Wie funktioniert maschinelles Lernen?

Ulrike von Luxburg
Universität Tübingen
ulrike.luxburg@uni-tuebingen.de

19. August 2020

1 Vorwort

Maschinelles Lernen und künstliche Intelligenz sind beherrschende Themen unserer Zeit. Es vergeht kaum eine Woche, in der nicht eine der großen Tageszeitungen ausführlich darüber berichtet. Einerseits wird von Durchbrüchen berichtet, die große Chancen bergen: Algorithmen, in eine Handy-App verpackt, können genauer als medizinische Fachleute vorhersagen, ob sich hinter einer Hautverfärbung vermutlich Krebs verbirgt oder nicht; durch automatische Übersetzung ist es möglich, sich im Urlaub in einem fremden Land über das Handy mit Leuten in ihrer Landessprache zu unterhalten; minimal-invasive, durch Drohnen gesteuerte Landwirtschaft kann den Verbrauch an Dünger, Insektiziden oder Wasser viel sparsamer und zielgerichteter einsetzen; und selbstfahrende Autos versprechen weniger Verkehrstote. Andererseits wird vor heraufziehenden Gefahren gewarnt: viele Arbeitsplätze könnten verloren gehen; die in der Demokratie so wichtige Berichterstattung kann durch perfekt gefälschte Videos unterminiert werden; und Algorithmen treffen doch keine so fairen Entscheidungen, wie mancher vermutet hätte. Umfassende Überwachung wie in China oder automatisierte Waffen sind dann das andere Ende des Spektrums, ganz zu schweigen von Berichten, dass "superintelligente" Roboter kurz vor der Verwirklichung stehen, die die gesamte Menschheit bedrohen könnten. In Unternehmen und Politik beschäftigt man sich damit, wie selbstlernende Algorithmen und künstliche Intelligenz unsere Wirtschaftskraft verändern könnten. Auf der anderen Seite fragen sich die Menschen, ob und welche Entscheidungen sie an Algorithmen delegieren wollen (sollen?) und wer eigentlich die Verantwortung übernimmt, wenn dabei etwas schief geht.

Als Laie muss man in dieser Berichterstattung fast zwangsläufig verloren gehen, hat doch jeder Experte und jede Expertin eine andere Meinung, die man selbst schwer beurteilen kann: denn was sich eigentlich hinter dem Begriff "künstliche Intelligenz" verbirgt, bleibt weitgehend im Nebel. Was ist "künstliche Intelligenz" eigentlich und wie funktioniert sie? Was sind "selbstlernende Algorithmen", und was hat es mit den "tiefen neuronalen Netzen" auf sich, die überall erwähnt werden? Was sind realistische Szenarien vom Einsatz von künstlicher Intelligenz, und welche Szenarien sind maßlose Übertreibungen?

In diesem Artikel will ich versuchen, eine allgemein verständliche Einführung in das Thema "künstliche Intelligenz" und "maschinelles Lernen" geben. Dazu beschreibe ich die grundlegenden Konzepte und Mechanismen, die diese beiden Gebiete vorantreiben. Ich richte mich bewußt an ganz "normale Leute", die keinerlei Vorwissen in Informatik besitzen.

2 Künstliche Intelligenz? Maschinelles Lernen!

Bevor wir richtig loslegen, will ich zunächst einige Begriffe diskutieren.

Algorithmus

Im Rest des Artikels wird oft von Algorithmen die Rede sein. Im Prinzip ist ein Algorithmus für einen Computer so etwas wie eine Ausführungsbestimmung. Wenn man so will, das Kochrezept. Oder auch: das Programm, dass die Programmiererin dem Computer eingegeben hat. Ein Algorithmus beschreibt in systematischer Reihenfolge die einzelnen Rechenschritte, die der Computer

nacheinander ausführen soll. An dieser Stelle möchte ich darauf verzichten, eine weitergehende Erklärung des Begriffs "Algorithmus" vorzunehmen, denn wir werden dies im Folgenden nicht brauchen.

Künstliche Intelligenz

In der öffentlichen Debatte wird fast immer der Begriff "künstliche Intelligenz" (kurz: KI) verwendet, wenn von selbstlernenden Algorithmen die Rede ist. Ich mag diesen Begriff überhaupt nicht, denn ich halte ihn für irreführend und manipulativ! Er suggeriert, dass selbstlernende Algorithmen entweder schon intelligent sind oder es demnächst sein werden. Beides ist meiner Meinung nach nicht der Fall, und das werde ich im Weiteren ausführlich begründen.

In der öffentlichen Debatte trägt der Begriff "künstliche Intelligenz" unterschwellig dazu bei, Nicht-Fachleute einzuschüchtern und aus der Diskussion auszuschließen: wer traut sich schon zu, hier mitzureden? Künstliche Intelligenz impliziert ja schon fast eine schöpferische Kraft, die dem Computer innewohnt und die nicht mehr erklärbar oder nachvollziehbar ist. Ein Einstein im Computer. Implizit dient der Begriff als einfache Entschuldigung dafür, viele Details gar nicht erst erklären zu müssen, denn künstliche Intelligenz ist ja offensichtlich so kompliziert, dass man sie nicht in wenigen Sätzen darstellen könnte. Dadurch werden die technischen Leistungen, die sich hinter den neueren Entwicklungen verbergen, nahezu unangreifbar, oder zumindest un-diskutierbar.

Verstehen Sie mich nicht falsch: ich behaupte nicht, dass der Begriff "künstliche Intelligenz" mit Absicht benutzt wird, um die oben erwähnten Effekte zu produzieren, ganz im Gegenteil. Ursprünglich stammt der Begriff aus der Wissenschaft der 1950er Jahre — damals hatten sich völlig überhöhte Hoffnungen mit den neuen Techniken verbunden, was sich im Namen des Forschungsgebietes niederschlug. Heutzutage verwenden Journalist*innen verwenden den Begriff, denn er macht es einem einfach, gute Geschichten darüber schreiben. Politiker*innen verwenden den Begriff, denn er lässt sich mit tollen zukünftigen Errungenschaften assoziieren (und lässt dabei im Nebel, worum es eigentlich geht). Man kann alles und nichts darüber aussagen.

Aber welche Begriffe wir in einer Debatte benutzen, beeinflußt immer auch auf subtile Weise die Inhalte der Debatte. **Damit wir eine sinnvolle Debatte über Chancen und Risiken führen können, halte ich es daher für wichtig, dass wir nicht von "künstlicher Intelligenz" sprechen, sondern den Begriff "maschinelles Lernen" oder "selbst-lernende Verfahren" verwenden.**

Maschinelles Lernen

Das maschinelle Lernen ist die Technologie, die für all die Durchbrüche verantwortlich ist, von denen man so oft in der Zeitung liest. Der ganze Rest meines Artikels soll erklären, was sich dahinter verbirgt. Lassen Sie uns an dieser Stelle mit einer kurzen Beschreibung beginnen, wie sie sich — so oder ähnlich — an vielen Stellen finden lässt:

> Die Wissenschaft des Maschinellen Lernens entwickelt Verfahren, durch die ein Computer bestimmte Aufgaben selbständig an Hand von Beispielen erlernen kann.

Diese Erklärung trägt natürlich noch nicht sehr viel zum Verständis bei. Trotzdem enthält sie einige Aspekte, die wichtig sind und die wir im Folgenden tiefer beleuchten werden. Ein selbstlernendes Verfahren hat zunächst einmal nicht das Ziel, eine wie auch immer geartete, abstrakte "Intelligenz" zu erreichen. Statt dessen wird ein Verfahren durchgeführt, das den Computer dazu in die Lage versetzen soll, **eine (!) bestimmte (!) Aufgabe** auszuführen. Die Aufgabe kann sein, Bilder von Hautverfärbungen in die Kategorien "Hautkrebs" oder "kein Hautkrebs" zu sortieren. Oder einen deutschen Satz ins Englische zu übersetzen. Oder eine Spielstrategie für das Brettspiel Halma zu entwickeln. Zu diesem Zweck werden dem Algorithmus **Beispiele** für die zu erlernende Aufgabe vorgeführt: er bekommt Bilder von Hautverfärbungen zu sehen, die von Ärzten als "Hautkrebs" oder "kein Hautkrebs" klassifiziert worden sind; er bekommt Beispiele von deutschen Sätzen mit englischen Entsprechungen gezeigt; oder er sieht Spielzüge aus Halma-Spielen und bekommt gesagt, ob der Spieler am Ende gewonnen oder verloren hat. Basierend auf diesen Beispielen besteht die Rolles des Computers nun darin, **selbständig** eine gute Strategie zu finden, mit deren Hilfe

die gestellte Aufgabe zufriedenstellend gelöst werden kann. Dabei sucht er sich aus einer Menge von vorgegebenen, möglichen Strategien diejenige aus, die insbesondere bei den vorgegebenen Beispielen gute Ergebnisse erzielt hätte.

Die Rolle der Wissenschaft ist nun, dieses Auswahlverfahren — das maschinelle Lernen — so zu gestalten, dass die Wahl, die vom Computer dabei getroffen wird, auch für zukünftige Beispiele gut funktioniert: auch bei neuen Patienten soll der Computer Hautkrebsstellen erkennen; er soll auch bisher nicht gesehene deutsche Sätze ins Englische übersetzen können; oder er soll auch in einer neuen Spielsituation von Halma einen sinnvollen Zug machen. Diese Eigenschaft heißt die "Generalisierung": ein selbstlernendes System soll nicht nur bei den schon vorgegebenen Trainings-Beispielen die richtigen Vorhersagen treffen — das wäre ja witzlos — sondern auch für neue, bisher nicht gesehene Test-Beispiele funktionieren.

An dieser Stelle ist es mir wichtig zu betonen, dass der Computer genau die eine, vorbestimmte Aufgabe anhand der gezeigten Beispiele erlernen kann. Er kann danach aber nicht einfach eine ganz andere, neue Aufgabe aus einem anderern Themenfeld durchführen. Für jede aneue Aufgabe muss er neu trainiert werden.

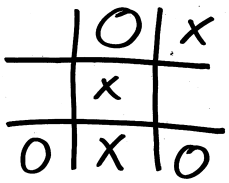
Wie das alles funktioniert, soll nun anhand von vielen verschiedenen Beispielen erklärt werden.

3 Der selbstlernende Briefumschlag-Computer: warum ein selbstlernendes System nicht intelligent sein muss

In diesem Kapitel stelle ich Ihnen ein ganz einfaches, selbstlernendes System vor: den Briefumschlag-Computer. Er besteht nur aus Papier, und trotzdem kann er selbständig aus Erfahrungen lernen, ein perfekter Drei-Gewinnt-Spieler zu werden. Zugegeben, der Briefumschlag-Computer ist einfacher als das, was die meisten von uns im Kopf haben, wenn sie über selbstlernende Computer oder "künstliche Intelligenz" nachdenken. Trotzdem beinhaltet der Briefumschlag-Computer eigentlich alle Zutaten, die auch ein richtiger, selbstlernender Algorithmus hat.

Drei-gewinnt

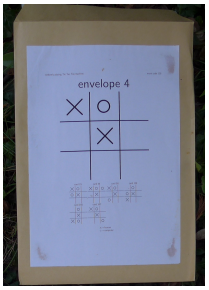
Aus Ihrer Schulzeit kennen viele von Ihnen sicher noch das Spiel "Drei-Gewinnt", auch Kreis-und-Kreuz oder Tic-Tac-Toe genannt. Ein Spieler ist Kreuz, der andere Kreis. Abwechselnd markieren die Spieler auf einem 3-mal-3 Felder großen Spielfeld ein Feld mit Ihrem Symbol. Es hat gewonnen, wer zuerst drei seiner Symbole in einer Reihe, Spalte oder der Diagonalen hat.



