Class 10

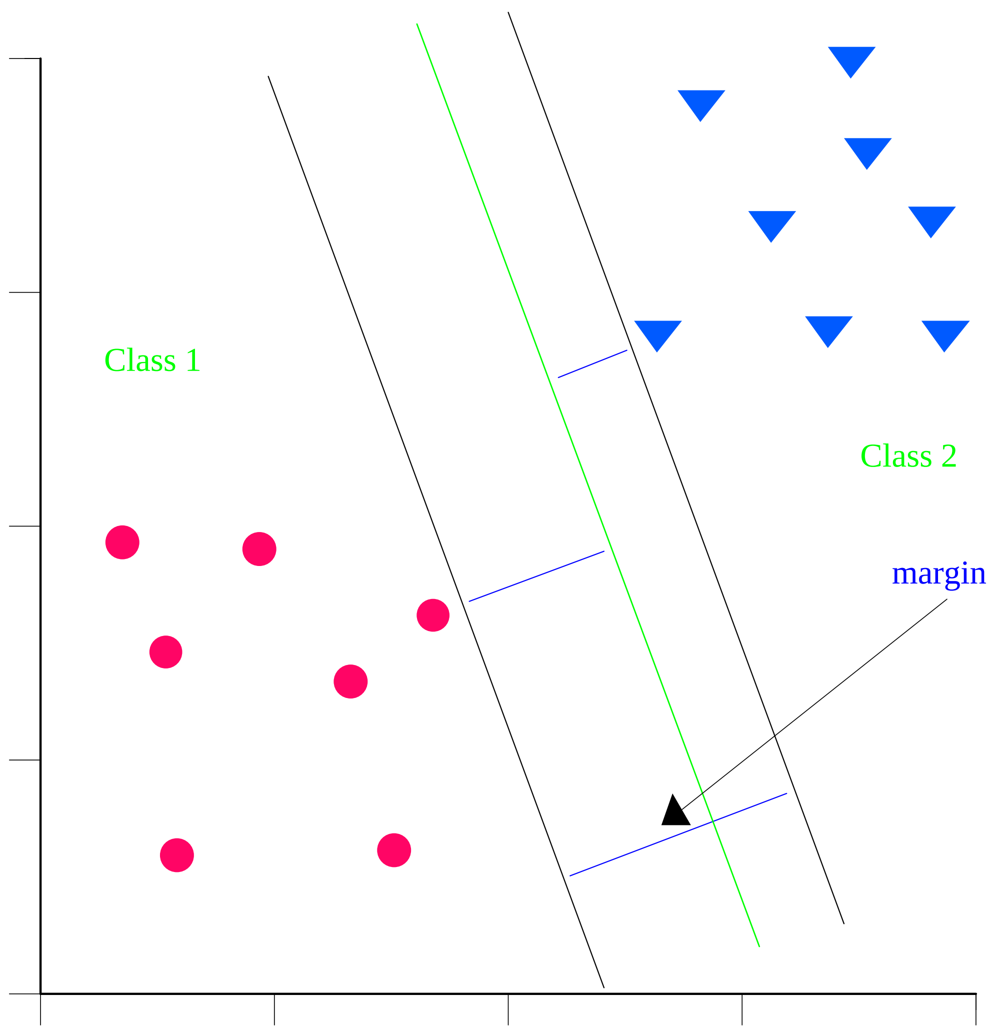
- Experimental results;

- SVM for regression;

- SVM and *sparse approximation*;

A new induction principle

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



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:

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

SVM minimizes an upper bound on the VC-dimension (maximizes the margin)

$$\Downarrow$$

SVM approximately implements the Structural Risk Minimization Principle.

Support Vector Machines

SVM find the optimal margin linear classifier in the *feature space*:

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

where ϕ_i are fixed functions.

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 ($\mathbf{x}$).

Support Vector Machines

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 $D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, where we have defined

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

Support Vector Machines

The final classifier has the form

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

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

The *kernel* K represents the scalar product in feature space.

Example: Polynomial of Degree 2 in 2 Variables

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

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

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

$$\Downarrow$$

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

Example: Polynomial of Degree 3 in 2 Variables

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

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

$$\Downarrow$$

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

Example: Infinite Dimensional Feature Space in One Variable

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

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

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

$$\Downarrow$$

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

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

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!

Training a support vector machine

Training a support vector machine involves the following steps:

- choice of the kernel function, that determines the shape of the decision surface;
- choice of the "smoothing" parameter in the kernel function (variance of the gaussian, degree of the polynomial, ...);
- choice of the constant C for the cost of the misclassification errors;
- solution of a QP problem in l variables.

