Machine Learning

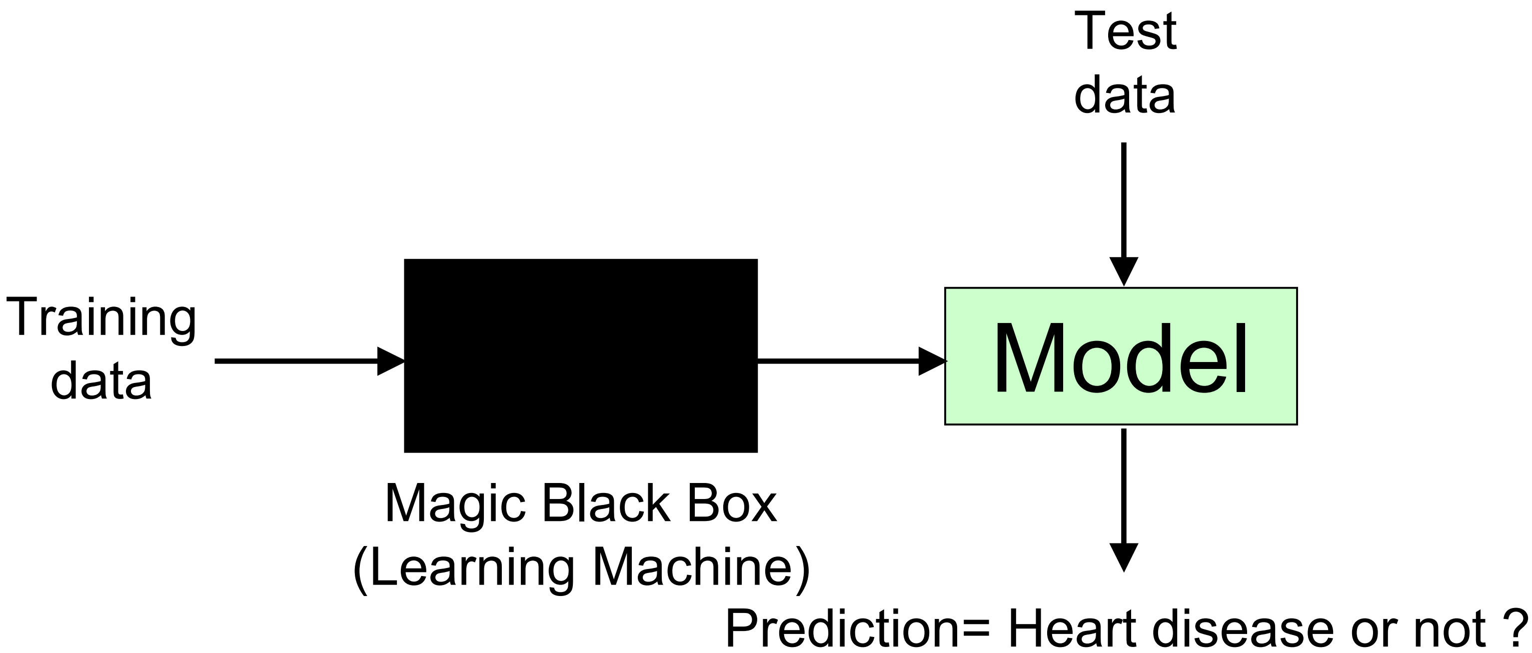
"… a computer program that can learn from experience with respect to some class of tasks and performance measure ..." (Mitchell, 1997)

"… the capacity of a computer to learn from experience, i.e. to modify its processing on the basis of newly acquired information ..." (Oxford Dictionary, 1989)

Examples are:
- Support Vector Machines
- Artificial Neural Networks
- Fuzzy Logic
- Evolutionary Computation

Slides collected from "National Workshop on Support Vector Machine" organized by Machine Intelligence Unit, Indian Statistical Institute, Kolkata in the year 2006.

