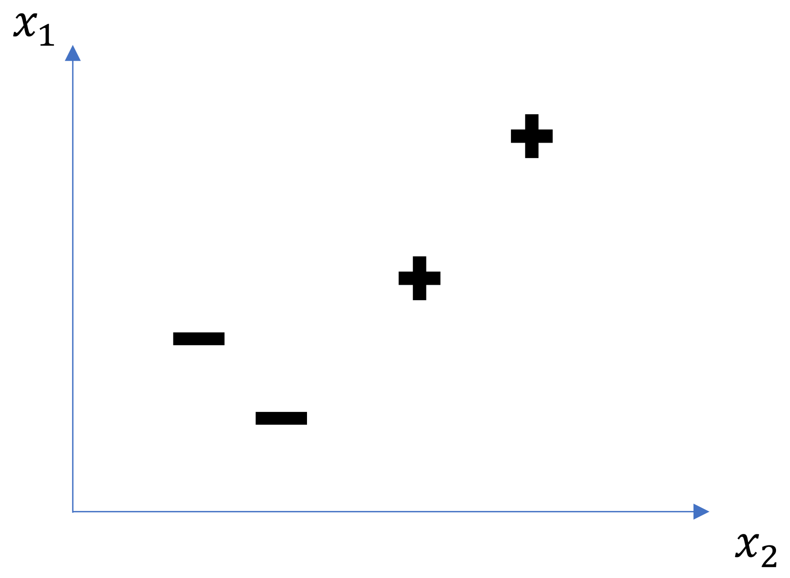
Support Vector Machines

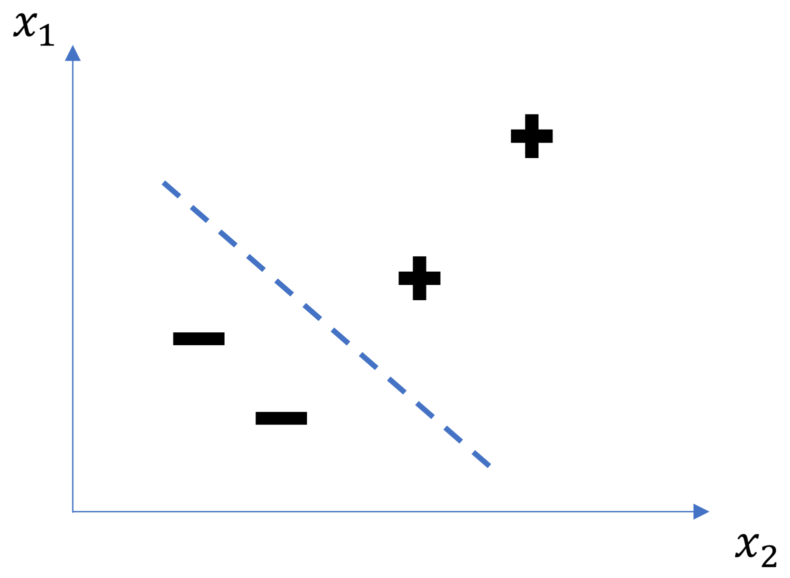
Large Margin Classifier

SVM Intuition



Let x_1, and x_2 be two features in a data space. Let **+** and **–** be samples belonging respectively to positive and negative classes.

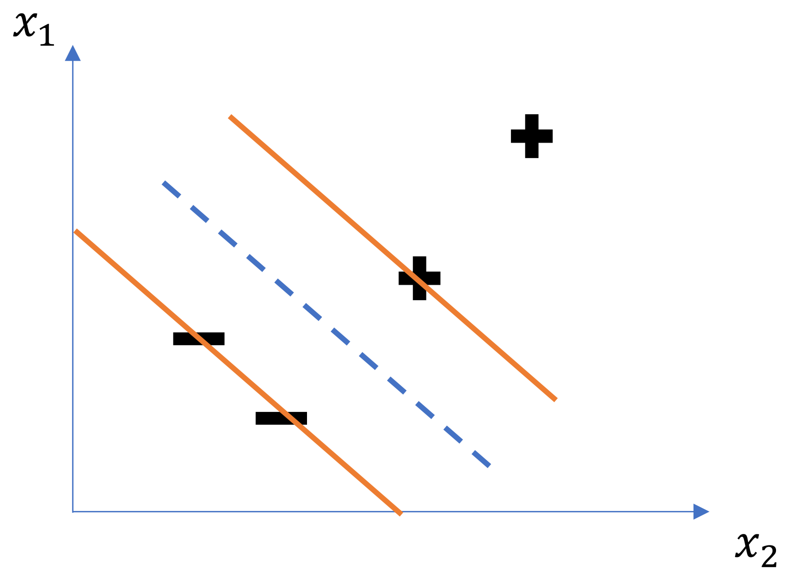
SVM Intuition



Let $\vec{\omega} \cdot \vec{x} + b$ be a separating hyperplane that we want to be «optimal».

<u>Observation</u>: This is equivalent to $\theta_0 + \theta_1 x_1 + \theta_2 x_2$ we already know

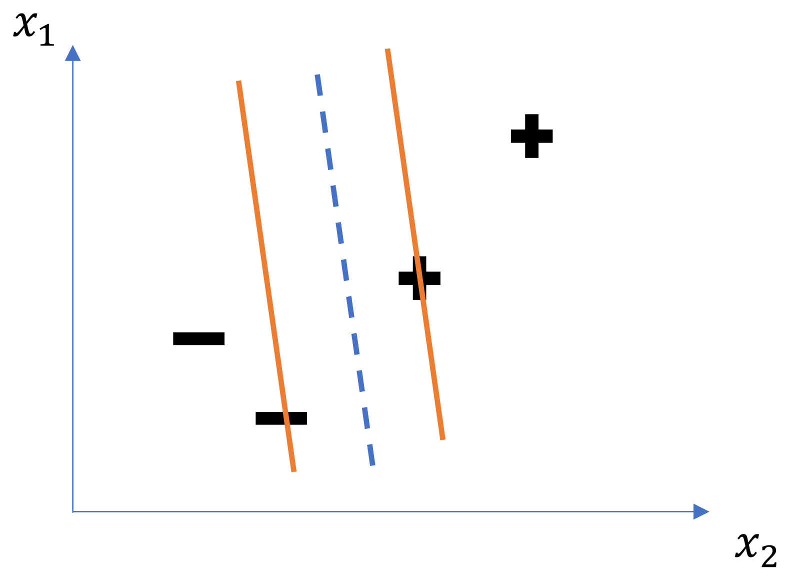
SVM Intuition



the optimal street is as wide as possible, we try to find the hyperplane with the largest separation between the classes

