Yapay Zeka

Pekiştirmeli Öğrenme ile Labirentten Çıkış

Mustafa Katipoğlu

16011084

Pekiştirmeli Öğrenme

Pekiştirmeli Öğrenme :

Pekiştirmeli öğrenmede ajan hatalarından öğrenerek kendini geliştirmesi sonucu zeki bir şekilde davranır.

Ajan'ın yaptığı her hareket sonucu çevresinden bir tepki, durum değişimi alır. Bu tepkinin iyi-kötü olmasına göre ajan farklı ödüller alır. Kötü ödüller ajanın yapılan hareketten uzak durması yönünde mesaj verirken iyi ödüller ajanın doğru sonuca ilerlediğini gösterir. Ajan topladığı ödülleri maximize ederek eğitim süresinde kendini gelitştirir ve sonrasında eğitilmiş olan bu ajan zeki bir hareket sergiler.

Pekiştirmeli Öğrenme Çeşitleri :

Pekiştirmeli öğrenmenin bir çok çeşidi vardır. Bu projede üzerinde durulan pekiştirmeli öğrenme çeşitleri şunlardır:

1- Q Learning
2- SARSA
3- Deep Q Learning

1- Q Learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Q Learning metodunda, olası durum sayısı satırlı ve olası hamle sayısı sütünlu bir matris tutuyoruz ve yukardaki formül ile, her durum-hareket ikilisi sonucu bu tablodaki değerleri güncelliyoruz. Eğitim süresi sonunda tablodaki değerler optimum seviyeye ulaşması sonucu (muhtemelen) verilen görevi başarıyla yerine getirir.

Burada Gamma, gelecek hamlenin doğruluğuna verdiğimiz önem, alpha öğrenme oranı, ve algoritmada kullandığımız epsilon rastgele mi hamle yapılıcak yoksa Q tablosundaki değerlere göre mi hamle yapılıcak olduğuna karar veren hiperparametrelerimizdir.

2- SARSA(State Action Reward State)

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

SARSA, Q learning metoduna çok benzer olup tek farkı bir sonraki hamlenin en iyi durumunu ele almak yerine bir sonraki hamleyi oynuyup seçilicek olan hareket sonucu oluşan durum-hareket ikilisini ele alır.

3- Deep Q Learning

Deep Q Learning, derin nörön ağlarını kullanan bir yöntemdir ve Q-Learning de olduğu gibi hafızada matris tutmak yerine, yapay sinir ağı içerisinde ağırlık olarak gerekli verileri tutar.

Geliştirme Sürecinde Yaşananlar

1- Scale – Labirentin Büyümesi

Her ne kadar Q-Learning metodu 10x10, 20x20, 30x30 gibi küçük ölçekli labirentlerde çok başarılı sonuçları çok hızlı sürede ortaya çıkartabilsede, labirent boyutları 40x40 ve üzeri çıkmaya başladığı zaman eğitim süreleri birden onlarca kat artmaya ve hatta bazen tam anlamıyla başarılı sonuçlar üretememeye başladı. Bu sorunu çözmek için hiperparametreleri rastgele belirlemek yerine optimizasyonu yoluna gittim.

2- Hiperparametre Optimizasyonu

Hiperparametre optimizasyonu için Genetik Algoritma kullanarak en optimum hiperparametreleri bulmaya çalıştım. Optimizasyon süresinde en büyük problem çok uzun çalışma süreleri oldu. Uzun çalışma süreleri sonucu iyi olduğunu düşündüğüm hiperparametre değerleri elde etsem dahi bu değerler labirentin çalışmasında beklediğim derecede büyük bir iyileşme göstermedi.

Bazen 30x30 yerine 40x40 ları çözer hale geldi fakat halen daha büyük labirentler çözmede başarılı olma konusu çok büyük bir sorundu.

Burada hiperparametre optimizasyonu çok uzun süre aldığı için küçük labirentler üzerinde deniyerek optimum değerler elde ettim. Fakat bu değerler büyük labirentlerde beklediğim başarıyı göstermedi.

3- Uzun Eğitim Süreleri

Büyük labirentleri çözmede Q-Learning algoritmasının çok uzun zaman alması sonucu, bu algoritmanın yeterince iyi olmadığı düşüncesiyle farklı algoritmalara yöneldim.

Burada ilk algoritma olan SARSA algoritmasını denedim fakat Q-Learning e göre iyi sonuçlar elde etmekden ziyada daha uzun süreler çalıştı ve daha başarısız sonuçlar elde ettim.

Daha sonrasında, her ne kadar iç çalışma prensibini tam olarak bilmesemde, hazır kodları kullanarak büyük labirentleri Deep Q-Learning metodu ile çalıştırmayı denedim. Burada çalışma süreleri ve başarı oranları çok garip bir şekilde daha kötü bir şekilde gerçekleşti.

8x8 labirentlerin çözümü dahi 2dk üzerinde zaman gerektiriyordu ve 100x100 matrisin sonucunu saatler geçmesine karşın elde etmem mümkün olmadı. Halbuki Q-Learning algoritması ile 20x20, 30x30 labirentler dahi 1-2dk süre arasında başarıyla eğitilebiliyordu.

Sistemin Başarısının Değerlendirilmesi

Farklı algoritmaların başarılarının değerlendirilmesinde en büyük ölçek, eğitim süresinin kaç epoch sürmesi ve başarılı bir sonuç çıkarması oldu.

Eğitim sürelerini karşılaştırdığımızda en iyi sonuçlar

Q-Learning algoritması ile gelirken en kötü sonuçlar

Deep Q-Learning algoritmasıyla geldi.

Kaynakça

[1] SARSA Reinforcement Learning(https://www.geeksforgeeks.org/sarsa-reinforcement-learning/)

[2] Q-Learning (https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/)

[3] Deep Reinforcement Learning For Maze Solving (https://www.samyzaf.com/ML/rl/qmaze.html)

[4] Reinforcement learning: Temporal-Difference, SARSA, Q-Learning & Expected SARSA in python (https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e)

[5] Qrash Course: Reinforcement Learning 101 & Deep Q Networks in 10 Minutes (https://towardsdatascience.com/qrash-course-deep-q-networks-from-the-ground-up-1bbda41d3677)

[6] Reinforcement Learning in Python (https://github.com/sichkar-valentyn/Reinforcement_Learning_in_Python)

[7] Hyperparameter Optimization using Genetic Algorithms (https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b)