Interpretation regelbasierter Zwischenkonzepte von Autoencodern

Master-Thesis von Frederik Buss-Joraschek
September 2016

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Knowledge Engineering Group

Interpretation regelbasierter Zwischenkonzepte von Autoencodern

Vorgelegte Master-Thesis von Frederik Buss-Joraschek

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Eneldo Loza Mencía

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. September 2016

(Frederik Buss-Joraschek)

Zusammenfassung

Neuronale Netze sind zwar ausgesprochen leistungsfähig, erzeugen dabei jedoch ein Modell, das schwierig zu erklären ist. Viele Untersuchungen haben bereits betrachtet, wie dieses Problem durch Regelextraktion verringert werden kann. In nahezu allen Fällen lag dabei Fokus auf der Umsetzung, aber weniger der Interpretation der erlernten Konzepte.

In dieser Masterarbeit wurde im Rahmen einer Fallstudie ein Autoencoder für einen Datensatz trainiert und anschließend interpretiert. Dafür wurden sämtliche Neuronen im Hidden Layer einzeln betrachtet und eine Beschreibung des umgesetzten Konzeptes bestimmt. Anschließend wurde betrachtet, inwiefern sich die erlernten Konzepte auf neue Aufgaben anwenden lassen. Dafür wird die Technik der *separaten Klassifikation* vorgestellt, die einen Regellerner basierend auf den Daten des Hidden Layers trainiert und damit eine Klassifikation durchführt. Ein zweiter Ansatz untersucht ebenfalls die Klassifikationseignung. Die zuvor erlernten Gewichte des Autoencoders werden als Initialgewichte eines neuen neuronalen Netzes verwendet. Dies wird dann für eine Klassifikationsaufgabe erneut trainiert. Für alle Techniken wurden auch untersucht, welche Unterschiede sich zwischen normalisierten und diskretisierten Eingabedaten ausbilden.

Die Untersuchung hat gezeigt, dass Neuronen nur sehr selten ein einziges Feature modellieren. Häufig deckt ein Neuron grundverschiedene Aspekte ab, die nur durch Kombination für eine Aussage genutzt werden können. Dennoch sind die erlernten Konzepte für eine Klassifikation gut geeignet. Selbst als das Hidden Layer eine Kompression von nahezu 90% durchführte, betrug die Klassifikationsleistung noch etwa 80% und war damit nur leicht schlechter als das beste Ergebnis. Beim Gewichtstransfer konnte beobachtet werden, dass in wenigen Fällen Konzepte ohne Änderung in das neue Netz übernommen wurden und dabei auch relevant für die Klassifikation waren. In allen Versuchen hat sich gezeigt, dass ein Netz schwieriger zu interpretieren ist, wenn die Daten normalisiert und nicht diskretisiert wurden.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Künstliche Intelligenz ist derzeit ein Thema mit vielen Anwendungsgebieten. Besonders die Methode der künstlichen Neuronalen Netzen (KNN) hat dabei eine große Bedeutung. Diese können Wissen, ähnlich wie das menschliche Gehirn, über Verknüpfungen von Neuronen modellieren. Dadurch ist diese Technik äußerst flexibel und kann in unterschiedlichsten Bereichen eingesetzt werden. Gleichzeitig kann sie dabei unter Umständen bessere Leistung als andere Methoden zeigen.

Einer der jüngsten Erfolge dieser Netze ist Google DeepMind mit AlphaGo gelungen. Es handelt sich dabei um ein Programm, welches es erstmals geschafft hat einen professionellen Go-Spieler in einem Turnier zu besiegen. Dabei wurden unter anderem verschiedene neuronale Netze eingesetzt, um mögliche Züge und deren Wert zu bestimmen.

Weitere Anwendungsgebiete von KNNs reichen von simplen Klassifikationsaufgaben oder Mustererkennung bis hin zu Autoencodern. Ein solcher Encoder ist ein spezielles KNN, das als Ausgabe die Eingabe zu rekonstruieren versucht. Dabei soll nicht die Identität sondern eine effiziente Kodierung für die Daten gefunden und gelernt werden. Dies wird beispielsweise genutzt um Daten zu komprimieren oder verrauschte Daten zu bereinigen.

Die Flexibilität von KNNs führt dazu, dass gelernte Modelle äußerst komplex in ihrer Repräsentation sind. Wird das Netz vergrößert steigt die Komplexität exponentiell an. Aufgrund der Größe wird es damit für Menschen unmöglich ein gelerntes Model zu interpretieren. In manchen Bereichen, wie z.B. medizinischen Anwendungen, ist es jedoch unerlässlich ein tieferes Verständnis für die gelernten Konzepte zu entwickeln. Nur so kann geprüft werden, ob Fehler existieren die möglicherweise eine sichere Benutzung verhindern. Eine Interpretation ist auch deshalb interessant, weil das KNN möglicherweise bisher unbekannte Zusammenhänge in den Daten gefunden hat, die an andere Stelle verwendet werden könnten.

Eine Möglichkeit diese Problem zu lösen ist es, dass Wissen in eine andere Form zu übertragen. Für die Transformation kommt es darauf an das KNN möglichst genau abzubilden und keinen zusätzlichen Fehler zu erzeugen. Häufig werden als Zielform Regeln gewählt. Diese sind für Menschen im Allgemeinen gut verständlich und bieten dabei gleichzeitig eine hohe Ausdrucksstärke.

In der Vergangenheit wurden bereits diverse Untersuchungen durchgeführt wie Regeln aus einem KNN extrahiert werden können. Die erzeugten Regeln wurden dabei mit Metriken auf ihre generelle Interpretierbarkeit untersucht. Es gibt jedoch nur sehr wenige Untersuchung, die eine konkrete Interpretation der extrahierten Regeln durchführen. Es ist gut vorstellbar, dass obwohl die extrahierten Regelmengen klein und die Regeln kurz sind, diese semantische Widersprüche aufweisen. Darüber hinaus könnten sie auch Wissen repräsentieren das für Menschen selbst bei näherer Betrachtung keinen erkennbaren Zusammenhang hat.

1.2 Zielsetzung

Diese Arbeit beschäftigt sich damit eine konkrete, regelbasierte Interpretation eines Autoencoders anhand eines Beispieldatensatzes durchzuführen. Dabei liegt der Hauptfokus auf den Konzepten die im Hidden Layer gelernt wurden. Dabei wird nicht nur das Hidden Layer als Ganzes bewertet, sondern die einzelnen Neuronen werden getrennt analysiert. Es wird ermittelt, ob eine Korrelation zwischen der Qualität des Konzepts eines Neurons und dessen Verwendungshäufigkeit in der Ausgabeschicht vorliegt. Weiterhin wird untersucht, ob die ausgebildeten Konzepte auf neue Aufgaben übertragen werden können. Dafür wird zum einen untersucht inwieweit die Konzepte des Autoencoders es erlauben eine

Klassifikationsaufgabe durchzuführen. Ein zweiter Aspekt beschäftigt sich mit der Frage, ob Konzepte erhalten bleiben, wenn die Gewichte als Initialisierung für ein neues neuronales Netz genutzt werden. Neben dem semantischen Aspekt wird auch jeweils der Einfluss auf die Klassifikationsleistung betrachtet.

Mit diesen Versuchen soll ein tieferes Verständnis der Konzepte im Hidden Layer, sowie deren Möglichkeiten erreicht werden. Diese Arbeit soll außerdem eine Grundlage für weitere Untersuchung schaffen, indem sie Techniken aufzeigt, die für neue Anwendungen genutzt werden können.

1.3 Aufbau der Arbeit

In Kapitel 2 werden zunächst einige grundlegende Begriffe und Konzepte aus dem Bereich der Regeln und der künstliche Neuronale Netze vorgestellt. Ebenfalls wird dort der verwendete Datensatz und dessen Attribute vorgestellt. Nachfolgend werden in Kapitel 3 existierende Ansätze vorgestellt, die sich mit der Extraktion von Regeln beschäftigen. Auch eine Vorstellung von anderen Analysetechniken sowie beispielhaft eine konkrete semantische Interpretation werden dort präsentiert. Eine Vorstellung der eigenen Analyse findet in Kapitel 4 statt. Dieses beschreibt den eigenen Regelextraktionsalgorithmus, das Vorgehen zur semantischen Analyse sowie zwei weitere Techniken die das Hidden Layer für eine neue Aufgabe nutzen. Die Evaluation dieser Techniken findet in Kapitel 5 statt. Dort findet auch die semantische Analyse statt, die für jedes Neuron eine Beschreibung in wenigen Worten anbietet. Daran anschließend folgt in Kapitel 6 die Diskussion der Evaluation sowie eine Bewertung der Ergebnisse im Kontext des Problemfeldes. Weiterhin wird aufgezeigt, welche Möglichkeiten für zukünftige Untersuchungen existieren und sinnvoll sind.

2 Grundlagen

2.1 Künstliche Neuronale Netze

Künstliche neuronale Netze haben eine Funktionsweise ähnlich dem des menschlichen Gehirns. Wissen wird dabei durch Verknüpfungen von Neuronen dargestellt. Der nachfolgende Abschnitt zeigt auf, wie solche Neuronen und deren Verknüpfungen repräsentiert werden. Es wird ebenfalls demonstriert, wie ein solches Netz aufgebaut und verwendet wird. Weiterhin wird auf die speziellen Eigenschaften von Autoencodern eingegangen. Innerhalb dieser Arbeit sind alle Neuronen und Netze auch ohne Zusatz als künstlich und nicht biologisch zu verstehen.

2.1.1 Aufbau

Die grundlegendste Komponente in einem KNN ist ein Neuron. Ein Neuron besteht aus vier verschiedenen Bestandteilen: Eingabegewichte, Übertragungsfunktion, Bias-Gewicht und der Aktivierungsfunktion. Deren Zusammenhang ist in Grafik 2.1 dargestellt. Als Eingabe erhält ein Neuron einen N-dimensionalen Eingabevektor $x \in \mathbb{R}^N$.

Im ersten Schritt wird die Netzaktivität des Neurons berechnet. Hierfür gibt es verschiedene Möglichkeiten, die sich in ihrer Komplexität und Funktion unterscheiden. In den meisten Fällen werden die Eingabewerte jedoch lediglich gewichtet aufsummiert. Die Gewichte werden durch einen Vektor w definiert, der die gleiche Dimension wie x hat.

$$n = \sum_{i=1}^{N} w_i x_i$$

Diese Netzaktivität wird zusammen mit einem Bias auf die Aktivierungsfunktion f angewendet, die damit die finale Ausgabe des Neurons berechnet.

$$o = f(n + b)$$

Die Wahl der Aktivierungsfunktion hängt von den gewünschten Eigenschaften des Neurons in Bezug auf die Ausgabewerte ab. Grundsätzlich sind sowohl lineare als auch nicht-lineare Funktionen möglich. Typischerweise werden Funktionen wie *tanh* oder *sigmoid* verwendet um eine S-förmige Verteilung zu erhalten. Der Bias-Wert wird verwendet um den Aktivierungswert und damit die Ausgabe des Neurons zu verschieben. Bei Verwendung von *sigmoid* als Aktivierungsfunktion kann ein hoher negativer Bias sicherstellen, dass das Neuron nur bei einem hohen Aktivierungswert eine 1 ausgibt. Da so die Grenze zur Aktivierung justiert wird, wird der Bias auch Schwellenwert genannt.

Grundsätzlich ist es durch Wahl der Aktivierungsfunktion möglich jeden beliebigen Ausgabebereich zu ermöglichen. In den meisten Fällen werden jedoch Neuronen verwendet, die eine Ausgabe im Bereich $[0, 1]$ erzeugen. In einem solchen Fall wird ein Neuron wenn es einen Wert nahe 1 ausgibt als „feuernd“ oder „aktiviert“ bezeichnet.

Durch Komposition von mehreren Neuronen entsteht ein neuronales Netz. Dafür wird die Ausgabe von einem oder mehreren Neuronen als Eingabe für ein anders Neuron verwendet. Um die Regelextraktion zu vereinfachen werden in dieser Arbeit nur geschichtete Feedforward-Netze verwendet. Bei diesen werden die Neuronen als Gruppen (Schichten) angeordnet. Alle Neuronen einer Gruppe können dabei als Eingabe nur die Ausgabe der Neuronen der vorherigen Schicht verwenden. Wird das Netz als Graph betrachtet ist es damit zyklenfrei.

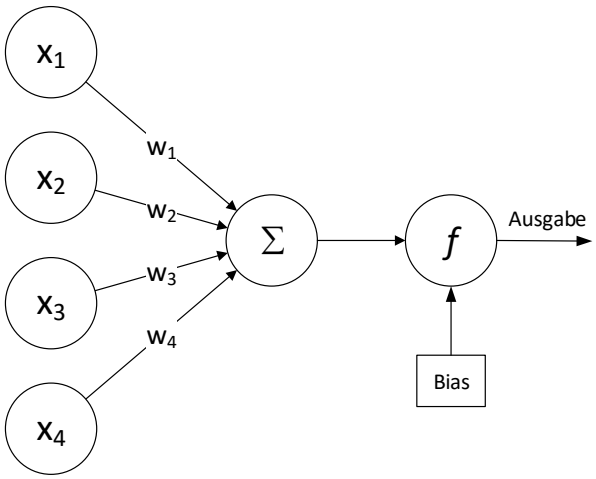

Abbildung 2.1: Einzelnes Neuron

Ein KNN besteht immer aus mindestens zwei Schichten, der Ein- und Ausgabeschicht. In Grafik 2.2 ist ein Beispiel für ein neuronales Netz zu sehen. Konkret handelt es sich dabei um einen Autoencoder, der in Abschnitt 2.1.2 noch näher beschrieben wird. Die Kreise stellen die einzelnen Neuronen dar und die Pfeile die gerichteten Verbindungen zwischen diesen. Das dargestellte Netz besitzt fünf Schichten, wobei die Schicht auf der linken Seite die Eingabeschicht ist und auf der rechten Seite die Ausgabeschicht. Die Schichten, die sich dazwischen befinden, werden Hidden-Layer bzw. verborgene Schichten genannt, da sie außerhalb des Netzes nicht beobachtbar sind. Die Neuronen darin werden als Hidden Neurons oder Hidden Units bezeichnet.

2.1.2 Autoencoder

Ein Autoencoder ist ein neuronales Netz, das als Ausgabe die Eingabe zu rekonstruieren versucht. Damit das Netz nicht für alle Neuronen die Identität lernt, wird die Anzahl der Neuronen im Hidden-Layer dabei geringer gewählt als Ein- und Ausgabeneuronen vorhanden sind. Dadurch lernt das Netz eine effiziente Kodierung der Daten, sowie deren Dekodierung. Dies kann z.B. zur Kompression von Datensätzen verwendet werden. Wie in Grafik 2.2 dargestellt, besteht ein Autoencoder aus zwei grundlegenden Komponenten: Dem Encoder und dem Decoder. Der Encoder führt die Kompression durch, wohingegen der Decoder das Ursprungssignal wiederherstellt. Beide Komponenten werden gleichzeitig gelernt, können aber durch Extraktion der Gewichte auch getrennt verwendet werden.

Eine andere Anwendung ist es Daten zu entrauschen. Dafür wird bereits beim Training auf der Eingabeseite absichtlich Rauschen eingefügt, um das Netz anzuregen kleine Abweichungen zu ignorieren und nur die primären Bestandteile zu rekonstruieren. Solche Autoencoder werden „Denoising Autoencoder“ genannt und finden häufig in der Signalverarbeitung Anwendung.

Um größere Reduktionen zu erreichen können wie in Grafik 2.2 mehrere Autoencoder verwendet werden. Dafür wird ein Autoencoder von einem weiteren umklammert. Dies kann mehrfach wiederholt werden. Eine solche Anwendung wird als gestapelte Autoencoder bezeichnet und erlaubt es komplexere Kodierungen zu lernen.

2.2 Regel Lerner

Regeln gehören zu den am einfachsten verständlichen Methoden, um Konzepte zu repräsentieren. Sie modellieren einen Zusammenhang zwischen verschiedenen Attributen. Regeln haben üblicherweise ei-

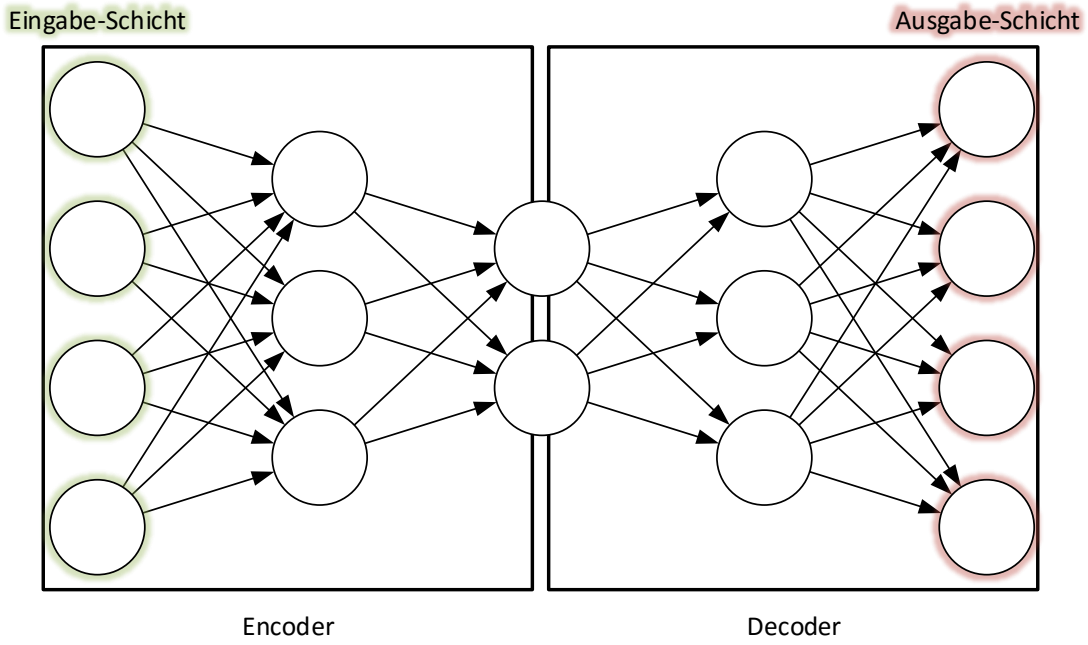

Abbildung 2.2: Mehrschichtiger Autoencoder

ne Prämisse, die die Anwendbarkeit der Regeln definiert sowie eine Schlussfolgerung, die das Ergebnis beschreibt. Die Komplexität und Ausdrucksstärke von Regeln unterscheidet sich je nach gewählter Form. Die in dieser Arbeit verwendeten „Wenn-Dann“, auch „IF-THEN“ genannten, Regeln gehören mit zu den einfachsten. Neben diesen gibt es beispielsweise die Sequenzenkalküle, die bei automatischen Theorembeweisern oft Anwendung finden und durch eine Vielzahl von Quantoren und Operatoren eine erhöhte Komplexität haben. Die „Wenn- Dann“ Regeln hingegen bestehen lediglich aus Vergleichen und Konjunktionen. Ein simples Beispiel wie „**Wenn** FarbeBanane= Grün **Dann** ZustandBanane= unreif“ zeigt das diese ein vergleichsweise natürliches Verständnis der Zusammenhänge vermitteln.

Die generelle Form dieser Regeln ist:

$$\text{Wenn } b_1 \wedge b_2 \wedge \cdots \wedge b_n \text{ Dann } v$$

Dabei sind b_1 bis b_n die Bedingungen der Prämisse, die sich aus einem Attribut, einem Operator und einem möglichen Wert des Attributs zusammensetzt. Es ist nicht notwendig, dass sämtliche Attribute eines Datensatzes Verwendung in einer Regel finden. Darüber hinaus wird von vielen Lernern ein Attribut lediglich in einer einzigen Bedingung und nicht mehreren verwendet. Typische Vergleichsoperatoren die bei Regeln verwendet werden sind $\leq, \geq$ oder $=$. Bei obigem Beispiel mit der Banane ist damit die einzige Bedingung „FarbeBanane=Grün“. Die Schlussfolgerung der Regeln wird durch v dargestellt und definiert die Vorhersage. Wie die Vorhersage aufgebaut ist hängt von der Art des verwendeten Lerners ab. Bei Regressionsproblemen könnte die Regel eine Formel für den Vorhersagewert liefern: „Bonus = Gehalt $\cdot$ 0.5“. Bei Klassifikationsaufgaben beschreibt sie hingegen üblicherweise einen konkreten Wert des zu klassifizierenden Attributs. Beim zuvor verwendeten Bananenbeispiel wäre das damit „ZustandBanane=unreif“.

Wenn alle definierten Bedingungen einer Regel erfüllt sind, findet die angegeben Vorhersage statt. In diesem Fall wird davon gesprochen, dass die Regel das Beispiel „abdeckt“. Sollte mindestens eine Bedingung falsch sein, so kann mit dieser Regel keine Aussage getroffen werden. Attribute die in keiner Bedingung vorkommen sind für diese Regel irrelevant und haben keinen Einfluss auf das Ergebnis.

Hat eine Regel überhaupt keine Bedingungen, so wird diese die „Default-Regel“ genannt, da sie unabhängig von Eingaben eine Standardvorhersage liefert. Ein Klassifizierer der lediglich diese Regel besitzt und damit die größte Klasse vorhersagt wird häufig als Vergleichswert für die Qualität von anderen gelernten Modellen verwendet. Die Default Regel ist die einfachste, gültige Regel.

In nahezu allen Fällen ist es unmöglich eine gute Klassifikationsleistung mit nur einer einzigen Regel zu erreichen. Dementsprechend verwenden Regellerner im Normalfall mehrere Regeln, die sich ergänzen. Dadurch kann der Fall auftreten, dass ein Beispiel durch mehrere Regeln zeitgleich abgedeckt wird. Wenn alle Regeln die gleiche Vorhersage treffen ist dies kein Problem, unterscheiden sie sich jedoch muss dies aufgelöst werden. Dafür gibt es verschiedene Techniken. Zum einen können die Regeln nach Priorität geordnet werden und die Regel mit der höchsten Priorität bestimmt die Vorhersage. Eine andere Möglichkeit ist es die Situation wie eine Abstimmung zu betrachten, bei der die Vorhersage, die durch die meisten Regeln getroffen wird gewinnt. Es ist sinnvoll, dass die Default-Regel dabei die geringste Priorität hat beziehungsweise als letztes angewendet wird.

Werden in der Regelmenge für verschiedene Attribute Vorhersagen getroffen, handelt es sich um Multiklassen-Klassifizierer. In dieser Arbeit werden jedoch nur Regelmengen betrachtet, die für ein einziges Attribut pro Klassifizierer eine Vorhersage treffen.

2.2.1 Ripper

Viele Regellerner, wie der überaus bekannte C4.5 bzw. dessen Regel Version, lernen zunächst ein komplexes Konzept, um dieses anschließend zu vereinfachen. Das Ziel davon ist es die Regeln kurz zu halten und damit ein zu starkes Anpassen an die Trainingsdaten zu vermeiden. Fürnkranz und Widmer [FW94] haben mit IREP einen Algorithmus präsentiert, der bereits während des Lernvorgangs die Regeln vereinfacht und dafür kein Zwischenkonzept mehr lernen muss. Die Funktionsweise folgt dabei dem „separate-and-conquer“ Prinzip. Danach wird zunächst eine Regel gesucht und falls diese akzeptabel ist wird sie der Regelmenge hinzugefügt. Alle dadurch abgedeckten Beispiele werden aus der Trainingsmenge entfernt. Dies wird wiederholt bis keine positiven Beispiele mehr vorhanden sind.

Der in dieser Arbeit verwendete Ripper Algorithmus [Coh95] basiert auf IREP und verbessert ihn bezüglich der Klassifikationsleistung. Der Pseudocode von Ripper ist in Listing 2.1 dargestellt und hat große Ähnlichkeit mit IREP. Die wesentlichen Unterschiede die eingeführt wurden sind ein leicht verändertes Pruning Verhalten sowie eine Änderung des Abbruchkriteriums.

Für das Training von neuronalen Netzen werden häufig große Trainingsmengen verwendet, diese sollen auch zum Training der Regeln benutzt werden. Viele Regellerner haben den Nachteil, dass sie nur schlecht mit großen Datenmengen skalieren, wenn diese Rauschen enthalten. Wie Cohen gezeigt hat, können IREP beziehungsweise Ripper bei deutlich geringerem Zeiteinsatz eine zu C4.5 vergleichbare Klassifikationsleistung erreichen.

In Abschnitt 4.1 wird näher darauf eingegangen, wie die Eingabedaten des Regellerners erstellt werden. Zu diesem Zeitpunkt reicht es zu wissen, dass die Eingabedaten gesampelt werden und daher unter Umständen Rauschen enthalten. Dementsprechend ist Ripper für die Bedürfnisse dieser Arbeit besser geeignet als andere Regellerner.

2.2.2 Overfitting und Pruning

Für Regellerner ist es leicht eine perfekte Regelmenge zu finden, die sämtliche Beispiele in den Trainingsdaten korrekt klassifizieren kann. Dafür ist es lediglich notwendig, für jedes Beispiel eine eigene Regel anzulegen. Es ist offensichtlich, dass dies große und komplexe Regelmengen zur Folge hat und äußerst schlecht skaliert. Diese starke Anpassung auf die Trainingsdaten wird Overfitting genannt.

Eine schöne Definition hierfür hat Mitchell präsentiert:

```
1  input:
2          Pos = Positive Examples
3          Neg = Negative Examples
4
5  begin
6          Ruleset := ∅
7          while Pos ≠ ∅ do
8                  /* grow and prune a new rule */
9                  split (Pos, Neg) into (GrowPos, GrowNeg) and (PrunePos, PruneNeg)
10                 Rule := GrowRule(GrowPos, GrowNeg)
11                 Rule := PruneRule(Rule, PrunePos, PruneNeg)
12                 if the error rate of Rule on (PrunePos, PruneNeg) exceeds 50% then
13                         return Ruleset
14                 else
15                         add Rule to Ruleset
16                         remove examples covered by Rule from (Pos, Neg)
17                 endif
18         endwhile
19         return Ruleset
20 end
```

Listing 2.1: Ripper [Coh95]

Definition 1. *Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that h has a smaller error than h' on the training examples, but h' has a smaller error than h on the entire distribution of instances.[Mit97]*

Die Überanpassung auf die Trainingsdaten kann ein erhebliches praktisches Problem darstellen. Ein überangepasstes Modell kann nur schlecht Wissen generalisieren und auf neue Situationen anwenden. Im obigen Extremfall könnte kein Beispiel, das nicht in den Trainingsdaten vorhanden war, sinnvoll klassifiziert werden. Das neue Beispiel würde zwangsläufig der Vorhersage der Default-Regel folgen. Auch in weniger extremen Situationen kann, durch beispielsweise Fehler in den Daten, dies zu unnötig strengen Regeln führen.

