Support Vector Machine

Präsentiert von
Robert Gruber und Stephanos Potamianakis
am 02.02.2021
im Modul „Vertiefung der Grundlagen der Computerlinguistik“

Gliederung

1. Einleitung in SVM
2. Lossfunktion
3. Decision Boundary
4. Hyper-Parameter C
5. Intuition einer Kernel SVM
6. Verwendung einer SVM

Quellen

Einleitung in SVM

- Classification & Supervised

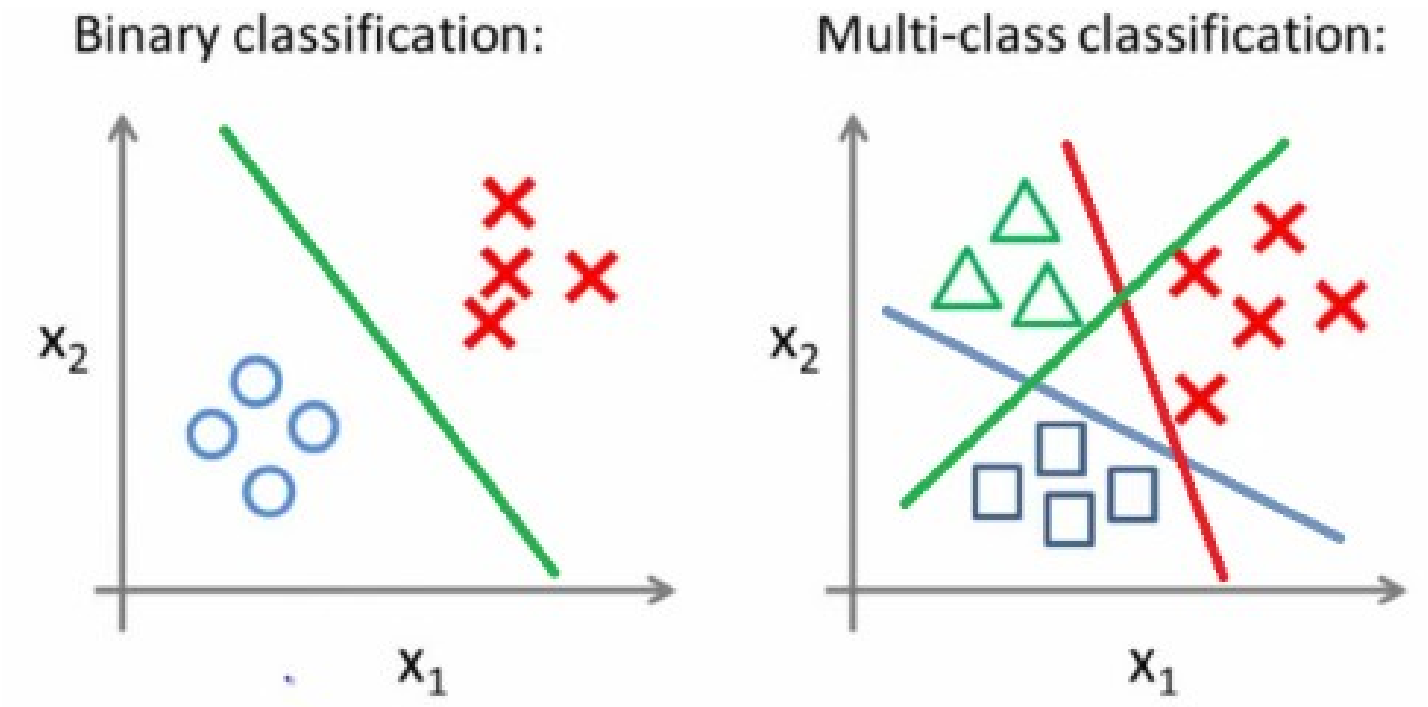
- Ziel: ideale Grenze + großmöglichster Abstand der Klassen

