KÜNSTLICHE INTELLIGENZ, TEIL 1

Vom Machine Learning zum Deep Learning

Welche Algorithmen gibt es, um dem Computer Intelligenz beizubringen?

Einmal Gott spielen und Leben erschaffen! Seit der Antike haben die Menschen davon geträumt, einen Ebenbürtigen künstlich zu schaffen, der wie wir Menschen aussehen, handeln und vor allem denken könnte. Der antike Bildhauer Pygmalion soll die Galatea aus Elfenbein und der Feuergott Hephaistos die Pandora aus Lehm geschaffen haben, die dann zum Leben erweckt wurden.

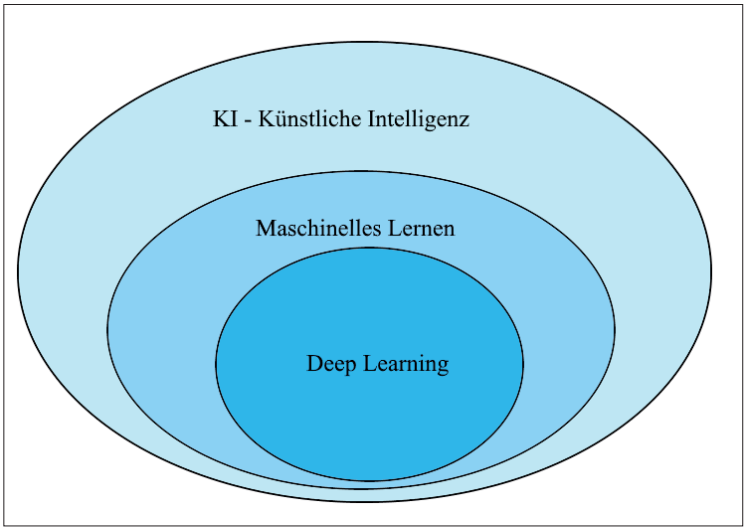
Im viktorianischen England hatte sich Gräfin Ada Lovelace – Freundin und Partnerin des Erfinders der Analytical Engine [1], Charles Babbage – die Frage gestellt, ob Rechenmaschinen denken können, als sie Notizen zu ihrer Übersetzung aus dem Französischen ins Englische von L. Menabreas „Sketch of the Analytical Engine" verfasste [2]. Damals hatte sie die Maschinen-Intelligenz nicht für möglich gehalten: *„Die Analytical Engine kann nichts erfinden. Sie kann [nur] das tun, was wir ihr befehlen. Sie kann Analyseschritte verfolgen, aber sie kann keine analytischen Beziehungen oder Wahrheiten vorhersagen. Ihre Aufgabe ist, für uns das verfügbar zu machen, was wir bereits kennen."*

Diese Auffassung jedoch wurde ein Jahrhundert später von dem britischen Informatiker und Vater der künstlichen Intelligenz, Alan Turing [3], in seinem visionären Artikel „Computer Machinery and Intelligence" [4] kritisiert. Damals argumentierte Turing, dass eine Maschine durchaus eine Intelligenz beziehungsweise intelligentes Verhalten zeigen könne. *„Die Analytical Engine war ein universeller digitaler (mechanischer) Computer. Wenn seine Speicherkapazität und Geschwindigkeit angemessen wären, wäre es durch geeignete Programmierung machbar gewesen, infrage kommende (intelligente) Maschinen nachzuahmen. Wahrscheinlich fiel dieses Argument der Gräfin (Ada Lovelace) oder dem Babbage nicht ein."*

In demselben Artikel wird unter anderem auch der Turing-Test vorgestellt [5]: ein Gedankenexperiment, wie man feststellen könnte, ob ein Computer beziehungsweise eine Maschine eine mit dem Menschen vergleichbare Intelligenz aufweist.

KI – Künstliche Intelligenz

Alan Turing hatte Ende 1940 den Weg zur künstlichen Intelligenz als Informatikdisziplin geebnet. Allerdings hatte er damals den Begriff Intelligent Machinery verwendet, um intelligente Verhaltensmuster von Computern zu beschreiben. Der Begriff Artifical Intelligence (künstliche Intelligenz) wurde von John McCarthy [6] eingeführt, als er am 31. August 1955 zusammen mit Marvin Minsky, Nathaniel Rochester und Claude Shannon (Bell) einen Förderantrag an die Rockefeller-Stiftung gestellt hatte.



Wie sich KI, Machine– und Deep Learning zueinander verhalten (Bild 1)

Hier wurde Artifical Intelligence zum ersten Mal erwähnt. Ein Jahr später, im Sommer 1956, fand das „Dartmouth Summer Research Project on Artificial Intelligence" statt, das als Geburtsstunde von künstlicher Intelligenz als akademisches Fachgebiet gilt.

Kurz gesagt bedeutet KI eine Automatisierung intellektueller Aufgaben, die sonst von Menschen erledigt werden (siehe Kasten Künstliche Intelligenz (KI)). Insbesondere in den letzten paar Jahren erlebte das Thema KI einen regelrechten Hype. Begriffe wie KI, maschinelles Lernen und Deep Learning kommen in zahlreichen Artikeln, Vorträgen und Berichten vor. Außer einer Beleuchtung rein technischer Aspekte wird auch eine mal utopische, mal düstere Zukunft prophezeit: von selbstfahrenden Fahrzeugen und selbstdenkenden Agenten bis zu Horrorszenarien, wenn unkontrollierte intelligente Waffensysteme die Menschheit von der Erde tilgen würden.

Um die Spreu vom Weizen zu trennen, lassen Sie uns klären, was man meint, wenn man über künstliche Intelligenz, maschinelles Lernen und Deep Learning spricht, und wie sich diese Begriffe zueinander verhalten. Das wird in Bild 1 sichtbar – Deep Learning ist eine Art des maschinellen Lernens, was wiederum einen wichtigen großen Zweig der gesamten künstlichen Intelligenz in der Informatik darstellt.