Um diesem Problem vorzubeugen, wird dem Lernsystem üblicherweise ein Fehlerspielraum vorgegeben. Der Lerner produziert Regeln, die einen gewissen Fehler machen dürfen, dabei aber so generell wie möglich sein sollen. Welche Werte hierfür ideal sind ist situationsabhängig. Werden einfache Regeln bevorzugt, ist es förderlich einen größeren Fehler zuzulassen. Es existieren auch Ansätze, die den Fehlerspielraum dynamisch anpassen können und dafür Kriterien wie die Komplexität der Regelmenge bewerten.

Um Pruning durchzuführen, gibt es zwei wesentliche Ansätze. Beim *Pre-Pruning* wird bereits beim Erstellen der Regel frühzeitig aufgehört wenn Regeln zu spezifisch werden. Hingegen wird beim *Post-Pruning* zunächst ein Modell gelernt, das sehr wenige Fehler macht. In einem zweiten Schritt werden dann wieder Bedingungen entfernt und das Modell vereinfacht. Häufig wird auch eine Kombination beider Ansätze genutzt.

Bei dem in dieser Arbeit verwendeten Lerner wird die Pruningstärke durch einen Parameter „-N“ bestimmt. Dieser definiert, wie viele Beispiele eine Regel mindestens abdecken muss, damit noch eine Bedingung hinzugefügt werden darf. Dadurch können zwar unter Umständen Regeln mit größerem Fehler entstehen, aber es erlaubt auch eine sehr anschauliche Spezifizierung der Pruningstärke. Es bleibt damit notwendig bei der Interpretation der Regeln immer auch den Fehler zu beachten.

2.3 Datensätze

Dieser Abschnitt befasst sich damit, wie Datensätze vorverarbeitet werden, damit sie zum Lernen geeignet sind. Dabei wird einerseits beschrieben welche Techniken angewendet werden müssen bzw. sollten um Eingabedaten für neuronale Netze anzupassen. Andererseits wird darauf eingegangen wie die Leistung eines Lerners beurteilt werden kann.

2.3.1 Nominale Attribute

Nominale Attribute sind Eigenschaften, die einen diskreten Wert aus einem endlichen, abzählbaren Bereich annehmen. Ein Attribut, das nur zwei Werte annehmen kann wird binär genannt.

Auch wenn es in manchen Fällen semantisch möglich wäre, ist keine Ordnung für die Elemente definiert. Dies ist ein Problem wenn ein KNN mit einem nominalen Attribut verwendet werden soll, da die Neuronen einen kontinuierlichen Eingabewert benötigen. Im Spezialfall des binären Attributes ist die Lösung einfach. Hierbei wird für jeden der beiden Werte eine numerische Konstante festgelegt. Im Normalfall werden die Werte 0 und 1 verwendet.

Im Fall von mehr als zwei möglichen Werten kann auf die *One-Hot* Kodierung zurückgegriffen werden. Hierbei wird eine Binärisierung des Attributs durchgeführt. Das bedeutet, dass für jeden möglichen Wert ein neues binäres Attribut erzeugt wird, das das Vorhandensein oder dessen Abwesenheit anzeigt. Das originale Attribut wird anschließend entfernt. Beispielsweise wird $\{Farbe = \{\text{Rot}, \text{Grün}, \text{Blau}\}\}$ zu $\{Rot = \{\text{ja}, \text{nein}\}, Grün = \{\text{ja}, \text{nein}\}, Blau = \{\text{ja}, \text{nein}\}\}$.

Ein Nachteil hierbei ist der Verlust von Informationen über die Beziehung der Werte untereinander. Es ist für das neuronale Netz also nicht mehr erkennbar, dass von den Attributen Rot, Grün und Blau zu jedem Zeitpunkt lediglich eines den Wert „ja“ haben kann.

2.3.2 Numerische Attribute

Grundsätzlich gibt es keine technischen Gründe, die eine Anpassung von numerischen Attributen erfordern. Es hat sich jedoch gezeigt, dass $[0, 1]$ verteilte Werte von KNNs besser gelernt werden können.

Eine besondere Situation entsteht, wenn ein Autoencoder trainiert werden soll, der mehrere Eingabeattribute mit verschiedenen Wertebereichen nutzt. Während des Trainings eines KNNs werden die Gewichte so angepasst, dass der Fehler zwischen erwarteter und tatsächlicher Vorhersage minimiert wird. Wenn die Wertebereiche sich nun stark unterscheiden, führt dies zu Problemen bei den Metriken die üblicherweise zur Fehlerabschätzung genutzt werden. Angenommen es existieren zwei Attribute mit den Wertebereichen $[0, 1]$ und $[0, 100]$. Hier ist sofort erkennbar, das die Relevanz des Fehlers, bei gleich großem absolutem Fehler, stark davon abhängt bei welchem Attribut er aufgetreten ist.

Für eine optimale Fehlerbewertung ist es also hilfreich, die Attribute auf einen gemeinsamen Wertebereich anzupassen. Dafür existieren drei wesentliche Techniken: Diskretisierung, Normalisierung und Standardisierung. Die ersten beiden sind dabei für diese Arbeit relevant und werden daher kurz erläutert.

Diskretisierung

Bei der Diskretisierung wird eine kontinuierliche Variable in eine diskrete umgewandelt. Dafür wird zuerst der Wertebereich der Daten ermittelt. Nachfolgend wird dieser in mehrere kleinere Intervalle aufgeteilt. Für jeden Wert aus der Eingabemenge wird nun bestimmt in welchem Intervall er liegt.

Zur Bestimmung der Intervalle gibt es viele verschiedene Techniken. Die einfachste ist es mehrere gleich große Intervalle anzulegen. Es existieren auch Techniken die Intervallbreiten flexibel wählen, um so eine gleichmäßige Verteilung der Daten zu erreichen.

Normalisierung

Die Normalisierung führt eine Änderung des Wertebereichs auf den Bereich $[0, 1]$ durch. Damit kann der neue Wert wie eine Prozentangabe gelesen werden. Wie bei der Diskretisierung wird zunächst Kenntnis über den tatsächlichen Wertebereich benötigt. Anschließend kann ein Wert x mit $x' = (x - \min) \cdot \frac{1}{\max - \min}$ zum normalisierten Wert x' überführt werden.

2.4 Evaluationsmetriken

Nachdem ein Vorhersagesystem trainiert wurde ist es wichtig zu beurteilen, wie die Qualität des gelernten Modells ist. Dafür gibt es eine Reihe von Möglichkeiten: Die aufwendigste, aber beste Methode ist es, dass ein Experte der Domäne die gelernten Konzepte betrachtet und beurteilt. Je nach Komplexität ist dies mit hohen Kosten und Zeitaufwand verbunden. Dementsprechend greift man für eine Abschätzung auf numerische Metriken zurück.

Für eine Abschätzung ist es notwendig, dass ein Satz von gelabelten Daten zur Verfügung steht, der zum Testen verwendet wird. Diese Daten sollten nicht bereits Teil der Trainingsdaten gewesen sein, da das Verhalten bei unbekannten Daten beurteilt werden soll.

2.4.1 Binäre Klassifikation

Für die Bewertung von binären Klassifizierern wird normalerweise eine Konfusionsmatrix benutzt. Dafür wird der Klassifikator auf die Testdaten angewendet und die Ausgabe mit dem Label verglichen. Es können dabei verschiedene Situationen auftreten:

- True Positive: Die Klasse ist positiv und wurde positiv vorhergesagt.
- False Positive: Die Klasse ist negativ wurde aber als positiv vorhergesagt.
- False Negative: Die Klasse ist positiv wurde aber als negativ vorhergesagt.
- True Negative: Die Klasse ist negativ und wurde negativ vorhergesagt.

Bei der Auswertung wird gezählt, wie oft jede Situation auftritt und in die Konfusionsmatrix in Tabelle 2.1 eingetragen. Basierend darauf können verschiedene Metriken berechnet werden. Die einfachste ist hierbei *Accuracy* = $\frac{tp+tn}{tp+tn+fp+fn}$. Der bestmögliche Wert ist 1.0 und der schlechteste 0. Diese Metrik hat jedoch ein großes Problem, wenn die Klassenverteilung ungleichmäßig ist. Angenommen es gibt sehr viele Negativ-Beispiele und der Klassifizierer sagt immer negativ voraus. Dann ist die Anzahl der True negatives so hoch, dass die False Negatives kaum berücksichtigt werden.

Der *F1-Score* vermindert dieses Problem. Dieser kombiniert Precision und Recall über das harmonische Mittel, um ein ausgeglicheneres Ergebnis zu erhalten. Er berechnet sich aus $F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Allerdings besitzt der F1-Score zwei Grenzfälle:

- tp = fn = 0
  Dieser Fall tritt auf, wenn kein einziges Positiv Beispiel im Testdatensatz vorhanden ist. Dies kann auf eine möglicherweise ungeeignete Wahl der Testdaten hindeuten.
- tp = fp = 0
  Wenn der Klassifizierer gelernt hat immer negativ vorherzusagen tritt dieser Fall auf. Dies passiert, wenn der Klassifizierer kein Konzept lernen konnte, dass besser ist als immer die größte Klasse vorherzusagen. Im Gegensatz zum vorherigen Fall kann im Vorfeld wenig getan werden, um dies zu verhindern. Für diesen Fall existiert keine explizite Definition. Innerhalb dieser Arbeit wird in einem solchen Fall eine Precision von 0 festgesetzt und mit diesem Wert weitergerechnet.

		Vorhersage		
		A	nicht A	
Tatsächliche Klasse	A	true positives (tp)	false negative (fn)	Recall = $\frac{tp}{tp+fn}$
	nicht A	false positive (fp)	true negative (tn)	
		Precision = $\frac{tp}{tp+fp}$		

Tabelle 2.1: Binäre Konfusionsmatrix

2.4.2 Multiklassen Klassifikation

Der F1 Score ist nur für binäre Klassifizierer definiert, er lässt sich aber auch auf die Multiklassen Vorhersage übertragen. Dafür wird zunächst für jede Klasse eine Konfusionsmatrix berechnet. Anschließend wird diese verrechnet. Dafür gibt es drei wesentliche Verfahren:

Micro-F1
: Hierbei wird eine globale Konfusionsmatrix durch Addition der Einzelmatrizen erstellt. Beispielhaft für die true positives: $\text{tp}_{total} = \sum_{n=1}^{AnzahlKlassen} tp_n$. Dadurch entsteht eine Binäre Konfusionsmatrix für die der F1-Score definiert ist.

Macro-F1
: Zunächst wird der F1-Score für alle Klassen berechnet. Anschließend wird der ungewichtete Durchschnitt bestimmt.

Weighted-F1
: Hier wird ebenfalls der F1-Score aller Klassen miteinander verrechnet. Dabei erfolgt eine Gewichtung anhand der positiven Beispiele der jeweiligen Klasse.

Innerhalb dieser Arbeit wird, solange nicht explizit erwähnt, immer der Weighted-F1 Score verwendet.

2.5 Adult Datensatz

Das United States Census Bureau führt etwa alle zehn Jahre Volkszählungen durch. Die daraus entstandenen demografischen Daten für das Jahr 1994 bilden die Grundlage des „Adult“ Datensatzes. Barry Becker führte die ursprüngliche Extraktion der Daten durch. Die extrahierten Datensätze weisen dabei viele Aspekte auf, die für die Verwendung zum Maschinellen Lernen problematisch sind. Es ist offensichtlich, dass bei einer Volkszählung der Fokus auf einer möglichst präzisen Datenerhebung liegt. Dementsprechend sind viele Attribute sehr feingranular aufgebaut. Die zugehörigen Wertebereiche bestehen dabei zum Teil aus Stichpunkten, die miteinander kombiniert werden. Darüber hinaus gibt es viele Datensätze, in denen einzelne Einträge fehlen. Der „Adult“ Datensatz ist eine Version, die von Ronny Kohavi vereinfacht und bereinigt wurde. Dabei wurden einige Daten entfernt und Attribute zusammengefasst.

Der „Adult“ Datensatz hat einige Eigenschaften, die für diese Analyse besonders wichtig sind. So besitzt er einen verhältnismäßig großen Anteil von nominalen Attributen. Diese sind aufgrund des überschaubaren Wertebereichs einfacher auszuwerten als kontinuierliche. Ein weiterer wichtiger Aspekt ist die Menge der Attribute. Es gibt genügend Attribute, so dass nicht jedes Neuron dieselben Attribute für seine Logik verwenden muss. Auf der anderen Seite ist die Menge klein genug, dass ein Mensch den gesamten Regelraum überblicken kann. Ein weiterer wichtiger Punkt ist die Größe des Datensatzes. Neuronale Netze brauchen im Allgemeinen eine gewisse Menge von Trainings-Datensätzen, um eine befriedigende Leistung zu erreichen. Mit etwa 50000 Beispielen bietet „Adult“ eine ausreichende Menge von Daten.

Attribut	Mögliche Werte
age	kontinuierlich [17, 90]
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
fnlwgt	kontinuierlich
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	kontinuierlich [1, 16]
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, Male
capital-gain	kontinuierlich
capital-loss	kontinuierlich
hours-per-week	kontinuierlich [1, 99]
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands
class	>50K, <=50K

Tabelle 2.2: Attribute des Adult-Datensatzes

Der wohl wichtigste Punkt ist Interpretierbarkeit der Domäne. Beim „Adult“ Datensatz ist es nicht notwendig sehr viel Vorwissen aus der Domäne zu haben, um die Attribute zu verstehen. In den meisten Fällen reicht die Allgemeinbildung, so dass Attribute ohne weitere Erklärung verständlich sind. Auch besitzen die meisten Menschen eine natürliche Intuition über mögliche Zusammenhänge von Attribute in diesem Kontext.

Der Datensatz ist grundsätzlich als Klassifikationsproblem in Bezug auf das Einkommen entworfen. Die Daten sind so gelabelt, dass sie einem Befragte zuordnen, ob er mehr oder weniger als 50000 Dollar Jahreseinkommen verdient. Die Klassifikationslabel sind ein weiter Grund für die Auswahl dieses Datensatzes.

2.5.1 Attribute

Tabelle 2.2 enthält eine Übersicht über alle Attribute sowie deren Wertebereiche. Eine Auswahl von Attributen, die nicht sofort verständlich sind, werden im Folgenden näher erläutert. Dabei wird auch auf die Bedeutung von einzelnen Werten eingegangen.

fnlwgt

„fnlwgt“ ist die Abkürzung für Final Weight. Dies ist ein Gewichtswert, der von der Größe der demografischen Gruppe des Befragten abhängt. Es ist wichtig diesen Wert zu berücksichtigen, wenn demografische Analysen durchgeführt werden. Bei einer Klassifikationsaufgabe sollte er nicht verwendet werden und wurde deshalb für diese Arbeit aus dem Datensatz entfernt.

education-num

Dieses Attribut ist eine numerische Version von education die im Gegensatz zum nominalen Attribut eine Ordnung zwischen den Werten zulässt. Dieser Wert beschreibt den höchsten Abschluss, den die Person zum Befragungszeitpunkt besaß.

num	1	2	3	4	5	6	7	8
education	Preschool	1-4th	5-6th	7-8th	9th	10th	11th	12th
num	9	10	11	12	13	14	15	16
education	High School	Some-college	Assoc-voc	Assoc-acdm	Bachelor	Master	Prof-school	Doctorate

Tabelle 2.3: Zuordnung von education-num zu education

marital-status

Die Werte dieses Attributs sind weitestgehend selbsterklärend. Anzumerken ist lediglich der Unterschied zwischen „ Married-civ-spouse“ und „Married-AF-spouse“. Hierbei steht civ für einen Zivilist und AF für die amerikanischen Streitkräfte. Somit bezeichnet der eine Wert die Ehepartnerschaft zwischen Zivilisten und der andere von mindestens einem Militärangehörigen.

Bei „Married-spouse-absent“ gilt ein Ehepartner nur bei längerer Abwesenheit als „absent“. Geschäfts- und Dienstreisen fallen nicht darunter.

occupation

Dieses Attribut beschreibt die Beschäftigung einer Person. Aufgrund der Berufsvielfalt findet dabei eine Einordnung in verschiedene Berufskategorien statt. Da die Werte des Attributs Abkürzungen sind, sind sie in einigen Fällen nur schwer zu erraten. Entsprechend sind sie im Folgenden noch einmal vollständig abgedruckt.

- Executive, administrative and managerial
- Professional specialty
- Technicians and related support
- Sales
- Administrative support, including clerical
- Private household service
- Protective service
- Other service (except private and protective service)
- Precision production, craft, and repair
- Machine operators, assemblers, and inspectors

- Transportation and material moving
- Handlers, equipment cleaners, etc.
- Farming, forestry and fishing

relationship

„relationship“ beschreibt die Beziehung zur Referenzperson des Haushalts. Diese ist im Allgemeinen diejenige, der das Haus oder die Wohnung gehört. Im Falle von Ehepaaren kann dies sowohl der Mann als auch die Frau sein. Auch hier existieren einige Werte, die sich nicht unmittelbar erschließen.

Own-child beschreibt ein Kind, das durch Geburt, Adoption oder Heirat ein Teil der Familie ist.

Other-Relative umfasst alle Verwandten außer dem Kind und dem Ehepartner. Dies sind beispielsweise Eltern, Geschwister oder Pflegekinder.

Not-in-family beschreibt jemanden, der alleine lebt oder mit jemandem zu dem er keine verwandtschaftliche Beziehung unterhält.

Unmarried definiert unverheiratete Paare.

class

Der Datensatz ist primär für Klassifikationsaufgaben ausgelegt. Das Klassenattribut beschreibt ob das Brutto-Einkommen des Befragten größer oder kleiner gleich 50000 Dollar ist. Die Klassenverteilung weist eine starke Tendenz Richtung $<=$50K auf, so entfallen bereits 75% aller Einträge auf diese Klasse. Bei einer Bewertung der Klassifikationsleistung ist es wichtig dies zu berücksichtigen.

3 Stand der Forschung

Im Bereich des maschinellen Lernens konnten künstliche neuronale Netze bereits einige Erfolgen zeigen. Besonders im Bereich der Vorhersagegenauigkeit können diese sich gegen andere Modelle durchsetzen und bessere Leistungen erreichen [Thr93; HBV06]. Huysmans, Baesens und Vanthienen sprechen hierbei sogar von überragenden Resultaten in verschiedenen Benchmarks, merken aber gleichzeitig an, dass die Verbreitung von Neuronalen Netzen durch die schlechte Erklärbarkeit gehemmt wird.

Neben ihrer gute Vorhersageleistung haben neuronale Netze den weiteren Vorteil, dass sie sich durch parallele Verarbeitung auch effizient auf große Datenmengen anwenden lassen. Bei der heutigen Entwicklung von Speichermedien und des Internets steigt die Verfügbarkeit von Informationen immer weiter an. Besonders im Bereich Audio und Video fallen mittlerweile sehr große Datenmengen an. Klassifikation und Annotation von Videomaterial ist dabei ein Thema, dass bereits einige Aufmerksamkeit bekommen hat [Wan+09; Li+12] und viele interessante Möglichkeiten eröffnet. In diesem Bereich spielen die neuronalen Netze aufgrund ihrer Effizienz dabei bereits eine wichtige Rolle. In diesem Umfeld ist es häufig so, dass für die Klassifikation notwendige Features initial unbekannt sind. Es hat sich hierbei als sinnvoll erwiesen komprimierende Autoencoder auf den Daten zu trainieren. Dabei entwickeln sich in den tieferliegenden Schichten sogenannte Hidden Features. Diese automatisch entdeckten Muster können dann genutzt werde, um eine Klassifikation durchzuführen [Gen+15]. Sowohl dynamische Inhalte wie Video und Audio, als auch statische Elemente wie Bilder oder Text, sind anfällig für Fehler, die beispielsweise durch Sensorrauschen entstehen. Autoencoder können auch hierbei helfen [LFZ16; Gu+15], indem sie das Rauschen vermindern und damit die Robustheit der Klassifikation für neue und unbekannte Daten verbessern.

Künstliche neuronale Netze verfügen über ein äußerst komplexes Modell, wodurch die Erklärbarkeit erheblich eingeschränkt wird. Durch die Repräsentation des Wissens durch Gewichte, entstehen viele Freiheiten des Netzes für die Entscheidungsfindung. Dadurch können völlig unterschiedliche Eingaben zu ähnlichen Ausgaben führen [SAG99].

Es stellt sich die Frage, wie ein Netz zu seiner Entscheidung kommt. Menschen neigen dazu einem System nicht zu vertrauen, wenn sie dessen Entscheidung nicht nachvollziehen können. Dies kann in kritischen Situationen zu Problemen führen. In [MST94, Kapitel 2] wird der Fall des Three-Mile Island Kernkraftwerks geschildert. Dort trat die Situation auf, dass es zu einer Störung kam und die dortigen Systeme eine sofortige Abschaltung empfahlen. Das Bedienpersonal konnte die Gründe für diese Empfehlung nicht nachvollziehen und reagierte nicht darauf.

In sicherheitskritischen Bereichen ist es wichtig, dass dem Modell vertraut werden kann [ADT95]. Damit dies stattfinden kann, ist es unabdingbar, dass die Funktionsweise des Netzes erklärt werden kann. Insbesondere sollte das Verhalten sich auch bei ungewöhnlichen oder völlig anderen Eingaben erklären lassen. In Bereichen, in denen nur schlecht alle möglichen Eingaben systematisch getestet werden können, ist dies von noch größerer Bedeutung

Neben dem Vertrauensaspekt ist es auch wünschenswert das Modell erklären zu können, wenn das Netz eine Lösung für ein ungelöstes oder neues Problem findet. Beispielsweise im medizinischen Bereich kann dies genutzt werden, um unbekannte Zusammenhänge zwischen Krankheiten und Symptomen zu erkennen [Lav99]. Wenn das hierdurch neu erlangte Wissen auch an anderer Stelle verwendet werden soll, ist es notwendig eine belastbare Begründung für das Verhalten des Netzes angeben zu können. Selbst wenn die eigentliche Korrektheit oder Vertrauenswürdigkeit nicht betroffen ist, kann es wichtig sein eine Begründung angeben zu können. Wenn beispielsweise in einer Bank ein System über die Vergabe eines Kredits entscheidet, kann es notwendig sein dem Kunden eine Begründung für eine Absage geben zu können [Huy+11].

Im Kontext von Regelextraktionstechniken wird häufig der „accuracy-comprehensibility-tradeoff“ genannt. Dieser beschreibt die Notwendigkeit einer Abwägung zwischen der Verständlichkeit und der Vor-

hersageleistung. Im Normallfall kann eine Lerntechnik nicht in beiden Aspekten gleichzeitig sehr gute Ergebnisse liefern. Es muss also eine Entscheidung getroffen werden, welcher Aspekt priorisiert werden soll [JNK04; SMM12] . Dies kann sich je nach Anwendungszweck unterscheiden. Im Bereich von Knowledge Discovery und Data Mining ist es häufig eher wichtig eine gute Verständlichkeit zu erreichen. Die Gegenseite ist nun die typische Klassifikationsaufgabe, bei der es vor allem um Genauigkeit geht. Regelextraktionstechniken verbinden diese beiden Extreme miteinander. Aus dem genauen, aber komplexen Modell werden einfachere Regel extrahiert. Für die erstellten Regel finden sich unterschiedliche Anwendungen. In manchen Situationen werden die extrahierten Regel lediglich für die Erklärung genutzt und in anderen wiederum auch für die eigentliche Vorhersage [Joh+06].

In der Vergangenheit gab es bereits viele Untersuchungen zu den Modellen. In den kommenden Abschnitten soll zunächst beleuchtet werden, welche Techniken entwickelt wurden, um neuronale Netze in eine verständliche Form zu bringen. Dabei werden diese in verschiedene Kategorien eingeordnet. Anschließend werden Aspekte vorgestellt, die für die Interpretierbarkeit sowie einen Vergleich notwendig sind. Weiterhin werden einige Techniken vorgestellt, die bereits zur Wissensextraktion genutzt wurden.

3.1 Regelextraktion

Algorithmen und Ansätze für die Regelextraktion gibt es sehr viele. Eine Untersuchung die einen guten Überblick liefert haben Andrews, Diederich und Tickle in [ADT95] vorgestellt. Darin stellen sie umfangreich vor, wie Regelextraktionsalgorithmen klassifiziert werden können. Dafür wird unter Anderem die Regelform, die Ausdrucksstärke der Regeln und deren Qualität betrachtet. Der wichtigste Aspekt ist jedoch die Beziehung der Regeln zur inneren Struktur des Ausgangsmodells. Dabei werden drei wesentliche Kategorien definiert: „decompositional“, „ pedagogical“ und „eclectic“. Die letzte Kategorie umfasst dabei die Ansätze, die Elemente aus den beiden vorherigen kombinieren. Nachfolgend sollen einige interessante Regelextraktionstechniken vorgestellt werden. Dabei findet eine Strukturierung und Einordnung in die eben genannten Kategorien statt.

3.1.1 Decompositional

Die erste Kategorie wird in älterer Literatur auch häufig „connectionist“ genannt und fasst Algorithmen zusammen, welche die Struktur des Netzes berücksichtigen. Das bedeutet, dass der Fokus auf den einzelnen Neuronen sowie deren Beziehungen liegt. Dabei können beispielsweise die Gewichte der Neuronen Einfluss auf die Regeln nehmen. Ein in der Literatur häufig genannter Algorithmus aus dieser Kategorie ist KT [Fu91; Fu94], der zu den ersten vorgestellten Verfahren aus dieser Kategorie zählt. Die Grundidee des Algorithmus orientiert sich dabei an der Funktionsweise der Neuronen. Diese feuern wenn ein bestimmter Schwellenwert überschritten wird. Entsprechend bildet KT Regeln durch heuristischen Kombination von Attributen, so dass der Schwellenwert des Neurons überschritten wird. Dabei werden zum einen bestätigende Regeln „confirming Rules“, als auch ablehnende Regeln „disconfirming Rules“ gesucht. Die bestätigenden Regeln liefert dabei eine Aussage, wann das Neuron 1 liefert, wohingegen die ablehnenden Regeln eine 0 vorhersagen. Diese Regelgenerierung wird für sämtliche Neuronen durchgeführt. Abschließend findet ein Überarbeitungsschritt statt, in dessen Rahmen die Regeln von tieferen Schichten so umgeschrieben werden, dass sie auf die Eingabeattribute und nicht die vorherigen Neuronen referenzieren.

Da bei den Algorithmen in dieser Kategorie Zugriff auf das innere des Netzes besteht, ist es auch möglich dieses zu verändern. Ansätze wie NeuroRule [LSL95] oder NNRE [TG99] nutzen dies, um vor der Extraktion noch einen Pruning Schritt durchzuführen. In diesem werden nach Neuronenverbindungen gesucht, die nur einen geringen Wert haben. Verbindungen die redundant sind oder nur einen geringen Gewichtswert haben, können entfernt werden. Dabei ist es wichtig darauf zu achten, dass sich der Fehler des Netzes nicht erhöht. Durch das Pruning wird das Netz deutlich einfacher und erlaubt es bei der Extraktion einfachere und besser verständlichere Regeln zu erzeugen.

