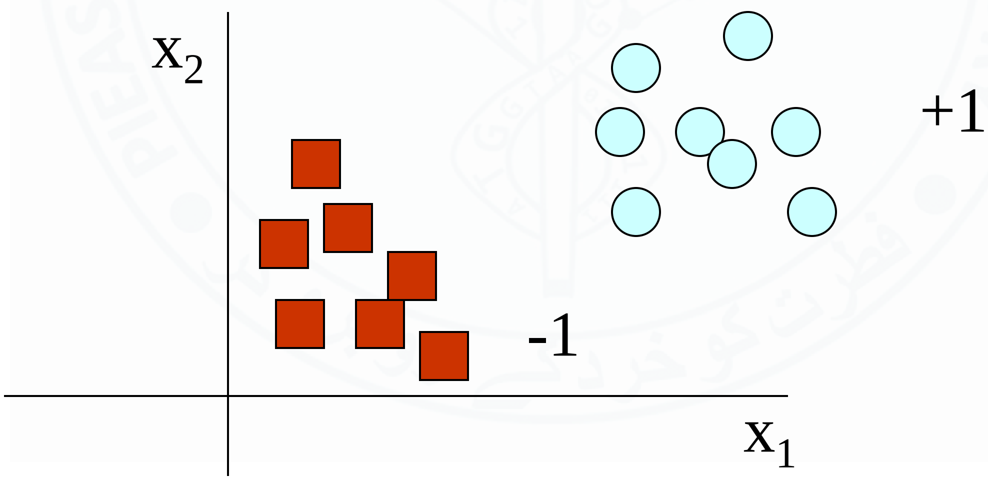
Support Vector Machines

Dr. Fayyaz ul Amir Afsar Minhas
PIEAS Biomedical Informatics Research Lab
Department of Computer and Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad, Pakistan
http://faculty.pieas.edu.pk/fayyaz/

Classification

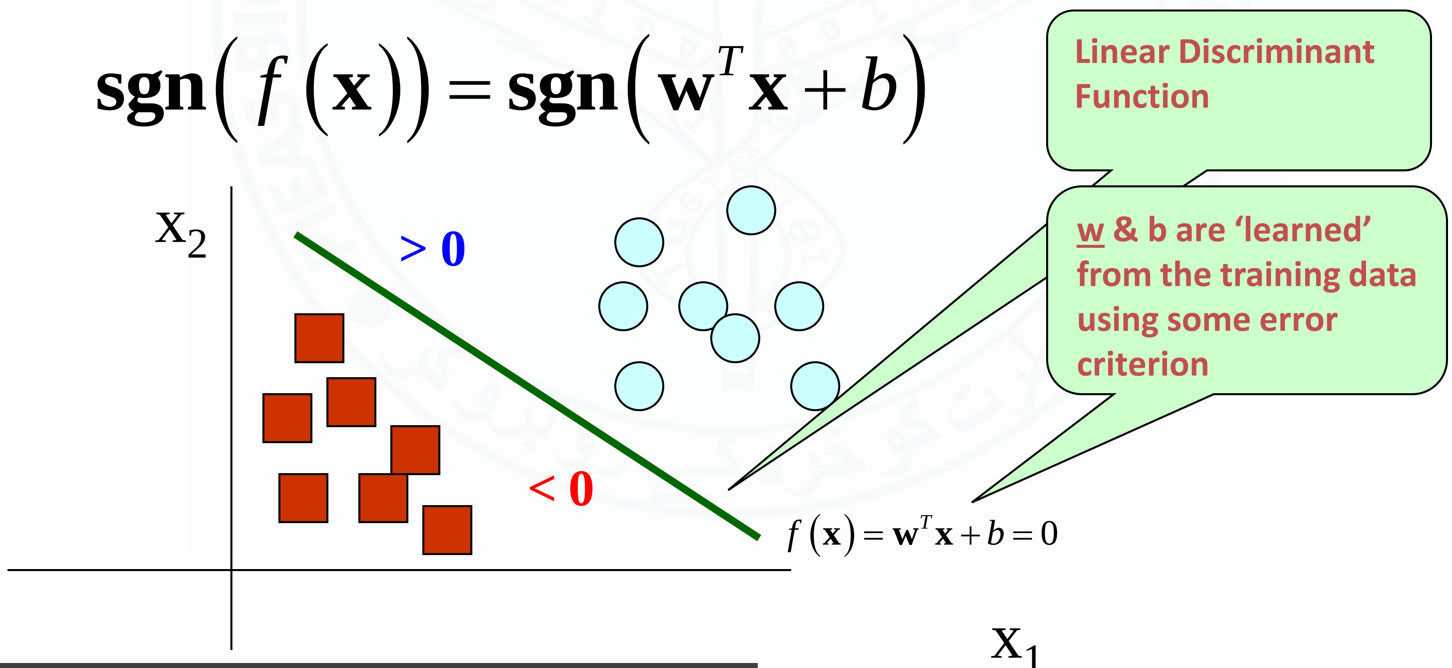
- Before moving on with the discussion let us restrict ourselves to the following problem
  - T = Given Training Set = $\{(\underline{x}^{(i)}, y_i), i = 1 \ldots N\}$
    - $\underline{x}^{(i)} \varepsilon R^m$ {Data Point i }
    - y_i: class of data point i (+1 or -1)



Use of Linear Discriminant in Classification

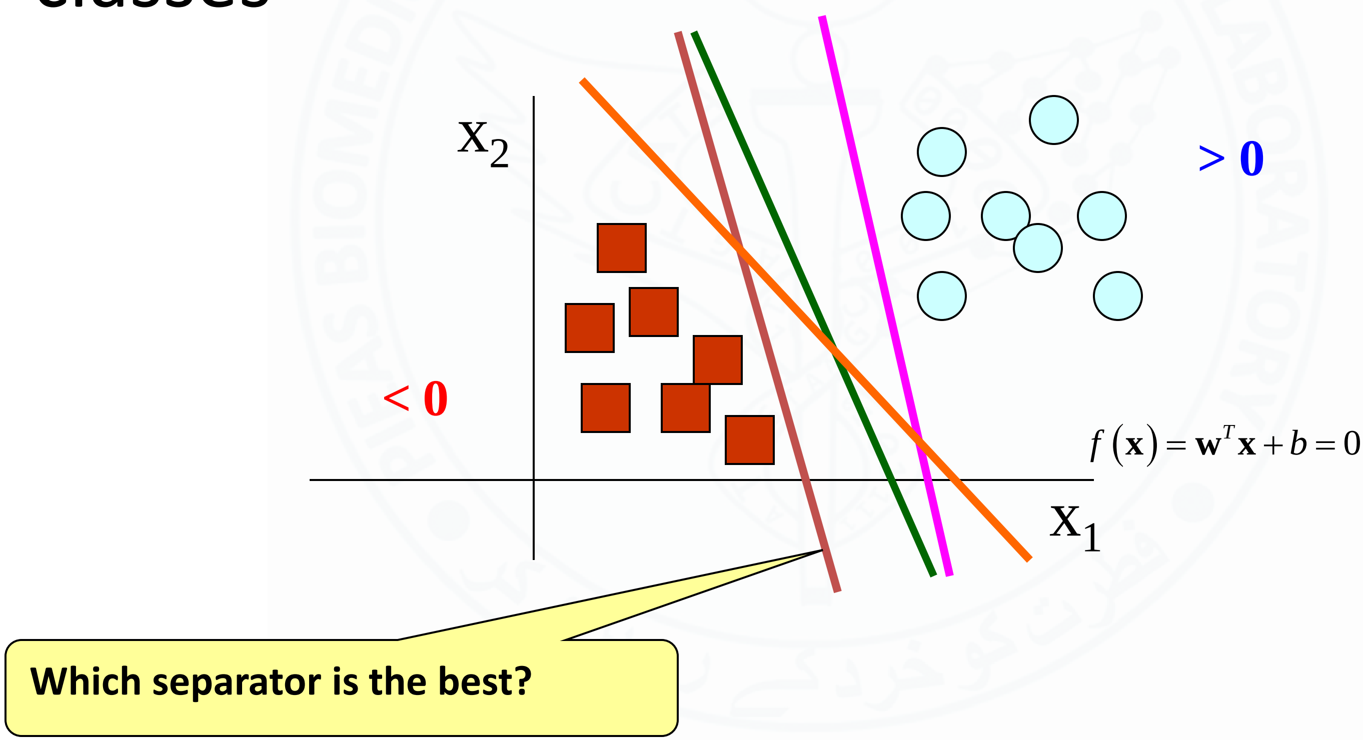
- Classifiers such as the Single Layer Perceptron (with linear activation function) and SVM use a linear discriminant function to differentiate between patterns of different classes
- The linear discriminant function is given by

$$\mathbf{sgn}\left(f\left(\mathbf{x}\right)\right)=\mathbf{sgn}\left(\mathbf{w}^{T}\mathbf{x}+b\right)$$

