Angeleitetes Lernprojekt: Reinforcement learning für intelligente Brownsche Dynamik

Hannah Dilling, Tobias Plasczyk

EINLEITUNG

In unserem FP Versuch zum Thema Reinforcement Learning haben wir uns mit dem Q-Learning Algorithmus beschäftigt. Er ist ein Werkzeug um Computern/Maschinen die Bewältigung von Aufgaben auf Basis selbst generierter Daten beizubringen.
Im Zentrum steht dabei der sogenannte „Agent“, welcher sich in einem Environment bewegt und dessen Ziel es ist, eine effiziente Strategie zu entwickeln, um einen bestimmten Ort zu erreichen. Das Environment besteht in unserem Fall aus einem eindimensionalen, diskreten Raum. Die Felder dieses Raumes tragen Indizes und der zu erreichende Ort wird durch einen konkreten Index-Wert spezifiziert.
Diese Abstraktion kann man erweitern auf physikalische Probleme, indem man vom Diskreten auf kontinuierliche Räume wechselt, um dort beispielsweise Teilchen in kolloidalen Flüssigkeiten zu beschreiben. Aspekte wie Diffusion werden dabei durch zufällige Teilchenbewegung/zufällige Aktion des Agenten ebenso berücksichtigt.
Ein Methodischer Lerneffekt dieses Versuches war der Umgang mit objektorientiertem Programmieren in Python 3.

Aufgabe 1

In der ersten Aufgabe betrachteten wir das mittlere Verschiebungsquadrat (MSD) $\langle x^2 \rangle$ um einen Zusammenhang zwischen der Diffusionskonstante und der Schrittwahrscheinlichkeit zu finden. Das Einführen einer solchen Wahrscheinlichkeit war notwendig, um den diskreten Charakter des Modells zu umgehen, der feste Werte für den Zeitschritt (τ) und die Schrittweite (a) vorsah.
Der Zusammenhang zwischen D und den genannten Größen ist dabei wie folgt:

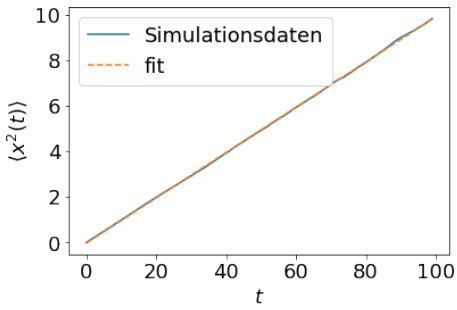
$$D := \frac{a^2}{2\tau},$$

Hier wirkt die Wahrscheinlichkeit ähnlich einer Stellschraube zwecks Variation von D, mit der der tatsächliche zeitliche Schrittabstand τ verändert wird und auf diese Weise D.
Diese Wahrscheinlichkeit P_0 lässt sich für gegebene Diffusionskonstante wie folgt berechnen:

$$P_0 = \frac{2D\Delta t}{a^2}$$

Zunächst ließen wir den Agent mit dieser Wahrscheinlichkeit einen zufälligen Schritt nach links bzw. rechts machen und ermittelt zu jedem Zeitpunkt das MSD. Im Anschluss haben wir einen linearen Fit erstellt und gemäß der Formel $\langle x^2 \rangle = 2 \cdot D_{eff} \cdot t$

die tatsächliche Diffusionskonstante D_{eff} bestimmt. Vergleicht man für drei verschiedene D die vorgegebenen Werte mit den ermittelten so findet man, dass D_{eff} tatsächlich mit D übereinstimmt. Das bedeutet, dass das Einführen einer solchen Wahrscheinlichkeit ein gutes Mittel ist, um die Diffusion in unserem diskretisierten Problem zu simulieren.



In dem Bild erkennt man eine Steigung von 10/100 = 0.1 = 2·D_{eff}.
Demnach ist D_{eff} hier 0.1/2 = .05. Dieser Wert stimmt mit dem vorgegebenen D überein.