Eine weitere interessante Technik wurde von Towell und Shavlik [TS91] vorgestellt. Der MofN Algorithmus nutzt eine besondere Form der Wenn-Dann Regeln, bei der nicht alle Bedingungen gleichzeitig eintreten müssen. Dabei verfügt jede Regel über eine Reihe von Bedingungen und es ist lediglich vorgegeben, dass n von m existierenden Bedingungen erfüllt sein müssen, damit diese Regel für die Vorhersage genutzt wird. Die zugrundeliegende Idee hierbei ist, dass es Gruppen von Bedingungen gibt, bei denen es keine einzelne, unabdingbare Bedingung gibt, die auf jeden Fall erfüllt sein muss. Auch wenn hierbei ein Bezug zu einzelnen Neuronen naheliegt, so ist das Konzept von MofN auch bei Algorithmen anzutreffen, die das Netz eher als Blackbox betrachten [CS94].

Bei der Untersuchung von Extraktionsalgorithmen findet häufig eine Fokussierung auf neuronale Netze mit nur einem Hidden Layer statt. Besonders bei komplexeren Problem ist dies unter Umständen nicht ausreichen, um eine gute Leistung zu erzielen. In [Zil15] wird der DeepRED Algorithmus als Erweiterung von CRED[ST01] vorgestellt, der explizit auf Netze mit mehreren Schichten ausgerichtet ist. Dabei werden wie im Original mehrere Entscheidungsbäume aufgebaut. Für jedes Ausgabeneuron wird zunächst ein Baum erstellt, der, basierend auf den Aktivierungswerten der vorherigen Neuronen, die Klasse vorhersagt. Anschließend wird für jeden verwendeten Splitpunkt ein weiterer Entscheidungsbaum aufgebaut, der die Ausgabe in Bezug auf die vorherige Schicht erklärt. Dies wird wiederholt bis die Eingabeschicht erreicht ist. Wie bei den meisten Algorithmen aus dieser Kategorie existiert auch hier ein finaler Schritt, in dem die Zwischenregeln beziehungsweise Zwischenbäume zusammengefasst werden.

3.1.2 Pedagogical

Algorithmen für, die die innere Struktur keine Relevanz für die Extraktion hat, fallen in die Kategorie „pedagogical“. Das Ursprungsmodell wird hierbei als Blackbox betrachtet, ohne dabei Wissen über Form und Inhalt zu besitzen. Es ist jedoch möglich beliebige Eingabedaten auf das Modell anzuwenden und das Ergebnis zu beobachten. Durch die Betrachtung auf diesem hohen Abstraktionsnivau sind viele Techniken nicht auf neuronale Netze beschränkt. Sie können auch für andere Lernverfahren mit schwer verständlichem Modell, wie beispielsweise Support Vector Machines, verwendet werden.

Ein interessanter Ansatz aus dieser Kategorie ist die „Validity-Interval Analysis“ [Thr93; Thr95]. Dabei handelt es sich um ein System, um das Ein- und Ausgabeverhalten eines neuronalen Netzes zu analysieren. Jedem Neuron wird dafür ein sogenannter „validity intervall“ zugeordnet, der beschreibt in welchem Bereich sich die Aktivierungswerte dieses Neurons bewegen. Das Analyse System kann dann prüfen, ob die Bedingungen, die durch diese Intervalle definiert werden, konsistent sind. Diese kann auch zur Extraktion von Regeln verwendet werden. Eine Regel umfasst dabei die Intervalle der Eingabeneuronen und spannt damit einen Hyperwürfel auf. Liegt ein Eingabewert innerhalb dieses Würfels, so wird diese Regel aktiv. Diese Analyse geht davon aus, dass die Aktivierungswerte der Neuronen unabhängig voneinander sind. Da dies nicht immer der Fall sein muss, wird in [SAG99] kritisiert, dass Regeln in einem solchen Fall nicht „maximally general“ sind.

Viele Ansätze aus dieser Kategorie betrachtet die Extraktion nicht als ein Suchproblem, sondern als eine Aufgabe aus dem Bereich des überwachten Lernens. Während zuvor versucht wurde die Ausgabe des Netzes anhand der Eingabewerte zu erklären, wird nun versucht die Beziehung zwischen Ein- und Ausgabedaten zu lernen [CS94]. Dabei wird ausgenutzt, dass insbesondere neuronale Netze effizient berechnet werden können und es so möglich ist eine große Menge von Beispielen zu klassifizieren. Dieser Ansatz des Samplings ist in Listing 3.1 als Pseudocode abgebildet. Dabei wird zunächst eine Menge von Eingabedaten bestimmt, die dann auf das Blackbox-Modell angewendet wird. Mit den so erhaltenen Eingabe- Ausgabepaaren kann dann ein anderer Lerner trainiert werden. Craven und Shavlik [CS94] berichten, dass damit deutlich schneller gute Ergebnisse erreicht können als mit anderen Ansätzen. Interpretierbarkeit und Regelqualität war dabei vergleichbar mit den Ansätzen, denen Informationen über die Gewichte zur Verfügung standen.

```
1 input:
2         model = Blackbox model
```

```
3
4  begin
5          inputData = getInputData()
6          outputData = sample(inputData, model)
7
8          newModel = trainModel(inputData, outputData)
9          return newModel
10 end
```

Listing 3.1: Pseudocode Samplingansatz

Für die Wahl der Eingabedaten gibt es verschiedene Möglichkeiten. Die einfachste ist es die Trainingsdaten, die auch für das Netz verwendet wurden, erneut für die Extraktion zu nutzen. Eine weitere Möglichkeit ist, dass für die Extraktion eine weitere Menge von unbenutzten Beispielen zur Verfügung steht. Eine dritte Möglichkeit ist die Verwendung von künstlichen Daten. Mit BIO-RE [TG99] existiert ein Algorithmus, der systematisch alle möglichen Eingabekombinationen ausprobiert. Dieser Ansatz ist auf binäre Ein- und Ausgabeneuronen beschränkt, da es für kontinuierliche Werte nicht möglich wäre den gesamten Eingaberaum abzusuchen. Mit den ermittelten Ergebnissen wird eine vollständige Wahrheitstabelle erzeugt, die dann wiederum durch einen Lerner abgebildet wird. Wird kein Pruning verwendet, kann dieser Ansatz das Netz perfekt nachbilden. Gleichzeitig ist aber leicht erkennbar, dass dieser Ansatz nur sehr schlecht mit steigender Anzahl von Eingabeneuronen skaliert.

Ein anderer Sampling-Ansatz ist ANN-DT [SAG99], der auch bei vielen Eingabeneuronen noch gut funktioniert. Der Algorithmus nutzt ein existierendes Trainingsset und kann bei Bedarf neue Eingabevektoren aus Zufallswerten erzeugen. Dabei sollen diese neuen Eingabedaten möglichst ähnlich zu denen sein, die bereits existieren. Dafür werden zunächst die existierenden Trainingsbeispiele geclustert. Anschließend wird für jeden potentiellen Messpunkt die minimale Distanz zu einem dieser Cluster bestimmt. Ist ein Messpunkt zu weit entfernt befindet er sich nicht mehr in der Nachbarschaft und wird verworfen. Verworfene Eingabevektoren gelten als untypisch für den Trainingsdaten und würden bei der Regelextraktion den Fokus auf unwichtige Aspekte lenken.

Zur Verwendung von künstlichen Daten wurden bereits einige Untersuchungen durchgeführt. In [JN09] werden diese auch „Oracle Data“ genannt, da sich die Blackbox wie ein Orakel verhält, dessen Antwort nicht direkt nachvollzogen werden kann. In der Untersuchung wurde festgestellt, dass die Kombination von originalen und künstliche Daten die Klassifikation deutlich verbessert. Wurden ausschließlich künstliche Daten verwendet war das Ergebnis jedoch schlechter als bei den Originaldaten. In [Joh+06] stellten Johansson u. a. fest, dass die dort extrahierten Entscheidungsbäume extrem kompakt waren, wenn ausschließlich künstlichen Trainingsdaten verwendet wurden.

3.2 Interpretation

Es ist schwierig ein allgemein gültiges Maß für die Interpretierbarkeit von Regelmengen zu definieren. Bei den meisten Vorstellungen von Regelextraktionstechniken wird dabei auf die Größe und Komplexität des Modells zurückgegriffen. Die Größe wird dabei üblicherweise mit der Anzahl von Regeln und die Komplexität durch die Anzahl von Bedingungen gemessen. Durch die numerische Ausdrucksweise ist es leicht diese Werte zu bestimmen und zu vergleichen. Für eine wirkliche Bewertung sind diese Werte allerdings nicht ausreichend. Die Interpretierbarkeit hängt in nicht unerheblichem Maße von den eigentlichen Inhalten des Modells ab [Fre14] . Ein größeres Modell ist nicht notwendigerweise schlechter als ein kleineres. So konnte in [Lav99] in einem medizinischen Umfeld festgestellt werden, dass die Experten das größere Modell bevorzugten. Die kleinere Version war den Experten zu stark vereinfacht und generalisierte zu sehr über die Symptome. Damit war das Modell ungeeignet einen Arzt bei seiner Entscheidung effektiv zu unterstützen. Neben dem Informationsgehalt ist auch die persönliche Ausdauer des Betrachters entscheidend. Freitas vergleicht dabei zwei Fälle. Bei einem Fall fanden die Nutzer 41

Regeln als „überwältigend“ groß, wohingegen in einem zweiten Fall ein Nutzer die Ausdauer hatte etwa 30 000 Regeln zu analysieren. Dabei konnten in diesem Fall letztendlich eine kleine Teilmenge von interessanten Regeln extrahiert werden.

Bei großen Regelmengen kann es sein, dass eine kleine Menge von Regeln die wichtigen Fälle abdeckt und die weiteren Regeln sich nur um Sonderfälle und Ausnahmen kümmern. Bei einer Interpretation wäre entsprechend nur eine geringe Menge von Regeln für die Hauptbedeutung interessant. Selbst wenn diese Regeln sehr gut zu interpretieren wären, würde bei einer Bewertung ausschließlich über die Größe dies gegebenenfalls zu falschen Schlüssen führen. Eine Metrik die versucht dieses Problem zu beheben ist die „prediction-explanation size“ [OF13]. Für diese wird die Anzahl der Bedingungen, die tatsächlich ausgewertet werden müssen um ein Beispiel zu klassifizieren, für alle Beispiele aus dem Testset gezählt. Der Durchschnitt dieser Werte ergibt das Endresultat. Die Metrik berücksichtigt die verminderte Relevanz von wenig genutzten Regeln und erlaubt damit eine bessere Einschätzung der tatsächlichen Komplexität. Dies erlaubt eine Abschätzung wie viele Bedingungen ein Nutzer auswerten muss, um den Inhalt des Modells zu verstehen.

Neben der Größe ist auch die Form des Modells ein erheblicher Faktor. Nicht immer ist die Regelform die bestmögliche Repräsentation des Wissens. Wenn beispielsweise die Eingabe die Pixel eines Bildes sind, ist eine visuelle Darstellung des Modells möglicherweise besser als eine Sammlung von Regeln. Auch wenn erkannt wurde, dass Regeln eine für das Problem angemessene Repräsentation sind, existieren Unterschiede. Bei Extraktion von Regeln aus einem neuronalen Netz bauen viele Algorithmen einen Entscheidungsbaum auf. Dieser kann einfach in eine Regelmenge überführt werden, indem für jeden möglichen Pfad eine eigene Regel erzeugt wird. Ein Nachteil dieser Vorgehensweise ist jedoch, dass sich die erzeugten Regeln Attribute teilen. Die Bedingung, die im Entscheidungsbaum im obersten Knoten steht, findet sich so in sämtlichen Regeln wieder. Auch andere Attribute werden wiederholt für die Regeln genutzt. Bei Regelmengen, die direkt erzeugt wurden besteht dieses Problem nicht. Dort kann jede Regel aus völlig unterschiedlichen Attributen beziehungsweise Bedingungen bestehen. Ohne Vorgaben aus vorherigen Regeln kann der Lerner die Konzepte besser verständlich ausdrücken [FWA10; He+06]

Neben der Regelextraktion existieren auch andere Verfahren für die Analyse der inneren Struktur eines Netzes. Mitchell [Mit92] stellt eine Methode vor, um das Verhalten von einzelnen Neuronen im Hidden Layer „geometrisch“ in Bezug auf Bereiche von Eingabewerten auszuwerten. Dabei wird auch präsentiert, wie dieser Ansatz dazu verwendet werden kann, um Redundanzen zwischen Neuronen zu ermitteln.

Eine weitere Technik ist die „Contribution Analysis“[San89]. Die Beteiligung eines Neurons ist definiert als „the product of the hidden unit’s activation when the net is presented with the specified input and the weight from the hidden unit to the output unit.“ Für alle Hidden und Output Neuronen wird eine Hauptkomponentenanalyse durchgeführt. Dadurch kann für jedes Hidden Neuron bestimmt werden, für welches Ausgabeneuron es zuständig ist. In der Untersuchung wurde dieser Ansatz auf ein neuronales Netz angewendet, das phonetische Features (Position der Zunge, etc.) aus Buchstabenpaaren bestimmt. Dabei konnte festgestellt werden, dass diese Technik bessere Ergebnisse liefert als lediglich die Gewichte zwischen Hidden Layer und Ausgabeschicht zu betrachten.

Es ist keine theoretische Idee, dass Regeln genutzt werden können, um neues neues Wissen zu erlangen. Beispielsweise konnten durch Arbeiten von Clare und King[CK01; KC+03] im Bereich der Genetik neue Zusammenhänge entdeckt werden. Dies findet häufig im Umfeld von Data Mining statt, bei dem die Regeln direkt aus den Eingabedaten gelernt werden. Untersuchungen, die eine semantische Betrachtung im Bereich der Regelextraktion vornehmen, sind hingegen äußerst selten. Viele Veröffentlichungen stellen Techniken zur Extraktion vor, ohne die Resultate konkret zu interpretieren. Es findet meist eine Beschränkung auf die Bewertung der generellen Interpretierbarkeit statt. Eine der wenigen Arbeiten, die ein neuronales Netz interpretieren und die Konzepte des Hidden Layers untersucht, findet sich in [GS88]. In dieser Arbeit wurde ein Netz trainiert, um die Ergebnisse eines Sonars auszuwerten. Das Ziel war es eine Unterscheidung von einem Metallzylinder und einem zylindrischen Stein zu realisieren. Das Netz konnte dabei eine sehr hohe Accuracy von etwa 99% erreichen. Um zu erkennen welche Konzepte das Netz dafür nutzt, wurde eine kleinere Version mit drei Neuronen im Hidden Layer analysiert. Zu-

nächst wurde versucht durch Visualisierung der Gewichten einen Zusammenhang herzustellen. Hierbei zeigte sich, dass dies nicht möglich war „The classification strategy of the network cannot be readily understood by visually inspecting weight displays“. Im nächsten Schritt wurden für alle Hidden Units die Beispiele anhand der Eingabeaktivierung hierarchisch geclustert. Die Cluster wurden nach der Stärke der Aktivierung des jeweiligen Neurons geordnet. Durch Betrachtung der Eigenschaften der Beispiele innerhalb der Clusters, konnten die Autoren bei allen Neuronen einige realisierte Features konkret benennen. Es wurde festgestellt, dass kein Neuron ein einzelnes Feature modellierte, sondern mehrere Teilaspekte zum Tragen kamen. Diese Features unterschieden beispielsweise schmal- und breitbandige Signale oder betrachteten den Anfang oder das Ausklingverhalten des Signals. Viele Signale konnten bereits durch ein einzelnes Neuron korrekt klassifiziert werden. Bei anderen Signalen war das Zusammenwirken aller Neuronen für die Klassifizierung notwendig. Interessanterweise konnten nicht alle Beispiele durch die erwähnten Features erkannt werden. Einige Signale wurden dadurch klassifiziert, dass die Neuronen nach Signalspitzen an ganz bestimmten Positionen suchten. Diese Kodierung wurde wohl entwickelt zur Wiedererkennung von einzelne Ausnahmen und Besonderheiten.

4 Aspekte und Methodik

Im vorherigen Abschnitt wurden verschiedene Techniken zur Regelextraktion sowie bestehende Überlegungen zur Interpretierbarkeit vorgestellt. Dieses Kapitel soll zunächst den eigenen Extraktionsansatz für Regeln mittels Sampling darlegen. Anschließend wird vorgestellt, wie die semantische Analyse des Hidden Layers stattfindet und wie die Regeln analysiert werden. Dabei wird beschrieben welche Aspekte von Interesse sind sowie welche Punkte besondere Aufmerksamkeit erfordern. Im Anschluss daran werden zwei neue Ansätze dargelegt, die versuchen die Konzepte des Hidden Layers für eine neue Aufgabe umzufunktionieren. Mit diesen wird untersucht inwieweit die Konzepte eines Autoencoders für Klassifikationsaufgaben geeignet sind.

4.1 Regel Extraktion

Für die Regelextraktion dieser Arbeit wurde ein Sampling-Ansatz gewählt. Die grundlegende Idee hierbei wurde bereits in Abschnitt 3.1.2 vorgestellt. Die meisten der existierenden Sampling-Ansätze betrachtet das gesamte Netz als Blackbox und samplen dieses. Da in dieser Arbeit jedoch einzelne Neuronen von Relevanz sind, betrachtet dieser Ansatz die Neuronen jeweils als einzelne Blackboxen.

Wenn das Innere des Netzes von Interesse ist, wird üblicherweise eher ein Algorithmus aus der Kategorie „Decompositional“ gewählt, da diese bereits Zwischenregeln für einzelne Neuronen erzeugen. In dieser Arbeit wurde darauf verzichtet einen existierenden Algorithmus zu verwenden und stattdessen ein eigener Ansatz gewählt. In Bezug auf die Analyse hat dieser einige Vorteile, die bestehende Algorithmen nicht bieten.

Konfigurierbarkeit
: Dadurch dass die Neuronen einzeln betrachtet werden, ist es grundsätzlich möglich für jedes Neuron eine eigene Konfiguration zu definieren. Dies erlaubt es, dass einzelne Schichten oder Neuronen bei Bedarf mit einer anderen Pruningstufe extrahiert werden können. Auch ein Austausch des Regellerners wäre möglich.

Laufzeit
: Da das Training von Neuronen der gleichen Schicht unabhängig voneinander stattfindet, kann der Lernprozess parallelisiert und damit beschleunigt werden. Bei großen Netzen, die über eine Vielzahl von Neuronen verfügen, kann ein Cluster mit vielen CPU-Kernen dabei eine Verbesserung der Laufzeit erreichen. Bei großen Netzen kann dies entscheidend sein, ob eine Extraktion überhaupt in annehmbarer Zeit möglich ist.

Flexibilität
: Es ist möglich im Nachhinein die Modelle von einzelnen Neuronen auszutauschen ohne das gesamte Modell neu zu trainieren. Stellt ein Anwender fest, dass die für ein Neuron gelernten Regeln nicht gut sind, so könnte für dieses Neuron ein erneutes Training mit anderen Parametern oder anderem Startwert durchgeführt werden. Es wäre nicht notwendig die Extraktion für das gesamte Netz zu wiederholen.

Das Sampling wurde so umgesetzt, dass das Netz dabei Schicht für Schicht von vorne nach hinten abgebildet wird. Jede Schicht wird durch eine Menge von Regellernern repräsentiert, die jeweils genau ein Neuron der Schicht abbilden. Jeder dieser Regellerner lernt wiederum eine Menge von Regeln. Die Regelmengen der verschiedenen Regellerner werden nicht zusammengefasst. Die Eingabe für jeden Regellerner besteht dabei in der Ausgabe der Regellerner der vorherigen Schicht. Abbildung 4.1 zeigt

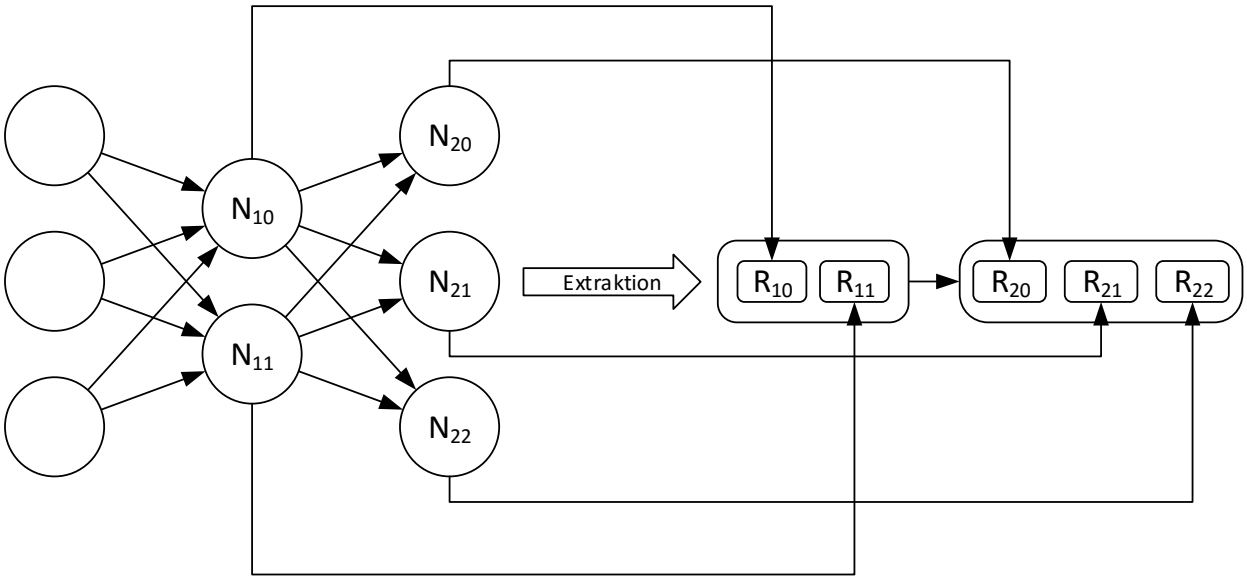

Abbildung 4.1: Ergebnismenge der Regelextraktion

dabei schematisch das Ergebnis einer Extraktion. Die beiden Schichten 2 und 3, die durch Regellerner abgebildet wurden, sind dabei als Kästen dargestellt.

Es wurden keine Regellerner für Schicht 1 gelernt, da diese die Eingabeschicht ist. Da diese keine Logik realisiert ist es nicht notwendig sie abzubilden. Um die Ausgabe für die Regellerner einer Schicht zu berechnen, ist es zwingend notwendig, dass das Ergebnis aller Regellerner der vorherigen Schicht vorliegt.

4.1.1 Ablauf

Listing 4.1 zeigt den Extraktionsvorgang in Pseudocode. Um die Regellerner einer Schicht zu lernen, werden zunächst die Eingabedaten für diese Schicht in Zeile 7 bestimmt. Hier ist es wichtig zu unterscheiden zwischen Netzeingabe und Schichteingabe. Die Netzeingabe bezeichnet die Daten, die in der Eingabeschicht des Neuronales Netzes anliegen. Diese Daten sind die Eingabedaten aus dem Trainingsdatensatz. Die Schichteingabe ergibt sich durch Anwendung aller vorhergehenden Schichten. Um diese für eine Schicht n zu berechnen, wird zunächst die Netzeingabe auf die Regellerner der Schicht 1 angewendet. Deren Ausgabe ist dann die Eingabe für die Regellerner der 2. Schicht. Dieser Vorgang wird solange wiederholt, bis die Ausgabe der Schicht n-1 vorliegt. Dies wird für alle Trainingsbeispiele wiederholt und ergibt damit die Eingabedaten der späteren Trainingsdaten.

In Zeile 8 findet das namensgebende Sampling statt. Für jedes Trainingsbeispiel wird die Netzeingabe an das Netz angelegt und es wird bestimmt welche Ausgabe die Neuronen der Ziel-Schicht erzeugen. Es ergibt sich eine Menge von Vektoren, deren Dimensionalität der Anzahl der Neuronen in der Schicht entspricht. Sämtliche Elemente dieser Vektoren sind kontinuierlich und deren Verteilung wird bestimmt durch die Konfiguration des Netzes.

Grundsätzlich könnte dieser Ansatz jene Daten ohne weitere Verarbeitung zum Training der Regellerner verwenden. Dafür wäre es notwendig einen Lernalgorithmus zu wählen, der für Regressionsprobleme geeignet ist und kontinuierliche Vorhersagen erlaubt. Unabhängig davon wurde sich gegen eine solche Umsetzung entschieden. Kontinuierliche Werte würden eine Interpretation des Hidden Layers deutlich erschweren, da beliebig viele Zwischenwerte existieren. Diskrete Werte hingegen haben einen überschau-

baren Ergebnisraum und eignen sich daher deutlich besser. Bei binären Werte ist die Interpretation am einfachsten, da hierbei nur noch „ja/nein“ Konzepte entstehen können. In dieser Arbeit findet daher eine Beschränkung auf [0,1] verteilte, binäre Neuronen im Hidden Layer statt. Die Ausgabe der Neuronen wird nach dem Sampling elementweise binärisiert. Da sich gezeigt hat, dass die Ausgaben meist normalverteilt um 0.5 sind, wurde ebenjener Wert auch als Grenzwert für die Binärisierung gewählt.

Aus den Ergebnissen von Zeile 7 und 8 ergeben sich die Trainingsdaten für alle Regellerner dieser Schicht.

Von Zeile 10 bis 15 findet der eigentliche Lernprozess statt. Da sämtliche Neuronen einer Schicht unabhängig voneinander abgebildet werden können, ist diese Stelle gut dafür geeignet eine Parallelisierung durchzuführen. Der Lernprozess der einzelnen Regellerner ist der ressourcen- und damit zeitaufwendigste Teil dieses Ansatzes. Durch eine Parallelisierung kann daher ein erheblicher Zeitgewinn realisiert werden. Insbesondere da der Lernvorgang einer Schicht keine Synchronisation erfordert, skaliert diese Vorgang in der Theorie nahezu linear mit der Anzahl von verwendeten Threads. Das Limit ist erreicht, wenn die Anzahl der Threads der Anzahl der Neuronen in der Schicht entspricht. Das dies in der Praxis nicht ganz so gut funktioniert, wird in Abschnitt 5.6 näher beleuchtet.

Ein weiterer Vorteil des unabhängigen Lernens ist es, dass es nicht notwendig ist komplexe Zustände über den Lernvorgang eines Neurons hinaus zu speichern. Es ist ausreichend die Regelmenge als Endresultat zu speichern. Dies erlaubt es das Ergebnis unmittelbar nach Abschluss persistent zu speichern. Dadurch kann der Extraktionsprozess jederzeit unterbrochen werden und zu einem späteren Zeitpunkt fortgesetzt werden. Der einzige Informationsverlust der dabei auftritt, sind die Neuronen, die zum Zeitpunkt des Abbruchs gelernt wurden. Andere Ansätze verfügen meist nicht über diesen Komfort und können nicht unterbrochen werden. Besonders bei aufwendigen Berechnungen, die auf Clustern mit Maximallaufzeiten pro Job durchgeführt werden, kann dies ein Problem darstellen. Diese Persistenzoperationen wurden in Listing 4.1 zu Gunsten der Übersichtlichkeit nicht abgedruckt.

