Business Analytics

Seiter

2., komplett überarbeitete und erweiterte Auflage 2019
ISBN 978-3-8006-5871-8
Vahlen

Die Ergebnisse der Regressionsanalyse müssen **evaluiert** werden. Hierzu haben sich eine Vielzahl von Maßen und Tests etabliert. Drei besonders verbreitete sind (vgl. Kumar 2017, S. 240):

- das Bestimmtheitsmaß R^2,
- eine Prüfung der Regressionskoeffizienten und
- die Analyse der Ausreißer sowie
- Maße zur Prognosegüte.

Das **Bestimmtheitsmaß** R^2 wie viel der Varianz der abhängigen Variablen durch das gewonnen Prognosemodell erklärt wird. Es ist definiert als der Quotient aus der durch das Prognosemodell erklärten Varianz und der Gesamtvarianz der abhängigen Variablen. Folglich reicht der Wertebereich von 0 bis 1, wobei höhere Werte eine bessere Anpassung an die Daten anzeigen.

Der Regressionsalgorithmus kam zwar zum Ergebnis, dass ein Zusammenhang zwischen unabhängiger und abhängiger Variable besteht, aber es wurde keine Aussage darüber getroffen, ob der Zusammenhang auch statistisch signifikant ist. Dazu wird mittels eines t-Test die Nullhypothese getestet: „Es existiert kein Zusammenhang zwischen unabhängiger und abhängiger Variable". Hierzu wird der p-Wert (vgl. Backhaus et al. 2016, S. 90) ermittelt. Dieser wird mit einem vorab definierten **Signifikanzniveau** verglichen. Dabei werden häufig die Werte 0,01 oder 0,05 gewählt, um einen gewissen Grad an Glaubwürdigkeit zu etablieren. Ist der p-Wert geringer als das Signifikanzniveau kann die Nullhypothese verworfen werden und ein Zusammenhang angenommen werden. Jedoch ist kein Beweis für einen kausalen Einfluss der unabhängigen auf die abhängige Variable besteht. Für eine solche Schlussfolgerung ist Klärung des Mechanismus notwendig, der dem Zusammenhang zugrunde liegt (vgl. hierzu Kap. 5.3).

Ein weiterer Evaluationsschreit ist die **Ausreißeranalyse**. Starke Ausreißer können einen enormen Einfluss auf die Regressionskoeffizienten haben. Daher ist es notwendig, zu prüfen, wie viele Ausreißer vorliegen und ob diese in die Analyse mit einbezogen werden sollten.

Ebenfalls Gegenstand der Evaluation ist die Prognosegüte. Basis ist hierfür ein Teil der Datenmatrix, der nicht zur Ermittlung der Regressionsgleichung verwendet wurde. Dazu werden die Daten in Trainingsdaten und Testdaten aufgeteilt. Die Testdaten haben in der Regel einen Anteil von 10 % bis 30 % an den Gesamtdaten. Für diesen vorab von den Trainingsdaten getrennten Teil werden verschiedene Evaluationsmaße berechnet. Sie sollen anzeigen, wie gut die Prognose tatsächlich erfolgt. Ein Beispiel ist der Root Mean-Square Error mit p als prognostiziertem Wert und a als tatsächlichem Wert:

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$$

Ein zweites Beispiel ist der Root Relative-Squared Error mit $\bar{a}$ als Mittelwert der tatsächlichen Werte

$$RRSE = \sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \ldots + (a_n - \bar{a})^2}}$$

Beides sind weit verbreitete Maße (vgl. Witten et al. 2017, S. 195). Neben diesen Evaluationsmaßen existiert eine Vielzahl weiterer, die u. a. auf Korrelationen zwischen prognostizierten und tatsächlichen Werten beruhen. In der Praxis ist die Wahl des Evaluationsmaßes für den Vergleich verschiedener Regressionsmodelle allerdings nicht entscheidend, da die verschiedenen Evaluationsmaße regelmäßig das gleiche Modell als das Beste anzeigen (vgl. Witten et al. 2017, S. 196).

