

DIN EN ISO/IEC 23053



ICS 35.020

Einsprüche bis 2023-05-03

Entwurf**Framework für Systeme der Künstlichen Intelligenz (KI) basierend auf
maschinellem Lernen (ML) (ISO/IEC 23053:2022);
Deutsche und Englische Fassung prEN ISO/IEC 23053:2023**

Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)
(ISO/IEC 23053:2022);
German and English version prEN ISO/IEC 23053:2023

Cadre méthodologique pour les systèmes d'intelligence artificielle (IA) utilisant
l'apprentissage machine (ISO/IEC 23053:2022);
Version allemande et anglaise prEN ISO/IEC 23053:2023

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2023-03-03 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an nia@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an den DIN-Normenausschuss Informationstechnik und Anwendungen (NIA), 10772 Berlin oder Am DIN-Platz, Burggrafenstr. 6, 10787 Berlin.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 88 Seiten

DIN-Normenausschuss Informationstechnik und Anwendungen (NIA)

Nationales Vorwort

Der Text von ISO/IEC 23053:2022 wurde vom Technischen Komitee ISO/IEC JTC 1 „Information technology“ der Internationalen Organisation für Normung (ISO) erarbeitet und als prEN ISO/IEC 23053:2023 durch das Technische Komitee CEN/CLC/JTC 21 „Künstliche Intelligenz“ übernommen, dessen Sekretariat von DS (Dänemark) gehalten wird.

Das zuständige nationale Normungsgremium ist der Gemeinschaftsarbeitsausschuss NA 043-01-42 GA „DIN/DKE Gemeinschaftsarbeitsausschuss Künstliche Intelligenz“ im DIN-Normenausschuss Informationstechnik und Anwendungen (NIA).

Um Zweifelsfälle in der Übersetzung auszuschließen, ist die englische Originalfassung beigelegt. Die Nutzungsbedingungen für den deutschen Text des Norm-Entwurfes gelten gleichermaßen auch für den englischen Text.

Für die in diesem Dokument zitierten Normen wird im Folgenden auf die entsprechenden deutschen Normen hingewiesen:

ISO/IEC 17789 siehe DIN ISO/IEC 17789

Aktuelle Informationen zu diesem Dokument können über die Internetseiten von DIN (www.din.de) durch eine Suche nach der Dokumentennummer aufgerufen werden.

Nationaler Anhang NA
(informativ)

Literaturhinweise

DIN ISO/IEC 17789, *Informationstechnik — Cloud Computing — Referenzarchitektur (ISO/IEC 17789:2014)*

– *Entwurf* –

E DIN EN ISO/IEC 23053:2023-04

– Leerseite –

Framework für Systeme der Künstlichen Intelligenz (KI) basierend auf maschinellem Lernen (ML) (ISO/IEC 23053:2022)

Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML) (ISO/IEC 23053:2022)

Cadre méthodologique pour les systèmes d'intelligence artificielle (IA) utilisant l'apprentissage machine (ISO/IEC 23053:2022)

Inhalt

	Seite
Europäisches Vorwort	4
Vorwort	5
Einleitung	6
1 Anwendungsbereich	7
2 Normative Verweisungen	7
3 Begriffe	7
3.1 Modellentwicklung und -nutzung	7
3.2 Tools	8
3.3 Daten	8
4 Abkürzungen	9
5 Überblick	10
6 Maschinelles Lernsystem	10
6.1 Überblick	10
6.2 Aufgabe	11
6.2.1 Allgemeines	11
6.2.2 Regression	12
6.2.3 Klassifizierung	12
6.2.4 Clustering (Gruppenzuordnung)	12
6.2.5 Anomalieerkennung	13
6.2.6 Dimensionalitätsverringern	13
6.2.7 Andere Aufgaben	13
6.3 Modell	14
6.4 Daten	14
6.5 Tools	16
6.5.1 Allgemeines	16
6.5.2 Datenaufbereitung	16
6.5.3 Kategorien von ML-Algorithmen	16
6.5.4 ML-Optimierungsverfahren	21
6.5.5 ML-Bewertungsmetriken	23
7 Ansätze für maschinelles Lernen	26
7.1 Allgemeines	26
7.2 Überwachtes maschinelles Lernen	27
7.3 Unüberwachtes maschinelles Lernen	29
7.4 Teilüberwachtes maschinelles Lernen	29
7.5 Selbstüberwachtes maschinelles Lernen	30
7.6 Bestärkendes maschinelles Lernen	30
7.7 Transfer Learning	31
8 Pipeline des maschinellen Lernens	32
8.1 Allgemeines	32
8.2 Datenerfassung	33
8.3 Datenaufbereitung	34
8.4 Modellierung	36
8.5 Verifizierung und Validierung	37
8.6 Bereitstellung des Modells	38
8.7 Betrieb	38
8.8 Beispiel für einen maschinellen Lernprozess auf der Grundlage der ML-Pipeline	39
Anhang A (informativ) Beispielhafte Datenfluss- und Datenverwendungserklärungen für überwachter Lernprozesse	41
A.1 Allgemeines	41
A.2 Datenflüsse im Prozess des überwachten maschinellen Lernens	41
A.2.1 Allgemeines	41

A.2.2	Beschreibungen von Datenflüssen	41
A.3	Datennutzung beim maschinellen Lernen	42
A.3.1	Allgemeines	42
A.3.2	Beispiel für eine Datenverwendungserklärung A	42
A.3.3	Beispiel für eine Datenverwendungserklärung B	42
A.3.4	Beispiel für eine Datenverwendungserklärung C	42
	Literaturhinweise	43

Bilder

Bild 1	— Elemente eines ML-Systems	11
Bild 2	— Konzeptdiagramm: Daten und Datensätze	15
Bild 3	— Beispiele verschiedener Kategorien von ML-Algorithmen	17
Bild 4	— Kurve der Betriebseigenschaften des Empfängers	25
Bild 5	— Konfusionsmatrix	26
Bild 6	— ML-Verfahren, kategorisiert als überwachtes maschinelles Lernen, unüberwachtes maschinelles Lernen und bestärkendes maschinelles Lernen	27
Bild 7	— ML-Modellerstellung durch überwachtes maschinelles Lernen	28
Bild 8	— Typisches überwachtes maschinelles Lernen	28
Bild 9	— ML-Modellerstellung durch unüberwachtes maschinelles Lernen	29
Bild 10	— Typischer Prozess des bestärkenden maschinellen Lernens	31
Bild 11	— Typischer Prozess von Transfer Learning	32
Bild 12	— Pipeline für maschinelles Lernen und Zuordnung zum Lebenszyklus von KI-Systemen	33
Bild 13	— Beispiel für einen überwachten maschinellen Lernprozess auf der Grundlage der ML-Pipeline	39

Europäisches Vorwort

Der Text von ISO/IEC 23053:2022 wurde vom Technischen Komitee ISO/IEC JTC 1 „Information technology“ der Internationalen Organisation für Normung (ISO) erarbeitet und als prEN ISO/IEC 23053:2023 durch das Technische Komitee CEN/CLC/JTC 21 „Künstliche Intelligenz“ übernommen, dessen Sekretariat von DS gehalten wird.

Dieses Dokument ist derzeit zur CEN-Umfrage vorgelegt.

Anerkennungsnotiz

Der Text von ISO/IEC 23053:2022 wurde von CEN als prEN ISO/IEC 23053:2023 ohne irgendeine Abänderung genehmigt.

Vorwort

ISO (die Internationale Organisation für Normung) und IEC (die Internationale Elektrotechnische Kommission) bilden das auf die weltweite Normung spezialisierte System. Nationale Normungsorganisationen, die Mitglieder von ISO oder IEC sind, beteiligen sich an der Entwicklung von Internationalen Normen in Technischen Komitees, die von der jeweiligen Organisation eingerichtet wurden, um spezifische Gebiete technischer Aktivitäten zu behandeln. Auf Gebieten von beiderseitigem Interesse arbeiten die Technischen Komitees von ISO und IEC zusammen. Weitere internationale staatliche und nichtstaatliche Organisationen, die in engem Kontakt mit ISO und IEC stehen, nehmen ebenfalls an der Arbeit teil.

Die Verfahren, die bei der Entwicklung dieses Dokuments angewendet wurden und die für die weitere Pflege vorgesehen sind, werden in den ISO/IEC-Direktiven, Teil 1 beschrieben. Im Besonderen sollten die für die verschiedenen ISO-Dokumentenarten notwendigen Annahmekriterien beachtet werden. Dieses Dokument wurde in Übereinstimmung mit den Gestaltungsregeln der ISO/IEC-Direktiven, Teil 2 erarbeitet (siehe www.iso.org/directives) oder www.iec.ch/members_experts/refdocs).

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. ISO und IEC sind nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren. Details zu allen während der Entwicklung des Dokuments identifizierten Patentrechten finden sich in der Einleitung und/oder in der ISO-Liste der erhaltenen Patenterklärungen (siehe www.iso.org/patents) oder der IEC-Liste der erhaltenen Patenterklärungen (siehe patents.iec.ch).

Jeder in diesem Dokument verwendete Handelsname dient nur zur Unterrichtung der Anwender und bedeutet keine Anerkennung.

Für eine Erläuterung des freiwilligen Charakters von Normen, der Bedeutung ISO-spezifischer Begriffe und Ausdrücke in Bezug auf Konformitätsbewertungen sowie Informationen darüber, wie ISO die Grundsätze der Welthandelsorganisation (WTO, en: World Trade Organization) hinsichtlich technischer Handelshemmnisse (TBT, en: Technical Barriers to Trade) berücksichtigt, siehe www.iso.org/iso/foreword.html. In der IEC, siehe www.iec.ch/understanding-standards.

Dieses Dokument wurde vom gemeinsamen Technischen Komitee ISO/IEC/JTC 1, *Information technology*, Unterkomitee SC 42, *Artificial Intelligence* erarbeitet.

Rückmeldungen oder Fragen zu diesem Dokument sollten an das jeweilige nationale Normungsinstitut des Anwenders gerichtet werden. Eine vollständige Auflistung dieser Institute ist unter www.iso.org/members.html und www.iec.ch/national-committees zu finden.

