Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU

Support Vector Machines (SVM)

Philipp Wicke, PhD
Centrum für Sprach- und Informationsverarbeitung
Ludwig-Maximilians-Universität München
pwicke@cis.lmu.de

Support Vector Machines

- Die SVM ist ein Binärer Klassifikator. (Für Multi-Klassen-Klassifikation können mehrere binäre SVMs kombiniert werden.)

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

- Erfunden wurde der SVM-Algorithmus 1963 (Chervonenkis), Cortes und Vapnik entwickelten ihn weiter (1995).

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

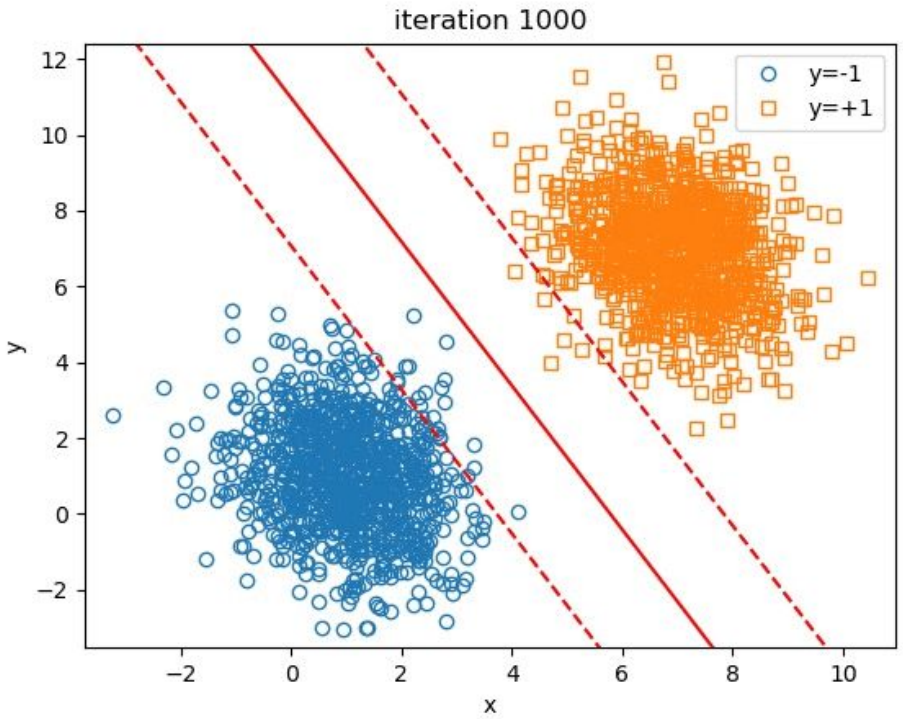
Support Vector Machines

- Es gibt 2 Unterschiede, durch die die SVM *manchmal* besser funktioniert als z.B. Perzeptron oder Logistic Regression:
    - Die SVM lernt eine Decision-Boundary, die die zwei Klassen in den Trainingsdaten (positiv, negativ) mit möglichst viel “Zwischenraum” (Margin) voneinander trennt.
    - Merkmale können mit bestimmten Funktionen (Kernels) transformiert werden, so dass die Vorhersage-Scores auch **nicht-lineare Decision-Boundaries** beschreiben können.
- Ein Nachteil der SVM (im Gegensatz zu Logistic Regression) ist, dass die Scores nicht als Wahrscheinlichkeiten interpretiert werden können.

