Algorithmen, maschinelles Lernen und die Grenzen der künstlichen Intelligenz

Sebastian Höfer

(Preprint, finale Version erscheint in Jusletter 26. November 2018)

Unser tägliches Leben wird immer stärker durch computergesteuerte Entscheidungen beeinflusst. Dieser Artikel erklärt auf einfache und verständliche Weise die Schlüsseltechnologien, die diesen computergesteuerten Entscheidungen zu Grunde liegen: Algorithmen und maschinelles Lernen. Gegenstand des Artikels sind die Fragen: 1) Was sind Algorithmen und wie funktionieren sie? 2) Wie können Verfahren des maschinellen Lernens, eines Teilgebiets der künstlichen Intelligenz, selbstständig Algorithmen lernen? 3) Was sind die Grenzen des aktuellen Forschungsstandes des maschinellen Lernens?

Einleitung

Wenn ich morgens aufwache, steht meist mein 4jähriger Sohn vor mir und fordert mit einem breiten Grinsen vehement sein Frühstück ein. Da Widerstand in diesem Fall völlig zwecklos wäre, begebe ich mich zügig in die Küche und bereite ihm sein Frühstück zu; wenn vor dem Kindergarten noch genug Zeit ist, oftmals ein paar Waffeln. Um diese zuzubereiten, befolge ich immer das gleiche Rezept: Ich vermische zuerst Zucker, Eier und Margarine, füge daraufhin Mehl, Milch und Backpulver hinzu, vermische das Ganze, und gieße schließlich mit einer Schöpfkelle kleine Mengen dieser Masse in ein vorgeheiztes Waffeleisen.

Warum erzähle ich Ihnen das? Das Rezept bzw. die Handlungsanweisung, die ich morgens zum Waffelnmachen befolge, ist ein Beispiel eines *Algorithmus*: eine eindeutige Abfolge elementarer Schritte, die zur Erfüllung einer Aufgabe abgearbeitet werden müssen. In diesem Artikel möchte ich Ihnen näherbringen, was es bedeutet, Computer mit Algorithmen zu füttern, und vor allem wie es möglich ist, dass Computer *Algorithmen selbständig erlernen*. Gegenstand dieses Artikels sind die Fragen, 1) was Algorithmen sind und wie sie funktionieren, 2) wie Verfahren des *maschinellen Lernens,* eines Teilgebiets der *künstlichen Intelligenz,* selbstständig Algorithmen lernen können, 3) und was die Grenzen des aktuellen Forschungsstandes des maschinellen Lernens sind.

Algorithmen

Widmen wir uns der Frage, wie ein Algorithmus aufgebaut sein muss, damit wir einen Computer damit füttern können. Ich könnte beispielsweise versuchen einen Algorithmus zu entwerfen, der einem Computer erlaubt Waffeln zu machen. Leider würde ich dabei jedoch einer Mammutaufgabe gegenüberstehen: Der Computer weiß nicht nur nicht, was Waffeln

sind - er weiß auch nicht, was Margarine ist, wie ein Waffeleisen aussieht, geschweige denn, wie man es anschaltet oder eine Schöpfkelle mit Waffelteig befüllt. Um nämlich Computer mit einem Algorithmus - auch *Code* oder *Programm* genannt - zu füttern, muss der Algorithmus in einer sehr primitiven und elementaren Sprache verfasst werden, welche der Computer auch versteht: einer *Programmiersprache*. Zwar helfen sogenannte *Softwarebibliotheken* dem fleißigen Programmierer weiter, indem sie den Wortschatz des Computer erweitern. Leider ist der Wortschatz zum Zubereiten von Waffeln jedoch noch in keiner Softwarebibliothek abgelegt, sondern Gegenstand aktueller Forschung[1].

Daher möchte ich mich aus pädagogischen Gründen einem Beispiel aus einer einfacheren Domäne widmen, nämlich den Brett- und Gesellschaftsspielen. Spiele wie Schach oder Go werden oft als eine der Kernkompetenzen menschlicher Intelligenz erachtet. Nicht zuletzt aus diesem Grund haben die Nachrichten, dass IBM 1997 mit Deep Blue[2] und 2016 Google DeepMind mit AlphaGo[3] Computerprogramme entwickeln konnten, welche die besten menschlichen Spieler im Schach und Go eindeutig besiegen konnten, viel Aufsehen erregt. Daher möchte ich im Folgenden anhand von Brettspielen die Kernideen von Algorithmen und maschinellem Lernen erläutern. Da allerdings sowohl Schach als auch Go sehr komplexe Spiele sind (und ich leider weder Schach noch Go besonders gut beherrsche), sich dieselben Prinzipien aber zum Glück auch leicht anhand einfacherer Spiele erklären lassen, möchte ich mich hier dem *TicTacToe* widmen. Wie in Abbildung 1A illustriert, besteht dieses Spiel aus einem 3x3 großen Spielfeld, auf welches zwei Spieler abwechselnd einen Kreis oder ein Kreuz setzen. Der Spieler, welcher zuerst drei seiner Symbole in einer Reihe vertikal, horizontal oder diagonal platziert hat, gewinnt. Lassen Sie mich zu Anschauungszwecken einen Algorithmus entwickeln, der eine - zugegebenermaßen sehr primitive - Taktik zum Spielen von Tic-Tac-Toe implementiert:

Sebastians TicTacToe-Algorithmus Version 1.0

```
1 Für jede Spalte des Spielbrettes:
2       Für jede Zeile des Spielbrettes:
3             Wenn das Feld an (Spalte, Zeile) leer ist
4                   Dann
5                         Setze Kreuz auf Feld an (Spalte, Zeile)
6                         übergib Kontrolle an Gegenspieler
7             Sonst
8                   Gehe zu nächstem Feld
```

[1] Antonia Schaefer: Projekt „RoboHow": Roboter lernen kochen. 13. März 2017 (http://www.spiegel.de/wissenschaft/technik/projekt-robohow-roboter-lernen-kochen-a-1138488.html, alle Websites zuletzt besucht am 15. Oktober 2019).

[2] Murray Campbella, Joseph Hoane Jr., Feng-hsiung Hsu. Deep Blue. Artificial intelligence, 2002.

[3] David Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016.

