Wissensfabrik
Mehr Wissen. Mehr Können. Mehr Zukunft.

IT2School

Gemeinsam IT entdecken

KI-A1 – Die Bananenjagd

Computer selbst lernen lassen

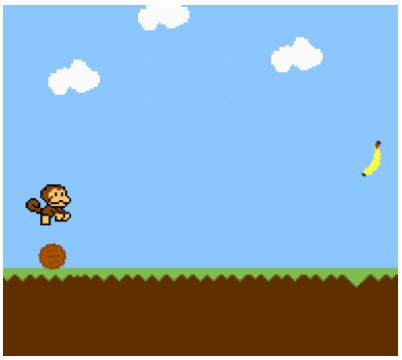
Eine Entwicklung in Kooperation von:

Freie Universität Berlin

Wissensfabrik
Mehr Wissen. Mehr Können. Mehr Zukunft.

Inhalt

1 Die Bananenjagd



In diesem Aufbaumodul schlüpfen die Schülerinnen und Schüler selbst in die Rolle von KI-Entwicklern. Im Gegensatz zu vielen anderen Ansätzen wenden sie dabei nicht nur vortrainierte Modelle an oder setzen existierende Bibliotheken ein, um beispielsweise ihre Daten zu klassifizieren. Stattdessen implementieren die Schülerinnen und Schüler den tatsächlichen Algorithmus, der den Computer "lernen" lässt, in einer blockbasierten Programmiersprache selbst. Damit gestalten die Schülerinnen und Schüler selbst KI-Systeme und durch den Blick "hinter die Kulissen" wird die vermeintliche "Magie" solcher Verfahren demystifiziert.

Lernfeld/Cluster:	Mit KI gestalten	
Zielgruppe/Klassenstufe:		4. bis 5. Klasse
		6. bis 7. Klasse
	X	8. bis 10. Klasse
	X	11. bis 12. Klasse
Geschätzter Zeitaufwand:	4 Stunden (aufbauend auf Modul 1), alternativ 7 Stunden (alleinstehend)	
Lernziele:	Nach Abschluss des Teilmoduls können die Schülerinnen und Schüler… ● KI-Verfahren für überwachtes, unüberwachtes und verstärkendes Lernen in Snap! umsetzen (Wie kann ein Computer lernen?).	
Vorkenntnisse der Schülerinnen und Schüler:	● Modul KI-B3 Schlag den Roboter ● Modul B5 - Programmieren - Leichter Programmiereinstieg	
Vorkenntnisse der/des Lehrenden:	● Modul KI-B3 Schlag den Roboter ● erste Erfahrungen mit Snap!	
Vorkenntnisse der Unternehmensvertreterin/des Unternehmensvertreters:	Empfohlen: ● Beispiele für KI-Anwendungen aus dem eigenen Unternehmen benennen und erläutern können	
Sonstige Voraussetzungen:	keine	

2 Warum gibt es das Modul?

Eine zentrale Rolle in KI-Systemen spielen "KI-Modelle". Diese repräsentieren das "Erlernte" (oder, im Falle klassischer KI, modellierte) Wissen und stellen somit die Basis für weitere Anwendungen dar. Anstatt, wie in zahlreichen anderen Ansätzen, lediglich auf fertig trainierte Modelle zurückzugreifen implementieren die Schülerinnen und Schüler verschiedene Verfahren des maschinellen Lernens selbst. So wird KI transparent, die Funktionsweise sichtbar und die Schülerinnen und Schüler erleben, dass KI eben kein "Hexenwerk" ist. Das Ziel dieses Moduls

ist es, ein tatsächliches und vertieftes Verständnis der zugrunde liegenden Prinzipien und Ideen, die es ermöglichen, dass Computer lernen können, zu vermitteln.

3 Ziele des Moduls

Dieses Teilmodul erlaubt damit einen Blick hinter die Kulissen und fördert so ein vertieftes Verständnis für maschinelles Lernen und die entsprechenden Prinzipien, in dem die Schülerinnen und Schüler tatsächlich in die Black-Box schauen und eigene KI-Artefakte selbst und kreativ gestalten. Damit trägt dieses Teilmodul insbesondere zur technologischen aber auch zur anwendungsorientierten Dagstuhl-Perspektive bei.

- Aus technologischer Perspektive entwickeln die Schülerinnen und Schüler ein vertieftes Verständnis für die Funktionsweise, Prinzipien und Anwendungsmöglichkeiten maschineller Lernverfahren.
- Die Frage, wie KI-Verfahren genutzt werden können, um eigene Ideen umzusetzen und Ziele zu erreichen, wird aus anwendungsorientierter Perspektive u.a. mit selbst erstellten Projekten beantwortet.

4 Die Rolle der Unternehmensvertreterin/ des Unternehmensvertreters

In diesem Modul hat die Unternehmensvertreterin bzw. der Unternehmensvertreter mehrere Möglichkeiten aktiv mitzuwirken. Hier einige Anregungen:

- Als Special Guest in der Schule über die Bedeutung von maschinellem Lernen in der Wirtschaft und insbesondere im eigenen Unternehmen berichten
- Schülerinnen und Schülern eine Exkursion in das eigene Unternehmen ermöglichen und zeigen, wie künstliche Intelligenz in der Praxis eingesetzt wird
- Fragen&Antworten-Runden mit „KI-ExpertInnen“/Data Scientists des eigenen Unternehmens organisieren, die berichten, warum sie sich für ein Studium im Bereich Informatik / Data Science entschieden haben
- „Kreativworkshop“ / „Wettbewerb“ ausrichten, wie das gelernte im Alltag der Schülerinnen und Schüler eingesetzt werden könnte und nach bestimmten Bewertungskriterien (Kreativität, Umsetzbarkeit, Innovationsgrad…) die entwickelten bzw. eingereichten Ideen bewerten
- Unterstützung von Jugend-forscht-Projekten im Bereich KI, die sich aus dem Unterricht ergeben.

5 Inhalte des Moduls

Im Basismodul haben Schülerinnen und Schüler bereits die Grundlagen der verschiedenen Arten des maschinellen Lernens erfahren. Darauf aufbauend implementieren sie in diesem Modul jeweils ein konkretes Verfahren für überwachtes, unüberwachtes und verstärkendes Lernen.

Im Folgenden werden nun die verwendeten Algorithmen genauer erläutert. Die Ideen hinter überwachtem, unüberwachtem und verstärkendem Lernen sind im Modul KI-B3 *Schlag den*

Roboter! nachzulesen. Als Lehrkraft sollten Sie sich außerdem vorab mit den grundlegenden Funktionen von Snap! vertraut machen.

5.1 Verstärkendes Lernen: Q-Table-Learning

Q-Table-Learning ist ein Algorithmus für verstärkendes Lernen, dessen zentrales Prinzip es ist, pro Zustand die jeweiligen Aktionen zu bewerten. Dazu wird eine (Q-)Tabelle erzeugt, mit der ein Agent für jeden Zustand eine Bewertung für jede Aktion verwaltet. Dieser (Q-)Wert in der Tabelle gibt an, welche Qualität (daher auch der Name Q-Learning) eine Aktion in einem Zustand hat. Die Qualität gibt also an, welche Belohnung bei Durchführung dieser Aktion in der aktuellen Situation zu erwarten ist. Durch eine Erhöhung dieses Wertes lernt der Agent, d.h. die Wahrscheinlichkeit dieses Verhalten erneut zu zeigen steigt.[1]

28	A	B	C
1	22	0	1.694824218
2	21	0.999938964	0
3	9	0	0.75
4	8	0	1.988631316
5	7	0.999996185	0
6	-5	0	0
7	-6	1.898815155	0
8	-18	0	1.880790710
9	-19	1.811657321	0
10	-23	1.001953125	1.999999059
11	-22	0	1.999984222
12	16	0	0
13	15	0.9375	0
14	3	0	0

Abb. 1.: Äffchenspiel (links) und Ausschnitt der Q-Tabelle für das Spiel (rechts)

Ein Projekt dieser Sequenz, das Q-Table-Learning nutzt, ist das Bananenjagd-Spiel. Der Agent ist in diesem Fall das Äffchen. Aufgabe des Äffchens ist es, über heranrollende Fässer zu springen. Dazu hat es zwei Möglichkeiten: Springen oder stehen bleiben (vgl. Abb. 1, links). Die zugehörige Q-Tabelle hat drei Spalten (vgl. Abb. 1, rechts). Der Zustand ist in Spalte A angetragen, während Spalte B (springen) und C (nichts tun) die Qualität für die jeweilige Aktion in diesem Zustand beinhalten. Das Äffchen wird sich immer für die Aktion mit der höheren Qualität entscheiden. Einzige Ausnahme: Es möchte neues Verhalten explorieren und daher eine andere Aktion ausprobieren. Basierend auf der für eine gewählte Aktion erhaltenen Belohnung oder Bestrafung wird der Wert in der Tabelle angepasst, und so die Wahrscheinlichkeit dieses Verhalten erneut zu zeigen beeinflusst. Belohnung führt dazu, dass der Wert erhöht und damit das Verhalten häufiger gezeigt wird, Bestrafung dazu, dass der Wert verringert und das Verhalten seltener gezeigt wird.

Exkurs: Weiterer technischer Hintergrund

Die Anpassung der Bewertung ist keine einfache Addition der Belohnung: Einerseits beeinflusst die **Lernrate**, wie stark neue Erfahrungen gewichtet werden. Die Lernrate ist dabei ein Wert

[1] Eine Video-Erklärung zu Q-Table-Learning und der Umsetzung in Snap! am Beispiel des Projekts der Sequenz findet sich hier: https://www.youtube.com/watch?v=llpbHXyNk9Q&list=PL9NH5FGrzyCXEiqCIt0AE_XGV2dI4TTLi

zwischen 0 und 1. Eine Lernrate von 0 führt dazu, dass der Agent überhaupt nichts lernt, weil neu erworbenes Wissen einfach nicht genutzt wird. Eine Lernrate von 1 wiederum bedeutet, dass nur die aktuellste Information berücksichtigt wird. Normalerweise wird die Lernrate daher häufig zwischen diesen beiden Extremen liegen.

Andererseits beeinflusst der **Diskontierungsfaktor** das Lernen: Im Mini-Schachspiel (vgl. auch Modul KI-B3) wird tatsächlich nur die "letzte Aktion" verstärkt/bestraft. Dies liegt jedoch an der sehr einfachen Natur des Spiels, das bei maximal 3 Computerzügen mit Sieg oder Niederlage endet. Im Normalfall ist dies nicht der Fall. Im Fall vom ebenfalls in dieser Sequenz beinhalteten Pong (und generell komplexen Anwendungsfällen), gilt: Würde nur die letzte Aktion berücksichtigt, würde das "richtige" Verhalten oft bestraft werden – zum Beispiel, wenn der Schläger sich in die richtige Richtung bewegt und trotzdem den Ball nicht erreicht. Wichtig sind auch die vorangegangenen Aktionen, die zu diesem Zustand geführt haben, und ebenfalls verstärkt oder bestraft werden müssen. Es sollten üblicherweise also nicht nur die aktuellen, sondern auch die vorherigen Zustände und gewählten Aktionen in den Lernprozess einbezogen werden. Damit beeinflusst nicht nur die kurzfristige Belohnung für eine Aktion in einem Zustand die Entscheidung, es wird versucht die langfristige Belohnung zu maximieren. Beim Q-Table-Learning-Algorithmus, den wir bei der Bananenjagd bzw. dem Pong verwenden, wird statt der Vergangenheit die „zukünftige Belohnung“ berücksichtigt. D.h. anstatt in die Vergangenheit zu schauen, wird die (erwartete) Zukunft unserer Aktionen mit einbezogen und mit dem Diskontierungsfaktor gewichtet. Auch damit werden alle Aktionen belohnt, die zu einer Belohnung geführt haben (allerdings nicht beim ersten Aufsuchen des Zustands).

Der Diskontierungsfaktor y (typischerweise $0 < y < 1$) sagt also aus, dass eine Belohnung oder Bestrafung, die der Agent n-Schritte in der Zukunft erhält, um den Faktor y^n diskontiert wird, also weniger Wert ist. Ein Faktor von 0 bewirkt, dass der Agent nur aktuelle kurzfristige erreichbare Belohnungen berücksichtigt. Durch Einstellen dieses Faktors ist es also möglich zu steuern, wie stark die zukünftige zu erwartende Belohnung und damit die langfristige Auswirkung einer Aktion berücksichtigt wird.

