

# \*\*\* Applied Machine Learning Fundamentals \*\*\*

## Support Vector Machines

M. Sc. Daniel Wehner

SAP SE

Winter term 2019/2020



Find all slides on [GitHub](#)

# Lecture Overview

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Probability Density Estimation
Unit V	Regression
Unit VI	Classification I
Unit VII	Evaluation
<b>Unit VIII</b>	<b>Classification II</b>
Unit IX	Clustering
Unit X	Dimensionality Reduction

# Agenda for this Unit

## 1 Linear SVMs

- Introduction
- Discriminant Functions
- Linear Separability
- Maximum Margin Classifiers
- Lagrangian Optimization

## 2 Kernel SVMs

- Non-linear Data
- Feature Mapping
- Kernels
- Mercer's Condition

## 3 Soft-Margin SVMs

- Overlapping Data
- Slack Variables

## 4 Multi-Class Classification

- Multiple Classes
- One-vs-Rest (OVR)
- One-vs-One (OVO)

## 5 Wrap-Up

- Summary
- Self-Test Questions
- Lecture Outlook
- Recommended Literature and further Reading
- Meme of the Day

Section:  
**Linear SVMs**



# What is a Support Vector Machine (SVM)?

- A support vector machine is a **binary classifier** [Vapnik and Chervonenkis]
  - The classes are denoted by  $\{-1; +1\}$
  - Techniques for multi-class classification:
    - **OVR (One-vs-Rest)**
    - **OVO (One-vs-One)**
- An SVM finds the **best separating hyperplane**
- Why is it called '*machine*'?
  - It's no physical machine, it's a mathematical construct
  - 'Machine' refers to 'Machine Learning'

What is the best separating hyperplane?

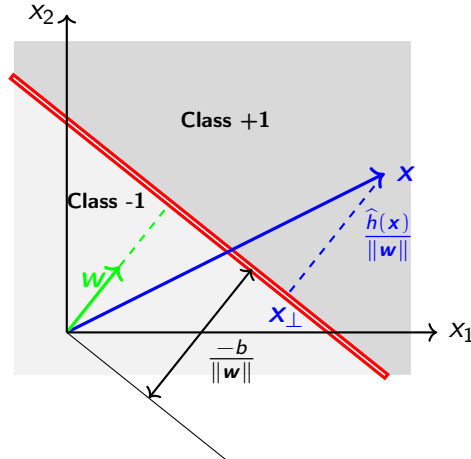
# Discriminant Functions

- The simplest discriminant is a linear function of the form:

$$\hat{h}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = \sum_{j=1}^m w_j x_j + b = w_1 x_1 + w_2 x_2 + \cdots + w_m x_m + b \quad (1)$$

- $\theta = \{\mathbf{w}, b\}$ :  $\mathbf{w}$  is called the **weight vector** and  $b$  is the **bias**
- An input vector  $\mathbf{x}$  is assigned class  $\mathcal{C}_1$  if  $\hat{h}(\mathbf{x}) \geq 0$ , class  $\mathcal{C}_2$  otherwise
- The **decision boundary** is defined by the relation:  $\hat{h}(\mathbf{x}) = 0$
- The boundary is a  $(D - 1)$ -dimensional hyperplane within the  $D$ -dimensional input space

## Discriminant Functions (Ctd.)



## Discriminant Functions (Ctd.)

- Consider two points  $\mathbf{x}_A$  and  $\mathbf{x}_B$  which lie on the decision surface
- Since  $\hat{h}(\mathbf{x}_A) = \hat{h}(\mathbf{x}_B) = 0$ , we have  $\mathbf{w}^\top(\mathbf{x}_A - \mathbf{x}_B) = 0$ , hence  $\mathbf{w}$  is **orthogonal** to every vector lying within the decision surface
- $\mathbf{w}$  determines the orientation of the decision surface
- Similarly, if  $\mathbf{x}$  is a point on the decision surface, then  $\hat{h}(\mathbf{x}) = 0$  and the normal distance from the origin to the decision surface is given by:

$$\frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|} = 0 \Leftrightarrow \frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|} = -\frac{b}{\|\mathbf{w}\|} \quad (2)$$

- $b$  controls the offset from the origin



## Discriminant Functions (Ctd.)

- $\hat{h}(\mathbf{x})$  gives a **signed measure** of the perp. distance of  $\mathbf{x}$  to the boundary
- Consider an arbitrary point  $\mathbf{x}$  and its orth. projection  $\mathbf{x}_\perp$  onto the surface

