

Hochschule Wismar

Fachbereich Wirtschaft

Projektdokumentation Gruppe 5

Datenanalyse zum Kündigungsverhalten von Stromabnehmern

Projektarbeit im Rahmen der Lehrveranstaltung

Wissensextraktion mittels Neuronaler Netze (SS 2004)

an der Hochschule Wismar

Name	Matrikelnr.	Studiennr.
Jens Jahnke	103212	WI01
Jörn Seidel	103632	WI02
Sebastian Straus	102439	WI02
Florian Scholz	102481	WI02

Wismar, den 15. Juni 2004

Inhaltsverzeichnis

1	Vorwort	5
1.1	Ausgangssituation	5
1.2	Szenario	5
2	Vorbereitung	6
2.1	Datenvorverarbeitung	6
2.2	Software	6
3	Durchführung	7
3.1	Datenvorverarbeitung	7
3.1.1	Sortierung	7
3.1.2	Fehlwerte	7
3.1.3	Erstellen der Pattern - Dateien	8
3.2	Netze	10
3.2.1	Entwicklung und Test	10
3.2.2	Das Problem	11
3.2.3	Die Lösung	12
4	Auswertung	14
4.1	Ergebnisse	14
4.2	Interpretation und Bewertung	15
5	Nachwort	16
A	Merkmalsbeschreibung	17
B	C-Programm zur Auswertung	20
C	Tabellen	22
D	Das Netz	26

Abbildungsverzeichnis

3.1	Die Fehlerkurve des finalen Netzes für die Trainingsdaten (RProp).	12
3.2	Die Analyzerausgabe des finalen Netzes	13
3.3	Auswertung mit einem Schwellwert von 0.112 für die Trainingsmenge	13
4.1	Auswertung mit einem Schwellwert von 0.12 für die Testmenge .	14
D.1	Der „obere“ Teil des Netzes.	26
D.2	Der „untere“ Teil des Netzes.	27
D.3	Das gesamte Netz.	28

Tabellenverzeichnis

3.1	Netzstruktur	11
4.1	Ergebnisse	14
C.1	Fehlwerte der Testmenge	22
C.2	Fehlwerte der Trainingsmenge	23
C.3	Mittelwerte der Testmenge	24
C.4	Mittelwerte der Trainingsmenge	25

1 Vorwort

Die bearbeitete Aufgabe bestand aus der Aufgabenstellung für den Data Mining Cup 2002.

1.1 Ausgangssituation

Der deutsche Markt der Energieversorger ist seit 1999 dereguliert. Das heißt, dass seit diesem Zeitpunkt die Stromkunden in Deutschland frei ihren Anbieter wechseln können – vorher war dies nicht möglich.

Das ist eine neue Situation für Anbieter und Kunden. Der Kunde wird oft einen Wechsel vollziehen, um durch günstigere Preise an den Stromkosten zu sparen. Die Anbieter ihrerseits stehen vor der neuen Herausforderung, ihre Kunden mit guten und interessanten Angeboten binden zu müssen. Sie versuchen dabei, ihre Kunden individuell und bedürfnisgerecht anzusprechen. Vor diesem Hintergrund wird Data Mining u. a. zur Berechnung von Kundenprofilen eingesetzt.

1.2 Szenario

Ein Energieversorger möchte mit zielgerichteten Kundenbindungsmaßnahmen seine Kunden halten. Dazu möchte er potenziellen Kündigern einen günstigeren Tarif (Tarif „Rabatt“) anbieten. Da dieser Tarif für den Energieversorger einen geringeren Gewinn bedeutet, will er die vorraussichtlich treuen Kunden von diesem Angebot nicht informieren, um nicht zu vielen von diesen Kunden den Tarif „Rabatt“ gewähren zu müssen.

Ziel des Einsatzes von Data Mining ist es hierbei, die potenziellen Kündiger von den treuen Kunden zu unterscheiden. Potenziellen Kündigern wird dann der Tarif „Rabatt“ angeboten, um sie damit zu binden. Insgesamt ist es für den Energieversorger deutlich günstiger, einen potenziellen Kündiger weiterhin im Tarif „Rabatt“ zu versorgen als ihn ganz zu verlieren.

Es wird geschätzt, dass in den nächsten zwei Jahren ca. 10 % der Kunden des Energieversorgers kündigen würden, falls keine Kundenbindungsmaßnahmen durchgeführt werden. Als Trainingsmenge für das Erlernen des Kündigerprofils ist deshalb eine geschichtete Stichprobe von 9000 Kunden und 1000 Kündigern des letzten Jahres gezogen worden.

2 Vorbereitung

2.1 Datenvorverarbeitung

Es gab zur Datenvorverarbeitung zwei verschiedene Ansätze. Zum Einen haben wir überlegt ob es sinnvoll wäre keine Attribute zu streichen und nur eine gute Kodierung zu finden, da eine interessante Eigenschaft neuronaler Netze ja in ihrer Verarbeitungsfähigkeit von sog. „real world“ Daten besteht.

Der zweite Ansatz war dann der überflüssig erscheinende Attribute zu streichen. Hier haben wir im Vorfeld die folgenden ausgewählt:

- ID: ist keine Merkmalsausprägung ansich, sondern stellt eine Art Kundennummer ohne Aussagekraft dar. Da sie desweiteren einzigartig ist, würde sie die Ergebnisse mit Sicherheit verfälschen.
- Status: ist bereits verteilt über andere Attribute enthalten
- alle PWK-Indizes bis auf Dichte: erschien uns unlogisch

2.2 Software

Bei der Suche nach Software haben wir uns auf die bekannten Programme *SNNS* und *JavaNNS* beschränkt, da diese im KI-Labor zur Verfügung stehen und zumindest der *JNNS* auch auf den heimischen Rechnern ohne Probleme lief.

