EINE EINFÜHRUNG INS MACHINE LEARNING

Wir ertrinken in Information und dürsten nach Wissen. – John Naisbitt.

Haben Sie sich schon einmal gefragt, wie Google es schafft, Ihnen nur interessante Nachrichten (und Werbung) zu präsentieren? Oder warum die Vorschläge der „Das könnte Sie auch interessieren"-Kategorie bei Amazon Sie so häufig zu einem zusätzlichen, ungeplanten Kauf animieren? Solche Programme entstehen durch Algorithmen aus der künstlichen Intelligenz, genauer gesagt aus dem Machine Learning.

WAS IST MACHINE LEARNING?

Unter dem Begriff Machine Learning, oder Data Mining, versteht man Algorithmen, die es Computern ermöglichen, zu lernen. Lernen bedeutet hier, mit zunehmender Erfahrung, also Daten, die Performance in einer bestimmten Aufgabe zu verbessern.

Ein Untergebiet des Machine Learnings ist die Klassifikation. Hier werden eine Reihe von Daten einer bestimmten Gruppe zugeordnet. Als Beispiel kann man sich eine Bank vorstellen, die Kreditanträge ihrer Kunden in die Gruppen *Wird ausfallen* und *Wird abbezahlt* aufteilen möchte. Dazu verwendet sie vorhandene Daten wie das Alter, das Einkommen und den Schufa-Score, um damit vorherzusagen, ob diese Person die Kreditraten pünktlich zahlen wird.

Falls man nicht eine Kategorie, sondern eine Zahl vorhersagen möchte, befindet man sich im Gebiet der Regression. Ein Beispielproblem ist die Vorhersage der Nettomiete einer Wohnung anhand von Variablen wie Wohnfläche, Zimmeranzahl, Baujahr, Stadtbezirk oder Wohnlage (normal/gut/beste). Tabelle 1 zeigt einen Ausschnitt eines solchen Datensatzes.

TERMINOLOGIE

Ein Machine-Learning-Projekt beginnt immer mit einer Problemstellung und einem Datensatz. Man nennt diese Daten **Trainingsdaten**, da sie zum Trainieren, also zum Lernen des letztendlichen Modells benutzt werden. Die Trainingsdaten sind wiederum aufgeteilt in sogenannte **Features** – das sind die vorhandenen Variablen wie etwa die Wohnungsfläche und die Anzahl der Zimmer im oberen Beispiel – und in den **Outcome**, die vorherzusagende Variable. Dieser Outcome ist nur in den Trainingsdaten der Vergangenheit vorhanden, in neuen Daten ist er noch nicht verfügbar und muss prognostiziert werden. Häufig muss der Outcome manuell von einem Experten hinzugefügt werden. Denn wäre er schon verfügbar, müsste man ihn in den zukünftigen Daten ja nicht vorhersagen. Zum Lernen der Zusammenhänge zwischen Features und Outcomes werden **Modelle** trainiert, die entstehen, indem man einen bestimmten Machine Learning-Algorithmus auf seine Trainingsdaten anwendet. Mit diesen Modellen versucht man, möglichst gute Vorhersagen zu treffen. „Möglichst gut" ist hier abhängig von einem **Gütemaß**, das die Performance des Modells misst. Das einfachste Gütemaß für Klassifikationsprobleme ist zum Beispiel die Fehlklassifikationsrate, also der Anteil an Daten, für den eine falsche Klasse vorhergesagt wurde.

ÜBERWACHTES UND UNÜBERWACHTES LERNEN

Die beiden oben erwähnten Probleme der Vorhersage der Kreditwürdigkeit bzw. der Nettomiete einer Wohnung haben gemeinsam, dass man die vorherzusagende Variable beobachten kann. Man kann also Daten aus der Vergangenheit erheben, in denen man sieht, ob dieser Kredit ausgefallen ist, bzw. für wieviel diese Wohnung vermietet wurde. Man bezeichnet diese Probleme als *supervised learning*, oder überwachtes Lernen, da man bei der Modellerstellung überwachen kann, wie gut eine einzelne Beobachtung vorhergesagt wurde.

Beim *unsupervised learning*, dem unüberwachten Lernen, hingegen kann man die Zielvariable nicht beobachten. Dies ist zum Beispiel bei der Clusteranalyse der Fall. Hier möchte man einen Datensatz in mehrere, zueinander möglichst ähnliche Gruppen aufteilen. Auch Googles PageRank-Algorithmus fällt in diese Kategorie. Dort möchte man eine große Menge von Webseiten anhand ihrer Relevanz für eine bestimmte

Nettomiete	Wohnfläche	Zimmer	Baujahr	Bezirk	Gute Lage	Beste Lage
521.52	46	2	1918	3	1	0
506.18	55	2	1960	5	1	0
392.14	40	1	1960	2	1	0
?	36	1	1957	9	0	0
?	74	3	1966	20	1	0

Tabelle 1: Mithilfe der Trainingsdaten erstellt man ein Modell. Vollständiger Datensatz unter [1].

