Machine Learning

Lecture 9: SVM and Kernels

Prof. Dr. Stephan Günnemann
Aleksandar Bojchevski

Data Analytics and Machine Learning Group
Technical University of Munich

Winter term 2020/2021

Roadmap

1. Support Vector Machines (SVM)
2. Soft Margin Support Vector Machines
3. Kernels

Section 1

Support Vector Machines (SVM)

Linear classifier

A linear classifier assigns all $\boldsymbol{x}$ with

$$\boldsymbol{w}^T\boldsymbol{x} + b > 0$$

to class blue and all $\boldsymbol{x}$ with

$$\boldsymbol{w}^T\boldsymbol{x} + b < 0$$

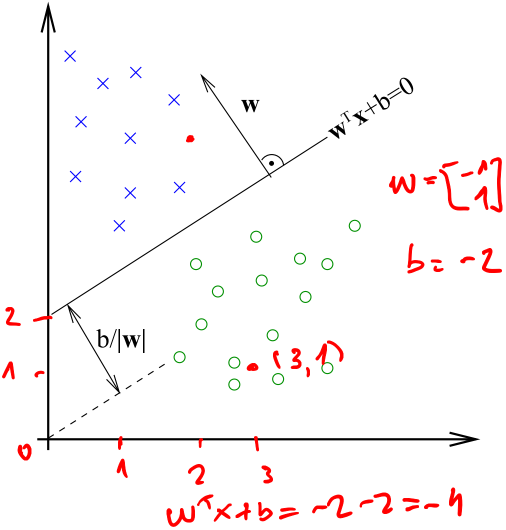
to class green.

Thus the class of $\boldsymbol{x}$ is given by

$$h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^T\boldsymbol{x} + b)$$

with

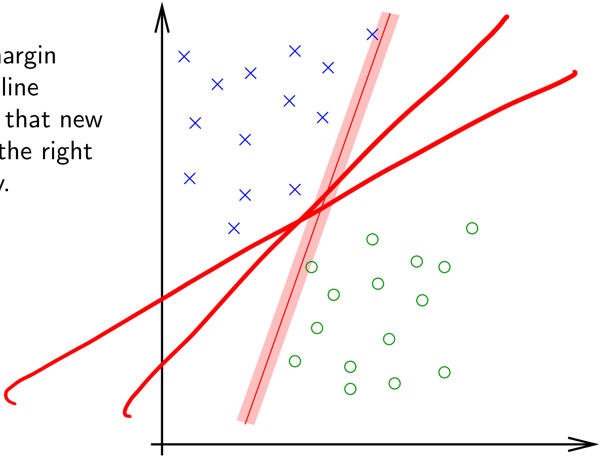
$$\operatorname{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}.$$



w
$\mathbf{w}^T\mathbf{x}+b=0$
$b/|\mathbf{w}|$
$w = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
$b = -2$
(3, 1)
2
1
0
1
2
3
$w^T x + b = -2 - 2 = -4$

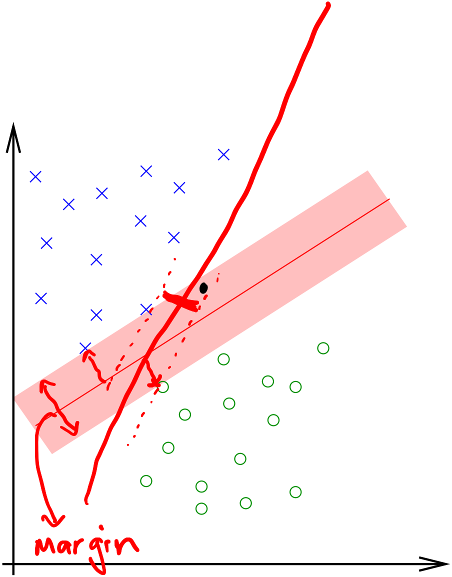
Maximum margin classifier

- Intuitively, a wide margin around the dividing line makes it more likely that new samples will fall on the right side of the boundary.



Maximum margin classifier

- Intuitively, a wide margin around the dividing line makes it more likely that new samples will fall on the right side of the boundary.
- Actual rigorous motivation comes from Statistical Learning Theory [1]
- Objective:
  Find a hyperplane that separates both classes with the maximum margin.



[1] V. Vapnik - "Statistical Learning Theory", 1995

Linear classifier with margin

We add two more hyperplanes that are parallel to the original hyperplane and require that no training points must lie between those hyperplanes.

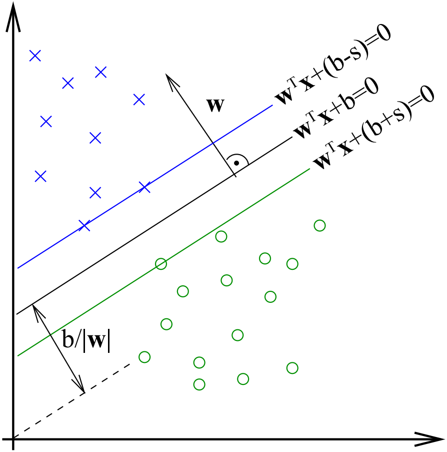
Thus we now require

$$\boldsymbol{w}^T \boldsymbol{x} + (b - s) > 0$$

for all $\boldsymbol{x}$ from class blue and

$$\boldsymbol{w}^T \boldsymbol{x} + (b + s) < 0$$

for all $\boldsymbol{x}$ from class green.