Aufgabe 2

Nun widmeten wir uns der Implementation des Algorithmus in unser Programm, ohne dabei die Diffusion zu entfernen, sodass der Agent eine Strategie entwickelt, um das Ziel möglichst effizient zu erreichen. Zu einer gegebenen Wahrscheinlichkeit ε macht der Agent hierbei einen zufälligen Diffusionsschritt und mit der Gegenwahrscheinlichkeit (1-ε) wählt er eine Aktion basierend auf dem Lernfortschritt (s.u.). Mit wachsender Datenlage sinkt die Wahrscheinlichkeit für einen Zufallsschritt linear ab auf 0 (siehe Anhang 1).
Mögliche Aktionen des Agent sind nun je ein Schritt nach links, rechts und stehenbleiben. Sollte sich der Agent aus dem Intervall heraus bewegen, wird er auf der anderen Seite eingesetzt. Das führt zu einem periodischen Environment.
Zunächst wurde die Diffusionskonstante auf 1/8 gesetzt, um im Durchschnitt in jedem vierten Schritt einen Diffusionsschritt zu erhalten.
Weitere, nun benötigte Parameter sind die Anzahl der Episoden, die Lernrate α und der Discount-Factor γ. Eine Episode beschreibt dabei den Weg vom Startpunkt des Agent bis zum Erreichen des Ziels. Das Ziel gilt als erreicht sobald der Agent auf dem Zielfeld verweilt.
Im Kern des Lernprozesses steht die Matrix Q, welche einerseits den Lernfortschritt abbildet und andererseits die Basis für die Aktionswahl ist. Q ist eine (Anzahl Zustände) x (Anzahl Aktionen) Matrix, sodass für jede Kombination aus Zustand i und Aktion j ein Wert gespeichert werden kann. In einem festen Zustand wählt der Agent die Aktion mit dem höchsten Wert:
Im Anschluss wird der Lernfortschritt gemäß folgender Formel dokumentiert:

$$Q_{ij}^{neu} = Q_{ij}^{alt} + \alpha \left(\mathcal{R} + \gamma \max_{j}(Q_{i'j}) - Q_{ij}^{alt} \right).$$

Hierbei beschreibt α die Lernrate, also mit welcher Gewichtung der Lerneffekt in die neue Lernmatrix einfließt. R (Reward) ist die Belohnung im Falle des Erreichens des Zieles und ist in allen anderen Fällen null. γ der Discount-factor bestimmt, wie stark die Erwartung an eine Belohnung in der Zukunft die Gewichtung der Aktionen im aktuellen Zustand beeinflusst.
Noch zu erwähnen sei, dass max($Q_{i'j}$) hier den höchsten Wert der Matrix in der Zeile des Zustandes nach Ausführen der Aktion meint und nicht den Index.
Um die Ergebnisse zu evaluieren, speicherten wir die Matrix Q nach dem Durchlaufen mehrerer Episoden (in unserem Fall 10^4). Wir sahen, dass im Zustand des Zieles die Aktion „Stehenbleiben“ die bevorzugte Aktion war (also den höchsten Wert hatte) und links davon „ein Schritt nach rechts“ den größten Eintrag hatte. Auf der rechten Seite des Zieles war „ein Schritt nach links“ mit dem höchsten Wert versehen (siehe Bild).
Diese Tendenz setzte sich auf Grund der periodischen Randbedingungen fort, bis wir bei der gegenüberliegenden Seite (Abstand N/2 vom Ziel) einander angeglichene Aktionswerte fanden.

```
[ 65.61        72.9         81.          ]
[ 72.9         81.          90.          ]
[ 81.          90.         100.          ]
[ 90.         100.          90.          ]
[100.          90.          81.          ]
[ 90.          81.          72.9         ]
[ 81.          72.9         65.61        ]
```

Ausschnitt aus der Q Matrix.
Die Zeilen entsprechen den Zuständen, mit dem Ziel in der 4. Zeile. Je weiter unten die Zeile, desto weiter rechts ist der Zustand. In den Spalten findet man die Aktionen links, stehenbleiben, rechts in dieser Reihenfolge.

