Evaluation der Performanz von Algorithmen im Reinforcement Learning

Niklas Wenzel

Hochschule für Angewandte Wissenschaften, Hamburg, Deutschland
niklas.wenzel@haw-hamburg.de

Zusammenfassung. Verglichen werden die drei Reinforment Lernening Verfahren Cross-Entropy-Method, REINFORCE und Advantage Actor Critic. Für die beiden Environments CartPole und Breakout sollen Policys erlernt werden und an Hand der Returns dieser Policys werden die Verfahren miteinander verglichen. In CartPole konnten CEM und A2C mit GAE konstant sehr schnell das Maximum erreichen, in der Regel benötigten beide Verfahren nur 10 - 50 Episoden. Das Verfahren REINFORCE war hingegen sehr instabil. Für das Spiel Breakout konnte keines der Verfahren eine gute Policy erlernen. Die Verfahren REINFORCE, A2C mit one-Step Return und A2C mit GAE konnten im Durchschnitt nur einen Return von 1.6, 2.1 und 2.4 erreichen. A2C mit GAE erreichte zwar den höchsten Reward, aber auch dieses Verfahren konnte keine Policy erlenen, die konstant mehr als vier Punkte erzielen kann. Mittels des Spiels Pong konnte gezeigt werden, dass die Verfahren grundlegend in der Lage sind solche Spiele zu spielen, der Observationspace von Breakout jedoch in dieser Implementation für alle Verfahren zu komplex und somit die Varianz trotz eigentlich geeigneter Methoden zu hoch ist.

Schlüsselwörter: Reinforment Learning· OpenAI-Gym · Cross-Entropy-Method · REINFORCE · Advantage Actor-Critic · Generalized Advantage Estimation

1 Einführung

Das Maschinelle Lernen (ML) teilt sich in die drei Bereiche Überwachtes Lernen, Unüberwachtes Lernen und Verstärkendes Lernen, auch Reinfocement Learning (RL). RL grenzt sich dabei von den anderen Bereichen ab. Beim überwachten Lernen wird ein Problem mittels gelabelter Trainingsdaten trainiert, das unüberwachte Lernen ziel darauf ab Strukturen innerhalb der Daten zu erkennen und zu rekonstruieren. Beim verstärkenden Lernen hingegen wird ein Agent beobachtet, der mit seiner Umgebung interagiert und für die Aktionen Belohnungen oder Bestrafungen erhält. Bei dieser Betrachtung der Interaktion des Agenten handelt dieser meist Sequenziell und erhält die Rückmeldung entweder direkt nach einer Aktion, nach mehreren ausgeführten Aktionen oder am Ende einer Episode. Das Ziel ist des Agenten ist es, die Belohnungen zu maximieren,

sodass dessen Verhalten maßgeblich durch die Belohnungsfunktion beeinflusst wird. Dabei ist es wichtig, dass eine Aktion nicht nur die sofortige Belohnung beeinflusst, sondern auch zukünftige Belohnungen, die aus diesem Folgezustand resultieren. Zu Beginn handelt der Agent nach dem Verfahren trial-and-error, er probiert sich aus, um eine möglichst große Zahl Zustände abzudecken und zu explorieren und versucht dann aus diesen Erkenntnisse in Zukunft schlauer handeln zu können. Formell betrachtet sucht der Agent die optimale Lösung eines Markov-Decision-Process (MDP), der ihm unbekannt ist. Diese definieren lediglich einen erfassbaren Status, Aktionen und ein Ziel, das erreicht werden soll. Auf diesen drei Aspekten baut der Agent auf und versucht den Prozess zu erlernen. [4] Der Agent steht dabei immer zwischen den beiden Aspekten Exploitation und Exploration, die Ausnutzung des bisherigen Wissens durch die Policy und die Erkundung neuer Punkte im MDP. Zentrale Rolle im Verhalten spielt die Policy, die aus den Erfahrungen gelernt wird. Diese beschreibt auf Basis des zu erwartenden Rewards und gibt bei der Wahl einzelner Aktionen somit an, wie sich der Agent verhalten soll. Durch Evaluation und Optimierung der Policy wird das Verhalten des Agenten mit der Zeit besser und ist in der Lage das Ziel zu erreichen. Es gibt verschiedene Optimierungsverfahren oder auch Policy Gradient Algorithmen die diese Policy auf verschiedene Weise anpassen und optimieren. In dieser Arbeit sollen die drei Verfahren Cross-Entropy, REINFORCE und Advantage-Actor-Critic (A2C) im Bezug auf die Performance evaluiert und miteinander verglichen werden.

2 Grundlagen

Bei den vorgestellten Verfahren handelt es sich um Policy Gradient Methoden. Klassische Verfahren lernen die Action-Value-Function eines State und entscheiden nach dieser Action-Value-Schätzung, welche Aktion gewählt werden soll. Policy Gradient Methoden lernen hingegen direkt die parametrisierte Policy, die die Aktion ohne die Verwendung einer Value-Function vorgeben. [3] Es gibt Verfahren, die weiterhin eine Value-Function lernen, beispielsweise sind dies Actor-Critic-Methoden, diese wird jedoch nicht für die Auswahl der Action benötigt. Stattdessen beschreibt die Policy eine Wahrscheinlichkeitsverteilung für die Wahl der jeweiligen Aktion. Die Policy wird durch einen parametrisierten Vektor θ ergänzt, so ergibt sich die Schreibweise $\pi(a|s,\theta) = Pr\{A_t = a|S_t = s, \theta_t = \theta\}$ für die Wahrscheinlichkeit eine Aktion zum Zeitpunkt t unter Beachtung jeweiligen States und des Vektors. Beim Lernen der Methoden wird der Vektor angepasst und versucht den zu erzielenden Reward zu maximieren. Die Anpassung des Vektors erfolgt dabei nach folgender Updateregel, die den Vektor um den Gradienten anpasst:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta J(\theta_t) \tag{1}$$

Da die optimale Policy durch eine Wahrscheinlichkeitsverteilung angenähert wird spricht man auch von einer Stochastic Policy Estimation. [4]
Bei den drei Verfahren handelt es sich um Beispiele für Online-Reinforcement

