

# Verteilte Systeme für mobiles Maschinelles Lernen

## BACHELORARBEIT

für die Prüfung zum

Bachelor of Science

des Studiengangs Informatik / Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

**Jens Döllmann**

Abgabedatum 30. August 2021

Bearbeitungszeitraum

3 Monate

Matrikelnummer

8876462

Kurs

TINF18B4

Betreuer der Ausbildungsfirma

Florian Seiter

Gutachter der Studienakademie

Gerhard Wolf

## Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema: „Verteilte Systeme für mobiles Maschinelles Lernen“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

---

Ort      Datum

---

Unterschrift

**Zusammenfassung**

**Abstract**

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Was ist Maschinelles Lernen? . . . . .	2
1.2	Die unterschiedlichen Lernmethoden . . . . .	3
1.3	Beschreibung des Problems . . . . .	7

# Kapitel 1

## Einleitung

Hören die meisten Personen die Begriffe „Künstliche Intelligenz“ und „Maschinelles Lernen“ stellen sie sich Roboter vor: treue Diener, die all deine Aufgaben übernehmen oder eine tödliche Arme von superintelligenten Cyborgs im Kampf gegen die Menschheit. Künstliche Intelligenz (kurz KI) ist jedoch schon lange nicht mehr nur eine futuristische Fantasie und Teil von Science-Fiction-Filmen; sie wird bereits großflächig verwendet und findet Einsatz in einer Vielzahl von Anwendungen aus Bereichen in nahe zu allen Teilen der Wirtschaft. Andrew Ng, welcher unter anderem durch seine beliebten Onlinekurse zum Thema Maschinellen Lernen (ML) und Deep Learning (DL) bekannt ist,<sup>1</sup> vergleicht KI mit der neuen Elektrizität.

*„Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don't think AI will transform in the next several years“*

— Andrew Ng (Lynch 2017)

Die Anwendung, mit der Maschinelles Lernen erstmalig öffentliche Aufmerksamkeit erlangte und und noch immer das tägliche Leben vieler Menschen beeinflusst, stammt aus den 90er-Jahren: Die Rede ist von dem E-Mail-Spamfilter (Géron 2017, S. 1). Es handelt sich um eine scheinbar einfache Aufgabe, welche mit traditioneller Programmierung aber dennoch nur schwer gelöst werden kann. Ein klassisches Programm besteht aus vielen statischen Regeln, im Fall des Spamfilters können diese dazu dienen, auffällige Schlüsselwörter

---

<sup>1</sup><https://www.coursera.org/instructor/andrewng>

und Satzteile zu erkennen (wie „Kreditkarte“, „kostenlos“, „kaufen“ oder andere Merkmale). Ein statisches Programm ist nicht gut dafür geeignet, ein dynamisches Problem zu beschreiben. Die Absender der Spammessages könnten erkennen, welche E-Mails blockiert werden und diese leicht abändern. Arbeiten sie so durchgehend um das Problem herum, müssen immer wieder neue Regeln im Programm aufgenommen werden. Ein Spamfilter hingegen, welcher auf Maschinellen Lernen basiert, kann diese Regeln selbstständig erkennen und automatisch anpassen wenn es eine Veränderung erkennt. Der Algorithmus durchläuft hierfür eine Trainingsphase und verwendet dabei Beispiel E-Mails (z. B. diese, die vom Benutzer markiert wurden) und kann so lernen Spammessages zu unterscheiden. Diese Trainingsdaten werden als Trainingsdatensatz (engl. *train set*) bezeichnet. Es folgt ein Programm, welches weitaus kürzer, einfacher zu verwalten und wahrscheinlich auch präziser ist.