$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \quad (3)$$

- Multiplying both sides by  $\mathbf{w}^\top$  and adding  $b$ , and making use of:

$$\hat{h}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \quad \text{and} \quad \hat{h}(\mathbf{x}_\perp) = \mathbf{w}^\top \mathbf{x}_\perp + b = 0$$

- We get:

$$r = \frac{h(\mathbf{x})}{\|\mathbf{w}\|} \quad (4)$$

# Linear Separability

- We have  $n$  input vectors  $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$
- With corresponding target values  $y^{(1)}, y^{(2)}, \dots, y^{(n)}$  where  $y^{(i)} \in \{-1, +1\}$
- New data points are classified according to the sign of  $\hat{h}(\mathbf{x})$ :  $\text{sign}(\hat{h}(\mathbf{x}))$

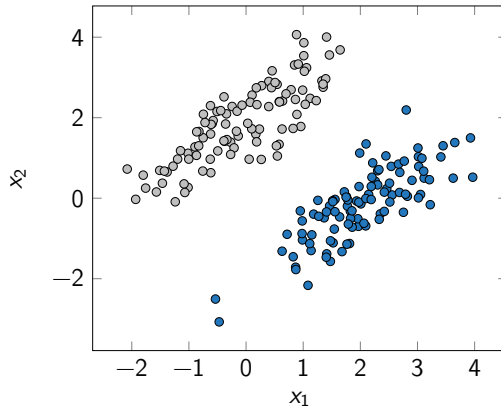
A data set is **linearly separable** in feature space, if  $\exists(\mathbf{w}, b)$  such that

$$\hat{h}(\mathbf{x}^{(i)}) = \mathbf{w}^T \mathbf{x}^{(i)} + b > 0 \quad \forall \mathbf{x}^{(i)} \text{ with } y^{(i)} = +1 \quad (5)$$

$$\hat{h}(\mathbf{x}^{(i)}) = \mathbf{w}^T \mathbf{x}^{(i)} + b < 0 \quad \text{otherwise } (y^{(i)} = -1) \quad (6)$$

This can also be written as:  $y^{(i)} \hat{h}(\mathbf{x}^{(i)}) > 0 \quad \forall i$

## Example Data Set (linearly separable)



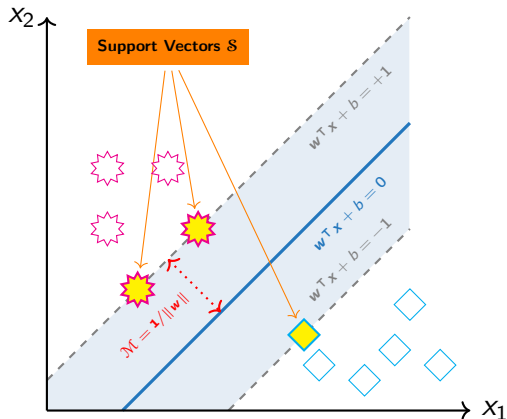
- This data set is **linearly separable** (you can find a straight line to separate the two classes)
- The possible number of hyperplanes is infinite...
- Which hyperplane should be chosen?

# Maximum Margin Classifiers

- An SVM is a so-called **maximum margin classifier**
- It maximizes the margin  $\mathcal{M}$

$$\max_{w,b} \mathcal{M}$$

- The larger  $\mathcal{M}$  the less likely are false predictions
- Only the **support vectors** determine the hyperplane



## Maximum Margin Classifiers (Ctd.)

- Recall the perpendicular distance of a point  $\mathbf{x}$  to the hyperplane:

$$\frac{|\hat{h}(\mathbf{x})|}{\|\mathbf{w}\|} \quad (7)$$

- Furthermore, we are only interested in solutions for which all data points are correctly classified, i. e.  $y^{(i)}\hat{h}(\mathbf{x}^{(i)}) > 0 \ \forall i$ , thus the distance is given by:

$$\frac{y^{(i)}\hat{h}(\mathbf{x}^{(i)})}{\|\mathbf{w}\|} = \frac{y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|} \quad (8)$$

## Maximum Margin Classifiers (Ctd.)

- The margin is given by the perp. distance to the closest data point  $\mathbf{x}^{(i)}$
- We wish to optimize the parameters  $\mathbf{w}$  and  $b$  to maximize this distance
- We have to solve:

$$\mathbf{w}^*, b^* = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \overbrace{\min_i \{y^{(i)} (\mathbf{w}^\top \mathbf{x}^{(i)} + b)\}}^{\text{'closest data points' }} \right\} \quad (9)$$

- Note that  $1/\|\mathbf{w}\|$  does not depend on  $i$
- A direct solution of this optimization would be very complex  $\Rightarrow$  rewrite!