Einleitung

Bei Systemen der künstlichen Intelligenz (KI) handelt es sich im Allgemeinen um technische Systeme, die Ergebnisse wie Inhalte, Vorhersagen, Empfehlungen oder Entscheidungen für eine bestimmte, von Menschen definierte Zielsetzung erzeugen. KI deckt ein breites Spektrum an Technologien ab, die unterschiedliche Ansätze zur Behandlung dieser komplexen Probleme widerspiegeln.

ML ist ein Zweig der künstlichen Intelligenz, der Computertechniken verwendet, um Systeme in die Lage zu versetzen, aus Daten oder Erfahrungen zu lernen. Mit anderen Worten: ML-Systeme werden durch die Optimierung von Algorithmen entwickelt, um sie an Trainingsdaten anzupassen oder ihre Leistung durch die Maximierung einer Belohnung zu verbessern. Zu den ML-Verfahren gehört das tiefe Lernen, das ebenfalls in diesem Dokument behandelt wird.

In diesem Dokument werden Begriffe wie Wissen, Lernen und Entscheidungen verwendet. Es ist jedoch nicht die Absicht, das maschinelle Lernen (ML) zu vermenschlichen.

Ziel dieses Dokuments ist es, ein Framework für die Beschreibung von KI-Systemen zu schaffen, die ML verwenden. Durch die Festlegung einer gemeinsamen Terminologie und eines gemeinsamen Konzepts für solche Systeme bietet dieses Dokument eine Grundlage für die eindeutige Erläuterung der Systeme und verschiedene Überlegungen, die für deren Technik und Verwendung gelten. Dieses Dokument richtet sich an eine breite Zielgruppe, darunter Experten und Nicht-Berufsangehörige. Einige der Abschnitte (die im Überblick in Abschnitt 5 aufgeführt sind) enthalten jedoch ausführlichere technische Beschreibungen.

Dieses Dokument bildet auch die Grundlage für weitere Normen, die sich mit bestimmten Aspekten von ML-Systemen und deren Komponenten befassen.

1 Anwendungsbereich

In diesem Dokument wird ein Framework für künstliche Intelligenz (KI) und maschinelles Lernen (ML) zur Beschreibung eines generischen KI-Systems mit ML-Technologie festgelegt. Das Framework beschreibt die Systemkomponenten und ihre Funktionen im KI-Ökosystem. Dieses Dokument ist für alle Arten und Größen von Organisationen, einschließlich öffentlicher und privater Unternehmen, öffentlicher Stellen und gemeinnütziger Organisationen anwendbar, die KI-Systeme implementieren oder nutzen.

2 Normative Verweisungen

Die folgenden Dokumente werden im Text in solcher Weise in Bezug genommen, dass einige Teile davon oder ihr gesamter Inhalt Anforderungen des vorliegenden Dokuments darstellen. Bei datierten Verweisungen gilt nur die in Bezug genommene Ausgabe. Bei undatierten Verweisungen gilt die letzte Ausgabe des in Bezug genommenen Dokuments (einschließlich aller Änderungen).

ISO/IEC 22989, *Information technology — Artificial intelligence — Artificial intelligence concepts and terminology*

3 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach ISO/IEC 22989 und die folgenden Begriffe.

ISO und IEC stellen terminologische Datenbanken für die Verwendung in der Normung unter den folgenden Adressen bereit:

- ISO Online Browsing Platform: verfügbar unter <https://www.iso.org/obp>
- IEC Electropedia: verfügbar unter <https://www.electropedia.org/>

3.1 Modellentwicklung und -nutzung

3.1.1

Klassifizierungsmodell

<maschinelles Lernen> maschinelles Lernmodell, dessen erwartete Ausgabe für eine festgelegte Eingabe einer oder mehreren Klassen entspricht

3.1.2

Regressionsmodell

<maschinelles Lernen> maschinelles Lernmodell, dessen erwartete Ausgabe für eine festgelegte Eingabe einer kontinuierlichen Variablen entspricht

3.1.3

Verallgemeinerung

<maschinelles Lernen> Fähigkeit eines trainierten Modells, korrekte Vorhersagen für bis dahin unbekannte Eingabedaten zu treffen

Anmerkung 1 zum Begriff: Ein maschinelles Lernmodell, das sich gut verallgemeinern lässt, verfügt über eine akzeptable Vorhersagegenauigkeit, wenn es bis dahin unbekannte Eingabedaten verwendet.

Anmerkung 2 zum Begriff: Verallgemeinerung ist eng mit Überanpassung verbunden. Ein überangepasstes maschinelles Lernmodell lässt sich nicht gut verallgemeinern, da das Modell zu genau an die Trainingsdaten angepasst ist.

3.1.4

Überanpassung

<maschinelles Lernen> Erstellung eines Modells, das sich zu genau an die Trainingsdaten anpasst und nicht auf neue Daten verallgemeinert werden kann

Anmerkung 1 zum Begriff: Eine Überanpassung kann auftreten, weil das trainierte Modell aus unwesentlichen Merkmalen in den Trainingsdaten gelernt hat (d. h. Merkmale, die sich nicht zu nützlichen Ergebnissen verallgemeinern lassen), weil die Trainingsdaten übermäßig verrauscht sind (z. B. eine übermäßige Anzahl von Ausreißern) oder weil das Modell zu komplex für die Trainingsdaten ist.

Anmerkung 2 zum Begriff: Eine Überanpassung kann festgestellt werden, wenn ein erheblicher Unterschied zwischen den an den Trainingsdaten und den an separaten Test- und Validierungsdaten gemessenen Fehlern besteht. Die Leistung von überangepassten Modellen wird besonders dann beeinträchtigt, wenn eine erhebliche Diskrepanz zwischen Trainingsdaten und Produktionsdaten besteht.

3.1.5

Unteranpassung

<maschinelles Lernen> Erstellung eines Modells, das den Trainingsdaten nicht genau genug entspricht und bei neuen Daten falsche Vorhersagen macht

Anmerkung 1 zum Begriff: Unteranpassung kann auftreten, wenn die Merkmale schlecht ausgewählt sind, die Trainingszeit nicht ausreicht oder das Modell aufgrund einer begrenzten Modellkapazität (d. h. Ausdrucksstärke) zu einfach ist, um aus umfangreichen Trainingsdaten zu lernen.

3.2 Tools

3.2.1

Backpropagation

Trainingsverfahren für neuronale Netze, bei dem der Fehler in der Ausgabeschicht verwendet wird, um die Gewichte für die Verbindungen der aufeinanderfolgenden vorherigen Schichten anzupassen und zu optimieren

3.2.2

Lernrate

Schrittweite für ein Gradientenverfahren

Anmerkung 1 zum Begriff: Die Lernrate bestimmt, ob und wie schnell ein Modell zu einer optimalen Lösung konvergiert, und ist daher ein wichtiger Hyperparameter für neuronale Netze.

3.3 Daten

3.3.1

Klasse

vom Menschen definierte Kategorie von Elementen, die Teil des Datensatzes sind und gemeinsame Attribute haben

BEISPIEL „Telefon“, „Tisch“, „Stuhl“, „Kugellager“ und „Tennisball“ sind Klassen. Die Klasse „Tisch“ umfasst: einen Arbeitstisch, einen Esstisch, einen Schreibtisch, einen Couchtisch, eine Werkbank.

Anmerkung 1 zum Begriff: Klassen sind in der Regel Zielvariablen und werden durch einen Namen bezeichnet.

3.3.2

Cluster

automatisch ausgelöste Kategorie von Elementen, die Teil des Datensatzes sind und gemeinsame Attribute haben

Anmerkung 1 zum Begriff: Cluster haben nicht unbedingt einen Namen.

3.3.3

Merkmal

<maschinelles Lernen> messbare Eigenschaft eines Objekts oder Ereignisses in Bezug auf eine Reihe von Merkmalen

Anmerkung 1 zum Begriff: Merkmale spielen beim Training und bei der Vorhersage eine Rolle.

Anmerkung 2 zum Begriff: Merkmale bieten eine maschinenlesbare Möglichkeit, die relevanten Objekte zu beschreiben. Da der Algorithmus nicht auf die Objekte oder Ereignisse selbst zurückgreift, sind die Merkmalsdarstellungen so gestaltet, dass sie alle nützlichen Informationen enthalten.

3.3.4

Abstand

<maschinelles Lernen> gemessene Nähe zweier Punkte im Raum

Anmerkung 1 zum Begriff: Beim maschinellen Lernen wird in der Regel der euklidische bzw. geradlinige Abstand verwendet.

3.3.5

unmarkiert

Eigenschaft eines Samples, das keine Zielvariable enthält

4 Abkürzungen

KI	Künstliche Intelligenz
API	Anwendungsprogrammierschnittstelle (en: Application Programming Interface)
AUC	Fläche unter der Kurve (en: area under the curve)
BM	Boltzmann-Maschinen
CapsNet	neuronales Kapselnetz (en: capsule neural network)
CG	konjugierter Gradient (en: conjugate gradient)
CNN	Convolutional Neural Network (faltungsneuronales Netz)
DBN	tiefe Vertrauensnetze (en: deep belief networks)
DCNN	Deep Convolutional Neural Network
FFNN	neuronales Vorwärtsnetz (en: feed forward neural network)
FNR	Falsch-Negativ-Rate
FPR	Falsch-Positiv-Rate
GRU	Gated Recurrent Unit
LSTM	langes Kurzzeitgedächtnis (en: long short-term memory)
MAE	mittlerer absoluter Fehler (en: mean absolute error)
MEP	Markow-Entscheidungsprozess
ML	maschinelles Lernen
NN	neuronales Netzwerk
NEEF	Neural Network Exchange Format (Datenaustauschformat für künstliche neuronale Netze)
NPV	negativer Vorhersagewert (en: negative predictive value)
ONNX	offenes Format zur Darstellung von Deep-Learning-Modellen (en: open neural network exchange)
PCA	Hauptkomponentenanalyse (en: principal component analysis)
PHI	persönliche oder geschützte Gesundheitsinformationen (en: personal or protected health information)
pbD	personenbezogene Daten

PPV	positiver Vorhersagewert
REST	Representational State Transfer (repräsentative Zustandsübertragung)
RNN	rekurrentes neuronales Netz
ROC	Betriebseigenschaften des Empfängers (en: receiver operating characteristics)
SGD	stochastischer Gradientenabstieg (en: stochastic gradient descent)
SVM	Stützvektormaschine
TNR	Richtig-Negativ-Rate (en: true negative rate)
TPR	Richtig-Positiv-Rate (en: true positive rate)

5 Überblick

ISO/IEC 22989 definiert ML als den Prozess zur Optimierung von Modellparametern durch computergestützte Verfahren, so dass das Verhalten des Modells den Daten oder Erfahrungen entspricht. Seit den frühen 1940er Jahren werden die Modellierung von Neuronen (d. h. neuronalen Netzen) und die Entwicklung von Computerprogrammen, die aus Daten lernen können, erforscht. ML ist ein expandierendes Gebiet mit dem Aufkommen neuer Anwendungen in einem breiten Spektrum von Industriezweigen. Dieser Fortschritt wird durch die Verfügbarkeit von großen Datenmengen und Rechenressourcen ermöglicht. Zu den ML-Verfahren gehören neuronale Netze und tiefes Lernen.

