PAC-Lernen zur Insolvenzerkennung und Hotspot-Identifikation

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
“Dr. rerum naturalium“
an der Georg-August-Universität Göttingen

vorgelegt von

Thomas Brodag

geboren in Hann.Münden

Göttingen, April 2008

D 7

1. Referent: Professor Dr. Stephan Waack
2. Korreferent: Professor Dr. Carsten Damm

Tag der mündlichen Prüfung: 28. Mai 2008

Inhaltsverzeichnis

1. Einführung ... 6
1.1. Maschinelles Lernen in diversen Anwendungsbereichen ... 6
1.2. Das Problem Insolvenzerkennung ... 6
1.3. Das Problem der Hotspot-Identifikation ... 8
1.4. Ziel dieser Arbeit ... 8
2. Bisherige Ansätze zur Insolvenzerkennung ... 9
2.1. Grundlagen ... 9
2.2. Neuronale Netze ... 9
2.2.1. Einführung ... 9
2.2.2. Das Perzeptron und lineare Klassifikation ... 10
2.2.3. Neuronale Netze durch mehrere Perzeptrons ... 11
2.2.4. Funktionsapproximation mit neuronalen Netzen ... 12
2.2.5. Lernverfahren: Gradientenabstieg ... 13
2.3. Nearest-Neighbor-Verfahren ... 14
2.4. Multivariate Diskriminanzanalyse ... 15
2.5. Kendall-Verfahren ... 17
2.6. Support Vector Maschinen mit Kernen ... 18
2.7. Regressionsanalysen: Logit- und Probit-Modell ... 20
2.7.1. Einführung ... 20
2.7.2. Maximum-Likelihood-Verfahren zur Parameterschätzung ... 21
2.7.3. Logit-Modell ... 22
2.7.4. Probit-Modell ... 23
3. Theoretische Hintergründe des PAC-Lernens ... 24
3.1. Einführung der Formalien ... 24
3.1.1. Lernuniversum ... 24
3.1.2. Konzeptklasse und Hypothesenklasse ... 24
3.1.3. Lernstichprobe ... 24
3.1.4. Fehlerschranke und Unsicherheit ... 25
3.2. Eigenschaften von PAC-Lernalgorithmen ... 26
3.2.1. Konsistenz ... 26
3.2.2. Effizienz ... 26
3.2.3. Widerspruchsfreiheit ... 27
3.3. PAC-Lernen am Beispiel ... 27
3.3.1. Problemstellung und Algorithmus ... 27
3.3.2. Beweis: Konsistenz von SER ... 28
3.4. Pattern Klassen ... 31
3.4.1. Einführung ... 31
3.4.2. Patternanalyse-Algorithmen ... 33
3.4.3. Konsistenz ... 34
3.5. Rademacher-Komplexität ... 34
3.6. Empirische Risiko-Minimierung ... 35
3.7. Hauptsatz der Rademacher-Theorie ... 37
3.8. PAC-Lernen mit Rauschen auf der Klassifikation ... 38
3.8.1. Einführung ... 38
3.8.2. PAC-Lernalgorithmus mit Rauschen auf der Klassifikation ... 40
3.8.3. Effizienz ... 40
3.8.4. Konsistenz bei PAC-Lernen mit Rauschen auf der Klassifikation ... 40
3.9. Schwache empirische Risiko-Minimierung ... 41

3.9.1. Arbeitsweise des Lernalgorithmus ... 41
3.9.2. Abtasten der Rauschrate ... 42
3.10. Statistical Queries ... 43
3.10.1. Einführung ... 43
3.10.2. Simulation einer Statistical Query ... 44
3.10.3. Berechnung der Labelwahrscheinlichkeiten ... 45
3.10.4. Simulation einer Conditional Statistical Query ... 45
4. Entwicklung einer Konzeptklasse zur Insolvenzerkennung ... 47
4.1. Klassifikation durch Schwellwerte ... 47
4.1.1. Einführung und grundlegende Idee ... 47
4.1.2. Formalisierung ... 47
4.1.3. Auswahl und Zusammenhänge der Konzeptklassen ... 52
4.1.4. Einschränkung der Abtastbereiche für Schwellwerte ... 52
4.1.5. Eigenschaften von Schwellwertfunktionen ... 53
4.2. Beweis: NP-Schwierigkeit im Falle des PAC-Lernens mit Rauschen ... 54
4.2.1. Das Problem Set-Cover ... 54
4.2.2. Reduktionsbeweis ... 55
4.3. Lernalgorithmus durch Brute-Force ... 58
4.4. Lernalgorithmus durch dynamische Programmierung ... 58
4.4.1. Lösung des Rucksackproblems mit dynamischer Programmierung ... 59
4.4.2. Lernen von Monomen mit dynamischer Programmierung ... 60
4.4.3. Lösung und Fehlerskalierung ... 66
4.4.4. Laufzeitanalyse ... 68
4.5. Heuristischer Lernalgorithmus ... 68
4.6. Effiziente Schwellwertabtastung für Klauseln ... 72
4.7. Lernalgorithmus durch Statistical Queries ... 74
4.8. Rademacher-Komplexität der vorgestellten Konzeptklassen ... 79
5. Anwendung auf das Problem der Insolvenzerkennung ... 82
5.1. Einführung ... 82
5.2. Datenlage ... 83
5.3. Behandlung von fehlenden Werten in Jahresabschlüssen ... 85
5.4. Kennzahlen und Jahresabschlüsse ... 86
5.5. Kennzahlenauswahl und Beschreibung ... 88
5.5.1. Jahresüberschuss / Jahresfehlbetrag Tsd EUR ... 90
5.5.2. Cash-Flow ... 90
5.5.3. Current Ratio (Liquidität 3. Grades) ... 92
5.5.4. Eigenkapitalrentabilität ... 93
5.5.5. Gesamtkapitalrentabilität ... 94
5.5.6. Return on Investment ... 94
5.5.7. Verschuldungsgrad ... 95
5.5.8. Eigenkapitalquote ... 96
5.5.9. Fremdkapitalquote ... 96
5.5.10. Dynamischer Verschuldungsgrad ... 96
5.5.11. Working Capital ... 97
5.5.12. EBIT ... 98
5.5.13. EBITDA ... 98
5.5.14. Net Working Capital ... 99
5.5.15 Verbleibende Uthoff Kennzahlen ... 99
5.6. Analyse der Daten ... 100
5.6.1. Kern-PCA ... 100
5.7. Parameter der Verfahren ... 105

5.7.1. Thresholder ... 106
5.7.2. Multivariate Diskriminanzanalyse ... 106
5.7.3. Nearest-Neighbor ... 106
5.7.4. Neuronales Netz ... 106
5.7.5. Support Vector Machine (SVM) ... 107
5.8. Ergebnisse und Vergleich ... 107
5.8.1. Das Experiment und Vergleich der Verfahren ... 107
5.8.2. Interpretation der erlernten Hypothesen ... 112
5.8.3. Fazit, Ausblick und kritische Bewertung ... 114
6. Anwendung auf das Problem der Hotspot-Identifikation ... 116
6.1. Einführung ... 116
6.2. Datenlage und Aufbereitung ... 116
6.3. Ergebnisse und Evaluierung ... 120
6.4. Abschließende Betrachtung ... 125
7. Das Tool InSolve ... 126
8. Zusammenfassende Betrachtung und Fazit ... 127
Anhang A ... 128
A.1. Funktionen und Dokumentation von InSolve ... 128
A.1.1. Start von InSolve und Menus ... 128
A.1.2. Eingabedateien und Import-Dialog ... 130
A.1.3. Training und Vorhersage mit InSolve ... 133
A.1.4. Ratiofile-Dialog: Erstellung eigener Kennzahlen ... 136
A.1.5. Keyfile-Dialog: Merkmalsbeschränkung und Eingabekontrolle ... 139
A.1.6. Paths-Dialog ... 141
A.1.7. Plugin-Dialog ... 142
A.1.8. Method-Combination-Dialog ... 144
A.1.9. Visualization-Dialog ... 146
A.2. Release und JavaDoc ... 150
A.2. Technische Hilfsmittel ... 150
Literaturverzeichnis ... 151
Danksagung ... 155
Lebenslauf ... 156

Abbildungsverzeichnis

Abbildung 1: Phasen einer Unternehmenskrise 7
Abbildung 2: Perzeptron 10
Abbildung 3: Neuronales Netz mit einer verdeckten Schicht 11
Abbildung 4: Schematische Peak Bildung durch Überlagerung 12
Abbildung 5: Schematische Funktionsapproximation mit Peaks 13
Abbildung 6: Schematische Abbildung in den Merkmalsraum 19
Abbildung 7: Fehlerbereich (grau) der Ausgabe des Algorithmus SER zum Zielkonzept g 29
Abbildung 8: Konstanz der Positivbeispiele bei Schwellenauswahl 64
Abbildung 9: KPCA der 356 Unternehmen mit linearem Kern 101
Abbildung 10: Eigenwertspektrum zur KPCA mit linearem Kern 102
Abbildung 11: KPCA der 356 Unternehmen mit Polynomkern 103
Abbildung 12: Eigenwertspektrum zur KPCA mit Polynomkern 103
Abbildung 13: KPCA der 356 Unternehmen mit Radialbasiskern 104
Abbildung 14: Eigenwertspektrum zur KPCA mit Radialbasiskern 105
Abbildung 15: Hotspot-Verteilung nach ASEdb Hotspot-Datenbank 120
Abbildung 16: InSolve Hauptdialog 128
Abbildung 17: InSolve File Menu 129
Abbildung 18: InSolve Method Menu 129
Abbildung 19: InSolve Settings Menu 129
Abbildung 20: Aufbau einer CSV-Datei für InSolve 130
Abbildung 21: Import-Dialog: CSV Import 131
Abbildung 22: Import-Dialog: Missing Values 132
Abbildung 23: Import-Dialog: Preview 133
Abbildung 24: Icons: Trainiert / Nicht trainiert 134
Abbildung 25: Results-Dialog 135
Abbildung 26: Result-Details-Dialog 136
Abbildung 27: Ratiofile-Dialog 137
Abbildung 28: Ratiofile-Dialog Autovervollständigung 138
Abbildung 29: Ratiofile-Dialog: Eingabe eines Namens für eine neue Kennzahl 139
Abbildung 30: Keyfile-Dialog 140
Abbildung 31: Paths-Dialog 141
Abbildung 32: InSolve Hauptdialog: Quick Options 141
Abbildung 33: Plugin-Dialog 142
Abbildung 34: MethodsCombination-Dialog 144
Abbildung 35: Icons: Trainiert / Nicht trainiert für kombinierte Methoden 145
Abbildung 36: Visualization-Dialog 146
Abbildung 37: Visualization-Dialog: Unternehmensinformation für Datenpunkte 147
Abbildung 38: Visualization-Dialog: Show All Pairs Modus 148
Abbildung 39: Visualization-Dialog: Manuelle Diskriminanzanalysen 149

Tabellenverzeichnis

Tabelle 1: Kennzahlensystem 89
Tabelle 2: Verfahrensvergleich mit Missing Value Strategie ‚Zero' 107
Tabelle 3: Verfahrensvergleich mit Missing Value Strategie ‚Mean' 108
Tabelle 4: Verfahrensvergleich mit Missing Value Strategie ‚Median' 108
Tabelle 5: Thresholder Klassifikationsraten bei einem Merkmal 109
Tabelle 6: Thresholder Klassifikationsraten bei zwei Merkmalen 109
Tabelle 7: Thresholder Klassifikationsraten bei drei Merkmalen 110
Tabelle 8: Thresholder Klassifikationsraten bei vier Merkmalen 110
Tabelle 9: Thresholder Klassifikationsraten bei sechs Merkmalen 110
Tabelle 10: Thresholder Klassifikationsraten bei neun Merkmalen 111
Tabelle 11: Erlernte Hypothesen für verschiedene Parameter (Heuristik) 113
Tabelle 12: Hypothesen für Merkmalstyp SingleRel (Heuristik) 122
Tabelle 13: Hypothesen für Merkmalstyp SingleRel (Statistical Queries) 123
Tabelle 14: Hypothesen für Merkmalstyp SingleByPairRel (Heuristik) 123
Tabelle 15: Hypothesen für Merkmalstyp SingleByPairRel (Statistical Queries) 124
Tabelle 16: Hypothesen für Merkmalstyp PairRel (Heuristik) 124

1. Einführung

1.1. Maschinelles Lernen in diversen Anwendungsbereichen

Die Methoden des maschinellen Lernens erweisen sich nicht nur in der Bioinformatik als sehr effektiv, sondern werden vor allem aufgrund ihrer Universalität in verschiedensten Bereichen angewendet. Im Falle des überwachten Lernens haben alle Ansätze eines gemeinsam: Es wird versucht, anhand von Daten bestimmte, generalisierbare Verteilungseigenschaften zu lernen (Training), um somit auch auf ungesehenen Daten zuverlässige Vorhersagen treffen zu können. Bekannte Verfahren sind Support Vector Maschinen und neuronale Netze, deren Training durch die Lösung eines Optimierungsproblems mit speziellen Algorithmen erfolgt.
Support Vector Maschinen zeichnen sich dabei vor allem durch ihre Effizienz und eine gewisse Resistenz gegenüber mangelhafter Generalisierungsfähigkeit (Overfitting) aus.
Neuronale Netze erlauben das Design komplexer Modelle (Netze), die durch ein Standardverfahren wie dem Gradientenabstieg oder der Delta-Lernregel trainiert werden können. Die Leistung des Anwenders liegt hier also stärker in der Konzeption als in der Entwicklung eines Lernalgorithmus.
Die Anwendung der Methoden setzt häufig kein exaktes Wissen der Lernalgorithmen oder der Lerntheorie voraus, da lediglich die Eingabedaten vom Anwender in aufbereiteter Form zu übergeben sind, um diese anschließend verarbeiten zu können.
Ein weiteres Modell im Bereich des maschinellen Lernens ist das Konzept-Lernen (hier PAC-Lernen). In diesem Modell kann die Optimalität eines Lernalgorithmus auf ein einheitliches Induktionsprinzip zurückgeführt werden. Die Anwendung dieses Modells setzt zum einen die Konzeption des Lernziels (Konzeptklasse) voraus, sowie die Entwicklung eines für das Lernziel angemessenen Lernalgorithmus.

1.2. Das Problem Insolvenzerkennung

In diversen Situationen ist es für ein Unternehmen der Wirtschaft sinnvoll, einen Kredit in Anspruch zu nehmen, wenn etwa eine größere Investition bevorsteht oder es darum geht, eine finanzielle Krise zu überstehen. Für einen potentiellen Kreditgeber ist dies mit einem Risiko verbunden, da der Kreditnehmer unter Umständen nicht in der Lage ist, den Kredit in vollem Umfang zurückzuzahlen.

Der Kreditgeber versucht dabei durch gezielte Analysen der Bonität des Kreditnehmers das Risiko zu minimieren oder zumindest kalkulierbar zu machen, denn der schlimmste Fall wäre eine Insolvenz des Schuldners, die zum Verlust des Kredits führt.
In diesem Zusammenhang durchlaufen Unternehmen im Regelfall mehrere Phasen, bis als letzte Konsequenz das Insolvenzverfahren eingeleitet wird (siehe [NW04]).

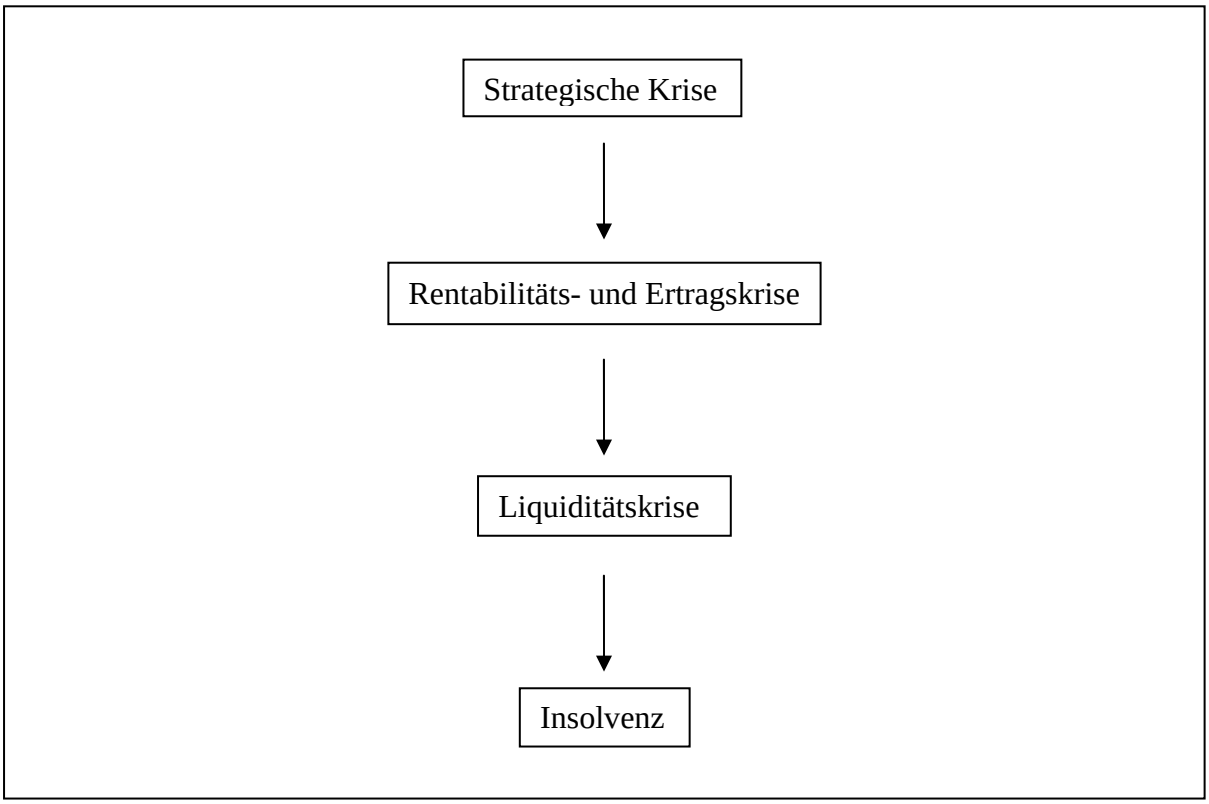


Abbildung 1: Phasen einer Unternehmenskrise

Der Kreditgeber versucht daher, ein gefährdetes Unternehmen frühzeitig zu erkennen, um eine Kreditanfrage rechtzeitig ablehnen zu können. Diese Aufgabe wird normalerweise von Wirtschaftsexperten übernommen, die über entsprechende Erfahrung und Branchenkenntnisse verfügen. Es werden zum einen Managementbefragungen durchgeführt, aber auch Insiderwissen über die in Rede stehende Branche genutzt, um so eine möglichst exakte Einschätzung vornehmen zu können.
Obwohl laut einer Studie des Verbandes der Vereine Creditreform e.V. (siehe [CR07]) die Anzahl der Unternehmensinsolvenzen im Jahr 2007 um 10,4% auf 27.490 Fälle zurückgegangen sind, hat sich die Rückgangsquote zum Vorjahr wieder leicht verringert (im Jahr 2006: 30.680 Unternehmensinsolvenzen bei einer Rückgangsquote von 16,7%).
Erste Schätzungen der Schadensvolumina durch Unternehmensinsolvenzen werden von der Creditreform für das Jahr 2007 auf 29,2 Milliarden Euro geschätzt.

Aufgrund der aktuellen Finanzmarktturbulenzen ist es besonders für größere Unternehmen schwieriger geworden, Kredite zu erhalten, da Kreditgeber verschärft die Bonität von Unternehmen beurteilen. Dementsprechend existiert unverändert eine Nachfrage nach zuverlässigen Methoden zur frühzeitigen Erkennung von Unternehmensinsolvenzen.

1.3. Das Problem der Hotspot-Identifikation

Proteine steuern oder katalysieren biochemische Prozesse in Organismen und sind daher seit vielen Jahren Forschungsgegenstand in der Biologie. Das Studium dieser Moleküle erlaubt ein besseres Verständnis biochemischer Prozesse. In vielen Fällen bestehen Proteine aus verschiedenen Untereinheiten, die erst nach korrekter Zusammensetzung eine bestimmte Funktion übernehmen können. Da die Experimente zur Bestimmung von Interaktionspartnern im Labor sehr aufwändig sind, existiert ein großer Bedarf an automatisierten Verfahren, die potentielle Interaktionspartner bestimmen können. Wie Bindestellen (Interfaces) genau gefunden werden, und wodurch die Stabilität der Bindung zustande kommt, ist noch nicht hinreichend geklärt.
Eine verbreitete Theorie ist die Hotspot-Theorie, die die Stabilität der Interfaces auf einzelne Aminosäuren in den Interfaces zurückführt. Durch Erkennung dieser Stellen wäre somit eine brauchbare Ausgangslage für den Einsatz von Dockingalgorithmen gefunden, die in der Lage sind, ein Interface von einer Startposition heraus genauer zu charakterisieren.

1.4. Ziel dieser Arbeit

Diese Arbeit soll die Anwendbarkeit von Methoden des maschinellen Lernens in verschiedenen Bereichen demonstrieren. Dabei liegt der Schwerpunkt auf dem Modell des PAC-Lernens mit Rauschen auf der Klassifikation. Letztlich soll eine Aussage darüber getroffen werden können, wie gut sich diese Methoden für die Anwendung in der Praxis eignen. Es sollen dazu auch das zugrunde liegende Modell und die nötigen Lernalgorithmen analysiert werden. Die entwickelte Methode wird auf das Problem der Insolvenzerkennung sowie der Hotspot-Identifikation angewandt und mit anderen Verfahren verglichen. Um eine wissenschaftliche Anwendung in der Praxis zu ermöglichen, sowie Nachvollziehbarkeit der Ergebnisse und eine Weiterentwicklung zu gewährleisten wird weiterhin ein Java-Tool entwickelt, das eine grafischen Oberfläche und einige Zusatzfunktionen bietet, die vor Allem im Zusammenhang mit der Insolvenzvorhersage gebraucht werden. Das Programm umfasst nicht nur das in dieser Arbeit vorgestellte Verfahren, sondern auch einige der bekanntesten Konkurrenzverfahren.

2. Bisherige Ansätze zur Insolvenzerkennung

In der Vergangenheit sind bereits zahlreiche Verfahren angewendet worden, um zukünftig insolvente Unternehmen zu erkennen. Der folgende Abschnitt soll einige bewährte Verfahren kurz vorstellen.

2.1. Grundlagen

Grundlage für die Anwendung der folgenden Verfahren ist eine gelabelte Lernstichprobe, anhand der ein Klassifikator ermittelt wird, um ungesehene Beispiele beurteilen zu können. Diese sei wie folgt definiert:

$$U[1,m] := ((\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_m, y_m)) \qquad y_i \in \{+1,-1\},\ i = 1,...,m$$

Im Falle des PAC-Lernens (siehe Kapitel 3) wird diese noch allgemeiner definiert. Zur Einführung wird lediglich davon ausgegangen, dass Daten in Form von *m* Vektoren mit jeweils *n* Komponenten (Merkmalen bzw. Kennzahlen) vorliegen.

2.2. Neuronale Netze

2.2.1. Einführung

Neuronale Netze orientieren sich an der Grundidee eines vernetzten Systems von Neuronen, ähnlich dem menschlichen Gehirn. Heutzutage wird häufig von „Brain inspired neuronal networks“ gesprochen, da das ursprüngliche Ziel, ein menschliches Gehirn nachzubilden, zu hoch gegriffen war. Selbst bis zum heutigen Tage kann die Hirnforschung noch nicht ergründen, wie ein Bewusstsein entsteht oder Lernprozesse genau funktionieren. Daher sind neuronale Netze im Wesentlichen mathematische Hilfsmittel.
Einer der Nachteile neuronaler Netze ist, dass die gewonnen Ergebnisse nur schwer interpretierbar sind, da komplexe Wechselwirkungen der Merkmale untereinander für die Klassifikation verantwortlich sind und eine hohe Anzahl von Variablen gelernt werden. Folglich ist auch die Vorgehensweise stark verschleiert und auch eine exakte Reproduktion von neuronalen Netzen anhand von Beschreibungen nur schwer möglich.

2.2.2. Das Perzeptron und lineare Klassifikation

Die Grundlage für ein neuronales Netz bildet das Perzeptron:

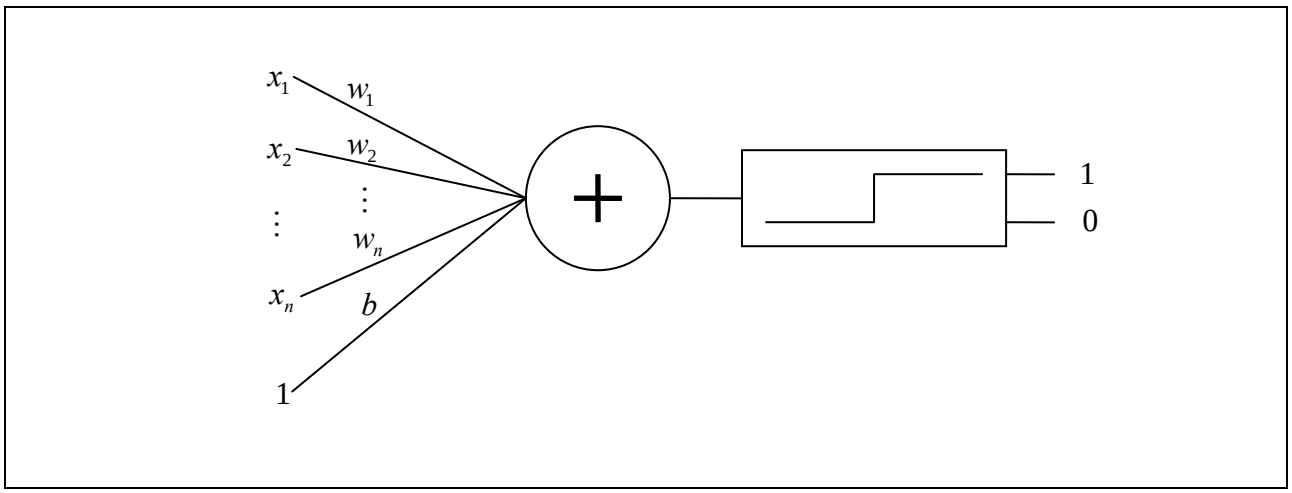


Abbildung 2: Perzeptron

Das Perzeptron entspricht einem Neuron im neuronalen Netz. Die Komponenten des Eingabevektors $x_1,...,x_n$ werden mit den Gewichten $w_1,...,w_n$ jeweils multipliziert und aufsummiert. Es existiert zusätzlich eine Konstante *b* die ebenfalls addiert wird. Diese Vorgehensweise kann auch als lineare Diskriminanzfunktion geschrieben werden:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b = b + \sum_{i=1}^{n} w_i x_i$$

Das Ergebnis der linearen Diskriminanzfunktion wird anschließend an eine Aktivierungsfunktion übergeben, die das Ergebnis der Diskriminanzfunktion in einen booleschen Wert transformiert: $g : R \to \{1,0\}$. Es ergibt sich damit die Diskriminanzfunktion:

$$c(\vec{x}) = g(f(\vec{x}))$$

Es ist nicht üblich nur lineare Aktivierungsfunktionen zu verwenden, auch nicht-lineare Funktionen können dabei eingesetzt werden:

$$g(t) = \frac{1}{1+e^{-t}} \qquad g(t) = \tanh(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}$$

Die Bestimmung der Gewichte kann durch die Perzeptron-Lernregel erfolgen, die an dieser Stelle jedoch nicht beschrieben werden soll. Interessierte seien auf die Bachelor Arbeit von Nico Pfeifer und mir (siehe [NPTB2004]) verwiesen.

2.2.3. Neuronale Netze durch mehrere Perzeptrons

Ein neuronales Netz entsteht, wenn mehrere Perzeptrons (Neuronen) miteinander vernetzt werden. Dazu ist es üblich, eine innere Schicht von *p* Neuronen einzuführen, die mit den Komponenten des Eingabevektors verbunden werden. Es entsteht folgendes Netz:

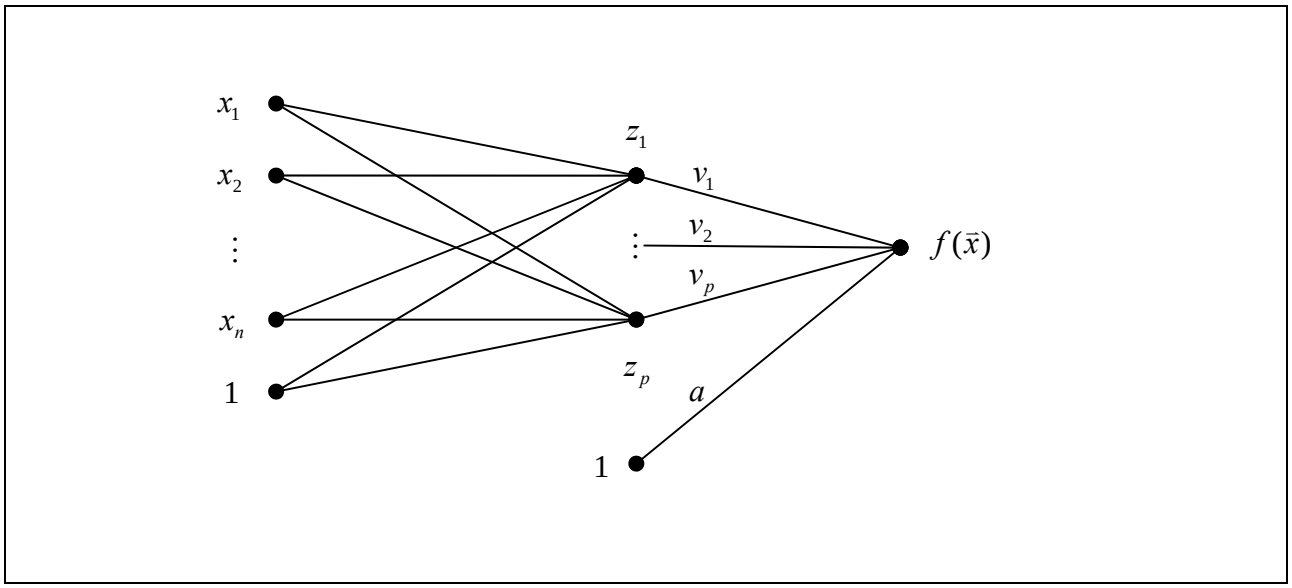


Abbildung 3: Neuronales Netz mit einer verdeckten Schicht

Das abgebildete neuronale Netz entspricht der folgenden Diskriminanzfunktion:

$$
\begin{aligned}
f(\vec{x}) &= \sum_{j=1}^{p} v_j z_j + a \\
&= \vec{v} \cdot \vec{z} + a \\
&= a + \sum_{j=1}^{p} v_j \cdot g\left(\sum_{i=1}^{n} w_{ij} \cdot x_i + b_j \right) \\
&= a + \sum_{j=1}^{p} v_j \cdot g\left(\vec{w}_j \cdot \vec{x} + b_j \right)
\end{aligned}
$$

Es ist möglich, auf diese Weise mehrere verdeckte Schichten einzuführen.

2.2.4. Funktionsapproximation mit neuronalen Netzen

Neuronale Netze mit einer verdeckten Schicht sind in der Lage, eine hinreichend glatte Funktion beliebig genau zu approximieren, sofern die Anzahl der Neuronen in der verdeckten Schicht groß genug ist. Das kann im Fall $n=1$ veranschaulicht werden. Liegt als Aktivierungsfunktion die Sigmoid-Funktion vor, so ergibt sich mit einer verdeckten Schicht von zwei Neuronen die folgende Funktion:

$$f(x) = a + \sum_{j=1}^{2} v_j \cdot g\left(w_j \cdot x + b_j\right)$$
$$= a + v_1 \cdot g\left(w_1 \cdot x + b_1\right) + v_2 \cdot g\left(w_2 \cdot x + b_2\right)$$

Es werden also dabei zwei Aktivierungsfunktionen eines transformierten Eingabewertes summiert. Dies entspricht anschaulich der folgenden, rot dargestellten Überlagerung:

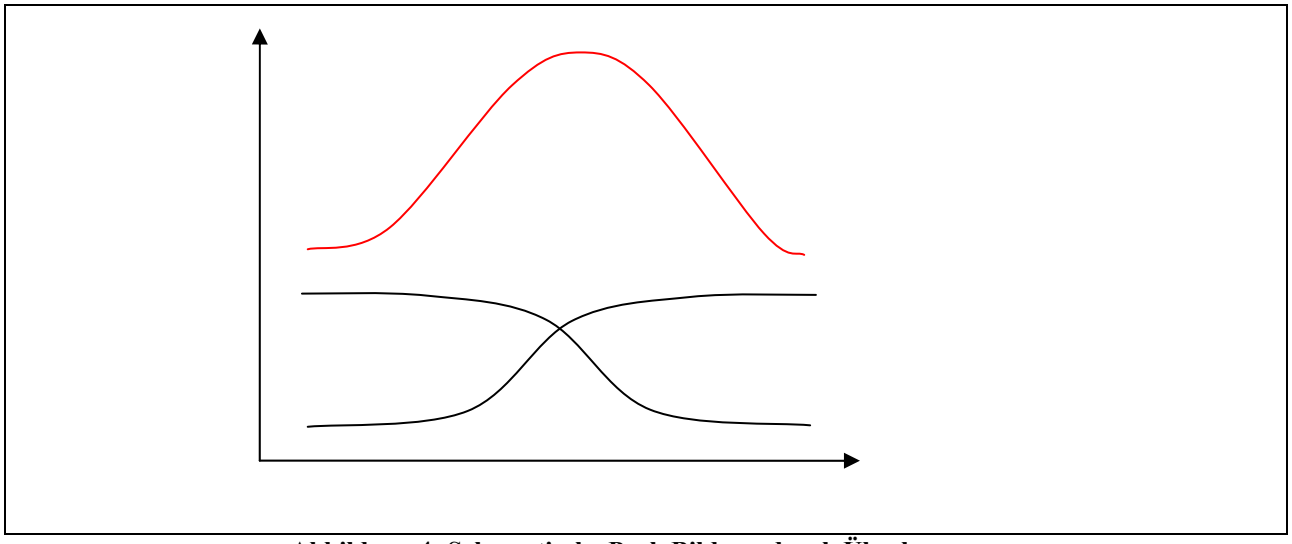


Abbildung 4: Schematische Peak Bildung durch Überlagerung

Mit ausreichend vielen lokalisierten Peaks lässt sich eine Zielfunktion *h(x)* beliebig genau approximieren:

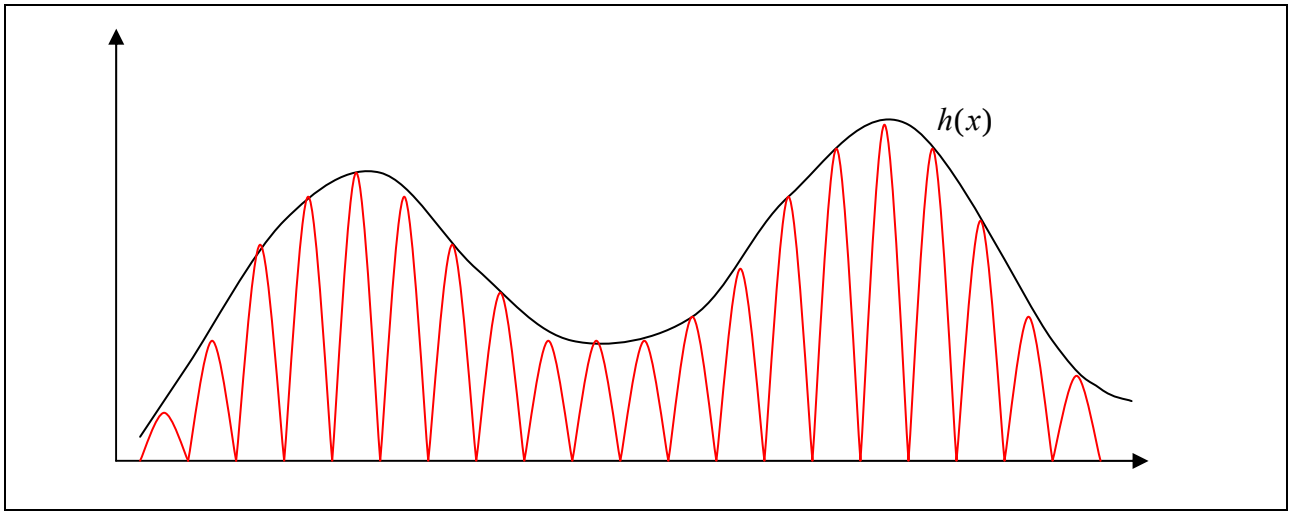


Abbildung 5: Schematische Funktionsapproximation mit Peaks

Aus diesem Grunde ist es zwingend erforderlich, dass die Neuronen der verdeckten Schicht, also der Parameter *p*, beschränkt wird. Andernfalls ist keine Generalisierung möglich, da für jedes Trainingsbeispiel ein eigener Peak erzeugt wird und so lediglich die Stichprobe auswendig gelernt wird.

2.2.5. Lernverfahren: Gradientenabstieg

Die Gewichtsvektoren in neuronalen Netzen müssen über ein Lernverfahren bestimmt werden. Der Ansatz dazu ist die Minimierung des Fehlerterms, der gegeben ist durch:

$$E\left(\vec{v}, W, \vec{b}, a\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f\left(\vec{x}_i\right)\right)$$

Eine häufig angewandte Variante ist der Gradientenabstieg. Dazu werden alle partiellen Ableitungen von *E* bezogen auf die Parameter mit der Kettenregel gebildet. Man erhält dann:

$$\nabla E = \begin{pmatrix} \partial E / \partial v_1 \\ \partial E / \partial v_2 \\ \vdots \end{pmatrix}$$

Durch ein iteratives Verfahren, das die Parameter in negativer Richtung des Gradienten abtastet, kann der Fehlerterm minimiert werden. Die Gefahr besteht jedoch, dass lediglich ein lokales und kein globales Minimum gefunden wird, da der Startpunkt an dem der „Abstieg beginnt“ von der Initialisierung der Parameter abhängt.

2.3. Nearest-Neighbor-Verfahren

Das Nearest-Neighbor-Verfahren gehört zu den instanzbasierten Verfahren. Der Klassifikator kann dabei auf die komplette Lernstichprobe zugreifen und liefert bei der Klassifikation eines ungesehenen Beispiels das Label des nächstgelegenen Trainingsbeispiels zurück. Dazu wird zunächst eine Metrik definiert:

$$d : R^m \times R^m \longrightarrow R$$

Häufig wird die euklidische Distanz angewendet:

$$d_{euclid}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Der Nearest-Neighbor-Klassifikator ist dann gegeben durch

$$f^{NB}(\vec{x}) = y_{ind(\vec{x})}$$

mit

$$ind(\vec{x}) = \arg\min_{i=1,\ldots,m} \{d(\vec{x}, \vec{x}_i)\}$$

Das Verfahren lässt sich mühelos auf *k*-Nearest-Neighbor erweitern, bei dem nicht nur der nächste Nachbar, sondern die *k* nächsten Nachbarn ausgewertet werden und das überwiegende Label zurückgegeben wird.

2.4. Multivariate Diskriminanzanalyse

Bei der multivariaten Diskriminanzanalyse (siehe [MMT71]) wird eine annährende Normalverteilung der vorliegenden Daten angenommen. Zunächst werden die Erwartungswerte für jede Komponente geschätzt:

$$\mu_j = \frac{1}{n}\sum_{k=1}^{n} x_{kj} \quad \text{für} \quad j = 1,...,n$$

sowie die Varianzen bzw. Kovarianzen:

$$Cov_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki} - \mu_i)(x_{kj} - \mu_j)$$

Die geschätzten Erwartungswerte werden in einen Vektor geschrieben:

$$\vec{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

Für jede vorliegende Klasse wird ein separater Erwartungswertvektor geschätzt:

$$\vec{\mu}_c = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix} \text{ für } c \in \{y_i \mid i = 1,...,n\}$$

Im Falle der Insolvenzerkennung ist zunächst $c = 2$. Anders wäre es bei Ratingsystemen, die jedoch nicht Gegenstand dieser Arbeit sind.
Ein ungesehenes Datum $\vec{x}$ wird nun jener Klasse zugeordnet, deren Verteilung $\vec{x}$ mit der höchsten Wahrscheinlichkeit erzeugt hat. Dies geschieht im Regelfall durch die Überprüfung, welchem Erwartungswert $\vec{x}$ am nächsten liegt. Auch hier ist ein Distanzmaß erforderlich, und es bietet sich zunächst die euklidische Distanz an.

Es gilt also allgemein:

$$f_d(\vec{x}) = \underset{\forall c \in \{y_i | i=1,\ldots,n\}}{\arg\min} d(\vec{x}, \vec{\mu}_c)$$

Im Falle hoher Varianzen kann es jedoch zu starken Überlappungen der Verteilungen kommen, so dass ein anderes Distanzmaß bevorzugt wird, nämlich die Mahalanobis-Distanz.
Dazu werden die geschätzten Varianzen und Kovarianzen zunächst in eine Matrix geschrieben:

$$C = \begin{pmatrix} Cov_{11} & Cov_{12} & \cdots & Cov_{1n} \\ Cov_{21} & Cov_{22} & \cdots & Cov_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Cov_{n1} & Cov_{n2} & \cdots & Cov_{nn} \end{pmatrix}$$

Diese so genannte Kovarianzmatrix ist stets symmetrisch und positivdefinit. Die Diagonalelemente entsprechen den einfachen Varianzen der einzelnen Merkmale, die restlichen Elemente den Kovarianzen.
Die Mahalanobis-Distanz ist dann gegeben durch:

$$d_{Mah}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T C^{-1} (\vec{x} - \vec{y})$$

Anschaulich wird jedes Merkmal zusätzlich mit den entsprechenden Varianzen und Kovarianzen normiert.
Die Wahl der Diskriminanzfunktion ist jedoch nicht verbindlich. Es gibt andere Diskriminanzfunktionen, wie zum Beispiel eine gewichtete Summation der einzelnen Merkmale:

$$s(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$$

Die Gewichte werden dann üblicherweise mit der Methode der kleinsten Quadrate bestimmt.
Es wird zusätzlich im Zweiklassenfall eine globalen Schwelle $\tau \in R$ eingeführt, so dass:

$$f_{sum}(\vec{x}) = \begin{cases} +1 & falls \quad s(\vec{x}) > \tau \\ -1 & sonst \end{cases}$$

2.5. Kendall-Verfahren

Das Kendall-Verfahren [KEN95] ist ein Verfahren zur Insolvenzerkennung, das eine Art Rangfolge von Kennzahlen ermittelt, die für eine Insolvenz entscheidend sind. Dazu wird für jedes gegebene Merkmal (Kennzahl) ein Intervall bestimmt, so dass alle Werte der Lernstichprobe oberhalb des Intervalls das Label -1 bzw. unterhalb das Label +1 tragen. Die Schwellen werden also so gewählt, dass eine fehlerfreie Klassifikation für einige Beispiele der Lernstichprobe möglich ist. Die verbleibenden Beispiele innerhalb des Intervalls (dem „Graubereich“) werden in einer weiteren Iteration für alle übrigen Merkmale demselben Verfahren unterzogen. Der folgende Algorithmus leistet das Gewünschte:

Algorithmus 1 (Kendall-Training)

Eingabe: $U[1,m] := ((\vec{x}_1, y_1),(\vec{x}_2, y_2),...,(\vec{x}_m, y_m))$

Ausgabe: Eine Folge von Intervallen und Merkmalsindizes als Klassifikator

$$S_0 \leftarrow \bigcup_{j=1}^{n} \{x_{1j}, x_{2j},..., x_{mj}\} \times \{x_{1j}, x_{2j},..., x_{mj}\} \times \{j\}$$

Für $i = 1,2,\ldots,n$ führe aus:

$$(\tau_i^{left}, \tau_i^{right}, j_i) \leftarrow \underset{(\tau^{left}, \tau^{right}, j) \in S}{\arg\max} \#\{\vec{x}_k \mid ((x_{kj} < \tau^{left} \wedge y_j = +1) \vee (x_{kj} > \tau^{right} \wedge y_j = -1)) \wedge k \in \{1,...,m\}\}$$

$$S_i \leftarrow \{(\tau^{left}, \tau^{right}, j) \mid (\tau^{left}, \tau^{right}, j) \in S_{i-1} \wedge j \neq j_i\}$$

Return $((\tau_1^{left}, \tau_1^{right}, j_1),(\tau_2^{left}, \tau_2^{right}, j_2),...,(\tau_n^{left}, \tau_n^{right}, j_n))$

Die Laufzeit des Algorithmus ist $O(m^2 n^2)$. Allerdings lässt sich das Verfahren in der Praxis optimieren, da ein Abbruch der Hauptschleife erfolgen kann, sobald alle noch zur Verfügung stehenden Merkmale lediglich zur Graubereichsklassifikation führen.

Um ein ungesehenes Beispiel mit dem Kendall-Verfahren zu klassifizieren, wird die Liste der Merkmale durchlaufen und das Label der ersten Schwellenkombination vergeben, die es nicht in den Graubereich einordnet.

Der folgende Algorithmus leistet das Gewünschte:

Algorithmus 2 (Kendall-Klassifikation)

Eingabe: $T := \left(\left(\tau_1^{left}, \tau_1^{right}, j_1\right), \left(\tau_2^{left}, \tau_2^{right}, j_2\right), \ldots, \left(\tau_n^{left}, \tau_n^{right}, j_n\right)\right)$ und ein Beispiel $\vec{x}$

Ausgabe: Ein Label $y \in \{-1,+1\}$ für $\vec{x}$

Für $i = 1,2,\ldots,n$ führe aus:

Falls $x_{j_i} > \tau_i^{right}$ so führe aus:

Return -1

Falls $x_{j_i} < \tau_i^{left}$ so führe aus:

Return +1

Return 0 // Nicht klassifizierbar

Es kann Fälle geben, in denen das Kendallverfahren keine Klassifikation liefert. Dies ist allerdings nicht zwingend ein Nachteil, da im Falle der Insolvenzvorhersage eine falsch-negative Vorhersage kritischer sein kann, als keine Aussage zu treffen. Es besteht jedoch die Gefahr, dass diese Aussage zu häufig vorkommt.
Ein weiterer Nachteil des Kendall-Verfahrens ist die Ausreißeranfälligkeit. Es ist möglich, dass die gewählten Intervallschwellen nur extreme Ausreißer ausgrenzen, daher keinerlei Generalisierungseffekt liefern und damit nahezu jedes andere Beispiel in den Graubereich fällt. Außerdem wird die Bewertung eines neuen Beispiels faktisch an einem einzigen Merkmal vorgenommen (Rangefolge), da das erste Merkmal der Liste zur Klassifikation gewählt wird, das das Beispiel nicht dem Graubereich zuordnet. Daher sind Wechselwirkungen zwischen Merkmalen in diesem Verfahren nicht erfasst.

2.6. Support Vector Maschinen mit Kernen

Support-Vektor-Maschinen (SVMs) sind effektive Maschinen zur Klassifikation, die sich zum einen durch hohe Effizienz auszeichnen und aufgrund der Verwendung von Kern-Funktionen weniger anfällig für Overfitting-Probleme sind. Aufgrund dieser Eigenschaften sind sie im Regelfall neuronalen Netzen überlegen. An dieser Stelle soll nur das grobe Prinzip von SVMs

beschrieben werden, Interessierte seien auf die Bachelorarbeit von Nico Pfeifer und mir (siehe [NPTB2004]) bzw. auf eine Arbeit von Schölkopf (siehe [SS02]) verwiesen.
Bei SVMs wird der Ansatz linearer Trennungen von Daten verfolgt, da lineare Funktionen leicht zu handhaben sind. Das Problem liegt jedoch klar auf der Hand, da die Flexibilität von linearen Funktionen meist nicht ausreichend ist, um eine angemessene Datentrennung zu erzielen. Daher wird eine Anreicherung des Eingaberaums durch Redundanz vorgenommen, was häufig mit einer Dimensionserhöhung verbunden ist. Dies wird formal als Abbildung vom Eingaberaum in den so genannten Merkmalsraum aufgefasst, in dem wiederum versucht wird die Daten linear zu trennen:

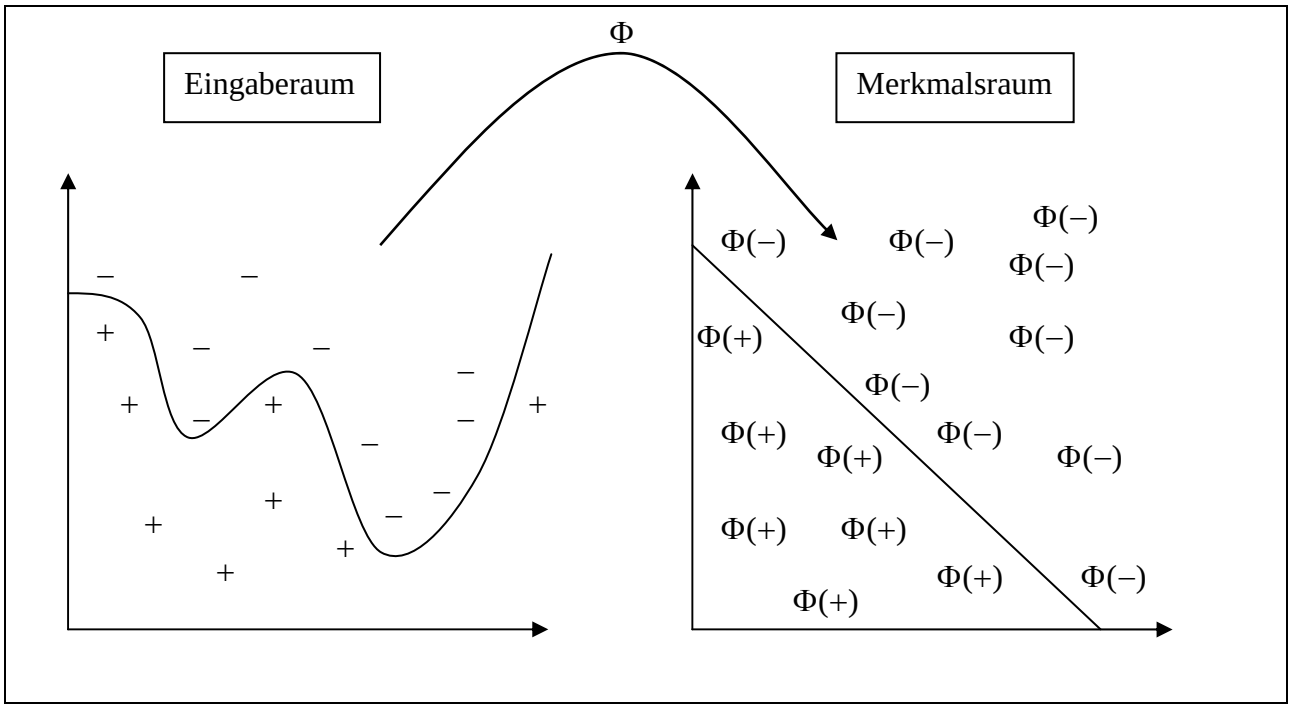


Abbildung 6: Schematische Abbildung in den Merkmalsraum

Die primale Form ist dabei wie folgt gegeben:

$$f(\vec{x}) = \vec{w}_F \cdot \Phi(\vec{x}) + b$$

Dies entspricht einer linearen Funktion im Merkmalsraum. Der Eingabevektor $\vec{x}$ wird in den Merkmalsraum abgebildet und ein entsprechend höher dimensionaler Gewichtsvektor $\vec{w}$ ist zu lernen, sowie eine Verschiebung in Y-Richtung *b.* Mit:

$$\vec{w}_F = \sum_{i=1}^{m} \alpha_i \cdot \Phi(\vec{x}_i)$$

folgt schließlich die duale Form:

$$f(\vec{x}) = \sum_{i=1}^{m} \alpha_i \cdot \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b$$

Dies kann so interpretiert werden, dass der gesuchte Gewichtsvektor eine Linearkombination aus gewichteten Trainingsbeispielen ist. In diesem Fall müssen die α_i für $i = 1,...,m$ bestimmt werden. Der Vorteil der SVM liegt darin, dass bei einer linearen Trennung nur die Vektoren der Trainingsstichprobe relevant sind, die der gesuchten Ebene am nächsten liegen. Alle anderen Vektoren tragen nichts zur Lage der Ebene bei, daher werden diese relevanten Vektoren auch als Supportvektoren bezeichnet. Im Rahmen der SVM bedeutet dies, dass viele der α_i gleich null gesetzt werden können und somit eine höhere Effizienz gewährleistet ist.
Das noch zu lösende Problem ist jedoch, dass der Merkmalsraum häufig hochdimensional ist, was wiederum zu schlechter Laufzeit führen kann, sowie die Entstehung von Overfitting Problemen begünstigt. Aus diesem Grunde wird das Skalarprodukt der Vektoren im Merkmalsraum der dualen Form durch eine Kernfunktion ersetzt, die die Abbildung in den Merkmalsraum implizit ausführt. Es folgt schließlich:

$$\begin{aligned} f(\vec{x}) &= \sum_{i=1}^{m} \alpha_i \cdot \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b \\ &= \sum_{i=1}^{n} \alpha_i \cdot k(\vec{x}_i, \vec{x}) + b \end{aligned}$$

Die einzige Möglichkeit in den Prozess des Lernens einer SVM einzugreifen, stellt somit die Kernfunktion dar. Es existieren allerdings viele, leistungsfähige Standardkerne, die in den meisten Fällen eine ausreichend gute Trennung der Daten ermöglichen. Folglich macht es Sinn, diese Kernfunktionen zuerst zu verwenden, bevor sich die Entwicklung eigener Kerne lohnt.