## Maximum Margin Classifiers (Ctd.)

- We note that rescaling  $\mathbf{w}$  and  $b$  by a factor  $\zeta$  **does not change the distance** to the decision boundary
- Therefore, for the points that are closest to the surface, we can set:

$$y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) = 1 \quad (10)$$

- In this case, all data points  $\mathbf{x}^{(i)}$  satisfy the constraint:

$$y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 \quad i = 1, 2, \dots, n \quad (11)$$

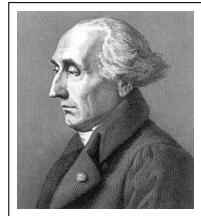
- It is sufficient to solve:  $\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$  ( $1/2$  for later mathematical convenience)

## Maximum Margin Classifiers (Ctd.)

$$\mathbf{w}^*, b^* = \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (12)$$

- This is a **quadratic optimization (QP)** problem
- A global optimum exists due to **convexity**
- How to solve such problems?

**Lagrangian optimization**  
(named after Joseph-Louis Lagrange)





# Lagrangian Optimization: A simple Example

- Lagrangian optimization is optimization **subject to constraints**
- **Example:**

$$\overbrace{f(x_1, x_2) = 1 - x_1^2 - x_2^2}^{\text{function to optimize } f(\mathbf{x})} \quad \text{s. t.} \quad \overbrace{g(x_1, x_2) = x_1 + x_2 - 1 = 0}^{\text{linear constraint } g(\mathbf{x}) = 0} \quad (13)$$

- To find a solution we have to formulate the **Lagrangian equation**:  
General form:  $\mathcal{L}(\mathbf{x}, \alpha) = f(\mathbf{x}) + \alpha g(\mathbf{x})$

$$\mathcal{L}(\mathbf{x}, \alpha) = 1 - x_1^2 - x_2^2 + \alpha(x_1 + x_2 - 1) \quad (14)$$

## Lagrangian Optimization: A simple Example (Ctd.)

$$\mathcal{L}(\mathbf{x}, \alpha) = 1 - x_1^2 - x_2^2 + \alpha(x_1 + x_2 - 1)$$

We determine the partial derivatives w. r. t.  $x_1$ ,  $x_2$  and  $\alpha$  and set them to zero:

$$\frac{\partial \mathcal{L}(\mathbf{x}, \alpha)}{\partial x_1} = -2x_1 + \alpha \stackrel{!}{=} 0 \quad (15)$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \alpha)}{\partial x_2} = -2x_2 + \alpha \stackrel{!}{=} 0 \quad (16)$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \alpha)}{\partial \alpha} = x_1 + x_2 - 1 \stackrel{!}{=} 0 \quad (17)$$

## Lagrangian Optimization: A simple Example (Ctd.)

- Solving the first two equations for  $x_1$  and  $x_2$ , respectively, we get:

$$x_1 = 1/2\alpha \quad (18)$$

$$x_2 = 1/2\alpha \quad (19)$$

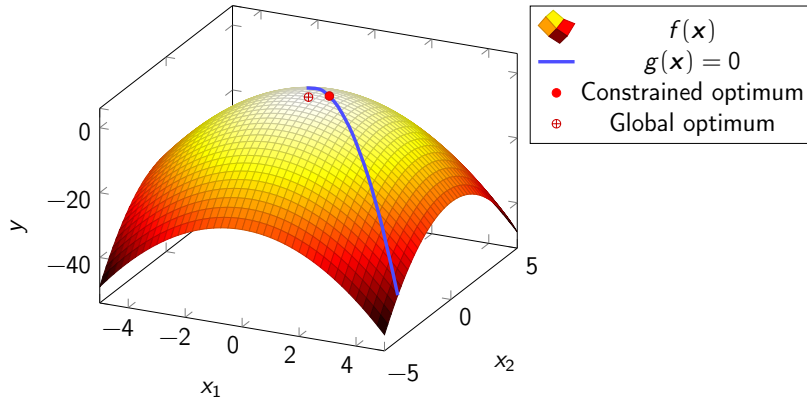
- Substitution into the third equation  $x_1 + x_2 - 1 = 0$ :

$$\tilde{\mathcal{L}}(\alpha) = 1/2\alpha + 1/2\alpha - 1 = 0 \Leftrightarrow \alpha = 1 \quad (20)$$