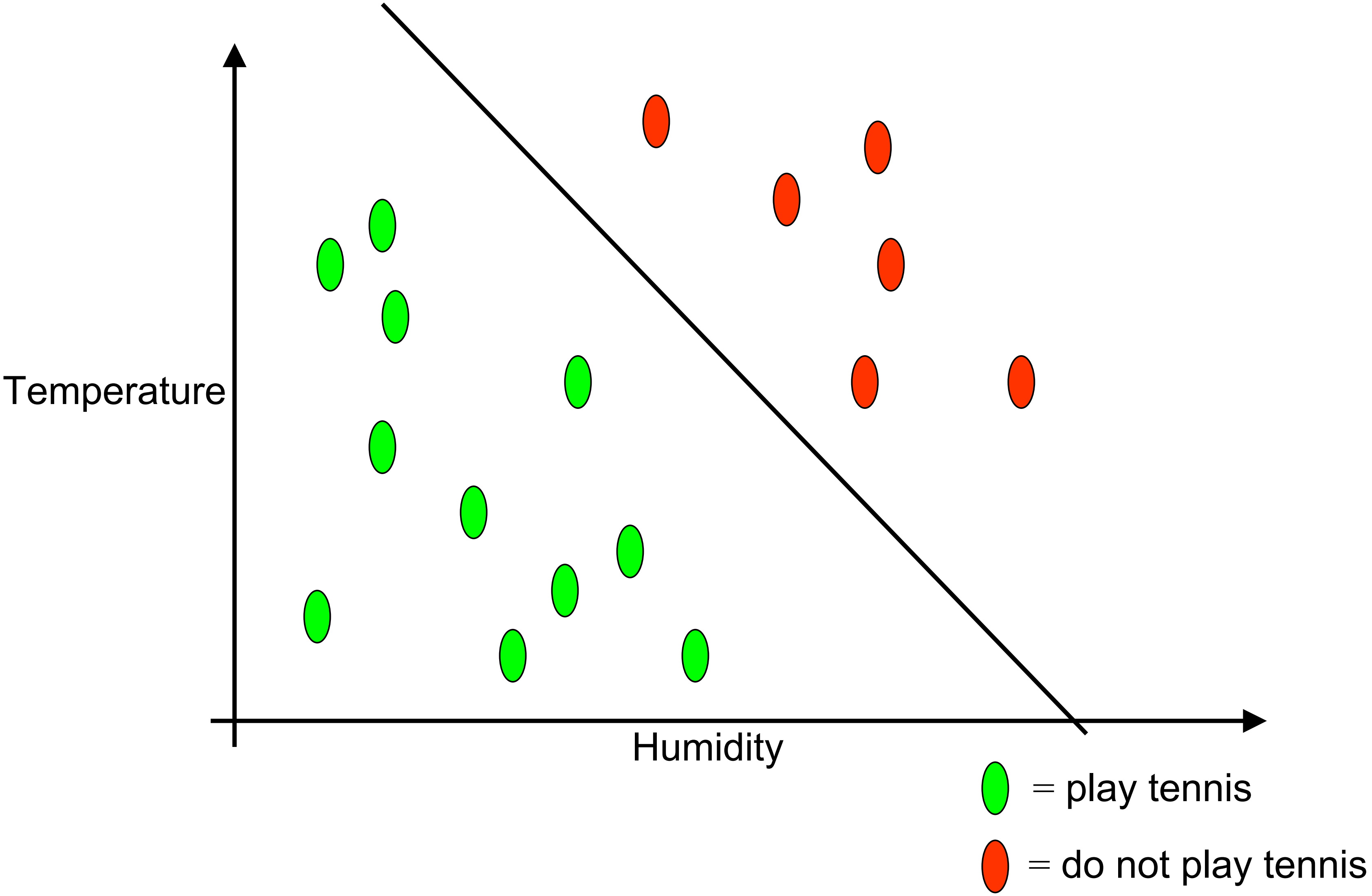
Black Box View of Machine Learning



Training data: Expression pattern of some disease and healthy person

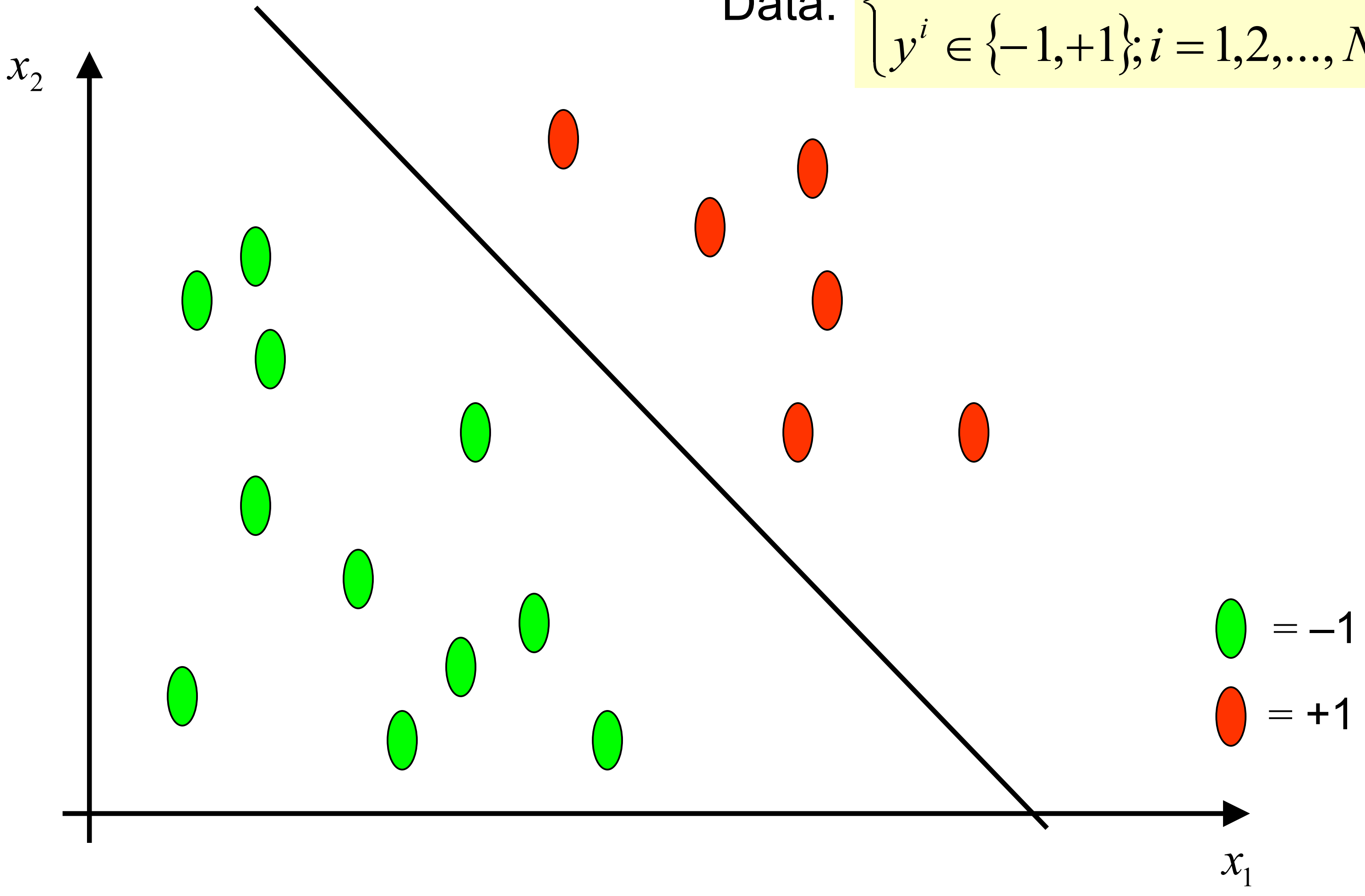
Model: Can distinguish between healthy and sick person. Can be used for prediction

