Institut Informatika & Bisnis DARMAJAYA
Yayasan Alfian Husin

SAINS DATA
Institut Informatika & Bisnis Darmajaya

Kampus Merdeka INDONESIA JAYA

MERDEKA BELAJAR

DATA SCIENCE DARMAJAYA
"YOUR BEST FUTURE IN DATA"

MEETING: [10]

INTRODUCTION TO SUPPORT VECTOR MACHINE

BY: HENDRA KURNIAWAN

[12-09-2023] | [KODE MK: BDG23407 | MATAKULIAH: DATA SCIENCE AND BUSINESS ANALYTICS | SKS: 2/2 | VERSI: 01

Introduction to Support Vector Machine

1. Classification via Mathematical Functions
2. Regression via Mathematical Functions
3. Class Probability Estimation and Logistic “Regression”
4. Prerequisites for DS (To Become Data Scientist one Should Know the Various Technique)

Introduction

- SVMs provide a learning technique for
  - Pattern Recognition
  - Regression Estimation
- Solution provided SVM is
  - Theoretically elegant
  - Computationally Efficient
  - Very effective in many Large practical problems
- It has a simple geometrical interpretation in a high-dimensional feature space that is nonlinearly related to input space
- By using kernels all computations keep simple.
- It contains ANN, RBF and Polynomial classifiers as special cases.

Learning Machine

- A bound on the Generalization Performance of Learning Machine
  - Expected Risk:
  - Empirical Risk:

$$R(\alpha) = \int \frac{1}{2} \left| y - f(\vec{x}, \alpha) \right| dP(\vec{x}, y)$$

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} \left| y_i - f(\vec{x}_i, \alpha) \right|$$

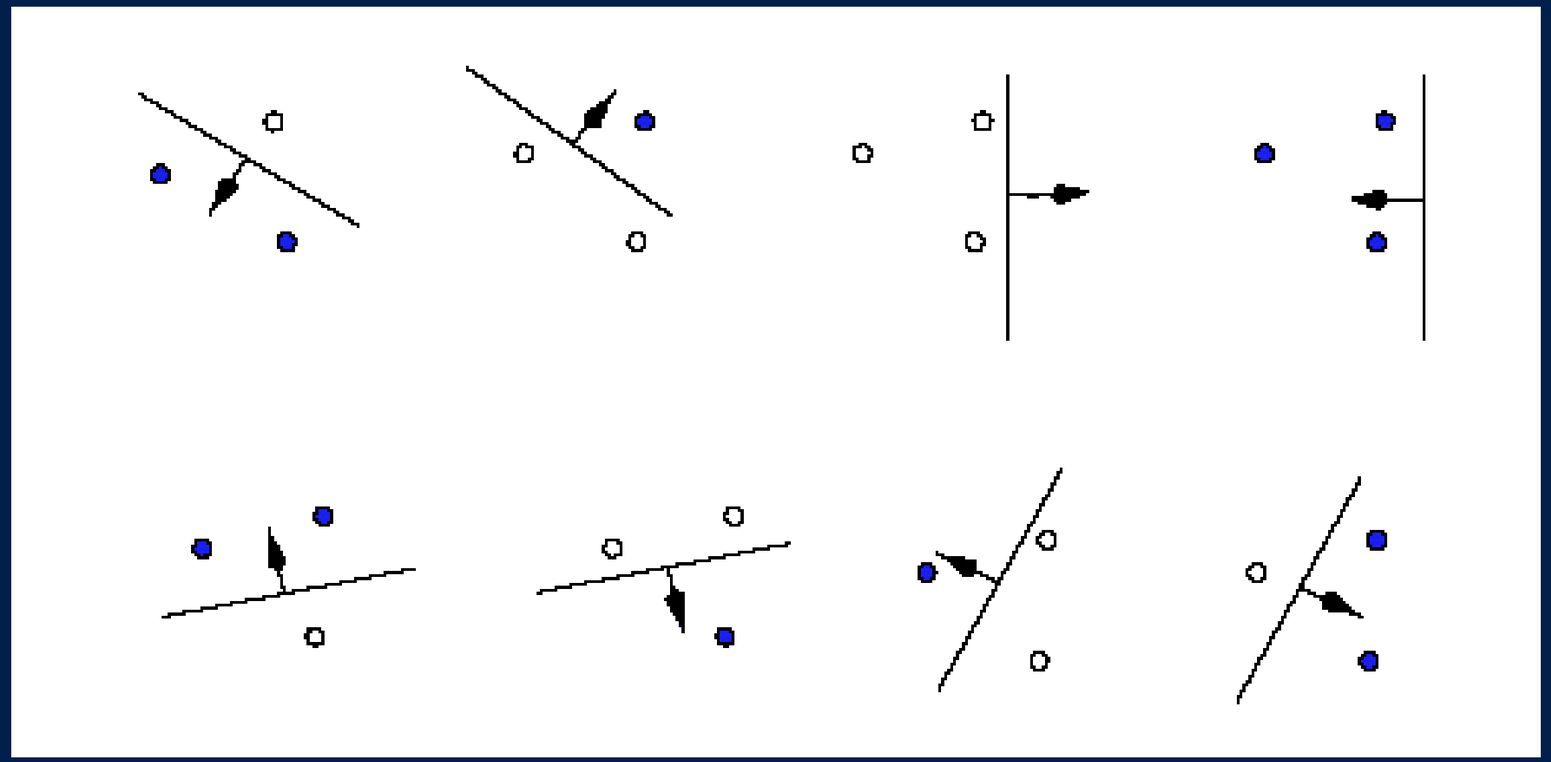
$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l} \right)}$$

- h is the VC dimension, a measure of the notion of capacity of a classifier.

VC Dimension

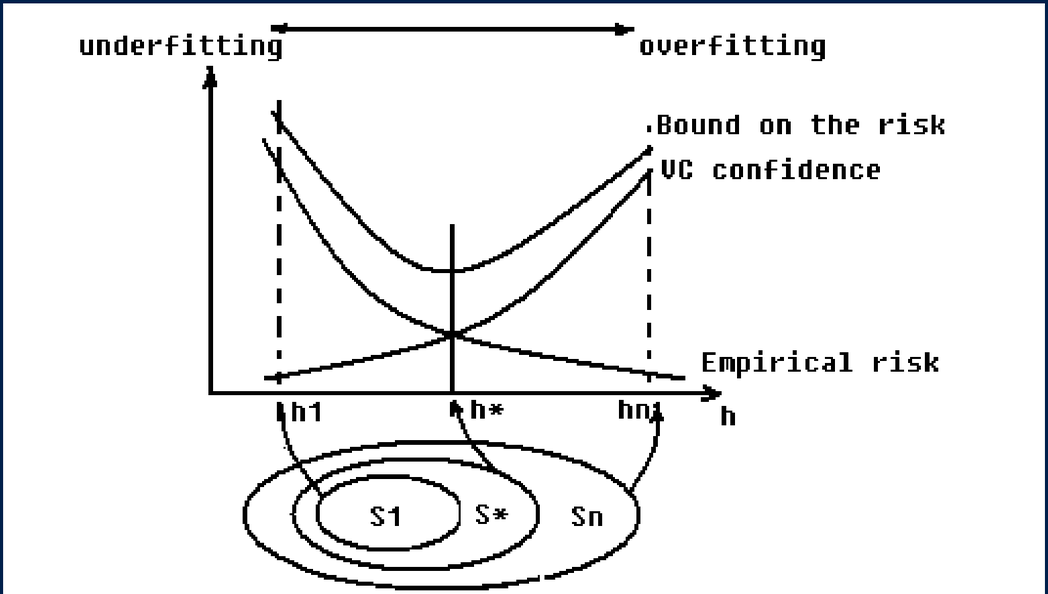
- The VC dimension is a property of a set of functions $\{f(\alpha)\}$, and can be defined for various classes of function f .
- The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum number of training points that can be shattered by $\{f(\alpha)\}$.
- VC dimension gives concreteness to the notion of the capacity of a given set of functions.
- The number of parameters of Learning Machines is not proportional to the VC dimension.