Learning. Das bedeutet, dass die Verfahren während des Trainings die nach der eigenen Policy interagieren und diese während des Trainings verändern. Beim Offline-Lernen hingegen würde die Policy auf Basis der Erfahrungen einer anderen Policy trainiert werden. In dem Fall würde die Policy das Verhalten des anderen Agenten beobachten und daraus Schlüsse für das eigene Verhalten ziehen. Dieses Verfahren bietet sich beispielsweise beim Trainieren sicherheitsrelevanter Anwendungen an, so zum Beispiel dem Trainieren eines autonomen Autos. Der Agent würde einen Fahrer beobachten um grundlegende Fähigkeiten zu erlangen und so das Risiko eines Unfalls zu reduzieren. Besonders wenn keine Simulationsumgebung zur Verfügung steht oder die Environment sehr komplex ist bietet sich Off-Policy-Lernen an. Beim On-Policy-Lernen hingegen ist es wahrscheinlich, dass der Agent deutlich mehr Sonderfälle entdeckt und so ein robusteres Verhalten entwickelt, da die optimale Policy, die als Basis des Offline-Lernens dient, eventuell nicht alle States abdeckt und so Handlunganweisungen für diese fehlen. [4]
Policy Gradient Verfahren haben zudem zwei Probleme. Zum einen müssen große Datenmengen erzeugt werden, um möglichst viele States des MDPs abzudecken und zum anderen muss eine ständige Verbesserung erzielt werden, auch wenn die eingehenden Daten stark variieren. Diese Probleme werden besonders bei komplexen MDPs deutlich. Gelöst werden sie durch die Verwendung einer Schätzung der Value-Function, welche die Varianz in den Daten reduziert und ein stabiles Training ermöglicht. Zudem werden durch den Einsatz von Neuronalen Netzen zu große Einflüsse neuer Daten reduziert, da diese zwar in das Training einfließen aber die Policy nur beeinflussen und nicht direkt eine neue Policy angenommen wird. In dieser Arbeit wird besonders das erste Problem betrachtet, da die verwendeten Verfahren verschiedene Ansätze verwenden die Varianz zu verringern. [10]

2.1 Cross-Entropy-Method

Bei der Cross-Entropy-Methode handelt es sich um ein sogenanntes Monte-Carlo-Verfahren. Das bedeutet, dass der Agent bis zum Ende einer Episode nach der gleichen Policy handelt und diese dann mit der gesamten Trajektorie der Episode verbessert wird. Die Trajektorie umfasst dabei das Tupel $\{state, action, reward\}$ für jeden Schritt in der Episode. Die Cross-Entropy-Methode ist dabei eine evolutionäre Methode, die mehrere Trajektorien sammelt und lediglich mit den besten weiterarbeitet. Die grundlegende Idee ist, verschiedene Agenten gegeneinander antreten zu lassen, deren Policy anhand ihres erreichten Rewards zu evaluieren und die Gewichte der besten Agenten mit einer stochastischen Streuung auf die schlechteren Agenten zu übertragen. Diese Idee ist an biologische Verfahren angenähert und gleicht dem Sprichwort „Survival of the fittest". Dadurch konvergiert die Policy [1]
Eine Variante der CEM mit mehreren Agenten beschränkt sich auf die Verwendung eines einzelnen Agenten. Auf diese Weise ist es nicht nötig die Gewichtsvektoren mehrerer Agenten anzupassen. In diesem Fall generiert ein Agent mehrere

Trajektorien mit der gleichen Policy. Im zweiten Schritt werden diese Trajektorien nach ihrem erreichten Reward geordnet und der eine Agent lernt aus den besten dieser Trajektorien, in dem er die gewählten Aktionen analysiert und sich daran anpasst. Das Verfahren basiert lediglich auf den gewählten Aktionen als One-Hot Encoded Vektor und beachtet dabei nicht den erzielten Reward. Dadurch ist das Verfahren sehr simpel zu implementieren und das Lernen der Policy sehr einfach. [5]

2.2 REINFORCE mit Baseline

Genau wie CEM handelt es sich auch bei dem Verfahren REINFORCE um eine Monte-Carlo-Methode. Der Agent folgt ebenfalls seiner Policy bis zum Ende einer Episode und verwendet dann die Trajektorie zum Verbessern dieser. Im Gegensatz zur Cross-Entropy-Method verwendetdas Verfahren REINFORCE allerdings den Reward zum Verbessern seiner Policy, weshalb auch mit nicht idealen Trajektorien gelernt werden kann. Auf diese Weise werden mehr States traversiert und die Policy robuster, benötigt aber auch mehr Zeit um zur optimalen Policy zu konvergieren. Das Verfahren generiert ebenfalls einen One-Hot Encoded Vektor aus den Informationen der Trajektorie, multipliziert diesen aber zusätzlich mit dem in diesem State erhaltenen Reward. Auf diese Weise werden gute Aktionen in den States verbessert und schlechte Aktionen bestraft. Deshalb ist es nicht nötig, anders als bei CEM, nur der aktuell besten Trajektorie zu Folgen. [3, 5] Ein großes Problem, dass Monte-Carlo-Verfahren haben, ist die hohe Varianz, der sie unterliegen, da nur ganze Episoden betrachtet werden und viele unterschiedliche States und Verläufe gewählt werden können. Ein beliebter Weg, diese Varianz zu verringern, ist die Subtraktion einer Baseline. REINFORCE kann sowohl mit als auch ohne Baseline verwendet werden. Diese normalisiert die Rewards, dass bedeutet, sie sind mittelwertbefreit und standartisiert. Die Differenz ($A_t = G_t - b(s_t)$) wird auch als als Advantage bezeichnet. [3]
Das Update der Parameter erfolgt bei REINFORCE über folgenden Term:

$$Y = (1 - \alpha * R_t) * p_t + \alpha * R_t * \mathbb{I}_{a_t} \tag{2}$$

Dieser leitet sich aus der Updateregel 1 ab. Die Komponente p_t beschreibt die Wahrscheinlichkeitsverteilung der aktuellen Policy, wodurch der linke Term den bestehenden Parametervektor θ steht. Zu diesem wird der Rechte Term, die Veränderung des Vektor in diesem State addiert. Durch das Alpha kann die Lerngeschwindigkeit reguliert werden, je größer das α gewählt wird, desto größer ist der Einfluss des neuen Terms. [5]

