Support Vector Machines

Outline

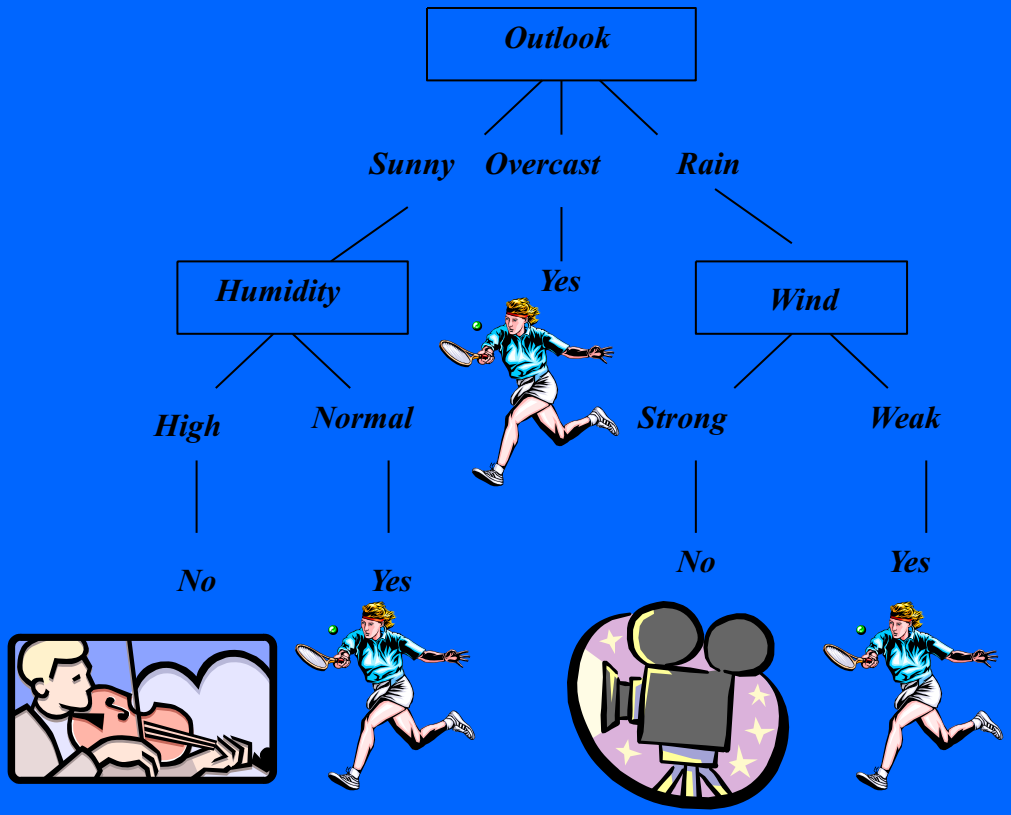
- What do we mean with classification, why is it useful
- Machine learning- basic concept
- Support Vector Machines (SVM)
  - Linear SVM – basic terminology and some formulas
  - Non-linear SVM – the Kernel trick
- An example: Predicting protein subcellular location with SVM
- Performance measurments

Classification

- Everyday, all the time we classify things.
- Eg crossing the street:
  - Is there a car coming?
  - At what speed?
  - How far is it to the other side?
  - Classification: Safe to walk or not!!!

- **Decision tree learning**

IF (*Outlook = Sunny*) ^ (*Humidity = High*)
THEN *PlayTennis =NO*
IF (*Outlook = Sunny)^ (Humidity = Normal)*
THEN *PlayTennis = YES*



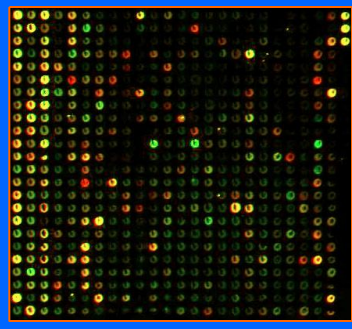
Classification tasks

- Learning Task
  - Given: Expression profiles of leukemia patients and healthy persons.
  - Compute: A model distinguishing if a person has leukemia from expression data.
- Classification Task
  - Given: Expression profile of a new patient + a learned model
  - Determine: If a patient has leukemia or not.

Problems in classifying data

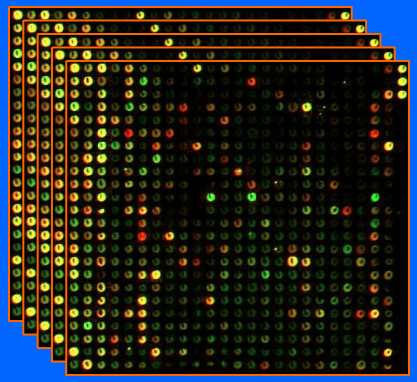
- Often high dimension of data.
- Hard to put up simple rules.
- Amount of data.
- Need automated ways to deal with the data.
- Use computers – data processing, statistical analysis, try to learn patterns from the data (Machine Learning)

