Support vector machines

Václav Hlaváč

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Center for Machine Perception
http://cmp.felk.cvut.cz/~hlavac, hlavac@fel.cvut.cz

Courtesy: V. Franc.

Outline of the talk:

- Generative vs. discriminative classifier. Maximal margin classifier.
- Minimization of the structural risk.
- SVM, task formulation, solution: quadratic programming.
- Linearly separable case.
- Linearly non-separable case.
- Kernels.

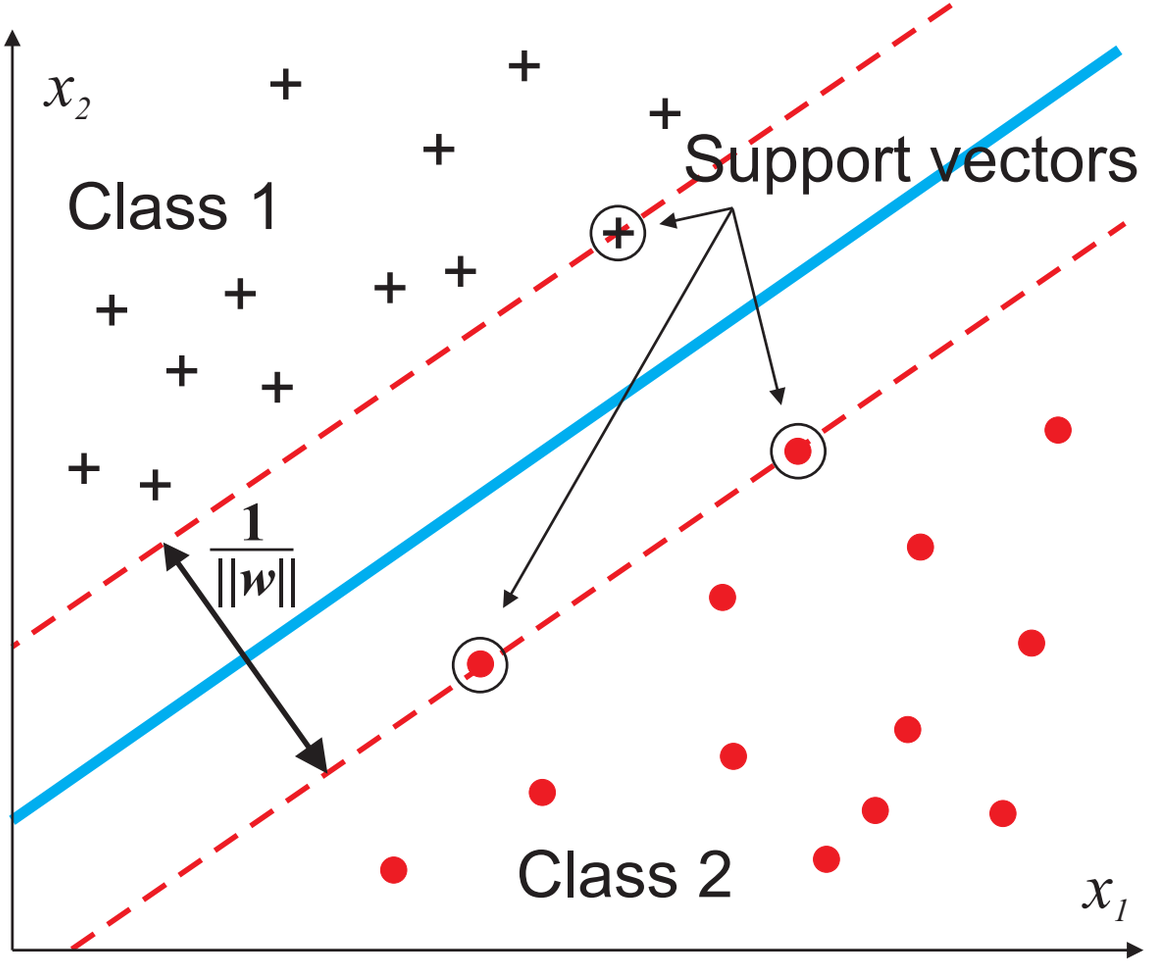
Introduction

- There are two principal approaches to the classifier design:
  - Generative. The statistical model describes the joint distribution $p(x, y)$. To learn it, all combinations of x, y have to be observed, which can be untractable.
  - Discriminative (also conditional). The conditional probability $p(y|x)$ or (in SVMs) $\log \frac{p(y=+1|x)}{p(y=+1|x)} \lessgtr \Theta$ is learned.
- So far, the generative approach was used. A known statistical model was assumed $\Rightarrow$ decision rule.
- Now, we will consider discriminative approach. We will assume that the class of decision rules is known and we have to choose (discriminate) one of them.

  V. Vapnik: "Learning is the selection of one decision rule from the class of rules".

Maximal margin classifier

- Maximizes margin between classes which increases generalization ability.
- The Vapnik's Support Vector Machine is based on the same idea.