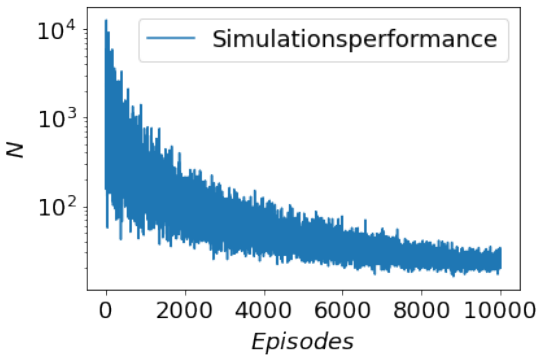
Aufgabe 3

In der nächsten Aufgabe ging es darum, sich mit den Hyperparametern auseinander zu setzen, die den Erfolg des Lernprozesses maßgeblich beeinflussen.
Für ein besseres Verständnis von der Entwicklung von Q berechneten wir die Matrix am Beispiel von vier Zuständen mit je den Aktionen ‚Schritt nach links' und ‚Schritt nach rechts'. Hier ist Q also eine 4x2-Matrix.
Um das Problem weiter zu vereinfachen, wurde der Agent immer auf Feld 0 eingesetzt, das Ziel lag immer auf Feld 3 und es gab keine periodischen Randbedingungen, sodass ein Schritt aus dem Intervall heraus als ausgeführt zählte, jedoch am Zustand nichts veränderte. Eine Episode galt als beendet, sobald das Zielfeld betreten wurde.
Zu Beginn der ersten Episode waren alle Einträge von Q Null, sodass sich bei beliebiger Schrittabfolge nur der letzte Schritt von Zustand 2 zu Zustand 3 auf die Matrix Q auswirkt (ausführliche Rechnung in Anhang 2).
Somit erhalten wir nach dem ersten Durchlauf:

$$Q_{21} = \alpha \cdot R, \text{ alle anderen } Q_{ij} = 0$$

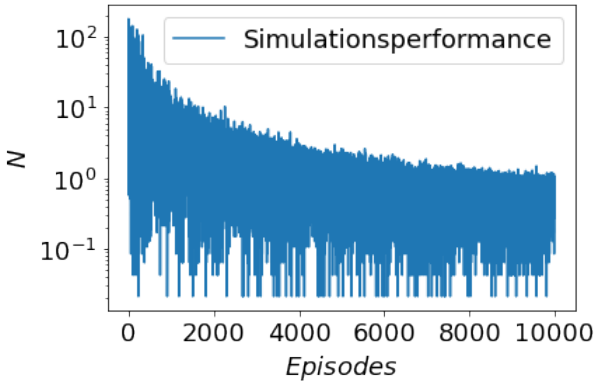
Berechnet man Q für die zweite und dritte Epoche, so erkennt man, dass sich das Wissen um das Ziel auf Feld 3 mit jeder Epoche um ein Feld ausbreitet. Der Wert, welcher in Q gespeichert wird, hängt dabei von α und γ ab, sowie linear von R. Dabei steigt mit jedem Durchlauf die Potenz von α, während die Potenz von γ pro Zustand konstant bleibt und mit jedem Schritt vom Ziel weg um eins steigt.
Betrachtet man das nun errechnete Q, so erkennt man, dass der Agent in diesem Beispiel nicht lernen kann, nach links zu gehen, sofern die Parameter entsprechend gewählt werden, dass kein Eintrag von Q negativ wird.
Typischerweise wird γ zwischen 0,5 und 1,0 gewählt, sodass die Belohnung keinen zu starken Einfluss nimmt, jedoch immer noch genug, um eine Entscheidung in Richtung des Ziels bei weiter entfernten Zuständen zu bewirken. Dies spielt aber erst bei mehreren Belohnungen oder unterschiedlichen Aktionen eine signifikante Rolle.

Der zweite Teil der Aufgabe widmete sich der Lernrate α.
Hierbei wurde das Problem aus Aufgabe 2 ohne Diffusion betrachtet. Lässt man die Simulation über mehrere Episoden mit gleichem Startfeld laufen und vergleicht die Schrittzahl pro Episode, so erkennt man, dass diese etwa exponentiell abnimmt.



