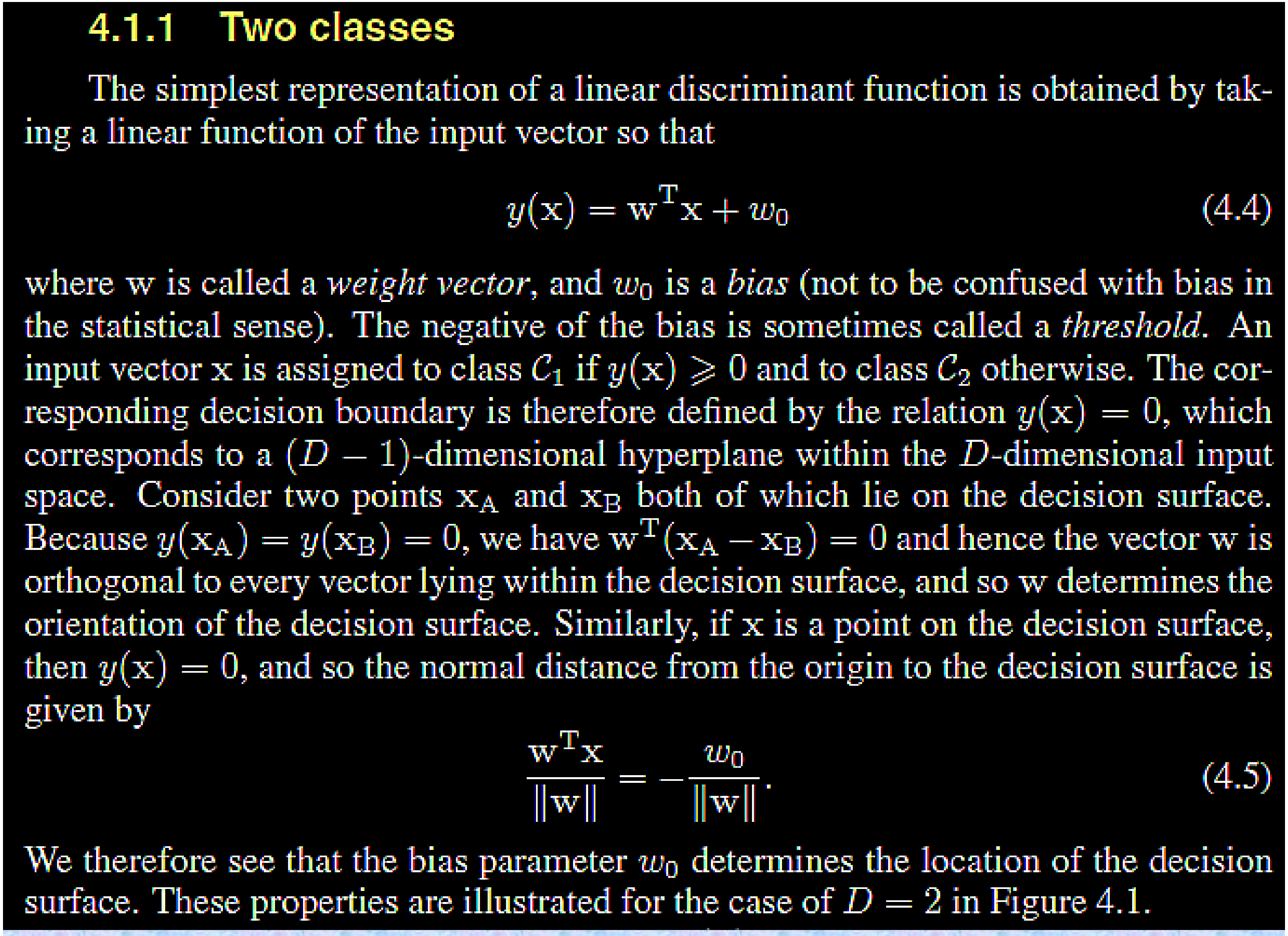
3-D case: $(\omega_1 \omega_2 \omega_3)\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 0$ **is a plane passing through the origin.**

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

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:

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



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

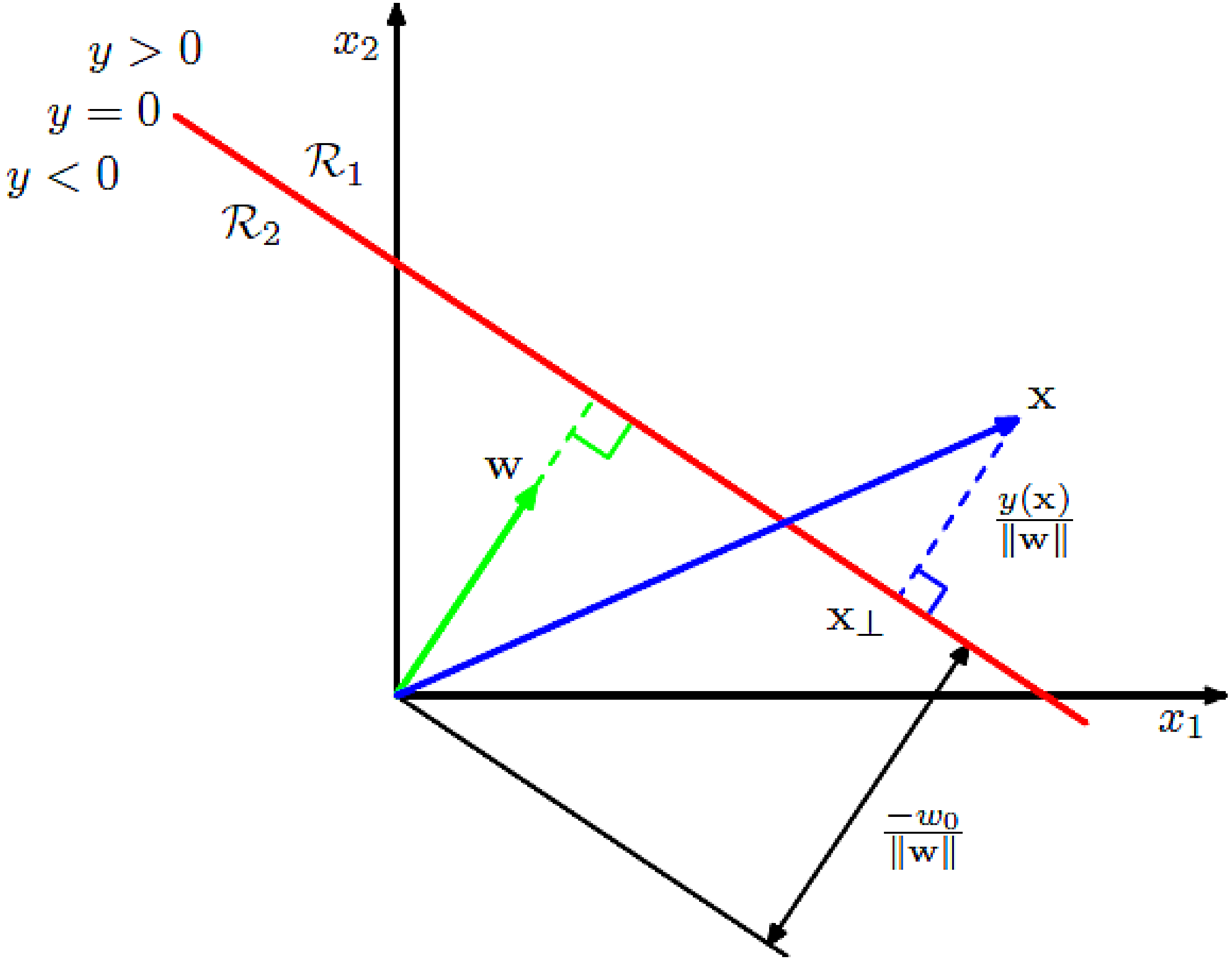
$$y(\mathrm{x}) = \mathrm{w}^{\mathrm{T}}\mathrm{x} + w_0 \tag{4.4}$$

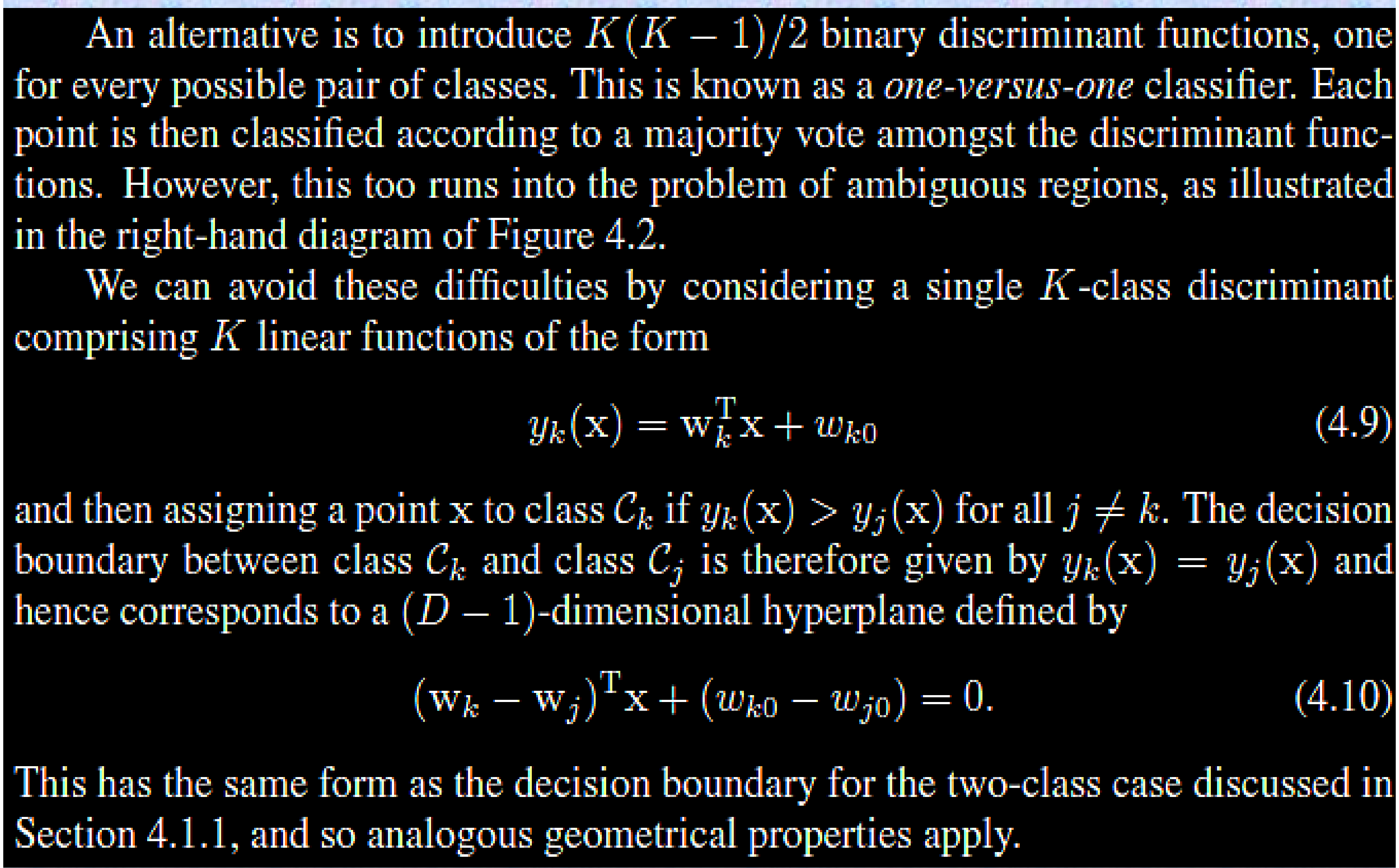
where w is called a *weight vector*, and w_0 is a *bias* (not to be confused with bias in the statistical sense). The negative of the bias is sometimes called a *threshold*. An input vector x is assigned to class $\mathcal{C}_1$ if $y(\mathrm{x}) \geqslant 0$ and to class $\mathcal{C}_2$ otherwise. The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

$$\frac{\mathrm{w}^{\mathrm{T}}\mathrm{x}}{\|\mathrm{w}\|} = -\frac{w_0}{\|\mathrm{w}\|}. \tag{4.5}$$

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

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

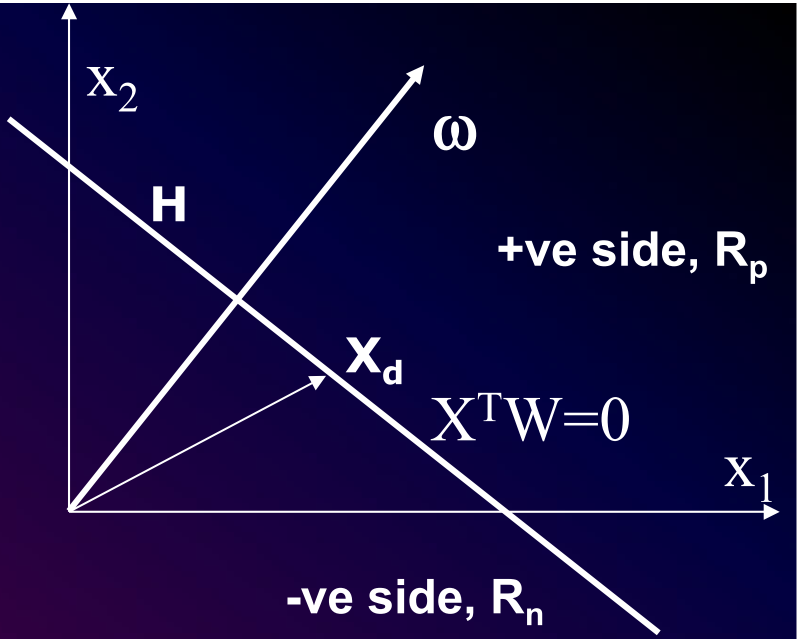
and then assigning a point x to class $\mathcal{C}_k$ if $y_k(\mathrm{x}) > y_j(\mathrm{x})$ for all $j \neq k$. The decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ is therefore given by $y_k(\mathrm{x}) = y_j(\mathrm{x})$ and hence corresponds to a $(D-1)$-dimensional hyperplane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

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

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(X):

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



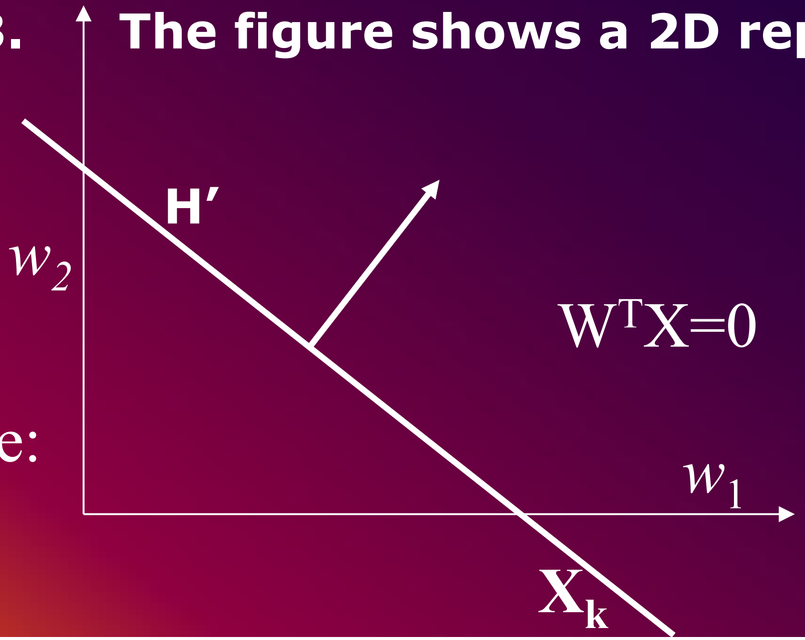
Pattern/feature Space

Orientation of H is determined by ω.

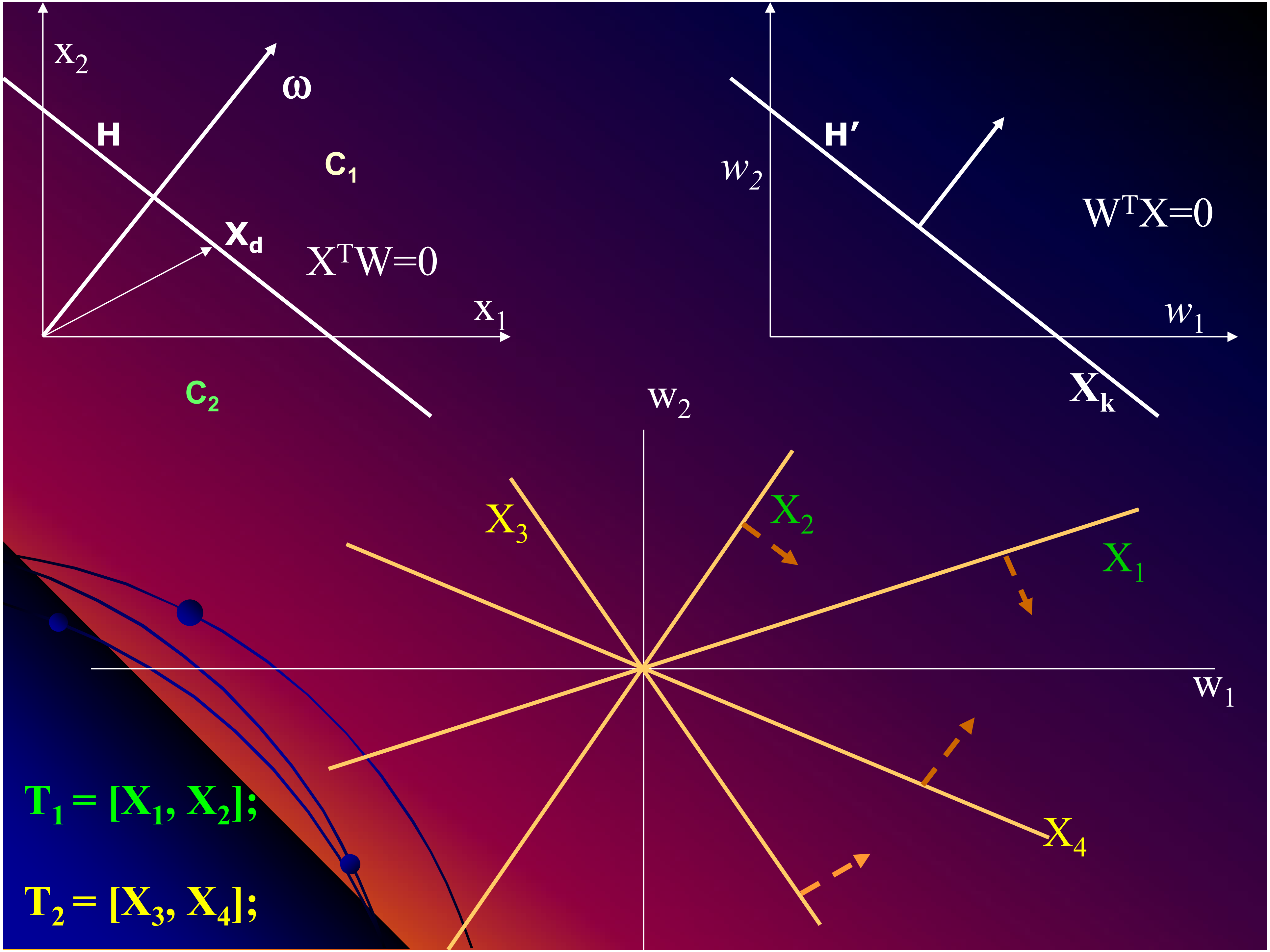
Location of H is determined by d.

