Klassifikation II

Praktikum
Data Warehousing und
Data Mining

Kostenbewusstes Lernen

Kostenbewusstes Lernen I

- „Falsch ist nicht gleich falsch!“
- Kosten-Matrix wenn keine Kosten vorgegeben:

		Vorhersage: Ja		Vorhersage: Nein	
Tatsächliche Klasse	Ja	0	(TP)	1	(FN)
	Nein	1	(FP)	0	(TN)

- Die Einträge in der Matrix beschreiben die Kosten die bei FP und FN entstehen.
- Beispiel Geldscheinprüfer:
  - FP=50,00 EUR; FN=0,01 EUR

Kostenbewusstes Lernen II

- Kostenbewusstes Lernen:
  - Detaillierte Vorgaben über Kosten für Fehlklassifizierungen
  - Variante: Gewinn-Matrix gegeben
- Möglichkeiten zum kostenbewussten Lernen:
  - Variieren der Klassengrößen im Lerndatensatz (durch Vervielfachung von Instanzen)
  - Einführen von Klassengewichtungen (z.B. Terminierungsbedingungen und Werte in Blattknoten von Entscheidungsbäumen)
  - Schwellwertsetzung auf der Konfidenz eines Klassifikators („Klassifiziere nur dann mit JA, wenn Konfidenz > 70%“)
    - Achtung: Schwellwert-Findung ist Teil des Lernens!
- Verschiedene Tools wie WEKA und C5.0 in Clementine bieten Kosten-Matrizen von Haus aus an.

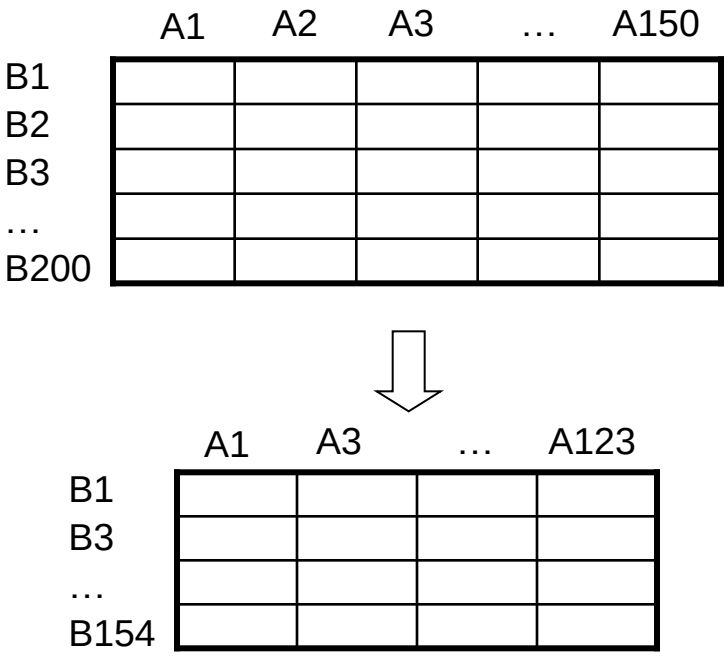
z.B. duplizieren der positiven Bewertung. weka-Knoten: meta-cost

Man benötigt Anwendungswissen - erhält man aus Erfahrung

beim Lernen: in dem Blattknoten hat man Tupel, die letztendlich gewichtet werden.

diese beinhaltet die Varianten des kostenbewussten Lernens

Feature Selection



Subset der Spalten und Zeilen:
Samplen damit weniger Instanzen
Spalten die nicht wichtig für Klassifikation

Feature Selection I

- Ziele:
  - Skalierbarkeit von Klassifikationsalgorithmen durch das Entfernen „irrelevanter“ Attribute
  - Eingrenzen des Curse of Dimensionality

    `in hochdimensionalen Räumen sind Abstände nicht mehr so aussagekräftig`
- Vorgehen:
  1. Expertenwissen nutzen („Es war in unserer Versicherung seit ich denken kann so, dass Kunden mit einem Einkommen unter 20.000 EUR Probleme mit der pünktlichen Zahlung haben.“) `nicht beim DMC möglich!`
  2. Automatische Entscheidung, welche Attribute gut sind

     `h`
- Warnung:
  - Manche Attribute sind nur in Kombination mit anderen Attributen von Interesse!

Feature Selection II

- Screening mit einfacher Statistik (Beispiele!):
  - Anzahl Ausprägungen eines Attributs: Löschen wenn weniger als n (Beispiel: l=3)
  - Anteil der NULL-Werte eines Attributs: Löschen wenn mehr als k% (Beispiel: l_k=70%)
  - Minimale Varianz eines Attribute: Löschen wenn kleiner als l_m (Beispiel: m = 0,1)
- Verwendung von Attributauswahlmaßen (Rankings) – selektiere TOP-n(%) Attribute
  - Korrelation mit dem Klassenattribut (Pearsons Korrelationskoeffizient, Chi-Quadrat-Maß etc.)
  - Informationsgehalt (InfoGain oder besser GainRatio)
  - Gini-Index u.a. Maße bekannt von Entscheidungsbäumen
- Zuhilfenahme eines Klassifikators
  - Welche Attribute wählt ein Entscheidungsbaum bzw. ein anderer Klassifikator und wie wichtig sind diese dafür?

das was wir herausfinden wollen

Klassifikator: wie wichtig sind die Attribute in Kombination?