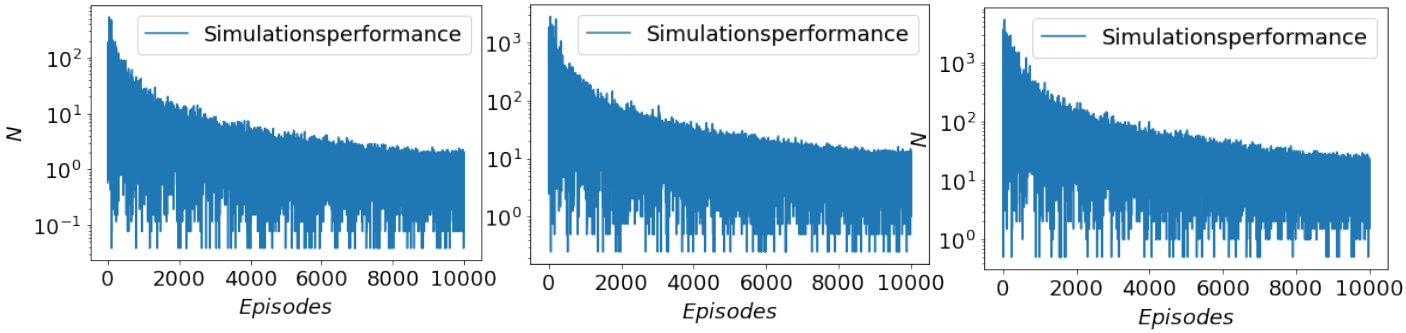
Learning curve bei festem Startwert ohne Diffusion

Lässt man den Agent bei einer zufälligen Position starten, so erkennt man, dass die Anzahl der Schritte pro Episode stärker schwankt, da die Distanz zum Ziel variiert (siehe z.B. „Aufgabe_3.3.2_Plasczyk_Tobias_random_start_22-08-04T17_20_48.png").
Sinnvoller ist es hier, in jeder Episode das Verhältnis der tatsächlichen Schritte zu der Mindestanzahl an Schritten abhängig vom Startort zu betrachten.



Learning Curve bei zufälligem Startwert und Verhältnis N aus tatsächlichen Schritten und Minimaler Schrittzahl, konvergiert gegen 1
Aufgabe_3.3.2_Plasczyk_Tobias_random_start_22-08-04T17_37_52.png

Variiert man nun die Learning rate, so erkennt man, dass ein höheres *α* eine schlechtere Konvergenz ergibt. Die Schwankung, die durch die zufällige Startposition erzeugt wird, bleibt erhalten, während der ausgelernte Agent mit steigendem *α* mehr Schritte in der letzten Episode machen muss.

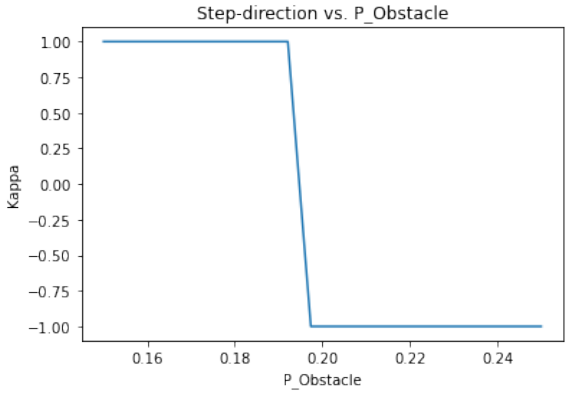


Learning Curve mit α=0.05; α=0.1 und α=0.999999

Der Nachteil von sehr kleinen Lernraten ist jedoch, dass sich die Laufzeit erhöht. Daher muss dieser Parameter vorher sorgfältig abgewogen werden. Ein Hilfsmittel hierbei ist, *α* ähnlich zu ε mit steigender Episodenzahl abfallen zu lassen.

