Paradigmenwechsel vom klassischen zum datengetriebenen Problemlösen im Informatikunterricht

Lukas Höper – Carsten Schulte

Die Bedeutung datengetriebener Technologien aus dem Kontext der künstlichen Intelligenz oder speziell des maschinellen Lernens ist in den letzten Jahren rasant gestiegen. Das Unterrichten von Aspekten künstlicher Intelligenz und maschinellen Lernens ist allerdings nicht mit klassischen Paradigmen sinnvoll möglich. Wir zeigen relevante Unterschiede zwischen klassischem und datengetriebenem Problemlösen auf und argumentieren dann für einen nötigen Paradigmenwechsel im Informatikunterricht.

1 Motivation

In zahlreichen Alltagskontexten finden sich datengetriebene Technologien, die uns assistieren, unterstützen und unser Leben bestimmten. Diese verwenden oft Methoden des maschinellen Lernens (ML) als Teilbereich der künstlichen Intelligenz (KI). Seit einiger Zeit werden daher verstärkt Unterrichtsansätze und -materialien entwickelt, die das Ziel haben, Aspekte der künstlichen Intelligenz oder auch nur des maschinellen Lernens zu erklären. Insgesamt scheint es bei den meisten Ansätzen jedoch das Ziel zu sein, die konkreten Algorithmen von KI- bzw. ML-Verfahren zu vermitteln. Bei veröffentlichten Unterrichtsansätze und -materialien findet damit ein wichtiger Aspekt wenig Beachtung: Die Rolle der Daten als Grundlage der KI- bzw. ML-Verfahren, die bei der Entwicklung und des Einsatzes von datengetriebenen Systemen äußerst relevant sind. Da sich klassisches Problemlösen mit klassischen Algorithmen von jenen mit datengetriebenen Algorithmen (z.B. ML-Verfahren) in einigen Aspekten unterscheidet, argumentieren wir für die Notwendigkeit eines Paradigmenwechsels beim Unterrichten von Aspekten der KI und des ML. Daher diskutieren wir Unterschiede zwischen klassischem und datengetriebenem Problemlösen sowie dessen Bedeutung für den Informatikunterricht. Dabei gehen wir insbesondere auf die Bedeutung von Daten ein, wobei wir den interdisziplinären Zusammenhang von Data Science und KI aufgreifen. Dieser Artikel widmet sich somit insgesamt der Frage, welche Änderungen datengetriebenes Problemlösen für den Informatikunterricht erfordert.

2 Datengetriebenes Problemlösen im Vergleich zum klassischen Problemlösen

Beim klassischen Problemlösen werden zu einem zu lösenden Problem Regeln für Abläufe aufgestellt (Algorithmus), die anschließend in einem Programm implementiert werden. Traditionelle Beispiele aus dem Informatikunterricht sind Sortier- und Suchverfahren. Die Lernenden entwerfen ein regelbasiertes Vorgehen zum Lösen eines gegebenen Problems, entwickeln dafür einen Algorithmus und implementieren ihn anschließend. Im Gegensatz dazu werden beim datengetriebenen Problemlösen mittels ML-Verfahren Modelle entwickelt oder trainiert, die basierend auf Daten eine Approximation einer Funktion darstellen. Im Prinzip werden in Daten statistische Muster identifiziert und damit Regeln generalisiert (der Lernbegriff bei ML ist nicht mit menschlichem Lernen vergleichbar und sollte auch nicht gleichgesetzt werden, da hier lediglich aus Daten Muster i.S. statistischer Zusammenhänge generalisiert werden). Ein ML-Modell wird somit nicht durch das eigene Aufstellen von Regeln für die Verarbeitung von Inputs entwickelt. Dieser Unterschied zwischen dem klassischen und dem datengetriebenen Problemlösen mittels ML-Verfahren diskutieren wir nun schrittweise ausführlicher, wobei wir auf ausgewählte Merkmale der Unterscheidung näher eingehen.

2.1 Erstellung des Programms bzw. Modells

Zunächst gehen wir auf die Unterschiede der Prozesse zur Erzeugung der Problemlösung ein. Beim klassischen Problemlösen soll üblicherweise aus einem Input X ein bestimmter Output Y ermittelt werden. Nach der Problemanalyse wird eine Problemlösung entworfen, d.h. ein Algorithmus für einen regelbasierten, schrittweisen Ablauf aufgestellt. Nach dessen Implementierung kann das Programm kompiliert und ausgeführt werden. Dieses liefert für einen Input X – Korrektheit vorausgesetzt – den gewünschten Output Y. Die Problemlösung wird durch einen Algorithmus oder als Programm dargestellt. Dies wird in Abbildung 1 zusammenfassend visualisiert.

