Machine-Learning

Autoren:

Prof. Dr. Andreas Knoblauch

Andreas Nolle, M.Sc.

Hochschule Albstadt-Sigmaringen

Modul 30200
Machine-Learning

Studienbrief 1: Einführung & Wiederholung

Autoren:
Prof. Dr. Andreas Knoblauch
Andreas Nolle, M.Sc.

1. Auflage

Hochschule Albstadt-Sigmaringen

Studiengang Data Science
Johannesstraße 3
72458 Albstadt

1. Auflage (2017-02-10)

Didaktische und redaktionelle Bearbeitung:
Johannes Maneljuk (M. A. Rhetorik/Germanistik)

Um die Lesbarkeit zu vereinfachen, wird auf die zusätzliche Formulierung der weiblichen Form bei Personenbezeichnungen verzichtet. Wir weisen deshalb darauf hin, dass die Verwendung der männlichen Form explizit als geschlechtsunabhängig verstanden werden soll.

Inhaltsverzeichnis

Einleitung zu den Studienbriefen

I. Abkürzungen der Randsymbole und Farbkodierungen

Kontrollaufgabe	✔
Quelltext	</>
Übung	✎

II. Zu den Autoren

Prof. Dr. Andreas Knoblauch ist seit 2013 Professor für Technische Informatik an der Hochschule Albstadt-Sigmaringen. Nach einem Diplom-Abschluss im Jahr 2000 in Informatik und Elektrotechnik an der Universtität Ulm promovierte er dort 2003 über Neuronale Netze und Assoziativspeicher. Nach zwei PostDoc-Jahren 2003–2005 im Bereich Neuro-Roboter und Sprachverarbeitung am Institut für Neuroinformatik der Universität Ulm und der Cognition and Brain Sciences Unit am MRC Cambridge (UK), arbeitete er von 2005 bis 2013 als Senior Scientist am Honda Research Institute Europe in Offenbach/Main an Neuronalen Netzen, Systemen zur Situationserkennung und Verfahren für autonomes maschinelles Lernen, was bis heute seinen Forschungsinteressen entspricht.

Andreas Nolle absolvierte 2010 das Master-Studium der Wirtschaftsinformatik an der Hochschule Albstadt-Sigmaringen und der Hochschule Ravensburg-Weingarten. Nachdem er kurze Zeit als Software-Entwickler tätig war, wechselte er 2011 an die Hochschule Albstadt-Sigmaringen, wo er neben der Lehre, speziell in den Bereichen Data-Warehousing, Business-Intelligence und Machine-Learning, auch in der Forschung tätig ist. Er ist seit Anfang 2014 zudem externer Doktorand an der Universität Mannheim mit dem Forschungsschwerpunkt „Federated Ontology Debugging“ bei seinem Doktorvater Prof. Dr. Heiner Stuckenschmidt.

III. Modullehrziele

Als „Machine-Learning“ (zu dt. „maschinelles Lernen“) bezeichnet man Methoden zur Realisierung von automatisierten Entscheidungsverfahren, insbesondere in Verbindung mit größeren Datenmengen. Dabei lernt das System aus Beispielen und „erkennt“ Gesetzmäßigkeiten, so dass es nach Beendigung der Lernphase in der Lage ist, zu verallgemeinern und auch neue unbekannte Daten zu beurteilen. Zu den klassischen Beispielen des maschinellen Lernen gehören Klassifikation, Regression, Clustering oder Faktoranalyse.

Verfahren des maschinellen Lernens haben ein sehr breites Anwendungsspektrum in diversen Themenfeldern der Informatik. Aus dem weiten Spektrum möglicher Anwendungen seien hier exemplarisch automatisierte Diagnoseverfahren, Erkennung von Kreditkartenbetrug, Aktienmarktanalysen, Klassifikation von DNA-Sequenzen, Sprach-, Schrift- und Bilderkennung, und autonome intelligente Systeme genannt.

Aufbauend auf *Modul 10200: Mathematical Foundations for Data Science*, *Modul 10300: Data-Mining* und *Modul 20400: Decision Support* vermittelt Ihnen dieses Modul neben den theoretischen Grundlagen einen systematischen, vereinheitlichenden Überblick über Methoden des maschinellen Lernens und deren Anwendungsmöglichkeiten. Neben dem Kennen und Verstehen eines breiten „Arsenals“ von Methoden vermögen Sie darüber hinaus – je nach Problemstellung – geeignete Verfahren des maschinellen Lernens auszuwählen, anzuwenden und zu evaluieren. Wir konzentrieren uns hier auf Methoden des überwachten Lernens, wobei wir uns im Wesentlichen auf das Werk von Bishop (2006) stützen. Im Rahmen von Kontroll- und Übungsaufgaben erlangen Sie zudem die erforderlichen Kenntnisse und Fähigkeiten, ausgewählte Algorithmen in `Python` zu implementieren und auf praxisorientierte Datensätze anzuwenden. Ergänzend zum Inhalt dieses Moduls finden Sie die erforderlichen mathematische Grundlagen sowie weiterführende Informationen, wie etwa Beweise, weitere Details und Beispiele, in dem Ihnen online zur Verfügung gestellten Grundlagen-Skript zur Mathematik („*Vorlesungsskript Mathematik I*“) von Herrn Prof. Dr. Knoblauch.

Das vorliegende Modul gliedert sich dabei in folgende Punkte:

1. Einführung & Wiederholung
2. Einfache lineare Modelle
3. Neuronale Netze
4. Kernel-Methoden
5. Grafische Modelle
6. Lernen probabilistischer Modelle & Modellkombination

Nach Abschluss dieses Moduls sind Sie in der Lage, maschinelles Lernen und weitere Begrifflichkeiten entsprechender Teilgebiete und verwandter Fachbereiche zu deuten, zu interpretieren und entsprechend einzuordnen. Überdies werden Ihnen unterschiedliche Methoden und Verfahren des überwachten maschinellen Lernens vermittelt, welche Sie qualifiziert beurteilen und im Kontext praktischer Anwendungs- bzw. Problemfälle mithilfe geeigneter Techniken auswählen, (in `Python`) implementieren und anwenden können.

Studienbrief 1 Einführung & Wiederholung

1.1 Lernergebnisse

Der vorliegende Studienbrief vermittelt Ihnen das erforderliche theoretische Grundwissen, praktische Anwendungsfälle maschinellen Lernens zu identifizieren sowie den Bedarf und praktischen Nutzen entsprechender Verfahren argumentativ zu vertreten. Ferner können Sie den Begriff „maschinelles Lernen“ und verwandte Begriffe differenziert auslegen und in den jeweiligen Kontext einordnen. Neben grundlegenden mathematischen Methoden und Theorien wissen Sie zudem mit typischen Problemen maschineller Lernmethoden und zugrunde gelegter Daten umzugehen. Sie sind darüber hinaus mit den grundlegenden Modelltypen des überwachten Lernens vertraut und können diese durch Einsatz geeigneter Methoden entsprechend vergleichen und evaluieren.

1.2 Advance Organizer

Um mittels künstlicher Intelligenz (KI) die richtigen Entscheidungen treffen können, muss einem technischen System (Maschine) entsprechendes Wissen angelernt werden. Die Entwicklung maschineller Lernverfahren (Algorithmen) wird dabei als eines der bedeutendsten Teilbereiche der künstlichen Intelligenz angesehen. Ziel dabei ist, Systeme lernfähig zu machen, so dass diese mittels vorhandener Daten selbstständig Wissen generieren und, darauf basierend, ein gegebenes Problem lösen können.

Zu Beginn des vorliegenden Studienbriefs wird der Bereich des maschinellen Lernens mit Bezug auf praktische Anwendungsfälle eingeführt. Ferner werden grundlegende mathematische Methoden und Theorien, welche die in den folgenden Studienbriefen dieses Moduls angeführten Lernalgorithmen aufgreifen, vermittelt. Neben typischen Problemen maschineller Lernmethoden, wie etwa der Überanpassung, werden ebenfalls Erschwernisse und Herausforderungen, bezogen auf die zur Verfügung stehenden Daten (Stichwort: Fluch der Dimensionalität), diskutiert. Das wesentliche Ziel in praktischen Anwendungsfällen ist, das zugrunde gelegte Modell und dessen Parameter so zu bestimmen, dass darüber möglichst optimale Prognosen erstellt werden können. Für die Bewertung einzelner und den Vergleich unterschiedlicher Modelle werden Sie daher mit geeigneten Evaluationsmethoden vertraut gemacht.

