Reinforcement learning für intelligente Brownsche Dynamik

Versuchsprotokoll als angeleitetes Lernprojekt

von

Jane Nadworna (2933256)

Marcel Funk (2963307)

Gutachter: Paul A. Monderkamp

Versuchsdatum: Januar 2023

Inhaltsverzeichnis

1 Einleitung

Folgendes Protokoll behandelt den Inhalt zur Bearbeitung des angeleiteten Lernprojekts „Reinforcement learning für intelligente Brownsche Dynamik“, welches am Institut der Theoretischen Physik II für weiche Materie angeboten wird. Dieses Lernprojekt soll einen Einstieg in die Anwendung des *reinforcement learning* bieten, welches einen der verschiedenen Bereiche des *Maschine Learning* (ML) bildet. ML, als Teilgebiet der Künstlichen Intelligenz, hat in den letzten Jahren zunehmend an Beliebtheit gewonnen. Nicht nur in der Forschung, sondern auch in der Wirtschaft(?) sind der Umgang und erste Erfahrungen mit ML in verschiedensten Feldern gefragt, was sich schnell bemerken lässt, wenn man sich nach einer Arbeitsstelle in der IT-Branche umschaut. Die durch dieses Lernprojekt vermittelten Grundlagen des reinforcement learning sind deshalb unter anderem auch für die eine oder andere Person in der Zukunft nutzbar.
Neben dem Element des ML wird in diesem Lernprojekt die Durchführung der Aufgaben mit der Theorie des *Random Walk* kombiniert, um ein besseres Verständnis von diffusiven Prozessen in der weichen Materie zu erlangen.

2 Theorie

2.1 Random Walk

Der Random Walk (RW) stellt ein mathematisches Modell dar, welches die theoretischen Grundlagen von diffusiven Prozesse beschreibt und genutzt wird, um einfache Bewegungen die in der Natur vorkommen zu modellieren. Beispiele dafür sind die Brownsche Bewegung, bei welcher es sich um einen sogenannten unbiased (dt. unvoreingenommen) RW handelt oder die Bewegung von manchen Bakterienarten, dessen Bewegung durch den biased (dt.voreingenommenen) RW beschrieben wird.

In den folgenden Aufgaben beschäftigen wir uns mit dem unbiased RW, was bedeutet dass sich das Beobachtungsobjekt, meist ein Teilchen, mit derselben Wahrscheinlichkeit in jede Raum Richtung bewegt [1]. Bei dieser isotropischen Bewegung wird im Ein-dimensionalen Fall angenommen, dass sich das Teilchen auf einem Zahlenstrahl bewegt, welcher in einheitlichen Abständen in diskrete Punkte eingeteilt ist und das Teilchen bei x = 0 startet. Nach einem Zeitschritt τ kann sich das Teilchen mit einen Schritt $\Delta x = \pm 1$ nach links l oder rechts r auf dem Strahl bewegen. Die Wahrscheinlichkeitsverteilung $P_N(m)$ für die Position $m = -l + r$ nach N Zeitschritten ist gegeben durch folgende Form einer Binomialverteilung [2]

$$P_N(m) = \frac{N!}{(m+N)/2!(N-(m+N)/2)!} p^{\frac{m+N}{2}} (1-p)^{\frac{N-m}{2}}. \tag{1}$$

Dabei beschreibt p die Wahrscheinlichkeit eine Schritt nach rechts zu gehen und ist beim unbiased RW mit $p = q = 1/2$ gegeben. Wobei q die Wahrscheinlichkeit nach links zu gehen darstellt und die Beziehung $p + q = 1$ gilt. Für große N konvergiert diese Verteilung gegen die Normalverteilung $P(x,t)$

$$P(x,t) = \frac{1}{\sqrt{4Dt}} exp\Big(\frac{-x^2}{4Dt}\Big), \tag{2}$$

mit der Diffusionskonstante $D = \frac{\Delta x^2}{2\tau}$.

