Support Vector Machines

Main Ideas

- Supervised, binary, linear, discriminative
  - Used for classification or regression
- Optimal hyperplane
  - Outputs an optimal hyperplane to separate +ve and –ve training examples

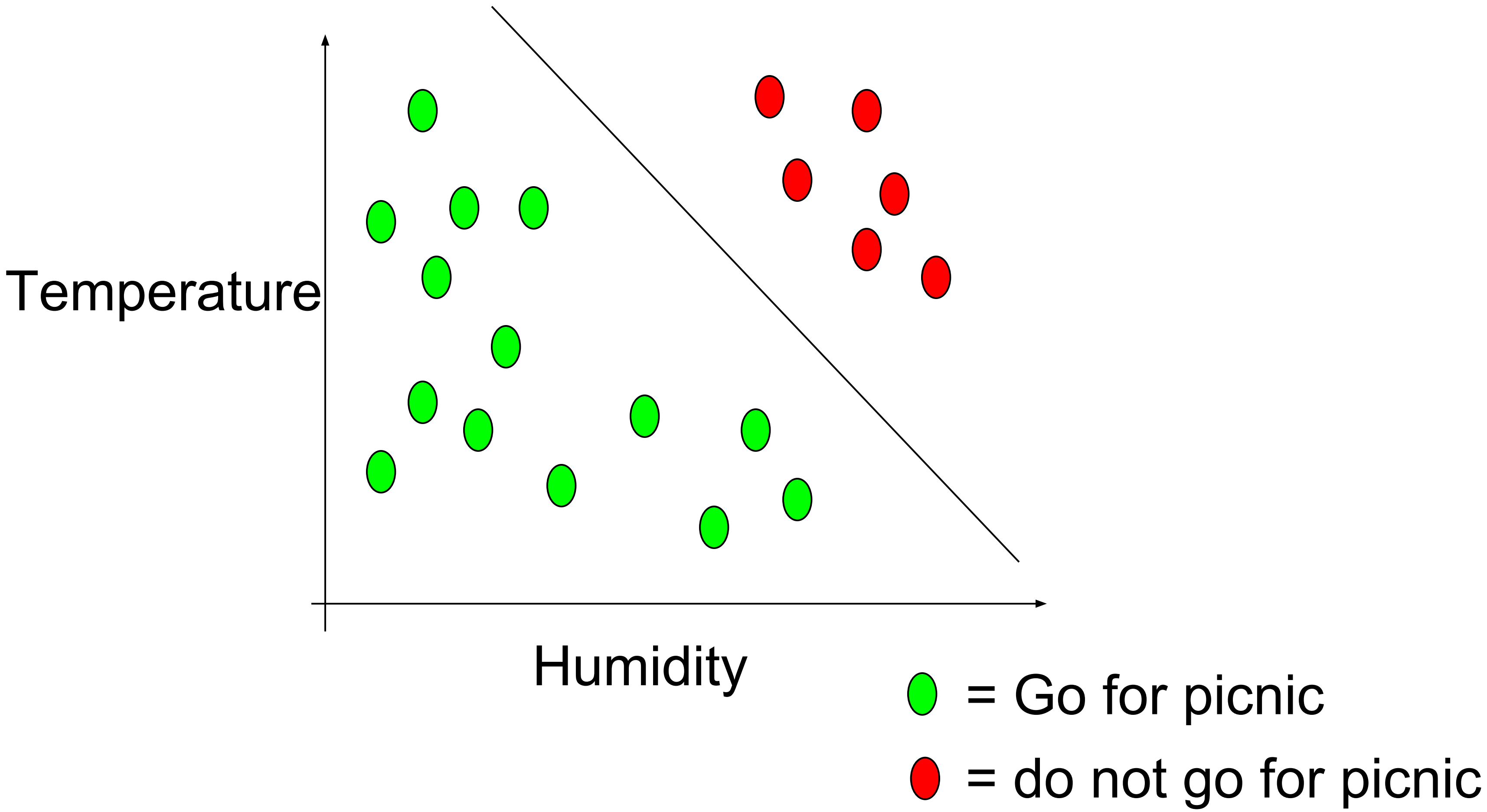
- Maximize Margin of separation Classifier
  - Formalizes notion of the best linear separator

- Kernels
  - Projecting non-linearly separable data into higher-dimensional space makes it linearly separable

Main Ideas

- Lagrangian Multipliers
  - Way to convert a constrained optimization problem to one that is easier to solve
- Complexity
  - Depends only on the number of training examples, not on dimensionality of the kernel space!
- Regularization
  - Has a regularization parameter to relax optimization constraints for non separable data