Dieser Sampling Ansatz hat nur eine schwache Verbindung zum verwendeten Regellerner. Entsprechend einfach ist es, diesen zu modifizieren oder auszutauschen. Da dadurch nicht das vom neuronalen Netz gelernte Konzept beeinflusst wird, ist dies eine gute Möglichkeit verschiedene Regelmengen zum gleichen Konzept zu erstellen. Ein naheliegender Parameter ist hierfür das Pruning, das in Ripper bereits integriert ist. Damit kann die Regelmenge bei verschiedenen Pruningstufen betrachtet werden, um so die Relevanz von Attributen zu beurteilen. Es bietet sich auch an verschiedene Schichten mit verschieden starken Pruning zu lernen. Es kann beispielsweise vorteilhaft sein im Hidden Layer ein schwächeres Pruning vorzunehmen, um komplexere Regeln zu ermöglichen.

4.1.2 Mögliche Sampling Fehler

Beim Sampling können leicht Fehler entstehen, die den gesamten Extraktionsprozess beeinflussen. Diese Fehler können auf mehrere Arten entstehen, aber zum Teil auch automatisch korrigiert werden. Um zu demonstrieren welche Möglichkeiten existieren, ist in Tabelle 4.1 ein Beispiel mit willkürlichen Daten für die erzeugten Trainingssets zu erkennen. Als Grundlage wurde das in Abbildung 4.1 präsentierte Netz gewählt. Die Eingabedaten sind dabei in den Spalten mit „x“ gekennzeichnet und die binärisierte Ausgabe des Netzes als „f“. Die Eingabedaten x_{00} bis x_{02} stammen direkt aus der Eingabeschicht und können daher noch keinen Fehler enthalten. Diese Werte sind bei einem idealen Autoencoder identisch mit den Ausgabe f_{20} bis f_{22}.

In diesem Beispiel hat bereits beim ersten Trainingsbeispiel das neuronale Netz eine Abweichung erzeugt und am Neuron f_{22} 1 statt 0 vorhergesagt. Dieser Fehler muss nicht unbedingt ein Problem darstellen. Beispielsweise ist bei Denoising Autoencodern keine exakte Rekonstruktion gewollt. Andererseits kann sich das Netz auch „unsicher“ gewesen sein und einen Wert nahe 0,5 erzeugt haben. Die Binärisierung nimmt den wahrscheinlicheren Wert und verliert die Information über die Unsicherheit. Auf diese Art von Fehler kann zu diesem Zeitpunkt kein Einfluss mehr genommen werden, sondern nur durch wiederholtes oder besseres Training des Netzes.

```
input:
        TrainingData = A set of training examples
        NeuralNet = The trained neural net

RuleLearners := ∅
foreach layer of the neural net
        layerInput = getInputForLayer(TrainingData, RuleLearners, layer)
        outputForAllNeurons = getNeuralNetOutputForLayer(TrainingData, NeuralNet, layer)

        foreach neuron of the layer
                labels = getLabelsForNeuron(outputForAllNeurons, neuron)
                configuration = getConfiguration(layer, neuron)
                learner = learnRules(configuration, layerInput, labels)
                add learner to RuleLearners
        end
end
```

Listing 4.1: Regelextraktion durch Sampling

x_{00}	x_{01}	x_{02}	f_{10}	f_{11}
0	1	0	1	0
0	0	1	1	1
1	0	0	0	1

x_{10}	x_{11}	f_{20}	f_{21}	f_{22}
1	0	0	1	1
0	1	0	0	1
0	1	1	0	0

Tabelle 4.1: Beispiel für gesampelte Trainingsdaten

Im zweiten Trainingsbeispiel ist ebenfalls ein Fehler eingetreten. Hier hat die Regelmenge R_{10}, die für die Erzeugung von x_{10} zuständig ist, einen Fehler produziert. Solche Fehler sind unvermeidbar, da sich die Regelmenge niemals perfekt an die Trainingsdaten anpassen soll, um Overfitting zu vermeiden. Dadurch dass die Regelmenge von Schicht 2 sich an den Daten des Netzes orientiert, ist es im idealen Fall möglich Regeln zu lernen, die den Fehler wieder korrigieren. Fehler dieser Art propagieren daher nicht zwangsläufig durch alle Schritte und erhöhen den Fehler bei der Ausgabe. Wenn jedoch die Fehler, wie im Beispiel zuvor, durch das Netz entstanden sind, existiert keine Möglichkeit zur automatischen Korrektur.

Das dritte Trainingsbeispiel hat in keinem Zwischenschritt einen Fehler, wobei die Fehlerfreiheit nicht abschließend festgestellt werden kann. Nach dem Training von Schicht 2 ist es möglich, dass die darin erzeugte Regelmengen wieder einen Fehler erzeugen.

4.2 Untersuchte Aspekte

Eine Interpretation der Konzepte im Hidden Layer kann in Bezug auf viele verschiedene Dinge durchgeführt werden. Der nachfolgende Abschnitt behandelt die verschiedenen untersuchten Aspekte und verdeutlicht wie diese ermittelt werden und welches Ziel damit verfolgt wird.

4.2.1 Semantische Interpretation des Hidden Layers

Hierbei soll versucht werden für jedes Neuron im Hidden Layer eine Beschreibung zu finden, die das Konzept zusammenfasst. Dabei soll diese ein höheres Abstraktionsniveau haben, als die Aussagen, die durch die einzelnen Regeln getroffen werden. Im Idealfall lässt sich die gesamte Regelmenge durch einen

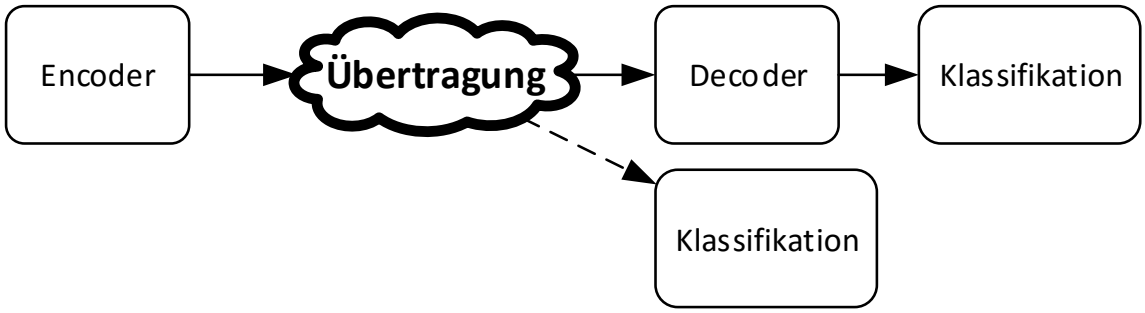

Abbildung 4.2: Anwendungsbeispiel für Klassifikation auf komprimierten Daten

einzigen Satz ausdrücken. Um eine solche Kurzbeschreibung zu finden, werden verschieden Aspekte der Menge berücksichtigt. Grundsätzlich ist die semantische Interpretation deutlich stärker subjektiv und weniger reproduzierbar als eine quantitative Untersuchung. Zudem bietet sie viele Unsicherheiten und hat einen großen Ermessensspielraum. Um die Interpretation objektiver zu gestalten, werden einige quantitative Werte hinzugezogen.

Hierunter fällt beispielsweise die Verwendungshäufigkeit von Attributen. Kommt ein Attribut in allen Regeln vor, ist dies ein Indiz das es wichtig ist. Kommt es jedoch in allen Neuronen des Hidden Layers vor, reduziert dies seine Bedeutung. Darüber hinaus gibt es verschiedene Attribute beziehungsweise Werte, die semantische Ähnlichkeiten aufweisen. So geben „Married-civ-spouse“ und „Husband“ beide eine Aussage über den Heiratsstatus ab. Je nach Kontext ist es sinnvoll, kleinere Unterschiede zu vernachlässigen

Bei Attributen, die kontinuierliche Eingabewerte besitzen, ist es ähnlich wie bei binären Attributen. Splitpunkte, die den Wertebereich möglicherweise nur halbieren sind deutlich weniger aussagekräftig als sehr hohe bzw. niedrige Schwellenwerte. Wenn ein Splitpunkt nur wenige Werte zulässt, ist dies offensichtlich sehr restriktiv und damit gut für die Interpretation. Ein Splitpunkt, der sehr viele Werte zulässt kann jedoch negiert interpretiert werden. Wenn eine Aussage $x \leq 0.9$ vorhanden ist, kann dies auch interpretiert werden als „nicht >0.9“. Der Interpretationswert solcher Eingaben ist entsprechend stark abhängig von der Verteilung der Eingabedaten.

Durch Variation der Pruningstärke kann eine Abstufung der Relevanz von Attributen erreicht werden. Selten vorkommende Attribute, die jedoch auch bei sehr hohen Pruningstärken noch vorhanden sind, haben erwartbar eine hohe Aussagekraft. Indem das Regelmodell mehrfach mit verschiedenen Pruningparametern erstellt wird und dessen Ergebnisse verglichen werden, kann somit nach Gemeinsamkeiten und Unterschieden gesucht werden.

4.2.2 Klassifikation basierend auf dem Hidden Layer des Autoencoders

Als weiterer Punkt wird untersucht inwieweit sich das Hidden Layer eines Autoencoders zur Klassifikation eignet. Dafür soll die eigentliche Klassifikation nicht durch ein neuronales Netz vorgenommen werden, sondern durch einen Regellerner. Diese Technik wird in Zukunft kurz als „separate Klassifikation“ bezeichnet. Die Grundidee ist es, dass mit Hilfe der durch Regeln abgebildeten Encoder-Komponente eine Kompression durchgeführt wird und anschließend die Ausgabe des Hidden Layers als Eingabe für den Lerner verwendet wird. Ein Autoencoder lernt lineare Abhängigkeiten, die nicht direkt für die Klassifikation notwendig sind. Es wird betrachtet, ob diese gelernten Konzepte geeignet sind eine Klassifikation zu ermöglichen, obwohl sie nicht zu dessen Zweck gelernt wurden. In diesem Kontext ist ebenfalls interessant, welche Konzepte des Autoencoders benutzt werden und wie diese aufgebaut sind. Darüber

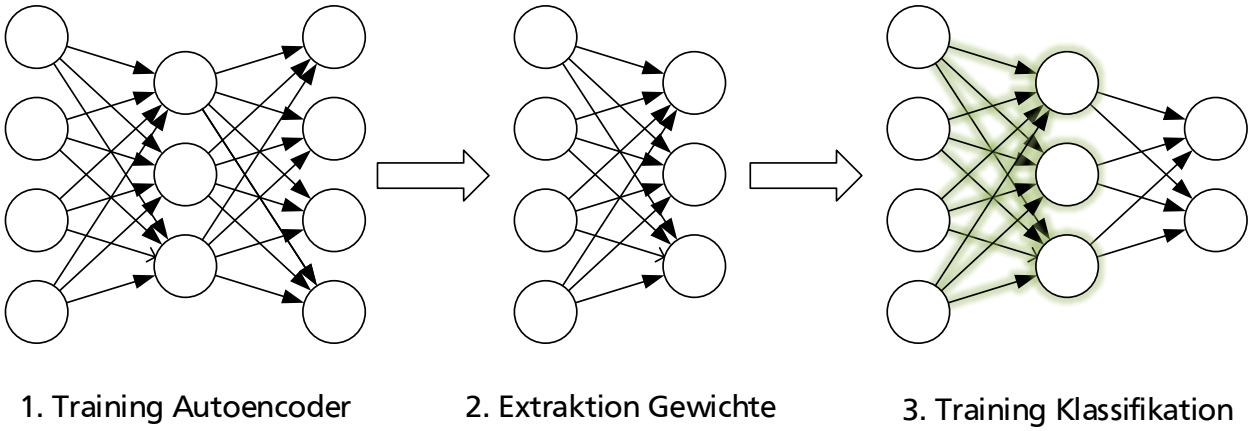

Abbildung 4.3: Gewichtsextraktion aus einem Autoencoder

hinaus wird dies mit den Konzepten verglichen, die entstehen wenn ein Regellerner direkt auf den Originaldaten und nicht dem Hidden Layer lernt. Es wird betrachtet, ob diese sich fundamental unterscheiden bzw. inwieweit diese die gleichen Konzepte entwickeln.

Die Erwartung ist es, dass der Autoencoder in seinem Hidden Layer Konzepte besitzt, die bei einem Netz zur Klassifikation nicht gelernt würden, aber dennoch bei der Klassifikation helfen. Dies kann Rückschlüsse über die Beziehung innerhalb der Datenmenge liefern, die aus dem reinen Klassifikationsmodell nicht ersichtlich wären.

Kompression ist eine der Fähigkeiten von Autoencodern. Bei Anwendungen, bei denen großen Datenmengen anfallen, kann dies genutzt werden, um notwendige Ressourcen zur Speicherung oder Übertragung zu minimieren. Abbildung 4.2 soll einen solchen Fall zeigen. Daten werden durch den Encoder komprimiert und anschließend übertragen. Auf einem zweiten System werden diese dekodiert und weiterverarbeitet. Wenn die Konzepte im Hidden Layer geeignet sind, wäre es nun möglich den Dekompressionsschritt zu überspringen und direkt die komprimierten Daten zu klassifizieren. Damit kann die für die Dekodierung notwendige Laufzeit vermieden werden. Im Kontext von Big Data und zeitkritischen Anwendungen kann dies eine nennenswerte Verbesserung darstellen.

4.2.3 Transfer von Gewichten

Im vorangegangen Abschnitt wurde das Konzept der separaten Klassifikation vorgestellt. Ein zweiter Ansatz soll ebenfalls die Eignung der Konzepte des Hidden Layers zur Klassifikation untersuchen. Der Autoencoder selbst wird dabei diesmal nicht durch eine zweite Technik ergänzt wie zuvor. Stattdessen dient er als Initialisierung für eine Transformation. Im Rahmen dieser Umwandlung soll der Autoencoder in ein neues neuronales Netz überführt werden, dass die Klassifikationsaufgabe übernimmt. Dieses neue Netz ist dafür so konfiguriert, dass es dem Autoencoder im Aufbau sehr ähnlich ist.

Die Grundidee dieser Transformation ist es, die bereits gelernten Gewichte als Initialwerte für das neue Netz zu verwenden. Dadurch ist es möglich, dass generelle Konzepte nicht neu gelernt werden müssen, sondern vom Encoder übernommen werden können. Dadurch kann sich das Netz möglicherweise bessere und verständlichere Ergebnisse liefern.

Abbildung 4.3 zeigt die Schritte mit denen das neue Klassifikationsnetz erstellt wird.

Training des Autoencoders

Im ersten Schritt erfolgt das Training des Autoencoders und die Bestimmung der initialen Konzepte.

An dieser Stelle muss darauf geachtet werden, dass der gelernte Autoencoder ein zufriedenstellendes Modell gelernt hat. Sollte dem nicht so sein, ist es dennoch möglich ein gutes Netz als Ergebnis der Transformation zu erhalten. Da allerdings untersucht werden soll, ob und wie sich die Konzepte durch diesen Ansatz verändern, ist es wünschenswert bereits im Autoencoder gut interpretierbare Konzepte zu besitzen. Dies macht es einfacher etwaige Unterschiede zu identifizieren.

Extraktion der Gewichte

Neuronale Netze repräsentieren ihre gelernten Konzepte intern über mehrere Gewichte. Da dies lediglich numerische Werte sind, können diese einfach kopiert werden. In diesem Schritt werden alle Gewichte kopiert, die zu den Neuronen des Hidden Layers gehören. Bei dem Beispiel in Abbildung 4.3 sind dies die 12 Werte, die durch die Pfeile dargestellt sind, sowie drei Biaswerte. Da lediglich die Konzepte für die Encoder-Stufe relevant sind, ist es auch nur notwendig diese zu übertragen. Die Gewichte und Konzepte der Decoder-Stufe werden in keinem weiteren Schritt mehr benötigt und können daher vollständig ignoriert werden.

Training der Klassifikation

Im letzten Schritt werden die zuvor kopierten Gewichte eingefügt. Hierbei ist es wichtig, dass das Klassifikationsnetz im Bereich der Encoder-Stufe die gleiche Konfiguration aufweist wie der Autoencoder. Bei Abweichungen in der Neuronenzahl wäre es nicht möglich die Gewichte zu übertragen, da entweder zu viele oder zu wenige Gewichte vorhanden wären. Bei Änderungen in der Art der Neuronen könnte es zu Problemen kommen. Bei einem Wechsel von binäre zu kontinuierliche Neuronen dürfte dabei das Risiko zwar gering sein, es ist jedoch ratsam eine identische Konfiguration zu verwenden. In der Abbildung sind die eingefügten Gewichte grün umrandet.

In diesem Beispiel wird ein Netz für eine binäre Klassifikation mit One-Hot Kodierung dargestellt. Die Konfiguration des Netzes nach der ehemaligen Encoder-Stufe ist keinen Einschränkungen unterworfen. Wie bei einem regulären Klassifikationsnetz können noch beliebig viele weitere Schichten verwendet werden. Da diese Analyse ihren Fokus auf dem Hidden Layer des Encoders hat, werden allerdings nur einfache Klassifikationsnetze verwendet, die nach der Encoder-Stufe nur noch eine Ausgabeschicht besitzen. Diese Vereinfachung wurde getroffen, damit der Einfluss der Konzepte auf das Klassifikationsergebnis noch manuell nachvollzogen werden kann. Würden hier weitere Schichten existieren, wäre dies aufgrund der Komplexität nur schwer möglich.

Für die Gewichte der Ausgabeschicht existiert keine Vorgabe bezüglich der Startwerte, diese können beispielsweise durch eine Xavier-Initialisierung [GB10] bestimmt werden.

Als letzter Teil dieses Schritts findet ein erneutes Training statt. Da es sich nun um ein Klassifikationsproblem handelt, muss der zugehörige Datensatz auch ein Klassifikationsproblem beschreiben und mit entsprechenden Klassenlabels ausgestattet sein. Durch das Training lernt das Netz die kopierten Konzepte für die Klassifikation zu nutzen. Dabei können sich auch die Gewichte und damit die Konzepte der Encoder-Stufe ändern. Weiterhin ist es auch möglich, dass die alten Konzepte verworfen werden und völlig neue erzeugt werden.

5 Evaluation

Eine semantische Evaluation erfordert ein tiefes Verständnis der Problemdomäne. Entsprechend findet in dieser Arbeit eine Fokussierung auf einen einzelnen Datensatz statt. Der Aufbau und die Eigenschaften des Census-Datensatz wurden bereits zuvor vorgestellt. Es existieren viele weitere Datensätze, die bei der Generierung von Regeln als Datengrundlage dienen. Häufig werden beispielsweise *iris* oder *breast-cancer* aufgrund ihrer überschaubaren Größe verwendet. Der Census-Datensatz hat gegenüber diesen den Vorteil, dass er erheblich größer ist und über mehr Attribute verfügt. Dies erlaubt mehr Variationen in den Regeln und damit möglicherweise interessantere Konzepte im Hidden Layer.

Der Census-Datensatz verfügt über viele numerische Attribute mit völlig unterschiedlichen Wertebereichen. In Abschnitt 2.3.2 wurde bereits angesprochen, dass dies das Lernen erschweren kann und es häufig sinnvoll ist, einen Datensatz auf die Verwendung in einem neuronalen Netz vorzubereiten. Es wurden zwei verschiedene Varianten des Datensatzes durch Vorverarbeitung erzeugt. Eine normalisierte und eine diskretisierte Version. Bei der diskreten Variante wurden die numerischen Attribute jeweils in 10 gleich große Intervalle aufgeteilt. Mit den beiden Varianten soll ermittelt werden, ob diese unterschiedliche Konzepte erzeugen. Auch generelle Unterschiede in Bezug auf die Interpretierbarkeit werden betrachtet.

Für die Evaluation wurde der Datensatz in ein Trainingsset mit 27800 und ein Testset mit 15060 Beispielen aufgeteilt. Um die Klassifikationsleistung der verschiedenen Ansätze auch ohne Cross-Validation aussagekräftig zu bestimmen, werden jeweils mehrere Extraktionen durchgeführt. Durch Veränderung der Startwerte der Regellerner entstehen verschiedene Regelmengen. Für diese wird der Weighted F1-Score berechnet und der Durchschnitt bestimmt. Im Gegensatz dazu wird bei der semantischen Betrachtung jeweils nur ein Netz untersucht.

5.1 Qualität des Autoencoders

Um eine fundierte Aussage über die Konzepte zu treffen, ist es nötig, dass der Autoencoder eine gute Qualität besitzt. Nur wenn es ihm gelingt die Eingabedaten gut zu reproduzieren, kann davon ausgegangen werden das die gelernten Konzepte die zu Grunde liegenden Beziehungen in den Daten zuverlässig repräsentieren. Nur wenn dies der Fall ist, können auch aussagekräftig Regelmengen extrahiert werden. Es ist naheliegend, dass auch die anderen Ansätze, die die Klassifikationseignung untersuchen, auf sinnvolle Konzepte angewiesen sind.

Für die Leistungsbeurteilung des Autoencoders existieren verschiedene Möglichkeiten. Beispielsweise kann dies über eine einfache Distanzfunktion realisiert werden, die für jedes Beispiel die Distanz von Eingabe und tatsächlichem Ausgabevektor bestimmt. Dies hat jedoch den Nachteil, dass der semantische Bezug von Werten eines Attributs verloren geht. Das heißt es ist nicht möglich zu betrachten, ob es Attribute gibt die besonders gut oder schlecht reproduziert werden können. Diese Analyse verwendet daher eine andere Technik. Für die Auswertung wird zunächst der Ein- und Ausgabevektor jeweils in so viele Sub-Vektoren aufgeteilt wie es Attribute gibt. Alle diese Sub-Vektoren enthalten nur Elemente, die zum gleichen Attribut gehören. Damit modellieren dieser Vektoren die Attribute in One-Kodierung.

Beim diskretisierten Datensatz kann so jedes Attribut wie ein herkömmliches Klassifikationsproblem ausgewertet werden. Der spezifische Eingabevektor ist das Label und der spezifische Ausgabevektor ist die Vorhersage. Auf Basis der jeweiligen Konfusionsmatrizen können die Einzelleistungen bewertet werden, die dann gemittelt werden, um den Final-Wert zu erhalten.

Im Falle des normalisierten Datensatzes ist dies nicht möglich, da hier kontinuierliche Werte existieren. Diese sind nicht durch One-Kodierung darstellbar und somit auch nicht auswertbar. Stattdessen wird die 2-Norm verwendet, um die Distanz attributweise zu bestimmen. Für die Endauswertung wird der

Root Mean Squared Error für jedes einzelne Attribut berechnet. Der Durchschnitt dieser Werte wird als Gesamtmaß zum Vergleichen verschiedener Konfigurationen verwendet.

Es ist naheliegend, dass ein Autoencoder bessere Ergebnisse erreicht, wenn es mehr Neuronen im Hidden Layer gibt. Dies erlaubt es mehr Zusammenhänge zu modellieren und speichern. Auf der anderen Seite sollte das Hidden Layer auch nicht zu groß sein, da die Neuronen sonst keine interessanten Konzepte entwickeln. Es könnte passieren, dass die Neuronen lediglich lernen die Eingabedaten weiterzuleiten. Auch soll ein Autoencoder eine Kompression durchführen, was bei einem großen Hidden Layer nicht gegeben wäre. Insbesondere im Kontext dieser Analyse ist es wünschenswert das Hidden Layer nicht allzu groß zu wählen, da jedes Neuron einzeln interpretiert wird. Damit steigt die Gesamtinterpretationszeit proportional mit der Anzahl der Neuronen.

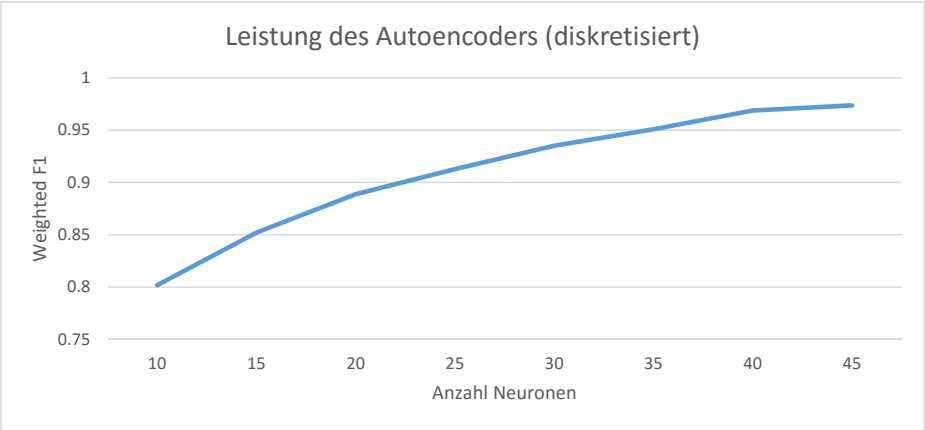

Abbildung 5.1: Leistung des Autoencoders (diskretisierter Datensatz)

Zunächst wurden verschiedene Konfigurationen des Autoencoders trainiert und bewertet. Abbildung 5.1 zeigt die Ergebnisse für den diskretisierten Datensatz bei verschiedenen Größen des Hidden Layers. Es ist zu erkennen, dass dessen Qualität schon früh recht gut ist und sich tendenziell mit der Anzahl der Neuronen verbessert. Ab einer Größe von 40 Neuronen verbessert sich die Leistung nur noch in kleinen Schritten.

Es wurde entschieden einen Autoencoder mit 20 Neuronen im Hidden Layer zu verwenden. Mit einem F1-Score von etwa 89% erreicht er gute Ergebnisse und befindet sich dennoch in einem Bereich, bei dem eine Einzelinterpretation durchführbar ist. Ein weiterer Grund auf Leistung zu Gunsten von Größe zu verzichten ist es, dass damit die einzelnen Konzepte weniger fein granular und allgemeiner werden. Dies lässt sich leichter interpretieren.

Für den normalisierten Datensatz zeigt sich ein abweichendes Verhalten. Bei steigender Neuronenzahl sinkt zunächst der durchschnittliche Fehler pro Attribut auf 0,08 bei 30 Neuronen. Wird das Hidden Layer weiter vergrößert steigt der Fehler jedoch wieder leicht an. Es ist daran bemerkenswert, dass die Qualität der kontinuierlichen Attribute bei mehr Neuronen besser wird, die der One-Hot kodierten jedoch abnimmt. Insbesondere nach den Ergebnissen des diskretisierten Datensatzes ist dies überraschend. Aus den gleichen Gründen wie beim diskretisierten Datensatz wurde entschieden ebenfalls 20 Neuronen im Hidden Layer zu verwenden. Dabei steigt der durchschnittliche Fehler auf 0,1. Den schlechtesten Wert hat dabei das Alter mit einem Fehler von 0,14. Da die normalisierten Daten [0, 1] verteilt sind, ist dieser Fehler durchaus signifikant. Es scheint für den Autoencoder schwierig zu sein, dass Alter zu rekonstruieren.

