Skript Reinforcement Learning
KI Video

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Einleitung

(Cooles Deckblatt anzeigen)

Habt ihr euch jemals gefragt, wie ein selbstfahrendes Auto lernt sich fortzubewegen oder wie Schachprogramme ihre eigene Strategie mit der Zeit verbessern? Das ist die beeindruckende Fähigkeit des Reinforcement Learnings, einem Teilgebiet der künstlichen Intelligenz. Der entscheidende Vorteil dieser Variante ist, dass es dadurch möglich wird, Systeme zu entwerfen, die sich durch die Interaktion mit ihrem Umfeld kontinuierlich verbessern und sich so an verschiedenste Szenarien anpassen können. Doch wie lässt sich so etwas umsetzen?
(Ansicht ähnlich wie in der Folie mit dem Spielfeld und der Spielfigur)

1.1 Environment

Um euch Schritt für Schritt zu erklären, wie alles funktioniert, werden wir uns an einem Beispiel orientieren. Hier unser Szenario:
Alles dreht sich um den sogenannten „agent“. Stellt euch das so vor wie eine Spielfigur in einem Videospiel. In diesem Szenario folgen wir einem Elfen, der versucht, ein Geschenk zu erreichen. Dafür muss er sich über einen zugefrorenen See bewegen, indem er sich wie auf einem Schachbrett von Feld zu Feld bewegt. Jeder Schritt ist eine sogenannte „action“ Bei jeder „action“ verändert sich der Zustand des Szenarios. Den nennt man „state“. Wichtig ist, dass der „agent“ typischerweise kein Vorwissen über die Umgebung hat. . Dabei muss er nicht nur den richtigen Weg finden und die Löcher in der Eisdecke umgehen, in dem er die richtige Abfolge von Schritten in die richtige Richtung macht, sondern auch damit umgehen, dass die Eisdecke rutschig ist. Es kann also passieren, dass er statt in die gewählte Richtung auch in eine andere Richtung rutscht. (visuelle Darstellung 3 Pfeile, jeweils ein Drittel). Diese Definition von Übergängen zwischen den States nennt man „transition function“. (Fachbegriff anzeigen?)

1.2 Transition- und Reward-Function

Zuletzt bleibt noch die sogenannte „Reward-Function“. Damit unser „agent“ weiß, welchen Übergang er am besten verwenden soll, wird jedem „state“ in unserem Szenario ein positiver oder negativer „reward“ zugeordnet, also entweder eine Belohnung oder eine Bestrafung, je nachdem, ob der „state“ hilfreich oder schädlich für das zu erreichende Ziel ist. In unserem Szenario erhält der Elf dann eine Belohnung, wenn er es geschafft hat, das Geschenk zu erreichen, ansonsten geht er leer aus. (visuell 0en und 1en?) zusammengenommen bezeichnet man diese Art von Problemstellung als Markov Decision Process (MDP) nach Andrej Markov (Bild?). Nun ist es die Aufgabe des „agents“ nach diesen Gegebenheiten eine optimale „policy“ zu entwerfen. Aber was ist eine Policy? Über eine Policy erlernt der „agent“ welche „action“ in einem „state“ am besten zu wählen ist. Die Besonderheit von „Reinforcement Learning“ Problemen ist, dass dem „agent“ weder die „Reward-Function“, noch die „Transition-Function“ bekannt ist. Also muss er durch die Interaktion mit seiner Umgebung lernen. Dabei befindet er sich nach jedem Schritt in einem neuen „state“ und erhält dazu passend einen „reward“. Um seinen „reward“ zu maximieren, arbeitet der „agent“ eine immer bessere „policy“ aus. (Schritte machen und reward anzeigen oder rote und grüne Pfeile)

1.3 Q-Tabelle

Unser Elf arbeitet dafür mit einer sogenannten Q-Tabelle. Die ist praktisch das Gehirn des Elfs. Jede Zeile steht hier für ein Feld in unserer Welt. Dort wird die Bewertung von jedem Feld nach dem Lernprozess gespeichert und so kann der Elf je nach Wert entscheiden, welcher Zug der beste in der aktuellen Situation ist.