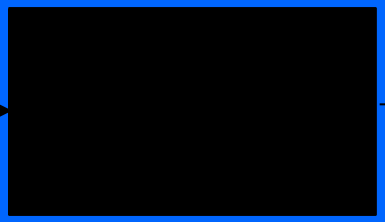
Black box view of Machine Learning



Test-data

Training data





Magic black box
(learning machine)

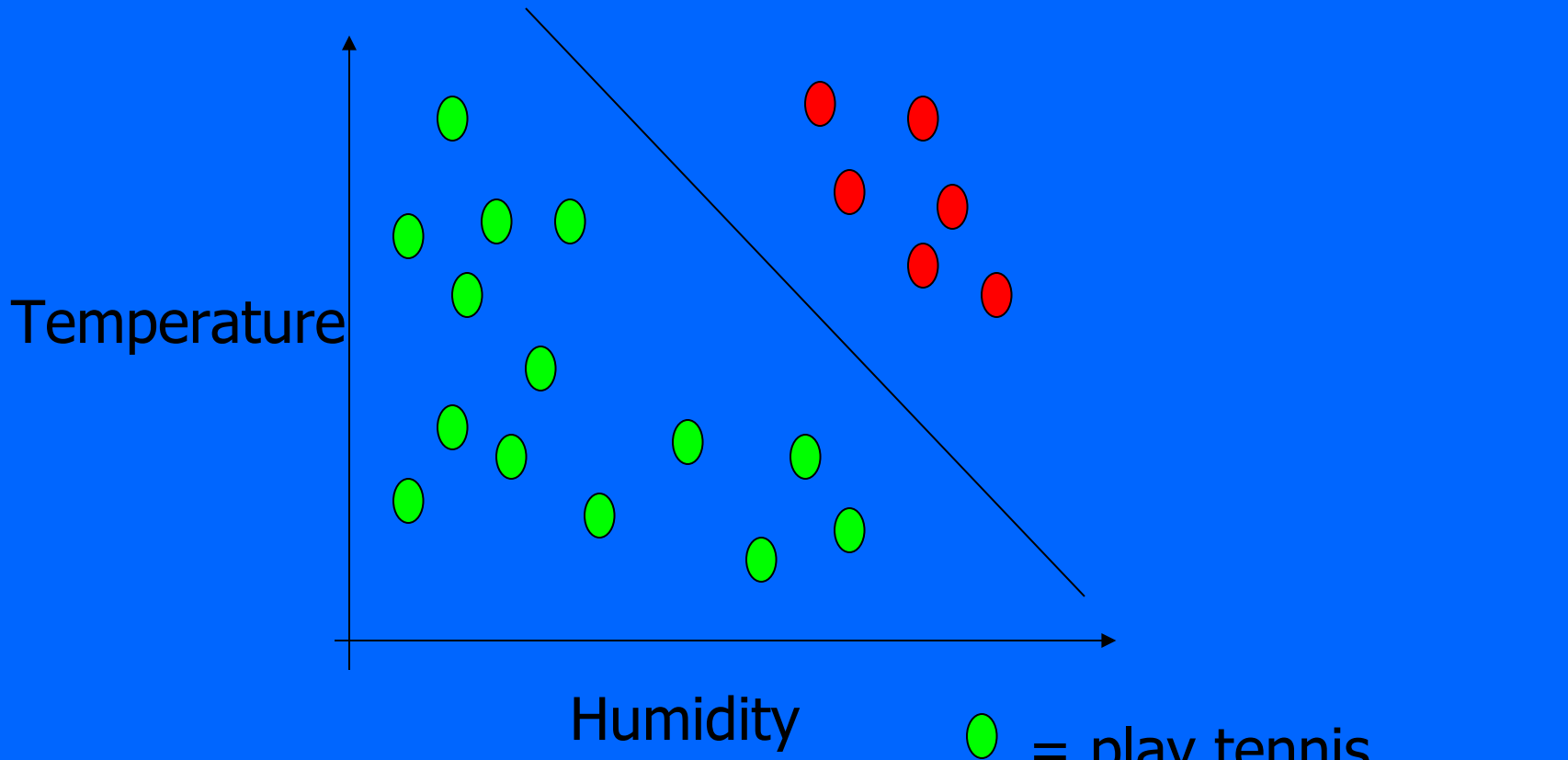
Model

Prediction = cancer or not

Training data: -Expression patterns of some cancer +
expression data from healty person