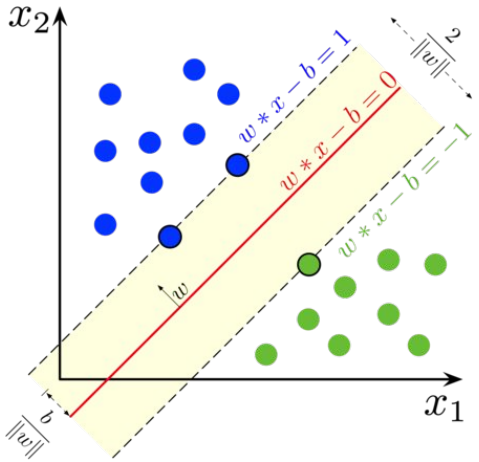
Support Vector Machines

- Es gibt 2 Unterschiede, durch die die SVM *manchmal* besser funktioniert als z.B. Perzeptron oder Logistic Regression:
    - Die SVM lernt eine Decision-Boundary, die die zwei Klassen in den Trainingsdaten (positiv, negativ) mit möglichst viel “Zwischenraum” (Margin) voneinander trennt.
    - Merkmale können mit bestimmten Funktionen (Kernels) transformiert werden, so dass die Vorhersage-Scores auch **nicht-lineare Decision-Boundaries** beschreiben können.
- Ein Nachteil der SVM (im Gegensatz zu Logistic Regression) ist, dass die Scores nicht als Wahrscheinlichkeiten interpretiert werden können.

Support Vector Machines



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.



Loss Funktionen (Training)

Eine **Loss-Funktion** ist eine mathematische Funktion, die den Unterschied zwischen vorhergesagten Werten und tatsächlichen Werten in einem Machine-Learning-Modell misst. Das Ziel ist es, den Wert der Loss-Funktion zu minimieren, um die Genauigkeit des Modells zu verbessern.

Wir lernen 3 Loss-Funktionen kennen: Zero-One-Loss, Logistic Loss und Hinge-Loss

Zero-One-Loss:

- Binäre Klassifikationsprobleme: eine Instanz einer von zwei Klassen zuzuordnen.
- Weist einen Wert von 0 zu, wenn die vorhergesagte Klasse mit der tatsächlichen Klasse übereinstimmt, und einen Wert von 1, wenn dies nicht der Fall ist.
- Es bestraft das Modell für alle Fehlklassifizierungen gleichermaßen, unabhängig davon, wie sicher das Modell in seiner Vorhersage war.

Beispiel: In einem Spam-Klassifikationsproblem würde die Zero-One-Loss-Funktion einen Wert von 0 zuweisen, wenn eine E-Mail korrekt als Spam oder Nicht-Spam klassifiziert wurde, und einen Wert von 1, wenn sie falsch klassifiziert wurde.

Loss Funktionen (Training)

Logistic-Loss:

- Binäre Klassifikationsprobleme: Für jede Instanz einen **Wahrscheinlichkeitsscore** zwischen 0 und 1 ausgibt.
- Bestraft das Modell auf der Grundlage der Sicherheit seiner Vorhersage. Es weist eine höhere Strafe für Fehlklassifizierungen zu, bei denen das Modell in seiner Vorhersage sicherer war.

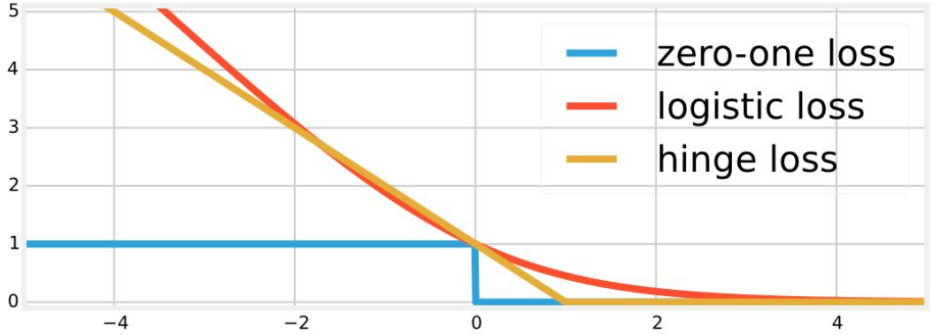
Beispiel: In einem medizinischen Diagnoseproblem würde die logistische Loss-Funktion das Modell stärker bestrafen, wenn es mit hoher Sicherheit vorhersagt, dass ein Patient eine Krankheit hat, die der Patient tatsächlich nicht hat.