In ISO/IEC 22989 wird ein KI-Ökosystem in Form von Funktionsschichten dargestellt, und ML ist eine wichtige Komponente dieses KI-Ökosystems. Bild 1 veranschaulicht das ML-System, das sich in die Komponenten Modell, Software-Tools und -Techniken sowie Daten aufgliedert.

In Abschnitt 6 dieses Dokuments werden die verschiedenen Komponenten des ML-Systems näher beschrieben.

In Abschnitt 7 dieses Dokuments werden verschiedene ML-Ansätze beschrieben und ihre Abhängigkeit von Trainingsdaten erläutert.

In Abschnitt 8 dieses Dokuments wird eine ML-Pipeline beschrieben: die Prozesse, die mit der Entwicklung, dem Einsatz und dem Betrieb eines ML-Modells verbunden sind.

6.5 und Abschnitt 7 sind technischer als der Rest des Dokuments. Ein fundierter technischer Hintergrund kann dem Leser helfen, diese Inhalte besser zu verstehen.

6 Maschinelles Lernsystem

6.1 Überblick

In Bild 1 sind die Elemente eines ML-Systems dargestellt. Sie legen die Rollen und ihre ML-spezifischen Funktionen fest, die von verschiedenen Entitäten (z. B. verschiedenen Anbietern) implementiert werden können. Die in Bild 1 dargestellten Beispiele sind nicht als erschöpfende Liste zu verstehen. Weitere Erläuterungen zu den einzelnen Rubriken von Bild 1 befinden sich in Abschnitt 6.

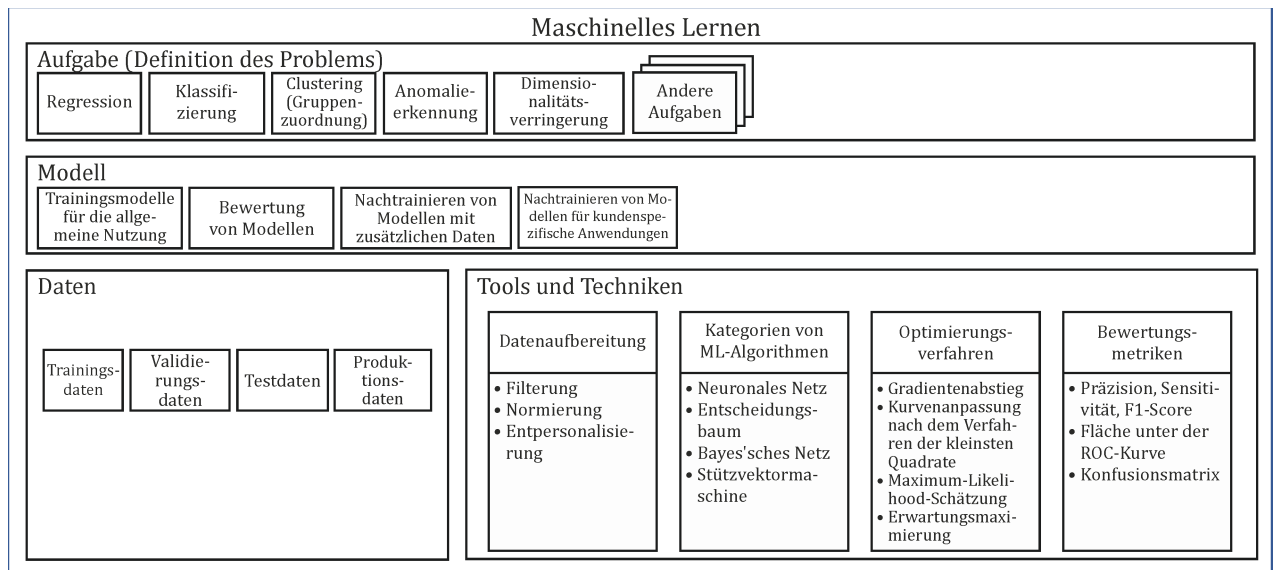


Bild 1 — Elemente eines ML-Systems

In Bild 1 können die Unterelemente der Modellentwicklung und -nutzung als ein mehrschichtiger Ansatz betrachtet werden, d.h. Anwendungen werden aus Modellen aufgebaut, die zur Lösung von Aufgaben verwendet werden. Die Entwicklung und Verwendung von Modellen hängt wiederum von Software-Tools und -Techniken sowie von Daten ab.

Ein einzelnes ML-System kann aus mehreren ML-Modellen bestehen, die in Kombination verwendet werden. Die Systemkomponenten können anhand ihrer Eingaben, Ausgaben und ihrer Absicht oder Funktion beschrieben werden. Die Komponenten können unabhängig voneinander geprüft werden.

ML-Modelle erzeugen bei ihrem Einsatz Ergebnisse wie Vorhersagen oder Entscheidungen. Ein vortrainiertes Modell ist ein ML-Modell, das bereits trainiert wurde, als man es erhielt. In einigen Fällen kann das entwickelte Modell auf eine ähnliche Aufgabe in einer anderen Domäne angewendet werden. Transfer Learning ist eine Technik zur Modifizierung eines vortrainierten ML-Modells, um eine andere verwandte Aufgabe zu erfüllen.

In diesem Dokument bezieht sich der Begriff Anwendung sowohl auf die beabsichtigte Verwendung eines oder mehrerer ML-Modelle als auch auf die konkrete Software, die diese Verwendung implementiert. ML-Modelle werden in der Regel mit anderen Softwarekomponenten integriert, um Anwendungen zu erstellen. Anwendungen, die ML verwenden, unterscheiden sich durch die Art der Eingabedaten, die sie verarbeiten, und durch die Art der Aufgaben, die sie ausführen. In einigen Anwendungen trifft ML Vorhersagen oder Entscheidungen auf hoher Ebene, während ML in anderen Anwendungen Antworten auf eng definierte Probleme liefert.

Unterschiede bei den Eingabedaten und Aufgaben sowie Faktoren wie Einsatzmöglichkeiten, Genauigkeit und Zuverlässigkeit führen zu unterschiedlichen Anwendungsgestaltungen. KI-Anwendungen können proprietäre benutzerdefinierte Gestaltungen verwenden oder domänenspezifischen Gestaltungsmustern folgen.

Die Anwendungslogik wird durch das Format der Eingabedaten, die Ausgabedaten und möglicherweise die Transformation und den Datenfluss zwischen den verwendeten ML-Modellen bestimmt. In allen Fällen wird die Wahl der ML-Algorithmen und Datenaufbereitungstechniken auf die Aufgaben der Anwendung zugeschnitten.

6.2 Aufgabe

6.2.1 Allgemeines

Der Begriff „Aufgabe“ bezieht sich auf Aktionen, die zur Erreichung eines bestimmten Ziels erforderlich sind. Bei ML bedeutet dies, dass ein Problem ermittelt wird, das mit Hilfe des ML-Modells gelöst werden soll. Für eine ML-Anwendung können eine oder mehrere ML-Aufgaben festgelegt werden. Anstatt ein Problem mit Hilfe

einer spezifischen Funktion zu lösen, die als eine Reihe von Schritten dargestellt und in einem Softwarecode implementiert ist, wird das definierte Problem durch Anwendung eines trainierten ML-Modells auf die Produktionsdaten gelöst. Im Grunde genommen implementiert das trainierte ML-Modell eine Zielfunktion, die eine Annäherung an die hypothetische Funktion darstellt, die ein Programmierer zur Lösung des Problems geschrieben hätte.

Der Aufbau einer ML-Aufgabe umfasst die Definition des Problems, des Datenformats und der Merkmale.

Die in den folgenden Unterabschnitten beschriebenen Aufgaben sind Beispiele und erheben keinen Anspruch auf Vollständigkeit.

6.2.2 Regression

Regressionsaufgaben umfassen die Vorhersage einer kontinuierlichen Variablen durch Lernen einer Funktion, die am besten zu einem Satz von Trainingsdaten passt. Bei einer Regressionsaufgabe stellt das trainierte Regressionsmodell einen kundenspezifischen Raum dar. Wenn das trainierte Modell auf eine neue Produktionsdateninstanz angewendet wird, wird die Instanz in den durch das trainierte Regressionsmodell definierten benutzerdefinierten Raum projiziert.

Die Regression wird hauptsächlich verwendet, um numerische Werte eines realen Prozesses auf der Grundlage früherer Messungen oder Beobachtungen desselben Prozesses vorherzusagen. Zu den Anwendungsfällen für Regression gehören:

- Vorhersage von Börsenkursen;
- Vorhersage des Alters eines Zuschauers von Streaming-Videos;
- Vorhersage der Menge des prostataspezifischen Antigens im Körper auf der Grundlage verschiedener klinischer Messungen.

6.2.3 Klassifizierung

Klassifizierungsaufgaben umfassen die Vorhersage der Zuweisung einer Instanz von Eingabedaten zu einer bestimmten Kategorie oder Klasse. Die Klassifizierung kann binär (d. h. richtig oder falsch), in mehreren Klassen (d. h. eine von mehreren Möglichkeiten) oder als Multi-Label (d. h. eine beliebige Anzahl von mehreren Möglichkeiten) erfolgen. Mit Hilfe der Klassifizierung lässt sich zum Beispiel vorhersagen, ob es sich bei einem Objekt auf einem Bild um eine Katze oder einen Hund oder sogar um eine völlig andere Tierart handelt. Die Klassen stammen in der Regel aus einer diskreten und ungeordneten Menge, so dass das Problem nicht als Regressionsaufgabe formalisiert werden kann. Eine medizinische Diagnose für eine Reihe von Symptomen kann beispielsweise {Schlaganfall, Drogenüberdosis, Krampfanfall} lauten, wobei die Klassenwerte nicht geordnet sind und es keinen kontinuierlichen Wechsel von einer Klasse zur anderen gibt.

