LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

Trends in Mobilen und Verteilten Systemen

Curiosity und Diversity im Kontext von Reinforcement Learning – Ein Vergleich

Bearbeiter: David Jonathan Müller
Lukas Johannes Rieger

Betreuer: Thomas Gabor

Aufgabensteller: Prof. Dr. Claudia Linnhoff-Popien

Dokument erstellt
19. Juli 2020

Curiosity und Diversity im Kontext von Reinforcement Learning – Ein Vergleich

Zusammenfassung

Ansätze, die das traditionelle Reinforcement-Learning um zusätzliche *interne* Belohnungen erweitern, stellen in vielen Anwendungsfällen die deutlich effektivere Wahl dar. In dieser Arbeit stellen wir insbesondere den Ansatz des *Curiosity*-basierten Reinforcement-Learning vor. Wir erklären dessen theoretische Grundlagen, wie sie nach Schmidhuber erläutert werden. Anschließend stellen wir einen *Diversity*-basierten Reinforcement-Learning Ansatz vor und stellen ihn Schmidhubers Ansatz vergleichend gegenüber. Wir zeigen, wie beide Konzepte es einem RL-Agenten erlauben, auch in Umgebungen mit spärlichen externen Belohnungen zu agieren. Weiter legen wir dar, wie beide Ansätze trotz unterschiedlicher Herangehensweisen gewisse operationale Gemeinsamkeiten in Hinsicht auf die Selektion zukünftiger Aktionen aufweisen.

Inhaltsverzeichnis

1 Einleitung

Eine der nützlichsten Fähigkeiten des Menschen ist wohl seine Fähigkeit, komplexe Probleme in unterschiedlichsten Umgebungen und oftmals ohne vorheriges Wissen zu lösen. Diese Fähigkeit ist nicht nur für organische Lebewesen, sondern auch für künstliche Aktoren in einer Vielzahl an Szenarien äußerst nützlich. Ein häufig verwendeter Ansatz im Bereich der künstlichen Intelligenz stellt das *Reinforcement Learning* (RL) dar. Dieser Ansatz erlaubt es Agenten, diverse Aktionen im Kontext ihrer jeweiligen Umgebung zu erlernen und zu verbessern, ohne dabei notwendigerweise durch eine dritte Instanz überwacht werden zu müssen [9]. Für den Designer eines RL-Algorithmus stellt hierbei das Finden und Konfigurieren von *Belohnungen*, die dem Agenten Rückmeldung über die von ihm ausgeführten Aktionen geben sollen, einen zentraler Problembereich dar.

Da nach [11] in den meisten realen Anwendungsfällen Belohnungen von der Umgebung nur spärlich oder überhaupt nicht vorhanden sind (man spricht hier von *Sparse Reward*), benötigt man zusätzliche Methoden, um einen Lernfortschritt zu erzielen. Intuitiv lässt sich ein Lernanreiz entweder als extrinsisch, oder als intrinsisch definieren. Dient der Lernvorgang dem Erreichen eines externen, vorgegebenen Ziels, lässt sich dieser als extrinsisch kategorisieren. In diesem Fall fällt es leicht, die Belohnung eines Aktors in Abhängigkeit seines Zielfortschritts zu definieren. Da in realistischen Szenarien eine solche direkte Bewertung meist nur schwer vorzunehmen ist, wäre es von Vorteil, wenn eine etwaige Belohnung ebenfalls intrinsische Anreize in Betracht ziehen würde. Von solchen intrinsischen Anreizen spricht man dann, wenn die zugrundeliegende Aktion *inhärent* sinnvoll erscheint, also nicht von externen Gegebenheiten abhängt.

So scheint es etwa für den Menschen intuitiv sinnvoll, in einer unbekannten Umgebung möglichst viele unterschiedliche Vorgehensweisen auszuprobieren und sich dabei auch Strategien zu merken, welche vielleicht erst zu einem späteren Zeitpunkt Anwendung finden. Ein solches Vorgehen wäre auch bei einer künstlichen Intelligenz wünschenswert. Zu diesem Zweck existieren unterschiedliche Ansätze.

In dieser Arbeit betrachten und vergleichen wir zwei Methoden, welche das klassische Reinforcement Learning in dieser Hinsicht erweitern. Zu diesem Zweck werden zunächst in Kapitel 2 Markov Entscheidungsprozesse erklärt und das Sparse Reward Problem vorgestellt. Die anschließenden Kapitel fokussieren sich auf zwei Ansätze zur Bewertung des Lernfortschritts. In diesem Rahmen stellen wir in Kapitel 3 zuerst den Ansatz der "Curiosity" nach [12] vor. Wir betrachten außerdem in Kapitel 4 das Konzept der "Diversity" und dessen Anwendung im Reinforcement Learning nach [3]. Diese stellen wir daraufhin in Kapitel 5 vergleichend gegenüber und gehen auf deren Gemeinsamkeiten und Kernunterschiede ein. Kapitel 6 beinhaltet verwandte wissenschaftliche Arbeiten. In Kapitel 7 schließen wir mit einem Fazit.

2 Grundlagen

Im Folgenden werden *Markov Entscheidungsprozesse*, ein fundamentales Modell im RL, vorgestellt. Anschließend gehen wir kurz auf das *Sparse Reward Problem* ein.

2.1 Markov Entscheidungsprozesse

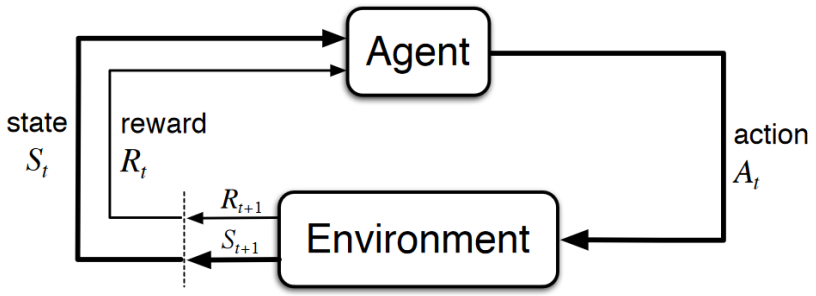


Abbildung 1: Typischer Reinforcement Learning Zyklus

Quelle: [2]

Das wohl elementarste Modell im RL sind Markov Entscheidungsprozesse (MDPs von engl. *Markov Decision Processes*), hier erklärt nach [2].

Ein MDP besitzt einen Entscheidungsträger, den so genannten *Agenten* (im Folgenden auch als *Aktor* bezeichnet), welcher mit seiner Umgebung (dem *Environment*) interagiert. Diese Interaktion findet sequentiell statt. In jedem Zeitschritt t verfügt der Agent über eine Repräsentation seiner Umgebung, den Zustand oder auch *State* S_t. Aufgrund der dem Agenten vorliegenden Information wählt dieser eine Aktion A_t. Dies führt dazu, dass das Environment in einen neuen Zustand S_{t+1} überführt wird. Außerdem erhält der Aktor von der Umgebung eine Belohnung R_{t+1}, den *Reward*.