VC Dimension (Vapnik-Chervonenkis) – An example



The VC-Dimension of the set of oriented hyperplanes in $\mathbf{R}^n$ is (n+1)

Structural Risk Minimization



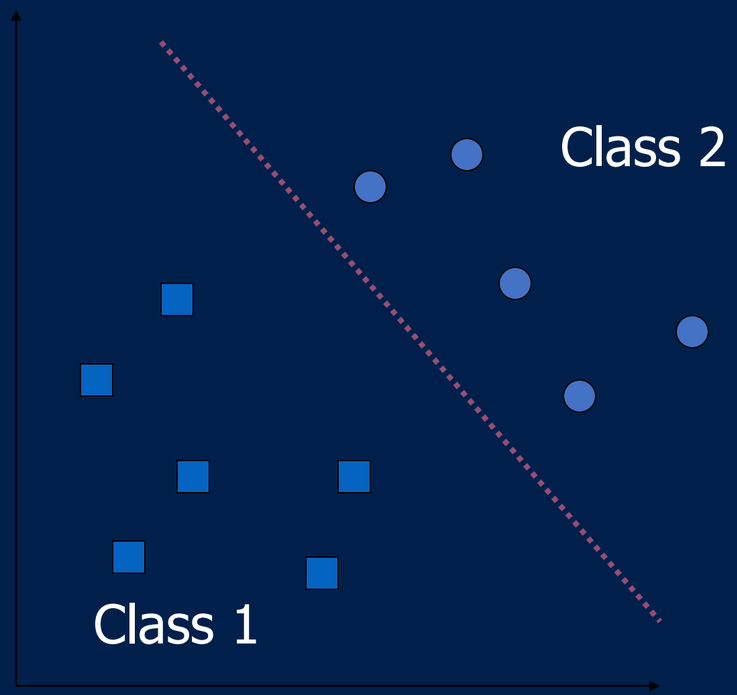
$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\left(\frac{h\left(\log\ (2l/h)+1\right) - \log\ (\eta/4)}{l}\right)}$$

Two Approaches

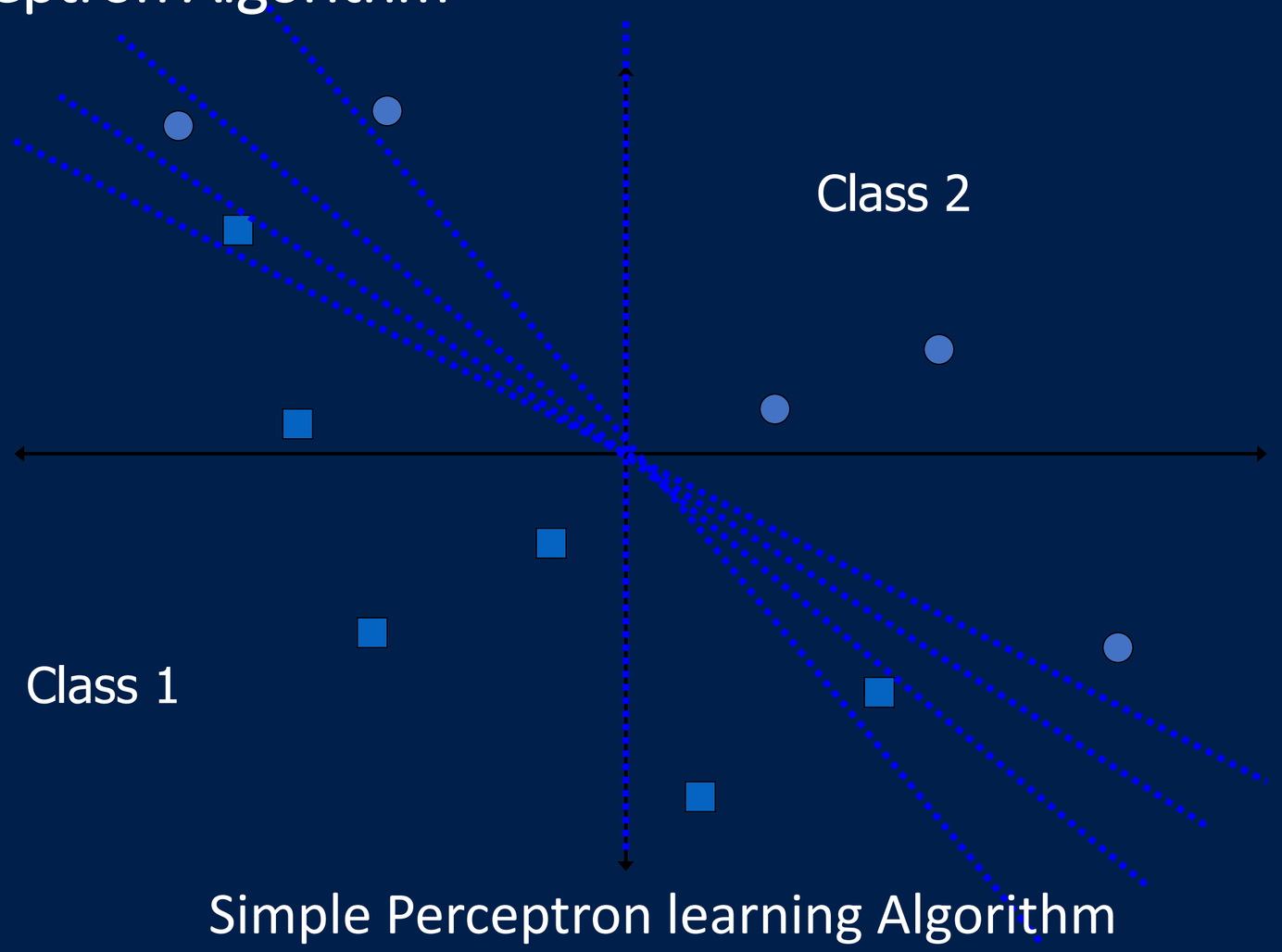
- **Goal:** To find a trained machine in the series whose sum of empirical risk and VC confidence is minimal.
- **Neural Network**
  - Fix the VC confidence and minimize the empirical risk
- **Support Vector Machine**
  - Fix the empirical risk and minimize the VC confidence

The Two Class Problem

- Several decision boundaries can separate these two classes.
- Perceptron Algorithm learns any separating hyperplane.
- SVM learns the best separating hyperplane.

