STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

CPE/EE 695: Applied Machine Learning

Lecture 6-1: Support Vector Machines (SVM)

Dr. Shucheng Yu, Associate Professor
Department of Electrical and Computer Engineering
Stevens Institute of Technology

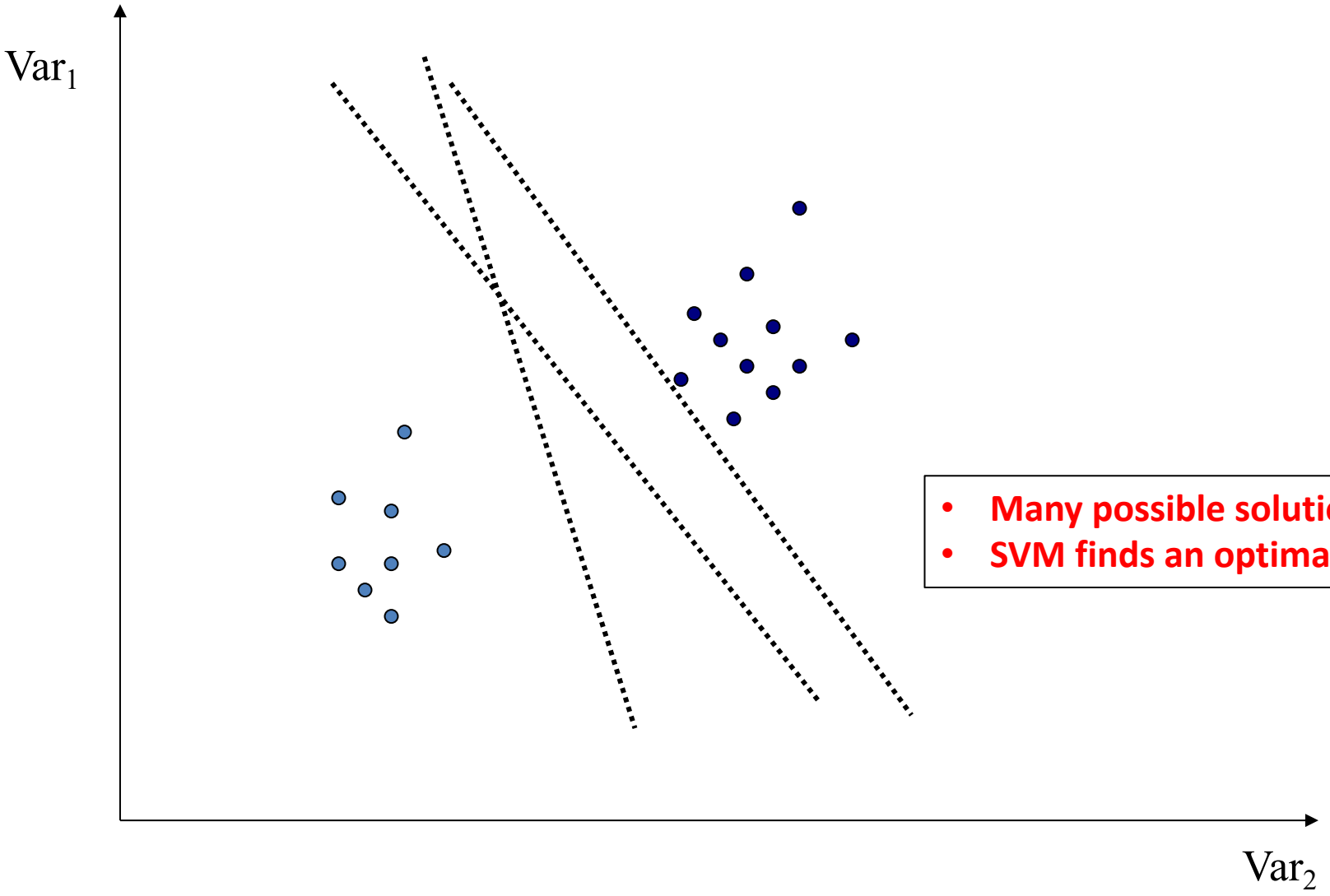
Background

Goal: to find an optimal decision surface (i.e., a hyperplane) in feature space to classify instances.

Key ideas:

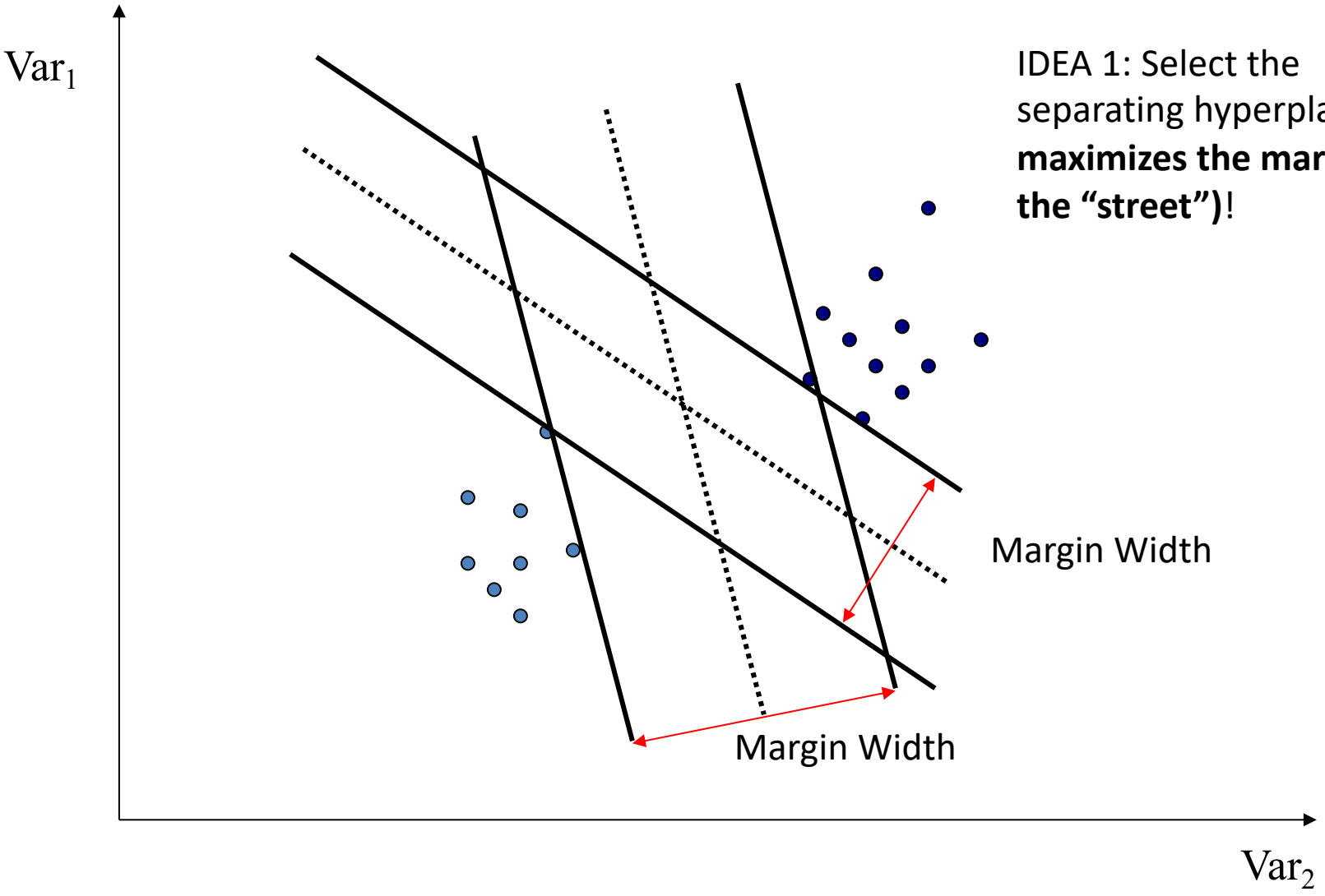
- Separability of non-linear regions using “kernel functions”
- Using quadratic optimization problem to avoid “local minimum”

Linear Classifier



- Many possible solutions;
- SVM finds an optimal one.

