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/

Preliminaries: Introduction to COP

- The constrained optimization problem (COP) can be expressed in its general form as follows

$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$

subject to

$$g_i(\mathbf{x}) \le 0 \qquad i = 1 \ldots m$$

$$h_i(\mathbf{x}) = 0 \qquad i = 1 \ldots p$$

- Example

Constrained Optimization: Example

- You are given a string of length L. You are to tie it around a rectangular gift box (l x w x h) with a constant or fixed width ‘w’ using any length of this string (up to L).
- Can you find the dimensions (length and height only since the width is fixed) of the box with the largest volume that you can tie with this string?
- Mathematically
  - Optimize the objective function
    - Max V=wlh
  - Subject to constraints
    - $2(l+h) \leq L$

Lagrangian Formulation

- Lagrange proposed a method for the solution of COP

$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$

subject to

$$g_i(\mathbf{x}) \le 0 \qquad i = 1 \ldots m$$

$$h_i(\mathbf{x}) = 0 \qquad i = 1 \ldots p$$

  - f(x) and $g_i(x)$ are convex functions
  - $h_i(x)$ are affine functions

Lagrangian Formulation…

- The solution proposed by Lagrange is based on the following unconstrained minimization

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \sum_{i=1}^{m} L_i^g(\mathbf{x}) + \sum_{i=1}^{p} L_i^h(\mathbf{x})$$

- Where $L_i^g(\mathbf{x})$ and $L_i^h(\mathbf{x})$ have the following properties

$$L_i^g(\mathbf{x}) = \begin{cases} \infty & g_i(\mathbf{x}) > 0 \\ 0 & else \end{cases}, \qquad i = 1 \ldots m$$

$$L_i^h(\mathbf{x}) = \begin{cases} \infty & h_i(\mathbf{x}) \neq 0 \\ 0 & else \end{cases}, \qquad i = 1 \ldots p$$

  - Large penalties added when the constraints are not satisfied
    - Unconstrained optimization now leads to satisfaction of the constrains and then optimization of the original objective function

Lagrangian Formulation…

- One possible way of achieving the above mentioned properties for the two penalty functions is as follows

$$L_i^g(\mathbf{x}) = \max_{\alpha_i \geq 0} \quad \alpha_i g_i(\mathbf{x})$$

When the constraint is violated ($g_i(x)>0$) the maximization with respect to α_i leads to infinity as long as α_i is non-negative

Similarly

$$L_i^h(\mathbf{x}) = \max_{\beta_i} \quad \beta_i h_i(\mathbf{x})$$

When the constraint is not violated the maximization with respect to α_i leads to zero as long as α_i is non-negative

Lagrangian Formulation…

- Thus we can write the optimization problem as

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \sum_{i=1}^{m} \max_{\alpha_i \geq 0} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \max_{\beta_i} \beta_i h_i(\mathbf{x})$$

$$\equiv \min_{\mathbf{x}} \quad f(\mathbf{x}) + \max_{\alpha_i \geq 0, \beta_i} \left(\sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \beta_i h_i(\mathbf{x}) \right)$$

$$\equiv \min_{\mathbf{x}} \max_{\alpha_i \geq 0, \beta_i} \quad f(\mathbf{x}) + \left(\sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \beta_i h_i(\mathbf{x}) \right)$$

- α_i and β_i are called Lagrange multipliers (or dual variables) and the function (below) is called the Lagrange Function

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \left(\sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \beta_i h_i(\mathbf{x}) \right)$$

Lagrangian function: Gift example

- The problem can be rewritten as
  - Min f(x) = $-x_1x_2$
  - Subject to constraints

    $2(x_1+x_2) \leq L$, OR

    $g(x) = 2(x_1+x_2) - L \leq 0$

- This implies

$$L(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \alpha_1 g_1(\mathbf{x})$$
$$= -x_1x_2 + \alpha_1\left(2(x_1+x_2) - L\right)$$

Lagrangian function: example…

- This can be solved as

$$\min_{\mathbf{x}} \max_{\alpha_i \geq 0} \quad L(x_1, x_2, \alpha) = -x_1 x_2 + \alpha_1 \left(2(x_1 + x_2) - L\right)$$

$$\frac{\partial L}{\partial \alpha} = 2(x_1 + x_2) - L = 0 \quad \Rightarrow \quad x_1 + x_2 = L/2$$

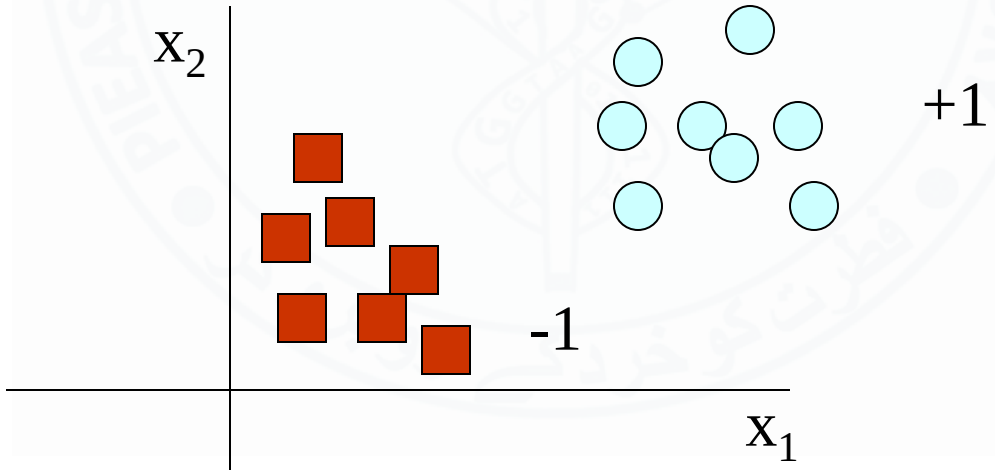
$$\frac{\partial L}{\partial x_1} = -x_1 + 2\alpha_1 = 0 \quad \Rightarrow \quad x_1 = 2\alpha_1$$

$$\frac{\partial L}{\partial x_2} = -x_2 + 2\alpha_1 = 0 \quad \Rightarrow \quad x_2 = 2\alpha_1$$

$$\Rightarrow \quad x_1 = x_2 = L/4$$

Classification

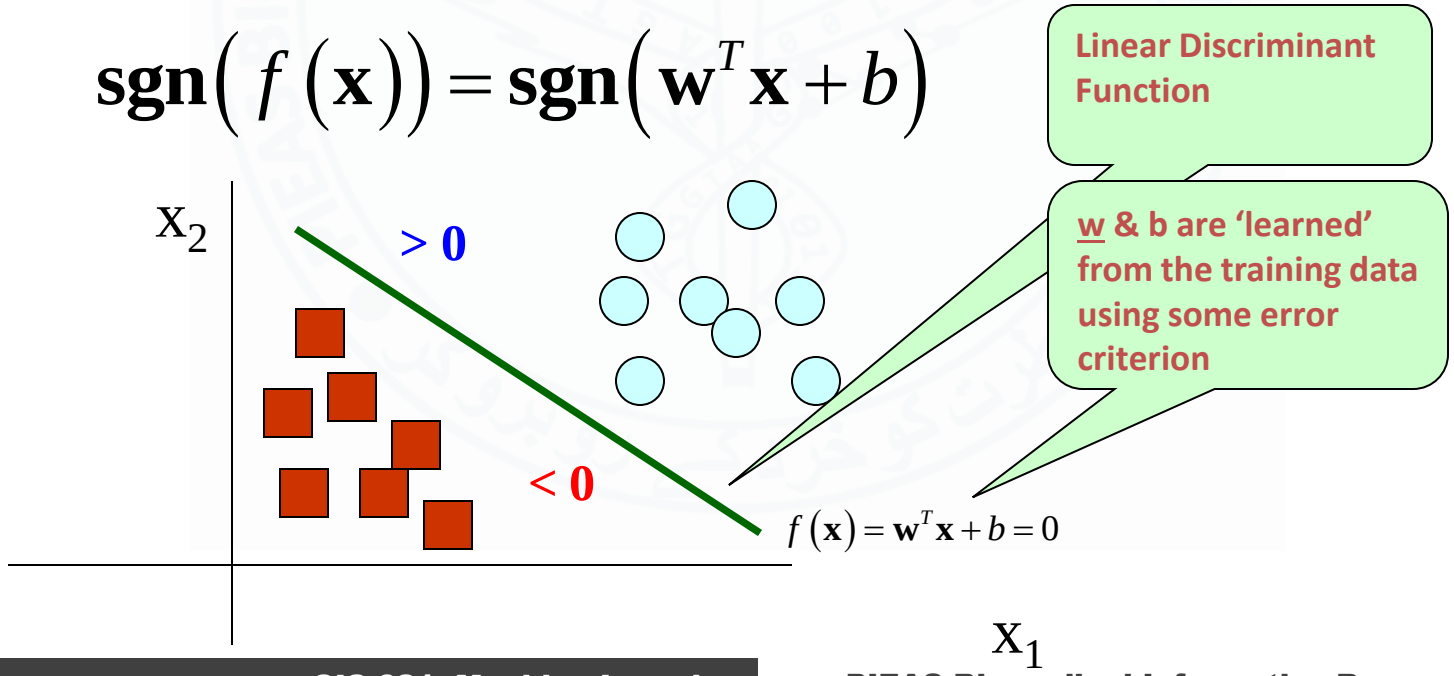
- Before moving on with the discussion let us restrict ourselves to the following problem
  - T = Given Training Set = $\{(\underline{x}^{(i)}, y_i), i = 1 \ldots N\}$
    - $\underline{x}^{(i)} \varepsilon R^m$ {Data Point i }
    - y_i: class of data point i (+1 or -1)



Use of Linear Discriminant in Classification

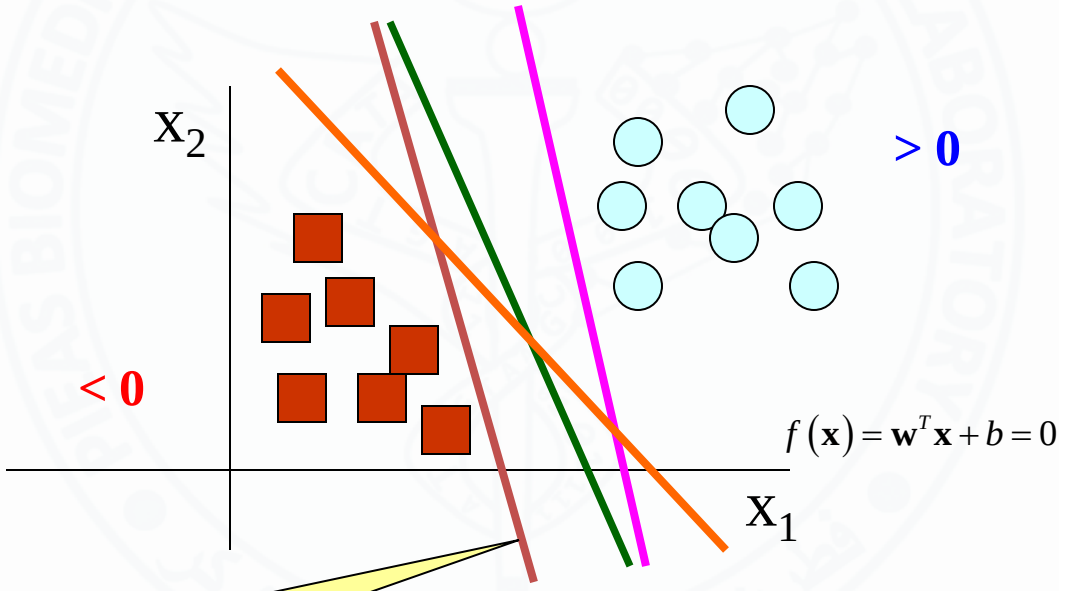
- Classifiers such as the Single Layer Perceptron (with linear activation function) and SVM use a linear discriminant function to differentiate between patterns of different classes
- The linear discriminant function is given by