2.3 Advantage Actor Critic

Im Gegensatz zu den anderen beiden Verfahren handelt es sich bei Advantage Actor Critic um ein sogenanntes TD(λ)-Verfahren oder auch n-Step TD Verfahren, welches somit Bootstrapping verwendet. Das bedeutet, dass die Policy nach jedem einzelnen Schritt angepasst wird und nicht bis zum Ende der Episode

gewartet wird. Zudem werden Aspekte von Monte-Carlo-Verfahren mitverwendet, durch die Verwendung eines großen λ ähnelt der diskontierte Reward dem tatsächlich erreichten Return eines Monte-Carlo-Verfahrens. Auf diese Weise ist es möglich mittels einer Generalized Advantage Estimation ein Monte-Carlo-verfahren anzunähern und dies in jedem Update und Step anwenden zu können. Durch den n-Step Return können die Varianz und Streuung durch das Einbeziehen weiterer Werte deutlich gesenkt und die Lerngeschwindigkeit erhöht werden. Durch den n-Step Return wird die Value-Funktion geschätzt. Die weitere Besonderheit des Verfahrens ist die Definition von zwei Netzwerken. Wie bereits erwähnt lernen Actor-Critic-Verfahren neben der Policy die Value-Function, um die Ergebnisse der Policy beim Bootstrapping evaluieren zu können. Dies ist nötig, da anders als bei Monte-Carlo-Methoden, der finale Reward noch nicht bekannt ist und somit geschätzt werden muss. Dieses Netz wird als Critic bezeichnet. Dieser schätzt auf Basis des aktuellen States die State-Value-Function um, ähnlich wie zur Baseline, den Advantage berechnen zu können. Mit diesem Advantage wird zeitgleich die Policy vom Actor-Netz erlernt, welches die Aktionen auswählt und vorgibt. [8] Der Advantage für $\lambda = 1$ berechnet sich aus dem aktuellen Reward und den gewichteten Schätzungen der State-Value-Functions des Critics:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) = r_t + \gamma * V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{3}$$

Als Advantage definiert wird die Differenz aus State-Value-Function und Action-Value-Function. Beide geben der zu erwartenden Reward bis zum Ende der Trajektorie an, wenn man konsequent der Policy folgt, jedoch einmal generell in diesem State und einmal bei der Wahl einer bestimmten Aktion. Durch die Subtraktion beider Werte kann der Vorteil der gewählten Aktion, gegenüber dem Gesamtreward, in diesem State, bestimmt werden. Bei der Action-Value-Function $Q^{\pi}(s_t, a_t) = r_t + \gamma * V^{\pi}(s_{t+1})$ handelt es sich um das TD-Target, dass zu optimierende Ziel des Actors, basierend State-Value-Schätzung des Critic. Auf diese Weise kann nach einem Schritt im MDP diese Aktion evaluiert werden und der Actor kann somit die optimale Policy erlernen. [5, 9]
Für Advantage Actor Critic können jedoch veschiedene Ansätz gewählt werden, diesen Advantage zu berechnen. Bereits vorgestellt wurde für $\lambda = 1$ die Schätzung des Advantage durch die Action- und State-Value-Function mittels eines one-Step TD-Verfahrens. Wird für die Schätzung allerdings ein n-Step TD-Error verwendet, spricht man von einer Generalized Advantage Estimation (GAE). Diese entspricht einer diskontierten Summe des one-Step TD-Error für die folgenden States, weshalb diese Verfahren äquivalent verwendet werden können. Für GAE ergibt sich mit dem λ ein weiterer Hyperparameter, der maßgeblich den Erfolg des Trainings und die Schätzung des Advantages beeinflussen kann. Je größer dieses gewählt wird, desto größer wird der Einfluss der Folgezustände. Auf diese Weise können die beiden Effekte Bias und Varianz beeinflusst werden. Ein Hohes λ führt zu einer starken Gewichtung der Zukunft und somit zu einer kleinen Streuung aber großen Varianz. Ein kleines λ hingegen hat einen höheren Bias aber kleinere Varianz. [9] Durch die Verwendung von GAE

kann die Varianz der Eingangsdaten weiter reduziert werden, da durch den n-step-Return weitere Returns miteinbezogen werden und so die Value-Function einzelner States bessert abgeschätzt und angepasst werden kann. Auf diese Weise kann besonders das Training des Critic und daraus folgend auch des Actors verbessert werden. [10]

3 Versuch

Für die Experimente mit den drei Verfahren werden zwei verschiedene Environments verwendet, in denen sich die Agenten bewegen. Auf Grund der einfachen Bedienung wird mit dem Framework OpenAI-Gym gearbeitet. Dieses biete die Möglichkeit solche Environments zu erstellen oder zu verwenden und dann mit diesen zu interagieren. Bei der ersten Environment handelt es sich um das Cart-Pole, ein Schlitten, der nach Links oder Rechts bewegt werden kann, mit dem Ziel, ein Pendel senkrecht zu balancieren. Der State beschreibt die vier Parameter $\{CartPosition, CartVelocity, PoleAngle, PoleAngularVelocity\}$ und die Aktionen umfassen die Bewegung des Carts nach $\{Links, Rechts\}$. [7]
Als zweites wird das Atarispiel Breakout trainiert. Ziel des Spiels ist, mit dem Schlitten den Ball im Spiel zu halten und damit die Blöcke oberhalb zu zerstören. Anders als beim CartPole wird der State hier nicht durch Position und Winkelgeschwindigkeit des Pendels, sondern durch ein Bild des aktuellen Spiels beschrieben. Die Aktionen beziehen sich auf die Bewegung des unteren Balken: $\{NoOperation, Fire, Right, Left\}$, wobei die Aktionen „No Operation"den Stillstand und „Fire"den Neustart des Spiels nach einer Pause bedeuten. Auf diese Weise ist der Stateraum deutlich komplexer und auch schwieriger zu erlernen. Zusätzlich bietet sich in dieser Umgebung das Problem, dass der Reward stark verzögert ist und nur selten gerenell ein Reward erzielt wird. Dies ist beim Cartpole anders, hier erhält der Agent nach jedem Schritt eine Rückmeldung seiner Umwelt. Aus diesem Grund wird erwartet, dass besonders A2C mit GAE, also einem n-Step Return besser abschneidet, da dies verzögerte Rewards besser in den einzelnen States verarbeiten kann. [7]
Im vorherigen Kapitel wurden die drei Verfahren, die in dieser Arbeit implementiert werden, vorgestellt. Als Policy Gradient Methoden lernen alle drei Verfahren einen Parametervektor, der im Training angepasst wird, um den Reward zu maximieren. In diesem Anwendungsfall wird die Policy durch ein Neuronales Netz erlernt, welches als Eingangsdaten den jeweiligen State erhält. Die Gewichte innerhalb des Netzes entsprechen hierbei dem Parametervektor, der in Policy Gradient Methoden erlernt wird. Im Verlauf des Trainings sind diese Gewichte die veränderbaren Parameter, die sich an die optimale Policy anpassen. Auf diese Weise können Neuronale Netze auf das Reinforcement Learning angewendet und das automatische Lernen dieser genutzt werden, ohne die Gewichte selber anpassen zu müssen. [5]
Während für die Verarbeitung der Informationen ein Neuronales Netz mit Dense-Layern verwendet werden kann, sind für die Verarbeitung des Bildinputs bei Breakout Faltungsnetze nötig. Die drei Verfahren erhalten jeweils die selbe Netz-

