Aufgabe 1 Zeitstrahl | Constantin Rech DSA 8028907

2000: REINFORCE: MC Policy Gradient

Ziel:

Das Ziel des Reinforcement Learning ist es, eine optimale Verhaltensstrategie für den Agenten zu finden, um optimale Belohnungen zu erhalten

Funktion:

Berechnung des Gradienten der vorhergesagten kumulativen Belohnung in Bezug auf die Policy-Parameter mit Hilfe des Likelihood-Ratio-Tricks (Nutzung von Monte Carlo Techniken)

Anwendung:

Verschiedene Aufgaben des Reinforcement Learning, z. B. Spiele und Robotik, …

2000: Vanilla policy gradient

Ziel:

Ziel der policy gradients ist es, die Wahrscheinlichkeiten von Handlungen, die zu einem höheren Return führen, zu erhöhen und die Wahrscheinlichkeiten von Handlungen, die zu einem niedrigeren Return führen, zu verringern, bis man zur optimalen Policy gelangt.

Funktion:

Verwendet zwei Netzwerke: Eines zum Lernen der Politik und eines zum Lernen der Zustandswertfunktion. Verwendet letztere, um dann die Vorteilsfunktion zu schätzen (nicht akteurskritisch, da kein Bootstrapping)

Anwendung:

Bereiche, in denen Reinforcement Learning anwendbar ist.

Es wurde bereits erfolgreich bei Aufgaben wie Robotersteuerung, Computerspielen, autonomem Fahren und vielen anderen eingesetzt

2005: Neural fitted Q iteration (NFQ)

Ziel:

NFQ zielt darauf ab, die Q-Funktion als value function zu approximieren

Funktion:

NFQ verwendet neuronale Netze als Funktionsapproximatoren.

Es handelt sich dabei um eine spezifische Konfiguration des value-based Deep-RL-Algorithmen building Prozesses.

Anwendung:

Aufgaben des Reinforcement Learnings, bei denen der State-Action-Space zu groß für das standardmäßige Q-learning ist und eine Funktionsapproximation erforderlich ist.

2015: Deep Q-Network (DQN)

Ziel:

Optimierungsmethoden wurden auf der Grundlage der IID-Verteilung entwickelt, die beim Online-Lernen nicht gegeben ist. Das DQN geht dieses Problem an

Funktion:

Lösung des Problems durch die Einführung eines separaten Netzes für Zielaktualisierungen, des Zielnetzes. Dies stabilisiert das Training, verlangsamt aber das Lernen.

Anwendung:

Einsatz bei einer Vielzahl von Aufgaben, einschließlich Spielen in der Atari-Domain. Außerdem wird DQN noch in anderen komplexen Umgebungen eingesetzt.

2015: Double Deep Q-Network (DDQN)

Ziel:

Vermeidung der Maximierungsverzerrungen der Q-Learning Agenten und des DQN, indem die Aktualisierungen von verzerrten Schätzungen getrennt werden

Funktion:

Verwendung zweier separater Q-Wert-Schätzer, von denen jeder zur Aktualisierung des anderen verwendet wird. Mithilfe dieser unabhängigen Schätzer können unverzerrte Q-Wert-Schätzungen der Aktionen vorgenommen werden, die mit dem anderen Schätzer ausgewählt wurden

Anwendung:

Wird bei Aufgaben verwendet bei denen eine Überschätzung ein Problem darstellen kann

2015: Dueling Double Deep Q-Network (Dueling DDQN)

Ziel:

Optimierung des DDQN Modells durch zwei getrennte Schätzer für die state-value- und die action-advantage-Funktion, die sich interne Knotenpunkte teilen müssen

Funktion:

Das Duelling DDQN hat zwei getrennte Schätzer für die State-Value- und die Action-Advantage-Funktion, die sich interne Knotenpunkte teilen müssen. Die beiden Schätzungen werden in einer Ausgabe aggregiert, die die Q-Funktionsschätzung enthält.

Anwendung:

Wird bei Aufgaben verwendet, bei denen es wichtig ist, den Wert verschiedener Handlungen in jedem Zustand zu unterscheiden, und bei denen eine Überschätzung ein Problem darstellen kann.

2015: Prioritized experience replay (PER)

Ziel:

Verbesserung der Effizienz des Lernens aus vergangenen Erfahrungen beim Reinforcement Learning

Funktion:

Es priorisiert die Wiederholung vergangener Erlebnisse auf der Grundlage des Ausmaßes ihres TD-Fehlers. Es gibt verschiedene Methoden dies zu tun.

- Proportional prioritization
- Rank-based prioritization
- Weighted importance sampling

Anwendung:

Wird in Verbindung mit verschiedenen Algorithmen des Reinforcement Learning verwendet, um deren Effizienz

2015: Generalized advantage estimation (GAE)

Ziel:

Die Varianz weiter reduzieren.