Suchanfrage sortieren. Dafür bekommt jede Seite ein Gewicht, das davon abhängt, wieviele andere Seiten auf diese Seite verlinken. Diese Seiten sollen selbst jeweils ein möglichst hohes Gewicht haben. Es gibt in dieser Problemklasse keine Möglichkeit, eine Zuordnung als richtig oder falsch einzustufen.

Probleme aus dem unüberwachten Lernen sind allgemein komplizierter als die aus dem überwachten Lernen, weswegen das Hauptaugenmerk der Forschung und Anwendung derzeit auf Letzterem liegt. Mit steigender Rechenpower werden in Zukunft aber die Türen für das unüberwachte Lernen geöffnet. Diese Entwicklung wird eine neue Ära der Datenanalyse einleiten, da diese Problemklasse um Einiges vielseitiger ist: Zum Beispiel lernen Menschen und Tiere auf diese Art. Da man hier auch nicht auf einen Experten angewiesen ist, der manuell einen Outcome für die Trainingsdaten erstellen muss, kann man in diesem Gebiet direkt mit Unmengen an gesammelten Daten arbeiten, und dort interessante Strukturen aufdecken. So sind etwa sehr viel mehr Bilder im Internet verfügbar, bei denen der Outcome, also Katze, Haus oder Auto, fehlt. Nur mit Methoden des unüberwachten Lernens kann man diese Daten analysieren.

KLASSIFIKATION

In den meisten Anwendungsfällen im Machine Learning möchte man für neu erhobene Daten eine gewisse Gruppe vorhersagen. Als Beispiel sehen wir uns einen Datensatz mit 178 Weinen an, für den 13 Features gemessen wurden [2]. Wir konzentrieren uns vorerst auf den Alkoholgehalt in Prozent und den Farbton des Weines. Anhand dieser zwei Features möchten wir für den Wein eine von drei Weinsorten vorhersagen. In Abbildung 1 fehlt die Weinsorte für die drei schwarzen Beobachtungen. Mit dem Auge kann man für die zwei oberen Punkte die Gruppe gut schätzen, aber für den unteren Wein ist die Sache nicht so eindeutig. Gehört er in die grüne oder blaue Gruppe?

K-NEAREST-NEIGHBOR

Der einfachste Algorithmus zum Klassifizieren dieser drei Weine ist der k-Nearest-Neighbor-Algorithmus. Hier werden für eine neue Beobachtung einfach die nächsten k Nachbarn gesucht, und deren Sorte betrachtet. Die Sorte, die auf diesem Plot in der Umgebung am häufigsten vorkommt, ist nun die vorhergesagte Sorte für den neuen Wein.

Dieser Algorithmus hat einen **Tuningparameter**, nämlich k. Sieht man sich für k=1 nur den einen nächsten Nachbarn an, wird der untere Wein in die grüne Gruppe zugeordnet. Eventuell ist dies aber falsch, da der grüne Punkt eine Ausnahme in einer hauptsächlich blauen Umgebung ist. Wir können also die nächsten k=5 Nachbarn ansehen, und würden den unteren Punkt dann in die blaue Gruppe zuordnen, da vier der nächsten fünf Nachbarn aus der blauen Gruppe kommen.

Für das Bestimmen von k gibt es keine allgemeingültige Regel wie „wähle k so groß wie möglich“. Für große Werte von k verliert das Modell nämlich an Lokalität, d.h. es werden irrelevante Punkte mit einbezogen, die sehr weit entfernt liegen. Vielmehr gibt es für k einen Punkt irgendwo zwischen 1 und „sehr groß“, für den die Vorhersage am

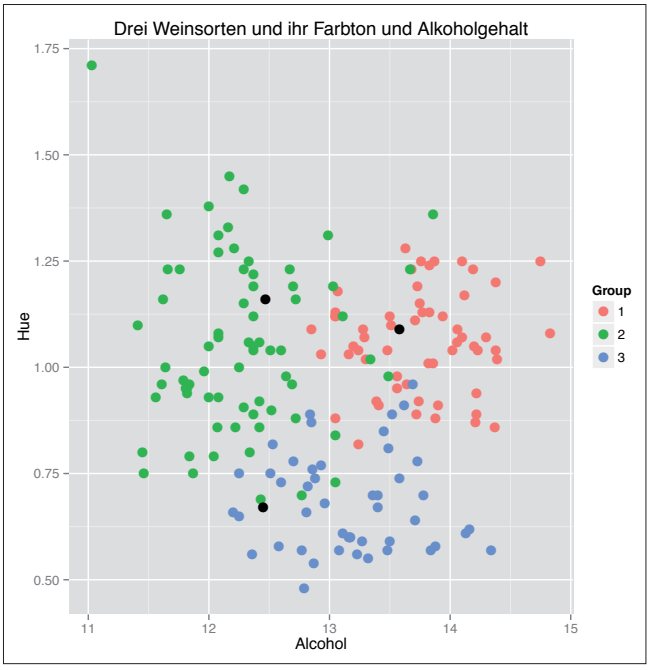

Abbildung 1: Drei Weinsorten anhand ihres Farbtons und Alkoholgehalts grafisch dargestellt.

besten ist. Dieser Wert ist aber für jeden Datensatz und jedes Problem unterschiedlich, und man muss ihn irgendwie finden. Hierzu gibt es verschiedene Verfahren, bei denen der Outcome für einen kleinen Teil der Trainingsdaten entfernt wird, und dann mit Hilfe der übrigen Daten viele Modelle, eines für jeden Tuningparameter, trainiert werden. Ausgewählt wird dann der Tuningparameter, der auf den weggelassenen Daten, für die man die wahre Klasse ja kennt, die beste Prädiktionsgüte hat.