2.2 Machine Learning/ Q-Algorithmus

Das ML umfasst unterschiedliche Anwendungsbereiche und wird dementsprechend in Kategorien klassifiziert. Die gröbste Unterscheidung wird zwischen dem supervised learning, dem unsupervised learing und dem reinforcment learning gemacht. Beim supervised learing wird der Lernprozess überwacht und es stehen bereits Daten zur Verfügung um die Input-Daten mit den schon zu Verfügung stehenden Daten zu vergleichen. Beim unsupervised learning wird dem lernenden System kein Feedback bereitgestellt und wird oft beim Clustering von großen Datensets genutzt . Beim Reinforcement learning wird auf die Erzeugung eigener Daten gesetzt, welche genutzt werden um den Lernprozess während der Ausführen stetig zu optimieren und die gegebene Aufgabe zu komplettieren [3].
Eine häufig verwendete Methode des reinforcment learning ist der Q-Algorithmus, mit welchem wir uns in den folgenden Aufgaben beschäftigen.

3 Aufgaben

3.1 Mittleres Verschiebungsquadrat

Als Einstieg beschäftigen wir uns mit der zufälligen Bewegung eines Teilchens. Dabei betrachten wir die sogenannte *Mean squared Displacement* (MSD), um Aussagen über diese Bewegung treffen zu können. Die MSD beschreibt die gemittelte quadrierte Verschiebung des gegeben Teilchens relativ zu seiner Ausgangsposition. Im vorliegenden Navigationsproblem sind sowohl die Bewegung als auch die Zeit diskret gegeben. Das heißt das nach Ausführen eines Diffusionsschritts nach der Zeit $\tau = 1$, sich die momentane Position des Teilchens um $\Delta x = \pm 1$ ändert. Aus dem Zusammenhang

$$D = \frac{\Delta x^2}{2\tau} \tag{3}$$

ergibt sich somit für jeden Diffusionsschritt der Wert $D = 1/2$ für die Bewegung. Die Diffusionskonstante kann im Laufe der Simulation effektiv verändert werden, indem eine Wahrscheinlichkeit eingeführt wird, mit welcher das Durchführen eines Diffusionsschritts zu jedem Zeitpunkt beeinflusst wird. Diese Wahrscheinlichkeit $P_{diffstep}$ wurde in Abhängigkeit der festgelegten Diffusions-

konstante D auf $P_{diffstep} = 2D$ gelegt. Nun wird jeder Schritt Δx mit einer Wahrscheinlichkeit von $P_{diffstep}$ ausgeführt.

Im gegebenen Code wird die MSD aus $N_{episodes} = 20000$ Episoden bei einer Laufzeit von jeweils $tmax_{MSD} = 100$ errechnet und ein linearer Fit an die Daten angelegt, aus welchem die effektive Diffusionskonstante D_{eff} bestimmt werden kann. D_{eff} ergibt aus der Steigung m des Fits zu $D_{eff} = \frac{m}{2}$. Bei Auftragung der MSD gegen die Laufzeit t (siehe Abb.1) ist erkennbar, dass sich für die Simulationsdaten ein linearer Zusammenhang zwischen der MSD und der Laufzeit ergibt. Aus diesem Zusammenhang lässt sich auch erkennen, dass es sich um eine diffusive Bewegung handelt.

Das Variieren der gegebenen Diffusionskonstante D ergibt eine Veränderung der effektiven Diffusionskonstante D_{eff}. Wobei die Diffusionskonstante auf den Wert $D = 0.5$ beschränkt ist, da größere Werte zu einer Wahrscheinlichkeit $P_{diffstep} > 1$ führen würden, welche nicht gegeben sein kann da Wahrscheinlichkeiten den Wert von 1 nicht übersteigen.