Approach: Maximizing the Margin



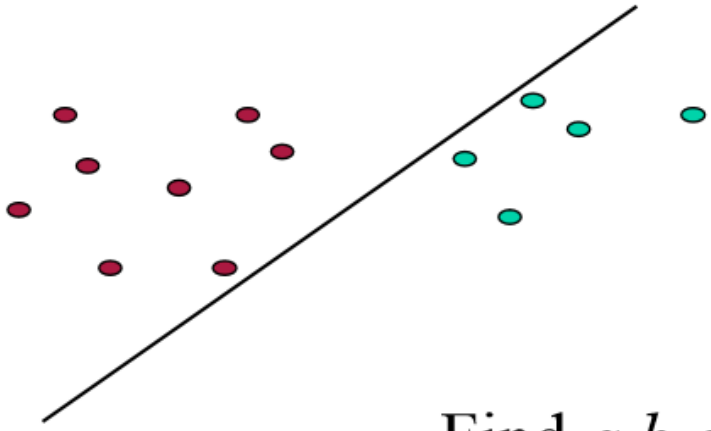
IDEA 1: Select the separating hyperplane that **maximizes the margin (or the "street")**!

SVM Algorithm Input / Output

Input: set of (input, output) training pair samples; call the input sample features x_1, $x_2 \ldots x_n$, and the output result y. Typically, there can be lots of input features x_i.

Output: set of weights **w** (or w_i), one for each feature, whose linear combination predicts the value of y. (So far, just like neural nets…)
Important difference: we use the optimization of maximizing the margin ('street width') to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matter' in deciding the separating line(hyperplane)…these nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane)

2-D Case



Find a,b,c, such that

$ax + by \geq c$ for red points

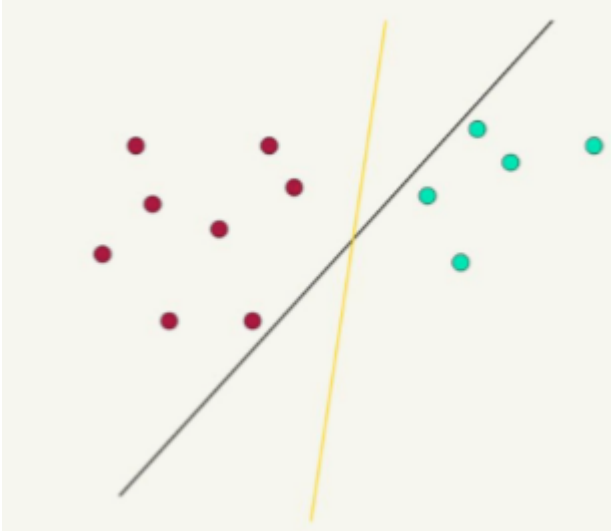
$ax + by \leq (\text{or} <)\ c$ for green points.

Finding Optimal Hyperplane

Lots of possible solutions for a, b, c;

Which points influence optimality?

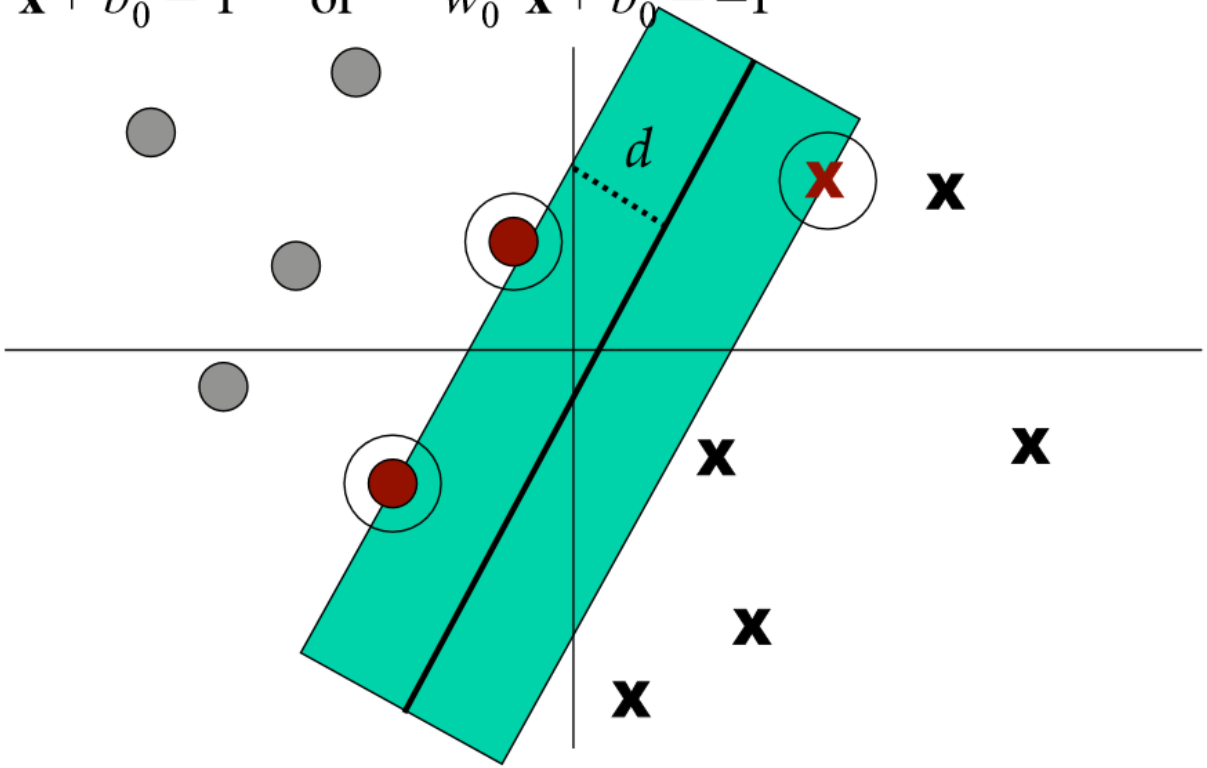
- All point? (Linear regression, NN)
- Or only “difficult points” closest to decision boundary (SVM)