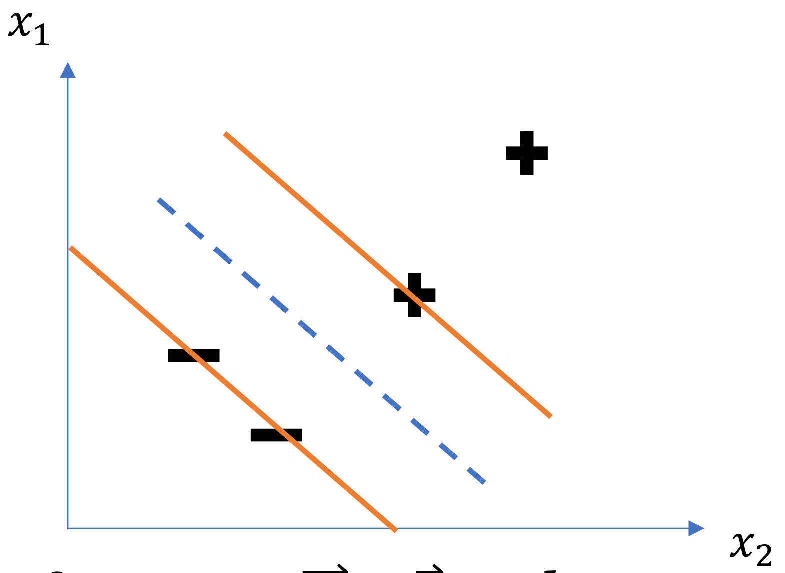
We assume that «optimal» means that we could draw a «street» around the hyperplane **as wide as possible.**

SVM Intuition



Please note that there are infinite feasible hyperplanes. However, we are looking for the only one with the **maximum width**.

Decision rule

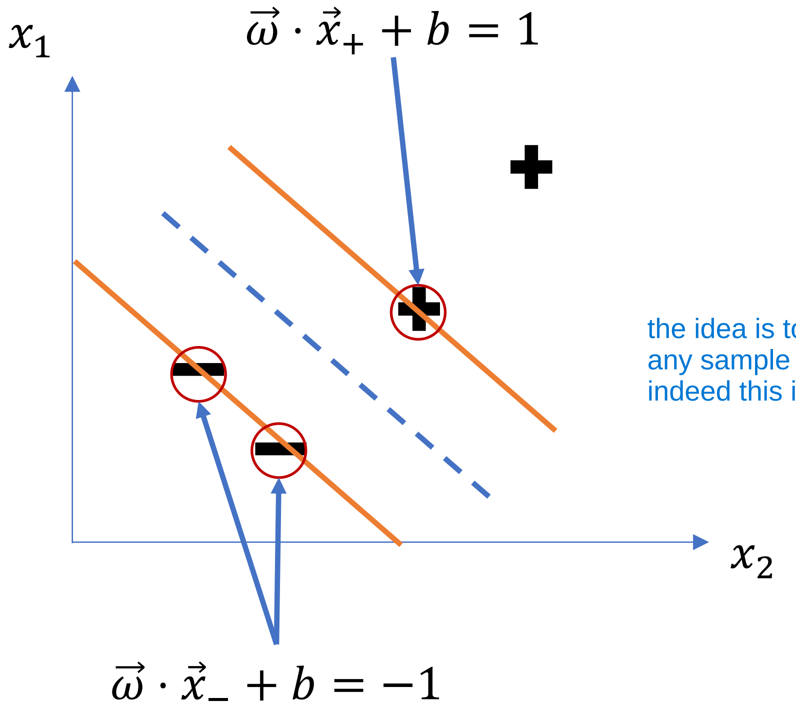


$$\boxed{\begin{array}{r}\text{IF } \vec{\omega}\cdot\vec{u}+b \geq 0 \text{ THEN } + \\ \text{DECISION RULE}\end{array}}$$

1

If we set $\vec{\omega} \cdot \vec{x} + b = 0$ we will obtain all the points on the hyperplane. However, if for a certain sample x, $\vec{\omega} \cdot \vec{x} + b > 0$, x will be on the «right side» of the hyperplane, together with positive samples.

Decision rule (cont.d)



x_1
$\vec{\omega} \cdot \vec{x}_+ + b = 1$
x_2
$\vec{\omega} \cdot \vec{x}_- + b = -1$

the idea is to not have
any sample inside the road.
indeed this is the support vector

$$\vec{\omega} \cdot \vec{x_+} + b \geq 1 \qquad y_i \cdot (\vec{\omega} \cdot \vec{x_+} + b) \geq 1$$

$$\vec{\omega} \cdot \vec{x_-} + b \leq -1 \qquad y_i \cdot (\vec{\omega} \cdot \vec{x_-} + b) \leq 1$$

