Class 9

Support Vector Machines

Federico Girosi

Linear Classifier

2 classes problem: d-dimensional data set $\{(\mathbf{x}_i, y_i\}_{i=1}^{l}$

$$y_i = 1 \text{ if } \mathbf{x}_i \in C_1 \ , \qquad y_i = -1 \text{ if } \mathbf{x}_i \in C_2$$

A linear classifier (*Perceptron*) has the form

$$f(\mathbf{x}) = \theta(\mathbf{w} \cdot \mathbf{x} + b)$$

The parameters $\mathbf{w}$ and b are determined by minimizing the empirical risk:

$$\mathbf{w}_0, b_0 = \arg\min_{\mathbf{w},b} \sum_{i=1}^{l} |y_i - f(\mathbf{x}_i)|$$

Algorithms for training a Perceptron have been found many years ago, together with estimates of their generalization performances (Rosenblatt, 1956; Novikoff, 1962).

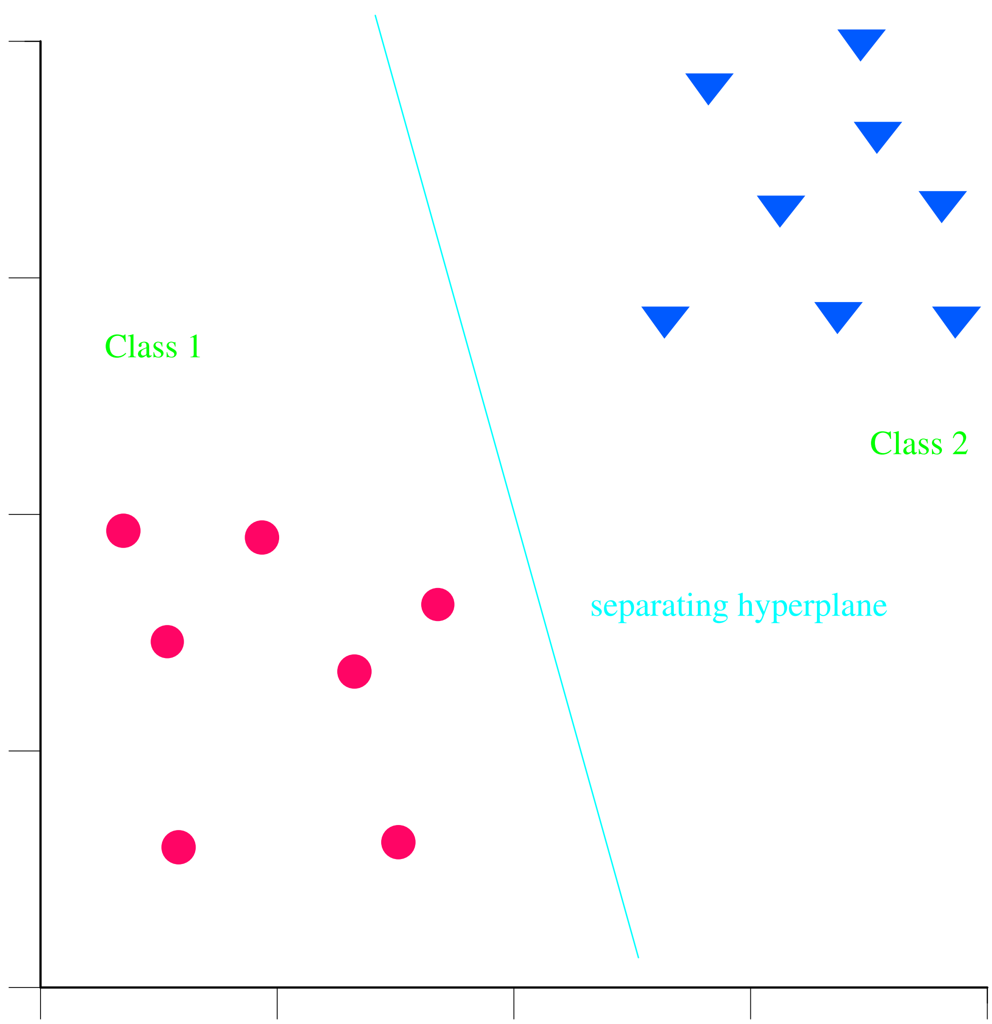
More complex models have also been proposed (Multilayer Perceptrons), but the question is:

Is the empirical risk minimization a good induction principle?

Is there any good alternative?

Linearly separable classes

A Perceptron algorithm will produce an hyper-plane which perfectly separates the training data.



Perceptrons

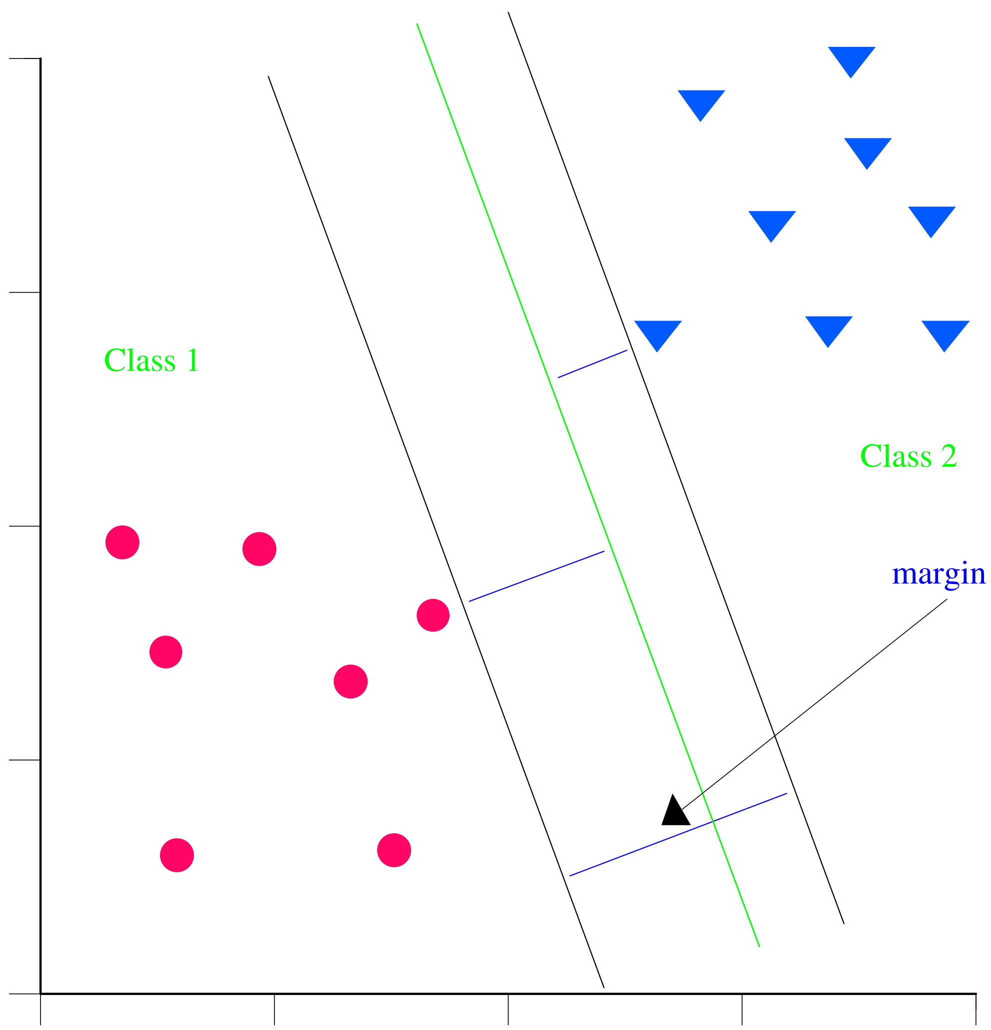
Why should a Perceptron generalize well?

There are many hyperplanes which can separate the data, but there is no criterion to choose.