2.7. Regressionsanalysen: Logit- und Probit-Modell

2.7.1. Einführung

Ein gegenwärtig häufig gebrauchtes Modell ist das so genannte Logit- bzw. Probit-Modell auf Basis der linearen Regression (siehe [D06]). Der Ansatz ist relativ ähnlich zu einer multivariaten

Diskriminanzanalyse, daher wird dieses Verfahren lediglich vorgestellt und nicht auf reale Daten angewendet.

2.7.2. Maximum-Likelihood-Verfahren zur Parameterschätzung

Das Maximum-Likelihood-Verfahren (siehe [PW98]) kann zur Parameterschätzung verwendet werden, sofern die Dichte bzw. Wahrscheinlichkeitsfunktion einer kontinuierlichen Zufallsvariablen X von einem festen Parameter $q \in R$ abhängt. Dieser Parameter soll so geschätzt werden, dass die in einer Stichprobe beobachteten Werte für das Label $Y \in \{-1,+1\}$ in Abhängigkeit von X mit maximaler Wahrscheinlichkeit auftreten.
Im Falle von zwei Klassen kann von einem *m*-fachen Bernoulli-Experiment ausgegangen werden und entsprechend wird die Wahrscheinlichkeit dafür, das Label +1 in Abhängigkeit von X zu beobachten, wie folgt definiert:

$$P(Y = +1) = q$$

Werden in einer Stichprobe der Länge *m* genau $k \in N^+$ Beispiele mit dem Label $+1$ beobachtet, so ist nach der Binomialverteilung die Wahrscheinlichkeit dieses Ereignisses gegeben durch:

$$Bin(m,k,q) = \binom{m}{k}(q)^k (1-q)^{m-k}$$

Durch Maximierung dieser Wahrscheinlichkeit, wird so eine bestmögliche Schätzung des Parameters *q* erzielt. Um das Maximum zu finden, ist der obige Term nach *q* zu differenzieren. Durch die Bestimmung der Nullstellen der ersten Ableitung kann dieses Problem gelöst, und der Parameter *q* bestimmt werden. Allgemein ist die Maximum-Likelihood-Funktion gegeben durch:

$$L(q) = \prod_{i=1}^{m} F_{x_{i1}}(x_i; q)$$

Wobei F_{x_i} eine Dichte- bzw. Wahrscheinlichkeitsfunktion darstellt. Das Verfahren kann analog für die Schätzung von mehreren Parametern erweitert werden.

2.7.3. Logit-Modell

Bei der linearen Regression kann nicht sichergestellt werden, dass die Ausgabe einer binären Klassifikation entspricht. Eine einfache Skalierung führt dabei zu einer Verringerung der Klassifikationsleistung. Aus diesem Grund wird das Logit-Modell eingeführt.
Der Ansatz beruht auf Wahrscheinlichkeitsquotienten, den sogenannten Odds:

$$Odd(Y_i) = \frac{P(Y_i = +1)}{P(Y_i = -1)} = \frac{P(Y_i = +1)}{1 - P(Y_i = +1)} \qquad \text{für } i = 1,...,m$$

Es ist leicht nachvollziehbar, dass die Odds lediglich Werte zwischen $+\infty$ und 0 annehmen können. Durch Einführung von Gewichten $w_0,...,w_n$ sei:

$$z_i = w_0 + \sum_{j=1}^{n} w_j x_{ij} \qquad \text{für } i = 1,...,m$$

Die Logistische Regressionsfunktion wird dann definiert als:

$$P(Y_i = +1 \mid \vec{X}_i = \vec{x}_i) = \frac{e^{z_i}}{1 + e^{z_i}} = \frac{1}{1 + e^{-z_i}} \qquad \text{für } i = 1,...,m$$

Durch Logarithmierung des Terms folgt:

$$\begin{aligned}
\ln(Odd(Y_i)) &= \ln\left(\frac{P(Y_i=+1)}{1-P(Y_i=+1)}\right) \\
&= \ln\left(\frac{\frac{1}{1+e^{-z_i}}}{1-\frac{1}{1+e^{-z_i}}}\right) \\
&= \ln\left(\frac{1}{\left(1+e^{-z_i}\right)-\left(\frac{1+e^{-z_i}}{1+e^{-z_i}}\right)}\right) \\
&= \ln\left(\frac{1}{1+e^{-z_i}-1}\right) \\
&= \ln\left(e^{z_i}\right) \\
&= z_i
\end{aligned}$$

Entsprechend kann zur Schätzung der Gewichte $w_0,...,w_n$ für die Variablen z_i für $i=1,...,m$ das Maximum-Likelihood-Verfahren angewendet werden.

2.7.4. Probit-Modell

Im Probit-Modell werden, im Gegensatz zum Logit-Modell, die Wahrscheinlichkeiten anhand der Standardnormalverteilung geschätzt:

$$P(Y_i=+1 \mid X_i=\vec{x}_i)=\Phi(z)=\int_{-\infty}^{z}\frac{1}{2\pi}\cdot\exp\left(\frac{z^2}{2}\right)dz \quad \text{mit } z=w_0+\sum_{j=1}^{n}w_j x_{ij} \text{ für } i=1,...,m$$

Das Vorgehen ist dann analog zum Logit-Modell.

3. Theoretische Hintergründe des PAC-Lernens

In diesem Kapitel soll das Modell des PAC-Lernens (siehe [Val98]) vorgestellt und an einem einfachen Beispiel angewendet werden. Das Akronym „PAC“ steht dabei für „Probably Approximately Correct“. Weiterhin werden Formalien zur Analyse eingeführt, und die grundlegenden Prinzipien beschrieben.

3.1. Einführung der Formalien

3.1.1. Lernuniversum

Seien $\aleph \subset R^n$ und $\Lambda \subset \{-1,+1\}$ der Eingaberaum, bzw. der binäre Ausgaberaum. Dann ist das Produkt $\Im = \aleph \times \Lambda$ das Lernuniversum. Die Menge Λ repräsentiert die möglichen Labels im Falle der binären Klassifikation (Diskrimination). Das Zufallselement X induziert auf $\aleph$ eine Verteilung P_X, die zugesicherte Eigenschaften haben kann. Es gilt im Weiteren die Schreibweise $P_X(B) = P(X \in B)$ für jede messbare Menge B.

3.1.2. Konzeptklasse und Hypothesenklasse

Die Konzeptklasse C wird als Menge von Funktionen $f : \aleph \to \Lambda$ aufgefasst und stellt anschaulich die Menge von „Lernzielen“ dar. Ein festes Zielkonzept $g \in C$ liefert dann zu einem Vektor des Eingaberaums ein Label aus der Menge Λ.
Die Hypothesenklasse H ist häufig gleich der Konzeptklasse. Sie stellt die Menge von Funktionen dar, aus der eine möglichst zum Zielkonzept passende ausgewählt werden soll. Diese Auswahl übernimmt ein für die Konzeptklasse C entwickelter Lernalgorithmus.

3.1.3. Lernstichprobe

Einem Lernalgorithmus A wird eine Lernstichprobe vorgesetzt, die einer beliebigen Verteilung P_X folgt:

$$U[1,m] := ((X_1, Y_1), (X_2, Y_2), ..., (X_m, Y_m))$$

Anhand dieser Lernstichprobe soll eine Hypothese $f \in H$ berechnet werden, die das gesuchte Zielkonzept möglichst gut approximiert. Entscheidend dabei ist, dass die Elemente der Lernstichprobe unabhängig und identisch verteilt sind.
Im Falle des PAC-Lernens existiert das zuvor beschriebene Zielkonzept *g*, das nach den Modellannahmen die Labels anhand des jeweiligen Datenvektors erzeugt hat. Es gilt also $Y_i = g(X_i)\ \forall i \in \{1,2,...,m\}$. Dies ist der entscheidende Punkt des PAC-Lernens, denn das Zielkonzept sichert zu, dass eine Hypothese existiert, die eine vollständig korrekte Klassifikation der Trainingsdaten erzielt. Die Lernstichprobe hängt also in diesem Modell auch von dem Zielkonzept *g* ab:

$$U^g[1,m] := \big((X_1, g(X_1)), (X_2, g(X_2)), ..., (X_m, g(X_m))\big)$$

3.1.4. Fehlerschranke und Unsicherheit

Zur Bewertung einer Hypothese *f* die von einem Lernalgorithmus *A* ausgegeben wird, ist die Fehlerfunktion für jede Verteilung P_X auf $\aleph$ wie folgt definiert:

$$err(f) := P(g(X) \neq f(X))$$

Der Fehler einer Hypothese *f* ist also die Wahrscheinlichkeit dafür, dass eine zufällige Eingabe durch *f* falsch klassifiziert wird. Dies entspricht somit der Abweichung vom zu lernenden Zielkonzept *g*.
Zunächst erscheint es logisch, dass bei wachsender Stichprobenlänge *m* der Fehler einer von *A* berechneten Hypothese *f* gegen null konvergiert. Es wäre also möglich, für jede Fehlerschranke $\varepsilon \in (0,1)$ eine Stichprobenlänge $m_0 = m_0(\varepsilon)$ so anzugeben, dass, wenn die Länge der gegebenen Stichprobe größer oder gleich m_0 ist, der Fehler kleiner oder gleich ε ist.
Diese Forderung ist jedoch nicht zu erfüllen, da auch die Zusammensetzung der Stichprobe zufällig ist. Im schlimmsten Fall ist anhand unrepräsentativer Daten gelernt worden, und der tatsächliche Fehler auf ungesehenen Daten ist dadurch hoch. Aus diesem Grund wird eine zweite Schranke, die so genannte Unsicherheit $\delta \in (0,1)$ eingeführt. Dies ist äquivalent zu:

$$P(err(f) \geq \varepsilon) \leq \delta$$

Es ist also die Wahrscheinlichkeit dafür, dass der Lernalgorithmus *A* bei beliebiger Verteilung P_X des Zufallselementes *X* auf $\aleph$ und beliebigem Zielkonzept *g,* angesetzt auf eine Lernstichprobe $U^g[1,m]$ mit $m \geq m_0$ eine Hypothese *f* berechnet, deren Fehler größer oder gleich $\varepsilon \in (0,1)$ nach oben durch $\delta \in (0,1)$ beschränkt.

3.2. Eigenschaften von PAC-Lernalgorithmen

In diesem Kapitel werden notwendige Eigenschaften für PAC-Lernalgorithmen vorgestellt (siehe [Fis99]).

3.2.1. Konsistenz

Ein Lernalgorithmus der Konzeptklasse *C* durch die Hypothesenklasse *H* heißt konsistent, wenn es für jede Fehlerschranke $\varepsilon > 0$, jede Unsicherheit $\delta > 0$ und jeden Dimensionsparameter $n \in N^+$ eine minimale Lernstichprobenlänge $m_0 = m_A(\varepsilon, \delta, n)$ gibt, so dass für jede Verteilung P_X auf dem Eingaberaum $\aleph$, jedes Zielkonzept $g \in C$ und jedes $m \geq m_0$ gilt: Angesetzt auf eine Lernstichprobe $U^g[1,m]$ zum Zielkonzept *g* der Länge *m* gibt der Lernalgorithmus *A* eine Hypothese $A(U^g[1,m])$ aus, so dass gilt:

$$P(err(A(U^g[1,m])) \geq \varepsilon) \leq \delta$$

Die Funktion $m_A(\varepsilon, \delta, n)$ heißt Lernstichprobenkomplexität.

3.2.2. Effizienz

Ein konsistenter Lernalgorithmus *A* von *C* durch *H* heißt effizient, wenn
für die Lernstichprobenkomplexität

$$m_A(\varepsilon, \delta, n) = \left\lceil \frac{1}{\varepsilon} \right\rceil^{O(1)} \cdot \left\lceil \frac{1}{\delta} \right\rceil^{O(1)} \cdot n^{O(1)}$$

gilt, und der Lernalgorithmus polynomiell in der Eingabe, also der Größe der Lernstichprobe, arbeitet.

3.2.3. Widerspruchsfreiheit

Ein Lernalgorithmus *A* heißt widerspruchsfrei, wenn *A* auf jeder Lernstichprobe eine Hypothese *f* berechnet, die jedes Beispiel der Lernstichprobe korrekt klassifiziert.

3.3. PAC-Lernen am Beispiel

Zum besseren Verständnis sollen die Definitionen aus Kapitel 3.1. und 3.2. an einem einfachen Beispiel angewendet werden. Es wird dazu ein Lernalgorithmus vorgestellt, der ein achsenparalleles Viereck lernt (siehe [Fis99]).

3.3.1. Problemstellung und Algorithmus

Als Konzeptklasse *C* liegen alle achsenparallelen Vierecke der reellen Ebene vor. Anhand eines Zufallselementes *X* aus $\aleph \subset R^2$ soll ein Lernalgorithmus *A* eine möglichst gute Approximation eines Zielkonzepts $g \in C$ liefern. Die Hypothesenklasse ist in diesem Fall gleich der Konzeptklasse, also *C = H.* Der folgende Algorithmus leistet das Gewünschte.

Algorithmus 3 (SER)

Eingabe: $U^g[1,m] := \left(\left(X_1 = \left(X_{11}, X_{12}\right), g(X_1)\right), ..., \left(X_m = \left(X_{m1}, X_{m2}\right), g(X_m)\right)\right)$

Ausgabe: Ein achsenparalleles Viereck $F = A(U^g[1,m])$ als Hypothese

$V_{\min} \leftarrow \min\{X_{i1} \mid i = 1,2,...,m \wedge g(X_i) = 1\}$ // Kleinste X-Koordinate

$V_{\max} \leftarrow \max\{X_{i1} \mid i = 1,2,...,m \wedge g(X_i) = 1\}$ // Größte X-Koordinate

$W_{\min} \leftarrow \min\{X_{i2} \mid i = 1,2,...,m \wedge g(X_i) = 1\}$ // Kleinste Y-Koordinate

$W_{\max} \leftarrow \max\{X_{i2} \mid i = 1,2,...,m \wedge g(X_i) = 1\}$ // Größte Y-Koordinate

Return $[V_{\min}, V_{\max}] \times [W_{\min}, W_{\max}]$ // 4 Eckpunkte des Vierecks

Der Algorithmus *SER* berechnet also ein achsenparalleles Viereck, das alle positiven Beispiele umfasst. Das Zielkonzept *g* sichert dabei zu, dass ein solches existiert und damit kein negatives Beispiel innerhalb des Vierecks liegen kann (Dies ist ausgeschlossen, da die Konzeptklasse nur aus achsenparallelen Vierecken besteht). Der Lernalgorithmus kann also aufgrund dieser

Bedingung die Verarbeitung von negativen Beispielen komplett ignorieren und ist außerdem ein *widerspruchsfreier* Hypothesenfinder. Es gilt demgemäß $F(X_i) = g(X_i)\ \forall i \in \{1,2,...,m\}$.

3.3.2. Beweis: Konsistenz von SER

Behauptung:

Für jede Fehlerschranke $\varepsilon > 0$, jede Unsicherheit $\delta > 0$, jede Verteilung P_X auf dem Eingaberaum $\aleph$ und jedes achsenparallele Rechteck g gilt: Ist die Stichprobenlänge $m \geq \left\lceil \frac{4}{\varepsilon} \cdot \ln\left(\frac{4}{\delta}\right) \right\rceil$, so gilt:

$$P(err(SER(U^g[1,m])) \geq \varepsilon) \leq \delta$$

Beweis:

Sei die Dichte der Verteilung P_X stetig.

Sei weiterhin $g = [l,r] \times [b,t]$ das zu lernende Zielkonzept, dann sind die Randrechtecke gegeben durch

$$g_1 = [l,r^*] \times [b,t] \qquad g_2 = [l,r] \times [b,t^*] \qquad g_3 = [l^*,r] \times [b,t] \qquad g_4 = [l,r] \times [b^*,t]$$

mit

$$r^* = \inf\left\{x \mid P_X([l,x] \times [b,t]) \geq \frac{\varepsilon}{4}\right\} \qquad t^* = \inf\left\{y \mid P_X([l,r] \times [b,y]) \geq \frac{\varepsilon}{4}\right\}$$

$$l^* = \sup\left\{x \mid P_X([x,r] \times [b,t]) \geq \frac{\varepsilon}{4}\right\} \qquad b^* = \sup\left\{y \mid P_X([l,r] \times [y,t]) \geq \frac{\varepsilon}{4}\right\}$$

Da das von SER als Hypothese bestimmte Rechteck alle positiven Beispiele der Lernstichprobe umfasst, liegt es stets innerhalb des Zielrechtecks *g* (Zielkonzept).

Der Bereich zwischen der Hypothese und dem Zielkonzept entspricht dem Fehlerbereich:

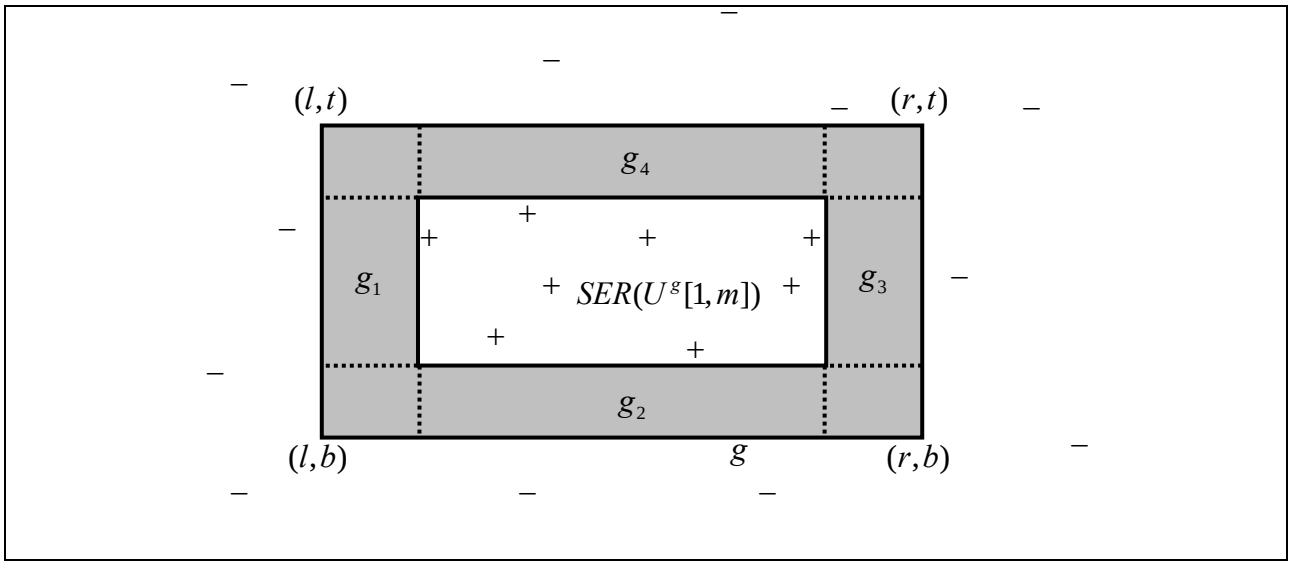


Abbildung 7: Fehlerbereich (grau) der Ausgabe des Algorithmus SER zum Zielkonzept g

Aufgrund der stetigen Dichte der Verteilung P_X folgt:

$$P_X(g_1) = P_X(g_2) = P_X(g_3) = P_X(g_4) = \frac{\varepsilon}{4}$$

Für die Fehlermenge gilt dann:

$$G^* \subseteq \bigcup_{i=1}^{4} g_i$$

und somit:

$$P_X(G^*) \leq \varepsilon$$

Die Frage ist nun, wie lang eine Stichprobe sein muss, damit dieses Ereignis mit Wahrscheinlichkeit $1-\delta$ auftritt. Dazu kann das komplementäre Ereignis nach oben abgeschätzt werden: Wie groß muss *m* gewählt werden, damit die Wahrscheinlichkeit dafür, dass mindestens eines der vier Randvierecke durch *kein* positives Beispiel der Lernstichprobe getroffen wird, kleiner als δ ist?

Die Wahrscheinlichkeit für ein Element der Lernstichprobe ein festes Randviereck nicht zu treffen, ist nach oben durch $1-\left(\frac{\varepsilon}{4}\right)$ per Definition beschränkt. Somit ist die Wahrscheinlichkeit, dass dies für alle Beispiele der Lernstichprobe gilt gegeben durch: $\left(1-\left(\frac{\varepsilon}{4}\right)\right)^m$, da die Beispiele der Lernstichprobe unabhängig sind. Bezieht man alle vier Randrechtecke ein, so ist die gesuchte Wahrscheinlichkeit nach oben durch $4\cdot\left(1-\left(\frac{\varepsilon}{4}\right)\right)^m$ beschränkt. Da $e^x > 1+x \;\forall x \neq 0$ folgt schließlich:

$$4\cdot\left(1-\left(\frac{\varepsilon}{4}\right)\right)^m < 4\cdot e^{-m\cdot\frac{\varepsilon}{4}}$$

Nun kann diese Wahrscheinlichkeit durch δ beschränkt, und die Ungleichung nach *m* aufgelöst werden:

$$\begin{aligned} & 4\cdot e^{-m\cdot\frac{\varepsilon}{4}} \leq \delta \\ \Leftrightarrow \quad & 4\cdot\frac{1}{\delta} \leq e^{m\cdot\frac{\varepsilon}{4}} \\ \Leftrightarrow \quad & \ln\left(\frac{4}{\delta}\right) \leq m\cdot\frac{\varepsilon}{4} \\ \Leftrightarrow \quad & \frac{4}{\varepsilon}\cdot\ln\left(\frac{4}{\delta}\right) \leq m \end{aligned}$$

q.e.d.

Da die Stichprobenlänge durch ein Polynom in $\left\lceil\frac{1}{\delta}\right\rceil$, $\left\lceil\frac{1}{\varepsilon}\right\rceil$ und n gegeben ist, und der Algorithmus SER in $O(m)$ arbeitet, folgt daraus auch die Effizienz von SER.

3.4. Pattern Klassen

3.4.1. Einführung

In diesem Kapitel sollen Patternklassen (siehe [Kol01]) vorgestellt werden. Ähnlich wie bei dem PAC-Lernen, dient das bekannte Lernuniversum $\Im$ als Grundlage, jedoch wird der Ansatz hier stärker verallgemeinert. Zum einen gilt $\Lambda \subset R$.

Ziel ist es, ein Pattern π aus einer Patternklasse $\mathrm{P} \subset Abb(\Im,[a,b])$ mit $0 \le a < b$ zu bestimmen, so dass die Datenquelle $U = (X,Y)$ möglichst gut approximiert wird. Patterns werden also als Abbildungen der Datenquelle auf ein kompaktes Intervall aufgefasst.

In diesem Fall werden nur Patternklassen betrachtet, die aus einer Hypothesenklasse *H* und einer Verlustfunktion *l* gebildet werden können, also:

$$\mathrm{P} = \mathrm{P}(H,l) = \{l(y, f(x)) \mid f \in H\}$$

Dabei ist:

$$l : \Lambda \times \Lambda \to R^+ \cup \{0\}$$

Es sollen für *l* die folgenden Eigenschaften erfüllt sein:

$$\begin{aligned} l(y,y) &= 0 && \forall y \in \Lambda \\ l(y,y') &\ge 0 && \forall y, y' \in \Lambda \\ l(y,y') &= l(y',y) && \forall y, y' \in \Lambda \end{aligned}$$

Gebräuchliche Verlustfunktionen sind:

- Die quadratische Verlustfunktion $l^{quad}(y_1,y_2) = (y_1 - y_2)^2$
- Die 0/1-Verlustfunktion $l^{0/1}(y_1,y_2) = \begin{cases} 1 & falls\ y_1 \neq y_2 \\ 0 & sonst \end{cases}$

Die höchste Passgenauigkeit für ein Pattern $\pi \in \mathrm{P}$ ist gegeben, wenn

$$P(\pi(U) = 0) = 1$$

gilt. Es ist leicht nachzuvollziehen, dass ein exaktes Pattern bei gestörten Daten nicht existieren kann.
Das Risiko eines Patterns ist dessen Generalisierungsfehler

$$risk(\pi) := E\pi(U)$$

bzw.

$$risk(f) := P(f(X) \neq Y)$$

bei fester 0/1-Fehlerfunktion. Das Risiko einer Patternklasse P ist die größte untere Schranke des Generalisierungsfehlers:

$$risk(\mathrm{P}) := \inf_{\pi \in \mathrm{P}} risk(\pi)$$

Beispiel: Pattern-Klasse des PAC-Lernens
Sei:

- $U = (X, Y) \in \mathfrak{I} = \aleph \times \Lambda \subset \{+1, -1\}^{n+1}$
- $Y = g(X)$ für ein $g \in C$
- $\mathrm{P}(H, l^{0/1})$ über die Hypothesenklasse H und 0/1-Verlustfunktion

Ein exaktes Pattern $\pi \in \mathrm{P}$ für die Datenquelle U wäre dann gegeben durch $l^{0/1}(y, g(x))$. In diesem Fall ist auch das Risiko gleich dem Fehler, also $risk(f) = err(f)$.

Beispiel: Pattern-Klasse des PAC-Lernens mit Rauschen auf der Klassifikation
Sei

- $U = (X, Y) \in \mathfrak{I} = \aleph \times \Lambda \subset \{+1, -1\}^n \times \{+1, -1\}$
- $Y = g(X) \cdot B$ für ein $g \in C$
- $B \in \{+1, -1\}$ eine unabhängig verteilte Zufallsvariable mit $P(B = -1) = \eta \leq \eta_0 < 1/2$

Dann gilt:

$$risk\left(\mathrm{P}(H, l^{0/1}\right) = risk\left(l^{0/1}(y, g(x))\right) = \eta$$

Demnach ist $risk(f) \neq err(f)$. Das Zielkonzept selbst hat also einen Fehler von null, aber ein Risiko von η. Im Falle des PAC-Lernens mit Rauschen auf der Klassifikation wird das Zielkonzept *g* mit Hilfe der Fehlerfunktion *err* approximiert. Dazu muss auch das Rauschen herausgerechnet werden. Das Ziel ist schärfer als bei der Patternanalyse, bei der es lediglich darum geht, die vorliegenden Daten zu approximieren. Wie in diesem Fall vorgegangen werden muss, wird in Kapitel 3.8 diskutiert.

3.4.2. Patternanalyse-Algorithmen

Ähnlich wie in dem Kapitel des PAC-Lernens vorgestellt, liegt eine unabhängige und identisch verteilte Stichprobe vor:

$$U[1,m] := \left(\left(X_1, g(X_1)\right), \left(X_2, g(X_2)\right), ..., \left(X_m, g(X_m)\right)\right)$$

Ein Lernalgorithmus *A* durch eine Patternklasse P ist dann wie folgt spezifiziert:

Eingabe: Eine Lernstichprobe $U[1,m]$ der Länge *m*

Ausgabe: Ein zufälliges Pattern $\Pi_A = A(U[1,m])$. Ist die Verlustfunktion *l* fest gewählt, so wird lediglich eine zufällige Hypothese $F_A \in H$ ausgegeben

Die Algorithmen sollen außerdem einige Zusatzqualifikationen aufweisen. Zum einen soll bei der Eingabe unterschiedlicher Lernstichproben derselben Datenquelle ein ähnliches Pattern ausgegeben werden. Vor allem aber soll das ausgegebene Pattern die Datenquelle beschreiben und demnach auch ungesehene Beispiele mit hoher Zuverlässigkeit vorhersagen können (statistische Stabilität). Außerdem sind reale Daten häufig Störungen unterworfen, die den Algorithmus nicht über alle Maßen negativ beeinflussen dürfen. Weiterhin ist gefordert, dass der Algorithmus effizient, also polynomiell in Abhängigkeit von der Länge der Lernstichprobe *m*, arbeitet.

3.4.3. Konsistenz

Ein Lernalgorithmus A durch P heißt konsistent, wenn es für jede Fehlerschranke $\varepsilon > 0$, jede Unsicherheit $\delta > 0$ eine minimale Lernstichprobenlänge $m_0 = m_A(\varepsilon, \delta)$ gibt, so dass für jede Verteilung P_X auf dem Eingaberaum $\aleph$, jedes Zielkonzept $g \in C$ und jedes $m \geq m_0$ gilt: Angesetzt auf eine Lernstichprobe $U^g[1,m]$ zum Zielkonzept g der Länge m gibt der Algorithmus A eine Hypothese $A(U^g[1,m])$ aus, so dass gilt:

$$P(err(A(U^g[1,m])) \geq \varepsilon) \leq \delta$$

Die Funktion $m_A(\varepsilon, \delta)$ heißt Lernstichprobenkomplexität.

3.5. Rademacher-Komplexität

Der folgende Abschnitt ist aus verschiedenen Quellen (siehe [BM01], [BM02] und [Kol01]) entnommen.

Sei $R[1,m] = (R_1, ..., R_m)$ eine Folge von unabhängigen und gleichverteilten Zufallsvariablen, den so genannten Rademachervariablen, über $\{-1,+1\}$. Es gilt dabei für $1 \leq i < j \leq m$:

$$E(R_i) = 0 \qquad Var(R_i) = 1 \qquad R_i^{\,2} = 1 \qquad E(R_i R_j) = 0$$

Die Rademacher-Stichprobenfunktion bildet dann den Zufallsvektor $(U[1,m], R[1,m])$ auf eine Zufallsvariable ab:

$$rsf_m^{\mathrm{P}}(U[1,m], R[1,m]) = \frac{2}{m} \sup_{\pi \in \mathrm{P}} \left(\sum_{i=1}^{m} \pi(U_i) \cdot R_i \right)$$

Es wird also ein Pattern der Patternklasse P so gewählt, dass die Summe über die Fehlerwerte dieses Patterns, multipliziert mit der jeweiligen Rademacher-Variablen maximal werden. Das Pattern muss also stets so gewählt werden, dass $\pi(U_i)$ einen kleinen bzw. großen Wert annimmt, falls $R_i = -1$ bzw. $R_i = +1$ ist, um den Term zu maximieren. Ist demnach der Wert der Rademacherstichprobenfunktion hoch, so ist dies ein Zeichen für die Erfassung reinen Zufalls, denn die durch das Pattern produzierten Fehlerwerte hängen nur wenig bis gar nicht von den

vorliegenden Beispielen der Lernstichprobe ab; die Anpassung an die Rademachervariablen ist hingegen hoch. Es ist intuitiv klar, dass bei steigender Stichprobenlänge *m* die Wahl eines Patterns, das besagte Eigenschaften besitzt, schwieriger wird. Entsprechend wird die Rademacher-Komplexität kleiner. Steht hingegen eine größere Auswahl von Patterns zur Verfügung, so ist eine bessere Anpassung an die Rademacher-Variablen möglich.
Die empirische Rademacher-Komplexität ist gegeben durch:

$$rc_m(\mathrm{P} \mid u[1,m]) = \frac{2}{m} E\left(\sup_{\pi \in \mathrm{P}} \left(\sum_{i=1}^{m} \pi(u_i) \cdot R_i \right) \right)$$

Es wird also der Erwartungswert der Rademacher-Stichprobenfunktion gebildet.
Die Rademacher-Komplexität ist dann der Erwartungswert der empirischen Rademacher-Komplexität:

$$rc_m(\mathrm{P}) = \int rc_m(\mathrm{P} \mid u[1,m]) dP_{U[1,m]}(u[1,m])$$

Die Rademacher-Komplexität erlaubt eine Aussage über das Potential einer Pattern-Klasse P in den vorliegenden Daten reinen Zufall zu erfassen. Demnach führt eine Vergrößerung der Patternklasse auch stets zu einer Vergrößerung der empirischen Rademacher-Komplexität sowie der Rademacher-Komplexität selbst.
Im Falle endlicher Patternklassen kann gezeigt werden, dass die Rademacher-Komplexität wie folgt beschränkt ist:

$$rc_m(\mathrm{P} \mid U[1,m]) \le \frac{c}{2\sqrt{m}} \sqrt{2 \cdot \ln|\mathrm{P}|} \qquad \text{bzw.} \quad rc_m(\mathrm{P}) \le \frac{c}{2\sqrt{m}} \sqrt{2 \cdot \ln|\mathrm{P}|}$$

Mit $c = b - a$.

3.6. Empirische Risiko-Minimierung

Im Folgenden wird das Grundprinzip erläutert, anhand dessen ein Lernalgorithmus *A* durch P zu entwerfen ist.

Dazu wird der Begriff des empirischen Risikos eingeführt, das anhand der Lernstichprobe bestimmt wird:

$$risk_{emp}(\pi \mid U[1,m]) := \frac{1}{m}\sum_{i=1}^{m}\pi(U_i)$$

mit

$$\pi(U[1,m]) = \left(\pi(U_1), \pi(U_2), ..., \pi(U_m)\right).$$

Weiterhin ist:

$$E\, risk_{emp}(\pi \mid U[1,m]) = risk(\pi).$$

Mit Hilfe der Schranke von McDiarmid (siehe [McD89]) kann die fast sichere Konvergenz des empirischen Risikos gegen das Risiko gezeigt werden. Das empirische Risiko der Patternklasse P zur Instanz $u[1,m]$ der Lernstichprobe $U[1,m]$ ist dann gegeben durch:

$$risk_{emp}(\mathrm{P}, u[1,m]) = \inf_{\pi \in \mathrm{P}} risk_{emp}(\pi \mid u[1,m])$$

Es gilt weiterhin für das Risiko einer Patternklasse:

$$E\left(risk_{emp}(\mathrm{P} \mid U[1,m])\right) \le risk(\mathrm{P})$$

Das Induktionsprinzip der empirischen Risiko-Minimierung besagt dann, dass auf jeder Instanz *u[1,m]* der Lernstichprobe *U[1,m]* ein Pattern, dessen empirisches Risiko für diese Instanz eine Approximation des empirischen Risikos $risk_{emp}(\mathrm{P} \mid u[1,m])$ der gesamten Patternklasse P mit einer vom Fehler ε abhängigen Güte ist.
Ein Lerner hat lediglich eine Lernstichprobe zur Verfügung. Aus diesem Grund besteht eine große Chance, dass das Verhalten auf der Lernstichprobe besser ist als im Mittel. Diese Tatsache steht der Konsistenz des Lernalgorithmus im Wege.

Ein Maß dafür ist der so genannte Auswuchs einer Patternklasse P:

$$\sup_{\pi \in \mathrm{P}} \left(risk(\pi) - risk_{emp}(\pi \mid U[1,m]) \right)$$

Aufgrund dieser Formel wird deutlich, dass die uniforme stochastische Konvergenz des empirischen Risikos gegen das Risiko für $m \to \infty$ eine Voraussetzung für die Konsistenz eines Lernalgorithmus ist. Durch Anwendung der Schranke von McDiarmid (siehe [McD89]) auf den Auswuchs folgt:

$$P\left(\sup_{\pi \in \mathrm{P}} \left(risk(\pi) - risk_{emp}(\pi \mid U[1,m]) \right) - E\left(\sup_{\pi \in \mathrm{P}} \left(risk(\pi) - risk_{emp}(\pi \mid U[1,m]) \right) \right) \geq \in \right) \leq \exp\left(\frac{-2\varepsilon^2 m}{c^2} \right)$$

3.7. Hauptsatz der Rademacher-Theorie

Der Hauptsatz besagt, dass das empirische Risiko eines Patterns auf einer gegebenen Lernstichprobe für ausreichende Lernstichprobengröße gegen das tatsächliche Risiko konvergiert (siehe [Kol01]).

Sei P eine Patternklasse zum Lernuniversum $\Im$ und sei $k > 0$ eine natürliche Zahl.

Ist A_k ein Lernalgorithmus durch P, der für jedes natürliche $m > 0$ angesetzt auf jede beliebige Instanz $u[1,m]$ der Lernstichprobe $U[1,m]$ ein Pattern $A_k(u[1,m])$ so ausgibt, dass

$$risk_{emp}\left(A_k(U[1,m]) \mid u[1,m]\right) \leq (1+1/k)\, risk_{emp}(\mathrm{P} \mid u[1,m])$$

ist, so gilt für jede Unsicherheit $\delta \in (0,1)$ und jede Lernstichprobenlänge $m > 0$:

1. Mit Wahrscheinlichkeit $1-\delta$ sind die folgenden beiden Ungleichungen zusammen erfüllt:

$$risk\left(A_k(U[1,m])\right) - risk(\mathrm{P}) < rc_m(\mathrm{P}) + \left(2c + \frac{c}{k}\right)\sqrt{\frac{\ln(2/\delta)}{2m}} + \frac{c}{k}$$

$$risk_{emp}\left(A_k(U[1,m]) \mid U[1,m]\right) - risk(\mathrm{P}) < \left(c + \frac{c}{k}\right)\sqrt{\frac{\ln(2/\delta)}{2m}} + \frac{c}{k}$$

2. Mit Wahrscheinlichkeit $1-\delta$ sind die folgenden beiden Ungleichungen zusammen erfüllt:

$$risk\left(A_k\left(U[1,m]\right)\right)-risk\left(\mathrm{P}\right)<rc_m\left(\mathrm{P}\mid U[1,m]\right)+\left(4c+\frac{c}{k}\right)\sqrt{\frac{\ln\left(4/\delta\right)}{2m}}+\frac{c}{k}$$

$$risk_{emp}\left(A_k\left(U[1,m]\right)\mid U[1,m]\right)-risk\left(\mathrm{P}\right)<\left(c+\frac{c}{k}\right)\sqrt{\frac{\ln\left(2/\delta\right)}{2m}}+\frac{c}{k}$$

Der Hauptsatz verknüpft die Rademacher-Komplexität mit der Konsistenz eines Lernalgorithmus.
Die Kernaussage des Hauptsatzes ist, dass für gegen null strebende Rademacher-Komplexität einer Patternklasse P das Prinzip der empirischen Risiko-Minimierung zu einem konsistenten Lernalgorithmus führt.

3.8. PAC-Lernen mit Rauschen auf der Klassifikation

3.8.1. Einführung

Wie in den vorherigen Kapiteln beschrieben, zeichnet sich das PAC-Lernen durch die Existenz eines Zielkonzepts aus, was garantiert, dass stets eine Hypothese *f* gefunden werden kann, die jedes Beispiel der Lernstichprobe korrekt klassifiziert, sofern $H \subseteq C$ gilt. Bei Grundlegung realer Daten ist jedoch leicht nachvollziehbar, dass eine fehlerfreie Klassifikation nicht erreicht werden kann, zumindest nicht im Rahmen einer Konzeptklasse, die nicht nur das Auswendiglernen von Stichproben gestattet. Dies ist in der Praxis vor allem darauf zurückzuführen, dass die Daten nicht anhand des vorliegenden Modells erzeugt werden. Aus diesem Grunde wird im Rahmen des PAC-Lernens dieser Sachverhalt mit einem zusätzlichen Zufallsexperiment auf der Ausgabevariablen modelliert. Dazu wird ein einfaches Bernoulli-Experiment für jedes Beispiel der Lernstichprobe ausgeführt. Bei Erfolg wird das vorliegende Label invertiert. Die Erfolgswahrscheinlichkeit für dieses Bernoulli-Experiment wird *Rauschrate* genannt und ist im Folgenden durch $\eta < \frac{1}{2}$ bezeichnet. Bisher konnte vorausgesetzt werden, dass:

$$\varphi_Y(y \mid X = x) = \begin{cases} 1 & falls\ y = g(x) \\ 0 & sonst \end{cases}$$

Mit der Einführung der Rauschrate gilt nun:

$$\varphi_Y(y \mid X = x) = \begin{cases} 1-\eta & falls\ y = g(x) \\ \eta & sonst \end{cases}$$

Das Zufallsexperiment sieht dann wie folgt aus:

- Führe das zu dem Eingabeelement X zugehörige Zufallsexperiment aus
- Transformiere das Ergebnis x unter Verwendung des Zielkonzepts g zu $g(x) \in \{-1,+1\}$
- Führe ein unabhängiges Bernoulli-Experiment mit Erfolgswahrscheinlichkeit η aus. Falls dieses erfolgreich ist, so invertiere das Label aus dem vorherigen Schritt

Der Umgang mit verrauschten Daten beruht auf der Idee, dass überwiegend das korrekte Label gesehen wird. Diese Tatsache verschlechtert sich, je weiter sich die Rauschrate an den Wert $1/2$ annähert. Es wird weiterhin angenommen, dass die tatsächliche Rauschrate nicht bekannt ist. Sie ist aber durch eine reelle Konstante $\eta_0 \in [0,1/2[$ nach oben beschränkt, so dass gilt: $\eta \leq \eta_0$. Die Lernaufgabe ist nun erschwert, da der PAC-Lernalgorithmus lediglich verrauschte Daten zur Verfügung hat, um das Zielkonzept g zu approximieren. Da das Zielkonzept g jedoch unbekannt ist, kann kein „empirischer Fehler“ berechnet werden, und somit ist eine einfache Anwendung des Induktionsprinzips der empirischen Risiko-Minimierung nicht möglich.
Das zu erreichende Ziel geht aus der folgenden Definition hervor.

3.8.2. PAC-Lernalgorithmus mit Rauschen auf der Klassifikation

Ein Lernalgorithmus der Konzeptklasse *C* durch die Hypothesenklasse *H* heißt *PAC-Lernalgorithmus mit Rauschen auf der Klassifikation,* wenn es für jede Fehlerschranke $\varepsilon > 0$, jede Unsicherheit $\delta > 0$ und jeden Dimensionsparameter $n \in N^+$ eine minimale Lernstichprobenlänge $m_0 = m_A(\varepsilon, \delta, n, \eta_0)$ gibt, so dass für jede Verteilung P_X auf dem Eingaberaum $\aleph$, jedes Zielkonzept $g \in C$ und jedes $m \geq m_0$ gilt: Angesetzt auf eine Lernstichprobe $U^g[1,m]$ zum Zielkonzept *g* der Länge *m* gibt der Algorithmus *A* eine Hypothese $A(U^g[1,m])$ aus, so dass gilt:

$$P(err(A(U^g[1,m])) \geq \varepsilon) \leq \delta$$

3.8.3. Effizienz

Ein *PAC-Lernalgorithmus mit Rauschen auf der Klassifikation A* von *C* durch *H* heißt effizient, wenn für die Lernstichprobenkomplexität

$$m_A(\varepsilon, \delta, \eta_0) = \left\lceil \frac{1}{\varepsilon} \right\rceil^{O(1)} \cdot \left\lceil \frac{1}{\delta} \right\rceil^{O(1)} \cdot n^{O(1)} \cdot \left\lceil \frac{1}{1-2\eta_0} \right\rceil^{O(1)}$$

gilt, und der Lernalgorithmus polynomiell in der Eingabe, also der Größe der Lernstichprobe, arbeitet.

3.8.4. Konsistenz bei PAC-Lernen mit Rauschen auf der Klassifikation

Es kann gezeigt werden, dass im allgemeinen Fall gilt:

$$err(f) \geq risk(f) - \eta \geq (1-2\eta) \cdot err(f)$$

Im Falle des PAC-Lernens mit Rauschen auf der Klassifikation gilt:

$$err(f) = \frac{risk(f) - \eta}{1-2\eta}$$

Aus diesen Gleichungen lässt sich ableiten:

$$risk(f) - \eta < \varepsilon \cdot (1 - 2\eta_0) \Rightarrow err(f) < \varepsilon$$

Das bedeutet also, dass der Fehler $err(f)$, nämlich der Abweichung einer Hypothese *f* vom Zielkonzept, durch ε beschränkt ist, sofern das Risiko durch $\varepsilon \cdot (1 - 2\eta_0) + \eta$ beschränkt ist. Durch die uniforme stochastische Konvergenz des empirischen Risikos gegen das Risiko folgt, dass das Prinzip der empirischen Risiko-Minimierung auch im Falle des PAC-Lernens mit Rauschen auf der Klassifikation zu einem konsistenten Lernalgorithmus führt.

3.9. Schwache empirische Risiko-Minimierung

Die Minimierung des empirischen Risikos ist häufig NP-schwer, so dass das Induktionsprinzip nicht effizient angewendet werden kann. Eine Möglichkeit, diesem Problem zu begegnen, ist der Einsatz des Statistical Query Modells von Kearns (siehe [Kea98]). Dieser Ansatz löst jedoch keineswegs ein NP-schweres Problem, sondern schränkt das Modell von der Seite der zugrunde liegenden Verteilung ein. Das Problem wird somit stark vereinfacht und ist nicht länger NP-schwer. Das empirische Risiko wird in diesem Fall lediglich für typische Eingaben minimiert und nicht für alle möglichen Eingaben. In der Praxis hat sich dieser Ansatz jedoch bewährt. Das Vorgehen wird im folgenden Kapitel beschrieben.

3.9.1. Arbeitsweise des Lernalgorithmus

Im Falle NP-schwerer Optimierungsprobleme der empirischen Risiko-Minimierung ist es möglich, diese mit einem Approximationsalgorithmus zu lösen. Häufig ist es jedoch der Fall, dass kein Polynomialzeit Approximationsschema existiert oder letzteres nicht bewiesen werden kann. Deshalb kann das folgende Vorgehen als Prinzip der schwachen empirischen Risiko-Minimierung aufgefasst werden. Ein Lernalgorithmus *A* wird dazu in zwei Phasen eingeteilt:

Selektionsphase

Die Lernstichprobe wird unterteilt, so dass $U[1,m] = U[1,m_1]U[m_1+1,m_2]$ gilt.

Anschließend wird die Patternklasse P zu $\mathrm{P}(U[1,m_1])$ anhand der ersten Hälfte der Lernstichprobe verkleinert, so dass gilt:

$$rc_{m_2}(\mathrm{P}(u[1,...,m_1] \mid u[1,...,m_2]) \leq \frac{\sigma_A(\varepsilon,\delta,n)}{\sqrt{m_2}}$$

Dabei ist $u[1,m] = u[1,m_1]u[m_1+1,m_2]$ eine Instanz der Lernstichprobe und $\sigma_A(\varepsilon,\delta,n,s)$ polynomiell beschränkt durch $1/\delta$, $1/\varepsilon$, n und s.
Weiterhin muss gesichert sein, dass eine minimale Stichprobengröße $m_{A,1}(\varepsilon,\delta,n,s)$ existiert, so dass für $m_1 \geq m_{A,1}(\varepsilon,\delta,n,s)$, jedes Zielkonzept der Größe von höchstens s und jede Verteilung P_X von X die Menge $\mathrm{P}(U[1,m_1]) \subset \mathrm{P}$ eine Hypothese h mit Wahrscheinlichkeit größer $\frac{1-\delta}{2}$ enthält, so dass $err(f) \leq \frac{\varepsilon}{2}$.
Die praktische Umsetzung der Selektionsphase erfolgt durch Statistical Queries (siehe Kapitel 4.7) die anhand der vorliegenden Daten simuliert werden.

