An introduction to Support Vector Machines

Giorgio Valentini

Dipartimento di Scienze dell' Informazione, Università di Milano

e-mail: valenti@disi.unige.it

Outline

- Linear classifiers and maximal margin classifiers
- SVM as maximal margin classifiers
- The primal and dual optimization problem associated to SVM algorithm
- Support vectors and sparsity of the solutions
- Relationships between VC dimension and margin
- Soft margin SVMs
- Non-linear SVMs and kernels
- Comparison SVM - MLP
- Implementation of SVMs
- On line sw, tutorial and books on SVMs.

Goal of learning

Considering, for instance a 2-class classification problem:

Given $\mathcal{Z} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^N, y_i \in \mathcal{Y} = \{-1, 1\}$
we want to estimate a function $f : \mathbb{R}^N \to \mathcal{Y}$ in order to minimize the *expected risk*:

$$R[f] = \int L(f(\mathbf{x}), y) dP(\mathbf{x}, y)$$

But usually $P(\mathbf{x}, y)$ is unknown, and we try to minimize a function *close* to the expected risk, using the available data, the *empirical risk*:

$$R_{emp}[f] = \frac{1}{n} \sum_{i}^{n} L(f(\mathbf{x}_i), y_i)$$

Avoid overfitting

Giving some conditions on the learning machine we have that if $n \to \infty$ then *empirical risk* $\to$ *expected risk*.

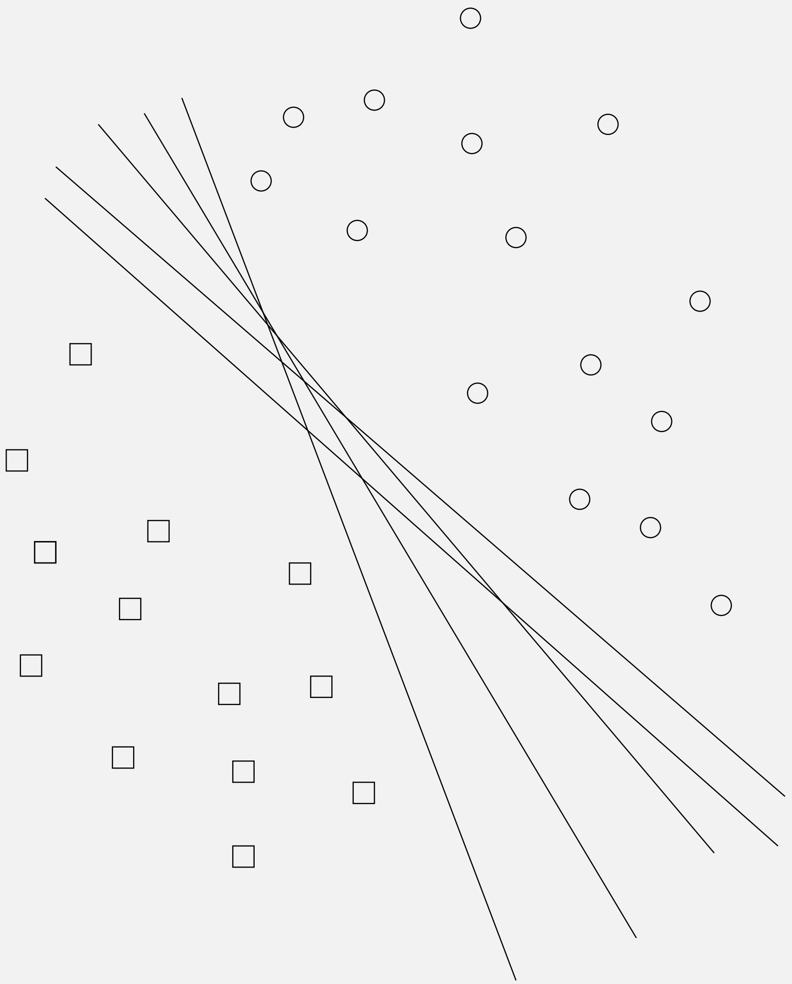
For small n instead overfitting might occur. To avoid this we could restrict the the complexity of the function class computable by the learning machine:

- regularization to limit the complexity of the functions
- choosing simple class of functions

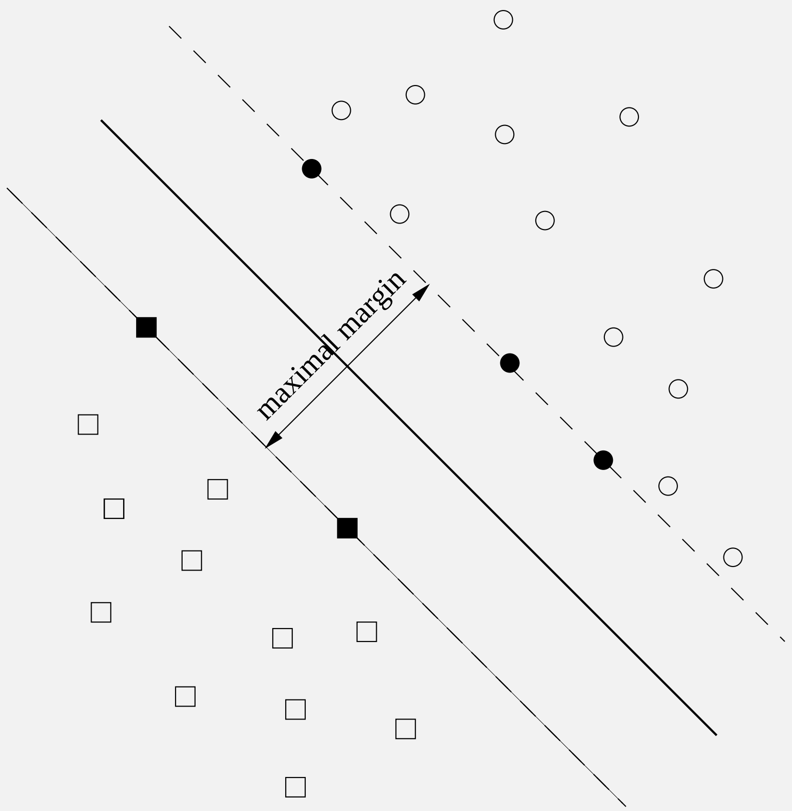
Example: choosing *linear functions*:

but underfitting might occur ...

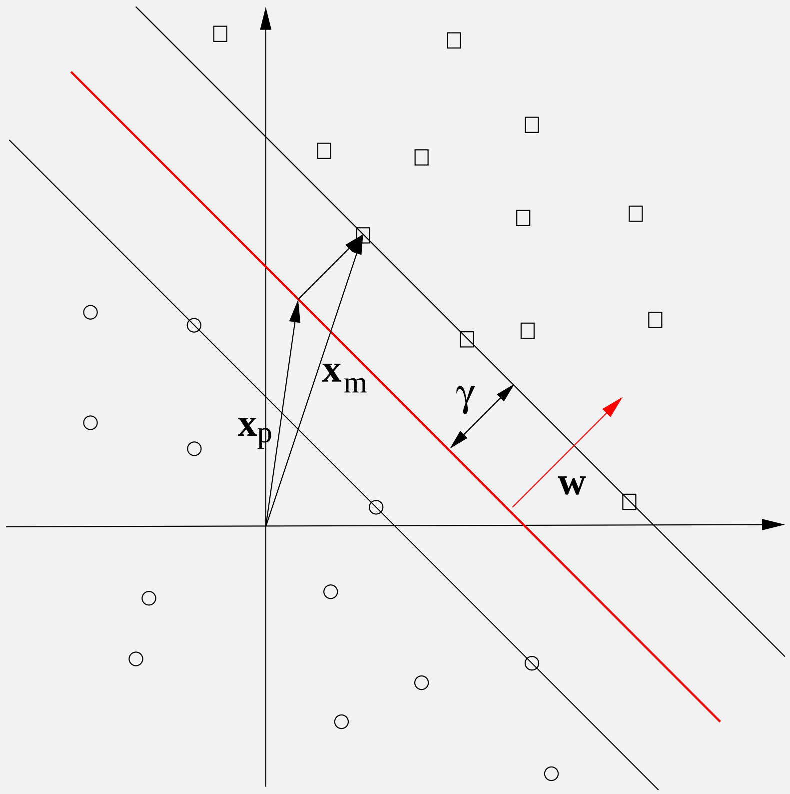
Which linear classifier?