Picnic example

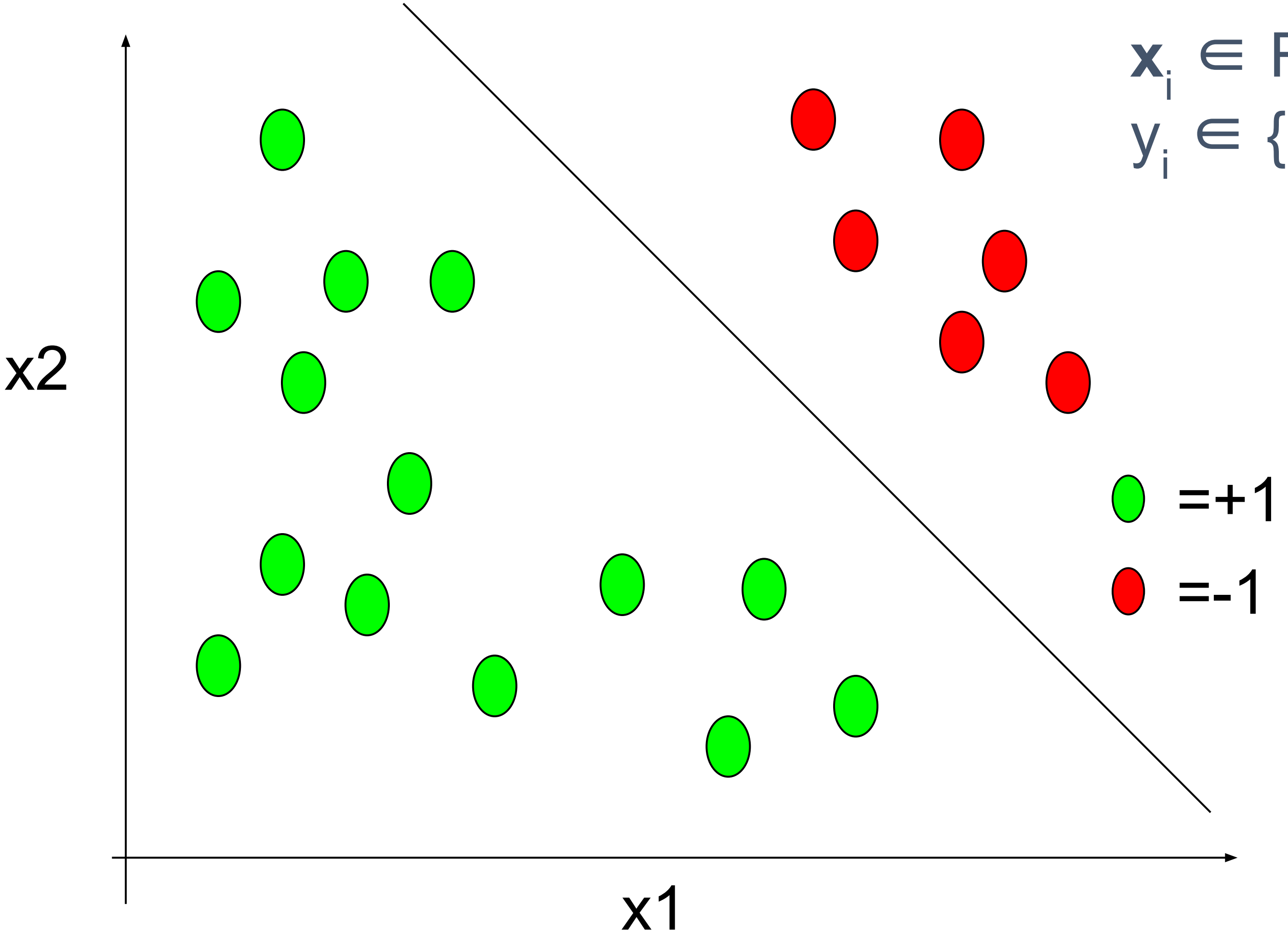


Linear Support Vector Machines

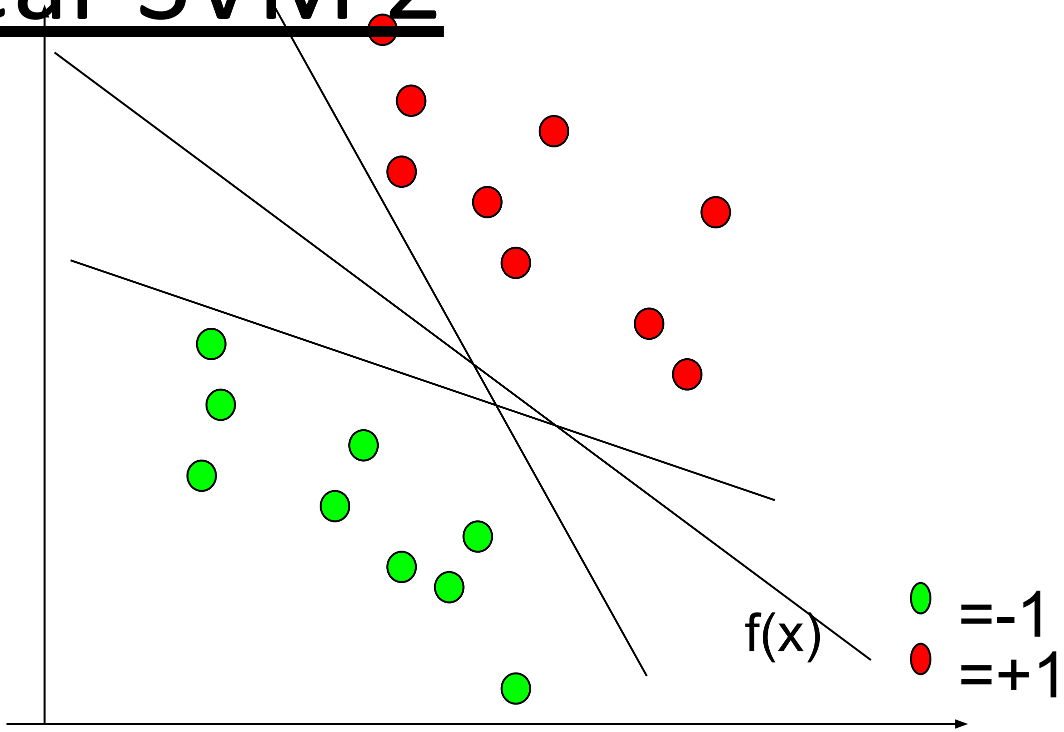
Data: $<\mathbf{x}_i, y_i>$, $i=1,..,l$

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

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



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 parameterize by a vector (**w**) and a constant b.
Can be expressed as $\mathbf{w}^T \bullet \mathbf{x}+b=0$

$$\mathbf{W}=\begin{Bmatrix} \underline{w1} \\ \underline{w2} \\ \dots \end{Bmatrix} \quad \boldsymbol{x}=\begin{Bmatrix} \underline{x1} \\ \underline{x2} \\ \dots \end{Bmatrix}$$

Our aim is to find such a hyperplane <u>f(x)=sign(**w**•**x**+b),</u> that correctly classify our data.

Definitions

Define the hyperplane H such that:

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

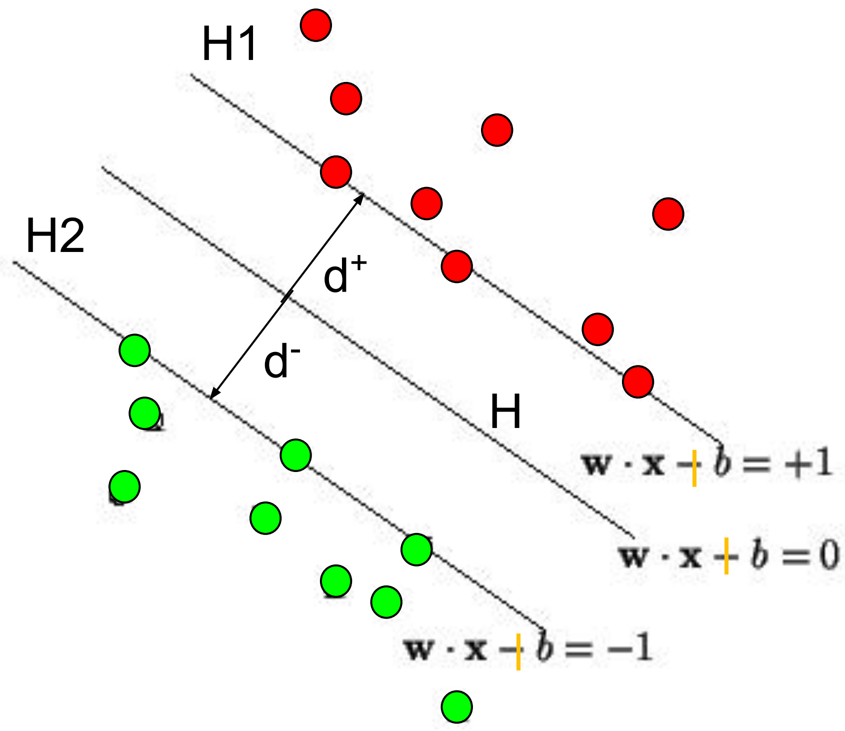
$\mathbf{x}_i \bullet \mathbf{w} + b \leq -1$ when $y_i = -1$

H1 and H2 are the planes:

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

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

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



d+ = the shortest distance to the closest positive point

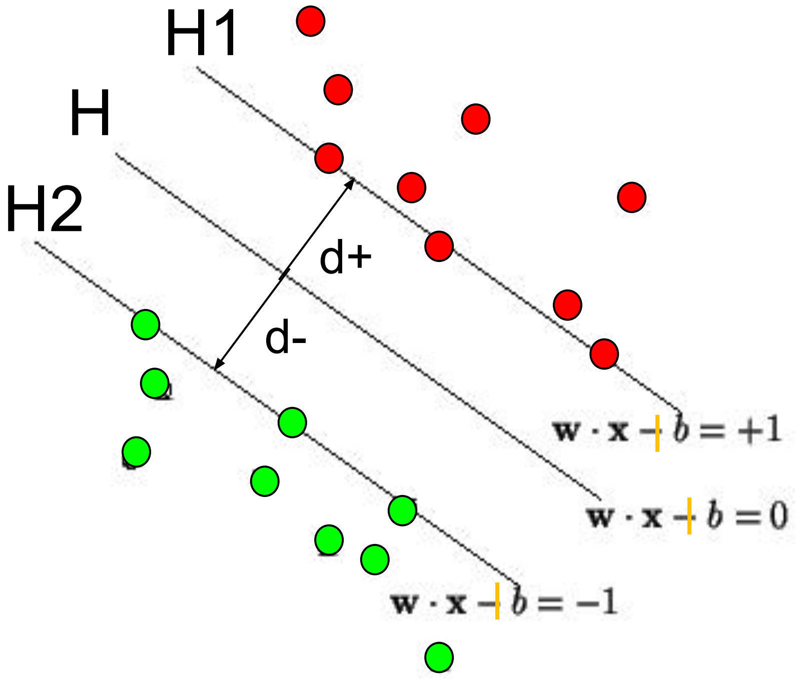
d- = the shortest distance to the closest negative point