Minimierungsphase
Die verbleibenden Beispiele der Lernstichprobe werden genutzt, um aus der verkleinerten Patternklasse $\mathrm{P}(U[1,m_1])$ eine Hypothese $H \in \mathrm{P}(U[1,m_1])$ zu berechnen, so dass:

$$risk_{emp}(H \mid U[m_1+1,m_2]) = \inf_{f \in \mathrm{P}(U[1,m_1])} risk_{emp}(f \mid U[m_1+1,m_2])$$

3.9.2. Abtasten der Rauschrate

Die im vorherigen Kapitel vorgestellten Simulationen für Statistical Queries setzen die Kenntnis der Rauschrate voraus. Um diese zu bestimmen, werden verschiedene Rauschraten η geraten und anhand einer zu dieser Rauschrate berechneten Hypothese f verifiziert. Ist $err(f) \leq \varepsilon$, so ist die geratene Rauschrate hinreichend genau. In der Praxis kann das Raten der Rauschrate nach Kearns und Aslam (siehe [Kea98]) durch eine Einteilung des Intervalls $[0,\eta_0]$ erfolgen:

$$\beta_*(\varepsilon) = \left\lfloor \frac{3\eta_0}{\gamma_*(1-2\eta_0)(\sqrt{2}-1)} \right\rfloor$$

Dabei ist $\gamma_* = \gamma(\varepsilon)$ eine untere Schranke für den Genauigkeitsparameter einer Statistical Query. Die Mittelpunkte dieser Teilintervalle werden schließlich nacheinander abgetastet.

3.10. Statistical Queries

3.10.1. Einführung

Das Statistical Query Model ist von Kearns (siehe [Kea98]) und Aslam (siehe [Asl95]) eingeführt worden. Die nachfolgenden Kapitel sind vollständig aus diesen Quellen entnommen. Lernalgorithmen durch Statistical Queries lernen nicht anhand einzelner Beispiele einer Lernstichprobe, sondern an globalen, statistischen Eigenschaften der zugrunde liegenden Verteilung. Aus diesem Grund erhält der Lernalgorithmus keinen Zugriff auf die Lernstichprobe selbst, sondern stattdessen die Möglichkeit, Anfragen an ein Orakel zu stellen. Dazu ist eine Query-Funktion der folgenden Form erforderlich:

$$\chi : \aleph \times \Lambda \to [a, b]$$

Der Lernalgorithmus kann dann eine *Statistical Query* (SQ) $[\chi, \alpha]$ für eine Schätzung $\hat{e}_\chi$ des Erwartungswerts:

$$e_{\chi,g} = E\chi(X, g(X))$$

stellen, so dass $\left| e_{\chi,g} - \hat{e}_{\chi,g} \right| \leq \alpha$ gilt. Es kann also davon ausgegangen werden, dass die Schätzung auf α genau ist.
Eine *Conditional Statistical Query* (CSQ) ist gegeben durch:

$$e_{\chi,g,y_0} = E\chi(X, g(X) \mid g(X) = y_0)$$

Da in der Praxis keine Datenquelle vorliegt, sondern lediglich Daten, die außerdem verrauscht sind, müssen die obigen Erwartungswerte anhand der vorliegenden Lernstichprobe geschätzt werden. Dazu wird zunächst vorausgesetzt, dass die Rauschrate η bekannt ist.

3.10.2. Simulation einer Statistical Query

Behauptung:

$$e_{\chi,g} = E\chi(X, g(X)) = \frac{(1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y)}{1-2\eta}$$

Beweis:

Sei:

I. $E\chi(X,Y) = (1-\eta)E\chi(X, g(X)) + \eta E\chi(X,-g(X))$

II. $E\chi(X,-Y) = (1-\eta)E\chi(X,-g(X)) + \eta E\chi(X, g(X))$

Nach Multiplikation der Gleichung I. mit $1-\eta$ bzw. Gleichung II. mit η folgt:

I. $(1-\eta)E\chi(X,Y) = (1-\eta)^2 E\chi(X, g(X)) + \eta(1-\eta)E\chi(X,-g(X))$

II. $\eta E\chi(X,-Y) = (1-\eta)\eta E\chi(X,-g(X)) + \eta^2 E\chi(X, g(X))$

Nach Subtraktion der Gleichung II. von Gleichung I. folgt:

$$(1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y) = (1-\eta)^2 E\chi(X, g(X)) - \eta^2 E\chi(X, g(X))$$

$$\Leftrightarrow \quad (1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y) = E\chi(X, g(X)) \cdot \left((1-\eta)^2 - \eta^2\right)$$

$$\Leftrightarrow \quad \frac{(1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y)}{\left((1-\eta)^2 - \eta^2\right)} = E\chi(X, g(X))$$

$$\Leftrightarrow \quad \frac{(1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y)}{\left(1-2\eta+\eta^2 - \eta^2\right)} = E\chi(X, g(X))$$

$$\Leftrightarrow \quad E\chi(X, g(X)) = \frac{(1-\eta)E\chi(X,Y) - \eta E\chi(X,-Y)}{1-2\eta}$$

q.e.d.

Die folgenden Werte können nach Aslam (siehe [Asl95]) direkt aus der vorliegenden Lernstichprobe geschätzt werden:

$$e_{\chi} = E\chi(X,Y) \qquad e_{\bar{\chi}} = E\chi(X,-Y)$$

mit $\bar{\chi}(x,y) = \chi(x,-y)$.

3.10.3. Berechnung der Labelwahrscheinlichkeiten

Behauptung:

$$P(g(X)=y_0)=\frac{P(Y=y_0)-\eta}{1-2\eta}$$

Beweis:

Die Labelwahrscheinlichkeiten sind wie folgt gegeben:

I. $P(Y=-1)=P(g(X)=-1)(1-\eta)+P(g(X)=+1)\eta$

II. $P(Y=+1)=P(g(X)=-1)\eta+P(g(X)=+1)(1-\eta)$

Der Beweis wird vollständig analog zu Kapitel 3.10.2 geführt.

q.e.d.

3.10.4. Simulation einer Conditional Statistical Query

Behauptung:

$$e_{\chi,g,y_0}=E\chi(X,g(X)\mid g(X)=y_0)=\frac{(1-\eta)E\big(\delta(Y,y_0)\chi(X,Y)\big)-\eta E\big(\delta(-Y,y_0)\chi(X,-Y)\big)}{P(Y=y_0)-\eta}$$

Beweis:

Die Kronecker-Funktion ist wie folgt definiert:

$$\delta(i,j)=\begin{cases}1 & falls\ i=j\\ 0 & sonst\end{cases}$$

Aufgrund der Formel für die bedingte Wahrscheinlichkeit folgt:

$$E\chi(X,g(X)\mid g(X)=y_0)=\frac{E\big(\delta(g(X),y_0)\chi(X,g(X))\big)}{P(g(X)=y_0)}$$

Unter Anwendung der in Kapitel 3.10.2 hergeleiteten Gleichung gilt:

$$E\big(\delta(g(X),y_0)\chi(X,g(X))\big)=\frac{(1-\eta)E\big(\delta(Y,y_0)\chi(X,Y)\big)-\eta E\big(\delta(-Y,y_0)\chi(X,-Y)\big)}{1-2\eta}$$

Mit $P(g(X)=y_0)=\frac{P(Y=y_0)-\eta}{1-2\eta}$ folgt schließlich:

$$E\chi(X,g(X)\mid g(X)=y_0)=\frac{(1-\eta)E\big(\delta(Y,y_0)\chi(X,Y)\big)-\eta E\big(\delta(-Y,y_0)\chi(X,-Y)\big)}{\frac{P(Y=y_0)-\eta}{1-2\eta}\cdot(1-2\eta)}$$

$$\Leftrightarrow \qquad E\chi(X,g(X)\mid g(X)=y_0)=\frac{(1-\eta)E\big(\delta(Y,y_0)\chi(X,Y)\big)-\eta E\big(\delta(-Y,y_0)\chi(X,-Y)\big)}{P(Y=y_0)-\eta}$$

q.e.d.

Die folgenden Werte können direkt aus der vorliegenden Lernstichprobe geschätzt werden:

$$e_{\chi,y_0}=E\big(\delta(Y,y_0)\chi(X,Y)\big) \qquad e_{\bar{\chi},y_0}=E\big(\delta(-Y,y_0)\chi(X,-Y)\big) \qquad e_{y_0}=P(Y=y_0)$$

4. Entwicklung einer Konzeptklasse zur Insolvenzerkennung

4.1. Klassifikation durch Schwellwerte

In diesem Kapitel soll die Grundidee der Klassifikation mit Hilfe von Schwellwerten vorgestellt werden. Außerdem werden dazu notwendige Grundlagen und Eigenschaften aus der booleschen Algebra diskutiert, die für die Entwicklung der Algorithmen relevant sind.

4.1.1. Einführung und grundlegende Idee

Die Grundlage des Klassifikators bilden Schwellwerte nach einem einfachen Prinzip: Liegt der Wert eines Merkmals oberhalb bzw. unterhalb des zugehörigen Schwellwertes, so wird dies als positives bzw. negatives Indiz gewertet. Dieses Vorgehen bildet in etwa das Prinzip ab, nach dem ein Unternehmen durch eine Kennzahl beurteilt werden soll. Der Ansatz ist nicht neu und wird bereits im Kendall-Verfahren (siehe Kapitel 2.5) in ähnlicher Weise angewendet. Bei einem einzigen Merkmal ist der Fall klar, bei mehreren hingegen muss eine Kombination zugehöriger Schwellwerte gefunden werden und eine Kombination zu einem Gesamturteil erfolgen. Es bietet sich an, den Ansatz weiter zu verallgemeinern und boolesche Funktionen zu verwenden, wie etwa disjunktive und konjunktive Normalformen. Um mögliche Kombinationen von Kennzahlen sinnvoll abzubilden, ist eine disjunktive Normalform vorzuziehen. Auf diese Weise kann die Idee abgebildet werden, dass unterschiedliche Kombinationen von Kennzahlen für die Insolvenz eines Unternehmens eine Rolle spielen können. Um diese Idee umzusetzen, wird das PAC-Lernen mit Rauschen auf der Klassifikation angewendet.

4.1.2. Formalisierung

Die Jahresabschlüsse von *m* Unternehmen werden als Elemente einer Lernstichprobe aufgefasst. Dabei entspricht der Dimensionsparameter *n* der Anzahl der zur Verfügung stehenden Kennzahlen. Analog zu Kapitel 3 ergibt sich:

$$U^g[1,m] := \left(\left(X_1 = \left(X_{11},...,X_{1n}\right), g(X_1)\right),...,\left(X_m = \left(X_{m1},...,X_{mn}\right), g(X_m)\right)\right)$$

Dabei wird im Falle des PAC-Lernens mit Rauschen auf der Klassifikation angenommen, dass nur verrauschte Daten beobachtet werden. Das tatsächliche Zielkonzept hat also zunächst „wahre“ Labels erzeugt, die dann mit Wahrscheinlichkeit η invertiert werden. In Algorithmen

wird, zur deutlicheren Unterscheidung zwischen Vektoren und einzelnen Merkmalen (Komponenten) die folgende Schreibeweise bevorzugt:

$$u[1,m] := \left((\vec{x}_1, y_1),...,(\vec{x}_m, y_m)\right)$$

mit

$$(\vec{x}_i, y_i) = \left((x_{i1},...,x_{in}), y_i\right) \quad \forall i = 1,...,n$$

sowie

$$(\vec{x}, y) = \left((x_1,...,x_n), y\right)$$

Die Schwellwertfunktionen zu einer Schwelle $\tau \in R$ sind dann definiert als:

$$\mathrm{T}_{i,\tau}^{above}(\vec{x}) = \begin{cases} +1 & falls \quad x_i > \tau \\ -1 & sonst \end{cases} \qquad \mathrm{T}_{i,\tau}^{below}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \leq \tau \\ -1 & sonst \end{cases}$$

Im Folgenden werden die Typen *above* und *below* der Schwellwertfunktionen als Schwellwertfunktionstypen bezeichnet. Alternativ können die Schwellwerte zu Intervallen $[a,b]$ mit $a,b \in R$ erweitert werden. Es ergeben sich die Intervallfunktionen:

$$\mathrm{I}_{i,[a,b]}^{above}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \in [a,b] \\ -1 & sonst \end{cases} \qquad \mathrm{I}_{i,[a,b]}^{below}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \notin [a,b] \\ -1 & sonst \end{cases}$$

Diese lassen sich mit Hilfe einer Heuristik (siehe Kapitel 4.5) analog zu Schwellwerten lernen, indem Paare (a,b) anstelle von Schwellen τ abgetastet werden. Zur Vereinfachung, und da Intervallfunktionen eine untergeordnete Rolle spielen sollen, werden im Folgenden nur Schwellwertfunktionen behandelt.
Die Ergebnisse der Funktionen entsprechen den booleschen Wahrheitswerten nach der Vorschrift:

$$+1 = true \qquad -1 = false$$

In der Praxis ist es erforderlich, die zur Auswahl stehenden Schwellwerte auf eine endlich abzählbare Menge für jedes Merkmal zu beschränken:

$$S_i = \{\tau \mid \tau \in R\} \text{ für } i = 1,...,n \text{ mit } \#S_i \in N$$

Wie genau diese Mengen zu wählen sind, wird später diskutiert. Letztlich stehen folgende Schwellwertfunktionen zur Auswahl:

$$\Gamma = \left\{ \mathrm{T}_{i,\tau}^{above}, \mathrm{T}_{i,\tau}^{below} \mid \tau \in S_i \wedge i \in \{1,...,n\} \right\}$$

Es ist klar, dass:

$$\#\Gamma = 2 \cdot \sum_{i=1}^{n} \#S_i$$

Die mit Hilfe der Schwellwertfunktion diskretisierten Merkmale werden schließlich zu einem Monom bzw. einer Klausel vereinigt:

$$\mu = \alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_p$$

$$c = \alpha_1 \vee \alpha_2 \vee ... \vee \alpha_p$$

mit

$$\alpha_i \in \Gamma \text{ für } i = 1,...,p \text{ und } p \in N$$

Die α_i werden in Anlehnung an die boolesche Algebra auch als Literale bezeichnet.

Da zu Beginn unklar ist, ob der Wert eines Merkmals oberhalb oder unterhalb einer Schwelle ein Indikator für die eine oder andere Klasse ist, muss einer der beiden Schwellwertfunktionstypen gewählt werden. Es dürfen also nicht beide Typen im selben Monom bzw. Klausel erscheinen. Anders ausgedrückt: Jedes Merkmal darf höchstens einmal in einem Monom bzw. einer Klausel vorkommen.

Weiterhin gilt:

$$\text{Falls } \alpha_i = \mathrm{T}_{k,\tau_1}^{above} \Rightarrow \forall j \neq i : \alpha_j \neq \mathrm{T}_{k,\tau_2}^{below} \text{ für } i \in \{1,...,p\}, k \in \{1,...,n\}, \tau_1, \tau_2 \in S_k$$

und

$$\text{Falls } \alpha_i = \mathrm{T}_{k,\tau_1}^{below} \Rightarrow \forall j \neq i : \alpha_j \neq \mathrm{T}_{k,\tau_2}^{above} \text{ für } i \in \{1,...,p\}, k \in \{1,...,n\}, \tau_1, \tau_2 \in S_k$$

Es gilt weiterhin $p \leq \hat{p} \leq n$ für $\hat{p} \in N^+$, da das wiederholte Vorkommen desselben Literals in einem Monom bzw. einer Klausel ausgeschlossen wird:

$$\text{Falls } \alpha_i = \mathrm{T}_{k,\tau_1}^{above} \Rightarrow \forall j \neq i : \alpha_j \neq \mathrm{T}_{k,\tau_2}^{above} \text{ für } i \in \{1,...,p\}, k \in \{1,...,n\}, \tau_1, \tau_2 \in S_k$$

und

$$\text{Falls } \alpha_i = \mathrm{T}_{k,\tau_1}^{below} \Rightarrow \forall j \neq i : \alpha_j \neq \mathrm{T}_{k,\tau_2}^{below} \text{ für } i \in \{1,...,p\}, k \in \{1,...,n\}, \tau_1, \tau_2 \in S_k$$

Es ist $\hat{p} = n$, wenn die Anzahl der Literale lediglich durch die im Datensatz vorkommenden Merkmale beschränkt ist.
Ein Monom bzw. eine Klausel wird im Folgenden auch über die Menge der Literale identifiziert:

$$\mu = \{\alpha_1, \alpha_2, ..., \alpha_p\}$$
$$c = \{\alpha_1, \alpha_2, ..., \alpha_p\}$$

Demgemäß liegt die Konzeptklasse aller zuvor beschriebenen booleschen Monome bzw. Klauseln mit Schwellwertfunktionen vor:

$$C_{\hat{p}}^{Monom} = \{\mu \mid \mu \text{ ist Monom mit höchstens } \hat{p} \text{ Literalen}\}$$
$$C_{\hat{p}}^{Klausel} = \{c \mid c \text{ ist Klausel mit höchstens } \hat{p} \text{ Literalen}\}$$

Da die Mengen S_i für $i = 1,...,n$ maximal m Werte enthalten können (siehe Kapitel 4.1.4), gilt stets $\#S_i \leq m$. Es folgt:

$$\#C_{\hat{p}}^{Monom} = \#C_{\hat{p}}^{Klausel} = \sum_{i=1}^{\hat{p}} 2^i \cdot \binom{n}{i} \cdot m^i = \sum_{i=1}^{\hat{p}} \binom{n}{i} \cdot (2m)^i$$

Schließlich lassen sich die Konzeptklassen erweitern:

$$\mu_k = \alpha_{k1} \wedge \alpha_{k2} \wedge ... \wedge \alpha_{kp_k} \text{ für } k = 1,...,r \text{ mit } r \in N^+ \text{ und } p_k \in N$$

$$c_k = \alpha_{k1} \vee \alpha_{k2} \vee ... \vee \alpha_{kp_k} \text{ für } k = 1,...,r \text{ mit } r \in N^+ \text{ und } p_k \in N$$

sowie

$$\alpha_{ki} \in \Gamma \text{ für } i = 1,..., p_k$$

Für einzelne Monome bzw. Klauseln gelten stets alle Zusatzbedingungen die zuvor vorgestellt worden sind. In der Praxis kann es außerdem sinnvoll sein, die Anzahl der in Monomen bzw. Klauseln vorkommenden Literale global zu beschränken. Es gilt dann analog $p_i \le \hat{p} \le n$ für $i = 1,...,r$. Die Anzahl der Monome bzw. Klauseln sei ebenfalls durch $1 \le r \le \hat{r}$ mit $\hat{r} \in N^+$ beschränkt.
Schließlich ist die disjunktive Normalform bzw. konjunktive Normalform gegeben durch:

$$\lambda = \mu_1 \vee \mu_2 \vee ... \vee \mu_r = (\alpha_{11} \wedge \alpha_{12} \wedge ... \wedge \alpha_{1p_1}) \vee (\alpha_{21} \wedge \alpha_{22} \wedge ... \wedge \alpha_{2p_2}) \vee ... \vee (\alpha_{r1} \wedge \alpha_{r2} \wedge ... \wedge \alpha_{rp_r})$$

$$\vartheta = c_1 \wedge c_2 \wedge ... \wedge c_r = (\alpha_{11} \vee \alpha_{12} \vee ... \vee \alpha_{1p_1}) \wedge (\alpha_{21} \vee \alpha_{22} \vee ... \vee \alpha_{2p_2}) \wedge ... \wedge (\alpha_{r1} \vee \alpha_{r2} \vee ... \vee \alpha_{rp_r})$$

Die disjunktive bzw. konjunktive Normalform wird im Folgenden auch durch die Menge der enthaltenen Literale identifiziert:

$$\lambda = \{\mu_1, \mu_2, ..., \mu_r\} = \{\alpha_{11}, ..., \alpha_{1k_1}, \alpha_{21}, ..., \alpha_{2k_2}, ..., \alpha_{rk_r}\}$$

$$\vartheta = \{c_1, c_2, ..., c_r\} = \{\alpha_{11}, ..., \alpha_{1k_1}, \alpha_{21}, ..., \alpha_{2k_2}, ..., \alpha_{rk_r}\}$$

Schließlich ergeben sich die Konzeptklassen:

$$C_{\hat{p},\hat{r}}^{DNF} = \{\lambda \mid \lambda \text{ ist } DNF \text{ mit höchstens } \hat{r} \text{ Monomen und höchstens } \hat{p} \text{ Literalen pro Monom}\}$$

$$C_{\hat{p},\hat{r}}^{KNF} = \{\vartheta \mid \vartheta \text{ ist } KNF \text{ mit höchstens } \hat{r} \text{ Klauseln und höchstens } \hat{p} \text{ Literalen pro Klausel}\}$$

Mit $z_{\hat{p}} = \#C_{\hat{p}}^{Monom} = \#C_{\hat{p}}^{Klausel} = \sum_{i=1}^{\hat{p}} \binom{n}{i} \cdot (2m)^i$ gilt natürlich auch:

$$\#C_{\hat{p},\hat{r}}^{DNF} = \#C_{\hat{p},\hat{r}}^{KNF} = \sum_{i=1}^{\hat{r}} \binom{z_{\hat{p}}}{i}$$

4.1.3. Auswahl und Zusammenhänge der Konzeptklassen

Im vorherigen Kapitel sind verschiedene Konzeptklassen vorgestellt worden, die sich jedoch in der Handhabung kaum voneinander unterscheiden.
Da die Interpretation von disjunktiven Normalformen (DNF) intuitiver ist, soll die Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ das Lernziel sein.

Ein Monom kann als Spezialfall einer konjunktiven Normalform (KNF), bei der alle Klauseln eine Länge von eins haben bzw. als Spezialfall einer DNF, die nur aus einem Monom besteht, betrachtet werden.
Diese Zusammenhänge können Beweise stark vereinfachen. In den jeweiligen Kapiteln wird auf die Vorgehensweise genauer eingegangen.

4.1.4. Einschränkung der Abtastbereiche für Schwellwerte

Da es im Folgenden um die Minimierung des empirischen Risikos geht, ist es möglich, die zur Auswahl stehenden Schwellwerte auf die vorkommenden Werte der gegebenen Lernstichprobe zu beschränken. Demnach ist der Abtastbereich für das Merkmal *j* gegeben durch:

$$S_j \leftarrow \{x_{1j}, x_{2j}, ..., x_{mj}\} \text{ für } j = 1, ..., n$$

mit

$$u[1,m] = ((\vec{x}_1, y_1) = ((x_{11}, ..., x_{1n}), y_1), ..., (\vec{x}_m, y_m) = ((x_{m1}, ..., x_{mn}), y_m))$$

4.1.5. Eigenschaften von Schwellwertfunktionen

Bei Betrachtung von Schwellwerten gilt stets:

$$\text{Falls } a \geq \tau \text{ mit } a, \tau \in R \text{ so ist } b \geq \tau \quad \forall b \geq a$$

Ist also beispielsweise 5 größer als 2, so ist auch 7 größer als 2.

Da über Schwellwerte ein reeller Wert in einen booleschen Wert transformiert wird, ergibt sich analog die folgende Eigenschaft für die Klassifikation durch Schwellwertfunktionen:

$$\left\{\vec{x}_i \mid T^{above}_{j,\tau_1}(\vec{x}_i) = -1 \wedge i = 1,...,m\right\} \subseteq \left\{\vec{x}_i \mid T^{above}_{j,\tau_2}(\vec{x}_i) = -1 \wedge i = 1,...,m\right\} \quad \forall \tau_1 \leq \tau_2 \text{ und } \tau_1, \tau_2 \in R$$

bzw.

$$\left\{\vec{x}_i \mid T^{below}_{j,\tau_1}(\vec{x}_i) = +1 \wedge i = 1,...,m\right\} \subseteq \left\{\vec{x}_i \mid T^{below}_{j,\tau_2}(\vec{x}_i) = +1 \wedge i = 1,...,m\right\} \quad \forall \tau_1 \leq \tau_2 \text{ und } \tau_1, \tau_2 \in R$$

Wird also ein Schwellwert $\tau_1 \in R$ auf den Wert $\tau_2 \in R$ erhöht, so ändert sich die Klassifikation der einzelnen Beispiele, deren Merkmalswert ursprünglich kleiner als der Schwellwert τ_1 war, nicht, da sie auch kleiner als τ_2 sind. Diese Tatsache lässt sich bei der Entwicklung von Lernalgorithmen ausnutzen.

Es gilt weiterhin:

$$\#\left\{\vec{x}_i \mid T^{d}_{j,\tau}(\vec{x}_i) = -1 \wedge i = 1,...,m\right\} = m - \#\left\{\vec{x}_i \mid T^{d}_{j,\tau}(\vec{x}_i) = +1 \wedge i = 1,...,m\right\} \quad \tau \in R,\ d \in \{above, below\}$$

Ist also die Anzahl der durch eine Schwellwertfunktion negativ klassifizierten Beispiele bekannt, so werden die verbleibenden Beispiele unweigerlich als positiv klassifiziert und umgekehrt.

Zur Vereinfachung der Beweise und Analysen kann außerdem angenommen werden, dass jeder mögliche Schwellwert als eine einzelne, boolesche Variable kodiert ist. Dadurch kann die Betrachtung von Schwellwertfunktionen vernachlässigt werden, denn es liegen lediglich boolesche Vektoren vor.

Entsprechend werden aus *n* Merkmalen $\sum_{i=1}^{n} \#S_i$ Merkmale nach der folgenden Vorschrift:

$$x_{i,j,k} = \begin{cases} +1 & falls\ x_{i,j} \geq \tau_k \\ -1 & sonst \end{cases}$$

$$\text{mit } i = 1,...,m,\ j = 1,...,n \text{ und alle } \tau_k \in S_i \text{ für } k = 1,...,\#S_i$$

Bei derartiger Kodierung sind die booleschen Variablen für festes *i* und *j* stark voneinander abhängig. Dies ist theoretisch betrachtet jedoch lediglich eine Verteilungseigenschaft. In der Praxis ist dieser Ansatz jedoch nicht geeignet, da eine große Anzahl boolescher Variablen zu hohem Speicherverbrauch und auch schlechter Laufzeit führen kann.

4.2. Beweis: NP-Schwierigkeit im Falle des PAC-Lernens mit Rauschen

Das Design der Konzeptklasse lässt bereits vermuten, dass die Lösung des Problems der empirischen Risiko-Minimierung der Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ nicht trivial ist.

Im Folgenden soll gezeigt werden, dass im Falle des PAC-Lernens mit Rauschen auf der Klassifikation die empirische Risiko-Minimierung der Konzeptklasse C_n^{Monom} ein NP-schweres Optimierungsproblem ist. Daraus folgt unweigerlich auch, dass die empirische Risiko-Minimierung der Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ ebenfalls NP-schwer ist, denn es gilt $C_n^{Monom} = C_{n,1}^{DNF}$. In diesem Fall liegt eine disjunktive Normalform mit lediglich einem Monom vor.

Der Beweis wird durch Reduktion des Problems Set-Cover geführt und ist in einer E-Mail-Korrespondenz mit Berthold Vöcking entwickelt worden. Weiterhin wird zur Vereinfachung und ohne Beschränkung der Allgemeinheit angenommen, dass alle zur Auswahl stehenden Schwellwerte als einzelne, boolesche Variablen vorliegen (siehe Kapitel 4.1.5).

4.2.1. Das Problem Set-Cover

Es liegt eine Menge mit *w* Elementen vor:

$$M = \{e_1,...,e_w\}$$

Weiterhin liegen *n* Teilmengen (diese Variable entspricht im folgenden Reduktionsbeweis der Anzahl der Merkmale einer Lernstichprobe und ist daher identisch benannt) von *M* vor:

$$C_i \subset M \text{ für } i = 1,...,n$$

Außerdem gilt, dass die Vereinigung dieser Teilmengen die Menge *M* ergibt:

$$M = \bigcup_{i=1}^{n} C_i$$

Gesucht ist nun die minimale Indexmenge $I \subseteq \{1,2,...,n\}$ so dass:

$$\bigcup_{i \in I} C_i = M$$

Dieses Optimierungsproblem ist NP-äquivalent und wird im folgenden Kapitel zur Reduktion verwendet.

4.2.2. Reduktionsbeweis

Behauptung: Das Problem *L* der Minimierung des empirischen Risikos der Konzeptklasse C_n^{Monom} ist NP-schwer.

Beweis:

Die Behauptung wird durch Reduktion von Set-Cover auf *L* bewiesen.

Es muss ein Polynomialzeit-Algorithmus angegeben werden, der das Set-Cover Problem in eine Eingabe transformiert, die von *L* akzeptiert wird, so dass mit dieser Lösung wiederum Set-Cover gelöst werden kann. Es sind dazu eine Lernstichprobe *U* mit entsprechenden Beispielen zu generieren.

Jede Komponente eines Vektors soll einer zur Überdeckung verfügbaren Menge entsprechen, deshalb wird für jedes e_i mit $i = 1,...,m$ zunächst ein Vektor mit *n* Komponenten auf die folgende Weise erzeugt, der außerdem ver-*n*-facht wird:

$$\vec{x}_{j,k} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix} \text{ mit } t_i = \begin{cases} -1 & falls \quad e_j \in C_i \\ +1 & sonst \end{cases} \qquad \text{für } i = 1,...,n,\ j = 1,...,w \text{ und } k = 1,...,n$$

$$y_{j,k} = -1$$

Ein solcher Vektor weist also für das zugehörige Element $e \in M$ an der Stelle *i* eine -1 auf, falls *e* in C_i vorkommt. Diese Vektoren erhalten in der Lernstichprobe das Label -1, da sie für die Merkmalsauswahl entscheidend sind. Zudem muss, um die minimale Anzahl von Mengen, die zur Überdeckung von *M* notwendig sind, zu erhalten (Diese Bedingung macht letztlich die NP-Schwierigkeit von Set-Cover aus), für jedes Merkmal ein unerfüllbares Beispiel eingefügt werden. Dadurch wird der Lernalgorithmus gezwungen, insgesamt möglichst wenige Merkmale zu wählen, da die Auswahl mehrerer Merkmale einen größeren Fehler liefert. Dementsprechend müssen die folgenden Vektoren mit dem Label +1 hinzugefügt werden:

$$\vec{x}_{w+j} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix} \text{ mit } t_i = \begin{cases} -1 & falls \quad j = i \\ +1 & sonst \end{cases} \qquad \text{für } i = 1,...,n \text{ und } j = 1,...,n$$

$$y_{w+j} = +1$$

Der Algorithmus zur empirischen Risiko-Minimierung muss gezwungen werden, alle positiven Beispiele korrekt zu klassifizieren, damit, bezogen auf das Set-Cover Problem, letztlich jedes Element überdeckt wird. Dies wird durch Ver-*n*-fachung eines jeden negativen Beispiels erreicht. Auf diese Weise liefert die Fehlklassifikation eines festen Negativbeispiels und damit auch der *n* Duplikate, einen Fehler, der genauso hoch ist wie die Fehlklassifikation aller Positivbeispiele. Entsprechend kann die Fehlklassifikation eines festen Negativbeispiels nicht zu einem minimalen, empirischen Risiko führen. Abschließend wird die Lernstichprobe mit der Länge $m = w \cdot n + n$ generiert.

Es folgt dass *L* NP-schwer ist.

q. e. d.

Beispiel:

$M = \{e_1, e_2, e_3\}$

$C_1 = \{e_1, e_3\}$

$C_2 = \{e_1\}$

$C_3 = \{e_2\}$

$C_4 = \{e_3\}$

Dieses Beispiel wird zu folgenden Vektoren verarbeitet:

$$\vec{x}_{1,k} = \begin{pmatrix} -1 \\ -1 \\ +1 \\ +1 \end{pmatrix} \qquad \vec{x}_{2,k} = \begin{pmatrix} +1 \\ +1 \\ -1 \\ +1 \end{pmatrix} \qquad \vec{x}_{3,k} = \begin{pmatrix} -1 \\ +1 \\ +1 \\ -1 \end{pmatrix} \quad \text{für } k = 1,...,n$$

sowie den zugehörigen Labels:

$$y_{1,k} = -1 \qquad y_{2,k} = -1 \qquad y_{3,k} = -1 \qquad \text{für } k = 1,...,n$$

Schließlich werden die folgenden Vektoren zur Bestrafung der Auswahl eines jeden Merkmals gebildet:

$$\vec{x}_4 = \begin{pmatrix} -1 \\ +1 \\ +1 \\ +1 \end{pmatrix} \qquad \vec{x}_5 = \begin{pmatrix} +1 \\ -1 \\ +1 \\ +1 \end{pmatrix} \qquad \vec{x}_6 = \begin{pmatrix} +1 \\ +1 \\ -1 \\ +1 \end{pmatrix} \qquad \vec{x}_7 = \begin{pmatrix} +1 \\ +1 \\ +1 \\ -1 \end{pmatrix}$$

sowie die zugehörigen Labels:

$$y_4 = +1 \qquad y_5 = +1 \qquad y_6 = +1 \qquad y_7 = +1$$

4.3. Lernalgorithmus durch Brute-Force

Eine mögliche Lösung das empirische Risiko der Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ zu minimieren, ist die Anwendung der Brute-Force-Methode, die jedoch bereits ab zwei Merkmalen sehr rechenintensiv ist. Dabei wird jedes Element der Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ auf der Stichprobe evaluiert. Der folgende Algorithmus leistet das Gewünschte:

Algorithmus 4 Thresholder Brute Force (TBF)

Eingabe: $u[1,m] := ((\vec{x}_1, y_1) = ((x_{11},...,x_{1n}), y_1),...,(\vec{x}_m, y_m) = ((x_{m1},...,x_{mn}), y_m))$

Ausgabe: Ein Monom aus Schwellwertfunktionen (Literalen) als Hypothese

$$\lambda^{\max} = \arg\max_{\lambda \in C_{\hat{p},\hat{r}}^{DNF}} \{\vec{x}_j \mid \lambda(\vec{x}_j) = y_j \wedge j = 1,...,m\}$$

Return $\lambda^{\max}$

Der Algorithmus hat eine Laufzeit von $O(\hat{r} \cdot m^{n\hat{r}})$, da die Klasse $C_{\hat{p},\hat{r}}^{DNF}$ exponentielle Größe besitzt. Trotz der exponentiellen Laufzeit, kann es unter gewissen Umständen sinnvoll sein, diesen Algorithmus zu verwenden. Im Falle der Insolvenzvorhersage ist es häufig erforderlich, die Anzahl der Merkmale stark einzuschränken. Insbesondere ist bei spärlicher Datenlage eine Merkmalseinschränkung sogar zwingend erforderlich, um eine Generalisierung zu gewährleisten und ein Overfitting zu verhindern.

4.4. Lernalgorithmus durch dynamische Programmierung

Ein weiterer, interessanter Ansatz ist es, das empirische Risiko für die Klasse $C_{\hat{p}}^{Monom}$ mit Hilfe des Prinzips der dynamischen Programmierung zu minimieren. Dieser Algorithmus besitzt jedoch exponentielle Laufzeit und steht nicht im Widerspruch zur NP-Schwierigkeit, auch wenn es auf den ersten Blick anders erscheint. Die exponentiellen Faktoren können eliminiert werden, wodurch der Algorithmus zu einer Heuristik wird.

Da die Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ hauptsächlich als Lernziel fungiert soll, kann der folgende Algorithmus nur im Fall $\hat{r} = 1$ angewendet werden.

Es besteht weiterhin eine entfernte Verwandtschaft zu einem Algorithmus zur Lösung des Rucksackproblems. Zur Veranschaulichung des Algorithmus und um die Verbindung zum Rucksackproblem herzustellen, wird letzteres kurz vorgestellt.

4.4.1. Lösung des Rucksackproblems mit dynamischer Programmierung

Es liegen *n* Gegenstände vor, die einen Nutzen c_i und ein Gewicht w_i für $i = 1,...,n$ haben. Gesucht sind nun $(\beta_1,...,\beta_n) \in \{0,1\}^n$ so dass

$$\sum_{i=1}^{n} \beta_i c_i$$

maximal wird, und dabei die folgende Bedingung erfüllt ist:

$$\sum_{i=1}^{n} \beta_i w_i \leq W \text{ mit } W \in N^+$$

Anschaulich wird also eine Auswahl von Gegenständen gesucht, die einen maximalen Nutzen liefert, jedoch das Maximalgewicht *W* des Rucksacks, in dem die beteiligten Gegenstände transportiert werden sollen, nicht überschreitet. Mit Hilfe der dynamischen Programmierung kann nun eine optimale Lösung nach der folgenden Vorschrift berechnet werden:

$$Opt[k,V] = \max\{Opt[k-1,V], Opt[k-1,V-w_k] + c_k\}$$

Die Zeilen der Matrix entsprechen dabei den verfügbaren Gegenständen, die Spalten dem jeweiligen Maximalgewicht (Gewichtsschranke). Im ersten Fall der obigen Gleichung ist die Konfiguration ohne den aktuell in Rede stehenden Gegenstand *k* bei aktueller Gewichtsschranke *V* optimal und wird verwendet. Im zweiten Fall bringt der aktuell in Rede stehende Gegenstand *k* einen größeren Nutzen. Dementsprechend muss eine Konfiguration gewählt werden, die hinreichend Platz für diesen Gegenstand liefert. Der Nutzen des Gegenstandes *k* wird dann zu dem Nutzen dieser Konfiguration addiert.

Eine typische Matrix sieht beispielsweise wie folgt aus:

K	c_k	w_k	$Opt[k,V]$ V =	1	2	3	4	5	6	7	8	9
1	38	3		0	**0**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
2	63	5		0	0	38	38	**63**	**63**	**63**	**101**	**101**
3	64	6		0	0	38	38	63	**64**	**64**	101	**102**
4	21	2		0	**21**	38	38	63	64	**84**	101	102
5	19	2		0	21	38	**40**	63	64	84	101	**103**

Tab 1: Dynamische Programmierung bei Rucksackproblem

Um die optimale Lösung anhand der Matrix zu rekonstruieren, ist ein Backtracking-Algorithmus erforderlich. Dieser beginnt in der rechten unteren Zelle $Opt[5,9]$ und arbeitet sich wie folgt vor: Ist ein Wert fett gedruckt, so ist dessen Entstehung auf den zweiten Teil der obigen Optimalitätsgleichung zurückzuführen, und der Gegenstand *k* ist Teil der Lösung. Die korrekte Anknüpfzelle kann an der Position $Opt[k-1,V-w_k]$ gefunden werden. Ist der Wert nicht fett gedruckt, so ist der Gegenstand *k* nicht Teil der Lösung und die Anknüpfzelle ist $Opt[k-1,V]$. Die gesuchte Lösung in diesem Fall wäre folglich $(0,1,0,1,1)$.

Dieser Algorithmus besitzt jedoch im allgemeinen Fall auch exponentielle Laufzeit, da das Maximalgewicht *W* nicht beschränkt ist und so bei *i* vorliegenden Bits den Wert 2^i annehmen kann. Der Korrektheitsbeweis basiert auf der Idee, dass durch das Entfernen des *k*-ten Gegenstandes die Matrix eine optimale Lösung für die *k – 1* verbleibenden Gegenstände darstellt.

4.4.2. Lernen von Monomen mit dynamischer Programmierung

Der zu entwickelnde Algorithmus wird zeilenorientiert arbeiten. Letztlich soll jede Zeile einem zu wählenden Merkmal (Gegenstand) mit einer passenden Schwellwertfunktion entsprechen, die Spalten wiederum verschiedenen Gewichtsschranken. Der Nutzen soll analog zum Rucksackproblem der Anzahl, der durch eine zu einem Merkmal ausgewählten Schwellwertfunktion korrekt klassifizierten Negativbeispielen entsprechen. Das Gewicht hingegen sei die Anzahl der durch diese Schwellwertfunktion falsch klassifizierten Positivbeispiele.

Das Maximalgewicht im Falle des Rucksackproblems wäre gegeben durch:

$$W = \sum_{i=1}^{n} w_i \leq n \cdot \max\{w_i \mid i = 1,...,n\} = n \cdot m$$

Im Falle des Algorithmus zum Lernen von Monomen ändern sich jedoch Gewicht und Nutzen aller Gegenstände in Abhängigkeit von zuvor gewählten Merkmalen (Gegenständen). Der Grund dafür ist, dass die Wahl eines Merkmals mit einer passenden Schwellwertfunktion in einem Monom eine bestimmte Anzahl von Positivbeispielen irreversibel falsch klassifiziert, und die Wahl eines weiteren Merkmals an dieser Tatsache nichts ändern kann. Daher dürfen diese Beispiele bei der Wahl eines weiteren Merkmals nicht zur Auswahl stehen, was sich im minimierten Gewicht aller verbleibenden Gegenstände widerspiegelt. Dasselbe gilt für den Nutzen eines Merkmals, denn auch dieser verringert sich in aller Regel, da die durch die bisherig gewählten Merkmale korrekt klassifizierten Negativbeispiele auf lediglich dieses angewiesen sind, denn auch diese gewonnene Klassifikation ist irreversibel. Folglich ist das Gesamtgewicht durch *m* beschränkt und es gilt stets $W \leq m$, denn es können maximal alle Positivbeispiele falsch klassifiziert werden. Der Extremfall $W = m$ tritt nur dann ein, wenn keine Negativbeispiele vorliegen. Exakt gilt also Folgendes:

$$W = \#\{\vec{x}_i \mid y_i = +1 \wedge i = 1,...,m\}$$

Die Spaltenanzahl der Optimalmatrix ist dann durch W gegeben. Das Maximalgewicht W kann auch zu Beginn des Algorithmus beschränkt werden um eine höhere Effizienz zu erzielen. Inwiefern dies sinnvoll sein kann, wird im Kapitel 4.4.3. diskutiert. Zu Beginn des Algorithmus wird für jedes Merkmal ein Schwellwertfunktionstyp festgelegt:

$$(d_1,...,d_n) \in \{above, below\}^n$$

Es ist offensichtlich, dass:

$$\#\{above, below\}^n = 2^n$$

Jede Zelle muss als Datenstruktur definiert werden und besitzt folgende Attribute, die jeweils für $i = 1,...,n$ und $k = 0,...,W$ definiert sind.

Cell

- $C_{i,k}$ Maximaler Nutzen im Bezug auf aktuelle Konfiguration.
- $\mu_{i,k}$ Monom, das maximalen Nutzen $C_{i,k}$ liefert.
- $U_{i,k} = \{(\vec{x}_j, y_j) \mid \mu_{i,k}(\vec{x}_j) = +1 \wedge j = 1,...,m\}$ Eine Lernstichprobe die alle noch nicht irreversibel klassifizierten Beispiele enthält.

Zu Beginn des Algorithmus ist kein Merkmal mit einer Schwellwertfunktion gewählt, also gilt

$$\mu_{0,k} = \{\ \} \text{ für } k = 0,...,W$$

und demnach auch:

$$U_{0,k} = \{(\vec{x}_j, y_j) \mid j = 1,...,m\} \text{ für } k = 0,...,W$$

Es stehen also noch alle Beispiele der Lernstichprobe zur Verfügung
Der Nutzen der Schwelle *j* zum Merkmal *i* bezogen auf das Maximalgewicht *k* ist dann gegeben durch:

$$c_{i,j,k}^{above} = \#\{\vec{x} \mid (\vec{x}, -1) \in U_{i-1,k} \wedge \mathrm{T}_{i,x_{j,i}}^{above}(\vec{x}) = -1\}$$

für jedes $i = 1,...,n$ und jedes $j = 1,...,m$ und jedes $k = 0,...,W$

$$c_{i,j,k}^{below} = \#\{\vec{x} \mid (\vec{x}, -1) \in U_{i-1,k} \wedge \mathrm{T}_{i,x_{j,i}}^{below}(\vec{x}) = -1\}$$

für jedes $i = 1,...,n$ und jedes $j = 1,...,m$ und jedes $k = 0,...,W$

Die zugehörigen Gewichte sind gegeben durch:

$$w_{i,j,k}^{above} = \#\left\{\vec{x} \mid (\vec{x},+1) \in U_{i-1,k} \wedge \mathrm{T}_{i,x_{j,i}}^{above}(\vec{x}) = -1\right\}$$

für jedes $i = 1,...,n$ und jedes $j = 1,...,m$ und jedes $k = 0,...,W$

$$w_{i,j,k}^{below} = \#\left\{\vec{x} \mid (\vec{x},+1) \in U_{i-1,k} \wedge \mathrm{T}_{i,x_{j,i}}^{below}(\vec{x}) = -1\right\}$$

für jedes $i = 1,...,n$ und jedes $j = 1,...,m$ und jedes $k = 0,...,W$

Für festes *i* entsprechen die Variablen also den Nutzen bzw. Gewichten für unterschiedliche Schwellwerte $x_{j,i}$ des Merkmals *i* bezogen auf die vorherige Konfiguration, die das Maximalgewicht *k* nicht überschreitet. Da ein Merkmal maximal einmal in einem Monom vorkommen soll, muss für festes *i* genau ein Schwellwert sowie ein Schwellwertfunktionstyp gewählt werden, der den besten Nutzen liefert.
Zu diesem Zweck werden die Zellen der Temporärmatrix für ein festes $i \in \{1,...,n\}$ und $V = 0,...,W$ wie folgt berechnet:

$$Tmp[i,j,V] = \max\left\{Opt[i-1,V], Opt[i-1,V-k] + c_{i,j,k}^{d_i} \mid w_{i,j,k}^{d_i} \le V-k \wedge k = V,...,0\right\}$$

Falls die Bildung des Maximum nicht durch die Wahl des ersten Wertes $Opt[i-1,V]$ zustande gekommen ist, so wird der verantwortliche Index *k* in der Variablen $k_{i,j,V}^{\max}$ abgelegt.
Sobald die Temporärmatrix erstellt ist, wird sie sofort zu einer einzigen Zeile verdichtet:

$$j_{i,V}^{\max} = \underset{\forall j \in \{1,...,m\}}{\arg\max}\left\{Tmp[i,j,V] \mid j = 1,...,m\right\}$$

Die optimale Lösung wird anschließend in die Optimalmatrix eingetragen:

$$Opt[i,V] = Opt[i, j_{i,V}^{\max}, V] \text{ für festes } i \in \{1,...,n\} \text{ und } V = 0,...,W$$

Es müssen also für jeden der *2m* Schwellwerte sämtliche links liegenden Spalten, inklusive der *V*-ten Spalte der *i – 1*-ten Zeile der Optimalmatrix betrachtet werden. Der Vorteil liegt jedoch

darin, dass so maximal $\frac{V \cdot (V-1)}{2} < W$ Werte für jeden Wert der Temporärmatrix betrachtet werden müssen. Gäbe es diesen Verdichtungsschritt nicht, so wären erneut *2m* Werte der $i-1-$ten Temporärmatrix und damit insgesamt $4m^2$ Berechnungsschritte erforderlich.
Der Verdichtungsschritt verletzt nicht die Optimalitätsbedingung, was auf die Natur der Schwellwerte zurückzuführen ist. Bei Betrachtung von Nutzen und Gewicht und Prüfung der *m* in Rede stehenden Schwellwerte mit demselben Schwellwertfunktionstyp einer sortierten Folge, gibt es nur eine Möglichkeit für die Zusammensetzung der Positivbeispiele bei gleichem Gewicht (siehe Kapitel 4.1.5).

Beispiel zur Veranschaulichung:

V	1	2	3	4	5	6	7	8	9	10
τ_1	-5	-2	5	13	15	19	19	552	757	801
x_1	-5	-2	5	13	15	19	19	552	757	801
y	+1	+1	-1	+1	-1	-1	-1	-1	+1	+1
c	0	0	0	1	1	2	3	4	5	5
W	0	1	2	2	3	3	3	3	3	4
	$T_{1,-2}^{above}$			$T_{1,15}^{above}$				$T_{1,757}^{above}$		

Abbildung 8: Konstanz der Positivbeispiele bei Schwellenauswahl

Die Schwellwertfunktion $T_{1,15}^{above}$ liefert im obigen Beispiel eine Missklassifikation der drei links stehenden Positivbeispiele (grau markiert) für das Merkmal x_1. Bei Betrachtung der Schwellwertfunktion $T_{1,-2}^{above}$ würde die Anzahl der Missklassifikationen verringert. Zeitgleich würde auch der Nutzen verringert, der aber bei gegebener Gewichtsschranke maximiert werden soll. Ein geringerer Nutzen, der ebenso eine geringere Gewichtsschranke hätte, wäre also bereits weiter links in der Matrix vorgefunden worden.
Bei Betrachtung der Schwellwertfunktion $T_{1,757}^{above}$ ist ersichtlich, dass die Zusammensetzung der falsch klassifizierten Positivbeispiele im Vergleich zur Funktion $T_{1,15}^{above}$ gleich geblieben ist. Letztere liefert jedoch einen höheren Nutzen und ist damit zu bevorzugen. Anders ausgedrückt, gibt es für die Auswahl einer Schwelle zu einem festen Merkmal keine unterschiedlichen Mengen von falsch klassifizierten Positivbeispielen (siehe Kapitel 4.1.5).

In diesem Fall gilt daher:

$$\left\{\vec{x}_j \mid T^d_{i,\tau_1}(\vec{x}_j) = -1 \wedge y_j = +1 \wedge j = 1,...,m\right\} \subseteq \left\{\vec{x}_j \mid T^d_{i,\tau_2}(\vec{x}_j) = -1 \wedge y_j = +1 \wedge j = 1,...,m\right\}$$

für

$$\forall \tau_2 \geq \tau_1 \text{ mit } \tau_1 \in R$$

Diese Eigenschaft rechtfertigt die Maximumsbildung bezüglich des Nutzens bei gleichem Gewicht, denn lediglich die Zusammensetzung der falsch klassifizierten Positivbeispiele steht der Optimalität der Gesamtlösung im Wege. Es wird also lokal maximiert, ohne weitere Fehlklassifikationen in Kauf zu nehmen. Der Aufbau der Temporärmatrix gibt bereits vor, dass lediglich Nutzen (korrekt klassifizierte Negativbeispiele) zur Auswahl stehen, die die aktuelle Gewichtsschranke nicht verletzen, so dass die Maximumsbildung unkritisch ist.

Effektiv liegt also durch den Verdichtungsschritt eine $n \times W + 1$ Optimalmatrix vor, die bis zum Ende des Algorithmus im Speicher zu halten ist. Im letzten Schritt werden die fehlenden Variablen der Zellen entsprechend aktualisiert:

$$\mu_{i,V} = \mu_{i-1,\left(k^{\max}_{i,V}\right)} \cup \left\{T^{d_i}_{i,\tau^{\max}_{i,V}}\right\} \text{ mit } k^{\max}_{i,V} = k^{\max}_{i,\left(j^{\max}_{i,V}\right),V} \text{ und } \tau^{\max}_{i,V} = x_{\left(j^{\max}_{i,V}\right),i}$$

Und wie oben:

$$U_{i,k} = \left\{(\vec{x}_j, y_j) \mid \mu_{i,k}(\vec{x}_j) = +1 \wedge j = 1,...,m\right\}$$

Aufgrund dieser Aktualisierungen der Datenstrukturen ergibt sich ein weiterer Laufzeit- und Speichervorteil, denn diese Informationen müssen lediglich für jede Zelle der Optimalmatrix gehalten und aktualisiert werden.
Die Teilstichproben $U_{i,V}$ für $V = 0,...,W$ werden dann nach dem Merkmal *i + 1* sortiert. Dies beschleunigt den Aufbau der Temporärmatrix erheblich, da so für jede vorkommende Schwellwertfunktion Nutzen und Gewicht durch eine Iteration mit Hilfe der Beispiele aus $U_{i,V}$ bestimmt werden können, denn die abzutastenden Schwellwerte entsprechen per Definition den vorkommenden Werten in der Lernstichprobe (siehe Kapitel 4.1). Folglich muss für den Aufbau

einer Temporärmatrix jede Teilstichprobe für alle Schwellen insgesamt nur einmal durchlaufen werden.