Solution: the largest margin classifier



Functional and geometric margin - 1



Functional and geometric margin - 2

The linear classifier computes: $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$

For a point $\mathbf{x}_p$ on the separating hyperplane $f(\mathbf{x}_p) = \mathbf{w} \cdot \mathbf{x}_p + b = 0$

A point $\mathbf{x}_m$ on the margin whose width is γ can be expressed as

$$\mathbf{x}_m = \mathbf{x}_p + \frac{\mathbf{w}}{||\mathbf{w}||}\gamma$$

Then $f(\mathbf{x}_m) = \mathbf{w} \cdot \mathbf{x}_m + b = \mathbf{w} \cdot \mathbf{x}_p + \frac{\mathbf{w} \cdot \mathbf{w}}{||\mathbf{w}||}\gamma + b = \gamma||\mathbf{w}||$.

$\gamma||\mathbf{w}||$ is the *functional margin*.

The *geometric margin* is $\gamma = \frac{f(\mathbf{x}_m)}{||\mathbf{w}||}$

Functional and geometric margin - 3

To obtain the *canonical separating hyperplane* we need to normalize w.r.t the functional margin:

$f_c(\mathbf{x}) = \frac{f(\mathbf{x})}{\gamma||\mathbf{w}||}$

The *canonical functional margin* is $f_c(\mathbf{x}_m) = \frac{f(\mathbf{x}_m)}{\gamma||\mathbf{w}||} = 1$

The *canonical margin* is $\gamma_c = \frac{1}{||\mathbf{w}||}$

From this point we consider only the canonical iperplane (that is the hyperplane with canonical margin $1/||\mathbf{w}||$.

Maximizing the margin

1. Maximize the margin $\gamma = \frac{1}{||\mathbf{w}||} \iff$ Minimize $||\mathbf{w}||$ or equivalently $\mathbf{w} \cdot \mathbf{w}$

2. We need also to correctly separate the examples:
   $\forall i, y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$

We need to solve a constrained quadratic optimization problem

The primal optimization problem

$$
\begin{aligned}
\text{Minimize} \quad & \mathbf{w} \cdot \mathbf{w} \\
\text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \\
& 1 \leq i \leq n
\end{aligned}
$$

The hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ that solves this quadratic oprimization problem is the *maximal margin iperplane* with margin $\gamma = \frac{1}{||\mathbf{w}||}$

Lagrangian associated with the primal optimization problem

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1)$$

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i = \mathbf{0} \\
\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} &= \sum_{i=1}^{n} y_i \alpha_i = 0
\end{aligned}$$

$$\begin{aligned}
\mathbf{w} &= \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i \\
0 &= \sum_{i=1}^{n} y_i \alpha_i
\end{aligned}$$

Building the dual

Putting the relations obtained into the primal we have:

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i \\
&= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)
\end{aligned}
$$

Obtaining the associated dual optimization problem

The dual optimization problem

Maximize $\Phi(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$

subject to $\sum_{i=1}^{n} y_i \alpha_i = 0$

$\alpha_i \geq 0, \quad 1 \leq i \leq n$

The hyperplane whose weight vector $\mathbf{w}^* = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$ solves this quadratic oprimization problem is the *maximal margin iperplane* with geometric margin $\gamma = \frac{1}{||\mathbf{w}||}$.