THE CURSE OF DIMENSIONALITY

Der einfache k-Nearest Neighbors-Algorithmus funktioniert bei einem Datensatz mit zwei Features („zweidimensional“) sehr gut. Man kann den Datensatz sehr schön grafisch visualisieren und sich den Algorithmus auch gut vorstellen. Sobald aber mehrere Features vorhanden sind, bricht dieser Algorithmus zusammen. Man nennt dieses Problem den Fluch der Dimensionalität, oder *the curse of dimensionality*. Die Daten liegen dann nicht mehr auf einer Ebene der Dimension 2, sondern in einem Hyperwürfel der Dimension *n*. Abstände zwischen den Datenpunkten werden so exponential größer, und die nächsten Nachbarn findet man erst meilenweit entfernt. Dieses Problem tritt bei verschiedenen Algorithmen mehr oder weniger stark auf. Eine Methode, um mit solchen **hochdimensionalen Daten** dennoch gut zurechtzukommen, sind Klassifikationsbäume.

KLASSIFIKATIONSBÄUME

Ein Klassifikationsbaum, oder Entscheidungsbaum, erstellt eine hierarchische Folge von Regeln, die den Outcome anhand einer Reihe von Aufteilungen der Features prognostiziert. Da Bäume besonders bei vielen Features Sinn machen, sehen wir uns in Tabelle 2 mehrere Features des Weindatensatzes an.

Bei drei beispielhaften Beobachtungen könnte man vermuten, dass Weine mit viel Apfelsäure *(Engl.: malic acid)* und einer hohen Farbintensität vielleicht in Gruppe 3 gehören, wobei Weine mit einer hohen Alkalität und

Group	Alcohol	Hue	Malic.acid	Ash	Alcalinity.of	Magnesium	Total.phenol	Flavanoids	Color.intensi
1	14,12	1,17	1,48	2,32	16,8	95	2,2	2,43	5
2	12,99	1,31	1,67	2,6	30	139	3,3	2,89	3,35
3	13,84	0,57	4,12	2,38	19,5	89	1,8	0,83	9,01

Tabelle 2: Drei Drei beispielhafte Weine aus den Trainingsdaten inklusive des Outcomes „Weinsorte".

Group	Alcohol	Hue	Malic.acid	Ash	Alcalinity.of	Magnesium	Total.phenol	Flavanoids	Color.intensi
?	13,58	1,09	1,66	2,36	19,1	106	2,86	3,19	6,9
?	12,47	1,16	1,52	2,2	19	162	2,5	2,27	2,6
?	12,45	0,67	3,03	2,64	27	97	1,9	0,58	7,5

Tabelle 3 zeigt die vollständigen Features der drei schwarzen Punkte, mit noch unbekanntem, zu prognostizierendem Outcome.

einem hohen Magnesiumgehalt eher in Gruppe 2 gehören.

Abbildung 2 zeigt das Resultat, wenn man auf den vollständigen Daten nun einen Klassifikationsbaum trainiert.

Der Algorithmus fand also drei Variablen, die am besten zur Vorhersage der Gruppe verwendet werden können. In dieser Grafik ist JA immer der linke Zweig, und NEIN der rechte Zweig des Baumes. Zuerst wird für einen neuen Wein geprüft, ob die Farbintensität kleiner als 3.8 ist. Falls ja, geht man nach links und landet in der Vorhersage für Gruppe 2. Falls die

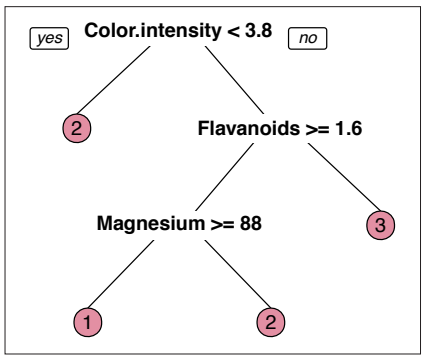

Abbildung 2: Der Klassifikationsbaum auf den vollständigen Daten trainiert

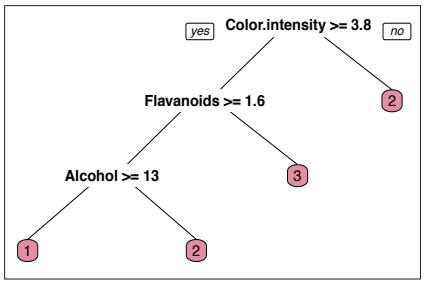

Abbildung 3: Der Klassifikationsbaum bei einer Zufallsauswahl von 90% der Daten.

