|  |  |
| --- | --- |
|  |  |

# Wie die KI Spielen lerntE

## Eine kuriose Geschichte

Atari Spiele sind Klassiker und ziemlich einfach zu spielen. Doch es ist selbst für Menschen schwer, diese Spiele zu spielen, ohne sie je gesehen zu haben, geschweige denn das Ziel oder die Regeln zu kennen. Genau das kann jedoch mit künstlicher Intelligenz erreicht werden: Die Firma DeepMind hat es geschafft, mithilfe der Bellman Gleichung eine KI zu programmieren, die das Spiel nach einiger Zeit sogar besser spielt als der Mensch.



Abb 1: Atari Breakout wird von DeepMind gespielt

Wie die KI mithilfe der Bellmann Gleichung die beste Spielstrategie findet, wird im Laufe dieses Blogbeitrags erläutert. In diesem Blog möchte ich beschreiben, was dahintersteckt und zeigen, dass die Theorie eigentlich nicht allzu schwer ist. Auch hier gilt wie bei vielen Themen rund um die neuronalen Netze: Das Außergewöhnliche kommt durch das Handwerk.

Doch nun Schritt für Schritt zum **Deep Reinforcement Learning**:

## Die Interfaces

Zuerst muss das Programm in der Lage sein, mit dem Spiel zu kommunizieren. Alles was es dafür braucht ist der Bildschirm, der Punktestand und die möglichen Aktionen.

Das Programm hat also die Möglichkeit, irgendeine Aktion auszuführen und bekommt sowohl die Reaktion auf diese Aktion am Bildschirm als auch den neuen Punktestand zu sehen. Das sind auch schon die drei Komponenten, die Menschen und Maschinen zum Lernen brauchten:

### Aktionen (Action)

Ich kann aus einer Liste von möglichen Aktionen eine auswählen. Bei Atari Spielen sind das 8 verschiedene Richtungen jeweils mit gedrücktem oder nicht gedrücktem Joystick Button, der Button alleine ist eine weitere mögliche Aktion und nichts zu tun ist ebenfalls eine Aktion. Macht zusammen (2 \* 8 + 1 + 1) 18 verschiedene Aktionen. Die Aktionen werden von 0 bis 17 durchnummeriert und können dem Atari Spiel übermittelt werden.



Abb 2: Atari Joystick

### Status:

Jede Aktion, die man ausführt, bringt einen in eine neue Situation oder eben in einen neuen Status. Ist meine Aktion zum Beispiel, durch eine Tür zu gehen, ist mein neuer Status, dass ich nun vielleicht in der Küche bin oder ich stehe auf dem Bahnsteig oder in einem Kaufhaus. Bei Atari Spielen sieht man den neuen Status auf dem Bildschirm.

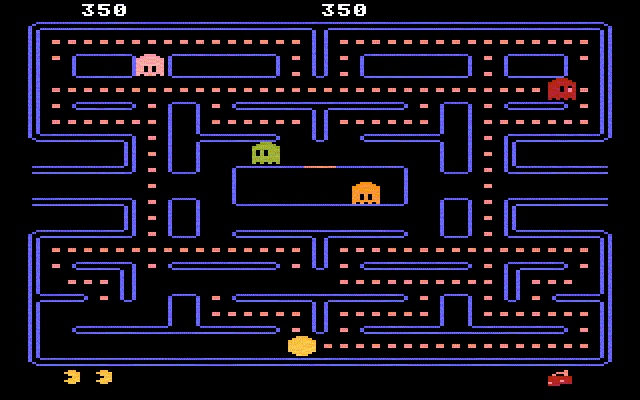


Abb 3: Ein Status mitten im Spiel

Die einzelnen Screens werden dem Deep Reinforcement Learning Programm als Bilder übermittelt. Die Bilder werden natürlich nicht mit einem Fully Connected Network gelesen, sondern mit einem Convolutional Network. Wenn Du wissen möchtest, wie das Funktioniert, kannst Du meine Blogs über Convolutional Networks lesen.

Wichtig ist hier, dass wirklich nur drei simple Convolutional Layer aufgeschichtet werden, ohne das die sonst üblichen Pooling Layer dazwischen gepackt werden. Pooling Layer schneiden gefundene Bildfragmente (Ein Kreis, eine Kante, ein Auto, …) frei und suchen sie auf dem gesamten Bild. Das ist hier natürlich eine dumme Idee, da es schon wichtig ist, wo welcher Gegenstand ganz genau zu finden ist.