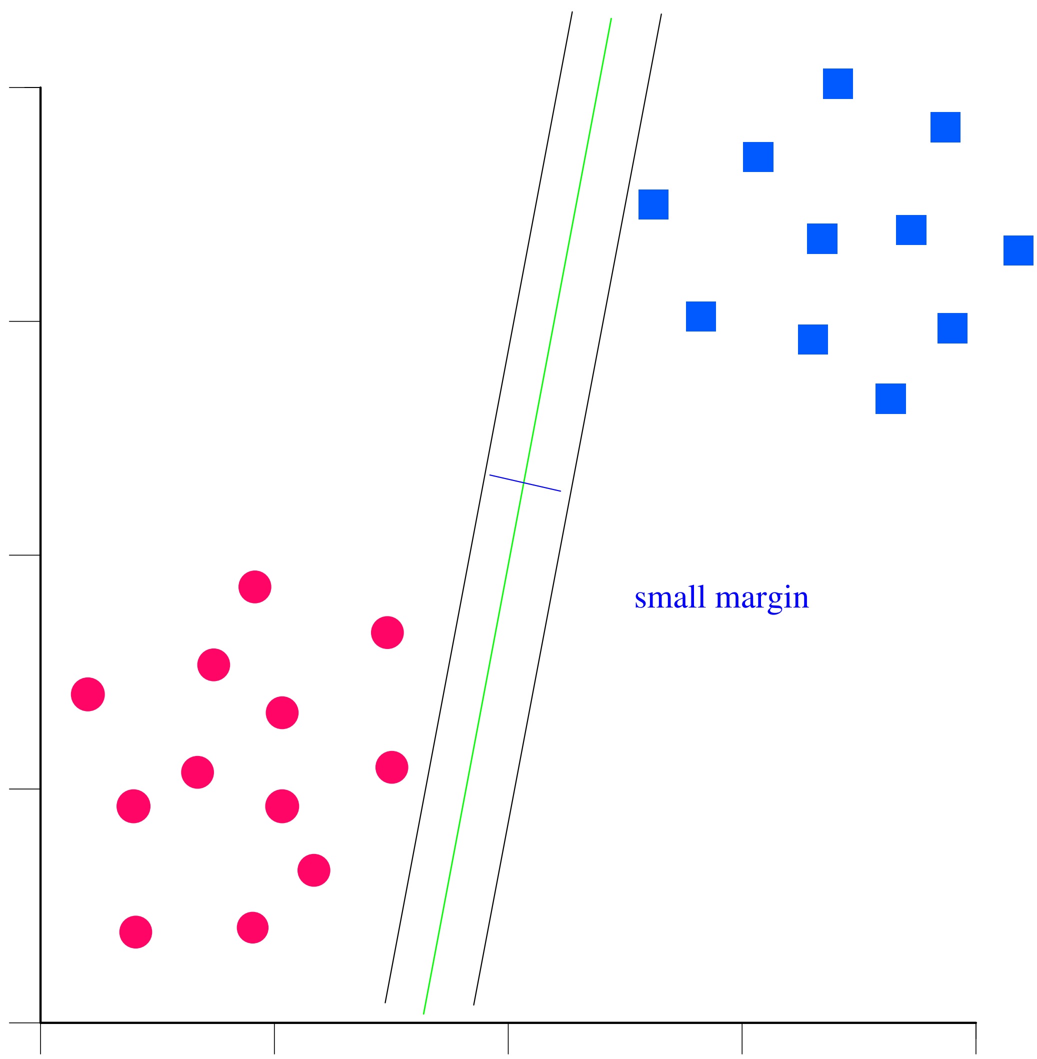
A criterion for choosing the optimal hyperplane can be embedded in an appropriate cost function ...

A new induction principle

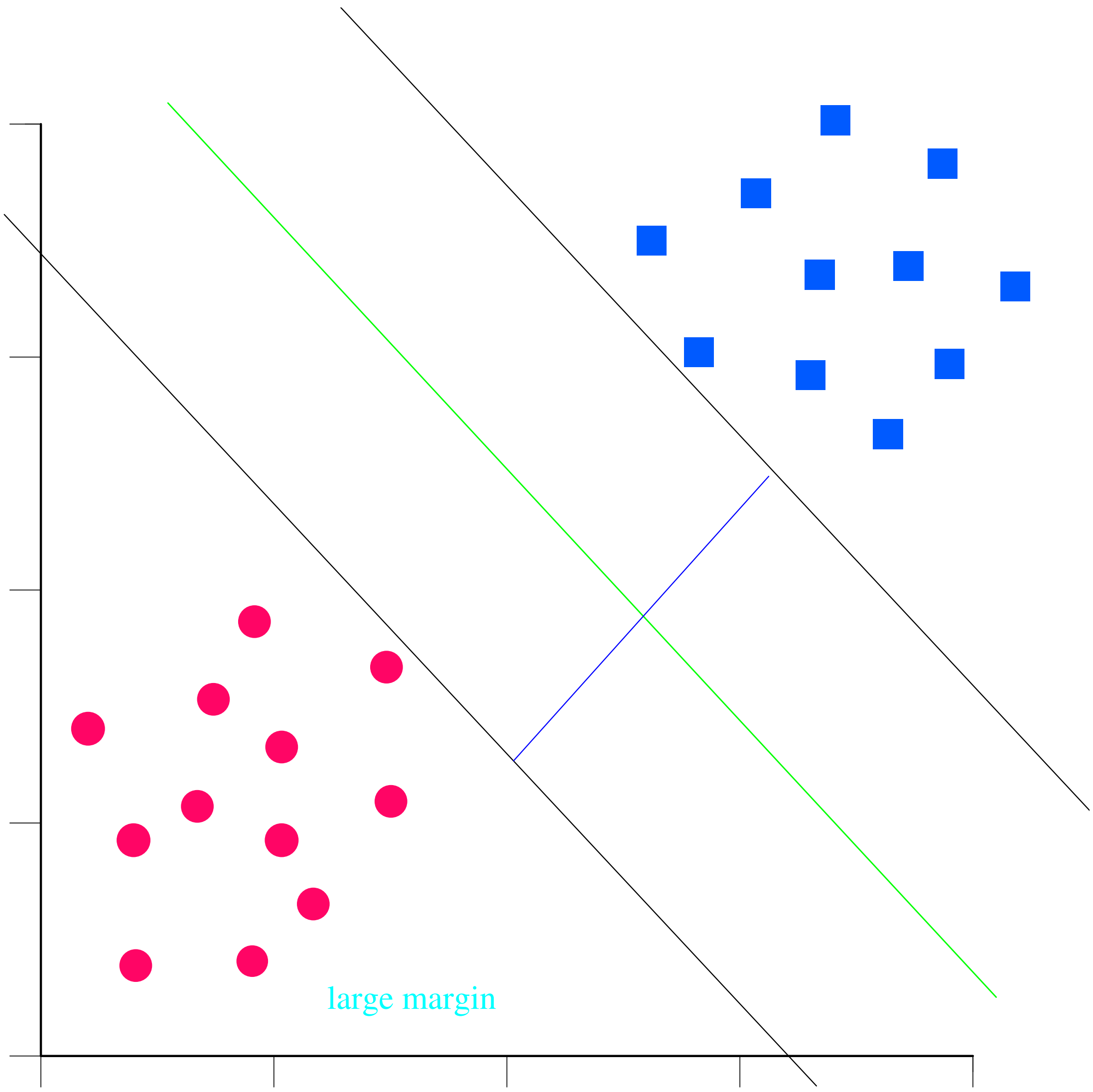
Among the hyperplanes which minimize the training error, look for the one with the largest *margin* (Vapnik, 1965, 1992).



Probably poor generalization



Probably good generalization



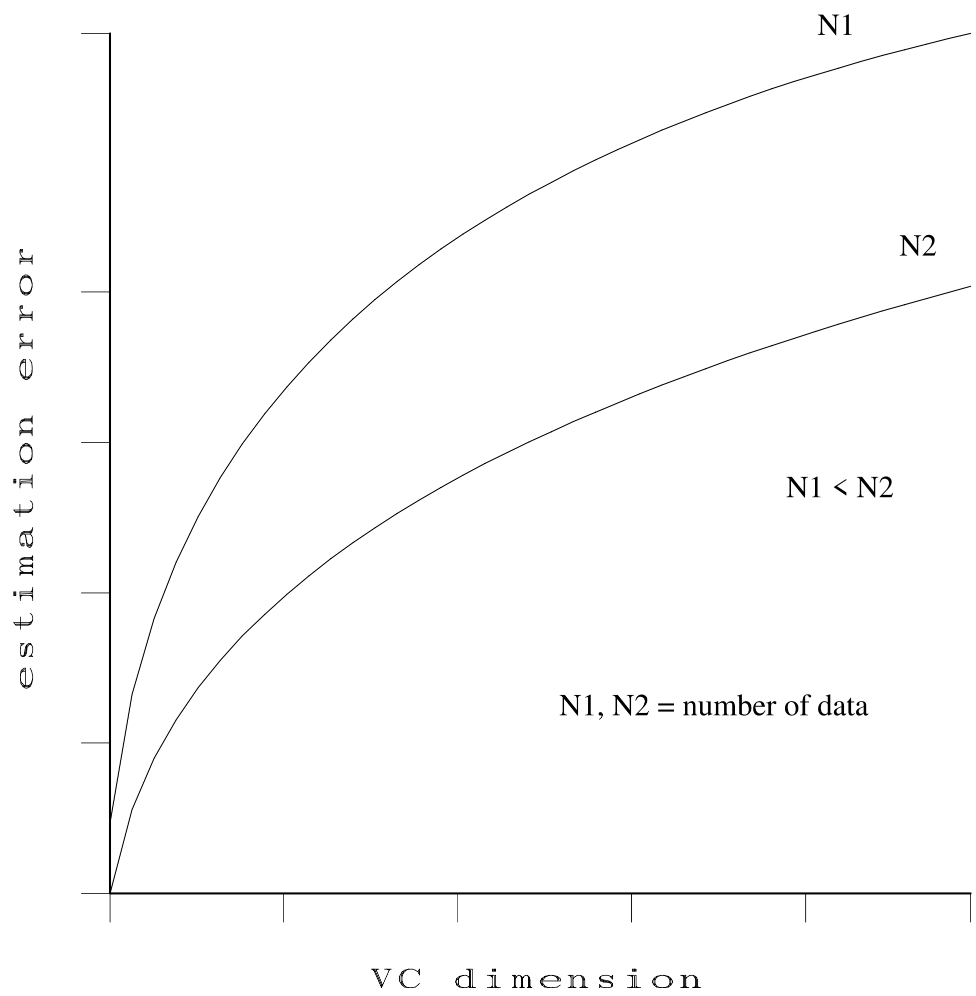
Is there a way to motivate rigorously the principle of maximum margin?