$\mathbf{w}^T\mathbf{x}+(b\text{-}s)=0$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+(b+s)=0$
$b/|\mathbf{w}|$

Size of the margin

Signed distance from the origin to the hyperplane is given by

$$d = -\frac{b}{||\boldsymbol{w}||}\,.$$

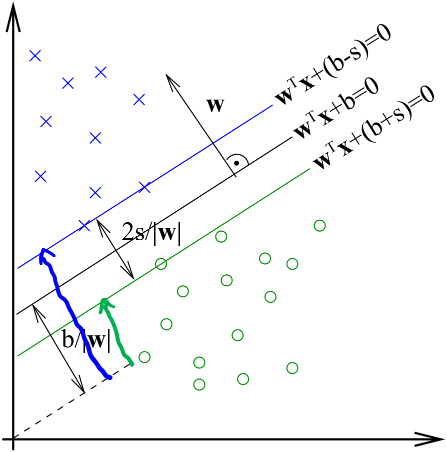
Thus we have

$$d_{blue} = -\frac{b-s}{||\boldsymbol{w}||}$$

$$d_{green} = -\frac{b+s}{||\boldsymbol{w}||}$$

and the margin is

$$m = d_{blue} - d_{green} = \frac{2s}{||\boldsymbol{w}||}\,.$$

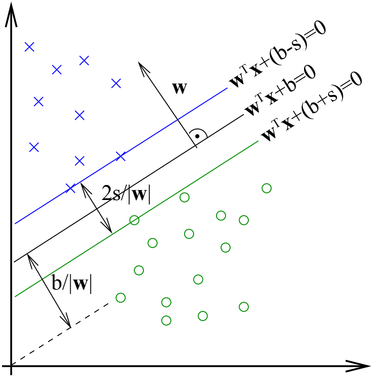

$\mathbf{w}^T\mathbf{x}+(b\text{-}s)=0$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+(b+s)=0$
$\mathbf{w}$
$2s/|\mathbf{w}|$
$b/|\mathbf{w}|$

Redundancy of parameter s

The size of the margin,

$$m = \frac{2s}{||\boldsymbol{w}||}$$

only depends on the ratio, so w.l.o.g. we can set $s = 1$ and get



$\mathbf{w}^T\mathbf{x}+(b\text{-}s)=0$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+(b+s)=0$
$\mathbf{w}$
$2s/|\mathbf{w}|$
$b/|\mathbf{w}|$

Redundancy of parameter s

The size of the margin,

$$m = \frac{2s}{||\boldsymbol{w}||}$$

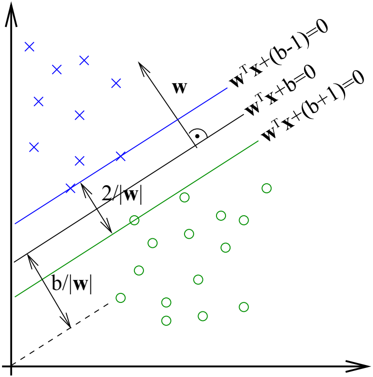
only depends on the ratio, so w.l.o.g. we can set $s = 1$ and get

$$m = \frac{2}{||\boldsymbol{w}||}$$

Although the distance from the origin to the black plane,

$$d = -\frac{b}{||\boldsymbol{w}||},$$

also depends on two parameters we *cannot* set $b = 1$ as this would link the distance d to the size of the margin m.



$\mathbf{w}^T\mathbf{x}+(b-1)=0$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+(b+1)=0$
$\mathbf{w}$
$2/|\mathbf{w}|$
$b/|\mathbf{w}|$

Set of constraints

Let $\boldsymbol{x}_i$ be the ith sample, and $y_i \in \{-1, 1\}$ the class assigned to $\boldsymbol{x}_i$.

The constraints

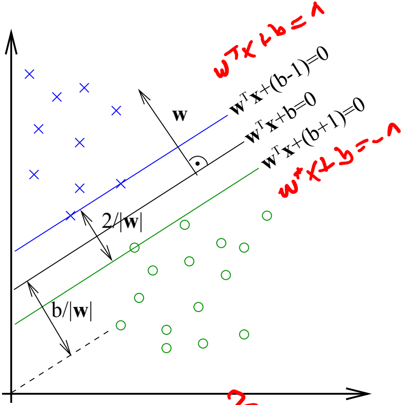
$$\boldsymbol{w}^T \boldsymbol{x}_i + b \geq +1 \quad \text{for } y_i = +1\,,$$
$$\boldsymbol{w}^T \boldsymbol{x}_i + b \leq -1 \quad \text{for } y_i = -1$$

can be condensed into

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 \quad \text{for all } i\,.$$

If these constraints are fulfilled the margin is

$$m = \frac{2}{||\boldsymbol{w}||} = \frac{2}{\sqrt{\boldsymbol{w}^T \boldsymbol{w}}}\,.$$



$w^Tx+b=1$
$\mathbf{w}^T\mathbf{x}+(b-1)=0$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+(b+1)=0$
$w^Tx+b=-1$
$2/|\mathbf{w}|$
$b/|\mathbf{w}|$

$$\max_{w,b} \frac{2}{\|w\|}$$

SVM‘s Optimization problem

Let $\boldsymbol{x}_i$ be the ith data point, $i = 1, \ldots, N$, and $y_i \in \{-1, 1\}$ the class assigned to $\boldsymbol{x}_i$.