Das Problem mittels einer eigenen Software zu lösen war aufgrund diverser Hilfsmittel (z.B. der „Fast Artificial Neural Network Library“¹) durchaus verlockend, aber aufgrund der knapp bemessenen Zeit und der Tatsache, daß dies unser erster Schritt ohne vorherige Erfahrungen auf diesem Programmiergebiet gewesen wäre, entschieden wir uns doch für die Arbeit mit *JNNS*.

¹<http://fann.sourceforge.net/>

3 Durchführung

3.1 Datenvorverarbeitung

Zur Vereinfachung der Datenvorverarbeitung haben wir ein Skript in Python geschrieben, da klar war, daß eventuell im Verlaufe der Netztests eine nochmalige Modifizierung von Attributauswahl und Kodierung anstehen würde.

3.1.1 Sortierung

Um ein Durcheinander zu vermeiden wurden die Trainingsdaten und die Testdaten nach ID sortiert. Dies geschah noch mit Excel, weil es einfach schneller ging als mit Python. Die Testdaten wurden auf diesem Weg auch gleich mit dem Attribut „canceler“ zusammengeführt.

3.1.2 Fehlwerte

Mit einem Pythonskript haben wir die Daten zeilenweise untersucht. Wichtig war dabei zu schauen, ob die einzelnen Werte der Attribute in ihrem Definitionsbereich liegen. Deshalb wurde nach dem kleinsten und dem jeweils größten Wert pro Attribut gesucht und mit den Vorgaben verglichen. Fehlwerte sind uns dabei nicht aufgefallen.

Anschließend haben wir nach fehlenden Werten gesucht, d.h. nach Zeilen in den ein oder mehrere Attribute keinen Wert enthalten. Von jeweils 10000 Datensätzen in Trainings- und Testmenge ergab sich die in Tabelle C.1 dargestellte Auflistung.

In der Testmenge gibt es insgesamt 2352 Zeilen in denen Attribute fehlende Werte aufweisen und in der Trainingsmenge befinden sich 2392 Zeilen mit fehlenden Werten (s. Tab. C.1 und C.2).

Um weiterhin mit diesen Zeilen arbeiten zu können und ein Ergebnis über alle Daten zu erhalten, wurden fehlende Werte durch den Mittelwert des jeweiligen Attributes ersetzt. Die Berechnung des Mittelwertes und das Einfügen übernahm ebenfalls des Pythonskript.

Für die Mittelwerte wurde soweit gerundet, wie es für die spätere Klassifizierung notwendig war, d.h. Fließkommazahlen waren nicht von Bedeutung.

Die, die Leerstellen ersetzenden Mittelwerte sind in den Tabellen C.3 und C.4 aufgelistet.

3.1.3 Erstellen der Pattern - Dateien

Im nächste Schritt soll aus den jetzt komplett vorhandenen Daten die für JavaNNS benötigten Pattern Dateien erstellt werden.

Dafür ist es notwendig die einzelnen Attribute zu kodieren. Da wir erst durch Probieren auf Attribute verzichten wollten, wurden alle Attribute kodiert. Dafür gab es unterschiedliche Ansätze, z.B. die Kodierung eines Attributes für mehrere Neuronen. Dafür hat sich das Attribut `powerconsumption` angeboten, was wir später aber wieder ohne Unterschied durch eine Kodierung für nur ein Neuron ersetzt haben.

Unsere Lösungen variierten von 55 Neuronen bis 22 Neuronen. Und für unsere endgültige Lösungen haben wir eine Pattern Datei mit allen zur Verfügung stehenden Attributen außer „id“ benutzt, wobei jedem Attribut ein Neuron zugeordnet wurde und für „powerconsumption“ wurden für 2 Neuronen kodiert.

In dem Pythonskript wurden zeilenweise die Attribute in einem Datentyp Dictionary gespeichert:

```

datasrc=dict({ \
'id':line[0:7].rstrip(), \
'payment_type':line[8:15].rstrip(), \
'power_consumption':line[16:23].rstrip(), \
'hhh':line[24:31].rstrip(), \
'hgew':line[32:39].rstrip(), \
'mtreg0g':line[40:47].rstrip(), \
'mtkau0g':line[48:55].rstrip(), \
'mtstr0g':line[56:63].rstrip(), \
'mtbeb0g':line[64:71].rstrip(), \
'mtsta0g':line[72:79].rstrip(), \
'mtbon0g':line[80:87].rstrip(), \
'mtade0g':line[88:95].rstrip(), \
'mtalt0g':line[96:103].rstrip(), \
'mtfam0g':line[104:111].rstrip(), \
'mtkdi0g':line[112:119].rstrip(), \
'mtkle0g':line[120:127].rstrip(), \
'mtkkl0g':line[128:135].rstrip(), \
'mtkgb0g':line[136:143].rstrip(), \
'mtkgl0g':line[144:151].rstrip(), \
'scmwgr2':line[152:159].rstrip(), \
'scmwgr3':line[160:167].rstrip(), \
'scmwgr4':line[168:175].rstrip(), \
'scmwgr5':line[176:183].rstrip(), \
'scmwgr6':line[184:191].rstrip(), \
'scmwgr7':line[192:199].rstrip(), \
'scmwgr21':line[200:207].rstrip(), \
'scmwgr22':line[208:215].rstrip(), \
'pharm1':line[216:223].rstrip(), \
'pharm2':line[224:231].rstrip(), \
'pharm3':line[232:239].rstrip(), \
'pharm4':line[240:247].rstrip(), \
'pharm5':line[248:255].rstrip(), \
'pharm6':line[256:263].rstrip(), \
'canceler':line[264:271].rstrip()})