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 in the feature space.

SVM for Face Detection

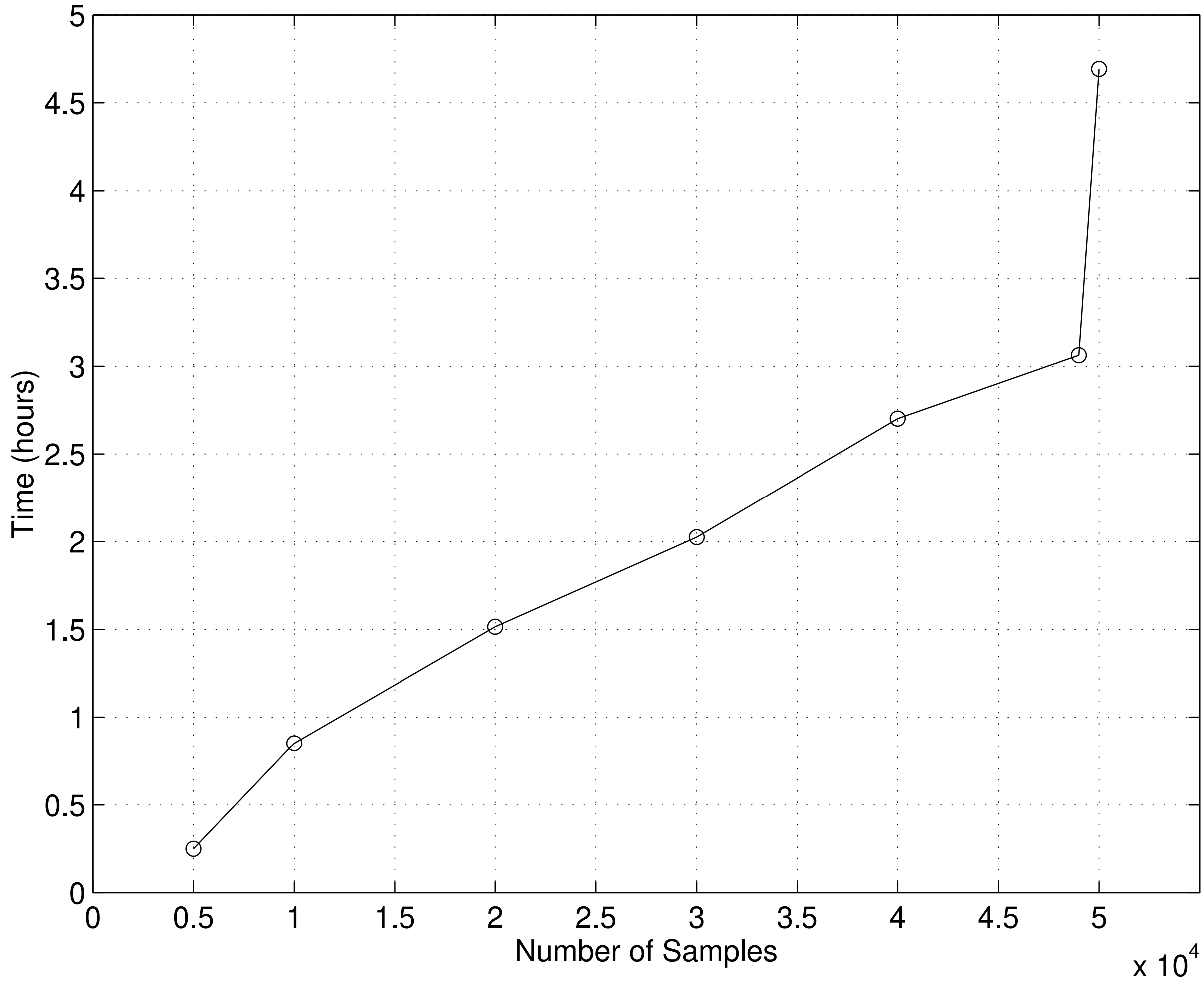
Data Base: up to 50,000 data points;