Tennis Example

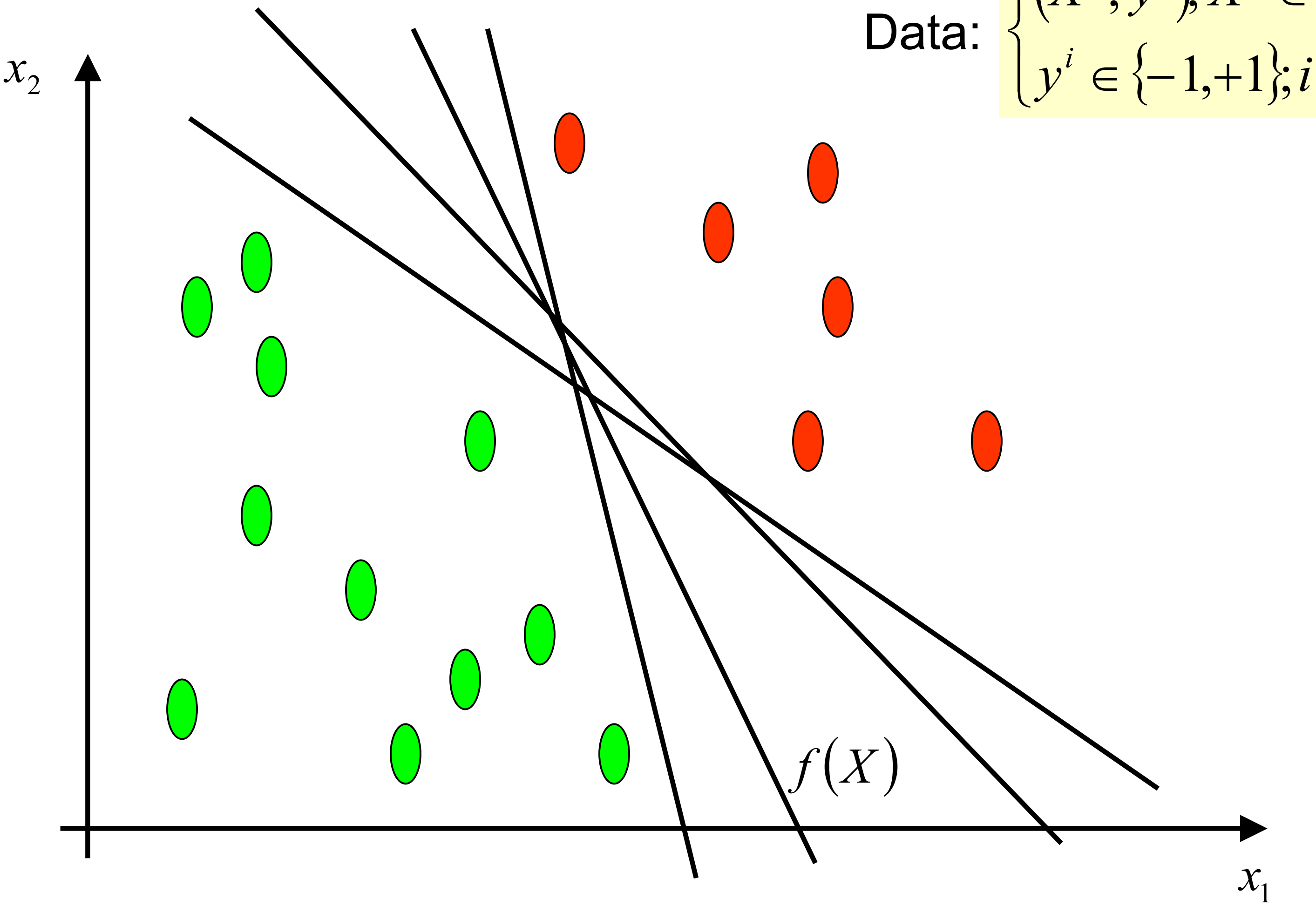


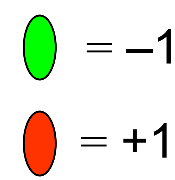
Linear Support Vector Machine

Data: $\left\{ \begin{array}{l} \left(X^i, y^i\right); X^i \in \Re^n; \\ y^i \in \{-1,+1\}; i = 1,2,\ldots,N \end{array} \right\}$



Data: $\left\{ \begin{array}{l} \left(X^i, y^i\right); X^i \in \Re^n; \\ y^i \in \{-1,+1\}; i = 1,2,\ldots,N \end{array} \right\}$





All hyperplanes in $\Re^n$ are parameterized by w and a constant b

Objective: To find a hyperplane $f(X) = sign\left(\langle w.X \rangle + b\right)$ that correctly classify the data