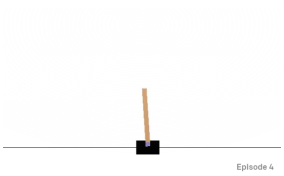

Abb. 1. Environment CartPole [7]

Abb. 2. Environment Breakout [7]

architektur zum erlernen der Policy, um diese Vergleichbar zu halten. Lediglich bei A2C wird ein weiteres Netz, mit der selben Achirektur, aber einem Einzeloutput, zur Vorhersage der Value-Function verwendet, Für eine effizinetere Verarbeitung der Eingagsbilder werden diese von {210 * 160 * 3} Pixel auf {80 * 80 * 1} Pixel reduziert. Dadurch gehen keine relevanten Bildinformationen verloren, das Training des Neuronalen Netzes wird aber deutlich effizienter.

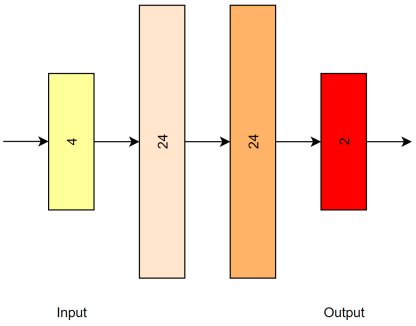

Abb. 3. Netzaufbau CartPole

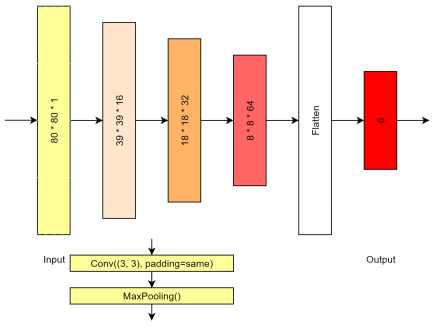

Abb. 4. Netzaufbau Breakout

4 Ergebnisse

4.1 CartPole

In der CartPole-Environment erhält der Agent für jeden Schritt, den das Pendel nicht eine bestimmte Gradzahl oder Position überschritten hat, einen Reward von 1. In der verwendeten Version gilt eine Episode sowohl als Beendet, wenn das Pendel umgefallen ist oder die maximale Anzahl von 200 Schritten balanciert wurde. In den folgenden Abbildungen sind die Rewards der einzelnen Verfahren in den Trainingsdurchläufen aufgeführt. Aus den Graphen wird deutlich, dass für das CartPole die Cross-Entropy-Methode mit am schnellsten gegen den maximalen Reward konvergiert und den geringsten Verlust des Trainings in späteren Episoden aufweist. Im Verlauf des durchschnittlichen Rewards (Blau) sind zwar Einbrüche zu erkennen, allerdings erreicht eine Trajektorie fast immer das Maximum, weshalb diese keine Effekte auf den Trainingsverlauf haben. Bereits nach 25 bis 50 somit erreicht das Verfahren das Maximum.

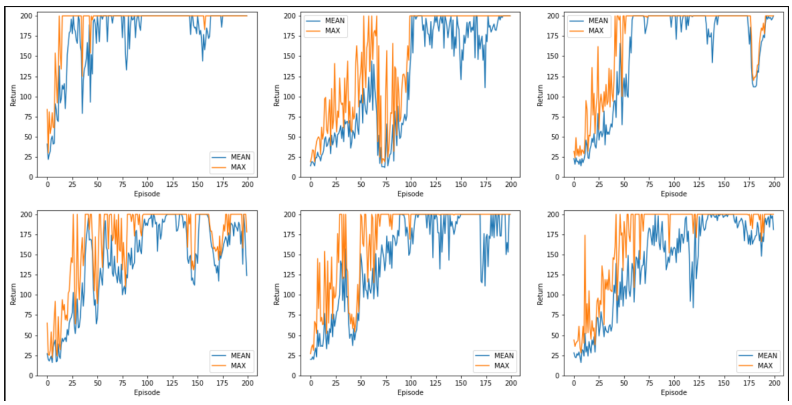

Abb. 5. Erzielter Reward Cross-Entropy-Method

Deutlich länger für diese Konvergenz benötigt das Verfahren REINFORCE. Im Mittel werden hier 100 Episoden benötigt, bis das Verfahren konvergiert, es kann aber auch variieren. Anders als bei der Cross-Entropy-Method sind zudem im späteren Verlauf des Training immer wieder Verluste des gesamten bisherigen Lernerfolgs zu erkennen. Dies kann durch eine Reduktion der Lerngeschwindigkeit, also eine Verkleinerung des α im Update-Term, behoben werden. Durch Reduktion des α lernt der Algorithmus nicht mehr aus den neuen Erfahrungen, erreicht also sehr konstant den bereits erlernten Bestwert, ist im Gegenzug aber auch nicht mehr in der Lage gegen ein schlechtes Training gegenzusteuern und zu einem späteren Zeitpunkt die Policy zu erlernen. Die Graphen zeigen aber für den Fall, dass die Policy gut approximiert wurde, dass der maximale Reward konstant erreicht wird.