The <u>margin</u> 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: $2/||w||$

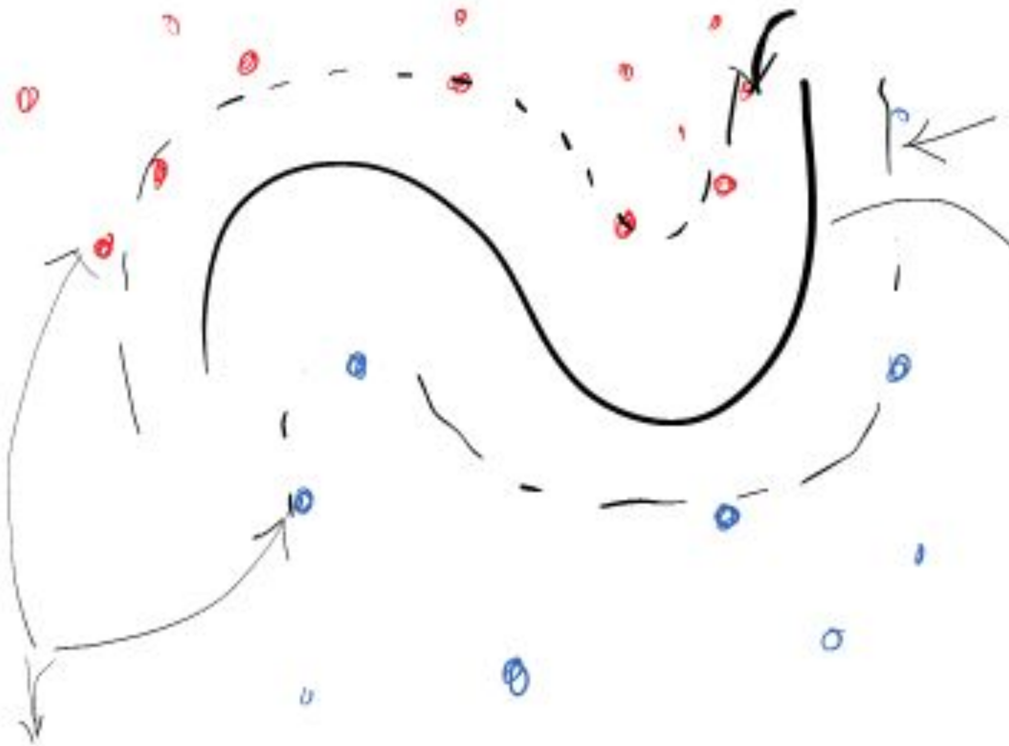
In order to maximize the margin, we need to minimize ||w||. With the condition that there are no datapoints between H1 and H2:

$\left.\begin{array}{l}\mathbf{x}_i\bullet\mathbf{w}+b \geq +1 \text{ when } y_i = +1 \\ \mathbf{x}_i\bullet\mathbf{w}+b \leq -1 \text{ when } y_i = -1\end{array}\right\}$ **Can be combined into $y_i(\mathbf{x}_i\bullet\mathbf{w}) \geq 1$**

SVM – Maximize margin of separation ρ

Euclidean norm

$$\rho = \frac{2}{\|\omega_0\|}$$



Training data $\{(\vec{x}_i, y_i)\}$

$$\boxed{\vec{\omega}_0^{\,T}\vec{x} + b = 0}$$

$\vec{\omega}_0^{\,T}\vec{x}_i + b \geq 1$ if $y_i = +1$

$\vec{\omega}_0^{\,T}\vec{x}_i + b_0 \leq -1$ if $y_i = -1$

Optimization problem

(A) OBJECTIVE FUNCTION

MINIMIZE

$$\boxed{\phi(\vec{\omega}) = \frac{1}{2}\omega^T\omega}$$

$$\rho = \frac{2}{\|w_o\|}$$

(B) CONSTRAINTS FULFILL

$$\boxed{y_i(\vec{\omega}^T\vec{x}_i + b) \geq 1 \quad \text{for } i = 1 \ldots N}$$

COMBINE (A) & (B) using Langrange Multipliers α

(M) $$J(\vec{\omega}, b, \alpha) = \frac{1}{2}w_i^T w_i - \sum_{i=1}^{N} \alpha_i \left[y_i (\omega_i^T \cdot \vec{x}_i + b) - 1 \right]$$

An Optimization problem and its dual

$\partial J / \partial w = 0 \qquad \partial J / \partial b = 0$

yields $\qquad \vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$ —— ①

$\sum_{i=1}^{N} \alpha_i y_i = 0$ —— ②

Quadratic Programming

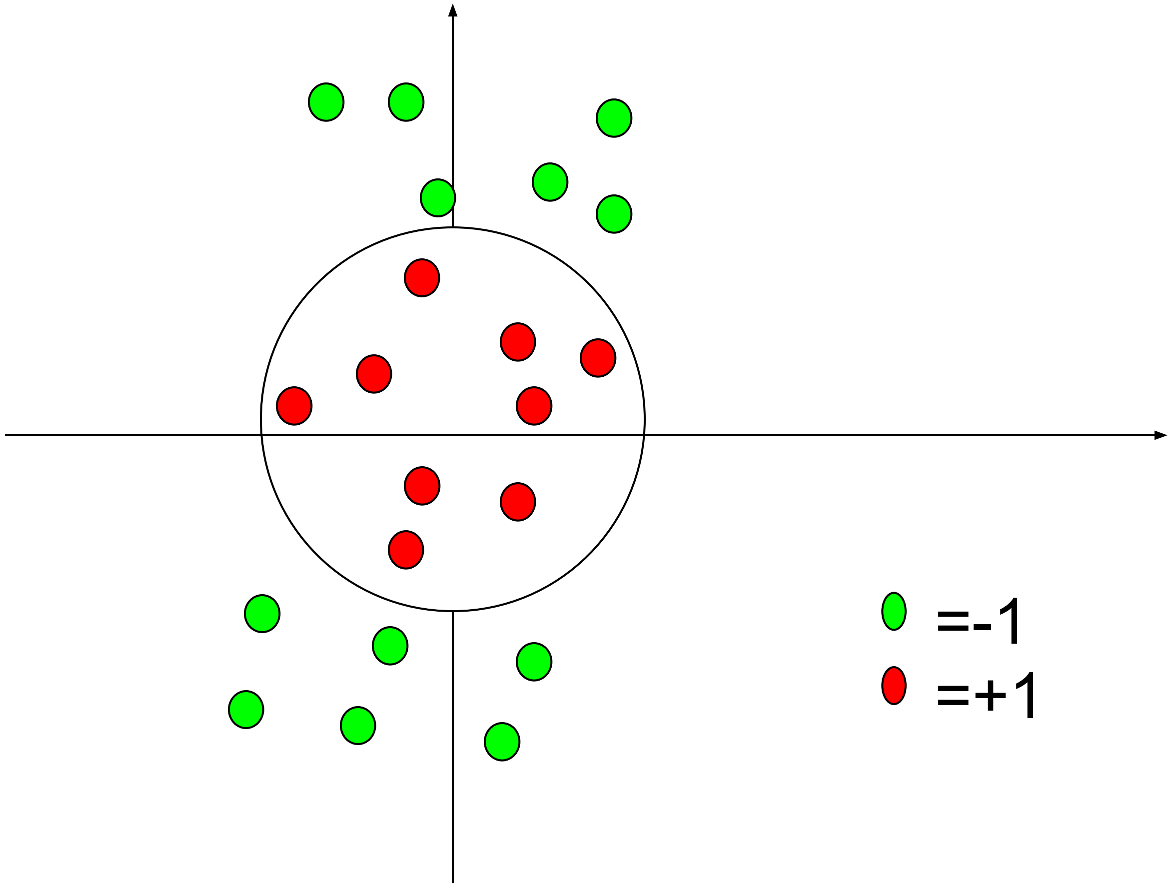
- Why is this reformulation a good thing?
- The problem