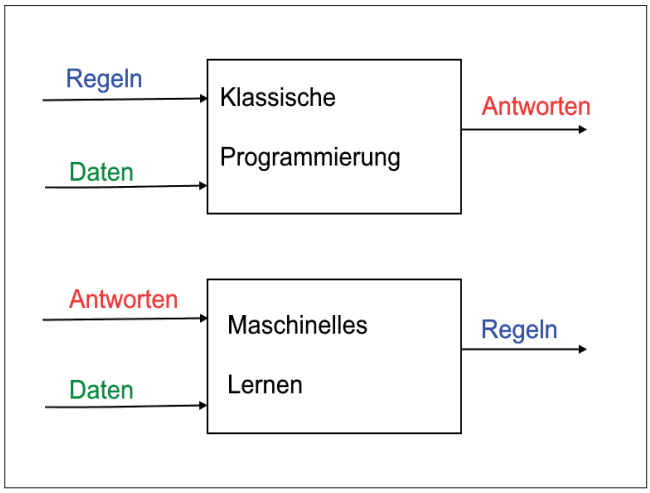
Wie man sieht, schließt die künstliche Intelligenz zwar das maschinelle Lernen und damit auch das Deep Learning ein, ist aber nicht darauf begrenzt. Künstliche Intelligenz hat andere Methoden im Programm, die keinen Lernmechanismus aufweisen. Zum Beispiel wurden frühere Schachprogramme ohne jegliche Lernmöglichkeiten nur mit hart codierten Regeln programmiert. Ziemlich lange Zeit glaubten viele Experten, eine menschliche Intelligenz zu erreichen, indem man eine genügend große Anzahl des hart codierten Wissens und der Regeln zu dessen Manipulation eingeben würde. Diese Vorgehensweise ist als symbolische KI (Symbolic AI) bekannt und galt als dominierendes Paradigma von 1950 bis 1980.

Ein weiteres prominentes Beispiel von Symbolic AI sind die in den 1980er Jahren boomenden Expertensysteme [7]. Die Methoden und Ansätze der symbolischen KI eigneten sich zwar recht gut für bestimmte, logisch wohldefinierte Aufgaben wie zum Beispiel das Schachspiel oder die Steuerung/Planung eines deterministischen Prozesses, sie scheiterten aber bei der Lösung komplexerer, undeutlicher Aufgaben wie Bildklassifikation, Spracherkennung oder Übersetzung. Um diese neuen Aufgaben zu lösen, war ein neues KI-Paradigma nötig: das maschinelle Lernen.

ML – Maschinelles Lernen

Maschinelles Lernen (Machine Learning) als Informatikdisziplin beantwortet die Frage mit ja, ob ein Computer über seine Grenzen hinweggehen kann, indem er nicht nur das tut, was ihm gesagt wurde, sondern sich anhand gegebener Daten und gegebenenfalls Fehlern nötige Regeln selbst erarbeitet.

Die positive Antwort auf die letzte Frage wird durch ein neues Programmierparadigma ermöglicht (Bild 2). Während im klassischen Programmiermodell (auch bei symbolischer KI) ein Mensch die Regeln (Programm) und die Daten eingibt und die Antworten als Ergebnis bekommt, werden beim maschinellen Lernen die Daten und Antworten eingegeben und die Regeln als Ergebnis produziert. Die so gewonnenen Regeln wiederum können anschließend auf neue Daten angewendet werden, um richtige Antworten zu bekommen.



Programmierparadigmen der klassischen Programmierung und des Machine Learning (Bild 2)

Maschinelles Lernen (ML)

Maschinelles Lernen (engl. Machine Learning, kurz ML) ist ein Oberbegriff für das künstliche Generieren von Wissen aus Erfahrung. Ein künstliches System lernt aus Beispielen und kann nach Beendigung der Lernphase verallgemeinern. Das heißt, es lernt nicht einfach die Beispiele auswendig, sondern es erkennt in den Lerndaten Gesetzmäßigkeiten. Für Software heißt das nach Thomas Mitchell: Ein Computerprogramm lernt beim Lösen einer bestimmten Klasse der Aufgaben (T), wenn seine messbare Leistung (P) sich mit der Erfahrung (E) im Lauf der Zeit erhöht [30].

Im klassischen Fall sagt man, dass ein Programm programmiert wird. Beim Fall des maschinellen Lernens spricht man davon, dass ein System trainiert wird. Man präsentiert einem ML-System viele für die aktuelle Aufgabe relevante Datenbeispiele, und es versucht, passende statistische Strukturen zu finden, die es dann später erlauben, die Zielaufgabe zu automatisieren.

Will man zum Beispiel Tierbilder kategorisieren, gibt man dem ML-System viele bereits von einem Menschen (oder gar einem anderen geprüften ML-System) kategorisierte Bilder, und es wird statistische Regeln lernen, um mit ihnen neu ankommende Bilder selbst zu kategorisieren. Beginnend mit den 1990ern ist das maschinelle Lernen insbesondere in den letzten Jahren erfolgreich und populär geworden, zu dem nicht zuletzt schnelle Hardware und große Datenmengen beigetragen haben.

Der Autor hat in der dotnetpro das maschinelle Lernen auf Azure ML Studio vorgestellt [8]. Dort findet man auch eine Zusammenfassung von dem, was man sich unter dem Begriff maschinelles Lernen vorstellt (siehe den Kasten Maschinelles Lernen (ML)).

ML nutzt statistische und mathematische Methoden und Algorithmen, um Aufgaben wie konzeptionelles Lernen, Daten-Vorhersagen/Auswertung, Clustering und Entdeckung verwertbarer Muster zu lösen. Dabei wird versucht, so viel wie möglich zu automatisieren, indem ein ML-Modell seine Entscheidungen und Auswertungen so intelligent wie möglich trifft, sodass im Idealfall gar keine oder zumindest nur minimale menschliche Interaktion notwendig ist.

Fast jeder ML-Algorithmus kann als eine Lösung zum Optimierungsproblem [9] gesehen werden, wenn man einen Lösungsraum sucht, bei dem die relevante Bewertungsfunktion (auch Zielfunktion benannt) ihren maximalen beziehungsweise minimalen Wert hat. ▶

Typischerweise besteht eine Zielfunktion aus zwei Komponenten: einem Regularisierer (Regularizer), der die Komplexität des ML-Modells steuert, und dem Verlust (Lost), der den Fehler des Modells in den Trainingsdaten misst.

Während man logischerweise versucht, den Verlust zu minimieren, hilft optimale Regularisierung des ML-Modells, einen Kompromiss zwischen dem minimalen Verlustwert und zu hoher Komplexität zu finden und sogenannte Überanpassung (Overfitting) zu vermeiden.