Ich will Ihnen nun zeigen, wie wir nur aus Briefumschlägen und Zetteln einen "Computer" basteln können, der selber lernt, ein perfekter Drei-Gewinnt-Spieler zu sein. Am Anfang kennt er nur die Spielregeln, das heißt, er kann gültige Züge machen, aber mehr auch nicht. Er wird daher viele dumme Züge machen und oft verlieren. Wir werden ihn dann aber anhand von Beispielen so lange trainieren, bis er gute von dummen Zügen unterscheiden kann und ein perfekter Drei-Gewinnt-Spieler wird.

Der Briefumschlag-Computer

Mechanismus. Der Computer ist wie folgt aufgebaut. Es spielt immer ein Mensch gegen den Computer. Der Mensch benutzt das Symbol "Kreuz", der Computer das Symbol "Kreis". Der Computer besteht aus vielen Briefumschlägen. Jeder Briefumschlag beschreibt eine konkrete Spielsituation, in der der Computer sich befinden kann. Diese ist immer vorne auf dem Umschlag abgebildet. In



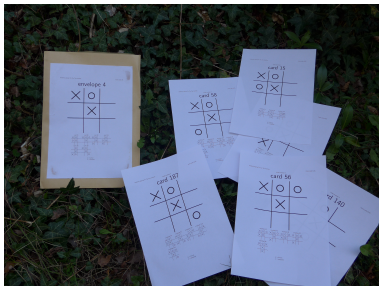


Abbildung 1: Links: der Briefumschlagcomputer. Mitte: einer der Umschläge. Rechts: einer der Umschläge und die darin befindlichen Zettel.

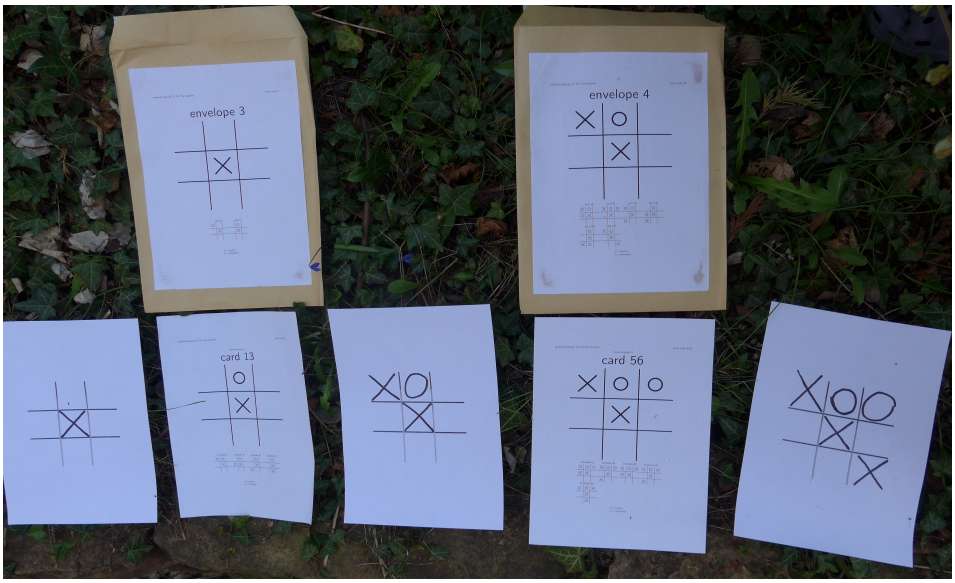
dem Briefumschlag befinden sich mehrere Zettel. Jeder Zettel steht für einen Spielzug, den der Computer in dieser Situation machen kann. Schauen wir uns dieses Prinzip anhand von Abbildung 1 an. In der in der Mitte abgebildeten Spielsituation hat der Mensch schon zwei Kreuze gesetzt, der Computer hat einen Kreis gesetzt (wie auf dem braunen Umschlag zu sehen ist). Nun ist der Computer am Zug, einen weiteren seiner Kreise zu setzen. Dafür gibt es genau sechs Möglichkeiten, nämlich die sechs leeren Kästchen auf dem Spielfeld. Im Umschlag befinden sich nun sechs Zettel, von denen jeder einen der möglichen Züge abbildet. Wir simulieren nun einen Spielzug des Computers, indem wir mit geschlossenen Augen einen der Zettel aus dem Umschlag ziehen. Der auf diesem Zettel abgebildete Zug wird nun ausgeführt. Danach ist der Mensch wieder an der Reihe und macht seinen nächsten Zug. Nun befinden wir uns in einer neuen Spielsituation, die wieder durch einen Briefumschlag beschrieben ist. Aus diesem ziehen wir nun wieder einen Zettel, um den Zug des Computers auszuführen, und so weiter. Ein Beispiel für ein vollständiges Spiel finden Sie in Abbildung 2.

Das Lernverfahren. Am Anfang starten wir mit einem völlig unwissenden Computer: in jedem seiner Umschläge sind einfach alle möglichen Spielzüge enthalten, die er in dieser Situation machen könnte. Unter diesen Spielzügen wählt der Computer zufällig einen aus — ohne jegliche Wertung, ob der Zug gut oder schlecht ist. Gegen diesen Computer fällt es leicht zu gewinnen, weil er viele schlechte Züge macht. Aber jetzt kommt der entscheidende Schritt: der Computer soll nun lernen, schlechte Spielzüge zu vermeiden und gute Spielzüge zu bevorzugen.

Dazu gehen wir wie folgt vor. Zunächst spielt ein menschlicher Spieler ein ganz normales Spiel gegen den Computer. Dabei merken wir uns, welche Züge der Computer während des Spieles durchgeführt hat. Dazu legen wir einfach die benutzten Umschläge in einer Reihe aus und legen den jeweils benutzten Zettel unter den jeweiligen Umschlag (wie in Abbildung 2 dargstellt). Am Ende des Spiels stellen wir fest, ob der Computer gewonnen oder verloren hat. Falls der Computer gewonnen hat, gibt es nichts zu verbessern. Wir stecken einfach alle Zettel, die im Spiel verwendet wurden, wieder in die Umschläge zurück, und das Spiel kann von vorne losgehen. Falls der Computer verloren hat, muss er aus seinen Fehlern lernen. Insbesondere soll er nicht die schlechten Spielzüge wiederholen, die zu seiner Niederlage geführt haben. Um das zu erreichen, ergreifen wir eine radikale Trainingsmaßnahme: wir vernichten die Zettel von allen Zügen, die der Computer im ganzen Spiel durchgeführt hat und stecken sie nicht wieder in die Umschläge zurück. Das hat zur Folge, dass der Computer im nächsten Spiel keinen der Züge mehr durchführen kann, die er in diesem Spiel gespielt hat. Zumindest wird er also nicht genau den gleichen Fehler wieder machen.

Das ist also das **Lernprinzip**: Wir spielen oft gegen den Computer. Wenn er gewonnnen hat machen wir nichts. Wenn er verloren hat, vernichten wir die entsprechenden Zettel. Dieses Trainingsprogramm wiederholen wir nun oft, etwa 500 Spiele müssen zum Training gespielt werden. Am Ende dieses Trainings hat der Briefumschlag-Computer "gelernt", ein perfekter Drei-Gewinnt-Spieler zu sein. Er wird nie wieder verlieren, wenn er Drei-Gewinnt spielt!

An dieser Stelle werden üblicherweise Einwände laut: *Was, wenn ein Umschlag im Training ganz leer wird, weil alle Zettel daraus entfernt worde sind? (Ja, kann passieren, allerdings höchst selten). Kann man auch Belohnungen bei gewonnenen Spielen einführen? (Ja, macht Sinn). Das funktioniert vielleicht bei Drei-Gewinnt, aber bei Schach klappt das nicht mehr, oder? (Stimmt.)* Das oben



Zug 1:	Zug 2:	Zug 3:	Zug 4:	Zug 5:
Mensch setzt sein Kreuz.	**Computer setzt seinen Kreis.**	**Mensch setzt sein Kreuz.**	**Computer setzt seinen Kreis.**	**Mensch setzt sein Kreuz.**

Abbildung 2: Ein komplettes Spiel gegen den (noch untrainierten) Briefumschlag-Computer. Der Mensch fängt an und setzt sein Kreuz in die Mitte. Diese Spielsituation ist auf Umschlag 3 abgebildet. Wir simulieren einen Zug des Computers, indem wir einen der Zettel ziehen, die sich in Umschlag 3 befinden. Dieser Zettel zeigt, dass der Computer seinen Kringel über das Kreuz des Menschen setzt, das ist Zug 2. Nun ist wieder der Mensch am Zug, und er wählt ein Kreuz in der oberen linken Ecke. Im vierten Zug ziehen wir wieder einen Zettel aus dem Umschlag des Computers, der daraufhin einen Kringel in die rechte obere Ecke setzte. Im fünften und letzten Spielzug setzt der Mensch sein Kreuz in die untere rechte Ecke und hat gewonnen.

vorgestellte Trainingsprogramm ist wirklich ziemlich einfach, und man kann es auf viele erdenkliche Weisen verbessern. Wir wollen das hier aber nicht weiter diskutieren, um uns auf das allgemeine Prinzip zu konzentrieren.

Die Grundprinzipien.

Das oben vorgestellte Trainingsprogramm ist so einfach, dass es schon fast enttäuschend wirkt. Das soll nun maschinelles Lernen sein? Ja, im Prinzip schon. Denn der Briefumschlag-Computer benutzt genau die gleichen Zutaten, aus denen auch die "echten" Algorithmen des maschinellen Lernens aufgebaut sind.

1. **Die Aufgabe:** Der Briefumschlag-Computer hat ein klar definiertes Ziel: nach dem Training soll er Drei-Gewinnt spielen können (und sonst nichts). Diese Aufgabe ist dem Computer **fest vorgegeben.**

2. **Der Suchraum:** Das Ziel des Trainings ist es, zu jeder Spielsituation zu lernen, welches gute Züge sind. Dazu hat der Computer eine Menge von "möglichen Zügen", aus denen er auswählen kann. Diese Züge sind ihm **fest vorgegeben**. Er kann also nicht beliebige Dinge tun, etwa ein neues Symbol erfinden, ein schon belegtes Kästchen nochmal belegen, oder einfach aussetzen. Diese Menge der Möglichkeiten, die der Computer bei jedem Zug hat, nennen wir den "Suchraum": in diesem "Raum" kann der Computer nach einer guten Strategie suchen, aber nirgends sonst.

3. **Die Bewertungsfunktion:** Irgendwie müssen wir dem Computer sagen können, welcher Zug gut oder schlecht war. In diesem Fall machen wir das ganz einfach. Falls der Computer gewonnen hat, deklarieren wir alle getätigten Züge als gut; falls er verloren hat, als schlecht. Diese Bewertungsfunktion ist dem Computer **fest vorgegeben.**

4. **Die Trainingsdaten:** Nun müssen wir Trainingsdaten erzeugen. Das machen wir dadurch, dass wir gegen den Computer spielen. Am Ende jedes Spieles werten wir aus, ob der Computer gewonnen oder verloren hat. Die Trainingsdaten bestehen also aus einer Folge von Spielen, zusammen mit deren Ergebnissen. Diese Daten werden zu einem gewissen Grade **zufällig erzeugt**, denn wir können im Vorhinein nicht wissen, welche Spielzüge genau der Computer ausprobieren wird.

5. **Das Lernverfahren / der Lernalgorithmus**: Das Lernverfahren selbst besteht nun darin, die Trainingsdaten auszuwerten und die Spielstrategie anzupassen. In diesem Fall besteht der Algorithmus darin, bei gewonnenen Spielen einfach nichts zu verändern (Zettel unverändert wieder in den Umschlag stecken), und bei verlorenen Spielen die benutzen Spielzüge zu eliminieren (Zettel vernichten). Dieses Lernverfahren ist dem Computer **fest vorgegeben**, er erfindet es nicht selber und kann auch nicht davon abweichen.

