Image Processing Laboratory
Computer Science Department - University of Catania

Linear Learning Machines and Support Vector Machines

for classification and regression

Giovanni Maria Farinella
gfarinella@dmi.unict.it

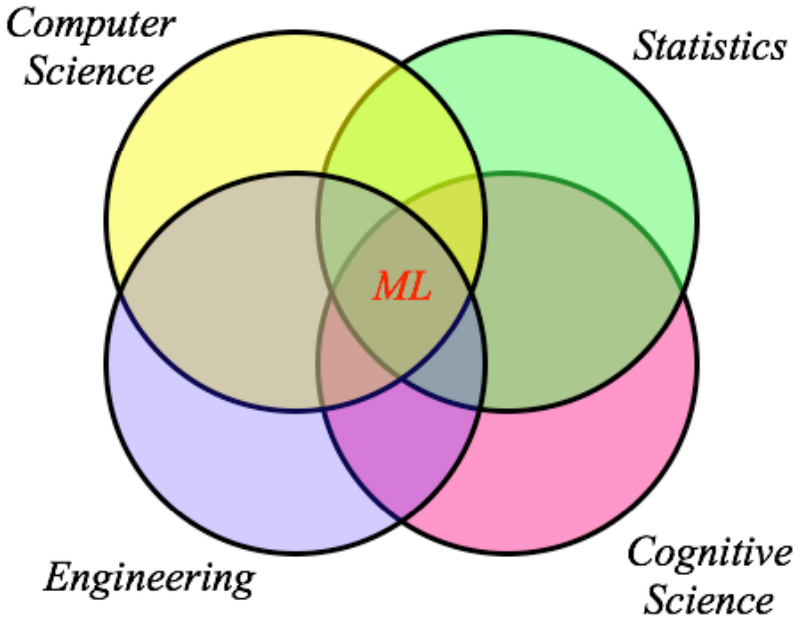
Outline of this talk

- Machine Learning and Pattern Recognition
  - Linear Learning Machine for Regression
  - Support Vector Machines
    - Classification
    - Regression

Please ask question!

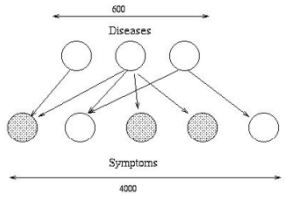
What is ML and PR?

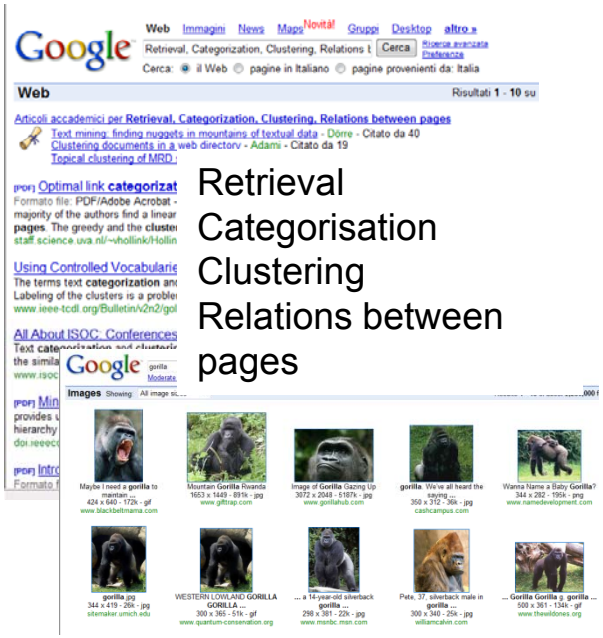
- Interdisciplinary field focusing on both the mathematical foundations and practical applications of systems that learn, reason and act.



What is ML and PR useful for?

Medical diagnosis





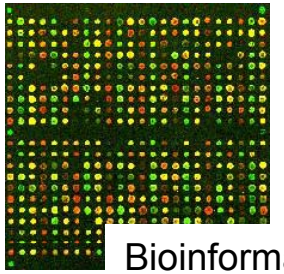
Retrieval
Categorisation
Clustering
Relations between pages

Automatic speech recognition

Object and Gesture Recognition

MITSUBISHI ELECTRIC RESEARCH LABORATORIES

Robotics



Bioinformatics

Challenge: to improve the accuracy of movie preference predictions ($1m, 2006)

Learning Approaches

- Imagine an organism/machine which experiences a series of sensory inputs:

$$x_1, x_2, \ldots x_n$$

- **Supervised learning:** the organism/machine is also given desired outputs $y_1, y_2, \ldots, y_n$, and its goal is to learn to produce the correct output given a new input.

- **Unsupervised learning:** the goal of the organism/machine is to build a model of x that can be used for reasoning, decision making, predicting things, etc.

- **Reinforcement learning:** the organism/machine can also produce actions $a_1, a_2, \ldots$ which affect the state of the world, and receives rewards (or punishments) $r_1, r_2, \ldots$. Its goal is to learn to act in a way that maximises rewards in the long term.

Typical Supervised problems

- **Classification/Recognition:** the aim is to assign each input vector to one of a finite number of category

- **Regression:** the desidered output consist of one o more continuous variables

Regression

- Regression is the determination of a general function $y=f(\mathbf{x})$ fitting a set of points $\mathbf{S}=\{(\mathbf{x_i}, y_i), i=1,\ldots n, y_i \in R\}$.
- The function depends on a number of *unknown* parameters.
- Interpretations:

  Deterministic: samples $(\mathbf{x_i}, y_i)$ are viewed as pairs of known numbers. These numbers might be the result of mesurements involving random error; however this is not used in solving the problem and the goodness of fit is not interpreted in a statistical sense.

  Statistical: $\mathbf{x_i}$ are known numbers, but y_i is the observed values of a RVS Y_i with expected value $E(y_i)$ (es: $E(y_i) = x_{11} w_1 + x_{12} w_2 + \ldots + x_{1m} w_{1m} + b$)

  Prediction: the pairs $(\mathbf{x_i}, y_i)$ are samples of two RVS $\mathbf{X}$ and Y and the objective is to find the best predictor (in terms of statistical properties) $Y^*=f(\mathbf{X})$ of Y in terms of $\mathbf{X}$

Regression framework

Learning phase

- Given a dataset $\mathbf{S}=\{(\mathbf{x}_i, y_i), i=1,\ldots n, y_i \in R\}$
- Given some hypotheses regarding f(**x**)
- Estimate parameters of f(**x**) by using **S**

Recognition phase

- Use the estimated parameters to determine f(**x**) at new **x**

Linear Learning Machine

for regression

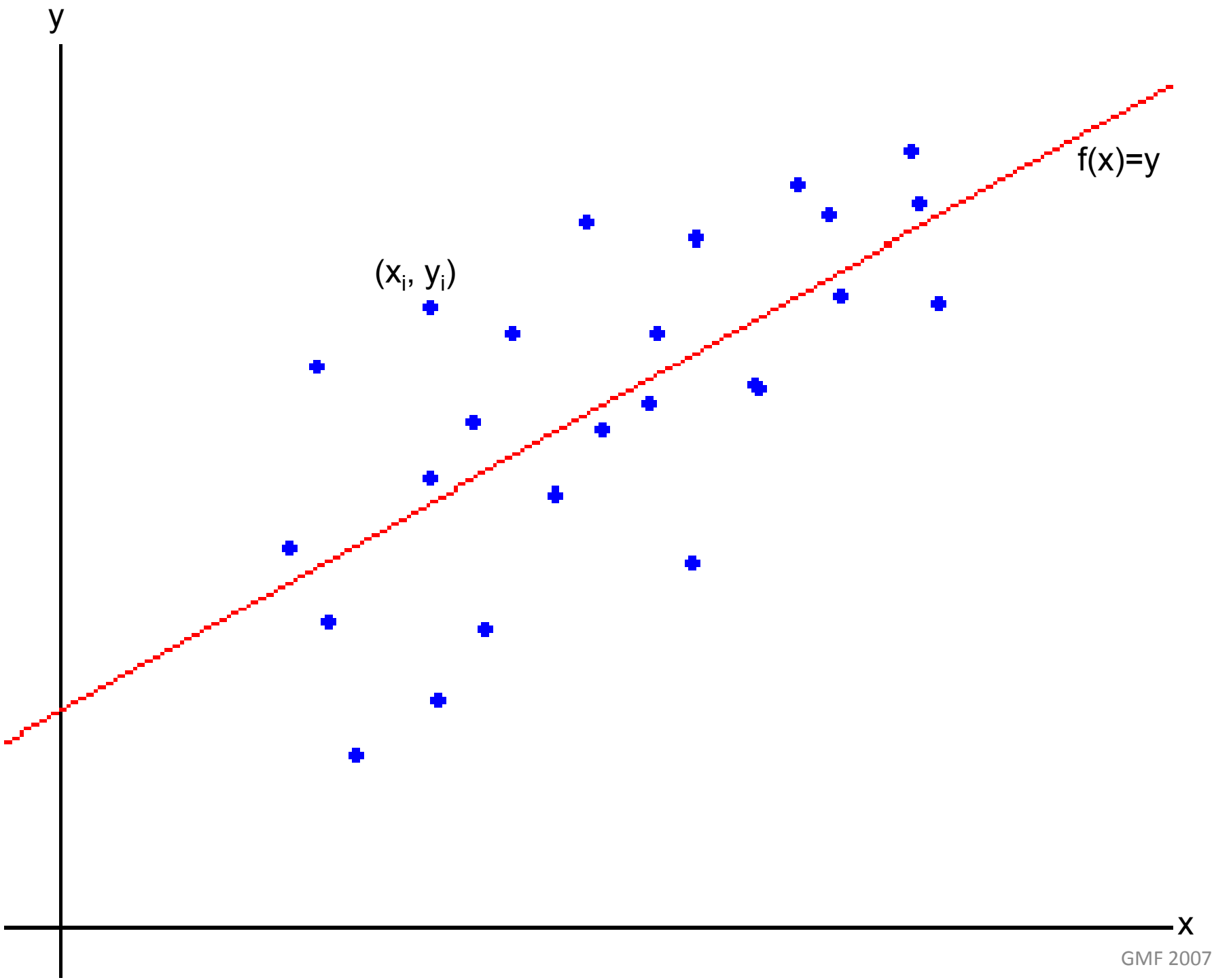
Linear Learning Machine

- Learning machines using hypotheses that form linear combination of input variables
- In particular, the problem of linear regression consist in finding a linear function

$$y = f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

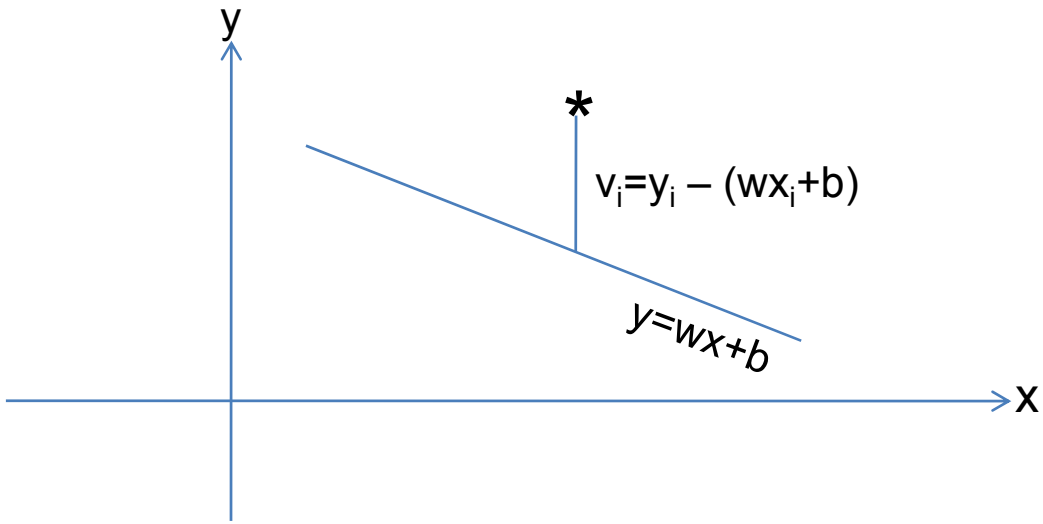
that best fits a given set **X** of training points labelled from **Y**$\subseteq$ **R**

- Interpretation: deterministic


y
f(x)=y
(x_i, y_i)
x

“fitting”

- Errors are deviations of y_i from $\left\langle \mathbf{w} \cdot \mathbf{x_i} \right\rangle + b$

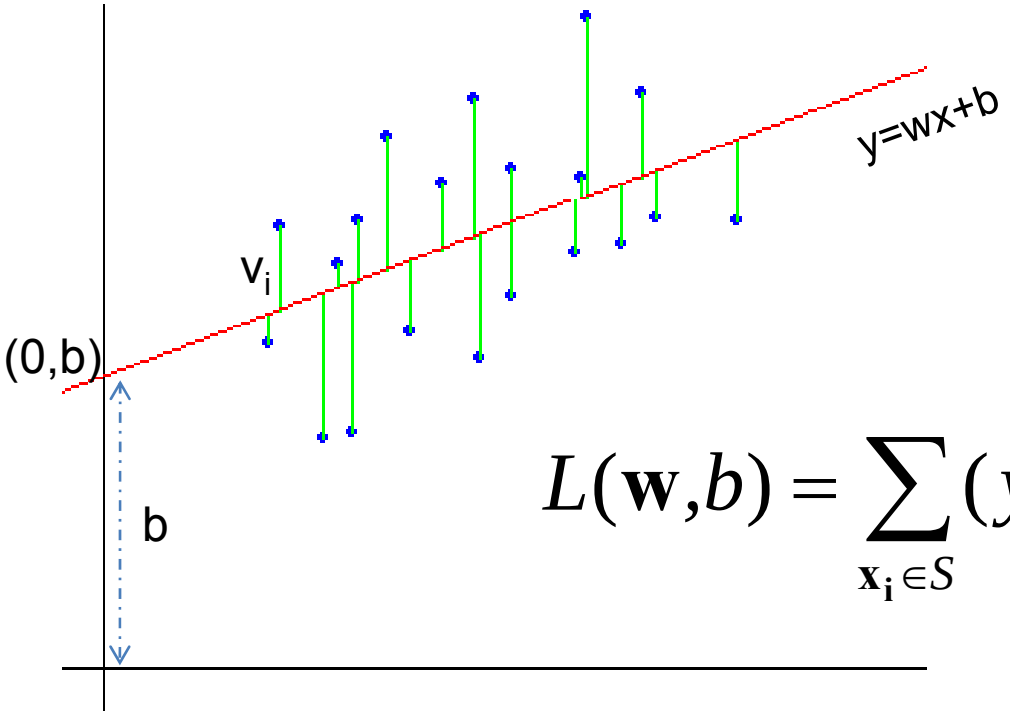


- The error can, of course, be defined in others ways.

Was first used by Gauss in the 18th century for astronomical problems.

