
Data Warehousing und Data Mining

Eine Einführung in entscheidungsunterstützende Systeme

Interaktive Folien zu Kapitel 8
Neuronales Lernen



Das menschliche Nervensystem nachbilden?

Einordnung

Nutzwertanalyse am Beispiel von AHP

- ✓ Kioskstandort 📌, Personalauswahl

Was Wenn-Analyse

- ✓ Erfolgsrechnung 📌
- ✓ Anzeigenplanung 📌, Produktionsplanung

Regelbasierte Systeme

- ✓ Spesen 📌, Betriebskredit
- ✓ Regelverkettung

Data Warehouses

- ✓ Anlageberatung 📌
- ✓ Lieferfrist, Handel, Verkauf 📌

Data Mining - Ein Überblick

- ✓ Zeitschriften 📌, Bank

Regelinduktion

- ✓ Spesen 📌, Bonitätsklassifikation 📌

⇒ **Neuronale Netze**

- Bonitätsklassifikation 📌, Bonitätsvorhersage
- EindimPerzeptron, ZweidimPerzeptron
- MehrklassPerzeptron
- MehrstufPerzeptron

Unterrichtsmaterial

Software

- Kurzdemonstration zu SPSS *Neural Connection*
- Demonstrationsversion von *Predict* (Add In zu *MS Excel*)
- *MS Excel* mit *Visual Basic für Applikationen*

Beispiele und Übungen

- Bonitätsklassifikation mit *Predict* 📌
- Bonitätsvorhersage mit *Predict*
- EindimPerzeptron mit *MS Excel*
- ZweidimPerzeptron mit *VBA*
- MehrklassPerzeptron mit *VBA*
- MehrstufPerzeptron mit *VBA*

Produktinformation

- <http://www.neuralware.com/>
- <http://www.spss.com/>

Grundlagen

Grundlagen

⇒ Neuron	<u>7</u>
⇒ Neuronales Netz	<u>10</u>
⇒ Neuronales Lernen	<u>13</u>

Anwendung

• Anwendungsklassen	<u>34</u>
• Entwicklung mit <i>Predict</i> 📌	<u>40</u>

Einblick in die Theorie

• Transferfunktion und Schwellenwert	<u>82</u>
• Zweiklassen-Perzeptron	<u>90</u>
• Eindimensionales Perzeptron	<u>93</u>
• Zweidimensionales Perzeptron	<u>107</u>
• Mehrklassen-Perzeptron	<u>127</u>
• Neuronales Lernen mit Fehlerrückführung	<u>145</u>

Ziel

- Gehirnfunktionen *verstehen*
(naturwissenschaftliche Fragestellung)
- Gehirnfunktionen *nachbilden*
(ingenieurwissenschaftliche Fragestellung)



Im Betrieb anwenden

(betriebswirtschaftliche Fragestellung)

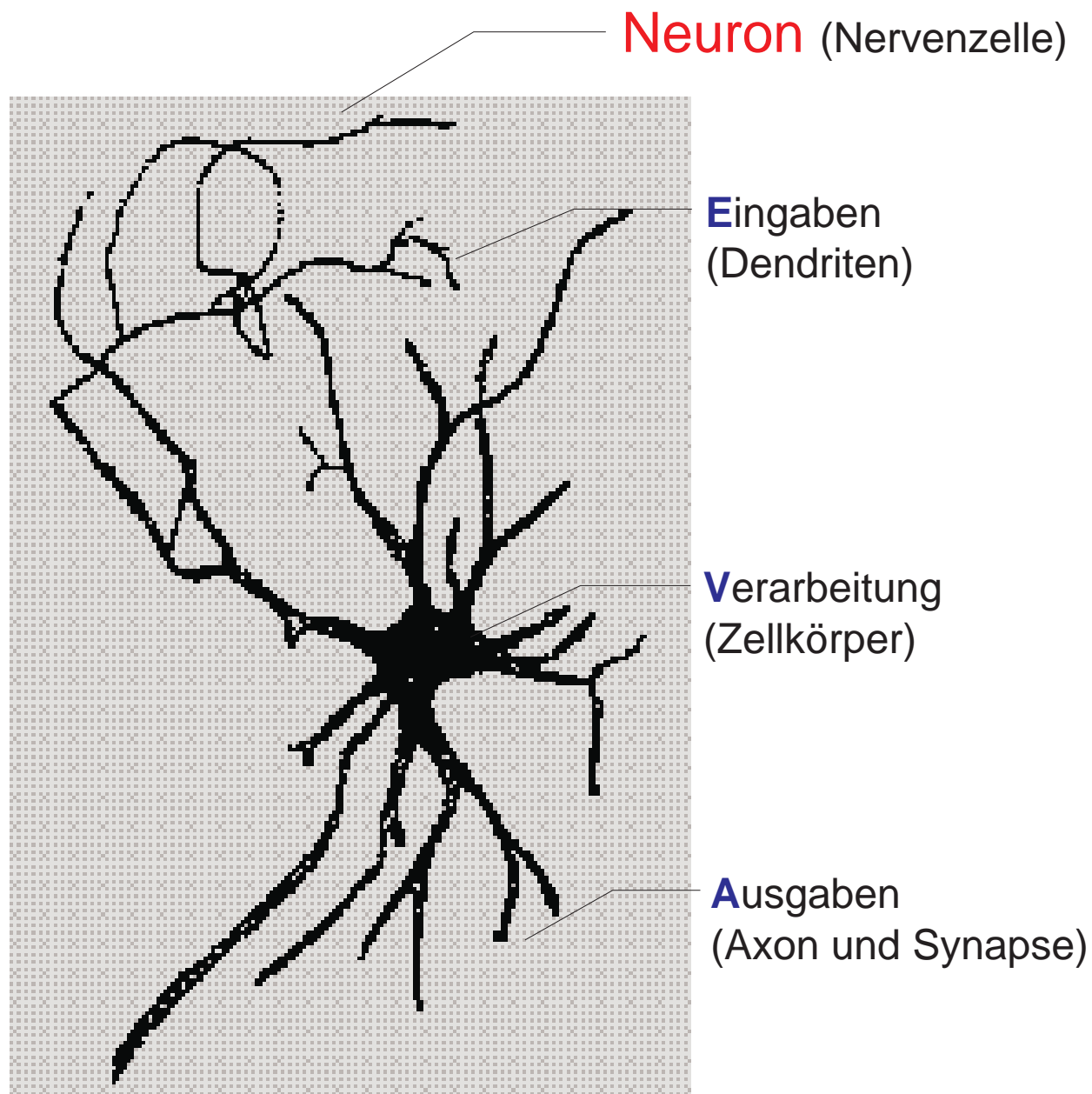
- ›OCR
- ›Direct Mailing †
- ›Bonitätsbeurteilung †
- ...



Weg

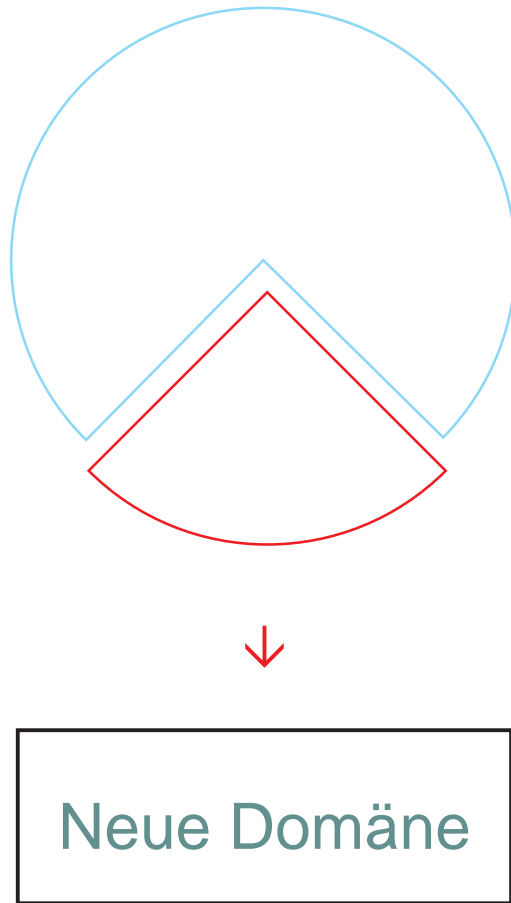
- Aus Fehlern lernen
- Das Gelernte verallgemeinern

8.2 Biologische Nervenzellen



Aspekte des **Lernens biologischer** Nervensysteme auf
das Lernen **künstlicher** neuronaler Netze verallgemeinern

Comparaison n'est pas raison

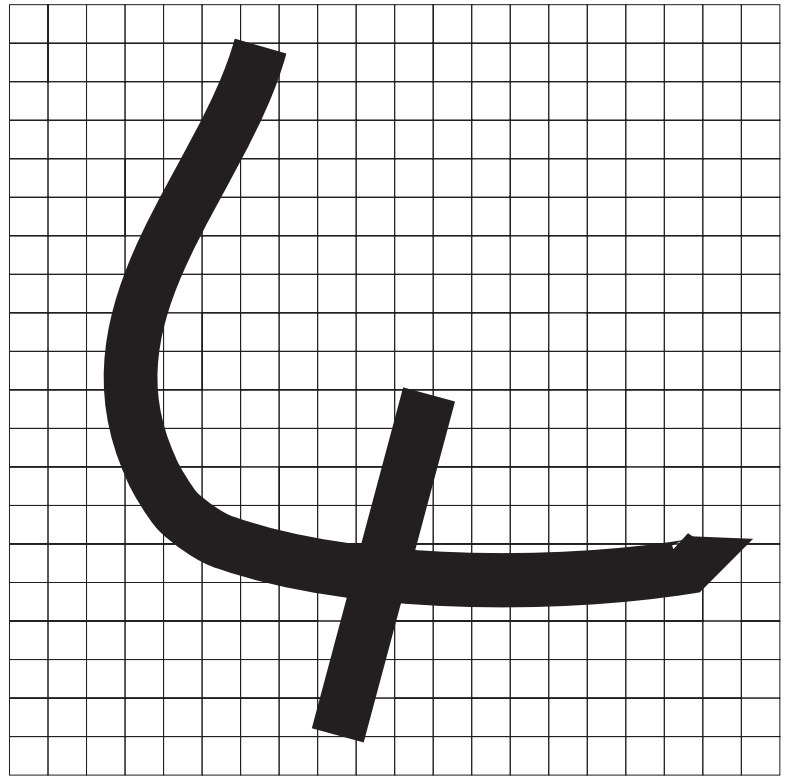


Analogien und Modelle ...

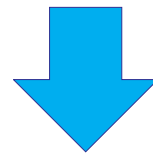
- **isolieren** einen Teil der Wirklichkeit
- **abstrahieren** von der Komplexität dieses Teils
(zum Beispiel jener des menschlichen Gehirns)
- **verallgemeinern** auf eine neue Domäne
(zum Beispiel auf ausgewählte betriebliche Entscheidungen)

8.2 Mustererkennung als neuronales Lernen

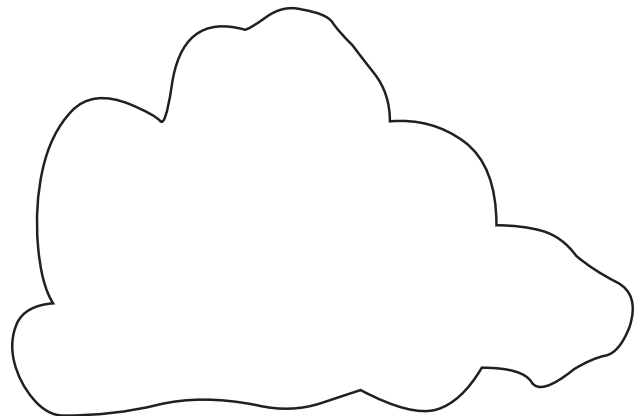
Umwelt
(20 x 20 Sensoren)



Eingabe
(Rasterbild einer Ziffer)



Neuronales Netz
(Gewichte lernen)

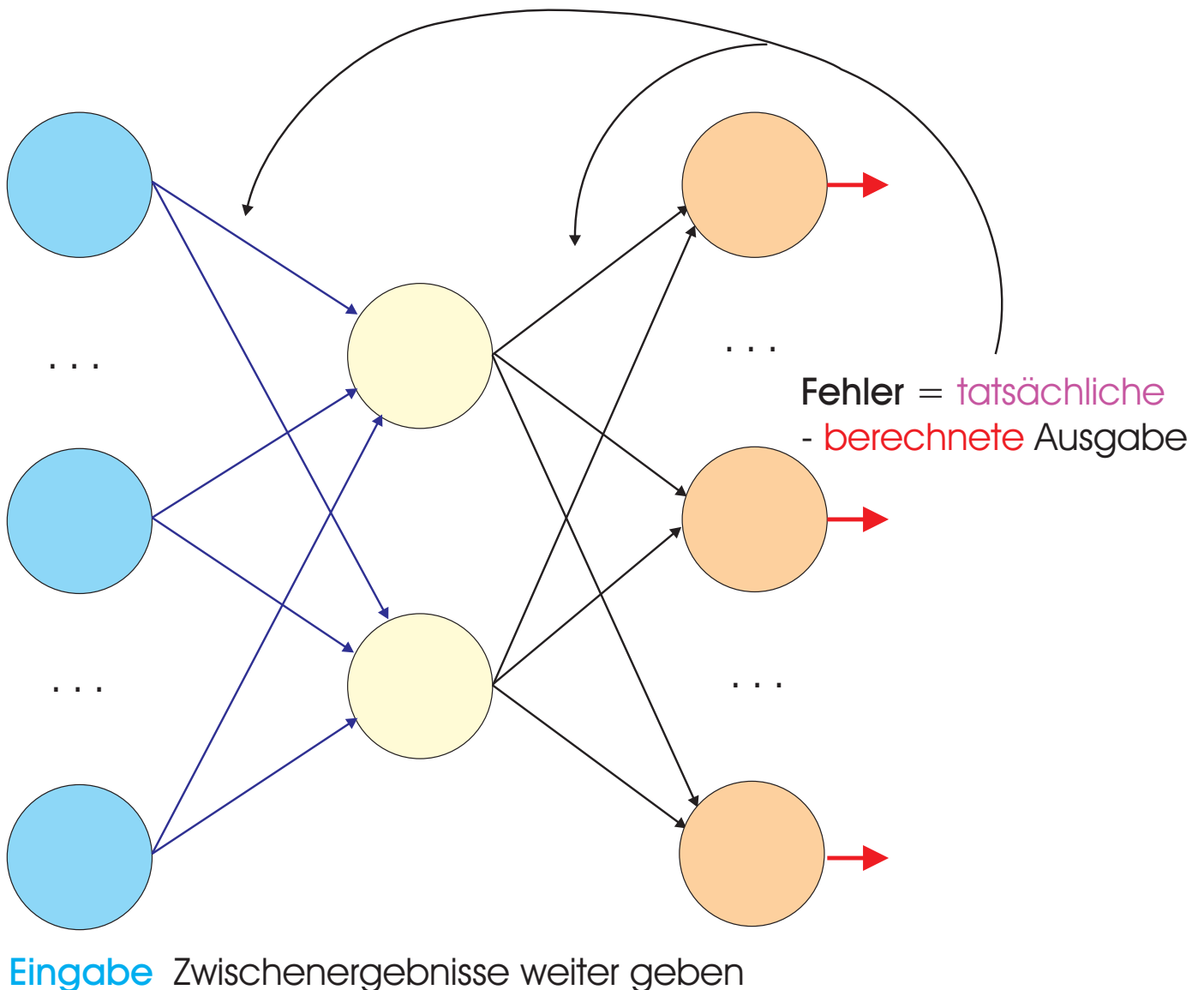


Ausgabe
(erkannte Ziffer)

0 1 2 3 4 5 6 7 8 9

8.3 Aufbau eines neuronalen Netzes

Gewichte aus beobachteten Fehlern lernen



Gegeben

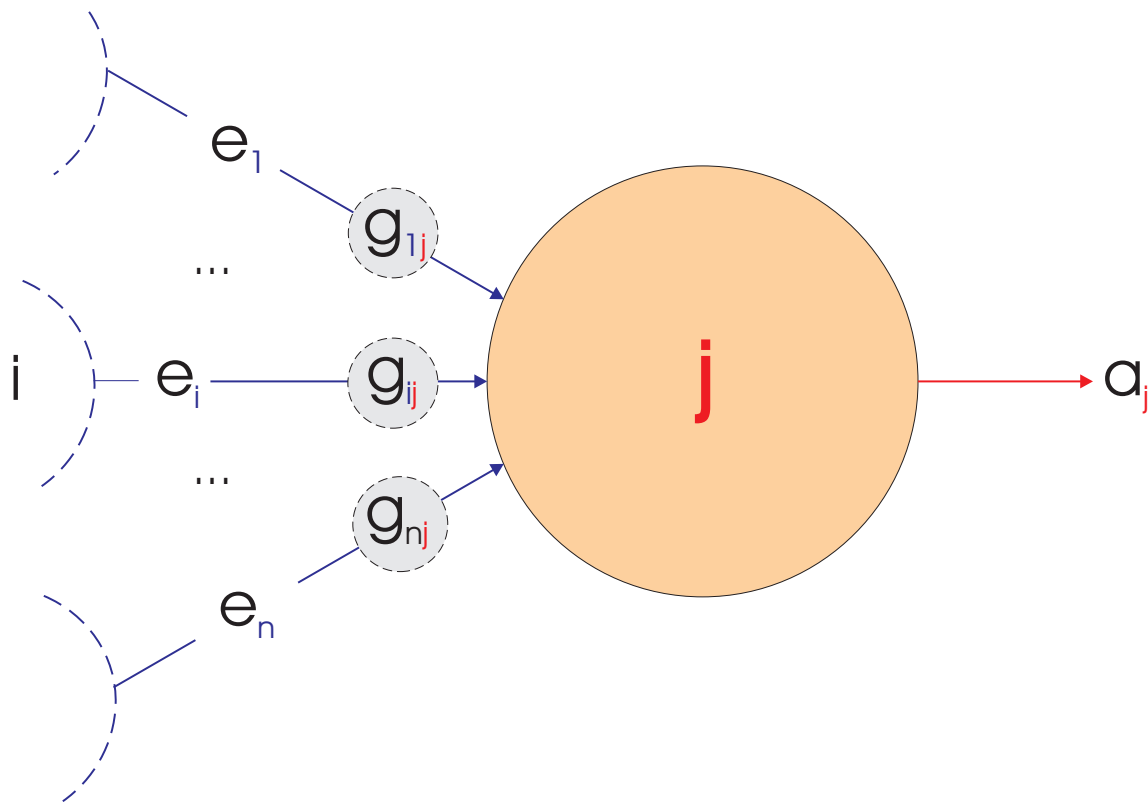
- **Eingabe** (Stichprobe von Rasterbildern der Ziffern 0 bis 9)
- **tatsächliche Ausgabe** (korrekte Interpretationen aller Ziffern)

Gesucht

Algorithmus, der für eine beliebige Eingabe die korrekte Ausgabe **berechnet** (der das Rasterbild einer beliebigen Ziffer interpretiert)

○ **Netzelement** (Neuron, Nervenzelle)

8.4 Künstliches Neuron



Eingaben

Verarbeitung

Ausgabe

EVA: Ein **Neuron** verarbeitet *mehrere* **E**ingaben über **V**erarbeitungsgewichte zu einer *einzig*en **A**usgabe

Eingaben e_i

→

Verbindungsgewicht g_{ij}

Ausgabe a_j

von Neuron i oder der Umwelt
gerichtete und gewichtete Verbindung
zwischen Neuron i und Neuron j
des Neurons j ¹

¹ Ausgabe eines einzelnen Neurons ungleich Ausgabe des ganzen Netzes !

Künstliches neuronale Netz

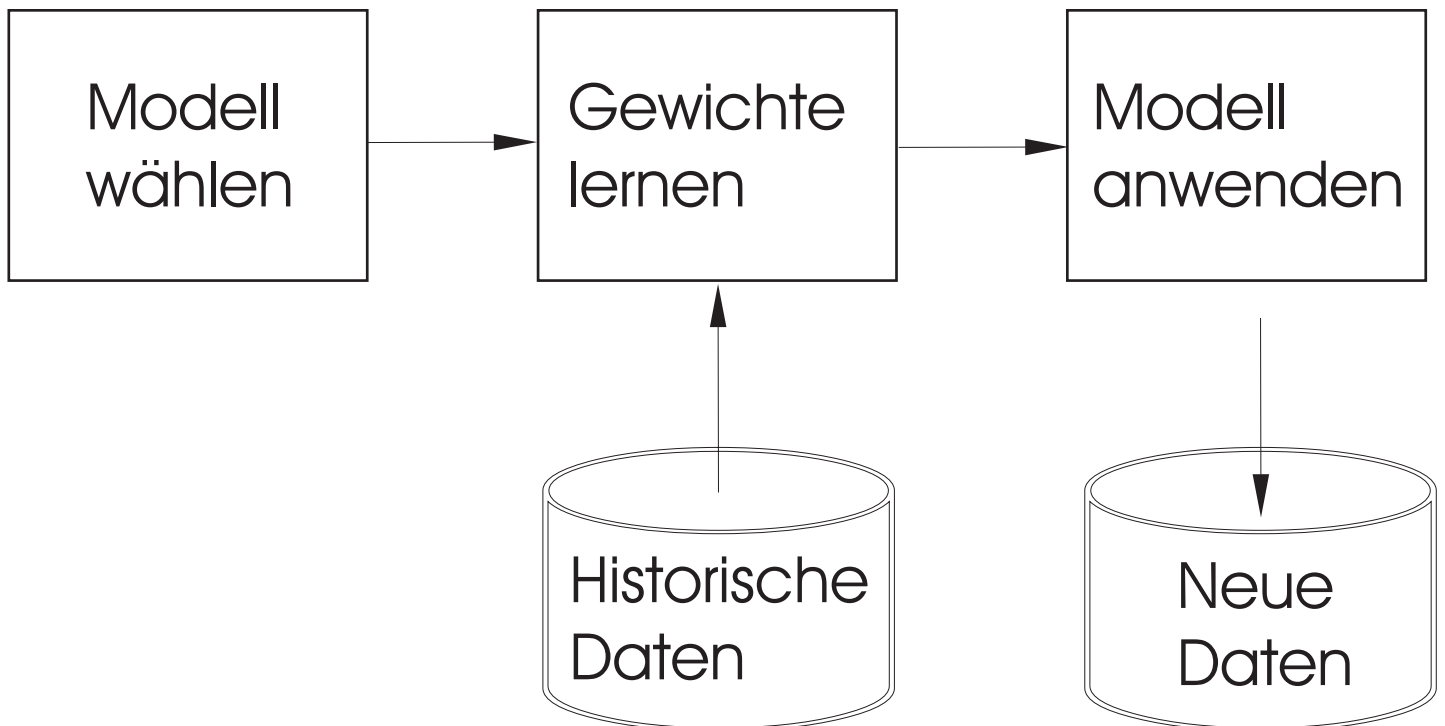
Neuron :=

- ✓ **lernfähige** Verarbeitungseinheit, die ...
- ✓ **Eingaben** aus Neuronen
- ✓ **verarbeitet** und
- ✓ an andere Neuronen **ausgibt**

Neuronales Netz :=

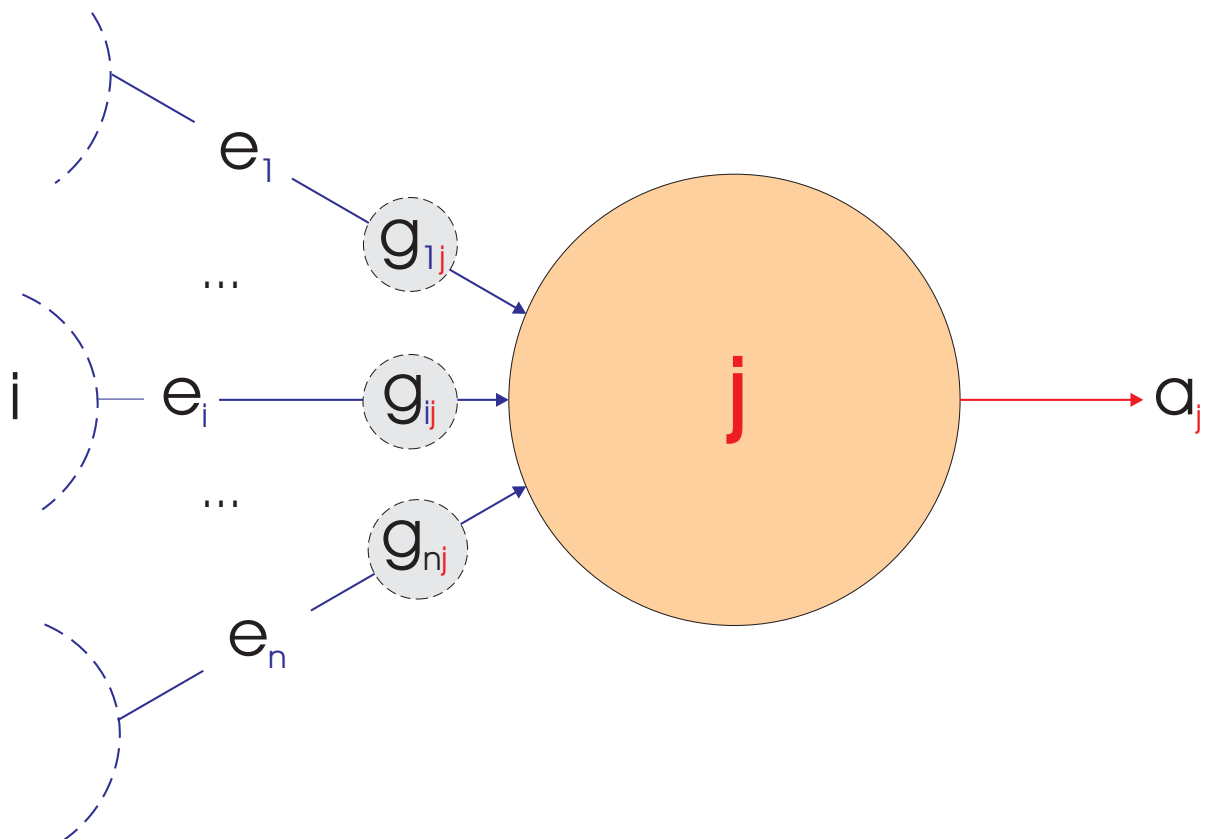
- ✓ Menge von **Neuronen**
(von denen jedes aus Eingaben eine einzige Ausgabe berechnet)
- ✓ mit einfacher **Architektur**
(statt der Millionen und Milliarden biologischer Nervenzellen)
- ✓ **wenigen zahlenübertragenden Verbindungen**
(statt der bis zu 100'000 biologischen Verbindungen)

8.5 Methodischer Ansatz neuronaler Verfahren



Neuronale Verfahren lernen an **historischen** Daten und wenden das Gelernte auf **neue** Daten an. Die Lernmethoden orientieren sich grob an den Lernprinzipien biologischer Nervensysteme

Neuronales Lernen



Wie kann ich die Gewichte g_{ij} so berechnen, dass die Ausgabe a_j herauskommt ?



Lernen heisst Gewichte anpassen

Gewichte so lange anpassen, bis ein neuronales Netz handschriftliche Zeichen akzeptabel klassifiziert

① Überwachtes Lernen

Ein “Lehrer” ordnet jedem Eingabentupel die Lösung zu

- ✓ Lernmenge aus **Eingaben** und korrekten **Ausgaben** (Musterlösungen)
- ✓ Analoge Verfahren aus der Regelinduktion
- ✓ Architekturen
 - einstufige und mehrstufige Perzeptrons
 - ›CCN
 - ...

② Unüberwachtes Lernen

Ein “Schüler” lernt allein aus der Beobachtung der Eingaben

- ✓ Lernmenge aus **Eingaben** ohne explizite Rückmeldung
- ✓ Analoge Verfahren aus der ›Clusteranalyse
- ✓ Architekturen
 - ›Kohonen-Netzwerke
 - ...

Direktwerbung

Werbemassnahmen, die den Empfänger mit einem **selbständigen** Werbemittel **direkt** (ohne vermittelnde Partner) ansprechen, zum Beispiel mit ...

- Werbebrief
- Prospekt oder Katalog
- Reaktionskarte

Direct Mailing

Form der Direktwerbung, die auf der Basis eigener oder gekaufter **Adressen** eine Zielgruppe auswählt und deren Mitglieder **einzel**n anschreibt



Antwortrate einer Direct Mailing-Kampagne
durch neuronales Lernen maximieren

📌 Ziel ist die Verbesserung der Antwortrate

Gegeben

- Merkmale von Individuen
- mit bekanntem Antwort- und Kaufverhalten

Gesucht

Verfahren, das ...

- eine Zielgruppe auswählt
- deren Antwort- und Kaufverhalten besser ist
- als jenes der übrigen Individuen

📌 Prädiktoren und Kriterien

Vorgehen

Prädiktoren (unabhängige Variablen)

a) Persönliche Merkmale wie ...

- Alter
- Geschlecht
- Zivilstand

b) Finanzielle Merkmale wie ...

- minimaler und maximaler Kontenstand
- Kreditkarte(n)

Kriterium (abhängige Variable)

Antwort- und Kaufverhalten

Werkzeugbeispiel

SPSS *Neural Connection* kombiniert ▶RBF-Netzwerke, die persönliche und finanzielle Prädiktoren getrennt ermitteln, zu einem dritten Netzwerk

Demonstration mit SPSS Neural Connection

(*Contents - Demonstrations - Direct Mailing* (Doppelklick))

Ein erster Werkzeugkontakt

SPSS Neural Connection

Funktionsumfang

a) Arten neuronaler Netze

- Mehrstufiges ›Perzeptron mit Fehlerrückführung
- ›RBF-Netz
- ›Kohonen-Netz

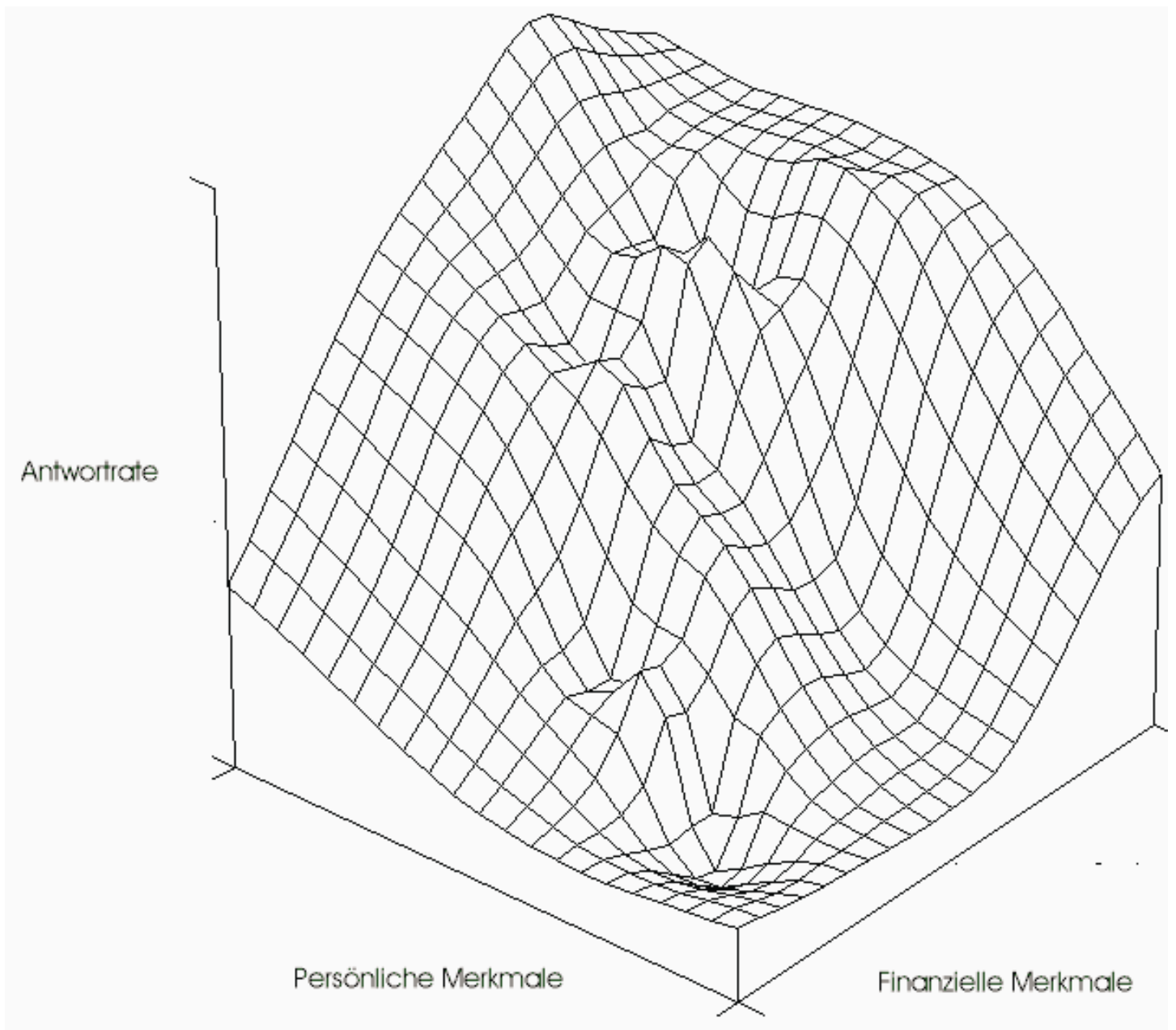
b) Statistische Verfahren

- ›Mehrfachregression
- ›Clusteranalyse

, Benutzeroberfläche

Planung des Data Mining-Prozesses in einem Graph aus Icons

📌 Ein reduziertes Ergebnis



\$ UND - ① Problem

⇒ ① Problem

- ② Datenaufbereitung
- ③ Modell

Wahrheitstabelle der UND-Verknüpfung

<i>Aussage e_1</i>		<i>Aussage e_2</i>	<i>Aussage a_3</i>	
wahr	UND	wahr	⇒	wahr
falsch	UND	falsch	⇒	falsch
wahr	UND	falsch	⇒	falsch
falsch	UND	wahr	⇒	falsch



Wie lerne ich die Gewichte einer Funktion,
die aus den **Wahrheitswerten von zwei**
Aussagen den **Wahrheitswert der UND-**
Verknüpfung berechnet ?