6. **Das Ergebnis:** Nachdem der Briefumschlag-Computer die Trainingsdaten ausgewertet und verarbeitet hat, ist durch seinen "Zustand" — die Zettel in seinen Umschlägen — eine Strategie festgelegt, mit der er in Zukunft spielen wird. Diese gelernte Strategie ist das Einzige am Briefumschlag-Computer, das **nicht fest vorgegeben ist und das wir auch nur bedingt kontrollieren können.** Wir können nur hoffen, dass am Ende des Trainings wirklich ein perfekter Briefumschlag-Computer enstanden ist.

Lassen Sie mich alles nochmal zusammenfassen. Der Briefumschlag-Computer soll eine klar definierte Aufgabe erlernen (Drei-Gewinnt spielen). Er hat eine Menge von möglichen Strategien (Spielzüge) fest vorgegeben, aus der er auswählen kann. Auch das Lernverfahren ist fest vorgegeben (Zettel unverändert lassen oder vernichten). Mit Hilfe dieses Lernverfahrens und der vorgegebenen Bewertungsfunktion kann der Computer anhand von Trainingsbeispielen eine Spielstrategie entwickeln. Der Computer ist "selbstlernend": wir geben ihm den Mechanismus vor, mit dessen Hilfe er lernen kann (Suchraum, Bewertungsfunktion, Lernalgorithmus). Die Spielstrategie selbst, die er dadurch lernt, können wir nicht vorhersagen, denn die entwickelt der Computer "selber". Je nachdem, welche Trainingsspiele in welcher Reihenfolge gespielt werden, können hierbei durchaus unterschiedliche Spielstrategien herauskommen (die im Beispiel von Drei-Gewinnt so gut wie immer den Computer unbesiegbar machen).

Im Laufe des Artikels werden wir noch viele weitere Lernverfahren betrachten. Auch wenn sie in der Umsetzung komplizierter sind, bestehen sie im Wesentlichen aus genau den Zutaten, die für den Briefumschlag-Computer wichtig waren.

Intelligenz? Sicher nicht!

Wir haben ein Verfahren entwickelt, mit dessen Hilfe der Briefumschlag-Computer selbst aus Erfahrungen lernen kann, Drei-Gewinnt zu spielen. Ist der Briefumschlag-Computer also intelligent? Sicher nicht. Er besteht nur aus einem Haufen von Briefumschlägen und Zetteln, und er kann nichts ausser Drei-Gewinnt spielen. An diesem Beispiel können wir sehen, dass ein selbstlernendes System nicht notwendigerweise Intelligenz besitzen muss. Das ist für mich eine der wichtigsten Erkenntnisse, die in diesem Artikel enthalten sind: Viele technische Systseme vollbringen erstaunliche Leistungen, und fast automatisch schreiben wir solchen Systemen Intelligenz oder auch Kreativität zu. Beides ist jedoch keine notwendige Voraussetzung für die erstaunlichen Fähigkeiten des Systems.

4 Wie ein Computer lernen kann, Blumen zu bestimmen

Stellen Sie sich vor, dass Sie mit einer Freundin einen Spaziergang machen. Ihre Freundin ist Botanikerin und weist Sie auf verschiedene Blumensorten hin. Ihnen gefallen besonders die Schwertlilien, und Ihre Freundin ist begeistert. In ihrem Überschwang erklärt Sie Ihnen, dass obwohl sich die Schwertlilien alle sehr ähnlich sehen, hier drei unterschiedliche Arten von Schwertlilien vorliegen: Borsten-Schwertlilie (Iris Setosa), Schillernde Schwertlilie (Iris Versicolor) und Blaue Sumpfschwertlilie (Iris Virginica), siehe Abbildung 3. Sie würden nun gerne lernen, diese drei Arten auseinander zu halten. Aber obwohl Ihnen die Freundin genau die Unterschiede erklärt, gelingt es Ihnen nicht, die komplizierten Muster der Blütenblätter zu unterscheiden.



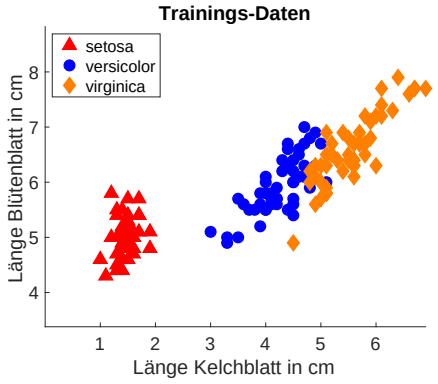
Abbildung 3: Drei Arten Schwertlilien: Iris Setosa, Iris Versicolor, Iris Virginica (Copyright: wikimedia)

Klassifikation mit Papier und Bleistift

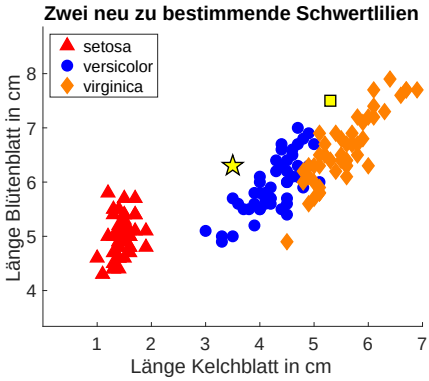
Daraufhin überlegen Sie sich gemeinsam eine andere Vorgehensweise, die wie ein Verfahren des maschinellen Lernens funktioniert. Von jeder Lilie, der Sie von nun ab begegnen, erheben Sie "Daten": Sie messen die Länge des größten Blütenblattes und die Länge des größten Kelchblattes (die Kelchblätter sind die kleinen grünen Blätter, die sich an der Unterseite der Blüte beim Übergang zum Stängel befinden). Nun führen Sie Buch: von jeder Schwertlilie notieren Sie sich die Länge von Blüten- und Kelchblatt, und Ihre Freundin bestimmt, um welche der drei Arten von Schwertlilien es sich handelt. Am Ende des Spaziergangs haben Sie 150 Schwertlilien vermessen, und Ihre Daten sehen wie folgt aus:

Länge Blütenblatt (in cm)	Länge Kelchblatt (in cm)	Schwertlilien-Art
5,1	1,4	Iris-setosa
7,0	4,7	Iris-versicolor
7,1	5,9	Iris-virginica
4,9	1,4	Iris-setosa
6,4	4,5	Iris-versicolor
6,3	5,6	Iris-virginica
...	...	...

Nun übertragen wir diese Daten in ein Koordinatensystem. Jede Blume wird durch einen Punkt dargestellt: der Wert an der x-Achse zeigt die Länge des Kelchblattes an, der Wert an der y-Achse zeigt die Länge des Blütenblattes an. Die Farbe des Punktes symbolisiert die Art der Blume.



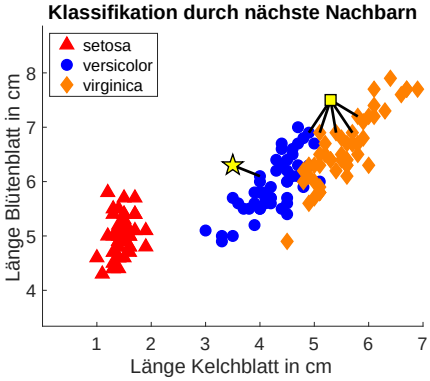
Und schon erkennen wir ein Muster: die Blumen der verschiedenen Arten sind in der Abbildung relativ klar voneinander getrennt. Stellen Sie sich nun vor, sie wollen nun eine neue, unbekannte Schwertlilie bestimmen, ohne Ihre Freundin zu fragen. Sie können ganz einfach vorgehen. In Ihre Abbildung tragen Sie die neue Blume ein und "schauen scharf hin": in welche der drei Farbgruppen passt die Blume am besten hinein?



Mit Hilfe der Abbildung kann man diese Frage oft einfach beantworten. Zum Beispiel ist die durch den gelben Stern dargestellte neue Blume vermultlich von der gleichen Art wie die blauen Kreise, also eine Iris-versicolor. Bei dem gelben Quadrat sind wir uns vielleicht nicht ganz so sicher, aber würden vermutlich die orangene Klasse vorhersagen, also Iris-virginica.

Automatische Klassifikation durch nächste Nachbarn

Nun wollen wir verstehen, wie ein Computer diese Aufgabe selbständig lösen könnte. Dazu müssen wir das "scharf Hinsehen", das wir selbst gerade anhand der Abbildung durchgeführt haben, einem Computer übertragen. Eine einfache Möglichkeit hierfür ist die "Nächste-Nachbar-Regel". Für die neue Blume, die es zu bestimmen gilt, berechnet der Computer zunächst, in welchem Abstand sie sich im Koordinatensystem von den schon bekannten Trainings-Blumen befindet. Dann wertet er aus, welche der schon bekannten Blumen den kürzesten Abstand zur neuen Blume hat: sie wird der nächste Nachbar genannt. Dann sieht der Computer nach, was die Lilienart des nächsten Nachbarn ist, und sagt diese für die neue Blume vorher. Zum Beispiel ist der nächste Nachbar des gelben Sterns ein blauer Kreis, also eine Iris versicolor. Daher sagt der Computer auch diese Klasse für den gelben Stern voraus.

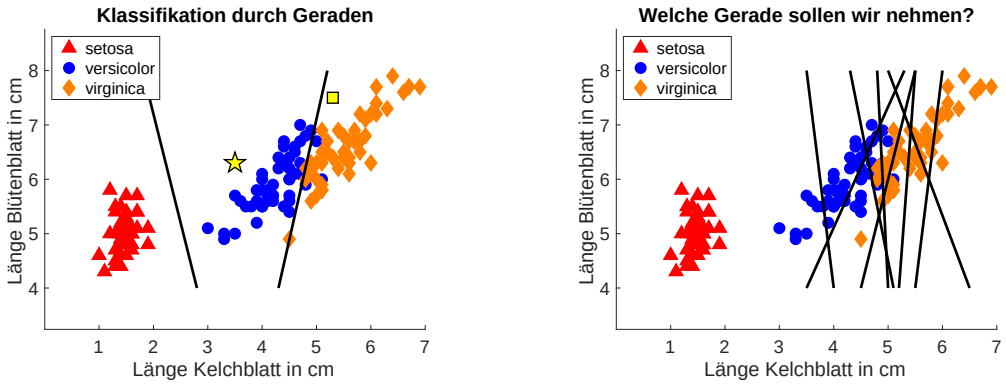


Wenn man möchte, kann man die Nächste-Nachbar-Regel noch robuster machen: statt nur den allernächsten Nachbarn zu betrachten, könnten wir zum Beispiel immer die fünf nächsten Nachbarn heraussuchen und diese abstimmen lassen, welches die richtige Klasse des neuen Punktes ist. In der Abbildung sind zum Beispiel die fünf nächsten Nachbarn des gelben Quadrates eingezeichnet. Hiervon sind vier Nachbarn orange und ein Nachbar ist blau. Die Mehrheit der Nachbarn ist also orange, und das sagen wir nun auch für das neue gelbe Viereck voraus: Iris-virginica.

Und hier sind wir mitten im maschinellen Lernen angekommen. Die Nächste-Nachbar-Regel ist ein beliebter Algorithmus, um einfache Klassifikationsprobleme zu lösen.

Automatische Klassifikation durch Geraden (lineare Klassifikation)

Es gibt noch ein weiteres Klassifikations-Verfahren, dass sich am Beispiel der Lilien gut erklären lässt und das auch sehr weit verbreitet ist. Es heißt "lineare Klassifikation". Wir veranschaulichen es wieder durch eine Abbildung. Das Ziel des Verfahrens ist es, Geraden zu finden, die je zwei Klassen von Lilien gut voneinander trennen können. Im Bild trennt die linke Gerade die rote und die blaue Klasse voneinander, und die rechte Gerade versucht das gleiche für die blaue und orange Klasse. Das gelbe Sternchen würde also als blau (Iris-versicolor) und das gelbe Quadrat als orange (Iris-virginica) bestimmt. Schon anhand der Abbildung können wir vermuten, dass die Trennung zwischen rot und blau ziemlich gut funktioniert, während die Trennung zwischen blau und orange auch einige Fehler produziert (manche blaue Punkte liegen auf der falschen Seite).