1.3 Einführung in maschinelles Lernen

Lernen & Intelligenz

Bevor wir in die Details des maschinellen Lernens einsteigen, wollen wir uns der Frage widmen, was eigentlich „Lernen“ ist. So ist das Lernen aus biologischer Sicht eines der charakteristischen Merkmale von intelligentem Verhalten. Der Lernprozess selbst umfasst dabei

- neben dem Erwerben von neuem Wissen
- sowie dessen Einordnung in bereits vorhandenes Wissen,
- auch die Erlangung von motorischen und kognitiven Fähigkeiten durch Anleitung oder praktische Anwendung,
- die Organisation (neuen) Wissens in Form von allgemein gültigen, effektiven Repräsentationen,
- das Aufdecken neuer Erkenntnisse und Theorien durch Beobachten und Experimentieren,
- sowie das Transformieren, Neu- und Umorganisieren von bereits vorhandenem Wissen.

Zwar existieren in einschlägiger Literatur unterschiedliche Versuche, Lernen eindeutig zu präzisieren, allerdings gestaltet sich dies äußerst schwierig. Ähnlich diffizil ist auch der Begriff „Intelligenz“ zu fassen, weshalb auch hier keine allgemein geteilte Definition existiert.

Motivationen maschinellen Lernens

Im Forschungsgebiet der künstlichen Intelligenz (KI), welches im Allgemeinen zum Ziel hat, die menschliche bzw. biologische Intelligenz mit Software nachzuahmen bzw. zu imitieren, hat das Teilgebiet des „maschinellen Lernens (ML)“ (engl. machine learning) folgende Motivationen:

- Untersuchung von Prinzipien des menschlichen Lernens sowie dessen Simulation mit Hilfe von operationalen Modellen (kognitionsorientiert)
- Theoretische Untersuchung von möglichen Lernmethoden und Operationalisierung von Schlußfolgerungen in Lernalgorithmen (theorieorientiert)
- Arbeitserleichterung von Usern bei der Durchführung eines vordefinierten Aufgabentyps durch den Einsatz lernfähiger Systeme (aufgaben- bzw. praxisorientiert)

Intention maschinellen Lernens

Damit Systeme mit künstlicher Intelligenz die richtigen Entscheidungen treffen können, muss entsprechendes Wissen angelernt werden. Bedenkt man nun, dass der Mensch den Maschinen, insbesondere bezogen auf die Lernfähigkeit, weit überlegen ist, so kann das Erforschen von Methodiken des Lernens und die Entwicklung von maschinellen Lernverfahren als eines der bedeutendsten Teilbereiche der künstlichen Intelligenz angesehen werden. Ziel des maschinellen Lernens ist dabei die Entwicklung von Lernmethoden (Algorithmen), um damit technische Systeme (Maschinen) lernfähig zu machen, so dass diese auf Basis von Inputdaten selbstständig Wissen generieren können, das sie in die Lager versetzt, ein gegebenes Problem zu lösen bzw. besser zu lösen als zuvor.

Einordnung in angrenzende Themenbereiche

Der Bereich des maschinellen Lernens ist eng verwandt mit „Data-Mining“, welches insbesondere auf das Aufdecken von neuen Mustern, Zusammenhängen, Regel- und Gesetzmäßigkeiten fokussiert. So werden im Bereich des maschinellen Lernens entwickelte Algorithmen im Data-Mining angewendet. Überdies kann der Prozess der „Knowledge Discovery in Databases (KDD)“ (zu dt. Wissensentdeckung in Datenbanken), welcher das Data-Mining um entsprechende Schritte des Data-Preprocessings zur Aufbereitung und Transformation der Daten ergänzt, zur Generierung und Vorverarbeitung von Lerndaten für das maschinelle Lernen eingesetzt werden.

prädestinierte Anwendungsfälle maschinellen Lernens

Das reine „Auswendiglernen“ ist, im Gegensatz zum Menschen, für Computer durch einfaches Abspeichern kein Problem. Ebenso kann das Durchführen von Berechnungen und automatisierbaren Prozessen mithilfe einer Programmiersprache problemlos umgesetzt werden. Während diese Fälle für die KI nicht von Interesse sind, existiert eine Reihe von Anwendungsfällen, bei denen der Einsatz maschinellen Lernens teils unabdingbar ist. Ein Beispiel hierfür könnte etwa die Identifikation von handschriftlichen Postleitzahlen auf Briefen sein. Das Ziel wäre nun, eine Maschine zu entwickeln, welche zu einem Bildausschnitt je Ziffer (wie etwa in Abbildung 1.1 illustriert) als Input die zugehörige Nummer aus dem Wertebereich $[0, 9]$ identifiziert. Während es für Menschen ein Leichtes ist, diese Aufgabe zu bewältigen, ist es für eine Maschine aufgrund der großen Varianz individueller Handschriften ein nicht triviales Problem.

das Trainingsset

Für die Verwendung von Ansätzen des maschinellen Lernens werden die Parameter eines adaptiven Modells mithilfe einer möglichst großen Menge mit N Ziffern $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$, welche auch als „Trainingsset“ (oder Trainingsdatensatz) bezeichnet wird, abgestimmt und damit das erforderliche Grundwissen „antrainiert“. Jedes

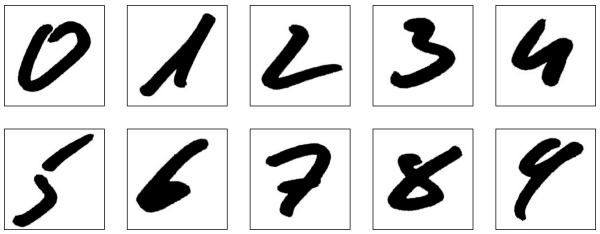

Abb. 1.1: Beispiele handschriftlicher Ziffern

Element $\mathbf{x}_i$, wobei $i = 1..N$, eines Trainingssets ist dabei als Datenpunkt mit D Elementen aufzufassen und kann somit als Eingabevektor (engl. input vector) in Form eines D-dimensionalen Spaltenvektors respektive transponierten Zeilenvektors repräsentiert werden:

$$\mathbf{x}_i = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} = (x_1, ..., x_D)^{\mathrm{T}}. \tag{1.1}$$

Das Trainingsset mit N D-dimensionalen Datenpunkten $\mathbf{x}_i$ mit $i = 1..N$ kann daher auch als Datenmatrix $\mathbf{X}$ bezeichnet werden und ist definiert über

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix} = \begin{pmatrix} x_{1,1} & ... & x_{1,D} \\ x_{2,1} & ... & x_{2,D} \\ \vdots & \ddots & \vdots \\ x_{N,1} & ... & x_{N,D} \end{pmatrix}. \tag{1.2}$$

Anmerkung zur mathematischen Notation

In diesem Modul erfolgt die mathematische Notation in Anlehnung an Bishop (2006, S. xi ff). Vektoren und Matrizen werden dabei in Serifenschrift gesetzt und gefettet, wobei Vektoren (wie etwa $\mathbf{x}$) durch Kleinbuchstaben und Matrizen (wie etwa $\mathbf{X}$) durch Großbuchstaben symbolisiert werden. Variablen (wie x) werden im Gegensatz hierzu in geschwungener Serifenschrift gesetzt und kursiv geschrieben. Für eine bessere Lesbarkeit von Dezimalzahlen in Punkt- und Vektorangaben verwenden wir für die Darstellung von Dezimalzahlen die englische Form (also bspw. $5/2 = 2.5$).