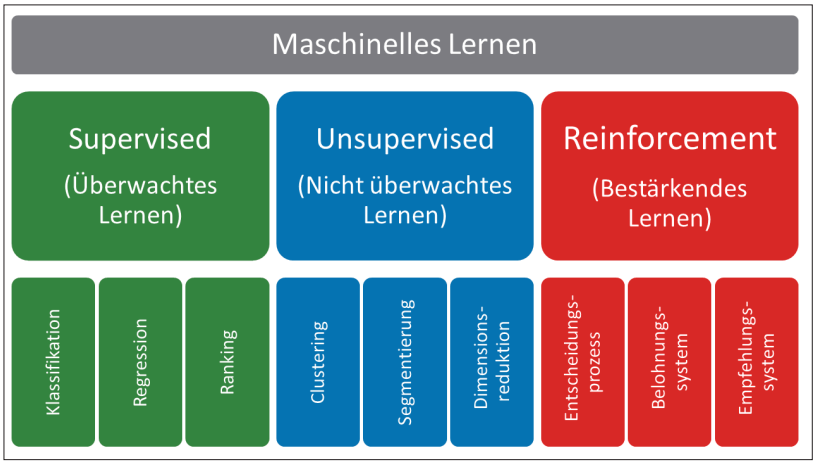
Von der Datenperspektive her braucht ein Lernprozess normalerweise folgende drei Datensätze:

- **Trainingssatz:** Eine Wissensbasis, mit dem ein ML-Modell trainiert wird. Unter dem Training versteht man die iterative Anpassung von speziellen Gewichtungsparametern, bis die relevante Zielfunktion ihren optimalen Wert erreicht.
- **Validierungssatz:** Daten, die zum Feintuning von Parametern des ML-Modells verwendet werden. Zum Beispiel kann man mit einem Validierungssatz die Anzahl der versteckten Schichten in einem Deep-Learning-Modell oder einen Stopp-Punkt für einen Backpropagation-Algorithmus finden. Man entwickelt also mit dem Validierungssatz ein ML-Modell zu einem optimalen Zustand, darum heißt er auch manchmal Entwicklungssatz (Dev/Development Set).
- **Testsatz:** Er wird verwendet, um die Leistung eines ML-Modells bei ungesehenen Daten zu prüfen/testen. Nach der Beurteilung des endgültigen ML-Modells mit dem Testsatz wird dieses in die Produktionsumgebung verteilt.

Der Lernprozess eines ML-Models erfordert also eine Aufteilung zur Verfügung stehender Daten auf zwei oder mehr Teile. Dieser Schritt ist in der ML-Terminologie als Splitting bekannt. Bei den klassischen ML-Methoden wird normalerweise ein Splitting auf zwei der oben beschriebenen Datensätze vorgenommen: Training und Test wie beim Beispiel der Einnahmenvorhersage mit Azure ML Studio in [8].

Bei den Deep-Learning-Modellen (DL) dagegen braucht man noch einen Validierungsdatensatz dazu, um eine Anpassung/Entwicklung eines DL-Modells vorzunehmen, indem man zum Beispiel die Anzahl und Größe von Modellschichten für bessere Performance optimiert. Der Anteil von jedem der Splitting-Datensätze am gesamten Datenvolumen ist in der Regel intuitiv, variabel und anpassbar, dennoch werden häufig als Anfangswerte 70 und 30 Prozent jeweils für Trainings- und Testsatz beim zweifachen Splitting und 50, 20 und 30 Prozent für Trainings-, Validierungs- und Testsatz beim dreifachen Splitting angesetzt.

Die Theorie des maschinellen Lernens stützt sich auf drei führende Lernparadigmen des überwachten, unüberwachten und bestärkenden Lernens. In der nachfolgenden Auflistung und in Bild 3 finden Sie eine kurze Beschreibung jedes Lerntyps sowie die Aufgabenkreise, welche die auf dem jeweiligen ML-Typ aufgebauten Modelle lösen.



ML-Paradigmen und korrespondierende Probleme (Bild 3)

- **Überwachtes Lernen** (Supervised Learning, SL) ist das bekannteste und populärste Paradigma des maschinellen Lernens [10]. Es basiert auf einer Anzahl von vordefinierten Beispielen, in denen relevante Kategorien (auch Label genannt) einzelner Eingangsdatenelemente bereits bekannt sind. In diesem Fall ist das entscheidende Problem die Verallgemeinerung (auch bekannt als Generalisation) der analysierten Datensätze. Nach der Analyse einer typischen Stichprobe von Beispielen sollte das Supervised-ML-System ein SL-Modell erzeugen, das für alle möglichen Eingaben gut funktioniert. Kategorisierte Datenelemente mit dazugehörigen Werten bilden in einem SL-Modell logischerweise einen Trainingsdatensatz, wobei die Kategorien für Aufgaben wie die Klassifikation und die Werte von einzelnen Datenelementen im Trainingsdatensatz für die Regression benutzt werden können (siehe Tabelle 1).
- **Beim nicht überwachten Lernen** [11] (unüberwachtes Lernen, Unsupervised Learning, USL) wird dem USL-System während der Trainingsphase ein Eingangsdatensatz zur Verfügung gestellt. Im Gegensatz zum überwachten Lernen (Su-

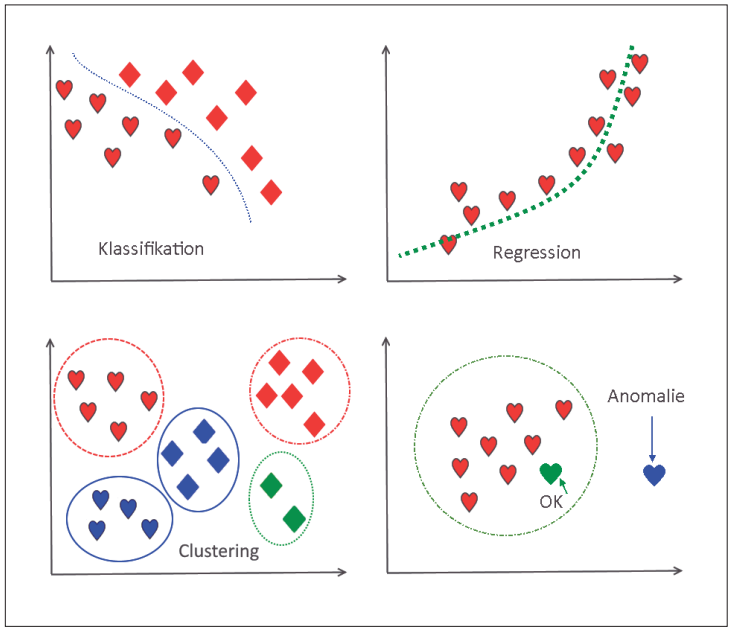
Künstliche Intelligenz (KI)

Künstliche Intelligenz (KI) (AI – Artifical Intelligence) ist eine Informations- und Ingenieurswissenschaft, die sich der Herstellung intelligenter Maschinen und speziell intelligenter Computerprogramme widmet. KI befasst sich mit der Aufgabe, mithilfe von Computeranwendung menschliche Intelligenz zu verstehen. Aber KI muss sich nicht auf die Methoden beschränken, die biologisch beobachtbar sind. Unterschiedliche Arten und Grade der Intelligenz treten bei den Menschen, bei vielen Tieren und in einigen Maschinen auf. Das heißt also, ein Computer wird so gebaut und/oder programmiert, dass er eigenständig Probleme bearbeiten und lösen, aus den Fehlern lernen, Entscheidungen treffen, seine Umgebung wahrnehmen und mit Menschen auf natürliche Weise (zum Beispiel sprachlich) kommunizieren kann.

