

# Kernel Learning And Its Application In Nonlinear Support Vector Machines

Anastasia S., Sven N., Joshua S.

Otto-Friedrich-University Bamberg

*anastasia.sinitsyna@stud.uni-bamberg.de*  
*sven-jason-waldemar.nekula@stud.uni-bamberg.de*  
*joshua-guenter.simon@stud.uni-bamberg.de*

June 9, 2021

## 1 Introduction

- Linearly separable data classes
- Similarity, dot product and vector norm
- Hyperplane classifiers - Solving an optimization problem

## 2 Linear SVMs

- Maximum margin separator
- Limitations

## 3 Nonlinear SVMs

- The kernel trick
- Solving non linearly separable classification problems
- A closer look at kernels

## 4 More kernel applications

# Linearly separable data classes

First, let's consider a given data set  $\mathcal{X}$  of labeled points (inputs) with individual labels  $y_i \in \{-1, 1\}$ , e.g.  $(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \{-1, 1\}$ .

Our goal is to implement a classification method, which is able to classify new and unlabeled data points with the right or "best" label.

# Linearly separable data classes

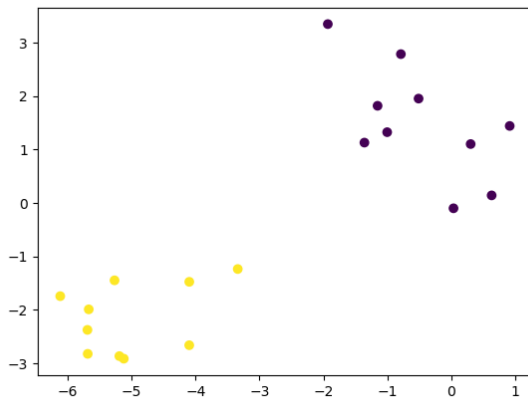


Figure: An example for linearly separable data.

# Linearly separable data classes

In machine learning, a well established classification method are the so called **Support Vector Machines** (SVM). Developed by Vladimir Vapnik and his coworkers in the 1990s, SVMs are still a relevant topic and an even more powerful tool for **classification** and **regression**.

To perform a classification, a similarity measure is needed. Finding a suitable measure is a core problem of machine learning. For now let's consider

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\rightarrow \mathbb{R} \\ (x, x') &\mapsto k(x, x') \end{aligned} \tag{1}$$

where  $k$  is a function that, given two patterns  $x$  and  $x'$ , returns a real number characterizing their similarity. This function  $k$  is called a **kernel**. Unless stated otherwise,  $k(x, x') = k(x', x)$ .

# Dot product and vector norm

A simple type of similarity measure is a **dot product**. Given two vectors  $x, x' \in \mathbb{R}^n$  the canonical dot product is defined as

$$\langle x, x' \rangle = (x')^T x = \sum_{i=1}^n x_i x'_i. \quad (2)$$

Futhermore this allows a calculation of the **norm** (length) of a single vector  $x$  as

$$\|x\| = \sqrt{\langle x, x \rangle}. \quad (3)$$

More properties of vector spaces, dot products and norms can be found in [Liesen, 2015].

# Hyperplane classifiers

The underlying learning algorithm of SVMs yields to find a hyperplane in some dot product space  $\mathcal{H}$ , which separates the data. A hyperplane of the form

$$\langle w, x \rangle + b = 0 \quad (4)$$

where  $w \in \mathcal{H}$ ,  $b \in \mathbb{R}$  shall be considered [Schölkopf, 2002](p. 11). Furthermore decision functions

$$f(x) = \text{sgn}(\langle w, x \rangle + b) \quad (5)$$

can be assigned.



# Hyperplane classifiers - A constrained optimization problem

The **optimal hyperplane** can be calculated by finding the normal vector that leads to the largest margin. Thus we need to solve the optimization problem

$$\begin{aligned} \min_{w \in \mathcal{H}, b \in \mathbb{R}} \quad & \tau(w) = \frac{1}{2} \|w\|^2 \\ \text{subject to} \quad & y_i (\langle w, x \rangle + b) \geq 1 \quad \forall i = 1, \dots, m. \end{aligned} \tag{6}$$

The constraints in (6) ensure that  $f(x_i)$  will be  $+1$  for  $y_i = +1$  and  $-1$  for  $y_i = -1$ . The  $\geq 1$  on the right hand side of the constraints effectively fixes the scaling of  $w$ . This leads to the maximum margin hyperplane. A detailed explanation can be found in [Schölkopf, 2002](Chap 7).