H is a hyperplane for d > 3. The figure shows a 2D representation.

The complementary role of a sample in parametric space:

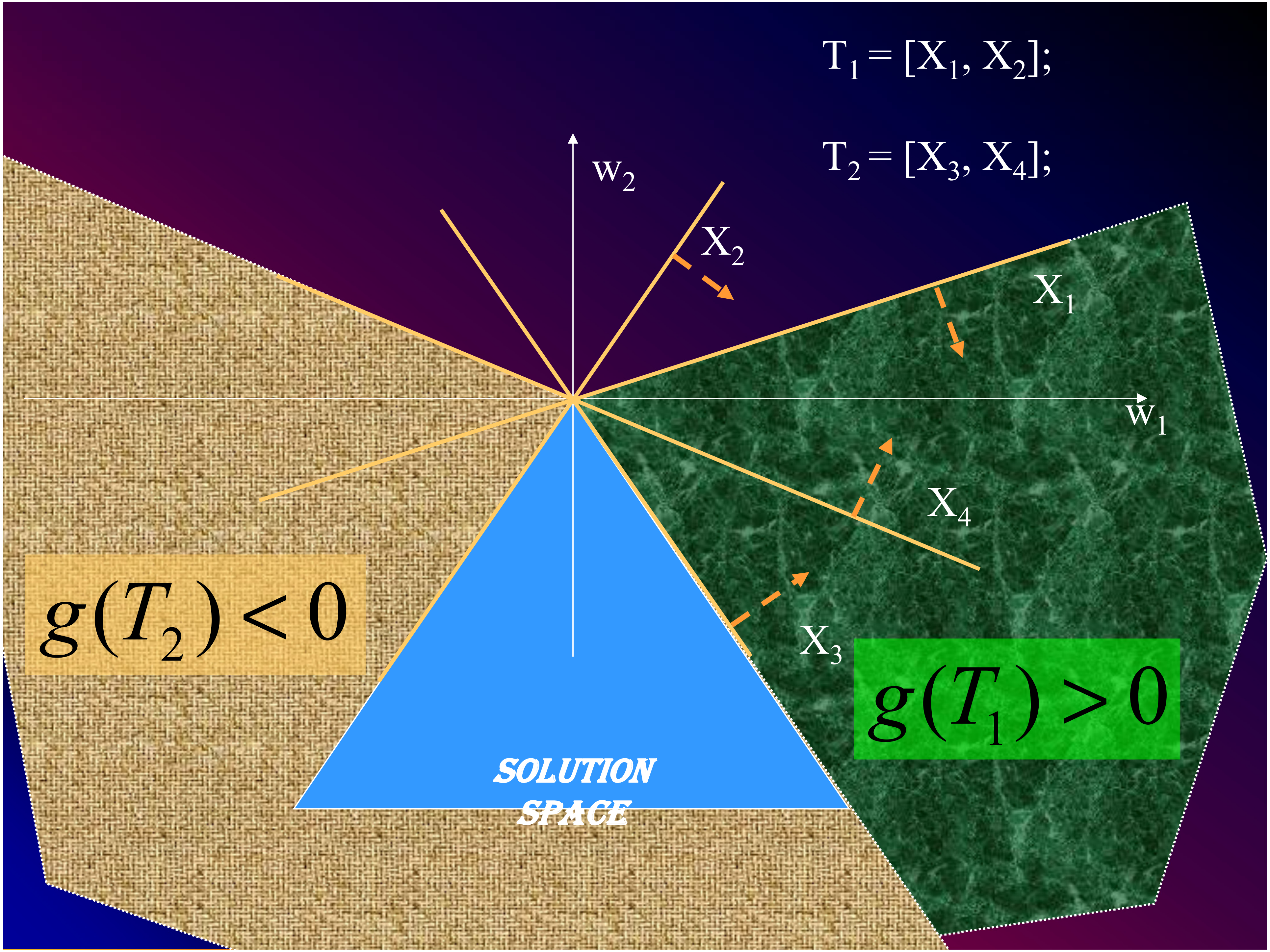


Weight Space



$T_1 = [X_1, X_2];$

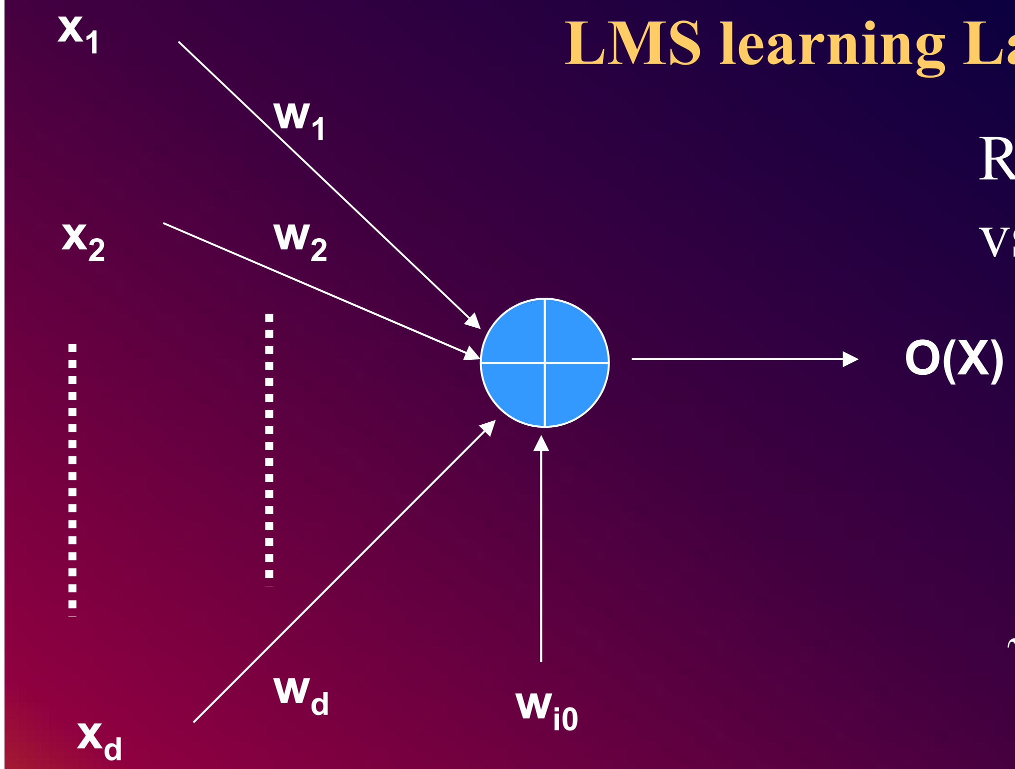
$T_2 = [X_3, X_4];$



$T_1 = [X_1, X_2];$

$T_2 = [X_3, X_4];$

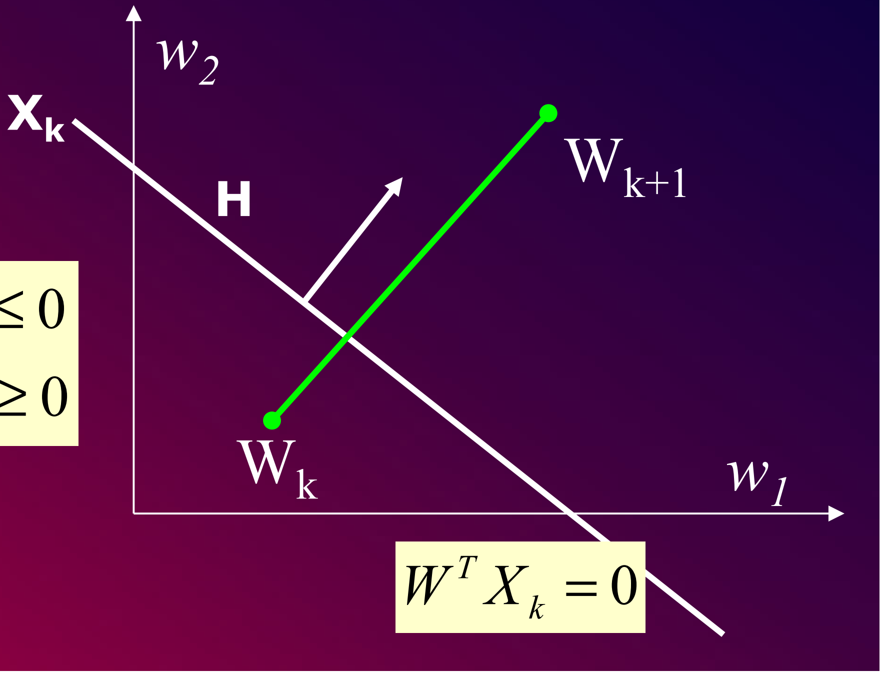
LMS learning Law in BPNN or FFNN models



Read about perceptron
vs. multi-layer feedforward network

$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } \mathrm{X}_k^{\mathrm{T}} W_k \le 0 \\ W_k & \text{if } \mathrm{X}_k^{\mathrm{T}} W_k \ge 0 \end{cases}$$

η_κ is the learning rate parameter

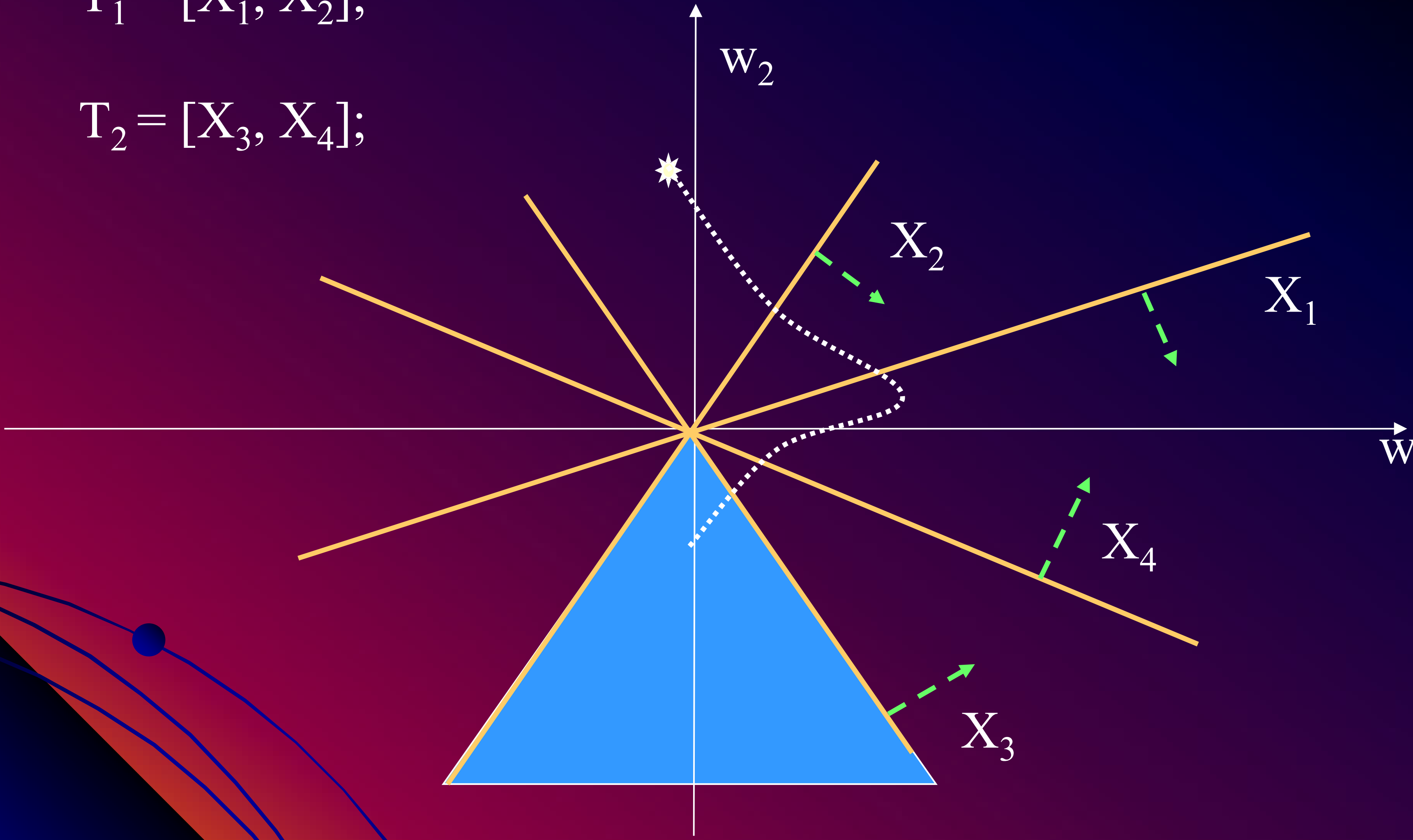


$$W^T X_k = 0$$

$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } \ X_k \in \mathrm{X}_1 \text{ and } X_k^T W_k \le 0 \\ W_k - \eta_k X_k & \text{if } \ X_k \in \mathrm{X}_0 \text{ and } X_k^T W_k \ge 0 \end{cases}$$

$T_1 = [X_1, X_2];$

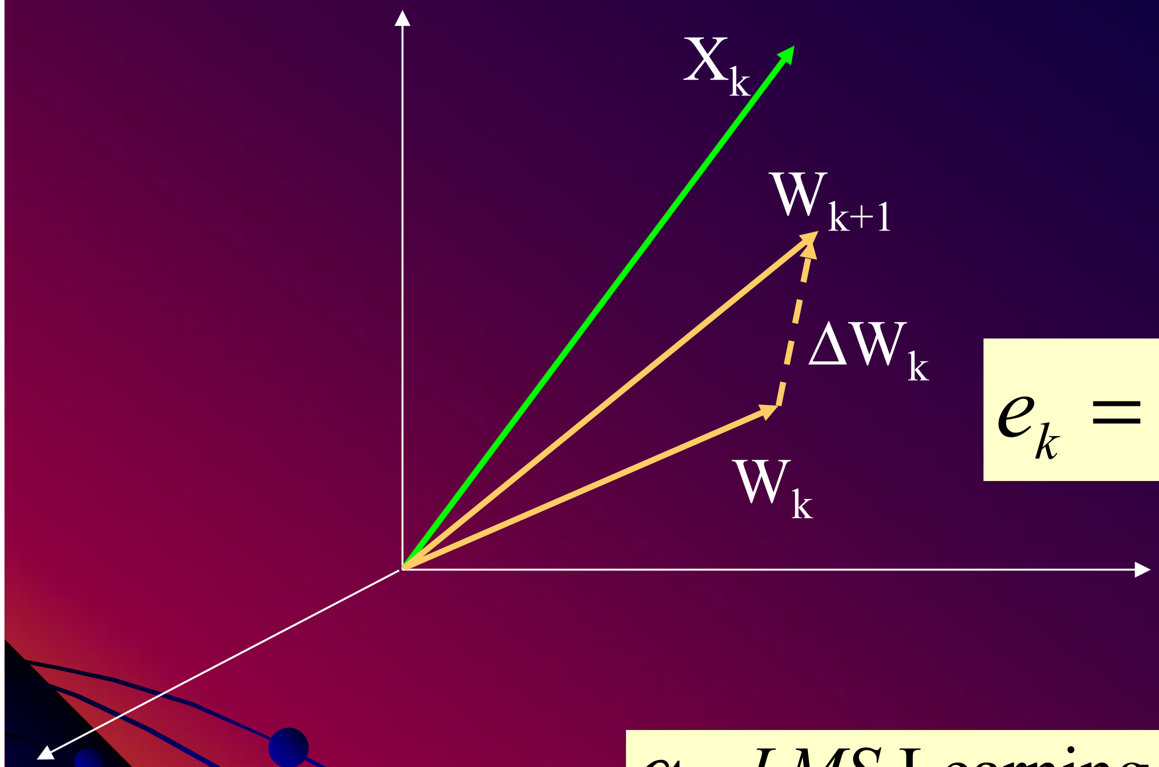
$T_2 = [X_3, X_4];$



η_κ decreases with each iteration

$$W_{k+1} = \begin{cases} W_k + \eta_k X_k & \text{if } X_k \in \mathrm{X}_1 \text{ and } X_k^T W_k \le 0 \\ W_k - \eta_k X_k & \text{if } X_k \in \mathrm{X}_0 \text{ and } X_k^T W_k \ge 0 \end{cases}$$