The optimal hyperplane H is such that

$\left\langle w.X^i \right\rangle + b \geq +1$ when $y^i = +1$

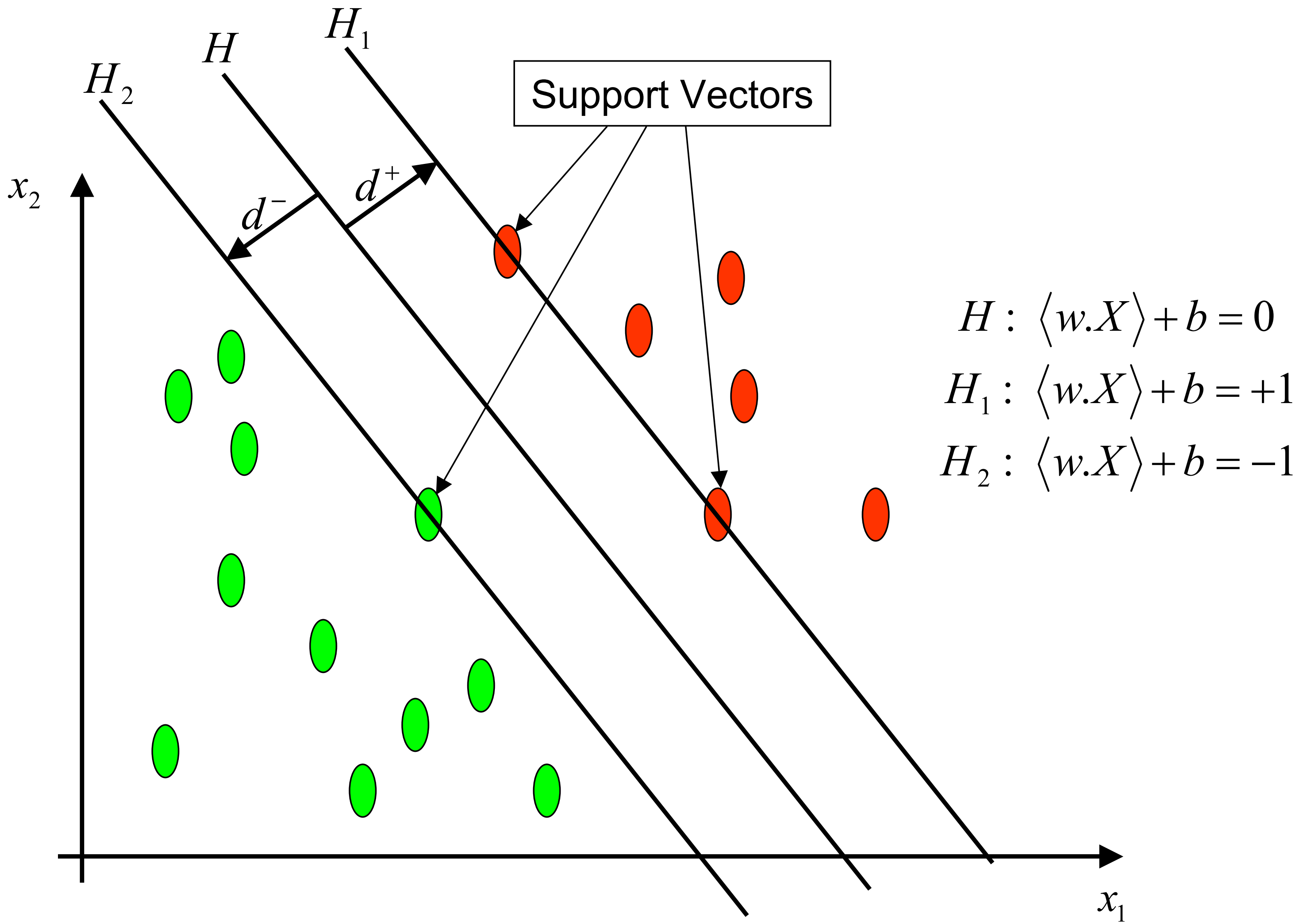
$\left\langle w.X^i \right\rangle + b \leq -1$ when $y^i = -1$

$$H: \left\langle w.X \right\rangle + b = 0$$

Define

$$H_1: \left\langle w.X \right\rangle + b = +1$$

$$H_2: \left\langle w.X \right\rangle + b = -1$$



$d^+ / d^- =$ shortest distance to the closest positive / negative point

Margin of separation = $d^+ + d^-$

Desired: A Classifier with as big margin as possible

Distance between H and H_1 = $\frac{1}{\|w\|_2}$ (geometric margin)

Distance between H_2 and H_1 = $\frac{2}{\|w\|_2}$ (total margin)

In order to maximize the margin we need to minimize $\|w\|$

Under this condition there are no data points between H_1 and H_2

$$\left.\begin{array}{l} \langle w.X^i \rangle + b \geq +1 \quad \text{when} \quad y^i = +1 \\ \langle w.X^i \rangle + b \leq -1 \quad \text{when} \quad y^i = -1 \end{array}\right\} \text{Can be combined as} \quad \boxed{y^i\left(\langle w.X^i \rangle + b\right) \geq 1}$$

The maximum Margin Classifier can be obtained by solving the following optimization problem:

$$\underset{w,b}{\text{Minimize}} \quad \langle w.w \rangle$$
$$y^i\left(\langle w.X^i \rangle + b\right) \geq 1, \ i = 1,2,\ldots,N$$

Formulating the Lagrangian, the primal Lagrangian is obtained as,

$$L(w,b,\alpha) = \frac{1}{2}\langle w.w \rangle - \sum_{i=1}^{N} \alpha_i \left[y^i\left(\langle w.X^i \rangle + b\right) - 1 \right]$$

where $\alpha_i \geq 0$ are Lagrangian multipliers

The corresponding dual is found by differentiating w.r.t w and b and imposing stationarity,

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{N} y^i \alpha_i X^i = 0$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{N} y^i \alpha_i = 0$$

Simplifying the relations,

$$w = \sum_{i=1}^{N} y^i \alpha_i X^i$$

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

Resubstituting back in the primal Lagrangian we get,

$$
\begin{aligned}
L(w,b,\alpha) &= \frac{1}{2}\langle w.w\rangle - \sum_{i=1}^{N}\alpha_i\left[y^i\left(\langle w.X^i\rangle + b\right) - 1\right] \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha_i \alpha_j \langle X^i.X^j\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha_i \alpha_j \langle X^i.X^j\rangle + \sum_{i=1}^{N}\alpha_i \\
&= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha_i \alpha_j \langle X^i.X^j\rangle
\end{aligned}
$$

