Neuronale Netze: Ein Blick in die Black Box

Originalartikel

Backup

<html> Nina Schaaf 14. Januar 2020<div class=„ce-textpic ce-left ce-intext“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>© Adobe: yingyaipumi</figcaption></figure> Künstliche Intelligenz und allem voran Deep Learning ist momentan in aller Munde. Hierbei dominiert größtenteils die Diskussion um die gesellschaftlichen Auswirkungen, welche meist zwischen Utopie und Horrorszenarien schwankt. In dieser teils sehr aufgeregten Debatte kann schnell untergehen, was eigentlich hinter den allgegenwärtigen Begriffen KI und Deep Learning steckt. Wie sind künstliche neuronale Netze aufgebaut und wie funktionieren sie? Darüber hinaus lohnt sich ein Blick auf eine der Herausforderungen neuronaler Netze: deren „Black-Box“-Charakter. Wieso sind neuronale Netze eine Black Box, für welche Anwendungen ist dies eher kritisch und welche Lösungsansätze existieren bereits? Denkt man an Anwendungen mit Künstlicher Intelligenz (KI), so kommen den meisten Menschen sicherlich Sprachassistenten, wie Alexa und Google Assistant, oder das autonome Fahren in den Sinn. In der öffentlichen Debatte um KI dominieren Meldungen dieser Domänen, seien es Fehlschläge (ein autonomes Fahrzeug von Uber tötet eine Fußgängerin [1]) oder Erfolgsnachrichten (Google Assistant bucht am Telefon einen Friseurtermin [2]). An einen Münzsortierautomaten, der KI einsetzt, hätten wohl die wenigsten gedacht. Doch auch hier können intelligente Systeme heute schon helfen, schwer beherrschbare Anforderungen wie Verarbeitung in Echtzeit, hohe Qualitätsansprüche, geringe Fehlertoleranzen und strikte gesetzliche Vorgaben zusammenzuführen. So verlangt eine EU-Richtlinie von Betreibern von Münzsortierautomaten, dass die Automaten die zugeführten Münzen zuverlässig nach Originalen, Fremdwährungen und Falschgeld unterscheiden können. Da Münzgeld jedoch, anders als Banknoten, keine spezifischen Sicherheitsmerkmale aufweist, müssen alternative und nichtsdestotrotz hochzuverlässige Prüfmethoden entwickelt werden. Maschinelle Lernverfahren (ML) wie künstliche neuronale Netze eignen sich hierfür sehr gut, da sie in der Lage sind, selbständig zu lernen. Anders als bei der Entwicklung klassischer Software, die nach vom Menschen explizit festgelegten Regeln arbeitet, können ML-Algorithmen die Regeln für das Lösen bestimmter Aufgaben selbst lernen und sind dadurch in der Lage, in großen, hochdimensionalen Datensätzen Zusammenhänge und Muster zu finden, die kein Mensch je entdecken würde. Voraussetzung hierfür ist ein ausreichend großer Pool an Beispieldaten für die drei zu differenzierenden Fälle Original, Fremdwährung und Fälschung. Diese Daten können dann dafür genutzt werden, ein neuronales Netz zu entwickeln. Während der sogenannten Trainingsphase lernt das

neuronale Netz selbständig Regeln, anhand derer die drei Klassen unterschieden werden können. Ergebnis des Trainingsprozesses ist ein neuronales Netz oder Modell, das in Münzsortierautomaten neu zugeführte Münzen sortieren kann. <div class=„ce-textpic ce-left ce-below“> Entgegen der weitläufigen Meinung sind neuronale Netze keine neue Erfindung, sondern haben eine lange Geschichte, die ihren Anfang schon in den 1940er Jahren hat. Dabei sind die frühen Lernalgorithmen, die wir heute kennen, inspiriert von der neurowissenschaftlichen Hypothese, dass die mentale Aktivität primär in Nervenzellen, den sogenannten