Mathematisch lässt sich das wie folgt ausdrücken, wobei q_{neu} die neue Bewertung und q_{alt} die alte Bewertung einer Aktion im aktuellen Zustand ist:

$$q_{neu} = q_{alt} + Lernrate \cdot (Belohnung + y * Schätzung\ zukünftige\ Belohnung - q_{alt})$$

Während des Lernens ist der Agent stets in einem Konflikt. So muss der Agent sich stets entscheiden, ob er sich weiter daran macht, seine Belohnung mit seiner bekannten besten Strategie zu erhöhen oder ob er nun versucht, bisher unbekannte bessere Aktionen und Zustände zu finden (Exploration), die bisher gar nicht ausprobiert wurden. Über die sog. **Explorationsrate** lässt sich daher steuern, wie bereitwillig der Agent andere Aktionen als die zurzeit am besten bewertete ausprobiert. In unserer Umsetzung bestimmt die Explorationsrate daher die Wahrscheinlichkeit, mit der statt die “beste” Aktion zu wählen, eine zufällige ausgewählt wird. Eine hohe Explorationsrate bedeutet also, dass der Agent eher bereit ist, neue Aktion auszuprobieren.

Lässt sich dieses Prinzip jetzt einfach auf FIFA, Super Mario usw. übertragen?

Ja! Während es aber bei Spielen wie Pong oder dem Äffchenspiel noch möglich ist, eine Tabelle aller Zustände zu führen, ist dies bei komplexeren Spielen (genauso wie bei der realen Welt) kaum mehr möglich. Bei Tic Tac Toe wären bereits 19,683 verschiedene Zustände nötig und bei

Spielen wie Fifa oder Dota2 würde die Tabelle mehrere Millionen Zeilen enthalten und wäre damit so groß, dass das Training viel zu lange dauern würde. Solche Situationen erfordern ein Verfahren, das in der Lage ist, das in verschiedenen Zuständen erworbene Wissen zu verallgemeinern und auch auf bisher ungesehene Situationen anwenden zu können. An dieser Stelle kommen beispielsweise neuronale Netze als Datenstruktur ins Spiel (Deep Q-Learning). Für jeden Zustand approximiert dann ein neuronales Netz den Q-Wert für jede mögliche Aktion.

Videos mit weiteren Beispielen zu Machine Learning in Videospielen:

- Google DeepMind's Deep Q-learning playing Atari Breakout (Deep-Q-Learning): https://www.youtube.com/watch?v=V1eYniJ0Rnk
- AI playing Super Mario World with Deep Reinforcement Learning (Deep Reinforcement Learning): https://www.youtube.com/watch?v=L4KBBAwF_bE
- StarCraft II Deep Reinforcement Learning Agent (Deep Reinforcement Learning): https://www.youtube.com/watch?v=gEyBzcPU5-w
- Deep-Q-Learning mit Mario Kart: https://www.youtube.com/watch?v=Tnu4O_xEmVk

Warum Q-Table-Learning?

Der Vorteil von Q-Table-Learning ist vor allem, dass das Training bei wenigen möglichen Zuständen sehr schnell gehen kann: Je nach gewählter Zustandsbeschreibung sind Effekte in unter drei Minuten sichtbar. Zudem kann der Q-Learning Algorithmus durch Nutzung einer Tabelle, in der Zustände und die Q-Werte für die jeweiligen Aktionen gespeichert sind, nicht nur leicht verstanden und visualisiert, sondern auch von Schülerinnen und Schülern selbst implementiert werden. Natürlich ist Q-Table-Learning nicht auf Snap! beschränkt. Das Prinzip eignet sich auch, um auf Processing, Java oder Stride übertragen zu werden.

Wie wähle ich passende Parameter?

Tatsächlich gibt es kein richtig oder falsch für die Wahl von Lernrate, Diskontierungsfaktor, Zustand oder Belohnung. Die Schülerinnen und Schüler sind hier ein Stück weit gefordert, darüber nachzudenken, welche Informationen in ihrer Situation wichtig sind und welche Aspekte die Höhe der Belohnung bzw. Bestrafung bestimmen. Und trotz aller Überlegungen ist das Umsetzen eines Lernalgorithmus auch immer mit Experimentieren verbunden.

5.2 Überwachtes Lernen: Lineare Regression

Die lineare Regression ist ein überwachtes Lernverfahren zur Beschriftung eines unbekannten Datenpunkts bei vorhandenen beschrifteten Daten mit einer Zahl. Dabei wird versucht, den Wert eines Zielmerkmals durch ein Eingabemerkmal vorherzusagen. Lineare Regression wird etwa bei Zusammenhängen (Welchen Einfluss hat die Fläche eines Hauses auf den Verkaufspreis?) oder bei Prognosen (Wie hoch wird der Absatz in einigen Wochen sein?) eingesetzt. Oft lässt sich dieser Zusammenhang näherungsweise durch eine Gerade beschreiben[2]. Für einen

[2] Ist der Zusammenhang zwischen Eingabe- und Zielmerkmal linear ist dies möglich, ansonsten benötigen wir andere Regressionsverfahren, die aber nicht Teil dieser Unterrichtseinheit sind.

Datensatz mit gegebenem Eingabemerkmal lässt sich das Zielmerkmal, dann mithilfe der Gerade vorhersagen (vgl. Abb. 2).

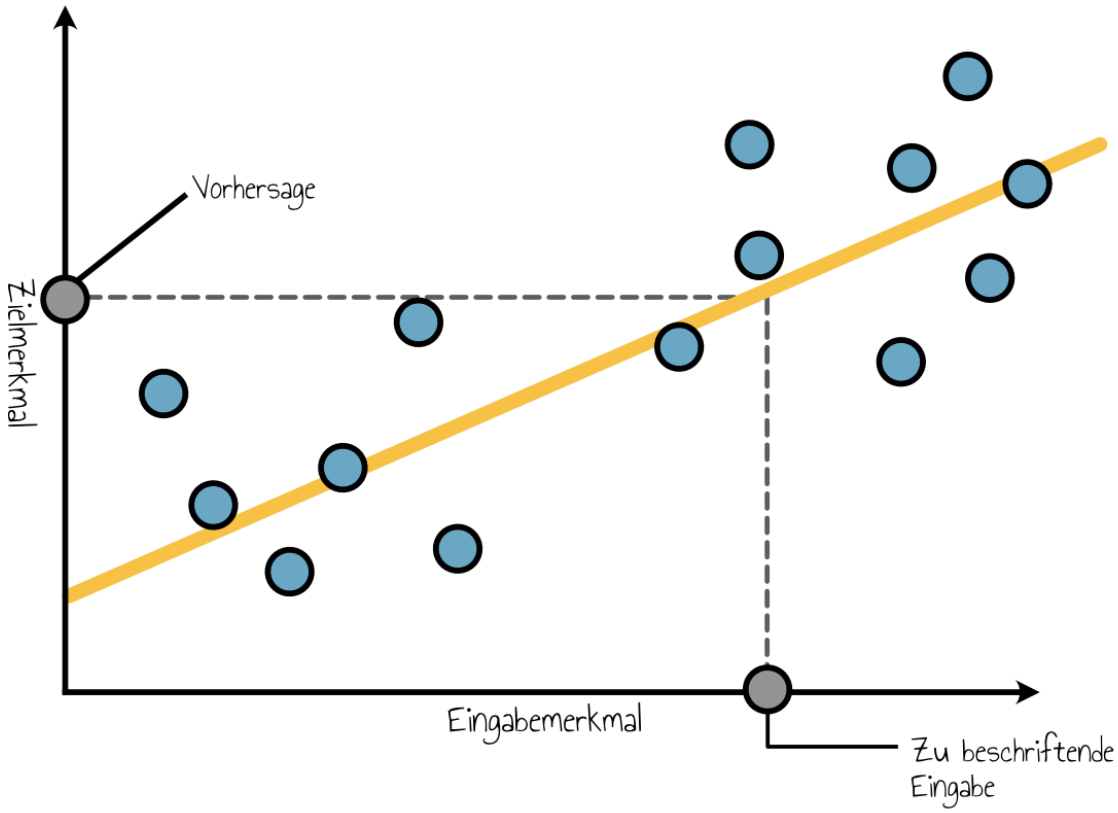


Abb. 2: Die Idee der linearen Regression ist es, den Zusammenhang zwischen zwei Merkmalen durch eine Gerade zu beschreiben und diese für Vorhersagen zu nutzen.

Eine Gerade ist gegeben durch die Gleichung y = a x + b. Ein Erlernen dieser Gerade durch den Computer, bedeutet daher lediglich, geeignete Parameter a und b zu finden. Diese beiden Parameter zusammen mit der Information, dass es sich um eine Geradengleichung handelt, stellen also das Modell dar, das gelernt werden soll.

Dazu verbessert der Algorithmus beginnend mit a = 0 und b = 0 die Parameter schrittweise, indem er für die bereitgestellten Trainingsdaten den aktuell vom Modell vorhergesagten Wert des Zielmerkmals mit dessen eigentlichen Wert vergleicht und a und b so anpasst, dass die Abweichung geringer wird (vgl. Abb. 3).

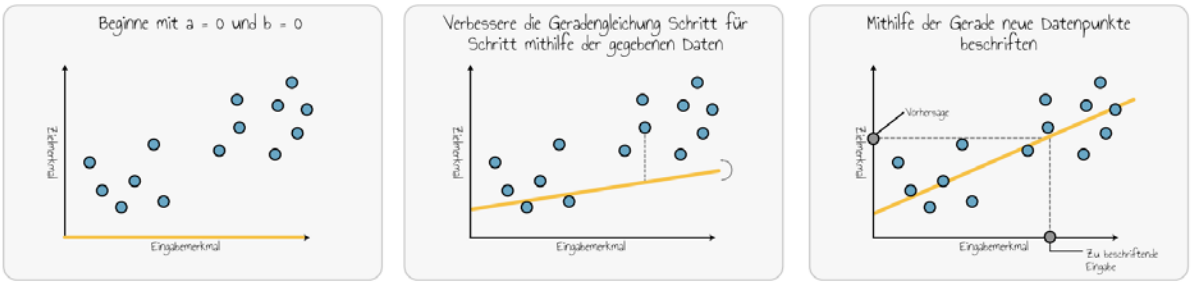


Abb. 3: Ablauf von linearer Regression. Die Gerade ist in Gelb angetragen.

Um das Zielmerkmal noch genauer vorhersagen zu können, ist es in der Praxis zumeist hilfreich, mehr als ein Eingabemerkmal zu verwenden. Die sogenannte multiple lineare Regression erlaubt dazu auch mehrere Eingabemerkmale zu verwenden.

5.3 Unüberwachtes Lernen: Vektorquantisierung

In der Gold-Rush-Aktivität aus *KI-B3 Schlag den Roboter!* entwickeln die Schülerinnen und Schüler selbst ein unüberwachtes Lernverfahren, das sich *Vektorquantisierung (VQ)* nennt. Dabei werden Cluster im Datensatz gefunden, indem der Algorithmus Punkte findet, die ein Cluster prototypisch beschreiben (vgl. Abb. 4).[3]

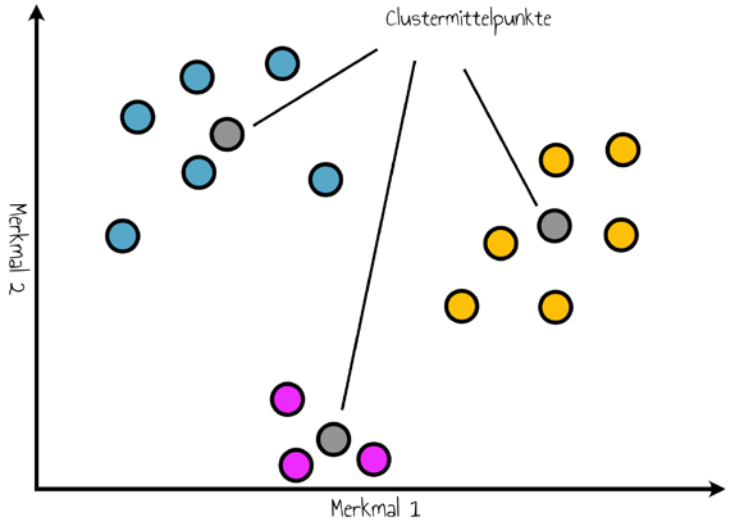


Abb. 4: Die Idee der Vektorquantisierung ist es, Punkte zu finden, die ein Cluster prototypisch beschreiben (Clustermittelpunkte)

Dazu werden sogenannte Prototypen herangezogen, die an jeden Datenpunkt angepasst werden, um diese Clustermittelpunkte zu identifizieren. Zunächst wird die Anzahl der zu findenden oder zu erwartenden Cluster festgelegt und entsprechend viele Prototypen zufällig positioniert[4]. Anschließend wird zufällig einer der Trainingsdatenpunkte ausgewählt, der nächste Prototyp identifiziert und um einen Teil der Strecke (z.B. die Hälfte) in Richtung des ausgewählten Datenpunkts bewegt. Dies wird mindestens einmal für alle Datenpunkte wiederholt (vgl. Abb. 5).