- SVMs basieren (genau wie z.B. Pezeptron und MaxEnt) auf der Berechnung linearer Vorhersage-Scores (Skalar-Produkt), welche eine Decision-Boundary beschreiben

- Erfunden im Jahre 1963 (Chervonenkis), Cortes und Vapnik entwickelten ihn weiter (1995)

- Viele Erfolge für NLP-Anwendungen (ca. 1995-2010). Immer noch wichtiges und einfach zu verwendendes Verfahren

- Binärer Klassifikator (für Multiklassifikation: Kombination mehrerer binärer SVMs)
- Multiklassifikation: one vs rest (eine Klasse vs. alle anderen), rot vs. [blau, grün], blau vs. [rot, grün], grün vs [rot, blau]



Einleitung in SVM

-Typische Anwendung: Mustererkennung

-Stärke der SVM: Gute Generalisierungsfähigkeit (richtige Kategorisierung neuer Daten)

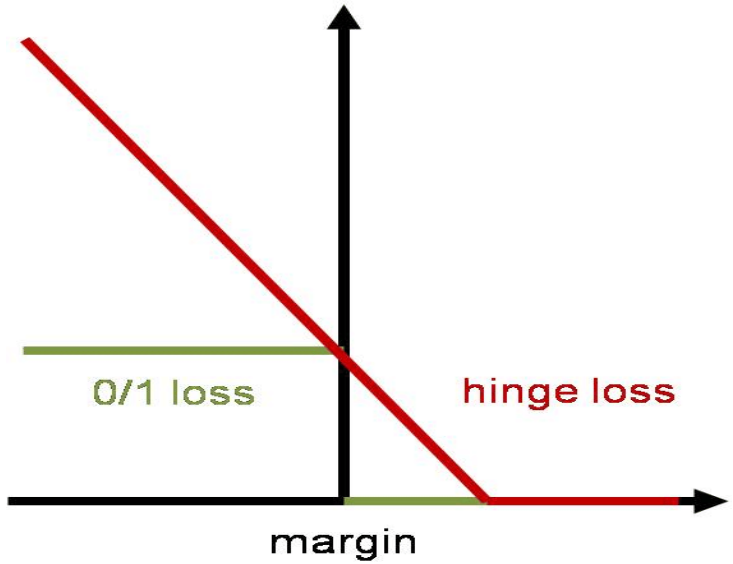
-Die zwei Klassen sollen mit möglichst viel “Zwischenraum” (Margin) voneinander getrennt werden

- Merkmale können mit bestimmten Funktionen (Kernels) transformiert werden, so dass die Vorhersage-Scores auch nicht-lineare Decision-Boundaries beschreiben kann

Lossfunktion

- Lossfunktion: Hinge loss (Knick in Funktion)
- Funktion: Wie gut ist der Algorithmus das erwartete Ergebnis vorherzusagen?
- Falsche Vorhersage: hohe Zahl, richtige Vorhersage: niedrige Zahl. Optimal nahe 0
- X-Achse: Score für Trainingsinstanz mit positivem Label
- Y-Achse: Wie schlecht