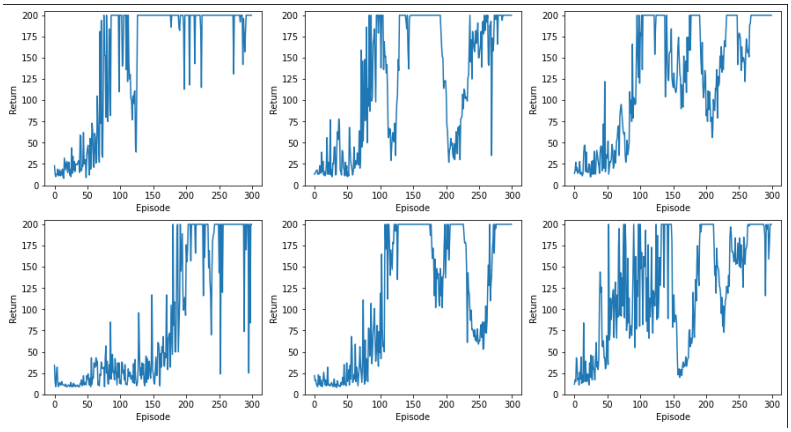

Abb. 6. Erzielter Reward REINFORCE ohne linear α-decay

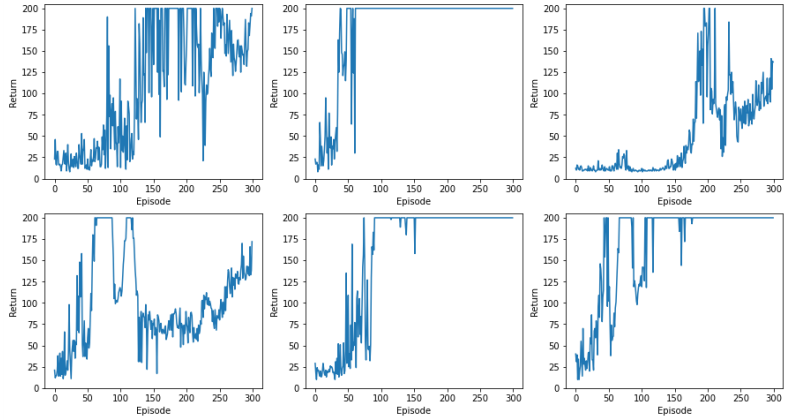

Abb. 7. Erzielter Reward REINFORCE mit linear α-decay

Am längsten und instabilsten ist das Training des Verfahrens Advantage Actor Critic mit one-Step Return. Diese hat zu Beginn einen stetigen Lernerfolg, braucht dadurch aber recht lange, das Maximum zu erreichen und bricht meistens nach wenigen Episoden wieder ein. Anders als bei REINFORCE kann hier nicht einfach die Lerngeschwindigkeit verringert werden, stattdessen wird ein n-Step Return in Form einer Generalized Advantage Estimation verwendet. Diese ist in der unteren Abbildung dagestellt. Bei der Verwedung des GAE anstatt des one-Step Advantages sind im Experiment die besten Ergebnisse erzielt worden. Das Verfahren konvergiert bereits nach wenigen Episoden und ist sehr konstant, besonders im Gegensatz zum A2C ohne GAE.

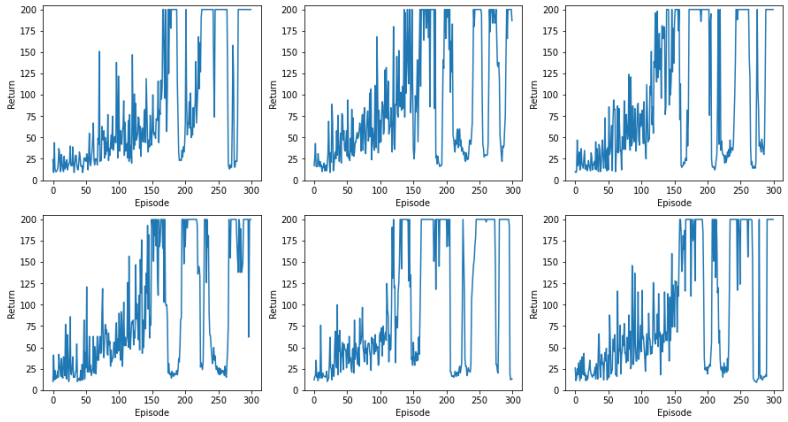

Abb. 8. Erzielter Reward mit one-Step Advantage Actor Critic

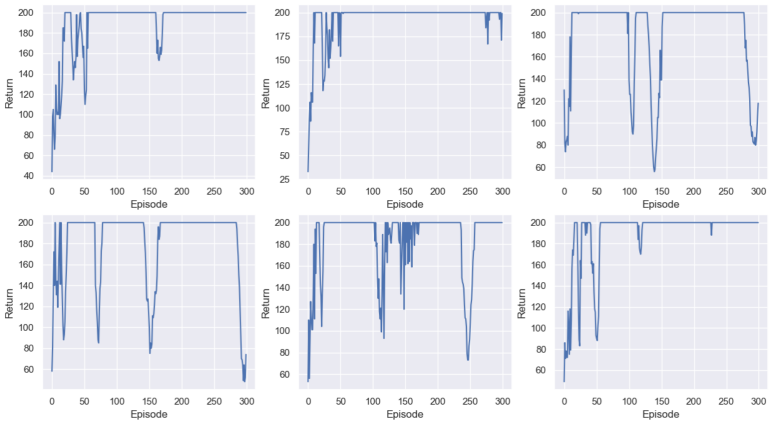

Abb. 9. Erzielter Reward mit n-Step GAE Advantage Actor Critic

4.2 Breakout

Anders als bei der CartPole-Umgebung ist bei Breakout der Observationspace, die Anzahl der möglichen States, deutlich größer und deshalb ist die Optimierung der Policy deutlich aufwendiger. Zudem ist die Anzahl der Aktionen deutlich größer und das Ziel komplexer, weshalb die Cross-Etropy-Method es nicht schafft diese zu erlernen. In den Ergebnissen ist keine Konvergenz oder ansatzweise Verbesserung des Spiels zu erkennen. Gleiches gilt für das Verfahren REINFORCE. Auch dieses Verfahren hat es nicht geschafft das Spiel Breakout zu erlenen und wirklich gut zu spielen. und auch das Verfahren A2C konnte dieses Ziel nicht erreichen, weder mit der Verwendung von GAE noch ohne GAE. Die Abbildung 10 zeigt den Verlauf für das Verfahren als one-Step TD-Verfahren, dieser unterscheidet sich aber nicht von den Verläufen der anderen Verfahren.

Beim Verfahren A2C mit GAE ergibt sich das gleiche Bild auf den gesamten Traiingsverlauf gesehen, in diesem ist jedoch am Anfang eine stufenweise Verbesserung des Spiels zu erkennen. In Abbildung 11 in den Episoden 0 bist 100 wird das Verfaren besser, stabilisiert sich dann aber und erreicht keine weiteren Erfolge, die dauerhaft messbar sind. Anders als erwartet ist diese dauerhafte Verbesserung nach 2100 Episoden nicht zu erkennen. Die anderen Verfahren konnte auch nach deutlich mehr Episoden keine Verbesserung aufweisen.