Funktion:

Nutzung einer Methode besser als das n-schrittige Bootstrapping:
Das λ-Target

Verwendet die λ-Target-Analogie zum Senken der Varianz durch Hinzufügen einer Verzerrung (Bias)

Anwendung:

3D locomotion tasks

Erlernen von Laufschritten für simulierte zwei- und vierbeinige Roboter

Erlernen einer Strategie, die den Zweibeiner dazu bringt, vom Boden aus aufzustehen

2015: Deep deterministic policy gradient (DDPG)

Ziel:

Das Maximieren des erwartete kumulativen langfristigen Reward, durch einen actor-critic Reinforcement Learning Agent, der nach einer optimalen Strategie sucht

Funktion:

Lernt gleichzeitig eine Q-Funktion und eine Policy. Es verwendet off-Policy Daten und die Bellman-Gleichung, um die Q-Funktion zu lernen, und verwendet die Q-Funktion, um die Policy zu lernen

Anwendung:

Einsatz bei Aufgaben mit kontinuierlichen Aktionsräumen, wie z. B. robotic control tasks

2016: Advantage actor-critic (A2C)

Ziel:

Verbesserung des A3C Algorithmusses -> Die Varianz weiter reduzieren, sowie Verbesserung der Stabilität und Effizienz

Funktion:

synchrone Version von A3C -- Änderungen:

- Single NN für value Funktion and policy
- Mehrere Arbeiter, ein Lerner

Anwendung:

- Motor Kontroll Probleme
- Navigation durch zufällig generierte 3D mazes
- Einsatz in Atari domain

2016: Asynchronous advantage actor-critic (A3C)

Ziel:

Die Varianz weiter reduzieren, sowie Verbesserung der Stabilität und Effizienz

Funktion:

Um Variance weiter zu verringern verwendet der Algorithmus n-step bootstrapping returns und concurrent agents, um mehr Erfahrungsproben zu sammeln

Nutzt einen style names Hogwild

Anwendung:

Anwendungsbereiche sind Gameplay und Robotik.

Heutzutage wird RL für alltägliche Aufgaben wie Planung, Navigation, Optimierung und Szenariosimulation in verschiedenen vertikalen Ketten eingesetzt

2017: Proximal policy optimization (PPO)

Ziel:

Dieselbe zugrundeliegende Architektur wie A2C mit mehreren Umgebungen, Mini-Batches, Kritik für GAE-Schätzungen, ähnliches Training

-> Verbesserung der Stichprobeneffizienz und der Stabilität von Policy-Gradienten-Methoden

Funktion:

PPO verfügt über eine Ersatz-Zielfunktion, die das Verhältnis zwischen der neuen und der alten Policy festlegt, um große Policy-Updates zu verhindern, was mehrere Optimierungsschritte pro Mini-Batch ermöglicht.

Anwendung:

Zurzeit einer der populärsten Algorithmen in diesem Bereich, da er ein ausgewogenes Verhältnis zwischen der Effizienz der Stichproben und der Rechenleistung aufweist.

2018: Soft actor-critic (SAC)

Ziel:

Algorithmus, der eine stochastische Policy auf eine Off-Policy-Art optimiert

Funktion:

Beinhaltet den Clipped-Double-Q-Trick, und aufgrund der inhärenten Stochastizität der Policy profitiert er auch von einer Art Glättung der Target Policy

Das zentrale Merkmal von SAC ist die Entropie-Regularisierung. Die Policy wird so trainiert, dass ein Kompromiss zwischen erwarteter Return und Entropie, einem Maß für die Zufälligkeit der Policy, maximiert wird.

Anwendung:

Wird bei Aufgaben mit kontinuierlichen Aktionsräumen verwendet, insbesondere wenn Exploration und Robustheit wichtig sind.

Bspw. robotic motion tasks

2018: Twin-delayed DDPG (TD3)

Ziel:

Eine häufige Fehlerart bei DDPG ist, dass die gelernte Q-Funktion beginnt, die Q-Werte drastisch zu überschätzen, was dann zum Zusammenbruch der Policy führt,Twin Delayed DDPG (TD3) ist ein Algorithmus, der dieses Problem durch die Einführung von drei entscheidenden Tricks behebt

Funktion:

Fügt doppeltes Lernen hinzu ("Twin")

Verzögert die Aktualisierungen von Policy-Netz, Zielnetz und Zwillings-Zielnetz ("delayed")

Fügt nicht nur der Ausgangsaktion, sondern auch den Zielaktionen Rauschen hinzu

Anwendung:

Genau wie bei DDPG:

Einsatz bei Aufgaben mit kontinuierlichen Aktionsräumen, wie z. B. robotic control tasks