Hinge-Loss:

- Binäre Klassifikationsprobleme: Für jede Instanz wird ein Score ausgegeben.
- Bestraft das Modell auf der Grundlage der Differenz zwischen dem vorhergesagten Score und dem tatsächlichen Score. Die Loss-Funktion bestraft nur Fehlklassifizierungen, die außerhalb eines bestimmten Margins liegen, und ignoriert Fehlklassifizierungen, die nahe am Margin liegen.

Beispiel: In einem Sentiment-Analyseproblem würde die Hinge-Loss-Funktion das Modell stärker bestrafen, wenn es für eine Bewertung eine negative Stimmung vorhersagt, die tatsächlich positiv ist, aber einen sehr negativen Score vergibt, anstatt eines Scores nahe Null.

Loss Funktionen (Training)

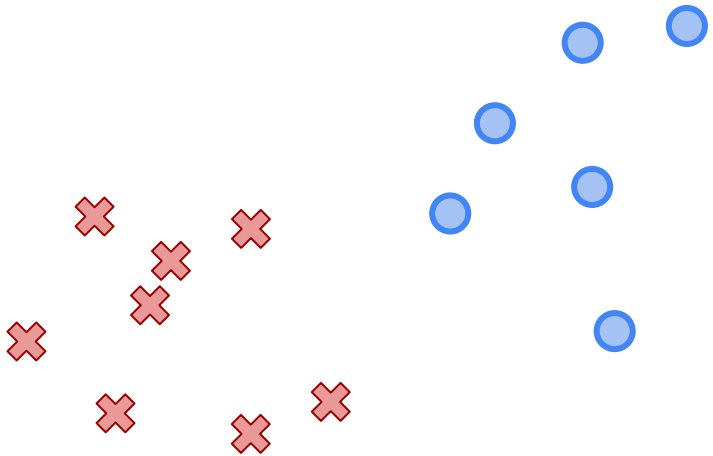


X-Achse: Vorhersage-Score für eine Trainings-Instanz (mit positivem Label – bei negativem Label ist der Graph gespiegelt)
Y-Achse: Wie schlecht der Trainings-Algorithmus diesen Score findet. (Loss)
→ Während des Trainings wird versucht, den Loss möglichst nahe 0 zu bekommen.

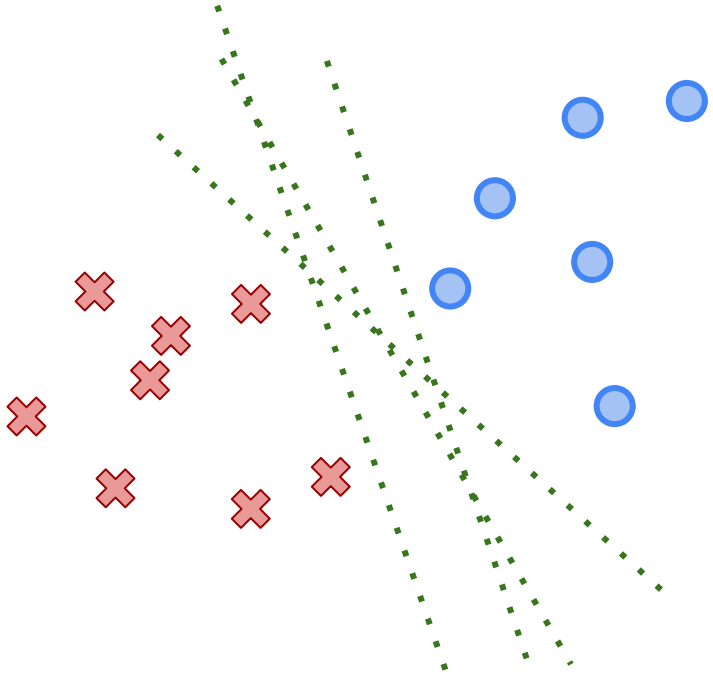
Loss für gesamten Datensatz: Summe der Losse aller Instanzen