findet der Algorithmus den Score



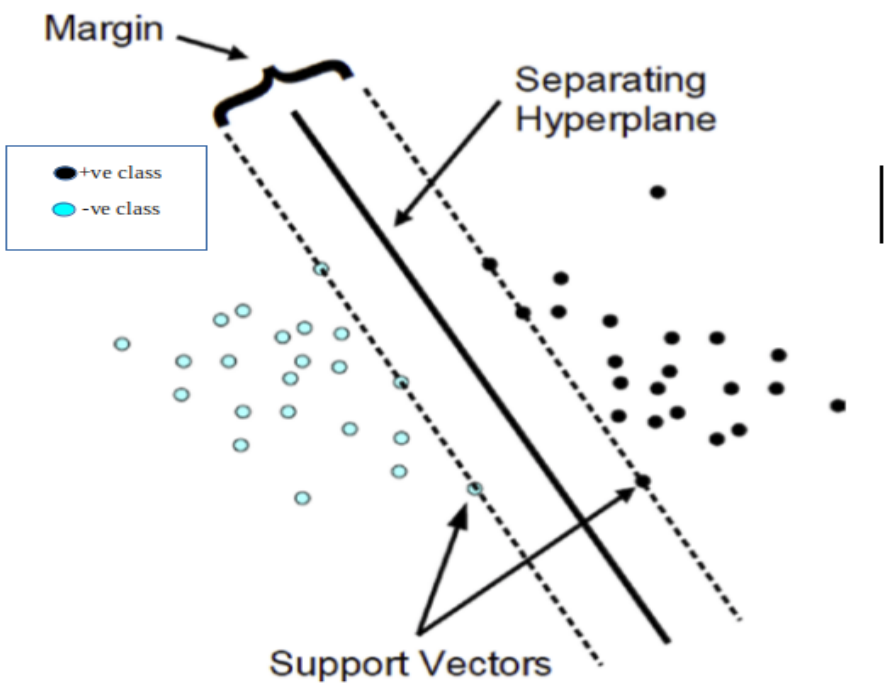
Decision Boundary

- Decision Boundary auswählen: Abstand der Instanzen einer Klasse und Instanzen der anderen Klasse maximieren

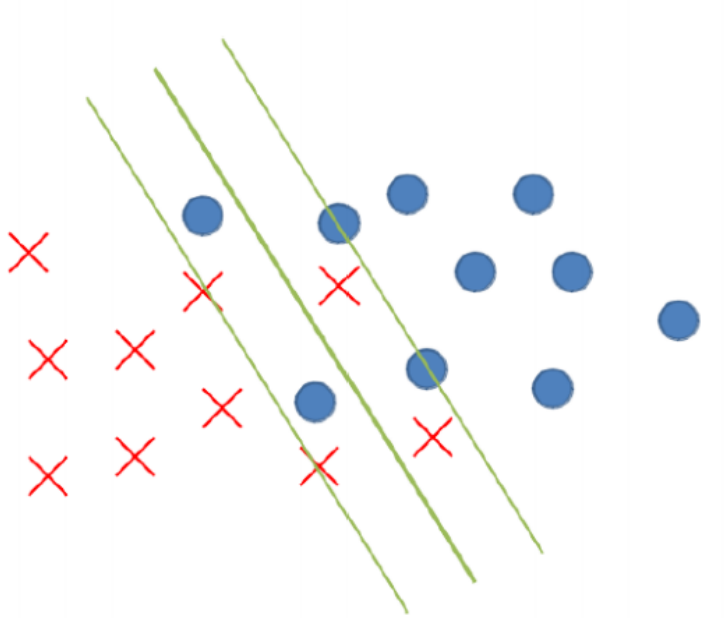
- Nur Supportvektoren wichtig

- Decision hyperplane: $\langle w, x \rangle + b = 0$

- Klassifikation: $y = (\text{sgn} \langle w, x \rangle + b)$



Nicht trennbarer Fall

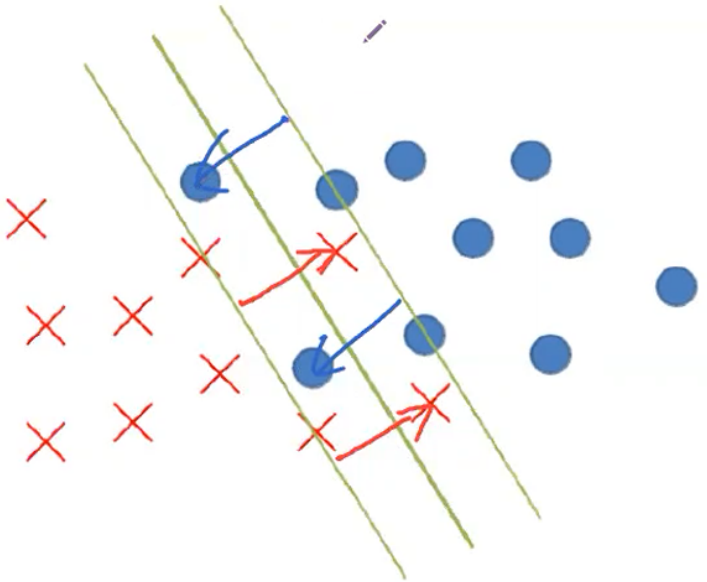


- Es werden 2 Ziele bei nicht linear trennbaren Datensätzen gegeneinander abgewogen:

1. Größe der Margin
2. Anzahl bzw. Entfernung der Elemente, die jenseits der Margin liegen

=> Hyper-Parameter C

Nicht trennbarer Fall



- Es werden 2 Ziele bei nicht linear trennbaren Datensätzen gegeneinander abgewogen:

1. Größe der Margin
2. Anzahl bzw. Entfernung der Elemente, die jenseits der Margin liegen

=> Hyper-Parameter C

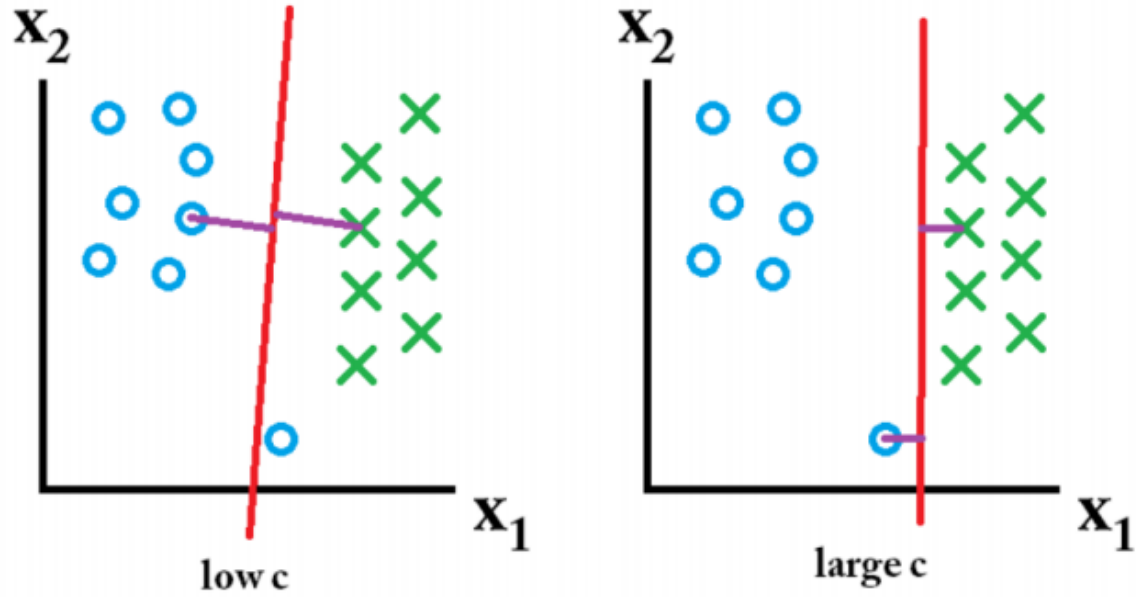
Hyper-Parameter C

$$\min_{w,b,\{\xi_n\}} \frac{1}{2}||w||_2 + C\sum_n \xi_n$$

- C beeinflusst ob die Decision Boundary weniger komplex oder sehr komplex sein soll