Least Squares

- Choose the line that minimises the sum of the squares of the distances from the training points



$$L(\mathbf{w},b) = \sum_{\mathbf{x_i} \in S} (y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b)^2 = \sum_{\mathbf{x_i} \in S} v_i^{\,2}$$

Minimise the square loss function

- We can minimise *L* by differentiating with respect to the parameters (**w**,b) and setting the n+1 linear expression to zero.

$$L(\mathbf{w},b) = \sum_{\mathbf{x_i} \in S} (y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b)^2 = \sum_{\mathbf{x_i} \in S} v_i^{\,2}$$

Matrix Notation:

$$\hat{\mathbf{w}} = (\mathbf{w}', b)' \qquad \hat{\mathbf{x}}_\mathbf{i} = (\mathbf{x}_\mathbf{i}', 1)' \qquad \hat{\mathbf{X}} = \begin{pmatrix} \hat{\mathbf{x}}_\mathbf{1}' \\ \vdots \\ \hat{\mathbf{x}}_\mathbf{n}' \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$L(\hat{\mathbf{w}}) = (\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})'(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}}) = \mathbf{v}'\mathbf{v}$$

$$\frac{\partial L}{\partial \hat{\mathbf{w}}} = -2\hat{\mathbf{X}}'\mathbf{y} + 2\hat{\mathbf{X}}'\hat{\mathbf{X}}\hat{\mathbf{w}} = \mathbf{0} \qquad \mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}} = \begin{pmatrix} y_1 - \sum_l w_l x_{1l} - b \\ \vdots \\ y_n - \sum_l w_l x_{nl} - b \end{pmatrix}$$

$\hat{\mathbf{X}}'\hat{\mathbf{X}}\hat{\mathbf{w}} = \hat{\mathbf{X}}'\mathbf{y}$ Normal equations

If $\det(\hat{\mathbf{X}}'\hat{\mathbf{X}}) \neq 0$ the inverse of $\hat{\mathbf{X}}'\hat{\mathbf{X}}$ exist, and the solution of the least square problem is

$$\hat{\mathbf{w}} = (\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}'\mathbf{y}$$

If $\hat{\mathbf{X}}'\hat{\mathbf{X}}$ is singular, the pseudo-inverse (obtained through SVD) can be used or the technique of ridge regression can be applied.

Ridge Regression

- The Problem:
  - There is not enough data to ensure that the matrix $\hat{\mathbf{X}}'\hat{\mathbf{X}}$ is invertible → ill-conditioned problem
  - noise in the data: unwise to try to match the target output exactly

- Regularization Approach:
  - Restrict the choice of function in some way.
  - Simplest regularizer: favour function that have small norm → Ridge Regression

Linear Regression: Dual Representation

If the inverse of $\hat{\mathbf{X}}'\hat{\mathbf{X}}$ exists

$$\hat{\mathbf{w}} = (\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}'\mathbf{y} = \mathbf{I}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}'\mathbf{y} =$$

$$= \hat{\mathbf{X}}'\hat{\mathbf{X}}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}'\mathbf{y} =$$

$$= \hat{\mathbf{X}}'\hat{\mathbf{X}}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-2}\hat{\mathbf{X}}'\mathbf{y} =$$

$$= \hat{\mathbf{X}}'(\hat{\mathbf{X}}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-2}\hat{\mathbf{X}}'\mathbf{y}) = \hat{\mathbf{X}}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \hat{\mathbf{X}}(\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-2}\hat{\mathbf{X}}'\mathbf{y}$$

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

The solution is a linear combination of the training points.

Linear Regression: Dual Representation

$$\boldsymbol{\alpha} = \hat{\mathbf{X}} \left(\hat{\mathbf{X}}' \hat{\mathbf{X}} \right)^{-2} \hat{\mathbf{X}}' \mathbf{y} =$$

$$= \hat{\mathbf{X}} \left(\hat{\mathbf{X}}' \hat{\mathbf{X}} \right)^{-1} \left(\hat{\mathbf{X}}' \hat{\mathbf{X}} \right)^{-1} \hat{\mathbf{X}}' \mathbf{y} =$$

$$= \hat{\mathbf{X}} \hat{\mathbf{X}}^{-1} \hat{\mathbf{X}}'^{-1} \left(\hat{\mathbf{X}}' \hat{\mathbf{X}} \right)^{-1} \hat{\mathbf{X}}' \mathbf{y} =$$

$$= \mathbf{I} \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}^{-1} \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}' \mathbf{y} =$$

$$= \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}^{-1} \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}' \mathbf{y} =$$

$$= \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}^{-1} \mathbf{I} \mathbf{y} =$$

$$= \hat{\mathbf{X}}'^{-1} \hat{\mathbf{X}}^{-1} \mathbf{y} = \left(\hat{\mathbf{X}} \hat{\mathbf{X}}' \right)^{-1} \mathbf{y} = \hat{\mathbf{G}}^{-1} \mathbf{y}$$

The Gram Matrix

$$\hat{\mathbf{G}} = \hat{\mathbf{X}}\hat{\mathbf{X}}'$$

$$\hat{\mathbf{G}}_{ij} = < \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j >$$

Dot product is a type of similarity measure that is of particular mathematical appeal.

Geometrical Interpretation: it computes the cosine of the angle between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, provided they are normalized to length 1. Moreover, it allows the computation of the length of a vector $\mathbf{x}$ as sqrt(<$\mathbf{x}$,$\mathbf{x}$>), and of the distance between two vectors as the length of the difference vector. Therefore all geometrical construction that can be formulated in terms of angles, lengths and distances can be carried out.

Linear Regression: Dual Representation

$$\boldsymbol{\alpha} = \hat{\mathbf{G}}^{-1}\mathbf{y} \qquad \hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i \hat{\mathbf{x}}_i$$

Solving for α involve solving n linear equation with n unknown.
Complessity $O(n^3)$ rather than $O(d^3)$

$$f(\mathbf{x}) = \langle \hat{\mathbf{w}} \cdot \hat{\mathbf{x}} \rangle = \left\langle \left(\sum_{i=1}^{n} \alpha_i \hat{\mathbf{x}}_i \right) \cdot \hat{\mathbf{x}} \right\rangle =$$

The dual representation depend on the Gram Matrix of the inner product of the training example.

$$= \sum_{i=1}^{n} \alpha_i \langle \hat{\mathbf{x}}_i \cdot \hat{\mathbf{x}} \rangle = \sum_{i=1}^{n} \alpha_i k_i$$

The information about a novel example **x** required by the predictive function is just the inner product between training points and the new example

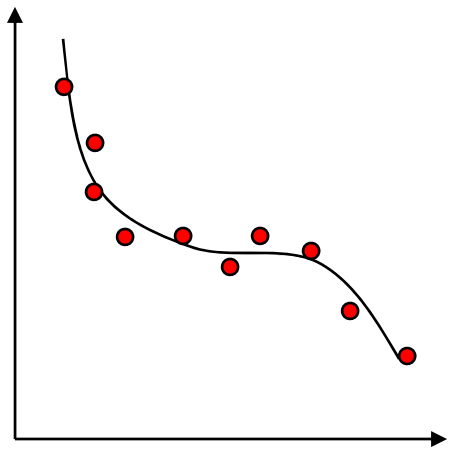
$$k_i = \langle \hat{\mathbf{x}}_i \cdot \hat{\mathbf{x}} \rangle$$

If the dimension of the feature space is larger than the number of training examples solving dual representation is more efficient.

Evaluation complexity O(nd) rather than O(d)

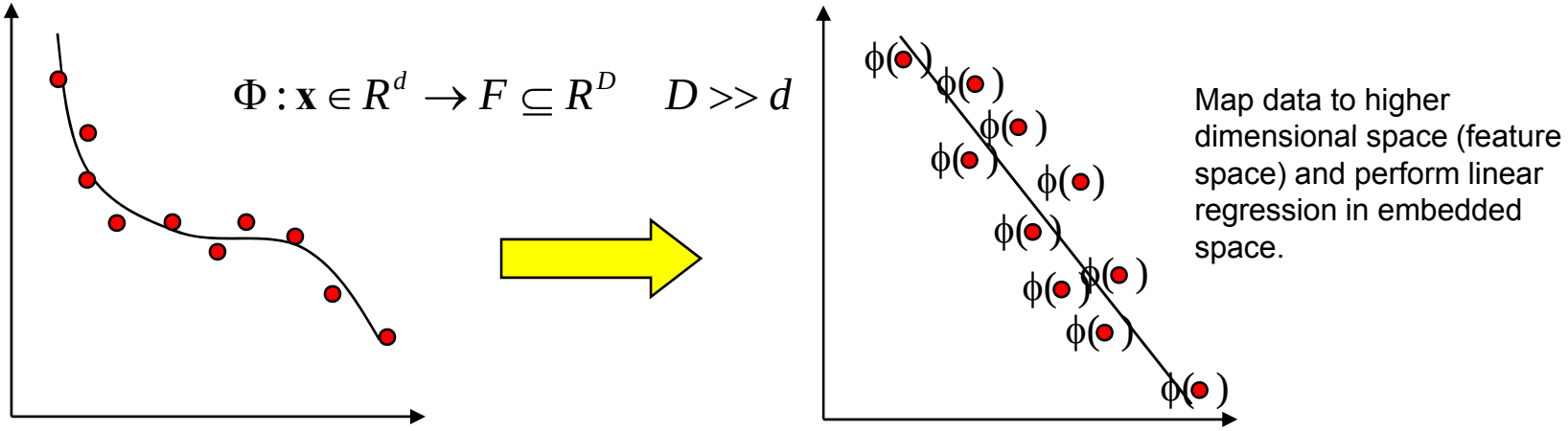
Nonlinear features space

- Linear Regression: address the problem of identifying linear relations between one selected variable and the input variables where the relation is assumed linear
- Often, however, the relations that are sought are nonlinear



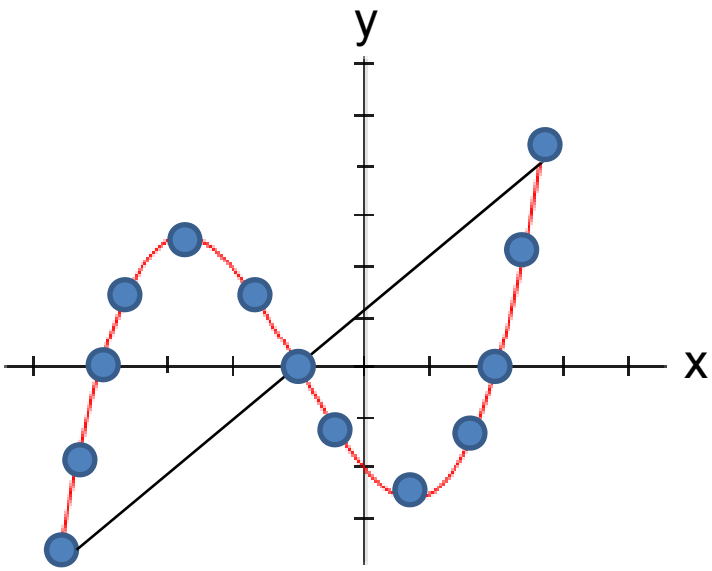
Nonlinear features space

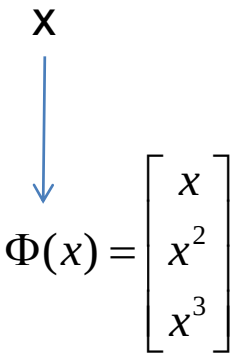
- Key Idea: Map the input space into a new features space in such way that the sought relations can be represented in a linear form and hence the linear regression algorithm described above will be able to detect them.



The choice of the map ϕ aims to convert the nonlinear relations into linear ones. Hence the map reflect our expectations about the relation y=f(x)

Polynomial features space





$f(x) = w_1 x + b$

$f(x) = \langle \mathbf{w}, \phi(x) \rangle + b = w_1 x + w_2 x^2 + w_3 x^3 + b$

$w_1 = 1/4$
$w_2 = 1/4$
$w_3 = -1/2$
$B = -2$

The effect of ϕ

- Recode our dataset $\mathbf{S}=\{(\mathbf{x}_i, y_i), i=1,\ldots n, y_i \in R\}$ as $\mathbf{S}_\phi = \{(\phi(\mathbf{x}_i), y_i), i=1,\ldots n, y_i \in R\}$

$$\hat{\mathbf{G}} = \hat{\mathbf{X}}\hat{\mathbf{X}}'$$

$$\hat{\mathbf{G}}_{ij} = < \mathbf{\Phi}(\hat{\mathbf{x}}_i), \mathbf{\Phi}(\hat{\mathbf{x}}_j) >$$

$$k_i = \left\langle \Phi(\hat{\mathbf{x}}_i) \cdot \Phi(\hat{\mathbf{x}}) \right\rangle$$

Although the primal form could be used, problem will arise if d is very large making the solution of the dxd system very expensive.
$O(d^3)$ $O(d)$
If, on the other hand, we consider the dual solution, we have shown that all the information the algorithm need is the inner products between data points in the features space.
$O(n^3+n^2d)$ $O(nd)$

Kernel-Induced Features Spaces

- The inner products can sometime be computed more efficiently as direct function of the input variables, without explicity computing the mapping Φ. In other words mapping step can be by-passed.
- A function that perform this direct computation is known as kernel function

Kernel function

- A kernel is a function *k* that for all **x**, **z** of the input space satisfies

$k(\mathbf{x},\mathbf{z})=<\phi(\mathbf{x}),\phi(\mathbf{z})>$

Where ϕ is a mapping from the input space to an (inner product) feature space F

Note: Kernels make possible the use of feature spaces with an exponential or even infinite number of dimensions.

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

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

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

- An inner product in the feature space is

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

- 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

Some Kernel Fuctions

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\langle \mathbf{x_i}, \mathbf{x_j} \rangle + R)^d \quad \textbf{polynomial}$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\|\mathbf{x_i} - \mathbf{x_j}\|^2 / (2\sigma^2)) \quad \textbf{radial basis functions}$$

Can be inner product in Infinite dimensional space

Assume $x \in R^1$ and $\gamma > 0$.