Zu den Anwendungsfällen für die Klassifizierung gehören:

- Dokumentenklassifizierung und E-Mail-Spamfilterung, bei der Dokumente in verschiedene Klassen eingeteilt werden. Ein Spam-Filter zum Beispiel verwendet zwei Klassen, nämlich „Spam“ und „kein Spam“;
- Klassifizierung der Arten eines Exemplars. Ein ML-Klassifizierungsmodell kann beispielsweise die Art einer Blume vorhersagen, wenn es Daten erhält, die die Länge und Breite des Kelchblattes sowie die Länge und Breite der Blütenblätter angeben;
- Bildklassifizierung. Bei einer Reihe von Bildern (z. B. von Möbeln) kann ein ML-System verwendet werden, um die auf diesen Bildern abgebildeten Objekte zu erkennen und zu benennen.

6.2.4 Clustering (Gruppenzuordnung)

Clustering-Aufgaben umfassen die Gruppierung von Eingabedateninstanzen. Anders als bei Klassifizierungsaufgaben sind die Klassen bei Clustering-Aufgaben nicht vordefiniert, sondern werden im Rahmen des

Clustering-Prozesses bestimmt. Clustering kann als Datenvorbereitungsschritt verwendet werden, um homogene Daten zu identifizieren, die dann als Trainingsdaten für überwachtes maschinelles Lernen verwendet werden können. Clustering kann auch verwendet werden, um Ausreißer oder Anomalien zu erkennen, indem Eingabedateninstanzen identifiziert werden, die nicht mit anderen Samples vergleichbar sind. Anwendungsbeispiele für Clustering-Aufgaben sind das Sortieren oder Organisieren von Dateien.

6.2.5 Anomalieerkennung

Die Erkennung von Anomalien umfasst die Identifizierung von Eingabedaten, die nicht mit einem erwarteten Muster übereinstimmen. Die Erkennung von Anomalien kann für Anwendungen wie die Aufdeckung von Betrug oder ungewöhnlichen Aktivitäten nützlich sein. Bei der Erkennung von Anomalien sagt das ML-Modell voraus, ob eine Eingabedateninstanz typisch für eine bestimmte Verteilung ist.

6.2.6 Dimensionalitätsverringern

Die Dimensionalitätsverringern besteht darin, die Anzahl der Attribute oder Dimensionen je Sample zu verringern und dabei den Großteil der nützlichen Informationen beizubehalten.

Die Dimensionalitätsverringern kann die nützlichsten Merkmale eines Datensatzes hervorheben und dadurch die Berechnungskosten verringern.

Die Dimensionalitätsverringern mildert die verschiedenen ungünstigen Auswirkungen einer zu großen Anzahl von Merkmalen, die allgemein als „Fluch der Dimensionalität“ bekannt sind. Die Dimensionalitätsverringern ist auch für die Datenexploration und die Modellanalyse nützlich.

Es gibt unüberwachte, überwachte und halbüberwachte Verfahren zur Dimensionalitätsverringern [1].

6.2.7 Andere Aufgaben

Es gibt viele andere Aufgaben, die unterschiedliche Zwecke und erwartete Ergebnisse haben. Diese Aufgaben können spezifisch für eine bestimmte Anwendung sein. Beispiele für andere Aufgaben sind die semantische Segmentierung von Text oder Bildern, maschinelle Übersetzung, Spracherkennung oder -synthese, Objektlokalisierung und Bilderzeugung.

Bei der Planung besteht die Aufgabe darin, eine Abfolge von Aktionen eines oder mehrerer Agenten durch Beobachtung des Umgebungszustands zu optimieren.

Trotz ihrer Verschiedenartigkeit wurde eine Reihe von Konzepten formuliert, um Verbindungen zwischen einigen dieser anderen Aufgaben herzustellen. Ein solches Konzept ist die strukturierte Vorhersage, die sich auf Aufgaben bezieht, bei denen die erwartete Ausgabe des Modells ein strukturiertes Objekt und nicht nur ein einzelner Wert ist.

Die strukturierte Vorhersage erfordert Berechnungsverfahren, die Regelmäßigkeiten in der Ausgabe berücksichtigen können, indem sie diese entweder explizit modellieren oder indem sie die gesamte Struktur gemeinsam mit einem Modell vorhersagen, das die Regelmäßigkeiten intern modelliert.

Zu den Anwendungsfällen für strukturierte Vorhersagen gehören:

- einen Parsebaum für einen Satz einer natürlichen Sprache zu erstellen;
- einen Satz in einer Sprache in einen Satz in einer anderen Sprache zu übersetzen;
- die Proteinstruktur vorherzusagen;
- die semantische Segmentierung eines Bildes.

6.3 Modell

ISO/IEC 22989:2022, 3.2.11, definiert ein ML-Modell als mathematisches Konstrukt, das auf der Grundlage von Eingabedaten oder -informationen einen Rückschluss bzw. eine Vorhersage generiert. Das ML-Modell umfasst eine Datenstruktur und eine Software zur Verarbeitung der Struktur, die beide durch einen ausgewählten ML-Algorithmus bestimmt werden. Das Modell wird mit Eingaben und Ausgaben konfiguriert, die für die Lösung des gegebenen Problems wichtig sind.

Das Modell wird mit Daten befüllt (auch als „trainiert“ bezeichnet), um die relevanten statistischen Eigenschaften der Trainingsdaten darzustellen. Durch den Trainingsprozess „lernt“ das Modell, wie das Problem für die Trainingsdaten zu lösen ist, mit dem Ziel, dieses erworbene Wissen auf eine reale Anwendung anzuwenden.

ML-Modelle liefern Ergebnisse, die Annäherungen an optimale Lösungen sind. ML-Algorithmen verwenden statistische Optimierungsverfahren, um diese Annäherung durchzuführen. Die sich daraus ergebende Zuordnung zwischen den Eingaben und den Ausgaben des Modells spiegelt die aus den Trainingsdaten gelernten Muster wider. Muster können sich auf Korrelationen, kausale Beziehungen oder Kategorien von Datenobjekten beziehen. ML-Modelle sind das Ergebnis der verwendeten Trainingsdaten. Wenn also die verwendeten Daten unvollständig sind oder eine gesellschaftliche Verzerrung aufweisen, wird sich dies auch in der Modellleistung widerspiegeln. Daher sollte man bei den Datensätzen, die für das Training von Modellen verwendet werden, behutsam vorgehen. Die Logik, die im Prozess des maschinellen Lernens entsteht und durch das trainierte Modell dargestellt wird, wird nicht von einem Programmierer festgelegt, sondern entwickelt sich während der Trainingstätigkeiten.

Um zu sehen, wie leistungsfähig das Modell ist, wird es mit Hilfe von Bewertungsmetriken bewertet.

Beim Nachtrainieren wird ein trainiertes Modell durch Training mit anderen Trainingsdaten aktualisiert. Dies kann aus vielen Gründen notwendig sein, unter anderem wegen des Mangels an großen Trainingsdaten, Datendrift und Konzeptdrift.

Bei der Datendrift nimmt die Genauigkeit der Modellvorhersagen im Laufe der Zeit aufgrund von Änderungen der statistischen Merkmale der Produktionsdaten ab. In diesem Fall muss das Modell mit neuen Trainingsdaten, die die Produktionsdaten besser darstellen, nachtrainiert werden.

Bei der Konzeptdrift scheint sich die Entscheidungsgrenze zu verschieben, wodurch sich die Genauigkeit der Vorhersagen ebenfalls verschlechtert, auch wenn sich die Daten nicht geändert haben. Im Falle einer Konzeptdrift müssen die Zielvariablen in den Trainingsdaten neu markiert und das Modell nachtrainiert werden.

Ein Nachtrainieren kann auch zu Zwecken wie Transfer Learning und Optimierung oder Modifizierung des ML-Modells erfolgen.

Einige Modelle sind bereits verfügbar und können dann nachtrainiert und für einen bestimmten Anwendungsfall optimiert oder unverändert verwendet werden. Ein Beispiel ist ein handelsübliches maschinelles Übersetzungsmodell, das für die Übersetzung juristischer Dokumente nachtrainiert werden kann.

Kontinuierliches Lernen ist ein Sonderfall des Nachtrainierens, bei dem sich die Leistung des Modells ständig weiterentwickelt, was auf das laufende Training des Modells mit Produktionsdaten zurückzuführen ist. In solchen Fällen kann es erforderlich sein, die Leistung des Modells kontinuierlich zu überwachen oder „Schutzmechanismen“ für das akzeptable Verhalten des Modells zu setzen.

6.4 Daten

Bild 2 ist ein Rake-Diagramm, das zeigt, dass das Datenkonzept in vier sich gegenseitig ausschließende Kategorien unterteilt ist:

- a) Trainingsdatensatz, der zur Schätzung der Parameter der Kandidatenmodelle verwendet wird;

- b) Validierungsdatensatz, je nach Gebiet der KI auch Entwicklungsdatensatz genannt (z. B. bei der Verarbeitung natürlicher Sprache), der zur Auswahl des besten Modells nach einem Leistungskriterium verwendet wird;
- c) Testdatensatz, der zur Überprüfung der Verallgemeinerungsfähigkeit eines Modells verwendet wird und seine Leistung bei zukünftigen Daten bestimmt;
- d) Produktionsdaten, bestehend aus Betriebsdaten, die vom Modell für die Vorhersage verwendet werden. Die Verteilung der Produktionsdaten kann sich von der des Trainings-, Validierungs- und Testdatensatzes unterscheiden.

Trainings-, Validierungs- und Testdatensätze können durch simulierte und gestörte Daten ergänzt werden.