- Finally, we get:

$$x_1 = 1/2 \quad x_2 = 1/2$$

## Lagrangian Optimization: A simple Example (Ctd.)



# SVM Parameter Optimization

We have to solve the Lagrangian:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \overbrace{1/2 \|\mathbf{w}\|^2}^{f(\mathbf{x})} - \sum_{i=1}^n \alpha_i \overbrace{[y^{(i)} \cdot (\mathbf{w}^\top \mathbf{x}^{(i)} + b) - 1]}^{g(\mathbf{x})=0} \quad (21)$$

- $\alpha$  is a vector of **Lagrangian multipliers**
- There is **one constraint per data point!**
- The Lagrangian multipliers will be non-zero for all support vectors

# SVM Parameter Optimization (Ctd.)

We have to compute the partial derivatives w. r. t.  $\mathbf{w}$  and  $b$  and set them to zero:

linear combination of input!

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \alpha_i y^{(i)} \mathbf{x}^{(i)} \stackrel{!}{=} 0 \Rightarrow \boxed{\mathbf{w} = \sum_{i=1}^n \alpha_i y^{(i)} \mathbf{x}^{(i)}} \quad (22)$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^n \alpha_i y^{(i)} \stackrel{!}{=} 0 \Rightarrow \boxed{\sum_{i=1}^n \alpha_i y^{(i)} = 0} \quad (23)$$



## SVM Parameter Optimization (Ctd.)

As a next step the partial derivatives are substituted in into  $\mathcal{L}$ :

$$\begin{aligned}\tilde{\mathcal{L}}(\alpha) = & \frac{1}{2} \left( \sum_{i=1}^n \alpha_i y^{(i)} \mathbf{x}^{(i)} \right) \left( \sum_{j=1}^n \alpha_j y^{(j)} \mathbf{x}^{(j)} \right) - \left( \sum_{i=1}^n \alpha_i y^{(i)} \mathbf{x}^{(i)} \right) \left( \sum_{j=1}^n \alpha_j y^{(j)} \mathbf{x}^{(j)} \right) \\ & - \sum_{i=1}^n \alpha_i y^{(i)} b + \sum_{i=1}^n \alpha_i\end{aligned}\quad (24)$$

$$= \boxed{\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle} \quad \text{s. t. } \alpha_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^n \alpha_i y^{(i)} = 0 \quad (25)$$

**Wolfe dual**



## SVM Parameter Optimization (Ctd.)

- Once we know  $\alpha$ , we can determine  $b$  by noting that any support vector satisfies  $y^{(i)} \widehat{h}(\mathbf{x}^{(i)}) = 1$ : ( $\mathcal{S} \equiv$  indices of support vectors)

$$y^{(i)} \left( \sum_{j \in \mathcal{S}} \alpha_j y^{(j)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + b \right) = 1 \quad (26)$$

- Average over all support vectors to compute  $b$ : ( $n_{\mathcal{S}} \equiv$  number of support vectors)

$$b = \frac{1}{n_{\mathcal{S}}} \sum_{i \in \mathcal{S}} \left( y^{(i)} - \sum_{j \in \mathcal{S}} \alpha_j y^{(j)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \right) \quad (27)$$



## Updated Decision Rule

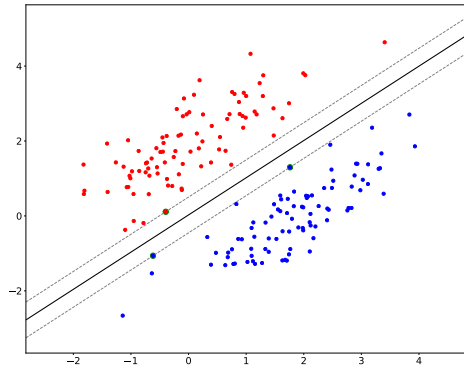
- Given our derivations, we can rewrite the SVM decision rule as follows:

$$h(\mathbf{x}) = \text{sign} \left( \sum_{i \in \mathcal{S}} \alpha_i y^{(i)} \langle \mathbf{x}^{(i)}, \mathbf{x} \rangle + b \right) \quad (28)$$

- $\mathbf{x}$  is an unknown instance for which the class label is not known

**Since all  $\alpha_i$  will be zero for non-support vectors, the decision for a class depends on the support vectors only!** This makes predictions fast, even for large data sets. **The number of support vectors can also be used as an evaluation criterion.**