» Klassifikationsanalysen

Ein Unternehmen stellt fest, dass sich zu wenig seiner Bestandskunden zu A-Kunden entwickeln. Zu viele Kunden wechseln zu konkurrierenden Unternehmen, bevor sie diesen Status erreichen. Als **Lösungsidee** möchte das Unternehmen zielgerichtet Bestandskunden, die noch keine A-Kunden sind, zu solchen entwickeln. Allerdings ist eine Auswahl geeigneter Bestandskunden notwendig, da die zur Entwicklung notwendigen personellen Ressourcen begrenzt sind. Das **Analytics-Problem** besteht folglich in der Frage, auf Basis welcher Eigenschaften prognostiziert werden kann, ob Bestandskunden ein Potenzial aufweisen, zum A-Kunden entwickelt zu werden. Gelöst werden kann dieses Problem mithilfe von Klassifikationsanalysen.

Ziel der Klassifikationsanalyse ist die Unterteilung einer Datenmatrix in vorgegebene **Klassen** (vgl. Cleve/Lämmel 2016, S. 59). Kennzeichen der Klassen ist ein Attribut der Datenmatrix, das als **Klassenattribut** bezeichnet wird. Grundsätzlich kann jedes Attribut als Klassenattribut dienen. In der Regel sind es Attribute mit wenigen Ausprägungen. Ein Beispiel ist das Attribut „Ausfall einer Maschine“ mit den Ausprägungen „Ausfall“ und „kein Ausfall“.

Klassifikationsanalysen sind **überwachte Segmentierungen**, da die Segmente, also die Klassen, durch die Ausprägungen des Klassenattributs bestimmt sind (vgl. Provost/Fawcett 2013, S. 48). Hier zeigt sich ein wesentlicher Unterschied zu Clusteranalysen, bei denen die Segmentierung in Cluster gerade keinen Vorgaben, außer ggf. der Anzahl der Segmente, unterliegen.

Klassifikationsanalysen unterstützen die Gewinnung von Prognosemodellen auf Basis von Trainingsdaten. Die Modelle dienen der Prognose des Klassenattributs einer Instanz, die nicht Teil der Trainingsdaten ist (vgl. Aggarwal 2015, S. 18).

Mithilfe der Klassifikationsalgorithmen werden jene Attribute identifiziert, anhand derer die Segmentierung durchgeführt wird. In der Literatur werden diese auch als „informative attributes“ bezeichnet, da sie einen spezifischen informatorischen Mehrwert liefern – die spätere Prognosegrundlage (vgl. Provost/Fawcett 2013, S. 49 f.).

Die Varianten der Klassifikation sind vielfältig. In den folgenden Ausführungen liegt der Fokus auf

- Entscheidungsbäumen,
- probabilistischen Ansätzen sowie
- neuronalen Netzen.

Für einen Überblick über weitere Varianten, wie bspw. Support Vector Machines, sei hier auf vertiefende Literatur verwiesen (bspw. Aggarwal 2015; Cleve/Lämmel 2016).

Ein **Entscheidungsbaum** ist ein gerichteter Graph, der aus Knoten, Kanten und Blättern besteht (vgl. Provost/Fawcett 2013, S. 63). Im Falle der hier betrachteten univariaten Bäume wird an jedem Knoten ein Attribut zur Segmentierung der Datenmatrix in Teilmengen verwendet (vgl. Witten et al. 2017, S. 105 f.). Die Unterteilung erfolgt gemäß der Ausprägung des betreffenden **Attributs**. Im Fall von nominal-skalierten Attributen mit einer geringen Anzahl Ausprägungen, stellt jede Ausprägung eine eigene Kante dar. Im Falle von metrisch-skalierten Attributen führt eine separate Berücksichtigung aller Ausprägungen zu einem nicht mehr darstellbaren, da zu komplexen, Baum. Als Lösung für dieses Problem bietet sich die **Einführung von Intervallen** an (vgl. Cleve/Lämmel 2016, S. 107). Der einfachste Fall ist die Unterteilung in zwei Intervalle durch einen Schwellenwert. Durch die Einführung mehrere Schwellenwerte können weitere Intervalle gebildet werden. Wieder ist die zunehmende Komplexität der begrenzende Faktor.