- Zu geringes C: Underfitting
- Zu großes C: Overfitting

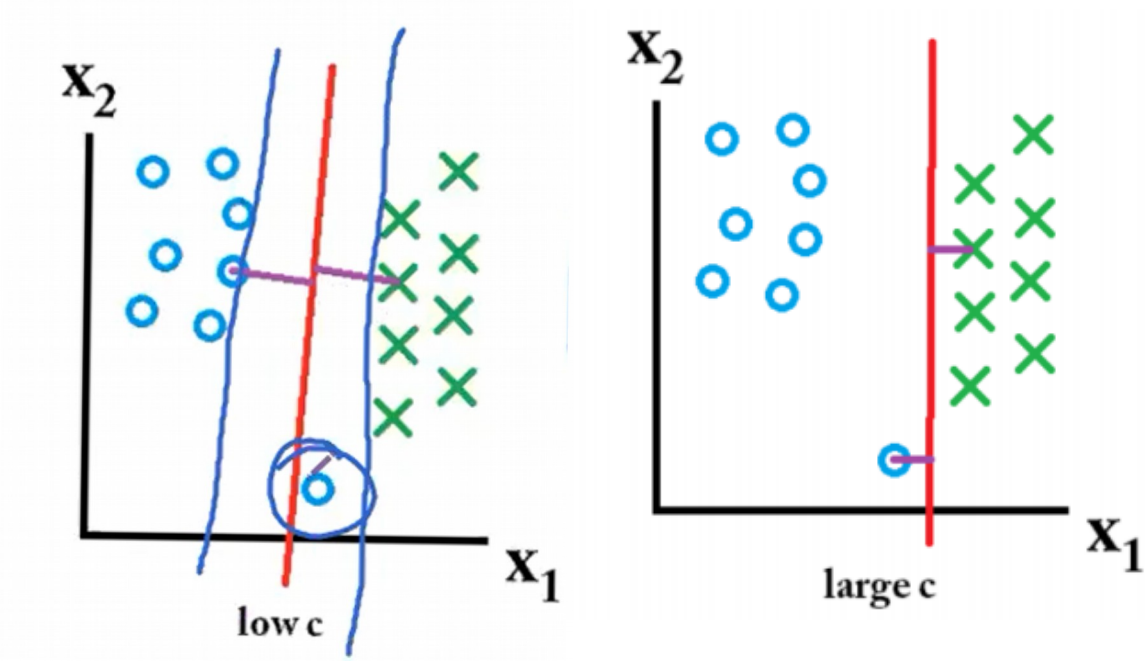
Hyper-Parameter C



- Hyper-Parameter C (z.B 0.1, 1, 10. …)

  - Kleines C: Margin ist groß, dafür aber Trainings-Instanzen jenseits der Margin
  - Großes C: wenige Trainings-Instanzen jenseits der Margin, dafür aber geringere Margin