Farbintensität aber größer als 3.8 ist, prüft man im Anschluss, ob der Flavonoidgehalt größer oder gleich 1.6 ist. Falls nein, prognostiziert dieser Baum die Gruppe 3; falls ja, wird analog dazu der Magnesiumgehalt überprüft, und abhängig davon Gruppe 1 bzw. 2 prognostiziert.

Versuchen Sie doch, die Gruppe für die drei schwarzen Punkte aus Abbildung 1 nun mit diesem Baum vorherzusagen. Tabelle 3 zeigt ihre vollständigen Features.

RANDOM FORESTS

Klassifikationsbäume sind einfache, aber gleichzeitig sehr mächtige Modelle. Sie bringen allerdings einen großen Nachteil mit sich: Sie sind sehr instabil gegenüber Veränderungen in den Daten. Wenn man seine Trainingsdaten nur ein bisschen verändert, kann der resultierende Baum ganz anders aussehen. Abbildung 3 zeigt zum Beispiel den Klassifikationsbaum auf einer zufälligen Auswahl von 90% der Trainingsdaten.

In diesem Baum fällt nun der Magnesiumgehalt zu Gunsten des Alkoholgehalts als Feature heraus. Das ist ein Problem, da die Robustheit eines Algorithmus eine wünschenswerte Eigenschaft ist, um gut auf zukünftige Daten generalisieren zu können.

Diese hohe Variabilität kann man mit Random Forests umgehen. Wie der Name schon vermuten lässt, besteht dieser Algorithmus aus einer Vielzahl an einzelnen Bäumen, die jeweils auf eine Zufallsauswahl der Trainingsdaten losgelassen werden, und dort jeweils nur mit einer Zufallsauswahl der Features arbeiten. Man erhält am Ende eine Menge an einzelnen Bäumen, die jeweils eine Vorhersage für neue Daten machen können. Die prognostizierte Klasse im Random Forest ist nun die mit der Stimmenmehrheit aller einzelnen Bäume. Abbildung 4 veranschaulicht das.

Dieses Prinzip ist als *Wisdom of the crowds* bekannt. Es besagt, dass die aggregierte Meinung einer Gruppe oft eine bessere Schätzung darstellt als die Meinung eines einzelnen Experten.

Random Forests verlieren nichts an der Prädiktionsgüte einzelner Bäume, aber reduzieren die Variabilität der Modelle enorm. Sie sind nicht umsonst einer der verbreitetsten Machine Learning-Algorithmen, die auch auf Plattformen wie kaggle.com, wo Data Scientists gegeneinander antreten, häufig in den besten Modellen vertreten sind.

REGRESSION

Random Forests sind sogenannte **Black-Box-Verfahren**. Sie sind sehr gut zur Prädiktion geeignet, aber der Anwender weiß in der Regel nicht, was im Inneren des Modells vor sich geht. Man kann den Einfluss und die Wichtigkeit einzelner Features nicht oder nur schwer quantifizieren. Falls die Interpretierbarkeit eines Modells im Vordergrund steht, ist man mit statistischen Modellen aus dem Bereich der Regression gut bedient. Regressionsmodelle leiden

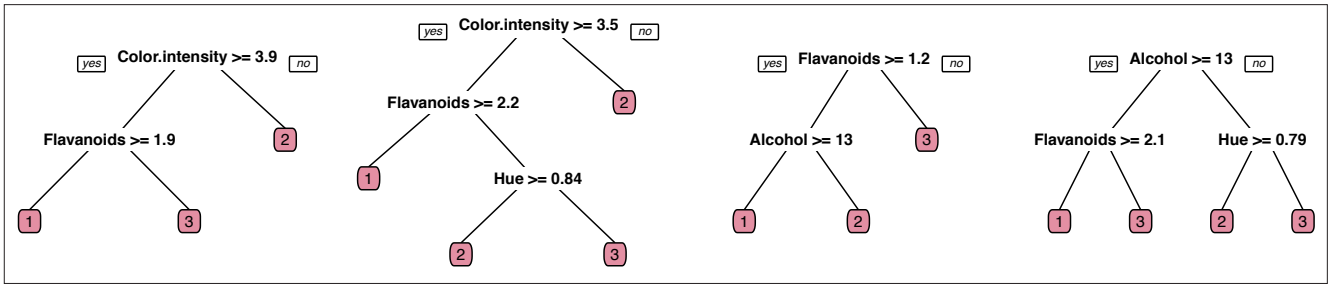

Abbildung 4: Vier beispielhafte Bäume, die zusammen einen (sehr kleinen) Random Forest ergeben.