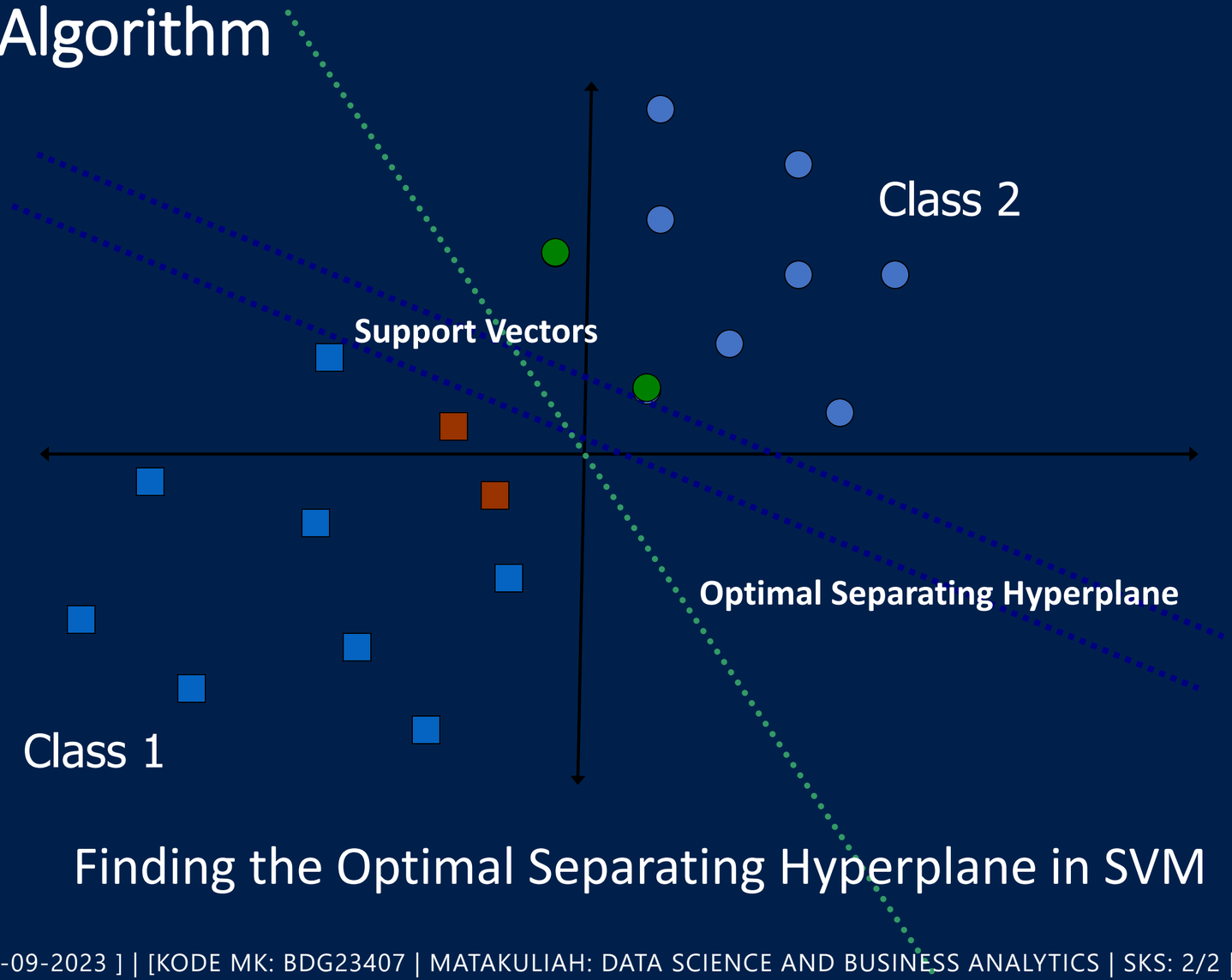


Perceptron Algorithm



Simple Perceptron learning Algorithm

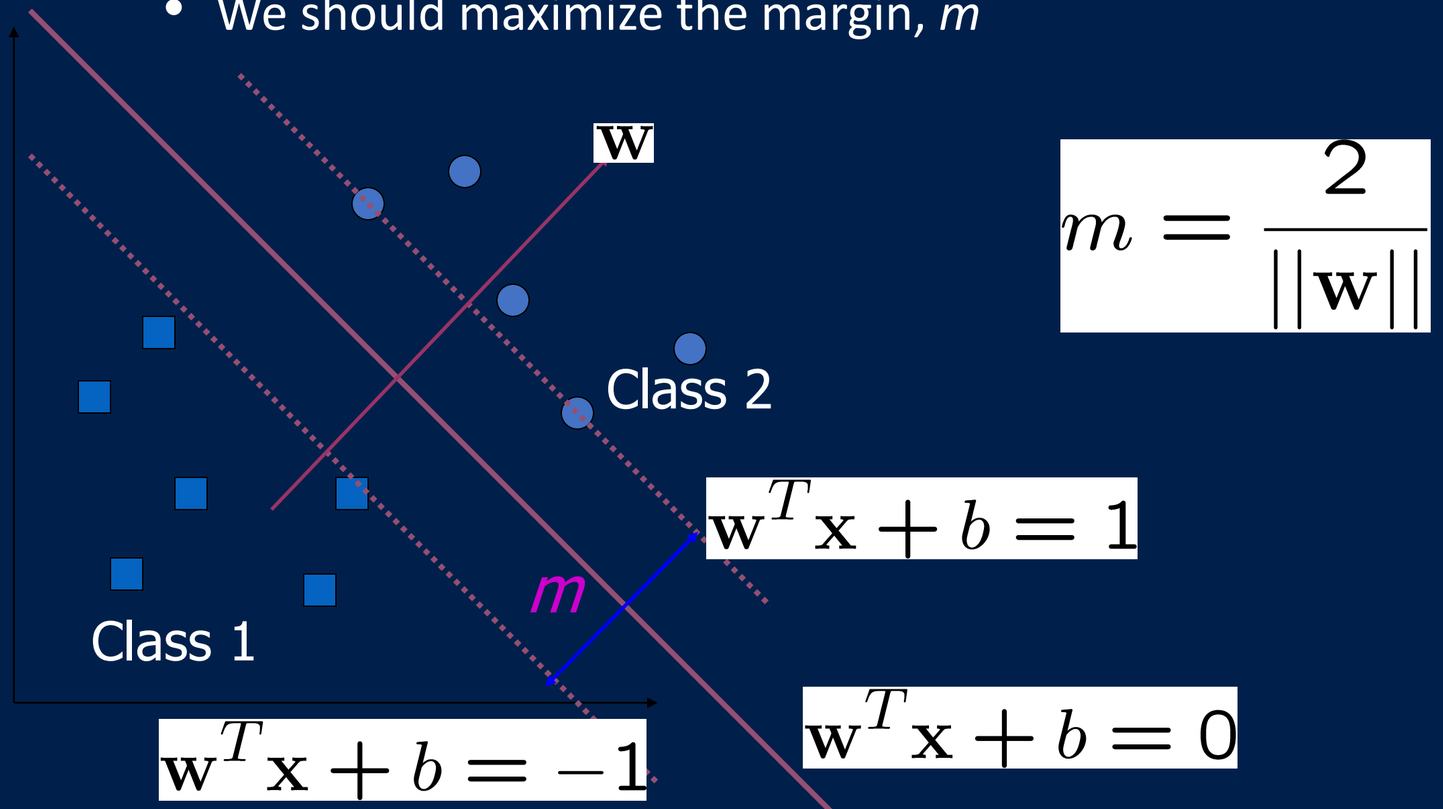
SVM Algorithm



Finding the Optimal Separating Hyperplane in SVM

Decision Boundary

- The decision boundary/hyperplane should be as far away from the data of both classes as possible.
  - We should maximize the margin, *m*



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

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set
- And let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$ $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

Dual Formulation

- The Lagrangian for this problem is

$$L(w,b,\mu)=\frac{1}{2}w^T w+\sum_{i=1}^{l}\mu_i[1-y_i(w^T x_i+b)]$$

where $\mu=[\mu_1..\mu_l]^T\in\Re^l$ are the Lagrange multipliers

- Quadratic cost Optimization and linear constraints.
- The Kuhn-Tucker Conditions for the problem

$$\nabla_w L = w^* - \sum_{i=1}^{l}\mu_i^* y_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l}\mu_i^* y_i = 0$$

$$1 - y_i(x_i^T w^* + b^*) \le 0, \forall i,$$

$$\mu_i^* \ge 0$$

$$\mu_i^*[1 - y_i(x_i^T w^* + b^*)] = 0, \forall i$$

where (w*,b*) is the global solution of L and μ* is the optimal Lagrange multiplier.

Support Vectors

- Complimentary Slackness condition

$$\mu_i^* [1 - y_i (x_i^T w^* + b^*)] = 0, \forall i$$

- We must have

$$y_i (x_i^T w^* + b^*) > 1 \Rightarrow \mu_i^* = 0$$

$$\mu_i^* > 0 \Rightarrow y_i (x_i^T w^* + b^*) = 1$$

- Support Vectors are the set of x_i's that have $\mu^*_i > 0$

The Dual Problem

- We can transform the problem to its dual

$$\max q(\mu) = \sum_{i=1}^{l} \mu_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \mu_i \mu_j y_i y_j (x_i^T x_j)$$