Der Algorithmus in der obigen Form liefert jedoch keine optimale Lösung. Bereits die Tatsache, dass die Gewichtsschranke W durch m beschränkt ist, gibt zu dieser Annahme Anlass, da in diesem Fall ein Polynomialzeit-Algorithmus für ein NP-schweres Problem gefunden wäre.
Das Problem liegt in der Reihenfolge der Zeilen der Optimalmatrix, denn diese ist keinesfalls fest wählbar, da durch die Wahl eines Merkmals später zu wählende Merkmale negativ beeinflusst werden, deren Beteiligung an der optimalen Lösung jedoch nicht auszuschließen ist. Im Gegensatz zum Rucksackproblem sind Nutzen und Gewichte - wie zuvor erwähnt - nicht konstant, so dass die Optimalitätsbedingung nicht greift.
Eine optimale Lösung ist demnach nur dann gewährleistet, wenn jede mögliche Reihenfolge der vorliegenden Merkmale zur Bestimmung des finalen Optimums herangezogen wird. Damit besäße der vorgestellte Algorithmus jedoch exponentielle Laufzeit, da $n!$ Optimalmatrizen bestimmt werden müssen. Betrachtet man lediglich eine feste Optimalmatrix, so ist der Algorithmus eine Heuristik.
Darüber hinaus wäre noch eine Auswahl der Schwellwertfunktionstypen für jedes Merkmal zu treffen, für die es weitere 2^n Kombinationsmöglichkeiten gibt. Diese gehen ebenfalls als Faktor in die Anzahl der zu berechnenden Matrizen ein.

4.4.3. Lösung und Fehlerskalierung

Ein großer Vorteil des Algorithmus liegt in der Skalierbarkeit der Fehler 1. Art und 2. Art. Die Spaltenposition legt dabei den gewünschten Fehler 2. Art fest, denn die Gewichtsschranke gibt die maximal zulässige Fehlklassifikation der positiven Beispiele an. Über die Beziehung

$$\alpha = \frac{k}{W}$$

kann die gesuchte Spalte k durch festlegen des Fehlers 1. Art wie folgt gefunden werden:

$$k = \lfloor \alpha \cdot W \rfloor$$

Um die optimale Lösung zu erhalten, die den gewünschten Fehler 1. Art liefert, müssen alle $n!\cdot 2^n$ Optimalmatrizen durchlaufen werden, um den maximalen Nutzen zu finden:

$$c = \underset{i=1,\ldots,n!\cdot 2^n}{\arg\max} \left\{ Opt_i[n,k] \mid i = 1,\ldots,n!\cdot 2^n \right\}$$

Dabei bezeichnet Opt_i die *i*-te Optimalmatrix. Der Fehler 2. Art wäre dann gegeben durch

$$\beta = \frac{C_{n,k}}{m - W}$$

der *c*-ten Optimalmatrix. Das gesuchte Monom wäre in $\mu_{n,k}$ abgelegt.

Da insbesondere bei der Insolvenzerkennung die Skalierung der Fehler 1. und 2. Art wünschenswert ist, denn ein Kreditgeber möchte ein bestimmtes Kreditrisiko eingehen, ist diese Eigenschaft von großem Vorteil, denn ein einziges Training genügt, um eine beliebige Skalierung der Fehlerwerte ohne weiteren Rechenaufwand vornehmen zu können. Die optimale Lösung ist dabei ebenfalls garantiert und rechtfertigt den hohen Rechenaufwand.
Weiterhin kann im Vorfeld der Parameter *W* beschränkt werden, wenn sehr hohe Fehler 1. Art von beispielsweise 60% nicht in Frage kommen sollen; dies beschleunigt die Laufzeit des Algorithmus.

4.4.4. Laufzeitanalyse

In diesem Kapitel soll die Laufzeit des vorgestellten Algorithmus bestimmt werden. Bei der Ausführung entstehen die folgenden Kosten:

- $O(W \cdot n)$ Anzahl der Zellen der Optimalmatrix die berechnet werden müssen.
- $O(m \cdot \left(\frac{W \cdot (W-1)}{2} \right))$ Berechnung aller Zellen einer festen Temporärmatrix.
- $O(W \cdot m \cdot \log(m))$ Sortierung der Teilstichproben einer Zeile nach dem Verdichtungsschritt.
- $O(n!)$ Anzahl der zu berechnenden Optimalmatrizen, da *n!* Ordnungen der *n* Merkmale existieren.
- $O(2^n)$ Anzahl der Schwellwertfunktionstypen bei *n* Merkmalen.

Dies ergibt mit $W \leq m$ eine Gesamtlaufzeit von:

$$O\left(n! \cdot 2^n \cdot n \cdot m^3\right)$$

4.5. Heuristischer Lernalgorithmus

In diesem Kapitel soll eine effiziente Heuristik entwickelt werden, die das NP-schwere Problem der empirischen Risiko-Minimierung für die Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ approximativ löst. Dazu wird zunächst eine Heuristik zur empirischen Risiko-Minimierung der Konzeptklasse $C_{\hat{p}}^{Monom}$ entwickelt.
Die Grundidee ist einfach: Jeder Iterationsschritt führt zu der Auswahl eines Merkmals sowie einer zugehörigen Schwelle. Dabei werden alle zuvor gewählten Schwellen in angemessener Form berücksichtigt. Zu diesem Zweck wird erneut ausgenutzt, dass in einem Monom ein einziges Literal mit dem Wert -1 dessen Gesamtwert bestimmt:

$$\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_p = -1 \text{ falls } \exists \alpha_i \text{ mit } \alpha_i = -1 \text{ für } i \in \{1,2,..., p\}$$

Es wird daher in jedem Iterationsschritt diejenige Schwelle gewählt, die eine möglichst große Anzahl negativer Beispiele richtig klassifiziert, bei möglichst geringer Fehlklassifikation

positiver Beispiele. Dies wird rechnerisch durch einen Quotienten dieser Häufigkeiten realisiert. Der folgende Algorithmus leistet das gewünschte:

Algorithmus 6 Thresholder Heuristic Single Monom (THSM)

Eingabe: $U[1,m] = ((\vec{x}_1, y_1) = ((x_{11},...,x_{1n}), y_1),...,(\vec{x}_m, y_m) = ((x_{m1},...,x_{mn}), y_m))$, $\omega > 0$,

$\hat{p} \in N^+$

Ausgabe: Ein Monom mit Schwellwertfunktionen als Hypothese

Initialisierung

$S_j \leftarrow \{x_{1j}, x_{2j},..., x_{mj}\}$ für $j = 1,...,n$

$H_0 \leftarrow \{\ \}$

$U_0 \leftarrow \{(\vec{x}_i, y_i) \mid i = 1,...,m\}$

$k \leftarrow 0$

$adv \leftarrow 0$

$maxAdv \leftarrow 0$

Wiederhole:

$maxAdv \leftarrow 0$

Für j = 1, 2, …, n führe aus:

Für jedes $\tau \in S_j$ führe aus:

$$norm^{pos} \leftarrow \#\{\vec{x} \mid (\vec{x},+1) \in U_k\}$$

$$norm^{neg} \leftarrow \#\{\vec{x} \mid (\vec{x},-1) \in U_k\}$$

$$gain^{above} \leftarrow \frac{\#\{\vec{x} \mid (\vec{x},-1) \in U_k \wedge \mathrm{T}_{j,\tau}^{above}(\vec{x}) = -1\}}{norm^{neg}}$$

$$gain^{below} \leftarrow \frac{\#\{\vec{x} \mid (\vec{x},-1) \in U_k \wedge \mathrm{T}_{j,\tau}^{below}(\vec{x}) = -1\}}{norm^{neg}}$$

$$loss^{above} \leftarrow \frac{\#\{\vec{x} \mid (\vec{x},+1) \in U_k \wedge \mathrm{T}_{j,\tau}^{above}(\vec{x}) = -1\}}{norm^{pos}}$$

$$loss^{below} \leftarrow \frac{\#\{\vec{x} \mid (\vec{x},+1) \in U_k \wedge \mathrm{T}_{j,\tau}^{below}(\vec{x}) = -1\}}{norm^{pos}}$$

$$adv \leftarrow \max\left\{\left(\frac{gain^{above}}{\omega \cdot loss^{above}}\right),\left(\frac{gain^{below}}{\omega \cdot loss^{below}}\right)\right\}$$

Falls *adv* > *maxAdv* so führe aus:

$maxAdv \leftarrow adv$

Falls $\frac{gain^{above}}{\omega \cdot loss^{above}} > \frac{gain^{below}}{\omega \cdot loss^{below}}$ so führe aus:

$$\left(j_k^{\max}, \tau_k^{\max}, d_k^{\max}\right) \leftarrow (j, \tau, above)$$

Sonst führe aus:

$$\left(j_k^{\max}, \tau_k^{\max}, d_k^{\max}\right) \leftarrow (j, \tau, below)$$

Falls *maxAdv* > 1 so führe aus:

$$H_{k+1} \leftarrow H_k \cup \left\{T_{j_k^{\max}, \tau_k^{\max}}^{d_k^{\max}}\right\}$$

$$U_{k+1} \leftarrow \left\{(\vec{x}, y) \mid (\vec{x}, y) \in U_k \wedge H_{k+1}(\vec{x}) = 1 \wedge y \in \{-1, +1\}\right\}$$

$$k \leftarrow k + 1$$

Solange *maxAdv* > 1 und $k < \hat{p}$

Return H_k

Der Parameter ω erlaubt eine Gewichtung der Positivbeispiele bei der Bestimmung des Advantage-Quotienten *adv.* Auf diese Weise lässt sich der Fehler 1. Art und Fehler 2. Art bis zu einem gewissen Grad skalieren.
In der Praxis zeigt sich weiterhin, dass eine vereinfachte Variante des Quotienten *adv* bessere und deutlich stabilere Ergebnisse lieferte:

$$adv \leftarrow \max\left(\left(gain^{above} - \omega \cdot loss^{above}\right),\left(gain^{below} - \omega \cdot loss^{below}\right)\right)$$

Die Hauptschleife terminiert in diesem Szenario nicht, solange *maxAdv* > 0 gilt. Es ist zu überlegen, ob in der Praxis ein Abbruchwert größer 0 zu besserer Generalisierung führt, da so einzelne Merkmale, die eventuell aufgrund von einzelnen Ausreißer-Beispielen ein noch positives Advantage liefern, nicht in das Monom aufgenommen werden.
Weiterhin wird in der Praxis ein Ausgleichsverfahren (Equalizer) für die Schwellwerte verwendet.

Dabei wird eine gewählte Schwelle so verändert, dass die Klassifikationsrate unbeeinflusst bleibt:

$$\tau_j^{equalized} = \frac{\tau_j + \min\{(e - \tau_j) \mid e \in S_j \wedge \tau_j < e\}}{2}$$

Auf diese Weise liegt der ausgeglichene Schwellwert stets zwischen zwei Trainingsbeispielen.

Die Heuristik wird nun erweitert, so dass die Konzeptklasse $C_{\hat{p},\hat{r}}^{DNF}$ gelernt werden kann.
Die Grundidee dafür ist die mehrfache Anwendung des Lernalgorithmus THSM für die Klasse $C_{\hat{p}}^{Monom}$. Diesem werden die Beispiele, die durch das erste Monom als negativ klassifiziert werden, erneut vorgesetzt. Durch die neu hinzugekommene Alternative besteht die Möglichkeit, dass gegenwärtig falsch klassifizierte Positivbeispiele ein korrektes, positives Label erhalten. Die Auswahl von Schwellen erfolgt also nach denselben Kriterien, lediglich auf eingeschränkter Datenbasis. Der folgende Algorithmus leistet das Gewünschte:

Algorithmus 7 Thresholder Heuristic Multiple Monoms (THMM)

Eingabe: $U[1,m] = ((\vec{x}_1, y_1) = ((x_{11},...,x_{1n}), y_i),...,(\vec{x}_m, y_m) = ((x_{m1},...,x_{mn}), y_m))$, $\omega > 0$,
$\hat{p} \in N^+$, $\hat{r} \in N^+$

Ausgabe: Eine disjunktive Normalform als Hypothese

Initialisierung
$U_0 \leftarrow \{(\vec{x}_i, y_i) \mid i = 1,...,m\}$
$H_0 \leftarrow \{\ \}$

Für $i = 1,2,...,\hat{r}$ führe aus:
$\mu_i \leftarrow THSM(U_{i-1})$
$H_i \leftarrow H_{i-1} \cup \mu_i$
$U_i \leftarrow \{(\vec{x}, y) \mid (\vec{x}, y) \in U_{i-1} \wedge \mu_i(\vec{x}) = -1\}$

Return H_i

4.6. Effiziente Schwellwertabtastung für Klauseln

Die Berechnung der Wahrscheinlichkeiten für Statistical Queries, Klassifikationsraten sowie der Überdeckungsanteile für den im nächsten Kapitel vorgestellten Algorithmus lassen sich mit Hilfe der dynamischen Programmierung für feste Merkmale und Schwellwertfunktionstypen einer Klausel optimieren. Dazu werden die Größen *TP* (Wahr Positiv), *FP* (Falsch Positiv), *TN* (Wahr Negativ) und *FN* (Falsch Negativ) anhand der Lernstichprobe bestimmt. Der folgende Algorithmus demonstriert das Vorgehen für die Größen *TP* und *FN* bei einer Klausellänge von zwei. Weiterhin werden mögliche Negationen der Literale einer Klausel zur Vereinfachung nicht berücksichtigt. Der Ansatz lässt sich analog auf Klauseln beliebiger Länge verallgemeinern:

Algorithmus 5 Efficient Threshold Scan (TP, FN)

Eingabe: $U[1,m] = ((\vec{x}_1,+1) = ((x_{11},...,x_{1n}),+1),...,(\vec{x}_m,+1) = ((x_{m1},...,x_{mn}),+1)),$

$d_1, d_2 \in \{above, below\},\ j_1, j_2 \in \{1,...,n\}$

Ausgabe: Für jede 2-Schwellwertkombination die Anteile *TP* und *FN* bezogen auf $U[1,m]$

Initialisierung

$S_j \leftarrow \{x_{1j}, x_{2j},..., x_{mj}\}$ für $j = 1,...,n$

$(a_1,...,a_m) \in \{(a_1,...,a_m) \mid x_{a_1 j_1} \le x_{a_2 j_1} \le ... \le x_{a_m j_1}\}$ // Indizes der Sortierung nach Merkmal j_1

$(b_1,...,b_m) \in \{(b_1,...,b_m) \mid x_{b_1 j_2} \le x_{b_2 j_2} \le ... \le x_{b_m j_2}\}$ // Indizes der Sortierung nach Merkmal j_2

$$(y_1^{pred},...,y_m^{pred}) \leftarrow \begin{cases} \underbrace{(-1,...,-1)}_{m} & falls\ d_1 = above \\ \underbrace{(+1,...,+1)}_{m} & falls\ d_1 = below \end{cases}$$

$R \leftarrow \{\ \}$

$i \leftarrow 1$

Für jedes $s_1 \in S_{j_1}$ in aufsteigender Reihenfolge führe aus:

Solange $x_{a_i j_1} \le s_1$ führe aus:

Falls $T_{j_1,s_1}^{d_1}(x_{a_i j_1}) = -1$ so führe aus:

$FN_1 \leftarrow FN_1 + 1$

$TP_1 \leftarrow m - FN_1$

$y_{a_i}^{pred} \leftarrow -1$

Sonst führe aus:

$TP_1 \leftarrow TP_1 + 1$

$FN_1 \leftarrow m - TP_1$

$y_{a_i}^{pred} \leftarrow +1$

$i \leftarrow i + 1$

$$TP_2 \leftarrow \begin{cases} TP_1 & falls\ d_2 = above \\ m & sonst \end{cases}$$

$$FN_2 \leftarrow \begin{cases} FN_1 & falls\ d_2 = above \\ 0 & sonst \end{cases}$$

$k \leftarrow 1$

Für jedes $s_2 \in S_{j_2}$ in aufsteigender Ordnung führe aus:

Solange $x_{b_k j_2} \leq s_2$ führe aus:

Falls $T_{j_2, s_2}^{d_2}(x_{b_k j_2}) = -1$ so führe aus:

Falls $y_{b_k}^{pred} = -1$ so führe aus:

$FN_2 \leftarrow FN_2 + 1$

$TP_2 \leftarrow m - FN_2$

Sonst führe aus:

Falls $y_{b_k}^{pred} = -1$ so führe aus:

$TP_2 \leftarrow TP_2 + 1$

$TP_2 \leftarrow m - FN_2$

$k \leftarrow k + 1$

$R \leftarrow R \cup \{(s_1, s_2, TP_2, FN_2)\}$

Return *R*

Die Berechnung der Größen *FP* und *TN* erfolgt vollständig analog anhand der negativen Beispiele der Lernstichprobe.

4.7. Lernalgorithmus durch Statistical Queries

Zum leichteren Verständnis und um die Verwandtschaft zum Algorithmus von Haussler (siehe [Hau88]) zu verdeutlichen, werden zunächst konjunktive Normalformen gelernt. Diese können anschließend in eine disjunktive Normalform überführt werden. Als Grundlage für die erste Phase dient die Konzeptklasse $C_{\hat{p}}^{Klausel}$. Für die Klasse $C_{\hat{p},\hat{r}}^{KNF}$ wird der Parameter $\hat{r} \in N^+$ fixiert.

Der folgende Algorithmus stellt im Modell mit Rauschen auf der Klassifikation die Konsistenz mit den positiven Beispielen her. Die genaue Analyse des Algorithmus und die Herleitung der Genauigkeitsparameter ist nicht Thema dieser Arbeit, sondern wird in der zugehörigen Publikation (siehe [WT08]) genau behandelt.

Algorithmus 8 (Phase 1: Haussler Kearns Prune Stage)

Eingabe: $U[1,m] = ((\vec{x}_1,+1) = ((x_{11},...,x_{1n}),+1),...,(\vec{x}_m,+1) = ((x_{m1},...,x_{mn}),+1)),\ \hat{r} \in N^+,\ \varepsilon > 0$

Ausgabe: Mengen $R_1,...,R_{\beta_*}$ von Klauseln, konsistent mit den Positivbeispielen

Initialisierung

$$z \leftarrow \begin{cases} \lceil 2\hat{r} \cdot \ln(2/\varepsilon) \rceil & falls\ \hat{r} > 1 \\ 1 & falls\ \hat{r} = 1 \end{cases}$$

$$\gamma_1 \leftarrow \left\lfloor \frac{\varepsilon}{4z} \right\rfloor$$

$$\gamma_2 \leftarrow \left\lfloor \frac{\varepsilon^2}{4(\varepsilon+6)} \right\rfloor$$

$$\gamma_* \leftarrow \min\{\gamma_1, \gamma_2\}$$

$$\beta_* \leftarrow \left\lfloor \frac{3\eta_0}{\gamma_*(1-2\eta_0)(\sqrt{2}-1)} \right\rfloor$$

Berechne $\hat{e}_{-1,c} = P(Y=-1, c(X)=-1) = \frac{TN}{m}$ für alle $c \in C_{\hat{p}}^{Klausel}$

Berechne $\hat{e}_{+1,c} = P(Y=+1, c(X)=-1) = \frac{FN}{m}$ für alle $c \in C_{\hat{p}}^{Klausel}$

Berechne $\hat{e}_{+1} = P(Y=+1)$

Für $i = 1,..., \beta_*$ führe aus:

$$\hat{\eta} \leftarrow \frac{(i-1/2)\eta_b}{\beta_*}$$

$$R_i \leftarrow \{\ \}$$

Für alle $c \in C_{\hat{p}}^{Klausel}$ führe aus:

Berechne $\hat{p}_{c,i} = P(c(X) = -1 \mid g(X) = +1) = \frac{\left((1-\hat{\eta})\hat{e}_{+1,c} - \hat{\eta}\hat{e}_{-1,c}\right)}{\hat{e}_{+1} - \hat{\eta}}$

Falls $\hat{p}_{c,i} \le \frac{\varepsilon}{2z}$ führe aus:

$$R_i \leftarrow R_i \cup \{c\}$$

Return $(R_1,..., R_{\beta_*})$

Die zweite Phase des Algorithmus gewinnt die finalen Hypothesen durch Überdeckung der Negativbeispiele mit Hilfe der Klauseln aus den Mengen $(R_1,...,R_{\beta_*})$. Dem Algorithmus steht die in Phase 1 berechnete Variable β_* weiterhin zur Verfügung.

Algorithmus 9 (Phase 2: Haussler Kearns Covering Stage)

Eingabe: $U[1,m] = ((\vec{x}_1,-1) = ((x_{11},...,x_{1n}),-1),...,(\vec{x}_m,-1) = ((x_{m1},...,x_{mn}),-1)),\ \hat{r} \in N^+,\ \varepsilon > 0$

Ausgabe: Konjunktive Normalformen $\{H_1, H_2,..., H_{\beta_*}\}$ als Hypothesen

Initialisierung

Für $i = 1,..., \beta_*$ führe aus:

$U_i \leftarrow \{(\vec{x}_j, y_j) \mid j = 1,..., m \wedge y_i = -1\}$

$H_i \leftarrow \{\ \}$

$\hat{q}_i \leftarrow 1$

$j \leftarrow 0$

Solange $j < \hat{r}$ führe aus:

$j \leftarrow j + 1$

Für $i = 1,..., \beta_*$ führe aus:

$$\hat{\eta} \leftarrow \frac{(i - 1/2)\eta_0}{\beta_*}$$

$$\hat{\theta} \leftarrow \frac{P(Y=-1) - \hat{\eta}}{(1-2\hat{\eta})(1-P(Y=-1))}$$ // Es gilt $\hat{\theta} = \hat{\eta}$ für $P(Y=-1) = 1/2$

Falls $\hat{q}_i > \hat{\theta}$ so führe aus:

$$c_i \leftarrow \arg\max_{c \in R_i} \left| \{(\vec{x}_j, \vec{y}_j) \mid c(\vec{x}_j) = -1 \wedge j = 1,..., m\} \right|$$

$$U_i \leftarrow U_i - \{(\vec{x}_j, \vec{y}_j) \mid c_i(\vec{x}_j) = -1 \wedge j = 1,..., m\}$$

$R_i \leftarrow R_i - \{c_i\}$

$H_i \leftarrow H_i \cup \{c_i\}$

$$\hat{q}_i \leftarrow \frac{\#U_i}{m}$$

Return $\{H_1, H_2,..., H_{\beta_*}\}$

Da für die praktische Anwendung disjunktive Normalformen gelernt werden sollen, muss eine entsprechende Anpassung erfolgen. Es gilt wie bisher (siehe Kapitel 4.1.2):

$$\lambda = \mu_1 \vee \mu_2 \vee ... \vee \mu_r = \left(\alpha_{11} \wedge \alpha_{12} \wedge ... \wedge \alpha_{1p_1}\right) \vee \left(\alpha_{21} \wedge \alpha_{22} \wedge ... \wedge \alpha_{2p_2}\right) \vee ... \vee \left(\alpha_{r1} \wedge \alpha_{r2} \wedge ... \wedge \alpha_{rp_r}\right)$$

Die disjunktive Normalform ist gegeben durch:

$$\begin{aligned}
\lambda &= \left(\mu_1 \vee \mu_2 \vee ... \vee \mu_r\right) \\
&= \overline{\overline{\left(\mu_1 \vee \mu_2 \vee ... \vee \mu_r\right)}} \\
&= \overline{\overline{\left(\alpha_{11} \wedge \alpha_{12} \wedge ... \wedge \alpha_{1p_1}\right) \vee \left(\alpha_{21} \wedge \alpha_{22} \wedge ... \wedge \alpha_{2p_2}\right) \vee ... \vee \left(\alpha_{r1} \wedge \alpha_{r2} \wedge ... \wedge \alpha_{rp_r}\right)}} \\
&= \overline{\overline{\left(\alpha_{11} \wedge \alpha_{12} \wedge ... \wedge \alpha_{1p_1}\right)} \wedge \overline{\left(\alpha_{21} \wedge \alpha_{22} \wedge ... \wedge \alpha_{2p_2}\right)} \wedge ... \wedge \overline{\left(\alpha_{r1} \wedge \alpha_{r2} \wedge ... \wedge \alpha_{rp_r}\right)}} \\
&= \overline{\left(\overline{\alpha_{11}} \vee \overline{\alpha_{12}} \vee ... \vee \overline{\alpha_{1p_1}}\right) \wedge \left(\overline{\alpha_{21}} \vee \overline{\alpha_{22}} \vee ... \vee \overline{\alpha_{2p_2}}\right) \wedge ... \wedge \left(\overline{\alpha_{r1}} \vee \overline{\alpha_{r2}} \vee ... \vee \overline{\alpha_{rp_r}}\right)}
\end{aligned}$$

Auf diesem Wege kann die Logik des Algorithmus beibehalten werden. Es ist lediglich notwendig, die Teilstichproben der beiden Phasen zu vertauschen. In Phase 1 wird die Konsistenz mit den negativen Beispielen erzielt, indem lediglich Klauseln herangezogen werden, die das Label +1 liefern. Durch die äußere Negation wird dann das korrekte Label zugewiesen. In Phase 2 werden Klauseln zur Überdeckung gewählt, die der jeweils maximalen Anzahl von Positivbeispielen das Label -1 zuweisen. Die Berechnung der Wahrscheinlichkeit muss dann entsprechend angepasst werden zu:

$$\hat{p}_{c,i} = P(c(X) = +1 \mid g(X) = -1) = \frac{\left((1-\hat{\eta})\hat{e}_{-1,c} - \hat{\eta}\hat{e}_{+1,c}\right)}{\hat{e}_{-1} - \hat{\eta}}$$

Die einzelnen Merkmale der Klauseln sind entsprechend der obigen Umformung zu negieren. Also ist stets $c \in \neg C_{\hat{p}}^{Klausel}$. Dabei ist $\neg C_{\hat{p}}^{Klausel}$ die Menge aller Klauseln, deren Literale jeweils negiert sind.

4.8. Rademacher-Komplexität der vorgestellten Konzeptklassen

In diesem Kapitel soll die Rademacher-Komplexität anhand einer Abschätzung für die Konzeptklasse $C_{n,r}^{DNF}$ bestimmt werden. Dazu wird zunächst die Größe der Konzeptklasse C_n^{Monom} nach oben abgeschätzt. Es gilt:

$$\#C_{\hat{p}}^{Monom} = \sum_{i=1}^{\hat{p}} \binom{n}{i} \cdot (2m)^i$$

Im Allgemeinen ist $\hat{p} = n$ und somit folgt:

$$\#C_n^{Monom} = \sum_{i=1}^{n} \binom{n}{i} \cdot (2m)^i$$

Unter Anwendung der binomischen Reihe

$$\sum_{i=0}^{\infty} \binom{n}{i} \cdot x^i = (1+x)^n$$

folgt schließlich:

$$\#C_n^{Monom} = (1+2m)^n - 1$$

Diese Formel lässt sich direkt auf die Konzeptklasse $C_{n,\hat{r}}^{DNF}$ anwenden und nach oben abschätzen durch:

$$\#C_{n,\hat{r}}^{DNF} = \sum_{i=1}^{\hat{r}} \binom{(1+2m)^n - 1}{i} \leq \hat{r} \cdot \left((1+2m)^n - 1\right)^{\hat{r}}$$

Die Anzahl der Funktionen in Patternklasse und Konzeptklasse ist gleich, da die Verlustfunktion stets fixiert ist. Die Bezeichnungen der Patternklassen sind völlig analog zu den Bezeichnungen der Konzeptklassen gewählt. Daher folgt für die Rademacher-Komplexität:

$$\begin{aligned} rc_m(\mathrm{P}_n^{Monom}) &\leq \frac{c}{2\sqrt{m}} \sqrt{2 \cdot \ln\left|\mathrm{P}_n^{Monom}\right|} \\ &= \frac{1}{\sqrt{m}} \sqrt{2 \cdot \ln\left((1+2m)^n - 1\right)} \\ &\approx \frac{1}{\sqrt{m}} \sqrt{2 \cdot \ln\left((1+2m)^n\right)} \\ &= \frac{1}{\sqrt{m}} \sqrt{2n \cdot \ln\left((1+2m)\right)} \end{aligned}$$

bzw.

$$\begin{aligned} rc_m(\mathrm{P}_{n,\hat{r}}^{DNF}) &\leq \frac{c}{2\sqrt{m}} \sqrt{2 \cdot \ln\left|\mathrm{P}_{n,\hat{r}}^{DNF}\right|} \\ &= \frac{1}{\sqrt{m}} \sqrt{2 \cdot \ln\left(\hat{r} \cdot \left((1+2m)^n - 1\right)^{\hat{r}}\right)} \\ &= \frac{1}{\sqrt{m}} \sqrt{2 \cdot \left(\ln(\hat{r}) + \ln\left(\left((1+2m)^n - 1\right)^{\hat{r}}\right)\right)} \\ &\approx \frac{1}{\sqrt{m}} \sqrt{2 \cdot \left(\ln(\hat{r}) + \ln\left(\left((1+2m)^n\right)^{\hat{r}}\right)\right)} \\ &= \frac{1}{\sqrt{m}} \sqrt{2 \cdot \left(\ln(\hat{r}) + n \cdot \hat{r} \cdot \ln\left((1+2m)\right)\right)} \end{aligned}$$

Für die Insolvenzvorhersage mit *m = 400*, *n = 18* und *r = 3* (siehe Kapitel 5) ergibt sich:

$$rc_m(\mathrm{P}_3^{Monom}) \leq \frac{1}{\sqrt{400}} \sqrt{2 \cdot \ln\left((1+2 \cdot 400)^{18} - 1\right)} \approx 0{,}77$$

$$rc_m(\mathrm{P}_{18,3}^{DNF}) \leq \frac{1}{\sqrt{400}} \sqrt{2 \cdot \ln\left(3 \cdot \left((1+2 \cdot 400)^{18} - 1\right)^3\right)} \approx 1{,}35$$

Für die Hotspotidentifikation mit *m = 3672*, *n = 20* und $\hat{r} = 7$ (siehe Kapitel 6) ergibt sich:

$$rc_m(\mathrm{P}_{20}^{Monom}) \leq \frac{1}{\sqrt{3672}} \sqrt{2 \cdot \ln\left((1+2 \cdot 3672)^{20} - 1\right)} \approx 0{,}31$$

$$rc_m(\mathrm{P}_{20,7}^{DNF}) \leq \frac{1}{\sqrt{3672}} \sqrt{2 \cdot \left(\ln(7) + 20 \cdot 7 \cdot \ln\left((1+2 \cdot 3672)\right)\right)} \approx 0{,}82$$

Die Konzeptklasse $\#C_{\hat{p},\hat{r}}^{DNF}$ wird bei Anwendung auf das Problem der Hotspot-Identifikation zusätzlich auf $\hat{p} = 2$ eingeschränkt. Daraus ergibt sich eine Rademacher-Komplexität von:

$$rc_m(\mathrm{P}_{2,7}^{DNF}) \leq \frac{1}{\sqrt{3672}} \sqrt{2 \cdot \left(\ln(7) + 7 \cdot \ln\left(\sum_{i=1}^{2} \binom{n}{i} \cdot (2 \cdot 3672)^i\right)\right)} \approx 0{,}30$$

Eine Beschränkung der Literale pro Monom senkt die Rademacher-Komplexität also stark ab.

5. Anwendung auf das Problem der Insolvenzerkennung

Im Folgenden sollen die vorgestellten Verfahren auf das Problem der Insolvenzerkennung angewendet werden. Dieses Kapitel soll Aufschluss über die verwendeten Jahresabschlussdaten und die damit verbundenen Probleme bei deren Auswahl geben, um repräsentative Ergebnisse gewinnen zu können. Anschließend soll auf die in dieser Arbeit verwendeten Kennzahlen eingegangen werden. Zum Abschluss werden die vorgestellten Verfahren zur Insolvenzvorhersage miteinander verglichen und bewertet.

5.1. Einführung

Ein wichtiger Teil bei einer Insolvenzerkennung ist die Jahresabschlussanalyse (siehe [BS96], [D06] und [BE00]). Dabei wird versucht, die Bonität von Unternehmen anhand ihrer Jahresabschlüsse einzuschätzen. In diesem Rahmen werden Bilanzkennzahlen genutzt, um zum einen Vergleichbarkeit zu anderen Unternehmen herstellen zu können und zum anderen verschiedene Information kompakt darzustellen. Nur auf diese Weise kann eine Analyse auf aussagekräftigen Merkmalen vorgenommen werden. Es ist jedoch leicht einzusehen, dass eine Jahresabschlussanalyse keine eindeutigen Ergebnisse erzielen kann, da die Faktoren für die Insolvenz eines Unternehmens zu komplex sind, als dass sie allein in Jahresabschlussdaten erfasst sein könnten.
Da Jahresabschlussanalysen sehr aufwändig sind, und die Verarbeitung zahlreicher Merkmale (Kennzahlen) für den Menschen nur begrenzt bis gar nicht möglich ist, besteht eine Nachfrage nach automatisierten Verfahren.
Ziel ist es außerdem, die Effizienz der Verarbeitung von Kreditanfragen zu steigern, da meist eine Vielzahl von Kreditnehmern existiert.
Da Jahresabschlussdaten eine Form von quantitativen Daten sind, bietet es sich an statistische Verfahren anzuwenden, insbesondere die Verfahren des maschinellen Lernens.
Um eine Analyse vornehmen zu können, ist also zunächst die Frage nach geeigneten Merkmalen (Kennzahlen) zu klären. In diesem Zusammenhang besteht die Anforderung der Objektivität (siehe [D06]) der Merkmalsauswahl, die im Falle der subjektiven Auswahl von Experten häufig verletzt ist. Es ist jedoch nicht von der Hand zu weisen, dass eine subjektive Auswahl auch den Einfluss von Erfahrungswerten enthält und daher, wenn auch statistisch nicht begründbar, durchaus Sinn macht. Bei der Anwendung von statistischen Verfahren und insbesondere denen des maschinellen Lernens besteht permanent die Gefahr des Lernens von Artefakten, die in einem speziellen Testszenario viel versprechend erscheinen, jedoch nicht zu verallgemeinern

sind. Daher kann eine automatisierte Merkmalsauswahl häufig auf Eigenheiten innerhalb eines Datensatzes zurückzuführen sein, wenn diese nicht nach bestimmten Kriterien erfolgt.
Eine weitere Anforderung an ein Verfahren ist die Minimierung bzw. Skalierbarkeit des Fehlers 1. Art. In der Testtheorie wird mit dem Fehler 1. Art die Wahrscheinlichkeit die Nullhypothese abzulehnen bezeichnet, obwohl sie wahr ist. Im Falle der Insolvenzerkennung ist die Nullhypothese „Das Unternehmen ist insolvent“ und damit der Fehler 1. Art die Wahrscheinlichkeit ein Unternehmen fälschlicherweise als solvent einzustufen. In diesem Fall erhält der Kreditgeber den Kreditbetrag nicht zurück und dies bedeutet hohen wirtschaftlichen Schaden. Dies steht im Gegensatz zum Fehler 2. Art, bei dem ein solventes Unternehmen fälschlicherweise als insolvent eingestuft wird und somit dem Kreditgeber ein Opportunitätsverlust entsteht (siehe [BS96]).
Eine weitere Anforderung ist die Transparenz des Verfahrens (siehe [D06]). Im Normalfall wird die Bewilligung einer Kreditanfrage nicht alleine durch ein automatisiertes Verfahren entschieden. Daher wird erforderlich, dass die Ursachen für eine Einordnung nachvollziehbar bleiben, bzw. dass erlernte Zusammenhänge angemessen darstellbar sind, um gegebenenfalls den Wirtschaftsexperten bei der nachfolgenden Analyse zu unterstützen.

5.2. Datenlage

Das erste Problem zur Validierung eines Modells zur Insolvenzvorhersage ist die Beschaffung brauchbarer Trainings- und Testdaten. Mehrere Anläufe, brauchbare Daten zu beschaffen, scheiterten in diesem Projekt.
Durch persönliche Korrespondenz konnten zum einen Daten des Kreditversicherers Coface beschafft werden, die sich jedoch als Grundlage der Analyse nicht eignen. Dieser Datensatz besteht aus 475 solventen und 473 insolventen Jahresabschlüssen von insgesamt 100 Unternehmen. Die Datenlage an sich ist zu gering, um eine Gruppierung der Unternehmen nach Branchen vorzunehmen. Des Weiteren erweist sich die Bildung von Kennzahlen als unmöglich, da die meisten Positionen der Jahresabschlüsse keine Werte besitzen. Lediglich eine Position ist von 93% der Unternehmen befüllt. Die verbleibenden Positionen liegen schnell unter 50% und weniger.
Da an diesem Datensatz lediglich eine Analyse auf Rohmerkmalen möglich ist, die wirtschaftlich aber weder anerkannt noch Sinn zu machen scheint, werden die Experimente auf einem Datensatz des Verbandes der Vereine Creditreform e.V. durchgeführt.

Die vom Verband der Vereine Creditreform e.V. angebotene Datenbank DAFNE (siehe [DAFNE]) kann für einen Pauschalpreis abboniert werden und stellt Jahresabschlüsse von rund 90.000 Unternehmen zur Verfügung. DAFNE ist im Rahmen des Projekts abonniert worden und bietet eine relativ vollständige Datenbasis, inklusive mehrerer Jahresabschlüsse pro Unternehmen. Ein Unternehmen wird dann durch den jeweils neuesten Jahresabschluss repräsentiert, da davon ausgegangen werden muss, dass Zeichen für eine bevorstehende Insolvenz in älteren Jahresabschlüssen unter Umständen schwerer erkennbar sind. Die Daten erlauben die Bildung verschiedener Kennzahlen, die auch in anderen Arbeiten, insbesondere in der von Uthoff (siehe [U97]) verwendet werden.

Trotz der großen Anzahl von Unternehmen, ist die Gewinnung der Labels (Solvent/Insolvent) sehr problematisch. Nach deutschem Recht ist die Bekanntgabe des Beginns eines Insolvenzverfahrens gegen ein Unternehmen nur 14 Tage öffentlich zugänglich zu machen. Nach dieser Frist stehen Details über den Status des Insolvenzverfahrens nur mit Hilfe von detaillierten Informationen über das Schuldnerunternehmen durch eine Suchfunktion beim Bundesministerium (siehe [BDI]) zur Verfügung.

Eine skriptgesteuerte Auswertung dieser Datenbank im Rahmen einer wissenschaftlichen Arbeit ist wie erwartet nicht zulässig. Das ist auch durch eine persönliche Korrespondenz mit dem Bundesministerium nochmals bestätigt worden.

Dieses Problem kommt auch in der Datenbank DAFNE der Creditreform zum Tragen und erschwert demnach die Pflege der Solvent/Insolvent Informationen. Dementsprechend liegen lediglich 220 insolvente Unternehmen verschiedener Branchen vor.

Aufgrund dieses Szenarios ist zu erwarten, dass auch in der Auswahl solventer Unternehmen einige, wenn auch wenige, tatsächlich insolvente Unternehmen zu finden sind, die eine Verzerrung verursachen können. Eine Statistik der Creditreform e.V. besagt, dass das Verhältnis von solventen zu insolventen Unternehmen etwa 100:1 entspricht. Daher wäre bei einer Datenlage von 90.000 Unternehmen mit etwa 900 insolventen Unternehmen zu rechnen, jedoch lediglich 24% dieser Menge liegen tatsächlich vor.

Aufgrund dieser Tatsachen ist eine Branchengruppierung nur mit einer Merkmalseinschränkung möglich: Wird eine Gruppierung nach Branche vorgenommen, so entstehen kleine Gruppen von maximal 29 Unternehmen, die bei der Verwendung mehrerer Kennzahlen keine repräsentativen Ergebnisse liefern können, denn es besteht die Gefahr, dass durch reinen Zufall trennscharfe Merkmale existieren, die einzig und allein auf die Zusammensetzung der Daten zurückzuführen sind und dadurch die Generalisierung schlecht ist.

Alternativ können mehrere Branchen in einem Datensatz verarbeitet werden, wodurch allerdings mit einer größeren Inhomogenität zu rechnen ist.
Um die Daten geringfügig zu homogenisieren, werden alle insolventen Unternehmen entfernt, die alleinige Vertreter einer Branche im Datensatz sind. Anders ausgedrückt, jede Branche wird nach der Bereinigung von mindestens zwei Unternehmen im Datensatz repräsentiert. Nach dieser Selektion liegen noch 177 insolvente Unternehmen vor, die die Grundlage für die Analyse bilden sollen.
Die Auswahl vergleichbarer solventer Unternehmen geschieht auf zufälliger Basis nach Profilen. Das heißt für jedes insolvente Unternehmen wird genau ein zufälliges, solventes Unternehmen aus derselben Branche gezogen, so dass die Stichprobe der solventen Unternehmen exakt dieselbe Anzahl an Repräsentanten pro Branche besitzt. Dies soll vor allem gegen dem Einwand vorbeugen, dass branchenspezifische Merkmale für eine Trennung verantwortlich sind.

Die Erkennung insolventer Unternehmen besitzt eine höhere Priorität. Daher macht es Sinn, dieselbe Anzahl solventer sowie insolventer Unternehmen zu verwenden, denn bei Verfahren des maschinellen Lernens gehen die Häufigkeiten der einzelnen Labels der Trainingsstichprobe in die Bewertung ein, so dass die Vorhersage eines häufig vorkommenden Labels wahrscheinlicher ist.

5.3. Behandlung von fehlenden Werten in Jahresabschlüssen

Die vorliegenden Jahresabschlüsse sind im Bezug auf die angegebenen Positionen und Kennzahlen lückenhaft. Dies ist ein häufiges Problem bei Jahresabschlussanalysen, dem auf verschiedene Weisen begegnet werden kann.
Eine mögliche Lösung des Problems ist es, wie bereits von Uthoff (siehe [U97]) beschrieben, einen unvollständigen Jahresabschluss von der Analyse auszuschließen. Dies kommt jedoch aufgrund der spärlichen Datenlage insolventer Unternehmen nicht in Frage.

Alternativ kann der volle Datensatz in Kombination mit einer der folgenden Vervollständigungsstrategien verwendet werden:

- **S1** Der Datensatz wird in seiner unvollständigen Struktur in die Analyse einbezogen
- **S2** Das fehlende Merkmal wird in der Analyse durch einen Schätzwert ersetzt
  - **S2a** Der Wert wird gleich 0 gesetzt
  - **S2b** Merkmalsspezifischer Mittelwert: Bildung des Mittelwertes über alle zur Verfügung stehenden Datensätze
  - **S2c** Gruppenspezifischer Mittelwert: Bildung des Mittelwertes über die Datensätze einer Gruppe (Solvent/Insolvent)
- **S3** Unvollständige Merkmale werden von der Analyse ausgeschlossen

Die Auswahl der Strategie ist nicht zwingend für alle Verfahren gleich zu treffen. Im Falle der MDA ist die Strategie *S2a* nicht geeignet, da Null-Werte die Erwartungswertschätzungen erheblich stören können. Strategie *S3* sollte ebenfalls vermieden werden, da nur sehr wenige Jahresabschlüsse (maximal drei pro Merkmal) einen fehlenden Wert aufweisen. Aus diesen Gründen ist Strategie *S2* die angemessene Variante.
Alle Verfahren die auf Distanzen und Schätzungen zurückgreifen, sollten den gruppenspezifischen Mittelwert *S2c* als Schätzer verwenden, um die Verzerrung möglichst gering zu halten. Für die Validierungsstichprobe hingegen muss der merkmalsspezifische Mittelwert *S2b* verwendet werden, da ansonsten Information über die wahrhaftige Klassifikation in die Validierungsstichprobe einfließt, was zu nicht repräsentativen Ergebnissen führen kann.

5.4. Kennzahlen und Jahresabschlüsse

Betriebswirtschaftliche Kennzahlen werden vorwiegend für die Beurteilung von Unternehmen oder deren Teilbereiche eingesetzt (siehe [SH01]). Sie haben weiterhin eine wichtige Funktion bei der Festlegung und anschließenden Überprüfung von Unternehmenszielen. Dabei sollen sie bestimmte Sachverhalte möglichst gut darstellen, um Probleme innerhalb eines Unternehmens erkennen zu können oder generell Informationen liefern. Weiterhin dienen sie der Kontrolle eines Unternehmens und schaffen Möglichkeiten für die Dokumentation und Koordination. Das vordergründige Ziel besteht darin, Informationen zu verdichten, so dass sie für einen bestimmten Sachverhalt eine deutliche Aussagekraft besitzen.

In Deutschland schreibt das Handelsgesetzbuch (HGB) vor, welche Unternehmen Daten ihrer Geschäftstätigkeit offen legen müssen. Unternehmen mit internationalem Tätigkeitsbereich unterliegen mitunter weiteren Vorschriften zur Offenlegung von Daten (z.B. nach US-GAAP oder IFRS/IAS). Welche Daten zu veröffentlichen sind, hängt in der Regel von der Größe und der Rechtsform des Unternehmens ab.
Zu den Daten, die veröffentlicht werden müssen, zählen vor allem Jahresabschlüsse und Geschäftsberichte. Aus den Positionen der Bilanz oder der Gewinn- und Verlustrechnung lassen sich dann Kennzahlen berechnen. Mitunter werden aber auch Kennzahlen direkt angegeben. Dies gilt insbesondere für solche Kennzahlen, die für das jeweilige Unternehmen einen besonderen Stellenwert besitzen.
Über ihre Offenlegungspflicht hinaus ist es den Unternehmen, bis auf wenige Ausnahmen, freigestellt, weitere Informationen zu veröffentlichen. Viele Unternehmen veröffentlichen daher im Rahmen der Öffentlichkeitsarbeit einen Teil ihrer Kennzahlen auf freiwilliger Basis.
Die Berechnung von Kennzahlen ist jedoch weitestgehend nicht standardisiert und variiert international. Aufgrund dessen ist ein länderübergreifender Unternehmensvergleich im Allgemeinen nicht möglich.
Kennzahlen, die in Beziehung zueinander stehen, können zu einem Kennzahlensystem zusammengefasst werden, die wiederum auf logischen, empirischen oder hierarchischen Zusammenhängen beruhen. Sie werden je nach System eingesetzt, um Informationen stark zu verdichten oder um Aufgaben im Bereich der Planung, Steuerung und Kontrolle zu unterstützen.

Ein weiteres Einsatzgebiet von Kennzahlen ist das so genannte Benchmarking. Dabei versuchen Unternehmen möglichst ähnliche Fremdunternehmen zu finden, um eine Vergleichbarkeit herzustellen. Auf der Basis von ausgewählten, für die Geschäftstätigkeit wichtigen Kennzahlen, wird versucht, die eigene Stellung im Vergleich zu den Fremdunternehmen bewerten zu können. Einem Unternehmen wird damit aufgezeigt, in welchen Bereichen es sich lohnt Prozesse und Strategien zu optimieren.
Kennzahlen bilden die Basis für das Treffen von Managemententscheidungen. Dies ist zugleich einer der wichtigsten Verwendungszwecke und steht in einem direkten Zusammenhang zu dem Gedanken der Informationsverdichtung. Die Unternehmensdaten werden für die Entscheidungsträger aufbereitet, um Entscheidungsprozesse zu vereinfachen und zu beschleunigen. Auswirkungen lassen sich in der Folge relativ einfach überprüfen.
Weiterhin werden Kennzahlen häufig in zwei Kategorien gegliedert, die absoluten und die relativen Kennzahlen. Bei den absoluten Kennzahlen ist der Informationsgehalt durch die

Aussagekraft der Zahl selbst definiert. Dies ist z.B. bei Cash-Flow und EBIT der Fall. Diese repräsentieren in der Regel feste Werte in der entsprechenden Währung, aber auch die Mitarbeiterzahl ist streng genommen eine absolute Kennzahl. Relative Kennzahlen hingegen werden normalerweise aus zwei in Beziehung zueinander stehenden Kennzahlen gebildet (häufig durch Quotientenbildung). Dadurch soll die Aussagekraft erhöht werden.
Die Verwendung von Kennzahlen geht mitunter so weit, dass sie, vorwiegend bei großen Unternehmen, sogar zur Gehaltsbestimmung der Mitarbeiter herangezogen werden. Letztere sind dann natürlich daran interessiert, den Wert einer Kennzahl zu steigern und arbeiten gegebenenfalls effizienter.

5.5. Kennzahlenauswahl und Beschreibung

Für die Experimente ist eine Kennzahlenauswahl zu treffen. Das System orientiert sich an dem von Uthoff angewandten Kennzahlensystem (siehe [U97]). Es werden außerdem Kennzahlen gebildet, die von Uthoff als besonders trennscharfe Größen identifiziert worden sind. Es muss des Weiteren berücksichtigt werden, welche Kennzahlen überhaupt auf der Grundlage der vorliegenden Daten gebildet werden können. Als Resultat ergibt sich das folgende System mit 18 Kennzahlen. Qualitative Kennzahlen, die beispielsweise durch Managementbefragungen ermittelt werden, sollen nicht Gegenstand der Analyse sein.

Nr.	Kennzahl	Block	Herkunft
1	Jahresüberschuss / Jahresfehlbetrag Tsd EUR	Erfolgskennzahl	Creditreform
2	Cash-Flow Tsd EUR	Erfolgskennzahl	Creditreform
3	Current Ratio	Liquiditätskennzahl	Creditreform
4	Eigenkapitalrentabilität (%)	Rentabilitätskennzahl	Creditreform
5	Gesamtkapitalrentabilität (%)	Rentabilitätskennzahl	Creditreform
6	Return on Investment (%)	Rentabilitätskennzahl	Creditreform
7	Verschuldungsgrad	Kennzahl zur Kapitalstruktur	Creditreform
8	Eigenkapitalquote (%)	Kennzahl zur Kapitalstruktur	Creditreform
9	Dynamischer Verschuldungsgrad	Kennzahl zur Kapitalstruktur	Creditreform
10	Working Capital Tsd EUR	Kennzahl zur Kapitalstruktur	Creditreform
11	EBIT Tsd EUR	Erfolgskennzahl	Creditreform
12	EBITDA Tsd EUR	Erfolgskennzahl	Creditreform
13	Fremdkapitalquote (%)	Kennzahl zur Kapitalstruktur	Uthoff
14	Net Working Capital (%)	Liquiditätskennzahl	Uthoff
15	Cash-Flow/Fremdkapital	Erfolgskennzahl	Uthoff
16	Working Capital/Gesamtkapital	Kennzahl zur Kapitalstruktur	Uthoff
17	Reingewinn/Gesamtkapital	Erfolgskennzahl	Uthoff
18	Cash-Flow/Kurzfristiges Fremdkapital	Erfolgskennzahl	Uthoff

Tabelle 1: Kennzahlensystem

Die verwendeten Kennzahlen sollen im Folgenden kurz beschrieben, und ihre Berechnungsweise erläutert werden. Dazu sind verschiedene Quellen (siehe [SH01], [PS07] und [H06]) herangezogen worden. Die von Uthoff (siehe [U97]) verwendeten Kennzahlen werden zusammenfassend und allgemein beschrieben.