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

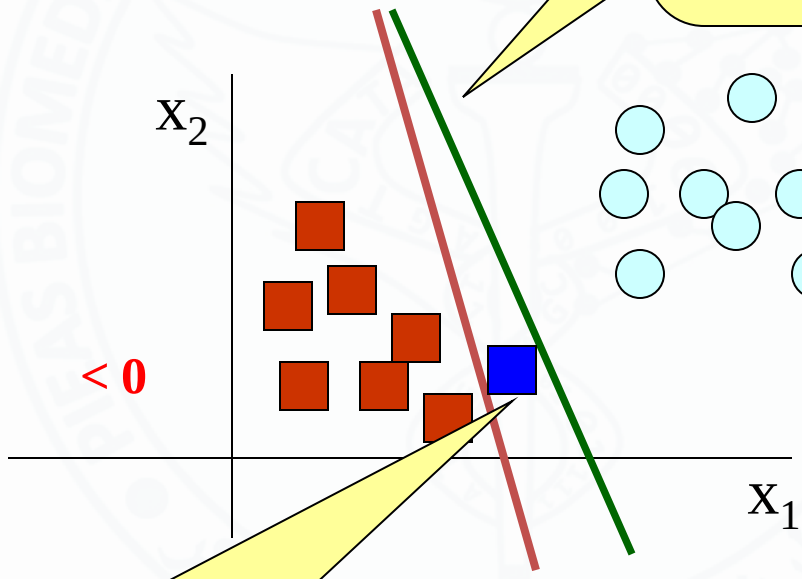


Use of Linear Discriminant in Classification

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

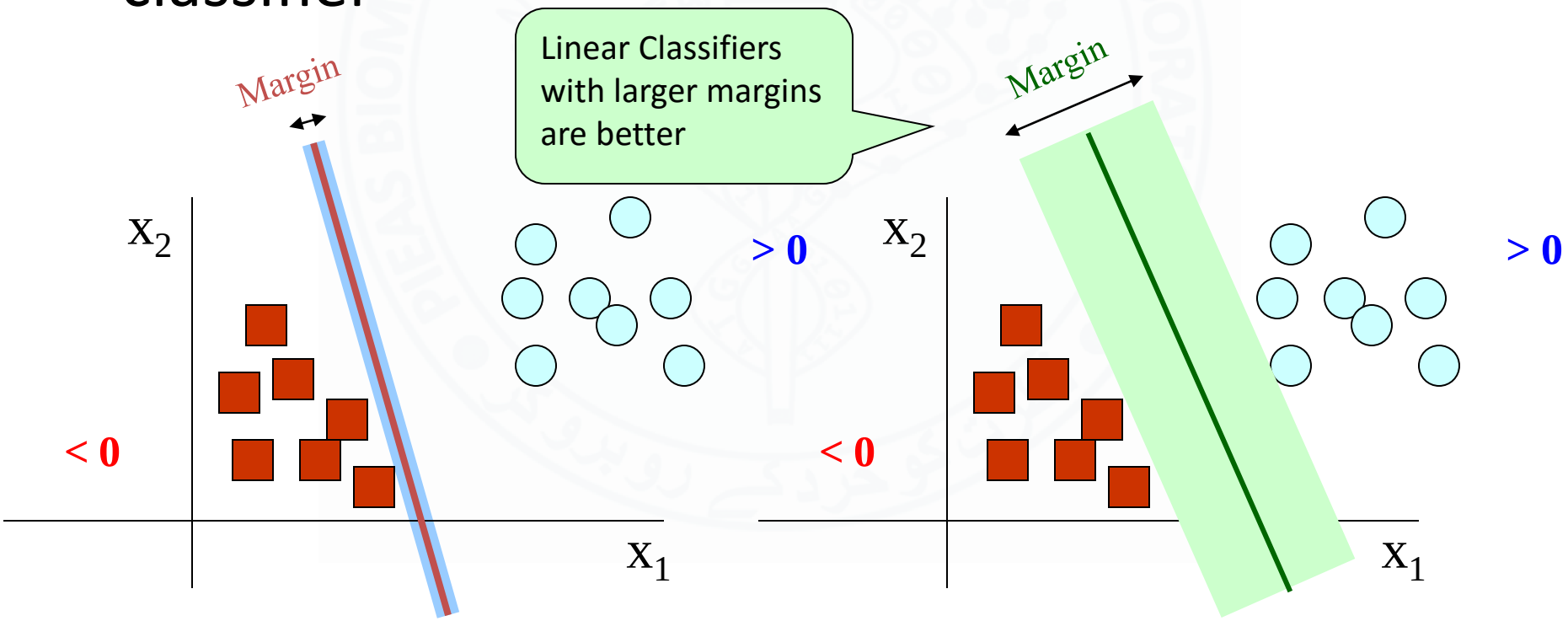


Use of Linear Discriminant in Classification



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)

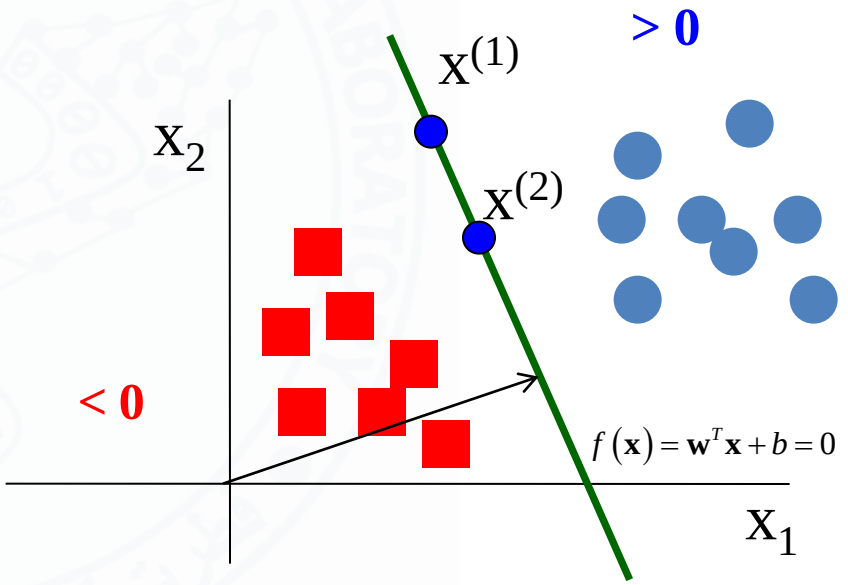
- Support Vector Machines are linear classifiers that produce the optimal separating boundary (hyper-plane)
  - Find **w** and b in a way so as to maximize the margin while classifying all the training patterns correctly (for linearly separable problem)

Finding Margin of a Linear Classifier

- Consider a linear classifier with the boundary

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

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



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

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

Subtracting (1) from (2)

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

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,

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

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

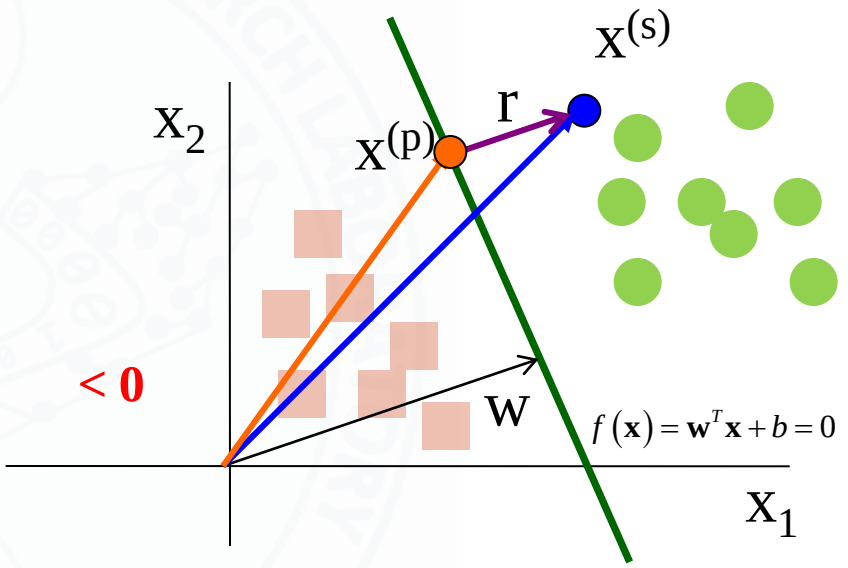
$$= \mathbf{w}^T\left(\mathbf{x}^{(p)} + r\hat{\mathbf{w}}\right) + b$$

$$= \mathbf{w}^T\mathbf{x}^{(p)} + r\mathbf{w}^T\hat{\mathbf{w}} + b$$

$$= \mathbf{w}^T\mathbf{x}^{(p)} + b + r\mathbf{w}^T\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$= 0 + r\|\mathbf{w}\| = r\|\mathbf{w}\|$$

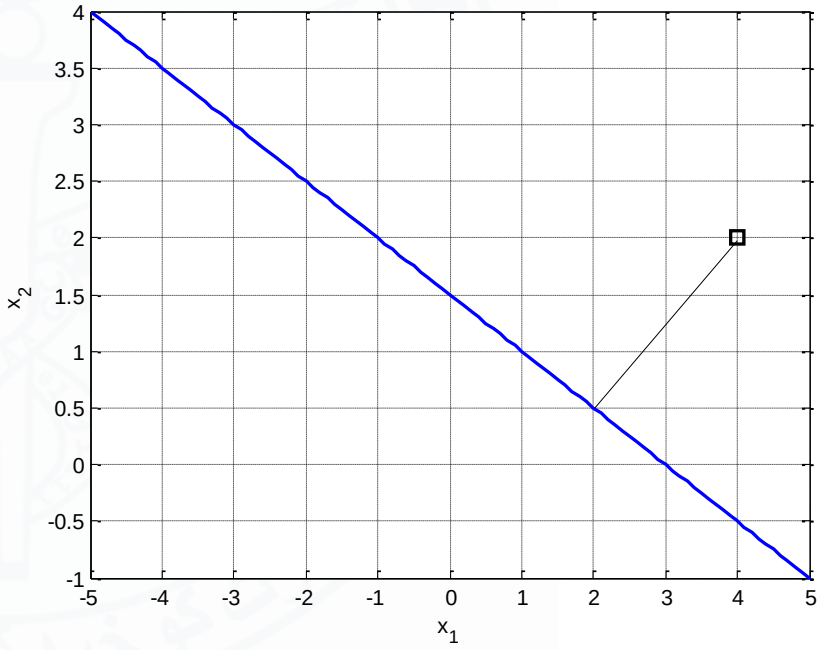
$$\Rightarrow r = \frac{f\left(\mathbf{x}^{(s)}\right)}{\|\mathbf{w}\|}$$