5.2 Qualität der Regeln

Im vorangegangen Abschnitt wurde die Qualität des Autoencoders diskutiert. Neben dessen Qualität ist auch die Qualität der extrahierten Regeln wichtig. Hierbei ist es entscheidend, dass die Regeln den Encoder möglichst gut abbilden. Gleichzeitig darf das Modell nicht zu komplex werden, da es sonst schwierig oder sogar unmöglich ist eine Interpretation durchzuführen. Entsprechend ist es notwendig geeignete Pruning-Parameter zu finden, die das Modell klein halten, aber gleichzeitig nicht zu sehr verschlechtern.

In Abbildung 5.2 ist dargestellt wie gut es den Regeln für die diskreten Variante gelingt den Autoencoder nachzubilden. Dabei wird zusätzlich die durchschnittliche Anzahl von Regeln im Hidden Layer durch die Balken angegeben.

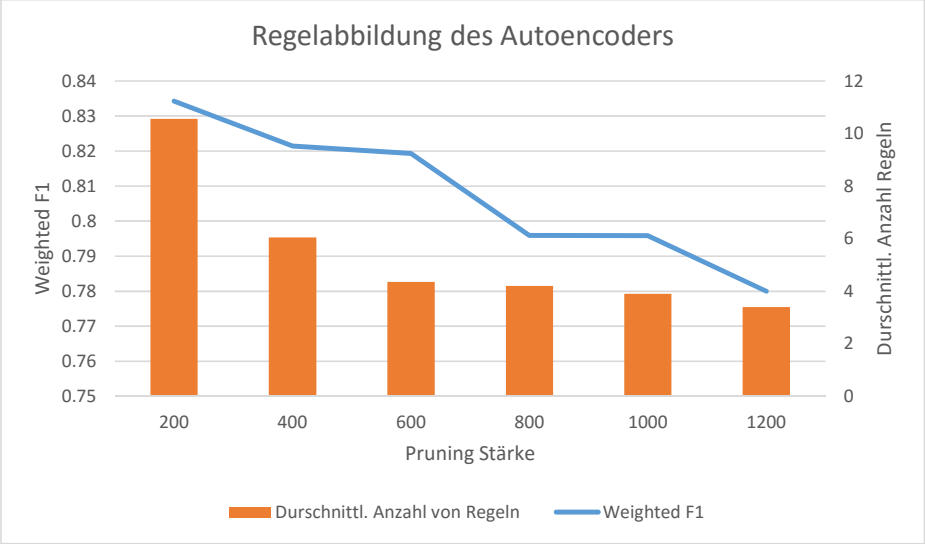

Abbildung 5.2: Regelbasierte Abbildung des Autoencoders (diskretisierter Datensatz)

Wenig überraschend sinkt die Qualität bei steigender Pruningstärke. Bei 1000 sind durchschnittlich nur noch 3.9 Regeln vorhanden. Im Vergleich zu vorherigen Stufen haben diese auch eine deutlich geringere Komplexität, so dass diese sich noch gut zusammenfassen lassen. Bei stärkerem Pruning sinkt die Performance deutlich stärker ab, ohne eine nennenswerte Verringerung der Regelzahl zu erreichen. In Abschnitt 4.2.1 wurde bereits angesprochen das mehrere Pruningstufen herangezogen werden sollen, um zu erkennen ob und welche Attribute aus der Regelmenge verschwinden. Hierfür bietet sich eine Stärke von 400 an, da dort die Regelzahl schon deutlich reduziert ist, ohne dabei große Einbußen bei der Leistung hinnehmen zu müssen. Obwohl bei 600 die Leistung nur gering schlechter ist und weniger Regeln besitzt, ist 400 dennoch besser geeignet. Durch mehr Regeln können Details möglicherweise besser dargestellt werden. Für die Interpretation wurden entsprechend 400 und 1000 als Vergleichsstufen gewählt.

Für den normalisierten Datensatz ist es nicht möglich diese Metrik zu berechnen, da der zur Regelextraktion verwendete Klassifizierer nur für nominale Klassenattribute trainiert werden kann. Zum besseren Verständnis der Regeln betrachtet diese Arbeit nur binäre Ausgabeneuronen. Die erzeugten Regeln für die Ausgabeschicht sind beim normalisierten Datensatz dementsprechend ungenau und sind daher konzeptionell nicht in der Lage eine sinnvolle Vorhersage zu treffen. Für die Regelextraktion im Hidden Layer ist dies allerdings kein Problem, da diese unabhängig von anderen Schichten stattfindet. Es existiert lediglich keine Gesamtabschätzung über die Fähigkeit der Regeln den Autoencoder abzubilden.

5.3 Konzepte im Autoencoder

In den Tabellen 5.1 und 5.2 ist eine Übersicht über alle Neuronen und deren identifizierte Konzepte zu sehen. Dabei wurde versucht den oder die wesentlichen Kernpunkte des Konzepts mit wenigen Stichpunkten zusammenzufassen. Bei manchen Neuronen enthalten die Tabellen mehrere Zeilen. In solchen Fällen deckt das Konzept mehrere Bereiche ab, die nicht zusammengefasst werden konnten. Auf Logik-Ebene kann dies als eine *Oder*-Verknüpfung der Beschreibung betrachtet werden. Die Spalte „Ausgabe“ zeigt an welche Ausgabe das Neuron erzeugt wenn das genannte Konzept erkannt wird. Dies ist notwendig, da

Manche Einträge enthalten ein „?“. In solchen Fällen war es nicht möglich eine kompakte oder verständliche Beschreibung zu finden. Dies trat auf, wenn die Regelmenge besonders komplex war und/oder viele verschiedene Attribute behandelte. Auch kann es sein, dass die Beschreibung genauso lang wie die Regelmenge wäre. In einem solchen Fall wurde auch bei der Ausgabe lediglich ein „-“ vermerkt.

Die Neuronen werden in dieser Dokumentation nach dem Schema <n|d>_<LayerId>_<NeuronId> benannt. Der erste Buchstabe gibt die Variante des Datensatzes an, zu dessen Netz dieses Neuron gehört.

Ein *n* bezeichnet die normalisierte und ein *d* die diskretisierte Version. Der Layer Index beginnt bei 0, welches das Input Layer beschreibt. Entsprechend bezeichnet 1 das Hidden Layer und 2 das Output Layer. Dies unterscheidet sich etwas von der technischen Umsetzung innerhalb der Software. Dort bezeichnet ein vorangestelltes x die Eingabeattribute des Klassifizierers und ein f das Klassifikationsergebnis. Diese Notation findet sich auch in den abgedruckten Regeln.

Im Folgenden werden einige Neuronen näher betrachtet. Diese dienen als Beispiele für Schwierigkeiten und Probleme die während der Interpretation auftreten. Weiterhin sollen einige Konzepte näher erläutert werden, die besonders interessante Eigenschaften aufweisen oder bei denen ein semantischer Zusammenhang zwischen scheinbar willkürlichen Regeln entdeckt wurde.

5.3.1 Diskretisierter Datensatz

d_1_0

```
1 (education-num='(8.5-10]' <= 0) and (education-num='(5.5-7]' <= 0) => f_1_0=0.0 (10931.0/624.0)
2  => f_1_0=1.0 (16869.0/2067.0)
```

Listing 5.1: Regelmenge von Neuron d_1_0

In Listing 5.1 ist die Regelmenge des Neurons d_1_0 zu erkennen. Jede Zeile repräsentiert eine Regel. Im vorliegenden Fall gibt es genau zwei Regeln, wobei eine davon die Default-Regel ist. Die Zahlen in den Klammern hinter der Regel geben Aufschluss über die zugehörige Abdeckung der Beispiele im Trainingsset. Die erste Zahl nennt die insgesamt abgedeckten Beispiele, die zweite Zahl nennt die Anzahl der davon fehlerhaften.

In der ersten Regel sind Bedingungen zu sehen, die lediglich Werte ausschließen. Wenn ein Attribut über sehr viele mögliche Werte verfügt, wie beispielsweise education oder andere diskretisierte Attribute, dann liefert der Ausschluss eines einzelnen Wertes nur wenig Wissen für eine Interpretation.

Grundsätzlich kann immer versucht werden die Darstellungsform der Regel(n) zu ändern, um so die Interpretationsmöglichkeit zu erhöhen. Die Regeln unterliegen den grundlegenden Bedingungen der Aussagenlogik und können entsprechend umgeformt werden. Die Idee ist es zu betrachten, welche Bedingungen notwendig sind, damit die Default-Regel Anwendung findet. Bei der obigen Regelmenge ist leicht zu erkennen, dass dies umgeformt werde kann zu:

```
1 (education-num='(8.5-10]' = 1) => f_1_0=1.0
2 (education-num='(5.5-7]' = 1) => f_1_0=1.0
3  => f_1_0=0.0
```

Hierbei ist nun erkennbar, dass dieses Neuron den Bereich 6-10 von education-num abdeckt. Dies entspricht Personen mit einem Bildungsniveau zwischen der 10. Klasse bis zum Beginn des College. Das durchschnittliche Bildungsniveau im Datensatz beträgt leicht über 10. Es gibt nur relativ wenige Personen, deren Bildungsniveau unter 9 liegt. Damit lässt sich dieses Neuron so interpretieren, dass es die gering bis durchschnittlich gebildeten Personen erkennt.

d_1_4

```
1 (education=Some-college >= 1) and (age='(-inf-24.3]' <= 0) and ... and (age='(53.5-60.8]' <= 0) and ... ↙
      => f_1_4=1.0 (2973.0/47.0)
2 ...
3 (occupation=Prof-specialty >= 1) and (race=White >= 1) and (age='(53.5-60.8]' <= 0) and (age='(-inf-24.3]'↙
      <= 0) => f_1_4=1.0 (1712.0/134.0)
4 ...
```

Listing 5.2: Auszug aus der Regelmenge von Neuron d_1_4

Neuron	Ausgabe	Konzeptbeschreibung
d_1_0	1	Niedrige bis durchschnittliche Bildung
d_1_1	0	HS-Abschluss 41-50 Stunden pro Woche Arbeitszeit Verheiratet und in der Privatwirtschaft
d_1_2	-	?
d_1_3	1	31-40 Stunden pro Woche Arbeitszeit und College besucht
d_1_4	1	College Abbrecher oder Akademiker
d_1_5	1	16-24 Jährige Mechaniker Akademiker
d_1_6	1	HS-Abschluss Manager
d_1_7	0	Verheiratet
d_1_8	1	Familienhaushalt
d_1_9	1	Monteure und Bedienpersonal in der Privatwirtschaft Weiße Dienstleister in der Privatwirtschaft Ausländer in der Privatwirtschaft
d_1_10	1	Mechaniker Männer im Alter 24-38
d_1_11	1	Nie verheiratete Kinder, die bei den Eltern wohnen Ehemänner Mechaniker
d_1_12	1	Unverheiratete Weiße in den USA
d_1_13	1	HS-Abschluss College Abschluss
d_1_14	1	Ehemänner
d_1_15	0	Mechaniker College besucht (nicht abgeschlossen) Unverheiratet
d_1_16	1	41-50 Stunden pro Woche Arbeitszeit Mechaniker
d_1_17	1	31-40 Stunden pro Woche Arbeitszeit und Schulabschluss bis angefangenes College
d_1_18	1	31-40 Stunden pro Woche Arbeitszeit
d_1_19	1	Ehemänner Männer in einer Familie

Tabelle 5.1: Konzeptübersicht (diskretisiert)

Bei diesem Neuron ist es möglich eine begründete Vermutung über die Semantik zu treffen, die über die reinen Bedingungen hinausgeht. Der education Wert „Some-College“ beschreibt einen angefangenen aber (noch) nicht abgeschlossenen College Besuch. Üblicherweise ist ein Jugendlicher 18-22 Jahre alt wenn er das College besucht. Da diese Gruppe hier jedoch gleichzeitig ausgeschlossen wird, kann dies so interpretiert werden, dass es sich um College-Abbrecher handelt. Eine weitere, wenn auch vermutlich kleinere Gruppe, dürften Personen sein, die bereits im Berufsleben stehen und nun doch einen College-Abschluss anstreben. In der Regelmenge existiert darüber hinaus noch eine Regel, die einen gewissen Gegensatz aufweist. Dort wird nach Personen mit Akademikerberufen gefiltert. Im Normalfall ist für diese ein abgeschlossenes Studium notwendig. Zusammenfassend filtert diese Neuron also nach Personen die zumindest eine Zeit eingeschrieben waren und die ihr Studium entweder abgebrochen oder abgeschlossen haben und nun einen passenden Berufs ausüben.

d_1_19

```
1 (relationship=Husband <= 0) and (relationship=Not-in-family <= 0) => f_1_19=0.0 (9099.0/622.0)
2 (relationship=Husband <= 0) and (sex=Male <= 0) and (race=Black <= 0) => f_1_19=0.0 (3081.0/672.0)
3  => f_1_19=1.0 (15620.0/2851.0)
```

Listing 5.3: Regelmenge von Neuron d_1_9

Diese Regelmenge kann wieder am besten durch Invertierung interpretiert werden. Es ist leicht zu sehen, dass hierbei im Wesentlichen eine Erkennung von (Ehe-)Männern stattfindet.

Zusammen mit Neuron 11 sind diese auch die Hauptkriterien der Ausgabeschicht zur Unterscheidung Mann / Frau. Es ist erstaunlich, dass in keinem Neuron im Hidden Layer eine Abfrage nach dem rein weiblichen Attributwert „relationship= Wife“ stattfindet. Wird die Ausgabeschicht und dort die Rekonstruktion von „relationship=Wife“ betrachtet, ist jedoch festzustellen, dass diese misslingt. Die Regelmenge für jenes Neuron besteht lediglich aus der Default-Regel. Somit hat vermutlich bereits der Autoencoder für diesen Wert keine geeignete Rekonstruktion gelernt, obwohl dies leicht durch Negation der Neuronen für die Erkennung von Männern realisiert werden könnte.

Allgemeine Beobachtungen

Es ist interessant, dass es mehrere Neuronen gibt, die eine Vorhersage bezüglich der Arbeitszeit im Bereich 31-50 Stunden treffen. Es scheint, als ob es dem Autoencoder nicht möglich ist diese Werte aus den anderen Konzepten zu rekonstruieren. Es ist offenbar nötig diese Werte durch das Hidden Layer „durchzureichen“. Werden die Ausgabeneuronen für 31-40 und 41-50 betrachtet, benutzen diese hauptsächlich die Neuronen 16, 17 und 18 zur Vorhersage. Das andere verwendete Neuronen nur mit geringer Relevanz in den Regeln auftauchen, untermauert den Verdacht des „Durchreichens“.

Es ist wenig überraschend, dass die Rekonstruktion der Arbeitszeit eine schwierige Aufgabe ist, da etwa 72% aller Datensätze in diesem Intervall liegen. Vermutlich existieren keine linearen Abhängigkeiten oder Zusammenhänge, die es erlauben die Rekonstruktion durch andere Daten zu stützen.

Ein weiterer interessanter Aspekt in Bezug auf die Arbeitszeit ist bei Neuron 17 und 18 feststellbar. In Tabelle 5.1 sind deren Konzepte vereinfacht dargestellt. Tatsächlich scheinen die beiden Neuronen gegensätzliche Personengruppen abzubilden. In Neuron 17 findet sich zusätzlich die Bedingung (education↙ −num=’(8.5−10]’ >= 1) wohingegen sie in Neuron 18 invertiert ist (education−num=’(8.5−10]’ <=↙ 0).

5.3.2 Normalisierter Datensatz

n_1_6

Neuron	Ausgabe	Konzeptbeschreibung
n_1_0	1	Akademiker oder angefangenes College
n_1_1	1	War oder ist verheiratet und besitzt einen Bachelor-Abschluss Wohnt bei den Eltern War verheiratet und lebt alleine
n_1_2	1	Ehemann und Mechaniker Männlicher Manager und ist oder war verheiratet Regierungsmitarbeiter Aufseher
n_1_3	1	Mechaniker Akademiker Wohnt bei Eltern
n_1_4	1	Männliche Bachelor-Absolventen Ehemänner
n_1_5	1	US-Bürger und Manager US-Bürger und verheirateter Verkäufer US-Bürger und Regierungsmitarbeiter
n_1_6	1	Nie verheiratet und allein oder bei Eltern lebend Selbstständig
n_1_7	0	Angefangenes College aber kein Manager Verkäufer Geringere Bildung (bis 12. Klasse)
n_1_8	1	Ehemann
n_1_9	1	Ehemann Nie verheiratete Männer mit HS Abschluss oder angefangenem College
n_1_10	1	Weiße Personen
n_1_11	0	Ist oder war verheiratet und ist ein Manager Ist oder war verheiratet und hat maximal einen Abschluss der 12. Klasse
n_1_12	0	Unverheiratet aber lebt nicht alleine Manager mit akademischem Grad.
n_1_13	0	Geschieden Manager mit akademischem Grad.
n_1_14	0	Frauen, die nie verheiratet waren und nicht im Büro arbeiten
n_1_15	0	Verheiratet
n_1_16	0	Frauen, die verheiratet sind oder waren, aber nicht alleine leben Verkäufer Geschieden
n_1_17	1	Weiße ohne High School oder Bachelor-Abschluss Weiße Verkäufer
n_1_18	-	?
n_1_19	0	Nicht in der Privatwirtschaft tätig Einfache Arbeitskräfte im Privatsektor Akademiker im Privatsektor

Tabelle 5.2: Konzeptübersicht (normalisiert)

```
1 ...
2 (marital-status=Never-married >= 1) and (education-num >= 0.6) and (occupation=Other-service <= 0) and (↙
      relationship=Unmarried <= 0) and (occupation=Tech-support <= 0) and (workclass=Local-gov <= 0) and ↙
      (occupation=Machine-op-inspct <= 0) and (relationship=Other-relative <= 0) => f_1_6=1.0
3 ...
```

Listing 5.4: Auszug aus der Regelmenge von Neuron n_1_6

Wenn eine Regelmenge aus mehreren Regeln besteht ist eine Invertierung häufig schwierig. Die entstehende Regelmenge kann zum Teil sogar komplexer als die nicht invertierte Version sein. Das obige Beispiel zeigt eine Regel aus einer nicht invertierbaren Regelmenge und demonstriert die Schwierigkeit mancher Regeln. Nahezu alle Bedingungen prüfen auf 0 und schließen dabei nur Werte aus. In vielen Fällen sind es jedoch nicht genug Ausschlüsse von einem Attribut, so dass eine fundierte Aussage getroffen kann, welche Attributwerte die Kernaspekte sind.

Wird bei der obigen Regel *relationship* betrachtet, so ist zunächst zu erkennen, dass *Other-relative* und *Unmarried* ausgeschlossen werden. Da keine weiteren expliziten Bedingungen vorhanden sind, ist es nun notwendig die semantischen Zusammenhängen der Attribute für weitere Hinweise zu nutzen. Die Bedingung *marital-status=Never-married* liefert das Wissen, dass keine Ehe vorliegt. Dementsprechend können die Werte *Husband* und *Wife* nicht auftreten. Damit verbleiben *Own-Child* und *Not-in-family* als einzige mögliche Werte.

n_1_19

```
1 (workclass=Private <= 0) => f_1_19=0.0 (7355.0/379.0)
2 (education-num <= 0.533333) and (occupation=Other-service >= 1) and (native-country=United-States >= 1)↙
      => f_1_19=0.0 (1327.0/201.0)
3 (education-num <= 0.533333) and (occupation=Transport-moving >= 1) and (native-country=United-States ↙
      >= 1) => f_1_19=0.0 (787.0/26.0)
4 (occupation=Machine-op-inspct >= 1) and (education-num <= 0.533333) and (native-country=United-States ↙
      >= 1) => f_1_19=0.0 (1136.0/219.0)
5 (occupation=Prof-specialty >= 1) and (education=Bachelors <= 0) and (education=Some-college <= 0) => ↙
      f_1_19=0.0 (972.0/243.0)
6  => f_1_19=1.0 (16223.0/2019.0)
```

Listing 5.5: Regelmenge von Neuron n_1_19

Grundsätzlich geht JRip die Regelmenge von oben nach unten durch und prüft, ob eine Regel das Beispiel abdeckt. Somit wird die zweite Regel nur geprüft, wenn nicht bereits die erste das Beispiel abdeckt. Dies kann genutzt werden um weitere Informationen zu erhalten, die nicht explizit in den nachfolgenden Regeln modelliert sind. Bei der Regelmenge von Neuron n_1_19 besitzt die erste Regel nur eine einzige Bedingung. Dies macht es sehr einfach zu bestimmen, welche Beispiele sie nicht erfüllen. Damit ist implizit bekannt, dass alle nachfolgenden Regeln *workclass=Private* $= 1$ voraussetzen.

Ein interessanter Aspekt dieses Neurons ist es, dass es Personen aus verschiedenen Bildungsebenen vereint. Da sind zum einen Arbeitskräfte mit einer Bildung bis zum High-School Abschluss (Regel 4). Der größte Teil aller Personen in dieser Berufsgruppe verfügt über dieses Bildungsniveau. Dennoch schließt es die besser ausgebildeten Personen dieser Berufsgruppe, die vermutlich die Vorarbeiter und Aufseher Positionen innehaben, aus. Weiterhin werden auch Logistikkräfte abgedeckt, die tendenziell eher als einfachere Berufe angesehen werden können.

Zum anderen wird der nächsten Regel nach Akademikerberufen gefiltert. Diese haben im Normalfall ein abgeschlossenes Studium mit mindestens einem Bachelor Abschluss. Diese Regel jedoch ist weitaus strenger und setzt mindestens einen Master voraus. Auch wenn diese Regel im Vergleich mit den anderen einen erhöhten Fehleranteil hat, ist es erstaunlich welcher Gegensatz zwischen diesen Gruppen liegt. Es

liegt nahe, dass eine unbekannte Gemeinsamkeit existiert, die nicht durch die vorliegenden Bedingungen modelliert und ausgedrückt wird.

Bei stärkerem Pruning werden die berufsbezogenen Bedingungen durch neue ersetzt, die dann einzelne Berufe ausschließen. Die Filterung bis zum High-School Abschluss bleibt dabei erhalten. Da der Fehler der Default-Regel dabei deutlich ansteigt, legt dies nahe, dass die gefilterten Berufsgruppen relevant sind. Eine semantische Beziehung außerhalb der Regeln konnte nicht gefunden werden.

5.3.3 Unterschiede und Gemeinsamkeiten von diskretisiertem und normalisiertem Datensatz

Wenn beide Datensatzvarianten miteinander verglichen werden fällt auf, dass der normalisierte Datensatz etwas schwieriger zu interpretieren ist. Ein Vergleich der Neuronenbeschreibungen in den Tabellen 5.1 und 5.2 zeigt, dass diese beim diskretisierten Datensatz etwas einfacher sind. Es war häufiger möglich die Aufgabe eines Neurons in weniger Worte zu fassen als dies beim bei der normalisierten Variante möglich war. Natürlich sollte die Anzahl der Worte nicht als Metrik zu Interpretierbarkeit gesehen werden, da die Einschätzung des Kernaspekts eine subjektive Angelegenheit ist.

Die subjektiv schlechtere Interpretierbarkeit zeigt sich tendenziell auch bei der durchschnittlichen Anzahl von Regeln pro Neuron. Beim normalisierten Datensatz ist diese ist im Vergleich zum diskretisierten Datensatz leicht erhöht. Dieser verfügte über etwa 6 Regeln, wohingegen die normalisierte Version 6,5 benötigt. Dies ist besonders erstaunlich, wenn die verminderte Kompression berücksichtigt wird. Der normalisierte Datensatz besitzt 103 Eingabeneuronen und damit deutlich weniger als die diskretisierte Variante mit 148. Damit muss weniger weniger stark komprimiert werden, um mit 20 Neuronen im Hidden Layer auszukommen. Entsprechend wäre zu erwarten gewesen, dass das Hidden Layer und die Regelmengen einfacher aufgebaut werden könnten.

Es fällt bei beiden Varianten auf, dass Neuronen existieren, die Gruppen ohne erkennbaren Zusammenhang abdecken. Um zu verstehen wofür und wie der Autoencoder diese Neuronen nutzt, ist es notwendig die Ausgabeschicht zu untersuchen. In den Ausgabeneuronen ist gut zu erkennen, wie das Netz geschickt die vorherigen Neuronen nutzt, um die Eingabe zu rekonstruieren. Um dies zu demonstrieren sei als Beispiel das Neuron n_2_36 betrachtet.

```
1 (x_1_7 <= 0) and (x_1_5 >= 1) and (x_1_13 >= 1) and (x_1_1 >= 1) and (x_1_12 >= 1) and (x_1_11 >= 1) ↙
      => f_2_36_occupation=Sales=1.0 (1110.0/0.0)

2 (x_1_7 <= 0) and (x_1_5 >= 1) and (x_1_13 >= 1) and (x_1_14 <= 0) => f_2_36_occupation=Sales=1.0 ↙
      (1089.0/183.0)

3 (x_1_7 <= 0) and (x_1_16 <= 0) and (x_1_11 >= 1) => f_2_36_occupation=Sales=1.0 (1487.0/643.0)
4  => f_2_36_occupation=Sales=0.0 (24114.0/483.0)
```

Listing 5.6: Regelmenge von Neuron n_2_36

Dies trifft die Vorhersage bezüglich *occupation=Sales* und verwendet dafür primär die Neuronen n_1_7 und n_1_5. Ersteres deckt Verkäufer ab und schließt Manager aus, wohingegen zweiteres sowohl Manager als auch Verkäufer abdeckt. Entsprechend kann durch diese beiden Neuronen bereits gut entschieden werden, ob eine Person im Verkauf tätig ist. Das sowohl die erste als auch die zweite Regel nur einen geringen Fehler besitzen, zeigt dass das Hidden Layer für die Vorhersage dieses Attributwerts gut geeignet ist. Bei anderen Neuronen in der Ausgabeschicht zeigt sich eine ähnliche Verwendung des Hidden Layers. Ausnahmen stellen hierbei die „durchgereichten“ Werte da wie „Married-civ-spouse“ oder beim diskreten Datensatz manche Werte bezüglich der Arbeitszeit. Diese verfügen auch in der Ausgabeschicht nur über sehr einfache Konzepte und übernehmen im Wesentlichen das zugehörige Ergebnis aus dem Hidden Layer.

Es existieren jedoch auch Neuronen, bei den es möglich ist eine Gemeinsamkeit aller Regeln und Aspekte zu finden. In solchen Fällen ist es möglich eine Formulierung zu finden, die das gesamte Neuron beschreibt. Allerdings waren dies in allen aufgetretenen Fällen relativ einfache Konzepte. Diese orientierten sich stark an grundlegenden Aspekten, die auch als Attribute im Datensatz vorhanden waren. Beispielsweise sind dies Aspekte, die die Bildung oder das Geschlecht betreffen.