Alter	Kredithöhe	Kredithistorie	Zweck	Job	Risiko
67	1169	Kritisch	Radio/TV	Angestellter	Gut
22	5951	Aktive Kredite	Radio/TV	Angestellter	Schlecht
49	2096	Kritisch	Bildung	Arbeiter	Gut
45	7882	Aktive Kredite	Möbel	Angestellter	Gut
53	4870	Verzögerung	Neuwagen	Angestellter	Schlecht
35	9055	Aktive Kredite	Bildung	Arbeiter	Gut

Tabelle 4: Ausschnitt eines Datensatzes

aber häufig unter einer schlechteren Prädiktionsgüte, weswegen man hier einen Kompromiss finden muss, der abhängig von der jeweiligen Fragestellung ist.

Ein Klassiker unter den Regressionsmethoden ist die **logistische Regression.** Mit ihr wird eine binäre (ja/nein) Zielgröße vorhergesagt. Spezifisch wird hier die Wahrscheinlichkeit vorhergesagt, mit der eine Beobachtung in eine der beiden Gruppen fällt. Das oben schon angesprochene Beispiel einer Bank, die Kunden in eine gute und eine schlechte Risikoklasse einteilen will, wäre ein Anwendungsfall dieses Problems. Tabelle 4 zeigt einen Ausschnitt eines dafür verwendbaren Datensatzes [3].

Die letzte Spalte ist die Risikoklasse einer Person, die für neue Daten vorhergesagt werden muss. Als Features gibt es zwei numerische Variablen, das Alter und die gewünschte Kredithöhe einer Person, und drei kategoriale Variablen:
Die Kredithistorie: Keine bisherigen Kredite / Alle bisherigen Kredite bezahlt / Aktive Kredite vorhanden / Zahlungsverzögerung vorgekommen / Kritische Historie.
Der gewünschte Zweck des Kredites: Neuwagen / Gebrauchtwagen / Möbel / Radio bzw. TV / Haushaltsgeräte / Reparaturen / Bildung / Urlaub / Umschulung / Geschäftlich / Sonstiges.
Der Job des Antragsstellers: Arbeitslos / Arbeiter / Angestellter / Leitend.

In einem Regressionsmodell bekommt man nun für jedes Feature bzw. Kategorie einen Parameter, der die Stärke des Zusammenhangs beschreibt, sowie einen p-Wert,

Koeffizient	Schätzer	p-Wert
(Intercept)	0,3614	0,3843
Alter	-0,0200	0,0052
Kredithöhe	0,0001	0,0000
KredithistorieAlle bezahlt	1,1581	0,0003
KredithistorieKeine	1,1200	0,0022
KredithistorieKritisch	-0,8244	0,0000
KredithistorieVerzögerung	-0,1430	0,5905
ZweckGebrauchtwagen	-1,9273	0,0000
ZweckGeschäftlich	-0,9456	0,0154
ZweckHaushaltsgeräte	-0,5970	0,3973
ZweckMöbel	-0,7751	0,0261
ZweckNeuwagen	-0,3580	0,2851
ZweckRadio/TV	-1,1575	0,0007
ZweckReparaturen	-0,3884	0,4899
ZweckSonstiges	-1,1467	0,1318
ZweckUmschulung	-2,3620	0,0380
JobArbeiter	-0,0466	0,8101
JobArbeitslos	-0,1756	0,7425
JobLeitend	0,1511	0,5056

Tabelle 5: Die geschätzten Parameter des logistischen Regressionsmodells.

der Aufschluss darüber gibt, ob der Zusammenhang für dieses Feature statistisch signifikant ist. Meistens wird ein p-Wert kleiner als 0.05 als signifikant angesehen.
Tabelle 5 zeigt die geschätzten Parameter für diese Daten. Wir sehen anhand des Vorzeichens der Schätzer, ob die Variable einen positiven oder negativen Einfluss auf die Bonitätsgruppe Gut hat. Beim Alter sehen wir beispielsweise, dass mit steigendem Alter die Wahrscheinlichkeit fällt, in die gute Kreditwürdigkeitsgruppe eingestuft zu werden. Für die Kredithöhe ist es umgekehrt: Je höher der gewünschte Kredit ausfällt, desto wahrscheinlicher wird der Antragsteller in die gute Gruppe eingestuft.