Note the input vectors $\mathbf{x}$ appear only as dot products.

Discriminant function computed by the SVM

The linear SVMs compute the family of linear functions:

$$\mathcal{F}(\mathbf{x}, \mathbf{w}, b) = \{\mathbf{x} \cdot \mathbf{w} + b, \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}\}$$

If α^* is the solution of the dual optimization problem then

- $\mathbf{w}^* = \sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i$ is the weight vector of the maximal margin hyperplane
- $f(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + b^* = \sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*$ is the corresponding discriminant function.
- The decision function $g : \mathbb{R}^n \to \{-1, +1\}$ is
  $g(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*)$

Support Vectors

Support vectors: the most difficult patterns to be learnt (patterns that lie on the margin):

$$\mathcal{SV} = \{\mathbf{x}_i | y_i(\sum_{j=1}^{n} y_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i + b) = 1, \quad 1 \leq i \leq n\}$$

Using the support vectors we can compute b:

If $\mathbf{x}_i$ is a support vector then

$$y_i(\sum_{j=1}^{n} y_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i + b) = 1 \quad \Rightarrow \quad b = 1/y_i - \sum_{j=1}^{n} y_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i$$

Considering all the support vectors we can get a more stable solution:

$$b = \frac{1}{|\mathcal{SV}|} \sum_{i \in \mathcal{SV}} (y_i - \sum_{j=1}^{n} y_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i)$$

The Karush-Kuhn-Tucker conditions - 1

KKT conditions are satisfied at the solution of any constrained optimization problem:

$$\begin{array}{lll} \text{Minimize} & f(\mathbf{w}), & \mathbf{w} \in \mathbb{R}^n \\ \text{subject to} & g_i(\mathbf{w}) \geq 0, & 1 \leq i \leq k \\ & h_i(\mathbf{w}) = 0, & 1 \leq i \leq m \end{array}$$

If the problem is convex (convex objective function with constraints which give a convex feasible region), KKT conditions are necessary and sufficient for $\mathbf{w}, \alpha, b$ to be a solution (see next).

The Karush-Kuhn-Tucker conditions - 2

If f is convex and g_i, h_i linear, necessary and sufficient conditions for $\mathbf{w}$ to be an optimal solution are the existence of α, b such that:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} &= \mathbf{0} & \\ \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} &= 0 & \\ \alpha_i g_i(\mathbf{w}) &= 0 & 1 \leq i \leq k \\ g_i(\mathbf{w}) &\geq 0 & 1 \leq i \leq k \\ \alpha_i &\geq 0 & 1 \leq i \leq k \end{aligned}$$

KKT conditions for SVM optimization imply sparsity of the solutions

From KKT conditions the optimal solutions $\mathbf{w}^*, \alpha^*, b^*$ must satisfy:

$$\alpha_i^* g_i(\mathbf{w}^*) = 0 \Rightarrow \alpha_i^*(y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1) = 0$$

If the i^{th} constraint is active, that is $g_i(\mathbf{w}^*) = 0 \Rightarrow \alpha_i^* \geq 0$

If the i^{th} constraint is inactive, that is $g_i(\mathbf{w}^*) > 0 \Rightarrow \alpha_i^* = 0$

Hence only if $g_i(\mathbf{w}^*) = 0 \Rightarrow \alpha_i^* y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) = 1$ we can have non zero α_i^*

Only for inputs $\mathbf{x}_i$ whose functional margin is 1 (*support vectors*) the corresponding α_i^* can be non zero

Sparsity of the solutions

Indeed

$$f(\mathbf{x}, \alpha^*, b) = \sum_{i=1}^{n} y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^* = \sum_{i \in \mathcal{SV}} y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*$$

Complexity and bounds on the expected risk

The concept of complexity of a function class can be captured by the VC (Vapnik-Chervonenkis) dimension.