Support vector machines, the task

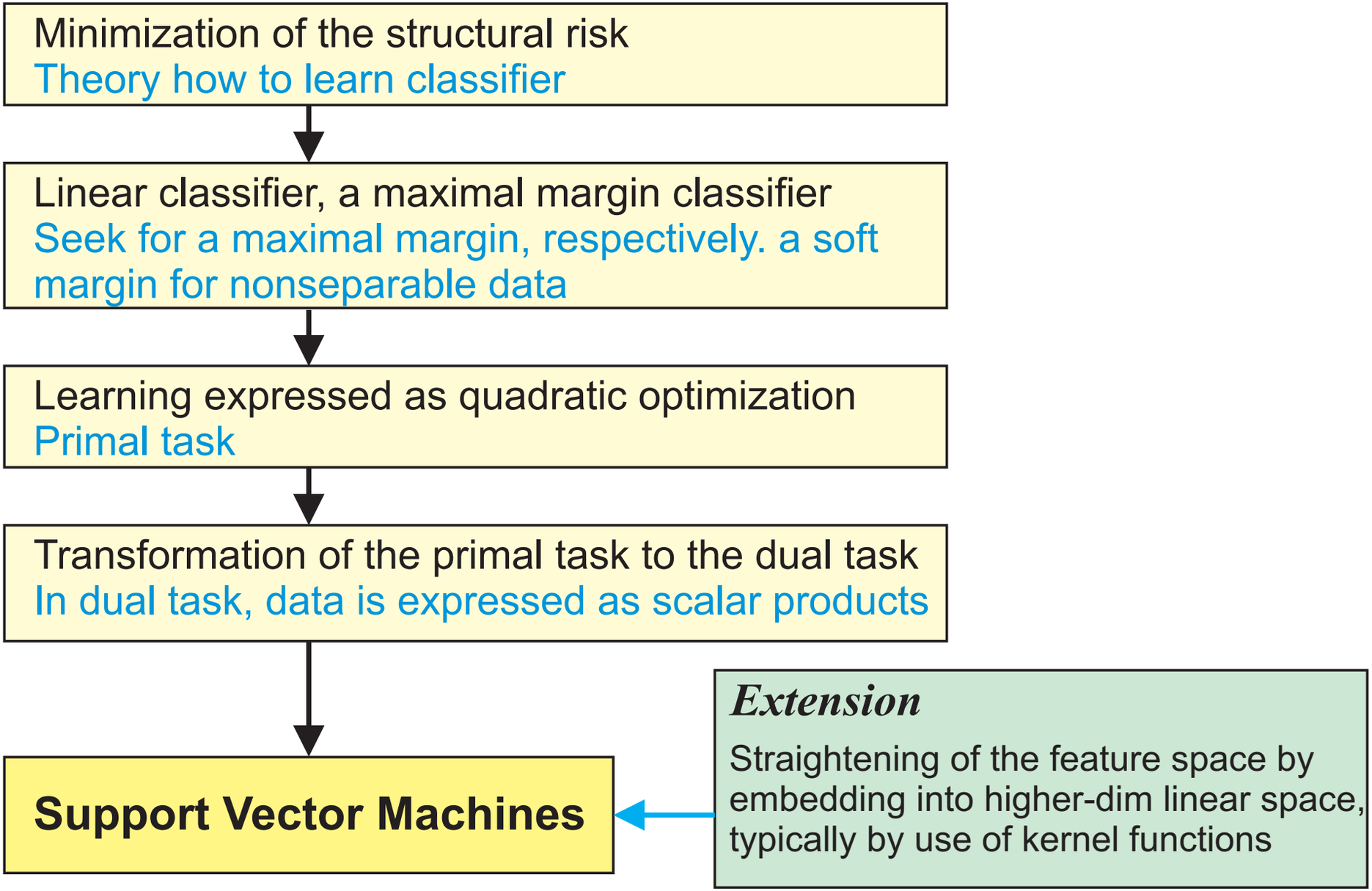
- Two hidden states (classes) only, y_1, y_2.
- A separable hyperplane is sought which maximizes a distance (margin) between classes.
- The task is converted into a quadratic programming task

$$(w^*, b^*) = \underset{w,b}{\operatorname{argmin}} \frac{1}{2} \|w\|^2$$

under constraints

$$\begin{aligned} \langle w, x_j \rangle + b \geq \quad 1 \qquad & \text{for} \quad y_j = 1 \\ \langle w, x_j \rangle + b \leq \quad -1 \qquad & \text{for} \quad y_j = -1 \end{aligned}$$

Support vector machines, a road map



Minimization of the structural risk (1)

Introduction

- The classifier is learnt from a finite training set.
- The statistical model $p(x, y)$ is unknown. Chervonenkis and Vapnik derived an upper bound on

$$\sum_x \sum_y p(x, y)(y \neq Q(x)) \; .$$

  which does not involve $p(x, y)$.
- The upper bound is provided which sums errors on the training set and the generalization error. When learning is performed it should minimize training error and also the complexity of the classifier has to be controlled.