Hence the dual problem is

$$
\begin{aligned}
&\operatorname*{Max}_{\alpha} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y^i y^j \alpha_i \alpha_j \langle X^i.X^j\rangle \\
&\text{subject to} \quad \sum_{i=1}^{N} y^i \alpha_i = 0, \\
&\qquad\qquad\quad \alpha_i \geq 0, i = 1,2,\ldots,N
\end{aligned}
$$

The dual problem can be easily solved by readily available quadratic programming solvers to give,

$$\alpha^*$$

The weight vector that realises the maximal margin hyperplane is given by

$$w^* = \sum_{i=1}^{N} y^i \alpha_i^* X^i$$

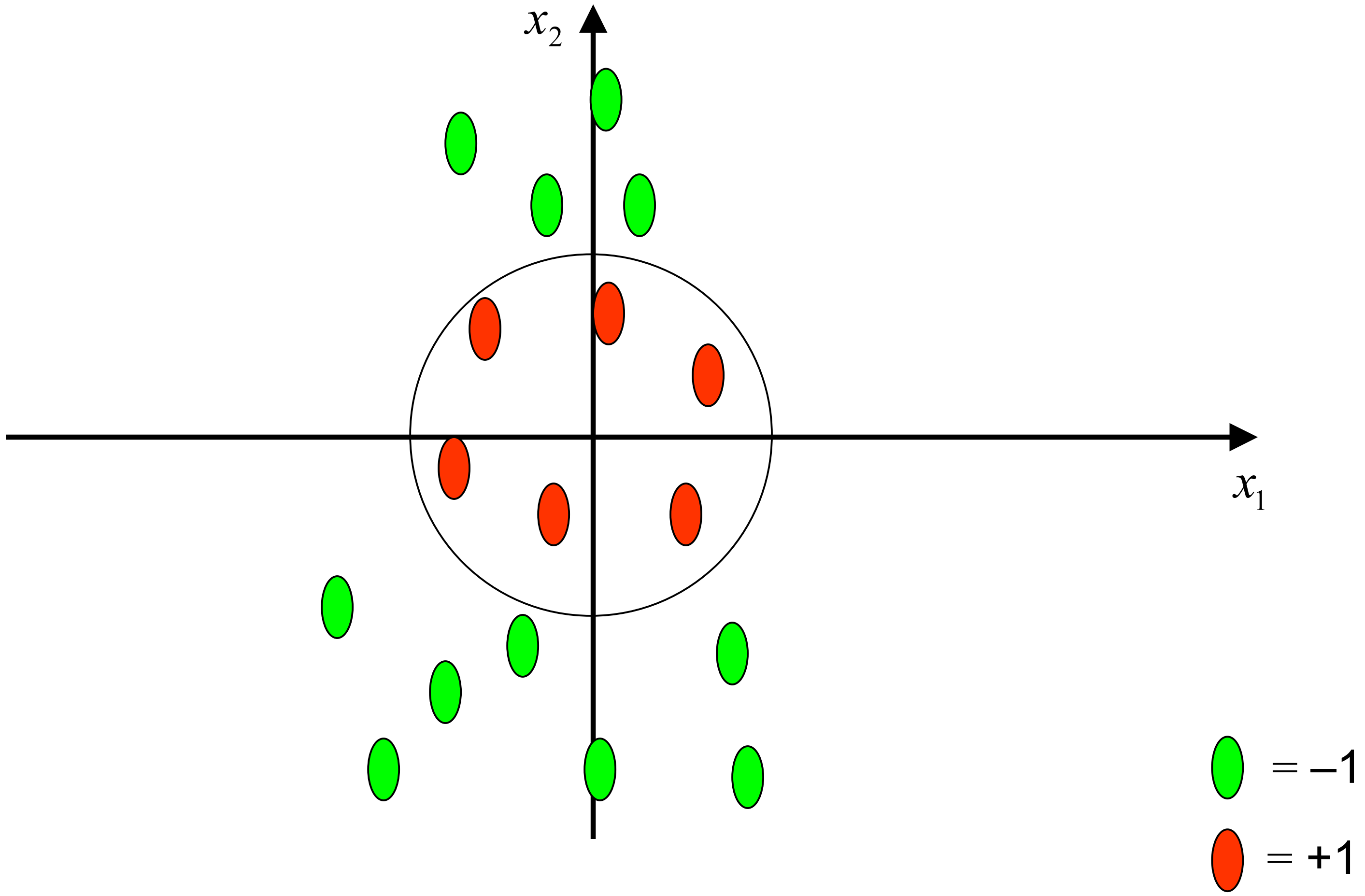
The value of b does not appear in the dual problem and so b^* is found from primal constraints

$$b^* = -\frac{\max_{y^i=-1}\left(\left\langle w^* . X^i \right\rangle\right) + \min_{y^i=1}\left(\left\langle w^* . X^i \right\rangle\right)}{2}$$

In the dual solution it turns out that most of the α_i^* are zero. Non-zero α_i^* occur for the points that are closest to the hyperplane. These are know as the **support vectors** (SV). The optimal hyperplane is given by

$$f(X) = \sum_{i=1}^{N} y^i \alpha_i^* \left\langle X^i . X \right\rangle + b^*$$
$$= \sum_{i \in SV} y^i \alpha_i^* \left\langle X^i . X \right\rangle + b^*$$

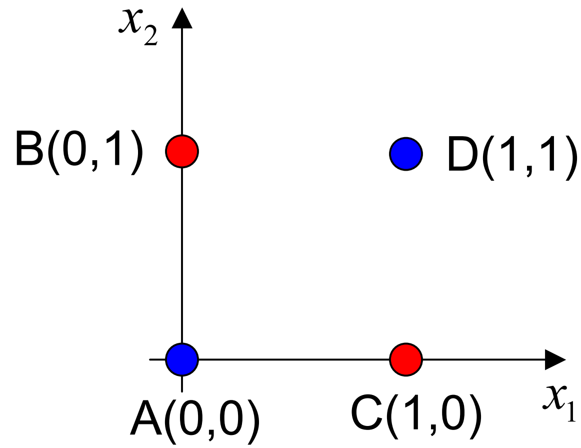
Problems with Linear SVM



What if the decision function is not linear ?