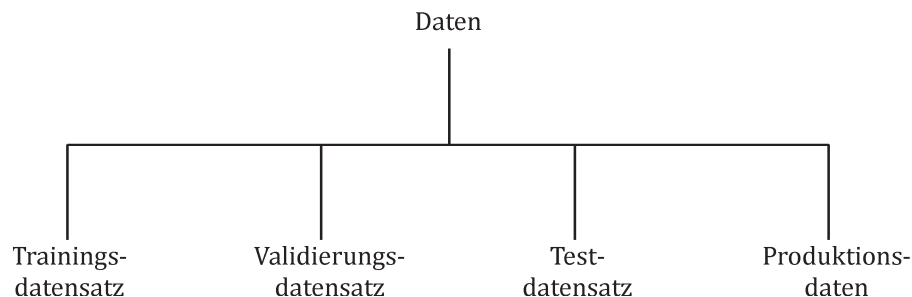


Bild 2 — Konzeptdiagramm: Daten und Datensätze

Diese verschiedenen Datenarten können entweder aus reinen Eingabedaten oder aus Eingabedaten in Verbindung mit Labels (den erwarteten Ausgabedaten) bestehen. Validierungs- und Testdaten sind häufig markiert, während die Produktionsdaten in der Regel unmarkiert sind. Bei den Trainingsdaten kommt es auf den ML-Ansatz an: sie können unmarkiert, teilweise markiert oder vollständig markiert sein. Markierte Trainingsdaten ermöglichen es dem ML-Algorithmus, statistische Beziehungen zwischen Eingabevariablen und der Zielvariablen zu erkennen. Unmarkierte Trainingsdaten ermöglichen es dem ML-Algorithmus, statistische Korrelationen und Strukturen innerhalb der Eingabedaten zu erkennen.

Validierungs- und Testdaten werden beide mit statistischen Leistungsmessungen verwendet (die in 6.5.5 erörtert werden), aber ihre Verwendungszwecke unterscheiden sich: Validierungsdaten werden zur Abstimmung der Hyperparameter verwendet, während es bei Testdaten um die Bewertung des Modells geht. Mit den Testdaten soll verifiziert werden, dass das trainierte Modell bei Produktionsdaten leistungsfähig ist bzw. sich gut verallgemeinern lässt. Trainierte Modelle, die sich nicht gut verallgemeinern lassen, werden als „überangepasst“ (an die Trainingsdaten) bezeichnet.

Es ist zu beachten, dass diese Verwendung des Begriffs Testdaten auf ML-spezifische Prozesse beschränkt ist und sich von der Verwendung des Begriffs im Zusammenhang mit der Verifizierung und Validierung eines integrierten Systems, das ML-Komponenten verwendet, unterscheidet; in diesem Fall würde er sich auf alle Daten beziehen, die zu Verifizierungs- und Validierungszwecken verwendet werden, ohne einen besonderen Bezug zu ML. Ferner ist zu beachten, dass die ML-spezifische Verwendung des Begriffs „Validierungsdaten“ nichts mit den Verifizierungs- und Validierungsprozessen zu tun hat, da sie sich auf die Modellentwicklung beziehen.

Für eine zuverlässige Anwendung der ML-Verfahren müssen Trainings-, Validierungs- und Testdaten voneinander getrennt sein. Trainings-, Validierungs- und Testdatensätze können aus demselben Datensatz mit Hilfe von Datensplits gewonnen werden, oder sie können getrennt erfasst werden. In einer idealen Konfiguration haben sie alle die gleiche statistische Verteilung, aber je nach Anwendungsfall und ML-Ansatz kann es notwendig sein, anders vorzugehen.

Um das trainierte Modell zuverlässig bewerten zu können, müssen die Testdaten eine möglichst ähnliche Verteilung wie die Produktionsdaten aufweisen.

Die Produktionsdaten werden dem Modell erst nach seinem Einsatz angezeigt. Damit das Modell genaue Vorhersagen machen kann, müssen die Produktionsdaten im Allgemeinen eine ähnliche Verteilung aufweisen wie die Trainings- und Validierungsdaten, obwohl es spezielle Techniken gibt, die die Verschlechterung im Falle einer Diskrepanz abmildern können.

Im Laufe der Zeit kann sich die Verteilung der Produktionsdaten ändern, so dass das Modell anhand neuer Daten nachtrainiert werden muss. In Fällen, in denen sich das Modell dynamisch an neue Muster in den Produktionsdaten anpassen muss, kann das Modell kontinuierlich nachtrainiert werden, indem die aus den Produktionsdaten gewonnenen Informationen genutzt werden. Solche Fälle werden in 6.3 erörtert.

6.5 Tools

6.5.1 Allgemeines

Bei der Erstellung von ML-Modellen werden Tools verwendet, die in die Kategorien Datenaufbereitung, ML-Algorithmen, Optimierungsverfahren und Bewertungsmetriken unterteilt sind. Die Leistung von ML-Modellen wird mit Hilfe von Tools bewertet, die Bewertungsmetriken erstellen.

Die Erstellung von ML-Modellen erfordert aufgrund der Rechenanforderungen und der Verwendung großer Trainingsdatensätze häufig hochleistungsfähige Rechenkapazitäten. Die Rechen- und Speicherleistung kann auch beeinflussen, wie schnell ML-Modelle entwickelt und trainiert werden können.

Zu den grundlegenden Herausforderungen von ML gehören die statistische Analyse, die Gestaltung von Algorithmen und die Optimierung. Die statistische Analyse beruht auf den Grundsätzen mathematischer Modelle, die aus Trainingsdaten abgeleitet werden. Die Gestaltung des Algorithmus ist die Art und Weise der Implementierung der algorithmischen Techniken, die zum Aufbau des ML-Modells verwendet werden. Die Optimierung eines bestimmten ML-Modells ist ebenfalls ein wichtiges Thema bei der Implementierung von ML. Eine weitere Herausforderung besteht darin, das Potential und die Möglichkeiten von ML zu verstehen. Es stützt sich auf Daten und wird daher in vielen Fällen bestehende Fehler und Benachteiligungen vervielfältigen, verstärken und fördern.

6.5.2 Datenaufbereitung

Die Datenaufbereitung wird in 8.3 erörtert.

6.5.3 Kategorien von ML-Algorithmen

6.5.3.1 Allgemeines

Die Wahl des ML-Algorithmus bestimmt die Berechnungsstruktur des ML-Modells und seinen Trainingsansatz.

Algorithmen können für verschiedene ML-Zwecke eingesetzt werden, z. B.:

- einen Algorithmus zur Informationsdarstellung, der an der Datenaufbereitung beteiligt sein kann. Dies hängt mit dem Feature Engineering zusammen;
- ein Algorithmus, der bei der Erstellung eines ML-Modells verwendet wird.

Die Beziehung zwischen ML-Algorithmen und ML-Modellen lässt sich anhand der Lösung einer univariaten linearen Funktion $y = \theta_0 + \theta_1 x$ veranschaulichen, wobei y eine Ausgabe oder ein Ergebnis, x eine Eingabe, θ_0 ein Achsenabschnitt (der Wert von y , bei dem $x = 0$ ist) und θ_1 ein Gewicht ist. Bei ML wird der Prozess der Bestimmung des Achsenabschnitts und des Gewichts für eine lineare Funktion als lineare Regression bezeichnet. Wenn eine univariate lineare Funktion ($y = \theta_0 + \theta_1 x$) mittels linearer Regression trainiert wurde, kann das resultierende Modell $y = 3 + 7x$ sein.

Bild 3 zeigt Beispiele verschiedener Kategorien von ML-Algorithmen.

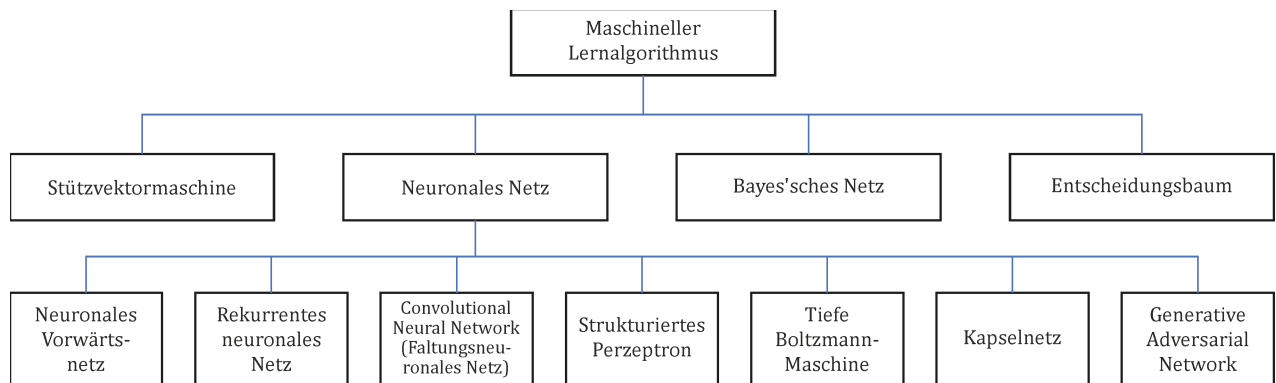


Bild 3 — Beispiele verschiedener Kategorien von ML-Algorithmen

Die Wahl eines ML-Algorithmus ist oft unzureichend für die Bestimmung der Struktur des ML-Modells und seines Trainingsansatzes. Für viele Algorithmen müssen auch Hyperparameter gewählt werden.

Hyperparameter sind Merkmale eines ML-Algorithmus, die seinen Lernprozess beeinflussen. Sie können in Prozessen verwendet werden, um die Schätzung von Modellparametern zu unterstützen. Beispiele für Hyperparameter für neuronale Netze sind die Anzahl der Netzschichten, die Breite der einzelnen Schichten und die Art der Aktivierungsfunktion. Ein praktischer Ansatz zur Bestimmung des optimalen Hyperparametersatzes unter allen möglichen Kombinationen ist die Durchführung von Zufallssuchen, die durch eine Beschränkungsfunktion geleitet werden, und die Messung der Leistung anhand eines Validierungsdatensatzes. Dieser Schritt wird als Modellauswahl oder Abstimmung der Hyperparameter bezeichnet.