Tabelle 1: Typische ML-Aufgaben

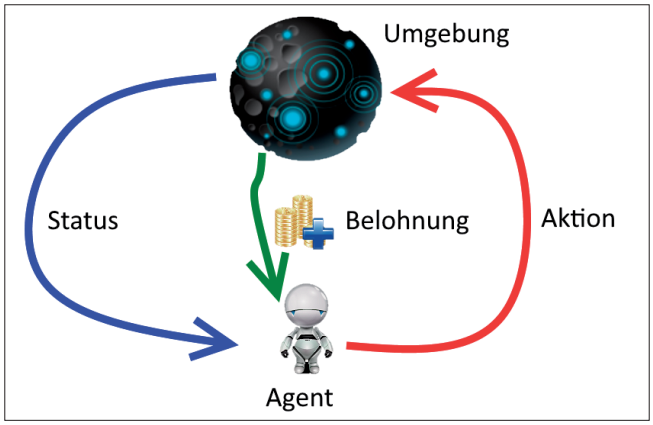
Aufgabe	Beschreibung	Typ des Lernalgorithmus
Klassifikation (Classification)	Die Eingangsdaten werden bestimmten Kategorien (Labels) zugordnet. Wenn es nur zwei mögliche Kategorien gibt (zum Beispiel „wahr" oder „falsch" für eine Aussage beziehungsweise „Apfel" oder „Birne" für eine Obstmenge et cetera) spricht man von der Zweierklassen- oder der binomialen Klassifikation, andernfalls von Multiklassen-Klassifikation.	Supervised
Regression (Regression)	Bei Regressionsaufgaben muss man einen numerischen Wert bei einer gegebenen Eingabe vorhersagen. Regressionsaufgaben ähneln der Klassifikation, nur dass das Ausgabeformat unterschiedlich ist (numerischer Wert anstelle einer Kategorie/Label). Ein Beispiel für eine Regressionsaufgabe ist die Vorhersage des erwarteten Schadensbetrags, den eine versicherte Person leisten wird, oder die Vorhersage zukünftiger Preise von Wertpapieren. Diese Regressionsvorhersagen werden zum Beispiel auch beim algorithmischen Handel verwendet.	Supervised
Clustering (Clustering)	Eine ungesehene Menge von Objekten wird so gruppiert, dass Objekte in derselben Gruppe – sogenannte Cluster – einander ähnlicher sind als solche in anderen Gruppen/Clustern. Ein Clustering-Algorithmus könnte automatisch die Songs in einer Musikbibliothek nach Stil gruppieren: Rock, Pop, Rap et cetera.	Unsupervised
Dimensionsreduktion (Dimensionality Reduction)	Dimensionalitätsreduktion (DR) oder Dimensionsreduktion reduziert die Anzahl der betrachteten unabhägigen Variablen eines ML-Systems, indem man sich eine Reihe von Hauptvariablen auswählt. DR kann in Feature-Auswahl und Feature-Extraktion unterteilt werden. Bei Feature-Auswahl wird versucht, eine Teilmenge der ursprünglichen Variablen (auch als Features oder Attribute bezeichnet) zu finden. Feature-Extraktion transformiert die Daten im hochdimensionalen Raum in einen Raum mit weniger Dimensionen, in dem jede unabhängige Variable eine Raumdimension darstellt.	Unsupervised
Anomalie-Ermittlung (Anomaly Detection)	Es wird eine Reihe von Ereignissen oder Objekten durchsucht. Einige, die ungewöhnlich oder atypisch erscheinen, werden hervorgehoben. Ein Beispiel für eine Anomalieerkennungsaufgabe ist die Kreditkartenbetrugserkennung. Durch die Modellierung typischer Einkaufsgewohnheiten des Kunden kann ein Kreditkartenunternehmen den Missbrauch einer Karte erkennen und ein relevantes Konto sperren.	Unsupervised
Transkription (Transcription)	Es wird eine relativ unstrukturierte Datendarstellung irgendeiner Art beobachtet und dann in diskreter Textform beschrieben. Zum Beispiel wird bei einer optischen Zeichenerkennung dem Computerprogramm ein Bild eines Textes gezeigt mit der Aufgabe, dieses Bild in der Textform zurückzugeben. Google Street View verwendet Deep Learning, um Adressnummern auf diese Weise zu verarbeiten. Ein anderes Beispiel ist die Spracherkennung, bei der dem Computerprogramm eine Audiowellenform übergeben wird und dieses eine Zeichensequenz ausgibt, die in der Audioaufzeichnung gesprochene Wörter beschreibt. Deep Learning ist eine entscheidende Komponente moderner Spracherkennungssysteme.	Semi-Supervised
Maschinelle Übersetzung (Machine Translation)	Bei einer maschinellen Übersetzungsaufgabe besteht die Eingabe bereits aus einer Sequenz von Symbolen in einer Sprache, und das Computerprogramm muss dies in eine Sequenz von Symbolen in einer anderen Sprache umwandeln. Dies wird häufig bei natürlichen Sprachen angewendet, zum Beispiel beim Übersetzen vom Englischen ins Deutsche. Deep Learning hat in der jüngeren Vergangenzeit einen wichtigen Einfluss auf diese Art von Aufgaben.	Semi-Supervised
Strukturierte Ausgabe (Structured Output)	Als strukturierte Ausgabe kann jede Aufgabe gesehen werden, bei der der Ausgang einen Vektor (oder eine andere Datenstruktur, die mehrere Werte enthält) mit wichtigen Beziehungen zwischen den verschiedenen Elementen enthält. Dies ist eine breite Kategorie und subsumiert die oben beschriebenen Transkriptions- und Übersetzungsaufgaben sowie viele andere Aufgaben. Ein Beispiel ist das Mapping eines natürlichen Sprachsatzes in einen Baum, der seine grammatische Struktur beschreibt, indem Knoten der Bäume als Verben, Substantive, Adverbien und so weiter markiert werden. Beim Image Caption beschreibt das Computerprogramm ein Bild in natürlicher Sprache. Bei diesen Aufgaben werden mehrere Werte ausgegeben, die alle eng miteinander verknüpft sind. Zum Beispiel müssen die von einem Image-Caption-Programm erzeugten Wörter einen gültigen Satz bilden.	Semi-Supervised
Synthese und Sampling (Synthesis and Sampling)	Es werden neue Beispiele erzeugt, die denen in den Trainingsdaten ähnlich sind. Synthese und Sampling können nützlich bei den Medienanwendungen sein, wo manuelles Erzeugen großer Datenmengen häufig zu teuer, langweilig oder zeitaufwendig ist. Zum Beispiel können für Videospiele Texturen für große Objekte oder Landschaften automatisch erzeugt werden, statt dass ein Künstler jedes Pixel manuell zeichnet. Manchmal möchte man auf eine bestimmte Eingabe eine spezifische Art der Ausgabe generieren. Für eine Sprachsyntheseaufgabe stellt man beispielsweise einen geschriebenen Satz bereit und fordert eine Audiowellenform an.	Self-Supervised
Entscheidung / Empfehlung (Decision / Recommendation)	Dieser Aufgabentyp tritt sehr häufig in Computerspielen und Simulationen oder auch beim autonomen Fahren auf. Hier muss ein Agent (ein KI-Programm) eine Entscheidung treffen und eine bestimmte Aktion in seiner Umgebung ausführen, um seine Belohnung zu maximieren.	Reinforcement