```

Erst erfolgte das Überprüfen der einzelnen Attribute, wie oben beschrieben, auf Fehlwerte und das Ersetzen der leeren Werte durch den Mittelwert. Danach konnte im selben Schritt die Kodierung von jedem Attribut stattfinden in dem Dictionary wird dann der kodierte Werte gespeichert. Die Kodierung für meh-

3 Durchführung

rere Neuronen geschah durch das Einfügen neuer Attribute in das Dictionary. Die Kodierung erfolgte dann, wie an folgenden Beispielen zu sehen ist:

```
if int(datasrc['power_consumption'])>0 and \
    int(datasrc['power_consumption'])<=500:
    datasrc['power_consumption1']='0'
if int(datasrc['power_consumption'])>500 and \
    int(datasrc['power_consumption'])<=1000:
    datasrc['power_consumption1']='0.33'
if int(datasrc['power_consumption'])>1000 and \
    int(datasrc['power_consumption'])<=1500:
    datasrc['power_consumption1']='0.66'
if int(datasrc['power_consumption'])>1500 and \
    int(datasrc['power_consumption'])<=2000:
    datasrc['power_consumption1']='1'
if int(datasrc['power_consumption'])>2000 and \
    int(datasrc['power_consumption'])<=2500:
    datasrc['power_consumption2']='0'
if int(datasrc['power_consumption'])>2500 and \
    int(datasrc['power_consumption'])<=3000:
    datasrc['power_consumption2']='0.25'
if int(datasrc['power_consumption'])>3000 and \
    int(datasrc['power_consumption'])<=3500:
    datasrc['power_consumption2']='0.5'
if int(datasrc['power_consumption'])>3500 and \
    int(datasrc['power_consumption'])<=4000:
    datasrc['power_consumption2']='0.75'
if int(datasrc['power_consumption'])>3500:
    datasrc['power_consumption2']='1'

if int(datasrc['mtstr0g'])=='1':
    datasrc['mtstr0g']='0'
elif int(datasrc['mtstr0g'])=='2':
    datasrc['mtstr0g']='0.25'
elif int(datasrc['mtstr0g'])=='3':
    datasrc['mtstr0g']='0.5'
elif int(datasrc['mtstr0g'])=='4':
    datasrc['mtstr0g']='0.75'
elif int(datasrc['mtstr0g'])=='5':
    datasrc['mtstr0g']='1'
```

Das Python Skript schreibt nach der Kodierung die mit Leerzeichen separierten Daten in eine neue Datei, die dann nur noch um den jeweiligen Pattern Header ergänzt werden muß. Die einzelnen Attribute wurden immer zwischen 0 und 1 kodiert. Nur im Fall von „powerconsumption“ haben wir uns die Verteilung der Daten anschauen müssen und danach klassifiziert, wie auch in dem Codebeispiel zu sehen ist.

3.2 Netze

3.2.1 Entwicklung und Test

Durch Änderungen am Pythonskript war es relativ einfach, sich verschiedene Patterndateien generieren zu lassen. Größtenteils haben wir jedoch Wert darauf gelegt lediglich die Kodierung der Attribute zu verändern.

Wir wollten, wenn möglich, mit allen zur Verfügung stehenden Attributen arbeiten. Da dies die Anwendbarkeit der entwickelten Netze auf „Rohdaten“ unterstreicht. Die vorhandenen Daten brauchen also lediglich kodiert zu werden, ohne erst noch umfangreiche Attributmodifikationen vornehmen zu müssen.

Wir haben die Netze mit dem **JavaNNS** entwickelt und getestet. Hierzu nutzen wir die in der Software vorhandenen Funktionen (Fehlergraph und Analyser). Sah ein Netz vielversprechend aus, so haben wir das Netz auf die Testdaten angewendet und mit Hilfe eines selbstgeschriebenen C-Programms den „Lift“ berechnet (s. Anhang B).

Netzaufbau

Es wurden im Laufe der Tests verschiedenste Netzstrukturen getestet, allerdings haben wir uns auf Feed-Forward Netze beschränkt, obwohl sich Selbstorganisierende Karten (SOMs) vielleicht auch bewährt hätten.

Das Netz, welches sich gegen Ende der Tests herauskristallisierte, ist folgendermaßen aufgebaut (s. Tab. 3.1):

- Eine Eingangsschicht bestehend aus 33 Neuronen mit logistischer Aktivierungsfunktion.
- Eine Zwischenschicht aus 8 Neuronen des Typs Hidden, die mit den ersten 14 Neuronen der Eingangsschicht verbunden sind.
- 1 Neuron des Typs Hidden, welches mit den Neuronen für die PKW-Indizes aus der Eingangsschicht verbunden ist.
- 1 Neuron des Typs Hidden, welches mit den Neuronen für die Versicherungstypen aus der Eingangsschicht verbunden ist.
- 1 Neuron des Typs Hidden, welches mit den Neuronen für die Pharmatypen aus der Eingangsschicht verbunden ist.
- 1 Neuron des Typs Hidden, das mit den 8 o.g. Neuronen verbunden ist.
- Ein Ausgangsneuron, welches mit den vier letztgenannten Neuronen und der 8'ter Zwischenschicht verbunden ist.

Es war unser Ziel, die verschiedenen Nuancen eines Merkmals (z.B. Pharmatyp) nicht zuviel Gewicht gewinnen zu lassen, daher haben wir für die drei Ausprägungen *PKW-Indizes*, *Versicherungstyp* und *Pharmatyp* jeweils ein Neuron

3 Durchführung

Tabelle 3.1: Netzstruktur

Schicht	Neuronen	Typ
1	33	Input
2	8+4 (Erklärung s. Text)	Hidden
3	1	Output

zwischengeschaltet, das die Signale aus den jeweiligen Blöcken von 5, 6 und 8 auf 1 reduziert.