Perpendicular Distance of a point from a line

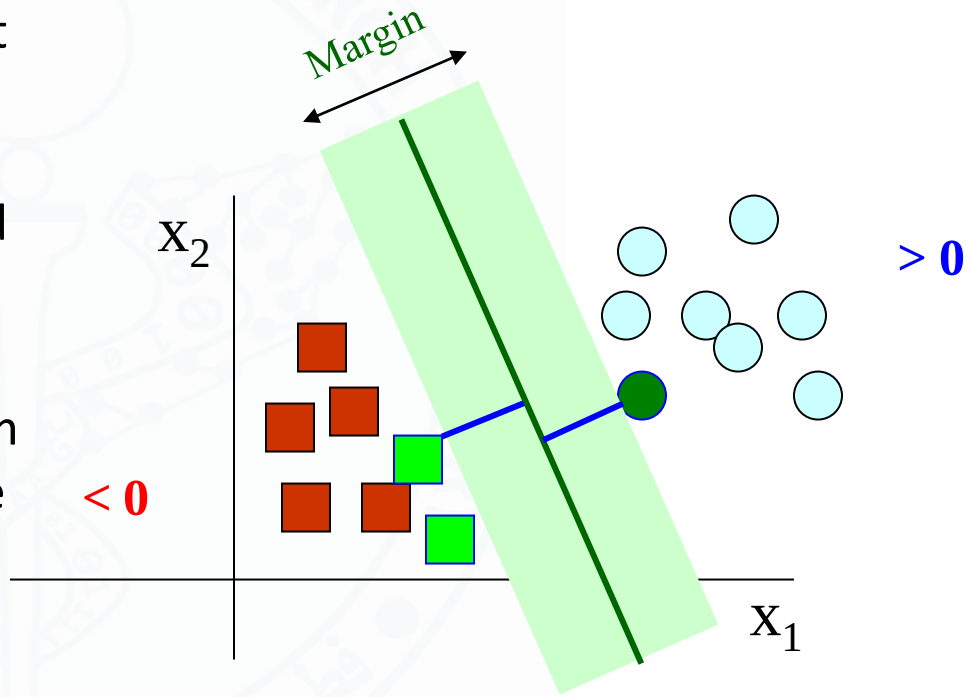
Example

- 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

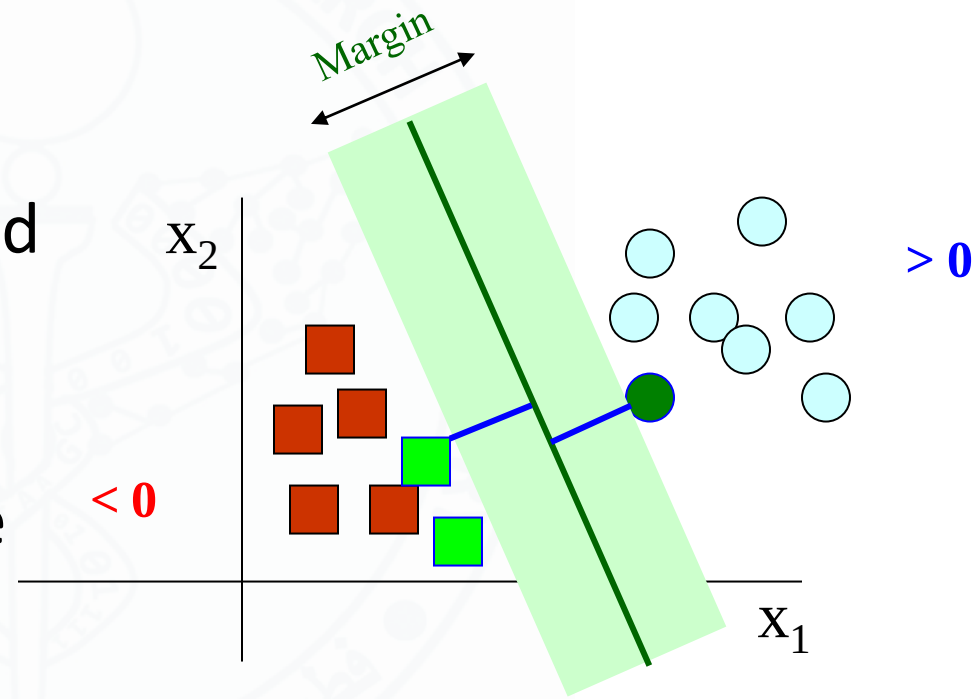


$$\rho = 2|r| = 2\left|\frac{f(\mathbf{x}^*)}{\|\mathbf{w}\|}\right|$$

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 $\{(x^{(i)}, y_i)\}$

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

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

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

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

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

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

$$s.t.$$

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

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

OR

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

$$s.t.$$

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

SVM as an Optimization problem

- Combining the objective function and the constraints (represented as losses) as follows

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

Produces a large +ive value when the ith constraint is violated

- The constraint cost function can be written as

$$L_i = \max_{\alpha_i} \quad \alpha_i \left\{1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\} = \begin{cases} 0 & 1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \le 0 \\ \infty & else \end{cases}$$

$$\alpha_i \ge 0$$

When the constraint is being strictly satisfied, α_i must be zero

α_i must be positive when the constraint is being violated

SVM as an Optimization problem

- Combining

$$Let \quad P = \min_{\mathbf{w},b}\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\left[\max_{\alpha_i \geq 0}\alpha_i\left\{1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\}\right]\right\}$$

$$= \min_{\mathbf{w},b}\max_{\alpha_i \geq 0}\underbrace{\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\left[\alpha_i\left\{1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\}\right]\right\}}$$

This is the Lagrangian (α_i are the Lagrange Multipliers)

- This is called the primal form of the optimization problem
- The corresponding dual can be written as

$$Let \quad D = \max_{\alpha_i \geq 0}\min_{\mathbf{w},b}\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\left[\alpha_i\left\{1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\}\right]\right\}$$

SVM as an Optimization problem

- Since the optimization is convex therefore if the Karush-Kuhn-Tucker conditions are satisfied then the primal and dual optimal values will be equal
- In simple words the optimization problem in SVM can be interpreted as

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w},b} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \left[\alpha_i \left\{ 1 - y_i \left(\mathbf{w}^T\mathbf{x}^{(i)} + b \right) \right\} \right] \right\}$$

$$s.t. \qquad \alpha_i \geq 0$$

SVM as an Optimization problem

$$\max_{\alpha_i \geq 0} \min_{\mathbf{w},b} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \left[\alpha_i \left\{ 1 - y_i \left(\mathbf{w}^T \mathbf{x}^{(i)} + b \right) \right\} \right] \right\} = \max_{\alpha_i \geq 0} \left\{ \theta_D(\mathbf{w},b) \right\}$$

$$s.t. \qquad \alpha_i \geq 0$$

$$\theta_D(\mathbf{w},b) = \min_{\mathbf{w},b} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \left[\alpha_i \left\{ 1 - y_i \left(\mathbf{w}^T \mathbf{x}^{(i)} + b \right) \right\} \right] \right\} \qquad [1]$$

Finding the saddle point of $\theta_D(\mathbf{w},b)$

$$\frac{\partial \theta_D(\mathbf{w},b)}{\partial \mathbf{w}} = \mathbf{w} + \sum_{i=1}^{N} \left[\alpha_i \left\{ -y_i \mathbf{x}^{(i)} \right\} \right] = 0 \qquad \Rightarrow \mathbf{w}^* = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}^{(i)}$$

$$\frac{\partial \theta_D(\mathbf{w},b)}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

SVM as an Optimization problem

Putting in $[1]$

$$\theta_D(\mathbf{w}^*, b^*) = \frac{1}{2}\mathbf{w}^{*T}\mathbf{w}^* + \sum_{i=1}^{N}\left[\alpha_i\left\{1 - y_i\left(\mathbf{w}^{*T}\mathbf{x}^{(i)} + b^*\right)\right\}\right]$$

$$= \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}^{(i)}\right)^T\left(\sum_{j=1}^{N}\alpha_j y_j \mathbf{x}^{(j)}\right) + \sum_{i=1}^{N}\left[\alpha_i\left\{1 - y_i\left(\left(\sum_{j=1}^{N}\alpha_j y_j \mathbf{x}^{(j)}\right)^T\mathbf{x}^{(i)} + b^*\right)\right\}\right]$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i y_i \mathbf{x}^{(i)^T}\alpha_j y_j \mathbf{x}^{(j)} + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\left[\alpha_i y_i\left(\sum_{j=1}^{N}\alpha_j y_j \mathbf{x}^{(j)}\right)^T\mathbf{x}^{(i)} + b^*\right]$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}^{(i)^T}\mathbf{x}^{(j)} + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}^{(i)^T}\mathbf{x}^{(j)} + b^*\sum_{i=1}^{N}\alpha_i y_i$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}^{(i)^T}\mathbf{x}^{(j)}$$

SVM as an Optimization problem

- Thus the problem can be written as

$$\max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} \right\}$$

$$s.t \qquad \alpha_i \geq 0$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

- This quadratic optimization problem can be solved for α_i using standard optimization packages

SVM as an Optimization problem

- The training points with their corresponding α_i greater than zero lie on the boundary and are thus called support vectors as they support the boundary
- Once α_i have been found, the weight can be calculated as

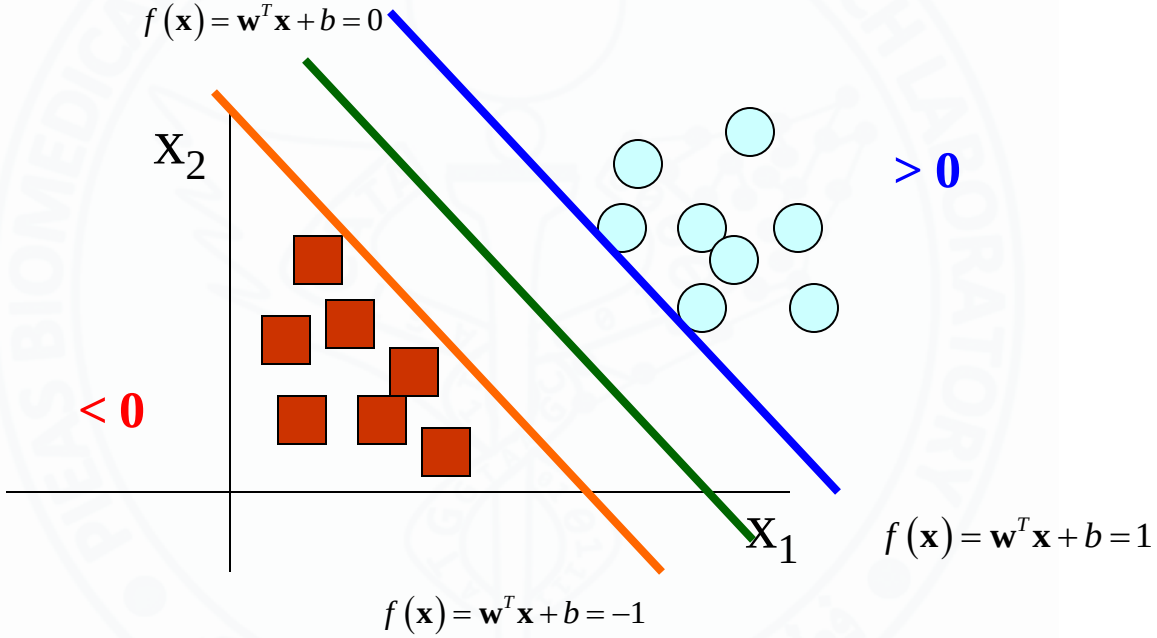
$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}^{(i)}$$

$$\mathbf{w}^{*T}\mathbf{x} + b^* = y \quad \Rightarrow \quad b^* = y - \mathbf{w}^{*T}\mathbf{x}$$