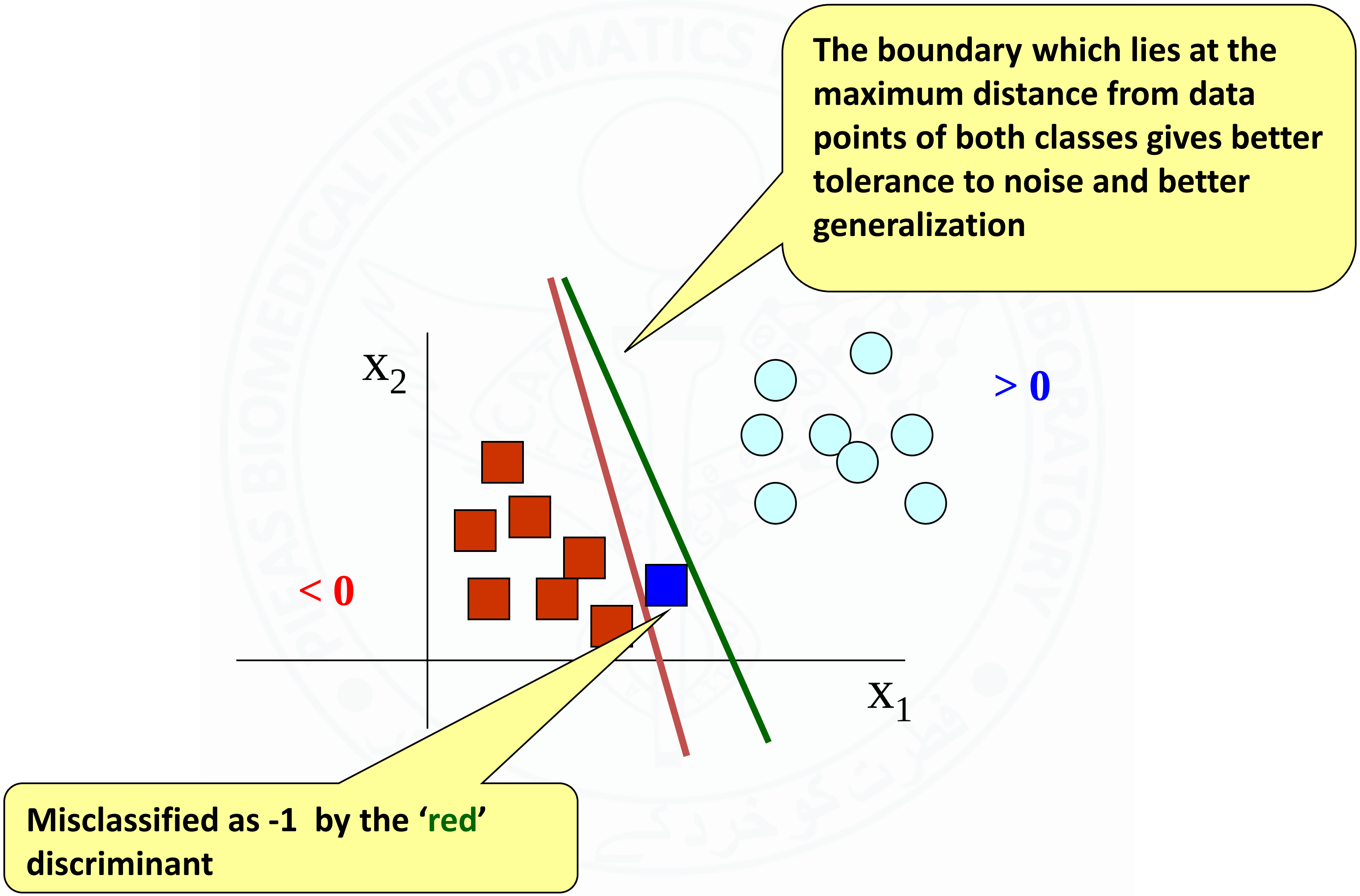


Use of Linear Discriminant in Classification

- There are a large number of lines (or in general ‘hyperplanes’) separating the two classes

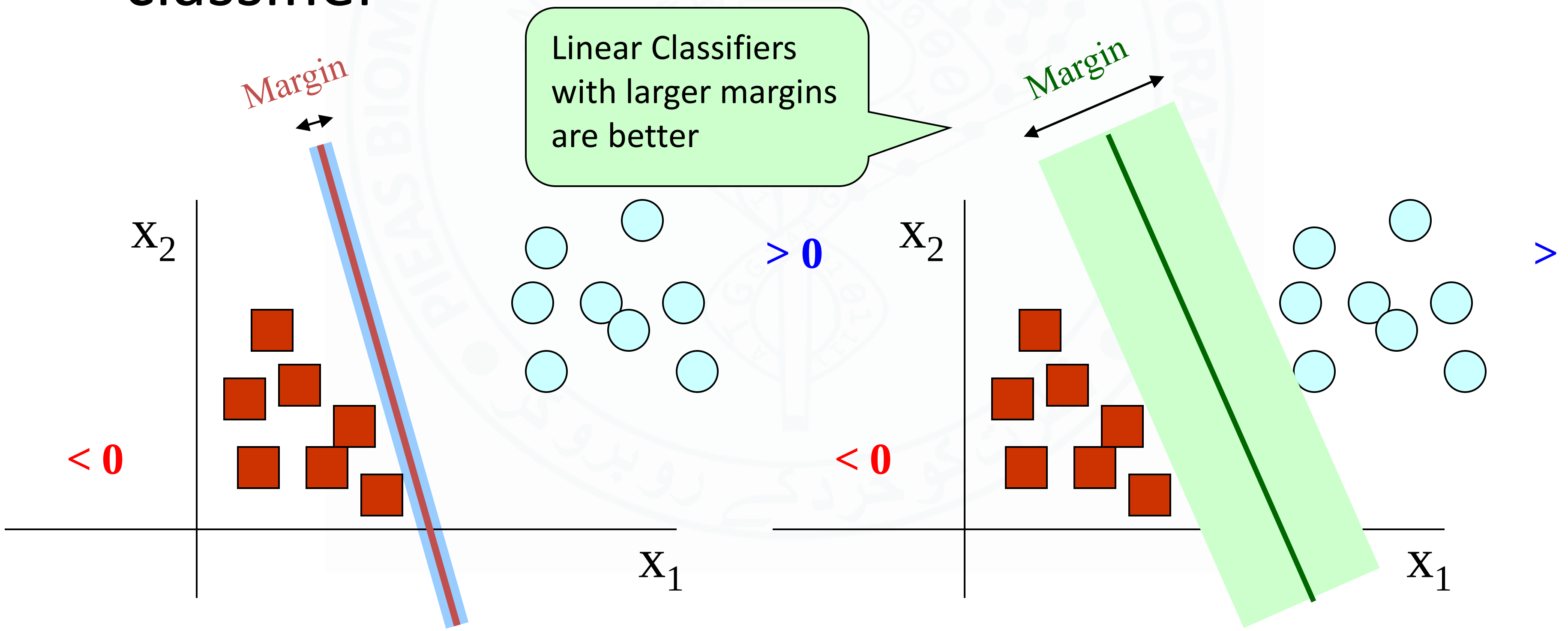


Use of Linear Discriminant in Classification



Margin of a linear classifier

- The width by which the boundary of a linear classifier can be increased before hitting a data point is called the margin of the linear classifier



Support Vector Machines (SVM)

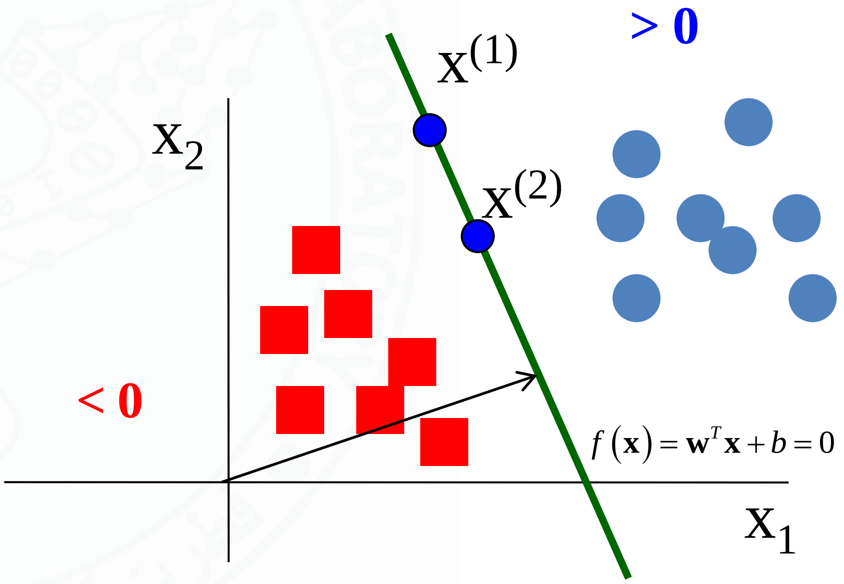
- Support Vector Machines are linear classifiers that produce the optimal separating boundary (hyper-plane)
  - Find **w** and b in a way so as to maximize the margin while classifying all the training patterns correctly (for linearly separable problem)

Finding Margin of a Linear Classifier

- Consider a linear classifier with the boundary

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0 \quad \textbf{for all } \mathbf{x} \textbf{ on the boundary}$$

- We know that the vector w is perpendicular to the boundary
  - Consider two points $x^{(1)}$ and $x^{(2)}$ on the boundary



$$f\left(\mathbf{x}^{(1)}\right) = \mathbf{w}^T\mathbf{x}^{(1)} + b = 0 \qquad (1)$$

$$f\left(\mathbf{x}^{(2)}\right) = \mathbf{w}^T\mathbf{x}^{(2)} + b = 0 \qquad (2)$$

Subtracting (1) from (2)

$$\mathbf{w}^T\left(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\right) = 0 \quad \Rightarrow \quad \mathbf{w} \perp \left(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\right)$$

Finding Margin of a Linear Classifier

- Let $x^{(s)}$ be a point in the feature space with its projection $x^{(p)}$ on the boundary

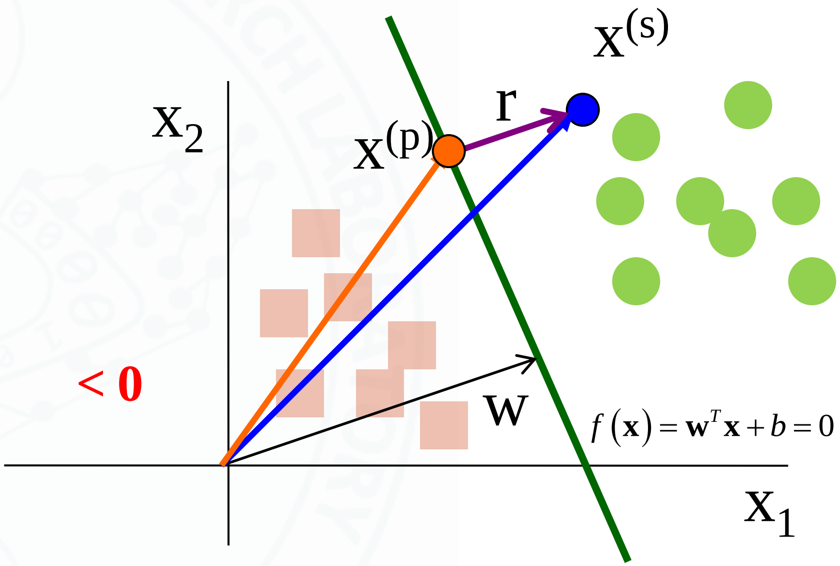
$$f\left(\mathbf{x}^{(p)}\right) = \mathbf{w}^T\mathbf{x}^{(p)} + b = 0$$