Neuronen, stattfindet. Die Arbeitsweise künstlicher neuronaler Netze orientiert sich an den Vorgängen im menschlichen Gehirn, das aus schätzungsweise 86 Milliarden, über Dendriten verbundene Neuronen besteht (s. Abb. 1 links). Die Kommunikation zwischen Neuronen erfolgt mittels elektrischer Signale, die durch die Synapsen von einer Nervenzelle zur nächsten übertragen werden. Das Lernen im Gehirn erfolgt im Wesentlichen durch das Verstärken oder Abschwächen der Synapse [3]. Hierdurch wird die Intensität der elektrischen Signale und somit die Stärke der Verbindung zwischen Neuronen beeinflusst. Die an einem Neuron gleichzeitig ankommenden Signale addieren sich und das Neuron „feuert“, wenn die Summe einen Grenzwert überschreitet: Ein elektrischer Impuls schießt am Axon entlang. <div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>Abb. 1: Links: Nervenzelle (Neuron), Rechts: künstliches Neuron (Perzeptron). Quelle: Nina Schaaf</figcaption></figure> <div class=„ce-textpic ce-center ce-above ce-bodytext“> Die Grundeinheit eines künstlichen neuronalen Netzes ist ein einzelnes Neuron. Einfach ausgedrückt, ist dieses ein Element, das gewichtete Eingaben entgegen nimmt, verarbeitet und eine Ausgabe erzeugt. In Abb. 1 (rechts) ist ein einfaches mathematisches Modell eines Neurons, das sogenannte Perzeptron, dargestellt. Das Perzeptron ist über gewichtete Verbindungen – Kanten genannt – mit einem Satz von Eingaben verbunden, wobei das Gewicht der Stärke der Synapsen entspricht. Zusätzlich erhält das künstliche Neuron eine Dummy-Eingabe mit dem Wert 1 und einem eigenen Gewicht, welches als Bias, sprich Achsenabschnitt, bezeichnet wird. Die Eingaben, die das Perzeptron erhält, werden auch als Eingabemerkmale bezeichnet. Bezogen auf das Eingangsbeispiel zur Münzklassifikation könnten einzelne Münzen etwa durch Merkmale wie ihre Dicke und ihren Durchmesser beschrieben werden. Um eine Ausgabe zu erhalten, werden die gewichteten Eingaben sowie die Dummy-Eingabe aufsummiert und darauf eine Aktivierungsfunktion f angewandt. Im einfachsten Fall ist die Aktivierungsfunktion eine Schwellwertfunktion, die entweder 0 oder 1 ausgibt. <div class=„ce-textpic ce-right ce-intext“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><img class=„image-embed-item“ title=„Abb. 2: Schwellwertfunktion als Aktivierungsfunktion: überschreitet die gewichtete Summe des Neurons den Schwellwert y>=0, so ist die Ausgabe der Aktivierungsfunktion 1,

andernfalls ist sie 0. Quelle: Nina Schaaf“ alt=„Abb. 2: Schwellwertfunktion als Aktivierungsfunktion: überschreitet die gewichtete Summe des Neurons den Schwellwert y>=0, so ist die Ausgabe der Aktivierungsfunktion 1, andernfalls ist sie 0. Quelle: Nina Schaaf“ src=„https://www.informatik-aktuell.de/fileadmin/_processed_/9/9/csm_abb2_formale_schaaf_e6022c739d.png“ width=„350“ height=„121“ referrerpolicy=„no-referrer“ /><figcaption class=„image-caption“>Abb. 2: Schwellwertfunktion als Aktivierungsfunktion: überschreitet die gewichtete Summe des Neurons den Schwellwert y>=0, so ist die Ausgabe der Aktivierungsfunktion 1, andernfalls ist sie 0. Quelle: Nina Schaaf</figcaption></figure> Überschreitet die gewichtete Summe des Neurons den Schwellwert, so ist die Ausgabe der Aktivierungsfunktion 1 – das Neuron „feuert“. Mathematisch lässt sich dieses Verhalten wie in Abb. 2 dargestellt beschreiben. Ein Perzeptron kann eine mehrdimensionale lineare Funktion repräsentieren, wodurch