$$e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 \right.$$

$$\left. + \sqrt{\frac{(2\gamma)^3}{3!}} x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} x_j^3 + \cdots\right) = \phi(x_i)^T \phi(x_j),$$

where

$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \cdots\right]^T.$$

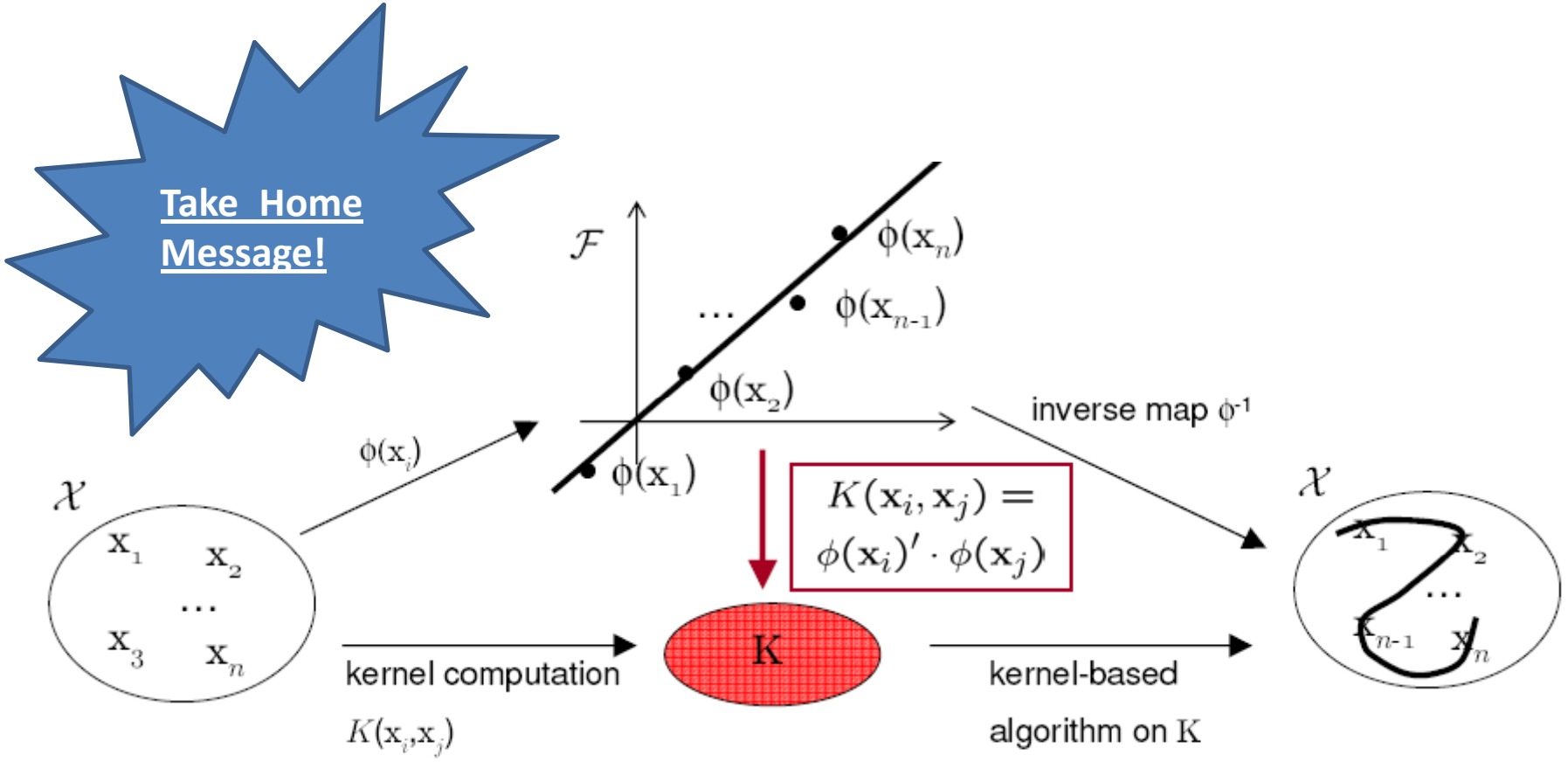
Mercer's Theorem

- The kernel matrix (Gram Martix) is Symmetric Positive Definite (**x'Gx ≥ 0**)
- Any symmetric positive definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space

Given a function k, it is possible to verify that it is a kernel.

More Formal: Every (semi) positive definite, symmetric function is a kernel, i.e., there exists a mapping Φ such that it is possible to write $k(x_1,x_2)=< \Phi(x_1), \Phi(x_2)>$

Kernel Methods Framework



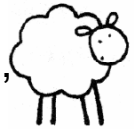
Support Vector Machines

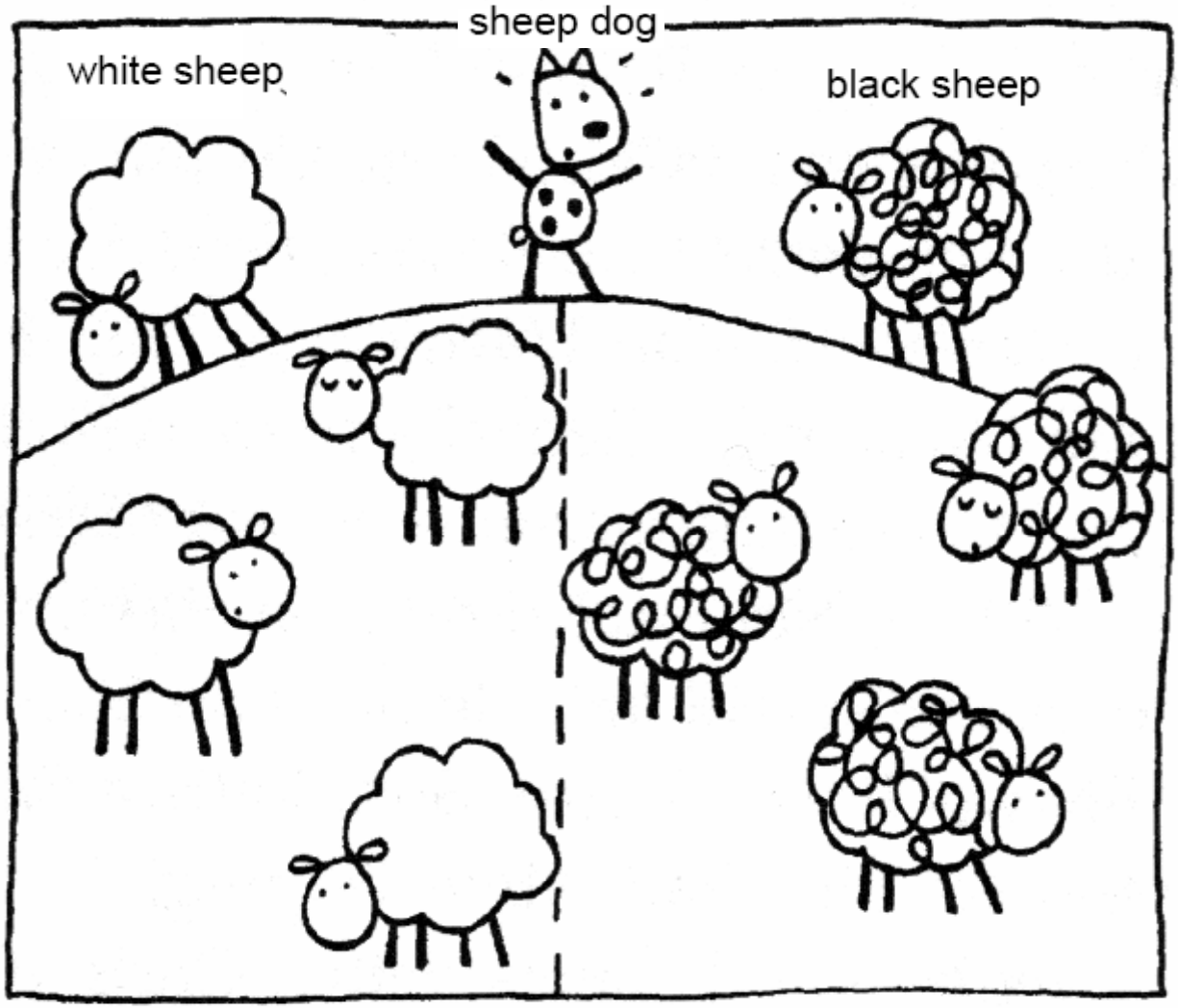
for classification and regression

Let me start with two class classification problem...

f(sheep_features)=y

sheep_features is a vector ϵ dot product features space

y ϵ { = -1,  =1 }



Hyperplane classifier

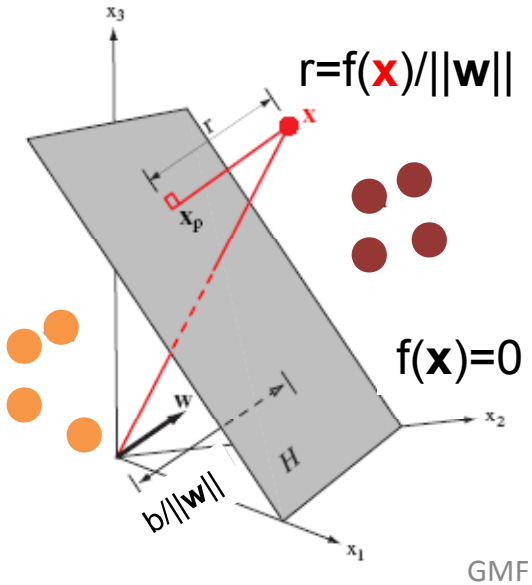
- Theorem: In the *separable case* exist <u>at least one</u> choice of parameters **w,** b such that

$$f(\mathbf{x})=<\mathbf{w},\mathbf{x}>+b$$

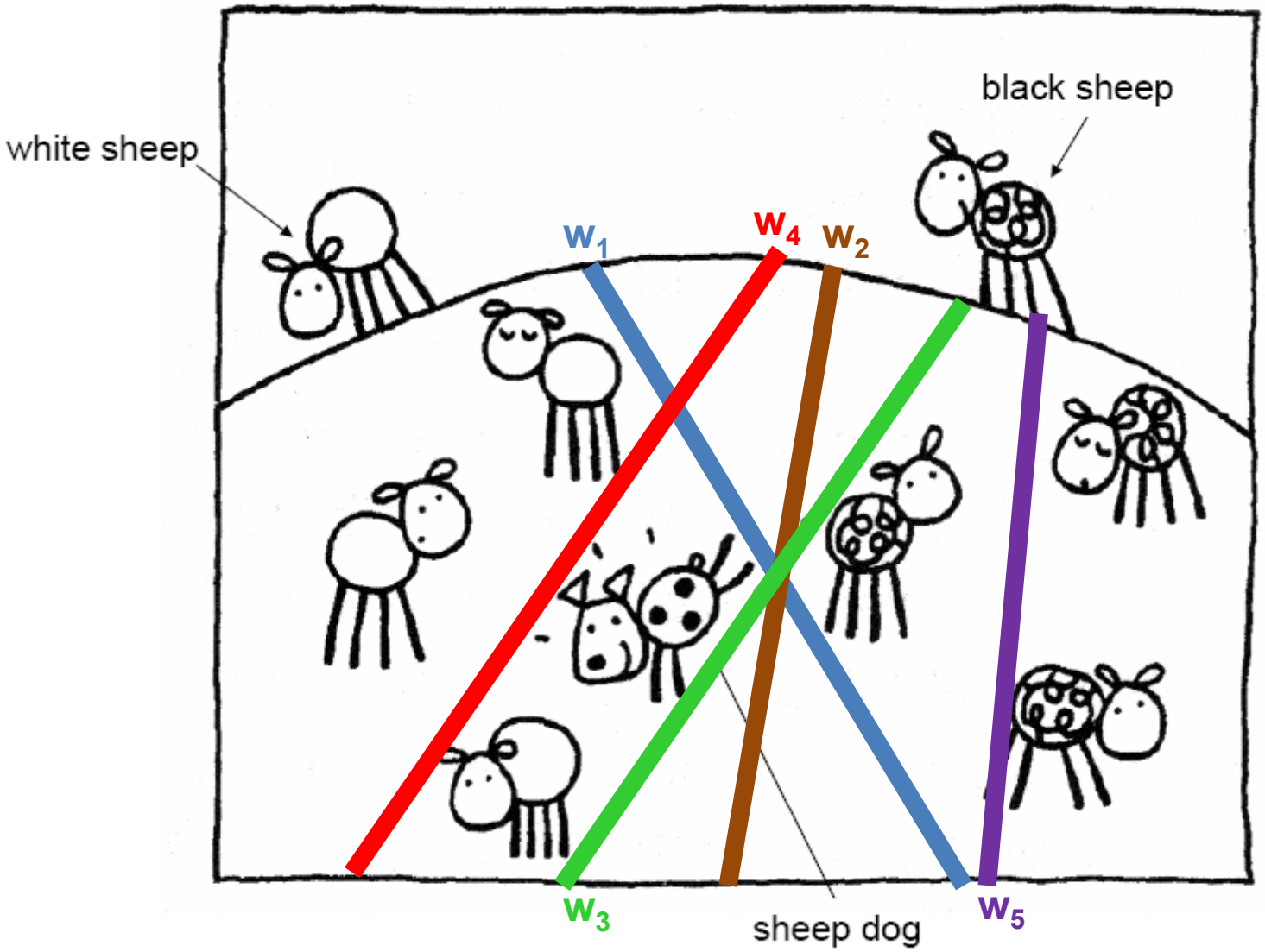
satisfy

$$y_i f(\mathbf{x}_i)>0 \quad i=1...n$$

Decision function: sng(f(**x**))

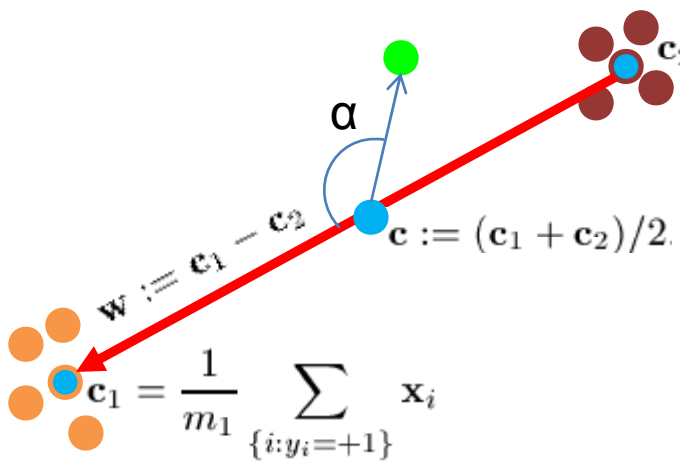


How choose the Hyperplane?



The simplest Hyperplane Classifier

- Compute the mean of the two classes in the features space
- Assign a new point **x** to the class whose mean is closest to it.



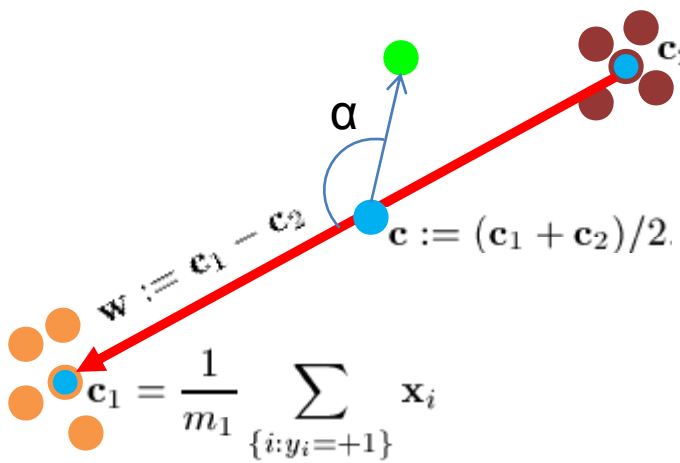
decision function

$$
\begin{aligned}
y &= \operatorname{sgn}\left((\mathbf{x} - \mathbf{c}) \cdot \mathbf{w}\right) \\
y &= \operatorname{sgn}\left((\mathbf{x} - (\mathbf{c}_1 + \mathbf{c}_2)/2) \cdot (\mathbf{c}_1 - \mathbf{c}_2)\right) \\
&= \operatorname{sgn}\left((\mathbf{x} \cdot \mathbf{c}_1) - (\mathbf{x} \cdot \mathbf{c}_2) + b\right).
\end{aligned}
$$

$$
b := \frac{1}{2}\left(\|\mathbf{c}_2\|^2 - \|\mathbf{c}_1\|^2\right)
$$

The simplest Hyperplane Classifier

- Compute the mean of the two classes in the features space
- Assign a new point **x** to the class whose mean is closest to it.



decision function

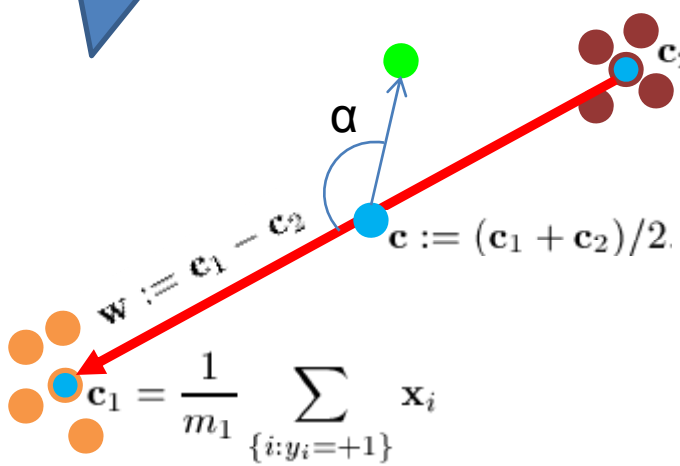
$$
\begin{aligned}
y &= \operatorname{sgn}\left((\mathbf{x} - \mathbf{c}) \cdot \mathbf{w}\right) \\
y &= \operatorname{sgn}\left((\mathbf{x} - (\mathbf{c}_1 + \mathbf{c}_2)/2) \cdot (\mathbf{c}_1 - \mathbf{c}_2)\right) \\
&= \operatorname{sgn}\left((\mathbf{x} \cdot \mathbf{c}_1) - (\mathbf{x} \cdot \mathbf{c}_2) + b\right).
\end{aligned}
$$

$$
b := \frac{1}{2}\left(\|\mathbf{c}_2\|^2 - \|\mathbf{c}_1\|^2\right)
$$

The simplest Hyperplane Classifier

- Comput[illegible]an of the two classes in the feat[illegible]
- As[illegible]x to the class whose mean is[illegible]t.