Repräsentation in Form von Vektoren

Bezogen auf das Erkennen von handschriftlichen Postleitzahlen wird somit jedes Bild einer einzelnen Ziffer durch ein $\mathbf{x}_i$ als Eingabevektor repräsentiert. Dies könnte beispielsweise dadurch erreicht werden, dass der Grauwert eines jeden Pixels als Dimension[1] des Vektors aufgetragen wird. So ergibt sich für jedes in Abbildung 1.1 gezeigte Bild einer Ziffer mit 200×200 Pixeln ein Eingabevektor mit 40.000 Dimensionen (Elementen). Die Kategorien C_j, welche den einzelnen Ziffern zugeordnet werden können, entsprechen der Menge {*null*, *eins*, *zwei*, *drei*, *vier*, *fünf*, *sechs*, *sieben*, *acht*, *neun*}. Um nun eine Zuordnung der Kategorien für die einzelnen Bilder des Trainingssets auszudrücken, wird für jedes Ziffernbild $\mathbf{x}_i$ ein sogenannter Targetvektor $\mathbf{t}_i$ verwendet. Ein Targetvektor ist dabei als Zielmuster und damit als gewünschter Antwortvektor zu sehen. Eine gängige Methode zur Repräsentation unterschiedlicher Kategorien in Form von Vektoren ist das sogenannte 1-of-K-Kodierungsschema. Abhängig von der Anzahl K an Kategorien wird die Länge eines jeden Targetvektors $\mathbf{t}$ bestimmt, wodurch dieser wie folgt definiert ist:

$$\mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_K \end{bmatrix} = (t_1, ..., t_K)^{\mathrm{T}}. \tag{1.3}$$

[1] Als Dimension ist hierbei nicht die Dimension der Datenmatrix oder des Vektors selbst zu verstehen, sondern vielmehr die Dimensionen des Raums, in dem der Vektor aufgetragen werden kann.

Wird nun einem Ziffernbild (und damit jedem Eingabevektor $\mathbf{x}_i$) die Kategorie j zugeordnet, dann wird allen Elementen t_k des zugehörigen Targetvektors $\mathbf{t}_i$ der Wert 0 zugewiesen, während das Element t_j als Ausnahme den Wert 1 erhält. Dem dritten Bildausschnitt aus Abbildung 1.1, welcher die Ziffer 2 zeigt, würde über folgenden Targetvektor $\mathbf{t}$ die Kategorie $zwei$ zugewiesen werden:

$$zwei \coloneqq (0,0,1,0,0,0,0,0,0,0)^{\mathrm{T}}.$$

Ergänzend kann hierbei der Wert eines jeden Elements t_k auch als Wahrscheinlichkeit betrachtet werden, mit der das Bild $\mathbf{x}_i$ der Kategorie C_j zuzuordnen ist.

Anwendung und Ergebnis maschineller Lernalgorithmen

Das Ergebnis eines maschinellen Lernalgorithmus kann als Funktion $\mathbf{y}(\mathbf{x})$ ausgedrückt werden, die ein „neues“ Ziffernbild $\mathbf{x}$ als Eingabevektoren fordert und einen Ausgabevektor (engl. output vector) $\mathbf{y}$ generiert, der dieselbe Kodierung wie der Targetvektor besitzt. Die Präzisierung der Funktion $\mathbf{y}(\mathbf{x})$ erfolgt auf Basis des Trainingssets in der „Trainingsphase“, welche treffenderweise auch als „Lernphase“ bezeichnet wird. Nachdem ein Modell trainiert ist, kann dieses zur Bestimmung der Klassen- bzw. Kategorienzugehörigkeit von neuen Bildern mit Ziffern, dem sogenannten „Testset“, eingesetzt werden. Die Fähigkeit, neue Elemente, welche von denen des Trainingsset abweichen, korrekt einer Kategorie zuzuordnen, wird auch als „Generalisierung“ bezeichnet. In praktischen Anwendungsfällen ist die Generalisierung eines der zentralen Ziele der Mustererkennung, da das Trainingsset nur einen geringen Bruchteil aller möglichen Eingabevektoren umfassen kann.

Preprocessing der Inputdaten

Typischerweise müssen die Inputdaten einem Preprocessing unterzogen und dabei entsprechend transformiert werden, so dass das Problem des maschinellen Lernens bzw. der Mustererkennung möglichst vereinfacht wird. Für das zuvor angeführte Beispiel der Erkennung von handschriftlichen Ziffern werden die jeweiligen Bilder, sowohl des Training- als auch des Testsets, üblicherweise auf dieselbe Größe skaliert. Aufgrund der einheitlichen Skalierung aller Ziffernbilder kann die Vielfältigkeit der einzelnen Klassen (Kategorien) stark reduziert und damit die nachfolgende Anwendung von Algorithmen zur Mustererkennung effizienter – im Sinne einer Komplexitätsminderung aber ebenso einer Performanceoptimierung – gestaltet werden. Die auf den Elementen des Trainingsset durchgeführten Preprocessing-Operationen müssen zwangsläufig in derselben Art und Weise auf den Elementen des Testsets angewendet werden. Da die Anzahl an Merkmalen (Vektorelemente) auf die als relevant betrachteten Merkmale reduziert wird, kann diese Art des Preprocessings auch als Dimensionsreduktion gesehen werden. Hierbei ist allerdings darauf zu achten, dass wichtige Informationen nicht verloren gehen und damit die Genauigkeit der Ergebnisse negativ beeinflusst wird. Die Phase des Preprocessings wird daher treffenderweise auch als „Feature Extraction“ (zu dt. Extraktion bzw. „Freistellung“ von Merkmalen) bezeichnet.

1.3.1 Praktische Anwendungsfälle maschinellen Lernens

Durch die Eigenschaft, das Wissen und die Fähigkeiten künstlicher Systeme zu erweitern, finden Methoden des maschinellen Lernens in vielen Bereichen Anwendung. Neben klassischen Bereichen, wie der bereits erläuterten Erkennung handschriftlicher Ziffern, lernenden Spamfiltern für den alltäglichen E-Mail-Verkehr oder auch Internetsuchmaschinen, existieren auch innovative Ansätze, etwa aus den Gebieten der Robotik oder der Luft- und Raumfahrt. Aktuelle Forschungsgebiete, bei denen maschinelle Lernalgorithmen zum Einsatz kommen, umfassen etwa die Entwicklung autonomer Fahrzeuge, welche sich auch in Umgebungen zurechtfinden, die für das System unbekannt sind. Weitere Anwendungsgebiete sind etwa:

- Erkennung von Betrugsfällen (engl. fraud detection), beispielsweise bei Kreditkarten
- Erkennung von Hackingangriffen
- autonome Robotik
- Kognitionswissenschaften, wie etwa zur Spracherkennung
- Gesichtserkennung
- Börsenanalyse
- medizinische Diagnose
- Bioinformatik
- adaptive Systeme in der Spieltheorie
- Verwaltung und Verarbeitung enormer Datenmengen (Stichwort: Big Data), wie beispielsweise Sensordaten
- Data-Mining
- Business-Intelligence

1.3.2 Formen des Schlußfolgerns

Formale Grundlage von Lernalgorithmen ist das logische Schlussfolgern (Inferenz), wobei hier zwischen folgenden Formen unterschieden wird:

- **Deduktion**
  Die Deduktion ist der Schluss vom Allgemeinen auf das Besondere. Konkret heißt dies, dass aus einer gegebenen Menge allgemeiner Aussagen (Regeln), welche zusammengenommen die Hypothese H bilden, und einer Menge an Bedingungen bzw. Einzelfällen (Prämissen) P eine Menge K an zwingend geltenden Konklusionen abgeleitet wird. Formal ausgedrückt bedeutet dies:
  $$H \cup P \vdash K \tag{1.4}$$
  Beispiel einer Deduktion wäre etwa, wenn aus der allgemeinen Regel, dass alle Frösche quaken, und der Prämisse, dass Kermit ein Frosch ist, der Schluss gezogen wird, dass Kermit quakt. Formal in Prädikatenlogik ausgedrückt:
  $$\{\forall x \mid Frosch(x) \rightarrow quakt(x), Frosch(Kermit)\} \vdash \{quakt(Kermit)\}$$
- **Induktion**
  Im Gegensatz zur Deduktion beschreibt die Induktion das Schließen vom Besonderen auf das Allgemeine. Ausgehend von einer Menge an Prämissen P und einer Menge an Konklusionen K wird somit eine Menge an allgemeinen Regeln H abgeleitet:
  $$P \cup K \prec H \tag{1.5}$$
  Bezogen auf die Prämisse, dass Kermit ein Frosch ist, und die Konklusion, dass Kermit quakt, wäre ein induktiver Schluss daraus, dass alle Frösche quaken:
  $$\{Frosch(Kermit), quakt(Kermit)\} \prec \{\forall x \mid Frosch(x) \rightarrow quakt(x)\}$$
  Zwar ist die allgemeine Regel, dass alle Frösche quaken, durchaus wahrscheinlich, allerdings nicht zwingend. So gilt diese Verallgemeinerung zwar für alle uns bekannten Frösche, würde jedoch ein weiterer Frosch,

speziell ein Krallenfrosch[2] mit als Belegfall aufgenommen werden, so wäre diese Verallgemeinerung nicht mehr haltbar.