greater not
less

$$y_i \text{ such that } y_i = +1 \text{ for + samples}$$

$$y_i = -1 \text{ for - samples}$$

$$y_i \cdot (\vec{\omega} \cdot \vec{x_i} + b) - 1 \geq 0$$

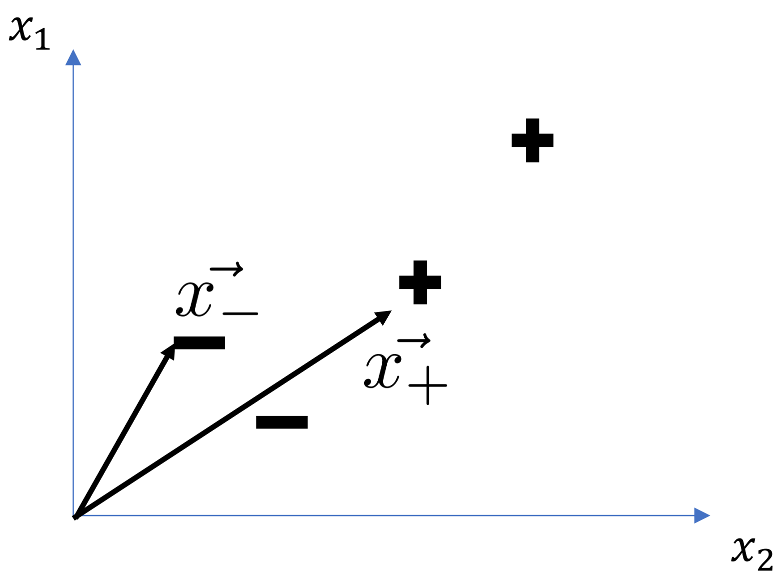
$$\boxed{\begin{array}{l} y_i \cdot (\vec{\omega} \cdot \vec{x_i} + b) = 1 \\ for\ x_i\ on\ the\ borders \end{array}}$$

2

Problem definition

In a real scenario, the optimal hyperplane is unknown.

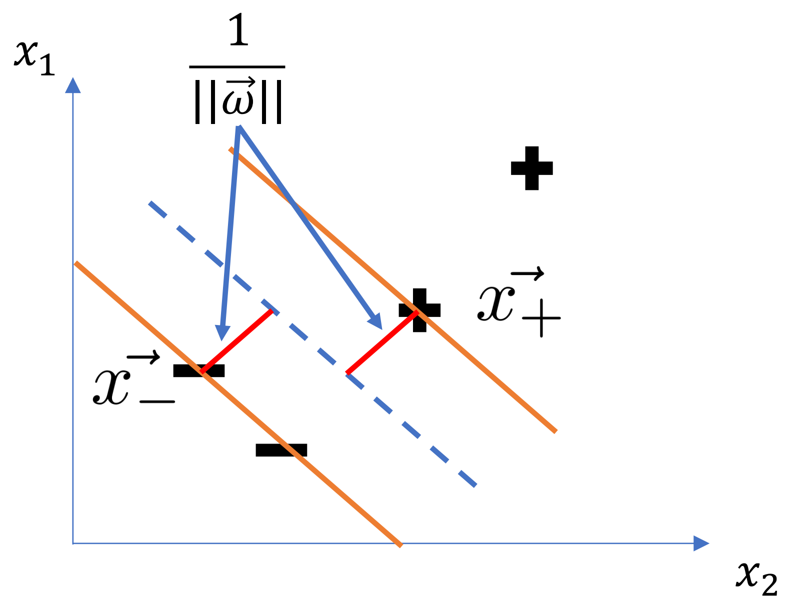
We want to figure out how to set the line such as the «street» between the positive and negative examples is as wide as possible



Politecnico di Bari

Problem definition (cont.d)

We don't know the width of the street but we can compute the difference



$$y_i(\vec{\omega} \cdot \vec{x_i} + b) = 1$$

$$y_i \text{ such that } y_i = +1 \text{ for } + \text{ samples}$$

$$y_i = -1 \text{ for - samples}$$

We know that the distance of a point x_i from a hyperplane is

$$\frac{|(\vec{\omega} \cdot \vec{x_i}) + b|}{||\vec{\omega}||}$$

$$\text{width} = \frac{2}{\|\omega\|}$$

this is the quantity we want to maximize

Problem definition (cont.d)

What we want is to maximize the width of the «street».

In order to do this we can perform a couple of mathematically convenient steps.

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

$$\max \frac{1}{\|\omega\|}$$

$$\min \|\omega\|$$

$$\boxed{\min \frac{1}{2}\|\omega\|^2}$$

3

Politecnico di Bari

Problem definition (cont.d)

$$minimize \ \frac{1}{2}\|\omega\|^2$$

$$s.t. \ \forall_i \ y^{(i)} \cdot \left(\omega^T \cdot x^{(i)} + b\right) \geq 1$$

for every point this constraint is to be respected

not only an optimization problem but also an constraint one

Constrained optimization (digression)

Lagrange duality is widely used to solve constrained optimization problems.

Let us define a generic problem with 1 equality constraint:

$$\min f(\omega)$$
$$s.t.\ h(\omega) = 0$$

Where $f(\omega)$, and $h(\omega)$ are convex functions. Now we introduce the Lagrangian function $L(\omega, \alpha)$:

$$L(\omega, \alpha) = f(\omega) + \alpha \cdot h(\omega)$$

Where alphas are named Langrangian multipliers.

Constrained optimization (digression)

It can be proved that for the considered problem, there exists a value ω^* which is a global minimum for $f(\omega)$ subject to the following condition:

$$\exists \alpha^* \; such \; that \qquad \frac{\partial L}{\partial \omega}(\omega^*, \alpha^*) = 0$$

minimizing the lagrangian function is equal of minimizing the original problem

$$\frac{\partial L}{\partial \alpha}(\omega^*, \alpha^*) = 0$$

Constrained optimization (digression)

- In case we have an equality constraint $h(\omega)$ and an inequality constraint $g(\omega)$ (e.g. $\geq$) the corresponding Lagrangian is defined as

$$L(\omega, \alpha, \beta) = f(\omega) + \alpha \cdot g(\omega) + \beta \cdot h(\omega)$$

- Corresponding to the dual problem

$$\text{maximize } L(\omega, \alpha, \beta)$$
$$\text{s.t. } \alpha \geq 0$$

Primal form of the optimization problem

In our case, we have

$$L = \frac{1}{2}\|\vec{\omega}\|^2 - \sum \alpha_i \left[y_i(\vec{\omega} \cdot \vec{x_i} + b) - 1\right]$$

With $\alpha_i \geq 0$

Primal problem

$$\frac{\partial L}{\partial \vec{\omega}} = \vec{\omega} - \sum \alpha_i y_i \vec{x_i} = 0$$

$$\boxed{\vec{\omega} = \sum_i \alpha_i y_i \vec{x_i}} \quad 4a$$

This is an excellent result because the vector ω is a linear sum of the samples.

BUT NOT ALL the samples, because some alphas could be zero.