Das Ziel der Unterteilung ist die Herstellung homogener Teilmengen hinsichtlich des zu prognostizierenden Klassenattributs (vgl. van der Aalst 2011, S. 67). Vollkommen homogen ist eine Teilmenge dann, wenn sie nur Instanzen enthält, deren Klassenattribut dieselbe Ausprägung aufweisen. Ist eine hinreichende **Homogenität** nach einem Knoten erreicht, dann endet der Ast bezeichnet, mit einem **Blatt**. Liegt keine hinreichende Homogenität vor, wird am Ende der entsprechenden Kante ein weiterer Knoten vorgesehen.

Zur Verdeutlichung dieses Konstruktionsprinzips zeigt Abbildung 67 einen mehrstufigen Entscheidungsbaum aus dem Kontext Predictive Maintenance. Das Klassenattribut ist der Ausfall der betrachteten Maschine in der nächsten Woche mit den

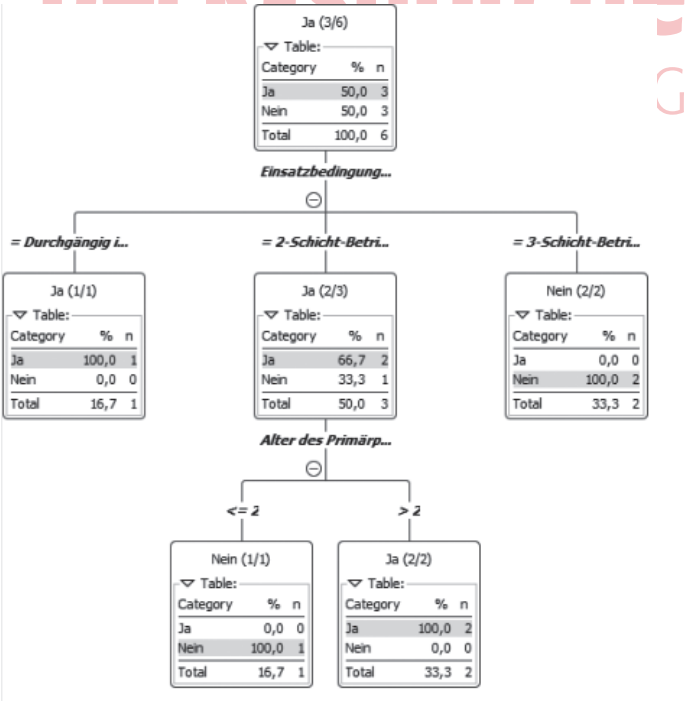

Abbildung 67: Mehrstufiger Entscheidungsbaum

Ausprägungen „Ja“ und „Nein“. Das erste unterteilende Attribut sind die Einsatzbedingungen, mit den drei Ausprägungen „durchgängiger Betrieb“, „2-Schicht-Betrieb“ und „3-Schicht-Betrieb“. Das zweite unterteilende Attribut ist das Alter der Maschine mit den zwei Intervallen „jünger oder genau 2 Jahre“ und „älter als 2 Jahre“.

Das Beispiel zeigt die elementare Frage bei der Konstruktion eines Entscheidungsbaums: Welches Attribut soll an welcher Stelle mit welchen Ausprägungen zur Segmentierung der Datenmatrix herangezogen werden (vgl. Witten et al. 2017, S. 106)? Diese Wahl determiniert wesentlich die Struktur und damit die Prognosequalität des Entscheidungsbaums.

Es wird ein Maß benötigt, das es erlaubt, die möglichen Segmentierungsattribute in eine eindeutige Rangfolge zu bringen. Das dominante Maß ist der **Informationsgewinn** (vgl. Provost/Fawcett 2013, S. 51). Er kann interpretiert werden als die Erhöhung der Homogenität durch die Anwendung eines bestimmten Segmentierungsattributs. Es ist folglich die Differenz aus der Homogenität vor der Segmentierung und der Homogenität nach Segmentierung auf Basis eines bestimmten Attributs. Als Maß für den Informationsgewinn wird im Folgenden die **Entropie** erörtert. Entropie H_n ist definiert als (vgl. Cleve/Lämmel 2016, S. 105):

$$H_n = -\sum_{i=1}^{n} p_i \cdot log_2(p_i)$$

Dabei ist P_i die Wahrscheinlichkeit von Ausprägung i des Klassenattributs im betrachteten Segment. Folglich ist der Wert 1, wenn alle Instanzen des Segments diese Ausprägung aufweisen und 0, wenn keine Instanz diesen Wert aufweist. Abbildung 68 zeigt den Verlauf der Entropie eines Segments S in Abhängigkeit der Häufigkeiten der beiden Ausprägungen des Klassenattributs: „+“ und „–“.