5.5.1. Jahresüberschuss / Jahresfehlbetrag Tsd EUR

Der Jahresüberschuss ist das positive Geschäftsergebnis eines Geschäftsjahres und spiegelt den Erfolg eines Unternehmens wider. Nimmt dieser einen negativen Wert an, so liegt ein Jahresfehlbetrag vor. Der Jahresüberschuss wird in der Gewinn- und Verlustrechnung eines Unternehmens angegeben.

Jahresüberschuss / Jahresfehlbetrag
= Erträge - Aufwendungen

5.5.2. Cash-Flow

Der Cash-Flow beschreibt das periodisch erwirtschaftete Finanzvolumen, das für die Zwecke von Investitionen, Schuldentilgung und in Höhe des Gewinns zur Ausschüttung genutzt werden könnte – oder auch mit Ausnahme der Gewinnausschüttung schon genutzt wurde.
Er ist ein Maßstab für die Selbstfinanzierungskraft eines Unternehmens und zeigt die finanzielle Flexibilität und finanzielle Unabhängigkeit von außenstehenden Geldgebern an.
Der Cash-Flow zeigt also keine tatsächlich verfügbaren Finanzmittel an, da die periodisch erwirtschafteten Finanzmittel zumeist sofort wieder im Leistungsprozess eingesetzt werden.
Verliert ein Unternehmen in einer Periode Kapital, so liegt ein negativer Cash-Flow vor. Die Berechnung des Cash-Flows ist nach dem US-GAAP (Generally Accepted Accounting Principles) sowie auch nach dem deutschen HGB im Rahmen einer Quartals- oder Jahresabschlussrechnung für Aktiengesellschaften und Kommanditgesellschaften auf Aktien verbindlich vorgeschrieben.
Es wird mit dem Cash-Flow versucht, die tatsächlichen Zahlungsströme zu ermitteln, die im bilanziellen Ergebnis eines Unternehmens so nicht erkennbar sind, da sie beispielsweise Abschreibungen und Rückstellungen enthalten.

Darüber hinaus wird versucht, die folgenden Sachverhalte näher zu analysieren:

- Kann ein Unternehmen aus eigener Kraft Investitionen tätigen?
- Wieviel Geld ist für Schuldentilgung, Zinszahlung und zur Ausschüttung an Gesellschafter vorhanden?
- Ist eine Insolvenz zu erwarten? (Ein negativer Cash-Flow über mehrere Jahre führt mit hoher Wahrscheinlichkeit zu einer Insolvenz)

Der Cash-Flow soll also weniger reale Handlungsmöglichkeiten beschreiben, sondern vielmehr Beziehungen zwischen dem Erfolgs- und Finanzbereich abbilden, die auf dem in einer Periode erwirtschafteten Finanzüberschuss aufbauen.
Es lässt sich jedoch sagen, dass ein niedriger Cash-Flow zu einer Schwächung des Eigenkapitals und somit zu einem erhöhten Verschuldungsgrad führt. Daraus folgt in der Regel eine wachsende Zinslast. Das Wachstumspotential des Unternehmens wird folglich eingeschränkt.
Im Folgenden werden die Berechnungsweisen des Cash-Flows näher erläutert. Der Cash-Flow in seiner einfachsten Form ist der so genannte Brutto-Cash-Flow. Er stellt den gesamten vom Unternehmen erwirtschafteten Cash-Flow dar. Aus dem Brutto-Cash-Flow lässt sich der Netto-Cash-Flow berechnen, der eine Aussage über die Fähigkeit zur Schuldentilgung und Rücklagenbildung eines Unternehmens erlaubt, da Faktoren wie Steuern und Privatentnahmen herausgerechnet, d.h. subtrahiert werden. Zahlungswirksame Erträge, die nach der eigentlichen Bilanzierung getätigt werden, müssen hingegen addiert werden. Diese Größe wird daher auch als bereinigter Cash-Flow bezeichnet. Aus dem Netto-Cash-Flow wiederum lässt sich der Free-Cash-Flow, also der frei verfügbare Cash-Flow, ermitteln. Dieser verdeutlicht, welche Mittel für die Dividenden der Aktionäre bzw. Gesellschafter verbleiben. Dabei werden Desinvestitionen (Gegenteil von Investitionen) addiert, denn es handelt sich dabei um den Erlös verkaufter Vermögensgegenstände.

Brutto-Cash-Flow

= Jahresüberschuss / Jahresfehlbetrag
+ Abschreibungen
- Zuschreibungen
+ Zunahme der langfristigen Rückstellungen inklusive Pensionsrückstellungen und Sonderposten mit Rücklagenanteil
- Abnahme der langfristigen Rückstellungen inklusive Pensionsrückstellungen und Sonderposten mit Rücklagenanteil

Netto-Cash-Flow

= Brutto-Cash-Flow
- Steuern
- Privatentnahmen bei Personengesellschaften
- Rücklagenauflösung
+ Rücklagenzuführung

Free-Cash-Flow

= Netto-Cash-Flow
- Investitionen (aus Geschäftstätigkeit: Ersatz- und Erweiterungsinvestionen)
+ Desinvestitionen

5.5.3. Current Ratio (Liquidität 3. Grades)

Die Liquidität 3. Grades ist eine Liquiditätskennzahl, die das Verhältnis des kurzfristigen Umlaufvermögens zu den kurzfristigen Verbindlichkeiten eines Unternehmens angibt. Unter kurzfristigem Umlaufvermögen wird die Liquidierbarkeit innerhalb eines Jahres bzw. Vorräte, die durch Kundenzahlungen gedeckt werden, verstanden. Kurzfristige Verbindlichkeiten bezeichnen die Fälligkeit innerhalb eines Jahres.
Die Liquidität 3. Grades ist also eine Verhältniszahl mit Einbeziehung der Wiedergeldwerdungsdauer. Gerade das ist für ein Unternehmen besonders wichtig, denn sowohl

kurzfristiges Umlaufvermögen als auch kurzfristige Verbindlichkeiten sind in der Regel gut beeinflussbar.
Ist der Wert der Liquidität 3. Grades kleiner als 1, so ist ein Unternehmen nicht in der Lage die kurzfristigen Verbindlichkeiten (Forderungen von Lieferanten und anderen kurzfristigen Gläubigern) selbstständig zu decken. Folglich muss es gegebenenfalls Anlagevermögen verkaufen oder, sofern überhaupt möglich, langfristige Verbindlichkeiten aufbauen, um diese Deckungsbeträge zu erlangen. Die Liquidität 3. Grades steht auf die folgende Art und Weise mit dem Working Capital in Verbindung:

- Liquidität 3. Grades = 1 ⇔ Working Capital = 0
- Liquidität 3. Grades > 1 ⇔ Working Capital > 0
- Liquidität 3. Grades < 1 ⇔ Working Capital < 0

Für die Liquidität 3. Grades wird i.A. ein Prozentsatz um die 20% gefordert („Banker's Rule"). Jedoch wird eine unnötig hohe Liquidität im Regelfall zu Lasten der Rentabilität gehen.

Current Ratio (Liquidität 3. Grades)

= Kurzfristiges Umlaufvermögen / Kurzfristige Verbindlichkeiten

5.5.4. Eigenkapitalrentabilität

Die Eigenkapitalrentabilität gehört zu den Rentabilitätskennzahlen und zeigt den Grad der Verzinsung des von Kapitalgebern investierten Kapitals innerhalb einer Rechnungsperiode an. Der so genannte Leverage-Effekt kann die Finanzneutralität der Kennzahl aufheben, da durch die Aufnahme von Fremdkapital die Eigenkapitalrentabilität gesteigert werden kann, so lange die Fremdkapitalzinsen, abzüglich des in der Regel vorhandenen Tax Shield (Wert, den Fremdkapitalzinsen, Schulden oder Verlustvorträge zum Unternehmenswert beitragen), unterhalb der Gesamtkapitalrentabilität liegt.
Der Maßstab für die Eigenkapitalrentabilität kann der marktübliche Zins für ein längerfristig angelegtes Kapital sein. Der angestrebte Zielwert ist stark branchenabhängig, in den meisten Bereichen wird aber ein Wert um 20 % angestrebt.

Eigenkapitalrentabilität

= (Gewinn / Eigenkapital) x 100

5.5.5. Gesamtkapitalrentabilität

Die Gesamtkapitalrentabilität ist eine weitere Rentabilitätskennzahl. Sie ist ein Maßstab für die Effizienz des Kapitaleinsatzes eines Investorunternehmens innerhalb einer Betrachtungsperiode. Mit dieser Kennzahl lassen sich die Nachteile der Eigenkapitalrentabilität umgehen und somit auch der zuvor beschriebene Leverage-Effekt.
Zu beachten ist, dass bei der Berechnung die Fremdkapitalzinsen dem Gewinn hinzuzurechnen sind, da sie in der gleichen Periode erwirtschaftet worden sind, aber als Aufwand den Gewinn schmälern. Die Gesamtkapitalrendite ist also die gemeinsame Rendite sowohl der Eigen- als auch der Fremdkapitalgeber.

Gesamtkapitalrentabilität

= (Gesamtperiodenerfolg + Fremdkapital-Zinsen) / Gesamtkapital

5.5.6. Return on Investment

Bei Return on Investment (RoI) handelt es sich um eine Rentabilitätskennzahl. Der RoI ist eine periodische Bezugsgröße, die den finanziellen Erfolg des ganzen innerhalb eines Unternehmens gebundenen Kapitals beurteilt.
Der RoI ist die Spitzenkennzahl des DuPont-Kennzahlensystems. Dabei werden verschiedene Kennzahlen zueinander in Beziehung gesetzt, um eine höhere Aussagekraft gegenüber einzeln stehenden Kennzahlen erreichen zu können. Indirekt wird durch das Kennzahlensystem so festgelegt, wie das Unternehmen zu steuern ist.
Die Grundidee des RoI ist, dass die Umsatzrendite in Bezug zum Kapitalumschlag gesetzt wird, so dass erkannt werden kann, ob ein veränderter RoI durch eine Veränderung der Gesamtkapitalrentabilität und/oder des Kapitalumschlags verursacht wurde. Je höher der Kapitalumschlag, desto besser, und dies gilt natürlich auch für die Umsatzrentabilität. Steigt ein Wert und sinkt der andere, ergibt sich eventuell ein identischer RoI, aber beide Einflussfaktoren

können durch den RoI gesondert erfasst und beurteilt werden. Anzustreben ist in mittelständischen Unternehmen ein RoI von ca. 10 % und mehr.
Zu beachten ist, dass der RoI keine Beurteilung des Erfolgs von Einzelinvestitionen erlaubt, da die Informationen sehr stark aggregiert sind, und das Kennzahlensystem durch die Abbildung buchhalterischer Daten vergangenheitsorientiert ist und damit keine Berücksichtigung zukünftiger Entwicklungen zulässt.

Kapitalumschlag

= (Umsatz / Gesamtkapital)

Umsatzrendite

= (Gewinn / Umsatz)

Return on Investment

= Umsatzrendite x Kapitalumschlag x 100
= (Gewinn / Umsatz) x (Umsatz / Gesamtkapital) x 100

5.5.7. Verschuldungsgrad

Der Verschuldungsgrad ist eine Kennzahl zur Kapitalstruktur. Dieser setzt das Eigenkapital in Verhältnis zu dem Fremdkapital. Demnach ist ein hoher Verschuldungsgrad ein Zeichen dafür, dass das in Rede stehende Unternehmen stark von Fremdkapitalgebern abhängig ist.
Je höher der Verschuldungsgrad ist, desto schlechter werden in der Regel die Konditionen für Kredite, die das Unternehmen aufnehmen möchte, und umso fragwürdiger wird ein längerfristiger Fortbestand des Unternehmens. Als Konsequenz ergibt sich, dass ein solches Unternehmen auf lange Sicht hohe Zuflüsse erzielen muss, um eine weitere Finanzierung sicherzustellen.

Verschuldungsgrad

= (Fremdkapital / Eigenkapital) x 100

5.5.8. Eigenkapitalquote

Die Eigenkapitalquote ist eine weitere Kennzahl zur Kapitalstruktur, die das Eigenkapital in Verhältnis zu dem Gesamtkapital setzt. Bei der Berechnung muss unter Umständen noch der Abzug von Aktivpositionen bzw. die Zurechnung von Passivpositionen beachtet werden. Generell ist eine hohe Eigenkapitalquote ein Indiz dafür, dass das in Rede stehende Unternehmen über ein finanzielles Sicherheitspolster verfügt. Je höher demnach die Eigenkapitalquote eines Unternehmens ist, desto sicherer ist die Unternehmenslage, denn die Kapitalbasis ist gut genug um auch kurz- und mittelfristig Verluste verkraften zu können. Gegenwärtig ist eine Eigenkapitalquote zwischen 25% und 30 % je nach Branche zufriedenstellend. Liegt diese über 40%, so ist dies ein außerordentlich positives Zeichen.

Eigenkapitalquote

= (Eigenkapital / Gesamtkapital) x 100

5.5.9. Fremdkapitalquote

Die Fremdkapitalquote ist eine Kennzahl zur Kapitalstruktur eines Unternehmens. Diese gibt den Anteil des Fremdkapitals an dem Gesamtkapital an. Das Verhalten der Fremdkapitalquote ist analog zu der Eigenkapitalquote.

Fremdkapitalquote

= (Fremdkapital / Gesamtkapital) x 100

5.5.10. Dynamischer Verschuldungsgrad

Der dynamische Verschuldungsgrad ist eine Kennzahl zur Kapitalstruktur und eignet sich als ergänzende Kennzahl zur Beurteilung der Schuldentilgungsfähigkeit eines Unternehmens. Dieser

berechnet sich durch Division des Fremdkapitals durch den Cash-Flow des letzten Geschäftsjahres. Der dynamische Charakter der Kennzahl ergibt sich daraus, dass der Cash-Flow als zeitraumbezogene Größe einfließt. Da der Cash-Flow aufzeigt, welche Mittel ein Unternehmen in der abgelaufenen Periode durch den Umsatzprozess erwirtschaften konnte, d.h. in welcher Höhe diese Mittel zur Verfügung standen, gibt der dynamische Verschuldungsgrad somit an, in wie vielen Jahren es einem Unternehmen unter sonst gleichen Bedingungen möglich wäre, seine Effektivschulden aus dem Cash-Flow vollständig zu tilgen. Das bedeutet, je kleiner der Wert dieser Kennzahl ist, desto schneller kann ein Unternehmen Schulden aus Mitteln tilgen, die im eigenen Umsatzprozess erwirtschaftet worden sind. Ein Unternehmen mit kleinem dynamischen Verschuldungsgrad ist unabhängiger von seinen Gläubigern und gilt als finanziell stabiler.

Dynamischer Verschuldungsgrad

= (Fremdkapital / Cashflow) x 100

5.5.11. Working Capital

Das Working Capital bezeichnet den Überschuss der kurzfristig liquidierbaren Aktiva eines Unternehmens über die kurzfristigen Passiva. Unter kurzfristig wird in der Regel der Zeitraum eines Jahres verstanden. Das Working Capital ist also der Teil des Umlaufvermögens, der nicht durch kurzfristige Verbindlichkeiten verdeckt ist, und somit im Beschaffungs-, Produktions- und Absatzprozess arbeiten kann.
Obwohl das Working Capital im weiteren Sinne eine Kennzahl zur Kapitalstruktur ist, kann es aber ebenso als Maß für die Liquidität eines Unternehmens gesehen werden. Es gibt den Überhang der langfristigen Mittel über die Teile des Anlagevermögens an, die innerhalb eines Jahres liquidierbar sind.
Ferner lässt sich mit dem Working Capital erkennen, in welchem Umfang Teile des kurzfristig freisetzbaren Umlaufvermögens mittel- bis langfristig finanziert sind. Das zeigt das nicht ausgenutzte, langfristige Finanzierungsvolumen auf und ist ein generelles Maß für die Expansionskraft eines Unternehmens.

Working Capital

= Kurzfristiges Umlaufvermögen - Kurzfristige Verbindlichkeiten

5.5.12. EBIT

Der EBIT (Earnings Before Interest and Taxes) ist eine Erfolgskennzahl, die eine Aussage über den betrieblichen Gewinn eines Unternehmens innerhalb einer Periode erlaubt. Die wörtliche Übersetzung von EBIT wäre Gewinn vor Zinsen und Steuern, in der Praxis jedoch hat es die Bedeutung von Gewinn vor Finanzergebnis, außerordentlichem Ergebnis und Steuern.
Durch die Eliminierung dieser drei genannten Faktoren kann eine vergleichbarere Aussage über die eigentliche operative Ertragskraft einer Unternehmung getroffen werden und zwar unabhängig von der individuellen Kapitalstruktur.
Wird der Jahresüberschuss oder die Netto-Umsatzrendite zum Vergleich verwendet, so schneiden Unternehmen mit einer höheren Eigenkapitalquote aufgrund geringerer Fremdkapitalkosten tendenziell besser ab.

EBIT

= Gewinn
+ Steueraufwand
- Steuererträge
+ außerordentlicher Aufwand
- außerordentliche Erträge
+ Finanzaufwand
- Finanzerträge

5.5.13. EBITDA

Der EBITDA (Earnings Before Interest, Taxes, Depreciation and Amortization) ist eine weitere Erfolgskennzahl, die einen Zusammenhang zwischen dem Ertrag und dem Cash-Flow eines Unternehmens herstellt. Die wörtliche Übersetzung von EBITDA lautet: Ertrag vor Zinsen, Steuern, Abschreibungen auf Sachanlagen und Abschreibungen auf immaterielle

Vermögensgegenstände. Der EBITDA misst also den Jahresüberschuss vor Steuern, Zinsergebnis, außerordentlichem Ergebnis und vor Abschreibungen des Unternehmens.
Da international betrachtet die Gesellschaften unter unterschiedlichen Gesetzgebungen bilanzieren, ermöglicht die Kennzahl EBITDA, aufbauend auf dem EBIT, aussagekräftigere Vergleiche der operativen Ertragskraft, als sie durch den ausgewiesenen Jahresüberschuss zu erhalten wären. Der EBITDA besitzt also eine Art Bereinigungscharakter.
Beispielsweise weisen investitionsfreudige Unternehmen ergebnismindernde Abschreibungen und damit einen geringeren Jahresüberschuss im Vergleich zu weniger investitionsfreudigen Unternehmen auf.

EBITDA

= EBIT
+ Abschreibungen auf das Anlagevermögen
- Zuschreibungen zum Anlagevermögen

5.5.14. Net Working Capital

Das Net Working Capital im engeren Sinne ist das Kapital, das für ein Unternehmen Umsatz generiert, ohne dass im Gegenzug Kapitalkosten verursacht werden. Hier wird jedoch die relative Kennzahl verwendet, die denselben Namen trägt und sich aus diesem Zusammenhang direkt ableitet.
Das Net Working Capital gibt demnach den prozentualen Anteil an, zu dem das kurzfristige Fremdkapital durch das Umlaufvermögen gedeckt ist.

Net Working Capital

= (Umlaufvermögen / Kurzfristiges Fremdkapital) x 100

5.5.15 Verbleibende Uthoff Kennzahlen

Die restlichen von Uthoff verwendeten Kennzahlen sind empirisch ermittelte Werte, mit dem Ziel, die Zusammensetzung so zu optimieren, dass eine gute Trennbarkeit zwischen solventen

und später insolventen Unternehmen erzielt werden kann. Auf dieser Grundlage basiert auch die Entscheidung, zumindest einige dieser Kennzahlen in die Betrachtung einzubeziehen.
Die erste dieser Kennzahlen setzt den Cash-Flow in Beziehung zu dem Fremdkapital. Bei genauer Betrachtung stellt sich heraus, dass es sich um den reziproken Wert des dynamischen Verschuldungsgrades handelt. Die Kennzahl gibt folglich an, wie groß der Anteil der Effektivschulden ist, der durch den Umsatzprozess getilgt werden kann.
Die zweite Kennzahl wird aus dem Quotienten Working Capital und dem Gesamtkapital gebildet. Das Working Capital ist der Teil des Umlaufvermögens, der nicht zur Abdeckung der kurzfristigen Verbindlichkeiten gebunden ist. Deshalb kann in Bezug auf die Interpretation dieser Kennzahl gesagt werden, dass sie den Anteil des im Beschaffungs-, Produktions- und Absatzprozess arbeitenden Vermögens am Gesamtvermögen aufzeigt.
Die dritte Kennzahl stellt das Verhältnis von Reingewinn zu dem Gesamtkapital dar. Der Begriff Reingewinn ist eine gebräuchliche Bezeichnung für den Jahresüberschuss. Es wird hier also abgebildet, wie hoch der Anteil des unternehmerischen Erfolges gemessen an dem Gesamtkapital ist.
Die letzte dieser Kennzahlen betrifft den Zusammenhang von Cash-Flow und kurzfristigem Fremdkapital. Es handelt sich also um eine Verfeinerung der ersten, in diesem Abschnitt beschriebenen Kennzahl. Der entscheidende Unterschied ist, dass die Zeitrahmen der beiden zueinander in Beziehung gesetzten Kennzahlen anglichen werden. Die Kennzahl beschreibt, welcher Teil der zeitnah fällig werdenden Verbindlichkeiten durch den laufenden Umsatzprozess gedeckt werden kann.
Auch wenn diese Kennzahlen und deren Interpretationen eingängig erscheinen, so beruhen diese größtenteils nicht auf theoretischen Überlegungen, sondern auf empirischen Versuchen. Dies kann bei näherer Betrachtung mitunter zu Problemen führen.

5.6. Analyse der Daten

5.6.1. Kern-PCA

Zur Analyse vorab wird eine lineare KPCA für die Jahresabschlüsse der 177 insolventen und den zugehörigen 177 solventen Unternehmen wie in Kapitel 5.2 beschrieben durchgeführt, um einen ersten Eindruck des Informationsgehalts der Daten zu gewinnen. Das Verfahren der KPCA soll an dieser Stelle nicht beschrieben werden. Interessierte seien auf meine Masterarbeit (siehe [B05]) verwiesen, bei dem das Verfahren ausführlich beschrieben und auf die DNA von Mikroorganismen angewendet wird.

Der lineare Kern ist wie folgt definiert:

$$k_{linear}(\vec{x},\vec{y}) = \langle \vec{x},\vec{y} \rangle = \sum_{i=1}^{n} y_i x_i$$

Die Kernmatrix demnach:

$$K_{ij} = k_{linear}(\vec{x}_i,\vec{x}_j)$$

Die KPCA liefert dann folgendes Resultat:

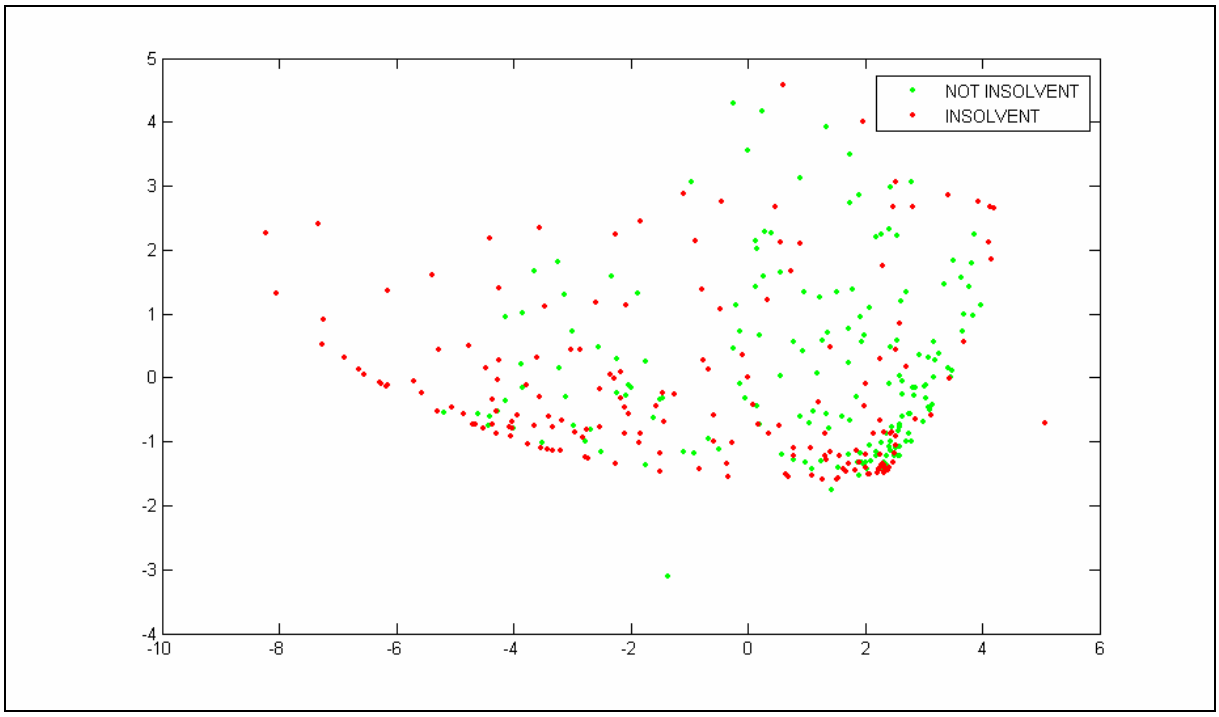


Abbildung 9: KPCA der 356 Unternehmen mit linearem Kern

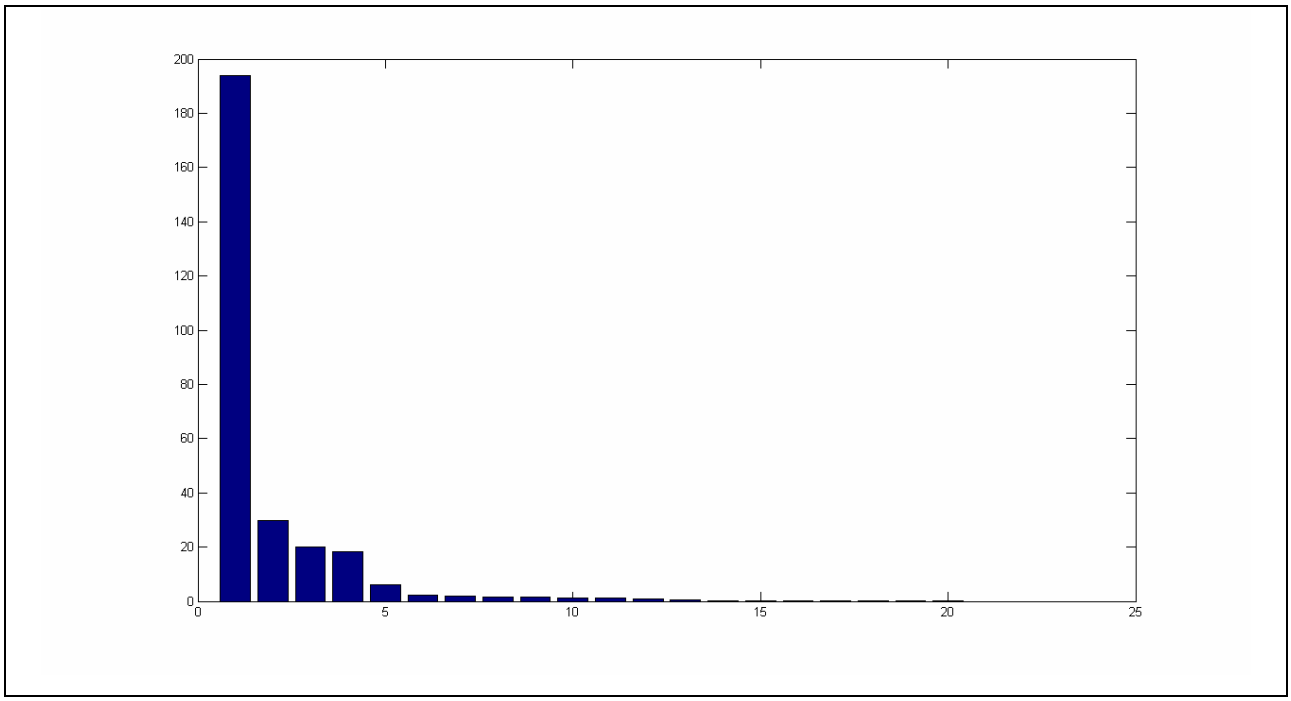


Abbildung 10: Eigenwertspektrum zur KPCA mit linearem Kern

Die KPCA deutet auf eine sehr schlechte Trennbarkeit der Daten hin, zumindest wenn die Annahme zugrunde gelegt wird, dass hohe Varianz auch hoher Informationsgehalt bedeutet. Das Eigenwertspektrum zeigt, dass bereits in der ersten Hauptkomponente ein sehr großer Teil der Gesamtvarianz abgebildet ist. Es wird in einem zweiten Schritt der Polynomkern getestet, der wie folgt definiert ist:

$$k^d_{poly}(\vec{x},\vec{y}) = \langle \vec{x},\vec{y} \rangle^d = \left(\sum_{i=1}^{n} y_i x_i \right)^d$$

Die Kernmatrix mit Standardgrad $d = 3$ demnach:

$$K_{ij} = k^3_{poly}(\vec{x}_i, \vec{x}_j)$$

Die KPCA liefert damit folgendes Resultat

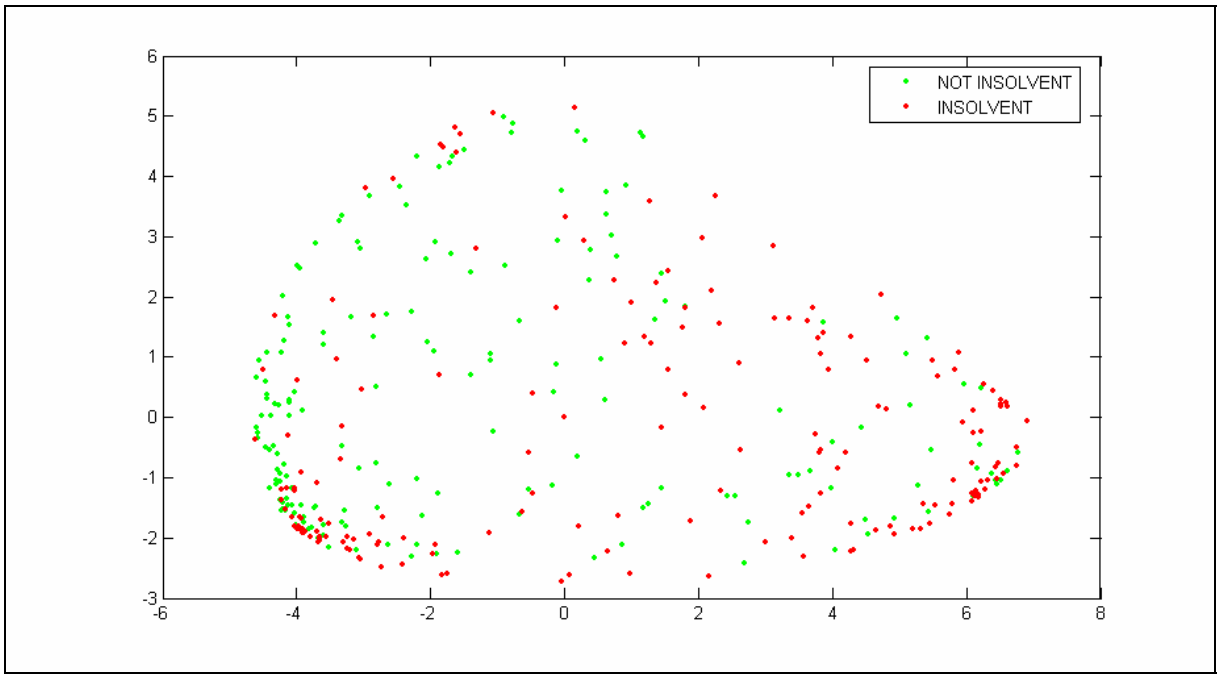


Abbildung 11: KPCA der 356 Unternehmen mit Polynomkern

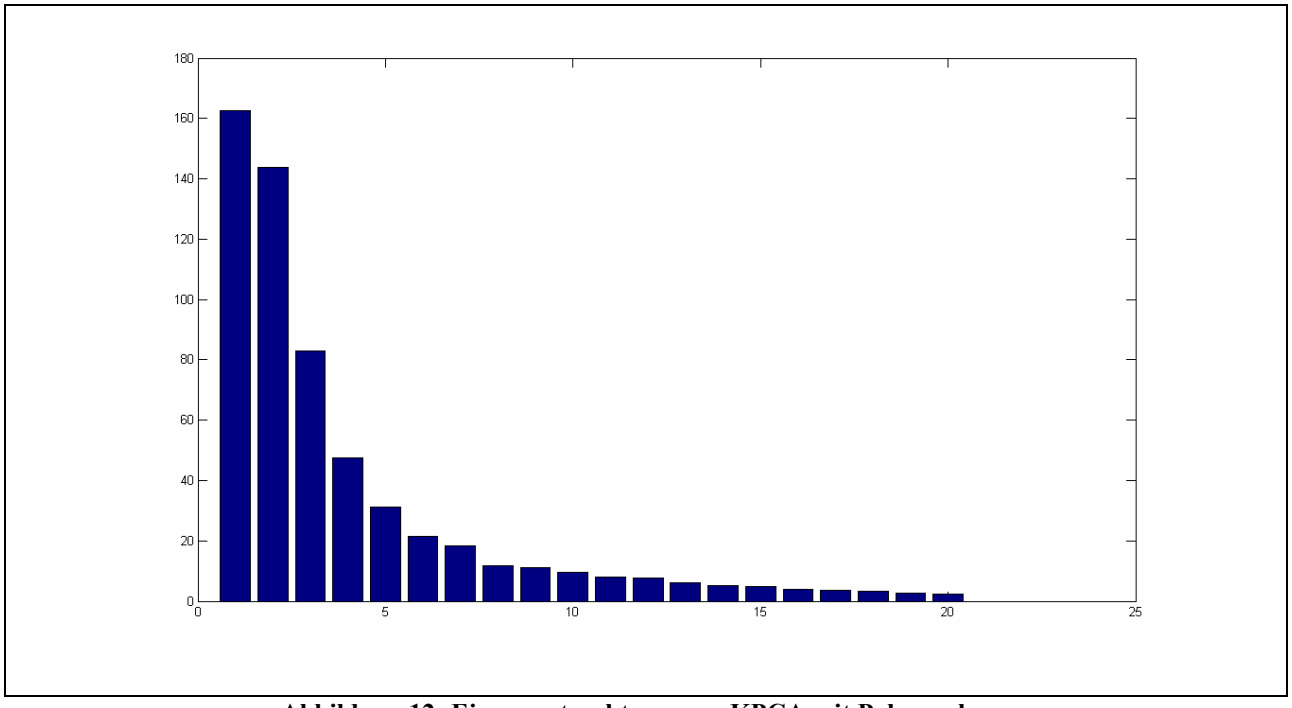


Abbildung 12: Eigenwertspektrum zur KPCA mit Polynomkern

Es zeigt sich ein vergleichbares Ergebnis wie mit dem linearen Kern. Die insolventen Unternehmen scheinen sich in den Randbereichen etwas stärker zu gruppieren, jedoch liegt eine starke Überlagerung in der Mitte des Plots vor, was dieselbe Schlussfolgerung wie bei dem linearen Kern zulässt. Das Eigenwertspektrum fällt weniger schnell ab als im Falle des linearen

Kerns, was auf eine geringere Abbildung der Gesamtvarianz in den ersten beiden Hauptkomponenten hindeutet. Schließlich wird der Radialbasiskern angewendet. Dieser ist definiert als

$$k_{RBF}^{\gamma}(\vec{x},\vec{y}) = \exp\left(-\gamma\left\|\vec{x}-\vec{y}\right\|^2\right)$$

und die Kernmatrix mit Standardwert $\gamma = 1$ demnach:

$$K_{ij} = k_{poly}^{3}(\vec{x}_i,\vec{x}_j)$$

Die KPCA liefert damit folgendes Resultat

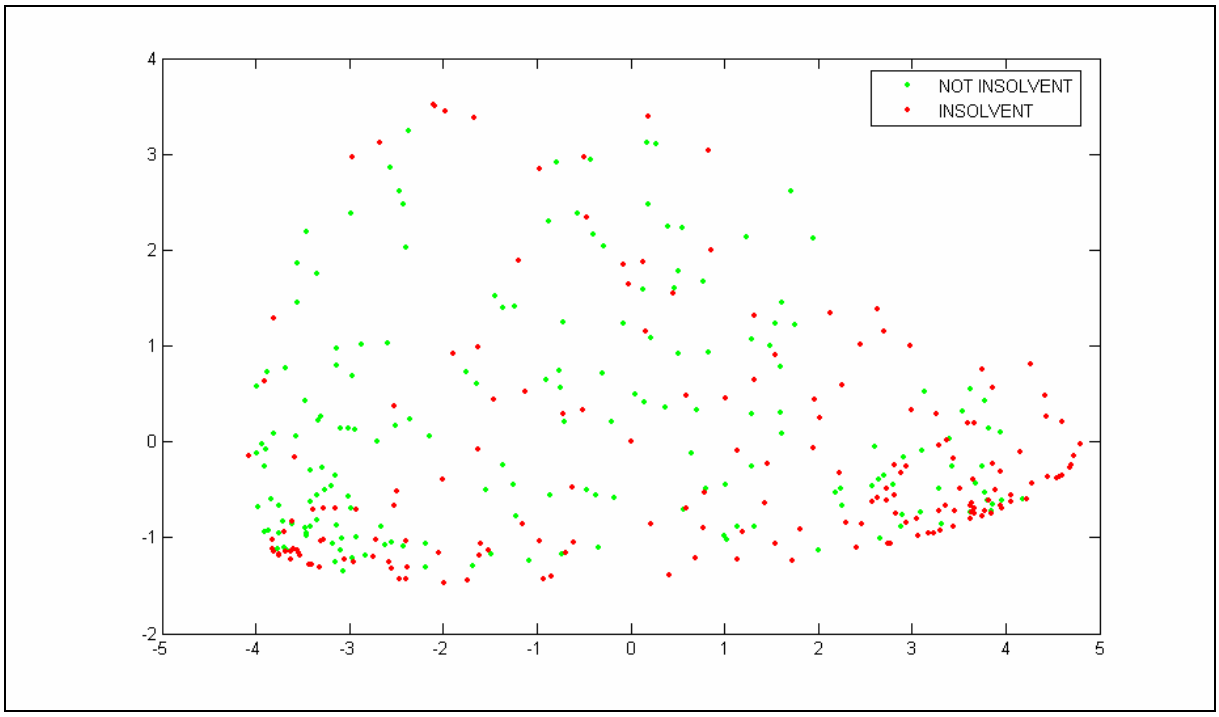


Abbildung 13: KPCA der 356 Unternehmen mit Radialbasiskern

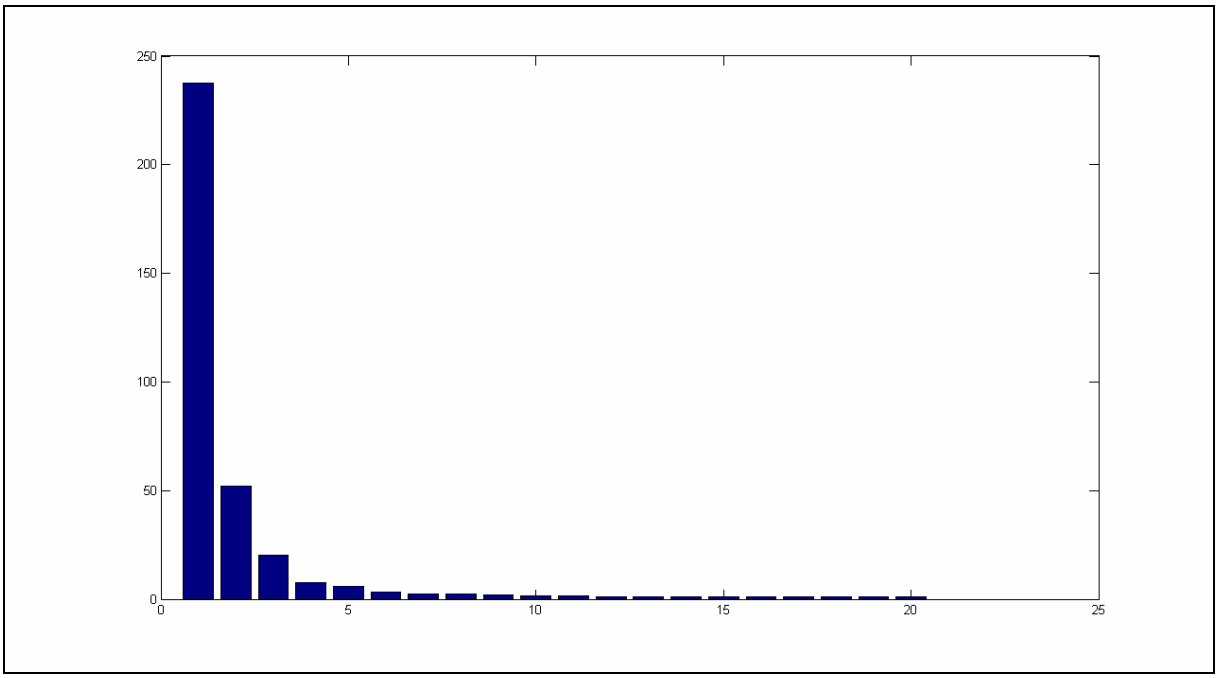


Abbildung 14: Eigenwertspektrum zur KPCA mit Radialbasiskern

Es scheint als sei die starke Überlagerung der Wolken in der Mitte des Plots etwas geringer, jedoch ist keine zufrieden stellende Trennung der Klassen erkennbar, denn auch bei diesem Kern fällt das Eigenwertspektrum stark ab, was wieder ein Zeichen für eine sehr gute Abbildung der Gesamtvarianz im zugehörigen Plot ist. Trotz Wahl unterschiedlicher Kerne hat sich das Ergebnis nicht stark verändert. Dies bestätigt einmal mehr, dass Strukturen, die in den Daten nicht vorhanden sind, durch kein noch so gutes Verfahren gefunden werden können. Die Vorab-Analyse gibt Anlass, die Ergebnisse und die Jahresabschlussanalyse in diesem Zusammenhang kritisch zu bewerten, insbesondere da reale Daten vorliegen und keine Labels, die anhand der Daten berechnet worden sind, nachgelernt werden. Dies ist aber im Falle des VC-Index in der Arbeit von Uthoff (siehe [U97]) der Fall.
Allerdings muss auch beachtet werden, dass ein kernbasiertes Verfahren lediglich durch die Wahl der Kernfunktion eingegriffen werden kann, was bei dieser Analyse nicht erfolgt ist. Es sind lediglich universelle Kerne angewendet worden, die sich unter Umständen weniger gut für die Darstellung eignen.

5.7. Parameter der Verfahren

In diesem Kapitel sollen die angewandten Varianten der Verfahren und die verwendeten Parameter kurz beschrieben werden.

5.7.1. Thresholder

Die Ansätze des PAC-Lernen mit Rauschen auf der Klassifikation werden hier unter dem Begriff „Thresholder-Verfahren“ zusammengefasst.
Für das Thresholder-Verfahren wird die in Kapitel 4.5 vorgestellte Heuristik in der Differenz-Advantage-Variante zur Konzeptklasse $C_{\hat{p},r}^{DNF}$ verwendet. Die Gesamtzahl der Merkmale einer Hypothese wird auf drei beschränkt, indem die maximale Anzahl von Literalen pro Monom auf $\hat{p}=1$, und die maximale Anzahl der Monome in einer disjunktiven Normalform auf $\hat{r}=3$ beschränkt wird. Auf diese Weise wird außerdem der Fehler 1. Art kleiner, was im Sinne der Insolvenzvorhersage ist. Um weitere Hyperparameter zu eliminieren, wird der Gewichtungsfaktor $\omega=1$ gewählt.

5.7.2. Multivariate Diskriminanzanalyse

Für die multivariate Diskriminanzanalyse werden verschiedene Distanzmaße getestet. Die Verwendung der Mahalanobis-Distanz führte auf diesem Datensatz zu einer Verschlechterung, weshalb hier die euklidische Distanz verwendet wird. Diese hat sich als deutlich robuster gegenüber inhomogenen Stichproben erwiesen.

5.7.3. Nearest-Neighbor

Für das Nearest-Neighbor-Verfahren werden die Werte für *k = 1, 3, 5* getestet. Die besten Ergebnisse konnten mit dem Wert *k = 5* erzielt werden.

5.7.4. Neuronales Netz

Ein neuronales Netz ist anhand der Vorgaben von Uthoff (siehe [U97]) erstellt worden, ohne die Einbeziehung qualitativer Merkmale, da in dieser Arbeit lediglich quantitative Merkmale betrachtet werden. Als Lernregel ist der Gradientenabstieg verwendet worden, sowie ein Standardnetz mit einer verdeckten Schicht und zwei inneren Neuronen, ähnlich wie das BP-14 (siehe [BKU96]). Als Lernparameter werden ein Momentum von 0.5 und eine Lernschrittweite von 0.0001 verwendet. Diese Parameter sind vorab durch praktische Tests bestimmt worden.

5.7.5. Support Vector Machine (SVM)

Für die SVM wird eine 5-fach-Kreuzvalidierung zur Schätzung der Parameter C und γ durchgeführt. Es werden der Linearkern, Polynomkern sowie der Radialbasiskern angewendet. Für den Polynomkern wird der Grad auf $d = 3$ festgelegt.

5.8. Ergebnisse und Vergleich

5.8.1. Das Experiment und Vergleich der Verfahren

Um die verschiedenen Verfahren zu validieren, wird die Jacknive-Methode, auch bekannt als „Leave-one-out" – Methode, verwendet. Dieses Verfahren ist zwar rechenintensiv, da für jedes Beispiel der Trainingsstichprobe ein Klassifikator trainiert wird, stellt aber eine maximale Trainingsmenge zur Verfügung, was aufgrund der spärlichen Datenlage eine angemessene Alternative darstellt. Weiterhin werden die Merkmale auf das Intervall [-1, +1] skaliert, um insbesondere bei euklidischen Distanzen der MDA eine Gleichgewichtung der Merkmale zu gewährleisten.

Aufgrund der eingeschränkten Datenlage ist es nicht möglich, auf verschiedenen Stichproben zu validieren. Dies bringt das statistische Problem mit sich, dass zwar einzelne Experimente eine zuverlässige Aussagekraft besitzen, jedoch nicht alle zugleich. Dieses Problem lässt sich jedoch aufgrund der Datenlage nicht umgehen.

Die erzielten Klassifikationsraten der verschiedenen Verfahren, die Trennrate (Vorhersage der Trainingsstichprobe) sowie die verwendeten Vervollständigungsstrategien können der folgenden Tabelle entnommen werden:

Verfahren	Klassifikationsrate	Fehler 1. Art	Fehler 2. Art	Missing Values	Trennrate
Thresh	74,9%	20,9%	29,4%	Zero	77,6%
MDA	64,1%	18,6%	53,1%	Zero	65,6%
SVM (Linear)	64,7%	44,6%	26,0%	Zero	65,3%
SVM (RBF)	67,2%	36,2%	29,4%	Zero	71,3%
SVM (Poly)	59,6%	24,9%	55,9%	Zero	64,9%
Neuronales Netz	62,7%	46,9%	27,7%	Zero	64,7%
5–Nearest Neighbor	67,8%	36,2%	28,2%	Zero	-

Tabelle 2: Verfahrensvergleich mit Missing Value Strategie ‚Zero'

Es zeigt sich, dass das Thresholder-Verfahren eine deutlich höhere Klassifikationsrate erreicht, sowie einen akzeptablen Fehler 1. Art unter der Verwendung von drei Merkmalen. Dies wird insbesondere im Vergleich zur SVM deutlich, bei der keine Möglichkeit zur Skalierung des

Fehlers 1. Art besteht. Das Nearest-Neighbor-Verfahren schneidet auch mit einer guten Klassifikationsrate ab, jedoch ebenso mit einem hohen Fehler 1. Art. Neuronale Netze schneiden in dieser Konstellation lediglich durchschnittlich ab, sind aber nicht anfällig für Overfitting-Probleme. Dies ist jedoch auf die Beschränkung der Neuronen der inneren Schicht zurückzuführen.

Verfahren	Klassifikationsrate	Fehler 1. Art	Fehler 2. Art	Missing Values	Trennrate
Thresh	78,0%	10,7%	33,3%	Mean	80,4%
MDA	64,4%	18,6%	53,1%	Mean	65,8%
SVM (Linear)	65,0%	44,6%	25,4%	Mean	65,4%
SVM (RBF)	67,2%	35,6%	29,9%	Mean	71,2%
SVM (Poly)	57,6%	27,7%	57,1%	Mean	60,7%
Neuronales Netz	63,0%	45,8%	28,2%	Mean	64,4%
5–Nearest Neighbor	67,2%	36,2%	29,4%	Mean	-

Tabelle 3: Verfahrensvergleich mit Missing Value Strategie ‚Mean'

Unter Verwendung der Vervollständigungsstrategie ‚Mean' wird der Vorsprung des Thresholder-Verfahrens noch deutlicher. Der Fehler 1. Art liegt bei knapp 11% mit einem Fehler 2. Art von 33,3%. Für die verwendeten Merkmale scheint der Erwartungswert daher ein akzeptabler Schätzer zu sein. Im Vergleich ist diese überaus hohe Klassifikationsrate jedoch ein Ausnahmefall und muss in Relation zu den sonstigen Klassifikationsraten des Thresholder-Verfahrens gesetzt werden.