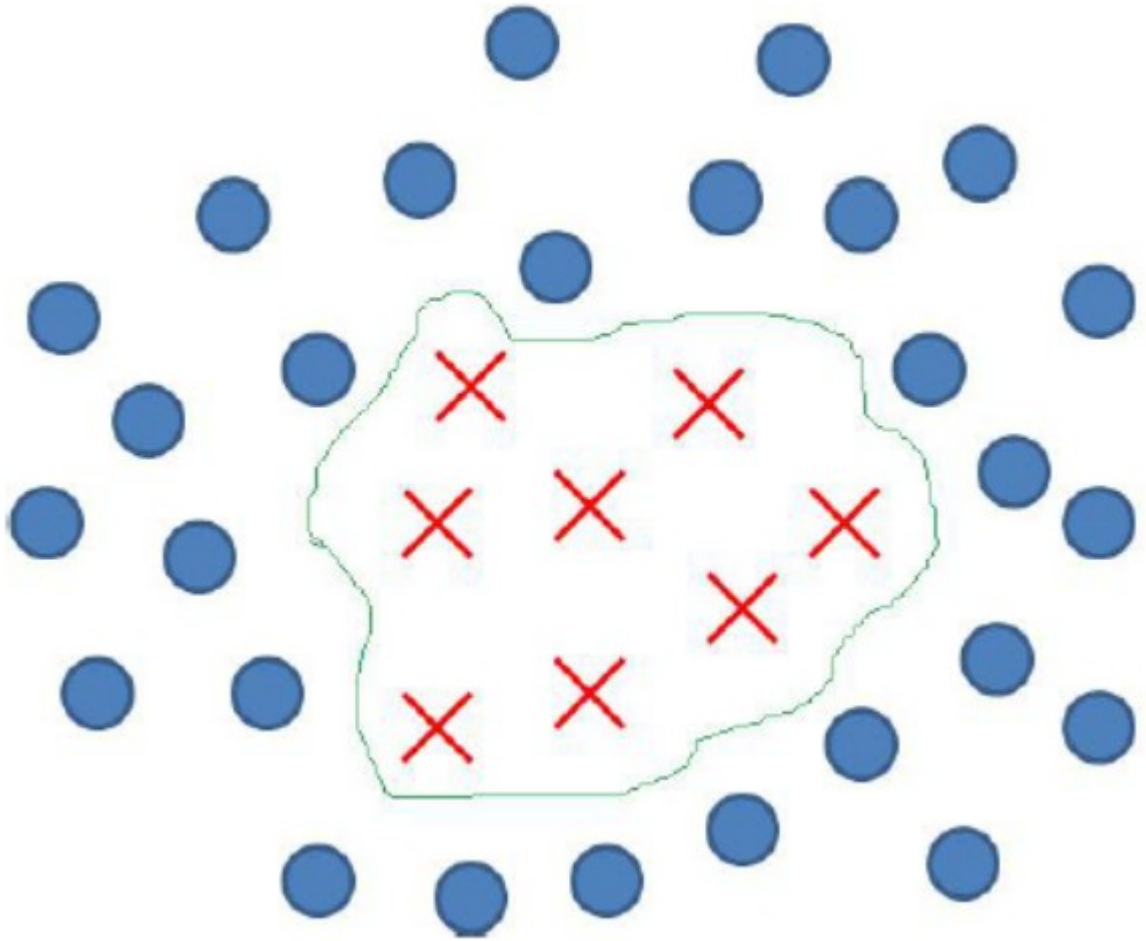
Hyper-Parameter C



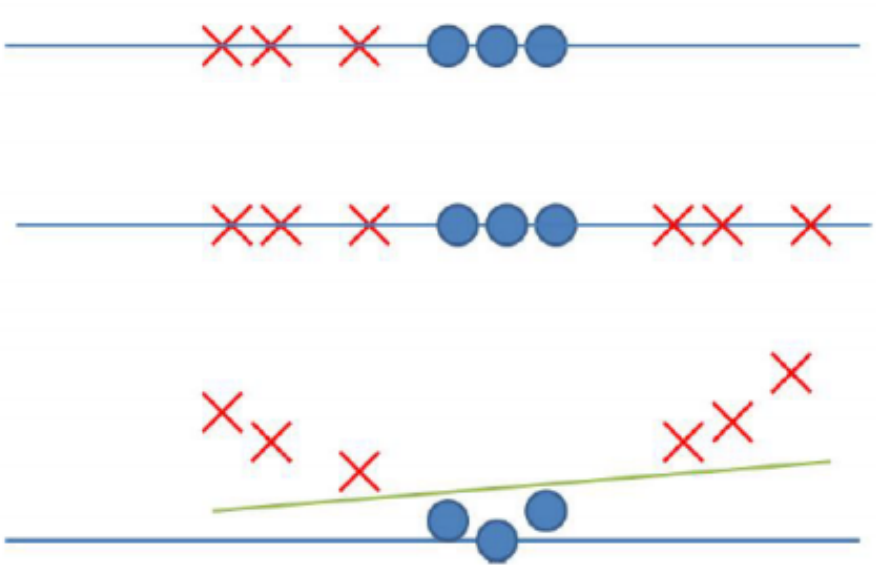
- Hyper-Parameter C (z.B 0.1, 1, 10. …)

  - Kleines C: Margin ist groß, dafür aber Trainings-Instanzen jenseits der Margin
  - Großes C: wenige Trainings-Instanzen jenseits der Margin, dafür aber geringere Margin

Linear trennbar oder nicht ?