subject to

$$\mu_i \geq 0, \forall i$$

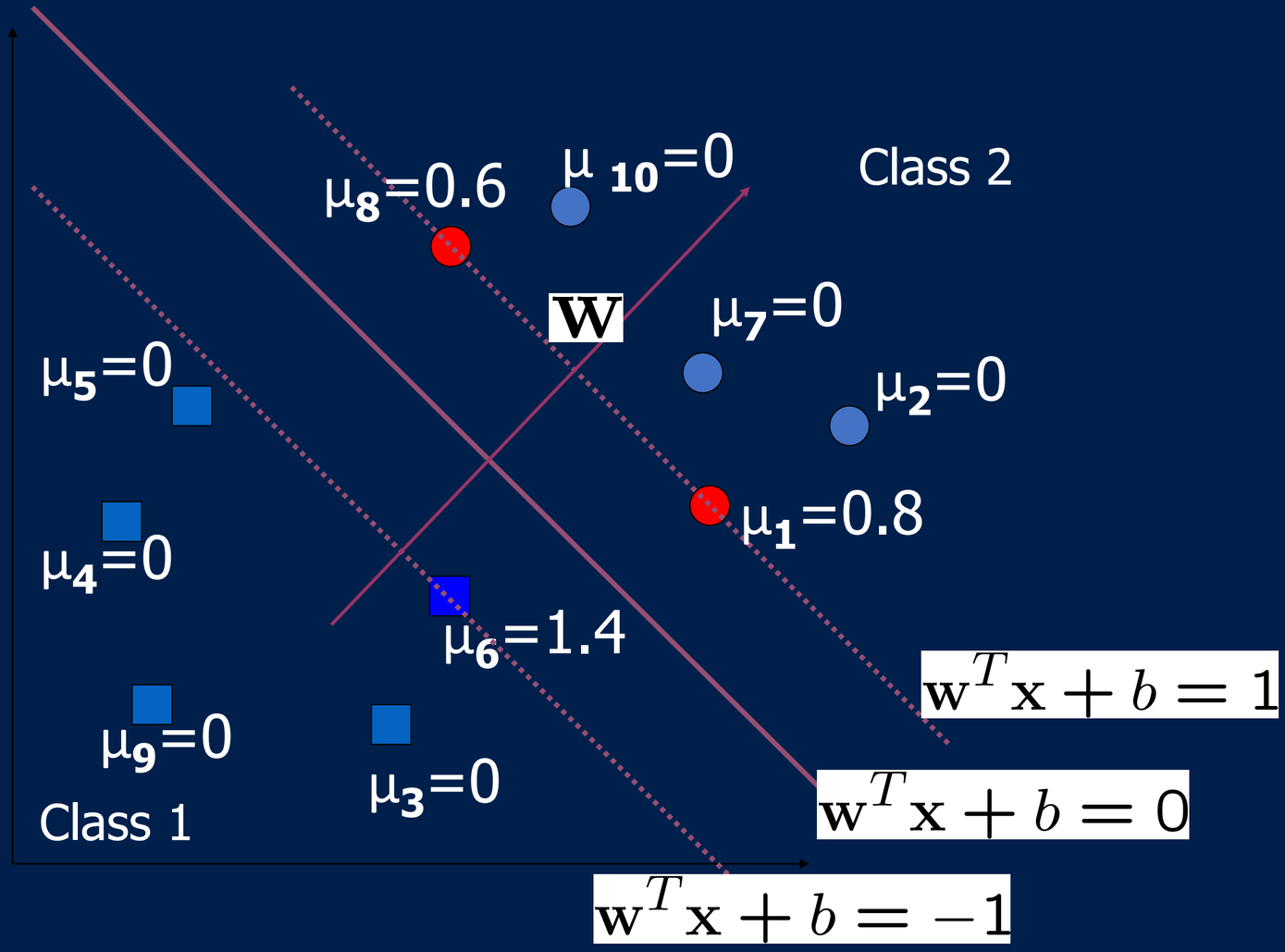
$$\sum_{i=1}^{l} \mu_i y_i = 0$$

- This is a quadratic programming (QP) problem
- **w*,b*** can be recovered by

$$w^* = \sum_{i \in S} \mu_i^* y_i x_i$$

$$b^* = y_i - x_i^T w^*, \mu_i > 0$$

A Geometrical Interpretation

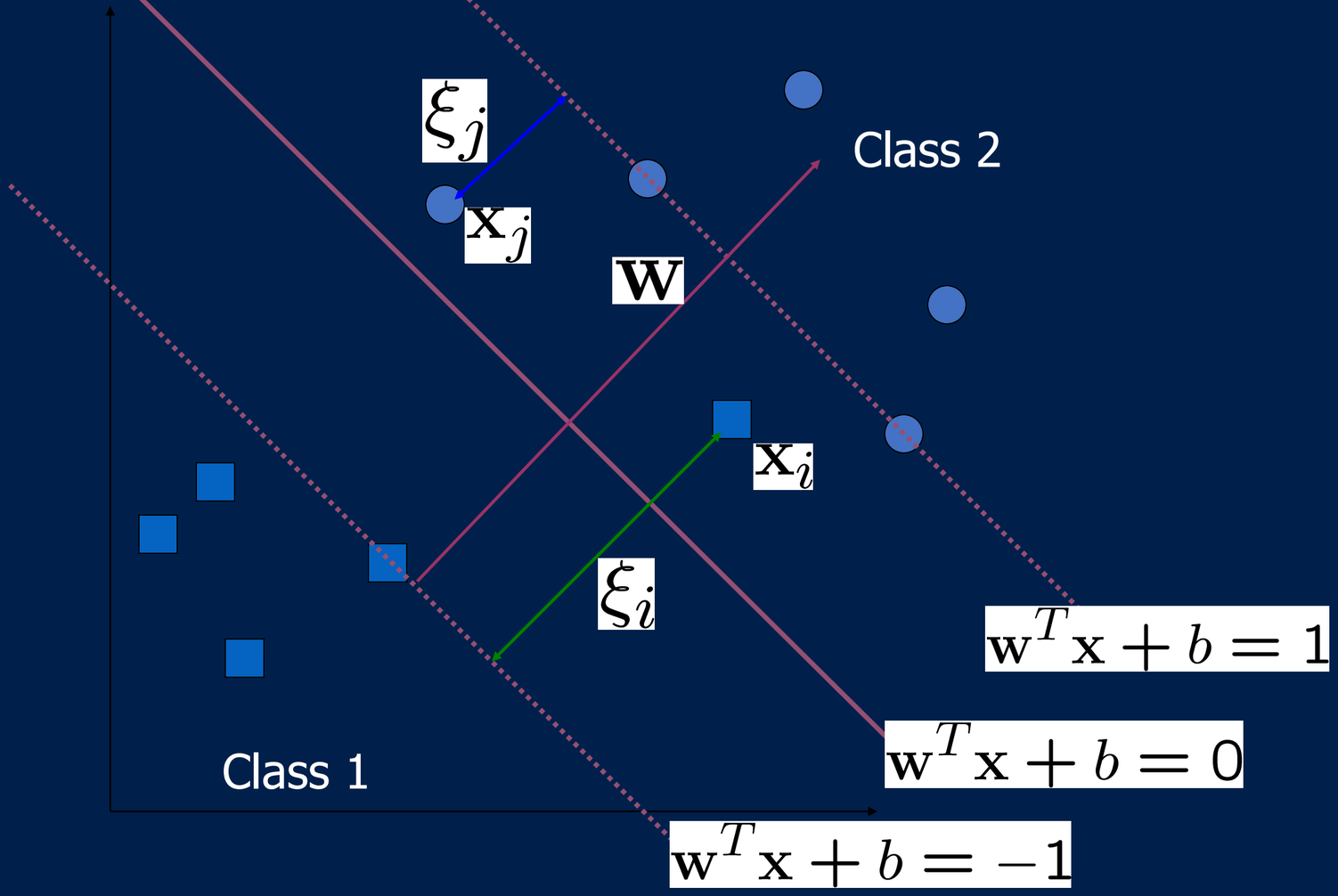


Some Notes

- There are theoretical upper bounds on the error on unseen data for SVM
  - The larger the margin, the smaller the bound
  - The smaller the number of SV, the smaller the bound
- Note that in both training and testing, the data are referenced only as inner product, $\mathbf{x}^T\mathbf{y}$
  - This is important for generalizing to the non-linear case

If Not Linearly Separable

- We allow "error" ξ_i in classification



Soft Margin Hyperplane

- Define ξ_i=0 if there is no error for x_i
  - ξ_i are just “slack variables” in optimization theory

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$
  - *C* : tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The New Optimization Problem

- The dual of the problem is

$$\text{max. } W(\mu) = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \mu_i \mu_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