The VC dimension measures how many points can be *shattered* (that is separated) for all possibile labelings

Theorem (Vapnik)
Let h denote the VC dimension of the function class $\mathcal{F}$ and R_{emp} the empirical risk for the 0/1 loss function. Then $\forall f \in \mathcal{F}$ and $\forall \delta > 0$ the following bound on the expected risk $R[f]$ holds with $P \geq 1 - \delta$ for $n > h$:

$$R[f] \leq R_{emp} + \sqrt{\frac{h(\ln \frac{2n}{h} + 1) - \ln(\delta/4)}{n}}$$

VC dimension and margin of the separating hyperplane

Theorem (Vapnik).
Consider hyperplanes $f(\mathbf{x}, \mathbf{w}, b) = \mathbf{x} \cdot \mathbf{w} + b = 0$ in canonical form, that is such that:

$$\min_{1 \leq i \leq n} |f(\mathbf{x}_i, \mathbf{w}, b)| = 1$$

Then the set of decision functions $g(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x}, \mathbf{w}, b)$ that satisfy the constraint $||\mathbf{w}|| < \Lambda$ has a VC dimension h satisfying:

$$h \leq R^2 \Lambda^2$$

where R is the smallest radius of the sphere around the origin containing all the training points.

Maximizing the margin is equivalent to minimize the VC dimension of linear classifiers

From the two previous Vapnik's theorems we have:

1. Maximizing the margin, that is equivalently minimizing $||\mathbf{w}||$ (recall $\gamma = 1/||\mathbf{w}||$) we minimize the VC dimension of the SVM.
2. The minimization of the expected risk depends on both minimizing the *empirical risk* and the *confidence interval*
3. The *confidence interval* depends mainly on the ratio h/n
4. SVM algorithm minimizes both the empirical risk and the confidence interval

$\Downarrow$

SVM directly implements the structural risk mimimization induction principle

SVM as maximal margin classifier

SVM maximizes margin and minimize empirical risk:

1. Low VC dimension (SVM implements the structural risk mimimization)
2. From KKT conditions → Sparsity of the solutions

But we considered only linearly separable data ...

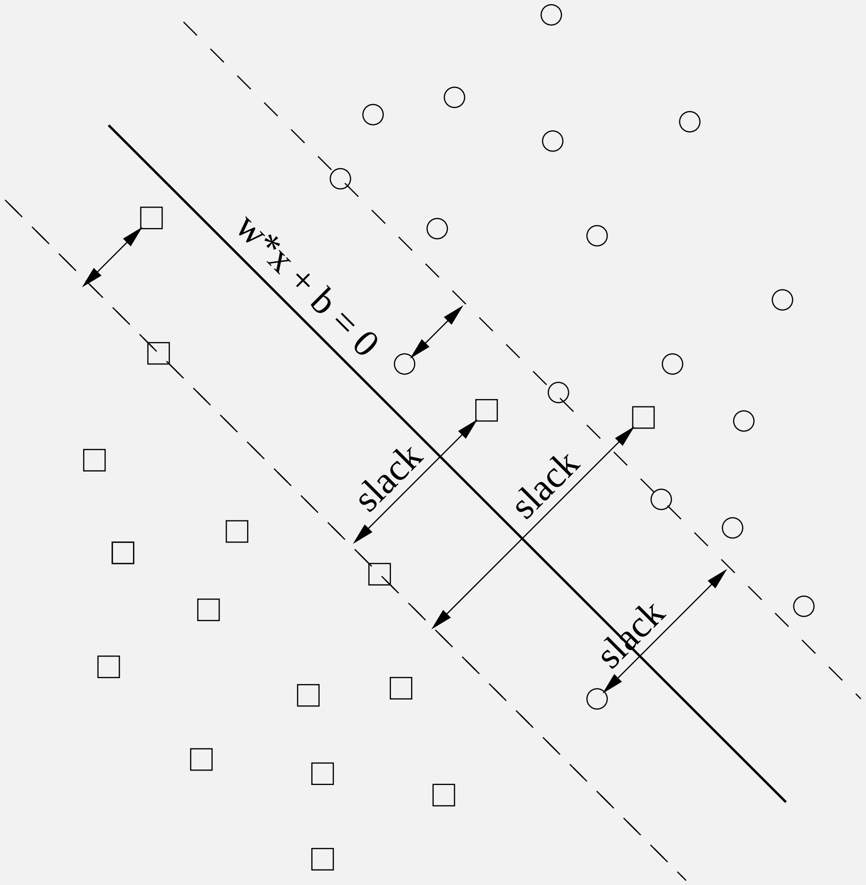
What about non linearly separable data ?