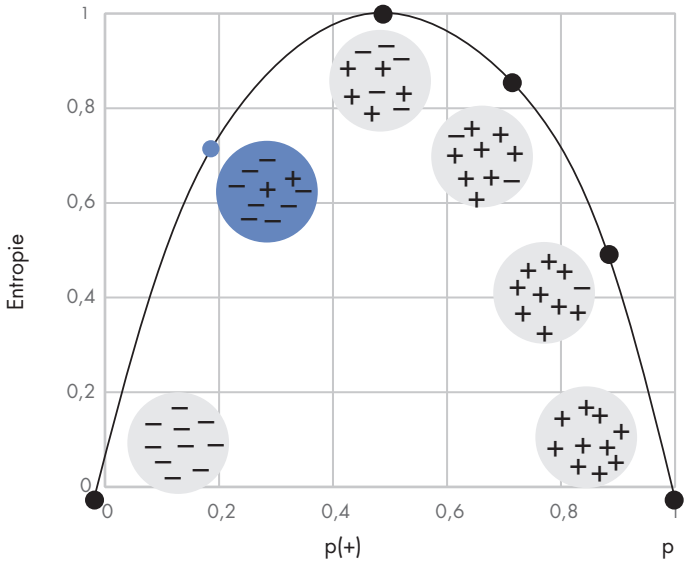

Abbildung 68: Entropie (in Anlehnung an Provost/Fawcett 2013, S. 52)

Im Beispiel umfasst das Segment *S* insgesamt 10 Instanzen. Die Wahrscheinlichkeiten der Ausprägungen der beiden Klassenattribute im Falle des blau gefärbten Segments sind:

$$p_1 = 2/10 = 0{,}2$$
$$p_2 = 8/10 = 0{,}8$$

Die Entropie *H (S)* ist dann

$$H(S) = -\left[0{,}2 * log_2(0{,}2) + 0{,}8 * log_2(0{,}8)\right] \approx 0{,}722$$

Für die Konstruktion eines Entscheidungsbaumes kann u. a. der weit verbreitete **C4.5-Algorithmus** verwendet werden (vgl. Wu et al. 2008, S. 3). Als erster Schritt wird aus der Attributmenge A das Attribut $a \in A$ als Wurzel des Entscheidungsbaums ausgewählt, das zum höchsten Informationsgewinn führt. ω_a ist dabei die Menge aller Werte, die das Attribut a annehmen kann. Für jede Ausprägung $\omega \in \omega_a$ wird das entsprechende Segment gebildet. Diese enthalten jeweils alle Instanzen, die die Ausprägung ω aufweisen. Die Segmentierung der entstandenen Segmente wird so fortgesetzt, bis ein Abbruchkriterium erreicht ist. Eine Möglichkeit ist das Erreichen eines Segments, das hinsichtlich des Klassenattributs vollkommen homogen ist.

Der C4.5-Algorithmus identifiziert das jeweils Entropie-optimierende Attribut zur Segmentierung. Er nimmt eine Unterteilung der numerischen Attribute in Intervalle vor und transformiert sie dadurch in ordinale Attribute. Besitzt ein Attribut a n Ausprägungen ($A_1, \ldots A_n$), so erfolgt die Bildung der Intervalle $[a|a \leq A_i]$ und $[a|a > A_i]$ für jedes $i = 1, \ldots, n - 1$. Diese zwei Intervalle fungieren dann als Ausprägungen des Attributs A. Es wird jene Intervallstruktur gewählt, welche dem größten Informationsgewinn entspricht und somit zu möglichst homogenen Segmenten führt (vgl. Cleve/Lämmel 2016, S. 107).

Die Beschreibung des Algorithmus zeigt die Anfälligkeit von Klassifikationsanalysen im Falle von Missing Values. Da es sich um eine überwachte Segmentierung handelt, ist eine zentrale Anforderung an die Trainingsdaten, dass in jeder Instanz das Klassenattribut vorhanden sein muss. Fehlende Werte stellen folglich ein Problem im Lernprozess dar.