To find the separating hyperplane with the maximum margin we need to find $\{\boldsymbol{w}, b\}$ that

$$\begin{aligned} \text{minimize} \quad & f_0(\boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} = \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & f_i(\boldsymbol{w}, b) = y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq 0 \quad \text{for } i = 1, \ldots, N\,. \end{aligned}$$

This is a constrained convex optimization problem (more specifically, quadratic programming problem).

$$\frac{2}{\|w\|}$$

We go from $||\boldsymbol{w}||$ to $||\boldsymbol{w}||^2 = \boldsymbol{w}^T\boldsymbol{w}$ as square root is a monotonic function that doesn't change the location of the optimum.

Optimization with inequality constraints

$\theta = \{w, b\}$

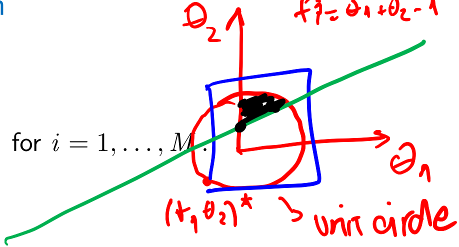
$f_0 = \theta_1 + \theta_2$

$f_1 = \theta_1^2 + \theta_2^2 - 1$

Constrained optimization problem

Given $f_0 : \mathbb{R}^d \to \mathbb{R}$ and $f_i : \mathbb{R}^d \to \mathbb{R}$,

$$\begin{aligned} \text{minimize}_{\boldsymbol{\theta}} \quad & f_0(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \le 0 \quad \text{for } i = 1, \ldots, M. \end{aligned}$$



Feasibility

A point $\boldsymbol{\theta} \in \mathbb{R}^d$ is called feasible if and only if it satisfies the constraints of the optimization problem, i.e. $f_i(\boldsymbol{\theta}) \le 0$ for all $i \in \{1, \ldots, M\}$.

Minimum and minimizer

We call the optimal value the minimum p^*, and the point where the minimum is obtained the minimizer $\boldsymbol{\theta}^*$. Thus $p^* = f_0(\boldsymbol{\theta}^*)$.

Lagrangian

$$\begin{aligned} \text{minimize}_{\boldsymbol{\theta}} \quad & f_0(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \leq 0 \quad \text{for } i = 1, \ldots, M \end{aligned}$$

Definition (Lagrangian)

We define the Lagrangian $L : \mathbb{R}^d \times \mathbb{R}^M \to \mathbb{R}$ associated with the above problem as

$$\boxed{L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = f_0(\boldsymbol{\theta}) + \sum_{i=1}^{M} \alpha_i f_i(\boldsymbol{\theta})}.$$

We refer to $\alpha_i \geq 0$ as the Lagrange multiplier associated with the inequality constraint $f_i(\boldsymbol{\theta}) \leq 0$.

Lagrange dual function

Definition (Lagrange dual function)

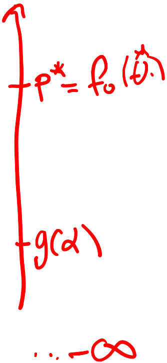
The Lagrange dual function $g : \mathbb{R}^M \rightarrow \mathbb{R}$ maps $\boldsymbol{\alpha}$ to the minimum of the Lagrangian over $\boldsymbol{\theta}$ (possibly $-\infty$ for some values of $\boldsymbol{\alpha}$),

$$\boxed{g(\boldsymbol{\alpha}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^d} L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \left(f_0(\boldsymbol{\theta}) + \sum_{i=1}^{M} \alpha_i f_i(\boldsymbol{\theta}) \right)}.$$

It is concave in $\boldsymbol{\alpha}$ since it is the point-wise minimum of a family of affine functions of $\boldsymbol{\alpha}$.

unconstrained

Interpretation of the Lagrangian



$$\begin{aligned} \text{minimize}_{\boldsymbol\theta} \quad & f_0(\boldsymbol\theta) \\ \text{subject to} \quad & f_i(\boldsymbol\theta) \le 0, \quad i = 1, \dots, M \end{aligned}$$

$$\boxed{L(\boldsymbol\theta, \boldsymbol\alpha) = f_0(\boldsymbol\theta) + \sum_{i=1}^{M} \alpha_i f_i(\boldsymbol\theta)}$$

For every choice of $\boldsymbol\alpha$, the corresponding *unconstrained* $g(\boldsymbol\alpha) = \min_{\boldsymbol\theta \in \mathbb{R}^d} L(\boldsymbol\theta, \boldsymbol\alpha)$ is a lower bound on the optimal value of the constrained problem:

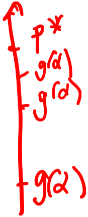
$$\min_{\substack{\boldsymbol\theta\in\mathbb{R}^d\\ f_i(\boldsymbol\theta)\le 0}} f_0(\boldsymbol\theta) = f_0(\boldsymbol\theta^*) \ge f_0(\boldsymbol\theta^*) + \underbrace{\sum_{i=1}^{M} \underbrace{\alpha_i}_{\ge 0} \underbrace{f_i(\boldsymbol\theta^*)}_{\le 0}}_{\le 0} = L(\boldsymbol\theta^*, \boldsymbol\alpha) \ge \underbrace{\min_{\boldsymbol\theta\in\mathbb{R}^d} L(\boldsymbol\theta, \boldsymbol\alpha)}_{g(\boldsymbol\alpha)}$$

Hence, $\forall \boldsymbol\alpha \; f_0(\boldsymbol\theta^*) \ge g(\boldsymbol\alpha)$.

Lagrange dual problem

For each $\boldsymbol{\alpha} \geq \mathbf{0}$ the Lagrange dual function $g(\boldsymbol{\alpha})$ gives us a lower bound on the optimal value p^* of the original optimization problem.

What is the best (highest) lower bound?