Primal problem (cont.d)

Now we have to derive with respect to b.

$$L = \frac{1}{2}\|\vec{\omega}\|^2 - \sum \alpha_i \left[y_i(\vec{\omega} \cdot \vec{x_i} + b) - 1 \right]$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0$$

$$\boxed{\sum_i \alpha_i y_i = 0}$$

4b

Primal problem (cont.d)

If we derive with respect to alpha.

$$L = \frac{1}{2}\|\vec{\omega}\|^2 - \sum \alpha_i \left[y_i(\vec{\omega} \cdot \vec{x_i} + b) - 1\right]$$

$$\frac{\partial L}{\partial \alpha} = \boxed{\left[y_i(\vec{\omega} \cdot \vec{x_i} + b) - 1\right] = 0} \quad \text{4c}$$

Only the support vetors contribute to the computation of the solution

Dual problem (cont.d)

This is a quadratic optimization problem.

However we can perform some more steps:

$$L = \frac{1}{2}\|\vec{\omega}\|^2 - \sum \alpha_i \left[y_i(\vec{\omega} \cdot \vec{x_i} + b) - 1 \right]$$

- We can substitute ω in L $\boxed{\vec{\omega} = \sum_i \alpha_i y_i \vec{x_i}}$ $\boxed{\sum_i \alpha_i y_i = 0}$ 4

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x_i}\right) \cdot \left(\sum_j \alpha_j y_j \vec{x_j}\right) - \left(\sum \alpha_i y_i \vec{x_i}\right) \cdot \left(\sum_j \alpha_j y_j \vec{x_j}\right) - \sum_i \alpha_i y_i b + \sum_i \alpha_i$$

Dual problem (cont.d)

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x_i}\right)\cdot\left(\sum_j \alpha_j y_j \vec{x_j}\right) - \left(\sum_i \alpha_i y_i \vec{x_i}\right)\cdot\left(\sum_j \alpha_j y_j \vec{x_j}\right) \underbrace{- \sum_i \alpha_i y_i b}_{0} + \sum_i \alpha_i$$

$$L = \boxed{\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \boxed{\vec{x_i}\cdot\vec{x_j}}} \qquad \textcircled{5}$$

The optimization only depends on pairs of samples.

Dual problem (cont.d)

- The dual problem is then

Maximize $L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j}$

s.t. $\alpha_i \geq 0$

$$\sum_i \alpha_i y_i = 0$$

the goal is to understand the alpha

Dual problem (cont.d)

Now we have computed the optimal values of alpha and the separator hyperplane can be obtained:

$$\vec{\omega} \cdot \vec{x} + b = \sum \alpha_i \cdot y_i \cdot \overrightarrow{x_i} \cdot \vec{x} + b$$

Where the samples x_i considered are only support vectors

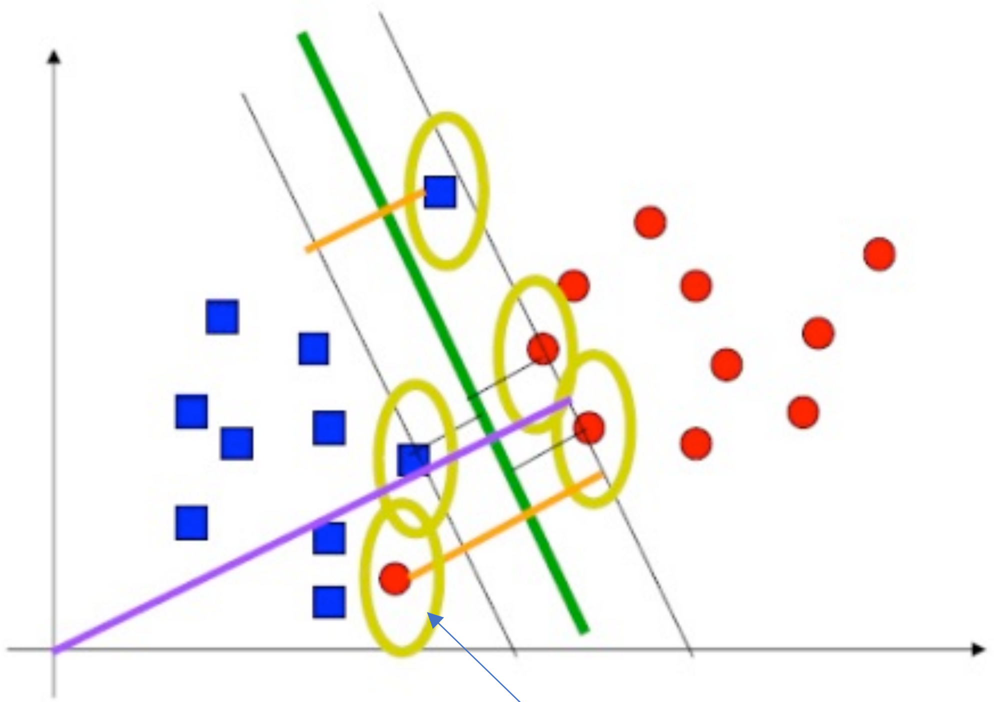
all the point that aren't in support vector has alpha=0

Politecnico di Bari

Support Vector Machines

Nonlinear separable data

SVM nonseparable data Intuition



Misclassification error

SVM nonseparable data

When data are non-linearly separable, we may get a separation between classes with a hyperplane only allowing that, after having defined the separating hyperplane, some patterns of the training set with positive label are classified as negative and viceversa. We must accept that some constraints are **violated**.

