

# Grundlagen von Support Vector Machines (SVM)

Jens Pönisch

2019-03-16



- Masterarbeit Sandy Bitterlich 2016
- Optimierungsverfahren im Machine Learning
- ADMM  $\rightarrow$  SVM  $\rightarrow$  MPI
- Eigene Implementierung des kompletten Prozesses.

$$\mathbf{x} = (x_1, \dots, x_n)^T$$

$\mathbb{R}^n$

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \dots + x_n y_n$$

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \dots + x_n^2}$$

$$\nabla f(\mathbf{x}) = \left( \frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)^T$$

$$\arg \min_{\mathbf{x}} f(\mathbf{x})$$

s. t.

Vektor mit  $n$  Komponenten.

Menge aller  $n$ -dimensionalen reellwertigen Vektoren

Skalarprodukt der Vektoren  $\mathbf{x}$  und  $\mathbf{y}$ .

Euklidische Norm (= Länge) des Vektors  $\mathbf{x}$ .

Partielle Ableitung der reellwertigen Funktion  $f(\mathbf{x})$  nach allen Komponenten.

Der Vektor  $\mathbf{x}$ , für den  $f(\mathbf{x})$  minimal wird.

Nebenbedingungen (*subject to*).

- Teilgebiet der *Künstlichen Intelligenz*.
- Ziel: «Erlernen» eines Modells aus vorhandenen Trainingsdaten.
- Verfahrensklassen des *Machine Learning*:
  - Supervised Learning: Für Trainingsdaten sind korrekte Ergebnisse bekannt, für neue Daten soll Ergebnis vorhergesagt werden.
  - Unsupervised Learning: Es gibt keine Ergebnisse, gesucht sind Strukturen innerhalb der Daten (z. B. Cluster).
  - Reinforcement Learning: Eine bestimmte Strategie wird bewertet, Suche nach optimaler Strategie (Geländeerkundungen, optimale Wege).
- *SVM* ist Verfahren des *Supervised Learning*.
- Gesuchtes Ergebnis für neue Daten ist entweder Zugehörigkeit zu bestimmter Klasse (Klassifikation) oder stetige Größe (Regression).
- Hier nur Klassifikation.

- Ein neues Objekt (Datenpunkt) soll einer Klasse zugeordnet werden  
(SPAM – HAM, Katze – Hund – Pferd, Kunde wird Produkt kaufen oder nicht, ...).
- Für jeden Datenpunkt gleiche Menge von Daten (Features, Komponenten) vor, die als reelle Zahlen codiert sind  
(Anzahl Worthäufigkeiten, Pixelwerte, Alter, Geschlecht, bisher gekaufte Produkte, ...).
- Eine bestimmte Zahl von Daten ist bereits klassifiziert (Label).
- Zunächst genau zwei Klassen mit den Labels -1 und +1.
- Aus den klassifizierten Daten (Trainingsdaten) wird das Modell erstellt.
- Mithilfe des Modells werden neue Daten klassifiziert.

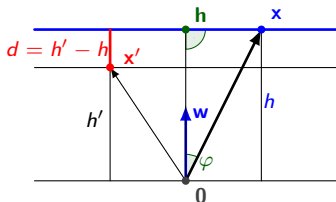
- Lineare SVM: Vapnik und Chervonenkis 1963.
- Kernel Trick: Boser, Guyon, Vapnik 1992.
- Soft Margin: Cortes, Vapnik 1993–1995.
- Weitere Entwicklungen bei Optimierungsverfahren und Mehrklassen-SVM.

- Betrachten Datenpunkte als Punkte im (euklidischen) Raum.
- Konstruieren eine Hyperebene, die beide Punktklassen voneinander «optimal» trennt → Modell.
- Für neuen Punkt wird bestimmt, auf welcher Seite der Hyperebene er liegt → Klasse.

Fragen:

- Bestimmen der Hyperebene?
- Was ist «optimal»?
- Bestimme die Seite der Ebene für einen Punkt.