- Most of real data are not linearly separable
- Forcing separation of the data $\rightarrow$ overfitting
- Noisy data and outliers.

We need two more steps:

1. Soft margin SVM
2. Kernels

Soft margin SVM: introducing slack variables



The 1-Norm soft-margin SVM optimization problem

$$
\begin{array}{ll}
\text{Minimize} & \mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \\
 & \xi_i \geq 0 \\
 & 1 \leq i \leq n
\end{array}
$$

C controls the tradeoff between the accuracy w.r.t. to the training data and the maximization of the margin. It can be interpreted also as a regularization term.

The dual 1-Norm soft-margin optimization problem

$$
\begin{aligned}
\text{Maximize} \quad & \Phi(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\
\text{subject to} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0 \\
& 0 \leq \alpha_i \leq C, \quad 1 \leq i \leq n
\end{aligned}
$$

The hyperplane whose weight vector $\mathbf{w}^* = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$ solves this quadratic oprimization problem is the *soft margin hyperplane* with slack variable ξ_i defined w.r.t the geometric margin $\gamma = \frac{1}{||\mathbf{w}||}$.

Box constraints

The 1-Norm soft-margin optimization problem is *equivalent* to that of the maximal margin hypeplane *with the additional constraint* $\alpha_i \leq C$ (*Box constraints*).

This approach limits the effect of outliers (for which α_i tends to be very large).

In this case, from KKT conditions we have:

$$
\begin{aligned}
\alpha_i = 0 \quad &\Rightarrow \quad y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} \quad \xi_i = 0 \\
0 < \alpha_i < C \quad &\Rightarrow \quad y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0 \\
\alpha_i = C \quad &\Rightarrow \quad y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} \quad \xi_i \geq 0
\end{aligned}
$$

Kernels

The basic idea:

Applying a simple linear algorithm in a projected high dimensional space:

- It is more likely to achieve a linear separation a high dimensional space (*Cover's theorem*).
- SLT tells us that we can defeat the *curse of dimensionality* problem if we use *simple* class of decision rules.