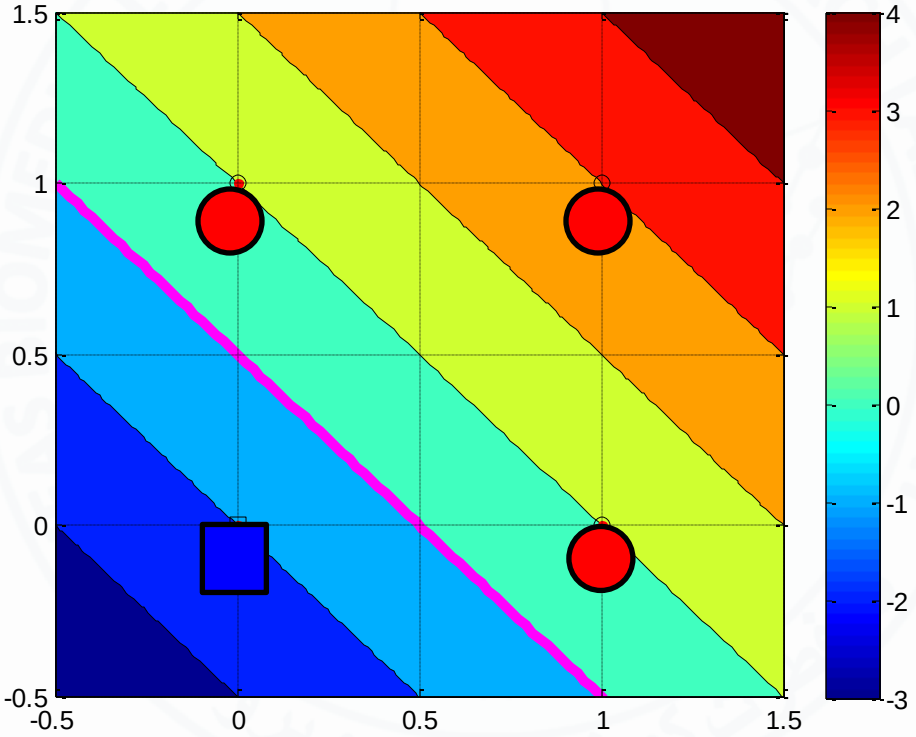
$$or \qquad b^* = \frac{1}{n_{SV}} \sum_{\alpha_i > 0} \left(y_i - \mathbf{w}^{*T}\mathbf{x}^{(i)} \right)$$

- Classification
  - The label of an unknown point can be determined by

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


$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$
X_2
> 0
< 0
X_1
$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 1$
$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = -1$

Example problem



Matrix formulation of SVM Problem

$$\max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} \right\}$$

$$s.t \quad \alpha_i \geq 0$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$max_{\boldsymbol{\alpha}} \; \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\alpha}$$

Subject to:

$$\boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$

$$\boldsymbol{y}^T \boldsymbol{\alpha} = 0$$

- If we define the following:
  - $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_N]^T$
  - $\boldsymbol{y} = [y_1 \quad y_2 \quad \dots \quad y_N]^T$
  - $\mathbf{1}_N = [1 \quad 1 \quad \dots \quad 1]^T$
  - $\boldsymbol{X}_{(d \times N)} = [\boldsymbol{x_1} y_1 \quad \boldsymbol{x_2} y_2 \quad \dots \quad \boldsymbol{x_N} y_N]$

Solving the SVM using QP

- CVXOPT is a Python package that implements a quadratic programming solver
  - http://abel.ee.ucla.edu/cvxopt/userguide/coneprog.html#quadratic-programming
- cvxopt.solvers.qp(*P*, *q*[, *R*, *s*[, *U*, *v*[, *solver*[, *initvals*]]]])
  - Solves the following problem for $\boldsymbol{z}$:

$$min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T\boldsymbol{P}\boldsymbol{z} + \boldsymbol{q}^T\boldsymbol{z}$$

Subject to:

$$\boldsymbol{R}\boldsymbol{z} \preccurlyeq \boldsymbol{s}$$
$$\boldsymbol{U}\boldsymbol{z} = \boldsymbol{v}$$

$\boldsymbol{z} = \boldsymbol{\alpha}$
$\boldsymbol{P} = \boldsymbol{X}^T\boldsymbol{X}$
$\boldsymbol{q} = -\mathbf{1}_N$
$\boldsymbol{R} = -\boldsymbol{I}_{N\times N}$
$\boldsymbol{s} = \mathbf{0}_N$
$\boldsymbol{U} = \boldsymbol{y}^T$
$\boldsymbol{v} = 0$

SVM Problem

$$max_{\boldsymbol{\alpha}}\ \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\alpha}$$

Subject to:

$$\boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$
$$\boldsymbol{y}^T\boldsymbol{\alpha} = 0$$

Programming the SVM

Example: Solution of the OR problem

Optimal separating boundary

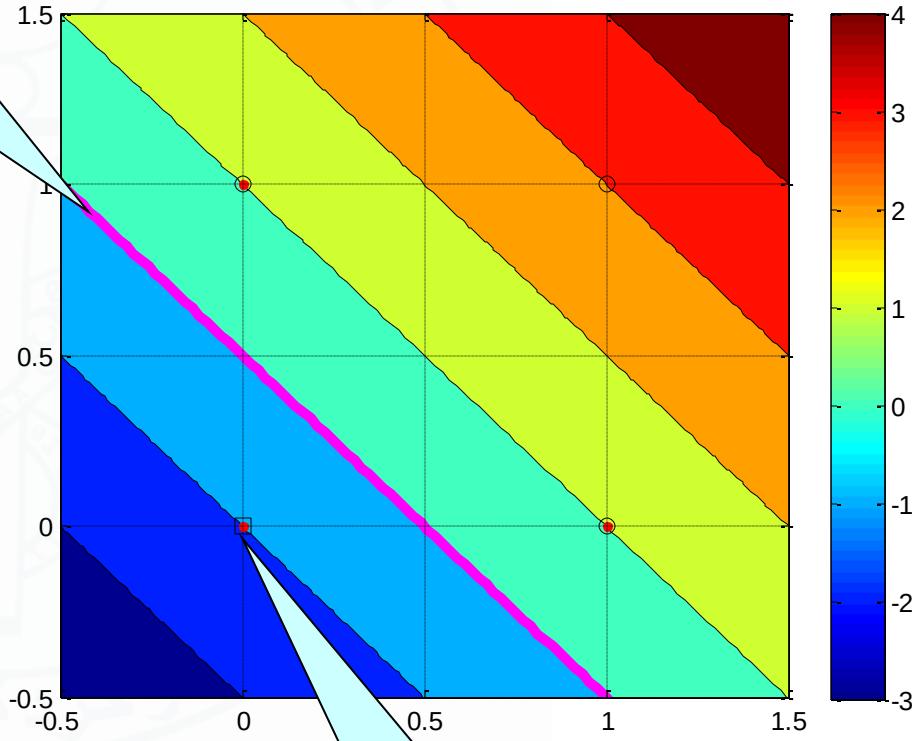
$$\mathbf{x}^i = \begin{bmatrix}0\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix}$$

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

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

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

$$b^* = -1$$



One of the 3 SVs

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

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

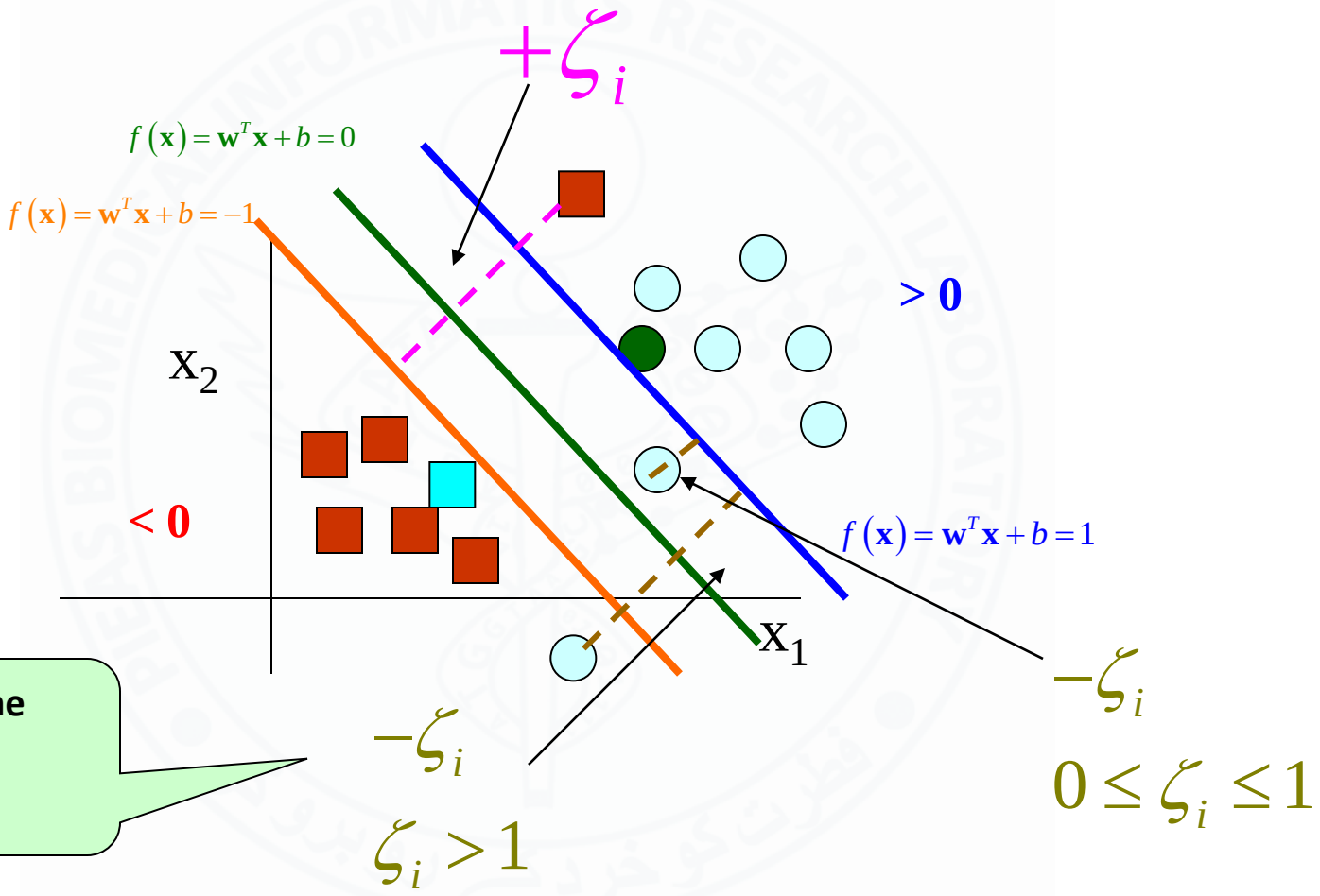
- This is achieved as

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

$$\zeta_i\geq 0$$

Slack Variables

Handling Non-Separable Patterns (Soft Margin)