- We know that,

$$\mathbf{x}^{(\mathrm{s})} = \mathbf{x}^{(\mathrm{p})} + r\hat{\mathbf{w}}$$

$$\begin{aligned}
\Rightarrow f\left(\mathbf{x}^{(\mathrm{s})}\right) &= \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{s})} + b \\
&= \mathbf{w}^{\mathrm{T}}\left(\mathbf{x}^{(\mathrm{p})} + r\hat{\mathbf{w}}\right) + b \\
&= \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{p})} + r\mathbf{w}^{\mathrm{T}}\hat{\mathbf{w}} + b \\
&= \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{p})} + b + r\mathbf{w}^{\mathrm{T}}\frac{\mathbf{w}}{\|\mathbf{w}\|} \\
&= 0 + r\|\mathbf{w}\| = r\|\mathbf{w}\|
\end{aligned}$$

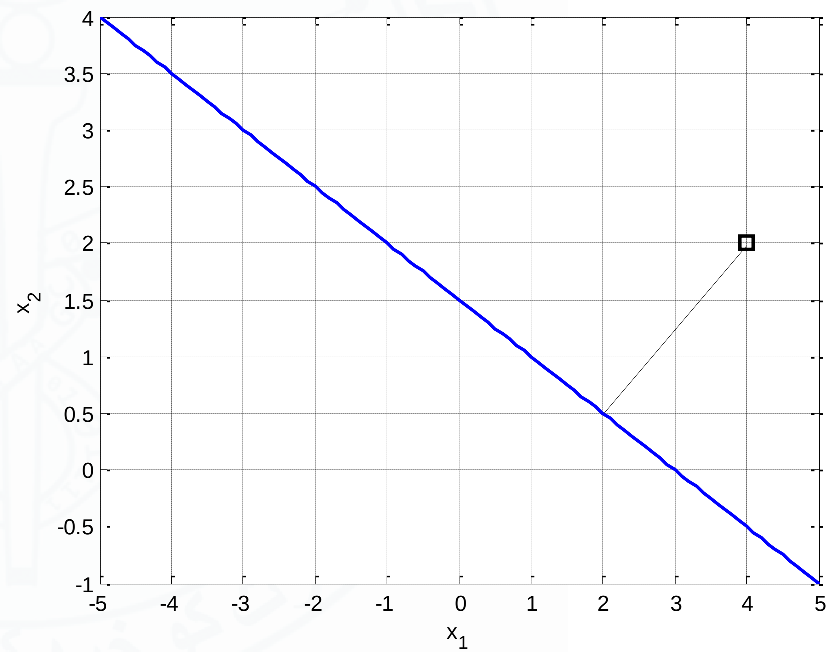
$$\Rightarrow r = \frac{f\left(\mathbf{x}^{(\mathrm{s})}\right)}{\|\mathbf{w}\|}$$



Perpendicular Distance of a point from a line

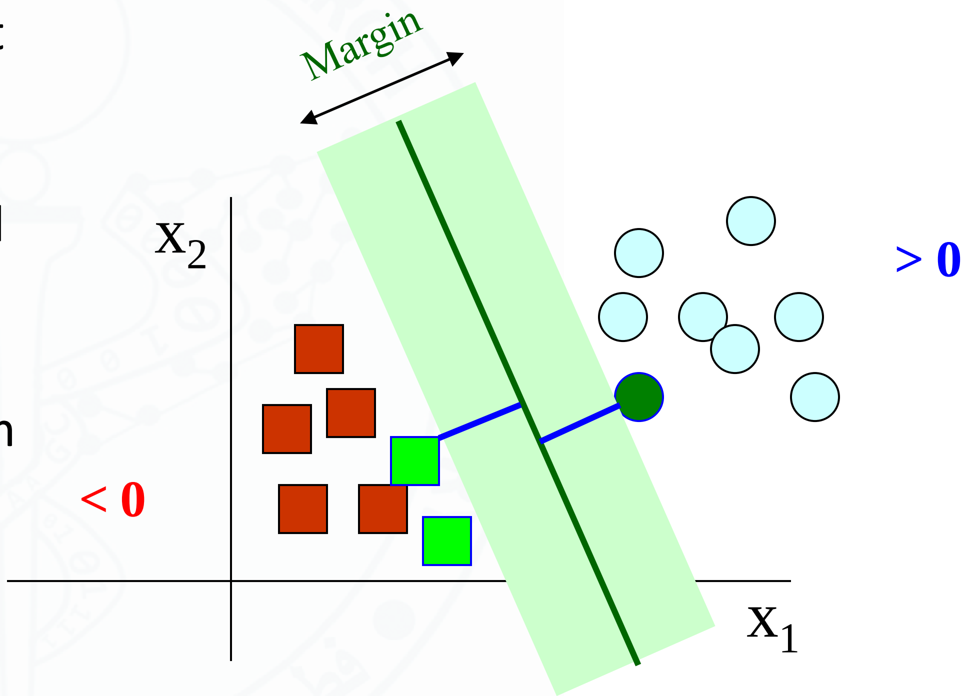
Example

- Consider the line
  - $x_1+2x_2+3 = 0$
- The distance of (4,2) is
  - $r = 4.92$



Finding Margin of a Linear Classifier

- The points in the training set that lie closest (having minimum perpendicular distance) to the separating hyper-plane are called support vectors
- Margin is twice of perpendicular distances of the hyper-plane from the nearest support vector of the two classes

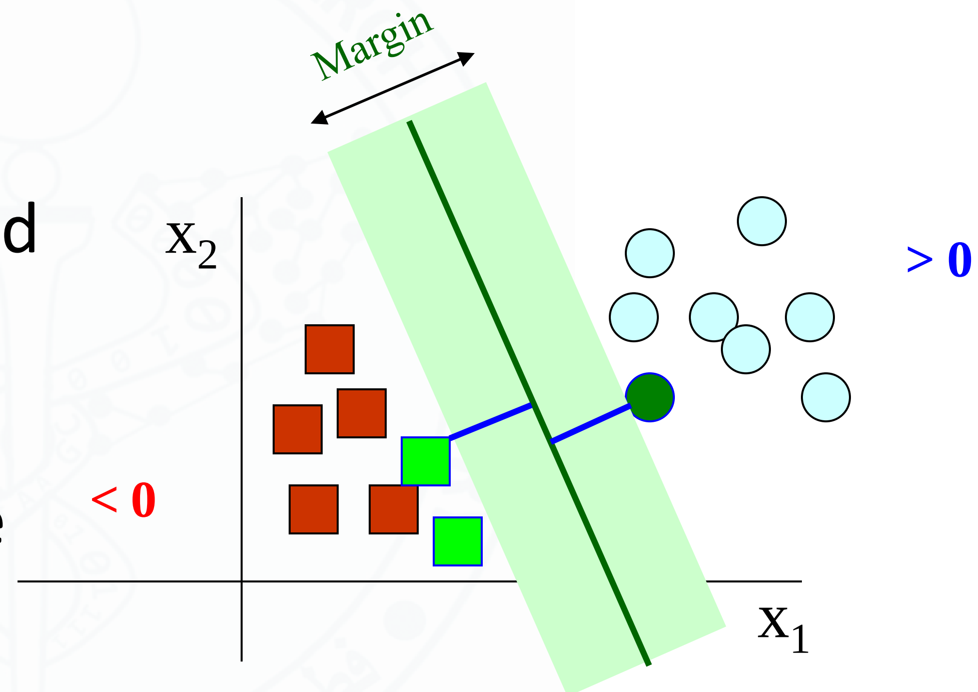


$$\rho = 2|r| = 2\left|\frac{f\left(\mathbf{x}^*\right)}{\|\mathbf{w}\|}\right|$$

Nearest Support Vector

Geometric vs. Functional Margin

- Functional Margin
  - This gives the position of the point with respect to the plane, which does not depend on the magnitude.
- Geometric Margin
  - This gives the distance between the given training example and the given plane.



Finding Margin of a Linear Classifier

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(x^{(i)}, y_i)\}$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} + b \leq -1 \quad \text{if } y_i = -1$$

$$\Rightarrow \rho = 2|r| = 2\left|\frac{f\left(\mathbf{x}^*\right)}{\|\mathbf{w}\|}\right| = 2\left|\frac{\pm 1}{\|\mathbf{w}\|}\right|$$

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Support Vector Machines

- Support Vector Machines, in their basic form, are linear classifiers that are trained in a way so as to maximize the margin
- Principles of Operation
  - Define what an optimal hyper-plane is (in way that can be identified in a computationally efficient way)
    - Maximize margin
      - Allows noise tolerance
  - Extend the above definition for non-linearly separable problems
    - have a penalty term for misclassifications
  - Map data to an alternate space where it is easier to classify with linear decision surfaces
    - reformulate problem so that data is mapped implicitly to this space (using kernels)

Margin Maximization in SVM

- We know that if we require

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}^{(\mathbf{i})} + b \leq -1 \quad \text{if } y_i = -1$$

- Then the margin is

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

- Margin maximization can be performed by reducing the norm of the w vector

SVM as an Optimization problem

- We can present SVM as the following optimization problem

$$\mathbf{max} \quad \rho = \frac{2}{\|\mathbf{w}\|}$$

$$s.t.$$

$$\mathbf{w}^T\mathbf{x}^{(i)} + b \le -1, \qquad \forall i \ \textbf{s.t.}\ y_i = -1$$

$$\mathbf{w}^T\mathbf{x}^{(i)} + b \ge +1, \qquad \forall i \ \textbf{s.t.}\ y_i = +1$$