Kernel: polynomial of degree 2;

dimensionality of input space: 283

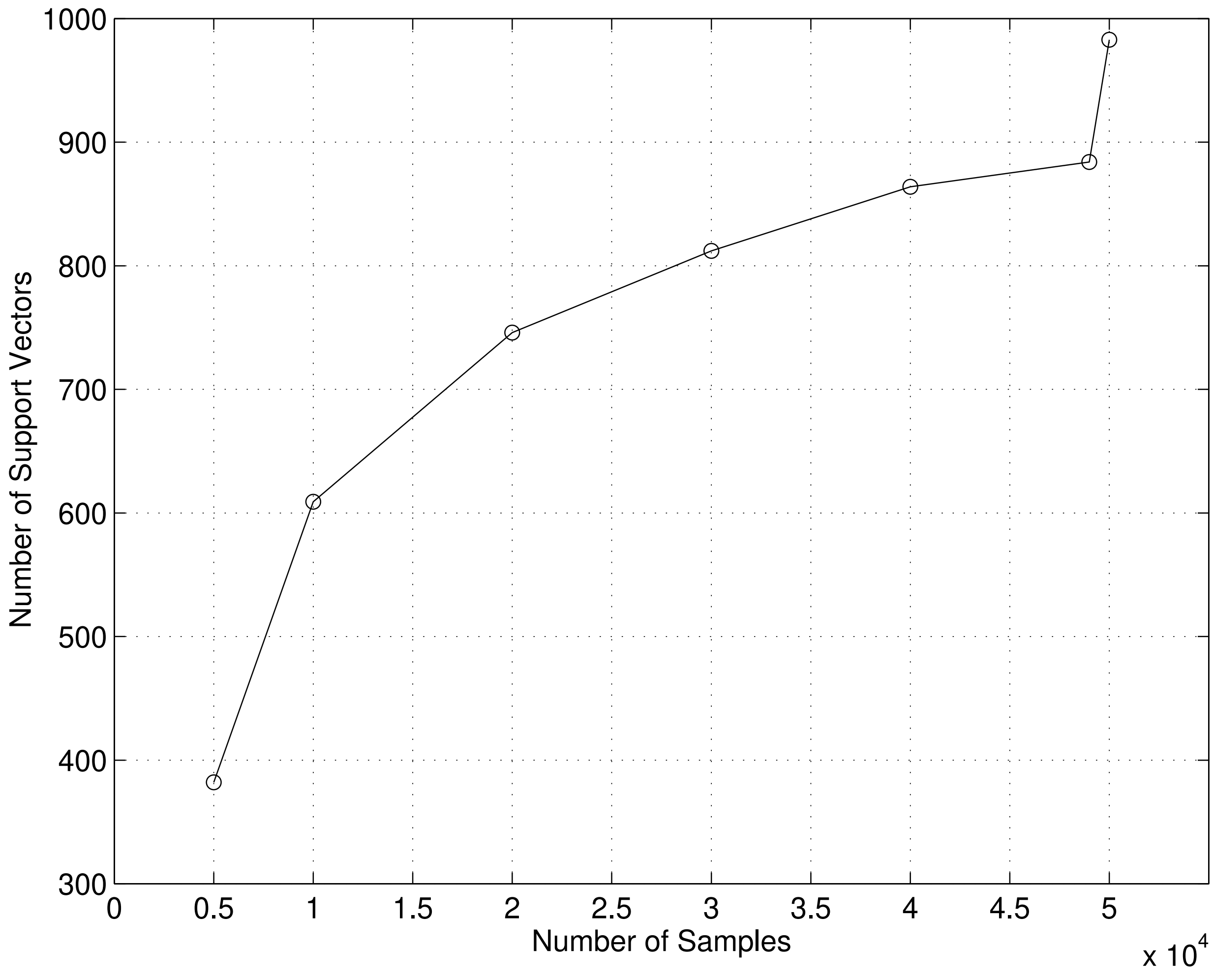
dimensionality of feature space: 40752

Training Time on a SPARC station-20 vs. number of samples



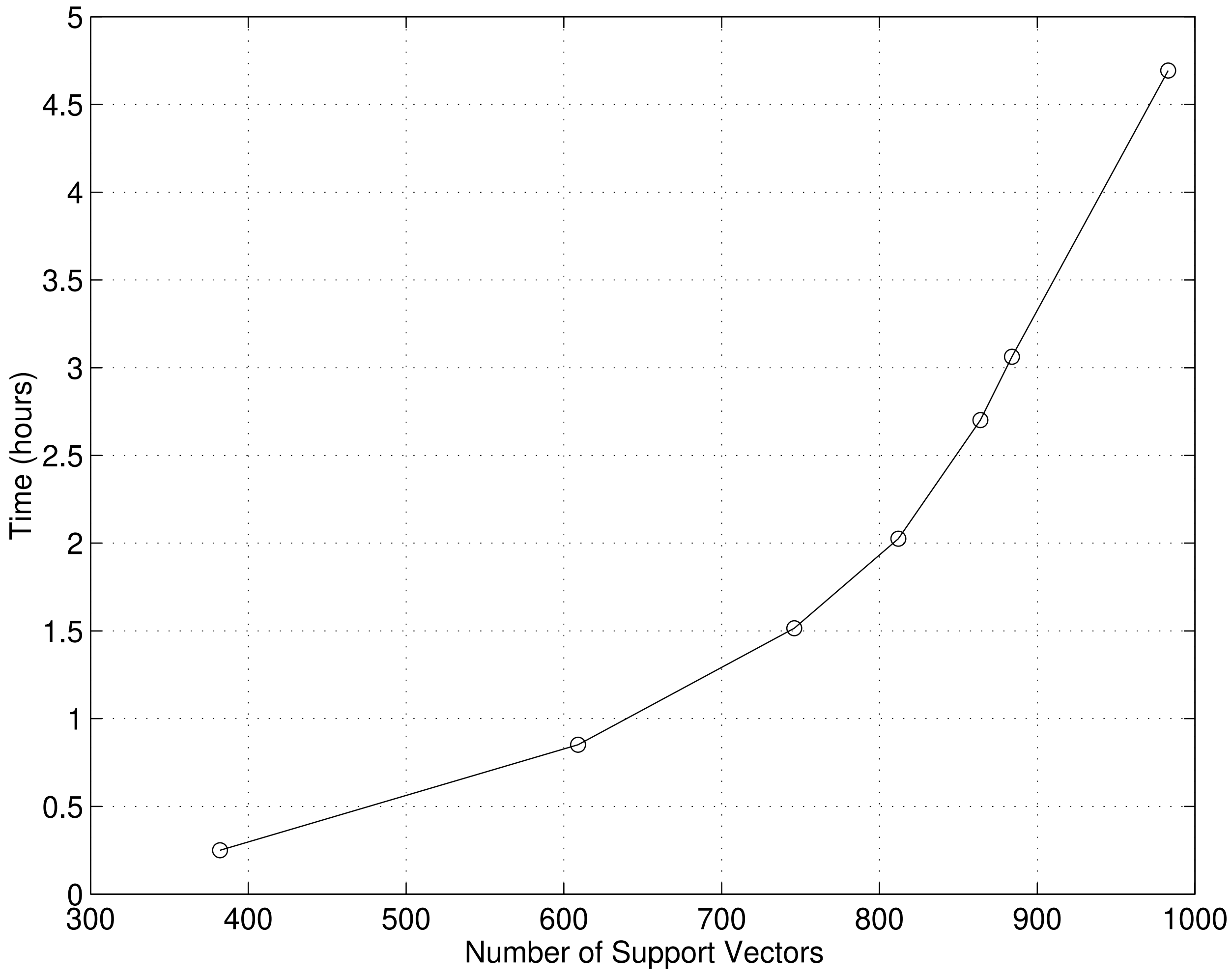
(E. Osuna, R. Freund and F. Girosi, 1997)

Number of Support Vectors vs. number of samples



(E. Osuna, R. Freund and F. Girosi, 1997)