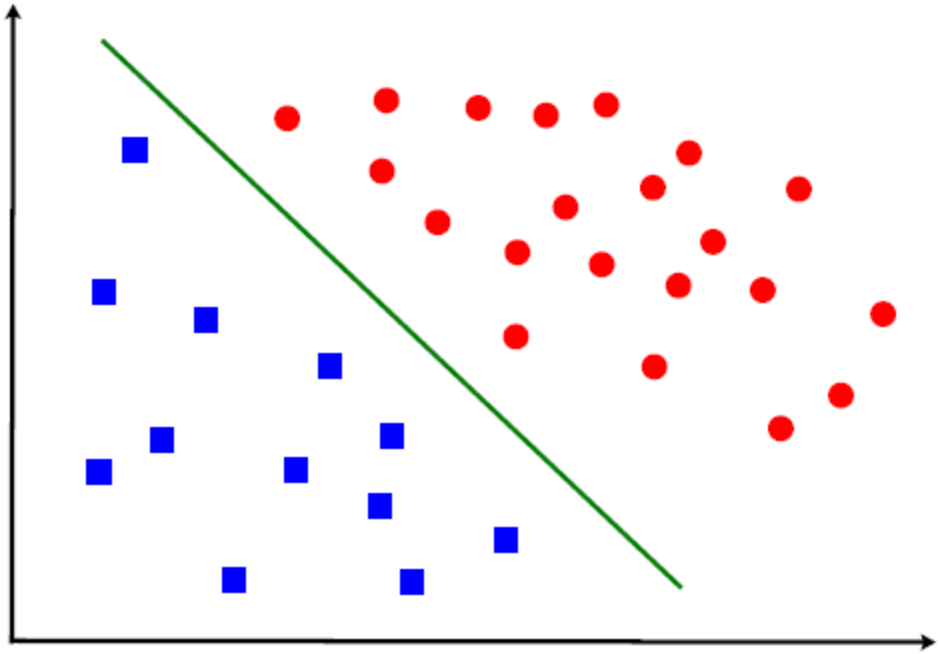
Model: - The model can distinguish between healty
and sick persons. Can be used for prediction.

Tennis example 2



Linearly Separable Classes

$$Y = \text{sgn}[\mathbf{w}^\top \mathbf{x} + b > 0] = \begin{cases} +1 & \mathbf{w}^\top \mathbf{x} + b > 0 \\ -1 & \mathbf{w}^\top \mathbf{x} + b \leq 0 \end{cases}$$