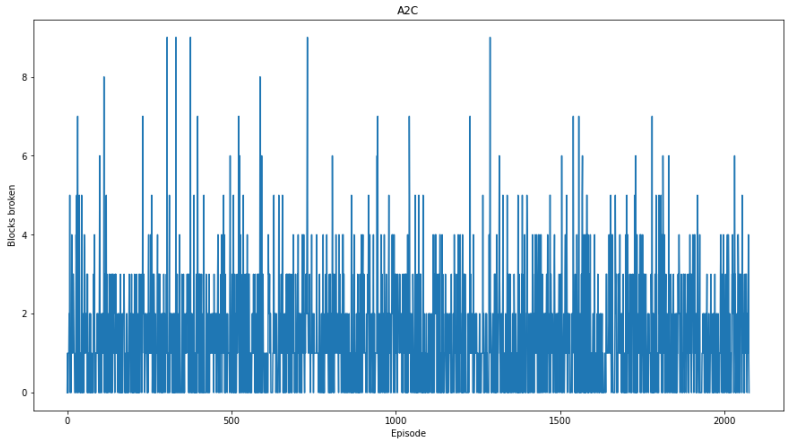

Abb. 10. Erzielter Reward mit one-Step Advantage Actor Critic

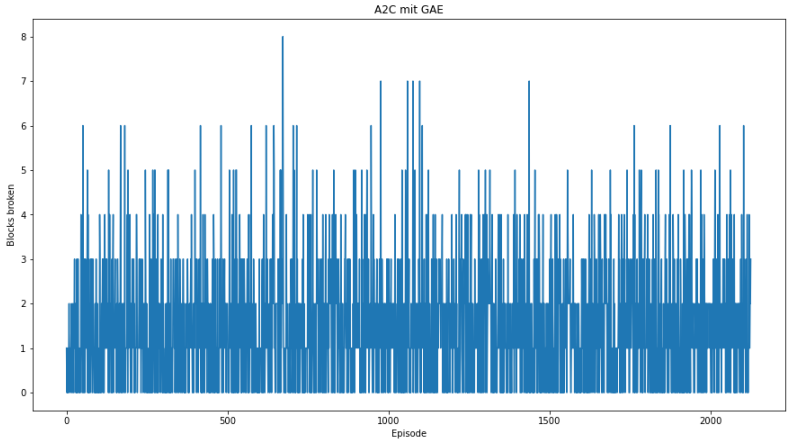

Abb. 11. Erzielter Reward mit n-Step GAE Advantage Actor Critic

Da keines der Verfahren auf Breakout während des Trainings zu guten Ergebnissen kam, wurde ein weiteres Experiment mit REINFORCE auf dem Spiel Pong durchgeführt, um die Funktionsweise der Verfahren auf Bilddaten überprüfen zu können. Dieses Spiel ähnelt Breakout, hat aber eventuell einen kleineren Observationspace, da nicht für jeden zerstörten Block sich der State bei Breakout ändert, bei Pong ist das Spielfeld konstant. In diesem Fall variieren nur die Position der Spieler und des Balls, weshalb die Varianz innerhalb der States geringer ist und diese so besser analysiert werden können. [7] Das Experiment wurde zwar frühzeitig abgebrochen, es ist jedoch im Verlauf zu erkennen, dass das Verfahren sehr schnell immer mehr Punkte erzielen und das Spiel gegen Ende fast mit 0 verlorenen Punkten gewinnen konnte.

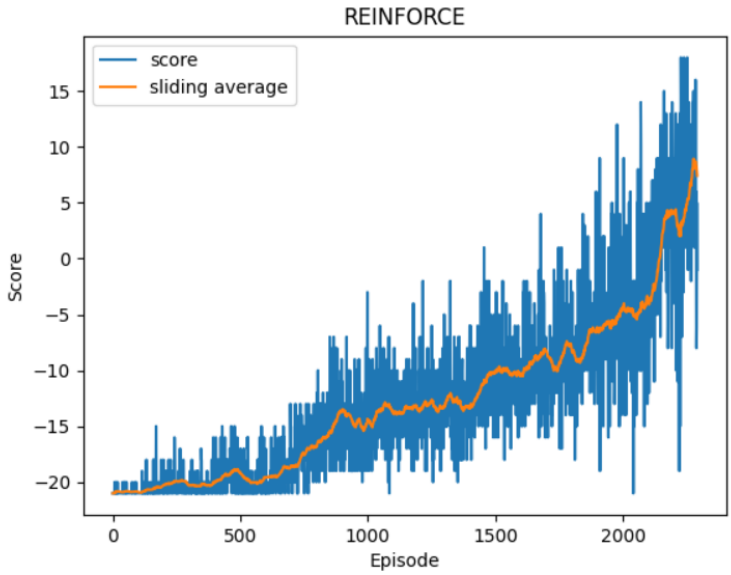

Abb. 12. Erzielter Reward mit REINFORCE auf Pong

5 Evaluation der Verfahren

In den Ergebnissen konnte gezeigt werden, dass das Verfahren CEM besonders gut in einfachen Enviroments mit einem kleinem Observationspace und wenig Aktionen zurecht kommt. Es konvergiert ähnlich schnell wie A2C mit GAE, ist in den erzielten Max-Rewards jedoch deutlich konstanter als dieses. Die gute Performanz des Verfahrens in der Cartpole-Umgebung ist auf die Simplizität und nur das Lernen von der besten Trajektorie zurückzuführen. In dieser Environment ist es für einen Agenten sehr einfach zufällig ein gutes Ergebnis zu

erzielen, welches dann leicht nachzuempfinden ist. Zudem kann in dem Verfahren eingestellt werden, wie viel Prozent der Trajektorien für das Training verwendet werden sollen. Wählt dieser Parameter so gewählt, dass nur von der besten Trajektorie gelernt werden soll. Sobald das Verfahren jedoch zwei oder mehr Trajektorien verwendet, werden die Ergebnisse deutlich schlechter und konvergieren nur noch selten. Das Verfahren schafft es allerdings nicht in komplexeren Umgebungen wie Breakout oder Pong mit Bildeingabe zu konvergieren, da hier zufällige Erfolge sehr selten sind und der Reward stark verzögert ist. Außerdem ist CEM als Monte-Carlo-Verfahren einer großen Varianz ausgesetzt und hat keinen Ansatz diese zu verkleinern. Besonders in dieser größeren Environment ist die Varianz jedoch hoch, weshalb das Verfahren nicht konvergieren und die Policy nicht erlernen kann. Das zeigt, dass die Cross-Entropy-Methode zwar sehr effektiv in kleinen Environments ist, aber schnell an seine Grenzen kommt und dann keinen Lernerfolg mehr erzielen kann. Dieses Ergebnis wird auch deckt sich mit anderen Arbeiten. Ohne Anpassungen war das Verfahren auch nicht in der Lage eine Policy für das Spiel Tetris zu erlernen und kam häufig durch die hohe Varianz zu falschen Ergebnissen und konvergierte zu einer schlechten Policy. [2]