Attribut, dass dauernd geteilt wird und auf 70% des Entscheidungsbaums verteilt ist sollte nicht so sehr beachtet werden sollte

Bayes Klassifikator

Bayes Klassifikation - Idee

- Gegeben sei
  - Ereignis X
  - Hypothese H: Data Tupel X gehört zu Klasse C

- Ziel der Klassifikation
  - Maximiere p(H | X)
    - p(H | X): Bedingte Wahrscheinlichkeit, dass H stimmt, gegeben Tupel X
  - Problem:
    - P(H | X) lässt sich nicht aus Daten bestimmen

- Bayes Theorem
  - $p(H \mid X) = \frac{p(X \mid H)\, p(H)}{p(X)}$
  - Vorteil: p(X), p(H) und p(X | H) lassen sich hier bestimmen

Naive Bayes Klassifikator I

- Geben sei
  - Trainingsmenge D
  - Attribute A_i mit i = {1, 2, …, n}
  - m Klassen C_j mit j = {1, 2, …, m}
- Tupel gehört zu C_i, wenn
  $p(C_i \mid X) > p(C_j \mid X)$ für $1 \leq j \leq m, j \neq i$
- Ziel also
  - Maximierung von $p(C_i \mid X) = \frac{p(X \mid C_i)\, p(C_i)}{p(X)}$
  - p(X) ist konstant für alle Klassen, also
  - Maximierung von $p(X \mid C_i)\, p(C_i)$

Naive Bayes Klassifikator II

- Vereinfachungen für die Berechnung
  - Bestimmung von $p(C_i)$ `wie oft kommt klasse in lerndatensatz vor?`
    - Abschätzung:
      - $p(C_i) = |C_{i,D}| / |D|$
        - $|C_{i,D}|$ ist Anzahl der Trainingstupel von Klasse C_i in D
  - Bestimmung von $p(X \mid C_i)$:
    - Die Unabhängigkeit der Wirkung der einzelnen Attribute auf die Klasse angenommen, gilt:
      $p(X \mid C_i) = \pi^n_{k=1} p(x_k \mid C_i)$
      `annahme: naive annahme: wirkung der einzelnen attribute von der klassenzugehörigkeit unabhängig`
    - $p(x_k \mid C_i) = z / |C_{i,D}|$
      - z ist die Anzahl der Tupel in Klasse C_i mit Attributwert x_k
      - $|C_{i,D}|$ ist die Anzahl der Trainingstupel von Klasse C_i in D
- Klassifikation
  - Klasse von X bestimmt durch Berechnung von $p(X \mid C_i)\, p(C_i)$ für alle Klassen

Naiver Bayes Klassifikator - Beispiel

Gesucht: Klassifikation für:

X = {jung, mittel, ja, schlecht}

p(ja) = 9/14

p(nein) = 5/14

p(jung | ja) = 2/9 `einfach die Kombination`

p(jung | nein) = 3/5

p(mittel | ja) = 4/9

p(mittel | nein) = 2/5

p(ja | ja) = 6/9

p(ja | nein) = 1/5

p(schlecht | ja) = 6/9

p(schlecht | nein) = 2/5

~~P(X | ja)~~ = 9/14 * 2/9*4/9*6/9*6/9

`p(C_i)*bed.Wahrsch.` = 0,0282

~~P(X | nein)~~ = 5/14 * 3/5*2/5*1/5*2/5

= 0,0069

Vorhersage: ja (kauft PC)

Alter	Einkommen	Student	Kreditwürdigkeit	Klasse: Kauft PC
Jung	Hoch	Nein	Schlecht	Nein
Jung	Hoch	Nein	Gut	Nein
Mittelalt	Hoch	Nein	Schlecht	Ja
Senior	Mittel	Nein	Schlecht	Ja
Senior	Niedrig	Ja	Schlecht	Ja
Senior	Niedrig	Ja	Gut	Nein
Mittelalt	Niedrig	Ja	Gut	Ja
Jung	Mittel	Nein	Schlecht	Nein
Jung	Niedrig	Ja	Schlecht	Ja
Senior	Mittel	Ja	Schlecht	Ja
Jung	Mittel	Ja	Gut	Ja
Mittelalt	Mittel	Nein	Gut	Ja
Mittelalt	Hoch	Ja	Schlecht	Ja
Senior	Mittel	Nein	Gut	Nein