OR

$$\mathbf{min} \quad \rho = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$s.t.$$

$$y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \ge +1$$

Example: Solution of the OR problem

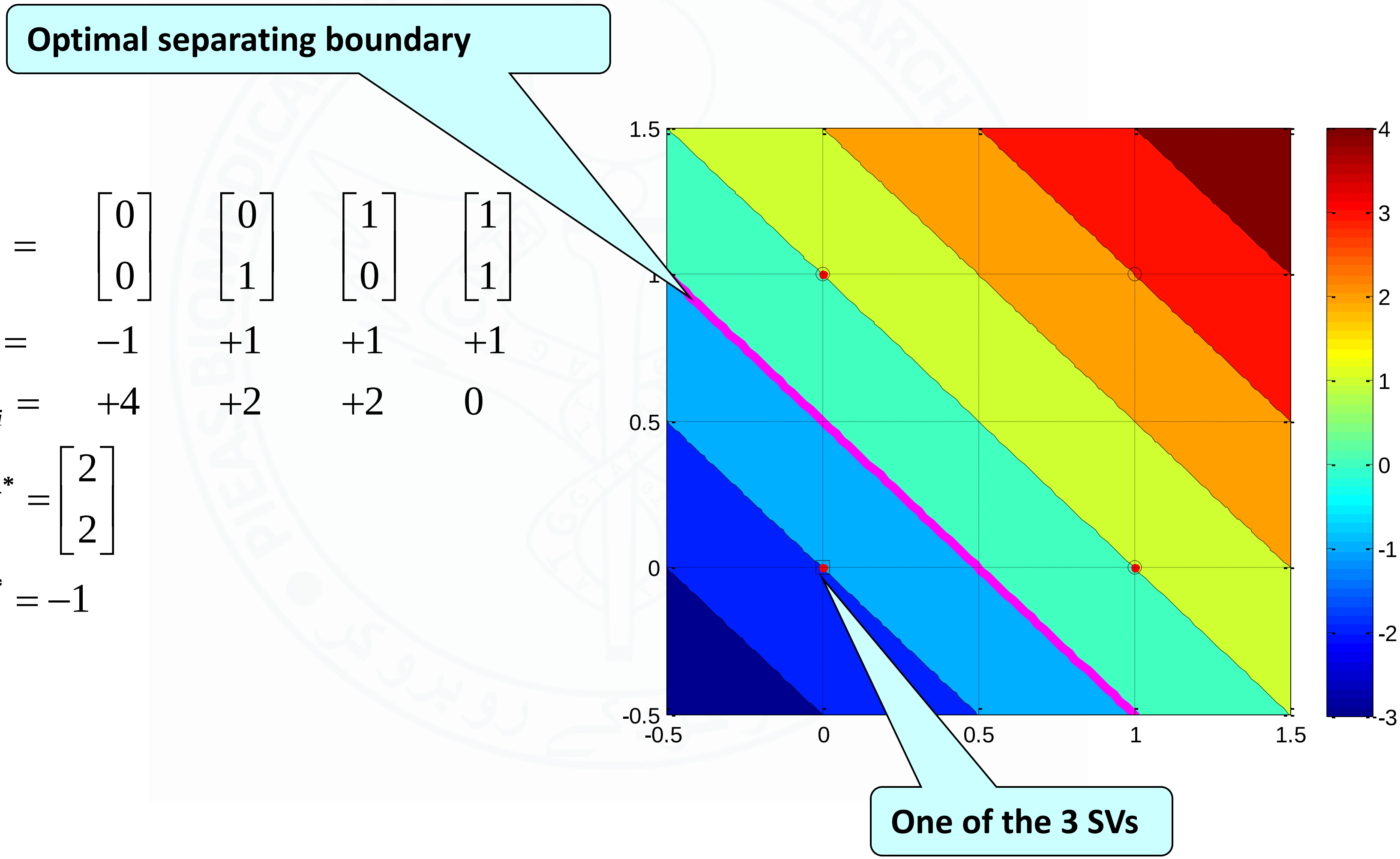
$$\mathbf{x}^i = \begin{bmatrix}0\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix}$$

$$y = -1 \quad +1 \quad +1 \quad +1$$

$$\alpha_i = +4 \quad +2 \quad +2 \quad 0$$

$$\mathbf{w}^* = \begin{bmatrix}2\\2\end{bmatrix}$$

$$b^* = -1$$



Handling Non-Separable Patterns

- All the derivation till now was based on the requirement that the data be linearly separable
  - Practical Problems are Non-Separable
- Non-separable data can be handled by relaxing the constraint

$$y_i\left(\mathbf{w}^T\mathbf{x}+b\right)\geq 1$$

- This is achieved as

Slack Variables

$$\left.\begin{aligned}&\mathbf{w}^T\mathbf{x}^{(i)}+b\geq 1-\zeta_i && \forall y_i=+1\\ &\mathbf{w}^T\mathbf{x}^{(i)}+b\geq -1+\zeta_i && \forall y_i=-1\end{aligned}\right\}\quad y_i\left(\mathbf{w}^T\mathbf{x}^{(i)}+b\right)\geq 1-\zeta_i$$

$$\zeta_i\geq 0$$

Soft Margin SVM as an Optimization Problem

- The overall optimization problem can be written as

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i$$

$$s.t. \quad y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0$$

Or

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i$$

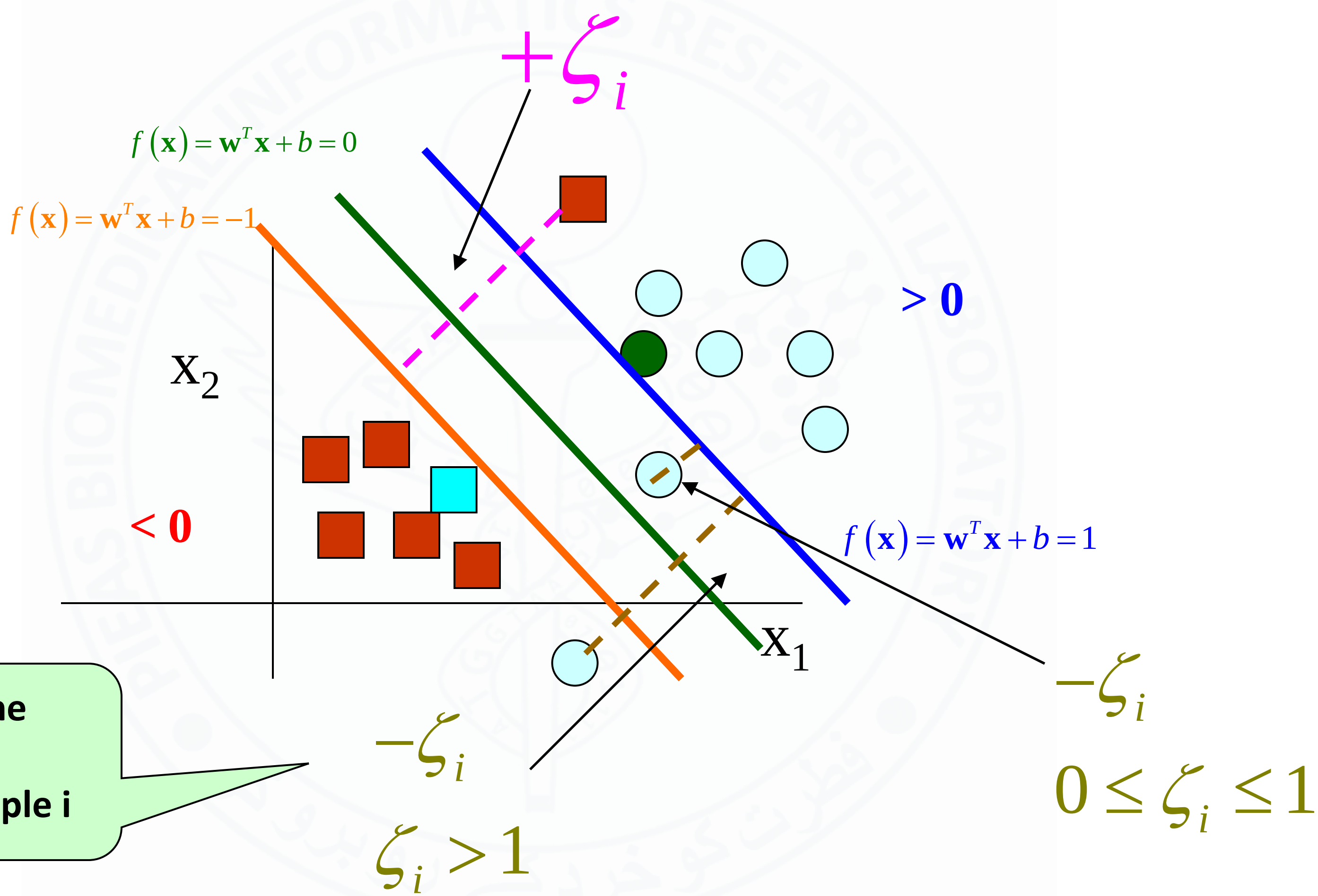
$$s.t. \quad 1 - \zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \leq 0$$

$$-\zeta_i \leq 0$$

Weight of the penalty due to margin violation

The objective function now optimizes two conflicting objectives: Maximization of the margin and minimization of the margin violations

Handling Non-Separable Patterns (Soft Margin)



ζ_i tells us about the extent of margin violation of example i

Example

$$\mathbf{x}^i = \begin{bmatrix}0\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix} \quad \begin{bmatrix}0.8\\0.5\end{bmatrix}$$