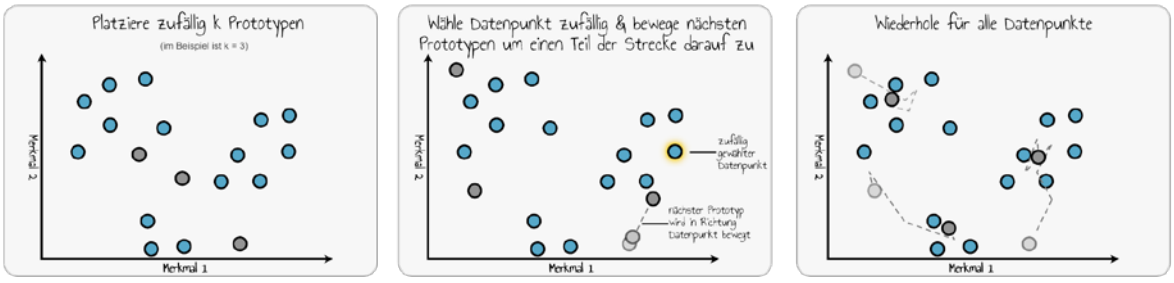


Abb 5: Ablauf von VQ. Die Prototypen sind grau hinterlegt, die verschiedenen Cluster sind zum eigenen Verständnis farblich eingefärbt. Der Algorithmus verfügt über diese Information jedoch nicht, es ist sein Ziel, diese Cluster zu finden!

---

[3] Eine Video-Erklärung zu VQ und der Umsetzung in Snap! am Beispiel des Projekts der Sequenz findet sich hier: https://www.youtube.com/watch?v=Ym-fy_O5H38&list=PL9NH5FGrzyCXEiqCIt0AE_XGV2dI4TTLi&index=10

[4] Hierfür existieren Faustregeln und Heuristiken, die an dieser Stelle jedoch zu weit führen würden. Für diese Sequenz gilt daher, dass die Anzahl an Prototypen für die konkrete Aufgabe sinnvoll gewählt werden sollte (wenn wir bspw. Bushaltestellen positionieren wollen, dann entspricht die Anzahl an Prototypen der Anzahl an möglichen Bushaltestellen).

Über diese einfache Version hinaus sind verschiedene Verbesserungen des Verfahrens denkbar, so kann der Datensatz mehrfach durchlaufen, die Schrittweite reduziert oder die Sensitivität eines Prototyps mit jeder Runde, in der er nicht bewegt wird, erhöht werden.

Warum VQ?
Der Vorteil des VQ-Algorithmus für den Einsatz im Unterricht ist die einfache Umsetzung, gerade für zweidimensionale Räume. Gleichzeitig ist der Algorithmus durchaus anschlussfähig, nachdem er konzeptionell sehr ähnlich zu bekannten unüberwachten Lernverfahren wie k-means ist.

6 Unterrichtliche Umsetzung

In der unterrichtlichen Umsetzung setzen die Schülerinnen und Schüler die verschiedenen Arten, wie Maschinen lernen, konkret in Snap*!* um. Die Stunden können dabei jeweils isoliert oder in der gesamten Sequenz (in beliebiger Reihenfolge) umgesetzt werden. Die Einheiten können somit auch bspw. in Modul KI-B3 integriert werden.

6.1 Grober Unterrichtsplan

Je nachdem, ob das Modul KI-B3 Schlag den Roboter! bereits durchgeführt wurde, stehen die folgenden 2 Varianten für die unterrichtliche Umsetzung zur Auswahl.

Variante I: KI-B3 bereits durchgeführt

Unterrichtsszenarien	Kurze Zusammenfassung
Verstärkendes Lernen (2 UZE)	Die SuS ● setzen den Q-Table-Learning-Algorithmus in Snap! um
Überwachtes Lernen (1 UZE)	Die SuS ● setzen lineare Regression in Snap! um
Unüberwachtes Lernen (1 UZE)	Die SuS ● setzen den VQ-Algorithmus in Snap! um

Variante II: KI-B3 noch nicht durchgeführt

Sollte Modul KI-B3 nicht bereits durchgeführt worden sein, erhöht sich der Zeitbedarf für jedes der Lernverfahren um eine Unterrichtsstunde, in der zunächst die jeweilige Unplugged-Aktivität gemäß der Beschreibung in Modul KI-B3 durchgeführt wird.

Unterrichtsszenarien	Kurze Zusammenfassung
Verstärkendes Lernen (3 UZE)	Die SuS ● entdecken die zugrundeliegende Idee verstärkenden Lernens mithilfe des Spiels „Schlag den Roboter“ ● setzen den Q-Table-Learning-Algorithmus in Snap! um
Überwachtes Lernen (2 UZE)	Die SuS ● entdecken die zugrundeliegende Idee überwachten Lernens mithilfe des „Gute-Äffchen-Böse-Äffchen-Spiels“ ● setzen lineare Regression in Snap! um
Unüberwachtes Lernen (2 UZE)	Die SuS ● entdecken die zugrundeliegende Idee verstärkenden Lernens mithilfe des Spiels „Goldrausch“ ● setzen den VQ-Algorithmus in Snap! um

6.2 Stundenverlaufsskizzen

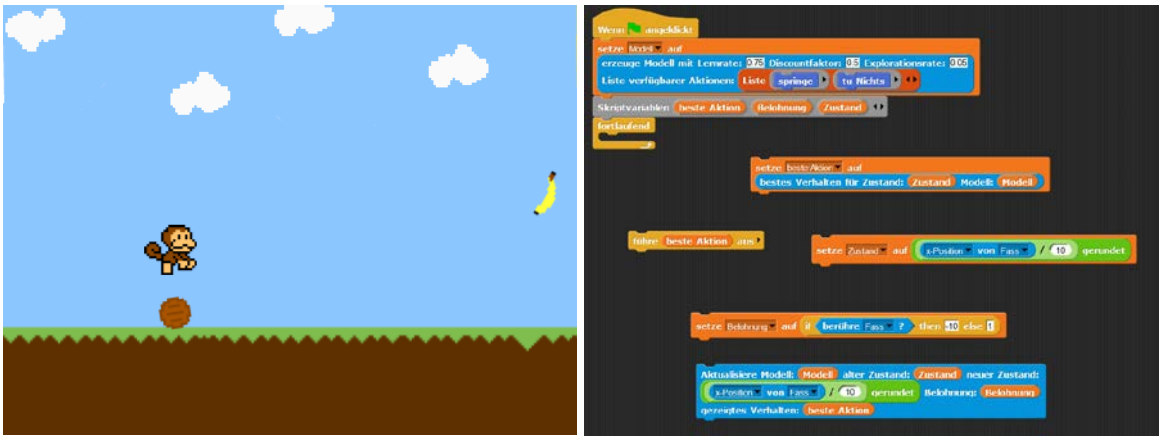
6.2.1 Stunde 1 und 2: Verstärkendes Lernen

In der ersten Einheit implementieren die Schülerinnen und Schüler selbst einen lernenden Agenten für zwei Spiele in Snap!.

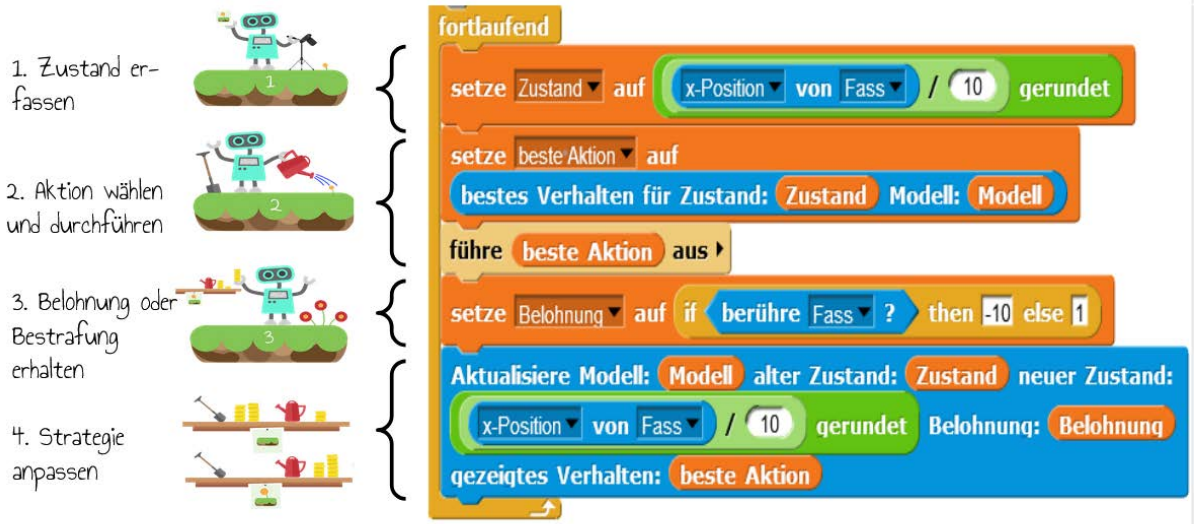
Zeit	Phase	Sozialform/ Lehrerimpuls	Inhalt/Unterrichtsgeschehen	Material
30	(optional)		Durchführung der Unplugged-Aktivität zu verstärkendem Lernen, siehe Modul KI-B3.	KI-B3.2.1, KI-B3.2.3 Spielsteine
15	(optional)	Plenum	Sicherung der zugrunde liegenden Idee, siehe Modul KI-B3.	
30	Erarbeitung I	Partnerarbeit	Zunächst zeigen Sie als Lehrkraft kurz das Bananenjagd-Projekt und wie der Affe lernt, über die Fässer zu springen. Anhand des Arbeitsblattes setzen die Schülerinnen und Schüler das Projekt nun selbst um.	KI-A1.1.1
10	Sicherung	Plenum	Lassen sie die Schülerinnen und Schüler das Bananenjagd-Projekt mit den Mini-Schachspiel vergleichen.	
40	Erarbeitung II	Plenum	Die Schülerinnen und Schüler übertragen das Prinzip auf das bereitgestellte Pong-Projekt. Auf dem Arbeitsblatt finden sich weitere Experimentier- und Beobachtungsaufträge. Legen sie die Hilfekarten im Klassenzimmer aus.	KI-A1.1.1, KI-A1.1.2 Hilfekarten
5	Sicherung und Showcase	Plenum	Einzelne Gruppen der Schülerinnen und Schüler demonstrieren ihre Ansätze der Belohnung und Bestrafung.	
5	Ausblick	Plenum	Zum Abschluss wird anhand der exemplarisch bereitgestellten Videos der Einsatz von Verstärkendem Lernen/Q-Learning in der Praxis betrachtet.	Linkliste siehe S. 7

Erarbeitung I: Bananenjagd

Aufbauend auf der Unplugged-Aktivität aus Modul KI-B3 wird das dort erarbeitete Prinzip verstärkenden Lernens auf einen “Affen”-Agenten übertragen, um ihn so bei der Bananenjagd zu unterstützen! Ziel des Äffchens ist es, über das Fass zu springen. Die Schülerinnen und Schüler arbeiten mithilfe des Arbeitsblattes KI-A1.1.1 und erhalten eine Vorlage für das Spiel “Bananenjagd”. Dort sind alle Blöcke, die für den selbstlernenden Agenten von Relevanz sind, enthalten – allerdings nicht in der richtigen Reihenfolge. Die Aufgabe der Schülerinnen und Schüler ist es, diese in die richtige Reihenfolge zu bringen.