Wir wollen nun die Aufgabe, solche trennenden Geraden zu finden, einem Computer übertragen. Betrachten wir der Einfachkeit halber nur das Problem, die blauen und orangen Lilien voneinander zu unterscheiden (Iris versicolor und Iris virginica). Das Ziel des Algorithmus ist, diese beiden Arten von Lilien gut voneinander zu unterscheiden. Der Suchraum ist die Menge aller möglichen Geraden. Nur welche Gerade er aussucht, das wollen wir dem Computer überlassen. Dazu benötigt der Computer wieder eine Bewertungsfunktion, die besagt, wie gut eine spezielle Gerade ist. Hier nehmen wir den sogenannten Trainingsfehler: Für jede Gerade berechnet der Computer, wie viele der Trainings-Lilien von dieser Gerdade in die falsche Klasse eingeteilt werden. Also zum Beispiel, wie viele orange Trainingspunkte fälschlicherweise auf der linken Seite, und wie viele blaue Trainingspunkte fälschlicherweise auf der rechen Seite der Gerade liegen. Je weniger Fehler gemacht werden, desto besser ist die Gerade. Der Lernalgorithmus besteht nun darin, im Suchraum aller möglichen Geraden diejenige Gerade zu finden, die den kleinsten Trainingsfehler hat. Das ist nun ein ganz normales Optimerungsproblem, wie es von Informatik und Mathematik seit Jahrzehnten studiert worden ist. Wir tragen dem Computer also auf, dieses Optimierungsproblem durch ein bekanntes Optimierungsverfahren zu lösen. Die Gerade, die der Computer dadurch erhält, ist nun die Lösung des Lernproblems: Wir können alle neuen Schwertlilien mit Hilfe dieser Geraden den Klassen Iris versicolor und Iris virginica zuordnen. Vielleicht passieren dabei einzelne Fehler, aber im Großen und Ganzen wird es einigermaßen funktionieren.

Fast alle Zutaten für diese Herangehensweise haben wir dem Computer fest vorgegeben: das Ziel (Lilien bestimmen), den Suchraum (alle Geraden), die Trainingsdaten (die Messwerte der Lilien), die Bewertungsfunktion (Trainingsfehler), und den Lernalgorithmus (ein Optimierungsverfahren). Nur was dabei herauskommt — welche Gerade genau — das können wir nicht von Anfang an vorhersehen.

Im allgemeinen Fall heisst das soeben vorgestellte Verfahren "lineare Klassifikation" — in der Sprache der Mathematik sind Geraden "lineare Funktionen". Lineare Klassifikation funktioniert auch, wenn wir mehr als zwei Messwerte haben. Die Details möchte ich nicht beschreiben, das Prinzip ist aber genau das gleiche wie oben. Neben der Nächsten-Nachbar-Regel ist die lineare Klassifikation ein absolutes Standardverfahren im Bereich des maschinellen Lernens.

Es kann zum Beispiel zum Einsatz kommen, wenn eine Firma Kunden in verschiedene Gruppen einteilen will. Von jedem Kunden werden "Messwerte" genommen: Wo wohnt er, wie viel Geld hat

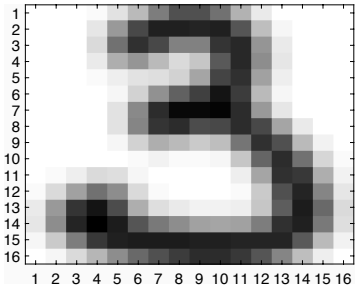
er bisher ausgegeben, welche Dinge hat er bisher gekauft, zu welcher Tageszeit, wie lange besucht er welche Webseiten, usw. Anhand dieser "Messwerte" kann man nun versuchen, bestimmte Eigenschaften des Kunden vorherzusagen: Ist der Kunde ein Mann oder eine Frau? Wird er sich eher ein Laptop oder einen Fernseher kaufen? Oft funktionieren lineare Klassifikatoren zwar nicht perfekt, aber doch erstaunlich gut, um solche Fragen zu lösen.

Eine ähnliche Vorgehensweise kann man sich einen medizinischen Kontext vorstellen. Die Messwerte des Patienten sind medizinische Daten: Blutdruck, Leberwerte, Temperatur, Alter, vorherige Erkrankungen, usw. Anhand dieser Werte kann man dann zum Beispiel versuchen vorherzusagen, ob man eine bestimmte Krankheit eher mit dem einen oder dem anderen Medikament behandeln sollte.

Die wichtigste Erkenntnis aus diesem Kapitel ist für mich: Maschinelles Lernen besteht oft darin, aus einer vorgegebenen Menge von möglichen Funktionen die "beste" auszuwählen. Ob eine Funktion gut oder schlecht ist, kann (in einfachen Fällen) durch die Anzahl ihrer Fehler auf den Trainingsdaten bewertet werden.

5 Handgeschriebene Ziffern bestimmen

Als nächstes möchte ich Ihnen ein "echtes" Problem vorstellen, das näher an Anwendungen ist als Drei-Gewinnt oder die Klassifikation von Lilien. Es geht um die Erkennung von handgeschriebenen Ziffern, wie sie die Post täglich in ihren Briefsortieranlagen lösen muss: Um Briefe an den Empfänger zu schicken, muss die Postleitzahl auf dem Umschlag erkannt werden. Stellen wir uns vor, eine Kamera hat ein Bild einer handgeschriebenen Ziffer aufgenommen. Das Ergebnis könnte so aussehen:



0.0	0.0	0.0	0.0	0.1	0.3	0.5	0.7	0.7	0.6	0.3	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.4	0.7	0.9	0.9	0.9	0.9	0.7	0.3	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.2	0.5	0.8	0.7	0.5	0.5	0.8	0.8	0.4	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.3	0.4	0.3	0.1	0.2	0.6	0.8	0.4	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.3	0.7	0.8	0.4	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.7	0.9	0.8	0.3	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.5	0.8	1.0	1.0	1.0	0.8	0.4	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.5	0.8	0.8	0.7	0.7	0.8	0.7	0.4	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.2	0.3	0.3	0.2	0.3	0.5	0.8	0.7	0.3	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.2	0.6	0.8	0.5	0.2	0.0
0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.8	0.8	0.3	0.1
0.0	0.1	0.4	0.6	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.7	0.9	0.5	0.1
0.1	0.4	0.8	0.9	0.7	0.3	0.1	0.1	0.1	0.1	0.1	0.2	0.6	0.9	0.6	0.2
0.1	0.4	0.9	1.0	0.9	0.7	0.5	0.4	0.4	0.3	0.4	0.5	0.8	0.9	0.5	0.1
0.0	0.2	0.5	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.8	0.7	0.3	0.1
0.0	0.0	0.1	0.3	0.5	0.6	0.7	0.8	0.8	0.8	0.8	0.7	0.5	0.3	0.1	0.0

In diesem Beispiel besteht das Bild aus 16 mal 16 Pixeln. Jedes Pixel ist ein kleines Quadrat, das einen bestimmten "Grauwert" enthält. Die Grauwerte werden durch Zahlen beschrieben. Weiss erhält die Zahl 0, Schwarz die Zahl 1, und alle Zahlen zwischen 0 und 1 beschreiben verschiedene Grautöne: je näher an 0 die Zahl ist, desto heller, und je näher an 1, desto dunkler ist der Grauwert. Die handgeschriebene Ziffer in diesem Bild wird also duch $16 \times 16 = 256$ Zahlen zwischen 0 und 1 beschrieben — ein Grauwert für jedes der Pixel. Und genau so wird ein Bild im Computer repräsentiert: als eine "Matrix" von 256 Zahlen (siehe Abbildung, rechte Seite).
Nun kann man vielleicht erahnen, warum es nicht offensichtlich ist, wie ein Computer handgeschriebene Ziffern bestimmten kann. Er bekommt die Matrix von 256 Zahlen als Eingabe, und soll nun bestimmen, um welche der Ziffern von 0 bis 9 es sich bei dem dadurch repräsentierten Bild handelt.

Hier kommt nun wieder das maschinelle Lernen zum Einsatz. Wir erstellen zunächst einen Trainingsdatensatz: Wir bitten viele Leute, Ziffern auf ein Blatt Papier zu schreiben, nehmen diese Ziffern mit einer Kamera auf und verwandeln sie in die Darstellung von 256 Grauwerten. Diese Matrizen von Grauwerten geben wir nun dem Computer als Trainigsdaten, zusammen mit der Information, um welche Ziffern es sich dabei jeweils handelt. Nun soll der Computer lernen, solche Grauwert-Matrizen richtig zu klassifizieren.

In diesem Fall stellt sich heraus, dass sich das Problem schon mit einem einfachen Ansatz erstaunlich gut lösen lässt, nämlich mit der Nächsten-Nachbar-Regel. Im Wesentlichen brauchen wir hierfür nur noch eine Zutat: wir müssen dem Computer eine Regel vorgeben, wie er den Abstand

zwischen zwei Bildern, also zwischen zwei Grauwert-Matrizen, berechnen soll. Dafür gibt es in der Mathematik eine ganz offensichtliche Lösung, die auch auf Anhieb funktioniert (diese Abstandsfunktion heißt der "Eulidische Abstand", aber das ist hier nicht weiter wichtig). Um nun eine neue handgeschriebene Ziffer zu klassifizieren, gehen wir also wie folgt vor: wir nehmen die neue handgeschriebene Ziffer (dargestellt durch eine Matrix von 256 Grauwerten), und geben sie dem Computer ein. Der Computer berechnet nun den Abstand zwischen der neuen Test-Ziffer und den Ziffern aus dem Trainingsdatensatz. Dann bestimmt er, welches die – sagen wir 15 – nächsten Trainingsziffern sind, und nimmt deren Mehrheitsvotum als Vorhersage für die neue Test-Ziffer. Mit solch einem Verfahren kann man in diesem Fall schon eine erstaunlich hohe Genauigkeit erzielen, etwa 90% der Vorhersagen sind richtig. Das reicht der Post vielleicht noch nicht, weil immer noch zu viele Briefe fehlgeleitet werden, aber mit komplizierteren Verfahren kann man die Genauigkeit noch deutlich erhöhen. Weiter unten werden wir dafür Beispiele sehen.

Rekapitulieren wir noch einmal kurz, wie wir das Problem der Erkennung handgeschriebener Ziffern mit Hilfe des maschinellen Lernens lösen können. Wir nehmen Trainingsdaten in Form von Kamerabildern von handgeschriebenen Ziffern auf. Diese Bilder werden in die vorgegebene Darstellung von 256 Grauwerten überführt. Wir schreiben dem Computer vor, dass er neue Bilder anhand der 15-Nächsten-Nachbar-Regel klassifizieren soll. Wir geben ihm vor, wie der Abstand zwischen zwei Bildern zu berechnen ist, um die nächsten Nachbarn zu bestimmen. Mit all diesen Zutaten kann der Computer nun neue handgeschriebene Ziffern richtig klassifizieren, und dabei kommt in 90% der Fälle das richtige Ergebnis heraus. Gar nicht so kompliziert, oder?