Lagrange dual problem

$$\begin{aligned} \text{maximize}_{\boldsymbol{\alpha}} \quad & g(\boldsymbol{\alpha}) \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

The maximum d^* of the Lagrange dual problem is the best lower bound on p^* that we can achieve by using the Lagrangian.

Note: since we are maximizing g we are not interested in dual multipliers $\boldsymbol{\alpha}$ such that $g(\boldsymbol{\alpha}) = -\infty$, so the condition $g(\boldsymbol{\alpha}) \neq -\infty$ is usually added as an additional constraint to the dual problem → we call $\boldsymbol{\alpha}$ feasible if and only if all $\alpha_i \geq 0$ and $L(\boldsymbol{\theta}, \boldsymbol{\alpha})$ is bounded from below for $\boldsymbol{\theta} \in \mathbb{R}^d$.

Duality

Weak duality (always)

Since for all $\boldsymbol{\alpha} \geq \mathbf{0}$ it holds that $g(\boldsymbol{\alpha}) \leq p^*$ we have weak duality,

$$d^* \leq p^* \, .$$

The difference $p^* - d^* \geq 0$ between the solution of the original and the dual problem is called the duality gap.

Strong duality (under certain conditions)

Under certain conditions we have

$$d^* = p^* \, ,$$

i.e. the maximum to the Lagrange dual problem is the minimum of the original (primal) constrained optimization problem (i.e. $f_0(\boldsymbol{\theta}^*) = g(\boldsymbol{\alpha^*})$).

SVM's Primal problem

We apply a recipe for solving the constrained optimization problem.

1. Calculate the Lagrangian

$$L(\underbrace{\boldsymbol{w}, b}_{\theta}, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i \underbrace{[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1]}_{f_i} .$$

$$\min_{w,b} w^T w$$

2. Minimize $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ w.r.t. $\boldsymbol{w}$ and b.

$$g(\alpha) = \min_{w,b} L(w,b,\alpha)$$

$$\forall i \quad \underbrace{y_i(w^T x_i + b) - 1}_{f_i} \geq 0$$

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} \overset{!}{=} 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i \overset{!}{=} 0$$

Thus the weights are a linear combination of the training samples,

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} .$$

SVM‘s Dual problem

Substituting both relations back into $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ gives the Lagrange dual function $g(\boldsymbol{\alpha})$.

Thus we have reformulated our original problem as

$$\begin{aligned} \text{maximize} \quad & g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x_i}^T \boldsymbol{x_j} \\ \text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \qquad \text{for} \quad i = 1, \dots, N\,. \end{aligned}$$

3. Solve this problem.

Solving the dual problem

We can rewrite the dual function $g(\boldsymbol{\alpha})$ in vector form

$$g(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1}_N$$

where $\boldsymbol{Q}$ is a symmetric negative (semi-)definite matrix, and the constraints on $\boldsymbol{\alpha}$ are linear.

This is an instance of a quadratic programming problem.
There exist efficient algorithms for its solution, such as Sequential minimal optimization (SMO) [2].

A number of implementations, such as `LIBSVM` [3] are available and are widely used in practice.

[2] `http://cs229.stanford.edu/materials/smo.pdf`

[3] C.-C. Chang and C.-J. Lin. *LIBSVM : a library for support vector machines*, 2011

Recovering $\boldsymbol{w}$ and b from the dual solution $\boldsymbol{\alpha}^*$

Having obtained the optimal $\boldsymbol{\alpha}^*$ using our favorite QP solver, we can compute the parameters defining the separating hyperplane.

Recall, that from the optimality condition, the weights $\boldsymbol{w}$ are a linear combination of the training samples,

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i^* y_i \boldsymbol{x_i}$$

KKT

$$f_i(\theta^*) = 0$$

$$y_i(w^T x_i + b) - 1 = 0$$

From the complementary slackness condition $\alpha_i^* f_i(\boldsymbol{\theta}^*) = 0$ we can easily recover the bias.

When we take any vector $\boldsymbol{x}_i$ for which $\alpha_i \neq 0$. The corresponding constraint $f_i(\boldsymbol{w}, b)$ must be zero and thus we have

$$\boldsymbol{w}^T \boldsymbol{x_i} + b = y_i \,.$$

$$d^* = p^*$$

$$g(\alpha^*) = f_0(\theta^*)$$

Solving this for b yields the bias

$$b = y_i - \boldsymbol{w}^T \boldsymbol{x_i}$$

$$f_0(\theta^*) + \underbrace{\sum \alpha_i^* f_i(\theta^*)}_{=0} = f_0(\theta$$

We can also average the b over all support vectors to get a more stable solution.

Support vectors

From complimentary slackness

$$\alpha_i^* f_i(\boldsymbol{\theta}^*) = 0\,.$$

we have

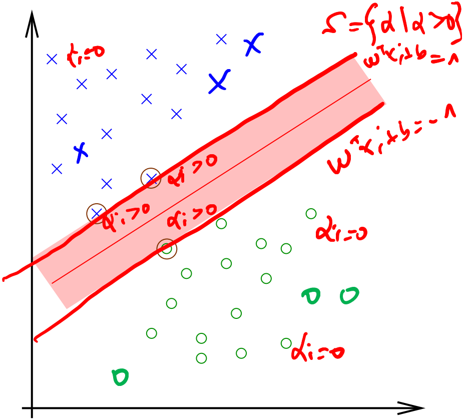
$$\alpha_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1] = 0 \quad \text{for all } i\,.$$

Hence a training sample $\boldsymbol{x}_i$ can only contribute to the weight vector ($\alpha_i \neq 0$) if it lies on the margin, that is

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1\,.$$

A training sample $\boldsymbol{x}_i$ with $\alpha_i \neq 0$ is called a support vector.