Training Time on a SPARCstation-20 vs. Number of Support Vectors



(E. Osuna, R. Freund and F. Girosi, 1997)

A comment: bootstrapping and negative examples

- Train an initial SVM with the available data set;

- test the system on images *without* faces;

- collect the false positives and include these "negative examples" in the data base;

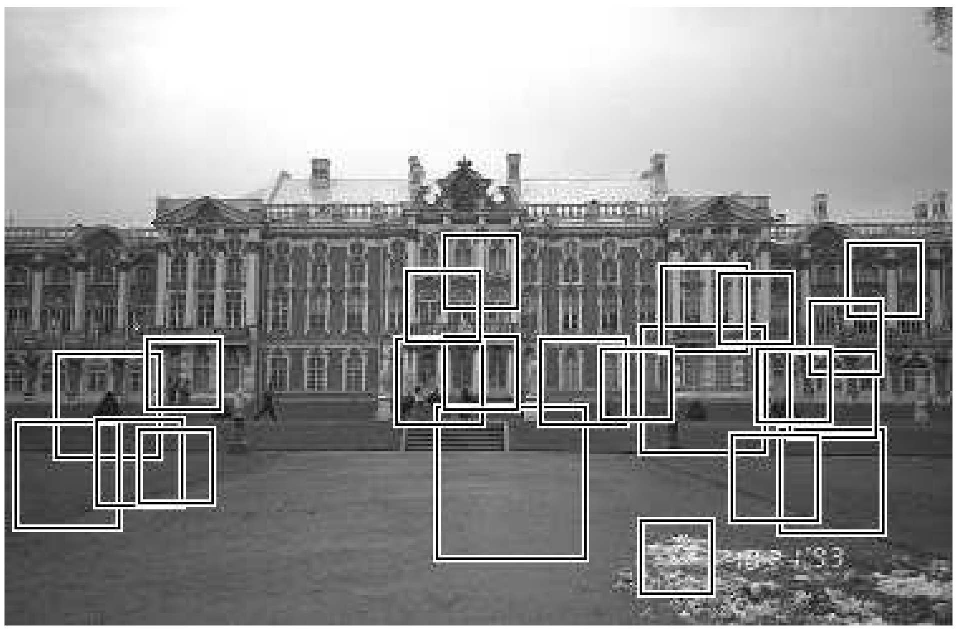
- Train SVM again;