In der CartPole-Environment liegt die Performance von REINFORCE deutlich hinter CEM. Es benötigt mehr Episoden um das Maximum zu erreichen und es fällt dem Verfahren danach, besonders ohne linear-α-decay, sehr schwer, das bereits erlernte Wissen nicht wieder zu vergessen. Durch die Verwendung des Decays konnte zu Beginn eine höhere Lerngeschwindigkeit gewählt werden ohne im späteren Verlauf Erlerntes zu vergessen. Dies führte zu stellenweise ähnlich guten Ergebnissen wie CEM, war aber immer noch langsamer und inkonstanter. Dies liegt daran, dass das Verfahren deutlich weniger Episoden zum lernen hat und auch schlechte Episoden verarbeiten muss. Die Parameter bei CEM wurden bewusst so gewählt, dass nur die Beste Trajektorie verwendet wird. Dies zeigt, dass in der kleinen Environment CartPole zwar besser abschneidet REINFORCE durch sein aufwendigeres Training jedoch deutlich sicherer in der Wahl der richtigen Aktion auch in selten erreichten States ist. Das liegt an der Verknüpfung des diskontierten Rewards mit der gewählten Aktion. Das es sich bei dem Verfahren um eine Monte-Carlo-Methode handelt, ist dieser Reward auch der tatsächlich erreichte, anders als A2C, wo dieser nur geschätzt wird. Aus diesem Grund approximiert REINFORCE eine Action-Value-Function und ist in der Lage auch zu erkennen, wenn eine Aktion in einem State zu schlechten Ergebnissen führt. Das Verfahren ist deshalb robuster auch in zufälligen Environments mit zufälligen Startpositionen und die Gesamttrainingszeit ist auch deutlich geringer, da die Anzahl der erfassten Trajektorien deutlich geringer ist und somit deutlich schneller gelernt werden kann. In der Breakout-Umgebung konnte, wie auch schon CEM, auch REINFORCE, anders als erwartet, keinen Lernerfolg erzielen. In dem Versuch mit Pong konnte gezeigt werden, dass REINFORCE grundsätzlich in der Lage ist durch die Anwendung der Baseline die Varianz der Trainingsdaten deutlich zu reduzieren und mit dem delayed Reward trotzdem eine Policy zu erlernen. Wie bereits erwähnt, wird aber für Breakout

erwartet, dass der Observation Space durch das Verschwinden der Blöcke und somit hinzukommen neuer States deutlich größer ist, weshalb dieses Spiel deutlich komplexer zu erlenen ist. Als Monte-Carlo-Verfahren war die Trainingszeit wahrscheinlich nicht lang genug und durch die zusätzlichen States die Varianz, auch Falle kleinerer Erfolge, zu hoch. Das Verfahren erreichte sehr konstant 2 Punkte als Reward, höhere Rewards waren jedoch selten. Im Durchschnitt wurde im Training ein Reward von 1.6 Punkten erreicht. Eventuell ist es möglich nach deutlich längerer Trainingszeit eine Policy zu erlernen, die Breakout spielen kann, das war im Rahmen dieser Arbeit jedoch nicht umsetzbar. Auch die Verwendung von Verbesserungen und Anpassungen, wie der α-decay sind in dieser Anwendungsfall nur schwer nutzbar, da der Trainingserfolg nur schwer abgeschätzt werden kann.

Die beste Performance in beiden Umgebungen konnte mit dem Verfahren Advantage Actor Critic mit Generalized Advantage Estimation erzielt werden. Das Verfahren erlernte in CartPole in jedem Durchlauf die Policy innerhalb weniger Episoden und konnte so sehr schnell den maximalen Reward erzielen und war danach, ähnlich wie REINFORCE mit α-decay sehr stabil. Zudem konnte auch bei Verlust des erlernten die Policy sehr schnell neu erlernt werden. Die schlechte Performance von A2C ohne GAE hingegen zeigt, dass der Vorteil des Verfahrens in der Verwendung von Bootstrapping und dem n-Step Return liegt. Durch das frühe Anpassen der Policy und der Value-Estimation an die richitgen Werte, besonders mit den frühen Rewards durch die CartPole-Umgebung, kann bereits das Training auf der ersten Episode positiv beeinflusst und deutlich verlängert werden. Zudem ist die Schätzung der jeweiligen werde durch den n-Step Returns deutlich besser als durch das one-Step Verfahren, weshalb die Werte besser angenähert werden können und einer deutlich geringeren Varianz unterliegen. Auf diese Weise wird der tatsächliche Reward, der bei Monte-Carlo vorliegt, nachgeahmt und somit die Vorteile eines Monte-Carlo-Verfahrens übernommen. Auch in der Breakout-Environment war A2C in beiden Konfigurationen nicht in der Lage konstant einen Score über 6 zu erreichen. Besonders zu beginn ist der Lernerfolg zu erkennen, die konvergiert jedoch sehr schnell gegen den maximal erreichten Reward von sechs, der Stellenweise übertroffen wurde. Da dies jedoch nicht regelmäßig passiert, wird dies nicht als wirklich gelernt angenommen. Im Durchschnitt erreicht das Verfahren mit GAE jedoch mit 2.4 einen leicht höheren Score als ohne die Verwendung von GAE mit 2.1. Auch wenn ohne GAE stellenweise höhere Rewards erzielt werden können, erreicht das Verfahren mit GAE konstanter die 4 Punkte.