6.5.3.2 Neuronales Netz

6.5.3.2.1 Allgemeines

Neuronale Netze (NN) versuchen, die intelligente Fähigkeit zu simulieren, komplexe Probleme zu beobachten, zu lernen, zu analysieren und Entscheidungen zu treffen. Die NN-Gestaltung orientiert sich daher an der Art und Weise, wie die Neuronen im Gehirn von Menschen und Tieren miteinander verbunden sind. Sie verwenden mehrschichtige Netze aus einfachen Recheneinheiten (oder „Neuronen“). In diesen NN kombiniert jede Einheit eine Reihe von Eingabewerten, um einen Ausgabewert zu erzeugen, der wiederum an andere, nachgeschaltete Neuronen weitergegeben wird. Die Struktur eines NN besteht aus miteinander verknüpften Verarbeitungselementen. Ein Netz von Knoten (oder Neuronen) ist durch gewichtete Ränder verbunden. Die einfachste Form des Netzes besteht aus einem oder mehreren Eingabeneuronen, die ein oder mehrere Merkmale aus den Eingabedaten aufnehmen, sowie einer Ausgabeschicht, die ein oder mehrere Neuronen enthält. Jedes Neuron kann eine oder mehrere Eingaben aus der vorhergehenden Schicht akzeptieren, und seine Ausgabe wird durch eine Aktivierungsfunktion bestimmt, die in der Regel nichtlinear von der gewichteten Kombination der Eingaben aus der vorhergehenden Schicht abhängt. Das NN „lernt“, indem es mit bekannten Eingaben trainiert, die tatsächliche Ausgabe mit der erwarteten vergleicht und den Fehler zur Anpassung der Gewichte verwendet.

Verborgene Schichten können zwischen der Ausgabe- und der Eingabeschicht eines NN liegen; in diesem Fall spricht man von einem mehrschichtigen Perzeptron oder einem mehrschichtigen NN. Tiefe neuronale Netze oder tiefes Lernen beziehen sich speziell auf neuronale Netze mit vielen verborgenen Schichten. Tiefes Lernen ist ein Ansatz zur Erstellung umfangreicher hierarchischer Darstellungen durch das Training neuronaler Netze mit vielen verborgenen Schichten. Dieses Verfahren ermöglicht es dem neuronalen Netz, die endgültige Ausgabe schrittweise zu verfeinern. Tiefes Lernen kann die Notwendigkeit des Feature-Engineerings verringern oder beseitigen, da die relevantesten Merkmale automatisch identifiziert werden.

Neuronale Netze können in drei allgemeine Arten eingeteilt werden: diskriminativ, generativ und hybrid (Kombination der beiden erstgenannten Arten).

Sie können als konnektionistischer Ansatz betrachtet werden. Der Konnektionismus verwendet ein Netz von miteinander verbundenen Einheiten, bei denen es sich im Allgemeinen um einfache Recheneinheiten han-

delt [2]. Das Verhalten des Netzes kann durch Änderung der jeder Verbindung zugeordneten Gewichte verfeinert werden, was, wie oben beschrieben, durch Training erreicht wird. Die Knotenpunkte des Netzes verarbeiten Informationen parallel.

Tiefes Lernen kann viel Zeit und Rechenressourcen erfordern. Die ML-NN benötigen oft eine beträchtliche Menge an Rechenleistung und Speicherplatz, um zu funktionieren. Eine Verringerung der Modellkomplexität kann für die Nutzung dieser Netze auf mobilen und eingebetteten Geräten von Vorteil sein. Die Auswirkungen auf die Laufzeit und der Energieverbrauch können minimiert werden, und in einigen Fällen ermöglicht die Reduzierung sogar die Ausführung der NN in Echtzeit auf mobilen Geräten, ohne auf Cloud-Dienste angewiesen zu sein. Durch Komprimierung der Gewichte oder Verkleinerung der Architektur kann die Komplexität des Modells bei nahezu identischer Leistung um einen erheblichen Faktor reduziert werden.

6.5.3.2.2 Neuronale Vorwärtsnetze

Vorwärtsgerichtete neuronale Netze (FFNN) sind die einfachsten NN-Architekturen. Sie leiten Informationen von der Eingabeschicht zur Ausgabeschicht nur in eine Richtung. Es gibt keine Verbindungen zwischen den Neuronen innerhalb einer bestimmten Schicht. Zwei benachbarte Schichten sind in der Regel „vollständig verbunden“, d. h. jedes Neuron in einer Schicht hat eine Verbindung zu jedem Neuron in der nachfolgenden Schicht. Jeder Verbindung ist ein Gewicht zugeordnet. FFNN werden in der Regel durch Backpropagation mit markierten Trainingsdaten trainiert, wobei jedes Sample mit der Grundwahrheit für den erwarteten Wert der Ausgabedaten markiert wird. Die Differenz zwischen der tatsächlichen Ausgabe des FFNN und der Grundwahrheit wird als Fehler bezeichnet [3]. Bei der Backpropagation wird der Fehler in der Ausgabeschicht verwendet, um die Gewichte für die Verbindungen der aufeinanderfolgenden vorherigen Schichten anzupassen. Sie wird häufig zusammen mit dem Gradientenabstieg [4] verwendet.

6.5.3.2.3 Rekurrentes neuronales Netz

6.5.3.2.3.1 Allgemeines

Rekurrente neuronale Netze (RNN) [5] sind NN, die darauf abzielen, mit sequentiellen Eingaben umzugehen, die in einer geordneten Abfolge erscheinen und bei denen die Reihenfolge der Eingaben in der Abfolge eine Rolle spielt. Beispiele für solche Eingaben sind dynamische Sequenzen wie Ton- und Videoströme, aber auch statische Abfolgen wie Text oder sogar einzelne Bilder. RNN können grundsätzlich auf vielen Gebieten eingesetzt werden, da die meisten Datenformen, die keine Zeitachse haben (d. h. im Gegensatz zu Ton oder Video), als Abfolge dargestellt werden können. Ein Bild oder eine Textfolge kann pixel- oder zeichenweise eingespeist werden, so dass die zeitabhängigen Gewichte für das verwendet werden, was in der Abfolge davor lag, und nicht für das, was zu einem früheren Zeitpunkt geschah.

RNN haben eine zustandsabhängige Eigenschaft, die durch vergangenes Lernen beeinflusst wird. Es wird erwartet, dass die Eingabe in einer Abfolge von Durchläufen über die Zeit erfolgt, und das RNN speichert Informationen aus einem Durchlauf, um sie in einem späteren Durchlauf zu verwenden, d. h. das RNN hat einen Speicher. In einem RNN haben die Neuronen in einer Schicht also nicht nur gewichtete Verbindungen von Neuronen in einer vorherigen Schicht für den aktuellen Durchlauf, sondern auch Eingaben von Neuronen aus vorherigen Durchläufen. RNN werden häufig in der Spracherkennung, der maschinellen Übersetzung, der Zeitreihenvorhersage und der Bilderkennung eingesetzt. Im Allgemeinen sind RNN eine gute Wahl, wenn es darum geht, Informationen weiterzuleiten oder zu vervollständigen, z. B. bei der automatischen Vervollständigung.

RNN sind gut geeignet, um aufeinanderfolgende Eingabedaten mit variabler Länge zu verarbeiten oder aufeinanderfolgende Daten mit variabler Länge auszugeben. Zu den gebräuchlichen Arten von RNN gehören Netzwerke mit langem Kurzzeitgedächtnis (LSTM) und Gated Recurrent Unit (GRU), eine einfachere Variante des LSTM.

6.5.3.2.3.2 Netze des langen Kurzzeitgedächtnisses

Netze mit langem Kurzzeitgedächtnis (LSTM) sind eine Form von RNN, die für Probleme entwickelt wurden, die das Erinnern von Informationen mit sowohl längeren als auch kürzeren Zeitunterschieden erfordern, was sie geeignet macht, langfristige Verbindungen zu lernen. Sie wurden eingeführt, um das Problem des verschwin-

denden Gradienten in RNN zu lösen, das mit Backpropagation verbunden ist. Jede LSTM-Zelle hat einen internen Speicher und einen verborgenen Zustand. RNN sind beim Training auf Backpropagation angewiesen, aber dieser Ansatz kann entweder unter dem Problem des verschwindenden Gradienten oder dem Problem des explodierenden Gradienten leiden [6]. Während des Trainings werden die RNN-Gewichte auf der Grundlage der partiellen Ableitung der Fehlerfunktion in jeder Iteration des Trainings aktualisiert. Die Ableitung kann entweder sehr klein sein, was zu einem RNN führt, das praktisch nicht trainierbar ist, oder sie kann sehr groß sein, was zu einem RNN führt, das sehr instabil ist.

Ein LSTM ist für das Lernen langfristiger Abhängigkeiten gestaltet und hat eine Architektur, die auf einer Kombination von Neuronen basiert, mit einer Zelle (die die Speicherfähigkeit hat), einem Eingabe-Gate, einem Ausgabe-Gate und einem Vergessen-Gate. Die Speicherzelle ist mit jedem Neuron verbunden, das Informationen über seinen vorherigen Zustand enthält. Die Funktion dieser Gates besteht darin, die Informationen zu schützen, indem sie den Informationsfluss stoppen oder zulassen. Das Eingabe-Gate bestimmt, wie viel von der Information aus der vorherigen Schicht in der Zelle gespeichert wird. Das Ausgabe-Gate am anderen Ende bestimmt, wie viel vom Zustand dieser Zelle an die nächste Schicht weitergegeben wird. Das Vergessen-Gate ermöglicht es dem Netz, seinen internen Zustand zu löschen. Wenn es sich bei der Eingabe beispielsweise um einen Absatz handelt und ein neuer Satz beginnt, kann es erforderlich sein, dass das Netz einige Zeichen des vorherigen Satzes vergisst. Da jedes der LSTM-Neuronen ein eigenes Gewicht für jedes der Gates hat, ist dies komplexer als andere Arten von NN, und infolgedessen benötigen LSTM in der Regel mehr Ressourcen zum Trainieren und zum Betrieb. LSTM sind nachweislich in der Lage, komplexe Sequenzen zu lernen, z. B. das Schreiben wie Shakespeare oder das Komponieren von Musik.