# Linear SVM: Example

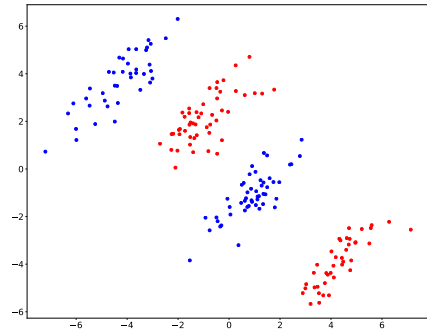


Section:  
**Kernel SVMs**



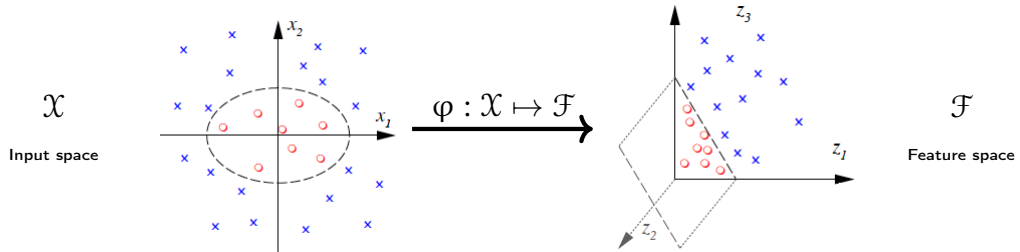
# Non-Linear SVMs / Non-Linear Separability

- So far we have assumed **linear separability** of the data
- What if the data is not **linearly separable**?  
(which will be the case in practice...)
- We cannot find a straight line...
- *Remedy*  $\implies$  **Feature maps, Kernels**



# Feature Mapping

The mapping function  $\varphi$  maps from input space  $\mathcal{X}$  to feature space  $\mathcal{F}$ :



$$\varphi(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2) = (z_1, z_2, z_3)$$

## Feature Mapping (Ctd.)

- A **feature map** **explicitly** transforms the data to a higher dimension where classification becomes easier
- Computing the feature map can – from a computational point of view – become very expensive
- And how do you know how many dimensions to add? What transformations should be used?
- **A more tractable solution is required  $\Rightarrow$  Kernels**

# What is a Kernel?

- A kernel can be considered a **similarity function**
- Many algorithms have been '*kernelized*', e. g. *Kernel PCA*, *Kernel SVM*
- Think of it as projecting the data in a higher dimensional space to make it linearly separable

A kernel allows the SVM to operate in a **high-dimensional, implicit feature space** without ever computing the coordinates of the data in that space, but rather by simply computing the **inner products** between the images of **all pairs of data** in the feature space.  $\Rightarrow$  **Kernel trick** [Wikipedia]

# What is a Kernel? (Ctd.)

- The explicit computation of a feature map  $\varphi(\mathbf{x})$  is avoided...
- ...by replacing the dot product with the kernel  $\mathcal{K}$ :

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') \Leftrightarrow \varphi(\mathbf{x})^\top \varphi(\mathbf{x}') \quad (29)$$

- Instead of mapping features explicitly, we calculate the **Gram matrix**  $\mathbf{K} \in \mathbb{R}^{n \times n}$ , where:

$$K_{ij} = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad (30)$$



# Well-known Kernels

- Linear kernel

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}' \quad (31)$$

- Polynomial kernel

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^p \quad (32)$$

- Radial-Basis-Function (RBF) kernel

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right\} = \exp \{ -\gamma \|\mathbf{x} - \mathbf{x}'\|^2 \} \quad (33)$$

# Power of Kernels

- Suppose  $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^m$  with  $m = 2$
- Polynomial feature mapping ( $c = 0$ ):

$$\varphi(\mathbf{x}) = [x_1x_1, x_1x_2, x_2x_1, x_2x_2]^\top \quad \varphi(\mathbf{x}') = [x'_1x'_1, x'_1x'_2, x'_2x'_1, x'_2x'_2]^\top \quad (34)$$

- Using a polynomial kernel:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^2 = \left( \sum_{i=1}^m x_i x'_i \right) \left( \sum_{j=1}^m x_j x'_j \right) = \sum_{i=1}^m \sum_{j=1}^m (x_i x_j) (x'_i x'_j) = \varphi(\mathbf{x})^\top \varphi(\mathbf{x}') \quad (35)$$

**We need  $\mathcal{O}(n^2)$  to compute  $\varphi(\mathbf{x})$  and  $\varphi(\mathbf{x}')$ , but  $\mathcal{O}(n)$  to compute  $\mathcal{K}(\mathbf{x}, \mathbf{x}')$**