Es ist auffällig, dass die Regeln für den normalisierten Datensatz als einziges kontinuierliches Attribut *education-num* verwenden. Während in der diskretisierten Version noch andere Attribute wie *age* oder *hours-per-week* Anwendung finden, sind sie auch bei niedriger Pruningstärke nicht mehr vorhanden. Es liegt die Vermutung nahe, dass der Autoencoder oder der Regellerner die größeren Abstände bei der One-Hot Kodierung bevorzugt. Als diese in der normalisierten Version nicht mehr vorhanden waren, wurden stattdessen andere Konzepte entwickelt. Dies zeigt sich auch insofern, als dass es keine dedizierten Neuronen mehr für die Arbeitszeit gibt wie noch in der diskreten Variante.

Generell zeigt sich, dass es nur wenige Konzepte gibt, die beiden Datensatzvarianten gemeinsam haben. Dies sind einfache Dinge wie der Heiratsstatus oder das Geschlecht. Bei diesen gab es konzeptionelle identische Neuronen bei beiden Datensätzen. Bei den Neuronen, die mehrere Personengruppen und damit komplexeren Sachverhalten abbilden, gibt es Ähnlichkeiten zwischen den Neuronen, ohne das diese sich in ihrer Gesamtheit ähnlich wären. Dies scheint ein Ergebnis der zuvor angesprochenen Tendenz des Netzes zu sein, die Ausgaben über Kombination von sich ausschließenden Neuronen zu erzeugen.

Auch bei der generellen konzeptionellen Ausrichtung gibt es Unterschiede. Die normalisierte Version fokussiert stark Heiratsstatus, Geschlecht und einen College Besuch. Im Gegensatz dazu standen bei der normalisierten Variante die Arbeitszeit, ethnische Zugehörigkeit und der High-School Abschluss im Vordergrund.

5.4 Separate Klassifikation

Die Eignung des Hiddel Layers für eine Klassifikation lässt sich bestimmen, indem versucht wird mit dessen Konzepten ein sinnvolles Modell zu lernen. Wenn dies eine gute Leistung erbringen kann, verfügt das Hidden Layer über geeignete Konzepte. Vor der Betrachtung der Regelmenge ist es notwendig zunächst eine quantitative Bewertung der eigentlichen Klassifikationsleistung vorzunehmen. Damit kann auch gegebenenfalls eine Parameteranpassung zum Verbessern der Ergebnisse vorgenommen werden. Durch eine weitere semantische Betrachtung kann geprüft werden, ob die genutzten Konzepte sinnvoll sind.

5.4.1 Klassifikationsleistung

Um die Ergebnisse im Kontext einzuordnen, ist es wichtig einen Bezugspunkt zu haben. Es bietet sich an dafür die Klassifikationsleistung auf den Originaldaten zu verwenden. Dabei ist mit „Originaldaten“ nicht der unmodifizierte Datensatz, sondern die beiden angepassten Datensatzvarianten, die als Eingabe für die Autoencoder dienten, gemeint.

In Tabelle 5.3 sind die Ergebnisse als F1-Score bei verschiedenen Pruningstärken zu sehen. Die besten Ergebnisse sind dabei grün hervorgehoben. Es zeigt sich, dass diese bei relativ niedrigen Pruningstufen erreicht werden. Die normalisierte Version ist etwa 1,7% und damit nennenswert besser als die diskretisierte Version. Dies lässt vermuten, dass mindestens ein kontinuierliches Attribut verwendet wird, das die Klassifikation verbessert. Es ist wenig überraschend, dass die Diskretisierung die Ergebnisse verschlechtert, da bei dieser bereits prinzipiell Informationen verloren gehen, die nicht wiederhergestellt werden können.

Tabelle 5.4 zeigt die Klassifikationsleistung basierend auf dem Hidden Layer. Wieder sind die besten Ergebnisse grün hinterlegt. In der Spalte „Pruningstärke“ finden sich zwei Konfigurationen „1000/100“ und „400/100“. Die Zahl von dem / bezeichnet die Pruningstärke für die Regeln der Encoderstufe, wohingegen die zweite Zahl die Pruningstärke für die separate Klassifikation bezeichnet. Die Werte 1000 und 400 wurden bereits in Abschnitt 5.2 beleuchtet. Diese werden daher erneut für den Vergleich herangezogen, um sowohl eine stark als auch eine schwächer geprunte Encoder-Stufe zu berücksichtigen. Der Wert 100 bezieht sich auf die Pruningstärke mit der die separate Klassifizierung gelernt wurde. In nicht abgebildeten Vortests hat sich gezeigt, dass 100 hierbei ein guter Wert ist, da sich bei geringeren

Pruningstärke	Datensatzvariante	
	Diskretisiert	Normalisiert
50	81,88	83,58
75	81,99	83,07
100	81,64	82,96
200	81,50	82,2
400	81,28	80,82

Tabelle 5.3: Klassifikationsleistung der Originaldaten (in %)

Variante	Pruningstärke	Größe des Hidden Layers						Vergleichswert
		10	15	20	30	40	50	
Diskretisiert	1000/100	78,16	79,19	80,64	81,92	81,27	81,89	81,99
	400/100	77,98	79,40	81,07	81,78	81,61	81,67	
Normalisiert	1000/100	79,01	80,20	80,50	80,96	81,01	80,87	83,58
	400/100	80,12	79,86	80,78	80,09	80,54	80,46	

Tabelle 5.4: Klassifikationsleistung basierend auf dem Hidden Layer mit verschiedenen Größen (in %)

Stärken die Leistung nur minimal verbessert. Für beiden Datensatzvarianten sind die Ergebnisse für verschiedenen Größen des Hidden Layers erkennbar. Zur besseren Vergleichbarkeit wurde der zugehörige Referenzwert aus Tabelle 5.3 in der letzten Spalte nochmals angegeben.

Es zeigt sich, dass mit dem diskretisierte Datensatz die gleiche Leistung wie mit den Originaldaten erreicht werden kann. Bei der normalisierten Variante war dies nicht möglich. Dort war die separate Klassifikation im besten Fall 2,6 % schlechter und bleibt damit noch hinter der diskretisierten Version zurück. Es ist zu vermuten, dass die konzeptionell bedingte Binärität der Neuronen im Hidden Layer implizit eine Diskretisierung erzeugt. Da diese vermutlich nicht optimal stattfindet, kann damit auch nicht das Niveau der Referenzmessung erreicht werden. Aus dem diskreten Datensatz konnten offensichtlich die besseren Klassifikationskonzepte abgeleitet werden.

Bereits in Abschnitt 5.1 wurde festgestellt, dass ein größeres Hidden Layer beim normalisierten Datensatz nicht notwendigerweise eine bessere Leistung erzeugt. Dieses Verhalten ist auch bei der separaten Klassifikation zu beobachten. Dort zeigen sich die optimalen Ergebnisse bei 40 Neuronen. Dies entspricht ziemlich genau der Größe, bei dem auch der Autoencoder für diese Variante sein Optimum erreichte.

Erstaunlicherweise kann mit nur 10 Neuronen im Hidden Layer noch ein F1-Score von bis zu 80% erreicht werden. Dies entspricht einer Kompression von über 90% und ist damit eine erhebliche Reduktion. An dieser Stelle sollte jedoch die Möglichkeit von Feature Subset Selection auf den Originaldaten nicht außer Acht gelassen werden. Damit wäre es unter Umständen möglich ebenfalls eine Kompression zu erreichen. Ob und inwieweit dies vergleichbar ist, wurde nicht untersucht.

Insgesamt zeigt sich, dass obwohl der Autoencoder keine Konzepte im Hinblick auf eine Klassifikationsaufgabe gelernt hat, die Konzepte im Hidden Layer geeignet sind nicht nur eine Rekonstruktion sondern auch eine Klassifikation zu ermöglichen. Dabei können Ergebnisse erzielt werden, die mit denen der Originaldaten übereinstimmen oder nur leicht schlechter sind. Um die besten Ergebnisse zu erreichen, empfiehlt es sich, bei binären Neuronen im Hidden Layer, nur diskretisierte Eingabedaten zu verwenden.

5.4.2 Verwendete Konzepte bei den Originaldaten

Bevor betrachtet wird, welche Konzepte die separate Klassifkation aus dem Hidden Layer nutzt, ist es vorteilhaft zunächst zu verstehen, welche Konzepte auf den Originaldaten gelernt werden. Basierend dar-

auf kann untersucht werden, ob Gemeinsamkeiten existieren oder ob die separate Klassifikation anderen Aspekte benutzt.

Zuvor wurde bereits untersucht bei welchen Pruningstärken die besten Ergebnisse erreicht werden. Dennoch sollen nicht die Regeln jener Konfigurationen betrachtet werden, da diese aufgrund des schwachen Prunings noch recht umfangreich sind. Es wurde für beide Varianten entschieden die Regeln für Pruningstärke 100 zu verwenden. Diese haben in beiden Fällen nur eine gering schwächere Leistung, aber eine deutlich bessere Verständlichkeit. Beispielsweise sinkt bei der diskretisierten Version die Regelzahl von 11 auf 5 und dabei wird lediglich 0,3% Leistung eingebüßt. Listing 5.7 und 5.8 zeigen vollständigen Regelmengen der beiden Varianten. Eine „1“ als Vorhersage signalisiert in beiden Fällen ein Einkommen von über 50 000$

```
1 (marital-status=Married-civ-spouse >= 1) and (education-num='(11.5-13]' >= 1) => f_1_0=1.0
2 (marital-status=Married-civ-spouse >= 1) and (occupation=Prof-specialty >= 1) => f_1_0=1.0
3 (marital-status=Married-civ-spouse >= 1) and (occupation=Exec-managerial >= 1) => f_1_0=1.0
4 (marital-status=Married-civ-spouse >= 1) and (education=Masters >= 1) => f_1_0=1.0
5  => f_1_0=0.0
```

Listing 5.7: Regelmenge für Originaldaten (diskretisiert)

Es fällt unmittelbar auf, dass die Regeln der diskretisieren Varianten sehr einfach zu verstehen sind. Die Regelmenge ist recht klein und verfügt über Regeln, die nur wenige Bedingungen besitzen. Bereits in Abschnitt 5.3.1 fiel dies bei der Analyse des Hidden Layers auf. Dies deutet darauf hin, dass der diskretisierte Datensatz generell besser und einfacher mit Regeln zu repräsentieren ist als der normalisierte.

Semantisch fällt sofort die starke Präsenz von *marital-status=Married-civ-spouse* auf. Da diese bei allen Regeln vorhanden ist, handelt es sich offenbar um ein wichtiges Grundkriterium der Gehaltserkennung. Bei den Regeln 1, 2 und 4 fällt der Bezug zur Bildung ins Auge. Der Intervall 12-13 umfasst Personen mit akademischen Grad sowie Bachelor Absolventen. Regel 4 erweitert die Gruppe um Master Absolventen. Das Bild wird abgerundet durch Regel 2, die Personen in Akademikerberufen auswählt. Damit lassen sich diese drei Regeln zusammenfassend beschreiben als Personen, die sehr gut ausgebildet sind.

Eine zweite wichtige Personengruppe sind Manager. Diese gehören zum größten Teil auch der vorigen Gruppe an und verfügen über eine College Ausbildung. Ein nicht unerheblicher Anteil jedoch verfügt über einen High-School oder geringeren Abschluss. Entsprechend ist bei Managern offenbar auch eine geringere Bildung nicht hinderlich.

Insgesamt ist die Eingrenzung auf Manager und sehr gut ausgebildete Personen nicht überraschend, da diese häufig sehr gut bezahlte Berufe wahrnehmen. Das jedoch der Heiratsstatus ein entscheidendes Merkmal ist, war nicht abzusehen.

```
1 (marital-status=Married-civ-spouse >= 1) and (education-num >= 0.733333) and (capital-gain >= 0.051781) ↙
      => f_1_0=1.0
2 (marital-status=Married-civ-spouse >= 1) and (education-num >= 0.733333) and (capital-loss >= 0.418962) ↙
      and (capital-loss <= 0.453857) => f_1_0=1.0
3 (marital-status=Married-civ-spouse >= 1) and (education-num >= 0.6) and (education-num >= 0.8) and (age ↙
      >= 0.260274) and (age <= 0.520548) => f_1_0=1.0
4 (marital-status=Married-civ-spouse >= 1) and (education-num >= 0.6) and (occupation=Exec-managerial >= ↙
      1) => f_1_0=1.0 (851.0/310.0)
5 (marital-status=Married-civ-spouse >= 1) and (education-num >= 0.6) and (education-num >= 0.8) and (↙
      hours-per-week >= 0.397959) => f_1_0=1.0
6 (marital-status=Married-civ-spouse >= 1) and (capital-gain >= 0.051781) => f_1_0=1.0
7 (marital-status=Married-civ-spouse >= 1) and (age >= 0.287671) and (education-num >= 0.466667) and (↙
      capital-loss >= 0.418962) => f_1_0=1.0
8  => f_1_0=0.0
```

Listing 5.8: Regelmenge für Originaldaten (normalisiert)

Genauso wie bei der vorigen Regelmenge verhält sich der normalisierte Datensatz (Listing 5.8) ebenfalls konform zu den Erfahrungen aus dem Hidden Layer. Es gibt vergleichsweise viele Regeln, die zudem mit zahlreichen Bedingungen ausgestattet sind. Gleichzeitig existiert jedoch ein Unterschied zu den Regeln, die im Hidden Layer aufgetreten sind. Dort wurden nahezu keine Referenzen auf die kontinuierlichen Eingabeattribute festgestellt. Bei diesen Regeln, die direkt aus den Originaldaten gelernt wurden, ist dies grundverschieden. Sämtliche vorhandenen kontinuierlichen Attribute werden mindestens einmal verwendet und oft sogar mehrmals. In dieser Regelmenge tauchen erstmals überhaupt die Attribute *capital-gain* und *capital-loss* auf. Diese wurden zuvor weder von Neuronen im Hidden Layer für den diskretisierten als auch den normalisierten Datensatz verwendet. Als mögliche Ursache kommt hierfür in Betracht, dass die genannten Attribute vollständig aus anderen Werten rekonstruiert werden können. In einem solchen Fall wäre es für das Netz beziehungsweise das Hidden Layer unnötig diese Bedingungen zu verwenden. Bei der diskreten Varianten zeigt sich jedoch, dass die Regeln der Ausgabeneuronen für *capital-gain/-loss* lediglich die Default-Regeln sind. Zumindest bei dieser Datensatzvariante ist damit klar, dass es dem Netz nicht gelungen ist eine sinnvolle Struktur dafür zu lernen. Möglicherweise war die Wahl der Diskretisierungsintervalle im Vorverarbeitungsschritt ungeeignet. Bei der normalisierten Variante hingegen ist es denkbar, dass die numerischen Unterschiede zu gering waren. Die Werte von *capital-gain* bewegen sich im Bereich [0, 100000). Wenn dies normalisiert wird ergeben sich bereits bei leichtem Rauschen große Differenzen und entsprechend schwierig kann die Rekonstruktion sein.

Semantisch zeigen sich sowohl Ähnlichkeiten als auch deutliche Unterschiede zu der Klassifizierungsregelmenge in Listing 5.7. Am leichtesten ist wieder der Heiratsstatus als wichtigstes Kriterium erkennbar. Als zweitstärkste Bedingung folgt wie zuvor das Bildungsniveau. Der Wert 0,733 entspricht dabei einer 12 und ist damit ebenfalls äquivalent zur vorherigen Menge. Deutlich interessanter sind bei Regel 1 und 2 die Bedingungen zu Kapitalerträgen und Verlusten. Es handelt sich in beiden Fällen um relativ große Werte im Bereich 40-50 tausend Dollar. Personen die derartig großen Kapitaländerungen erzeugen, müssen auch ein entsprechend großes Kapital besitzen. Auch wenn es grundsätzlich nicht ausgeschlossen ist, dass es sich dabei um beispielsweise Erbschaften handelt, ist es plausibel zu folgern, dass diese Personen ebenfalls über eine gehobene Position und damit ein entsprechend großes Einkommen besitzen.

Die Gruppe der Manager findet sich ebenfalls in dieser Menge wieder und wird in Regel 4 gefiltert. Ähnlich wie zuvor ist es offenbar für Manager ausreichend eine etwas geringere Bildung zu besitzen. Der Wert 0,6 entspricht dabei 10 und repräsentiert ein angefangenes College.

Die Regeln 3 und 5 sind in gewisser Weise ein Kuriosum, da das Attribut *education-num* direkt hintereinander mit verschiedenen Splitpunkten verwendet wird. Da jeweils die zweite Wiederholung deutlich strenger als die Erste ist, wird diese damit überflüssig.

5.4.3 Verwendete Konzepte aus dem Hidden Layer

```
1 (x_1_7 <= 0) and (x_1_0 <= 0) and (x_1_14 >= 1) and (x_1_6 >= 1) => f_2_0_>50K=1.0 (1147.0/247.0)
2 (x_1_7 <= 0) and (x_1_0 <= 0) and (x_1_14 >= 1) and (x_1_16 <= 0) and (x_1_19 >= 1) => f_2_0_>50K=1
      .0 (842.0/215.0)
3 (x_1_7 <= 0) and (x_1_14 >= 1) and (x_1_0 <= 0) and (x_1_13 >= 1) and (x_1_10 <= 0) and (x_1_9 <= 0)
      => f_2_0_>50K=1.0 (752.0/234.0)
4 (x_1_7 <= 0) and (x_1_4 >= 1) and (x_1_5 >= 1) and (x_1_0 <= 0) => f_2_0_>50K=1.0 (446.0/112.0)
5 (x_1_7 <= 0) and (x_1_4 >= 1) and (x_1_15 <= 0) and (x_1_14 >= 1) and (x_1_6 >= 1) => f_2_0_>50K=1.0
      (343.0/123.0)
6 (x_1_7 <= 0) and (x_1_9 <= 0) and (x_1_4 >= 1) and (x_1_10 <= 0) and (x_1_3 >= 1) and (x_1_8 <= 0) and
      (x_1_15 <= 0) => f_2_0_>50K=1.0 (439.0/215.0)
7  => f_2_0_>50K=0.0 (23831.0/4050.0)
```

Listing 5.9: Regelmenge für Separate Klassifikation (diskretisiert, Pruning: 1000/100)

Die obigen Menge repräsentiert die Regeln der separaten Klassifikation für eine Pruningstärke von 1000 im Hidden Layer und 100 für die Vorhersage.

Das Neuron 7 sticht in dieser Regelmenge mit großer Deutlichkeit hervor. Ein Blick in Tabelle 5.1 verrät, dass es sich hierbei um ein Neuron handelt, das eine Aussage über den Heiratsstatus vornimmt. Konkret handelt es sich um die Heirat mit einem Zivilisten. Dies ist exakt die gleiche Bedingung die bereits bei der Klassifizierung der Originaldaten die wichtigste Position einnahm.

Weitere Neuronen die häufig verwendet werden sind 0, 14 und 4. Dabei handelt es sich bei Neuron 0 um eine Erkennung von geringer bis durchschnittlicher Bildung. In sämtlichen Bedingungen wird dieses Neuron auf 0 geprüft. Dies bedeutet eine Negation und kann damit als eine Filterung auf überdurchschnittliche Bildung verstanden werden. Genauso wie der Heiratsstatus konnte die Notwendigkeit von hoher Bildung bereits zuvor festgestellt werden. Auch Neuron 4, das unter anderem nach Akademikern filtert, fällt in diese Kategorie. Das Neuron 14 filtert nach verheirateten Männern. Dies unterscheidet sich deutlich von den vorherigen Klassifikationsregelmengen. Das Geschlecht war dort zu keinem Zeitpunkt relevant.

In der ersten Regel findet sich bei Neuron 6 eine Aussage über den Managerberuf. Dieses Neuron filtert nach Managern und HS-Absolventen. Gleichzeitig ist aus Neuron 0 bekannt, dass eine überdurchschnittliche Bildung vorliegt und damit der HS-Abschluss ausgeschlossen werden kann. Damit lässt sich die Regel als verheiratet, überdurchschnittliche Bildung und Manager zusammenfassen. Genau diese Personengruppe wurde auch zuvor als einkommensstark klassifiziert.

Die zweite Regel ist sehr ähnlich zur ersten. Jedoch wird hierbei nicht der Managerberuf als Kriterium sondern eine überdurchschnittliche Arbeitszeit herangezogen.

Eine weitere interessante Regel ist Nummer 4. Neuron 4 als auch 5 filtern nach verschiedenen Aspekten. Da Akademiker die einzige Schnittmenge sind, kann durch Kombination beider Neuronen, nach ebenjenen gefiltert werden.

Die beiden letzten Regeln sind uninteressant, da sie mit 35 und 49% über einen ziemlich großen Fehler verfügen. Entsprechend sind ungeeignet für qualifizierte Auswertung.

```
1 (x_1_15 <= 0) and (x_1_1 >= 1) => f_2_0_>50K=1.0 (2474.0/816.0)
2 (x_1_15 <= 0) and (x_1_0 >= 1) and (x_1_7 >= 1) => f_2_0_>50K=1.0 (1657.0/506.0)
3 (x_1_15 <= 0) and (x_1_5 >= 1) and (x_1_7 >= 1) => f_2_0_>50K=1.0 (840.0/277.0)
4 (x_1_15 <= 0) and (x_1_2 <= 0) and (x_1_11 >= 1) and (x_1_17 >= 1) and (x_1_13 <= 0) => ↙
      f_2_0_>50K=1.0 (534.0/252.0)
5 (x_1_15 <= 0) and (x_1_0 >= 1) and (x_1_16 <= 0) and (x_1_14 >= 1) => f_2_0_>50K=1.0 (303.0/147.0)
6  => f_2_0_>50K=0.0 (21992.0/3027.0)
```

Listing 5.10: Regelmenge für Separate Klassifikation (normalisiert, Pruning: 1000/100)

Auch bei der Regelmenge in Listing 5.10 zeigt sich ein einzelnes Neuron sehr dominant. Aufgrund der Ergebnisse bei den früheren Ergebnisse ist es bereits einfach zu erahnen, dass es sich dabei wieder um den Heiratsstatus als Grundbedingung handelt. Interessanterweise führt ein schwächeres Pruning zu einem anderen dominanten Neuron. Wird die Regelmenge für die Pruningstärken 400/100 betrachtet, so wird dort Neuron 8 als wichtigstes ausgewählt. Die restlichen Bedingungen bleiben weitestgehend unangetastet. Neuron 8 ist einfach aufgebaut und verfügt nur über eine Bedingung. In dieser wird geprüft ob die Person ein Ehemann ist. Damit prüft auch dieses Neuron den Heiratsstatus, ist aber strenger, da es zusätzlich das männliche Geschlecht erfordert.

Generell erscheint diese Regelmenge zunächst einfacher als die vorherige. So sind bei dieser zum einen weniger Regeln vorhanden und zum anderen verwenden sie weniger Bedingungen. Bei der ersten Regel sind sogar nur zwei Neuronen involviert. Allerdings sind die verwendeten Neuronen komplexer und verfügen über mehrere abgedeckte Personengruppe. Dies erschwert es wieder zu einer Schlussfolgerung zu kommen.

Bereits die erste und kürzeste Regel lässt sich nicht so gut vereinfachen wie die komplexen Regeln aus vorherigen Regelmengen. Mit dem Wissen, dass die Person verheiratet ist, lässt sich lediglich der 3. Fall von Neuron 1 ausschließen. Damit verleiben Personen die verheiratet sind und einen Bachelor-Abschluss

besitzen sowie Verheiratete die noch bei den Eltern wohnen. Die zweite Gruppe ist unerwartet, da sie zuvor nicht in Erscheinung getreten ist. Da im Datensatz der Anteil von Verheirateten, die bei den Eltern wohnen, äußerst gering ist, handelt es sich möglicherweise um einen Fehler den der Regellerner in Kauf genommen hat.

In Regel 2 ist eine konzeptionelle Ähnlichkeit zu den Originaldaten erkennbar. Neuron 0 deckt Akademiker und Personen mit angefangenem College ab. Das nachfolgende Neuron 7 kann dann unter dieser Prämisse betrachtet werden. Damit dieses Neuron eine 1 als Ausgabe liefert, darf der Fall „Angefangenes College aber kein Manager“ nicht eintreten. Wenn das angefangene College vorliegt muss somit die Person eine Managerposition innehaben. Falls die Person Akademiker ist, lässt sich keine weitere Eingrenzung treffen. Dass die Person ein Akademiker ist, schließt bereits eine Beschäftigung als Verkäufer sowie eine geringe Bildung aus. Zusammenfassend deckt diese Regel damit verheiratete Akademiker und Manager mit angefangenem Studium ab. Diese Personengruppen sind bereits aus allen vorherigen Klassifikationsregelmengen bekannt.

Bei der dritten Regel kann durch Neuron 7 wieder der Verkäuferberuf und geringe Bildung ausgeschlossen werden. Damit lässt sich aus Neuron 7 folgern, dass Bürger aus den USA die als Manager oder bei der Regierung arbeiten durch diese Regel abgedeckt werden. Da bereits Regel 2 Manager abdeckt und dabei genereller ist, kann davon ausgegangen werden, dass sich diese Regel primär auf die Regierungsmitarbeiter fokussiert. Da dieser Aspekt ist völlig neu ist und bei keiner anderen Regelmenge beobachtet wurde, ist unklar ob es sich hierbei um einen Fehler oder tatsächlich um eine bisher unbekannte, aber relevante Beziehung handelt.

Die beiden letzten Regeln haben einen Fehler von 47 und 48 %. Da diese Werte recht hoch und auch schon sehr dicht am Abbruchkriterium von 50% sind, kann wie beim diskretisierten Datensatz davon ausgegangen werden, dass es nicht förderlich ist diese näher zu betrachten. Generell zeigt sich, dass die Regelmenge des normalisierten Datensatzes einen vergleichsweise hohen Fehleranteil hat. So liegt der Fehler bei allen Regel über 30% wohingegen er bei der normalisierten Version in den meisten Fällen um 25% liegt. In beiden Fällen kommt der hohe Fehleranteil durch das starke Pruning zustande. Wird dies reduziert, verringert sich der Fehler erheblich.

Insgesamt fällt auf, dass diese Regelmenge keine Personen abdeckt die sehr gut ausgebildet sind, aber keinen Akademiker oder Managerberuf besitzen. Es wird zwar nach einem Bachelor-Abschluss gefiltert, dieser Filter ist aber nur eine „genau dann“ Bedingung, die keine höheren Abschlüsse berücksichtigt. Die Akademiker Filterung dürfte zwar bereits einen großen Teil der gut ausgebildeten Personen abdecken, jedoch sollte nicht davon ausgegangen werden, dass jede Person mit einem Master-Abschluss auch einen akademischen Beruf ausübt.

5.4.4 Fazit

Aus den vorliegenden Ergebnissen ist ersichtlich, dass die separate Klassifikation vergleichbare Ergebnisse wie eine direkte Klassifikation erreicht. Bei den quantitativen Ergebnisse zeigt sich, dass es bei der diskreten Datensatzvariante gelingt eine gleichwertige Lösung zu erzielen. Die normalisierten Versionen ist zwar etwas schlechter, erreicht aber dennoch nahezu den Referenzwert. Dies ist wohl darauf zurückzuführen, dass beiden Techniken im Wesentlichen die gleichen Konzepte zur Klassifikation nutzen. Es wurden dabei nur geringe Abweichungen festgestellt. Die normalisierte Version bringt zwar Konzepte und Ideen ein, welche die diskrete Variante nicht modelliert, dennoch sind die Hauptkonzepte gleich.