sich beispielsweise lineare Klassifikationsprobleme lösen lassen. Hierbei trennt die lineare Funktion die gegebenen Daten derart in zwei Mengen auf, dass die Daten einer Menge zum Feuern des Perzeptrons führen, während die Daten der zweiten Menge das Perzeptron ruhen lassen. Im Falle der Münzklassifikation könnte ein solcher linearer Klassifikator zwischen originalen Euro-Münzen und Fälschungen unterscheiden. Allerdings sind die meisten realen Problemstellungen nicht durch lineare Funktionen abbildbar. Beispielhaft sei hier die Unterteilung von Münzen in die drei Klassen Original, Fremdwährung und Fälschung genannt. Zur Lösung nichtlinearer Fragestellungen können deshalb mehrere Neuronen miteinander vernetzt und dadurch ein künstliches neuronales Netz aufgebaut werden. Hierzu werden Neuronen in mehreren Schichten angeordnet und gerichtet miteinander verbunden, d. h. die Neuronen einer Schicht erhalten die Ausgaben der vorherigen Schicht als Eingabe. Die erste Schicht eines neuronalen Netzes wird als Eingabeschicht, die letzte Schicht als Ausgabeschicht bezeichnet. Die Menge der Neuronen in der Eingabeschicht wird durch die Anzahl der Eingabedaten bestimmt, die Menge der Neuronen in der Ausgabeschicht hängt von der gewünschten Anzahl an Ergebnissen ab. Dazwischen können beliebig viele verdeckte Schichten, bestehend aus beliebig vielen Neuronen, liegen. <div class=„ce-textpic ce-left ce-above“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>Abb. 3: Lernen durch Optimierung der Gewichte eines neuronalen Netzes. Quelle: Nina Schaaf</figcaption></figure> Das „Wissen“ eines Perzeptrons, also die Regeln, nach denen es bestimmte Aufgaben lösen kann, ist in den Gewichten gespeichert. Diese sind anfangs zufällig gewählt und müssen erst gelernt werden. Im Falle des Perzeptrons wird gelernt, indem die Gewichte des Perzeptrons derart angepasst werden, dass die gegebenen Daten möglichst fehlerfrei durch eine lineare Funktion getrennt werden. Der hierfür verwendete Perzeptron-Algorithmus passt die Gewichte nur dann an, wenn der Ausgabewert des Neurons vom Sollwert abweicht. Andernfalls bleiben die Gewichte unverändert. Da bei mehrschichtigen neuronalen Netzen nicht mehr direkt von der Eingabe auf die Ausgabe geschlossen werden kann, ist hierfür der Perzeptron-Lernalgorithmus nicht mehr anwendbar. Der Fehler, also der Unterschied zwischen Ausgabewert und Sollwert, kann nur für die Ausgabeschicht, nicht aber für die verdeckten Schichten gemessen werden. Die Lösung bietet der sogenannte Backpropagation-Algorithmus, der es erlaubt, den Fehler von der Ausgabeschicht durch das Netzwerk zurückzupropagieren – schematisch dargestellt in Abb. 3. Dem anfangs noch unkalibrierten Netz wird ein Trainingsdatum, bestehend aus Eingabemerkmalen und

zugehöriger Sollausgabe, zugeführt. Das Netz erstellt basierend auf den Eingabemerkmalen und den Gewichten eine Ausgabe y‘. Die Abweichung zwischen der Ausgabe des Netzes und der Sollausgabe y wird über eine Fehlerfunktion E gemessen. Der gemessene Fehler wird durch das gesamte Netz zurückgeführt, sodass darauf basierend die Gewichte des Netzes angepasst werden können. Diese Gewichtsanpassung kann als Optimierungsproblem verstanden werden, bei dem es gilt, E zu minimieren. <div class=„ce-textpic ce-right ce-intext“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>Abb. 4: Gradientenabstieg: Anpassung des Gewichtsvektors durch Hinabklettern der Fehlerfunktion. Quelle: Nina Schaaf, abgewandelt von [4].