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:

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

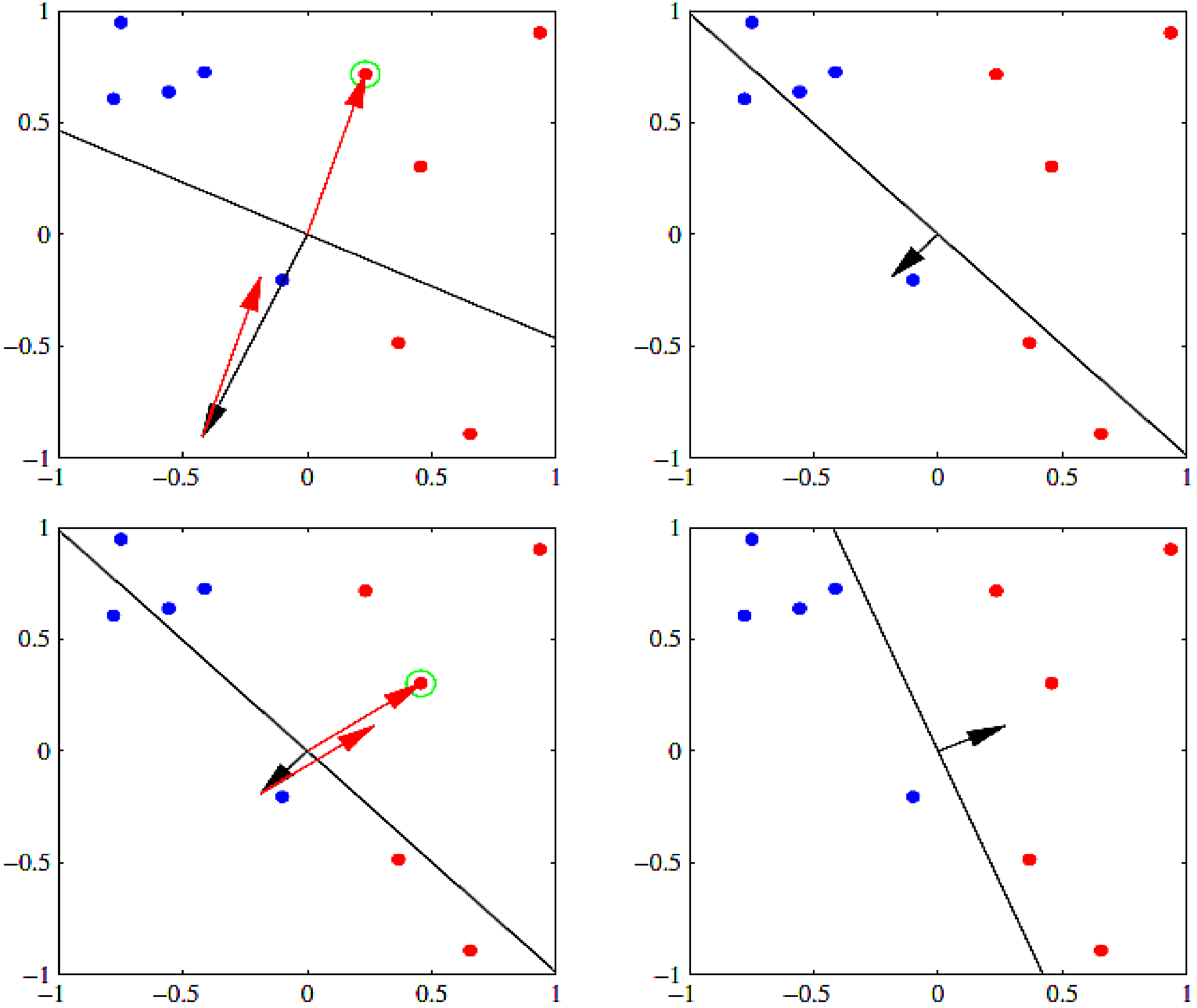


Figure 4.7 Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space (ϕ_1, ϕ_2). The top left plot shows the initial parameter vector $\mathbf{w}$ shown as a black arrow together with the corresponding 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.

Lets look at Bishop chap 5;

Start Sec. 4.1.7, pp 192.

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

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

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

$$R = E[X_k X_k^T] = E[\begin{bmatrix} 1 & x_1^k & & x_n^k \\ x_1^k & x_1^k x_1^k & & x_1^k x_n^k \\ & & & \\ x_n^k & x_n^k x_1^k & & x_n^k x_n^k \end{bmatrix}$$

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

Thus,

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

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

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

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

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

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

CASE A. – Same diagonal Σ, with identical diagonal elements.

Canceling in class-invariant terms:

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

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

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

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

where $\omega_i = \mu_i \big/ \sigma^2$ 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 2nd 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

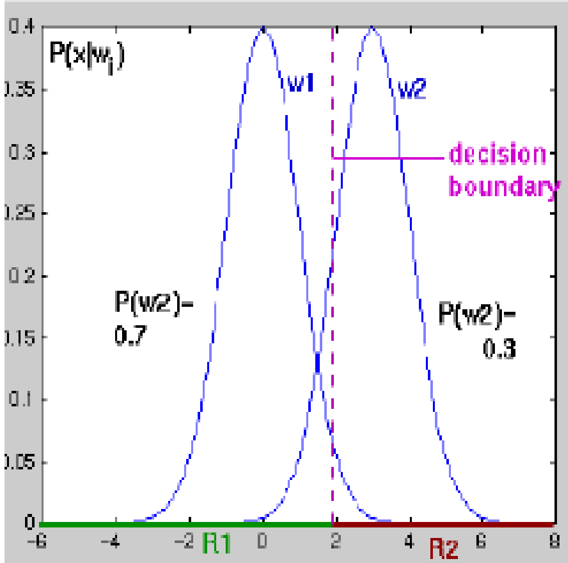
CASE – A. – Same diagonal Σ, with identical diagonal elements (Contd.)

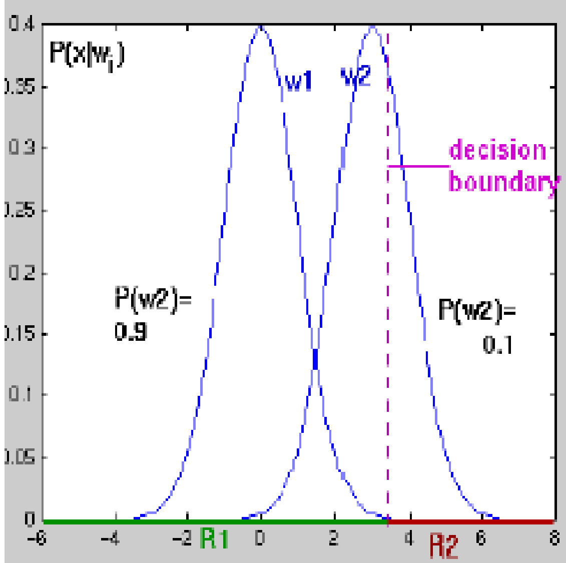
Linear DB:

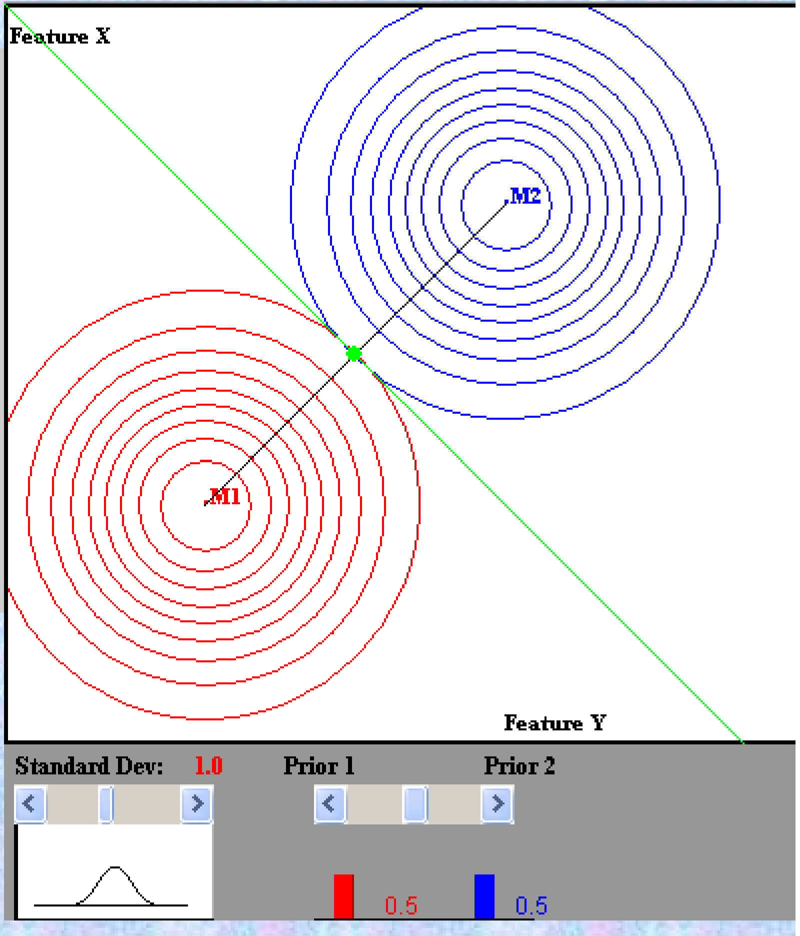
$$W^T(X - X_0) = 0;$$

$$\text{where, } \quad W = \mu_k - \mu_l$$

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



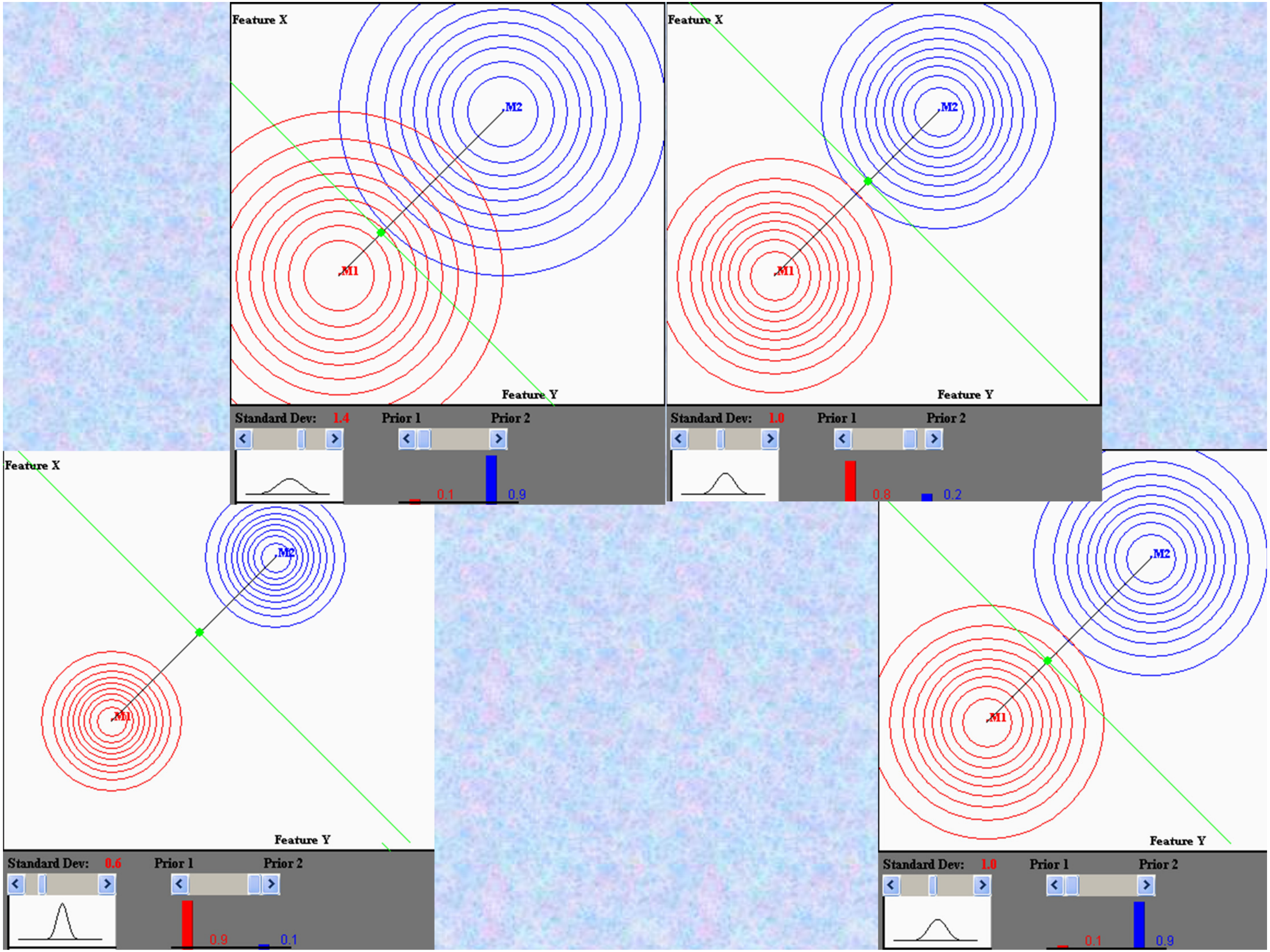




Standard Dev: 1.0

Prior 1 0.5

Prior 2 0.5

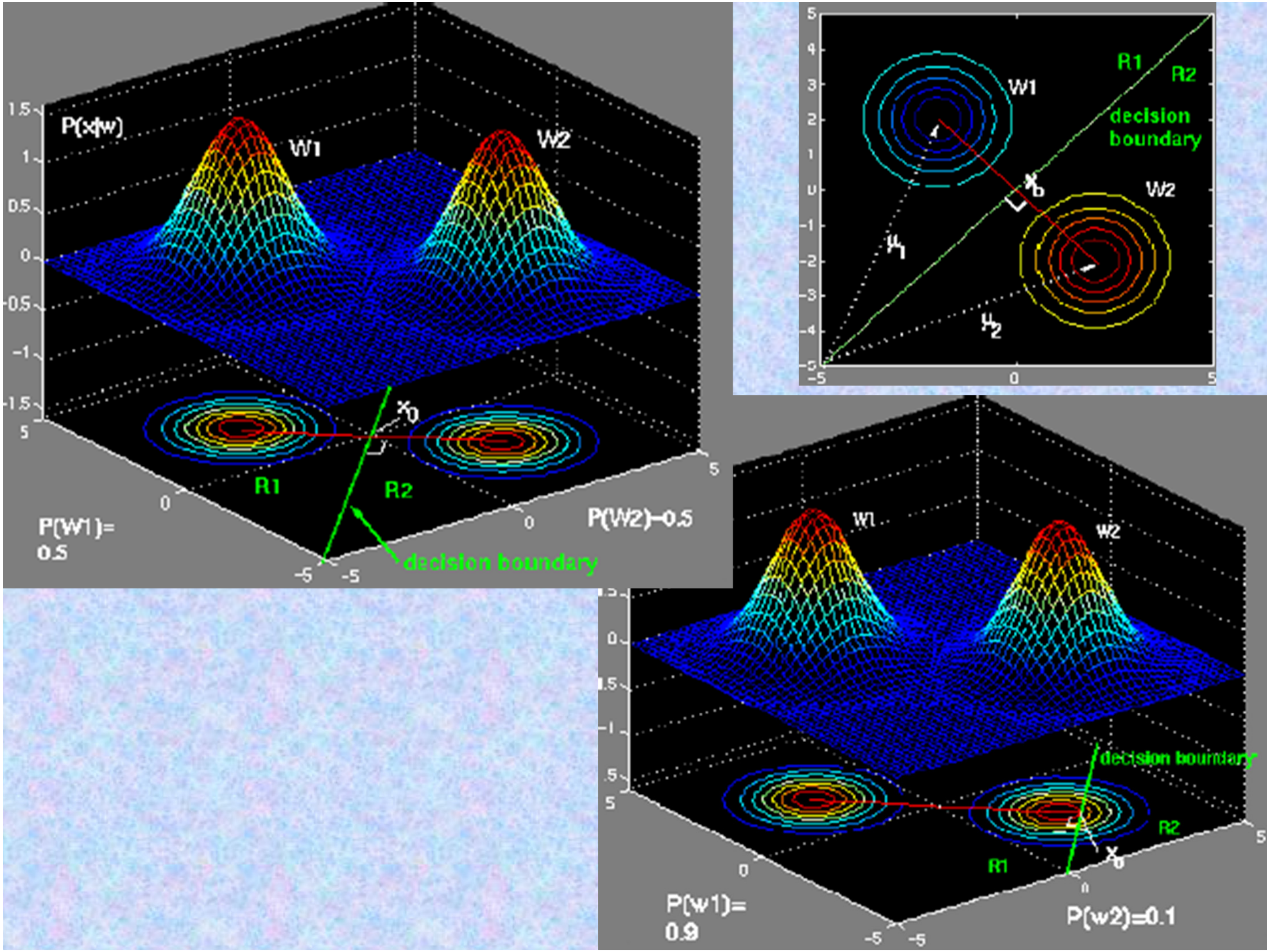


Standard Dev: 1.4 Prior 1 0.1 Prior 2 0.9

Standard Dev: 1.0 Prior 1 0.8 Prior 2 0.2

Standard Dev: 0.6 Prior 1 0.9 Prior 2 0.1

Standard Dev: 1.0 Prior 1 0.1 Prior 2 0.9


P(x|w)
W1
W2
1.5
1
0.5
0
-0.5
-1
-1.5
5
0
-5
R1
R2
x0
P(W1)=
0.5
P(W2)=0.5
decision boundary
W1
W2
R1
R2
decision boundary
x0
μ1
μ2
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5
W1
W2
decision boundary
R1
R2
x0
P(w1)=
0.9
P(w2)=0.1

CASE – B. – Arbitrary Σ, 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

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

← ***Prove it.***

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

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

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

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

The transformation matrix is a symmetric Σ^{-1}**.**

The DB is thus -

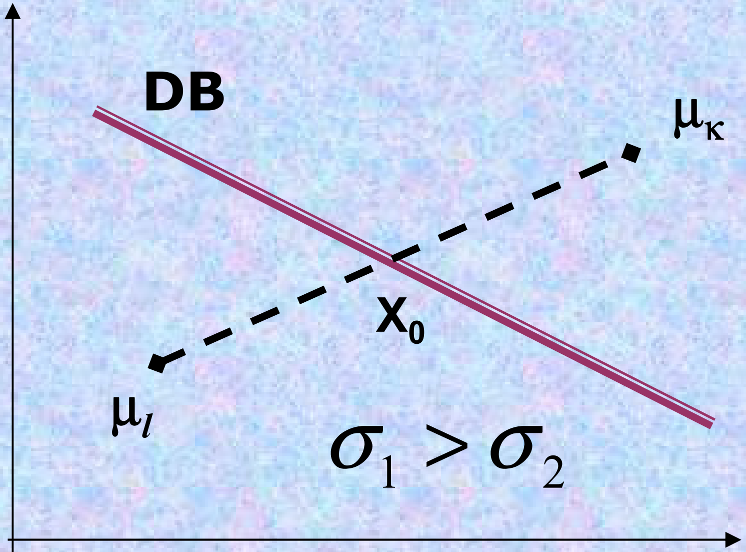
a tilted (rotated) vector joining the two means.