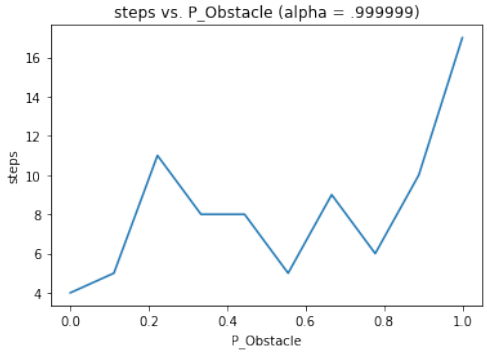
Aufgabe 4

Zuletzt wurde das Problem etwas komplexer, indem ein stochastisches Hindernis als Widerstand gegen die Bewegung des Agent eingeführt wurde.
Wenn der Agent sich in einem der Hindernisfelder befand, wurde er mit einer Wahrscheinlichkeit von *P_obstacle* nach links verschoben. Es wurden in jeder Episode wieder ein fester Start und ein festes Ziel rechts davon gewählt.
Betrachtet wurde in jeder Episode die Richtung κ der Bewegung des Agent, indem die gesamte Verschiebung des Agent im Vergleich zur Startposition durch die gesamte Anzahl an Schritte geteilt wurde. Ist das Ergebnis -1, so geht der Agent nach dem Lernprozess nach links, ist es +1, so geht er nach rechts.
Wir vermuten einen Umschlag der Richtung, wenn der Erwartungswert der Schrittzahl beim Gehen durch das Hindernis größer bzw. gleich der Schrittanzahl nach links (hier 11+1 Verweilen) ist.



Numerisches Ergebnis:
Ab einer Hinderniswahrscheinlichkeit von 0.2 bevorzugt der Agent den Weg außen herum. Bei P_obstacle < 0.2 nimmt er den Weg hindurch.

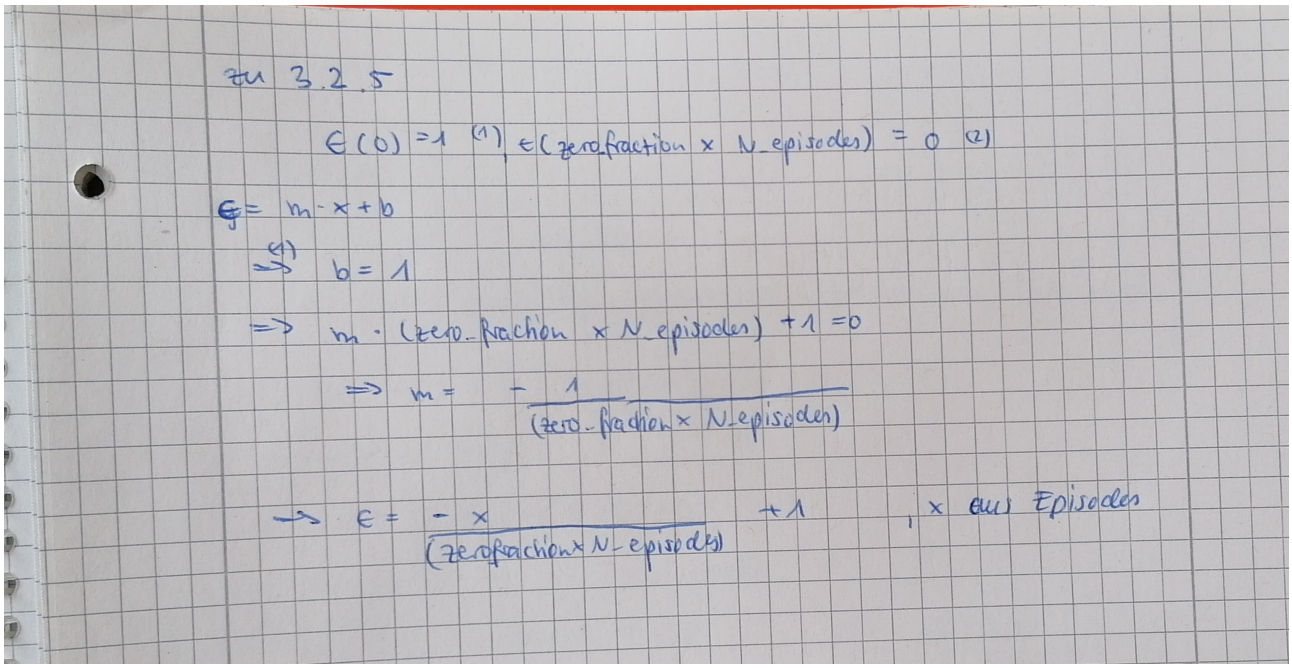
Setzt man nun die Lernrate *α* wieder auf 0.999999, so erkennt man (Bild unten), dass der Algorithmus gegen eine schlechte Strategie konvergiert. Bei einer Rückstoßwahrscheinlichkeit von 1 weiß der Agent nicht, dass er linksherum schneller am Ziel ankommt und benötigt ca. 16 statt der 12 Schritte auf dem kürzesten Weg. Er scheint also trotzdem in das Hindernis hineinzulaufen.