# Abstand eines Punktes von Ebene



- $w$  Vektor senkrecht zur Ebene,  $x$  beliebiger Punkt auf Ebene,  $\varphi$  Winkel zwischen  $w$  und  $x$ .
- Abstand der Ebene vom Ursprung:  $h = \|x\| \cos \varphi$  gegeben.
- Skalarprodukt:  
 $\langle w, x \rangle = \|w\| \cdot \|x\| \cdot \cos \varphi$
- Also  $h = \frac{\langle w, x \rangle}{\|w\|}$

- Setzen  $b := -h \cdot \|w\|$ , also

$$\frac{\langle w, x \rangle + b}{\|w\|} = 0$$

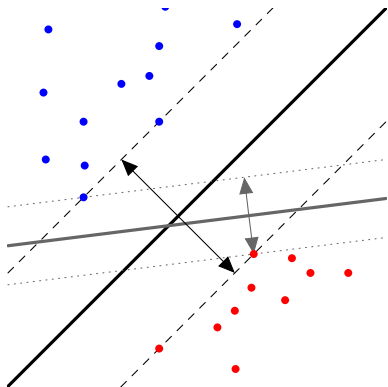
(Ebenengleichung).

- Punkt  $x'$  liegt auf einer Parallelebene mit Abstand  $h' = \frac{\langle w, x' \rangle}{\|w\|}$  vom Ursprung, also Abstand von der Ebene

$$\begin{aligned} d &= h' - h = h' + \frac{b}{\|w\|} \\ &= \frac{\langle w, x' \rangle + b}{\|w\|} \end{aligned}$$

- Multiplikation von  $w$  und  $b$  mit positiven Faktor verändert Abstand nicht.
- Multiplikation mit negativem Faktor vertauscht Seiten der Ebene.





- Lege die Hyperebene so, dass der kleinste Abstand eines Punktes zur Ebene möglichst groß ist (und auf der «richtigen» Seite/Klasse  $t$  liegt):

$$t \left( \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} \right) \rightarrow \max, \quad t \in \{-1, +1\}$$

- Möglichst breiter «Streifen» unterhalb und oberhalb der Ebene, der frei von Trainingspunkten ist.
- Punkte auf dem «Rand» heißen *Supportvektoren*.
- Neue Punkte werden damit hoffentlich «richtig» klassifiziert.

$$\begin{aligned} & \arg \max_{\mathbf{w}, b} \left\{ \min_{m=1, \dots, M} t_m \left( \frac{\langle \mathbf{w}, \mathbf{x}_m \rangle + b}{\|\mathbf{w}\|} \right) \right\} \\ & = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{m=1, \dots, M} t_m (\langle \mathbf{w}, \mathbf{x}_m \rangle + b) \right\} \end{aligned}$$

- Wähle die Ebene  $\mathbf{w}$ ,  $b$  so, dass der kleinste Abstand eines Punktes zur ihr maximiert wird.
- Doppeloptimierung schwierig, deshalb Umformung.

# Vereinfachung des Optimierungsproblems

- Sei  $\kappa$  das Minimum, damit hat jeder Punkt mindestens den Abstand  $\pm\kappa$  von der Ebene, also

$$t_m(\langle \mathbf{w}, \mathbf{x}_m \rangle + b) \geq \kappa > 0, m = 1, \dots, M$$

- Teilen  $\mathbf{w}$  und  $b$  durch  $\kappa$  (Ebenenabstand ändert sich nicht):

$$t_m(\langle \mathbf{w}', \mathbf{x}_m \rangle + b') \geq 1, m = 1, \dots, M$$

- Wir erhalten damit das neue Problem mit Nebenbedingungen:

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\}$$

$$\text{s. t. } t_m(\langle \mathbf{w}, \mathbf{x}_m \rangle + b) \geq 1, m = 1, \dots, M$$

- Wenn  $\frac{1}{\|\mathbf{w}\|}$  maximal, dann  $\|\mathbf{w}\|$  minimal, ebenso  $\|\mathbf{w}\|^2$ .
- Fügen noch einen Faktor  $\frac{1}{2}$  ein.

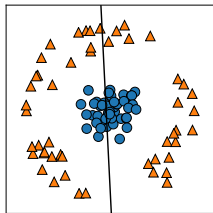
$$\begin{aligned} & \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s. t. } & t_m (\langle \mathbf{w}, \mathbf{x}_m \rangle + b) \geq 1, \quad m = 1, \dots, M \end{aligned}$$

- Lösung mit iterativen Verfahren, z. B. mit *Stochastic Gradient Descent* (SGD) in `scikit-learn`.
- Komplexität:

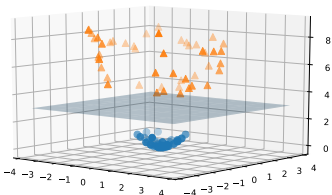
$$O(k \cdot M \cdot \tilde{n})$$

mit Iterationszahl  $k$  und Komponentenzahl  $\tilde{n}$  jedes Datenpunktes ungleich 0.