Die ersten 14 Eingangsneuronen dagegen laufen in eine normale Zwischenschicht, welche dann mit dem Ausgangsneuron verbunden ist. Gleichzeitig ist diese mit noch einem Neuron verbunden, das wiederum zum Ausgangsneuron führt. Wir experimentierten hier mit unterschiedlichen Verbindungen. Jedoch war der Effekt quasi nicht meßbar, weshalb das Neuron eigentlich überflüssig ist und entfernt werden könnte.

Lernverfahren

Verschiedene uns bekannte Lernverfahren wurden ausprobiert und nach Anpassung ihrer Lernparameter entweder verworfen oder weitergenutzt.

Beim Quickprop Verfahren war eine starke Schwankung der Fehlerkurve zu erkennen, ohne daß sich diese normalisierte. Daher haben wir dieses Verfahren verworfen.

Backpropagation, Backpropagation mit Momentum und Resilient Propagation waren in ihren Ergebnissen etwa gleichwertig. Die Kurve unterschied sich hier lediglich auf den ersten 20 bis 30 Lernschritten, danach flachte sie fast zu einer Geraden ab. Weder das Erhöhen der Lernschritte noch das Variieren der Parameter führten zu einem stärkeren Abfall (s. Abb. 3.1).

Wir beobachteten lediglich, daß ein „langes“ Trainieren des Netzes zwar zu einem minimalen Abfall der Fehlerkurve, aber auch zu einem Anstieg derselben für die Validierungsdaten führte.

3.2.2 Das Problem

Beim Auswerten der Ergebnisse (Speichern in Result Datei und Anwendung des Programms) fiel uns auf, daß die Fehlerrate aller Netze mieserabel war.

Der normalerweise erwartete Schwellwert von 0,8 für Neuronen wurde nur mit einigen speziellen Kodierungen bzw. „frisierten“ Patterndateien erreicht. Beim Trainieren mit den Trainingsdaten gelang es uns teilweise den Fehler¹ auf unter

¹Hiermit ist der folgende Fehler (JNNS) gemeint:

$$\frac{1}{n} * \sum_{i=1}^n e^2$$

3 Durchführung

0,08 zu senken (vgl. Abb. 3.1).

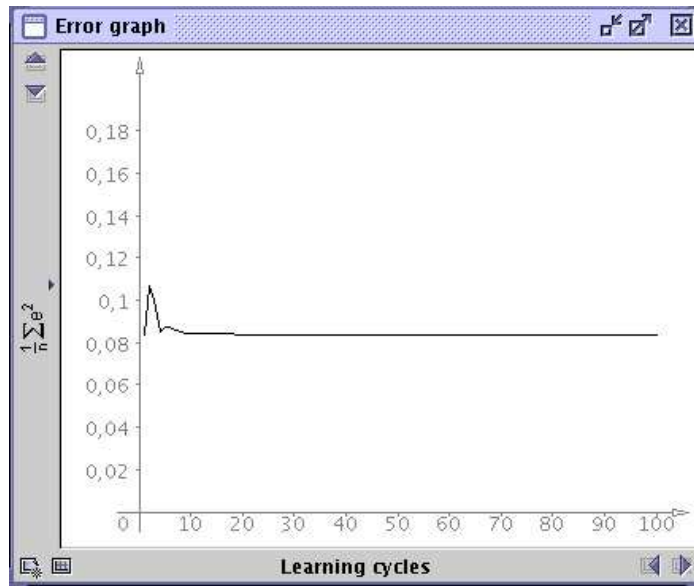


Abbildung 3.1: Die Fehlerkurve des finalen Netzes für die Trainingsdaten (RProp).

Jedoch pegelten die Werte des Ausgabeneurons zwischen 0 und 0,7. Wobei selbst Werte oberhalb von 0,3 nur als Spitzen (Ausreißer) auftraten (s. Abb. 3.2).

3.2.3 Die Lösung

Beim Betrachten der Analyzerausgabe (vgl. Abb. 3.2) kam die Idee auf, daß die Verteilung der Werte der realen Verteilung von Kündigern und Nichtkündigern entspräche, nur eben in einem anderen „Maßstab“. Daher erweiterten wir das Auswertungsprogramm um die Möglichkeit den Schwellwert für die „0 oder 1 Entscheidung“ selbst zu wählen.

Aufgrund der Ausgabe des Analyzers konnten wir abschätzen, daß der optimale Schwellwert irgendwo im Bereich um 0,2 liegt. Daher waren einige Test erforderlich, die jedoch sehr erfolgversprechend verliefen. Bereits nach einigen Versuchen konnten wir einen für unsere Lösung optimalen Schwellwert bei 0,112 feststellen. Abbildung 3.3 zeigt die Ausgabe des Auswertungsprogramms für den Schwellwert 0,112.