Dabei müssen die Schritte des verstärkenden Lernens (die mit Hilfe des Mini-Schachs bereits erarbeitet wurden) den Code-Schnipseln zugeordnet werden: Zunächst wird der aktuelle Zustand bestimmt, dann die beste Aktion bestimmt und ausgeführt. Anschließend wird die Belohnung bzw. Bestrafung berechnet sowie das Modell aktualisiert und damit die Strategie angepasst:



Optional kann die Snap-Vorlage um ein Skript erweitert werden, das auf Tastendruck ein Fass erzeugt. Damit wird das Spiel für die Schülerinnen und Schülern interaktiver.[5]

[5] Beispielhaft ist dies hier umgesetzt: https://snap.berkeley.edu/snap/snap.html#present:Username=seegerer&ProjectName=it2school-RL-Bananenjagd-Puzzle-Fass-erscheint-bei-Leertaste

Anschließend erhalten die Schülerinnen und Schüler auf dem Arbeitsblatt Beobachtungsaufträge: Sie sollen zunächst festhalten, wie sie das Lernen beschreiben würden. Dabei stellen Sie fest, dass das Verhalten zunächst zufällig erscheint, der Agent aber zunehmend lernt, länger auf dem Boden zu bleiben. Allerdings rennt das Äffchen trotzdem hin und wieder ins Fass hinein und auch der Doppelsprung ist noch ein Problem.

Weiterhin erhalten die Schülerinnen und Schüler den Auftrag die Tabelle anzusehen, die das Modell darstellt. Diese kann durch Ausführen des Blocks 

eingesehen werden. Mit Hilfe des Schritt-für-Schritt Debuggers[6] können die Schülerinnen und Schüler beobachten, wie sich die Werte nach jedem Schritt ändern und auf Basis der Beobachtungen weitere Fragen beantworten.

Durch die Beobachtung der Werte, können die Schülerinnen und Schüler das Lernen des Algorithmus „live“ miterleben und erfahren dabei die Grundprinzipien von verstärkendem Lernen. Auch der Computer lernt durch Belohnung und Bestrafung. Im Unterschied zu menschlichem Lernen bedeutet dies aber lediglich die Anpassung der entsprechenden numerischen Bewertung einer bestimmten Aktion.

Sicherung

Anschließend sollten im Unterrichtsgespräch die Gemeinsamkeiten des Bananenjagd-Projektes zum Minischach hergestellt werden. Im Minischach ist das Modell die Tabelle der Spielsituationen, bei der die Strategie über Schokolinsen und deren Häufigkeit gespeichert ist. Genauso wird im Bananenjagd-Projekt in der Tabelle für jeden Zustand 

(Spalte A) für jede der möglichen Aktionen

(Spalte B & Spalte C) die “Qualität” der jeweiligen Aktionen in diesem Zustand mit Hilfe eines numerischen Wertes gespeichert und bei Belohnung bzw. Bestrafung angepasst.

Erarbeitung II: Pong

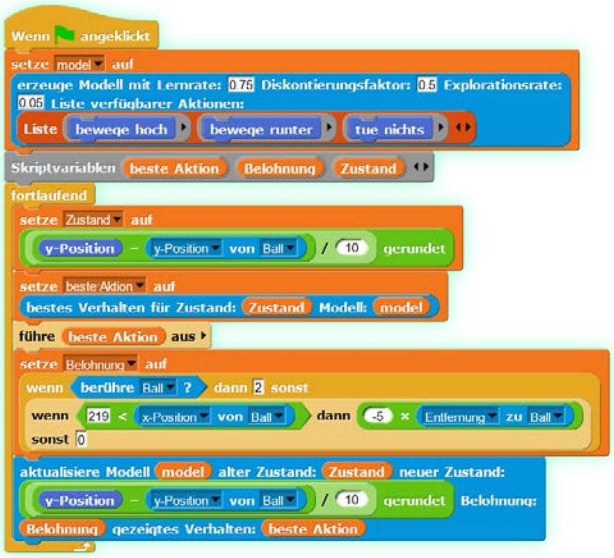
Im nächsten Schritt soll nun ein Agent für das Spiel Pong umgesetzt werden. Dazu identifizieren die Schülerinnen und Schüler die Unterschiede von Pong zur Bananenjagd mithilfe des Arbeitsblatts bezüglich folgender Aspekte:

- Mögliche Aktionen
- Belohnungsfunktion
- Erfassung des Zustands

Dies sind die einzigen drei Bestandteile, die abhängig vom Kontext sind und damit für das Pong- oder beliebige weitere Spiele angepasst werden müssen.

Anschließend implementieren die Schülerinnen und Schüler, analog zur Bananenjagd, den rechten Schläger im Pongprojekt. Dabei experimentieren sie mit verschiedenen Belohnungsfunktionen und Zuständen und beschreiben deren Effekte auf das Lernverhalten. Zur Unterstützung stehen Hilfekarten (KI-A1.1.2) bereit, die im Klassenzimmer ausgelegt werden können.

[6] Der Schritt-für-Schritt-Debugger lässt sich in Snap*!* über einen Klick auf die Fußspuren in der oberen Leiste aktivieren.



Eine mögliche Differenzierung ist es, auch den Einfluss der Parameter Lernrate, Explorationsrate und Diskontierungsfaktor auf den Lernprozess zu untersuchen. Dazu variieren die Schülerinnen und Schüler einzelne Parameter und beobachten den Effekt.

Übertragbarkeit auf andere Projekte
Prinzipiell lässt sich Verstärkendes Lernen auf jedes beliebige Spiel übertragen. Für die Auswahl eines Unterrichtsbeispiels mit dem Q-Table-Algorithmus in Snap! gelten jedoch einige Regeln und Einschränkungen:

Im Idealfall hat ein Beispiel von vornherein eine relativ begrenzte Anzahl an Zuständen, wie etwa Pong oder die Bananenjagd (1). Für komplexere Beispiele, wie etwa Space Invaders, sollte versucht werden, die Anzahl an Zuständen “von Hand” zu begrenzen – das könnte hier etwa bedeuten, dass die Koordinaten der Aliens nur in Bereichen abgefragt werden (2). Treffen auf ein Beispiel weder 1) noch 2) zu, d.h. es gibt zu viele mögliche Zustände, die nicht sinnvoll zu begrenzen sind, ist das Beispiel für den Unterrichtseinsatz nicht geeignet, da der Lernvorgang zu lange andauern würde.

Sicherung und Showcase

Lassen Sie zum Abschluss mehrere Gruppen ihr Pong-Projekt vorstellen und von ihren Erfahrungen mit Belohnungsfunktion und Zustandserfassung berichten.

Ausblick

Anschließend kann in Form eines Ausblicks anhand der exemplarisch bereitgestellten Videos der Einsatz von Verstärkendem Lernen/Q-Learning in der Praxis betrachtet werden.

6.2.2 Stunde 3: Überwachtes Lernen

In dieser Stunde implementieren die Schülerinnen und Schüler mit der linearen Regression selbst ein überwachtes Lernverfahren und nutzen es, um Hauspreise vorherzusagen.

Zeit	Phase	Sozialform/ Lehrerimpuls	Inhalt/Unterrichtsgeschehen	Material
30	(optional)	Partnerarbeit	Durchführung der Unplugged-Aktivität zu überwachtem Lernen, siehe Modul KI-B3.	KI-B3.4.1
15	(optional)	Plenum	Sicherung der zugrunde liegenden Idee, siehe Modul KI-B3.	
30	Erarbeitung	Partnerarbeit	Nun kann überwachtes Lernen auch dazu verwendet werden, Zahlen vorherzusagen. Genau das machen die Schülerinnen und Schüler in dieser Stunde, indem sie selbst ein überwachtes Lernverfahren im realweltlichen Beispiel “Hauspreise” mit Snap! umsetzen und dazu das vorgegebene Skript mithilfe des Arbeitsblatts ergänzen.	KI-A1.2
10	Sicherung	Plenum		
5	Ausblick	Plenum		

Erarbeitung:

Aufbauend auf die Unplugged-Aktivität aus Modul KI-B3 wird das dort erarbeitete Prinzip überwachten Lernens statt für ein Klassifikations- für ein Regressionsproblem verwendet. Die Schülerinnen und Schüler wenden lineare Regression auf dem Arbeitsblatt KI-A1.2 zunächst händisch an, um den Preis einer Immobilie in Abhängigkeit ihrer Wohnfläche vorherzusagen. Anschließend implementieren sie das Verfahren schrittweise in Snap!, um ein Modell für zur Vorhersage der Hauspreise zu trainieren.

Sicherung:

Lassen Sie verschiedene Schülergruppen ihre Lösung vorstellen. Diskutieren Sie, inwieweit sich die Regression von Klassifikation unterscheidet: So unterscheiden sich die Beschriftungen der Daten. Im Falle von Klassifikation, sind die Trainingsdaten mit Kategorien beschriftet, bei Regression mit numerischen Werten. Bei der Klassifikation werden Daten dann mithilfe des Modells den Kategorien zugeordnet (beißt/beißt nicht), bei Regression werden numerische Werte (Hauspreis) vorhergesagt. Außerdem können sie die konkreten Verfahren, die zur Klassifikation bzw. Regression eingesetzt werden, unterscheiden: Während das resultierende Modell der Regression nur die zwei Parameter a und b (und die Information, dass es sich um eine Gerade handelt) enthält, wurde beim Lernen von Entscheidungsbäumen ein Baum als Modell gelernt. Modelle im maschinellen Lernen können also von unterschiedlichster Form sein.

Reflektieren Sie weiterhin, wie hier gelernt wurde: Durch die schrittweise Überprüfung des Fehlers der Vorhersage, wird das Modell schrittweise verbessert. Eine solche schrittweise Annäherung im Unterschied zur direkten “Gesamtlösung” (wie etwa beim Lernen von Entscheidungsbäumen) ist eine weitere typische Funktionsweise von überwachten Lernverfahren.

Ausblick:

Abschließend diskutieren Sie die Antworten auf Aufgabe 4 des Arbeitsblattes: Normalerweise hängt der Hauspreis natürlich nicht nur von den Quadratmetern ab. Welche weiteren Merkmale wären relevant? Mithilfe von multipler linearer Regression können mehrere Merkmale verarbeitet werden. Auch ist der Zusammenhang zwischen Beschriftung und Merkmalen nicht immer linear: Auch hierfür existieren verschiedene komplexere Regressionsvarianten, die nicht-lineare Zusammenhänge beschreiben können.

6.2.3 Stunde 4: Unüberwachtes Lernen

In dieser Stunde implementieren die Schülerinnen und Schüler mit der Vektorquantisierung selbst ein unüberwachtes Lernverfahren, um Kundengruppen zu identifizieren.

Zeit	Phase	Sozialform/ Lehrerimpuls	Inhalt/Unterrichtsgeschehen	Material
30	(optional)	Partnerarbeit	Durchführung der Unplugged-Aktivität zu unüberwachtem Lernen, siehe Modul KI-B3.	KI-B3.5 3 Kupfermünzen (oder andere Tokens in ähnlicher Größe)
15	(optional)	Plenum	Sicherung der zugrunde liegenden Idee, siehe Modul KI-B3.	
15	Erarbeitung I	Partnerarbeit	Die Schülerinnen und Schüler setzen selbst ein unüberwachtes Lernverfahren im realweltlichen Beispiel “Kundendaten” mit Snap! um, indem sie das Sprite “Prototyp” um die Umsetzung des Algorithmus aus der Unplugged Aktivität ergänzen.	KI-A1.3
5	Sicherung I	Plenum	Verschiedene Schülergruppen stellen ihre Lösungen vor. Diskutieren Sie gemeinsam, welche Probleme der Algorithmus in seiner jetzigen Form hat.	
15	Erarbeitung II	Partnerarbeit	Die Schülerinnen und Schüler verbessern den Algorithmus.	
5	Sicherung II	Plenum	Einzelne Gruppen der Schülerinnen und Schüler demonstrieren ihre Ansätze zur Verbesserung des Algorithmus.	
5	Ausblick	Plenum	Demonstration von Beispielen in mehreren Dimensionen.	