Non-linear mapping in Kernel Feature Spaces

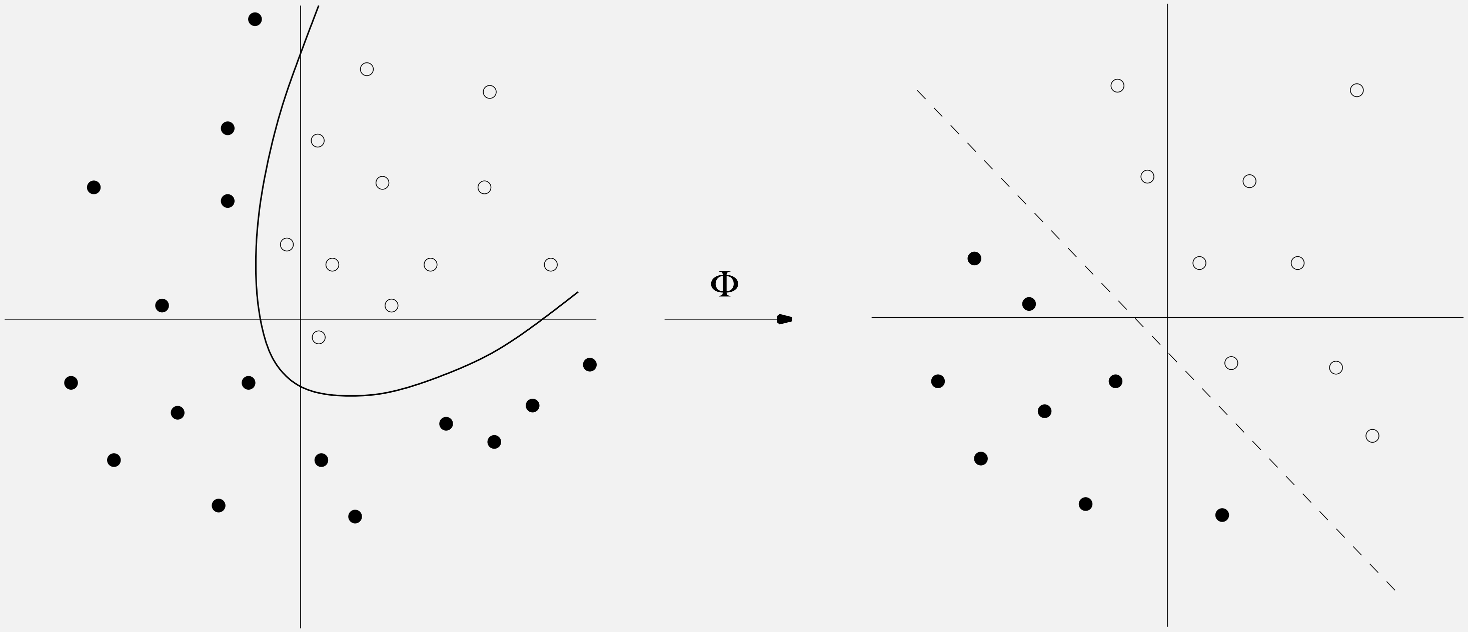
$$\Phi : \mathbb{R}^d \rightarrow \mathcal{F} \subset \mathbb{R}^m$$

where it can be that the dimensionality $d << m$.

For a given learning problem (e.g. classification) one considers the same algorithm in $\mathcal{F}$ instead of $\mathbb{R}^n$, working with examples $\{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_n), y_n)\} \in \mathcal{F} \times Y$.

Then in $\mathcal{F}$ the maximal margin hyperplane or the soft-margin algorithm is applied.

Patterns non-linearly separable in Input Space can be separable in Feature Space



Kernel examples

See kernel-ex.pdf

Characterization of kernels

If $K(\mathbf{x}, \mathbf{x}')$ is a symmetric function satisfing *Mercer's conditions*, that is:

$$\int\int K(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

for all f, $\int f^2(\mathbf{x}) d\mathbf{x} < \infty$

then we can expand $K(\mathbf{x}, \mathbf{x}')$ in a some inner product feature space:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{\infty} \lambda_j \phi(\mathbf{x}) \phi(\mathbf{x}')$$

Non-linear SVMs - 1

Note that in the dual representation of SVMs the inputs appears only in a dot-product form:

$$\begin{array}{ll} \text{Maximize} & \Phi(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^{n} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad 1 \leq i \leq n \end{array}$$

The discriminant function obtained from the solution is

$$f(\mathbf{x}, \alpha^*, b) = \sum_{i=1}^{n} y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*$$

We can substitute the dot-products int the input space with a kernel function.