pervised Learning) sind die Eingangsdatenelemente nicht mit einer relevanter Klasse/Label markiert. Diese Art des Lernens ist wichtig, weil es im menschlichen Gehirn wahrscheinlich viel häufiger vorkommt als das überwachte Lernen. Für die Klassifikation gehen wir davon aus, dass uns ein Trainingsdatensatz mit korrekt markierten (klassifizierten) Daten vorliegt. Leider hat man diesen Luxus nicht immer, wenn die Daten aus der realen Welt kommen – Videokameras, Mikros, jegliche Art Sensoren und so weiter. Das einzige Objekt im Bereich der Lernmodelle ist in diesem Fall die beobachtete Datenmenge, die oft als unabhängige Stichprobe einer unbekannten, zugrunde liegenden Wahrscheinlichkeitsverteilung angenommen wird. Obwohl die Datenelemente im Eingangsdatensatz nicht markiert sind, kann man immer noch das nötige Feature-Engineering durchführen und eine Gruppe von Objekten so gruppieren, dass die Objekte in der gleichen Gruppe (Cluster genannt) in gewissem Sinne einander ähnlicher sind (Bild 4) als denen der anderen Gruppen (Cluster).



Die zentralen ML-Aufgaben: Klassifikation, Regression, Clustering und Anomalie-Erkennung (Bild 4)

- **Semi-überwachtes Lernen** [12] (Semi-Supervised Learning, Semi-SL) ist eine spezifische Klasse des überwachten Lernens (Supervised Learning), die sowohl markierte als auch unmarkierte Eingangsdaten im Trainingsdatensatz enthält – normalerweise kleine Mengen markierter und eine große Menge nicht markierter Daten. Semi-SL ist also ein Mix aus Supervised- und Unsupervised Learning. ML-Forscher haben herausgefunden, dass unmarkierte Daten in Verbindung mit einer kleinen Menge markierter Daten im Trainingsdatensatz zu einer beträchtlichen Verbesserung der Lerngenauigkeit führen können. Da die Erfassung markierter Daten sehr oft ein ziemlich teures Unterfangen ist (zum Beispiel Kosten für einen menschlichen Experten und/oder Experimentkosten), kann man einen vollständigen Markierungsprozess aus Zeit- und/oder Kostengründen manchmal nicht durchführen, während die Erfassung nicht markierter Daten relativ kostengünstig ist (zum Beispiel die Video-/Bildaufnahmen einer Kamera oder das Speichern von Sensordaten). In solchen Fällen kann semi-überwachtes Lernen die einzige Möglichkeit sein, ein ML-Modell zu trainieren. Semi-überwachtes Lernen kann als Modell für menschliches Lernen betrachtet werden – da das menschliche Gehirn oft nur ein halbes Bild und halbe Information zur Kenntnis nimmt und versucht, daraus ein gesamtes Bild zu produzieren.
- **Selbstüberwachtes Lernen** (Self-Supervised Learning, Self-SL) ist eine spezifische Klasse des überwachten Lernens (Supervised Learning). Ähnlich wie Supervised Learning (SL) hat auch Self-SL mit Labeln markierte Datenelemente im Trainingsdatensatz, aber es sind keine von Menschen gemachte Labels vorhanden. Man kann also Self-SL als ein ML-System mit Supervised Learning ohne den Menschen in der Trainingsprozess-Schleife beschreiben. Nötige Markierungen werden von Eingangsdaten erzeugt – typischerweise mit einem heuristischen Algorithmus. Zum Beispiel sind die Autoencoder eine bekannte Instanz für das selbstüberwachte Lernen, bei denen die generierten Zieldaten den nicht modifizierten Eingangsdaten entsprechen sollten. Auf gleiche Art und Weise versuchen die Self-SL-Modelle, das nächste Bild in einem Video vorherzusagen, das nach der bekannten Bildreihe folgen wird, oder das nächste Wort in einem bereits eingegebenen Text. In diesen Fällen kommen die Supervise-Werte, also Werte, mit denen eine bereits gemachte Vorhersage abgeglichen werden kann, aus den zukünftigen Eingabedaten.
- **Bestärkendes Lernen** (Reinforcement Learning, RL) [13] als Zweig des maschinellen Lernens wurde lange vernachlässigt. Aber in der jüngsten Zeit wird es überraschend ins Rampenlicht gerückt. Erst recht, nachdem Google mit RL-Methodik dem Computer zuerst beigebracht hat, Atari-Spiele zu meistern, bis er dann schließlich im März 2016 mit dem AlphaGo-Programm den 18-maligen Go-Weltmeister Lee Sedol in fünf Spielen mit 4:1 besiegen konnte. Beim be-