We introduce a **slack** variable ξi for each constraint, in order to allow an **error tolerance**:

$$y^{(i)}(\omega^T \cdot x^{(i)} + b) \geq 1 - \xi^{(i)}$$

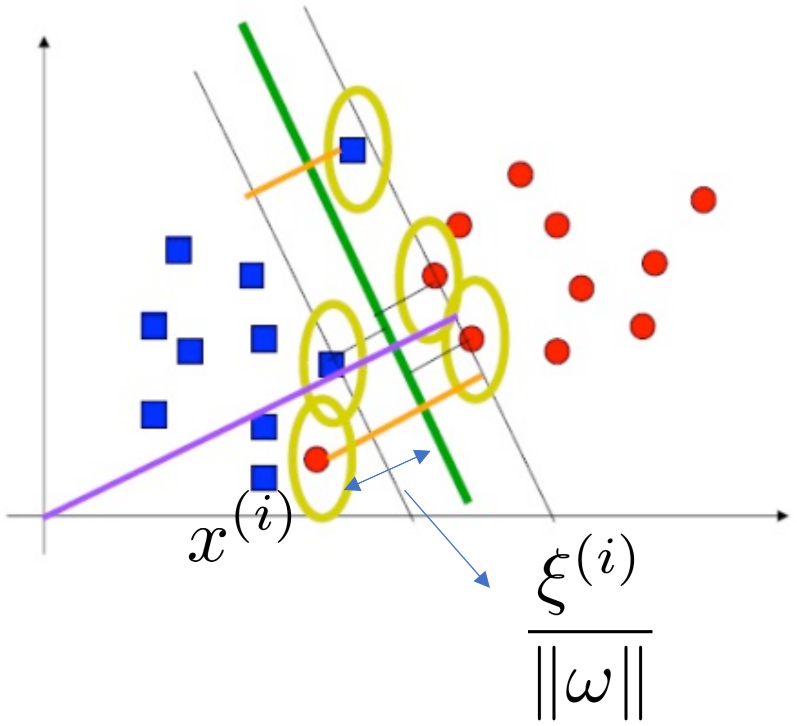
An additional term *C* is introduced in the cost function to **penalize misclassification** errors.

$$\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi^{(i)}$$

SVM nonseparable data (cont.d)

ξi are cost variables proportional to how far the misclassified pattern is from the hyperplane.

ξi >1 indicates a misclassification error



SVM nonseparable data (cont.d)

C (**regularization** parameter) lets to control the trade-off between hypothesis space complexity and the admissible number of errors.

A big value for C gives a stronger penalization to errors.

The optimization problem to solve becomes:

Minimize: $$\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi^{(i)}$$

s.t. $$y^{(i)}(\omega^T \cdot x^{(i)} + b) \geq 1 - \xi^{(i)}$$

$$\xi^{(i)} \geq 0$$

SVM nonseparable data (cont.d)

The dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Dual variables are now bounded with *C*.

The proposed solution could not be enough. It does not guarantee good performances since a hyperplane can only represent a dichotomy in the space of instances/patterns.

Cover’s theorem

Cover’s theorem on the separability of patterns

“A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space”

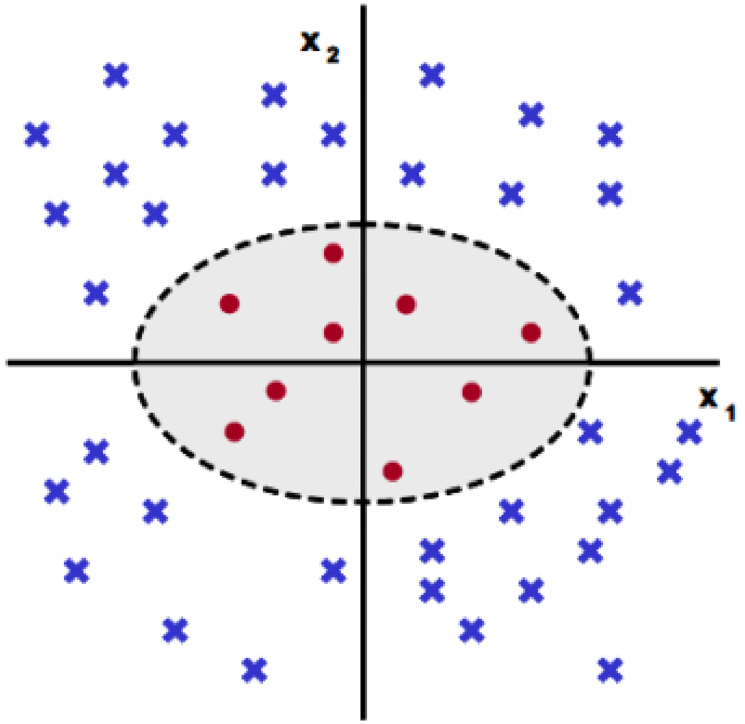
1. Patterns (input space) are mapped into a space with (much) higher dimension (feature space) through kernel functions;

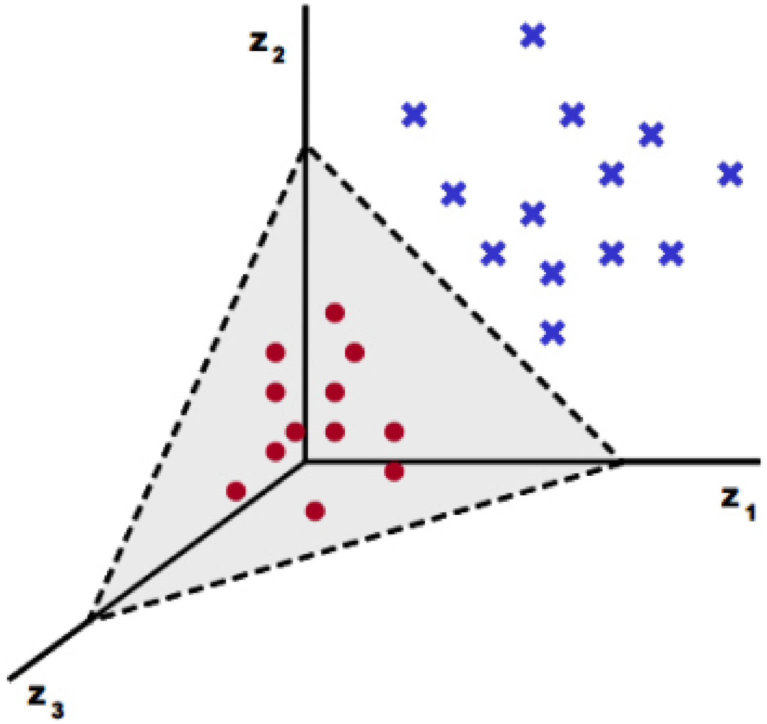
2. The optimal hyperplane is defined within this feature space.

