

# 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 I
- 5.12. Clustering II
- 12.12. Evaluation I
- 19.12. Evaluation II
- 9.1. Entscheidungsbäume, Klassifikationsregeln
- 16.1. Lineare Modelle, Numerische Vorhersage
- 23.1. Clustering
- 30.1. Attribut-Selektion, Diskretisierung, Transformationen
- 6.2. Kombination von Modellen, Lernen von nicht-klassifizierten Beispielen

# Lineare Modelle

- Sind auf numerischen Attributen definiert
- Standard Technik für numerische Vorhersage: lineare Regression
  - Ergebnis ist Linearkombination der Attribute

$$x = w_0 + w_1 a_1 + w_2 a_2 + \dots + w_k a_k$$

- Gewichte berechnet aus den Trainingsdaten
- Vorhergesagter Wert für die erste Trainings-Instanz  $\mathbf{a}^{(1)}$  (Index oben bezeichnet i-te Instanz)

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \dots + w_k a_k^{(1)} = \sum_{j=0}^k w_j a_j^{(1)}$$

# Methode der kleinsten Quadrate

- Wähle  $k + 1$  Koeffizienten um das Quadrat des Fehlers auf den Trainingdaten zu minimieren
- Quadrierter Fehler: 
$$\sum_{i=1}^n \left( x^{(i)} - \sum_{j=0}^k w_j a_j^{(i)} \right)^2$$
- Berechne Koeffizienten mittels standard Matrix Operationen
- Sinnvoll wenn mehr Instanzen als Attribute ( $n > k$ )
- Minimieren des *absoluten Fehlers* ist schwieriger

# Klassifikation

- Jede Regressionstechnik kann zur Klassifikation genutzt werden
  - Training: führe Regression für jede Klasse  $c$  durch indem für die Trainingsinstanzen, die zu  $c$  gehören, das Ergebnis 1 und für den Rest auf 0 gesetzt wird
  - Vorhersage: berechne Regressionswert für alle Klassen, wähle Klasse nach größtem Ergebnis aus (Mitgliedschaftswert)
- Für lineare Regression heißt dies *multi-response linear regression*

# Klassifikation durch paarweise Regression

- Zweite Möglichkeit Regression zur Klassifikation zu nutzen:
  - Ein Regressionsfunktion für jedes Klassenpaar unter Verwendung nur der Instanzen dieser zwei Klassen
  - Ergebnis für die eine Klasse ist +1 für die andere -1
- Vorhersage durch Abstimmung
  - Klasse mit den meisten Stimmen wird ausgewählt
  - Alternative: “weiss nicht” falls kein Übereinstimmung

# Logistische Regression

- Problem: Lineare Regression für mehrere Klassen liefert Mitgliedschaftswerte außerhalb  $[0, 1]$ , daher keine Wahrscheinlichkeiten
- *Logistische* Regression: Alternative zu linearer Regression
  - Schätzt Klassenwahrscheinlichkeiten direkt
    - mittels *maximum likelihood* Methode
  - Nutzt lineares Modell:

$$\log\left(\frac{P}{1-P}\right) = w_0 a_0 + w_1 a_1 + w_2 a_2 + \dots + w_k a_k$$