Support Vectors

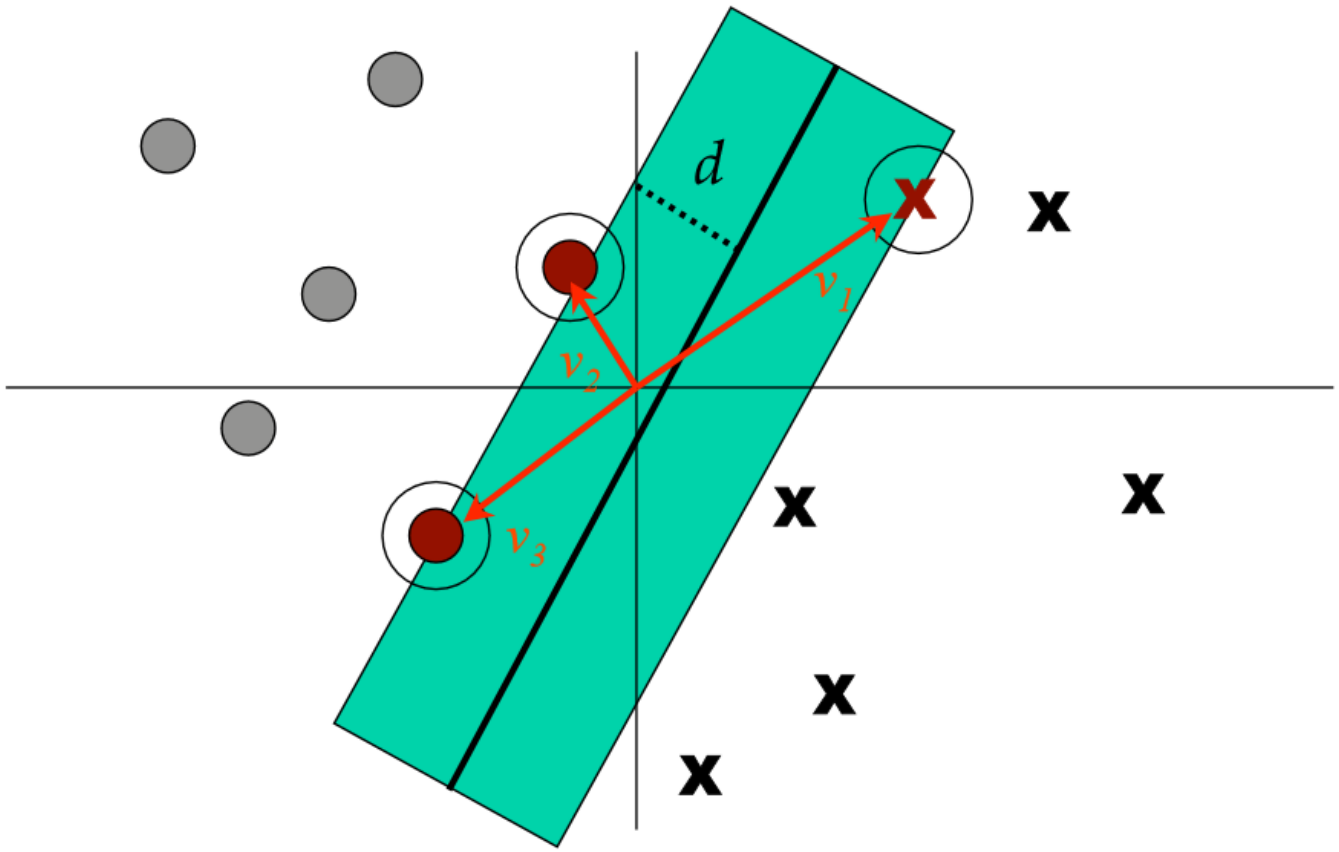
Support Vectors: Input vectors that just touch the boundary of the margin (street) – circled below, there are 3 of them (or, rather, the 'tips' of the vectors

$w_0^T\mathbf{x} + b_0 = 1$ or $w_0^T\mathbf{x} + b_0 = -1$

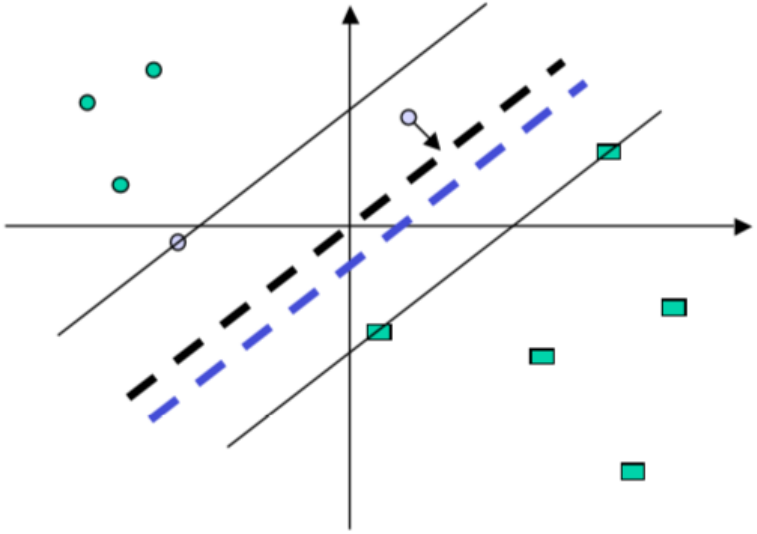


Support Vectors

Actual support vectors v_1, v_2, v_3.

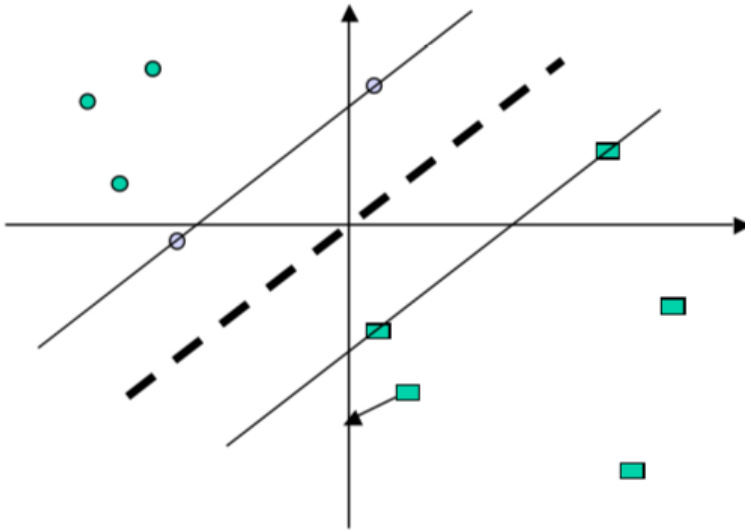


Support Vectors



Moving a support vector moves the decision boundary

Moving the other vectors has no effect



Definition of Hyperplanes

Define the hyperplanes H such that:

$w \bullet x_i + b \geq +1$ when $y_i = +1$

$w \bullet x_i + b \leq -1$ when $y_i = -1$

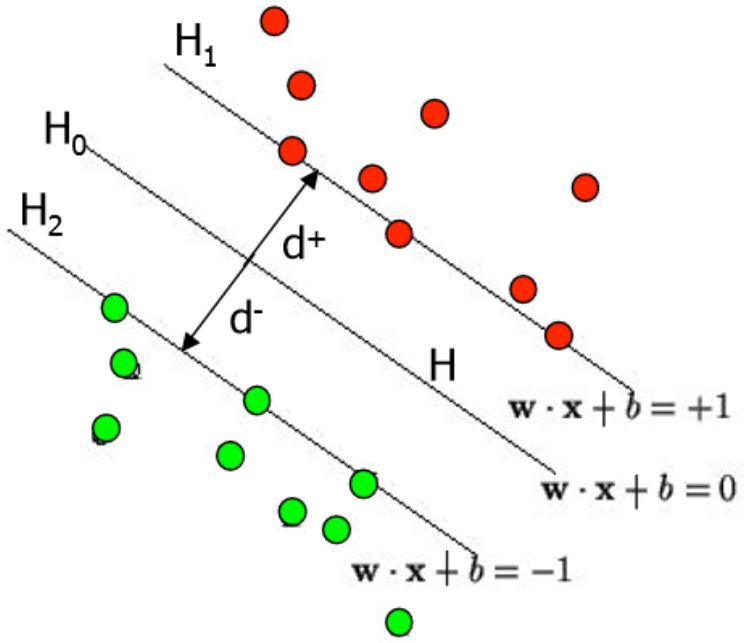
H_1 and H_2 are the planes:

H_1: $w \bullet x_i + b = +1$

H_2: $w \bullet x_i + b = -1$

The points on the planes H_1 and H_2 are the tips of the Support Vectors

The plane H_0 is the median in between, where $w \bullet x_i + b = 0$



d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin (gutter) of a separating hyperplane is d+ + d–.