6 Die Fähigkeit zur Generalisierung

Manchen Lesern ist vermutlich schon etwas aufgefallen. Bisher habe ich immer recht genau beschrieben, was die Zutaten eines Lernverfahren sind und wie es durchgeführt wird. Beim Ergebnis des Lernverfahrens bin ich jedoch vage geblieben und habe lediglich die "Hoffnung" geäußert, dass ein gutes Ergebnis herauskommt: eine gute Spielstrategie bei Drei-Gewinnt, oder eine Gerade, die zukünftige Lilien richtig bestimmt. Diese Hoffnung zu erfüllen ist natürlich der Knackpunkt am maschinellen Lernen. Wir sprechen von der "Fähigkeit zur Generalisierung": eine gelernte Strategie soll von den bekannten Trainingsbeispielen auf neue, bisher unbekannte Beispiele verallgemeinern können.

Wie gut ist eine Klassifikationsregel? Trainings- und Testfehler bei Schwertlilien

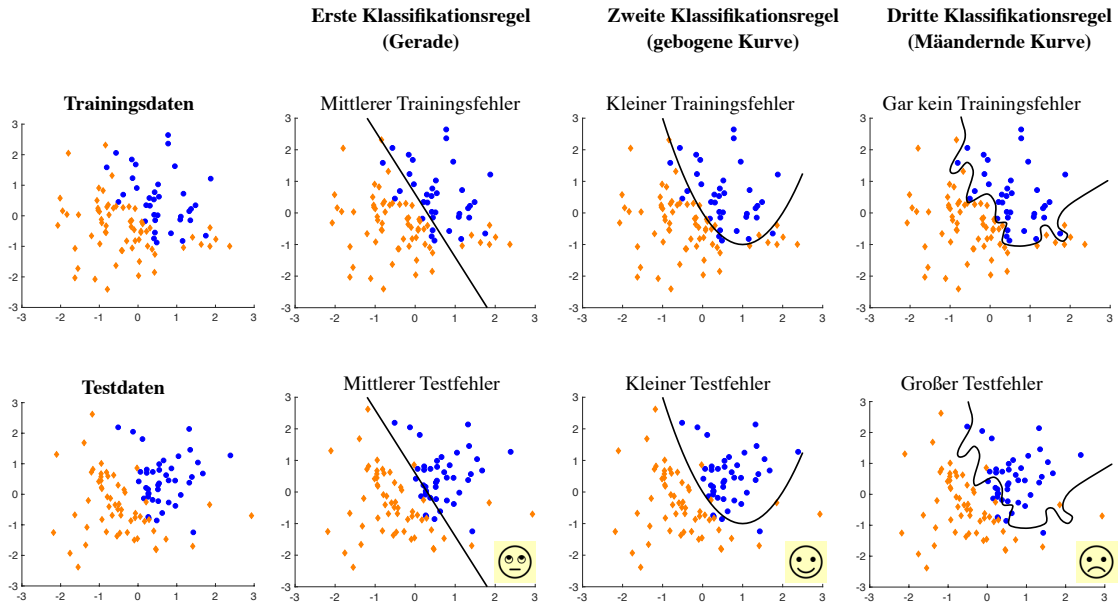
Lassen Sie uns den Lern- und Evaluationprozess anhand der Schwertlilien beschreiben. Um eine Klassifikationsregel lernen zu können, sammeln wir zunächst einen Trainingsdatensatz von 150 Blumen und lernen mit ihrer Hilfe eine Gerade, wie oben beschrieben. Für diese Gerade können wir auswerten, wie viele der Trainings-Blumen von dieser Gerade falsch klassifiziert werden. Dies ist der "Trainingsfehler" der Gerade. Nun "hoffen wir", dass diese Gerade auch zukünftige Lilien, die wir bisher noch nicht gesehen hatten, richtig klassifiziert.

Um das systematischer auszuwerten, erheben wir nun einen weiteren Datensatz, den Test-Datensatz. Wieder machen wir einen Spaziergang, messen Kelch-und Blütenblatt von Schwertlilien und fragen unsere Expertin, um welche Blumen es sich handelt. Wir tragen nun diese Testblumen in das bestehende Koordinatensystem ein und werten aus, ob die von uns zuvor gewählte Gerade die Testpunkte richtig klassifiziert. Der hierbei entstehende Fehler heißt der Testfehler der Funktion. **Dieser Testfehler ist die Größe, die uns am maschinellen Lernen interessiert!**

Ganz wichtig ist hierbei, dass wir die Testdaten erst zu sehen bekommen, wenn der Lernprozess abgeschlossen ist. Die Testdaten haben keinerlei Einfluss darauf, welche Gerade wir lernen. Sie sollen als neutrale Testinstanzen dienen, um die Güte der gelernten Gerade abzuschätzen.

Wie verhalten sich Trainings- und Testfehler zueinander? “Overfitting” und “Underfitting”

Das Grundproblem des maschinellen Lernens besteht also darin, eine Klassifikationsregel zu finden, die hoffentlich einen kleinen Testfehler hat. Allerdings können wir zur Trainingszeit natürlich nur den Trainingsfehler auswerten. Es stellt sich nun heraus, dass der Trainingsfehler nicht unbedingt eine gute Vorhersage darüber abgibt, wie hoch der Testfehler wohl sein wird. Lassen Sie mich das anhand eines weiteren Beispiels illustrieren. Stellen Sie sich wieder vor, dass wir Trainingsbeispiele haben, die in einem Koordinatensystem eingetragen sind, wie es bei den Schwertlilien der Fall war. Wir haben Punkte aus zwei Klassen, blau und orange.



In der ersten Reihe der Abbildung sehen wir zunächst die Trainingsdaten. Es gibt viele verschiedene Möglichkeiten, wie wir die Trainingsdaten klassifizieren können. In der Abbildung habe ich drei Möglichkeiten eingetragen: eine Gerade, eine einfache gebogene Kurve, und eine sehr geschwungene, mäandernde Kurve. Welche dieser Funktionen sollten wir wählen, wenn wir die “Hoffnung” erfüllen wollen, dass die Funktion auch für zukünftige Beispiele gut funktioniert?

Betrachten wir zunächst die Trainingsfehler dieser drei Klassifikationsregeln. Bei der ersten Klassifikationregel, der Gerade, liegen 7 blaue und 8 orange Punkte auf der jeweils falschen Seite, sie werden also falsch klassifiziert. Zusammen sind das 15 der insgesamt 100 Trainingspunkte, der Trainingsfehler beträgt also 15/100, oder 15%. Die zweite Klassifikationsregel hat einen kleineren Trainingsfehler, etwa 7/100 oder 7%. Die dritte Klassifikationsregel (mäandernde Kurve) ist extra so gewählt worden, dass sie gar keinen Trainingspunkt falsch klassifiziert, sie hat also 0/100 oder 0% Trainingsfehler.

Aber nun interessieren wir uns ja nicht für den Trainingsfehler, den eine Klassifikationsregel auf den schon bekannten Punkten macht. Wir wollen wissen, wie gut die Regel für neue, noch nicht gesehene Punkte funktioniert. Um das auswerten zu können, brauchen wir eine unabhängige “Testmenge” an Punkten, die zur Trainingszeit noch nicht bekannt waren. In unserem Beispiel zeigt die zweite Reihe der Abbildung eine solche Testmenge. Die dort gezeigten Punkte stammen von der gleichen Population wie die Trainingspunkte, sind aber zur Trainingszeit noch nicht bekannt gewesen. Nun können wir für jede der Klassifikationsregeln den “Testfehler” auswerten: wie viele Fehler macht die Regel auf den neuen Testpunkten? Im gewählten Beispiel können wir sehen, dass die Gerade einen Testfehler von 8% macht (4 orange Punkte und 4 blaue Punkte werden falsch klassifiziert), die zweite Klassifikationsregel macht einen Testfehler von 5% (3 blaue und 2 orange Punkte sind falsch), und die mäandernde Regel macht sogar einen Testfehler von 11% (9 blaue und 2 orange Punkte).

Bei diesem Beispiel können wir sehen, dass ein guter Trainingsfehler nicht immer auch auf einen guten Testfehler schließen läßt. Bei der mäandernden Funktion haben wir einen perfekten Trainingsfehler von 0%, aber der Testfehler ist hoch. Hier liegt ein Phänomen vor, dass wir "Overfitting" nennen: die Funktion, die gelernt worden ist, ist zu komplex. Sie passt sich allen zufälligen Ausreißern in den Datenpunkten sehr genau an. Dadurch passt sie aber nicht mehr so gut zu den Testpunkten, denn hier liegen ja andere Punkte mit etwas anderen Unregelmäßigkeiten vor. Das gegensätzliche Phänomen ist das "Underfitting", das bei der Gerade passiert. Ganz offensichlich ist eine Gerade nicht geeigent, die Daten gut zu klassifizieren, und zwar weder die Trainings- noch die Testdaten. Sie ist nicht komplex genug, die zugrunde liegende Population zu beschreiben. Die mittlere, sanft geschwungene Funktion scheint hier einen besseren Mittelweg gefunden zu haben. Sie scheint "komplex genug", um die Daten adäquat zu beschreiben, aber passt sich nicht allen kleinen Abweichungen an. Sie hat einen moderaten Trainingsfehler und einen moderaten Testfehler. Welche Funktion würden wir also wählen? Die Mittlere! Denn sie hat den kleinsten Testfehler, und alleine darauf kommt es an.

Mathematische Garantien für viele Lernverfahren

Was wir gerade an einem Beispiel erläutert haben, kann man natürlich auch systematisch untersuchen: welche Funktion sollte ein Lernverfahren auswählen, damit am Ende der Testfehler möglichst klein ist? Die Wissenschaft des maschinellen Lernens hat hierzu seit mehr als 30 Jahren viele grundlegende Erkenntnisse gewonnen. Es wurde untersucht, unter welchen Umständen man den Testfehler eines Lernverfahrens gut kontrollieren kann, und **für viele Lernverfahren gibt es beweisbare mathematische Aussagen darüber, unter welchen Umständen sie "gut" funktionieren.** Ähnlich wie in der Statistik lassen diese Aussagen es jedoch nicht zu, hundertprozentige Garantien für einen konkreten Datensatz zu geben. Die Aussagen sind immer mit Unsicherheit behaftet: "Mit hoher Wahrscheinlichkeit ist der Testfehler der gefundenen Funktion klein". Das liegt aber in der Natur der Sache, genauso wie man bei einer medizinischen Behandlung den Erfolg nicht mit Sicherheit voraussagen kann, sondern nur "mit hoher Wahrscheinlichkeit".
Ich will an dieser Stelle nicht weiter darauf eingehen, wie genau die beweisbaren Garantien aussehen, denn das wird dann doch etwas komplizierter. Was ich Ihnen aber als wichtige Nachricht mit auf den Weg geben will: sehr viele Verfahren des maschinellen Lernens sind theoretisch sehr gut verstanden. Sie sind in der Praxis kontrollierbar, und man kann Garantien darüber abgeben, wie verlässlich ihre Ergebnisse sind. Die Wissenschaft kann ziemlich genau charakterisieren, wann die "Hoffnung" auf ein gutes Lernergebnis berechtigt ist und wann eher nicht.