Non-linear SVMs - 2

Maximize $\Phi(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i \mathbf{x}_j)$

subject to $\sum_{i=1}^{n} y_i \alpha_i = 0$

$0 \leq \alpha_i \leq C, \quad 1 \leq i \leq n$

The discriminant function obtained from the solution is

$$f(\mathbf{x}, \alpha^*, b) = \sum_{i=1}^{n} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^*$$

The SVM receives as inputs pattern $\mathbf{x}$ in the input space, but works in a high dimensional (possibly infinite) feature space:

1. No computational burden due to high dimensional vector
2. Not dimensionality, but complexity of the function class matters

Architecture of SVMs

See SVM-arch.pdf

Comparison SVM-MLP

1. In MLPs complexity is controlled by keeping number of hidden nodes small, while in SVMs complexity is controlled independently of dimensionality.
2. In non-linear SVMs the decision surface is constructed in a very high (often infinite) dimensional space. However, the curse of dimensionality is avoided by using the notion of an inner product kernel and optimising the weights in the input space. In MLPs the "dimensionality of the feature space" is limited to the number of hidden units in the hidden layer. Weight decay and regularization can be viewed as a sort of complexity control.
3. SVMs can use the theory of VC dimension and structural risk minimisation and can get bounds on the generalization error.
4. Finding the weights is a quadratic programming with global minimum.

Thus the algorithm is efficient and SVMs generate near optimal classification and are insensitive to overtraining.

5. Obtain good generalisation performance due to high dimension of feature space
6. SVM automatically computes network parameters for that kernel. E.g. RBF SVM: automatically selects the number and position of hidden nodes (and weights and bias).
7. Slow training (compared to RBFNs/MLPs) due to computationally intensive solution to QP problem especially for large amounts of training data $\Rightarrow$ need special algorithms.
8. Generates more complex solutions w.r.t. MLPs, especially for large amounts of training data.

Implementation of SVMs

To solve the SVM problem one has to solve a convex quadratic programming (QP) problem.

Unfortunately the dimension of the Kernel matrix is equal to the number of input patterns.

For large problems (e.g. $n = 10^5$) we need to store in memory 10^{10} elements, and so standard QP solver cannnot be used.

Exploiting the sparsity of the solutions and using some heuristics several approaches have been proposed to overcome this problem:

1. Chunking (Vapnik, 1982)
2. Decompostion methods (Osuna et al., 1996)
3. Sequential Minimal Optimization (Platt, 1999)
4. Others ...

Software for SVM classification and regression

A large collection is available from `http://www.kernel-machines.org`.

In particular:

1. SVM-light (Joachims, Cornell, USA):
   `http://svmlight.joachims.org`
2. SVMFu (Rifkin, MIT, USA):
   `http://five-percent-nation.mit.edu/SvmFu/index.html`
3. LIBSVM (Chung and Lin):
   `http://www.csie.ntu.edu.tw/ cjlin/libsvm`
4. SVM-Torch (Collobert, IDIAP, CH):
   `http://www.idiap.ch/learning/SVMTorch.html`

Tutorials on SVMs

C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*2, p.121-167, 1998.

A. Smola and B. Scholkopf. A tutorial on support vector regression. NeuroColt2 TR 1998-03, 1998.

C. Campbell. An Introduction to Kernel Methods. In *Radial Basis Function Networks: Design and Applications* . R.J.Howlett and L.C.Jain (eds). Springer Verlag, Berlin, 2000.

K. Muller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopk. An Introduction to Kernel-Based Learning Methods *IEEE Transactions on Neural Networks*, 12(2), 2001

Books on SVMs

Introductory books :

N.Cristianini and J. Shawe Taylor. *An Introduction to Support Vector Machines and other Kernel Based Learning Methods* . Cambridge University Press, 2000

Ralf Herbrich. *Learning Kernel Classifiers.* MIT Press, Cambridge, MA, 2002

Bernhard Schoelkopf and Alex Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002

Chapters on SVMs in classical machine learning books :

V.Vapnik. *The nature of Statistical Learning* . Springer, 1995 (Chap.5)

S.Haykin. *Neural Networks* . Prentice Hall, 1999 (Chap.6 p. 318-350)

V.Cherkassky, F.Mulier. *Learning from data* . Wiley & Sons, 1998 (Chap.9 p. 353-387).

A classical paper on SVM :

C.Cortes and V.Vapnik. Support Vector Networks. *Machine Learning* 20, p.273-297, 1995

You can find more at `http://www.kernel-machines.org`.