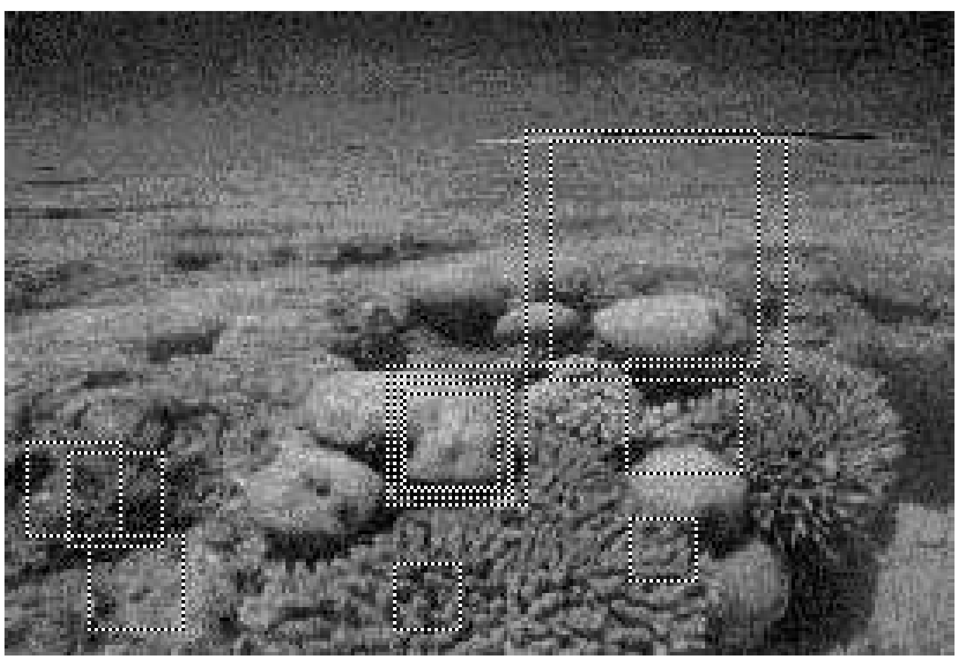
- (Possibly re-iterate this procedure)

This technique allows to collect "difficult" negative examples, that is non-faces that look like faces.

(Sung and Poggio, 1995)

False Positives used for Bootstrapping





SVM regression: linear case

Linear approximation: $f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b$.

In order to implement the SRM principle we need a structure on the set of linear functions:

$$S_n \equiv \{f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b \mid \ \|\mathbf{w}\|^2 \leq c_n\}$$

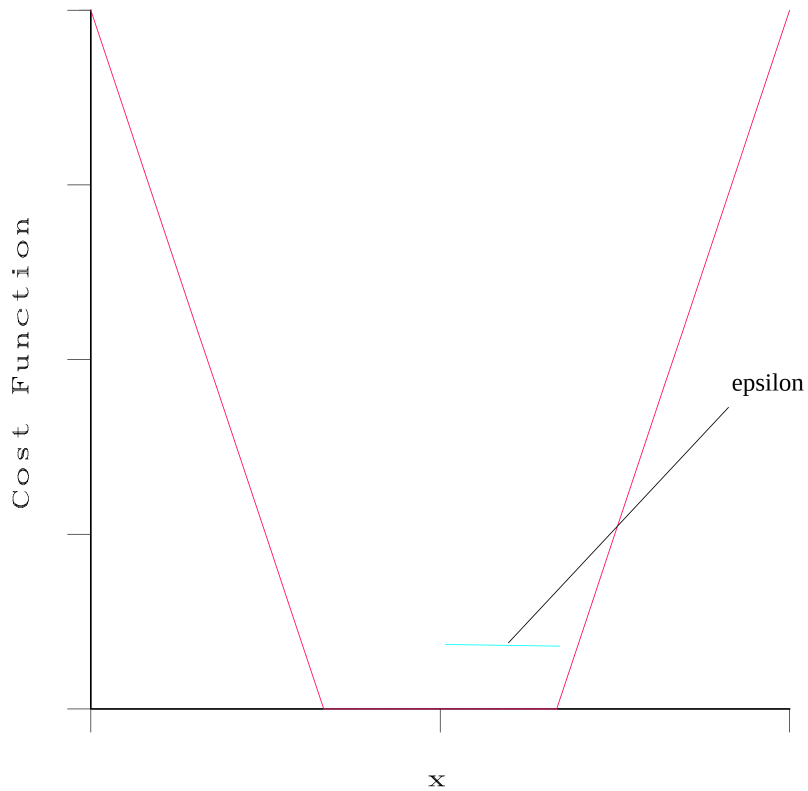
where $\{c_n\}$ is a monotonically increasing function of n (which does not need to be an integer).