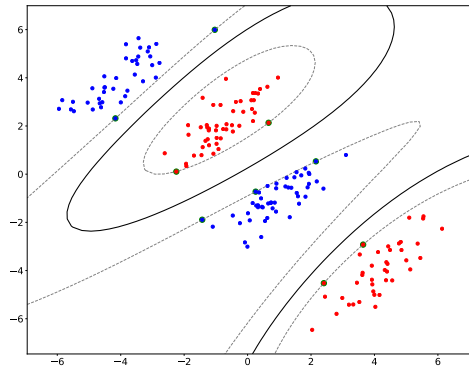
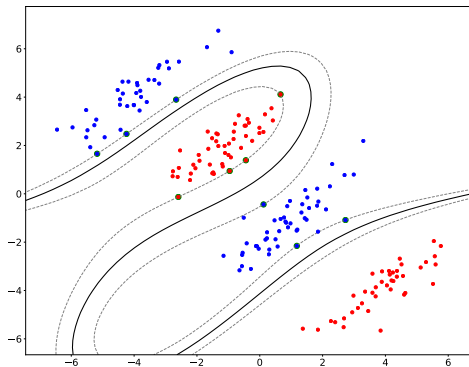
# Incorporating the Kernel Function

- The kernel function  $\mathcal{K}$  replaces each occurrence of  $\mathbf{x}^\top \mathbf{x}'$
- **Example:**

$$\tilde{\mathcal{L}} = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad (36)$$

$$h(\mathbf{x}) = \text{sign} \left( \sum_{i \in \mathcal{S}} \alpha_i y^{(i)} \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}) + b \right) \quad (37)$$

# Polynomial Kernel vs. RBF Kernel



# Mercer's Condition

- A kernel is valid, if it fulfills **Mercer's condition**
- This is the case, if for all **square-integrable functions**  $g(\mathbf{x})...$

$$\int_{-\infty}^{\infty} |g(\mathbf{x})|^2 d\mathbf{x} < \infty \quad (38)$$

- ...it holds:

$$\iint g(\mathbf{x}) \mathcal{K}(\mathbf{x}, \mathbf{x}') g(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0 \quad (39)$$



## Mercer's Condition (Ctd.)

- Suppose  $\mathcal{K}$  is a valid kernel and let  $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$  be given
- For any vector  $\mathbf{z} \in \mathbb{R}^n$ :

$$\mathbf{z}^\top \mathbf{K} \mathbf{z} = \sum_i \sum_j z_i K_{ij} z_j = \sum_i \sum_j z_i \varphi(\mathbf{x}^{(i)})^\top \varphi(\mathbf{x}^{(j)}) z_j \quad (40)$$

$$= \sum_i \sum_j z_i \sum_k (\varphi(\mathbf{x}^{(i)}))_k (\varphi(\mathbf{x}^{(j)}))_k z_j = \sum_k \sum_i \sum_j z_i (\varphi(\mathbf{x}^{(i)}))_k (\varphi(\mathbf{x}^{(j)}))_k z_j \quad (41)$$

$$= \sum_k \left( \sum_i z_i \varphi(\mathbf{x}^{(i)})_k \right)^2 \geq 0 \implies \mathbf{K} \geq 0 \quad (42)$$

- $\mathbf{K} \geq 0$  means that matrix  $\mathbf{K}$  must be **positive semi-definite (psd)**

# Mercer's Condition (Ctd.)

## Mercer's Theorem:

$\mathcal{K}$  is a valid kernel, iff for any set of  $n$  training examples  $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$  kernel matrix  $\mathbf{K} \in \mathbb{R}^{n \times n}$  is **positive semi-definite**. The kernel is then called **Mercer kernel**.

- This entails:  $K_{ij} \geq 0 \ \forall i, j$
- **Example:**
  - $\mathcal{K}(\mathbf{x}, \mathbf{x}) = -1 \neq \varphi(\mathbf{x})^\top \varphi(\mathbf{x})$
  - $\mathcal{K}$  cannot be a valid kernel

## Constructing new Kernels

- It is not always easy to check if Mercer's condition is satisfied, but it is possible to construct **new kernels out of known ones**
- If  $\mathcal{K}_1(\mathbf{x}, \mathbf{x}')$  and  $\mathcal{K}_2(\mathbf{x}, \mathbf{x}')$  are valid kernels, so are:
  - $c \cdot \mathcal{K}_1(\mathbf{x}, \mathbf{x}')$
  - $\mathcal{K}_1(\mathbf{x}, \mathbf{x}') + \mathcal{K}_2(\mathbf{x}, \mathbf{x}')$
  - $\mathcal{K}_1(\mathbf{x}, \mathbf{x}') \cdot \mathcal{K}_2(\mathbf{x}, \mathbf{x}')$
  - $f(\mathbf{x}) \cdot \mathcal{K}_1(\mathbf{x}, \mathbf{x}') \cdot f(\mathbf{x}')$
  - etc.