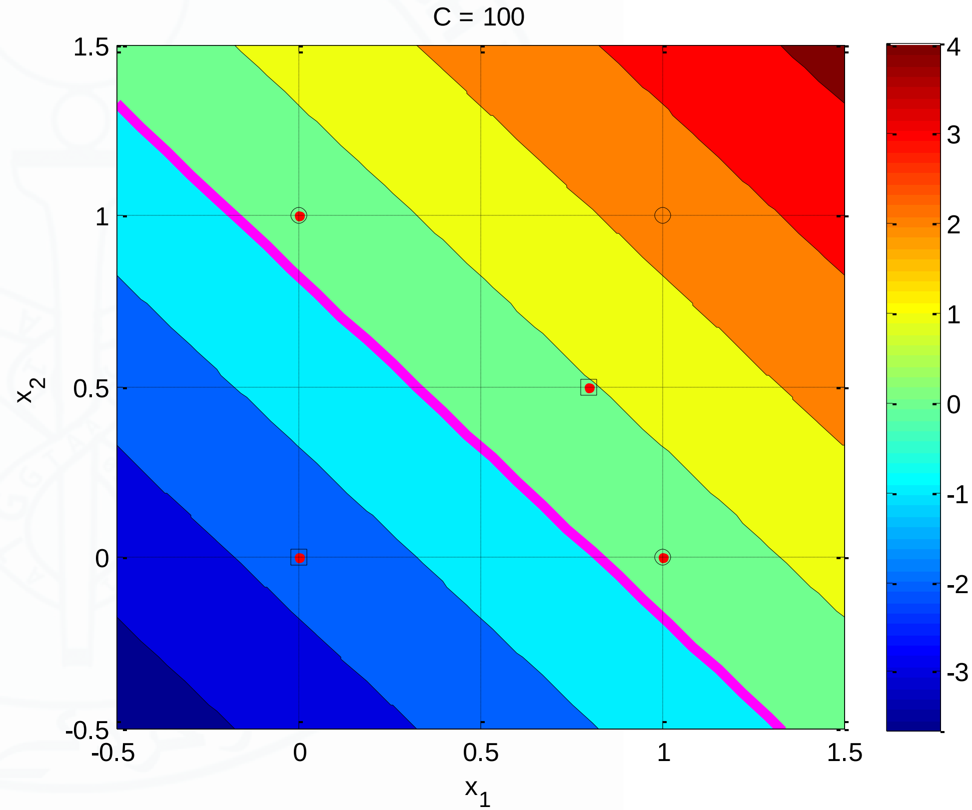
$C = 100$

$y = \quad -1 \quad +1 \quad +1 \quad +1 \quad -1$

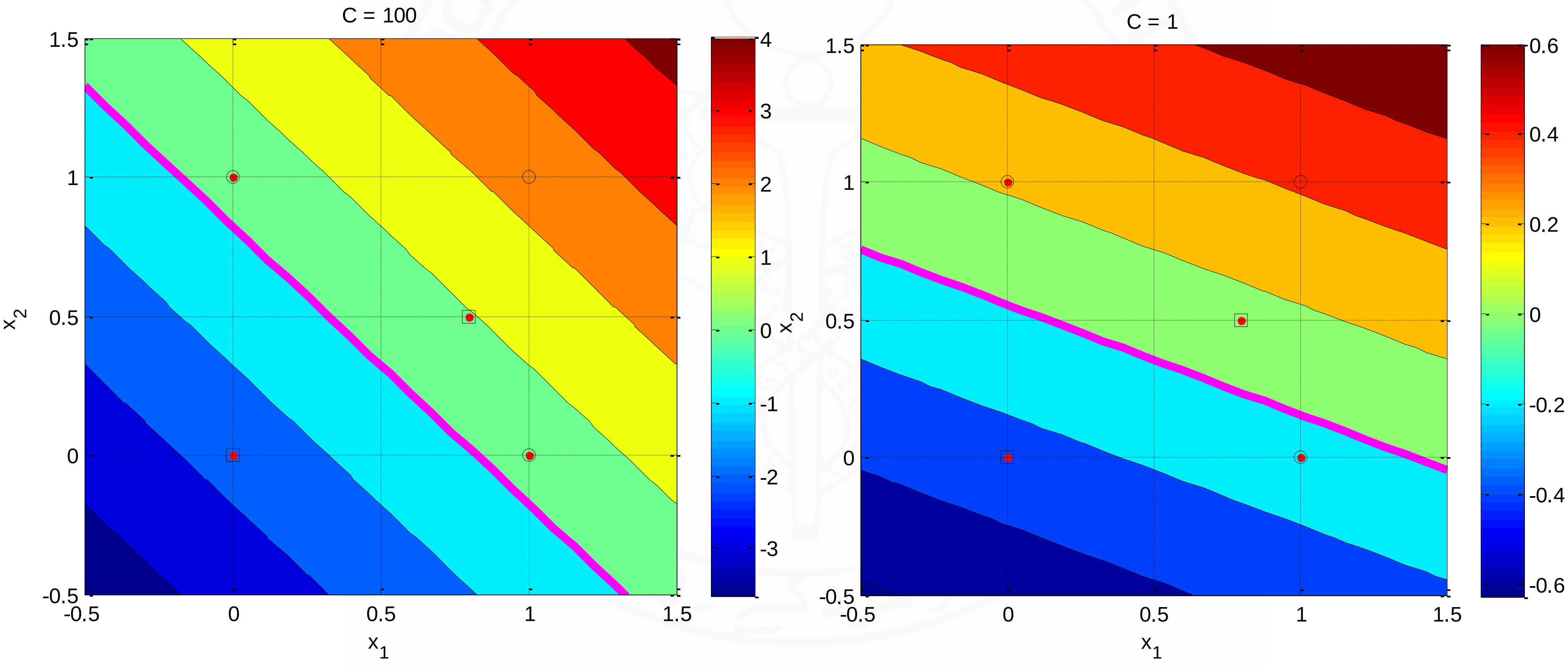
$\alpha_i = \quad +34 \quad +52 \quad +82 \quad 0 \quad +100$

$$\mathbf{w}^* = \begin{bmatrix}2\\2\end{bmatrix}$$

$b^* = -1.65$



Example…

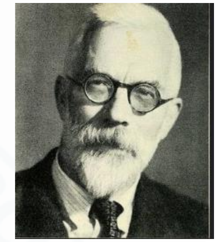


Wanna Play?

- Use the Java Applet at:
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Set “-t 0 -c 100”

SVMs uptil now

- Vapnik and Chervonenkis:
  - Hard SVM (1962)
  - Theoretical foundations for SVMs
  - Structural Risk Minimization
- Corinna Cortes
  - Soft SVM (1995)
- Enter: Bernard Scholkopf (1997)
  - Representer Theorem
  - Complete Kernel trick!
  - Kernels not only allow nonlinear boundaries but also allow representation of non-vectoral data



R. A. Fisher
1890-1962

Rosenblatt
1928-1971

V. Vapnik
1936 -

Chervonenkis
1938 - 2014

C. Cortes
1961 -

Scholkopf
1968 -

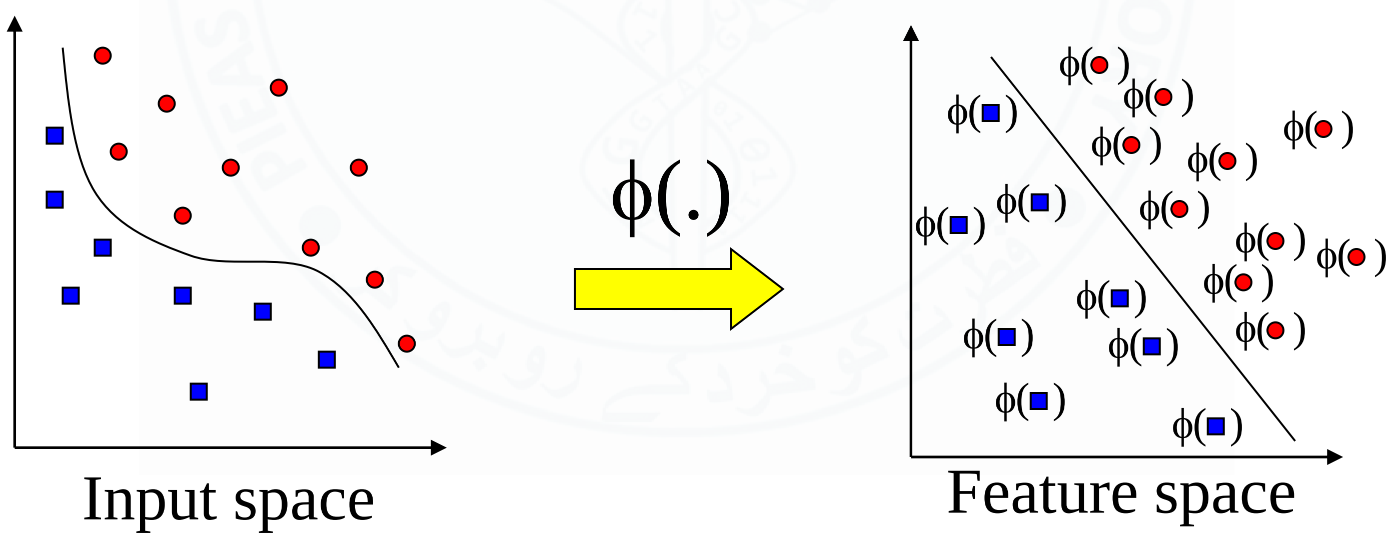
http://www.svms.org/history.html

Reading

- Sections 10.1-10.3
- Sections 13.1-13.3
- Alpaydin, Ethem. Introduction to Machine Learning. Cambridge, Mass. MIT Press, 2010.