$$\text{Maximize} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

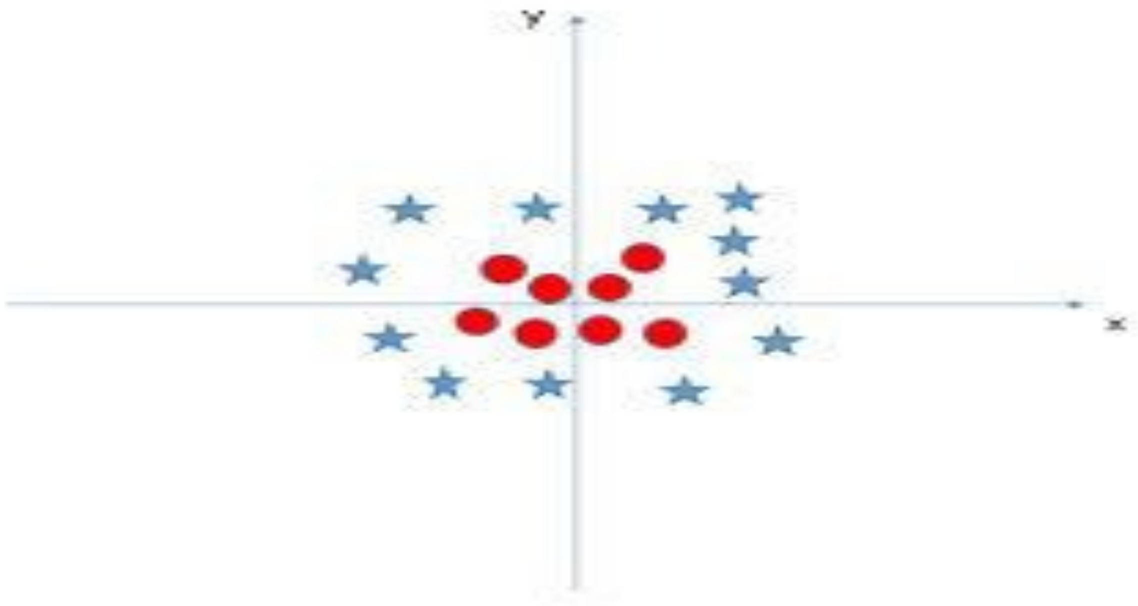
$$\text{subject to} \sum_i y_i \alpha_i = 0 \quad \text{and} \quad \alpha_i \geq 0$$

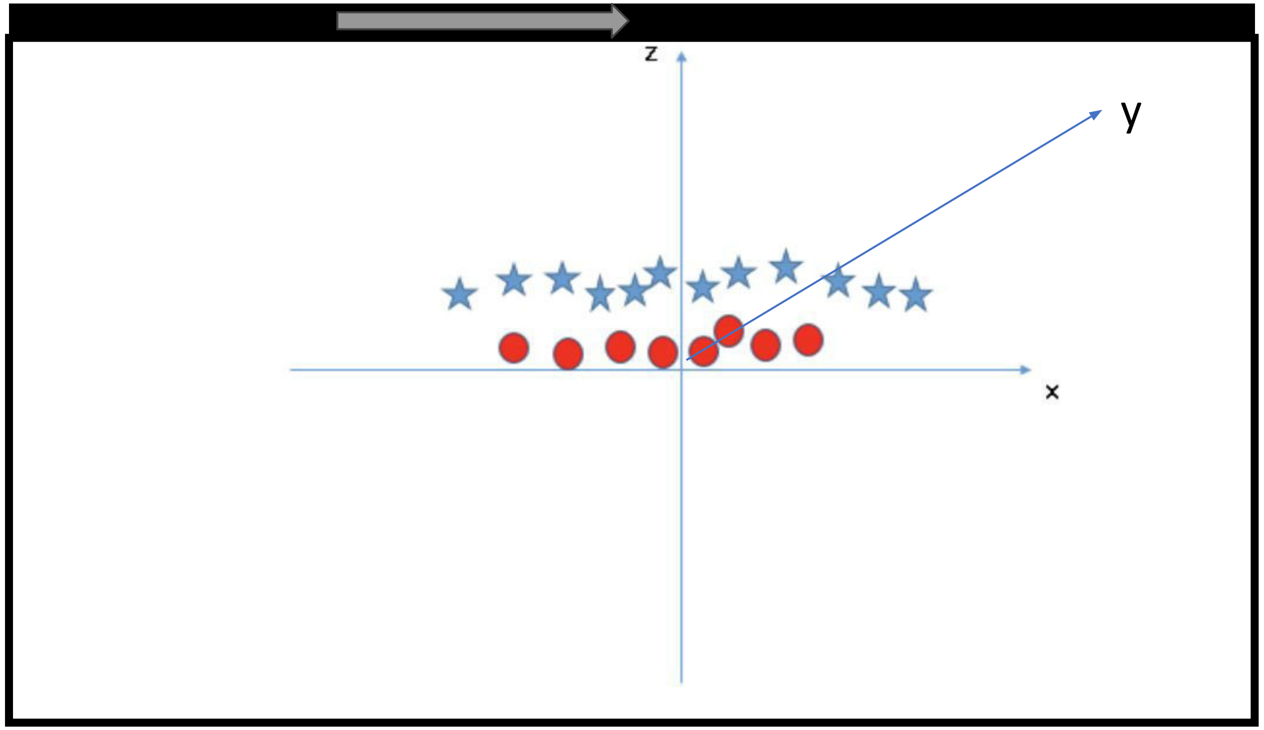
- is an instance of what is called a positive, semi-definite programming problem
- For a fixed real-number accuracy, can be solved in time order = $O(|D|^2 \log |D|^2)$

Problems with linear SVM



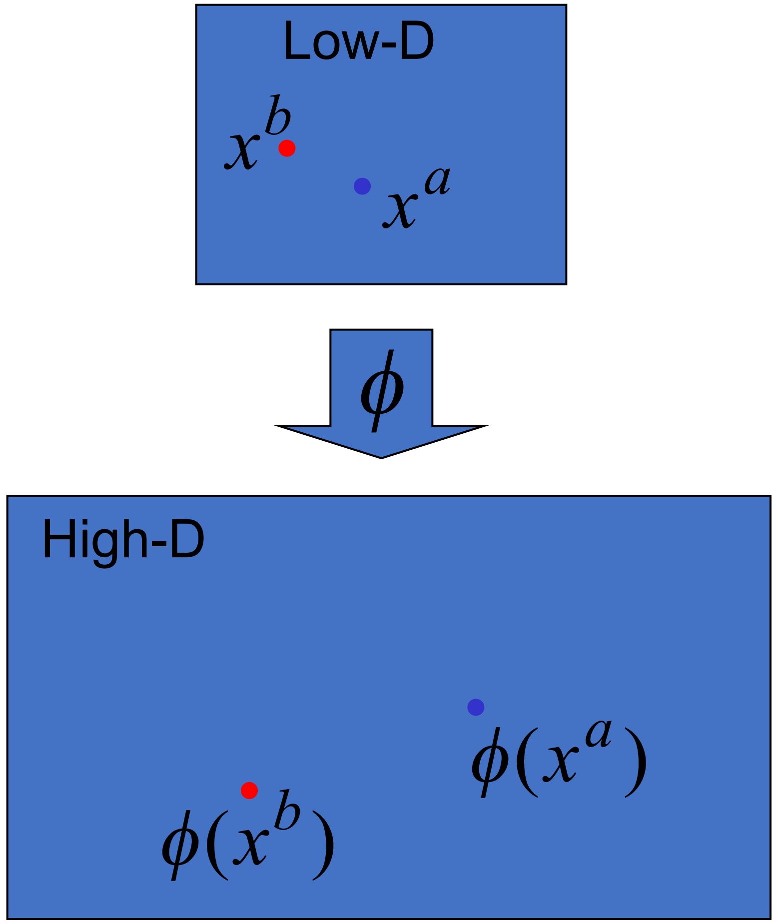
What if the decision function is not a linear?


y
x


z
y
x

The kernel trick

- For many mappings from a low-D space to a high-D space, there is a simple operation on two vectors in the low-D space that can be used to compute the scalar product of their two images in the high-D space.