SVM regression: linear case

A new empirical risk is defined:

$$I_{\text{emp}}^{(\epsilon)}(\mathbf{w}, b) = \frac{1}{l}\sum_{i=1}^{l} |y_i - \mathbf{x}_i \cdot \mathbf{w} - b|_\epsilon$$

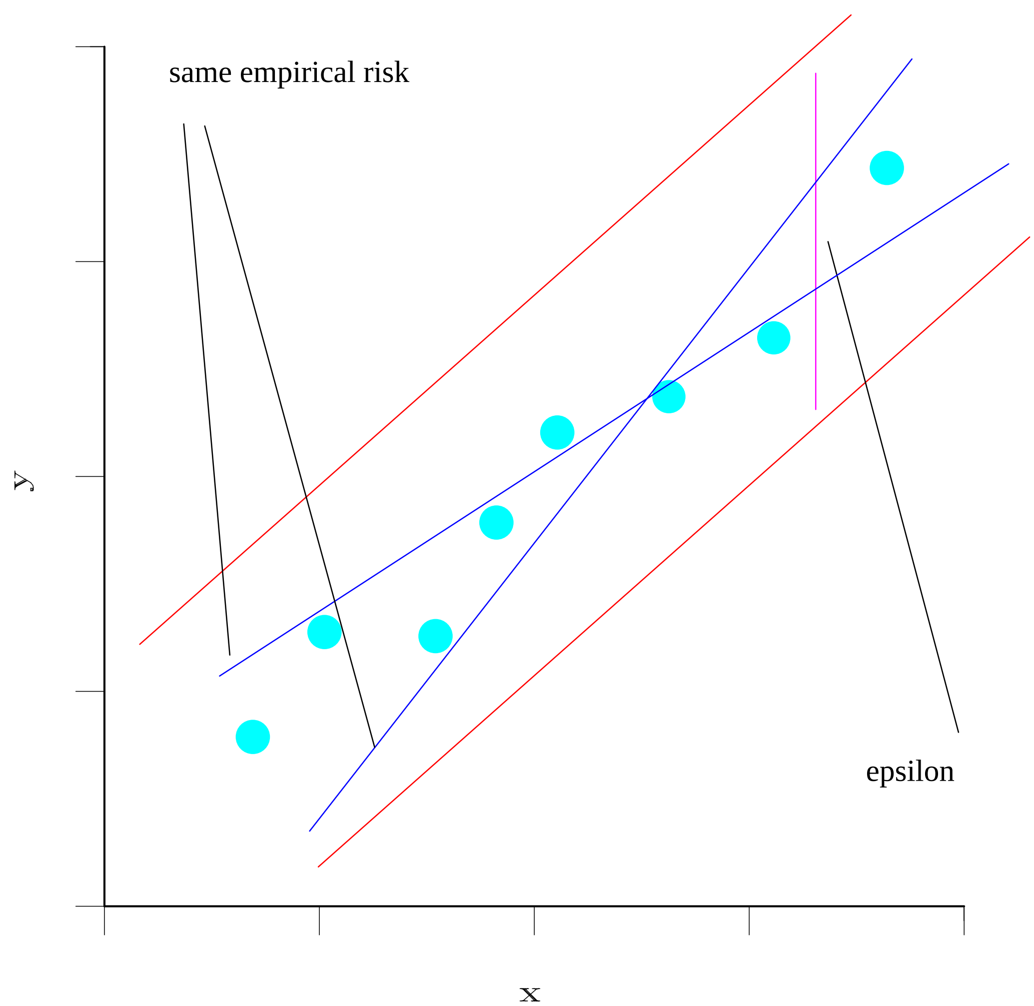
where the function $|x|_\epsilon$ is plotted below.