XOR Problem

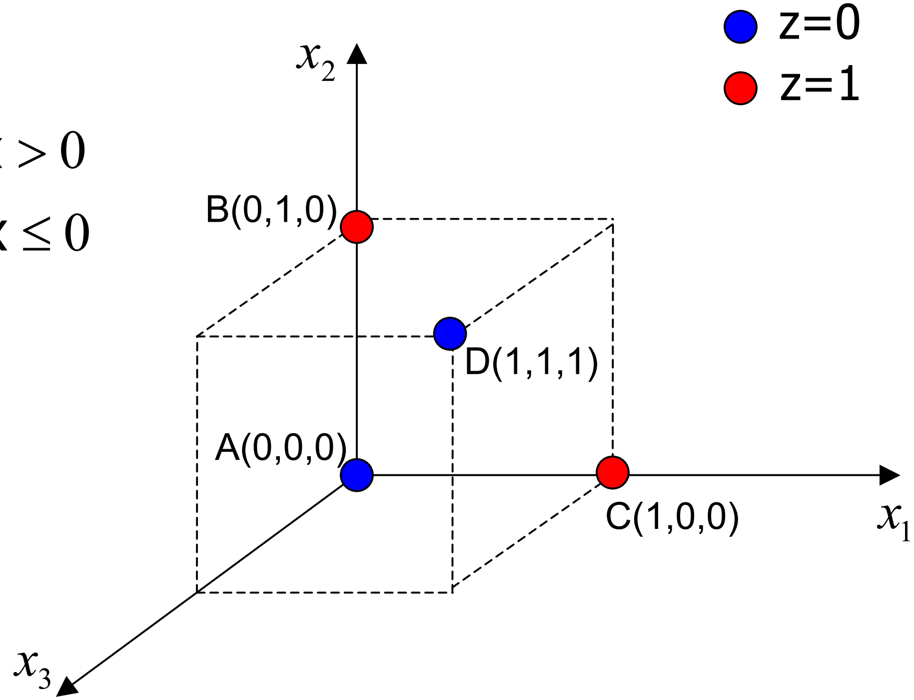
x_1	x_2	z
0	0	0
0	1	1
1	0	1
1	1	0

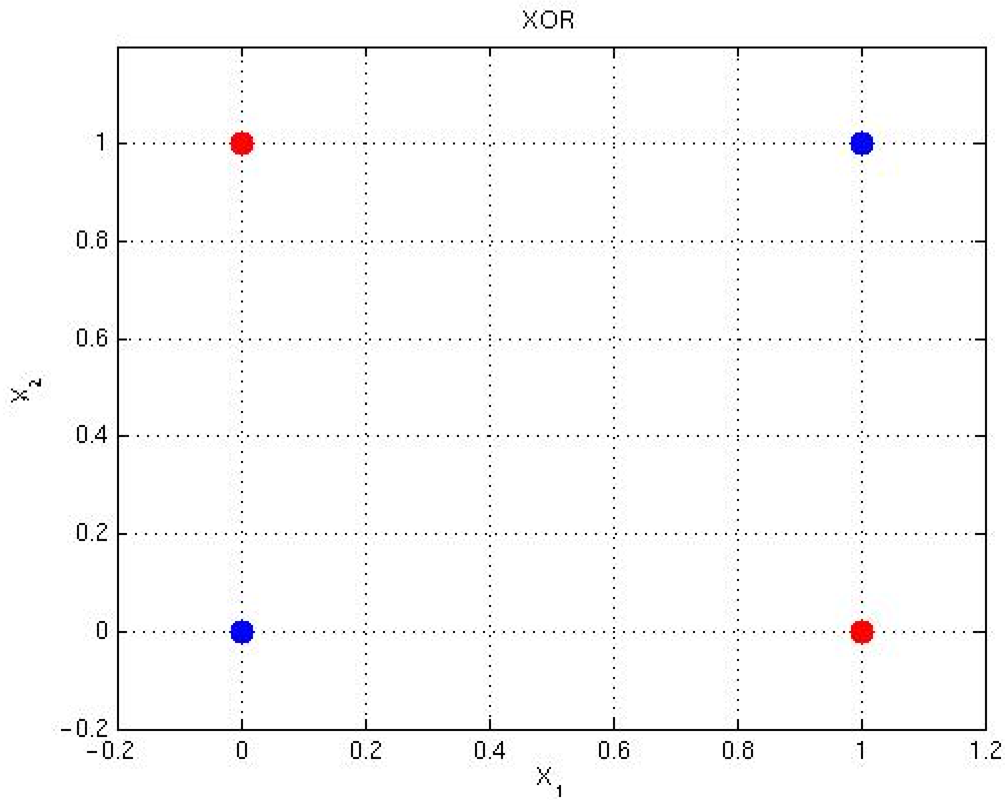


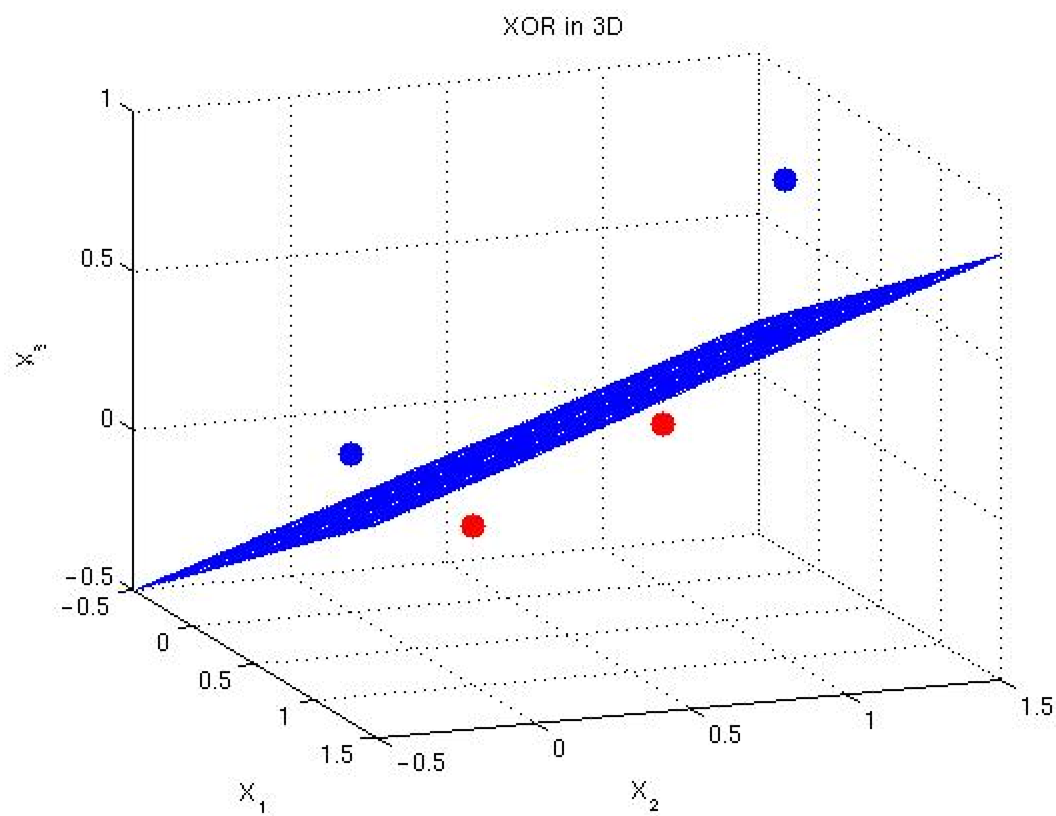
$$[x_1 \quad x_2] \Rightarrow [x_1 \quad x_2 \quad x_3]$$

x_3=T(x_1x_2–0.5) where $T(x) = \begin{cases} 1, & x > 0 \\ 0, & x \le 0 \end{cases}$

x_1	x_2	x_3	z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



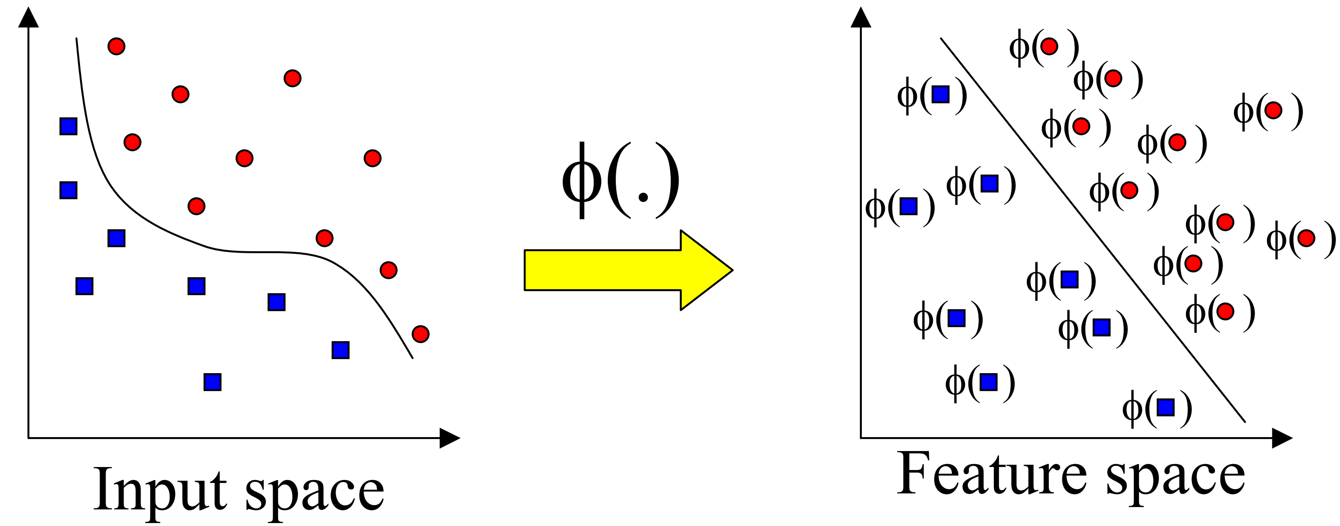

XOR
1
0.8
0.6
0.4
0.2
0
−0.2
−0.2
0
0.2
0.4
0.6
0.8
1
1.2
X_1
X_2


XOR in 3D
1
0.5
0
−0.5
X_3
−0.5
0
0.5
1
1.5
X_1
−0.5
0
0.5
1
1.5
X_2

Extension to Non-linear Decision Boundary

- So far, we only consider large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - 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
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable

Transforming the Data



- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

The Kernel Trick

- Recall the SVM optimization problem

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

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

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

An Example for $\phi(.)$ and K(.,.)