Ein weiteres Problem ist das Overfitting. Unter **Overfitting** wird im Allgemeinen die Tendenz verstanden, die Modelle auf die Trainingsdaten ideal anzupassen – oftmals als Trainieren bezeichnet – jedoch zu Lasten der Verallgemeinerbarkeit auf vorher nicht entdeckte Daten (vgl. Provost/Fawcett 2013, S. 113). Die Konstruktion eines Entscheidungsbaumes bis zu dem Punkt, an dem alle Blätter homogene Segmente sind, führt grundsätzlich zu Overfitting (vgl. Provost/Fawcett 2013, S. 116 f.). Eine Gegenmaßnahme stellt das sogenannte **Pruning** dar. Es ist das manuelle Verkürzen eines Entscheidungsbaumes, um Overfitting zu vermeiden. Zwei Ansätze können unterschieden werden (vgl. Cleve/Lämmel 2016, S. 109):

- **Pruning während der Baumentwicklung** („Prepruning"): Hierzu wird die Forderung eingeführt, dass jeder Unterbaum eine Mindestanzahl an Instanzen

umfassen muss. Wenn diese unterschritten wird, endet der Ast mit einem Blatt. Das zentrale Problem dieses Ansatzes ist, die Mindestanzahl der Instanzen zu bestimmen.

- **Pruning nach der Baumentwicklung** („Postpruning"): Zunächst wird der kompletten Entscheidungsbaum entwickelt. Anschließend werden sukzessive bereits generierte Unterbäume durch einzelne Blätter ersetzt (vgl. für das Vorgehen bei C4.5 Wu et al. 2008, S. 4). Auch hier stellt sich das Problem, ab welcher Stelle gekürzt werden soll.

Ansätze zur Überwindung der Probleme des Prunings sind vielfältig. Eine Variante ist die Bestrafung von Komplexität bei der Berechnung der Entropie. Eine zweite Möglichkeit ist das Reduced Error-Pruning (vgl. Witten et al. 2017, S. 215): Mittels eines Teils der Datenmatrix der nicht zum Training des Entscheidungsbaums verwendet wurde, werden die Fehlerraten der Knoten und der zugehörigen Blätter berechnet. Ein Schnitt wird dann gesetzt, wenn er zu geringeren Fehlerraten führen würde. Durch Pruning wird der Entscheidungsbaum angehalten, den betrachteten Sachverhalt zu verallgemeinern (vgl. Cleve/Lämmel 2016, S. 109).

Es existieren zahlreiche **weitere Algorithmen** zur Generierung von Entscheidungsbäumen. Zu den verbreitetsten gehören neben dem C4.5-Algorithmus und seinen Weiterentwicklungen noch eine Vielzahl weiterer wie bspw. CHAID und CART (vgl. Wu et al. 2008). Eine tiefergehende Besprechung der verschiedenen Algorithmen erfolgt hier nicht, da hierzu spezialisierte Literatur existiert (vgl. bspw. Aggarwal 2015; Wu et al. 2008).

Eine Weiterentwicklung von Entscheidungsbäumen stellen sogenannte **Random Forrests** dar. Sie sind eine Ausprägung des Ensemble Learnings. Grundgedanke des Ensemble Learnings ist, dass bessere Entscheidungen, hier Klassifikationen, getroffen werden, wenn mehr als eine Methode Grundlage der Entscheidungen sind. Im Falle der Random Forrests werden aus einer Datenmatrix mehrere unterschiedliche Bäume erarbeitet. Die konkrete Klassifikation ist dann eine Mehrheitsentscheidung der Bäume des erarbeiteten Walds. Die Bäume werden auf Basis von Teilmengen der Datenmatrix entwickelt, die jeweils mit Bootstrapping gewonnen wurden, also dem wiederholten Ziehen einer Stichprobe aus der immer vollständigen Datenmatrix. Allerdings wären diese Bäume zu ähnlich, als das tatsächlich eine verbesserte Klassifikation gegenüber einem einzelnen Baum zu erwarten ist. Daher werden nicht alle Attribute, sondern nur ein pro Abzweigung zufällig ausgewählter Teil der Attribute zur Entwicklung des Baums herangezogen. Je kleiner die Anzahl der Attribute, desto geringer die Ähnlichkeit der Bäume (vgl. Hastie et al. 2009, S. 588 f.).