Minimization of the structural risk (2)

Assumptions

- $x \in \mathbb{R}^n$ ... observation of the object (a vector of measurements).
- $y \in \{-1, 1\}$ ... hidden states. This notation leads to more compact derivations and formulas.
- There is a training set available

  $\{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)\}$,

  which is drawn randomly and generated by an unknown probability distribution $p(x, y)$.

Minimization of the structural risk (3)

The aim is to find a classifier (decision strategy) $q(x, \Theta)$,

where Θ is a parameter (usually vector of parameters) with the minimal expected classification error

$$R_{exp}(q(x,\Theta)) = \int \frac{1}{2} \left| y - q(x,\Theta) \right| \, \mathrm{d}\, p(x,y)$$

The simple approximation of R_{exp} is the empirical risk R_{emp},

$$R_{emp}(q(x,\Theta)) = \frac{1}{L} \sum_{i=1}^{L} \frac{1}{2} \left| y_i - q(x_i,\Theta) \right| .$$

Note: a 1/0 loss (penalty) function is used, i.e.,

$$\frac{1}{2} \left| y - q(x,\Theta) \right| = \begin{cases} 0 & \text{if } y = q(x,\Theta) \, , \\ 1 & \text{if } y \neq q(x,\Theta) \, . \end{cases}$$

Minimization of the structural risk (4)

Complications

The expected risk $R_{exp}(q(x,\Theta))$ cannot be calculated because the joint probability distribution $p(x,y)$ is unknown.

Solution

Use the upper bound called guaranteed or structural risk $J(\Theta)$ as proposed by Vapnik-Chervonenkis.

$$R(\Theta) \leq J(\Theta) = R_{\text{emp}}(\Theta) + \sqrt{\frac{h\left(\log\left(\frac{2L}{h}\right)+1\right) - \log\left(\frac{\eta}{4}\right)}{L}}\,.$$

Minimization of the structural risk (5)

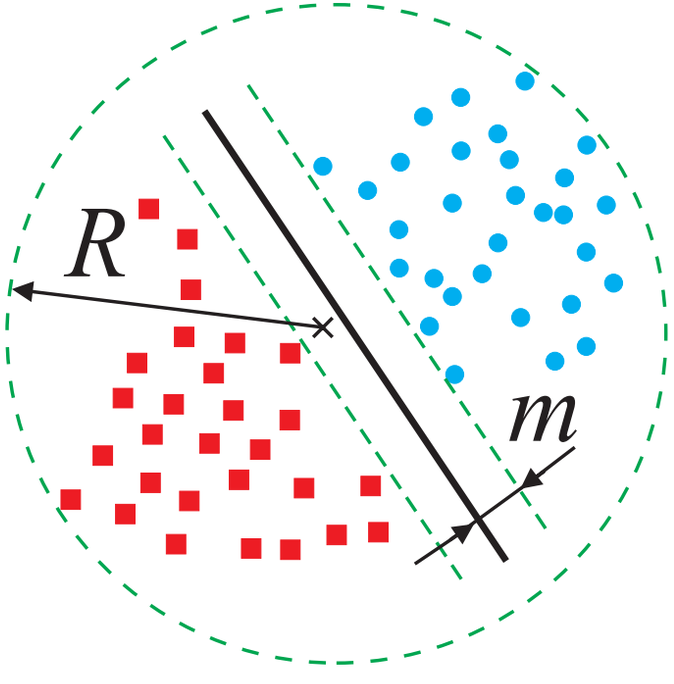
- $R_{emp} = \frac{1}{L} \sum_{i=1}^{L} \frac{1}{2}|y_i - f(x_i, \Theta)|$ is the empirical risk.
- h is a VC dimension characterizing the class of decision functions $q(x, \Theta) \in Q$.
- L is the length of the training multi-set.
- η is the degree of belief into the bound $R(q(x, \Theta))$, i.e., $0 \leq \eta \leq 1$.

- Structural risk minimization principle means a selection of a classifier based on a minimization of the guaranteed risk $J(\Theta)$.
- Support Vector Machines implement an instance of the structural risk minimization principle.

Linearly separable SVM (1)

The aim is to find a linear discriminant function

$$q(x, w, b) = \text{sign}(\langle w, x \rangle + b) = \text{sign}\left(w^{\mathsf{T}} x + b\right)$$