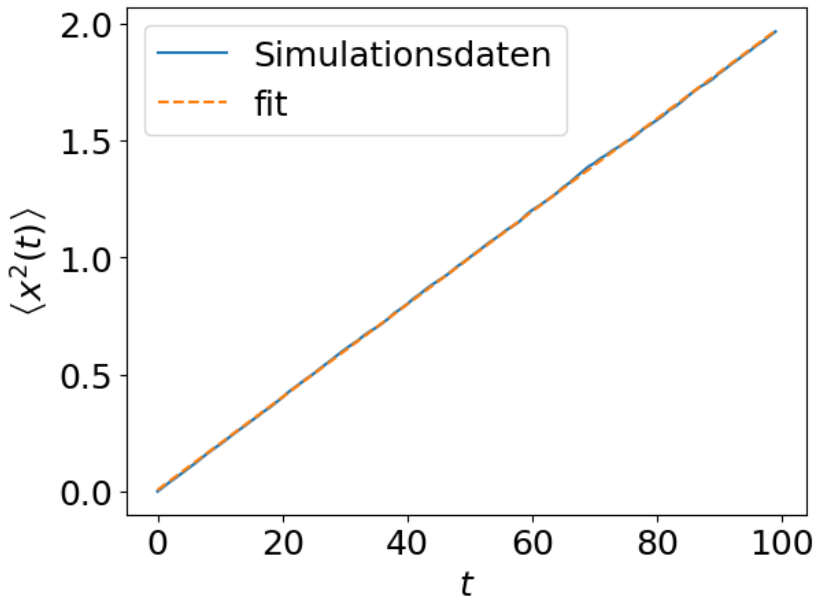


Abbildung 1: MSD eines Teilchens bei der Durchführung eines random walks mit $D_{eff} \approx 0.01$

3.2 Implementation des Lernalgorithmus

Im folgenden Problem hat ein Agent die Aufgabe, beginnend von einer festgelegten Startposition, eine Zielposition zu erreichen, auf welcher sich eine entsprechende Belohnung R befindet. Dabei

soll der Agent erlernen dies auf dem schnellstmöglichen Weg zu tun, das heißt nicht durch zufällige Bewegung durch die Umgebung sondern durch Anwenden einer Strategie. Damit der Agent dieses Problem erfolgreich bewältigen kann muss ihm zunächst das Lernen aus seinen gesammelten Erfahrungen beigebracht werden. Dazu führt der Agent im Laufe einer Episode, nach jedem Schritt eine von drei mögliche Aktionen aus und speichert die Information über diese Aktion in einer Q-Matrix (neuronales Netzwerk). Erneut liegt die Umgebung in der sich der Agent befindet und auch die möglichen Aktionen diskret vor, sodass jede Position und jede Aktion einen Index trägt. Dabei entspricht die Position dem Zustand in dem sich der Agent befindet und die möglichen Aktionen werden durch die Indizes $0 =$ nach links, $1 =$ auf der momentanen Position verbleiben und $2 =$ nach rechts gehen, gekennzeichnet.

In dieser Aufgabe wurde die Diffusionskonstante auf $D = 0.125$ gesetzt, sodass der Agent mit einer Wahrscheinlichkeit von 25% einen Diffusionsschritt durchführt. Welche Aktion anschließend ausgeführt wird (0, 1 oder 2) wird nach dem Zufallsprinzip in jedem Schritt neu entschieden. Zu beachten gilt, dass durch die Diskretisierung der Umgebung, das Intervall in dem sich der Agent bewegt begrenzt ist. Mit Hilfe von periodischen Randbedingungen wird der Agent nach Auswahl einer Aktion am jeweils anderen Ende des Intervalls platziert, sobald dieses auf einer Seite verlassen wird.