Defining the Separating Hyperplane

- Form of equation defining the decision surface separating the classes is a hyperplane of the form:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

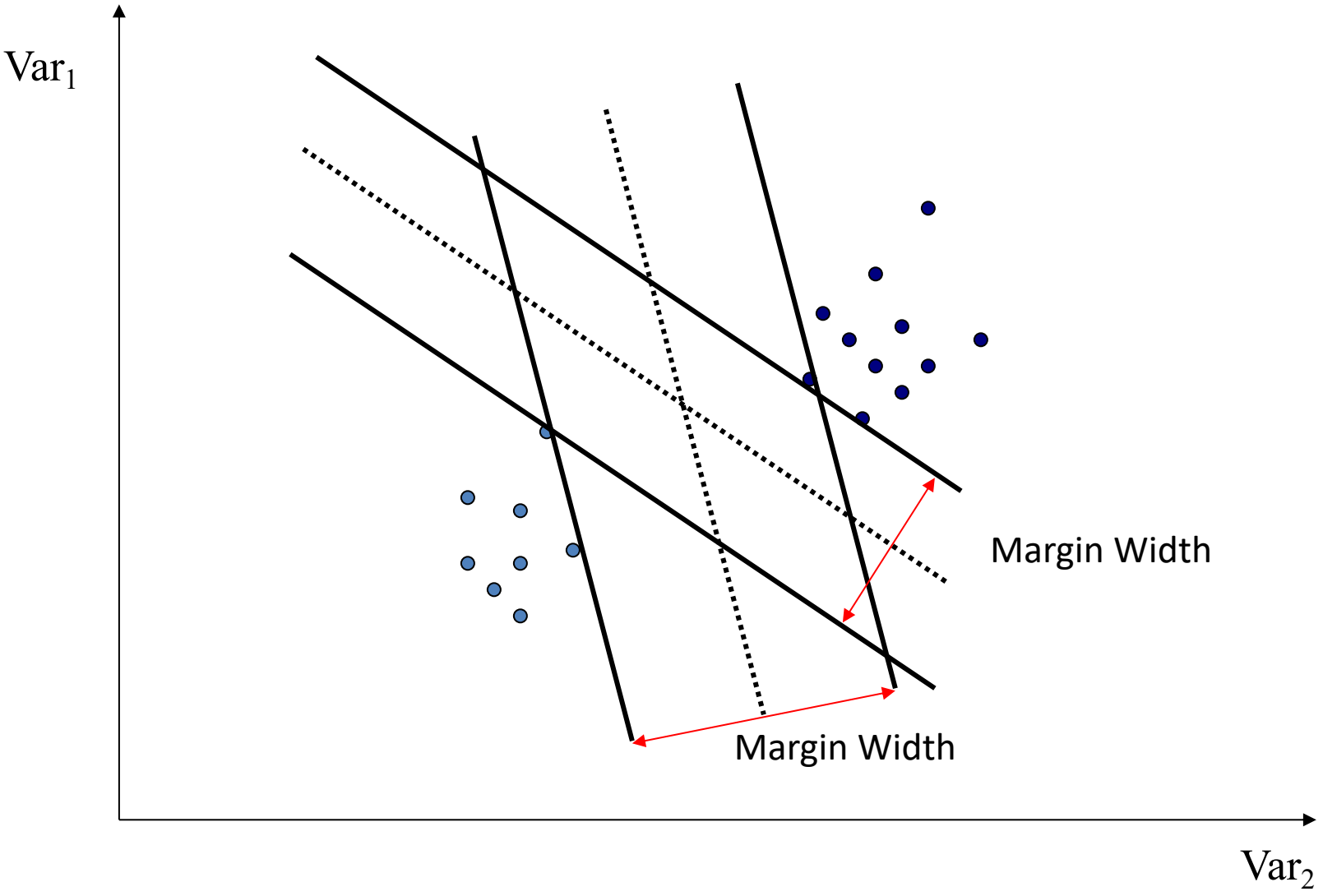
  - **w is a weight vector**
  - **x is input vector**
  - **b is bias**
- Allows us to write

$$\mathbf{w}^T\mathbf{x} + b \geq 0 \text{ for } d_i = +1$$
$$\mathbf{w}^T\mathbf{x} + b < 0 \text{ for } d_i = -1$$

Defining the Separating Hyperplane

- Margin of Separation (d): the separation between the hyperplane and the closest data point for a given weight vector $\mathbf{w}$ and bias b.
- Optimal Hyperplane (maximal margin): the particular hyperplane for which the margin of separation d is maximized.