Cover’s theorem

$$\varphi : R^2 \rightarrow R^3$$

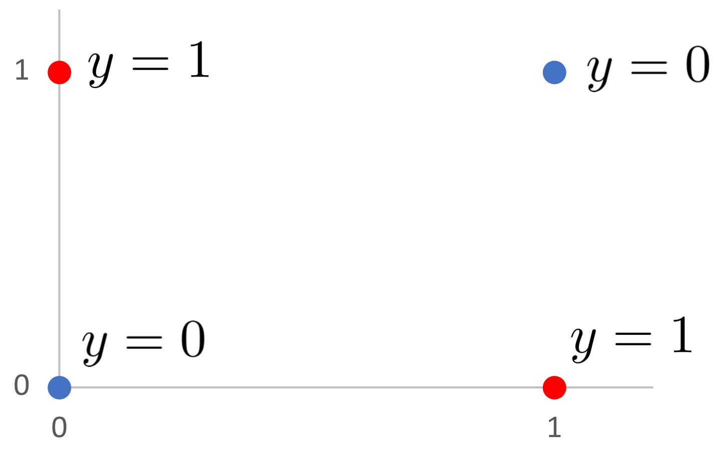
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

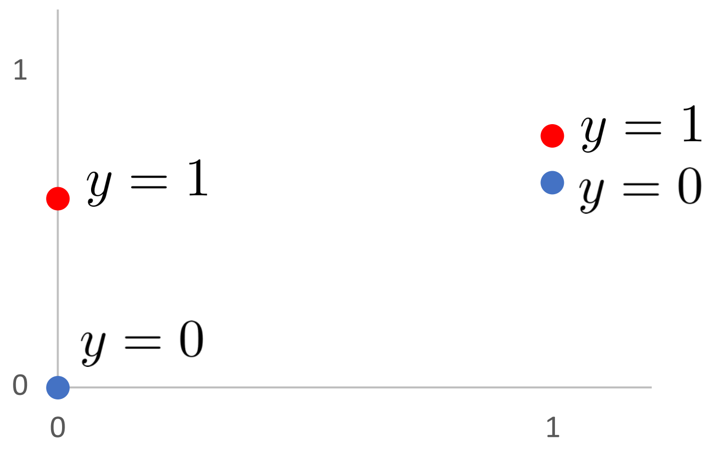




[Schölkopf, 2002 @; http://kernel-machines.org/]

The case of the non linearly separable space





In this case the samples are not linearly separable and we are not able to find a solution for our problem, but we can move to another space that is more convenient for our purposes:

$$\vec{x} \Rightarrow \Phi(\vec{x})$$

The case of the non linearly separable space

we care only of the product
of the XiXj and not the projection

$$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$$

This means that the optimization can be done using pairs of samples, and the decision can be computed again using a sample and an unknown vector.

This implies that ALL WE NEED is a function K that corresponds to the dot product between the samples in the new space.

This function K is called **kernel function**.

We have no need of the transformation function Phi!!

$$\xcancel{\Phi(\vec{x_i})}$$

Kernel trick

- Instead of increasing the complexity of the classifier (still remains a hyperplane) we increase the **dimension** of features space.
- BUT a higher dimensional space provokes serious computational issues because the learning algorithm must work with high dimensional vectors. Actually, the **projection** in a higher dimensional space is only **implicit** thanks to appropriate *Kernel* functions (***Kernel trick***).

To solve the optimization problem, the product $\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$ does not have to be explicitly computed in the feature space once we find a kernel function (specifically a positive definite kernel).

Kernel trick $$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$$

The kernel is a function that returns the (scalar) product of projections: it avoids you to explicitly compute the projection and make the product between the projected vectors. The explicit form of φ may be ignored.

Kernel trick (cont.d)

If a kernel function is defined, that is a function such that:

$$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j}) = \sum_{k=1}^{m} \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$$

The optimization problem becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Kernel trick (cont.d)

If a kernel function is defined, that is a function such that:

$$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j}) = \sum_{k=1}^{m} \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$$

The optimization problem becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

The projection $\Phi(\vec{x_i})$ into the feature space must not be explicitly computed.

I should compute the scalar product $\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$, but I don't have to, since I can **indirectly** obtain it with the kernel function

$$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$$

Kernel examples

Linear kernel

$$(\vec{u} \cdot \vec{v})$$

Polinomial kernel

$$(\vec{u} \cdot \vec{v} + 1)^d$$

Multi-Layer Perceptron tanh

$$\tanh(b(\vec{u} \cdot \vec{v}) - c)$$

Kernel examples (cont.d)

