Vorlesungsplan

- 17.10. Einleitung
- 24.10. Ein- und Ausgabe
- 31.10. Reformationstag, Einfache Regeln
- 7.11. Naïve Bayes, Entscheidungsbäume
- 14.11. Entscheidungsregeln, Assoziationsregeln
- 21.11. Lineare Modelle, Instanzbasiertes Lernen
- 28.11. Clustering
- 5.12. Evaluation
- 12.12. Evaluation
- 19.12. Lineare Algebra für Data Mining
- 9.1. Statistik für Data Mining
- 16.1. **Lineare Modelle, Support Vector Machines (SVM)**
- 19.1. Vorlesung statt Übung: Bayes-Netze
- 23.1. Clustering
- 26.1. Vorlesung statt Übung: Kombination von Modellen, Lernen von nicht-klassifizierten Beispielen
- 30.1. Finden von häufigen Teilstrukturen
- 2.1. Klausur

Erweiterung von linearen Modellen

- Lineare Klassifikatoren können keine nicht-linearen Klassengrenzen modellieren
- Einfacher Trick:
  - Bilde die Attribute in einen neuen Raum ab, dessen Dimensionen Attributkombinationen sind
  - z.B. : alle Produkte mit *n* Faktoren, die aus den Attributen konstruiert werden können
- Beispiel mit zwei Attributen und *n* = 3:

$$w_1 x_1^{\ 3} + w_2 x_1^2 x_2 + w_3 x_1 x_2^2 + w_3 x_2^3$$

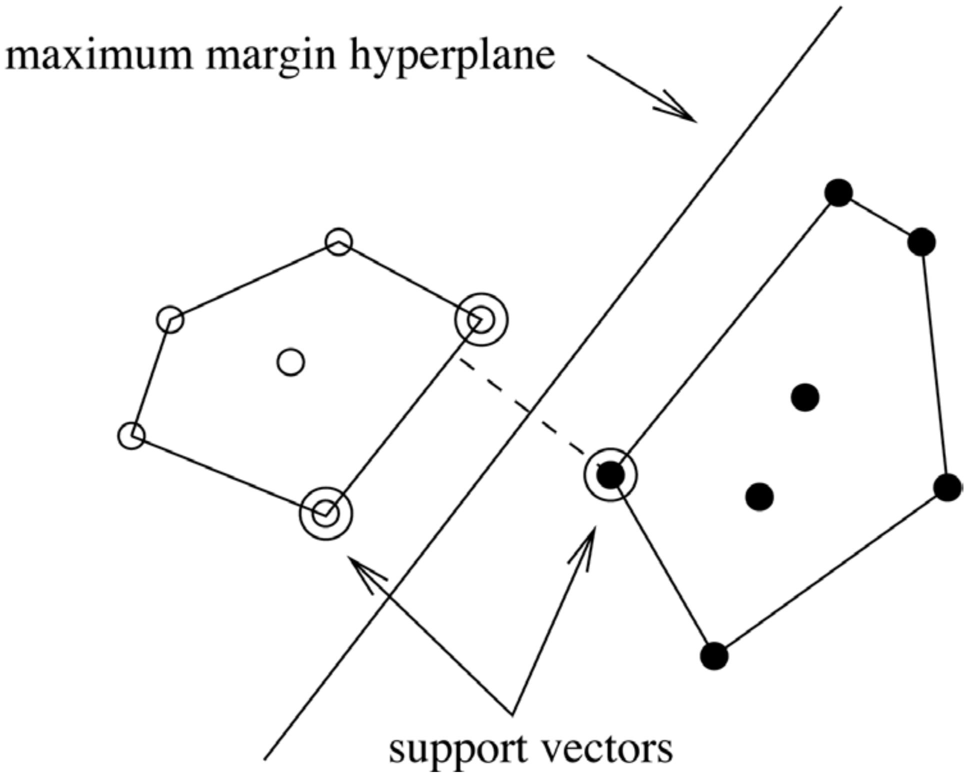
Probleme dieses Ansatzes

- 1. Problem: Geschwindigkeit
  - 10 Attribute und $n = 5 \Rightarrow >2000$ Koeffizienten
  - Nutze lineare Regression mit Attributauswahl
  - Laufzeit ist kubisch in der Anzahl der Attribute
- 2. Problem: Overfitting
  - Anzahl der Koeffizienten ist groß im Verhältnis zur Anzahl der Trainingsinstanzen

Support vector machines

- *Support vector machines* (SVM) sind Algorithmen zur Lernen linearer Klassifikatoren
- Robust gegen Overfitting weil sie eine spezielle lineare Klassengrenze lernen:
  - Die Hyperebene mit dem maximalen Abstand zwischen den Klassen (*maximum margin hyperplane)*
- Schnell im nicht-linearen Fall
  - Nutzen einen mathematischen Trick um die "Pseudo-Attribute" nicht zu erzeugen
  - Der nicht-lineare Raum wird implizit erzeugt