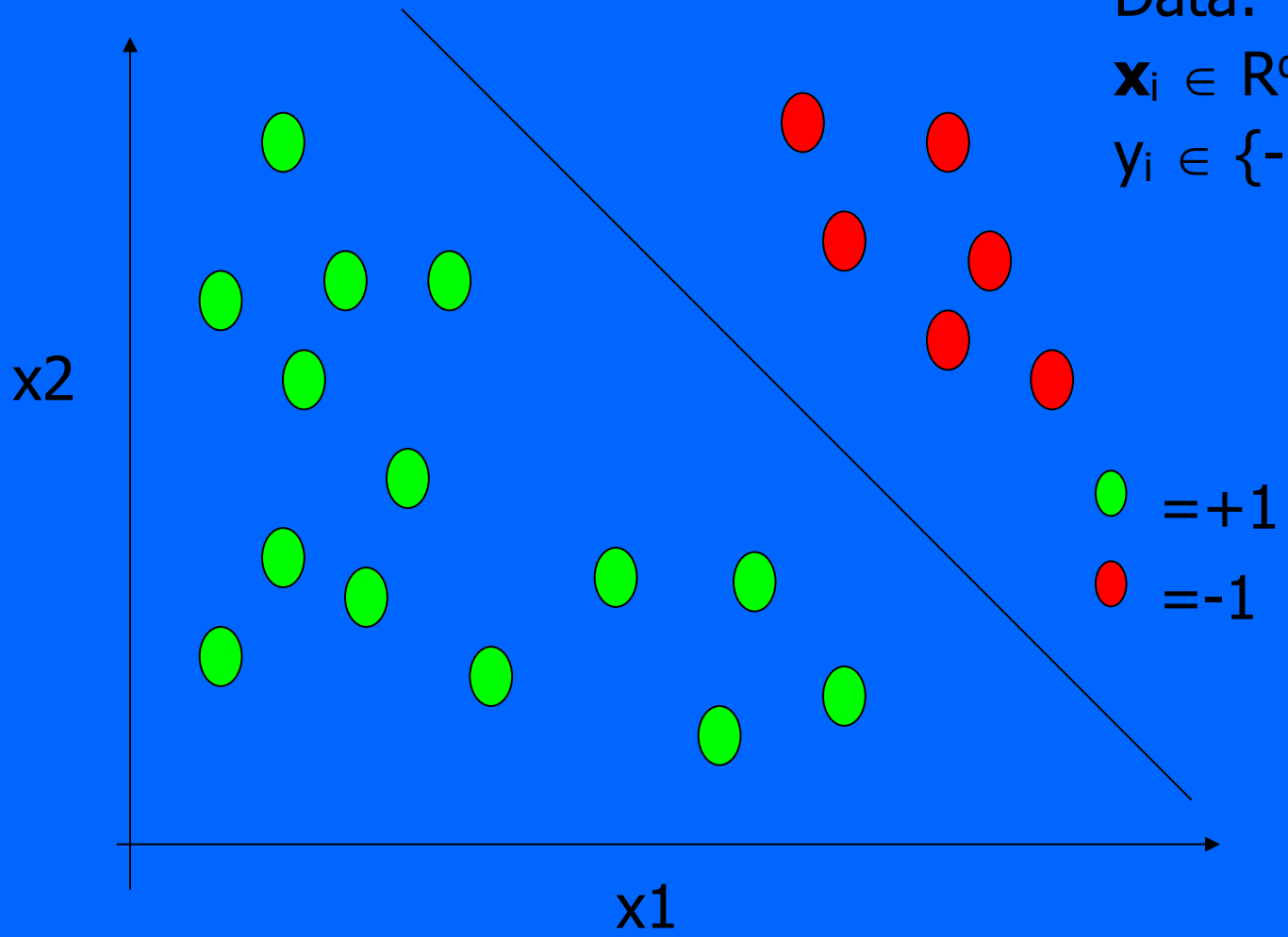


Linear Support Vector Machines

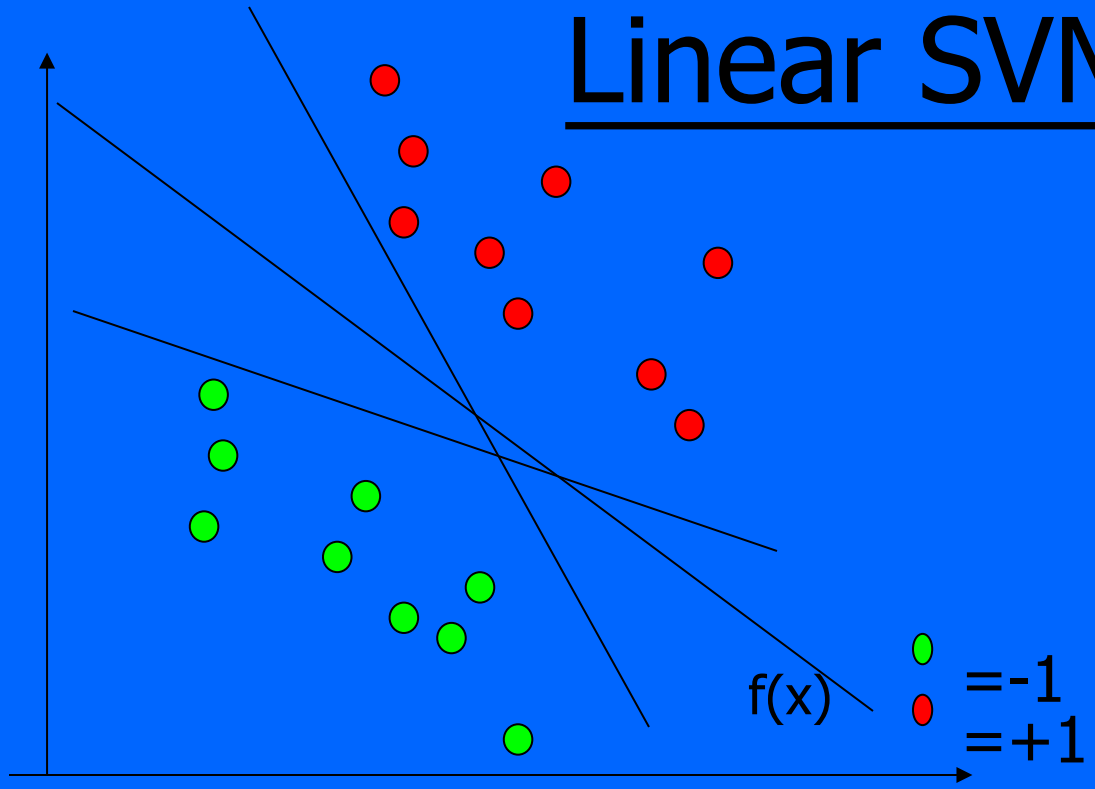
Data: $<\mathbf{x}_i, y_i>$, i=1,..,l

$\mathbf{x}_i \in R^d$

$y_i \in \{-1,+1\}$



Linear SVM 2



Data: $<\mathbf{x}_i, y_i>$, i=1,..,l

$\mathbf{x}_i \in R^d$

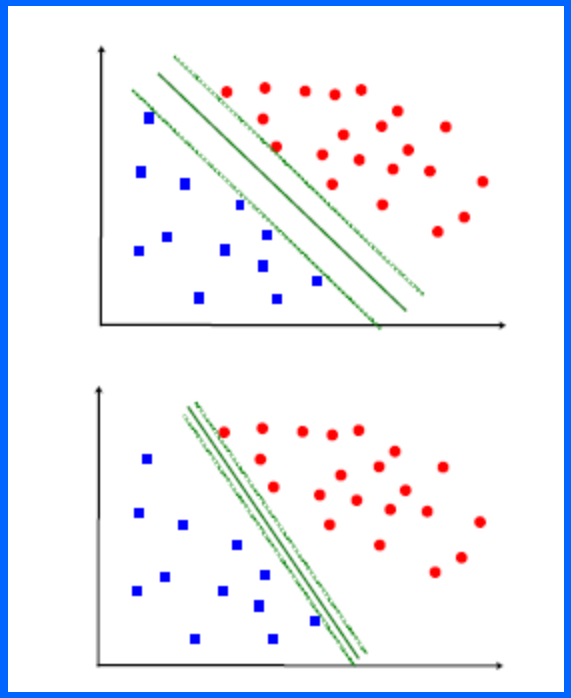
$y_i \in \{-1,+1\}$

All hyperplanes in R^d are parameterized by a vector ($\mathbf{w}$) and a constant b. Can be expressed as $\mathbf{w} \bullet \mathbf{x} + b = 0$ (remember the equation for a hyperplane from algebra!)