Datenaufbereitung

- ✓ ① Problem
- ⇒ ② Datenaufbereitung
- ③ Modell

Wahr codieren wir als **1** und *falsch* als 0



<i>Aussage e_1</i>		<i>Aussage e_2</i>	<i>Aussage a_3</i>	
1	UND	1	⇒	1
0	UND	0		0
1	UND	0		0
0	UND	1		0



Ausgaben neuronaler Netze liegen oft zwischen 0 und 1 (inklusive)

Modell

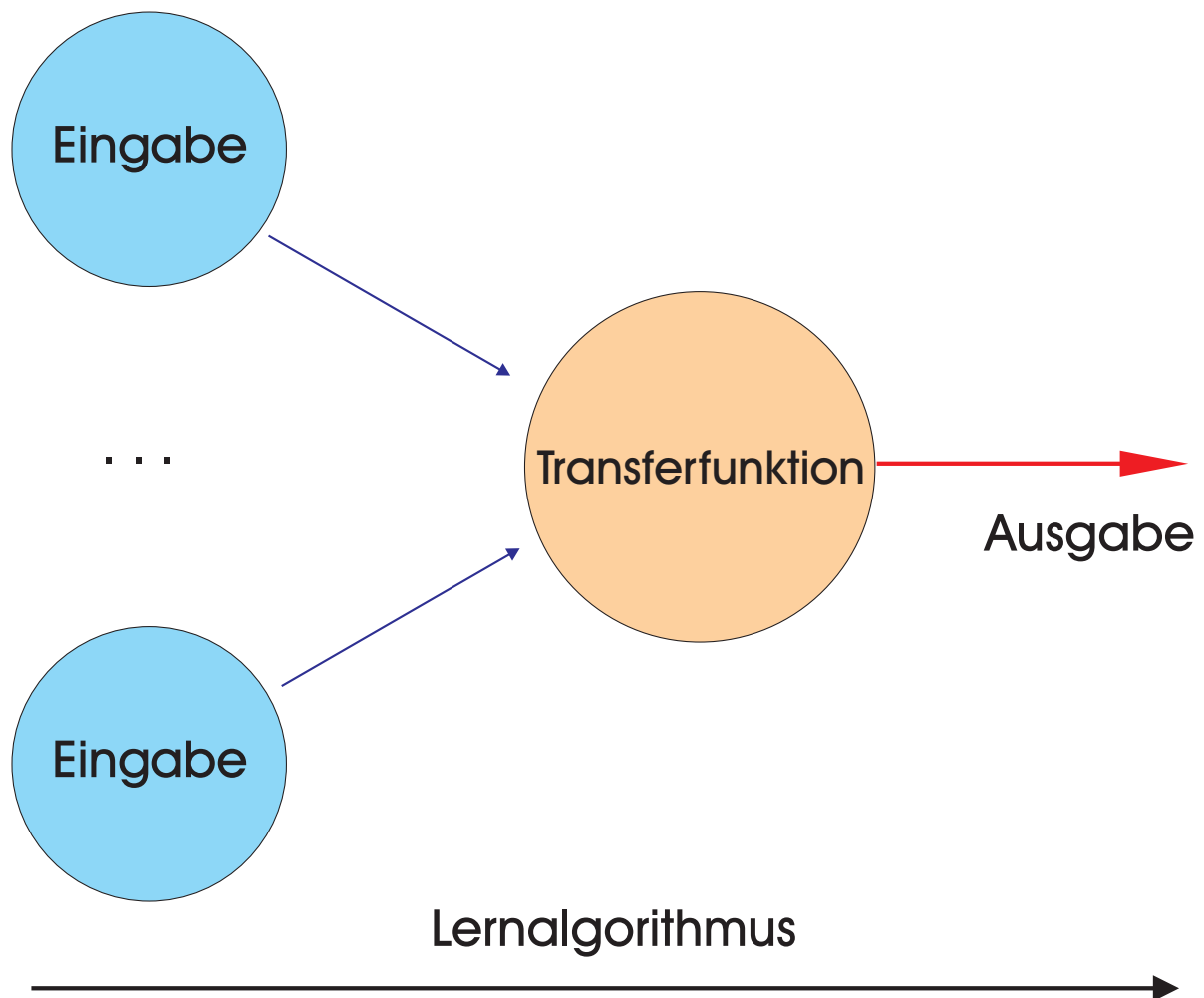
- ✓ ① Problem
- ✓ ② Datenaufbereitung
- ⇒ ③ Modell

<i>Aussage e_1</i>		<i>Aussage e_2</i>	<i>Aussage a_3</i>	
1	UND	1	⇒	1



Wie lehre ich einem **neuronalen Netz** die UND-Verknüpfung zweier Aussagen?

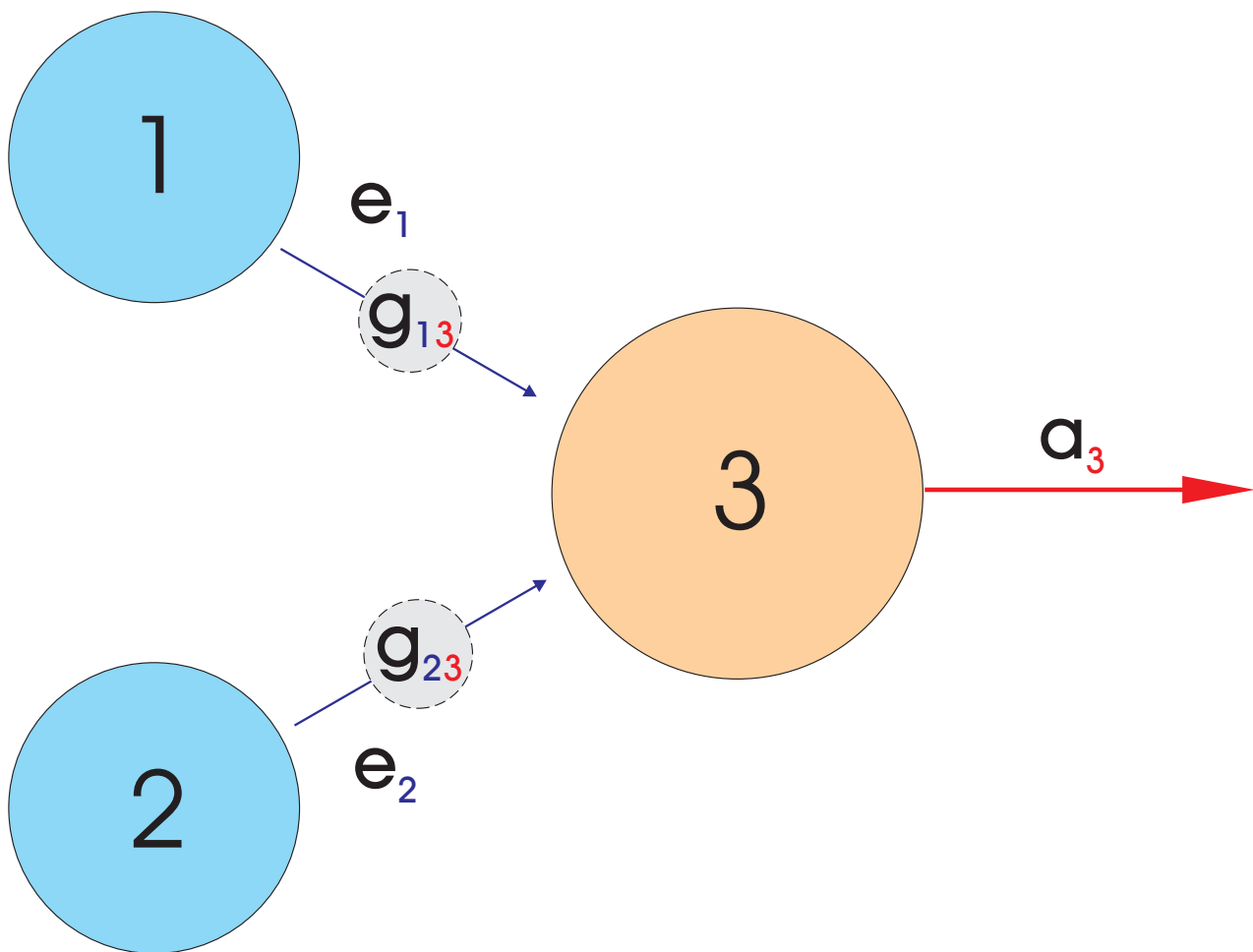
8.6 Modell eines einfachen neuronalen Netzes



Modell aus Architektur, Transferfunktion und Lernalgorithmus

Architektur	definiert Eingabe -, Ausgabe - und Verarbeitungsneuronen und ihre Beziehungen (siehe Bild)
Transferfunktion f	leitet die Eingaben an andere Neuronen und die Umwelt weiter
Lernalgorithmus	berechnet die Gewichte von f

8.7 1) Modellkomponente **Architektur**



✓ ① Problem

✓ ② Datenaufbereitung

⇒ ③ Modell

⇒ Architektur

Form der Transferfunktion f

Lernalgorithmus

Modellkomponente **Transferfunktion f**

e_1	e_2	Tatsächliche Ausgabe a_3
1	1	1
0	0	0
1	0	0
0	1	0

Welche g_1 und g_2 bilden die e_i auf das a_3 ab, wenn f **multiplikativ** ?



$$a_3 = g_{13} e_1 \cdot g_{23} e_2$$

Ein *mögliches* Ergebnis des Lernalgorithmus ist $g_{13} = 2$, $g_{23} = 1/2$



$g_{13} e_1$	\cdot	$g_{23} e_2$	Berechnete Ausgabe
2 · 1	·	1/2 · 1	= 1
2 · 0	·	1/2 · 0	= 0
2 · 1	·	1/2 · 0	= 0
2 · 0	·	1/2 · 1	= 0

Suchen Sie die Regel, der die unendlich vielen Lösungen gehorchen

Transferfunktionen sind häufiger *additiv* als multiplikativ

\$ Modellkomponente **Lernalgorithmus**

- ✓ ① Problem
- ✓ ② Datenaufbereitung
- ⇒ ③ Modell

Architektur

Form der Transferfunktion f

⇒ Lernalgorithmus

Suche einen Lernalgorithmus, der aus den **Eingaben** und den **tatsächlichen** Ausgaben die Gewichte der der gegebenen Transferfunktionsform **berechnet**



Initialisiere g_{13} und g_{23} beliebig

BIS **berechnete** Ausgaben = **tatsächliche** Ausgaben

FALLS $g_{13} e_1 \cdot g_{23} e_2 < a_3$

Vergrößere g_{13} oder g_{23} um 0.1

SONST

Verkleinere g_{13} oder g_{23} um 0.1



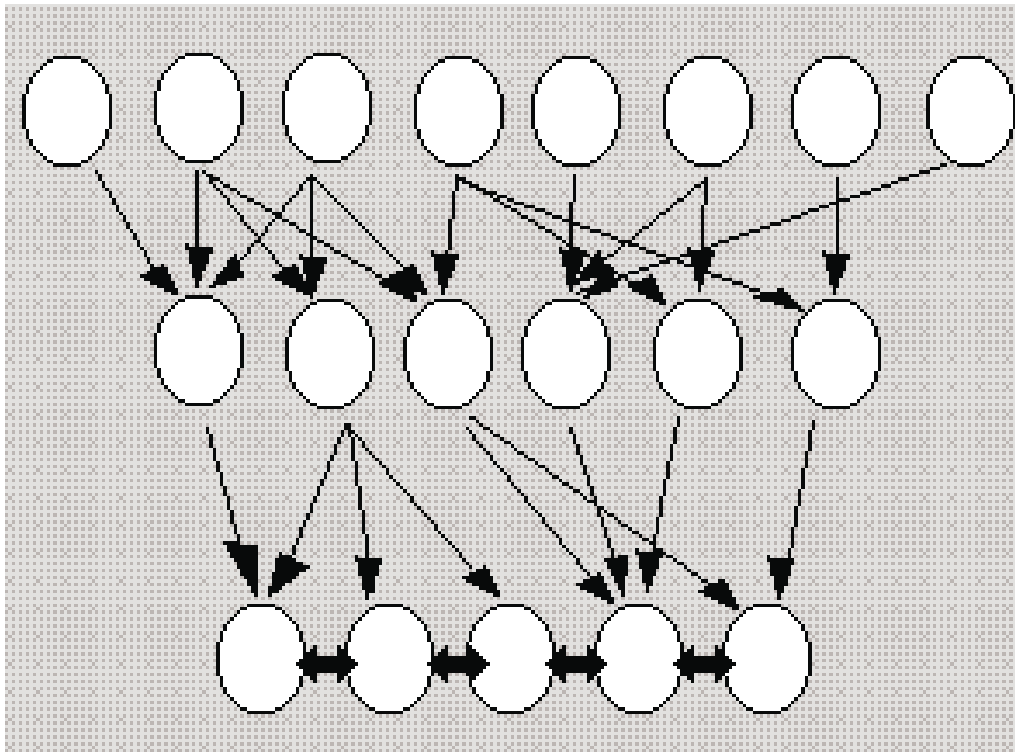
Ergebnis: Eine mögliche Transferfunktion ist $a_3 = 2 e_1 \cdot 1/2 e_2$

Lernalgorithmen berechnen die Gewichte der Transferfunktion in der Regel **iterativ** und sind deshalb **rechenintensiv**

Forschungsfragen

- **Allgemeinerer** Lernalgorithmus, der beliebige Eingaben auf beliebige Ausgaben abbildet ?
⇒ Bsp. Passt der Lernalgorithmus auch auf ODER?
- **Architekturen** grösserer neuronaler Netze?
⇒ Zahl und Anordnung der Neuronen
- Neuronale Netze für **Praxis**probleme?
⇒ z.B. die Schriftenerkennung oder Börsenvorhersage
- ...

Praxis - Dutzende bis Tausende von Neuronen



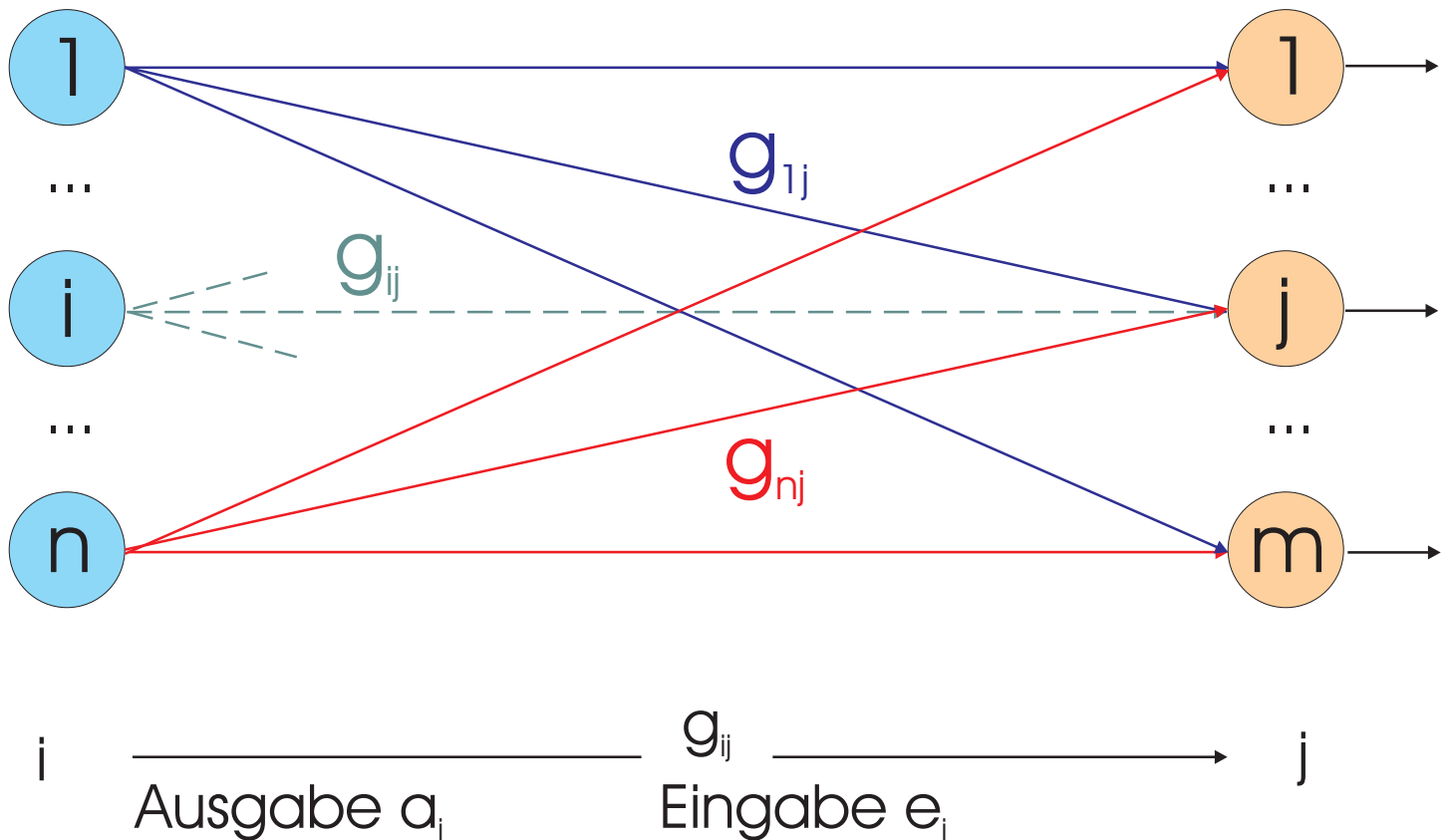
mit charakteristischen **Modellen** aus ...

Architektur
Transferfunktion(en)
Lernalgorithmus



Dutzende neuronaler Modelle!

8.8 Einstufige Architektur

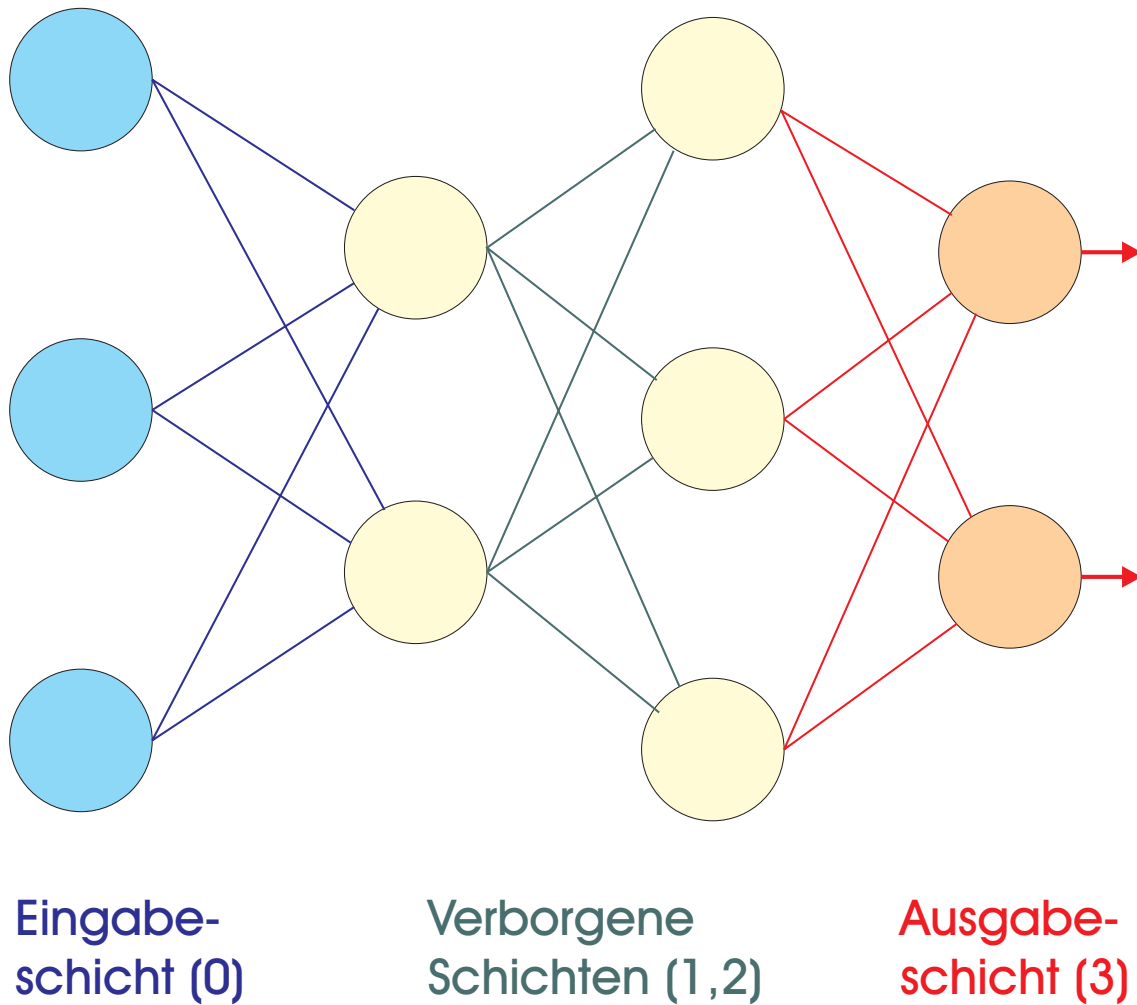


Einstufiges neuronales Netz := neuronales Netz aus **Eingabe**- und **Ausgabe**neuronen, die nur über *eine* Gewichtungsstufe verbunden sind



Bsp. *einstufiges Perzeptron* →

Mehrstufige Architektur



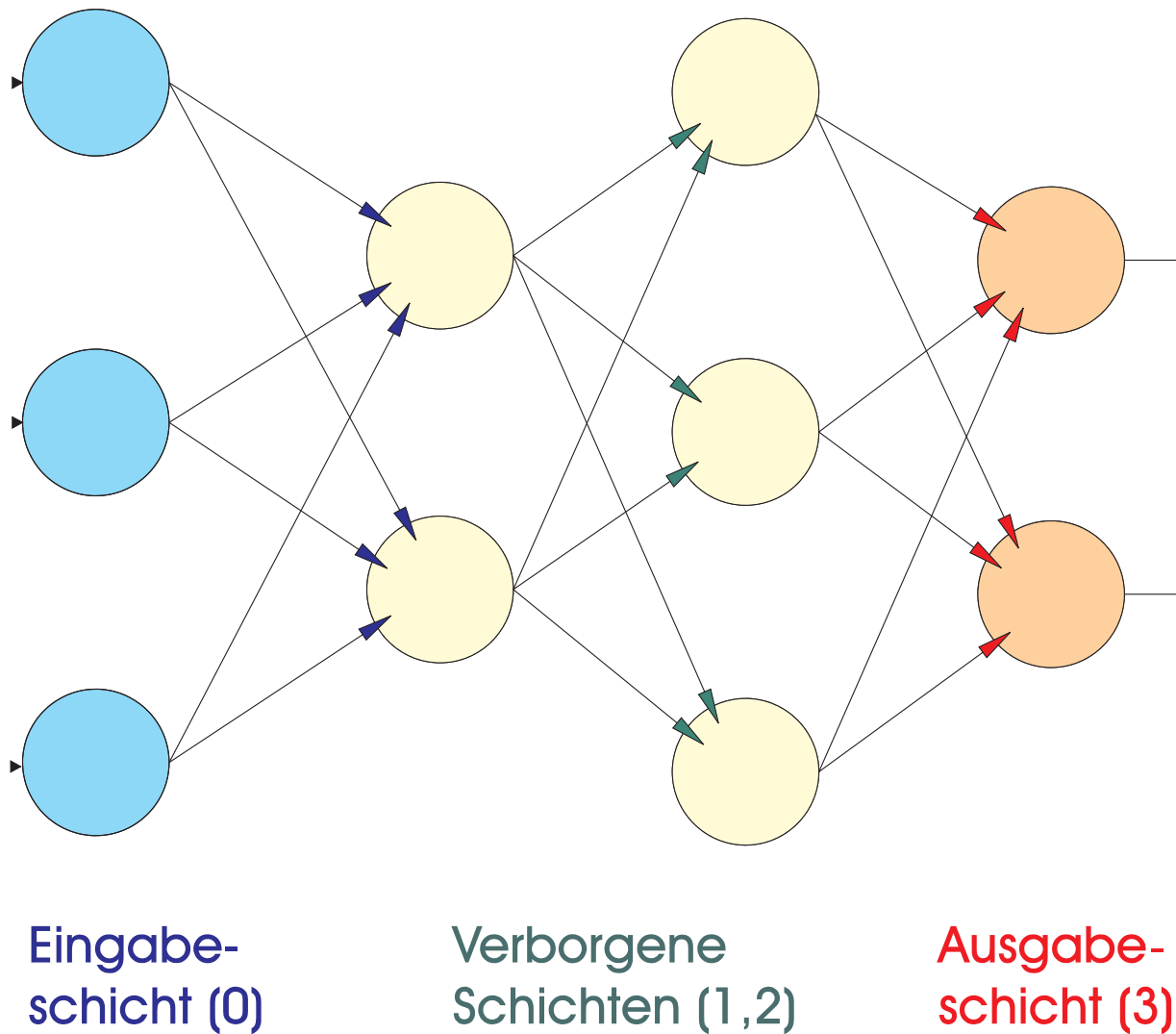
Mehrstufiges neuronales Netz := neuronales Netz aus Eingabe-, verborgenen - und Ausgabeneuronen, die durch *mehr als eine* Gewichtungsstufe verbunden sind

Mehrstufige neuronale Netze sind weit verbreitet



Bsp. *mehrstufiges vorwärtsgerichtetes Netz* →

8.9 3stufige vorwärtsgerichtete 3-2-3-2-Architektur



- **Verbindungsrichtung**
vorwärts (Eingabe → Ausgabe, engl. feedforward): Rekursive, seitliche und Rückwärtsbeziehungen sind nicht erlaubt
- **Verbindungspartner**
sind nur Neuronen *benachbarter* Schichten
- **Neuronenzahl**
 $10 = 3 + 2 + 3 + 2$
- **Schichten**
4 (3-2-3-2)

Bsp. ▶CCN-Netze (Cascade Correlation Neural Networks) →

Zusammenfassung

Neuron :=

- ✓ Grundelement eines neuronalen Netzes, das ...
- ✓ mehrere **Eingaben**
- ✓ mit einer Transferfunktion
auch Aktivierungsfunktion
- ✓ zu einer **Ausgabe** verarbeitet
auch Aktivierung

Neuronales Netz :=

Menge von Neuronen mit ...

- ✓ **1** Architektur
Zahl und Verbindung der Neuronen
- ✓ Transferfunktion(en)
Vorschrift, die einer Zusammenfassung gewichteter **Eingaben** eine einzige **Ausgabe** zuordnet
- ✓ Lernalgorithmus
Verfahren, das aus einer Lernmenge repräsentativer Fälle die Gewichte der Transferfunktion(en) berechnet

Anwendung

Grundlagen

✓ Neuron	<u>7</u>
✓ Neuronales Netz	<u>10</u>
✓ Neuronales Lernen	<u>13</u>

Anwendung

⇒ Anwendungsklassen	<u>34</u>
⇒ Entwicklung mit <i>Predict</i> 📌	<u>40</u>

Einblick in die Theorie

• Transferfunktion und Schwellenwert	<u>82</u>
• Zweiklassen-Perzeptron	<u>90</u>
• Eindimensionales Perzeptron	<u>93</u>
• Zweidimensionales Perzeptron	<u>107</u>
• Mehrklassen-Perzeptron	<u>127</u>
• Neuronales Lernen mit Fehlerrückführung	<u>145</u>

8.10 Wichtige Anwendungsklassen

Bereich	Eingaben → Ausgaben	Modelle	Beispiel
Klassifikation	Individuen bekannten Klassen zuordnen	▸CCN ▸Perzeptron u.a.	▸OCR
Clustering und Mustererkennung	Individuen unbekannten Gruppen zuordnen	CCN u.a.	▸Marktkorb-analyse
Vorhersage	Kontinuierliche Vorhersagewerte aus bekannten Faktoren berechnen	CCN u.a.	▸Bonitäts-beurteilung
Optimierung	Optimale Lösungen unter Nebenbedingungen suchen	Rekursive Netze u.a.	▸ Was-Wenn-Analyse



Klassifikation und Vorhersage an
am Fallbeispiel *Bonitätsbeurteilung*

Anwendungsklasse Vorhersage

Anwendungs- beispiel	Ausgabebeispiel	Eingabebeispiele
Kredit	Zahl der bezahlten Raten	Ausgaben/Einkommen, Zivilstand
Portfolio	Rangfolge von Wertschriften	Kursentwicklung, Branchenzugehörigkeit
Direct Mail	Antwortwahrscheinlich- keiten möglicher Ziel- gruppen einer Direct Mail-Kampagne	Einkommen, Beschäftigung

Typische Anwendungsumgebung

- Grosse Lernmenge
- Ungenaue Beziehungen zwischen Eingaben und Ausgaben



Bonitätsbeurteilung

Physische Modelle

Entscheidende Faktoren messbar

Bsp. ›OCR



Hohe Vorhersagegüte

Verhaltensmodelle

- Menschliches *Verhalten* involviert
- *Viele* Faktoren nicht messbar

Bsp. ›Bonitätsbeurteilung →



Niedrige Vorhersagegüte

📌 Betriebliche Anwendungen

- ✓ Bonitätsbeurteilung →
- ✓ Identifikation betrügerischer Kreditkarten-Transaktionen
- ✓ Vorhersage konkursiter Kreditkarten-Inhaber
- ✓ Vorhersage von Wechselkursschwankungen
- ✓ Planung einer Direct Mail-Kampagne
- ✓ ...

Ein betriebliches Anwendungsproblem

Fall Schneider führte zu Milliarden-Kreditflops



Prüfung der Kreditwürdigkeit (Bonitätsbeurteilung)!



Automatisierung der Bonitätsbeurteilung?

📌 Ein einfaches Problemlösungsmodell

Lernmenge historischer Daten

- ✓ **Ausgabe** (abhängige Variable)
 - *kreditwürdig*
 - *nicht kreditwürdig*
- ✓ **Eingaben** (unabhängige Variablen)
 - *Einkommen*
 -¹



Wie **Ausgaben** aus **Eingaben** vorhersagen?



computergeeigneter Lernalgorithmus?