Dieser Algorithmus enthält mehrere wichtige Bestandteile, die fast jeder programmierte Algorithmus auf der Welt aufweist:

- Er enthält *Schleifen* („**Für jede**"), die Teile des Programms wiederholen und dabei einen vorgegebene Wertbereich durchlaufen. Man nennt das in der Fachsprache *iterieren*. Mein Beispielalgorithmus wiederholt die Programmzeilen 3-8 jeweils einmal für jede Kombination von Spalten und Zeilen des Spielfelds. In Fachsprache: der Unteralgorithmus in Programmzeilen 3-8 iteriert über jedes Feld des Spielbretts.
- Er enthält eine bedingte Anweisung („**Wenn**", … „**Dann**", … „**Sonst**"), die den Fluss des Algorithmus beeinflusst. In diesem Fall wird nur dann ein Kreuz gesetzt, wenn das Feld, über welches der Algorithmus iteriert, leer ist;
- und schließlich enthält er Aufrufe von *Funktionen*, welche Unterprozeduren anstoßen und ausführen, z. B. „*male Kreuz*".

Welche Taktik setzt dieser Algorithmus um? Stellen Sie sich vor, Sie würden gegen meinen Algorithmus spielen, wie würde das Spiel verlaufen? Wie könnten Sie ihn besiegen? Nehmen Sie sich bitte einen Moment Zeit, um darüber nachzudenken.

In der Tat ist der Algorithmus relativ primitiv: Bei jedem Aufruf durchkämmt er das Spielfeld von links oben nach rechts unten, erst die Spalten und dann die Zeilen ablaufend, und setzt ein Kreuz auf das erste freie Feld. Daher ist der Algorithmus leicht zu schlagen: Um gegen meinen Algorithmus zu gewinnen, könnten Sie einfach Ihre Kreise in der mittleren vertikalen Spalte platzieren, oben beginnend.

Intuitiv werden Sie wahrscheinlich bereits ein paar Ideen haben, mit welcher Taktik Sie spielen würden. Können Sie diese in eine algorithmische Form bringen? Versuchen Sie ruhig mal Ihren Intuitionen mit Hilfe von Wenn-Dann-Verzweigungen und Schleifen Ausdruck zu verleihen.

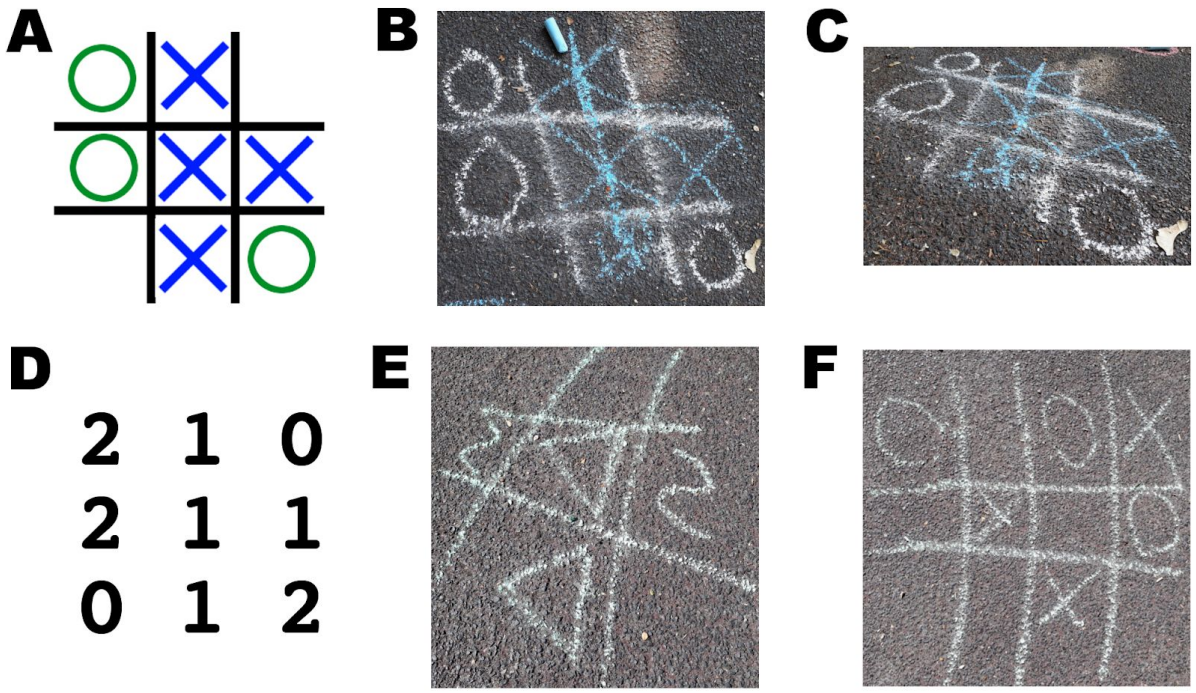

Abbildung 1: Verschiedene Formen von TicTacToe. (Ausführliche Erklärungen im Verlauf des Textes.)
(A) Eine TicTacToe-Partie, in der Spieler 1 (Kreuz) gewonnen hat. (B) Dieselbe Partie wie in A, aber in einem echten Umfeld: mit Kreide. (C) Dieselbe Partie wie in B, aber aus einer anderen Perspektive. (D) Damit eine Maschinelles-Lernen-Methode TicTacToe lernen kann, muss das Spielfeld in maschinenlesbarer Form repräsentiert werden, d. h. durch Zahlen. (E) Das Spiel bleibt im Prinzip das gleiche, selbst wenn die Symbole verändert werden. (F) Eine Variante, in der es eine zusätzliche Spalte gibt.

Maschinelles Lernen: Lernende Algorithmen

Wenn es Ihnen geht wie mir, werden Sie nach einiger Überlegung feststellen, dass eine optimale Taktik nur mit Hilfe von Wenn-Dann-Regeln und Schleifen gar nicht so einfach hinzuschreiben ist. Es passiert leicht, dass der vermeintlich so gut taktierende Algorithmus Schwächen aufweist, die man übersehen hat, da man an bestimmte Sonderfälle nicht gedacht hat. Und wir sollten uns noch einmal vor Augen führen, dass wir hier nur TicTacToe und nicht Schach oder Go spielen!