# Problem 1: Keine lineare Trennung möglich



- Lineare Trennung nicht möglich.



- Mit Radius als neue Komponente.

- Nichtlineare Transformation:  
Verknüpfen von Komponenten,  
hier

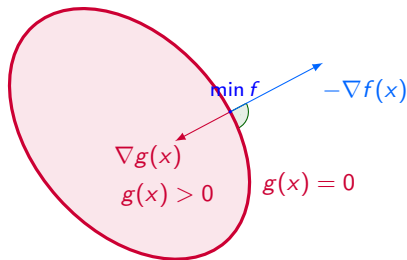
$$\Phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2)^T$$

- Allgemein: Verknüpfung von  
Komponentenpaaren  
 $a_{ii}x_i^2 + a_{ij}x_i x_j + a_{jj}x_j^2 + b_i x_i + b_j x_j$ .

Probleme:

- Zahl der Komponenten steigt  
extrem: vielleicht auch Tripel,  
Quadrupel, höhere Potenzen  
notwendig.
- Große Zahl frei wählbarer  
Parameter: welche Werte wählen?

# Duales Problem



- Verallgemeinertes Problem:  
 $f(\mathbf{x}) \rightarrow \min, \text{ s. t. } g(\mathbf{x}) \geq 0.$
- Lagrange-Formulierung:

$$\mathcal{L}_a(\mathbf{x}) = f(\mathbf{x}) - a \cdot g(\mathbf{x})$$

s. t.  $g(\mathbf{x}) \geq 0, a \geq 0$

- Liegt das Optimum auf dem Rand  $g(\mathbf{x}) = 0$ , liegen  $\nabla f$  und  $\nabla g$  auf einer Geraden, also:

$$\nabla \mathcal{L}_a(\mathbf{x}) = \nabla f(\mathbf{x}) - a \cdot \nabla g(\mathbf{x}) = \mathbf{0}$$

- Wenn Optimum nicht auf Rand, dann  $\nabla f(\mathbf{x}) = 0$ , damit  $a = 0$ .
- Umformen von  $\nabla \mathcal{L}_a(\mathbf{x}) = 0$  nach  $\mathbf{x}$ .
- Eliminieren in  $\mathcal{L}$  Parameter  $\mathbf{x} \rightarrow$  neues Optimierungsproblem in  $a$ .
- Mehrere Nebenbedingungen  $g_m(\mathbf{x}) \geq 0$ :

$$\mathcal{L}_a(\mathbf{x}) = f(\mathbf{x}) - \sum_{m=1}^M a_m \cdot g_m(\mathbf{x})$$

- Was bringt das?

- Nebenbedingung umformulieren ( $g(\mathbf{x}) \geq 0$ ):

$$t_m (\langle \mathbf{w}, \mathbf{x}_m \rangle + b) - 1 \geq 0$$

- Lagrangefunktion in  $\mathbf{w}$  und  $b$ , Lagrange-Parameter  $\mathbf{a}$

$$\mathcal{L}_{\mathbf{a}}(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{m=1}^M a_m (t_m (\langle \mathbf{w}, \mathbf{x}_m \rangle + b) - 1)$$

- Setze alle Ableitungen nach  $w_i$  und  $b$  auf 0, ersetze alle  $w_i$  und  $b$ .
- Optimierungsproblem in  $\mathbf{a}$ .

$$\arg \min_{\mathbf{a}} \left\{ \frac{1}{2} \sum_{i=1}^M \sum_{k=1}^M a_i a_k t_i t_k \langle \mathbf{x}_i, \mathbf{x}_k \rangle - \sum_{i=1}^M a_i \right\}$$

s. t.  $a_i \geq 0, \quad \sum_{i=1}^M a_i t_i = 0, \quad i = 1, \dots, M$

- $a_m \neq 0$  nur für Supportvektoren  $\mathbf{x}_m$  (KKT-Bedingung).
- Klassifikation eines neuen Punktes:

$$y(\mathbf{x}) = \sum_{s \in S} a_s t_s \langle \mathbf{x}, \mathbf{x}_s \rangle + b \quad S \text{ Indices der Supportvektoren}$$

$$\text{mit } b = \frac{1}{|S|} \sum_{s \in S} \left( t_s - \sum_{u \in S} a_u t_u \langle \mathbf{x}_s, \mathbf{x}_u \rangle \right)$$

- Nichtsupportvektoren werden im Modell nicht benötigt.