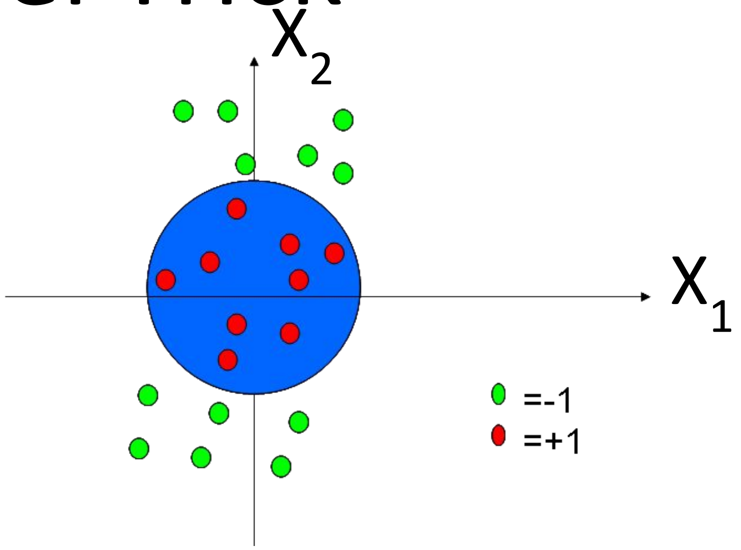


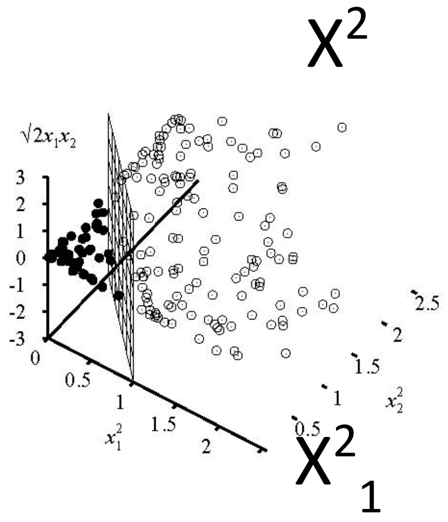
$$K(x^a, x^b) = \phi(x^a) . \phi(x^b)$$

Letting the kernel do the work

doing the scalar product in the obvious way

Kernel Trick





Data points are linearly separable in the space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$

We want to maximize $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)\rangle$

Define $K(\mathbf{x}_i, \mathbf{x}_j) = \langle F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)\rangle$

K is often easy to compute directly!

Here,

$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle^2$

Some commonly used kernels

Polynomial: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y} + 1)^p$

Gaussian radial basis function: $K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2 / 2\sigma^2}$

Neural net: $K(\mathbf{x},\mathbf{y}) = \tanh(k\,\mathbf{x}.\mathbf{y} - \delta)$

Parameters that the user must choose

Mercers condition

Kernel M should be 1) real 2) symmetric

3) +ve semi-definite $z^T M z$ is strictly +ve

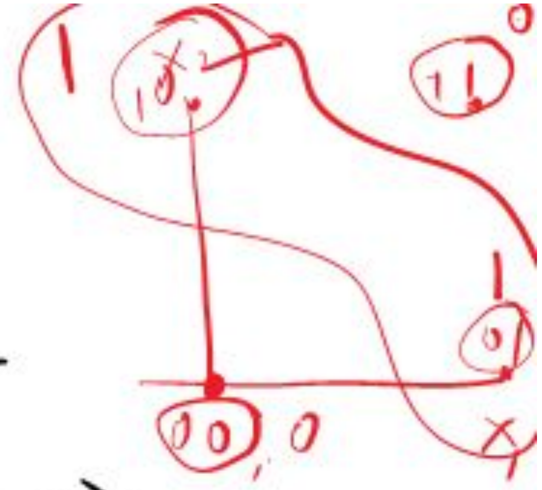
E.g. Identity Matrix

$$\underbrace{\begin{bmatrix} a & b \end{bmatrix}}_{z^T} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{z} = a^2 + b^2$$

$\rightarrow$ +ve

XOR example

	Input vector $\vec{x}$ x_1	x_2	Label y
x^1	-1	-1	-1
x^2	-1 ✓	$+1$ ✓	$+1$
x^3	$+1$	-1	$+1$
x^4	$+1$	$+1$	-1



Let

$$K(\vec{x}, \vec{x}_i) = (1 + \vec{x}^T \vec{x}_i)^2$$

$$\Rightarrow \left(1 + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}\right)^2 \Rightarrow (1 + x_1 x_{i1} + x_2 x_{i2})^2 \Rightarrow$$

$$1 + (x_1 x_{i1} + x_2 x_{i2})^2 + 2(x_1 x_{i1} + x_2 x_{i2})$$

$$\Rightarrow 1 + \boxed{x_1^2}\,\boxed{x_{i1}^2} + 2 x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2 x_1 x_{i1} + 2 x_2 x_{i2}$$

Image of input vector x in high dimensional feature space

$$\varphi(\vec{x}) = [1, \ x_1^2, \sqrt{2}\, x_1 x_2, \ x_2^2, \ \sqrt{2}\, x_1, \ \sqrt{2}\, x_2]^T$$

$\varphi(x_i) \cdot \varphi(x_j)$

$$\varphi(\vec{x}_i) = [1 \ \ x_{i1}^2, \ \sqrt{2} x_{i1} x_{i2}, \ x_{i2}^2, \ \sqrt{2}\, x_{i1}, \ \sqrt{2}\, x_{i2}]$$

$$K[1,1] = 1 + (-1)^2(-1)^2 + 2(-1)(-1)(-1)(-1) + (-1)^2(-1)^2 + 2(-1)(-1) + 2(-1)(-1)$$

$x_1 \ x_1$

$$= 9$$

Likewise calculate $K[1,2]$, $K[1,3]$ $K[1,4]$ $K[2,2]$, $K[2,3]$

$K[2,4]$, $K[3,3]$, $K[3,4]$, $K[4,4]$

$x_1 \ (-1 \ \ -1) \ \ -1$

$$K[1,2] = 1 + (-1)^2(-1)^2 + 2(-1)(-1)(-1)(+1) + (-1)^2(1)^2 + 2(-1)(-1) + 2(-1)1$$

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix} \rightarrow \text{KERNEL}$$

$$Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \, K(x_i, x_j)$$

Now $Q(\alpha) = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2}\Big(\underbrace{9\alpha_1^2}_{\alpha_1\alpha_1 y_1 y_1 K(x_1, x_1)} - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3$

$\alpha_1\alpha_1(-1)(-1) \quad 9$

$+ 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2$

Note Q can be found only in terms of αs optimize $Q(\alpha)$ w.r.t. $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4$:

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \quad \text{--- } (1)$$

$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1 \quad \text{--- } (2)$$

$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1 \quad \text{--- } (3)$$

$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \quad \text{--- } (4)$$

Solving $\alpha_{0,1} = \alpha_{0,2} = \alpha_{0,3} = \alpha_{0,4} = 1/8$

$$Q_0(\alpha) = \frac{1}{4}$$

We need to calculate equation of plane-→ ω b Eq.

$Q(\alpha) = \frac{1}{2}\|w_0\|^2 = \frac{1}{4}$: because constraints vanish.