Our aim is to find such a hyperplane $f(x) = sign(\mathbf{w} \bullet \mathbf{x} + b)$, that correctly classify our data.

Selection of a Good Hyper-Plane

Objective: Select a `good' hyper-plane using only the data!
Intuition:
(Vapnik 1965) - assuming linear separability
(i) Separate the data
(ii) Place hyper-plane `far' from data



Definitions

Define the hyperplane H such that:

$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1$ when $y_i = +1$

$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$ when $y_i = -1$

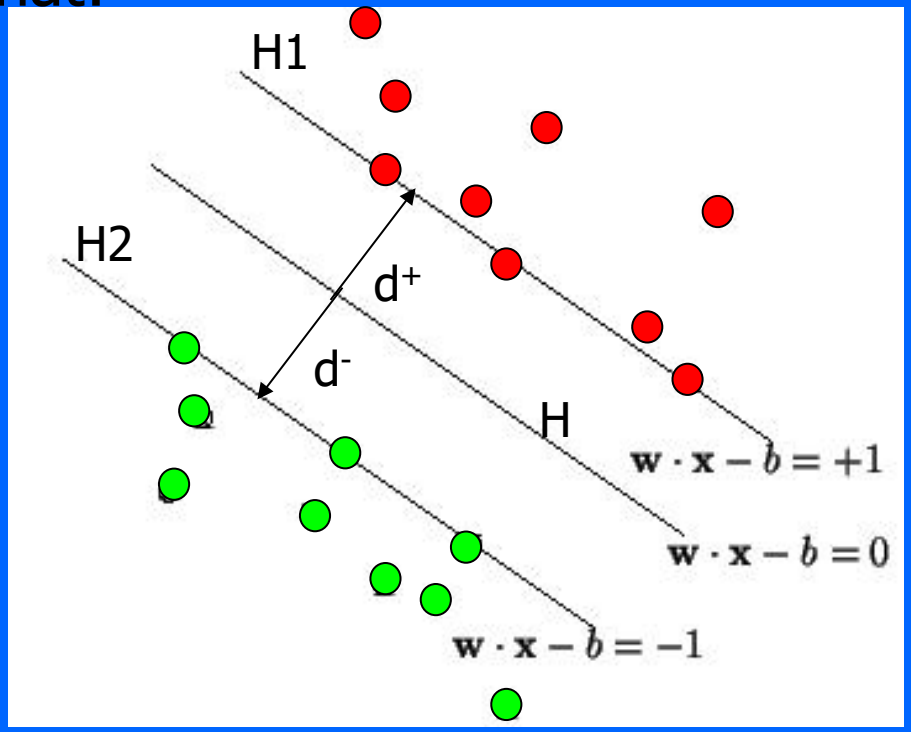
H1 and H2 are the planes:

H1: $\mathbf{x}_i \cdot \mathbf{w} + b = +1$

H2: $\mathbf{x}_i \cdot \mathbf{w} + b = -1$

The points on the planes H1 and H2 are the **Support Vectors**:

$$\{\mathbf{x}_i : |\mathbf{w}^\top \mathbf{x}_i + b| = 1\}$$



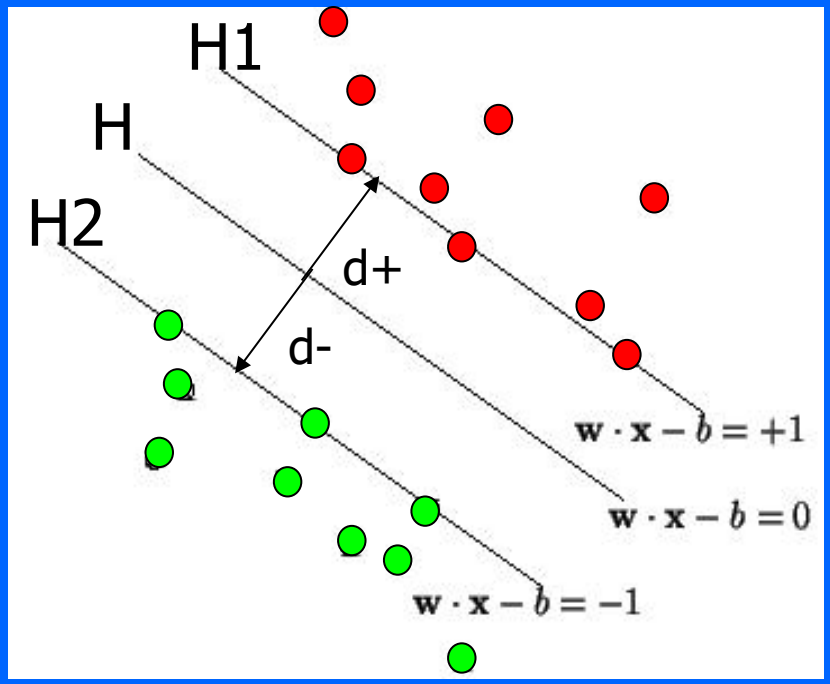
d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $d^+ + d^-$.