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

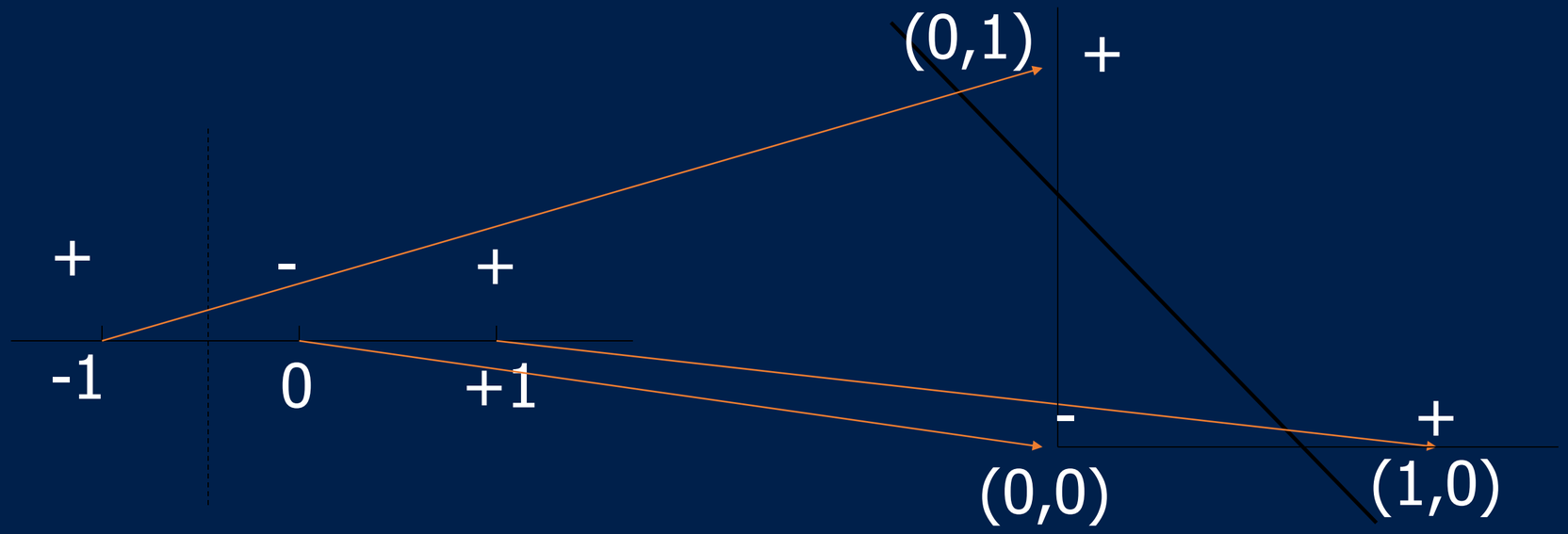
- **w** is also recovered as $\mathbf{w} = \sum_{j=1}^{s} \mu_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound *C* on α_i
- A QP solver can be used to find $\boldsymbol{\mu}_i$'s

Extension to Non-linear Decision Boundary

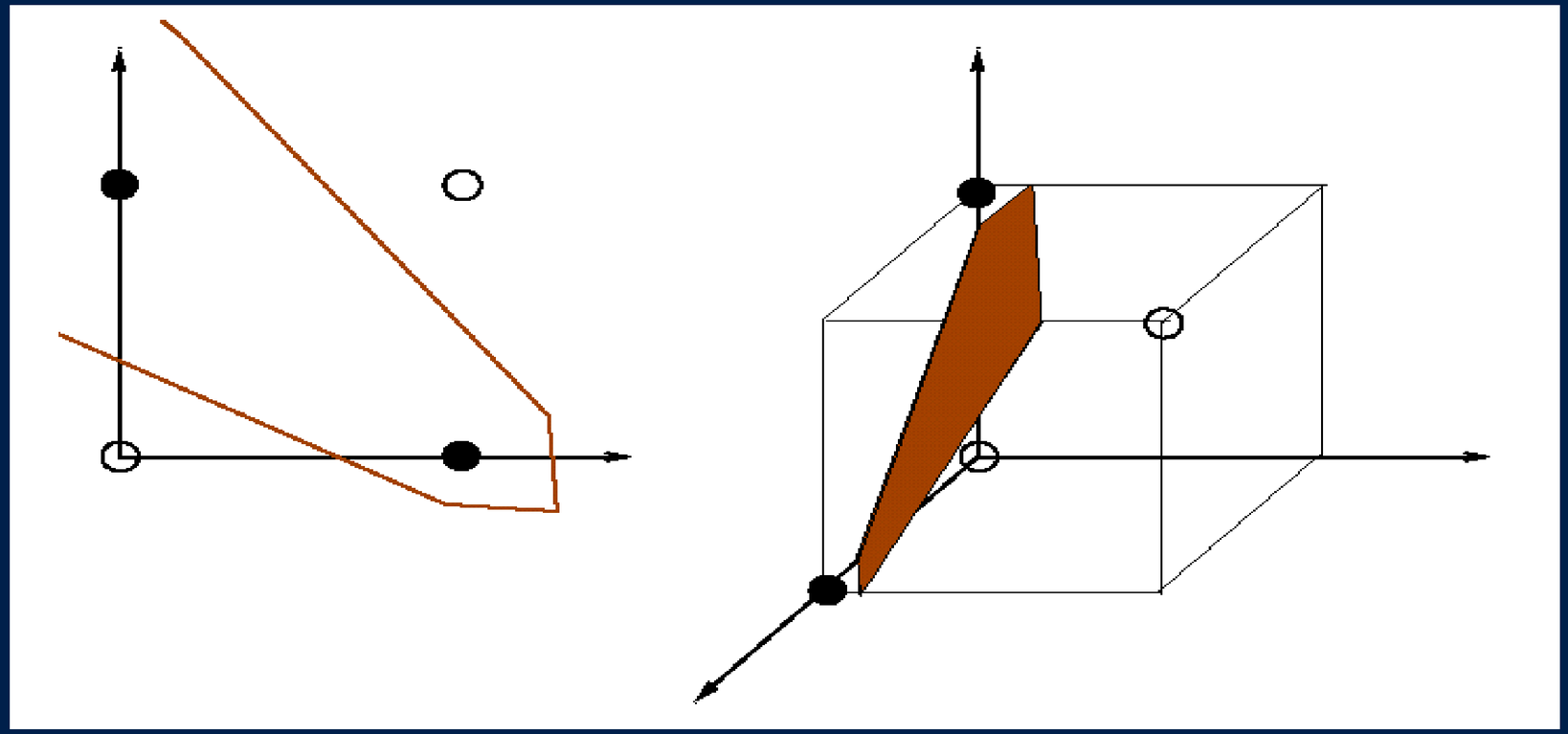
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to “make classes linearly separable”
  - Input space: the space $\mathbf{x}_i$ are in
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be “easier” with a proper transformation. Example: XOR

Higher Dimensions

- Project the data to high dimensional space where it is linearly separable and then we can use linear SVM – (Using Kernels)



The XOR problem

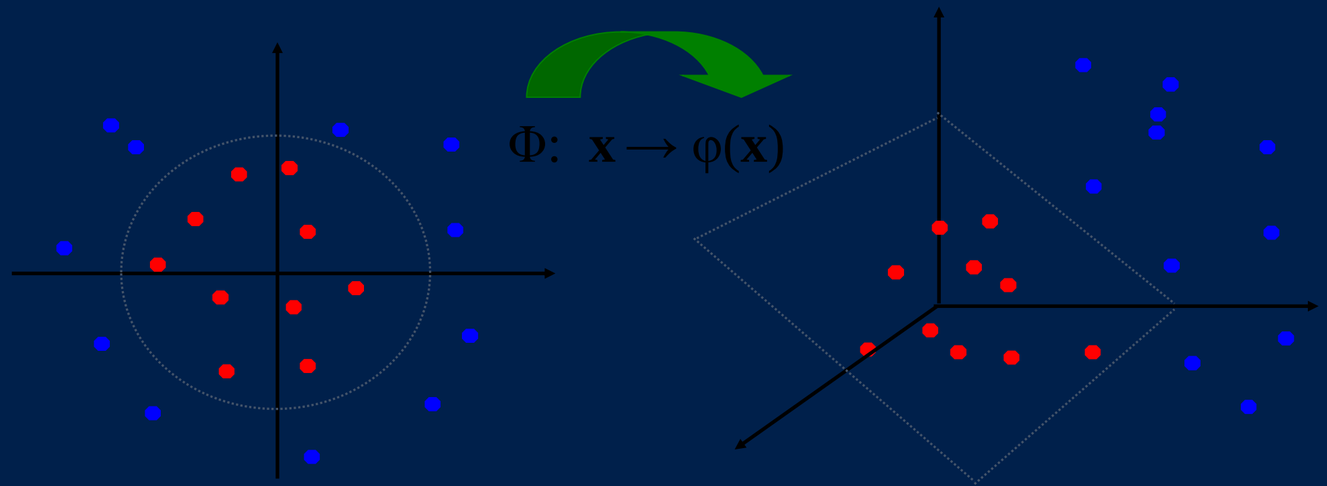


$X = (x_1, x_2)$

$Z = (x_1, x_2, x_1x_2)$

Extension to Non-linear Boundary

- Possible problem of the transformation
  - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
  - Kernel tricks for efficient computation
  - Minimize $||\mathbf{w}||^2$ can lead to a “good” classifier



What is Kernel?

$$(\mathbf{x} \cdot \mathbf{y})^2 = \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right)^2$$

$$= \left(\begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix} \cdot \begin{pmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{pmatrix} \right)$$

$$= (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})),$$

$$K(x, y) = (x \cdot y)^2$$

Example Transformation

- Define the kernel function $K(\mathbf{x},\mathbf{y})$ as

$$K(\mathbf{x},\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned} \langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x},\mathbf{y}) \end{aligned}$$