Auf irgendeine Weise sind Menschen im Stande, gute Taktiken für derartige Spiele - und für allerhand andere Aufgaben - zu erlernen und auszuüben. Würden wir von einem intelligenten Computer nicht auch erwarten, dass er selbständig den besten Algorithmus erlernen kann um Tic-Tac-Toe und andere Aufgaben zu bewältigen? Genau diese Fähigkeit verspricht das Forschungsfeld des maschinellen Lernens - ein wichtiges Teilgebiet der künstlichen Intelligenz.

Maschinelles TicTacToe lernen

Lassen Sie uns gemeinsam erarbeiten, auf welche Art und Weise das am weitesten verbreitete Paradigma des maschinellen Lernens angewandt werden kann, um TicTacToe zu erlernen. Dieses Paradigma nennt sich *Supervised Learning* (*überwachtes Lernen*). Die Grundidee des Supervised Learning ist folgende: Wir sammeln eine gewisse Menge an *Trainingsdaten*, die aus verschiedenen Spielzügen bestehen. Ein TicTacToe-Experte bewertet dann jeden Zug hinsichtlich seiner Güte, im einfachsten Fall als *gut* oder *schlecht.* Mit diesen Trainingsdaten füttern wir dann eine *Methode des maschinellen Lernens* (abgekürzt *ML-Methode*), welche daraus lernt gute von schlechten Zügen zu unterscheiden, und so eine zielführende Strategie erlernt.

Auch wenn diese Vorgehensweise im Prinzip relativ einfach erscheint, ergeben sich in der konkreten Umsetzung einige Fragen. Die wichtigsten sind: Was ist eine ML-Methode eigentlich und wie funktioniert sie? Nun, verblüffenderweise ist eine ML-Methode selbst wieder nichts anderes als ein Algorithmus! Genauer gesagt ist eine ML-Methode ein *Meta-Algorithmus*, der zum Ziel hat neue Algorithmen aus Trainingsdaten zu erlernen. Es gibt viele verschiedene ML-Methoden, die auf verschiedene Anwendungsgebiete und Problemstellungen zugeschnitten sind und nach jeweils leicht anderen Prinzipien funktionieren. Beispiele für verbreitete Methoden des Supervised Learning sind *lineare Regression*, *Entscheidungsbäume* oder *künstliche neuronale Netze*. Alle funktionieren jedoch nach dem gleichen Prinzip: Sie konsumieren Trainingsdaten, um daraus einen neuen Algorithmus zu lernen. Ich werde mich im Folgenden auf die *künstlichen neuronale Netze* beschränken, da diese momentan eine der beliebtesten Methoden des Supervised Learning sind.

Überwachtes Lernen

Bevor wir allerdings in die Details von künstlichen neuronalen Netzen gehen, stellt sich die Frage, wie die Anwendung von maschinellem Lernen bzw. Supervised Learning für TicTacToe konkret aussehen würde. Wie zuvor erwähnt müssen zunächst Trainingsdaten gesammelt werden. Dazu setzen sich Experten aus dem Anwendungsgebiet zusammen und annotieren (in ML-Sprech: *labeln*) eine Menge an Trainingsbeispielen. Diese Trainingsbeispiele liegen beim Supervised Learning immer in Form von Paaren vor: Eine *Eingabe*, also in unserem Fall eine Spielstellung und ein Zug, und eine *Bewertung* des Zuges (gut oder schlecht), auch *Label* genannt. Abbildung 2 zeigt zwei Trainingsbeispiele für TicTacToe, einen guten und einen schlechten Zug. Es ist von großer Wichtigkeit, dass die Trainingsdaten repräsentativ für das Problem sind, d.h. dass sie möglichst viele Facetten des Problems abbilden. Wir werden in Kürze auf die Frage zurückkommen, wie viele Trainingsdaten eigentlich benötigt werden und wie diese beschaffen sein müssen.

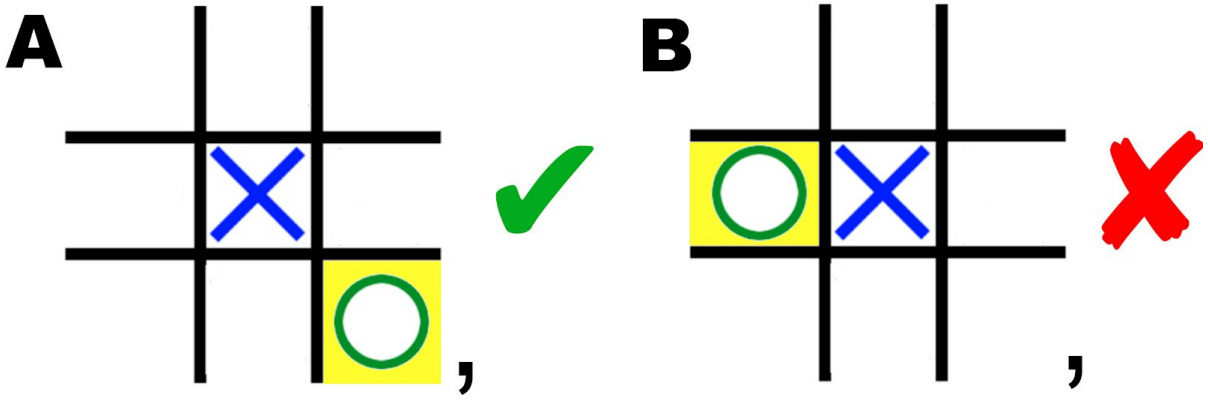

Abbildung 2A-2B: Ein positives und ein negatives Trainingsbeispiel für unseren TicTacToe-Lerner, aus Sicht von Spieler 2 (Kreis). In beiden Fällen hat Spieler 1 sein Kreuz in die Mitte gesetzt. In diesem Fall muss Spieler 2 sein Symbol in eine Ecke setzen (A). Setzt Spieler 2 sein Symbol nämlich daneben (B), verliert er - vorausgesetzt Spieler 1 spielt optimal. Das liegt daran, dass ein Symbol in einer Ecke mehr potentielle Dreierreihen blockiert (nämlich drei: horizontal, vertikal, diagonal) als ein Symbol am Rand (nämlich nur zwei, vertikal und horizontal).