Wie in 1 dargestellt findet dieser Prozess in einem andauernden Zyklus statt. Ziel des Agenten ist es, die Summe aller Rewards, die er für die Ausführung von Aktionen bekommt, zu maximieren. Sprich der Agent will nicht nur den folgenden Reward, sondern viel eher die Gesamtheit aller Rewards, die er auf Dauer bekommt, maximieren.

2.2 Sparse Reward Problem

Ein zentrales Problem im RL ist nach [7] der Umgang mit Umgebungen, in denen Belohnungen nur spärlich vorhanden sind. Eine Reihe von Belohnungen, von denen die meisten nicht positiv sind, wird als *Sparse Reward* bezeichnet. Diese machen es für RL-Algorithmen sehr schwer, eine Reihe von Aktionen mit

einem weit entfernten Reward zu verbinden. In extremen Fällen findet der Agent eventuell gar keine Belohnung und wird so niemals lernen, wie er die gegebene Aufgabe bewältigt [7].

Um das Problem zu umgehen müssen die Belohnungen im traditionellen RL sehr sorgfältig durchdacht sein. Dies erfordert viel Zeit und Verständnis für die Umgebung. Besser währe es also, wenn der Agent mit Sparse Rewards umgehen kann und die Ziele somit abstrakter und langfristiger definiert werden können, wodurch komplexere Aufgaben behandelt werden können [7].

Um dies zu erreichen existieren bereits mehrere Ansätze, von denen zwei im Folgenden vorgestellt werden.

3 Das Curiosity-Modell nach Schmidhuber

Um Schmidhubers System verstehen zu können, ist es wichtig, zuerst einige zentrale Grundbegriffe zu definieren. Die Säulen seines Werks bilden vor allem vier zentrale Prinzipien, die in *Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes* dargelegt werden [12].

Information ist "heilig" Diesem Prinzip zufolge muss die gesamte Abfolge an Aktionen und Beobachtungen eines Aktoren gespeichert werden. Begründet wird dies auf der Grundlage, dass diese Informationen die absolute Basis für jegliches Wissen darstellt, das der Aktor über seine Umgebung besitzt. Grundlegend für dieses Prinzip ist hierbei die Annahme, dass es überhaupt realistisch *möglich* ist, einen solchen Katalog aller Daten physisch zu speichern. Schmidhuber argumentiert, dass dies durchaus nicht unrealistisch sei und zeigt dies mit Hilfe eines Beispiels. Für dieses sind zuerst einige Annahmen nötig:

- Ein menschliches Leben dauert im Durchschnitt nicht länger als 3×10^9 Sekunden an.
- Ein menschliches Gehirn verfügt in etwa über 10^{10} Neuronen, welche jeweils über ca. 10^4 Synapsen verfügen.

Nimmt man nun an, dass lediglich die halbe Hirnkapazität genutzt wird, um rohe Daten zu speichern und jede Synapse höchstens 6 bit speichern kann, so ist es durchaus möglich, die gesamten sensorischen Eindrücke eines Lebens bei einer Rate von 10^5 $bits/s$ zu sichern.

Verbessern von subjektiver Komprimierbarkeit Demzufolge lässt sich jede Regularität im Informationsstrom dazu nutzen, diesen weiter zu komprimieren. Eine solche komprimierte Version des ursprünglichen Datenstroms lässt sich also als eine Art *vereinfachter Erklärung* von diesem interpretieren. Um die Verbesserung dieser subjektiven Komprimierbarkeit zu erreichen gilt also, dass ein Agent zumindest einen Teil seiner Rechenzeit darauf verwenden sollte, mithilfe eines Kompressionsalgorithmus seine Daten zu komprimieren.

Intrinsische Belohnungen spiegeln Kompressionsfortschritt wieder Ein Agent sollte seinen Kompressionsfortschritt überwachen und bei erfolgreicher Komprimierung entsprechend darauf reagieren. In Abhängigkeit der eingesparten Bits wird dementsprechend eine intrinsische Belohnung für den Agenten generiert.

Intrinsische Belohnungen maximieren Ein entsprechender Reinforcement Learning Algorithmus kann nun versuchen, die zu erwartende intrinsische Belohnung zu maximieren. Laut Schmidhuber würde sich ein solcher Algorithmus

vor allem auf solche Aktionen fokussieren, die es erlauben, neue aber erlernbare Regularitäten zu finden oder zu erstellen.

3.1 Funktionsweise des Kompressors

Die folgenden Abschnitte verwenden zur Darstellung zeitabhängiger Variablen Q zum Zeitpunkt t die Schreibweise $Q(t)$. Gleichfalls stellt $Q(\leq t)$ die geordnete Sequenz an Werten $Q(1), \ldots, Q(t)$ und $Q(< t)$ wiederum die Sequenz $Q(1), \ldots, Q(t-1)$ dar. Diese Schreibweise wurde zur besseren Übersichtlichkeit direkt von Schmidhubers Ausführungen übernommen [12].

Schmidhuber merkt im Bezug auf Kompressoren an, dass das *Vorhersagen* zukünftiger Eingabewerte $x(\tau)$ gegeben einer Historie $h(< \tau)$ durchaus dazu verwendet werden kann, um $h(\leq t)$ kompakt zu kodieren [12, p. 18] Ein Beispiel für einen solchen Kompressor/Prediktor findet sich in Solomonoffs universeller, induktiver Inferenz [12, p. 18]. Dieser Ansatz versucht das Problem zu lösen, mit welcher Wahrscheinlichkeit sich von einer langen Sequenz auf die anschließende Teilfolge schließen lässt [13].

3.1.1 Interne Symbole als Konsequenz effizienter Komprimierung

Schmidhuber beschreibt weiter, aufbauend auf einer generellen Historie an Beobachtungen, wie ein vorausschauendes neurales Netz, das als Kompressor fungiert, spezielle interne Repräsentierungen generiert, welche über häufig auftretende Dinge abstrahieren. Diese internen Darstellungen werden kurz *Symbole* genannt [12, p. 6]. Ein naheliegendes Beispiel findet sich laut Schmidhuber etwa im Tag/Nacht-Rhythmus. Da der Sonnen Auf- und Untergang sich beständig wiederholt, ist es effizienter, ein spezifisches Symbol für diesen Prozess zu generieren und die Komprimierung dadurch voranzutreiben.