Nonlinear Separation through Transformation

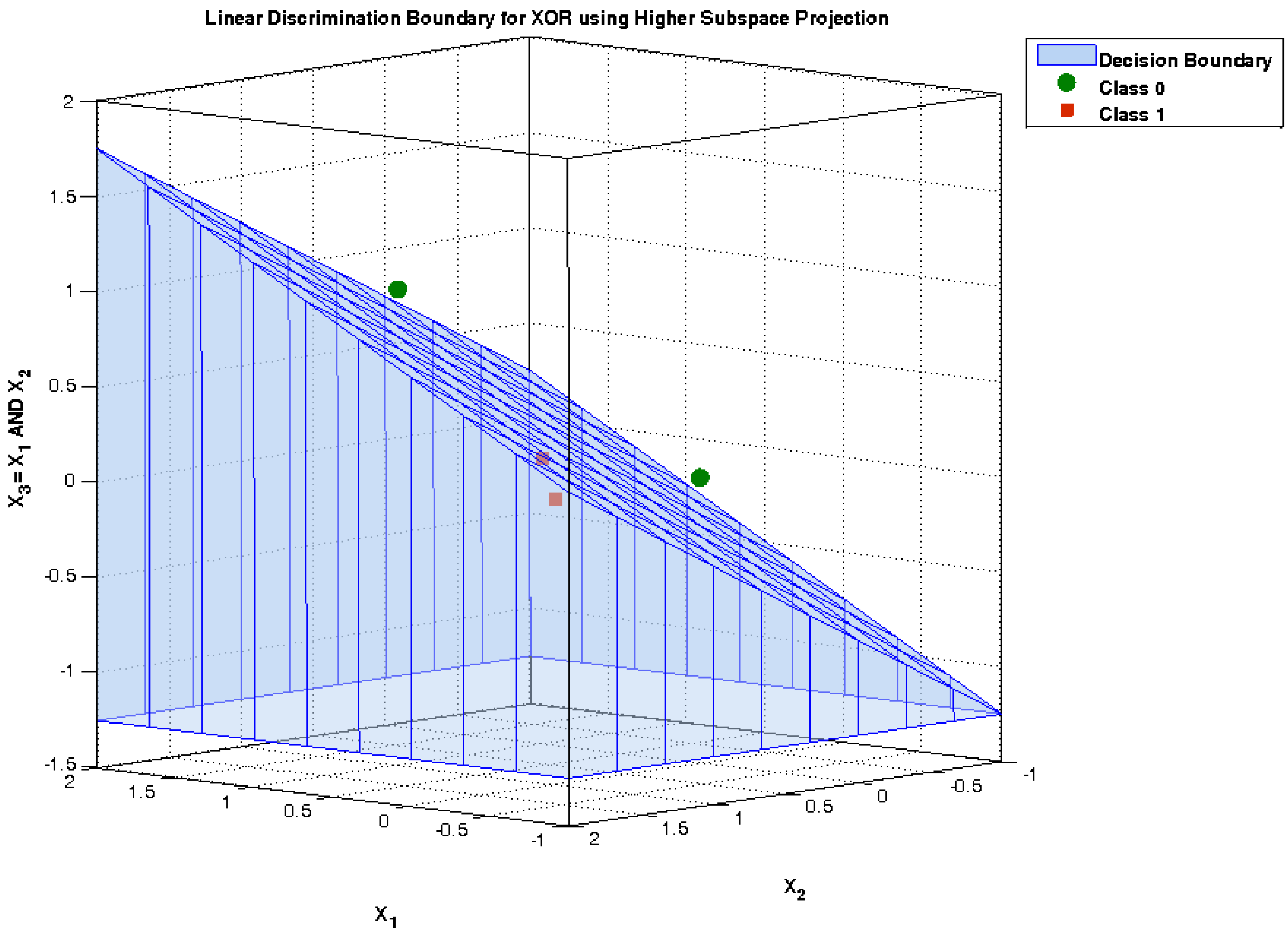
- Given a classification problem with a nonlinear boundary, we can, at times, find a mapping or transformation of the feature space which makes the classification problem linear separable in the transformed space



Examples: Transformation

- Let's define the mapping
  - $\boldsymbol{\phi}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$

XOR: Linear Separability



Examples: Transformation

– Does this mapping do it?

- $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\\ 1\end{bmatrix}$

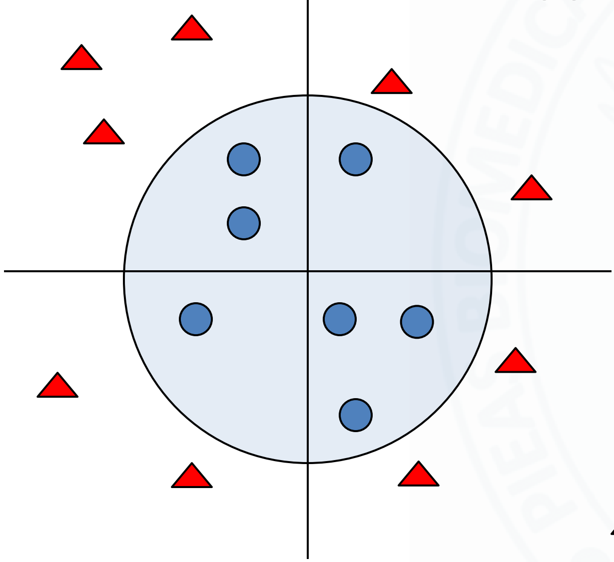
– What about this one?

- $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = (\boldsymbol{x_1} + \boldsymbol{x_2} - 1)^{\boldsymbol{2}}$

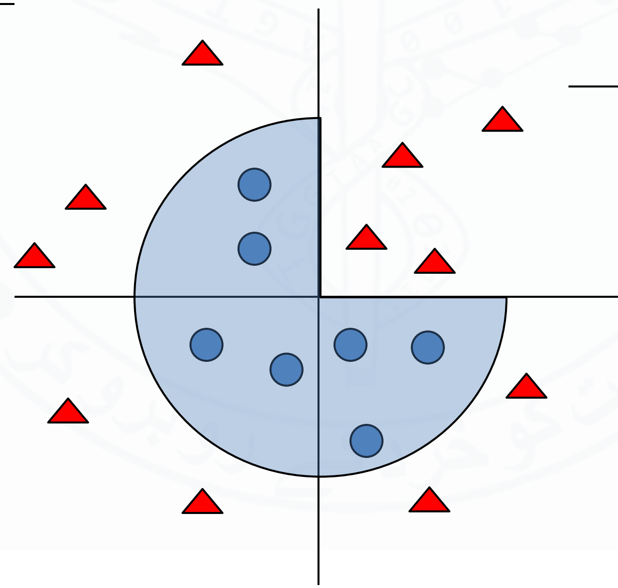
Transformation Examples

- Can you find a transform that makes the following classification problems linear separable? Can you draw the data points in the new transformed feature space?

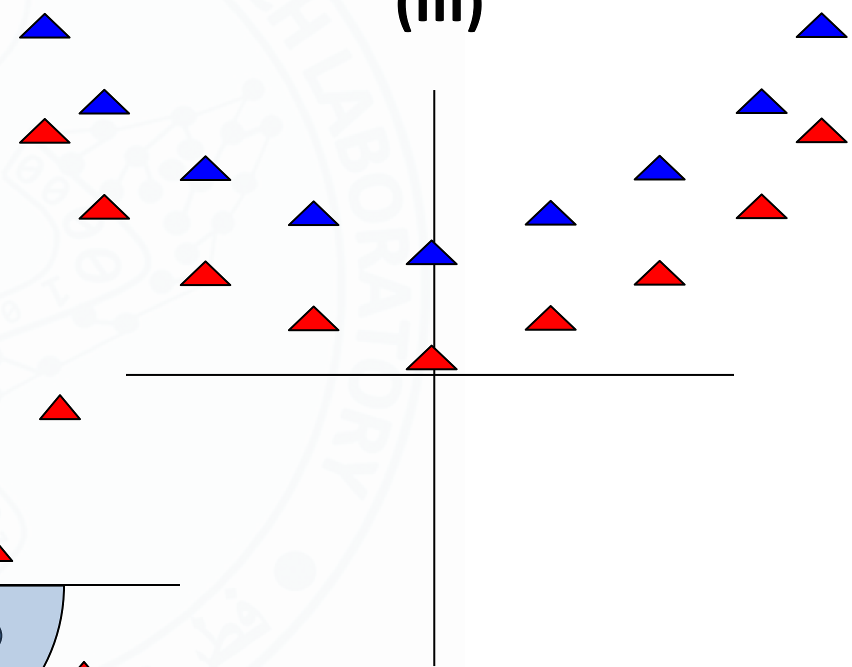
(I)



(II)



(III)



So How can we make a non-linear SVM?

- Transform the data and then apply the SVM!
- But defining transformations is hard
- A mathematical trick allows us to do this easily!
  - Kernel Trick!

Kernelized SVM

- We know that the discriminant function of the SVM can be written as:
  - $f(\boldsymbol{x}) = w^T\boldsymbol{x} + b$

- The Representer theorem (Scholkopf 2001) allows us to represent the SVM weight vector as a linear combination of input vectors with each example's contribution weighted by a factor α_i
  - $w = \sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i$

Kernelized SVM

- Thus, we can write
  - $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = b + \sum_{j=1}^{N} \alpha_j \boldsymbol{x}_j^T \boldsymbol{x}$
- Notice, that there is a dot product of the test example with every given input example. Let's denote the dot product as $k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}$
- This allows to write: $f(\boldsymbol{x}) = b + \boldsymbol{w}^T\boldsymbol{x} = b + \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x})$
- Now, $\boldsymbol{w}^T\boldsymbol{w} = \left(\sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i\right)^T \sum_{j=1}^{N} \alpha_j \boldsymbol{x}_j = \sum_{i,j=1}^{N} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Thus, the SVM can be rewritten as

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i$$

$$s.t. \quad y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0$$

$\longrightarrow$

$$min_{\alpha,b} \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + C\sum_{i=1}^{N} \xi_i$$

Such that for all $i = 1 \ldots N$ and $\xi_i \geq 0$:

$$y_i\left(b + \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i)\right) \geq 1 - \xi_i$$

Kernelized SVM

- In this formulation

$$min_{\boldsymbol{\alpha},b} \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + C\sum_{i=1}^{N} \xi_i$$

Such that for all $i = 1 \ldots N$ and $\xi_i \geq 0$:

$$y_i \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + b \geq 1 - \xi_i$$

- The weight vector is not present
- We know $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any two given training examples
- All the dot products have been replaced with $k(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- The optimization solution will be to obtain the $\boldsymbol{\alpha}$
- So if we can solve this problem
- So that we can test the patterns using: $f(\boldsymbol{x}) = b + \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x})$

Kernelized SVM

- Now the best thing about this formulation is that we can replace the regular Euclidean dot product with a dot product in a transformed space which will allow non-linear classification!
- It doesn't really matter what $k(u, v)$ is!
- Thus, instead of $k(u, v) = \boldsymbol{u}^T \boldsymbol{v}$
- We can have $k(u, v) = \boldsymbol{\phi}(u)^T \boldsymbol{\phi}(v)$
- As a matter of fact, we can define any similarity measure as k that satisfies certain properties called "Mercer's conditions"
- $k(\boldsymbol{u}, \boldsymbol{v})$ is called a kernel function and resulting SVM is called the kernelized SVM
  - It can solve classification problems that are not linearly separable