7 Neuronale Netze

Nun wird es Zeit, dass wir uns einem anderen Zweig des maschinellen Lernens zuwenden, den neuronalen Netzen. Diese Verfahren sind verantwortlich für viele der Durchbrüche, die in den letzten Jahren im Bereich des maschinellen Lernens erzielt worden sind, insbesondere bei Bild- und Spracherkennung. Das Vorbild für künstliche neuronale Netze ist das menschliche Gehirn. Es besteht

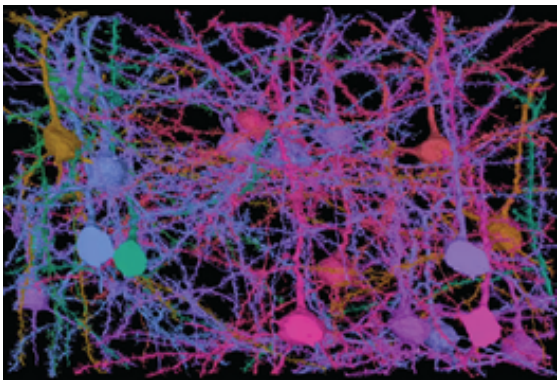
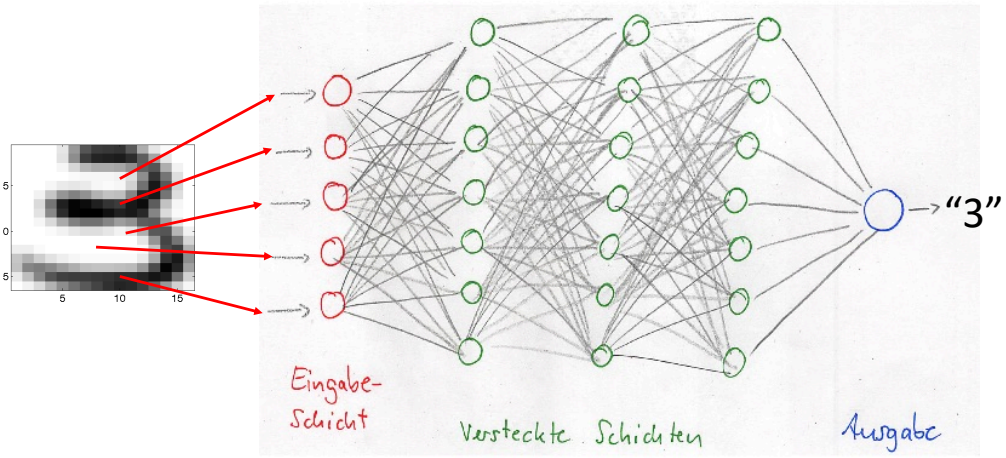


Abbildung 4: Aufnahme eines neuronalen Netzwerkes im Gehirn (copyright: Andreas Tolias Lab, Baylor College of Medicine)

aus vielen Zellen — “Neuronen” — die miteinander kommunizieren können. Jede der Zellen ist mit anderen Zellen durch “Kanäle” verbunden. Durch diese Kanäle gelangen elektrische Signale in eine Zelle hinein. Sobald die Summe der eingehenden Signale einen bestimmten Schwellwert überschreitet, sendet die Zelle selbst ein neues Signal zu anderen Zellen, mit denen sie verbunden ist. Ein ganzer Wissenschaftsbereich, die Neurowissenschaft, beschäftigt sich damit, wie das Gehirn es schafft, basierend auf solch einem einfachen Mechanismus all die Leistungen zustande zu bringen, die die kognitiven Fähigkeiten von Menschen und Tieren ausmachen. Umgekehrt hat die Informatik schon seit den 1950er Jahren versucht, das grundlegende Prinzip des Gehirns in Computern nachzuahmen und war damit bis zu einem gewissen Grade auch erfolgreich. Die bahnbrechenden Durchbrüche jedoch sind erst vor wenigen Jahren erzielt worden.

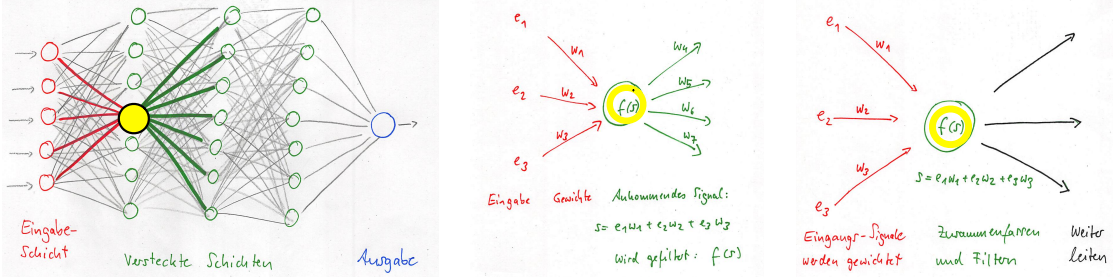
Wie künstliche neuronale Netze rechnen können

Im Prinzip ist ein künstliches neuronales Netz ein kompliziertes Konstrukt, mit dem man Eingaben in Ausgaben umwandeln kann. Ein solches Netz besteht aus vielen “Zellen”, die durch Kanten miteinander verbunden sind. Die Zellen sind in “Schichten” angeordnet.



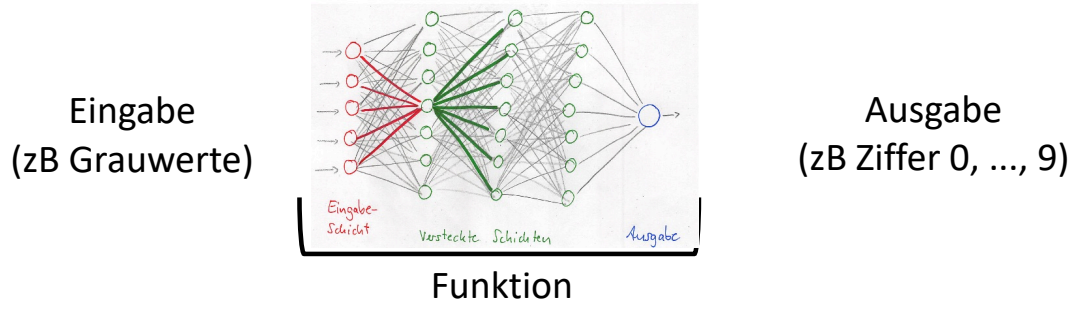
Links im Bild sehen wir die Eingabeschicht. Sie dient dazu, eine Eingabe einzulesen, zum Beispiel die Grauwerte einer handgeschriebenen Ziffer. Diese Eingabe wird dann durch viele weitere Schichten von links nach rechts weitergeleitet: Jede Zelle erhält Signale von ihren linken Nachbarn, führt eine eigene Berechnung durch, und gibt das Ergebnis an ihre rechten Nachbarn weiter. Am Ende erreichen wir die “Ausgabeschicht”, in der das Gesamtergebnis der Berechnung des neuronalen Netzes erscheint.

Stellen wir uns wieder das Beispiel der Erkennung von handgeschrieben Ziffern vor. Die Ziffer wird durch 256 Grauwerte beschrieben. Unser neuronales Netz bauen wir nun so, dass die Eingabeschicht genau 256 Zellen hat. Jede Zelle bekommt den Grauwert von einem der Pixel des Bildes als Eingabe, wobei der Grauwert wie oben beschrieben eine Zahl zwischen 0 und 1 ist. Um zu verstehen, wie die Berechnung des Netzes nun vonstatten geht, picken wir uns als Beispiel eine Zelle aus dem Netz heraus: die gelb markierte Zelle in der folgenden Abbildung (links).



Die roten Linien deuten die Eingabekanäle der Zelle an, die grünen Linien die Ausgabekanäle. Betrachten wir nun die Zelle und ihre Kanäle mal etwas genauer, siehe die mittlere Abbildung. Durch jeden Eingabekanal erhält die Zelle eine Zahl als Eingabe (zum Beispiel, den Grauwert eines Pixels, oder auch die Ausgabe einer Zelle der vorigen Schicht im neuronalen Netz). Diese Zahl ist in der mittleren Abbildung mit e_1, e_2, und e_3 bezeichnet: zum Beispiel steht e_1 für den Grauwert des ersten Pixels, e_2 für den des zweiten Pixels, und so weiter. Weiterhin hat jeder der Ein- und Ausgabekanäle ein bestimmtes "Gewicht", das in der Abbildung mit $w_1, w_2, ...$ bezeichnet ist. Das Gesamtsignal, das an der gelben Zelle ankommt, ist nun einfach die gewichtete Summe der verschiedenen Eingangskanäle: $s = w_1 \cdot e_1 + w_2 \cdot e_2 + w_3 \cdot e_3$. (So ähnlich macht es eine Zelle im Gehirn auch: die einzelnen Eingabesignale sind elektrische Impulse, und je stärker ausgeprägt der Verbindungskanal zwischen zwei Zellen ist, desto stärker kommt der entsprechende Impuls bei der Zelle an. Die Zelle reagiert auf den Gesamtimpuls aller eingehenden elektrischen Signale, das heißt sie summiert alle eingehenden Signale auf.) Nun nimmt die Zelle das eingehende Gesamtsignal s und bearbeitet es weiter; wir sagen, sie "filtert" das Signal. Zum Beispiel könnte die Zelle auswerten, ob das Signal größer als ein bestimmter Mindestwert ist. Wenn ja, leitet sie das Signal weiter: sie schickt ihr gefiltertes Signal $f(s)$ über alle ihre Ausgangskanäle an ihre rechten Nachbarn weiter. Diese verfahren nun genauso: sie summieren wiederum ihre gewichteten Eingabesignale auf, filtern sie, und leiten ihr Ergebnis an ihre Nachbarn weiter. Dieser Prozess wird so von jeder Zelle ausgeführt, bis das Signal von ganz links (Eingabeschicht) nach ganz rechts (Ausgabeschicht) gewandert ist. Im Beispiel der Ziffern besteht die Ausgabeschicht aus einer einzelnen Zelle, die nur einen Ausgabekanal hat. Diese Zelle verwandelt ihr Eingangssignal in eine Ausgabe, die eine Ziffer zwischen 0 und 9 ist (wir können uns zum Beispiel vorstellen, dass die Zelle ihr Eingangsignal einfach rundet). Diese Ausgabe-Ziffer ist nun die Antwort des neuronalen Netzes auf das eingegebene Bild.

Vergessen wir für einen Moment, wie genau die neuronalen Netze intern aufgebaut sind. Alles, was wir im Folgenden wissen müssen, ist folgende Erkenntnis: ein neuronales Netz ist nichts anderes als ein sehr komplizierter Mechanismus, eine Funktion, mit dem wir ein Eingabesignal (die Grauwerte eines Bildes zum Beispiel) in ein Ausgabesignal (wie zum Beispiel eine Ziffer zwischen 0 und 9) verwandeln können.

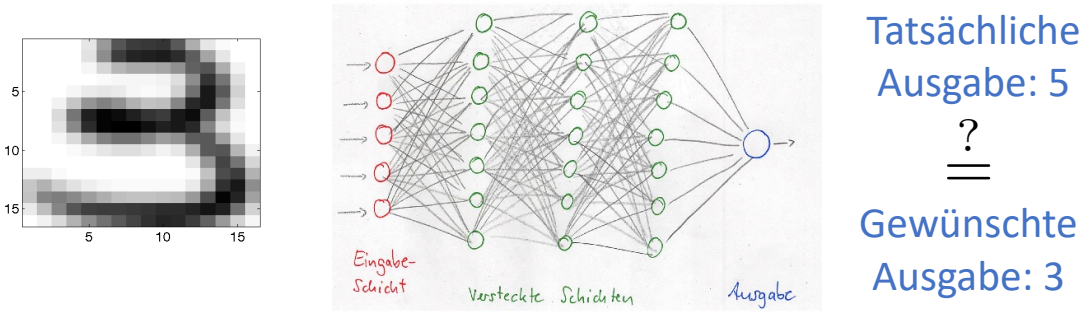


Das Training von neuronalen Netzen

Bisher haben wir gesehen, dass ein vorgegebenes neuronales Netz eine Funktion darstellt, die durch eine Berechnung eine Eingabe in eine Ausgabe überführen kann. Der Trick bei selbst-lernenden neuronalen Netzen ist nun, diese Funktion anhand von Trainingsbeispielen so einzustellen, dass das Netz zu den Eingaben der Trainingsbeispiele auch die richtigen Ausgaben produziert. Denken wir an die lineare Klassifikation der Lilien zurück. Hier war die Frage, welche Gerade wir genau nehmen sollen, um die Trainings-Lilien richtig bestimmen zu können. Wir mussten also die Steigung und Verschiebung der Gerade so anpassen, dass die Gerade einen kleinen Trainingsfehler erzielt. Das Gleiche müssen wir nun bei neuronalen Netzen erreichen: Wir müssen die Kantengewichte (also die Verbindungstärken zwischen den Neuronen) so anpassen, dass das neuronale Netz zu den Trainings-Eingaben die gewünschten Ausgaben produziert.