- VC dimension (capacity) depends on the margin m

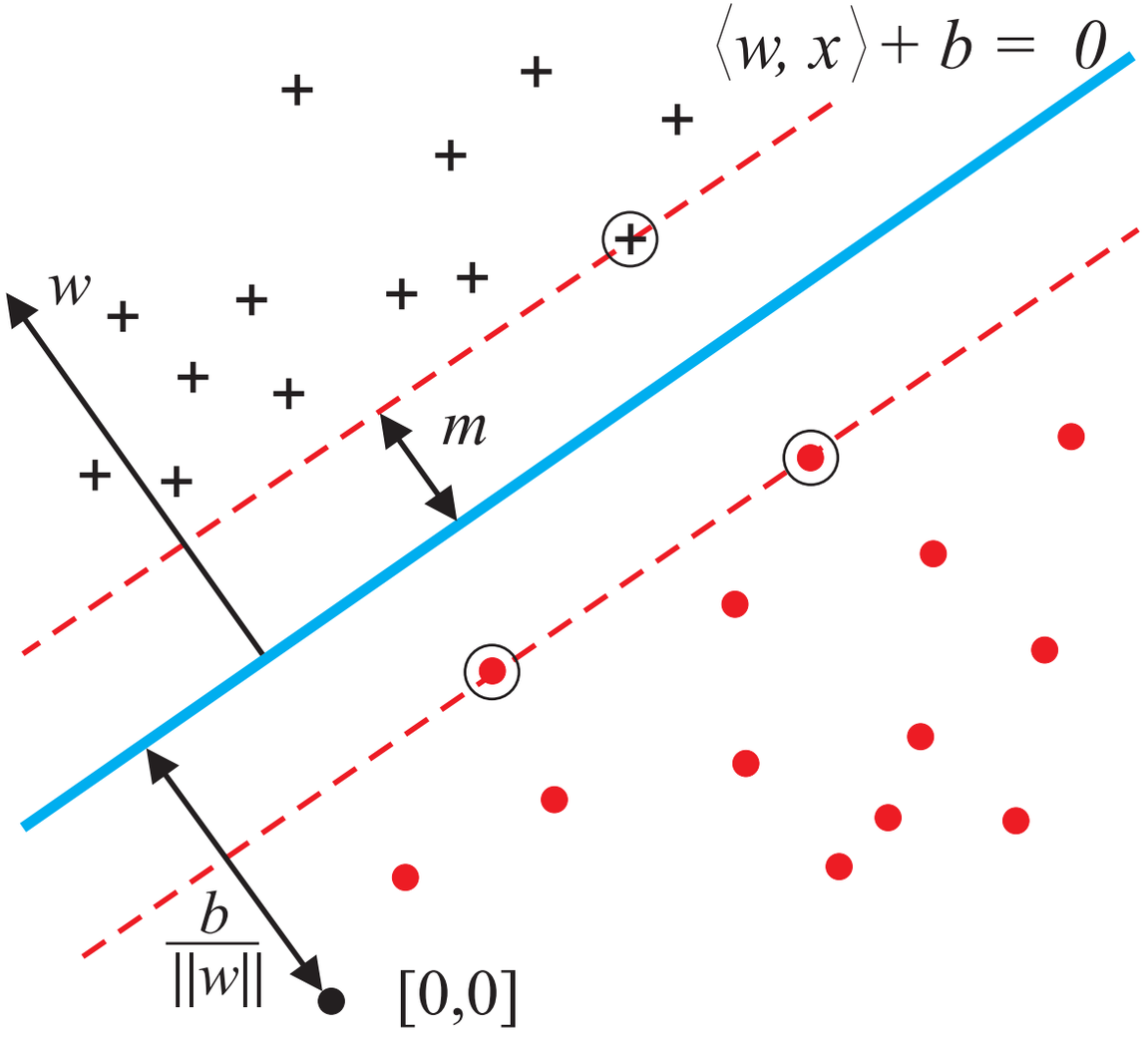
$$h \leq \frac{R^2}{m^2} + 1$$

- R is given by the data itself.
- Margin m can be optimized in the classifier design.

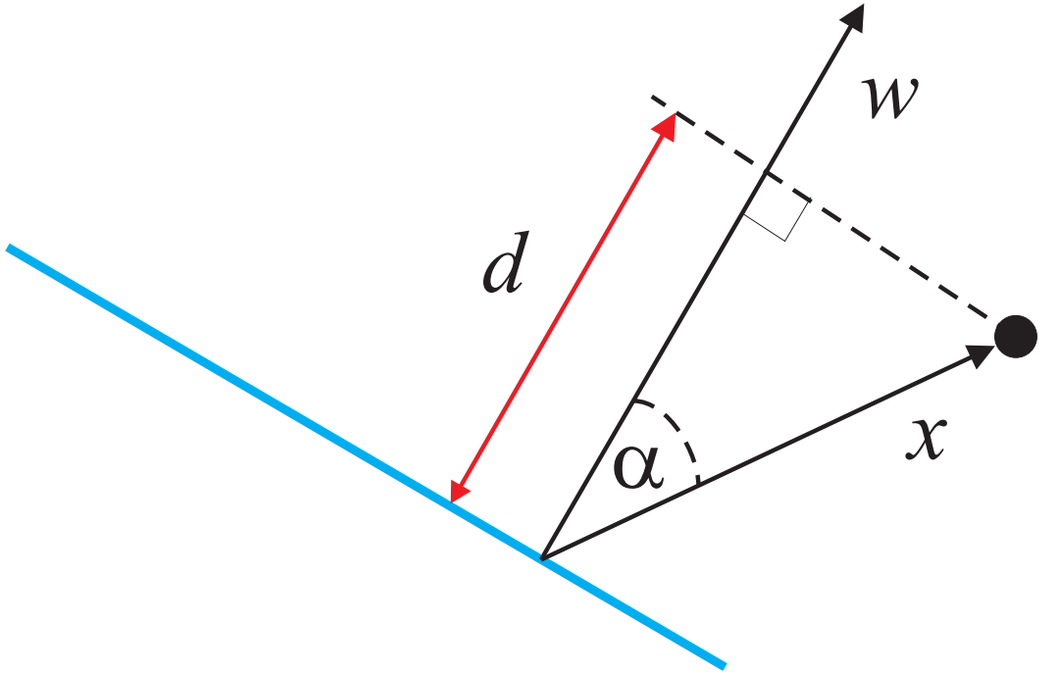
Conclusion: separation hyperplanes with larger margin have lower VC dimension $\Leftrightarrow$ lower value of the upper bound.