Abb. 1. Schematische Darstellung zum klassischen Problemlösen (Quelle: Eigene Darstellung)

MNU-Journal – Ausgabe 04.2023 – ISSN 0025-5866 – © Verlag Klaus Seeberger, Neuss

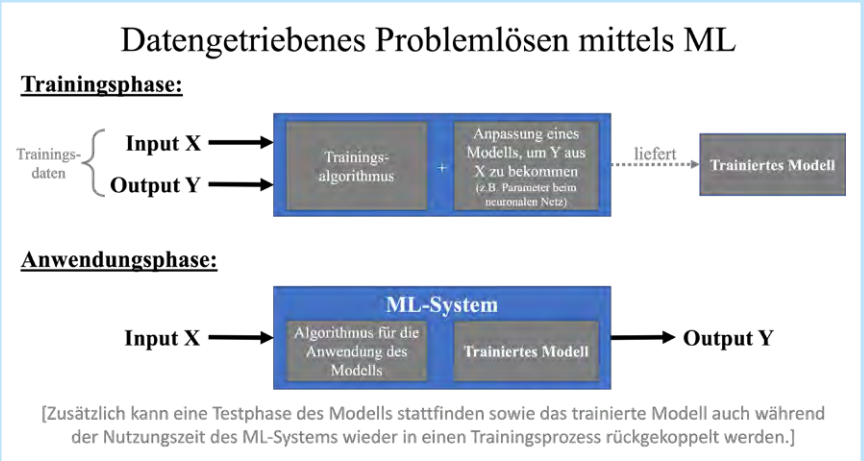

Abb. 2. Schematische Darstellung zum datengetriebenen Problemlösen mittels ML-Verfahren (die Erhebung der Trainingsdaten sowie deren Aufbereitung wurden zur Übersichtlichkeit ausgelassen; Quelle: Eigene Darstellung)

Wenn nun eine Problemlösung mithilfe eines datengetriebenen Ansatzes ermittelt werden soll, läuft dieser Prozess etwas anders ab (unsere Diskussion bezieht sich exemplarisch auf das überwachte Lernen; die meisten Ausführungen sind aber auch auf die anderen Kategorien von ML-Verfahren, wie etwa des unüberwachten und bestärkende Lernens, übertragbar). Für den Problemkontext werden zunächst Daten benötigt, die bereits existieren oder erhoben werden. In diesen Daten sind oft für die Inputdaten bereits der jeweilige gewünschte Output bekannt (X-Y Paare). Die Daten werden anschließend aufbereitet (Bereinigen, Filtern, Labeln, …). Für diese exemplarischen Trainingsdaten wird mithilfe eines Trainingsalgorithmus ein Modell trainiert; das heißt, für ein X-Y Paar passt der Trainingsalgorithmus das Modell so an, dass für das X der mit dem Modell berechnete Output möglichst wenig von dem gewünschten Output Y abweicht. Die Wahl des ursprünglichen Modells bzw. ML-Verfahrens (z.B. Entscheidungsbaum, neuronales Netz, logistische Regression oder Clusteringverfahren) wird kontext- bzw. problemspezifisch getroffen. In der Trainingsphase entsteht ein trainiertes ML-Modell, das an die jeweiligen Trainingsdaten angepasst ist. Dieses trainierte Modell kann nun auf neue Daten angewendet werden, wodurch der jeweilige Output ermittelt wird. Anschließend wird ein trainiertes ML-Modell in einer Testphase evaluiert (dazu später mehr). Somit stellt beim datengetriebenen Problemlösen das trainierte ML-Modell im Wesentlichen die Problemlösung dar. Das datengetriebene Problemlösen wird in Abbildung 2 zusammenfassend visualisiert.

Während beim klassischen Problemlösen die Analyse und der Entwurf einer algorithmischen Lösung zentral sind, stehen beim datengetriebenen Problemlösen die Daten für das Trainieren des ML-Modells im Mittelpunkt. In Diskussionen zu ML werden oft Darstellungen (Abb. 3) verwendet, die aufzeigen, dass der Code (kleine schwarze Box) bei ML-Systemen lediglich einen kleinen Teil darstellt und die Prozesse bezüglich der Daten und des Trainierens des Modells viel umfangreicher sind.

2.2 Daten als Eingabe zum Verarbeiten vs. Daten als Grundlagen zum Trainieren

Wie zuvor beschrieben, haben Daten beim klassischen im Vergleich zum datengetriebenen Problemlösen eine unterschiedliche Rolle. Beim Ersteren dienen Daten als Input, für den der Algorithmus bzw. das Programm einen gewünschten Output ermittelt. Die Entwicklung des Programms ist insofern von diesen Daten losgelöst, dass etwa nur die Beschaffenheit der Daten und eventuell Zusammenhänge bezüglich des Kontexts für die Datenverarbeitung im Programm beachtet wird. Das wäre vergleichbar mit der Anwendungsphase eines ML-Modells, in der dieses auf Inputdaten angewendet wird, um den gewünschten Output zu ermitteln.