Wie würde eine ML-Methode nun vorgehen, um einen TicTacToe-spielenden Algorithmus aus Trainingsdaten zu erlernen? Auf irgendeine Weise muss die ML-Methode einen Algorithmus finden, der so ähnlich wie der oben skizzierte *Sebastians TicTacToe-Algorithmus Version 1.0* aufgebaut ist. Jedoch „programmieren" ML-Methoden in der Regel nicht in der gleichen Programmiersprache wie Menschen. Das liegt daran, dass ML-Methoden grundlegend anders lernen als Menschen. Daher haben Forscher sozusagen spezielle Programmiersprachen entwickelt, die von Maschinen wesentlich besser gehandhabt werden können. ML-Forscher nennen eine auf ML-Methoden zugeschnittene Programmiersprache ein *Modell*, und jede ML-Methode beruht auf und trainiert eine andere Art von Modell.

Künstliche neuronale Netze für TicTacToe

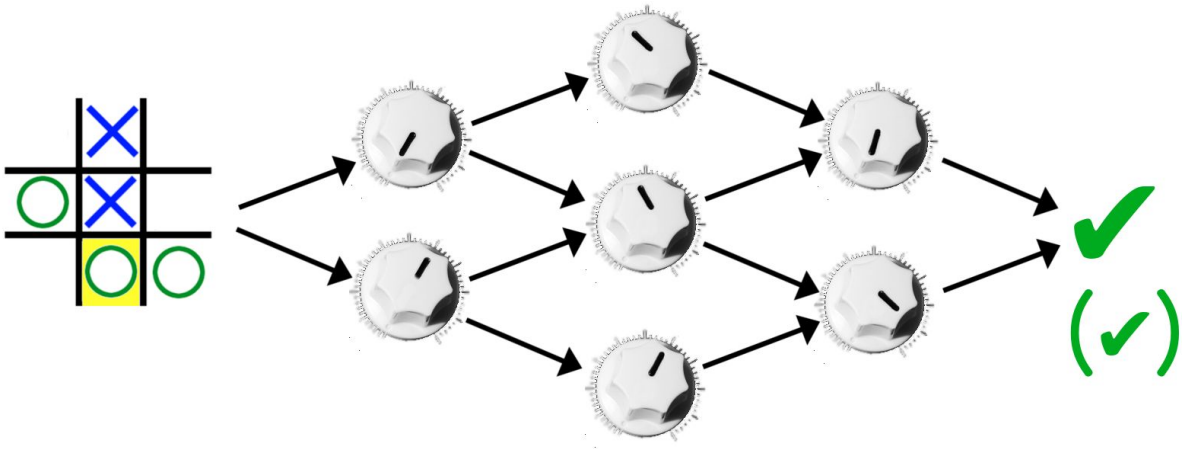

Abbildung 3: Eine schematische Darstellung eines künstlichen neuronalen Netzes. Auf der linken Seite wird als Eingangssignal ein TicTacToe-Spielfeld mit einem vorgeschlagenen Zug (in gelb) präsentiert. Das Netz schickt dieses Eingangssignal durch verschiedene Schichten, in denen es „verzerrt" wird. Die Verzerrung hängt von der Stellung jedes „Reglers" im Netz ab. Am Ende wird das Eingangssignal so verzerrt, das die Ausgabe entsteht, also hier eine Bewertung des Zugs. Um das Netz zu trainieren, wird die Ausgabe dann mit den Trainingsdaten verglichen (dargestellt durch das Häkchen in Klammern). Wenn die Ausgabe und die Trainingsdaten nicht übereinstimmen, dreht der Lernalgorithmus an den Reglern, um eine Übereinstimmung herzustellen.

Ein momentan sehr beliebtes Modell, also sozusagen eine beliebte Programmiersprache unter ML-Methoden, ist das *künstliche neuronale Netz*. Die Methode wird oft auch als *Deep Learning* bezeichnet. Künstliche neuronale Netze haben eigentlich bis auf eine lose Assoziation nichts mit menschlichen neuronalen Netzen zu tun. Allerdings umgibt sie gerade aufgrund dieser Assoziation sowohl im akademischen als auch nicht-akademischen Umfeld eine gewisse Magie. Man kann sich ein künstliches neuronales Netz vorstellen als eine Kaskade hinter- und nebeneinander geschalteter Drehregler, ähnlich wie Lautstärke-, Tiefen- und Höhenregler an einer Stereoanlage, skizziert in Abbildung 3. Links bekommt das neuronale Netz als Eingangssignal das TicTacToe-Feld mit einer Stellung von Kreuzen und Kreisen sowie einen Zug, und rechts spuckt es als Ausgabe die Bewertung des Zuges aus. Um einmal von links nach rechts durch das Netz zu kommen, verzerrt das neuronale Netz die Eingabe derart durch die verschiedenen Regler, dass am Ende die gewünschte Ausgabe herauskommt. Wie die Regler einer Stereoanlage filtert und deformiert das neuronale Netz dazu das Eingangssignal. Während die Regler der Stereoanlage jedoch lediglich die Tiefen und Höhen eines Liedes herausfiltern kann, kann das neuronale Netz das Eingangssignal derart drastisch deformieren, dass es das TicTacToe-Feld zu einer Bewertung eines Spielzugs deformiert.

Die Aufgabe der ML-Methode ist es nun die richtige Reglerstellung zu finden. Das funktioniert meistens nach folgendem Schema, umgesetzt durch einen sogenannten *Lernalgorithmus*:

1) Zuerst werden die Regler auf eine zufällige Stellung gedreht.
2) Dann werden die Eingabedaten, also die TicTacToe-Spielstellungen und Züge, in das Netz gefüttert und dadurch eine Ausgabe, die Bewertung, produziert. Da die Regler am Anfang zufällig stehen, ist auch diese Bewertung zunächst zufällig und daher meist falsch.
3) Nun setzt das eigentliche Lernen ein: die ausgegebene Bewertung wird mit der gewünschten Bewertung (*Label*) verglichen, und daraufhin werden alle Regler ein kleines bisschen so angepasst, dass das neuronale Netz weniger Fehler beim Bewerten der präsentierten Trainingsbeispiele macht.
4) Dies wird so lange wiederholt, bis das Netzwerk möglichst viele Trainingsbeispiele richtig bewertet.