- **Abduktion**
  Eine weitere Form des Schlussfolgerns ist das Schließen von einer Menge allgemeiner Regeln H und einer Menge an Konklusionen K auf eine Menge an Prämissen P:
  $$H \cup K \succ P \tag{1.6}$$
  Im Kontext unseres Beispiel wäre eine Abduktion:
  $$\{\forall x \mid Frosch(x) \rightarrow quakt(x), quakt(Kermit)\} \succ \{Frosch(Kermit)\}$$
  Zwar ist aus der allgemeinen Regel, dass alle Frösche quaken, und der Konklusion, dass Kermit quakt, die Schlussfolgerung, dass Kermit auch ein Frosch ist, zwar möglich, jedoch nicht unbedingt wahrscheinlich. So könnte Kermit etwa auch eine Ente sein. Da bei einer Abduktion kein gesicherter Belegfall vorliegt, ist die Folgerung lediglich richtungsweisend und damit ziemlich unsicher.

Soundness & Completeness

In Bezug auf eine Menge an allgemeinen Regeln werden diese nur dann als korrekt (engl. sound) bezeichnet, wenn, informal gesagt, aus wahren Aussagen nichts Falsches abgeleitet wird. Komplementär wird eine Menge an allgemeinen Regeln als vollständig (engl. complete) bezeichnet, wenn, informal gesagt, aus dieser Menge alle wahren Aussagen abgeleitet werden können. Auf eine formale Definition der beiden Begriffe Korrektheit (engl. soundness) und Vollständigkeit (engl. completeness) wird an dieser Stelle verzichtet.

ML fokussiert auf induktives Schließen

Zusammenfassend lässt sich sagen, dass nur der deduktive Schluss erfahrungsunabhängig und auch zwingend ist. Die Abduktion und Induktion hingegen sind nicht zwangsläufig wahrheitserhaltend und repräsentieren damit nur potenzielle logische Schlüsse. Während die Deduktion insbesondere in der mathematischen Logik und in formalen Systemen Anwendung findet, konzentrieren sich Methoden des maschinellen Lernens vornehmlich auf den induktiven Schluss.

1.3.3 Klassifizierung von Lernmethoden

Wie bereits erwähnt, streben Methoden und Algorithmen des maschinellen Lernens das Verstehen von kontextbezogenen Zusammenhängen, wie etwa durch Erkennung von Mustern oder Abhängigkeiten, und damit auch das Erzeugen von neuem Wissen im Sinne von induktivem Lernen an. Das eigentliche Lernproblem hierbei ist, anhand von einer kleinen Teilmenge von Beispieldaten präzise Regeln zur generischen Beschreibung der Daten zu finden. Die entsprechenden Lernmethoden hierfür lassen sich prinzipiell in folgende drei Bereiche einteilen:

Überwachtes Lernen

Zum Bereich des überwachten Lernens (engl. supervised learning) zählen diejenigen Anwendungsfälle, bei denen das Trainingsset sowohl Eingabe- als auch die zugehörigen Targetvektoren umfasst. Eine präzise Beschreibung der Daten wäre nun eine Funktion, welche zu den Eingabevektoren korrekte Ausgabewerte entsprechend der gegebenen Targetvektoren liefert. Die eigentliche Aufgabe besteht nun darin, anhand von gegeben Paaren mit Eingabevektoren $\mathbf{x}_i$ und Targetvektoren $\mathbf{t}_i$ in der Form $(\mathbf{x}_i, \mathbf{t}_i)$, wobei $i = 1..N$, eine deterministische Funktion zu finden, welche die gegebenen Zuweisungen generalisiert und damit Ergebnisse (in Form von konkreten Zuweisungen oder auch Wahrscheinlichkeiten) für neue Daten mit einer möglichst geringen Fehlerrate liefert. Dadurch, dass den vorliegen-

[2] Krallenfrösche quaken nicht.

den Objekten die passenden Zielwerte, wie beispielsweise Klassen oder tatsächliche Werte, typischerweise manuell zugewiesen werden, wird das überwachte Lernen teils auch als „Lernen mit Lehrer“ bezeichnet. Abhängig vom gewünschten Typ des Ergebnisses kann das überwachte Lernen weiter in Klassifikationslernen (engl. classification learning) und Regessionslernen (engl. regression learning) differenziert werden.

Klassifikationslernen

Die Aufgabe, jedem Eingabevektor eine konkrete Kategorie aus einer endlichen Anzahl eigenständiger, disjunkter Kategorien (direkt oder über Angabe eines Wahrscheinlichkeitwerts) zuzuweisen, wird als Klassifikationsproblem bezeichnet. Lernalgorithmen, welche dieses Problem angehen, werden dabei auch als Klassifizierer bezeichnet. Insbesondere das Klassifizieren ist ein charakteristisches Problem aus dem Bereich der Mustererkennung. Ein typischer Anwendungsfall ist etwa das Erkennen von handschriftlichen Ziffern.

Regressionslernen

Umfasst der angestrebte Ausgabevektor hingegen Werte einer oder mehrerer stetiger Variablen, wird dies als Regression oder teils auch als Funktionslernen bezeichnet. Typisches Beispiel für eine Regression ist etwa die Prognose von Aktienwertentwicklungen auf dem Börsenmarkt.

Unüberwachtes Lernen

Das Pendant zum überwachten Lernen ist das unüberwachte Lernen (engl. unsupervised learning), sprich das „Lernen ohne Lehrer“. Im Gegensatz umfasst das Trainingsset hier ausschließlich Eingabevektoren $\mathbf{x}_i$, wobei $i = 1..N$, und damit keine vorab zugewiesenen Targetvektoren. Das primäre Ziel des unüberwachten Lernens ist es, relevante Strukturen in den Daten aufzudecken. Ein typisches Beispiel hierfür ist etwa die Segmentierung von Texten oder Bildern, etwa um Ausschnitte mit Gesichtern in einem Bild zu erkennen.

Clustering-Algorithmen

Einen großen Teilbereich des unüberwachten Lernens bilden Clustering-Algorithmen. Auf Basis einer festgelegten Clusteranzahl oder eines vordefinierten Ähnlichkeitsmaßes werden Objekte auf Gemeinsamkeiten analysiert und in entsprechende Cluster eingruppiert.

Density Estimation

Zudem zählt die sogenannte Dichteeinschätzung (engl. density estimation) zum Bereich des unüberwachten Lernens. Ziel hierbei ist die Bestimmung der Wahrscheinlichkeiten, mit denen ein bestimmter Wert in einem bestimmten Bereichsintervall liegen wird.

Dimensionsreduktion

Eine weitere Methode, die zum unüberwachten Lernen zählt, ist die Dimensionsreduktion. Zielsetzung dieser Art von Algorithmen ist, hochdimensionale Daten auf Räume mit wesentlich weniger Dimensionen zu projizieren und dabei die ursprünglichen Informationen zu bewahren. Die Intention hierbei ist, den erforderlichen Speicherbedarf zu reduzieren sowie die Performance von Analysen auf den Daten zu optimieren.