2 Training

2.1 Training-Funktionalität

Schauen wir uns jetzt einmal an, wie ein solcher Trainingsdurchlauf funktioniert: 2 Zu Beginn des Trainings gibt es in der Q-Tabelle noch keinerlei Präferenzen für die Züge, weil „agent“ noch kein Wissen über die Umgebung und seine besten „actions“ hat. Er verhält sich daher zunächst zufällig, da er so möglichst viele Informationen über die „rewards“ der jeweiligen Situation sammeln kann. Dieser Ansatz wird als „active learning“ bezeichnet, da der „agent“ aktiv Entscheidungen trifft und dann basierend auf den „rewards“ lernt. Alternativ kann man auch „passive learning“ verwenden. Dabei fungiert der „agent“ eher als Beobachter und verhält sich nach einer vorgegebenen „policy“, um Informationen über die Umgebung zu sammeln. (vorher/nachher Clip des ausgeführten Modells?)

2.2 Exploration vs. Exploitation (Fachbegriffe vielleicht einblenden)

Im Training für Reinforcement Learning Agenten gibt es ein fundamentales Dilemma: Der Exploration vs. Exploitation tradeoff. Exploration bezeichnet dabei das Erkunden der Umgebung unabhängig von der Policy, damit bessere Ansätze auch dann gefunden werden, wenn sie stark von den aktuellen Daten abweichen. Wendet man Exploitation an, so folgt man streng der aktuell besten policy, da das der beste aktuell bekannte Weg ist. Beides ist notwendig, da Exploitation das Optimum möglicherweise verfehlt und Exploration zu ungerichtet ist, um in sinnvoller Zeit Ergebnisse zu erzielen. Hier wird dieses Problem durch die „exploration decay rate“ gelöst. Diese legt die Wahrscheinlichkeit fest mit der der Agent von der aktuellen policy abweicht um Exploration anzuwenden. Diese wird mit der Zeit immer kleiner, sodass man sich immer mehr auf das Gefundene verlässt.

2.3 Bewertungsfunktion im Training (Formel einblenden und nach und nach beschriften

Um die Q-Tabelle im Training zu bearbeiten, wird eine Bewertungsfunktion eingesetzt, wir verwenden hier eine leicht abgewandelte Bellman-Optimality-Equation. Man versucht also, die Tabelle immer weiter so zu verbessern, dass sie sich der optimalen Lösung der Gleichung annähert. Dafür berechnet man je nach Belohnung für den aktuellen Zug eine Bewertung mit der man die Q-Tabelle updatet. In dieser Gleichung gibt es mehrere Faktoren die man für ein optimales Training passend wählen muss: Die Learning Rate steuert wie stark eine neu gelernte Information die aktuelle Bewertung beeinflusst, also wie stark sich die Werte nach einem Durchlauf ändern. Der reward ist die Bewertung der aktuellen Entscheidung. Da man aber nicht nur den aktuellen reward betrachten will, sondern auch welche rewards man als Folge der aktuellen Entscheidung in Zukunft erreichen kann, braucht man die sogenannte discount-rate, die den Einfluss von weiter in der Zukunft liegenden rewards etwas abschwächt, damit die lokal gesehen besten Entscheidungen getroffen werden. (Gleichung einblenden und nach und nach beschriften)

2.4 Abschlusssatz zu Neural-Networks

Abschließend wollen wir noch erwähnen, dass es sich bei unserem Ansatz mit der Q-Tabelle natürlich nur um einen einzelnen Ansatz in einem großen Feld handelt. Man kann natürlich viele andere Varianten, wie beispielsweise neurale Netze, die die Neuronen-Struktur eines Gehirns nachahmen sollen, verwenden und vieles mehr. Das alles würde aber den Rahmen dieses kurzen Videos sprengen. Wir hoffen, dass wir einen guten Einblick geben und für die nächste

Begegnung mit Reinforcement Learning ein wenig Licht ins Dunkel bringen konnten und laden dazu ein, sich bei Interesse den beigelegten Code für das gezeigte Szenario anzuschauen. Vielen Dank fürs Zuschauen!