Maximizing the margin

We want a classifier with as big margin as possible.



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

The distance between H and H1 is:
$|\mathbf{w}\bullet\mathbf{x}+b|/||w||=1/||w||$

The distance between H1 and H2 is: $2/||w||$

In order to maximize the margin, we need to minimize $||w||$. With the condition that there are no datapoints between H1 and H2:

$\mathbf{x}_i\bullet\mathbf{w}+b \geq +1$ when $y_i = +1$
$\mathbf{x}_i\bullet\mathbf{w}+b \leq -1$ when $y_i = -1$

Can be combined into $y_i(\mathbf{x}_i\bullet\mathbf{w}) \geq 1$

Optimization Problem

Distance from support vector to $H(\mathbf{w}, b)$

$$\frac{\mathbf{w}^\top \mathbf{x}_i + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|}$$

$$\textbf{Margin} = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

minimize	$\frac{1}{2}\mathbf{w}^\top \mathbf{w}$	
subject to	$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$	$i = 1, 2, \ldots, n$

1. Convex quadratic program
2. Linear inequality constraints (many!)
3. $d + 1$ parameters, n constraints

The Lagrangian trick

Reformulate the optimization problem:
A ”trick” often used in optimization is to do an Lagrangian formulation of the problem.The constraints will be replaced by constraints on the Lagrangian multipliers and the training data will only occur as dot products.

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

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad ; \qquad \alpha_i \geq 0$$

What we need to see: $x_i$and x_j (input vectors) appear only in the form of dot product – we will soon see why that is important.

Non-Separable Case

Objective: find a good separating hyper-plane for the non-separable case

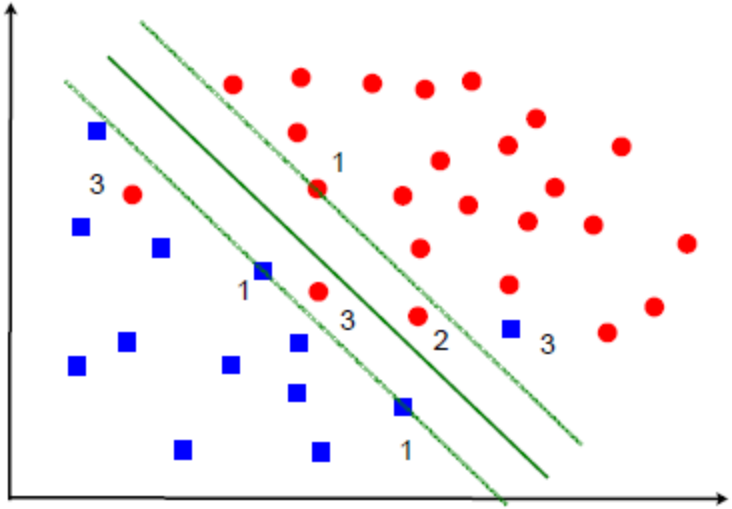
Problem: Cannot satisfy $y_i[\mathbf{w}^\top \mathbf{x}_i + b] \geq 1$ for all i

Solution: *Slack* variables

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq +1 - \xi_i \quad && \text{for } y_i = +1, \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq -1 + \xi_i \quad && \text{for } y_i = -1, \\ \xi_i &\geq 0 \quad && k = 1, 2, \ldots, n. \end{aligned}$$

An error occurs if $\xi_i > 1$. Thus,

$$\sum_{i=1}^{n} I(\xi_i > 1) = \#\text{ errors}$$



Proposed solution: Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} I(\xi_i > 1) \qquad \text{(non – convex!)}$$

Suggestion: Replace $I(\xi_i > 1)$ by ξ_i (upper bound)

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & L_P(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Tradeoff: Large C - penalize errors, Small C penalize complexity

Dual Problem: Same as in separable case, except that $0 \leq \alpha_i \leq C$

Support vectors: $\alpha_i > 0$ - but lose geometric interpretation!