Der Eintrag Q_{ij} in Zeile i der Q-Matrix, wird nach Ausführen der Aktion j wie folgt aktualisiert

$$Q_{ij}^{neu} = Q_{ij}^{alt} + \alpha(R + \gamma \max_{j}(Q_{i'j}) - Q_{ij}^{alt} \tag{4}$$

Dabei beschreibt Q_{ij}^{neu} den aktualisierten Wert und Q_{ij}^{alt} den Wert des Eintrags vor Ausführen der Aktion j. α bezeichnet die Rate mit welcher der Agent aus den ausgeführten Aktionen lernt und der Discount factor γ stellt einen Wichtungsfaktor dar. R ist die Belohnung für das Erreichen der Zielposition und ist dementsprechend nur an dieser vorzufinden.

Dabei trifft der Agent im Laufe der Simulation entweder zufällige Entscheidungen oder basierend auf den in jener Q-Matrix gesammelten Daten, um eine nächste Aktion auszuführen. Die Wahrscheinlichkeit ε mit der der Agent eine Entscheidung unabhängig von den Informationen in der Q-Matrix trifft, nimmt mit zunehmender Episode linear ab. Ab einer bestimmten Episode ist ε fortlaufend Null, sodass die Aktionen nur noch basierend auf der Q-Matrix ausgewählt werden. Nach Durchlaufen mehreren Episoden ist der Lernprozess abgeschlossen und der Agent ist in der Lage

den schnellstmöglichen Weg zur Zielposition auszuwählen (siehe Abb.2).

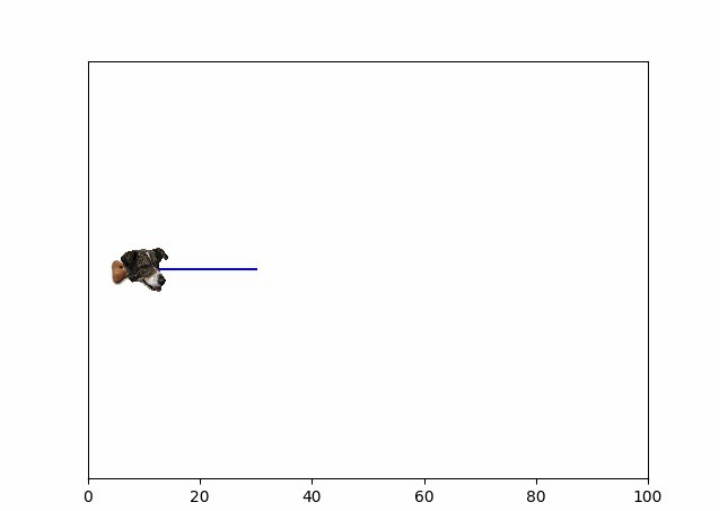


Abbildung 2: Der Hund Lasse ist auf direkten Weg an der Zielposition angekommen

3.3 Hyperparameter

Um eine möglichst effiziente Konvergenz zum Algorithmus in der Thematik des machine learnings zu erreichen, so ist die Wahl der Hyperparameter α und γ eine wichtige Entscheidung. Als sinnvoll ereignet es sich, diese Parameter möglichst strategisch zu wählen, sodass der Lernprozess letztlich nicht scheitert, sondern schnell gute Ergebnisse liefert. Als Ausführung sind folgende Kapitel eine mögliche Erläuterung, wie diese auszusehen haben.

3.3.1 Vereinfachtes Modellproblem als Beispiel

Zum Verständnis kann es hilfreich sein, sich zunächst ein vereinfachtes Modell anzuschauen. Wir nehmen an, dass sich der Agent in einer von vier Positionen (gekennzeichnet mit den Indizes 0,1,2,3) befinden kann. Zusätzlich ist nur der Wechsel in benachbarte Position als jeweilige Aktion (also nach links = Aktion 0, nach rechts = Aktion 1) möglich. Startet der Agent nun auf der Position 0 und der Reward befindet sich auf Position 3, dann endet die Epoche nur, wenn sich der Agent von Position von 2 auf 3 bewegt. Eine Bewegung von der Position 0 mit der Aktion 0 ist nicht möglich und die Position bleibt daher unverändert (Q wird dennoch aktualisiert).