Tatsächlich funktioniert diese bemerkenswert einfache Prozedur in der Praxis sehr gut. ML-Methoden sind hervorragend darin, sehr viele Regler gleichzeitig um winzige Nuancen so zu verändern, dass am Ende das richtige Ergebnis herauskommt. Deswegen sind diese Art von Modellen bzw. Programmiersprachen sehr viel besser geeignet für Maschinen als für Menschen.

Generalisieren, nicht Auswendiglernen

Vielleicht stellt sich Ihnen nun die Frage, warum wir etwas derart Kompliziertes wie eine Kaskade von Reglern überhaupt benötigen - wir könnten doch die Trainingsbeispiele einfach auswendig lernen! Dazu könnten wir sie beispielsweise in einer großen Tabelle abspeichern, und vor jedem unserer Züge in der Tabelle den besten Zug nachschlagen. Tatsächlich ist das eine absolut valide Idee: Die effiziente Suche von Einträgen in Tabellen ist das Spezialgebiet von *Suchalgorithmen*, die unter anderem von Suchmaschinenbetreibern wie Google perfektioniert werden. Die Suche in Tabellen ist nur eben kein Lernen im Sinne des maschinellen Lernens: Diese Vorgehensweise fällt nämlich dann in sich zusammen, wenn wir einer neuen Spielsituation begegnen, die *nicht* in der Tabelle auftaucht. Und es ist leicht zu sehen, dass für kompliziertere Spiele eine solche Tabelle sehr schnell viel zu groß werden würde: Zum Beispiel wäre eine derartige Tabelle für das Go-Spiel so groß, dass sie mehr Einträge hätte als es Partikel im Universum gibt[4]!

Der Clou am maschinellen Lernen ist daher die Fähigkeit zu *generalisieren*: Entscheidungen treffen zu können in Situationen, die *nicht* Teil der Trainingsdaten sind. Künstliche neuronale Netze mit ihren vielen Reglern eignen sich in vielen Fällen sehr gut dazu diese Generalisierung zu erreichen. Allerdings hängt diese Fähigkeit stark davon ab, dass die Netze auch mit genügend relevanten Daten gefüttert werden. Sonst droht das sogenannte *Overfitting*: In diesem Fall lernt die ML-Methode ein neuronales Netz, welches die Trainingsdaten sehr gut vorhersagt, aber überhaupt nicht mit den neuen, während des Trainings ungesehenen Daten zurechtkommt. Overfitting zu erkennen und zu vermeiden

[4] Die Anzahl der regelkonformen Go-Spielpositionen werden auf etwa 10^{170} (eine Zahl mit 170 Nullen) beziffert, die Anzahl der Partikel im Universum auf 10^{86}. Siehe https://tromp.github.io/go/legal.html und http://mrob.com/pub/math/numbers-19.html.

erfordert viel Expertenwissen und praktische Erfahrung, und ist eine der Hauptschwierigkeiten bei der Anwendung des maschinelles Lernens.[5]

Lernen durch Ausprobieren: Reinforcement Learning

Bisher haben wir maschinelles Lernen als passiven Prozess dargestellt: Eine ML-Methode bekommt kuratierte Trainingsdaten vorgesetzt und sucht nach Mustern in diesen Trainingsdaten. Dies ist jedoch fundamental anders davon wie Menschen lernen: Menschen bekommen nicht im Detail jeden Schritt von einem Experten erklärt - in vielen Fällen ist es auch gar nicht möglich, jedes Trainingsbeispiel eindeutig zu labeln. Anders als unser maschineller Lerner treten Menschen daher gegen andere Spieler an, probieren verschiedene Spielzüge in verschiedenen Situation aus und entwickeln so selbstständig Strategien.

In der Tat beschäftigt sich die aktuelle Forschung nicht nur mit dem Supervised-Learning-Paradigma, sondern auch mit Paradigmen, die maschinelles Lernen als interaktiven Prozess auffassen. Diese Paradigmen werden oft unter dem Begriff des *Reinforcement Learning* (*bestärkendes Lernen*) zusammengefasst. Die Idee ist hier, dass der Lerner nicht in jedem Spielzug gesagt bekommt, was der beste Zug ist, sondern erst am Ende des Spiels ein positives oder negatives Feedback erhält, je nachdem ob das Spiel gewonnen oder verloren wurde. Mit diesem Feedback muss der Lerner dann auf bessere Spielzüge kommen und diese immer wieder ausprobieren. Diese Art des Lernens liegt auch AlphaGo, dem Go-Algorithmus von Google DeepMind zugrunde, der 2017 einen der besten Go-Spieler der Welt schlug. Dieses Resultat konnten die Forscher von Google DeepMind dadurch erzielen, dass sie in AlphaGo verschiedene Techniken kombinierten, nämlich Reinforcement Learning, künstlichen neuronalen Netzen (in Kombination auch *deep reinforcement learning* genannt) und klassische Suchalgorithmen, ähnlich wie sie bei IBM's Schachprogramm Deep Blue zum Einsatz kamen.

Reinforcement Learning wird momentan von vielen intensiv erforscht, doch ist es oft schwierig, diese Methode in praktischen Anwendungen einzusetzen. Der Grund ist, dass Reinforcement Learning um ein Vielfaches datenhungriger als Supervised Learning ist: AlphaGoZero[6], die neueste Version von AlphaGo, hat ganze 49 Millionen Spiele gegen sich selbst spielen müssen, um Weltklasseniveau zu erreichen. Zum Vergleich: man schätzt, dass ein professioneller Go-Spieler etwa 50.000 Partien in seinem Leben gespielt hat, um Weltklasseniveau zu erreichen.

Diese Diskrepanz zeigt nicht nur auf, dass Menschen viele Dinge immer noch viel effizienter Lernen als Maschinen. Es zeigt auch auf, warum maschinelles Lernen gerade in den letzten Jahren so erfolgreich geworden ist: Den Hauptanteil am Erfolg des maschinellen Lernens haben die immense Rechenpower und die riesigen Datenmengen, die heutzutage verfügbar sind.

---

[5] Forscher und Anwender von maschinellem Lernen haben sogar mehr Angst vor Overfitting als vor den Zombies in Michael Jacksons's Thriller: https://www.youtube.com/watch?v=DQWI1kvmwRg

[6] David Silver et al. Mastering the game of Go without human knowledge. Nature, 2016.