Werkzeug?

¹ Suchen Sie nach sinnvollen Eingaben

Der Endbenutzer ...

- **bearbeitet** Daten in *MS Excel*
- **verarbeitet** sie dann in *Predict*
- **präsentiert** Ergebnisse in *MS Excel*

Der Programmierer ...

- **bearbeitet** Daten im *eigenen Programm*
- **verarbeitet** sie dann mit der *Predict-DLL*
(oder lässt Quellcode generieren)
- **präsentiert** Ergebnisse im *eigenen Programm*

8.11 📌 BONITÄTSKLASSIFIKATION - **Fallbeispiel**

Problem

Aus einer Stichprobe ehemaliger Konsumkreditnehmer ein neuronales Modell zur Prüfung der Bonität (Kreditwürdigkeit) künftiger Antragsteller erstellen

Gegeben

Stichprobe aus 400 Datensätzen, die abgeschlossene Kreditgesuche der letzten zwei Jahre beschreiben.

Gesucht

Parameter eines neuronalen Klassifikationsmodells, das jedes künftige Kreditgesuch einer der Klassen "Annahme" und "Ablehnung" zuordnet

Entwicklung

- Stichprobendaten in *MS Excel* sammeln
- Abhängige und Unabhängige unterscheiden
- Neuronales Netz mit *Predict* erstellen

[Bonitätsklassifikation.xls](#)

8.12 📌 Lernen und Anwenden

	<i>Lernmenge</i>	<i>Anwendungsmenge</i>
<i>Datenge- samtheit</i>	Kreditnehmer der <i>letzten 2 Jahre</i>	Konsumkreditanträge der <i>nächsten Jahre</i>
<i>Eingaben</i>	bonitätsrelevante Ei- genschaften <i>ehemali- ger</i> Kreditnehmer	bonitätsrelevante Eigen- schaften <i>neuer</i> Kreditnehmer
<i>Ausgabe</i>	tatsächliche Bonität	berechnete Bonität
<i>Verarbeitung</i>	Gewichte der Verbin- dungen zwischen Eingaben und Ausga- be lernen	Ausgabe aus <i>neuen</i> Eingaben und <i>gelernten</i> Gewichten berechnen

Entwicklungsphasen

Problem **spezifizieren**


Bonität von Antragstellern für einen Konsumkredit beurteilen

② Lern- und Validierungsdaten **sammeln**

Stichprobe der im letzten Jahr beendeten Konsumkreditgeschäfte in einem Tabellenkalkulationsblatt darstellen

③ Daten **aufbereiten**

Falsche und redundante Daten entfernen

*Ausgewählte Rohdaten um die Teuerung bereinigen
(vgl. Datentransformation in  [Data Warehouses](#))*

④ **Vorhersagevariablen** wählen

*Unabhängige Variablen wie Wohneigentum und Einkommen
manuell oder automatisch wählen*

Neuronales Modell spezifizieren und lernen

*Parameter des Vorhersagemodells von Predict eingeben
und Modell lernen lassen*

Modell validieren

Validität des gelernten Netzes an unabhängigen Daten prüfen

Modell anwenden

Validiertes Netz auf neue Kreditgesuche anwenden



Problem spezifizieren

Modell aus der **Lernmenge** berechnen

Lernmenge

Kreditnehmer der *letzten* 2 Jahre

Eingaben

Bonitätsrelevante Eigenschaften der Kreditnehmer

Tatsächliche Ausgabe

Bonität der Kreditnehmer

Verarbeitung

Gewichte der Verbindungen zwischen Eingaben und Ausgabe lernen

Modell auf eine **Anwendungsmenge** übertragen

Anwendungsmenge

Konsumkreditanträge der *nächsten* 2 Jahre

Eingaben

Bonitätsrelevante Eigenschaften *neuer* Kreditnehmer

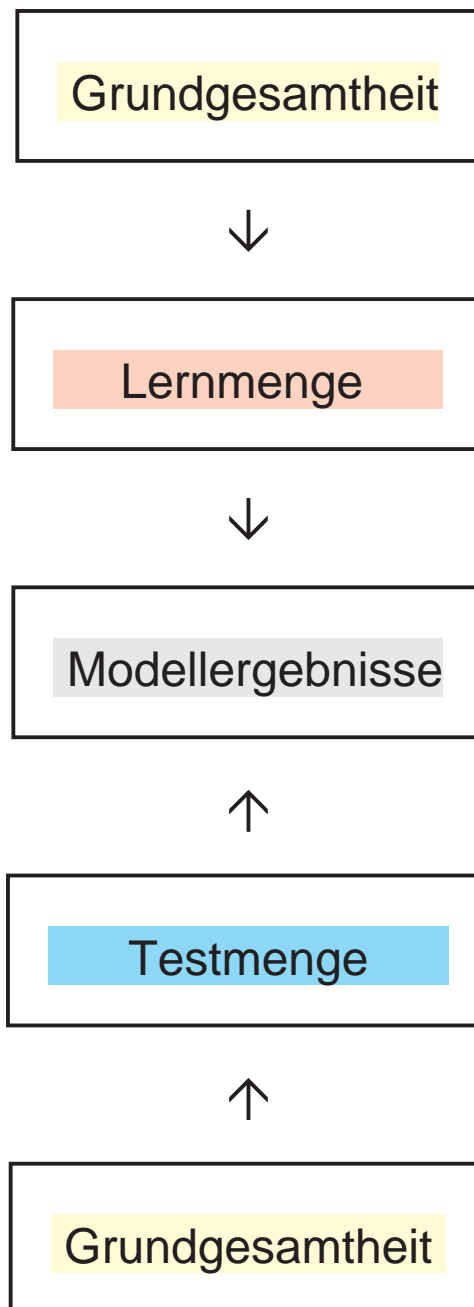
Berechnete Ausgabe

Bonität *neuer* Kreditnehmer

Verarbeitung

Ausgabe aus neuen Eingaben und gelernten Gewichten berechnen

, Lern- und Testdaten sammeln



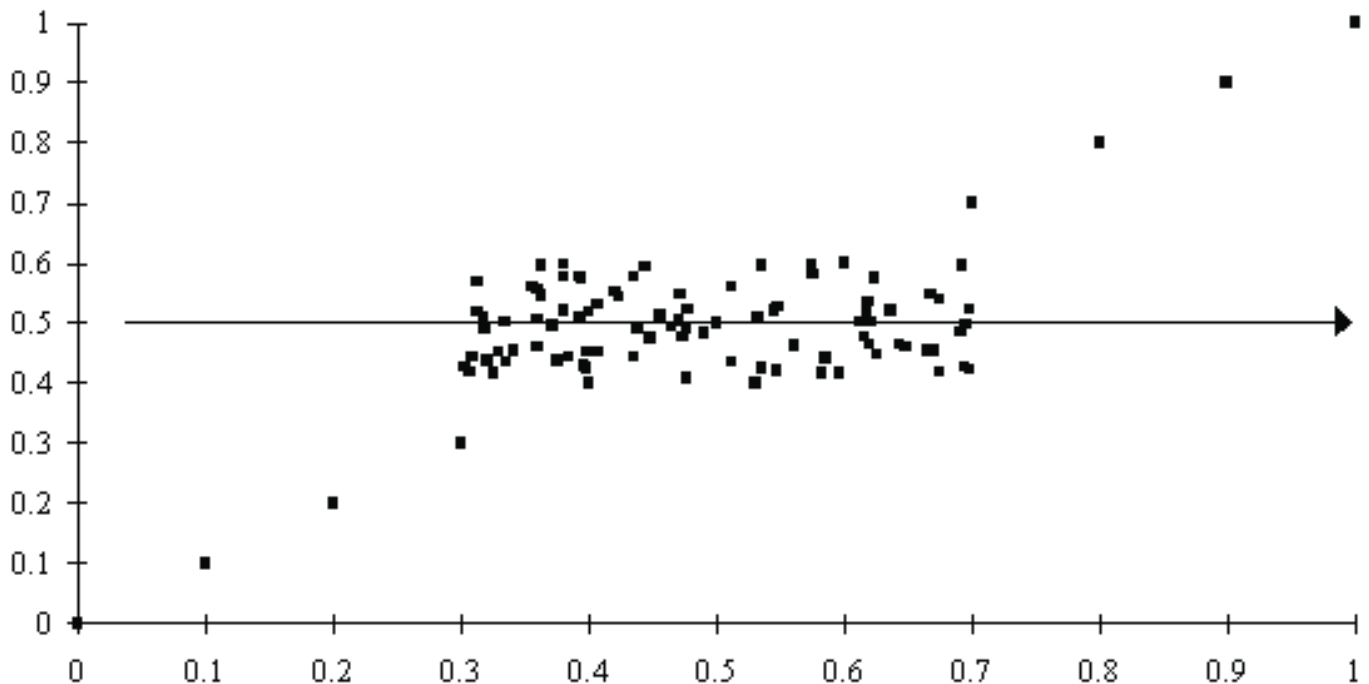
Lernmenge := Stichprobe, aus welcher die Gewichte des Modells berechnet werden

Testmenge := Stichprobe, die beurteilen hilft, ob sich das Modell verallgemeinern lässt



Viele Lern- und Testmengen sind möglich!

8.13 Daten sammeln - Repräsentativität



Ist die Punktwolke **in der Mitte** oder **entlang der Diagonalen** repräsentativ ?



Die **Repräsentativität** der Lernmenge beeinflusst die Validität des Netzes



Zufallsstichprobe?

Beurteilung grosser Stichproben

- ✓ Komplexere Architekturen möglich
- ✓ Mehr Erhebungsfehler tolerierbar
- ✓ Genauere Ergebnisse möglich
- ✗ ▶ Overfitting wahrscheinlicher →
- ✗ Rechenanforderungen grösser



- ✓ Problem spezifizieren
- ✓ ② Lern- und Validierungsdaten sammeln
- ⇒ **Daten aufbereiten**
- ④ Vorhersagevariablen wählen
- Netzmodell spezifizieren und lernen
- Modell validieren
- Modell anwenden

📌 Daten sammeln

unabhängige Variablen (Prädiktoren)

abhängige Variable (Kreditgesuch angenommen, falls alle Kreditraten bezahlt)

J, N ja, nein

W, M weiblich, männlich

V, W, A verheiratet, verwittwet, alleinstehend

Name	Geschlecht	Zivilstand	Kinder	Beschäftigung	Wohnungsm	Monatliches Einkommen	Ersparnisse	Kreditgesuch angenommen	Modell-klassifikation
Anita Conte	W	W	0	unbekannt	N	2211	J	nein	
Anita de Vargas	W	V	1	FacharbeiterIn	N	3100	J	ja	
Anita Farner	W	A	0	unbekannt	N	2983	J	nein	
Anita Fontana	W	V	0	ungelernt	N	2599	N	nein	
Anita Gehrig	W	V	2	angelernt	N	3007	J	nein	
Anita Gruner	W	V	2	Lehrer	N	7201	J	ja	
Anita Hauser	W	A	1	Führungskraft	N	8835	J	ja	
Anita Indermaur	W	V	0	unbekannt	J	2534	J	nein	
Anita Maurer	W	V	0	unbekannt	N	2664	J	nein	
Anita Mieschke	W	V	2	FacharbeiterIn	N	4019	J	ja	
Anita Sager	W	A	2	Lehrer	N	3155	J	ja	
Anita Schnyder	W	V	1	VerwaltungsangestellteR	N	6352	J	ja	
Anita Silbermann	W	V	2	Führungskraft	N	4667	J	nein	
Anita Steinmann	W	V	1	Führungskraft	N	3127	J	nein	
Anita Thüning	W	V	3	Lehrer	N	5410	J	ja	
Anita Waser	W	W	0	unbekannt	N	2555	J	nein	
Anna-Rösli Bachmann	W	V	1	FacharbeiterIn	N	3279	J	nein	
Anna-Rösli Böckli	W	V	1	Lehrer	N	8698	J	ja	
Anna-Rösli Conte	W	V	1	Führungskraft	N	6285	J	ja	

📌 Lern - und Testmenge

Predict/New

Define What to Test

Select the Portions of data you want to test on.
A separate analysis is done for each check box.
Check as many boxes as you want.

- ☒ All Input Data Range
- ☐ Primary Working Set
- ☐ Secondary Working Set
- ☒ Training Set
- ☒ Test Set
- ☐ Validation Set
- ☐ User-defined Range

- ☒ Set K-S Threshold on Analysis
- ☒ Set Gains Table on Analysis

OK

Cancel

Review

Help

Die **Aufteilung in Lern- und Testdaten** kann vom Benutzer beeinflusst werden. Wichtig ist ...

- der **Prozentsatz** der Gesamtdaten
- ob die Auswahl **zufällig** oder **systematisch** ist

Skalenniveau (Messqualität)

- **Nominalskala**

Bsp. *Kioskstandort* : [Warenhaus, Einkaufszentrum, Altstadt]

Kodierungsbeispiele : [**1 0 0** , **0 1 0** , **0 0 1**] oder [**0.1** , **0.2** , **0.3**]

Vergleichsmassstab: **gehört (nicht) zu**

- **Ordinalskala**

Bsp. *Kundenfrequenz* : [sehr klein, klein, mittel, gross, sehr gross]

Kodierungsbeispiel : [**0.2** , **0.4** , **0.6** , **0.8** , **1.0**]

Vergleichsmassstab: **grösser als**

- **Verhältnisskala**

Bsp. *Kundenfrequenz* : 1 ... 10'000

Vergleichsmassstab: **um so viel grösser als**



Die **Skalenwahl** beeinflusst die Modellwahl

③ Daten aufbereiten - **Skalierung / Codierung**

Neuronale Daten sind **numerisch** und beschränken sich oft auf einen **Wertebereich**



- ⇒ Symbolische Daten **codieren**
(Bsp. "alt" und "jung" als 0 und 1 codiert)
- ⇒ Ausgaben (und ev. Eingaben) zwischen 0 und 1 **skalieren**



Codierung und Skalierung ...

- erschweren oft das Verständnis
- beeinflussen die Wahl des Modells

Vorhersagevariablen wählen

- ✓ Problem spezifizieren
- ✓ , Lern- und Validierungsdaten sammeln
- ✓ f Daten aufbereiten
- ⇒ Vorhersagevariablen wählen
- ... Netzmodell spezifizieren und lernen
- † Modell validieren
- ‡ Modell anwenden



Unabhängige Variablen aus dem Marketing, z.B. aus der ...

Transaktionsgeschichte

- Häufigkeit und zeitliche Verteilung der Käufe
- Kaufbeträge
- Kaufbereiche
- Zahlungsgewohnheiten
- ...

Werbegeschichte

- Medium
- Adressatenkreis und -zahl
- Zahl der Wiederholungen
- ...

Variablen

Welche unabhängigen Variablen ergeben die beste Vorhersage ?

- Wohneigentum
- Einkommen
- Zivilstand
- ...



Wahl der unabhängigen Variablen ...

- **manuell** durch den Benutzer
aufgrund seiner **Domänen**kenntnisse
(z.B. des ›Direct Marketing)
- **automatisch** durch das Programm
aufgrund klassischer **statistischer** Verfahren
(z. B. der schrittweisen ›Regression)



Wenn von dreissig Variablen zehn wegfallen, so bedeutet dies bei gekauften Daten oder grossen Sammelanstrengungen *finanzielle* Einsparungen und bei der Auswertung Rechenzeit-Einsparungen

📌 Variablenbeispiele

Abhängige Variable (Ausgabeschicht)

- ✓ Kreditgesuch abgelehnt bzw. angenommen

Unabhängige Variablen (Eingabeschicht ¹⁾)

- ✓ Geschlecht
- ✓ Zivilstand
- ✓ Kinderzahl
- ✓ Beschäftigung
- ✓ Wohneigentum
- ✓ Einkommen
- ✓ Ersparnisse

¹ Diskutieren Sie die wahrscheinliche Vorhersagegüte der Eingabevariablen und ergänzen Sie die Liste um zusätzliche Eingaben

Netzmodell spezifizieren und lernen

Netzmodell



Anwendungsklasse

z.B. ›Klassifikation, ›Clustering, ›Vorhersage

+

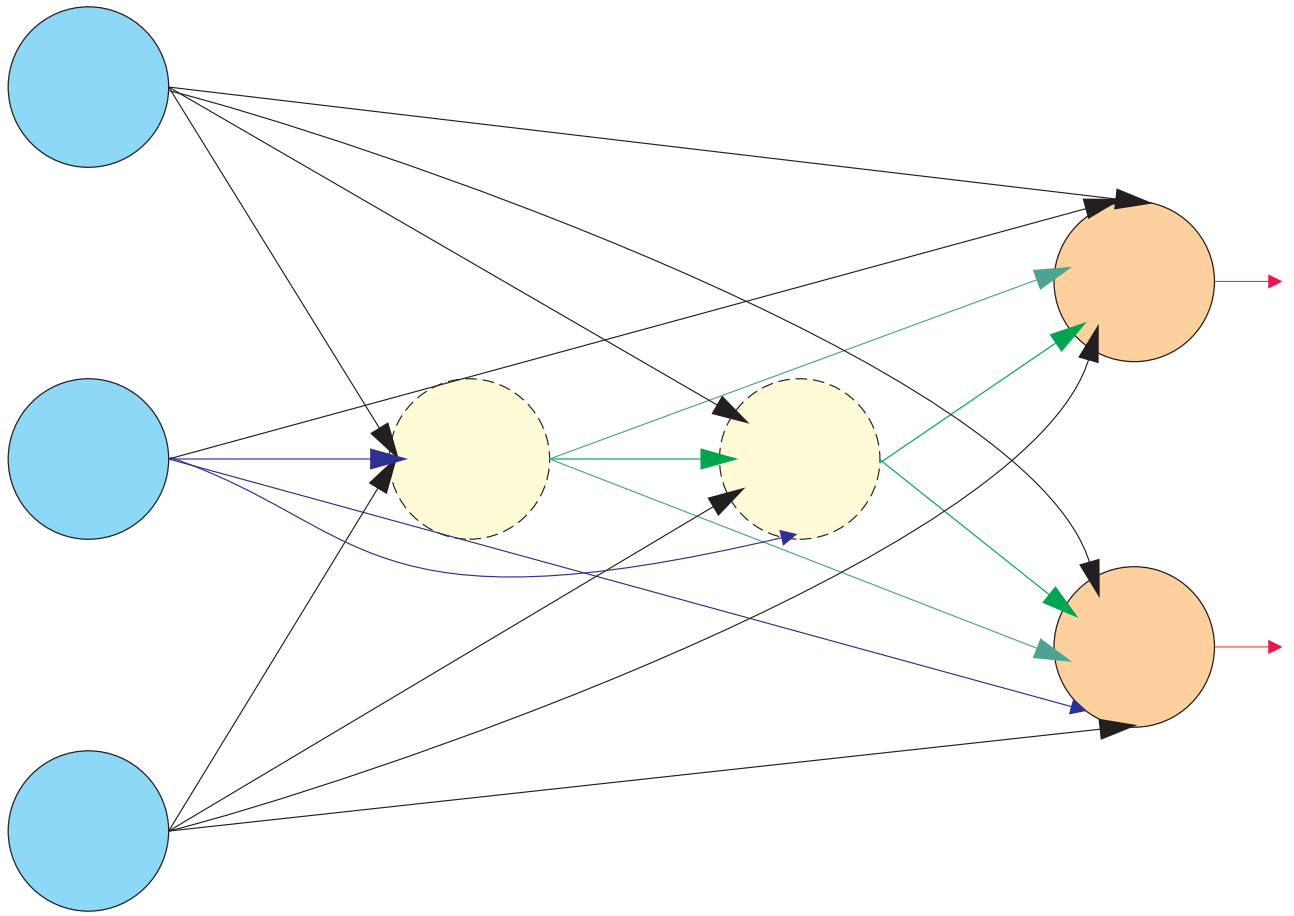
Datentyp der Eingabevariablen

z.B. binär, dezimal

+

Erforderliche Lerngeschwindigkeit

8.14 Modell von *Predict*



3 Eingabe- neuronen	2 hinzugefügte Neuronen in 2 verborgenen Schichten	2 Ausgabe- neuronen
------------------------	---	------------------------

CCN (Cascade Correlation Neural Net)

Netzmodell, das je nach Aufgabenstellung keine oder mehr verborgene Schichten inkrementell aufbaut →

8.15 📌 Modellparameter

Der Benutzer definiert Werte für die folgenden Modellparameter :

Modellparameter	Bedeutung
First Example: <i>Input Range</i>	1. Zeile der <u>un</u> abhängigen Var.
Second Example: <i>Input Range</i>	2. Zeile der <u>un</u> abhängigen Var.
First Example: <i>Output Range</i>	1. Zelle der <u>ab</u> hängigen Variable
All <i>Input Data Range</i>	Alle Zeilen der <u>un</u> abhängigen V.
Variablennamen	Bereich der Spaltenbezeichner
Noise	Zuverlässigkeit der Daten
Data Analysis and Transformation	Zahl der Variablen, die zur Verbesserung der Vorhersagevalidität transformiert wurden
Input Variable Selection ⇔	Zeitbudget zur Bestimmung der benötigten <i>unabhängigen Variablen</i>
Neural Network Search	Zeitbudget für den <i>Lernprozess</i>

Legende

Zellbereiche des Tabellenblattes

Für das Beispiel BONITÄT genügen die *Voreinstellungen*

8.15 † Modellparameter II

Edit the Network

Project: E:\DSS\Buch\BeispieleAufgaben\Bonitätsklassifikat Browse...

[Bonitätsklassifikation.xls]Antragsteller!\$A\$11 First Example Inputs

[Bonitätsklassifikation.xls]Antragsteller!\$A\$12 Second Example Inputs

[Bonitätsklassifikation.xls]Antragsteller!\$I\$11 First Example Outputs

[Bonitätsklassifikation.xls]Antragsteller!\$A\$11 All Input Data

[Bonitätsklassifikation.xls]Antragsteller!\$A\$10 Field Names

Jede Zeile beschreibt einen Antragsteller. Ein neuronales Klassifikationsmodell soll die Daten der Spaltenattribute so gewichten, dass sie die Antragsteller nach ihrer Bonität klassifizieren.

Project Description (254 characters max.)

classification Problem Type

moderately noisy data Noise Level

moderate data transformation Data Transformation

comprehensive variable selection Variable Selection

comprehensive network search Network Search

Logging...

Save

Set Defaults

Advanced...

Expert...

OK Cancel Help



Netzmodell wählen
(*Predict* gibt das CCN-Modell vor)



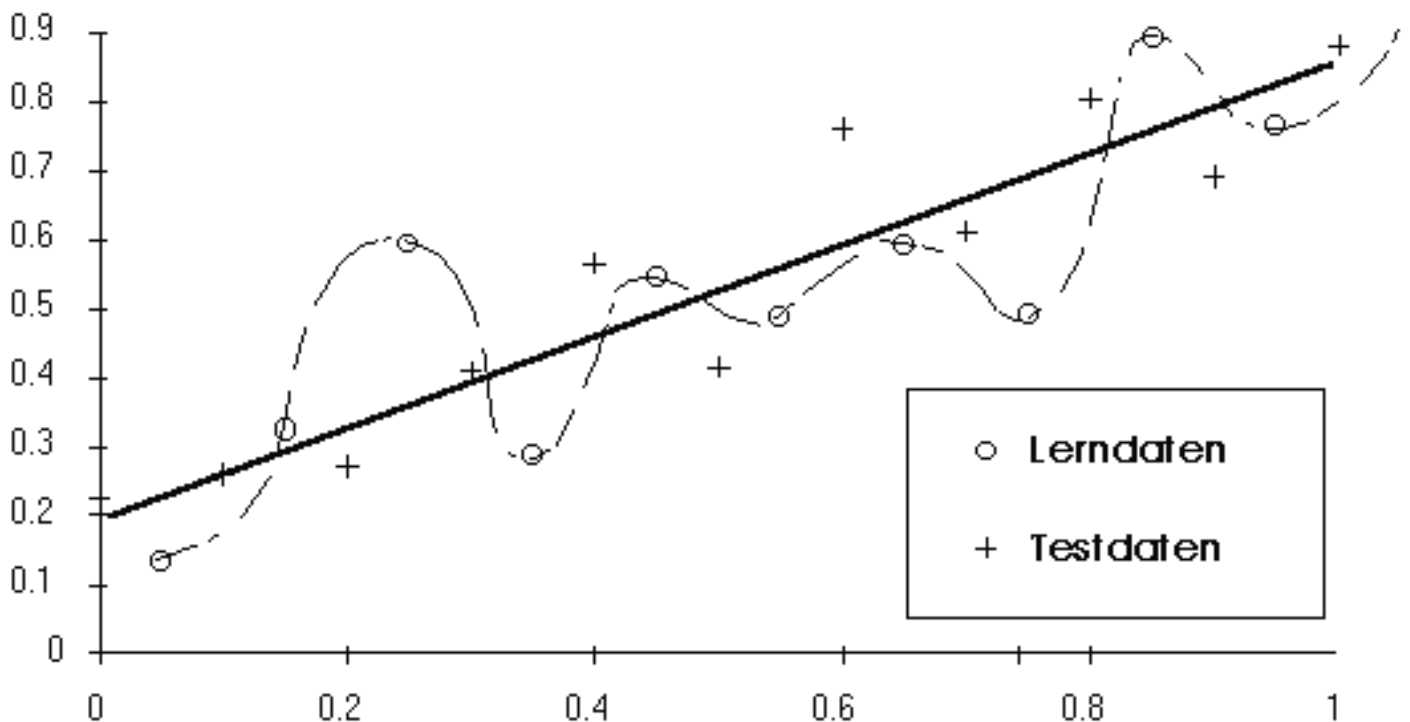
Modell validieren

📌 Tatsächliche und berechnete Klassen

Kreditgesuch angenommen?

<i>Tatsächliche Klasse</i>	<i>Berechnete Klasse</i>
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	nein
nein	ja
nein	nein
...	...

8.17 Modell validieren - Overfitting



Verallgemeinere nur das Wesentliche
statt solange zu lernen, bis sich alle
Zufälligkeiten der Lernmenge in den
Gewichten spiegeln (**Overfitting**)



- ⇒ Entdecke Overfitting durch **Validierung**
- ⇒ Einfachere Modelle sind robuster !

Modell validieren - Ein Beispiel

Tanks von anderen Objekten unterscheiden !

Lerndaten unterscheiden Tanks perfekt, aber **Validierung** scheitert

Weshalb ?

Tankbilder zufällig bei sonnigem Wetter und
Restbilder bei schlechtem Wetter aufgenommen



Algorithmus unterscheidet Wetter statt Objekte

Schwierige Schätzung ...

- des Anteils intervenierender Variablen
- des Beitrags einer einzelnen Variablen



Validität := Verallgemeinerungsfähigkeit

⑥ Modell validieren - **Validität messen**

Validität := Verallgemeinerungsfähigkeit

Validität **messen** heisst ...

- ✓ eine **repräsentative** Stichprobe ziehen
- ✓ **berechnete** und **tatsächliche** Ausgaben vergleichen
- ✓ ein **Validitätsmass** definieren, zum Beispiel ...

$$\sum_i |a_i - a'_i|$$

- ✓ Validitäten der **Lern-** und **Testmengen** vergleichen

Validität **verbessern** heisst ...

- ✓ die Stichprobe *repräsentativer* ziehen
- ✓ die Stichprobe *vergrössern*
- ✓ das Modell besser *begründen*
- ✓ das Modell *einfacher* (und so robuster) gestalten
- ✓ ...

8.17 📌 **Klassifikationsvalidität**

- ✓ Problem spezifizieren
- ✓ , Lern- und Validierungsdaten sammeln
- ✓ f Daten aufbereiten
- ✓ Vorhersagevariablen wählen
- ✓ ... Netzmodell spezifizieren und lernen
- ⇒ † **Modell validieren**
- ‡ Modell anwenden

	Trefferquote (accuracy)	Entropie	Zahl der Datensätze
Gesamtmenge	0.88	0.15	404
Lernmenge	0.88	0.15	282
Testmenge	0.89	0.15	122

- **Trefferquote** von 0.88 ⇒
Modell hätte 88% der Kreditnehmer richtig klassifiziert
- Je näher die **Entropie** bei 0, desto besser die
Klassifikation

Die **Klassifikationsleistung** neuronaler Modelle ist sehr gut.
Wegen der besseren **Erklärungsleistung** eignen sich aber
oft Entscheidungsbäume (Regelinduktion) besser

Klassifikation und Vorhersage

✓ *Fallbeispiel* zur Bonitäts**klassifikation**

Gesuchsteller in zwei Klassen einteilen:

- "Annahme" des Kreditgesuchs durch einen Bonitätsexperten
- "Ablehnung" des Kreditgesuchs durch einen Bonitätsexperten

⇒ *Aufgabe* zur Bonitäts**vorhersage**


Für jeden Gesuchsteller auf einer kontinuierlichen Skala:

- Zahl der Ratenzahlungen vorhersagen

BONITÄTSVORHERSAGE mit *Predict* (A 8.1)

Die Anwendung BONITÄTSKLASSIFIKATION hat die Kreditnehmer in nur zwei Bonitätsklassen eingeteilt. Ziel der folgenden Aufgabe ist ein *Predict*-Modell, das für einen Kreditnehmer die Zahl der der bezahlten Raten vorhersagt. Dazu verwendet es die gleichen unabhängigen Variablen wie BONITÄTSKLASSIFIKATION.

Laden Sie die Arbeitsmappe [Bonitätsvorhersage.xls](#). Sie erhalten auf drei Arten Hilfe:

- Wenn Sie den Cursor auf ein Toolbar-Symbol positionieren, erscheint eine Kurzbeschreibung des Symbols.
- Ausführliche Hilfe erhalten Sie auf einem Menüpunkt mit *Shift/F1*.
- Wenn Sie den Cursor auf die orange Zelle “Hilfe” bewegen, erscheint eine Anleitung zum Problem “Anzeigenplanung”. Für Details bewegen Sie den Cursor über Zellen mit .

Daten vorbereiten

a) Interpretieren Sie das Tabellenblatt

- Aus welchen Attributen besteht die Tabelle?
- Welches sind die Datentypen der Attribute?
- Zwischen welchen Attributen vermuten Sie Beziehungen?
- Welche Messskalen (Nominal-, Ordinal- oder Verhältnisskala) gelten für die einzelnen Variablen?

Daten aufbereiten

b) Ändern Sie ausgewählte Daten des Tabellenblatts nach Plausibilitätsüberlegungen:

- Ändern Sie Zellen.
- Fügen Sie neue Zeilen (neue Antragsteller) hinzu.
- Fügen Sie neue Spalten (unabhängige Variablen) hinzu.

Modell erstellen und testen

- c) Erstellen Sie in MS Excel mit dem Menüpunkt *Predict/New* ein Modell BONITÄTSVORHERSAGE.NPR (NeuralWorks Predict):
- Geben Sie die Datenbereiche an.
 - Bestimmen Sie die übrigen Modellparameter.
 - Lassen Sie Predict das Netzwerk erstellen.
- d) Testen Sie das Modell und interpretieren Sie die Ergebnisse (*Predict/Run*):
- Was bedeutet die Spalte “R”?
 - Weshalb heisst die Korrelation linear?
 - Was bedeutet der Wert in der Spalte “Accuracy (20%)”?
 - Was bedeutet der Wert in der Spalte “Conf. Interval (95%)”?

Gelerntes Netzwerk anwenden

- e) Zeigen Sie die vorhergesagten Werte an.
- f) Vergleichen Sie vorhergesagte und tatsächliche Werte grafisch (*Diagrammassistent*)
- g) Welche Berufsgattung ist am kreditwürdigsten?
- h) Berechnen Sie Ihre eigene Bonität.

Modell anpassen

- i) Experimentieren Sie mit den folgenden Modellparametern (*Predict/Edit ...*) :
- Problem Type
 - Noise
 - Input Variable Selection
 - Neural Network, Search
 - Lern- und Testmengen.

- j) Untersuchen Sie das Modell mit *Predict/Expert/Transforms ...* und *Predict/Expert/Go Re-train ...*
- Welche Eingabevariablen sind relevant?
 - Wie wurden die Variablen transformiert?

Alternative Methode einsetzen

- k) Verwenden Sie die statistische Methode der linearen Regression von MS Excel, um aus der gleichen Tabelle die gleiche Ausgabevariable (abhängige Variable) zu prognostizieren.