# Hyperplane classifiers - Lagrangian

The constrained optimization problem in (6) can be re-written using the method of Lagrange multipliers. This leads to the Lagrangian

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i (\langle w, x \rangle + b) - 1) \quad (7)$$

subject to  $\alpha_i \geq 0 \ \forall i = 1, \dots, m$ . Here,  $\alpha_i$  are the Lagrange multipliers. The Lagrangian  $L$  has to be minimized with respect to the primal variables  $w$  and  $b$  and maximized with respect to the dual variables  $\alpha_i$  (in other words, a saddle point has to be found).

# Hyperplane classifiers - KKT conditions

The Karush-Kuhn-Tucker (KKT) complementarity conditions of optimization theory state, that at the saddle point, the derivatives of  $L$  with respect to the primal variables must vanish, since

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0 \text{ and } \frac{\partial}{\partial w} L(w, b, \alpha) = 0 \quad (8)$$

leads to

$$\sum_{i=1}^m \alpha_i y_i = 0 \text{ and } w = \sum_{i=1}^m \alpha_i y_i x_i. \quad (9)$$

The solution vector  $w$  thus has an expansion in terms of a subset of the training patterns, namely those patterns with non-zero  $\alpha_i$ , called Support Vectors (SVs).

# Hyperplane classifiers - Dual optimization problem

We can again re-write our optimization problem by substituting (9) into the Lagrangian (7) to eliminate the primal variables. This yields the dual optimization problem, which is usually solved in practice

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & \alpha_i \geq 0 \quad \forall i = 1, \dots, m \text{ and } \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned} \tag{10}$$

# Hyperplane classifiers - Dual optimization problem

Finally, the decision function can be re-written using (9) as

$$f(x) = \operatorname{sgn} \left( \sum_{i=1}^m \alpha_i y_i \langle x, x_i \rangle + b \right), \quad (11)$$

where  $b$  can be computed by exploiting  $\alpha_i [y_i (\langle x_i, w \rangle + b) - 1] = 0$ , which follows from the KKT conditions.

Details on mathematical optimization and convex constrained problems can be found in [Jarre, 2019]. Explanations on dealing with nonlinear problems are given in [Reinhardt, 2012].

# Maximum margin separator

We now have all the theoretical background to go back to our initial classification problem. We can implement a SVM as a maximum margin separator for the given data set  $\mathcal{X}$ .

# Maximum margin separator

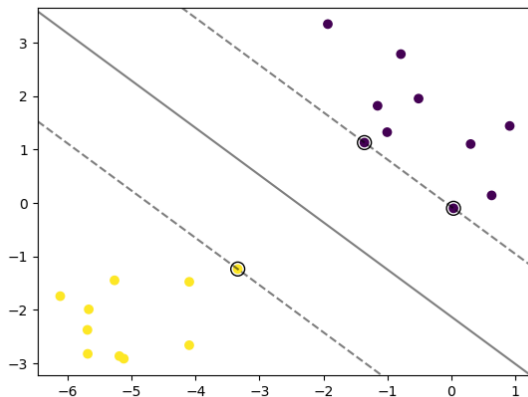


Figure: Implementation of a SVM using the 'linear' Kernel.

# Limitations

Let's consider the following data set.

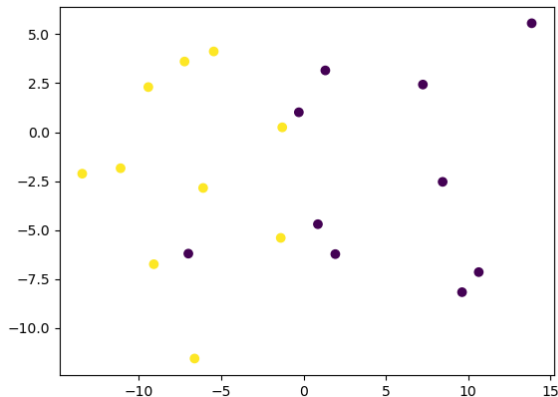


Figure: Linearly or separable or not?