Kernel Functions ↔ Feature Transformation

- For the XOR problem we defined the transformation
  - $\boldsymbol{\phi}\left(\begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{x_1}\\ \boldsymbol{x_2}\\ \boldsymbol{x_1 x_2}\end{bmatrix}$
- We can thus define a kernel
  - $k\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)^{\boldsymbol{T}} \boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right) = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_2^{(i)} x_2^{(i)} \end{bmatrix} \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \\ x_2^{(i)} x_2^{(j)} \end{bmatrix}$
    $= x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(j)} + x_1^{(i)} x_2^{(i)} x_1^{(j)} x_2^{(j)}$
- Thus, the transformation produces a kernel
- Any valid kernel function, implicitly, transforms the examples into "some" feature space
  - The SVM can now found an optimal separating boundary in that space
  - That boundary can be non-linear in the original space

Some kernel functions

- We can use arbitrary kernels, for example …
  - The dot-product kernel
    - $\boldsymbol{K}(u, v) = \boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{v}$
      - Feature transform: $\boldsymbol{\phi}(\boldsymbol{u}) = \boldsymbol{u}$
  - The homogenous polynomial kernels
    - $\boldsymbol{K}(u, v) = \left(u^{\boldsymbol{T}} v\right)^{\boldsymbol{p}}$, $\boldsymbol{p}$ is called degree
      - For $\boldsymbol{p} = \boldsymbol{2}$ and 2D data, $\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$: $\boldsymbol{\phi}(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ \sqrt{2}u_1 u_2 \end{bmatrix}$
  - The Radial Basis Function (RBF) Kernel
    - $\boldsymbol{K}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{e}^{-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2}}$, $\boldsymbol{\sigma}$ controls the spread of the Gaussian
      - Feature transform?
- The RBF and Homogenous Polynomial kernels implement non-linear boundaries

Example

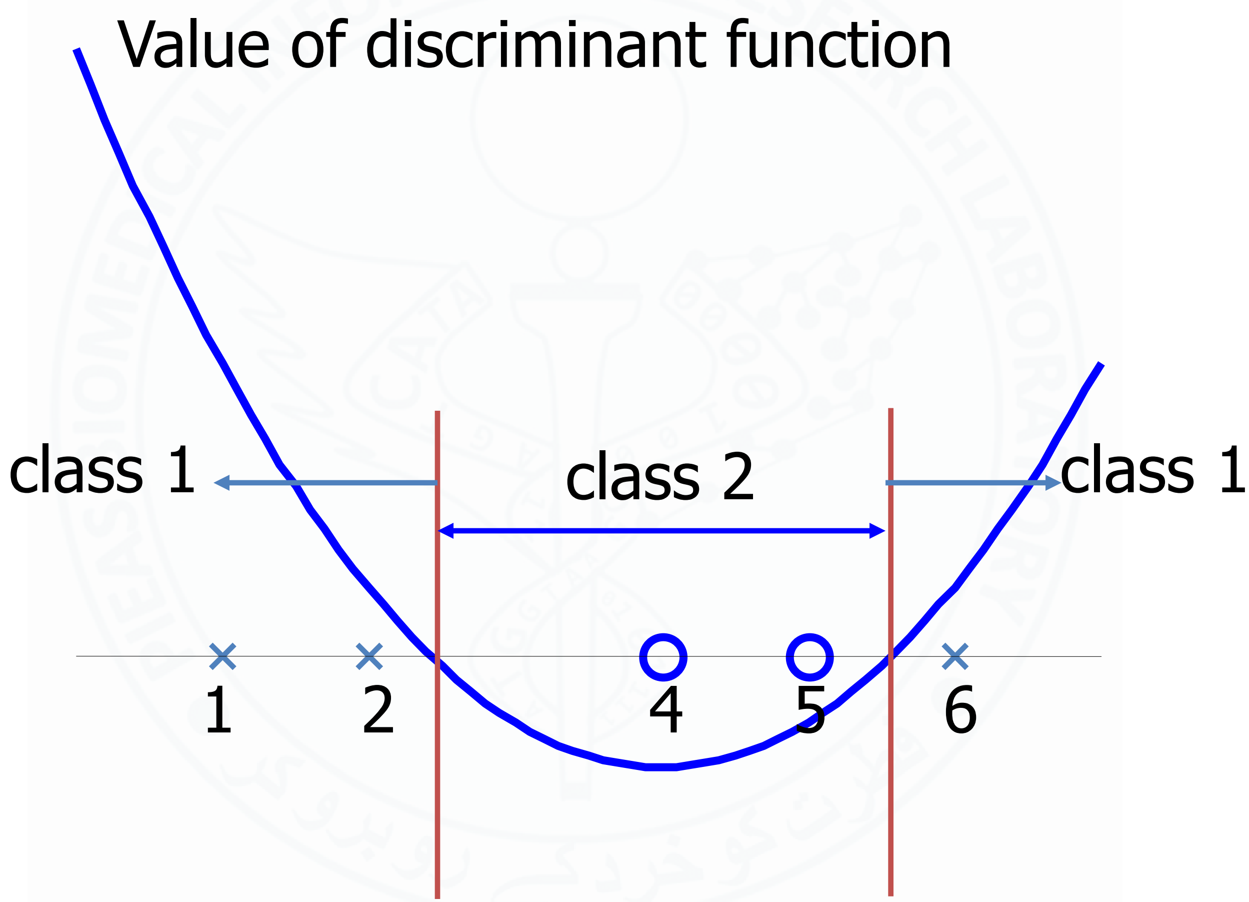
- Suppose we have 5 1D data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - $K(u,v) = (uv+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5)

Example

- By using a QP solver, we get
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - Note that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
  - The bias turns out to be b = 9
- The discriminant function is