Bei den kategorialen Features gibt es einen Schätzer pro möglichem Wert. Für die Kredithistorie sieht man zum Beispiel, dass die Kreditwürdigkeit stark sinkt, wenn die Kredithistorie kritisch ist. Für Antragsteller ohne bisherige Kredite und die, die alle bisherigen Kredite pünktlich bezahlt haben, steigt die Kreditwürdigkeit dagegen. Die Gruppe *Aktive Kredite* taucht nicht auf, da sie hier die Referenzkategorie bildet. Ihr Schätzwert ist in der ersten Reihe, im Intercept, enthalten, und die Schätzer der übrigen Kategorien werden relativ zu dieser Gruppe betrachtet.
Den genauen Wert der Wahrscheinlichkeit, in die Gruppe *Gut* eingestuft zu werden, berechnet man durch die Formel $\frac{e^{\eta}}{1+e^{\eta}}$
Hier bezeichnet η den sogenannten linearen Prädiktor: $\eta = \beta_0 + x_1\beta_1 + \cdots + x_p\beta_p$
Man berechnet ihn, indem man den Intercept β_0 zu allen relevanten geschätzten Koeffizienten, multipliziert mit dem Wert des jeweiligen Features, addiert. Für einen 30-jährigen Arbeiter, der aktive Kredite hat, und 5000 Euro für einen Gebrauchtwagen leihen möchte, hätten wir also
η=0.3614-30*0.01999+5000*0.0001269-1.927-0.04664=
-1.577
Damit können wir die Wahrscheinlichkeit für die Bonitätsgruppe *Gut* berechnen: $\frac{e^{-1.577}}{1+e^{-1.577}} = 0.1711$
Mit einer prognostizierten Wahrscheinlichkeit von 17,11% für eine gute Bonität sieht es für diesen Kreditantragssteller also eher mau aus.

DEEP LEARNING UND NEURONALE NETZE

Neuronale Netze bezeichnen eine Modellklasse, die in den letzten Jahren einen gigantischen Hype erfahren hat. Das liegt zum einen an ihrer Namensgebung: neuronale Netze imitieren mit Hilfe vieler einzelner Neuronen und komplexer Verbindungen, den Synapsen, das Lernverhalten eines Gehirns. Zum anderen liegt das aber auch an der immensen Vielseitigkeit der Aufgaben, die diese Modelle bewältigen können. Sie können sowohl mit Regressions- als auch Klassifikationsproblemen umgehen, und dort jeweils sehr komplexe Zusammenhänge erkennen.

Ein einfaches neuronales Netz funktioniert wie folgt: Das Netz verwendet eine Reihe von Features, und transformiert diese Features mit Hilfe mehrerer Lineartransformationen in eine Reihe von Units – das sind Zwischenwerte, die zusammen ein sogenanntes Hidden Layer ergeben (siehe Abbildung 5). Auf diese transformierten Features wird nun eine nichtlineare Funktion

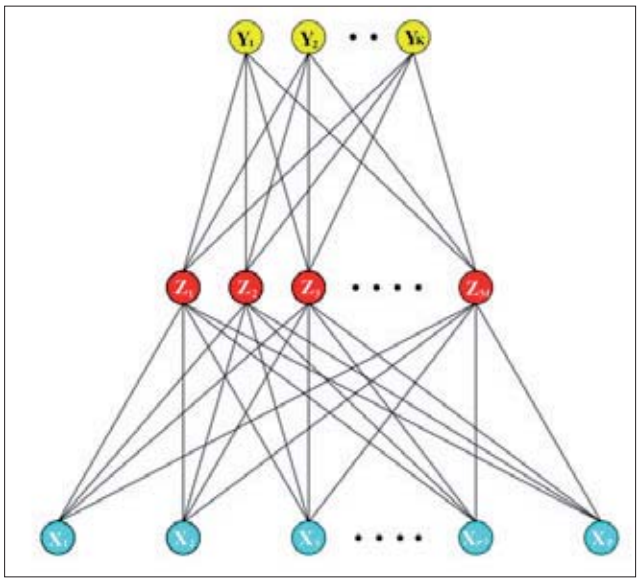

Abbildung 5: Schema eines neuronalen Netzes mit einem Hidden Layer Z, den Input-Features X, und dem Output-Layer Y. Quelle: [7]

angewendet, wodurch das neuronale Netz seine große Flexibilität erhält. In den frühen Jahren war diese Funktion eine einfache 0/1-Aktivierungsfunktion, was die Analogie zu einem Gehirn erklärt: Sobald der Input eines Neurons eine bestimmte Grenze überschreitet, feuert es – das heißt sein Output wird 1.

Mit der Anzahl an Hidden Layers steuert man die Komplexität des Netzes, ein Kompromiss zwischen benötigter Rechenpower und Flexibilität des Modells. In der Abbildung werden die Input Features X1,..., XP in das Hidden Layer Z1,..., ZM transformiert, aus denen das Output Layer Y1,...,YK erstellt wird. Für Regressionen besteht das Output Layer meist aus nur einer Einheit Y1, für Klassifikationen gibt es ein Neuron pro Klasse, und jedes Neuron Yp repräsentiert die Wahrscheinlichkeit, in Klasse p zu gehören.

Insbesondere mit Deep Learning, welches sich sehr vieler Hidden Layers behilft, kann man vieler aktueller Probleme Herr werden: Sie werden bereits erfolgreich für Aufgaben wie Spracherkennung, automatische Übersetzung, und Bilderkennung verwendet.