This geometrical construction can be formulated in terms of dot product!



decision function

$$
\begin{aligned}
y &= \operatorname{sgn}\left((\mathbf{x} - \mathbf{c}) \cdot \mathbf{w}\right) \\
y &= \operatorname{sgn}\left((\mathbf{x} - (\mathbf{c}_1 + \mathbf{c}_2)/2) \cdot (\mathbf{c}_1 - \mathbf{c}_2)\right) \\
&= \operatorname{sgn}\left((\mathbf{x} \cdot \mathbf{c}_1) - (\mathbf{x} \cdot \mathbf{c}_2) + b\right).
\end{aligned}
$$

$$
b := \frac{1}{2}\left(\|\mathbf{c}_2\|^2 - \|\mathbf{c}_1\|^2\right)
$$

The simplest Hyperplane Classifier

decision function

$$y = \operatorname{sgn}\left(\frac{1}{m_1}\sum_{\{i:y_i=+1\}}(\mathbf{x}\cdot\mathbf{x}_i)-\frac{1}{m_2}\sum_{\{i:y_i=-1\}}(\mathbf{x}\cdot\mathbf{x}_i)+b\right)$$

$$= \operatorname{sgn}\left(\frac{1}{m_1}\sum_{\{i:y_i=+1\}}k(x,x_i)-\frac{1}{m_2}\sum_{\{i:y_i=-1\}}k(x,x_i)+b\right).$$

$$b := \frac{1}{2}\left(\frac{1}{m_2^2}\sum_{\{(i,j):y_i=y_j=-1\}}k(x_i,x_j)-\frac{1}{m_1^2}\sum_{\{(i,j):y_i=y_j=+1\}}k(x_i,x_j)\right)$$

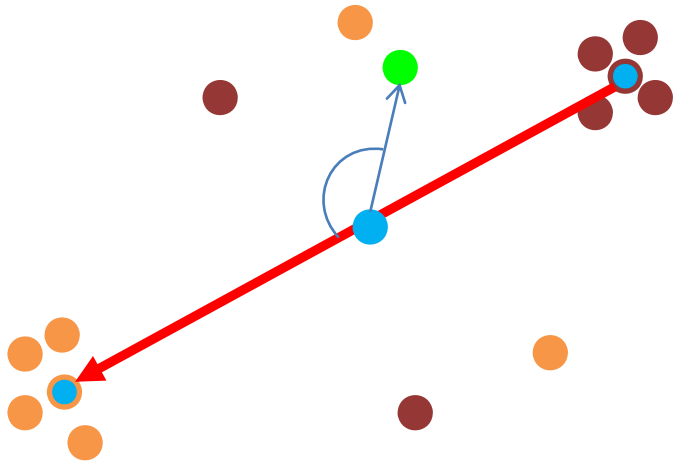
$$k(x,x') := (\mathbf{x}\cdot\mathbf{x}') = (\Phi(x)\cdot\Phi(x'))$$

This classifier **is quite close to the type of learning machine that we are interested in**.
It is linear in the features space, while the input domain, it is represented by kernel expansion.
It is exampled based in the sense that the kernel are centered on the training samples.
The main point where the more sophisticated techniques to be discussed later will deviate from the above classifier is in the selection of examples that the kernels are centered on, and in the weight that is put on the individual kernels in the decision function.

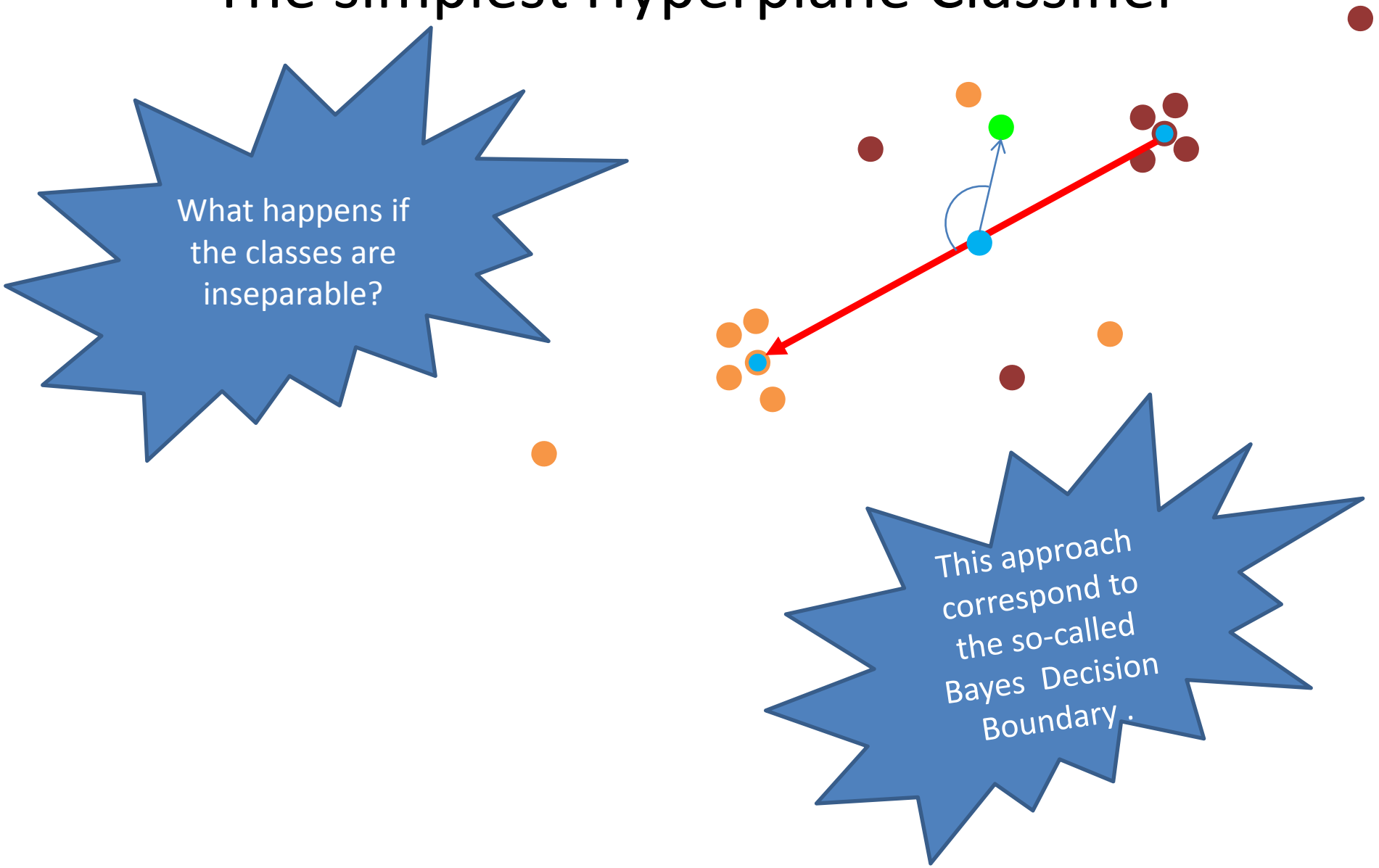
In the features space representation, this statement corresponds to saying that we will study all normal vector w of the decision hyperlplanes that can be represented as linear combination of the training examples. For instance, we want to remove the influence of patterns that are very far form the decision boundary, either since we expect that they will not improve the generalization error of the decision function, or since we would like to reduce the computational cost of evaluating the decision function.
The hyperplane will then only depend on a subset of training examples, called support vectors.

The simplest Hyperplane Classifier

What happens if the classes are inseparable?

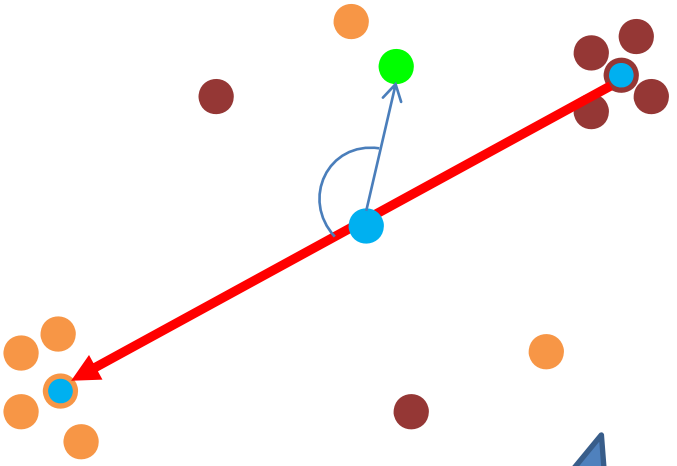


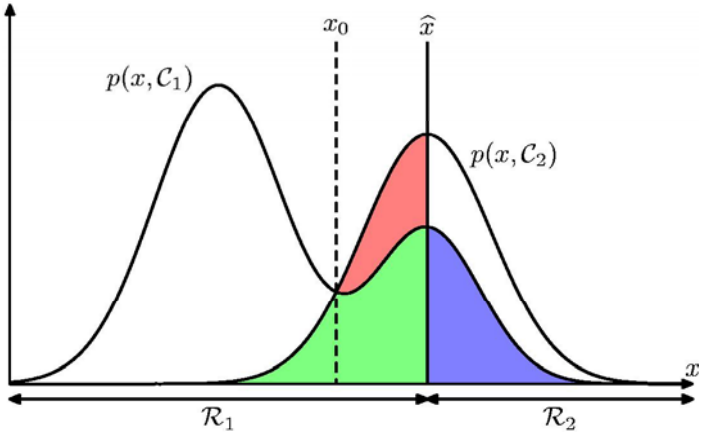
The simplest Hyperplane Classifier



The simplest Hyperplane Classifier

What happens if the classes are inseparable?



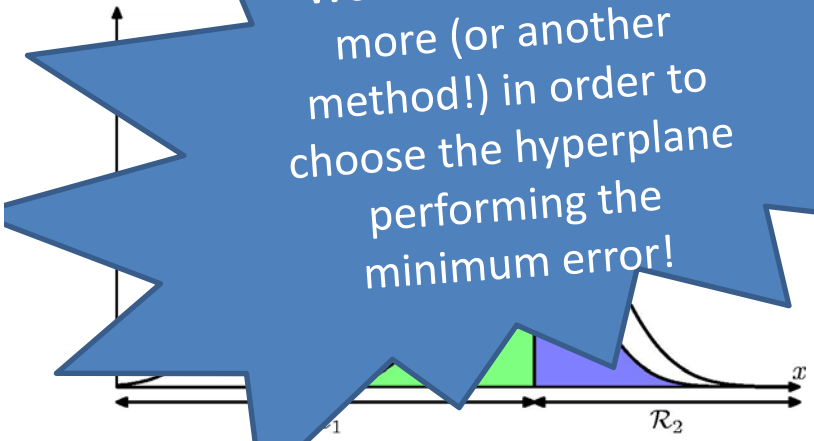


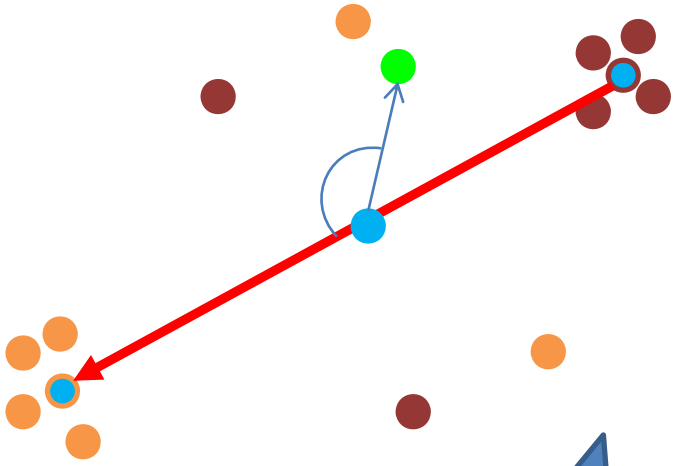
This approach correspond to the so-called Bayes Decision Boundary .

The simplest Hyperplane Classifier

What happens if the classes are inseparable?

We need something more (or another method!) in order to choose the hyperplane performing the minimum error!





This approach correspond to the so-called Bayes Decision Boundary .

Special Case

Let us consider one well-known special case of this type of classifier. Assume that the class means have the same distance to the origin (hence $b = 0$), and that k can be viewed as a density, i.e. it is positive and has integral 1,

$$\int_{\mathcal{X}} k(x, x')dx = 1 \quad \text{for all } x' \in \mathcal{X}. \tag{12}$$

In order to state this assumption, we have to require that we can define an integral on $\mathcal{X}$.