Dieses Modell kann wie folgt durch Q repräsentiert werden:

$$Q_{ij}^{0} \quad = \quad \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \\ Q_{31} & Q_{32} \\ Q_{41} & Q_{42} \end{pmatrix} \quad = \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{5}$$

wobei Q_{ij}^{0} das neuronale Netzwerk zur nullten Episode angibt, die Anzahl der Spalten entsprechen der Anzahl der Aktionen, analog werden die Positionen mit den Zeilen verknüpft. Da der Agent in der nullten Episode noch nichts gelernt hat, entspricht dies einer mit null gefüllten Matrix.

Sobald wir nun die erste Episode gedanklich durchspielen, so folgt gemäß Gl.(4) die folgende Matrix:

$$Q_{ij}^{1} \quad = \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \alpha R \\ 0 & 0 \end{pmatrix} \tag{6}$$

Sobald also Q_{31} als Ereignis vorkommt, so erreicht der Agent sein Reward und für alle anderen Matrixelemente folgt zunächst weiterhin die null.

Senden wir den Agenten jetzt auf die Mission nach der Aktionsfolge mit $\rightarrow\leftarrow\rightarrow\rightarrow\rightarrow$, dann ergibt sich:

$$Q_{ij}^{2} \quad = \quad \begin{pmatrix} 0 & 0 \\ 0 & \alpha^2 R\gamma \\ 0 & \alpha R \\ 0 & 0 \end{pmatrix} \tag{7}$$

und zuletzt lassen wir den Agenten ein weiteres Mal die Aktion $\rightarrow$ durchlaufen, sodass unser neuronales Netzwerk zur dritten Episode als

$$Q_{ij}^{3} \quad = \quad \begin{pmatrix} 0 & \alpha^3 R\gamma^2 \\ 0 & \alpha^2 R\gamma \\ 0 & \alpha R \\ 0 & 0 \end{pmatrix} \tag{8}$$

gegeben ist. Wir sehen somit, dass der Agent bereits in der Aktion Q_{01} einen Eintrag besitzt und somit von der Startposition bereits das Ziel erkennen kann, da wir aus jedem Zustand heraus eine gewissen Eintrag im neuronalen Netzwerk Q_{ij} besitzen und der Agent sich anhand dieser Einträge orientiert. Somit ist er letztlich auch nicht in der Lage nach links zu laufen, sobald wir ε auf ein Minimum reduzieren, da es dann keine zufälligen Bewegungen gibt und der Agent sich immer für die Aktion 1 (Bewegung nach rechts) entscheiden wird, unabhängig von seinem Zustand.

3.3.2 Geschwindigkeit für die Konvergenz

In diesem Abschnitt werden wir uns verstärkt mit dem Hyperparamater α beschäftigen und schauen uns an, wie der Agent über die Anzahl der Episoden lernt, wo sich das Ziel befindet und wie dieser somit seine Strategie zum Ziel verfeinert. Hierbei setzen wir die Werte der Diffusionskräfte zunächst auf ein Minimum, sodass wir dort einen besseren Einblick bekommen.

Zählen wir zunächst die Schritte des Agenten zum Ziel und schauen uns an, inwiefern sich das neuronale Netzwerk letztlich dazu eignet, dass wir weniger Schritte zum Ziel benötigen (vgl. Abb. 3).

Dadurch, dass wir anhand der zufälligen Startposition bisher nicht erkennen können, ob die Strategie zum Ziel die optimal gewählte Strategie ist, müssen wir uns einen besseren Vergleich ausdenken (vgl. Abb. 4).

