NAVIGATE (AGENT MAULWURF)

Sehen, hören, tasten: Menschen finden sich in neuen Umgebungen gut zurecht. Kann das auch eine Maschine?
In einer Simulation schicken wir eine künstliche Intelligenz (KI) auf Erkundungsmission:
Mit verschiedenen Trainingsprotokollen und Belohnungssystemen trainieren wir „Agent Maulwurf“. Ein neuronales Netzwerk verarbeitet die Trainingsprotokolle und Belohnungssysteme (Deep Reinforcement Learning). Er lernt, Entscheidungen zu treffen und findet sich in fremdem Terrain zurecht.

ZIEL

Entwicklung einer Künstlichen Intelligenz, die sich autonom durch ein Tunnelsystem bewegen kann. Dabei soll sie zu einem Ziel navigieren, ohne mit der Umgebung zu kollidieren.

UMSETZUNG

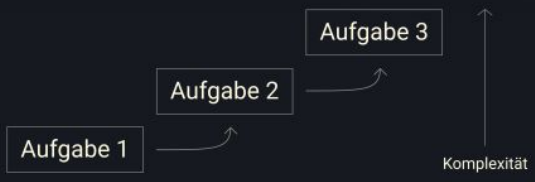
Unity Machine Learning Agents

Als Entwicklungsumgebung nutzen wir Unity in Verbindung mit dem hauseigenen Toolkit “ML-Agents, welches sich derzeit noch in der Beta-Phase befindet. ML-Agents basiert auf der Machine-Learning Library “TensorFlow” mit C# als Programmiersprache für die Schnittstelle. Über benutzerdefinierte Konfigurationen erstellt TensorFlow ein künstliches neuronales Netzwerk mit welchem der “Agent” trainiert werden kann.

Angewandte Methoden

Curriculum Learning

Komplexe Aufgaben können in mehreren Phasen trainiert werden. Wenn eine einfachere Teilaufgabe erledigt wurde, erhält der Agent eine neue, schwierigere Aufgabe.



Curiosity-Driven Learning

Der Agent erhält keine festgelegten Rewards für bestimmte Aktionen sondern bekommt einen intrinsischen Reward wenn etwas Unvorhergesehenes passiert. Somit wird er „Neugierig“ und erkundet ohne, von uns festgelegte, extrinsische Belohnungen die Umgebung.

Curiosity-Driven Learning

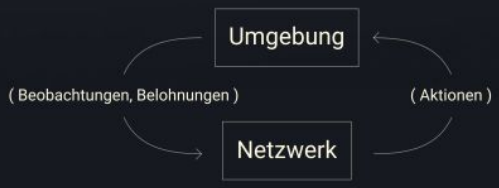
Das Netzwerk hat ein „Gedächtnis“ in welchem es über kurze Zeit gesammelte Informationen für einen längeren Zeitraum speichern kann. Dadurch kann bei gleichen Inputs die Erzeugung sinnvoller Outputs beschleunigt werden.

Imitation Learning

Das Netzwerk erhält als Beobachtungen die von einem Spieler ausgeführten Aktionen und erlernt anhand dessen Verhalten die Aufgabe selbstständig zu lösen.

Deep Reinforcement Learning

Einem Netzwerk werden Inputwerte in Form von Beobachtungen der Umgebung und Belohnungen übergeben. Dieses erzeugt daraus Outputwerte, welche als Aktionen gedeutet werden.
Nachdem die Aktionen (z.B. Vorwärtsbewegung) ausgeführt wurden, erhält das Netzwerk die neuen Beobachtungen und Belohnungen als Bewertung



Mit diesem Feedback erlernt und optimiert das Netzwerk eine Strategie (“policy”) um für die kommenden Inputwerte möglichst sinnvolle Outputs zu erzeugen.

Beobachtungen

Es werden entweder Entfernungen zu umliegenden Objekten / Wänden übergeben (Vector Observations) oder die Bilder einer virtuellen Kamera, welche die Umgebung aufzeichnet (Visual Observations).

Belohnungen

Die vom Netzwerk als Output zurückgegebenen und dann vom Agent ausgeführten Aktionen werden bewertet. Diese Bewertung erfolgt in Form von Belohnungen (Rewards), die positiv oder negativ sein können.

Aktionen

Der Agent kann sich auf zwei bzw. drei Achsen bewegen:
Vorwärts/Rückwärts, Links/Rechts, Hoch/Runter

ERGEBNIS

Zur Bewertung der einzelnen Methoden können nun folgende Schlüsse gezogen werden:

Vector Observations sind einfach zu verwenden und benötigen wenig Rechenleistung. Sie können sehr effektiv zur Erzeugung elementarer Beobachtungen der Umgebung genutzt werden.

Visual Observations können genutzt werden um optische Informationen der Umgebung zu erhalten, sie benötigen aber viel Rechenleistung.

Curriculum Learning eignet sich sehr gut, um einem Agent schrittweise komplexere Aufgaben beizubringen.

Curiosity Learning eignet sich besonders bei Aufgaben bei denen die Modellierung von Belohnungssystemen sehr schwierig ist.

Recurrent Neural Networks sind sehr ressourcenintensiv, sind aber bei Aufgaben, bei denen bestimmte Informationen zu späteren Zeitpunkten benötigt werden, unentbehrlich.

Imitation Learning ist geeignet für „monotone“ Aufgaben, für Abstraktionen nicht zu empfehlen.

Das Spiel

Ein mit Vector Observations trainiertes Modell ist nun in der Lage einen Agent auch durch unbekannte Umgebungen zu navigieren.

Um unsere Ergebnisse zu veranschaulichen und die Leistungsfähigkeit eines trainierten Agents darzustellen haben wir ein Spiel entwickelt. Ziel ist es, vor dem Agent das Ende eines Tunnels zu erreichen. Dabei werden beide Kontrahenten bei Kontakt mit der Wand oder anderen Hindernissen verlangsamt. Genauso wie dem Spieler ist die zu bewältigende Strecke dem Agent unbekannt.

Der Agent schlägt sich dann besonders gut gegen einen menschlichen Kontrahenten, wenn dieser an die Grenzen seiner kognitiven und physischen Fähigkeiten stößt. Wenn die maximale Bewegungsgeschwindigkeit im Spiel höher ist, wird es für den Menschen schwieriger den Agent zu besiegen.