Der zentrale Unterschied liegt aber in der Trainingsphase, da die Trainingsdaten die Grundlage des ML-Modells bilden; sodass in dem Modell Zusammenhänge aus den Trainingsdaten genera-

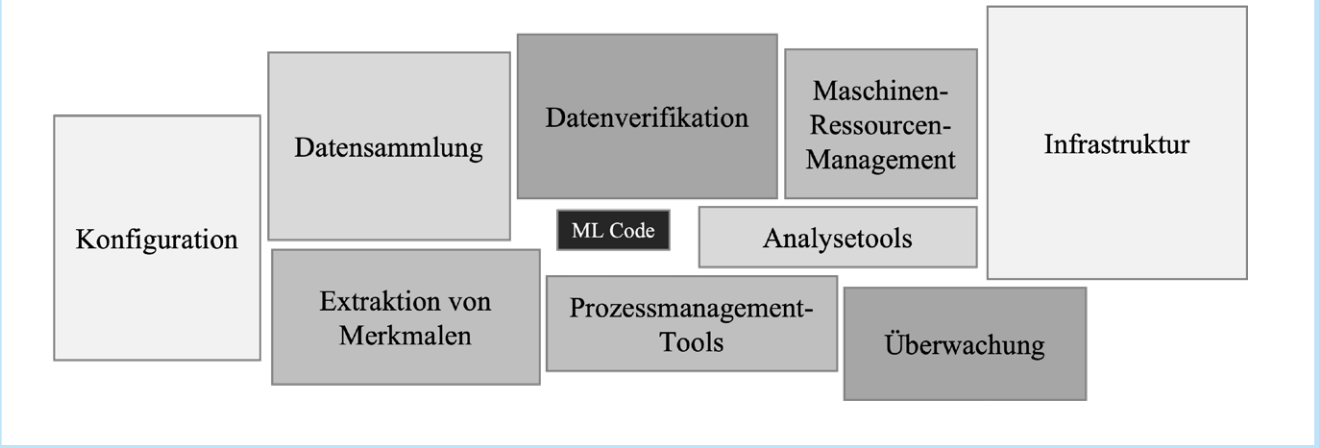

Abb. 3. ML-Code als sehr kleiner Bestandteil eines ML-Systems (Quelle: Anlehnung an (Sculley et al., 2015, 4))

lisiert werden (genauer gesagt trainiert ein Trainingsalgorithmus dieses Modell damit). Somit bildet ein ML-Modell statistische Zusammenhänge aus den Trainingsdaten ab, sodass etwa datenbasierte Entscheidungsregeln für Objektklassifikationen aufgestellt werden können. Wichtig dabei ist, dass ein trainiertes ML-Modell mit anderen Trainingsdaten meistens andere Ergebnisse liefert. Die Wahl der Trainingsdaten beeinflusst maßgeblich das trainierte Modell und damit die Ergebnisse der Problemlösung. Das Modell ist also stark von dem Kontext der Trainingsdaten abhängig und kann somit nicht beliebig für andere Kontexte verwendet werden.

2.3 Korrektheit deduktiv prüfen vs. induktives Evaluieren und Performance testen

Klassische Algorithmen oder Programme können deduktiv hinsichtlich der Korrektheit geprüft werden, also ob die Problemlösung die Spezifikationen des zu lösenden Problems ‚richtig' und sinnvoll abdeckt. Bei der Problemanalyse, dem Modellieren und Entwurf der Lösung werden problemspezifische Zusammenhänge aufgedeckt und genutzt. Die Spezifikationen sollen dabei die richtigen und wichtigen Aspekte des Problems erfassen; und – metaphorisch gesprochen – nicht etwa anstelle einer gewünschten Baumschaukel einen alten Reifen um den Baumstamm wickeln und als angemessene Problemlösung vorstellen. Im Idealfall kann nach geprüfter Korrektheit davon ausgegangen werden, dass die Problemlösung den jeweils richtigen Output liefert, der dabei insbesondere auch vorhersehbar ist.

Beim datengetriebenen Problemlösen werden Modelle basierend auf Daten generiert und können anschließend angewendet werden. Dabei prägen insbesondere die Trainingsdaten das trainierte Modell. Das heißt, wenn für ein Problem mit demselben Trainingsalgorithmus aber unterschiedlichen Trainingsdaten zwei ML-Modelle trainiert werden, können sie bei deren Anwendung unterschiedliche Ergebnisse liefern. Folgendes Beispiel kann das veranschaulichen: Es gibt mehrere Beispiele für ML-Modelle zur Gesichtserkennung, bei denen weiße Personen besser als farbige erkannt werden. Ein Grund dafür liegt in der Wahl der Trainingsdaten, wenn diese etwa nicht ausreichend viele Datenbeispiele für farbige Personen beinhalten. Angepasste Trainingsdaten können das Problem möglicherweise verringern (z.B. Buolamwini & Gebru, 2018).