- Datenpunkte  $\mathbf{x}$ ,  $\mathbf{x}_m$  erscheinen nur in Skalarprodukten.
- Betrachten nichtlineare Transformation:  $\mathbf{x} \rightarrow \Phi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^\infty$
- Ersetzen

$$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k) \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^\infty \times \mathbb{R}^\infty \rightarrow \mathbb{R}$$

durch Kernelfunktion

$$k(\mathbf{x}_i, \mathbf{x}_k) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$$

- Statt  $\Phi(\cdot)$  wird *Kernelfunktion*  $k(\cdot, \cdot)$  benötigt.
- Jede Funktion geeignet, deren *Gram-Matrix*  $[k(\mathbf{x}_i, \mathbf{x}_k)]_{i,k=1}^M$  positiv semidefinit ist.

- «Gauß-Kernel» (Radial Base Function):

$$k(\mathbf{x}_i, \mathbf{x}_k) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_k\|^2}$$

*Stationärer* Kernel (nur von Punktabstand abhängig), freier Parameter  $\gamma$ .

- Polynomkernel:

$$k(\mathbf{x}_i, \mathbf{x}_k) = (\langle \mathbf{x}_i, \mathbf{x}_k \rangle + r)^d$$

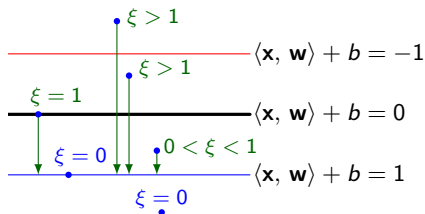
Freie Parameter  $r, d$ .

- Sigmoid-Kernel:

$$k(\mathbf{x}_i, \mathbf{x}_k) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_k \rangle - \delta)$$

Freie Parameter  $\kappa, \delta$ .

## Problem 2: Behandlung von Ausreißern



- Durch «Ausreißer» liegen einzelne Datenpunkte in der «falschen» Klasse, keine Trennung möglich.
- Erlauben einzelne Ausreißer durch *Slackvariable*, addieren Korrekturwert auf die eigentlich verletzte Randbedingung, um diese zu erfüllen.
- Die Summe der Korrekturen soll möglichst klein sein.

$$\arg \min_{\mathbf{w}, b, \xi_1, \dots, \xi_M} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{m=1}^M \xi_m \right\}$$

$$\text{s. t. } t_m (\langle \mathbf{w}, \mathbf{x}_m \rangle + b) + \xi_m \geq 1, \xi_m \geq 0, m = 1, \dots, M$$

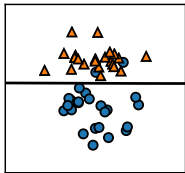
- $C$  ist *Regularisierungsparameter*.
- Je größer  $C$ , desto komplexer das Modell (höhere «Bestrafung» der Ausreißer).
- Statt  $C \sum \xi_m$  (*Lasso-Regularisierung*)  
auch  $C \sum \xi_m^2$  (*Ridge-Regularisierung*).
- Duales Problem: Lagrange-Funktion mit Variablen  $\mathbf{w}, b, \xi_1, \dots, \xi_M$  und neuen Parametern für s. t.  $a_1, \dots, a_M, \alpha_1, \dots, \alpha_M$ .

$$\arg \min_{\mathbf{a}} \left\{ \frac{1}{2} \sum_{i=1}^M \sum_{k=1}^M a_i a_k t_i t_k \langle x_i, x_k \rangle - \sum_{i=1}^M a_i \right\}$$

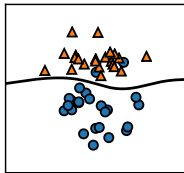
s. t.  $0 \leq a_i \leq C, \quad \sum_{i=1}^M a_i t_i = 0, \quad i = 1, \dots, M$

- Lagrange-Faktoren  $\alpha_m$  kürzen sich weg.
- Quadratisches Problem mit *box constraints*.
- Lösungsverfahren *Sequential Minimal Optimization* (SMO) in `libsvm`, `scikit-learn`.
- Komplexität:  $O(M^2 \cdot n)$  bis  $O(M^3 \cdot n)$  (datenabhängig).
- Komplexität begrenzt Anzahl der Trainingsdaten.