Erarbeitung I:
Aufbauend auf der Unplugged-Aktivität aus Modul 1 wird der dort entwickelte Algorithmus mithilfe des Arbeitsblattes KI-A1.3 nun in Snap! umgesetzt. Anstatt von Goldfunden werden hier nun Kundendaten analysiert. In der Vorlage werden bereits zufällige Cluster von Kundendaten erzeugt 

. Die Schülerinnen und Schüler ergänzen nun das Sprite “Prototyp” um die Umsetzung des VQ-Algorithmus entsprechend der Regeln aus der Unplugged Aktivität:

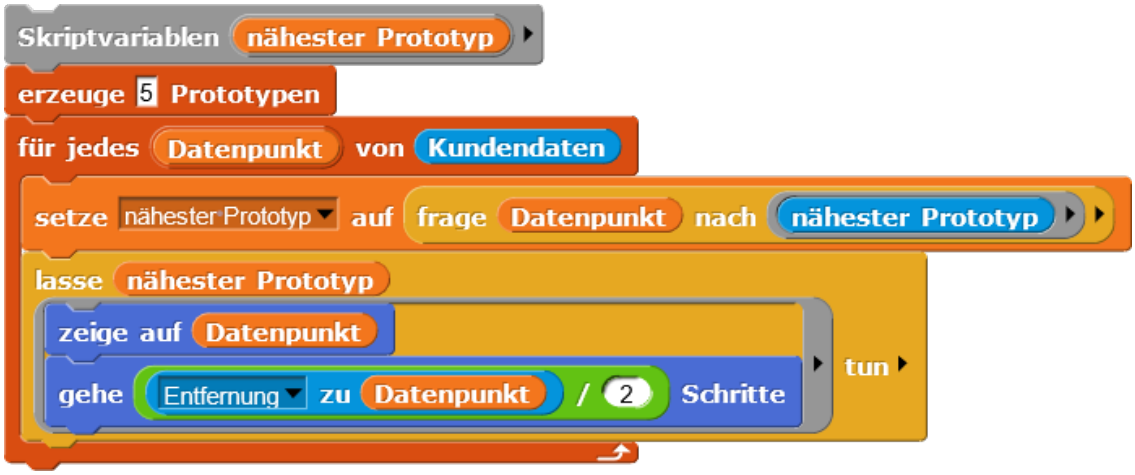
Für jeden Datenpunkt…

1. identifiziere den nächsten Prototyp
2. bewege den Prototyp aus Schritt 1 die Hälfte der Distanz in Richtung des Datenpunkts

In der Vorlage werden bereits zufällig Prototypen erzeugt 

. Auch die Bestimmung des nächsten Prototyps ist bereits durch einen Block vorgegeben 

. Die Schülerinnen und Schüler müssen also lediglich die Abfrage des nächsten Prototyps (1) und das Aktualisieren der Position des jeweiligen Prototyps ergänzen (2).



Durch einen Klick auf die grüne Flagge werden die Datenpunkte neu verteilt, durch einen Klick auf obiges Skript wird der Algorithmus zur Clusteranalyse ausgeführt[7]. Dies ermöglicht es, das Programm wiederholt auf denselben Daten auszuführen.

Durch die Simulation können nun deutlich größere Datenmengen verarbeitet werden, die sich in jedem Durchlauf unterscheiden. Hierzu kann nun eine Vielzahl weiterer Experimente mit dem Verfahren durchgeführt werden: Die Anzahl der Prototypen stimmt in unserer Unplugged-Aktivität zufälligerweise mit der Anzahl der Cluster überein. Das ist hier nicht mehr der Fall! Lassen Sie die Schülerinnen und Schüler daher verschiedene Prototypenanzahlen ausprobieren!

Auf Basis der Prototypen sollen die Schülerinnen und Schüler die gefundenen Kundengruppen auf dem Arbeitsblatt beschreiben und Empfehlungen für individualisierte Werbung angeben. Auch hier sollte erneut die zugrunde liegende Idee von unüberwachtem Lernen aufgegriffen und dessen Merkmale thematisiert und von der Klassifikation im überwachten Lernen unterschieden werden.

[7] Das Skript hat keinen “Wenn grüne Flagge gedrückt”-Hut-Block, weshalb es nicht automatisch mit Klick auf die grüne Flagge ausgeführt wird.

Sicherung I:
Lassen Sie anschließend verschiedene Schülergruppen ihre Lösungen vorstellen. Fragen Sie die Schülerinnen und Schüler, welche Probleme der Algorithmus in seiner jetzigen Form hat. Dazu können Sie Impulse geben wie:

- Wie verhält sich euer Algorithmus, wenn ein Datenpunkt ganz isoliert von den anderen liegt und dieser zufällig als letztes betrachtet wird?
- Wie verhält sich der Algorithmus bei zwei Punktwolken, wenn die Prototypen in der Mitte zwischen den beiden Punktwolken starten?

Einschub: Informatik im Kontext und eigene Daten

Alternativ können Sie die Implementierung des Algorithmus in Snap! auch auf eigenen Daten, die zuvor mit oder durch die Schülerinnen und Schülern erhoben wurden, durchführen. Ein einfaches Beispiel dazu wäre, Breiten- und Längengrade der Wohnorte der Schülerinnen und Schüler zu sammeln, und als csv-Datei in Snap! zu importieren. Diese können nun mit Hilfe der Snap!-Bibliothek "world map" dargestellt werden. Der VQ-Algorithmus kann nun genutzt werden, um geeignete Bushaltestellen für den Schulbus zu identifizieren, an der Implementierung des Algorithmus ändert sich dabei nichts. Ein solches, an "Informatik im Kontext" angelehntes Vorgehen, kontextualisiert die Inhalte in der Lebenswelt der Schüler. Konzepte des Datenmanagements, wie die Erfassung, Bereinigung oder Archivierung von Daten (Data Lifecycle) sowie Methoden der Statistik und Datenanalyse, in unserem Fall Maschinelles Lernen, können hierbei aufgegriffen werden.

Erarbeitung II:
Im nächsten Schritt wollen wir nun unser maschinelles Lernverfahren verbessern: Die Schülerinnen und Schüler haben vermutlich im Rahmen der Unplugged-Aktivität und/oder der Implementierung erste Erfahrungen mit Schwächen des simplen VQ-Algorithmus gemacht. So reagiert der Algorithmus beispielsweise empfindlich darauf, wenn ein Ausreißer (ein isoliert liegender Datenpunkt) als allerletztes verarbeitet wird (das lässt sich gut in Snap! zeigen). Dieses und weitere Probleme (bspw., dass manche Prototypen gar nicht bewegt werden) dient als Anlass, die Schülerinnen und Schüler über eine Optimierung des Algorithmus nachdenken zu lassen.

Mögliche Ideen zur Verbesserung des Algorithmus sind:

- Alle Punkte mithilfe eines "wiederhole"-Blocks mehrfach verarbeiten
- Die Schrittweite der Prototypen mit jedem Durchgang (vgl. Idee zuvor) verringern
- Die Anfangsposition der Prototypen nicht mehr zufällig wählen, sondern mit gleichen Abständen verteilen
- Mitzählen, wie oft ein Prototyp bewegt wurde; Prototypen, die lange nicht bewegt wurden, eher berücksichtigen als solche, die sehr oft bewegt werden

Tipp: Da hier auch konzeptionelle Überlegungen gefragt sind, bietet es sich an, den Schülerinnen und Schülern weiterhin die Materialien der Unplugged-Aktivität zur Verfügung zu stellen, sodass diese ihre Optimierungen zunächst mithilfe des Spiels planen können, ehe diese in Snap*!* umgesetzt werden.

Alternativ bietet sich zur Optimierung des Algorithmus auch ein Wettbewerb an: Den Schülerinnen und Schülern werden eigens erhobene Beispieldaten via csv-Datei zur Verfügung gestellt, die als Datenpunkte eingelesen und mit Hilfe des plotte-2D-Daten-Blocks visualisiert werden. Die Ergebnisse der Schülerinnen und Schüler können dann verglichen werden.

Sicherung II:
Lassen sie nun einzelne Schülergruppen ihre Ansätze zur Verbesserung des Algorithmus vorstellen und demonstrieren.

Ausblick:
Greifen Sie zum Abschluss der Stunde auf, das wir bisher lediglich in zwei Dimensionen gearbeitet haben. Nur so ist eine Visualisierung des Algorithmus möglich. Natürlich ist der VQ-Algorithmus bzw. Unüberwachtes Lernen nicht auf zwei Dimensionen beschränkt. Im Gegenteil, die Suche nach Clustern oder Ausreißern wird vorwiegend für Probleme mit mehr Variablen eingesetzt. Ein mögliches Beispiel in drei Dimensionen (und damit immer noch visualisierbar) stellt die Bildkompression dar:

Bei der Kompression von Grafikdateien werden häufig verschiedene ähnliche Farbtöne zu einer Farbe zusammengefasst, um Speicherplatz zu sparen. Mit Hilfe des VQ-Algorithmus können “ähnlichen Farben” bestimmt werden. Die drei Farbkanäle R, G, und B stellen unsere Dimensionen dar. In der Visualisierung unter https://www.stefanseegerer.de/vq-visualization-3d/ wird gezeigt, wie analog zu den Goldfunden oder Kundendaten für jedes im Bild enthaltene Pixel der nächste Prototyp bestimmt und dessen Position angepasst wird. Zum Abschluss kann das Bild nun komprimiert werden, in dem für jedes Cluster alle zugehörigen Pixel in der Farbe des Prototyps eingefärbt werden.

In realweltlichen Kontexten wie bspw. der Analyse von Kundendaten gilt es oft noch viele weitere Merkmale (Alter, Wert des Warenkorbs, Einkommen, Kunde seit, …) zu berücksichtigen. So viele Merkmale lassen sich dann natürlich nicht mehr graphisch visualisieren, allerdings lässt sich der Abstand zwischen zwei Punkten auch dann definieren, wenn wir nicht mehr nur zwei oder drei, sondern beispielsweise fünf, siebzehn oder zweihundert Dimensionen haben.

7 Einbettung in verschiedene Fächer und Themen

Als Einbettung in ein spezielles Unterrichtsfach bietet sich die Informatik an. Da Informatik und insbesondere das Thema Künstliche Intelligenz nicht in allen Bundesländern fester Bestandteil der Schuldbildung ist es eine mögliche Alternative, dieses Modul fächerübergreifend z.B. im Rahmen einer Projekt- bzw. Themenwoche einzubinden. Künstliche Intelligenz ist ein Querschnittsthema, entsprechend lässt es sich mit vielen anderen Fächern verbinden. Die unter Abschnitt 6 genannten Ideen böten z.B. einen fächerübergreifenden Unterricht mit Mathematik, Geographie / Erdkunde oder Wirtschaft an. Weitere Kombinationen sind denkbar und der Kreativität keine Grenzen gesetzt.

8 Anschlussthemen

Als Anschlussthemen im Zusammenhang mit *IT2School* bieten sich die folgenden Module an:

Von Daten und Bäumen - selbst Daten mit KI auswerten

Möchten Sie sich aufbauend auf die Entwicklung von KI-Algorithmen maschinelles Lernen als Werkzeug zur Analyse großer Datenmengen beschäftigen, empfehlen wir das Modul *Von Daten und Bäumen*.

9 Literatur und Links

- Lindner, A., & Seegerer, S. (2019): **AI Unplugged:** aiunplugged.org
- Seegerer, S., Michaeli, T., & Jatzlau, S. (2020): **Übersicht "So lernen Maschinen":** https://computingeducation.de/proj-ml-uebersicht/
- Seegerer, S., & Michaeli, T. (2021): **MOOC "Die Welt der KI entdecken"**: https://open.sap.com/courses/ai1-de
- Lämmel, U. & Cleve, J. (2012): Künstliche Intelligenz, Hanser, 4. Auflage

10 Arbeitsmaterialien

Das Modul besteht aus folgenden Materialien. Zu allen Arbeitsblättern gibt es auch Musterlösungen.