Neben den Entscheidungsbäumen existieren verschiedene **probabilistische Varianten der Klassifikationsanalyse**. Dazu gehören die Naive Bayes-Klassifikation und die logistische Regression: Die **Naive Bayes-Klassifikation** hat das Ziel, die wahrscheinlichste Klasse, der eine Instanz angehört, zu prognostizieren. Die Naive Bayes-Klassifikation ist sehr effektiv und in vielen Fällen komplexeren Klassifikationsalgorithmen überlegen (Witten et al. 2017, S. 105). Der größte Nachteil des Naive Bayes-Algorithmus ist jedoch seine grundlegende Annahme, dass die Attribute unabhängig voneinander sind. Diese Annahme ist in der Realität nicht haltbar, da die Attribute einer Datenmatrix regelmäßig korrelieren (vgl. Aggarwal 2015, S. 310). Einen Überblick über diesen Algorithmus geben Wu et al. 2008.

Ein weiterer Klassifikationsalgorithmus ist die **logistische Regression**. Sie testet, ob ein Zusammenhang zwischen einer abhängigen binären Variablen, dem Klassenattribut, und mehreren unabhängigen Variablen besteht (vgl. Provost/Fawcett 2013, S. 88). Im Gegensatz zur Naive Bayes-Klassifikation bedient sich die logistische Regression einer Diskriminanzfunktion, die die Wahrscheinlichkeit der Klassenzugehörigkeit durch die betrachteten Attribute ausdrückt (vgl. Aggarwal 2015, S. 310).

Als letzte Klassifikationsvariante erörtern wir **künstliche neuronale Netze**. Sie sind inspiriert durch biologische neuronale Netze, wie dem menschlichen Gehirn. Als einfachstes neuronales Netz kann das **Perzeptron** angesehen werden. Abbildung 69 zeigt dessen grundsätzlichen Aufbau. Es besteht aus mehreren Input-Knoten und einem Output-Knoten. Die Gesamtheit der Input-Knoten wird als Input-Layer bezeichnet (vgl. Aggarwal 2018, S. 5).

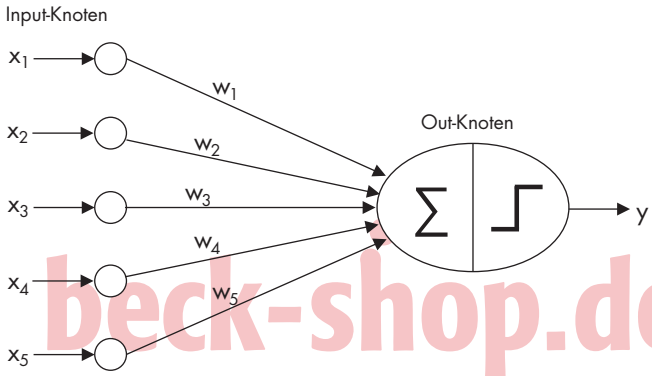

Abbildung 69: Perzeptron (in Anlehnung an Aggarwal 2018, S. 5)

Das durch den Output-Koten berechnete Attribut ist das Klassenattribut y. Ein ist der Maschinenzustand mit den Ausprägungen „Ausfall“ und „Betrieb“. Die Berechnung erfolgt auf Basis der Inputvariablen x_i, die mit den Gewichtungsfaktoren w_i gewichtet werden. Inputvariablen sind sämtliche Attribute der verwendeten Datenmatrix außer des Klassentributs. In unserem Beispiel u. a. Temperatur der Maschine, Maschinenbediener, Luftfeuchtigkeit etc. Der prognostizierte Wert des Klassenattributs $\hat{y}$ berechnet sich dann mittels der sogenannten **Aktivierungsfunktion** (vgl. Aggarwal 2018, S. 5):

$$\hat{y} = sign\left\{\sum_{i=1}^{d} w_i x_i\right\}$$

Der Outputwert ist –1 oder +1; was einer binären Klassifikation entspricht. Das Training des Perzeptrons basiert auf den Trainingsdaten. Im Kern ist es die Gewinnung jener Ausprägungen der Gewichte w_i auf Basis der Differenz zwischen dem tatsächlichen Wert des Klassenattributs y und dem prognostizierten Wert $\hat{y}$. Dieses Vorgehen wird auch als **Delta-Regel** bezeichnet und erfordert das Festlegen einer Lernrate α (vgl. Aggarwal 2018, S. 7):

$$\overline{W} \leftarrow \overline{W} + \alpha\left(y - \hat{y}\right)\overline{X}$$