Leistungsmessung eines Kompressors Zur Bewertung eines Kompressors p, der eine Historie $h(\leq t)$ zum Zeitpunkt t komprimiert, führt Schmidhuber den Wert

$$C_l(p, h(\leq t)) = l(p) \tag{1}$$

ein. In diesem Kontext stellt $l(p)$ die Länge von p in Bits dar. Je kürzer der Kompressor also ist, desto mehr Regelmäßigkeit lässt sich in den bisherigen Beobachtungen finden [12, p. 19]. Das maximale Limit von $C_l(p, h(\leq t))$ stellt laut Schmidhuber die Kolmogorov-Komplexität $K^*(h(\leq t))$, also das kürzeste Programm, welches eine Ausgabe produziert, die mit $h(\leq t)$ beginnt, dar.

Während eine Leistungsmessung dieser Art zwar möglich ist, muss idealerweise auch die *Zeit* in Betracht gezogen werden, die ein Kompressor benötigt, um eine gegebene Historie h zu komprimieren. Andernfalls wäre ein Szenario denkbar, in der ein Kompressor zwar eine extrem kompakte Repräsentation der vorliegenden Daten produzieren könnte, für diesen Prozess aber extrem viel Rechenzeit benötigt und somit praktisch unbrauchbar wäre. Aus diesem Grund

stellt Schmidhuber noch einen zweiten Messungsansatz vor, welcher eine Reduzierung der Rechenzeit um $\frac{1}{2}$ äquivalent zu einer Komprimierung um 1 Bit behandelt [12, p. 19]

$$C_{l\tau}(p, h(\leq t)) = l(p) + \log \tau(p, h(\leq t)) \tag{2}$$

3.2 Funktionsweise des Agenten

Der in Kapitel 3.1 beschriebene Kompressor stellt offensichtlich nur einen Teil des Ganzen dar. Der Agent an sich verfügt ebenfalls über einen *Controller*, der auf Basis der aktuellen Beobachtung eine entsprechende Reaktion generiert. Im Folgenden wird nun erläutert, auf welchen Prinzipien diese Aktionen gewählt und ausgeführt werden.

Subjektive Interessantheit als Ableitung subjektiver Schönheit Damit es einem Agenten möglich sein kann, auf eine Weise *neugierig* zu handeln, benötigt man offensichtlich eine formale Definition dessen, was eine *interessante* Beobachtung ausmacht. Zu diesem Zweck benötigt man allerdings zuerst ein Konzept von *subjektiver Schönheit*.

Laut Schmidhuber lässt sich die subjektive Schönheit einer Beobachtung als die Menge an benötigten Bits definieren, um diese Beobachtung zu kodieren [12, p. 7]. Genauer formuliert bedeutet dies:
Sei $O(t)$ der Zustand eines subjektiven Beobachters am Zeitpunkt t. Die subjektive Schönheit sei gegeben durch $B(D, O(t))$, wobei es sich bei D um die jeweilige Beobachtung handelt. Dieses Maß der subjektiven Schönheit ist proportional zur Menge an Bits um D zu beschreiben, wobei jegliches vorheriges Wissen des Beobachters ebenfalls in Betracht gezogen werden muss [12, p. 7]. Schmidhuber erklärt diese Definition anhand des Beispiels eines menschlichen Gesichts. So würde es sich für einen Kompressor anbieten, eine interne Repräsentation eines archetypischen Gesichts zu generieren, um bereits beobachtete Gesichter möglichst effizient zu kodieren. Die Beobachtung eines neuen Gesichts erlaubt es dann, lediglich die spezifischen Abweichungen dieses neuen Gesichts vom internen Prototypen zu speichern. Ein solcher Beobachter würde also genau solche Gesichter als subjektiv am schönsten klassifizieren, die am geringsten vom internen Gesichtsprototypen des Beobachters abweichen. [12, p. 7]

Schmidhuber führt nun, aufbauend auf der subjektiven Schönheit, den Begriff der *subjektiven Interessantheit* ein. Er argumentiert, dass eine subjektiv schöne Beobachtung nur so lange interessant ist, bis der Beobachter die spezifische Regularität des Objekts komplett verarbeitet hat [12, p. 7-8]. In anderen Worten ist eine Beobachtung also nur dann für einen Beobachter interessant, wenn dessen Kompressor diese Beobachtung noch nicht optimal zusammenfassen kann.

Genauer definiert Schmidhuber nun die *zeitabhängige* subjektive Interessantheit $I(D, O(t))$ einer Beobachtung D in Relation zu einem Beobachter O am

Zeitpunkt t als

$$I(D, O(t)) \sim \frac{\partial B(D, O(t))}{\partial t} \tag{3}$$

, also der ersten Ableitung der subjektiven Schönheit [12, p. 8]. Ein Agent kann seine Beobachtungen also über Zeit besser analysieren und etwaige Wiederholungen oder Regelmäßigkeiten erkennen, wodurch die beobachteten Daten subjektiv schöner werden. Solange diese Komprimierungsprozess anhält, handelt es sich um *interessante* Daten. [12, p. 8]

3.2.1 Wie neugierige Agenten operieren

Auf Basis der subjektiven Schönheit und darauf aufbauend, der subjektiven Interessantheit, lässt sich nun erläutern, wie etwa ein auf Reinforcement Learning basierender Agent handeln würde. Schmidhuber setzt an, dass im Falle von fehlenden äußeren Belohnungen ein entsprechender Agent versuchen würde, die *Interessantheit* zu maximieren [12, p. 8]. Es werden also genau solche Sequenzen an Aktionen vom Agenten ausgewählt, welche zukünftig den zu erwartenden Kompressionsfortschritt maximieren [12, p. 8]

Betrachtet wird zuerst ein Agent, der seine Umgebung in konkreten Zeit Intervallen $t = 1, 2, \ldots, T$ wahrnimmt und verändert. Nimmt man an, dass der Agent zu jedem beliebigen Zeitpunkt t einen reellen Eingabewert $x(t)$ von der Umgebung erhält und daraufhin eine reelle Aktion $y(t)$ ausführt, so ergibt sich als finales Ziel für den Agenten die Maximierung des zukünftigen *Nutzens*:

$$u(t) = E_\mu \left[\sum_{\tau=t+1}^{T} r(\tau) \,\middle|\, h(\leq t) \right] \tag{4}$$

Dabei stellt $r(t)$ einen zusätzlichen reellen Belohnungswert, $h(t)$ das geordnete Tripel $[x(t), y(t), r(t)]$ und $E_\mu(\cdot \mid \cdot)$ den bedingten Erwartungsoperator in Bezug auf eine möglicherweise unbekannte Verteilung μ aus einer Menge M möglicher Verteilungen darstellt. [12, p. 17] In diesem Kontext stellt $h(\leq t)$ also die Historie bis zu dem Zeitpunkt t dar.