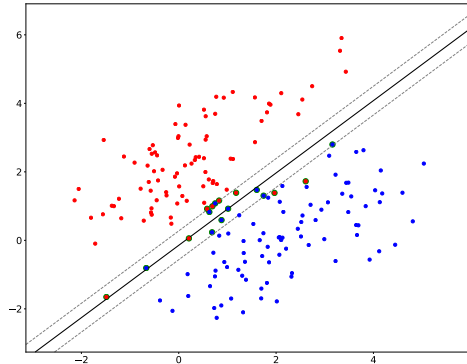
Section:  
**Soft-Margin SVMs**



# Overlapping Distributions

- We assumed linearly separable data  
⇒ **SVM gives exact solution**
- **But:** The classes may overlap  
⇒ **Exact separation leads to poor generalization**
- **Soft-margin SVM:** Allow some data points to be misclassified
- To this end, a penalty is introduced:
  - Misclassifications are penalized
  - This penalty increases linearly with the distance from the decision boundary
- This is done using **slack variables**

# Overlapping Distributions: Example



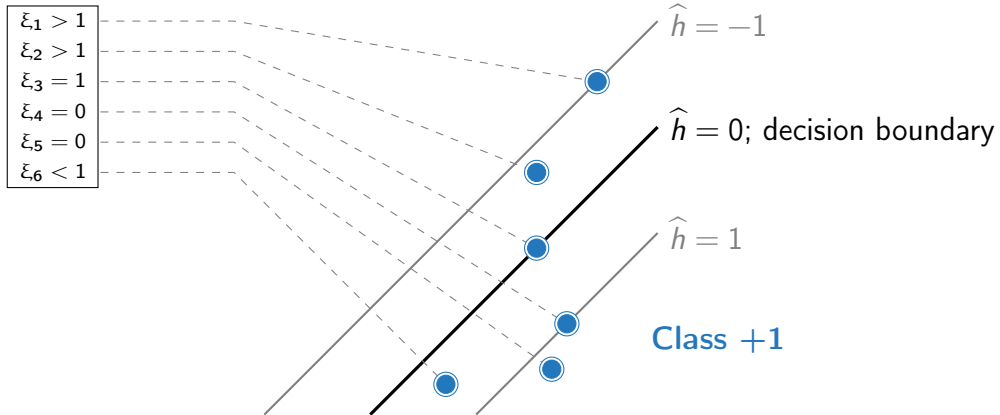
# Slack Variables

- The slack is denoted by  $\xi_i$  (where  $\xi_i \geq 0; i = 1, \dots, n$ ), one per data point
- Different cases:**
  - $\xi_i = 0$  if  $\mathbf{x}^{(i)}$  is on or inside the correct margin boundary
  - $0 < \xi_i < 1$  if  $\mathbf{x}^{(i)}$  lies inside the margin, but on the correct side
  - $\xi_i = 1$  if  $\mathbf{x}^{(i)}$  is on the decision boundary
  - $\xi_i > 1$  if  $\mathbf{x}^{(i)}$  lies on the wrong side of the decision boundary (misclassification)
- The classification constraints are replaced with:

$$y^{(i)} \hat{h}(\mathbf{x}^{(i)}) \geq 1 - \xi_i \quad i = 1, \dots, n \quad (43)$$

- We get a **soft-margin classifier**

## Slack Variables (Ctd.)



## Soft SVM Parameter Optimization

- We want to maximize the margin, while softly **penalizing points which lie on the wrong side of the boundary**:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s. t.} \quad y^{(i)}\hat{h}(\mathbf{x}^{(i)}) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \forall i \quad (44)$$

- $C > 0$  controls the '*degree of softness*', the larger  $C$  the more we penalize
- The Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b, \alpha, \mu) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \{y^{(i)}\hat{h}(\mathbf{x}^{(i)}) - 1 + \xi_i\} - \sum_{i=1}^n \mu_i \xi_i \quad (45)$$

## Soft SVM Parameter Optimization (Ctd.)

- It turns out that the dual objective function looks exactly the same:

$$\tilde{\mathcal{L}}(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \quad (46)$$