## 1.1 Was ist Maschinelles Lernen?

Maschinelles Lernen ist die Wissenschaft (und Kunst) der Programmierung, die es dem Computer ermöglicht, von Daten zu lernen. Arthur Samuel und Tom Mitchell haben im Jahr 1959 und 1997 eine allgemeine und formale Definition gegeben (Géron 2017, S. 2):

*„Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.“*

— Arthur Samuel, 1959

*„A computer program is said to learn from experience  $E$  with respect to some task  $T$  and some performance measure  $P$ , if its performance on  $T$ , as measured by  $P$ , improves with experience  $E$ .“*

— Tom Mitchell, 1997

Betrachten wir erneut das Beispiel des Spamfilters, dann ist die Aufgabe  $T$  das Markieren von Spam, die Erfahrung  $E$  sind die Trainingsdaten und der Bewertungsmaßstab  $P$  bleibt zu definieren. Es könnte zum Beispiel der Anteil der als richtig markierten E-Mails verwendet werden. Dieser Maßstab heißt Genauigkeit und wird häufig verwendet wenn es darum geht eine Klassifizierungsaufgabe zu bewerten.

## 1.2 Die unterschiedlichen Lernmethoden

Maschinelles Lernen kann genutzt werden, um eine Vielzahl von Problemen zu lösen. Dies macht es zu einem mächtigen Werkzeug, aber auch eines, in dem es schwierig sein kann, den Überblick zu behalten. Da es viele verschiedene Systemen und Lösungsansätze gibt, ist es sinnvolle diese in grobe Kategorien zu unterteilen. Ein System kann unterschieden werden, basierend auf den folgenden Kriterien (Géron 2017, S. 7):

- Trainiert das System mit oder ohne menschlicher Betreuung, dann ist die Rede von überwachten und unüberwachten Lernen.
- Kann das System inkrementell mit einer Teilmenge der Trainingsdaten lernen oder verwendet es die gesamten Daten. Dieser Unterschied nennt sich Online- vs. Batch-Learning.
- Funktioniert das System indem es lediglich neue Daten mit alten vergleicht oder kann es Muster erkennen, um beispielsweise einen Datensatz in Klassen zu unterteilen. Diese Lernmethoden nennen sich instanzbasiertes und modellbasiertes Lernen.

Keine der eben beschriebenen Kriterien schließen sich aus. In einem echten System kann es sinnvoll sein, die verschiedenen Eigenschaften zu kombinieren, um das gewünschte Ergebnis zu erzielen. Der zuvor beschriebene E-Mail-Spamfilter ist ein klassisches überwachtes und instanzbasiertes Lernsystem. Es ist überwacht, da der Typ der E-Mail (z. B. durch Benutzereingabe) bekannt ist und instanzbasiert da der Algorithmus neue E-Mails mit alten vergleicht und das Ergebnis so wählt, worin die größte Ähnlichkeit besteht. Die Ergebnisse bei überwachten Lernen können wie im Falle des Spamfilters vom Benutzer stammen oder durch Naturgesetzte und Expertenwissen gegeben sein. Diese Daten werden im System als Label bezeichnet. Wir können nun die einzelnen Lernmethoden etwas genauer untersuchen.

**Klassifizierung versus Regression** Das bisherige Spamfilter Beispiel ist ein typisches Klassifizierungsproblem. Zu einer Menge von eine Eingabewerten (auch genannt Features) soll die Wahrscheinlichkeit bestimmt werden, dass diese Feature zu einer bestimmten Klasse gehören. Was ist aber wenn keine Wahrscheinlichkeiten prognostiziert werden sollen, sondern numerische Werte wie der Preis eines Autos? Dieses Problem nennt sich Regression. Ein Beispiel soll zeigen wie das Prinzip funktioniert. Es soll ein einfaches Modell

trainiert werden welches eine lineare Funktion modelliert. Als ersten Schritt müssen die Trainingsdaten generiert werden, der folgende Python Code zeigt diesen Vorgang mithilfe von NumPy:

```
import numpy as np
rng = np.random.default_rng(42)
x = 6 * rng.random((100, 1))
y = 5 + 3 * x
```

NumPy ist eine Python Bibliothek, welche effizientes Rechnen mit multidimensionalen Daten ermöglicht (z. B. der Vektor- und Matrixmultiplikation).<sup>2</sup> Operationen in NumPy werden immer elementweise durchgeführt. Der folgende Programmausschnitt zeigt dieses Verhalten:

```
>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])
>>> a * 3
array([3, 6, 9])
```

CPUs und GPUs sind sehr gut darin, diese Art der Berechnungen durchzuführen, was NumPy sehr viel schneller macht als die Verwendung einer expliziten Schleife. Die Technik, nicht mehr mit skalaren Werten zu rechnen, nennt sich Vektorisierung und kann die Laufzeit eines rechenintensiven Algorithmus, wie sie in der KI zum Einsatz kommen, drastisch verringern.