Verstärkungslernen

Der dritte Bereich wird als Verstärkungslernen (engl. reinforcement learning) bezeichnet und umfasst das Problem, eine passende Aktion zu finden, die in einer gegebenen Situation durchzuführen ist, um damit den erzielten Nutzen im Sinne der Zielerreichung zu maximieren. Im Gegensatz zum überwachten Lernen, bei dem der Algorithmus optimale Ergebnisse für einzelne Beispiele liefert, ist hier die eigentliche Aufgabe, die optimalen Ergebnisse in einem Prozess durch Trial-and-Error (zu dt. Versuch und Irrtum) herauszufinden. Typischerweise existiert hierbei eine Abfolge von Situationen und Aktionen, in denen der Algorithmus mit seiner (dynamischen) Umgebung interagiert. Eine durchgeführte Aktion hat da-

bei nicht nur Auswirkungen auf den direkten Erfolg, sondern auch auf den Erfolg aller nachfolgender Schritte. So existiert für eine gegebene Situation keine optimale Lösung, stattdessen versucht der Lernalgorithmus, eine geeignete Aktion zu finden, so dass über eine bestimmte Zeit das Ziel möglichst effizient erreicht wird. Beispiel hierfür ist etwa ein Schachcomputer. Der Computer lernt mithilfe einer Methode des Verstärkungslernens auf Basis einer Brettposition den besten Zug abzuleiten, mit dem letztendlichen Ziel, das Spiel zu gewinnen. Anhand unzähliger Spiele gegen einen Menschen oder sich selbst werden schlechte Entscheidungen durch Verlieren des Spiels bestraft und gute Entscheidungen entsprechend belohnt. Typisch für Methoden des Verstärkungslernens ist, dass die Belohnung bzw. Bestrafung für durchgeführte Aktionen nicht direkt, sondern verzögert erfolgt, wie etwa erst am Ende des Spiels. Da es keine optimale Aktion für eine gegebene Situation gibt, liegt die generelle Herausforderung des Verstärkungslernens in der Kompromissfindung zwischen „Ausprobieren“ (engl. exploration), wobei versucht wird, neue Aktionen anzuwenden, um deren Effektivität zu beurteilen, und „Ausschöpfen“ (engl. exploitation), im Sinne einer Anwendung bewährter Aktionen aus den bisher gemachten Erfahrungen.

Kontrollaufgabe 1.1: Klassifikation versus Clustering

Worin unterscheiden sich Verfahren für Klassifikation gegenüber Verfahren für Clustering?

Während Themen des unüberwachten Lernens, wie etwa Dimensionsreduktion, bereits in *Modul 10400: Business-Intelligence & -Warehouses* und Clustering-Algorithmen in *Modul 10300: Data-Mining* ausführlich behandelt werden, konzentrieren wir uns in diesem Modul auf Methoden des überwachten Lernens. Ein umfassender Einblick in Methoden des Verstärkungslernens würde den Rahmen dieses Moduls sprengen, weshalb an dieser Stelle auf das Werk von Sutton u. Barto (1998) verwiesen sei.

1.4 Ausgleichsrechnung

approximative Beschreibung der Daten

Wie bereits kurz angerissen, bedient sich das überwachte Lernen zur Prognose von Werten Methoden der Ausgleichsrechnung, wie etwa der Regression, deren grundsätzliches Ziel es ist, Daten über eine Funktion approximativ zu beschreiben.[3] Konkret soll dabei, anhand reeller Werte einer Inputvariablen x, der reelle Wert einer Zielvariablen t prognostiziert werden. Ausgehend von N zu betrachteten Werten von x ergibt sich die Datenmatrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}. \tag{1.7}$$

Aufgrund dessen, dass jeder Eingabevektor $\mathbf{x}_i$ nur durch die Variable x beschrieben wird und damit eindimensional ist, repräsentiert die Datenmatrix einen Spaltenvektor und wird durch $\mathbf{x}$ symbolisiert. Bitte beachten Sie hierbei die (auch durch unterschiedliche Textauszeichnung) gekennzeichnete Unterscheidung[4] zwischen dem N-dimensionalen Spaltenvektor $\mathbf{x}$ (in serifenloser Schrift) und dem Spaltenvektor x (in Serifenschrift), welcher die Dimension D hat. Damit

[3] siehe ergänzend auch *Modul 10200: Mathematical Foundations for Data Science*, *Studienbrief 3: Induktive Statistik*, Kapitel 3.5: Regression

[4] siehe ergänzend hierzu Anmerkungen zur Notation in Kapitel 1.3 in Anlehnung an (Bishop, 2006, S. xi ff.)

umfasst das Trainingsset neben N Werten der Variable x, repräsentiert durch $\mathbf{x} \equiv (x_1, ..., x_N)^{\mathrm{T}}$, die beobachteten zugehörigen Werte von t über $\mathbf{t} \equiv (t_1, ..., t_N)^{\mathrm{T}}$.

synthetisch generiertes Trainingsset

Betrachten wir hierfür ein künstlich bzw. synthetisch erzeugtes Trainingsset, welches auf Basis der Funktion $\sin(2\pi x)$ generiert wurde. Der Eingabevektor $\mathbf{x}$ umfasst $N = 10$ Datenpunkte, die gleichmäßig auf den Wertebereich $[0, 1]$ verteilt sind. Der Targetvektor $\mathbf{t}$ des Trainingssets umfasst die entsprechenden Werte der Funktion $\sin(2\pi x)$, mit einer gewissen zufälligen Abweichung (engl. random noise). Durch diese Art der Trainingsdatengenerierung lässt sich die typische Eigenschaft einer zugrunde liegenden Gesetzmäßigkeit mit einer gewissen Fehlerrate bzw. Unsicherheit in Form von zufälligen Abweichungen in realen Datensätzen imitieren. Nachstehende Abbildung 1.2 zeigt die grafische Repräsentation des generierten Trainingssets, wobei die einzelnen Datenpunkte als blaue Kreise und die zugrunde gelegte Funktion $\sin(2\pi x)$ als grüne Kurve dargestellt sind.

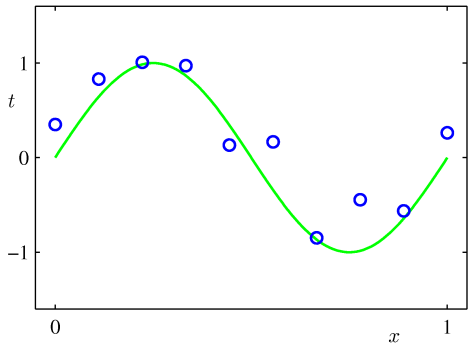

Abb. 1.2: Grafische Darstellung des generierten Trainingssets (Bishop, 2006, S. 4)

Ziel und Herausforderung der Ausgleichsrechnung

Ziel ist es nun, unter Ausnutzung des Trainingssets und ohne Kenntnis über die Funktion (grüne Kurve), diese Gesetzmäßigkeit zu „erlernen“ und damit den Wert $\widehat{t}$ der Zielvariablen für neue Werte $\widehat{x}$ der Inputvariablen vorherzusagen (zu prognostizieren). Die eigentliche Herausforderung liegt dabei in der Generalisierung einer endlichen Menge an Daten, welche überdies eine gewisse Fehlerrate bzw. Unsicherheit aufweisen.

Fitting

Ein einfacher Ansatz hierfür basiert auf der sogenannten Funktionsanpassung oder dem „Fitting“ (engl. curve fitting), wobei versucht wird, die Parameter einer vorgegebenen Funktion zu bestimmen bzw. abzuschätzen. Eine Möglichkeit, Funktionen anzunähern, ist die Verwendung einer Polynomfunktion, wie etwa der Form

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j. \tag{1.8}$$

Das Polynom ist dabei die Summe von Vielfachen von Potenzen mit natürlichzahligen Exponenten einer Variablen (im obigen Beispiel j), wobei über M der Grad des Polynoms bestimmt wird. Die Koeffizienten des Polynoms $w_0, ..., w_M$ werden als Vektor $\mathbf{w}$ zusammengefasst. Obwohl die Polynomfunktion $y(x, \mathbf{w})$ eine nichtlineare Funktion von x ist, ist sie dennoch eine lineare Funktion der Koeffizienten $\mathbf{w}$. Diese Art von Funktionen, welche linear für die unbekannten Parameter sind, besitzen entscheidende Eigenschaften und werden als „lineare Modelle“ bezeichnet, auf die in *Studienbrief 2: Einfache lineare Modelle* näher eingegangen wird.