Der Belohnungswert $r(t)$ kann laut Schmidhuber noch weiter in eine Form von $r(t) = g(r_{ext}(t), r_{int}(t))$ aufgeteilt werden [12]. Die Funktion g bildet hierbei die beiden reellen Werte r_{ext} und r_{int} auf einen weiteren reellen Wert ab. In diesem Kontext entspricht r_{ext} dem traditionell verwendeten Wert der *externen* Belohnung, welche durch die Umgebung generiert wird und beispielsweise entsteht, wenn der Agent ein gegebenes Ziel erreicht. Da sich der Fokus allerdings hauptsächlich auf den *internen* Belohnungswert $r_{int}(t)$ konzentriert, kann man im Folgenden zum einfacheren Verständnis stets von $r_{ext}(t) = 0$ ausgehen.

Bewertung des Kompressorfortschritts Um den internen Belohnungswert zu berechnen, ist nicht die generelle Leistung eines Kompressors an sich von Interesse. Ähnlich wie schon im Abschnit 3.2 liegt der Fokus viel mehr auf der *Änderung* dieses Werts über mehrere Zeitschritte hinweg - also der *Verbesserung* des Kompressors. Dementsprechend definiert Schmidhuber den internen

Belohnungswert in Reaktion auf den Fortschritt des Kompressors als

$$r_{int}(t+1) = f\left[C(p(t), h(\leq t+1)), C(p(t+1), h(\leq t+1))\right], \tag{5}$$

wobei es sich bei um eine Funktion handelt, welche als Eingabe jeweils die Leistung des alten und des neuen Kompressors über die Historie bis zum Zeitpunkt $t+1$ erhält. Die wohl einfachste Funktion, welche potentiell für f in Frage käme, wäre beispielsweise $f(a,b) = a - b$. Diese würde also eine positive, interne Belohnung generieren, welche direkt der diskreten Verbesserung der neuen Kompressorleistung gegenüber der alten Leistung entspricht.[12, p. 19]

3.3 Agent und Kompressor in Wechselwirkung

Aufbauend auf den bisher genannten Prinzipien von Schmidhubers neugierigen Agenten kann nun ein informeller Algorithmus beschrieben werden, der das (in diesem Fall asynchrone) Zusammenspiel zwischen dem Controller des Agenten, also dem aktiv agierenden Teil, und dem Kompressor aufzeigt.

Sei $p(t)$ das Kompressorprogramm zum Zeitpunkt t und $s(t)$ der Controller zum Zeitpunkt t [12, p. 20]:

Controller Solange für t gilt $1 \leq t \leq T$:

1. Der aktuelle Controller $s(t)$ nutzt Teile der Historie $h(t)$ um eine Aktion $y(t+1)$ auszuführen.
2. Beobachte die Eingabe $x(t+1)$.
3. Prüfe, ob der asynchron laufende Kompressor eine interne Belohnung $r_{int}(t+1)$ größer Null generiert hat. Andernfalls, setze $r_{int}(t+1) = 0$.
4. Der Reinforcement-Learning-Algorithmus des Controllers kann nun die Historie $h(t+1)$ zusammen mit $r_{int}(t+1)$ nutzen, um einen neuen Controller $s(t+1)$ zu erhalten, der wiederum die zu erwartende *Nützlichkeit* maximiert, wie in Formel (4) beschrieben.

Kompressor Wir nehmen an, dass p_{new} zu Beginn dem initialen Kompressor entspricht. Für jeden Zeitschritt t durchläuft der Kompressor somit:

1. Setze $p_{old} = p_{new}$. Setze zudem $h_{old} = h(\leq t)$. h_{old} entspricht also der Historie am aktuellen Zeitpunkt t.
2. Wende Kompressor p_{old} auf h_{old} an und berechne, Sektion 3.1.1 entsprechend, $C(p_{old}, h_{old})$. Gerade dieser Schritt kann unter Umständen sehr zeitaufwändig sein.
3. Wende nun einen, dem jeweiligen Agentenziel entsprechenden, Algorithmus an, um einen besseren Kompressor p_{new} auf Basis von h_{old} zu generieren. In diesem Schritt ist zu beachten, dass p_{new} nicht zwingend *optimal* sein muss.

4. Wende, wie bereits in Schritt 2, p_{new} auf h_{old} an, um $C(p_{new}, h_{old})$ zu erhalten.

5. Gegeben dem aktuellen Zeitpunkt τ, berechne

$$r_{int}(\tau) = f\left[C(p_{old}, h_{old}), C(p_{new}, h_{old})\right] \tag{6}$$

Sowohl Controller, als auch Kompressor agieren also größtenteils unabhängig voneinander. Da gerade der Kompressor die Komponente darstellt, welche den größten Arbeitsaufwand übernimmt, muss berücksichtigt werden, dass es unter Umständen zu großen zeitlichen Abständen zwischen den *Aktionen* des Controllers und einer entsprechenden Belohungsantwort des Kompressors kommen kann [12, p. 20].

3.4 Fazit

Betrachtet man Schmidhubers Konzept der *Neugierde* und wie er dieses auf den Themenbereich des Reinforcement-Learning anwendet, fällt besonders die relative Simplizität ins Auge. Schmidhuber selbst beschreibt sein Konzept zuletzt als ein *"surprsingly simple algorithmic principle"*[12, p. 16], welches nichtsdestotrotz eine effektive Lösung für das *Sparse Reward Problem* findet. Zusätzlich liefert es eine Strategie zur ausgeglichenen Erkundung der Umgebung, indem es bereits bekannte Regelmäßigkeiten als weniger Interessant einstuft.

4 Diversity im Reinforcement Learning

In diesem Kapitel beschäftigen wir uns damit, wie das Konzept von *Diversity* im Bereich des RL Anwendung finden kann. Ziel ist, dass ein RL-Agent in einer unüberwachten Phase zunächst Fähigkeiten (*Skills*) erlernt, welche das Bewältigen von Aufgaben in der darauffolgenden, überwachten Phase erleichtern sollen. Dieser Prozess geschieht ohne jegliche Belohnung von außen.

Dieses Kapitel stützt sich zu einem Großteil auf *Diversity is all you need: Learning skills without a reward function* von Eysenbach u. a. [3]. Falls nicht anders angegeben, wurden die Informationen hieraus entnommen.

4.1 Wichtige Begriffe aus der Informationstheorie

Im Folgenden werden Begriffe aus der Informationstheorie verwendet, welche es zunächst zu klären gilt. Wir betrachten *Entropie* und *Transinformation* nach [1] und [14].

Entropie Die Entropie beschreibt in der Informationstheorie den mittleren Informationsgehalt bzw. die Ungewissheit einer Quelle.