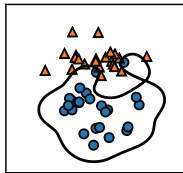
$\gamma = 0.005, C = 0.5$



$\gamma = 0.1, C = 100$

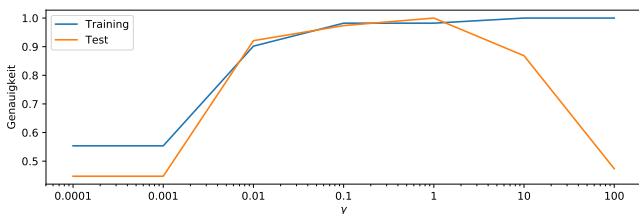


$\gamma = 1.0, C = 1000$

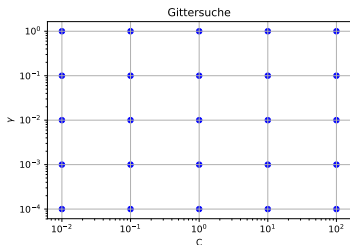


- SVM-Modell ist eigentlich deterministisch und kann im Prinzip exakt berechnet werden.
- Aber: Modell ist von mehreren Parametern abhängig, diese müssen optimiert werden.
- Underfitting: Modell ist zu einfach und klassifiziert zu schlecht.
- Overfitting: Modell «lernt» die Trainingsdaten auswendig und abstrahiert nicht.

# Bewerten des Modells



- Daten teilen in Trainings- und Testmenge (z. B. 80 % / 20 %).
- Aufteilung muss statisch sein, Testdaten dürfen nie ins Training eingehen.
- Modellerstellung mit Trainingsmenge, Evaluierung mit Testmenge.
- Underfitting: Geringe Trefferzahl auf Trainingsmenge.
- Overfitting: extrem gute Trefferzahl auf Trainings-, schlechte Trefferzahl auf Testmenge.
- Optimales Modell: etwa gleich gute Trefferzahl auf Trainings- und Testmenge.

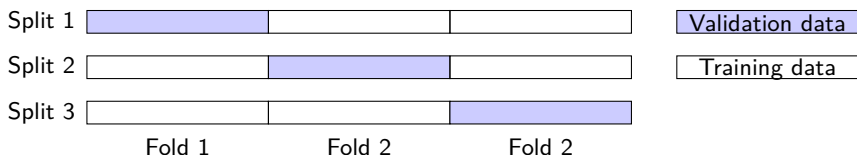


- Gezielte Suche nach bestem Parametertupel.
- Systematisches Durchwandern der möglichen Parameterwerte in geometrischer Folge (Faktor 2 oder 10).

```
MinFehler  $\leftarrow \infty$ 
for  $C \in (10^{-1}, 1, 10, 100)$  do
  for  $\gamma \in (10^{-5}, 10^{-4}, \dots, 0.1, 1)$  do
    Bestimme Supportvektoren mit
    Parametertupel
    Berechne Fehler
    if Fehler < MinFehler then
      MinFehler  $\leftarrow$  Fehler
       $C_{best} \leftarrow C$ 
       $\gamma_{best} \leftarrow \gamma$ 
    end if
  end for
end for
```



# Kreuzvalidierung



Problem: Gittersuche für komplexe Modelle führt zu Overfitting.

Lösung: Kreuzvalidierung.

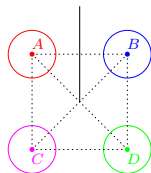
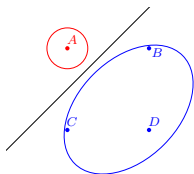
- Aufteilen der Trainingsmenge in  $k$  Gruppen (Folds).
- Jede Gruppe muss Daten beider Klassen enthalten.
- Berechnung von  $k$  Modellen je Parameterpaar:  $k - 1$  Gruppen für Modellberechnung, eine Gruppe für Validierung.
- Bewertung der Genauigkeit aller Modelle, Zusammenfassung (Mittelwert).

Anschließend: Modellberechnung für *alle* Trainingsdaten mit besten Parametern.