- But the constraints differ slightly:

$$1) \quad \sum_{i=1}^n \alpha_i y^{(i)} = 0 \quad (47)$$

$$2) \quad \boxed{0 \leq \alpha_i \leq C \quad i = 1, \dots, n} \quad (48)$$

- Constraint 2) is called **boxed constraint**

Section:  
**Multi-Class Classification**



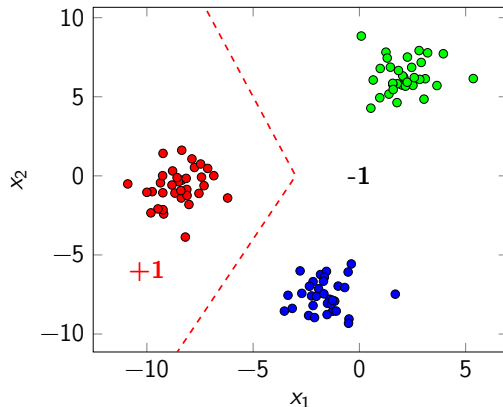


# Multi-Class Classification

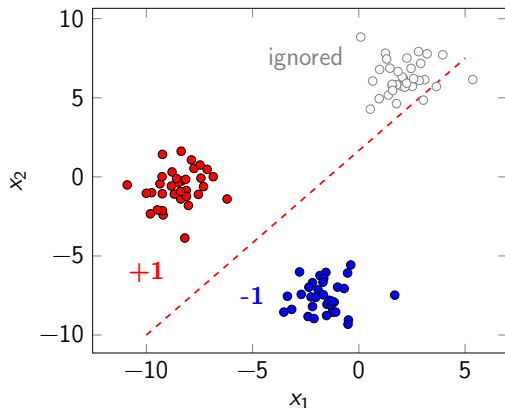
- An SVM can handle two classes only, namely  $-1$  and  $+1$
- **What if there are more than two classes?**
- Two common techniques:
  - **One-vs-Rest (OVR)**  $\Rightarrow$  One-against-All
  - **One-vs-One (OVO)**  $\Rightarrow$  Pairwise classification
- Several classifiers are trained
- During prediction the classifiers **vote for the correct class**
- Such techniques can be used for all binary classifiers (e. g. logistic regression)

# Multi-Class Classification: One-vs-Rest (OVR)

- **Train one classifier per class**  
(expert for that class)
- We get  $|\mathcal{C}|$  classifiers
- The  $k$ -th classifier learns to distinguish the  $k$ -th class from all the others
- Set the labels of examples from class  $k$  to  $+1$ , all the others to  $-1$



## Multi-Class Classification: One-vs-One (OVO)



- Train one classifier for each pair of classes
- We get  $\binom{|C|}{2}$  classifiers
- Ignore all other examples that do not belong to either of the two classes
- **Voting:** Count how often each class wins; the class with the highest score is predicted

Section:  
**Wrap-Up**



# Summary

- SVMs assume the data to be **linearly separable**
- **Generalization guarantee:** SVMs are **maximum margin** classifiers
- The set of **support vectors** defines the decision boundary
- We have to solve a quadratic optimization problem to obtain the support vectors which are needed for prediction
- **Important concept:** Kernels (cf. Mercer's condition)
- Slack variables allow for **soft-margin classification**
- Apply multi-class classification techniques like OVR / OVO if you have more than two classes



# Self-Test Questions

- 1 What is a maximum-margin classifier?
- 2 Which data points are needed for prediction? How do we get them?
- 3 What is a kernel? Can every function serve as a kernel?
- 4 What prerequisite allows for the usage of kernels?
- 5 Name famous kernels and write down the equation to compute them!
- 6 What is slack? What can we do with it?

# What's next...?

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Probability Density Estimation
Unit V	Regression
Unit VI	Classification I
Unit VII	Evaluation
Unit VIII	Classification II
Unit IX	Clustering
Unit X	Dimensionality Reduction

# Recommended Literature and further Reading I



## [1] Pattern Recognition and Machine Learning

*Christopher Bishop. Springer. 2006.*

→ [Link](#), cf. chapter 7



## [2] Machine Learning: A Probabilistic Perspective

*Kevin Murphy. MIT Press. 2012.*

→ [Link](#), cf. chapter 14.5



# Meme of the Day



Thank you very much for the attention!

**Topic:** \*\*\* Applied Machine Learning Fundamentals \*\*\* Support Vector Machines

**Term:** Winter term 2019/2020

**Contact:**

M.Sc. Daniel Wehner

SAP SE

[daniel.wehner@sap.com](mailto:daniel.wehner@sap.com)

Do you have any questions?