</figcaption></figure> Dieser Vorgang lässt sich am besten vor Augen führen, wenn man sich die Fehlerfunktion als eine Oberfläche mit Höhen und Tiefen vorstellt (s. Abb. 4), die sich über den kontinuierlichen Gewichtsraum spannt, d.h. den Raum, der alle möglichen Gewichtseinstellungen abdeckt. Um nun den Punkt des minimalen Fehlers zu finden – also die Einstellung an Gewichten, für die der Fehler minimal ist – muss man sich auf den tiefsten Punkt der Fläche zubewegen, diese also „hinab klettern“. Hierfür wird der Gradient der Fehlerfunktion benötigt, welcher in die Richtung des größten Anstiegs des Fehlers zeigt. Soll also die Fehlerfunktion minimiert werden, so folgt man schrittweise der Richtung des negativen Gradienten, der Richtung des größten Abstiegs. Nach jedem Schritt werden die Gewichte angepasst und der Gradient wird unter Berücksichtigung der neuen Gewichte wiederholt evaluiert. Dieser Vorgang wird so lange wiederholt, bis eine Konvergenz mit dem kleinstmöglichen Fehler erreicht wird. Diese schrittweise Anpassung des Gewichtsvektors wird als Gradientenabstieg bezeichnet. <div class=„ce-textpic ce-center ce-above ce-bodytext“> Anders als man denken könnte, verweist das Wort „Deep“ nicht etwa auf ein besonders tiefes Problem- oder Lösungsverständnis, das mit Deep Learning erreicht werden kann. Tatsächlich bezieht sich die Tiefe auf die Struktur der verwendeten neuronalen Netze, konkret die Verwendung vieler verdeckter Schichten. Je tiefer das Netz, d. h. je mehr verdeckte Schichten es hat, umso komplexer ist die extrahierte Datenrepräsentation. Wie viele Schichten und Neuronen für eine bestimmte Problemstellung erforderlich sind, lässt sich jedoch nicht pauschal definieren. Wählt man beispielsweise zu wenige Neuronen, erhält man ein zu einfaches Modell, das die Daten nur unvollständig repräsentieren kann. Bei zu vielen Neuronen und Schichten hingegen „merkt“ sich das Netz alle Daten, die es während des Trainings gesehen hat, was einem Auswendiglernen gleichkommt. Dieses als Overfitting bezeichnete Phänomen hat den Effekt, dass das Netz schlecht generalisiert, also bei unbekannten Daten schlechte Ergebnisse liefert. Tatsächlich ist das Design der verdeckten Schichten immer noch ein äußerst aktives Forschungsgebiet – es existieren noch keine universellen theoretischen Richtlinien, denen man folgen könnte [5]. In der Praxis wird beispielsweise Kreuzvalidierung zur Bewertung geeigneter Architekturen eingesetzt, d. h. es wird mit verschiedenen Strukturen

experimentiert und diejenige gewählt, welche am besten geeignet ist [6]. Eine weitere gängige Methode ist, bereits bekannten Architekturen anderer Anwendungen zu folgen und deren Struktur zu übertragen und an einen individuellen Anwendungsfall anzupassen. <h2>Algorithmische Komplexität – Fluch und Segen</h2> Einer der größten Vorteile des Deep Learning – die enorme Komplexität neuronaler Netze und damit verbunden auch ihre Fähigkeit zur Approximation beliebig komplexer Funktionen – ist gleichzeitig auch einer der größten Kritikpunkte. Man sagt auch, tiefe neuronale Netze seien eine Black Box, da die von ihnen gelernten Zusammenhänge und Datenrepräsentationen so komplex und abstrakt sind, dass Menschen – und selbst Experten – sie nicht mehr nachvollziehen können. Begründet ist dies darin, dass Black-Box-Modelle nicht transparent sind. Transparenz kann grundsätzlich