Handling Non-Separable Patterns (Soft Margin)

- Our objective in designing a SVM here is to maximize the margin while minimizing the misclassification error
- The misclassification error can be written as:

$$\Phi(\zeta) = \sum_{i=1}^{N} I(\zeta_i - 1) \qquad I(\zeta) = \begin{cases} 0 & \zeta \le 0 \\ 1 & else \end{cases}$$

- Since this function is non-linear and non-convex, therefore we choose to use an approximate given by

$$\Phi(\zeta) = \sum_{i=1}^{N} \zeta_i$$

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

Soft Margin SVM as an Optimization Problem

- Writing the constraints as losses, we have

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

$$L_i = \max_{\alpha_i} \alpha_i\left\{1 - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\} = \begin{cases} 0 & 1-\zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \le 0 \\ \infty & else \end{cases}, \qquad \alpha_i \ge 0$$

$$Z_i = \max_{\beta_i} \beta_i\{-\zeta_i\} = \begin{cases} 0 & -\zeta_i \le 0 \\ \infty & else \end{cases}, \qquad \beta_i \ge 0$$

Thus

$$\min_{\mathbf{w},b,\zeta}\max_{\alpha_i,\beta_i} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i + \sum_{i=1}^{N}\alpha_i\left\{1-\zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\} - \sum_{i=1}^{N}\beta_i\zeta_i$$

$$s.t \quad \alpha_i \ge 0, \beta_i \ge 0$$

- This is the primal form of the optimization problem

Soft Margin SVM as an Optimization Problem

- The Dual Form is

$$\max_{\alpha_i,\beta_i} \min_{\mathbf{w},b,\zeta} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i + \sum_{i=1}^{N}\alpha_i\left\{1-\zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right\} - \sum_{i=1}^{N}\beta_i\zeta_i$$

$$s.t \qquad \alpha_i \geq 0, \beta_i \geq 0$$

- Since the optimization problem is convex, therefore the optimal values of the dual and primal are equal
- Therefore the optimization problem can be written solved in its dual form

Soft Margin SVM as an Optimization Problem

$$\max_{\alpha_i,\beta_i} \min_{\mathbf{w},b,\zeta} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i + \sum_{i=1}^{N}\alpha_i\left\{1-\zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)}+b\right)\right\} - \sum_{i=1}^{N}\beta_i\zeta_i$$

$$= \max_{\alpha_i,\beta_i} \theta_D(\boldsymbol{\alpha},\boldsymbol{\beta})$$

$$\theta_D(\boldsymbol{\alpha},\boldsymbol{\beta}) = \min_{\mathbf{w},b,\zeta} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i + \sum_{i=1}^{N}\alpha_i\left\{1-\zeta_i - y_i\left(\mathbf{w}^T\mathbf{x}^{(i)}+b\right)\right\} - \sum_{i=1}^{N}\beta_i\zeta_i$$

Solving

$$\frac{\partial\theta_D}{\partial\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}^{(i)} = 0 \qquad \Rightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}^{(i)}$$

$$\frac{\partial\theta_D}{\partial b} = \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$\frac{\partial\theta_D}{\partial\zeta_i} = C - \alpha_i - \beta_i = 0 \qquad \Rightarrow \alpha_i + \beta_i = C$$

Soft Margin SVM as an Optimization Problem

$$\max_{\alpha_i, \beta_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} \right\}$$

$$s.t \qquad \alpha_i \geq 0, \beta_i \geq 0$$

$$\alpha_i + \beta_i = C, \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

Simplifying further

$$\max_{\alpha_i, \beta_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} \right\}$$

$$s.t \qquad 0 \leq \alpha_i \leq C, \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

- This problem can be solved using standard optimization packages

Soft Margin SVM as an Optimization Problem

- Some Observations
  - α_i will be non-zero (positive) only for the points that are support vectors
  - $\beta_i = C - \alpha_i$ will be zero for the points that violate the margin condition
  - C is the upper bound on α_i
  - C is the weight of the penalty of the term representing margin violation
    - If C is small, then more margin violations will occur
    - If C is large, lesser margin violations will result
  - C can be found out through cross-validation

Example

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

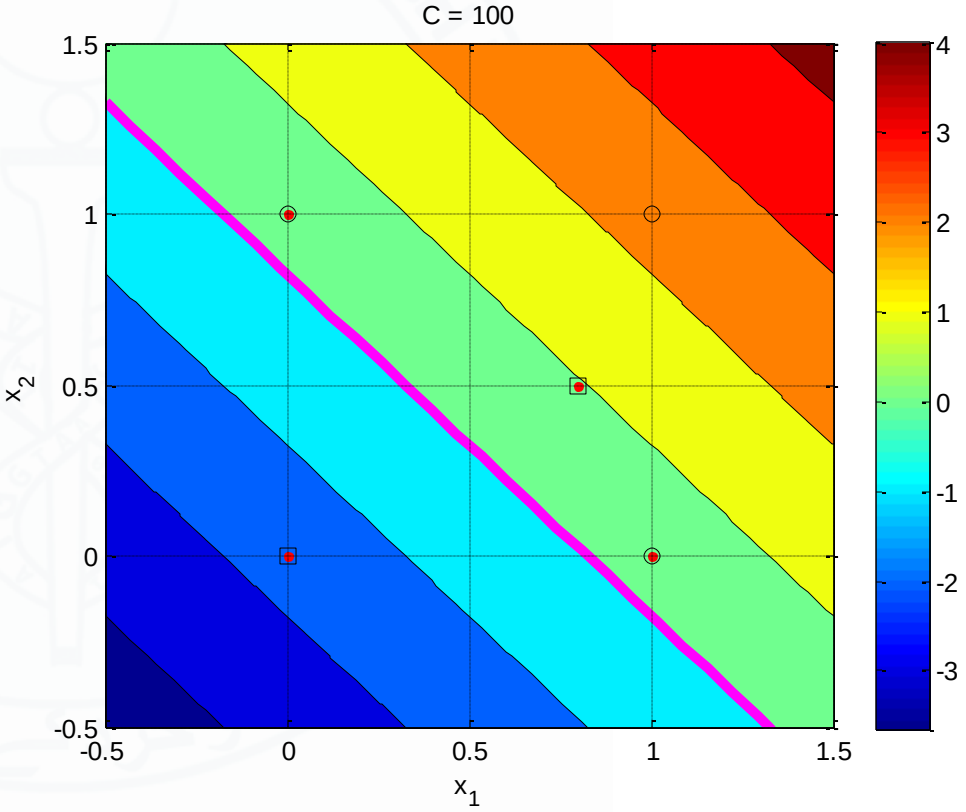
$$C = 100$$

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

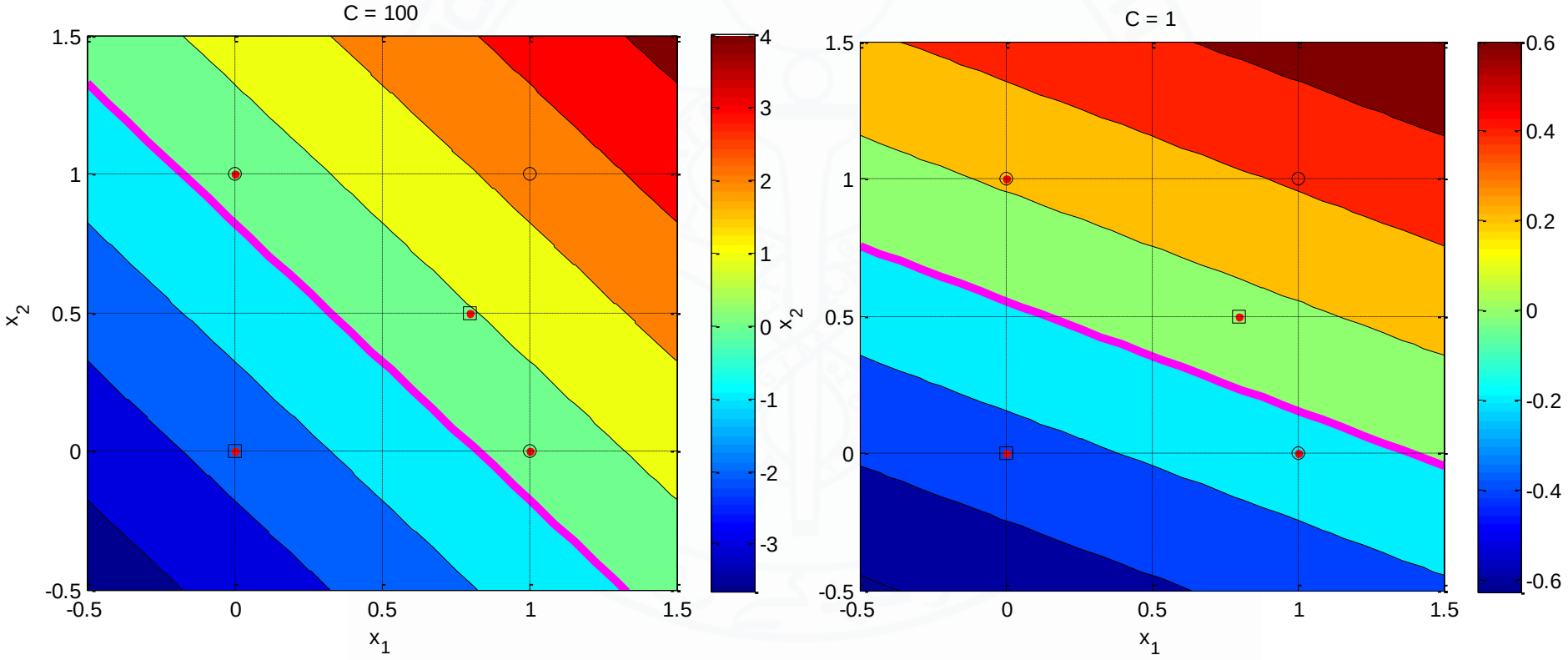
$$\alpha_i = \quad +34 \quad +52 \quad +82 \quad 0 \quad +100$$

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

$$b^* = -1.65$$



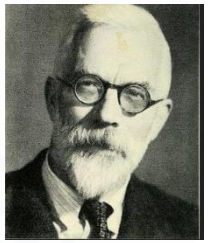
Example…



SVMs uptil now

- Vapnik and Chervonenkis:
  - Hard SVM (1962)
  - Theoretical foundations for SVMs
- Corinna Cortes
  - Soft SVM (1995)

- Enter: Bernard Scholkopf (1997)
  - 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 -