Ist beispielsweise bei einer Ereignismenge jedes Ereignis gleich wahrscheinlich, so ist die Ungewissheit maximal. Tritt hingegen ein bestimmtes Ereignis mit verhältnismäßig großer Wahrscheinlichkeit ein, ist die Ungewissheit gering [14].

Nach [1] besitzt eine diskrete Zufallsvariable X mit dem Zeichenvorrat $\mathcal{X}$ und der Wahrscheinlichkeitsfunktion $p(x)$ die *Entropie*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \cdot log\ p(x)$$

Diese lässt sich nach [1] auch über den Erwartungswert E_p berechnen unter Verwendung von

$$H(X) = E_p\ log\ \frac{1}{p(X)}$$

woraus unter Anwendung der Rechenregeln für Logarithmus und Erwartungswert trivial

$$H(X) = -E_p\ log\ p(X) \tag{7}$$

folgt.

Desweiteren ist die Formel für die *bedingte Entropie*, also die Ungewissheit einer Zufallsvariable Y gegeben X, nach [1] gegeben durch

$$H(Y|X) = -E_{p(x,y)}\ log\ p(Y|X) \tag{8}$$

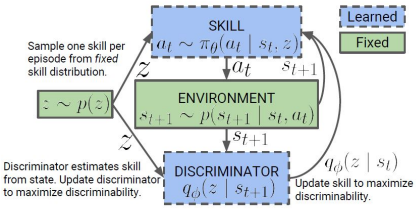
.

Transinformation Die Transinformation beschreibt die Menge an Information, die eine Zufallsvariable über eine andere enthält bzw. die Reduktion der Ungewissheit aufgrund des Wissens um die jeweils andere Zufallsvariable [1].

Mathematisch wichtig für uns ist lediglich der Zusammenhang zwischen *Transinformation* und *Entropie*, welcher nach [1] gegeben ist durch

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{9}$$

4.2 Funktionsweise



Algorithm 1: DIAYN

while *not converged* **do**
  Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$
  for $t \leftarrow 1$ **to** $steps\_per\_episode$ **do**
    Sample action $a_t \sim \pi_\theta(a_t \mid s_t, z)$ from skill.
    Step environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$.
    Compute $q_\phi(z \mid s_{t+1})$ with discriminator.
    Set skill reward $r_t = \log q_\phi(z \mid s_{t+1}) - \log p(z)$
    Update policy (θ) to maximize r_t with SAC.
    Update discriminator (ϕ) with SGD.

Abbildung 2: "Diversity is all you need" Algorithmus

Quelle: [3]

Das selbstständige Erlernen von brauchbaren Fähigkeiten, so wie in [3] beschrieben, baut auf drei Grundideen auf.

(1) Unterschiedliche Fähigkeiten sollten andere Zustände besuchen, sodass ihre Unterscheidbarkeit gewährleistet ist.

(2) Um besagte Fähigkeiten zu unterscheiden, werden nicht die Aktionen, sondern die Zustände betrachtet. Das liegt daran, dass Aktionen, welche die Umgebung nicht beeinflussen, für einen Beobachter nicht erkennbar sind. Das Paper verdeutlicht dies mit einem treffenden Beispiel: Für einen außenstehenden Betrachter ist nicht ersichtlich, wie fest ein Roboterarm eine Tasse in der Hand hält, falls sich diese nicht bewegt. Die exakten Aktionen, die für das Halten der Tasse getätigt werden, sind also letztendlich für die Unterscheidung weniger relevant als die beobachtbaren Zustände.

(3) Schließlich soll erreicht werden, dass die Fähigkeiten so vielfältig (*diverse*) wie möglich sind. Die Idee ist, dass unterscheidbare Skills mit einer hohen Entropie einen Zustandsraum abdecken, welcher sich weit entfernt von anderen Fähigkeiten befindet.

Um diese Punkte zu realisieren, definieren wir zunächst die Zufallsvariablen S und A für Zustände und Aktionen. Sei nun $Z \sim p(z)$ eine latente Variable, unter deren Bedingung die Strategie definiert wird. Bei fixem Z sprechen wir hier von einer *Fähigkeit*. Entropie sowie Transinformation werden zur Basis e berechnet.

Nun soll die Transinformation zwischen Skill und State, $I(S;Z)$, maximiert werden. Dies stellt sicher, dass die Fähigkeit bestimmt, welche Zustände vom Agenten durchlaufen werden. Anders formuliert lässt sich aus den besuchten Zuständen auf die Fähigkeit schließen.

Außerdem minimieren wir die Transinformation zwischen Skill und Aktion bei gegebenen Zustand, $I(A;Z|S)$. So wird garantiert, dass nicht Aktionen, sondern Zustände für die Unterscheidung von Fähigkeiten betrachtet werden.

Zuletzt maximieren wir die Entropie $H(A|S)$.

Zusammengefasst maximieren wir nach [3] also das *Objective*

$$\begin{aligned}\mathcal{F}(\theta) &\triangleq I(S;Z) + H(A|S) - I(A;Z|S) && (10)\\ &= (H(Z) - H(Z|S)) + H(A|S) - (H(A|S) - H(A|S,Z))\\ &= H(Z) - H(Z|S) + H(A|S,Z) && (11)\end{aligned}$$

Der Term wurde unter Verwendung der Gleichung (9) umgeformt.

Da sich $p(z|S)$ nicht genau berechnen lässt, wird das Folgende mit einem Discriminator q_ϕ approximiert. Mit der Jensenschen Ungleichung haben wir laut [3] eine untere Schranke $\mathcal{G}(\theta,\phi)$ für $\mathcal{F}(\theta)$:

$$\begin{aligned}\mathcal{F}(\theta) &= H(A|S,Z) - H(Z|S) + H(Z)\\ &= H(A|S,Z) + E_{z\sim p(z),s\sim\pi(z)}(log\ p(z|s)) - E_{z\sim p(z)}(log\ p(z)) && (12)\\ &\geq H(A|S,Z) + E_{z\sim p(z),s\sim\pi(z)}(log\ q_\phi(z|s) - log\ p(z)) \triangleq \mathcal{G}(\theta,\phi)\end{aligned}$$

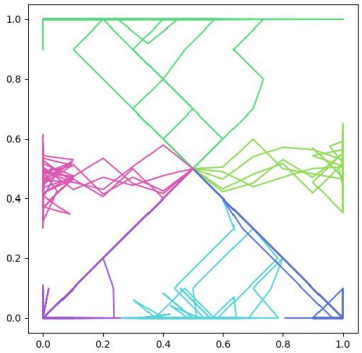
In (12) wurden die Entropien nach (7) und (8) umgeformt.