SVM regression: linear case

The new empirical risk $I_{\text{emp}}^{(\epsilon)}$ is called ϵ-*insensitive loss function*, and it resembles some cost functions that are used in *robust regression* to achieve robustness against outliers.

Errors which are smaller than ϵ do not count.



The empirical risk of the two linear hypothesis is the same.

SVM regression: linear case

The idea underlying SVM for regression is to minimize the empirical risk $I_{\mathrm{emp}}^{(\epsilon)}$ and $\|\mathbf{w}\|^2$ at the same time.

SVM regression: linear case

The minimization problem to be solved is the following:

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

Subject to the constraints:

$$\begin{array}{rcll}
y_i - \mathbf{w}\cdot\mathbf{x}_i - b & \leq & \epsilon + \xi_i^* & \quad i = 1,\ldots,l \\
\mathbf{w}\cdot\mathbf{x}_i + b - y_i & \leq & \epsilon + \xi_i & \quad i = 1,\ldots,l \\
\xi_i & \geq & 0 & \quad i = 1,\ldots,l \\
\xi_i^* & \geq & 0 & \quad i = 1,\ldots,l
\end{array}$$

SVM regression: linear case

It can be shown that the optimal vector $\mathbf{w}$ has the form

$$\mathbf{w}_0 = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)\mathbf{x}_i$$

where the coefficients α_i and α_i^* solve a QP problem with linear inequality constraint. The solution is therefore

$$f(\mathbf{x}) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)\mathbf{x}_i \cdot \mathbf{x} + b$$

As in the classification case *only some of the coefficients* $(\alpha_i^* - \alpha_i)$ *will be different from zero, and the corresponding data points are called support vectors.*

SVM regression: linear case

The coefficients α_i and α_i^* are found by minimizing the following cost function:

$$W(\alpha, \alpha^*) = \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i (\alpha_i^* - \alpha_i) +$$

$$+ \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to the constraint

$$\sum_{i=1}^{l} \alpha_i = \sum_{i=1}^{l} \alpha_i^*$$

$$0 \leq \alpha_i^* \leq C \ , \qquad\qquad i = 1, \ldots, l$$

$$0 \leq \alpha_i \leq C \ , \qquad\qquad i = 1, \ldots, l$$

SVM regression: nonlinear case

In order to obtain nonlinear approximating surfaces we do linear approximation in the feature space:

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

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b \Rightarrow \sum_{i=1}^{\infty} w_i \phi_i(\mathbf{x}) + b$$

The solution is

$$f(\mathbf{x}) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) \mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}_i) + b$$

and using the representation of the scalar product in the feature space:

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{x}_i) + b}$$

SVM regression: nonlinear case

The coefficients α_i and α_i^* are found by minimizing the following cost function:

$$
\begin{aligned}
W(\alpha, \alpha^*) = \; & \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i (\alpha_i^* - \alpha_i) + \\
+ \; & \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned}
$$

subject to the constraint

$$\sum_{i=1}^{l} \alpha_i = \sum_{i=1}^{l} \alpha_i^*$$

$$0 \leq \alpha_i^* \leq C \;, \qquad\qquad i = 1, \ldots, l$$

$$0 \leq \alpha_i \leq C \;, \qquad\qquad i = 1, \ldots, l$$

SVM and Sparse Approximation

In the case of regression the SVM algorithm can be derived in a different way.

The principle of **Sparse Approximation** can be used to derive an approximation technique which coincides with SVM under certain conditions.

Redundant Dictionaries

Consider the following model:

$$f(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{N} c_i \phi_i(\mathbf{x})$$

where

- N is very large;
- $\{\phi_i\}_{i=1}^{N}$ is a **redundant dictionary**

Method of Frames (Daubechies)

Matching Pursuit (Mallat and Zhang)

Basis Pursuit (Chen, Donoho and Saunders)

Olshausen and Field

Sparse Approximation Principle

If the vocabulary is redundant there are probably many different linear combinations of basis functions that give the same empirical risk.

The *sparse approximation principle* suggests to choose the one with the *least number of basis functions*.

Sparse Approximation Principle

Consider the following model:

$$f(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{N} c_i \phi_i(\mathbf{x})$$

The coefficients can be chosen by minimizing the following cost function:

$$E[\mathbf{c}, \xi] = \|f(\mathbf{x}) - \sum_{i=1}^{N} c_i \xi_i \phi_i(\mathbf{x})\| + \epsilon \sum_{i=1}^{N} \xi_i$$

where $\xi \in \{0, 1\}$ and $\epsilon \geq 0$.