Die ‚Korrektheit' eines datengetriebenen Systems würde bedeuten, ob das ML-Modell das Problem adäquat löst. Da ein ML-Modell statistische Zusammenhänge und Muster in den Trainingsdaten generalisiert, kann eine ‚Korrektheit' nicht deduktiv geprüft werden; es kann nur nach dem Trainieren in einer Testphase evaluiert werden. Dafür wird das trainierte ML-Modell auf Testdaten angewendet, bei denen bereits die gewünschten Ergebnisse bekannt sind. Der Vergleich der ermittelten Ergebnisse mit den gewünschten liefert eine Fehlerrate als Abweichung, womit eine Performance des Modells beurteilt werden kann. Damit wird entschieden, wie gut das Modell bei den Testdaten ‚funktioniert'. Das Validieren eines ML-Modells basiert somit auf Testen und Experimentieren und wird über statistische Maße der Abweichung interpretiert. Dabei handelt es sich um eine Genauigkeit des Modells (basierend auf den gewählten Testdaten), es wird aber keine ‚Korrektheit' nachgewiesen.

2.4 Nachvollziehbarkeit vs. Mangelnde Erklärbarkeit und Transparenz

Klassische Algorithmen oder Programme liefern anhand regelbasierter Abläufe einen Output, der damit auch nachvollziehbar ist. Eine möglicherweise gesuchte Erklärung für das Ergebnis kann anhand des Ablaufs im Algorithmus bzw. Programm rekonstruiert werden. Limitiert wäre dies lediglich durch fehlende Fähigkeiten beim Verstehen komplexer Programme oder der gewollten Intransparenz eines Anbieters des Systems zum Selbstschutz und zur Bewahrung der eigenen Konkurrenzfähigkeit (Burrell, 2016). Dies widerspricht jedoch nicht der grundsätzlichen Nachvollziehbarkeit der Ergebnisse eines klassischen, algorithmischen Systems.

Bei datengetriebenen Systemen ist die Nachvollziehbarkeit, Erklärbarkeit oder auch Transparenz komplizierter. Zudem erschwert die Komplexität von ML-Modellen oft die Erklärbarkeit. Das Beispiel der künstlichen neuronalen Netze kann dies verdeutlichen: Ein solches besitzt aufgeteilt auf Layer viele künstliche Neuronen, deren Verbindungen gewichtet sind. Ein trainiertes Modell mit den Gewichten ist üblicherweise sehr groß (bedürfen oft mehrere Gigabyte), sodass ein Output insofern nicht mehr nachvollziehbar ist, als dass der Zusammenhang der vielen Gewichte für den Output unverständlich wird (Denning & Denning, 2020). Diese Komplexität von ML-Modellen sorgt oft dafür, dass die Ergebnisse der Anwendung von ML-Modellen oft für Nutzende sowie die Entwickelnden undurchsichtig sind (Burrell, 2016). Zudem gibt es etwa eine fortlaufende Dynamik bezüglich der Anpassungen des ML-Modells (teilweise auch im laufenden Betrieb). All dies erschwert das Verstehen der trainierten Modelle. Daher sind vor einiger Zeit unter anderem die neuen Forschungsgebiete Explainable AI oder auch Algorithmic Accountability entstanden, die nach Verfahren zur Überwindung der mangelnden Erklärbarkeit von ML-Modellen suchen (Rahwan et al., 2019; Rohlfing et al., 2021).

2.5 Korrelation und Kausalität

Bedeutsam im Zusammenhang bezüglich der Erklärbarkeit ist folgender Aspekt: Korrelation ist nicht Kausalität.

Beim klassischen Problemlösen werden Algorithmen systematisch für den Problembereich entwickelt, sind daher problemspezifisch angelegt und enthalten in ihrer Modellierung grundlegendes Wissen über den Anwendungsbereich und dessen innere, semantische Logik – die bei der Problemanalyse erschlossen wird.

Im Gegensatz dazu werden beim datengetriebenen Problemlösen Modelle mit ‚inhaltsneutralen' Trainingsalgorithmen trainiert; sie können ‚lediglich' statistische Zusammenhänge in den Trainingsdaten identifizieren. Daher könnte man die ML-Verfahren im Grunde besser als „Revolution der Computerstatistik [...] statt als Revolution der Intelligenz" (Fry, 2019, 25) bezeichnen. Das Verhalten von ML-Modellen als Problemlösung kann mit dem „klugen Hans" Beispiel verdeutlicht wer-

den: Ein Pferd, das angeblich rechnen konnte, hat tatsächlich aber nur die Körpersprache des Trainers lesen können und hat so das richtige Ergebnis einer Rechnung anzeigte (Lapuschkin et al., 2019). Ein ML-Modell repräsentiert somit Korrelationen in den Trainingsdaten und beschreibt keineswegs kausale oder logische Zusammenhänge (Domingos, 2012). Da für Erklärungen eines Outputs der Anwendung eines ML-Modells semantisch nach dem ‚Warum?' gefragt wird, erschwert auf Korrelationen beruhende ML-Modelle zusätzlich die Erklärbarkeit und Transparenz (Burrell, 2016).