3 Durchführung

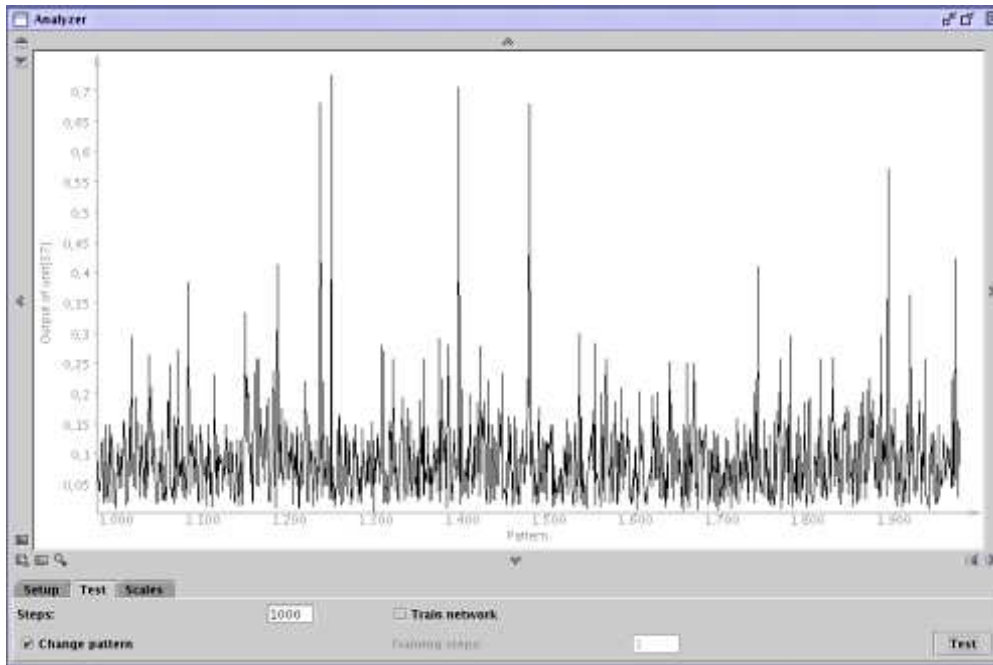


Abbildung 3.2: Die Analyzerausgabe des finalen Netzes

```
jens@nifelheim:Projekt/Das_Netz.alt# ./auswertung
Bitte Dateinamen eingeben: v_train.res
Bitte Schwellwert eingeben (Zahl oder "auto"): auto
Das koennte laenger dauern.
Optimaler Schwellwert bei 0.112 (Lift: 7714.188).
10000 Datensätze bearbeitet.
Davon 3284 "Einsen"...
Schwellwert: 0.112
Variablen:      c      nc      k      nk
Werte:         534     466     6250    2750
ziel = 43.80 * c + 66.30 * nk + 72.00 * k = 655714.188
lift = ziel - 72.00 * 9000 = 7714.188
```

Abbildung 3.3: Auswertung mit einem Schwellwert von 0.112 für die Trainingsmenge

4 Auswertung

4.1 Ergebnisse

Vermittels unseres Lösungsweges erreichten wir bei den klassierten Daten (Testdaten) einen „Lift“ von **9328.812**, was nach einem Vergleich mit der Ergebnistabelle des Data Mining Cups 2002 dem 1. Platz entspräche. Der ideale Schwellwert lag für die Testmenge bei 0,12 (s. Abb. 4.1). Die Ergebnisse sind nochmal in Tabelle 4.1 zusammengefaßt.

```
jens@nifelheim:Projekt/Das_Netz.alt# ./auswertung
Bitte Dateinamen eingeben: Ergebnisse.res
Bitte Schwellwert eingeben (Zahl oder "auto"): auto
Das koennte laenger dauern.
Optimaler Schwellwert bei 0.120 (Lift: 9328.812).
10000 Datensätze bearbeitet.
Davon 2801 "Einsen"...
Schwellwert: 0.120
Variablen:      c      nc      k      nk
Werte:         511     489     6710    2290
ziel = 43.80 * c + 66.30 * nk + 72.00 * k = 657328.812
lift = ziel - 72.00 * 9000 = 9328.812
```

Abbildung 4.1: Auswertung mit einem Schwellwert von 0.12 für die Testmenge

Tabelle 4.1: Ergebnisse

Variable	Wert	Bedeutung
c	534	Anzahl der Kündiger, die angeschrieben werden (richtig erkannt)
nc	466	Anzahl der Kündiger, die nicht angeschrieben (erkannt) werden
k	6250	Anzahl der Nichtkündiger, die nicht angeschrieben (richtig erkannt) werden
nk	2750	Anzahl der Nichtkündiger, die angeschrieben werden (falsch erkannt)

4.2 Interpretation und Bewertung

Die Anzahl der erkannten Kündiger ist in unseren Augen durchaus befriedigend, es wurden zwar nur 53,4% derselben korrekt klassifiziert, aber bei höheren *c*-Werten war die Fehlerrate immer inakzeptabel. Andererseits ist die Zahl der irrtümlich angeschriebenen Kunden nicht zu hoch (30,56%), was eine positive Kostenreduktion bedeutet.

Der sich insgesamt ergebende „Lift“ von 9328 Euro ist jedoch sehr überzeugend, d.h. der Stromanbieter macht im Gegensatz zur „Nichts tun Strategie“ 9328 EUR Gewinn. Hochgerechnet auf die wahre Kundenanzahl von 1 Million ergibt sich damit ein Profit von rund 933 Tausend Euro (+ 933 TEUR).

5 Nachwort

Nach Abschluß des Projektes und Abgleich mit den Ergebnissen der anderen Gruppen haben wir unser Netz noch einmal überarbeitet. Wir wollten die Neuronen der 8'er Zwischenschicht - wenn möglich - auf 1-2 reduzieren. Dabei stießen wir auf ein interessantes Verhalten.

Nach Reduktion der Zwischenschicht auf 2 Neuronen und anschließendem Training des Netzes war der Lift für die Trainingsdaten zunächst in etwa gleich dem vorherigen (ca. 7700-7800). Gegen Ende der Trainingsläufe zeigte sich jedoch, daß das Netz einen besseren Lift für die Trainingsdaten erzielte, nämlich einen Wert von 8115,625.