← *Klassenwahrscheinlichkeit*

# Lineare Klassifikation mit Perzeptronen

- Perzeptron lernt Hyperebene um zwei Klassen zu trennen

$$w_0 a_0 + w_1 a_1 + \dots + w_k a_k = 0$$

Set all weights to zero

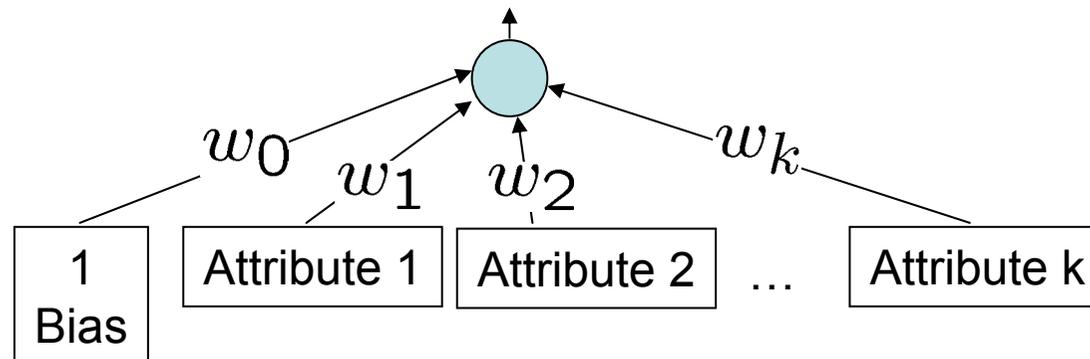
Until all instances in the training data are classified correctly

For each instance  $I$  in the training data

If  $I$  is classified incorrectly by the perzeptron

If  $I$  belongs to the first class then add it to the weight vector

Else subtract it from the weight vector



# Diskussion linearer Modelle

- Nicht sinnvoll wenn Daten starke nicht-lineare Abhängigkeiten enthalten
- Aber: können als Bausteine für komplexere Schemata dienen (z.B. Regressionsbäume)
- Beispiel: Multi-response lineare Regression definiert eine *Hyperebene* für jedes Klassenpaar:

$$(w_0^{(1)} - w_0^{(2)})a_0 + (w_1^{(1)} - w_1^{(2)})a_1 + (w_2^{(1)} - w_2^{(2)})a_2 + \dots + (w_k^{(1)} - w_k^{(2)})a_k > 0$$

# Instanz-basierte Repräsentation

- Einfachste Form des Lernen: *Auswendiglernen*
  - Trainingsinstanzen werden nach der Instanz durchsucht, die einer neuen Instanz am nächsten kommt
  - Die Instanzen selbst repräsentieren das Wissen
  - Auch *instanz-basiertes* Lernen genannt
- Distanz/Ähnlichkeitsfunktion definiert was gelernt heißt
- Instanz-basiertes Lernen ist *faul*
- Methoden:
  - *nächste Nachbar*
  - *k-nächste Nachbarn*
  - ...

# Die Distanz Funktion

- Einfachster Fall: ein numerisches Attribut
  - Distanz ist die Differenz zwischen zwei Attributwerten (oder einer Funktion davon)
- Mehrere numerische Attribute: häufig wird Euklidische Distanz genutzt und Attribute sind normalisiert
- Nominales Attribut: Distanz ist 1 falls Werte verschieden sind, 0 sonst
- Sind alle Attribute gleichwichtig?
  - Gewichtung der Attribute kann notwendig sein

# Normalisierung

- Unterschiedliche Attribute werden auf verschiedenen Skalen gemessen  $\Rightarrow$  Normalisierung
  - Min-Max Normalisierung 
$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

$v_i$ : the actual value of attribute  $i$
  - Zero Mean, Varianz 1 
$$a_i = \frac{v_i - \text{mean}(v_i)}{\text{sdev}(v_i)}$$
- Oft angewandte Regel für fehlende Werte: Annahme, diese sind maximal weit weg, bezüglich normalisierter Attribute

# Diskussion von 1-NN

- Oft sehr genau
- ... aber sehr langsam:
  - einfache Version vergleicht gegen alle Trainingsdaten für eine Vorhersage
- Annahme, daß alle Attribute gleichwichtig sind
  - Abhilfe: Attributauswahl oder Gewichte
- Möglichkeiten gegen verrauschte Instanzen :
  - Mehrheit der  $k$  nächsten Nachbarn
  - Löschen von verrauschten Instanzen aus den Trainingsdaten (schwierig)
- Statistiker nutzen  $k$ -NN seit den frühen 1950zignern
  - Falls  $n \rightarrow \infty$  und  $k/n \rightarrow 0$ , Fehler wird minimal