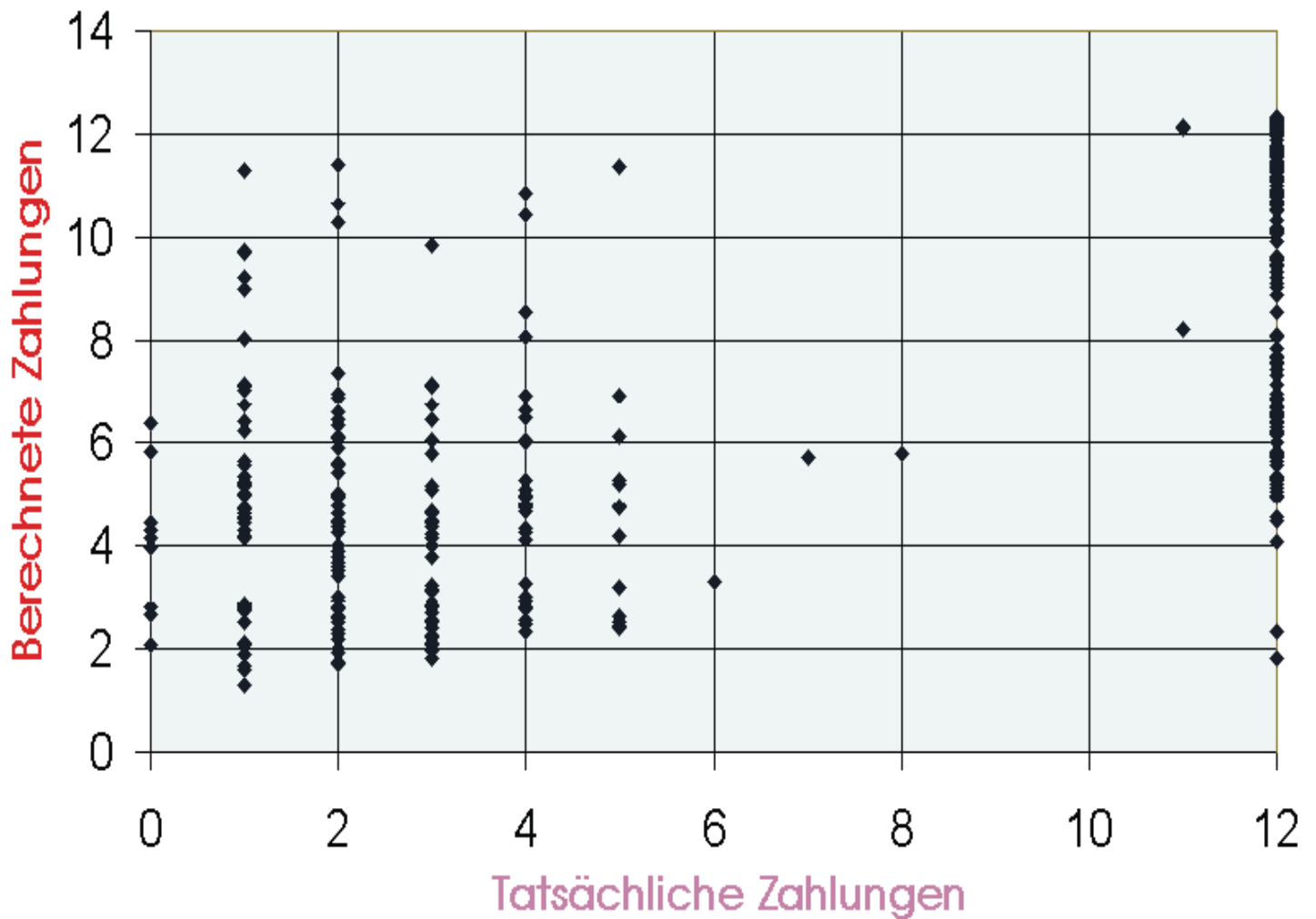
Tatsächliche und vorhergesagte Ratenzahl

<i>Anwendungsklasse</i>	<i>Abhängige</i>	<i>Beispiel</i>
Klassifikation	diskret	[“Raten bezahlt”, “Raten nicht bezahlt”]
Regression (Vorhersage)	kontinuierlich	Zahl der bezahlten Raten

Tatsächliche Ratenzahl	Vorhergesagte Ratenzahl
1	1.58
1	3.12
1	2.10
1	1.29
1	2.19
1	1.65
...	...
1	1.45
1	1.22
1	3.71
1	1.54
1	1.23
1	1.29
1	8.25
...	...

Grafisches Validitätsmass

Streuung der berechneten um die tatsächlichen Werte



Vorhersagevalidität

Lineare Korrelation r zwischen **tatsächlicher** und **berechneter** Ausgabe (zwischen 1.0 und 0.0)

Durchschnittlicher absoluter Fehler zwischen **tatsächlicher** und **berechneter** Ausgabe

Konfidenzintervall (Mit einer "Wahrscheinlichkeit" von 95% bewegt sich der **tatsächliche** Wert in einem Konfidenzintervall 6.258 vom **berechneten** Wert der Ausgabe)

	<i>r zwischen Spalten I und J</i>	<i>Durchschn. absoluter Fehler</i>	<i>Konfidenz- intervall von 95%</i>	<i>Anzahl Datensätze</i>
<i>Insgesamt</i>	0.83	1.82	5.24	404
<i>Lernmenge</i>	0.82	1.86	5.35	282
<i>Testmenge</i>	0.84	1.71	5.06	122

Funktionalität von *Predict*

Eingabe

- ✓ Eingabe- und Ausgabevariablen definieren
- ✓ Lern- und Validierungsdaten aus einer *Excel*-Tabelle übernehmen
- ✓ Modellparameter festlegen
- ✓ Unabhängige Variablen wählen

Verarbeitung

- ✓ Netz nach dem CCN-Modell aus der Stichprobe lernen

Ausgabe

- ✓ Validitätsstatistiken berechnen
- ✓ Ergebnisse in *MS Excel* grafisch und rechnerisch präsentieren

Modellieren heisst oft Experimentieren

Variiere ...

Lern- und Validierungsmengen

Modellparameter

- **Lernalgorithmus**
z.B. Anfangsgewichte
z.B. Lernrate
z.B. Abbruchkriterium
- **Architekturen**
- **Transferfunktionen**



Validitäten vergleichen !

Zeitreihenanalyse :=

Auswertung einer Folge numerischer Beobachtungen in Abhängigkeit von der Zeit

Problem

Zerlegung des Verlaufs einer Zeitreihe in ihre zeitabhängigen *Komponenten*:

$$\text{Zeitreihenwert}_t = \text{Trendwert}_t + \text{Zykluswert}_t + \text{Zufallswert}_t$$

- ✓ **Trend** (längerfristige Bewegung, welche den Grundverlauf der Zeitreihe bestimmen)
- ✓ **zyklische** (z.B. saisonale) Bewegungen
- ✓ **zufällige** Schwankungen

Verbreitete Verfahren

- Methode der gleitenden Durchschnitte
 - Methode der kleinsten Quadrate (▷ Regressionsanalyse)
- ⇒ ▷ Neuronale Netze

Anwendungen

- Vorhersage des Sozialprodukts
 - Börsenkursprognose
- ⇒ Wechselkursprognose
- ...

📌 Ziel ist die Wechselkursprognose

Gegeben

Zeitreihe aus 700 täglichen Abschlussergebnisses des Wechselkurses von Dollar und Deutscher Mark

Gesucht

Vorhersage eines oder mehrerer Tageskurse

Ansätze

- *Kausalanalyse*

Vorhersage aus mehreren Faktoren der Wechselkursschwankungen (unabhängige Variablen wie Zinssatz, Beschäftigungsrate, Börsenindex)

⇒ *Zeitreihenanalyse*

Vorhersage künftiger Wechselkurse aus einer Teilmenge vergangener Wechselkurse

Varianten

- ① Vorhersage des nächsten Tageskurses aus den vergangenen 15 Tagessätzen
- ② Vorhersage der nächsten 35 Tagekurse, wobei (mit Ausnahme der ersten Prognose) die bisher vorhergesagten Werte in die neue Prognose eingehen

Werkzeug

SPSS *Neural Connection* berechnet aus den Zeitreihenbeobachtungen (und in der zweiten Variante vorangehender Prognosewerte) ein »RBF-Netzwerk

Demonstration zu SPSS Neural Connection

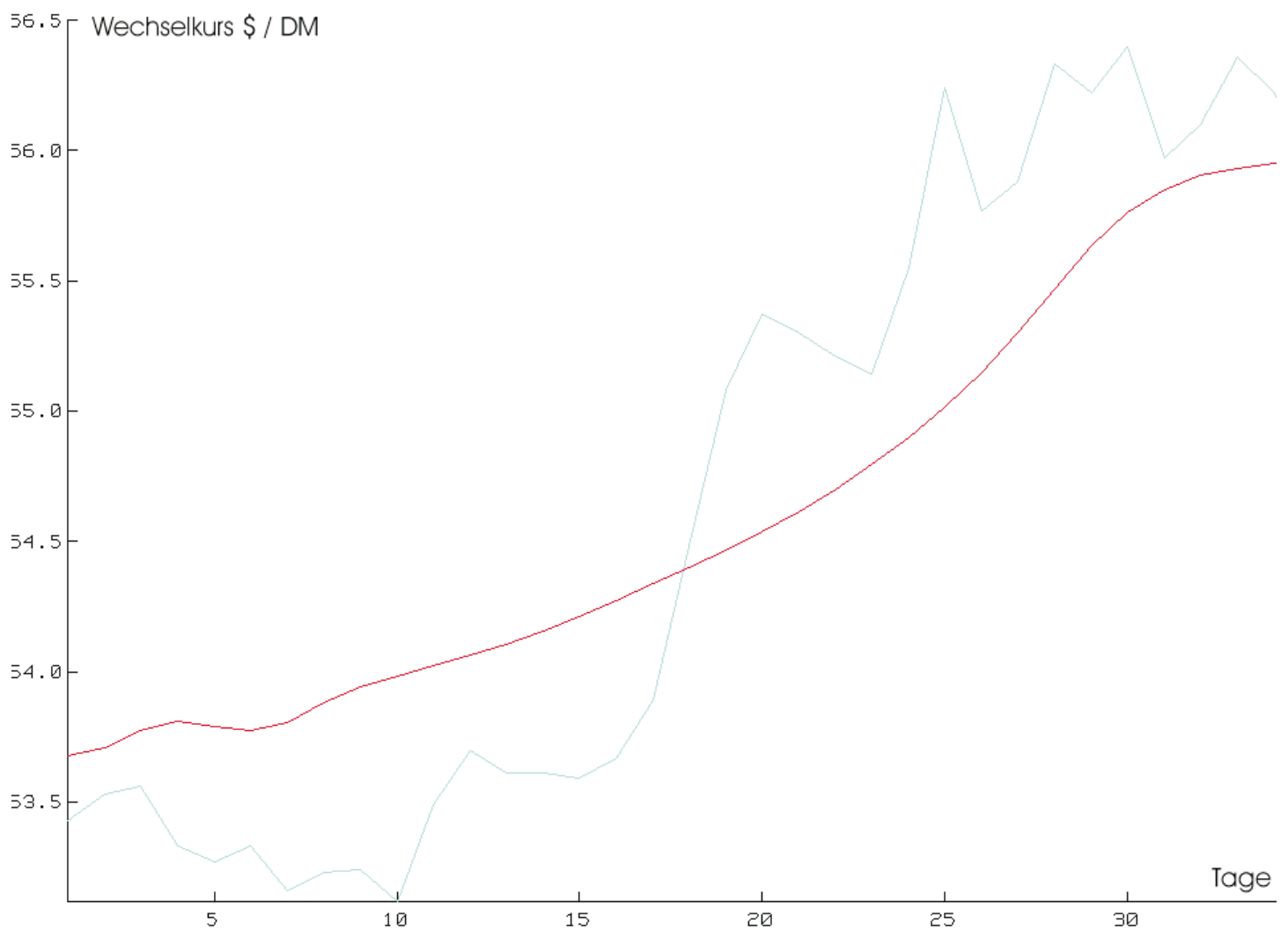
(*Contents - Demonstrations - Financial Time Series* (Doppelklick))

📌 Ergebnisvariante



tatsächliche Wechselkurse
berechnete Wechselkurse

📌 Ergebnisvariante ,



tatsächliche Wechselkurse
berechnete Wechselkurse

Zur Mächtigkeit neuronaler Netze

Bestimmte Arten mehrstufiger vorwärts - gerichteter Netze, zum Beispiel RBF-Netzwerke, können alle Funktionen

$$\text{Ausgabe} = f(\text{Eingaben})$$

mit beliebiger Genauigkeit approximieren

8.56 Methode im Vergleich

Kriterium	AHP	Optimierung	OLAP	Regelbasierte Systeme	Induktion	Neuronale Netze	Regression
Methode breit anwendbar		–	+			1	–
Automatisierungsgrad	–	+	–		+	+	+
Ergebnis genau	–	+	+		+	+ ²	+
Unabhäng. Var. gewichtbar	–	–	–	–		– ³	+
Lösungsweg begründbar		–		+	+	– ⁴	–
Methode plausibel	+		+	+		– ⁴	
Ergebnis einbettbar	–	+			+		+
Entwicklungsaufwand	+	+	–	–	+		
Rechnerbelastung	+	+	–		+	–	+

Kriterien

- 1 vor allem auf numerische Probleme
- 2 insbesondere beliebig genaue Anpassung nichtlinearer Beziehungen
- 3 nur mit Hilfe externer Verfahren (insb. über eine Sensitivitätsanalyse)
- 4 Hauptnachteil
- 5 Integration in RDBMS wegen der hohen Rechenintensität aufwendig
- 6 Vorbereitung der Daten aufwendig

Zusatzvorteile

- Fehlertoleranz bei fehlenden oder fehlerhaften Daten
- Eignung für *parallele* Verarbeitung von Daten

Theoretischer Hintergrund

Es gibt nichts Praktischeres
als eine gute Theorie !



Einfache Ausschnitte aus der
Theorie der neuronalen Netze

Einblick in die Theorie

Grundlagen

✓ Neuron	<u>7</u>
✓ Neuronales Netz	<u>10</u>
✓ Neuronales Lernen	<u>13</u>

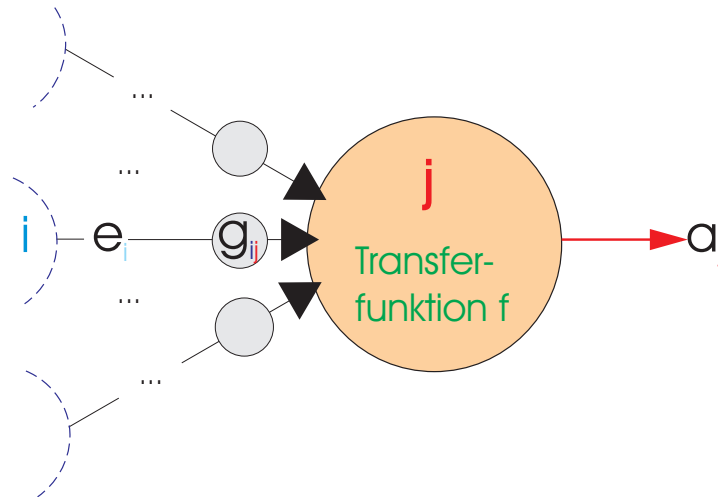
Anwendung

✓ Anwendungsklassen	<u>34</u>
✓ Entwicklung mit <i>Predict</i> 📌	<u>40</u>

Einblick in die Theorie

⇒ Transferfunktion und Schwellenwert	<u>82</u>
⇒ Zweiklassen-Perzeptron	<u>90</u>
• Eindimensionales Perzeptron	<u>93</u>
• Zweidimensionales Perzeptron	<u>107</u>
⇒ Mehrklassen-Perzeptron	<u>127</u>
⇒ Neuronales Lernen mit Fehlerrückführung	<u>145</u>

8.18 Transferfunktion



Transferfunktion f

- transferiert alle mit g_{ij} gewichteten Eingaben e_i
- als Ausgabe a_j auf einem bestimmten Wertebereich (oft -1..+1)
- an ein anderes Neuron oder die Umwelt

$$a_j = f(g_{1j} e_1, \dots, g_{nj} e_n)$$

i, j	Neuronen i und j
e_i	Eingabe vom i -ten Neuron
g_{ij}	Gewicht der Verbindung zwischen i und j
a_j	Ausgabe des j -ten Neurons
f	Transferfunktion ¹



Gebräuchliche Transferfunktionen?

¹ auch Aktivierungsfunktion genannt

Addierende Transferfunktion

Verknüpfung der Eingaben e_i eines Neurons zu einer einzigen Ausgabe a :

$$g_{1j} e_1 \dots g_{nj} e_n = g_1 e_1 \dots g_n e_n$$



Beschränkung auf die folgende additive Verknüpfung:

$$\underline{\text{Nettoeingabe}} = \text{sum} = g_1 e_1 + \dots + g_n e_n^1$$



Einführung einer Funktion der Nettoeingabe

$$a = f(\text{Nettoeingabe}) = f(g_1 e_1 + \dots + g_n e_n)$$

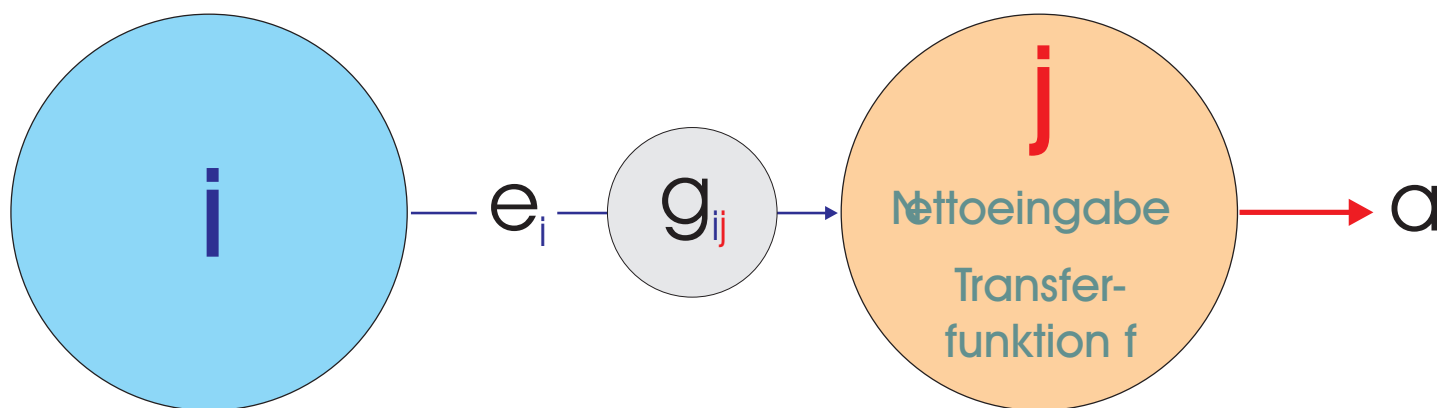


Beispiele ⇒

⏮ ⏭

¹ Seltener werden die gewichteten Eingaben anders verknüpft (Vgl. die multiplizierende Transferfunktion im bereits behandelten Lernbeispiel UND)

Nettoeingabe als Skalarprodukt



$$\text{Nettoeingabe} = \text{sum} = g_1 e_1 + \dots + g_n e_n =$$

$$\sum_{i=1}^n g_i \cdot e_i =$$

$$\mathbf{g} \cdot \mathbf{e}$$

Skalarprodukt der **Gewichtezeile** und der **Eingabenspalte**

Rechenbeispiel

			20
			10
			5
0.1	0.2	0.4	$0.1 \cdot 20 + 0.2 \cdot 10 + 0.4 \cdot 5$

8.19 Beispiele additiver Transferfunktionen

① Identitätsfunktion

$$f(\text{sum}) = \text{sum}$$

② Treppenfunktion \Rightarrow

$$\text{z.B. } f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} > 0 \\ -1 & \text{sonst} \end{cases}$$

a ist +1, falls **e** zu einer ersten Klasse gehört

a ist -1, falls **e** zu einer zweiten Klasse gehört

③ Monotone nichtlineare Funktion

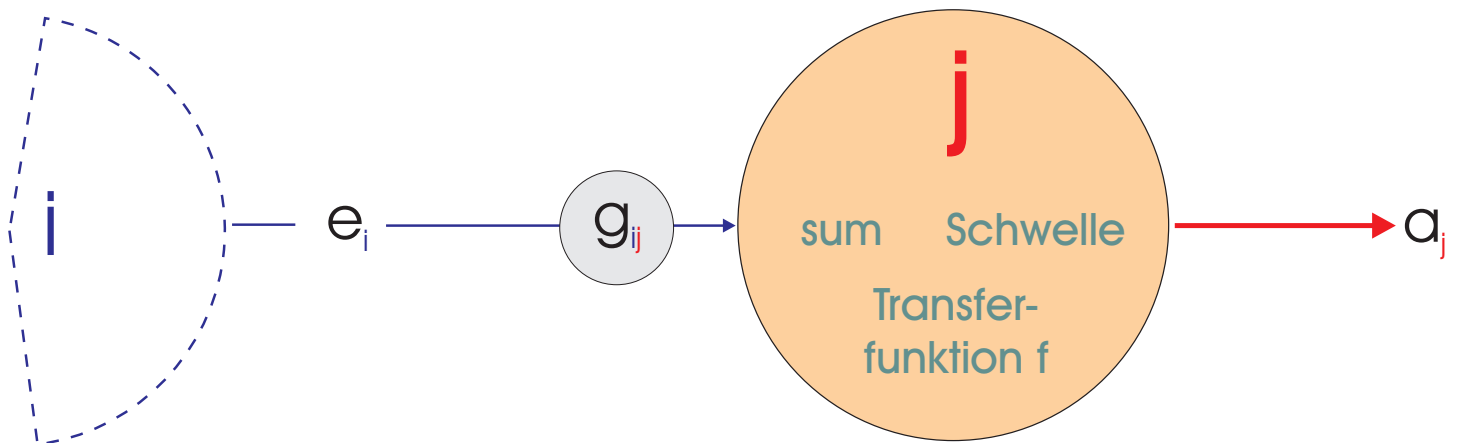
z.B. Sigmoidfunktion (logistische -) wie ...

$$f(\text{sum}) = \frac{1}{1 + \exp(-\text{sum})}$$

Eine Transferfunktion gilt für ...

- ✓ ein einzelnes Neuron oder
- ✓ eine Neuronenschicht oder
- ✓ alle Neuronen eines ganzen Netzes

Schwellenkonstante



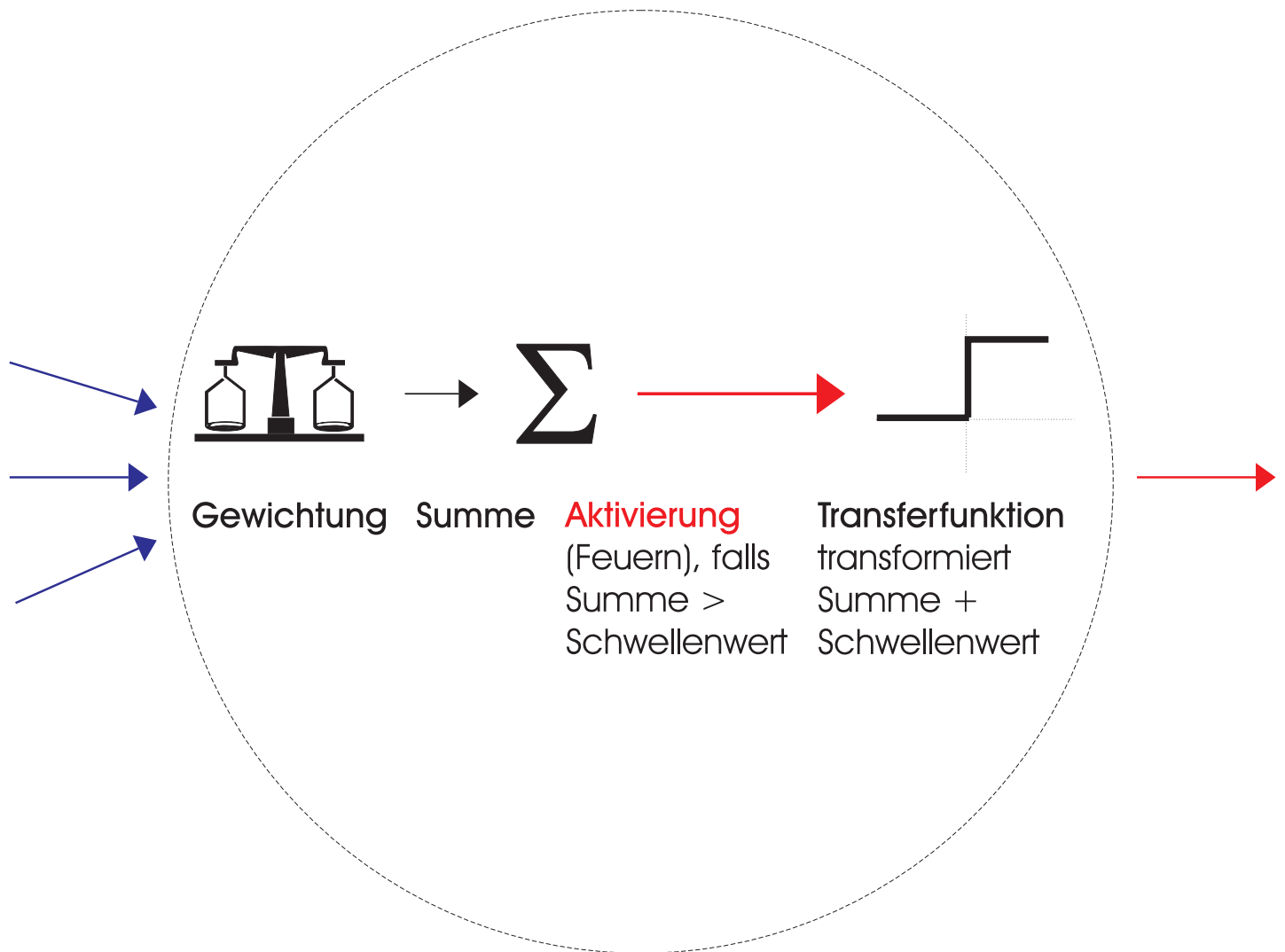
Zur Nettoeingabe addieren wir eine Konstante g_0 :

$$a = f (g_0 + \text{sum})$$

Schwellenkonstante g_0 := Wert, den die Nettoeingabe kompensieren muss, damit das Neuron gerade noch feuert

*Bsp.: Wenn die Schwellenkonstante -4 ist,
dann muss die Nettoeingabe mehr als +4 sein*

8.20 Schwellenkonstante im Zusammenhang



Ausgabe und Schwellenkonstante

Wir integrieren künftig die Schwellenkonstante g_0 in sum und definieren dazu eine *fiktive Eingabe* $e_0 = 1$, so dass ...

$$e_0 \cdot g_0 = 1 \cdot g_0$$

$$\text{sum} = \sum_{i=0}^n g_i \cdot e_i \quad i = 0, 1, \dots, n$$

Beispiel

$$\text{sum} = g_0 + g_1 e_1 = g_0 e_0 + g_1 e_1 = \sum_{i=0}^1 g_i \cdot e_i$$



$$a = f \left(\sum_{i=0}^n g_i \cdot e_i \right) = f (\mathbf{g} \cdot \mathbf{e})$$

wobei \mathbf{g} Gewichtevektor und \mathbf{e} Eingabenvektor

Klassifikation

Eine **Klassifikation** ist eine Funktion, die einem Merkmalsvektor einen Klassennamen aus einer vorgegebenen Menge zuordnet



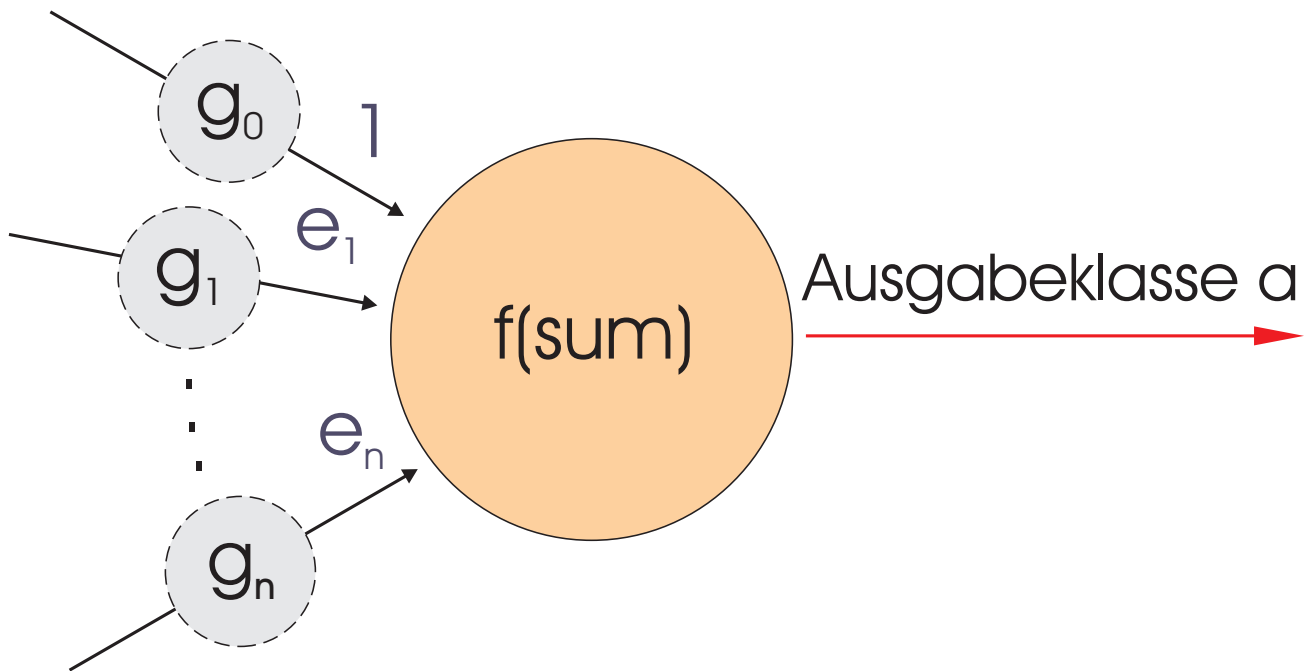
Beispiel BONITÄT

Gesucht ist eine Funktion, die jedem Merkmalsvektor [Beschäftigung, Wohneigentum, Einkommen, ...] die Klasse *Annahme* oder *Ablehnung* zuordnet



Die Klassifikation wird vor allem von bestimmten Modellen, zum Beispiel **Perzeptrons** unterstützt →

Klassisches Perzeptron



(Einstufiges) **Perzeptron** := einstufiges neuronales Netz, das einem **Eingabenvektor** mit einer meist *linearen* oder Treppenfunktion eine **Ausgabe**, i.d.R. eine Klasse, zuordnet

Perzeptrons veranschaulichen Grundsätze,
die für alle neuronalen Netze gelten :



Architekturen ?
Transferfunktionen ?
Lernalgorithmen ?

Wie klassifizieren Perzeptrons?

① als neuronale Netze mit ...

- **einer** Gewichtungsstufe (klassisches Perzeptron)
- **mehreren** Gewichtungsstufen

② in zwei oder mehr Klassen

- **Zweiklassen** - Perzeptrons
- **Mehrklassen** - Perzeptrons

③ nach einer oder mehr Eingabedimensionen

- **Eindimensionale** Perzeptrons
- **Mehrdimensionale** Perzeptrons

nach den Datentypen der Variablen

- **binäre** Eingaben bzw. Ausgaben
- **reelle** Eingaben bzw. Ausgaben



Wir betrachten die ursprünglichen Perzeptrons,
nämlich **Perzeptrons mit Treppenfunktion**

Beispiele und Aufgaben zu Perzeptrons

Neuronale Netze

Perzeptrons i.w.S. (ein- oder mehrstufige Perzeptrons)

Perzeptrons i.e.S. (einstufige Perzeptrons)

Perzeptrons mit Treppen-Transferfunktion

eindimensionale Perzeptrons

EINDIMPERZEPTRON  mit *Excel*

mehrdimensionale Perzeptrons

Zweiklassen-Perzeptrons

ZWEIDIMPERZEPTRON mit *VBA*

Mehrklassen-Perzeptrons

MEHRKLASSPERZEPTRON *VBA*

Perzeptrons mit anderer Transferfunktion

...

mehrstufige Perzeptrons (Netze mit Backpropagation)

MEHRSTUFPERZEPTRON mit *VBA*

CCN-Netze

BONITÄTSKLASSIFIKATION  mit *Predict*

BONITÄTSVORHERSAGE mit *Predict*

...