2.6 Umgang mit sozio-kulturellen Herausforderungen

Bei zu lösenden Problemen bestehen oft sozio-kulturelle Herausforderungen, insbesondere wenn sie Menschen betreffen; etwa bezüglich Forderungen nach Fairness, keinen negativen Konsequenzen, Schutz der Privatsphäre und Berücksichtigung des Datenschutzes sowie ethischen und rechtlichen Prinzipien (z.B. Rat der EU, 2022; Datenethikkommission der Bundesregierung, 2019). In diesem Abschnitt diskutieren wir exemplarisch die Unterschiede der beiden Problemlösearten bezüglich Bias und Fairness. Unter Bias wird in der Regel eine systematische Verzerrung von Ergebnissen eines Systems gemeint; im Kontext von ML bedeutet das, dass die Modelle den Kontext nicht ‚adäquat' repräsentieren (Es werden inzwischen mehr als 180 Bias Arten charakterisiert, exemplarisch sei auf folgende Liste mit einer Verlinkung zur Cognitive Bias Codex Visualisierung verwiesen: https://de.wikipedia.org/wiki/Liste_kognitiver_Verzerrungen (11.03.2023)).

Beim klassischen Problemlösen sind diese Aspekte insbesondere in der Problemanalyse sowie des Entwurfs der Problemlösung relevant. Sie entstehen bei mangelnder Passung von Problemlösung zum Problem oder nicht adäquater Modellierung des Problemkontexts. Da die Korrektheit eines klassischen Algorithmus deduktiv überprüft werden kann, können idealerweise nach erfolgreicher Modellierung des Problems und Entwicklung der Problemlösung bias- und fairnessbezogene Fragen geklärt werden.

Daten als Grundlage für das Modell einer datengetriebenen Problemlösung können ebenfalls Bias hervorrufen. Dabei sind verschiedene Arten von Bias relevant, wie etwa ein durch Algorithmen oder ML-Verfahren hervorgerufener Algorithmic Bias oder der Selection/Sampling Bias, der durch die Wahl der Daten entsteht, etwa bezüglich der in den Trainingsdaten erfassten Merkmale (Mehrabi et al., 2022). Ein durch Trainingsdaten bewirkter Bias kann mit dem „garbage in, garbage out"-Prinzip veranschaulicht werden: Wenn die Trainingsdaten nicht gut zur Problemlösung passen oder das zu lösende Problem nicht adäquat repräsentieren, entsteht ein Bias und die Ergebnisse sind nicht gut. Das passiert etwa beim Erfassen bestimmter für das Problem relevante Merkmale bzw. nicht-relevante Merkmale in den Daten. Inwiefern Trainingsdaten die Realität und den Problemkontext genau repräsentieren sollen, kann fraglich sein, da dann ungünstige Effekte in der Gesellschaft in das ML-Modell übernommen werden, etwa existierende Fairnessprobleme. Es gibt viele Beispiele für durch die Trainingsdaten verursachte Bias, wie etwa Googles Spracherkennung, die besser bei Männern als bei Frauen funktionierte (Howard & Borenstein, 2018). Möglicherweise gab es in den Trainingsdaten mehr Daten von Männern als von Frauen. Das Beispiel verdeutlicht, dass die bei datengetriebenen Systemen zugrunde liegenden Daten über die Erstellung der Problemlösung hinaus bias- und fairnessbezogene Probleme entwickeln können.

Um das zu beheben oder abzumildern, könnten in Trainings-Daten Repräsentationen der Realität bewusst verändert werden, um absichtlich ‚Verzerrungen' etwa zum Beheben dieser gesellschaftlichen Probleme hervorzurufen; Matzner (2016) bezeichnet dies als performative Daten.

3 Paradigmenwechsel beim Problemlösen im Informatikunterricht

Zuvor haben wir ausgewählte Aspekte für Unterschiede zwischen klassischem und datengetriebenem Problemlösen diskutiert. Für weitere relevante Unterschiede verweisen wir auf eine Ausführung bezogen auf Computational Thinking von Tedre et al. (2021), die etwa auch auf Testen und Debuggen eingehen. Bezogen auf Veränderungen für Computational Thinking formulieren sie es treffend als: „*Children cannot learn to think about and design ML technology from learning classical programming.*" Aufbauend auf die Diskussion im vorherigen Abschnitt beschreiben wir nun, welche Bedeutung die Unterschiede in den Herangehensweisen im Problemlösen für den Informatikunterricht haben, argumentieren für einen nötigen Paradigmenwechsel und beschreiben anschließend Empfehlungen, wie dieser im Informatikunterricht erfolgen könnte.