- Verfahren behandelt alle Dimensionen gleichgewichtet.
- Werte aller Dimensionen sollten im gleichen Intervall liegen (z. B.  $[0, 1]$  oder Mittelwert 0 und Varianz 1)  $\rightarrow$  Umskalieren.
- Behandlung von Nominaldaten (mehrere Kategorien):
  - One-Hot-Codierung: Eine Variable pro Kategorie, genau eine dieser Variablen hat Wert 1, alle anderen 0.
  - Textdaten (jedes mögliche Wort eine Dimension): *Word embeddings*, z. B. *Word2Vec*.
- Behandlung fehlender Komponenten in Datenpunkten: Datenpunkt löschen oder Komponente durch Mittelwert bzw. Median ersetzen.

# Mehrfachklassifikation

- Statt 2 nun  $K$  Klassen.
- SVM kann nur zwei Klassen trennen  $\rightarrow$  mehrere SVMs.



## One-versus-Rest:

Berechne  $K$  SVMs für jede Trennung  $k - \text{nicht } k$  ( $M$  Daten je SVM).

Neuen Punkt mit jeder SVM bewerten und Klasse mit größtem Abstand wählen.

## One-versus-One:

Berechne  $\binom{K}{2}$  SVMs für jedes Klassenpaar mit Trainingsdaten dieses Paares ( $\approx 2 \frac{M}{K}$  Daten pro SVM). Neuer Punkt wird in die Klasse eingeordnet, die die meisten SVMs wählen.

In `scikit-learn`: lineare SVM mit OvR, Kernel-SVM mit OvO (Laufzeit).

- `libsvm`: Referenz- und Basisimplementierung, Kommandozeilen-Tool.  
Viele andere Bibliotheken sind einfach Schnittstellen zu `libsvm`.
- Python: `scikit-learn`: Sammlung von ML-Werkzeugen auf Basis von `numpy`, `SciPy`.
- R: `e1071` (`libsvm`-Schnittstelle), `kernlab` (`libsvm` und `bsvm`), `klaR` (`svmlight`).
- ...

- Beliebtes Werkzeug des ML (Kaggle Survey 2017: 26 %).
- Für Datensätze mittlerer Größe bei Kernel-SVM ( $O(m^2) \dots O(m^3)$ ).
- Recht gute Generalisierung.
- Auch für große Datenmengen hochdimensionaler Daten (Text Mining) mit linearer SVM.
- Auch weniger Trainingsdaten als Dimensionen möglich (Overfitting?) mit linearer SVM.
- Hoher Aufwand für mehr als 2 Klassen.
- Online-Learning (Nachtrainieren mit neuen Daten) im Prinzip möglich (nicht in `scikit-learn`).
- Berechnung auf GPUs möglich (ThunderSVM)
- Modelle schwer interpretierbar.

- [1] Arens et al.: Mathematik. Springer Spektrum. 2015.
- [2] Bishop: Pattern Recognition and Machine Learning. Springer. 2006.
- [3] Bitterlich: Numerische Verfahren zur Lösung von Support Vector Machines. Masterarbeit, TU Chemnitz. 2016.
- [4] Bordes et al.: Fast Kernel Classifiers with Online and Active Learning. Journal of Machine Learning Research. 2005.  
<http://www.jmlr.org/papers/volume6/bordes05a/bordes05a.pdf>
- [5] Cortes, Vapnik: Support-vector networks. In: Machine Learning, 20. 1995.
- [6] Hsu, Chang, Lin: A Practical Guide to Support Vector Classification. National Taiwan University. 2016. <https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

- [7] Kaggle 2017 Survey Results.  
<https://www.kaggle.com/amberthomas/kaggle-2017-survey-results>
- [8] Mikolov et al.: Efficient Estimation of Word Representations in Vector Space. arXiv 1301.3781. 2013.  
<https://arxiv.org/pdf/1301.3781.pdf>
- [9] Müller, Guido: Einführung in Machine Learning mit Python. O'Reilly. 2017.
- [10] Řehůřek: Scalability of Semantic Analysis in Natural Language Processing. Ph. D. Thesis. Brno. 2011.  
[https://radimrehurek.com/phd\\_rehurek.pdf](https://radimrehurek.com/phd_rehurek.pdf)
- [11] Wen et al.: ThunderSVM: A Fast SVM Library on GPUs and CPUs. Journal of Machine Learning Research 19. 2018. <http://www.jmlr.org/papers/volume19/17-740/17-740.pdf>