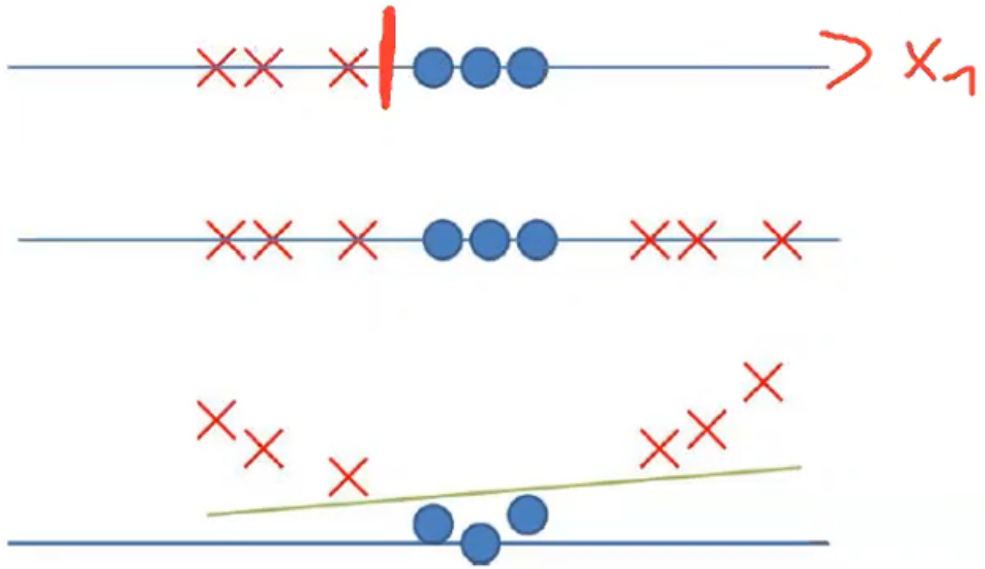


Intuition einer Kernel SVM



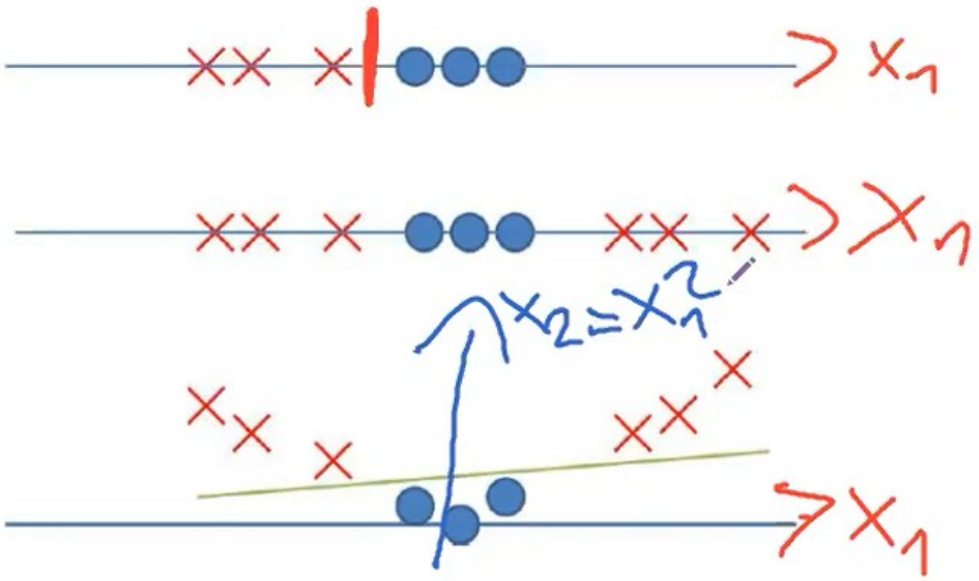
- Funktionen wie z.B die Quadratfunktion, sog. Kernels, können die Eingabe-Merkmale umformen und zusätzliche Merkmale liefern. $\boldsymbol{x} \rightarrow \psi(\boldsymbol{x})$
- Mit neuen Merkmalen bzw. zusätzlichen Dimensionen wird das Problem oftmals besser trennbar
- Diese Methode wird *Kernel-Trick* genannt:
  linear nicht trennbar → linear trennbar