3.1 Problemlösen im Sinne des klassischen Problemlösen und Programmieren im Informatikunterricht

Im Informatikunterricht mit klassischem Programmieren sind Algorithmen ein Kernthema und werden im Rahmen der GI-Bildungsstandards (https://informatikstandards.de) wie folgt beschrieben:

> „Algorithmen sind endliche Beschreibungen von Abläufen zur Lösung von Problemen und ergeben bei einer Ausführung eine eindeutig definierte Abfolge von Handlungen. Eine automatische Ausführung auf einem Computer bedarf der Formulierung in einer Programmiersprache."

Es geht also um schrittweise und eindeutige Handlungsvorschriften zum Lösen von Problemen. Am Ende eines solchen Problemlösens steht in der Regel ein implementierter Algorithmus, für dessen Entwurf verschiedene Kompetenzen zum Verstehen und Analysieren des Problems, zum Finden einer Lösung und schließlich zum Formalisieren nötig sind. Eine große Bedeutung trägt dabei der Prozessbereich *Modellieren und Implementieren*. Ein Algorithmus als Problemlösung kann anschließend für verschiedene Daten angewendet werden. Wie zuvor charakterisiert, liefert die Anwendung des Algorithmus bei gleichen Inputdaten stets dasselbe Ergebnis, dessen Korrektheit erklärt und deduktiv nachgewiesen werden kann.

Was oft implizit bleibt, in diesem Zusammenhang aber sehr relevant ist: Die Problemanalyse und der anschließende Entwurf einer Lösung dienen dazu, problemspezifische wichtige (insbesondere kausale) Zusammenhänge aufzudecken und diese für die Lösung zu nutzen. Was ein echter sinnvoller Zusammenhang ist, ergibt sich aus Alltags- bzw. Domänenwissen. Der Korrektheitsbeweis kann im Grunde nur die Passung von Spezifikation zur Lösung nachweisen – implizit mitgedacht wird aber oft, dass die Spezifikation problemspezifisch die richtigen und wichtigen Aspekte des Problems erfassen und somit keine Missverständnisse bzgl. der Passung der Lösung zum Problem entstehen.

3.2 Problemlösen im Sinne eines datengetriebenen Programmierens im Informatikunterricht

Das datengetriebene Problemlösen unterscheidet sich sehr vom klassischen Problemlösen und entfaltet neue Herausforderungen für den Informatikunterricht. Es gibt zwar – verkürzt dargestellt – auch hier ein Problem, Algorithmen, Daten und am Ende eine Problemlösung. Deren Rolle unterscheidet sich aber stark von jenen beim klassischen Problemlösen. Bei datengetriebenen Systemen ist das auf Trainingsdaten basierende Modell zentral, die Algorithmen zum Trainieren und Anwenden des Modells sind eher zweitrangig. Daten sind nicht nur eine Eingabe, für die ein Output ermittelt werden soll, sondern beeinflussen als Trainingsdaten maßgeblich das Modell als Problemlösung.

Zentrale erste Schritte beim datengetriebenen Problemlösen sind: Sammeln oder Wählen von Daten für den Problemkontext; Aufbereiten der Daten für den Trainingsprozess; und Trainieren eines Modells mittels eines gewählten ML-Verfahrens. Ganz anders wird beim klassischen Problemlösen das Problem analysiert und formalisiert, dann ein Algorithmus entworfen und ein Programm als Problemlösung implementiert.

Unterricht zum datengetriebenen Problemlösen sollte nicht Algorithmen in das Zentrum stellen, diese werden im Wesentlichen ‚nur' für das Trainieren des Modells sowie dessen Anwendung genutzt. Zentral sind eher die geschickte Wahl und Aufbereitung der Daten zum Trainieren eines Modells basierend auf einem ML-Verfahren. Nach der Trainingsphase wird das Modell üblicherweise noch evaluiert, woraufhin zur Optimierung die Trainingsphase angepasst wird. Dadurch entsteht ein zyklischer Tüftelprozess beim Trainieren von Modellen durch Experimentieren an den Daten sowie den Parametern des ML-Verfahrens; das ist essenziell für datengetriebenes Problemlösen.

Trotz sorgfältiger Entwicklung hat eine datengetriebene Problemlösung keine perfekte Performance, sodass eine ‚Korrektheit' von Problemlösungen umgedacht werden muss: Ein klassisches Programm kann hinsichtlich des zu lösenden Problems richtig oder falsch sein, bei datengetriebenen Lösungen kann stattdessen nur eine Fehlerrate bei Testdaten experimentell bestimmt werden. Zu klären ist also, was eine ‚richtige' Problemlösung von Schüler/inne/n sein soll. Dann geht es etwa eher um die Schritte im Problemlöseprozess und die Wahl der Trainingsdaten.