Zusätzlich könnten wir an dieser Stelle uns die minimalen Schritte, die von der Startposition benötigt werden können, ausgeben und mit den Schritten des Agenten vergleichen. Dabei gilt zu beachten, dass die Differenz zwischen der Startposition und der Zielposition nicht genügt, sondern

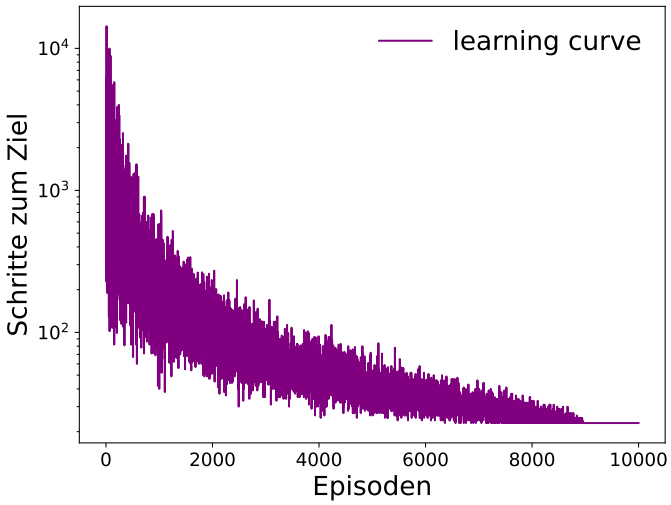


Abbildung 3: Hier werden die Schritte des Agenten über die Anzahl der Episoden dargestellt. Man erkennt einen exponentiellen Abfall bis zum Punkt, wo $episode = agent.zero_fraction$ ist. Danach bleibt der Wert konstant. Diese Art von Graphen werden als *learning curve* bezeichnet.

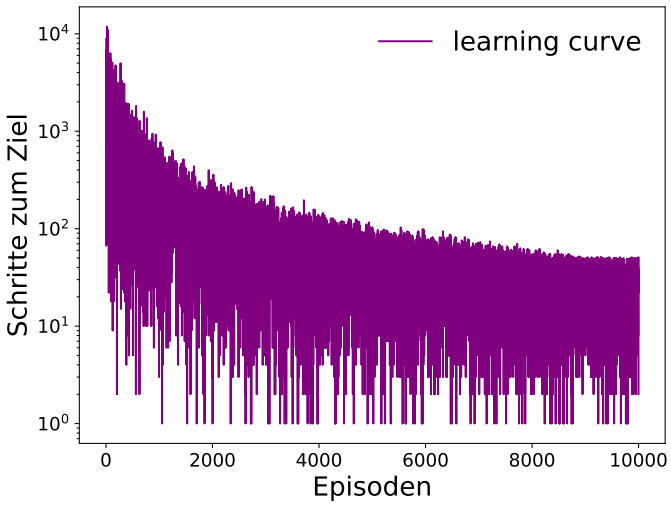
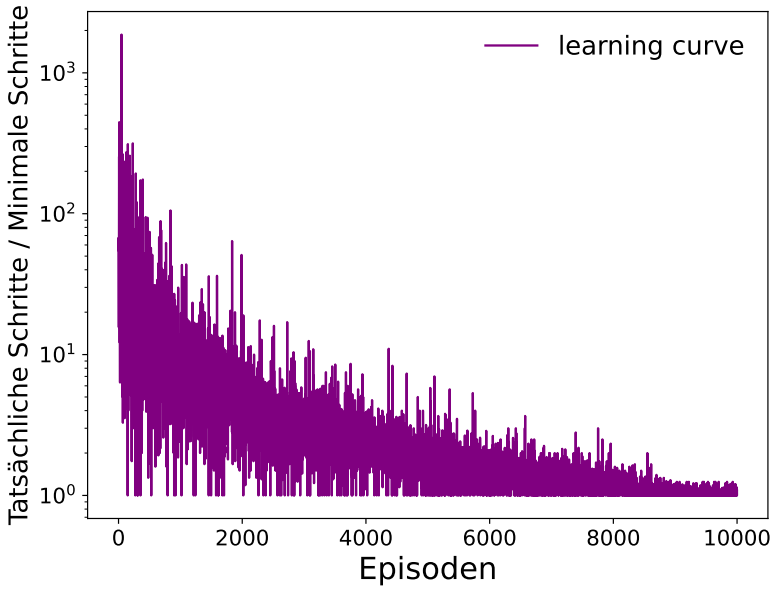
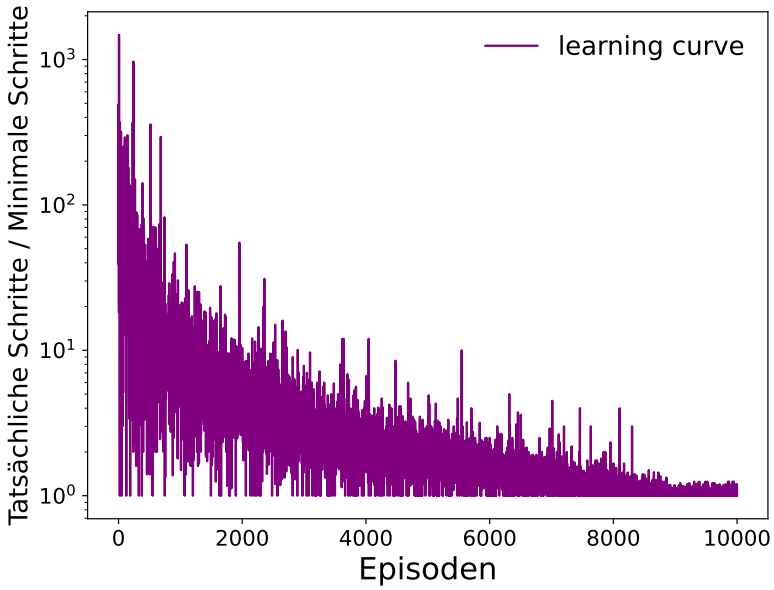