Verfahren	Klassifikationsrate	Fehler 1. Art	Fehler 2. Art	Missing Values	Trennrate
Thresh	72,0%	18,1%	37,9%	Median	75,8%
MDA	64,1%	18,6%	53,1%	Median	65,6%
SVM (Linear)	64,7%	44,6%	26,0%	Median	65,4%
SVM (RBF)	67,8%	34,5%	29,9%	Median	71,3%
SVM (Poly)	57,6%	28,2%	56,5%	Median	59,8%
Neuronales Netz	61,9%	47,5%	28,8%	Median	64,7%
5–Nearest Neighbor	66,7%	35,6%	31,1%	Median	-

Tabelle 4: Verfahrensvergleich mit Missing Value Strategie ‚Median'

In diesem Fall liegt die Rate des Thresholder-Verfahrens etwas niedriger, ebenso die der anderen Verfahren, was den Verdacht erhärtet, dass der Median für eine angemessene Schätzung aufgrund der Größe der Stichprobe weniger geeignet ist. Die anderen Verfahren erweisen sich robust gegenüber der gewählten Vervollständigungsstrategie.

Das Thresholder-Verfahren liefert in allen Varianten deutlich höhere Klassifikationsraten. Dies ist zum einen darauf zurückzuführen, dass Wissen über die Natur der Kennzahlen abgebildet worden ist, was einen Vorteil gegenüber universellen Verfahren bietet. Darüber hinaus liegt enormes Potential in der Beschränkung auf wenige, trennscharfe Merkmale in einer Hypothese, da auf diese Weise störende Merkmale nicht zur Vorhersage herangezogen werden.
Die folgenden Tabellen zeigen, dass bereits bei 3 verwendeten Merkmalen mit Instabilitäten bezüglich der Vorhersage auf ungesehenen Daten zu rechnen ist.
Außerdem werden die Auswirkungen verschiedener Parameter des Thresholder-Verfahrens bei unterschiedlichen Vervollständigungsstrategien dokumentiert.
Dies ändert sich bei einer besseren Datenlage (siehe Kapitel 6).

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
1	1	1,0	71,8%	33,9%	22,6	Zero	71,9%
1	1	1,0	70,3%	26,0%	33,3	Median	70,7%
1	1	1,0	69,2%	41,8%	19,8	Mean	69,4%
1	1	1,2	68,1%	32,2%	31,6	Zero	71,4%
1	1	1,2	68,1%	26,0%	37,9	Median	70,3%
1	1	1,2	66,7%	27,7%	39,0	Mean	69,3%

Tabelle 5: Thresholder Klassifikationsraten bei einem Merkmal

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
2	1	1,0	73,7%	35,6%	16,9	Zero	73,5%
2	1	1,0	72,9%	28,2%	26,0	Median	72,8%
2	1	1,0	68,9%	42,9%	19,2	Mean	70,2%
2	1	1,2	69,5%	36,2%	24,9	Zero	73,0%
2	1	1,2	67,5%	28,2%	36,7	Median	71,6%
2	1	1,2	64,7%	31,6%	39,0	Mean	69,9%
1	2	1,0	75,7%	20,9%	27,7%	Zero	76,9%
1	2	1,0	72,3%	18,1%	37,3%	Median	75,3%
1	2	1,0	75,4%	26,0%	23,2%	Mean	77,1%
1	2	1,2	73,4%	19,2%	33,9%	Zero	77,1%
1	2	1,2	70,3%	15,8%	43,5%	Median	75,0%
1	2	1,2	72,0%	15,8%	40,1%	Mean	74,8%

Tabelle 6: Thresholder Klassifikationsraten bei zwei Merkmalen

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
1	3	1,0	74,9%	20,9%	29,4%	Zero	77,6%
1	3	1,0	72,0%	18,1%	37,9%	Median	75,8%
1	3	1,0	78,0%	10,7%	33,3%	Mean	80,4%
1	3	1,2	72,9%	19,2%	35,0%	Zero	77,8%
1	3	1,2	69,8%	15,8%	44,6%	Median	75,4%
1	3	1,2	73,4%	7,3%	45,8%	Mean	76,8%
3	1	1,0	74,3%	36,2%	15,3	Zero	74,0%
3	1	1,0	71,8%	30,5%	26,0	Median	74,0%
3	1	1,0	68,6%	44,1%	18,6	Mean	70,4%
3	1	1,2	70,6%	36,2%	22,6	Zero	74,0%
3	1	1,2	66,7%	30,5%	36,2	Median	72,3%
3	1	1,2	64,7%	31,6%	39,0	Mean	70,2%

Tabelle 7: Thresholder Klassifikationsraten bei drei Merkmalen

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
2	2	1,0	71,2%	30,5%	27,1%	Zero	79,4%
2	2	1,0	74,6%	20,3%	30,5%	Median	77,6%
2	2	1,0	75,7%	27,7%	20,9%	Mean	78,2%
2	2	1,2	71,8%	24,9%	31,6%	Zero	78,9%
2	2	1,2	71,2%	15,8%	41,8%	Median	76,7%
2	2	1,2	69,8%	20,3%	40,1%	Mean	75,3%

Tabelle 8: Thresholder Klassifikationsraten bei vier Merkmalen

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
3	2	1,0	74,6%	28,2%	22,6%	Zero	79,7%
3	2	1,0	73,4%	22,6%	30,5%	Median	78,9%
3	2	1,0	75,4%	28,8%	20,3%	Mean	78,6%
3	2	1,2	75,1%	23,7%	26,0%	Zero	79,4%
3	2	1,2	71,2%	18,1%	39,5%	Median	77,3%
3	2	1,2	69,5%	20,9%	40,1%	Mean	75,5%
2	3	1,0	72,9%	27,1%	27,1%	Zero	80,3%
2	3	1,0	75,7%	18,1%	30,5%	Median	78,6%
2	3	1,0	75,1%	26,0%	23,7%	Mean	80,6%
2	3	1,2	72,0%	24,3%	31,6%	Zero	79,7%
2	3	1,2	71,2%	14,1%	43,5%	Median	77,1%
2	3	1,2	71,8%	12,4%	44,1%	Mean	78,9%

Tabelle 9: Thresholder Klassifikationsraten bei sechs Merkmalen

$\hat{p}$	$\hat{r}$	ω	Klassifikation	Alpha	Beta	Missing Values	Trennrate
3	3	1,0	74,6%	27,7%	23,2%	Zero	81,0%
3	3	1,0	74,9%	19,8%	30,5%	Median	80,0%
3	3	1,0	74,9%	27,1%	23,2%	Mean	81,1%
3	3	1,2	75,4%	23,2%	26,0%	Zero	80,7%
3	3	1,2	71,5%	15,8%	41,2%	Median	77,8%
3	3	1,2	70,6%	16,9%	41,8%	Mean	79,4%

Tabelle 10: Thresholder Klassifikationsraten bei neun Merkmalen

Die grau markierten Einträge dokumentieren die Verringerung des Fehlers 1. Art bei steigender Alternativenzahl. Dies ist leicht nachvollziehbar, da jedes zusätzlich gewählte Merkmal zu einer größeren Menge von positiv klassifizierten Beispielen führt, da es alleinig in einer Alternative auftaucht. Dadurch bieten drei Merkmale anschaulich drei „Chancen“, das Label „Insolvent“ auszugeben. Umgekehrt lässt sich bei steigender Zahl von Literalen pro Alternative erkennen, dass der Fehler 2. Art sinkt, da der gegenteilige Fall eintritt: Die „Chance“, das Label „Insolvent“ auszugeben, erfordert drei erfüllende Merkmale.

Im Falle von drei Alternativen, erreicht der Fehler 1. Art einen Wert von weniger als 11%. Durch die Gewichtung mit dem Faktor $\omega = 1{,}2$ lässt sich dieser Fehler sogar auf 7,3% verringern, bei geringfügig höherem Fehler 2. Art.

Bei Betrachtung der Abweichungen von Trainingsfehler und Vorhersagefehler über all diese Konfigurationen, kann eine durchschnittliche Abweichung von 4% ermittelt werden. Diese Abweichung deutet bereits auf ein leichtes Overfitting hin, weshalb die letzte Tabelle die absolut größte Konzeptklasse bei dieser Datenlage darstellen sollte. Insbesondere sind bei der Wahl von neun Merkmalen vereinzelte Abweichungen von fast 10% zu erkennen. Im Idealfall sollten sich Trainingsfehler und Vorhersagefehler nicht unterscheiden. Auch die bestimmte Rademacher-Komplexität für dieses Szenario (siehe Kapitel 4.8) deutet auf dieses Problem hin.

Der Gewichtungsfaktor ω führt zu einem höheren Gewinn bei der Klassifikation positiver Beispiele, wodurch der Fehler 1. Art weiter verringert werden kann. Dies ist jedoch nur bis zu einem gewissen Grad möglich. In der Praxis hat sich ein Wert von $\omega = 1{,}2$ als maximal erwiesen. Um diesen Hyperparameter zu eliminieren, kann er jedoch auf 1 gesetzt werden, so wie es in den Tabellen 1 bis 3 erfolgt ist.

5.8.2. Interpretation der erlernten Hypothesen

In diesem Kapitel sollen die erlernten Hypothesen dargestellt und interpretiert werden. Da für die Vervollständigungsstrategie „Mean“ gute Ergebnisse erzielt werden konnten, werden lediglich Hypothesen dieser Form betrachtet. Aus dem ursprünglich vorgestellten Kennzahlensystem sind die folgenden sechs Kennzahlen Teil der Hypothesen:

Nr.	Kennzahl	Abkürzung	Herkunft
1	Current Ratio	CURRA	Creditreform
2	EBIT Tsd EUR	EBIT	Creditreform
3	Working Capital Tsd EUR	WC	Creditreform
4	Working Capital/Gesamtkapital	WCGSK	Uthoff
5	Reingewinn/Gesamtkapital	RGGSK	Uthoff
6	Fremdkapitalquote (%)	FRKQ	Uthoff

Die Schwellwertfunktionen sind, wie zuvor definiert, gegeben durch:

$$\mathrm{T}_{i,\tau}^{above}(\vec{x}) = \begin{cases} +1 & falls \quad x_i > \tau \\ -1 & sonst \end{cases} \qquad \mathrm{T}_{i,\tau}^{below}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \leq \tau \\ -1 & sonst \end{cases}$$

Zur besseren Übersicht wird im Folgenden anstelle des Merkmalsindex $i \in \{1,...,n\}$ die Abkürzung der jeweiligen Kennzahl aus der obigen Tabelle eingetragen.

Die folgenden Hypothesen sind für $\omega = 1{,}0$ unter Verwendung der Schwellwertfunktionen T^{above} und T^{below} erlernt worden:

$\hat{p}$	$\hat{r}$	**Erlernte Hypothese (Heuristik)**
1	1	$\left(T^{below}_{RGGSK,0.0061}\right)$
1	2	$\left(T^{below}_{RGGSK,0.0061}\right) \vee \left(T^{above}_{CURRA,17.4150}\right)$
1	3	$\left(T^{below}_{RGGSK,0.0061}\right) \vee \left(T^{above}_{CURRA,17.415}\right) \vee \left(T^{below}_{WCGSK,0.1636}\right)$
2	1	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000}\right)$
2	2	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000}\right) \vee \left(T^{above}_{CURRA,17.4150} \wedge T^{below}_{EBIT,1021.0000}\right)$
2	3	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000}\right) \vee \left(T^{above}_{CURRA,17.4150} \wedge T^{below}_{EBIT,1021.0000}\right) \vee \left(T^{above}_{FRKQ,99.9282} \wedge T^{below}_{RGGSK,0.1810}\right)$
3	1	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000} \wedge T^{below}_{WC,181200.50}\right)$
3	2	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000} \wedge T^{below}_{WC,181200.50}\right) \vee \left(T^{above}_{CURRA,17.4150} \wedge T^{below}_{EBIT,1021.0000}\right)$
3	3	$\left(T^{below}_{RGGSK,0.0061} \wedge T^{above}_{CURRA,0.4000} \wedge T^{below}_{WC,181200.50}\right) \vee \left(T^{above}_{CURRA,17.4150} \wedge T^{below}_{EBIT,1021.0000}\right) \vee \left(T^{above}_{FRKQ,99.9282} \wedge T^{below}_{RGGSK,0.1810}\right)$

Tabelle 11: Erlernte Hypothesen für verschiedene Parameter (Heuristik)

Es ist erkennbar, dass selbst bei der Beschränkung auf neun Kennzahlen pro Hypothese durch die Parameter $\hat{p} = 3$ und $\hat{r} = 3$ lediglich fünf Kennzahlen aufgenommen werden.

Auffällig ist auch, dass die Kennzahlen *RGGSK* bzw. *CURRA* in jeder Alternative auftauchen. Zudem sind stets die von Uthoff als trennscharf identifizierte Kennzahlen (*RGGSK* bzw. *FRKQ),* Teil der Hypothese.

Es existiert außerdem eine obere Grenze für den *EBIT* von etwa 1 Million Euro. Liegt ein Unternehmen unterhalb dieses Wertes, so ist dies ein Zeichen für bevorstehende Insolvenz.

Widersprüchlich erscheint, dass die Kennzahl *CURRA* als Mindestwert, also mit dem Schwellwertfunktionstyp *above* vorkommt. So ist ein Mindestwert von etwa 17 ein Indikator für

bevorstehende Insolvenz, obwohl hohe Werte dieser Kennzahl gesunden Unternehmen zuzuordnen sind. Jedoch ist diese Kennzahl in Korrelation mit anderen Kennzahlen derselben Alternative zu bewerten und kommt außerdem mit einem sehr kleinen Schwellwert im Falle der ersten Alternative vor.
Weiterhin ist auffällig, dass keine Cash-Flow-Kennzahlen Teil der Hypothese sind. Obwohl diese als trennscharf gelten, scheinen die gewählten Kennzahlen erstere vollständig abzudecken.

Außerdem lässt sich festhalten, dass nur wenige Kennzahlen ausreichend sind, um eine gute Klassifikationsleistung zu erzielen. Demnach erscheint es sinnvoll, kleine Kennzahlensysteme zu verwenden.

5.8.3. Fazit, Ausblick und kritische Bewertung

Es hat sich gezeigt, dass der entwickelte Klassifikator sehr gute Ergebnisse erzielt, insbesondere auf dem vorliegenden, inhomogenen Datensatz. Der Ansatz einer Schwellwertbildung scheint vernünftig und wird auch durch die Kennzahlenhintergründe (siehe Kapitel 5.4 und 5.5) untermauert. Die Vorgehensweise bei dem Kendall-Verfahren ist ähnlich, jedoch werden keinerlei Korrelationen zwischen Merkmalen erfasst, was dem Thresholder-Verfahren einen klaren Vorteil im Bereich der Trennschärfe verschafft, sowie eine Resistenz gegenüber Ausreißern. Die praktische Erfahrung hat gezeigt, dass die Quotienten-Advantage-Variante im Schnitt deutlich schlechter abschneidet, was aber vor allem auf die Stichprobengröße zurückzuführen ist, durch die Schätzungen das Ergebnis verzerren können.

Im Rahmen der Arbeit sind auch Resampling-Methoden zur Verbesserung der Datenlage angewendet worden, jedoch mit einem schlechten Ergebnis. Es ist der Eindruck entstanden, dass die Schätzungen von Kovarianzmatrizen sehr ungenau sind und sie deshalb bei dieser Datenlage mit Vorsicht zu gebrauchen sind.

Es ist wahrscheinlich, dass im Falle homogenerer Daten (beispielsweise durch Branchengruppierung) die Verfahren bessere Klassifikationsraten liefern. Nach den Erfahrungen im Rahmen dieser Arbeit ist jedoch über offizielle Kanäle kein besseres Datenmaterial zu beschaffen. Außerdem konnte nach Korrespondenz mit mehreren Kreditinstituten der Eindruck gewonnen werden, dass zusätzliche Daten von Anbietern wie beispielsweise dem Verband der Vereine Creditreform e.V. erworben werden müssen, da das eigene Datenmaterial häufig nicht ausreichend ist.

Banken und Kreditinstitute mit qualitativ hochwertigen Daten wiederum werden diese unter Verschluss halten, um den Vorteil gegenüber der Konkurrenz zu sichern. Daher ist eine wissenschaftliche Analyse nicht möglich.
Diese Arbeit bleibt somit im Vergleich zu anderen Arbeiten zur Insolvenzvorhersage überprüfbar, da Algorithmen und verwendete Daten (gegen Gebühr bei der Creditreform e.V.) öffentlich zugänglich sind.

Ein weiterer Kritikpunkt an anderen Arbeiten zur Insolvenzvorhersage liegt darin, dass Bewertungsverfahren nachgelernt werden, insbesondere in der Arbeit von Uthoff in [U97], werden Labels anhand der vorliegenden Daten erzeugt. Zum einen durch den VC-Index, der einer Summe von gewichteten Kennzahlen entspricht, zum anderen dem BP-14, das ebenfalls anhand der Daten eine Klassifikation liefert. Es ist leicht nachvollziehbar, dass auf diese Weise höhere Klassifikationsraten erzielt werden, da der Zusammenhang von Label und Daten definitiv gegeben ist. Die tatsächliche Praxisrelevanz, und die Frage, ob der Zusammenhang zwischen Merkmalen und Label bei realen Daten tatsächlich gegeben ist, bleiben hier jedoch offen.

Es ist geplant, das Thresholder-Verfahren auf einen Multiklassenfall zu erweitern, so dass eine Einordnung von Unternehmen in Rating-Klassen möglich wird. Es existieren bereits Ideen zu diesem Ansatz, jedoch sollen sie nicht Thema dieser Arbeit sein.

6. Anwendung auf das Problem der Hotspot-Identifikation

6.1. Einführung

Proteine bestehen aus Ketten von Aminosäuren, die über Peptidbindungen miteinander verknüpft sind (Primärstruktur). Diese Ketten falten sich im Raum aufgrund verschiedener Wechselwirkungen zwischen Aminosäuren zu einer 3D-Struktur, die erst dann in der Lage ist, eine bestimmte Funktionalität zu entwickeln (siehe [S03]). In zahlreichen Fällen gliedern sich Proteine in einzelne Untereinheiten, die zu einem größeren Proteinkomplex aggregieren.
Ein aktuelles Forschungsgebiet im Bereich der Biologie ist daher die Untersuchung von Protein-Protein-Wechselwirkungen, in dem untersucht wird, welche Proteine mit anderen interagieren und wie diese Interaktion zustande kommt. Die Kontaktflächen, an denen zwei Proteine eine Bindung ausbilden, werden Interface genannt.
Eine der gegenwärtigen Theorien ist die Hotspot-Theorie (siehe [W08] und [ABKT98]), die besagt, dass innerhalb eines Interfaces nur sehr wenige Aminosäuren an der Ausbildung einer festen Bindung beteiligt sind. Diese Aminosäuren werden demnach Hotspots genannt. Es erscheint einleuchtend, dass für eine solche Interaktion eine räumliche Nähe erforderlich ist und folglich Wissen über die 3D-Struktur dieser Proteine einfließen muss. Die Identifikation dieser Hotspots soll im Folgenden mit einem sehr ähnlichen Ansatz wie dem zur Insolvenzerkennung behandelt werden.

6.2. Datenlage und Aufbereitung

Die Grundlage der Analyse bilden die Daten der PDB Datenbank (siehe [PDB])
Es werden 1836 Homo- und Heterodimere, basierend auf einer Liste von Ruth Nussinov verwendet (siehe [W08]). Da später eine gleiche Anzahl negativer Beispiele generiert werden soll, gilt $m = 2 \cdot 1836 = 3672$.
Über die vorliegende 3D-Struktur der PDB-Dateien ist es möglich, die räumliche Nähe einzelner Aminosäuren mit Hilfe der BALL-Bibliothek (siehe [BALL]) zu bestimmen.
Es wird angenommen, dass räumliche Nähe eine Voraussetzung für Interaktion im Interface ist. Daher ist ein möglicher Ansatz, dicht beieinander liegende Aminosäurenpaare von interagierenden Ketten als zum Interface gehörig anzusehen (siehe [ZGK06]).

Das Alphabet der Aminosäuren ist wie folgt definiert:

$$\Sigma^{AS} = \left\{ \begin{matrix} GLY, ASP, ASN, LEU, TYR, TRP, ILE, ALA, VAL, PRO, \\ ARG, CYS, SER, PHE, HIS, MET, THR, GLN, GLU, LYS \end{matrix} \right\}$$

Es werden weiterhin die folgenden Datenstrukturen mit den jeweiligen Attributen benötigt:

Atom

- X (X-Koordinate im Raum)
- Y (Y-Koordinate im Raum)
- Z (Z-Koordinate im Raum)

Residue

- *Acid* (Name dieser Aminosäure aus dem Alphabet Σ^{AS})
- *Atoms* (Menge der zugehörigen Elemente vom Typ *Atom*)

Chain

- *Residues* (Menge der zugehörigen Elemente vom Typ *Residuum*)

Mit dem folgenden Algorithmus können interagierende Aminosäuren-Paare bestimmt werden (siehe [W08]):

Algorithmus 10 Pairwise Atom Distance (PAD)

Eingabe: Zwei interagierende Ketten vom Typ *Chain* c_1, c_2 und ein Schwellwert d^{thresh}

Ausgabe: Die Mengen I^{single}, I^{pair}, der zum Interface gehörenden Aminosäuren bzw. Paare

$I^{single} \leftarrow \{\ \}$

$I^{pair} \leftarrow \{\ \}$

Für jedes *Residue* $r_1 \in c_1.residues$ führe aus:

Für jedes *Atom* $a_1 \in r_1.atoms$ führe aus:

Für jedes *Residue* $r_2 \in c_2.residues$ führe aus:

Für jedes *Atom* $a_2 \in r_2.atoms$ führe aus:

$$d_{euc} = \sqrt{(a_1.x - a_2.x)^2 + (a_1.y - a_2.y)^2 + (a_1.z - a_2.z)^2}$$

Falls $d_{euc} - (VanDerWaalsRadius(a_1) + VanDerWaalsRadius(a_2)) \leq d^{thresh}$ so führe aus:

$I^{pair} \leftarrow I^{pair} \cup \{r_1, r_2\}$

$I^{single} \leftarrow I^{single} \cup \{r_1\} \cup \{r_2\}$

Return (I^{single}, I^{pair})

Das Interface eines Proteins wird dann durch eine Distanz von $d^{thresh} = 0.5$ Å charakterisiert. Zu jedem Protein *i* für $i = 1,...,1836$ sind weiterhin die interagierenden Ketten c_{i1} und c_{i2} aus der Nussinov-Liste bekannt. Es ergeben sich die folgenden Interface-Mengen:

$$(I_i^{single}, I_i^{pair}) \leftarrow PAD(c_{i1}, c_{i2}, 0.5) \text{ für } i = 1,..., m/2$$

Weiterhin können mit Hilfe der BALL-Bibliothek Aminosäuren bestimmt werden, die an der Oberfläche eines Proteins lokalisiert sind. Dies erfolgt über ein iteratives Verfahren, das vollständig in der BALL-Bibliothek implementiert ist. Dies bietet die Möglichkeit, zufällige

Oberflächenpaare zu generieren und diese als negative Beispiele zu verwenden. Dies geschieht durch die Schätzung einer Häufigkeitsverteilung anhand der positiven Beispiele und anschließendem Ziehen zufälliger Aminosäuren bzw. Aminosäurenpaare aus der Oberflächenverteilung. Die analog zum Algorithmus PAD entstehenden Oberflächen-Mengen seien definiert als:

$$\left(O_i^{\text{sin}\,gle}, O_i^{pair}\right) \text{ für } i = m/2 + 1, \ldots, m$$

Ziel ist es, trennscharfe Merkmale zu bestimmen und diese zu bewerten. Für diese Zwecke kann das Thresholder-Verfahren ohne weitere Anpassung verwendet werden.
Die Merkmalsmengen sind wie folgt gegeben:

$$A^{\text{sin}\,gle} = \left\{\{a\} \mid a \in \Sigma^{AS}\right\}$$
$$A^{pair} = \left\{\{a_1, a_2\} \mid a_1, a_2 \in \Sigma^{AS}\right\}$$

Seien weiterhin

$$acid^{\text{sin}\,gle} : \{1, \ldots, 20\} \to A^{\text{sin}\,gle}$$
$$acid^{pair} : \{1, \ldots, 210\} \to A^{pair}$$

bijektive Funktionen, die einer natürlichen Zahl eine Aminosäure bzw. ein Aminosäurenpaar zuordnen. Anschließend können die folgenden Typen von Merkmalen für $i = 1, \ldots, m/2$ gebildet werden:

SingleByPairRelative

$$x_{ij} = \frac{\#\left\{\{r_1, r_2\} \in I_i^{pair} \mid \{r_1.acid\} = acid^{\text{sin}\,gle}(j) \vee \{r_2.acid\} = acid^{\text{sin}\,gle}(j)\right\}}{2 \cdot \# I_i^{pair}} + \frac{\#\left\{\{r_1, r_2\} \in I_i^{pair} \mid r_1.acid = r_2.acid = acid^{\text{sin}\,gle}(j)\right\}}{2 \cdot \# I_i^{pair}} \quad \text{für } j = 1, \ldots, 20$$

SingleRelative

$$x_{ij} = \frac{\#\left\{\{r\} \in I_i^{\sin gle} \mid \{r.acid\} = acid^{\sin gle}(j)\right\}}{\#I_i^{\sin gle}} \text{ für } j = 1,...,20$$

PairRelative

$$x_{ij} = \frac{\#\left\{\{r_1, r_2\} \in I_i^{pair} \mid \{r_1.acid, r_2.acid\} = acid^{pair}(j)\right\}}{\#I_i^{pair}} \text{ für } j = 1,...,210$$

Die Merkmale aus den Oberflächenmengen werden analog gebildet. Dazu wird lediglich I_i^{pair} durch O_i^{pair} und $I_i^{\sin gle}$ durch $O_i^{\sin gle}$ ersetzt für $i = m/2 + 1,...,m$.

Die so gebildeten Merkmale können dann durch zufällige Ziehung auf eine Lernstichprobe der Länge *m* abgebildet werden. Dazu fungieren die Merkmale aus den Interfacemengen als Positivbeispiele und die Merkmale aus den Oberflächenmengen als Negativbeispiele.

6.3. Ergebnisse und Evaluierung

Basierend auf der Hotspot-Datenbank ASEdb [ASEDB], liegt die folgende Verteilung für Hotspots vor:

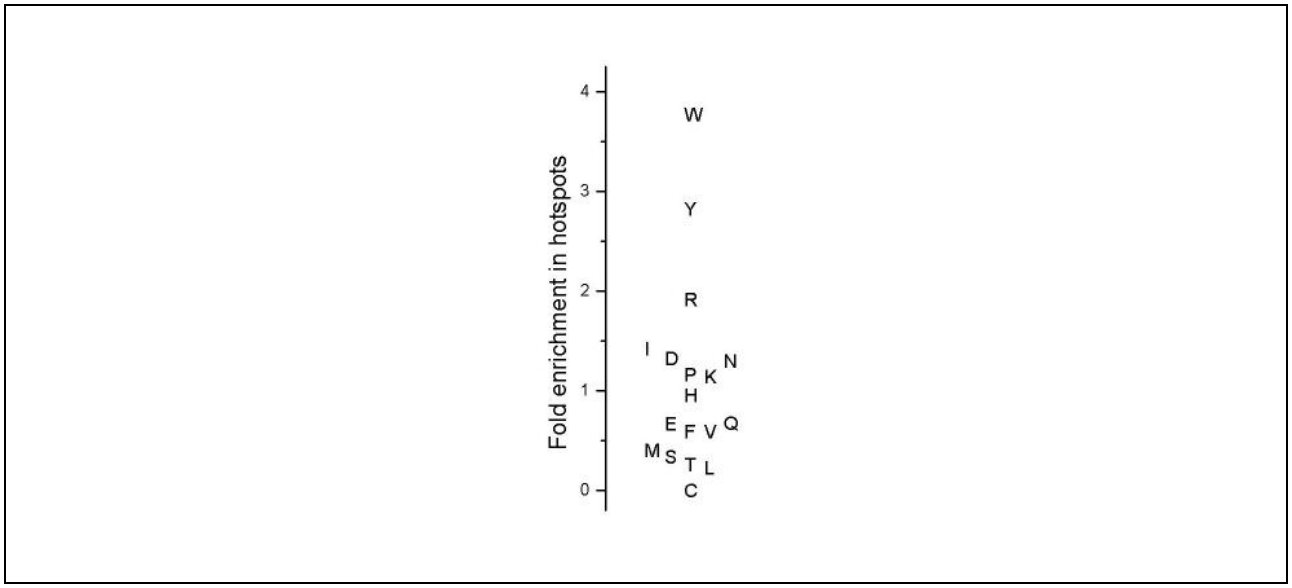


Abbildung 15: Hotspot-Verteilung nach ASEdb Hotspot-Datenbank

Entsprechend ist es zu erwarten, dass *TRP* (*W*), *TYR* (*Y*), *ARG* (*R*) und gegebenenfalls noch *ILE* (*I*) bevorzugt in Interfaces vorkommen und sich dementsprechend auch für das Diskriminationsexperiment eignen. Das Thresholder-Verfahren wird mit Standardparametern initialisiert, das heißt es ist $\omega = 1$. Für die Konzeptklasse $C^{DNF}_{\hat{p},\hat{r}}$ wird die Anzahl von Monomen auf $\hat{r} = 7$ gesetzt, da praktische Tests gezeigt haben, dass die Klassifikationsleistung an diesem Punkt nicht merklich verbessert werden kann und gemäß der Problemstellung wenige Merkmale (Hotspots) für den Klassifikator herangezogen werden sollen. Die Anzahl der Merkmale pro Monom wird dementsprechend auf $\hat{p} = 2$ gesetzt. Für das Training durch Statistical Queries (siehe Kapitel 4.7) gilt weiterhin $\varepsilon = 0{,}01$, $\eta = 0{,}4$ und $\tau = 0{,}1$. Diese Parameter haben sich in der Praxis bewährt. Die Schwellwertfunktionstypen bzw. Intervallfunktionstypen sind, wie zuvor definiert, gegeben durch:

$$\mathrm{T}^{above}_{i,\tau}(\vec{x}) = \begin{cases} +1 & falls \quad x_i > \tau \\ -1 & sonst \end{cases} \qquad \mathrm{T}^{below}_{i,\tau}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \leq \tau \\ -1 & sonst \end{cases}$$

$$\mathrm{I}^{above}_{i,[a,b]}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \in [a,b] \\ -1 & sonst \end{cases} \qquad \mathrm{I}^{below}_{i,[a,b]}(\vec{x}) = \begin{cases} +1 & falls \quad x_i \notin [a,b] \\ -1 & sonst \end{cases}$$

Zur besseren Übersicht entspricht die Variable *i* nicht dem Index des jeweiligen Merkmals, sondern der Abkürzung für eine Aminosäure bzw. ein Aminosäurenpaar. Im Falle der Aminosäure Asparagin ergibt sich beispielsweise:

$$\mathrm{T}^{above}_{ASN,\tau}(\vec{x}) = \begin{cases} +1 & falls \quad x_{acid^{\sin gle^{-1}}(\{ASN\})} > \tau \\ -1 & sonst \end{cases} \qquad \mathrm{T}^{below}_{ASN,\tau}(\vec{x}) = \begin{cases} +1 & falls \quad x_{acid^{\sin gle^{-1}}(\{ASN\})} \leq \tau \\ -1 & sonst \end{cases}$$

Es ergeben sich die folgenden Ergebnisse für den Merkmalstyp *SingleRel*:

Zulässige Funktionen	Trenn-rate	Erlernte Hypothese (Heuristik)
I^{above}	84,2%	$\left(I^{above}_{GLU,[0.0,0.067]} \wedge I^{above}_{ASP,[0.0,0.241]}\right) \vee \left(I^{above}_{ALA,[0.0,0.01]} \wedge I^{above}_{SER,[0.0,0.206]}\right) \vee \left(I^{above}_{PHE,[0.074,0.37]} \wedge I^{above}_{GLY,[0.0,0.147]}\right) \vee$ $\left(I^{above}_{TYR,[0.085,0.235]} \wedge I^{above}_{LYS,[0.0,0.206]}\right) \vee \left(I^{above}_{MET,[0.064,0.435]}\right) \vee \left(I^{above}_{VAL,[0.108,0.357]} \wedge I^{above}_{PRO,[0.0,0.097]}\right) \vee \left(I^{above}_{LEU,[0.145,0.528]} \wedge I^{above}_{ARG,[0.025,0.286]}\right)$
T^{above}, T^{below}	84,2%	$\left(T^{below}_{GLU,0.0675} \wedge T^{below}_{ASP,0.2455}\right) \vee \left(T^{below}_{ALA,0.0104} \wedge T^{below}_{SER,0.2070}\right) \vee \left(T^{above}_{PHE,0.0725} \wedge T^{below}_{GLY,0.1475}\right) \vee$ $\left(T^{above}_{TYR,0.0845} \wedge T^{below}_{LYS,0.2070}\right) \vee \left(T^{above}_{VAL,0.1075} \wedge T^{below}_{PRO,0.0975}\right) \vee \left(T^{above}_{MET,0.0635}\right) \vee \left(T^{above}_{LEU,0.1440} \wedge T^{above}_{ARG,0.0250}\right)$
T^{below}	81,9%	$\left(T^{below}_{GLU,0.0675} \wedge T^{below}_{ASP,0.2455}\right) \vee \left(T^{below}_{ALA,0.0104} \wedge T^{below}_{SER,0.207}\right) \vee \left(T^{below}_{ASN,0.0030} \wedge T^{below}_{LYS,0.17}\right) \vee$ $\left(T^{below}_{GLN,0.0030} \wedge T^{below}_{GLY,0.0635}\right) \vee \left(T^{above}_{PRO,0.0040} \wedge T^{above}_{ILE,0.0305}\right) \vee \left(T^{above}_{ARG,0.0115} \wedge T^{above}_{TRP,0.0255}\right) \vee \left(T^{below}_{THR,0.0040} \wedge T^{below}_{PHE,0.0215}\right)$
T^{above}	84,6%	$\left(T^{above}_{PHE,0.0525}\right) \vee \left(T^{above}_{TYR,0.0725}\right) \vee \left(T^{above}_{VAL,0.0885}\right) \vee \left(T^{above}_{TRP,0.0455}\right) \vee \left(T^{above}_{MET,0.0635}\right) \vee \left(T^{above}_{THR,0.1365}\right) \vee \left(T^{above}_{LEU,0.1465}\right)$

Tabelle 12: Hypothesen für Merkmalstyp SingleRel (Heuristik)

Es zeigt sich, dass eine Trennrate von über 84% erreicht wird, sowohl unter Verwendung der Intervallfunktion als auch der Schwellwertfunktion. Die erlernten Kombinationen sind auffällig einfach im Falle der Schwellwertfunktion T^{above}. Diese charakterisiert ein Interface durch eine Schwelle, die die minimal notwendige Menge einer Aminosäure repräsentiert. Es zeigt sich, dass knapp 7% Tyrosin und 5% Tryptophan erforderlich sind, was sich mit den Häufigkeiten der ASEdb deckt. Ferner sind knapp 14% Threonin, sowie 15% Leucin erforderlich, die jedoch nach ASEdb eine untergeordnete Rolle spielen. Es ist daher fraglich, ob ein großes Vorkommen von Leucin im Interface auf dessen Neutralität zurückzuführen ist, oder andere Faktoren dafür relevant sind.

Bei Betrachtung der Schwellwertfunktion I^{above} ist auffällig, dass die Mengen für Tyrosin zwischen 8% und 24% liegen sollten. Ebenfalls ist Arginin mit bis zu 28% beteiligt, Tryptophan wiederum fehlt in der Auswahl. Leucin hingegen belegt mit bis zu 50% den ersten Platz. Aufgrund der höheren Komplexität dieser und der verbleibenden Hypothesen scheint die Charakterisierung der Interfaces durch die Schwellwertfunktion T^{above} am sinnvollsten. Dies entspricht auch der intuitiven Überlegung, dass Mindestmengen von bestimmten Aminosäuren im Interface erforderlich sind.

Zulässige Funktionen	Trenn-rate	Erlernte Hypothese (Statistical Queries)
T^{above}, T^{below}	85,8%	$\left(T^{above}_{PHE,0.0640} \wedge T^{below}_{GLY,0.1420}\right) \vee \left(T^{above}_{TYR,0.0790}\right) \vee \left(\wedge T^{below}_{ASP,0.1230} T^{above}_{VAL,0.1000}\right) \vee \left(T^{below}_{LYS,0.0320} \wedge T^{above}_{THR,0.0120}\right) \vee \left(T^{below}_{GLN,0.0000} \wedge T^{above}_{ILE,0.0280}\right) \vee \left(T^{above}_{MET,0.0410} \wedge T^{below}_{GLU,0.1060}\right) \vee \left(T^{below}_{PRO,0.1620} \wedge T^{above}_{LEU,0.1480}\right)$
T^{below}	82,6%	$\left(T^{below}_{ASP,0.3130} \wedge T^{below}_{GLU,0.0600}\right) \vee \left(T^{below}_{Gly,0.3750} \wedge T^{below}_{ALA,0.0140}\right) \vee \left(T^{below}_{LYS,0.0440}\right) \vee \left(T^{below}_{GLN,0.0090} \wedge T^{below}_{VAL,0.1980}\right) \vee \left(T^{above}_{ARG,0.1410} \wedge T^{above}_{ASN,0.0000}\right)$
T^{above}	86,2%	$\left(T^{above}_{PHE,0.0670}\right) \vee \left(T^{above}_{TYR,0.0770}\right) \vee \left(T^{above}_{VAL,0.1030}\right) \vee \left(T^{above}_{MET,0.0550}\right) \vee \left(T^{above}_{THR,0.014} \wedge T^{above}_{LEU,0.1290}\right) \vee \left(T^{above}_{TRP,0.0550}\right) \vee \left(T^{above}_{PRO,0.0090} \wedge T^{above}_{ILE,0.0880}\right)$

Tabelle 13: Hypothesen für Merkmalstyp SingleRel (Statistical Queries)

Die erlernten Hypothesen unter Verwendung des Lernalgorithmus mit Statistical Queries (siehe Kapitel 4.7) liefern noch höhere Trennraten. Diese Ergebnisse zeigen, dass das Modell des PAC-Lernens mit Rauschen auf der Klassifikation und Statistical Queries in der Praxis gut geeignet ist. Die Zusammensetzung der erlernten Hypothesen weichen lediglich leicht von denen der Heuristik ab.

Für den Merkmalstyp *SingleByPairRel* werden folgende Ergebnisse erzielt:

Funktion	Trenn-rate	Erlernte Hypothese (Heuristik)
I^{above}	83,8%	$\left(I^{above}_{LYS,[0.0,0.063]} \wedge I^{above}_{ARG,[0.0,0.417]}\right) \vee \left(I^{above}_{GLU,[0.0,0.054]} \wedge I^{above}_{GLY,[0.0,0.214]}\right) \vee \left(I^{above}_{ALA,[0.0,0.0]} \wedge I^{above}_{VAL,[0.0,0.161]}\right) \vee \left(I^{above}_{PRO,[0.0,0.0]} \wedge I^{above}_{THR,[0.0,0.13]}\right) \vee \left(I^{above}_{PHE,[0.09,0.357]}\right) \vee \left(I^{above}_{TYR,[0.095,0.357} \wedge I^{above}_{LEU,[0.025,0.294]}\right) \vee \left(I^{above}_{ASP,[0.0,0.018]} \wedge I^{above}_{SER,[0.0,0.102]}\right)$
T^{above}, T^{below}	83,8%	$\left(T^{below}_{LYS,0.0635} \wedge T^{below}_{ARG,0.4585}\right) \vee \left(T^{below}_{GLU,0.0545} \wedge T^{below}_{GLY,0.2165}\right) \vee \left(T^{below}_{ALA,0.0025} \wedge T^{below}_{VAL,0.1615}\right) \vee \left(T^{below}_{PRO,0.0030} \wedge T^{below}_{THR,0.1305}\right) \vee \left(T^{above}_{PHE,0.0905}\right) \vee \left(T^{above}_{TYR,0.0945} \wedge T^{above}_{LEU,0.0255}\right) \vee \left(T^{below}_{ASP,0.0185} \wedge T^{below}_{SER,0.1025}\right)$
T^{below}	83,2%	$\left(T^{below}_{LYS,0.0635} \wedge T^{below}_{ARG,0.4585}\right) \vee \left(T^{below}_{GLU,0.0545} \wedge T^{below}_{GLY,0.2165}\right) \vee \left(T^{below}_{ALA,0.0025} \wedge T^{below}_{VAL,0.1615}\right) \vee \left(T^{below}_{PRO,0.0030} \wedge T^{below}_{THR,0.1305}\right) \vee \left(T^{below}_{ASP,0.0185} \wedge T^{below}_{SER,0.1025}\right) \vee \left(T^{below}_{GLN,0.0025} \wedge T^{below}_{TRP,0.0355}\right) \vee \left(T^{below}_{ASN,0.0020} \wedge T^{below}_{ILE,0.0615}\right)$
T^{above}	83,4%	$\left(T^{above}_{PHE,0.0475}\right) \vee \left(T^{above}_{TYR,0.0705}\right) \vee \left(T^{above}_{LEU,0.1195}\right) \vee \left(T^{above}_{ARG,0.1340}\right) \vee \left(T^{above}_{MET,0.0545}\right) \vee \left(T^{above}_{ILE,0.0905}\right) \vee \left(T^{above}_{PRO,0.1310}\right)$

Tabelle 14: Hypothesen für Merkmalstyp SingleByPairRel (Heuristik)

Dieses Experiment zeigt ein sehr ähnliches Ergebnis. Bei Betrachtung der Schwellwertfunktion T^{above} fällt auf, dass anstelle von Tryptophan mindestens 13% Arginin im Interface vorkommen sollte. Dies deckt sich erneut mit der ASEdb. Phenylalanin und Leucin erweisen sich ebenfalls als konstant, wohingegen andere Aminosäuren variieren. Die hier verwendeten Merkmale

unterscheiden sich lediglich durch Mehrfachzählungen im Bezug auf multiple Interaktionspartner.

Zulässige Funktionen	Trenn-rate	Erlernte Hypothese (Statistical Queries)
T^{above}, T^{below}	85,1%	$\left(T^{below}_{LYS,0.0680} \wedge T^{below}_{GLU,0.0770}\right) \vee \left(T^{above}_{PHE,0.0620} \wedge T^{below}_{ASP,0.2350}\right) \vee \left(T^{below}_{GLY,0.227} \wedge T^{above}_{TYR,0.0770}\right) \vee \left(T^{below}_{CYS,0.1540} \wedge T^{above}_{LEU,0.1140}\right) \vee \left(T^{above}_{MET,0.0490}\right) \vee \left(T^{above}_{ARG,0.1350} \wedge T^{below}_{ASN,0.0550}\right) \vee \left(T^{below}_{SER,0.070} \wedge T^{above}_{VAL,0.0930}\right)$
T^{below}	82,4%	$\left(T^{below}_{LYS,0.0560} \wedge T^{below}_{GLY,0.3330}\right) \vee \left(T^{below}_{GLU,0.0550}\right) \vee \left(T^{below}_{THR,0.2350} \wedge T^{below}_{ASP,0.0330}\right) \vee \left(T^{below}_{SER,0.0170} \wedge T^{above}_{ALA,0.1230}\right)$
T^{above}	85,0%	$\left(T^{above}_{PHE,0.0640}\right) \vee \left(T^{above}_{LEU,0.1160}\right) \vee \left(T^{above}_{TYR,0.0840}\right) \vee \left(T^{above}_{ARG,0.1330} \wedge T^{above}_{ILE,0.0130}\right) \vee \left(T^{above}_{MET,0.0510}\right) \vee \left(T^{above}_{TRP,0.0610}\right) \vee \left(T^{above}_{CYS,0.0300} \wedge T^{above}_{VAL,0.0490}\right)$

Tabelle 15: Hypothesen für Merkmalstyp SingleByPairRel (Statistical Queries)

Auch in diesem Fall ist eine höhere Trennrate im Falle der Schwellwertfunktion T^{above} zu verzeichnen. Die Zusammensetzung der erlernten Hypothese ist im Wesentlichen gleich.
Für den Merkmalstyp *PairRel* werden für $r = 10$ folgende Ergebnisse erzielt:

Funktion	Trennrate	Erlernte Hypothese (Heuristik)
T^{above}, T^{below}	72,1%	$\left(T^{above}_{LEU\text{-}LEU,0.0035}\right) \vee \left(T^{above}_{ARG\text{-}ASP,0.0425}\right) \vee \left(T^{above}_{LEU\text{-}PHE,0.0365}\right) \vee \left(T^{above}_{PHE\text{-}PHE,0.0265}\right) \vee \left(T^{above}_{ILE\text{-}ILE,0.0295}\right) \vee \left(T^{above}_{PHE\text{-}TYR,0.0285}\right) \vee \left(T^{above}_{ILE\text{-}VAL,0.0335}\right) \vee \left(T^{above}_{MET\text{-}MET,0.0040}\right) \vee \left(T^{above}_{LEU\text{-}TYR,0.0425}\right) \vee \left(T^{above}_{VAL\text{-}VAL,0.0425}\right)$
T^{below}	68,1%	$\left(T^{below}_{ASP\text{-}GLU,0.0125} \wedge T^{below}_{ARG\text{-}LYS,0.0115}\right)$
T^{above}	72,1%	$\left(T^{above}_{LEU\text{-}LEU,0.0035}\right) \vee \left(T^{above}_{ARG\text{-}ASP,0.0425}\right) \vee \left(T^{above}_{LEU\text{-}PHE,0.0365}\right) \vee \left(T^{above}_{PHE\text{-}PHE,0.0265}\right) \vee \left(T^{above}_{ILE\text{-}ILE,0.0295}\right) \vee \left(T^{above}_{PHE\text{-}TYR,0.0285}\right) \vee \left(T^{above}_{ILE\text{-}VAL,0.0335}\right) \vee \left(T^{above}_{MET\text{-}MET,0.0040}\right) \vee \left(T^{above}_{LEU\text{-}TYR,0.0425}\right) \vee \left(T^{above}_{VAL\text{-}VAL,0.0425}\right)$

Tabelle 16: Hypothesen für Merkmalstyp PairRel (Heuristik)

Für diesen Merkmalstyp werden deutlich schlechtere Klassifikationsraten erreicht. Diese Beobachtung ist jedoch nicht überraschend, da die Information in den Daten über 210 Merkmale verteilt ist, jedoch nur die ersten 10 Merkmale und somit lediglich 4% der Gesamtinformation zur Klassifikation herangezogen werden. Wird der Parameter r erhöht, so werden geringfügig bessere Klassifikationsraten erreicht. Bei dieser Betrachtungsweise sind die Klassifikationsraten als verhältnismäßig hoch einzustufen. Außerdem zeigt sich, dass der Schwellwertfunktionstyp *above* deutlich bessere Raten und kürzere Alternativen liefert.

Bei Betrachtung der erlernten Hypothesen lassen sich dennoch Ähnlichkeiten zu den Merkmalstypen *SingleRel* und *SingleByPairRel* erkennen.
Der Lernalgorithmus durch Statistical Queries kann auf diese Datenlage nicht angewendet werden, da aufgrund der hohen Merkmalszahl eine effiziente Berechnung nicht mehr möglich ist. Da die Heuristik aber in den vorherigen Experimenten vergleichbare Ergebnisse geliefert hat, sind auch in diesem Fall keine größeren Abweichungen zu erwarten.

Für alle erlernten Hypothesen ist ebenfalls eine Validierung gemäß der Jacknive-Methode durchgeführt worden. Die Generalisierung ist wie erwartet sehr gut, so dass die Abweichung von Trennrate und Klassifikationsrate ungesehener Beispiele im Mittel unter 0,2% liegt. Dieses Ergebnis ist für die Betrachtung der erlernten Hypothesen zur Hotspot-Identifikation nicht relevant, aber sichert den Ansatz insofern ab, als eine gute Generalisierungsfähigkeit vorliegt und somit keine Artefakte gelernt worden sind.
Für alle Merkmalstypen ist ebenfalls die Trennrate der SVM mit Radialbasiskern bestimmt worden, die lediglich drei Prozentpunkte höher, also bei etwa 89%, liegt. Dieses Ergebnis dokumentiert die Konkurrenzfähigkeit des PAC-Lernens gegenüber der SVM.
Im Falle des Merkmalstyps *PairRel* liegt die SVM bei einer Trennrate von 94,0% und einer Vorhersagerate auf ungesehenen Beispielen von 84,3%. Dies bestätigt zudem die Erwartung, dass bei hoher Merkmalszahl die Generalisierungsfähigkeit stark nachlässt und bereits 72,1% Trennrate und Vorhersagerate mit nur zehn Merkmalen erreicht werden können.

6.4. Abschließende Betrachtung

Die Ergebnisse zeigen eine klare Tendenz, jedoch ist der Ansatz für die Identifikation von Hotspots noch nicht ausreichend. Dabei ist zu berücksichtigen, dass bei diesem Ansatz nicht auf verifizierten Hotspot-Daten gelernt worden ist, sondern in umgekehrter Weise vorgegangen wird. Es wird versucht anhand von potentiellen Hotspot-Kandidaten über die Trennbarkeit einen Rückschluss auf diese vorzunehmen, um so eine Tendenz zu erkennen. Der Ansatz zeigt jedoch klar den hohen Informationsgehalt und einen deutlichen Unterschied zwischen Oberflächenverteilung und Interfaceverteilung.
Abschließend zeigt sich auch die gute Eignung der gewählten Konzeptklasse zur Trennung der Daten, die nur geringfügig schlechter als die der SVM ist.

7. Das Tool InSolve

Im Rahmen dieser Arbeit ist das wissenschaftliche Tool InSolve entwickelt worden, das eine Analyse von Jahresabschlussdaten erlaubt. Dieses Tool ist nicht für das Massengeschäft gedacht, da aufgrund der sehr problematischen Datenlage keine exakte Aussage über die tatsächliche Klassifikationsleistung möglich ist.
InSolve stellt unter anderem die vorgestellten Verfahren als Plugin bereit und erlaubt die Implementierung eigener Plugins in Java. Auf diese Weise ist es möglich, die in dieser Arbeit gewonnen Ergebnisse und Erkenntnisse zu überprüfen, aber vor allem auch Tests auf anderen Daten zu ermöglichen. Außerdem werden verschiedene Funktionen zur Visualisierung und Evaluierung angeboten, sowie Möglichkeiten zur einfachen Erstellung von Kennzahlen aus Bilanzdaten. Auch wenn der Schwerpunkt des Tools auf der Insolvenzerkennung liegt, so ist es auch für die Analyse beliebiger anderer Daten geeignet, sofern diese im CSV-Format vorliegen. Derzeit hat sich dieses Format als ausreichend erwiesen; es existieren aber entsprechende Schnittstellen, um auch die Verarbeitung anderer Eingabedateien zu ermöglichen. Die Dokumentation zu dem Tool InSolve befindet sich im Anhang.