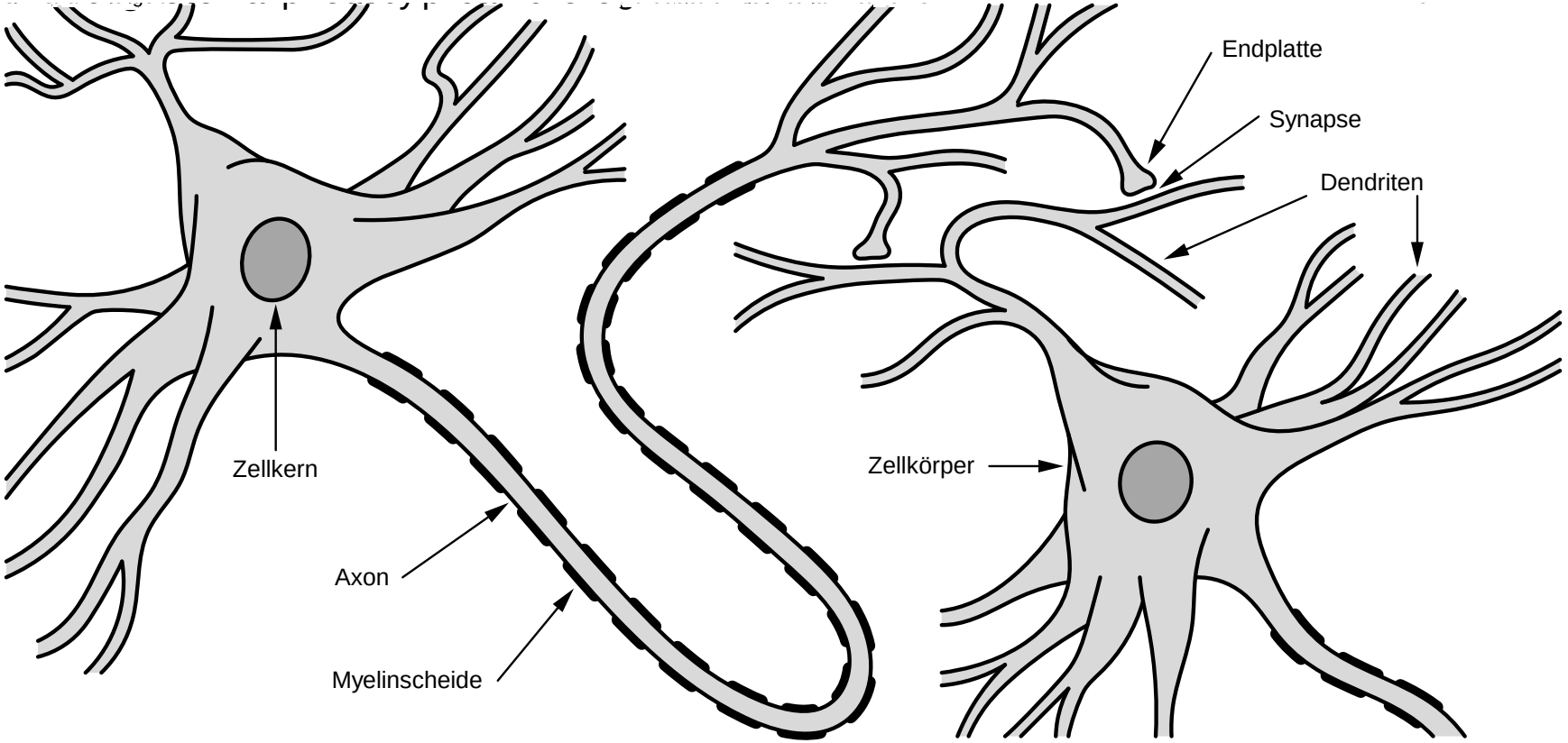
Künstliche Neuronale Netze

Künstliche Neuronale Netze – Idee

- Ausgangssituation
  - Eingabegrößen: Mehrere beliebige Attribute
  - Zielgröße: Vorhersage einer binären, kategorischen oder numerischen Variablen

- Idee: Nachbildung der kognitiven Fähigkeiten des menschlichen Gehirns
  - Netzwerk aus Neuronen (Nervenzellen) „verknüpft“ Eingabegröße mit Zielgröße
    - Beispiel: Auge sieht Bier, Gehirn meldet Durst
  - Definition Neuron
    - Binäres Schaltelement mit zwei Zuständen (aktiv, inaktiv)