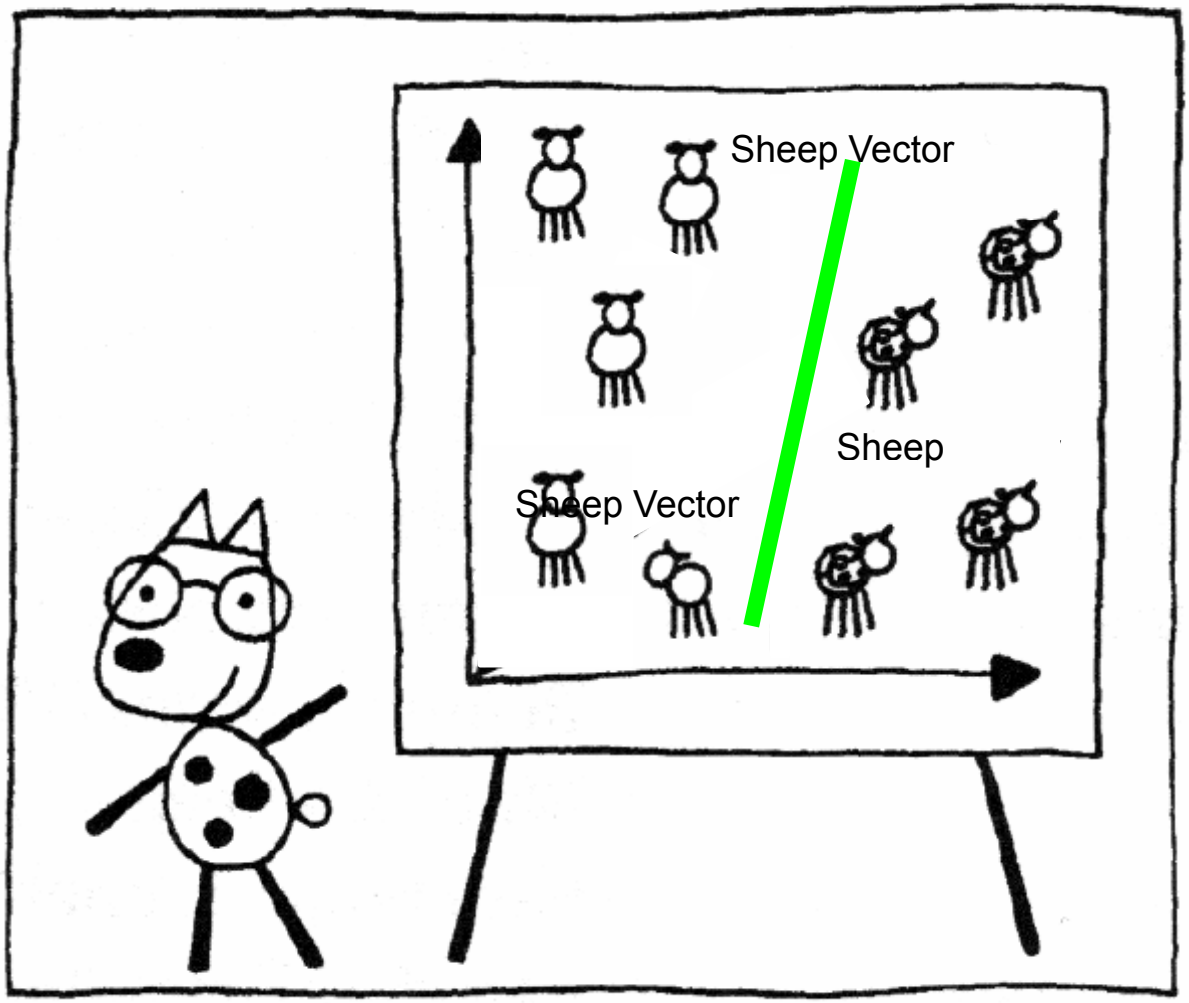
If the above holds true, then (10) corresponds to the so-called Bayes decision boundary separating the two classes, subject to the assumption that the two classes were generated from two probability distributions that are correctly estimated by the *Parzen windows* estimators of the two classes,

$$p_1(x) := \frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i) \tag{13}$$

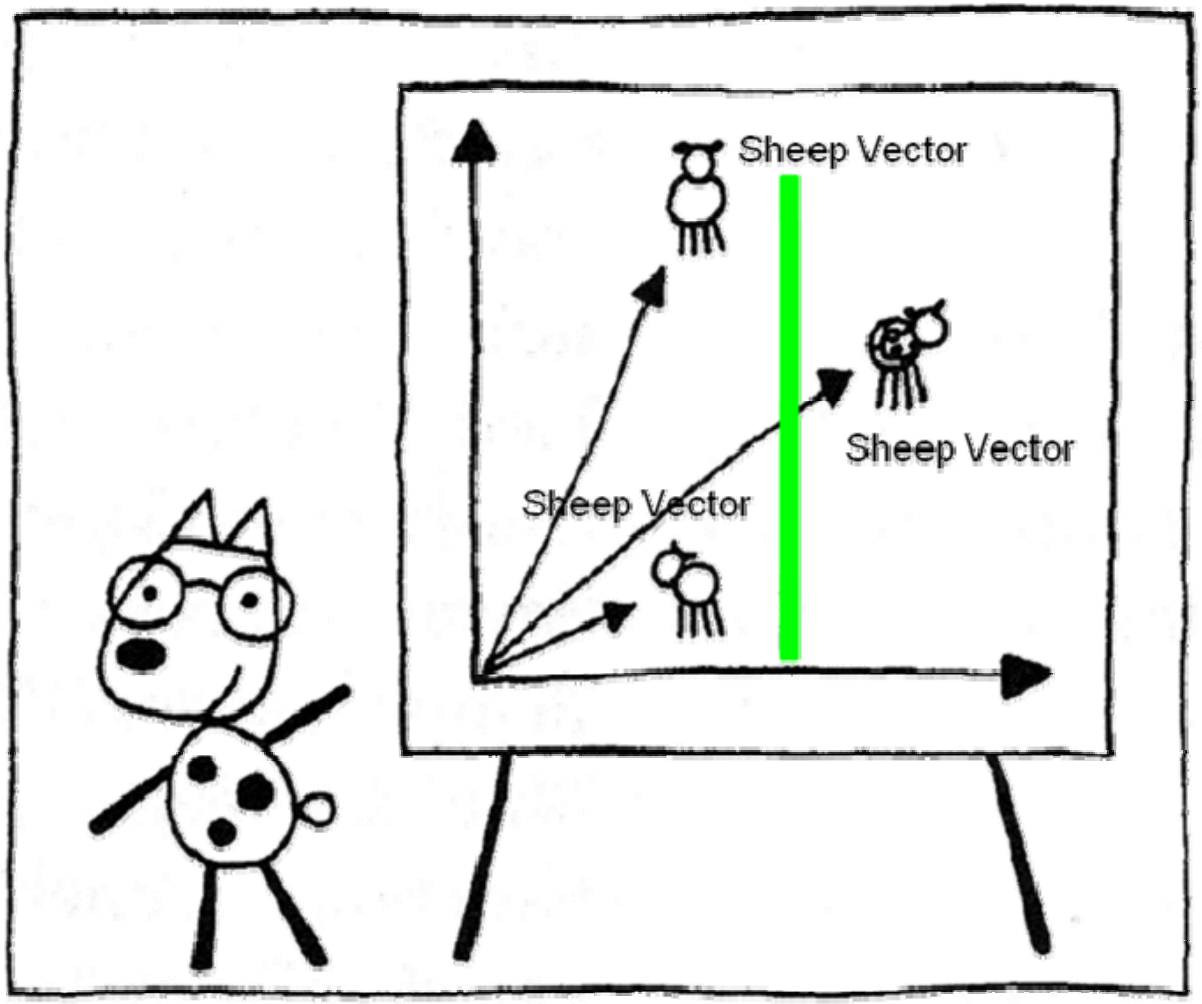
$$p_2(x) := \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i). \tag{14}$$

Given some point x, the label is then simply computed by checking which of the two, $p_1(x)$ or $p_2(x)$, is larger, which directly leads to (10). Note that this decision is the best we can do if we have no prior information about the probabilities of the two classes.

From the simplest Hyperplane Classifier



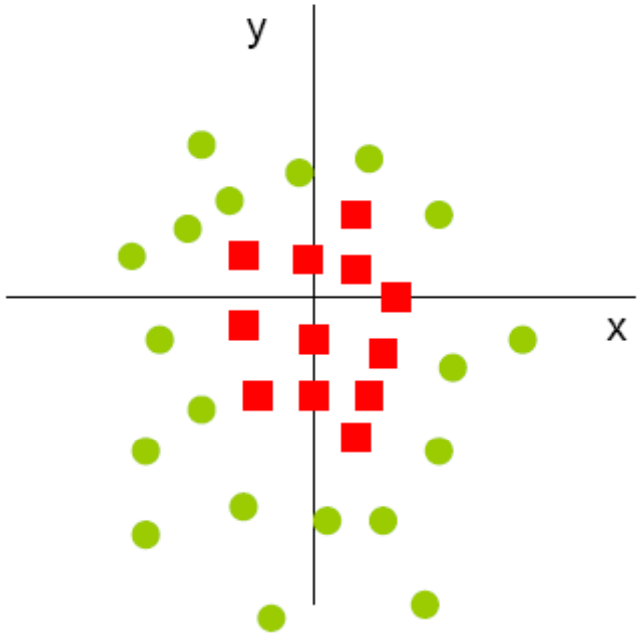
To Support Vector Machine



Non Linear separating plane

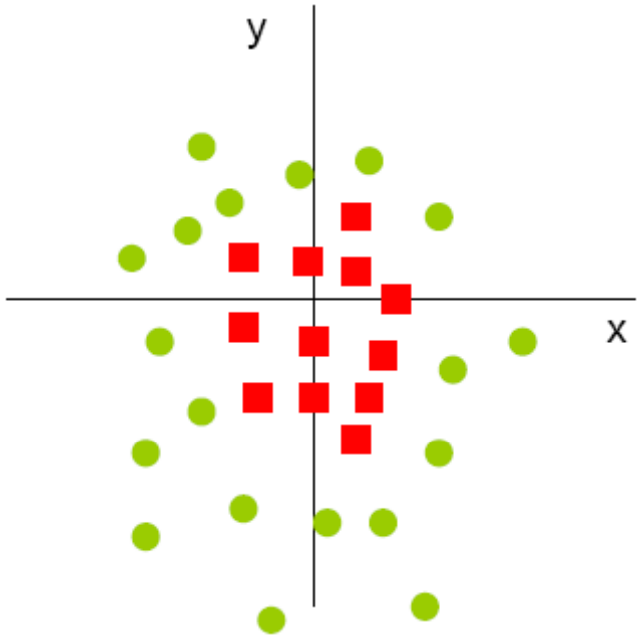
Linear Classifiers not very flexible/powerful?

Can we do better?



Non Linear separating plane

- Add distance to origin $(x^2+y^2)^{1/2}$ as a third feature
- Data now lives on a parabolic surface in 3D
- Linear separation in 3D
- In original feature space, boundary is an ellipse



Non Linear separating plane

- Data has been mapped to a new, higher dimensional space
- An alternatively way to think about this: data still lives in original space but the definition of *distance* or *inner product* has been changed (it is possible to use kernels to do this).
- Suddenly *linear methods* become *non-linear*

Multi-class Classification: One-against-the rest

- K classes
- train k binary classifier

1^{st} class vs. $(2\text{-}k)^{th}$ class

2^{st} class vs. $(1,3\text{-}k)^{th}$ class

:

K decision functions

$$f_1(\mathbf{x}) = \langle \mathbf{w_1}, \mathbf{x} \rangle + b_1$$

:

$$f_k(\mathbf{x}) = \langle \mathbf{w_k}, \mathbf{x} \rangle + b_k$$

Multi-class Classification: One-against-the rest

- Prediction

$$\text{argmax}_j \left(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j\right)$$

Reason: if the 1st class, then we showld have:

$f_1(\mathbf{x}) >= +1$

$f_2(\mathbf{x}) <= -1$

:

$f_k(\mathbf{x}) <= -1$

Multi-class Classification: One-against-one

- Train k(k-1)/2 binary classifier
- (1,2), (1,3), (1,4)... (1,k), (2,3), (2,4) ... , (k-1,k)

- If 4 classes→6 binary classifier

$y_i = 1$	$y_i = -1$	Decision functions
class 1	class 2	$f^{12}(\mathbf{x}) = (\mathbf{w}^{12})^T\mathbf{x} + b^{12}$
class 1	class 3	$f^{13}(\mathbf{x}) = (\mathbf{w}^{13})^T\mathbf{x} + b^{13}$
class 1	class 4	$f^{14}(\mathbf{x}) = (\mathbf{w}^{14})^T\mathbf{x} + b^{14}$
class 2	class 3	$f^{23}(\mathbf{x}) = (\mathbf{w}^{23})^T\mathbf{x} + b^{23}$
class 2	class 4	$f^{24}(\mathbf{x}) = (\mathbf{w}^{24})^T\mathbf{x} + b^{24}$
class 3	class 4	$f^{34}(\mathbf{x}) = (\mathbf{w}^{34})^T\mathbf{x} + b^{34}$

Multi-class Classification: One-against-one

- For a testing data, predicting all binary classifier

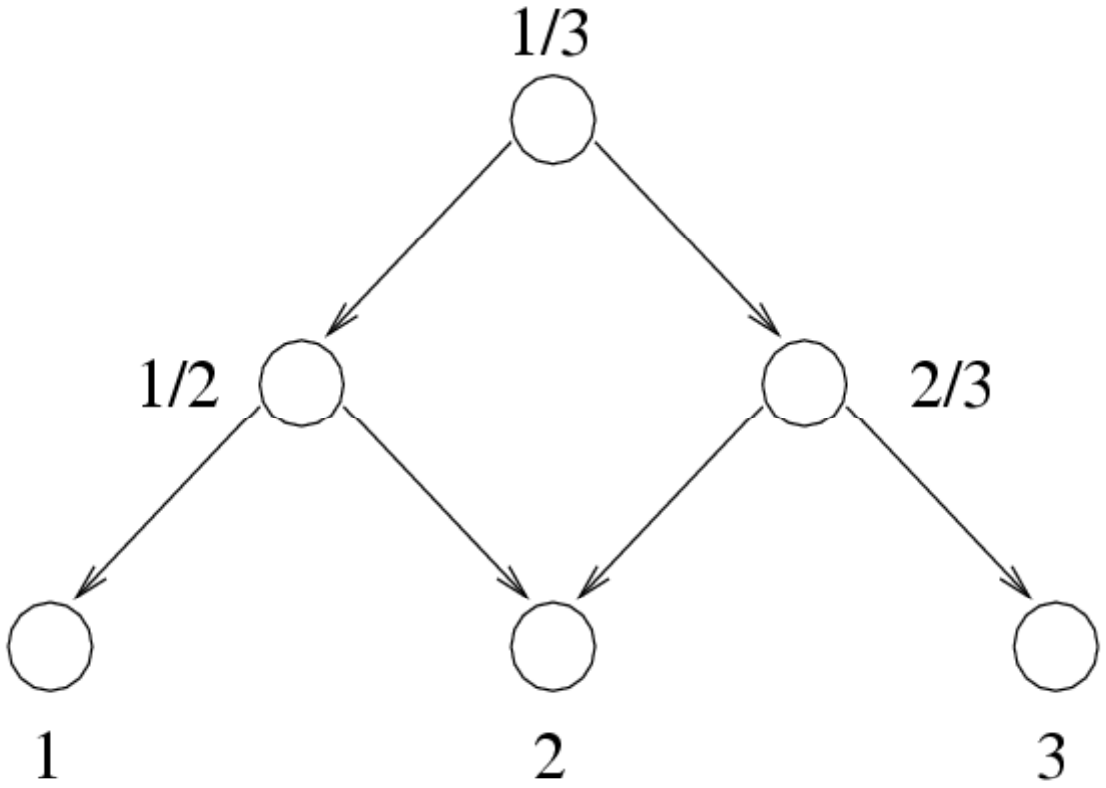
Classes		winner
1	2	1
1	3	1
1	4	1
2	3	2
2	4	4
3	4	3