Reinforcement Learning – bestärkendes Lernen (Bild 5)

stärkenden Lernen erhält ein Agent (siehe Bild 5) Informationen über seine Umgebung und lernt, verschiedene Aktionen mit der Berücksichtigung der Umgebungsrückmeldung immer wieder auszuführen, um eine Belohnung zu maximieren. Ein Beispiel für die Reinforcement-Learning-Methoden ist ein neuronales Netzwerk (siehe den Kasten KNN – Künstliches neuronales Netz), das auf den Bildschirm eines Videospiels schaut und die Spielaktionen mit dem Ziel maximaler Punktzahl ausführt. Momentan ist das Bestärkende Lernen ein Forschungsgebiet und hat (noch) keine großen Erfolge jenseits der Spieldomäne gefeiert. In der Zukunft erwartet man jedoch eine immer größere Palette realer Anwendungen mit Reinforcement-Learning-Komponenten – von selbstfahrenden Autos und selbstagierenden Robots bis zu Anwendungen für Ressourcenmanagement und Bildung.

Diese Artikelserie konzentriert sich auf das überwachte Lernen, weil es heute mit Abstand das dominanteste Paradigma des maschinellen Lernens im Allgemeinen und die Form von Deep Learning im Besonderen ist. Es gibt mittlerweile eine breite Palette von ML-Diensten, Werkzeugen und Industrieanwendungen, die genau auf den Ansätzen von Supervised Learning aufgebaut sind. Man muss aber auch sagen, dass die Grenzen zwischen überwachtem, semi-überwachtem und unüberwachtem Lernen fließend sind.

Algorithmen des maschinellen Lernens

Das Thema künstliche Intelligenz, maschinelles Lernen und Deep Learning hat in den letzten Jahren viel an Aufmerksamkeit und Bedeutung gewonnen und scheint sich neben Mobile, Cloud und IoT (Internet of Things) zu einer der nachhaltigen und zukunftsorientierten Mainstream-Technologien etabliert zu haben. Deep Learning ist schließlich auch das Hauptthema dieser Artikelserie. Es ist dennoch nicht die erfolgreichste Art des maschinellen Lernens. Man kann sogar mit ziemlich großer Sicherheit sagen, dass die meisten heutzutage angewandten ML-Algorithmen keine Deep-Learning-Algorithmen sind.

Als Beleg mag die Auswahlliste der ML-Algorithmen bei Azure Machine Learning [14] dienen: Der Neural-Network-Algorithmus – und Deep Learning basiert auf dem mathematischen Modell des mehrschichtigen künstlichen neuronalen Netzwerks [15] – ist einer von vielen. Deep Learning ist unter Umständen nicht der beste Algorithmus für die Lösung des gegebenen Problems. Manchmal liegt der Grund in den zu wenigen zur Verfügung stehender Trainingsdaten, was ziemlich oft ein entscheidendes Kriterium pro oder contra Deep Learning ist, oder es kann sich um ein spezifisches Problem handeln, das mit einem anderem ML-Algorithmus besser oder überhaupt erst zu lösen ist.

Hier schlägt wieder das Hammer-Nagel-Phänomen zu: Wenn man zum ersten Mal das maschinelle Lernen von Deep Learning aufbohrt und dieses zu nutzen und schätzen weiß, sieht jede Aufgabe wie ein Nagel aus, den man mit dem Hammer Deep Learning in die Wand klopfen kann. Um nicht in diese Falle zu geraten, muss man zumindest konzeptionell

KNN – Künstliches neuronales Netz

Auch bekannt als künstliches neuronales Netzwerk, KNN (Artificial Neural Network, ANN). Die KNNs sind auf dem biologischen Vorbild der vernetzen Neuronen–Zellen aufgebaut, wie es im Gehirn und im Rückenmark von höheren biologischen Organismen der Fall ist. Bei KNNs geht es jedoch mehr um eine Abstraktion, ein mathematisches Modell der vernetzten und gegebenfalls mehrstufigen Datenverarbeitung.

auch andere ML-Algorithmen kennenlernen und diese bei Bedarf wählen. Eine detaillierte Diskussion über alle aktuellen ML-Algorithmen würde den Artikel sprengen. Dennoch soll hier ein kurzer Überblick gegeben werden. Weitere Informationen zu ML-Algorithmen finden Sie beispielsweise im Artikel „How to choose algorithms for Microsoft Azure Machine Learning“ [14] – hier gibt es sehr gute Informationen zu ML-Algorithmen selbst und deren Auswahlkriterien – und in der Auswahlmappe [16].

Probabilistische Modellierung war eine frühere Art des maschinellen Lernens und wird heute noch oft verwendet. Der naive Bayes-Klassifikator [17] ist einer der bekannten Algorithmen aus dieser Reihe, der auf dem Bayes-Theorem [18] basiert und sehr effektiv unter anderem für die Spamfilter von E-Mails eingesetzt wird. Ein weiterer naheliegender Algorithmus ist die logistische Regression [19] (Logistic Regression, logreg), die manchmal als „Hello World“ des modernen maschinellen Lernens angesehen wird.

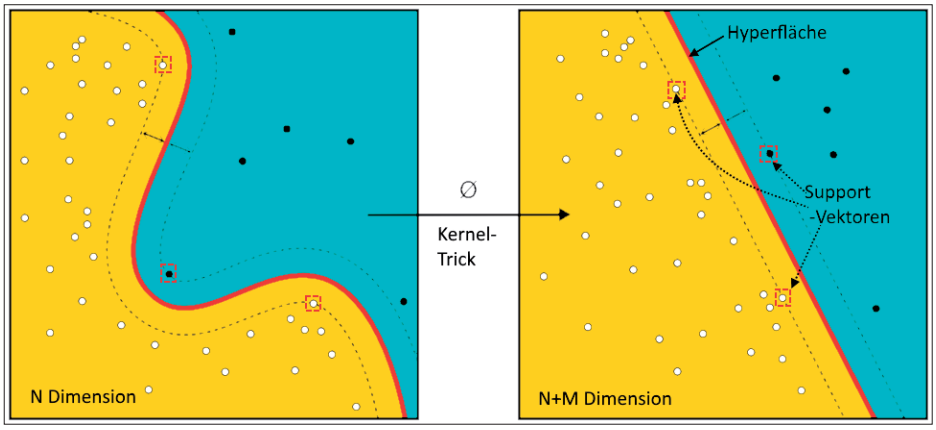
Man sollte sich nicht von dem Namen irreführen lassen: Beim Algorithmus der logistischen Regression geht es genau wie beim naiven Bayes um die ML-Aufgabe der Klassifikation (siehe Tabelle 1): Zuordnung eines Datenobjekts zu einer der vorgegebenen Kategorien. Ein weiteres Verwandtschaftsmerkmal für beide Algorithmen ist die Tatsache, dass sie schon vor der Erfindung des Computers manuell angewandt wurden und durch ihre Einfachheit und Effizienz heute noch erste Wahl für einen Datenwissenschaftler sind, der ein relevantes Klassifikationsproblem auswerten möchte.