- The inner product can be computed by K without going through the map $\phi(.)$

Kernel Trick

- The relationship between the kernel function *K* and the mapping $\phi(.)$ is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

  - This is known as the kernel trick
- In practice, we specify *K*, thereby specifying $\phi(.)$ indirectly, instead of choosing $\phi(.)$
- *K* (**x**,**y**) needs to satisfy Mercer condition in order for $\phi(.)$ to exist

Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

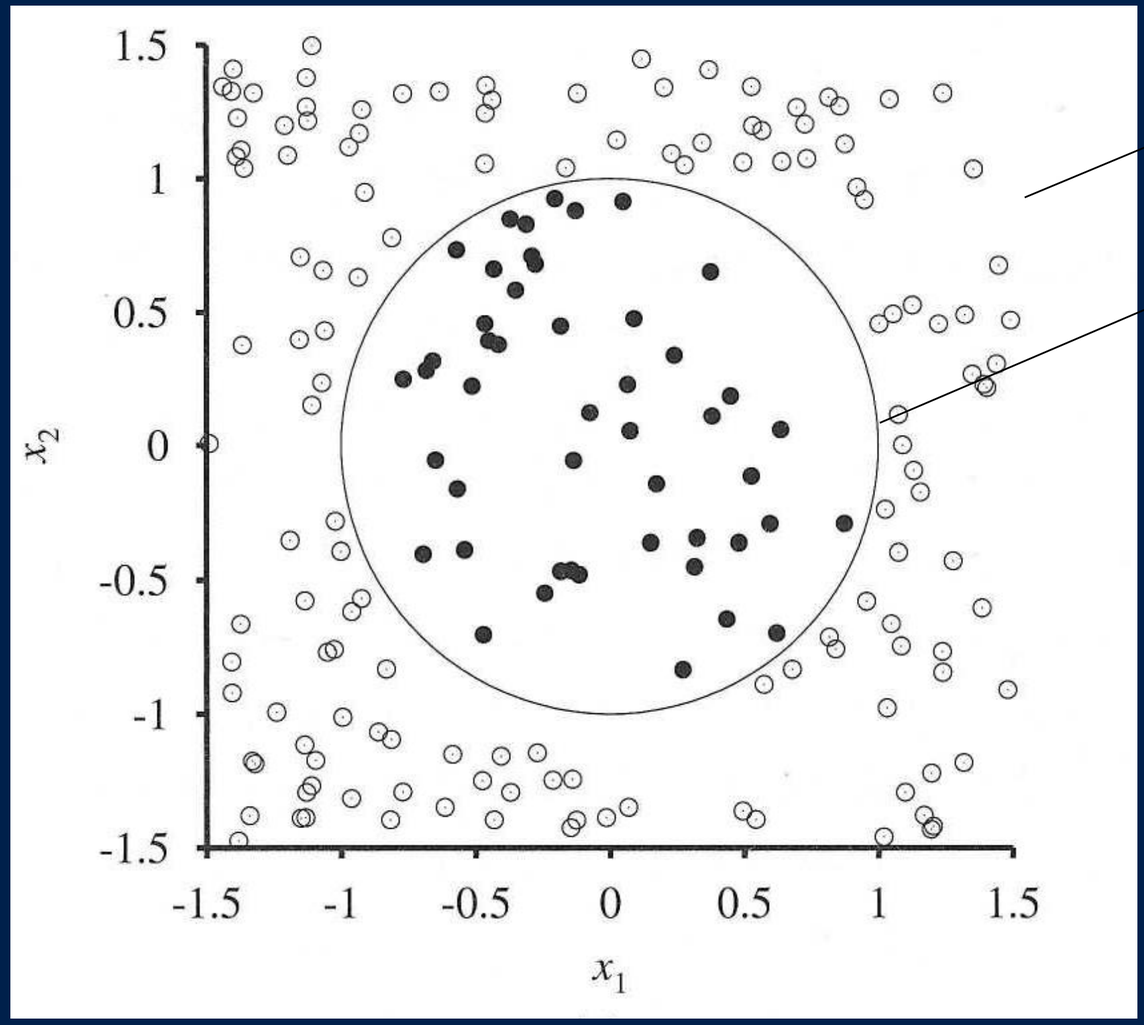
  - Closely related to radial basis function neural networks

- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - It does not satisfy the Mercer condition on all κ and θ

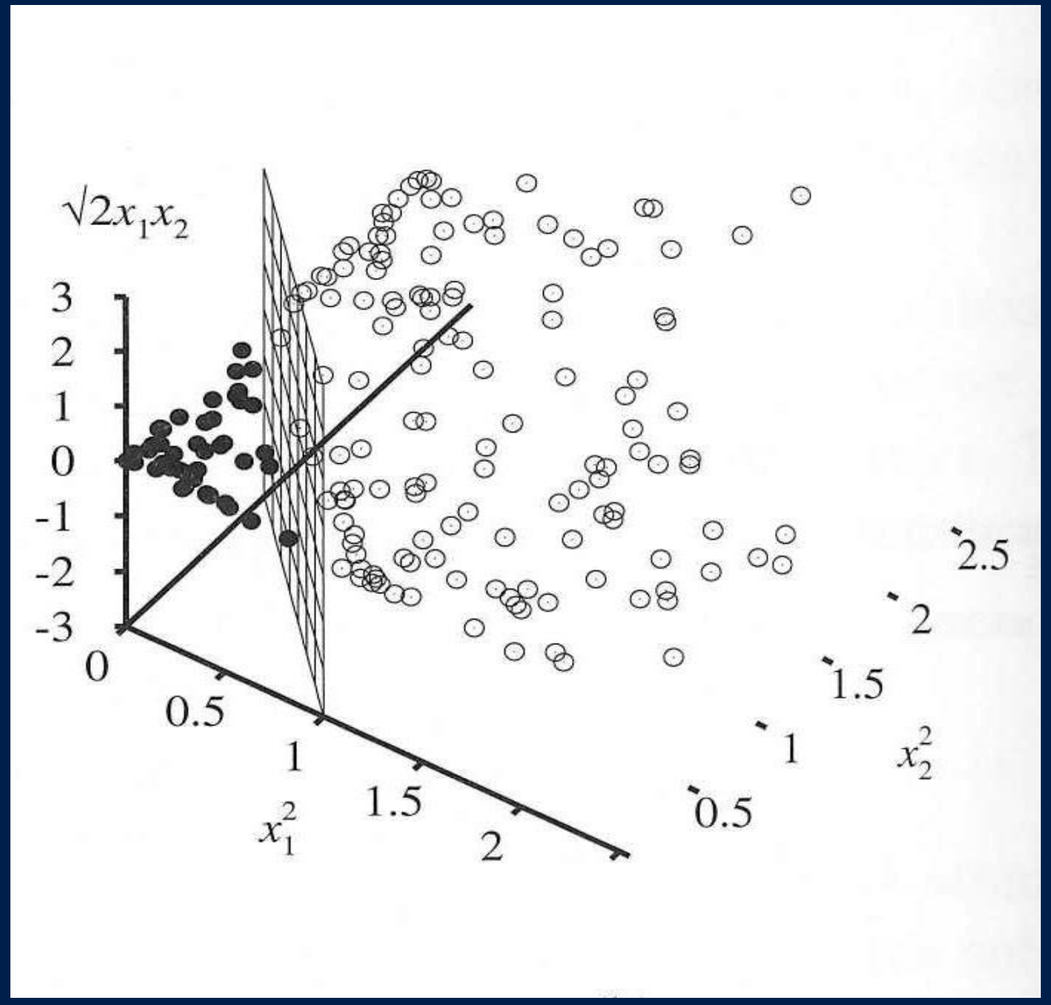
(x_1, x_2)



y = -1

y = +1

$$(x_1^2, x_2^2, \sqrt{2x_1x_2})$$



Optimization Algorithms

- Most popular optimization algorithms for SVMs are SMO [Platt '99] and SVM^{light} [Joachims' 99], both use *decomposition* to hill-climb over a subset of μ_i's at a time.
- Idea behind SMO
  - Adjusting only 2 μ_i's at each step

$$\text{Maximize } L(\mu_1, \mu_2) = \sum_{i=1}^{l} \mu_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \mu_i \mu_i y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } \mu_1 y_1 + \mu_2 y_2 = const \text{ and } 0 \le \mu_i \le C, \; i = 1, 2$$