6 Fazit

Aus den Ergebnissen der Experimente wird geschlossen, dass das Verfahrend A2C mit der Advantageschätzung GAE die beste Performance liefert. Diese Verfahren erreicht in der Umgebung CartPole am schnellsten konstant das Maximum von 200 Steps und konnte dieses sehr konstant halten. Auch in der Umgebung Breakout konnte mit diesem Verfahren das beste Ergebnis erzielt werden, auch wenn keines der Verfahren wirklich in der Lage diese Policy zu erlernen. Mit der zusätzlichen Verwendung von Pong für REINFORCE konnte gezeigt werden, dass die Probleme mit Breakout aus der Environment kommen, da das Verfahren grundsätzlich mit den vergrößerten Oberservationspaces durch die Bilder als Eingangsdaten zurecht kamen und in der Lage war Pong sicher zu erlenen. Auf Grund dieser Tatsache wird auch angenommen, dass A2C ebenfalls in der Lage ist Pong zu erlenen und den größeren Observationspace zu erfassen. Besonders in der einfachen Umgebung konnte auch CEM sehr gut abschneiden, da dieses Verfahren sehr schnell und simpel zu implemtieren ist und das CartPole sehr gut erlernt wurde. Allerdings gilt dies nur, wenn nur von der besten Trajektorie gerlernt wurde, wurden auch schlechtere Trajektorien mit einbezogen, war es nicht immer möglich eine Policy zu erlernen. Zudem dauert das Training des Verfahrens deutlich länger, das für jede Episode mehrere Trajektorien komplett abgelaufen werden, anders als bei den anderen Verfahren. Das schlechteste Ergebnis in der CartPole-Umgebung wurde durch das Verfahren REINFORCE erzielt. Es erlernte erst nach längerem Training und nicht immer eine gute Policy und brach auch mit verkleinertem α weiterhin ein. Allerdings ist diese Verfahren, anders als CEM auch in der Lage gewesen Pong mit Bildern als Input zu erlenen und konnte auch in Breakout Punkte erzielen.
Auf Grund der höheren Komplexität und begrenzten Zeit für das Training war es nicht möglich, mit einem der Verfahren eine Policy für Breakout zu erlenen. In diesem Fall wäre es besser gewesen stattdessen Pong zu verwenden. Für diese Environment konnte gezeigt werden, dass zumindest REINFORCE einen Policy erlernen kann. Auf Grund der ähnlichen Trainingsverläufe mit besseren Ergebissen wird angenommen, dass A2C mit GAE auch auf den Bilddaten bessere Ergebnisse liefert, als REINFORCE. Aus diesem Grund und den besten Ergebnissen in für das Cartpole wird angenommen, dass A2C mit GAE die beste Performance der Verfahren hat. Darauf folgt REINFORCE, welches mit α-decay mit halbwegs gute Ergebnisse im CartPole erreicht hat aber zusätzlich auch Pong erlenen konnte. Die aktuelle Implementation der CEM schneidet in diesem Experiment am schlechtesten ab, da zwar für das CartPole sehr schnell eine gute Policy erlernt werden konnte, auf der komplexen Environment jedoch kein messbares Lernen auf Grund der sehr hohen Varianz möglich war.

6.1 Ausblick

Die Ergebnisse zeigen, dass es für On-Policy-Verfahren sehr schwer ist, ein komplexes Spiel wie Breakout sicher zu erlernen. Wahrscheinlich ist die Anzahl der erreichten States und damit die Varianz deutlich zu hoch. Im Kapitel Grundlagen

wurde das Konzept des Off-Policy-lernen vorgestellt, welches besonders in sehr komplexem Umgebungen den Vorteil bietet auf Basis eines optimalen Agenten zu lernen. Auf diese Weise sollte es möglich sein dem Agenten in deutlich weniger Episoden eine Policy beizubringe. Jedoch wurde auch als Nachteil aufgefasst, dass ein solcher Agent schlechter generalisiert und deshalb in seinen Handlungen weniger robust ist. Eventuell bietet es sich in der Breakout-Environment an, eine Kombination aus On- und Off-Policy zu verwenden, um schneller zu einer gute Policy kommen zu können jedoch trotzdem durch selbständiges lernen in der Umgebung diese Generalisierungsfähigkeit zu erlangen.
Weiterhin stell das Paper [2] eine Möglichkeit vor, das Verfahren CEM mit Verrauschten Inputdaten an der Konvergenz zu hindern und so das Training künstlich zu verlangsamen und eine höhere Varianz der States zu erfassen und zu lernen mit diesen Umzugehen. Auf diese Weise ist CEM in der Lage ein Policy für das Spiel Tetris zu erlenen, was, wie in dem Paper gezeigt, ohne die verrauschten Daten nicht oder deutlich schwerer möglich ist. Unklar bleibt die Komplexität des Observationspaces, trotzdem könnte es sich hierbei um eine Methode handeln, auch mit CEM eine Policy zumindest für Pong eventuell aber auch für Breakout zu erlenen. [2]

Literatur

1. A. Khare, „Cross-entropy method for Reinforcement Learning“, Medium, 17. März 2020. https://towardsdatascience.com/cross-entropy-method-for-reinforcement-learning-2b6de2a4f3a0 (zugegriffen 3. März 2022).
2. I. Szita und A. Lőrincz, „Learning Tetris Using the Noisy Cross-Entropy Method“, Neural Computation, Bd. 18, S. 2006.
3. S. Causevic, „Policy Gradient REINFORCE Algorithm with Baseline“, Medium, 25. März 2021. https://towardsdatascience.com/policy-gradient-reinforce-algorithm-with-baseline-e95ace11c1c4 (zugegriffen 3. März 2022).
4. R. S. Sutton und A. G. Barto, Reinforcement learning: an introduction, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.
5. Pareigis, Stephan, „Reinforcement Learninig - VL08 Policy Gradient“. 2021.
6. F. Tree, „Understanding Baseline Techniques for REINFORCE“, Medium, 17. Oktober 2019. https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57 (zugegriffen 3. März 2022).
7. OpenAI, „Gym: A toolkit for developing and comparing reinforcement learning algorithms“. https://gym.openai.com (zugegriffen 15. Februar 2022).
8. M. Wang, „Advantage Actor Critic Tutorial: minA2C“, Medium, 26. Januar 2021. https://towardsdatascience.com/advantage-actor-critic-tutorial-mina2c-7a3249962fc8 (zugegriffen 6. März 2022).
9. A. V. de Kleut, „Actor-Critic Methods, Advantage Actor-Critic (A2C) and Generalized Advantage Estimation (GAE)“, Alexander Van de Kleut, 16. Juli 2020. https://avandekleut.github.io/a2c/ (zugegriffen 4. März 2022).
10. J. Schulman, P. Moritz, S. Levine, M. Jordan, und P. Abbeel, „High-Dimensional Continuous Control Using Generalized Advantage Estimation“, arXiv:1506.02438 [cs], Okt. 2018, Zugegriffen: 5. März 2022. [Online]. Verfügbar unter: http://arxiv.org/abs/1506.02438