Das funktioniert wie folgt. Am Anfang des Trainings starten wir mit einem Netz, dem wir zufällige Kantengewichte zugewiesen haben — das Netz ist also komplett "unwissend" und produziert völlig

zufällige Ausgaben. Dann zeigen wir dem Netz unser erstes Trainingsbeispiel, zum Beispiel eine handgeschriebene Ziffer 3. Das Netz liest die Grauwerte der Pixel ein, führt seine Berechnung von links nach rechts durch und gibt sein errechnetes Ergebnis als Ausgabe aus. Wir prüfen nun, ob diese ausgegebene Ziffer die richtige ist, in unserem Beispiel also eine 3. Falls ja, verändern wir nichts und gehen zum nächsten Trainigsbeispiel über. Falls nein, müssen wir die Gewichte des Netzes anpassen. Das passiert nun wieder mit einem bekannten mathematischen Optimierungsverfahren, einem sogenannten "Gradientenabstieg". Dabei verändert man die Kantengewichte auf systematische Art und Weise, so dass das Ergebnis des veränderten Netzes bei dem betrachteten Trainingsbeispiel näher an der gewünschten Ausgabe liegt.



Kommt Ihnen das bekannt vor? Im Falle des Briefumschlag-Computers hatten wir ein ganz ähnliches Lernverfahren. Der Computer startete mit einer zufälligen Spielstrategie. Wir spielten ein Trainings-Spiel nach dem anderen gegen den Computer. Wenn der Computer gewonnen hatte, veränderten wir nichts. Wenn der Computer verloren hatte, passten wir die Spielstrategie leicht an (indem wir einzelne Zettel vernichteten). Jedes Trainings-Spiel gegen den Briefumschlag-Computer entspricht also der Eingabe einer handgeschriebenen Ziffer in das neuronale Netz, und das Vernichten der Zettel entspricht dem Verändern der Kantengewichte.

Wieder hat unser Lernproblem die üblichen Zutaten. Die Aufgabe, die das Netz lernen soll, ist klar definiert; zum Beispiel soll das Netz handgeschriebene Ziffern erkennen. Der Suchraum, in dem nach einer Lösung gesucht wird, ist fest vorgegeben: Den Aufbau des Netzwerkes legen wir am Anfang ein für alle mal fest — also wie viele Schichten es hat, wie viele Zellen jede Schicht hat, und welche Zellen untereinander verbunden sind. Das einzige, was wir nicht festlegen, sind die Gewichte der Verbindungskanäle — diese Gewichte sind es, die der Algorithmus selber lernen soll. Der Suchraum des neuronalen Netzes besteht also aus allen Funktionen, die das neuronale Netz durch Veränderung der Kantengewichte darstellen kann. Auch die Bewertungsfunktion ist klar: wir messen den Traingsfehler, um zu beurteilen, ob ein neuronales Netz eine bestimmte Menge an Trainingsbeispielen gut klassifiziert oder nicht. Das Lernverfahren selbst besteht nun darin, Trainingsbeispiele vorzuführen, und je nach Antwort des neuronalen Netzes die Kantengewichte anzupassen.

Neuronale Netze in der Praxis: mächtig und problematisch zugleich

In den letzten Jahren hat sich gezeigt, dass neuronale Netze ein äußerst **mächtiges Werkzeug** darstellen, um bestimmte Probleme des maschinellen Lernens zu lösen — zum Beispiel bei der Bilderkennung oder der automatischen Übersetzung von Texten sind sie unschlagbar. Wann immer es Lernprobleme gibt, bei denen sehr große Mengen an Daten eingehen, für die es keine guten Modellbeschreibungen gibt, versuchen die Analyst*innen, neuronale Netze darauf zu trainieren. Viele der Durchbrüche, von denen man immer wieder in der Zeitung lesen kann, gehen auf neuronale Netze zurück.

Aus praktischer Sicht hat es sich jedoch auch gezeigt, dass neuronale Netze sehr unangenehme Nebeneffekte haben könnnen. Eines davon betrifft die sogenannte "Robustheit". Darunter versteht man die Eigenschaft eines Systems, nicht durch kleine Detailveränderungen sein komplettes Ver-

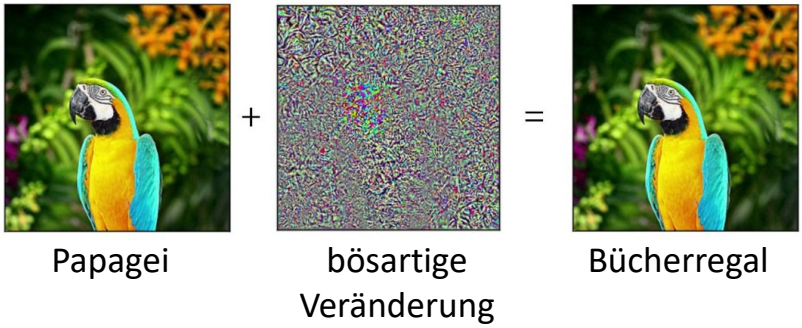


Abbildung 5: Links: Bild eines Papageis, das vom neuronalen Netz korrekt erkannt wird. Mitte: bösartige Veränderung, die wir auf das Bild "drauf addieren". Rechts: das Ergebnis des veränderten Bildes sieht für Menschen genau wie vorher aus, wir können keinen Unterschied erkennen. Das neuronale Netz jedocht glaubt nun, dass das Bild ein Bücherregal zeigt.

halten über Bord zu werfen. Stellen Sie sich vor, wir haben ein neuronales Netz darauf trainiert, Tiere und Gegenstände auf Bildern zu erkennen. Zeigen wir dem neuronalen Netz das Bild eines Papageis (siehe Abbildung 5), gibt es korrekt das Ergebnis "Papagei" zurück. Nun stellen wir uns einen Angreifer vor, der unser Netz in die Irre führen will. Er verändert das Bild auf ganz subtile Weise, indem er die Farben einzelner Pixel leicht verschiebt. Das Ergebnis davon ist das Bild auf der rechten Seite. Als Mensch können wir keine Veränderung zum linken Bild erkennen, und würden das rechte Bild offensichtlich immer noch als "Papagei" klassifizieren. Das neuronale Netz jedoch wird durch die subtile Änderung komplett in die Irre geführt und glaubt nun, ein "Bücherregal" vor sich zu haben. Ein solches Verhalten ist das Gegenteil von robust — eine winzige Änderung an einzelnen Farbwerten, die für Menschen nicht einmal sichtbar sind, können dazu führen, dass das neuronale Netz komplett falsch klassifiziert.

Ich möchte an dieser Stelle betonen, dass dieses Beispiel nicht bedeutet, dass neuronale Netze an sich schlecht funktionieren. Die "subtilen Veränderungen", von denen oben die Rede ist, muss man sehr genau konstruieren, und sie werden im Alltag nicht durch Zufall entstehen. Trotzdem ist es beunruhigend zu wissen, dass Angreifer im Prinzip Möglichkeiten hätten, ein neuronales Netz zu attackieren und zu Fehlern in der Klassifikation zu verleiten. Man stelle sich nur vor, dass ein selbstfahrendes Auto Verkehrsschilder falsch interpretiert, nur weil ein Angreifer auf bestimmte Weise kleine, kaum sichtbare Aufkleber auf die Schilder geklebt hat.

Neuronale Netze in der Theorie: schwer zu fassen

Nun wäre es natürlich sehr wünschenswert, eine mathematische Theorie zu entwickeln, die zu Garantien für Vorhersagen von neuronalen Netzen führen könnte. Das hat sich jedoch aus verschiedenen Gründen als erstaunlich schwierig herausgestellt. Insbesondere mussten die Wissenschaftler*innen lernen, dass die klassischen Methoden, mit denen für andere Lernverfahren Garantien bewiesen werden, auf neuronale Netze nicht anwendbar sind und nie sein werden. Das bedeutet nicht, dass es unmöglich ist, Garantien für neuronale Netze zu beweisen. Wir müssen dafür aber einen komplett neuen, mathematischen Werkzeugkasten entwickeln. Es gibt in diesem Zusammenhang erste interessante und auch überraschende Einsichten, aber wie lange es dauern wird, bis wir konkrete Garantien beweisen können und wie stark diese Garantien dann ausfallen, das kann man im Moment nicht absehen.

8 AlphaGo: ein Computer schlägt den Weltmeister

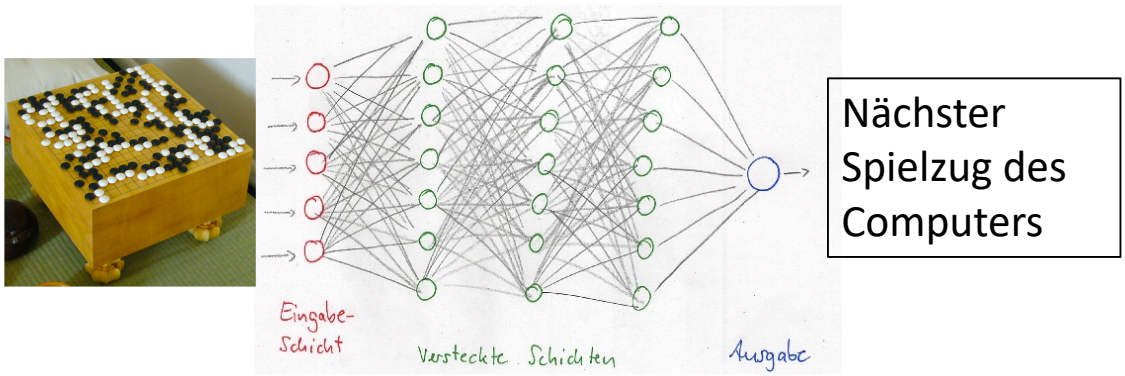
Es gab in den letzten Jahren viele interessante Neuigkeiten aus dem Bereich des maschinellen Lernens, aber kaum ein Durchbruch hat so viel Aufmerksamkeit erregt wie "AlphaGo". Go ist ein asiatisches Brettspiel, bei dem zwei Spieler*innen gegeneinander antreten. Ähnlich wie Schach ist es ein komplexes Spiel, das von Menschen in jahrelangem Training erlernt wird. Laut Experten

erfordert Go jedoch viel mehr Flexibilität und Kreativität als Schach, das in deutlich stärker vorhersehbaren Bahnen abläuft.

Sehr gute Schachcomputer gibt es schon lange, und 1996 hat der Großcomputer Deep Blue den damaligen Schachweltmeister Garry Kasparov geschlagen. Der Mechanismus, durch den das möglich war, beruhte zu großen Teilen auf schierer "Rechenkraft": der Computer konnte sehr viele mögliche Spielstellungen auswerten, die ausgehend von der derzeitigen Stellung innerhalb mehrere Spielzüge erreichbar waren, und dadurch einen Vorteil gegenüber menschlichen Spielern erreichen. Das war damals eine sehr beeindruckende Leistung, hat jedoch mit maschinellem Lernen nichts zu tun. Das Brettspiel Go hatte sich einem solchen Ansatz entzogen — es ist den Informatikern einfach nicht gelungen, es durch schiere Rechenkraft zu knacken. Dann im Jahr 2016 kam der Durchbruch mit Hilfe des Algorithmus AlphaGo, der eine ganz andere Strategie verfolgt: er beruht auf maschinellem Lernen.