8.22 \$ EINDIMPERZEPTRON - Problem

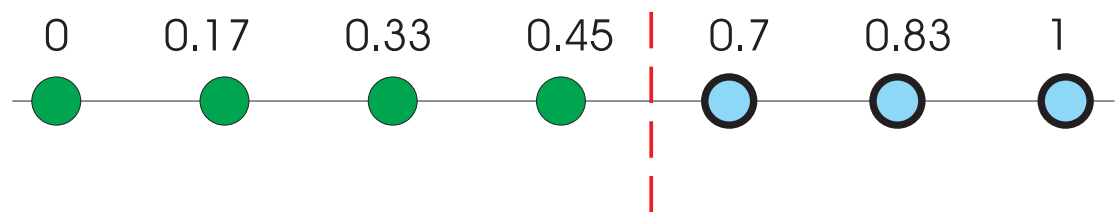
Problem

Erstellen Sie ein *MS Excel-Tabellenblatt*, das aus einer Stichprobe numerischer Daten die Einteilung beliebiger Zahlen in zwei Klassen lernt. Verwenden Sie dabei die Lernmethode eines eindimensionalen Zweiklassen-Perzeptrons.

Gegeben

Sieben Lernpaare der Form [Eingabezahl, Ausgabeklasse]

<i>Eingabe</i>	0	0.83	0.33	0.7	0.17	1	0.45
<i>Klasse</i>	-1	+1	-1	+1	-1	+1	-1



Trenngerade

Gesucht

Gewichte eines eindimensionalen Perzeptrons

Entwicklung

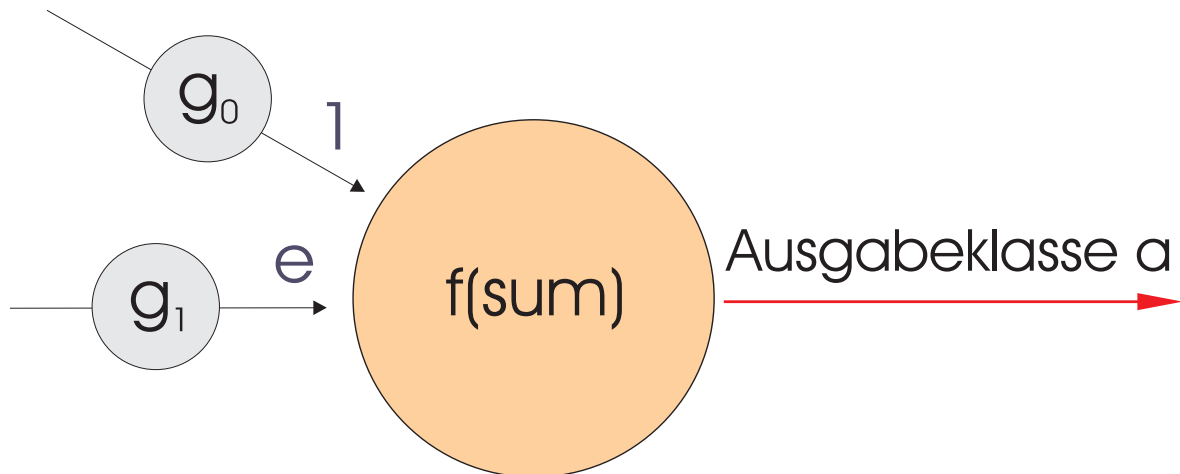
- ✓ ① Problem
- ⇒ ② Modell
- ③ Lernalgorithmus
- Tabellenblatt

8.23 \$, Modell

Ein **e**indimensionales **Z**weiklassen-Perzeptron besteht aus einem ...

- *kontinuierlichen* **E**ingabeneuron (und einer fiktiven Eingabe 1)
- *binären* **A**usgabeneuron

Architektur



Transferfunktion

Eingabevariable
Nettoeingabe

reelle Zahl e_1
 $\text{sum} = g_0 + g_1 e_1$

Ausgabeklasse

$a = f(\text{sum}) = +1$ falls $\text{sum} \geq 0$
 -1 sonst

Lernalgorithmus

Gewichte g_0 und g_1 bestimmen, die **E**ingaben den **K**lassen **1** oder **-1** zuteilen?

Gegeben

Lernpaare (e_1, k) , wobei k = tatsächliche Klasse

Lernalgorithmus und Lernrate α (beliebig zwischen 0 und 1)

8.24 \$ ③ Ein Lernalgorithmus

Initialisiere die Gewichte g_0 und g_1 beliebig ①
Berechne alle Ausgaben $a := f(\text{sum}) = f(g_0 + g_1 e_1)$ ②
GEHE WIEDERHOLT DURCH ALLE Lernpaare (e_1, k) ③
 FALLS a die laufende Eingabe e_1 fehlklassifiziert
 $g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)$
 $g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$
 Berechne alle Ausgaben a neu
BIS berechnete Klassen $a =$ tatsächliche Klassen k

bis Lernregel

FALLS $\text{berechnete} - \text{tatsächliche}$ Ausgabe $\neq 0$

$$g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)^1$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$$

α klein Lernen langsam, aber stabil

α gross Lernen schnell, aber ev. um k oszillierend

Abbruchkriterium

Perzeptrons ändern ihre Gewichte nur, wenn die laufende berechnete Ausgabe ungleich der tatsächlichen Ausgabe ist. Sie lernen also aus *schlechten* Erfahrungen

Es gibt mehrere Gewichtepaare, welche das Abbruchkriterium erfüllen. Die Höhe der Gewichte hängt von ihren *Initialisierungen* und der *Lernrate* ab

13 15

1 Welche Werte kann die Differenz $(k-a)$ annehmen?

Lernregeln

Perzeptrons unterscheiden sich in ...

- ✓ Architektur
- ✓ Transferfunktion
- ✓ Lernalgorithmus, insbesondere Lernregel

Lernregeln

⇒ ergeben Gewichte in Abhängigkeit von ...

- ✓ *Eingabe*
- ✓ *berechneter* - *tatsächlicher* Ausgabe

- ✓ ① Problem
- ✓ ② Modell
- ✓ ③ Lernalgorithmus
- ⇒ Tabellenblatt

EINDIMPERZEPTRON :
Zahlenklassifikation in *MS Excel*

Aufbau des Tabellenblatts

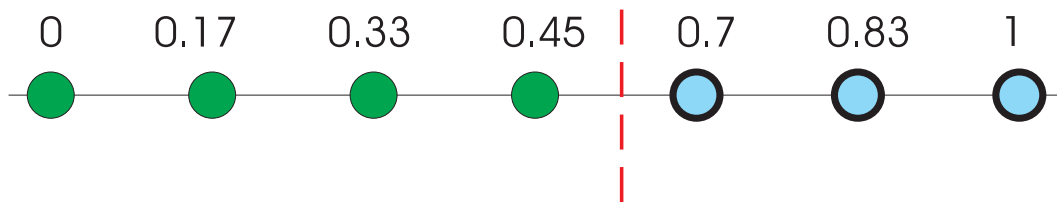


Ergebnis des Tabellenblatts



Formeln des Tabellenblatts

8.25 \$ Aufbau des Tabellenblatts



Gegeben	Rang von e_1 (1,2, ... 7)	7	4	2	6	3	1	5	$\alpha = 0.1$	
	Tatsächliche Klasse k	-1				+1			g_0	g_1
	Geordnete Eingaben e_1	0.0 ... 0.45				0.7 ... 1.0			① -1	① -0.36
Berechnet	Lernschritt 1
	Lernschritt 2
	...									
	⑧ e_1 richtig klassifiziert	Alle a sind -1				Alle a sind +1			Gesuchte g_0, g_1	

Der Lernalgorithmus *bricht ab*, sobald

⇒ die Transferfunktion f mit

⇒ g_0 und g_1 des laufenden Lernschrittes die Eingaben ...

- 0.0 bis 0.45 der Klasse **-1** und
- 0.7 bis 1.0 der Klasse **+1** zuordnet



Ergebnis des Tabellenblatts

8.26 \$ Ergebnis des Tabellenblatts

Elementfolge	0	4	2	6	3	1	5	$\alpha = 0.1$	
Klasse	-1	-1	-1	-1	1	1	1	g_0	g_1
Eingabe	0.00	0.17	0.33	0.45	0.70	0.83	1.00	-1	-0.36
Lernschritt 1	-1.00	-1.06	-1.12	-1.16	-1.25	-1.30	-1.36	-0.80	-0.19
2	-0.80	-0.83	-0.86	-0.89	-0.94	-0.96	-0.99	-0.80	-0.19
3	-0.80	-0.83	-0.86	-0.89	-0.94	-0.96	-0.99	-0.60	-0.05
4	-0.60	-0.61	-0.62	-0.62	-0.64	-0.64	-0.65	-0.60	-0.05
5	-0.60	-0.61	-0.62	-0.62	-0.64	-0.64	-0.65	-0.40	0.15
6	-0.40	-0.38	-0.35	-0.33	-0.30	-0.28	-0.25	-0.40	0.15
7	-0.40	-0.38	-0.35	-0.33	-0.30	-0.28	-0.25	-0.40	0.15
8	-0.40	-0.38	-0.35	-0.33	-0.30	-0.28	-0.25	-0.20	0.31
9	-0.20	-0.15	-0.10	-0.06	0.02	0.06	0.11	-0.20	0.31

Formel für die Berechnung von g_0 in Zelle J9 ?

siehe ...

 EindimPerzeptron.xls

8.27 \$ Ausgewählte Formeln

Jeder Schritt . . .

- lernt die neuen Gewichte g_0 und g_1
- berechnet aus den **Eingaben** und Gewichten die neue **Klasse**

<Reihenfolge der e_1 > <Tatsächliche Klasse k > <Eingaben e_1 >	$\alpha = 0.1$	
	g_0	g_1
	① -1	① -0.36
③ $\text{sum} = g_0 + g_1 \cdot e_1 =$ $-1 + -.36 \cdot .83 = \underline{-1.13}$. . .	$g_0 + \alpha \cdot 1 \cdot (k - a) =$ $-1 + .1 \cdot (1 - (-1)) = -1 + .1 \cdot 2 = -.8$. . .	$g_1 + \alpha \cdot e_1 \cdot (k - a) =$ $-.36 + .1 \cdot .83 \cdot (1 - (-1)) =$ $-.36 + .083 \cdot 2 =$ $-.36 + .166 = \underline{-.194}$. . .
③ Berechnet = tatsächlich	endgültig	endgültig

Berechnete Klassen gelb

Beliebige Startgewichte dunkelgrau

Berechnete Gewichte hellgrau



Zelle J9 in *MS Excel*

\$ Lernregel vereinfachen

Gegeben (Lernregel für eindimensionale Perzeptrons)

FALLS **berechnete** - **tatsächliche** Ausgabe \neq 0 DANN

$$g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$$

Behauptung

Die FALLS-Bedingung der Lernregel kann weggelassen werden

Beweis

Wenn $k = a$ dann $(k - a) = 0$, also gilt ...

$$g_0 := g_0 + \alpha \cdot 1 \cdot 0 = g_0$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot 0 = g_1$$

d.h. g_0 und g_1 bleiben nach einer richtigen Klassifikation unverändert.

Die Prüfung, ob $(k - a) \neq 0$ erübrigt sich also.



Implementation in MS Excel

⇒ VORZEICHEN(-1.3) ergibt die berechnete Klasse $a = -1$

VORZEICHEN(.11) ergibt $a = 1$

⇒ ZEILE() ergibt die Nummer der Tabellenblatt-Zeile,
deren Gewichte berechnet werden sollen

⇒ WVERWEIS sucht die Zelle, in der die tatsächliche Klasse k steht

8.28 \$ Formel für g_0 in Zelle J9

	C	D	E	F	G	H	I	J	K
5	0	4	2	6	3	1	5	$\alpha =$	0.1
6	-1	-1	-1	-1	+1	+1	+1	g_0	g_1
7	0.00	0.17	0.33	0.45	0.70	0.83	1.00	-1	-0.36
8	-1.00	-1.06	-1.12	-1.16	-1.25	-1.30	-1.36	-0.80	-0.19
9	-0.80	-0.83	-0.86	-0.89	-0.94	-0.96	-0.99	-0.80	-0.19
...	↑	...

Excel-Formel der vereinfachten Lernregel für Zelle J9

J8 + alpha

$g_0 +$

WVERWEIS (

(k

REST(ZEILE() ; 7); laufendes g_0 gehört zum 2. Eingabe, weil
 Rest(ZEILE() DIV 7)=2, wobei ZEILE()=9
\$C\$5:\$I\$25 ; Suchbereich von WVERWEIS
2 ; Zeilenr. von k innerhalb \$C\$5:\$I\$25
FALSCH) Nummernzeile nicht sortiert

VORZEICHEN (

a)

WVERWEIS (

REST(ZEILE(); 7); wie oben

\$C\$5:\$I\$25 ; wie oben

ZEILE()-4 ; -0.86 steht in der 9-4=5ten Zeile von
\$C\$5:\$I\$25

FALSCH) Die Nummernzeile ist nicht sortiert

)

= -0.80

\$ Zusammenfassung

Gegeben

Lernmenge	von Paaren (e_1, k) , wobei e_1 reeller Eingabeskalär und k tatsächliche Klasse von e_1 (-1 oder +1)
Gewichte	g_0 und g_1
Nettoeingabe	$\text{sum} = g_0 + g_1 e_1$
Berechnete Klasse	$a = f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} \geq 0 \\ -1 & \text{sonst} \end{cases}$
Lernrate	α

Gesucht

g_0 und g_1 , die jede neue Eingabe der Klasse -1 oder +1 zuordnen

Lernalgorithmus

Initialisiere die Gewichte g_0 und g_1 beliebig

Berechne alle Ausgaben: $a := f(\text{sum})$

GEHE WIEDERHOLT DURCH ALLE Lernpaare (e_1, k)

FALLS a die laufende Eingabe e_1 fehlklassifiziert

$$g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$$

Berechne alle Ausgaben a neu


BIS berechnete Klassen a = tatsächliche Klassen k


Satz

Der Algorithmus terminiert mit dieser Lernregel bei hinreichend kleinem α (Eine gute Wahl liegt zwischen 0 und 1)

EINDIMPERZEPTRON mit *Excel* (A 8.2)

Zuerst lernen Sie die Implementation eines eindimensionalen Zweiklassen-Perzeptrons aus der *Benutzersicht* kennen, dann analysieren, vervollständigen und modifizieren Sie das Tabellenblatt aus der Sicht des *Entwicklers*.

Laden Sie die Arbeitsmappe  [EindimPerzeptron.xls](#). Die Formeln sind geschützt. Sie können deshalb zuerst nur die Benutzeroberfläche betrachten.

- Wenn Sie den Cursor auf ein Toolbar-Symbol positionieren, erscheint eine Kurzbeschreibung des Symbols.
- Ausführliche Hilfe erhalten Sie auf einem Menüpunkt mit *Shift/F1*.
- Wenn Sie den Cursor auf die orange Zelle “Hilfe” bewegen, erscheint eine Anleitung zum Problem “Anzeigenplanung”. Für Details bewegen Sie den Cursor über Zellen mit .

Benutzeroberfläche kennen lernen

Bewegen Sie den Cursor auf die Zelle “Hilfe” und lesen Sie die Anleitung. Versuchen Sie dann, die Aufgaben der Benutzeroberfläche zu erraten. Wenn nötig, bewegen Sie den Cursor auf die Zellenkommentare.

- a) Unterscheiden Sie zwischen Zellbereichen für Daten, Benutzereingaben, Programmausgaben und Kommentare.
- b) Welche Bedeutung haben die verschiedenen Farben?
- c) Wie erkennen Sie, dass die Abbruchbedingung des Lernalgorithmus erfüllt ist?

Tabellenblatt anwenden

- d) Experimentieren Sie mit verschiedenen Eingaben, Lernraten und Startgewichten. Welche Schlüsse zur Höhe der Gewichte ziehen Sie daraus?

Tabellenblatt ergänzen

Laden Sie die Arbeitsmappe [EindimPerzeptronSkelett.xls](#). Sie enthält die gleiche Benutzeroberfläche wie `EindimPerzeptron.xls`, der Codeteil ist aber nicht mehr geschützt.

- e) Versuchen Sie die Berechnungsfolge anhand der Zellenkommentare zu verstehen (*Extras/Detektiv* veranschaulicht Formeln grafisch!)
- f) Die Formeln des ersten Lernschritts zur Berechnung von g_0 und g_1 sind unvollständig. Setzen Sie die richtigen Formeln ein. Nehmen Sie dabei die Zellenkommentare zu Hilfe.
- g) Versuchen Sie mit Hilfe des Zellenkommentars die Formel für Zelle J9 zu verstehen.
- h) Die Tabelle fasst nur wenige Lernschritte. Erweitern Sie das Tabellenblatt auf dreissig Lernschritte.
- i) Das Verständnis von `EindimPerzeptron.xls` ist nicht einfach. Begründen Sie, weshalb Entwicklung und Wartung grosser Tabellenblätter mit komplexen Formeln schwierig sind.

Abbruchkriterien sind nicht einheitlich

Lernalgorithmen für **komplexere Architekturen** ...

z.B. mehrdimensionale Perzeptrons oder Backpropagation

- haben **andere** Abbruchkriterien

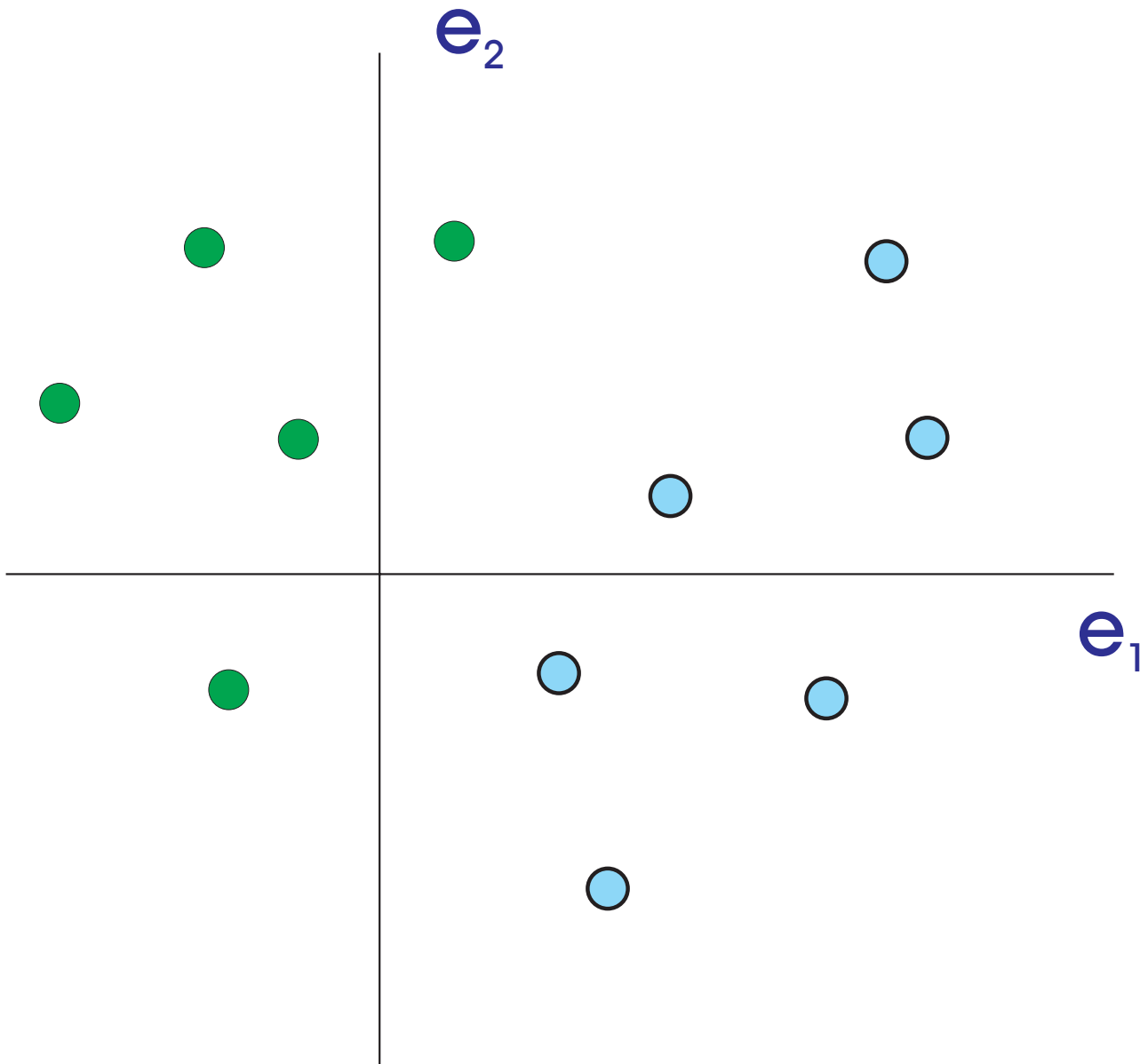
z.B. BIS durchschnittlicher Fehler zwischen berechneten Klassen **a** und tatsächlichen Klassen **k** *minimal*

- terminieren **nicht immer**

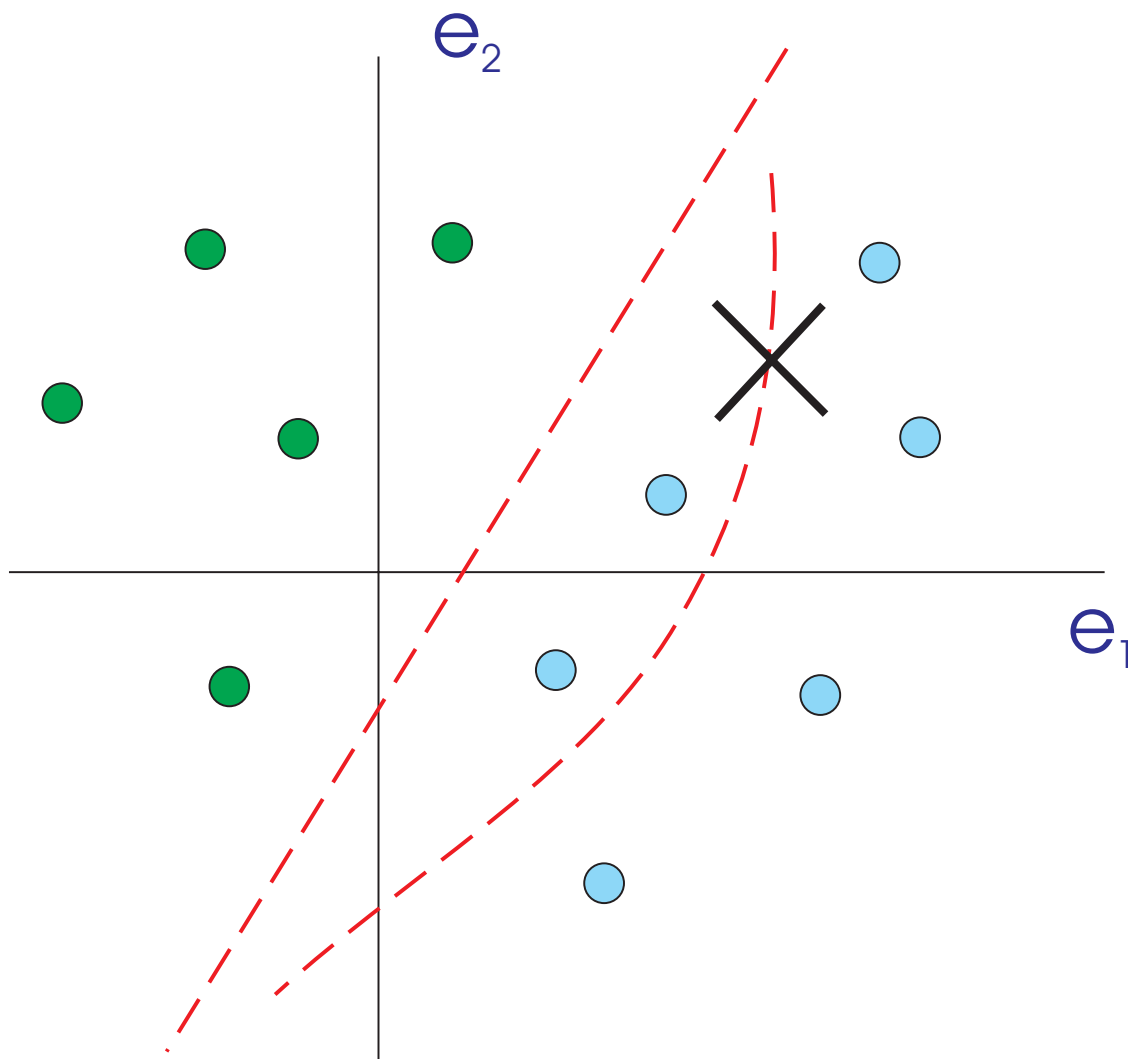
z.B. wenn das Klassifikationsproblem *nichtlinear* ist

8.29 Zwei Dimensionen und zwei Klassen

Wie lässt sich eine zweidimensionale Eingabe (e_1, e_2) *linear* einer von zwei Klassen zuordnen?



8.30 Trenngerade für 2 Dimensionen und 2 Klassen



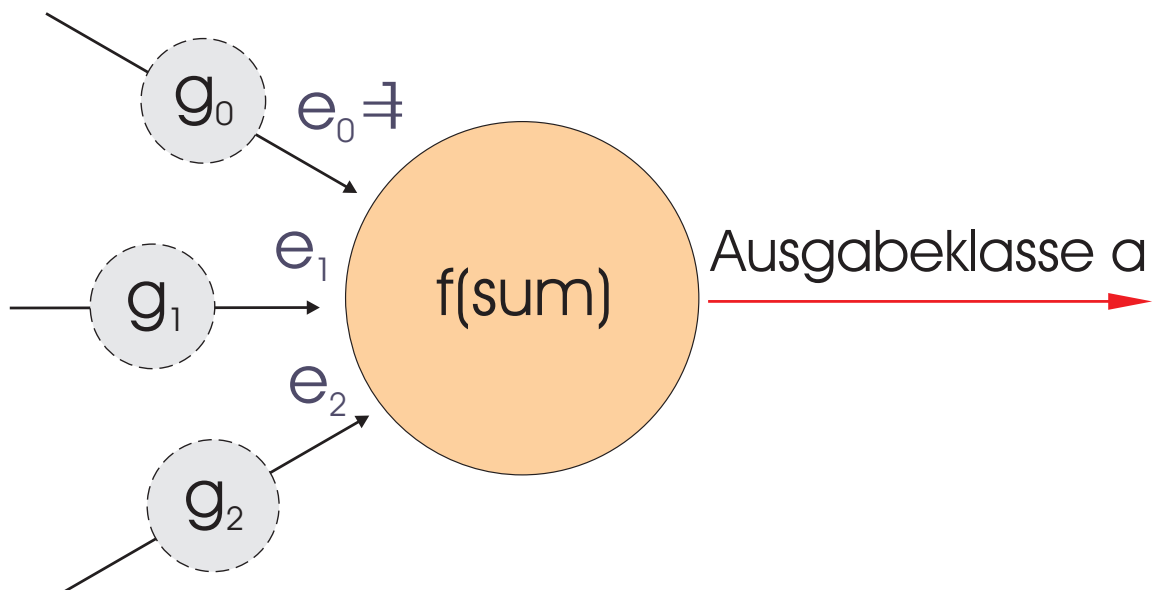
Perzeptrons können Eingabenvektoren nur **linear** (durch eine Trenngerade, eine Ebene oder eine Hyperebene) klassifizieren

Welches *neuronale* Modell teilt die Eingabepaare (e_1, e_2) den 2 Klassen zu?

(z.B. **Einkommen**, **Beruf** bzw. kreditwürdig, nicht kreditwürdig)

8.31 Modell für zwei Dimensionen und Klassen

Architektur



Transferfunktion

$$\text{sum} = g_0 + g_1 e_1 + g_2 e_2$$

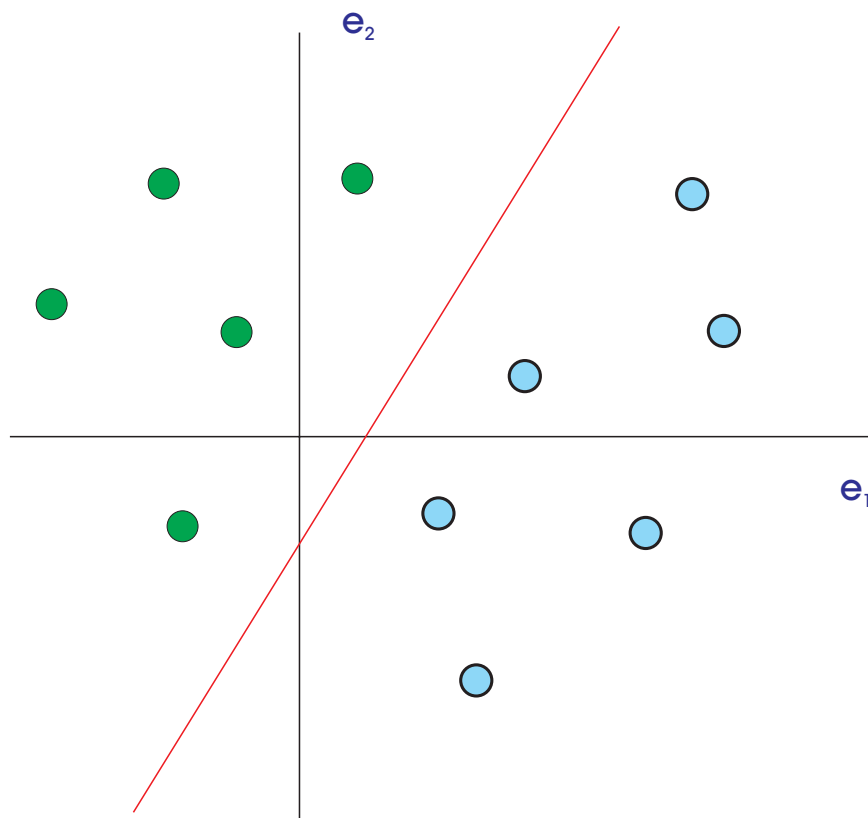
$$a = f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} \geq 0 \\ -1 & \text{sonst} \end{cases}$$

Lernalgorithmus

Verallgemeinerung des Lernalgorithmus für das *e*indimensionale Perzeptron auf den *zwei*dimensionalen Fall

$\text{sum} = g_0 + g_1 e_1 + g_2 e_2 = 0$ trennt zwischen den Klassen $+1$ und -1 und ist eine Umformung der Geradengleichung $y = ax + b$

Geometrische und algebraische Interpretation



Zwei Formen der Geradengleichung

① $\text{sum} = g_0 + g_1 e_1 + g_2 e_2 = 0$

Für die grünen Objekte gilt $\text{sum} \geq 0$

Für die blauen Objekte gilt $\text{sum} < 0$

Steigung a und Schnittpunkt b der

Trenngerade $y = ax + b$?

$$\Rightarrow g_2 e_2 = - (g_0 + g_1 e_1) \Rightarrow$$

② $e_2 = (-g_1/g_2) e_1 + -g_0/g_2$

Zweidimensionales Zweiklassen-Perzeptron

① Gegeben

Lernmenge	von Tripeln (e_1, e_2, k) , wobei e_1, e_2 reelle Eingabeskalare und k tatsächliche binäre Klasse der Eingaben
Gewichte	g_0, g_1, g_2
Nettoeingabe	$\text{sum} = g_0 + g_1 e_1 + g_2 e_2$
Berechnete Klasse	$a = f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} \geq 0 \\ -1 & \text{sonst} \end{cases}$
Lernrate	α

② Gesucht

Gewichte, die neue Paare (e_1, e_2) der Klasse -1 oder +1 zuordnen

③ Lernalgorithmus

Initialisiere die Gewichte g_0, g_1 und g_2 beliebig

Berechne alle Ausgaben: $a := f(\text{sum})$

GEHE WIEDERHOLT DURCH ALLE Tripel (e_1, e_2, k) der Lernmenge

FALLS a die Eingaben e_1 und e_2 fehlklassifiziert

$$g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$$

$$g_2 := g_2 + \alpha \cdot e_2 \cdot (k - a)$$

Berechne alle Ausgaben a neu

BIS berechnete Klassen $a =$ tatsächliche Klassen k

Satz

Der Algorithmus terminiert mit *dieser Lernregel* bei hinreichend kleinem α (gute Wahl zwischen 0 und 1)

8.33 \$ ZWEIDIMPERZEPTRON - Problem

Teilen Sie Zahlenpaare (e_1, e_2) nach dem Muster der folgenden Tabelle in zwei Klassen. e_1 und e_2 sind die unabhängigen Variablen, k ist die abhängige. Gegeben seien ausserdem die Startgewichte $g_0 = -0.8$, $g_1 = 0.2$ und $g_2 = 0.2$.

e_1	e_2	k
0	7	-1
-3	2	-1
0	-2	-1
-2	5	-1
0,5	4	-1
-0,5	-6	+1
2	-2	+1
2,5	4	+1
3	-4	+1
2,5	5	+1

Erstellen Sie ein Programm, das die Aufgabe und ihre Lösung in einem **Streudiagramm** visualisiert. Die Paare (e_1, e_2) sind die Punkte des Diagramms. Eine **Trenngerade** soll die beiden Punkteklassen trennen. Die Gerade soll sich an das Ergebnis jedes neuen Lernschritts anpassen.