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i = \sum_{S} \alpha_i y_i x_i$$



Classifying

The class of $\boldsymbol{x}$ is given by

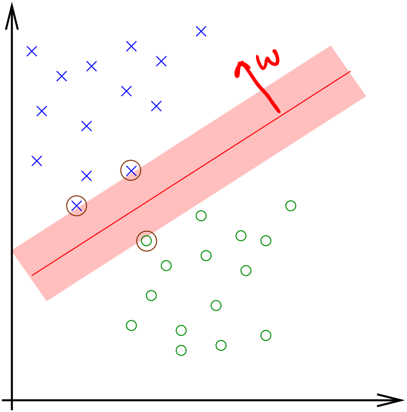
$$h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^T \boldsymbol{x} + b)\,.$$

Substituting

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i}$$

gives

$$h(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i}^T \boldsymbol{x} + b\right)\,.$$



Since the solution is sparse (most α_is are zero) we only need to remember the few training samples $\boldsymbol{x_i}$ with $\alpha_i \neq 0$.

Section 2

Soft Margin Support Vector Machines

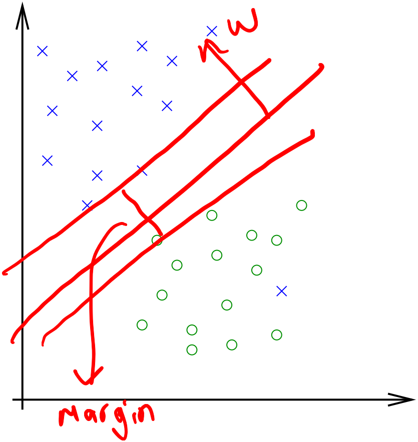
Dealing with noisy data

What if the data is not linearly separable due to some noise?

With our current version of SVM, a single outlier makes the constraint unsatisfiable.

The corresponding Lagrange multiplier α_i would go to infinity and destroy the solution.

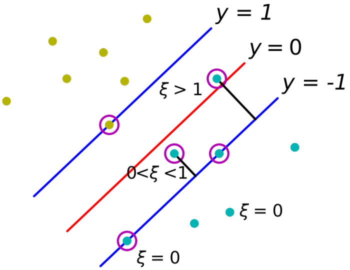
How to make our model more robust?



Slack variables

Idea: Relax the constraints as much as necessary but punish the relaxation of a constraint.

We introduce a slack variable $\xi_i \geq 0$ for every training sample $\boldsymbol{x}_i$ that gives the distance of how far the margin is violated by this training sample in units of $||\boldsymbol{w}||$.



Slack variables

Idea: Relax the constraints as much as necessary but punish the relaxation of a constraint.

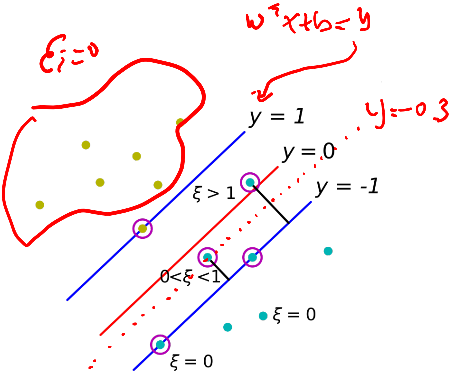
We introduce a slack variable $\xi_i \geq 0$ for every training sample $\boldsymbol{x}_i$ that gives the distance of how far the margin is violated by this training sample in units of $||\boldsymbol{w}||$.

Hence our relaxed constraints are

$$\boldsymbol{w}^T \boldsymbol{x}_i + b \geq +1 - \xi_i \quad \text{for } y_i = +1\,,$$
$$\boldsymbol{w}^T \boldsymbol{x}_i + b \leq -1 + \xi_i \quad \text{for } y_i = -1\,.$$

Again, they can be condensed into

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \text{for all } i\,.$$



$$\underbrace{w^T x_i + b}_{-0.3} \leq -1 + \underbrace{0.7}_{\xi_i}$$

Slack variables

Idea: Relax the constraints as much as necessary but punish the relaxation of a constraint.

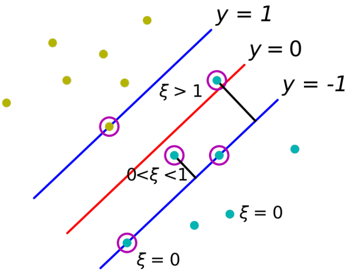
We introduce a slack variable $\xi_i \geq 0$ for every training sample $\boldsymbol{x}_i$ that gives the distance of how far the margin is violated by this training sample in units of $||\boldsymbol{w}||$.

Hence our relaxed constraints are

$$\boldsymbol{w}^T \boldsymbol{x}_i + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \,,$$
$$\boldsymbol{w}^T \boldsymbol{x}_i + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \,.$$

Again, they can be condensed into

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \text{for all } i \,.$$



The new cost function is,

$$f_0(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i \,.$$

The factor $C > 0$ determines how heavy a violation is punished.

$C \to \infty$ recovers hard-margin SVM.

Optimization problem with slack variables

Let $\boldsymbol{x}_i$ be the ith data point, $i = 1, \dots, N$, and $y_i \in \{-1, 1\}$ the class assigned to $\boldsymbol{x}_i$. Let $C > 0$ be a constant.