Es zeigt sich, dass die für die Klassifikation notwendigen Konzepte zum Teil bereits explizit im Hidden Layer modelliert sind, obwohl das Netz nicht dafür ausgelegt war. Der Heiratsstatus ist in allen Fällen das mit Abstand wichtigste Kriterium zur Unterscheidung. Im Hidden Layer beider Datensatzvarianten erhielt diese Bedingung ein eigenes Neuron, welches ausschließlich diese Bedingung prüft.

Bedauerlicherweise wurden jedoch nahezu keinerlei Konzepte verwendet, die nicht bereits bei der Originaldaten-Klassifikation verwendet wurden. Offenbar war das Hidden Layer nicht in der Lage neue Beziehungen zu entdecken, die dann für die Klassifikation hätten verwendet werden können. In der normalisierten Version wurde der Bezug zu US-Bürgern und dem männlichen Geschlecht modelliert.

Dies ist etwas, dass möglicherweise ein neues Konzept ist oder aber schlicht durch den erhöhten Fehler der Regelmenge entstanden sein könnte. In Medien wie Zeitungen und Fernsehen wird von Zeit zu Zeit thematisiert, dass Frauen für gleiche Arbeit schlechter bezahlt werden. Der Datensatz ist über 20 Jahre alt und stammt aus dem amerikanischen Kulturkreis. Es ist daher nur bedingt möglich die darauf basierende Ergebnisse unter den Gesichtspunkten einer heutigen Gesellschaft zu betrachten. Da sich die Emanzipation aber tendenziell mit der Zeit verbessert hat, ist davon auszugehen, dass die Filterung nach dem Geschlecht zum damaligen Zeitpunkt zumindest plausibel war.

Vom Standpunkt der Interpretierbarkeit betrachtet, ergibt sich ein klarer Vorteil der diskretisierten Variante. Werden die Originaldaten verwendet, ergeben sich sehr einfache Regeln und entsprechend gut zu erkennende Konzepte. Bei der separaten Klassifikation war die Regelmenge vergleichsweise komplex. Dafür waren jedoch die Konzepte der verwendeten Neuronen wiederum einfach gestaltet und ließen sich leicht interpretieren. Bei der normalisierten Variante zeigte sich eine insgesamt höhere Komplexität. Dadurch, dass bereits im Hidden Layer die Konzepte häufig viele Personengruppen abgedeckten, war es teilweise notwendig weiteres Wissen, aus der Datenverteilung im Datensatz als Hinweis auf die Intentionen zu verwenden. Dies erklärt auch warum bei dieser Variante die separate Klassifikation nicht mit der direkten mithalten konnte. Durch komplexere Konzepte, die nicht für die Klassifikation ausgelegt sind, wird es vergleichsweise schwierig diese für eine neue Aufgabe zu verwenden. Darüber hinaus wurde eine implizite Diskretisierung im Hidden Layer durchgeführt, die zusammen mit den komplexen Regeln einen erhöhten Fehler verursacht hat.

5.5 Transfer von Gewichten

Wie bei der Untersuchung der separaten Klassifikation soll auch hier zunächst die Klassifikationsleistung betrachtet werden. Diese liefert bereits schnell Informationen darüber, inwieweit die Konzepte aussagekräftig sind. Auch erlaubt dies es, die Ergebnisse des Gewichtstransfers im Kontext einzuordnen und in Bezug auf andere Verfahren zu beurteilen. Dafür werden die Ergebnisse von drei verschiedene Techniken miteinander verglichen. Das Ergebnisnetz des Gewichtstransfers wird mit einem Netz verglichen, das direkt aus den Daten trainiert wurde, ohne dabei einen Autoencoder als Zwischenschritt zu verwenden. Im Folgenden wird ein solches Netz auch als „direktes Klassifikationsnetz“ bezeichnet. Das Ergebnis nach dem Gewichtstransfer wird auch „indirektes Klassifikationsnetz“ genannt. Darüber hinaus sollen die Ergebnisse auch mit denen der separaten Klassifikation verglichen werden.

Nach dem rein quantiativen Vergleich der Klassifikationsergebnisse erfolgt eine qualitative Untersuchung der Konzepte. Es wird betrachtet inwieweit sich diese im indirekten Klassifkationsnetz von denen im ursprünglichen Autoencoder unterscheiden. Darüber hinaus wird betrachtet, ob es Unterschiede oder Gemeinsamkeiten zum direkten Netz gibt. Abschließend wird dann analysiert, welche Ähnlichkeiten es bezüglich der Klassenvorhersage gibt. Es wird auch betrachtet ob diese den bekannten Mustern aus der separaten Klassifikation entsprechen.

5.5.1 Klassifikationsleistung

In Tabelle 5.5 sind die Ergebnisse für verschiedene Konfigurationen dargestellt. Die ersten Spalte gibt die verwendete Datensatzvariante an. Danach folgt die Pruningstärke bei der Regelextraktion. Es wird hierbei unterschieden zwischen dem Pruning im Hidden Layer und dem in der Vorhersage beziehungsweise Ausgabeschicht. Bei der Klassifikation auf den Originaldaten sowie der separaten Klassifikation hat sich gezeigt, dass dafür deutlich geringere Pruningwerte sinnvoll sind. Entsprechend wurden Konfigurationen in die Untersuchung aufgenommen, bei denen die letzte Schicht ebenfalls nur schwach geprunt wird. In den folgenden Spalten finden sich dann die Klassifikationsergebnisse für die separate Klassifikation („SK“), das direkte Klassifikationsnetz („Direkt“) sowie das indirekte Klassifikationsnetz („Indirekt“). Die Ergebnisse wurden für ein Netz mit 20 Neuronen im Hidden Layer ermittelt, das nachfolgend auch als Basis für die Untersuchung der Konzepte dienen soll.

Datensatz	Pruningstärke Hidden	Pruningstärke Vorhersage	Konfiguration SK	Konfiguration Direkt	Konfiguration Indirekt
Diskretisiert	-	-		83,94	83,88
	400	100	81,07	80,20	81,60
	400	400		80,18	81,36
	1000	100	81,64	70,34	79,26
	1000	1000		70,34	78,55
Normalisiert	-	-		83,55	82,9
	400	100	80,78	81,42	81,01
	400	400		81,26	80,92
	1000	100	80,5	80,93	80,96
	1000	1000		80,93	80,36

Tabelle 5.5: Klassifikationsleistung nach Gewichtstransfer (in %)

Neben den Klassifikationsergbnissen die auf den extrahierten Regeln basieren, sind auch die Ergebnisse für die Klassifikation durch das Netz abgebildet. Diese sind jeweils in der obersten Zeile jedes Datensatzes dargestellt. Dies ist auch daran erkennbar, dass für jene Einträge keine Pruningstärke für Hidden- und Vorhersageschicht angegeben ist.

Das Auffälligste ist der starke Leistungseinbruch des direkten Netzes beim diskretisierten Datensatz für höhere Pruningstufen. Durch erhöhtes Pruning sank die Leistung um nahezu 10% und erreicht damit die bisher schlechtesten Werte überhaupt. Für die Ermittlung der Ergebnisse wurden mehrere Regelextraktionen mit verschiedenen Startwerten durchgeführt. Damit entstehen mehrere Messpunkte, deren Durchschnitte in der Tabelle angegeben sind. Bei den Konfigurationen 1000/100 und 1000/1000 kam es zu der Situation, dass mehrere Extraktionen in sehr schlechte Regelmengen resultierten. Wiederholt kam es zu Mengen, bei denen lediglich immer die größte Klasse vorhergesagt wurde. Diese Mengen haben einen F1-Score von etwa 65% und beeinflussen den Durchschnitt entsprechend stark. Gleichzeitig gab es auch Ergebnisse, bei denen die Regelmenge gute Resultate lieferte und Werte von 81% erreichte.

Auch bei den Messwerten für das indirekte Netz wurden bei starkem Pruning einige deutlich schlechtere Extraktionen festgestellt. Dabei war jedoch das schlechteste Einzelergebnis mit 77% nicht so stark betroffen wie beim direkten Netz. Eine gewisse Veränderung ist zwar aufgrund des unterschiedlichen Prunings zu erwarten, jedoch sind die Unterschiede zu erheblich, um sie nur damit zu erklären. Darüber hinaus wurden beim normalisierten Datensatz keine solch starken Schwankungen beobachtet. Dies deutet auf eine generelle Schwäche des diskreten Datensatzes hin. Da das Netz selbst gute Ergebnisse liefert, scheint diese Schwäche lediglich Auswirkungen auf die Regelextraktion zu haben. Die erzeugten Regelmengen sind bei Pruningstufe 400 bei allen Seeds relativ ähnlich und weisen nur geringen Abweichungen auf. Häufig existieren sogar identische Regeln. Bei den Extraktionen, die bei starkem Pruning erheblich abbauen, fällt auf, dass diese in ihrem Hidden Layer keine geeigneten Konzepte zur Klassifikation besitzen. Nahezu sämtliche Regelmengen treffen nur noch die Entscheidung „verheiratet=ja/nein“. Damit kann die Ausgabeschicht nur schlecht eine gute Vorhersage treffen. Offenbar war das Pruning in diesen Fällen deutlich zu stark. Der große Abstand von Minimal und Maximalleistung dieser Konfiguration deutet darauf hin, dass die Pruningstärke an einem Punkt ist, an dem bereits leichte Veränderungen zum Kippen der Ergebnisse führen. Auch wenn keine konkreten Ursachen gefunden wurden, so scheint der Gewichtstransfer diese jedoch abzumildern. Dies könnte in der größeren Diversität der Konzepte begründet liegen. Damit die Ergebnisse der nachfolgenden semantischen Analyse vergleichbar sind, wurde dort eine Extraktion genutzt die gute Ergebnisse liefert.

Bei den extrahierten Regeln zeigt sich allgemein, dass stärkeres Pruning die Leistung verschlechtert. Dies ist insofern nicht überraschend, da bereits der kleinste Wert für das Hidden Layer relativ groß ist. Die zu Beginn gewählten Werte 400 und 1000 sind ein Kompromiss zwischen Leistung und Interpre-

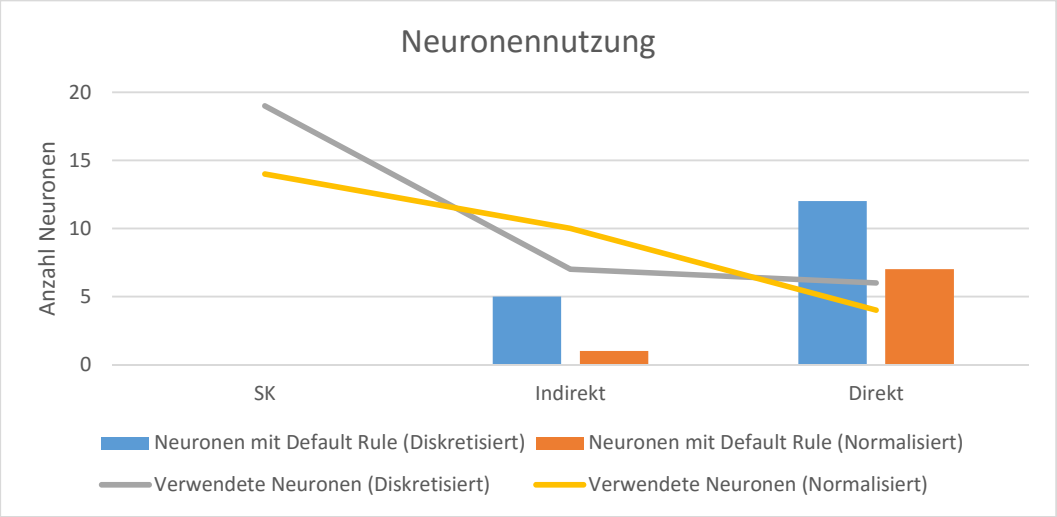

Abbildung 5.3: Neuronennutzung (Pruning: 400/100)

tierbarkeit. Beide Werte sind größer als für eine optimale Leistung sinnvoll ist. Dennoch zeigen sich Ergebnisse mit nur geringe Unterschiede zu denen der separaten Klassifikation. Durch größeren Fokus auf Parameteroptimierung wäre es vermutlich möglich die Unterschiede noch weiter zu reduzieren.

Ob in der Vorhersageschicht eine geringere Pruningstärke verwendet wird, hat nur minimalen Einfluss auf die Leistung. Mehrfach zeigten sich Messungen in denen die Leistung sogar völlig identisch war. Dies deutet darauf hin, dass die Regelmenge in der Ausgabeschicht bereits simpel genug ist, so dass sie keinen wesentlichen Änderungen unterworfen wird. Dies ist insofern etwas überraschend, da bei der separaten Klassifikation noch Unterschiede von bis zu 2% beobachtet werden konnten. Dies deutet daraufhin, dass die Konzepte in der Vorhersageschicht einfacher sind und die entscheidenden Aspekte bereits im Hidden Layer modelliert werden.

5.5.2 Neuronennutzung

Bei der Betrachtung der Regelmengen des direkten und indirekten Netzes fällt auf, dass sich mit stärkerem Fokus auf Klassifikation auch die semantische Struktur des Netzes ändert. Die Konzepte in der Ausgabeschicht werden deutlich einfacher. In Abbildung 5.3 ist mit den Linien dargestellt, wie viele Neuronen aus dem Hidden Layer für die Vorhersage genutzt werden. Dabei ist für beide Datensätze der oben angesprochene Trend gut zu erkennen. Beim direkten Netz für den normalisierten Datensatz sind lediglich vier Neuronen relevant für die Klassenerkennung. Bei der separaten Klassifikation gelingt die Vorhersage nicht ohne Weiteres und muss nahezu alle Neuronen für die Entscheidung heranziehen. Wird berücksichtigt, dass dessen Hidden Layer nicht primär zur Klassifikation trainiert wurde, ist dies nicht überraschend.

Erstaunlicherweise bleiben einige Neuronen völlig ungenutzt. Es existieren Neuronen, bei denen die Regelextraktion nur eine einzige Regel bestimmt: Die Default-Regel. Diese Neuronen sind nicht in der Lage irgendeine Logik zu realisieren und für die Vorhersage beizutragen. Beim direkten Netz ist die Zahl derer besonders hoch. Beim diskretisierten Datensatz ist bereits über die Hälfte aller Neuronen ohne Funktion. Von den verbliebenen acht sind nur vier relevant für die Vorhersage. Im Gegensatz dazu gibt es überhaupt keine funktionslosen Neuronen im Hidden Layer des Autoencoders. Damit ist eine klare Reduktion der Komplexität in Bezug auf Modell und Struktur erkennbar.

Die hybride Variante des indirekten Netzes zeigt ebenfalls funktionslose Neuronen. Da diese zuvor noch ein Konzept repräsentierten und eine Funktion erfüllten, muss diese verloren gegangen sein. Die zugehörigen Neuronen sind durch das zweite Training „abgestorben“. Im Vergleich zum direkten Netz

ist der Anteil jedoch erheblich geringer. Auch beim indirekten Netz werden viele Neuronen nicht für die Vorhersage verwendet. Im Gegensatz zum direkten Netz sterben sie allerdings nicht ab. Dadurch existieren viele ungenutzte Konzepte im Hidden Layer. Diese sind teilweise sehr ähnlich zu denen, die tatsächlich für die Vorhersage verwendeten werden. Es bilden sich offenbar redundante Konzepte. Bei der Stabilisierung des Netzes findet dann eine Entscheidung für eine Teilmenge dieser Neuronen statt.

5.5.3 Auswirkungen des Gewichtstransfers

Neuronen mit Konzeptübernahme

Durch den Gewichtstransfer sind teilweise erhebliche Änderungen erkennbar. Neben dem zuvor angesprochenen Verlust von Neuronen, zeigt sich auch, dass die Neuronen teilweise starken semantischen Änderungen unterworfen sind. Das erneute Training führt wiederholt zu tiefgreifenden Änderungen, wodurch bei vielen Neuronen das alte Konzept überhaupt nicht mehr zu erkennen ist. Gleichzeitig gibt es auch Neuronen, deren Semantik sich nur gering ändert. Für die Auswertung werden die Unterschiede der Neuronen zwischen dem Hidden Layer des Autoencoders und des indirekten Netzes in drei Kategorien eingeteilt:

- **Ähnliche** Neuronen weisen größere Gemeinsamkeiten auf. Die abgedeckten Personengruppen sind vergleichbar, können aber leichte Unterschiede aufweisen. Bei ähnlichen Neuronen existieren häufig Regeln, die identisch oder nur mit kleinen Änderungen in beiden Regelmengen zu finden sind. In beiden Regelmengen werden Bedingungen in semantisch gleicher Weise verwendet.
- Neuronen weisen eine **Tendenz** auf, wenn sie nicht ähnlich sind, aber dennoch erkennbare Gemeinsamkeiten zeigen. Hierbei ist es häufig der Fall, dass Bedingungen nicht mehr in der gleichen Abfolge auftreten, aber Teilaspekte gehäuft in den Regeln zu finden sind. In einigen Mengen des Autoencoders ist es der Fall, dass eine einzelne Bedingung extrem wichtig ist und als KO-Kriterium auftritt. Wenn eine solche Bedingung in der neuen Regelmenge nicht mehr als wichtigstes Kriterium zu erkennen ist, aber dennoch einen nennenswerten Einfluss hat, gilt dies auch als Gemeinsamkeit.

  Unter diese Kategorie fallen häufig Neuronen, die über komplexe Regelmengen verfügen. Aufgrund der erhöhten Anzahl von Bedingungen ist ein Vergleich auf semantische Ähnlichkeit in solchen Fällen nur schwierig bis gar nicht möglich. Gleichzeitig ist aber klar zu erkennen, dass das ursprüngliche Konzept einen Einfluss auf das neue Konzept hatte.
- Die dritte Kategorie fasst alle übrigen Neuronen zusammen. Diese verfügen über ein völlig anderes Konzept und es sind keinerlei Gemeinsamkeiten oder Zusammenhänge mit der vorherigen Regelmenge zu identifizieren.

Bei den Neuronen, die als ähnlich eingestuft wurden, gibt es deutliche Unterschiede im Grad der Ähnlichkeit. Bei einigen ist bereits auf den ersten Blick erkennbar, dass sie weitestgehend identisch sind. Bei anderen Neuronen sind Unterschiede erkennbar und es ist notwendig zu beurteilen wie tiefgreifend sie sind. Um dies zu verdeutlichen sollen einige Beispiele diskutiert werden.

```
1 (relationship=Husband <= 0) and (relationship=Not-in-family <= 0) => f_1_19=0.0 (9099.0/622.0)
2 (relationship=Husband <= 0) and ...
3 (relationship=Husband <= 0) and (age='(31.6-38.9]' >= 1) and ...
4 (relationship=Husband <= 0) and ...
5 (relationship=Husband <= 0) and (education-num='(8.5-10]' <= 0) and (education=Bachelors <= 0) and ...
6 ...
7  => f_1_19=1.0 (14273.0/1240.0)
```

Listing 5.11: Auszug aus der Regelmenge von Neuron d_1_19 des Autoencoders

```
1 (relationship=Husband >= 1) and (education-num='(8.5-10]' >= 1) and (age='(31.6-38.9]' <= 0) and (↙
      occupation=Sales <= 0) and ... => f_1_19=1.0 (3733.0/27.0)
2 (relationship=Husband >= 1) and (age='(31.6-38.9]' <= 0) and (education=Bachelors >= 1) and (↙
      occupation=Sales <= 0) => f_1_19=1.0 (1317.0/57.0)
3 (relationship=Husband >= 1) and (education-num='(8.5-10]' >= 1) and ... => f_1_19=1.0 (446.0/14.0)
4 (relationship=Husband >= 1) and (age='(31.6-38.9]' <= 0) and ... => f_1_19=1.0 (1965.0/871.0)
5  => f_1_19=0.0 (20339.0/1077.0)
```

Listing 5.12: Auszug aus der Regelmenge von Neuron d_1_19 des indirekten Netzes

Diese beiden Listings zeigen Auszüge der Regelmengen vor und nach dem Gewichtstransfer mit zweitem Training. Beide Mengen wurden dabei so gekürzt, dass die Gemeinsamkeiten gut erkennbar sind.

Es ist wichtig zu beachten, dass sich bei diesen Regelmengen die Default-Regel verändert hat. Während in Listing 5.11 diese noch 1 vorhersagte, so ändert sich dies in Listing 5.12 zu 0. Entsprechend ändern sich auch die Vorhersagen aller anderen Regeln. Der Lernalgorithmus hat entschieden, dass die Modellierung der Klasse 0 nun einfacher ist, als weiterhin Klasse 1 mit Regeln auszudrücken. Im Folgenden wird diese Situation als *invertierte Regelmenge* bezeichnet.

In diesen Regelmengen sind viele Gemeinsamkeiten erkennbar. Es ist sehr gut erkennbar, dass Ehemänner das mit Abstand wichtigste Kriterium sind. Bei der ersten Regelmenge führt der Status des Ehemanns unmittelbar dazu, dass das Neuron eine 1 ausgibt. Die anderen Bedingungen, wie die bezüglich des Alters oder die Bildung, fügen zusätzlichen Personengruppen hinzu.

Bei der Regelmenge des indirekten Netzes führt der Status eines Ehemanns nicht unmittelbar zu einer 1. Allerdings sind die Kriterien, die darüber hinaus notwendig sind, ähnlich zu den bereits zuvor beobachteten Bedingungen. Dadurch, dass die Regeln nun eine 1 statt 0 vorhersagen, hat sich der Aufbau der Regeln stark geändert. Es fällt aber auf, dass bei vielen Kriterien nun das Gegenteil als Bedingung vorhanden ist. Dies zeugt von einer starken semantischen Ähnlichkeit. Es ist klar erkennbar, dass diese nicht zufällig entstanden sein kann, sondern beim Training das alte Konzept in großen Teilen übernommen wurde. Es fällt jedoch auch auf, dass in den ersten beiden Regeln aus Listing 5.12 wiederholt der Verkäufer Beruf als Aspekt herangezogen wird. Dieser war zuvor, auch bei schwächerem Pruning, kein Teil der Regelmenge. Auch bei der früheren Regelmenge waren Bedingungen vorhanden, die nach dem erneuten Training verschwunden sind. Damit wird klar, dass dieses Neuron das Konzept nicht unmittelbar übernommen hat, sondern durch neue Aspekte ergänzt und erweitert hat.

Bei nahezu allen Neuronen, die als ähnlich bewertet wurden, ist dieses Verhalten zu beobachten. In den meisten Fällen beschränken sich die Veränderungen auf einzelne Bedingungen. Beispielsweise konnte mehrfach beobachtet werden, dass eine Prüfung auf das männliche Geschlecht hinzugefügt oder entfernt wurde.

Neben veränderten Neuronen konnte aber auch ein anderes Extrem beobachtet werden. In den Listings 5.13 und 5.14 sind wieder die Regel vor und nach dem Gewichtstransfer dargestellt.

```
1 (occupation=Craft-repair >= 1) => f_1_3=1.0 (3676.0/174.0)
2 (occupation=Prof-specialty >= 1) => f_1_3=1.0 (3713.0/452.0)
3 (relationship=Own-child >= 1) and (occupation=Adm-clerical <= 0) => f_1_3=1.0 (2767.0/782.0)
4  => f_1_3=0.0 (17644.0/2941.0)
```

Listing 5.13: Regelmenge von Neuron n_1_3 des Autoencoders

```
1 (relationship=Own-child >= 1) => f_1_3=1.0 (4101.0/248.0)
2 (education-num >= 0.8) => f_1_3=1.0 (6539.0/1772.0)
3 (occupation=Craft-repair >= 1) => f_1_3=1.0 (3125.0/549.0)
4  => f_1_3=0.0 (14035.0/1536.0)
```

Listing 5.14: Regelmenge von Neuron n_1_3 des indirekten Netzes

Hierbei weisen die beiden Neuronen eine erhebliche Ähnlichkeit auf. Die extrahierten Regel sind zwar in einer anderen Reihenfolge, ansonsten aber nahezu identisch. Die Abweichung bezüglich

education-num und occupation=Prof-specialty ist vernachlässigbar. Zum einen entspricht das Bildungsniveau von 0,8 einem Bachelor Abschluss, der die Grundlage für Akademikerberufe bildet. Damit werden bereits sehr viele Akademiker abgedeckt. Darüber hinaus verändert sich bei geringeren Pruningstufen im indirekten Netz die zweite Regel zu (education−num >= 0.8)**and** (occupation=Prof−specialty >= ↙ 1)=> f_1_3=1.0 (2620.0/89.0). Für die Vereinfachung ist nicht das Neuron, sondern das Pruning der Regelextraktion verantwortlich.

Aufgrund der verbleibenden Ungenauigkeit kann zwar keine definitive Identität garantiert werden, doch nach Betrachtung der Unterschiede erscheinen diese weitestgehend irrelevant. Das indirekte Netz hat das Konzept unverändert aus dem Autoencoder übernommen. Darüber hinaus trägt dieses Neuron nennenswert dazu bei, die Klassenvorhersage zu bestimmen. Bei Neuron n_1_4 kann ebenfalls eine sehr starke Ähnlichkeit beobachtet werde, diese ist aber nicht auf einem ähnlichen Niveau wie n_1_3. Es tritt lediglich bei diesen zwei Neuronen eine derart starke Ähnlichkeit auf.

Es ist auffällig das beide Neuronen im Rahmen des normalisierten Datensatzes aufgetreten sind und bei der diskretisierten Variante nichts Vergleichbares zu beobachten ist. Bei der diskreten Variante wurde der Großteil aller ähnlichen Regelmengen durch das zweite Training invertiert, wohingegen bei der normalisierten Version dies nie auftrat. Die Regelmengen, die invertiert wurden, zeichnen sich dadurch aus, das in allen Fällen ein dominantes Attribut existiert, das allen Regeln vorangestellt ist und erhalten bleibt.

Durch diesen Regelmengenaufbau wird die Semantik des Neurons durch ein einziges primäres Attribut beschrieben, welches durch Ausnahmen und Erweiterungen ergänzt wird. Wenn sich nun durch das Training die Neuronen verändern, wird es einfacher die Regeln für die gegenteilige Klasse zu bilden. Damit scheint der diskretisierte Datensatz zu instabileren Regeln mit mehr Fluktuation zu neigen.

In Tabelle 5.6 ist eine Heatmap dargestellt. Diese visualisiert zum einen die Relevanz jedes Neurons für die Klassifikation und zum anderen markiert sie alle (tendenziell) ähnlichen Neuronen. Je grüner ein Feld ist, desto wichtiger ist das zugehörige Neuron. Felder die weiß sind haben keinerlei Relevanz und erscheinen nicht in den Regelmengen der Ausgabeschicht. Um die Relevanz eines Neurons zu bewerten, wird die Regelmenge der nachfolgenden Schicht analysiert. Jedes Auftreten des Neurons wird mit der Position innerhalb der Regel gewichtet und aufsummiert. Die Gewichtung orientiert sich dabei am Lernverhalten des Regellerners und bewertet Bedingungen die sehr früh in der Regel auftreten wichtiger als spätere. Die aufsummierten Häufigkeiten werden anschließend normalisiert, so dass das wichtigste Neuron mit 1 bewertet wird. Diese Heatmap wurde für die Extraktion mit den Pruningstärken 400/100 erstellt. Wird die Regelmenge für 1000/100 herangezogen so bleibt sie im Kern ähnlich, wobei jedoch einzelne Neuronen relevant werden und andere irrelevant.