- Select the one with the largest vote

class	1	2	3	4
# votes	3	1	1	1

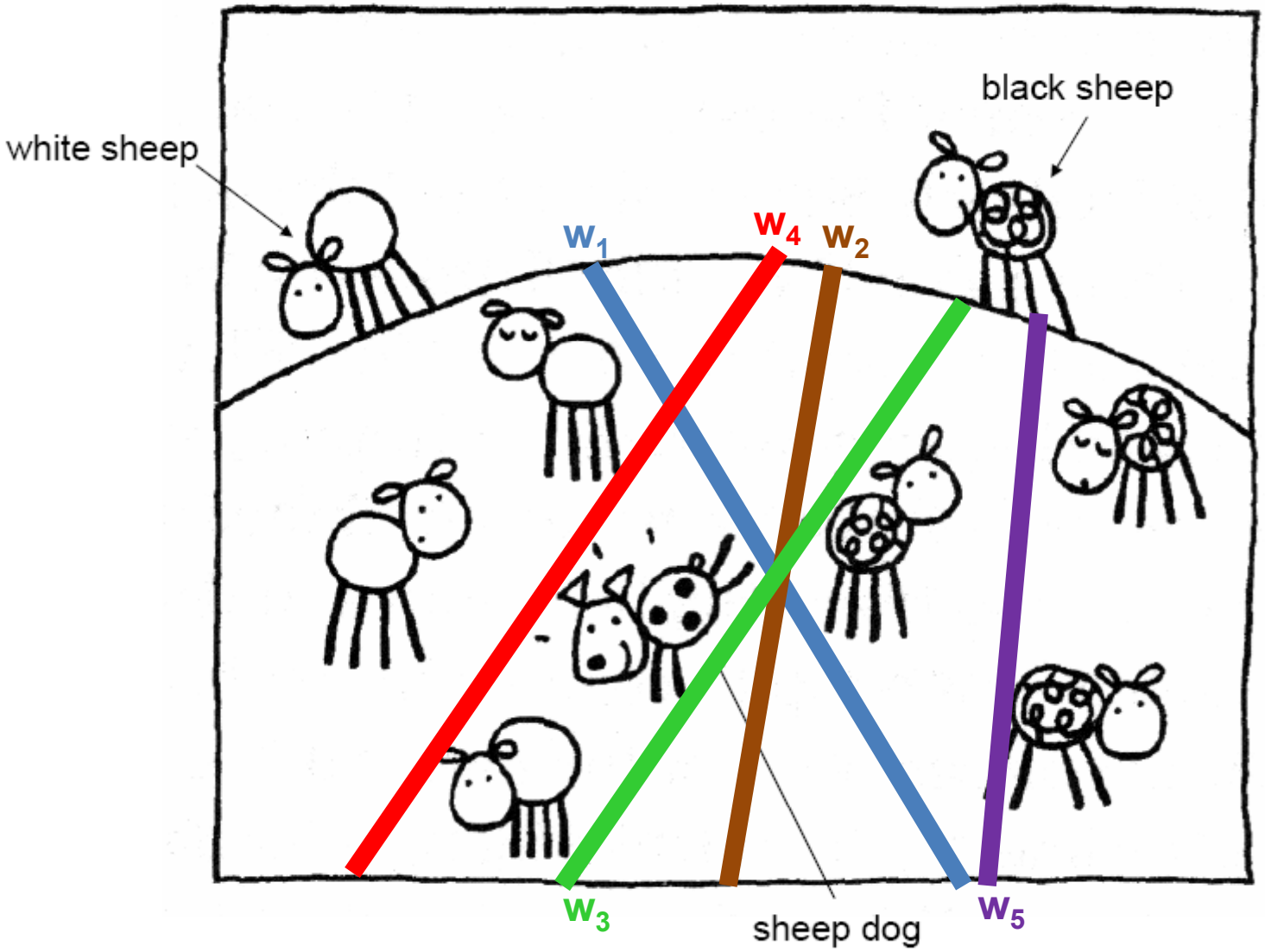
- May use decision values as well

Multi-class Classification: "Decision Tree"



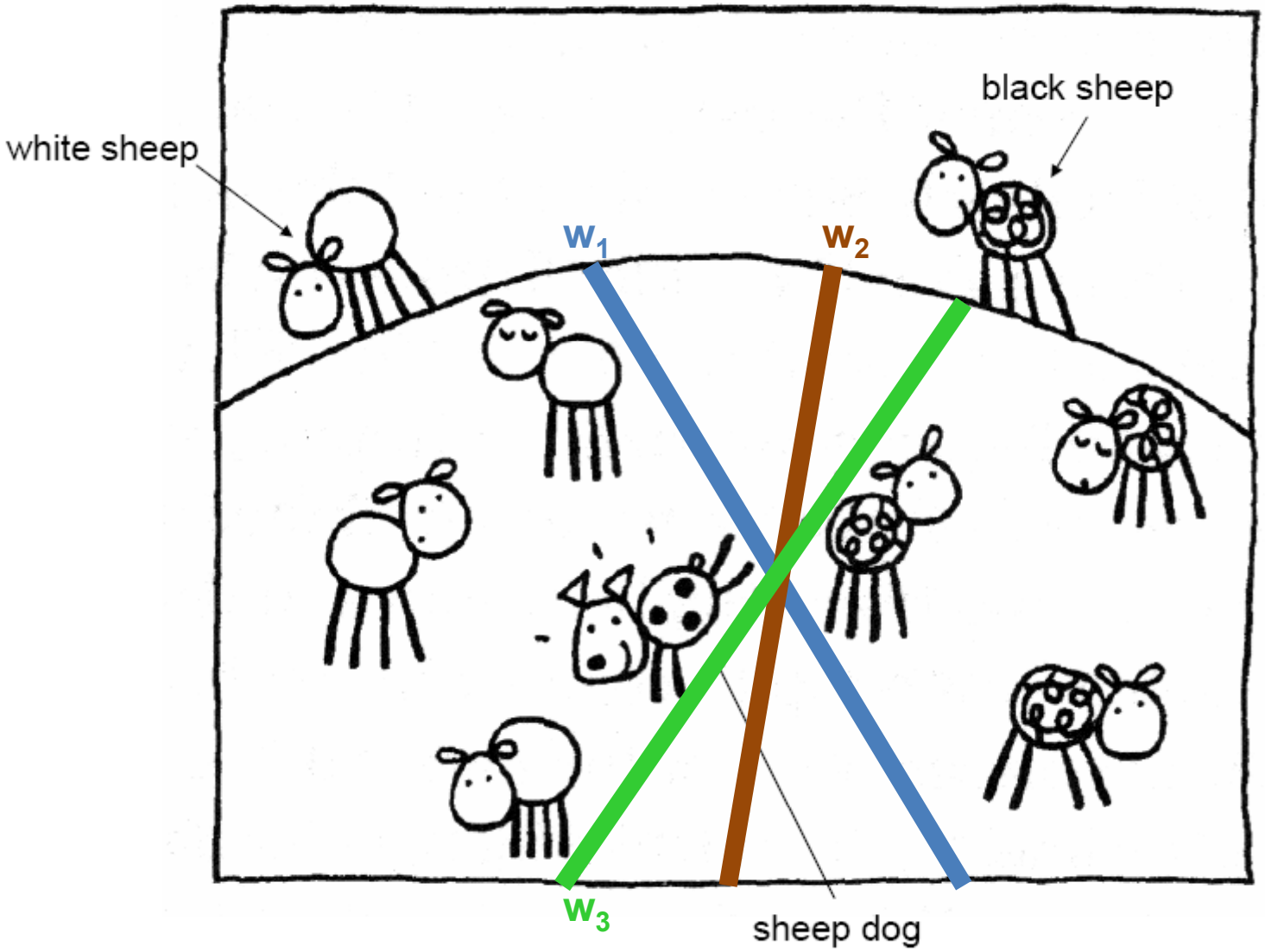
Back to the main two class classification problem...

How choose the Hyperplane?

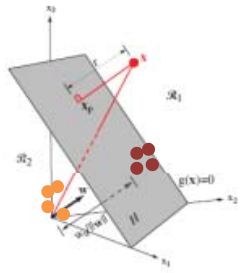


Back to the main two class classification problem...

How choose the Hyperplane?



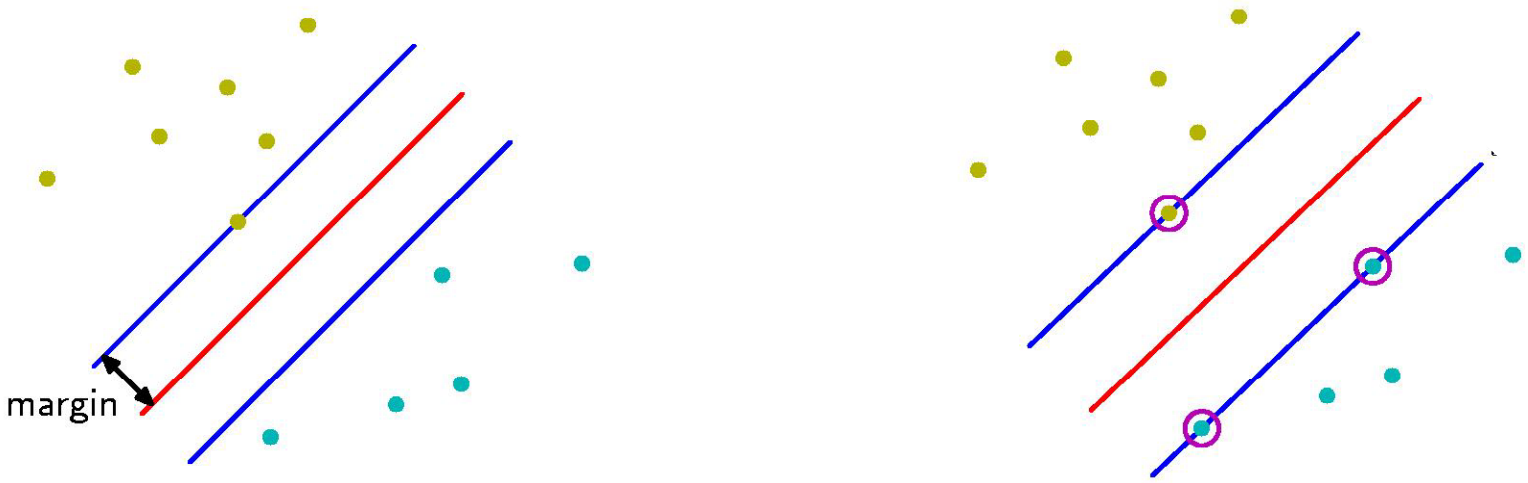
Maximal Margin Classifier



- There may exist many solutions that separate the classes exactly.
- We should try to find the one that will give the smallest generalization error (the expectation value of the loss function).
- **SVM approaches this problem through the concept of *margin*: *the smallest distance between the decision boundary and any of the samples.***
- In SVM the decision boundary is chosen to be the one for which the margin is maximized.
- The maximum margin solution can be motivated using *statistical learning theory (the theoretical bound on the generalization error is minimized)*.

$$R(\mathbf{w}) = \int Loss(\mathbf{w})\, p(\mathbf{x}, y) d\mathbf{x} dy \leq R_{emp}(\mathbf{w}) + VC_{confidence}$$

Maximal Margin Classifier trough SVM

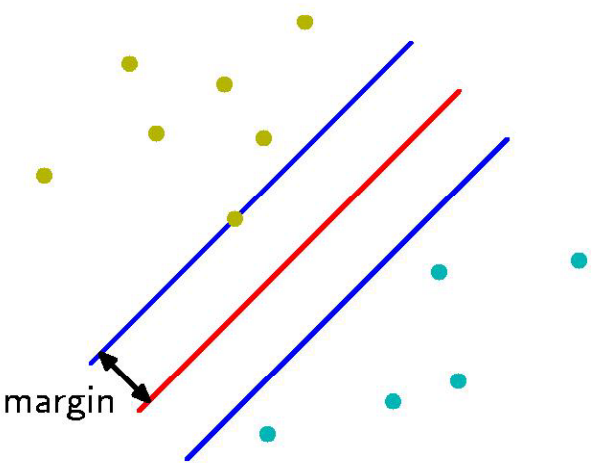


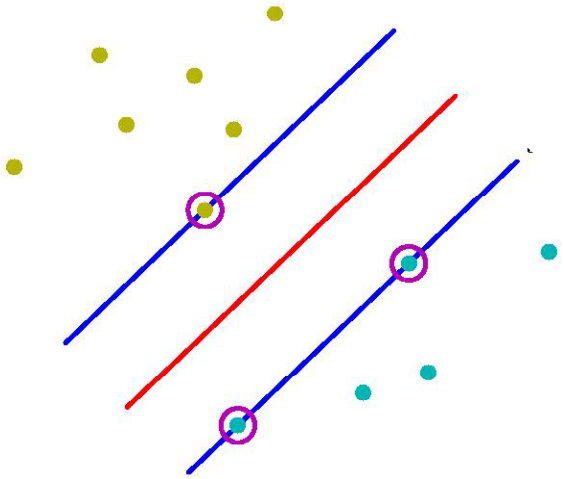
The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points (left figure).
Maximizing the margin leads to a particular choice of decision boundary (right figure).
The location of these boundary is determined by a subset of datapoints, known as *support vectors.*

We want the hyperplane such that:

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \; \min\{\| \mathbf{x} - \mathbf{x_i} \| : \; \mathbf{x} \in R^d, < \mathbf{w} \cdot \mathbf{x} > + b = 0, i = 1...n\}$$

Maximal Margin Classifier trough SVM



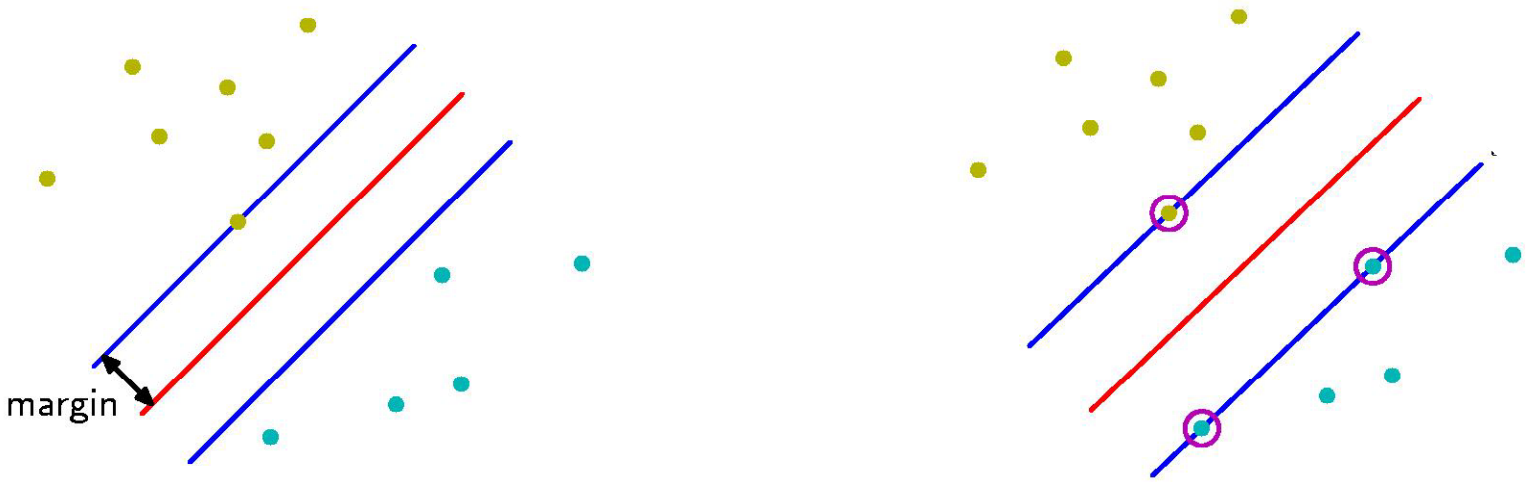


$$\underset{\mathbf{w},b}{\operatorname{argmax}} \; \min\{\| \mathbf{x} - \mathbf{x_i} \| : \; \mathbf{x} \in R^d, < \mathbf{w} \cdot \mathbf{x} > + b = 0, i = 1...n\}$$

Recall:
perpendicular distance of a point ***x*** *from an hyperplane defined by f(****x****)=0 is given by |f(x)|/||****w****||. The distance of a training point to the decision surface given by* $y_i f(x_i) / ||w|| = (y_i(<w,x_i>+b))/||w||$

The margin is given by the perpendicular distance to the closest point x_i from the training set and we wish to optimize the parameter w and b in order to maximize this distance.