We need to go back to the theory of VC bounds ...

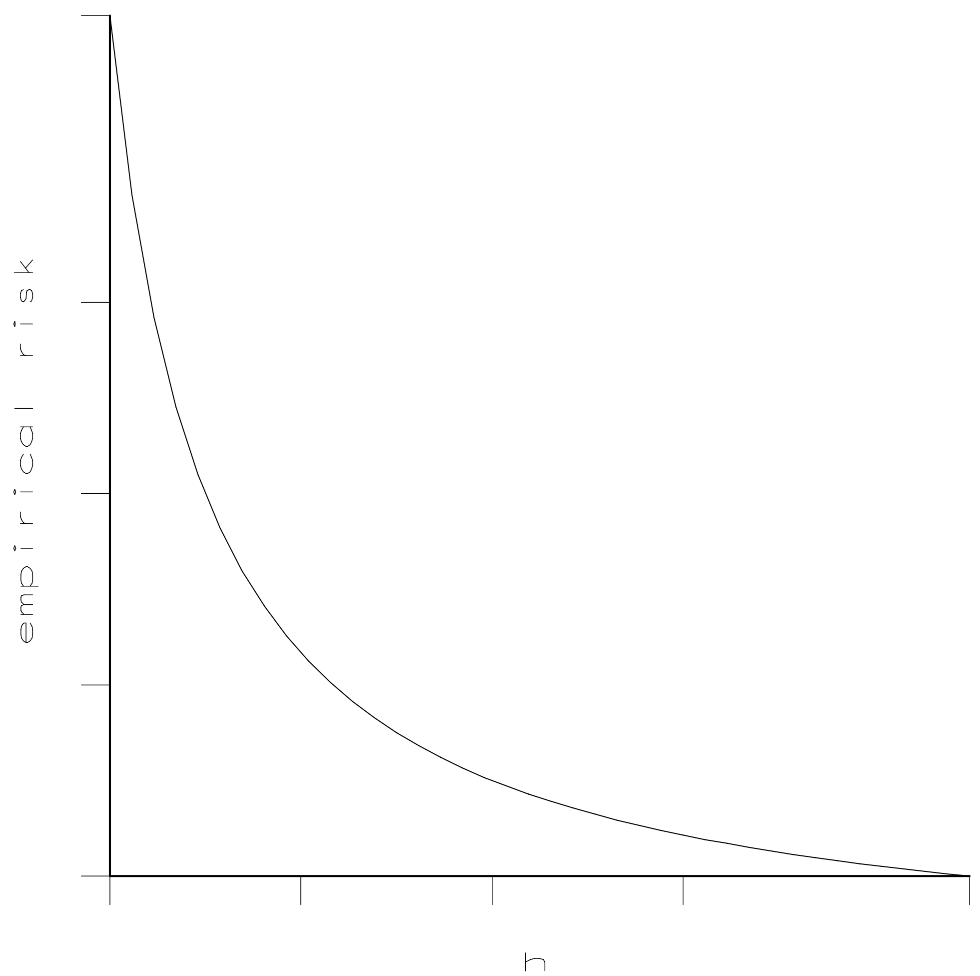
VC bounds

Bounds on the estimation error can be expressed in terms of the VC-dimension. For the classification case we have, with probability $1-\delta$ and for all $\alpha \in \Lambda$:

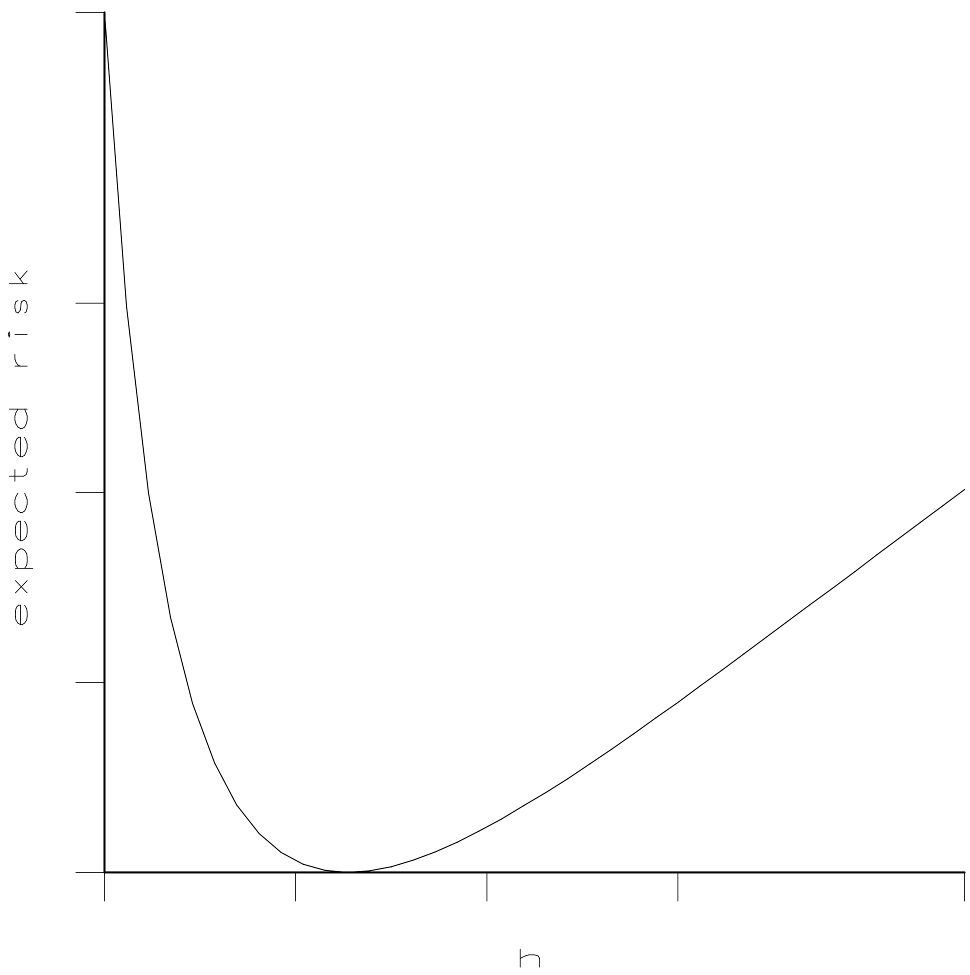
$$I(\alpha) \leq I_{\text{emp}}(\alpha) + \sqrt{\frac{h\left(\ln \frac{2l}{h} + 1\right) - \ln\left(\frac{\delta}{4}\right)}{l}}$$



Estimation error as a function of the VC dimension for different number of data points N_1 and N_2, with $N_1 < N_2$.



Empirical risk as a function of the VC-dimension h (typical behaviour).



Guaranteed risk (=bound on the expected risk) as a function of the VC-dimension h (typical behaviour).

Structural risk minimization (SRM)

$$\text{Guaranteed risk} = I_{\mathrm{emp}}(\alpha) + \sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right)-\ln\left(\frac{\delta}{4}\right)}{l}}$$

Expected risk $\leq$ Guaranteed risk

$$\Downarrow$$

Minimize the guaranteed risk, rather than the empirical risk!

This implies that the VC-dimension must be one of the variables of the problem.

Structural risk minimization (SRM)

Let S_h an hypothesis space of VC-dimension h, so that

$$S_1 \subset S_2 \subset \ldots \subset S_h \subset \ldots$$

Structural risk minimization consists in solving the following problem:

$$\min_{S_h} \left[R_{\text{emp}}(\alpha) + \sqrt{\frac{h\left(\ln \frac{2l}{h} + 1\right) - \ln\left(\frac{\delta}{4}\right)}{l}} \right]$$

Structural risk minimization (SRM)

Examples:

- $S_h \equiv \{$RBF nets with h centers$\}$

- $S_h \equiv \{$MLP nets with h units$\}$

- $S_h \equiv \{$Polynomials of degree $h\}$

- $S_h \equiv \{f \mid \phi[f] \leq h\}$

- $\ldots$

Structural risk minimization (SRM)

Structural Risk Minimization is difficult to implement in practice: the VC-dimension is a discrete variable.

What one ends up doing is an exhaustive search over hypothesis spaces of different VC-dimension: **it takes too long!.**

SRM applied to linear classifiers

In order to apply the SRM principle a structure on the hypothesis space has to be defined.

Notice that for the classifier

$$f(\mathbf{x}; \mathbf{w}, b) = \theta[\mathbf{x} \cdot \mathbf{w} + b]$$

there is some redundancy in the choice of $\mathbf{w}$ and b:

$$f(\mathbf{x}; \mathbf{w}, b) = f(\mathbf{x}; k\mathbf{w}, kb) \quad \forall k \neq 0$$

Some "canonical" rule to uniquely determine $\mathbf{w}$ and b has to be defined.

Canonical hyperplanes

Let X be a set of points $\{\mathbf{x}_i\}_{i=1}^{l}$ in the unit sphere in R^d.

Consider the hyperplane $(\mathbf{w}, b)$, that is the set of points such that

$$\mathbf{x} \cdot \mathbf{w} + b = 0$$

A **canonical hyperplane** corresponds to the choice of parameters $(\mathbf{w}, b)$ such that

$$\min_{\mathbf{x}_i \in X} |\mathbf{x} \cdot \mathbf{w} + b| = 1$$

The set of canonical hyperplanes coincides with the set of *all* hyperplanes.