- Stellen Sie die gegebenen Daten in einem **Tabellenblatt** dar.
- Implementieren Sie die **Benutzeroberfläche** (Steuerelemente und Diagrammfenster) auf dem gleichen Tabellenblatt.
- Codieren Sie die **Ereignisprozeduren** in VBA.

Neuronale Netze - Beispiel zum zweidimensionalen Zweiklassen-Perzeptron

Hilfe

e_1	e_2	k	a
0	7	-1	0
-3	2	-1	0
0	-2	-1	0
-2	5	-1	0
0,5	4	-1	0
-0,5	-6	1	0
2	-2	1	0
2,5	4	1	0
3	-4	1	0
2,5	5	1	0

g_0	g_1	g_2
-0,8	0,2	0,1

Trenngeraden-Gleichung ---

$$\text{sum} = g_0 + g_1 e_1 + g_2 e_2 = 0 \Rightarrow$$

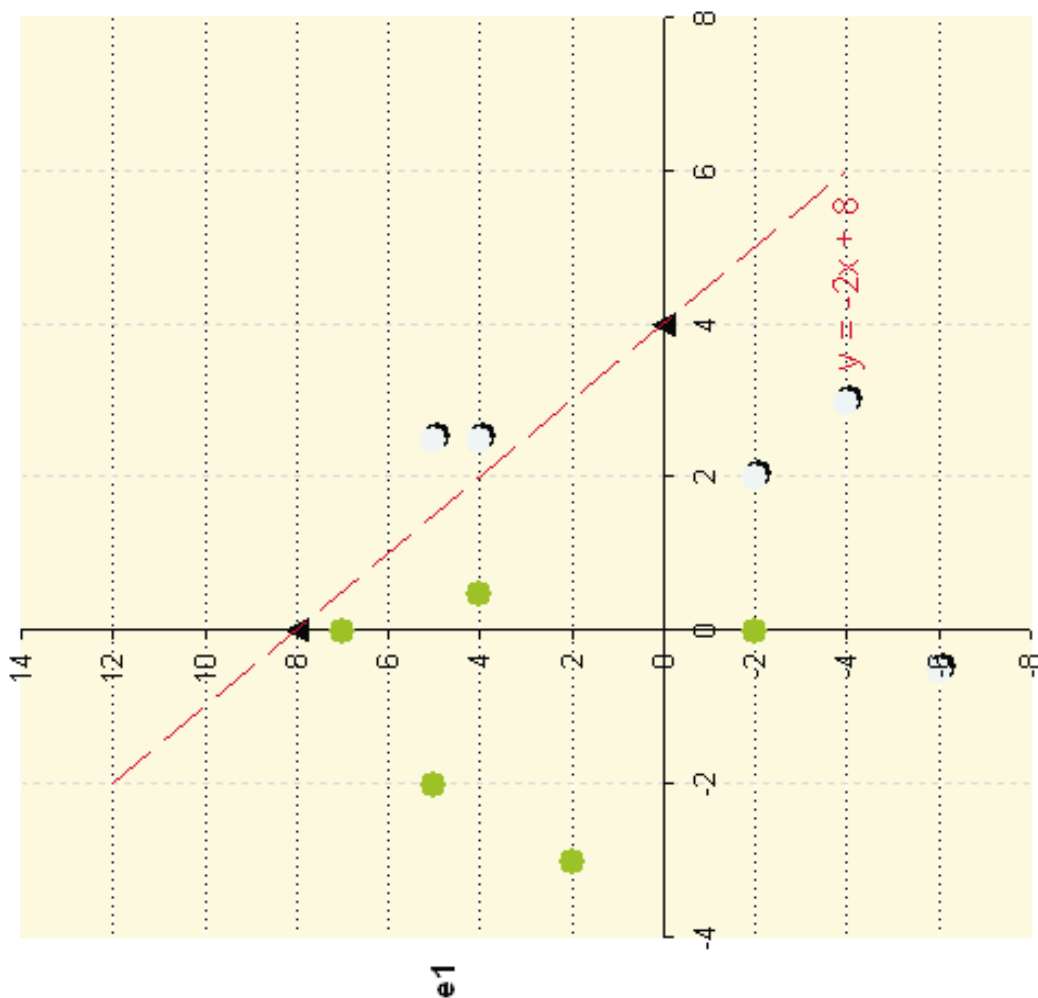
$$e_2 = -g_0/g_2 - (g_1/g_2) e_1$$

$$e_1 = -g_0/g_1 - (g_2/g_1) e_2$$

Trenngeraden-Schnittpunkte ▲

e_1	e_2	
0	8	(vgl. Diagramm)
4	0	(vgl. Diagramm)

Lernschritte : 0



- e_1, e_2 der Klasse -1
- e_1, e_2 der Klasse +1
- ▲ Schnittpunkte
- Trenngerade

Lernschritt

Reinitialisiere

Neuronale Netze - Beispiel zum zweidimensionalen Zweiklassen-Perzeptron

Hilfe

e_1	e_2	k	a
0	7	-1	-1
-3	2	-1	-1
0	-2	-1	-1
-2	5	-1	-1
0,5	4	-1	-1
-0,5	-6	1	1
2	-2	1	1
2,5	4	1	1
3	-4	1	1
2,5	5	1	1

g_0	g_1	g_2
-1,8	4,9	-0,8

Trenngeraden-Gleichung ---

$$\text{sum} = g_0 + g_1 e_1 + g_2 e_2 = 0 \Rightarrow$$

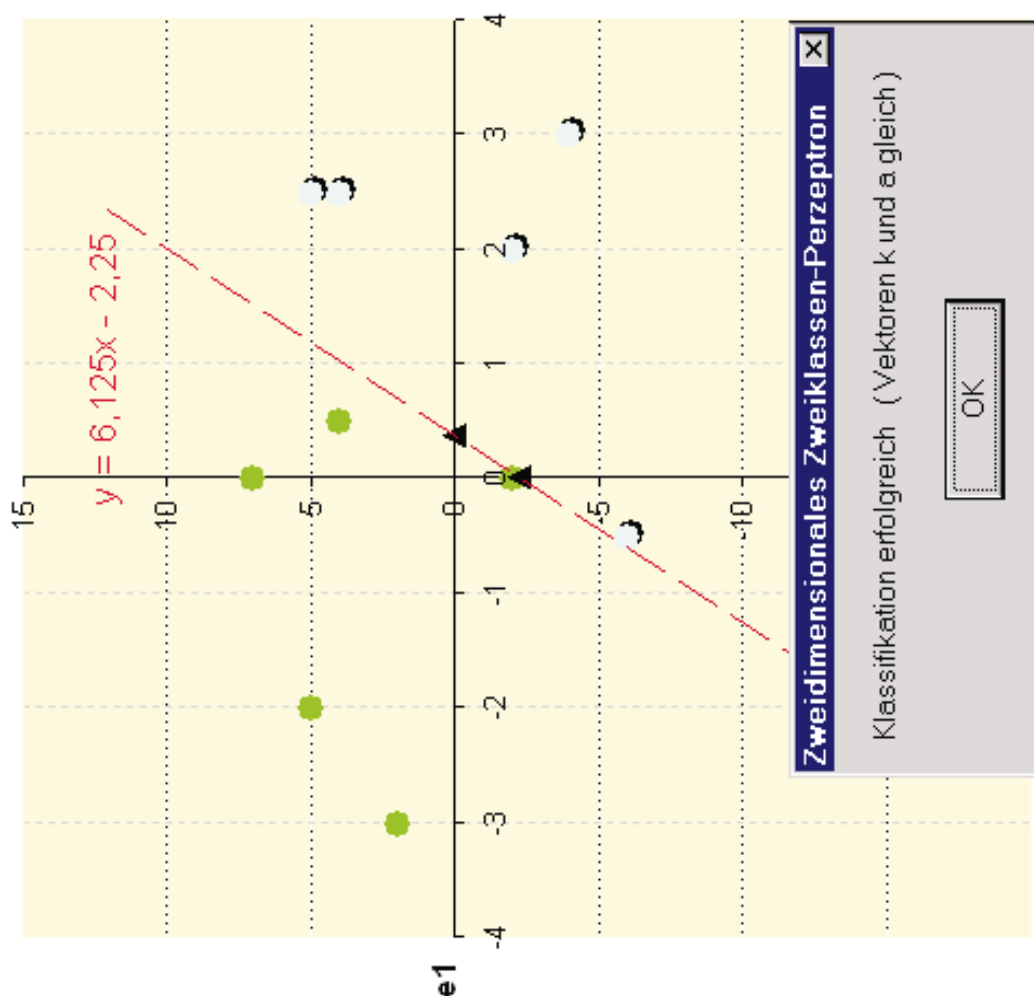
$$e_2 = -g_0/g_2 = -(g_1/g_2) e_1$$

$$e_1 = \frac{-g_0/g_1}{-(g_2/g_1)e_2}$$

Trenngeraden-Schnittpunkte ▲

e_1	e_2	
0	-2,25	(vgl. Diagramm)
0,3673	0	(vgl. Diagramm)

Lernschritte: 24



e2

e1,e2 der Klasse -1 ▲ Schnittpunkte

— Trenngerade

Lernschritt

Reinitialisiere

\$ f Ereignisse

Ereignisprozedur := Prozedur, die durch ein Ereignis (oft einen Klick auf einen **Schaltknopf**) aufgerufen wird

Steuerelemente von ZWEIDIMPERZEPTRON.XLS

Reinitialisiere

Initialisiere g_0 , g_1 und g_2 mit den Startwerten

Lernschritt

Gehe *einmal* durch alle Lerntripel (e_1, e_2, k)



Codiere die Ereignisprozedur Lernschritt

\$ Eine Ereignisprozedur

Initialisiere die Gewichte g_0 , g_1 und g_2 beliebig

Berechne alle Ausgaben: $a := f(\text{sum})$

GEHE WIEDERHOLT DURCH ALLE Tripel (e_1, e_2, k) der Lernmenge

FALLS a die laufende Eingaben e_1 und e_2 fehlklassifiziert

$$g_0 := g_0 + \alpha \cdot 1 \cdot (k - a)$$

$$g_1 := g_1 + \alpha \cdot e_1 \cdot (k - a)$$

$$g_2 := g_2 + \alpha \cdot e_2 \cdot (k - a)$$

Berechne alle Ausgaben a neu

BIS berechnete Klassen a = tatsächliche Klassen k

Sub Lernschritt_nachKlick()

Dim **i** As Integer `Index des lf. Eingabetripels

Dim **Sum** As Single `Nettoeingabe

`Gehe einmal durch alle Tripel (e_1, e_2, k) der Lernmenge
`-----

For i = 1 **To** Zahl_der_Lerntripel

 `Ändere Gewichte, falls Eingabe i fehlklassifiziert

If a.Cells(i) <> k.Cells(i) **Then**

$g_0 = g_0 + \alpha \cdot (k.Cells(i) - a.Cells(i))$

$g_1 = g_1 + \alpha \cdot e_1.Cells(i) \cdot (k.Cells(i) - a.Cells(i))$

$g_2 = g_2 + \alpha \cdot e_2.Cells(i) \cdot (k.Cells(i) - a.Cells(i))$

End If

Next i

 `Ausgaben nachführen

For i = 1 **To** Zahl_der_Lerntripel

 Sum = $g_0 + g_1 \cdot e_1.Cells(i) + g_2 \cdot e_2.Cells(i)$

If Sum >= 0 **Then** a.Cells(i) = 1 **Else** a.Cells(i) = -1

Next

End Sub

ZWEIDIMPERZEPTRON mit VBA (A 8.3)

Zuerst lernen Sie die Implementation eines zweidimensionalen Zweiklassen-Perzeptrons aus der Benutzersicht kennen, dann analysieren, vervollständigen und modifizieren Sie den Programmcode aus der Sicht des Entwicklers.

Laden Sie [ZweidimPerzeptron.xls](#). Der Programmcode ist geschützt. Sie können deshalb nur die Benutzeroberfläche betrachten.

- Wenn Sie den Cursor auf ein Toolbar-Symbol positionieren, erscheint eine Kurzbeschreibung des Symbols.
- Ausführliche Hilfe erhalten Sie auf einem Menüpunkt mit *Shift/F1*.
- Wenn Sie den Cursor auf die orange Zelle “Hilfe” bewegen, erscheint eine Anleitung zum Problem “Anzeigenplanung”. Für Details bewegen Sie den Cursor über Zellen mit ▼.

Oberfläche kennen lernen

Bewegen Sie den Cursor auf die Zelle “Hilfe” und lesen Sie die Anleitung. Versuchen Sie dann, die Aufgaben der Benutzeroberfläche zu erraten, ohne gleich mit den Schaltflächen zu experimentieren. Wenn nötig, bewegen Sie den Cursor auf die Zellenkommentare (▼).

- a) Unterscheiden Sie zwischen den folgenden Zellbereichen: Daten, Benutzereingaben, Programmausgaben und Kommentar.
- b) Welche Aufgaben erfüllen die Schaltflächen *Lernschritt* und *Reinitialisiere*?
- c) Prüfen Sie den Kommentar zur Trenngeradengleichung. Verifizieren Sie insbesondere die Ableitung der beiden expliziten Geradengleichungen aus der impliziten Gleichung ($\text{sum} = \dots$).
- d) Interpretieren Sie die Geradengleichung geometrisch. Wie werden die beiden Schnittpunkte rechnerisch ermittelt.
- e) Weshalb wird in der Geradengleichung die Nettoeingabe sum mit 0 gleichgesetzt?

Programm anwenden

- f) Lernen Sie eine Gewichtematrix, indem Sie solange *Lernschritt* klicken, bis die Trenngerade zwischen den blauen und grünen Eingaben richtig trennt (das Programm die erfolgreiche Klassifikation meldet).
- g) Experimentieren Sie mit verschiedenen Eingabevektoren.
- h) Experimentieren Sie mit verschiedenen Startgewichten.

Programm ergänzen

Laden Sie die Arbeitsmappe [ZweidimPerzeptronSkelett.xls](#). Sie enthält die gleiche Benutzeroberfläche wie *ZweidimPerzeptron.xls*, der Codeteil ist aber nicht mehr geschützt und enthält Lücken.

- i) Das Programm enthält zwei Ereignisprozeduren. Welche Aufgaben erfüllen sie? Antworten Sie, ohne den Code zu konsultieren.
- j) Wechseln Sie in den VBA-Editor (Alt-F11) und suchen Sie die Ereignisprozeduren. Weshalb enden alle mit `_nachKlick`?
- k) Die Ereignisprozedur `Initialisiere_nachKlick()` soll nicht nur den Gewichtevektor `g` initialisieren, sondern auch die Ausgabespalten der Eingabe-/Ausgabetablelle setzen. Vervollständigen Sie die Initialisierungsschleife.
- l) Verfolgen Sie den Programmablauf von `Initialisiere_nachKlick()`:
 - Setzen Sie den Cursor auf `Initialisiere_nachKlick()`
 - Starten Sie den Einzelschrittmodus (F8)
 - Gehen Sie schrittweise durch die Prozedur.
- m) Vergleichen Sie `Lernschritt_nachKlick()` mit dem bereits bekannten Entwurfs- und VBA-Code (siehe unten). Beschreiben und begründen Sie die Unterschiede.
- n) Wie codieren Sie `berechneGleichTatsächlicheAusgaben()` ohne `Exit Function`?

Mehrere Eingabedimensionen

Perzeptronprobleme mit **ein oder zwei** Eingabedimensionen lassen sich einfach neuronal modellieren



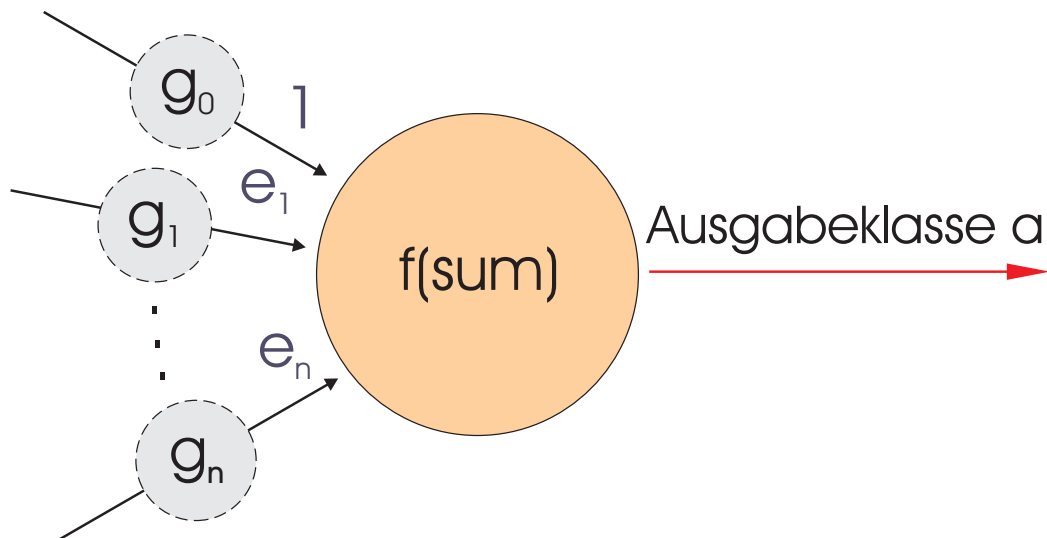
Wie lässt sich ein Perzeptronproblem mit **mehreren** Eingabedimensionen neuronal modellieren ?



- ① Graphisches **Modell**
- ② Formales **Problem**
- ③ Formaler **Lernalgorithmus**

8.36 Mehrdim. Zweiklassen-Perzeptron - Modell

Architektur



Transferfunktion

$$\text{sum} = g_0 + g_1 e_1 + \dots + g_n e_n$$

$$a = f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} \geq 0 \\ -1 & \text{sonst} \end{cases}$$

Lernalgorithmus

Verallgemeinerung der Lernalgorithmen für das ein- und zweidimensionale Perzeptron auf den mehrdimensionalen Fall mit n Eingabevariablen \Rightarrow

Mehrdim. Zweiklassen-Perzeptron - Problem

- ✓ ① Graphisches Modell
- ⇒ ② Formales Problem
- ③ Formaler Lernalgorithmus



Gegeben (Vektoren fett)

$\{ (\mathbf{e}, k), \dots \}$

Lernmenge von Paaren, wobei

$\mathbf{e} = [1, \mathbf{e}_1, \dots, \mathbf{e}_n]$

reller Eingabenvektor

k

tatsächliche Klasse (-1 oder +1)

$\mathbf{g} = [g_0, \dots, g_n]$

Initialisierungsgewichte g_i ($i = 0 \dots n$)

$\text{sum} = \mathbf{g} \cdot \mathbf{e} = \sum_{i=0}^n g_i \cdot \mathbf{e}_i$

Nettoeingabe

$f(\text{sum}) = \begin{cases} +1 & \text{falls } \text{sum} \geq 0 \\ -1 & \text{sonst} \end{cases}$

berechnete Klasse a

α

Lernrate (i.d.R. zwischen 0 und 1)

Gesucht

Gewichtevektor \mathbf{g} , der jedem neuen Eingabenvektor \mathbf{e} die Klasse +1 oder -1 zuordnet

Mehrdim. Zweiklassen-Perzeptron - Algorithmus

- ✓ ① Graphisches Modell
- ✓ ② Formales Problem
- ⇒ ③ Formaler Lernalgorithmus



Kurzbeschreibung

Gewichte beliebig initialisieren und dann systematisch ändern, bis die **berechneten** Klassen den **tatsächlichen** entsprechen

Algorithmus

Initialisiere **g** beliebig

Berechne alle Ausgaben: $\mathbf{a} := f(\text{sum})$

GEHE WIEDERHOLT DURCH ALLE Paare (**e**, **k**) der Lernmenge

FALLS **a** den laufenden Eingabenvektor **e** fehlklassifiziert

$$\mathbf{g} := \mathbf{g} + \alpha \cdot \mathbf{e} \cdot (\mathbf{k} - \mathbf{a})$$

Berechne alle Ausgaben neu: $\mathbf{a} := f(\text{sum})$

BIS Berechnete Klassen **a** = tatsächliche Klassen **k**

Sätze

- Der Algorithmus terminiert mit dieser Lernregel bei hinreichend kleinem α (Eine gute Wahl liegt zwischen 0 und 1)
- Eine lineare Transferfunktion kann zwar mehrdimensional, aber nur linear klassifizieren

Vergleich von Perzeptrons

Perzeptrons unterscheiden sich in ...

Architektur

- ✓ Zahl der Eingabevariablen
- ✓ Zahl der Ausgabeklassen

Transferfunktion

- ✓ Datentypen der Transferfunktion

8.38 Vergleich der Dimensionen

Perzeptrons unterscheiden sich in ...

Architektur

Transferfunktion

Zahl der Eingabevariablen?

Perzeptron	Eingabevariablen	sum
<i>Eindimensionales -</i>	1	$g_0 + g_1 e_1$
<i>Zwei -</i>	2	$g_0 + g_1 e_1 + g_2 e_2$
<i>Mehr -</i>	n	$g_0 + g_1 e_1 + \dots + g_n e_n$

8.38 Vergleich der Klassen

Perzeptrons unterscheiden sich in ...

Architektur

Transferfunktion

Zahl der Klassen?

- **Zweiklassen** - Perzeptrons

ordnen die Eingaben einer von *zwei* Klassen zu

- **Mehrklassen** - Perzeptrons

ordnen die Eingaben einer von *mehreren* Klassen zu

8.38 Vergleich der Datentypen

Perzeptrons unterscheiden sich in ...

Architektur

Transferfunktion

▸ Datentypen der Transferfunktion

- **binäre** Eingaben oder Ausgaben i.e.S.
[0, 1]
- **bipolare** Eingaben oder Ausgaben
[+1, -1]
- **reelle** Eingaben oder Ausgaben

Mehrere Dimensionen *und* Klassen

... am Beispiel “Binärkonversion”

① Problem

Dezimal- in Binärzahlen kodieren



② Modell

- mehrdimensional: 15 binäre Eingaben
- mehrere Klassen: 4 binäre Ausgaben



③ Lernalgorithmus

Eingaben Gewichte Ausgaben



Implementation

- Benutzeroberfläche
- Ereignisprozeduren

8.39 \$ MEHRKLASSPERZEPTRON - Problem

⇒ ① Problem

- ② Architektur
- ③ Transferfunktion
- Lernalgorithmus

Entwickeln Sie ein mehrdimensionales Mehrklassen-Perzeptron, das Dezimal- in Binärzahlen konvertiert. Zum Beispiel soll es die Dezimalzahl 5 in die Binärzahl 0101 konvertieren.

Bereiten Sie die Daten wie folgt auf:

- *Das verwendete neuronale Modell akzeptiert nur binäre Eingaben. Codieren Sie deshalb jede Dezimalzahl als binären, aber dezimal interpretierten Eingabevektor mit jeweils einer einzigen 1 und 14 Nullen. Der Binärvektor **000'0000'0001'0000** ergibt zum Beispiel dezimal interpretiert die Zahl 5, weil die fünftletzte Binärstelle gesetzt ist.*
- *Codieren Sie die binäre Ausgabe als vierstellige Binärzahl. Der dezimalen Eingabe 5 entspricht zum Beispiel die binäre Ausgabe **0101** (von rechts $2^2 + 2^0 = 5$)*
- *Gewichten Sie einen dezimal interpretierten Eingabevektor so, dass die Transferfunktion daraus die richtig Binärzahl berechnet.*



Klassenvektor statt -skalar
Gewichtematrix statt -vektor

\$ Problem - Beispiel I

Binärer **Eingabevektor** mit 15 Ziffern

000'0000'0001'0000 ist *dezimal* interpretiert 5

5. Stelle von rechts



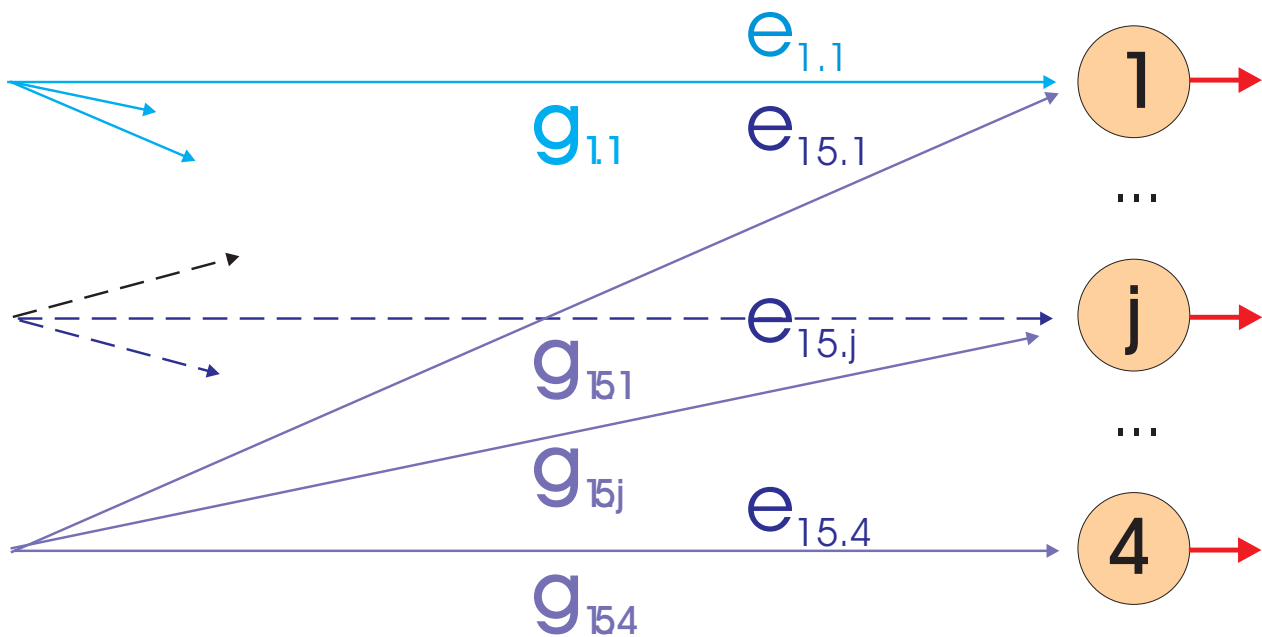
Binärer **Klassenvektor** mit 4 Ziffern

0101 ist *binär* interpretiert auch 5^1

$$2^2 + 2^0 = 5$$

Dieses mehrdimensionale Perzeptron ist komplexer, weil es 15 Eingabe- und 4 Ausgabeneuronen enthält

1 Wie lautet der Klassenvektor des Eingabevektors [000'0000'0000'0010]?



15 Binärziffern stellen die Zahl 5 **dezimal** dar :
000'0000'0001'0000

4 Binärziffern stellen die Zahl 5 **binär** dar : *0101*¹

✓ ① Problem

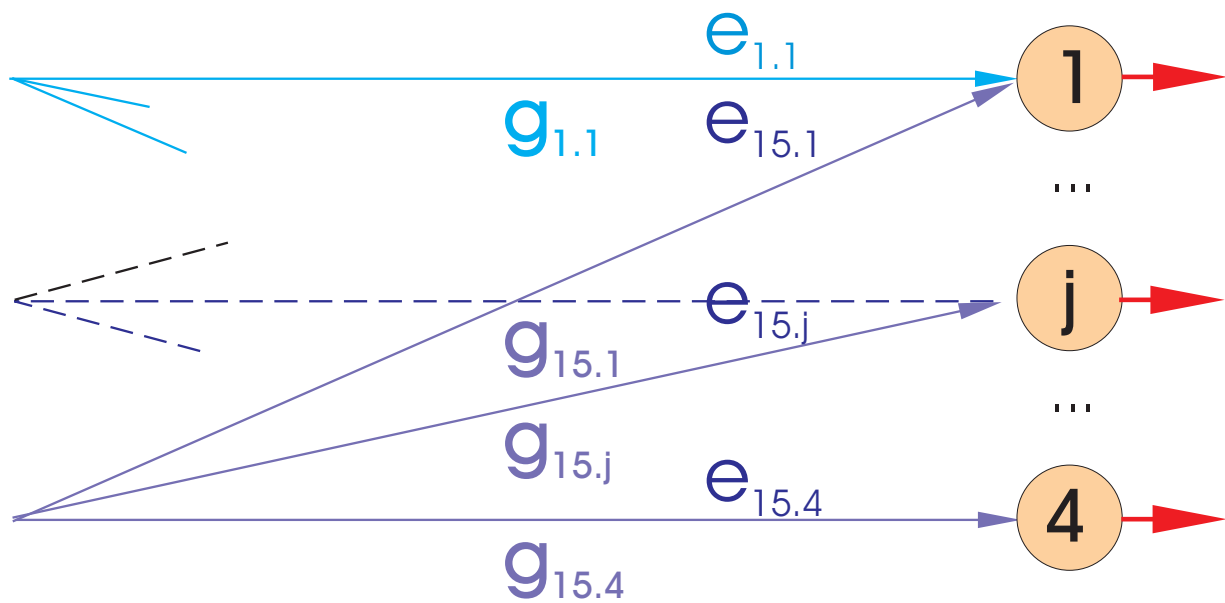
✓ ② Architektur

③ Transferfunktion

Lernalgorithmus

1 Wie viele Verbindungen existieren zwischen den Eingabe- und Ausgabeneuronen ?

8.40 \$ Problem - Beispiel II



Ziel

4stellige **berechnete** Klassenvektoren aus ...

- ✓ Lernpaaren, z.B. ($000'0000'0001'0000$, 0101) mit ...
 - 15stelligen **Eingabevektoren**
 - 4stelligen **tatsächliche** Klassenvektoren 0101
- ✓ einer mit dem Lernalgorithmus gefundenen **Gewichtematrix**

\$ f Transferfunktion

$\{ (\mathbf{e}, \mathbf{k}), \dots \}$

Lernmenge aus Paaren, wobei ...

$\mathbf{e} = [e_1, \dots, e_i, \dots, e_n]$

ein Eingabevektor

000'0000'0001'0000, $n=15$

$\mathbf{k} = [k_1, \dots, k_j, \dots, k_m]$

ein tatsächlicher Klassenvektor

0101, wobei $m = 4$

$\mathbf{a} = [a_1, \dots, a_j, \dots, a_m]$

berechneter Klassenvektor

$$\mathbf{G} = \begin{bmatrix} g_{11} & \dots & g_{1m} \\ \dots & g_{ij} & \dots \\ g_{n1} & \dots & g_{nm} \end{bmatrix}$$

Gewichtematrix

$\mathbf{g}_i = [g_{i1}, \dots, g_{in}]$

Gewichtevektor

$\text{sum}_i = \mathbf{g}_i \bullet \mathbf{e}$

Nettoeingabe von Neuron i

$\mathbf{sum} = \mathbf{e} \bullet \mathbf{G} = [\text{sum}_1, \dots, \text{sum}_m]$

Vektor der Nettoeingaben aller Ausgabeneuronen $1 \dots m$

$$f(\text{sum}_j) = \begin{cases} 1 & \text{falls } \text{sum}_j \geq 0 \\ 0 & \text{sonst} \end{cases}$$

Transferfunktion für Neuron j

$\mathbf{a} = \mathbf{f}(\mathbf{sum}) = [f(\text{sum}_1), \dots, f(\text{sum}_m)]$ berechneter Klassenvektor

Gesucht ist ...