Natürlich haben wir anschließend getestet ob sich das gleiche Phänomen bei den Testdaten wiederholen würde. Jedoch bewegte sich der Lift hier in die entgegengesetzte Richtung (nach unten). Der beste Lift, den wir mit dem Netz erzielen konnten lag bei 6759 (Schwellwert: 0,112).

Dieses Verhalten deutet auf einen von uns nicht erkannten Zusammenhang hin, der vielleicht eine eingehendere Untersuchung rechtfertigen würde. Anscheinend ist es gar nicht so gut bei den Trainingsmengen einen zu „guten“ Lift zu erzielen.

A Merkmalsbeschreibung

Data-Mining-Cup 2002 Beschreibung der Merkmale

=====

Die Daten des Data-Mining-Cups 2002 sind anonymisierte Kundendaten eines Energieversorgers. Neben der "ID" sind "payment_type" und "power_consumption" vom Versorger gespeicherte Kundenmerkmale. Das Merkmal "canceler" ist das Zielmerkmal und ist nur in der Trainingsdatei vorhanden.

Die übrigen Merkmale sind Informationen der mikrogeografischen Datenbank MIKROTYP der Firma Consodata und dienen der Beschreibung der Kunden. Eine ausführliche Beschreibung dieser Merkmale ist in der Datei "de_Beschreibung_Mikrogeographie_Consodata.pdf" enthalten.

Merkmalsname	Typ	Beschreibung
ID	Integer	ID-Merkmal, Datenbank-Schlüsselfeld
payment_type	Text	Art der Bezahlung: 1 Überweisung durch den Kunden 2 Abbuchung vom Kundenkonto 3 andere Regelung
power_consumption	Integer	letzter jährlicher Stromverbrauch in kWh
HHH	Integer	Anzahl Haushalte im Haus
HGEW	Integer	Anzahl Gewerbe im Haus
MTREGOG	Integer	Regionaltyp*
MTKAUOG	Integer	Kaufkraft*
MTSTROG	Integer	Straßentyp*
MTBEOG	Integer	Bebauungstyp*
MTSTAOG	Integer	Status*

A Merkmalsbeschreibung

MTBONOG	Integer	Prüfungsgrad Bonität*
MTADEOG	Integer	Anteil Deutscher*
MTALTOG	Integer	Altersstruktur*
MTFAMOG	Integer	Familienstand*
MTKDIOG	Integer	PKW-Indizes: PKW-Dichte*
MTKLEOG	Integer	PKW-Indizes: PKW-Leistungsindex*
MTKKLOG	Integer	PKW-Indizes: PKW-Kleinbusindex*
MTKGBOG	Integer	PKW-Indizes: PKW-Gebrauchtwagenindex*
MTKGLOG	Integer	PKW-Indizes: PKW-Geländewagenindex*
SCMWGR2	Integer	Psychonomics-Versicherungstyp: Treuer Vertreterkunde*
SCMWGR3	Integer	Psychonomics-Versicherungstyp: Anspruchsvoller Delegierer*
SCMWGR4	Integer	Psychonomics-Versicherungstyp: Preisorientierter Rationalist*
SCMWGR5	Integer	Psychonomics-Versicherungstyp: Überforderter Unterstützung- suchender*
SCMWGR6	Integer	Psychonomics-Versicherungstyp: Skeptisch-Gleichgültiger*
SCMWGR7	Integer	Psychonomics-Versicherungstyp: Distinguiert-Konservativer*
SCMWGR21	Integer	Psychonomics-Versicherungstyp: Affinität zu Direktwerbung*
SCMWGR22	Integer	Psychonomics-Versicherungstyp: Affinität zum Direktvertrieb*
PHARM1	Integer	Pharmatypologie: Gesunder Kraftprotz*
PHARM2	Integer	Pharmatypologie: Unkritischer Wehleidiger*

A Merkmalsbeschreibung

PHARM3	Integer	Pharmatypologie: Skeptischer Verweigerer*
PHARM4	Integer	Pharmatypologie: Informierter Körperbewusster*
PHARM5	Integer	Pharmatypologie: Eingeschränkter Kassenpatient*
PHARM6	Integer	Pharmatypologie: Konservativer Arztgläubiger*
canceler	yes/no	Zielmerkmal: yes Kündigung no Kunde

* (siehe MIKROTYP-Beschreibung in
de_Beschreibung_Mikrogeographie_Consodata.pdf)

B C-Programm zur Auswertung

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define DEBUG 0

float getLift(char name1[512], char name2[512], float schwelle,\
int auswertung)
{
    FILE *fd=0, *sd=0;
    unsigned int line=0, anz=0;
    int output=-1;
    float realoutput=0, ziel=0, lift=0;
    char zeile[512];
    unsigned int c=0, nc=0, k=0, nk=0;

    if ((fd=fopen(name1, "r")) && (sd=fopen(name2, "w")))
    {
        if (DEBUG) fprintf(stderr, "Dateien geöffnet.\n");
        /* Header ueberlesen
         * Die ersten 10 Zeilen brauchen wir nicht!
         */

        fscanf(fd, "%s", zeile);
        while (!strstr(zeile, "#1.1")) {
            fscanf(fd, "%s", zeile);
        }

        fscanf(fd, "%i", &output);
        if (DEBUG) fprintf(stderr, "%i, ", output);
        while (!feof(fd))
        {
            fscanf(fd, "%f", &realoutput);
            if (DEBUG) fprintf(stderr, "%f (%f, %i, %i)\n", realoutput, \
                schwelle, line, anz);
            if (realoutput > schwelle)
            {
                fprintf(sd, "1\n");
                anz++;
                if (output != 1) {
                    nk++; /* Nichtkuendiger falsch erkannt */
                } else {
                    c++; /* Kuendiger richtig erkannt */
                }
            } else {
                fprintf(sd, "0\n");
                if (output != 0) {
                    nc++; /* Kuendiger falsch erkannt */
                } else {
                    k++; /* Nichtkuendiger richtig erkannt */
                }
            }
        }
        fscanf(fd, "%s", zeile); /* #... ueberlesen */
        fscanf(fd, "%i", &output); /* output auslesen */
        if (DEBUG) fprintf(stderr, "%i, ", output);
        line++;
    }
}
```

B C-Programm zur Auswertung