Canonical Hyperplanes

The distance $d(\mathbf{w}, b)$ of a point $\mathbf{x}$ from the hyperplane $(\mathbf{w}, b)$ is:

$$d(\mathbf{w}, b) = \frac{|\mathbf{x} \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$$

In the canonical hyperplanes representation the distance between the hyperplane $(\mathbf{w}, b)$ and its closest point in the set $\{\mathbf{x}_i\}_{i=1}^{l}$ is

$$d(\mathbf{w}, b) = \frac{1}{\|\mathbf{w}\|}$$

A structure on the set of canonical hyperplanes

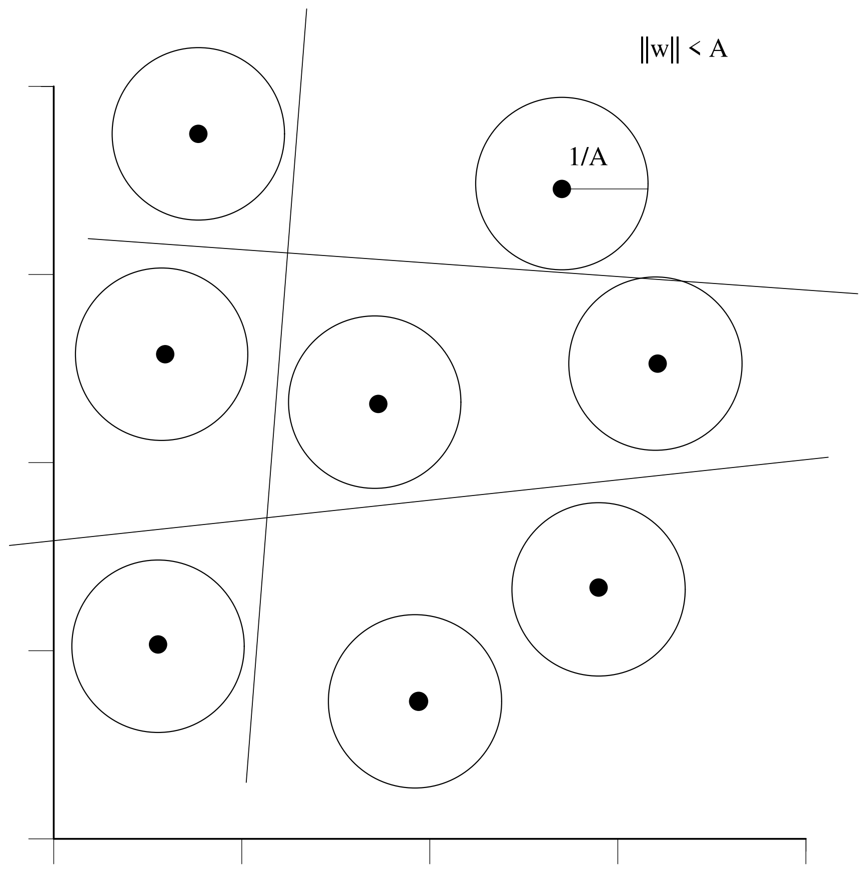
A structure on the set of canonical hyperplanes can be introduced by considering sets S_A such that

$$\|\mathbf{w}\| \leq A$$

If $A_1 \leq A_2 \leq \ldots \leq A_n \leq \ldots$ then

$$S_{A_1} \subset S_{A_2} \subset \ldots \subset S_{A_n} \subset \ldots$$

A structure on the set of canonical hyperplanes



If we impose the constraint $\|\mathbf{w}\| \leq A$ then the canonical hyperplanes cannot be closer than $\frac{1}{A}$ to any of the points $\mathbf{x}_i$.

Fundamental result

The VC-dimension of the set of canonical hyperplanes in R^d such that

$$\|\mathbf{w}\| \leq A$$

is

$$h \leq \min[A^2, d] + 1$$

Small $\|\mathbf{w}\| \Rightarrow$ small VC-dimension.

Linear separability: definition

A 2-class data set is said *linearly separable* if we can find $\mathbf{w}$ and b such that

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 & \quad \forall \mathbf{x}_i \in C_1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \quad \forall \mathbf{x}_i \in C_2 \end{aligned}$$

The equation of the separating hyperplane is $\mathbf{w} \cdot \mathbf{x} + b = 0$.

Let us define $y_i = 1$ is $\mathbf{x}_i \in C_1$ and $y_i = -1$ otherwise. The linear separability conditions above can be summarized as

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \forall \mathbf{x}_i \in C_1 \cup C_2$$

SRM applied to linearly separable problems

The idea is to minimize

$$I_{\text{emp}}(\mathbf{w}, b) + \sqrt{\frac{h\left(\ln\frac{2l}{h} + 1\right)}{l}}$$

Since the problem is linearly separable we can always find $\mathbf{w}$ and b such that $I_{\text{emp}}(\mathbf{w}, b) = 0$.

Among all such w and b find the ones that minimizes the VC-dimension h, or a bound on the VC-dimension h.

SRM for Linearly Separable Problems

Since the VC-dimension is proportional to $\|\mathbf{w}\|$, in order to find the hyperplane which separates the data points with minimum expected risk, solve the following minimization problem:

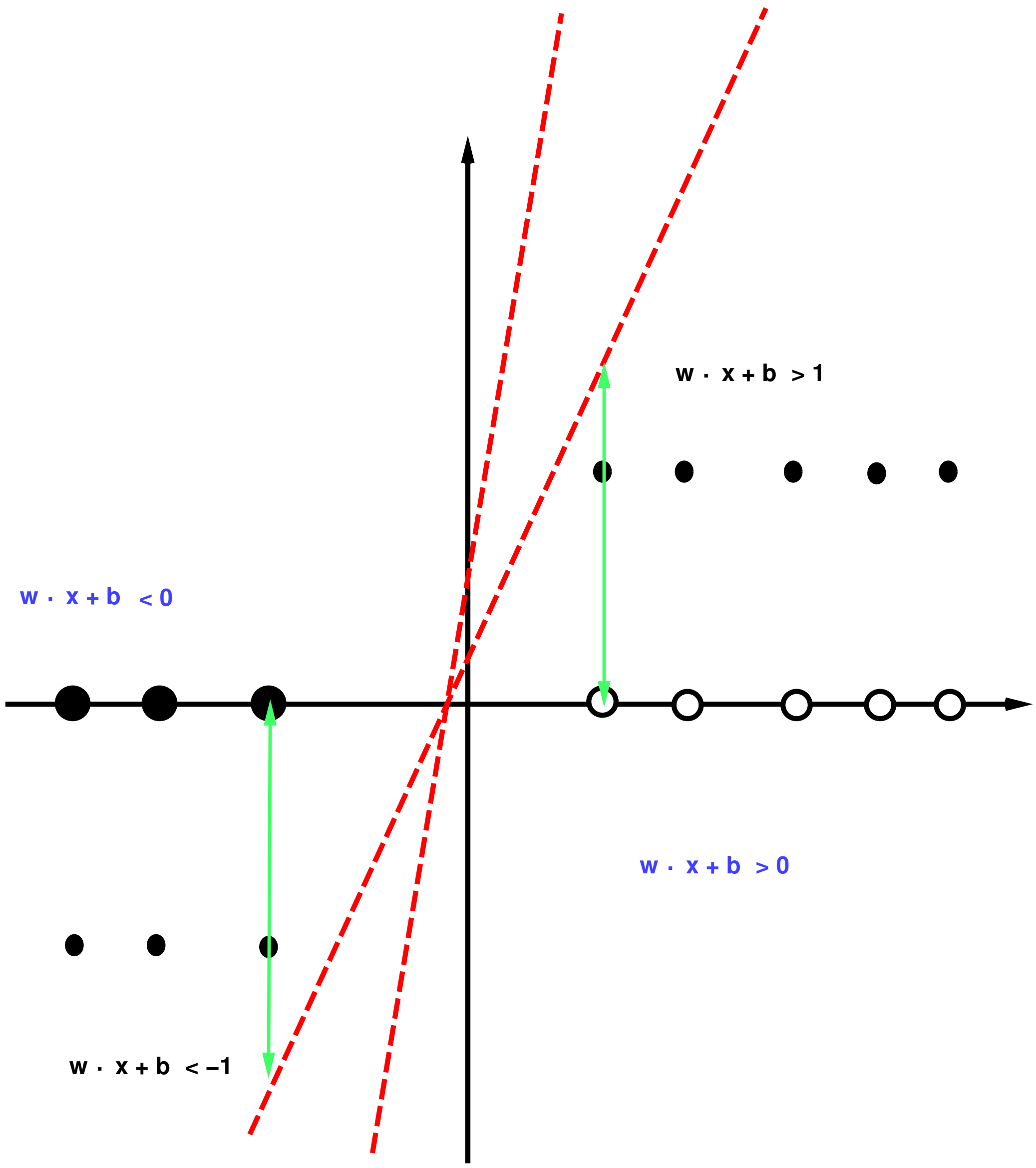
$$\min_{\mathbf{w}} \|\mathbf{w}\|^2$$

subject to the inequality constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$$

We have therefore to solve a classic *quadratic programming* problem.