Grenzen des maschinellen Lernens

Auf den letzten Seiten habe ich Ihnen dargelegt, wie Algorithmen funktionieren, und wie Lernalgorithmen es ermöglichen Algorithmen bzw. Modelle aus Daten zu lernen. Wir haben bereits die größte Herausforderung des maschinellen Lernens thematisiert: das Overfitting, d.h. dass eine ML-Methode sich auf die Trainingsdaten spezialisiert ohne die Fähigkeit über ungesehene Daten zu generalisieren. Da das Problem der Generalisierung absolut zentral ist, möchte ich im Folgenden tiefer auf dieses Problem eingehen und es von zwei Seiten beleuchten. Zum einen möchte ich aufzeigen, dass die Grenzen der Generalisierung eines gelernten Modells trotz sehr großer Mengen an Trainingsdaten tatsächlich sehr schnell erreicht werden. Zum anderen möchte ich auf die Problematik der Uninterpretierbarkeit von Modellen hinweisen, d.h. die Schwierigkeit im Nachhinein als Mensch zu charakterisieren, worüber und wie ein gelerntes Modell generalisiert - mit anderen Worten, was es eigentlich gelernt hat.

Repräsentation ist der Schlüssel

Bei unserer Abhandlung von TicTacToe habe ich bisher eine der wichtigsten Fragen überhaupt unterschlagen: Wie werden die Trainingsdaten eigentlich *repräsentiert*? Um besser zu verstehen, was ich mit Repräsentation meine, betrachten wir noch einmal Abbildung 1A. Wir erkennen diese Grafik sofort als TicTacToe-Spielsituation, aber wie erkennt ein Computer eine solche eigentlich? Eine Grundprämisse der Informatik ist, dass alle Daten durch Zahlen repräsentiert werden können. Im Falle eines TicTacToe-Spiels könnten wir also eine Repräsentation wie in Abbildung 1D wählen, in der wir jedes Kreuz in eine 1, jeden Kreis in eine 2, und jedes freie Feld in eine 0 übersetzen. Diese Zuordnung von Symbolen zu Zahlen ist zwar leicht für eine ML-Methode zu verwenden, natürlich aber arbiträr. Auch wenn diese Repräsentation für TicTacToe gut funktionieren kann, ist in vielen Fällen überhaupt nicht klar, mit welcher Repräsentation die ML-Methode am besten zurechtkommt. Andersherum ist in vielen Fällen die richtige Repräsentation schon die halbe Miete, um der ML-Methode das effiziente Auffinden einer Lösung zu ermöglichen.

1	2	3	4	5	6	7	8	9

Abbildung 4: Die Spielkarten des Zahlenspiels.

Um Ihnen zu verdeutlichen, wie sehr auch unsere menschliche Intelligenz von passenden Repräsentationen abhängig ist, möchte ich Ihnen kurz ein weiteres Spiel vorstellen[7]. In diesem Spiel liegen vor Ihnen und einem Gegenspieler neun Karten, numeriert von eins bis neun (Abbildung 4). Sie und Ihr Gegenspieler suchen sich abwechselnd eine der neun Karten aus und legen diese auf Ihren Stapel. Es gewinnt, wer zuerst drei Karten auf seinem Stapel hat, die in der Summe genau 15 ergeben. Gelingt das keinem, endet das Spiel unentschieden. Was ist Ihre Taktik, um zu gewinnen? Nehmen Sie sich einen Moment Zeit, um darüber nachzudenken.

[7] Jorge Aranda: Fun with representations I – Nine numbers, 2016. Blog post: https://catenary.wordpress.com/2006/08/19/fun-with-representations-i-nine-numbers/

Obwohl die Spielregeln nicht sehr kompliziert sind, fällt es mir persönlich sehr schwer, eine Gewinnstrategie zu finden. Interessanterweise ist es wesentlich einfacher, wenn wir die neun Karten in einem *magischen Quadrat* anorden, in dem jede Reihe, Spalte und Hauptdiagonale in der Summe die gleiche Zahl ergibt, hier 15:

4	9	2
3	5	7
8	1	6

Abbildung 5: Ein magisches Quadrat

Das Spiel sollte Ihnen nun bekannt vorkommen - es ist äquivalent zu TicTacToe! Entlang jeder Spalte, Zeile und Diagonale beträgt die Summer nämlich immer 15. Tatsächlich ist es sehr oft der Fall, dass dieselben Probleme auf sehr viele verschiedene Arten und Weisen betrachtet und repräsentiert werden können. Und genauso wie es für einen Menschen einfacher ist, ein Problem in einer gewissen Repräsentation zu lösen als in einer anderen, so gestaltet es sich auch für einen maschinellen Lerner einfacher aus Daten in geeigneten Repräsentationen zu lernen. Jedoch sind diese Repräsentationen in der Regel problemspezifisch - eine Tatsache, die es erschwert geeignete Repräsentationen zu finden.

Generalisierung über Repräsentationen

Kommen wir nach unserem kurzen Exkurs über Repräsentationen zurück zur Frage der Generalisierung. Selbstverständlich spielt die Repräsentation der Trainingsdaten eine wichtige Rolle für die Generalisierungsfähigkeit eines gelernten Modells. Vergleichen Sie Abbildungen 1A-1D. Alle Abbildungen stellen die gleiche TicTacToe-Spielsituation dar, und doch wäre ein Modell, welches nur Abbildung 1A bzw. die numerische Version in Abbildung 1D in seinen Trainingsdaten gesehen hat, völlig außer Stande mit Abbildungen 1B und 1C etwas anzufangen. Der Grund dafür ist, dass in 1B und 1C das Erkennen der Symbole wesentlich schwieriger ist, da sie sehr unsauber auf den Asphalt gekritzelt wurden - und auch noch aus verschiedenen Perspektiven aufgenommen wurden. Zwar könnten wir ein neuronales Netz trainieren, welches das Spielfeld und die TicTacToe-Symbole in Abbildungen 1B und 1C erkennt: dazu müsste aber wieder ein Experte mühevoll Trainingsbeispiele bestehend aus TicTacToe-Spielfeldern aus verschiedenen Perspektiven sammeln, diese labeln, und so weiter.