Nr.	Titel	Beschreibung
KI-A1.1.1	Verstärkendes Lernen - QTable	Arbeitsblatt zu Q-Table-Learning in Snap! Videoerklärung: https://www.youtube.com/watch?v=llpbHXyNk9Q&list=PL9NH5FGrzyCXEiqCIt0AE_XGV2dI4TTLi
KI-A1.1.2	Hilfekarten	Hilfekarten zu Q-Table-Learning in Snap!
KI-A1.2	Überwachtes Lernen – Regression	Arbeitsblatt zur linearen Regression in Snap!
KI-A1.3	Unüberwachtes Lernen - VQ	Arbeitsblatt zur Vektorquantisierung in Snap! Videoerklärung: https://www.youtube.com/watch?v=Ym-fy_O5H38&list=PL9NH5FGrzyCXEiqCIt0AE_XGV2dI4TTLi&index=10

Legende

- Material für Schülerinnen und Schüler
- Material für Lehrkräfte sowie Unternehmensvertreterinnen und Unternehmensvertreter
- Zusatzmaterial

11 Glossar

Begriff	Erläuterung
Agent	Computerprogramm, das zu autonomem Verhalten fähig ist
Maschinelles Lernen	Beim maschinellen Lernen (ML) leiten Computer Zusammenhänge aus Daten ab. Das Gelernte wird in einem Modell gespeichert.
Künstliche Intelligenz	Künstliche Intelligenz beschreibt ein Forschungsgebiet der Informatik, das sich damit beschäftigt, menschliche kognitive Fähigkeiten durch Computersysteme nachzubilden.
Verstärkendes Lernen	Beim verstärkenden Lernen lernt der Agent in Interaktion mit seiner Umwelt durch wiederholte Belohnungen oder Bestrafungen die Erfolgsaussichten seiner Aktionen besser einzuschätzen und somit seine Strategie zu optimieren.

Q-(Table-)Learning	**Q-Table-Learning ist ein Algorithmus für verstärkendes Lernen, dessen zentrales Prinzip es ist, pro Zustand die jeweiligen Aktionen zu bewerten. Dazu wird eine (Q-)Tabelle erzeugt, mit der ein Agent für jeden Zustand eine Bewertung für jede Aktion verwaltet. Dieser (Q-)Wert in der Tabelle gibt an, welche Qualität (daher auch der Name Q-Learning) eine Aktion in einem Zustand hat.**
Diskontierungsfaktor (verstärkendes Lernen)	Der Diskontierungsfaktor y (typischerweise $0 < y < 1$) sagt aus, dass eine Belohnung oder Bestrafung, die der Agent bei verstärkendem Lernen n-Schritte in der Zukunft erhält, um den Faktor y^n diskontiert wird, also weniger Wert ist.
Explorationsrate (verstärkendes Lernen)	Über die sog. Explorationsrate lässt sich steuern, wie bereitwillig der Agent bei verstärkendem Lernen andere Aktionen als die zurzeit am besten bewertete ausprobiert. Eine hohe Explorationsrate bedeutet, dass der Agent eher bereit ist, eine neue Aktion auszuprobieren.
Lernrate	Die Lernrate legt fest, wie stark neue Erfahrungen bei der Anpassung des Modells berücksichtigt werden. Bei der linearen Regression sind dies die Parameter a und b, bei Q-Learning die Bewertungen der Zustände. Die Lernrate ist dabei ein Wert zwischen 0 und 1. Eine Lernrate von 0 führt dazu, dass überhaupt nichts gelernt wird, weil neu erworbenes Wissen einfach nicht genutzt wird. Eine Lernrate von 1 wiederum bedeutet, dass nur die aktuellste Information berücksichtigt wird. Normalerweise wird die Lernrate daher zwischen diesen beiden Extremen liegen.
Überwachtes Lernen	Bei überwachtem Lernen wird aus beschrifteten Daten eine Zuordnung von Daten zu Beschriftung gelernt, die dann auf weitere, unbeschriftete Daten angewendet werden kann.
Lineare Regression	Die lineare Regression ist ein überwachtes Lernverfahren zur Beschriftung eines unbekannten Datenpunkts bei vorhandenen beschrifteten Daten mit einer Zahl
Unüberwachtes Lernen	Unüberwachtes Lernen versucht Ähnlichkeiten in unbeschrifteten Eingaben zu erkennen und so Muster (Ausgabe) zu finden.
Vektorquantisierung	Vektorquantisierung ist ein Algorithmus für unüberwachtes Lernen, dessen Idee es ist, Punkte zu finden, die ein Cluster prototypisch beschreiben.

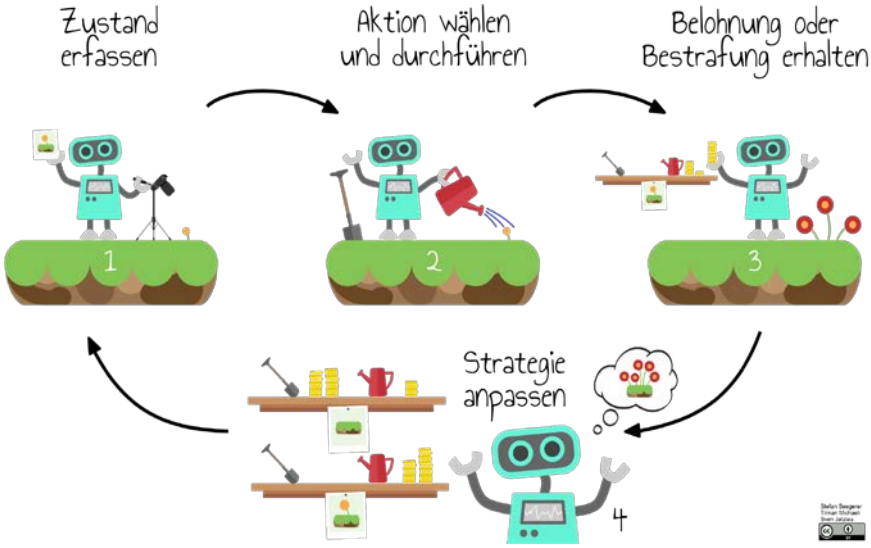
11 Fragen, Feedback, Anregungen

Sie haben das Modul ausprobiert und nun Fragen, Anregungen oder Feedback für uns? Darüber freuen wir uns, denn mit Ihren Erfahrungen können wir Schritt für Schritt einen FAQ (Frequently Asked Questions) für die neuen KI-Module aufbauen oder die Module weiter entwickeln.

Bitte füllen Sie folgende Umfrage über Surveymonkey aus: https://bit.ly/3DWXMZq über den folgenden QR-Code kommen Sie ebenfalls zur Surveymonkey-Umfrage:

Sie können sich auch gerne unter bildung@wissensfabrik.de melden.

Verstärkendes Lernen



Bananenjagd

Aufgabe 1:

Öffne die Vorlage des Spiels “Bananenjagd”: https://bit.ly/A1-ban

Sie enthält bereits alle benötigten Blöcke, um einen selbstlernenden Agenten zu erschaffen – allerdings nicht in der richtigen Reihenfolge!

Ordne die Blöcke in der passenden Reihenfolge. Falls du nicht weiterkommst, hilft dir das obige Schaubild.

Aufgabe 2: Beschreibe das Lernen des Agenten:

Aufgabe 3:
Ziehe folgende Blöcke in den Skriptbereich und führe diese durch eine Klick aus:

Betrachte die Tabelle, die das Modell repräsentiert. Mit einem Doppelklick kannst du sie fixieren.

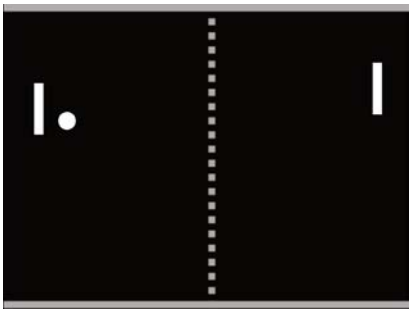
Erläuterungen zur Tabelle: Der Zustand ist in der Spalte A angetragen, Spalte B (springen) und C (nichts tun) beinhalten die Bewertung für die jeweilige Aktion in diesem Zustand.

Was bedeutet es, wenn der Wert in Spalte B größer ist als der in Spalte C?

Worauf lässt ein hoher negativer Wert schließen?

Pong

Die Idee von verstärkendem Lernen lässt sich auch auf andere Spiele übertragen, beispielsweise auf den Arcade-Klassiker Pong. Der Agent ist hier einer der Schläger.



Aufgabe 1: Vergleiche Pong mit der Bananenjagd und notiere die Antworten für Pong!

- Welche Aktionen kann der Agent ausführen? _____
- Was ist der Zustand der Umwelt? _____
- Wie sollte der Agent belohnt/bestraft werden? _____

Aufgabe 2: Öffne folgendes Projekt: https://bit.ly/A1-pon

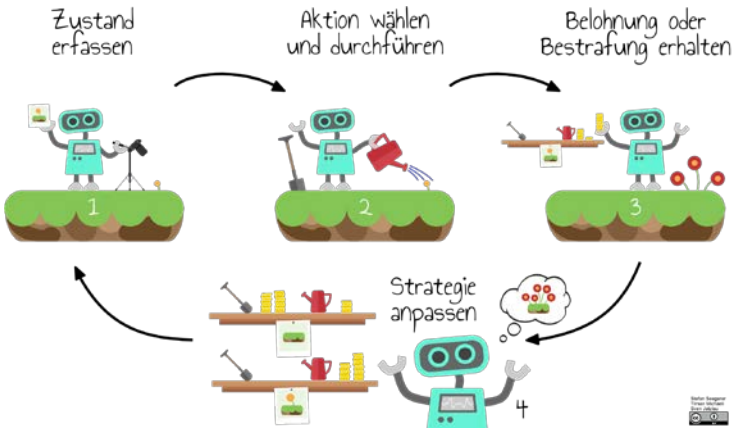
Der linke Schläger wird bereits durch den Computer gesteuert. Deine Aufgabe ist es, den rechten Schläger lernen zu lassen, erfolgreich Pong zu spielen. Übertrage das Prinzip aus dem Bananenjagdspiel nun auf Pong.

Gib deinem Modell einige Minuten Zeit und beobachte, ob es ein sinnvolles Verhalten lernt.

Experimentiere mit möglichen Belohnungen und Bestrafungen! Notiere, welche Werte sich als geeignet erwiesen haben:

Wie würdest du das gelernte Verhalten des Pong-Schlägers beschreiben?

Verstärkendes Lernen



Bananenjagd

Aufgabe 1:

Öffne die Vorlage des Spiels “Bananenjagd”: https://bit.ly/A1-ban

Sie enthält bereits alle benötigten Blöcke, um einen selbstlernenden Agenten zu erschaffen – allerdings nicht in der richtigen Reihenfolge!

Ordne die Blöcke in der passenden Reihenfolge. Falls du nicht weiterkommst, hilft dir das obige Schaubild.

https://bit.ly/A1-ban-l

Aufgabe 2: Beschreibe das Lernen des Agenten:

> Das Verhalten erscheint zunächst zufällig, der Agent lernt im Anschluss länger auf dem Boden zu bleiben, allerdings rennt das Äffchen trotzdem hin und wieder ins Fass hinein und auch der Doppelsprung ist noch ein Problem.

Aufgabe 3:
Ziehe folgende Blöcke in den Skriptbereich und führe diese durch eine Klick aus:

Betrachte die Tabelle, die das Modell repräsentiert. Mit einem Doppelklick kannst du sie fixieren.

Erläuterungen zur Tabelle: Der Zustand ist in der Spalte A angetragen, Spalte B (springen) und C (nichts tun) beinhalten die Bewertung für die jeweilige Aktion in diesem Zustand.

Was bedeutet es, wenn der Wert in Spalte B größer ist als der in Spalte C?

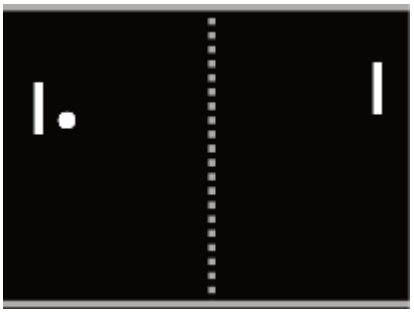
> “Springen” wird in diesem Zustand in Zukunft häufiger gezeigt als “Nichts tun”.

Worauf lässt ein hoher negativer Wert schließen?

Diese Aktion im jeweiligen Zustand wurde häufig bestraft.

Pong

Die Idee von verstärkendem Lernen lässt sich auch auf andere Spiele übertragen, beispielsweise auf den Arcade-Klassiker Pong. Der Agent ist hier einer der Schläger.



Aufgabe 1: Vergleiche Pong mit der Bananenjagd und notiere die Antworten für Pong!

- Welche Aktionen kann der Agent ausführen?

  nach oben, unten, nichts tun

- Was ist der Zustand der Umwelt?