Linear Separability



Definition of margin

Given an hyperplane $(\mathbf{w}, b)$ which separates the data set, its **margin** $\rho(\mathbf{w}, b)$ is defined as *the sum of the distances between the separating hyperplane and its closest points in each of the 2 classes*.

Denoting by $d(\mathbf{w}, b; \mathbf{x})$ the distance between the separating hyperplane $(\mathbf{w}, b)$ and a point $\mathbf{x}$, the margin is:

$$\rho(\mathbf{w}, b) = \min_{\mathbf{x} \in C_1} d(\mathbf{w}, b; \mathbf{x}) + \min_{\mathbf{x} \in C_2} d(\mathbf{w}, b; \mathbf{x})$$

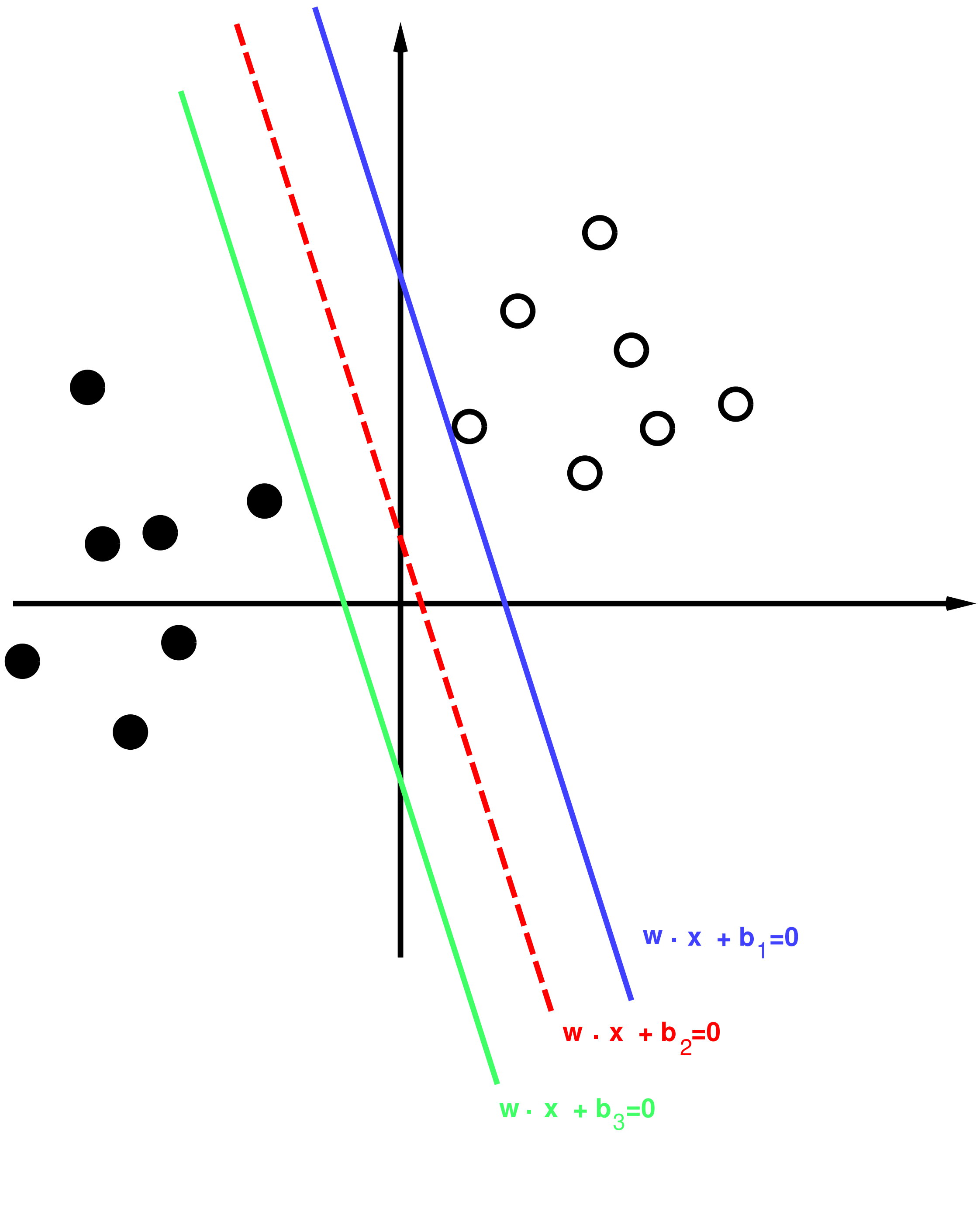
The distance between the separating hyperplane $(\mathbf{w}, b)$ and a point $\mathbf{x}$ is

$$d(\mathbf{w}, b; \mathbf{x}) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{||\mathbf{w}||}$$

The margin is therefore:

$$\rho(\mathbf{w}, b) = \frac{1}{||\mathbf{w}||} \left(\min_{\mathbf{x} \in C_1} |\mathbf{w} \cdot \mathbf{x} + b| + \min_{\mathbf{x} \in C_2} |\mathbf{w} \cdot \mathbf{x} + b| \right)$$

Notice that if $(\mathbf{w}, b)$ is a separating hyperplane then the margin does not depend on b.



Since the margin does not depend on b, we can always choose b in a such a way that

$$\begin{aligned} \min_{\mathbf{x} \in C_1} \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} &= \min_{\mathbf{x} \in C_2} \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} = \\ &= \min_{\mathbf{x} \in C_1 \cup C_2} \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} \end{aligned}$$

This equivalent to consider always the separating hyperplane which is halfway between class 1 and class 2.

A new expression for the margin

Using the canonical hyperplanes representation and considering always the separating hyperplane which is halfway between class 1 and class 2, the margin

$$\rho(\mathbf{w}, b) = \frac{1}{||\mathbf{w}||}\left(\min_{\mathbf{x}\in C_1} |\mathbf{w}\cdot\mathbf{x}+b| + \min_{\mathbf{x}\in C_2} |\mathbf{w}\cdot\mathbf{x}+b|\right)$$

becomes simply

$$\rho(\mathbf{w}, b) = \frac{2}{||\mathbf{w}||}$$

Optimal Margin Classifier, or Support Vector Machine (SVM)

Since maximizing $\frac{1}{\|\mathbf{w}\|}$ and minimizing $\|\mathbf{w}\|^2$ are clearly equivalent, the hyperplane with maximum margin can be found by solving the following *Quadratic Programming* problem:

$$(\mathbf{w}_0, b_0) = \arg\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the inequality constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad 1 = 1, \ldots, l$$

SVM minimizes an upper bound on the VC-dimension

$$\Downarrow$$

SVM implements the Structural Risk Minimization Principle.

Lagrange multipliers method

Minimizing a smooth, convex function $F(\mathbf{x})$ subject to the constraints

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, l$$

is equivalent to find the saddle point of the Lagrangian L:

$$L(\mathbf{x}; \mathbf{h}) = F(\mathbf{x}) + \sum_{i=1}^{l} h_i g_i(\mathbf{x})$$

where the minimum is taken with respect to $\mathbf{x}$, the maximum with respect to $\mathbf{h}$ and $\mathbf{h} = (h_1, \ldots, h_l)$ is a vector of *non-negative* Lagrange multipliers (auxiliary parameters).

At the minimum ($\mathbf{x}_0$) the following conditions (Kuhn-Tucker) are satisified:

$$h_i g_i(\mathbf{x}_0) = 0 \quad i = 1, \ldots, l$$

Solution of the quadratic programming problem

We first construct the Lagrangian:

$$L(\mathbf{w}, b; \mathbf{h}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} h_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

where $\mathbf{h} = (h_1, \ldots, h_l)$ is the vector of *non-negative* Lagrange multipliers.

Let us do the minimization over **w** and b first:

$$L(\mathbf{w}, b; \mathbf{h}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} h_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} h_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} h_i y_i = 0$$

Therefore we have the conditions:

$$\mathbf{w}_0 = \sum_{i=1}^{l} h_i y_i \mathbf{x}_i \ , \qquad \mathbf{y} \cdot \mathbf{h} = 0$$

Using the previous results we obtain:

$$L(\mathbf{h}) = \sum_{i=1}^{l} h_i - \frac{1}{2}\|\mathbf{w}_0\|^2$$

and therefore, in vector notation:

$$L(\mathbf{h}) = \mathbf{h} \cdot \mathbf{1} - \frac{1}{2}\mathbf{h} \cdot D\mathbf{h}$$

where $D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ and $\mathbf{1} = (1, 1, \ldots, 1)$.

The problem we have to solve

In conclusion, the following *dual* quadratic programming problem has to be solved:

$$\max_{\mathbf{h}} \left(\mathbf{h} \cdot \mathbf{1} - \frac{1}{2}\mathbf{h} \cdot D\mathbf{h}\right)$$

subject to the constraints:

$$\mathbf{h} \cdot \mathbf{y} = 0$$

$$\mathbf{h} \geq 0$$

Notice that the number of variables in this problem is equal to the number of data points l.