Diese Einsicht zeigt ganz klar die wichtigste Grenze des heutigen maschinellen Lernens auf: Die Generalisierung aktueller Methoden ist stark beschränkt durch die zum Trainieren verwendeten Daten. Ein Modell lässt sich leicht überlisten, wenn es auf Daten angewandt wird, die sich deutlich von den Trainingsdaten unterscheiden. Und “deutlich” ist in diesem Fall sehr viel weniger als man intuitiv erwarten würde. Schlimmer noch: Möchte man die Fähigkeiten eines Modells erweitern, müssen nicht nur zusätzliche Daten gesammelt

werden, das Modell muss in den meisten Fällen von Grund auf neu trainiert werden[8]. Mit verschiedenen Perspektiven umgehen? Neue Daten, neu trainieren! Mit anderen Symbolen umgehen, z.B. Dreieck und Schlange statt Kreis und Kreuz (Abbildung 1E)? Neue Daten, neu trainieren! Die Spielregeln ändern und eine Spalte hinzufügen (Abbildung 1F)? Neue Daten, neu trainieren!

Dieses Beispiel zeigt, dass man sehr viel Vorsicht dabei walten lassen muss, wieviel man in die vermeintlichen Fähigkeiten eines Modells interpretiert. Dies gilt selbstverständlich nicht nur für TicTacToe, sondern für ziemlich alle Anwendungen von maschinellem Lernen. Ein weiteres Beispiel ist die maschinelle Übersetzung. Heutige Systeme sind bereits so gut, dass sie sinnvolle kommerzielle Anwendung finden. Zwar sind die Übersetzungen immer noch nicht perfekt, aber sie sind gut genug, dass Menschen den Sinn verstehen und wichtige Informationen daraus gewinnen. Das ist ohne Zweifel ein immenser Fortschritt, der eine Vielfalt faszinierender Technologien hervorgebracht hat. Viele der Fehler, die maschinelle Übersetzer machen, sind jedoch auf fehlenden Bezug auf die wirkliche Welt (der maschinelle Übersetzer hat beispielsweise nie einen Baum gesehen oder berührt) und mangelndes Verständnis des Kontexts (Sprecher, Epoche) eines Textes oder fehlenden *Common Sense* zurückzuführen[9]. Es ist nach wie vor eine offene Forschungsfrage, wie Maschinen diese Probleme lösen können.

Adversarielle Beispiele

Die Generalisierung ist allerdings nicht nur durch die Repräsentation beschränkt, auch innerhalb der gleichen Repräsentation lassen sich heutige ML-Methoden in die Irre führen. Vor einiger Zeit sorgte die Entdeckung sogenannter adversarieller Beispiele[10] für großes Aufsehen. Hierbei wurde ein künstliches neuronales Netz, welches zur Erkennung von Bildern trainiert wurde, durch das gezielte Hinzufügen von Störsignalen zu natürlichen Bildern getäuscht (Abbildung 6). In einer Art und Weise gelang es den Forschern, die Modelle „optisch zu täuschen", ähnlich wie auch Menschen diversen optischen Täuschungen unterlegen sind[11]. Die Forscher fügten den Originalbildern dazu subtile Störsignale hinzu, welche für das menschliche Auge nicht wahrnehmbar sind, aber das trainierte Modell maximal in die Irre führen.

Ich möchte jedoch darauf hinweisen, dass adversarielle Beispiele für ein gegebenes System zu finden nicht trivial ist. In der erwähnten Forschungsarbeit gelang es den Wissenschaftlern solche Beispiele zu finden, da sie direkten Zugang zum trainierten neuronalen Netz hatten. Dies ermöglichte den Wissenschaftlern, so lange Millionen verschiedener Beispielbilder zu generieren, bis sie schlussendlich adversarielle Bilder mit den gewünschten Eigenschaften fanden. Im praktischen Anwendungsfall würde ein System das neuronale Netze unter

---

[8] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, Samuel J. Gershman. Building machines that learn and think like people. Behavioral and Brain Sciences, 2017
[9] Douglas Hofstadter: The Shallowness of Google Translate. The Atlantic. 30. Januar 2018 (https://www.theatlantic.com/technology/archive/2018/01/the-shallowness-of-google-translate/551570)
[10] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. Explaining and Harnessing Adversarial Examples. International Conference on Learning Representations, 2015.
[11] Man erinnere sich nur an das goldene bzw. blaue Kleid: https://en.wikipedia.org/wiki/The_dress

Verschluss halten und, ähnlich wie ein Bankautomat, nach dreimaliger falscher Eingabe den Zugang sperren.

Dennoch ist die Existenz solcher Beispiele ein Sicherheitsrisiko, wenn trainierte Modelle in sicherheitskritischen Gebieten eingesetzt werden. Aus Forschungssicht zeigt die Existenz adversarieller Beispiele zudem auf, dass diese Modelle visuelle Daten auf fundamental andere Art und Weise verarbeiten als der Mensch.

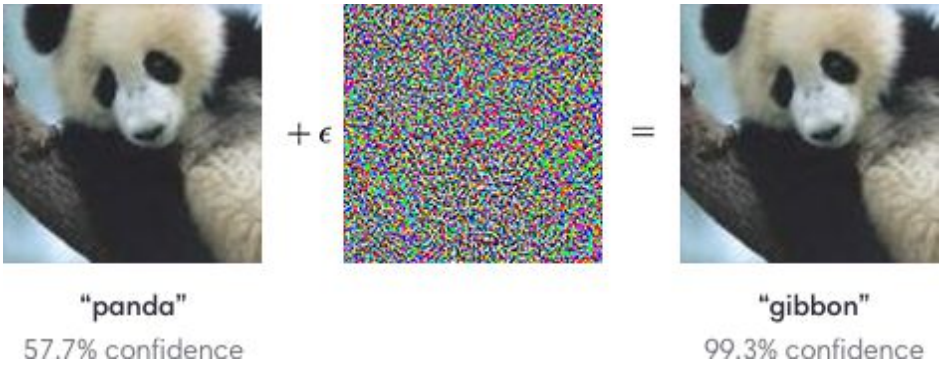

Abbildung 6: Künstliche neuronale Netze sind anfällig für gezielt ausgesuchte, für Menschen nicht wahrnehmbare Störsignale. Hier wird ein neuronales Netz davon überzeugt, das Bild eines Pandas als Gibbon zu fehlinterpretieren. (Bild entnommen aus [12])