  Position des Balles oder Abstand zum Schläger

- Wie sollte der Agent belohnt/bestraft werden?

  Wenn ein “Tor” erzielt wurde, bestrafen, wenn der Ball getroffen wurde, belohnen.

Aufgabe 2: Öffne folgendes Projekt: https://bit.ly/A1-pon

Der linke Schläger wird bereits durch den Computer gesteuert. Deine Aufgabe ist es, den rechten Schläger lernen zu lassen, erfolgreich Pong zu spielen. Übertrage das Prinzip aus dem Bananenjagdspiel nun auf Pong.

Gib deinem Modell einige Minuten Zeit und beobachte, ob es ein sinnvolles Verhalten lernt.

https://bit.ly/A1-pon-l

Experimentiere mit möglichen Belohnungen und Bestrafungen! Notiere, welche Werte sich als geeignet erwiesen haben:

individuell, ein Beispiel siehe Hilfekarten

Wie würdest du das gelernte Verhalten des Pong-Schlägers beschreiben?

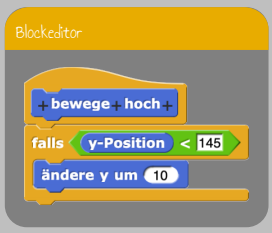
Der Schläger lernt, auf der Höhe des Balles zu bleiben

Modell anlegen

0

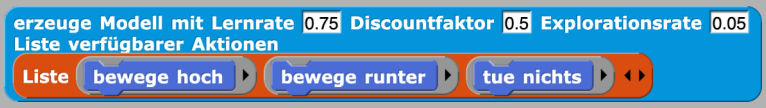
Modell anlegen

Mögliche Aktionen überlegen



Rechtsklick > Umringen

Modell erzeugen



Modell speichern

Neue Variable anlegen

Name vergeben

Modell in Variable speichern

Ins Programm einbauen

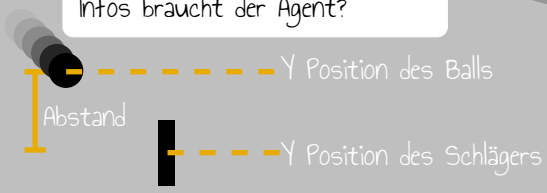
Zustand erfassen

1

Zustand erfassen

Zustand erfassen

Was ist wichtig, welche Infos braucht der Agent?



Tipps

Je größer die Anzahl der Zustände, desto länger dauert das Training

statt

Oft reichen wenige Informationen

Eine Liste kann mehrere Einträge enthalten, z.B.

Werte zu Bereichen zusammenfassen

Ins Programm einbauen

Aktion auswählen und ausführen

2

Aktion durchführen

Beste Aktion ermitteln

Zustand aus Karte 2

führe beste Aktion aus

Modell aus Karte 1

Die Variable enthält die beste Aktion, z.B.

beste Aktion

bewege hoch

Beste Aktion ausführen

Die Variable berichtet einen Block und kann daher so ausgeführt werden

führe beste Aktion aus

Ins Programm einbauen

Belohnung erhalten

3

Belohnung erhalten

Belohnung bestimmen

Wann sollten wir den Agenten belohnen, wann bestrafen?

Statt

kann auch ein eigener Block erzeugt werden.

Belohnung

Tipps

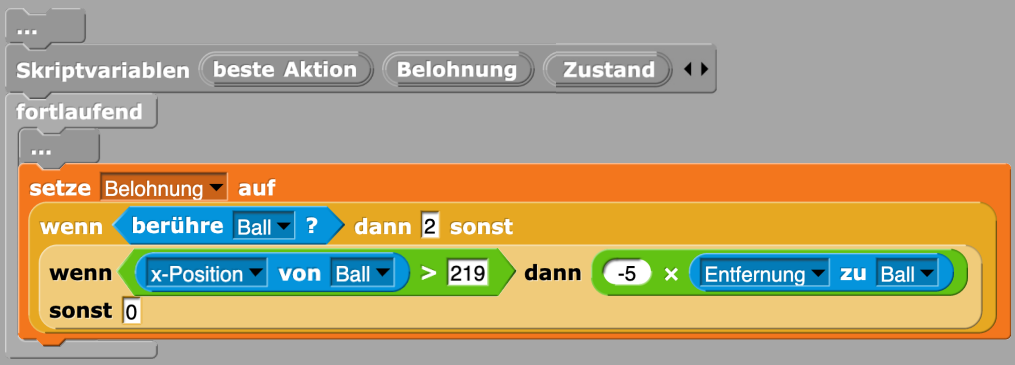
Bedingungen verschachteln

Ball gelangt an rechten Rand

Belohnung und Bestrafung dynamisch gestalten

Bestrafung abhängig von Entfernung zu Ball

Ins Programm einbauen

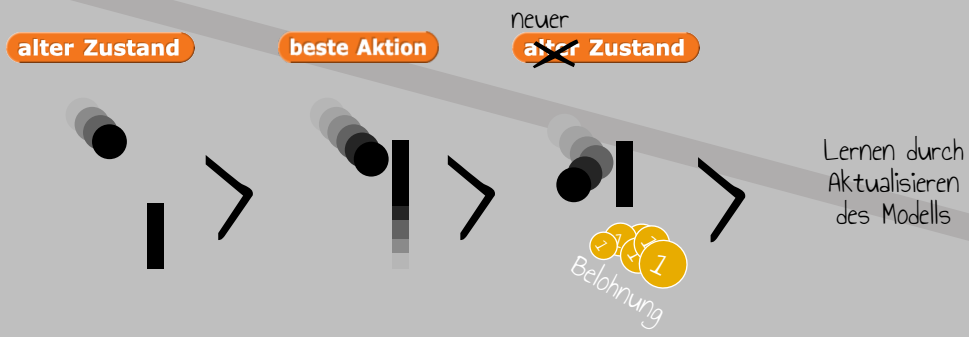


Strategie anpassen

4

Strategie anpassen

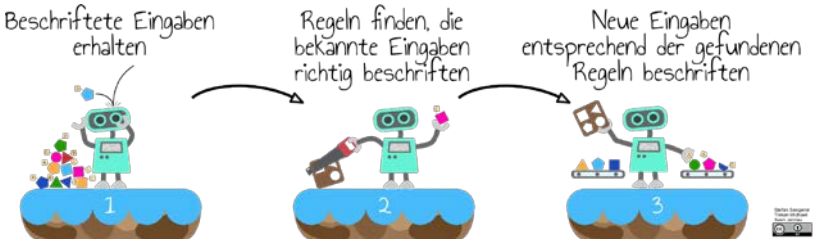
Informationen verarbeiten



Modell aktualisieren

Ins Programm einbauen

Überwachtes Lernen



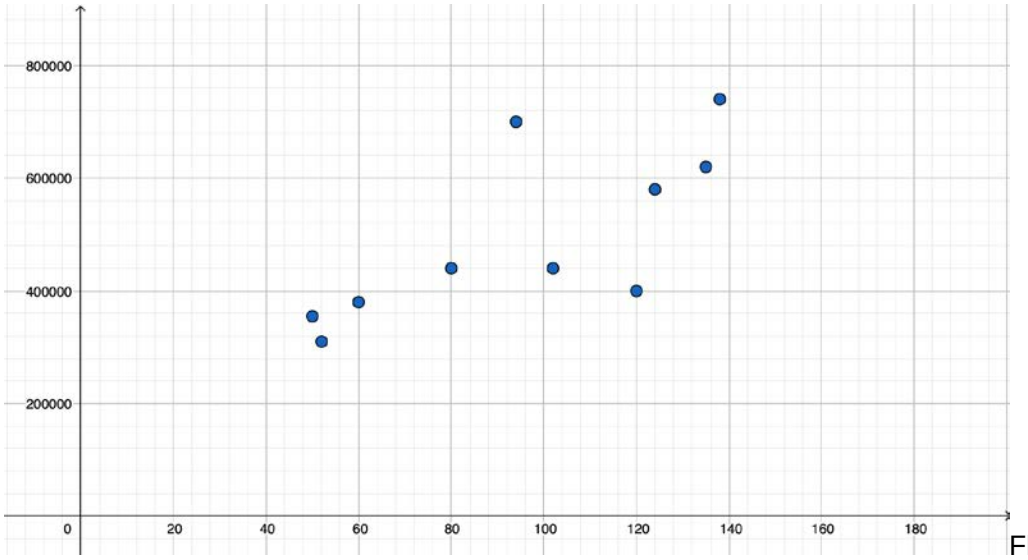
Überwachtes Lernen kann verwendet werden, um eine Beschriftung aus einer Reihe von vorgegebenen Beschriftungen wie **beißt** und **beißt nicht** vorherzusagen, es kann aber auch dazu verwendet werden, einen Zahlenwert vorherzusagen, etwa die voraussichtlich anfallenden Schadenssummen bei einer Versicherung oder die Entwicklung von Aktienkursen und Hauspreisen.

In diesem Arbeitsmaterial setzt du mit linearer Regression ein überwachtes Lernverfahren in Snap*!* um, um Hauspreise abhängig von ihrer Fläche in Quadratmetern vorherzusagen.

1. Aufgabe: Lineare Regression

a) In folgender Abbildung sind die Marktpreise und zugehörige Fläche in Quadratmetern für verschiedene Häuser angezeichnet. Zeichne eine Gerade ein, die sich den gegebenen Punkten bestmöglich annähert.

Marktpreis



b) Triff nun mithilfe deiner Gerade eine Vorhersage wie hoch der Preis für ein Haus mit 180m² Fläche ist: _____

Überwachtes Lernen wird neben Klassifikationsproblemen auch bei **Regressionsproblemen** eingesetzt, in denen Eingaben keine Klasse (wie „beißt“ oder „beißt nicht“) sondern ein numerischer Wert (wie etwa ein Hauspreis) zugeordnet werden soll. Der Computer versucht dazu einen Zusammenhang zwischen Eingabedaten (Fläche) und deren Beschriftung (Hauspreis) zu finden. Das Finden einer Gerade bezeichnet man als **lineare Regression**.

Mathematisch wird eine Gerade durch die **Steigung** a und den y-Achsenabschnitt b wie folgt beschrieben:

$$y = a \cdot x + b$$

Aufgabe des Computers ist es also, geeignete Werte für a und b zu finden. Die Parameter a und b stellen für den Computer also das Modell dar, mit deren Hilfe dann auch Vorhersagen für weitere Häuser mit ihrer gegeben Fläche getroffen werden können.

2. Aufgabe: Vorhersage in Snap!

Zunächst benötigen wir einen Block, der für ein bereits existierendes Modell Vorhersagen trifft.

a) Öffne die Vorlage des Projekts “Hauspreise”: https://bit.ly/A1-reg

Verwende anschließend den 

, um die Trainingsdaten auf der Bühne anzeigen zu lassen.

b) Implementiere den Block 

,

indem du via Rechtsklick>Bearbeiten… in den Bearbeitungsmodus wechselst und den Wert berechnest und über den berichte Block zurückgibst.

Zur Erinnerung: $Hauspreis = a \cdot x + b$ mit Fläche x

Die Vorlage speichert die Parameter a und b in den beiden gleichnamigen Skriptvariablen

c) Rufe den gerade implementierten Block mit den folgenden Eingaben auf und überprüfe, ob du jeweils das angegebene Ergebnis erhältst.

3. Aufgabe: Lernen

Noch fehlt uns allerdings der wichtigste Schritt eines maschinellen Lernverfahrens: das Lernen des Modells und damit das Finden geeigneter Parameter a und b. Der Computer löst das aber nicht wie wir vorhin mit “draufgucken”, sondern nimmt eine Gerade und passt sie schrittweise an.

Implementiere nun den Block 

, indem du via Rechtsklick>Bearbeiten… in den Bearbeitungsmodus wechselst.

Einige vorbereitende Schritte sind bereits erledigt: Die Variable 

enthält alle Eingabedaten, die Variable 

die zugehörigen Beschriftungen.