6.5.3.2.4 Convolutional Neural Network (Faltungsneuronales Netz)

Convolutional Neural Networks (CNN) sind eine Art NN, das mindestens eine Faltungsschicht enthält, um nützliche Informationen aus den Eingaben zu filtern. CNN werden in erster Linie für die Bildverarbeitung [7] und die Videobeschriftung verwendet, sind aber auch für andere Eingabearten wie Audio oder Text einsetzbar (manchmal unter Verwendung rekurrenter Versionen oder rekurrenter CNN). Die Konnektivitätsmuster von CNN für die Bildverarbeitung ähneln der Organisation des visuellen Kortex von Tieren. Im Gegensatz zu RNN ist jedes Neuron in einer bestimmten Schicht des CNN nur mit Neuronen aus einer vorherigen Schicht verbunden und erhält keine Informationen über seinen vorherigen Zustand. Die Menge der Eingabeneuronen eines bestimmten Neurons wird als sein rezeptives Feld bezeichnet. In einem CNN ist es typisch, dass das NN verschiedene Arten von Schichten enthält, einschließlich Faltungsschichten und Pooling-Schichten. Faltungsneuronalen Netze gehen davon aus, dass ihre Eingaben eine gitterartige Topologie haben, wie Bilder (2-D-Gitter) oder Zeitreihen (1-D-Gitter). Die Faltungsschichten behandeln ihre Eingabe als einen Vektor oder eine Matrix (z. B. ein zweidimensionales Bild) und wenden auf die Eingabe eine Faltung an, die ein gleitendes Punktprodukt oder eine Kreuzkorrelation ist, wobei ein Vektor oder eine Matrix, die als Kernel bezeichnet wird, auf die Eingabedaten angewendet wird. Eine bestimmte Faltung ist darauf ausgelegt, bestimmte Merkmale aus der Eingabe zu extrahieren, und erzeugt eine Merkmalskarte für die nächste Schicht im CNN. Mit diesem Ansatz kann das CNN ortsinvariant in Bezug auf die Merkmale in den Eingabedaten werden, was bei realen Bildern sehr wünschenswert ist.

Pooling-Schichten reduzieren die Dimensionen ihrer Eingabedaten entweder lokal (über eine kleine Anzahl von Eingaben) oder global über alle Eingaben der vorherigen Schicht. Durch das Pooling werden unbedeutende Details effektiv herausgefiltert. Mehrere Faltungsschichten und Pooling-Schichten können nacheinander verwendet werden, um bessere Ergebnisse zu erzielen.

Vollständig verknüpfte Schichten werden in der Regel in der Nähe der Ausgabe des CNN verwendet. Bei realen Implementierungen von CNN wird oft ein FFNN am Ende angehängt, um die Daten weiter zu verarbeiten, was hochgradig nichtlineare Abstraktionen ermöglicht.

6.5.3.2.5 Strukturiertes Perzeptron

Das strukturierte Perzeptron ist eine Erweiterung des von NN verwendeten Perzeptron-Algorithmus. Strukturierte Perzeptronen verwenden eine Neuronenschicht mehrmals hintereinander, um verschiedene Teile der Ausgabe vorherzusagen. Das strukturierte Perzeptrontraining spiegelt dieses Konzept wider: Eine einzige Zielvariable wird verwendet, um mehrere Vorhersagen gleichzeitig zu beurteilen.

6.5.3.2.6 Tiefe Boltzmann-Maschine

Boltzmann-Maschinen (BM) sind Netze aus binären Einheiten $\{0,1\}$, die aus einem Satz sichtbarer Einheiten und einem Satz verborgener Einheiten bestehen. Verbindungen bestehen nur zwischen benachbarten Einheiten. Sie sind bidirektional und in der Lage, unbekannte probabilistische Verteilungen zu lernen. Sie sind nützlich bei der Objekt- oder Spracherkennung. BM sind ein generativer Algorithmus.

Die einfachste Form von BM wird als eingeschränkte Boltzmann-Maschine (RBM, en: restricted Boltzmann machine) bezeichnet. Wenn mehrere Schichten von RBM übereinander gelegt werden, nennt man das Netz tiefe Boltzmann-Maschine (DBM, en: deep Boltzmann machine). Tiefe Glaubensnetze (DBN, en: deep belief network) ähneln den DBM, enthalten aber auch Neuronenschichten (d. h. unidirektionale Verbindungen).

6.5.3.2.7 Kapselnetz

Kapselnetze (CapsNet) sind NN, die dynamisches Routing implementieren und auf spärlichen Daten trainieren können. Sie verbessern einige Beschränkungen von CNN, indem sie Neuronen durch Strukturen ersetzen, die als Kapseln bezeichnet werden und aus Gruppen von eng verschachtelten Neuronen bestehen, die alle gleichzeitig aktualisiert werden. Die Verbindungen zwischen den Kapseln wurden ebenfalls verbessert, so dass CapsNets hierarchische Beziehungen besser erfassen und Darstellungen höherer Ordnung erstellen können. CapsNets sind eine unterscheidende Art.

6.5.3.2.8 Generative Adversarial Network

Generative Adversarial Networks (GAN) sind NN, die einen oder mehrere Generatoren enthalten, die versuchen, möglichst repräsentative Samples für den Datensatz zu erzeugen, und einen oder mehrere Diskriminatoren, die versuchen, generierte Samples von echten zu unterscheiden. Die Komponenten des Generators und des Diskriminators werden gemeinsam trainiert, um die inneren Darstellungen des Netzes zu verbessern.

GAN können für verschiedene Aufgaben (z. B. Klassifizierung), aber auch für andere Zwecke wie die Generierung künstlicher Daten oder die Anpassung an einen neuen Bereich verwendet werden.

6.5.3.3 Bayes'sches Netz

Bayes'sche Netze sind graphische Modelle, die zur Erstellung von Vorhersagen über die Abhängigkeiten zwischen Variablen verwendet werden. Sie können verwendet werden, um Wahrscheinlichkeiten für die Ursachen oder Variablen abzuleiten, die zu dem Ergebnis beitragen können. Diese Kausalität ist sehr nützlich für Anwendungen wie die medizinische Diagnose. Bayes'sche Netze sind auch nützlich für die Datenanalyse, den Umgang mit unvollständigen Daten und die Abschwächung der Überanpassung von Modellen an Daten. Bayes'sche Netze beruhen auf der Bayes'schen Wahrscheinlichkeit: Die Wahrscheinlichkeit eines Ereignisses wird als Grad des Glaubens an dieses Ereignis betrachtet. Es gibt verschiedene Bayes'sche statistische Verfahren, die in Verbindung mit Bayes'schen Netzen verwendet werden können, um Kausalität zu bestimmen oder Datenanalysen durchzuführen. Bayes'sche Netze verwenden häufig eine graphische Darstellung von Variablen, die als gerichtete azyklische Graphen bezeichnet werden. Diese Graphen haben die Eigenschaft, dass der Graph, wenn man die Verknüpfungen zwischen einer Variablen x und anderen Variablen verfolgt, niemals zu x zurückführt. Weitere Informationen zu Bayes'schen Netzen können in Verweisung [8] nachgelesen werden.

6.5.3.4 Naiver Bayes'scher Algorithmus

Der naive Bayes'sche Algorithmus ist eine Klassifizierungstechnik, die auf dem Bayes'schen Theorem basiert. Das Theorem bestimmt die Wahrscheinlichkeit des Eintretens eines Ereignisses auf der Grundlage von Wissen über frühere verwandte Ereignisse. Auf diese Weise lässt sich besser bestimmen, ob das Ereignis eintreten wird. Zum Beispiel kann die Diagnose eines medizinischen Zustands genauer gestellt werden, wenn der Lebensstil des Patienten berücksichtigt wird, z. B. sitzend oder aktiv. Bei der naiven Bayes'schen Klassifizierung wird davon ausgegangen, dass die Merkmale der Daten statistisch unabhängig voneinander sind. Dieses Klassifizierungsverfahren hat den Vorteil, dass es relativ einfach ist und nicht auf einen sehr großen Datensatz für das Training angewiesen ist.

6.5.3.5 Stützvektormaschine

Eine Stützvektormaschine ist ein maschinelles Lernverfahren, das häufig für die Klassifizierung und Regression verwendet wird. Ein SVM-Klassifikator markiert Datensamples in zwei verschiedenen Kategorien und ordnet dann neue Dateninstanzen der einen oder anderen Kategorie zu. SVM sind Klassifizierungsalgorithmen mit maximalem Grenzwert. Sie definieren eine Hyperebene, um die Daten in zwei Klassen zu trennen, indem sie den maximalen Abstand zwischen der Klassifizierungsebene und den nächstgelegenen Datenpunkten angeben. Die Punkte, die der Grenze am nächsten liegen, werden als „Stützvektoren“ bezeichnet. Der orthogonale Abstand zwischen den Stützvektoren und der Hyperebene ist die Hälfte des „Margin“ der SVM. SVM verwenden auch Kernel-Funktionen, um Daten aus dem Eingaberaum in einen höherdimensionalen (manchmal unendlichen) Raum abzubilden, in dem die klassifizierende Hyperebene gewählt wird.

Das Training einer SVM beinhaltet die Maximierung der Differenz zwischen den Daten der verschiedenen Kategorien, die sich auf der gegenüberliegenden Seite der Hyperebene befinden. Diese „Hard-Margin“-SVM werden in der Praxis nur selten verwendet. Ein „Hard-Margin“-Klassifikator funktioniert nur, wenn die Daten im eingebetteten Raum linear trennbar sind. Schon ein einziges Sample kann es unmöglich machen, eine trennende Hyperebene zu finden. Im Gegensatz dazu erlauben „Soft-Margin“-Klassifikatoren, dass Datensamples den Rand verletzen (d. h. auf der falschen Seite der Hyperebene liegen). Soft-Margin-Klassifikatoren versuchen, eine maximale Spanne zu erreichen und gleichzeitig Randverletzungen zu begrenzen. Beispiele für die Anwendung von SVM sind die Kategorisierung von unmarkierten Daten und die Verwendung in der Vorhersage und Mustererkennung.

Bei der Verwendung einer SVM für die Regression ist das Ziel das Gegenteil eines SVM-Klassifikators. Bei der SVM-Regression besteht das Ziel darin, so viele Dateninstanzen wie möglich in den Rand einzupassen und gleichzeitig Randverletzungen (d. h. die Samples außerhalb des Randes) zu begrenzen.

6.5.3.6 Entscheidungsbäume

Entscheidungsbäume verwenden eine Baumstruktur von Entscheidungen, um mögliche Ergebnisse zu codieren. Algorithmen für Entscheidungsbäume werden häufig für Klassifizierung und Regression verwendet. Der Baum wird aus Entscheidungsknoten und Blattknoten gebildet. Jeder Entscheidungsknoten hat mindestens zwei Verzweigungen, wobei die Blattknoten die endgültige Entscheidung oder Klassifizierung darstellen. Im Allgemeinen werden die Knoten im Hinblick auf die Entscheidung mit der stärksten Vorhersagekraft geordnet. Die Eingaben müssen in verschiedene Faktoren aufgeteilt werden, um das beste Ergebnis zu bestimmen. Entscheidungsbäume sind mit Flussdiagrammen vergleichbar, bei denen an jedem Entscheidungsknoten eine Frage gestellt werden kann, um zu bestimmen, zu welchem Zweig weitergegangen werden soll.