Uninterpretierbarkeit von Modellen

Gelernte Modelle verarbeiten Daten nicht nur anders als der Mensch; wie zuvor erläutert sind sie auch fundamental anders programmiert als von Menschen programmierte Algorithmen. Ich erwähnte zuvor, dass ML-Methoden sehr gut darin sind, viele kleine Regler zu tunen, um ein Problem zu lösen. Allerdings stellt sich schnell die Frage, was die gefundenen Reglereinstellungen eigentlich bedeuten. Tatsächlich ist in den vielen Fällen, besonders bei Anwendungen von künstlichen neuronalen Netzen, die Reglereinstellung für Menschen schwer oder gar nicht interpretierbar. Mit anderen Worten hat ein Mensch nur wenige Möglichkeiten nachzuvollziehen, was ein neuronales Netz eigentlich gelernt hat. Wenn das Netz zum Beispiel systematische Fehler macht, ist es nicht klar, wie es eigentlich repariert werden soll, außer dass man die Anzahl und Art der Regler variiert und zusätzliche Trainingsdaten hinzufügt. Wie im letzten Abschnitt erklärt, erfordert das aber in der Regel das komplette Neutrainieren des Modells - was zu komplett anderen Reglereinstellungen und unter Umständen auch anderen Fehlern führen kann.

Diese Uninterpretierbarkeit von Modellen ist vor allem deshalb besonders gravierend, weil die gelernten Modelle ja über die Trainingsdaten hinaus generalisieren sollen. Keine endliche Menge von automatisierten Tests kann daher garantieren, dass das Modell nicht etwas Seltsames gelernt hat. Und wenn wir nicht wissen, was das Modell gelernt hat, wie sollen wir einschätzen, ob es richtig generalisiert hat? Dies ist kein theoretisches Problem

[12] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel & Jack Clark: Attacking Machine Learning with Adversarial Examples. 24. Februar 2017 (https://blog.openai.com/adversarial-example-research/)

mehr, denn schon heutzutage kämpfen wir zum Beispiel mit sexistischen oder rassistischen Gesichtserkennungsmodellen, weil die genutzten Trainingsdaten vor allem weiße Männer abbilden[13].

Jedoch muss einschränkend gesagt werden, dass nicht alle Modelle derart schwierig zu interpretieren sind wie neuronale Netze. Beispielsweise lassen so genannte *lineare Modelle* durchaus Rückschlüsse darauf zu, weswegen eine bestimmte Entscheidung gefällt wurde. Zwar sind diese Modelle nicht so mächtig wie neuronale Netze, sind aber leichter zu interpretieren, und dazu noch wesentlich einfacher zu handhaben und weniger datenhungrig. Daher werden diese Modelle seit Jahrzehnten erfolgreich in kommerziellen Anwendungsgebieten sowie in der Forschung eingesetzt[14] und haben trotz der Popularität von neuronalen Netzen nicht an Relevanz eingebüßt.

Zusammenfassung und Ausblick

Auf den letzten Seiten habe ich Ihnen versucht näher zu bringen, was Algorithmen sind, wie sie automatisch gelernt werden können, und was die Grenzen der Fähigkeiten dieser Algorithmen sind. Es ist offensichtlich, dass maschinelles Lernen in den letzten Jahren immense technologische Fortschritte gebracht hat und selbst eine bahnbrechende technologische Errungenschaft ist. Die größten Fortschritte wurden dabei mit der Verarbeitung und Generierung von sogenannten *Echtwelt-Daten* erzielt. Das sind Daten, mit denen wir Menschen tagtäglich mit Leichtigkeit umgehen, beispielsweise Bilder oder Sprache. Maschinelles Lernen hat es geschafft eine Brücke zu bauen, damit wir mit unseren virtuellen Assistenten reden können und die Stimme dieser Assistenten nicht wie eine Roboterstimme klingt, damit Computer unsere Handschrift und unsere Gesichter auf Fotos erkennen. All dies ist möglich geworden durch die Verfügbarkeit nie dagewesener Rechenpower und Datenmengen.

Jedoch benötigt die erfolgreiche Anwendung maschinellen Lernens immer noch signifikantes Expertenwissen und menschliche Intervention sowie große Datenmengen. Dennoch ist die Generalisierungsfähigkeit der gelernten Modelle in vielen Gebieten noch immer weit unter dem, was man intuitiv erwarten würde. Zudem ist die mangelnde Interpretierbarkeit vieler Methoden ein wichtiges praktisches Problem, das je nach Anwendungsfall auch gesellschaftliche Fragen aufwerfen kann.

---

[13] Vgl. Parmy Olson: Racist, Sexist AI Could Be A Bigger Problem Than Lost Jobs. Forbes, 26. Februar 2018. (https://www.forbes.com/sites/parmyolson/2018/02/26/artificial-intelligence-ai-bias-google/); Hannah Steinharter, Michael Maisch: Wenn Algorithmen den Menschen diskriminieren. Handelsblatt, 26. August 2018 (https://amp.handelsblatt.com/unternehmen/banken/kuenstliche-intelligenz-wenn-algorithmen-den-menschen-diskriminieren/22949674.html)

[14] Beispielsweise sind diese Modelle in der Neurowissenschaft sehr verbreitet, um Rückschlüsse auf die Funktionsweise des Gehirns zu ziehen, siehe:
Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, Feix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. Neuroimage 87, 96-110 (http://doc.ml.tu-berlin.de/bbci/publications/HauMeiGoeDaeHayBlaBie13.pdf)

Die nächsten Jahre werden zeigen, wie die Forschung voranschreiten wird, um die Generalisierungsfähigkeit des maschinellen Lernens und der künstlichen Intelligenz zu erweitern und so nützliche neue Anwendungsgebiete zu erschließen.

Dr. rer. nat. Sebastian Höfer ist Forscher im Bereich Robotik und maschinelles Lernen. Die Meinungen, Forschung und Ergebnisse in diesem Artikel sind die des Autors und spiegeln nicht die Ansicht seines Arbeitgebers wider.