$\Rightarrow \quad \|w_0\| = \frac{1}{\sqrt{2}}$ —————— (1)

Now

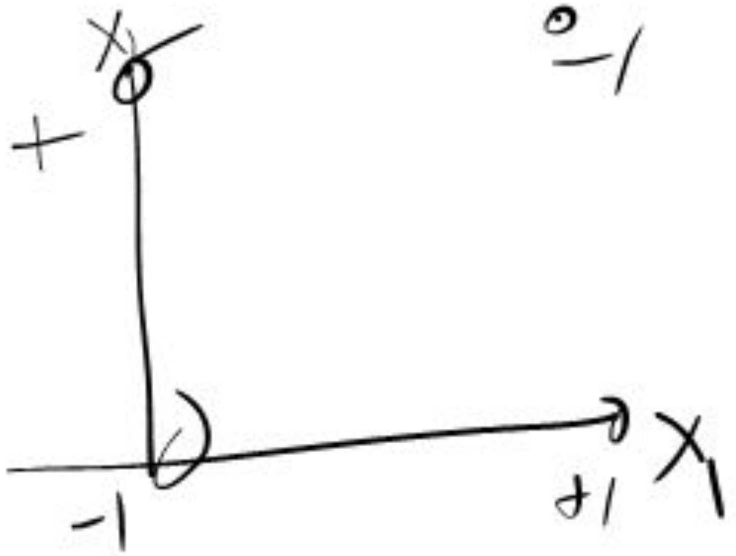
$$\vec{w_0} = \sum \alpha_{0i}\, y_i\, \varphi(x_i)$$

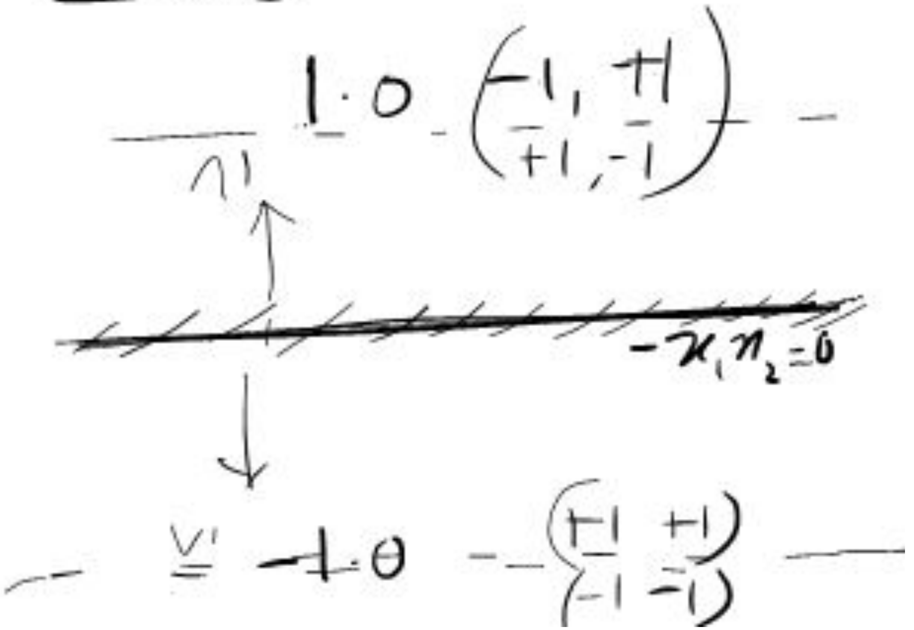
$$\frac{1}{8}\left[-1\begin{bmatrix}1\\1\\\sqrt{2}\\1\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} +1\begin{bmatrix}1\\1\\-\sqrt{2}\\1\\-\sqrt{2}\\2\end{bmatrix} +1\begin{bmatrix}1\\1\\-\sqrt{2}\\1\\\sqrt{2}\\-\sqrt{2}\end{bmatrix} -1\begin{bmatrix}1\\1\\\sqrt{2}\\1\\\sqrt{2}\\\sqrt{2}\end{bmatrix}\right] = \underbrace{\begin{bmatrix}0\\0\\-1/\sqrt{2}\\0\\0\\0\end{bmatrix}}_{\omega_0}$$

The Optimal hyperplane is $w_o^T.\varphi(x)=0$

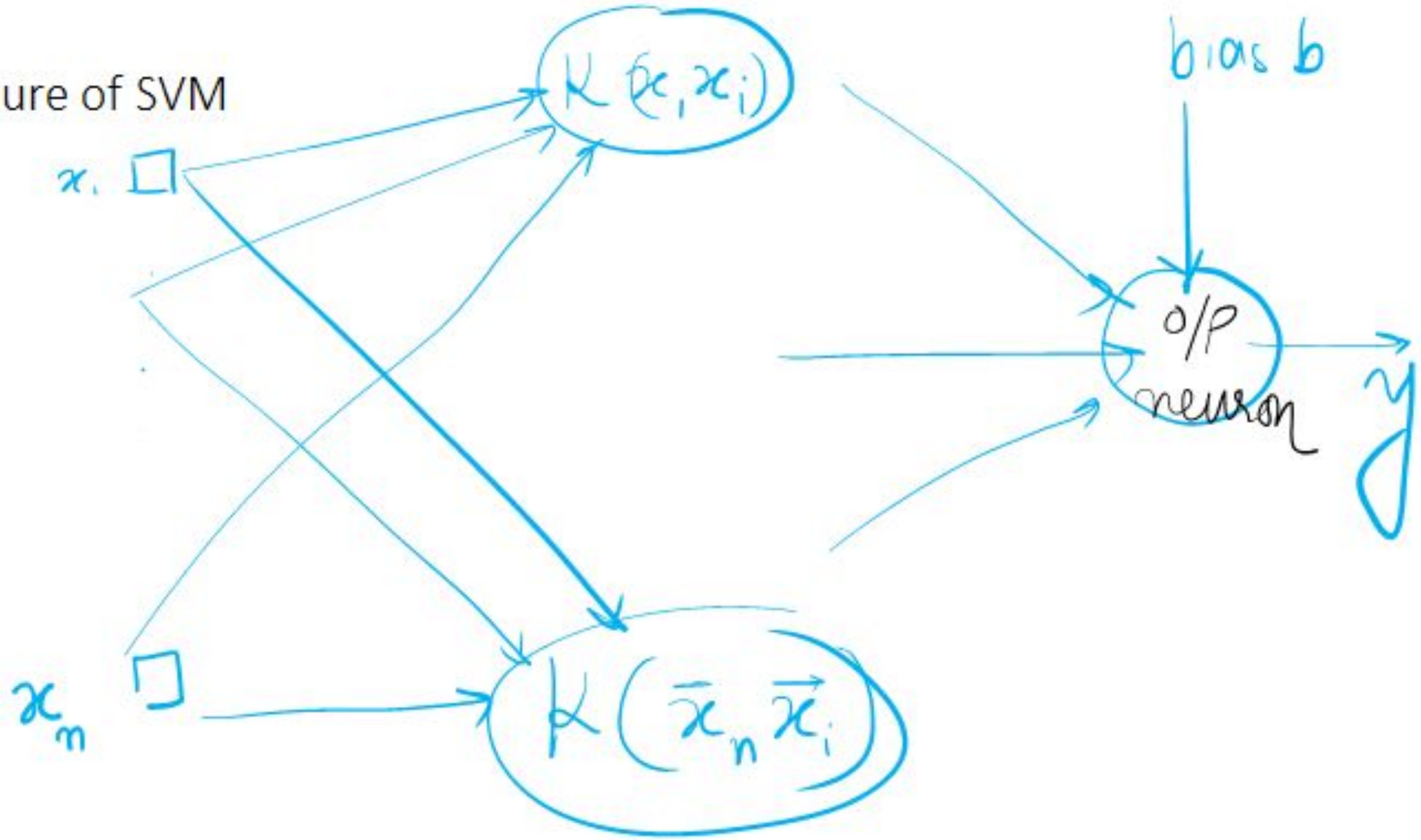
$\omega_0^T \varphi(x) = 0$ $\quad y = mx + b$

$$\begin{bmatrix} 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}\,x_1 x_2 \\ x_2^2 \\ \sqrt{2}\,x_1 \\ \sqrt{2}\,x_2 \end{bmatrix} = 0 \Leftrightarrow -x_1 x_2 = 0$$