Frühere künstliche neuronale Netzwerke sind mittlerweile durch ihre modernen Varianten (unter anderem auch Deep Learning) verdrängt. Und obwohl Deep Learning das Kernthema dieser Artikelserie ist und in deren nächster Folge ausführlicher dargestellt wird, ist es dennoch gut zu wissen, welche Wurzeln Deep Learning hat. Die Kernidee eines künstlichen Neurons als logisches Schwellenwertelement mit mehreren Eingängen und einem einzigen Ausgang wurde 1943 von McCulloch und Pitts in die Informatik eingeführt [20]. Im Jahr 1958 veröffentlichte Frank Rosenblatt dann das Modell des Perzeptrons (Wahrnehmer) [21], das bis heute die Grundlage künstlicher neuronaler Netze ausmacht.

Es gab aber auch einige Probleme damit. Damals zum Beispiel bei Lösung der XOR-Operation durch ein einfaches einlagiges Perzeptron, was vom KI-Pionier Marvin Minsky 1969 gezeigt wurde [22]. Aber ein Hauptproblem waren die ▶

Schwierigkeiten, große neuronale Netze effizient zu trainieren. Das führte zum KI-Winter, als die Forschungen und Investments im KI-Gebiet heruntergefahren wurden.

Dies änderte sich im Jahr 1986, als David Rumelhart, Geoffrey Hinton und Ronald Williams ihren Artikel im Journal „Nature“ [23] veröffentlichten, in dem sie den Backpropagation-Algorithmus [24] für neuronale Netze wiederentdeckten. Dabei handelt es sich um einen Weg, die parametrisierten Kettenoperationen mithilfe des Gradientenabstiegs-Algorithmus (Gradient Descent) zu trainieren. Man hatte angefangen, Backpropagation- und Gradientenabstiegs-Algorithmen für das Training der großen neuronale Netze anzuwenden. Damit konnte man bis dahin unlösbare Probleme bewältigen.



SVM – Support-Vector-Maschine (Bild 6)

Heute ist der Backpropagation-Algorithmus sozusagen ein Arbeitspferd des Lernens in künstlichen, neuronalen Netzwerken. Eine der ersten erfolgreichen praktischen Anwendungen hat Yann LeCun von Bell Labs präsentiert, als er Ideen von faltenden neuronalen Netzwerken (Convolutional Neural Network, CNN) [25] mit dem Backpropagation-Algorithmus kombinierte. Damit konnte ein Klassifikationsproblem von handgeschriebenen Ziffern gelöst werden. Ein solches neuronales Netzwerk, LeNet genannt, wurde von der US-Post in den 1990er Jahren für das automatisierte Lesen von Postleitzahlen auf Briefumschlägen verwendet.

Ohne Entscheidungsbaum (Decision Tree) wäre die Liste der modernen ML-Algorithmen nicht vollständig, insbesondere wenn es um strukturierte Daten geht, bei denen diese ML-Algorithmen sehr effektiv sind. Entscheidungsbäume sind hierarchische, flussdiagrammartige Strukturen, die die Eingangsdaten klassifizieren und/oder die Ausgabewerte vorhersagen lassen.

Dank der baumartigen Natur lassen sich die Entscheidungsbaum-Modelle leicht darstellen und interpretieren. Um die Klassifikationsgüte beim Einsatz von Entscheidungsbäumen zu steigern, war eine Weiterentwicklung der Entscheidungsbaumidee nötig. Entscheidungswälder (Decision Forest) handeln nicht mit einem einzelnen Baum, sondern mit vielen Bäumen.

Zum Beispiel ist der Random Forest ein robustes und schnelles Klassifikationsverfahren, das viele einzelne Entscheidungsbäume parallel aufbaut und auswertet und dann die Ergebnisse einsammelt. Dies zählt zu den sogenannten Ensemble-Techniken. Der Random-Forest-Algorithmus wird in verschiedenen Problembereichen verwendet. Man kann sogar behaupten, dass Random-Forest (fast) immer der zweitbeste Algorithmus für jegliche flache ML-Aufgabe ist.

Und er war sogar bis zum Jahr 2014 die erste Wahl. Dann aber hat der auf der Machine-Learning-Wettbewerb-Plattform Kaggle.com teilnehmende Algorithmus Gradient Boosting Machine [26][27][28] ihn vom Thron gestoßen. Die Gradient Boosting Machine ist – ähnlich dem Random-Forest-Algorithmus – eine iterative Verbesserung des Lernmodells, in der bei verbesserten Modellen im neuen Lernschritt die Schwachstellen der Vorgängermodelle aufgegriffen und ausgebessert werden. Die Gradient Boosting Machine ist einer der besten, wenn nicht der beste ML-Algorithmus, wenn es um strukturierte Daten geht. Zusammen mit Deep Learning steht er an der Spitze des Algorithmen-Berges des modernen maschinellen Lernens.

Kernel-Methoden [29] machen eine weitere Gruppe der Klassifikations-Algorithmen aus, die in den 1990er Jahren so populär und erfolgreich gewesen sind, dass sie die künstlichen neuronalen Netzwerke für eine weitere Dekade in den Schatten gestellt hatten.

Die bekannteste davon ist die Support-Vector-Maschine (SVM), die von Wladimir Vapnik und Alexey Chervonenkis 1963 eingeführt wurde. SVM löst ein Klassifikationsproblem durch die Ermittlung einer Entscheidungsgrenze (Decision Boundary) – eine Hyperfläche (siehe den Kasten Hyperfläche), die einen Vektorraum auf zwei Sätze aufteilt, die gesuchten Klassen (Kategorien) entsprechen.