Nach der unüberwachten Trainingsphase verfügt der Agent über eine Sammlung von verschiedenartigsten Fähigkeiten, von denen vermutlich einige nutzlos sind. Der von diesen abgedeckte Zustandsraum ist jedoch für die anderen Fähigkeiten tabu. So ist es aufgrund der implementierten Diversity sehr wahrscheinlich, dass es eine gewisse Anzahl an brauchbaren Fähigkeiten gibt, da sich diese von den unbrauchbaren möglichst stark unterscheiden sollen.

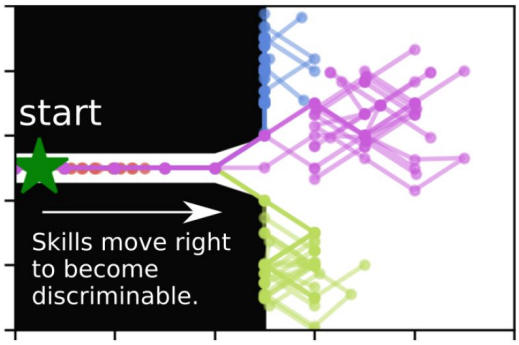
4.3 Experimente und Beispiele

Wie bereits mehrfach angemerkt ist das Ziel das eigenständige Erlernen von vielfältigen Fähigkeiten. [3] liefert zunächst zwei simple Beispiele einer zweidimensionalen Navigation in einem leeren Raum (siehe Abbildung 3). Hierbei stellen die farbigen Linien unterschiedliche Fähigkeiten dar. Eine Fähigkeit entspricht also im Experiment einem erkundeten Pfad innerhalb des Graphen. Die Menge der erreichbaren Koordinaten innerhalb des Graphen ist in diesem Experiment der Zustandsraum.

In Abbildung 3a ist somit sehr anschaulich zu erkennen, wie die Fähigkeiten unterschiedliche Zustandsräume abdecken. Es fällt außerdem auf, dass die Pfade eine starke Tendenz zu den Rändern haben. Dieses Verhalten macht durchaus



(a) Start in der Mitte



(b) Zeitweilig überlagernde Fähigkeiten

Abbildung 3: Experiment zum selbstständigen Finden von unterschiedlichen Pfaden im zweidimensionalen Raum

Quelle: [3]

Sinn, da alle in der Mitte des Koordinatensystems starten. Das logische Vorgehen zum maximalen Distanzieren der Pfade ist die Ausbreitung in unterschiedliche Richtungen, da der Abstand zwischen zwei angrenzenden Pfaden - und somit auch der zu allen anderen - nach außen hin wächst. Anders gesagt ist am Rand am meisten "Platz" um sich auszubreiten.

Einen Schritt weiter gedacht zeigt das Beispiel in Abbildung 3b, dass sich Fähigkeiten unter Umständen auch zeitweilig überlagern können, solange sie schlussendlich unterscheidbare Zustandsräume abdecken. Hier stellen die schwarzen Flächen Hindernisse dar, welche nicht von Pfaden betreten werden können. Es lässt sich erkennen, dass der blaue und der grüne Pfad sobald es möglich ist nach oben bzw. unten abbiegen, um sich vom lila Pfad zu distanzieren.

In beiden Beispielen ist gut zu sehen, wie sich die Fähigkeiten gegenseitig "abstoßen" und so automatisch ein Großteil des Zustandsraums abgedeckt wird. Die erkundeten Zustände liegen außerdem relativ gleichmäßig über dem Zustandsraum verteilt.

Interessanter wird es bei den komplexeren Versuchsaufbauten. Wir betrachten im Folgenden den von [3] als "Half Cheetah" betitelten Agenten. Hierbei handelt es sich um ein hundartiges, zweidimensionales Wesen, welches die Kontrolle über sein Vorder- und Hinterbein besitzt (siehe Abbildung 4).

Ohne jegliche externe Belohnung hat dieser Agent gelernt, nach vorne und hinten zu rennen beziehungsweise zu gehen. Außerdem besitzt er die Fähigkeit, einen Vorwärtssalto zu machen. Wir empfehlen zu besseren Veranschaulichung die Videos auf [4].

Eine Reward-Funktion, welche ein solches Verhalten hervorruft, manuell zu definieren ist nicht einfach.

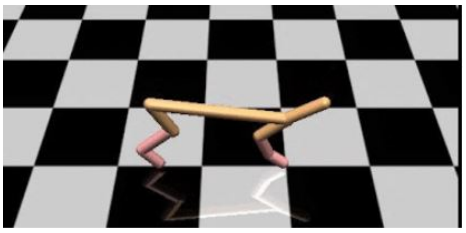


Abbildung 4: Experiment “Half Cheetah”

Quelle: [4]

4.4 Verwendung der gelernten Fähigkeiten

Nachdem nun eine Sammlung von Fähigkeiten erlernt wurde, stellt sich nun die Frage, wie sich diese effektiv nutzen lässt.

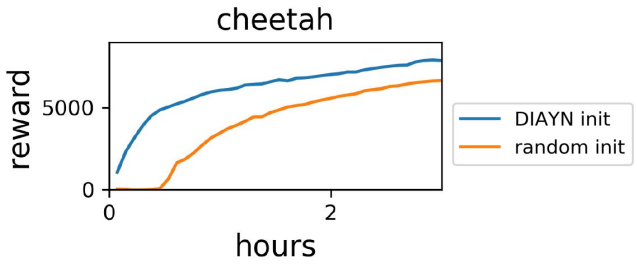


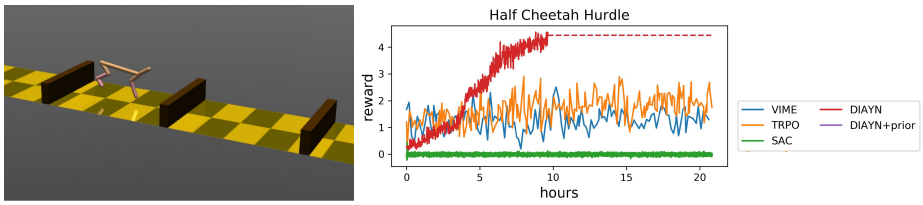
Abbildung 5: Trainingserfolg des Half Cheetah mit und ohne DIAYN (waagrecht: vergangene Zeit in Stunden, senkrecht: erhaltene Belohnung)

Quelle: [3]

[3] führt hier als erstes an, dass der DIAYN-Algorithmus als ein Vortraining genutzt werden kann. Danach extrahiert man die Fähigkeit mit dem größten Reward und passt entsprechend die Initialgewichte der eigentlichen Policy an. Abbildung 5 zeigt für das Beispiel des Half Cheetah, dass sich bei Initialisierung mit einer vorgelernten Fähigkeit (blaue Linie) schneller größere Rewards erzielen lassen als bei einer zufälligen Initialisierung (orangene Linie).