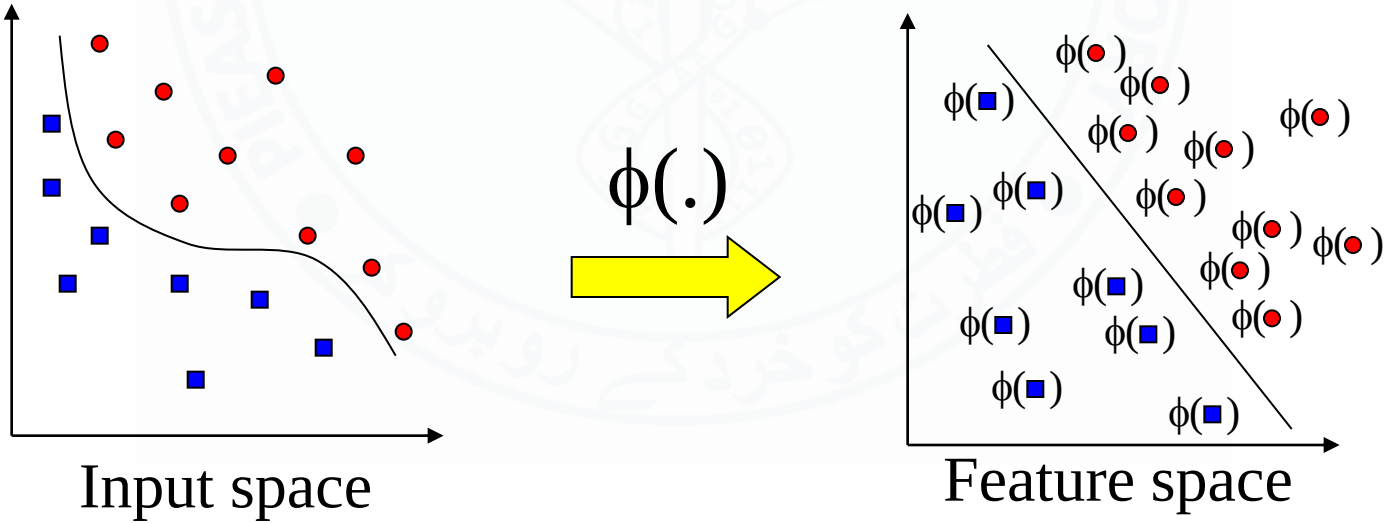
http://www.svms.org/history.html

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

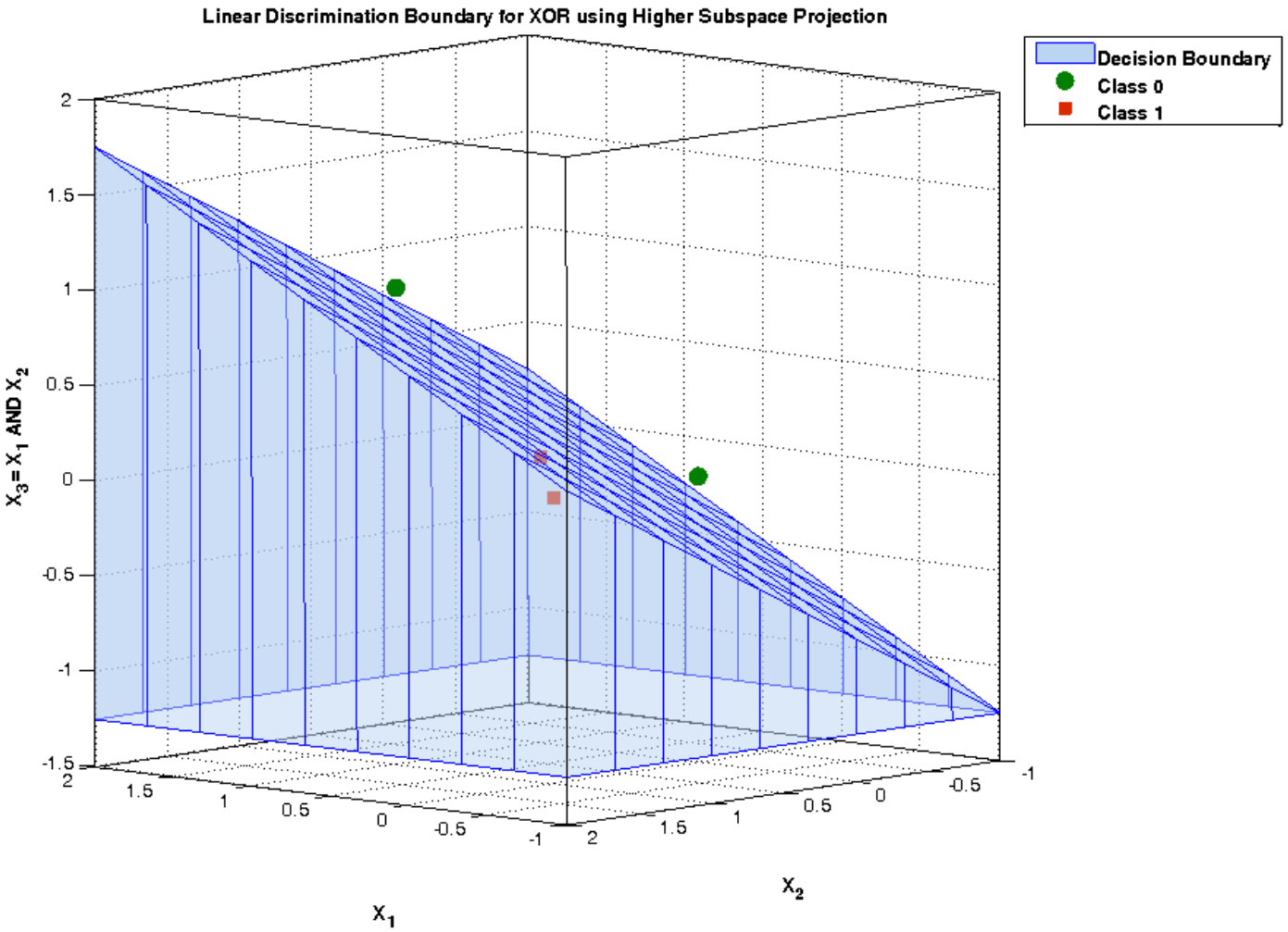
- Given a classification problem with a nonlinear boundary, we can, at times, find a mapping or transformation of the feature space which makes the classification problem linear separable in the transformed space



Examples: Transformation

- Let's define the mapping
  - $\boldsymbol{\phi}\left(\begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_1 x_2} \end{bmatrix}$
- With a mathematical proof show that the above mapping makes the XOR classification problem linearly separable.

XOR: Linear Separability

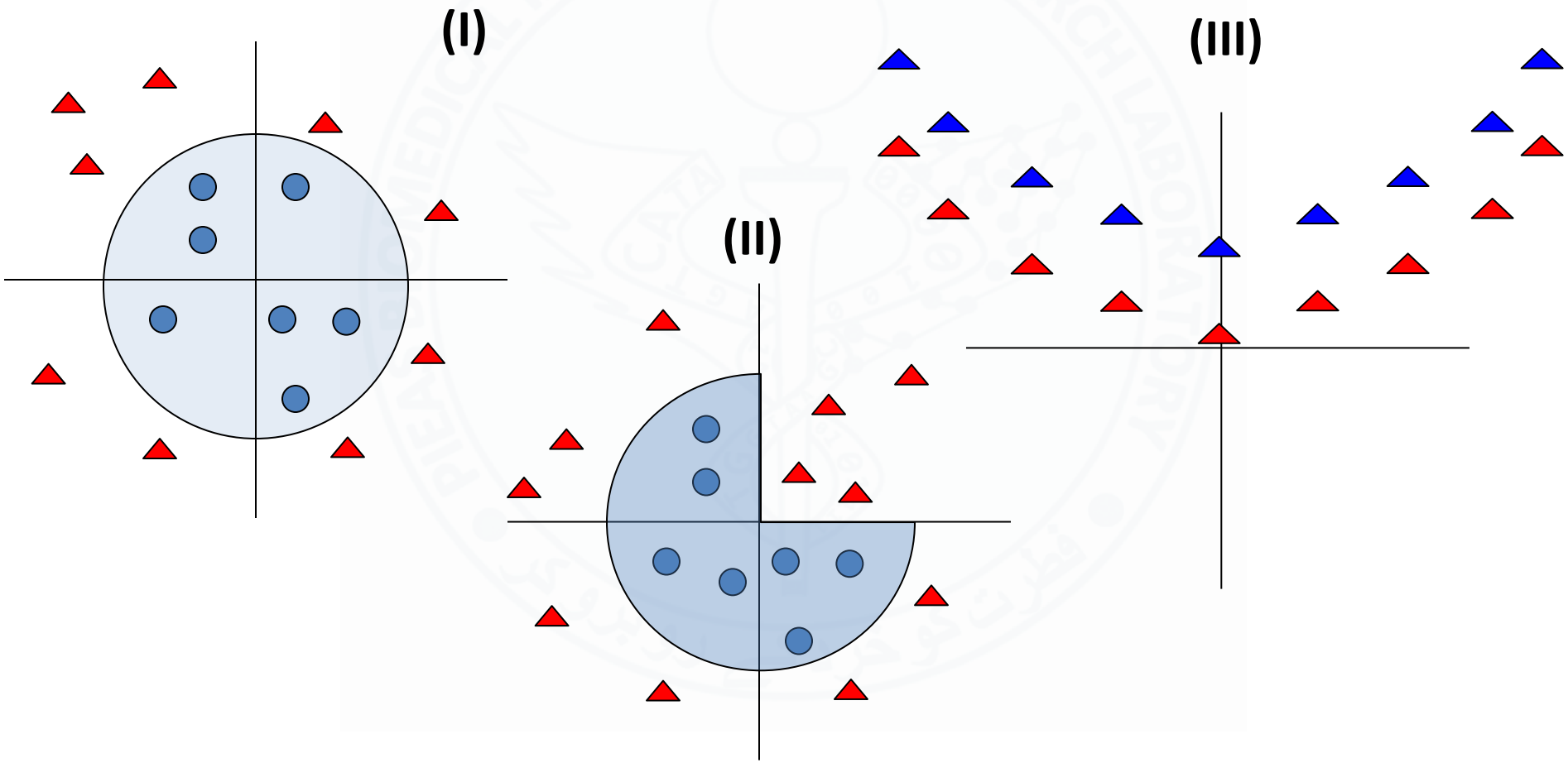


Examples: Transformation

- 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?