auf drei unterschiedliche Ebenen bezogen sein: auf Modell-Ebene (Simulierbarkeit), auf Komponenten-Ebene (Unterteilbarkeit) sowie auf algorithmischer Ebene (algorithmische Transparenz) [7]. Simulierbarkeit ist dann gegeben, wenn ein Mensch alle Rechenschritte des Modells in angemessener Zeit auswerten kann. Diese Anforderung ist bei einem tiefen neuronalen Netz, das unter Umständen Millionen von Gewichten hat, unmöglich zu erfüllen. Um dem Kriterium der Unterteilbarkeit zu entsprechen, müssten alle Komponenten des neuronalen Netzes intuitiv verständlich sein, also beispielsweise der Einfluss eines jeden Parameters nachvollziehbar sein. Diese Voraussetzung ist für neuronale Netze durch die schiere Anzahl an Parametern ebenfalls nicht gegeben. Zudem besitzen tiefe neuronale Netze in der Regel hochkomplexe Entscheidungsgrenzen, was sie folglich auch für die Transparenz auf algorithmischer Ebene disqualifiziert. Eine Frage, die immer wieder aufkommt, ist, warum wir neuronale Netze überhaupt verstehen wollen. Müssen wir wissen, wie ein Netz entscheidet oder genügt es, den Black-Box-Charakter zu akzeptieren, solange neuronale Netze eine möglichst hohe Vorhersagegenauigkeit haben? Oftmals wird hier der Vergleich zur menschlichen Intuition gezogen. Viele Entscheidungen, die wir tagtäglich treffen, erfolgen „aus dem Bauch heraus“, wir können also die genauen Beweggründe einer Entscheidung oftmals nicht solide begründen. Sollten wir also diesen Umstand auch für automatisiert getroffene Entscheidungen akzeptieren? Um diese Fragen zu beantworten, hilft es, einige Beispiele zu betrachten, bei denen die Erklärbarkeit neuronaler Netze zumindest wichtig, wenn nicht unerlässlich ist. Eine häufige Anwendung in der Produktion ist die Erkennung von Ausschuss. Dies können beispielsweise Werkstücke sein, die bestimmte Qualitätsanforderungen nicht erfüllen oder, wie eingangs beschrieben, das Aussortieren unerwünschter Objekte wie Falschgeld. Werden für diese Anwendungsfälle neuronale Netze eingesetzt, kann es vorkommen, dass das Netz andere Entscheidungen trifft als beispielsweise ein Mitarbeiter in der Qualitätssicherung. Entdeckt man eine solche Diskrepanz, kann es helfen, die Kriterien des neuronalen Netzes für die spezifische Entscheidung zu hinterfragen, um eventuelle Fehler im Modell aufdecken zu können. <div class=„ce-textpic ce-left ce-above“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>Abb. 5: Patient mit Lungenentzündung. Links:

Röntgenaufnahme, Rechts: Heatmap des neuronalen Netzes. Quelle: [11]</figcaption></figure> Besonders relevant ist dieser Aspekt auch für medizinische Anwendungen. Hier kann der Einsatz von KI Ärzte durch hochpräzise Vorhersagen bei der Diagnose von Krankheiten unterstützen und gleichzeitig helfen, menschlichen Fehlern vorzubeugen. Eine Anwendung von tiefen neuronalen Netzen in der Medizin ist die Analyse von Thoraxröntgenbildern, um Lungenentzündungen zu erkennen. Mithilfe von KI könnte eine Diagnose binnen 10 Sekunden und nicht wie bisher nach rund 20 Minuten erstellt werden [8]. Eine Gruppe von Forschern hat für ein solches Netz geprüft, aufgrund welcher Bildbereiche das Netz ein Risiko für Lungenentzündungen ausgibt [9]. Betrachtet man das in Abb. 5 links dargestellte Röntgenbild, so indiziert hier der Bereich in der oberen rechten Lunge eine Lungenentzündung. Untersucht man aber nun, welche Bildbereiche für das Netz wirklich relevant waren (Abb. 5, rechts), so sieht man, dass dem ursächlichen Bereich hier sogar ein negativer Einfluss zugewiesen wurde – erkennbar an den negativen Werten im linken oberen Bereich des Bildes. Erstaunlicherweise werden dem Schriftzug „Portable“ und den Seitenmarkierungen in der oberen rechten Ecke, die dem verwendeten mobilen Scanner zugeordnet sind, ein positiver Einfluss zugewiesen. Es haben also nicht die tatsächlich ausschlaggebenden Bildbereiche zur Risiko-Attestierung „Lungenentzündung“ beigetragen, sondern die Indikatoren für die Verwendung eines mobilen Röntgengerätes, welches in der Regel eingesetzt wird, wenn der Patient zu krank ist, um aus dem Bett zu kommen. Das neuronale Netz könnte also gelernt haben, einen Zusammenhang zwischen mobilen Röntgenaufnahmen und einem erhöhten Risiko für eine Lungenentzündung herzustellen (Korrelation), obwohl die mobile Aufnahme nicht die eigentliche Ursache der Lungenentzündung ist (Kausalität). Sollen deep-learning-basierte Anwendungen tatsächlich für die computergestützte Diagnose eingesetzt werden, so zeigt dieses Beispiel, dass eine genaue Überprüfung der Entscheidungsweise oder zumindest eine intensive Überprüfung des Modells unvermeidlich ist. Auch bei sicherheitskritischen Anwendungen, in denen neuronale Netze eingesetzt werden, ist das Verständnis der generellen inneren Abläufe dieser Netze unerlässlich. Kommt etwa ein neuronales Netz bei der Programmierung eines Roboters zum Einsatz, so muss dieses System zur Abnahme einer Sicherheitsprüfung unterzogen werden. Bei dieser muss die gesamte Funktionsweise des Systems ersichtlich und nachvollziehbar sein, um eine regelkonforme Arbeitsweise des Roboters garantieren zu können. Besonders relevant ist dieser Aspekt, wenn Roboter mit Menschen zusammenarbeiten sollen. Des Weiteren werden neuronale Netze aktuell bereits auch für die Anwendung von bisher regelbasiert arbeitenden Systemen wie Steuergeräten erprobt. Wie im vorherigen Fall ist auch hier die Sicherheitsprüfung dieser Systeme, etwa durch den TÜV, eine Hürde auf dem Weg, diese Verfahren in den praktischen Einsatz zu überführen. Es müssen also Verfahren gefunden werden, die die inneren Entscheidungswege neuronaler Netze erklären oder zumindest deren korrekte Funktionsweise verifizieren können. <h2>Blick in die Black Box: Erklärung neuronaler Netze</h2> Eine Möglichkeit, die generelle Arbeitsweise neuronaler Netze zu erklären, ist die Abbildung des Netzes mittels inhärent interpretierbarer Modelle, sogenannter White-Box-Modelle, wie Entscheidungsbäumen oder Entscheidungsregeln. Entscheidungsbäume bestehen aus internen Knoten, die zu überprüfende Bedingungen definieren, und Blattknoten, die Klassen darstellen. Möchte man ein Datum mithilfe eines Entscheidungsbaums klassifizieren, wird der Baum von oben nach unten traversiert, bis

man einen Blattknoten erreicht, der die Klasse codiert. Um ein Black-Box-Modell nun mithilfe eines solchen White-Box-Modells erklären zu können, wird im ersten Schritt das Black-Box-Modell, etwa ein neuronales Netz, gelernt (s. Abb. 6). Im Anschluss kann ein interpretierbares Stellvertretermodell – auch als Surrogat bezeichnet – aus dem neuronalen Netz extrahiert werden, welches dann zur Erzeugung von Erklärungen genutzt werden kann. <div class=„ce-textpic ce-left ce-above“><div class=„ce-gallery ce-row ce-column“ data-ce-columns=„1“ data-ce-images=„1“><figure class=„image“><figcaption class=„image-caption“>Abb. 6: Von der Black Box zur Erklärung: Extraktion von White Box-Modellen aus Black Box-Modellen. Quelle: [9].