Minimierung einer Fehlerfunktion

Ziel des Fittings ist es, die Koeffizienten so zu bestimmen, dass die Funktion bzw. das damit beschriebene Modell den Trainingsdaten angenähert wird, wobei dies im Allgemeinen durch Minimierung einer Fehlerfunktion (engl. error function) erfolgt. Die Fehlerfunktion misst die Abweichung zwischen der Funktion $y(x, \mathbf{w})$

und der durch das Trainingsset gegebene Datenpunktwolke. Eine gängige Fehlerfunktion ist die sogenannte „Methode der kleinsten Quadrate“ (engl. sum-of-squares oder least squares), deren Ziel es ist, die Summe der quadratischen Abweichungen der Kurve und damit der Prognosen $y(x_n, \mathbf{w})$ für jeden der beobachteten Datenpunkte x_n und den dazugehörigen Zielwerten t_n zu minimieren. Die Summe der Fehlerquadrate ist damit definiert über

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2. \tag{1.9}$$

Wie wir später sehen werden, erfordert die Bestimmung der Parameter die Ableitung dieser Funktion, weshalb hier aus Gründen der späteren Vereinfachung der Faktor $\frac{1}{2}$ verwendet wird. Damit nun die Parameter der Wahrscheinlichkeitsverteilung im Sinne der Maximum-Likelihood durch Maximieren der Likelihood-Funktion bestimmt werden können, muss das Maximum der Funktion über die erste Ableitung, deren Nullstellen sowie die zweite Ableitung ermittelt werden.

Die einzelnen Abweichungen werden auch als Residuen bezeichnet und ermöglichen Aussagen in Bezug auf die Genauigkeit und Zuverlässigkeit der Funktion bzw. des Modells. Das Ergebnis $E(\mathbf{w})$ ist eine natürliche Zahl, wobei $E(\mathbf{w}) = 0$ genau dann, wenn (gdw.) die Funktion $y(x, \mathbf{w})$ jeden einzelnen Datenpunkt des Trainingssets schneidet. Abbildung 1.3 visualisiert exemplarisch die der Fehlerfunktion zugrunde gelegten Residuen der Funktion $y(x, \mathbf{w})$ zu den einzelnen Zielwerten t_n für jeden der beobachteten Datenpunkte x_n in Form von grünen Balken.

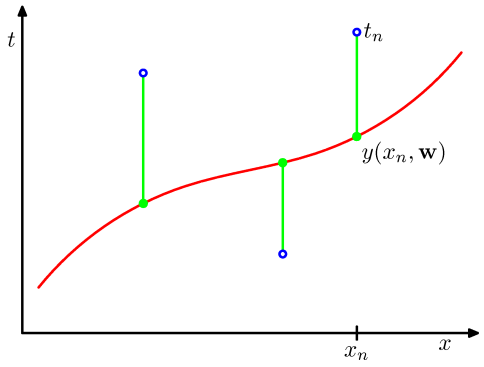

Abb. 1.3: Grafische Darstellung der Residuen (Bishop, 2006, S. 6)

Modellauswahl

Durch Minimierung der Fehlerfunktion lässt sich die für einen gegebenen Grad M optimal approximierte Lösung $\mathbf{w}^*$ bestimmen. Die eigentliche Auswahl eines geeigneten Grades M des Polynoms ist dabei von wesentlicher Bedeutung und wird als Modellvergleich (engl. model comparison) oder auch Modellauswahl (engl. model selection) bezeichnet. Abbildung 1.4 veranschaulicht den Vergleich von vier, im Sinne des Fittings auf das Trainingsset angepassten, Polynomen mit je unterschiedlichem Grad, wobei $M = \{0, 1, 3, 9\}$. Während die konstante ($M = 0$) sowie die lineare Funktion ($M = 1$) eher schlechte Ergebnisse in Bezug auf das gegebene Trainingsset liefern, stellt das Polynom dritten Grades ($M = 3$) eine relativ gute Approximation an die Funktion $\sin(2\pi x)$ dar. Betrachtet man hingegen das Polynom mit dem Grad $M = 9$, so bildet dieses die Datenpunkte des Trainingssets exakt ab, weshalb das Ergebnis der Fehlerfunktion auch mit $E(\mathbf{w}*) = 0$ minimal ist. Allerdings schwankt die resultierende Kurve sehr stark und stellt daher auch eine äußerst schlechte Approximation der Funktion $\sin(2\pi x)$ dar. Dies wird daher auch als Überanpassung (engl. overfitting) bezeichnet. Das Problem der Überanpassung kann durch Vergrößerung des Trainingssets reduziert werden. Eine bewährte Mindestgröße des Trainingssets ist etwa das Fünf- oder Zehnfache der Anzahl anpassbarer Parameter eines Models (Koeffizienten des Polynoms). Allerdings ist anzumerken, dass die Komplexität des Models nicht anhand der Trai-

ningssetgröße, sondern entsprechend der Komplexität des zu lösenden Problems ausgewählt werden sollte.

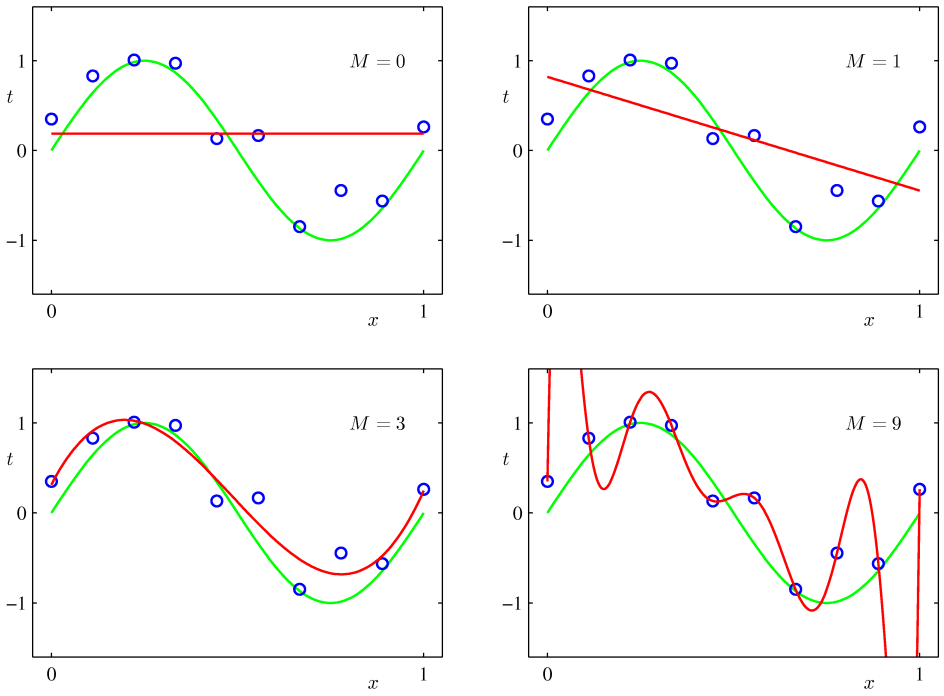

Abb. 1.4: Grafische Darstellung von Polynomen unterschiedlichen Grades (Bishop, 2006, S. 7)

Regularisierung

Typisch für den praktischen Einsatz der Funktionsanpassung ist die Anwendung von relativ komplexen und flexiblen Modellen auf begrenzt große Datensätze. Ein in diesen Fällen häufig verfolgter Ansatz, um das Problem der Überanpassung zu bewältigen, ist die sogenannte Regularisierung (engl. regularization). Dabei wird der Fehlerfunktion (1.9) ein sogenannter Strafterm (engl. penalty term) hinzugefügt, um zu verhindern, dass die Koeffizienten selbst, aber auch deren Anzahl und damit der Komplexitätsgrad des Modells, hohe Werte erreichen. Die einfachste Form eines solchen Strafterms ist die Summe der quadrierten Koeffizienten, wodurch sich folgende abgeänderte Fehlerfunktion ergibt:

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2 + \lambda\|\mathbf{w}\|^2, \tag{1.10}$$

wobei $\|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + ... + w_M^2$ und der Koeffizient λ die Gewichtung des Regularisierungsterms relativ zur Summe der Fehlerquadrate bestimmt. Häufig wird dabei der Koeffizient w_0 im Regularisierungsterm nicht mit berücksichtigt oder ein gesonderter Regularisierungskoeffizient hierfür bestimmt, da das Ergebnis sonst durch die Wahl eines Fixpunktes der Zielvariablen beeinflusst wird. Da hierdurch die Anzahl an Koeffizienten reduziert wird, bezeichnet man Ansätze für die Regularisierung in der Statistik auch als Schrumpfungsmethoden (engl. shrinkage methods).