Struktur des Neurons in der Biologie

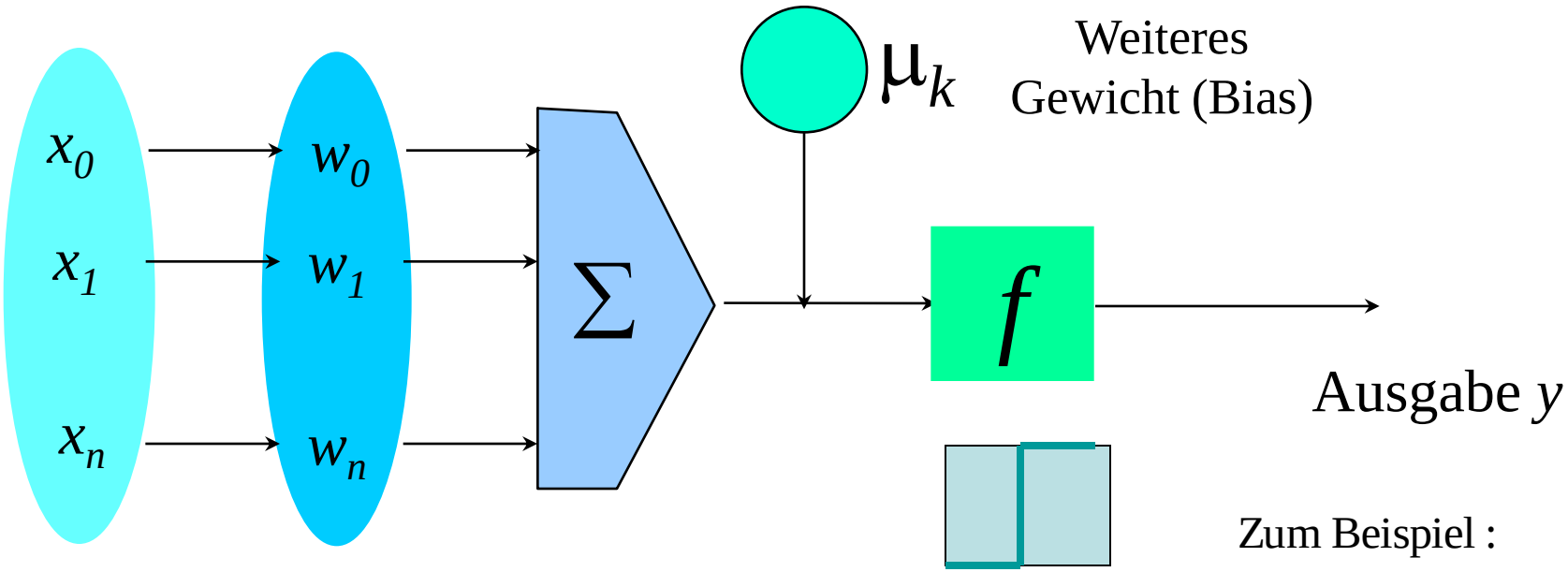


Arbeitsweise von Neuronen

- Die Synapsen an den Enden der Axone senden chemische Stoffe aus, sog. Neuro-Transmitter.
- Diese wirken auf die Rezeptoren der Dendriten, deren Spannungspotential ändert sich.
- Man unterscheidet zwischen
  - exzitatorischen (erregenden) Synapsen
  - inhibitorischen (hemmenden) Synapsen
- Bei genügend exzitatorischen Reizen (netto, über gewisse Zeitspanne) wird das Neuron aktiv.
- Aktive Neuronen senden selbst wieder Signale zu benachbarten Neuronen…

Das einfache Perzeptron (künstliches Neuron)

binäres Schaltelement



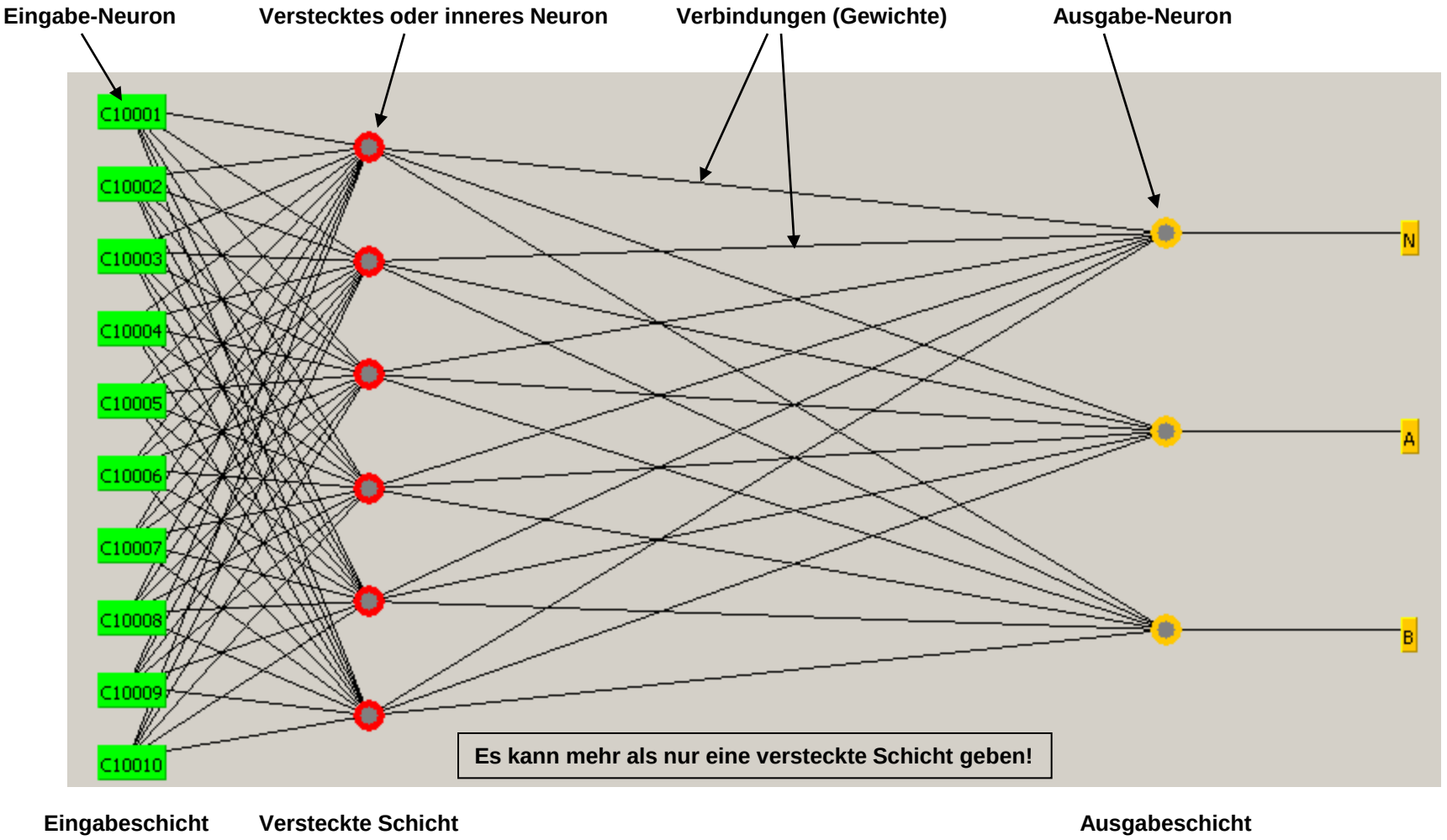
Zum Beispiel :