Wir haben gelernt, dass die Mechanismen des Machine Learning durchaus mächtig sind und viele Stellschrauben in Form von Parametern enthalten. Diese Parameter wollen aber mit Bedacht gewählt sein, da diese die Laufzeit und das Ergebnis mitunter stark beeinflussen können und damit auch zu schlechten, ineffizienten Strategien führen können.

Anhang

Anhang 1:



Anhang 2:

$$Q_{ij}^{neu} = Q_{ij}^{alt} + \alpha \left(R + \max_j (Q_{i'j}) \cdot \gamma - Q_{ij}^{alt} \right)$$

1. Episode:

Start $i=0$, alle $Q_{ij}=0$

Bei Schritt nach links ändert sich nichts, da alle Werte $Q_{ij}=0$.

$i=0 \rightarrow i=1 \Rightarrow j=1$

$Q_{01} = 0 + \alpha \cdot (0-0) = 0$ (↳ alle $Q_{ij}=0$)

$i=1 \rightarrow i=2 \Rightarrow j=1$

$Q_{11} = 0 + \alpha \cdot (0-0) = 0$

$i=2 \rightarrow i=3 \Rightarrow j=1$, Ziel erreicht

$Q_{21} = 0 + \alpha \cdot (R+0) = \alpha R$

$$\Rightarrow Q^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \alpha R \\ 0 & 0 \end{pmatrix}$$

2. Episode ① ② ③ ④ ⑤

① $i=0 \quad i'=1 \quad j=1$

$Q_{01}^{neu} = Q_{01}^{alt} + \alpha (0 + \gamma \overbrace{\max_j (Q_{1j})}^{0} - Q_{01}^{alt})$

$= 0 + \alpha (0) = 0$

② $i=1 \quad i'=0 \quad j=0$

$Q_{10} = Q_{10}^{alt} + \alpha (\gamma \max_j (Q_{0j}) - Q_{10}^{alt}) = 0$

③ siehe ①

④ $i=1 \quad i'=2 \quad j=1$

$Q_{11}^{neu} = Q_{11}^{alt} + \alpha (0 + \gamma \max_j (Q_{2j}) - Q_{11}^{alt})$

$= 0 + \alpha \cdot \gamma \cdot \alpha R = \gamma \alpha^2 R$

⑤ $i=2 \quad i'=3 \quad j=1$, Ziel erreicht

$Q_{21}^{neu} = Q_{21}^{alt} + \alpha (R + \gamma \max_j (Q_{3j}) - Q_{21}^{alt})$

$= \alpha R + \alpha R - \alpha^2 R = 2\alpha R - \alpha^2 R$

$$\Rightarrow Q^{(2)} = \begin{pmatrix} 0 & 0 \\ 0 & \gamma\alpha^2 R \\ 0 & 2\alpha R - \alpha^2 R \\ 0 & 0 \end{pmatrix}$$

3. Episode →

$i=0 \quad i'=1 \quad j=1$

$Q_{01} = Q_{01}^{alt} + \alpha (0 + \gamma \max_j (Q_{1j}) - Q_{01}^{alt})$

$= \alpha \cdot \gamma \cdot \gamma \alpha^2 R = \alpha^3 \gamma^2 R$

→ „Wissen" um das Ziel breitet sich mit jeder Episode um ein Feld aus