Solution:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$f(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b\right)$$

KKT conditions:

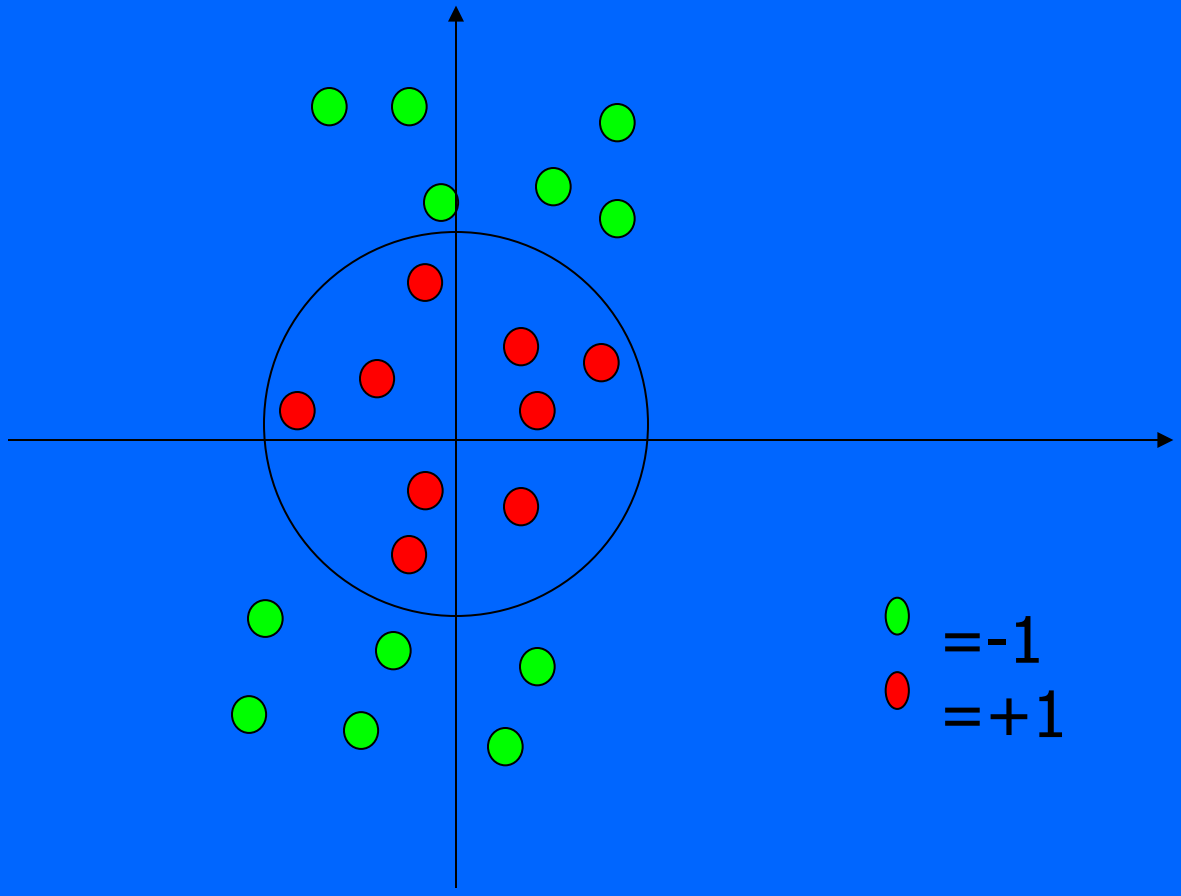
$$0 = \sum_{i=1}^{n} \alpha_i y_i$$

$$0 = \alpha_i (y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) - 1 + \xi_i)$$

$$0 = (C - \alpha_i)\xi_i$$

Support vectors: characterized by $\alpha_i >$

Problems with linear SVM



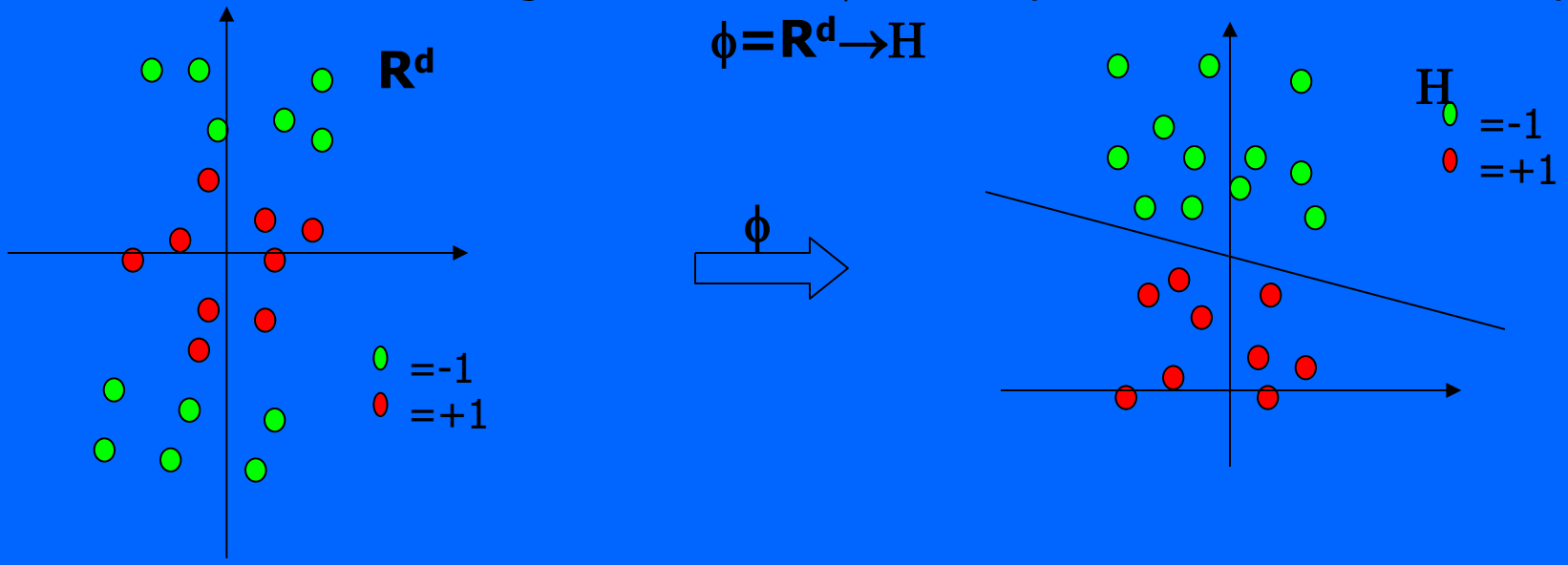
What if the decison function is not a linear?