Gewichtematrix **G**, die jedem neuen Eingabevektor **e** (binär codierte Dezimalzahl) einen Klassenvektor **a** (Binärzahl) zuordnet



Details der Gewichtematrix



Lernalgorithmus in Entwurfssprache

8.41 \$ Gewichte II

	1. Binärbit	j-tes Binärbit	4. Binärbit
1. Dezimalbit	$g_{1.1}$...	$g_{1.4}$
...	...	$g_{i.j}$...
15. Dezimalbit	$g_{15.1}$...	$g_{15.4}$

Eingabeschicht : Eingabevektor **e** [$e_1, \dots, e_i, \dots e_n$]

Ausgabeschicht : Ausgabevektor **a** [$a_1, \dots, a_j, \dots a_m$]

Beispiel

15 Binärziffern stellen die Zahl 5 dezimal dar: 000'0000'0001'0000



4 Binärziffern stellen die Zahl 5 binär dar: 0101

Kurzbeschreibung

- ⇒ Initialisiere die Gewichte beliebig und
- ⇒ ändere sie systematisch, ...
- ⇒ bis die **berechneten** Klassen den **tatsächlichen** entsprechen

Algorithmus

Initialisiere **G** beliebig

Berechne alle Ausgabevektoren: **a** := f (**sum**)

GEHE WIEDERHOLT DURCH ALLE Paare (**e**, **k**) der Lernmenge

FALLS **a** den Eingabevektor **e** fehlklassifiziert

$$\mathbf{G} := \mathbf{G} + \alpha \cdot (\mathbf{k} - \mathbf{a}) \cdot \mathbf{e}$$

Berechne alle Ausgabevektoren neu

BIS berechnete Klassen **a** = tatsächliche Klassen **k**

Satz

Der Algorithmus terminiert bei hinreichend kleinem α
(Eine gute Wahl liegt zwischen 0 und 1)

Unsere Implementationsansätze

① Tabellenblatt mit Zellenformeln

- ✓ *Eindimensionales* Perzeptron für *zwei* Klassen

② Tabellenblatt mit Ereignisprozeduren

- ✓ *Zweidimensionales* Perzeptron für *zwei* Klassen
- ⇒ *Mehrdimensionales* Perzeptron für *mehrere* Klassen



Benutzeroberfläche
Ereignisse
Ereignisprozeduren

Eingabe	Verarbeitung	Ausgabe
e i-ter Eingabevektor (i = 1 ... n)	Folienskapitel EUS Site Folienskapitel	Dezimalzahl
G Gewichtematrix	1 Initialisiere	e
a berechneter Ausgabevektor	Initialisiere	$G = G + \alpha \cdot (k - a) \cdot e$
k tatsächlicher Klassenvektor	2 Lerne	0.4 0.5 0.4 -0.7
α 0.2 nderbare Lernrate	Lernschritt	-0.8 -0.4 0.3 -0.3
n 15 Binärstellen in e_i	Lerne	-0.8 0.3 0.1 0.1
m 4 Binärstellen in a_i und k_i	3 Teste	0.3 -0.1 -0.8 0.2
	Nächstes Lernpaar	0.7 -0.4 0.3 -0.5
	Lernschritte 0	0.1 -0.3 -0.5 0.5
		-0.3 -0.5 0.4 0.5
		-0.1 -0.1 0.1 -0.2
		-0.1 0.1 0.1 0.8
		0.0 0.1 0.2 -0.4
		0.2 -0.2 -0.1 -0.5
		-0.3 -0.0 -0.2 0.4
		-0.7 0.5 0.3 -0.6
		0.2 0.2 -0.0 -0.5
		0.0 -0.4 -0.1 0.5
		1 0 0 1
		$\cdot 2^3 = \cdot 2^2 = \cdot 2^1 = \cdot 2^0 =$
		8 0 0 1
		0 1 0 0
		$\cdot 2^3 = \cdot 2^2 = \cdot 2^1 = \cdot 2^0 =$
		0 4 0 0
		9 ↕ 4

8.43 \$ Oberfläche nachher

Ausgabe

Dezimalzahl

e

$$G = G + \alpha \cdot (k - a) \cdot e$$

1

0

-0.2

-0.3

-0.1

0.7

2

0

-0.1

-0.0

0.1

-0.0

3

0

-0.8

-0.6

0.2

0.5

4

0

-0.2

0.0

-0.2

-0.0

5

0

-0.0

0.1

-0.1

0.3

6

0

-0.1

0.1

0.1

-0.1

7

0

-0.2

0.1

0.6

0.0

8

0

0.1

-0.1

-0.2

-0.4

9

0

0.2

-0.0

-0.0

0.2

10

0

0.2

-0.2

0.2

-0.3

11

0

0.1

-0.1

0.5

0.1

12

0

0.0

0.1

-0.1

-0.2

13

0

0.7

0.3

-0.1

0.3

14

0

0.4

0.0

0.1

-0.1

15

1

0.7

0.3

0.2

0.2

$$a = f(e \cdot G) =$$

$$[f(g_1 e), \dots, f(g_4 e)]$$

1

1

1

1

$$\cdot 2^3 = \cdot 2^2 = \cdot 2^1 = \cdot 2^0 =$$

8

4

2

1

15

k tatsächliche Dualzahl von **e** bzw. des Dezimalwerts

1

1

1

1

$$\cdot 2^3 = \cdot 2^2 = \cdot 2^1 = \cdot 2^0 =$$

8

4

2

1

15

↕

\$ Vier Ereignisse

Ereignisprozeduren := Prozeduren, die durch ein Ereignis (oft einen Klick auf eine **Schaltfläche**) aufgerufen werden

Initialisiere

Initialisiere die Gewichtematrix **G** beliebig

Lerne

Ändere **G** solange, bis für alle **e** gilt : **a** = **k**

Lernschritt

Ändere **G** für ein einziges **e** einmal

Nächste Zahl

Erhöhe den Dezimalwert um 1



Codiere Ereignisprozeduren mit den **Namen**
Initialisiere_nachKlick, **Lernschritt_nachKlick**,
Lerne_nachKlick, **nächsteZahl_nachKlick**

\$ Ereignisprozedur **Lerne_nachKlick**

GEHE WIEDERHOLT DURCH ALLE Paare (**e**, **k**) der Lernmenge
FALLS **a** den laufenden Eingabevektor **e** fehlklassifiziert

$$\mathbf{G} := \mathbf{G} + \alpha \cdot (\mathbf{k} - \mathbf{a}) \cdot \mathbf{e}$$

Berechne alle Ausgabevektoren neu

BIS berechnete Klassen **a** = tatsächliche Klassen **k**

```
Sub Lerne_nachKlick()  
  `Eingabenindex, Ausgaben- bzw. Klassenindex  
  Dim i As Integer, j As Integer  
  Do  
    `Zufallszahl zwischen 1 und n wählen  
    Dezimalwert = Int(Rnd() * emax) + 1  
    `Gewichtematrix ändern  
    For j = 1 To m  
      For i = 1 To n  
        G.Cells(i,j) = G.Cells(i,j) +  
          alpha*e.Cells(i) * (k.Cells(j) - a.Cells(j))  
      Next  
    Next  
    Lernschritte = Lernschritte + 1  
  Loop Until berechneteGleichTatsächlicheAusgaben()  
End Sub
```

- 1 Weshalb kann die Gewichtsrechnung in der innersten Schleife anders als im Entwurfscod - auch *unbedingt* sein?
- 2 Weshalb enthält *diese* Prozedur - anders als der Entwurf - keine explizite Neuberechnung der Ausgabevektoren?

MEHRKLASSPERZEPTRON mit VBA (A 8.4)

Zuerst lernen Sie eine Implementation der Binärkonversion aus der Benutzersicht kennen, dann analysieren und vervollständigen Sie den Programmcode aus der Sicht des Entwicklers.

Laden Sie [MehrklassPerzeptron.xls](#). Der Programmcode ist geschützt. Sie können deshalb nur die Benutzeroberfläche betrachten.

- Wenn Sie den Cursor auf ein Toolbar-Symbol positionieren, erscheint eine Kurzbeschreibung des Symbols.
- Ausführliche Hilfe erhalten Sie auf einem Menüpunkt mit *Shift/F1*.
- Wenn Sie den Cursor auf die orange Zelle “Hilfe” bewegen, erscheint eine Anleitung zum Problem “Anzeigenplanung”. Für Details bewegen Sie den Cursor über Zellen mit ▼.

Oberfläche kennen lernen

Bewegen Sie den Cursor auf die Zelle “Hilfe” und lesen Sie die Bedienungsanleitung. Versuchen Sie dann, die Aufgaben der Benutzeroberfläche zu erraten, ohne gleich mit den Schaltflächen zu experimentieren. Wenn nötig, bewegen Sie den Cursor auf die Zellenkommentare (▼).

- a) Welchen Sinn macht die Unterteilung in *Eingabe*, *Verarbeitung*, *Ausgabe*?
- b) Welche Aufgaben erfüllen die Schaltflächen *Initialisiere*, *Lernschritt*, *Lerne* und *Nächste Zahl*?
- c) Was bedeuten die farbigen Bereiche im Teil *Ausgabe* ?
- d) Unterscheiden Sie zwischen jenen Zellen, die das Programm ändert und jenen, welche die Benutzerin direkt modifiziert.

Programm anwenden

Sie lernen eine neue Gewichtematrix auf zwei Arten:

- Sie klicken für einen bestimmten Eingabevektor (siehe Schaltfläche *Nächste Zahl*) solange *Lernschritt*, bis die Gewichtematrix den Ausgabevektor aus genau diesem Eingabevektor richtig berechnet.
 - Sie klicken einmal *Lerne*, um eine Gewichtematrix zu lernen, die aus allen 15 möglichen Eingabevektoren die Ausgabevektoren richtig berechnet.
- e) Die Schaltfläche *Lernschritt* optimiert die Gewichtematrix für eine einzige Dezimalzahl. Durch wiederholtes Klicken können Sie die Binärkonversion genau dieser Zahl lernen. *Nächste Zahl* geht zur nächsten Zahl.

Experimentieren Sie mit der Schaltfläche *Lernschritt* und verschiedenen Eingabevektoren (Schaltfläche *Nächste Zahl*).

- f) Die Schaltfläche *Lerne* lernt so lange, bis das Abbruchkriterium $\mathbf{a} = \mathbf{k}$ für alle e erfüllt ist. Die jeweils mit dem Abbruchkriterium zu testende Dezimalzahl wird markiert. Die Markierung schreitet bis zu jener Dezimalzahl voran, die von der laufenden Gewichtematrix gerade nicht mehr konvertiert werden kann. Dies kann mehrere Male die gleiche Zahl sein, weil die neuen Gewichte nicht jedesmal zu einer besseren Konversion führen müssen.

Experimentieren Sie mit der Schaltfläche *Lerne*.

Programm testen

Sie testen die gelernte Gewichtematrix, indem Sie sich mit der Schaltfläche *Nächste Zahl* durch die erlaubten Eingabewerte von 1 bis 15 bewegen.

- g) Überprüfen Sie das Modell für alle Eingabevektoren.
- h) Rechnen Sie einen Ausgabevektor von Hand nach.
- i) Ändern Sie ausgewählte Gewichte von Hand.

Programm ergänzen

Laden Sie die Arbeitsmappe [MehrklassPerzeptronSkelett.xls](#). Sie enthält die gleiche Benutzeroberfläche wie MehrklassPerzeptron.xls, der Codeteil ist aber nicht mehr geschützt und enthält Lücken.

- j) Schliessen Sie aus der Benutzeroberfläche und dem Programmablauf, welche Ereignisprozeduren der Code enthält. Welche Aufgaben erfüllen sie? Antworten Sie, ohne gleich den Programmcode zu konsultieren.
- k) Wechseln Sie in den VBA-Editor (Alt-F11) und suchen Sie die Ereignisprozeduren. Weshalb enden alle mit `_nachKlick`?
- l) Die Ereignisprozedur `Initialisiere_nachKlick()` soll die Gewichtematrix `G` zufällig initialisieren. Vervollständigen Sie das Skelett anhand der folgenden Information:

Die Programmzeile `G.Cells(i, j) = Rnd() - Rnd()` initialisiert ein einzelnes Matrixelement (`Rnd()` ergibt jedesmal eine neue Zufallszahl zwischen 0 und 1). Programmieren Sie die Initialisierung der Matrix mit einer geschachtelten Zählschleife. Betrachten Sie zur Veranschaulichung die gelb unterlegte Gewichtematrix, nachdem Sie die Schaltfläche *Initialisiere* gedrückt haben.

- m) Beschreiben Sie Zweck und Ablauf von `berechneVektoren_e_k_a` am Ende von `Initialisiere_nachKlick()`? Die Prozedur besteht aus der Generierung der Eingaben **e** und **k** sowie der Berechnung der Ausgaben **a** aus der laufenden Gewichtematrix. Den *generierten* Eingabe-, Ausgabe- und Klassenvektoren entsprachen bisher die *vorgegebenen* Lernelemente der Stichprobe.
- n) Verfolgen Sie den Programmablauf von `Initialisiere_nachKlick()`:
- Setzen Sie den Cursor auf `Initialisiere_nachKlick()`
 - Starten Sie den Einzelschrittmodus (F8)

- Gehen Sie schrittweise durch die Prozedur, (1) ohne aufgerufene Unterprogramme zu betreten (Umschalt/F8) und (2), indem Sie auch `berechneVektoren_e_k_a` verfolgen.
- o) Vergleichen Sie `Lernschritt_nachKlick()` mit dem bereits bekannten Entwurfscod (siehe unten).
- p) Vervollständigen Sie die Prozedur `Lerne_nachKlick()` nach dem folgenden Lernalgorithmus. Was nicht kursiv steht, wird von anderen Prozeduren erledigt oder ist Teil des Skeletts.

Initialisiere **G** beliebig

Berechne alle Ausgabevektoren: **a** := f (**sum**)

GEHE WIEDERHOLT DURCH ALLE Paare (**e**, **k**) der Lernmenge

*FALLS **a** den laufenden Eingabevektor **e** fehlklassifiziert*

*Berechne die Gewichtematrix neu: **G** := **G** + $\alpha \cdot (\mathbf{k} - \mathbf{a}) \cdot \mathbf{e}$*

Berechne alle Ausgabevektoren neu: **a** := f (**sum**)

BIS berechnete Klassen **a** = tatsächliche Klassen **k**

Komplexere neuronale Netze

Einfache Perzeptrons lösen zwar *praktische* Aufgaben wie

- ✓ einfache automatische Zeichenerkennung (▷OCR)
- ✓ lineare ▷Klassifikation

Sie lösen aber bestimmte Aufgaben *nicht* ...

⇒ zum Beispiel nichtlineare Klassifikationsaufgaben

⇒ ...



Neuronale Modelle mit einer **komplexeren** ...

- ⇒ Architektur
- ⇒ Transferfunktion
- ⇒ Lernregel

Komplexere Modelle enthalten komplexere . . .

⇒ **Architekturen**, z.B.

- mehr Neuronen
- komplexere Datentypen der Eingabe- und Ausgabevariablen
- mehr Schichten
- mehr als eine Gewichtungsschicht
- auch rückwärtsgerichtete Verbindungen

⇒ **Transferfunktionen**, z.B.

- auch monotone nichtlineare Transferfunktionen
- mehr als einen Transferfunktionstyp

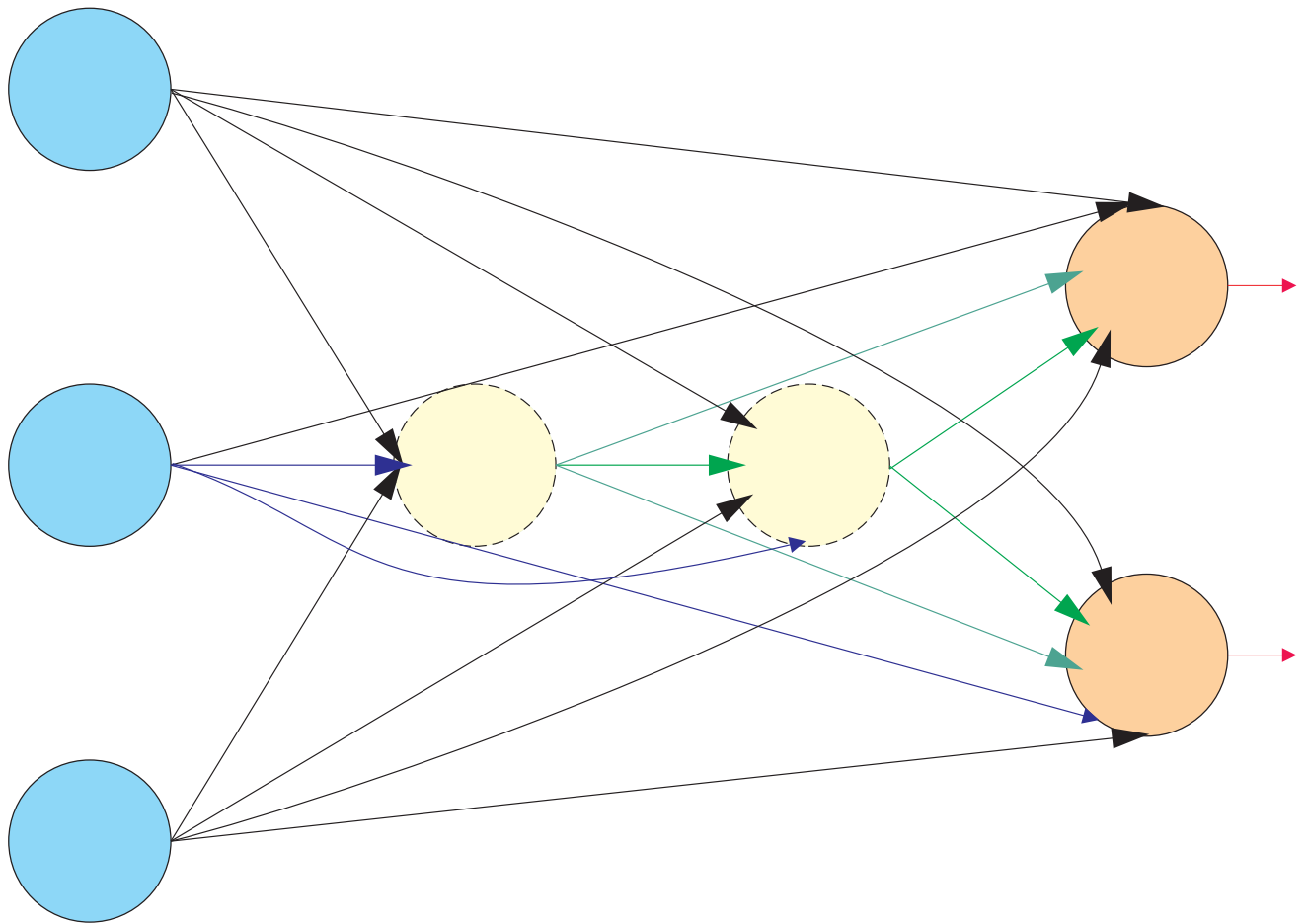
⇒ **Lernregeln**, z.B.

- die Backpropagationsregel



Ein komplexeres neuronales Modell ist zum Beispiel das vorwärtsgerichtete mehrstufige Netz mit **Backpropagation**

8.54 CCN - Architektur von *Predict*



3 Eingabe-
neuronen 2 hinzugefügte Neuronen in
2 verborgenen Schichten 2 Ausgabe-
neuronen

Der Lernalgorithmus von *Predict* passt die CCN-Architektur selbständig an die Lernmenge an. *Predict* eignet sich deshalb für viele Anwendungen

Die Anpassung verläuft kaskadenweise
(ein verborgenes Neuron nach dem anderen)



CCN (Cascade Correlation Neural Network)

Architektur

Neuronen

- Zahl der Eingabe- und Ausgabeneuronen benutzerdefiniert
- Zahl und Anordnung der verborgenden Neuronen lernbar

Verbindungen

- Vorwärtsgerichtet und direkt zwischen allen Neuronen
- Verbindungen zwischen Eingabe- und verborgenen Neuronen nach der ersten Anpassung fix

Transferfunktion

- je nach Anwendung linear oder nichtlinear

Lernalgorithmus

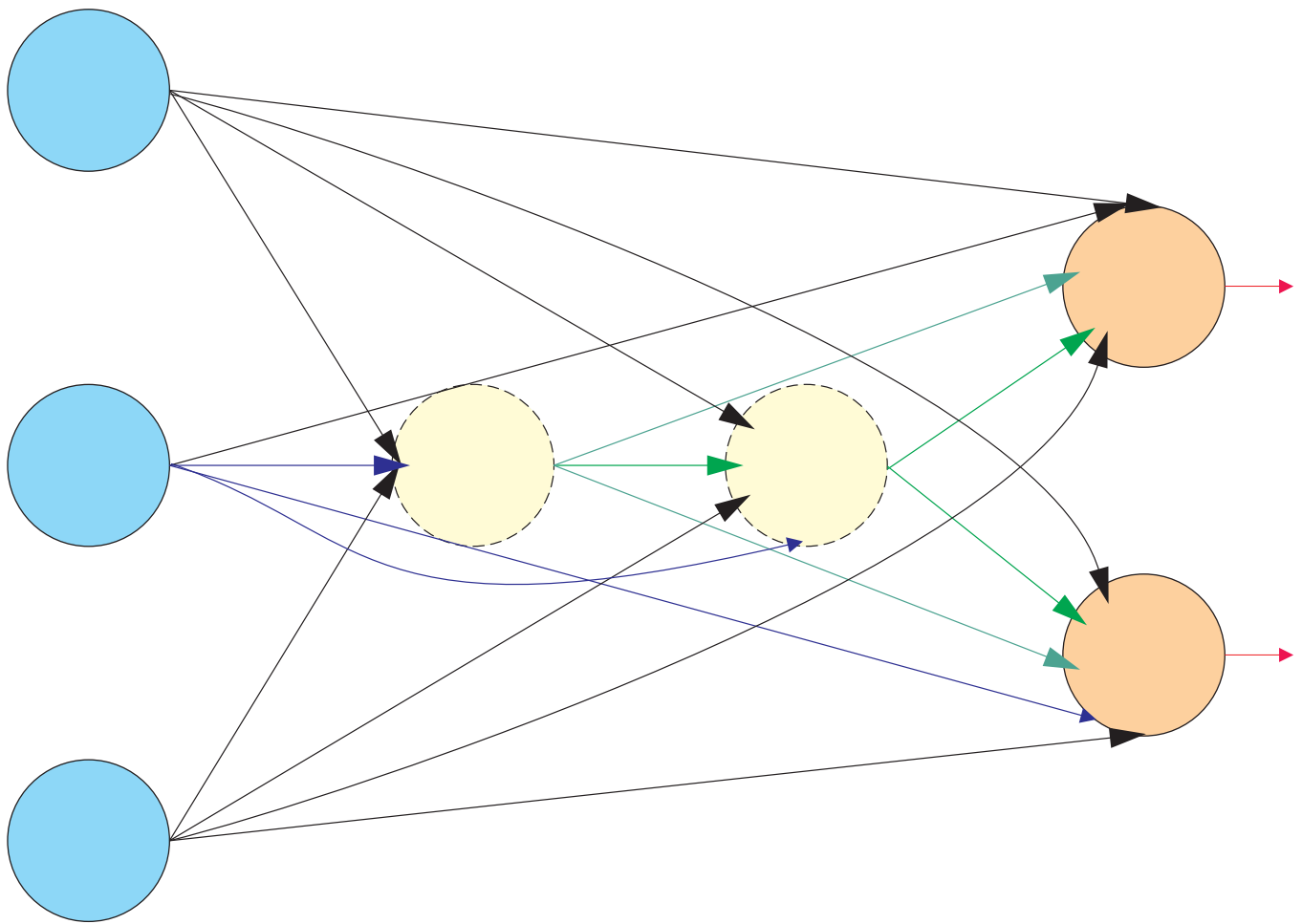
Lernregeln von *Predict*

- schnelle Variante der Fehlerrückführung
- Kalman--Lernregel

Abbruchkriterium von *Predict*

- geringe Änderungen der Genauigkeit in der Testmenge

8.54 CCN - Lernalgorithmus



3 Eingabe- 2 hinzugefügte Neuronen in 2 Ausgabe-
neuronen 2 verborgenen Schichten neuronen

Trainiere ein einstufiges Netzwerk

SOLANGE sich der Fehler noch verringern lässt

Füge ein **verborgenes Neuron** hinzu

Gewichte zwischen **Eingaben** und **Zusatzneuron**

Gewichte zwischen **Zusatzneuron** und Ausgabeneuronen

Gewichte zwischen verborgenen und Ausgabeneuronen

CCN - Einige Vorteile

- ✓ Dynamische Architektur **flexibler**
- ✓ Kein vollständiges Training bei **neuer Information**
- ✓ Lernen **schnell**
(z.B. im Vergleich mit Backpropagation)
- ✓ **Parallelisierbar**
(Zusatzneuronen interagieren seitlich nicht)

CCN - Eine Prognoseanwendung

Problem (Yamamoto/Zenios 1993)

Vorhersage vorzeitiger Hypotheken-Rückzahlungen

Lernmenge

10-jährige Zeitreihen für je einen Zinssatz

a) **Eingaben**

- Alter der Hypothek in Monaten (normalisiert reell)
- laufender Monat im Jahr (normalisiert reell)
- Aussenstände für gesamten Hypotheken-Pool (normalisiert reell)
- Hypothekarzins / Refinanzierungszins (tatsächlich)

b) **Tatsächlich Ausgabe**

vorzeitige Rückzahlungsraten

Modell

① **Architektur**

14 bis 18 verborgene Neuronen

② **Transferfunktionen**

lineare und nichtlineare (Ergebnisse vergleichbar)

③ **Lernalgorithmus**

Abbruch nach ca. 3000 Durchgängen (ca. 10 Sun Sparc-Minuten)

Fehlerrückführung - Ein mächtigeres Modell

Grundlagen

✓ Neuron	<u>7</u>
✓ Netz	<u>10</u>
✓ Lernen	<u>13</u>

Anwendung

✓ Anwendungsklassen	<u>31</u>
✓ Entwicklung mit <i>Predict</i> 📌	<u>40</u>

Einblick in die Theorie

✓ Transferfunktion und Schwellenwert	<u>82</u>
✓ Perzeptrons	<u>90</u>
⇒ Neuronales Lernen mit Fehlerrückführung	<u>145</u>

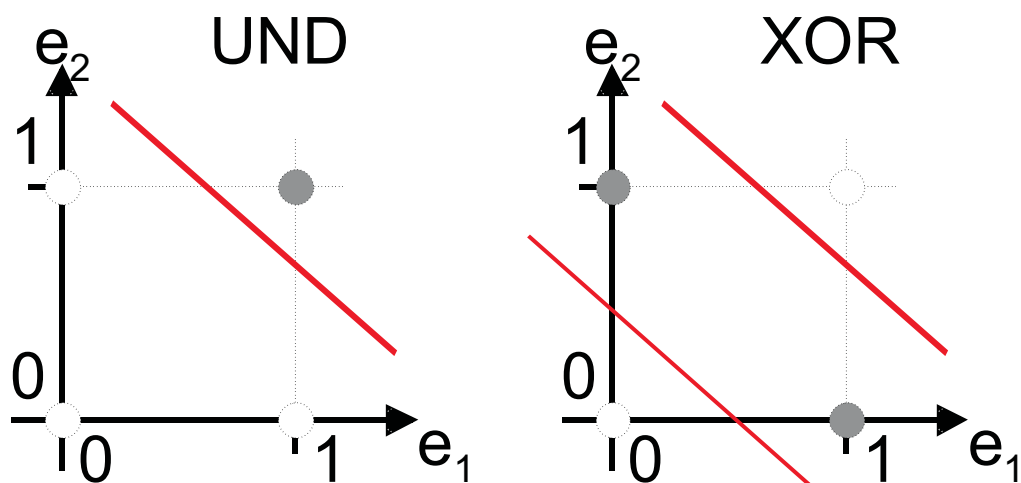
Weiter führen ...