To find the hyperplane that separates most of the data with maximum margin we

$$\begin{array}{rlr} \text{minimize} & f_0(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \underline{C\sum_{i=1}^{N}\xi_i} & \\ \text{subject to} & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \underline{\underline{\xi_i}} \geq 0 & i = 1, \dots, N\,, \\ & \underline{\xi_i \geq 0} & i = 1, \dots, N\,. \end{array}$$

Here we used the 1-norm for the penalty term $\sum_i \xi_i$. Another choice is to use the 2-norm penalty, $\sum_i \xi_i^2$.

The penalty that performs better in practice will depend on the data and the type of noise that has influenced it.

Lagrangian with slack variables

$L(\theta, \alpha) = f_0(\theta) + \sum_i \alpha_i f_i(\theta)$

$g(\alpha) = \min_{\theta} L(\theta, \alpha)$

$\alpha = [\alpha, \mu]$

$\theta = [w, b, \xi]$

1. Calculate the Lagrangian

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \overbrace{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N} \xi_i}^{f_0(\theta)}$$

$-\sum_i \alpha_i y_i b$

$\xi_i (C - \alpha_i - \mu_i)$

$$-\sum_{i=1}^{N} \alpha_i [\underbrace{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i}_{f_i}] - \sum_{i=1}^{N} \mu_i\underbrace{\xi_i}_{f_i}\,.$$

2. Minimize $L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$ w.r.t. $\boldsymbol{w}$, b and $\boldsymbol{\xi}$.

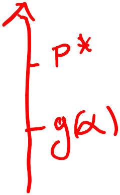
$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \boldsymbol{w} - \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} \overset{!}{=} 0; \qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i \overset{!}{=} 0,$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i \overset{!}{=} 0 \quad \text{for} \quad i = 1, \dots, N$$

From $\alpha_i = C - \mu_i$ and dual feasibility $\mu_i \geq 0$, $\alpha_i \geq 0$ we get

$$0 \leq \alpha_i \leq C\,.$$

Dual problem with slack variables



This leads to the dual problem:

$$\text{maximize} \quad g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x_i}^T \boldsymbol{x_j}$$

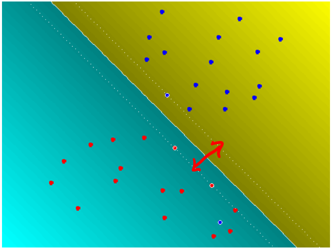
$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

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

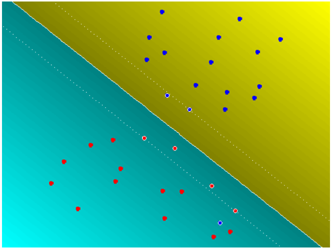
This is nearly the same dual problem as for the case without slack variables.

Only the constraint $\alpha_i \leq C$ is new. It ensures that α_i is bounded and cannot go to infinity.

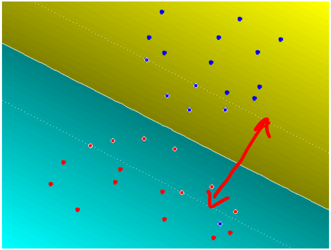
Influence of the penalty C



$C = 100$



$C = 10$



$C = 1$

Hinge loss formulation

We can have another look at our constrained optimization problem.

$$\text{minimize} \quad f_0(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geq 0, \qquad i = 1, \dots, N\,,$$

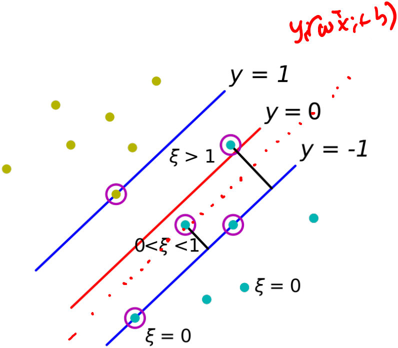
$$\xi_i \geq 0, \qquad i = 1, \dots, N\,.$$

Clearly, for the optimal solution the slack variables are

$$\xi_i = \begin{cases} 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b), & \text{if } \underbrace{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}_{\text{margin is violated}} < 1 \\ 0 & \text{else} \end{cases}$$

$\max(0, 1 - y_i(w^Tx_i+b))$

since we are minimizing over $\boldsymbol{\xi}$.



Hinge loss formulation

Thus, we can rewrite the objective function as an unconstrained optimization problem known as the hinge loss formulation

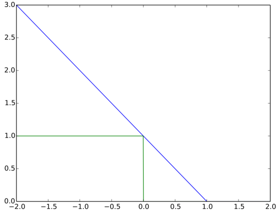
- perceptron

C=1

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\boxed{\sum_{i=1}^{N} \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\}}$$

The hinge loss function $E_{\text{hinge}}(z) = \max\{0, 1 - z\}$ penalizes the points that lie within the margin.

The name comes from the shape from the function, as can be seen in the figure to the right.



Hinge loss (blue) can be viewed as an approximation to zero-one loss (green).

We can optimize this hinge loss objective directly, using standard gradient-based methods.

Section 3

Kernels

Feature space

So far we can only construct linear classifiers.