Transformations

- Transformations can be used to make the data linearly separable
- But it may not always be possible to find a transformation
  - Use a soft-SVM

Another look at the SVM

$$\max_{\alpha_i,\beta_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} \right\}$$

$$s.t \qquad 0 \leq \alpha_i \leq C, \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

- We can replace the dot product $\boldsymbol{x^{(i)^T} x^{(j)}}$ with:
  - a generalized dot product (inner product)
    - $\langle \boldsymbol{x^{(i)}, x^{(j)}} \rangle = \boldsymbol{\phi}(\boldsymbol{x^{(i)}})^{\boldsymbol{T}} \boldsymbol{\phi}(\boldsymbol{x^{(j)}})$
    - Advantage: can implement feature transformations
  - or a function (called the kernel function)
    - $\boldsymbol{K_{ij} = K(i, j)}$
    - Advantage: No need for explicit feature representation

Replacement with a feature transformation

- For the XOR problem we defined the transformation
  - $\boldsymbol{\phi}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$
- We can thus define an inner product
  - $\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle = \boldsymbol{\phi}(\boldsymbol{x}^{(i)})^{\boldsymbol{T}} \boldsymbol{\phi}(\boldsymbol{x}^{(j)}) = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_2^{(i)} x_2^{(i)} \end{bmatrix} \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \\ x_2^{(i)} x_2^{(j)} \end{bmatrix}$

$$= x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(j)} + x_1^{(i)} x_2^{(i)} x_1^{(j)} x_2^{(j)}$$

- This inner product implements the transformation and can potentially lead to a non-linear boundary

Kernel Functions ↔ Feature Transformation

- Since $\langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle$ is always a scalar, we can actually use a function (called a kernel function $k(i,j)$) to map the two examples to a scalar value
  - For the previous transformation
    - $k(i,j) = x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(j)} + x_1^{(i)} x_2^{(i)} x_1^{(j)} x_2^{(j)}$
  - Thus, the inner product from a feature transformation can be written as a kernel and a valid kernel function can thus be considered as an inner product in some feature space (proven by Moore-Aronszajn Theorem)
    - We'll talk what makes kernel valid later
  - A kernel is thus a generalized dot product
    - A measure of how similar the two examples are
      - not of whether they belong to the same class

Some kernel functions

- We can use arbitrary kernels, for example …
  - The dot-product kernel
    - $\boldsymbol{K(a,b) = a^T b}$
      - Feature transform: $\boldsymbol{\phi(a) = a}$
  - The homogenous polynomial kernels
    - $\boldsymbol{K(a,b) = \left(a^T b\right)^p}$, $\boldsymbol{p}$ is called degree
      - For $\boldsymbol{p = 2}$ and 2D data, $\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$: $\boldsymbol{\phi(a)} = \begin{bmatrix} a_1^2 \\ a_2^2 \\ \sqrt{2}a_1 a_2 \end{bmatrix}$
  - The Radial Basis Function (RBF) Kernel
    - $\boldsymbol{K(a,b) = e^{-\frac{\|a-b\|^2}{2\sigma^2}}}$, $\boldsymbol{\sigma}$ controls the spread of the Gaussian
      - Feature transform?
- The RBF and Homogenous Polynomial kernels implement non-linear boundaries

Kernels as Similarity Functions

- We have been saying that kernels are similarity functions – here is how
  - Example
    - **For $\boldsymbol{K(x, y) = \langle x, y \rangle = x^T y}$**
    - For the dot product, the associated distance measure is
      - $\boldsymbol{d(x, y)^2 = \|x - y\|^2 = (x - y)^T (x - y) = x^T x - 2x^T y + y^T y = \|x\|^2 + \|y\|^2 - 2x^T y}$
      - This implies: $\boldsymbol{K(x, y) = \frac{1}{2}\left(\|x\|^2 + \|y\|^2 - d(x, y)\right)}$
      - Thus, the linear kernel measures the similarity between two points as the inverse of the square of the Euclidean distance between them (up to $\boldsymbol{\|x\|^2 + \|y\|^2}$)
        - » Thus the SVM with a linear kernel is no different, in its distance measurements, from a nearest neighbor classifier!!

Another advantage

$$\max_{\alpha_i,\beta_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(i,j) \right\}$$

$$s.t \qquad 0 \leq \alpha_i \leq C, \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

- Once we replace the dot product with a kernel function (i.e., perform the kernel trick or 'kernelize' the formulation), the above 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

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
  - 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

Kernel Trick

- You can develop a 'kernelized' version of the soft-SVM as well
- Advantage
  - Removes the explicit representation of data
  - Allows non-linear boundaries

- For understanding a 'valid' kernel, we need to introduce the concept of a kernel matrix

Kernel Matrix

- For $\boldsymbol{x}^{(1)} = \begin{bmatrix}0\\0\end{bmatrix}, \boldsymbol{x}^{(2)} = \begin{bmatrix}0\\1\end{bmatrix}, \boldsymbol{x}^{(3)} = \begin{bmatrix}1\\0\end{bmatrix}, \boldsymbol{x}^{(4)} = \begin{bmatrix}1\\1\end{bmatrix}$

  With the kernel: $k(i,j) = x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)} + x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)}$

  - $k(1,1) = 0$
  - $k(1,2) = 0$
  - …
- We get this matrix
  - $K_{ij} = k(i,j)$

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	3

- If you know the kernel matrix you don't need to know the original features anymore

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data z is classified as class 1 if $f > 0$, and as class 2 if $f < 0$

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

The discriminant function

- For a test object z, the discriminant function essentially is a weighted sum of the similarity between z and a pre-selected set of objects (the support vectors)

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

$\mathcal{S}$: the set of support vectors

Vectorization

- If we define the following:
  - $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_N]^{\boldsymbol{T}}$
  - $\boldsymbol{y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^{\boldsymbol{T}}$
  - $\boldsymbol{\alpha} \circ \boldsymbol{y} = [\alpha_1 y_1 \quad \alpha_2 y_2 \quad \cdots \quad \alpha_N y_N]^{\boldsymbol{T}}$
    - $\circ$ to mean element wise product
      - Hadamard product
  - $\mathbf{1}_N = [1 \quad 1 \quad \ldots \quad 1]^{\boldsymbol{T}}$
  - $\boldsymbol{X}_{(d \times N)} = [\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_N}]$
    - **This implies:** $\boldsymbol{K} = \boldsymbol{X}^T \boldsymbol{X}$

$$max_{\boldsymbol{\alpha}} \ \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} \circ \boldsymbol{y})^T \boldsymbol{X}^T \boldsymbol{X} (\boldsymbol{\alpha} \circ \boldsymbol{y})$$

Subject to:

$$\boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$

$$\boldsymbol{y}^T \boldsymbol{\alpha} = 0$$

$$max_{\boldsymbol{\alpha}} \ \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} \circ \boldsymbol{y})^T \boldsymbol{K} (\boldsymbol{\alpha} \circ \boldsymbol{y})$$

Subject to:

$$C \succcurlyeq \boldsymbol{\alpha} \succcurlyeq \mathbf{0}$$

$$\boldsymbol{y}^T \boldsymbol{\alpha} = 0$$

Notice that there isn't any data related term here

This replacement of the dot product with the kernel is called the **kernel trick**

Assignment 2B

- Implement a soft kernelized SVM

What is the behavior of an inner product?

- 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
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq \boldsymbol{0}$
    - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \boldsymbol{0}$ iff $\boldsymbol{x} = \boldsymbol{0}$
- These conditions need to be satisfied by the kernel function too

Kernel Matrix

- What's the kernel matrix for this problem with the linear kernel?

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	2

- With the polynomial kernel (p=2)?

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	4

Conditions for a function to be a kernel

- A kernel function must satisfy Mercer's condition
  - The kernel matrix must be symmetric positive semi-definite
  - What does that mean?
    - $K(x^{(1)}, x^{(2)}) = K(x^{(2)}, x^{(1)})$
    - $\gamma^T K \gamma \geq 0$ for any γ
      - The Eigen-values of K must be non-negative

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	2

Dimensions: 2
Since we have 4 points, there exist labelings for which the classification problem is not linearly separable

Eigen Values
0,0,1,3

0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	3

Dimensions: 3
Since we have 4 points, these points will always be separable in the corresponding feature space no matter how you label them or where you put them!

Eigen Values
0, 0.26, 1, 3.73

Example: Kernel

- $\boldsymbol{K(u,v) = e^{-\frac{\|u-v\|^2}{2\sigma^2}}}$
- The kernel matrix is (for $\boldsymbol{\sigma^2 = 0.5}$):

1	e^{-1}	e^{-1}	e^{-2}
e^{-1}	1	e^{-2}	e^{-1}
e^{-1}	e^{-2}	1	e^{-1}
e^{-2}	e^{-1}	e^{-1}	1

- Eigenvalues are 1.93, 0.73, 0.47, 0.86

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) $\kappa(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/(2\sigma^2))$.

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
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

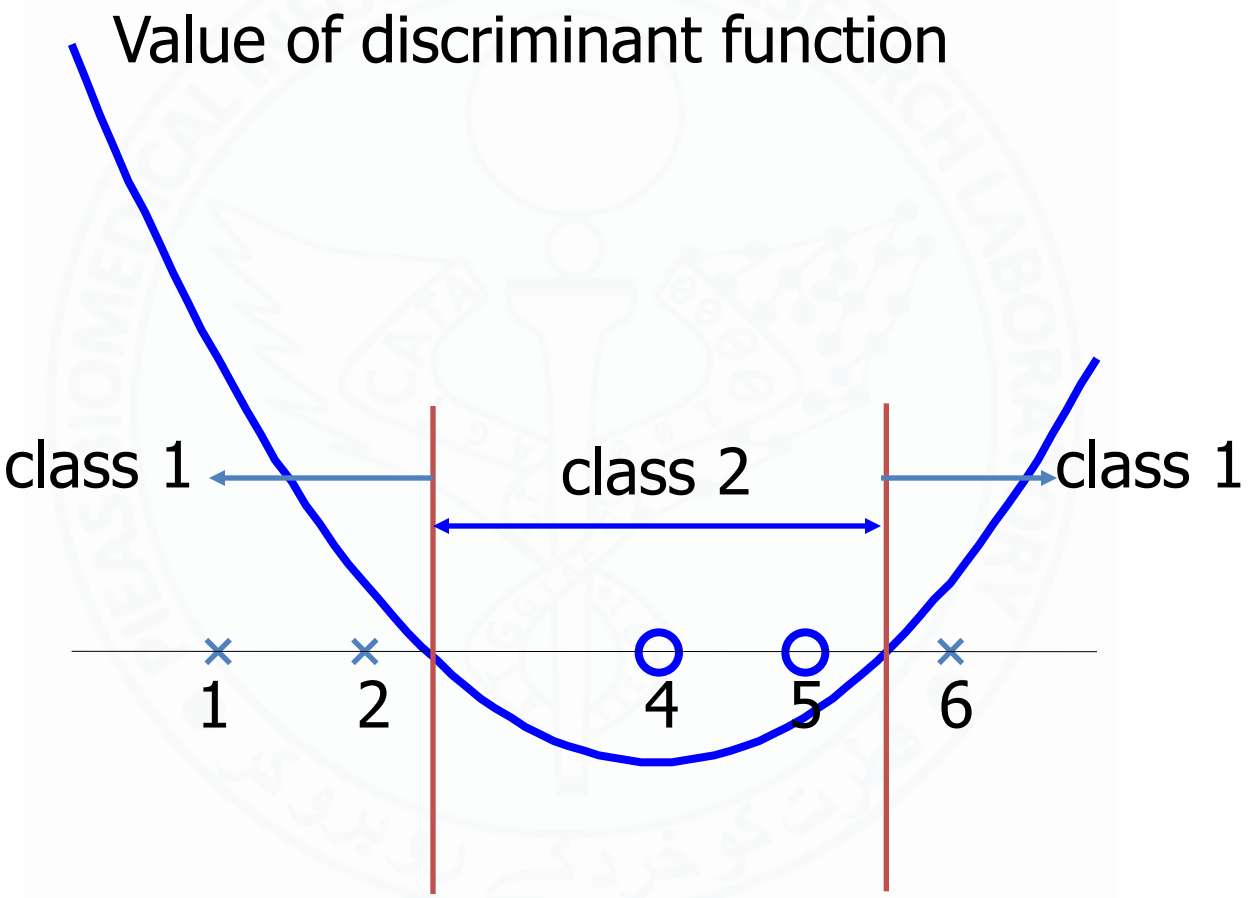
$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