</figcaption></figure> Konkret funktioniert die Extraktion eines Entscheidungsbaums aus einem neuronalen Netz wie in Abb. 6 links dargestellt. Zuerst wird unter Zuhilfenahme des neuronalen Netzes für alle Eingabedaten, bestehend aus Merkmalen und einer Sollausgabe, eine Vorhersage berechnet. Im Anschluss werden die Eingabemerkmale zusammen mit der Vorhersage des neuronalen Netzes dazu verwendet, einen Entscheidungsbaum zu trainieren. Oftmals besitzen die auf diese Weise extrahierten Entscheidungsbäume eine relativ geringe Vorhersagegenauigkeit und Wiedergabetreue, d. h. die Entscheidungen des Entscheidungsbaums stimmen nicht mit denen des neuronalen Netzes überein. Eine Möglichkeit, dieser Herausforderung zu begegnen, ist, die Struktur des neuronalen Netzes so zu beeinflussen, dass aus dem Netz extrahierte Entscheidungsbäume ein aussagekräftiges Erklärmodell liefern. Dieses Ziel kann durch eine Anpassung der Lernregel, konkret durch den Einsatz von Regularisierung, erreicht werden. Hierbei wird der Fehlerfunktion (vgl. Abb. 2), welche den Fehler zwischen Sollausgabe und Ausgabe des Netzes beschreibt, eine weitere Bedingung, der sogenannte Regularisierungsterm, hinzu addiert. Dieser bestraft bestimmte Eigenschaften der Gewichte, sodass diese während des Trainings „gezwungen“ werden, sich nach dem gewünschten Verhalten auszurichten. Üblicherweise wird Regularisierung dazu verwendet, das zuvor beschriebene Overfitting zu vermeiden. Es existieren jedoch bereits Ansätze, die Regularisierung einsetzen, um die Struktur von neuronalen Netzen so zu beeinflussen, dass extrahierte Surrogate bestimmten Kriterien (z. B. Größe, Zuverlässigkeit etc.) genügen. Der Tree Regularizer [10] schätzt die Größe des Entscheidungsbaums, während der L1-orthogonal Regularisierer in [11] neuronale Netze mit wenigen, aber orthogonalen Gewichten favorisiert. Beides führt zu neuronalen Netzen mit Entscheidungsgrenzen, die von Entscheidungsbäumen gut angenähert werden können. Die Bäume sind gleichzeitig von geringer Größe, wodurch sie für den Menschen leicht verständlich sind. <h2>Blick in die Glaskugel: wohin geht die Reise?</h2> Schon heute existiert eine Vielzahl an Methoden und Techniken, die uns Menschen erlauben, einen ersten Blick in die Black Box zu werfen. Doch sind wir noch lange nicht am Ziel angekommen. Stand heute gibt es bereits eine Vielzahl an Unternehmen, die die enormen Potenziale der künstlichen Intelligenz nutzt, sei es in hochrelevanten Anwendungen, wie der computergestützen Diagnose im Gesundheitswesen oder dem autonomen Fahren, aber auch zur anfangs erwähnten Bekämpfung des Falschgeldgebrauchs. Auf der anderen Seite stehen viele Firmen hier

aufgrund bisher nicht erkannter Potenziale, fehlenden Know-hows oder gesetzlicher Hürden erst am Anfang. Doch mit steigender Verbreitung der KI-Anwendungen wird auch der Bedarf von Erklärbarkeit der eingesetzten Algorithmen steigen. Hier gilt es, bereits existierende Verfahren in die Anwendung zu überführen und neue Methoden zu entwickeln. Auch die Frage, welche Art der Erklärung für welche Zielgruppe am besten geeignet ist, wird hierbei noch eine große Rolle spielen. Schließlich hat ein Endanwender ein anderes Interesse an einem KI-Verfahren als der Experte, der es programmiert hat und selbst besser verstehen will.</p> </html>