Important observation

From Kuhn-Tucker it follows that, if the minimum is achieved at $(\mathbf{w}_0, b_0, \mathbf{h}_0)$, then the following relationship holds:

$$h_i^0[y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1] = 0$$

Therefore $h_i^0 \neq 0$ only for those points $\mathbf{x}_i$ that satisfy the *equality*:

$$y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) = 1$$

These points are called **support vectors**

In conclusion

Once the multipliers $\mathbf{h}$ have been found, the optimal hyperplane is derived as:

$$\mathbf{w}_0 = \sum_{\text{support vectors}} h_i y_i \mathbf{x}_i$$

$$b_0 = -\frac{1}{2}(\mathbf{x}_i^{(1)} \cdot \mathbf{w}_0 + \mathbf{x}_i^{(2)} \cdot \mathbf{w}_0)$$

where $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$ denote any support vector respectively of class C_1 and C_2.

Another important observation

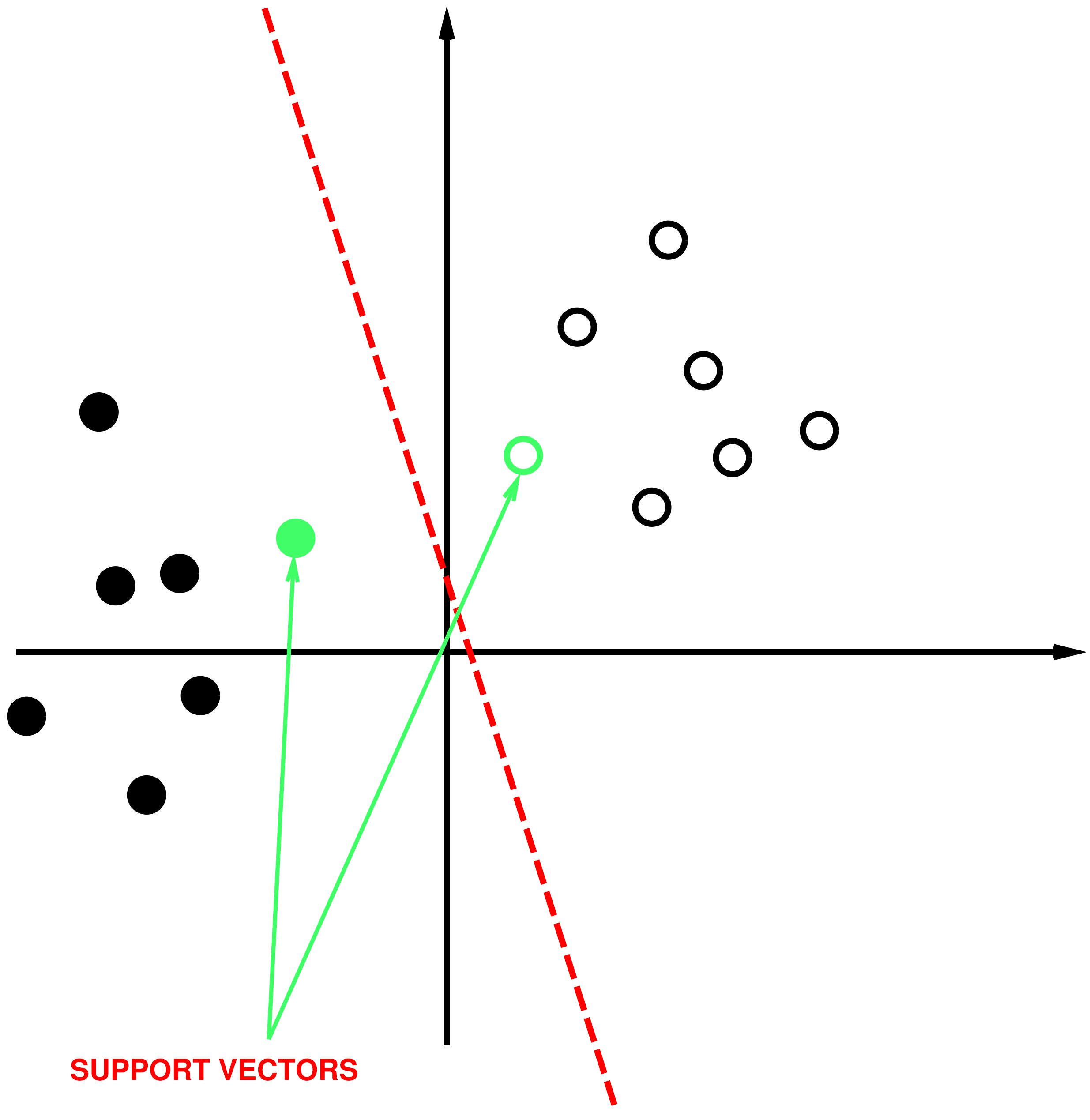
The fact that that the optimal hyperplane is determined only by the support vectors

$$\mathbf{w}_0 = \sum_{\text{support vectors}} h_i y_i \mathbf{x}_i$$

is most remarkable. *Usually the number of support vectors is very small*.

The whole data set could be replaced by these few points and the same hyperplane would be obtained.

All the information contained in the data set is summarized by the support vectors.


SUPPORT VECTORS

Performance estimates

Once the support vectors have been found, their number allows to estimate the generalization performances of this technique. In fact, *the expected probability of committing an error on a test example is bounded as follows:*

$$E[P(\text{error})] \leq \frac{E[\text{number of support vectors}]}{\text{number of training vectors}}$$

Notice how easy is to estimate this bound.

What if the 2 classes are not linearly separable?

The Support Vector Machine (Vapnik, 1992) can be applied to nonlinearly separable classes as well. The idea is to replace the inequality

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

with the inequality

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$$

where ξ_i are non-negative variables, that allow for some overlapping between the classes.

“Soft margin” classifier

When classes overlap we can try to maximize the margin and minimize the number of misclassification errors at the same time. We therefore solve for:

$$\min_{\mathbf{w},\xi_i} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i \right)$$

subject to the inequality constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \quad i = 1, \ldots, l$$

$$\xi_i \geq 0 \quad i = 1, \ldots, l$$

The effect of the term $C \sum_{i=1}^{l} \xi_i$ is to penalize the misclassification error. More complex cost functions can be devised.

“Soft margin” classifier

The Lagrangian is now:

$$
\begin{aligned}
L(\mathbf{w}, b, \xi; \mathbf{h}, \mathbf{r}) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i + \\
&- \sum_{i=1}^{l} h_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] + \\
&- \sum_{i=1}^{l} r_i\xi_i
\end{aligned}
$$

where $\mathbf{h} = (h_1, \ldots, h_l)$ and $\mathbf{r} = (r_1, \ldots, r_l)$ are Lagrange multipliers.

Let us do the minimization over $\mathbf{w}$, b and ξ first:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} h_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} h_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - h_i - r_i = 0$$

Therefore we have the conditions:

$$\mathbf{w}_0 = \sum_{i=1}^{l} h_i y_i \mathbf{x}_i \ , \quad \mathbf{y} \cdot \mathbf{h} = 0, \quad \mathbf{r} = C\mathbf{1} - \mathbf{h}$$

The problem we have to solve

In conclusion, the following *dual* quadratic programming problem has to be solved:

$$\max_{\mathbf{h}} \left(\mathbf{h} \cdot \mathbf{1} - \frac{1}{2}\mathbf{h} \cdot D\mathbf{h}\right)$$

subject to the constraints:

$$\mathbf{h} \cdot \mathbf{y} = 0$$

$$0 \leq \mathbf{h} \leq C\mathbf{1}$$

where $D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$.

Notice that the number of variables in this problem is equal to the number of data points l.

In conclusion

Once the multipliers $\mathbf{h}$ have been found, the classifier is simply $f(\mathbf{x}) = \theta(\mathbf{w}_0 \cdot \mathbf{x} + b_0)$ where $\mathbf{w}_0$ is derived as:

$$\mathbf{w}_0 = \sum_{\text{support vectors}} h_i y_i \mathbf{x}_i$$

How do we find b_0? From Kuhn-Tucker we know that

$$h_i[y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1 + \xi_i] = 0, \quad r_i \xi_i = 0 \ .$$

Take any support vector with $h_i < C$. Then $r_i = C - h_i \neq 0$. Then $\xi_i = 0$ and

$$b_0 = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

Who wants a linear classifier?

Linear classifiers in the classical sense are very limited. However, one can consider linear classifiers in *feature space*, rather than in the original space.

Idea: map $\mathbf{x}$ into a vector of features $\phi_1, \ldots, \phi_n$:

$$\mathbf{x} \to \mathbf{z}(\mathbf{x}) = (a_1\phi_1(\mathbf{x}), a_2\phi_2(\mathbf{x}) \ldots, a_n\phi_n(\mathbf{x}))$$

where ϕ_i are fixed functions, and consider a linear classifier in the variable $\mathbf{z}(\mathbf{x})$ instead of $\mathbf{x}$.