```
}
fclose(fd);
fclose(sd);

/* Berechnungen */
ziel = (float)(43.80*c + 66.30 * nk + 72.00 * k);
lift = (float)(ziel - (72 * 9000));

if (auswertung) {
    fprintf(stdout, "%i Datensätze bearbeitet.\n", line);
    fprintf(stdout, "Davon %i \"Einsen\"...\n", anz);
    fprintf(stdout, "Schwellwert: %.3f\n", schwelle);
    fprintf(stdout, "Variablen:\tc\tnc\tk\tkn\n");
    fprintf(stdout, "Werte: \t\t%i\t%i\t%i\t%i\n", c, nc, k, nk);
    fprintf(stdout, "ziel = 43.80 * c + 66.30 * nk + 72.00 * k = \
%.3f\n", ziel);
    fprintf(stdout, "lift = ziel - 72.00 * 9000 = %.3f\n", lift);
}

return(lift);
} else {
    fprintf(stderr, "Konnte Datei nicht öffnen!\n");
    return(43801);
}
}

int main(void)
{
    char name1[512], name2[512], swert[32];
    float schwelle, lift, l, s;

    fprintf(stdout, "Bitte Dateinamen eingeben: ");
    fscanf(stdin, "%s", name1);
    fprintf(stdout, "Bitte Schwellwert eingeben (Zahl oder \"auto\"): ");
    fscanf(stdin, "%s", swert);