Die Lernrate α determiniert, wie stark Klassifikationsfehler die Gewichte im nächsten Lernzyklus verändern. In der Regel wird dazu ein Wert zwischen 0,1 und 0,8 gewählt (Cleve/Lämmel 2016, S. 122).

Die Wahl der verwendeten **Aktivierungsfunktion** im Output-Knoten determiniert die Funktionsweise des neuronalen Netzes fundamental. Abbildung 70 zeigt drei Beispiele. Die Signum-Funktion ist geeignet für binäre Klassifikation. Die Sigmoid-Funktion eignet sich für Situationen, in denen die Wahrscheinlichkeit der Zugehörigkeit zu einer von zwei Klassen prognostiziert werden soll. Für die Prognose kontinuierlicher Variablen eignet sich die Identität-Funktion.

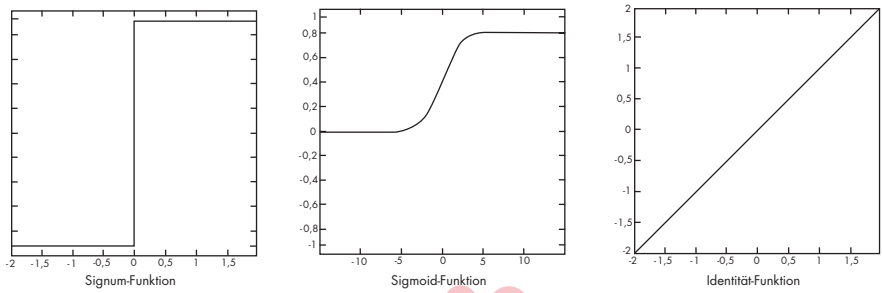

Abbildung 70: Beispielhafte Aktivierungsfunktionen (in Anlehnung an Aggarwal 2018, S. 13)

Das Perzeptron kann nicht nur als einfachstes neuronales Netz interpretiert werden, sondern auch als Grundbaustein für komplexe neuronale Netze. Eine typische Architektur ist das mehrschichtige **Feedforward-Netz**. Mehrere Perzeptrons werden dazu in Schichten organisiert. Die Architektur des neuronalen Netzes wird dann in Input-Layer, Output-Layer und Hidden Layer unterschieden. Die Aufgabe der **Hidden Layer** ist die Transformation der Inputvariablen in eine solche Form, die dem Output-Layer die Klassifikation überhaupt erst erlaubt (vgl. Aggarwal 2018, S. 42 f.). Abbildung 71 zeigt ein beispielhaftes mehrschichtiges Feedforward-Netz.

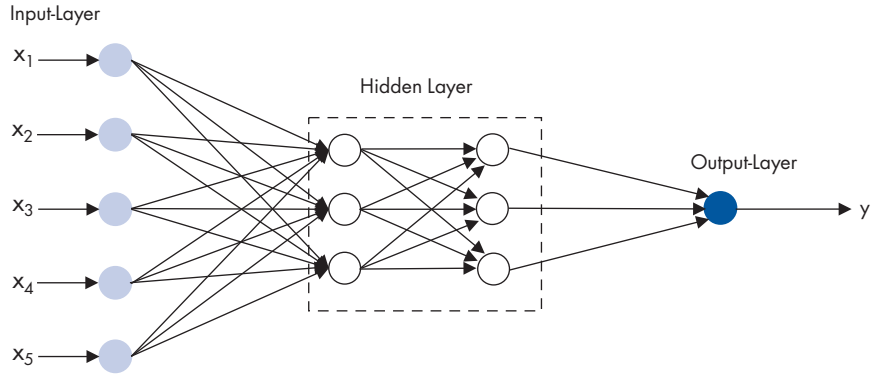

Abbildung 71: Mehrschichtiges Feedforward-Netz (in Anlehnung an Aggarwal 2018, S. 18)