Linearly separable SVM (2)

The separating hyperplane is sought which maximizes distance to data (margin).



Linearly separable SVM (3)



Derivation of the distance d between the observation x_i and the separating hyperplane $w^\mathsf{T} x_i + b = 0$

$$\cos\alpha = \frac{w^\mathsf{T} x_i}{\|w\|\|x_i\|}\,, \quad \cos\alpha = \frac{d}{\|x_i\|} \quad \Rightarrow \quad d = \frac{w^\mathsf{T} x_i + b}{\|w\|}$$

The parameter b gives the distance from the origin of coordinates.

Linearly separable SVM, a primal task

The optimization task

$$(w^*, b^*) = \underset{w,b}{\operatorname{argmax}} \min_{i=1,\ldots,L} \frac{w^\mathsf{T} x_i + b}{\|w\|} y_i$$

can be converted in to a standard quadratic programming problem (primal task)

$$(w^*, b^*) = \operatorname{argmin} \frac{1}{2} \|w\|^2$$

$$w^\mathsf{T} x_i + b \geq +1\,, \quad y_i = +1$$

$$w^\mathsf{T} x_i + b \leq -1\,, \quad y_i = -1$$

Towards the dual task

- The aim is to convert the problem into a dual formulation which allows to use kernel functions.
- Lagrange function L is introduced, α_i are Lagrange multipliers,

$$L(w, b, \alpha_i) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i \left(w^{\mathsf{T}} x_i + b\right) \; y_i + \sum_{i=1}^{L} \alpha_i \,. \quad \text{(Eq. 1)}$$

- Now we have formulated the dual task,

$$(w^*, b^*, \alpha^*) = \underset{w,b}{\operatorname{argmin}} \; \max_{\alpha \geq 0} \; L(w, b, \alpha) \qquad \text{Primal task.}$$

$$(w^*, b^*, \alpha^*) = \underset{\alpha \geq 0}{\operatorname{argmax}} \; \min_{w,b} L(w, b, \alpha) \qquad \text{Dual task.}$$

- For convex problems, both formulations lead to the same optimum.

The solution to the dual task

$$\min_{w,b} \max_{\alpha_i>0} L(w,b,\alpha_i) = \max_{\alpha_i>0} \min_{w,b} L(w,b,\alpha_i)$$

Seek optimum, i.e., 1st partial derivatives $= 0$,

$$\frac{\partial L}{\partial w} = 0 \;\Rightarrow\; w = \sum_{i=1}^{L} \alpha_i y_i x_i \,, \qquad \frac{\partial L}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{L} \alpha_i y_i = 0 \,.$$

Substitute to (Eq. 1), get rid off w, b and get

$$\alpha_i = \operatorname*{argmax}_{\alpha_i} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j x_i^{\mathsf{T}} x_j \,,$$

$$\alpha_i \geq 0 \,, \qquad \sum_{i=1}^{L} \alpha_i y_i = 0 \,.$$

SVM decision strategy

$$w = \sum_{i=1}^{L} \alpha_i \, y_i \, x_i \; .$$

$$q(x) = w^\top x + b = \sum_{i=0}^{L} \alpha_i \, x_i^\top x + b \; .$$

SVM – the primal and the dual tasks

Primal task

- Optimized according to vector $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$.
- Number of variables is $L + 1$.
- Number of linear constraints is $2L$.

Dual task

- Optimized according to $\alpha_1, \alpha_2, \ldots, \alpha_L$, $\alpha_i \in \mathbb{R}$.
- Number of variables is L.
- Number of linear constraints is $L + 1$.
- Data appear as scalar products only, i.e., $x_i^\mathsf{T} x_j$.