Mit anderen wesentlichen Schritten beim datengetriebenen Problemlösen entstehen weitere für den Informatikunterricht relevante Effekte. Während eine klassische Problemlösung transparent und nachvollziehbar ist, mangelt es datengetriebenen Problemlösungen oft an Nachvollziehbarkeit; sie sind eher mit der Metapher der Black-Box zu beschreiben. So können Schüler/innen beim klassischen Problemlösen auch Quelltext erkunden und so deren Funktionsweise erschließen und Funktionen ergänzen. Bei einem ML-Modell kann eher der Trainingsprozess oder die Wahl und Aufbereitung von Trainingsdaten angepasst oder ergänzt werden. Weiterhin sollte beachtet werden, dass Ergebnisse der Anwendung von ML-Modellen auf Korrelationen und nicht Kausalitäten basieren; ein Ergebnis erlaubt nicht direkt Aufschluss auf kausale Zusammenhänge.

Die Trainingsdaten sind zudem als Modell der Welt zu verstehen, sie können die Realität nicht vollständig abbilden. Orientiert an der Idee der performativen Daten stellt sich zudem die Frage, welche Merkmale die Daten abdecken sollen und welche modellhafte Vorstellung des Kontexts implizit mit den Trainingsdaten abgedeckt werden soll. So könnte das ‚Manipulieren' gesammelter Daten für das Aufheben gesellschaftlicher Probleme relevant sein. Zusätzlich gibt es beim Wählen von Trainingsdaten und Trainieren von Modellen soziokulturelle Herausforderungen, wie etwa zu Bias und Fairness.

Zusammenfassend lässt sich festhalten, dass datengetriebenes Problemlösen im Informatikunterricht nicht dem klassischen Problemlösen gleichzusetzen ist. Es gibt einige Änderungen die einen Paradigmenwechsel im Informatikunterricht erfordern. Wir empfehlen daher für datengetriebenes Problemlösen den Begriff des *Datenprojektes*. Ein Datenprojekt berücksichtigt die andere Vorgehensweise und stellt die neue Bedeutung von Daten heraus. Dies kann etwa gelingen, wenn authentische Daten verwendet werden, denn diese erlauben auch auf den Modellcharakter einzugehen sowie Fragen hinsichtlich Bias und Fairness aufzubringen. In einem Datenprojekt werden die skizzierten Schritte beim datengetriebenen Problemlösen herausgestellt, die Vorstellungen hinsichtlich der ‚Korrektheit' und Nachvollziehbarkeit einer Problemlösung vermittelt sowie das Ausprobieren beim Trainieren und Testen eines ML-Modells ermöglicht. Dann kann eine Vermittlung von Fehlvorstellungen vermieden werden, wie etwa dass eine KI universell für jegliche Probleme eingesetzt werden kann oder für das Trainieren beliebige Daten genutzt werden könnten (Kim et al., 2023).

4 Schlussfolgerungen

Wir argumentieren dafür, dass das Unterrichten von Aspekten zu datengetriebenen Technologien einen Paradigmenwechsel benötigt. Anders als der Titel dieses Beitrags suggerieren mag, zielt dieser Beitrag jedoch *nicht* darauf ab, das klassische Problemlösen oder Programmieren durch das datengetriebene Problemlösen zu ersetzen. Entgegen der ursprünglichen Idee des Paradigmenwechsels (i.S. Thomas Kuhn) meinen wir keinen Wechsel einer veralteten oder falschen Vorstellung durch eine neue. Stattdessen soll dieser Beitrag das nötige Umdenken her-

vorheben, wenn es darum geht datengetriebenes Problemlösen mittels ML-Methoden zu unterrichten, wofür das klassische Problemlösen nicht passt. Informatische Bildung sollte also um das Vermitteln adäquater Konzepte und Vorstellungen zum datengetriebenen Problemlösen oder auch Programmieren mittels ML-Methoden erweitert werden, damit informatische Bildung datengetriebene Technologien adäquat abbilden kann. Zukünftiger Informatikunterricht sollte also durchaus – wie auch schon vor einigen Jahren im Kontext von Programmierparadigmen – beide Paradigmen zum Problemlösen unterrichten.

Trotz vieler Vorteile datengetriebenen Problemlösens und deren massenhafter Einsatz in verschiedensten Diensten und Anwendungen haben klassische Algorithmen weiterhin ihre Relevanz. Insbesondere ist der Einsatz von datengetriebenen Systemen in gewissen Kontexten, bei denen etwa Erklärungen von Ergebnissen nötig sind, soziale Bewertungen von Personen vorgenommen werden oder ein Risiko zur Benachteiligung von Personen besteht, stark reguliert oder sogar verboten (Rat der EU, 2022). Folglich ergänzt datengetriebenes Problemlösen das klassische Problemlösen und ersetzt es nicht. Datengetriebenes Problemlösen hilft etwa auch Aufgaben zu bewältigen, bei denen das deduktive Aufstellen von Regeln für Handlungsvorschriften schwierig ist.