Abbildung 4: Modifikation, sodass der Agent mit jeder neuen Episode eine neue zufällige Startposition bekommt. Durch diese neue Variante verbessert der Agent zwar seine Strategie über die Anzahl der Episoden, dennoch ist dieser Wert final weiterhin stark variabel, durch die zufällige Abhängigkeit von der Entfernung zum Ziel.

wir einen weiteren Schritt addieren müssen, da der Agent auf dem Ziel einmal verweilen soll. Diese Implementation sieht wie folgt aus:



(a) *learningcurve* für $\alpha = 0.01$



(b) Im Vergleich zu Abb. 5a, wird hier der Wert α um ein Vielfaches erhöht, sodass

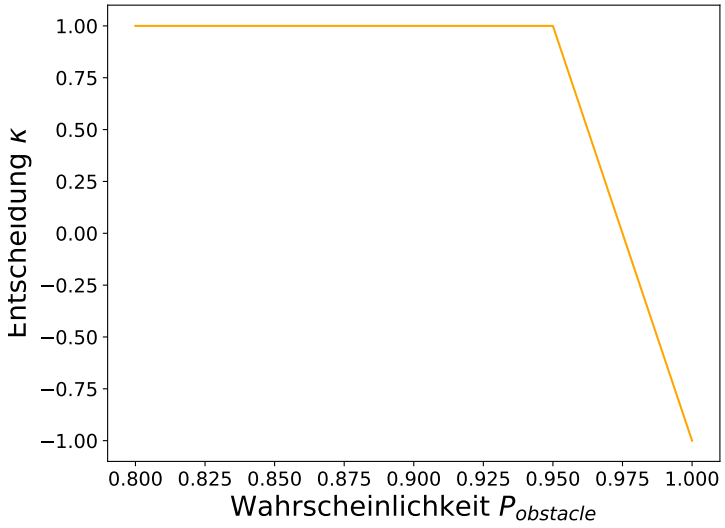
Abbildung 5: Die *learning curve* eines Agenten, wobei die tatsächlichen Schritte durch die minimalen Schritte geteilt werden und über die Anzahl der Episoden aufgetragen werden. Trotz der zufälligen Startposition wird sich dem Wert 1 angenähert, sodass wir einen optimalen Agenten zu diesem Zeitpunkt besitzen. Die Konvergenz im Fall (b) des Algorithmus sollte sich hier verschlechtern. Allerdings sind die Graphen ziemlich ähnlich zueinander.