- 0-1 loss: Perzeptron
- logistic loss: MaxEnt (bei 2 Klassen)
- hinge loss: SVM

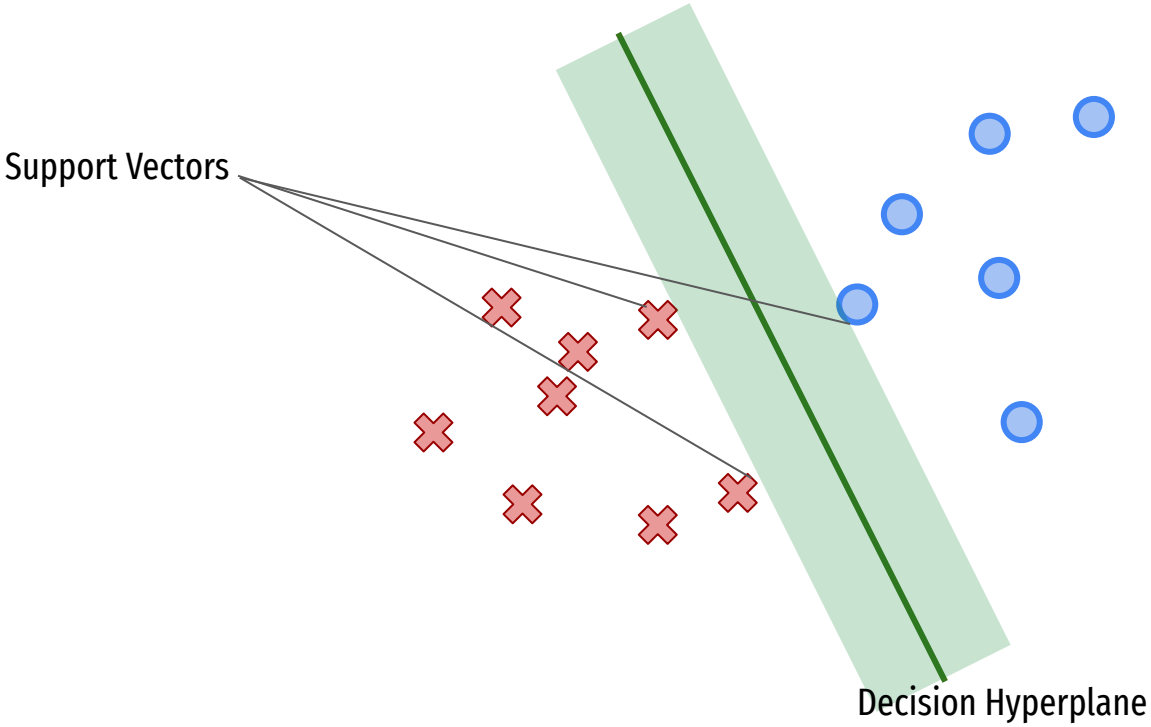
Was ist die beste Decision-Boundary?



Was ist die beste Decision-Boundary?