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

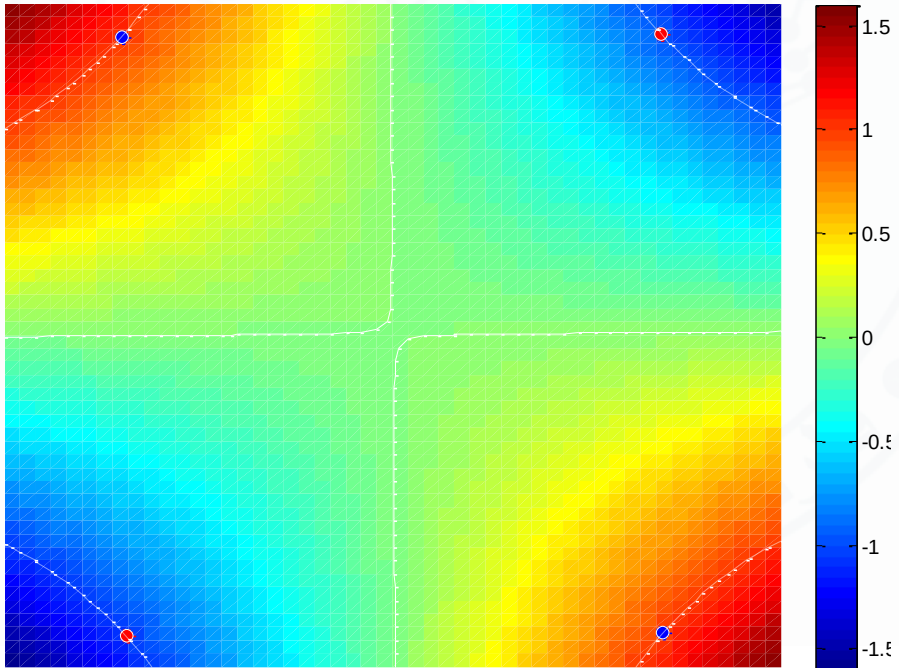
$$f(z) = 0.6667z^2 - 5.333z + 9$$

Example


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

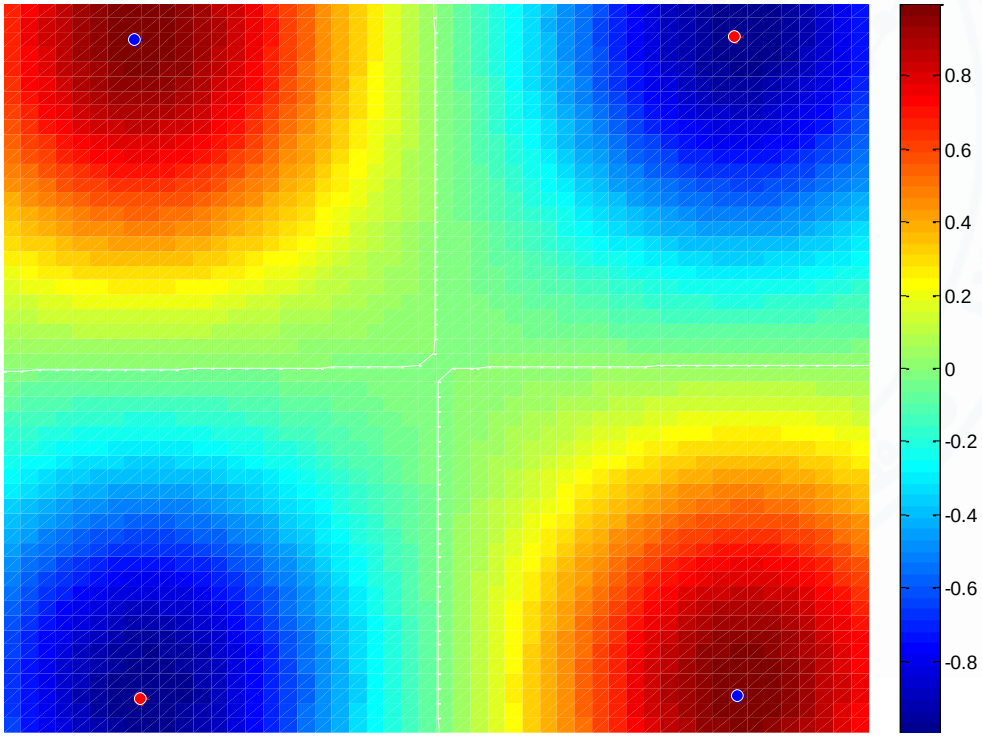
Solving the XOR

- C=1
- With polynomial kernel of degree 2



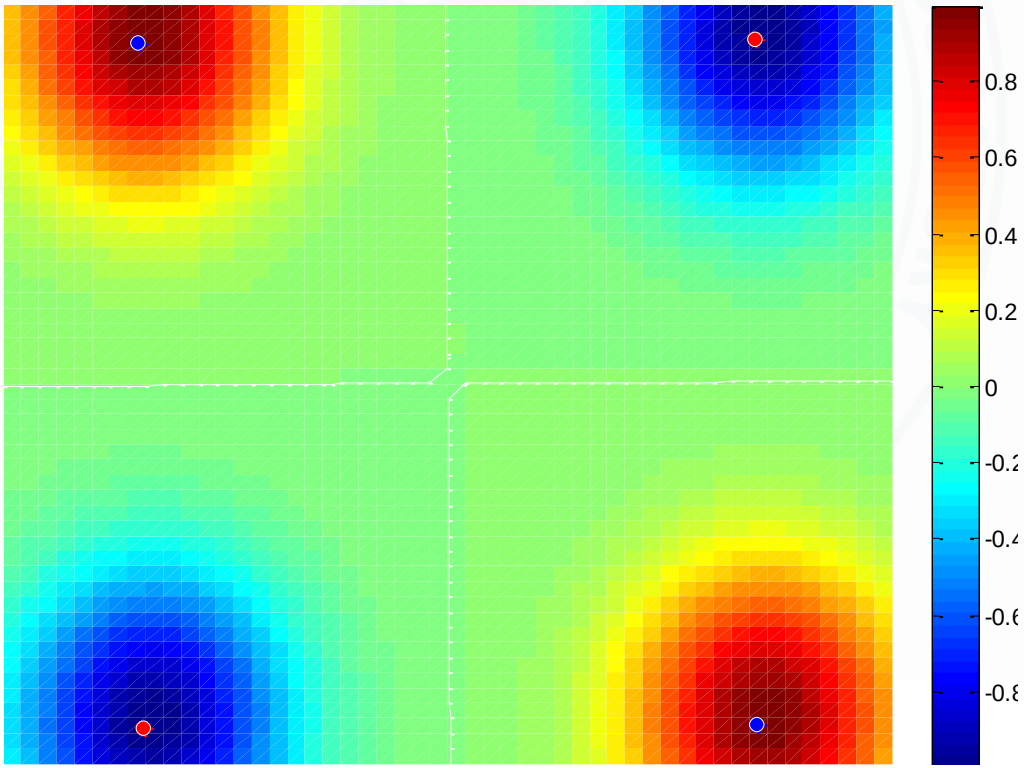
Solving the XOR

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

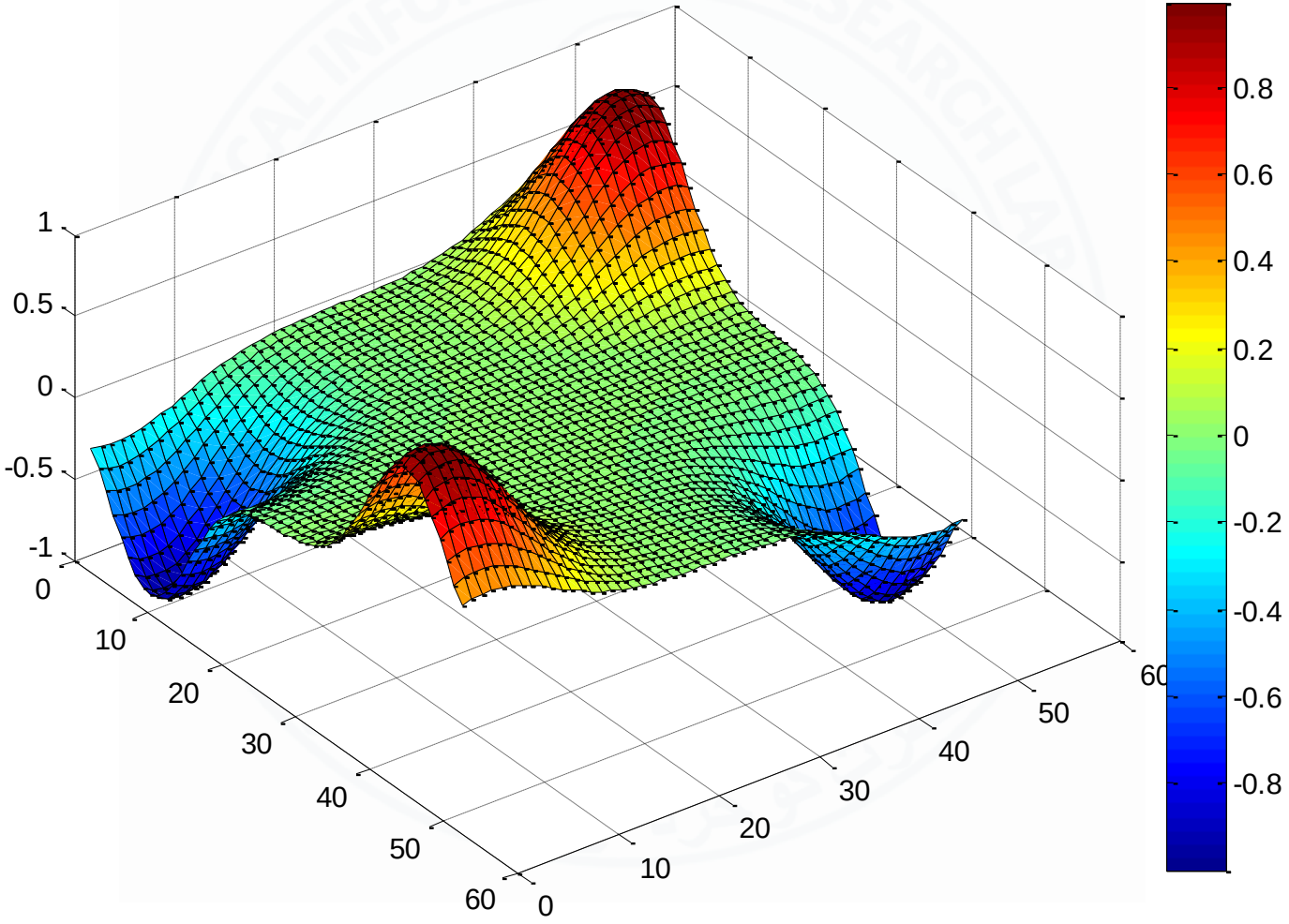


Solving the XOR

- 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
  - 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.

Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification and such SVMs do exist
- But you can also try to do multi-class classification

End of Lecture

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

- Danny Hillis