*„Vectorization is the process of converting an algorithm from operating on a single value at a time to operating on a set of values (vector) at one time“*

— (Khartchenko 2018)

Wir können nun die zuvor generierten Daten visualisieren und verwenden hierfür eine weitere Python Bibliothek mit dem Namen Matplotlib:<sup>3</sup>

```
import matplotlib.pyplot as plt
plt.plot(x, y, "b.")
plt.xlabel("$x_1$")
plt.ylabel("$y = 5 + 3x$")
plt.axis([0, 6, 5, 25])
plt.show()
```

Die hieraus resultierende Grafik ist in Abbildung 1.1 zu sehen.

---

<sup>2</sup><https://numpy.org/doc/stable/user/what-is-numpy.html#what-is-numpy>

<sup>3</sup><https://matplotlib.org/stable/index.html>



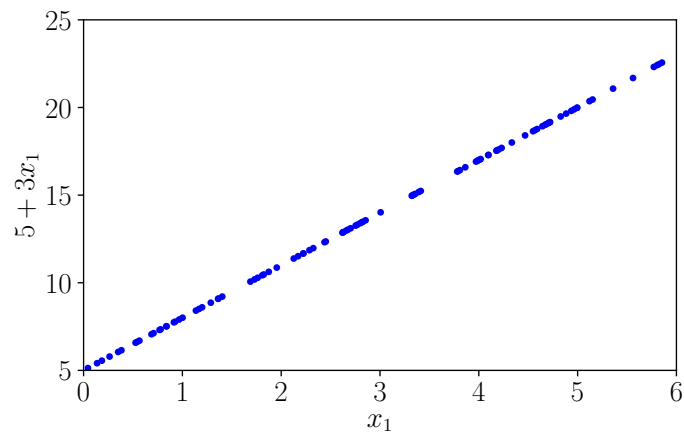


Abbildung 1.1: Die generierte Daten

Die Daten sehen sehr gerade aus, damit das Problem nicht zu einfach ist addieren auf Seiten der  $y$  Werte zufälliges Rauschen.

```
y += 3 * rng.standard_normal((100, 1))
```

Dies sollte das Problem ein wenig realistischer gestalten. Die so modifizierten Daten sind in Abbildung 1.2 zu sehen. Das Regressionsproblem besitzt eine Eingabe  $x_1$  und

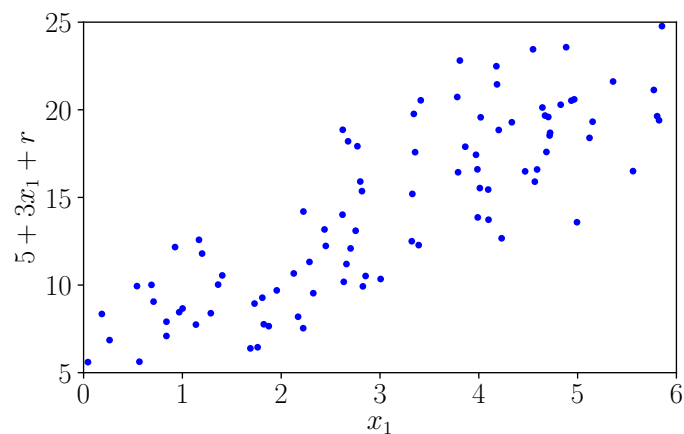


Abbildung 1.2: Die generierte Daten plus zufälliges Rauschen

eine Ausgabe  $y$ . Ein echtes Problem besteht in der Regel aus mehreren Eingaben und manchmal aus mehreren Ausgaben. Mehrdimensionale Daten lassen sich allerdings nur

schwierig visualisieren. Ziel des Modells ist es nun eine Gerade zu finden, welche die Daten möglichst gut beschreibt. Die Geradengleichung hat die folgende Form:

$$\begin{array}{ll} 1 \text{ Feature} & \hat{y} = \theta_0 + \theta_1 \cdot x_1 \end{array} \quad (1.1)$$

$$\begin{array}{ll} n \text{ Features} & \hat{y} = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \dots + \theta_n \cdot x_n \end{array} \quad (1.2)$$