# Soft Margin Hyperplanes

We introduce a slack variable

$$\xi_i \geq 0 \quad \forall i = 1, \dots, m \quad (12)$$

in the simplest case, this leads to

$$\min_{w \in \mathcal{H}, \xi \in \mathbb{R}^n} \tau(w, \xi) = \frac{1}{2} \|w\|^2 + \frac{C}{m} \sum_{i=1}^m \xi_i \quad (13)$$

subject to  $y_i (\langle w, x \rangle + b) \geq 1 - \xi_i \quad \forall i = 1, \dots, m.$

Our dual optimization problem also gets rewritten as

$$\max_{\alpha \in \mathbb{R}^m} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (14)$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{C}{m} \quad \forall i = 1, \dots, m \text{ and } \sum_{i=1}^m \alpha_i y_i = 0.$$

This classifier is referred to as C-SV classifier and can be used to prevent overfitting by allowing the classifier to make false classifications.

More classifiers using soft margins can be found in [Schölkopf, 2002](Chap. 7.5).

# Soft Margin Hyperplanes

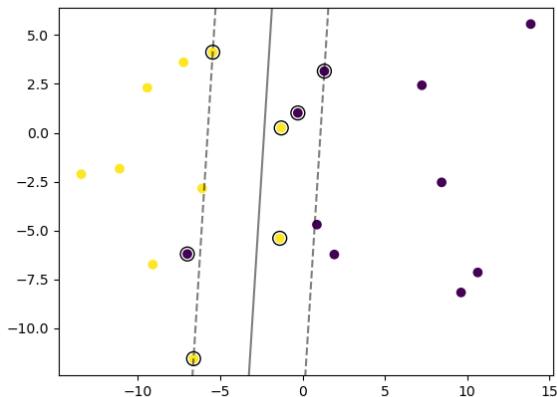


Figure: Implementation of a SVM using the 'linear' Kernel and a soft margin.

# Limitations

Let's consider the following data sets.

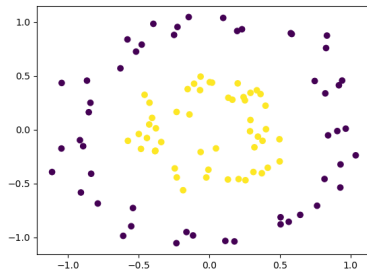
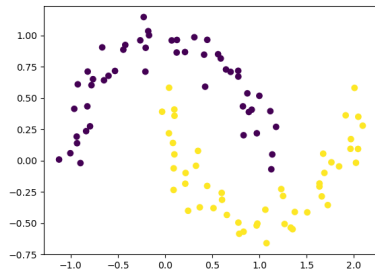


Figure: Two examples of data that can't be linearly separated.

# Limitations

What happens if you try to separate them linearly?

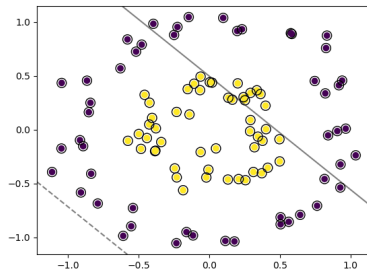
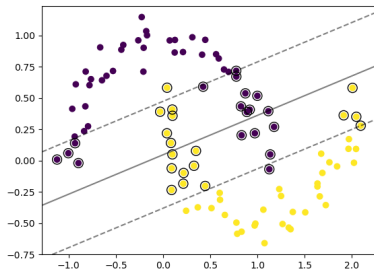


Figure: SVMs using the 'linear' Kernel.

# The kernel trick

To extend the introduced SVM algorithm, we can substitute (11) by applying a kernel of the form

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (15)$$

where

$$\begin{aligned} \Phi : \mathcal{X} &\rightarrow \mathcal{H} \\ (x) &\mapsto \Phi(x) \end{aligned} \quad (16)$$

is a function that maps an input from  $\mathcal{X}$  to a dot product space  $\mathcal{H}$ . This is referred to as the **kernel trick**.