The set $\{\xi_i\}_{i=1}^{N}$ tells us which basis functions are used, and the term $\sum_{i=1}^{N} \xi_i$ penalizes models with large numbers of basis functions.

Sparse Approximation

$$\min_{\mathbf{c},\xi}\left[\|f(\mathbf{x})-\sum_{i=1}^{N}c_i\xi_i\phi_i(\mathbf{x})\|+\epsilon\sum_{i=1}^{N}\xi_i\right]$$

If $\xi \in \{0,1\}$ this problem is extremely difficult.

$\xi \in \{0,1\} \Rightarrow \xi \in R^{+}$: **easier!**

$\Downarrow$ (if the c_i are bounded)

$$\min_{\mathbf{c}}\left[\|f(\mathbf{x})-\sum_{i=1}^{N}c_i\phi_i(\mathbf{x})\|+\epsilon\|\mathbf{c}\|_{L_1}\right]$$

(Chen, Donoho and Saunders, 1995)

(Olshausen and Field, 1996)

Sparse Approximation and SVM

Let us choose the following model:

$$f(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{N} c_i K(\mathbf{x} - \mathbf{x}_i)$$

where K is some positive definite kernel function.

Let us assume that we know the values of the unknown function f at the data points: $f(\mathbf{x}_i) = y_i$, and that f has zero mean. Let us impose the following extra constraint:

- The model $f(\mathbf{x}, \mathbf{c})$ has zero mean;

Reproducing Kernel Hilbert Space (RKHS)

a RKHS is an Hilbert space with a scalar product $< \cdot, \cdot >_H$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that

$$f(\mathbf{x}) =< f(\mathbf{y}), K(\mathbf{x}, \mathbf{y}) >_H$$

Example:

$$< f(\mathbf{x}), g(\mathbf{x}) >_H = \int_{R^d} d\mathbf{s} \, \frac{\tilde{f}(\mathbf{s})\tilde{g}^*(\mathbf{s})}{\tilde{K}(\mathbf{s})}$$

where K is positive definite (so that $\tilde{K}$ is positive, symmetric, and tends to zero). It is easy to see that

$$f(\mathbf{x}) =< f(\mathbf{y}), K(\mathbf{x} - \mathbf{y}) >_H$$

Sparse Approximation and SVM

Let us consider the following problem:

$$\min_{\mathbf{c}} \left[\frac{1}{2} \| f(\mathbf{x}) - \sum_{i=1}^{N} c_i K(\mathbf{x} - \mathbf{x}_i) \|_H^2 + \epsilon \|\mathbf{c}\|_{L_1} \right]$$

where $\|\cdot\|$ is the norm in the RKHS associated to K. The problem can be rewritten as:

$$\min_{\mathbf{c}} \left[- \sum_{i=1}^{N} c_i y_i + \frac{1}{2} \sum_{i,j=1}^{N} c_i c_j K(\mathbf{x}_i - \mathbf{x}_j) + \epsilon \sum_{i=1}^{N} |c_i| \right]$$

the "zero mean" condition implies that

$$< f(\mathbf{x}, \mathbf{c}), 1 >_H = \sum_{i=1}^{N} c_i = 0$$

Sparse Approximation and SVM

Let us observe that any vector **c** can be written as

$$\mathbf{c} = \alpha - \alpha^*$$

with

$$\alpha_i, \alpha_i^* \geq 0 \quad \forall i = 1, \ldots, N$$

$$\alpha_i \alpha_i^* = 0 \ \ \forall i = 1, \ldots, N$$

so that

$$\|\mathbf{c}\|_{L_1} = \sum_{i=1}^{N} (\alpha_i^* + \alpha_i)$$

Sparse Approximation and SVM

The minimization problem that has to be solved is:

$$W(\alpha, \alpha^*) = \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i(\alpha_i^* - \alpha_i) +$$
$$+ \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to the constraint

$$\sum_{i=1}^{l} \alpha_i = \sum_{i=1}^{l} \alpha_i^*$$

$$0 \leq \alpha_i^*, \qquad i = 1, \ldots, l$$

$$0 \leq \alpha_i, \qquad i = 1, \ldots, l$$

SAME AS SVM IN THE NOISELESS CASE!

Sparse Approximation and SVM

SVM give an approximating function that is a linear superposition of basis functions, one for each support vector.

When the kernel function K of the SVM is positive definite, like the Gaussian, the same solution can be obtained by looking for a Radial Basis Functions approximation that uses as centers a subset of the examples, and (approximately) minimizes the number of centers.

In this case, therefore

SVM $\equiv$ Sparse Approximation

Regularization Theory and SVM

To Be Written