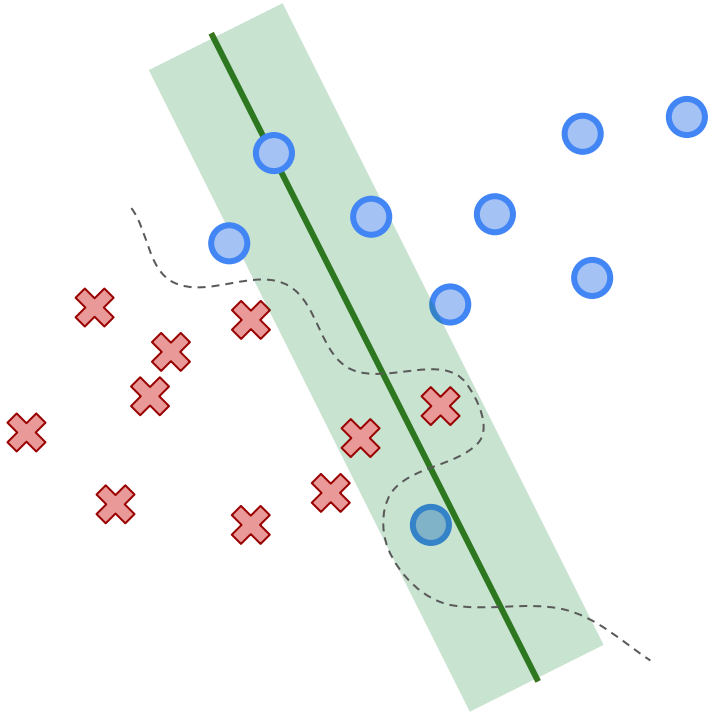


Was ist die beste Decision-Boundary?



- Effekt von Hinge-Loss: Decision Boundary wird nur durch Support-Vektoren (Trainings-Instanzen an der Margin) bestimmt.
- Andere/weitere Trainingsdaten, die bereits richtig klassifiziert würden, ändern das Ergebnis nicht.
- Modell-Parameter: Es müssen nur die Support-Vektoren gespeichert werden.

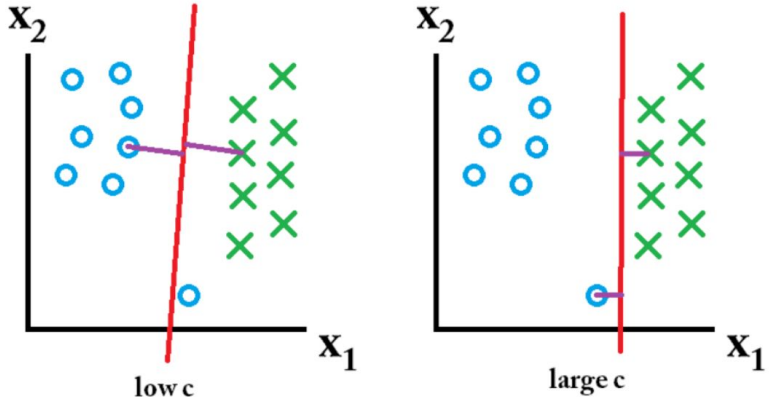
Was ist die beste Decision-Boundary?



- Bei nicht linear trennbaren Datensätzen werden 2 Ziele gegeneinander abgewogen:
  - Größe der Margin
  - Anzahl (und Entfernung) der Elemente jenseits der Margin

→ Hyper-Parameter C

Hyper-Parameter C

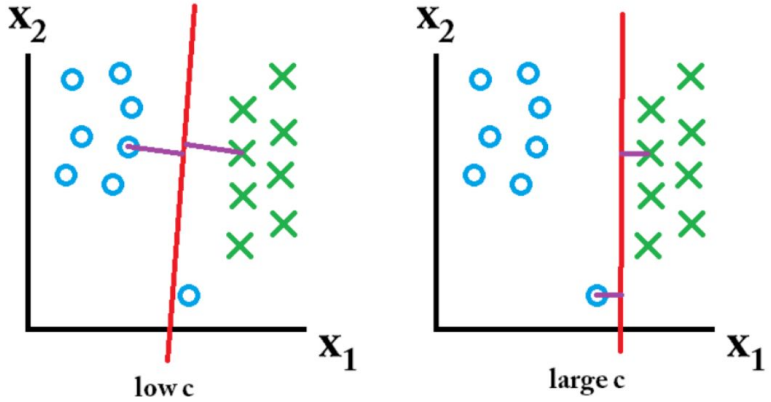


(Quelle: Stackoverflow)

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

- Kleines C: große Margin (Trainings-Instanzen jenseits der Margin OK)
- Großes C: wenige Trainings-Instanzen jenseits der Margin (kleine Margin OK)