Maximizing the Margin

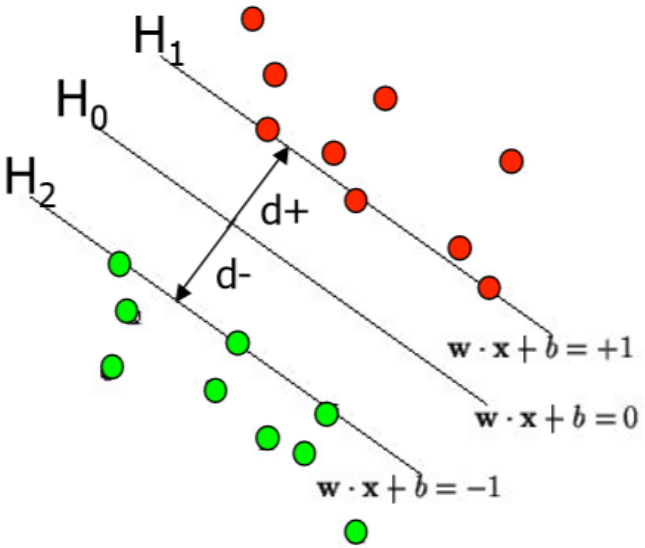


Maximizing the Margin

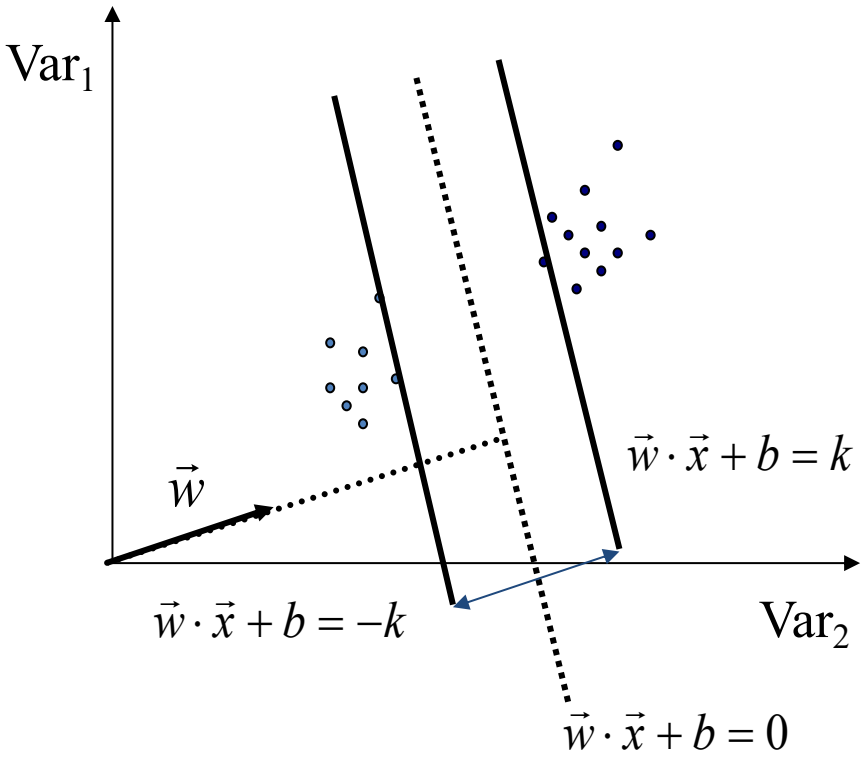
Recall the distance from a point(x_0,y_0) to a line:
$Ax+By+c = 0$ is: $|Ax_0 +By_0 +c|$/sqrt(A^2+B^2), so,
The distance between H_0 and H_1 is then:
$|w\bullet x+b|/||w||=1/||w||$, so

The total distance between H_1 and H_2 is thus: $2/||w||$

where $\| w \| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$



Setting Up the Optimization Problem for SVM



The width of the margin is the distance between two Hyperplanes:

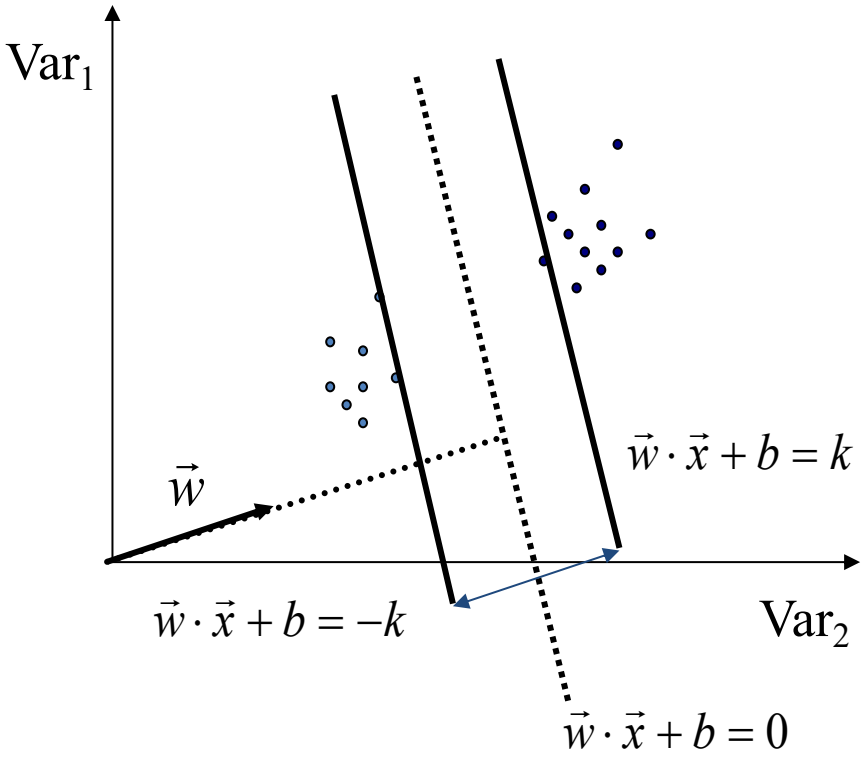
$$\vec{w}\cdot\vec{x}+b=k \qquad \vec{w}\cdot\vec{x}+b=-k$$

Which is: $\dfrac{2|k|}{\|w\|}$

where $\|w\|=\sqrt{w_1^2+w_2^2+\ldots+w_n^2}$

Recall the distance from a point(x_0,y_0) to a line: $Ax+By+c=0$ is: $|Ax_0+By_0+c|/\text{sqrt}(A^2+B^2)$

Setting Up the Optimization Problem for SVM



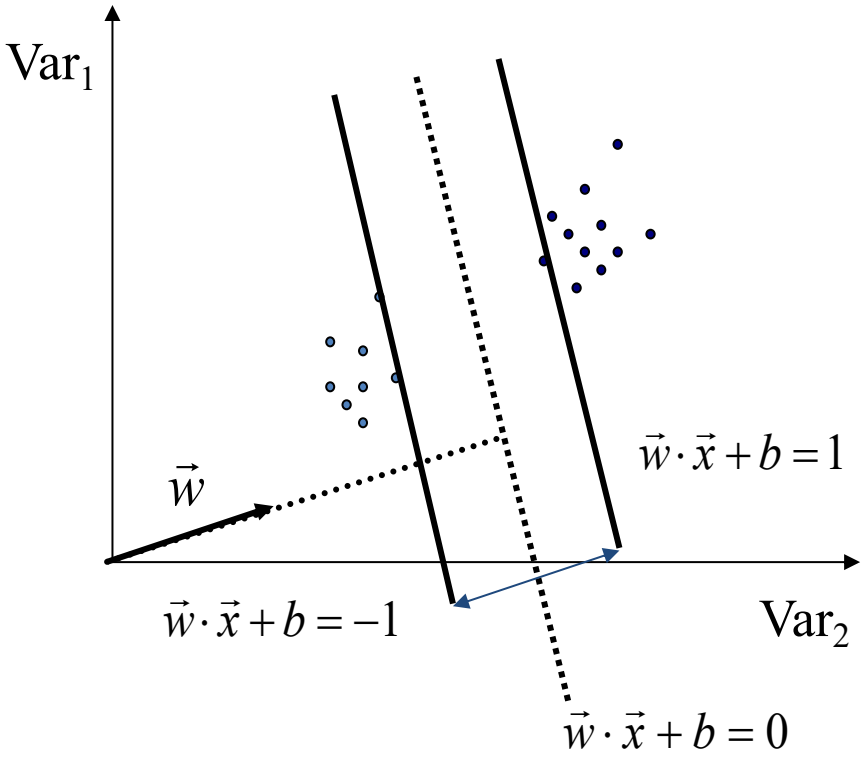
So, the problem is:

$$\max \frac{2|k|}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq k,\ \forall x \text{ of class 1}$$

$$(w\cdot x+b)\leq -k,\ \forall x \text{ of class 2}$$

Setting Up the Optimization Problem for SVM



There is a scale and unit for data so that *k=1*. Then problem becomes:

$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq 1,\ \ \forall x \text{ of class 1}$$

$$(w\cdot x+b)\leq -1,\ \ \forall x \text{ of class 2}$$

Setting Up the Optimization Problem for SVM

- If class 1 corresponds to 1 and class 2 corresponds to -1, we can rewrite

$$(w \cdot x_i + b) \geq 1, \ \forall x_i \ \text{with} \ y_i = 1$$
$$(w \cdot x_i + b) \leq -1, \ \forall x_i \ \text{with} \ y_i = -1$$