# The kernel trick

We then obtain decision functions of the form

$$f(x) = \text{sgn} \left( \sum_{i=1}^m \alpha_i y_i \langle \Phi(x), \Phi(x_i) \rangle + b \right) \quad (17)$$

$$= \text{sgn} \left( \sum_{i=1}^m \alpha_i y_i k(x, x_i) + b \right) \quad (18)$$

and the optimization problem

$$\max_{\alpha \in \mathbb{R}^m} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(x, x_i) \quad (19)$$

$$\text{subject to } \alpha_i \geq 0 \quad \forall i = 1, \dots, m \text{ and } \sum_{i=1}^m \alpha_i y_i = 0.$$

# The kernel trick

The  $m \times m$  Matrix  $K$  with elements  $K_{ij} = k(x_i, x_j)$  is called the **Gram matrix** (or kernel matrix) of  $k$ .

A kernel  $k$  is called **positive definite kernel**, when the Gram matrix  $K$  is positive definite.

As stated in [Schölkopf, 2002](Chap. 2): *Given an algorithm which is formulated in terms of a positive definite kernel  $k$ , one can construct an alternative algorithm by replacing  $k$  by another positive definite kernel  $\tilde{k}$ .*



# The kernel trick

The kernel trick can be applied since all feature vectors only occurred in dot products. A more precise explanation can be found in [Schölkopf, 2002](Chap. 2).

# A suitable kernel

Going back to our problem of non linearly separable data, we can use a kernel function of the form

$$k(x, x') = \exp \left( -\frac{\|x - x'\|^2}{2\sigma^2} \right), \quad (20)$$

a so called **Gaussian radial basis function** (GRBF or RBF kernels) with  $\sigma > 0$ .

# Solving the nonlinear problem

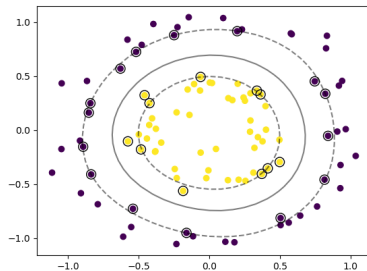
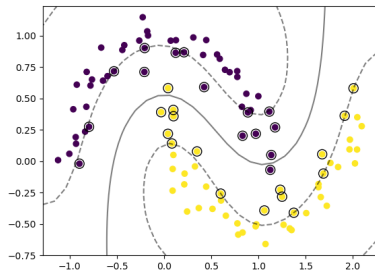


Figure: SVMs using the 'rbf' Kernel.

# Examples of kernels

An overview of common kernels:

- **Linear:**  $k(x, x') = \langle x, x' \rangle$
- **Polynomial:**  $k(x, x') = \langle x, x' \rangle^d, d \in \mathbb{N}$
- **Inhomogeneous Polynomial:**  $k(x, x') = (\langle x, x' \rangle + c)^d, d \in \mathbb{N}, c \geq 0$
- **Gaussian:**  $k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \sigma > 0$
- **Sigmoid:**  $k(x, x') = \tanh(\kappa \langle x, x' \rangle + \vartheta), \kappa > 0, \vartheta < 0$

These kernels are implemented in the Python modul `scikit-learn` `sklearn.svm` based on the `libsvm` implementation in C++ by Chih-Chung Chang and Chih-Jen Lin [Chang, 2011].

# More kernel applications

Some interessting kernel applications.

# References



Schölkopf, Bernhard, Alexander J. Smola

Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press, 2002.



Liesen, Jörg, Volker Mehrmann

Lineare Algebra. Wiesbaden, Germany: Springer, 2015.



Jarre, Florian, Josef Stoer

Optimierung: Einführung in mathematische Theorie und Methoden. Springer-Verlag, 2019.



Reinhardt, Rüdiger, Armin Hoffmann, Tobias Gerlach


Nichtlineare Optimierung: Theorie, Numerik und Experimente. Springer-Verlag, 2012.



Chang, Chih-Chung, Chih-Jen Lin

LIBSVM : A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

# Time for your questions!

Follow our development on GitHub 

<https://github.com/JoshuaSimon/Kernel-Learning>