Example

$$
\begin{aligned}
&\mathbf{x} = (x, y, z)\\
&\phi_1(\mathbf{x}) = x\\
&\phi_2(\mathbf{x}) = y\\
&\phi_3(\mathbf{x}) = z\\
&\phi_4(\mathbf{x}) = x^2\\
&\phi_5(\mathbf{x}) = y^2\\
&\phi_6(\mathbf{x}) = z^2\\
&\phi_7(\mathbf{x}) = xy\\
&\phi_8(\mathbf{x}) = xz\\
&\phi_9(\mathbf{x}) = yz
\end{aligned}
$$

$$\mathbf{z}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_9(\mathbf{x}))$$

A linear classifier in the feature space ($\mathbf{z}(\mathbf{x})$)

$$f(\mathbf{x}) = \theta(\mathbf{w} \cdot \mathbf{z}(\mathbf{x}) + b)$$

$$\Downarrow$$

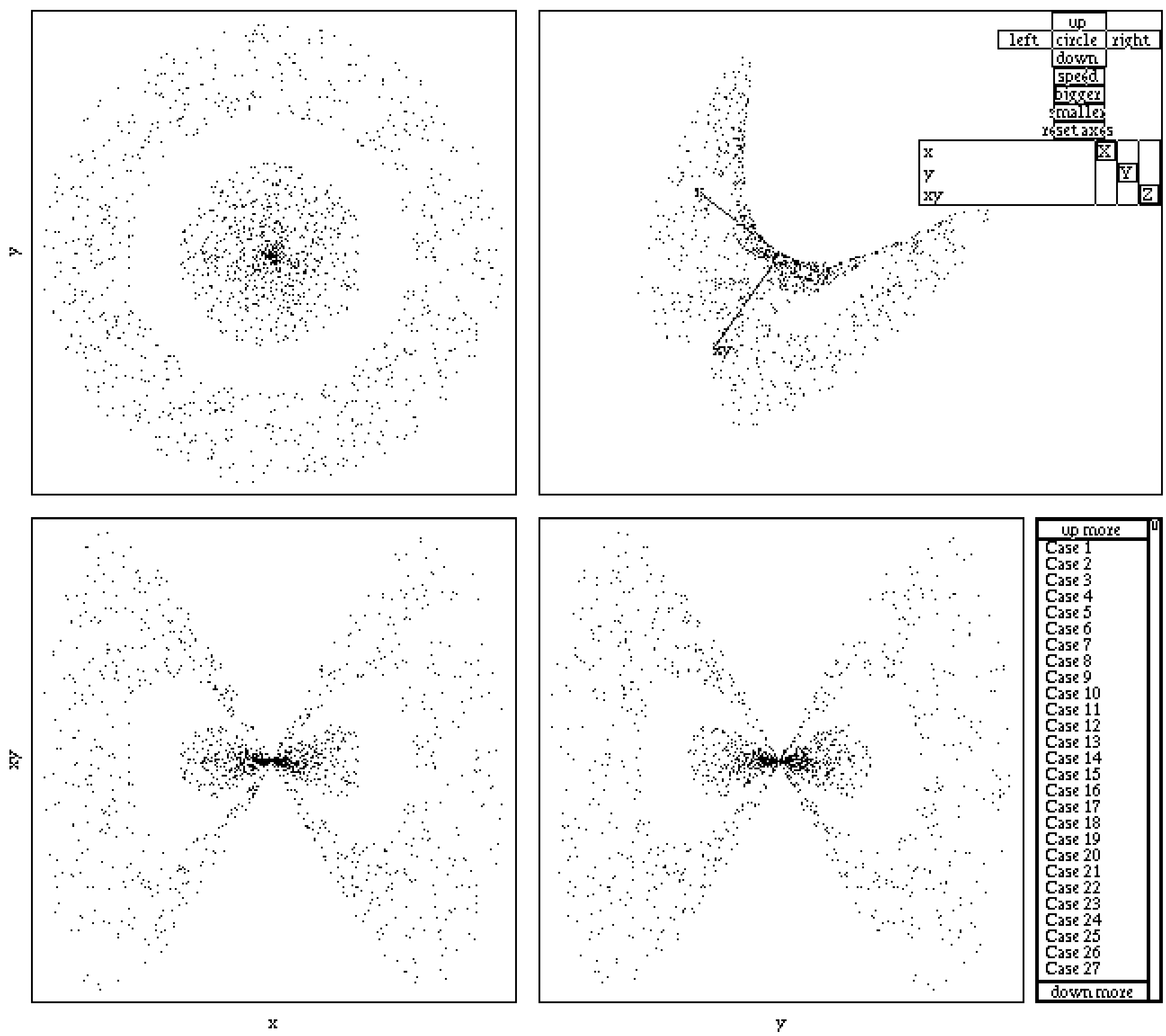
a polynomial classifier in the original space ($\mathbf{x}$).

Original Data Set

x	y	$class$
-0.26	-0.12	1
-0.57	-0.05	1
0.07	0.54	1
0.03	0.07	1
-0.58	-0.47	1
0.45	-2.25	-1
-0.51	-1.67	-1
1.55	-1.03	-1
-2.42	0.26	-1
...	...	...

Data Set in Feature Space

x	y	$x * y$	$class$
-0.26	-0.12	0.03	1
-0.57	-0.05	0.03	1
0.07	0.54	0.04	1
0.03	0.07	0.00	1
-0.58	-0.47	0.28	1
0.45	-2.25	-1.03	-1
-0.51	-1.67	0.85	-1
1.55	-1.03	-1.61	-1
-2.42	0.26	-0.65	-1
...	...	...	

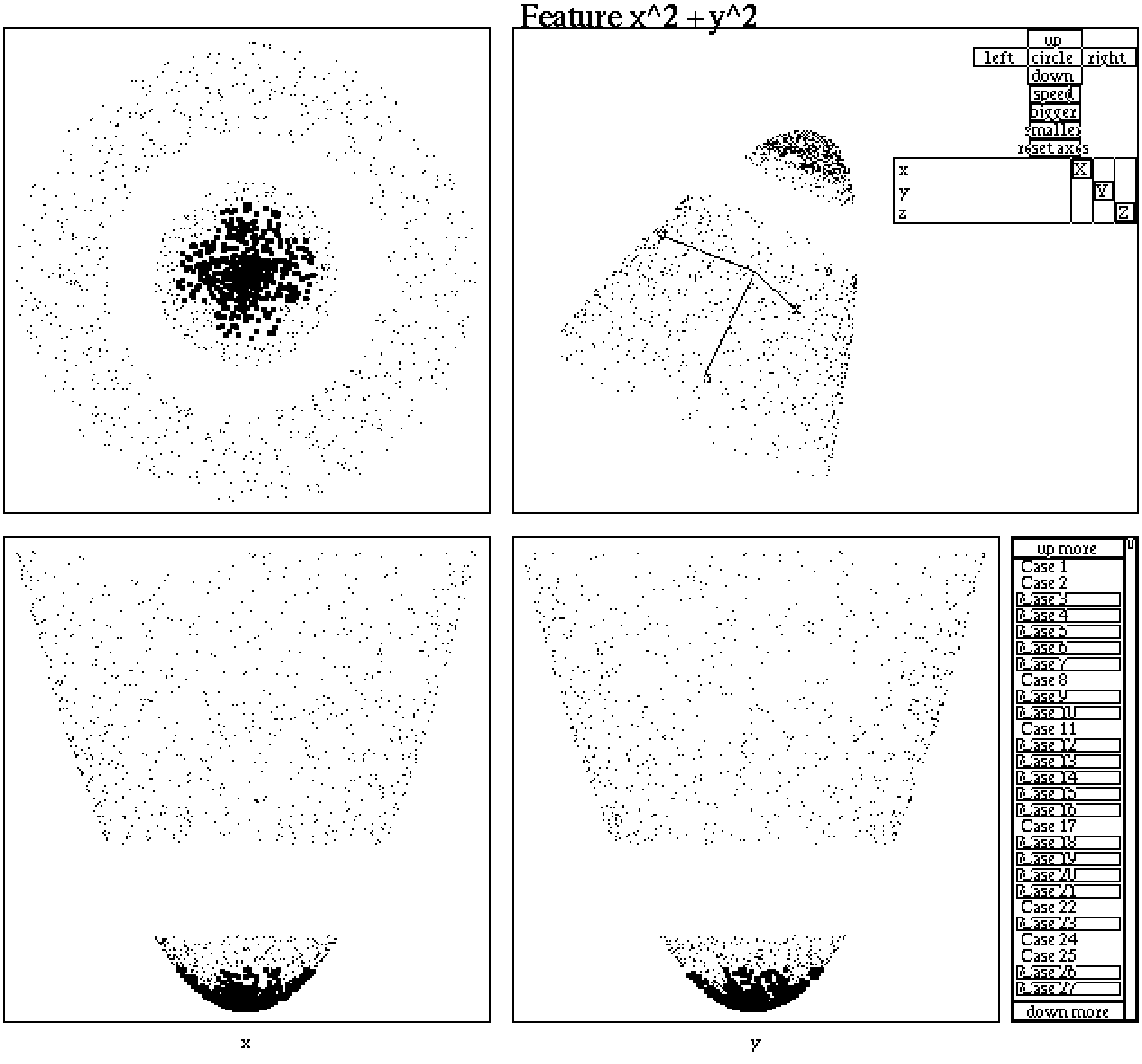


y
up
left
circle
right
down
speed
bigger
smaller
reset axes
x
y
xy
X
Y
Z
xy
up more
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14
Case 15
Case 16
Case 17
Case 18
Case 19
Case 20
Case 21
Case 22
Case 23
Case 24
Case 25
Case 26
Case 27
down more
x
y

Data Set in Feature Space

x	y	$x^2 + y^2$	$class$
-0.26	-0.12	0.08	1
-0.57	-0.05	0.33	1
0.07	0.54	0.30	1
0.03	0.07	0.00	1
-0.58	-0.47	0.57	1
0.45	-2.25	5.28	-1
-0.51	-1.67	3.07	-1
1.55	-1.03	3.49	-1
-2.42	0.26	5.97	-1
...	...	...	