Radial basis function (RBF) kernel

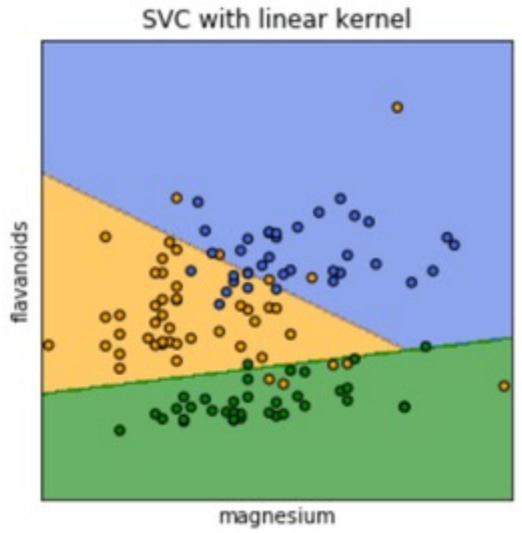
$$e^{-\frac{\|x_i - x_j\|}{\sigma}}$$

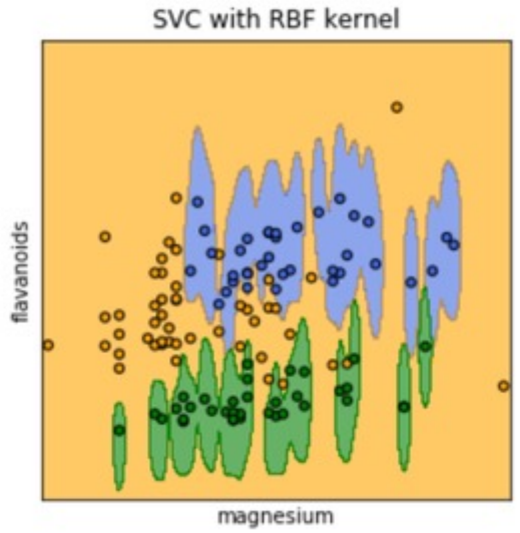
the most used

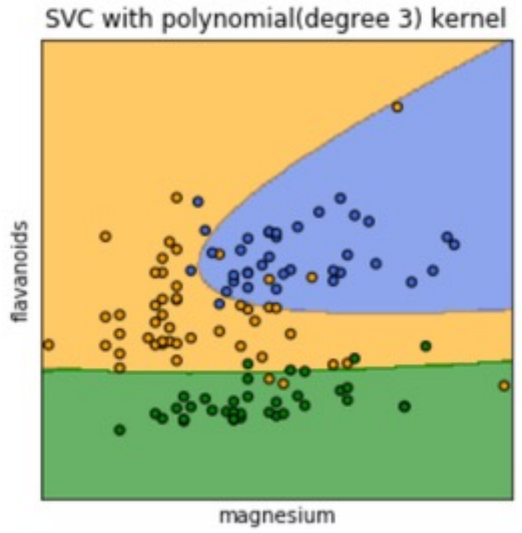
Gaussian Radial basis function kernel

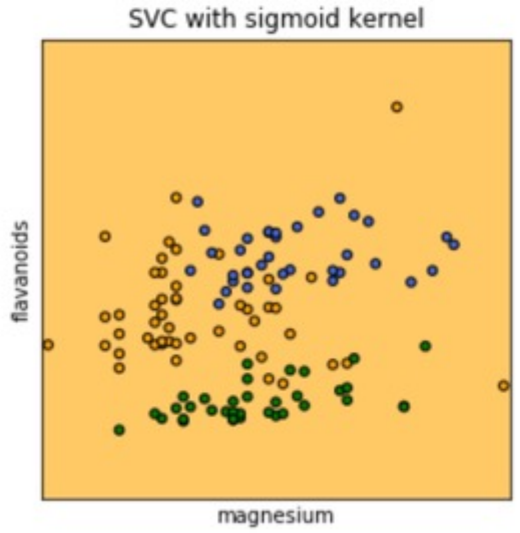
$$e^{-\frac{(x_i - x_j)^2}{2\sigma^2}}$$

Kernel examples (cont.d)



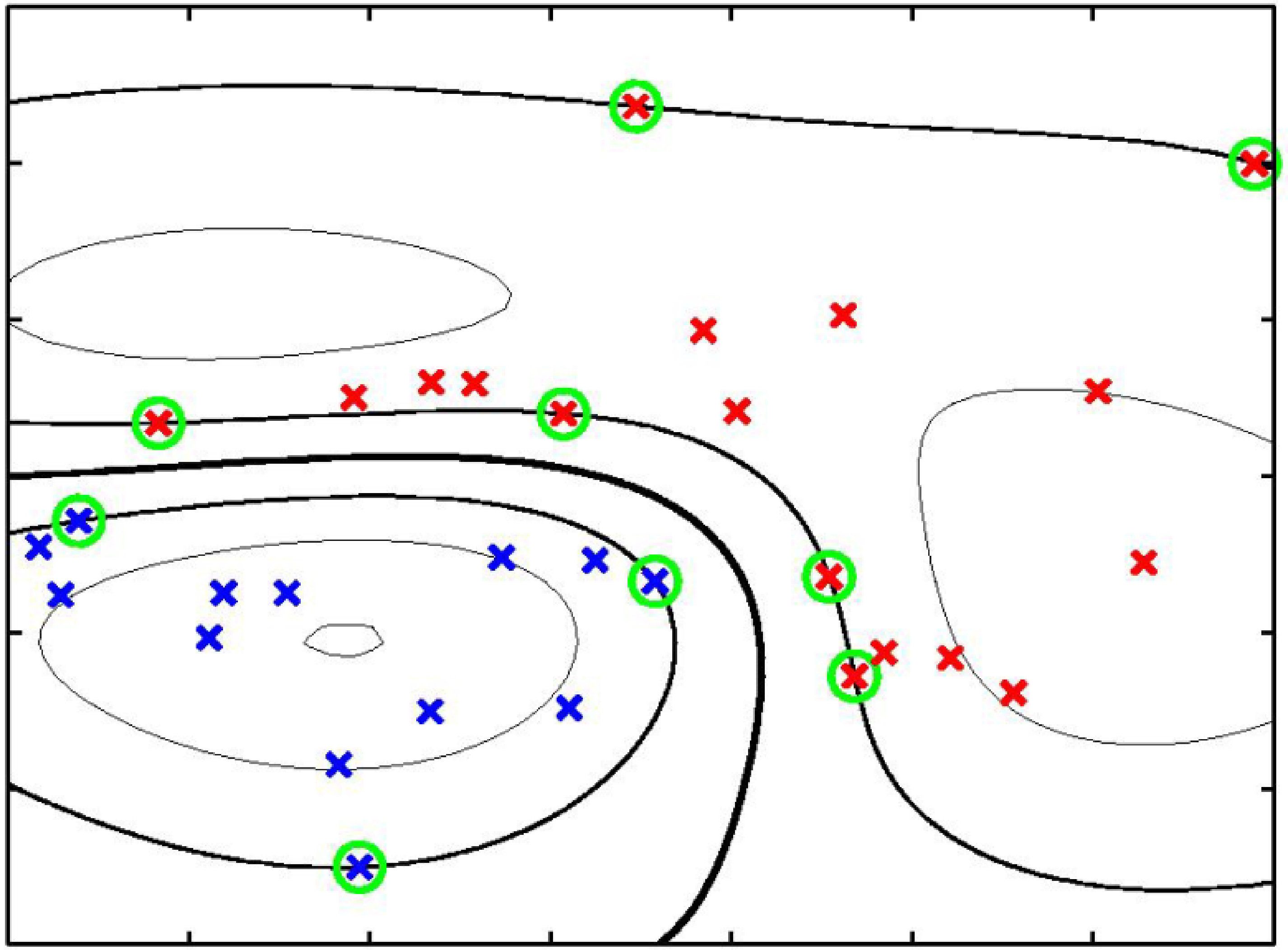






Kernel examples (cont.d)

Gaussian kernel



Parameters tuning

To use Support Vector Machines you have to define:

- the kernel function;
- potential parameters of the kernel function;
- The value for the regularization parameter *C*.