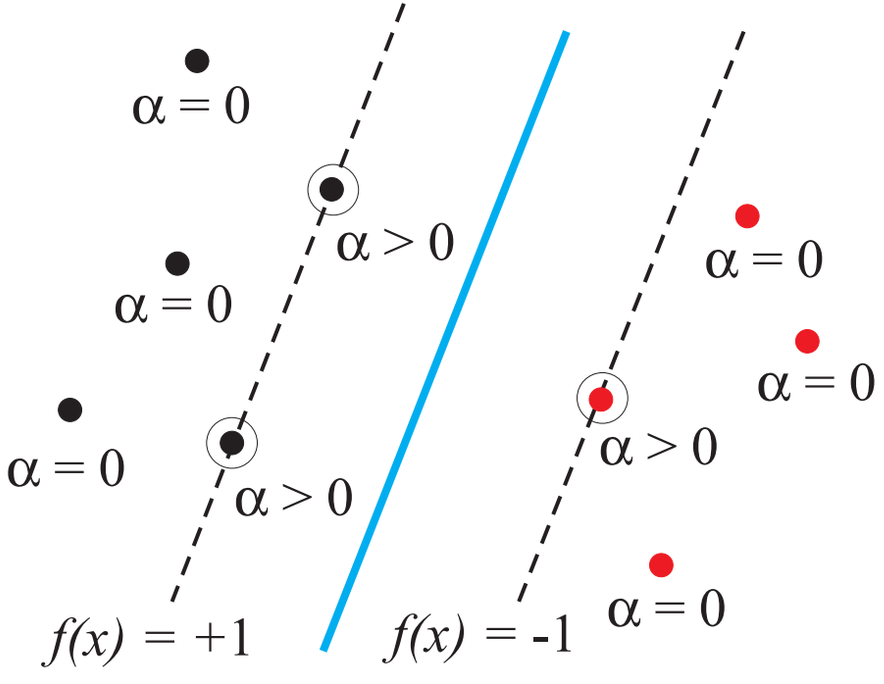
The dual task properties, cont.

- The solution is sparse. Many α_i equal to 0.

  $\alpha_i = 0 \Rightarrow y_i(w^\mathsf{T} x_i + b) \geq 1.$

  $\alpha_i > 0 \Rightarrow y_i(w^\mathsf{T} x_i + b) = 1.$

- Data x_i for which $\alpha_i > 0$ are called Support Vectors.



$$w = \sum_{i=1}^{L} \alpha_i y_i x_i = \sum_{i \in \mathrm{SV}} \alpha_i y_i x_i$$

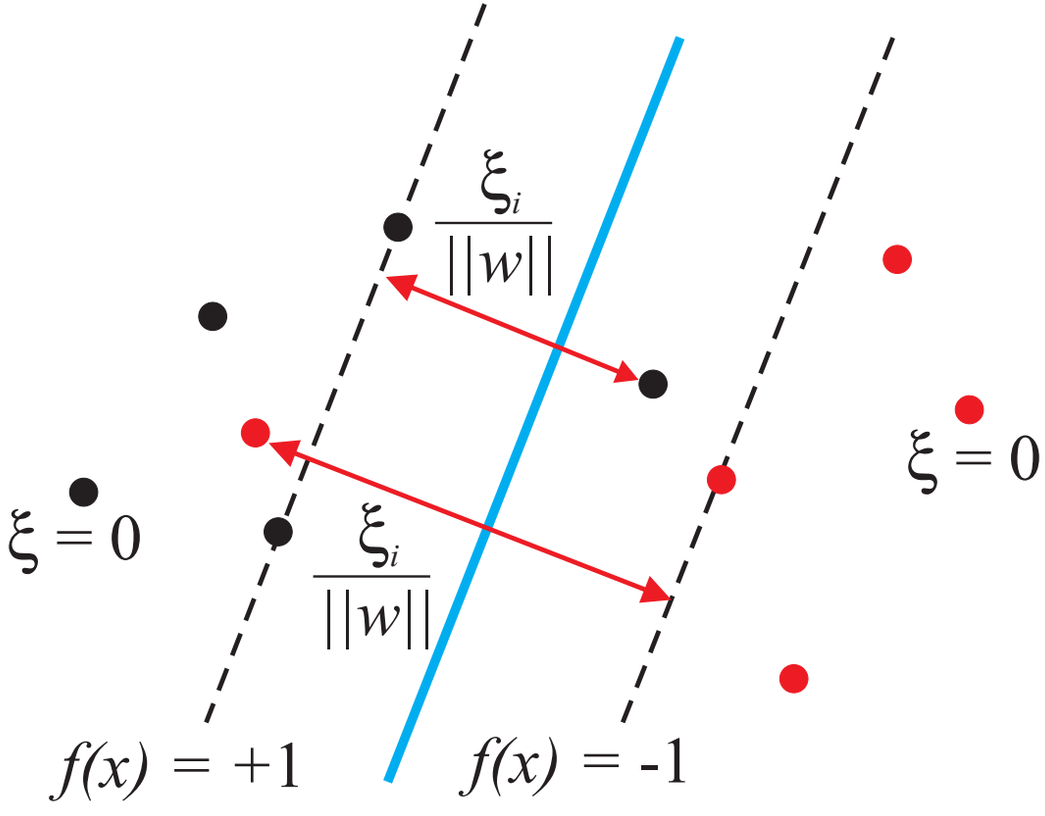
Calculation of b for $i \in$ SV:

$$y_i(w^\mathsf{T} x_i + b) = 1 \Rightarrow b = \frac{1 - y_i w^\mathsf{T} x_i}{y_i} = y_i \langle w,$$

One support vector should be enough.
Practically, many many support vectors, mean of b.