Feature x^2 + y^2



SVM in feature space

The following *dual* quadratic programming problem has to be solved:

$$\max_{\mathbf{h}} \left(\mathbf{h} \cdot \mathbf{1} - \frac{1}{2}\mathbf{h} \cdot D\mathbf{h}\right)$$

subject to the constraints:

$$\mathbf{h} \cdot \mathbf{y} = 0$$

$$0 \leq \mathbf{h} \leq C\mathbf{1}$$

where $D_{ij} = y_i y_j \mathbf{z}(\mathbf{x}_i) \cdot \mathbf{z}(\mathbf{x}_j)$.

SVM in feature space

The final classifier has the form

$$f(\mathbf{x}) = \theta\left(\sum_{i=1}^{l} h_i y_i \mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}_i) + b\right)$$

and therefore it is not linear in $\mathbf{x}$ anymore. Notice that:

- We never need to compute $\mathbf{z}(\mathbf{x})$, but only scalar products, like $\mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}_i)$ and $\mathbf{z}(\mathbf{x}_i) \cdot \mathbf{z}(\mathbf{x}_j)$

- if the dimensionality of $\mathbf{x}$ is large, the dimensionality of $\mathbf{z}$ could be huge (a 2nd degree polynomial in 265 variables has 35245 terms ...).

A crucial step

Let us define the *kernel function* K as following:

$$\mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$$

Notice that we only need to know $K(\mathbf{x}, \mathbf{y})$ in order to solve the problem.

Good features should ...

- lead to "interesting" decision surfaces:

$$f(\mathbf{x}) = \theta\left(\sum_{i=1}^{l} h_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$$

- be such that the scalar product $K(\mathbf{x}, \mathbf{y})$ is easy to compute even for large n.

Example: Polynomial of Degree 2 in 2 Variables

$$\mathbf{x} = (x_1, x_2) \ , \quad \mathbf{y} = (y_1, y_2)$$

$$\mathbf{z}(\mathbf{x}) = (1, \sqrt{2}\ x_1, \sqrt{2}\ x_2, x_1^2, x_2^2, \sqrt{2}\ x_1 x_2)$$

$$\mathbf{z}(\mathbf{y}) = (1, \sqrt{2}\ y_1, \sqrt{2}\ y_2, y_1^2, y_2^2, \sqrt{2}\ y_1 y_2)$$

$$\Downarrow$$

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

Example: Polynomial of Degree 3 in 2 Variables

$$\mathbf{z}(\mathbf{x}) = (1, \sqrt{3}\,x_1, \sqrt{3}\,x_2, \sqrt{3}\,x_1^2, \sqrt{3}\,x_2^2, \sqrt{6}x_1x_2, \sqrt{3}\,x_1^2x_2, \sqrt{3}\,x_1x_2^2, x_1^3\,, x_2^3\,)$$

$$\mathbf{z}(\mathbf{y}) = (1, \sqrt{3}\,y_1, \sqrt{3}\,y_2, \sqrt{3}\,y_1^2, \sqrt{3}\,y_2^2, \sqrt{6}y_1y_2, \sqrt{3}\,y_1^2y_2, \sqrt{3}\,y_1y_2^2, y_1^3\,, y_2^3\,)$$

$$\Downarrow$$

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^3$$

Example: Infinite Dimensional Feature Space in One Variable

Let $x \in [0, 2\pi]$ and h a positive number such that $h < 1$:

$$\mathbf{z}(x) = (1, h^{\frac{1}{2}} \sin(x), h^{\frac{1}{2}} \cos(x), h \sin(2x), h \cos(2x), \ldots, h^{\frac{n}{2}} \sin(nx), h^{\frac{n}{2}} \cos(nx), \ldots)$$

$$\mathbf{z}(y) = (1, h^{\frac{1}{2}} \sin(y), h^{\frac{1}{2}} \cos(y), h \sin(2y), h \cos(2y), \ldots, h^{\frac{n}{2}} \sin(ny), h^{\frac{n}{2}} \cos(ny), \ldots)$$

$$\Downarrow$$

$$K(x, y) = \mathbf{z}(x) \cdot \mathbf{z}(y) = 1 + \sum_{i=1}^{\infty} h^n \sin(nx) \sin(ny) + \sum_{i=1}^{\infty} h^n \cos(nx) \cos(ny) =$$

$$= \frac{1}{2\pi} \frac{1 - h^2}{1 - 2h \cos(x - y) + h^2}$$

Admissible kernels

A function $K(\mathbf{x}, \mathbf{y})$ can represent a scalar product in the feature space if

$$\int \int d\mathbf{x} d\mathbf{y} \; K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) > 0 \quad \forall g \in L_2(R^d)$$

The corresponding features are the eigenvectors of an integral operator associated to K:

$$\int d\mathbf{x} \; K(\mathbf{x}, \mathbf{y}) \phi_i(\mathbf{x}) = \lambda_i \phi_i(\mathbf{x})$$

The kernel function has the following expansion in terms of the ϕ_i:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{y})$$

Admissible features

If we find a set of functions $\{\phi_i\}_{i=1}^{\infty}$ such that

$$\int d\mathbf{x}\ K(\mathbf{x}, \mathbf{y})\phi_i(\mathbf{x}) = \lambda_i \phi_i(\mathbf{x})$$

then the choice of features:

$$\mathbf{z}(\mathbf{x}) = \left(\sqrt{\lambda}_1 \phi_1(\mathbf{x}), \ldots \sqrt{\lambda}_n \phi_n(\mathbf{x}), \ldots\right)$$

is "admissible" in the sense that the scalar product is easy to compute:

$$\mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$$

A pleasant surprise

Here are some admissible K:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}}$$

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^n$$

$K(\mathbf{x}, \mathbf{y}) = \tanh(\mathbf{x} \cdot \mathbf{y} + \sigma)$ (only for certain values of σ).

From these 3 choices we can obtain different decision surfaces: Radial Basis Functions, polynomials and Multilayer Perceptrons!

Example

When the kernel is $K(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}}$, with σ given, the resulting classifier has the form:

$$f(\mathbf{x}) = \theta \left(\sum_{i \in SV} y_i h_i e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\sigma^2}} + b \right)$$

where SV is the set of support vectors.

This is a Radial Basis Functions classifier, where the centers are the support vectors, and the coefficients are the Lagrange multipliers (except for the sign).

Example

When the kernel is $K(\mathbf{x}, \mathbf{y}) = \tanh(\mathbf{x} \cdot \mathbf{y} + \sigma)$ (where σ is fixed) the resulting classifier has the form:

$$f(\mathbf{x}) = \theta \left(\sum_{i \in SV} y_i h_i \tanh(\mathbf{x} \cdot \mathbf{x}_i + \sigma) + b \right)$$

This is a (special case) Multilayer Perceptron classifier, where the weights are given by the support vectors.

Example

When the kernel is $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^n$, for a given n, the resulting classifier has the form:

$$f(\mathbf{x}) = \theta \left(\sum_{i \in SV} y_i h_i (\mathbf{x} \cdot \mathbf{x}_i + 1)^n + b \right)$$

This is a (very sparse) polynomial classifier.

Training a support vector machine

Training a support vector machine requires the choice of few relevant quantities:

- the kernel function, that determines the shape of the decision surface;

- a "smoothing" parameter in the kernel function (variance of the gaussian, degree of the polynomial, ...);

- the constant C for the cost of the misclassification errors;

Support Vector Machines

- Training is equivalent to solve a quadratic programming problem with as many variables as data points;

- Decision surfaces can be, among others, Radial Basis Functions, polynomials and Multilayer Perceptrons;

- few parameters to set;

- Generalization performances can be estimated after training, counting the number of support vectors;

- provide as a side-effect a compact description of the data;

- Unlikely to over-fit: careful control of the VC-dimension and of the bias/variance trade-off;

- excellent performances on OCR and face detection problems;

Main difficulty

When the number of data points is large (> 2000) the quadratic programming problem (QP) becomes extremely difficult to solve with standard methods.

Data bases with 50,000 are not uncommon (OCR, face detection).

In these cases a "divide and conquer" strategy can be adopted, and instead of solving one huge QP problem, an appropriate sequence of smaller problems ($\approx 2,000$) is solved.

Experimental results on OCR

U.S. Postal service data base (zip codes): 7,300 training patterns and 2,000 test patterns.

Each pattern is a 16×16 image.

Classifier	*Raw error %*
Human performance	2.5
Decision tree, C4.5	16.2
Best two-layer neural network	5.9
Five-layer network (LeNet 1)	5.1

Type of SV classifier	*Parameters of classifier*	*Number of support vectors*	*Raw error %*
Polynomials	d = 3	274	4.0
RBF classifier	$\sigma^2 = 0.3$	291	4.1
Neural network	$b = 2$, $c = 1$	254	4.2

(From Vapnik, 1995, pp. 143-145)

An experimental finding

For the OCR data base, many support vectors seem to be same, independently of the choice of the kernel K.

	Polynomial	*RBF*	*NN*	*Common*
total # of SV	1677	1727	1611	1377
% of common SV	82	80	85	100

The support vectors seem to be a geometric property of the data base.

Intuitively the support vectors are points on the border between the two classes.