Let Σ (2–D) be diagonal, with non-identical diagonal elements: σ_1 and σ_2

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

$d = 2$ case. $\quad Direction\ of\ \ DB =$

$$\begin{bmatrix} \quad \end{bmatrix}$$



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

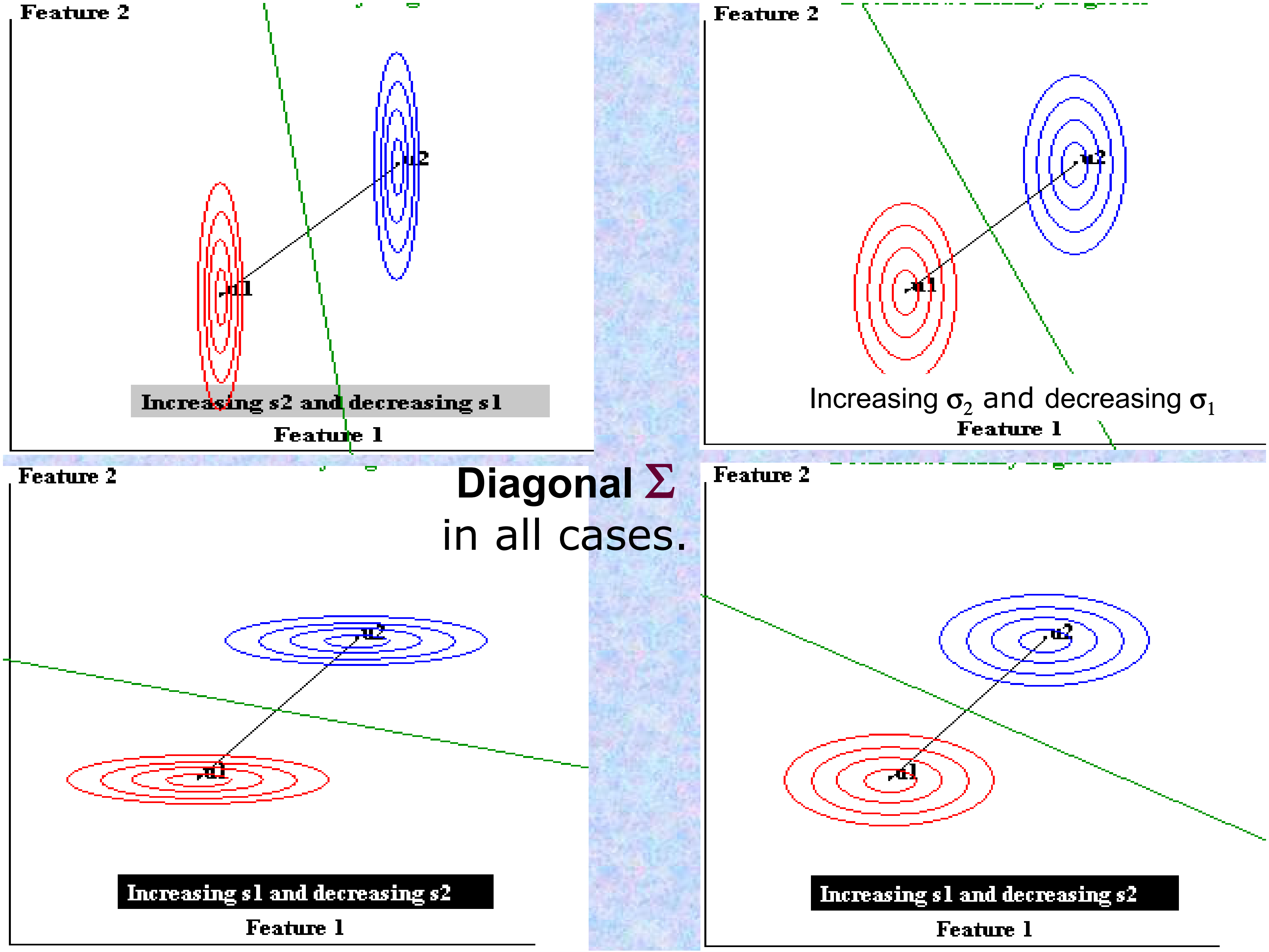
where, $W = \omega_k - \omega_l$ $\quad where\ \omega_i = \Sigma^{-1}\mu_i$

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

Special case:

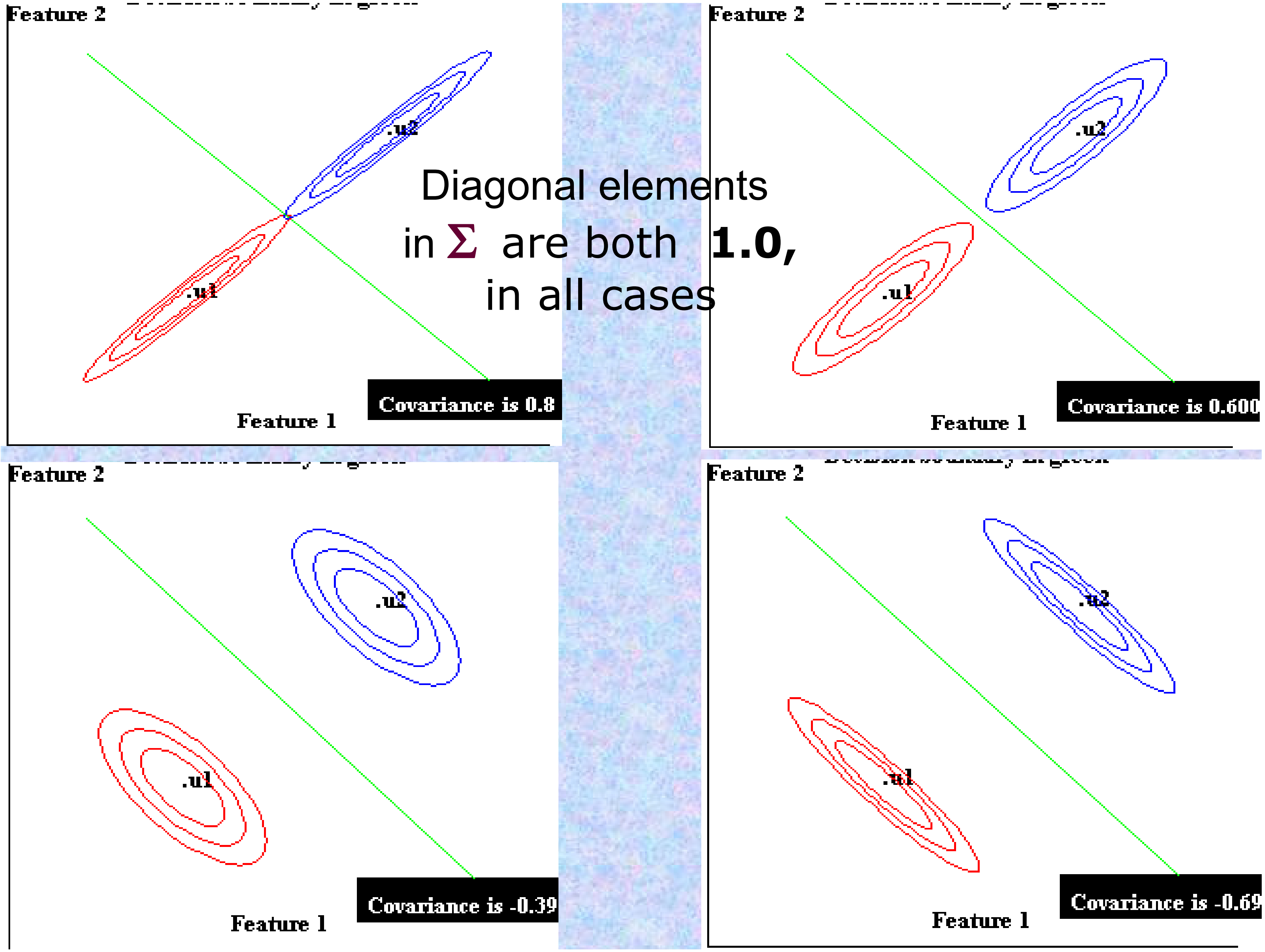
Let, Σ (2–D) be arbitrary, but with diagonal elements (=1).

Solve for **W** in this case, and compare with the diagonal Σ case.

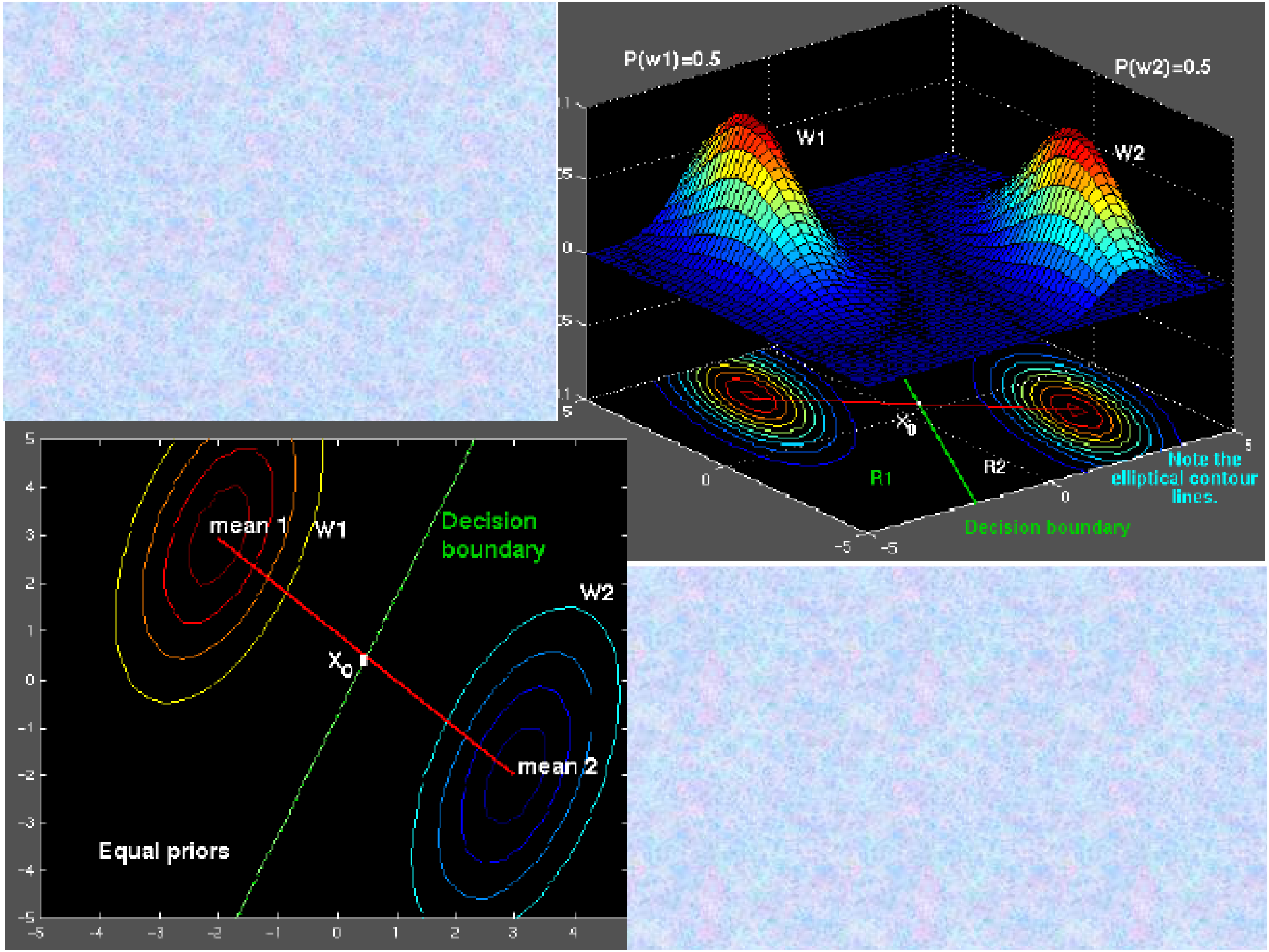


Increasing σ_2 and decreasing σ_1

Diagonal Σ
in all cases.

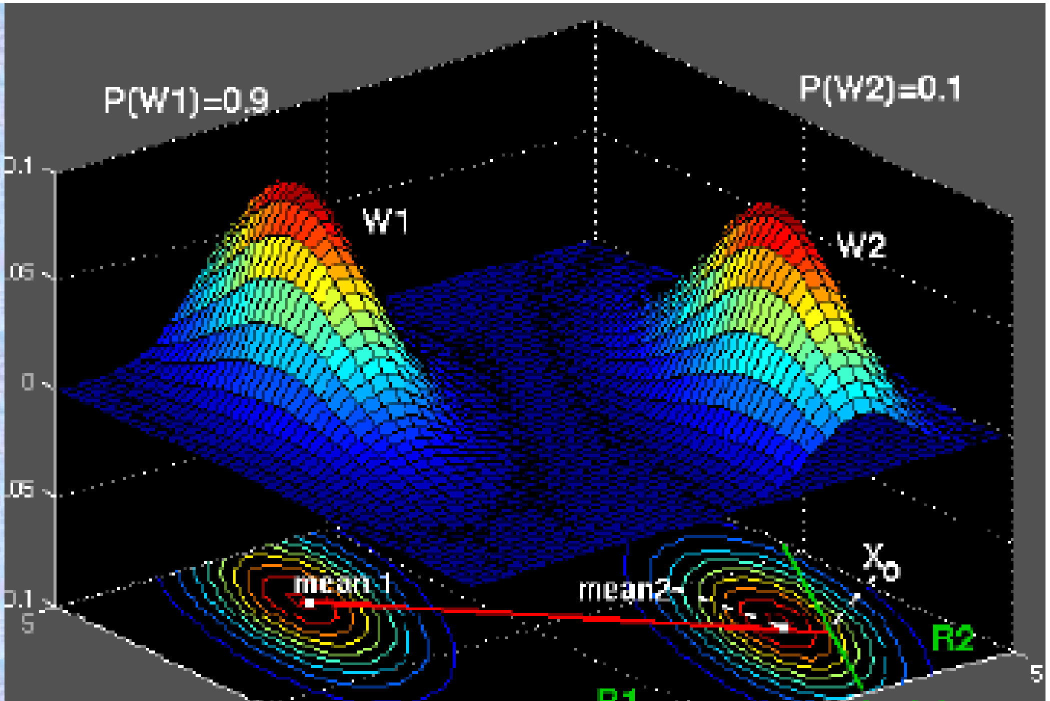


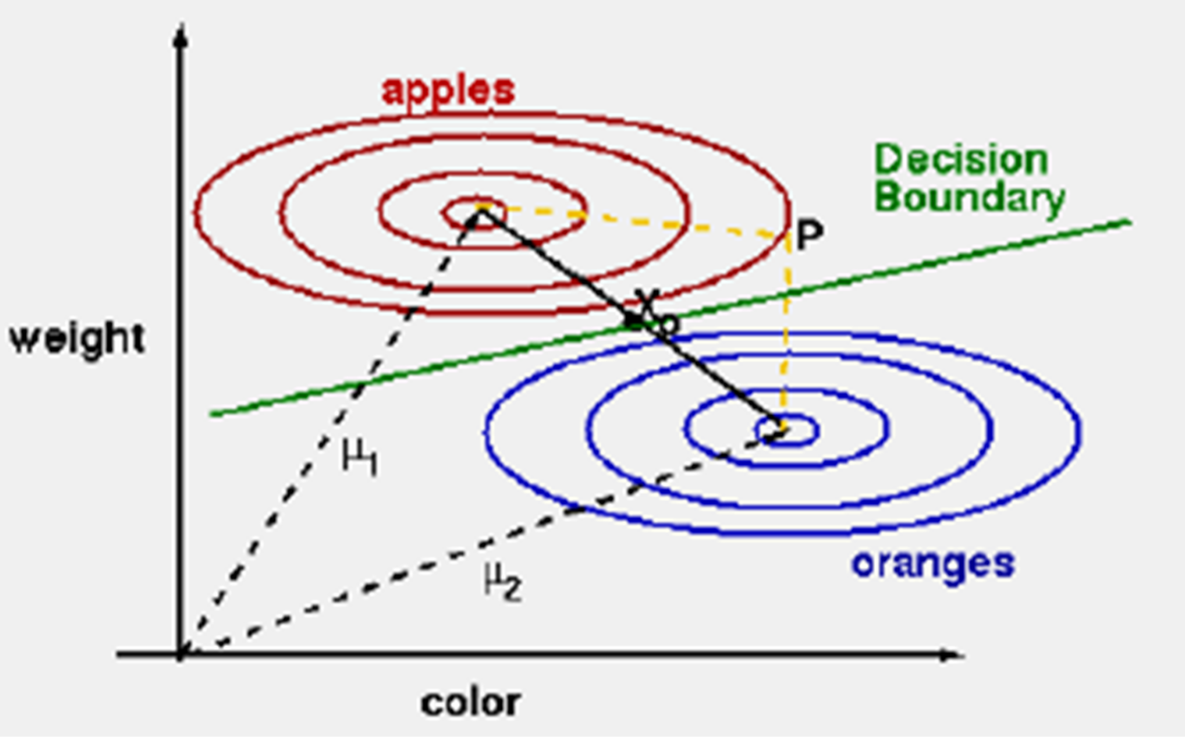
Diagonal elements
in Σ are both **1.0,**
in all cases


P(w1)=0.5
P(w2)=0.5
W1
W2
X0
R1
R2
Decision boundary
Note the elliptical contour lines.
0.1
0.05
0
-0.05
-0.1
5
0
-5
-5
0
5
mean 1
W1
Decision boundary
W2
X0
mean 2
Equal priors
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4

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 Σ, 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 $\mathrm{W_i} = \frac{-1}{2}\Sigma_i^{-1};$

$\omega_i = \Sigma_i^{-1}\mu_i$ 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]:

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

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

Draw and Visualize (qualitatively) the iso-contours

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

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

Assume; P(w$_1$) = P(w$_1$) = 0.5;

Get expression of DB:

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, …….., x_N} taking values in *n*-dimensional image space.**

- **Each image belongs to one of *c* classes {X_1, X_2,.…., 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 –**

$$y_k = W^T x_k \qquad k = 1, 2,……, N$$

where, $W \in R^{nxm}$ 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 μ € *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 *{y₁,y₂,….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.*,**

$$W_{opt} = \arg\max_{W} \left| W^T S_T W \right|$$

where *{w_i | i= 1,2,….,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.