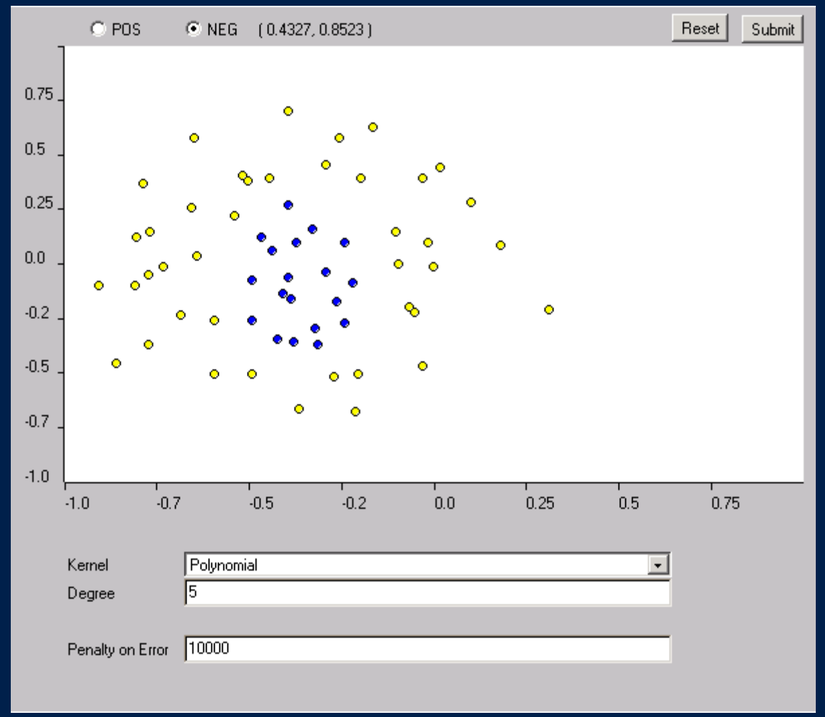
  - All μ_i's are initialized to be zero

SVM vs. Neural Networks

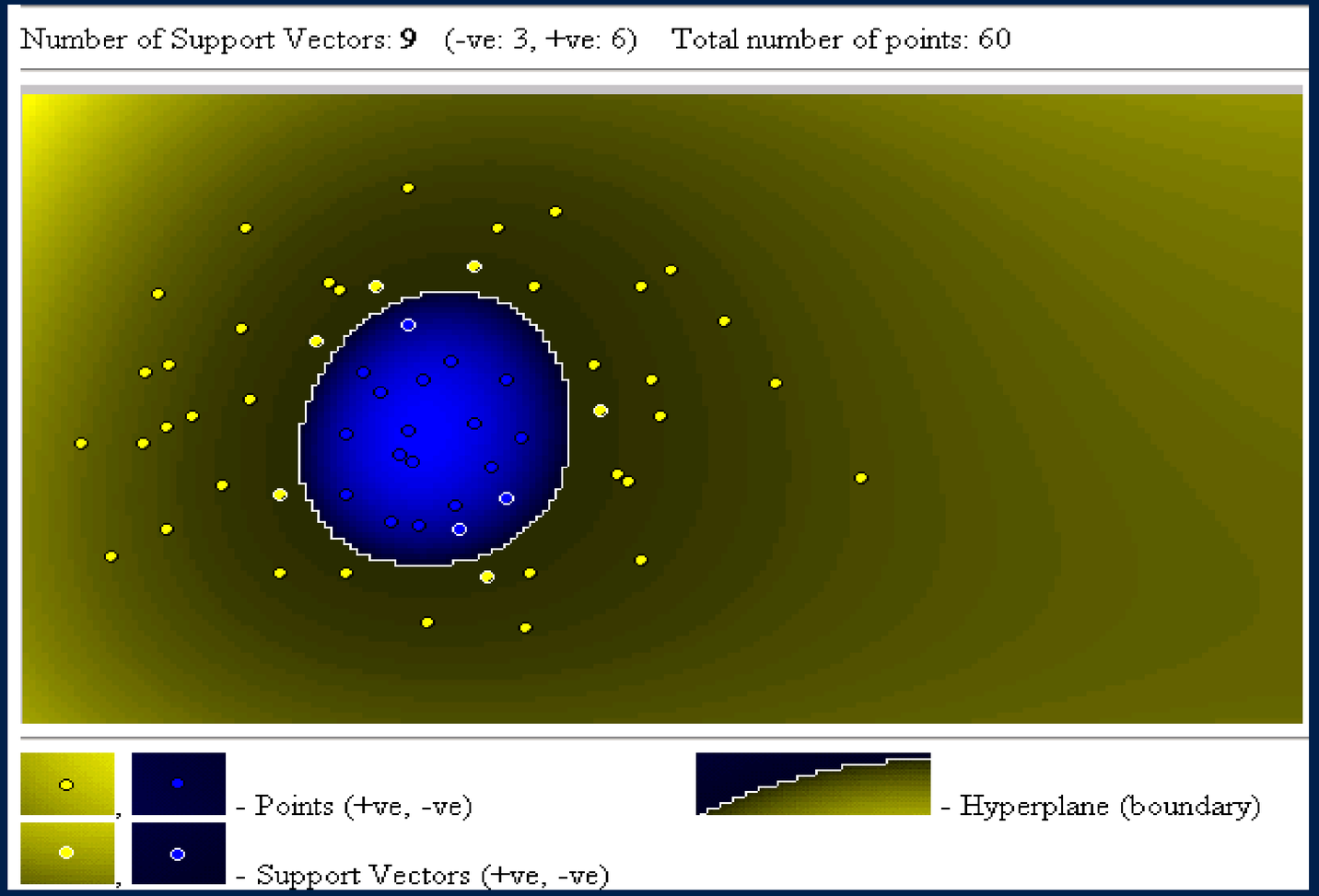
- SVM
  - Relatively new concept
  - Nice Generalization properties
  - Hard to learn – learned in batch modes using QP techniques
  - Using kernels can learn very complex functions

- Neural Networks
  - Generalizes well but doesn’t have mathematical foundation
  - Can easily be learnt in incremental fashion
  - To learn complex function – use complex multi layer structure.