$$y = \text{sign}(\sum_{i=0}^{n} w_i x_i + \mu_k)$$

- Der *n*-dimensionale Eingabe-Vektor $\boldsymbol{X}$ wird durch ein Skalarprodukt und eine nichtlineare Funktion auf *y* abgebildet.

entweder positiv oder negativ

Neuronale Netze - Multilayer-Perceptron (MLP)



Künstliche Neuronale Netze – Arbeitsweise

- Vorgehen Klassifikation/Regression
  - Gegeben: Netzwerk aus Neuronen
  - Alle Neuronen inaktiv, senden keine Signale
  - Eingabeneuronen gemäß Eingabegrößen gereizt
    $\Rightarrow$ Gereizte Neuronen senden Signale
  - Signale werden über Netzwerk zum Ausgabeneuron weitergeleitet
  - Regression:
    - Ausgabeneuron liefert kontinuierlichen Wert.
  - Klassifikation (binär):
    - Schwellwertsetzung am Ausgabeneuron.
  - Klassifikation (allgemein)
    - Ausgabeneuron mit „höchstem Reiz“ definiert Klasse.

Lernen von neuronalen Netzen

- Zunächst: Definition der Netzstruktur
  - Erfahrungswerte oder „Trial and Error“…

- Dann: Lernen der Gewichte
  1. Initialisiere Gewichte und Bias mit zufälligen Werten
  2. Propagiere die Werte eines Lerntupels durch das Netz
  3. Berechne den Fehler, Anpassen von Gewichten und Bias
  4. Wiederhole 2 und 3 bis Stoppkriterium erreicht
     (z.B. Fehler hinreichend klein oder Zeitüberschreitung)
  - Anpassung findet entweder nach jedem Tupel statt oder nach jeder Epoche (ganzer Lerndatensatz)
    - Variante: Eine Epoche besteht aus *n* zufälligen Lerndatensätzen.

Lernen der Gewichte – einfaches Perzeptron

- Anpassen erfolgt durch Delta-Regel:

- das neue w_i
  $w_i' = w_i + \Delta w_i$
  gewichtsvektor

$$\Delta w_i = \begin{cases} 0 & wenn \quad y_p = y \\ +\sigma x_i & wenn \quad y_p = 0 \wedge y = 1 \\ -\sigma x_i & wenn \quad y_p = 1 \wedge y = 0 \end{cases}$$

vorhersage = ist

- $\mu' = \mu + \Delta\mu$

$$\Delta\mu = \begin{cases} 0 & wenn \quad y_p = y \\ -\sigma & wenn \quad y_p = 0 \wedge y = 1 \\ +\sigma & wenn \quad y_p = 1 \wedge y = 0 \end{cases}$$

- w_i: Ein Gewicht des Perzeptrons
- μ: Bias des Perzeptrons
- $(x_1, x_2, \ldots, x_n)$: Ein Eingabemuster
- y: Zugehöriger Zielwert
- y_p: Berechneter Ausgabewert
- σ: Lernrate (Benutzerdefiniert)

Grenzen von Perzeptron & Delta-Regel

- Perzeptron-Konvergenztheorem:
  - Wenn es eine Lösung gibt, dann findet die Delta-Regel auch eine geeignete Kombination von Gewichten (vereinfacht).

- Was kann das einfache Perzepton lösen?
  - Linear-separierbare Klassifikationsprobleme.
  - Das MLP kann auch nicht linear separierbare Probleme lösen.

Gerade, die beide Typen trennt

Entscheidungsbaum zieht horizontale und vertikal Linie.