Intuition einer Kernel SVM



- Funktionen wie z.B die Quadratfunktion, sog. Kernels, können die Eingabe-Merkmale umformen und zusätzliche Merkmale liefern. $\boldsymbol{x} \rightarrow \psi(\boldsymbol{x})$
- Mit neuen Merkmalen bzw. zusätzlichen Dimensionen wird das Problem oftmals besser trennbar
- Diese Methode wird *Kernel-Trick* genannt:
  linear nicht trennbar → linear trennbar

Intuition einer Kernel SVM



- Funktionen wie z.B die Quadratfunktion, sog. Kernels, können die Eingabe-Merkmale umformen und zusätzliche Merkmale liefern. $\boldsymbol{x} \rightarrow \psi(\boldsymbol{x})$
- Mit neuen Merkmalen bzw. zusätzlichen Dimensionen wird das Problem oftmals besser trennbar
- Diese Methode wird *Kernel-Trick* genannt:
  linear nicht trennbar → linear trennbar

Verwendung der SVM

Die SVM kann in Scikit-learn mit 3 verschiedenen Methoden implementiert werden:

- **SVC** und **NuSVC**: Support Vector Machines, welche die Möglichkeiten haben verschiedene Kernel-Funktionen zu verwenden
  - Langsam! (Anzahl der Trainingsinstanzen quadratisch)

- **LinearSVC**: Support Vector Machine ohne eine spezielle Kernel-Funktion (**linear kernel**)
  - Der einfache lineare Kernel funktioniert oft gut bei NLP-Anwendungen mit hochdimensionalen Merkmalsräumen (z.B. Wortmerkmale, großes Vokabular)
  - Schnell(er) (Anzahl der Trainingsinstanzen linear)

Anwendungsbeispiele

- **Gesichtserkennung**: grundsätzliche Erkennung von Bildern, welche vorverarbeitet wurden
- **Handschriftenerkennung**: Vorallem durch Verwendung der 10 Ziffern
- **Spamfilterkennung**: Unerwünschte Wörter wie z.B. „Money“ oder andere typische Zeichen filtern
- **Bionformatik**: Erkennung von bestimmten Gensequenzen, z.B. Identifikation eines Proteins

Quellen

- https://towardsdatascience.com/demystifying-maths-of-svm-13ccfe00091e
- https://medium.com/analytics-vidhya/understanding-loss-functions-hinge-loss-a0ff112b40a1
- https://stats.stackexchange.com/questions/23391/how-does-a-support-vector-machine-svm-work?fbclid=IwAR1mFun9iUtLIceOSGOvX0nKHn8wDDjD9zk9mcupjcCFaUw7q-hLHK71aQ
- https://www.knowledgehut.com/blog/data-science/support-vector-machines-in-machine-learning?fbclid=IwAR3zkvvjnECEYNTPHn9lOuYBCuwsh-BCyZL0Lu4wDwc5y6lU-x6_waWkoCg
- https://learnopencv.com/support-vector-machines-svm/?fbclid=IwAR20DvxoN4wz512OlLmRouAfmYrEeJcI8_jZy1OCyLTVr68yB7PL3l9eCy4
- Benjamin Roth – „Maximum Entropy Klassifikator; Support Vector Machine; Klassifikation mit Scikit-Learn“ in Computerlinguistische Anwendungen (2020)
- http://campar.in.tum.de/twiki/pub/Far/MachineLearningWiSe2003/kunze_ausarbeitung.pdf