Hinweis: Da wir hier immer dieselben Operationen für alle Eingabedaten und erwarteten Ausgaben durchführen, können wir den Blöcken als Eingaben statt Einzelwerten auch Listen übergeben.

a) Rufe zunächst den Block 

mit 

und dem aktuellen Modell 

auf. Speichere das Ergebnis in der Variablen 

.

b) Diese Vorhersagen sollen nun mit den erwarteten Ausgaben verglichen werden. Bestimme dazu den Fehler, indem du die Differenz aus 

und 

bestimmst.

c) Um diesen Fehler zu reduzieren, muss das Modell und damit die Variablen a und b entsprechend angepasst werden. Passe dazu a und b gemäß folgender Formel an:

d) Überprüfe die gelernte Gerade visuell, indem du den folgenden Block ausführst:

4. Aufgabe: Mehr als nur Quadratmeter

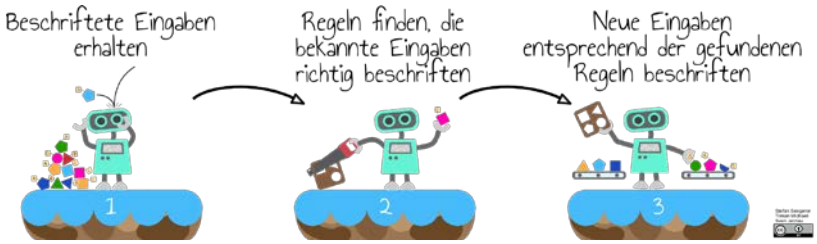
In der Realität hängt der Hauspreis aber natürlich nicht nur von der Fläche, sondern auch von vielen weiteren Faktoren ab. Sammle weitere für den Hauspreis relevante Merkmale:

_____

_____

_____

Überwachtes Lernen



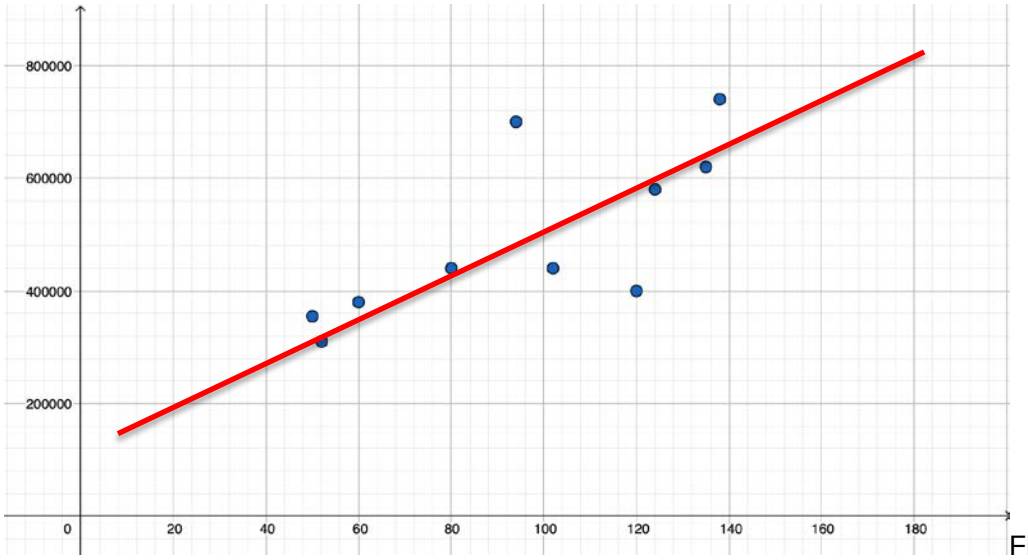
Überwachtes Lernen kann verwendet werden, um eine Beschriftung aus einer Reihe von vorgegebenen Beschriftungen wie **beißt** und **beißt nicht** vorherzusagen, es kann aber auch dazu verwendet werden, einen Zahlenwert vorherzusagen, etwa die voraussichtlich anfallenden Schadenssummen bei einer Versicherung oder die Entwicklung von Aktienkursen und Hauspreisen.

In diesem Arbeitsmaterial setzt du mit linearer Regression ein überwachtes Lernverfahren in Snap*!* um, um Hauspreise abhängig von ihrer Fläche in Quadratmetern vorherzusagen.

1. Aufgabe: Lineare Regression

a) In folgender Abbildung sind die Marktpreise und zugehörige Fläche in Quadratmetern für verschiedene Häuser angezeichnet. Zeichne eine Gerade ein, die sich den gegebenen Punkten bestmöglich annähert.

Marktpreis



b) Triff nun mithilfe deiner Gerade eine Vorhersage wie hoch der Preis für ein Haus mit 180m² Fläche ist: abhängig von der gewählten Gerade, aber um die 800.000 €

Überwachtes Lernen wird neben Klassifikationsproblemen auch bei **Regressionsproblemen** eingesetzt, in denen Eingaben keine Klasse (wie „beißt“ oder „beißt nicht“) sondern ein numerischer Wert (wie etwa ein Hauspreis) zugeordnet werden soll. Der Computer versucht dazu einen Zusammenhang zwischen Eingabedaten (Fläche) und deren Beschriftung (Hauspreis) zu finden. Das Finden einer Gerade bezeichnet man als **lineare Regression**.

Mathematisch wird eine Gerade durch die **Steigung** a und den y-Achsenabschnitt b wie folgt beschrieben:

$$y = a \cdot x + b$$

Aufgabe des Computers ist es also, geeignete Werte für a und b zu finden. Die Parameter a und b stellen für den Computer also das Modell dar, mit deren Hilfe dann auch Vorhersagen für weitere Häuser mit ihrer gegeben Fläche getroffen werden können.

2. Aufgabe: Vorhersage in Snap!

Zunächst benötigen wir einen Block, der für ein bereits existierendes Modell Vorhersagen trifft.

Fertiges Projekt als Lösung für Aufgabe 2 und 3: https://bit.ly/A1-reg-l

a) Öffne die Vorlage des Projekts “Hauspreise”: https://bit.ly/A1-reg

Verwende anschließend den 

, um die Trainingsdaten auf der Bühne anzeigen zu lassen.

b) Implementiere den Block 

,

indem du via `Rechtsklick>Bearbeiten…` in den Bearbeitungsmodus wechselt und den Wert berechnest und über den berichte Block zurückgibst.

Zur Erinnerung: $Hauspreis = a \cdot x + b$ mit Fläche x

Die Vorlage speichert die Parameter a und b in den beiden gleichnamigen Skriptvariablen

c) Rufe den gerade implementierten Block mit den folgenden Eingaben auf und überprüfe, ob du jeweils das angegebene Ergebnis erhältst.

3. Aufgabe: Lernen

Noch fehlt uns allerdings der wichtigste Schritt eines maschinellen Lernverfahrens: das Lernen des Modells und damit das Finden geeigneter Parameter a und b. Der Computer löst das aber nicht wie wir vorhin mit “draufgucken”, sondern nimmt eine Gerade und passt sie schrittweise an.

Implementiere nun den Block 

, indem du via `Rechtsklick>Bearbeiten…` in den Bearbeitungsmodus wechselst.

Einige vorbereitende Schritte sind bereits erledigt: Die Variable Eingabedaten enthält alle Eingabedaten, die Variable Erwartete Ausgaben die zugehörigen Beschriftungen.

Hinweis: Da wir hier immer dieselben Operationen für alle Eingabedaten und erwarteten Ausgaben durchführen, können wir den Blöcken als Eingaben statt Einzelwerten auch Listen übergeben.

a) Rufe zunächst den Block sage Wert vorher für Eingabe: mit Modell: mit Eingabedaten und dem aktuellen Modell Liste a b auf. Speichere das Ergebnis in der Variablen Vorhersagen.

b) Diese Vorhersagen sollen nun mit den erwarteten Ausgaben verglichen werden. Bestimme dazu den Fehler, indem du die Differenz aus Vorhersagen und Erwartete Ausgaben bestimmst.

c) Um diesen Fehler zu reduzieren, muss das Modell und damit die Variablen a und b entsprechend angepasst werden. Passe dazu a und b gemäß folgender Formel an:

d) Überprüfe die gelernte Gerade visuell, indem du den folgenden Block ausführst:

4. Aufgabe: Mehr als nur Quadratmeter

In der Realität hängt der Hauspreis aber natürlich nicht nur von der Fläche, sondern auch von vielen weiteren Faktoren ab. Sammle weitere für den Hauspreis relevante Merkmale:

Etwa Lage/Viertel, Baujahr, Energieverbrauch, letzte Sanierung, Stockwerk, Anzahl der Räume, …

Unüberwachtes Lernen in Snap!

Analyse von Kundendaten

Aufgabe 1:

Öffne folgende Vorlage: https://bit.ly/A1-war

Im Sprite “Prototyp” wird bereits eine feste Anzahl an Prototypen erzeugt und zufällig im Raum platziert. Implementiere mit Hilfe der bereits im Skriptbereich befindlichen Blöcke folgendes Verhalten:

Für jeden Datenpunkt…

1. Frage den Datenpunkt nach dem nächsten Prototyp.
2. Lasse diesen Prototypen die Hälfte der Distanz in Richtung des Datenpunkts bewegen.

Durch einen Klick auf die grüne Flagge werden die Datenpunkte neu verteilt, durch einen Klick auf das Skript wird der Algorithmus zur Clusteranalyse ausgeführt. Dies ermöglicht es, das Programm wiederholt auf denselben Daten auszuführen.

Aufgabe 2:

Klicke auf die grüne Flagge und erzeuge so einen neuen Datensatz. Beschreibe, welche Cluster der Algorithmus in den Daten findet! Wähle dazu eine geeignete Anzahl der Prototypen für die Daten.

Welche Werbeangebote würdest du als Betreiber eines Onlineshops für die verschiedenen Cluster anbieten?

Optimierung

Aufgabe:

Verbessere den Algorithmus in Snap! Dazu bietet es sich an, mit Hilfe des “Unplugged”-Spieles zu experimentieren.

Hinweis: Eine erste Idee könnte es sein, den Algorithmus mehrfach auszuführen.

Unüberwachtes Lernen in Snap!

Analyse von Kundendaten

Aufgabe 1:

Öffne folgende Vorlage: https://bit.ly/A1-war

Im Sprite “Prototyp” wird bereits eine feste Anzahl an Prototypen erzeugt und zufällig im Raum platziert. Implementiere mit Hilfe der bereits im Skriptbereich befindlichen Blöcke folgendes Verhalten:

> Für jeden Datenpunkt…
>
> 1. Frage den Datenpunkt nach dem nächsten Prototyp.
> 2. Lasse diesen Prototypen die Hälfte der Distanz in Richtung des Datenpunkts bewegen.

Durch einen Klick auf die grüne Flagge werden die Datenpunkte neu verteilt, durch einen Klick auf das Skript wird der Algorithmus zur Clusteranalyse ausgeführt. Dies ermöglicht es, das Programm wiederholt auf denselben Daten auszuführen.

https://bit.ly/A1-war-l

Aufgabe 2:

Klicke auf die grüne Flagge und erzeuge so einen neuen Datensatz. Beschreibe, welche Cluster der Algorithmus in den Daten findet! Wähle dazu eine geeignete Anzahl der Prototypen für die Daten.

> Individuell, etwa Kundengruppen die günstig, aber häufig kaufen (Schnäppchenjäger), Kunden die seltene teure Einkäufe tätigen, oder aber Kunden die häufig und teuer einkaufen.

Welche Werbeangebote würdest du als Betreiber eines Onlineshops für die verschiedenen Cluster anbieten?

> Für Kunden mit häufigen, aber günstigen Einkäufen könnte etwa ein „Kaufe 3 für 2“ Angebot gestaltet werden, für Kunden mit seltenen, aber teuren Einkäufen etwa ein bestimmter Rabatt sobald ein gewisser Warenkorbwert erreicht ist. Zahlreiche weitere Möglichkeiten sind hier denkbar.

Optimierung

Aufgabe:

Verbessere den Algorithmus in Snap! Dazu bietet es sich an, mit Hilfe des “Unplugged”-Spieles zu experimentieren.

Hinweis: Eine erste Idee könnte es sein, den Algorithmus mehrfach auszuführen.

- Das Verfahren in mehreren Iterationen durchführen, um den Einfluss von Ausreißern zu reduzieren:
  https://snap.berkeley.edu/snap/snap.html#present:Username=seegerer&ProjectName=MOOC-UL-Gold-Rush-DE-solution-multiple-iterations
- Die Schrittweite der Prototypen mit der Zeit reduzieren:
  https://snap.berkeley.edu/snap/snap.html#present:Username=seegerer&ProjectName=MOOC-UL-Gold-Rush-DE-solution-decrease-step-size