SVM linearly non-separable

Nonseparable data. $\Leftrightarrow$ It is not possible to find separable hyperplane without errors.



Solution: Regularization, i.e., introduction of slack variables $\xi \geq 0 \Rightarrow$ Soft Margin SVM.

A soft margin SVM

$$(w^*, b^*, \xi^*) = \underset{w,b,xi_i}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{L} \xi_i^y \,, \quad \text{where}$$

C is a regularization constant. $C = \infty$ represents the separable case.

$$w^{\mathsf{T}} x_i + b \geq +1 - \xi_i \,, \quad y_i = +1$$

$$w^{\mathsf{T}} x_i + b \leq -1 + \xi_i \,, \quad y_i = -1$$

Optimization criterion, marginal behavior

- $\min \|w\|^2$ – maximization of the margin.
- $\sum_{i=1}^{L} \xi_i^y$ – number misclassified training points (upper bound on the empirical error).

Quadratic programming for the dichotomic task, i.e., $y = 1, 2$ or $|\mathsf{Y}| = 2$.

SVM linearly non-separable, cont.

- Transform to the dual task, analogically to the separable case.

$$\alpha_i = \underset{\alpha_i}{\operatorname{argmax}} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j x_i^\top x_j \,,$$

$$0 \leq \alpha_i \leq C \,, \qquad \sum_{i=1}^{L} \alpha_i y_i = 0 \,.$$

Note: $\leq C$ above is the only difference when comparing to the linearly separable case.

- The decision strategy is

$$q(x) = w^\top x + b = \sum_{i=0}^{L} \alpha_i \, x_i^\top x + b \,.$$

Soft margin SVM, the theoretic backing

$$\text{Risk} = \frac{C}{L}\left(\frac{R^2 + \left(\sum_{i=1}^{L}\xi_i\right)\log\left(\frac{1}{L}\right)}{m^2}\log^2 L + \log\left(\frac{1}{\eta}\right)\right)$$

is minimized when

$$\|w\|^2 R + \left(\sum_{i=1}^{L}\xi_i\right)\log\left(\frac{1}{\sqrt{(\|w\|)}}\right)$$

will be minimal.

This matches to Soft Margin SVM criterion with exception to the last term on the right side.

Interpretation of the constant C

- Parameter C represents a trade-off between misclassification and classifier complexity (given by the VC-dimension).
  - Large values of C favor solutions with few misclassifications.
  - Small values of C express a preference towards low-complexity solutions.
- Parameter C can be viewed as a regularization parameter.
- A suitable value for C is typically determined by trying several values of $C = C_1, \ldots, C_m$. The best value is selected by cross-validation.

SVMs and nonlinear tasks

- We learned already about the possibility to explicitly straighten quadratic decision strategies by embedding the problem into a higher dimensional linear space and the linear classifier for them.
- $\Phi\colon \mathcal{X} \to \mathcal{F}$, such as $q'(x, w, b) = w^\top \Phi(x) + b = \sum_{i=1}^{n} w_i \Phi_i(x_i) + b$.
- After the mapping $\Phi(x)$ is used, the decision strategy $q'(x, w, b)$ is linear in $\mathcal{F}$.
- The problem is the excessively high dimension of the obtained linear space, $\dim(\mathcal{F}) \gg \dim(\mathcal{X})$. The original n dimensional nonlinear space is transformed into the $\left(n + \frac{1}{2}\, n(n+1)\right)$-dimensional feature space.

  Example:

Old dimension	1	2	3	4	5	6	10	20
New dimension	2	5	9	14	20	27	65	230

Problems of a naïve feature space straightening

Two major problems:

1. Statistical: operation on high-dimensional spaces is ill-conditioned due to the 'curse of dimensionality' and the subsequent risk of overfitting.
2. Computational: working in high-dimensions requires higher computational power, which poses limits on the size of the problems that can be tackled.

SVM solution to the problems:

1. Generalization capabilities in the high-dimensional manifold are ensured by enforcing the largest margin classifier. Generalization in SVMs is strictly a function of the margin (or the VC-dimension), regardless of the dimensionality of the feature space.
2. Projection onto a high-dimensional manifold is only implicit because observations from the training set appear as dot products only. Non-linear mapping is realized by kernels.

Kernel functions

- There is a 'kernel trick' which allows to perform calculations more efficiently if the classification algorithm uses observations x only in the form of a dot product.
- Kernel functions $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are symmetric and positive-definite.
- Kernel function $\kappa(x_1, x_2) \doteq \Phi^\top(x_1)\,\Phi(x_2)$, i.e., the kernel function equals to the scalar product of the non-linearly mapped original features.
- Kernel function $\kappa(x_1, x_2)$ can be evaluated without explicit mapping $\Phi\colon \mathcal{X} \to \mathcal{F}$.