3.4 Stochastisches Hindernis

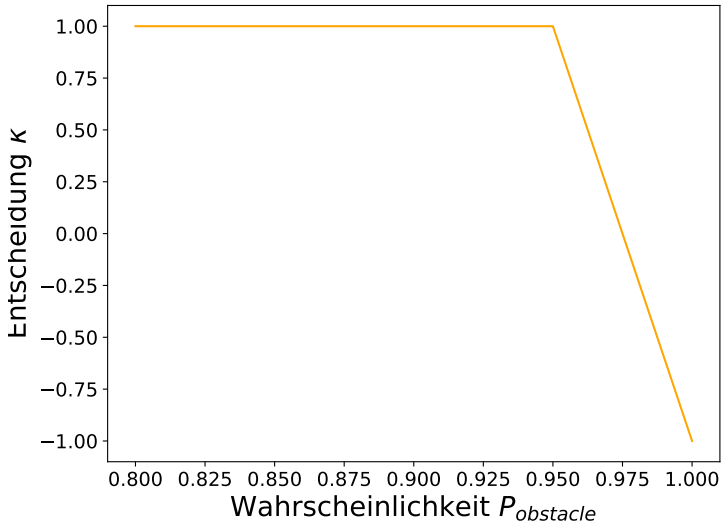
In diesem Abschnitt beschäftigen uns wir mit einem stochastischen Hindernis. Dabei versuchen wir die Lernstrategie zu untersuchen, während es nicht ganz so offensichtlich ist, welche Entscheidung als möglichst effizient gilt.

Wir definieren dafür ein Hindernis zwischen einer festgesetzten Position des Agenten und des Ziels. Nehmen wir an, dass der Agent sich im Zustand 8 befindet und das Ziel im Zustand 12, dann ist unser Hindernis im Bereich zwischen 9 und 12 definiert und besitzt eine Wahrscheinlichkeit $P_{obstacle}$ dafür, dass wenn der Agent innerhalb des Hindernisses ist, dann wird dieser einen Schritt weiter entfernt vom Ziel platziert. Eine beispielhafte Simulation für kleine Werte von α werden in Abb. 6a gezeigt.

Leider sind die Ergebnisse nicht ereignisreich, sodass daraus keine gewonnen Informationen für eine Strategie gewonnen werden können. Eher erwartet wären Ergebnisse bei einer Wahrscheinlichkeit von $P_{obstacle} \approx 0.2$, da der Agent ab diesem Zeitpunkt immer wieder mit der Wahrscheinlichkeit konfrontiert wird und der Agent ab dieser Stelle lieber den Weg nach links nehmen sollte, statt sein Glück durch das Hindernis zu erproben.



(a) Lernvorgang beim stochastischen Hindernis mit $\alpha = 0.01$.



(b) Lernvorgang beim stochastischen Hindernis mit $\alpha = 0.999999$.

Abbildung 6: Diese Graphen zeigen, welche Entscheidung κ der Agent trifft in Abhängigkeit der Wahrscheinlichkeit $P_{obstacle}$. Die Entscheidung -1 gibt an, dass der Agent den langen Weg wählt, die Entscheidung 1 ist quer durch das Hindernis hindurch. Im Hindernis gibt es dann die Wahrscheinlichkeit $P_{obstacle}$, dass der Agent einen Schritt vom Hindernis gedrückt wird.

Literatur

[1] Edward A Codling, Michael J Plank, and Simon Benhamou. Random walk models in biology. *Journal of the Royal society interface*, 5(25):813–834, 2008.

[2] Paul Monderkamp. Angeleitetes lernprojekt: Reinforcement learning für intelligente brownsche dynamik. 2022.

[3] Beverly Park Woolf. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.