In der Häufigkeit von (tendenziell) ähnlichen Neuronen heben sich beide Datensätze kaum voneinander ab. Dass es im normalisierten Datensatz nur ein ähnliches Neuron weniger gibt, ist zu vernachlässigen, da die semantische Ähnlichkeit ein subjektiver Aspekt ist. Zudem zeigen sich beim normalisierten Datensatz die Neuronen mit der stärksten Ähnlichkeit. Bedauerlicherweise ist auch nach intensiver Suche keinerlei Zusammenhang erkennbar, der erklärt welche Neuronen beim indirekten Netz vom Autoencoder übernommen werden. In Bezug auf die Klassifikationsrelevanz ist keine Korrelation erkennbar. Weder in Bezug auf das indirekte Netz, noch bei der vorherigen separaten Klassifikation. Weiterhin zeigen sich, abgesehen von Heiratsstatus, keine semantischen Gemeinsamkeiten zwischen den Neuronen. Teilweise unterscheiden sich die Konzepte hierbei sogar recht stark.

Spezialisierung von Neuronen

Unabhängig vom Datensatz bilden nahezu sämtliche Neuronen durch das zweite Training eine Bedingung bezüglich des Heiratsstatus aus. Besonders die bereits sehr gut bekannte Bedingung „marital-status=Married-civ-spouse“ ist häufig vorhanden. Selbst Neuronen, die zuvor überhaupt keinen Bezug hierzu hatten, enthalten in der neuen Regelmenge entsprechende Referenzen. Anders als zuvor gibt es kein Neuron mehr, das sich ausschließlich dem Heiratsstatus widmet. Durch das zweite

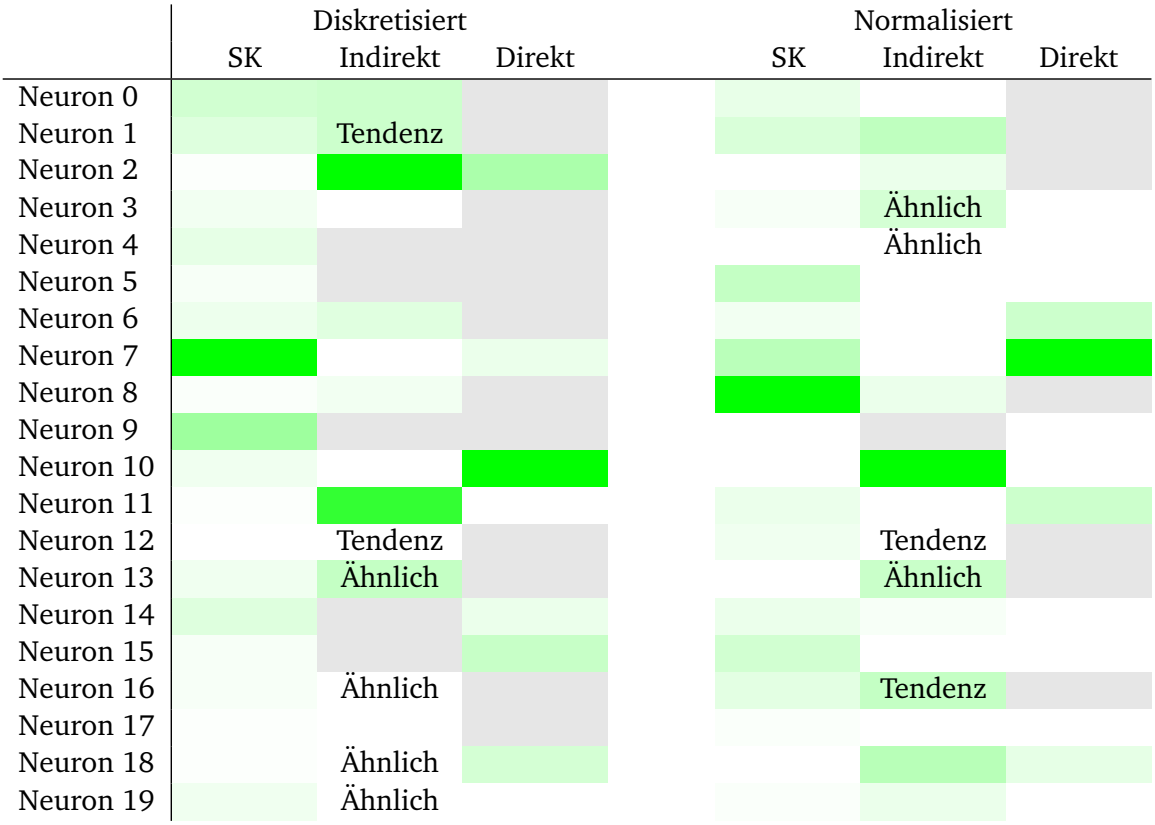

Tabelle 5.6: Heatmap der Relevanz für die Vorhersage (Pruning: 400/100)

Training ist dies überflüssig geworden, da dieses Konzept in alle anderen integriert wurde. Dadurch vereinfacht sich auch die Ausgabeschicht, da diese weniger Neuronen nutzen muss.

Generell ist eine Spezialisierung der Neuronen auch in Bezug auf andere Kriterien beobachtbar. Die Aspekte, die schon zuvor als wichtig für die Klassifizierung erkannt wurden, treten nun häufiger in den Neuronen auf. Elemente die keine Rolle spielen, wie beispielsweise die Zugehörigkeit zur Regierung, sind stark reduziert. Gleichwohl bleiben sie vorhanden und werden von den Neuronen auch weiterhin gelegentlich verwendet. Einige Neuronen spezialisieren sich dabei stärker auf die Klassifikationsaufgabe als andere und modellieren bereits Zusammenhänge, die sehr dicht an den bereits bekannten Muster sind. Entsprechend sind diese Neuronen oft besonders relevant in der Ausgabeschicht.

Aus der Heatmap ist gut zu erkennen, dass die Klassifikation meistens besonders stark von einem einzigen Neuron abhängig ist. Nur beim indirekten Netz des diskretisierten Datensatzes sind zwei Neuronen besonders relevant und nahezu gleich wichtig. In den beiden Mengen der separaten Klassifikation waren dies jeweils Neuronen über den Heiratsstatus. Bei den indirekten Netzen existiert ein solches Neuron nicht mehr und ist auch nicht notwendig, da nahezu jedes diese Information bereits enthält. Stattdessen zeigt sich auch hier die erhöhte Spezialisierung von einzelnen Neuronen.

```
1 (marital-status=Married-civ-spouse >= 1) and (education-num='(11.5-13]' >= 1) and (education=Bachelors ↙
      >= 1) => f_1_2=0.0
2 (marital-status=Married-civ-spouse >= 1) and ... and (occupation=Prof-specialty >= 1) => f_1_2=0.0
3 (marital-status=Married-civ-spouse >= 1) and (education=HS-grad <= 0) and (occupation=Exec-managerial ↙
      >= 1) => f_1_2=0.0
4 ...
5  => f_1_2=1.0
```

Listing 5.15: Gekürzte Regelmenge von Neuron d_1_2 des indirekten Netzes

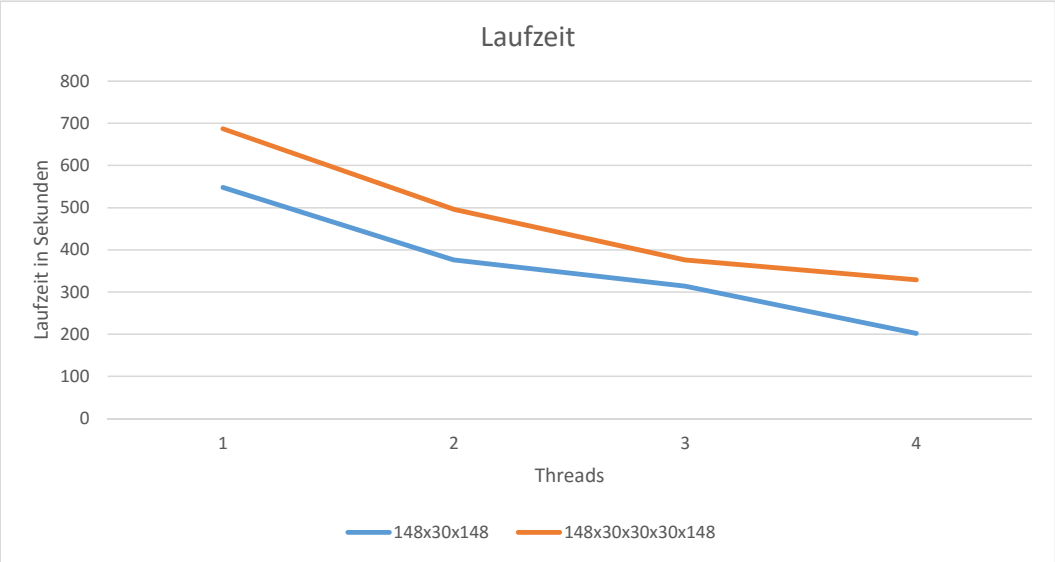

Abbildung 5.4: Laufzeitperformance im Mehrkernbetrieb

Listing 5.15 zeigt ein solches besonders relevantes Neuron des indirekten Netzes. Dabei ist gut zu erkennen, dass es bereits sehr stark auf die Klassifikationsaufgabe ausgerichtet ist. Alleine dieses Neuron modelliert die meisten Kriterien, die für die Klassifikation notwendig sind. Selbst bei einer ausschließlichen Verwendung dieses Neurons könnte vermutlich ein gute Ergebnis erreicht werden. Umso überraschender ist es, dass ein weiteres, fast gleich relevantes, Neuron verwendet wird. Dieses ist deutlich weniger stark auf die Klassifikation ausgerichtet und nutzt sehr viele Bedingungen die Attribute auf 0 prüfen. Dadurch können relativ feine Ausschlüsse modelliert werden, um das Ergebnis zu verbessern. So werden beispielsweise junge Personen unter 25 Jahren generell ausgeschlossen.

Beim normalisierten Datensatz kann keine derartig starke Spezialisierung im Neuron 10 festgestellt werden. Es wird der Heiratsstatus und nach einem angefangenen College gefiltert. Die spezifischeren Aspekte werden erst in nachfolgenden Neuronen verwendet. Dabei verteilt sich das Wissen jedoch über deutlich mehr Neuronen. Die Spezialisierung der Neuronen ist bei diesem Datensatz insgesamt deutlich geringer. Dennoch sind mehrere Neuronen identifizierbar, die genau wie bei der diskreten Variante, bereits große Ähnlichkeit mit den übergeordneten Klassifikationskonzepten besitzen.

Die durchschnittliche Regelzahl im Hidden Layer beträgt für den normalisierten Datensatz etwa 4,6. Bei der diskreten Version liegt sie lediglich bei 3,7. Auch in der Ausgabeschicht existieren mit 6 und 4,5 deutliche Unterschiede. Bereits bei separater Klassifikation sowie der Betrachtung der Konzepte im Autoencoder wurden für den normalisierten Datensatz komplexere Regeln festgestellt. Damit bestätigt sich dieses Verhalten erneut auch für das indirekte Netz.

5.6 Laufzeit

Abschließend wird auf die Laufzeitperformance der Regelextraktion eingegangen. Da der Fokus dieser Arbeit nicht auf der Entwicklung eines effizienten Extraktionsalgorithmus lag, wird die Performance nur kurz angerissen. Die Laufzeit ist von einer großen Anzahl von Faktoren abhängig. Darunter fallen beispielsweise die Pruningstärke, die Anzahl der Neuronen sowie deren Verknüpfungen. Die Trainingszeit eines einzelnen Neurons hängt stark von der Anzahl der Neuronen im vorherigen Layer ab. Bereits dadurch entsteht eine große Anzahl von möglichen Messpunkten, deren Eruierung nicht Ziel dieser Arbeit ist. Daher werden nur zwei verschiedene Konfigurationen miteinander verglichen. Zum einen die bereits bekannte 148x30x148, sowie eine neue mit 148x30x30x30x148. Die zweite soll dabei das

Verhalten bei komplexeren Netzen mit mehr als einem Hidden Layer zeigen. Beide Konfigurationen sollen als Beispiele dienen, um eine ungefähre Abschätzung der Leistung zu ermöglichen.

Alle Analysen und Untersuchungen wurden unter Windows 10 mit dem Java Runtime Environment 1.8.0_20 durchgeführt. Die maximale Heapgröße der Java Virtual Machine wurde auf sechs Gigabyte festgelegt. Als Prozessor wurde ein ein Intel i5 2500K eingesetzt. Dieser ist ein Endverbraucherprodukt und zum Entstehungszeitpunkt dieser Arbeit bereits leicht veraltet.

In Abbildung 5.4 ist die Laufzeit bei Pruningstärke 1000 und einer unterschiedlichen Anzahl von Threads abgebildet. Es ist zu erkennen, dass sich die Laufzeit sich beim Einsatz mehrere Prozessorkerne deutlich verringert. Allerdings ist die Verbesserung nicht linear und geringer als zu erwarten wäre. Bei steigender Threadzahl können zwar mehrere Klassifizierer gleichzeitig trainiert werden, deren Trainingszeit steigt dabei aber an. Es konnte ermittelt werden, dass die verwendete Implementierung des Ripper-Algorithmus aus dem Weka Framework dafür verantwortlich ist. Es ist zu vermuten, dass in der Implementierung Codeabschnitte existieren, die bei einem Mehrkernbetrieb nicht gut skalieren.

Unabhängig von der Skalierbarkeit zeigt das Verfahren eine gute allgemeine Performance. Selbst bei dem deutlich komplexeren Netz konnte eine Regelextraktion in etwa 5 Minuten abgeschlossen werden. Es ist wenig überraschen, dass wenn die Pruningstärke reduziert wird, die Laufzeit steigt. Bei Stärke 400 beträgt die Laufzeit etwas weniger als das doppelte. Auch damit ist die Laufzeit in einem Bereich, der relativ kurz ist. Die Laufzeiten des Verfahrens sind generell recht kurz und erlauben es damit auch komplexere neuronale Netze in überschaubarer Zeit zu analysieren.

6 Diskussion und Ausblick

In dieser Arbeit wurde die Interpretation des Hidden Layers eines Autoencoders für einen Beispieldatensatz vorgestellt. Dafür wurden zum einen die Konzepte im Hidden Layer für jedes Neuron analysiert und ausgewertet. Darüber hinaus wurden zwei Ansätze untersucht, bei denen die erlernten Konzepte für für neue Aufgaben angewendet wurden. Eine dieser Techniken ist die *separate Klassifikation*, die ein Klassifikationsproblem auf den komprimierten Daten des Encoders umsetzt. Die zweite Technik betrachtet das Verhalten des Netzes, wenn existierende Konzepte als Initalisierungswerte genutzt werden. Neben einer quantitativen Auswertung lag der Fokus bei allen Aspekten dabei besonders auf den qualitativen Ergebnissen.

Im Rahmen der Evaluation konnte ein einzelnes Attribut als besonders wichtig identifiziert werden. Dieses erhält als einziges ein eigenes Neuron, das ausschließlich auf die Weiterleitung des Attributes ausgerichtet ist. Alle anderen Neuronen modellieren Konzepte, die verschiedene Aspekte miteinander vermischen. Die meisten Neuronen kombinieren so Personengruppen, bei denen keine übergeordnete Gemeinsamkeit erkennbar ist. In den Fällen in denen eine Gemeinsamkeit aller Regeln gefunden wurde, beschränkt sich diese auf einfache Dinge wie beispielsweise das Geschlecht. Es konnte generell festgestellt werden, dass der normalisiere Datensatz erkennbar schwieriger zu interpretieren ist und komplexere Konzepte besitzt.

Für die separate Klassifikation konnte festgestellt werden, das diese eine gute Klassifikationsleistung erreicht. Sie ist dabei vergleichbar mit einer Regelmenge, die direkt aus den Eingabedaten gelernt wurde. Gleichzeitig weist der Eingabevektor für die separate Klassifikation nur einen Bruchteil der Größe auf. Selbst eine Konfiguration die eine Kompression von über 90% hatte, erreichte gute Leistung. Diese war nur 2% schlechter als das beste beobachtete Ergebnis. Obwohl der Autoencoder nicht dafür trainiert wurde, stellte er für die Klassifikation die gleiche Konzepte zur Verfügung, wie jene, die bei der direkten Klassifikation entwickelt wurden. Der Autoencoder fand zwar keine völlig neuen Aspekte für die Klassifikation, doch er konnte eine erhöhte Relevanz des Geschlechts ermitteln. Dies zeigt deutlich, dass die gelernten Konzepte eines Autoencoders nicht nur zur Rekonstruktion von Daten, sondern auch zur Klassifikation geeignet sind.

Beim Gewichtstransfer konnte eine Spezialisierung des Netzes beobachtet werden. Die meisten zuvor erlernten Konzepte gehen dabei verloren und sind im neuen Netz nicht mehr erkennbar. Gleichzeitig existieren jedoch auch Neuronen, bei denen eine deutliche Ähnlichkeit feststellbar ist. In einigen wenigen Fällen wurde festgestellt, dass Konzepte nahezu identisch aus dem Autoencoder übernommen wurden. Bedauerlicherweise konnte kein Muster erkannt werden, nachdem die Konzeptübernahme stattfindet. Es konnte ebenfalls keine Verbesserung der Leistung im Vergleich mit einem regulären neuronalen Netz zur Klassifikation festgestellt werden.

Für die Zukunft wäre es sinnvoll, weitere Untersuchungen in Richtung der separaten Klassifikation vorzunehmen. Diese konnte von allen vorgestellten Ansätzen die vielversprechendsten Resultate aufweisen. Da sich die Arbeit primär auf die qualitativen Aspekte konzentriert hat, ist hier noch Potential für weitere Optimierungen. Diese Arbeit hat zwar breit gestreut viele Aspekte betrachtet, dafür aber nur einen einzelnen Datensatz untersucht. Entsprechend ist es unverzichtbar die interessanten Aspekte bei anderen Datensätzen zu bestätigen.

Literatur

[ADT95] Robert Andrews, Joachim Diederich und Alan B Tickle. „Survey and critique of techniques for extracting rules from trained artificial neural networks“. In: *Knowledge-based systems* 8.6 (1995), S. 373–389.

[CK01] Amanda Clare und Ross D. King. „Knowledge Discovery in Multi-label Phenotype Data“. In: *Principles of Data Mining and Knowledge Discovery: 5th European Conference, PKDD 2001, Freiburg, Germany, September 3–5, 2001 Proceedings*. Hrsg. von Luc De Raedt und Arno Siebes. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 42–53. ISBN: 978-3-540-44794-8. DOI: 10.1007/3-540-44794-6_4. URL: http://dx.doi.org/10.1007/3-540-44794-6_4.

[Coh95] William W. Cohen. „Fast Effective Rule Induction“. In: *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, S. 115–123.

[CS94] Mark W. Craven und Jude W. Shavlik. „Using Sampling and Queries to Extract Rules from Trained Neural Networks“. In: *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, S. 37–45.

[Fre14] Alex A. Freitas. „Comprehensible Classification Models: A Position Paper“. In: *SIGKDD Explor. Newsl.* 15.1 (2014), S. 1–10. ISSN: 1931-0145. DOI: 10.1145/2594473.2594475. URL: http://doi.acm.org/10.1145/2594473.2594475.

[Fu91] LiMin Fu. „Rule Learning by Searching on Adapted Nets.“ In: *AAAI*. Bd. 91. 1991, S. 590–595.

[Fu94] LiMin Fu. „Rule generation from neural networks“. In: *IEEE Transactions on Systems, Man, and Cybernetics* 24.8 (1994), S. 1114–1124. ISSN: 0018-9472. DOI: 10.1109/21.299696.

[FW94] Johannes Fürnkranz und Gerhard Widmer. „Incremental reduced error pruning“. In: *Proceedings of the 11th International Conference on Machine Learning (ML-94)*. 1994, S. 70–77.

[FWA10] Alex A. Freitas, Daniela C. Wieser und Rolf Apweiler. „On the Importance of Comprehensible Classification Models for Protein Function Prediction“. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7.1 (2010), S. 172–182. ISSN: 1545-5963. DOI: http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.47.

[GB10] Xavier Glorot und Yoshua Bengio. „Understanding the difficulty of training deep feedforward neural networks“. In: *International conference on artificial intelligence and statistics*. 2010, S. 249–256.

[Gen+15] J. Geng u. a. „High-Resolution SAR Image Classification via Deep Convolutional Autoencoders“. In: *IEEE Geoscience and Remote Sensing Letters* 12.11 (2015), S. 2351–2355. ISSN: 1545-598X. DOI: 10.1109/LGRS.2015.2478256.

[GS88] R Paul Gorman und Terrence J Sejnowski. „Analysis of hidden units in a layered network trained to classify sonar targets“. In: *Neural networks* 1.1 (1988), S. 75–89.

[Gu+15] Feng Gu u. a. „Marginalised Stacked Denoising Autoencoders for Robust Representation of Real-Time Multi-View Action Recognition“. In: *Sensors* 15.7 (2015), S. 17209. ISSN: 1424-8220. DOI: 10.3390/s150717209. URL: http://www.mdpi.com/1424-8220/15/7/17209.

[HBV06] Johan Huysmans, Bart Baesens und Jan Vanthienen. „Using rule extraction to improve the comprehensibility of predictive models“. In: *Available at SSRN 961358* (2006).

[He+06] Jieyue He u. a. „Transmembrane Segments Prediction and Understanding Using Support Vector Machine and Decision Tree“. In: *Expert Syst. Appl.* 30.1 (2006), S. 64–72. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2005.09.045. URL: http://dx.doi.org/10.1016/j.eswa.2005.09.045.

[Huy+11] Johan Huysmans u. a. „An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models“. In: *Decision Support Systems* 51.1 (2011), S. 141–154.

[JN09] U. Johansson und L. Niklasson. „Evolving decision trees using oracle guides“. In: *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*. 2009, S. 238–244. DOI: 10.1109/CIDM.2009.4938655.

[JNK04] Ulf Johansson, Lars Niklasson und Rikard König. „Accuracy vs. comprehensibility in data mining models“. In: *Proceedings of the Seventh International Conference on Information Fusion*. Bd. 1. Citeseer. 2004, S. 295–300.

[Joh+06] U. Johansson u. a. „Rule Extraction from Opaque Models– A Slightly Different Perspective“. In: *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*. 2006, S. 22–27. DOI: 10.1109/ICMLA.2006.46.

[KC+03] Ross Donald King, Amanda Clare u. a. „Predicting gene function in Saccharomyces cerevisiae“. In: (2003).

[Lav99] Nada Lavrač. „Selected techniques for data mining in medicine“. In: *Artificial intelligence in medicine* 16.1 (1999), S. 3–23.

[LFZ16] Yanan Liu, Xiaoqing Feng und Zhiguang Zhou. „Multimodal video classification with stacked contractive autoencoders“. In: *Signal Processing* 120 (2016), S. 761 –766. ISSN: 0165-1684. DOI: http://dx.doi.org/10.1016/j.sigpro.2015.01.001. URL: http://www.sciencedirect.com/science/article/pii/S016516841500002X.

[Li+12] Guangda Li u. a. „In-video Product Annotation with Web Information Mining“. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 8.4 (2012), 55:1–55:19. ISSN: 1551-6857. DOI: 10.1145/2379790.2379797. URL: http://doi.acm.org/10.1145/2379790.2379797.

[LSL95] Hongjun Lu, Rudy Setiono und Huan Liu. „NeuroRule: A Connectionist Approach to Data Mining“. In: *Proceedings of the 21th International Conference on Very Large Data Bases*. VLDB '95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, S. 478–489. ISBN: 1-55860-379-4. URL: http://dl.acm.org/citation.cfm?id=645921.758364.

[Mit92] John Mitchell. „A geometric interpretation of hidden layer units in feedforward neural networks“. In: *Network: Computation in Neural Systems* 3.1 (1992), S. 19–25.

[Mit97] Tom Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0071154673.

[MST94] Donald Michie, David J Spiegelhalter und Charles C Taylor. *Machine learning, neural and statistical classification*. Prentice Hall, 1994. ISBN: 013106360X.

[OF13] Fernando E.B. Otero und Alex A. Freitas. „Improving the Interpretability of Classification Rules Discovered by an Ant Colony Algorithm“. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. ACM, 2013, S. 73–80. ISBN: 978-1-4503-1963-8. DOI: 10.1145/2463372.2463382. URL: http://doi.acm.org/10.1145/2463372.2463382.

[SAG99] Gregor PJ Schmitz, Chris Aldrich und Francois S Gouws. „ANN-DT: an algorithm for extraction of decision trees from artificial neural networks“. In: *IEEE Transactions on Neural Networks* 10.6 (1999), S. 1392–1401.

[San89] Dennis Sanger. „Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks“. In: *Connection Science* 1.2 (1989), S. 115–138.

[SMM12] Kamal Kumar Sethi, Durgesh Kumar Mishra und Bharat Mishra. „Extended taxonomy of rule extraction techniques and assessment of kdruleex“. In: *International Journal of Computer Applications* 50.21 (2012).

[ST01] M. Sato und H. Tsukimoto. „Rule extraction from neural networks via decision tree induction“. In: *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*. Bd. 3. 2001, 1870–1875 vol.3. DOI: 10.1109/IJCNN.2001.938448.

[TG99] I. A. Taha und J. Ghosh. „Symbolic interpretation of artificial neural networks“. In: *IEEE Transactions on Knowledge and Data Engineering* 11.3 (1999), S. 448–463. ISSN: 1041-4347. DOI: 10.1109/69.774103.

[Thr93] Sebastian Thrun. *Extracting Provably Correct Rules from Artificial Neural Networks*. Techn. Ber. University of Bonn, 1993.

[Thr95] Sebastian Thrun. „Extracting Rules from Artificial Neural Networks with Distributed Representations“. In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 7*. MIT Press, 1995, S. 505–512.

[TS91] Geoffrey Towell und Jude W Shavlik. „Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules“. In: *NIPS*. Bd. 4. 1991, S. 37–43.

[Wan+09] M. Wang u. a. „Beyond Distance Measurement: Constructing Neighborhood Similarity for Video Annotation“. In: *IEEE Transactions on Multimedia* 11.3 (2009), S. 465–476. ISSN: 1520-9210. DOI: 10.1109/TMM.2009.2012919.

[Zil15] Jan Ruben Zilke. „Extracting Rules from Deep Neural Networks“. Master's Thesis. TU Darmstadt, Knowledge Engineering Group, 2015. URL: http://www.ke.tu-darmstadt.de/lehre/arbeiten/master/2015/Zilke_Jan.pdf.