***Note related to the notation**: Kernel functions are usually denoted by y in the literature. We follow the notation used in the book Schlesinger, Hlavac 2002 and use y for hidden state. This would create a conflict. We solve the conflict by denoting kernel functions by the Greek letter κ (kappa). Many authors denote hidden state by y and not have this problem.*

SVM, non-separable linearly with kernels

- Transition to the dual task and related optimization using Lagrange coefficients α_i, $i = 1 \dots L$, is analogical to the linear non-separable case.
- The dot product $x_i^\top x_j$ is replaced by the kernel $\kappa(x_i, x_j)$.
- The optimization problem reads now

$$\alpha_i = \underset{\alpha_i}{\operatorname{argmax}} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \,,$$

$$0 \leq \alpha_i \leq C \,, \qquad \sum_{i=1}^{L} \alpha_i y_i = 0 \,.$$

- The decision strategy is

$$q(x) = w^\top \Phi(x) + b = \sum_{i=0}^{L} \alpha_i \, \kappa(x_i, x) + b \,.$$

How do we select a kernel function?

- Mercer's condition tells us whether or not a candidate kernel is actually an inner-product kernel in some space.
- Let $\kappa(x_1, x_2)$ be a continuous symmetric kernel defined in the closed interval $a \leq x \leq b$. The kernel can be expanded into series $\sum_{i=1}^{\infty} \lambda_i \phi(x_1)\phi(x_2)$. Functions ϕ reside in the Hilbert space, a 'generalization' of an Euclidean space where the inner product can be any inner product, not just the common dot product.
- Consider $\lambda_i > 0$, $\forall i$. The necessary and sufficient condition for the uniform convergence of the above series is that

$$\int_a^b \int_a^b \kappa(x_1, x_2)\, \psi(x_1)\, \psi(x_2)\, \mathrm{d}x_1 \mathrm{d}x_2 \geq 0\,,$$

holds for all $\psi(.)$, for which $\int_a^b \psi^2(x)\mathrm{d}x < \infty$.

How do we select a kernel function?, cont.

- Functions $\phi_i(.)$ are called eigenvectors of the expansion. Numbers λ_i are eigenvectors.
- The fact that all of the eigenvalues are positive means that the kernel is positive definite.
- Notice that the dimensionality of the implicit space can be infinitely large.
- Mercer's condition only tells us whether a kernel is actually an inner-product kernel. It does not tell us how to construct the functions $\phi_i(x)$ for the expansion.

Kernels complying to Mercer's condition

- Polynomial kernels of degree p, $\kappa(x_1, x_2) = \left(x_1^\top x_2 + 1\right)^p$.
- Radial basis functions

$$\kappa(x_1, x_2) = \exp\left(\frac{-1}{2\sigma^2}\left\|x_1 - x_2\right\|^2\right) .$$

  The width of the kernel σ is a user defined parameter. The number of radial basis functions is determined automatically.

- Two layer perceptron $\kappa(x_1, x_2) = \tanh(\beta_0\, x_1^\top x_2 + \beta_1)$

  The number of hidden neurons and their weight vectors are determined automatically by the number of support vectors and their values, respectively. The hidden-to-output weights are the Lagrange multipliers α_i. However, this kernel will only meet Mercer's condition for certain values of β_0 and β_1.

Problems with kernel functions

Kernel methods yield the solution in the form

$$q(x) = \Phi^\top(x) \cdot \underbrace{\sum_{i=1}^{L} \alpha_i \, \Phi(x_i)}_{w \in \mathcal{F}} + b = \sum_{i=1}^{L} \alpha_i \, \kappa(x, x_i) + b \,,$$

where $[x_1, \ldots, x_L]$ is the sequence of vectors in the training set.

- The decision function $q(x) = \sum_{i=1}^{L} \alpha_i \, \kappa(x, x_i) + b$ is not sparse, i.e., a lot of α_i are non-zero $\Rightarrow$ slow evaluation.
- Representation of the training set in terms of a dot product is memory demanding for large L since the full kernel matrix $\mathbb{Y}_{i,j} = \kappa(x_i, x_j)$ has dimension $[L \times L]$.
- Evaluation of $\kappa(x_i, x_j)$ can be computationally demanding.

Notes on SVM performance

- SVMs work very well in practice.
- The user chooses the kernel function, its parameters and the regularization constant C. The rest is automatic.
- SVMs can be expensive in time and space for large datasets.
  - The computation of the maximum-margin hyper-plane has a lower bound $\mathcal{O}(L^2)$ for the nonlinear case and $\mathcal{O}(L)$ for the linear case.
  - All the support vectors have to be stored in a memory.

Multi-class SVMs

Several approaches are used:

- Direct multi-class formulation.
- One-against all.
- One against one.
- DAG, Directed Acyclic Graphs.

Multi-class SVMs, one-against-all

- So far, we have considered only SVMs handling two-class problems, i.e, dichotomic classification.
- If the task is to classify into N classes then then learn N independent SVMs such that
  - SVM 1 learns $y = 1$ vs. $y \neq 1$.
  - SVM 2 learns $y = 2$ vs. $y \neq 2$.
  - ...
  - SVM N learns $y = N$ vs. $y \neq N$.
- In a run mode when deciding about new observation, apply all N SVMs and select the class by looking which SVM puts the prediction the furthest into the positive region.