Lernen der Gewichte – MLP

- Generalisierung der Delta-Regel: **Backpropagation**
- Ziel: Minimierung des Fehlers und Festlegen der Gewichte/Bias-Werte; Netzwerk ist vorgegeben.
- Lösung: Gradientenverfahren
  - Aktivierungsfunktion muss differenzierbar sein:

    Sigmoidfunktion statt *sign*: $sig(x) = 1 / (1 + e^{-x})$

    Mit Bias und Steilheit α: $sig(x) = 1 / (1 + e^{-\alpha(x-\mu)})$

    `ähnlich der sign(), nur diffbar. b`
  - Fehlerfunktion muss differenzierbar sein.
- Funktioniert auch bei mehreren versteckten Ebenen und mehreren Ausgabeneuronen.
- Gradientenverfahren liefert lokales Minimum
  - σ ändern oder initiale Gewichte bzw. Bias variieren.

Neuronale Netze - Bewertung

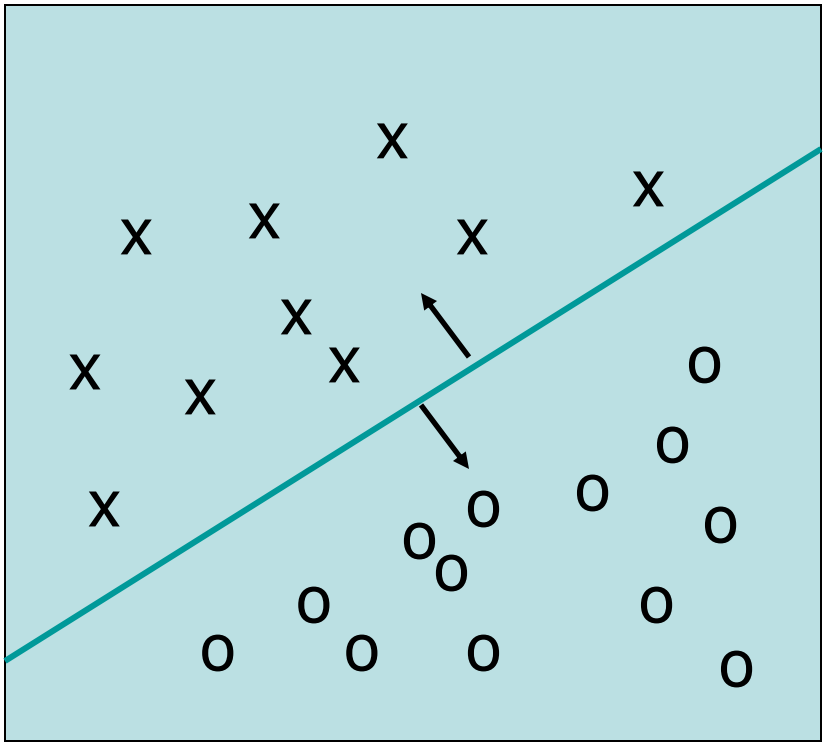
- Herausforderungen
  - Aufbereiten der Daten
    - Üblich: Normalisierung auf 0…1
    - Bei kategorischen Daten:
      ggf. ein Eingabeneuron pro Attribut-Ausprägung
  - Aufbau des Netzes
    - Erfahrungswerte oder „Trial and Error“.
  - Verhinderung von Overfitting
    - Evaluation mit neuen Daten
  - Voraussagewert bei Regressionsproblemen
    - Lineare Funktion an Ausgabeneuron und Skalieren des Wertes
- Vorteile
  - Gutes Verhalten bei neuen und verrauschten Daten
- Nachteile
  - Lernen oft vergleichsweise aufwändig
  - Ergebnis schwer zu interpretieren

Support-Vektor-Maschinen (SVMs)

sehr populär, "State of the Art"