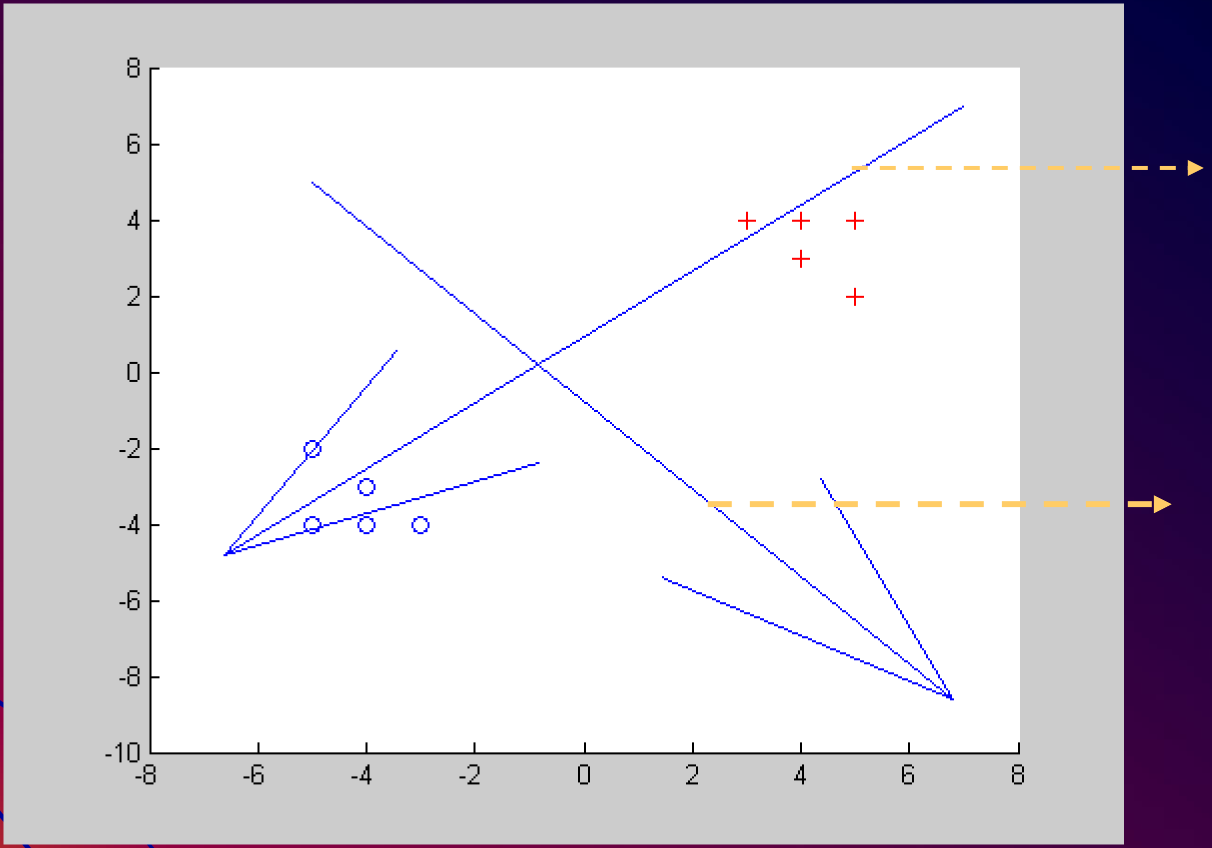
- Any data (say, face) can be reconstructed approximately as a weighted sum of a small collection of images that define a facial basis (eigen images) and a mean image of the face.

- 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

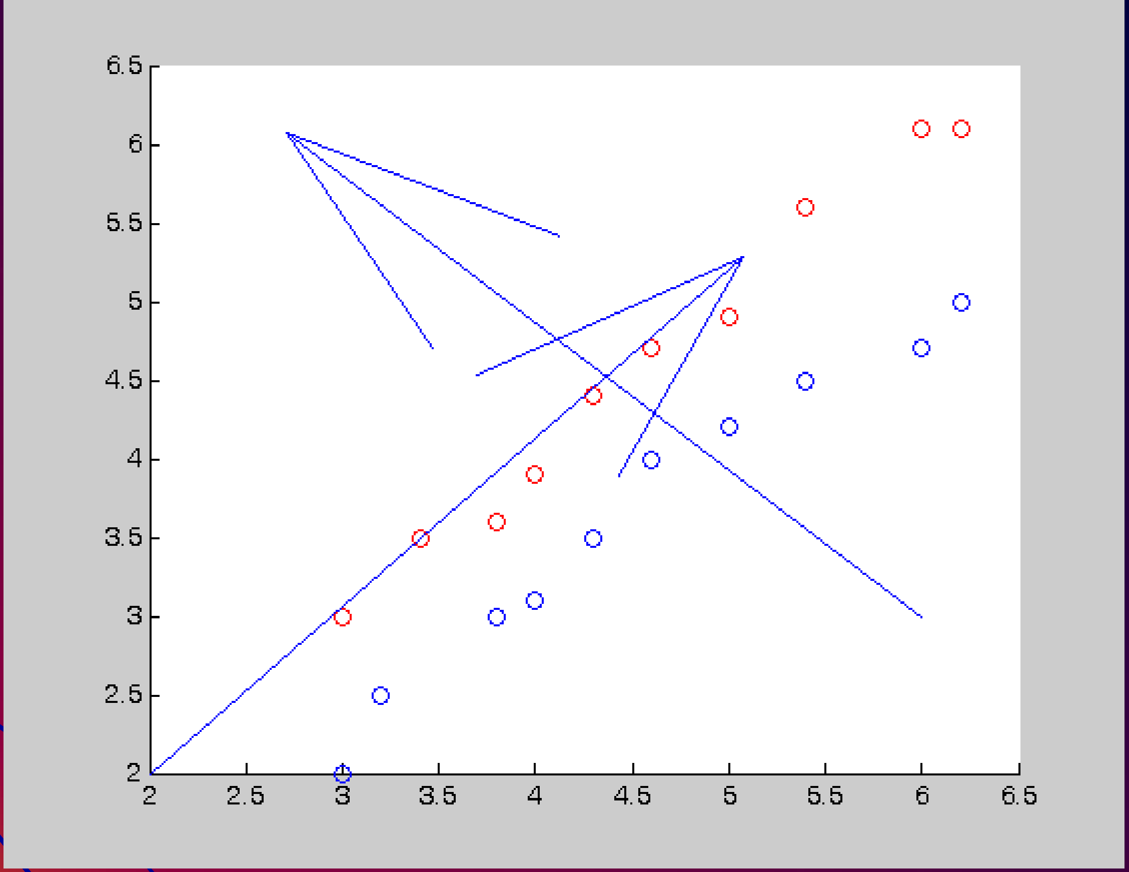
Demonstration of KL Transform



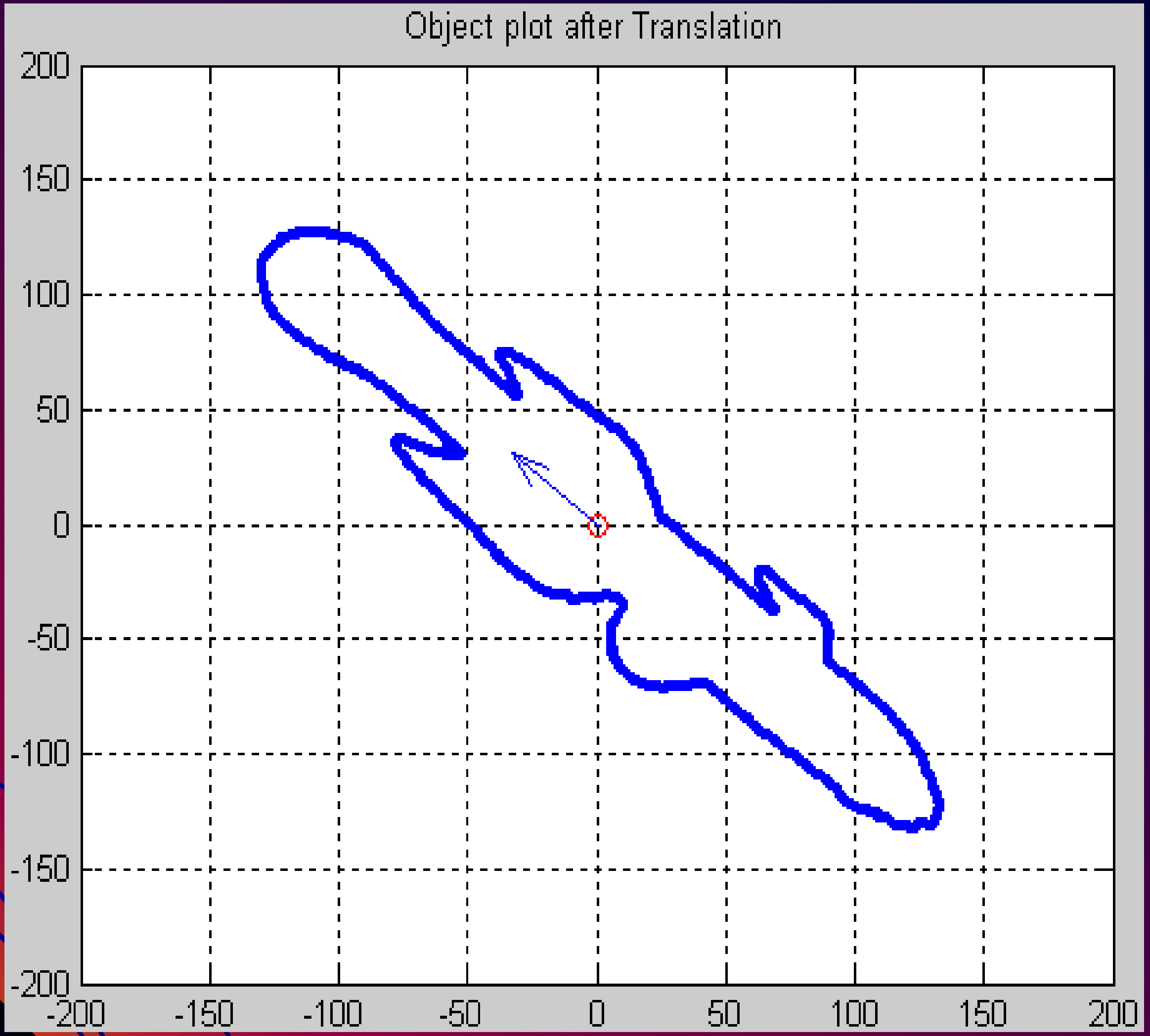
First eigen vector

Second eigen vector

Another One



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

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

The rows of W are the principal components of X, which are also the eigenvectors of 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

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 over K, the matrix Σ 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 *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 Σ 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 Σ is uniquely determined by M (though the matrices U and V are not).

For p = min(m,n) — U is m-by-p, Σ is p-by-p, and V is n-by-p.

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?) =:*

$$COV(X) = XX^T = W\Sigma\Sigma^T W^T = WDW^T$$

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 dxd orthonormal transformation matrix W^T, such that:

$$Y = W^T X$$

with the constraint that:
Cov(Y) is a diagonal matrix, and W^{-1} = W^T.

$$COV(Y) = E[YY^T] = E[(W^T X)(W^T X)^T]$$

$$= E[(W^T X)(X^T W)] = W^T E[XX^T]W$$

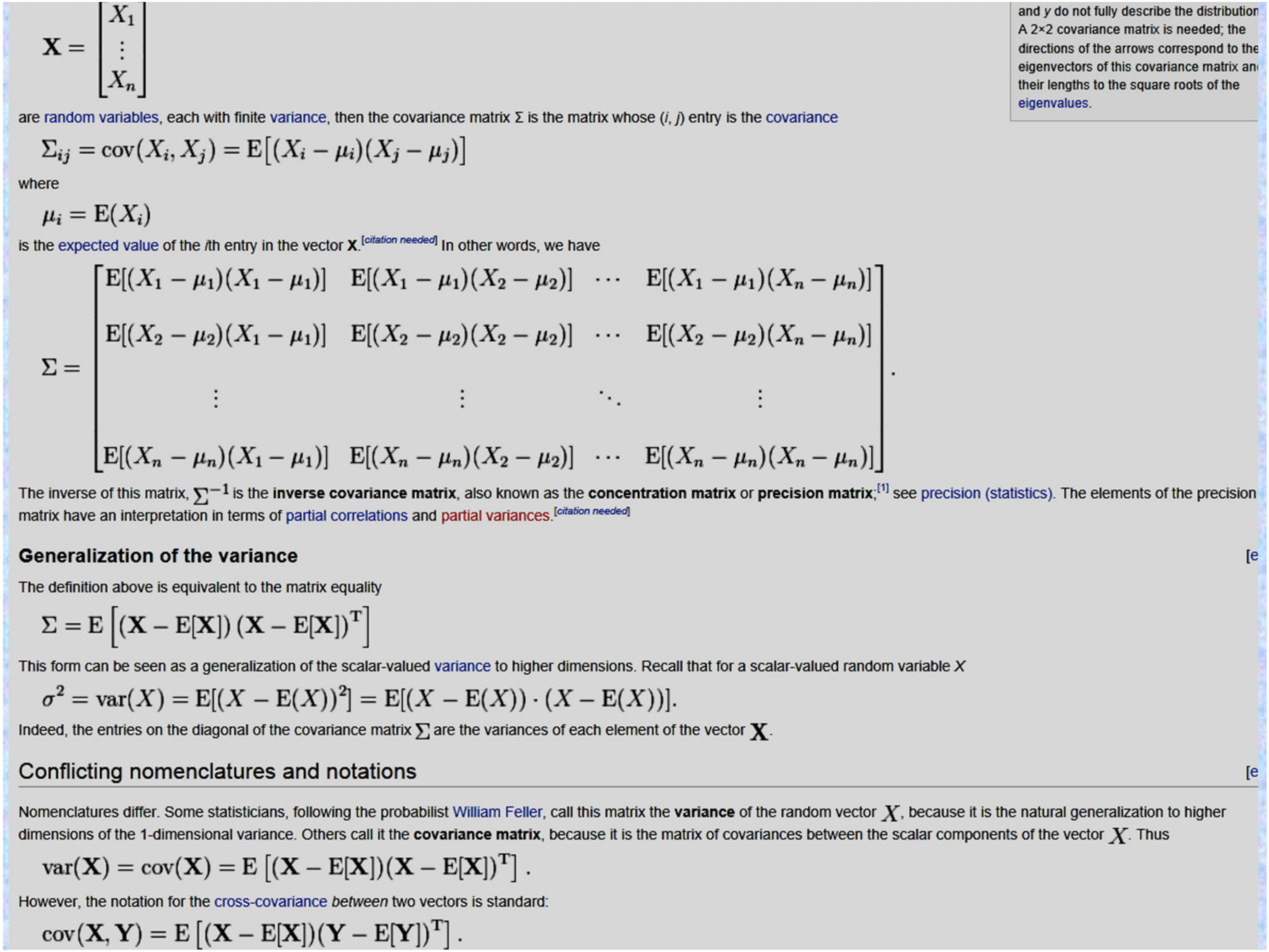
$$= W^T COV(X)W = W^T (WDW^T)W = D$$

$$WCOV(Y) = WW^T COV(X)W = COV(X)W$$

Can you derive from the above, that:

$$[\lambda_1 W_1, \lambda_2 W_2, \ldots\ldots, \lambda_d W_d] =$$

$$[COV(X)W_1, COV(X)W_2, \ldots\ldots, COV(X)W_d]$$



and *y* do not fully describe the distribution A 2×2 covariance matrix is needed; the directions of the arrows correspond to the eigenvectors of this covariance matrix an their lengths to the square roots of the eigenvalues.

$$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

are random variables, each with finite variance, then the covariance matrix Σ is the matrix whose (*i*, *j*) entry is the covariance

$$\Sigma_{ij} = \mathrm{cov}(X_i, X_j) = \mathrm{E}\begin{bmatrix}(X_i - \mu_i)(X_j - \mu_j)\end{bmatrix}$$

where

$$\mu_i = \mathrm{E}(X_i)$$

is the expected value of the *i*th entry in the vector **X**.[citation needed] In other words, we have

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

The inverse of this matrix, Σ^{-1} is the **inverse covariance matrix**, also known as the **concentration matrix** or **precision matrix**;[1] see precision (statistics). The elements of the precision matrix have an interpretation in terms of partial correlations and partial variances.[citation needed]

Generalization of the variance

[e

The definition above is equivalent to the matrix equality

$$\Sigma = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right]$$

This form can be seen as a generalization of the scalar-valued variance to higher dimensions. Recall that for a scalar-valued random variable *X*

$$\sigma^2 = \mathrm{var}(X) = \mathrm{E}[(X - \mathrm{E}(X))^2] = \mathrm{E}[(X - \mathrm{E}(X)) \cdot (X - \mathrm{E}(X))].$$

Indeed, the entries on the diagonal of the covariance matrix Σ are the variances of each element of the vector $\mathbf{X}$.