Der SVM-Algorithmus geht in zwei Schritten vor:

- Im ersten Schritt werden die Daten so lange in einen (immer) höher dimensionierten Vektorraum überführt, bis die Entscheidungsgrenze zu einer Hyperfläche wird. Dann wird das Klassifikationsproblem linear, und folglich werden

Hyperfläche

Hyperfläche ist ein Raum mit einer um eine Stufe niedrigerer Dimensionalität wie der Vektorraum der untersuchten Daten. Die Hyperfläche stimmt nur im Fall eines 3D-Vektor-Datenraums mit der normalen zweidimensionalen Fläche überein. Für anders dimensionierte Vektorräume wird die Hyperfläche auch andere (um eins niedrigere) Dimensionen annehmen. Zum Beispiel wäre für einen 2D-Raum die Hyperfläche eine Linie (1D-Raum), für einen 4D-Raum wäre die Hyperfläche ein 3D-Objekt und so weiter.

die Zielklassen leicht linear trennbar sein.

- Im zweiten Schritt wird die Entscheidungsgrenze berechnet, indem man die Distanz zwischen den Punkten jeder Kategorie zu dieser Entscheidungsgrenze maximiert. Der entscheidende Kernel-Trick besteht darin, dass man nicht alle Datenpunkte (Vektoren) des neu geschaffenen höherdimensionalen Raumes berechnen muss, sondern letztendlich nur den Abstand der am äußersten liegenden Datenpunkte zur Hyperfläche der Entscheidungsgrenze (siehe Bild 6). Diese angrenzenden Datenpunkte werden Support-Vektoren (Support Vectors) genannt, was auch im Algorithmus-Namen zu erkennen ist.

SVMs bringen gute Ergebnisse bei relativ einfachen Klassifikationsproblemen wie zum Beispiel der Erkennung handgeschriebener Ziffern, der Text-/Hypertext-Klassifikation sowie zahlreichen Anwendungen in Chemie, Bio- und Geowissenschaften. SVM-Algorithmen stützen sich zudem auf eine umfangreiche Theorie, lassen ernsthafte mathematische Analysen zu, sind gut verständlich und leicht zu interpretieren. Dadurch waren SVMs in der ML-Szene lange Zeit sehr beliebt – und sind es immer noch.

Die SVMs sind jedoch bei großen Datenmengen schlecht skalierbar und liefern weniger gute Ergebnisse bei kognitiven Aufgaben wie etwa der Bildklassifizierung.

Weil SVM ein flacher Algorithmus ist, erfordert das im Fall einer kognitiven Aufgabe zuerst das manuelle Extrahieren nützlicher Darstellungen (ein Schritt namens Feature Engineering), was unter Umständen schwierig sein kann. Wie sich in den letzten paar Jahren herausgestellt hat, können die künstlichen neuronalen Netze mit darauf basierenden Deep-Learning-Algorithmen deutlich bessere Leistung bei kognitiven und einigen anderen KI-Aufgaben zeigen. Darüber werden Sie im nächsten Teil der Serie mehr erfahren. ■

[1] *Analytical Engine, www.dotnetpro.de/SL1809DeepLearning1*

[2] *Ada Lovelace, Notizen zu L. Menabrea's „Sketch of the Analytical Engine invented by Charles Babbage", www.dotnetpro.de/SL1809DeepLearning2*

[3] *Alan Turing, B. Jack Copeland (Editor), The Essential Turing. Clarendon Press, 2004*

[4] *Alan Turing, Computing Machinery and Intelligence, Mind-Journal, 1950, www.dotnetpro.de/SL1809DeepLearning3*

[5] *Turing-Test, www.dotnetpro.de/SL1809DeepLearning4*

[6] *John McCarthy, www.dotnetpro.de/SL1809DeepLearning5*

[7] *Expertensystem, www.dotnetpro.de/SL1809DeepLearning6*

[8] *Mykola Dobrochynskyy, Computer auf der Schulbank, dotnetpro 2/2016, S. 8 ff, www.dotnetpro.de/A1602DeepLearning*

[9] *Optimierungsproblem, www.dotnetpro.de/SL1809DeepLearning7*

[10] *Überwachtes Lernen, www.dotnetpro.de/SL1809DeepLearning8*

[11] *Unüberwachtes Lernen, www.dotnetpro.de/SL1809DeepLearning9*

[12] *Semi-überwachtes Lernen, www.dotnetpro.de/SL1809DeepLearning10*

[13] *Bestärkendes Lernen, www.dotnetpro.de/SL1809DeepLearning11*

[14] *How to choose algorithms for Microsoft Azure Machine Learning, Azure ML Studio Docs, www.dotnetpro.de/SL1809DeepLearning12*

[15] *Wikipedia, Künstliches neuronales Netz, www.dotnetpro.de/SL1809DeepLearning13*

[16] *Algorithmen-Auswahlmappe, Azure ML Studio Dokumentation, www.dotnetpro.de/SL1809DeepLearning14*

[17] *Naiver Bayes-Klassifikator, www.dotnetpro.de/SL1809DeepLearning15*

[18] *Satz von Bayes, www.dotnetpro.de/SL1809DeepLearning16*

[19] *Logistische Regression, www.dotnetpro.de/SL1809DeepLearning17*

[20] *Perzeptron, www.dotnetpro.de/SL1809DeepLearning18*

[21] *Frank Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychological Reviews 65 (1958), Seite 386–408, www.dotnetpro.de/SL1809DeepLearning19*

[22] *M. L. Minsky, S. A. Papert, Perceptrons, 2nd Edition, MIT-Press 1988, ISBN 978-0-262-63111-2, www.dotnetpro.de/SL1809DeepLearning20*

[23] *David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning representations by back-propagating errors, Nature-Journal, www.dotnetpro.de/SL1809DeepLearning21*

[24] *Backpropagation, www.dotnetpro.de/SL1809DeepLearning22*

[25] *Convolutional Neural Network, www.dotnetpro.de/SL1809DeepLearning23*

[26] *Jerome H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, www.dotnetpro.de/SL1809DeepLearning24*

[27] *XGBoost – Gradient Boosting Machine Library, www.dotnetpro.de/SL1809DeepLearning25*

[28] *Gradient Boosting, www.dotnetpro.de/SL1809DeepLearning26*

[29] *Kernel-Methoden, www.dotnetpro.de/SL1809DeepLearning27*

[30] *Thomas Mitchell, Machine Learning, McGraw-Hill, 1997, ISBN 978-0-07-042807-2*

Mykola Dobrochynskyy
ist Geschäftsführer von Software Factories. Seine Schwerpunkte sind Softwarearchitektur, modellgetriebene Softwareentwicklung, generative Programmierung sowie Cloud- und dienstorientierte Softwarearchitekturen.
ceo@soft-fact.de

dnpCode A1809DeepLearning