Hyperebene mit dem maximalen Abstand

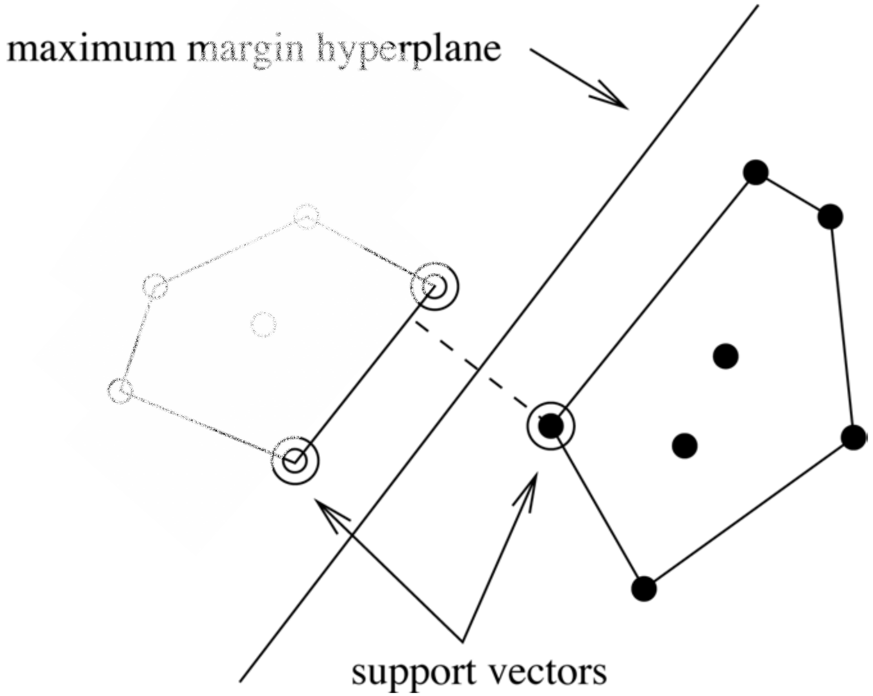


- Die nächsten Instanzen zur Maximum Margin Hyperplane werden *Support Vektoren* genannt

Support Vektoren

- Die Support Vektoren definieren die Hyperebene mit dem maximalen Abstand
  – Alle anderen Instanzen können gelöscht werden, ohne die Position und Orientierung dieser Ebene zu ändern
- Das bedeutet, daß die Ebene geschrieben werden kann als

$$w_0 + w_1 x_1 + w_2 x_2$$
$$= b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i \mathbf{x}_i \bullet \mathbf{x}$$



Bestimmen von Support Vektoren

$$b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i \mathbf{x}_i \cdot \mathbf{x}$$

- Support Vektor: Trainingsinstanz mit $\alpha_i > 0$
- Bestimmen der α_i und *b* ?— Quadratisches Optimierungsproblem mit Nebenbedingungen
  - Standardwerkzeuge für diese Probleme
  - Spezialalgorithmen sind schneller
  - Beispiel: Platt's Algorithmus (implementiert in WEKA)
- Bemerkung: linear separierbare Daten sind vorausgesetzt

Nicht-lineare SVMs

- “Pseudoattribute” repräsentieren Attributkombinationen
- Overfitting ist kein großes Problem, da Hyper-ebene mit dem maximalen Abstand stabil ist
  - Die Anzahl der Support-Vektoren ist relativ klein im Verhältnis zur Größe der Trainingsmenge
- Rechenzeit ist noch problematisch
  - Für jedes zu berechnende Skalarprodukt müssen alle “Pseudo Attribute” eingebunden werden

Der Kernel Trick

- Vermeidet die Berechnung der “Pseudo Attribute”!
- Berechne Skalarprodukt vor der nicht-linearen Transformation
- Beispiel: für

$$b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i \mathbf{x}_i \cdot \mathbf{x}$$

  berechne

$$b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i (\mathbf{x}_i \cdot \mathbf{x})^q$$

- Korrespondiert zu einer Abbildung in den Raum der durch alle Produkt mit *q* Attributen aufgespannt wird

Andere Kern-Funktionen

- Abbildung wird “Kern-Funktion” genannt
- Polynomieller Kern $$b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i (\mathbf{x}_i \cdot \mathbf{x})^n$$
- We can use others: $$b + \sum_{i \text{ ist Supp. Vektor}} \alpha_i r_i K(\mathbf{x}_i \cdot \mathbf{x})$$
- Einzige Bedingung: $$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \bullet \phi(\mathbf{x}_j)$$
- Beispiele: $$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \bullet \mathbf{x}_j + 1)^d$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-(\mathbf{x}_i - \mathbf{x}_j)^2}{2\sigma^2}}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i \bullet \mathbf{x}_j + b)$$

Rauschen

- Bisher haben wir angenommen, daß die Daten linear separierbar sind (im Original- oder im transformierten Raum)
- SVMs können verrauschte Daten klassifizieren, wenn ein “Rausch”-Parameter C eingeführt wird
- C beschränkt den Einfluß einer Trainingsinstanz auf die Klassengrenze
- Das Optimierungsproblem bleibt quadratisch
- C muß experimentel bestimmt werden

Dünne Daten

- SVM Algorithmen können stark beschleunigt werden, wenn dünne Daten vorliegen, d.h. viele Werte in den Instanzvektoren 0 sind
- Grund: es werden viele Skalarprodukte berechnet
- Dünne Daten $\Rightarrow$ Berechnung von Skalarprodukten ist sehr effizient
- SVMs können dünne Daten mit 10^4-10^5 Attributen bearbeiten

Anwendungen

- Bildverarbeitung
  - Gesichtserkennung
  - Erkennen von Postleitzahlen
- Bioinformatik:
  - Vorhersage der Sekundärstruktur von Proteinen
- Text Klassifikation
- SVM Technik kann für numerische Vorhersage angepaßt werden