Die griechischen Buchstaben  $\theta_i$  beschreiben die lernbaren Parameter und der Funktionswert  $\hat{y}$  (gesprochen „y-Hut“) beschreibt die Prognose. Das Training kann in Python mit nur wenigen Zeilen Code durchgeführt werden:

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x, y)
```

Wir verwenden das `LinearRegression` Modell von Scikit-Learn <sup>4</sup> und rufen die `fit` Methode zusammen mit den Trainingsdaten und Labels auf, um das Modell zu trainieren. Die trainierten Parameter befinden sich in den Attributen `intercept_` und `coef_`:

```
>>> lin_reg.intercept_, lin_reg.coef_
(array([4.84609918]), array([[3.03831479]]))
```

Nicht schlecht: Das Modell vermutet  $\hat{y} = 4.85 + 3.04x_1$ , wobei die ursprüngliche Funktion  $y = 5 + 3x_1 + r$  war. Die genauen Werte konnten aufgrund des Rauschens nicht wiedergewonnen werden. Die Regressionsgerade ist in Abbildung 1.3 zu sehen.

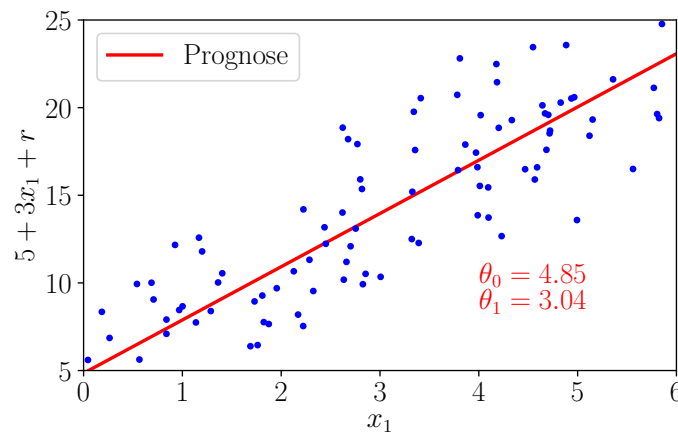


Abbildung 1.3: Regressionsgerade

---

<sup>4</sup><https://scikit-learn.org/stable/>

**Batch- und Online-Learning** Ein weiterer Punkt, anhand dessen ein KI-System unterschieden werden kann, ist die Methode, wie es Daten verarbeitet (Géron 2017, S. 15). Dieser Unterschied nennt sich Batch- und Online-Learning. Beim Batch-Learning werden Daten stapelweise verarbeitet. Das Modell bzw. die Parameter werden erst dann angepasst, nachdem der gesamte Stapel (Batch) das Training durchlaufen hat. Dies braucht in der Regel viel Zeit und Speicherkapazität, weshalb es typischerweise offline durchgeführt wird. Nachdem ein System veröffentlicht wurde, wendet es, dass an, was es gelernt hat, kann aber keine neuen Schlüsse mehr ziehen. Soll das System auf neue Daten reagieren, muss ein neues Modell trainiert werden. Das Online-Learning funktioniert anders. Im Training werden die Daten nicht stapelweise verarbeitet, sondern einzeln nacheinander. Dies spart Zeit und Platz und kann hilfreich sein, wenn es darum geht, mit sehr großen Datensätzen zu trainieren, die nicht alle gleichzeitig in den Speicher passen würden. Ein weiterer Vorteil ist der, dass neue Daten spontan auf dem Weg verarbeitet werden können. Online-Learning eignet sich also gut, um kontinuierliche Daten zu analysieren (z. B. Aktienkurse).

## 1.3 Beschreibung des Problems

In dieser Bachelorarbeit geht es um Maschinelles Lernen auf mobilen Geräten. Zu Beginn der Arbeit stellte sich der Autor die Frage: „Wie kann ein System entwickelt werden, mit dem verschiedene KI-Modelle verwaltet werden können, welches modular und einfach erweiterbar ist und über gemeinsame Schnittstellen im Netz kommuniziert?