Für diese komplexen Aufgaben sind neuronale Netze wie geschaffen: In der Bilderkennung etwa können sie mit Hilfe der vielen nacheinandergeschalteten Schichten in aufeinanderfolgenden Schritten zum Beispiel zuerst nach einfachen Kanten in einem Bild suchen, in einem zweiten Schritt dann nach Gruppen von Kanten, die zusammen Formen wie Dreiecke oder Kreise bilden, und

Abbildung 6: Ziel des MNIST-Datensatzes [5] ist es, anhand der 16x16 Pixel die Ziffer zu prognostizieren.

in späteren Schritten dann Gesichter oder Gesichtsausdrücke erkennen. Diese Vielseitigkeit und Vorhersagegüte geht allerdings auf Kosten der Interpretierbarkeit der trainierten Modelle. Da die Vorhersage auf einer tiefen Verschachtelung linearer und nichtlinearer Transformationen der Daten beruht, ist es für den Anwender z.B. unmöglich, im Anschluss herauszufinden welche Features für die Vorhersage wichtig sind, und wie groß ihr Einfluss ist. Das macht neuronale Netze für Felder wie die Medizin, in der Interpretierbarkeit ein Schlüsselfaktor ist, weniger anwendbar.

Ein klassisches Beispiel, in dem neuronale Netze triumphieren, ist das ZIP-Code-Problem (siehe Abbildung 6). Hier wollte der U.S. Postal Service anhand vieler gescannter Postleitzahlen ein System zur automatischen Erkennung und Sortierung entwickeln. Dieser Datensatz wird seit Jahren als Standardbenchmark für Machine Learning-Algorithmen verwendet. Neuronale Netze können mit diesen Daten sehr gut umgehen, und erreichen eine Fehlerrate von etwa 1%.

UNÜBERWACHTES LERNEN

CLUSTERANALYSE

Die Clusteranalyse ist ein klassischer Vertreter im Bereich des unüberwachten Lernens. Man versucht hier, einen Datensatz in mehrere Untergruppen zu segmentieren, innerhalb derer sich die Daten einander möglichst ähneln. Anwendungen finden sich zum Beispiel in der Bioinformatik, wo man Gene mit noch unbekannter Funktion in Cluster zusammenfasst, um Verwandtschaften zwischen Genen aufzudecken. Auch in der Marktforschung bildet man Cluster von Konsumenten, um Gemeinsamkeiten und typische Verhaltensweisen von Kundengruppen zu finden. Unüberwacht ist diese Methode deshalb, weil es keinen wahren Cluster für eine Beobachtung vorherzusagen gibt, sondern wir ihn durch die Clusteranalyse erst erstellen.

Zentral bei einer Clusteranalyse ist das sogenannte **Distanzmaß**, mit dem man die Ähnlichkeit zwischen zwei Beobachtungen misst. Welches Distanzmaß man wählt, ist wieder abhängig von der Fragestellung und der Struktur der Daten. Am häufigsten kommt wohl die euklidische Distanz zum Einsatz: der einfache Abstand wie wir ihn kennen. Die euklidische Distanz zwischen zwei Beobachtungen (d.h. Vektoren) x und y mit jeweils n Elementen ist definiert als

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Mit diesem Distanzmaß hat man nun eine Möglichkeit, die Ähnlichkeit zwischen zwei Beobachtungen zu quantifizieren, und einen Clusteralgorithmus zu starten. Der verbreitetste ist der **k-means-Algorithmus**. Man bestimmt hier vorher eine Anzahl an k Clustern, in die die Daten segmentiert werden, und geht dann folgendermaßen vor:

1. Verteile die k Clusterzentren beliebig in den Daten.

2. Iteriere die folgenden Schritte, bis sich nichts mehr verändert:

- ***a.*** Ordne jede Beobachtung in den Cluster mit dem nächstliegenden Zentrum zu.
- ***b.*** Berechne nun die neuen Clusterzentren als

Schwerpunkt aller Daten, die in dem jeweiligen Cluster liegen.

Dieser Algorithmus konvergiert im Allgemeinen recht schnell. Möchten wir anhand der beiden Variablen Alkoholgehalt und Apfelsäuregehalt etwa drei Cluster aus den 178 Weinen bilden, sehen die ersten drei Iterationen wie in Abbildung 7 dargestellt aus.

In Iteration 0 haben wir die Clusterzentren beliebig auf den Daten verteilt. In der ersten Iteration wird für jede Gruppe der neue Schwerpunkt berechnet, wodurch das rote Zentrum nach oben, und das schwarze und grüne Zentrum nach unten gezogen werden. Zwischen der zweiten und dritten Iteration ändert sich nicht mehr viel, und nach fünf Iterationen wird dieser Algorithmus konvergiert sein.