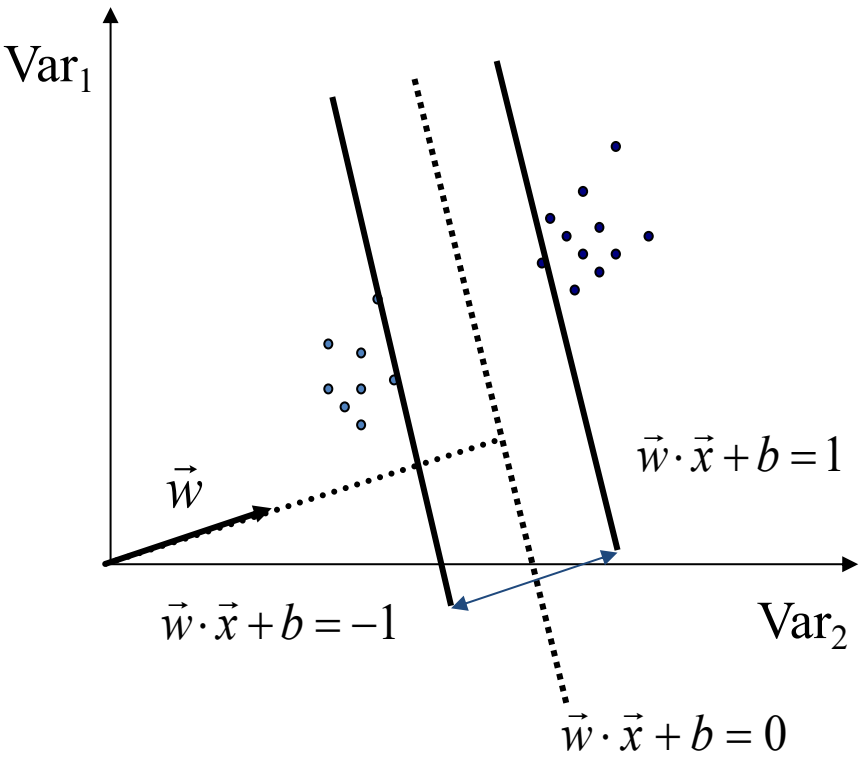
- as

$$y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

- So the problem becomes:

$$\max \frac{2}{\|w\|} \quad \text{or} \quad \min \frac{1}{2}\|w\|^2$$
$$s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i \qquad s.t. \ y_i (w \cdot x_i + b) \geq 1, \ \forall x_i$$

Computing the margin width

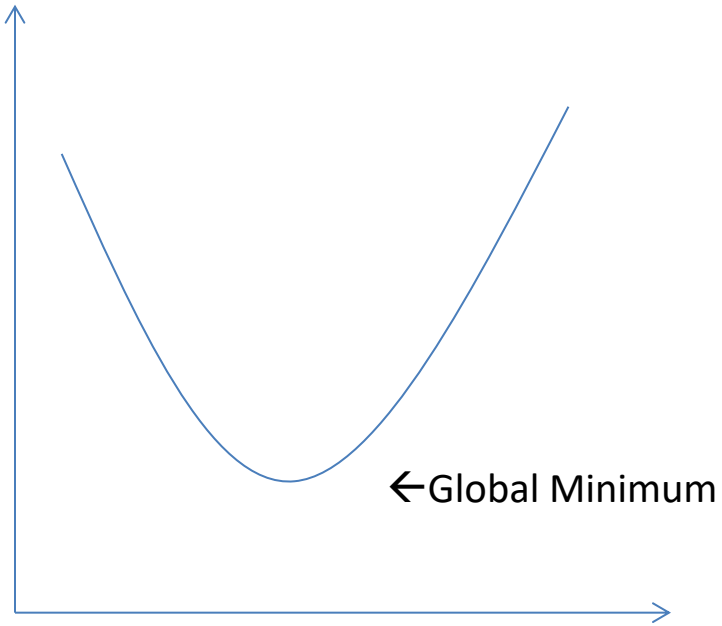


now we just need to write a program to search the space of **w**'s and ***b***'s to find the **maximum margin** that matches all the data points. *How?*

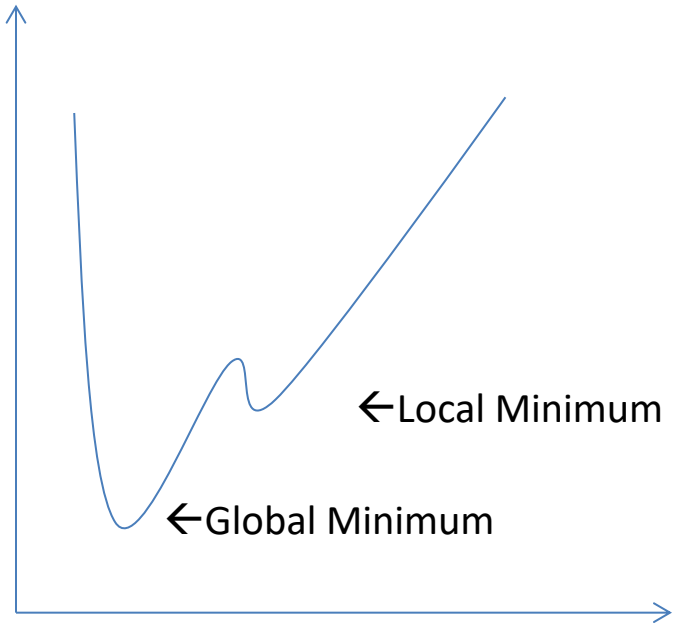
Basics of optimization: Convex functions

A function is called ***convex*** *if the function lies below the straight line segment connecting two points, for any two points in the interval.*

Property: Any local minimum is a global minimum!



Convex Function



Non-Convex Function

Basics of optimization: Quadratic programming (QP)

- Quadratic programming (QP) is a special optimization problem: the function to optimize ("*objective*") *is quadratic, subject to linear constraints.*
- Convex QP problems have convex objective functions.
- These problems can be solved easily and efficiently by **greedy algorithms** (because every local minimum is a global minimum).

Lagrangian Formulation

- So in the SVM problem the Lagrangian is

$$\min L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i, a_i \geq 0$ where l is the # of training points

- From the property that the derivatives at min = 0 we get:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} a_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{l} a_i y_i = 0 \quad \text{so}$$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

Primal problem:

$$\min L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

s.t. $\forall i \; a_i \geq 0$

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

Dual problem:

$$\max L_D(a_i) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} a_i a_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

s.t. $\sum_{i=1}^{l} a_i y_i = 0 \;\&\; a_i \geq 0$

(note that we have removed the dependence on **w** and b)

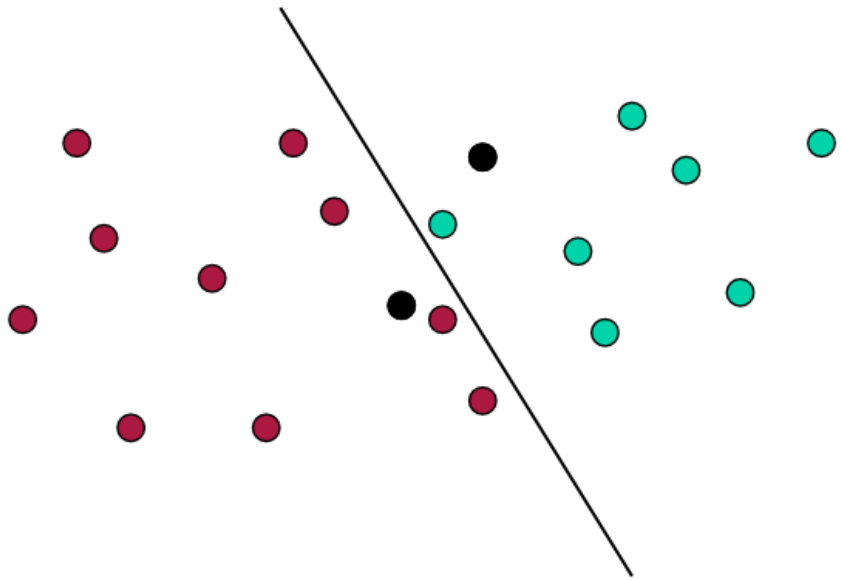
Knowing a_i's, w can be calculated as:

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$$

And now, after training and finding the **w** by this method, given an unknown point u measured on features x_i we can classify it by looking at the sign of:

$$f(x) = \mathbf{w} \bullet \mathbf{u} + b = (\sum_{i=1}^{l} a_i y_i \mathbf{x_i} \bullet \mathbf{u}) + b$$

Non-Separable Case



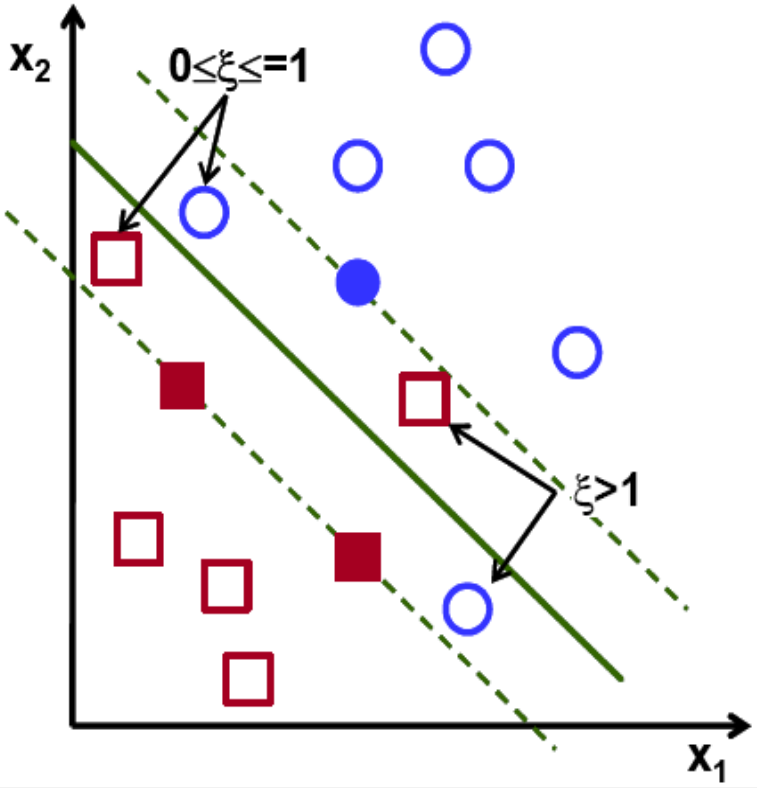
Find a line that penalizes points on “the wrong side”

Non-Separable Case

- **The solution for the non-separable case is to introduce slack variables ξ_i that relax the constraints of the canonical hyperplane equation**

$$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \forall i = 1 \ldots N$$

- **The slack variables measure deviation from the ideal condition**
  - For $0 \leq \xi \leq 1$, the data point falls on the right side of the separating hyperplane but within the region of maximum margin
  - For $\xi > 1$, the data point falls on the wrong side of the separating hyperplane



Non- separable case

Using similar approach as we just discussed

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

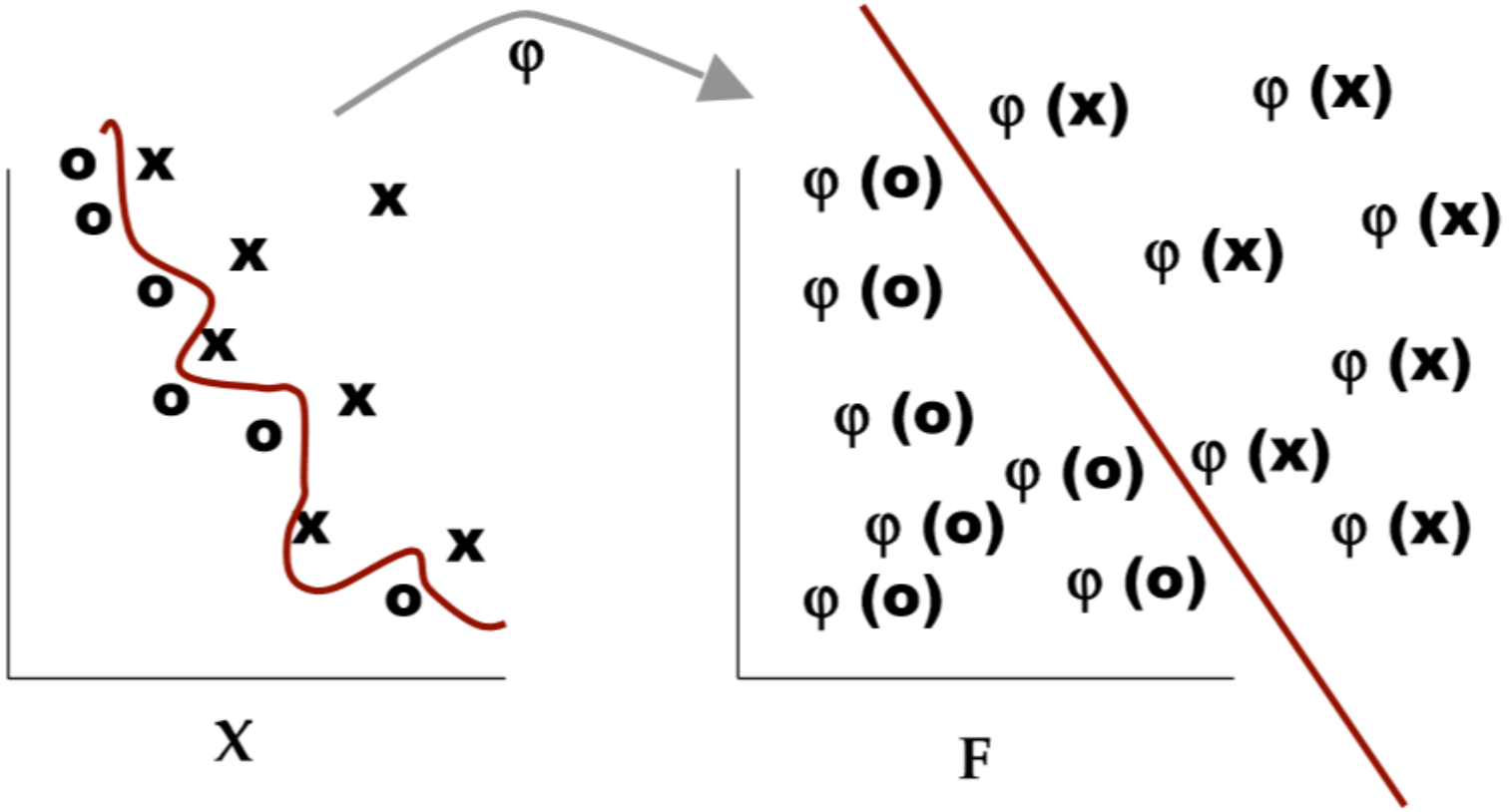
subject to the constraints $\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \quad i = 1 \ldots N \end{cases}$