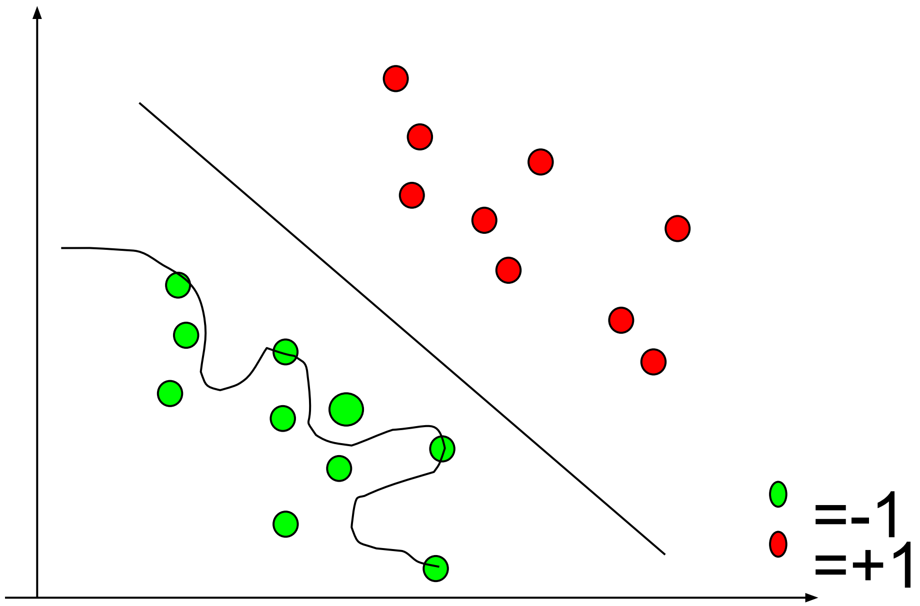
Architecture of SVM



Overtraining/overfitting

A well known problem with machine learning methods is overtraining.
This means that we have learned the training data very well, but
we can not classify unseen examples correctly.

An example: A environmentalist knows rivers very well.
Everytime he sees a new river, he says it is not a river!



Overtraining/overfitting 2

A measure of the risk of overtraining with SVM (there are also other measures).

It can be shown that: The portion, n, of unseen data that will be missclassified is bounded by:

$n \leq$ Number of support vectors / number of training examples

Ockham´s razor principle: Simpler system are better than more complex ones.

In SVM case: fewer support vectors mean a simpler representation of the hyperplane.

Example: Understanding a certain cancer if it can be described by one gene is easier than if we have to describe it with 5000.

Regularization

- Also the ‘ C ‘ parameter in Python’s SkLearn Library
- Optimises SVM classifier to avoid misclassifying the data.

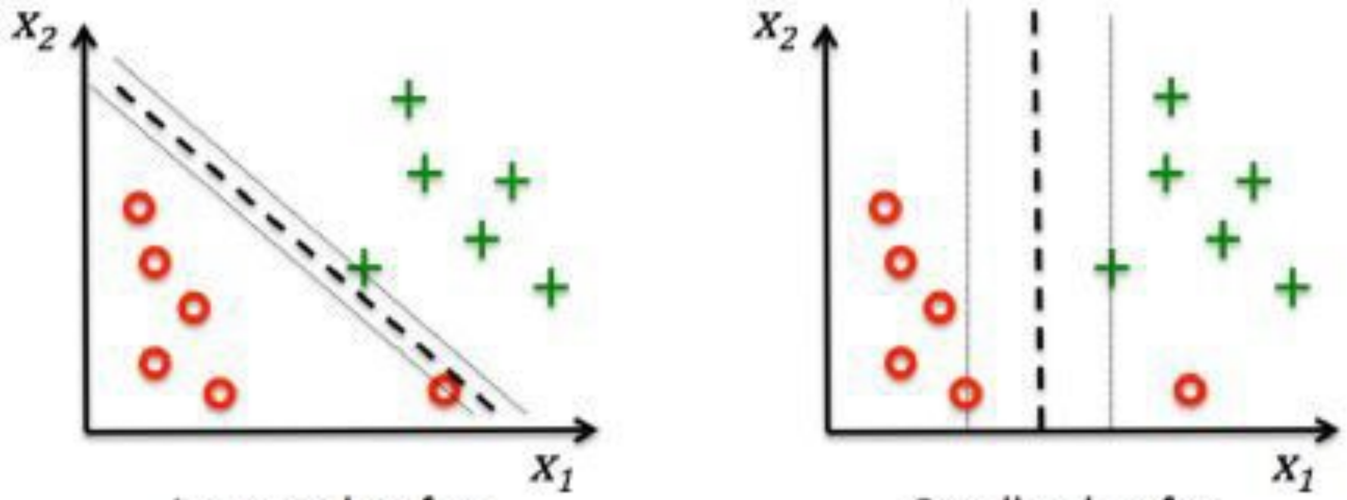
- C → large
- C → small

Margin of hyperplane → small

Margin of hyperplane → large

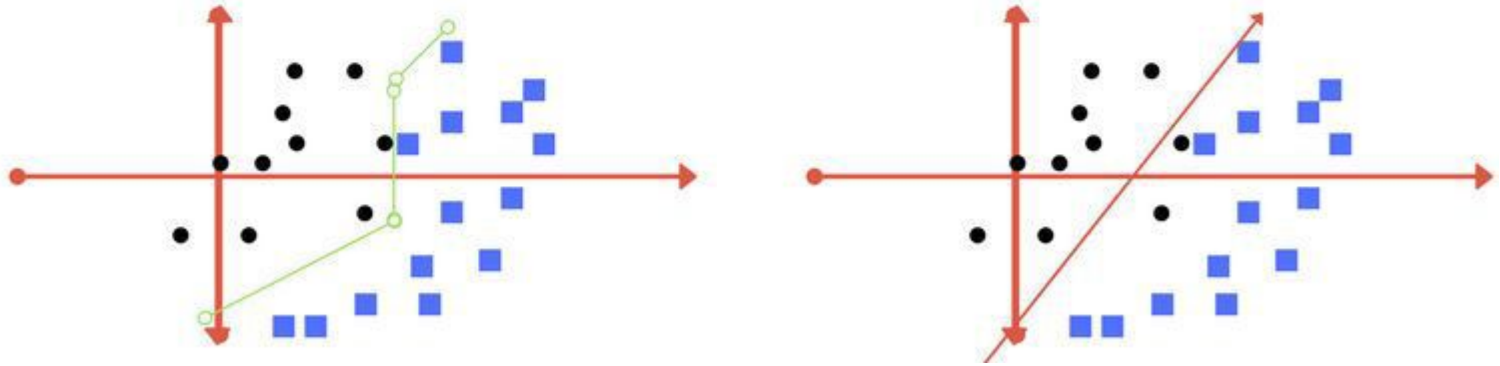
- misclassification(possible)