8. Zusammenfassende Betrachtung und Fazit

Diese Arbeit hat demonstriert, dass die Verfahren des PAC-Lernens gut für die Anwendung in der Praxis geeignet sind. Außerdem ist erkennbar, dass in der Beschränkung des Lernzieles und damit im Einfluss von externem Wissen ein großes Potential liegt, denn diese Informationen können in angemessener Weise in eine Konzeptklasse eingearbeitet werden, um bessere Resultate zu erzielen. Außerdem bleiben die Parameter eines erlernten Klassifikators nachvollziehbar und damit interpretierbar. Dies steht im Gegensatz zu universellen Verfahren wie SVMs oder neuronalen Netzen.

Weiterhin zeigt sich, dass auch das Modell des PAC-Lernens mit Rauschen auf der Klassifikation in der Praxis brauchbar ist, da Lernalgorithmen durch Statistical Queries bessere Hypothesen finden konnten als einfache Heuristiken.

Bei guter Datenlage (Hotspot-Identifikation) bleiben die Verfahren des PAC-Lernens in den erzielten Trennraten konkurrenzfähig zu SVMs, wenngleich sie geringfügig schlechter abschneiden. Die Generalisierungsfähigkeit ist jedoch bei beiden Verfahren gemäß der Trennrate gut.

Die Anwendung der Verfahren des PAC-Lernens auf Jahresabschlussdaten hat sich als verhältnismäßig instabil erwiesen, was jedoch auf die schlechte Datenlage zurückzuführen ist. Insbesondere, da hoch qualitatives Datenmaterial beschafft werden konnte, geben diese Ergebnisse Anlass, die automatisierte Insolvenzerkennung auf der Basis von Jahresabschlussdaten kritisch zu bewerten. Auch die Kern-PCA deutet auf einen geringen Informationsgehalt in den Daten hin.

Durch die Entwicklung des Tools InSolve bleiben die Ergebnisse nachvollziehbar und die Methoden auch in anderen Bereichen anwendbar. Außerdem kann das Tool InSolve auch als Entwicklungsplattform für größere Kreditinstitute dienen, wenngleich viele bereits eigene Ratingsysteme entwickeln.

Anhang A

A.1. Funktionen und Dokumentation von InSolve

A.1.1. Start von InSolve und Menus

Nach dem Start von InSolve durch die mitgelieferte Batch-Datei „InSolve.bat“, erscheint der Hauptdialog:

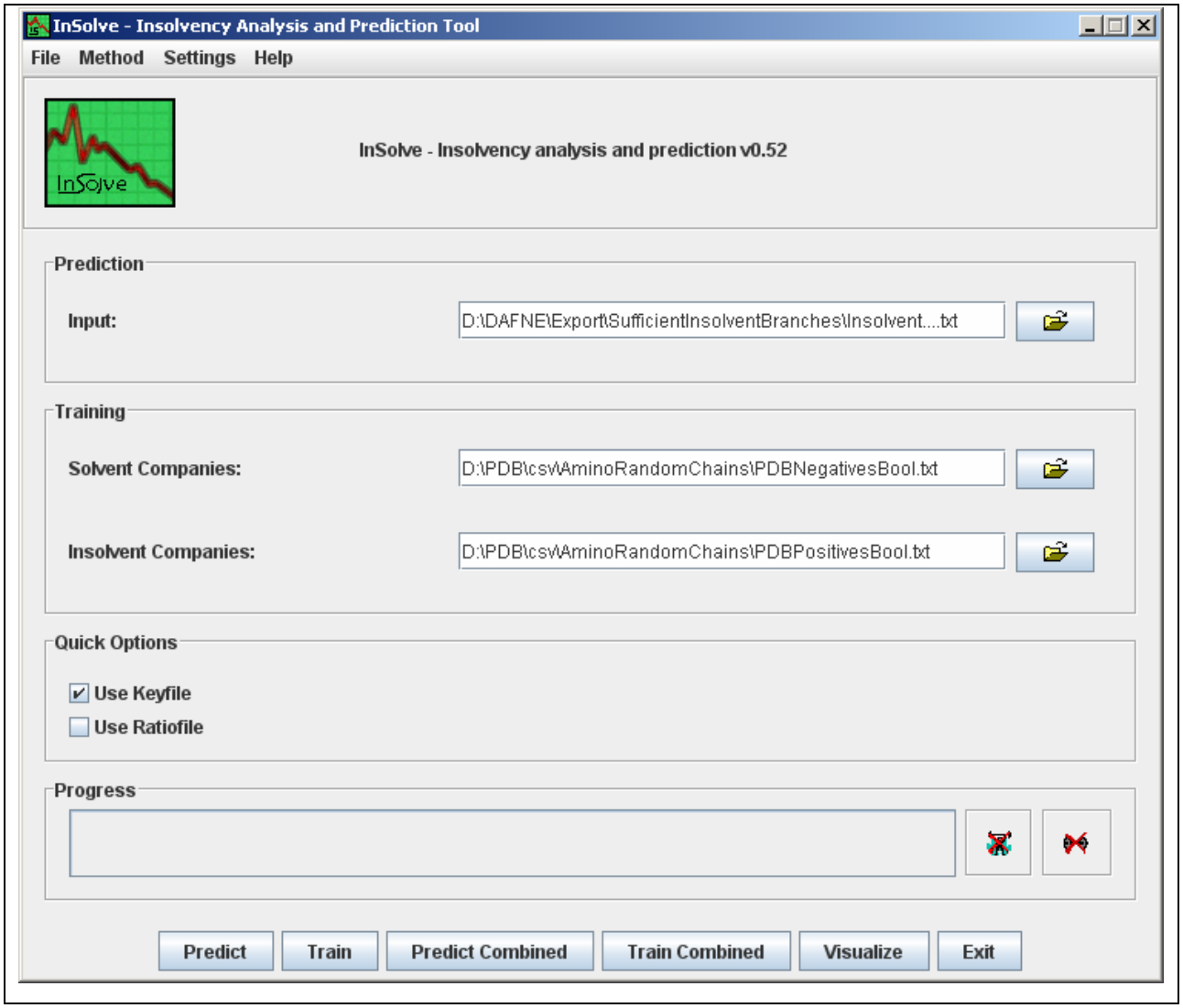


Abbildung 16: InSolve Hauptdialog

Alle wichtigen Funktionen können über das Hauptmenü gesteuert werden.

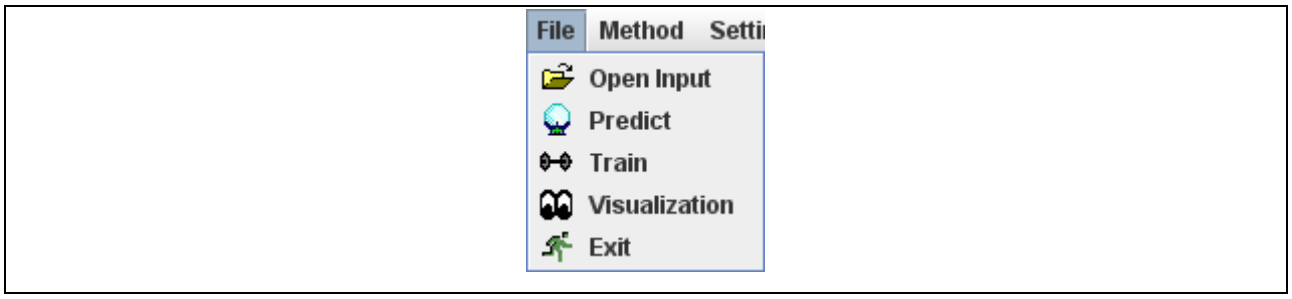


Abbildung 17: InSolve File Menu

Das Menü „File“ stellt die Funktionen zum öffnen einer Eingabe-Datei zur Vorhersage (Open Input), Start des Trainings (Train) und Start der Vorhersage (Predict) sowie Visualisierung (Visualization) zur Verfügung.

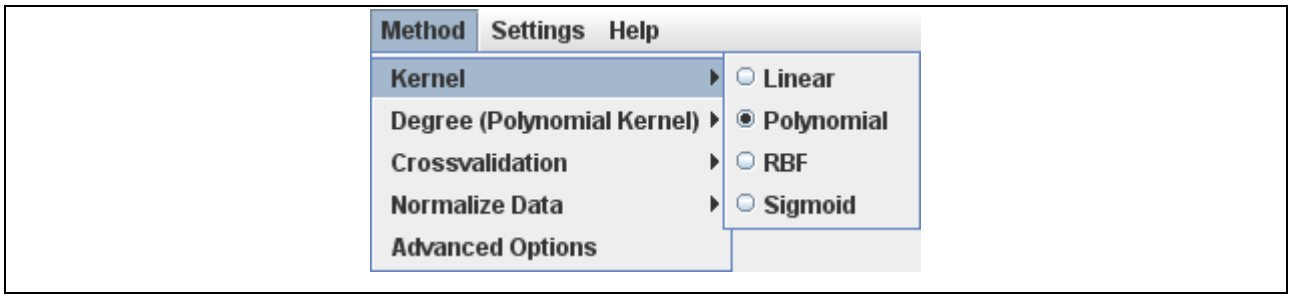


Abbildung 18: InSolve Method Menu

Das Menü „Method“ ist von dem gewählten Plugin abhängig (siehe Kapitel A.1.7) und wird dynamisch erzeugt. Es erlaubt die Konfiguration der aktuell gewählten Methode.

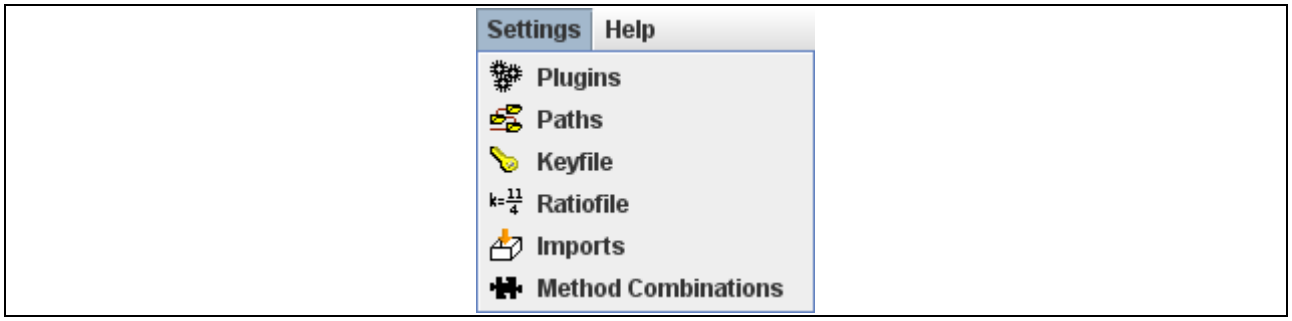


Abbildung 19: InSolve Settings Menu

Das Menü „Settings“ erlaubt den Zugriff auf wichtige Dialoge von InSolve. Dazu zählen der Plugin-Manager (Plugins), Keyfile-Dialog (Keyfile), Ratiofile-Dialog (Ratiofile), Import-Dialog (Imports) und Method-Combination-Dialog (Method Combinations). Weiterhin kann der Pfad-

Dialog (Paths) aufgerufen werden, der aber oberflächlich bereits durch den Keyfile-Dialog und Ratiofile-Dialog konfiguriert werden kann.

A.1.2. Eingabedateien und Import-Dialog

Der erste Schritt für das Training eines Klassifikators mit InSolve ist die Erstellung von Eingabedateien. InSolve akzeptiert CSV-Dateien, die z.B. mit Microsoft Excel oder OpenOffice.org erzeugt werden können. Jede Zeile einer CSV-Datei entspricht dabei einem Jahresabschluss, jede Spalte einer Kennzahl oder einer beliebigen anderen Information. Die erste Zeile der CSV-Datei fungiert dabei als Header und bezeichnet damit die verschiedenen Kennzahlen oder anderer Informationen. Es ist zwingend erforderlich, dass der Header eindeutige Bezeichnungen aufweist, da andernfalls nicht alle Spalten korrekt verarbeitet werden können:

Name	**Datum**	**Branche**	**Bilanzsumme**	**Cash-Flow**
“U1 GmbH“	“31.12.2003“	“Bau“	2051	1.51
“U1 GmbH“	“31.12.2004“	“Bau“	2501	1.71
“U1 GmbH“	“31.12.2005“	“Bau“		1.91
“U2 AG“	“31.12.2005“	“Dienst“	22115	-1.5
“U3 AG“	“31.12.2004“	“Dienst“	551	3.56
“U3 AG“	“31.12.2005“	“Dienst“	422	3.4

Abbildung 20: Aufbau einer CSV-Datei für InSolve

Entscheidend ist bei Gleitkommawerten, dass sie mit einem Punkt anstelle eines Kommas angegeben werden. Dies ist Standard in vielen Programmiersprachen und wird auch bei InSolve nicht anders akzeptiert. InSolve erkennt automatisch Spalten mit numerischen Werten und konvertiert diese entsprechend. Die aktuellen Plugins unterstützen ausschließlich die Verarbeitung numerischer Werte.
Weiterhin bietet InSolve für das Einlesen von CSV-Dateien einen eigenen Dialog, der die eingelesenen Dateien in einer Preview-Tabelle anzeigt, um Konvertierungsfehler auszuschließen.

Der Import-Dialog wird über den Menüpunkt *Settings→Imports* aufgerufen:

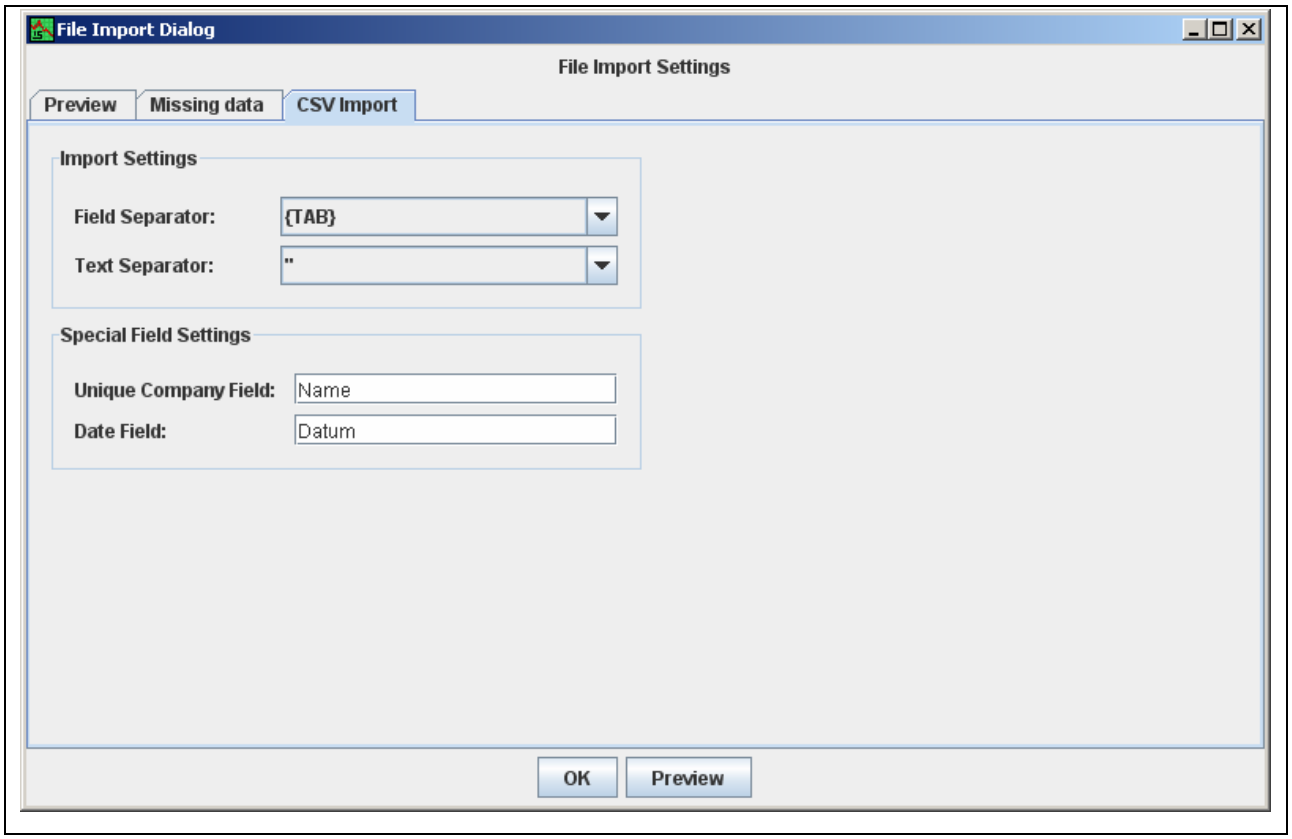


Abbildung 21: Import-Dialog: CSV Import

Im Reiter „CSV Import“ lassen sich Option für den CSV-Parser festlegen. Diese sind zum einen der „Field Separator“ für die CSV-Datei (Tabulator im oberen Beispiel) sowie der „Text Separator“ (Anführungszeichen im oberen Beispiel). Weiterhin muss die CSV-Datei über eine Spalte mit eindeutigen Werten für jedes Unternehmen verfügen, das so genannte „Unique Company Field“. Über den hier angegebenen Spaltennamen erfolgt die Zuordnung verschiedener Jahresabschlüsse zu einem Unternehmen. Wird „Unique Company Field“ für das obere Beispiel auf „Name“ gesetzt, so werden die ersten drei Jahresabschlüsse dem Unternehmen „U1 GmbH“ zugeordnet. „U2 AG“ besitzt einen Jahresabschluss und „U3 GmbH“ die letzten zwei.
Ein weiteres, optionales Feld ist das „Date Field“, in das für gewöhnlich der Spaltenname des jeweiligen Jahresabschlussdatums angegeben werden sollte. Sinnigerweise wird in obigem Beispiel „Datum“ eingetragen. Diese Option hat lediglich Auswirkung auf den Visualisierungsdialog und zeigt die Information für den jeweiligen Datenpunkt in einem Popupmenu an. Aus diesem Grund können auch beliebige andere Informationen eingeblendet werden.

Der Reiter „Missing Data“ des Import-Dialogs erlaubt die Auswahl verschiedener Vervollständigungsstrategien für fehlende Werte:

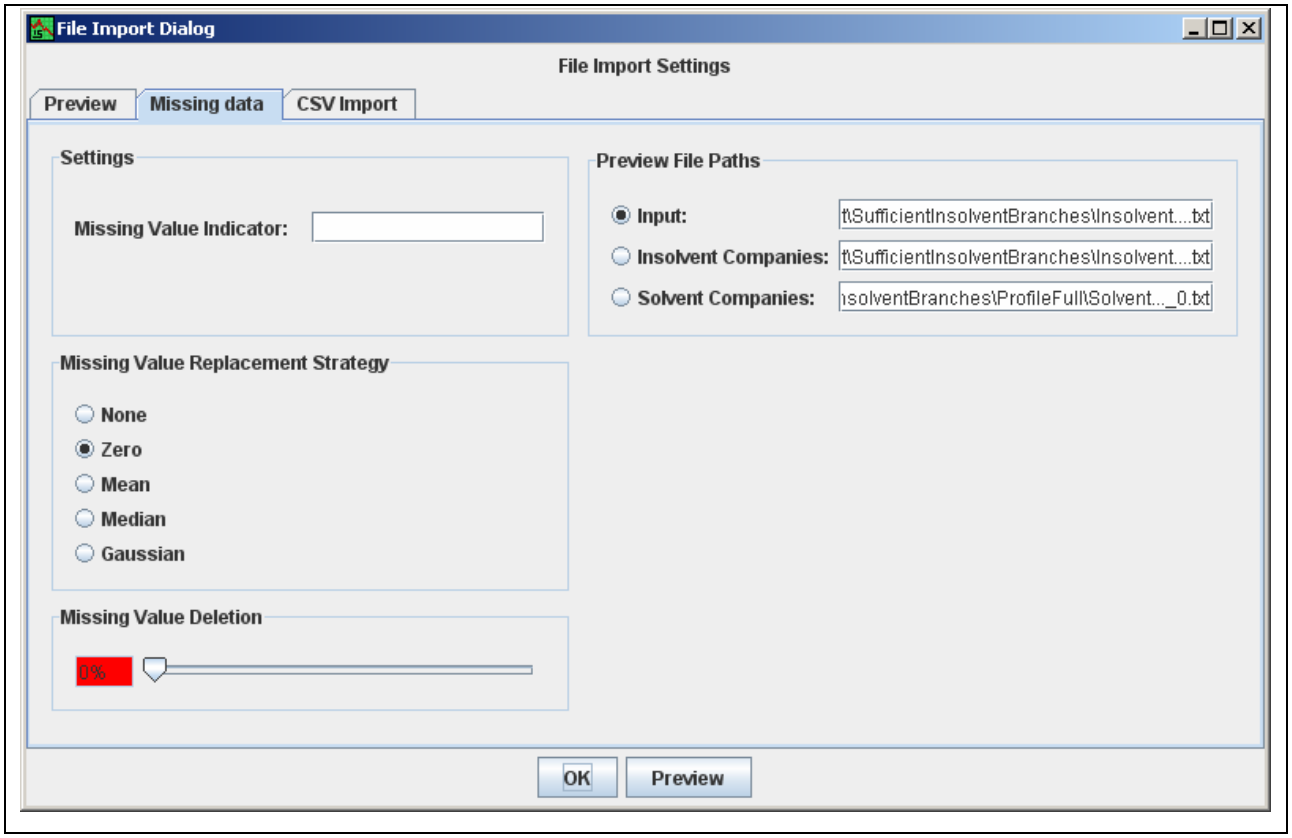
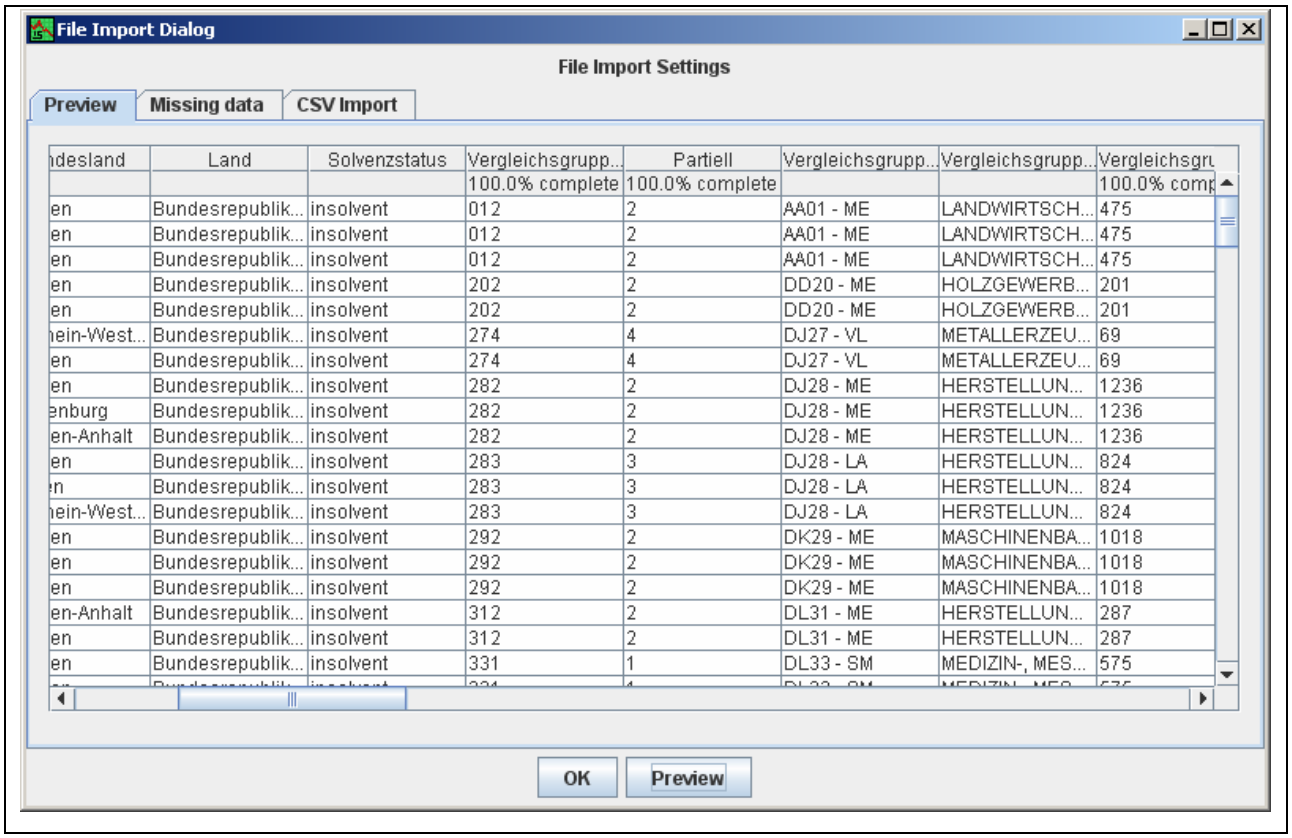


Abbildung 22: Import-Dialog: Missing Values

In diesem Dialog muss zunächst der „Missing Value Indicator“ eingegeben werden, der die als fehlend zu interpretierenden Werte in der CSV-Datei angibt. Im Regelfall werden fehlende Werte in einer CSV-Datei leer gelassen, daher muss im Normalfall nichts eingetragen werden. Im obigen Beispiel besitzt das Unternehmen „U1 GmbH“ einen Jahresabschluss mit fehlender Bilanzsumme. Dieser Wert würde von der gewählten Vervollständigungsstrategie ersetzt.
In der Kategorie „Missing Value Replacement Strategy“ kann die Strategie zur Vervollständigung der fehlenden Werte ausgewählt werden. Die folgenden Strategien stehen zur Auswahl:

- Zero (Fehlender Wert wird durch 0 ersetzt)
- Mean (Fehlender Wert wird durch gruppenspezifischen Erwartungswert ersetzt)
- Median (Fehlender Wert wird durch gruppenspezifischen Median ersetzt)
- Gaussian (Fehlender Wert wird durch gruppenspezifischen, zufälligen Wert ersetzt, der aus einer empirisch geschätzten Standardnormalverteilung gezogen wird)

In der Kategorie „Missing Value Deletion“ kann angegeben werden, wie viel Prozent der vorliegenden Jahresabschlüsse einen Wert in einer Spalte aufweisen müssen, damit dieses Merkmal nicht von der Verarbeitung ausgeschlossen wird. Ein Merkmal, das in weniger als 50% aller Jahresabschlüsse auftaucht, eignet sich schlecht für eine Schätzung und sollte nicht verwendet werden. Es wird daher empfohlen, den Regler auf 50% oder höher einzustellen.
Die Kategorie „Preview File Paths“ enthält automatisch die eingegebenen Pfade des InSolve Hauptdialogs. Alternativ können aber auch andere Dateinamen eingetragen werden. Die hier ausgewählte Eingabedatei wird durch einen Klick auf den Button „Preview“ eingelesen und unter dem Reiter „Preview“ angezeigt:



ndesland	Land	Solvenzstatus	Vergleichsgrupp...	Partiell	Vergleichsgrupp...	Vergleichsgrupp...	Vergleichsgru
			100.0% complete	100.0% complete			100.0% comp
en	Bundesrepublik...	insolvent	012	2	AA01 - ME	LANDWIRTSCH...	475
en	Bundesrepublik...	insolvent	012	2	AA01 - ME	LANDWIRTSCH...	475
en	Bundesrepublik...	insolvent	012	2	AA01 - ME	LANDWIRTSCH...	475
en	Bundesrepublik...	insolvent	202	2	DD20 - ME	HOLZGEWERB...	201
en	Bundesrepublik...	insolvent	202	2	DD20 - ME	HOLZGEWERB...	201
hein-West...	Bundesrepublik...	insolvent	274	4	DJ27 - VL	METALLERZEU...	69
en	Bundesrepublik...	insolvent	274	4	DJ27 - VL	METALLERZEU...	69
en	Bundesrepublik...	insolvent	282	2	DJ28 - ME	HERSTELLUN...	1236
enburg	Bundesrepublik...	insolvent	282	2	DJ28 - ME	HERSTELLUN...	1236
en-Anhalt	Bundesrepublik...	insolvent	282	2	DJ28 - ME	HERSTELLUN...	1236
en	Bundesrepublik...	insolvent	283	3	DJ28 - LA	HERSTELLUN...	824
n	Bundesrepublik...	insolvent	283	3	DJ28 - LA	HERSTELLUN...	824
hein-West...	Bundesrepublik...	insolvent	283	3	DJ28 - LA	HERSTELLUN...	824
en	Bundesrepublik...	insolvent	292	2	DK29 - ME	MASCHINENBA...	1018
en	Bundesrepublik...	insolvent	292	2	DK29 - ME	MASCHINENBA...	1018
en	Bundesrepublik...	insolvent	292	2	DK29 - ME	MASCHINENBA...	1018
en-Anhalt	Bundesrepublik...	insolvent	312	2	DL31 - ME	HERSTELLUN...	287
en	Bundesrepublik...	insolvent	312	2	DL31 - ME	HERSTELLUN...	287
en	Bundesrepublik...	insolvent	331	1	DL33 - SM	MEDIZIN-, MES...	575

Abbildung 23: Import-Dialog: Preview

Dieser Dialog zeigt außerdem die prozentuale Vollständigkeit eines Merkmals in der zweiten Zeile an. Hier sollten die eingelesenen Daten einem manuellen Plausibilitätscheck unterworfen werden, um Einlesefehler zu erkennen.

A.1.3. Training und Vorhersage mit InSolve

Zunächst müssen zwei CSV-Dateien (solvente und insolvente Unternehmen) für das Training des Klassifikators bereitgestellt werden. Der Pfad der Dateien muss in die entsprechenden Textboxen

unter „Training“ eingetragen oder über den Dateibrowser ausgewählt werden. Anschließend kann der Trainingsvorgang über den Button „Train“ der unteren Buttonleiste oder den entsprechenden Menüpunkt gestartet werden. Ist das Training erfolgreich, so ändert sich das Symbol für einen trainierten Klassifikator entsprechend:

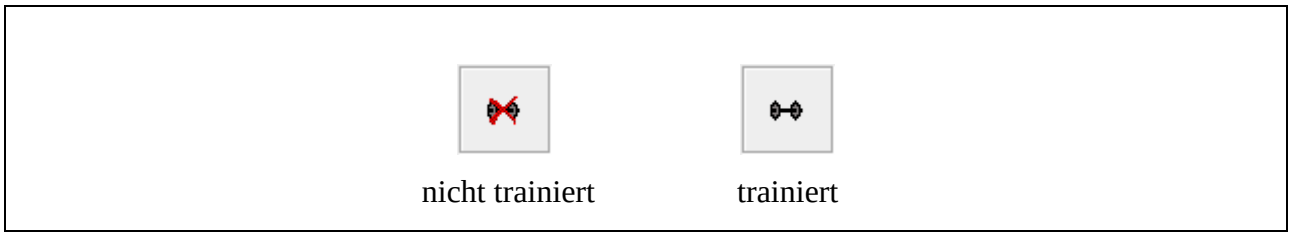


Abbildung 24: Icons: Trainiert / Nicht trainiert

Standardmäßig ist das Thresholder-Verfahren gewählt und wird daher für Training und Vorhersage verwendet. Andere Verfahren können über den Plugin-Dialog ausgewählt werden (siehe Kapitel A.1.7).
Um eine Beurteilung von neuen Unternehmen vorzunehmen, muss der Pfad zu der gewünschten CSV-Datei unter der Kategorie „Prediction“ eingetragen werden. Anschließend kann die Vorhersage mit einem Klick auf den Button „Predict“ gestartet werden.

Sobald die Vorhersage abgeschlossen ist, wird der Ergebnisdialog angezeigt:

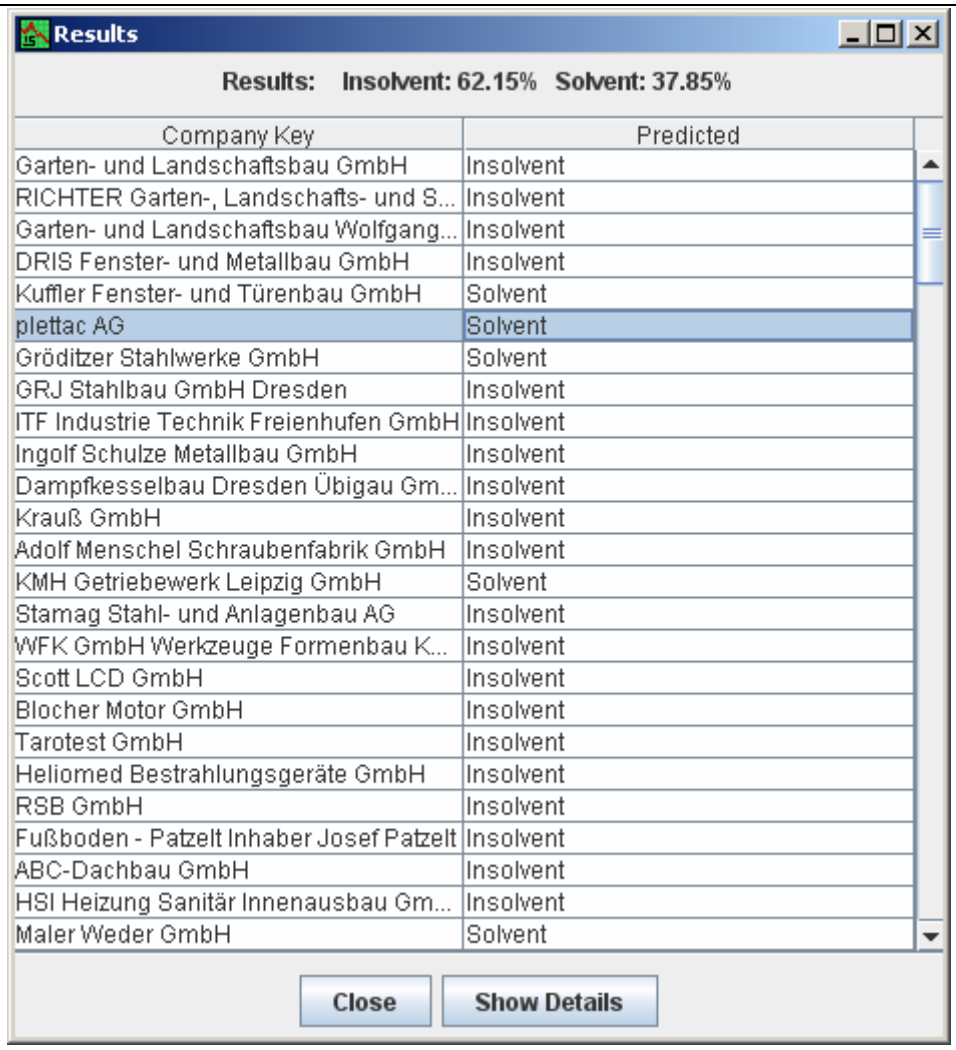


Abbildung 25: Results-Dialog

Der Results-Dialog zeigt dabei das vorhergesagte Label für jedes Unternehmen in der Spalte „Predicted“ sowie die prozentualen Anteile aller vorhergesagten Labels. Da für die Insolvenzvorhersage Transparenz entscheidend ist, besitzen die verwendeten Plugins eine Dokumentationsmöglichkeit, die über den Button „Show Details“ angezeigt werden kann.

Für das ausgewählte Unternehmen wird dann der Result-Details-Dialog angezeigt:

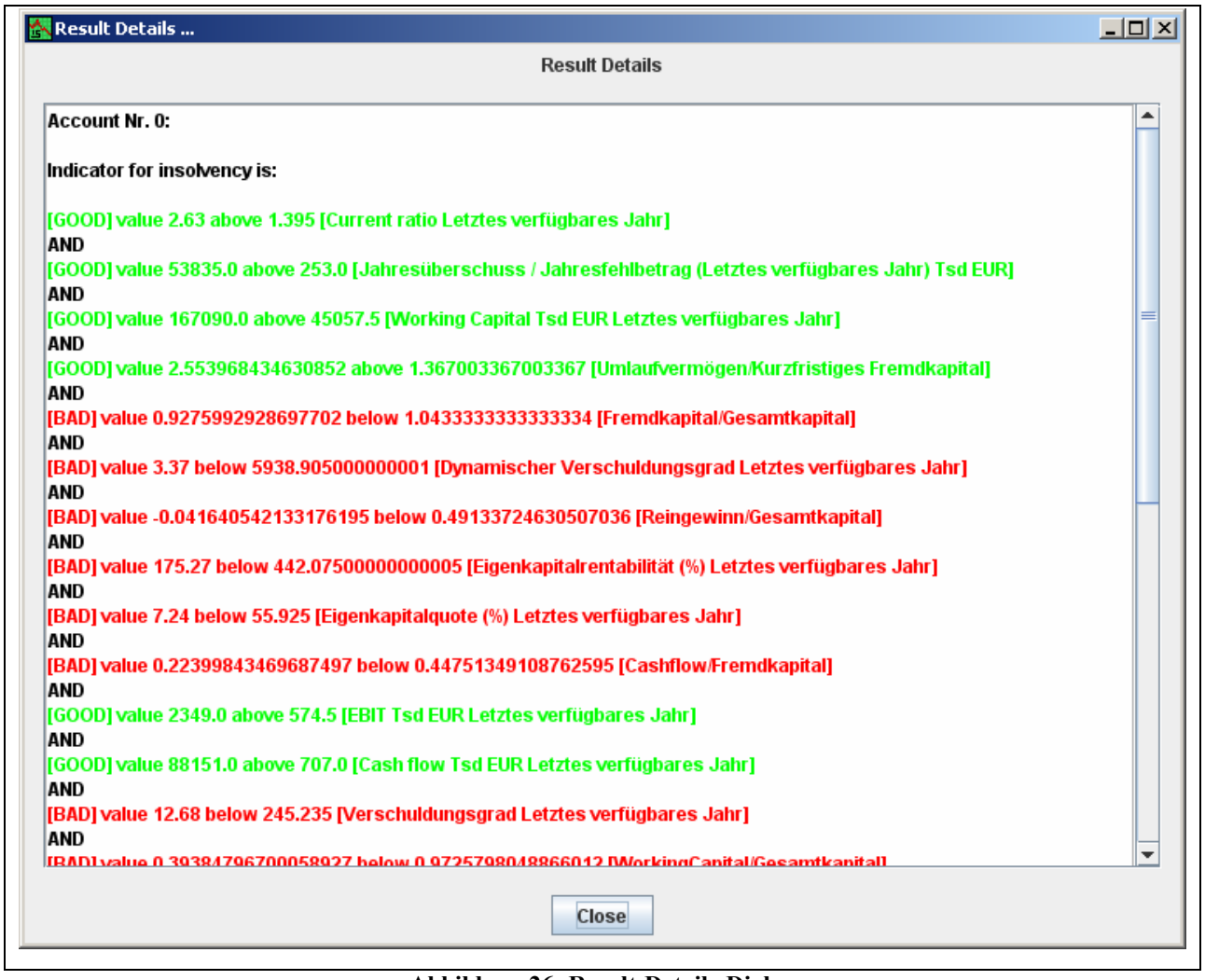


Abbildung 26: Result-Details-Dialog

Das Thresholder-Plugin dokumentiert für jedes Merkmal und jede Schwelle, welche Werte des zu beurteilenden Unternehmens auf eine Insolvenz hindeuten (rot) und welche Werte dagegen sprechen (grün).
InSolve fragt den Nutzer bei Beendigung des Programms, ob der trainierte Klassifikator gespeichert werden soll. Für jede Methode kann ein Klassifikator gespeichert werden.

A.1.4. Ratiofile-Dialog: Erstellung eigener Kennzahlen

InSolve bietet die Möglichkeit, einfache Rechenoperationen auf den eingelesenen CSV-Dateien auszuführen. Auf diese Weise ist es möglich, anhand von Rohdaten bestimmte Kennzahlen automatisch berechnen zu lassen, ohne dass diese in den CSV-Dateien selbst vorberechnet sein müssen. Die Informationen für diese Rechenvorschrift kann in einem Ratiofile mit Endung .rat

gespeichert und wieder verwendet werden. Die Erstellung der Ratiofiles kann mit dem Ratiofile-Dialog erfolgen, der über den Menüpunkt *Settings→Ratiofile* aufgerufen wird:

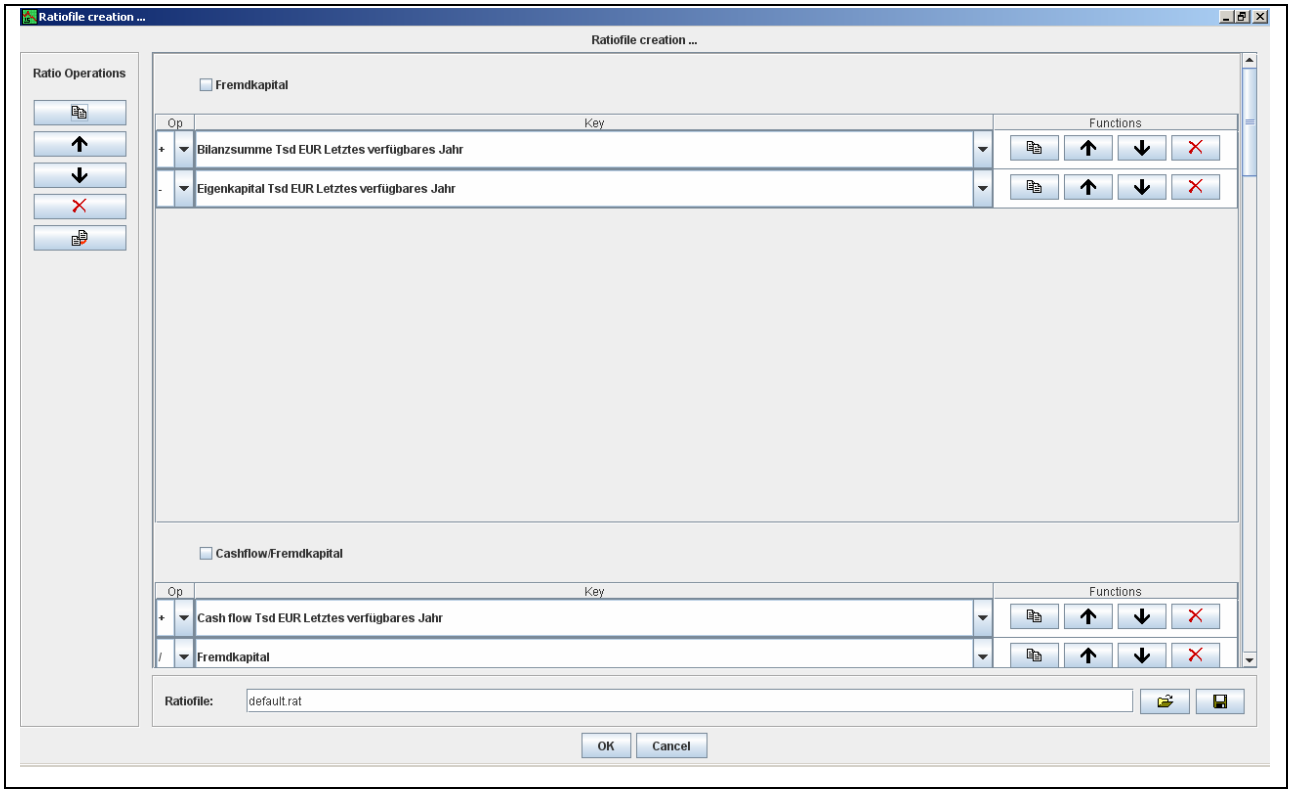


Abbildung 27: Ratiofile-Dialog

Dieser Dialog ist größtenteils selbsterklärend. Jede zu berechnende Kennzahl ist durch eine Checkbox repräsentiert (die Abbildung zeigt die zu berechnenden Größen „Fremdkapital“ und „Cashflow/Fremdkapital“). Darunter werden die einzelnen Operanden über die Headerbezeichnungen der Trainings-CSV-Dateien (Bilanzsumme und Eigenkapital bzw. Cash-Flow) in jeweils einzelnen Comboboxen angezeigt.

InSolve prüft bereits beim Aufruf des Ratiofile-Dialogs die Trainingsdateien auf Konsistenz und liefert gegebenenfalls eine entsprechende Meldung.

Links des Operanden kann der Operator (+, -, *, /) auf gleiche Weise ausgewählt werden. Eine Klammernsetzung ist nicht direkt möglich, sondern muss durch die Berechnung von Teilgrößen vorgenommen werden, wie im obigen Beispiel bei der Berechnung des Fremdkapitals gezeigt.

Um einen bestimmten Operanden zu finden, bietet InSolve eine Autovervollständigungsfunktion an:

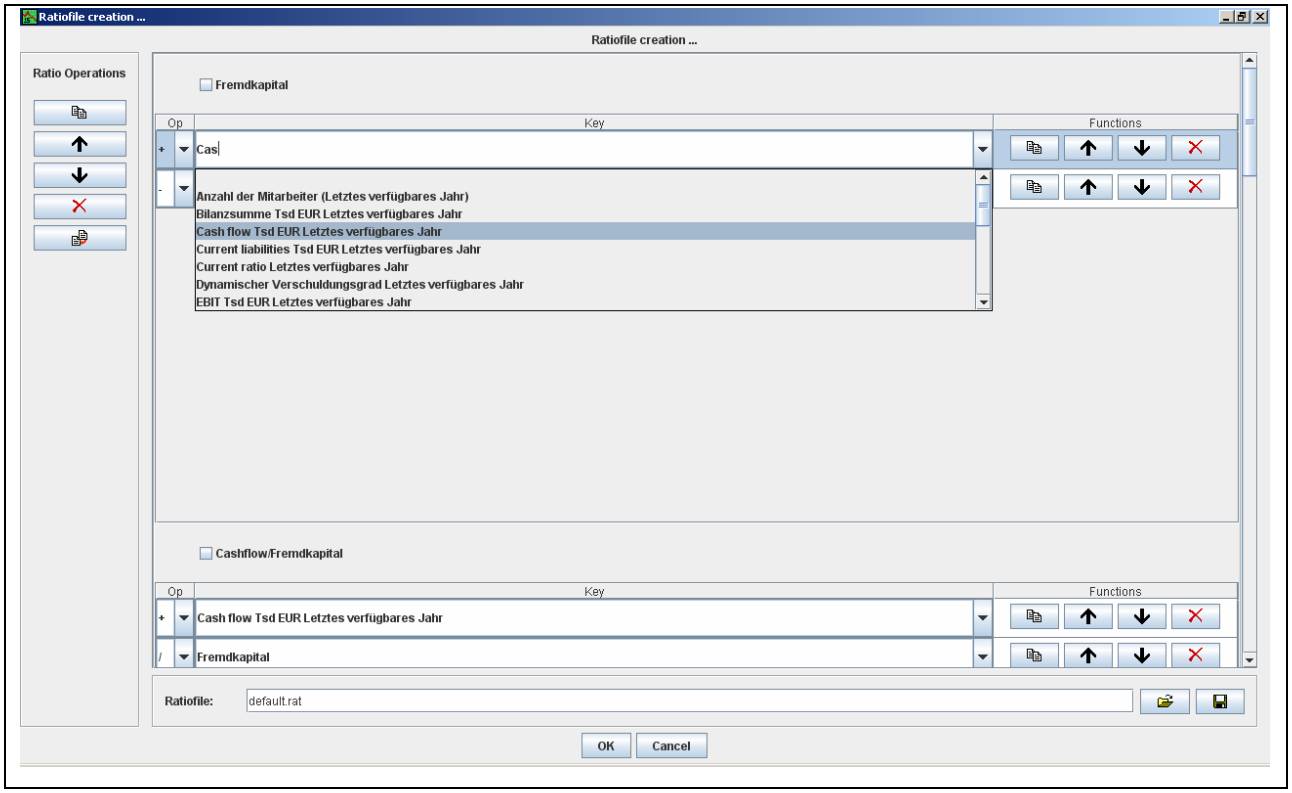


Abbildung 28: Ratiofile-Dialog Autovervollständigung

Parallel zur Nutzereingabe werden die verfügbaren Headerwerte aus den Trainingsdateien durchsucht. Sobald der korrekte Eintrag in der Liste markiert ist, genügt ein Druck auf „Enter“, um die Eingabe zu übernehmen. Achtung, der gewünschte Wert kann auch durch einen Mausklick aus der Liste ausgewählt werden, jedoch muss die Eingabe abschließend mit einem Druck auf „Enter“ bestätigt werden, da andernfalls die Änderungen nicht übernommen werden. Es ist möglich die Reihenfolge der Operanden zu vertauschen, indem auf den entsprechenden Button bzw. rechts neben dem gewünschten Eintrag geklickt wird. Ein neuer Operand lässt sich über den Button einfügen, der außerdem den aktuell gewählten Operanden vollständig kopiert. Das Löschen eines Operanden erfolgt über den Button .

Die Reihenfolge der zu berechnenden Kennzahlen entspricht der tatsächlichen Berechnungsreihenfolge. Aus diesem Grunde steht eine berechnete Kennzahl darunter stehenden Kennzahlen als Operand zur Verfügung.

Um eine neue Kennzahl zu erstellen, muss der Button auf der linken Toolleiste „Ratio Operations“ angeklickt werden. InSolve fordert den Nutzer dann auf, einen Namen für die Kennzahl einzugeben:

Abbildung 29: Ratiofile-Dialog: Eingabe eines Namens für eine neue Kennzahl

Auch hier ist es erneut wichtig, eine eindeutige Bezeichnung zu wählen. InSolve prüft die Eingabe und weist gegebenenfalls auf ein Problem hin.
Es ist auch möglich, mehre Kennzahlen gleichzeitig bezüglich ihrer Position und damit ihrer Berechnungsreihenfolge zu verschieben. Dazu müssen die entsprechenden Checkboxen der Kennzahlen markiert und der der Button bzw. angeklickt werden. Analog wird für das Löschen mehrerer Kennzahlen vorgegangen und der Button angeklickt.

Ein so konfiguriertes Ratiofile kann schließlich über die untere Pfadleiste gespeichert werden. Durch einen Klick auf den Button wird das Ratiofile unter dem angegebenen Pfad gespeichert. Wird der Dialog mit einem Klick auf den Button „OK“ geschlossen, so fragt InSolve, ob das aktuelle Ratiofile für die nächste Vorhersage übernommen werden soll. Wird dieser Dialog mit „Ja“ bestätigt, so wird das Ratiofile automatisch in den Paths-Dialog (siehe Kapitel A.1.6) übernommen.