Für uns interessanter ist allerdings die Verwendung für *Hierarchisches Reinforcement Learning* (HRL).

Nach [5] lernen HRL-Methoden eine Policy, welche aus mehreren Schichten besteht. Jede Ebene kontrolliert hierbei eine andere temporäre Abstraktionsebene. Dies ermöglicht es dem Agenten, nicht nur Basisoperationen auszuführen, sondern auch komplexere Aktionen wie zum Beispiel Sequenzen von Operationen einer niedrigeren Ebene.



(a) Versuchsaufbau (b) Lernerfolg der Algorithmen

Abbildung 6: Hürdenexperiment mit dem Half Cheetah (waagrecht: vergangene Zeit in Stunden, senkrecht: erhaltene Belohnung)

Quelle: [3]

Laut [3] eignet sich DIAYN hervorrangend als Grundstein für HRL. Für ein Experiment diesbezüglich betrachten wir wieder den Half Cheetah. Dieser hat nun die Aufgabe, über Hürden zu springen (siehe Abbildung 6a) und erhält Rewards für deren Überwinden.

Um die gelernten Fähigkeiten für HRL zu nutzen, wird DIAYN um einen *meta-controller* erweitert. Dieser kontrolliert, welche 10 Fähigkeiten jeweils als nächstes ausgeführt werden.

Das Experiment wird außerdem mit aktuellen Reinforcement Learning Algorithmen (VIME, TRPO und SAC) durchgeführt.

Wie die Abbildung 6b zeigt, schneiden diese im Experiment sehr schlecht ab und es gibt quasi keine merkliche Steigerung des Rewards. Im Gegensatz dazu zeigt der Ansatz mit dem DIAYN-Algorithmus schnell deutliche Forschritte und übertrifft merklich die anderen.

Aufgrund der Knappheit von externen Rewards gestaltet sich diese Aufgabe extrem schwierig für traditionelle RL-Algorithmen. Diese müssen quasi zufällig eine Hürde übersteigen, um eine Belohnung von außen zu bekommen.

Dies zeigt nach [3], dass das eigenständige Lernen von Fähigkeiten einen effektiven Mechanismus bereit stellt, um bei Herausforderungen beim Erkunden und mit geringem externen Reward Lernerfolge zu erzielen.

5 Curiosity und Diversity im Vergleich

Im Folgenden betrachten wir einige konzeptuelle Ähnlichkeiten zwischen den in Kapitel 3 und 4 betrachteten Ansätzen. Es bleibt zu beachten, dass sich beide Ansätze trotz ihrer praktischen Überschneidungen vor allem in der Philosophie ihrer Herangehensweise unterscheiden.

5.1 Zusammenhang von Komprimierung und vielfältigen Skills

Wie bereits in Kapitel 3 erläutert, tendiert ein neugieriger Agent in Schmidhubers Theorie dazu, Regularitäten zu komprimieren und zusammenzufassen. Darauf aufbauend versucht ein solcher Agent intern, den zu erwartenden *zukünftigen* Komprimierungsfortschritt zu maximieren. In Kombination mit der auf auf Neugierde basierenden Aktionsselektion, wie in Kapitel 3.2 beschrieben, führt ein Agent also tendenziell solche Aktionen aus, welche es dem Kompressor in Zukunft erlauben, die Historie des Agenten besser zu beschreiben. Auf Basis der Definition von subjektiver Interessantheit, wie sie in Kapitel 3.2 beschrieben ist, gilt im Umkehrschluss aber auch, dass sich der Agent von nicht mehr weiter komprimierbaren Beobachtungen abwendet und nach neuen Regelmäßigkeit kundschaftet, die als interessant erachtet werden können.

So erzeugt der Agent auf Dauer unterschiedliche Gruppierungen von Beobachtungen, welche sich nicht weiter zusammenfassen lassen. Distinkte *Symbole* unterscheiden sich so stark voneinander, dass sie keine gemeinsamen Regelmäßigkeiten aufweisen.

Besagte Beobachtungen enthalten bekanntlich Zustände, Aktionen und Belohnungen. In Kapitel 4.2 wird erläutert, dass die Unterscheidung der Fähigkeiten im DIAYN-Algorithmus nach deren besuchten Zuständen erfolgt. Nichtsdestotrotz handelt es sich um eine Abfolge von Aktionen, um besagte Zustände zu durchlaufen. Der in Kapitel 4 beschriebene wichtigste Grundsatz der erlernten Skills ist, dass diese möglichst vielfältig, also maximal unterschiedlich sind. Es lässt sich also argumentieren, dass die Komprimierung von Beobachtungen nach Schmidhuber einen ähnlichen Effekt hat wie das Erlernen von vielfältigen Fähigkeiten.

5.2 Unabhängigkeit von externen Belohnungen

Eine weit auffälligere Gemeinsamkeit der beiden Ansätze ist jedoch die Unabhängigkeit von externen Belohnungen.

Wie in Kapitel 4 und sogar im Titel von [3] ("Diversity is all you need: Learning skills without a reward function") bereits beschrieben findet das Erlernen von Fähigkeiten in der unüberwachten Phase ohne jeglichen externen Reward statt. Der DIAYN-Algorithmus erzeugt stattdessen interne Belohnungen anhand der Unterscheidbarkeit der betrachteten Fähigkeit und ist somit nicht auf Rückmeldung der Umgebung angewiesen.

Weniger intuitiv verhält es sich bei der Verwendung der erlernten Fähigkeiten. Wir betrachten hierfür deren Anwendung innerhalb HRL wie in Kapitel 4.3. Die Umgebung liefert im beschriebenen Versuchsaufbau nur spärlich Belohnungen. Bei der Betrachtung der Ergebnisse des Experiments in Abbildung 6b wird deutlich, dass die Benutzung der zuvor gelernten Fähigkeiten einen großen Vorteil gegenüber anderen, modernen RL-Algorithmen bringt. Während diese ihren Reward kaum merklich steigern können, vergrößert sich der des DIANY-Ansatzes sehr deutlich und überholt schnell alternative Methoden.

Auch auf Curiosity basierende Agenten sind in der Lage, in Umgebungen zu operieren, die kaum oder gar keine externen Belohnungen generieren. Ein Agent wendet sich in solch einer Situation schlicht der Maximierung seiner internen Belohnung $r_{int}(t)$ zu. In anderen Worten sucht ein solcher Agent in Ermangelung externer Belohnungen stets nach denjenigen Aktionen, die ihm am *interessantesten* erscheinen.