General rules for the set up do not exist, but you should make your choice on a validation set, usually through cross validation.

Advantages of SVMs

- There are **no local** minima (the optimization problem is quadratic -> ∃! optimal solution)
- The optimal solution can be found in polynomial time.
- There are few parameters to set up (*C*, type of kernel and specific kernel parameters)
- Solution is stable (ex. there is no problem of randomly initializing of weights just as in Neural Networks)
- Solution is sparse: it just involves support vectors.

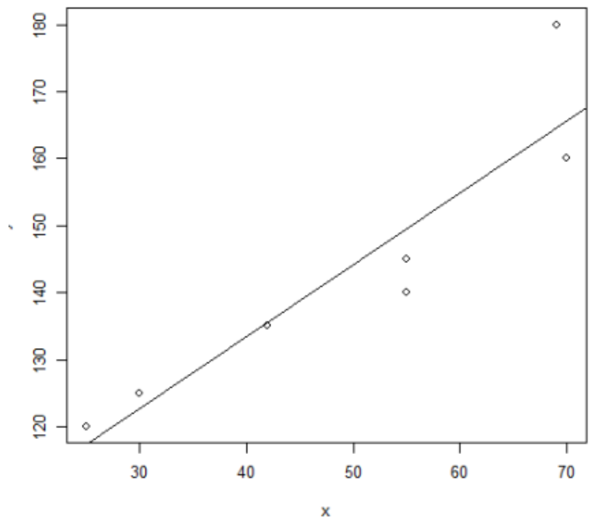
Generalized Linear Models

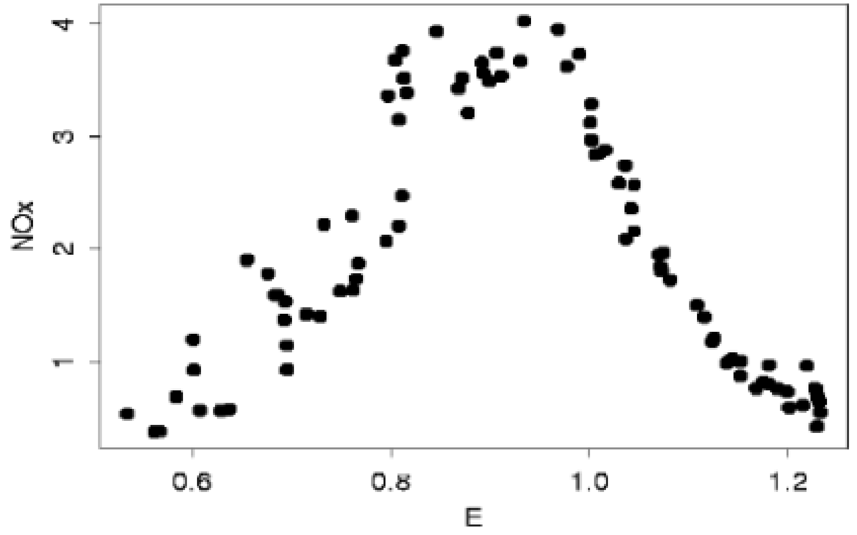
Basic functions

Generalized Linear Models

It is usual that our data cannot be approximated to a linear function.

What we need is to find a way to model nonlinear relations without increasing too much the complexity of the algorithm





Generalized Linear Models

The main and limiting characteristics of linear regression is the linearity of parameters

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot x_i = \theta^T x$$

h(x) keeps a linear relation w.r.t. the features space X. This linearity represents a limitation of the expressiveness of the model because the hypothesis is only able to approximate linear functions of the input. GLMs represent an extension to linear models that allow nonlinear transformations of the input

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot \phi(x_i) = \theta^T \phi(x)$$

Phi are called basic functions

$$Example \quad x_1, x_2, x_3 \rightarrow x_1, x_2, x_3, x_1x_2, x_1x_3, x_1^2, x_2^2, x_3^2$$

Generalized Linear Models

The main kinds of basic functions are:

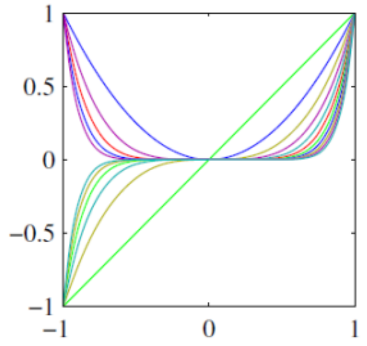
$$Linear \qquad \phi_i(x) = x_i$$

$$Polynomial \qquad \phi_i(x) = x_i^p$$

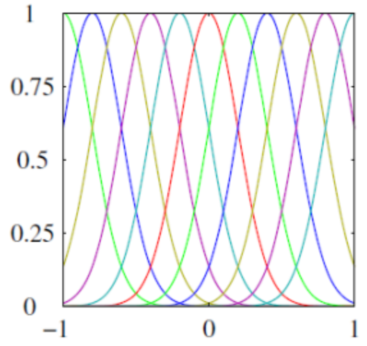
$$Gaussian \qquad \phi_i(x) = e^{-\frac{x-\mu_i}{2\sigma^2}}$$

$$Sigmoidal \qquad \phi_i(x) = \frac{1}{1+e^{-\frac{x-\mu_i}{2\sigma^2}}}$$

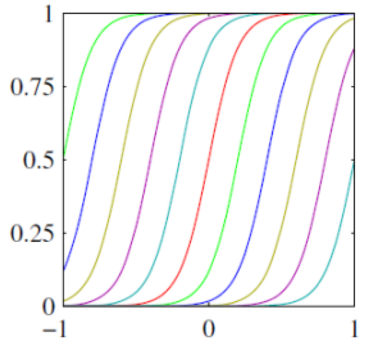
Polynomial



Gaussian



Sigmoidal



1
0.75
0.5
0.25
0
−1
0
1