A.1.5. Keyfile-Dialog: Merkmalsbeschränkung und Eingabekontrolle

Keyfiles erlauben eine bessere Kontrolle der vorliegenden CSV-Dateien und bieten außerdem eine einfache Möglichkeit, die aktuelle Merkmalsauswahl zu verändern, ohne dazu neue CSV-Dateien erzeugen zu müssen.

Der Keyfile-Dialog unterstützt den Nutzer bei der Erstellung von Keyfiles und wird über den Menüpunkt *Settings→Keyfile* aufgerufen:

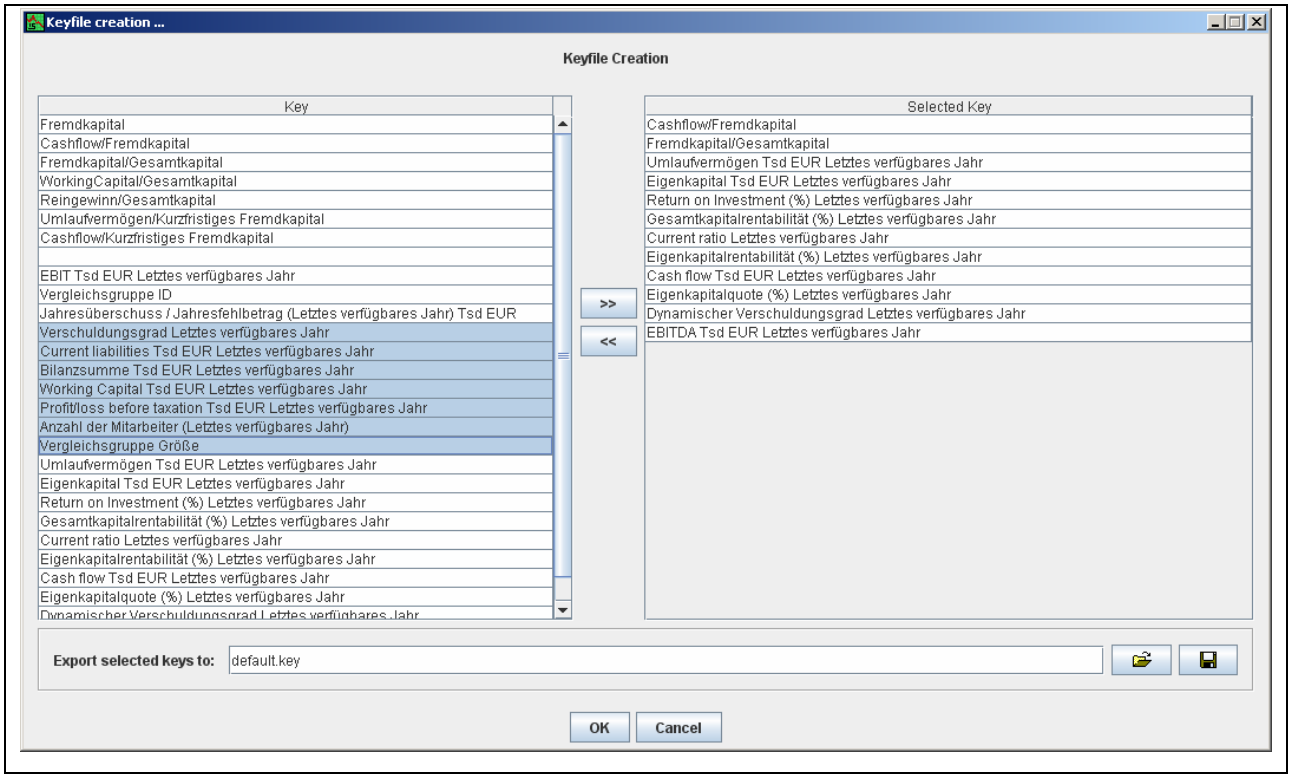


Abbildung 30: Keyfile-Dialog

Die linke Tabelle „Key“ listet alle zur Auswahl stehenden Headerbezeichnungen der Trainingsdateien auf, inklusive selbst berechneter Kennzahlen durch ein Ratiofile (siehe Kapitel A.1.4). Die rechte Tabelle „Selected Key“ listet die ausgewählten Schlüssel auf. Mit Hilfe der Buttons „>>“ und „<<“ können einzelne oder mehrere Schlüssel der rechten Tabelle hinzugefügt bzw. aus ihr entfernt werden.

Ein so konfiguriertes Keyfile kann schließlich über die untere Pfadleiste gespeichert werden. Durch einen Klick auf den Button wird das Keyfile unter dem angegebenen Pfad gespeichert. Wird der Dialog mit einem Klick auf den Button „OK“ geschlossen, so fragt InSolve, ob das aktuelle Keyfile für die nächste Vorhersage übernommen werden soll. Die Einbindung und Verwaltung von Keyfiles erfolgt über den Paths-Dialog (siehe Kapitel A.1.6).

A.1.6. Paths-Dialog

Der Paths-Dialog wird über den Menüpunkt *Settings*→*Paths* aufgerufen:

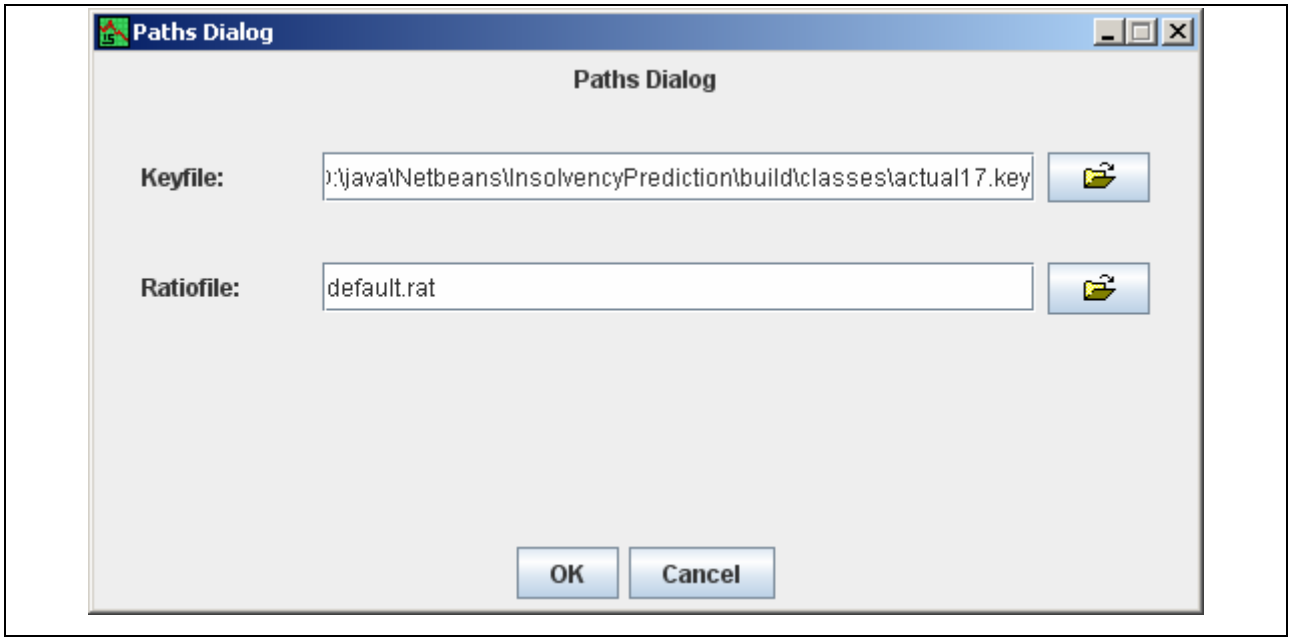


Abbildung 31: Paths-Dialog

Der Paths-Dialog ermöglicht die Angabe von Pfaden für Keyfile und Ratiofile, die zuvor mit den Dialogen Ratiofile bzw. Keyfile erstellt worden sind. Die Pfade können von Hand oder über einen Dateibrowser angegeben werden. Der Hauptdialog von InSolve bietet außerdem die Möglichkeit, die Verwendung eines hier angegebenen Ratiofiles oder Keyfiles komplett auszusetzen:

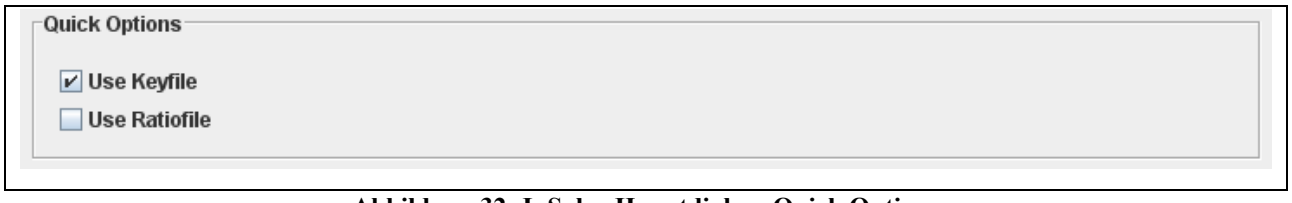


Abbildung 32: InSolve Hauptdialog: Quick Options

Dazu muss lediglich die entsprechende Checkbox aktiviert bzw. deaktiviert werden. Es ist jedoch nicht empfehlenswert, ohne ein Keyfile zu arbeiten, da auf diese Weise Fehler in den Eingabedateien schwerer zu erkennen sind.

A.1.7. Plugin-Dialog

Der Plugin-Dialog wird über den Menüpunkt *Settings→Plugins* aufgerufen:

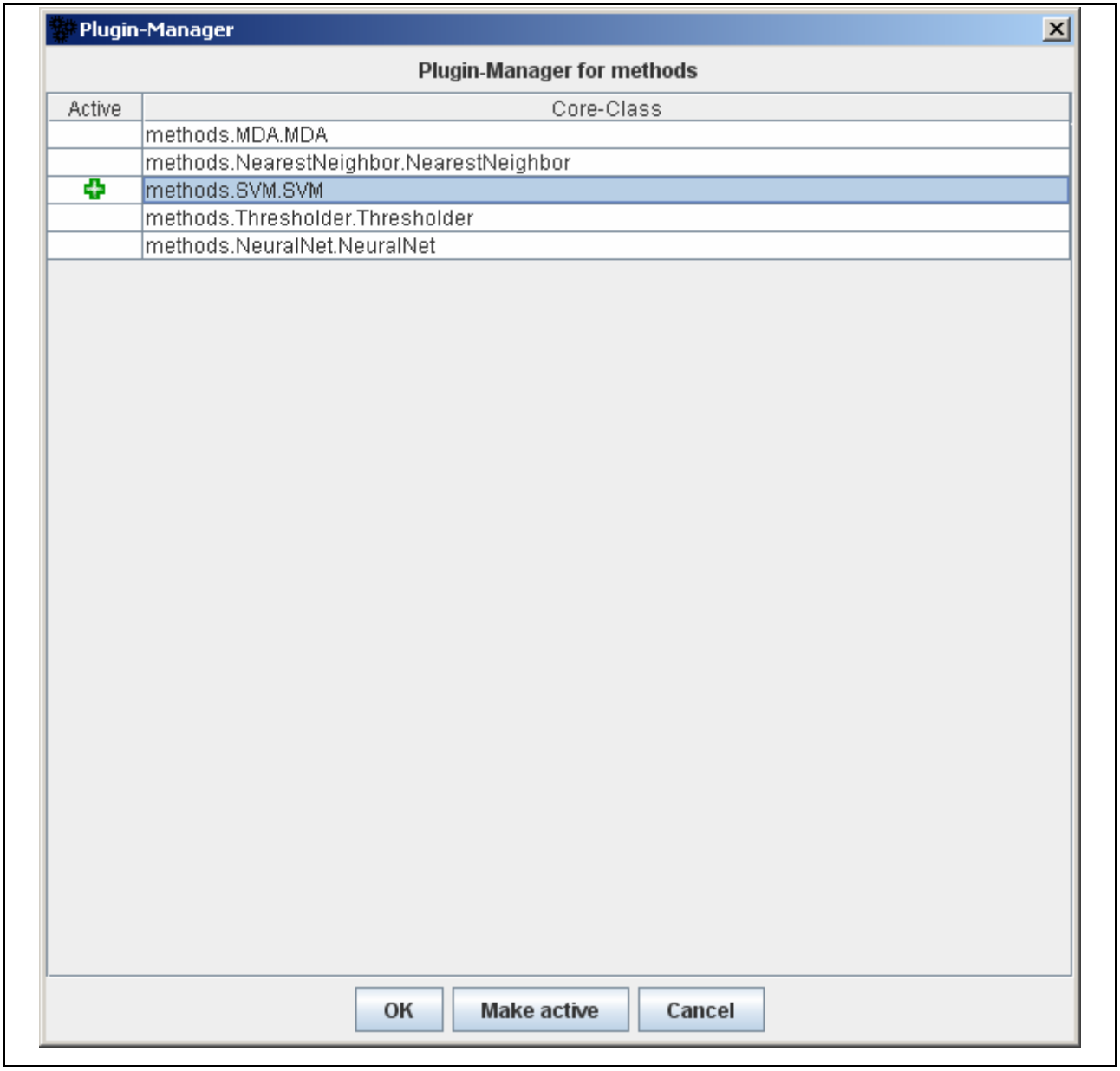


Abbildung 33: Plugin-Dialog

Der Plugin-Dialog ermöglicht die Auswahl und Aktivierung neuer Plugins für InSolve. Plugins sind Methoden-Pakete, die speziell für InSolve entwickelt wurden und lassen sich separat erwerben.

Bei Auslieferung existieren die folgenden Plugins:

- MDA (Multivariate Diskriminanzanalyse)
- Thresholder (Das in dieser Arbeit vorgestellte Schwellwert-Verfahren)
- Nearest Neighbor (Methode des nächsten Nachbarn)
- SVM (Support Vector Machine basierend auf libsvm (siehe [LSVM])
- NeauralNet (Neuronales Netz basierend auf Joone (siehe [JOONE])

Um ein weiteres Plugin zu installieren, muss das Plugin-Paket in das Unterverzeichnis „plugins“ des InSolve Hauptverzeichnisses kopiert werden. Bei einem erneuten Aufruf des Plugin-Dialogs werden diese Pakete automatisch erkannt und verifiziert. Um ein Plugin zu aktivieren, muss dieses in der Liste ausgewählt und der Button „Make active“ angeklickt werden. Das grüne Kreuz markiert dann das gegenwärtig aktive Plugin. Durch einen Klick auf den Button „OK“ wird der Dialog geschlossen und das Menü „Methods“ des InSolve Hauptdialogs entsprechend aktualisiert. Das Plugin kann dann wie gewohnt über diesen Menüpunkt konfiguriert werden.

A.1.8. Method-Combination-Dialog

Der Method-Combination-Dialog wird über den Menü-Punkt *Settings* → *Method Combinations* aufgerufen:

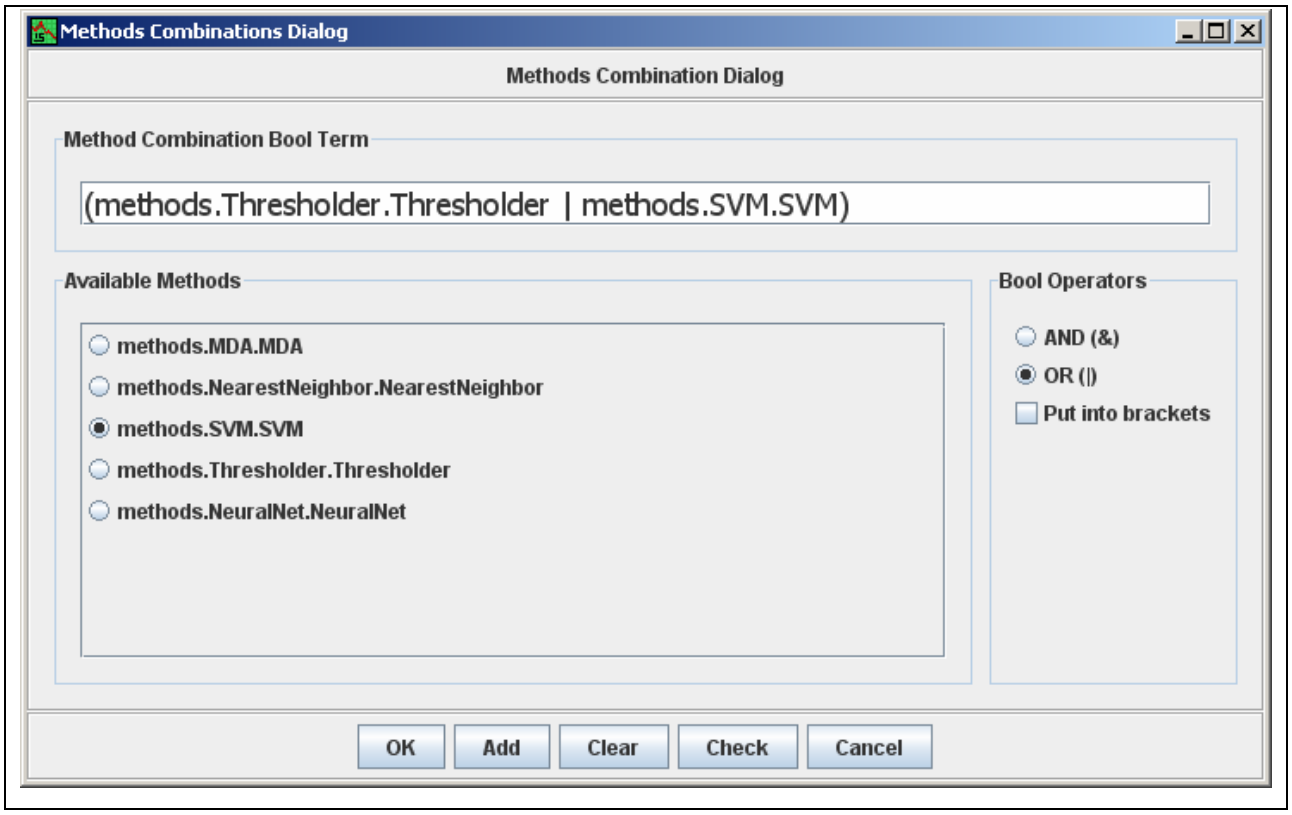


Abbildung 34: MethodsCombination-Dialog

Der Method-Combination-Dialog ermöglicht es, mehrere Methoden über eine boolesche Funktion miteinander zu kombinieren. Der Grundgedanke hinter diesem Dialog ist die Tatsache, dass meistens mehrere Verfahren für die Insolvenzerkennung eingesetzt werden. Daher ist es möglich, mit InSolve die Ergebnisse mehrerer Verfahren nach einer bestimmten Logik zu einem Gesamtergebnis zu vereinigen. Es ist weiterhin möglich, die Details über den Result-Details-Dialog jedes einzelnen Verfahrens abzurufen.

Die Kombination der Verfahren erfolgt über eine boolesche Funktion. Diese Funktion kann über die Kategorie „Bool Operators“ und „Available Methods“ einfach erzeugt werden. Dazu müssen die korrekten Einträge lediglich ausgewählt und mit einem Klick auf den Button „Add“ hinzugefügt werden.

Soll beispielsweise eine hohe Insolvenzerkennung mit den Methoden „Thresholder“ und „SVM“ erzielt werden, so wäre es möglich, die Ergebnisse beider Verfahren dem Operator „OR“ zu verknüpfen. Dazu wird das „Method Combination“ Textfeld zunächst gelöscht durch einen Klick

auf den Button „Clear“. Schließlich wird das Verfahren „methods.Thresholder.Thresholder“ aus der Kategorie „Available Methods“ gewählt und mit einem Klick auf den Button „Add“ hinzugefügt. Anschließend wird der Operator „OR (|)“ sowie die Methode „methods.SVM.SVM“ ausgewählt, und erneut der Button „Add“ angeklickt.
Diese boolesche Funktion würde also das Ergebnis „Insolvent“ liefern, sobald eine der beiden Methoden ein Unternehmen als „Insolvent“ einstuft.
Wird anstelle des Operators „OR (|)“ der Operator „And (&)“ verwendet, so würde lediglich dann das Ergebnis „Insolvent“ ausgegeben, wenn beide Methoden ein Unternehmen als „Insolvent“ einstufen.
Es ist auch möglich, die boolesche Funktion von Hand einzugeben. In diesem Fall sollte jedoch mit einem Klick auf den Button „Check“ die Syntax überprüft werden. Mit einem Klick auf den Button „OK“ werden die Änderungen übernommen und der Klassifikator kann über die gesonderten Buttons „Train Combined“ des InSolve Hauptdialogs trainiert werden. Die Vorhersage unter Verwendung des kombinierten Klassifikators erfolgt entsprechend über den Button „Predict Combined“.
Analog zu einem einfachen Training, wird das entsprechende Symbol für kombinierte Methoden bei Erfolg aktualisiert:

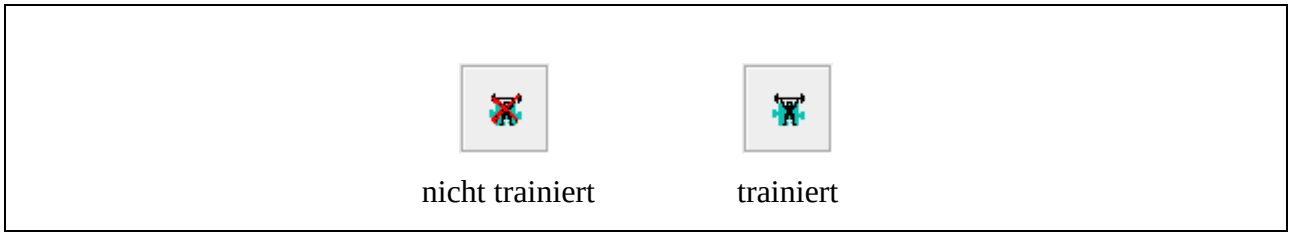


Abbildung 35: Icons: Trainiert / Nicht trainiert für kombinierte Methoden

Ein kombinierter Klassifikator kann analog zu einfachen Klassifikatoren (siehe Kapitel A.1.3) bei Beendigung von InSolve gespeichert werden.

A.1.9. Visualization-Dialog

Der Visualization-Dialog wird über den Menüpunkt *File→Visualization* aufgerufen:

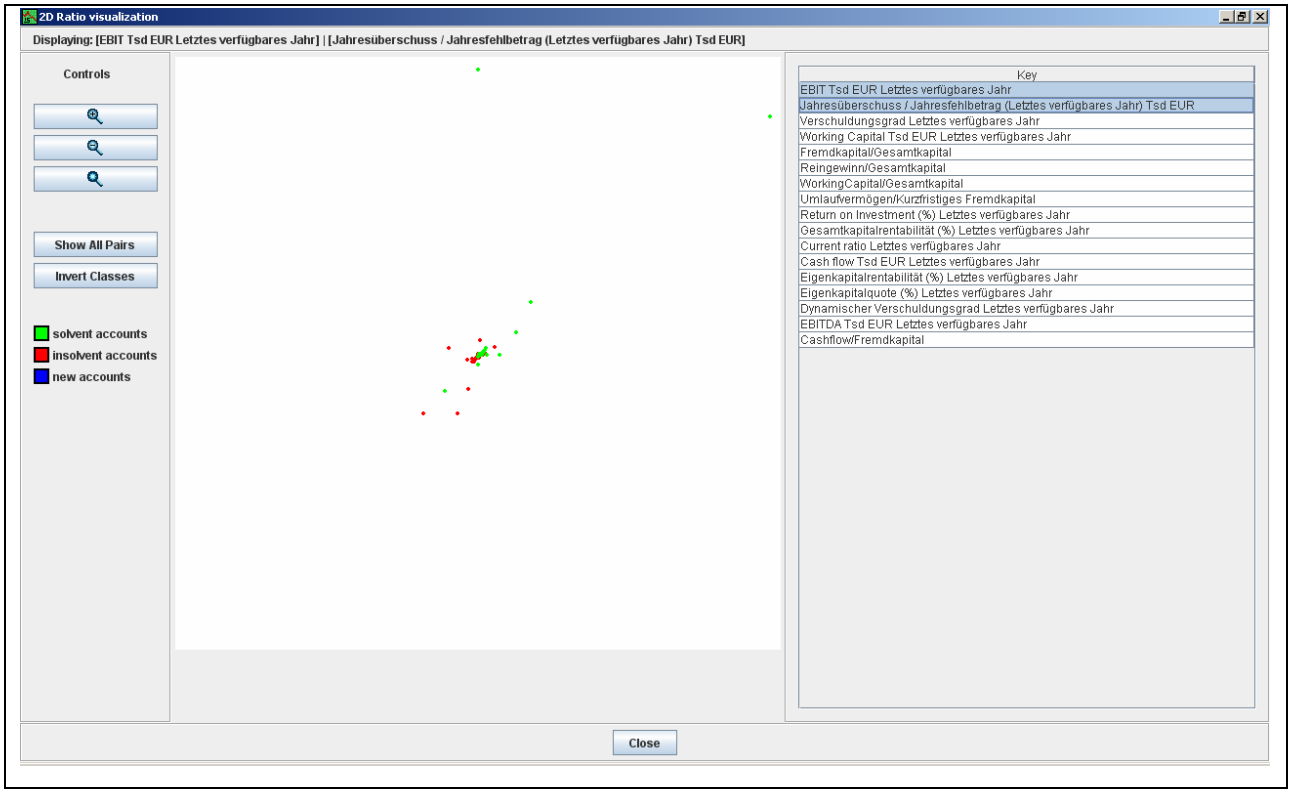


Abbildung 36: Visualization-Dialog

Der Visualization-Dialog erlaubt die Visualisierung von Kennzahlenpaaren. Jeder Datenpunkt entspricht dabei einem Jahresabschluss, also einer Zeile der Trainingsdateien und gegebenenfalls auch der Vorhersagedatei. Dabei werden Ratiofile und Keyfile ganz regulär berücksichtigt. Die rechte Tabelle stellt die zur Verfügung stehenden Kennzahlen dar, die mit Hilfe von STRG + Mausklick markiert werden können. Sobald genau zwei Einträge markiert sind, werden diese als Scatterplot auf der mittleren Zeichenfläche dargestellt. Mit Hilfe der linken Toolleiste, kann über die Buttons bzw. der gesamte Plot vergrößert bzw. verkleinert werden. Der Button ändert den Mauszeiger in ein Fadenkreuz und erlaubt dann die Vergrößerung und Zentrierung einer bestimmten Stelle des Plots.

Dies erfolgt durch einen Linksklick an die gewünschte Stelle des Plots, bzw. Rechtsklick um den Plot zu verkleinern.

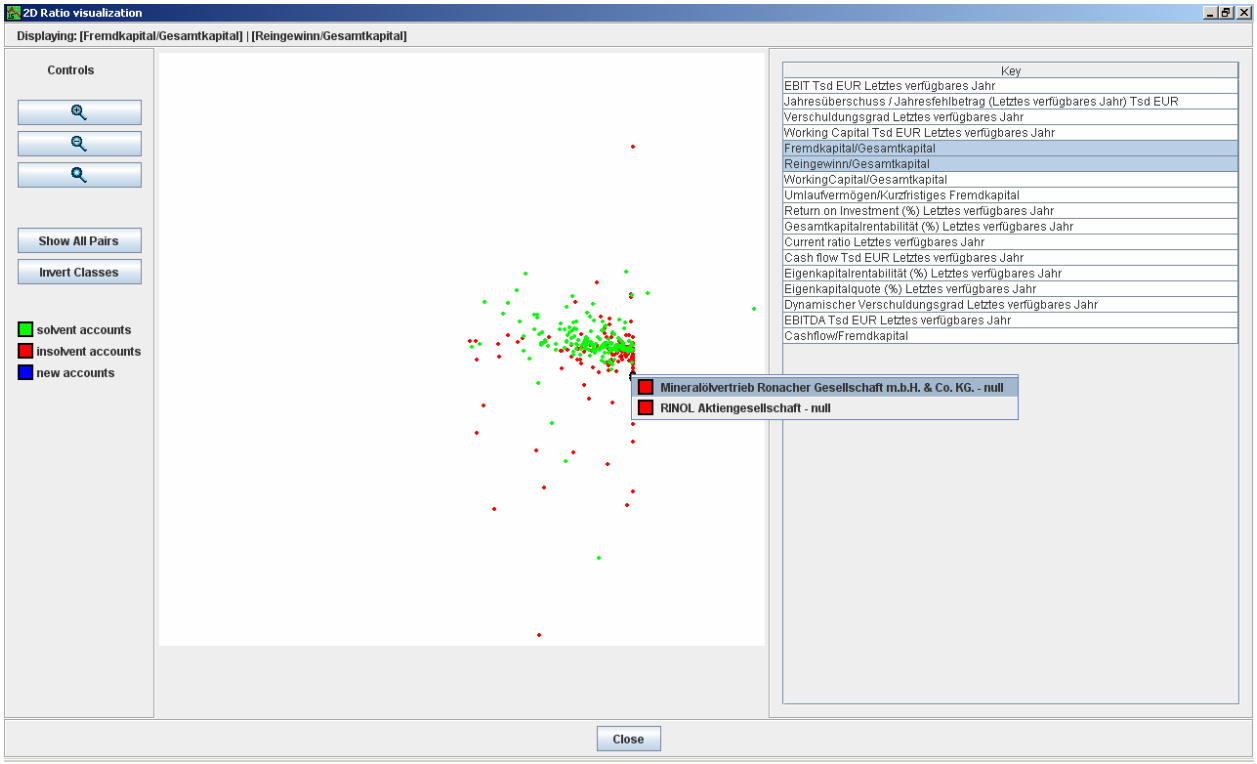


Abbildung 37: Visualization-Dialog: Unternehmensinformation für Datenpunkte

Der Visualization-Dialog kann außerdem ein Popupmenü mit den Jahresabschlussbezeichnungen anzeigen, das die Zuordnung der Datenpunkte zu den jeweiligen Jahresabschlüssen ermöglicht. Dazu müssen die gewünschten Datenpunkte lediglich angeklickt werden.

Die Information hinter dem Unternehmensnamen entspricht dem Eintrag der im Import-Dialog unter „Account Date“ eingetragenen Spalte.

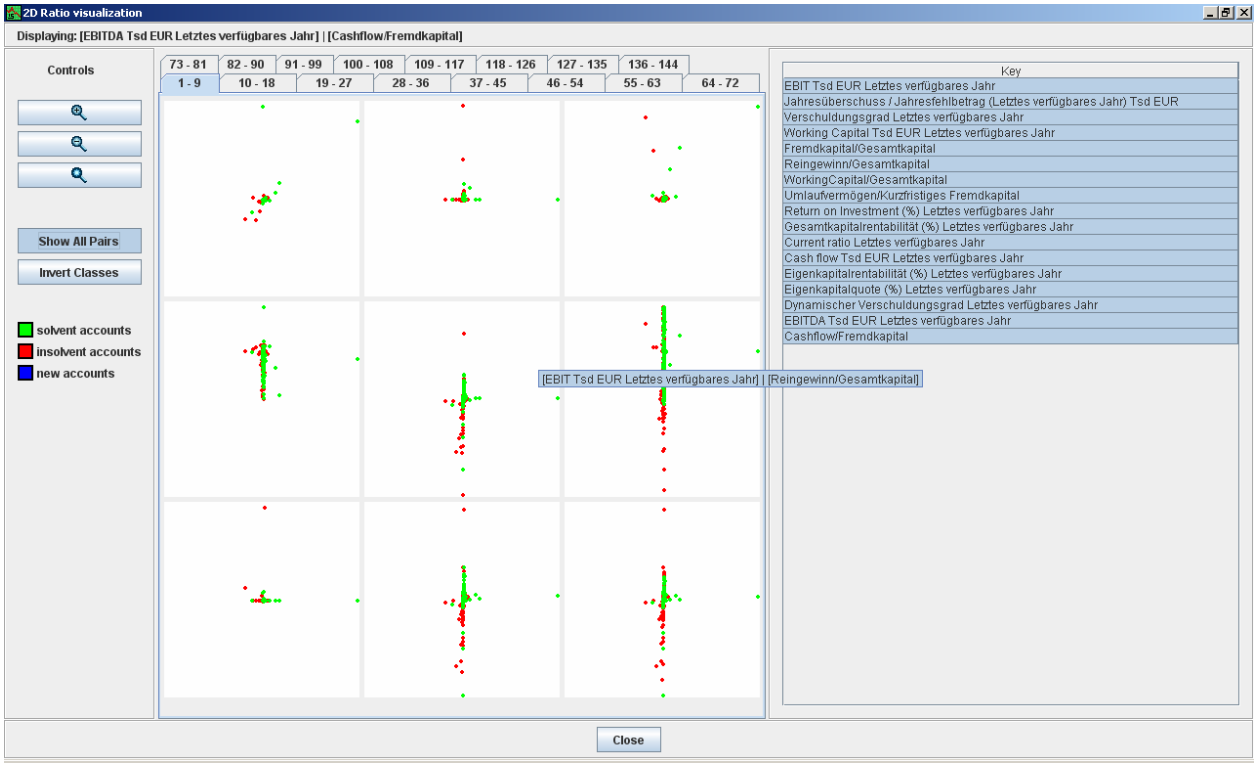


Abbildung 38: Visualization-Dialog: Show All Pairs Modus

Durch einen Klick auf den ToggleButton “Show All Pairs” werden alle möglichen Paare der aktuellen Kennzahlen-Auswahl angezeigt. Diese Funktion macht erst dann Sinn, wenn mehr als zwei Kennzahlen selektiert sind. Es werden entsprechende Reiter erzeugt, um alle möglichen Plots darstellen zu können.

Das Popupmenü eines jedes Plots gibt Aufschluss über die aufgetragenen Merkmale.

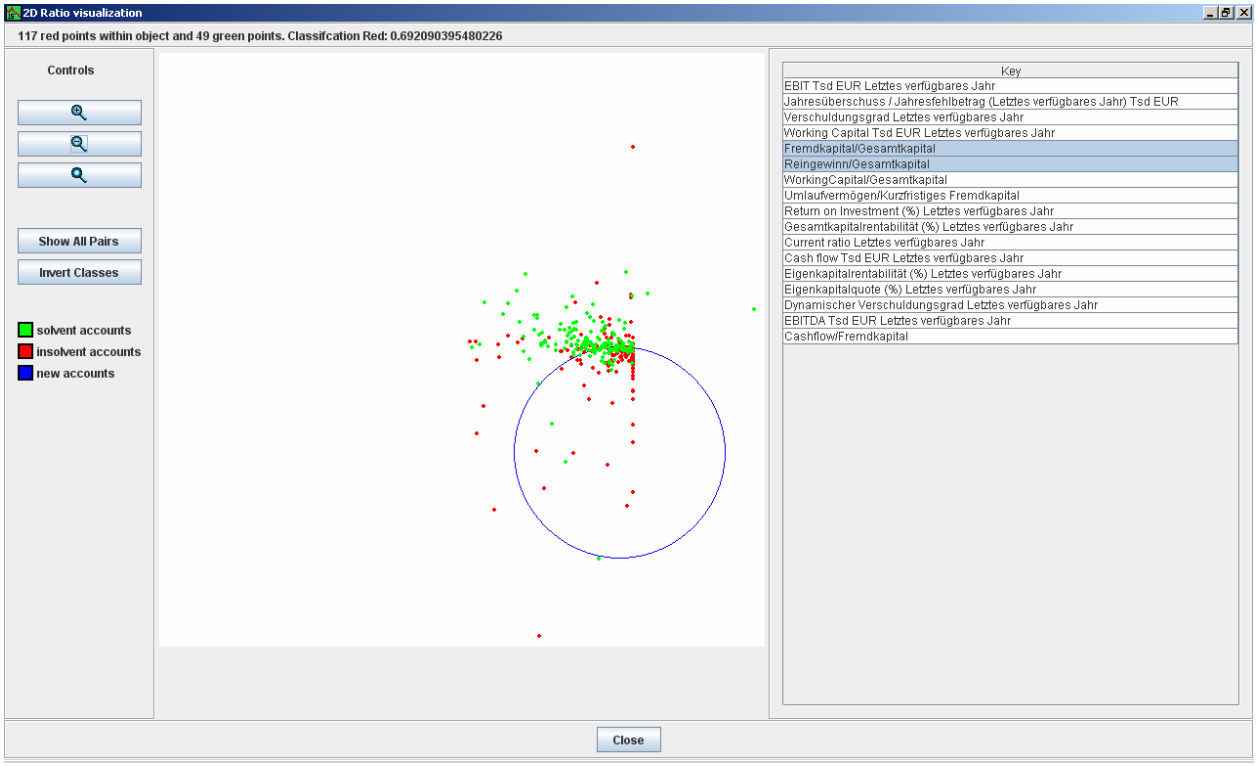


Abbildung 39: Visualization-Dialog: Manuelle Diskriminanzanalysen

Mit Hilfe von Drag-and-Drop auf der Plotfläche ist es möglich, einen Kreis in den Plot einzupassen, um einen Eindruck der Datenpunkte im Raum zu erhalten. Der Kreis dient dabei als Diskriminanzfunktion: Die roten Punkte innerhalb des Kreises sowie die grünen Punkte außerhalb des Kreises werden als korrekt klassifiziert angesehen. Diese Regel läst sich durch einen Klick auf den ToggleButton „Invert Classes“ umkehren. Die entstehenden Klassifikationsraten in der oberen Statusleiste werden dazu dynamisch berechnet. Dieser Vorgang ist mit der Zoomfunktionen kombinierbar und funktioniert ebenfalls im „Show All Pairs“ Modus.

A.2. Release und JavaDoc

InSolve ist vollständig in JavaDoc kommentiert und bietet detaillierte Informationen zur Entwicklung von Plugins innerhalb der erforderlichen Interface-Klassen. Die Klassen von InSolve stehen zum Download unter http://www.tcs.informatik.uni-goettingen.de/insolve bereit. Der Quellcode selbst ist auf Anfrage via E-Mail erhältlich (Thomas.Brodag@T-Online.de). Über diese Arbeit hinaus werden Supportleistungen angeboten im Hinblick auf Verbesserungsvorschläge oder auch zusätzliche Funktionswünsche.

A.2. Technische Hilfsmittel

- Für die Entwicklung des Java-Tools InSolve wurde *Netbeans IDE 5.0* eingesetzt.
- Scatterplots und Diagramme wurden unter *Matlab 7* erstellt.
- Formeln, Text und sonstige Grafiken wurden mit Hilfe von Microsoft Word 2003 erzeugt.

Literaturverzeichnis

[ABKT98] Bogan A.A., Thorn K.S. (1998) *"Anatomy of Hot Spots in Protein Interfaces"*

[D06] Daldrup, A. (2006) *„Rating, Ratingsystem und ratingbasierte Kreditrisikoquantifizierung"* Sub Göttingen eBook Series, Arbeitsbericht Nr. 17/2006

[AD98] Javed A. Aslam and Scott E. Decatur (1998) *"Specification and simulation of statistical query algorithms for efficiency and noise tolerance"* Journal of Computer and System Sciences, *56(2):191-208*

[AL87] Dana Angluin and Philip D. Laird (1978) *"Learning from noisy examples"* Machine Learning, 2(4):343-370

[ASEDB] Thorn K.S., Bogan A.A. (2001) *"ASEdb: a database of alanine mutations and their effects on the free energy of binding in protein interactions"* http://nic.ucsf.edu/asedb/results.php

[Asl95] J. A. Aslam. (1995) *"Noise Tolerand Algorithms for Learning and Searching"* PhD thesis, MIT

[B05] Brodag, T. (2005) *„Fremdgenerkennung mit Hidden-Markov-Modellen"* Mastarbeit ZFI Universität Göttingen

[BALL] BALL – Biochemical Algorithms Library, http://www.bioinf.uni-sb.de/OK/BALL/

[BBL02] Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi (2002) *"Model selection and error estimation*" Machine Learning, 48:85-113

[BBL04] O. Bousquet, S. Boucheron and G. Lugosi (2004) *"Introduction to statistical learning theory"* Advanced Lectures on Machine Learning, volume 3176 of Lecture Notes in Artificial Intelligence, pages 169-207, Springer, Heidelberg

[BDI] Justizportal des Bundes – Insolvenzbekanntmachungen, https://www.insolvenzbekanntmachungen.de/

[BE00] Blochwitz S., Eigermann J. (2000) *"Unternehmensbeurteilung durch Diskriminanzanalyse mit qualitativen Merkmalen"* zfbf 52 (Feb/2000)

[BKU96] Baetge Jörg, Kruse Ariane, Uthoff Carsten (1996) *„Bonitätsklassifikation von Unternehmen mit Neuronalen Netzen"* Wirtschaftsinformatik 39 (1996) 3, S. 273-281

[BM01] Peter L. Bartlett and Shahar Mendelson (2001) *"Rademacher and Gaussian complexities: Risk bounds and structural results"* In 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, volume 2111, pages 224-240, Springer, Berlin

[BM02] Peter L. Bartlett and Shahar Mendelson (2002) *"Rademacher and Gaussian complexities: Risk bounds and structural results"* J. Mach. Learn. Res., pages 463-482

[BS96] Baetge Jörg, Sieringhaus, Isabel (1996) *"Bilanzbonitäts-Rating von Unternehmen"* Handbuch Rating, hrsg. v. Büschgen, Hans E./Everling, Oliver, Wiesbaden, S. 221 – 249

[CR07] Creditreform (2007) „Insolvenzen 2007" G.I.B. Studie http://sanierungsberatung.gib-nrw.de/untersuchungen/studien

[DAFNE] Verband der Vereine Creditreform e.V. Bilanzdatenbank Dafne, http://www.creditreform-hannover.de/risiko_management/bilanz_auskuenfte/bilanzdatenbank_dafne.shtml

[Dec95] S. Decatur (1995) *"Efficient Learning from Faulty Data"* PhD thesis, Harvard University

[DFS97] Dietz J., Füser K., Schmidtmeier S. (1997) *„Neuronale Kreditwürdigkeitsprüfung im Konsumentenkreditgeschäft"* DBW 57 (1997) 4

[Fis99] P. Fischer (1999) *"Algorithmisches Lernen"* Teubner Verlag, Stuttgart, Leipzig

[GG00] Günther T., Grüning M. (2000) *„Einsatz von Insolvenzprognoseverfahren bei der Kreditwürdigkeitsprüfung im Firmenkundenbereich"* DBW 60 (2000)

[HKLW91] David Haussler, Michael J. Kearns, Nick Littlestone and Manfred K. Warmuth (1991) *"Equivalence of models for polynomial learnability"* Inf. Comput., 95(2):129-161

[Hau88] Haussler D. (1988) *"Quantifying inductive bias: AI learning algorithms and Valiant's learning framework"* Artificial Intelligence 36:177-221

[H06] Horváth P. (2006) „*Controlling*" München: Vahlen, ISBN 978-3-8006-3252-7

[HTF01] Hastie, T., Tibshirani, R., Friedman, J. (2001) *„The Elements of Statistical Learning"* Springer Verlag

[HWN03] Halperin I., Wolfson H., Nussinov R. (2003) *"Protein-protein interactions: coupling of structurally conserved residues and of hot spots across protein-protein interfaces"*

[JOONE] Joone – Java Object Oriented Neural Engine http://www.jooneworld.com/

[Kea98] Michael J. Kearns (1998) *"Efficient noise-tolerant learning from statistical queries"* J. ACM, 45(6):983-1006

[KEN95] Kendall, M. (1980) *"Multivariate Analysis, 2nd Edition"* London (1980)

[KMN05] Keskin O., Ma B., Nussinov R. (2005) *"Hot Regions in Protein-Protein Interactions: The organization and contribution of structurally conserved Hot Spot residues"*

[Kol01] V. Koltchinskii (2001) *"Rademacher penalties and structural risk minimization"* IEEE Transactions on Information Theory, 47:1902-1914

[LS98] Leker J., Schewe G. (1998) *„Beurteilung des Kreditausfallrisikos im Firmenkundengeschäft der Banken"* zfbf 50 (Okt/1998)

[LSVM] LIBSVM – A Library for Support Vector Machines http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[KR90] Kaufmann, L., Rousseeuw, P.(1990) *„Finding Groups in Data"* Wiley, New York

[MAS00] P. Massart (2000) *"Some applications of concentration inequalities to statistics"* Anales de la Faculté des Sciences de Toulouse, volume spécial dédié à Michel Talagrand: 245-303

[McD89] C. McDiarmid (1989) *"On the method of bounded differences"* In J. Siemans, editor, Surveys in Combinatorics, volume 141 of London Math. Soc. Lecture Notes, pages 149-188, Cambridge University Press

[Men02] S. Mendelson (2002) *"Rademacher averages and phases transitions in glivenko-cantelli classes"* IEEE Trans. Inf. Theory, 48:1977-1991

[MMT71] Maurice M. Tatsuoka (1971) "*Multivariate Analysis: Techniques for Educational and psychological Research"* John Wiley & Sons

[NICO02] Nicolas, P., Bize, L., Muri, F., Hoebeke, M., Rodophe, F., Dusko Ehrlich, S., Prum, B., Bessieres, P. (2002) *„Mining Bacillus Subtilis chromosome heterogeneities using hidden Markov models"* Oxford Journals Online

[NPTB04] Pfeifer, N., Brodag, T. (2004) *„Kernbasierte Verfahren zur TIS-Erkennung bei Prokaryoten"* Bachelorarbeit am Institut für numerische und angewandte Mathematik

[NW04] Werner, N. (2004) *"Kernbasierte Lernmethoden zur Unterstützung der Insolvenzprognose"* Diplomarbeit Universität Göttingen

[PDB] PDB – RCSB Protein Database http://www.rcsb.org/pdb/home/home.do

[PS07] Perridon L., Steiner M. (2007) „*Finanzwirtschaft der Unternehmung"* München: Vahlen, ISBN 978-3-8006-3359-3

[PWDM64] MacNaughton-Smith, P., Williams, W.T., Dale M.B., Mockett, L.G. (1964) *„Dissimilarity analysis: A new technic of hierarchical subdivision"* Nature, 202:1034-1035

[PW98] Wolf H. P. (1998) *"Maximum-Likelihood-Schätzung"* Universität Bielefeld http://www.wiwi.unibielefeld.de/~naeve/lehre/material_spezifisch/stat9798/like/index.html

[SH01] Schneider, Willy; Hennig, Alexander (2001) *„Kennzahlen Marketing und Vertrieb"* Landsberg am Lech: MI-Verlag, ISBN 978-3-4783-7440-8

[S03] Stryer L., Tymoczko L., Berg M., Jeremy M. (2003) *"Biochemie"* Spektrum Akademischer Verlag, Auflage 5

[SS02] B. Schölkopf and A. J. Smola (2002) *"Learning with Kernels"* MIT Press

[U97] Uthoff, Carsten (1997) *„Erfolgsoptimale Kreditwürdigkeitsprüfung auf der Basis von Jahresabschlüssen und Wirtschaftsauskünften mit Künstlichen Neuronalen Netzen"* M u. P-Schriftenreihe für Wissenschaft u. Forschung, ISBN-10: 3476460509

[Val84] L. Valiant (1984) *"A theory of learnability"* Communications of the ACM, 27(11):1134-1142

[W08] Wolowski, V. (2008) *"Computational analysis of protein-protein complexes related to knowledge-based predictions of interaction"* Diplomarbeit Universität Regensburg

[WEL87] Welsh, D. (1987) *„Codes and Cryptographie"* Oxford University Press

[WB08] Waack, S., Brodag, T. (2008) *„A generalized Noise-Model for PAC-Learning"* to be submitted 05/2008

[XL04] Li X., et al (2004) *"Protein-Protein Interactions: Hot Spots and structurally conserved residues often locate in complemented pockets that pre-organized in the unbound states: Implications to docking"*

[ZGK06] Zomorodian A., Guibas L., Koehl, P. (2006) **"***Geometric filtering of pairwise atomic interactions applied to the design of efficient statistical potentials"* Computer Aided Geometric Design
Volume 23, Issue 6, Special issue: Applications of geometric modeling in the life sciences, Pages: 531 - 544

Danksagung

Mein besonderer Dank gilt Herrn Professor Dr. Waack für seine umfangreiche Betreuung, für wertvolle Hinweise und die Möglichkeit, ein interessantes Thema im Rahmen dieser Dissertation bearbeiteten zu dürfen.

Ebenso bedanke ich mich bei Professor Dr. Carsten Damm für seine Bereitschaft als Korreferent tätig zu werden.

Mein Dank gilt außerdem Herrn Professor Dr. Schumann für die wertvollen Hinweise bezüglich der Rahmenbedingungen im Bereich der Betriebswirtschaft.

Weiterhin danke ich Herrn Dr. Rainer Merkl und Vincent Wolowski für anregende und hilfreiche Diskussionen bezüglich der PDB-Datenaufbereitung sowie Protein-Protein-Wechselwirkungen.

Ich bedanke mich außerdem bei Dr. Andre Daldrup für wertvolle Anregungen und informative Literaturhinweise.

Ich danke Berthold Vöcking für wertvolle Denkanstöße bezüglich Reduktionsalgorithmen.

Außerdem gilt mein Dank Nadine Blinn und Florian Lewin für anregende Diskussionen bezüglich der Insolvenzvorhersage.

Besonders bedanke ich mich bei meinen Eltern und meiner Großmutter für ihre Unterstützung und Geduld sowie wertvolle Hinweise bezüglich Stil und Ausdruck.

Göttingen, den 16.04.2008 Thomas Brodag

Lebenslauf

Persönliche Daten

Name:	Thomas Brodag
Geburtsdatum:	01.09.1981
Geburtsort:	Hann. Münden
Eltern:	Wolf-Dietrich und Margot Brodag

Schulausbildung

1987 – 1992	Grundschule Hedemünden
1992 – 1994	OS II Hann. Münden
1994 – 2001	Grotefend Gymnasium Hann. Münden
2001	Abitur

Universitäre Ausbildung

WS 01/02	Beginn des Studiums der angewandten Informatik mit Anwendungsfach Bioinformatik
14.10.2004	Bachelor of Science
20.10.2005	Master of Science
21.10.2005	Beginn der Promotion am Institut für numerische und angewandte Mathematik bei Professor Dr. Waack

Berufstätigkeit

22. 10. 2005 – 02.01.2007	Freier Mitarbeiter bei der CadCabel AG in Hohengandern
02. 01. 2007	Einstellung als Consultant bei der Sycor GmbH in Göttingen

Göttingen, 16.04.2008