Regression

Neben Fitting zählt auch die Regression zu den häufig eingesetzten Verfahren der Ausgleichsrechnung. Mithilfe der Regression wird hier ebenfalls eine mögliche Korrelation (funktionaler Zusammenhang) zwischen den Datenpunkten eines Trainingssets, genauer der Einflussgröße x und der Zielgröße y, untersucht. Im Gegensatz zum Fitting liegt hier die Annahme zu Grunde, dass die Datenpunkte

konstant sind und damit keine Fehlerrate im Sinne von Unsicherheiten aufweisen. Anhand einer angenommenen kontinuierlichen Funktion, wie etwa

$$y(x) = ax + b + \epsilon \tag{1.11}$$

bei vermutetem linearem Zusammenhang, wird analysiert, wie gut sich die Daten mit dieser Funktion beschreiben lassen. Neben den resultierenden Regressionsparametern, a und b, bei linearem Zusammenhang, wird teils die Störgröße ϵ als Zufallsvariable verwendet, um nicht erfassbare Messfehler abzubilden. Während die Regression durch Wahl der Funktion (wie etwa die der linearen Regression) eher analytisch orientiert ist, ist das Fitting durch Vergleichen von Modellen unterschiedlichen Grades ein mehr explorativer Ansatz.

Betrachtung aus Sicht der Wahrscheinlichkeitstheorie

Das Problem der polynomiellen Funktionsanpassung lässt sich nicht nur durch Minimierung einer Fehlerfunktion angehen, sondern zudem auch aus Sicht der Wahrscheinlichkeitstheorie betrachten. Die Unsicherheit (Fehlerrate) der Werte der Zielvariablen t lassen sich dabei unter Verwendung einer Wahrscheinlichkeitsverteilung beschreiben. Ausgehend von der Annahme, dass, gegeben den Wert von x, der zugehörige Wert von t normalverteilt (bzw. auch gaußverteilt) ist und das Mittel dem Funktionswert $y(x, \mathbf{w})$ der polynomiellen Kurve aus Formel 1.8 entspricht, lässt sich die bedingte Wahrscheinlichkeit von t über die Wahrscheinlichkeitsverteilung, genauer die multivariate Normalverteilung von t, wie folgt bestimmen:[5]

$$\begin{aligned} p(t|x, \mathbf{w}, \beta) &= \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \\ &= \frac{\exp\left(-\frac{(t-y(x,\mathbf{w}))^2}{2\beta^{-1}}\right)}{\sqrt{2\pi\beta^{-1}}} = \frac{\exp\left(-\frac{\beta}{2}(t - y(x, \mathbf{w}))^2\right)}{\sqrt{\frac{2\pi}{\beta}}}. \end{aligned} \tag{1.12}$$

Während x den eigentlichen Datenwert und $\mathbf{w}$ die Polynomkoeffizienten repräsentieren, ist β der Kehrwert (auch Inverse) der Varianz, welche als Quadrat der Standardabweichung σ definiert ist. Damit ist $\beta = \frac{1}{\sigma^2}$ und wird teils auch als „Genauigkeit“ (engl. precision parameter) bezeichnet. Wir verzichten an dieser Stelle auf eine formale Einführung grundlegender Begrifflichkeiten der Wahrscheinlichkeitstheorie, wie etwa der bedingten Wahrscheinlichkeit oder der Normalverteilung, und verweisen stattdessen auf das *Modul 10200: Mathematical Foundations for Data Science*.

Maximum-Likelihood

Das Trainingsset $\{\mathbf{x}, \mathbf{t}\}$ kann nun dazu verwendet werden, die unbekannten Parameter $\mathbf{w}$ und β mithilfe der Maximum-Likelihood-Methode (zu dt. maximale Wahrscheinlichkeit) zu bestimmen. Im Gegensatz zu vorher, als versucht wurde, anhand der Trainingsdaten die Parameter so zu bestimmen, dass die Fehlerfunktion minimiert wird (wie etwa mit der Methode der kleinsten Quadrate), ist es das Bestreben der Maximum-Likelihood-Methode, die Parameter der Funktion so zu bestimmen, dass die Wahrscheinlichkeit, das Trainingsset zu erhalten, maximiert wird. Dabei wird angenommen, dass die Daten einer Wahrscheinlichkeitsverteilung unterliegen, deren Parameter allerdings nicht bekannt sind. Zudem gilt die Annahme, dass die Daten des Trainingssets, unabhängig von deren Wahrscheinlichkeitsverteilung, aus der Gesamtdatenmenge gezogen werden und gleichmäßig verteilt sind. Diese zugrunde gelegten Annahmen erlauben eine Faktorisierung der einzelnen Wahrscheinlichkeiten, wodurch sich die entsprechende Likelihood-Funktion

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, \mathbf{w}), \beta^{-1}) \tag{1.13}$$

[5] Allgemein wird bei komplexen Exponenten die Exponentialfunktion e^a mit $\exp(a)$ ausgezeichnet.

ergibt. Damit nun die Parameter der Wahrscheinlichkeitsverteilung im Sinne des Maximum-Likelihood durch Maximieren der Likelihood-Funktion bestimmt werden können, muss das Maximum der Funktion über die erste Ableitung, deren Nullstellen sowie die zweite Ableitung ermittelt werden. Da die logarithmierte Funktion dieselben Maximen wie die ursprüngliche Funktion aufweist und die Berechnung von Ableitungen durch das Logarithmieren in den meisten Fällen erleichtert wird, verwendet man in diesem Zusammenhang häufig die sogenannte Log-Likelihood-Funktion. Damit ergibt sich die Log-Likelihood-Funktion

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi). \tag{1.14}$$

Durch Maximierung der Log-Likelihood-Funktion in Bezug auf $\mathbf{w}$ lässt sich die Maximum-Likelihood-Lösung $\mathbf{w}_{\mathrm{ML}}$ für die Polynomkoeffizienten bestimmen. Der damit bestimmte Parametervektor $\mathbf{w}_{\mathrm{ML}}$ lässt sich anschließend verwenden, um die Maximum-Likelihood-Lösung β_{ML} für den Parameter „Genauigkeit“ durch abermaliges Maximieren der Log-Likelihood-Funktion in Bezug auf β zu bestimmen. Nachdem die Parameter $\mathbf{w}$ und β bestimmt sind, können Prognosen für neue Werte von x gemacht werden. Da wir nun ein probabilistisches Modell haben, können die prognostizierten Werte in Form der „prädiktiven Verteilung“ (engl. predictive distribution), welche anstelle von einer einfachen Punktabschätzung die Wahrscheinlichkeitsverteilung über t angibt, ausgedrückt werden. Sie ist damit definiert über

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}). \tag{1.15}$$

Kontrollaufgabe 1.2: Ausgleichsrechnung

Gegeben seien die Datenpunkte $(1, 2)$, $(2, 5)$, $(3, 8)$, $(4, 20)$ und eine polynomielle Regressionsfunktion nach (1.8) mit Grad $M = 2$ und den Gewichten $\mathbf{w} = (w_0, w_1, w_2)^{\mathrm{T}}$.

(a) Wie lautet die Regressionsfunktion für Gewichte $\mathbf{w} = (6, -5, 2)^{\mathrm{T}}$?

(b) Wie groß ist dann die Fehlerfunktion der kleinsten Quadrate (1.9) für die gegebenen Datenpunkte?

(c) Sei nun $\mathbf{w}' = (5, -4, 2)^{\mathrm{T}}$ ein alternativer Gewichtsvektor. Welcher der beiden Gewichtsvektoren ($\mathbf{w}$ oder $\mathbf{w}'$) liefert das bessere Ergebnis?

(d) Wiederholen Sie (b) und (c) für die regularisierte Fehlerfunktion (1.10) mit Regularisierungskoeffizient $\lambda = 0.5$ (d. h. falls Fehlerquadrate und die quadrierte Länge des Gewichtsvektors gleich gewichtet werden).

[...]

1.10 Übungen

Übung 1.1: Nearest-Neighbor-Klassifikation, Kreuzvalidierung, Wald-Typen

Implementieren Sie einen Nearest-Neighbor-Klassifikator (NN) zur Klassifikation verschiedener (japanischer) Wald-Typen auf Satellitenbildern. Die Datensammlung `ForestTypesData.csv` (siehe Vorlesungsverzeichnis) enthält $N = 524$ Datensätze, wobei jeder Datensatz aus $D = 27$ Bild-Merkmalen der Satellitenaufnahmen besteht (deren genaue Bedeutung uns hier nicht näher zu interessieren braucht). Zusätzlich enthält jeder Datensatz in der ersten Spalte ein Klassenlabel, welcher den Datensatz einer von $C = 4$ Klassen zuordnet. Hierbei bedeuten: '*s*'=Sugi-Zypressenwald, '*h*'=Hinoki-Zypressenwald, '*d*'=Mischlaubwald, '*o*'=unbewaldet.