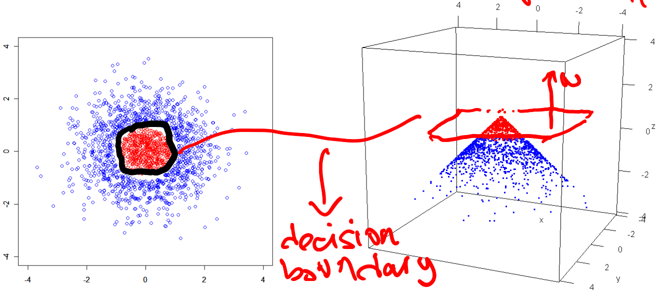
Before, we used basis functions $\phi(\cdot)$ to make the models nonlinear

$$\boldsymbol{\phi} : \mathbb{R}^D \rightarrow \mathbb{R}^M \qquad \boldsymbol{x}_i \mapsto \boldsymbol{\phi}(\boldsymbol{x}_i)$$

For example, with the following mapping the data becomes linearly separable

$$\boldsymbol{\phi}(x, y) = \begin{pmatrix} x \\ y \\ -\sqrt{x^2 + y^2} \end{pmatrix}$$

$\text{sign}(w^T \phi(x_i) + b)$



Kernel trick

In the dual formulation of SVM, the samples $\boldsymbol{x}_i$ only enter the dual objective as inner products $\boldsymbol{x_i}^T \boldsymbol{x_j}$

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x_i}^T \boldsymbol{x_j} \,,$$

For basis functions this means that

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{\phi(x_i)}^T \boldsymbol{\phi(x_j)}$$

Kernel trick

We can define a kernel function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$$

and rewrite the dual as

$x_i^T x_j$

$\downarrow$

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

This operation is referred to as the kernel trick.

It can be used not only for SVM. Kernel trick can be used in any model that can be formulated such that it only depends on the inner products $\boldsymbol{x}_i^T \boldsymbol{x}_j$. (e.g. linear regression, k-nearest neighbors)

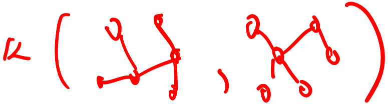
Kernel trick

The kernel represent an inner product in the feature space spanned by ϕ. Like before, this makes our models non-linear w.r.t. the data space.

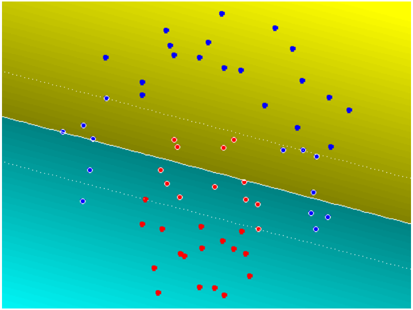
What's the point of using kernels if we can simply use the basis functions?

- Some kernels are equivalent to using infinite-dimensional basis functions. While computing these feature transformations would be impossible, directly evaluating the kernels is often easy.
- Kernels can be used to encode similarity between arbitrary non-numerical data, from strings to graphs.
  For example, we could define

$$k(\texttt{lemon}, \texttt{orange}) = 10 \quad \text{and} \quad k(\texttt{apple}, \texttt{orange}) = -5$$



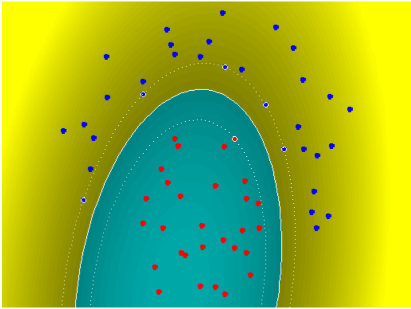
SVM using kernels example



Linear kernel (no kernel)

$K(a,b) = \boldsymbol{a}^T\boldsymbol{b} = \phi(a)^T\phi(b)$

$\phi(a) = a$



2nd order polynomial kernel

$K(a,b) = (\boldsymbol{a}^T\boldsymbol{b})^2$

What makes a valid kernel?

$k(a,b) = \phi(a)^T \phi(b)$

A kernel is valid if it corresponds to an inner product in some feature space. An equivalent formulation is given by Mercer's theorem.

Mercer's theorem

A kernel is valid if it gives rise to a symmetric, positive semidefinite kernel matrix $\boldsymbol{K}$ for any input data $\boldsymbol{X}$.

Kernel matrix (also known as Gram matrix) $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is defined as

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

What happens if we use a non-valid kernel?

Our optimization problem might become non-convex, so we may not get a globally optimal solution.

Kernel preserving operations

Let $k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be kernels, with $\mathcal{X} \subseteq \mathbb{R}^N$. Then the following functions k are kernels as well:

- $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) + k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
- $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = c \cdot k_1(\boldsymbol{x}_1, \boldsymbol{x}_2)$, with $c > 0$
- $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) \cdot k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
- $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_3(\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2))$, with the kernel k_3 on $\mathcal{X}' \subseteq \mathbb{R}^M$ and $\phi : \mathcal{X} \to \mathcal{X}'$
- $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1 \boldsymbol{A} \boldsymbol{x}_2$, with $\boldsymbol{A} \in \mathbb{R}^N \times \mathbb{R}^N$ symmetric and positive semidefinite

Examples of kernels

- Polynomial:

$$k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^T \boldsymbol{b})^p \quad \text{or} \quad (\boldsymbol{a}^T \boldsymbol{b} + 1)^p$$

$$e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \dots$$

- Gaussian kernel:

$$k(\boldsymbol{a}, \boldsymbol{b}) = \exp\left(-\frac{\|\boldsymbol{a} - \boldsymbol{b}\|^2}{2\sigma^2}\right)$$

- Sigmoid:

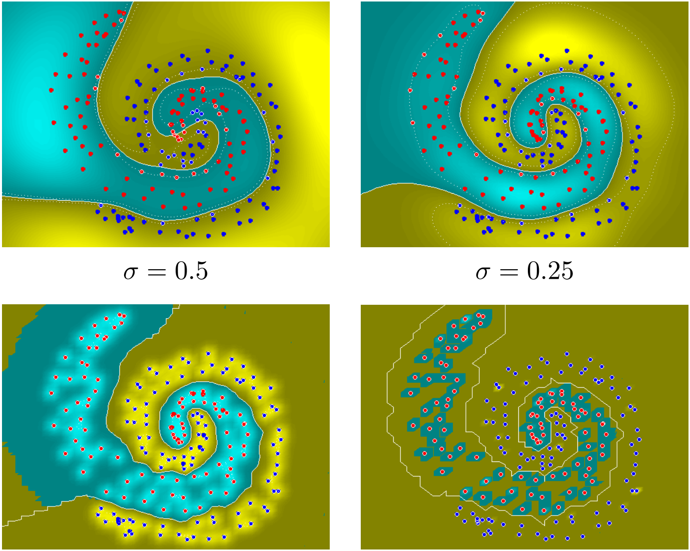
$$k(\boldsymbol{a}, \boldsymbol{b}) = \tanh(\kappa\, \boldsymbol{a}^T \boldsymbol{b} - \delta) \quad \text{for } \kappa, \delta > 0$$

  In fact, the sigmoid kernel is not PSD, but still works well in practice.

Some kernels introduce additional hyperparameters, that affect the behavior of the algorithm.

Note, that the sigmoid kernel is different from the sigmoid function from *Linear Classification*.

Gaussian kernel (C=1000)



$\sigma = 0.5$

$\sigma = 0.25$

$\sigma = 0.05$

$\sigma = 0.005$

Classifying a new point with kernelized SVM

We denote the set of support vectors as $\mathcal{S}$ (points $\boldsymbol{x}_i$ for which holds $0 < \alpha_i \leq C$). Note: If $0 < \alpha_i < C$ then $\xi_i = 0$; if $\alpha_i = C$ then $\xi_i > 0$.

From the complementary slackness condition, points $\boldsymbol{x}_i \in \mathcal{S}$ with $\xi_i = 0$ must satisfy

$$y_i \left(\sum_{\{j \mid \boldsymbol{x}_j \in \mathcal{S}\}} \alpha_j y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \right) = 1$$

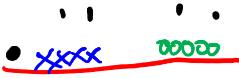
$$w = \sum_i \alpha_i y_i \phi(x_i)$$

$$\alpha_i^* f_i(\theta^*) = 0$$

Like for the regular SVM, the bias can be recovered as

$$b = y_i - \left(\sum_{\{j \mid \boldsymbol{x}_j \in \mathcal{S}\}} \alpha_j y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)$$

Thus, a new point $\boldsymbol{x}$ can be classified as



$$h(\boldsymbol{x}) = \operatorname{sign} \left(\sum_{\{j \mid \boldsymbol{x}_j \in \mathcal{S}\}} \alpha_j y_j \underbrace{k(\boldsymbol{x}_j, \boldsymbol{x})}_{x_j^T x} + b \right)$$

$+ 10$

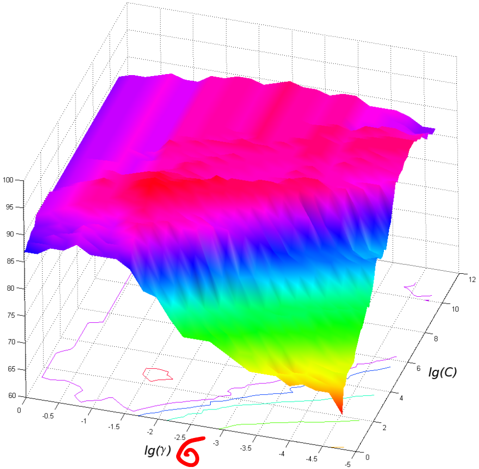
$-1\ 10$

$↑$

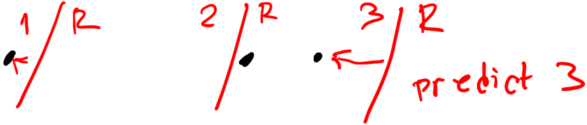
$-\ 3$

How to choose the hyperparameters?

The best setting of the penalty parameter C and the kernel hyperparameters (e.g., γ or σ) can be determined by performing cross validation with random search over the parameter space.



Multiple classes



The standard SVM model cannot handle multiclass data.
Two approaches to address this issue are:

One-vs-rest: Train C SVM models for C classes, where each SVM is being trained for classification of one class against all the remaining ones. The winner is then the class, where the distance from the hyperplane is maximal.

One-vs-one: Train $\binom{C}{2}$ classifiers (all possible pairings) and evaluate all. The winner is the class with the majority vote (votes are weighted according to the distance from the margin).

1	2	1
2	3	3
1	3	1

) 1 predict

Summary

- Support Vector Machine is a linear binary classifier that, motivated by learning theory, maximizes the margin between the two classes.
- The SVM objective is convex, so a globally optimal solution can be obtained.
- The dual formulation is a quadratic programming problem, that can be solved efficiently, e.g. using standard QP libraries.
- Soft-margin SVM with slack variables can deal with noisy data. Smaller values for the penalty C lead to a larger margin at the cost of misclassifying more samples.
- Linear soft-margin SVM (= hinge loss formulation) can be solved directly using gradient descent.
- We can obtain a nonlinear decision boundary by moving to an implicit high-dimensional feature space by using the kernel trick. This only works in the dual formulation.

Reading material

Reading material

- Bishop: chapters 7.1.0, 7.1.1, 7.1.2

Acknowledgements

- Slides are based on an older version by S. Urban