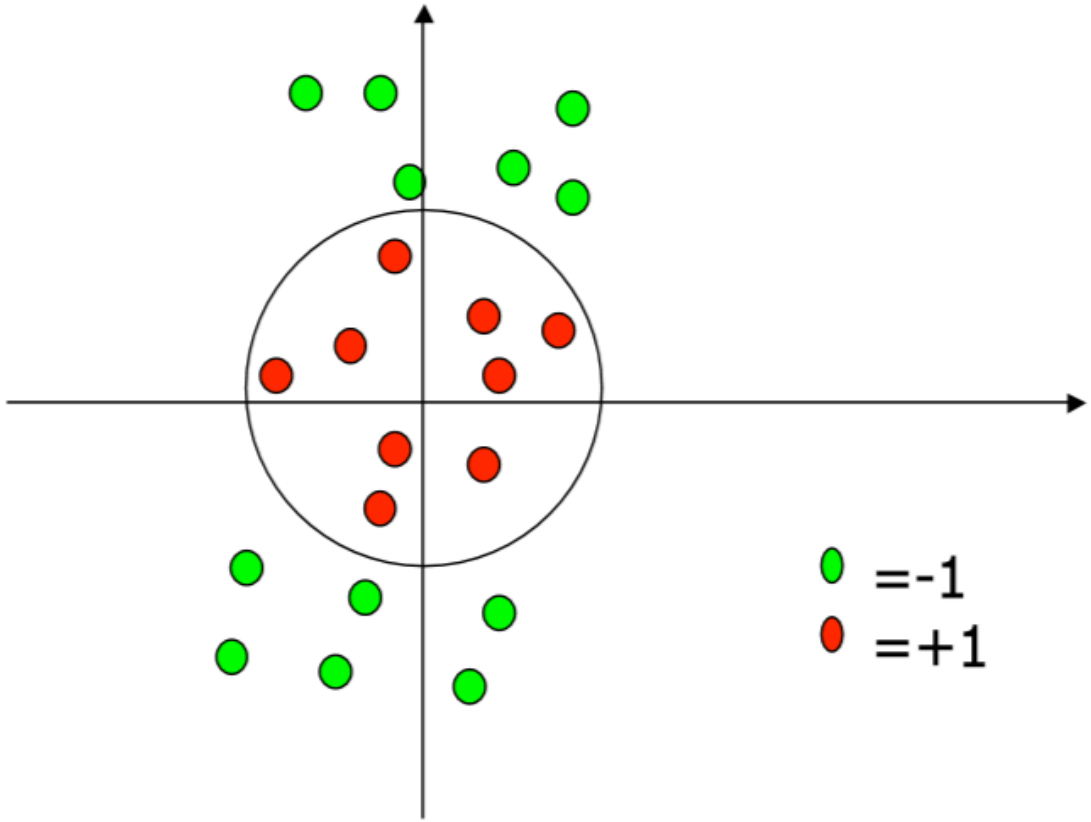
Comments

- Notice that neither the slack variables nor their associated Lagrange multipliers appear in the formulation of the dual problem
- Therefore, this represents the same optimization problem as the linearly separable case, with the exception that the constraints $\alpha_i \ge 0$ have been replaced by the more restrictive constraints $0 \le \alpha_i \le C$
  - The optimum solution for the weight vector remains the same:

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

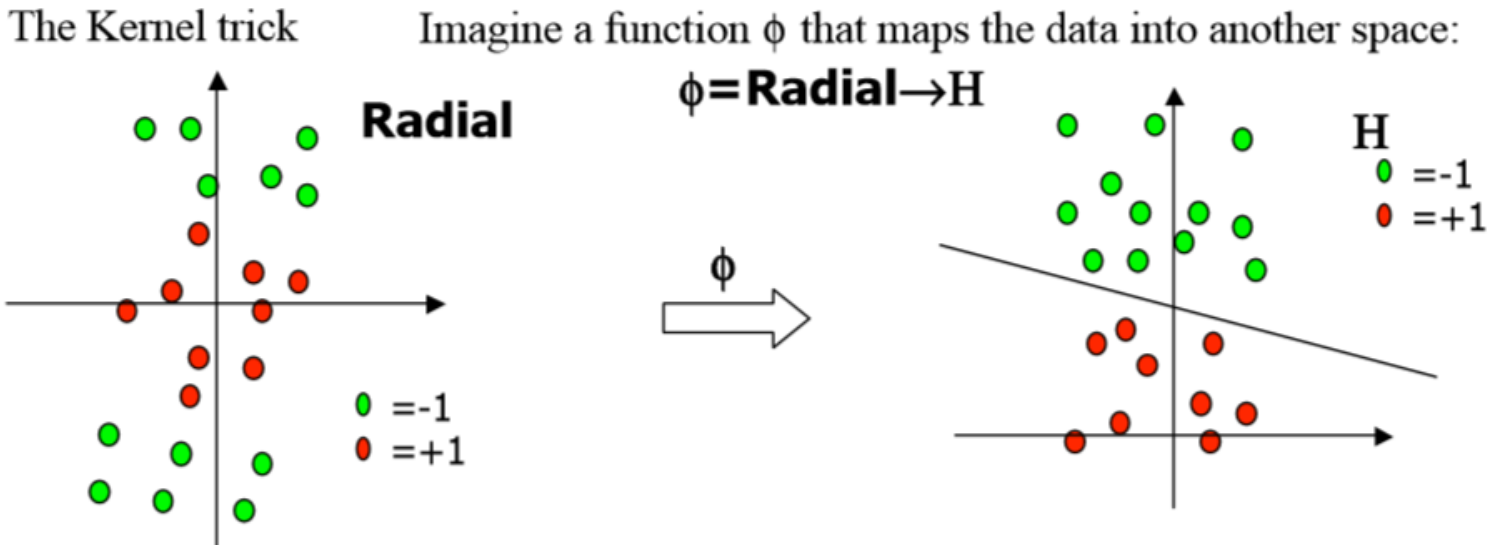
Another Approach: Transform to Separate





The decision function may be non-linear

The Kernel Trick



So, the function we end up optimizing is:

$$L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j K(x_i \bullet x_j),$$

The kernel function defines the "inner product" (similarity) in transformed space.

Popular Kernels

Examples:

1. Linear kernel $K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \bullet \vec{x}_j$
2. Gaussian kernel $K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2$
3. Exponential kernel $K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|$
4. Polynomial kernel $K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \bullet \vec{x}_j)^q$
5. Hybrid kernel $K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \bullet \vec{x}_j)^q \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2$
6. Sigmoidal $K(\vec{x}_i, \vec{x}_j) = \tanh(k\vec{x}_i \bullet \vec{x}_j - \delta)$

Note that one can define kernels over more than just vectors: strings, trees, structures, … in fact, just about anything

A very powerful idea: used in comparing DNA, protein structure, sentence structures, etc.

Acknowledgement

Part of the slide materials were based on Dr. Rong Duan's Fall 2016 course CPE/EE 695A Applied Machine Learning at Stevens Institute of Technology.

Part of the materials are from the following sources:

R. Berwick and Village Idiot, "An Idiot's guide to Support vector machines (SVMs)", http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

STEVENS

INSTITUTE *of* TECHNOLOGY

THE INNOVATION UNIVERSITY®

stevens.edu