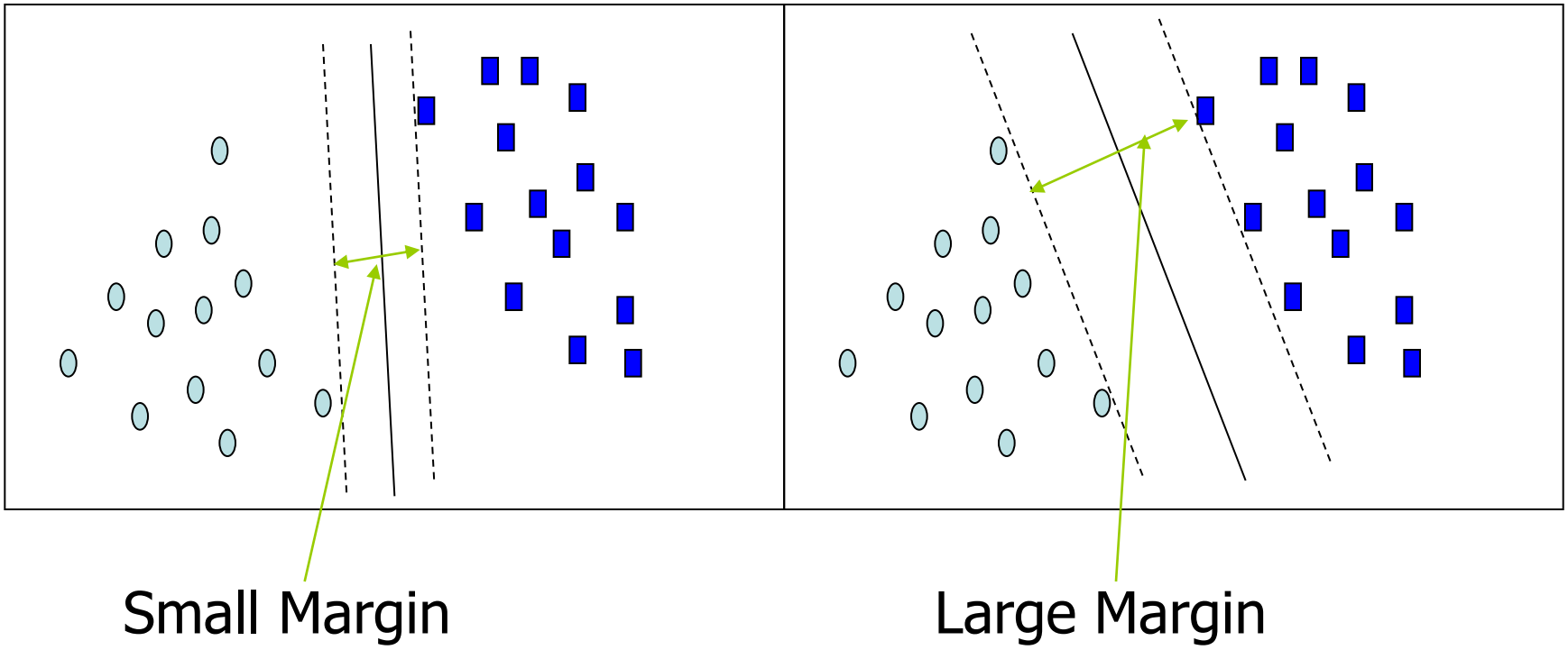
Support-Vektor-Maschinen - Motivation

- Relativ neue Klassifikationstechnik
- Nativ für binäre Probleme
- Gesucht ist eine Hyperebene, die optimal zwei Klassen separiert
  - 1D: Grenzwert
  - 2D: Gerade
  - 3D: Ebene
  - 4D etc.: Hyperebene
- Auch nicht linear separierbare Fälle lösbar…



Linear separierbares Beispiel für den 2D-Fall

SVMs - Finden von Hyperebenen (linear separierbar)



- Ziel: Finden einer Hyperebene mit max. Margin.
  - So entsteht ein generalisierender Klassifikator.

So schief, dass der Abstand zu der Geraden maximiert ist.

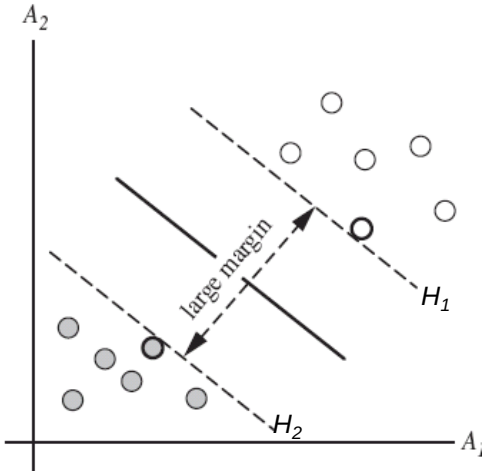
Finden einer separierenden Hyperebene

- Eine Hyperebene kann wie folgt beschrieben werden:

  $\boldsymbol{W} \bullet \boldsymbol{X} + w_0 = 0$

  $\boldsymbol{W} = \{w_1, w_2, \ldots, w_n\}$ ist Vektor von gesuchten Gewichten

  $\boldsymbol{X}$ ist Lerndatensatz



- Im 2D-Fall z.B.:

  $w_0 + w_1 x_1 + w_2 x_2 = 0$

- Für die Rand- Hyperebenen gilt dann:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ für $y_i = +1$, und

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ für $y_i = -1$

- Die Tupel des Lerndatensatzes auf H_1 und H_2 heißen Stützvektoren (support vectors)

Berechnung der Hyperebene

- Das Bestimmen von $\boldsymbol{W} = \{w_1, w_2, \ldots, w_n\}$ ist ein quadratisches Optimierungsproblem mit Constraints.
  - Lösbar mit der Lagrange-Multiplikatorenregel.
  - S. Bücher von V. Vapnik.
- Die Komplexität hängt von der Anzahl der Stützvektoren ab, nicht von der Dimension der Daten.

  Feature Selection nicht so wichtig, eher die Anzahl der Daten
- Auch mit wenigen Vektoren können gute Ergebnisse erzielt werden, auch im hochdimensionalen Raum.