- FAUSETT, L., Fundamentals of Neural Networks, Prentice Hall 1994 (Architekturen und Algorithmen)
- BIGUS, J., Data Mining with Neural Networks, McGraw-Hill 1996 (Betriebliche Anwendungen)

8.45 Ausschiessendes ODER

Aussage e_1		Aussage e_2	Aussage a_3	
1	UND	1	\Rightarrow	1
0	UND	0		0
1	UND	0		0
0	UND	1		0

Aussage e_1		Aussage e_2	Aussage a_3	
1	XOR	1	\Rightarrow	0
0	XOR	0		0
1	XOR	0		1
0	XOR	1		1



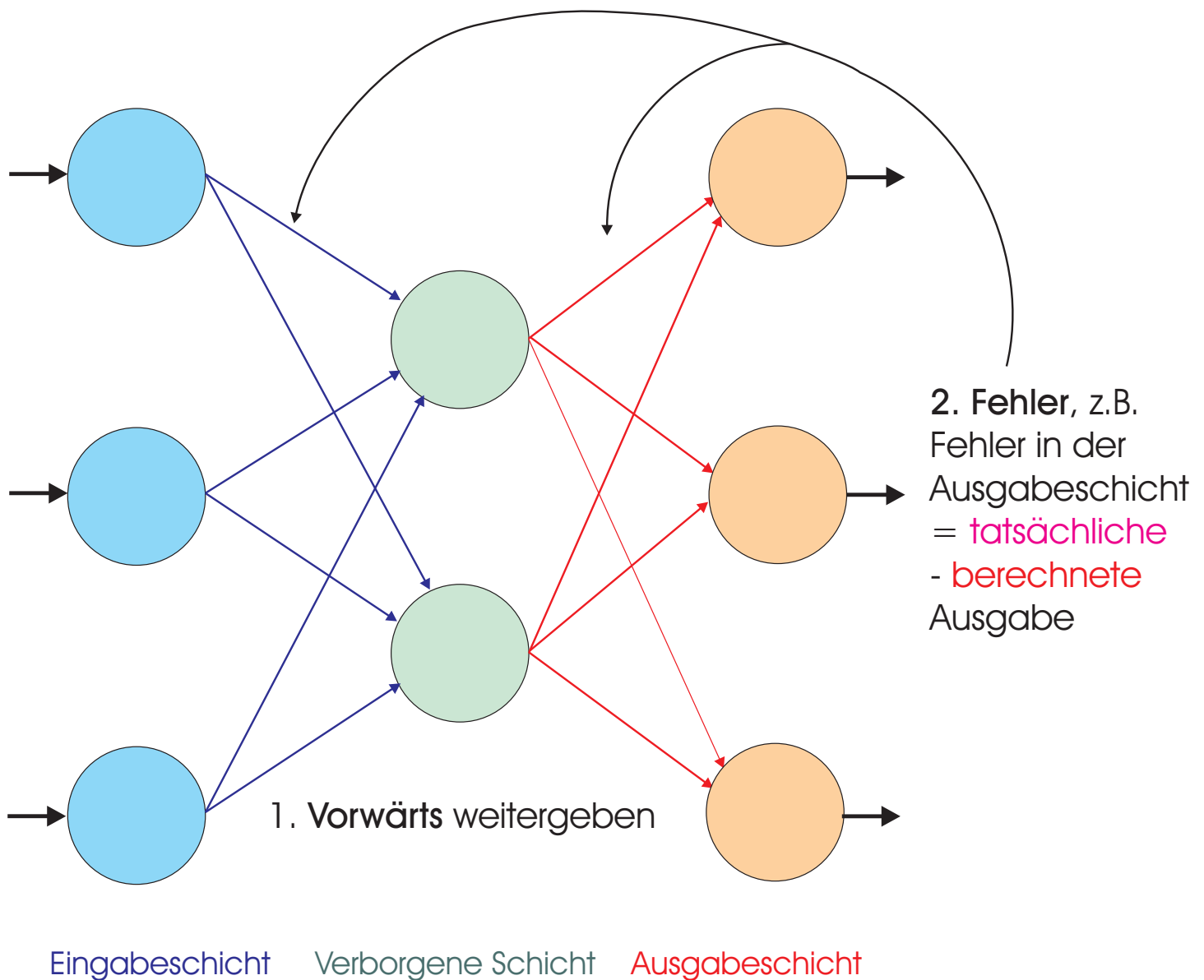
[Demonstration zu SPSS Neural Connection](#)
 (Contents - The Multi-Layer Perceptron (Doppelklick))

8.47 Ein- und mehrstufiges Perzeptron

	<i>Einstufiges Perzeptron</i>	<i>Mehrstufiges Perzeptron</i>
<i>Synonyme</i>	Perzeptron i.e.S.	Backpropagations-Netz, Fehlerrückführungs-Netz engl. Multi-Layer Perceptron
<i>Funktionalität</i>	Lineare Klassifikation	Nichtlineare Klassifikation
<i>Architektur</i>	einstufig vorwärtsgerichtet	mehrstufig vorwärtsgerichtet
<i>Transferfunktion</i>	Treppenfunktion oder linear	nichtlinear
<i>Lernalgorithmus</i>		Backpropagation
<i>Abbruchkriterium</i>	berechnete = tatsächliche Ausgaben	Minimum des Klassifikationsfehlers
<i>Verständlichkeit</i>	einfach	Basisvariante relativ einfach
<i>Rechenbedarf</i>	klein	gross

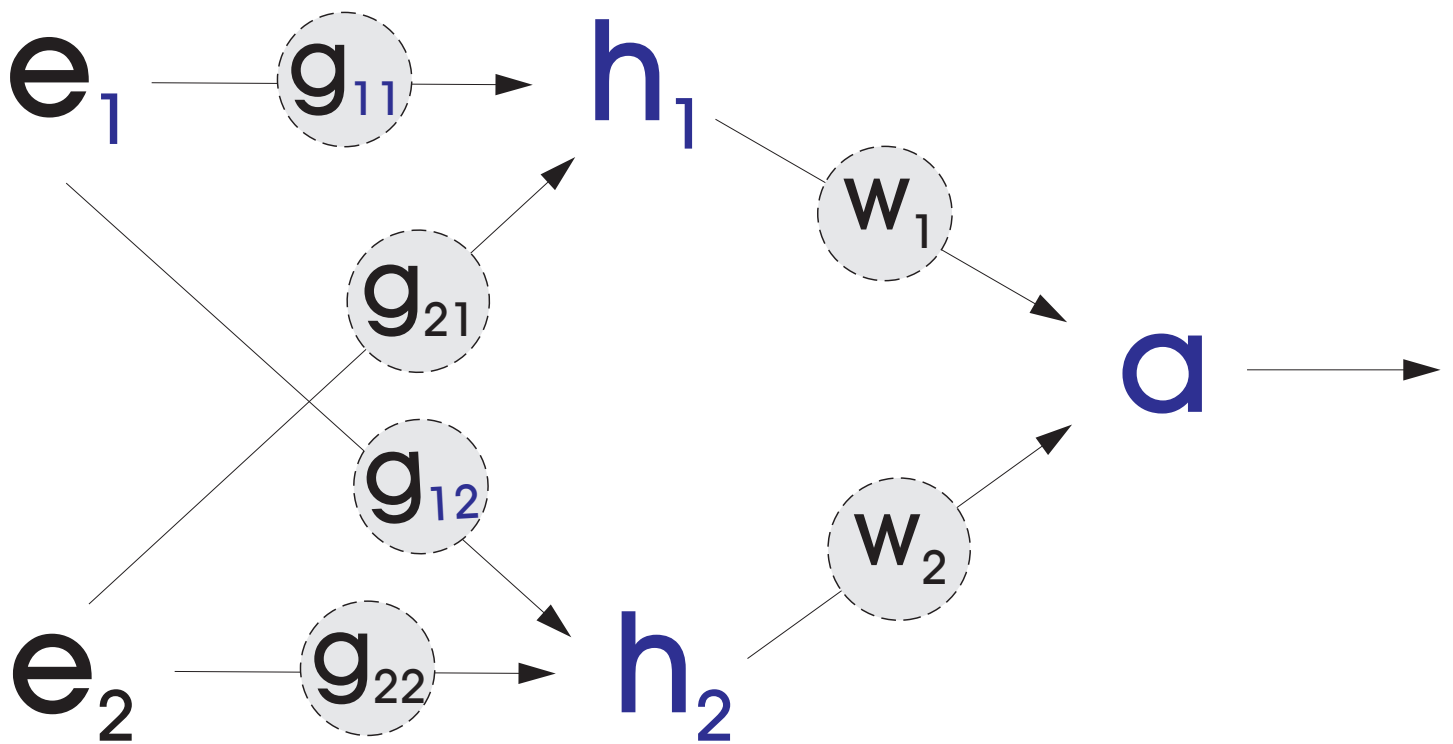
8.46 \$ MEHRSTUFPERZEPTRON - ② Architektur

3. Gewichte anpassen (Fehlerrückführung)



Eingabeschicht	Eingabevektor e
Verborgene Schichten	verborgene Vektoren h
Ausgabeschicht	Ausgabevektor a

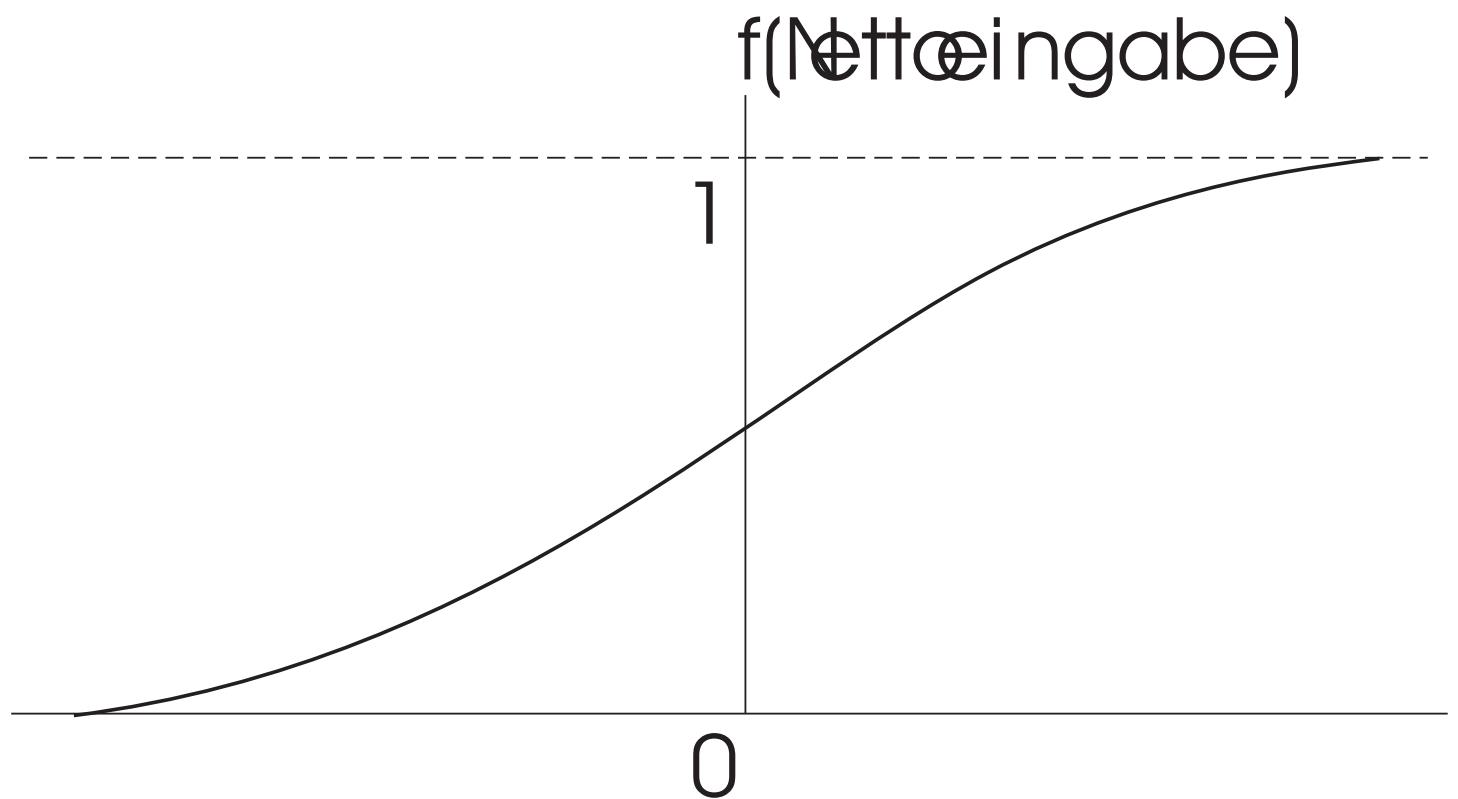
(Möglich ist auch mehr als eine verborgene Schicht)



\$ ② Architektur symbolisch

$\{ (\mathbf{e}, k), \dots \}$	gegebene Lernpaare (\mathbf{e}, k) , wobei ...
$[\mathbf{e}_1, \mathbf{e}_2]$	binärer Eingabevektor \mathbf{e}
k	tatsächliche binäre Klasse
$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$	Gewichtematrix \mathbf{G} der ersten Stufe
$\mathbf{G} \cdot \mathbf{e} = [\text{hsum}_1, \text{hsum}_2]$	Nettoeingaben-Vektor \mathbf{hsum} eines verborgenen Neurons
$\frac{1}{1 + e^{-\gamma \text{Nettoeingabe}}}$	binäre Sigmoidfunktion f
γ	Steigungsparameter
e	Basis des natürlichen Logarithmus \ln
$\mathbf{f}(\mathbf{hsum}) = [f(\text{hsum}_1), f(\text{hsum}_2)]$	verborgener Ausgabevektor \mathbf{h}
$[w_1, w_2]$	Gewichtevektor \mathbf{w} der zweiten Stufe
$\mathbf{w} \cdot \mathbf{h}$	Nettoeingabe-Skalar asum von a
$f(\text{asum})$	Ausgabeskalar a

8.49 \$ f Transferfunktion



$$\frac{1}{1 + e^{-\gamma \text{ Nettoeingabe}}}$$

γ

e

binäre Sigmoidfunktion f (monoton)

Steigungsparameter

Basis des natürlichen Logarithmus \ln

8.51 \$ Benutzeroberfläche I

Problem

Initialisiere

☐ zufällige Werte
☒ laufende Werte

Hilfe

e		k
0	0	0
0	1	1
1	0	1
1	1	0

1

«

①

Nächstes Lernpaar

(Gegebene Werte kursiv)

Architektur

Steigung der Transferfunktion $\gamma = 10$

Eingabeschicht	1. Stufe	e	0.2	0.2	
	G		0.100	-0.600	②
Versteckte Schicht			0.700	0.800	Wende G an
	hSum		0.0	0.0	
	2. Stufe	h	0.0	0.0	
	w			-0.100	③
Ausgabeschicht			0.500		Wende w an
	aSum			0.0	
Tatsächliche Klasse	a			0.0	④ Vergleiche
	k			0.0	a mit k !

8.52 \$ Benutzeroberfläche II

Durchgang lernen

Lernrate $\alpha = 0.5$

Lerne einen Durchgang

Korrigiere G und w

dG

0.000	0.000
0.000	0.000

dw

0.000
0.000

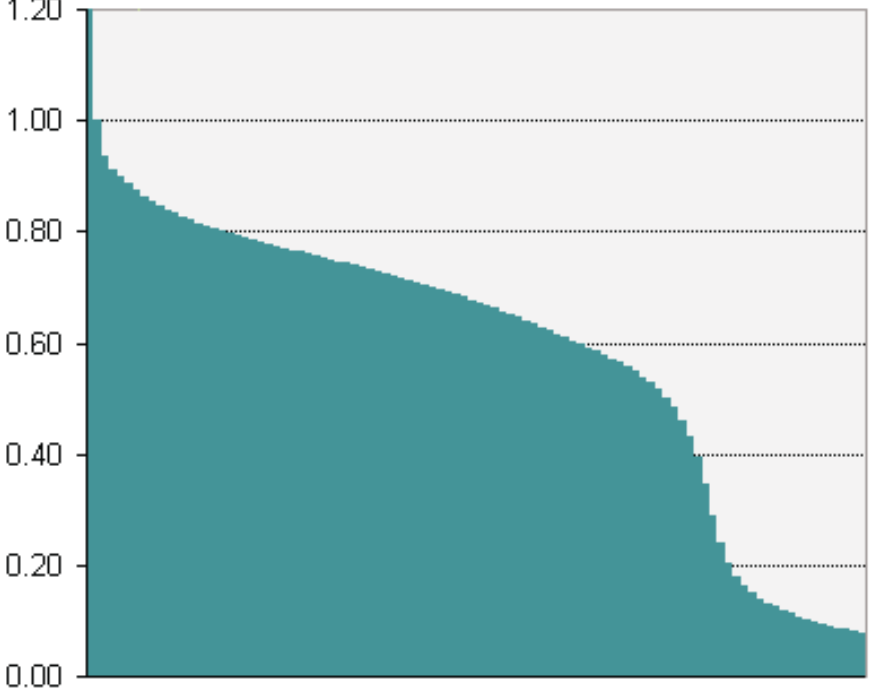
Fehlerrückführung

Mehrere Durchgänge lernen

Drücken Sie die Schaltfläche erneut, falls das Lernergebnis nicht befriedigt.

Lerne 800 Durchgänge

Klassifikationsfehler



$$F = \sum_{p=1}^4 (k_p - a_p)^2 \quad (p \text{ bezeichnet das laufende Lernpaar})$$

8.53 \$ ④ Lernalgorithmus I

Erste Annäherung

1. Berechne die Ausgaben ("Wende **G** an", "Wende **w** an")
2. Führe die Fehler zurück ("Fehlerrückführung")
3. Passe die Gewichte an ("Korrigiere **G** und **w**")

Zweite Annäherung

Initialisiere **G** und **w** beliebig

Berechne den verborgenen Vektor **h** = **f(hsum)**

Berechne die Ausgabe **a** = **f(asum)**

GEHE WIEDERHOLT DURCH die Lernmenge

 Korrigiere die Gewichtematrizen **G** und **w**

 Berechne den verborgenen Vektor **h** = **f(hsum)**

 Berechne die Ausgabe **a** = **f(asum)**

BIS Klassifikationsfehler minimal

\$ ④ Lernalgorithmus II

$$\mathbf{G} := \mathbf{G} + d\mathbf{G}$$

$$\mathbf{w} := \mathbf{w} + d\mathbf{w}.$$

Die **Korrekturmatrizen** $d\mathbf{G}$ und $d\mathbf{w}$ setzen sich analog zu \mathbf{G} und \mathbf{w} aus den Gewichten dg_{it} bzw. dw_t zusammen:

$$d\mathbf{G} = (dg_{it}) = \begin{bmatrix} dg_{11} & dg_{12} \\ dg_{21} & dg_{22} \end{bmatrix}, \text{ wobei } i = 1 \dots 2 \text{ und } t = 1 \dots 2$$

$$d\mathbf{w} = (dw_t) = [dw_1, dw_2], \text{ wobei } t = 1 \dots 2$$

Die Elemente der beiden Matrizen berechnen sich wie folgt:

$$dg_{it} = \alpha \cdot e_i \cdot dh_t$$

$$dw_t = \alpha \cdot h_t \cdot da$$


dh_t und da heissen **Fehlersignale**. Das Fehlersignal da des Ausgabeneurons a hängt vom Unterschied zwischen tatsächlicher und berechneter Klasse ab:

$$da = (k-a) \cdot a \cdot (1-a)$$


$$dh_t = w_t \cdot da \cdot h_t \cdot (1-h_t).$$

MEHRSTUFPERZEPTRON (A 8.5)

Zuerst lernen Sie eine Implementation des XOR-Problems aus der Benutzersicht kennen, dann analysieren Sie den Programmcode aus der Sicht des Entwicklers. Laden Sie dazu [MehrstufPerzeptron.xls](#).

- Wenn Sie den Cursor auf ein Toolbar-Symbol positionieren, erscheint eine Kurzbeschreibung des Symbols.
- Ausführliche Hilfe erhalten Sie auf einem Menüpunkt mit *Shift/F1*.
- Wenn Sie den Cursor auf die orange Zelle “Hilfe” bewegen, erscheint eine Anleitung zum Problem “Anzeigenplanung”. Für Details bewegen Sie den Cursor über Zellen mit .

Oberfläche kennen lernen und Netz automatisch trainieren

Bewegen Sie den Cursor auf die Zelle “Hilfe” und lesen Sie die Anleitung. Versuchen Sie dann, die Aufgaben der Benutzeroberfläche zu erraten, ohne gleich mit den Schaltflächen zu experimentieren. Wenn nötig, bewegen Sie den Cursor auf die Zellenkommentare (.

- a) Interpretieren Sie den Bereich *Problem*. Unterscheiden Sie dabei zwischen jenen Zellen, die das Programm ändert und jenen, welche die Benutzerin modifiziert. Welche Aufgaben erfüllen die Schaltflächen “Initialisiere” und “Nächstes Lernpaar”?
- b) Markieren Sie das Optionsfeld “laufende Werte”. Initialisieren Sie die dann die Gewichtematrizen mit den Startgewichten a) des Tabellenblatts “Anfangsbelegungen”.
- c) Klicken Sie auf die Schaltfläche *Lerne 800 Durchgänge*.

Die aufgerufene Ereignisprozedur geht 800-mal durch die Lernmenge und stellt den Lernfortschritt grafisch dar. Das Diagramm stellt den Klassifikationsfehler (Y-Achse) als Funktion der Lernschritte (X-Achse) dar. Interpretieren Sie das Diagramm.

- d) Interpretieren Sie den Bereich *Architektur*. Unterscheiden Sie dabei wiederum zwischen den Zellen, die das Programm ändert und jenen, welche die Benutzerin modifiziert.

- e) Durch fortgesetztes Drücken der folgenden Schaltflächen können Sie prüfen, ob das Netz die Aufgabe gelernt hat: “Nächstes Lernpaar”, “Wende G an”, “Wende w an”. Wenn “Lerne 800 Durchgänge” erfolgreich war, müssen das berechnete a und das tatsächliche k übereinstimmen. Prüfen Sie alle vier Lernpaare der Wahrheitstafel.
- f) Experimentieren Sie mit unterschiedlichen Startgewichten und Konstanten (Am besten kopieren Sie hintereinander die Konfigurationen a) bis f) des Tabellenblatts “Anfangsbelegungen”).

Netz manuell trainieren

Mit den Schaltflächen “Erstes Lernpaar”, “Nächstes Lernpaar”, “Fehlerrückführung” und “Korrigiere Gewichte” können Sie die Lernschritte einzeln verfolgen:

- g) Wählen Sie mit den Schaltflächen “Erstes Lernpaar” und “Nächstes Lernpaar” ein Lernpaar.
- h) Berechnen Sie die Ausgabe des Netzes, indem Sie die Gewichte **G** und **w** anwenden.
- i) Klicken Sie die Schaltfläche “Fehlerrückführung”. Die Korrekturen werden berechnet und zu den Korrekturmatrizen addiert.
- j) Führen Sie die Schritte g bis i für alle vier Lernpaare durch. Klicken Sie danach die Schaltfläche “Korrigiere Gewichte”, um die Gewichtematrizen zu korrigieren und die Korrekturmatrizen wieder auf Null zurück zu setzen. (Die Schaltfläche “Lerne Durchgang” führt Schritt j automatisch durch)

Ausgewählte Ergebnisse manuell nachvollziehen

- k) Wiederholen Sie die Schritte b und c. Füllen Sie dann mit dem Ergebnis die folgende Tabelle aus:

Lernpaar p	e_p	a_p
1	(0, 0)
2	(0, 1)
3	(1, 0)
4	(1, 1)

- Wie gross ist der Wert der Fehlerfunktion F für alle Lernpaare?
- Wie gross ist F , wenn die Klassifikation korrekt ist? Wie gross ist F im ungünstigsten Fall?
- Nehmen Sie die Fehlerfunktion als Abbruchkriterium des Lernprozesses. Wie klein muss F mindestens werden, damit das Netz die Eingabevektoren korrekt klassifiziert.

Programm nachvollziehen

Der Code von [MehrstufPerzeptron.xls](#) ist komplexer als jener der einstufigen Perzeptrons. Wir beschränken uns deshalb auf den *Nachvollzug* wichtiger Prozeduren.

- l) Gelangen Sie mit Alt/F11 von der Benutzeroberfläche in den Programmmeditor. Rufen Sie dann mit Ctrl/R das Projekt-Explorer-Fenster auf.
 - Gewinnen Sie einen Überblick über die Objekte und Module der Arbeitsmappe.
 - Wozu dienen die Klassenmodule (Ausführliche Erläuterungen finden Sie in [MehrstufPerzeptronKlassen.pdf](#)). Klären Sie zuerst ab, wo der Code der Benutzeroberfläche Instanzen der Klassenmodule deklariert, und schauen Sie sich dann den Code der Klassenmodule an.
- m) Gewinnen Sie einen Überblick über das Excel-Objekt "Benutzeroberfläche". Gehen Sie ins Codefenster und beschreiben Sie die Prozeduren, die im Listenfeld "(Allgemein)" am Kopf des Codefensters erscheinen.

n) Beschreiben Sie die folgenden Prozeduren entwurfssprachlich:
LerneEinenDurchgang_Click, Lerne800Durchgänge_Click, Fehlerückführung_Click, Fehler().

Clusteranalyse - Problem

Begriff

Individuen aufgrund von Ähnlichkeiten unbekannten Gruppen zuordnen, ohne dass die Gruppenzugehörigkeiten bekannt sind

Bsp. *Kunden aufgrund ihres Kaufverhaltens gruppieren*

Gegeben

Marktkörbe von Kunden eines grossen Lebensmittelladens
(Wieviel hat jeder Kunde in den 10 Abteilungen gekauft?)

Gesucht

Gruppen von Kunden, deren Kaufverhalten ähnlich ist

Ansatz

unüberwachtes Lernen mit einem Kohonen-Netzwerk

Werkzeug

SPSS *Neural Connection*

[Demonstration zu SPSS Neural Connection](#)(*Contents - Demonstrations - Customer Clustering* (Doppelklick))

Clusteranalyse - Methoden

Konventionell statistische -

Hierarchische -

- ✓ Stichprobe klein
- ✓ Clusterzahl nicht erforderlich
- ✓ Variablen ›nominal-, ›ordinal- oder ›intervallskaliert
- ✓ flexibler als K-Means-Clusteranalyse

K-Means -

- ✓ Stichprobe gross
- ✓ Clusterzahl erforderlich
- ✓ Variablen ›intervallskaliert
- ✓ effizienter als hierarchische -

Neuronale -

- ✓ Stichprobe gross
- ✓ Clusterzahl nicht erforderlich
- ✓ nichtlinear

Überlappung

Regressionsanalyse

Diskriminanzanalyse

Clusteranalyse

Hauptkomponentenanalys

...

Die Methodenwahl ist oft kontrovers !

<i>Indikation</i>	<i>Geeignete Methoden</i>
Schlecht spezifizierte nichtlineare Zusammenhänge	Neuronale Netze
Explizite Verteilungsannahmen nicht erforderlich	Neuronale Netze , ›Nichtparametrische Stat.
Kleine Stichproben möglich	Konventionelle Statistik
Stichprobenfehler schätzbar	›Inferenzstatistik
Konfirmatorisch (hypothesentestend)	Konventionelle Statistik, OLAP
Explorativ (entdeckend)	Neuronale Netze ›Regelinduktion, Explorative Datenanalyse
Ergebnisse gut erklärbar	Regelsysteme, OLAP
Kausalanalysen leicht möglich	Konventionelle Statistik, ›Regelinduktion
Symbolische Daten gut geeignet	Regelsysteme, OLAP
Fehlertoleranz, ›Robustheit	Neuronale Netze
<i>Kontraindikation</i>	<i>Methoden</i>
Datenaufbereitung aufwendig	Neuronale Netze
Methodenwissen wichtig	Konventionelle Statistik
Domänenwissen wichtig	OLAP, Regelsysteme
Benutzerinterventionen häufig	OLAP
Entwicklung aufwendig	Regelsysteme

Klassische Informatik und Neuroinformatik

	<i>Klassische Informatik</i>	<i>Neuroinformatik</i>
<i>Verarbeitung</i>	sequentiell	v.a. parallel
<i>Daten</i>	v.a. diskret	v.a. kontinuierlich
<i>Lösungssuche</i>	algorithmisch	heuristisch
<i>Methodenkern</i>	Logik / Algebra	Analysis / Geometrie
<i>Lernen</i>	Programmierung	Iteratives Lernen
<i>Fehlertoleranz</i>	nein	ja

[Web Quiz](#)

Folienverzeichnis (Ein Klick führt zur gewünschten Folie)

Neuronales Lernen	2
Einordnung	3
Unterrichtsmaterial	4
Grundlagen	5
Neuronales Lernen im Betrieb	6
8.2 Biologische Nervenzellen	7
Comparaison n'est pas raison	8
8.2 Mustererkennung als neuronales Lernen	9
8.3 Aufbau eines neuronalen Netzes	10
8.4 Künstliches Neuron	11
Künstliches neuronale Netz	12
8.5 Methodischer Ansatz neuronaler Verfahren	13
Neuronales Lernen	14
Lernarten	15
📌 DIRECT MAILING - Begriff	16
📌 Ziel ist die Verbesserung der Antwortrate	17
📌 Prädiktoren und Kriterien	18
📌 Ein erster Werkzeugkontakt	19
📌 Ein reduziertes Ergebnis	20
UND - ① Problem	21
② Datenaufbereitung	22
③ Modell	23
8.6 Modell eines einfachen neuronalen Netzes	24

8.7	Modellkomponente Architektur	25
	Modellkomponente Transferfunktion f	26
	Modellkomponente Lernalgorithmus	27
	Forschungsfragen	28
	Praxis - Dutzende bis Tausende von Neuronen	29
8.8	Einstufige Architektur	30
	Mehrstufige Architektur	31
8.9	3stufige vorwärtsgerichtete 3-2-3-2-Architektur	32
	Zusammenfassung	33
	Anwendung	34
8.10	Wichtige Anwendungsklassen	35
	Anwendungsklasse <i>Vorhersage</i>	36
	Physische und Verhaltensmodelle	37
	📌 Betriebliche Anwendungen	38
	📌 Ein betriebliches Anwendungsproblem	39
	📌 Ein einfaches Problemlösungsmodell	40
	📌 Ein Werkzeug - <i>NeuralWorks Predict</i>	41
8.11	📌 BONITÄTSKLASSIFIKATION - Fallbeispiel	42
8.12	📌 Lernen und Anwenden	43
	📌 Entwicklungsphasen	44
	📌 ① Problem spezifizieren	45
	② Lern- und Testdaten sammeln	46
8.13	Daten sammeln - Repräsentativität	47
	Daten sammeln - Stichprobengrösse	48

📌 Daten sammeln	49
📌 Lern - und Testmenge	50
③ Daten aufbereiten - Messqualität	51
③ Daten aufbereiten - Skalierung / Codierung	52
Vorhersagevariablen wählen	53
Variablen	54
📌 Variablenbeispiele	55
Netzmodell spezifizieren und lernen	56
8.14 Modell von <i>Predict</i>	57
8.15 📌 Modellparameter	58
8.15 📌 Modellparameter II	59
📌 Tatsächliche und berechnete Klassen	60
8.17 Modell validieren - Overfitting	61
Modell validieren - Ein Beispiel	62
Modell validieren - Validität messen	63
8.17 📌 Klassifikationsvalidität	64
📌 Klassifikation und Vorhersage	65
BONITÄTSVORHERSAGE mit <i>Predict</i> (A 8.1)	66
Tatsächliche und vorhergesagte Ratenzahl	69
Grafisches Validitätsmass	70
Vorhersagevalidität	71
Funktionalität von <i>Predict</i>	72
Modellieren heisst oft Experimentieren	73
📌 ZEITREIHENANALYSE - Begriffe	74

📌 Ziel ist die Wechselkursprognose	75
📌 Ergebnisvariante ①	76
📌 Ergebnisvariante ②	77
Zur Mächtigkeit neuronaler Netze	78
8.56 Methode im Vergleich	79
Theoretischer Hintergrund	80
Einblick in die Theorie	81
8.18 Transferfunktion	82
Addierende Transferfunktion	83
Nettoeingabe als Skalarprodukt	84
8.19 Beispiele additiver Transferfunktionen	85
Schwellenkonstante	86
8.20 Schwellenkonstante im Zusammenhang	87
Ausgabe und Schwellenkonstante	88
Klassifikation	89
Klassisches Perzeptron	90
Wie klassifizieren Perzeptrons?	91
Beispiele und Aufgaben zu Perzeptrons	92
8.22 EINDIMPERZEPTRON - ① Problem	93
8.23 , Modell	94
8.24 ③ Ein Lernalgorithmus	95
Lernregeln	96
Tabellenblatt	97
8.25 Aufbau des Tabellenblatts	98

8.26	Ergebnis des Tabellenblatts	99
8.27	Ausgewählte Formeln	100
	Lernregel vereinfachen	101
8.28	Formel für q_0 in Zelle J9	102
	Zusammenfassung	103
	EINDIMPERZEPTRON mit <i>Excel</i> (A 8.2)	104
	Abbruchkriterien sind nicht einheitlich	106
8.29	Zwei Dimensionen und zwei Klassen	107
8.30	Trenngerade für 2 Dimensionen und Klassen	108
8.31	Modell für zwei Dimensionen und Klassen	109
	Geometrische und algebraische Interpretation	110
	Zweidimensionales Zweiklassen-Perzeptron	111
8.33	ZWEIDIMPERZEPTRON - ① Problem	112
8.34	② Oberfläche vorher	113
8.35	② Oberfläche nachher	114
	③ Ereignisse	115
	Eine Ereignisprozedur	116
	ZWEIDIMPERZEPTRON mit <i>VBA</i> (A 8.3)	117
	Mehrere Eingabedimensionen	119
8.36	Mehrdim. Zweiklassen-Perzeptron - Modell	120
	Mehrdim. Zweiklassen-Perzeptron - Problem	121
	Mehrdim. Zweiklassen-Perzeptron - Algorithmus	122
	Vergleich von Perzeptrons	123
8.38	Vergleich der Dimensionen	124

8.38 Vergleich der Klassen	125
8.38 Vergleich der Datentypen	126
Mehrere Dimensionen <i>und</i> Klassen	127
8.39 MEHRKLASSPERZEPTRON - ① Problem	128
Problem - Beispiel I	129
8.40 ② Architektur grafisch	130
8.40 Problem - Beispiel II	131
③ Transferfunktion	132
Gewichte I	133
8.41 Gewichte II	134
8.42 Lernalgorithmus	135
Unsere Implementationsansätze	136
8.42 Oberfläche vorher	137
8.43 Oberfläche nachher	138
Ereignisse	139
Eine Ereignisprozedur	140
MEHRKLASSPERZEPTRON mit <i>VBA</i> (A 8.4)	141
Komplexere neuronale Netze	145
Komplexere Modelle enthalten komplexere . . .	146
8.54 CCN - Architektur von <i>Predict</i>	147
8.55 CCN - Cascade Correlation Neural Network	148
8.54 CCN - Lernalgorithmus	149
CCN - Einige Vorteile	150
CCN - Eine Prognoseanwendung	151

Fehlerrückführung - Ein mächtigeres Modell	152
8.45 Ausschliessendes ODER	153
8.47 Ein- und mehrstufiges Perzeptron	154
8.46 MEHRSTUFPERZEPTRON - ② Architektur	155
8.48 ② Architektur speziell	156
② Architektur symbolisch	157
8.49 ③ Transferfunktion	158
8.51 Benutzeroberfläche I	159
8.52 Benutzeroberfläche II	160
8.53 ④ Lernalgorithmus I	161
④ Lernalgorithmus II	162
MEHRSTUFPERZEPTRON (A 8.5)	163
Clusteranalyse - Problem	167
Clusteranalyse - Methoden	168
Neuronales und klassisches statistisches Lernen	169
Die Methodenwahl ist oft kontrovers !	170
Klassische Informatik und Neuroinformatik	171