Der Mechanismus von AlphaGo

Grob vereinfacht funktioniert das Ganze so: Eine Brettstellung von Go wird durch einen Matrix von 0-1-Werten kodiert, ganz ähnlich zu den Grauwerten der Pixel in einem Bild. Solch eine Brettstellung kann ein neuronales Netz als Eingabe erhalten. Die gewünschte Ausgabe des Netzes ist, welcher Zug als nächster gemacht werden soll. Am Anfang startet das Netz mit zufälligen Kantengewichten und macht zufällige Züge, die sicher nicht zum Ziel führen. Nun wurde das Netz ein erstes mal trainiert. Zunächst benutzte man tausende dokumentierte Spiele von menschlichen Expert*innen. Man trainierte das Netz darauf, in einer vorgegebenen Spielsituation die gleiche Entscheidung wie der menschliche Experte zu treffen. Danach ließ man zwei so trainierte neuronalen Netze gegeneinander spielen, und nach jedem Spiel wurden wiederum die Parmeter angepasst. In dieser zweiten Trainingsphase spielten die Netze immer besser, und am Ende war es dann so weit, dass das "AlphaGo" getaufte System den damaligen Go-Weltmeister schlagen konnte.



Was ich oben beschrieben habe, ist eine sehr vereinfachte Darstellung – natürlich gab es viele Details, die beim Entwurf von AlphaGo zu berücksichtigen waren, und tolle Ideen, ohne die der Erfolg nicht möglich gewesen wäre. Trotzdem ist für mich die Grundessenz: ganz am Ende funktioniert AlphaGo so ähnlich wie unser Briefumschlagcomputer! Das System hat eine Architektur (die Briefumschläge im einen Fall, das neuronale Netz im anderen Fall), die einen Spielmechanismus ermöglicht: gegeben eine Spielsituation, schlägt der Computer einen neuen Zug vor. Wir trainieren den Computer, indem wir ihn Trainigsspiele spielen lassen und auswerten, ob er gewonnen oder verloren hat. Entsprechend werden die Parameter des Computers nach jedem Trainingsspiel leicht verändert. Durch viele Trainingsspiele lernt der Computer dann in beiden Fällen, gute Spielzüge vorzuschlagen.

Nach der Veröffentlichung von AlphaGo wurde der Algorithmus im Jahr darauf noch einmal verändert und AlphaGoZero genannt. Der entscheidende Unterschied war nun, dass in der Eingangsphase des Trainings keine menschlichen Expertenspiele mehr verwendet wurden. Nachdem die Architektur des Computers festgelegt war, startete er mit zufälligen Parametern und spielte nur gegen sich selbst. Nur die Spielregeln waren natürlich vorgegeben. Das Ergebnis war genauso gut wie bei AlphaGo – ein Computer, der die besten Go-Spieler schlagen konnte.

9 Ist das jetzt doch intelligent? Oder kreativ? Versteht der Computer, was er da tut?

Insbesondere die Variante AlphaGoZero hat eine große Diskussion angefacht: der Computer kann ja nun ganz von alleine — ohne auf menschliches Expertenwissen zurückzugreifen — ein meisterhafter Go-Spieler werden. Er entdeckt also ganz von alleine, was erfolgversprechende Strategien sind, ohne jedes Zutun des Menschen. Heißt das nun vielleicht doch, dass **Intelligenz** im Spiel ist?

Zu dieser Frage gibt es viele verschiedene Meinungen, und um eine tiefere Diskussion zu führen, müssten wir erst einmal überlegen, was wir eigentlich unter Intelligenz verstehen wollen (diese Frage füllt viele Bücher und hat keine offensichtliche Antwort). Meine persönliches Gefühl ist hier das gleiche wie beim Briefumschlag-Computer: Wir haben einen Algorithmus gefunden, mit dem ein Computer auf effiziente Art und Weise einen riesigen Suchraum nach guten Spielstrategien durchforsten kann. Trotzdem kann dieser Computer keine andere Aufgabe lösen oder lernen, als Go zu spielen, und das kommt mir doch sehr eingeschränkt vor. Ich möchte die Diskussion aber an dieser Stelle nicht vertiefen.

Man kann auch die Frage nach **Kreativität** stellen. Wenn menschliche Experten von AlphaGo gespielte Spielzüge nachvollziehen, sind sie oft beeindruckt: es ist dem Computer gelungen, ganz neue, bisher nicht bekannte Spielstrategien zu finden. Er macht "kreative" Züge, auf die ein menschlicher Experte in dieser Spielsituation nicht gekommen wäre. Für mich ist das jedoch nichts, was ich besonders erstaunlich finde oder was mich umtreibt. Der Computer durchsucht den Raum aller möglichen Spielzüge auf ganz andere Weise, als ein Mensch das tut. Da finde ich es nicht verwunderlich, dass er teilweise auch anderere Strategien entwickelt als ein Mensch — die wir wiederum dann verblüffend kreativ finden.

Hat der Computer denn für das Go-Spiel tiefere Einsichten in gute Spielstrategien gewonnen, die darüber hinausgehen, in jeder Situation in seinem internen Lexikon nachzusehen, was nun der beste Zug ist? Hat er ein wie auch immer geartetes **Verständnis** gewonnen? Auch das ist eine Frage, die schwer zu beantworten ist. Wiederum, was heisst hier "verstehen" (dazu gibt es Berge von Literatur in der Philosophie)? Diese Frage nach Verständnis ist auch verknüpft mit der Frage der **Erklärbarkeit**: könnten wir AlphaGo vielleicht eine Erklärung dafür abringen, warum es in einer bestimmten Situation einen bestimmten Zug macht? Hier kommen wir schon näher an die Forschungsfront, denn Erklärbarkeit spielt in vielen Anwendungen des maschinellen Lernens eine wichtige Rolle. Es ist absehbar, dass solche Erklärungen in Zukunft möglich sein könnten. Stellen wir uns vor, wie ein Mensch vielleicht einen Spielzug erklären würde: *"Wenn ich meinen Stein an diese Stelle gesetzt hätte, dann hätte der Gegner in einem der nächsten Züge diese und jene Möglichkeit gehabt, das wollte ich nicht. Daher habe ich meinen Stein dort an diese andere Stelle gesetzt; damit bedrohe ich den Gegener und er muss erst mal darauf reagieren, bevor er mich angreifen kann."* Eine solche Erklärung könnte im Prinzip aus vielen beobachteten Spielen, die von menschlichen Spielern mit Erklärungen kommentiert werden, abgeleitet werden. Es gibt verwandte Themengebiete, in denen so etwas schon funktioniert. Ein Beispiel ist die Klassifikation von Vöglen. Man trainiert ein neuronales Netz, Bilder von verschiedenen Vogelarten zu unterscheiden. Dann trainiert man ein weiteres System, das Erklärungen dazu lernen kann, zum Beispiel: *"Dieser Vogel ist eine Amsel, weil der Körper schwarz und der Schnabel gelb ist. Es ist keine Krähe, denn eine Krähe hätte einen schwarzen Schnabel."*

Sie sehen schon, die Grenzen zwischen Intelligenz, Kreativität, Verstehen und Erklären sind fließend. Eine klare Abgrenzung ist nicht möglich, und wer in die Diskussion über diese Begriffe einsteigen möchte, der kann die vielschichtige Literatur zu diesen Themen in Philosophie und Psychologie nicht außen vor lassen. Für mich ist es am Ende jedoch gar nicht so wichtig, ob wir einen Algorithmus nun als intelligent bezeichnen. Entscheidend finde ich vielmehr zu verstehen, dass alle Algorithmen des maschinellen Lernens auf einfachen Grundprinzipien beruhen, und dass der autonome Roboter, der die Welt übernehmen und die Menschheit übertrumpfen kann, nicht vor der Tür steht, auch nicht ansatzweise. Aber das ist meine persönliche Einschätzung.

10 Ausblick

Ich hoffe, dass es mir in diesem Artikel gelungen ist, das Geheimnis um maschinelles Lernen zumindest ein kleines bisschen zu lüften. Vielleicht verspüren Sie nun eine gewisse Ambivalenz (zumindest mir würde es an dieser Stelle so gehen). Einerseits hört sich ja alles ganz einfach an: Beim maschinellen Lernen lernt ein Computer, eine bestimmte Aufgabe an Hand von Beispielen zu lernen. Der Mechanismus ist im wesentlichen immer gleich: in einem Suchraum wird nach guten Lösungen für die gestellte Aufgabe gesucht. Wie gut eine mögliche Lösung ist, wird anhand von Trainingsbeispielen und weiteren Kriterien bewertet, und durch einen Optimierungsalgorithmus wird dann eine gute Lösung gefunden. Andererseits: ganz so einfach kann es ja irgendwie doch nicht sein, und zugegebenermaßen habe ich natürlich ganze viele Details weggelassen. Das lässt sich aber nicht vermeiden und ist vielleicht auch nicht so schlimm.

Selbst mir geht es manchmal so, dass ich verblüfft bin, wenn ich von neuen, erfolgreichen Anwendungen des maschinellen Lernens lese. Eine wichtige Einsicht, die Sie vielleicht aus meinem Artikel ziehen können, ist die Folgende: Wenn ein Computer etwas Verblüffendes kann, muss nicht immer gleich Intelligenz im Spiel sein. Oft genügen erstaunlich einfache Mechanismen, um ein komplex wirkendes Verhalten hervorzubringen, und man sollte sich als menschlicher Betrachter davon nicht einschüchtern lassen.

Es gibt natürlich ganz viele wichtige Themen, die ich in diesem Artikel nicht diskutiert habe. Insbesondere habe ich im Bereich des maschinellen Lernens nur sogenannte "überwachte Lernverfahren" betrachtet, bei denen in den Trainingsdaten zu jedem Trainingspunkt auch gleich die gewünschte Ausgabe mitgeliefert wird. Zum Beispiel bekommt der Algorithmus bei den Schwertlilien bei jeder Trainings-Blume gesagt, um welche der drei Lilienarten es sich handelt. Im unüberwachten Lernen würde diese Information nicht vorliegen; statt dessen wäre das Ziel, die grobe Struktur der Daten zu untersuchen und zum Beispiel festzustellen, dass es in dem Datensatz mehrere verschiedene Unterarten von Lilien gibt. So etwas könnte zum Beispiel in der Medizin interessant sein, wo man auf diese Art unterschiedliche Ausprägung einer Krankheit feststellen könnte, die vorher so noch nicht bekannt war. Ich will nicht näher darauf eingehen, wie das funktioniert, vielleicht nur so viel: auch hier wird nur mit Wasser gekocht, und die zugrunde liegenden Prinzipien sind gar nicht so weit von den oben vorgestellten Mechansimen entfernt.

Ein Thema, das ich in diesem Artikel ganz außen vor gelassen habe, ist was das maschinelle Lernen eigentlich mit unserer Gesellschaft macht, im Guten sowie im Schlechten. Dieses Thema ist unglaublich wichtig und vielschichtig: Welche Anwendungen des maschinellen Lernens wollen wir in unserer Gesellschaft zulassen und welche nicht? Ist automatische Gesichtserkennung erwünscht, in welchen Situationen? Wollen wir wirklich, dass Entscheidungen über Kreditvergabe, Studienplätze oder Polizeikontrollen von Algorithmen getroffen werden? Welche Vor- und Nachteile sind dabei abzuwägen? Wie können wir verhindern, dass grundlegende ethische Prinzipien verletzt werden, und dafür sorgen, dass diese Entscheidungen "fair" sind und nicht bestimmte Gesellschaftsgruppen benachteiligt werden? Welche Garantien sollten Algorithmen erfüllen, wenn sie medizinische Diagnosen unterstützen? Wie viel sicherer als menschliche Fahrer muss ein selbstfahrendes Auto sein, bevor wir es auf die Straße lassen? Wir stecken schon mittendrin in dieser Diskussion, und die wird uns noch viele Jahre begleiten.

Umso wichtiger, dass Sie als Diskussionsteilnehmer*innen ein grundlegendes Verständnis dafür haben, wie maschinelles Lernen funktioniert. Ich hoffe, dass ich mit diesem Artikel dazu beitragen konnte.