Maximal Margin Classifier trough SVM



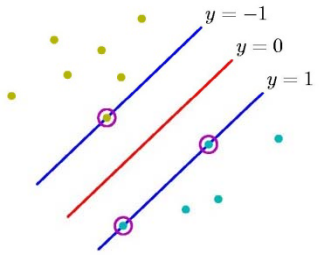
Thus the maximum margin solution is found by solving

$$\underset{\mathbf{w},b}{\operatorname{argmax}}\left\{\frac{1}{\|w\|}\min_{i}[y_n(<w\cdot x>+b)]\right\}$$

Constraints: We are only interested in solutions for which all data points of the training set are correctly classified so that $y_i f(x_i)>0$, i=1...n

Maximal Margin Classifier trough SVM

- Direct solution of this optimization problem would be very complex→ convert into an equivalent problem that is much easier to solve.
- Observation: if we make the rescaling w→kw and b→kb then the distance from any point xi to the decision surface, given by $y_i f(x_i)/||w||$, is unchanged.
- Rescaling w and b such that the point(s) closest to the decision surface satisfy ***$y_i(<w,x_i>+b)=1$ we obtain the canonical form of the decision hyperplane. In this case all data points of the training set will satisfy $y_i(<w,x_i>+b)>=1$***



Margin= 1/||w||

The Maximal Margin optimization problem

- We want **w**, b such that

$$\frac{\| \mathbf{w} \|^2}{2} = \frac{1}{2} < \mathbf{w}, \mathbf{w} > \quad \textbf{is minimized}$$

- Subject to the constraints

$$y_i (< \mathbf{w}, \mathbf{x_i} > + b) \geq 1 \qquad i = 1 \ldots n$$

- ***Is a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality***

→ ***this constrained optimization problem is dealt with by introducing Lagrange multiplies α_i>=0***

Lagrangian Rule: for constraints of the form c_i >=0, the constraints equations are multiplied by α_i >=0 and subtracted from the objective function

The Maximal Margin optimization problem: primal form

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{i=1}^{n} \alpha_i (y_i (< \mathbf{w}, \mathbf{x_i} > + b) - 1)$$

The Lagrangian Lp(**w**,b,**α**) has to be minimized with respect to the primal variables **w** and b and maximized with respect to the dual variables α_i.

1) We must now minimize Lp with respect to **w** and b and simultaneously require that the derivatives of Lp with respect to the all α_i vanish all subject to the constraint $\alpha_i >= 0$

2) We can equivalently solve the follows dual problem: maximize Lp subject to the constraints that the gradient of Lp with respect to **w** and b vanish and subject also to the constraints that $\alpha_i >= 0$

This particular dual formulation (called Wolfe dual) has the property that the maximum of Lp subject to constraints 2) occurs at the same values of the **w**, b and **α**, as the minimum Lp subject to contraints 1)

The Maximal Margin optimization problem: dual form

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{i=1}^{n} \alpha_i (y_i (< \mathbf{w}, \mathbf{x_i} > + b) - 1)$$

- Requiring that the gradient of Lp with respect w and b vanish give the conditions

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i} \qquad \sum_i \alpha_i y_i = 0 \qquad \alpha_i \geq 0 \qquad i = 1 \ldots n$$

- Since these are equality constraints in the dual formulation, we can substitute them into Lp to give:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < \mathbf{x_i}, \mathbf{x_j} >$$

$$k(x, x') := (\mathbf{x} \cdot \mathbf{x}') = (\Phi(x) \cdot \Phi(x'))$$

The Maximal Margin optimization problem: dual form

- The final SVM optimization problem will be

- Choose a Kernel function k(x,x')

- Maximize $L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j})$

Kernel induced Features Space

- Subject to the constraints

$$\sum_i \alpha_i y_i = 0 \qquad \alpha_i \geq 0 \qquad i = 1 \ldots n$$

The decision function

- We can use the parameters α* that solve the previous optimization problem to find the weight vector that realises the maximal margin hyperplane

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x_i} \qquad \mathbf{w}^* = \sum_i \alpha_i^* y_i \Phi(\mathbf{x_i})$$

The solution vector thus has an expansion in terms of a subset of training patterns, namely those patters whose α_i is non-zero, called Support Vector. The Support Vector lie in the margin.

$$D(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}(<\mathbf{w}^*, \mathbf{x}> + b >= \text{sgn}(\sum_{i=1}^{n} y_i \alpha_i <\mathbf{x}, \mathbf{x_i}> + b) = \text{sgn}(\sum_{s \in Support} y_s \alpha_s <\mathbf{x}, \mathbf{x_s}> + b)$$

$$D(\mathbf{x}) = \text{sgn}(\sum_{s \in Support} y_s \alpha_s k(\mathbf{x}, \mathbf{x_s}) + b)$$

Who is the bias b?

- The value of b does not appear in the dual problem
- Having solved the quadratic programming problem and found the values **α***, we can determine the value of b by noticy that any support $\mathbf{x_i}$ satisfy $y_i f(\mathbf{x_i})=1$

$$f(\mathbf{x_i}) = \sum_{s \in Support} y_s \alpha_s < \mathbf{x_i}, \mathbf{x_s} > + b$$

$$y_i \left(\sum_{s \in Support} y_s \alpha_s < \mathbf{x_i} \cdot \mathbf{x_s} > + b \right) = 1$$

$$\sum_{s \in Support} y_s \alpha_s < \mathbf{x_i}, \mathbf{x_s} > + b = y_i$$

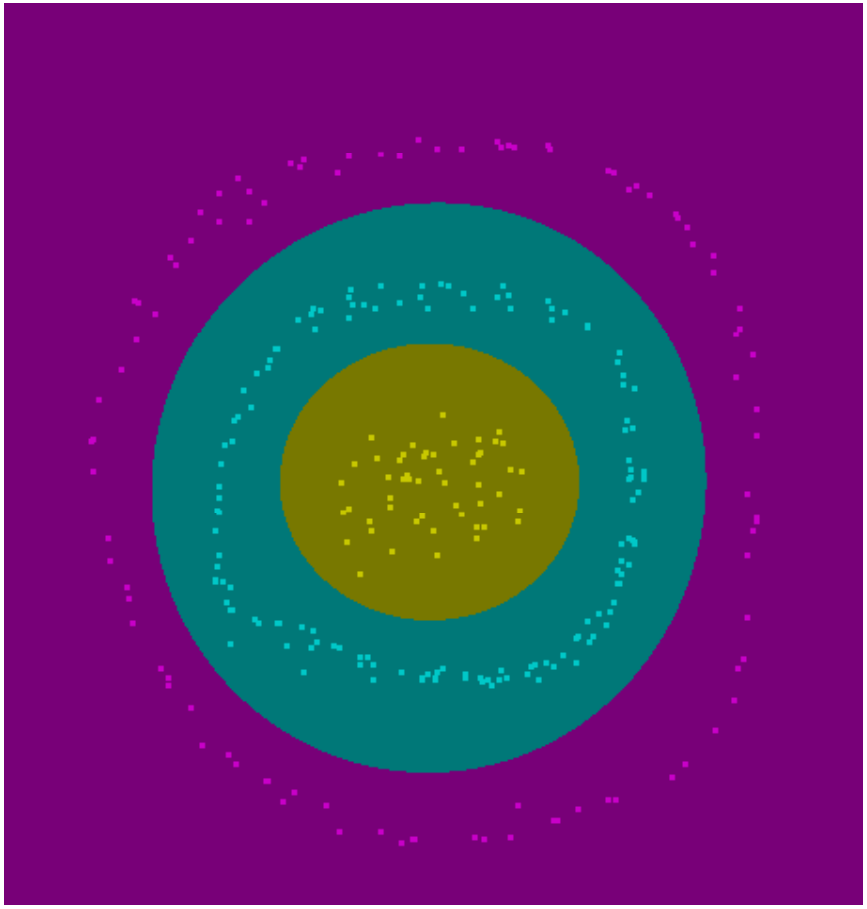
$$b = y_i - \sum_{s \in Support} y_s \alpha_s < \mathbf{x_i} \cdot \mathbf{x_s} >$$

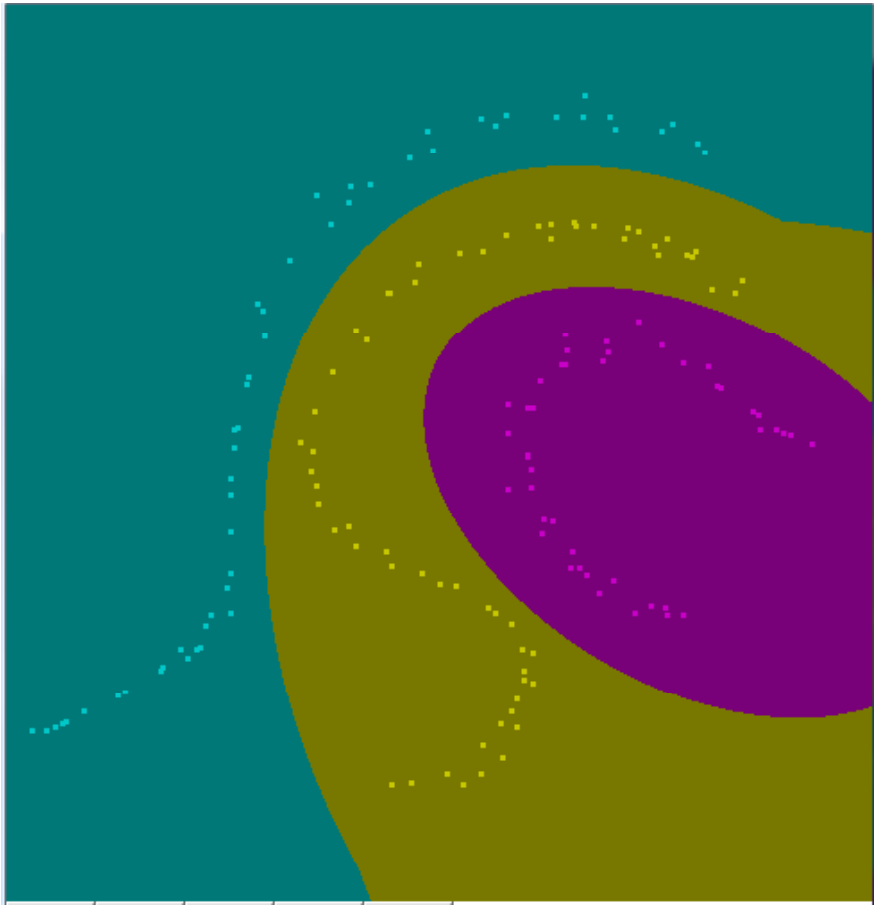
A numerically more stable solution is:

$$b = \frac{1}{|S|} \sum_{i \in S} \left(y_i - \sum_{s \in S} y_s \alpha_s < \mathbf{x_i}, \mathbf{x_s} > \right)$$

$$b = \frac{1}{|S|} \sum_{i \in S} \left(y_i - \sum_{s \in S} y_s \alpha_s k(\mathbf{x_i}, \mathbf{x_s}) \right)$$

SVM Toy Examples (LIBSVM)





Important Observations

Maximize: $$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

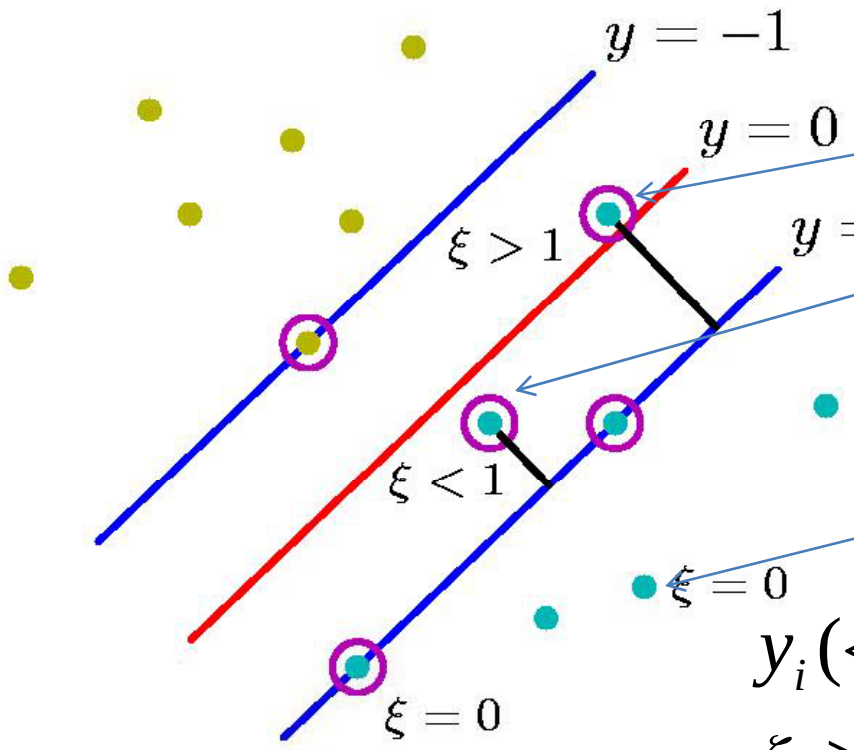
Subject to: $$\alpha_i \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

- Is a Quadratic Optimization problem: convex, no local minima
- Solvable in polynomial time