- Suppose $\phi(.)$ is given as follows

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

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)\right\rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out $\phi(.)$ explicitly

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

- This use of kernel function to avoid carrying out $\phi(.)$ explicitly is known as the kernel trick

Kernel Functions

- In practical use of SVM, only the kernel function (and not $\phi(.)$) is specified
- Kernel function can be thought of as a similarity measure between the input objects
- Not all similarity measure can be used as kernel function, however
  - The kernel function needs to satisfy the Mercer function, i.e., the function is "positive-definite"
  - This has the consequence that the kernel matrix, where the (i,j)-th entry is the $K(\mathbf{x}_i, \mathbf{x}_j)$, is always positive definite
- Note that $\mathbf{x}_i$ needs not be vectorial for the kernel function to exist. This opens up enormous opportunities for classification with sequences, graphs, etc., by SVM

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

Modification Due to Kernel Function

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

Original

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

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

With kernel function

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

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

Modification Due to Kernel Function

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

Original

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

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

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

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

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

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

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

Example

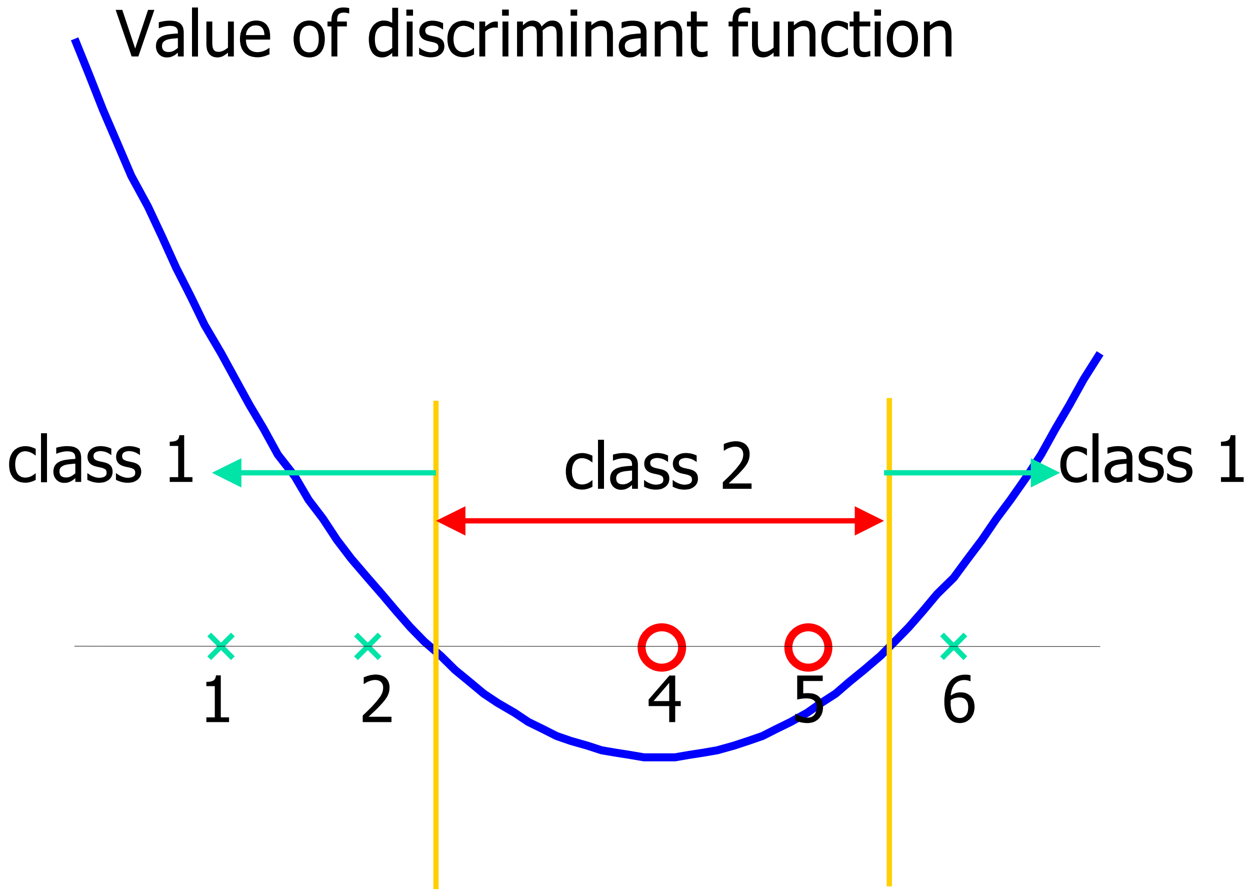
- By using a QP solver, we get
  - $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$\begin{aligned} &f(y) \\ &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on $y_i(\mathrm{w}^T\phi(z)+b)=1$ and all give b=9

$$\Rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Example



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

Summary: Steps for 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 summarize all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There are even research to estimate the kernel matrix from available information

- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks

Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts "intelligently" in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
  - Majority rule
  - Error correcting code
  - Directed acyclic graph

Why SVM Work?

- The feature space is often very high dimensional. Why don't we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is hard to estimate
- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier
- Typically, a classifier with many parameters is very flexible, but there are also exceptions
  - Let $x_i=10^i$ where i ranges from 1 to n. The classifier $y = \text{sign}\big(\sin(\alpha x)\big)$ can classify all x_i correctly for all possible combination of class labels on x_i
  - This 1-parameter classifier is very flexible

Why SVM works?

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the flexibility (capacity) of a classifier
  - This is formalized by the "VC-dimension" of a classifier
- The addition of $\frac{1}{2}||w||^2$ has the effect of restricting the VC-dimension of the classifier in the feature space
- The SVM objective can also be justified by structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized
- Another view: the SVM loss function is analogous to ridge regression. The term $\frac{1}{2}||w||^2$ "shrinks" the parameters towards zero to avoid overfitting