1. C ---> large , chance of overfit
2. C ---> small , chance of underfitting



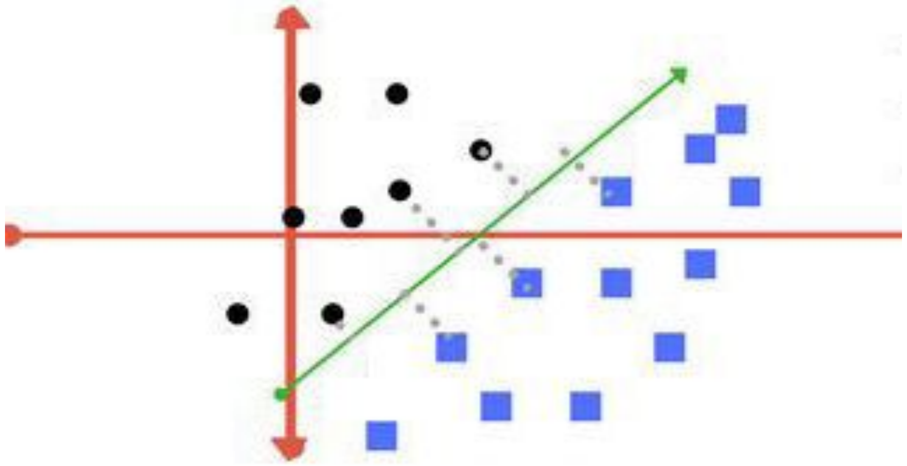
Large value for parameter *C*

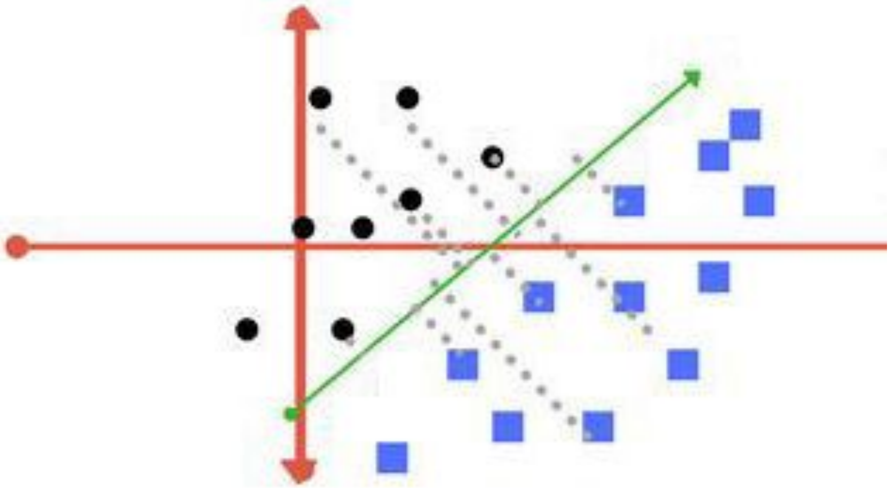
Small value for parameter *C*



Gamma

- Defines how far influences the calculation of plausible line of separation.
- Low gamma -----> points far from plausible line are considered for calculation
- High gamma -----> points close to plausible line are considered for calculation

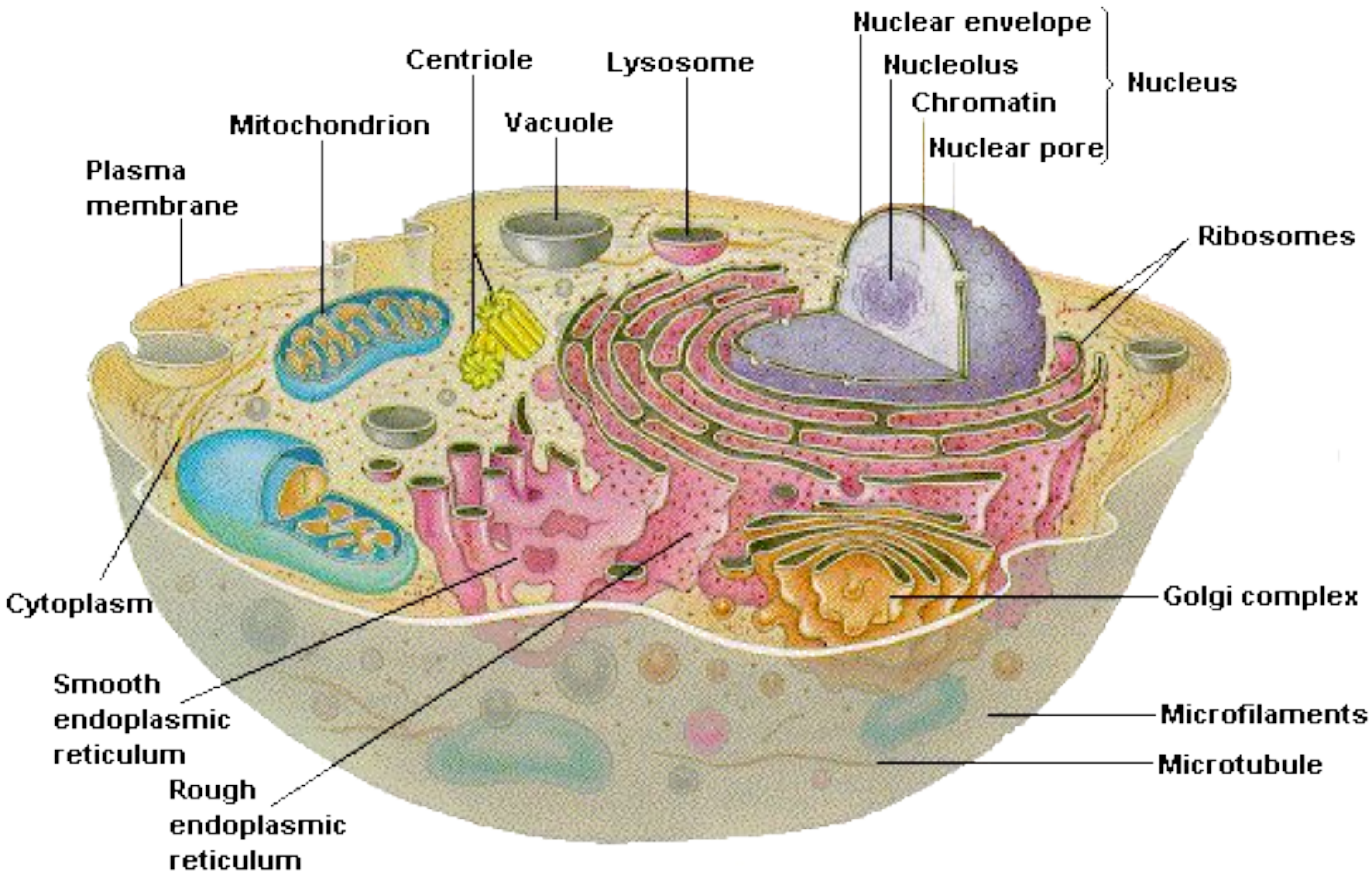




A practical example, protein localization

- Proteins are synthesized in the cytosol.
- Transported into different subcellular locations where they carry out their functions.
- Aim: To predict in what location a certain protein will end up!!!

Subcellular Locations



Method

- Hypothesis: The amino acid composition of proteins from different compartments should differ.
- Extract proteins with know subcellular location from SWISSPROT.
- Calculate the amino acid composition of the proteins.
- Try to differentiate between: cytosol, extracellular, mitochondria and nuclear by using SVM

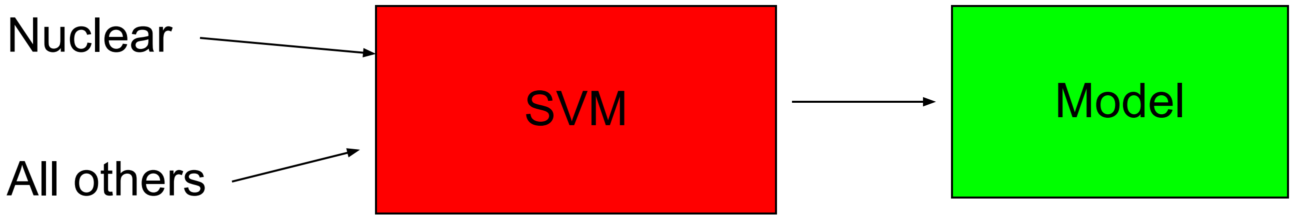
Input encoding

Prediction of nuclear proteins:

Label the known nuclear proteins as +1 and all others as –1.

The input vector xi represents the amino acid composition.

Eg xi =(4.2,6.7,12,….,0.5)
A , C , D,….., Y)



Caution on overfitting

Image classification of tanks. Autofire when an enemy tank is spotted.
Input data: Photos of own and enemy tanks.
Worked really good with the training set used.
In reality it failed completely.

Reason: All enemy tank photos taken in the morning. All own tanks in dawn.
The classifier could recognize dusk from dawn!!!!