Interessant ist übrigens auch, dass immer drei Bilder auf einmal durch das Netz geschickt werden, um Bewegungen mitzubekommen.

### Die Belohnung (**Reward**)

Seien wir ehrlich: Es sind die Belohnungen, die uns antreiben. Eine Eins unter der Klausur oder ein „Ich bin stolz auf dich“ von den Eltern.

Bei einem Computerspiel ist die Belohnung der neue Punktestand, der nach einer ausgeführten Aktion erzielt wurde.

## Das Ziel

Das Ziel beim Lernen ist es, die Belohnungen zu maximieren. Das wars eigentlich schon. Diesem Ziel liegt eine schlichte Annahme zu Grunde: Je besser ich bin, desto mehr Erfahrung habe ich, desto größer ist meine maximale Belohnung. Wenn ich also etwas gelernt habe, dann habe ich ein Gefühl dafür entwickelt, welche Entscheidungen ich in naher Zukunft treffen muss, um einen möglichst hohen Lohn zu bekommen. So eine Kette von Entscheidungen nennt man eine **Strategie (Policy)** und die beste Strategie beschert mir auf meinem Weg der Entscheidungen eine maximale Belohnung.

### Die beste Strategie

Die Frage, die nun noch bleibt, ist, wie findet man eigentlich die beste Strategie. Das ist ein sehr knifflige Frage und es hat gute 80 Jahre gedauert, bis sie beantwortet wurde. Der Anfang und das Ende dieser Reise werden von zwei Mathematikern gebildet.

**Andrei Andrejewitsch Markow** hat am Ende des neunzehnten Jahrhunderts damit angefangen Formalismen für Aktion Status Kombinationen zu erstellen, mit denen man Strategien ausdrücken kann. Wichtig ist hier eine ganz banal klingende Erkenntnis: Abgerechnet wird zum Schluss. Das bedeutet, dass ich auch mal eine ganze Reihe von sehr schlechten Einzelbelohnungen mit voller Absicht in Kauf nehme, um so zu einer riesigen Belohnung am Schluss zu kommen. Also: Ein Maß für Strategie ist die Summe der einzelnen Belohnungen.

Man kann auch schon mit Markow allein eine ganze Reihe von anspruchsvollen Aufgaben lösen, solange man alle Zustand Aktion Kombinationen aufschreiben kann. Man kann sich vorstellen, dass das bei Computerspielen allerdings schwer wird, da es eine extrem große Menge von Zuständen gibt. Man müsste also jeden möglichen Screenshot mit allen möglichen Aktionen kombinieren, um die Strategie maximieren zu können. Man kommt schon bei sehr einfachen Spielen in Situationen, wo unser Universum zu klein wäre, um alle möglichen Kombinationen aufzuschreiben.

In der zweiten Hälfte des zwanzigsten Jahrhunderts hat der Amerikanische Mathematiker **Richard Bellman** dieses Problem dann gelöst und ein Maß für Qualität eingeführt. Die sogenannte **Bellman Gleichung** ist der Dreh und Angelpunkt für das Deep Reinforce Learning und nachdem es hier um Qualität geht, bezeichnet man das Ganze als **Q – Learning**

### Q – Learning

Wenn man die Bellman Gleichung verstehen möchte, stellt man sich am besten vor, man hätte schon eine perfekte Formel, die einem für die Situation (den Status) in der ich mich gerade befinde unter allen möglichen Aktionen die vielversprechendste nennt: Q(s,a)

Ich bin also in dem Status s und muss mich für eine bestimmte Aktion a entscheiden. Dann frage ich die Funktion Q für alle möglichen Aktionen nach ihrem Wert und entscheide mich für das größte Ergebnis.

Natürlich hat man die Funktion Q nicht sofort, aber man kann sich ihr annähern, also immer ein bisschen besser werden und das funktioniert indem ich quasi Briefe in meine Vergangenheit schreibe.

Jedes Mal, wenn ich von einem Status über eine Aktion in einen nächsten gegangen bin füge ich der Bellman Gleichung für den alten Zustand die Information hinzu: Wenn du aus deinem Zustand in meinen gehst, bekommst du eine Belohnung von so und so viel. Damit ist es möglich sozusagen eine Spur der gemachten Erfahrungen hinter mir her zu ziehen. Als Gleichung sieht das so aus:

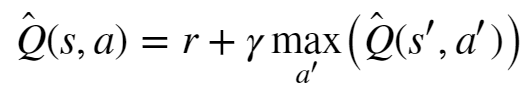


Abb 4: Bellman Gleichung

* Das Q hat ein Dach um anzuzeigen, dass man sich in die perfekte Lösung hineiniteriert.
* „r“ ist die Belohnung (reward), die ich bekomme, wenn ich in der Situation s die Aktion a ausführe
* „s‘ “ ist die Situation in die ich komme, wenn ich in „s“ die Aktion „a“ ausführe
* „a‘ “ sind die Aktionen, die ich in „s‘ “ ausführen kann.
* „max“ heißt, dass ich von allen Aktionen „a‘ “, die ich in „s‘ “ ausführen kann, die mit dem größten Wert (die vielversprechendste) nehme
* „γ“ ist ein Gewichtungsfaktor zwischen Null und Eins [0,1]. Den brauche ich, um die ungewissen Entscheidungen der Zukunft entsprechend abzuschwächen. Ich weiß ja nicht wirklich, ob ich in der Zukunft immer die Belohnung bekomme, die ich in der gleichen Situation schon Mal bekommen habe. Also würde ich eine 1 wählen, wenn ich mich in einer extrem starren Situation wie zum Beispiel einem statischen Irrgarten befinde. Das andere Extrem die 0 würde ich nehmen, wenn die Zukunft extrem ungewiss ist.

Die Formel mag auf den ersten Blick kompliziert aussehen und knapp hundert Jahre gebraucht haben, bis sie fertig war, wenn man sie aber mit der entsprechenden Vorstellungswelt unterlegt, ist sie doch sehr einfach zu begreifen.

### Q – Learning in neuronalen Netzen

Bis hier her brauche ich noch keine neuronalen Netze. Die Bellman Gleichung könnte ich auch einfach so im Speicher aufbauen, wenn ich jeden einzelnen Status und jede einzelne Aktion kenne (diskreter Zustand). Wie aber bereits angemerkt habe ich diesen Zustand in Computerspielen nicht. Und das aus zwei verschiedenen Gründen:

* Es gibt wirklich viele mögliche Screenshots, die ich machen kann, bis ich wirklich jeden möglichen Screenshot gesehen habe. Nicht vergessen: Ein Screenshot entspricht einem diskreten Zustand, in den ich durch eine diskrete Aktion gekommen bin
* Es gibt in einem Computerspiel sehr viele Situation, die so ähnlich wie ganz viele Situation an anderer Stelle sind. In Pac Man zum Beispiel gibt es an unterschiedlichsten Positionen für Pac Man die gleiche Strategie um zu überleben. Es ist also ergonomisch, wenn man diese ähnlichen Situationen auch ähnlich behandelt.

Genau diese zwei Gründe schreien für die Approximation der Bellman Gleichung nach einem ganz schlichten Fully Connected neuronalen Netz, denn das ist genau das was FCNN nun Mal machen. Sie gruppieren ähnliche Dinge. Also packt man hinter die Convolutional Layers, die man braucht, um die Screenshots besser zu erkennen noch einige Fully Connected Layers um sich zu einem Status, den man links ins Netz reinschiebt rechts einen Wichtungsvektor mit allen möglichen Aktionen herausgeben zu lassen.

Max Value

Status s

Convolutional Layers

Fully Connected Layers

**a1 = 12**

a2 = -3

a3 = 5

a4 = 16

an = …

Abb 5: Deep Reinforced Network

In den Trainings – Phasen merke ich mir also bei jedem Schritt den Status s wo ich herkam, die Aktion a, die ich ausführte, die neue Situation s‘ und die Belohnung r, die ich beim Übergang von s nach s‘ bekommen habe. Mit diesen Werten update ich die Bellman Gleichung für Q(s,a) und trainiere das Netz mit diesen Werten neu.

In der Theorie ist es das eigentlich schon. Die Praxis zeigt aber, dass es noch etliche handwerkliche Kniffe gibt, die das Netz in die Konvergenz zwingen. Und man kann viele sehr frustrierende Stunden und Tage damit verbringen, eine Metadatenkonfiguration nach der anderen auszuprobieren ohne ein anständiges Netz zu bekommen.

## Ausblick

In meinem nächsten Blog werde ich ein sauber konvergierendes Deep Reinforcement Netz vorstellen, das in der Lage ist, ein einfaches Computerspiel zu erlernen und zu spielen. Mit dem Wissen aus diesem Blog sollte es dann einfach sein, sich seine eigenen „Werkstücke“ in die KI Werkbank einzuspannen und ebenfalls Deep Reinforcement Netze zu bauen.

# Schlagwörter

Markov, Bellman, Q-Learning, Deep Reinforcement Networks, Atari, Reinforced Learning