(a) Verwenden Sie zunächst Ihren NN-Klassifikationsalgorithmus von Kontrollaufgabe 1.6 (a) und erweitern Sie ihn um eine Ladefunktion, welche die CSV-Datei einliest und entsprechend in Datenvektoren $\mathbf{x}$ und Klassenlabel $\mathbf{t}$ aufteilt.

Hinweis: Am einfachsten können Sie dies z. B. mit den pandas-Funktionen `read_csv` oder `read_table` machen (siehe *Modul 10100: Programming for Data Science*).

(b) Testen Sie Ihren Klassifikationsalgorithmus, indem Sie die Verwechslungsmatrix $p_{ij} := \mathrm{pr}[Klasse\, j | Klasse\, i]$ auf folgende Weisen bestimmen:

(i) Lerndaten = Testdaten = gesamte Datensammlung

(ii) Lerndaten = Datensätze 1-250; Testdaten = Datensätze 251-524

(iii) Kreuzvalidierung mit $S = 3$.

Vergleichen Sie die resultierenden Fehlerwahrscheinlichkeiten p_{ij} und die Laufzeiten ihres Algorithmus.
Was ändert sich, wenn Sie bei der Kreuzvalidierung S größer machen (z. B. $S = 10$)? Für welches S erhalten Sie die besten Schätzwerte für die tatsächlichen Fehlerwahrscheinlichkeiten?

Übung 1.2: K-Nearest-Neighbor-Klassifikation (KNN) mit Kreuzvalidierung, Wald-Typen

Wiederholen Sie Übung 1.1 (es genügt der Teil mit der Kreuzvalidierung), wobei Sie nun die Klassifikation mit Hilfe des KNN-Algorithmus von Kontrollaufgabe 1.6 (b) erledigen. Testen Sie Ihr Programm für verschiedene Werte, z. B. für $K = 1$, $K = 2$, $K = 5$, $K = 10$ und $K = 20$ und vergleichen Sie dies mit dem Ergebnis aus der vorigen Übungsaufgabe 1.1 ($K = 1$).
Welches K liefert die besten Ergebnisse (d. h. minimiert die Gesamtzahl der Klassifikationsfehler)?
Was passiert für sehr große Werte $K \to N$?

[...]

Liste der Lösungen zu den Kontrollaufgaben

Lösung zu Kontrollaufgabe 1.1 auf Seite 14

Ihre Lösung könnte folgende Argumentation umfassen:

Klassifikation:

- Methode des überwachten Lernens
- Ziel: Zuweisung von Daten zu entsprechenden Klassen
- die einzelnen Klassen sind bereits vorab bekannt

Clustering:

- Methode des unüberwachten Lernens
- Ziel: Gruppierung der Daten abhängig von Ähnlichkeiten
- ausschließlich die Clusteranzahl oder ein Ähnlichkeitsmaß sind vorab zu definieren

Lösung zu Kontrollaufgabe 1.2 auf Seite 19

$(y = 2; 5; 8; 20)$

(a) $y(x) = 6 - 5x + 2x^2$

(b) $y(1) = 3$, $y(2) = 4$, $y(3) = 9$, $y(4) = 18$ und damit $E(\mathbf{w}) = 0,5(1^2 + 1^2 + 1^2 + 2^2) = 3,5$

(c) hier ist $y(x) = 5 - 4x + 2x^2$ und damit $y(1) = 3$, $y(2) = 5$, $y(3) = 11$, $y(4) = 21$ und $E(\mathbf{w}') = 0,5(1^2 + 0^2 + 3^2 + 1^2) = 5,5$, d. h. $\mathbf{w}$ ist besser als $\mathbf{w}'$.

(d) Es ist $\|\mathbf{w}\|^2 = 36 + 25 + 4 = 65$ und $\|\mathbf{w}'\|^2 = 25 + 16 + 4 = 45$, also $\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda\|w\|^2 = 3,5 + 65/2 = 36$ und $\widetilde{E}(\mathbf{w}') = 5,5 + 45/2 = 28$. D. h. hier ist $\mathbf{w}'$ besser als $\mathbf{w}$.

Lösung zu Kontrollaufgabe 1.3 auf Seite 20

Bis auf den Faktor $0,5$ entspricht E genau der Wahrscheinlichkeit eines Klassifikationsfehlers.

Lösung zu Kontrollaufgabe 1.4 auf Seite 26

Um eine Diskriminanzfunktion.

Lösung zu Kontrollaufgabe 1.5 auf Seite 27

(a) Das Modell mit der größten Wahrscheinlichkeit $p(\mathcal{D}|\mathbf{w}_{\mathrm{ML}}^{(M)})$ ist das Polynom vom Grad $M = 9$. Dies ist problematisch, da die Gefahr des Overfittings besteht.

(b) Da $\ln p(\mathcal{D}|\mathbf{w}_{\mathrm{ML}}^{(M)}) - M = -17.3; -8.1; -6.2; -10.2$ für $M = 0; 1; 3; 9$, würde man nach dem AIC das Modell $\mathbf{w}_{\mathrm{ML}}^{(3)}$ mit $M = 3$ wählen.

Lösung zu Kontrollaufgabe 1.6 auf Seite 30

Für (a), (b) siehe folgendes Python-Programm:

Quelltext 1.1: Nearest-Neighbor

```
def getNearestNeighbor(database, x):
    D = len(x);                          # dimension of data
    dmin,imin=-1,-1;                     # init minimum distance
                                         #   and index of minimum
    for n in range(len(database)):       # loop over whole
                                         # database
        xn=database[n];                  # n-th data vector in
                                         #   database
        d=sum([(x[i]-xn[i])*(x[i]-xn[i]) for i in range(D)]);
                                         # distance between x,xn
        if(dmin<0) or (d<dmin) :         # new minimum distance?
            dmin,imin=d,n;               # assign distance and
                                         #   index of new NN
    return imin;                         # return index of final
                                         #   nearest neighbor

def getKNearestNeighbors(database, x, K=1):
    D=len(x);                            # dimension of data
    dmin=[-1 for i in range(K)];         # list of KNN distances
    imin=[-1 for i in range(K)];         # list of KNN indexes
    for n in range(len(database)):       # loop over whole
                                         #   database
        xn=database[n]                   # n-th data vector in
                                         #   database
        d=sum([(x[i]-xn[i])*(x[i]-xn[i]) for i in range(D)]);
                                         # distance between x,xn
        k=K;                             # k will be the neigh-
                                         #   borhood-rank of xn
        while(k>0):                      # k=0 would mean xn is
                                         #   the NN
            if(dmin[k-1]<0) or (d<dmin[k-1]):
                                         # does xn qualify for
                                         #   k-1-NN?
                k=k-1;                   # if yes, put xn 1 up...
                if(k<K-1): dmin[k+1],imin[k+1]=dmin[k],imin[k];
                                         # and old k-th NN 1 down
            else: break;                 # if no, break while-
                                         #   loop
        if(k<K): dmin[k],imin[k]=d,n;    # assign new k-th NN
    return imin;                         # return indexes of the
                                         #   KNN

# test
database = [(4,5,6), (5,7,8), (4,5,8), (1,5,7),
            (8,9,10), (-5,0,-1)];        # list of data vectors
x=database[3];                           # x is test vector
K=3;                                     # number of NNs
idx1=getNearestNeighbor(database,x);     # find nearest neighbor
idxK=getKNearestNeighbors(database, x,K);  # find K NNs

```

```
51 # print results
52 print ”Nearest-neighbors of”, x, ” has index ”, idx1;
53 print K,”-nearest-neighbors of”, x, ” have indexes ”, idxK
```

[...]

Verzeichnisse

I. Abbildungen

II. Kontrollaufgaben

III. Literatur

[Bishop, 2006] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. 1. Springer-Verlag, 2006.

[Sutton u. Barto, 1998] Sutton, Richard S. & Barto, Andrew G.: *Reinforcement learning: An introduction*. MIT press, 1998.

Stichwörter