SVMs – Nicht linear separierbare Probleme

- Trainingsdaten werden nichtlinear in einen höherdimensionalen Raum abgebildet.
- Dort wird nach linear separierender Hyperebene gesucht.
- Viele Mapping-Techniken (Kernels) verfügbar
  - Z.B.: Aus (x, y, z) wird (x, y, z, x^2, xy, xz) `ausprobieren!`
- Mit geeigneten Mapping-Techniken und hinreichend hohen Dimensionen kann meist eine separierende Hyperebene gefunden werden.
  - Theorem von Cover (1965): Die Wahrscheinlichkeit dass Klassen linear separierbar sind steigt wenn die Features nichtlinear in einen höheren Raum abgebildet werden.

SVMs zur Klassifikation - Bewertung

- Herausforderungen
  - Anwendung auf allgemeine Klassifikationsprobleme (allgemeine kategorische Zielgröße, nicht binäre): Lernen mehrerer SVMs und Zusammenführung der Ergebnisse.
  - Wahl von Kernel-Funktion und Dimensionalität.
- Vorteile
  - Oft hervorragende Ergebnisse.
  - Oft Bessere Generalisierung als neuronales Netzwerk.
- Nachteile
  - Skaliert schlecht für viele Lerndatensätze (Dimensionalität nicht problematisch).
  - Ergebnis im extrem hochdimensionalen Raum schwer zu interpretieren.
- Häufige Anwendungen:
  - Handschrifterkennung, Objekterkennung, Sprechererkennung

Weitere Klassifikationstechniken

Regelbasierte Klassifikatoren

- Klassifikation durch Regelsatz
  - Beispiel:
    1. petalwidth <= 0.6: Iris-setosa
    2. petalwidth <= 1.7 AND petallength <= 4.9: Iris-versicolor
    3. Sonst: Iris-virginica
- Übliches Vorgehen:
  - Entscheidungsbaum lernen
  - Deduktion der wichtigsten Regeln aus Baum
  - Nicht alle Tupel klassifiziert:
    - Default-Regel klassifiziert einige Tupel
    - Im Beispiel: Default-Regel: Iris-virginica
- Regelsätze oft einfacher als Entscheidungsbäume
  $\Rightarrow$ Generalisierung

Assoziationsregeln zur Klassifikation - Beispiel

- Gegeben:
  Folgende Assoziationsregeln
  - Saft -> Cola; conf: 80%
  - Cola -> Saft; conf: 100%
  - Cola -> Bier; conf: 75%
  - Bier -> Cola; conf: 100%

- Vorhersageattribut:
  - Kauft Kunde Cola?

- Beispieltupel:
  - Kunde kauft Bier
    $\Rightarrow$ Kunde kauft Cola (4. Regel)

Assoziationsregeln zur Klassifikation -Vorgehen

- Eine Regel passt:
  $\Rightarrow$ Klassifikation eindeutig (mit Konfidenz der Regel)
- Keine Regel passt:
  $\Rightarrow$ Mehrheits-Klasse bzw. unklassifiziert
- Mehrere Regeln passen:
  - Berücksichtigung der Regel mit höchster Konfidenz
    - Regel entscheidet
  - Berücksichtigung der *k* Regeln mit höchster Konfidenz (oder auch aller Regeln)
    - Häufigste auftretende Klasse
    - Klasse mit höchster durchschnittlicher Konfidenz der Regeln
  - …
- Hinweis:
  Verfahren eignet sich auch für sequentielle Regeln.

Organisatorisches zum Data-Mining-Cup

Zwischenpräsentation am 10.05.2010

- pro Gruppe 10 Minuten Vortrag, 5 Minuten Diskussion
- Status Quo beim Data-Mining-Cup:
  - Ergebnisse der Analyse der Daten
    - statistische Auffälligkeiten?
  - resultierende Vorverarbeitungsschritte
  - ausprobierte Verfahren
  - Punktzahlen von min. einem Modell
    - k-fache Kreuzvalidierung, k>=2, Partitionierung zufällig
    - Überprüfung im Tutorium `Clementine: 50% zum lernen, 50% zum testen`
  - nächste geplante Schritte

Quellen

- J. Han und M. Kamber: „Data Mining: Concepts and Techniques“, Morgan Kaufmann, 2006.
- I.H. Witten und E. Frank: "Data Mining - Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2005.
- Hand, H. Mannila und P. Smyth: "Principles of Data Mining", MIT Press, 2001.
- Vladimir N. Vapnik: “The Nature of Statistical Learning Theory”, Springer, 1995; “Statistical Learning Theory”, Wiley, 1998.
- T. M. Mitchell: „Machine Learning“, Mc Graw Hill, 1997.
- F. Klawonn: Folien zur Vorlesung „Data Mining“, 2006.
- C. Borgelt: Folien zur Vorlesung „Introduction to Neural Networks“, 2009; Folien zur Vorlesung „Intelligent Data Analysis“, 2004. Vorlesungsskript verfügbar (120 Seiten): http://fuzzy.cs.uni-magdeburg.de/studium/ida/txt/idascript.pdf
- M. Spiliopoulou: Vorlesung „Data Mining for Business Applications“, 2003.
- http://isl.ira.uka.de/neuralNetCourse/2006/Vorlesung_2006-05-09/applet-perceptron/Perceptron.html
- http://fbim.fh-regensburg.de/~saj39122/wabrpi/