    /* Automatik gewünscht? */
    if (!strcmp(swert, "auto")) {
        sprintf(name2, "%s-sw-auto.txt", name1);
        fprintf(stdout, "Das koennte laenger dauern.\n");
        l = 0; s = 0;
        for (schwelle=0.001;schwelle<1;schwelle=schwelle+0.001) {
            lift = getLift(name1, name2, schwelle, 0);
            if (lift > l) {
                l = lift;
                s = schwelle;
                fprintf(stdout, "+");
            } else {
                fprintf(stdout, ".");
            }
        }
        fprintf(stdout, "\nOptimaler Schwellwert bei %.3f \
(Lift: %.3f).\n", s, l);
        sprintf(name2, "%s-sw%f.txt", name1, s);
        lift = getLift(name1, name2, s, 1);
    } else {
        schwelle = atof(swert);
        sprintf(name2, "%s-sw%f.txt", name1, schwelle);
        lift = getLift(name1, name2, schwelle, 1);
    }
    return(0);
}
```

C Tabellen

Tabelle C.1: Fehlwerte der Testmenge

Attribut	Anzahl Leerzeilen
mtkdi0g	862
mtkgb0g	862
mtreg0g	856
scmwgr2	2255
hhh	853
scmwgr4	2255
scmwgr5	2242
scmwgr6	2139
scmwgr7	2243
scmwgr3	2139
mtbon0g	856
pharm5	1969
mtkg10g	862
scmwgr22	2254
scmwgr21	2255
mtade0g	856
mtkk10g	2075
mtsta0g	856
gesamt	2352
hgew	853
mtbeb0g	856
mtkau0g	856
mtalt0g	856
pharm4	1969
mtfam0g	856
pharm6	1969
mtkle0g	862
pharm1	1969
pharm2	1969
pharm3	1969
mtstr0g	856

Tabelle C.2: Fehlwerte der Trainingsmenge

Attribut	Anzahl Leerzeilen
mtkdi0g	854
mtkgb0g	854
mtreg0g	852
scmwgr2	2296
hhh	851
scmwgr4	2296
scmwgr5	2286
scmwgr6	2188
scmwgr7	2285
scmwgr3	2188
mtbon0g	852
mtfam0g	852
mtkgl0g	854
scmwgr22	229
scmwgr21	229
mtade0g	852
mtkkl0g	2117
mtsta0g	852
gesamt	2392
hgew	851
mtbeb0g	852
mtkau0g	852
mtalt0g	852
pharm4	2012
pharm5	2012
pharm6	2012
mtkle0g	854
pharm1	2012
pharm2	2012
pharm3	2012
mtstr0g	852

Tabelle C.3: Mittelwerte der Testmenge

Attribut	Mittelwert
paymenttype	2
powerconsumption	2526
hhh	3
hgew	0
mtreg0g	14
mtkau0g	-6
mtstr0g	2
mtbeb0g	2
mtsta0g	4
mtbon0g	4
mtade0g	7
mtalt0g	4
mtfam0g	5
mtkdi0g	7
mtkle0g	3
mtkkl0g	2
mtkqb0g	4
mtkgl0g	1
scmwgr2	3
scmwgr3	3
scmwgr4	3
scmwgr5	3
scmwgr6	3
scmwgr7	3
scmwgr21	3
scmwgr22	3
pharm1	4
pharm2	2
pharm3	5
pharm4	3
pharm5	4
pharm6	5

Tabelle C.4: Mittelwerte der Trainingsmenge

Attribut	Mittelwert
paymenttype	2
powerconsumption	2491
hhh	3
hgew	0
mtreg0g	14
mtkau0g	-6
mtstr0g	2
mtbeb0g	2
mtsta0g	4
mtbon0g	4
mtade0g	7
mtalt0g	4
mtfam0g	5
mtkdi0g	7
mtkle0g	3
mtkkl0g	2
mtkgb0g	5
mtkgl0g	1
scmwgr2	3
scmwgr3	3
scmwgr4	3
scmwgr5	3
scmwgr6	3
scmwgr7	3
scmwgr21	3
scmwgr22	3
pharm1	4
pharm2	2
pharm3	5
pharm4	3
pharm5	4
pharm6	5

D Das Netz

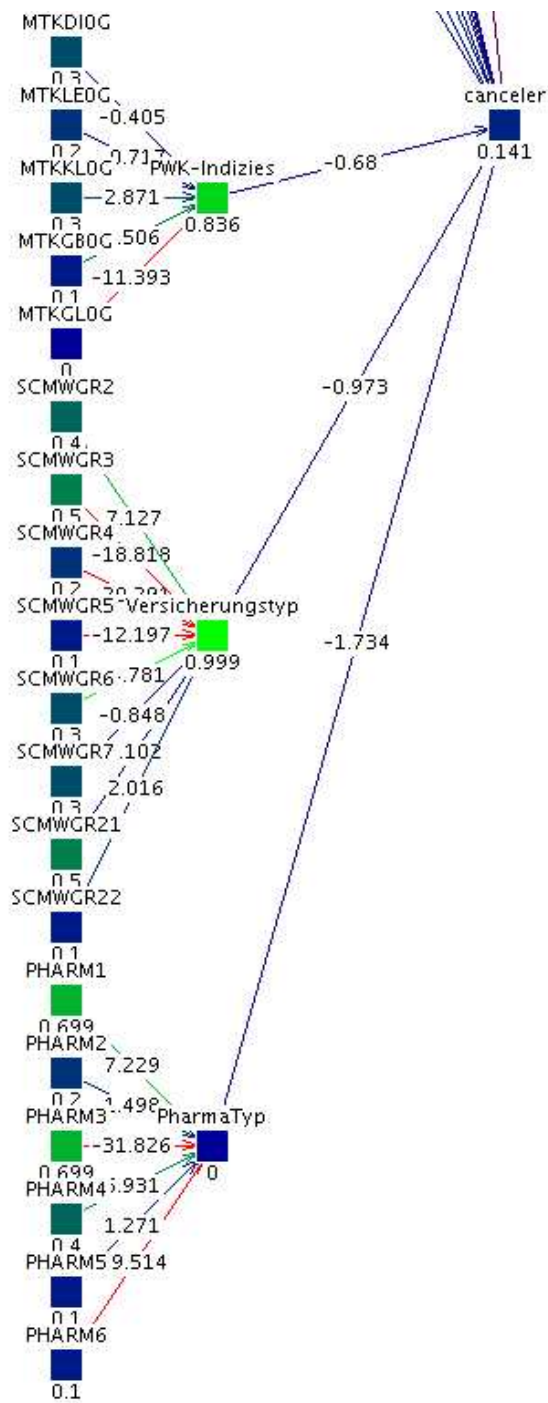


Abbildung D.2: Der „untere“ Teil des Netzes.

D Das Netz

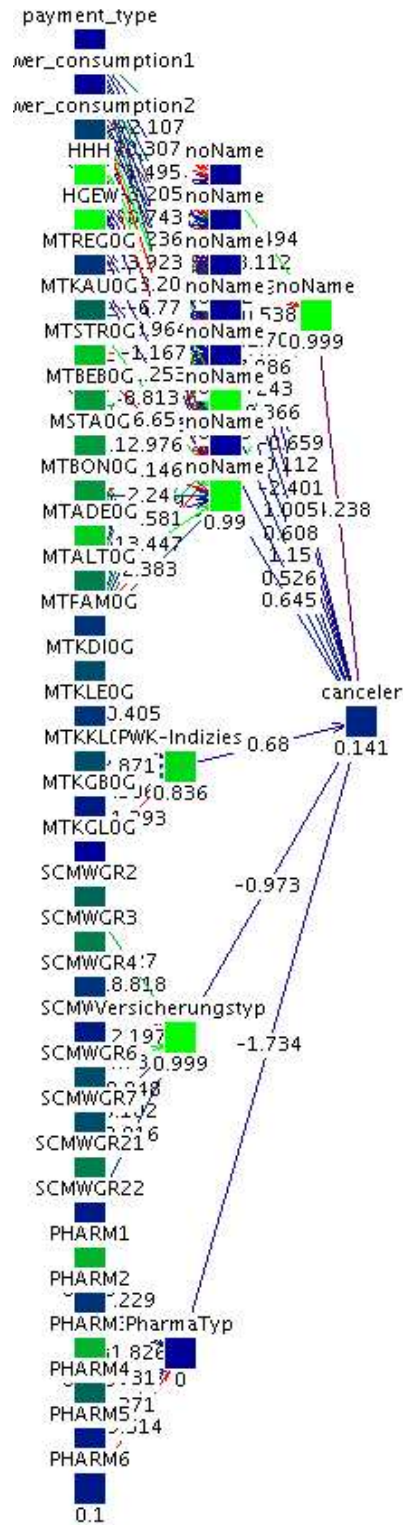


Abbildung D.3: Das gesamte Netz.

Literaturverzeichnis

- [1] Lämmel, Uwe; Cleve, Jürgen *Lehr- und Übungsbuch - Künstliche Intelligenz*, 2. Auflage, Fachbuch Verlag Leipzig, 2004, ISBN-3-446-22574-9