In verschiedenen Diskursen und ersten Erfahrungsberichten deutet sich an, dass datengetriebenes Problemlösen einstiegsfreundlich sein kann, da beispielsweise weniger Formalisieren und Entwerfen von Algorithmen gefordert wird und stattdessen durch Ausprobieren früh erste Modelle trainiert und getestet werden können, etwa für eine Bilderkennung (Shapiro et al., 2018; Tedre et al., 2021). Denkbar wäre auch eine parallele Einführung des datengetriebenen und klassischen Problemlösens sowie dessen Kombination. Im Unterricht sollte exemplarisch für datengetriebenes Problemlösen hervorgehoben werden, dass es sich bei KI um keine intelligenten und magischen Akteure handelt, sondern um technische Systeme, die durch klug gewähltes Training von Modellen basierend auf sehr vielen Daten durchaus andere Funktionen bieten können als wir es von klassischen Technologien gewohnt sind.

Förderhinweis

Mehrere in diesem Beitrag genannte Ideen sind durch die Zusammenarbeit der Autoren mit dem restlichen Team des ProDaBi Projekts inspiriert (www.prodabi.de). In dem von der Deutschen Telekom Stiftung initiierten und geförderten Projekt bereiten wir Aspekte von KI, ML und Big Data als Unterrichtsinhalte auf.

Literatur

Buolamwini, J., & Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. *Conference on Fairness, Accountability, and Transparency*, *81*.

Burrell, J. (2016). How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, *3*(1).

Rat der EU. (2022). *Gesetz über künstliche Intelligenz: Rat will sichere und die Grundrechte wahrende KI fördern. https://europa.eu/!MxwnFf* (11.03.2023)

Datenethikkommission der Bundesregierung (Hg.). (2019). *Gutachten der Datenethikkommission. https://www.bundesregierung.de/breg-de/service/publikationen/gutachten-der-datenethikkommission-langfassung-1685238* (11.03.2023)

Denning, P. J., & Denning, D. E. (2020). Dilemmas of artificial intelligence. *Communications of the ACM*, *63*(3), 22–24.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87.

Fry, H. (2019). *Hello world: Was Algorithmen können und wie sie unser Leben verändern* (S. Schmid, Übers.). München: C.H. Beck.

Howard, A., & Borenstein, J. (2018). The Ugly Truth About Ourselves and Our Robot Creations: The Problem of Bias and Social Inequity. *Science and Engineering Ethics*, *24*(5), 1521–1536.

Kim, K., Kwon, K., Ottenbreit-Leftwich, A., Bae, H., & Glazewski, K. (2023). Exploring middle school students' common naive conceptions of Artificial Intelligence concepts, and the evolution of these ideas. *Education and Information Technologies*.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K.-R. (2019). Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, *10*(1).

Matzner, T. (2016). Beyond data as representation: The performativity of Big Data in surveillance. *Surveillance & Society*, *14*(2), 197–210.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2022). A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*, *54*(6), 1–35.

Rahwan, I., Cebrian, M., Obradovich, N., Bongard, J., Bonnefon, J.-F. et al. (2019). Machine behaviour. *Nature*, *568*(7753), 477–486.

Rohlfing, K. J., Cimiano, P., Scharlau, I., Matzner, T., Buhl, H. M. et al. (2021). Explanation as a Social Practice: Toward a Conceptual Framework for the Social Design of AI Systems. *IEEE Transactions on Cognitive and Developmental Systems*, *13*(3), 717–728.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T. et al. (2015). Hidden Technical Debt in Machine Learning Systems. *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, *2*, 2503–2511.

Shapiro, R. B., Fiebrink, R., & Norvig, P. (2018). How machine learning impacts the undergraduate computing curriculum. *Communications of the ACM*, *61*(11), 27–29.

Tedre, M., Denning, P., & Toivonen, T. (2021). CT 2.0. In *21st Koli Calling International Conference on Computing Education Research*. ACM.

Lukas Höper, lukas.hoeper@upb.de, Universität Paderborn, Didaktik der Informatik, Warburger Str. 100, 33098 Paderborn. Interessen: Entwicklung des Konzepts Datenbewusstsein und zugehöriger empirischer Forschung im Informatikunterricht als Teil des Dissertationsvorhabens sowie Data Science und KI als Unterrichtsthemen (Projekt ProDaBi).

Prof. Dr. Carsten Schulte, carsten.schulte@upb.de, Universität Paderborn, Didaktik der Informatik, Warburger Str. 100, 33098 Paderborn. Interessen: Philosophie des Informatikunterrichts, Unterrichtsforschung, Data Science und KI als Unterrichtsthemen (Projekt ProDaBi), erklärbare KI (DFG-Sonderforschungsbereich Constructing Explainability).