Conflicting nomenclatures and notations

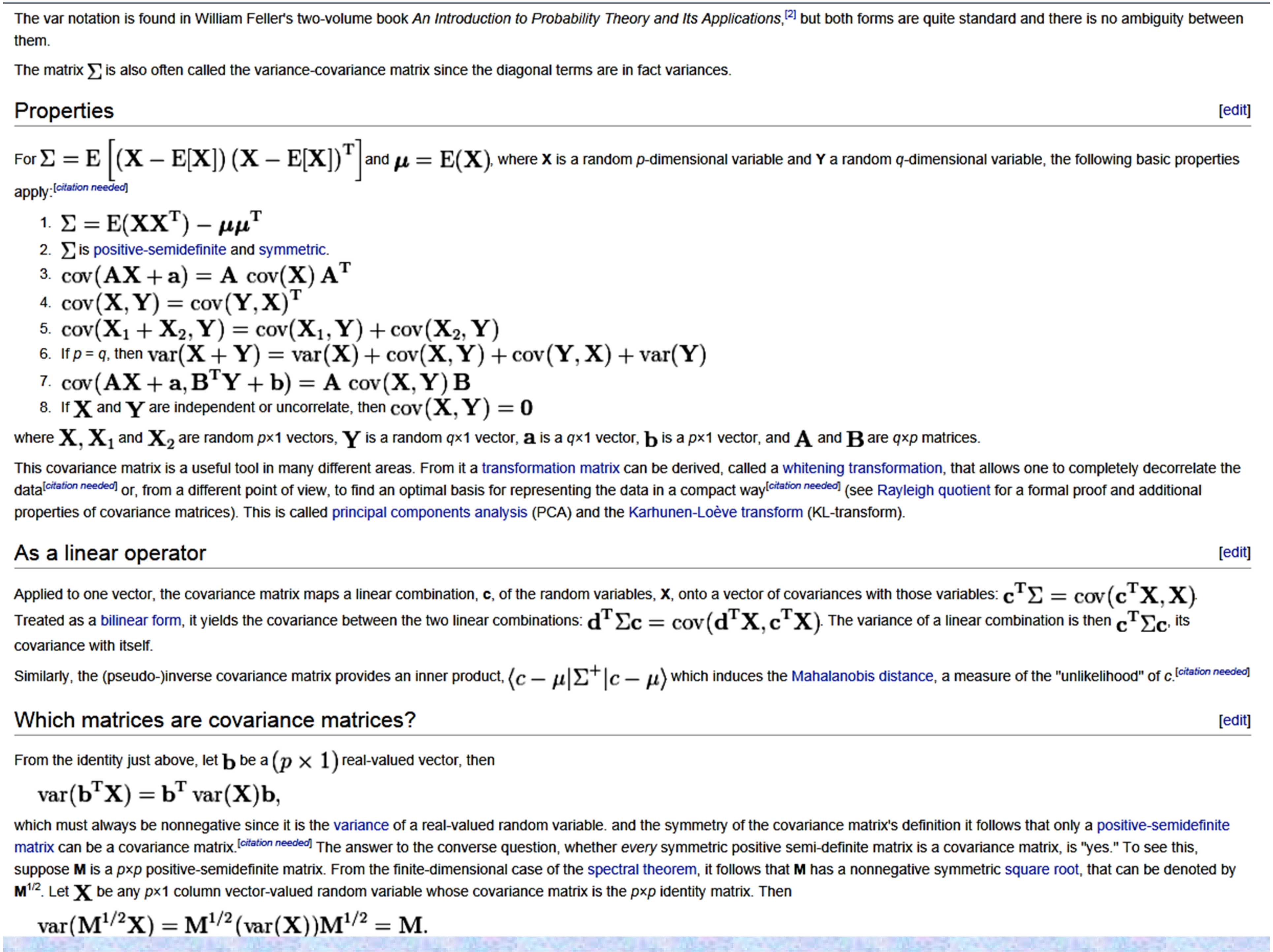
[e

Nomenclatures differ. Some statisticians, following the probabilist William Feller, call this matrix the **variance** of the random vector X, because it is the natural generalization to higher dimensions of the 1-dimensional variance. Others call it the **covariance matrix**, because it is the matrix of covariances between the scalar components of the vector X. Thus

$$\mathrm{var}(\mathbf{X}) = \mathrm{cov}(\mathbf{X}) = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right].$$

However, the notation for the cross-covariance *between* two vectors is standard:

$$\mathrm{cov}(\mathbf{X}, \mathbf{Y}) = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{Y} - \mathrm{E}[\mathbf{Y}])^{\mathrm{T}}\right].$$



The var notation is found in William Feller's two-volume book *An Introduction to Probability Theory and Its Applications*,[2] but both forms are quite standard and there is no ambiguity between them.

The matrix $\sum$ is also often called the variance-covariance matrix since the diagonal terms are in fact variances.

Properties

[edit]

For $\Sigma = \mathrm{E}\left[(\mathbf{X}-\mathrm{E}[\mathbf{X}])(\mathbf{X}-\mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right]$ and $\boldsymbol{\mu} = \mathrm{E}(\mathbf{X})$, where **X** is a random *p*-dimensional variable and **Y** a random *q*-dimensional variable, the following basic properties apply:[citation needed]

1. $\Sigma = \mathrm{E}(\mathbf{X}\mathbf{X}^{\mathrm{T}}) - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$
2. $\sum$ is positive-semidefinite and symmetric.
3. $\operatorname{cov}(\mathbf{AX}+\mathbf{a}) = \mathbf{A}\ \operatorname{cov}(\mathbf{X})\ \mathbf{A}^{\mathbf{T}}$
4. $\operatorname{cov}(\mathbf{X},\mathbf{Y}) = \operatorname{cov}(\mathbf{Y},\mathbf{X})^{\mathbf{T}}$
5. $\operatorname{cov}(\mathbf{X}_1+\mathbf{X}_2,\mathbf{Y}) = \operatorname{cov}(\mathbf{X}_1,\mathbf{Y}) + \operatorname{cov}(\mathbf{X}_2,\mathbf{Y})$
6. If *p* = *q*, then $\operatorname{var}(\mathbf{X}+\mathbf{Y}) = \operatorname{var}(\mathbf{X}) + \operatorname{cov}(\mathbf{X},\mathbf{Y}) + \operatorname{cov}(\mathbf{Y},\mathbf{X}) + \operatorname{var}(\mathbf{Y})$
7. $\operatorname{cov}(\mathbf{AX}+\mathbf{a},\mathbf{B}^{\mathbf{T}}\mathbf{Y}+\mathbf{b}) = \mathbf{A}\ \operatorname{cov}(\mathbf{X},\mathbf{Y})\,\mathbf{B}$
8. If $\mathbf{X}$ and $\mathbf{Y}$ are independent or uncorrelate, then $\operatorname{cov}(\mathbf{X},\mathbf{Y}) = \mathbf{0}$

where $\mathbf{X}, \mathbf{X}_1$ and $\mathbf{X}_2$ are random *p*×1 vectors, $\mathbf{Y}$ is a random *q*×1 vector, $\mathbf{a}$ is a *q*×1 vector, $\mathbf{b}$ is a *p*×1 vector, and $\mathbf{A}$ and $\mathbf{B}$ are *q*×*p* matrices.

This covariance matrix is a useful tool in many different areas. From it a transformation matrix can be derived, called a whitening transformation, that allows one to completely decorrelate the data[citation needed] or, from a different point of view, to find an optimal basis for representing the data in a compact way[citation needed] (see Rayleigh quotient for a formal proof and additional properties of covariance matrices). This is called principal components analysis (PCA) and the Karhunen-Loève transform (KL-transform).

As a linear operator

[edit]

Applied to one vector, the covariance matrix maps a linear combination, **c**, of the random variables, **X**, onto a vector of covariances with those variables: $\mathbf{c}^{\mathbf{T}}\Sigma = \operatorname{cov}(\mathbf{c}^{\mathbf{T}}\mathbf{X},\mathbf{X})$. Treated as a bilinear form, it yields the covariance between the two linear combinations: $\mathbf{d}^{\mathbf{T}}\Sigma\mathbf{c} = \operatorname{cov}(\mathbf{d}^{\mathbf{T}}\mathbf{X},\mathbf{c}^{\mathbf{T}}\mathbf{X})$. The variance of a linear combination is then $\mathbf{c}^{\mathbf{T}}\Sigma\mathbf{c}$, its covariance with itself.

Similarly, the (pseudo-)inverse covariance matrix provides an inner product, $\langle c-\mu|\Sigma^{+}|c-\mu\rangle$ which induces the Mahalanobis distance, a measure of the "unlikelihood" of *c*.[citation needed]

Which matrices are covariance matrices?

[edit]

From the identity just above, let $\mathbf{b}$ be a $(p \times 1)$ real-valued vector, then

$$\operatorname{var}(\mathbf{b}^{\mathbf{T}}\mathbf{X}) = \mathbf{b}^{\mathbf{T}}\operatorname{var}(\mathbf{X})\mathbf{b},$$

which must always be nonnegative since it is the variance of a real-valued random variable. and the symmetry of the covariance matrix's definition it follows that only a positive-semidefinite matrix can be a covariance matrix.[citation needed] The answer to the converse question, whether *every* symmetric positive semi-definite matrix is a covariance matrix, is "yes." To see this, suppose **M** is a *p*×*p* positive-semidefinite matrix. From the finite-dimensional case of the spectral theorem, it follows that **M** has a nonnegative symmetric square root, that can be denoted by $\mathbf{M}^{1/2}$. Let $\mathbf{X}$ be any *p*×1 column vector-valued random variable whose covariance matrix is the *p*×*p* identity matrix. Then

$$\operatorname{var}(\mathbf{M}^{1/2}\mathbf{X}) = \mathbf{M}^{1/2}(\operatorname{var}(\mathbf{X}))\mathbf{M}^{1/2} = \mathbf{M}.$$

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: $\mu_x = \begin{bmatrix} 1/3 \\ 4/3 \\ 2 \end{bmatrix};$

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

<u>Method – 1</u> (easiest)

$$\tilde{X} = \begin{bmatrix} -4/3 & -7/3 & 11/3 \\ -1/3 & 5/3 & -4/3 \\ 0 & -1 & 1 \end{bmatrix};$$

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.)

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

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

```
C1 =
    1.7778    0.4444         0
    0.4444    0.1111         0
         0         0         0
```

```
C2 =
    5.4444   -3.8889    2.3333
   -3.8889    2.7778   -1.6667
    2.3333   -1.6667    1.0000
```

```
C3 =
   13.4444   -4.8889    3.6667
   -4.8889    1.7778   -1.3333
    3.6667   -1.3333    1.0000
```

```
SigmaC =
   20.6667   -8.3333    6.0000
   -8.3333    4.6667   -3.0000
    6.0000   -3.0000    2.0000
```

```
COVAR =
SigmaC/2 =

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

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 Xavg 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^TX of dimension N*N is considered for eigen-analysis. Will it result in the same, after SVD? Lets check:

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

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

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 <> 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 =
    -3    -2    -1     4     5     6
    -3    -2    -1     4     5     7
```

Mean of the samples:

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

```
XM=
  -4.5000   -3.5000   -2.5000    2.5000   3.5000  4.5000
  -4.6667   -3.6667   -2.6667    2.3333   3.3333  5.3333
```

COVAR(X) = XM * XM^T

```
=      77.5000   82.0000
       82.0000   87.3333
```

XM^T * XM =

```
  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
```

COVAR(X) = XM * XM^T

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

U =

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

S =

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

V =

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

XM^T * XM =

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

U =

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

S =

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

V = U ??

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…..μ 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..

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

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

$$J_3 = tr(S_1) - \mu(trS_2 - c)$$

$$J_4 = \frac{trS_1}{trS_2}$$

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-

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

μ_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

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

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

$\{w_i \mid i = 1,2,\ldots\ldots,m\}$ 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*.

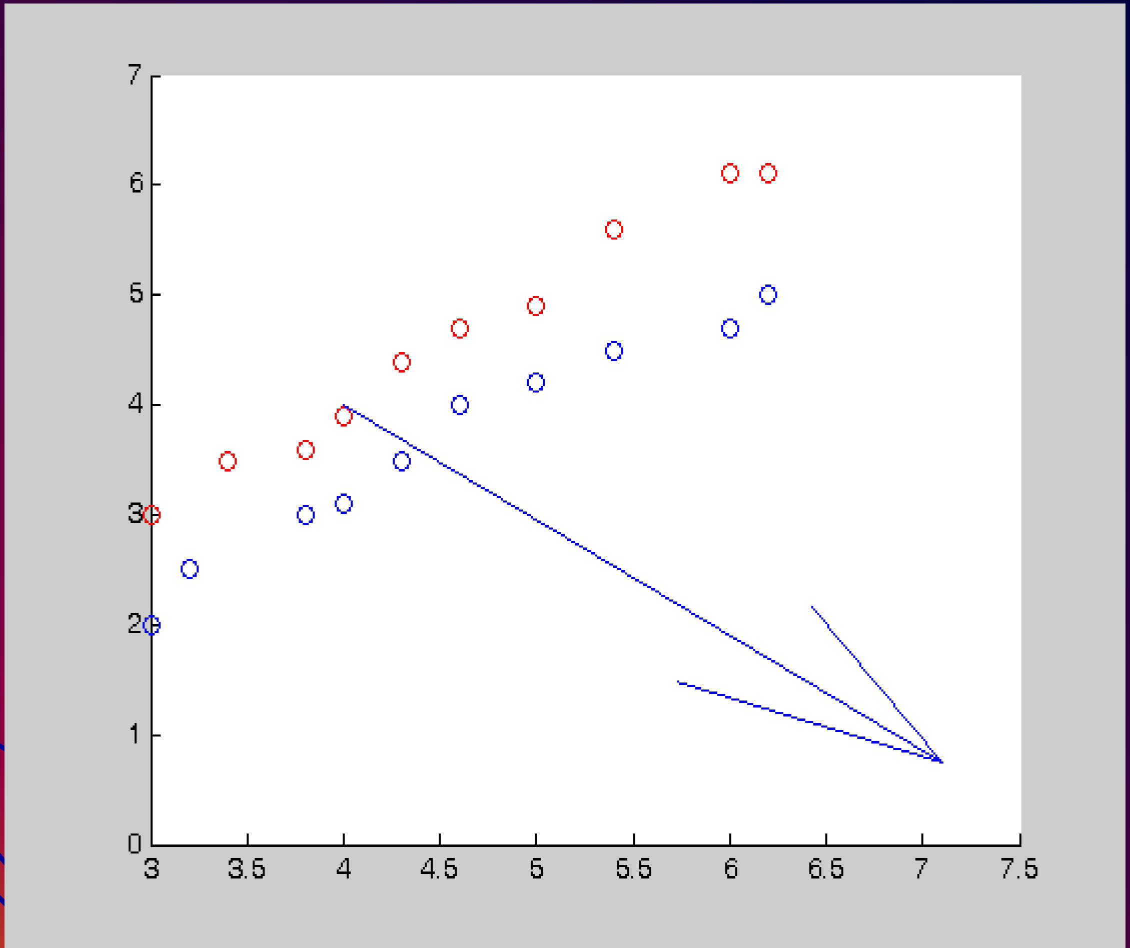
$\mathrm{W}_{\mathrm{opt}}$ is given by

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

$$W_{pca} = \underset{\mathrm{W}}{\arg\max} \left| W^T S_T W \right|$$

$$W_{fld} = \underset{\mathrm{W}}{\arg\max} \frac{\left| W^T W_{pca}^T S_B W_{pca} W \right|}{\left| W^T W_{pca}^T S_W W_{pca} W \right|}$$