“ Das System soll gut zusammen mit der Idee des Internets der Dinge (engl. Internet of Things, IoT) integriert werden können. Die Modularität ist deshalb wichtig, da auch IoT aus vielen vernetzten Einzelementen besteht. Der restliche Teil dieser Arbeit wird sich mit einer speziellen Form von KI-Systemen beschäftigen: diese sind künstliche neuronale Netze. Die Natur hat bereits eine unzählige Menge von Erfindungen inspiriert, es liegt daher nahe, die Funktionsweise des Gehirns zu untersuchen, um ein intelligentes System zu entwickeln. Der Trainingsprozess eines neuronalen Netzes nennt sich Deep Learning (DL). Die technische Idee von DL ist bereits mehrere Jahrzehnte bekannt: Der Neurophysiologie Warren McCulloch und der Mathematiker Walter Pitts haben das Prinzip im Jahr 1943 erstmalig öffentlich eingeführt (ebd., S. 280). Der frühe Erfolg von neuronalen Netzen brachte Enthusiasmus und große Erwartungen, doch als sich in den 70er-Jahren herausstellte, dass

diese Erwartungen nicht erfüllt werden können, verlangsamte sich die Forschung und die finanziellen Fördermittel wurden weniger (Géron 2017, S. 280). Es ist nur kürzlich, dass neuronale Netze einen neuen Aufschwung erleben und dies nicht nur eine Wiederholung der Vergangenheit ist. Die folgenden Punkte begründen diese Vermutung (ebd., S. 280):

- Durch die immer stärker digitalisierte Welt entstehen sehr große Mengen an Daten. Die Forschung und Praxis hat gezeigt, dass tiefe neuronale Netze umso besser funktionieren, je mehr Trainingsdaten zur Verfügung stehen. Diese steht im Gegensatz zu anderen Lernverfahren, wessen Performanz ab einer bestimmten Datenmenge typischerweise begrenzt ist.
- Rechenstarke CPUs und GPUs machen es möglich, auch große neuronale Netze schnell zu trainieren. Der Trainingsprozess eines neuronalen Netzes ist ein sehr iterativer Prozess. Es beginnt häufig mit einer Idee, gefolgt von der Umsetzung und

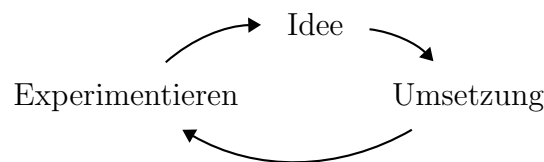


Abbildung 1.4: Der Trainingsprozess eines neuronalen Netzes

anschließenden Experimenten, um die ursprüngliche Idee zu bessern. Dieser Ablauf ist in Abbildung 1.4 zu sehen. Geht das Training schnell, kann dieser Kreislauf öfter durchlaufen werden und es wird ein besseres Ergebnis erzielt.

- Es gibt bereits viele hervorragende Anwendungen, die zeigen, wie gut neuronale Netze funktionieren, um komplexe Probleme zu lösen.

Maschinelles Lernen auf mobilen Geräten bringt einige zusätzlichen Herausforderungen mit sich.

# Literatur

- Géron, Aurélien (2017). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc. ISBN: 978-1-492-03264-9.
- Khartchenko, Evgueny (Jan. 2018). *Vectorization: A Key Tool To Improve Performance On Modern CPUs*. Einsichtsname: 21.05.2021. URL: <https://software.intel.com/content/www/us/en/develop/articles/vectorization-a-key-tool-to-improve-performance-on-modern-cpus.html>.
- Lynch, Shana (Mai 2017). *Andrew Ng: Why AI Is the New Electricity*. Einsichtsname: 17.05.2021. URL: <https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity>.