Non-linear SVM 1

The Kernel trick

Imagine a function ϕ that maps the data into another space:

$\phi = R^d \rightarrow H$



Remember the function we want to optimize: $L_D = \sum\alpha_i - \frac{1}{2}\sum\alpha_i\alpha_j y_i y_j x_i \bullet x_j$, x_i and x_j as a dot product. We will have $\phi(x_i) \bullet \phi(x_j)$ in the non-linear case. **If there is a "kernel function" K such as $K(x_i,x_j) = \phi(x_i) \bullet \phi(x_j)$,** we do not need to know ϕ explicitly. One example:

$$K(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$$

Linear Separability: More likely in high dimensions

Mapping: Map input into high-dimensional *feature space* $\mathbf{\Phi}$

Classifier: Construct *linear* classifier in $\mathbf{\Phi}$

Motivation: Appropriate choice of $\mathbf{\Phi}$ leads to linear separability.

Non-linearity and **high dimension** are essential (Cover '65)!

$$\mathbf{\Phi} : \mathbb{R}^d \mapsto \mathbb{R}^D \qquad (D \gg d)$$

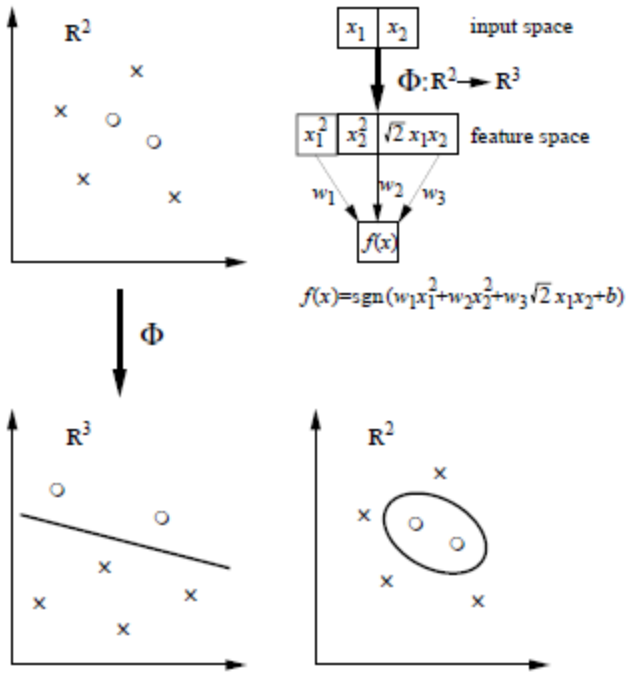
$$\mathbf{x} \mapsto \mathbf{\Phi}(\mathbf{x})$$

Hyper-plane condition: $\mathbf{w}^\top \mathbf{\Phi}(\mathbf{x}) + b = 0$

Inner products: $\mathbf{x}^\top \mathbf{x} \mapsto \mathbf{\Phi}^\top(\mathbf{x})\mathbf{\Phi}(\mathbf{x})$

Decisions in Input and Feature Space:

Problems becomes linearly separable in feature space (Fig. from Schölkopf & Smola 2002)



Recall for linear SVM

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + b \quad ; \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Obtained

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b$$

In feature space

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{\Phi}(\mathbf{x}_i)^\top \mathbf{\Phi}(\mathbf{x}) + b$$

Kernel: A symmetric function $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$

Inner product kernels: In addition

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{\Phi}(\mathbf{x})^\top \mathbf{\Phi}(\mathbf{z})$$

Motivation: $\mathbf{\Phi} \in \mathbb{R}^D$, where D may be very large - inner products expensive

Classifier:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{\Phi}(\mathbf{x}_i)^\top \mathbf{\Phi}(\mathbf{x}) + b\right)$$
$$= \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

The gain: Implement **infinite-dimensional** mapping, but do all calculations in **finite dimension**

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad ; \quad 0 \le \alpha_i \le C$$

Observe: Only difference from linear case is in the kernel

Optimization task is unchanged!

Homework

- XOR example (Section 5.2.1)
- Problem 5.3, p 131