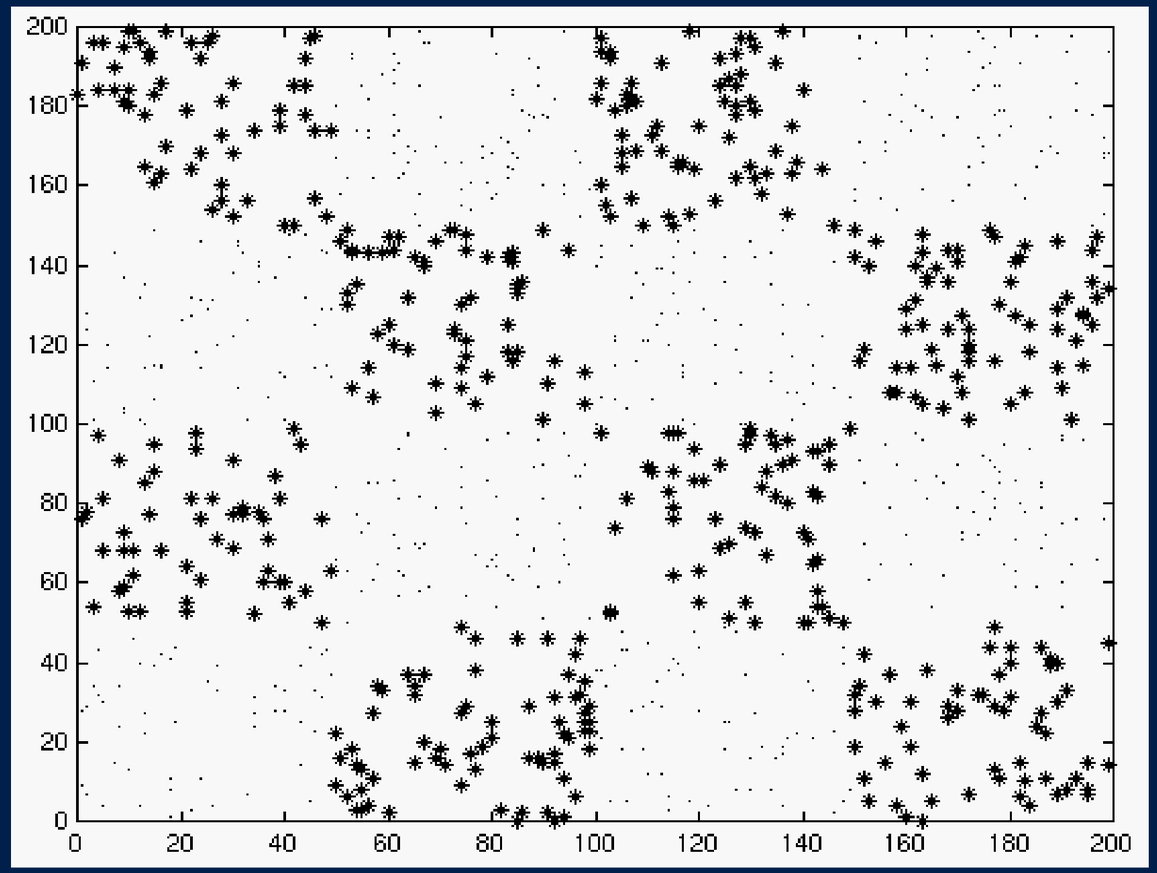
Example of Non-linear SVM



Results

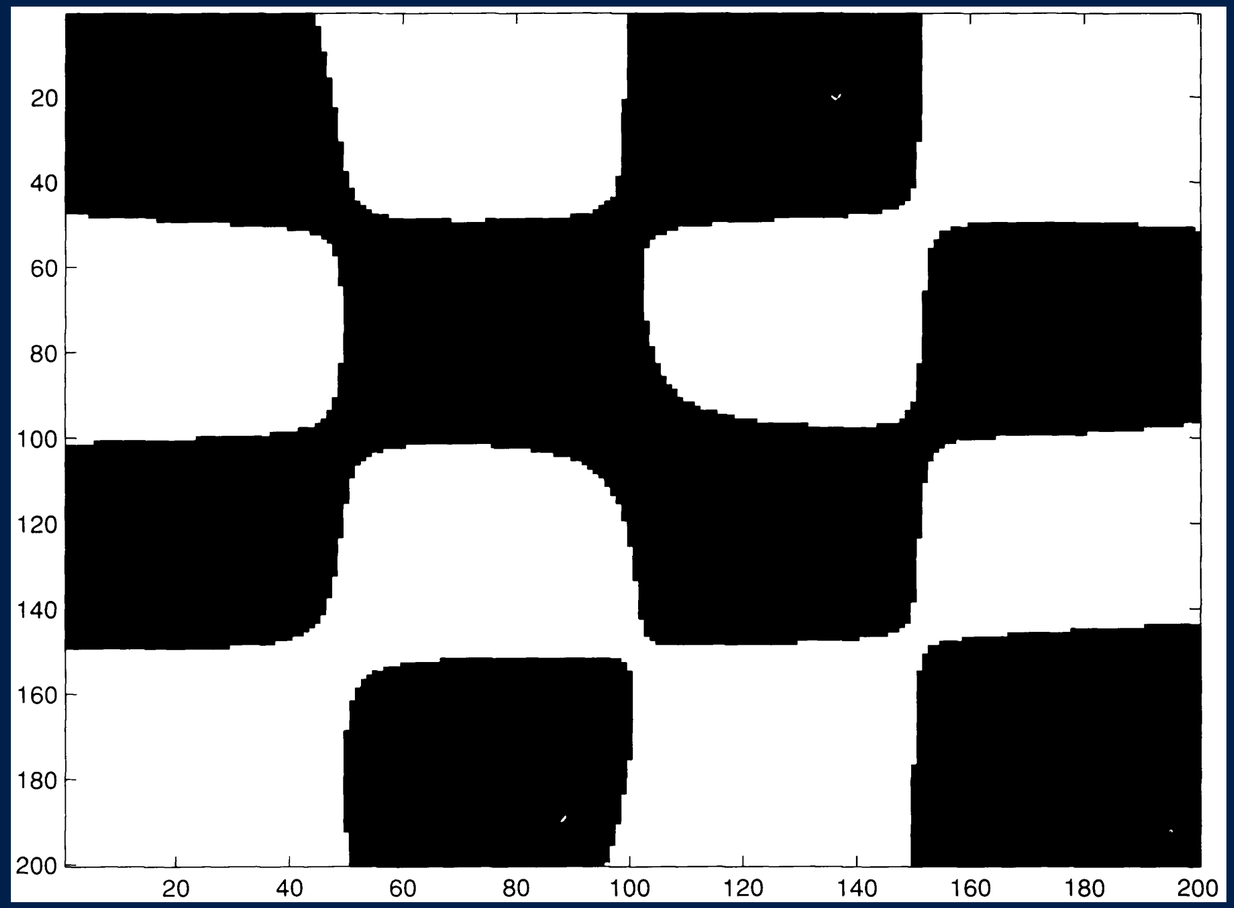

Number of Support Vectors: 9 (-ve: 3, +ve: 6) Total number of points: 60
- Points (+ve, -ve)
- Support Vectors (+ve, -ve)
- Hyperplane (boundary)

A Nonlinear Kernel Application

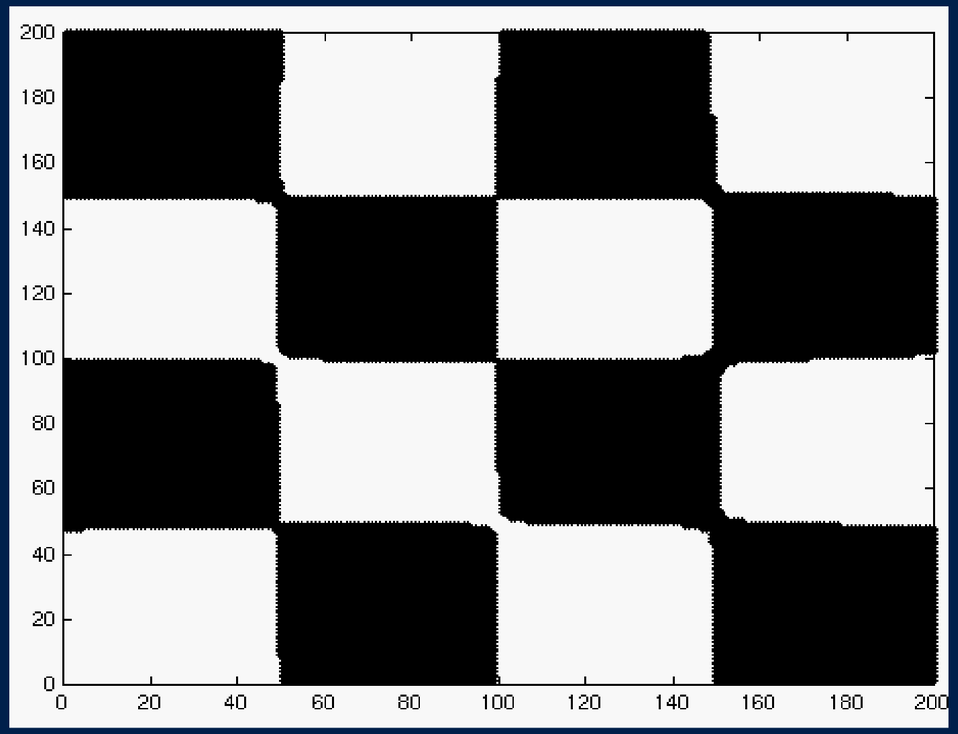


Checkerboard Training Set: 1000 Points in ***R2***
Separate 486 Asterisks from 514 Dots

Previous Work



Polynomial Kernel



$$K(A; A^{9}) = ((\frac{A}{100} \grave{a}\ 1)(\frac{A^{\upsilon}}{100} \grave{a}\ 1) \grave{a}\ 0{:}5)^{6}_{\ddot{\imath}}$$

SVM Applications

- Pattern Recognition
  - **handwriting recognition**
  - **3D object recognition**
  - **speaker identification**
  - **face detection**
  - **text categorization**
  - **bio-informatics**
- Regression estimation or function learning.
- More…

References

- [1] Debprakash Patnaik M.E (SSA)
- [2] C.J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, 1998
- [3] P.S. Sastry, “An Introduction to Support Vector Machine”
- [4] J. Platt, “Sequential minimal optimization: A fast algorithm for training support vector machines”, 1999

Institut Informatika & Bisnis DARMAJAYA
Yayasan Alfian Husin

SAINS DATA
Institut Informatika & Bisnis Darmajaya

Kampus Merdeka INDONESIA JAYA

MERDEKA BELAJAR

THANK YOU!!

DATA SCIENCE DARMAJAYA “YOUR BEST FUTURE IN DATA”