6.5.4 ML-Optimierungsverfahren

6.5.4.1 Allgemeines

ML-Optimierungsverfahren werden verwendet, um ein ML-Modell an ML-Daten anzupassen. Die Herausforderung bei der ML-Optimierung besteht darin, die optimalen Parameter eines ML-Modells zu finden, um eine festgelegte Verlustfunktion für die vorliegenden ML-Daten zu minimieren. Die meisten der folgenden Verfahren stützen sich auf eine Verlustfunktion (manchmal auch „Kostenfunktion“ genannt), um festzustellen, ob das Modell zu einer akzeptablen Lösung konvergiert oder divergiert. Die vorherrschenden ML-Optimierungsverfahren werden in den folgenden Unterabschnitten beschrieben.

Neben Optimierungsverfahren können auch andere Techniken die Anpassung des ML-Modells an die Daten verbessern. Ein solcher Aspekt ist die Wahl einer besseren Verlustfunktion. Die Regularisierung ist beispielsweise eine Technik, die zur Verringerung der Überanpassung und der hohen Vorhersagevarianz eingesetzt wird, indem neue Terme in die Verlustfunktion eingeführt werden. Dabei wird ein komplexes Modell benachteiligt und ein allgemeineres, weniger präzises Modell begünstigt.

6.5.4.2 Gradientenabstiegsverfahren

Der Gradientenabstieg ist eine iterative Technik, um den Mindestwert einer Funktion zu bestimmen: Während der gesamte Datensatz als Einheit eingespeist wird, werden die Parameter Schritt für Schritt aktualisiert,

und zwar in der gleichen Richtung wie die Ableitung erster Ordnung (Gradient) der Funktion. Die Lösung ist ein globales Optimum, wenn die Zielfunktion konvex ist. Wenn aktualisierte Parameter über zufällig ausgewählte kleinere Datenmengen berechnet werden, anstatt über den gesamten Datensatz als Einheit, wird dies als stochastischer Gradientenabstieg (SGD) bezeichnet. SGD reduziert die Berechnungskosten, riskiert aber eine Lösung, die in einem lokalen Minimum gefangen ist. Die Iterationsschrittweite des Gradienten wird als Lernrate bezeichnet.

Momentumverfahren führen die Geschwindigkeit als Variable ein, um die Richtung und die Geschwindigkeit der Änderung eines Parameters in seinem Raum darzustellen; durch diesen Vorgang bleibt der Einfluss der vorherigen Aktualisierungsrichtung für die nächste Iteration erhalten. Beim beschleunigten Gradientenabstieg nach Nesterov wird ein adaptives Momentum angewendet, das zu einer schnelleren Konvergenz führt.

Die Anpassung der Lernrate durch Variation der Schrittweite jeder Parameteraktualisierung verbessert die Verfahren des Gradientenabstiegs. Dies wird als adaptive Lernrate bezeichnet. Ein Beispiel ist AdaGrad (adaptiver Gradientenalgorithmus), der die Lernrate automatisch anpasst.

Redundante Informationen in den Trainingsdaten können bei SGD-Verfahren zu einer langsamen Konvergenz führen. Der stochastische Durchschnittsgradient enthält einen Parameter zur Aufzeichnung der Summe der aktuellsten Gradienten. Diese Summe wird bei jeder Iteration nach dem Zufallsprinzip aktualisiert und ersetzt bei jedem Zyklus den alten Gradienten durch einen neuen. Mit dieser Technik wird eine Verringerung der Varianz erreicht.

6.5.4.3 Newtonverfahren

Das Newtonverfahren verwendet die Ableitungsmatrix erster Ordnung (Gradient) und die Ableitungsmatrix zweiter Ordnung, auch bekannt als Hesse-Matrix, um die Zielfunktion durch eine quadratische Funktion zu approximieren. Das Newtonverfahren passt die lokale Oberfläche der aktuellen Position mit einer quadratischen Fläche an; dies steht im Gegensatz zum Gradientenabstiegsverfahren, das die aktuelle lokale Oberfläche mit einer Ebene anpasst.

6.5.4.4 Verfahren der konjugierten Gradienten

Das CG-Verfahren (en: conjugate gradient) berechnet das Hessesche Vektorprodukt, ohne dass die Hesse-Matrix wie beim Newtonverfahren direkt berechnet wird. CG erzeugt eine neue Suchrichtung nur unter Verwendung des vorherigen Vektors. Dadurch wird die Berechnung der Inversen der Hesse-Matrix vermieden.

6.5.4.5 Gaußprozesse

Ein Gaußprozess ist eine Sammlung von Zufallsvariablen mit konsistenten, gemeinsamen Gaußverteilungen. Der Gaußprozess basiert auf der Bayes'schen Theorie und dem statistischen Lernen. Seine Vorteile liegen in der flexiblen nichtparametrischen Inferenz und der starken Interoperabilität auf Kosten der Komplexität und des großen Speicherbedarfs.

6.5.4.6 Kurvenanpassung nach dem Verfahren der kleinsten Quadrate

Die Anpassung der kleinsten Quadrate ist eine Technik zur Anpassung einer Polynomfunktion an gegebene Daten durch Minimierung der Quadratsumme aus der Differenz zwischen den Ausgaben der Polynomfunktion und den gegebenen Daten. Die Ordnung des Polynoms muss zu den Messdaten passen, um gute Ergebnisse zu erzielen. Die Kurvenanpassung nach dem Verfahren der kleinsten Quadrate kann schubweise für einen vollständigen Datensatz oder in einem rekursiven Verfahren mit wachsenden Datensätzen durchgeführt werden.

6.5.4.7 Maximum-Likelihood-Schätzung

Die Schätzung der größten Wahrscheinlichkeit (Maximum Likelihood) ist ein Verfahren zur Schätzung der Parameter einer Wahrscheinlichkeitsverteilung durch Maximierung der Wahrscheinlichkeitsfunktion. Um ein probabilistisches Modell aus beobachteten Daten zu erstellen, wird bei der Maximum-Likelihood-Schätzung die Hypothese gewählt, die die Wahrscheinlichkeit der beobachteten Daten unter Berücksichtigung der Hypothese maximiert. In ihrer allgemeinen Form verwendet sie kein apriorisches Wissen, um eine bestimmte Hypo-

these zu begünstigen. Bei Maximum-Likelihood-Schätzverfahren wird aus rechnerischen Gründen häufig der Logarithmus der Wahrscheinlichkeit verwendet.

6.5.4.8 Erwartungsmaximierung

Die Erwartungsmaximierung ist ein iteratives Verfahren zum Lernen von Modellparametern mit verborgenen, in den Daten nicht beobachtbaren Variablen. Es besteht aus einem Wechsel zwischen Erwartungsschritten (Schätzung der verborgenen Variablen auf der Grundlage der aktuellen Parameter) und Maximierungsschritten (erneute Schätzung der Parameter, um die Wahrscheinlichkeit der Daten angesichts des aktuellen Werts der verborgenen Variablen zu optimieren), bis zur Konvergenz.

6.5.5 ML-Bewertungsmetriken

6.5.5.1 Allgemeines

Die ML-Aufgaben (siehe 6.2) bestimmen in der Regel geeignete Bewertungsmetriken. Das Verständnis des Anwendungsfalls eines Modells ist entscheidend für die Auswahl geeigneter Bewertungsmetriken. Um die Leistung eines Modells angemessen auszudrücken und zu untersuchen, können mehrere Metriken erforderlich sein. Die Verwendung einer einzelnen Metrik, selbst einer aggregierten Metrik wie dem F1-Score, ist ohne gründliche Prüfung der zugrunde liegenden Ergebnisse riskant. Darüber hinaus ist das Ungleichgewicht der Klassen in den Trainingsdaten ein Störfaktor, der verschiedene Metriken verzerren kann.

Viele dieser Metriken, insbesondere bei der Klassifizierung, beruhen auf den Konzepten von richtig-positiv (T_P , en: true positive, d. h. Samples, die richtig erkannt wurden), falsch-positiv (F_P , d. h. Samples, die erkannt wurden, obwohl sie nicht hätten erkannt werden sollen), von richtig-negativ (T_N , en: true negative, d. h. Proben, die richtig nicht erkannt wurden) und von falsch-negativ (F_N , d. h. Proben, die nicht erkannt wurden, obwohl sie hätten erkannt werden sollen).

Es gibt viele Metriken zur Darstellung der Leistung trainierter Modelle, z. B.:

- zur Bewertung von Klassifizierungsalgorithmen können die Genauigkeit, die Betriebseigenschaften des Empfängers (ROC), die Konfusionsmatrix, die Präzision, die Sensitivität und der F1-Score verwendet werden;
- der mittlere absolute Fehler (MAE, en: mean absolute error), die Wurzel des mittleren quadratischen Fehlers, der relative absolute Fehler, der relative quadratische Fehler, der mittlere Nullfehler und das Bestimmtheitsmaß sind gängige Metriken für Regressionsmodelle;
- zu den Bewertungsmetriken für Clustermodelle gehören der durchschnittliche Abstand zum Clusterzentrum, der durchschnittliche Abstand zum anderen Zentrum, die Anzahl der Punkte und der maximale Abstand zum Clusterzentrum.

Die folgenden Unterabschnitte beschreiben Beispiele für solche Metriken und erheben keinen Anspruch auf Vollständigkeit.

6.5.5.2 Präzision, Recall, Sensitivität und Spezifität

Eine Reihe von Metriken kann direkt als Prozentsatz zwischen T_P , F_P , T_N und F_N berechnet werden:

- die Richtig-Positiv-Rate (TPR) ist der Prozentsatz der richtig Positiven (T_P) unter allen tatsächlich Positiven ($T_P + F_N$). Ihr Gegenstück heißt Falsch-Negativ-Rate (FNR);
- die Richtig-Negativ-Rate (TNR) ist der Prozentsatz der richtig Negativen (T_N) unter allen tatsächlich Negativen ($T_N + F_P$). Ihr Gegenstück heißt Falsch-Positiv-Rate (FPR);
- der positive prädiktive Wert (PPV, en: positive predictive value) ist der Prozentsatz der richtig Positiven (T_P) unter allen vorhergesagten Positiven ($T_P + F_P$);