So lässt sich erkennen, dass beide Ansätze in Environments, in denen nur wenig externe Belohnungen vorhanden sind, wesentlich bessere Ergebnisse erzielen als herkömmliche RL-Algorithmen und somit eine adäquate Lösung für das *Sparse Reward Problem* finden. Dies ist ein wichtiger Punkt, da Rewards in den meisten realen Anwendungsgebieten nur spärlich oder sogar überhaupt nicht vorhanden sind.

6 Verwandte Arbeiten

Der Ansatz der Curiosity kommt überwiegend im Bereich des Reinforcement-Learning zum Einsatz. Das Konzept findet hier vielfältige Anwendung in Problemdomänen, in denen traditionelle Algorithmen aufgrund der spärlichen externen Belohnung nur mit einigen Schwierigkeiten agieren können. Beispielsweise nutzen Wesselmann, Wu und Gašić in [15] den Curiosity-Ansatz zur Dialoggenerierung, um mehr über den Nutzer zu erfahren. Ein ähnlicher Ansatz kommt in [10] zum Einsatz. Ziel des Reinforcement Algorithmus ist es hier, aussagekräftige Beschreibungen von Bildern aus möglichst unterschiedlichen Perspektiven zu generieren. Der Curiosity-Ansatz hilft laut Luo u. a. dabei, akkurate und zugleich unterschiedliche Texte zu einem gegebenen Bild zu erzeugen.

Es ist allerdings anzumerken, dass beide Ansätze ihren *Curisoty-Reward* auf Basis des *Prediction-Error* generieren. Ein solcher Ansatz wird von Schmidhuber auch in [12] beschrieben, allerdings kritisiert Schmidhuber diesen aufgrund einiger Mängel. Anders als der in Kapitel 3.2.1 beschriebene, auf dem Kompressor-Fortschritt basierende Ansatz, läuft der *Prediction-Error* basierte Ansatz Gefahr, stetig solche Aktionen auszuwählen, die zwar zu neuen, noch nicht gänzlich komprimierten Zuständen führen, welche allerdings nicht zwingend *interessant* sein müssen. [12]

Das Thema Diversity taucht nicht häufig in Zusammenhang mit RL Problemen auf. Jedoch nutzen Hong u. a. Diversity in [8] als Erkundungsstrategie für *Deep Reinforcement Learning*. Hierbei soll die modifizierte *Loss Function* den Agenten dazu anregen, sich von bisherigen Policies zu differenzieren, während er trotzdem optimal handelt. So wird erreicht, dass der Agent seine Umgebung effizient erkundet [8].

Viel eher wird Diversity allerdings nach [6] innerhalb von Literatur, die sich mit *Evolutionären Algorithmen* beschäftigen, behandelt. Die Anwendung auf diesem Gebiet ist wesentlich naheliegender, da hier eine vielfältige Population gewünscht ist. Diversity soll verhindern, dass lediglich ein lokales Optimum gefunden wird [6]. Hier lässt sich ein gewisser Zusammenhang zu [3] erkennen, da Diversity in beiden Fällen dazu eingesetzt wird, möglichst vielfältige Entitäten zu generieren. In [3] handelt es sich hierbei um die gelernten Fähigkeiten, bei Evolutionären Algorithmen sind es die Individuen einer Population, welche die Lösungskandidaten repräsentieren.

7 Schluss

In dieser Arbeit haben wir zwei erweiterte Ansätze des *Reinforcemnt Learning* vergleichend gegenübergestellt. Wir konnten zeigen, dass beide Konzepte, trotz ihrer theoretischen Differenzen, in gewissen Bereichen eine durchaus nicht unähnliche Funktionsweise aufweisen. Insbesondere findet sowohl das *Curiosity*-basierte Modell, als auch der *Diversity*-Ansatz eine adäquate Lösung für das in Kapitel 2.2 vorgestellte *Sparse Reward Problem*, indem beide nicht auf externe Belohnungen angewiesen sind.

Wir sehen in beiden Ansätzen großes Potential. Vor allem das noch nicht häufig thematisierte Vorgehen des Erlernens von Fähigkeiten mittels Diversity sollte weiter erforscht werden.

Autorenschaft

David Jonathan Müller hat die Abschnitte 2 und 4 verfasst. Lukas Johannes Rieger hat den Abschnitt 3 verfasst. Die Abschnitte 1, 5, 6 und 7 haben beide Autoren gemeinsam verfasst.

Literatur

[1] Thomas M Cover und Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[2] deeplizard.com, Hrsg. *Markov Decision Processes (MDPs) - Structuring A Reinforcement Learning Problem*. [Abgerufen am 22.06.2020]. Sep. 2018. URL: `https://deeplizard.com/learn/video/my207WNoeyA`.

[3] Benjamin Eysenbach u. a. "Diversity is all you need: Learning skills without a reward function". In: *arXiv preprint arXiv:1802.06070* (2018).

[4] Benjamin Eysenbach u. a. *Diversity is all you need: Learning skills without a reward function*. [Abgerufen am 20.06.2020]. URL: `https://sites.google.com/view/diayn/`.

[5] Yannis Flet-Berliac. "The Promise of Hierarchical Reinforcement Learning". In: *The Gradient* (2019).

[6] Thomas Gabor, Lenz Belzner und Claudia Linnhoff-Popien. "Inheritance-Based Diversity Measures for Explicit Convergence Control in Evolutionary Algorithms". In: *The Genetic and Evolutionary Computation Conference (GECCO)*. 2018. URL: `https://arxiv.org/pdf/1810.12470.pdf`.

[7] Joshua Hare. "Dealing with Sparse Rewards in Reinforcement Learning". In: *arXiv preprint arXiv:1910.09281* (2019).

[8] Zhang-Wei Hong u. a. "Diversity-driven exploration strategy for deep reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2018, S. 10489–10500.

[9] Leslie Pack Kaelbling, Michael L Littman und Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), S. 237–285.

[10] Yadan Luo u. a. *Curiosity-driven Reinforcement Learning for Diverse Visual Paragraph Generation*. 2019. arXiv: `1908.00169 [cs.CV]`.

[11] Nikolay Savinov u. a. "Episodic curiosity through reachability". In: *arXiv preprint arXiv:1810.02274* (2018).

[12] Juergen Schmidhuber. *Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes*. 2008. arXiv: `0812.4360 [cs.AI]`.

[13] R.J. Solomonoff. “A formal theory of inductive inference. Part I”. In: *Information and Control* 7.1 (1964), S. 1–22. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(64)90223-2. URL: http://www.sciencedirect.com/science/article/pii/S0019995864902232.

[14] Martin Werner. *Information und Codierung: Grundlagen und Anwendungen*. Springer-Verlag, 2009.

[15] P. Wesselmann, Y. Wu und M. Gašić. “Curiosity-driven Reinforcement Learning for Dialogue Management”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, S. 7210–7214.