$$f(z) = 0.6667z^2 - 5.333z + 9$$

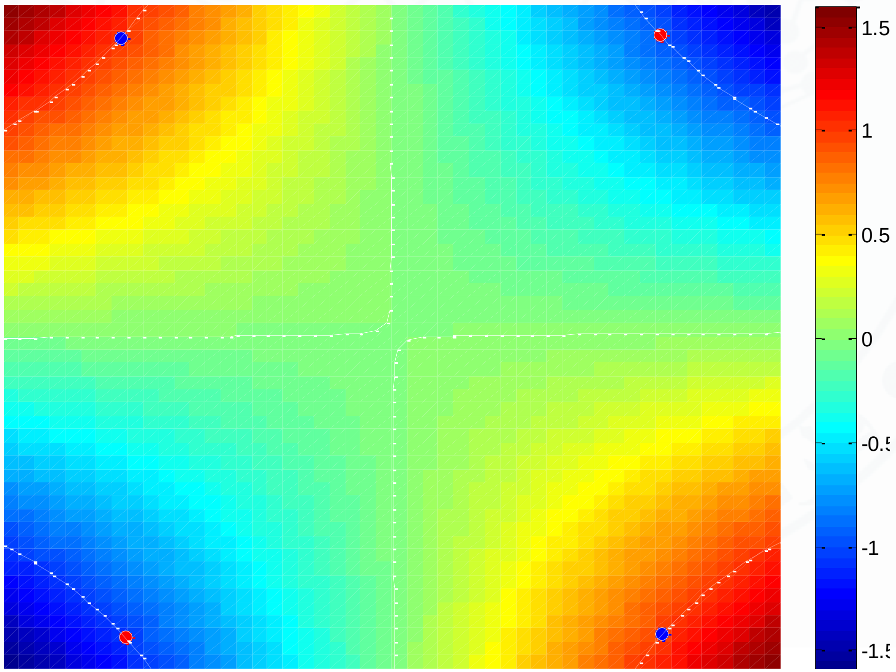
Example



Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

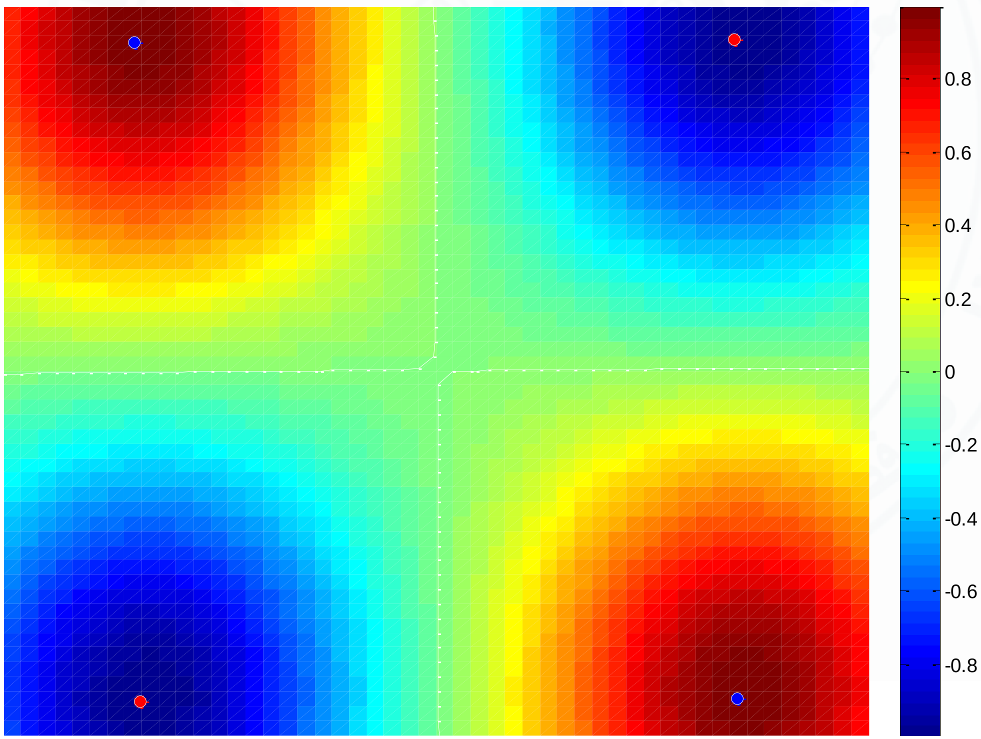
Solving the XOR

- C=1
- With polynomial kernel of degree 2



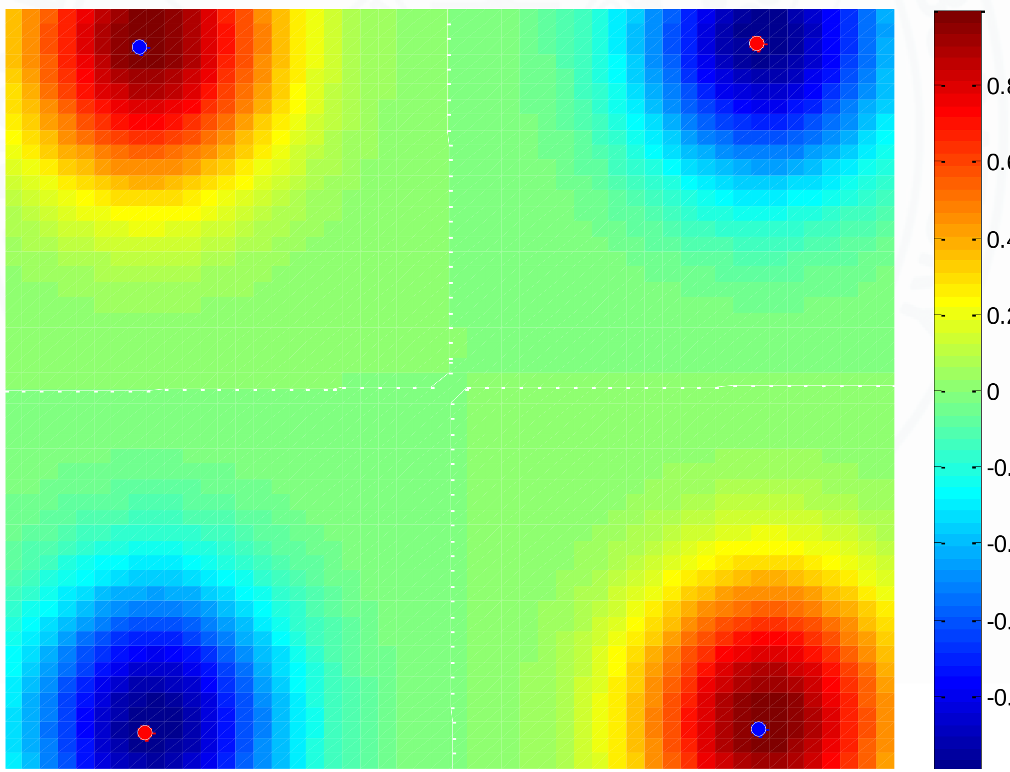
Solving the XOR

- C=1
- With RBF Kernel (sigma = 0.5)

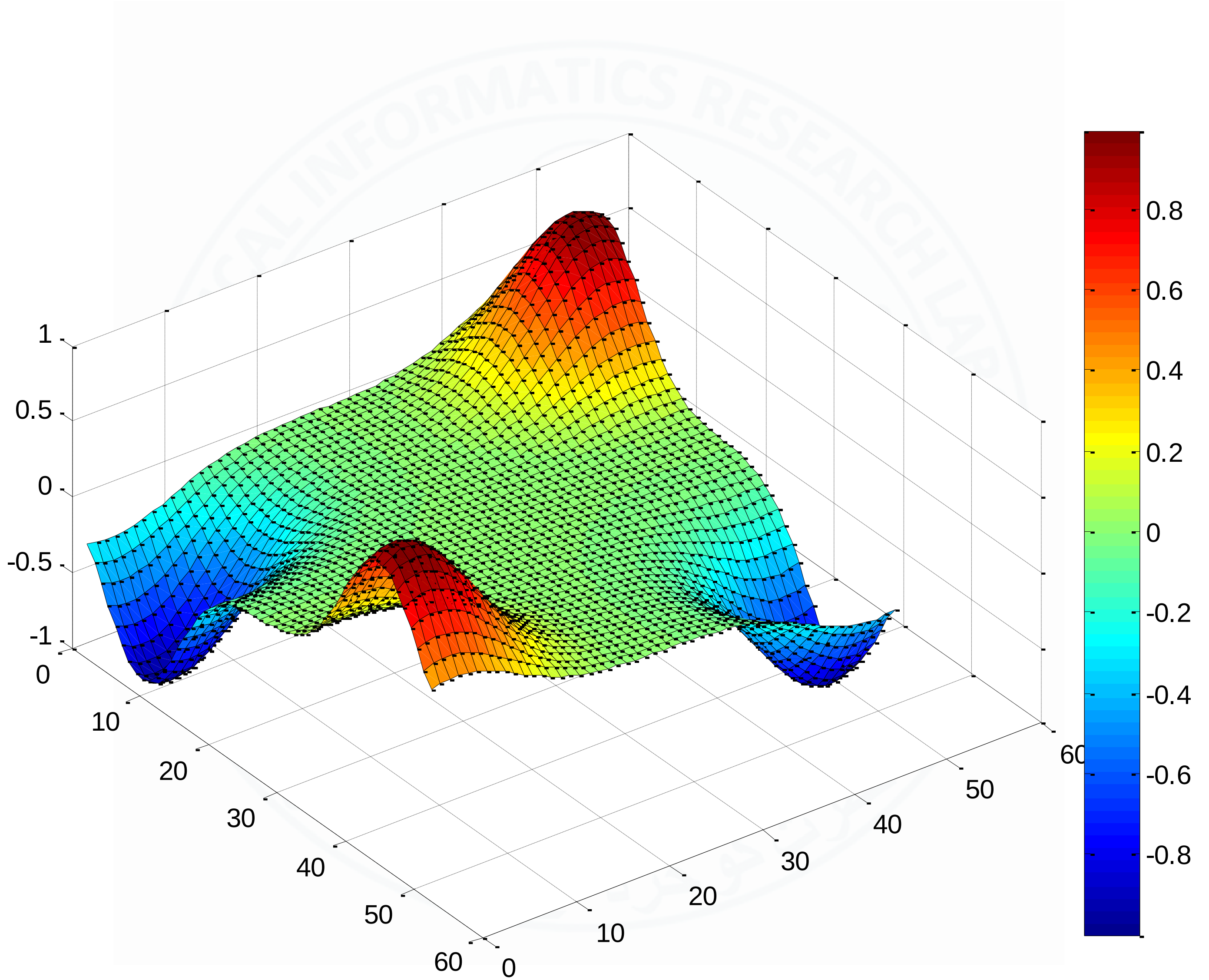


Solving the XOR

- C=1
- With RBF Kernel (sigma = 0.3)



3D Plot



1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
0
10
20
30
40
50
60
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

Using the SVM

- Read:

- Ben-Hur, Asa, and Jason Weston. 2010. “A User’s Guide to Support Vector Machines.” In *Data Mining Techniques for the Life Sciences*, edited by Oliviero Carugo and Frank Eisenhaber, 223–39. Methods in Molecular Biology 609. Humana Press. http://dx.doi.org/10.1007/978-1-60327-241-4_13

- http://pyml.sourceforge.net/doc/howto.pdf

-

Steps for Feature based Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

Choosing C

- Cross-validation
  - To assess how good a classifier is we can use cross-validation
    - Divide the data randomly into k parts
      - If the data is imbalanced, use stratified sampling
    - Use k-1 parts for training
    - And the held-out part for testing to evaluate accuracy or ROC curve or other performance metrics
- To choose C, you can do nested cross-validation

Handling data imbalance

- If the data is imbalanced (too much of one class and only a small number of examples from the other)
  - You can set an individual C for each example
  - Can also be used to reflect a priori knowledge

Strengths and Weaknesses of SVM

- Strengths
  - Only a few training points determine the final boundary
    - Support Vectors
  - Margin maximization and kernelized
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly (through C)
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need to choose a “good” kernel function.

Advantages of kernels

- Once we replace the dot product with a kernel function (i.e., perform the kernel trick or ‘kernelize’ the formulation), the SVM formulation no longer requires any features!
- As long as you have a kernel function, everything works
  - Remember a kernel function is simply a mapping from two examples to a scalar
    - Tells us how similar the two examples are to each other

But how is that an advantage?

- When the number of dimensions is very large, an implicit representation through a kernel is helpful
- Let’s say we have a document classification problem
  - We define a M-dimensional feature vector for each document that indicates if it has any of the pre-specified number of words in it (1) or not (0)
  - M can be very large (equal to the size of the dictionary)
  - The dot product of two feature vectors is equal to the number of common words between the two documents
  - Why not simply count the number of words?
    - We can now do that with the kernel trick

Sufficient Conditions for a kernel function

- Definition of an inner product
  - Symmetry: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$
  - Linearity: $\langle \alpha \boldsymbol{x} + \beta \boldsymbol{y}, \boldsymbol{z} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \beta \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
    - Principle of superposition
  - Positive Definiteness
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq \mathbf{0}$
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \mathbf{0}$ iff $\boldsymbol{x} = \mathbf{0}$
- These conditions need to be satisfied by the kernel function too
  - The kernel matrix must be symmetric, positive semi-definite
  - This requirement is called the **Mercer's condition**

Kernel Construction

Proposition 3.22 (Closure properties) *Let κ_1 and κ_2 be kernels over $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ a real-valued function on X, ϕ: $X \longrightarrow \mathbb{R}^N$ with κ_3 a kernel over $\mathbb{R}^N \times \mathbb{R}^N$, and $\mathbf{B}$ a symmetric positive semi-definite $n \times n$ matrix. Then the following functions are kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$,
(ii) $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$,
(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$,
(iv) $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,
(v) $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\phi(\mathbf{x}),\phi(\mathbf{z}))$,
(vi) $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z}$.

Proposition 3.24 *Let $\kappa_1(\mathbf{x}, \mathbf{z})$ be a kernel over $X \times X$, where $\mathbf{x}$, $\mathbf{z} \in X$, and $p(x)$ is a polynomial with positive coefficients. Then the following functions are also kernels:*

(i) $\kappa(\mathbf{x}, \mathbf{z}) = p(\kappa_1(\mathbf{x}, \mathbf{z}))$,
(ii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(\kappa_1(\mathbf{x}, \mathbf{z}))$,
(iii) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(-\left\|\mathbf{x} - \mathbf{z}\right\|^2 / (2\sigma^2))$.

SVM in Scikit Learn

- http://scikit-learn.org/stable/modules/svm.html

End of Lecture

We want to make a machine that will be proud of us.

\- Danny Hillis