Demonstration for LDA



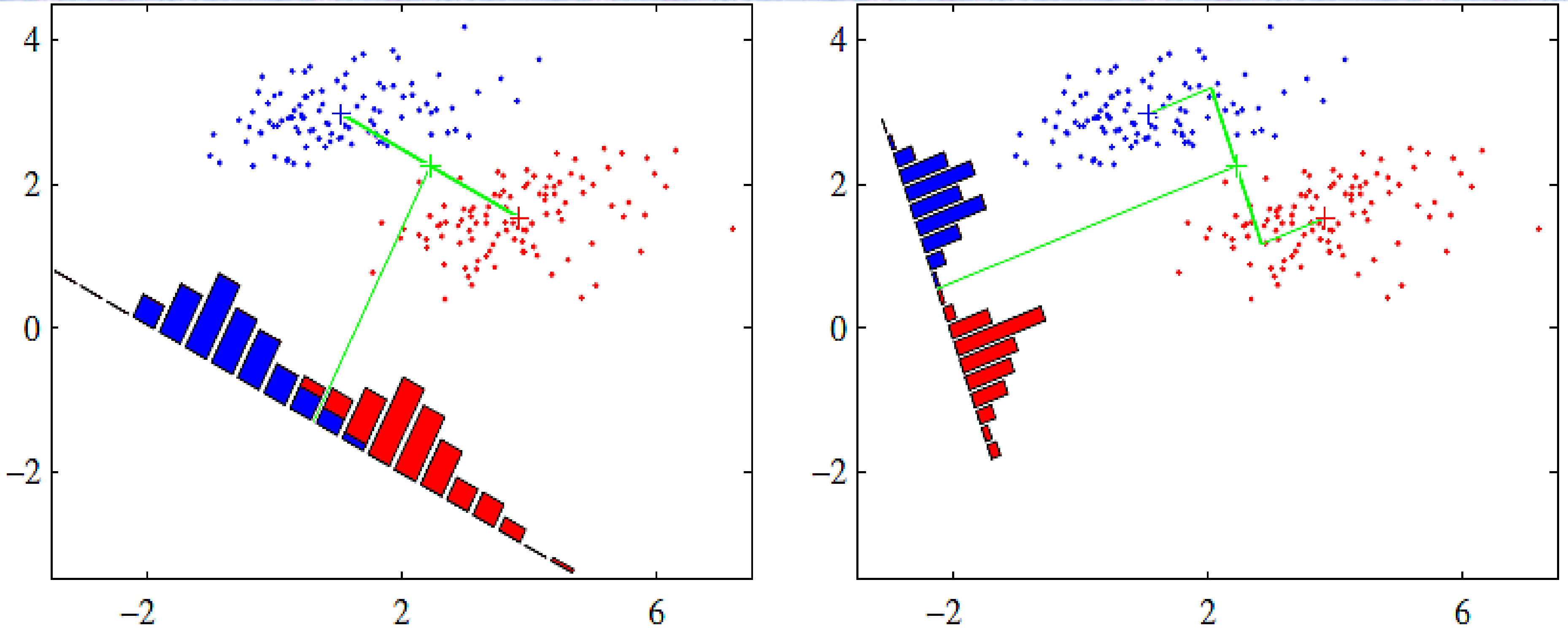
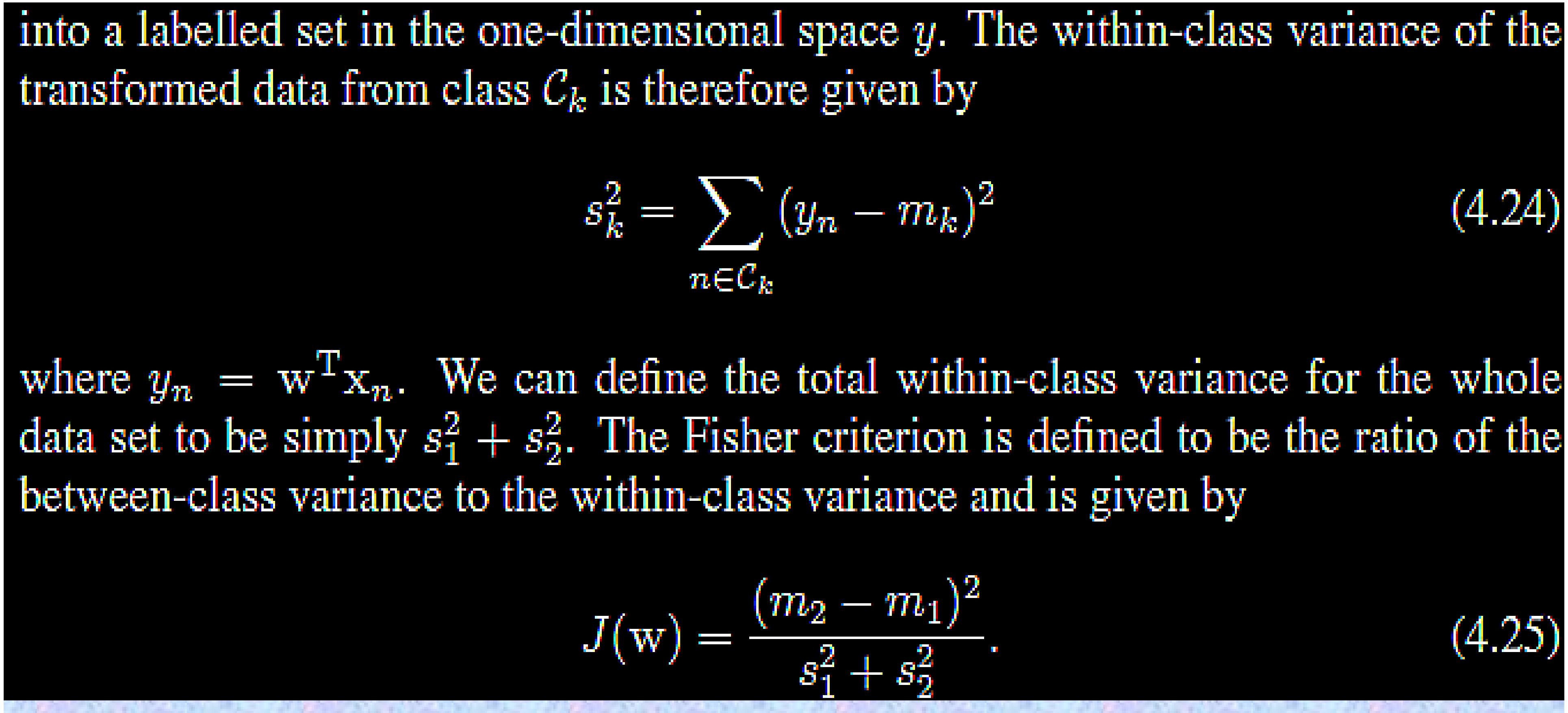


Figure 4.6 The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on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

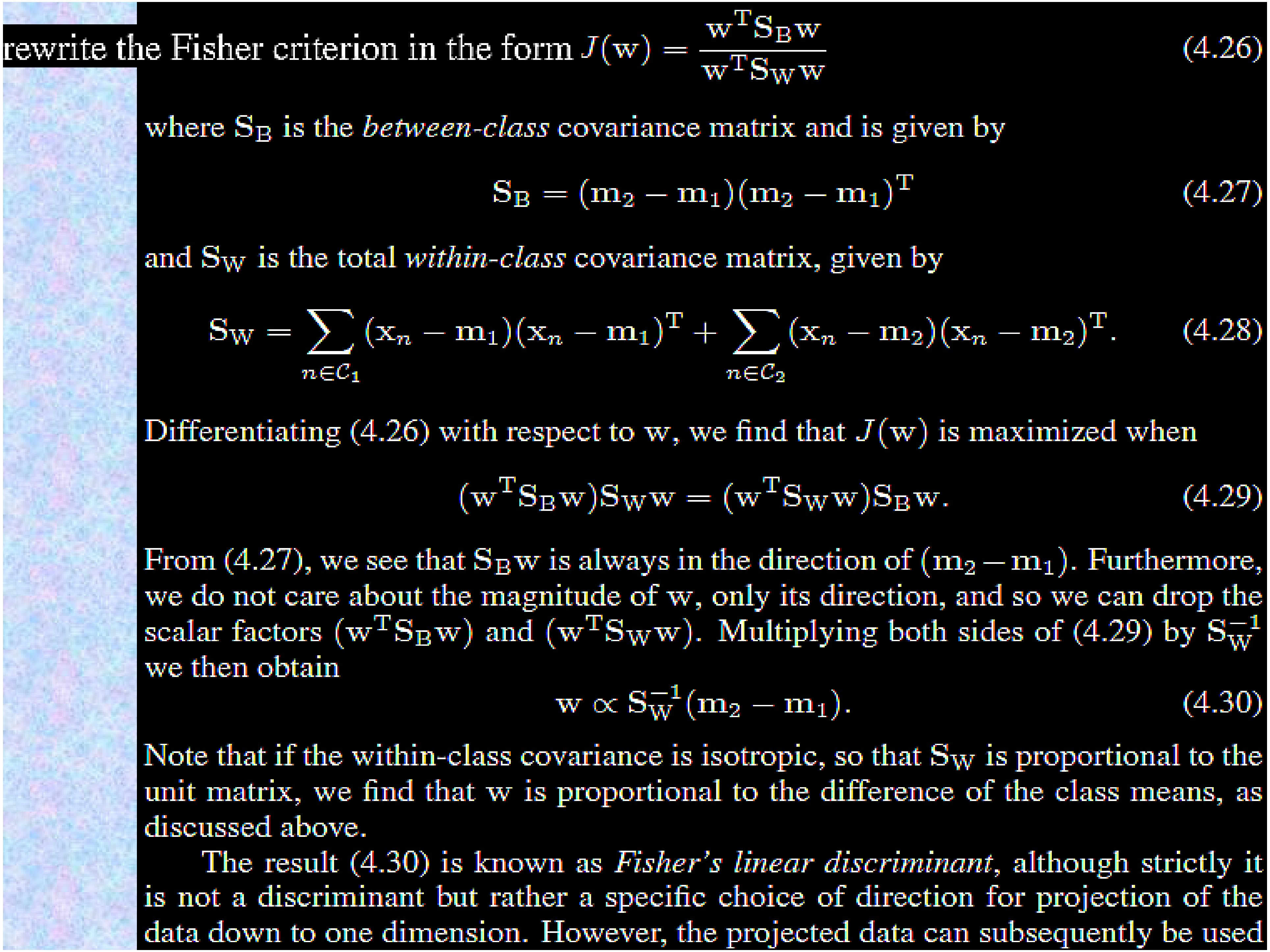


into a labelled set in the one-dimensional space y. The within-class variance of the transformed data from class $\mathcal{C}_k$ is therefore given by

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

where $y_n = \mathrm{w}^{\mathrm{T}}\mathrm{x}_n$. 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

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



rewrite the Fisher criterion in the form $J(\mathbf{w}) = \dfrac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}}$ (4.26)

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

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}} \tag{4.27}$$

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

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

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

$$(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})\mathbf{S}_{\mathrm{W}}\mathbf{w} = (\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})\mathbf{S}_{\mathrm{B}}\mathbf{w}. \tag{4.29}$$

From (4.27), we see that $\mathbf{S}_{\mathrm{B}}\mathbf{w}$ 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 $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})$ and $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})$. Multiplying both sides of (4.29) by $\mathbf{S}_{\mathrm{W}}^{-1}$ we then obtain

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1). \tag{4.30}$$

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

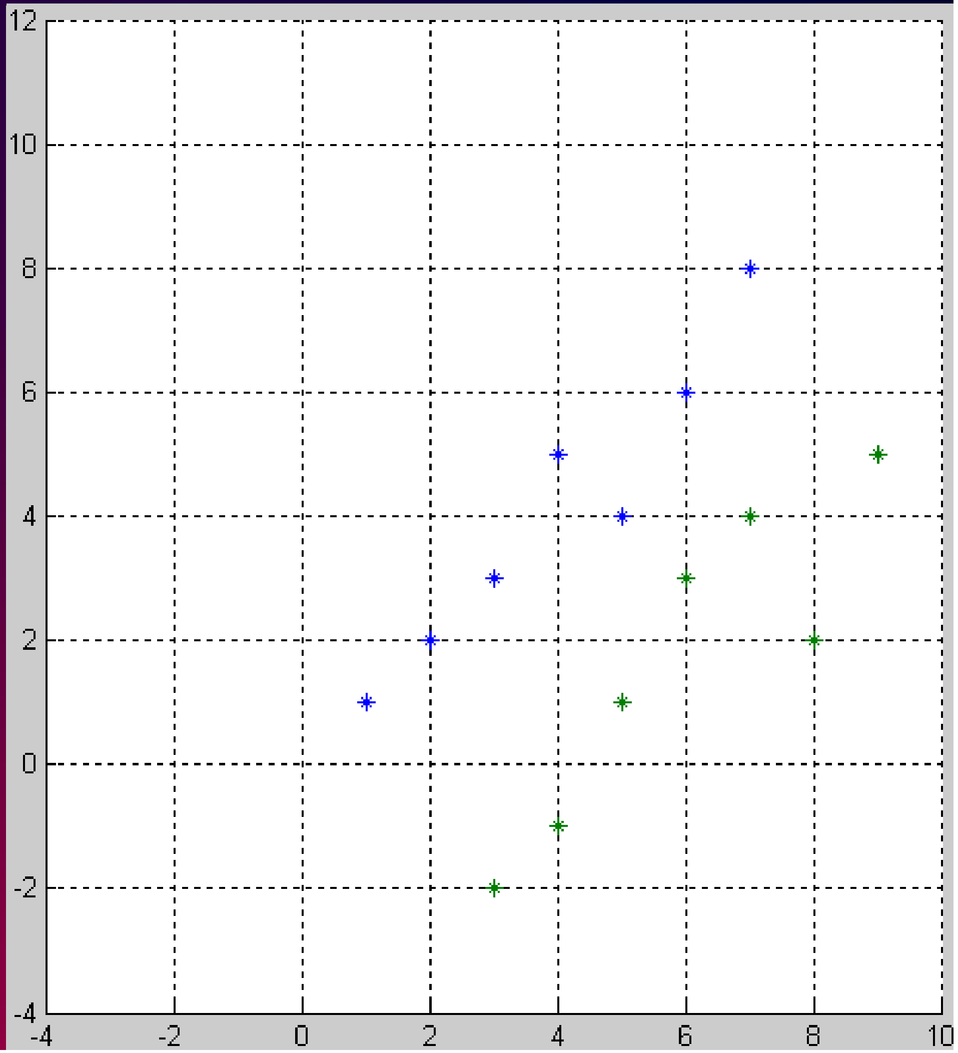
The result (4.30) is known as *Fisher's linear discriminant*, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension. However, the projected data can subsequently be used

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

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

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

$$\mathrm{INV}(S_w) \,.\, S_b =$$

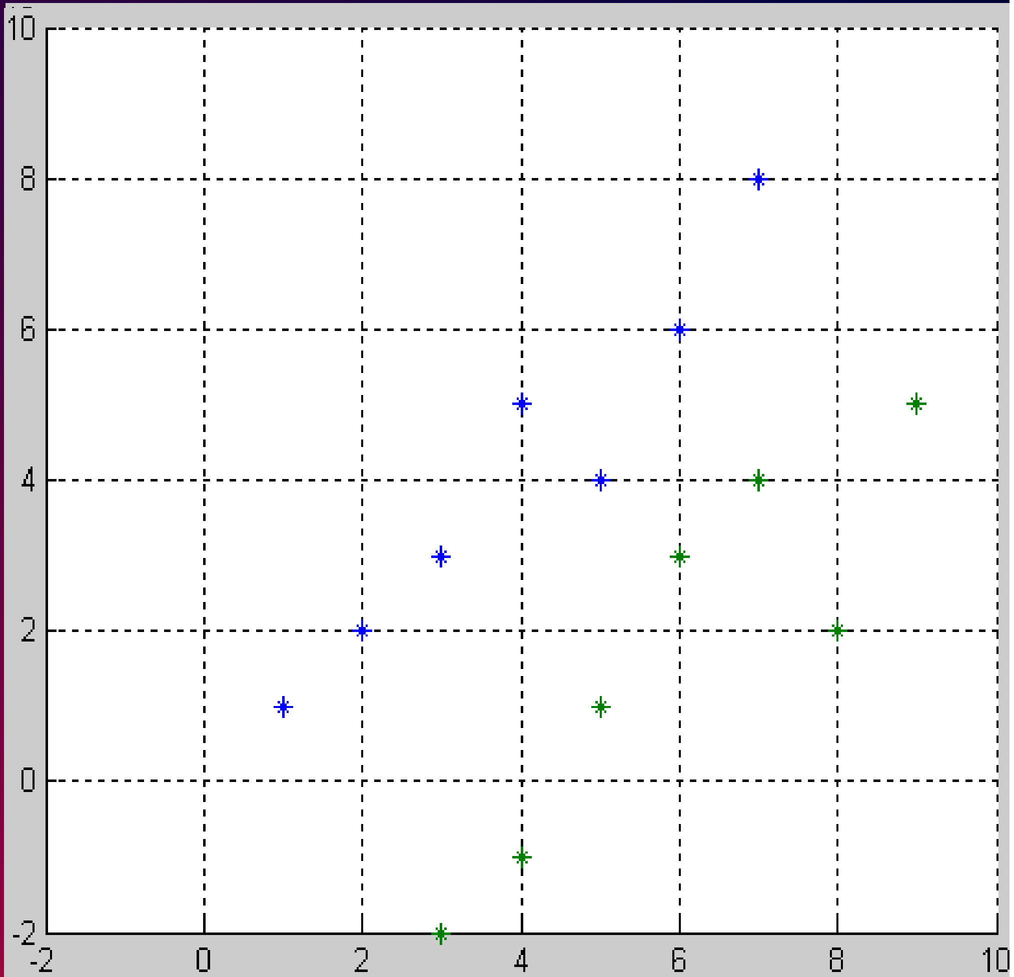
$$\begin{matrix} 27.20 & -22.40 \\ -31.268 & 25.75 \end{matrix}$$

Perform Eigendecomposition on above:

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

Eigenvectors:

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

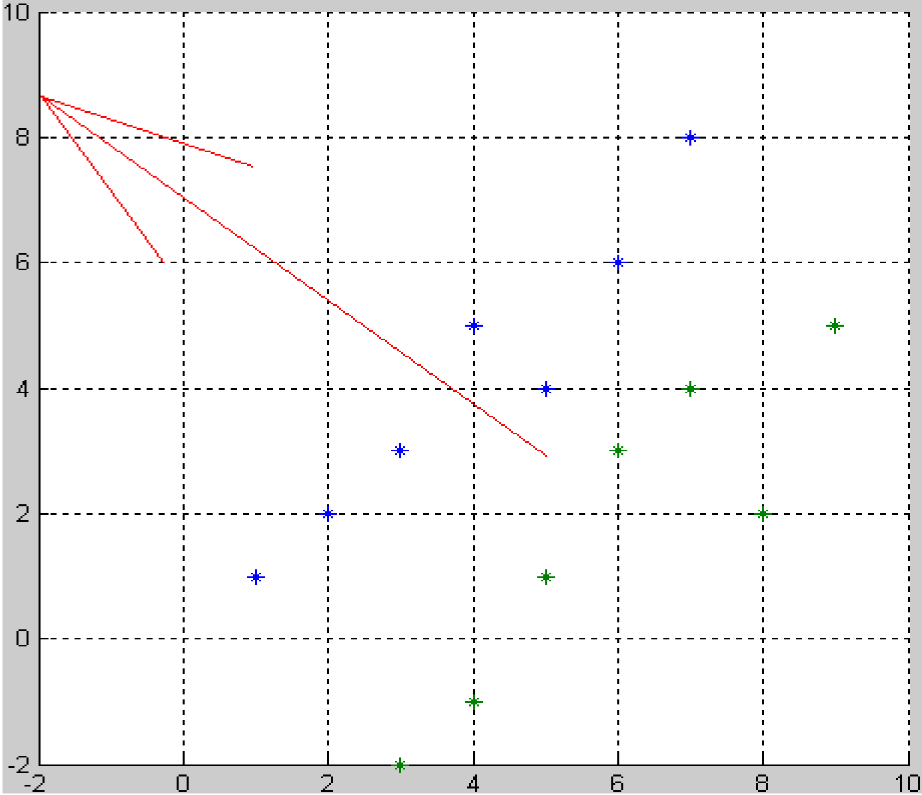


$$S_w = \begin{array}{cc} 10.6122 & 8.5714 \\ 8.5714 & 8.0000 \end{array}$$

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

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

Eigenvectors: $\begin{array}{cc} -0.7719 & 0.6357 \\ 0.6357 & 0.7719 \end{array}$

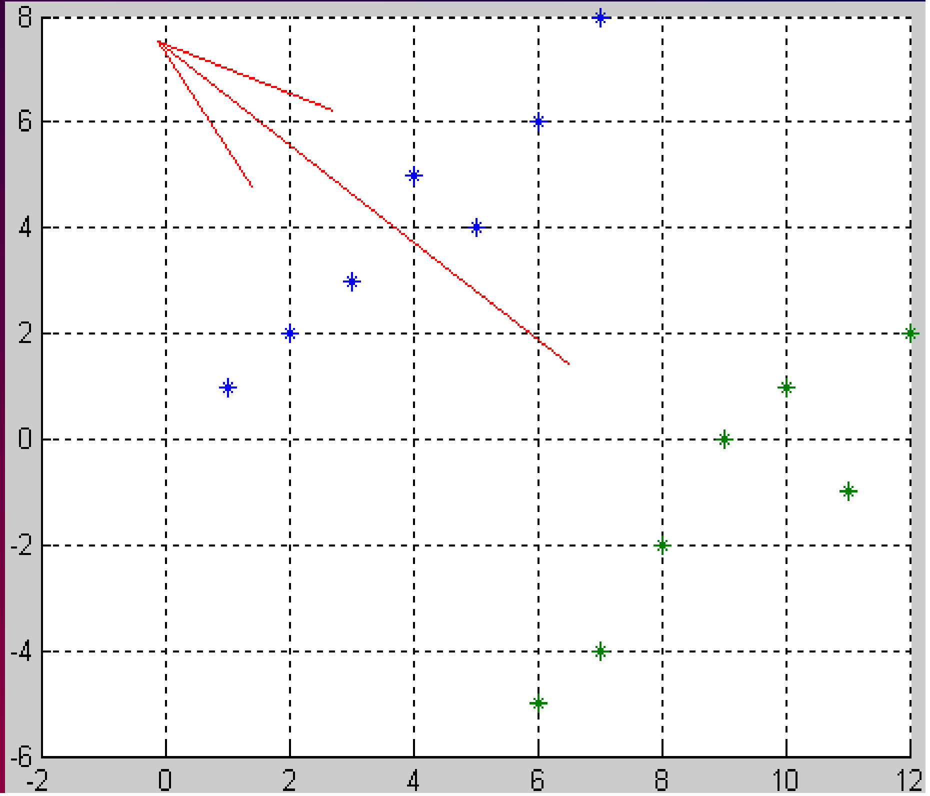


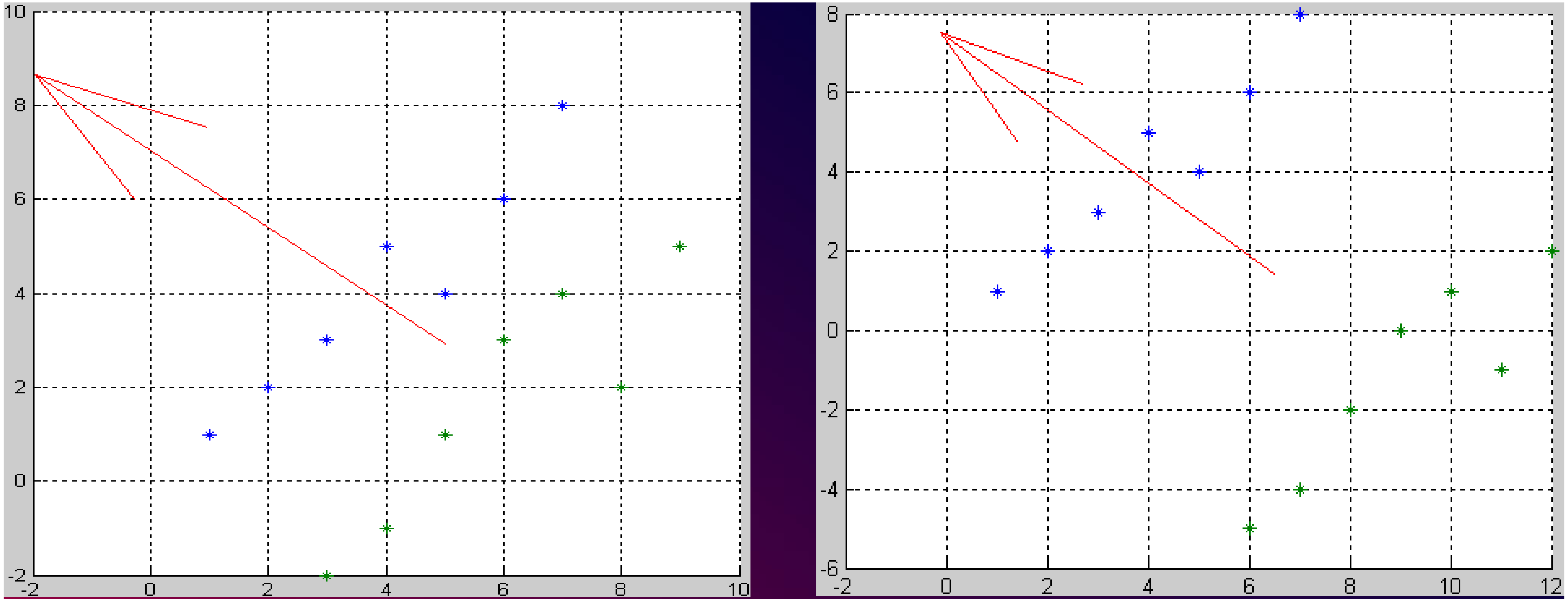
$$S_w = \begin{array}{cc} 10.6122 & 8.5714 \\ 8.5714 & 8.0000 \end{array}$$

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

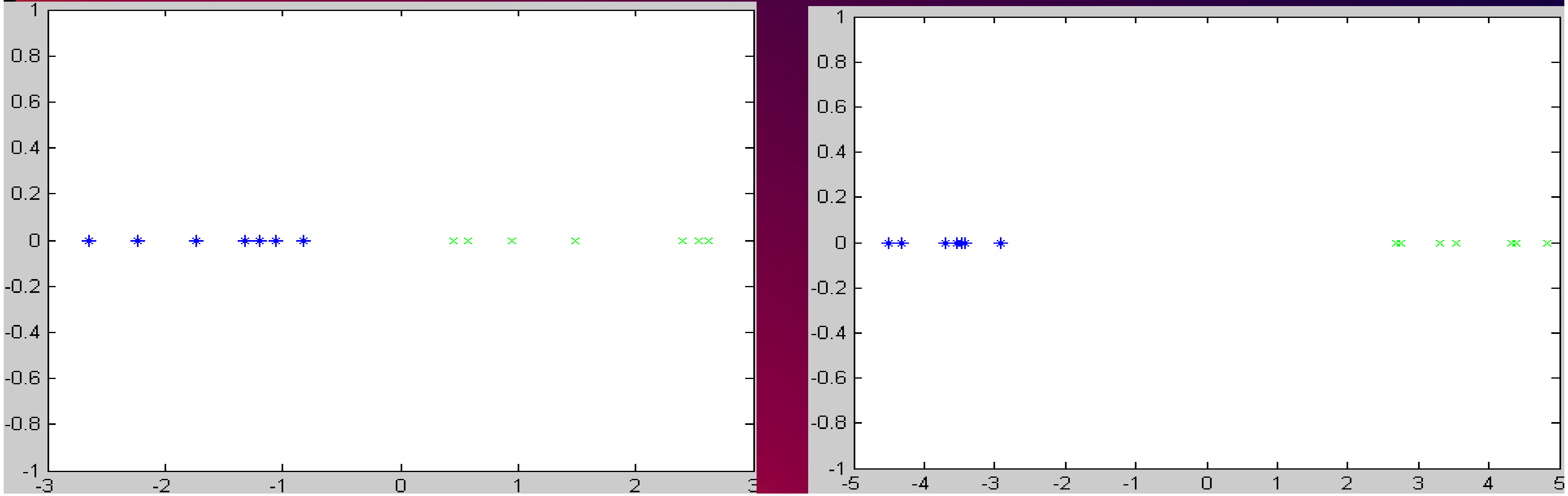
Eigenvalues of $S_w^{-1} S_b$: $\begin{array}{c} 297.83 \\ 0.0 \end{array}$

Eigenvectors: $\begin{array}{cc} -0.7355 & -0.6775 \\ 0.6775 & 0.7355 \end{array}$





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

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

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

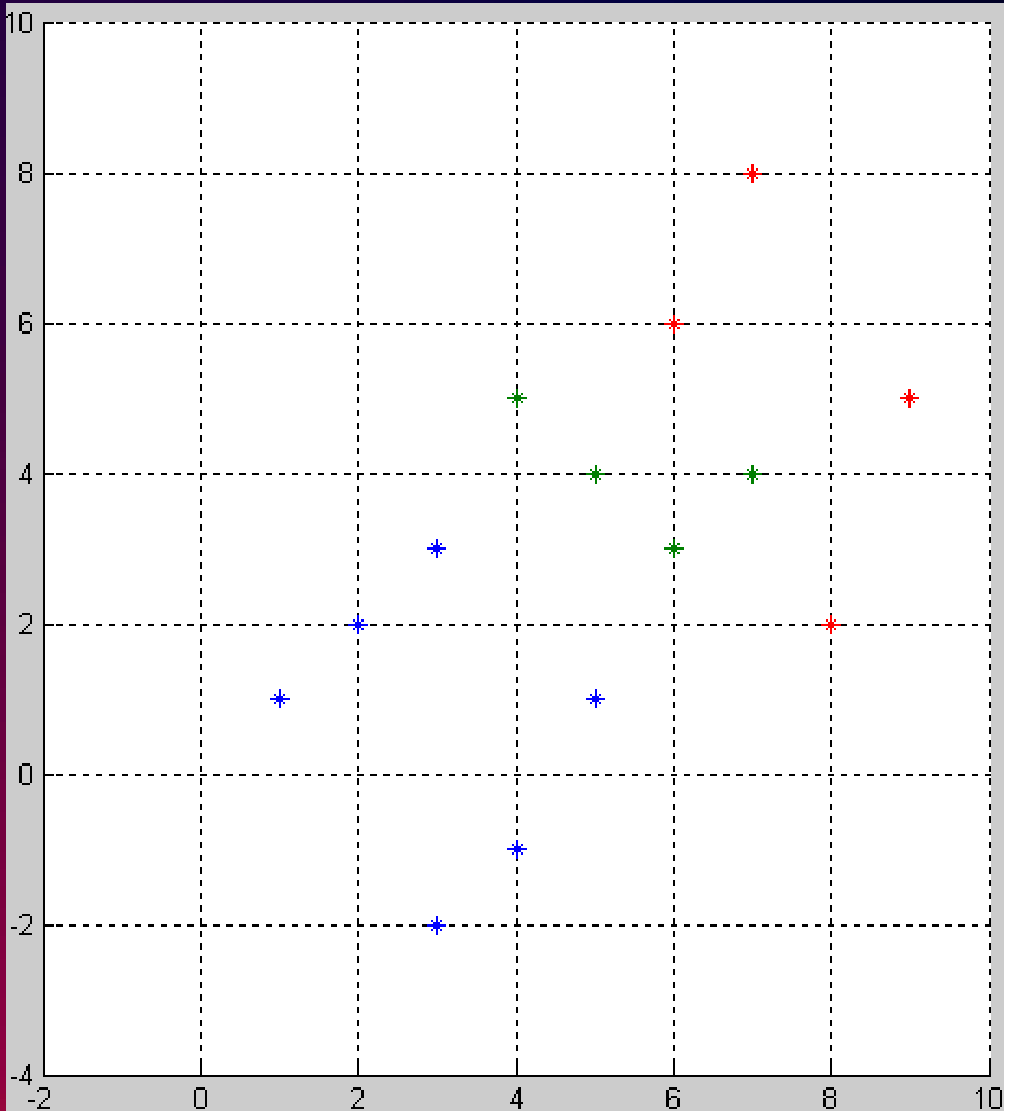
$$\mathrm{INV}(S_w)\ .\ S_b \ = \begin{matrix} 11.958 & 11.155 \\ 18.7 & 17.69 \end{matrix}$$

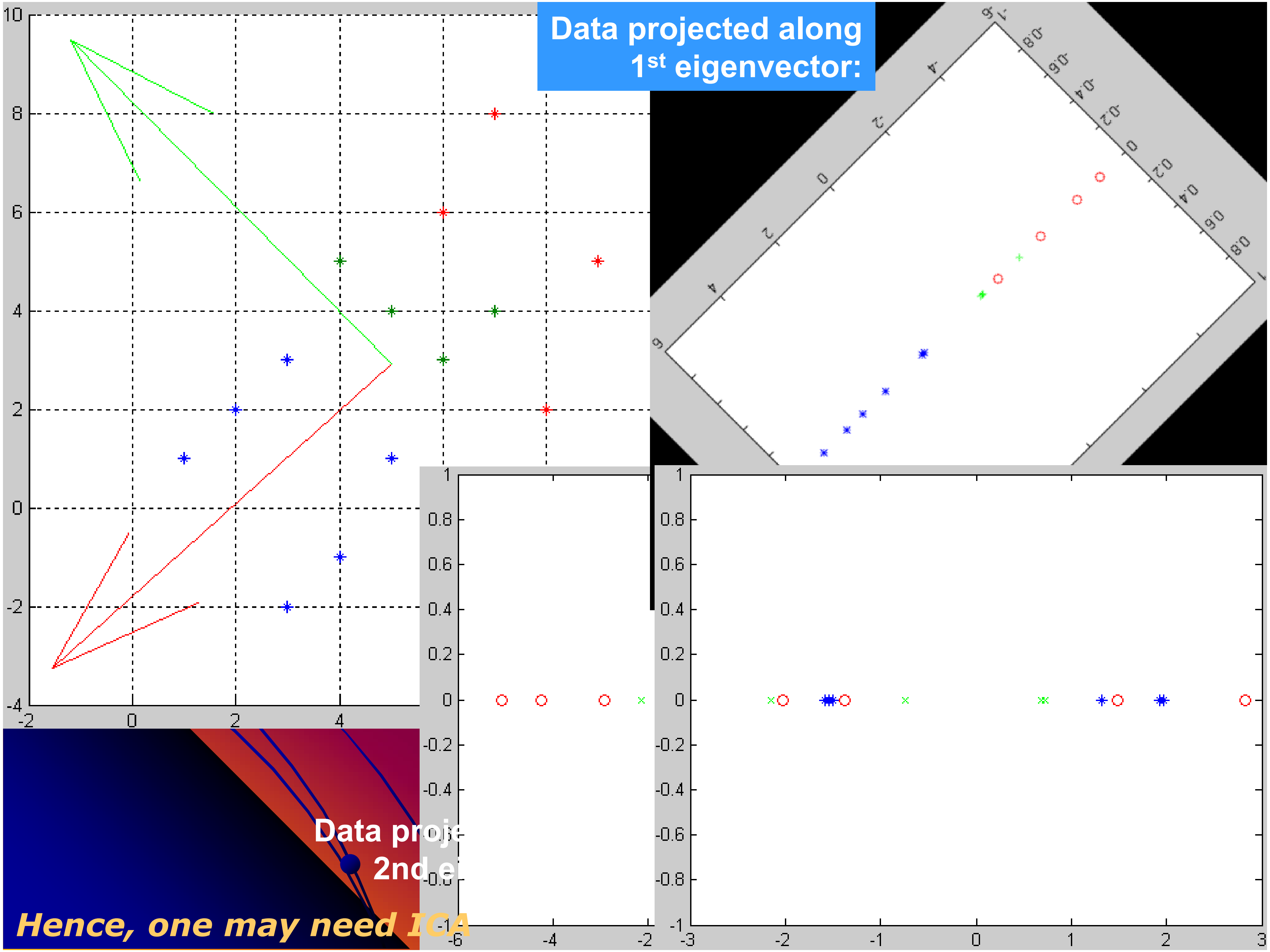
Perform Eigendecomposition on above:

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

Eigenvectors:

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





Data projected along 1st eigenvector:

Data proje
2nd ei

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 – semi-supervised, transfer learning, domain adaptation Manifold based learning, deep learning, MKL, ….**

REFERENCES

- Statistical pattern Recognition; S. Fukunaga; Academic Press, 2000.

- Bishop – PR

- Satish Kumar - ANN