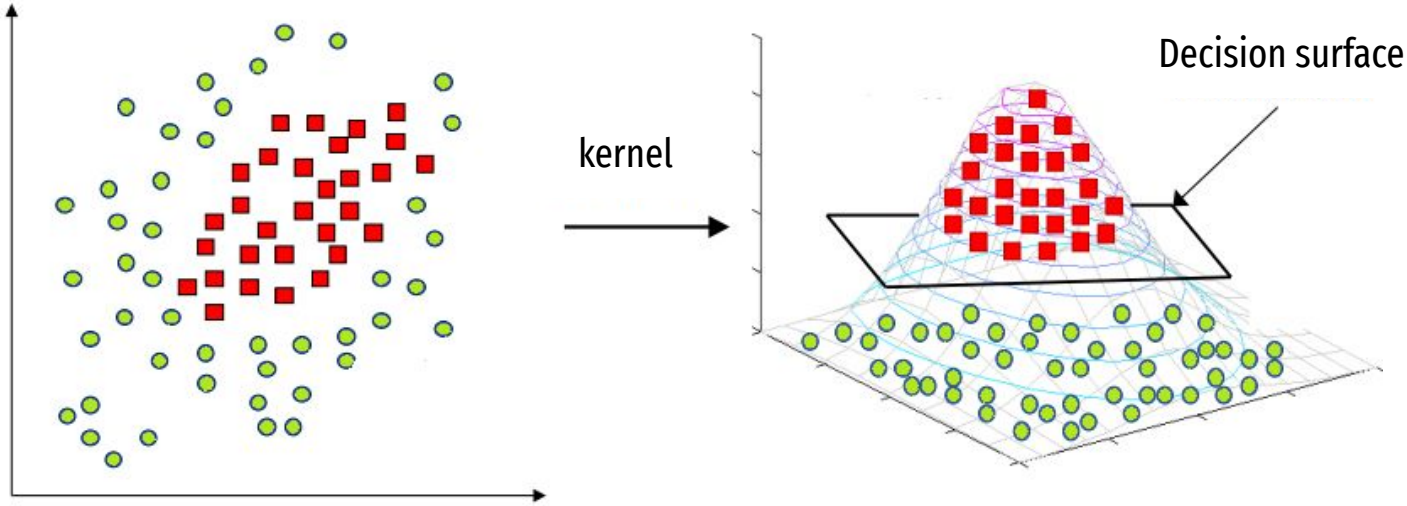
Hyper-Parameter C



(Quelle: Stackoverflow)

Mit C kontrolliert man die Strafe für falsche Klassifikationen, wobei höhere Werte von C eine höhere Strafe bedeuten. Die Wahl von C hängt von der spezifischen Aufgabe und den verfügbaren Daten ab und erfordert oft eine Abwägung zwischen der Genauigkeit des Modells auf den Trainingsdaten und dessen Fähigkeit, auf neuen Daten zu generalisieren.

Kernel SVM: Intuition



Bestimmte Funktionen (z.B. Quadratfunktion), sog. Kernels, können die Eingabe-Merkmale umformen und zusätzliche Merkmale liefern.

x → v(x)

Mit den neuen Merkmalen (zusätzliche Dimensionen) wird das Problem dann manchmal (besser) trennbar.
Dieses Vorgehen nennt man den Kernel-Trick: linear nicht trennbar → linear trennbar

Zusammenfassung

Maximum Entropy Klassifikator

- = Logistic Regression = logit model = log-lineares Modell
- Eine der wichtigsten Methoden zur Klassifizierung
- Lineare Regression + Exponentialfunktion + Normalisierung
- Kann als Wahrscheinlichkeitsmodell aufgefasst werden

Support Vector Machine

- Eine weitere viel verwendete Methode zur Klassifikation
- Lineare Vorhersage + Max-Margin + (optional) Kernel

MaxEnt und SVM sind mit Scikit-learn einfach zu verwenden