GOOGLES PAGERANK-ALGORITHMUS

Der PageRank-Algorithmus von Google ist eine der einflussreichsten modernen Entwicklungen im unüberwachten Lernen. Er bringt eine Reihe von N Suchergebnissen in eine Rangfolge, geordnet nach Relevanz für den Suchenden. Hierbei wird die Wichtigkeit einer Webseite bestimmt durch die Anzahl anderer Webseiten, die auf diese Seite verlinken. Die anderen Webseiten werden allerdings wieder mit ihrem jeweiligen PageRank gewichtet. Der PageRank-Algorithmus ist also rekursiv definiert:

Sei $L_{ij}=1$, wenn die Webseite j auf die Seite i verlinkt, und 0 wenn nicht. Sei $c_j = \sum_{i=1}^{N} L_{ij}$ die Gesamtzahl an Seiten, auf die die Webseite j verlinkt. Dann ist der PageRank p_i für die Webseite i definiert als

$$p_i = (1-d) + d\sum_{j=1}^{N}\left(\frac{L_{ij}}{c_j}\right)p_j$$

Die Grundidee ist, dass die Wichtigkeit von Webseite i die Summe der Wichtigkeiten aller Webseiten ist, die auf diese Seite verlinken. Die Gewichte c_j stellen sicher, dass jede Webseite durch Outlinks ein Gesamtgewicht von 1 auf andere Webseiten verteilen kann. Die Konstante d ermöglicht, dass jede Webseite mindestens einen PageRank von $1-d$ bekommt. Die Berechnung der PageRanks reduziert sich auf ein Eigenwertproblem, auf das hier aber nicht näher eingegangen wird. Details hierzu findet man im Originalartikel unter [6].

Das Machine Learning ist ein sehr weites Gebiet, mit dessen Werkzeugen man komplexe Fragestellungen aus breit gefächerten Problemgebieten lösen kann. Mit dem unüberwachten Lernen ist eine Klasse von Algorithmen entstanden, die selbstständig ohne menschliche Überwachung lernen kann, was im Zeitalter stets wachsender Datensätze eine Menge neuer Möglichkeiten eröffnet. Man darf gespannt sein, welche Entwicklungen sich in den nächsten Jahren noch ergeben werden.

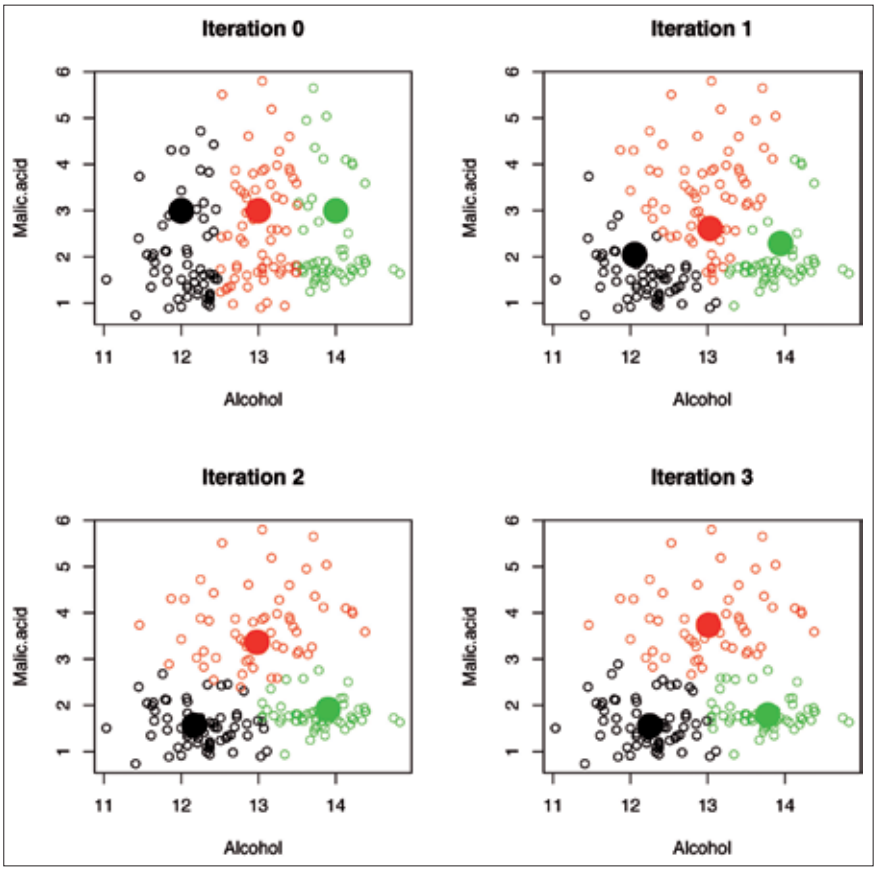

Abbildung 7 zeigt die ersten drei Iterationen des k-Means-Algorithmus auf dem Weindatensatz

QUELLEN:

[1] http://www.statistik.lmu.de/service/datenarchiv/miete/miete03.html

[2] https://archive.ics.uci.edu/ml/datasets/Wine

[3] https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

[4] http://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

[5] http://yann.lecun.com/exdb/mnist/

[6] http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[7] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The Elements of Statistical Learning. Vol. 2. Springer, Berlin: Springer series in statistics, 2011.

ALEXANDER ENGELHARDT ist Statistiker mit besonderem Interesse am Machine Learning. Er unterstützt als Freelancer Unternehmen bei Datenauswertungen und dem Erstellen von Prognosemodellen.

engelhardt@alpha-epsilon.de

http://www.alpha-epsilon.de/