Soft Margin: slack variables

- Relax constraints: allowing the margin constraints to be violated → allow some of the training data points to be misclassified



misclassified

Lie inside the margin, but on the correct side of the decision boundary

Correctly classified

$$y_i(<\mathbf{w}, \mathbf{x_i}> + b) \geq 1 - \xi_i \qquad i = 1...n$$

$$\xi_i \geq \mathbf{0} \qquad i = 1...n$$

Soft Margin optimization problem

- We want **w**, b such that

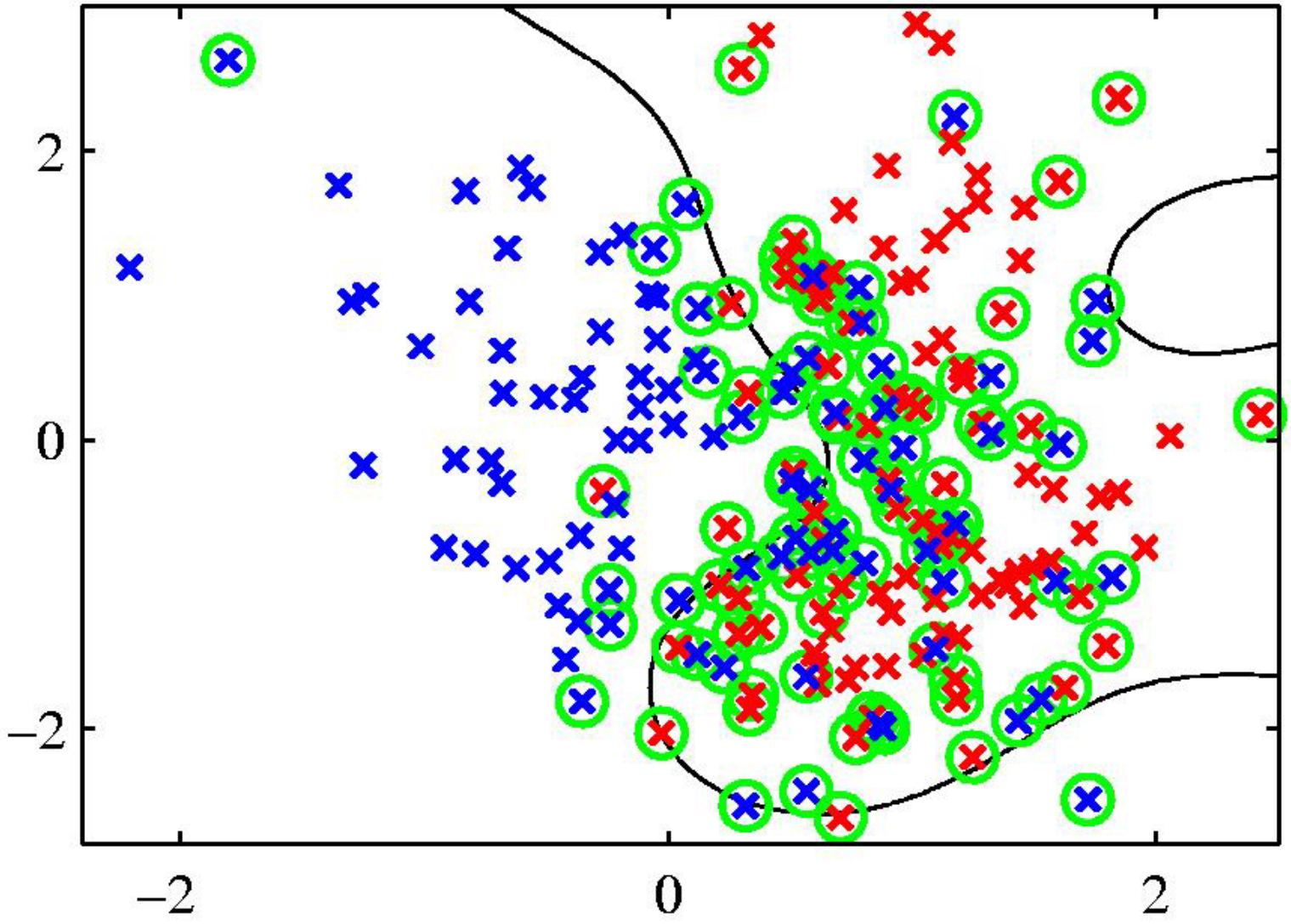
$$\frac{1}{2} < \mathbf{w}, \mathbf{w} > + C \sum_i \xi_i \quad \textbf{is minimized}$$

- Subject to the constraints

$$y_i (< \mathbf{w}, \mathbf{x_i} > + b) \geq 1 - \xi_\mathbf{i} \qquad i = 1 \ldots n$$

- Observation:
  - Solution can be obtained through Dual form
  - Kernels can be used

SVM Toy Example



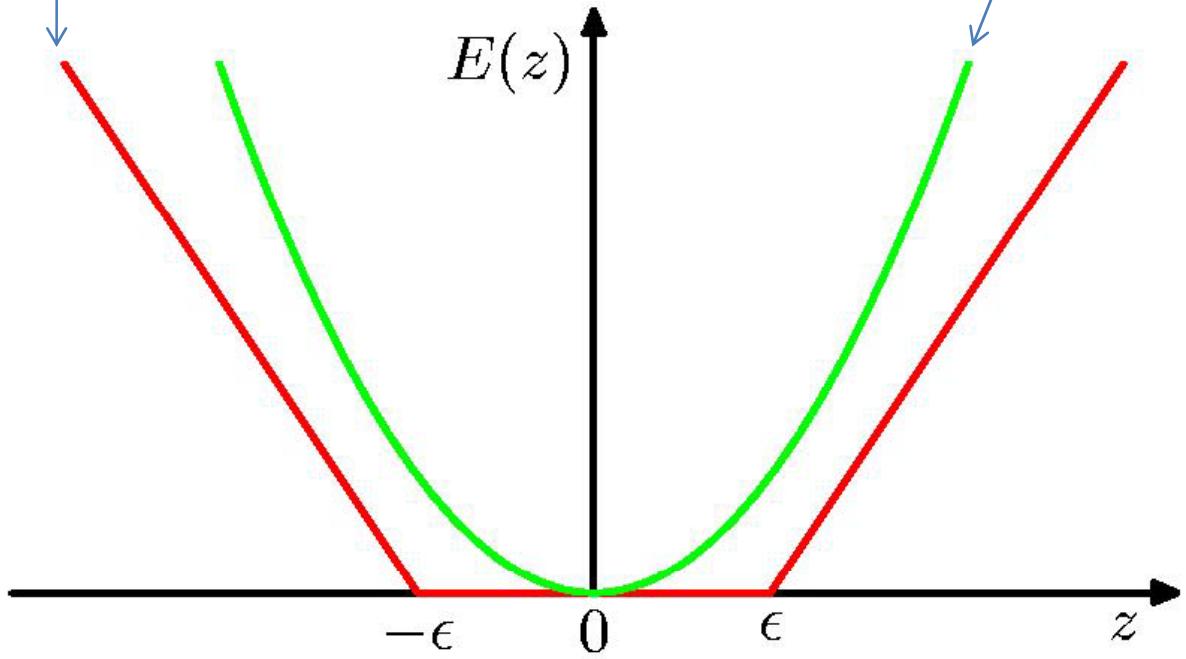
SVM for Regression: error function

ε-intensive error function

$$E_{\varepsilon}(\mathbf{x}, y) = \begin{cases} 0 & \text{if } |f(\mathbf{x}) - y| < \varepsilon \\ |f(\mathbf{x}) - y| - \varepsilon & \text{otherwise} \end{cases}$$

Linear Regression error function

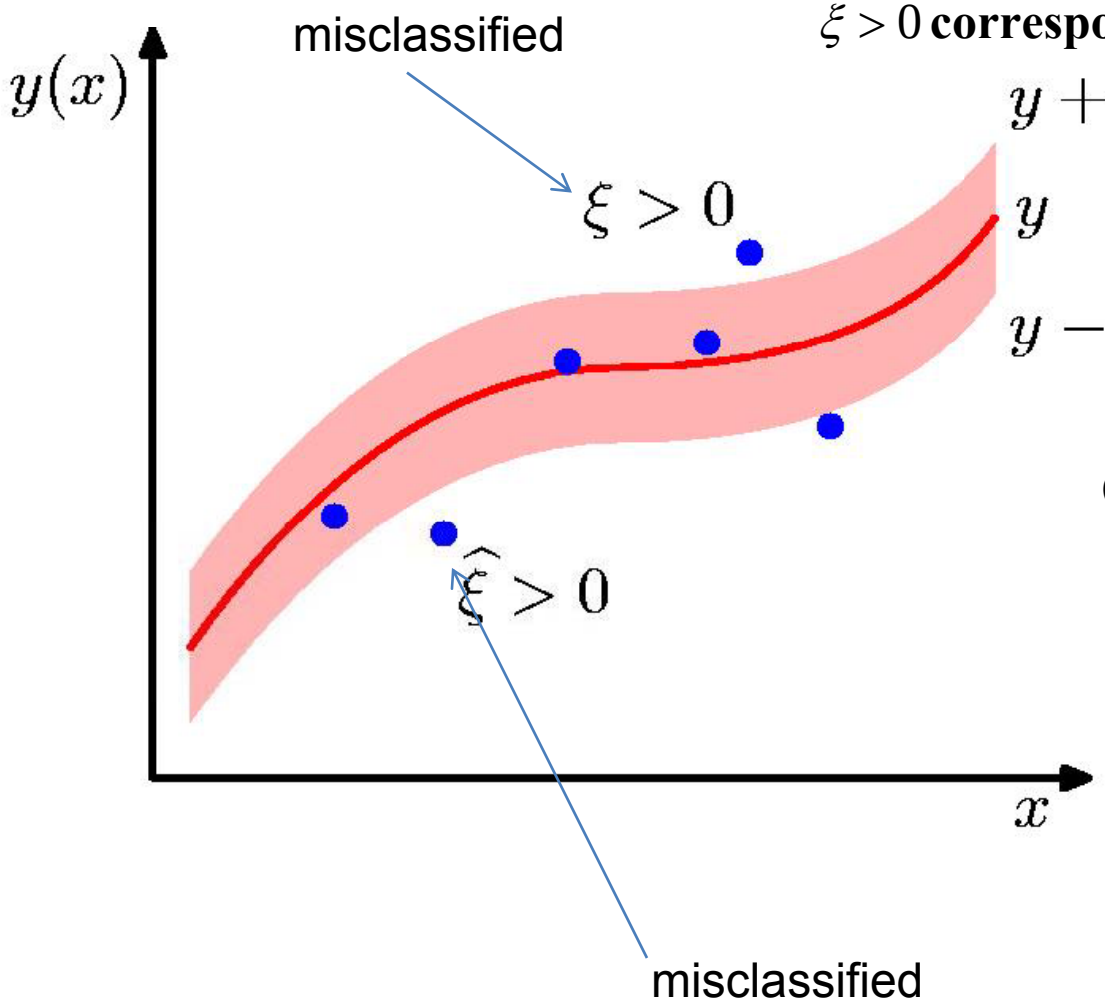
$$\frac{\lambda}{2} < \mathbf{w}, \mathbf{w} > + C \sum_i E(z_i)$$



SVM for Regression: ε-tube

$\xi > 0$ **corresponds to a point for which** $y > f(\mathbf{x}) + \varepsilon$

$\hat{\xi} > 0$ **corresponds to a point for which** $y < f(\mathbf{x}) - \varepsilon$



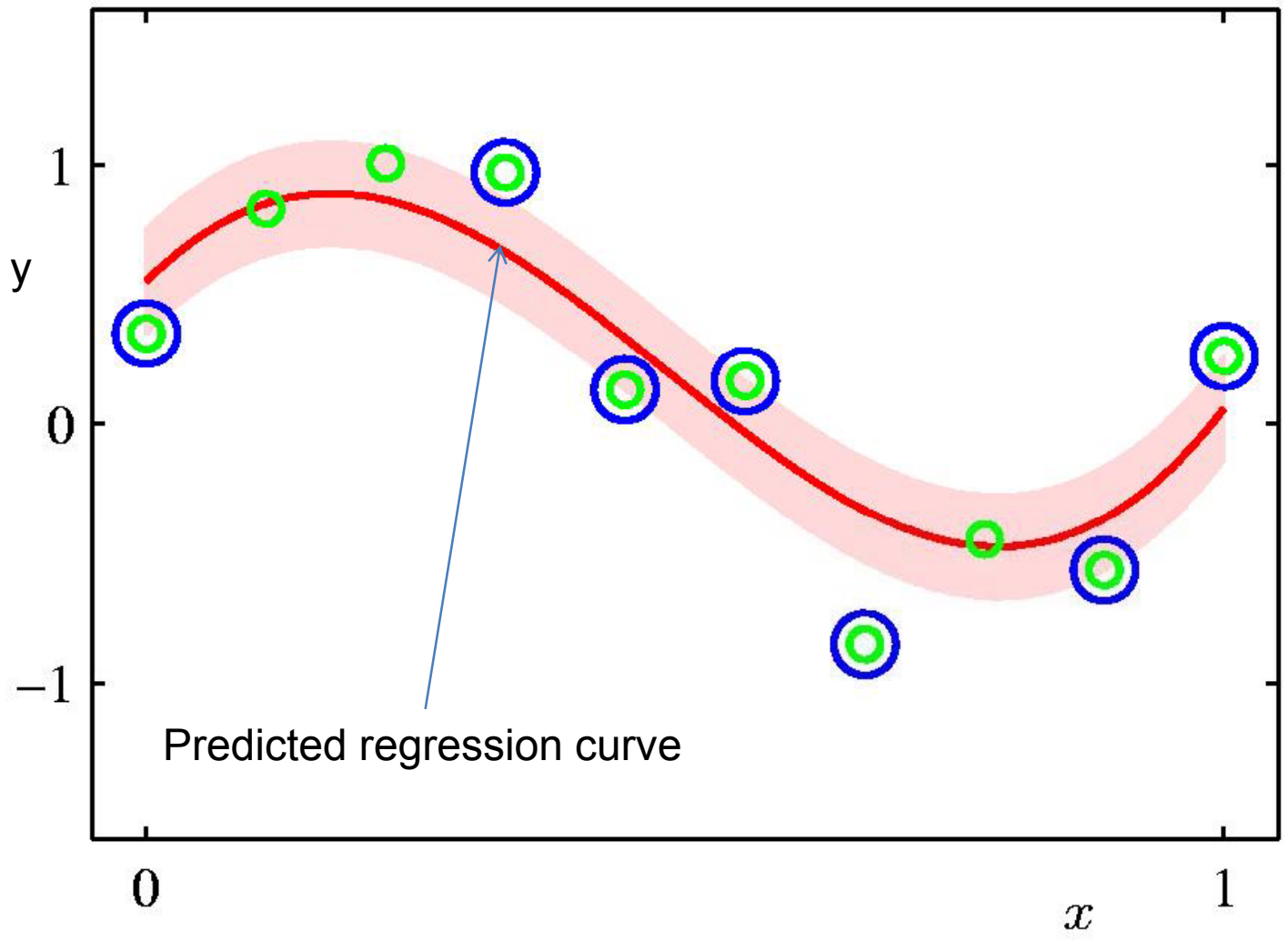
Corresponding constrains:

$$f(\mathbf{x}) - \varepsilon - \hat{\xi} \leq y \leq f(\mathbf{x}) + \varepsilon + \xi$$

Objective function for SVM:

$$\frac{1}{2} < \mathbf{w}, \mathbf{w} > + C \sum_i (\xi + \hat{\xi})$$

SVM for Regression: ε-tube



Learning Support Vector Machine

- The Naive solution is a simple on-line algorithm for *Gradient Ascent*: evaluating the gradient for just one patter at a time, and hence updating a single component α_i^t by the increment $\eta \frac{\partial W(\boldsymbol{\alpha}^t)}{\alpha_i}$

$$\alpha_i^{t+1} = \alpha_i^t + \frac{1}{k(\mathbf{x_i}, \mathbf{x_j})}\left(1 - y_i \sum_j \alpha_j^t y_j k(\mathbf{x_i}, \mathbf{x_j})\right)$$

References

- Papers:
  - C. J. C. Burges, A tutorial on Support Vector Machines for Pattern Recognition, Data mining and Knowledge Discovery, 2, 121-167, 1998
  - Bernhard Schölkopf, Statistical Learning and Kernel Methods, Technical Report MSR-TR-2000-23
- Books:
  - N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machine and other kernel-based methods, Cambridge Press, 2000
  - John Shawe-Taylor, Nello Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004
  - Bernhard Schölkopf, Alex Smola, Learning With kernel, MIT Press, Cambridge, MA, 2002
  - A. Papoulis, Probability and Statistics, Prentice Hall, 1990
  - Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006
  - Duda, Hart, Stork, Pattern Classification, Wiley Interscience, 2001
- Tutorials
  - Chih-Jen Lin ,Suport Vector Machine, Talk at Machine Learning summer School 2006
  - Nello Cristianini, Support Vector Machine and Kernel Methods, ICML-2001
  - Colin Campbell, Support Vector Machine and Kernel Methods, The analysis of Patterns 2006
- Slides of Courses
  - Zoubin Ghahramani, Machine Learning Course (4F13), Department of Engineering, University of Cambridge, 2006